ON THE MODELLING, ANALYSIS, AND MITIGATION OF DISTRIBUTED COVERT CHANNELS

ON THE MODELLING, ANALYSIS, AND MITIGATION OF DISTRIBUTED COVERT CHANNELS

BY

JASON JASKOLKA, M.A.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTING AND SOFTWARE

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Doctor of Philosophy (2015) McMaster University

(Software Engineering) Hamilton, Ontario, Canada

TITLE:	On the Modelling, Analysis, and Mitigation of Distributed Covert Channels
AUTHOR:	Jason Jaskolka M.A.Sc., (Software Engineering) McMaster University, Hamilton, Ontario, Canada
SUPERVISOR:	Dr. Ridha Khedri
NUMBER OF PAGES:	xxii, 305

To my family who has provided me with the support

and encouragement to find my path to success

Abstract

Covert channels are means of communication that allow agents in a system to transfer information in a manner that violates the system's security policy. Covert channels have been well studied in the constrained and old sense of the term where two agents are communicating through a channel while an intruder interferes to hide the transmission of a message. In an increasingly connected world where modern computer systems consist of broad and heterogeneous communication networks with many interacting agents, distributed covert channels are becoming increasingly available. For these distributed forms of covert channels, there are shortcomings in the science, mathematics, fundamental theory, and tools for risk assessment, and for proposing mechanisms and design solutions for averting these threats. Since current formal methods for specifying concurrent systems do not provide the tools needed to efficiently tackle the problem of distributed covert channels in systems of communicating agents, this thesis proposes *Communicating Concurrent Kleene Algebra* (C^2KA) which is an extension to the algebraic model of concurrent Kleene algebra (CKA) first presented by Hoare et al. C^2KA is used to capture and study the behaviour of agents, and description logic is used to capture and study the knowledge of agents. Using this representation of agents in systems of communicating agents, this thesis presents a formulation and verification approach for the necessary conditions for the existence of distributed covert channels in systems of communicating agents. In this way, this thesis

establishes a mathematical framework for the modelling, analysis, and mitigation of distributed covert channels in systems of communicating agents. This framework enhances the understanding of covert channels and provides a basis for thinking and reasoning about covert channels in new ways. This can lead to a formal foundation upon which guidelines and mechanisms for designing and implementing systems of communicating agents that are resilient to covert channels can be devised.

Acknowledgements

First and foremost, I wish to extend my deepest gratitude to my academic supervisor Dr. Ridha Khedri for his unwavering support, guidance, and encouragement throughout the development and advancement of my research. I am extremely grateful for his ability to motivate me to realise my full potential and to continually to pursue excellence in my research. His close mentoring has undoubtedly provided me with the necessary skills to achieve success in my academic career, now and in the future.

I would like to thank the members of my Supervisory Committee: Dr. Wolfram Kahl and Dr. Michael Soltys for all of their valuable, constructive, and thoughtful comments. I would also like to acknowledge all of those other individuals with whom I have interacted with throughout my doctoral studies. Their productive discussions, knowledge, and insight have significantly contributed to the progress of my research.

I want to extend my sincere thanks to the Natural Sciences and Engineering Research Council of Canada (NSERC) which has provided the financial support to undertake this research and prepare this thesis.

Last, but certainly not least, I wish to express my most genuine gratitude to my dear family for all of their love, encouragement, and support which has enlightened my path to success.

Contents

List of Tables

List of Figures

List of Symbols and Abbreviations

C^2KA	Communicating Concurrent Kleene Algebra	22
CKA	Concurrent Kleene Algebra	22
$\mathcal{ALB}$	Attributive Language with Boolean Algebras on Concepts and Roles	22
LTL	Linear-Time Temporal Logic	50
CTL	Computation Tree Logic	50
CTL^*	Extended Computation Tree Logic	50
CCS	Calculus of Communicating Systems	51
CSP	Communicating Sequential Processes	51
ACP	Algebra of Communicating Processes	51
$G \longrightarrow S$	guarded command with guard G and statement S	65
abort	abort statement	65
skip	skip statement	65
$v := E$	assignment of the evaluation of expression E to variable v	65
R	dependence relation	65

R^+	transitive closure of the dependence relation R	129
$\rightsquigarrow$	potential for direct communication	130
$\rightsquigarrow^+$	potential for communication	130
$\mathcal{N}_{\mathsf{CS}}$	knowledge component of a communication scheme	143
$\mathcal{B}_{\mathsf{CS}}$	behaviour component of a communication scheme	143
$\mathcal{N}_{\mathsf{A}} \sqcup \mathcal{N}_{\mathsf{B}}$	merged knowledge of agents A and B	154
φ	an axiom, terminological or assertional, from a knowledge base	164
Φ_C	set of confidential information	171

Chapter 1

Introduction

Modern computer systems are comprised of broad and heterogeneous communication networks with many interacting agents. They often consist of physical networks or virtual networks. They can be spread across a variety of application domains, each with their own security concerns with varying implications and priorities. The rising popularity of such expansive and complicated computer networks and systems has contributed to the increased storage and exchange of extraordinary amounts of information. Consequently, this has prompted growing concerns with respect to preserving the confidentiality of this vast amount of information. Therefore, an effort needs to be made in order to design and implement systems which conform to system requirements demanding increased security measures for sensitive information.

This chapter introduces the context and problem domain of this thesis and motivates the need for a mathematical framework for the modelling, analysis, and mitigation of distributed covert channels. More precisely, Section 1.1 gives an overview of systems of communicating agents and provides a general introduction to information security. Section 1.2 introduces covert channels, discusses the shift towards more distributed forms of covert channels, and examines the threat that they pose to the confidentiality of information, while outlining the

specific context of this thesis. Section 1.3 motivates and discusses the need for a mathematical framework for the modelling, analysis, and mitigation of distributed covert channels. Section 1.4 states the proposed research problem, outlines the objectives of this thesis, and discusses the highlights of the approach taken to achieve each objective. Section 1.5 summarises the contributions of this thesis to the enhanced understanding of distributed covert channels in systems of communicating agents. Section 1.6 notes the publications related to the work presented in this thesis. Finally, Section 1.7 outlines the structure of the remainder of this thesis.

1.1 General Context

From an architectural perspective, modern computer systems typically consist of numerous interacting agents. Ensuring that such systems are secure is a challenging task. Generally speaking, this thesis explores the area of information security and focusses on one issue related to the preservation of information confidentiality in systems of communicating agents: covert channels.

1.1.1 Systems of Communicating Agents

A *system of communicating agents* refers to any collection of interacting behavioural entities. Throughout this thesis, the terms *agent* and *communication* shall be treated in the sense used by Milner in [Mil89b] where an agent refers to any system whose behaviour consists of discrete actions and where each interaction, direct or indirect, of an agent with its neighbouring agents is called a *communication*.

In a system of communicating agents, each agent is comprised of two components: a *behaviour* and a *knowledge*. The behaviour of an agent dictates how it may communicate with other agents in the system. The knowledge of an agent determines what information it may communicate to other agents.

A system of communicating agents may consist of numerous agents with concurrent behaviours. In this way, a system of communicating agents is a *concurrent* or *distributed system* where each agent has the potential to behave independently of the other agents in the system. However, more often than not, concurrent systems feature complex interactions amongst the system agents in order to, for example, synchronise and sequence behaviour or coordinate access to shared resources. Therefore, one of the most essential aspects of concurrent systems is the notion of communication.

The communication between agents can be divided into two fundamental classes: *shared-variable communication* and *message-passing communication* [SGG07]. In shared-variable communication, agents transfer information through a shared medium (e.g., variables, memory, etc.). If two agents in a system have access to a shared environment, then they may engage in shared-variable communication by forcing changes in the shared environment and subsequently observing the forced changes. For example, suppose that two agents A and B have access to a shared variable called x. If agent A wanted to communicate the value 4 to agent B via shared-variable communication, it could do so by assigning $x := 4$. Then, when agent B reads the variable x, it can obtain the value 4. Conversely, in message-passing communication, agents transfer information explicitly through the exchange of data structures. If two agents in a system are exchanging data explicitly using the operations of a prescribed communication protocol, then they are engaged in message-passing communication. For example, suppose that two agents A and B adhere to a communication protocol offering prescribed send and $\mathsf{receive}$ operations. If agent A wanted to communicate the value 4 to agent B via message-passing communication, it could do so by using the prescribed send operation from the communication protocol. This is to say that agent A issues a command such as $\mathsf{send}\ (4)$ to agent B. Then, agent B can use the the prescribed $\mathsf{receive}$ operation from the communication protocol to receive the value 4. This is to say that agent B issues a command such as $\mathsf{receive}\ (x)$ to receive the value 4 that was sent by agent A and to store it in the variable x. In this case, the agents A and B engage in a communication *handshake*

when they synchronise their send and receive operations. The notion of handshaking is common in many process calculus models for communication and concurrency and will be discussed further in Section 2.2.

In a system of communicating agents, agents can communicate via their shared environments in the form of shared-variable communication and through their local communication channels in the form of message-passing communication. However, the system agents may also be influenced by external stimuli. When dealing with open systems, external stimuli may result from systems outside the boundaries of the system of communicating agents being considered and may have an impact on the way in which the agents in the considered system behave. For example, consider a complex system with multiple interacting subsystems. One of the subsystems may trigger an event that acts as an external stimulus that another subsystem must respond to. These events can be the result of explicit messages transmitted by a particular subsystem or by an environment variable passing a certain threshold, for example. From the perspective of behaviourism, a *stimulus* constitutes the basis for behaviour. In this way, agent behaviour can be explained without the need to consider the internal states of an agent [Wat30]. By considering this notion of the open world influence of external stimuli alongside the closed world shared-variable and message-passing communication, the communication amongst the agents in a system of communicating agents can be viewed as shown, for example, in Figure 1.1.

Consider the system of communicating agents consisting of agents A_1 and A_2 within the dotted box depicted in Figure 1.1. Agents A_1 and A_2 have a shared environment through which they can communicate. Additionally, they have some communication channels at their disposal for sending and receiving messages. However, the behaviours of agent A_1 and agent A_2 can also be influenced by the external stimuli coming from agent A_3, for example. Consider the case where agent A_1 is subjected to an external stimulus from agent A_3. Then, agent A_1 may respond to the stimulus by changing its behaviour which can affect the communication between it and agent A_2. In Figure 1.1, the system formed by agent A_5

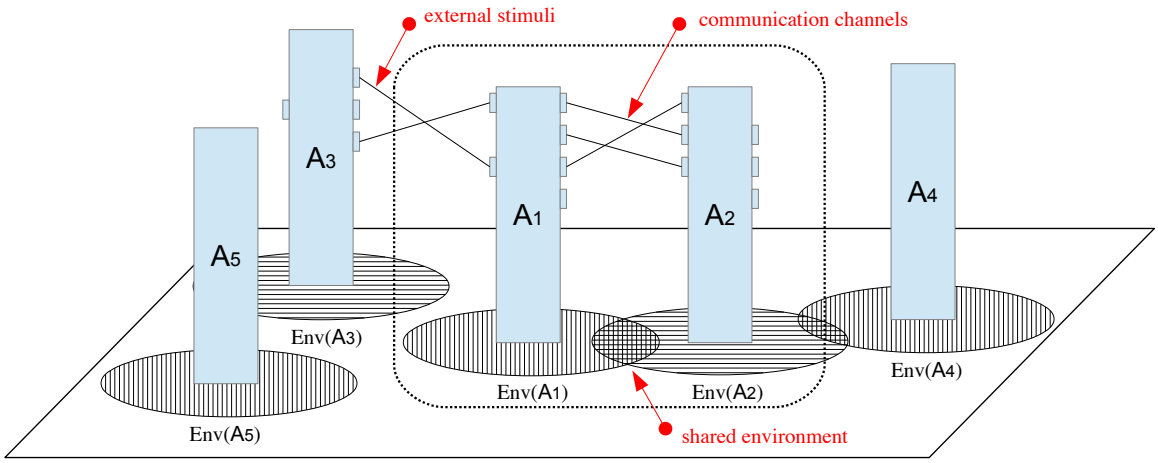

Figure 1.1: A hybrid view of agent communication

alone is a closed and independent system and does not communicate with the rest of the world neither by external stimuli nor by a shared environment.

These are the kinds of systems of communicating agents that this thesis focusses on when developing a mathematical framework for the modelling, analysis, and mitigation of distributed covert channels.

1.1.2 Information Security

Information security is rooted in three main aspects: *confidentiality*, *integrity*, and *availability* [Bis02]. The interpretation of each aspect depends on the context within which it arises. The literature defines confidentiality, integrity, and availability in many ways. This thesis adopts the definitions found in [DoTI91]. Confidentiality asserts the protection of information from unauthorised disclosure. The demand for the concealment of information arose from the increased use of computing systems in all spheres of the economy and in government, medical, and military domains. For example, military institutions restrict access to information to those individuals or groups who have a need for that information. Confidentiality is often established by implementing a "need to know" principle which grants access to information only to those who require the information. Integrity asserts the protection of information from unauthorised modification. Integrity consists of both data integrity which

refers to the trustworthiness of the content of the information, and origin integrity which refers to the trustworthiness of the source of the information. Origin integrity is more commonly known as *authentication*. The accuracy and credibility of information relies heavily on the integrity of information and is central to the proper operation of a system. Availability asserts the protection of information from unauthorised withholding. Availability is directly related to the reliability of a system since a system that is unavailable is at least as bad as no system at all. With respect to information security, availability has implications that extend to the ability of an agent to deliberately deny access to information or services by making them unavailable, thus rendering the system unusable.

It should be noted that these three aspects of information security are defined at a broad and general level for the purpose of covering a wide range of problems, and it can be argued that the list is not entirely complete [Gol11]. A more detailed discussion of information security can be established by examining several major sources of security evaluation criteria such as the U.S.A. Trusted Computer System Evaluation Criteria (TCSEC) [DoD85], the European Information Technology Security Evaluation Criteria (ITSEC) [DoTI91], the Canadian Trusted Computer Product Evaluation Criteria (CTCPEC) [Com93], and the Common Criteria for Information Technology Security Evaluation (CC) [Com09].

When discussing information security, there must exist a means of specifying what is, and what is not, a violation of security. Therefore, a security policy is required to state what is, and what is not, allowed. A *security policy* is a predicate on the knowledge of a set of agents that establishes what each agent is allowed to know and communicate [SKJ09b]. For example, consider a data store consisting of customer records for a financial institution. Suppose that each record contains information classified as *CustomerName*, *AccountNumber*, *AccountBalance*, and *AccountStatus*. A policy for this example may be that no agent should know both the *CustomerName* and *AccountNumber* of an individual customer. Subsequently, *confidential information* is defined as the information that is protected by the security policy.

In this day-and-age, computer systems face a large number of significant threats each and every day. A *threat* is considered to be any potential violation of any aspect of a system that affects its availability, and the confidentiality and integrity of its data. Threats come in many different forms and there exists a variety of security mechanisms that attempt to counter threats to the security of a system. However, more recently, a new type of threat, known as *insider threat*, has emerged and has quickly raised many new security concerns. An insider threat is defined as any agent who has special access to, or knowledge of, confidential information with the potentiality to cause harm or danger [CR06]. Insider threats can come in the form of malicious attacks or non-malicious attacks. Malicious attacks arise from insiders intentionally eavesdropping, stealing, or damaging information, or using information in a fraudulent manner whereas non-malicious attacks are typically the result of carelessness or lack of knowledge. For example, a malicious insider attack may involve an employee at a financial institution intentionally leaking confidential information such as the *CustomerName* and *AccountNumber* of individual customers to third parties which do not have the authority to access or possess that information. On the other hand, a non-malicious attack can be the result of an employee at a financial institution wrongly configuring the security access system, thus allowing it to be infiltrated by outsiders. Reported losses due to insider threats have been estimated to be in the millions of dollars annually [TK08]. Hence, insider threats are often cited as the most serious security problem in many studies since many organisations often overlook them [PHGB10]. For example, in a May 2014 study by the Ponemon Institute [Pon14], 42% of all data breaches globally, are caused by malicious attacks which include insider attacks, malware infections, phishing or social engineering attacks, and code injection. Additionally, corporate espionage is often committed by trusted insiders in such a way that many organisations do not even know it is occurring, and those who do rarely publicise it [CKC05]. For this reason, many people do not believe insider threat is a real problem. Every organisation has the potential to be a target of corporate espionage and insider threat. Because of this, an effort needs to be

made to devise effective and efficient security mechanisms and strategies with the ability to address this threat, as well as other emerging security concerns.

There are normally three strategies that are utilised when addressing security concerns: *prevention*, *detection*, and *recovery* [Bis02]. These strategies may be used separately or in some combination in order to achieve the goal of attaining a secure system (i.e., a system which has the properties of confidentiality, integrity, and availability). Prevention refers to the failure of an attempted attack on a system. For example, a data store of customer records for a financial institution requiring a password, which prevents (to a certain limit) unauthorised agents from accessing the records, is a simple prevention mechanism to ensure the confidentiality of the information in the data store. Prevention mechanisms are typically cumbersome and contribute to a reduction in overall system performance. Detection is often used when an attack cannot be prevented. Detection mechanisms accept that an attack will occur and attempt to determine if an attack is underway or has occurred. For example, a monitor watching the access to the financial institution's data store of customer records is a simple detection mechanism to detect and report any attempt to access both the *CustomerName* and *AccountNumber* of individual customer. Detection mechanisms commonly monitor various aspects of the system, looking for actions or information indicating that an attack is underway or has occurred. Recovery refers to the ability to stop an attack and to assess and repair any damage caused by that attack. For example, if an agent were to delete a customer record from the financial institution's data store, a simple recovery mechanism might be to restore the customer record from a backup. Recovery mechanisms often result in reduced system performance and are typically difficult to implement.

Information security relies on many aspects of a computer system. Achieving a secure system begins with an assessment of the nature of the threats being dealt with and countering those threats with security mechanisms, whether they be prevention, detection, or recovery mechanisms, in order to maintain confidentiality, integrity, and availability.

1.2 Specific Context

When dealing with information security, a number of concerns arise with respect to the confidentiality, integrity, and availability of information. One of the most pressing concerns to many organisations is the threat of confidential information leakage and data exfiltration. Specifically, this thesis studies the existence of distributed covert channels in systems of communicating agents that threaten the confidentiality of information by affording an ability to leak confidential information is a clandestine manner.

1.2.1 Covert Communication Channels

> He said to his friend, "If the British march
> By land or sea from the town to-night,
> Hang a lantern aloft in the belfry-arch
> Of the North-Church-tower, as a signal-light,–
> One if by land, and two if by sea;
> And I on the opposite shore will be,
> Ready to ride and spread the alarm
> Through every Middlesex village and farm,
> For the country-folk to be up and to arm."
>
> — Henry Wadsworth Longfellow, *Paul Revere's Ride* (6-14)

The art of hidden or secret communication has existed for centuries and can be concisely captured by the excerpt of Henry Wadsworth Longfellow's 1860 poem entitled "Paul Revere's Ride" given above [Lon14]. Paul Revere and his friend have shared a communication scheme which will allow for the secret communication of the way in which the British approach; "One if by land, and two if by sea". The poem embodies the fundamental essence of covert communication and gives a flavour of the kind of communication that this thesis examines.

The conception of covert channel communication is commonly attributed to Lampson's 1973 paper entitled "A Note on the Confinement Problem" [Lam73] where a covert channel was first defined as any communication channel that is neither designed nor intended to transfer information at all. Since then, several other definitions of covert channels have been proposed. For example, in [Kem83], Kemmerer defined a covert channel as a channel that uses entities not normally viewed as data objects to transfer information from one subject to another. Similarly, according to the U.S.A. National Computer Security Center [NCSC93], a covert channel is a parasitic communication channel that draws bandwidth from another channel in order to transmit information without the authorisation or knowledge of the latter channel's designer, owner, or operator. Despite that each of these are valid definitions for covert channels, this thesis adopts the definition given in the U.S.A. Department of Defense *Orange Book* [DoD85] which defines a covert channel as any communication means that allows an agent to transfer information in a manner that violates a system's security policy. This definition is chosen based on its generality and its relationship to the security policy employed by a given system. Under this definition, communication channels that may be hidden from the view of others are permitted to exist in a computer system provided they do not violate the system's security policy. Moreover, this definition is now commonly accepted [Sco07].

Covert Channel Terminologies

Over the years, covert channels have been well-studied in the constrained and old sense of the term where two agents are communicating while an intruder interferes to hide the transmission of a message. Several different terminologies related to covert communication channels have surfaced. For instance, in [Sim85], Simmons described a *subliminal channel* as a kind of channel that can be constructed in a cryptographic algorithm by one party giving up some of its ability to authenticate, without the knowledge of the host, in order to secretly transmit some bits of information. Additionally, in [Mur07], Murdoch commented

on two other terminologies that are commonly discussed: steganographic channels and side channels. A *steganographic channel* is a means of communication on an open channel, where the sender and receiver collude to prevent an observer from being able to reliably detect whether communication is taking place. A *side channel* is a communication channel which violates a security property, but where the sender unintentionally leaks information through unexpected external channels (e.g., timing, power consumption, etc.).

However, as the world becomes increasingly connected, opportunities to establish and operate covert channels are becoming more and more available. *Distributed covert channels* are covert channels that exist in distributed systems of communicating agents where multiple inter-connected agents coordinate their actions through communication. In this way, the study of distributed covert channels involves the analysis of multiple communication channels established through a variety of communication mediums, as well as the examination of the roles of intermediate agents in potential communication paths. Because of this, distributed covert channels express an evolution of the old and constrained understanding of covert channels.

While there is no standardised usage on each of these terminologies, it is clear that each of these variants are linked. For example, a subliminal channel can be viewed as special type of steganographic channel, and a steganographic channel can be used to construct a covert channel. Because of this, each of these variants is considered to be encompassed by the notion of covert channels as defined in [DoD85]. This is consistent with the current common usage and description provided by Wagner [Wag05]. Throughout the remainder of this thesis, covert channels are considered in the distributed sense discussed above. This understanding of covert channels still falls under the definition provided in [DoD85].

Defining Characteristics of Covert Channels

The defining characteristic of any covert channel is that it is typically constructed in such a way that its existence and usage is hidden from the view of third party observers. In this

way, the use of covert channels often results in third-party observers not even necessarily being aware that any communication is taking place at all. Referring back Longfellow's "Paul Revere's Ride", to any observer other than Paul Revere, whether there is one signal light or two has no particular meaning. In this way, it is only through the covert channel that is established between Paul Revere and his friend that the communication takes place. In 1984, Simmons [Sim84] provided a classical example known as the *prisoner's problem* to help understand the dynamics of covert channels. The prisoner's problem is expressed as follows:

> Two prisoners have been locked up in separate cells apart from one another. The prisoners wish to devise an escape plan. In accordance with prison rules, the prisoners are permitted to communicate with one another by sending messages, provided that the messages do not deal with escape plans. As such, all messages that are sent must pass through a warden. If the warden detects any sign of conspiracy, it will thwart the escape plans by transferring both prisoners to solitary confinement from which no prisoner has ever escaped. The prisoners were well aware of these facts, and before getting locked up, they shared a secret that they can exploit to allow for the embedding of hidden information into their seemingly innocent messages. If the prisoners are able to exchange information to coordinate their escape without raising any suspicion by the warden, then they will succeed in their plans for escape.

In essence, the prisoners establish a covert channel in order to secretly communicate their escape plans without the warden being aware that such communication is taking place. The most important aspect of the prisoners' covert channel is the sharing of a *communication scheme*. A communication scheme describes how information will be transferred and understood from the sender to the receiver. In the prisoner's problem, the communication scheme

is centred around the shared secret for concealing information about the escape. Furthermore, using the communication scheme, the prisoners communicate by sending messages through legitimate means while deliberately violating the security policy which stated that no transmitted messages contain any information regarding escape plans.

The prisoner's problem illustrates the primary objective of a covert channel, namely to conceal the fact that any communication is taking place at all. This objective differs from that of cryptography where no effort is made to hide the communication and the objective is to render the information that is being communicated unavailable to all but the intended receiver. Instead, the objective of covert channel communication aligns with that of steganography. It turns out that covert channel communication and steganography are closely related and often confused [JK11b]. In the past, it has been largely debated whether there exists a relationship between covert channels and steganography (e.g., [Ber07, BR05, GKT05, HZD05, PSCS07, PAK99, Sar06, ZAB07a]). However, in its simplest form, steganography can be perceived as a specific type of covert channel communication since, in steganography, information is embedded into some form of cover (i.e., images, audio, video, etc.) which can fundamentally be viewed as a data structure of some dimension which is transmitted from its source to its destination at some point in time or at some series of points in time, through one or more communication channels.

Examples of Covert Communication

Covert communication channels can be classified as either protocol-based, environment-based, or both [Jas10]. A *protocol-based covert channel* refers to any communication means that exploits a communication protocol to convey messages that violate a security policy. A simple illustrative example of a protocol-based covert channel is the card counting scheme given in Example 1.2.1. In this example, players A and B are communicating according to the protocol in that their behaviour is quite ordinary, but the information that they are communicating has a hidden meaning which is unknown to third party observers.

Example 1.2.1 (Card Counting)**.** *Suppose that two players* A *and* B *frequently visit the casino to play blackjack. In order to avoid significant losses and to maximise their payout each visit, player* A *and player* B *have become expert card counters. However, since card counting is highly frowned upon in the casinos, they develop a scheme to communicate the card counts to one another. The scheme consists of associations of common words, which would be part of natural conversation at a blackjack table, with particular card counts. Player* A *begins by sitting at a table and obtaining the count. Then, player* B *joins the table and player* A*, who has the current count, uses one of the agreed upon words in a sentence. For example, player* A *may utter the sentence "I wish there was more ice in my drink". This sentence contains the word "ice" which could indicate a count of* $+3$*, for example. In this way, player* B *learns the count and knows how to appropriately place its bets in order to maximise its profit.*

The word associations with the current card counts constitute the communication scheme that allows for the establishment of the covert channel. To the other players and the dealer, the conversation amongst the players at the table does not seem out of place and they are unaware that this communication of the count is taking place.

To place the idea of protocol-based covert channels in the context of computing and systems of communicating agents, consider the simple long and short message channel described in Example 1.2.2. By communicating according to the specified communication protocol, a sender is able to choose to send long and short messages in such a way as to encode a hidden message to be decoded by the receiver.

Example 1.2.2 (Long and Short Messages)**.** *According to a specified communication protocol, a sender is able to choose to send long and short messages. By carefully choosing a particular sequence of long and short messages, the sender is able to encode a hidden message which can be decoded by a receiver. For instance, the receipt of a long message may indicate a bit* 1 *and the receipt of a short message may indicate a bit* 0*. In this way,*

the sequence of long and short messages can map to a bit-string representing confidential information which should not be transmitted.

Conversely, an *environment-based covert channel* refers to any communication means that uses environmental resources, functionalities, or features, including timing information, to convey messages that violate a security policy. An example of an environment-based covert channel is one where two people, who are communicating openly in the same room, arrange the pens on a desk in a particular way so as to encode a message that is not detected by an observer as in Example 1.2.3.

Example 1.2.3 (Pen Arrangement)**.** *In an office space, two employees may utilise various arrangements of three pens on their desks in order to communicate any variety of information to one another. Depending on the orientation, colour, etc. of the pens, the employee who observes the changes in the pen arrangement can obtain a piece of information according to some previously agreed upon communication scheme.*
Any other employees who are unaware of the agreed upon communication scheme, will only see the pens on the desk (if they are even paying that close of attention) and will be unaware that any communication is taking place at all.

In the context of computing and systems of communicating agents, an example of an environment-based covert channel is one where a sender is able to modulate the timing of events in such a way that it can be detected by a receiver whereby the timing of events can be mapped to a particular encoding of information as in Example 1.2.4.

Example 1.2.4 (Memory Write Timing Modulation)**.** *Suppose that a sender has the ability to insert delays into memory write accesses on a shared memory space and that a receiver is able to observe the length of the writes from the sender on the shared memory space. By modulating the timing of the writes, the agents are able to devise a communication scheme that dictates the transmission of a bit 1 if the length of the write operation is above a predefined threshold and a bit 0 if it is below the threshold. By iterating this communication*

scheme, the sender is able to modulate the timing of its write operations so that a bit-string can be transmitted to the receiver that is observing the modulations in the timing of the write operations.

A particular feature of environment-based covert channels is that the communication of information occurs in an open system belonging to the public domain. Each user of the system has access to the information being presented, however, due to the restricted knowledge of the shared communication scheme, an observer may not be able to detect or interpret the information to gain a knowledge that the communication of information exists at all. Finally, it is important to note that in both protocol-based covert channels and environment-based covert channels, it is not the communication between agents that is in violation of a security policy, but rather, the information that is being communicated.

Necessary Conditions for the Existence of Distributed Covert Channels

The existence of covert channels has a significant impact on the confidentiality of a system of communicating agents, particularly, because they provide a mechanism for secretly divulging sensitive information to agents which are not authorised to access or obtain such information. One of the main challenges and first steps towards safeguarding systems of communicating agents against the threat of covert channels is determining the necessary conditions under which a covert channel may exist in a given system. A wide range of conditions for covert channel existence can be found in the literature. However, many existing conditions for covert channel existence are inconsistent and vary in terminology and in the way they are articulated. To address the inadequacy of existing conditions for covert channel existence, this thesis proposes a set of necessary and verifiable conditions for the existence of distributed covert channels in systems of communicating agents first presented in [JKZ12]. In a system of communicating agents, if there exists a distributed covert channel, then the following conditions are satisfied:

(1) **Potential for Communication**: If there exists an agent acting as a source of information and an agent acting as an information sink, such that the source and sink agents are different, and if there exists a pattern of communication allowing for information to transfer from the source to the sink through the synchronisation and sequencing of events, then the source and sink agents have a potential for communication.

(2) **Constraint on Communication**: If there exists confidential information in the data store of an agent, then there is a constraint on the communication of the agent and the agent can be a source of information.

The *potential for communication* condition considers the behaviour of the agents in the system and captures *how* information can be communicated by the agents in the system. In a system of communicating agents, an information flow may be established through the synchronisation of system events. The synchronisation of events may be initiated using timed events, where clocks are synchronised, or using a communication handshake, for example. Such a synchronisation of events can allow for the creation of a "pattern of communication" or simply a sequence of events that allows information to flow from one agent in the system to another. In essence, as long as there is a potential for information to flow from one agent to another, a communication channel can be established.
Conversely, the *constraint on communication* condition considers the knowledge of the agents in the system and captures *what* information can be communicated by the agents in the system. In this way, the constraint on communication condition provides that only communication channels affording an ability to violate the security policy employed by the system through the threat of confidential information leakage are considered to be covert channels. In a system of communicating agents, if there is no agent that has the knowledge of any confidential information, then regardless of any potential communication means, covert or otherwise, there is no possibility for any agent in the system to communicate any

confidential information to violate the security policy and therefore there is no possibility for the existence of a covert channel under the definition of covert channels adopted in this thesis.

Highlights of Covert Channel Research

Typically, covert channel research can be split into four categories: *explaining covert channels*, *finding covert channels*, *measuring covert channels*, and *mitigating covert channels* [Mil99]. Generally speaking, explaining covert channels refers to the development of models of systems that recognise the existence of covert channels. Finding covert channels usually involves identifying the existence of covert channels in a system. Measuring covert channels refers to estimating the bandwidth or capacity of covert channels in order to determine their seriousness. Lastly, mitigating covert channels deals with efforts to design systems in such a way that the existence of covert channels is reduced or eliminated. There are many existing techniques which cover these areas of covert channel research and they will be examined thoroughly in Chapter 2.

Specifically, this thesis examines the development of a mathematical framework for the modelling, analysis, and mitigation of distributed covert channels in systems of communicating agents. It looks to formally specify and verify the necessary conditions for the existence of distributed covert channels so that approaches for detecting confidential information leakage through the establishment and operation of distributed covert channels in systems of communicating agents can be devised. While it is nearly impossible to eliminate all covert channels in open systems due to their infinite nature, this thesis aims to aid in the development of mechanisms for designing and implementing systems of communicating agents that are more resilient to covert channels.

1.3 Motivation

Today, the protection of large amounts of sensitive information is at the forefront of security concerns. The existence of covert channels in systems of communicating agents poses a threat to the security of the system for a variety of reasons. Covert channeling techniques afford an ability to tamper with data stores in a manner that is unknown to the system and thus have the potential to compromise system integrity. Furthermore, the existence and usage of covert channels can significantly impact system availability. Since covert channels are typically based on obscure uses of system resources and functionalities, their usage can theoretically degrade system performance to a point where the system is rendered virtually unusable. This often results in serious reductions in productivity which often translates to a reduction in profit for many organisations (e.g., [HZD05]).

However, the most significant threat posed by the existence of covert channels is their ability to secretly transmit sensitive information which raises considerable confidentiality concerns. In particular, covert channels present themselves as a fruitful avenue for malicious insiders wishing to leak sensitive information to unauthorised agents. In modern organisations, the use of covert channels allows for insider threats, such as corporate espionage. In particular, these concerns have an increasing significance in large organisations wishing to maintain confidentiality regarding their secrets. This is the case with the increased use of large scale data stores and cloud computing. As organisations begin to store more and more information in the cloud, they must use prevention and detection mechanisms to protect their data from any sort of attack or leakage to ensure that the cloud is secure. Consequently, the prospect of confidential information leakage has now become among the highest fears of any executive [Sco07]. In a document published by Kaspersky Lab in 2014 [Kas14], the protection of confidential information against leakages is now the top priority for many organisations.

The existence of covert channels and their usage for breaching confidentiality policies also presents economical concerns. The cost of breaches of confidentiality is substantial and can cripple many organisations. According to a 2014 research report by IBM and the Ponemon Institute [Pon14], the average total cost of an information leak and the breach of confidential information is approximately $3.5 million. In addition to the costs associated with confidentiality breaches, the existence of covert channels allows for the transmission of information using existing systems without paying for the service provided which can also result in significant losses over time. This case is commonly exemplified when a system is infected by a *Trojan horse*. Trojan horses often masquerade as legitimate files or programs and commonly attempt to appear as helpful programs. This means that they can operate without the knowledge of the host user. The primary uses of Trojan horses include data theft in the form of retrieving passwords, credit card information, or other confidential information, as well as commandeering hosts to perform automated spamming or to distribute denial-of-service attacks.

These concerns, among others, have led the U.S.A. Department of Defense and the U.S.A. National Computer Security Center to include covert channel analysis as part of the evaluation criteria for the classification of secure systems outlined in [DoD85] and [NCSC93] respectively. However, the fact remains that each year, there are hundreds of news stories covering identity thefts and the leakage of confidential information to unauthorised parties. In recent memory, stories such as WikiLeaks, the rise of Stuxnet, the Heartbleed bug, and the breach of Apple's iCloud, for example, have underscored the need for mechanisms for protecting information confidentiality and for mitigating the existence and usage of covert channels. As many organisations depend on increasingly distributed, expansive, and complex communication networks, there are numerous possibilities for the exfiltration of sensitive information and the detection of such threats presents many challenges. Practical and foundational solutions are required to alleviate the threat of confidential information leakage in modern computer systems.

1.4 Problem Statement, Objectives, and Methodology

This section provides a statement of the problem that is addressed in this thesis. It also outlines the objectives of this thesis and discusses the methodology used to achieve those objectives.

1.4.1 Problem Statement

Currently, covert channels are poorly understood. According to the U.S.A. Department of Homeland Security [DoHS09], there are shortcomings in the science, mathematics, and fundamental theory to deal with covert channels in modern computer systems. Due to a lack of use of formal methods and the existence of ambiguous language peculiarities, protocol specifications often allow for unanticipated or unintended usage which enables the establishment of covert channels whilst adhering to the specification [SK06]. The resulting transmissions cannot be considered anomalous, leading to the difficulty of detecting such channels. Zander et al. [ZAB07b] suggest that there is a lack of formal methods for identifying covert channels during system design and that there is a necessity for a more comprehensive approach to mitigating covert channels. With the current lack of understanding of covert channels and fundamental theory to express and reason about covert channels, how can one expect to develop techniques for identifying and mitigating covert channels in computer systems, particularly at the early stages of system development?

This thesis aims to enhance the understanding of covert channel communication by developing a mathematical framework for the modelling, analysis, and mitigation of distributed covert channels in systems of communicating agents for the purpose of protecting information confidentiality in such systems. In particular, this thesis endeavours to provide a foundation upon which guidelines and mechanisms for designing and implementing systems of communicating agents that are resilient to covert channels can be devised. The achievement of these goals requires the articulation and examination of the necessary conditions for the existence of distributed covert channels presented in Section 1.2.1.

1.4.2 Objectives and Methodology

To reach a mathematical framework for the modelling, analysis, and mitigation of distributed covert channels, this thesis sets a number of objectives which are described below.

Objective 1: Modelling Systems of Communicating Agents

The first objective is to develop a suitable mathematical framework for modelling and specifying systems of communicating agents. Each agent in a system of communicating agents is comprised of a behaviour and a knowledge, and suitable mathematical structures to capture the intricacies of the behaviour and knowledge of system agents are needed. In order to accurately capture the concurrent and communicating behaviour of agents with respect to the complexities of distributed covert channels, this thesis develops a mathematical framework called *Communicating Concurrent Kleene Algebra* (C^2KA) which is an extension to the algebraic model of *concurrent Kleene algebra* (CKA) first presented by Hoare et al. [HMSW09a, HMSW09b, HMSW10, HMSW11]. The mathematical framework of C^2KA allows for the separation of communicating and concurrent behaviour in a system and its environment, and for the expression of the influence of external stimuli on the behaviours of a system of communicating agents. C^2KA also provides three levels of specification for agent behaviour giving the flexibility to select the most suitable level based on the context of the given problem. In this thesis, the knowledge of each agent in a system of communicating agents is captured using description logic. In particular, this thesis focuses on representing agent knowledge using a decidable description logic called $\mathcal{ALB}$ (Attributive Language with Boolean Algebras on Concepts and Roles) [HS00] since it provides the required expressivity for the articulation of the intricacies of covert communication schemes and reasoning on agent knowledge.

Objective 2: Formulating the Potential for Communication Condition

The second objective is to formulate and verify the potential for communication condition for the existence of distributed covert channels. The potential for communication condition states that if there exists the possibility for information to flow from one agent to another through the synchronisation and sequencing of events in a system, then the agents have the potential for communication. The potential for communication condition requires an examination of the behaviour of the agents in a given system. This thesis provides a formulation of the potential for communication condition using C^2KA. The use of the mathematical framework of C^2KA allows for the consideration of the potential for communication amongst system agents from two complementary perspectives. First, the potential for communication via external stimuli examines how stimuli generated from one agent in the system are able to influence the behaviour of other agents in the system. Second, the potential for communication via shared environments studies how communication can occur through shared events or variables and the dependencies between them. In this way, the formulation of the potential for communication condition presented in this thesis is able to account for and capture the intricacies of distributed covert channels in complex systems of communicating agents. Furthermore, it provides a more complete representation of the potential constraints and means for communication among system agents than what can be done with existing approaches.

Objective 3: Formulating the Constraint on Communication Condition

The third objective is to formulate and verify the constraint on communication condition for the existence of distributed covert channels. In a system of communicating agents, agents communicate by exchanging messages constructed from their knowledge. Therefore, the study of agent knowledge is necessary to accurately model agent communication. This thesis provides a formulation of the constraint on communication condition using description logic.

By establishing a description logic specification of the knowledge of each agent in a system of communicating agents, the constraint on communication condition can be formulated and used to identify potential sources of confidential information leakages. In this way, covert channels are considered only to be those that have the potential to allow for violations of the system security policy via confidential information leakage through some form of communication. Furthermore, an analysis of distributed covert channels that considers agent knowledge allows for the identification of those agents in a system for which it would be most beneficial to observe the communication to ensure that the security policy is being respected. The idea is that if an agent does not know any confidential information, then that agent is unable to communicate any confidential information to violate the security policy and it can be ignored in the analysis.

Objective 4: Developing Approaches for Mitigating Distributed Covert Channels

The fourth objective is to use the proposed mathematical framework to articulate and merge communication schemes into specifications of systems of communicating agents, and to develop approaches for detecting the existence of distributed covert channels used for leaking confidential information. When two agents decide to establish a covert channel, they must first devise a communication scheme that is to be shared between them. A communication scheme describes how information will be represented, transmitted, and understood from the sender to the receiver in the communication. In this way, a communication scheme is an amendment to both the knowledge and the behaviour of the sender and the receiver. Therefore, a communication scheme is comprised of two components: a knowledge base that contains all of the information required to carry out the communication of some information, including the shared representation of the information to be transmitted, and a specification of the behaviours of the sender and the receiver that will allow them to send, receive, and understand any transmitted message. In this thesis, the knowledge component

of a communication scheme is represented as a description logic knowledge base and the behaviour component of the communication scheme is represented as pre- and post-condition specifications of the behaviours of the sending agent and the receiving agent. This thesis presents an approach for merging a communication scheme into a specification of a system of communicating agents in order to augment the knowledge and behaviour of the system agents so that they may be able to establish and operate a covert channel. Such an approach is shown to be useful from two different perspectives, namely those of constructing and detecting covert channels. From the construction perspective, covert communication schemes can be devised in a very modular way and can be incorporated into the specification of a system of communicating agents. In this way, a covert channel can be mounted in the amended system of communicating agents and various existing techniques can be used in order to test the effectiveness of the constructed covert channel in terms of stealth, robustness, capacity, etc. Alternatively, communication schemes for existing covert channel techniques can be specified and merged into proposed specifications of systems of communicating agents. In this way, the execution of the system of communicating agents can be simulated and techniques such as those in [JKS11, JKS14] can be used to detect leakages of confidential information consistent with the perception of covert channel communication provided in [JK11b].

1.5 Contributions

The fulfilment of the objectives of this thesis described in Section 1.4.2 leads to the following contributions:

(1) **Necessary Conditions for the Existence of Distributed Covert Channels** (Section 1.2.1): This thesis proposes a set of necessary conditions for the existence of distributed covert channels in systems of communicating agents. Conditions for the existence of covert channels currently found in the literature vary in terminology and in

the way they are articulated. Many existing conditions are inconsistent, making formal techniques for the verification of the existence of distributed covert channels in systems of communicating agents a difficult task. The inadequacy of the varying existing conditions in the literature are discussed and consolidated to develop a set of necessary conditions for the existence of distributed covert channels. The set of necessary conditions for the existence of distributed covert channels proposed in this thesis helps to improve the current understanding of covert channels and can serve as a basis for developing effective and efficient mechanisms for mitigating distributed covert channels in systems of communicating agents. Work related to this topic has been published in [JKZ12].

(2) **Specification of Concurrent and Communicating Agent Behaviour** (Section 4.2): In order to accurately capture the concurrent and communicating behaviour of systems of communicating agents with respect to the complexities of distributed covert channels, this thesis develops Communicating Concurrent Kleene Algebra (C^2KA) which is an extension to the algebraic model of concurrent Kleene algebra (CKA) first presented by Hoare et al. [HMSW09a, HMSW09b, HMSW10, HMSW11]. C^2KA allows for the separation of communicating and concurrent behaviour in a system and its environment and for the expression of the influence of external stimuli on the behaviours of a system of agents. The strength of this work is that it allows for the inheritance of most, if not all, of the theory that has been previously developed with respect to CKA. C^2KA does not establish a new foundation, but rather builds atop well-established ones. All of this inherited theory provides the power and flexibility to specify all that can be done with existing formalisms while allowing for expansion beyond existing limitations. For example, existing formalisms such as CKA work only within closed systems, whereas C^2KA allows for the handling of open systems with the notion of external stimuli coming from outside the boundaries of the system being considered. C^2KA provides

a framework which presents a different view of communication and concurrency than what is traditionally given by existing formalisms and allows for the intricacies of distributed covert channels to be captured. Work related to this topic has been published in [JKZ13, JKZ14].

(3) **Specification of Agent Knowledge** (Section 4.3): When specifying systems of communicating agents for the purpose of covert channel analysis, in addition to both the concurrent and communicating behaviour, the knowledge of each agent needs to be considered. This thesis provides a representation of agent knowledge using the description logic $\mathcal{ALB}$ [HS00]. This representation of agent knowledge gives the power and flexibility to reason on agent knowledge at both the terminological or conceptual level and at the assertional or object level. The formal logic-based semantics of this representation of agent knowledge allows for the inference of implicitly represented knowledge from the knowledge that is explicitly contained in a knowledge base. Furthermore, with the proposed description logic representation of agent knowledge, the interplay between the knowledge and behaviour of the agents in systems of communicating agents can be studied, particularly, by looking at the evolution of agent knowledge through the execution of agent behaviour and through communication.

(4) **Formulation and Verification of the Potential for Communication Condition** (Chapter 5): One of the first steps towards uncovering whether covert channels can exist in a given system of communicating agents is to identify which agents have the potential for communication. This thesis proposes a formulation and verification approach for the potential for communication condition for the existence of distributed covert channels. The formulation is based on the mathematical framework of C^2KA. The use of C^2KA, allows for the consideration of the potential for communication amongst agents from two complementary perspectives. First, the potential for communication via external stimuli examines how stimuli generated from one agent in the system are

able to influence the behaviour of other agents in the system. Second, the potential for communication via shared environments studies how communication can occur through shared events or variables and the dependencies between them. By formulating the potential for communication condition for the existence of distributed covert channels using C^2KA, the satisfaction of the condition can be formally verified for a given system of communicating agents. The proposed formulation can serve as the basis for developing mechanisms for mitigating distributed covert channels in systems of communicating agents. This can allow for the strengthening of system designs so that they are more robust against covert channels. Work related to this topic has been published in [JK14a].

(5) **Formulation and Verification of the Constraint on Communication Condition** (Section 6.4): Often, security policies are based on the knowledge of the agents in the system for which the policy is defined. This thesis proposes a formulation and verification approach for the constraint on communication condition for the existence of distributed covert channels. The formulation is based on description logic [BMNP03]. The use of description logic provides the ability to reason about the knowledge of agents in terms of what an agent knows, or can come to know. By formulating the constraint on communication condition for the existence of distributed covert channels using description logic, the satisfaction of the condition can be formally verified for a given system of communicating agents. This leads to an analysis of covert channels that considers agent knowledge. The consideration of agent knowledge in the analysis of systems of communicating agents for the existence of distributed covert channels allows for the identification of those agents in a system for which it would be most beneficial to observe the communication to ensure that the security policy is being respected. It is acknowledged that further exploration into formulating and verifying the constraint on communication condition for the existence of distributed covert channels is needed.

(6) **Guidelines for Better Understanding Covert Channels and Attaining Systems Resilient to Their Existence and Use** (Chapter 6): The development and sharing of communication schemes is an integral part of the establishment of covert channels. The communication scheme describes how information will be interpreted and transferred from the sending agent to the receiving agent. This thesis presents a representation and classification for communication schemes. This classification provides useful information for anticipating the kind of amendments and reasoning that can be performed when a communication scheme is incorporated into a system of communicating agents. This thesis also proposes guidelines for modularly developing and merging covert communication schemes into systems of communicating agents. The proposed approach illustrates how covert channels can be designed early in the development of a system of communicating agents, particularly at the system specification level. It also provides a way of analysing the specification of a system of communicating agents for potential vulnerabilities to covert channels as it gives a way to specify and inject potential covert communication schemes into the specification of a system of communicating agents. This can lead to a better understanding of covert channels. Furthermore, this thesis presents an approach for the verification of confidential information leakage in systems of communicating agents via the establishment and operation of distributed covert channels. The proposed technique employs the framework presented in Chapter 4 for specifying systems of communicating agents and the formulations of the potential for communication condition and the constraint on communication condition described Chapter 5 and Section 6.4. An important part of this approach is the evolution of agent knowledge through the execution of concrete agent behaviours. This thesis also proposes a set of guidelines for which the evolution of agent knowledge can unfold. These contributions can serve as the basis for a comprehensive analysis of distributed covert channels in systems of communicating agents and can aid in the advancement of mechanisms for diminishing the threat of covert channels in systems

of communicating agents. As a whole, these contributions can also serve as part of a foundation for proposing guidelines for designing and implementing systems of communicating agents that are resilient to covert channels. However, it is recognised that a much deeper investigation into the ideas encompassed by this contribution is required.

1.6 Related Publications

Below is a list of the publications related to the research presented in this thesis.

1.6.1 Journal Articles

- [JK14b] J. Jaskolka and R. Khedri. Mitigating covert channels based on analysis of the potential for communication. *Theoretical Computer Science*, (40 pages), (*Submitted*, 2014).

1.6.2 Refereed Conferences

- [JKZ12] J. Jaskolka, R. Khedri, and Q. Zhang. On the necessary conditions for covert channel existence: A state-of-the-art survey. *Procedia Computer Science*, 10:458–465, August 2012. Proceedings of the 3rd International Conference on Ambient Systems, Networks and Technologies, ANT 2012.

- [JKZ14] J. Jaskolka, R. Khedri, and Q. Zhang. Endowing concurrent Kleene algebra with communication actions. In P. Höfner, P. Jipsen, W. Kahl, and M.E. Müller, editors, *Proceedings of the 14th International Conference on Relational and Algebraic Methods in Computer Science*, volume 8428 of *Lecture Notes in Computer Science*, pages 19–36. Springer International Publishing Switzerland, 2014.

- [JK14a] J. Jaskolka and R. Khedri. A formulation of the potential for communication condition using C^2KA. In A. Peron and C. Piazza, editors, *Proceedings of the 5th International Symposium on Games, Automata, Logics and Formal Verification*, volume 161 of *Electronic Proceedings in Theoretical Computer Science*, pages 161–174. Open Publishing Association, 2014.

1.6.3 Technical Reports

- [JKZ13] J. Jaskolka, R. Khedri, and Q. Zhang. Foundations of communicating concurrent Kleene algebra. Technical Report CAS-13-07-RK, McMaster University, Hamilton, ON, Canada, November 2013. Available: `http://www.cas.mcmaster.ca/cas/0template1.php?601`.

1.7 Structure of the Thesis

The remainder of this thesis is organised as follows:

Chapter 2 provides a survey of current state-of-the-art in the literature with respect to covert channels, and formalisms and representations for capturing the concurrent and communicating behaviour and the knowledge of agents in systems of communicating agents.
Chapter 3 introduces the required mathematical background including an overview of algebraic structures, concurrent Kleene algebra (CKA), Dijkstra's guarded command language, pre- and post-condition specifications, and description logic.
Chapter 4 describes how to specify systems of communicating agents by introducing the mathematical framework of Communicating Concurrent Kleene Algebra (C^2KA) for specifying the concurrent and communicating behaviour of agents. It also introduces a description logic representation of agent knowledge for specifying the knowledge of system agents.

Chapter 5 presents a formulation and verification approach of the potential for communication condition for the existence of distributed covert channels in systems of communicating agents based on C^2KA.

Chapter 6 gives an outlook for the proposed mathematical framework for the modelling, analysis, and mitigation of distributed covert channels. It presents a representation of communication schemes and guides an approach for merging communication schemes into specifications of systems of communicating agents. This chapter also proposes a set of guidelines for the evolution of agent knowledge through the execution of concrete agent behaviours and an approach for the verification of confidential information leakage in systems of communicating agents via the establishment and operation of distributed covert channels. The proposed approach for the verification of confidential information leakage also comments on a formulation and verification approach of the constraint on communication condition for the existence of distributed covert channels in systems of communicating agents based on description logic.

Chapter 7 highlights and assesses the contributions made by this thesis. This chapter also draws conclusions and suggests avenues for future work.

Chapter 2

Literature Survey

The current literature shows a wide range of existing approaches for classifying, modelling, and mitigating covert channels in systems of communicating agents. Additionally, there are a variety of ways in which systems of communicating agents can be specified using an assortment of existing formalisms for capturing different aspects of the behaviour and knowledge of agents.

This chapter surveys the literature and discusses existing work in the various areas of covert channel research. It also reviews existing approaches and formalisms for capturing the behaviour and knowledge of agents in systems of communicating agents. Specifically, Section 2.1 examines the current state of covert channel research. In particular, it looks at existing classifications of covert channels and current approaches for modelling, detecting, and preventing covert channels. Section 2.2 surveys and discusses existing formalisms for capturing agent behaviour in systems of communicating agents. Section 2.3 provides an overview and discusses existing formalisms for capturing agent knowledge in systems of communicating agents. Lastly, Section 2.4 summarises the material presented in this chapter and provides some additional concluding remarks.

2.1 Covert Communication Channels

The current state of covert channel research is quite scattered and suffers from much inconsistency and disagreement over many of the fundamentals of covert channels. This section aims to motivate the need for a better understanding of covert channels and to illustrate the inspiration for the research presented in this thesis. Specifically, this section examines the existing classifications of covert channels, as well as existing approaches for modelling, detecting, and preventing covert channels in systems of communicating agents.

2.1.1 Classification of Covert Channels

Currently, there is no generally agreed upon classification of covert channels. Traditionally, covert channels have been divided into two classifications: *storage channels* and *timing channels*. The following definitions are taken from [DoD85].

Storage Channel: A *storage channel* is a communication means involving the direct or indirect writing of object values by the sender and the direct or indirect reading of object values by the receiver.

Timing Channel: A *timing channel* is a communication means involving the sender signalling information by modulating the use of shared resources (e.g., CPU usage) over time in such a way that it is observable by the receiver.

Over the years, it has been suggested that there is no fundamental distinction between storage channels and timing channels [DoD85]. The only differences that may be drawn between them is the way in which information is encoded in each class of channel. This has led to a new classification model based on the dimension in which data is encoded, whether *time* or *space*, and also on the paradigm by which characters are encoded, either *value-based* or *transition-based*. The following classification is found in [WL05b] and consists of four types of channels:

Value-based Spatial Channel: A *value-based spatial channel* is a communication means which encodes data within a spatial container.

Transition-based Spatial Channel: A *transition-based spatial channel* is a communication means which encodes data by representing it as changes between spatial containers.

Value-based Temporal Channel: A *value-based temporal channel* is a communication means which encodes data by modulating the timing of the occurrence of events.

Transition-based Temporal Channel: A *transition-based temporal channel* is a communication means which encodes data by modulating intermediate events on the occurrence of events.

A parallel can be drawn between value-based spatial channels and traditional covert storage channels and similarly, between transition-based temporal channels and traditional covert timing channels. However, the introduction of transition-based spatial channels and value-based temporal channels gives new classes of covert channels. Transition-based spatial channels make explicit the fact that a covert storage channel can be created indirectly without requiring the sender having any control of the value of the object. Value-based temporal channels exploit the ability of a sender to alter the times at which events occur. More recently, a different classification of covert channels based on the medium used for embedding covert information was proposed [Jas10].

Protocol-based Covert Channel: A *protocol-based covert channel* is a communication means that uses the communication protocol to convey messages that violate a security policy.

Environment-based Covert Channel: An *environment-based covert channel* is a communication means that uses environmental resources, functionalities, or features to convey messages that violate a security policy.

Protocol-based covert channels typically involve the misuse of communication protocols and mechanisms in order to transmit covert information. Environment-based covert channels normally involve the use of factors in the shared environment of the channel users to transmit a covert message. A particular feature of environment-based covert channels is that the communication of information occurs in an open system which belongs to the public domain. Each user of the system has access to the information being presented, however, due to its encoding, an observer may not be able to detect or decode the information to gain any knowledge that the communication of information exists at all.

It is important to note that in both protocol-based covert channels and environment-based covert channels, it is not necessarily the communication between agents that is in violation of a security policy, but rather, the information that is being communicated. Because of this fundamental difference from previously existing classifications of covert channels, and due to its simplicity and generality, the protocol-based/environment-based classification of covert channels is adopted and considered throughout the remainder of this thesis.

2.1.2 Modelling Covert Channels

In order to develop an understanding of covert channels, researchers have proposed various models in an attempt to capture the characteristics of covert channels in computer systems. Typically, a model allows for an abstract, conceptual view of the system of interest and can aid in reasoning about the system and help in understanding the system in general.

Non-Interference Models

Some of the first models for covert channels were those based on the notion of non-interference. When discussing non-interference, a computer system is modelled as a machine with inputs and outputs, each classified as either low-level or high-level. A computer system has the non-interference property if and only if any sequence of low-level inputs will produce the same low-level outputs, regardless of what the high-level inputs are [GM82]. The study

of non-interference emerged from the need to understand why covert channels were possible [RMMG01]. Covert channel models based on non-interference are commonly attributed to Goguen and Meseguer [GM82]. Ryan et al. [RMMG01] suggest that the problem with non-interference models is a question of practicality; how can non-interference models be translated into efficient algorithms for detecting covert channels? According to [HKMY87], there is a cost for characterising security in terms of non-interference assertions due to the fact that a complicated induction is necessary to verify that a non-interference policy is satisfied.

Information Flow Models

The study of information flow is considered one of the main approaches for investigating confidentiality in computer systems [FGM03]. The aim of information flow-based models is to characterise any possible flow from high-level agents to low-level agents. One of the first, and likely most well-known, formal models allowing for the automation of the discovery of security leaks and illegal information flows is the Bell-LaPadula model [BL76].

Numerous other information flow-based models that can be used to describe covert channels have since been proposed. Denning [Den76] investigated a lattice structure derived from security classes to guarantee secure information flow in computer systems. Brewer and Nash [BN89] proposed the Chinese Wall Security Policy model where information access is not restricted by its security level. Instead, datasets are grouped into conflict of interest classes whereby an agent can have access to information of only one of these datasets. McDermid and Qi [MQ91] presented a model for evaluating secure systems and for testing for the existence of covert channels by combining static analysis and dynamic testing. Shen and Qing [SQ07] characterised the subject properties of information flows in a state machine model. Jaeger et al. [JSS07] developed a model to describe risk information flows which are information flows that may exist due to a combination of overt and covert information flows.

Shaffer et al. [SAIL07, SAIL08] presented an approach founded on the notion of interference and focussed on identifying covert channel vulnerabilities at the implementation level through static analysis techniques. Varadharajan [Var90] used a formalism extended from Petri nets to model information flow security requirements. Focardi and Gorrieri [FG94] introduced a CCS-like process calculus called *Security Process Algebra* (SPA) to verify that no information flow is possible from a high-level system to a low-level system. This model was extended in [FGM03] to a real-time setting with operators allowing for the capture of time dependent information flows in addition to logical information flows. Also, a variety of approaches for modelling and analysing information flows have been based on typing systems (e.g., [HPRW08, HRLS06, Kob05, VIS96]).

Finite State Models

According to [SC99], some covert channels can be modelled as finite state machines, while others cannot. Millen [Mil89a] proposed a number of techniques for estimating the capacity of covert channels that are modelled by finite state machines where each transition takes a constant amount of time. A variant of this model was proposed by Liu et al. [LHD10]. Gray [Gra91] used a probabilistic state machine model with a finite (but potentially very large) set of states and a set of communication channels providing the only interface to the external environment to describe computer systems. This model is similar to the synchronous state machine model described by Millen [Mil90] except that Gray's model has a probabilistic transition function rather than a non-deterministic one. Wang and Lee [WL05b] proposed a model based on finite state machines allowing for the analysis of covert channels on abstract system specifications and for articulating a set of minimum requirements for setting up a covert channel. Johnson et al. [JLY10] modelled what are coined as behaviour-based covert channels where communicating agents are represented as finite state machines. Using this model, Johnson et al. described how two agents can establish covert channels which allow their communication to bypass a monitoring agent.

Probabilistic Models

In the literature, there are a number of probabilistic models for covert channels (e.g., [TG88, KC09]). However, in [GGT10], Grusho et al. discussed some of the issues that are encountered when using statistical detection techniques and estimating covert channel capacities which quite often use probabilistic models of communication. The issue with using such models is that even small differences in the models can affect the topological structure of the sets associated with probability measures. These structural differences are often detectable using computer simulations. The results of these simulations may help or hinder the search for prohibitions of certain configurations and covert channels which cannot appear in legal communication. According to [GKT05], with enough statistical theory, covert channels based on statistics can be mitigated using such probability models and methods. However, there is an uncertainty associated with probabilistic models and the question remains, with what certainty can covert channels be mitigated using probabilistic models?

Discussion

One important observation which can be made through this examination of existing approaches for modelling covert channels is that many of the existing models suffer from restrictions which limit their applicability to devise or detect covert channels in computer systems. For instance, one such restriction is that many existing models for covert channels only consider the knowledge of the agents in a system of communicating agents by classifying them into security levels. However, many models only consider two security levels (high and low) when in reality there are many more. Based on these observations, it can be seen that a comprehensive formal model for covert channels is non-existent.

2.1.3 Detecting Covert Channels

The detection of covert channels embedded in various protocols is a relatively new area of research and recently, detection methods have begun being categorised into three categories: *signature-based*, *protocol-based*, and *behavioural-based* (e.g., [JK11a]). Signature-based detection methods involve examining the communication stream of information for specific pre-designed patterns. Protocol-based detection methods involve examining protocols by monitoring the communication stream for violations and anomalies. Lastly, behaviour-based detection methods involve constructing reference profiles with respect to a legitimate communication environment and constructing user profiles for each agent in the system. Then, the reference profiles are compared against the real-time user profiles checking for violations or illegal behaviours.

This section discusses and assesses the strengths and weaknesses of existing approaches for covert channel detection and examines why there is a need for new techniques to detect the existence of covert channels, particularly at the early stages of system development.

Non-Interference-Based Detection Techniques

A short while after the introduction of the confinement notion by Lampson [Lam73], the existence of covert channels was being examined through non-interference properties. Nagatou and Watanabe [NW06] devised a technique for detecting the use of covert channels at run-time by enforcing non-interference policies through flow control and access control mechanisms. Goguen and Meseguer [GM82] defined the existence of covert channels through non-interference properties in security policies. By this approach, security verification consists of checking that a given system model satisfies a given policy. Volpano and Smith [VS97] described non-interference through typing where a system contains interference if it cannot be correctly typed and Lowe [Low02] described non-interference using process calculi.

The notion and applicability of non-interference is questioned in [RMMG01] since the transfer of a single bit of information causes a non-interference violation. It is often the case that non-interference approaches attempt to classify data and processes of a system according to two security levels: high and low [HZD05]. However, it is rarely the case that real systems only have two security levels, which leads to a fundamental restriction of the use of non-interference properties to define the existence of covert channels outside of the theoretical realm.

Shared Resource Matrix

In 1983, Kemmerer introduced a technique for identifying the use of covert channels based on the ways in which shared system resources are used [Kem83]. The technique is appropriately called the *Shared Resource Matrix* (SRM). The shared resource matrix is motivated by the observation that covert channels require collaboration with an agent having the ability to signal or leak information to an unauthorised agent. The authorisation is normally granted on system objects which may include file locks, device busy flags, the passing of time, etc. The shared resource matrix technique involves two steps. First, an analyst needs to identify and enumerate all of the shared resources that can be referenced or modified by an agent in the system. Second, the analyst needs to carefully examine each resource to determine whether it is possible for the resource to covertly transfer information from one agent to another. A set of minimum criteria that must be satisfied for a covert channel to exist in the system is provided in [Kem83]. The verification of the minimum criteria considers both *storage channels* and *timing channels*.

The shared resource matrix technique has a number of advantages. It allows for the attributes that do not meet the preliminary criteria of being modified or referenced by an agent to be quickly discarded. It also provides a graphical design for developers in all stages of software design. However, the shared resource matrix technique is tedious and ad hoc in nature as the analyst must decipher scenarios in which the criteria might be satisfied.

It is also quite difficult to automate the technique which proves to be one of its significant weaknesses.

Covert Flow Trees

Kemmerer and Porras devised an approach, called *Covert Flow Trees* (CFTs) [KP91, PK91], for detecting covert channels in computer systems based on the representation of security violations arising from the application of fault trees. A CFT is a tree-structured representation of the sequence of operations that transfer information from one agent to another. The goal of covert flow trees is to identify operation sequences or information flows in the tree structure that support either the direct or indirect ability of an agent to detect when an attribute has been modified. This means that CFTs help to recognise when system attributes have been changed in some way by a sequence of operations.

The use of covert flow trees has the benefit of generating a comprehensive list of scenarios that could potentially support covert communication. Also, CFTs provide a graphical illustration of the routes that information travels as it is relayed from attribute to attribute, and eventually detected by the receiver. The drawback of the covert flow tree technique rests in the size of the covert flow trees that are generated and the scalability of the approach. The process of reducing and expanding the operation sequences produced by covert flow trees is currently ad hoc. Consequently, there is a risk of false positives since a complex hypothetical scenario consisting of numerous agents and system states can be generated to show the existence of a covert channel which is not possible in the system.

Information Flow Analysis

Information flow analysis approaches have been popular in developing techniques for detecting the existence of covert channels since information flows are often described through policies which aim to preserve confidentiality.

Denning and Denning [DD77] provided a technique for uncovering covert channels using the lattice-based model of information flow proposed in [Den76]. Andrews and Reitman provided an axiomatic definition for information flow in sequential programs [AR80]. Emphasis was placed on proof rules for programs containing assignment, alternation, iteration, composition, and procedure calls. The given definition closely resembles Hoare's deductive system for functional correctness [Hoa69]. The axiomatic approach analyses a program looking for information flows which violate the security policy of the system. A similar approach was taken by Sabri et al. [SKJ09b] where the satisfiability of security policies in communication protocols was verified using an amended version of Hoare logic. Murray and Lowe [ML10] examined information flow properties of object-capability patterns. In [TGC87, TGC90], Tsai et al. proposed a method for identifying potential storage channels based on the analysis of programming language data structures, code, and semantics used within the kernel to discover variable alterability and visibility. This analysis is similar to that used in the shared resource matrix technique [Kem83] for each matrix entry. Melliar-Smith and Moser [MSM91] looked at developing a technique for screening all system programs using a data dependency analysis.

Models of information flow attempt to describe all possible ways of comprising information at fine-grained views of a system. With this view, the information is recognised as low as the bit level. Techniques for conducting analysis on such models are able to successfully discover covert channels, however, they are known to be vulnerable to false positives where violations that do not really exist are flagged [Mil87]. Information flow analysis also typically occurs at later stages in software development, such as the implementation stage. For instance, typing systems are able to analyse information flows within program code, but not at an earlier stage in software development. In an attempt to detect information flow violations earlier in the software development process, Alghathbar et al. [AFW06] proposed a logic-based technique called FlowUML which aims at validating information flow policies in UML sequence diagrams.

Anomaly Detection Techniques

Anomaly detection refers to the detection of patterns in a given data set that do not conform to what is considered normal behaviour. Anomaly detection techniques are typically used in conjunction with machine learning and statistical approaches in order to provide measures to determine the existence of anomalies. Sohn et al. [SSM03] proposed an offline covert channel detection technique based on the idea of anomaly detection using a *support vector machine* (SVM) to search for anomalies in the header fields of network packets. A similar technique was proposed by Tumoian and Anikeev [TA05b, TA05a] involving the interception of all TCP traffic and a model of the initial sequence number generation. The idea is to use a neural network to create a model using only the intercepted TCP traffic without any knowledge of the data generation algorithm in an attempt to identify anomalies in the initial sequence numbers of the intercepted TCP segments. Berk et al. [BGC05a, BGC05b] investigated a methodology for detecting such channels based on a statistical measure of how well the capacity of a given channel is achieved. Jadhav and Kattimani [JK11a] proposed a statistical protocol-based detection method involving the capture of TCP segments from active network streams and analysing the covert channel vulnerable fields of TCP headers. Zhai et al. [ZLD10] proposed a method based on a TCP Markov model and the Kullback-Leibler divergence [KL51] to verify the existence of anomalies in the TCP Flags field. Zhao and Shi [ZS10] aimed to detect the existence of covert information embedded in TCP Initial Sequence Numbers using phase-space reconstruction to represent the dynamic nature of initial sequence numbers by building a four-dimensional space of the one dimensional initial sequence numbers.

Anomaly detection techniques which are based on statistics and probabilities typically have some level of uncertainty associated with the probability model used for making statistical decisions. Because of this, anomaly detection techniques for detecting covert channels very often result in an answer of “maybe” and rarely yield a definitive answer of “yes” or “no”.

Scenario-Based Detection Techniques

Hélouët et al. [HZD05, HZJ03] proposed an algorithm which characterises the presence of a covert channel in a scenario description. From a scenario description of a system, a covert channel is viewed as a game where a pair of corrupted users, sender and receiver, attempt to send information while the rest of the protocol aims to prevent the information from being communicated. The use of scenarios has several advantages. Scenarios often provide the first obtainable information regarding a system's behaviour since they are used to describe system requirements. Also, several recommendations [Com09, Com93, DoTI91, DoD85, NCSC93] request that covert channel use be documented with such models. It is often the case that model-based studies can overlook some covert channels or exhibit unrealistic scenarios. Despite that this approach can immediately offer scenarios for covert channel use, it only reveals "potential covert channels", the existence of which needs to be tested on a real implementation of the protocol.

Separability-Based Detection Techniques

A system is separable (i.e., multi-level secure) if and only if it is behaviourally equivalent to a collection of single level systems that have no interaction [Bro94]. In developing the notion of separability, Rushby developed a technique for security verification called *Proof of Separability* [Rus82]. The aim is to prove that the behaviour perceived by each agent of a shared system is indistinguishable from the behaviour that could be provided by an isolated machine for the agent's private use. Later, Jacob [Jac90] proposed a method for detecting covert channels where the idea is that if a system, where all known channels have been "cut", is separable, then there are no covert channels, otherwise, at least one covert channel exists. The shortcoming of Jacob's method is that it does not detect covert channels completely dependent on known channels.

Discussion

Many of the existing techniques for detecting the existence of covert channels in computer systems perform analyses at late stages in software development. For instance, many techniques are concerned with detecting covert channels at the implementation stage where analyses are performed on program code. It would be much more beneficial if covert channels could be detected at a much earlier stage in software development, namely the specification and design stages where few research has studied.

Also, many existing detection techniques suffer from a variety of drawbacks, such as scalability issues, applicability issues, and uncertainty, as is the case for statistical approaches. Moreover, detection techniques can be subject to false positives which stems from a lack of formality.

2.1.4 Preventing Covert Channels

Since the conception of covert channels, a wide variety of techniques for preventing their use in computer systems have been proposed. Typically, the goal of covert channel prevention is simply to reduce the usefulness of the covert channels existing in a system for communicating information. The aim is to deter malicious agents from having the desire to use covert channels.

The NRL Pump

In the early 1990's, researchers began to observe that in order to maintain reliable communication, acknowledgements were required. Shortly thereafter, it was realised that if a high-level agent passed acknowledgements directly to a low-level agent, then the higher-level agent could pass high-level information to the low-level agent by altering acknowledgement delays. This observation and realisation was the motivation for the development of the *NRL pump* (also known simply as the pump) [KM93, LMST$^+$04]. Developed by the U.S.A. Naval

Research Laboratory, the pump is a multi-level secure component (hardware or software) for preventing the use of covert timing channels with the goal of minimising information leakage from high-level agents to low-level agents without degrading average time performances.
A handful of variants of the pump have since been proposed. For example, Ogurtsov et al. [OOS+97], introduced the *quantised pump*, the *linear quantised pump*, and the *logarithmic quantised pump*. These variations differ in the interpretation of the bit that is transmitted from the high-level agent to the lower-level agent. Each variation of the pump offers their own tradeoffs in terms of system performance. Although the pump offers a formidable mitigation technique for covert channels, it suffers from scalability issues due to an inability to handle large state spaces [LMST+04].

Information Theoretic Prevention Techniques

A common approach for preventing covert channels in computer systems is to reduce their usefulness by minimising their capacity using information theoretic techniques. Information theory, which dates back to Shannon's seminal writings in 1948 [Sha48], involves the quantification of information and deals with finding the fundamental limits on reliably storing and communicating data. According to [WL05a], capacity estimation is an important part of covert channel analysis since it is often used as a measure of the severity of a covert channel. Mechanisms for computing the capacity of covert channels in computer systems are presented in a number of works (e.g., [Low02, SC99]). The idea behind capacity analysis is that if the capacity of a covert channel can be reduced to a reasonably small rate, then the channel is rendered unusable as a means of effectively transferring information. The guidelines outlined in [DoD85] and [NCSC93], state that covert channels with capacities less than one bit/second are usually considered acceptable, while capacities greater than 100 bits/second are unacceptable. A selection of approaches for reducing the capacity of covert channels are given by Martin et al. [MMA06] and Giles and Hajek [GH99, GH02].
Moskowitz and Kang [MK94] discussed some historical issues with information theoretic

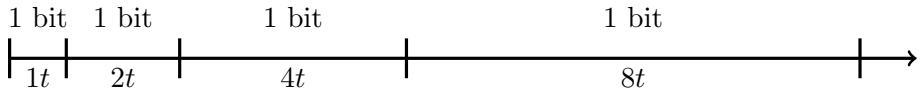

Figure 2.1: An illustration of the idea behind Moskowitz and Kang's zero capacity channel

approaches to covert channel prevention. They claimed that the use of capacity as the sole measure of insecurity is insufficient. To support this claim, Moskowitz and Kang presented an example of a channel which had a computed capacity of zero and upon which any message could be sent. In doing so, Moskowitz and Kang assumed a noiseless communication channel with two symbols. On the first use of the channel, either symbol can be sent within one clock tick, on second use, either symbol can be sent within two clock ticks, on third use, either symbol can be sent within four clock ticks, and so on such that on the n^{th} use, either symbol can be sent within 2^{n-1} clock ticks. This idea is illustrated in Figure 2.1. With such a channel, it is easy to see that by the n^{th} transmission there are 2^n different messages and that the total transmission time by the n^{th} transmission is $\sum_{i=0}^{n-1} 2^i$ clock ticks. After some time analysis (the details of which are ignored here), it can be computed that the capacity of this channel is zero. Furthermore, it can also be seen that any message can be sent on this channel with no loss of fidelity. This zero capacity channel illustrated that knowing that the capacity is zero does not ensure that the system is in a secure situation. This observation led to the introduction of the *small message criterion* which additionally considered the message length, fidelity, and time frame for transmission for determining the tolerable level of covert communication in the system.

Still, tolerances such as those provided by capacity estimation and the small massage criterion, can result in a greater threat than what is calculated using information theoretic techniques. This is particularly apparent when low capacity channels, which are typically considered innocuous, are used as the mechanism for initiating and correctly sequencing the communication between the sending agent and the receiving agent. This aligns with the

existence conditions of both storage and timing channels in Kemmerer's shared resource matrix technique [Kem83]. The use of these low capacity channels aids in the establishment of higher capacity covert channels. For example, in a distributed system, it is possible to establish a covert communication scheme involving the use of multiple low capacity channels such that their combination results in a much higher capacity channel which can be used to leak confidential information. In order to mitigate this threat, it should be noted that it is not uncommon for the capacity threshold to be zero, meaning that any covert channel with a capacity greater than zero is considered to be a threat [MK94].

Fuzzy Time

In an effort to mitigate the use of covert timing channels, Hu [Hu91] developed the notion of *fuzzy time*. Fuzzy time is a covert channel countermeasure that aims to reduce the capacity of covert timing channels by making all clocks available to a process noisy. Covert timing channels require two clocks: a reference clock (usually a system clock) to measure the absolute time and another clock to be modulated by the sending agent and observed by the receiving agent [Wra91]. The fuzzy time technique is based on this idea and attempts to isolate a process from all precise timing information by randomly altering the timings of I/O operations to diminish the precision and accuracy of system clocks. In [Tro93], Trostle explored additional countermeasures based on the notion of fuzzy time to help mitigate covert timing channels. Such countermeasures included ways to redesign the system process scheduler based on security levels, similar to the lattice scheduler proposed by Hu [Hu92].

Discussion

Existing covert channel prevention techniques simply attempt to reduce the capacity of covert channels to a rate which makes the use of covert channels inefficient. It should be noted that the primary target of a large proportion of existing prevention techniques is covert timing channels. The issues that are noted with the existing prevention techniques

lie in the scalability of the techniques and the impact they pose on the overall performance of the system. If covert channels can be prevented at earlier stages in system development, then the performance repercussions can be reduced.

2.2 Formalisms for Capturing Agent Behaviour

When discussing systems of communicating agents, one of the most important aspects of each agent is their concurrent and communicating behaviour. Concurrency and communication have been popular topics of study in computer science and software engineering since the 1960's. Over the course of the past 50 years, there have been many proposed models for concurrency. Each of these models attempts to capture the concurrent and communicating behaviour of agents in a system. This section provides an overview and discussion of existing formalisms for capturing the concurrent and communicating behaviour of agents in a system of communicating agents.

Systems interact with other systems resulting in the development of patterns of stimuli-response relationships. Therefore, models for concurrency are commonly constructed upon the assumption of uninterruptible system execution or atomic events. Models for concurrency differ in terms of how they capture this notion. A coarse-grained classification categorises models for concurrency as either *state-based* models or *event-based* models [CS96]. State-based models describe the behaviour of a system in terms of the properties of its states. Typical state-based approaches for capturing the concurrent and communicating behaviour of agents consist of representing system properties as formulae of temporal logics, for example, such as linear-time temporal logic (LTL) [Pnu77], computation tree logic (CTL) [CE82] and its extension (CTL*) [EH86], and model-checking the state space of the system against them. Conversely, event-based models represent systems via structures consisting of atomic events. There is an extensive variety of examples of event-based models for concurrency including labelled transition systems [Kel76], Petri nets [Pet62], process

calculi (e.g., calculus of communicating systems (CCS) [Mil80], communicating sequential processes (CSP) [Hoa78a], algebra of communicating processes (ACP) [BK84], and π-calculus [MPW92]), Hoare traces [Hoa85], Mazurkiewicz traces [Maz87], synchronisation trees [Mil80], pomsets [Pra86], event structures [Win87], and action algebras [Koz93, LG98, Pra91].

Existing state-based and event-based formalisms for communication and concurrency such as temporal logics, labelled transition systems, Petri nets, and process calculi are primarily interested in modelling the behaviour of a system either in terms of the properties of its states or in terms of the observability of events. However, many existing formalisms (e.g., [CE82, EH86, Hoa85, Kel76, Maz87, Pet62, Pra91, Pnu77, Win87]) do not deal directly with the notion of communication or they have a very convoluted or restricted view of the interactions among system agents. Perhaps the best formalisms for capturing the communication between system agents are the process calculi (e.g., [BK84, Hoa78a, Mil80, MPW92]). In an attempt to capture the communicating behaviour of agents, process calculi provide tools for describing the interactions, communications, and synchronisations between agents in a system. However, process calculi typically only consider the representation of interactions between agents as message-passing communication, and often neglect shared-variable communication. Additionally, one of the primary notions in many process calculi is the idea of conjugate actions and handshaking. The idea is that when two conjugate actions execute in parallel, a handshake occurs and the result is a silent, unobservable communication action, traditionally denoted by τ. However, it is not always the case that each and every event in a system has a primitive conjugate action and that each communication is a silent action.

Moreover, existing state-based and event-based formalisms do not directly, if at all, provide a hybrid model of communication and concurrency which encompass the characteristics of both state-based and event-based models. However, recently, Hoare et al. [HMSW09a, HMSW09b, HMSW10, HMSW11] proposed a formalism for modelling concurrency called

concurrent Kleene algebra (CKA). CKA extends the algebraic framework provided by Kleene algebra by offering, aside from choice and finite iteration, operators for sequential and concurrent composition. Concurrent Kleene algebra is perhaps the closest formalism to providing such a hybrid model capturing the elements of both state-based and event-based models. While CKA can be perceived as a hybrid theory for concurrency, the same cannot be said for communication. As presented in [HMSW09a, HMSW09b, HMSW10, HMSW11], communication in CKA is not directly captured. Variables and communication channels are modelled as sets of traces. Communication can be perceived only when programs are given in terms of the dependencies of shared events [HW11]. One needs to instantiate the low-level model of programs and traces for CKA in order to define any sort of communication. Furthermore, neither CKA nor any other existing formalism for capturing the concurrent and communicating behaviour of agents deals directly with describing how the behaviours of agents in a system are influenced by external stimuli, which is an important aspect that ought to be considered when capturing the concurrent and communicating behaviour of agents in systems of communicating agents.

2.3 Formalisms for Capturing Agent Knowledge

Epistemology, derived from Greek meaning "knowledge science", is rooted in philosophy, theoretical computer science, artificial intelligence, economics, and linguistics. In systems of communicating agents, each agent has a knowledge of various aspects of the system to which it belongs and the world in which it resides. In the literature, there are a number of existing formalisms for capturing agent knowledge.

Epistemic Logic

Perhaps one of the most popular existing formalisms for capturing and reasoning about agent knowledge is epistemic logic [MvdH04]. Epistemic logic is a modal logic concerned

with reasoning about knowledge. The relevance of epistemic logic is realised in the formal description of the knowledge of agents in distributed and intelligent systems in order to specify or verify protocols, represent knowledge, and formalise reasoning methods. In an epistemic logic, let $\mathcal{P}$ be a set of propositional constants such that $\mathcal{P} = \{p_n \mid n \in \mathbb{N}\}$. Additionally, let $\mathcal{A}$ be a set of m agents. The set $\mathcal{L}_{\mathcal{K}}^{m}(\mathcal{P})$ of epistemic formulas, $\varphi, \psi, \ldots$ over $\mathcal{A}$ is the smallest set closed under the following rules:

- If $p \in \mathcal{P}$ then $p \in \mathcal{L}_{\mathcal{K}}^{m}(\mathcal{P})$
- If $\varphi, \psi \in \mathcal{L}_{\mathcal{K}}^{m}(\mathcal{P})$ then $(\varphi \ \wedge \ \psi), \neg\varphi \in \mathcal{L}_{\mathcal{K}}^{m}(\mathcal{P})$
- If $\varphi \in \mathcal{L}_{\mathcal{K}}^{m}(\mathcal{P})$ then $\mathrm{K}_i\,\varphi \in \mathcal{L}_{\mathcal{K}}^{m}(\mathcal{P})$, for all $i \in \mathcal{A}$

In an epistemic logic, the formula $\mathrm{K}_i\,\varphi$ is read as "agent i knows that φ." As with many other modal logics, the semantics of epistemic logic are related to Kripke structures [Kri63]. Kripke structures are simple abstract machines which attempt to capture the idea of a computing machine, without adding unnecessary complexities. A *Kripke structure* $\mathbb{M}$ is a tuple $(S, \pi, R_1, \ldots R_m)$ where S is a non-empty set of *states*, $\pi : S \to (P \to \mathbb{B})$ is a truth assignment to the propositional atoms per state, and $R_i \subseteq S \times S \quad (i = 1, \ldots, m)$ are the so-called *accessibility relations*. Essentially, Kripke structures are graphs whose nodes represent reachable system states and whose edges represent state transitions. There also exists a labelling function which maps each node to a set of properties that hold in the corresponding state. A *Kripke world* $w = (\mathbb{M}, s)$ consists of a Kripke structure $\mathbb{M}$ together with a distinguished state $s \in S$. The notion of Kripke worlds provides an interpretation for the accessibility relations in a Kripke structure. The interpretation of $(s, t) \in R_i$ is that in world $(\mathbb{M}, s)$, agent i considers world $(\mathbb{M}, t)$ as a possible world on the basis of its knowledge. The satisfiability relation of Kripke worlds is defined with regard to epistemic formulae.

An extension to epistemic logic called *dynamic epistemic logic* (DEL) was proposed by van Ditmarsch et al. [vDvdHK03, vDvdHK07]. In dynamic epistemic logic, there are additionally actions that can cause changes in a Kripke structure. More specifically, dynamic epistemic logic has a notion of *action algebras* or *action structures* which represent a number of dynamic modal operators that may change the knowledge of the agents involved. In essence, dynamic epistemic logic adds dynamic modal operators (referred to as *knowledge actions*, *epistemic actions* or simply *actions*) to multi-agent epistemic logic for a set of agents and a set of atomic propositions. Actions may change the knowledge of the agents involved, but they do not change facts.

Epistemic logics provide a basis for modelling the knowledge of agents in systems of communicating agents and additionally offer extensions to handle additional notions of belief, common knowledge, temporal constraints, groups of agents, etc. Specifically, the extension to dynamic epistemic logic allows for a description of how knowledge changes as the result of the execution of actions in a system. Such logics are best suited for artificial intelligence applications where decisions are made based on the notion of possible or accessible worlds. However, epistemic logics require that all information be represented as proposition variables, which cannot always be done easily in many practical applications, particularly those involving distributed covert channels in systems of communicating agents. Also, they require each agent in the system to "speak" a common language which may not necessarily be the case in systems of communicating agents. Lastly, epistemic logics provide a static view of the system that is fixed at the beginning and once a possible world is no longer deemed possible, it is removed and cannot be recovered which provides many restrictions when reasoning about systems of communicating agents for the existence and usage of distributed covert channels.

Logic of Communication Graphs

In [PP05, PP07], Pacuit and Parikh presented the *logic of communication graphs*. The logic of communication graphs is a way to reason about the knowledge of agents in a system which are restricted to communicating according to some given communication graph. This means that each agent may only communicate with another agent if it respects the prescribed communication graph for the system. A multi-agent epistemic logic with communication modalities is developed where agents are assumed to communicate according to some fixed communication graph. Any graph $\mathcal{G} = (\mathcal{A}, E)$ is called a *communication graph* where $\mathcal{A}$ is a set of system agents and the intended interpretation of $(\mathsf{A}, \mathsf{B}) \in E$ is that agent A and agent B can communicate. Given a communication graph $\mathcal{G} = (\mathcal{A}, E)$, a sequence of communications (agent A learns a fact from agent B who learns a fact from agent C, etc.) *respects the communication graph* if agents only communicate with their immediate neighbouring agents in $\mathcal{G}$. The approach given in [PP05, PP07] rests on two fundamental assumptions, namely that all the agents share a common language and that the agents make available all possible pieces of (purely propositional) information which they know and which are expressible in the common language. In this approach, the situation in which agent A learns some ground fact φ (directly) from agent B is represented by a tuple $(\mathsf{A}, \mathsf{B}, \varphi)$ and is called a *communication event*. Using this idea of communication events, a history of events is captured in this approach and used to specify the communications that are legal in the sense that they respect the given communication graph. For an event $(\mathsf{A}, \mathsf{B}, \varphi)$ to take place after a history H, it must be the case that after H, B knows φ. Clearly agent A cannot learn from agent B something which agent B did not know. Whether a history is justified depends not only on the initial valuation, but also on the set of communications that have taken place prior to each communication in the history. With the logic of communication graphs, given a communication graph and a corresponding model with an initial state and history, the legality of the history and the satisfiability of a formula

with respect to the given model is defined by mutual recursion. This is due from the fact that the legality of a history can only be defined in terms of the knowledge of agents and the knowledge of agents, in turn, requires quantification over legal histories. While the logic of communication graphs provides a means for reasoning about the knowledge of agents in a system of communicating agents while conforming to the potential for communication among agents, it again provides a static view of the system and restricts agents to "speak" a common language. Furthermore, the notions of histories and communication graphs contribute to a significant amount of overhead with the use of this formalism for capturing agent knowledge in systems of communicating agents.

Information Algebra

While many existing approaches for representing agent knowledge are logical in flavour, there is a formalism that has a more algebraic flavour called information algebra. *Information Algebra* [KS07] is a mathematical structure that involves pieces of information Φ and a lattice of frames D for classifying pieces of information. Information algebra provides a different way of representing agent knowledge by allowing for each agent in a given system to classify information into different frames. This allows agents in the system to "speak" different languages. It also provides a representation of knowledge as an information store with operations for inserting, updating, and removing information, rather than a set of formulae denoting what an agent considers to be possible in a system as is the case with many logic-based formalisms for agent knowledge. However, information algebra comes with a complex representation with a large algebraic structure. It is also unclear how this formalism can be extended to capture some of the intricacies of distributed covert channels in systems of communicating agents.

Description Logic

As an alternative to the above mentioned formalisms for capturing agent knowledge in systems of communicating agents, this thesis uses description logic [BMNP03] for modelling agent knowledge. Prior to the 1980's, description logic languages were known as concept languages. Description logic was first introduced to provide a formal logic-based semantics which was not available with frames and semantic networks. There exist many variants of description logic languages. The base language is known as $\mathcal{AL}$ (Attributive Language) and allows for negation of atomic concepts and intersection of concepts, as well as universal restrictions and limited existential quantifications. All other languages are extensions to the language $\mathcal{AL}$. For example, a popular extension is the language $\mathcal{ALC}$ [BMNP03] which additionally allows for negation on any concept. Other extensions include the language $\mathcal{DLR}$ [CDL02] which allows for n-ary roles and $\mathcal{ALCK}$ [DLN$^+$92] which sees the addition of epistemic operators commonly found in epistemic logics to description logic. There are many more variations and extensions and an exhaustive list will not be provided in this thesis. However, a more detailed discussion of description logic, specifically the decidable description logic variant $\mathcal{ALB}$ [HS00], is provided in Section 3.5. The rationale for modelling agent knowledge using the description logic $\mathcal{ALB}$ lies in the needed expressivity required to reason about agent knowledge at both the terminological or conceptual level and at the assertional or object level which is essential when considering the complexities of distributed covert channels and covert communication schemes. Furthermore, the decidability of the description logic is important since it means that it is possible to determine whether an agent knows a particular fact (i.e., it can be determined if an arbitrary description logic formula is a theorem). This is a central feature that is needed in order to develop approaches for automating reasoning on agent knowledge.

2.4 Conclusion

This chapter highlights the fact that there exists a lack of understanding when it comes to dealing with covert channels in systems of communicating agents. For example, there is much confusion surrounding the classification of covert channels. Also, while there exists many ways in which covert channels can be modelled in systems of communicating agents, there does not exist any mathematical framework which can capture the common properties of covert channels. Similarly, there are many existing covert channel countermeasures focussed on detecting or preventing the existence and usage of covert channels in systems of communicating agents, each with its own strengths and weaknesses. However, many of these approaches exist at the implementation level and do not address the issue of covert channels early in the development of a system of communicating agents. This thesis aims to enhance the understanding of covert channels by developing a mathematical framework for the modelling, analysis, and mitigation of distributed covert channels. It looks to provide a foundation upon which guidelines and mechanisms for designing and implementing systems of communicating agents that are resilient to covert channels can be devised.

Additionally, there exists many formalisms which aim to capture the behaviour and knowledge of agents in systems of communicating agents. However, there is no formalism that directly captures the concurrent and communicating behaviour of agents in a system of communicating agents, particularly in terms of the influence of external stimuli and the intricacies of distributed covert channels. To address this gap in the literature, this thesis proposes a new way of formalising the concurrent and communicating behaviour of agents and capturing the knowledge of each agent in a system of communicating agents that is different than what is done with current formalisms for agent behaviour and knowledge. This new way of looking at systems of communicating agents will allow for the analysis of such systems with respect to the existence and usage of distributed covert channels for leaking confidential information.

Chapter 3

Mathematical Background

This chapter provides the mathematical background required for the remainder of this thesis. Particularly, Section 3.1 gives the mathematical preliminaries of the algebraic structures used in this thesis. Section 3.2 introduces concurrent Kleene algebra which is integral in specifying agent behaviour in systems of communicating agents. Section 3.3 summarises Dijkstra's guarded command language which is used when specifying of agent behaviour. Section 3.4 details the notion of pre- and post-condition specifications and gives a brief summary of Hoare triples. Section 3.5 gives an overview of description logic, specifically the language $\mathcal{ALB}$, which is used for specifying agent knowledge in systems of communicating agents. Lastly, Section 3.6 summarises what has been presented in this chapter and gives some closing remarks.

3.1 Algebraic Structures

An *algebraic structure* refers to some arbitrary set S called the *carrier set* with one or more finitary operations defined on S [Coh81]. The most common operations for algebraic structures are *unary operations* which take one input value to produce an output and *binary operations* which take two input values to produce an output.

Several common properties of binary operations in algebraic structures are given in Definition 3.1.1.

Definition 3.1.1 (Properties of Binary Operations — e.g., [Hun74])**.** *Let S be a set, $+ : S \times S \to S$ and $\cdot : S \times S \to S$ be two binary operations, and $\leq$ be a partial order. For all $a, b, c \in S$, it is said that:*

$$
\begin{aligned}
\cdot \textit{ right-distributes over } + &\iff (a + b) \cdot c = a \cdot c + b \cdot c \\
\cdot \textit{ left-distributes over } + &\iff a \cdot (b + c) = a \cdot b + a \cdot c \\
+ \textit{ is associative} &\iff a + (b + c) = (a + b) + c \\
+ \textit{ is commutative} &\iff a + b = b + a \\
+ \textit{ is idempotent} &\iff a + a = a \\
0 \textit{ is the identity of } + &\iff 0 + a = a = a + 0 \\
0 \textit{ annihilates } S \textit{ with respect to } \cdot &\iff 0 \cdot a = 0 = a \cdot 0 \\
+ \textit{ is right-isotone with respect to } \leq &\iff (a \leq b \implies a + c \leq b + c) \\
+ \textit{ is left-isotone with respect to } \leq &\iff (a \leq b \implies c + a \leq c + b)
\end{aligned}
$$

■

Several simple algebraic structures are presented below. These structures provide the building blocks for larger and more complex structures that are introduced later in this thesis.

Definition 3.1.2 (Semigroup — e.g., [Hun74])**.** *A* semigroup *is a mathematical structure $(S, \cdot)$ consisting of a non-empty set S, together with an associative binary operation $\cdot$. A semigroup is called* commutative *if $\cdot$ is commutative, and a semigroup is called* idempotent *if $\cdot$ is idempotent.* ■

Definition 3.1.3 (Monoid — e.g., [Hun74])**.** *A* monoid *is a mathematical structure $(S, \cdot, 1)$ consisting of a semigroup $(S, \cdot)$ and a distinguished constant 1 which is the identity with respect to $\cdot$. A monoid is called* commutative *if $\cdot$ is commutative, and a monoid is called* idempotent *if $\cdot$ is idempotent.* ■

Definition 3.1.4 (Semiring — e.g., [HW93])**.** *A* semiring *is a mathematical structure* $\big(S, +, \cdot, 0, 1\big)$ *where* $\big(S, +, 0\big)$ *is a commutative monoid and* $\big(S, \cdot, 1\big)$ *is a monoid such that operator* $\cdot$ *distributes over operator* $+$*. Element* 0 *is said to be* multiplicatively absorbing *if it annihilates* S *with respect to* $\cdot$*. A semiring is* idempotent *if operator* $+$ *is idempotent. Every idempotent semiring has a natural partial order* $\leq$ *on* S *defined by* $a \leq b \iff a + b = b$*. Operators* $+$ *and* $\cdot$ *are isotone on both the left and the right with respect to* $\leq$*.* ■

Definition 3.1.5 (Quantale — e.g., [HMSW09a])**.** *An idempotent semiring* $\big(S, +, \cdot, 0, 1\big)$ *is called a* quantale *if the natural order induces a complete lattice and the operation* $\cdot$ *distributes over arbitrary suprema.* ■

A *Kleene algebra* extends the notion of idempotent semirings with the addition of a unary operator commonly interpreted as finite iteration. Kleene algebras are most commonly known for generalising the operations of regular expressions.

Definition 3.1.6 (Kleene Algebra — e.g., [Koz97])**.** *A* Kleene algebra *is a mathematical structure* $\big(K, +, \cdot, ^*, 0, 1\big)$ *where* $\big(K, +, \cdot, 0, 1\big)$ *is an idempotent semiring with a multiplicatively absorbing* 0 *and identity* 1 *and where the following axioms are satisfied for all* $a, b, c \in K$*:*

(1) $1 + a \cdot a^* = a^*$

(2) $1 + a^* \cdot a = a^*$

(3) $b + a \cdot c \leq c \implies a^* \cdot b \leq c$

(4) $b + c \cdot a \leq c \implies b \cdot a^* \leq c$

■

Some important notions required for the proposed framework for specifying the concurrent and communicating behaviour of agents in systems of communicating agents are actions of monoids on sets and semimodules.

Definition 3.1.7 (Left $\mathcal{S}$-act — e.g., [KKM00])**.** *Let $\mathcal{S} = \big(S, \cdot, 1\big)$ be a monoid and K be a non-empty set. Then, K is called a* left $\mathcal{S}$-act*, denoted ${}_{\mathcal{S}}K$, if there exists a mapping $S \times K \to K$ denoted by juxtaposition such that for all $s, t \in S$ and $a \in K$:*

(1) $(s \cdot t)a = s(ta)$ (2) $1a = a$

■

A right $\mathcal{S}$-act, denoted $K_{\mathcal{S}}$, can be defined analogously[1].

Definition 3.1.8 (Left $\mathcal{S}$-semimodule — e.g., [HW93])**.** *Let $\mathcal{S} = \big(S, +, \cdot, 0_{\mathcal{S}}, 1\big)$ be a semiring and $\mathcal{K} = \big(K, \oplus, 0_{\mathcal{K}}\big)$ be a commutative monoid. Then, $\big({}_{\mathcal{S}}K, \oplus\big)$ is called a* left $\mathcal{S}$-semimodule *if there exists a mapping $S \times K \to K$ denoted by juxtaposition such that for all $s, t \in S$ and $a, b \in K$:*

(1) $s(a \oplus b) = sa \oplus sb$

(2) $(s + t)a = sa \oplus sb$

(3) $(s \cdot t)a = s(ta)$

(4) $\big({}_{\mathcal{S}}K, \oplus\big)$ *is called* unitary *if it also satisfies* $1a = a$

(5) $\big({}_{\mathcal{S}}K, \oplus\big)$ *is called* zero-preserving *if it also satisfies* $0_{\mathcal{S}}a = 0_{\mathcal{K}}$

■

A right $\mathcal{S}$-semimodule, denoted $\big(K_{\mathcal{S}}, \oplus\big)$, can be defined analogously. From Definition 3.1.8, it is easy to see that each unitary left $\mathcal{S}$-semimodule $\big({}_{\mathcal{S}}K, \oplus\big)$ has an embedded left $\mathcal{S}$-act ${}_{\mathcal{S}}K$ with respect to the monoid $\big(S, \cdot, 1\big)$.

[1] Every left act of a monoid $\mathcal{S} = \big(S, \cdot, 1\big)$ can be interpreted as a right act over the opposite monoid $\mathcal{S}^{\text{op}} = \big(S, \cdot^{\text{op}}, 1\big)$, which has the same elements and identity as $\mathcal{S}$, but where multiplication if defined by $s \cdot^{\text{op}} t = t \cdot s$. In this way, the two notions are essentially equivalent [KKM00].

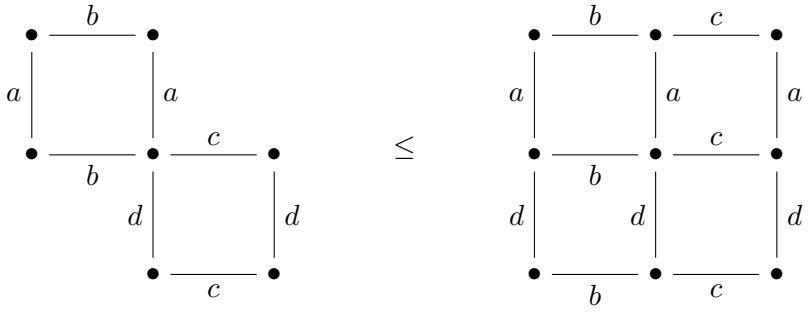

Figure 3.1: An illustration of the exchange axiom

3.2 Concurrent Kleene Algebra

Concurrent Kleene algebra (CKA) is an algebraic framework developed by Hoare et al. [HMSW09a, HMSW09b, HMSW10, HMSW11]. CKA extends Kleene algebra by offering operators for sequential and concurrent composition, along with those for choice and finite iteration. The operators for sequential and concurrent composition are related by an inequational form of the exchange axiom.

Definition 3.2.1 (Concurrent Kleene Algebra — e.g., [HMSW09a])**.** *A* concurrent Kleene algebra (CKA) *is a mathematical structure* $\big(K, +, *, ;, ^{\circledast}, ^{\odot}, 0, 1\big)$ *such that* $\big(K, +, *, ^{\circledast}, 0, 1\big)$ *and* $\big(K, +, ;, ^{\odot}, 0, 1\big)$ *are Kleene algebras linked by the* exchange axiom *given by* $(a * b) ; (c * d) \leq (b ; c) * (a ; d)$. ■

In a CKA, the operator $+$ is interpreted as a non-deterministic choice, the operator $;$ is interpreted as sequential composition, and the operator $*$ is interpreted as parallel composition. The operators $^{\circledast}$ and $^{\odot}$ are interpreted as finite parallel iteration and finite sequential iteration. The element 0 is the identity with respect to $+$ and annihilates with respect to $;$ and $*$ and the element 1 is the identity with respect to $;$ and $*$. Intuitively, the exchange axiom expresses a divide-and-conquer mechanism for how parallel composition may be sequentially implemented on a machine. This intuition is illustrated in Figure 3.1.

A selection of laws for CKA which are needed for the remainder of this thesis are given in Proposition 3.2.1. Each of the results presented in Proposition 3.2.1 are consequences of the exchange axiom.

Proposition 3.2.1 ([HMSW09a])**.** *For all* $a, b, c, d \in K$*,*

(1) $a * b = b * a$

(2) $(a * b) ; (c * d) \leq (a ; c) * (b ; d)$

(3) $a ; b \leq a * b$

(4) $(a * b) ; c \leq a * (b ; c)$

(5) $a ; (b * c) \leq (a ; b) * c$

An additional useful law is given in Proposition 3.2.2.

Proposition 3.2.2. *For all* $a \in K$*,* $a^{\circledcirc} \leq a^{\circledast}$*.*

Proof. The proof involves the application of Definition 3.1.6(3), Definition 3.1.6(1), and Proposition 3.2.1(3). The detailed proof is given in Appendix A.1. □

An important aspect of CKA is the notion of dependence and the idea of aggregation algebras.

Definition 3.2.2 (Aggregation Algebra — e.g., [HMSW09b])**.** *An* aggregation algebra *is a mathematical structure* $(S, +)$ *consisting of a set* S*, together with a binary operation* $+$*.* ■

In general, aggregation algebras are considered to be absolutely free in the sense that the operator + need not satisfy any laws.

Hoare et al. provided a definition for an independence relation in [HMSW09b]. By taking the negation of the definition of the independence relation, the following definition is obtained for a dependence relation.

Definition 3.2.3 (Dependence Relation)**.** *A* dependence relation *on an aggregation algebra* $\big(K, +\big)$ *is a bilinear relation* $\mathrm{R} \subseteq K \times K$*, i.e.,*

$$(a + b)\,\mathrm{R}\,c \iff (a\,\mathrm{R}\,c \ \vee \ b\,\mathrm{R}\,c)$$
$$a\,\mathrm{R}\,(b + c) \iff (a\,\mathrm{R}\,b \ \vee \ a\,\mathrm{R}\,c)$$

If $a\,\mathrm{R}\,b$*, it is said that* b *depends on* a*.* ■

3.3 Dijkstra's Guarded Command Language

In [Dij75], Dijkstra provided a language that compactly combines programming concepts called the *guarded command language*. The guarded command language is most often used when performing program verification and proving program correctness using Hoare logic. The most important element of the guarded command language is the guarded command. A *guarded command* takes the form $G \longrightarrow S$ where G is a boolean expression called the *guard* and S is a *statement* (or a sequence of statements). A guarded command is interpreted by evaluating the guard G when it is encountered. If G evaluates to true, then the statement S is executed. Otherwise, if G evaluates to false, the statement S is not executed. Note that a guarded command by itself is not a statement. Instead, it is component from which statements can be constructed.

There are two very simple but important statements in the guarded command language called abort and skip. The abort statement is the undefined instruction which is interpreted as "do anything". On the other hand, the skip statement is the empty instruction which is interpreted as "do nothing".

Another common statement is the assignment statement. The assignment statement allows values to be assigned to variables and takes the form: $v := E$ where v is a program variable and E is an expression to be evaluated. For example, $x := y + 4$ is a statement that assigns the evaluation of the expression $y + 4$ to the variable x.

The selection statement is a finite set of guarded commands enclosed by the if ... fi keywords:

$$
\begin{aligned}
&\textsf{if} \quad G_0 \longrightarrow S_0 \\
&[\!] \quad G_1 \longrightarrow S_1 \\
&\quad \ldots \\
&[\!] \quad G_n \longrightarrow S_n \\
&\textsf{fi}
\end{aligned}
$$

In a selection statement, only one of the guarded commands is chosen to execute. Each guarded command in the set is separated by the *alternation symbol* [] to indicate that each guarded command is one alternative in the set. Upon the execution of a selection statement, all of the guards are evaluated. If no guard evaluates to true, then execution of the selection aborts. Otherwise, one of the guards that evaluates to true is non-deterministically chosen and the corresponding statement is executed. For example, consider the following selection statement:

$$
\begin{aligned}
&\textsf{if} \quad a \geq b \longrightarrow max := a \\
&[\!] \quad b \geq a \longrightarrow max := b \\
&\textsf{fi}
\end{aligned}
$$

Upon the execution of this statement, if it is the case that $a = b$, then both guards evaluate to true. Therefore, one of the guarded commands will non-deterministically be chosen to execute. In this case, the new value for max will have equal results. Because at least one of the guards must be true for the selection statement to execute without aborting, the skip statement is often needed. For example, consider a program that requires a variable x to be reset to 0 when an error flag is set to true, otherwise the program shall do nothing. This behaviour is captured by the following statement:

$$
\begin{array}{ll}
\mathsf{if} & error = \mathsf{true} \longrightarrow x := 0 \\
[] & error = \mathsf{false} \longrightarrow \mathsf{skip} \\
\mathsf{fi} &
\end{array}
$$

The repetition statement is syntactically similar to the selection statement where instead a finite set of guarded commands is enclosed by the do ... od keywords:

$$
\begin{array}{ll}
\mathsf{do} & G_0 \longrightarrow S_0 \\
[] & G_1 \longrightarrow S_1 \\
 & \ldots \\
[] & G_n \longrightarrow S_n \\
\mathsf{od} &
\end{array}
$$

In a repetition statement, unlike in the selection statement, the guarded commands are executed repeatedly until none of the guards evaluate to true. Upon execution of a repetition statement, all of the guards are evaluated. If no guard evaluates to true, then skip is executed. Otherwise, one of the guards that evaluates to true is non-deterministically chosen and the corresponding statement is executed, after which the repetition is executed again. For example, consider the following repetition statement:

$$
\begin{array}{ll}
\mathsf{do} & a < b \longrightarrow b := b - a \\
[] & b < a \longrightarrow a := a - b \\
\mathsf{od} &
\end{array}
$$

This repetition ends only when $a = b$ since that is the point at which both guards evaluate to false, thus resulting in the execution of skip.

Lastly, in the guarded command language, statements can be separated by the sequential composition or concatenation operator ;. When the statements are selected for execution, all of the statements that are separated by the operator ; will be executed successively in the given order from left to right. For example, $x := 4 \,;\, y := x + 3$ first assigns the value 4 to the variable x, and then assigns the evaluation of the expression $x + 3$ (i.e., 7) to the variable y.

3.4 Pre- and Post-Condition Specifications and Hoare Triples

A program or behaviour can be specified in many different formalisms. One way of specifying the behaviour of a program is by a pre- and post-condition specification. In a pre- and post-condition specification, the *pre-condition* asserts what must be true before the program is executed and the *post-condition* asserts what is true after the execution of the program. A pre- and post-condition specification often takes the form of a Hoare triple.

Definition 3.4.1 (Hoare Triple — e.g., [Hoa69])**.** *A* Hoare triple *is of the form* $\{P\}\ S\ \{Q\}$ *where* P *is the pre-condition,* Q *is the post-condition, and* S *is a program statement (or a sequence of statements).* ■

A program S is said to satisfy a pre- and post-condition specification if the execution of S begins in any state in which the pre-condition P is satisfied and terminates in a state in which the post-condition Q is satisfied. A pre- and post-condition specification can be non-deterministic meaning that there may be many such programs S that satisfy the specification. For example, consider the following pre- and post-condition specification presented as a Hoare triple:

$$\{0 \leq x \leq 5\}\ S\ \{x > 5\}$$

In this case, there are infinitely many programs S that can satisfy the specification. As an example, one such program that satisfies the specification above is $S \stackrel{\text{def}}{=} x := x + 10$.

3.5 Description Logic

Description logic refers to a family of knowledge representation formalisms that represent the knowledge of an application domain by first defining the relevant *concepts* of the domain and then using these concepts to specify properties of *objects* occurring in the domain [BMNP03]. Description logics are equipped with a formal logic-based semantics with emphasis on reasoning as a central service. This allows one to infer implicitly represented knowledge from the knowledge that is explicitly contained in a knowledge base. In particular, this thesis adopts a decidable description logic called $\mathcal{ALB}$ (Attributive Language with Boolean Algebras on Concepts and Roles) [HS00] since it provides the necessary expressivity that is required for the representation of agent knowledge described in Section 4.3.

3.5.1 $\mathcal{ALB}$ Syntax

The signature of $\mathcal{ALB}$ is given by a tuple $\big(N_C, N_R, N_O\big)$ consisting of three disjoint sets of *concept* symbols, *role* symbols, and *object* symbols, respectively. Every concept symbol represents a concept and every role symbol represents a role relating two objects. Concept symbols are also called *atomic concepts* and roles symbols are also called *atomic roles*. Two special concepts, $\top$ and $\bot$, denote the *top concept* containing all objects and the *bottom concept* containing no objects, respectively. Similarly, two special roles, ∇ and Δ, denote the *top role* containing all pairs of objects and the *bottom role* containing no pairs of objects, respectively. If C and D are concepts, and R and S are roles, then complex concepts and roles are defined by induction using the constructors of Table 3.1.

A *knowledge base* $\mathcal{N} = \big(\mathcal{T}, \mathcal{A}\big)$ is comprised of two components: the *TBox* ($\mathcal{T}$) and the *ABox* ($\mathcal{A}$). The TBox contains sentences describing concept hierarchies (i.e., relations between concepts). More specifically, the TBox is any finite set of *terminological axioms*.

Concept Terms		Role Terms	
$\top$	top concept	∇	top role
$\bot$	bottom concept	Δ	bottom role
$\neg \mathsf{C}$	concept complement	$\neg \mathsf{R}$	role complement
$\mathsf{C} \sqcap \mathsf{D}$	concept intersection	$\mathsf{R} \sqcap \mathsf{S}$	role intersection
$\mathsf{C} \sqcup \mathsf{D}$	concept union	$\mathsf{R} \sqcup \mathsf{S}$	role union
$\forall \mathsf{R} . \mathsf{C}$	universal restriction	$\mathsf{R} \restriction \mathsf{C}$	domain restriction
$\exists \mathsf{R} . \mathsf{C}$	existential restriction	$\mathsf{R} \downharpoonright \mathsf{C}$	range restriction
		$\mathsf{R}^{\smile}$	role converse

Table 3.1: Constructors of the description logic $\mathcal{ALB}$

Definition 3.5.1 (Terminological Axiom — e.g., [BMNP03])**.** *A* terminological axiom *has the form of a* concept inclusion $\mathsf{C} \sqsubseteq \mathsf{D}$ *where* C *and* D *are concepts.* ■

As shorthand, $\mathsf{C} \equiv \mathsf{D}$ is written when $\mathsf{C} \sqsubseteq \mathsf{D}$ and $\mathsf{D} \sqsubseteq \mathsf{C}$. For example, the statement: "Every customer is a person" (denoted $\mathsf{Customer} \sqsubseteq \mathsf{Person}$) belongs in the TBox. Conversely, the ABox contains ground sentences stating where in the hierarchy objects belong (i.e., relations between objects and concepts). More specifically, the ABox is any finite set of *assertional axioms.*

Definition 3.5.2 (Assertional Axiom — e.g., [BMNP03])**.** *An* assertional axiom *has the form of a* concept assertion $\mathsf{C}(\mathsf{X})$ *or a* role assertion $\mathsf{R}(\mathsf{X}, \mathsf{Y})$ *where* C *is a concept,* R *is a role, and* X *and* Y *are object symbols.* ■

For example, the statements: "Richard is a customer" (denoted $\mathsf{Customer}(\mathsf{RICHARD})$) and "Richard has an account status of 'active' " (denoted $\mathsf{accountStatus}(\mathsf{RICHARD}, \mathsf{ACTIVE})$) belong in the ABox.

3.5.2 $\mathcal{ALB}$ Semantics

Formally, the semantics of the description logic $\mathcal{ALB}$ is defined by a *terminological interpretation* $(\mathcal{D}, \mathcal{I})$ over a signature (N_C, N_R, N_O) where $\mathcal{D}$ is a non-empty set called the *domain* and $\mathcal{I}$ is an *interpretation function* that assigns every concept C a set $\mathcal{I}(\mathsf{C}) \subseteq \mathcal{D}$, every role R

$$
\begin{aligned}
\mathcal{I}(\top) &= \mathcal{D} \\
\mathcal{I}(\bot) &= \emptyset \\
\mathcal{I}(\neg \mathsf{C}) &= \mathcal{D} \backslash \mathcal{I}(\mathsf{C}) \\
\mathcal{I}(\mathsf{C} \sqcap \mathsf{D}) &= \mathcal{I}(\mathsf{C}) \cap \mathcal{I}(\mathsf{D}) \\
\mathcal{I}(\mathsf{C} \sqcup \mathsf{D}) &= \mathcal{I}(\mathsf{C}) \cup \mathcal{I}(\mathsf{D}) \\
\mathcal{I}(\forall\, \mathsf{R} \,.\, \mathsf{C}) &= \{x \in \mathcal{D} \mid \forall (y \mid y \in \mathcal{D} : (x, y) \in \mathcal{I}(\mathsf{R}) \implies y \in \mathcal{I}(\mathsf{C}))\} \\
\mathcal{I}(\exists\, \mathsf{R} \,.\, \mathsf{C}) &= \{x \in \mathcal{D} \mid \exists (y \mid y \in \mathcal{D} : (x, y) \in \mathcal{I}(\mathsf{R}) \wedge y \in \mathcal{I}(\mathsf{C}))\} \\
\mathcal{I}(\nabla) &= \mathcal{D} \times \mathcal{D} \\
\mathcal{I}(\Delta) &= \emptyset \\
\mathcal{I}(\neg \mathsf{R}) &= (\mathcal{D} \times \mathcal{D}) \backslash \mathcal{I}(\mathsf{R}) \\
\mathcal{I}(\mathsf{R} \sqcap \mathsf{S}) &= \mathcal{I}(\mathsf{R}) \cap \mathcal{I}(\mathsf{S}) \\
\mathcal{I}(\mathsf{R} \sqcup \mathsf{S}) &= \mathcal{I}(\mathsf{R}) \cup \mathcal{I}(\mathsf{S}) \\
\mathcal{I}(\mathsf{R} \uparrow \mathsf{C}) &= \{(x, y) \in \mathcal{I}(\mathsf{R}) \mid x \in \mathcal{I}(\mathsf{C})\} \\
\mathcal{I}(\mathsf{R} \downarrow \mathsf{C}) &= \{(x, y) \in \mathcal{I}(\mathsf{R}) \mid y \in \mathcal{I}(\mathsf{C})\} \\
\mathcal{I}(\mathsf{R}^{\smile}) &= \{(x, y) \in \mathcal{D} \times \mathcal{D} \mid (y, x) \in \mathcal{I}(\mathsf{R})\}
\end{aligned}
$$

Table 3.2: Semantics of the description logic $\mathcal{ALB}$

a binary relation $\mathcal{I}(\mathsf{R}) \subseteq \mathcal{D} \times \mathcal{D}$, and every object X an element $\mathcal{I}(\mathsf{X}) \in \mathcal{D}$ [BMNP03]. It is required that $\mathcal{I}$ obeys the *unique name assumption*, meaning that $\mathcal{I}(\mathsf{X}) \neq \mathcal{I}(\mathsf{Y})$ for every pair of object symbols $\mathsf{X} \neq \mathsf{Y} \in N_O$ [HSG04]. The interpretation function $\mathcal{I}$ extends in a natural way to complex concepts and roles, as defined in Table 3.2 for concepts C and D, and roles R and S.

Let $\mathcal{N} = (\mathcal{T}, \mathcal{A})$ be a knowledge base and let $(\mathcal{D}, \mathcal{I})$ be a terminological interpretation. The *satisfiability relation* $\models$ is defined as shown in Table 3.3.

To this point, a satisfiability relation has been defined which determines whether a given axiom is true with respect to a given interpretation. However, an entailment relation which indicates whether an axiom is a logical consequence of a knowledge base is desired.

$$
\begin{array}{lcl}
(\mathcal{D},\mathcal{I}) \models \mathsf{C}(\mathsf{X}) & \iff & \mathcal{I}(\mathsf{X}) \in \mathcal{I}(\mathsf{C}) \\
(\mathcal{D},\mathcal{I}) \models \mathsf{C} \sqsubseteq \mathsf{D} & \iff & \mathcal{I}(\mathsf{C}) \subseteq \mathcal{I}(\mathsf{D}) \\
(\mathcal{D},\mathcal{I}) \models \mathsf{C} \equiv \mathsf{D} & \iff & \mathcal{I}(\mathsf{C}) = \mathcal{I}(\mathsf{D}) \\
 & & \\
(\mathcal{D},\mathcal{I}) \models \mathsf{R}(\mathsf{X},\mathsf{Y}) & \iff & (\mathcal{I}(\mathsf{X}),\mathcal{I}(\mathsf{Y})) \in \mathcal{I}(\mathsf{R}) \\
(\mathcal{D},\mathcal{I}) \models \mathsf{R} \sqsubseteq \mathsf{S} & \iff & \mathcal{I}(\mathsf{R}) \subseteq \mathcal{I}(\mathsf{S}) \\
(\mathcal{D},\mathcal{I}) \models \mathsf{R} \equiv \mathsf{S} & \iff & \mathcal{I}(\mathsf{R}) = \mathcal{I}(\mathsf{S}) \\
 & & \\
(\mathcal{D},\mathcal{I}) \models \mathcal{T} & \iff & \forall(\varphi \mid \varphi \in \mathcal{T} : \mathcal{I} \models \varphi) \\
(\mathcal{D},\mathcal{I}) \models \mathcal{A} & \iff & \forall(\psi \mid \psi \in \mathcal{A} : \mathcal{I} \models \psi) \\
(\mathcal{D},\mathcal{I}) \models \mathcal{N} & \iff & (\mathcal{D},\mathcal{I}) \models \mathcal{T} \text{ and } (\mathcal{D},\mathcal{I}) \models \mathcal{A}
\end{array}
$$

Table 3.3: Definition of the satisfiability relation $\models$ for the description logic $\mathcal{ALB}$

Definition 3.5.3 (Entailment Relation — e.g., [HSG04])**.** *Let φ be a terminological axiom or an assertional axiom. An axiom φ is said to be* entailed by *a knowledge base $\mathcal{N}$ (written $\mathcal{N} \models \varphi$) if and only if for every interpretation $(\mathcal{D},\mathcal{I})$ such that $(\mathcal{D},\mathcal{I}) \models \mathcal{N}$, it holds that $(\mathcal{D},\mathcal{I}) \models \varphi$ (i.e., $\mathcal{N} \models \varphi \iff \forall((\mathcal{D},\mathcal{I}) \mid (\mathcal{D},\mathcal{I}) \models \mathcal{N} : (\mathcal{D},\mathcal{I}) \models \varphi)$).* ■

When an axiom φ is entailed by a knowledge base, the agent to which the knowledge base belongs is said to *know* φ.

3.6 Conclusion

This chapter summarised the necessary mathematical background required for the remainder of this thesis. The presented algebraic structures and concurrent Kleene algebra are used to develop the mathematical framework of Communicating Concurrent Kleene Algebra (C^2KA) for specifying the concurrent and communicating behaviour of agents in Chapter 4 and for formulating the potential for communication condition for the existence of distributed covert channels in Chapter 5. Dijkstra's guarded command language is also used in Chapter 4 for providing the concrete behaviour specification of agents as part of the mathematical framework of C^2KA and in Chapter 6 as part of the approach for describing

the evolution of agent knowledge through the execution of agent behaviours. The ideas of pre- and post-condition specifications and Hoare triples are used in Chapter 6 to represent the behaviour component of communication schemes. The description logic $\mathcal{ALB}$ is used for specifying the knowledge of agents in Chapter 4. It is also used in Chapter 6 to represent the knowledge component of communication schemes and for formulating the constraint on communication condition for the existence of distributed covert channels when discussing an approach for verifying the existence of a potential confidential information leakage in a system of communicating agents.

Chapter 4

Specifying Systems of Communicating Agents

The first step towards analysing a system of communicating agents for distributed covert channels is to formally specify the system. Recall that each agent in a system of communicating agents is comprised of a *behaviour* and a *knowledge*. To specify a system of communicating agents, it suffices to provide a specification of the behaviour and knowledge of each agent in the system. This chapter precisely deals with this matter. First, Section 4.1 provides the description of a system of communicating agents that will serve as an illustrative running example throughout the remainder of this thesis. Section 4.2 introduces the mathematical framework of Communicating Concurrent Kleene Algebra (C^2KA). It also shows how to specify the behaviour of each agent using this framework. Section 4.3 introduces a representation of agent knowledge based on the description logic $\mathcal{ALB}$. It also shows how to specify the knowledge of each agent in a system of communicating agents. Lastly, Section 4.4 provides some concluding remarks about the specification of systems of communicating agents.

4.1 Running Example of a System of Communicating Agents

This section describes a simple example of a system of communicating agents whereby a sending agent and a receiving agent are able to establish and operate a covert communication channel based on a variation of the FTP command mapping communication scheme from [ZLSN05]. This example will serve as a running example throughout the remainder of this thesis. It will be used to illustrate the specification of agent behaviour and knowledge in this chapter, the analysis of the potential for communication condition for the existence of distributed covert channels in Chapter 5, and the development, representation, and merging of communication schemes into specifications of systems of communicating agents, as well as the evolution of agent knowledge through the execution of agent behaviours in Chapter 6. Ultimately, it will be shown in Chapter 6 that the system of communicating agents described in this section contains a covert channel that can be used to leak confidential information to an agent which should not know or possess that information.

Consider a system formed by a set $\mathcal{C}$ of communicating agents consisting of five agents: $\{\mathsf{C}, \mathsf{S}, \mathsf{P}, \mathsf{Q}, \mathsf{R}\}$. Suppose that this system of communicating agents makes use of a common FTP client software. The common FTP client software integrates a function, called the *idle prevention scheme*, which guarantees that at least one command will be sent in a fixed period of time [ZLSN05]. The set of commands that can be sent to implement the idle prevention scheme can be selected by a user or system agent. For the purpose of this example, let the command space of the idle prevention scheme for the FTP client be denoted by $\mathbb{FTP}$ (i.e., $\mathbb{FTP}$ is the set of all FTP commands selected to implement the idle prevention scheme in the system). For simplicity and brevity, consider an FTP command space of the idle prevention scheme consisting of only four commands such that $\mathbb{FTP} = \{\mathit{ABOR}, \mathit{ALLO}, \mathit{HELP}, \mathit{NOOP}\}$. In this way, the idle prevention scheme can choose to issue any of these commands at will.

In the system formed by a set $\mathcal{C}$ of communicating agents, let C denote a clock agent that increments a variable called `time` which is shared amongst all of the agents in the system. For the purpose of this example, assume that time is discrete and modelled by the set of natural numbers $\mathbb{N}$. In this way, the variable `time` represents the number of clock ticks that have occurred since the system has started operating. Let S be an agent that waits for the idle prevention scheme to determine the time to send a command from the command space $\mathbb{FTP}$. Once agent S receives a signal from the idle prevention scheme, it chooses and issues a command. Moreover, for the purpose of this example, assume that agent S has knowledge of some set of confidential information. It is acknowledged that the confidential information in a real system of communicating agents may be very large, but for simplicity and illustration, let the set of confidential information be the set $\{01\}$, containing only the (very small) bit-string 01. Additionally, suppose that the security policy for this system explicitly forbids any agent, other than agent S, from knowing or possessing this information. Let P and Q be agents that listen for FTP commands to be issued. When agent P receives an FTP command, it assigns an enumeration of the received command to a variable called `cmd` representing the most recent command that was received. Assume that agent P is configured with an enumeration mapping of all of the possible FTP commands when they are sorted alphabetically (i.e., P initially knows the following mapping $\{(ABOR, 1), (ALLO, 2), (HELP, 3), (NOOP, 4)\}$). Assume that agents S and R know how agent P behaves in the system. This means that agents S and R know the enumeration mapping that agent P is configured with. As it will become more clear in the subsequent chapters, it is through this shared knowledge of the operation of agent P, coupled with the potential for communication among the system agents, that agents S and R will be able to develop a covert communication scheme and establish a covert channel in the system. After storing the enumeration of the received command, agent P increments a counter denoted by a variable for each command (i.e., for all i such that $1 \leq i \leq |\mathbb{FTP}|$) called `num_i` where i represents the enumeration corresponding to the received command. For example,

if agent P receives a *NOOP* command, then it assigns `cmd` := 4 and increments `num_4`. In this example, assume that the counters `num_i` for $1 \leq i \leq |\mathbb{FTP}|$ and the variable `cmd` are shared with agents Q and R. Finally, when agent P has completed its computation, it forwards the FTP command that it received. When the idle prevention scheme signals that it is time to send a command, agent Q begins to wait for a command to be issued. When agent Q receives an FTP command, it computes the number of clock ticks that have passed since the last command was received and stores the value in a variable called `delta` that is shared with all of the other agents in the system except for the clock agent C. For example, when agent Q receives a command, it assigns `delta` := `time` − `last` where `last` is a local variable storing the time at which the last command was received. Once the computation is complete, agent Q forwards the FTP command that it received. Finally, let R be an agent that computes the average arrival time of each command using the values stored in the `cmd` and `delta` variables that it shares with agents P and Q. For instance, agent R updates a local variable for each command (i.e., for all i such that $1 \leq i \leq |\mathbb{FTP}|$) called `since_i` representing the total time that has passed since the command enumerated as i has been received by computing `since_i`+`delta`. Then, agent R computes `since_i`/`num_i` and stores it in a variable called `avg_i` that is shared with all of the other agents in the system except for the clock agent C. The operation of the system of communicating agents described in this section can be visualised as shown in Figure 4.1.

4.2 Specifying Agent Behaviour

This section proposes an extension to concurrent Kleene algebra, called *Communicating Concurrent Kleene Algebra* (C^2KA), as a mathematical framework for capturing the behaviour of agents in systems of communicating agents. C^2KA allows for the separation of communicating and concurrent behaviour in a system and its environment and is able to express the influence of external stimuli on the behaviours of a system of communicating

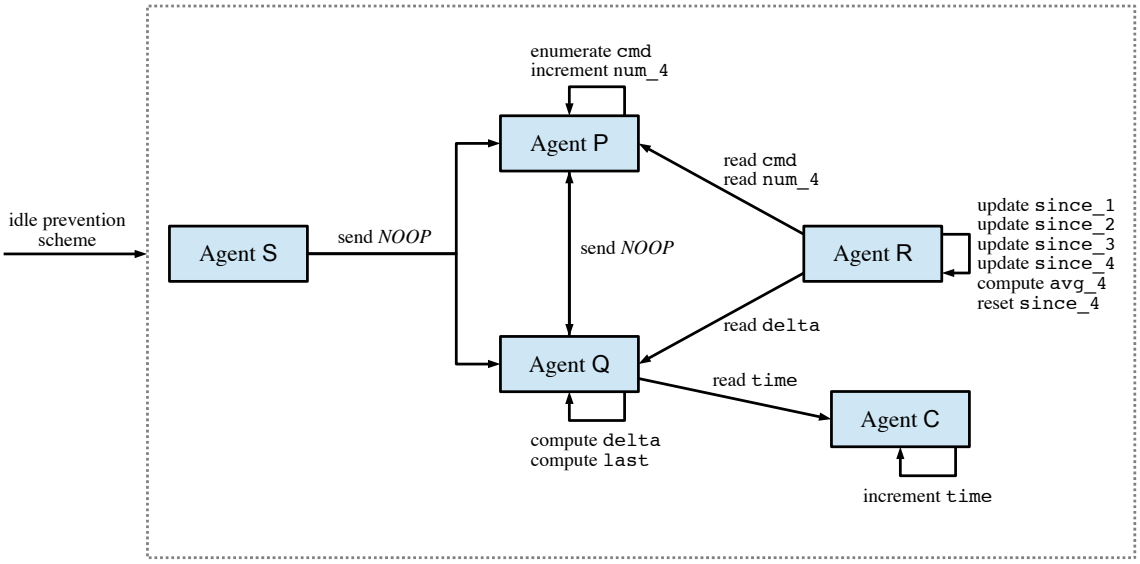

Figure 4.1: A visualisation of the operation of the running example system of communicating agents when the idle prevention scheme is implemented using a *NOOP* command

agents. It presents a different view of communication and concurrency than what is traditionally given by existing process calculi and other formalisms for capturing the concurrent and communicating behaviour of agents. In this thesis, C^2KA is the formalism that is used to specify agent behaviour in systems of communicating agents.

4.2.1 Rationale for C^2KA

As mentioned in Section 2.2, existing formalisms for capturing the concurrent and communicating behaviour of agents do not directly, if at all, provide a hybrid view of communication and concurrency encompassing the characteristics of both state-based and event-based models. Usually formalisms are either state-based or event-based. For example, temporal logics, such as LTL [Pnu77], CTL [CE82], and CTL* [EH86], represent state-based formalisms. Conversely, labelled transition systems [Kel76], Petri nets [Pet62], process calculi (e.g., CCS [Mil80], CSP [Hoa78a], ACP [BK84], and π-calculus [MPW92]) are examples representing event-based formalisms. Even with a formalism such as concurrent Kleene

algebra (CKA), which can be perceived as a hybrid model for concurrency, the notion of communication is not directly captured. Communication can be perceived only when programs are given in terms of the dependencies of shared events [HW11]. This requires the instantiation of a low-level model of programs and traces for CKA in order to define any sort of communication. Instead, a way in which communication can be specified in CKA without the need to articulate the state-based system of each action (i.e., at a convenient abstract level) is desired.

Furthermore, there is currently no such formalism that deals directly with describing how the behaviours of agents in a system are influenced by external stimuli. The influence of external stimuli is an important aspect that ought to be considered when capturing the concurrent and communicating behaviour of agents in systems of communicating agents, particularly when dealing with open systems. In open systems, external stimuli are required to initiate agent behaviours. This is to say that the agents in an open system need an external influence from the world in which they reside in order to begin their operation. Existing formalisms, such as CKA and process calculi, deal primarily with closed systems where there is no external influence on the behaviours of agents and they do not directly, if at all, consider agent behaviours in open systems.

To address these shortcomings of the current literature, this thesis proposes the mathematical framework of Communicating Concurrent Kleene Algebra. C^2KA offers an algebraic setting which can capture both the influence of external stimuli on agent behaviour as well as the communication and concurrency of agents at the abstract algebraic level. It uses notions from classical algebra to extend the algebraic foundation provided by CKA.

4.2.2 Structure of Agent Behaviours

In [HMSW09a, HMSW09b, HMSW10, HMSW11], Hoare et al. presented the framework of concurrent Kleene algebra, which captures the concurrent behaviour of agents. The framework of CKA is adopted in order to describe agent behaviours in systems of communicating

agents. In what follows, let $\mathcal{K} \stackrel{\text{def}}{=} \big(K, +, *, ;, \circledast, \circledcirc, 0, 1\big)$ be called a CKA (see Section 3.2 for more information on CKA).

In a CKA $\mathcal{K}$, the support set K represents a set of possible agent behaviours. The operator $+$ is interpreted as a choice between two behaviours, the operator $;$ is interpreted as a sequential composition of two behaviours, and the operator $*$ is interpreted as a parallel composition of two behaviours. The operators $\circledcirc$ and $\circledast$ are interpreted as a finite sequential iteration and a finite parallel iteration of behaviours, respectively. The element 0 represents the behaviour of the *inactive agent* and the element 1 represents the behaviour of the *idle agent* just as in many process calculi. Moreover, associated with a CKA is a natural ordering relation $\leq_{\mathcal{K}}$ representing the sub-behaviour relation. For behaviours $a, b \in K$, $a \leq_{\mathcal{K}} b$ indicates that a is a sub-behaviour of b if and only if $a + b = b$.

When speaking of agents and agent behaviours, $\mathsf{A} \mapsto \langle a \rangle$ is written to indicate that A is the name given to the agent and $a \in K$ is the agent behaviour. For $\mathsf{A} \mapsto \langle a \rangle$ and $\mathsf{B} \mapsto \langle b \rangle$, $\mathsf{A} + \mathsf{B}$ is written to denote the agent $\langle a + b \rangle$. In a sense, the operators on behaviours of K are extended to their corresponding agents. In this way, an agent is defined by simply describing its behaviour. Because of this, the terms agents and behaviours may be used interchangeably.

Consider the running example described in Section 4.1. Let the set of all agent behaviours K for the given system be generated using the operations of CKA and the set $\{$ABOR, ALLO, HELP, NOOP, $\text{SEND}_{\text{ABOR}}$, $\text{SEND}_{\text{ALLO}}$, $\text{SEND}_{\text{HELP}}$, $\text{SEND}_{\text{NOOP}}$, SINCE_1, SINCE_2, SINCE_3, SINCE_4, COUNT_1, COUNT_2, COUNT_3, COUNT_4, AVG_1, AVG_2, AVG_3, AVG_4, RESET_1, RESET_2, RESET_3, RESET_4, TICK, DELTA, LAST, WAIT, READ, 0, 1$\}$. ABOR, ALLO, HELP, and NOOP represent the behaviours that map each FTP command to their corresponding enumeration. $\text{SEND}_{\text{ABOR}}$, $\text{SEND}_{\text{ALLO}}$, $\text{SEND}_{\text{HELP}}$, and $\text{SEND}_{\text{NOOP}}$ represent the issuance of a particular FTP command. For all $1 \leq i \leq |\mathbb{FTP}|$, SINCE_i denotes the behaviour which computes and stores the total time that has passed since the command enumerated as i has been received. COUNT_i denotes the behaviour which increments the counter storing the number of times that the command

enumerated as i has been received. AVG_i represents the behaviour that computes and stores the average arrival time of the command enumerated as i. RESET_i denotes the behaviour that resets the total time that has passed since the command enumerated as i has been received to 0. TICK denotes the behaviour that increments the variable representing the number of clock ticks that have occurred since the system has started operating. DELTA represents the behaviour that computes and stores the number of clock ticks that have passed since the last command was received. LAST represents the behaviour that stores the time at which the last command was received. WAIT represents the behaviour that idly waits for an FTP command that implements the idle prevention scheme to be issued. READ represents the behaviour that reads the shared variable `cmd`. The inactive agent behaviour 0 represents a behaviour that fails, and the idle agent behaviour 1 represents a behaviour that does nothing.

4.2.3 Structure of External Stimuli

A stimulus constitutes the basis for behaviour. Because of this, each discrete, observable event introduced to a system, such as that which occurs through the communication among agents or from the system environment, is considered to be an external stimulus which invokes a response from each system agent.

Definition 4.2.1 (Stimulus Structure)**.** *Let* $\mathcal{S} \stackrel{\text{def}}{=} \left(S, \oplus, \odot, \mathfrak{d}, \mathfrak{n}\right)$ *be an idempotent semiring with a multiplicatively absorbing* $\mathfrak{d}$ *and identity* $\mathfrak{n}$*. The structure* $\mathcal{S}$ *is called a* stimulus structure*.* ■

Within the context of external stimuli, S is the set of external stimuli which may be introduced to a system. The operator $\oplus$ is interpreted as a choice between two stimuli and the operator $\odot$ is interpreted as a sequential composition of two stimuli. The element $\mathfrak{d}$ represents the *deactivation stimulus* which influences all agents to become inactive and the element $\mathfrak{n}$ represents the *neutral stimulus* which has no influence on the behaviour of all agents.

Definition 4.2.2 (Basic Stimulus)**.** *A stimulus $s \in S$ is called a* basic stimulus *if it is indivisible with regard to the $\odot$ operator (i.e., $\forall\big(t \mid t|s : t = \mathfrak{n} \vee t = s\big)$ and $\forall\big(t, r \mid s|(t \odot r) : s|t \vee s|r\big)$ where the divisibility relation $|$ is defined by $x|y \iff \exists(z \mid: y = x \odot z)$). The set of all basic stimuli is denoted as $S_\flat$.* ■

Furthermore, each stimulus structure has a natural ordering relation $\leq_{\mathcal{S}}$ representing the sub-stimulus relation. For external stimuli $s, t \in S$, $s \leq_{\mathcal{S}} t$ indicates that s is sub-stimulus of t if and only if $s \oplus t = t$.

Once again, consider the running example from Section 4.1. Let the set of all external stimuli S be generated using the operations of stimulus structures and the set $\{$*ips*, *abor*, *allo*, *help*, *noop*, $\mathfrak{n}$, $\mathfrak{d}\}$. The stimuli denoted by *abor*, *allo*, *help*, and *noop* represent the sending of each of the FTP commands in the command space $\mathbb{FTP}$. The stimulus *ips* represents the signal sent from the idle prevention scheme to the system of communicating agents in order to cause the system to issue a command to prevent the FTP client from entering an idle state. The set of external stimuli also includes the deactivation stimulus $\mathfrak{d}$ which is interpreted as a kill signal and the neutral stimulus $\mathfrak{n}$ which is interpreted as any stimulus with no influence that belongs to the complement of the set of external stimuli which may be introduced to a system.

4.2.4 Communicating Concurrent Kleene Algebra (C^{2}KA)

C^{2}KA extends the algebraic foundation of CKA with the notions of semimodules and stimulus structures to capture the influence of external stimuli on the behaviour of system agents.

In essence, a C^{2}KA consists of two semimodules which describe how a stimulus structure $\mathcal{S}$ and a CKA $\mathcal{K}$ mutually act upon one another in order to characterise the response invoked by an external stimulus on the behaviour of an agent as a next behaviour and a next stimulus.

First, the left $\mathcal{S}$-semimodule $\big({}_{\mathcal{S}}K, +\big)$ describes how the stimulus structure $\mathcal{S}$ acts upon the CKA $\mathcal{K}$ via the mapping $\circ$. The mapping $\circ$ is called the *next behaviour mapping* since it describes how an external stimulus invokes a behavioural response from a given agent. From $\big({}_{\mathcal{S}}K, +\big)$, the next behaviour mapping $\circ$ distributes over $+$ and $\oplus$. Additionally, since $\big({}_{\mathcal{S}}K, +\big)$ is unitary, it is the case that the neutral stimulus has no influence on the behaviour of all agents and since $\big({}_{\mathcal{S}}K, +\big)$ is zero-preserving, the deactivation stimulus influences all agents to become inactive. Second, the right $\mathcal{K}$-semimodule $\big(S_{\mathcal{K}}, \oplus\big)$ describes how the CKA $\mathcal{K}$ acts upon the stimulus structure $\mathcal{S}$ via the mapping λ. The mapping λ is called the *next stimulus mapping* since it describes how a new stimulus is generated as a result of the response invoked by a given external stimulus on an agent behaviour. From $\big(S_{\mathcal{K}}, \oplus\big)$, the next stimulus mapping λ distributes over $\oplus$ and $+$. Also, since $\big(S_{\mathcal{K}}, \oplus\big)$ is unitary, it is the case that the idle agent forwards any external stimulus that acts on it and since $\big(S_{\mathcal{K}}, \oplus\big)$ is zero-preserving, the inactive agent always generates the deactivation stimulus.

Definition 4.2.3 (Communicating Concurrent Kleene Algebra)**.** *A* Communicating Concurrent Kleene Algebra (C^2KA) *is a system* $\big(\mathcal{S}, \mathcal{K}\big)$*, where* $\mathcal{S} = \big(S, \oplus, \odot, \mathfrak{d}, \mathfrak{n}\big)$ *is a stimulus structure and* $\mathcal{K} = \big(K, +, *, ;, {}^{\circledast}, {}^{\textcircled{\scriptsize :}}, 0, 1\big)$ *is a* CKA *such that* $\big({}_{\mathcal{S}}K, +\big)$ *is a unitary and zero-preserving* left $\mathcal{S}$-semimodule *with mapping* $\circ : S \times K \to K$ *and* $\big(S_{\mathcal{K}}, \oplus\big)$ *is a unitary and zero-preserving* right $\mathcal{K}$-semimodule *with mapping*[1] $\lambda : S \times K \to S$*, and where the following axioms are satisfied for all* $a, b, c \in K$ *and* $s, t \in S$*:*

(1) $s \circ (a \,;\, b) = (s \circ a) \,;\, \big(\lambda(s, a) \circ b\big)$

(2) $a \leq_{\mathcal{K}} c \;\vee\; b = 1 \;\vee\; (s \circ a) \,;\, \big(\lambda(s, c) \circ b\big) = 0$

(3) $\lambda(s \odot t, a) = \lambda\big(s, (t \circ a)\big) \odot \lambda(t, a)$

(4) $s = \mathfrak{d} \;\vee\; s \circ 1 = 1$

(5) $a = 0 \;\vee\; \lambda(\mathfrak{n}, a) = \mathfrak{n}$

■

[1] An infix notation is used for the next behaviour mapping $\circ$ and a prefix notation for the next stimulus mapping λ. These notations are adopted in an effort to reach out to those in the communities of monoid acts and Mealy automata since they adopt a similar non-uniform notation.

In Definition 4.2.3, Axiom (1) describes the interaction of the next behaviour mapping $\circ$ with the sequential composition operator $;$ for agent behaviours. This axiom corresponds to the definition of the transition function for the cascading product (or synchronous serial composition) of Mealy automata [Hol04]. Axiom (2), which is referred to as the *cascading output law*, states that when an external stimulus is introduced to the sequential composition $(a\,;b)$, then either the cascaded stimulus must be generated by the behaviour a, or the behaviour b must be the idle agent behaviour 1. It allows distributivity of $\circ$ over $;$ to be applied indiscriminately. In order to illustrate the motivation for the cascading output law, consider a stimulus $s \in S$ and a behaviour $\big((a+b)\,;c\big) \in K$. Then, the calculation of $\big(s \circ (a+b)\,;c\big)$ is given in Example 4.2.1.

Example 4.2.1 (Cascading Output Law)**.** *Let $s \in S$ and $\big((a+b)\,;c\big) \in K$.*

$$
\begin{aligned}
& s \circ \big((a+b)\,;c\big) \\
= & \qquad \langle \textit{ Definition 4.2.3(1 } \rangle \\
& (s \circ (a+b))\,;\big(\lambda\big(s,(a+b)\big) \circ c\big) \\
= & \qquad \langle \textit{ Definition 3.1.8(1) for } \big({}_{\mathcal{S}}K,+\big) \rangle \\
& (s \circ a + s \circ b)\,;\big(\lambda\big(s,(a+b)\big) \circ c\big) \\
= & \qquad \langle \textit{ Definition 3.1.8(1) for } \big(S_{\mathcal{K}},\oplus\big) \rangle \\
& (s \circ a + s \circ b)\,;\big((\lambda(s,a) \oplus \lambda(s,b)) \circ c\big) \\
= & \qquad \langle \textit{ Definition 3.1.8(2) for } \big({}_{\mathcal{S}}K,+\big) \rangle \\
& (s \circ a + s \circ b)\,;\big(\lambda(s,a) \circ c + \lambda(s,b) \circ c\big) \\
= & \qquad \langle \textit{ Distributivity of } ; \textit{ over } + \rangle \\
& \big[(s \circ a + s \circ b)\,;\big(\lambda(s,a) \circ c\big)\big] + \big[(s \circ a + s \circ b)\,;\big(\lambda(s,b) \circ c\big)\big] \\
= & \qquad \langle \textit{ Distributivity of } ; \textit{ over } + \rangle
\end{aligned}
$$

$$
\begin{aligned}
& (s \circ a) \,;\, \big(\lambda(s,a) \circ c\big) + (s \circ b) \,;\, \big(\lambda(s,a) \circ c\big) + (s \circ a) \,;\, \big(\lambda(s,b) \circ c\big) + (s \circ b) \,;\, \big(\lambda(s,b) \circ c\big) \\
= & \qquad \langle \textit{ Definition 4.2.3(2) } \rangle \\
& (s \circ a) \,;\, \big(\lambda(s,a) \circ c\big) + 0 + 0 + (s \circ b) \,;\, \big(\lambda(s,b) \circ c\big) \\
= & \qquad \langle \textit{ Identity of } + \rangle \\
& (s \circ a) \,;\, \big(\lambda(s,a) \circ c\big) + (s \circ b) \,;\, \big(\lambda(s,b) \circ c\big) \\
= & \qquad \langle \textit{ Definition 4.2.3(1) } \rangle \\
& s \circ (a \,;\, c) + s \circ (b \,;\, c)
\end{aligned}
$$

Without the cascading output law, the result in Example 4.2.1 would have two additional sub-behaviours, namely, $(s \circ b) \,;\, \big(\lambda(s,a) \circ c\big)$ and $(s \circ a) \,;\, \big(\lambda(s,b) \circ c\big)$. However, due to the cascading output law, since each of these sub-behaviours contain a cascaded stimulus that results from a different behaviour than that of the first component of the sequential composition, and since the second component of the sequential composition in each case is not the idle behaviour 1, the resulting behaviour for each of these sub-behaviours is the inactive behaviour 0. In this way, the cascading output law ensures consistency between the next behaviour and next stimulus mappings with respect to the sequential composition of agent behaviours. Axiom (3) describes the interaction of the next stimulus mapping λ with the sequential composition operator $\odot$ for external stimuli. This can be viewed as the analog of Axiom (1) with respect to the next stimulus mapping λ when considering the action of $\big(S_{\mathcal{K}}, \oplus\big)$. Axiom (4), which is referred to as the *idle agent law*, states that the idle agent 1 is not influenced by any external stimulus other than the deactivation stimulus $\mathfrak{d}$. The idle agent law indicates that the idle agent is passive and can be seen as having no state-changing observed behaviour. In this way, the idle agent does not actively participate in the operation of a system. The idle agent law plays a role in proving Proposition 4.2.6. Finally, Axiom (5), which is referred to as the *neutral stimulus law*, states that no agent, other than the inactive agent 0, is able to generate a new stimulus without outside influence.

As mentioned in Section 4.2.1, external stimuli are required to initiate agent behaviours in open systems. In open systems, the stimulus structure plays a vital role in describing the concurrent and communicating behaviour of agents. However, in a closed system where there is no external influence, a C^{2}KA can be considered to have trivial stimulus structure (i.e., $S = \{\mathfrak{n}, \mathfrak{d}\}$). In this case, the next behaviour and next stimulus mappings are trivial and it becomes easy to see that the C^{2}KA reduces to a CKA. By this observation, it is clear that C^{2}KA provides and extension to CKA by allowing for the consideration of open systems and the influence of external stimuli on agent behaviours in systems of communicating agents. These observations pertaining to the open and closed world views of agent behaviour are particularly evident by considering the neutral stimulus law, in conjunction with the fact that $\left({}_{\mathcal{S}}K, +\right)$ is unitary.

The following proposition is a result of the axiomatisation of C^{2}KA.

Proposition 4.2.1. *Let* $\left(\mathcal{S}, \mathcal{K}\right)$ *be a* C^{2}KA. *For all* $a, b \in K$ *and* $s, t \in S$*:*

(1) $a \leq_{\mathcal{K}} b \;\wedge\; s \leq_{\mathcal{S}} t \implies s \circ a \leq_{\mathcal{K}} t \circ b$ (2) $a \leq_{\mathcal{K}} b \;\wedge\; s \leq_{\mathcal{S}} t \implies \lambda(s, a) \leq_{\mathcal{S}} \lambda(t, b)$

Proof. The detailed proofs can be found in Appendix A.2. □

The isotonicity laws given in Corollary 4.2.2 follow immediately from Proposition 4.2.1.

Corollary 4.2.2. *In a* C^{2}KA *where the underlying* CKA *and stimulus structure are built up from quantales, the following laws hold:*

(1) $a \leq_{\mathcal{K}} b \implies s \circ a \leq_{\mathcal{K}} s \circ b$

(2) $s \leq_{\mathcal{S}} t \implies s \circ a \leq_{\mathcal{K}} t \circ a$

(3) $s \circ (a \,;\, b + b \,;\, a) \leq_{\mathcal{K}} s \circ (a * b)$

(4) $s \circ a^{\textcircled{:}} \leq_{\mathcal{K}} s \circ a^{\circledast}$

(5) $s \circ a^{\textcircled{:}} = +(n \mid n \geq 0 \,:\, s \circ a^n)$

(6) $s \leq_{\mathcal{S}} t \implies \lambda(s, a) \leq_{\mathcal{S}} \lambda(t, a)$

(7) $a \leq_{\mathcal{K}} b \implies \lambda(s, a) \leq_{\mathcal{S}} \lambda(s, b)$

(8) $\lambda(s, (a \,;\, b + b \,;\, a)) \leq_{\mathcal{S}} \lambda(s, (a * b))$

(9) $\lambda(s, a^{\textcircled{:}}) \leq_{\mathcal{S}} \lambda(s, a^{\circledast})$

(10) $\lambda(s, a^{\textcircled{:}}) = \oplus(n \mid n \geq 0 \,:\, \lambda(s, a^n))$

Proof. The detailed proofs can be found in Appendix A.3. □

4.2.5 A Comment on a Model for C^{2}KA

In [HMSW09a, HMSW09b, HMSW10, HMSW11], Hoare et al. provided the following model for CKA. Let EV be a set of event occurrences. A *trace* is a set of events and a *program* is a set of traces. The set of all traces over EV is denoted by $TR(EV) \stackrel{\text{def}}{=} \mathcal{P}(EV)$ and the set of all programs is denoted by $PR(EV) \stackrel{\text{def}}{=} \mathcal{P}(TR(EV))$. Obviously, $\big(PR(EV), \cup, *, ;, \circledast, \odot, \emptyset, \{\emptyset\}\big)$ is a CKA [HMSW09a, HMSW09b, HMSW10, HMSW11]. Moreover, the structure of external stimuli can be modelled by sets of strings. In this way, it is easy to see that $\big(\mathcal{P}(\Lambda^*), \cup, \bullet, \emptyset, \{\epsilon\}\big)$ is a stimulus structure, where Λ is a set of alphabet symbols, $\bullet$ denotes set concatenation, and ϵ is the empty string.

In a C^{2}KA, the semimodules $\big({}_{\mathcal{S}}K, +\big)$ and $\big(S_{\mathcal{K}}, \oplus\big)$ contain a left $\mathcal{S}$-act ${}_{\mathcal{S}}K$ and a right $\mathcal{K}$-act $S_{\mathcal{K}}$, respectively. It is well known that monoid acts can be considered as semiautomata [KKM00, pg. 45]. The combination of these two semiautomata leads to a Mealy automaton. A Mealy automaton is given by a five-tuple $\big(Q, \Sigma, \Theta, F, G\big)$ [Hol04]. The set of states Q is a subset of $PR(EV)$ (i.e., the set K). In this way, each state of the Mealy automaton represents a possible program that can be executed by the system as a reaction to the stimulus (input) leading to the state. The input alphabet Σ and output alphabet Θ are given by the stimulus structure such that $\Sigma = \Theta = S$. Finally, the transition function $F : \Sigma \times Q \to Q$ and the output function $G : \Sigma \times Q \to \Theta$ correspond to the next behaviour mapping $\circ : S \times K \to K$ and next stimulus mapping $\lambda : S \times K \to S$, respectively. These mappings respectively correspond to the transition functions from the semiautomata representations of ${}_{\mathcal{S}}K$ and $S_{\mathcal{K}}$.

The proposed model is also equipped with two operations for Mealy automata. The operation ; is associative and the operation + is associative, idempotent, and commutative. The ; operation corresponds to the *cascading product* of Mealy automata and the operation + corresponds to the *full direct product* of Mealy automata [Hol04].

In [Hoa78b], Hoare provided a trace semantics for Dijkstra's guarded command language [Dij75]. Let P and Q be commands and let $TR(P)$ and $TR(Q)$ denote the associated sets of traces in accordance with the following recursive definition where $\bullet$ denotes set concatenation:

$$
\begin{aligned}
TR(\mathsf{abort}) &= \{\mathsf{false}\} \\
TR(\mathsf{skip}) &= \{\mathsf{true}\} \\
TR(x := E) &= \{x := E\} \\
TR(P\,;Q) &= TR(P) \bullet TR(Q) \\
TR(\mathsf{if}\ \ c \longrightarrow P\ [\!]\ d \longrightarrow Q\ \mathsf{fi}) &= \{c\} \bullet TR(P)\ \cup\ \{d\} \bullet TR(Q) \\
TR(\mathsf{do}\ \ c \longrightarrow P\ \mathsf{od}) &= \bigcup_{n \geq 0} S_n \\
&\quad \text{where}\ \ S_0 = \{\neg c\},\ \ S_{n+1} = \big(\{c\} \bullet TR(P) \bullet S_n\big)\ \cup\ S_0
\end{aligned}
$$

From this recursive definition, abort always fails and skip has a no effect and never fails. An assignment statement corresponds to itself. A trace of a sequential composition corresponds to a trace of the command P followed by a trace of the command Q. A trace of the selection statement is a trace of one of the alternatives, preceded by a record of the guard that evaluated to true. For a repetition statement, S_n represents the trace of exactly n iterations of the command. Each iteration of a repetition statement is a record of the truth of the guard followed by a trace of the command, and where the last iteration is followed by a record of the falsity of the guard. It is important to note that, at the expense of notational inconvenience, the selection statement and the repetition statement can be extended to contain a set of n guarded commands.

By considering Dijkstra's guarded command language with an additional parallel composition operation $*$ having an interleaving semantics, it becomes easy to see that Dijkstra's guarded command language may be used to specify the programs that represent the carrier

set of a CKA and the usage of guarded commands falls within the model of programs and traces for CKA as described above and in [HMSW09a, HMSW09b, HMSW10, HMSW11]. In this way, Dijkstra's guarded command language can be used to provide the state-level specification of agent behaviours (see Section 4.2.6) which, in turn, can be modelled as sets of traces.

4.2.6 Specifying Systems of Communicating Agents with C^{2}KA

C^{2}KA offers three levels of specification that may be considered when specifying a system of communicating agents. Depending on the context of a given problem, the most suitable level of specification can be selected.

Stimulus-Response Specification of Agents

At the stimulus-response specification of agents level, the specification of the next behaviour mapping $\circ$ and the next stimulus mapping λ for each agent in the system are provided. The running example described in Section 4.1 can be specified using the C^{2}KA constructed from the CKA generated by the set of agent behaviours $\{\text{ABOR}, \text{ALLO}, \text{HELP}, \text{NOOP}, \text{SEND}_{\text{ABOR}}, \text{SEND}_{\text{ALLO}}, \text{SEND}_{\text{HELP}}, \text{SEND}_{\text{NOOP}}, \text{SINCE}_1, \text{SINCE}_2, \text{SINCE}_3, \text{SINCE}_4, \text{COUNT}_1, \text{COUNT}_2, \text{COUNT}_3, \text{COUNT}_4, \text{AVG}_1, \text{AVG}_2, \text{AVG}_3, \text{AVG}_4, \text{RESET}_1, \text{RESET}_2, \text{RESET}_3, \text{RESET}_4, \text{TICK}, \text{DELTA}, \text{LAST}, \text{WAIT}, \text{READ}, 0, 1\}$ and the stimulus structure generated by the set of external stimuli $\{ips, abor, allo, help, noop, \mathfrak{n}, \mathfrak{d}\}$. The stimulus-response specifications of the agents in the running example are compactly specified as shown in Tables 4.1 to 4.5. It is important to note that all together, Tables 4.1 to 4.5 define a single next behaviour mapping $\circ$ and a single next stimulus mapping λ.

$\circ$	*ips*	*abor*	*allo*	*help*	*noop*
TICK	TICK	TICK	TICK	TICK	TICK

λ	*ips*	*abor*	*allo*	*help*	*noop*
TICK	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$

Table 4.1: Stimulus-response specification of agent C

$\circ$	*ips*	*abor*	*allo*	*help*	*noop*
$\text{SEND}_{\text{ABOR}}$	$\text{SEND}_{\text{ABOR}}$	$\text{SEND}_{\text{ABOR}}$	$\text{SEND}_{\text{ABOR}}$	$\text{SEND}_{\text{ABOR}}$	$\text{SEND}_{\text{ABOR}}$
$\text{SEND}_{\text{ALLO}}$	$\text{SEND}_{\text{ALLO}}$	$\text{SEND}_{\text{ALLO}}$	$\text{SEND}_{\text{ALLO}}$	$\text{SEND}_{\text{ALLO}}$	$\text{SEND}_{\text{ALLO}}$
$\text{SEND}_{\text{HELP}}$	$\text{SEND}_{\text{HELP}}$	$\text{SEND}_{\text{HELP}}$	$\text{SEND}_{\text{HELP}}$	$\text{SEND}_{\text{HELP}}$	$\text{SEND}_{\text{HELP}}$
$\text{SEND}_{\text{NOOP}}$	$\text{SEND}_{\text{NOOP}}$	$\text{SEND}_{\text{NOOP}}$	$\text{SEND}_{\text{NOOP}}$	$\text{SEND}_{\text{NOOP}}$	$\text{SEND}_{\text{NOOP}}$

λ	*ips*	*abor*	*allo*	*help*	*noop*
$\text{SEND}_{\text{ABOR}}$	*abor*	*abor*	*abor*	*abor*	*abor*
$\text{SEND}_{\text{ALLO}}$	*allo*	*allo*	*allo*	*allo*	*allo*
$\text{SEND}_{\text{HELP}}$	*help*	*help*	*help*	*help*	*help*
$\text{SEND}_{\text{NOOP}}$	*noop*	*noop*	*noop*	*noop*	*noop*

Table 4.2: Stimulus-response specification of agent S

$\circ$	*ips*	*abor*	*allo*	*help*	*noop*
ABOR	ABOR	ABOR	ALLO	HELP	NOOP
ALLO	ALLO	ABOR	ALLO	HELP	NOOP
HELP	HELP	ABOR	ALLO	HELP	NOOP
NOOP	NOOP	ABOR	ALLO	HELP	NOOP
COUNT_1	COUNT_1	COUNT_1	COUNT_2	COUNT_3	COUNT_4
COUNT_2	COUNT_2	COUNT_1	COUNT_2	COUNT_3	COUNT_4
COUNT_3	COUNT_3	COUNT_1	COUNT_2	COUNT_3	COUNT_4
COUNT_4	COUNT_4	COUNT_1	COUNT_2	COUNT_3	COUNT_4

λ	*ips*	*abor*	*allo*	*help*	*noop*
ABOR	$\mathfrak{n}$	*abor*	*allo*	*help*	*noop*
ALLO	$\mathfrak{n}$	*abor*	*allo*	*help*	*noop*
HELP	$\mathfrak{n}$	*abor*	*allo*	*help*	*noop*
NOOP	$\mathfrak{n}$	*abor*	*allo*	*help*	*noop*
COUNT_1	$\mathfrak{n}$	*abor*	*allo*	*help*	*noop*
COUNT_2	$\mathfrak{n}$	*abor*	*allo*	*help*	*noop*
COUNT_3	$\mathfrak{n}$	*abor*	*allo*	*help*	*noop*
COUNT_4	$\mathfrak{n}$	*abor*	*allo*	*help*	*noop*

Table 4.3: Stimulus-response specification of agent P

$\circ$	*ips*	*abor*	*allo*	*help*	*noop*
DELTA	WAIT	DELTA	DELTA	DELTA	DELTA
LAST	WAIT	LAST	LAST	LAST	LAST
WAIT	WAIT	DELTA	DELTA	DELTA	DELTA

λ	*ips*	*abor*	*allo*	*help*	*noop*
DELTA	$\mathfrak{n}$	*abor*	*allo*	*help*	*noop*
LAST	$\mathfrak{n}$	*abor*	*allo*	*help*	*noop*
WAIT	$\mathfrak{n}$	*abor*	*allo*	*help*	*noop*

Table 4.4: Stimulus-response specification of agent Q

$\circ$	*ips*	*abor*	*allo*	*help*	*noop*
SINCE_1	SINCE_1	SINCE_1	SINCE_1	SINCE_1	SINCE_1
SINCE_2	SINCE_2	SINCE_2	SINCE_2	SINCE_2	SINCE_2
SINCE_3	SINCE_3	SINCE_3	SINCE_3	SINCE_3	SINCE_3
SINCE_4	SINCE_4	SINCE_4	SINCE_4	SINCE_4	SINCE_4
READ	READ	READ	READ	READ	READ
AVG_1	AVG_1	AVG_1	AVG_1	AVG_1	AVG_1
AVG_2	AVG_2	AVG_2	AVG_2	AVG_2	AVG_2
AVG_3	AVG_3	AVG_3	AVG_3	AVG_3	AVG_3
AVG_4	AVG_4	AVG_4	AVG_4	AVG_4	AVG_4
RESET_1	RESET_1	RESET_1	RESET_1	RESET_1	RESET_1
RESET_2	RESET_2	RESET_2	RESET_2	RESET_2	RESET_2
RESET_3	RESET_3	RESET_3	RESET_3	RESET_3	RESET_3
RESET_4	RESET_4	RESET_4	RESET_4	RESET_4	RESET_4

λ	*ips*	*abor*	*allo*	*help*	*noop*
SINCE_1	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$
SINCE_2	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$
SINCE_3	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$
SINCE_4	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$
READ	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$
AVG_1	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$
AVG_2	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$
AVG_3	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$
AVG_4	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$
RESET_1	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$
RESET_2	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$
RESET_3	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$
RESET_4	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$

Table 4.5: Stimulus-response specification of agent R

$$
\begin{aligned}
\mathsf{C} &\mapsto \langle \text{TICK} \rangle \\
\mathsf{S} &\mapsto \langle \text{SEND}_{\text{ABOR}} + \text{SEND}_{\text{ALLO}} + \text{SEND}_{\text{HELP}} + \text{SEND}_{\text{NOOP}} \rangle \\
\mathsf{P} &\mapsto \langle \text{ABOR} \,;\, \text{COUNT}_1 + \text{ALLO} \,;\, \text{COUNT}_2 + \text{HELP} \,;\, \text{COUNT}_3 + \text{NOOP} \,;\, \text{COUNT}_4 \rangle \\
\mathsf{Q} &\mapsto \langle \text{DELTA} \,;\, \text{LAST} + \text{WAIT} \rangle \\
\mathsf{R} &\mapsto \langle \text{SINCE}_1 \,;\, \text{SINCE}_2 \,;\, \text{SINCE}_3 \,;\, \text{SINCE}_4 \,;\, \text{READ} \,;\, \big((\text{AVG}_1 \,;\, \text{RESET}_1) + \\
&\qquad (\text{AVG}_2 \,;\, \text{RESET}_2) + (\text{AVG}_3 \,;\, \text{RESET}_3) + (\text{AVG}_4 \,;\, \text{RESET}_4) \big) \rangle
\end{aligned}
$$

Figure 4.2: Abstract behaviour specifications of the agents in the running example system of communicating agents

Abstract Behaviour Specification

The second level of specification gives a specification of the abstract behaviour of each system agent. Whereas at the stimulus-response specification level, all possible responses to all external stimuli for each agent were specified, at the abstract behaviour specification level, the specification is restricted to the desired behaviour of an agent in the communicating system and it can be refined by computing the responses to the external stimuli that can be introduced into the system in the given context. The abstract behaviour specification for each agent behaviour is shown in Figure 4.2.

As a further example of this level of specification, consider the running example in a context where the system may only be influenced by the introduction of an *ips* stimulus since it is the only stimulus that an external agent, namely the idle prevention scheme, has control over. Additionally, for simplicity and illustrative purposes, assume that at this level of specification, only the behaviour of agent S composed with the behaviour of agent P is considered. This is to say that, the focus is pointed directly on this subset of the system behaviour. In the given context, the abstract behaviour of the composed agent $(\mathsf{S} \,;\, \mathsf{P})$ is specified as $\mathit{ips} \circ (\mathsf{S} \,;\, \mathsf{P})$. By systematic computation involving the use of Axiom (1) from Definition 4.2.3, the abstract behaviour of the composed agent $(\mathsf{S} \,;\, \mathsf{P})$ can be shown to be $(\text{SEND}_{\text{ABOR}} \,;\, \text{ABOR} \,;\, \text{COUNT}_1) + (\text{SEND}_{\text{ALLO}} \,;\, \text{ALLO} \,;\, \text{COUNT}_2) + (\text{SEND}_{\text{HELP}} \,;\, \text{HELP} \,;\, \text{COUNT}_3) +$

$(\text{SEND}_{\text{NOOP}} \,;\, \text{NOOP} \,;\, \text{COUNT}_4)$. This computation can also be automated using the prototype tool (see Appendix C.1). It is important to note that the specification of the composed agent $(\mathsf{S} \,;\, \mathsf{P})$ at the abstract behaviour specification level is a subset of the specification of the next behaviour and next stimulus mappings given at the stimulus-response specification level. Since agent S is free to choose which FTP command to issue in order to implement the idle prevention scheme, it is possible that when an *ips* stimulus is introduced to the system signalling that a command needs to be issued, agent S may issue any of the available FTP commands. Therefore, through the mathematics of C^{2}KA, it can be seen that the computed abstract behaviour of the composed agent $(\mathsf{S} \,;\, \mathsf{P})$ reflects each of the possibilities of choosing any particular FTP command. Furthermore, it can be observed that at the abstract behaviour specification level, the composed agent $(\mathsf{S} \,;\, \mathsf{P})$ has a more deterministic behaviour than that which is specified at the stimulus-response specification level.

At the abstract behaviour specification level, C^{2}KA can be viewed as an event-based model of communication. In C^{2}KA, the left $\mathcal{S}$-semimodule $\left({}_{\mathcal{S}}K, +\right)$ and the right $\mathcal{K}$-semimodule $\left(S_{\mathcal{K}}, \oplus\right)$ allow for the specification of how the external stimuli influence the behaviour of each agent in a given system. For this reason, this level of specification is best suited for describing message-passing communication where agents transfer information explicitly through the exchange of data structures, either synchronously or asynchronously.

Concrete Behaviour Specification

The concrete behaviour specification level involves providing the state-level specification of each agent behaviour. The state-level behaviours of agents are represented as programs which are defined over a set of events and that can be executed by the system. At this level, the concrete programs for each of the CKA terms which specify each agent behaviour are defined. In this thesis, the programs for each agent behaviour are specified using Dijkstra's guarded command language [Dij75]. The definition of Dijkstra's guarded command language is amended with two additional statements: $\mathsf{send}\ x$ and $\mathsf{receive}\ x$. The statement $\mathsf{send}\ x$,

where x is a program variable, denotes the sending of the variable x in a broadcast fashion where any agent that can receive the variable will do so. Similarly, the statement $\mathsf{receive}\ x$, where x is a program variable, denotes the receipt of the variable x. By using these two statements, explicit message-passing communication can be represented at the concrete behaviour specification level. Furthermore, the $\mathsf{send}\ x$ and $\mathsf{receive}\ x$ statements help to relate the external stimuli in the stimulus-response specification to the concrete behaviour specification for a given agent. For instance, there is a bijection between $\mathsf{send}\ x$ and x where $x \in S \backslash \{\mathfrak{n}, \mathfrak{d}\}$. Therefore, by extrapolation, it can be said that $\mathsf{send}\ x$ is a stimulus. Additionally, for all stimuli $x, y \in S \backslash \{\mathfrak{n}, \mathfrak{d}\}$ assume that $\mathsf{send}\ (x \oplus y) = \mathsf{send}\ x \oplus \mathsf{send}\ y$ and $\mathsf{send}\ (x \odot y) = \mathsf{send}\ x \odot \mathsf{send}\ y$. A similar assumption is made with respect to the $\mathsf{receive}\ x$ statement.

The concrete behaviour specifications of the agents in the running example are specified as programs in the extended Dijkstra's guarded command language as shown in Figures 4.3 to 4.7. For the running example, each external stimulus can be represented as a $\mathsf{send}\ x$ or $\mathsf{receive}\ x$ statement, along with the neutral stimulus $\mathfrak{n}$ and the deactivation stimulus $\mathfrak{d}$. For instance, the stimulus *abor* can be represented as send *abor* or as $\mathsf{receive}$ *abor*. The interpretation for the running example translates the next behaviour and next stimulus mappings as substitutions as shown in Figures 4.3 to 4.7.

Let S be generated using the operations of stimulus structures and the set $\{$*ips*, *abor*, *allo*, *help*, *noop*, $\mathfrak{n}$, $\mathfrak{d}\}$ and let C be a program such that:

$$\mathsf{C} \stackrel{\text{def}}{=} \texttt{time} := \texttt{time} + 1$$

and

$$\forall (x \mid x \in S \backslash \{\mathfrak{n}, \mathfrak{d}\} : (\mathsf{receive}\ x) \circ \mathsf{C} = \mathsf{C} \ \wedge\ \lambda\big((\mathsf{receive}\ x), \mathsf{C}\big) = \mathfrak{n}\,) \ \wedge$$
$$\mathfrak{n} \circ \mathsf{C} = \mathsf{C} \ \wedge\ \lambda\big(\mathfrak{n}, \mathsf{C}\big) = \mathfrak{n} \ \wedge\ \mathfrak{d} \circ \mathsf{C} = \mathsf{abort} \ \wedge\ \lambda\big(\mathfrak{d}, \mathsf{C}\big) = \mathfrak{d}$$

Figure 4.3: Concrete behaviour specification of agent C

Let S be generated using the operations of stimulus structures and the set $\{ips,\ abor,\ allo,\ help,\ noop,\ \mathfrak{n},\ \mathfrak{d}\}$, let S be a program, and let y be a variable in S such that:

$$\begin{array}{lll} \mathsf{S} \stackrel{\text{def}}{=} & \mathsf{if} & \mathsf{true} \longrightarrow y := \texttt{ABOR} \\ & [] & \mathsf{true} \longrightarrow y := \texttt{ALLO} \\ & [] & \mathsf{true} \longrightarrow y := \texttt{HELP} \\ & [] & \mathsf{true} \longrightarrow y := \texttt{NOOP} \\ & \mathsf{fi}; & \end{array}$$

and

$$\forall(x \mid x \in S\backslash\{\mathfrak{n},\mathfrak{d}\} : (\mathsf{receive}\ x) \circ \mathsf{S} = \mathsf{S} \wedge \lambda\big((\mathsf{receive}\ x), \mathsf{S}\big) = \mathsf{send}\ y) \wedge \mathfrak{n} \circ \mathsf{S} = \mathsf{S} \wedge \lambda\big(\mathfrak{n}, \mathsf{S}\big) = \mathfrak{n} \wedge \mathfrak{d} \circ \mathsf{S} = \mathsf{abort} \wedge \lambda\big(\mathfrak{d}, \mathsf{S}\big) = \mathfrak{d}$$

Figure 4.4: Concrete behaviour specification of agent S

Let S be generated using the operations of stimulus structures and the set $\{ips,\ abor,\ allo,\ help,\ noop,\ \mathfrak{n},\ \mathfrak{d}\}$, let P be a program, and let y be an expression variable in P such that:

$$\begin{array}{lll} \mathsf{P} \stackrel{\text{def}}{=} & \mathsf{if} & y >= \texttt{ABOR} \longrightarrow (\texttt{cmd} := 1;\ \texttt{num\_1} := \texttt{num\_1} + 1) \\ & [] & y >= \texttt{ALLO} \longrightarrow (\texttt{cmd} := 2;\ \texttt{num\_2} := \texttt{num\_2} + 1) \\ & [] & y >= \texttt{HELP} \longrightarrow (\texttt{cmd} := 3;\ \texttt{num\_3} := \texttt{num\_3} + 1) \\ & [] & y >= \texttt{NOOP} \longrightarrow (\texttt{cmd} := 4;\ \texttt{num\_4} := \texttt{num\_4} + 1) \\ & \mathsf{fi}; & \end{array}$$

and

$$\forall(x \mid x \in S\backslash\{\mathfrak{n},\mathfrak{d}\} : (\mathsf{receive}\ x) \circ \mathsf{P} = \mathsf{P}[y := x] \wedge \lambda\big((\mathsf{receive}\ x), \mathsf{P}\big) = \mathsf{send}\ x) \wedge \mathfrak{n} \circ \mathsf{P} = \mathsf{P} \wedge \lambda\big(\mathfrak{n}, \mathsf{P}\big) = \mathfrak{n} \wedge \mathfrak{d} \circ \mathsf{P} = \mathsf{abort} \wedge \lambda\big(\mathfrak{d}, \mathsf{P}\big) = \mathfrak{d}$$

Figure 4.5: Concrete behaviour specification of agent P

Let S be generated using the operations of stimulus structures and the set $\{ips, abor, allo, help, noop, \mathfrak{n}, \mathfrak{d}\}$, let Q be a program, and let y be an expression variable in Q such that:

$$\begin{aligned}\mathsf{Q} \stackrel{\text{def}}{=}\ & \mathsf{if} \quad (y >= \texttt{ABOR} \ \vee\ y >= \texttt{ALLO} \ \vee\ y >= \texttt{HELP} \ \vee\ y >= \texttt{NOOP}) \longrightarrow \\ & \qquad (\texttt{delta} := \texttt{time} - \texttt{last};\ \texttt{last} := \texttt{time}) \\ & [\!] \quad \neg(y >= \texttt{ABOR} \ \vee\ y >= \texttt{ALLO} \ \vee\ y >= \texttt{HELP} \ \vee\ y >= \texttt{NOOP}) \longrightarrow \mathsf{skip} \\ & \mathsf{fi};\end{aligned}$$

and

$$\forall(x \mid x \in S\backslash\{\mathfrak{n}, \mathfrak{d}\} : (\mathsf{receive}\ x) \circ \mathsf{Q} = \mathsf{Q}[y := x] \ \wedge\ \lambda\big((\mathsf{receive}\ x), \mathsf{Q}\big) = \mathsf{send}\ x\,) \ \wedge\ \mathfrak{n} \circ \mathsf{Q} = \mathsf{Q} \ \wedge\ \lambda\big(\mathfrak{n}, \mathsf{Q}\big) = \mathfrak{n} \ \wedge\ \mathfrak{d} \circ \mathsf{Q} = \mathsf{abort} \ \wedge\ \lambda\big(\mathfrak{d}, \mathsf{Q}\big) = \mathfrak{d}$$

Figure 4.6: Concrete behaviour specification of agent Q

Let S be generated using the operations of stimulus structures and the set $\{ips, abor, allo, help, noop, \mathfrak{n}, \mathfrak{d}\}$ and let R be a program such that:

$$\begin{aligned}\mathsf{R} \stackrel{\text{def}}{=}\ & \texttt{since\_1} := \texttt{since\_1} + \texttt{delta}; \\ & \texttt{since\_2} := \texttt{since\_2} + \texttt{delta}; \\ & \texttt{since\_3} := \texttt{since\_3} + \texttt{delta}; \\ & \texttt{since\_4} := \texttt{since\_4} + \texttt{delta}; \\ & n := \texttt{cmd}; \\ & \mathsf{if} \quad n = 1 \longrightarrow (\texttt{avg\_1} := \texttt{since\_1}/\texttt{num\_1};\ \texttt{since\_1} := 0) \\ & [\!] \quad n = 2 \longrightarrow (\texttt{avg\_2} := \texttt{since\_2}/\texttt{num\_2};\ \texttt{since\_2} := 0) \\ & [\!] \quad n = 3 \longrightarrow (\texttt{avg\_3} := \texttt{since\_3}/\texttt{num\_3};\ \texttt{since\_3} := 0) \\ & [\!] \quad n = 4 \longrightarrow (\texttt{avg\_4} := \texttt{since\_4}/\texttt{num\_4};\ \texttt{since\_4} := 0) \\ & \mathsf{fi}\end{aligned}$$

and

$$\forall(x \mid x \in S\backslash\{\mathfrak{n}, \mathfrak{d}\} : (\mathsf{receive}\ x) \circ \mathsf{R} = \mathsf{R} \ \wedge\ \lambda\big((\mathsf{receive}\ x), \mathsf{R}\big) = \mathfrak{n}\,) \ \wedge\ \mathfrak{n} \circ \mathsf{R} = \mathsf{R} \ \wedge\ \lambda\big(\mathfrak{n}, \mathsf{R}\big) = \mathfrak{n} \ \wedge\ \mathfrak{d} \circ \mathsf{R} = \mathsf{abort} \ \wedge\ \lambda\big(\mathfrak{d}, \mathsf{R}\big) = \mathfrak{d}$$

Figure 4.7: Concrete behaviour specification of agent R

When considering the concrete behaviour specification level, C^2KA can be viewed as a state-based model of communication. Since C^2KA extends concurrent Kleene algebra, it inherits this model of communication from CKA. Just as in CKA, the instantiation of a low-level model of programs and traces for C^2KA affords the ability to specify communication through shared events and the dependencies between them. This low-level model of programs and traces for C^2KA is exhibited through the specification of the agent behaviours as programs written using Dijkstra's guarded command language. Because of this, this level is specification is best suited for shared-variable communication where agents transfer information through a shared medium such as variables, memory locations, etc.

C^2KA provides a hybrid mathematical framework which is able to capture both the influence of external stimuli on agent behaviour as well the communication and concurrency of agents at the abstract algebraic level in systems of communicating agents. Depending on which level of specification if being considered, the model can be viewed as either event-based or state-based. This gives flexibility in allowing for the choice of the level of specification that is most suitable for the given problem and its context.

4.2.7 Orbits, Stabilisers, and Fixed Points in C^2KA

Orbits, stabilisers, and fixed points are notions that allow for the perception of a kind of topology of a system with respect to the stimulus-response relationships among the system agents. Because of this, some insight into the communication channels that can be established among system agents can be gained. This insight can aid in the analysis of the potential for communication condition for the existence of distributed covert channels (see Chapter 5). For example, C^2KA allows for the computation of the strong orbits (presented below) of the agent behaviours in a given system. The strong orbits represent the strongly connected agent behaviours in the system and therefore can provide some insight into the abilities of the agents in the same strong orbit to influence one another's behaviour through

communication. Furthermore, having an idea of the topology of the system allows for the abstraction of components of the overall system behaviour. This kind of abstraction can aid in separating the communicating and concurrent behaviour in a system and its environment. Moreover, computing the orbits and stabilisers of agent behaviours can aid in the analysis and verification of the existence of distributed covert channels in systems of communicating agents since it allows for modelling the possible reaction of a system to a stimulus. Also, in some cases, orbits allow for the analysis to be reduced to only some relevant orbits of a system. Similarly, stabilisers allow the analysis to be reduced to studying only the stimuli that influence the behaviour of an agent. It is conjectured that such reduction could, for example, alleviate the state explosion problem in model checking.

Since a C^{2}KA consists of two semimodules $\left({}_{\mathcal{S}}K, +\right)$ and $\left(S_{\mathcal{K}}, \oplus\right)$ for which there is a left $\mathcal{S}$-act ${}_{\mathcal{S}}K$ and a right $\mathcal{K}$-act $S_{\mathcal{K}}$, there are two complementary notions of orbits, stabilisers, and fixed points within the context of agent behaviours and external stimuli, respectively. In this way, one can use these notions to think about concurrent and communicating systems from two different perspectives, namely the behavioural perspective provided by the action of external stimuli on agent behaviours described by $\left({}_{\mathcal{S}}K, +\right)$ and the external event (stimulus) perspective provided by the action of agent behaviours on external stimuli described by $\left(S_{\mathcal{K}}, \oplus\right)$. This section focusses only on the treatment of these notions with respect to the left $\mathcal{S}$-semimodule $\left({}_{\mathcal{S}}K, +\right)$ and agent behaviours. In a very similar way, the same notions can be presented for the right $\mathcal{K}$-semimodule $\left(S_{\mathcal{K}}, \oplus\right)$ and external stimuli.

Definition 4.2.4 recalls the notions of orbits, stabilisers, and fixed points from the mathematical theory of monoids acting on sets [KKM00].

Definition 4.2.4. *Let* $\left({}_{\mathcal{S}}K, +\right)$ *be the unitary and zero-preserving left* $\mathcal{S}$*-semimodule of a* C^{2}KA *and let* $a \in K$.

(1) *The* orbit *of* a *in* $\mathcal{S}$ *is the set given by* $\mathrm{Orb}(a) = \{s \circ a \mid s \in S\}$.

(2) *The* strong orbit *of* a *in* $\mathcal{S}$ *is the set given by* $\mathrm{Orb}_{\mathrm{S}}(a) = \{b \in K \mid \mathrm{Orb}(b) = \mathrm{Orb}(a)\}$.

(3) *The* stabiliser *of a in $\mathcal{S}$ is the set given by* $\text{Stab}(a) = \{s \in S \mid s \circ a = a\}$.

(4) *An element $a \in K$ is called a* fixed point *if* $\forall(s \mid s \in S \backslash \{\mathfrak{d}\} : s \circ a = a)$. ■

A preorder on K can be defined as $a \preceq_{\mathcal{K}} b \iff \text{Orb}(a) \subseteq \text{Orb}(b)$. Given this preorder, an equivalence relation $\sim_{\mathcal{K}}$ can be obtained from the intersection of $\preceq_{\mathcal{K}}$ and $\succeq_{\mathcal{K}}$. The equivalence classes of $\sim_{\mathcal{K}}$ give the strong orbits [LPRR02]. The strong orbits can also be viewed as the strongly connected components of a directed graph [Ste10]. Additionally, when $a \in K$ is a fixed point, $\text{Orb}(a) = \{0, a\}$ and $\text{Stab}(a) = S \backslash \{\mathfrak{d}\}$. It is important to note that since $\left({}_{\mathcal{S}}K, +\right)$ is zero-preserving, every agent behaviour becomes inactive when subjected to the deactivation stimulus $\mathfrak{d}$. Due to this fact, this special case is excluded when discussing fixed point agent behaviours.

Before discussing the interplay between C^{2}KA and the notions of orbits, stabilisers, and fixed points, the partial order of sub-behaviours $\leq_{\mathcal{K}}$ is first extended to sets in order to express sets of agent behaviours encompassing one another.

Definition 4.2.5 (Encompassing Relation)**.** *Let $A, B \subseteq K$ be two subsets of agent behaviours. It is said that A* is encompassed by *B (or B* encompasses *A), written $A \lessdot_{\mathcal{K}} B$, if and only if* $\forall\big(a \mid a \in A : \exists(b \mid b \in B : a \leq_{\mathcal{K}} b)\big)$. ■

In essence, $A \lessdot_{\mathcal{K}} B$ indicates that every behaviour contained within the set A is a sub-behaviour of at least one behaviour in the set B. The encompassing relation $\lessdot_{\mathcal{S}}$ for external stimuli can be defined similarly.

Orbits

The orbit of an agent $a \in K$ represents the set of all possible behavioural responses from an agent behaving as a to any external stimulus from $\mathcal{S}$. In this way, the orbit of a given agent can be perceived as the set of all possible future behaviours for that agent. With regard to the specification of the running example given in Section 4.2.6, the orbits of each of the

agent behaviours can be computed. For instance, the orbits of the behaviours ABOR, READ, and COUNT_3 are given by $\text{Orb}(\text{ABOR}) = \{0, \text{ABOR}, \text{ALLO}, \text{HELP}, \text{NOOP}\}$, $\text{Orb}(\text{READ}) = \{0, \text{READ}\}$, and $\text{Orb}(\text{COUNT}_3) = \{0, \text{COUNT}_1, \text{COUNT}_2, \text{COUNT}_3, \text{COUNT}_4\}$, respectively. Proposition 4.2.3 provides an isotonicity law with respect to the orbits and the encompassing relation for agent behaviours.

Proposition 4.2.3. *Let* $\big(\mathcal{S}, \mathcal{K}\big)$ *be a* C^2KA. *Then,* $a \leq_{\mathcal{K}} b \implies \text{Orb}(a) \lessdot_{\mathcal{K}} \text{Orb}(b)$ *for all* $a, b \in K$.

Proof. The detailed proof can be found in Appendix A.4. □

A selection of additional properties follow immediately from Proposition 4.2.3 and are given in Corollary 4.2.4.

Corollary 4.2.4. *In a* C^2KA *the following laws hold for all* $a, b, c, d \in K$*:*

(1) $\text{Orb}(a) \lessdot_{\mathcal{K}} \text{Orb}(a + b)$

(2) $\text{Orb}((a * b) \,;\, (c * d)) \lessdot_{\mathcal{K}} \text{Orb}((a \,;\, c) * (b \,;\, d))$

(3) $\text{Orb}(a \,;\, b) \lessdot_{\mathcal{K}} \text{Orb}(a * b)$

(4) $\text{Orb}(a \,;\, b + b \,;\, a) \lessdot_{\mathcal{K}} \text{Orb}(a * b)$

(5) $\text{Orb}((a * b) \,;\, c) \lessdot_{\mathcal{K}} \text{Orb}(a * (b \,;\, c))$

(6) $\text{Orb}(a \,;\, (b * c)) \lessdot_{\mathcal{K}} \text{Orb}((a \,;\, b) * c)$

(7) $\text{Orb}(a^{\textcircled{:}}) \lessdot_{\mathcal{K}} \text{Orb}(a^{\circledast})$

(8) $\text{Orb}(a) \lessdot_{\mathcal{K}} \text{Orb}(c) \;\wedge\; \text{Orb}(b) \lessdot_{\mathcal{K}} \text{Orb}(c) \iff \text{Orb}(a) \cup \text{Orb}(b) \lessdot_{\mathcal{K}} \text{Orb}(c)$

Proof. The detailed proofs can be found in Appendix A.5. □

As stated before, without discussing the properties derived from the right $\mathcal{K}$-semimodule $\big(S_{\mathcal{K}}, \oplus\big)$, due to the cascading output law (see Definition 4.2.3 (2)), it is also the case that $\text{Orb}((s \circ a) \,;\, (\lambda(s, c) \circ b)) = \{0\}$ for any $(a \,;\, b) \in K$ and $\neg(a \leq_{\mathcal{K}} c) \;\wedge\; b \neq 1$.

Another Interpretation of Orbits

The influence of external stimuli on agent behaviours are called the *induced behaviours* via external stimuli. The notion of induced behaviours allows for some predictions to be made about the evolution of agent behaviours in a given system by providing some insight into the topology of the system and how different agents can respond to any external stimuli. Here, a formal treatment of the notion of induced behaviours is provided. While studying induced behaviours, a particular focus is placed on the next behaviour mapping $\circ$ and the effects of external stimuli on agent behaviours since there is an interest in examining the evolution of agent behaviours via the influence of external stimuli in a given system of communicating agents.

Definition 4.2.6 (Induced Behaviour)**.** *Let $a, b \in K$ be agent behaviours such that $a \neq b$. The behaviour b is said to be* induced by the behaviour a via external stimuli *(denoted by $a \lhd b$) if and only if* $\exists (s \mid s \in S : s \circ a = b)$. ■

Equivalently, this notion can be expressed as $a \lhd b \iff b \in \text{Orb}(a)$ for $a \neq b$. In this way, it can be seen that the orbit of a behaviour a represents the set of all behaviours which are induced by a via external stimuli. Considering the specification of the running example, it is plain to see, for instance, that $\text{ABOR} \lhd \text{ALLO}$ via the external stimulus *allo* and $\text{COUNT}_3 \lhd \text{COUNT}_4$ via the external stimulus *noop*.

Strong Orbits

Two agents are in the same strong orbit, denoted $a \sim_{\mathcal{K}} b$ for $a, b \in K$, if and only if their orbits are identical. This is to say when $a \sim_{\mathcal{K}} b$, if an agent behaving as a is influenced by an external stimulus to behave as b, then there exists an external stimulus which influences the agent, now behaving as b, to revert back to its previous behaviour a. Furthermore, if $a \sim_{\mathcal{K}} b$, then $\exists (s, t \mid s, t \in S : s \circ a = b \wedge t \circ b = a)$. In this case, the external stimuli s and t can be perceived as *inverses* (or *conjugates*) of one another and

allow an agent to revert back to its previous behaviour since $t \circ s \circ a = a$ and $s \circ t \circ b = b$ (i.e., $t \odot s \in \text{Stab}(a)$ and $s \odot t \in \text{Stab}(b)$). In this way, the notion of strong orbits presents a generalisation of the handshake found in many process calculi (e.g., [BK84, Hoa78a, Mil80, MPW92]). In process calculi, two primitive conjugate actions executing in parallel results in an unobservable communication action, denoted by τ. Rather than restricting to a view where handshaking can only occur between primitive actions, the notion of strong orbits shows how handshaking can occur between complex sequences of actions that result in the same kind of unobservable communication action. With regard to the specification of the running example, there are a number of strong orbits. A selection of these strong orbits are given by $\{\text{ABOR}, \text{ALLO}, \text{HELP}, \text{NOOP}\}$, $\{\text{READ}\}$, and $\{\text{COUNT}_1, \text{COUNT}_2, \text{COUNT}_3, \text{COUNT}_4\}$. As a specific example, the strong orbit represented by $\{\text{ABOR}, \text{ALLO}, \text{HELP}, \text{NOOP}\}$ indicates that $(\text{ABOR} \sim_{\mathcal{K}} \text{ALLO} \sim_{\mathcal{K}} \text{HELP} \sim_{\mathcal{K}} \text{NOOP})$.

Stabilisers

For any agent $a \in K$, the stabiliser of a represents the set of external stimuli which have no observable influence (or act as neutral stimuli) on the behaviour of an agent behaving as a. In the specification of the running example, the stabilisers of each of the agent behaviours can be computed. For example, $\text{Stab}(\text{ABOR})$ is generated by $\{s \odot \mathit{abor}, \mathit{ips}, \mathfrak{n} \mid s \in S \backslash \{\mathfrak{d}\}\}$, $\text{Stab}(\text{READ})$ is generated by $\{\mathit{abor}, \mathit{allo}, \mathit{help}, \mathit{ips}, \mathit{noop}, \mathfrak{n}\}$, and $\text{Stab}(\text{COUNT}_3)$ is generated by $\{s \odot \mathit{help}, \mathit{ips}, \mathfrak{n} \mid s \in S \backslash \{\mathfrak{d}\}\}$.

Proposition 4.2.5. *Let* $(\mathcal{S}, \mathcal{K})$ *be a* C^{2}KA. *Then,* $\text{Stab}(a) \cap \text{Stab}(b) \lessdot_{\mathcal{S}} \text{Stab}(a + b)$ *for all* $a, b \in K$.

Proof. The proof is straightforward by the definition of the encompassing relation $\lessdot_{\mathcal{S}}$ for external stimuli. The detailed proof can be found in Appendix A.6. □

However, consider a case where $\exists(s \mid s \in S : s \circ a = b \wedge s \circ b = a)$. Then, $s \notin \text{Stab}(a)$ and $s \notin \text{Stab}(b)$ but $s \in \text{Stab}(a+b)$. Therefore, it is easy to see that in general $\neg\big(\text{Stab}(a+b) \lessdot_{\mathcal{S}} \big(\text{Stab}(a) \cap \text{Stab}(b)\big)\big)$ and $\neg\big(\text{Stab}(a+b) \lessdot_{\mathcal{S}} \big(\text{Stab}(a) \cup \text{Stab}(b)\big)\big)$.

Fixed Points

Depending on the given specification of a system of communicating agents, there may be any number of fixed points with respect to the next behaviour mapping $\circ$. When an agent behaviour is a fixed point, it is not influenced by any external stimulus other than the deactivation stimulus $\mathfrak{d}$. With regard to the specification of the running example, it is easy to see, for instance, that the behaviour READ for agent R and the behaviour TICK for agent C, are fixed points, while the behaviours ABOR and COUNT_3 for agent P are not fixed points. The existence of fixed point behaviours is important when considering how agents can communicate via external stimuli. For instance, an agent that has a fixed point behaviour, does not have any observable response to any external stimuli (except for the deactivation stimulus) and therefore it can be seen that such an agent cannot be a receiver in any sort of communication via external stimuli.

Proposition 4.2.6 gives a selection of properties regarding fixed agent behaviours.

Proposition 4.2.6. *Let* $\big(\mathcal{S}, \mathcal{K}\big)$ *be a* C^2KA *and let* $a, b \in K$ *such that* a *and* b *are fixed points. Then:*

(1) 0 *is a fixed point*

(2) $a + b$ *is a fixed point*

(3) $a \,;\, b$ *is a fixed point*

(4) $a^{\textcircled{:}}$ *is a fixed point*

Proof. The proofs each use Definition 4.2.4(4). The proof for (1) is straightforward from the axiomatisation of C^2KA. The proof for (2) involves Definition 3.1.8(1) for $\big({}_{\mathcal{S}}K, +\big)$ and the proof for (3) uses Definition 4.2.3(1). The proof for (4) uses Proposition 4.2.1(5), the application of (3), Definition 4.2.3(4), and the definition of $a^{\textcircled{:}}$. The detailed proofs are given in Appendix A.7. □

In Proposition 4.2.6, Identity (1) states that the inactive agent 0 is a fixed point with respect to the next behaviour mapping $\circ$. In this way, the inactive agent is not influenced by any external stimulus. Similarly, it is easy to see that the deactivation stimulus $\mathfrak{d}$ is a fixed point with respect to the next stimulus mapping λ if the notion of a fixed point is considered in terms of external stimuli. Identities (2), (3), and (4) state that the choice, sequential composition, and sequential iteration of fixed point behaviours results in a fixed point behaviour, respectively. In general, even if $a, b \in K$ are both fixed points, nothing can be said about $(a * b)$ as a fixed point.

Perceiving the Topology of a System of Communicating Agents

As mentioned earlier, orbits, stabilisers, and fixed points allow for the perception of a kind of topology of a system which can be beneficial for analysing systems of communicating agents for the existence of distributed covert channels, particularly with respect to reasoning about the potential for communication amongst agents. Proposition 4.2.7 provides further insight into how the topology of a system of communicating agents can be perceived using $\mathrm{C}^2\mathrm{KA}$ and the notions of fixed points, strong orbits, and induced behaviours.

Proposition 4.2.7. *Let $a, b, c \in K$ be agent behaviours.*

(1) a *is a fixed point* $\implies \forall (b \mid b \in K \ \wedge \ b \neq 0 \ \wedge \ b \neq a \ : \ \neg(a \lhd b))$

(2) $a \sim_{\mathcal{K}} b \implies a \lhd b \ \wedge \ b \lhd a$

(3) $a \sim_{\mathcal{K}} b \implies (a \lhd c \iff b \lhd c)$

Proof. The proof for (1) follows straightforwardly from Definition 4.2.6 and the definition of the orbit of a fixed point. The proof for (2) is straightforward from Definition 4.2.6 and the definition of $\sim_{\mathcal{K}}$. The proof for (3) involves the shunting rule, the definition of $\sim_{\mathcal{K}}$, and Definition 4.2.6. The detailed proofs can be found in Appendix A.8. □

In Proposition 4.2.7, Identity (1) states that if an agent has a fixed point behaviour, then it does not induce any agent behaviours via external stimuli besides the inactive behaviour 0. This is a direct consequence of the fact that an agent with a fixed point behaviour is not influenced by any external stimuli (except for the deactivation stimulus $\mathfrak{d}$) and therefore remains behaving as it is. Identity (2) states that all agent behaviours which belong to the same strong orbit are mutually induced via some (possibly different) external stimuli. This is to say that if two agent behaviours are in the same strong orbit, then there exists inverse stimuli for each agent behaviour in a strong orbit allowing an agent to revert back to its previous behaviour. Finally, Identity (3) states that if two agent behaviours are in the same strong orbit, then a third behaviour can be induced via external stimuli by either of the behaviours within the strong orbit. This is to say that each behaviour in a strong orbit can induce the same set of behaviours (perhaps via different external stimuli). Therefore, the strong orbit to which these behaviours belong can be abstracted and perceived as an equivalent behaviour with respect to the behaviours which it can induce via external stimuli.

4.2.8 Specifying Agent Behaviour Using a Prototype Tool

A prototype tool has been implemented in order to support the automated analysis of systems of communicating agents for the existence of distributed covert channels, particularly, in terms of the potential for communication condition. The tool is implemented using the functional programming language *Haskell* and makes use of the Maude term rewriting system [CDE+03] for supporting the computation and rewriting of systems of communicating agents specified using C^2KA.

The prototype tool consists of two main components. The first component implements the C^2KA core and allows for the specification of the behaviour of agents in systems of communicating agents using C^2KA. It also supports the automated computation of orbits, strong orbits, stabilisers, and fixed points as described in Section 4.2.7. The second component allows for the automated verification of the satisfaction of the potential for communication

condition for a given system of communicating agents by implementing the formulation of the potential for communication condition presented later in Chapter 5. If the tool determines that there exists a potential for communication between two agents, it provides a list of all of the possible communication paths or patterns of communication between those agents. This section focusses only on the first component that allows for the behaviour of agents in a system of communicating agents to be specified.

Figure 4.8 shows the uses hierarchy of the C^{2}KA component of the prototype tool. It consists of 33 modules. The *Behaviour* and *Stimulus* modules implement the representation of terms of a CKA and a stimulus structure, respectively. The *NextBehaviour* and *NextStimulus* modules implement the representation of terms of the left $\mathcal{S}$-semimodule $\left({}_{\mathcal{S}}K, +\right)$ and the right $\mathcal{K}$-semimodule $\left(S_{\mathcal{K}}, \oplus\right)$ and allow for the specification of the next behaviour and next stimulus mappings, respectively. The *LevelThreeSpec* module provides the representation of Dijkstra's guarded command language that is used in the concrete behaviour specification of agents. The *Agent* module provides the representation of agents following the three levels of specification provided by the C^{2}KA framework and provides functions for loading agent specifications from files. The *SoCA* module provides the implementation of a system of communicating agents and provides functions for adding new agents to a system and establishing the sets of basic agent behaviours and basic external stimuli for a given system specification. Each of the above mentioned modules consists of three sub-modules containing the associated types, parsers, and printers. The *Orbits*, *StrongOrbits*, *Stabilisers*, and *FixedPoints* modules provide implementations of orbits, strong orbits, stabilisers, and fixed points, with respect to both the next behaviour mapping $\circ$ and the next stimulus mapping λ. Finally, the *MaudeInterface* module provides the interface to the Maude term rewriting system.

A detailed usage of the prototype tool for specifying and analysing the agent behaviours in the running example of the system of communicating agents described in Section 4.1 is provided in Appendix C.

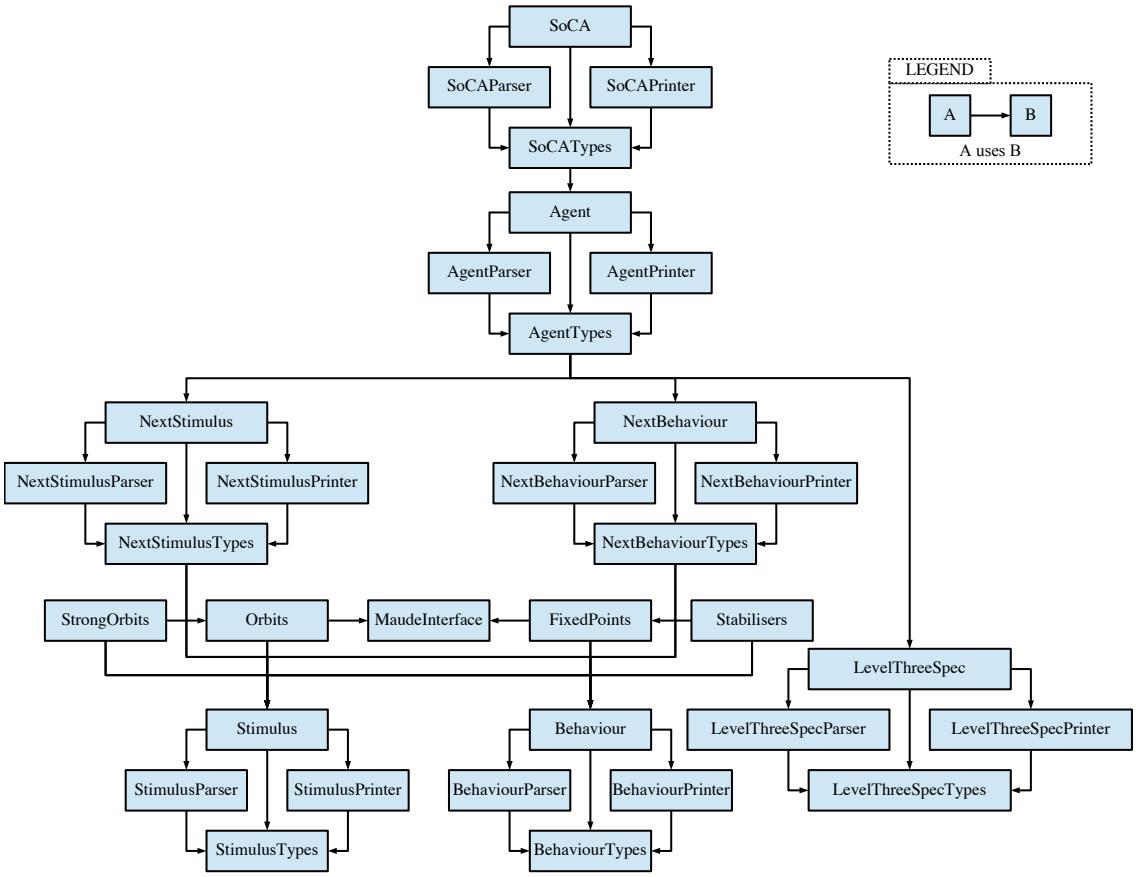

Figure 4.8: The uses hierarchy of the C^{2}KA component of the prototype tool

4.2.9 Discussion and Related Work

C^2KA offers an algebraic setting which can capture both the influence of external stimuli on agent behaviour, as well as the communication and concurrency of agents at the abstract algebraic level. It uses notions from classical algebra to extend the algebraic foundation provided by CKA. As mentioned earlier, by considering a C^2KA with a trivial stimulus structure (i.e., $S = \{\mathfrak{n}, \mathfrak{d}\}$), any system will have two orbits with respect to the next behaviour mapping $\circ$, namely $\{0\}$ and K. By merging these two orbits, the set K is obtained and therefore it becomes easy to see that the C^2KA reduces to a CKA.

In the past, communication has been studied in process calculi such as CCS [Mil80], CSP [Hoa78a], ACP [BK84], and π-calculus [MPW92]. As discussed in [HMSW09a, HMSW10, HMSW11], some analogies can be made relating CKA with process calculi. Therefore, by considering the case where a C^2KA has a trivial stimulus structure, then the same kind of analogies relating C^2KA with existing process calculi can be made.

In [HMSW09a, HMSW09b, HMSW10, HMSW11], Hoare et al. have taken steps towards investigating some aspects of communication through the derivation of rules for a simplified rely/guarantee calculus [Jon81] using CKA. However, this kind of communication is only captured by examining shared events and the dependencies between them. Since the proposed framework provides an extension of CKA, it is also capable of achieving these results. Furthermore, C^2KA supports the ability to work in either a state-based model (as illustrated in Figures 4.3 to 4.7) or an event-based model (as illustrated by Tables 4.1 to 4.5 and Figure 4.2) for the specification of concurrent and communicating systems. It offers the ability to separate the communicating and concurrent behaviour in a system and its environment. This separation of concerns allows for the consideration of the influence of stimuli from the world in which an agent resides as transformations of agent behaviours and yields the three levels of specification offered by C^2KA and described in Section 4.2.6. With these levels of specification, C^2KA is able to capture the notions of message-passing

communication and shared-variable communication consistent with the hybrid view of agent communication depicted in Figure 1.1 in Chapter 1. Specifically, the abstract behaviour specification level is interested only in the behaviour of an agent as dictated by the stimulus-response relationships that exist in the given system. In this way, the behaviour of an agent is dictated by its responses to external stimuli without the need to articulate the internal state-based system of each behaviour. On the other hand, by instantiating a concrete model of agent behaviour, such as that of programs and traces similar to what is done with CKA [HMSW09a, HMSW09b, HMSW10, HMSW11] at the concrete behaviour specification level, then the state-based model of agent behaviour can be defined. In this way, if a given problem requires insight into how external stimuli are processed by an agent, the concrete behaviour specification level affords the ability to specify such internal states of agent behaviours in terms of programs on concrete state variables. Because of this, C^2KA is flexible in allowing the context of the given problem to dictate which level of abstraction is most suitable. For example, if the given problem need not worry about the internal states of agent behaviours, then the system can be specified at the stimulus-response specification or abstract behaviour specification level without any modifications to the proposed framework. Moreover, C^2KA inherits the algebraic foundation of CKA with most, if not all, of its models and theory. It does not establish a new foundation, but instead builds atop well-established ones. All of this inherited theory provides the power and flexibility to specify all that can be done with existing formalisms while allowing for expansion beyond existing limitations. C^2KA provides a framework which presents a different view of communication and concurrency than what is traditionally given by existing formalisms and allows for the intricacies of distributed covert channels to be captured.

4.3 Specifying Agent Knowledge

In this thesis, the formalism that is used to specify agent knowledge in systems of communicating agents is the description logic $\mathcal{ALB}$ (see Section 3.5 for more information on $\mathcal{ALB}$). This section presents a representation of agent knowledge using $\mathcal{ALB}$ [HS00]. This representation of agent knowledge gives the power and flexibility to reason on agent knowledge in systems of communicating agents at both the terminological or conceptual level and at the assertional or object level.

4.3.1 Specifying Agent Knowledge using the Description Logic $\mathcal{ALB}$

Each agent A in a given system of communicating agents has a knowledge base, denoted by $\mathcal{N}_{\mathsf{A}} = \big(\mathcal{T}_{\mathsf{A}}, \mathcal{A}_{\mathsf{A}}\big)$. Each agent in the system begins with some *initial knowledge base* denoted by $\mathcal{N}_{\mathsf{A}}^{0}$. The initial knowledge base represents the specification of the knowledge of each agent before the system begins any kind of operation or function. The knowledge base of each agent captures all of the information that the agent knows, or can come to know through reasoning. The knowledge base of an agent evolves as information is inserted or updated through communication.

Specifying the Description Logic Signature

Recall that a signature $\big(N_C, N_R, N_O\big)$ specifies three disjoint sets of concept symbols, role symbols, and object symbols, respectively. When specifying the knowledge of an agent using the description logic $\mathcal{ALB}$, assume that there is a common signature for a system formed by a set $\mathcal{C}$ of communicating agents. The knowledge base of each agent $\mathsf{A} \in \mathcal{C}$ is specified with respect to this common signature.

Consider the running example described in Section 4.1. The signature for the system of communicating agents is given by $\big(N_C, N_R, N_O\big)$ as shown in Figure 4.9. In this signature,

$$
\begin{aligned}
N_C &= \{\mathsf{Command}, \mathsf{ConfInfo}, \mathsf{BitString}, \mathsf{Enumeration}\} \\
N_R &= \{\mathsf{CmdToEnum}, \mathsf{EnumToCmd}, \mathsf{Variable}\} \\
N_O &= \{\mathsf{ABOR}, \mathsf{ALLO}, \mathsf{HELP}, \mathsf{NOOP}, \mathsf{00}, \mathsf{01}, \mathsf{10}, \mathsf{11}, \mathsf{CMD}, \mathsf{DELTA}, \mathsf{LAST}, \\
&\quad \mathsf{TIME}, \mathsf{NUM\_1}, \mathsf{NUM\_2}, \mathsf{NUM\_3}, \mathsf{NUM\_4}, \mathsf{SINCE\_1}, \mathsf{SINCE\_2}, \\
&\quad \mathsf{SINCE\_3}, \mathsf{SINCE\_4}, \mathsf{AVG\_1}, \mathsf{AVG\_2}, \mathsf{AVG\_3}, \mathsf{AVG\_4}\} \cup \mathbb{N}
\end{aligned}
$$

Figure 4.9: The $\mathcal{ALB}$ signature for the example system of communicating agents

there are concepts pertaining to the commands available in the system, the notions of bit-strings and enumerations, and confidential information. Additionally, there a roles which denote mappings from commands to enumerations and from enumerations to commands, as well as those which denote variables in the system. Finally, there are objects representing the actual FTP commands available in the given system, the possible bit-strings that may be used in the system, and the names of all of the variables that exist in the given system. Additionally, the set of all natural numbers $\mathbb{N}$ is included as object symbols so that they may denote enumerations or values for variables.

Specifying Agent Knowledge Bases

In order to specify the knowledge base of each agent in a system of communicating agents, a definition of the TBox and the ABox must be given with respect to a given signature. Once again, consider the system of communicating agents described in Section 4.1. The description logic specifications for the initial knowledge base of each agent in the given system of communicating agents with respect to the signature $\big(N_C, N_R, N_O\big)$ are shown in Figures 4.10 to 4.14. To simplify the presentation of the specifications of the initial knowledge of each agent, assume the following sets of assertional axioms:

$$
\begin{aligned}
\text{CMDS} &= \{\mathsf{Command}(\mathsf{ABOR}), \mathsf{Command}(\mathsf{ALLO}), \mathsf{Command}(\mathsf{HELP}), \mathsf{Command}(\mathsf{NOOP})\} \\
\text{ENUMS} &= \{\mathsf{Enumeration}(1), \mathsf{Enumeration}(2), \mathsf{Enumeration}(3), \mathsf{Enumeration}(4)\} \\
\text{BITS} &= \{\mathsf{BitString}(00), \mathsf{BitString}(01), \mathsf{BitString}(10), \mathsf{BitString}(11)\} \\
\text{NUMS} &= \{\mathsf{Variable}(\mathsf{NUM\_1}, 0), \mathsf{Variable}(\mathsf{NUM\_2}, 0), \\
&\quad \mathsf{Variable}(\mathsf{NUM\_3}, 0), \mathsf{Variable}(\mathsf{NUM\_4}, 0)\} \\
\text{SINCES} &= \{\mathsf{Variable}(\mathsf{SINCE\_1}, 0), \mathsf{Variable}(\mathsf{SINCE\_2}, 0), \\
&\quad \mathsf{Variable}(\mathsf{SINCE\_3}, 0), \mathsf{Variable}(\mathsf{SINCE\_4}, 0)\} \\
\text{AVGS} &= \{\mathsf{Variable}(\mathsf{AVG\_1}, 0), \mathsf{Variable}(\mathsf{AVG\_2}, 0), \\
&\quad \mathsf{Variable}(\mathsf{AVG\_3}, 0), \mathsf{Variable}(\mathsf{AVG\_4}, 0)\} \\
\text{MAPS} &= \{\mathsf{CmdToEnum}(\mathsf{ABOR}, 1), \mathsf{CmdToEnum}(\mathsf{ALLO}, 2), \\
&\quad \mathsf{CmdToEnum}(\mathsf{HELP}, 3), \mathsf{CmdToEnum}(\mathsf{NOOP}, 4)\} \\
\text{VARS} &= \{\mathsf{Variable}(\mathsf{TIME}, 0), \mathsf{Variable}(\mathsf{DELTA}, 0)\}
\end{aligned}
$$

Additionally, assume that there is a notion of memory and naming consistency for shared variables. This means that variables with the same name have the same interpretation (i.e., they point to the same memory location). Under this assumption, for two distinct agents A and B in the same system of communicating agents, if agent A has an assertional axiom $\mathsf{Variable}(\mathsf{VAR}, \mathsf{X})$ in its ABox and agent B has an assertional axiom $\mathsf{Variable}(\mathsf{VAR}, \mathsf{Y})$ in its ABox, then it is the case that $\mathsf{X} = \mathsf{Y}$.

$$
\begin{aligned}
\mathcal{N}_{\mathsf{C}}^{0} &= (\mathcal{T}_{\mathsf{C}}^{0}, \mathcal{A}_{\mathsf{C}}^{0}) \text{ where} \\
\\
\mathcal{T}_{\mathsf{C}}^{0} &= \emptyset \\
\mathcal{A}_{\mathsf{C}}^{0} &= \{\mathsf{Variable}(\mathsf{TIME}, 0)\}
\end{aligned}
$$

Figure 4.10: Initial knowledge base specification of agent C

$$
\begin{aligned}
\mathcal{N}_{\mathsf{S}}^{0} &= (\mathcal{T}_{\mathsf{S}}^{0}, \mathcal{A}_{\mathsf{S}}^{0}) \text{ where} \\
\\
\mathcal{T}_{\mathsf{S}}^{0} &= \left\{ \mathsf{ConfInfo} \sqsubseteq \mathsf{BitString}, \mathsf{EnumToCmd} \equiv \mathsf{CmdToEnum}^{\smile} \right\} \\
\mathcal{A}_{\mathsf{S}}^{0} &= \text{CMDS} \cup \text{ENUMS} \cup \text{BITS} \cup \text{AVGS} \cup \text{MAPS} \cup \text{VARS} \cup \\
&\quad \left\{ \mathsf{ConfInfo}\big(01\big) \right\}
\end{aligned}
$$

Figure 4.11: Initial knowledge base specification of agent S

$$
\begin{aligned}
\mathcal{N}_{\mathsf{P}}^{0} &= (\mathcal{T}_{\mathsf{P}}^{0}, \mathcal{A}_{\mathsf{P}}^{0}) \text{ where} \\
\\
\mathcal{T}_{\mathsf{P}}^{0} &= \left\{ \mathsf{EnumToCmd} \equiv \mathsf{CmdToEnum}^{\smile} \right\} \\
\mathcal{A}_{\mathsf{P}}^{0} &= \text{CMDS} \cup \text{ENUMS} \cup \text{BITS} \cup \text{NUMS} \cup \text{AVGS} \cup \text{MAPS} \cup \text{VARS} \cup \\
&\quad \left\{ \mathsf{Variable}\big(\mathsf{CMD}, 0\big) \right\}
\end{aligned}
$$

Figure 4.12: Initial knowledge base specification of agent P

$$
\begin{aligned}
\mathcal{N}_{\mathsf{Q}}^{0} &= (\mathcal{T}_{\mathsf{Q}}^{0}, \mathcal{A}_{\mathsf{Q}}^{0}) \text{ where} \\
\\
\mathcal{T}_{\mathsf{Q}}^{0} &= \emptyset \\
\mathcal{A}_{\mathsf{Q}}^{0} &= \text{CMDS} \cup \text{NUMS} \cup \text{AVGS} \cup \text{VARS} \cup \\
&\quad \left\{ \mathsf{Variable}\big(\mathsf{CMD}, 0\big), \mathsf{Variable}\big(\mathsf{LAST}, 0\big) \right\}
\end{aligned}
$$

Figure 4.13: Initial knowledge base specification of agent Q

$$
\begin{aligned}
\mathcal{N}_{\mathsf{R}}^{0} &= (\mathcal{T}_{\mathsf{R}}^{0}, \mathcal{A}_{\mathsf{R}}^{0}) \text{ where} \\
\\
\mathcal{T}_{\mathsf{R}}^{0} &= \left\{ \mathsf{EnumToCmd} \equiv \mathsf{CmdToEnum}^{\smile} \right\} \\
\mathcal{A}_{\mathsf{R}}^{0} &= \text{CMDS} \cup \text{ENUMS} \cup \text{BITS} \cup \text{NUMS} \cup \text{SINCES} \cup \text{AVGS} \cup \\
&\quad \text{MAPS} \cup \text{VARS} \cup \left\{ \mathsf{Variable}\big(\mathsf{CMD}, 0\big) \right\}
\end{aligned}
$$

Figure 4.14: Initial knowledge base specification of agent R

With respect to the initial knowledge specification of each agent in the given system of communicating agents, the terminological and assertional axioms that are identical in the initial knowledges of agents C, S, P, Q, and R constitute the *shared knowledge* about the system. For example, each agent, except for the clock agent C, initially knows the set of FTP commands that are available in the system. Additionally, agents S, P, and R share the terminological axiom $\mathsf{EnumToCmd} \equiv \mathsf{CmdToEnum}^{\smile}$ which states that the role EnumToCmd is equivalent to the relational converse of the role CmdToEnum, and the assertional axioms that define the mapping from commands to enumerations. The sharing of this knowledge results from the fact that, in the given example, agents S and R know the enumeration mapping that agent P is configured with. Furthermore, there are shared variables in the system. For instance, from the specification of the initial knowledge of each agent, it is easy to see that the variable `cmd` is shared amongst agents P, Q, and R and the variable `time` is shared amongst all of the agents in the system. Lastly, it can be seen that only agent S knows that all confidential information are bit-strings (i.e., $\mathsf{ConfInfo} \sqsubseteq \mathsf{BitString}$). No other agent has any initial knowledge of the confidential information in the system.

4.3.2 Specifying Agent Knowledge Using the SPASS Theorem Prover

The SPASS theorem prover [TST14] is used to support the automated reasoning on agent knowledge specified using the description logic $\mathcal{ALB}$. SPASS is an automated theorem prover

for first-order logic with equality and it additionally supports modal logics and description logics. SPASS is based on the description logic $\mathcal{ALB}$ and implements a resolution-based decision procedure for reasoning on TBoxes and ABoxes.

In this thesis, SPASS is currently used to specify the knowledge base of each agent in a given system of communicating agents. Using these knowledge base specifications, the satisfaction of $\mathcal{ALB}$ formulae can be verified. This provides the needed support for automating the verification of the constraint on communication condition in systems of communicating agents presented later in Chapter 6. A detailed usage of the SPASS theorem prover for specifying and reasoning on agent knowledge for the system of communicating agents described in Section 4.1 is provided in Appendix D.

4.3.3 Discussion and Related Work

In the literature, there have been many proposed formalisms for capturing the knowledge of system agents. For instance, agent knowledge has been previously described using various logical formalisms such as epistemic logic and its variants (e.g., [MvdH04]), including dynamic epistemic logic (e.g., [vDvdHK07]), and logics of communication graphs [PP05, PP07]. Additionally, there have been other formalisms for representing agent knowledge such as information algebra [KS07], among others. While each of these formalisms allow for the specification of agent knowledge, they each have drawbacks such as requiring each agent to "speak the same language" or being restricted to a static view of the system of agents, as is the case for epistemic logics and the logic of communication graphs. Moreover, some formalisms such as information algebra, introduce large and complex algebraic structures which leads to much overhead in terms of comprehensibility. Instead, the representation of agent knowledge presented in this section is based on the description logic $\mathcal{ALB}$ since it is decidable and extendable allowing for the representation of various communication schemes and scenarios. This point will be further articulated and emphasised in Chapter 6. Furthermore, the representation of agent knowledge presented in this section allows for reasoning

at both the terminological or conceptual level and at the assertional or object level which is desirable when considering agent knowledge in the context of establishing and operating distributed covert channels.

4.4 Conclusion

In order to analyse a system of communicating agents for the existence of distributed covert channels, it must first be formally specified. This chapter has presented a means for specifying systems of communicating agents by providing ways in which the behaviour and knowledge of each agent in the system can be represented. First, the mathematical framework of Communicating Concurrent Kleene Algebra (C^2KA) was introduced as a means for specifying the concurrent and communicating behaviour of agents in a system. Using the running example, the use of C^2KA was demonstrated. Additionally, the use of a prototype tool for automating the specification of systems of communicating agents using C^2KA was discussed. Then, a representation of agent knowledge based on the description logic $\mathcal{ALB}$ was presented. Again, using the running example, the specification of the knowledge of each agent in the system of communicating agents was shown. Also, the use of the SPASS theorem prover for supporting the automated specification and reasoning on agent knowledge was discussed.

The formal specification of the behaviour and knowledge of agents presented in this chapter will be used later in this thesis, most notably in Chapter 5 and in Chapter 6 as the basis for the formulation and verification of the potential for communication condition and constraint on communication condition for the existence of distributed covert channels, respectively.

Chapter 5

Agent Behaviour and Potential for Communication

If a covert channel exists in a system of communicating agents, then the covert channel users must have the ability to communicate with one another, either directly or indirectly. This chapter presents the formulation and verification of the potential for communication condition for the existence of distributed covert channels based on the study of agent behaviour in systems of communicating agents. Section 5.1 discusses the potential for communication as found in the literature and motivates the potential for communication as a necessary condition for the existence of distributed covert channels. Section 5.2 presents a formulation of the potential for communication condition based on the mathematical framework of C^2KA as defined in Section 4.2. Section 5.3 demonstrates the verification of the potential for communication condition using the running example described in Section 4.1 and specified in Section 4.2.6. Section 5.5 discusses the proposed formulation and verification of the potential for communication condition along with related work. Finally, Section 5.6 provides concluding remarks.

5.1 The Potential for Communication Condition for the Existence of Distributed Covert Channels

In the literature, a study of the potential for communication among system agents has been an important part of existing work in detecting and preventing the existence and use of covert channels. This section examines the potential for communication as it is found in the literature and motivates the need for the potential for communication among agents in a system of communicating agents as a necessary condition for the existence of distributed covert channels. Lastly, this section informally states the potential for communication condition for the existence of distributed covert channels.

5.1.1 Potential for Communication in the Literature

When it comes to the potential for communication between system agents in terms of covert channel existence, the literature shows a variety of conditions which point to the necessity for covert channel users to be able to communicate either directly or indirectly.

Existence of Shared Resources

A condition existing in many sets of covert channel existence conditions found in the literature is related to the existence of shared resources. For both storage and timing channels, the condition states, "The sending and receiving processes must have access to the same attribute of a shared resource" [Kem83]. Similar conditions are given in [SC99, WJG+04, WL05b]. In [SC99], the condition is alternatively phrased in terms of the existence of a global variable rather than the existence of a shared resource. In the case of two agents (i.e., direct communication), this condition is required. However, it is not required in the general case. Take, for instance, a sending agent S and a receiving agent R. It is possible that the sending agent and receiving agent can communicate through a set of proxy agents P and Q, for example. In this case, the sending agent S can communicate with the proxy

agent P, which may react to the communication from the sending agent S by deterministically conveying a message to the proxy agent Q, which in turn deterministically writes to a memory location that the receiving agent R can read. Then, the sending agent S and the receiving agent R do not share any resources but can still communicate, provided that the sending agent S knows the deterministic mechanisms used by the proxy agents P and Q. The knowledge of such mechanisms is not uncommon if the proxy agents P and Q are servers with deterministic behaviours. It is not necessary that the sending agent S knows which memory location the proxy agent Q writes to. Instead, the sending agent S simply needs to know that the proxy agent Q writes to a memory location that the receiving agent R can read. This is an example of indirect communication between agents S and R. This sort of indirect communication can be imagined to exist between any arbitrary number of proxies.

Ability to Alter and Observe Changes in Shared Environments

Another condition which can be found in many existing sets of conditions for covert channel existence is the ability for a sending agent to alter a shared resource in such a way that the receiving agent can detect or observe the alteration. Often this condition is broken into two conditions detailing the existence of a sending mechanism for the sending agent and the existence of a receiving mechanism for the receiving agent. According to [Kem83], for storage channels, "There must be some means by which the sending process can force the shared attribute to change" and "There must be some means by which the receiving process can detect the attribute change". Again, in [SC99], similar conditions are given in terms of global variables rather than shared resources. In [McH95], a condition which attempts to express the same spirit as those given in [Kem83] can be found. The condition states, "There must be an effective procedure for exploiting a security flaw to form a channel for transmitting a useful quantity of information from the sending process to the receiving process in a timely manner." However, it is not necessary that a security flaw be exploited in order to form a communication channel. Covert channels are designed in such a way that

they intentionally bypass the security policy of the system. Therefore, the formation of a communication channel can be done by simply modifying the behaviour of agents so that they are able to use the system in unintended ways. Furthermore, this condition carries a significant amount of ambiguity. For instance, it is not clear what is meant by "an effective procedure for exploiting a security flaw" or by the ability to transmit "a useful quantity of information in a timely manner." With such ambiguity, one cannot expect to adequately determine whether such a condition is satisfied.

A condensed set of conditions which must be satisfied for covert channel existence is given in [WL05b]. The conditions read "If the sender is able to invoke change(s) in the visible space of the receiver, a covert channel may exist" and "If the sender is able to change when an object is updated relative to the observation made by the receiver, a covert channel may exist." These conditions essentially reduce to the sender being able to change a shared resource such that the receiver is able to observe the change. The first condition roughly corresponds to storage channels, where the sender and receiver are communicating through a shared object and the second condition roughly corresponds to timing channels, where the sender and receiver are communicating using a shared clock. The issue with such compact conditions is that clarity is lost since a single condition encompasses a number of simpler, more easily verifiable conditions which are equally concise. These simpler conditions are hidden behind vague language and complicated terminology, such as the term "visible space" for example. While conciseness is strongly preferred when articulating the conditions for covert channel existence, it should not come at the cost of clarity and expressiveness.

Distinguishing Covert Timing Channels

In [Kem83], a distinction is made between storage and timing channels. This distinction translates to the sets of existence conditions. The ability for a sending agent to alter a shared resource in such a way that the receiving agent can detect or observe the alteration is recast in terms of clocks and response times. According to [Kem83], for a timing channel to exist,

"The sending and receiving processes must have access to a time reference such as a real-time clock" and "The sender must be capable of modulating the receiver's response time for detecting a change in the shared attribute". An important observation about the conditions for the existence of a timing channel is that they imply that either the sending agent or the receiving agent has the ability to change the value of the shared attribute in order to implement the timing channel. However, there is no need to have separate conditions for covert storage channels and covert timing channels. It is well known that any time two agents share a processor, there exists a shared resource among the processes, namely a common clock or time reference. Actually, covert timing channels require two clocks: a reference clock (usually a system clock) to measure the absolute time and another clock to be modulated by the sending agent and observed by the receiving agent [Wra91]. Therefore, a common clock can be viewed simply as a shared attribute of the shared processor and the conditions for storage channels and timing channels can be consolidated into one set of conditions. This extends from the suggestion that there is no fundamental distinction between storage and timing channels [ZAB07b].

Ability to Correctly Synchronise and Sequence Events

Another condition for covert channel existence found in the literature is related to the correct synchronisation and sequencing of events so that communication between the sending and receiving agents can take place. According to [SC99], "The sender and the receiver must be able to synchronise their operations so that information flow can take place." A similar articulation of this condition, ensuring the correct order of communication events between the sending agent and the receiving agent, can be found in [WJG+04]. Also, in [Kem83], for both timing and storage channels, the condition states, "There must be some mechanism for initiating the communication between the sending and receiving processes and for sequencing the events correctly. This mechanism could be another channel with a smaller bandwidth." One of the inherent issues with the above condition is that one needs to devise

a scenario that satisfies the condition. This requires insight and imagination into the system being analysed which may not be a trivial exercise. A condition that does not require such heuristics in order to verify its satisfaction for a given system of communicating agents is desired. It can be seen that the collective behaviour of the agents in a system presents a pattern of communication that reflects the possibility of sequencing events. This pattern of communication is a study of the behaviour of the agents in the system and not a study of the available resources.

Potential to Communicate

The conditions found in [Kem83] have been rephrased and condensed in [SC99, TGC87, Sid03] by establishing conditions pertaining to the potential for communication. According to [SC99], "The sender and receiver of the covert channel have the potential to communicate in the system." In conjunction with the other conditions presented in [SC99, TGC87, Sid03], the above condition captures the idea that, if there is a sending agent acting as the source of information and a receiving agent acting as the sink, such that if information can flow from the source to the sink, then there is a potential for communication. However, this is not entirely clear from the condition presented as it is. For instance, it is not clear what exactly is meant by "potential to communicate." Instead, the inference of the existence of an information flow from the sender to the receiver is required.

After examining conditions pertaining to the potential for communication found in the literature, it is clear that there is ambiguity and a lack of consistent terminology. Because of this, it is difficult to verify the satisfaction of such conditions. A re-articulation and formalisation of the existing conditions concerning the potential for communication is required. This can help to improve the current understanding of covert channels and serve as a basis for developing effective and efficient mechanisms for mitigating covert channels in systems of communicating agents.

5.1.2 The Potential for Communication Condition

The *potential for communication* condition is introduced as one of the two necessary conditions for the existence of distributed covert channels in [JKZ12]. It is a rephrasing of the existing conditions found in the literature. This is done to ensure that the condition is clear, verifiable, and captures the ability to establish an information flow from a sending agent to a receiving agent in a system of communicating agents. The condition reads:

> If there exists an agent acting as a source of information and an agent acting as an information sink, such that the source and sink agents are different, and if there exists a pattern of communication allowing for information to transfer from the source to the sink through the synchronisation and sequencing of events, then the source and sink agents have a potential for communication.

The *potential for communication* condition captures how information can be exchanged and communicated by the agents in the system through a study of their behaviour. In systems of communicating agents, the synchronisation of system events can allow for the creation of event sequences, or "patterns of communication", that allow information to flow from one agent to another. In essence, the potential for communication condition indicates that as long as it is possible for information to flow from one agent to another, a communication channel can be established.

5.2 Formulating the Potential for Communication Condition

This section proposes a formulation of the potential for communication condition for the existence of distributed covert channels in systems of communicating agents. The proposed formulation is based on the mathematical framework of C^2KA that was presented in Section 4.2.4. In the following subsections, the potential for communication is examined from two complementary perspectives, namely the external stimuli perspective and the shared

environment perspective, consistent with the hybrid view of agent communication presented in Section 1.1.1 (see Figure 1.1).

5.2.1 Formulating Potential for Communication via External Stimuli

An examination of the potential for communication in a system of communicating agents from the perspective of external stimuli involves an investigation into the interactions of the agents. In a given system of communicating agents, each agent is subjected to each external stimulus. This means that when an agent generates a stimulus, it is broadcasted to all other agents and a response is invoked. This means that every agent in the system responds to each stimulus that is issued. However, it is not the case that the behaviour of each agent will be influenced by each stimulus. It is said that *communication via external stimuli* has taken place only when a stimulus that is generated by an agent influences (i.e., does not fix) the behaviour of another agent. With this view of stimuli, it is possible that more than one agent is influenced by the generation of the same stimulus by another agent in the system.

Consider a system formed by a set $\mathcal{C}$ of communicating agents with $\mathsf{A}, \mathsf{B} \in \mathcal{C}$ such that $\mathsf{A} \neq \mathsf{B}$.

Definition 5.2.1 (Potential for Direct Communication via External Stimuli)**.** *Agent* $\mathsf{A} \mapsto \langle a \rangle$ *is said to have the* potential for direct communication via external stimuli *with agent* $\mathsf{B} \mapsto \langle b \rangle$ *(denoted by* $\mathsf{A} \rightarrow_{\mathcal{S}} \mathsf{B}$*) if and only if* $\exists \big(s, t \mid s, t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s, a) : t \circ b \neq b \big)$ *where* S_b *is the set of all basic stimuli.* ■

If there exists a basic sub-stimulus that is generated by agent A that influences the behaviour of agent B, then there is a potential for direct communication via external stimuli from agent A to agent B. The existence of a basic stimulus $t \leq_{\mathcal{S}} \lambda(s, a)$ that does not fix the behaviour of B is required since it is possible that $\lambda(s, a) \circ b = b$. As an example, take $\mathsf{A} \mapsto \langle a \rangle$ and $\mathsf{B} \mapsto \langle b \rangle$ where $b = c + d$ and $\lambda(s, a) = s_1 \oplus s_2$ such that $s_1 \circ b = c + d$ and $s_2 \circ b = c$. Then, since $s_2 \leq_{\mathcal{S}} \lambda(s, a)$ and $s_2 \circ b \neq b$, it is the case that $A \rightarrow_{\mathcal{S}} B$.

Definition 5.2.2 (Potential for Communication via External Stimuli Using at Most n Basic Stimuli)**.** *Agent* A *is said to have the* potential for communication via external stimuli with agent B using at most n basic stimuli *(denoted by* $\mathsf{A} \rightarrow_{\mathcal{S}}^{n} \mathsf{B}$*) if and only if* $\exists\big(\mathsf{C} \mid \mathsf{C} \in \mathcal{C} \wedge \mathsf{C} \neq \mathsf{A} \wedge \mathsf{C} \neq \mathsf{B} : \mathsf{A} \rightarrow_{\mathcal{S}}^{(n-1)} \mathsf{C} \wedge \mathsf{C} \rightarrow_{\mathcal{S}} \mathsf{B}\big)$. ■

Definition 5.2.3 (Potential for Communication via External Stimuli)**.** *Agent* A *is said to have the* potential for communication via external stimuli *with agent* B *(denoted by* $\mathsf{A} \rightarrow_{\mathcal{S}}^{+} \mathsf{B}$*) if and only if* $\exists\big(n \mid n \geq 1 : \mathsf{A} \rightarrow_{\mathcal{S}}^{n} \mathsf{B}\big)$. ■

When $\mathsf{A} \rightarrow_{\mathcal{S}}^{+} \mathsf{B}$, there is a sequence of external stimuli (of arbitrary length; basic or composite) which allows for information to be transferred from agent A to agent B in the system of communicating agents.

Stimuli-Connected Systems of Communicating Agents

Two subsets X_1 and X_2 of $\mathcal{C}$ form a partition of $\mathcal{C}$ if and only if $X_1 \cap X_2 = \emptyset$ and $X_1 \cup X_2 = \mathcal{C}$.

Definition 5.2.4 (Stimuli-Connected)**.** *A system formed by a set* $\mathcal{C}$ *of communicating agents is said to be* stimuli-connected *if and only if for every* X_1 *and* X_2 *that form a partition of* $\mathcal{C}$*, it is the case that* $\exists(\mathsf{A}, \mathsf{B} \mid \mathsf{A} \in X_1 \wedge \mathsf{B} \in X_2 : \mathsf{A} \rightarrow_{\mathcal{S}}^{+} \mathsf{B} \vee \mathsf{B} \rightarrow_{\mathcal{S}}^{+} \mathsf{A})$. *Otherwise, the system is said to be* stimuli-disconnected. ■

In a stimuli-connected system, every agent is a participant, either as the source or sink, of at least one direct communication via external stimuli. For example, it is easy to verify that the system of communicating agents described in Section 4.1 is stimuli-disconnected. This verification can be done automatically using the prototype tool (see Appendix C.3).

Communication Fixed Points

The notion of an agent as a communication fixed point is important in terms of the potential for communication via external stimuli.

Definition 5.2.5 (Communication Fixed Point)**.** *An agent* $\mathsf{A} \in \mathcal{C}$ *is said to be a* communication fixed point *if and only if* $\forall\big(\mathsf{B} \mid \mathsf{B} \in \mathcal{C}\backslash\{\mathsf{A}\} : \neg(\mathsf{A} \rightarrow_{\mathcal{S}}^{+} \mathsf{B})\big)$. ■

Obviously, a communication fixed point does not have the potential for communication via external stimuli with any other agent. Thus, it is plain to see that an agent $\mathsf{A} \mapsto \langle 0 \rangle$ is a communication fixed point since for all $s \in S$, $\lambda(s, 0) = \mathfrak{d}$ and since $\mathfrak{d}$ is not a basic stimulus, it cannot have the potential for communication via external stimuli with any other agent. Additionally, if $\mathsf{A} \rightarrow_{\mathcal{S}}^{+} \mathsf{B}$, then the potential communication path from agent A to agent B contains at most one communication fixed point that is agent B. Consider the running example from Section 4.1. Agents C and R are communication fixed points, while agents S, P, and Q are not communication fixed points.

Universally Influential Agents

A universally influential agent is the dual of a communication fixed point.

Definition 5.2.6 (Universally Influential)**.** *An agent* $\mathsf{A} \in \mathcal{C}$ *is said to be* universally influential *if and only if* $\forall\big(\mathsf{B} \mid \mathsf{B} \in \mathcal{C}\backslash\{\mathsf{A}\} : \mathsf{A} \rightarrow_{\mathcal{S}}^{+} \mathsf{B}\big)$. ■

A universally influential agent is able to generate some stimulus that influences the behaviour, either directly or indirectly, of each other agent in the system. In this way, it is obvious that a communication fixed point cannot be universally influential. Again, considering the running example given in Section 4.1, there is no agent that is universally influential.

Proposition 5.2.1. *A system of communicating agents that contains a universally influential agent is stimuli-connected.*

Proof. Assume that a system formed by a set $\mathcal{C}$ of communicating agents is a stimuli-disconnected system and let agent $\mathsf{C} \in \mathcal{C}$ be universally influential. Then, using the definition of a stimuli-disconnected system (see Definition 5.2.4), instantiation with $\mathsf{B} = \mathsf{C}$,

and Definition 5.2.6, it is the case that either the system is stimuli-connected or agent C is not universally influential which is a contradiction to the assumption that the system is stimuli-disconnected and agent C is universally influential. The detailed proof can be found in Appendix A.9. □

Proposition 5.2.1 shows that the existence of a universally influential agent in a system of communicating agents yields a stimuli-connected system.

Proposition 5.2.2. *Let* $\mathsf{A} \mapsto \langle a \rangle$ *be an agent such that* a *is a fixed point behaviour. Then, there does not exist an agent* B *that has the potential for communication via external stimuli with agent* A*.*

Proof. The proof is straightforward using Definition 5.2.1. □

Proposition 5.2.2 states that no agent has the potential for communication via external stimuli with an agent that has a fixed point behaviour. This is due to the fact that if an agent has a fixed point behaviour, then it is not influenced by any external stimuli and therefore communication with that agent via external stimuli is not possible.

Proposition 5.2.3. *Let* $\mathsf{A} \mapsto \langle a \rangle$*,* $\mathsf{B} \mapsto \langle b \rangle$*, and* $\mathsf{C} \mapsto \langle c \rangle$ *be agents in a system formed by a set* $\mathcal{C}$ *of communicating agents.*

(1) *If* $\mathsf{B} \rightarrow_{\mathcal{S}} \mathsf{C}$ *then* $(\mathsf{A} + \mathsf{B}) \rightarrow_{\mathcal{S}} \mathsf{C}$*.*

(2) *If* $\mathsf{A} \rightarrow_{\mathcal{S}} \mathsf{B}$ *then* $\mathsf{A} \rightarrow_{\mathcal{S}} (\mathsf{B} + \mathsf{C})$ *if* $\forall\big(s, t \mid s, t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s, a) : \neg(t \circ b \leq_{\mathcal{K}} b + c \wedge t \circ c \leq_{\mathcal{K}} b + c)\big)$*.*

Proof. The proof of (1) uses Definition 5.2.1, the distributivity of λ over $+$, the definition of $\leq_{\mathcal{S}}$, and the fact that $\oplus$ is left-isotone with respect to $\leq_{\mathcal{S}}$. The proof of (2) involves Definition 5.2.1, involves monotonic $\exists$-body, anti-monotonic $\neg$, distributivity of $\circ$ over $+$, and substitution of $=$ by $=$. The detailed proofs can be found in Appendix A.10. □

Proposition 5.2.3 shows how the potential for communication via external stimuli can be preserved when non-determinism is introduced among agents. Specifically, Identity (1) states that when non-determinism is added at the source of a potential communication path via external stimuli, the potential for communication via external stimuli is always preserved. Intuitively, this is the case since there can always be a sub-stimulus generated by the source which results from agent B that can preserve the potential for communication via external stimuli with agent C. On the other hand, Identity (2) states that when non-determinism is added at the sink of a potential communication path via external stimuli, the potential for communication is preserved only if there does not exist any basic stimulus that is generated by the source that influences agent B and agent C to behave as a sub-behaviour of agent $\mathsf{B} + \mathsf{C}$. This condition ensures that agent $\mathsf{B} + \mathsf{C}$ cannot have a fixed point behaviour. If the non-determinism that is introduced causes a fixed point behaviour, then there will no longer be any potential for communication as stated by Proposition 5.2.2.

5.2.2 Formulating Potential for Communication via Shared Environments

The examination of communication via shared environments, either through shared variables, resources, or functionalities, has been the topic of study for a number of existing techniques for covert channel and information flow analysis (e.g., [Kem83, KP91, SKJ09b, SC99, WJG+04, WL05b]). When formulating the potential for communication via shared environments, the focus is centred on finding whether a particular agent has the ability to alter an element of the environment that it shares with a neighbouring agent such that the neighbouring agent is able to observe the alteration that was made.

Since the proposed formulation is based on $\mathrm{C}^2\mathrm{KA}$ which is an extension of CKA, the mechanisms provided by CKA are used to formulate the potential for communication via shared environments. Similar to what is done with existing information flow techniques for formulating the potential for communication via shared environments, the formulation studies the dependencies between events that are shared amongst system agents.

In what follows, consider the aggregation algebra $\big(K, +\big)$ (see Definition 3.2.2) where K is the set of agent behaviours from the CKA and $+$ is the choice between agent behaviours from the CKA, and let $a, b \in K$. Furthermore, let R be a *dependence relation* on $\big(K, +\big)$ (see Definition 3.2.3) where $a \,\mathrm{R}\, b$ denotes that the behaviour b depends on the behaviour a. Such a dependence relation may be a definition-reference relation between program variables in the specifications of agent behaviours. Assume that $\neg(a \,\mathrm{R}\, 0)$ and $\neg(0 \,\mathrm{R}\, a)$ and $\neg(a \,\mathrm{R}\, 1)$ and $\neg(1 \,\mathrm{R}\, a)$ for every $a \in K$. These are rather natural assumptions since the inactive and idle behaviours depend on nothing and nothing depends on them.
For the purpose of this formulation, consider a system formed by a set $\mathcal{C}$ of communicating agents and assume that a dependence relation R is given and let A, B such that $\mathsf{A} \neq \mathsf{B}$.

Definition 5.2.7 (Potential for Direct Communication via Shared Environments)**.** *Agent* $\mathsf{A} \mapsto \langle a \rangle$ *is said to have the* potential for direct communication via shared environments *with agent* $\mathsf{B} \mapsto \langle b \rangle$ *(denoted by* $\mathsf{A} \to_{\mathcal{E}} \mathsf{B}$*) if and only if* $a \,\mathrm{R}\, b$*.* ■

Definition 5.2.8 (Potential for Communication via Shared Environments)**.** *Agent* A *is said to have the* potential for communication via shared environments *with agent* B *(denoted by* $\mathsf{A} \to_{\mathcal{E}}^{+} \mathsf{B}$*) if and only if* $a \,\mathrm{R}^{+}\, b$ *where* R^{+} *is the transitive closure of the given dependence relation.* ■

This means that if two agents respect the given dependence relation, then there is a potential for communication via shared environments.

Proposition 5.2.4. *Assume a system formed by a set* $\mathcal{C}$ *of communicating agents and let* $\mathsf{A}, \mathsf{B}, \mathsf{C} \in \mathcal{C}$*.*

(1) *If* $\mathsf{B} \to_{\mathcal{E}} \mathsf{C}$ *then* $(\mathsf{A} + \mathsf{B}) \to_{\mathcal{E}} \mathsf{C}$.

(2) *If* $\mathsf{A} \to_{\mathcal{E}} \mathsf{B}$ *then* $\mathsf{A} \to_{\mathcal{E}} (\mathsf{B} + \mathsf{C})$.

Proof. The proofs are straightforward from Definition 5.2.7 and the bilinearity of the dependence relation R. □

Proposition 5.2.4 shows that the potential for communication via shared environments is preserved when non-determinism is introduced at the source or the sink of a potential communication path via shared environments. If there exists a dependency between two agent behaviours a and b, then given a choice between b and any other behaviours, it is possible to choose to behave as b in order to preserve the dependency. While this is not always the case, it is important to note that the focus is placed on the identification of the potential for communication, which means that if it is possible for an agent to choose a behaviour which yields the potential for communication, then in general the potential for communication exists.

5.2.3 A Formulation of the Potential for Communication Condition

By combining the definitions of potential for communication via external stimuli and via shared environments, a formulation of the potential for communication condition for the existence of distributed covert channels is obtained. In what follows, consider a system formed by a set $\mathcal{C}$ of communicating agents and let $\mathsf{A}, \mathsf{B} \in \mathcal{C}$ such that $\mathsf{A} \neq \mathsf{B}$.

Definition 5.2.9 (Potential for Direct Communication)**.** *Agent* A *is said to have the* potential for direct communication *with agent* B *(denoted by* $\mathsf{A} \rightsquigarrow \mathsf{B}$*) if and only if* $\mathsf{A} \rightarrow_{\mathcal{S}} \mathsf{B} \ \vee \ \mathsf{A} \rightarrow_{\mathcal{E}} \mathsf{B}$. ■

Definition 5.2.10 (Potential for Communication)**.** *Agent* A *is said to have the* potential for communication *with agent* B *(denoted by* $\mathsf{A} \rightsquigarrow^{+} \mathsf{B}$*) if and only if* $\mathsf{A} \rightsquigarrow \mathsf{B} \ \vee \ \exists\big(\mathsf{C} \mid \mathsf{C} \in \mathcal{C} : \mathsf{A} \rightsquigarrow \mathsf{C} \ \wedge \ \mathsf{C} \rightsquigarrow^{+} \mathsf{B}\big)$. ■

For a given system of communicating agents, if there exists a sequence of agents, starting with a source agent A and ending on a sink agent B, that have the potential for direct communication either via external stimuli or via shared environments, then agent A has the potential for communication with agent B.

With the above formulation of the potential for communication condition for the existence of distributed covert channels, the way in which the covert information is communicated is abstracted away. This is to say that, the formulation does not require any insight into *how* the covert information is being transmitted from the source to the sink. For instance, whether there exists a protocol-based covert channel where covert information is embedded into existing network packets for example, or whether there exists an environment-based covert channel where covert information is transmitted by modulating the timing of events, the verification of the potential for communication condition remains the same. This means that there is no need to verify different conditions for different classes of covert channels. This is because the given formulation of the potential for communication condition rests simply on the ability for one agent in a given system of communicating agents to influence the behaviour of another agent, or on the ability for one agent to alter a shared environment that can be observed by another agent. In this way, the covert communication scheme that is used by the covert channel users is irrelevant in this analysis. The notion of communication schemes is discussed further in Section 6.1.

5.3 Verifying the Potential for Communication Condition

Given a system of communicating agents, the verification of the potential for communication condition for the existence of distributed covert channels for any two agents follows directly from the application of the definitions given in Section 5.2.

Consider the running example system of communicating agents described in Section 4.1. Figure 5.3 shows a visualisation of the potential for communication amongst the agents in the system. For instance, it is easy to see that $(\mathsf{S} \rightarrow_{\mathcal{S}} \mathsf{P})$ and $\neg(\mathsf{P} \rightarrow_{\mathcal{S}} \mathsf{R})$ by Definition 5.2.1 and $(\mathsf{S} \rightarrow^{+}_{\mathcal{S}} \mathsf{Q})$ and $\neg(\mathsf{S} \rightarrow^{+}_{\mathcal{S}} \mathsf{R})$ by Definition 5.2.3. Similarly, it is the case that $(\mathsf{P} \rightarrow_{\mathcal{E}} \mathsf{R})$ and $\neg(\mathsf{C} \rightarrow_{\mathcal{E}} \mathsf{P})$ by Definition 5.2.7 and $(\mathsf{C} \rightarrow^{+}_{\mathcal{E}} \mathsf{R})$ and $\neg(\mathsf{Q} \rightarrow_{\mathcal{E}} \mathsf{P})$ by Definition 5.2.7. Finally, by the application of Definition 5.2.9, it is the case that $(\mathsf{S} \rightsquigarrow \mathsf{P})$ and $(\mathsf{P} \rightsquigarrow \mathsf{R})$ and by the application of Definition 5.2.10, it can be shown that $(\mathsf{S} \rightsquigarrow^{+} \mathsf{R})$ and $\neg(\mathsf{C} \rightsquigarrow^{+} \mathsf{S})$.

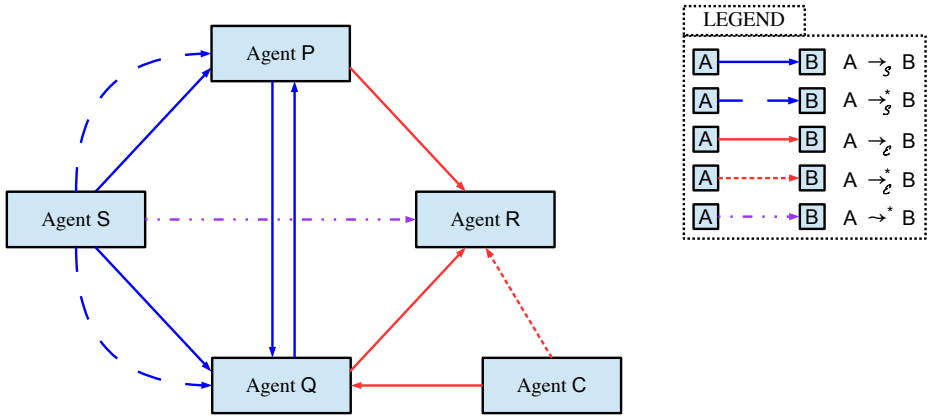

Figure 5.1: A visualisation of the potential for communication for the running example system of communicating agents

Now that it has been shown that the potential for communication condition is satisfied in the running example system of communicating agents (particularly from agent S to agent R), it remains to be verified whether the constraint on communication condition is satisfied as well. The formulation and verification of the constraint on communication condition is touched upon in Section 6.4.

5.3.1 Verifying the Potential for Communication Condition Using the Prototype Tool

The C^2KA component of the prototype tool was described in Section 4.2.8. This section describes the potential for communication component of the prototype tool. The potential for communication component allows for the automated evaluation of the satisfaction of the potential for communication condition for a given system of communicating agents by implementing the formulation of the potential for communication condition presented in Section 5.2. If the tool determines that there exists a potential for communication between two agents, it provides a list of all of the possible communication paths or patterns of communication between those agents.

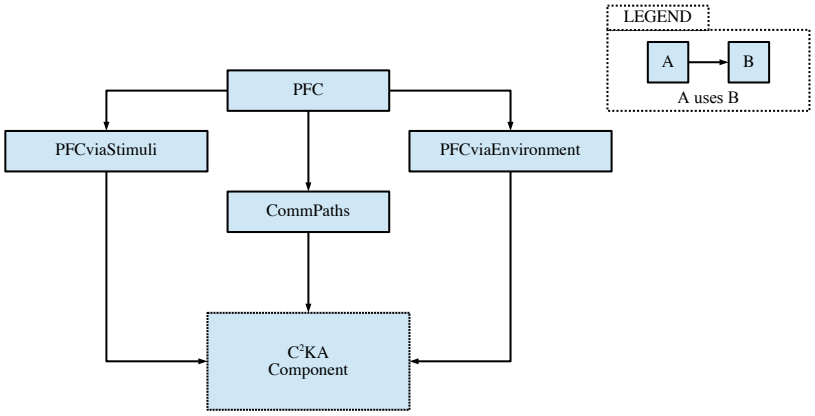

Figure 5.2: The uses hierarchy of the potential for communication component of the prototype tool

Figure 5.2 shows the uses hierarchy of the potential for communication component of the prototype tool. It consists of 4 modules. The *CommPaths* module implements the functionality for determining and displaying the potential communication paths between two agents in a given system of communicating agents. The *PFCviaStimuli* module implements the formulation of the potential for direct communication via external stimuli presented in Section 5.2.1. Similarly, the *PFCviaEnvironment* module implements the formulation of the potential for direct communication via shared environments presented in Section 5.2.2. Finally, the *PFC* module implements the formulation of the potential for communication condition for the existence of distributed covert channels presented in Section 5.2.3. It also provides functions for verifying whether a given system of communicating agents is stimuli-connected and whether a given agent in a system of communicating agents is a communication fixed point, or whether it is universally influential.

A detailed usage of the prototype tool for verifying the potential for communication condition for the running example of the system of communicating agents described in Section 4.1 is provided in Appendix C.4.

5.4 Modifying Agent Behaviours to Preserve or Disrupt the Potential for Communication

In a system of communicating agents, agent behaviours can be modified in many different ways with various effects on the potential for communication between agents. A useful result showing the effects of modifying the behaviour of an agent participating in a direct communication is given in Proposition 5.4.1.

Proposition 5.4.1. *Let* $\mathsf{A} \mapsto \langle a \rangle$, $\mathsf{B} \mapsto \langle b \rangle$, *and* $\mathsf{C} \mapsto \langle c \rangle$ *be agents in a system formed by a set* $\mathcal{C}$ *of communicating agents.*

(1) *If* $\mathsf{A} \rightsquigarrow \mathsf{B}$, *then* $\mathsf{A} \rightsquigarrow (\mathsf{C}\,;\mathsf{B})$ *if* $\forall\big(s,t \mid s,t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s,a) : t \circ c \neq c \vee \lambda(t,c) \circ b \neq b\big) \vee a\,\mathrm{R}\,(c\,;b)$.

(2) *If* $\mathsf{A} \rightsquigarrow \mathsf{B}$, *then* $(\mathsf{A}\,;\mathsf{C}) \rightsquigarrow \mathsf{B}$ *if* $\forall\big(s,t \mid s,t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s,a) : \lambda(t,c) = t\big) \vee (a\,;c)\,\mathrm{R}\,b$.

Proof. The proofs both use Definitions 5.2.1, 5.2.7, and 5.2.9. The proof of (1) also involves Definition 4.2.3(1), distributivity of $\vee$ over $\exists$, monotonic $\exists$-body, anti-monotonic $\neg$, and substitution of $=$ by $=$. The proof of (2) also uses Definition 3.1.8(3) for $\big(S_{\mathcal{K}}, \oplus\big)$. The detailed proofs can be found in Appendix A.11. □

Proposition 5.4.1 identifies the conditions constraining the modifications that can be made to the source or sink agent involved in a direct potential for communication . When considering the direct potential for communication, besides completely replacing the behaviour of one of the agents with the behaviour of another agent, a new agent behaviour can be sequentially composed either on the left of a sink agent or on the right of a source agent. Specifically, Identity (1) shows how the sequential composition of an additional behaviour on the left of a sink agent will not affect the potential for communication provided that every stimulus that is generated by the source agent either does not fix the behaviour of the first

component of the sequential composition, or causes the first component of the sequential composition to generate a stimulus that does not fix the behaviour of the second component of the sequential composition. Additionally, the potential for communication will not be affected provided that the composed behaviour preserves the dependency relation. Similarly, Identity (2) shows how the sequential composition of an additional behaviour on the right of a source agent will not affect the potential for communication provided that the additional behaviour fixes every stimulus that it receives from the source agent or that the composed behaviour preserves the dependency relation. These results are particularly useful since behaviours that satisfy these constraints can be constructed to de-couple the direct potential for communication between two agents. For example, consider the case when the composed agent behaviour is a fixed point behaviour. Then, in particular, this behaviour can be used to filter the stimuli that are received by the sink agent and to ensure that only potential for communication via shared environments is possible between the source agent and the sink agent.

Another useful result showing the effects of modifying the behaviour of an agent in the sequence of a potential communication path or pattern of communication between two agents is given in Proposition 5.4.2. In particular, Proposition 5.4.2 deals with the indirect potential for communication between agents, unlike Proposition 5.4.1 which dealt only with direct potential for communication .

Proposition 5.4.2. *Let* $\mathsf{A} \rightsquigarrow^{+} \mathsf{B}$ *such that* $\exists\big(\mathsf{C} \mid \mathsf{C} \in \mathcal{C} : \mathsf{A} \rightsquigarrow \mathsf{C} \wedge \mathsf{C} \rightsquigarrow \mathsf{B}\big)$ *where* $\mathsf{A} \mapsto \langle a \rangle$, $\mathsf{B} \mapsto \langle b \rangle$, *and* $\mathsf{C} \mapsto \langle c \rangle$. *Let* R *be the given dependence relation. Suppose* C *is replaced by another agent* $\mathsf{C}' \mapsto \langle c' \rangle$. *Then,*

(1) *If* $c' = (c\,;d)$, *then* $\mathsf{A} \rightsquigarrow^{+} \mathsf{B}$ *if* $\forall\big(s,t \mid s,t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s,c) : \lambda(t,d) = t\big) \vee (c\,;d)\,\mathrm{R}\,b$.

(2) *If* $c' = (c + d)$, *then* $\mathsf{A} \rightsquigarrow^{+} \mathsf{B}$ *if* $\forall\big(s,t \mid s,t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s,a) : \neg(t \circ c \leq_{\mathcal{K}} c + d \wedge t \circ d \leq_{\mathcal{K}} c + d)\big)$.

(3) *If* $c' = c^{\circlearrowleft}$, *then* $\mathsf{A} \rightsquigarrow^{+} \mathsf{B}$.

(4) *If* $c' = 0$ *or* $c' = 1$, *then* $\neg(\mathsf{A} \rightsquigarrow^{+} \mathsf{B})$.

(5) *If* $c' \in \mathrm{Orb}_{\mathrm{S}}(c)$, *then* $\mathsf{A} \rightsquigarrow^{+} \mathsf{B}$.

(6) *If* c' *is a fixed point behaviour, then* $\mathsf{A} \rightsquigarrow^{+} \mathsf{B}$ *only if* $a \,\mathrm{R}\, c' \;\wedge\; c' \,\mathrm{R}\, b$.

Proof. Each of the proofs involve the applications of Definitions 5.2.1, 5.2.7, and 5.2.9, as well as the basic axioms of $\mathrm{C}^2\mathrm{KA}$. The detailed proofs can be found in Appendix A.12. □

Proposition 5.4.2 identifies the conditions constraining the modifications allowable to the behaviour of an agent in a potential communication path in order to maintain the potential for communication between two agents. In this way, it demonstrates the conditions under which a modification to an agent behaviour can be made while maintaining the communicating behaviour of the agents in the system. Specifically, Identity (1) shows how the sequential composition of an additional behaviour with the existing agent will not affect the potential for communication provided that the additional behaviour fixes every stimulus that it receives from the original intermediate agent C or that the composed behaviour preserves the dependency relation with the behaviour b. Assuming that each agent behaviour takes some amount of time, this is useful since behaviours that satisfy this constraint can be constructed to introduce delay into the potential communication path in order to disturb a covert timing channel without the need to fully eliminate the communication. However, in general, nothing can be said about the behaviour d alone as a consequence of Definition 4.2.3(2). The stimuli that are generated by d are dependent on the stimuli generated by c and the effects of the stimuli cascaded from c to d cannot be determined since agent C' is viewed as a black-box. Identity (2) is an extension of Propositions 5.2.3 and 5.2.4 to general potential for communication. In general, provided that the introduction of non-determinism does not result in a fixed point behaviour, the potential for communication is maintained with the addition of non-determinism. However, if an agent were constructed such that the agent behaviour $(c+d)$ is a fixed point behaviour, then this result can be used

to eliminate the potential for communication via external stimuli. In this way, the potential for communication will only be preserved if the agents have the potential for communication via shared environments. Identity (3) follows from Identities (1) and (2) and shows that the sequential iteration of an agent behaviour does not affect the potential for communication. Identity (4) states that if an agent in a communication path is replaced with an inactive agent or an idle agent, then there is no longer a potential for communication. This can be useful in terms of eliminating the potential for communication among agents since it shows how the behaviour of some agents may be modified in order to eliminate the potential for communication and potentially thwart any attempts for establishing covert communication channels. However, it is noted that this is not a suitable solution in all cases since modifying agent behaviours in such a way can inadvertently modify the overall system behaviour and thereby undesirably render the system useless. Identity (5) states that replacing an agent in a given communication path with another agent in the same strong orbit will not affect the potential for communication. This is because agents in the same strong orbit always have the potential for communication via external stimuli with one another. Identity (6) states that the potential for communication is maintained when replacing an agent in a given communication path with another agent that has a fixed point behaviour only if the dependency relation is preserved. Proposition 5.2.2 showed that an agent with a fixed point behaviour does not have the potential for communication via external stimuli unless it is the source of a potential communication path. So, if an agent with a fixed point behaviour is not the source of the potential communication path, then it may only have the potential for communication via shared environments. Finally, it should be noted that if the behaviour of an agent in a potential communication path is restricted to a particular sub-behaviour, then the potential for communication is only preserved if the sub-behaviour maintains the communicating behaviour of the original agent.

The results of Proposition 5.4.1 and Proposition 5.4.2 form a basis for an approach for mitigating distributed covert channels in systems of communicating agents. Such a mitigation

approach is able to preserve overall system behaviours and the communication between its components and with its environment. The goal of such mitigation approaches based on an analysis of the potential for communication amongst system agents is to thwart the use of distributed covert channels without the covert channel users necessarily being aware that their attempt to use a covert channel is being mitigated.

5.5 Discussion and Related Work

Given a system of communicating agents, it is difficult to fully prevent the possibility of covert communication from taking place since it is often undesirable to completely eliminate the communication among agents. An integral part of safeguarding systems of communicating agents from covert channel communication is having the ability to identify when a covert channel may exist in a given system which involves determining if and when two agents have a potential for communication. While much of the existing work in attempting to mitigate covert channels has been based on information theoretic approaches (e.g., [GW07, GH02, GH99, HR10, Low02, MMA06, Mil87, Mil89a, MK94]), the proposed formulation looks to the issue of mitigating covert channels from a different perspective. Although, it is difficult to completely eliminate covert channels from modern computer systems, the proposed formalisation provides a means for analysing a system of communicating agents in order to devise mechanisms for strengthening the design of such systems in order to make them more robust against covert channels. It also builds the foundation for the ability to identify parts of a system where it would be most beneficial to observe or disrupt the communication among particular system agents. For example, once a sequence of agents that have the potential for communication has been identified, in order to detect confidential information leakage via protocol-based covert channels, monitors can be installed and configured to identify patterns of communication on the communication

channels available to the agents in the potential communication path using techniques similar to that presented in [JKS11]. Similarly, in order to mitigate the use of covert timing channels, mechanisms can be employed to de-couple or deteriorate any sort of timing information associated with the communication channels available to the agents in the potential communication path by injecting random delays similar to the NRL Pump [KM93].

As mentioned in Section 5.1.1, the literature contains existing works that have attempted to articulate and verify potential for communication conditions for covert channels. However, some of them are indirect or informal and require reasoning about potential scenarios in which the conditions might be satisfied (e.g., [SC99]). Furthermore, those works which do provide some level of formalism, focus primarily on the potential for communication via shared environments through various information flow analyses based on finite state machine models, information theory, and probability theory (e.g., [Gra91, JLY10, Mil89a, Mil90, WL05b]). Perhaps one of the most popular mechanisms for determining the potential for communication for identifying the existence of covert channels is the Shared Resource Matrix technique [Kem83]. It involves a careful analysis of the ways in which shared resources are used in a system to determine whether it is possible for a particular resource to covertly transfer information from one agent to another with respect to a set of minimum criteria. Similarly, Covert Flow Trees (e.g., [KP91]) attempt to identify information flows supporting either the direct or indirect ability of an agent to detect when an attribute of a shared resource has been modified. The Shared Resource Matrix technique and Covert Flow Trees can be used in the proposed formulation to concretely build the dependence relation discussed in Section 5.2.2.

While existing works focus on studying the potential for communication via shared environments, the proposed formulation of the potential for communication condition for the existence of distributed covert channels is based on the mathematical foundation of C^2KA and thereby also considers the potential for communication via external stimuli. If the use of CKA alone were to be considered for the formulation of the potential for communication

condition, only the dependencies between shared events could be used to define and verify any sort of potential for communication. The proposed formulation provides a more complete representation of the potential means for communication among system agents that encompasses what can be done using CKA alone as well as other existing information flow techniques.

5.6 Conclusion

The potential for communication condition is one of the necessary conditions for the existence of distributed covert channels. The study of the potential for communication requires the study of the behaviour of the agents in a given system of communicating agents. This chapter has presented the formulation of the potential for communication condition for the existence of distributed covert channels in systems of communicating agents. The formulation is based on the mathematical framework of C^2KA that was presented in Chapter 4. Then, the verification of the potential for communication condition was demonstrated using the running example first introduced in Section 4.1 and the use of the prototype tool for automating the verification was discussed. The verification showed that the potential for communication condition is indeed satisfied for the given running example system of communicating agents. It remains to be seen whether the constraint on communication condition also holds in the running example system of communicating agents. The formulation and verification of the constraint on communication condition is discussed in Section 6.4.

Chapter 6

Communication Schemes and Agent Knowledge Evolution

To this point, this thesis has presented a mathematical framework for formulating and verifying the potential for communication condition for the existence of distributed covert channels in systems of communicating agents. While this is an important first step in determining whether a system of communicating agents has the ability to harbour covert channels, it is also important to determine whether there exists ways in which potential sources of confidential information leakage in a given system of communicating agents can develop communication schemes allowing for the establishment and operation of distributed covert channels. This chapter gives an outlook for the proposed mathematical framework for the modelling, analysis, and mitigation of distributed covert channels. It explores ways in which the understanding of covert channels can be enhanced by investigating further applications of the proposed framework. In particular, this chapter aims to propose directions and guidelines towards identifying potential confidential information leakage via distributed covert channels in systems of communicating agents. Specifically, Section 6.1 gives a representation for communication schemes based on description logic knowledge bases and pre-

and post-condition specifications. Section 6.2 guides a systematic approach for merging communication schemes into specifications of systems of communicating agents to obtain amended specifications allowing for the establishment of covert channels. Section 6.3 proposes a set of guidelines for the evolution of agent knowledge through the execution of concrete agent behaviours as a means of supporting an approach for verifying confidential information leakage. Section 6.4 provides a formulation and verification approach for the constraint on communication condition for the existence of distributed covert channels. It also outlines an approach for the verification of confidential information leakage in systems of communicating agents by analysing the necessary conditions for the existence of distributed covert channels. Lastly, Section 6.5 provides several concluding remarks and comments on future directions for the material presented in this chapter.

6.1 Communication Schemes

As discussed earlier in this thesis, in a given system of communicating agents, each agent is represented by describing its behaviour and its knowledge of the system and the world in which it evolves. Both of these aspects play a crucial role in the establishment of distributed covert channels as the behaviour of an agent dictates how it may communicate with other agents in the system while the knowledge of an agent determines what information it may communicate to other agents.

This section aims to enhance the understanding of the modelling and construction of distributed covert channels in systems of communicating agents. In particular, this section articulates a representation for communication schemes based on description logic knowledge bases and pre- and post-condition specifications. Additionally, it shows how communication schemes can be derived from the initial specification of a system of communicating agents where the behaviour of each agent is specified using the mathematical framework of Communicating Concurrent Kleene Algebra (C^2KA) and the knowledge of each agent is specified using the description logic $\mathcal{ALB}$.

6.1.1 Components of Communication Schemes

When two agents decide to establish a covert channel, they must first devise a communication scheme that is to be shared between them. A *communication scheme* describes how information will be represented, transmitted, and understood from a sender to a receiver in the communication. In this way, a communication scheme is an amendment to both the knowledge and the behaviour of the sender and the receiver. Therefore, a communication scheme is comprised of two components $\left(\mathcal{N}_{\mathsf{CS}}, \mathcal{B}_{\mathsf{CS}}\right)$. The knowledge component $\mathcal{N}_{\mathsf{CS}} = \left(\mathcal{T}_{\mathsf{CS}}, \mathcal{A}_{\mathsf{CS}}\right)$ is a knowledge base containing all of the information required to carry out the communication of some information, including the shared representation of the information to be transmitted. The behaviour component $\mathcal{B}_{\mathsf{CS}}$ consists of specifications of the behaviours of the sending and receiving agents that will allow them to transmit and understand messages.

6.1.2 Classifications of Communication Schemes

The knowledge component $\mathcal{N}_{\mathsf{CS}}$ of a communication scheme may assume different forms. The form of the knowledge component of a communication scheme leads to the following classification of communication schemes.

Terminological Communication Schemes

A communication scheme with a knowledge component of the form $\mathcal{N}_{\mathsf{CS}} = \left(\mathcal{T}_{\mathsf{CS}}, \emptyset\right)$ is called *terminological*. A terminological communication scheme is one where the ABox is empty. This means that the communication scheme consists of only terminological axioms. Terminological communication schemes are typically employed when the two communicating agents have different understandings of the world and require concept translations (i.e., relationships between concepts) so that they may "speak the same language". For example, suppose that agent A and agent B wish to communicate about cars. In the world of agent A, cars may be understood by the concept $\mathsf{Vehicle}$, whereas in the world of agent B,

cars may be understood by the concept $\mathsf{Automobile}$. While agent A and agent B both have an understanding of cars, their terminology differs. Therefore, in order to communicate with one another about the subject of cars, they must have a communication scheme that asserts $\mathsf{Vehicle} \equiv \mathsf{Automobile}$. Only in this way, will agent A and agent B be able to share a common understanding of the subject of cars. Similarly, one can imagine a terminological communication scheme as a language translation. Consider two agents, one that speaks English and one that speaks French. A terminological communication scheme can be constructed to define a kind of English-French dictionary (e.g., $\mathsf{House} \equiv \mathsf{Maison}$) so that the agents may be able to communicate and understand one another.

Assertional Communication Schemes

A communication scheme with a knowledge component of the form $\mathcal{N}_{\mathsf{CS}} = \big(\emptyset, \mathcal{A}_{\mathsf{CS}}\big)$ is called *assertional*. An assertional communication scheme is one where the TBox is empty. This means that the communication scheme is comprised only of assertional axioms. Assertional communication schemes are primarily used when the two communicating agents already share a common understanding of the world and "speak the same language", but need to share relationships and knowledge about objects in the world with respect to concepts and roles. The most common form of an assertional communication scheme comes when two communicating agents need to establish a mapping between objects. For example, consider a scenario where agent A and agent B are part of a card-counting ring and they need to establish a communication scheme that associates code-words to card-counts. Assume that agent A and agent B share a common understanding of the world and each knows the concepts $\mathsf{CodeWord}$ and Count. Then, in order to understand that a particular code-word indicates a specific card-count, agent A and agent B must have a communication scheme that asserts, for example $\mathsf{Map}\big(\mathsf{ICE}, +3\big)$, where Map is a new role symbol that is augmented to the set of role symbols N_R in the description logic signature. In this way, agent A and agent B will know that when the code-word "ice" is spoken that the current count is $+3$.

Procedural Communication Schemes

A communication scheme with an empty knowledge component $\big($i.e., $\mathcal{N}_{\mathsf{CS}} = (\emptyset, \emptyset)\big)$ is called *procedural*. Procedural communication schemes are commonly used when the two communicating agents need not share any additional explicit knowledge, but rather must simply devise a means for altering their behaviour in order to send and receive information. In essence, a procedural communication scheme represents a specification of a behaviour for a sender to encode a particular message to be sent and a specification of a behaviour for a receiver to extract the message that has been sent based on the information it has received. These specifications can be thought of as being analogous to an encryption algorithm for the sender and a decryption algorithm for the receiver. For example, suppose that agent A wishes to inform agent B of its date of birth. However, agent A prefers not to send its date of birth directly and would like to obscure the information. Assume that agent A and agent B initially know and agree on the ISO date format (i.e., YYYY-MM-DD). Also, assume that agent A and agent B initially share three variables containing integer values. Let these three variables be denoted by x, y, and z. In this case, in order for agent A to send its date of birth in an obscured way to agent B, no additional explicit knowledge needs to be shared between agent A and agent B. Instead, they can use the knowledge that they already share to devise a communication scheme for communicating the information. For example, they can develop a procedure where agent A multiplies the value of x to the birth year, adds the value of y to the birth month, and subtracts the value of z from the birth day, and sends this information to agent B. Then, upon receiving this obscured birth date, agent B reverses the operations by dividing the received birth year by the value of x, subtracting the value of y from the received birth month, and adding the value of z to the received birth day, in order to uncover the actual birth date. In this way, the agents merely define a procedure for sending and receiving information using the knowledge that they already share.

Composite Communication Schemes

A communication scheme with a knowledge component of the form $\mathcal{N}_{\mathsf{CS}} = (\mathcal{T}_{\mathsf{CS}}, \mathcal{A}_{\mathsf{CS}})$, where both $\mathcal{T}_{\mathsf{CS}}$ and $\mathcal{A}_{\mathsf{CS}}$ are non-empty, is called *composite*. A composite communication scheme is merely a combination of a terminological communication scheme and an assertional communication scheme. This means that the communication scheme consists of both terminological and assertional axioms. Composite communication schemes are commonly used when the two communicating agents need to share relationships and knowledge about objects in the world with respect to concepts and roles and when they do not already agree on the understanding of the world or "speak the same language". An example of a composite communication scheme is discussed later in this chapter in Section 6.1.4.

6.1.3 An Example Communication Scheme

Consider the running example described in Section 4.1. Suppose that agent S would like to establish a covert channel with agent R. Before any communication can begin, agent S and agent R need to devise and agree upon a communication scheme in order to establish a covert channel. Suppose that agent S is designated as the sender and agent R is designated as the receiver. Let agents S and R agree to establish and use a variation of the FTP Command Mapping covert channel described in [ZLSN05] where each FTP command is encoded as a fixed-length bit-string. For example, the *NOOP* command can be mapped to the bit-string 11 and the *ALLO* command can be mapped to the bit-string 01. Recall that from the initial knowledge specification of the system of communicating agents (see Section 4.3.1), agents S and R have a knowledge of the behaviour of agent P; specifically the enumeration mapping used by agent P. Agents S and R exploit this knowledge to devise a communication scheme that will allow agent S to communicate a message to agent R, indirectly through agent P. Under the agreed upon communication scheme and with the establishment of a shared command-to-bit-string mapping for the available FTP commands

in the system of communicating agents, agent S can choose to issue particular FTP commands in order to implement the idle prevention scheme so that a specific bit-string message can be transmitted to agent R. The idea is that when agent P receives a command from agent S, it will store the enumeration of the received command in the variable `cmd` which is shared with agent R. In this way, when agent R reads the variable `cmd`, it can obtain the command that was sent by agent S and determine the corresponding fixed-length bit-string using the agreed upon command mapping established in the communication scheme and shared with agent S.

Once again, it is acknowledged that in a real system of communicating agents, messages that are to be sent between agents are likely to be quite large. However, in the interest of simplicity and brevity, assume that a message is only (a very small) 2-bits in length. In this way, a message is exactly the bit-string corresponding to the command that was issued. Roughly speaking, this corresponds to one iteration of the communication protocol described by the communication scheme. It is important to note, that this can be extended in a natural way by establishing a notion of "message blocks" and by iterating the protocol defined by the communication scheme so that the sender S can transmit a *structured message* to the receiver R. This is to say that a message can consist of some number of 2-bit blocks, each corresponding to the issuance of a particular FTP command, and arranged in a particular sequence. Therefore, after repeated iteration of the communication protocol defined by the communication scheme, the sender S can transmit some number of sequential blocks which can be reassembled by the receiver R to obtain a complete bit-string message of some arbitrary length.

6.1.4 Guidelines for Systematically Devising Communication Schemes

In order to specify a communication scheme $\left(\mathcal{N}_{\mathsf{CS}}, \mathcal{B}_{\mathsf{CS}}\right)$, a specification of both the knowledge component and the behaviour component is required. The knowledge component $\mathcal{N}_{\mathsf{CS}}$ of a communication scheme is specified as a description logic knowledge base (see Section 4.3)

and the behaviour component $\mathcal{B}_{\mathsf{CS}}$ is captured with pre- and post-condition specifications (see Section 3.4) of the behaviours of the sending and receiving agents. Based on an informal description of the communication scheme to be established and the initial knowledge base specifications of the sending and receiving agents, the specification of the communication scheme can be devised in a systematic way.

In what follows, consider the communication scheme described in Section 6.1.3 and the knowledge specifications of the agents given in Section 4.3 for the running example system of communicating agents from Section 4.1.

Step 1: Identify the Shared Knowledge of the Sender and Receiver

The first step towards systematically devising a specification for a communication scheme, particularly the knowledge component $\mathcal{N}_{\mathsf{CS}}$, is to identify the shared knowledge of the sender and receiver around which the communication scheme will be developed. This shared knowledge is incorporated as part of the knowledge component of the communication scheme to indicate that it is knowledge that is vital to the operation of the communication scheme when it is employed to establish a covert channel. With respect to the running example, the knowledge that is shared between the sender S and the receiver R includes the terminological axiom $\mathsf{EnumToCmd} \equiv \mathsf{CmdToEnum}^{\smile}$, as well as the assertional axioms represented by the sets CMDS, ENUMS, BITS, and MAPS (see Section 4.3.1, page 112). Particularly, the inclusion of the enumeration mapping in the specification of the communication scheme ensures that the sending agent S and receiving agent R know the same mapping and that they do, in fact, agree on it before a covert channel is established.

Step 2: Construct Additional Knowledge Needed for the Communication Scheme

The next step towards systematically devising the knowledge component $\mathcal{N}_{\mathsf{CS}}$ of the specification of a communication scheme is to construct any additional knowledge that may

be required to be shared between the sending and receiving agents that will use the communication scheme. This additional knowledge may come in the form of terminological axioms, assertional axioms, or both. In terms of the example communication scheme outlined in Section 6.1.3, the additional knowledge to be constructed and added to the knowledge component of the communication scheme consists of the assertional axioms represented in the set $\{\mathsf{CmdToStr}(\mathsf{ABOR}, 00), \mathsf{CmdToStr}(\mathsf{ALLO}, 01), \mathsf{CmdToStr}(\mathsf{HELP}, 10), \mathsf{CmdToStr}(\mathsf{NOOP}, 11)\}$ corresponding to a command-to-bit-string mapping. It also consists of the terminological axiom $\mathsf{StrToCmd} \equiv \mathsf{CmdToStr}^{\smile}$ showing that there is additionally a bit-string-to-command mapping. This mapping is the fundamental knowledge required in order to achieve an FTP Command Mapping covert channel as described in [ZLSN05] and outlined in Section 6.1.3.

Step 3: Add Concept Inclusions According to the Method of Transmission and Interpretation of Confidential Information

Another important aspect that the knowledge component of the communication scheme must capture is the information that is to be sent by the sending agent. In this way, both the sending agent and the receiving agent know what information the sending agent is expected to use in order to transmit the intended message. This knowledge comes in the form of a concept inclusion and can be determined, in part, from the additional axioms that were constructed in Step 2. For instance, with respect to the running example, Step 2 found that a command-to-bit-string mapping was required knowledge for the given communication scheme. Therefore, it follows that the sending agent will be sending FTP commands that will be interpreted as bit-strings by the receiving agent. This knowledge must be shared between the sender and the receiver, hence it needs to be included as part of the knowledge component of the communication scheme. The knowledge representing that the sending agent will be sending FTP commands is captured by the terminological axiom $\mathsf{Sent} \sqsubseteq \mathsf{Command}$. Similarly, the participating agents, particularly the receiving

agent, needs to know how to interpret the confidential information that it receives and stores. Once again, this knowledge is devised by imagining how the receiving agent will receive and store the confidential information resulting from the communication with the sending agent. For the running example, this knowledge is captured by the terminological axiom $\big((\exists\, \mathsf{Variable}^{\smile}\,.\,\mathsf{ConfVar}) \sqcap \mathsf{BitString}\big) \sqsubseteq \mathsf{ConfInfo}$, where $\mathsf{ConfVar}$ is a newly introduced concept that represents variable names that store confidential information. For the running example, this variable is defined as the variable named y through the concept assertion $\mathsf{ConfVar}\big(\mathsf{Y}\big)$. In this way, the agents know that the receiver will have some variable named y for which its contents is a bit-string representing the confidential information that was transmitted. It is important to note that this terminological axiom is derived from the communication scheme that is established between the sender and the receiver and that it may be different under different communication schemes.

Step 4: Extend the Description Logic Signature

In particular, Steps 2 and 3 saw the introduction of new terminological and assertional axioms when specifying the knowledge component of a communication scheme. Consequently, those steps typically require the addition of new concepts and roles. Because of this, it is often necessary to extend the description logic signature of the system of communicating agents for which the communication scheme is being defined with these additional concepts and roles. This extension is required to ensure that all of the terminological and assertional axioms incorporated as part of the specification of the knowledge component of the communication scheme are defined with respect to the signature.

Considering the running example, $\big(N_C, N_R, N_O\big)$ denotes the signature of the system of communicating agents for which the communication scheme is being defined (see Figure 4.9). Based on the introduction of the new terminological and assertional axioms from Steps 2 and 3, the signature of the communication scheme is the extension $\big(N'_C, N'_R, N'_O\big)$ shown in Figure 6.1. The signature is extended with an additional concept Sent which denotes

$$\begin{aligned} N'_C &= N_C \cup \{\mathsf{ConfVar}, \mathsf{Sent}\} \\ N'_R &= N_R \cup \{\mathsf{CmdToStr}, \mathsf{StrToCmd}\} \\ N'_O &= N_O \cup \{\mathsf{Y}\} \end{aligned}$$

Figure 6.1: The extended $\mathcal{ALCB}$ signature for the example communication scheme

the command that was sent by the sending agent, the concept ConfVar which denotes variable names that store confidential information, the roles CmdToStr and StrToCmd which represent the shared command-to-bit-string and bit-string-to-command mappings of the communication scheme, and the object Y which denotes the name of the variable that will store the confidential information, respectively.

Step 5: Determine Pre- and Post-Condition Specifications for the Sender and Receiver Behaviours

In the last step, the behaviour component $\mathcal{B}_{\mathsf{CS}}$ of the communication scheme needs to be specified by giving specifications of the sending and receiving agent behaviours. The sending and receiving agent behaviours are captured with pre- and post-condition specifications. In the specification of the sending and receiving behaviours, the pre-condition asserts what must be known to an agent before the program is executed and the post-condition asserts what is known after the execution of the program. The notation of Hoare triples is used for the specification of the behaviour component of a communication scheme.

With regard to the running example system of communicating agents and communication scheme, and for some $\mathsf{U}, \mathsf{V}, \mathsf{N} \in N_O$, the pre- and post-condition specification of the behaviour of the sending agent S is given by:

$$\{\mathcal{N}_{\mathsf{S}} \models \mathsf{ConfInfo}(\mathsf{U}) \ \wedge \ \mathsf{StrToCmd}(\mathsf{U}, \mathsf{V})\} \ \mathsf{S} \ \{\mathcal{N}_{\mathsf{S}} \models \mathsf{Sent}(\mathsf{V})\}$$

Similarly, the pre- and post-condition specification of the behaviour of the receiving agent R is given by:

$$\{\mathcal{N}_{\mathsf{R}} \models \mathsf{Variable}(\mathsf{CMD}, \mathsf{N}) \wedge \mathsf{EnumToCmd}(\mathsf{N}, \mathsf{V}) \wedge \mathsf{CmdToStr}(\mathsf{V}, \mathsf{U})\} \ \mathsf{R} \ \{\mathcal{N}_{\mathsf{R}} \models \mathsf{ConfInfo}(\mathsf{U})\}$$

The pre- and post-condition specification for the behaviours of the sending and receiving agents provide assertions on the knowledge of the agents. In this way, the pre- and post-conditions specify what an agent must know before and after executing their behaviour in order to establish and operate a covert channel according to the communication scheme. Particularly, the specification for the sending agent S shows that it must first know some confidential information. This follows from the definition of covert channels adopted in this thesis and the constraint on communication condition for the existence of distributed covert channels. It also shows that the sending agent S must know a mapping that maps bit-strings to commands. Then, after executing its behaviour, the sending agent S will know what command it issued in order to transmit the confidential information. Likewise, the specification for the receiving agent R shows that it must first know the value stored in the variable `cmd`, the enumeration mapping which will give the command corresponding to a given enumeration, and the mapping that will give the bit-string corresponding to a given command. Then, after executing its behaviour, the receiving agent R will know the confidential information that was transmitted by the sending agent S.

Complete Specification of the Communication Scheme

By following the guidelines proposed above, the complete specification of the communication scheme $(\mathcal{N}_{\mathsf{CS}}, \mathcal{B}_{\mathsf{CS}})$ described in Section 6.1.3 can be devised. The complete specification is shown in Figure 6.2. It represents a composite communication scheme since its knowledge component consists of both terminological and assertional axioms.

Perhaps one of the most interesting observations that can be made about the specification of the communication scheme given in Figure 6.2 is that the sending agent S and the receiving agent R are not required to directly share any variables. Instead, through the shared knowledge represented by the knowledge component of the communication scheme and through the pre- and post-condition specifications of the sending and receiving agent behaviours, the communication scheme defines the procedure by which the sending agent

$$
\begin{aligned}
\mathcal{N}_{\mathsf{CS}} &= \left(\mathcal{T}_{\mathsf{CS}}, \mathcal{A}_{\mathsf{CS}}\right) \text{ where} \\
\\
\mathcal{T}_{\mathsf{CS}} &= \{\mathsf{EnumToCmd} \equiv \mathsf{CmdToEnum}^{\smile}, \mathsf{StrToCmd} \equiv \mathsf{CmdToStr}^{\smile}, \mathsf{Sent} \sqsubseteq \mathsf{Command}, \\
&\quad \left((\exists\, \mathsf{Variable}^{\smile}\,.\, \mathsf{ConfVar}) \sqcap \mathsf{BitString}\right) \sqsubseteq \mathsf{ConfInfo}\} \\
\mathcal{A}_{\mathsf{CS}} &= \text{CMDS} \cup \text{ENUMS} \cup \text{BITS} \cup \text{MAPS} \cup \{\mathsf{ConfVar}\left(\mathsf{Y}\right), \mathsf{CmdToStr}\left(\mathsf{ABOR}, 00\right), \\
&\quad \mathsf{CmdToStr}\left(\mathsf{ALLO}, 01\right), \mathsf{CmdToStr}\left(\mathsf{HELP}, 10\right), \mathsf{CmdToStr}\left(\mathsf{NOOP}, 11\right)\}
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{B}_{\mathsf{CS}} &= \left(\mathcal{B}_{\mathsf{S}}, \mathcal{B}_{\mathsf{R}}\right) \text{ where } \mathsf{U}, \mathsf{V}, \mathsf{N} \in N_O \text{ and} \\
\\
\mathcal{B}_{\mathsf{S}} &= \{\mathcal{N}_{\mathsf{S}} \models \mathsf{ConfInfo}\left(\mathsf{U}\right) \ \wedge \ \mathsf{StrToCmd}\left(\mathsf{U}, \mathsf{V}\right)\} \\
&\qquad {\scriptstyle \mathsf{S}} \\
&\quad \{\mathcal{N}_{\mathsf{S}} \models \mathsf{Sent}\left(\mathsf{V}\right)\} \\
\mathcal{B}_{\mathsf{R}} &= \{\mathcal{N}_{\mathsf{R}} \models \mathsf{Variable}\left(\mathsf{CMD}, \mathsf{N}\right) \ \wedge \ \mathsf{EnumToCmd}\left(\mathsf{N}, \mathsf{V}\right) \ \wedge \ \mathsf{CmdToStr}\left(\mathsf{V}, \mathsf{U}\right)\} \\
&\qquad {\scriptstyle \mathsf{R}} \\
&\quad \{\mathcal{N}_{\mathsf{R}} \models \mathsf{ConfInfo}\left(\mathsf{U}\right)\}
\end{aligned}
$$

Figure 6.2: Specification of an example communication scheme $\left(\mathcal{N}_{\mathsf{CS}}, \mathcal{B}_{\mathsf{CS}}\right)$ based on FTP command mapping

can successfully transmit some confidential information to the receiving agent. However, it remains to be seen how the communication scheme affects the specifications of the sending and receiving agents. This notion is investigated further in Section 6.2.

6.1.5 Discussion and Related Work

While there exists a large and vast literature of covert communication schemes and covert channel constructions (e.g., [AA11, CBS04, GGLT02, HS96, Sal09, SWBS09]), there does not exist a formal approach for modularly developing and specifying communication schemes similar to what is presented in this section. After closely examining the literature, the representation of a communication scheme as a knowledge base and a specification of the behaviour of the sending and receiving agents does not exist in the literature. This leads to a new and innovative way of thinking about distributed covert channels.

6.2 Merging Communication Schemes into Systems of Communicating Agents

Assume that the specifications of a system of communicating agents and a communication scheme are given. Recall that a communication scheme is an amendment to both the knowledge and the behaviour of the sending agent and the receiving agent in the system of communicating agents. This section proposes an approach for systematically merging a communication scheme into the specification of a system of communicating agents in order to augment the knowledge and the behaviour of the designated sending agent and receiving agent so that they are able to establish and operate a distributed covert channel. The approach is presented in two parts. First, Section 6.2.1 shows how to merge the knowledge component of a communication scheme specification with the knowledge bases of the sending agent and the receiving agent in the system of communicating agents to obtain amended knowledge base specifications for the sending and receiving agents. Then, Section 6.2.2 describes how to amend the behaviour of the sending agent and the receiving agent so that they may be able to perform the required actions to engage in a covert communication.

6.2.1 Amendments to Agent Knowledge

Since the knowledge component $\mathcal{N}_{\mathsf{CS}}$ of a communication scheme is itself a knowledge base, an operation for merging two knowledge bases specified using description logic is defined.

Definition 6.2.1 (Merged Knowledge)**.** *In general, consider a system formed by a set $\mathcal{C}$ of communicating agents and let $\mathsf{A}, \mathsf{B} \in \mathcal{C}$ be agents with knowledge bases $\mathcal{N}_{\mathsf{A}} = (\mathcal{T}_{\mathsf{A}}, \mathcal{A}_{\mathsf{A}})$ with respect to the signature $(N_{C_{\mathsf{A}}}, N_{R_{\mathsf{A}}}, N_{O_{\mathsf{A}}})$ and $\mathcal{N}_{\mathsf{B}} = (\mathcal{T}_{\mathsf{B}}, \mathcal{A}_{\mathsf{B}})$ with respect to the signature $(N_{C_{\mathsf{B}}}, N_{R_{\mathsf{B}}}, N_{O_{\mathsf{B}}})$, respectively. Then, the* merged knowledge of agents A and B *is defined as $\mathcal{N}_{\mathsf{A}} \sqcup \mathcal{N}_{\mathsf{B}} \stackrel{\text{def}}{=} (\mathcal{T}_{\mathsf{A}} \cup \mathcal{T}_{\mathsf{B}}, \mathcal{A}_{\mathsf{A}} \cup \mathcal{A}_{\mathsf{B}})$ with respect to the signature $(N_{C_{\mathsf{A}}} \cup N_{C_{\mathsf{B}}}, N_{R_{\mathsf{A}}} \cup N_{R_{\mathsf{B}}}, N_{O_{\mathsf{A}}} \cup N_{O_{\mathsf{B}}})$.* ■

Merging two knowledge bases consists of merging both the knowledge bases (i.e., the TBoxes and the ABoxes), as well as the signatures for each knowledge base. In this way, the merging operation results in a new knowledge base with respect to an extended signature. The merging of the signatures is required to ensure that all of the axioms in the merged knowledge base are defined.

A keen reader will identify that, in general, the merging of two knowledge bases according to Definition 6.2.1 can raise many consistency issues with respect to the resulting knowledge base. It is noted that this is indeed an issue, and while there exists techniques to deal with such issues such as belief revision (e.g., [AGM85, DP97, FH94]), among others (e.g., [CKNZ10, LT97, LSWH09, TO95]), in the interest of simplicity, it is assumed that knowledge bases are always consistent and that no inconsistencies are introduced through the merging of knowledge bases. The issue of knowledge base consistency can be considered an issue of knowledge management and is not addressed in this thesis. Specifically, this thesis, and more precisely this section, aims to merge the knowledge component of a communication scheme with the knowledge base of a system agent. Since a communication scheme is typically constructed by system agents wishing to establish a communication channel, it is reasonable to assume that the agents would not define a communication scheme that is inconsistent with and contradictory to their existing knowledge.

Suppose that the specification of a communication scheme and a system of communicating agents with two agents identified as the sender S and the receiver R for the covert communication are provided. The knowledge of the sending and receiving agents are amended by applying Definition 6.2.1 using the knowledge component of the communication scheme $\mathcal{N}_{\mathsf{CS}}$ and the knowledge of the sender $\mathcal{N}_{\mathsf{S}}$ and the receiver $\mathcal{N}_{\mathsf{R}}$, respectively. This involves computing the merged knowledge of the communication scheme and the sender $\mathcal{N}_{\mathsf{CS}} \sqcup \mathcal{N}_{\mathsf{S}}$ and the merged knowledge of the communication scheme and the receiver $\mathcal{N}_{\mathsf{CS}} \sqcup \mathcal{N}_{\mathsf{R}}$, both of which are defined with respect to the extended signature $\left(N'_C, N'_R, N'_O\right)$ of which the knowledge component of the communication scheme is also defined (see Figure 6.1). By merging

the knowledge component of the communication scheme with the knowledge bases of the sender and the receiver, amended knowledge bases for the sender and the receiver which include the additional knowledge required in order to represent and interpret the transmitted messages according to the communication scheme are obtained.

Specification of Agent Knowledge After the Communication Scheme Merge

Consider the running example specified in Section 4.3.1 and the communication scheme specified in Section 6.1.4. Let agent S be identified as the sender and let agent R be identified as the receiver in the covert communication. The amended knowledge specifications for the sender S and the receiver R are obtained by applying the proposed approach for amending the initial knowledge of the sending and receiving agent by way of computing the respective merged knowledge bases with the knowledge component of the given communication scheme according to Definition 6.2.1. The amended knowledge specifications for the sender S and the receiver R are shown in Figure 6.3 and Figure 6.4, respectively. Note that both of the amended knowledge specifications for the sender and the receiver are defined with respect to the extended signature $\left(N'_C, N'_R, N'_O\right)$ shown in Figure 6.1.

As a result of merging the knowledge component $\mathcal{N}_{\mathsf{CS}}$ of the communication scheme with the initial knowledge of the sender S and the receiver R, the amended knowledge specifications $\mathcal{N}_{\mathsf{S}}^{\mathsf{CS}}$ and $\mathcal{N}_{\mathsf{R}}^{\mathsf{CS}}$ for the sender S and receiver R are respectively obtained. Each of these amended knowledge bases contain the additional terminological and assertional axioms provided by the communication scheme and constructed in Steps 2 and 3 in Section 6.1.4. Specifically, the knowledge bases for the sender S and the receiver R now explicitly contain the command-to-bit-string mapping defined by the communication scheme. In this way, agent S and agent R have gained the knowledge required to establish a distributed covert channel based on the given communication scheme. It is important to note that the knowledge base specifications of each other agent in the system of communicating agents in the running example remain unchanged as a result to the amendments to agent knowledge.

$$
\begin{aligned}
\mathcal{N}_{\mathsf{S}}^{\mathsf{CS}} &= \mathcal{N}_{\mathsf{CS}} \sqcup \mathcal{N}_{\mathsf{S}}^{0} = \left(\mathcal{T}_{\mathsf{S}}^{\mathsf{CS}}, \mathcal{A}_{\mathsf{S}}^{\mathsf{CS}}\right) \text{ where} \\
\mathcal{T}_{\mathsf{S}}^{\mathsf{CS}} &= \mathcal{T}_{\mathsf{S}}^{0} \cup \{\mathsf{Sent} \sqsubseteq \mathsf{Command}, \mathsf{StrToCmd} \equiv \mathsf{CmdToStr}^{\smile}, \\
&\qquad \left((\exists\, \mathsf{Variable}^{\smile} \,.\, \mathsf{ConfVar}) \sqcap \mathsf{BitString}\right) \sqsubseteq \mathsf{ConfInfo}\} \\
\mathcal{A}_{\mathsf{S}}^{\mathsf{CS}} &= \mathcal{A}_{\mathsf{S}}^{0} \cup \{\mathsf{ConfVar}\left(\mathsf{Y}\right), \mathsf{CmdToStr}\left(\mathsf{ABOR}, 00\right), \mathsf{CmdToStr}\left(\mathsf{ALLO}, 01\right), \\
&\qquad \mathsf{CmdToStr}\left(\mathsf{HELP}, 10\right), \mathsf{CmdToStr}\left(\mathsf{NOOP}, 11\right)\}
\end{aligned}
$$

Figure 6.3: Amended knowledge specification of the sending agent S resulting from the communication scheme merge

$$
\begin{aligned}
\mathcal{N}_{\mathsf{R}}^{\mathsf{CS}} &= \mathcal{N}_{\mathsf{CS}} \sqcup \mathcal{N}_{\mathsf{R}}^{0} = \left(\mathcal{T}_{\mathsf{R}}^{\mathsf{CS}}, \mathcal{A}_{\mathsf{R}}^{\mathsf{CS}}\right) \text{ where} \\
\mathcal{T}_{\mathsf{R}}^{\mathsf{CS}} &= \mathcal{T}_{\mathsf{R}}^{0} \cup \{\mathsf{Sent} \sqsubseteq \mathsf{Command}, \mathsf{StrToCmd} \equiv \mathsf{CmdToStr}^{\smile}, \\
&\qquad \left((\exists\, \mathsf{Variable}^{\smile} \,.\, \mathsf{ConfVar}) \sqcap \mathsf{BitString}\right) \sqsubseteq \mathsf{ConfInfo}\} \\
\mathcal{A}_{\mathsf{R}}^{\mathsf{CS}} &= \mathcal{A}_{\mathsf{R}}^{0} \cup \{\mathsf{ConfVar}\left(\mathsf{Y}\right), \mathsf{CmdToStr}\left(\mathsf{ABOR}, 00\right), \mathsf{CmdToStr}\left(\mathsf{ALLO}, 01\right), \\
&\qquad \mathsf{CmdToStr}\left(\mathsf{HELP}, 10\right), \mathsf{CmdToStr}\left(\mathsf{NOOP}, 11\right)\}
\end{aligned}
$$

Figure 6.4: Amended knowledge specification of the receiving agent R resulting from the communication scheme merge

6.2.2 Amendments to Agent Behaviour

Consider a given communication scheme and system of communicating agents with two agents identified to be the sender and the receiver of the covert communication. Recall that the communication scheme provides a pre- and post-condition specification for the behaviour of the sender and the receiver so that they may be able to perform the required actions to establish and operate a distributed covert channel. In general, the behaviour of the sender and the receiver in a system of communicating agents can be amended to be any behaviour provided that it satisfies the pre- and post-condition specification given by the communication scheme. There may be many behaviours that can satisfy the pre- and post-condition specification for the sender and receiver behaviours. In practice, a programmer can be relied on to perform this task and it may be possible that this task can be partially automated, however, this point is not discussed any further in this thesis.

Let S be generated using the operations of stimulus structures and the set $\{ips, abor, allo, help, noop, \mathfrak{n}, \mathfrak{d}\}$, let S be a program, and let x be a variable in S such that:

$$
\begin{array}{lll}
\mathsf{S} & \stackrel{\text{def}}{=} & \boxed{y := \texttt{conf\_info};} \\
 & & \mathsf{if} \quad \boxed{y = 00} \longrightarrow x := \texttt{ABOR} \\
 & & [] \quad \boxed{y = 01} \longrightarrow x := \texttt{ALLO} \\
 & & [] \quad \boxed{y = 10} \longrightarrow x := \texttt{HELP} \\
 & & [] \quad \boxed{y = 11} \longrightarrow x := \texttt{NOOP} \\
 & & \mathsf{fi};
\end{array}
$$

and

$$
\begin{aligned}
&\forall (z \mid z \in S \backslash \{\mathfrak{n}, \mathfrak{d}\} : (\mathsf{receive}\ z) \circ \mathsf{S} = \mathsf{S} \ \wedge\ \lambda\big((\mathsf{receive}\ z), \mathsf{S}\big) = \mathsf{send}\ x\,) \ \wedge \\
&\quad \mathfrak{n} \circ \mathsf{S} = \mathsf{S} \ \wedge\ \lambda\big(\mathfrak{n}, \mathsf{S}\big) = \mathfrak{n} \ \wedge\ \mathfrak{d} \circ \mathsf{S} = \mathsf{abort} \ \wedge\ \lambda\big(\mathfrak{d}, \mathsf{S}\big) = \mathfrak{d}
\end{aligned}
$$

Figure 6.5: Amended concrete behaviour specification of the sending agent S resulting from the communication scheme merge

Specification of Agent Behaviour After the Communication Scheme Merge

Once again, consider the running example specified in Section 4.3.1 and the communication scheme specified in Section 6.1.4. Figure 6.5 and Figure 6.6 provide one example of an amended concrete behaviour specification satisfying the pre- and post-condition specifications of the behavioural component of the communication scheme for the sending agent S and the receiving agent R in the given system of communicating agents, respectively. The amendments are highlighted in the form of framed statements. It should be noted that it is possible to formally verify that the programs representing the amended sending and receiving agent behaviours satisfy their corresponding pre- and post-condition specifications. However, this verification requires that there exist a translation between the representations of agent knowledge and the representation of agent behaviours (i.e., a translation between description logic axioms and commands given in Dijkstra's guarded command language). This translation is not currently expressed as part of this thesis and remains a fundamental aspect of the future work.

Let S be generated using the operations of stimulus structures and the set $\{ips,\ abor,\ allo,\ help,\ noop,\ \mathfrak{n},\ \mathfrak{d}\}$ and let R be a program such that:

$$
\begin{aligned}
\mathsf{R} \overset{\text{def}}{=}\ & \texttt{since\_1} := \texttt{since\_1} + \texttt{delta};\\
& \texttt{since\_2} := \texttt{since\_2} + \texttt{delta};\\
& \texttt{since\_3} := \texttt{since\_3} + \texttt{delta};\\
& \texttt{since\_4} := \texttt{since\_4} + \texttt{delta};\\
& n := \texttt{cmd};\\
& \mathsf{if}\quad n = 1 \longrightarrow (\texttt{avg\_1} = \texttt{since\_1}/\texttt{num\_1};\ \texttt{since\_1} := 0;\ \boxed{x := \texttt{ABOR}}\ )\\
& []\quad n = 2 \longrightarrow (\texttt{avg\_2} = \texttt{since\_2}/\texttt{num\_2};\ \texttt{since\_2} := 0;\ \boxed{x := \texttt{ALLO}}\ )\\
& []\quad n = 3 \longrightarrow (\texttt{avg\_3} = \texttt{since\_3}/\texttt{num\_3};\ \texttt{since\_3} := 0;\ \boxed{x := \texttt{HELP}}\ )\\
& []\quad n = 4 \longrightarrow (\texttt{avg\_4} = \texttt{since\_4}/\texttt{num\_4};\ \texttt{since\_4} := 0;\ \boxed{x := \texttt{NOOP}}\ )\\
& \mathsf{fi};\\
& \boxed{\mathsf{if}\quad x := \texttt{ABOR} \longrightarrow y := 00}\\
& \boxed{[]\quad x := \texttt{ALLO} \longrightarrow y := 01}\\
& \boxed{[]\quad x := \texttt{HELP} \longrightarrow y := 10}\\
& \boxed{[]\quad x := \texttt{NOOP} \longrightarrow y := 11}\\
& \boxed{\mathsf{fi};}
\end{aligned}
$$

and

$$
\begin{aligned}
&\forall (z \mid z \in S \backslash \{\mathfrak{n}, \mathfrak{d}\} : (\mathsf{receive}\ z) \circ \mathsf{R} = \mathsf{R} \wedge \lambda\big((\mathsf{receive}\ z), \mathsf{R}\big) = \mathfrak{n}) \wedge\\
&\qquad \mathfrak{n} \circ \mathsf{R} = \mathsf{R} \wedge \lambda\big(\mathfrak{n}, \mathsf{R}\big) = \mathfrak{n} \wedge \mathfrak{d} \circ \mathsf{R} = \mathsf{abort} \wedge \lambda\big(\mathfrak{d}, \mathsf{R}\big) = \mathfrak{d}
\end{aligned}
$$

Figure 6.6: Amended concrete behaviour specification of the receiving agent R resulting from the communication scheme merge

The amendments to the behaviour of an agent as a result of merging a communication scheme into the specification of the system of communicating agents very often increases the determinism in the behaviour of the amended agents. This is best exemplified by examining the initial (see Figure 4.4) and the amended (see Figure 6.5) concrete behaviour specifications of the sending agent S in the running example. Initially, the behaviour of the sender S is completely non-deterministic allowing the sender to choose, at will, which FTP command to issue in order to implement the idle prevention scheme. However, after

the amendments imposed by merging the communication scheme into the specification, the behaviour of the sender S has become deterministic. The choice of which FTP command to issue is now completely determined by the confidential information that agent S wishes to leak. The introduction of the communication scheme causes the sender S to behave in a more deliberate way. Similar modifications and increases in determinism can also be seen in the amendments to the behaviour of the receiver R.

By examining the amendments to the behaviour of the sender S and the receiver R, it can be seen that the complexity of the pre-condition is related to the amount of modification that needs to be done to the agent behaviour. It is easy to see that the simpler the pre-condition, the less modification will be required in the specification of the behaviour of the agent to satisfy it. In the context of establishing covert channels, this notion of simple pre-conditions can indicate the potential for a "good" covert channel in the sense that it indicates the requirement for minimal modification to the agent specification. This can be related to the inability to detect such modifications.

6.2.3 Applications of Merging Communication Schemes into Systems of Communicating Agents

The proposed approach for merging communication schemes into specifications of systems of communicating agents has two complementary applications. The first application is related to the detection of covert channel vulnerabilities in the specifications of systems of communicating agents. The second application is related to the construction of covert communication schemes and mounting covert channels in systems of communicating agents.

Detecting Covert Channels

The proposed approach for merging communication schemes into the specification of systems of communicating agents can be used to analyse systems of communicating agents for their ability to harbour some types of covert channels. Suppose that the specification of a

system of communicating agents is given. Using the proposed approach, a number of covert communication schemes that may be used in the given system can be articulated and each of these covert channels can be mounted in the system by merging the communication schemes into system specification. Once the amended specifications of the system of communicating agents have been obtained, techniques such as those found in [JKS11, JKS14] can be used to determine if it is possible for any of the mounted covert channels to be used to leak confidential information from a source agent to a sink agent in the system. Furthermore, the proposed approach allows for the development and combination of multiple communication schemes. In this way, the proposed approach allows for the realisation of the perception of covert channel communication provided in [JK11b] where some number of covert channels based on different communication schemes can be combined in order to create a complex covert communication mechanism for the purpose of leaking confidential information.

Constructing Covert Channels

Alternatively, the proposed approach for merging communication schemes into the specification of systems of communicating agents can be used to analyse existing networks of agents for ways in which covert communication schemes can be devised and mounted in the system. The idea is that covert communication schemes can be modularly designed and developed based on a given specification of a system of communicating agents. Then, a covert channel can be mounted in the system by choosing a sending and receiving agent and using the proposed approach to merge the developed covert communication scheme with the existing system specification. In this way, a new system specification can be obtained such that a covert channel based on the developed covert communication scheme can be established and used. Then, the new system specification can be analysed for the effectiveness of the mounted covert channel by measuring some qualities of covert channels such as stealth, robustness, and capacity, among others. This can be done using various existing techniques, metrics, and approaches, such as those based on information theoretical

concepts (e.g., [CT91]), for example. In this way, the proposed approach can be used as a tool for constructing and testing the quality of covert communication schemes with respect to systems of communicating agents at the early stages of system development, particularly at the system specification level.

6.2.4 Discussion and Related Work

The proposed approach illustrates how communication schemes and covert channels can be designed early in the development of a system of communicating agents, particularly, at the system specification level. The proposed approach also provides a way of analysing the specification of a system of communicating agents for potential vulnerabilities to covert channels as it gives a way to specify and inject potential communication schemes into the specification of the system of communicating agents and to determine whether the amended specification of the system of communicating agents can harbour a covert channel for the purpose of leaking confidential information. After closely examining the literature, a systematic approach for merging communication schemes into the specification of systems of communicating agents does not currently exist. Such an approach can aid in the enhanced understanding, construction, and detection of covert channels in systems of communicating agents.

6.3 Evolution of Agent Knowledge

This section outlines a set of guidelines towards the development of a framework for expressing the evolution of agent knowledge through the execution of concrete agent behaviours. The execution of the program statements in the concrete behaviour of an agent in a given system of communicating agents correspond to insertions and updates in agent knowledge bases. It is through these insertions and updates that the knowledge of an agent evolves

during system execution. The proposed guidelines and framework are based on the representation of agent knowledge described in Section 4.3.1 and the concrete behaviour specification of a system of communicating agents specified using Dijkstra's guarded command language [Dij75] given in Section 4.2.6.

6.3.1 Assumptions

When formulating the evolution of agent knowledge in systems of communicating agents, a number of basic assumptions are made in order to simplify the formulation. First, it is assumed that all agents are *honest*. This means that an agent can only communicate what it knows (i.e., there are no misinforming agents). Therefore, under this assumption, an agent A cannot learn something from agent B which agent B did not know. A similar assumption is made in [PP05]. Second, it is assumed that agents can only learn information from communication. This means that an agent cannot remove information from its knowledge base; it can only be inserted or updated. This assumption is made in order to simplify the formulation. Because of this assumption, the proposed guidelines and framework are only able to handle a subset of possible communication schemes. In future work, it is expected that this assumption can be lifted to allow for the handling of notions of "*forgetting*" (e.g., [Wan11, WWTP10, ZZ09]), however, this is not addressed in this thesis. When communicating agents learn new information, it is possible that the assimilation of new information leads to a new updated state which is inconsistent with respect to the existing information. This is to say that through communication, it is possible that agents learn contradicting information. While there exists techniques to deal with such issues such as belief revision (e.g., [AGM85, DP97, FH94]) and others (e.g., [CKNZ10, LT97, LSWH09, TO95]), in the interest of simplicity, it is assumed that knowledge bases are always consistent and that no inconsistencies are introduced through communication. The issue of knowledge base consistency can be considered an issue of knowledge management and is not dealt with in this thesis. It is also assumed that there is memory and naming consistency for shared

variables. This means that variables with the same name have the same interpretation (i.e., they point to the same memory location). This assumption is required to simplify the evolution of agent knowledge through communication via shared environments. Finally, throughout the formulation of the evolution of agent knowledge, it is assumed that the signature of the description logic is such that all terminological and assertional axioms that are being inserted or updated in the knowledge bases are well-formed axioms over that signature.

6.3.2 Operations for Updating Agent Knowledge

As part of the development of a framework for evolving the knowledge of agents in systems of communicating agents, a number of operations are defined. In what follows, consider a system formed by a set $\mathcal{C}$ of communicating agents and for each agent $\mathsf{A} \in \mathcal{C}$, let $\mathcal{N}_{\mathsf{A}}$ be its knowledge base.

- $getConcepts\left(\mathsf{X}, \mathcal{N}_{\mathsf{A}}\right) = \{\mathsf{C} \mid \mathcal{N}_{\mathsf{A}} \models \mathsf{C}(\mathsf{X})\}$

  This operation takes an object and an agent knowledge base and returns a set of all of the concepts to which the object is known to belong by the agent.

- $insert\left(\varphi, \mathcal{N}_{\mathsf{A}}\right) = \{\varphi\} \cup \mathcal{N}_{\mathsf{A}}$

  This operation inserts the axiom φ into the agent knowledge base $\mathcal{N}_{\mathsf{A}}$.

- For some $\mathsf{V} \in N_O$:

  $\forall\big(\mathsf{A} \mid \mathsf{A} \in \mathcal{C} : \mathcal{N}_{\mathsf{A}} \models \mathsf{Variable}\big(\mathsf{Y}, \mathsf{V}\big) \implies$

  $update\left(\mathsf{Variable}\big(\mathsf{Y}, \mathsf{X}\big), \mathcal{C}\right) = \left(\mathcal{N}_{\mathsf{A}} \backslash \{\mathsf{Variable}\big(\mathsf{Y}, \mathsf{V}\big)\}\right) \cup \{\mathsf{Variable}\big(\mathsf{Y}, \mathsf{X}\big)\}\,\big)$

  This operation updates each knowledge base $\mathcal{N}_{\mathsf{A}}$ for all agents $\mathsf{A} \in \mathcal{C}$ that have knowledge of the variable Y in the given system of communicating agents with the assertional axiom $\mathsf{Variable}\big(\mathsf{Y}, \mathsf{X}\big)$. This operation ensures that at most one assertional axiom for the role $\mathsf{Variable}$ with the domain element Y is included in each knowledge base $\mathcal{N}_{\mathsf{A}}$. Furthermore, this operation ensures that the value of a variable is the same for each

agent that shares the variable. This arises from the assumption of shared variable naming and memory consistency.

- For some $\mathsf{V} \in N_O$:

  $\left(\mathcal{N}_\mathsf{A} \models \mathsf{Sent}(\mathsf{V}) \implies \mathit{updateSent}\left(\mathsf{X}, \mathcal{N}_\mathsf{A}\right) = (\mathcal{N}_\mathsf{A} \backslash \{\mathsf{Sent}(\mathsf{V})\}) \cup \{\mathsf{Sent}(\mathsf{X})\} \cup \right.$
  $\left. \{\mathsf{Sent} \sqsubseteq \mathsf{C} \mid \mathsf{C} \in \mathit{getConcepts}\left(\mathsf{X}, \mathcal{N}_\mathsf{A}\right)\}\right) \vee$
  $\left(\mathcal{N}_\mathsf{A} \not\models \mathsf{Sent}(\mathsf{V}) \implies \mathit{updateSent}\left(\mathsf{X}, \mathcal{N}_\mathsf{A}\right) = \mathit{insert}\left(\mathsf{Sent}(\mathsf{X}), \mathcal{N}_\mathsf{A}\right)\right)$

  This operation updates the agent knowledge base $\mathcal{N}_\mathsf{A}$ with the concept inclusions and assertional axioms associated with the sending of the object X. This operation uses the special concept Sent and ensures that at most one assertional axiom for the concept Sent is included in the knowledge base $\mathcal{N}_\mathsf{A}$.

- For some $\mathsf{V} \in N_O$:

  $\left(\mathcal{N}_\mathsf{A} \models \mathsf{Received}(\mathsf{V}) \implies \mathit{updateReceived}\left(\mathsf{Y}, \mathcal{N}_\mathsf{A}\right) = (\mathcal{N}_\mathsf{A} \backslash \{\mathsf{Received}(\mathsf{V})\}) \cup \right.$
  $\left. \{\mathsf{Received}(\mathsf{Y})\} \cup \{\mathsf{Received} \sqsubseteq \mathsf{C} \mid \mathsf{C} \in \mathit{getConcepts}\left(\mathsf{Y}, \mathcal{N}_\mathsf{A}\right)\}\right) \vee$
  $\left(\mathcal{N}_\mathsf{A} \not\models \mathsf{Received}(\mathsf{V}) \implies \mathit{updateReceived}\left(\mathsf{Y}, \mathcal{N}_\mathsf{A}\right) = \mathit{insert}\left(\mathsf{Received}(\mathsf{Y}), \mathcal{N}_\mathsf{A}\right)\right)$

  This operation updates the agent knowledge base $\mathcal{N}_\mathsf{A}$ with the concept inclusions and assertional axioms associated with the receipt of the object Y. This operation uses the special concept $\mathsf{Received}$ and ensures that at most one assertional axiom for the concept $\mathsf{Received}$ is included in the knowledge base $\mathcal{N}_\mathsf{A}$.

6.3.3 Evolving Agent Knowledge Through the Execution of Concrete Agent Behaviours

Let *evolve* denote the result of updating the knowledge base $\mathcal{N}_\mathsf{A}$ by the execution of a concrete behaviour (program) S specified using Dijkstra's guarded command language. The result of *evolve* is an updated knowledge base for the given agent or for multiple agents in the case of an assignment to a variable that is shared with other agents in the system.

The *evolve* relation is defined by induction on program statements as follows where S_1, S_2, and S_n are program statements and G_1, G_2, and G_n are guards:

$$
\begin{aligned}
evolve(\mathsf{abort}, \mathcal{N}_\mathsf{A}) &= \emptyset \\
evolve(\mathsf{skip}, \mathcal{N}_\mathsf{A}) &= \mathcal{N}_\mathsf{A} \\
evolve(\mathtt{v} := \mathtt{E}, \mathcal{N}_\mathsf{A}) &= update\left(\mathsf{Variable}(\mathsf{V}, \mathsf{E}), \mathcal{C}\right) \\
evolve(\mathsf{IF}, \mathcal{N}_\mathsf{A}) &= \begin{cases} evolve(S_0, \mathcal{N}_\mathsf{A}) & \text{if } \mathcal{N}_\mathsf{A} \models G_0 \\ evolve(S_1, \mathcal{N}_\mathsf{A}) & \text{if } \mathcal{N}_\mathsf{A} \models G_1 \\ \ldots & \\ evolve(S_n, \mathcal{N}_\mathsf{A}) & \text{if } \mathcal{N}_\mathsf{A} \models G_n \\ evolve(\mathsf{abort}, \mathcal{N}_\mathsf{A}) & \text{otherwise} \end{cases} \\
evolve(\mathsf{DO}, \mathcal{N}_\mathsf{A}) &= \begin{cases} evolve(\mathsf{DO}, evolve(S_0, \mathcal{N}_\mathsf{A})) & \text{if } \mathcal{N}_\mathsf{A} \models G_0 \\ evolve(\mathsf{DO}, evolve(S_1, \mathcal{N}_\mathsf{A})) & \text{if } \mathcal{N}_\mathsf{A} \models G_1 \\ \ldots & \\ evolve(\mathsf{DO}, evolve(S_n, \mathcal{N}_\mathsf{A})) & \text{if } \mathcal{N}_\mathsf{A} \models G_n \\ evolve(\mathsf{skip}, \mathcal{N}_\mathsf{A}) & \text{otherwise} \end{cases} \\
evolve(S_1 \,;\, S_2, \mathcal{N}_\mathsf{A}) &= evolve(S_2, evolve(S_1, \mathcal{N}_\mathsf{A})) \\
evolve(\mathsf{send}\ x, \mathcal{N}_\mathsf{A}) &= updateSent\left(\mathsf{X}, \mathcal{N}_\mathsf{A}\right) \\
evolve(\mathsf{receive}\ y, \mathcal{N}_\mathsf{A}) &= updateReceived\left(\mathsf{Y}, \mathcal{N}_\mathsf{A}\right)
\end{aligned}
$$

where

$$
\begin{aligned}
\mathsf{IF} &\stackrel{\text{def}}{=} \mathsf{if} \quad G_0 \longrightarrow S_0 \ [] \quad G_1 \longrightarrow S_1 \ \ldots \ [] \quad G_n \longrightarrow S_n \ \mathsf{fi} \\
\mathsf{DO} &\stackrel{\text{def}}{=} \mathsf{do} \quad G_0 \longrightarrow S_0 \ [] \quad G_1 \longrightarrow S_1 \ \ldots \ [] \quad G_n \longrightarrow S_n \ \mathsf{od}
\end{aligned}
$$

According to the above definition, if the concrete behaviour of an agent is abort (inactive), then the knowledge is empty after the execution. This demonstrates that a program that behaves as an inactive agent has no knowledge. If the concrete behaviour of an agent is skip (idle), then no changes are made to the knowledge base of the agent during the execution. Suppose that the concrete behaviour of an agent is an assignment statement. Then, the knowledge base of all agents in the system of communicating agents which know about the variable that is being assigned to are updated by replacing the existing assertional axiom for the given defined variable with a new assertional axiom relating the new value to the variable. In this way, when a variable is modified by one agent in the system, then any agent which shares that variable has their knowledge updated as well. In each of these cases, there may only exist one assertional axiom pertaining to the role Variable with the same domain element (i.e., the variable name) in any given knowledge base. For instance, assume that there is a variable x that undergoes the following assignments $\mathtt{x} := 4;\ \mathtt{x} := 5$. After the execution of these assignment statements, the ABoxes of the agents that know the variable x contain only $\{\mathsf{Variable}(\mathsf{X}, 5)\}$ and not $\{\mathsf{Variable}(\mathsf{X}, 4), \mathsf{Variable}(\mathsf{X}, 5)\}$. If the concrete behaviour of an agent is a guarded selection statement, then the knowledge of the agent evolves in the way in which each statement for which the guard is true evolves. In this way, the definition follows the semantics of Dijkstra's guarded command language in that any statement for which the corresponding guard evaluates to true is non-deterministically selected, the corresponding statement is executed, and the knowledge of the agent evolves accordingly. The guards must be knowledge assertions to ensure that the behaviour proceeds only based on the knowledge of the agent, consistent with the assumption that an agent can only communicate what it knows. Additionally, in the case that none of the guards evaluate to true, then the execution of the selection aborts and the knowledge of the agent evolves in the same way as an abort statement (i.e., the knowledge becomes empty). A similar evolution occurs in the case where the concrete behaviour of an agent is a repetition statement. In this case, the knowledge of the agent evolves in the way in which each

statement for which the guard is true evolves and it repeats this evolution until all of the guards evaluate to false. When none of the guards evaluate to true, the resulting execution proceeds as a skip statement (i.e., the knowledge of the agent does not change). If the concrete behaviour of an agent is a sequential composition of program statements $(S_1 \,;\, S_2)$, then the knowledge of the agent evolves according to the statements represented by S_1 and then according to the statements represented by S_2. Finally, consider the case where the concrete behaviour of an agent is a send or a receive command denoting a message-passing communication. Generally speaking, with respect to the description logic signature for the given system, send x corresponds to the sending of some object $\mathsf{X} \in N_O$ and receive y corresponds to the receipt of some object $\mathsf{Y} \in N_O$. In this way, if the concrete behaviour of an agent is a command send x or receive y, then the knowledge of the agent is updated with terminological axioms according to the type of objects that each agent can send (i.e., X) or receive (i.e., Y) using concept inclusions and the special concepts Sent and Received, respectively. For example, if an agent A were to have a concrete behaviour send *abor* corresponding to the issuance of the stimulus *abor* and where $\mathcal{N}_{\mathsf{A}} \models \mathsf{Command}(\mathsf{ABOR})$, then after the execution of this command, the knowledge contains the concept inclusion $\mathsf{Sent} \sqsubseteq \mathsf{Command}$ to indicate that agent A sends commands, and the assertional axiom $\mathsf{Sent}(\mathsf{ABOR})$ to indicate that the object ABOR, representing the actual *ABOR* command, is indeed the command that has been sent. The sending and receiving of an object arises from the stimulus-response specification of agents in a system of communicating agents (see Section 4.2.6). Specifically, when the stimulus-response specification of an agent specifies that $\lambda(s, \mathsf{S}) = x$ for some stimulus $s \in S$ and sender behaviour S, there is a corresponding concrete behaviour send x for that agent. Similarly, when the stimulus-response specification of an agent specifies that $y \circ \mathsf{R} = a$ for some behaviour $a \in K$ and receiver behaviour R, there is a corresponding concrete behaviour receive y for that agent. In this way, the send and receive commands are similar to CSP's ! and ? operations [Hoa78a] and can be used for synchronisation.

Using the proposed framework and the *evolve* relation, the concrete behaviour of an agent A in a system of communicating agents can be specified as a program using Dijkstra's guarded command language which can be mapped to knowledge assertions during execution. Equivalently, the concrete behaviour of agent A can be specified as a sequence of knowledge assertions that an agent learns at each step in the sequence. The mapping provided above shows how the knowledge of each agent evolves through the execution of its concrete behaviours. It also shows the interplay between the behaviour and knowledge of each agent in a given system of communicating agents. In some sense, the *evolve* relation updates the state space of each agent as a result of the execution of concrete behaviours.

6.3.4 Illustrative Example of the Evolution of Agent Knowledge

Consider the amended concrete behaviour specifications for agent S and agent R as shown in Figure 6.5 and Figure 6.6, respectively. Also, consider the concrete behaviour specification of agent P as shown in Figure 4.5. Moreover, assume the merged knowledge specifications for agent S and agent R given in Figure 6.3 and Figure 6.4, as well as the initial knowledge specification for agent P given in Figure 4.12.

For the sake of illustration and with respect to the system of communicating agents from Section 4.1, let the value of the variable `time` be 3 when the concrete behaviours of each agent are executed $\big($i.e., $\mathsf{Variable}\big(\mathsf{TIME}, 3\big)$ for each agent$\big)$. Then, by applying the evolution of agent knowledge for the program $(\mathsf{S}\,;\mathsf{P}\,;\mathsf{R})$, the knowledge bases shown in Figures 6.7 to 6.9 are obtained.

6.3.5 Discussion and Related Work

The evolution of agent knowledge allows for agent knowledge to be updated as a result of the execution of the concrete behaviour of agents. A similar notion has been presented in [SKJ09b, SKJ09a] where an amended version of Hoare logic was used to verify information flow in multi-agent systems. The evolution of agent knowledge presented in this section

$$
\begin{aligned}
\mathcal{N}_{\mathsf{S}} &= (\mathcal{T}_{\mathsf{S}}, \mathcal{A}_{\mathsf{S}}) \text{ where} \\
\\
\mathcal{T}_{\mathsf{S}} &= \big\{\mathsf{ConfInfo} \sqsubseteq \mathsf{BitString}, \mathsf{EnumToCmd} \equiv \mathsf{CmdToEnum}^{\smile}, \mathsf{StrToCmd} \equiv \mathsf{CmdToStr}^{\smile}, \\
&\quad\ \mathsf{Sent} \sqsubseteq \mathsf{Command}, \big((\exists\, \mathsf{Variable}^{\smile}\,.\, \mathsf{ConfVar}) \sqcap \mathsf{BitString}\big) \sqsubseteq \mathsf{ConfInfo}\big\} \\
\mathcal{A}_{\mathsf{S}} &= \text{CMDS} \cup \text{ENUMS} \cup \text{BITS} \cup \text{AVGS} \cup \text{MAPS} \cup \\
&\quad \big\{\mathsf{Variable}\big(\mathsf{NUM\_1}, 0\big), \mathsf{Variable}\big(\mathsf{NUM\_2}, 1\big), \mathsf{Variable}\big(\mathsf{NUM\_3}, 0\big), \mathsf{Variable}\big(\mathsf{NUM\_4}, 0\big), \\
&\quad\ \mathsf{Variable}\big(\mathsf{TIME}, 3\big), \mathsf{Variable}\big(\mathsf{DELTA}, 3\big), \mathsf{ConfInfo}\big(01\big), \mathsf{ConfVar}\big(\mathsf{Y}\big), \\
&\quad\ \mathsf{CmdToStr}\big(\mathsf{ABOR}, 00\big), \mathsf{CmdToStr}\big(\mathsf{ALLO}, 01\big), \mathsf{CmdToStr}\big(\mathsf{HELP}, 10\big), \\
&\quad\ \mathsf{CmdToStr}\big(\mathsf{NOOP}, 11\big), \mathsf{Variable}\big(\mathsf{Y}, 01\big), \mathsf{Variable}\big(\mathsf{X}, \mathsf{ALLO}\big), \mathsf{Sent}\big(\mathsf{ALLO}\big)\big\}
\end{aligned}
$$

Figure 6.7: Evolved knowledge specification of agent S resulting from the simulation of its amended concrete behaviour

$$
\begin{aligned}
\mathcal{N}_{\mathsf{P}} &= (\mathcal{T}_{\mathsf{P}}, \mathcal{A}_{\mathsf{P}}) \text{ where} \\
\\
\mathcal{T}_{\mathsf{P}} &= \big\{\mathsf{EnumToCmd} \equiv \mathsf{CmdToEnum}^{\smile}, \mathsf{Received} \sqsubseteq \mathsf{Command}\big\} \\
\mathcal{A}_{\mathsf{P}} &= \text{CMDS} \cup \text{ENUMS} \cup \text{BITS} \cup \text{AVGS} \cup \text{MAPS} \cup \\
&\quad \big\{\mathsf{Variable}\big(\mathsf{NUM\_1}, 0\big), \mathsf{Variable}\big(\mathsf{NUM\_2}, 1\big), \mathsf{Variable}\big(\mathsf{NUM\_3}, 0\big), \mathsf{Variable}\big(\mathsf{NUM\_4}, 0\big), \\
&\quad\ \mathsf{Variable}\big(\mathsf{TIME}, 3\big), \mathsf{Variable}\big(\mathsf{DELTA}, 3\big), \mathsf{Variable}\big(\mathsf{CMD}, 2\big), \\
&\quad\ \mathsf{Variable}\big(\mathsf{X}, \mathsf{ALLO}\big), \mathsf{Received}\big(\mathsf{ALLO}\big)\big\}
\end{aligned}
$$

Figure 6.8: Evolved knowledge specification of agent P resulting from the simulation of its concrete behaviour

has a similar spirit in that it is meant to serve as a means for showing how agents learn new information resulting from the execution of their behaviours and their communication with other system agents. Currently, this evolution of agent knowledge is done through a manual simulation of the agent behaviour and application of the *evolve* relation presented in Section 6.3.3. A push towards automating this process is an essential part of the future work with regard to the evolution of agent knowledge.

$$
\begin{aligned}
\mathcal{N}_{\mathsf{R}} &= (\mathcal{T}_{\mathsf{R}}, \mathcal{A}_{\mathsf{R}}) \text{ where} \\
\\
\mathcal{T}_{\mathsf{R}} &= \big\{\mathsf{EnumToCmd} \equiv \mathsf{CmdToEnum}^{\smile}, \mathsf{StrToCmd} \equiv \mathsf{CmdToStr}^{\smile}, \mathsf{Sent} \sqsubseteq \mathsf{Command}, \\
&\quad \big((\exists\, \mathsf{Variable}^{\smile}\,.\,\mathsf{ConfVar}) \sqcap \mathsf{BitString}\big) \sqsubseteq \mathsf{ConfInfo}\big\} \\
\mathcal{A}_{\mathsf{R}} &= \text{CMDS} \cup \text{ENUMS} \cup \text{BITS} \cup \text{MAPS} \cup \\
&\quad \big\{\mathsf{Variable}\big(\mathsf{NUM\_1}, 0\big), \mathsf{Variable}\big(\mathsf{NUM\_2}, 1\big), \mathsf{Variable}\big(\mathsf{NUM\_3}, 0\big), \mathsf{Variable}\big(\mathsf{NUM\_4}, 0\big), \\
&\quad\ \mathsf{Variable}\big(\mathsf{AVG\_1}, 0\big), \mathsf{Variable}\big(\mathsf{AVG\_2}, 3\big), \mathsf{Variable}\big(\mathsf{AVG\_3}, 0\big), \mathsf{Variable}\big(\mathsf{AVG\_4}, 0\big), \\
&\quad\ \mathsf{Variable}\big(\mathsf{SINCE\_1}, 3\big), \mathsf{Variable}\big(\mathsf{SINCE\_2}, 0\big), \mathsf{Variable}\big(\mathsf{SINCE\_3}, 3\big), \\
&\quad\ \mathsf{Variable}\big(\mathsf{SINCE\_4}, 3\big), \mathsf{Variable}\big(\mathsf{TIME}, 3\big), \mathsf{Variable}\big(\mathsf{DELTA}, 3\big), \mathsf{Variable}\big(\mathsf{CMD}, 2\big), \\
&\quad\ \mathsf{ConfVar}\big(\mathsf{Y}\big), \mathsf{CmdToStr}\big(\mathsf{ABOR}, 00\big), \mathsf{CmdToStr}\big(\mathsf{ALLO}, 01\big), \mathsf{CmdToStr}\big(\mathsf{HELP}, 10\big), \\
&\quad\ \mathsf{CmdToStr}\big(\mathsf{NOOP}, 11\big), \mathsf{Variable}\big(\mathsf{X}, \mathsf{ALLO}\big), \mathsf{Variable}\big(\mathsf{Y}, 01\big)\big\}
\end{aligned}
$$

Figure 6.9: Evolved knowledge specification of agent R resulting from the simulation of its amended concrete behaviour

6.4 Verification of Confidential Information Leakage

The mathematical framework presented throughout this thesis can be used to verify if there is a possibility for confidential information leakage via distributed covert channels in a system of communicating agents. This section proposes guidelines for determining whether some agents can leak confidential information to other agents in a system of communicating agents by analysing the system for the necessary conditions for the existence of distributed covert channels and determining if an agent can come to know some confidential information that they are not authorised to know according to the system security policy.

In what follows, consider a system formed by a set $\mathcal{C}$ of communicating agents and a set of confidential information Φ_C which may consist of assertional axioms, terminological axioms, or both. The existence of a potential confidential information leakage is verified by performing the following steps.

Step 1: Identify Possible Sources of Confidential Information Leakage

The first step towards determining whether there exists a potential confidential information leakage via distributed covert channels in a system of communicating agents is to identify all of the possible sources of confidential information leakage and identifying whether there exists a constraint on communication in the given system of communicating agents. Recall from Section 4.3.1, that the initial knowledge of each agent $\mathsf{A} \in \mathcal{C}$ (denoted $\mathcal{N}^0_{\mathsf{A}}$) is specified using the description logic $\mathcal{ALB}$. An agent $\mathsf{A} \in \mathcal{C}$ is a *potential source of direct confidential information leakage* if and only if agent A initially knows, as a fact or by deduction, some confidential information $\big($i.e., A is a source $\iff \exists(\varphi \mid \varphi \in \Phi_C : \mathcal{N}^0_{\mathsf{A}} \models \varphi)\big)$. Any agent that has an initial knowledge of some confidential information can be a source of direct confidential information leakage. The idea is that only those agents that know some confidential information have an ability to leak that information. By considering only the potential sources of confidential information leakage, restrictions do not need to be placed on those agents which do not have the ability to leak any confidential information.

Using the notion of potential sources of confidential information leakage, a formulation of the constraint on communication condition for the existence of distributed covert channels is obtained. A system formed by a set $\mathcal{C}$ of communicating agents has a *constraint on communication* if and only if $\exists\big(\mathsf{A} \mid \mathsf{A} \in \mathcal{C} : \mathsf{A}$ is a source$\big)$. In essence, if a system of communicating agents contains at least one potential source of direct confidential information leakage, then there is a constraint on the communication of the agents in the system. The satisfaction of the constraint on communication condition indicates that there exists a constraint on the communication of some agents in a given system of communicating agents. This means that it is then necessary to determine whether there is any potential means for communicating this confidential information from a source agent to some other agent in the system that is not permitted to know this information according to the system security policy. If there is no constraint on communication in the given system of communicating

agents then there is no possibility for confidential information leakage via distributed covert channels.

Consider the running example system of communicating agents from Section 4.1. According to the security policy, the confidential information is given as the set $\Phi_C = \{\mathsf{ConfInfo}(01)\}$. Therefore, with respect to the description logic specifications for the initial knowledge base of each agent in the given system of communicating agents shown in Figures 4.10 to 4.14 in Section 4.3.1, it is easy to see that only agent S knows any confidential information in the system $\big($i.e., $\mathcal{N}_{\mathsf{S}}^0 \models \mathsf{ConfInfo}(01)\big)$. Therefore, agent S is a potential source of confidential information leakage and it follows directly that there is a constraint on communication in the given system of communicating agents. Furthermore, this can be done automatically using the SPASS theorem prover since it allows for the verification of the satisfiability of $\mathcal{ALB}$ formulae with respect to a given agent knowledge base (see Definition 3.5.3). A detailed usage of the SPASS theorem prover for verifying the constraint on communication condition for the running example of the system of communicating agents described in Section 4.1 is provided in Appendix D.

Step 2: Identify Possible Communication Paths and Information Sinks

The second step that is required for determining whether there exists a potential confidential information leakage via distributed covert channels in a system of communicating agents is to identify all of the possible communication paths from each source agent identified in Step 1 to each other agent in the system. This involves applying Definition 5.2.10 from Chapter 5 in order to verify if there exists a potential for communication between each source agent and each other agent in the system. Also, for each pair of agents for which there is a potential for communication, all possible communication paths need to be identified. This can be done automatically using the prototype tool as described in Section 5.3.1 and illustrated in Appendix C.4. Again, if there is no potential for communication from any source agent identified in Step 1 to any other agent in the given system of communicating agents, then

there is no possibility for confidential information leakage via distributed covert channels. Once again, consider the running example system of communicating agents described in Section 4.1. It was shown in Section 5.3 that there is indeed a potential for communication from agent S to each other agent in the system except for the clock agent C. In particular, there is a potential for communication from agent S to agent R (i.e., $S \rightsquigarrow^{+} R$). As an example, one possible path for achieving this is given by $\big(\mathsf{S} \rightarrow_{\mathcal{S}} \mathsf{P} \rightarrow_{\mathcal{E}} \mathsf{R}\big)$, though there are indeed three other paths. In this case, agent R is considered to be an information sink.

Step 3: Simulate Communication Paths And Evolve Agent Knowledge

Utilising each of the communication paths that were identified in Step 2, the next step for determining whether there exists a potential confidential information leakage via distributed covert channels in a system of communicating agents is to simulate the execution of each communication path and evolve the knowledge of each agent according to the procedure outlined in Section 6.3.

Consider the running example system of communicating agents given in Section 4.1, the amendments to the behaviours of the sending agent S and the receiving agent R (see Section 6.2.2) with respect to the communication scheme, and the communication path $\big(\mathsf{S} \rightarrow_{\mathcal{S}} \mathsf{P} \rightarrow_{\mathcal{E}} \mathsf{R}\big)$ that was identified in Step 2. The concrete behaviour corresponding to $\big(\mathsf{S}\,;\mathsf{P}\,;\mathsf{R}\big)$ (see Figure 6.5, Figure 4.5, and Figure 6.6) is simulated and the knowledge of each agent is evolved. The result of the evolution of the agent knowledge as the result of the simulation of the communication path $\big(\mathsf{S} \rightarrow_{\mathcal{S}} \mathsf{P} \rightarrow_{\mathcal{E}} \mathsf{R}\big)$ is given in Figures 6.7 to 6.9 in Section 6.3.

Step 4: Check for Security Policy Violations

The last step in verifying whether there exists a potential confidential information leakage via distributed covert channels in a system of communicating agents is to determine whether there is a violation of the system security policy after simulating the behaviour of the

system and evolving the agent knowledge accordingly. This involves checking whether any information sink knows some information that it is not authorised to know according to a given security policy. Suppose Φ_C denotes the set of confidential information protected by the system security policy. Then, this step involves verifying the following condition for each evolved agent knowledge resulting from the simulation of each communication path in Step 3:

$$\exists(\varphi \mid \varphi \in \Phi_C : \mathcal{N}_{\mathsf{sink}} \models \varphi)$$

where sink represents the agent acting as an information sink for the given communication path.

With regard to the running example system of communicating agents from Section 4.1, the security policy states that the set $\Phi_C = \{\mathsf{ConfInfo}(\mathsf{01})\}$ constitutes the confidential information for the system. Also, the security policy for this system explicitly forbids any agent, other than agent S, from knowing or possessing this information. Therefore, consider the evolved knowledge of agent R given in Figure 6.9 which corresponds to the simulation of the communication path $(\mathsf{S} \rightarrow_{\mathcal{S}} \mathsf{P} \rightarrow_{\mathcal{E}} \mathsf{R})$ that was identified in Step 2; in this case $\mathsf{sink} = \mathsf{R}$. It turns out that the condition given above evaluates to true for this case. This means that there is indeed a confidential information leakage resulting from this communication path in the given system of communicating agents. Specifically, agent R learns the confidential information indirectly from agent S. Furthermore, this result can be automatically verified using the SPASS theorem prover similar to the verification of the constraint on communication condition discussed in Step 1 (see Appendix D.2.6 for a representation of the evolved knowledge of agent R shown in Figure 6.9 used to support the automation of the proposed verification of confidential information leakage).

6.4.1 Discussion and Related Work

The order of the steps presented above is important from a computational perspective. First, the constraint on communication condition needs to be verified in order to determine whether there is a constraint on communication in the given system of communicating agents. If no such constraint exists, then there is no need to verify the potential for communication condition since all agents would be allowed to communicate freely. Furthermore, the constraint on communication condition allows for the development of a list of agents that have confidential information in their respective data stores, thereby allowing them to be the source agents in the potential for communication condition. By having this information, it is not necessary to look for a potential for communication from agents which cannot be source agents. This reduces the total number of possible patterns of communication and communication paths to consider.

While there exists a wide variety of techniques which aim to detect and prevent covert channel usage in systems of communicating agents (see Section 2.1.3 and Section 2.1.4 for a survey of existing covert channel detection and prevention techniques), there does not exist a technique, similar to that presented above, that takes into account the knowledge of agents in the given system under consideration. Also, no existing technique examines the problem of confidential information leakage by analysing the necessary conditions for covert channel existence. In this way, the proposed guidelines can serve as the basis for further developing approaches for mitigating covert channels in systems of communicating agents and for protecting the confidentiality of information.

6.5 Conclusion

This chapter provided an outlook for the proposed mathematical framework for the modelling, analysis, and mitigation of distributed covert channels in advancing the understanding of covert channels by investigating further applications of the proposed framework. First,

this chapter articulated a representation for communication schemes based on description logic knowledge bases and pre- and post-condition specifications. Then, it was shown how a communication scheme can be derived from the initial specification of a system of communicating agents where the behaviour of each agent is specified using the mathematical framework of Communicating Concurrent Kleene Algebra (C^2KA) and the knowledge of each agent is specified using the description logic $\mathcal{ALB}$. After that, this chapter presented an approach for systematically merging a communication scheme into a specification of a system of communicating agents. The proposed approach results in an amended specification of a system of communicating agents which allows for a designated sender and receiver to establish and operate a distributed covert channel according to the given communication scheme. In particular, the approach highlighted how covert channels can be designed and deployed early in the development of a system of communicating agents. This chapter also discussed an approach for the evolution of agent knowledge through the execution of concrete agent behaviours. Finally, a set of guidelines for using the mathematical framework presented throughout this thesis to identify potential confidential information leakage via distributed covert channels in systems of communicating agents was provided. This included a formulation and verification approach for the constraint on communication condition for the existence of distributed covert channels. It is important to note that this chapter merely presented a number of guidelines and outlined a variety of approaches which point to ways in which the proposed mathematical framework for the modelling, analysis, and mitigation of distributed covert channels can be used to enhance the understanding of covert channels in systems of communicating agents and to offer a means for identifying leakages of confidential information. This chapter is meant to serve as the basis for future work with regard to the mathematical framework presented in this thesis and requires further exploration, formulation, and articulation.

Chapter 7

Discussion, Conclusion, and Future Work

As discussed in Section 1.4.1, there is a lack of formal methods for dealing with distributed covert channels in modern computer systems. In an effort to address this issue, this thesis has presented a mathematical framework for the modelling, analysis, and mitigation of distributed covert channels. This framework aids in advancing the current understanding of covert channels. This chapter summarises and discusses the contributions of this thesis and points to future research directions resulting from this work. Specifically, Section 7.1 highlights, discusses, and assesses the contributions that are made by this thesis. Section 7.2 suggests avenues for future work resulting from the proposed mathematical framework and its applications and tools. Finally, Section 7.3 makes final comments and closing remarks.

7.1 Highlights of the Contributions

The contributions related to the proposed a mathematical framework for the modelling, analysis, and mitigation of distributed covert channels include:

(1) **Necessary Conditions for the Existence of Distributed Covert Channels**: This thesis proposed a set of necessary conditions for the existence of distributed covert channels in systems of communicating agents. If there is a covert channel in a given system of communicating agents, then the *potential for communication* condition and the *constraint on communication* condition are satisfied. These conditions aid in advancing the current understanding of covert channels and serve as a basis for developing effective and efficient mechanisms for mitigating distributed covert channels in systems of communicating agents.

(2) **Specification of Concurrent and Communicating Agent Behaviour**: This thesis developed a mathematical framework called Communicating Concurrent Kleene Algebra (C^2KA). It an extension to the algebraic model of concurrent Kleene algebra (CKA) first presented by Hoare et al. [HMSW09a, HMSW09b, HMSW10, HMSW11]. C^2KA allows for the concurrent and communicating behaviour of systems of communicating agents with respect to the complexities of distributed covert channels to be captured. It also allows for the separation of communicating and concurrent behaviour in a system and its environment and for the expression of the influence of external stimuli on the behaviours of a system of agents. C^2KA builds atop well-established foundations and inherits most, if not all, of the theory that has been previously developed with respect to concurrent Kleene algebra. In this way, it gains the power and flexibility to specify all that can be done with existing formalisms while allowing for expansion beyond existing limitations. Specifically, C^2KA allows for the handling of open systems with the notion of external stimuli coming from outside the boundaries of the system being considered whereas existing formalisms, such as CKA, work only within closed systems.

(3) **Specification of Agent Knowledge**: This thesis provided a representation of agent knowledge using the description logic $\mathcal{ALB}$ [HS00]. The proposed representation gives the power and flexibility to reason on agent knowledge at both the terminological or

conceptual level and at the assertional or object level. It also allows for the inference of implicitly represented knowledge from the knowledge that is explicitly contained in a knowledge base. Furthermore, the proposed description logic representation of agent knowledge plays a role in studying the interplay between the knowledge and behaviour of the agents in systems of communicating agents.

(4) **Formulation and Verification of the Potential for Communication Condition**: This thesis proposed a formulation and verification approach for the potential for communication condition for the existence of distributed covert channels. The formulation is based on the mathematical framework of C^2KA. The potential for communication amongst agents was considered from two complementary perspectives. First, the potential for communication via external stimuli examined how stimuli generated from one agent in the system are able to influence the behaviour of other agents in the system. Second, the potential for communication via shared environments studied how communication can occur through shared events or variables and the dependencies between them. Then, it was shown how the satisfaction of the potential for communication condition could be formally verified for a system of communicating agents. The automation of the proposed verification approach using a prototype tool was also discussed.

(5) **Formulation and Verification of the Constraint on Communication Condition**: This thesis proposed a formulation and verification approach for the constraint on communication condition for the existence of distributed covert channels. The formulation is based on description logic [BMNP03]. The use of description logic provided the ability to reason about the knowledge of agents in terms of what an agent knows, or can come to know. Then, it was shown how the satisfaction of the constraint on communication condition could be formally verified for a given system of communicating agents. The use of the SPASS theorem prover as a means for supporting the automation of the proposed verification approach was also discussed.

(6) **Guidelines for Better Understanding Covert Channels and Attaining Systems Resilient to Their Existence and Use**: This thesis presented a number of guidelines resulting from applications of the proposed mathematical framework and the formulation of the potential for communication and constraint on communication conditions for the existence of distributed covert channels. First, a representation and classification for communication schemes was given. This classification provides useful information for anticipating the kind of amendments and reasoning that can be performed when a communication scheme is incorporated into a system of communicating agents for the purpose of leaking confidential information via distributed covert channels. Then, a set of guidelines for modularly developing and merging covert communication schemes into systems of communicating agents was discussed. This approach illustrates how covert channels can be designed early in the development of a system of communicating agents, particularly at the system specification level. After that, a set of guidelines for determining how the knowledge of agents in systems of communicating agents evolve through the execution of concrete agent behaviours. Finally, an approach for the verification of confidential information leakage in systems of communicating agents via the establishment and operation of distributed covert channels was articulated. The proposed approach is based on the verification of the necessary conditions for the existence of distributed covert channels and the guidelines for evolving agent knowledge through the execution of concrete agent behaviours. These contributions provide the basis for a comprehensive covert channel analysis of systems of communicating agents and aid in the advancement of mechanisms for reducing the threat of covert channels in systems of communicating agents. As a whole, these contributions encompass part of a foundation for proposing guidelines for designing and implementing systems of communicating agents that are resilient to covert channels. This contribution merely scratches the surface in terms of what can be done with the proposed framework. Much more investigation and effort is required to further articulate these ideas.

7.2 Future Work

The mathematical framework presented in this thesis can be extended in a number of different directions. The following subsections describe possible extensions and further work with respect to the proposed theory on which the proposed framework is based, applications of the proposed framework, and tools for automating aspects of the proposed framework.

7.2.1 Theory: Models and Techniques

Concerning the proposed mathematical theory for capturing the behaviour and knowledge of agents in systems of communicating agents, the following directions can be explored:

(1) An investigation into providing more, and potentially better models, of C^2KA (and even CKA) ought to be undertaken. Section 4.2.5 commented on a model for the theory of C^2KA. This discussion served the purpose of showing that there exists at least one model for the proposed theory. However, it is acknowledged that there needs to be a push towards finding other models for the theory. This has been left as future work since the primary focus and goal of this thesis was not to develop a brand new theory of concurrency and communication, but rather to investigate the formalisation of the necessary conditions for the existence of distributed covert channels in systems of communicating agents.

(2) Further investigation into the interplay between external stimuli and the parallel composition operator $*$ is needed. This can lead to a more accurate view of the concurrent behaviours of agents in systems of communicating agents. It is also possible that a further study of how external stimuli interact with the parallel composition of agent behaviours can allow for the relaxation of the interleaving view of concurrency that is currently taken in this thesis.

(3) A further study into the relationship between the abstract and concrete specifications of systems of communicating agents using C^2KA is needed. For example, one can study how different formal specification languages can be used in place of Dijkstra's guarded command language in the concrete behaviour specification of agents. This can allow for different concrete representations of external stimuli, for example. Moreover, such an investigation can allow for different ways to reason about programs in concrete terms.

(4) There are a number of directions that can be explored with regard to extending the mathematical theory of C^2KA. For example, the addition of timing information to C^2KA in order to attain a kind of timed C^2KA would be useful, particularly in terms of its applicability in handling covert timing channels. Similarly, an investigation towards the development of a kind of probabilistic C^2KA, where agents respond to particular stimuli with some probability, would be of interest. Such an extension can provide a rich formalism for developing applications of C^2KA in domains such as social networks and offer some power for making predications about agent behaviours.

(5) With regard to the representation and theory of agent knowledge, there are a number of future research directions. For example, the relaxation of the assumptions, specifically those outlined in Section 6.3.1, such as allowing for the existence of misinforming agents in systems of communicating agents or for the introduction of a notion of "forgetting" where agents can remove information from their knowledge, can allow for the representation of more kinds of systems of communicating agents and communication schemes. Additionally, extensions which incorporate the introduction and handling of inconsistencies in agent knowledge may provide a much more accurate representation of real systems of communicating agents.

(6) The material presented in Chapter 6 can serve as the basis for many future research directions. In particular, work can be done in order to further articulate the details of the proposed approach for merging communication schemes into specifications of systems of

communicating agents. For example, this can involve an examination of ways in which behaviours can be systematically or automatically generated such that they satisfy the pre- and post-condition specifications outlined in the behaviour component of a communication scheme. It can also involve a formal verification that the resulting amended behaviours satisfy their pre- and post-condition specifications. As mentioned in Section 6.2.2, this requires a translation between the representations of agent knowledge and the representation of agent behaviours. Additionally, the approach for evolving the knowledge of agents resulting from the execution of concrete agent behaviours needs to be explored further. Also, a push towards automating the proposed approach for verifying confidential information leakage via distributed covert channels in systems of communicating agents needs to be made.

7.2.2 Applications

With respect to the possible applications of the proposed mathematical framework, the following directions can be investigated further:

(1) The range of the application domain of the mathematical framework of Communicating Concurrent Kleene Algebra (C^2KA) can explored further. Such domains may include distributed system architectures where there are a number of communicating agents with complex behaviours such as social networks, mobile device platforms, cloud computing, wireless sensor networks, and discrete event systems and supervisory control. This can lead to new and innovative ways to think and reason about such systems. For example, by modelling social networks using C^2KA, it may be possible to make predications about the behaviours of online communities which has been of interest in the areas of marketing and targeted adverting.

(2) A study of how the proposed mathematical framework can be used to support the mitigation of covert channels in systems of communicating agents is another fruitful future research opportunity. For example, the mitigation of covert channels in systems of communicating agents can proceed by studying ways in which systems can be analysed for the satisfaction of the necessary conditions for the existence of distributed covert channels. The development of approaches for falsifying the necessary conditions for the existence of distributed covert channels can lead to the elimination of the possibility for the existence and usage of covert channels in systems of communicating agents. This gives two fronts from which this can be addressed. First, ways in which the potential for communication among system agents can be disrupted or eliminated by modifying and restricting agent behaviours in a given system can be explored. Second, an examination of how to restrict the knowledge of system agents that are on a communication path so that they are unable to know any fragments of confidential information that can be leaked via covert channels and then reconstituted by the receiver can be examined.

(3) An exploration into how the proposed mathematical framework could be used to reverse engineer binaries for agents in systems of communicating agents for the purpose of devising covert communication schemes in order to perform code injections to establish and operate covert channels in the given system can be undertaken. This can lead to new and interesting ways in which to think about protecting systems of communicating agents from being exploited to harbour covert channels.

(4) Further investigations into how covert communication schemes can be developed such that covert channels that exhibit a set of desirable properties, such as high stealth and robustness can be explored. This can lead to new and innovative ways to construct covert channels. In turn, this can be used in new applications of covert channels such as key distribution in wireless sensor networks, for example.

7.2.3 Tools and Automation

This thesis developed a prototype tool to help in automating the specification of systems of communicating agents and in verifying the satisfaction of the potential for communication condition for the existence of distributed covert channels. Moreover, the SPASS theorem prover [TST14] has been used in order to support the automation of the verification of the constraint on communication condition for the existence of distributed covert channels. There are a number of ways in which these tools can be enhanced and extended to provide a more comprehensive tool for analysing systems of communicating agents for the existence and usage of distributed covert channels.

(1) The prototype tool can be enhanced by further testing and analysing its design, especially with respect to the use of the Maude term rewriting system [CDE$^+$03]. In particular, further work into investigating ways in which the tool can be optimised to provide results in a more timely manner is required.

(2) The prototype tool can be extended by incorporating the functionality to support the automation of the approach for merging communication schemes into specifications of systems of communicating agents, as well as the approach for detecting confidential information leakage in systems of communicating agents via distributed covert channels.

(3) It would be beneficial for the prototype tool to interface directly with the SPASS theorem prover so that the analysis and verification of both of the necessary conditions for the existence of distributed covert channels in systems of communicating agents could be handled within the same tool.

7.3 Closing Remarks

The elimination of covert channels from systems of communicating agents is an ongoing and ambitious endeavour, particularly with the respect to the increasing connectedness and complexity of modern computer systems. According to [WJ06], human intervention is a permanent part of every covert channel mitigation technique. The difference among the techniques is how much each depends on the human being. Human intervention plays a role on both sides of the covert channel fight. Analysts hoping to eliminate covert channels from systems of communicating agents often must perform a significant amount of labour in order to model, identify, analyse, and mitigate the use of covert channels, while users of covert channels are constantly evolving ways in which covert channels can be established and operated in order to evade detection and prevention techniques. The human aspect of covert channels increases the challenge of eliminating them in systems of communicating agents. In short, more attention needs to be placed on decreasing the amount of labour required of covert channel analysts and the focus must be turned towards evolving and enhancing the understanding of covert channels in order to keep up with the evolution of covert channel users.

Appendix A

Detailed Proofs

This appendix contains the detailed proofs of the propositions and corollaries presented in this thesis.

A.1 Detailed Proof of Proposition 3.2.2

For all $a \in K$, $a^{\textcircled{;}} \leq a^{\circledast}$.

$a^{\textcircled{;}} \leq a^{\circledast}$

$\Longleftrightarrow$ $\langle$ Right Identity of $;$ $\rangle$

$a^{\textcircled{;}} ; 1 \leq a^{\circledast}$

$\Longleftarrow$ $\langle$ Definition 3.1.6(3) for $;$ $\rangle$

$1 + a ; a^{\circledast} \leq a^{\circledast}$

$\Longleftrightarrow$ $\langle$ Definition 3.1.6(1) for $^{\circledast}$ $\rangle$

$1 + a ; a^{\circledast} \leq 1 + a * a^{\circledast}$

$\Longleftrightarrow$ $\langle$ Proposition 3.2.1(3) $\rangle$

true

A.2 Detailed Proof of Proposition 4.2.1

Let $\big(\mathcal{S}, \mathcal{K}\big)$ be a C^{2}KA. For all $a, b \in K$ and $s, t \in S$:

(1) $a \leq_{\mathcal{K}} b \;\wedge\; s \leq_{\mathcal{S}} t \implies s \circ a \leq_{\mathcal{K}} t \circ b$

$s \circ a \leq_{\mathcal{K}} t \circ b$

$\iff$ $\langle$ Definition of $\leq_{\mathcal{K}}$ $\rangle$

$s \circ a + t \circ b = t \circ b$

$\iff$ $\langle$ Hypothesis: $s \leq_{\mathcal{S}} t$ $\rangle$

$s \circ a + (s \oplus t) \circ b = t \circ b$

$\iff$ $\langle$ Definition 3.1.8(2) for $\big({}_{\mathcal{S}}K, +\big)$ $\rangle$

$s \circ a + s \circ b + t \circ b = t \circ b$

$\iff$ $\langle$ Definition 3.1.8(1) for $\big({}_{\mathcal{S}}K, +\big)$ $\rangle$

$s \circ (a + b) + t \circ b = t \circ b$

$\iff$ $\langle$ Hypothesis: $a \leq_{\mathcal{K}} b$ $\rangle$

$s \circ b + t \circ b = t \circ b$

$\iff$ $\langle$ Definition 3.1.8(2) for $\big({}_{\mathcal{S}}K, +\big)$ $\rangle$

$(s \oplus t) \circ b = t \circ b$

$\iff$ $\langle$ Hypothesis: $s \leq_{\mathcal{S}} t$ $\rangle$

$t \circ b = t \circ b$

$\iff$ $\langle$ Reflexivity of $=$ $\rangle$

true

(2) $a \leq_{\mathcal{K}} b \ \wedge \ s \leq_{\mathcal{S}} t \implies \lambda(s,a) \leq_{\mathcal{S}} \lambda(t,b)$

$\lambda(s,a) \leq_{\mathcal{S}} \lambda(t,b)$

$\iff$ $\langle$ Definition of $\leq_{\mathcal{S}}$ $\rangle$

$\lambda(s,a) \oplus \lambda(t,b) = \lambda(t,b)$

$\iff$ $\langle$ Hypothesis: $s \leq_{\mathcal{S}} t$ $\rangle$

$\lambda(s,a) \oplus \lambda\big((s \oplus t), b\big) = \lambda(t,b)$

$\iff$ $\langle$ Definition 3.1.8(1) for $\big(S_{\mathcal{K}}, \oplus\big)$ $\rangle$

$\lambda(s,a) \oplus \lambda(s,b) \oplus \lambda(t,b) = \lambda(t,b)$

$\iff$ $\langle$ Definition 3.1.8(2) for $\big(S_{\mathcal{K}}, \oplus\big)$ $\rangle$

$\lambda\big(s, (a + b)\big) \oplus \lambda(t,b) = t \circ b$

$\iff$ $\langle$ Hypothesis: $a \leq_{\mathcal{K}} b$ $\rangle$

$\lambda(s,b) \oplus \lambda(t,b) = \lambda(t,b)$

$\iff$ $\langle$ Definition 3.1.8(1) for $\big(S_{\mathcal{K}}, \oplus\big)$ $\rangle$

$\lambda\big((s \oplus t), b\big) = \lambda(t,b)$

$\iff$ $\langle$ Hypothesis: $s \leq_{\mathcal{S}} t$ $\rangle$

$\lambda(t,b) = \lambda(t,b)$

$\iff$ $\langle$ Reflexivity of $=$ $\rangle$

true

A.3 Detailed Proof of Corollary 4.2.2

Let $(\mathcal{S}, \mathcal{K})$ be a C^{2}KA where the underlying CKA and stimulus structure are built up from quantales. For all $a, b \in K$ and $s, t \in S$:

(1) $a \leq_{\mathcal{K}} b \implies s \circ a \leq_{\mathcal{K}} s \circ b$

$s \circ a \leq_{\mathcal{K}} s \circ b$

$\Longleftarrow$ $\langle$ Proposition 4.2.1(1) $\rangle$

$a \leq_{\mathcal{K}} b \ \wedge \ s \leq_{\mathcal{S}} s$

$\Longleftrightarrow$ $\langle$ Hypothesis: $a \leq_{\mathcal{K}} b$ & Reflexivity of $\leq_{\mathcal{S}}$ $\rangle$

true

(2) $s \leq_{\mathcal{S}} t \implies s \circ a \leq_{\mathcal{K}} t \circ a$

$s \circ a \leq_{\mathcal{K}} t \circ a$

$\Longleftarrow$ $\langle$ Proposition 4.2.1(1) $\rangle$

$a \leq_{\mathcal{K}} a \ \wedge \ s \leq_{\mathcal{S}} t$

$\Longleftrightarrow$ $\langle$ Hypothesis: $s \leq_{\mathcal{S}} t$ & Reflexivity of $\leq_{\mathcal{K}}$ $\rangle$

true

(3) $s \circ (a \,;\, b + b \,;\, a) \leq_{\mathcal{K}} s \circ (a * b)$

$s \circ (a \,;\, b + b \,;\, a) \leq_{\mathcal{K}} s \circ (a * b)$

$\Longleftarrow$ $\langle$ Corollary 4.2.2(1) $\rangle$

$a \,;\, b + b \,;\, a \leq_{\mathcal{K}} a * b$

$\Longleftrightarrow$ $\langle$ Proposition 3.2.1(1) & Proposition 3.2.1(3) & Idempotence of $+$ $\rangle$

true

(4) $s \circ a^{\textcircled{:}} \leq_{\mathcal{K}} s \circ a^{\circledast}$

$s \circ a^{\textcircled{:}} \leq_{\mathcal{K}} s \circ a^{\circledast}$

$\Longleftarrow$ $\langle$ Corollary 4.2.2(1) $\rangle$

$a^{\textcircled{:}} \leq_{\mathcal{K}} a^{\circledast}$

$\Longleftrightarrow$ $\langle$ Proposition 3.2.2 $\rangle$

true

(5) $s \circ a^{\textcircled{:}} = +(n \mid n \geq 0 : s \circ a^n)$ Recall the definition of $a^{\textcircled{:}}$:

$$a^{\textcircled{:}} = +(n \mid n \geq 0 : a^n) \tag{A.1}$$

where

$$a^0 \stackrel{\text{def}}{=} 1$$
$$a^{n+1} \stackrel{\text{def}}{=} a^n ; a$$

$s \circ a^{\textcircled{:}}$

$=$ $\langle$ Definition of $a^{\textcircled{:}}$: Equation (A.1) $\rangle$

$s \circ +(n \mid n \geq 0 : a^n)$

$=$ $\langle$ Definition 3.1.8(1) for $\left({}_{\mathcal{S}}K, +\right)$ $\rangle$

$+(n \mid n \geq 0 : s \circ a^n)$

(6) $s \leq_{\mathcal{S}} t \implies \lambda(s,a) \leq_{\mathcal{S}} \lambda(t,a)$

$\lambda(s,a) \leq_{\mathcal{S}} \lambda(t,a)$

$\Longleftarrow$ $\langle$ Proposition 4.2.1(2) $\rangle$

$a \leq_{\mathcal{K}} a \ \wedge \ s \leq_{\mathcal{S}} t$

$\Longleftrightarrow$ $\langle$ Hypothesis: $s \leq_{\mathcal{S}} t$ & Reflexivity of $\leq_{\mathcal{K}}$ $\rangle$

true

(7) $a \leq_{\mathcal{K}} b \implies \lambda(s,a) \leq_{\mathcal{S}} \lambda(s,b)$

$\lambda(s,a) \leq_{\mathcal{S}} \lambda(s,b)$

$\Longleftarrow$ $\langle$ Proposition 4.2.1(2) $\rangle$

$a \leq_{\mathcal{K}} b \ \wedge \ s \leq_{\mathcal{S}} s$

$\Longleftrightarrow$ $\langle$ Hypothesis: $a \leq_{\mathcal{K}} b$ & Reflexivity of $\leq_{\mathcal{S}}$ $\rangle$

true

(8) $\lambda(s,(a\,;b+b\,;a)) \leq_{\mathcal{S}} \lambda(s,(a*b))$

$\lambda(s,(a\,;b+b\,;a)) \leq_{\mathcal{S}} \lambda(s,(a*b))$

$\Longleftarrow$ $\langle$ Corollary 4.2.2(7) $\rangle$

$a\,;b+b\,;a \leq_{\mathcal{K}} a*b$

$\Longleftrightarrow$ $\langle$ Proposition 3.2.1(1) & Proposition 3.2.1(3) & Idempotence of $+$ $\rangle$

true

(9) $\lambda(s, a^{\textcircled{:}}) \leq_{\mathcal{S}} \lambda(s, a^{\circledast})$

$\lambda(s, a^{\textcircled{:}}) \leq_{\mathcal{S}} \lambda(s, a^{\circledast})$

$\Longleftarrow$ $\langle$ Corollary 4.2.2(7) $\rangle$

$a^{\textcircled{:}} \leq_{\mathcal{K}} a^{\circledast}$

$\Longleftrightarrow$ $\langle$ Proposition 3.2.2 $\rangle$

true

(10) $\lambda(s, a^{\textcircled{:}}) = \oplus(n \mid n \geq 0 : \lambda(s, a^n))$

$\lambda(s, a^{\textcircled{:}})$

$=$ $\langle$ Definition of $a^{\textcircled{:}}$: Equation (A.1) $\rangle$

$\lambda(s, +(n \mid n \geq 0 : a^n))$

$=$ $\langle$ Definition 3.1.8(2) for $\big(S_{\mathcal{K}}, \oplus\big)$ $\rangle$

$\oplus(n \mid n \geq 0 : s \circ a^n)$

A.4 Detailed Proof of Proposition 4.2.3

Let $\big(\mathcal{S}, \mathcal{K}\big)$ be a C^{2}KA. For all $a, b \in K$, $a \leq_{\mathcal{K}} b \implies \text{Orb}(a) \lessdot_{\mathcal{K}} \text{Orb}(b)$.

$\text{Orb}(a) \lessdot_{\mathcal{K}} \text{Orb}(b)$

$\Longleftrightarrow$ $\langle$ Definition 4.2.5 $\rangle$

$\forall\big(x \mid x \in \text{Orb}(a) : \exists(y \mid y \in \text{Orb}(b) : x \leq_{\mathcal{K}} y)\big)$

$\Longleftrightarrow$ $\langle$ Trading for $\forall$ & Trading for $\exists$ $\rangle$

$$\forall\big(x \mid: x \in \mathrm{Orb}(a) \implies \exists(y \mid: y \in \mathrm{Orb}(b) \ \wedge\ x \leq_{\mathcal{K}} y)\big)$$

$\Longleftrightarrow$ $\langle$ Set Membership Axiom $\rangle$

$$\forall\big(x \mid: \exists(s \mid s \in S : s \circ a = x) \implies \exists\big(y \mid: \exists(s \mid s \in S : s \circ b = y) \ \wedge\ x \leq_{\mathcal{K}} y\big)\big)$$

$\Longleftrightarrow$ $\langle$ Distributivity of $\wedge$ over $\exists$ $\rangle$

$$\forall\big(x \mid: \exists(s \mid s \in S : s \circ a = x) \implies \exists\big(y \mid: \exists(s \mid s \in S : s \circ b = y \ \wedge\ x \leq_{\mathcal{K}} y)\big)\big)$$

$\Longleftrightarrow$ $\langle$ Interchange of Dummies $\rangle$

$$\forall\big(x \mid: \exists(s \mid s \in S : s \circ a = x) \implies \exists\big(s \mid s \in S : \exists(y \mid: s \circ b = y \ \wedge\ x \leq_{\mathcal{K}} y)\big)\big)$$

$\Longleftrightarrow$ $\langle$ Trading for $\exists$ $\rangle$

$$\forall\big(x \mid: \exists(s \mid s \in S : s \circ a = x) \implies \exists\big(s \mid s \in S : \exists(y \mid s \circ b = y : x \leq_{\mathcal{K}} y)\big)\big)$$

$\Longleftrightarrow$ $\langle$ One-Point Rule $\rangle$

$$\forall\big(x \mid: \exists(s \mid s \in S : s \circ a = x) \implies \exists\big(s \mid s \in S : x \leq_{\mathcal{K}} s \circ b\big)\big)$$

$\Longleftarrow$ $\langle$ Monotonic $\exists$-Body $\rangle$

$$\forall\big(x \mid: \forall(s \mid s \in S : s \circ a = x \implies x \leq_{\mathcal{K}} s \circ b)\big)$$

$\Longleftrightarrow$ $\langle$ Definition of $\implies$ $\rangle$

$$\forall\big(x \mid: \forall(s \mid s \in S : s \circ a \neq x \ \vee\ x \leq_{\mathcal{K}} s \circ b)\big)$$

$\Longleftrightarrow$ $\langle$ Definition of $\leq_{\mathcal{K}}$ $\rangle$

$$\forall\big(x \mid: \forall(s \mid s \in S : s \circ a \neq x \ \vee\ x + (s \circ b) = s \circ b)\big)$$

$\Longleftarrow$ $\langle$ Hypothesis: $a \leq_{\mathcal{K}} b$ & Proposition 4.2.1(1) $\rangle$

$$\forall\big(x \mid: \forall(s \mid s \in S : \mathsf{true})\big)$$

$\Longleftrightarrow$ $\langle$ $\forall$-True Body $\rangle$

true

A.5 Detailed Proof of Corollary 4.2.4

Let $\big(\mathcal{S}, \mathcal{K}\big)$ be a C^{2}KA. For all $a, b, c, d \in K$:

(1) $\mathrm{Orb}(a) \lessdot_{\mathcal{K}} \mathrm{Orb}(a + b)$

$\mathrm{Orb}(a) \lessdot_{\mathcal{K}} \mathrm{Orb}(a + b)$

$\Longleftarrow$ $\langle$ Proposition 4.2.3 $\rangle$

$a \leq_{\mathcal{K}} a + b$

$\Longleftrightarrow$ $\langle$ Definition of $\leq_{\mathcal{K}}$ & Idempotence of $+$ $\rangle$

$a + b = a + b$

$\Longleftrightarrow$ $\langle$ Reflexivity of $=$ $\rangle$

true

(2) $\mathrm{Orb}((a * b)\,;(c * d)) \lessdot_{\mathcal{K}} \mathrm{Orb}((a\,;c) * (b\,;d))$

$\mathrm{Orb}((a * b)\,;(c * d)) \lessdot_{\mathcal{K}} \mathrm{Orb}((a\,;c) * (b\,;d))$

$\Longleftarrow$ $\langle$ Proposition 4.2.3 $\rangle$

$(a * b)\,;(c * d) \leq_{\mathcal{K}} (a\,;c) * (b\,;d)$

$\Longleftrightarrow$ $\langle$ Proposition 3.2.1(2) $\rangle$

true

(3) $\mathrm{Orb}(a\,;b) \lessdot_{\mathcal{K}} \mathrm{Orb}(a * b)$

$\mathrm{Orb}(a\,;b) \lessdot_{\mathcal{K}} \mathrm{Orb}(a * b)$

$\Longleftarrow$ $\langle$ Proposition 4.2.3 $\rangle$

$a\,;b \leq_{\mathcal{K}} a * b$

$\Longleftrightarrow$ $\langle$ Proposition 3.2.1(3) $\rangle$

true

(4) $\mathrm{Orb}(a\,;b + b\,;a) \lessdot_{\mathcal{K}} \mathrm{Orb}(a * b)$

$\mathrm{Orb}(a\,;b + b\,;a) \lessdot_{\mathcal{K}} \mathrm{Orb}(a * b)$

$\Longleftarrow$ $\langle$ Proposition 4.2.3 $\rangle$

$a\,;b + b\,;a \leq_{\mathcal{K}} a * b$

$\Longleftrightarrow$ $\langle$ Proposition 3.2.1(1) & Proposition 3.2.1(3) & Idempotence of $+$ $\rangle$

true

(5) $\mathrm{Orb}((a * b)\,;c) \lessdot_{\mathcal{K}} \mathrm{Orb}(a * (b\,;c))$

$\mathrm{Orb}((a * b)\,;c) \lessdot_{\mathcal{K}} \mathrm{Orb}(a * (b\,;c))$

$\Longleftarrow$ $\langle$ Proposition 4.2.3 $\rangle$

$(a * b)\,;c \leq_{\mathcal{K}} a * (b\,;c)$

$\Longleftrightarrow$ $\langle$ Proposition 3.2.1(4) $\rangle$

true

(6) $\mathrm{Orb}(a\,;(b * c)) \lessdot_{\mathcal{K}} \mathrm{Orb}((a\,;b) * c)$

$\mathrm{Orb}(a\,;(b * c)) \lessdot_{\mathcal{K}} \mathrm{Orb}((a\,;b) * c)$

$\Longleftarrow$ $\langle$ Proposition 4.2.3 $\rangle$

$a\,;(b * c) \leq_{\mathcal{K}} (a\,;b) * c$

$\Longleftrightarrow$ $\langle$ Proposition 3.2.1(5) $\rangle$

true

(7) $\mathrm{Orb}(a^{\textcircled{;}}) \lessdot_{\mathcal{K}} \mathrm{Orb}(a^{\circledast})$

$\mathrm{Orb}(a^{\textcircled{;}}) \lessdot_{\mathcal{K}} \mathrm{Orb}(a^{\circledast})$

$\Longleftarrow$ $\langle$ Proposition 4.2.3 $\rangle$

$a^{\textcircled{;}} \leq_{\mathcal{K}} a^{\circledast}$

$\Longleftrightarrow$ $\langle$ Proposition 3.2.2 $\rangle$

true

(8) $\mathrm{Orb}(a) \lessdot_{\mathcal{K}} \mathrm{Orb}(c) \;\wedge\; \mathrm{Orb}(b) \lessdot_{\mathcal{K}} \mathrm{Orb}(c) \;\Longleftrightarrow\; \mathrm{Orb}(a) \;\cup\; \mathrm{Orb}(b) \lessdot_{\mathcal{K}} \mathrm{Orb}(c)$

$\mathrm{Orb}(a) \;\cup\; \mathrm{Orb}(b) \lessdot_{\mathcal{K}} \mathrm{Orb}(c)$

$\Longleftrightarrow$ $\langle$ Definition 4.2.5 $\rangle$

$\forall\big(x \mid x \in \mathrm{Orb}(a) \;\cup\; \mathrm{Orb}(b) : \exists(y \mid y \in \mathrm{Orb}(c) : x \leq_{\mathcal{K}} y)\big)$

$\Longleftrightarrow$ $\langle$ Set Union Axiom $\rangle$

$\forall\big(x \mid x \in \mathrm{Orb}(a) \;\vee\; x \in \mathrm{Orb}(b) : \exists(y \mid y \in \mathrm{Orb}(c) : x \leq_{\mathcal{K}} y)\big)$

$\Longleftrightarrow$ $\langle$ Range Split $\rangle$

$$
\begin{array}{l}
\quad \forall\big(x \mid x \in \mathrm{Orb}(a) : \exists(y \mid y \in \mathrm{Orb}(c) : x \leq_{\mathcal{K}} y)\big) \wedge \\
\quad \forall\big(x \mid x \in \mathrm{Orb}(b) : \exists(y \mid y \in \mathrm{Orb}(c) : x \leq_{\mathcal{K}} y)\big) \\
\iff \qquad \langle \text{ Definition 4.2.5 } \rangle \\
\quad \mathrm{Orb}(a) \lessdot_{\mathcal{K}} \mathrm{Orb}(c) \wedge \mathrm{Orb}(b) \lessdot_{\mathcal{K}} \mathrm{Orb}(c)
\end{array}
$$

A.6 Detailed Proof of Proposition 4.2.5

For all $a, b \in K$, $\mathrm{Stab}(a) \cap \mathrm{Stab}(b) \lessdot_{\mathcal{S}} \mathrm{Stab}(a+b)$.

$$
\begin{array}{l}
\quad \mathrm{Stab}(a) \cap \mathrm{Stab}(b) \lessdot_{\mathcal{S}} \mathrm{Stab}(a+b) \\
\iff \qquad \langle \text{ Definition 4.2.5 for } \lessdot_{\mathcal{S}} \rangle \\
\quad \forall\big(x \mid x \in \mathrm{Stab}(a) \cap \mathrm{Stab}(b) : \exists(y \mid y \in \mathrm{Stab}(a+b) : x \leq_{\mathcal{S}} y)\big) \\
\iff \qquad \langle \text{ Trading for } \exists \rangle \\
\quad \forall\big(x \mid x \in \mathrm{Stab}(a) \cap \mathrm{Stab}(b) : \exists(y \mid y \in \mathrm{Stab}(a+b) \wedge x \leq_{\mathcal{K}} y)\big) \\
\impliedby \qquad \langle \exists\text{-Introduction} \rangle \\
\quad \forall\big(x \mid x \in \mathrm{Stab}(a) \cap \mathrm{Stab}(b) : x \in \mathrm{Stab}(a+b) \wedge x \leq_{\mathcal{K}} x\big) \\
\iff \qquad \langle \text{ Trading for } \forall \quad \& \quad \text{Reflexivity of } \leq_{\mathcal{K}} \rangle \\
\quad \forall\big(x \mid x \in \mathrm{Stab}(a) \cap \mathrm{Stab}(b) \implies x \in \mathrm{Stab}(a+b)\big) \\
\iff \qquad \langle \text{ Definition 4.2.4(3) } \rangle \\
\quad \forall\big(x \mid x \circ a = a \wedge x \circ b = b \implies x \circ (a+b) = a+b\big) \\
\iff \qquad \langle \text{ Definition 3.1.8(1) for } \big({}_{\mathcal{S}}K, +\big) \rangle \\
\quad \forall\big(x \mid \mathsf{true}\big) \\
\iff \qquad \langle \forall\text{-True Body} \rangle \\
\quad \mathsf{true}
\end{array}
$$

A.7 Detailed Proof of Proposition 4.2.6

Let $(\mathcal{S}, \mathcal{K})$ be a C^{2}KA and let $a, b \in K$.

(1) 0 is a fixed point w.r.t. $\circ$.

$$
\begin{array}{ll}
& 0 \text{ is a fixed point} \\
\iff & \langle \text{ Definition 4.2.4(4) } \rangle \\
& \forall(s \mid s \in S : s \circ 0 = 0) \\
\iff & \langle \ (_{\mathcal{S}}K, +) \text{ is zero-preserving } (\mathfrak{d} \circ a = 0) \ \rangle \\
& \forall(s \mid s \in S : s \circ (\mathfrak{d} \circ a) = 0) \\
\iff & \langle \text{ Definition 3.1.8(3) for } (_{\mathcal{S}}K, +) \ \rangle \\
& \forall(s \mid s \in S : (s \odot \mathfrak{d}) \circ a = 0) \\
\iff & \langle \ \mathfrak{d} \text{ is multiplicatively absorbing in } \mathcal{S} \ (s \odot \mathfrak{d} = \mathfrak{d}) \ \rangle \\
& \forall(s \mid s \in S : \mathfrak{d} \circ a = 0) \\
\iff & \langle \ (_{\mathcal{S}}K, +) \text{ is zero-preserving } (\mathfrak{d} \circ a = 0) \ \rangle \\
& \forall(s \mid s \in S : \mathsf{true}) \\
\iff & \langle \ \forall\text{-True Body} \ \rangle \\
& \mathsf{true}
\end{array}
$$

(2) a and b are fixed points $\implies$ $a + b$ is a fixed point

$$
\begin{array}{ll}
& a + b \text{ is a fixed point} \\
\iff & \langle \text{ Definition 4.2.4(4) } \rangle
\end{array}
$$

$\forall(s \mid s \in S\backslash\{\mathfrak{d}\} : s \circ (a + b) = a + b)$

$\iff$ $\langle$ Definition 3.1.8(1) for $\left({}_{\mathcal{S}}K, +\right)$ $\rangle$

$\forall(s \mid s \in S\backslash\{\mathfrak{d}\} : s \circ a + s \circ b = a + b)$

$\impliedby$ $\langle$ Hypothesis: a and b are fixed points $\rangle$

$\forall(s \mid s \in S\backslash\{\mathfrak{d}\} : a + b = a + b)$

$\iff$ $\langle$ Reflexivity of $=$ $\rangle$

$\forall(s \mid s \in S\backslash\{\mathfrak{d}\} : \mathsf{true})$

$\iff$ $\langle$ $\forall$-True Body $\rangle$

true

(3) a and b are fixed points $\implies$ $a\,;b$ is a fixed point

$a\,;b$ is a fixed point

$\iff$ $\langle$ Definition 4.2.4(4) $\rangle$

$\forall(s \mid s \in S\backslash\{\mathfrak{d}\} : s \circ (a\,;b) = a\,;b)$

$\iff$ $\langle$ Definition 4.2.3(1) $\rangle$

$\forall\big(s \mid s \in S\backslash\{\mathfrak{d}\} : (s \circ a)\,;\big(\lambda(s,a) \circ b\big) = a\,;b\big)$

$\impliedby$ $\langle$ $\lambda(s,a) \in S$ & Hypothesis: a and b are fixed points $\rangle$

$\forall(s \mid s \in S\backslash\{\mathfrak{d}\} : a\,;b = a\,;b)$

$\iff$ $\langle$ Reflexivity of $=$ $\rangle$

$\forall(s \mid s \in S\backslash\{\mathfrak{d}\} : \mathsf{true})$

$\iff$ $\langle$ $\forall$-True Body $\rangle$

true

(4) a is a fixed point $\implies$ $a^{\circledcirc}$ is a fixed point

$a^{\circledcirc}$ is a fixed point

$\iff$ $\langle$ Definition 4.2.4(4) $\rangle$

$\forall(s \mid s \in S\backslash\{\mathfrak{d}\} : s \circ a^{\circledcirc} = a^{\circledcirc})$

$\iff$ $\langle$ Corollary 4.2.2(5) $\rangle$

$\forall(s \mid s \in S\backslash\{\mathfrak{d}\} : +(n \mid n \geq 0 : s \circ a^n) = a^{\circledcirc})$

$\impliedby$ $\langle$ Hypothesis: a is a fixed point & Definition 4.2.3(4) & Proposition 4.2.6(3) $\rangle$

$\forall(s \mid s \in S\backslash\{\mathfrak{d}\} : +(n \mid n \geq 0 : a^n) = a^{\circledcirc})$

$\iff$ $\langle$ Definition of $a^{\circledcirc}$: Equation (A.1) $\rangle$

$\forall(s \mid s \in S\backslash\{\mathfrak{d}\} : a^{\circledcirc} = a^{\circledcirc})$

$\iff$ $\langle$ Reflexivity of $=$ $\rangle$

$\forall(s \mid s \in S\backslash\{\mathfrak{d}\} : \mathsf{true})$

$\iff$ $\langle$ $\forall$-True Body $\rangle$

true

A.8 Detailed Proof of Proposition 4.2.7

Let $a, b, c \in K$ be agent behaviours.

(1) a is a fixed point $\implies$ $\forall(b \mid b \in K \wedge b \neq 0 \wedge b \neq a : \neg(a \lhd b))$

$\forall(b \mid b \in K \wedge b \neq 0 \wedge b \neq a : \neg(a \lhd b))$

$\iff$ $\langle$ Definition 4.2.6 & Set Membership Axiom $\rangle$

$\forall(b \mid b \in K \ \wedge\ b \neq 0 \ \wedge\ b \neq a \ :\ b \notin \mathrm{Orb}(a))$

$\iff$ $\langle$ Trading for $\forall$ $\rangle$

$\forall(b \mid b \in K \ :\ (b \neq 0 \ \wedge\ b \neq a) \implies b \notin \mathrm{Orb}(a))$

$\impliedby$ $\langle$ Hypothesis: a is a fixed point $\implies \mathrm{Orb}(a) = \{0, a\}$ $\rangle$

$\forall(b \mid b \in K \ :\ (b \neq 0 \ \wedge\ b \neq a) \implies b \notin \{0, a\})$

$\iff$ $\langle$ $b \notin \{0, a\} \iff (b \neq 0 \ \wedge\ b \neq a)$ $\rangle$

$\forall(b \mid b \in K \ :\ (b \neq 0 \ \wedge\ b \neq a) \implies (b \neq 0 \ \wedge\ b \neq a))$

$\iff$ $\langle$ Reflexivity of $\implies$ $\rangle$

$\forall(b \mid b \in K \ :\ \mathsf{true})$

$\iff$ $\langle$ $\forall$-True Body $\rangle$

true

(2) $a \sim_{\mathcal{K}} b \implies a \lhd b \ \wedge\ b \lhd a$

$a \lhd b \ \wedge\ b \lhd a$

$\iff$ $\langle$ Definition 4.2.6 & Set Membership Axiom $\rangle$

$b \in \mathrm{Orb}(a) \ \wedge\ a \in \mathrm{Orb}(b)$

$\impliedby$ $\langle$ Hypothesis: $a \sim_{\mathcal{K}} b \iff \mathrm{Orb}(a) = \mathrm{Orb}(b)$ $\rangle$

$b \in \mathrm{Orb}(b) \ \wedge\ a \in \mathrm{Orb}(a)$

$\iff$ $\langle$ Definition of $\mathrm{Orb}(b)$ & Definition of $\mathrm{Orb}(a)$ $\rangle$

$b \in \{s \circ b \mid s \in S\} \ \wedge\ a \in \{s \circ a \mid s \in S\}$

$\iff$ $\langle$ Set Membership Axiom $\rangle$

$$\exists(s \mid s \in S : s \circ b = b) \wedge \exists(s \mid s \in S : s \circ a = a)$$

$\Longleftarrow$ $\langle$ $\exists$-Introduction $\rangle$

$$\mathfrak{n} \circ b = b \wedge \mathfrak{n} \circ a = a$$

$\Longleftrightarrow$ $\langle$ $\left({}_{\mathcal{S}}K, +\right)$ is unitary $(\mathfrak{n} \circ a = a)$ & Idempotence of $\wedge$ $\rangle$

$$\mathsf{true}$$

(3) $a \sim_{\mathcal{K}} b \implies (a \lhd c \iff b \lhd c)$

$$a \sim_{\mathcal{K}} b \implies (a \lhd c \iff b \lhd c)$$

$\Longleftrightarrow$ $\langle$ Mutual Implication $\rangle$

$$a \sim_{\mathcal{K}} b \implies \big[(a \lhd c \implies b \lhd c) \wedge (b \lhd c \implies a \lhd c)\big]$$

$\Longleftrightarrow$ $\langle$ Distributivity of $\implies$ over $\wedge$ $\rangle$

$$\big[a \sim_{\mathcal{K}} b \implies (a \lhd c \implies b \lhd c)\big] \wedge \big[a \sim_{\mathcal{K}} b \implies (b \lhd c \implies a \lhd c)\big]$$

$\Longleftrightarrow$ $\langle$ Shunting $\rangle$

$$\big[a \sim_{\mathcal{K}} b \wedge a \lhd c \implies b \lhd c\big] \wedge \big[a \sim_{\mathcal{K}} b \wedge b \lhd c \implies a \lhd c\big]$$

$\Longleftrightarrow$ $\langle$ Definition of $\sim_{\mathcal{K}}$ & Definition 4.2.6 $\rangle$

$$\big[\mathrm{Orb}(a) = \mathrm{Orb}(b) \wedge c \in \mathrm{Orb}(a) \implies c \in \mathrm{Orb}(b)\big] \wedge$$
$$\big[\mathrm{Orb}(a) = \mathrm{Orb}(b) \wedge c \in \mathrm{Orb}(b) \implies c \in \mathrm{Orb}(a)\big]$$

$\Longleftrightarrow$ $\langle$ $x \in A \iff \{x\} \subseteq A$ $\rangle$

$$\big[\mathrm{Orb}(a) = \mathrm{Orb}(b) \wedge \{c\} \subseteq \mathrm{Orb}(a) \implies \{c\} \subseteq \mathrm{Orb}(b)\big] \wedge$$
$$\big[\mathrm{Orb}(a) = \mathrm{Orb}(b) \wedge \{c\} \subseteq \mathrm{Orb}(b) \implies \{c\} \subseteq \mathrm{Orb}(a)\big]$$

$\Longleftrightarrow$ $\langle$ Transitivity of $\subseteq$ & Idempotence of $\wedge$ $\rangle$

$$\mathsf{true}$$

A.9 Detailed Proof of Proposition 5.2.1

Assume that a system formed by a set $\mathcal{C}$ of communicating agents is a stimuli-disconnected system and let agent $\mathsf{C} \in \mathcal{C}$ be universally influential. Also, assume that there exists a partition of $\mathcal{C}$, X_1 and X_2, such that $\mathsf{C} \in X_2$. The proof is by contradiction.

$\mathcal{C}$ is stimuli-disconnected $\wedge$ C is universally influential

$\iff$ $\langle$ Definition of Stimuli-Disconnected (see Definition 5.2.4) $\rangle$

$\forall(\mathsf{A}, \mathsf{B} \mid \mathsf{A} \in X_1 \wedge \mathsf{B} \in X_2 : \neg(\mathsf{A} \rightarrow_{\mathcal{S}}^{+} \mathsf{B}) \wedge \neg(\mathsf{B} \rightarrow_{\mathcal{S}}^{+} \mathsf{A})) \wedge \mathsf{C}$ is universally influential

$\implies$ $\langle$ Instantiation: $\mathsf{B} = \mathsf{C}$ $\rangle$

$\forall(\mathsf{A} \mid \mathsf{A} \in X_1 : \neg(\mathsf{A} \rightarrow_{\mathcal{S}}^{+} \mathsf{C}) \wedge \neg(\mathsf{C} \rightarrow_{\mathcal{S}}^{+} \mathsf{A})) \wedge \mathsf{C}$ is universally influential

$\implies$ $\langle$ Definition 5.2.6 $\rangle$

$\forall(\mathsf{A} \mid \mathsf{A} \in X_1 : \neg(\mathsf{A} \rightarrow_{\mathcal{S}}^{+} \mathsf{C}) \wedge \mathsf{false})$

$\iff$ $\langle$ Zero of $\wedge$ & $\forall$-False Body $\rangle$

false

A.10 Detailed Proof of Proposition 5.2.3

Let $\mathsf{A} \mapsto \langle a \rangle$, $\mathsf{B} \mapsto \langle b \rangle$, and $\mathsf{C} \mapsto \langle c \rangle$ be agents in $\mathcal{C}$.

(1) If $\mathsf{B} \rightarrow_{\mathcal{S}} \mathsf{C}$ then $(\mathsf{A} + \mathsf{B}) \rightarrow_{\mathcal{S}} \mathsf{C}$.

$(\mathsf{A} + \mathsf{B}) \rightarrow_{\mathcal{S}} \mathsf{C}$

$\iff$ $\langle$ Definition 5.2.1 $\rangle$

$\exists\big(s, t \mid s, t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s, a + b) : t \circ c \neq c\big)$

$\iff$ $\langle$ Distributivity of λ over $+$ $\rangle$

$$\exists\big(s,t \mid s,t \in S_b \ \wedge\ t \leq_{\mathcal{S}} \lambda(s,a) \oplus \lambda(s,b) \ :\ t \circ c \neq c\big)$$

$\iff$ $\langle$ Definition of $\leq_{\mathcal{S}}$ $\rangle$

$$\exists\big(s,t \mid s,t \in S_b \ \wedge\ t \oplus \lambda(s,a) \oplus \lambda(s,b) = \lambda(s,a) \oplus \lambda(s,b) \ :\ t \circ c \neq c\big)$$

$\iff$ $\langle$ Idempotence of $\oplus$ $\rangle$

$$\exists\big(s,t \mid s,t \in S_b \ \wedge\ \lambda(s,a) \oplus t \oplus \lambda(s,a) \oplus \lambda(s,b) = \lambda(s,a) \oplus \lambda(s,b) \ :\ t \circ c \neq c\big)$$

$\iff$ $\langle$ Definition of $\leq_{\mathcal{S}}$ $\rangle$

$$\exists\big(s,t \mid s,t \in S_b \ \wedge\ \lambda(s,a) \oplus t \leq_{\mathcal{S}} \lambda(s,a) \oplus \lambda(s,b) \ :\ t \circ c \neq c\big)$$

$\impliedby$ $\langle$ $\oplus$ is left-isotone with respect to $\leq_{\mathcal{S}}$ $\rangle$

$$\exists\big(s,t \mid s,t \in S_b \ \wedge\ t \leq_{\mathcal{S}} \lambda(s,b) \ :\ t \circ c \neq c\big)$$

$\impliedby$ $\langle$ Hypothesis: $\mathsf{B} \to_{\mathcal{S}} \mathsf{C}$ $\rangle$

true

(2) If $\mathsf{A} \to_{\mathcal{S}} \mathsf{B}$ then $\mathsf{A} \to_{\mathcal{S}} (\mathsf{B} + \mathsf{C})$ if $\forall\big(s,t \mid s,t \in S_b \ \wedge\ t \leq_{\mathcal{S}} \lambda(s,a) \ :\ \neg(t \circ b \leq_{\mathcal{K}} b + c \ \wedge\ t \circ c \leq_{\mathcal{K}} b + c)\big)$.

$$\mathsf{A} \to_{\mathcal{S}} \mathsf{B} \implies \mathsf{A} \to_{\mathcal{S}} (\mathsf{B} + \mathsf{C})$$

$\iff$ $\langle$ Definition 5.2.1 $\rangle$

$$\exists\big(s,t \mid s,t \in S_b \ \wedge\ t \leq_{\mathcal{S}} \lambda(s,a) \ :\ t \circ b \neq b\big) \implies$$
$$\exists\big(s,t \mid s,t \in S_b \ \wedge\ t \leq_{\mathcal{S}} \lambda(s,a) \ :\ t \circ (b+c) \neq (b+c)\big)$$

$\impliedby$ $\langle$ Monotonic $\exists$-Body $\rangle$

$$\forall\big(s,t \mid s,t \in S_b \ \wedge\ t \leq_{\mathcal{S}} \lambda(s,a) \ :\ t \circ b \neq b \implies t \circ (b+c) \neq (b+c)\big)$$

$\impliedby$ $\langle$ Anti-monotonic $\neg$ $\rangle$

$$\forall\big(s,t \mid s,t \in S_b \ \wedge\ t \leq_{\mathcal{S}} \lambda(s,a) \ :\ t \circ (b+c) = (b+c) \implies t \circ b = b\big)$$

$\iff$ $\langle$ Definition 3.1.8(1) for $\big({}_{\mathcal{S}}K, +\big)$ $\rangle$

$$\forall\big(s,t \mid s,t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s,a) : (t \circ b + t \circ c) = (b + c) \implies t \circ b = b\big)$$

$\iff$ $\langle$ Idempotence of $+$ $\rangle$

$$\forall\big(s,t \mid s,t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s,a) : (t \circ b + t \circ c) = (b + c + b + c) \implies t \circ b = b\big)$$

$\impliedby$ $\langle$ Substitution of $=$ by $=$ $\rangle$

$$\forall\big(s,t \mid s,t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s,a) : \big(t \circ b = (b + c) \wedge t \circ c = (b + c) \wedge (t \circ b + t \circ c) = (t \circ b + t \circ c)\big) \implies t \circ b = b\big)$$

$\iff$ $\langle$ Reflexivity of $=$ & Identity of $\wedge$ & Definition of $\implies$ $\rangle$

$$\forall\big(s,t \mid s,t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s,a) : \neg\big(t \circ b = (b + c) \wedge t \circ c = (b + c)\big) \vee t \circ b = b\big)$$

$\iff$ $\langle$ De Morgan $\rangle$

$$\forall\big(s,t \mid s,t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s,a) : t \circ b \neq (b + c) \vee t \circ c \neq (b + c) \vee t \circ b = b\big)$$

$\impliedby$ $\langle$ Hypothesis: $\forall\big(s,t \mid s,t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s,a) : \neg(t \circ b \leq_{\mathcal{K}} b + c \wedge t \circ c \leq_{\mathcal{K}} b + c)\big)$ & Weakening: $P \implies P \vee Q$ $\rangle$

true

A.11 Detailed Proof of Proposition 5.4.1

Let $\mathsf{A} \mapsto \langle a \rangle$, $\mathsf{B} \mapsto \langle b \rangle$, and $\mathsf{C} \mapsto \langle c \rangle$ be agents in $\mathcal{C}$.

(1) If $\mathsf{A} \rightsquigarrow \mathsf{B}$, then $\mathsf{A} \rightsquigarrow (\mathsf{C}\,;\mathsf{B})$ if $\forall\big(s,t \mid s,t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s,a) : t \circ c \neq c \vee \lambda(t,c) \circ b \neq b\big) \vee a\,\mathrm{R}\,(c\,;b)$.

$$\mathsf{A} \rightsquigarrow \mathsf{B} \implies \mathsf{A} \rightsquigarrow (\mathsf{C}\,;\mathsf{B})$$

$\iff$ $\langle$ Definition 5.2.9 $\rangle$

$$\big(\mathsf{A} \rightarrow_{\mathcal{S}} \mathsf{B} \vee \mathsf{A} \rightarrow_{\mathcal{E}} \mathsf{B}\big) \implies \big(\mathsf{A} \rightarrow_{\mathcal{S}} (\mathsf{C}\,;\mathsf{B}) \vee \mathsf{A} \rightarrow_{\mathcal{E}} (\mathsf{C}\,;\mathsf{B})\big)$$

$\iff$ $\langle$ Definition 5.2.1 & Definition 5.2.7 $\rangle$

$$
\begin{array}{l}
\big[\, \exists\big(s,t \mid s,t \in S_b \;\wedge\; t \leq_{\mathcal{S}} \lambda(s,a) \;:\; t \circ b \neq b\big) \;\vee\; a \,\mathrm{R}\, b\big] \;\Longrightarrow \\
\big[\, \exists\big(s,t \mid s,t \in S_b \;\wedge\; t \leq_{\mathcal{S}} \lambda(s,a) \;:\; t \circ (c\,;b) \neq (c\,;b)\big) \;\vee\; a \,\mathrm{R}\, (c\,;b)\big] \\
\Longleftarrow \qquad \langle\ \text{Distributivity of } \vee \text{ over } \exists\ \rangle \\
\exists\big(s,t \mid s,t \in S_b \;\wedge\; t \leq_{\mathcal{S}} \lambda(s,a) \;:\; t \circ b \neq b \;\vee\; a \,\mathrm{R}\, b\big) \;\Longrightarrow \\
\exists\big(s,t \mid s,t \in S_b \;\wedge\; t \leq_{\mathcal{S}} \lambda(s,a) \;:\; t \circ (c\,;b) \neq (c\,;b) \;\vee\; a \,\mathrm{R}\, (c\,;b)\big) \\
\Longleftarrow \qquad \langle\ \text{Monotonic } \exists\text{-Body}\ \rangle \\
\forall\big(s,t \mid s,t \in S_b \;\wedge\; t \leq_{\mathcal{S}} \lambda(s,a) \;:\; \big(t \circ b \neq b \;\vee\; a \,\mathrm{R}\, b\big) \;\Longrightarrow \\
\big(t \circ (c\,;b) \neq (c\,;b) \;\vee\; a \,\mathrm{R}\, (c\,;b)\big)\big) \\
\Longleftarrow \qquad \langle\ \text{Anti-monotonic } \neg\ \rangle \\
\forall\big(s,t \mid s,t \in S_b \;\wedge\; t \leq_{\mathcal{S}} \lambda(s,a) \;:\; \big(t \circ (c\,;b) = (c\,;b) \;\wedge\; \neg(a \,\mathrm{R}\, (c\,;b))\big) \;\Longrightarrow \\
\big(t \circ b = b \;\wedge\; \neg(a \,\mathrm{R}\, b)\big)\big) \\
\Longleftrightarrow \qquad \langle\ \text{Definition 4.2.3(1)}\ \rangle \\
\forall\big(s,t \mid s,t \in S_b \;\wedge\; t \leq_{\mathcal{S}} \lambda(s,a) \;:\; \big((t \circ c)\,;\big(\lambda(t,c) \circ b\big) = (c\,;b) \;\wedge\; \\
\neg(a \,\mathrm{R}\, (c\,;b))\big) \;\Longrightarrow\; \big(t \circ b = b \;\wedge\; \neg(a \,\mathrm{R}\, b)\big)\big) \\
\Longleftarrow \qquad \langle\ \text{Substitution of } = \text{ by } =\ \rangle \\
\forall\big(s,t \mid s,t \in S_b \;\wedge\; t \leq_{\mathcal{S}} \lambda(s,a) \;:\; \big(t \circ c = c \;\wedge\; \lambda(t,c) \circ b = b \;\wedge\; (t \circ c)\,;\big(\lambda(t,c) \circ b\big) = \\
(t \circ c)\,;\big(\lambda(t,c) \circ b\big) \;\wedge\; \neg(a \,\mathrm{R}\, (c\,;b))\big) \;\Longrightarrow\; \big(t \circ b = b \;\wedge\; \neg(a \,\mathrm{R}\, b)\big)\big) \\
\Longleftrightarrow \qquad \langle\ \text{Reflexivity of } = \quad \& \quad \text{Identity of } \wedge \quad \& \quad \text{Definition of } \Longrightarrow\ \rangle \\
\forall\big(s,t \mid s,t \in S_b \;\wedge\; t \leq_{\mathcal{S}} \lambda(s,a) \;:\; \neg\big(t \circ c = c \;\wedge\; \lambda(t,c) \circ b = b \;\wedge\; \neg(a \,\mathrm{R}\, (c\,;b))\big) \;\vee \\
\big(t \circ b = b \;\wedge\; \neg(a \,\mathrm{R}\, b)\big)\big) \\
\Longleftrightarrow \qquad \langle\ \text{De Morgan}\ \rangle \\
\forall\big(s,t \mid s,t \in S_b \;\wedge\; t \leq_{\mathcal{S}} \lambda(s,a) \;:\; t \circ c \neq c \;\vee\; \lambda(t,c) \circ b \neq b \;\vee\; a \,\mathrm{R}\, (c\,;b) \;\vee \\
\big(t \circ b = b \;\wedge\; \neg(a \,\mathrm{R}\, b)\big)\big) \\
\Longleftarrow \qquad \langle\ \text{Hypothesis: } \big(\forall\big(s,t \mid s,t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s,a) : t \circ c \neq c \vee \lambda(t,c) \circ b \neq b\big) \;\vee \\
\qquad\qquad a \,\mathrm{R}\, (c\,;b)\big) \quad \& \quad \text{Weakening: } P \;\Longrightarrow\; P \;\vee\; Q\ \rangle \\
\quad \mathsf{true}
\end{array}
$$

(2) If $\mathsf{A} \rightsquigarrow \mathsf{B}$, then $(\mathsf{A}\,;\mathsf{C}) \rightsquigarrow \mathsf{B}$ if $\forall\big(s,t \mid s,t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s,a) : \lambda(t,c) = t\big) \vee (a\,;c)\,\mathrm{R}\,b$.

$$
\begin{aligned}
& (\mathsf{A}\,;\mathsf{C}) \rightsquigarrow \mathsf{B} \\
\iff & \quad \langle \text{ Definition 5.2.9 } \rangle \\
& (\mathsf{A}\,;\mathsf{C}) \rightarrow_{\mathcal{S}} \mathsf{B} \vee (\mathsf{A}\,;\mathsf{C}) \rightarrow_{\mathcal{E}} \mathsf{B} \\
\iff & \quad \langle \text{ Definition 5.2.1 } \& \text{ Definition 5.2.7 } \rangle \\
& \exists\big(s,t \mid s,t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s,(a\,;c)) : t \circ b \neq b\big) \vee (a\,;c)\,\mathrm{R}\,b \\
\iff & \quad \langle \text{ Definition 3.1.8(3) for } \big(S_{\mathcal{K}}, \oplus\big) \rangle \\
& \exists\big(s,t \mid s,t \in S_b \wedge t \leq_{\mathcal{S}} \lambda\big(\lambda(s,a),c\big) : t \circ b \neq b\big) \vee (a\,;c)\,\mathrm{R}\,b \\
\impliedby & \quad \langle \text{ Hypothesis: } \big(\forall\big(s,t \mid s,t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s,a) : \lambda(t,c) = t\big) \vee \\
& \qquad (a\,;c)\,\mathrm{R}\,b\big) \quad \& \quad \big(\mathsf{A} \rightsquigarrow \mathsf{B} \implies \exists(t \mid t \in S_b : t \circ b \neq b)\big) \rangle \\
& \mathsf{true}
\end{aligned}
$$

A.12 Detailed Proof of Proposition 5.4.2

Let $\mathsf{A} \rightsquigarrow^{+} \mathsf{B}$ such that $\exists\big(\mathsf{C} \mid \mathsf{C} \in \mathcal{C} : \mathsf{A} \rightsquigarrow \mathsf{C} \wedge \mathsf{C} \rightsquigarrow \mathsf{B}\big)$.

(1) $\mathsf{C}' \mapsto \langle c\,;d \rangle$

$$
\begin{aligned}
& \mathsf{A} \rightsquigarrow \mathsf{C}' \wedge \mathsf{C}' \rightsquigarrow \mathsf{B} \\
\iff & \quad \langle \text{ Substitution: } \mathsf{C}' = (\mathsf{C}\,;\mathsf{D}) \text{ where } \mathsf{C} \mapsto \langle c \rangle \text{ and } \mathsf{D} \mapsto \langle d \rangle \rangle \\
& \mathsf{A} \rightsquigarrow (\mathsf{C}\,;\mathsf{D}) \wedge (\mathsf{C}\,;\mathsf{D}) \rightsquigarrow \mathsf{B} \\
\impliedby & \quad \langle \text{ Hypothesis: } \mathsf{A} \rightsquigarrow \mathsf{C} \implies \mathsf{A} \rightsquigarrow (\mathsf{C}\,;\mathsf{D}) \quad \& \quad \text{Identity of } \wedge \rangle
\end{aligned}
$$

$(\mathsf{C}\,;\mathsf{D}) \rightsquigarrow \mathsf{B}$

$\Longleftarrow$ $\langle$ Hypothesis: $\big[\mathsf{C} \rightsquigarrow \mathsf{B} \;\wedge\; \big(\,\forall\big(s,t \mid s,t \in S_b \wedge t \leq_{\mathcal{S}} \lambda(s,c) : \lambda(t,d) = t\,\big) \;\vee\; (c\,;d)\,\mathrm{R}\,b\big)\big]$ & Proposition 5.4.1(2) $\rangle$

true

(2) $\mathsf{C}' \mapsto \langle c + d \rangle$

$\mathsf{A} \rightsquigarrow \mathsf{C}' \;\wedge\; \mathsf{C}' \rightsquigarrow \mathsf{B}$

$\Longleftrightarrow$ $\langle$ Substitution: $\mathsf{C}' = (\mathsf{C} + \mathsf{D})$ where $\mathsf{C} \mapsto \langle c \rangle$ and $\mathsf{D} \mapsto \langle d \rangle$ $\rangle$

$\mathsf{A} \rightsquigarrow (\mathsf{C} + \mathsf{D}) \;\wedge\; (\mathsf{C} + \mathsf{D}) \rightsquigarrow \mathsf{B}$

$\Longleftrightarrow$ $\langle$ Definition 5.2.9 $\rangle$

$\big[\mathsf{A} \rightarrow_{\mathcal{S}} (\mathsf{C} + \mathsf{D}) \;\vee\; \mathsf{A} \rightarrow_{\mathcal{E}} (\mathsf{C} + \mathsf{D})\big] \;\wedge\; \big[(\mathsf{C} + \mathsf{D}) \rightarrow_{\mathcal{S}} \mathsf{B} \;\vee\; (\mathsf{C} + \mathsf{D}) \rightarrow_{\mathcal{E}} \mathsf{B}\big]$

$\Longleftarrow$ $\langle$ Hypothesis: $\big[\mathsf{A} \rightsquigarrow \mathsf{C} \;\wedge\; \forall\big(s,t \mid s,t \in S_b \;\wedge\; t \leq_{\mathcal{S}} \lambda(s,a) :$ $\neg(t \circ c \leq_{\mathcal{K}} c + d \;\wedge\; t \circ d \leq_{\mathcal{K}} c + d)\,\big)\big]$ & Proposition 5.2.3 & Proposition 5.2.4 $\rangle$

true

(3) $\mathsf{C}' \mapsto \langle c^{\circlearrowleft} \rangle$

$\mathsf{A} \rightsquigarrow \mathsf{C}' \;\wedge\; \mathsf{C}' \rightsquigarrow \mathsf{B}$

$\Longleftrightarrow$ $\langle$ Definition 5.2.9 $\rangle$

$\big(\mathsf{A} \rightarrow_{\mathcal{S}} \mathsf{C}' \;\vee\; \mathsf{A} \rightarrow_{\mathcal{E}} \mathsf{C}'\big) \;\wedge\; \big(\mathsf{C}' \rightarrow_{\mathcal{S}} \mathsf{B} \;\vee\; \mathsf{C}' \rightarrow_{\mathcal{E}} \mathsf{B}\big)$

$\Longleftrightarrow$ $\langle$ Definition 5.2.1 & Definition 5.2.7 $\rangle$

$$\big(\,\exists\big(s,t \mid s,t \in S_b \;\wedge\; t \leq_{\mathcal{S}} \lambda(s,a) \;:\; t \circ c^{\textcircled{:}} \neq c^{\textcircled{:}}\,\big) \;\vee\; a \,\mathrm{R}\, c^{\textcircled{:}}\big) \;\wedge$$

$$\big(\,\exists\big(s,t \mid s,t \in S_b \;\wedge\; t \leq_{\mathcal{S}} \lambda(s,c^{\textcircled{:}}) \;:\; t \circ b \neq b\,\big) \;\vee\; c^{\textcircled{:}} \,\mathrm{R}\, b\big)$$

$\Longleftarrow$ $\langle$ Definition of $^{\textcircled{:}}$ & Proposition 5.4.2(2) $\rangle$

true

(4) $\mathsf{C}' \mapsto \langle 0 \rangle$

$\mathsf{A} \rightsquigarrow \mathsf{C}' \;\wedge\; \mathsf{C}' \rightsquigarrow \mathsf{B}$

$\Longleftrightarrow$ $\langle$ Definition 5.2.9 $\rangle$

$\big(\mathsf{A} \to_{\mathcal{S}} \mathsf{C}' \;\vee\; \mathsf{A} \to_{\mathcal{E}} \mathsf{C}'\big) \;\wedge\; \big(\mathsf{C}' \to_{\mathcal{S}} \mathsf{B} \;\vee\; \mathsf{C}' \to_{\mathcal{E}} \mathsf{B}\big)$

$\Longleftrightarrow$ $\langle$ 0 is a fixed point behaviour & Proposition 5.2.2 & $\neg(a \,\mathrm{R}\, 0)$ $\rangle$

$\big(\mathsf{false} \;\vee\; \mathsf{false}\big) \;\wedge\; \big(\mathsf{C}' \to_{\mathcal{S}} \mathsf{B} \;\vee\; \mathsf{C}' \to_{\mathcal{E}} \mathsf{B}\big)$

$\Longleftrightarrow$ $\langle$ Idempotence of $\vee$ & Zero of $\wedge$ $\rangle$

false

The proof is similar when $\mathsf{C}' \mapsto \langle 1 \rangle$.

(5) $\mathsf{C}' \mapsto \langle c' \rangle$ such that $c' \in \mathrm{Orb}_{\mathrm{S}}(c)$

$\mathsf{A} \rightsquigarrow \mathsf{C}' \;\wedge\; \mathsf{C}' \rightsquigarrow \mathsf{B}$

$\Longleftrightarrow$ $\langle$ Definition 5.2.9 $\rangle$

$\big(\mathsf{A} \to_{\mathcal{S}} \mathsf{C}' \;\vee\; \mathsf{A} \to_{\mathcal{E}} \mathsf{C}'\big) \;\wedge\; \big(\mathsf{C}' \to_{\mathcal{S}} \mathsf{B} \;\vee\; \mathsf{C}' \to_{\mathcal{E}} \mathsf{B}\big)$

$\Longleftarrow$ $\langle$ Hypothesis: $\mathsf{A} \rightsquigarrow \mathsf{C} \;\wedge\; \mathsf{C} \rightsquigarrow \mathsf{B}$ & Hypothesis: $c' \in \mathrm{Orb}_{\mathrm{S}}(c) \implies$

$\exists(s,t \mid s,t \in S \;:\; s \circ c = c' \;\wedge\; t \circ c' = c) \implies \mathsf{C} \to_{\mathcal{S}}^{+} \mathsf{C}' \;\wedge\; \mathsf{C}' \to_{\mathcal{S}}^{+} \mathsf{C}$ $\rangle$

true

(6) $\mathsf{C}' \mapsto \langle c' \rangle$ such that c' is a fixed point behaviour

$$
\begin{array}{ll}
& \mathsf{A} \rightsquigarrow \mathsf{C}' \ \wedge \ \mathsf{C}' \rightsquigarrow \mathsf{B} \\
\iff & \quad \langle \text{ Definition 5.2.9 } \rangle \\
& \big(\mathsf{A} \rightarrow_{\mathcal{S}} \mathsf{C}' \ \vee \ \mathsf{A} \rightarrow_{\mathcal{E}} \mathsf{C}'\big) \ \wedge \ \big(\mathsf{C}' \rightarrow_{\mathcal{S}} \mathsf{B} \ \vee \ \mathsf{C}' \rightarrow_{\mathcal{E}} \mathsf{B}\big) \\
\iff & \quad \langle \text{ Definition 5.2.7 } \rangle \\
& \big(\mathsf{A} \rightarrow_{\mathcal{S}} \mathsf{C}' \ \vee \ a \,\mathrm{R}\, c'\big) \ \wedge \ \big(\mathsf{C}' \rightarrow_{\mathcal{S}} \mathsf{B} \ \vee \ c' \,\mathrm{R}\, b\big) \\
\impliedby & \quad \langle \text{ Hypothesis: } c' \text{ is a fixed point behaviour} \quad \& \quad \text{Proposition 5.2.2 } \rangle \\
& \big(\mathsf{false} \ \vee \ a \,\mathrm{R}\, c'\big) \ \wedge \ \big(\mathsf{C}' \rightarrow_{\mathcal{S}} \mathsf{B} \ \vee \ c' \,\mathrm{R}\, b\big) \\
\iff & \quad \langle \text{ Identity of } \vee \ \rangle \\
& a \,\mathrm{R}\, c' \ \wedge \ \big(\mathsf{C}' \rightarrow_{\mathcal{S}} \mathsf{B} \ \vee \ c' \,\mathrm{R}\, b\big) \\
\impliedby & \quad \langle \text{ Hypothesis: } a \,\mathrm{R}\, c' \ \wedge \ c' \,\mathrm{R}\, b \ \rangle \\
& \mathsf{true} \ \wedge \ \big(\mathsf{C}' \rightarrow_{\mathcal{S}} \mathsf{B} \ \vee \ \mathsf{true}\big) \\
\iff & \quad \langle \text{ Zero of } \vee \quad \& \quad \text{Idempotence of } \wedge \ \rangle \\
& \mathsf{true}
\end{array}
$$

Appendix B

Axioms of $\mathrm{C}^2\mathrm{KA}$

Let $\big(\mathcal{S}, \mathcal{K}\big)$ be a C^2KA consisting of a stimulus structure $\mathcal{S} = \big(S, \oplus, \odot, \mathfrak{d}, \mathfrak{n}\big)$ and a concurrent Kleene algebra $\mathcal{K} = \big(K, +, *, ;, \circledast, \overset{\cdot}{\circledast}, 0, 1\big)$. Also, let $\big({}_{\mathcal{S}}K, +\big)$ be a unitary and zero-preserving *left $\mathcal{S}$-semimodule* with mapping $\circ : S \times K \to K$ and $\big(S_{\mathcal{K}}, \oplus\big)$ be a unitary and zero-preserving *right $\mathcal{K}$-semimodule* with mapping $\lambda : S \times K \to S$.

B.1 Stimulus Structure $\mathcal{S}$ Axioms

For all $s, t, r \in S$:

(1) $s \oplus (t \oplus r) = (s \oplus t) \oplus r$ (associativity of $\oplus$)

(2) $s \oplus t = t \oplus s$ (commutativity of $\oplus$)

(3) $s \oplus \mathfrak{d} = s$ (identity of $\oplus$)

(4) $s \oplus s = s$ (idempotence of $\oplus$)

(5) $s \leq_{\mathcal{S}} t \iff s \oplus t = t$ (definition of $\leq_{\mathcal{S}}$)

(6) $s \odot (t \odot r) = (s \odot t) \odot r$ (associativity of $\odot$)

(7) $s \odot \mathfrak{n} = s$ (right identity of $\odot$)

(8) $\mathfrak{n} \odot s = s$ (left identity of $\odot$)

(9) $(s \oplus t) \odot r = s \odot r \oplus t \odot r$ (right distributivity of $\odot$ over $\oplus$)

(10) $s \odot (t \oplus r) = s \odot t \oplus s \odot r$ (left distributivity of $\odot$ over $\oplus$)

(11) $s \odot \mathfrak{d} = \mathfrak{d}$ (right annihilator of $\odot$)

(12) $\mathfrak{d} \odot s = \mathfrak{d}$ (left annihilator of $\odot$)

B.2 Concurrent Kleene Algebra $\mathcal{K}$ Axioms

For all $a, b, c, d \in K$:

(13) $a + (b + c) = (a + b) + c$ (associativity of $+$)

(14) $a + b = b + a$ (commutativity of $+$)

(15) $a + 0 = a$ (identity of $+$)

(16) $a + a = a$ (idempotence of $+$)

(17) $a \leq_{\mathcal{K}} b \iff a + b = b$ (definition of $\leq$)

(18) $a * (b * c) = (a * b) * c$ (associativity of $*$)

(19) $a * b = b * a$ (commutativity of $*$)

(20) $a * 1 = a$ (right identity of $*$)

(21) $a * (b + c) = a * b + a * c$ (left distributivity of $*$ over $+$)

(22) $a * 0 = 0$ (right annihilator of $*$)

(23) $a \,;(b \,;c) = (a \,;b) \,;c$ (associativity of $;$)

(24) $a \,;1 = a$ (right identity of $;$)

(25) $1 \,;\, a = a$ (left identity of $;$)

(26) $(a + b) \,;\, c = a \,;\, c + b \,;\, c$ (right distributivity of $;$ over $+$)

(27) $a \,;\, (b + c) = a \,;\, b + a \,;\, c$ (left distributivity of $;$ over $+$)

(28) $a \,;\, 0 = 0$ (right annihilator of $;$)

(29) $0 \,;\, a = 0$ (left annihilator of $;$)

(30) $(a * b) \,;\, (c * d) \leq_{\mathcal{K}} (a \,;\, c) * (b \,;\, d)$ (exchange axiom)

(31) $1 + a * a^{\circledast} = a^{\circledast}$ (right unfold rule of $^{\circledast}$)

(32) $1 + a^{\circledast} * a = a^{\circledast}$ (left unfold rule of $^{\circledast}$)

(33) $c + b * a \leq_{\mathcal{K}} b \implies c * a^{\circledast} \leq_{\mathcal{K}} b$ (right induction rule of $^{\circledast}$)

(34) $c + a * b \leq_{\mathcal{K}} b \implies a^{\circledast} * c \leq_{\mathcal{K}} b$ (left induction rule of $^{\circledast}$)

(35) $1 + a \,;\, a^{\circledcirc} = a^{\circledcirc}$ (right unfold rule of $^{\circledcirc}$)

(36) $1 + a^{\circledcirc} \,;\, a = a^{\circledcirc}$ (left unfold rule of $^{\circledcirc}$)

(37) $c + b \,;\, a \leq_{\mathcal{K}} b \implies c \,;\, a^{\circledcirc} \leq_{\mathcal{K}} b$ (right induction rule of $^{\circledcirc}$)

(38) $c + a \,;\, b \leq_{\mathcal{K}} b \implies a^{\circledcirc} \,;\, c \leq_{\mathcal{K}} b$ (left induction rule of $^{\circledcirc}$)

B.3 Left $\mathcal{S}$-semimodule $\left({}_{\mathcal{S}}K, +\right)$ Axioms

For all $a, b \in K$ and $s, t \in S$:

(39) $s \circ (a + b) = s \circ a + s \circ b$ (distributivity of $\circ$ over $+$)

(40) $(s \oplus t) \circ a = s \circ a + t \circ b$ (distributivity of $\circ$ over $\oplus$)

(41) $(s \odot t) \circ a = s \circ (t \circ a)$ (sequential application of $\circ$)

(42) $\mathfrak{n} \circ a = a$ (unitary $\big({}_{\mathcal{S}}K, +\big)$)

(43) $\mathfrak{d} \circ a = 0$ (zero-preserving $\big({}_{\mathcal{S}}K, +\big)$)

B.4 Right $\mathcal{K}$-semimodule $\big(S_{\mathcal{K}}, \oplus\big)$ Axioms

For all $a, b \in K$ and $s, t \in S$:

(44) $\lambda((s \oplus t), a) = \lambda(s, a) \oplus \lambda(t, b)$ (distributivity of λ over $\oplus$)

(45) $\lambda(s, (a + b)) = \lambda(s, a) \oplus \lambda(s, b)$ (distributivity of λ over $+$)

(46) $\lambda\big(s, (a\,;b)\big) = \lambda\big(\lambda(s, a), b\big)$ (sequential application of λ)

(47) $\lambda(s, 1) = s$ (unitary $\big(S_{\mathcal{K}}, \oplus\big)$)

(48) $\lambda(s, 0) = \mathfrak{d}$ (zero-preserving $\big(S_{\mathcal{K}}, \oplus\big)$)

B.5 Communicating Concurrent Kleene Algebra Axioms

For all $a, b, c \in K$ and $s, t \in S$:

(49) $s \circ (a\,;b) = (s \circ a)\,;\big(\lambda(s, a) \circ b\big)$ (cascading law)

(50) $a \leq_{\mathcal{K}} c \;\vee\; b = 1 \;\vee\; (s \circ a)\,;\big(\lambda(s, c) \circ b\big) = 0$ (cascading output law)

(51) $\lambda(s \odot t, a) = \lambda\big(s, (t \circ a)\big) \odot \lambda(t, a)$ (sequential output law)

(52) $s = \mathfrak{d} \;\vee\; s \circ 1 = 1$ (idle agent law)

(53) $a = 0 \;\vee\; \lambda(\mathfrak{n}, a) = \mathfrak{n}$ (neutral stimulus law)

Appendix C

Analysing Agent Behaviour Using the Prototype Tool

In what follows, a brief overview of the usage of the main functions for using the prototype tool for specifying and analysing the agent behaviours in the running example system of communicating agents described in Section 4.1 is given.

C.1 Specifying Systems of Communicating Agents

The first step towards specifying a system of communicating agents using the prototype tool is to provide the specification of the system to be analysed. This involves setting the set of basic external stimuli and the set of basic agent behaviours to define the stimulus structure and the CKA of the C^2KA to be used for the specification. This is done using the `setStimSet` and `setCKASet` functions, respectively. It also involves setting the set of defined constants used in the concrete behaviour specifications of the agents using the `setConstants` function. The set of defined constants is used in the potential for communication analysis component of the prototype tool to distinguish defined constants and names in the concrete behaviour specifications of agents from mutable program variables. For the specification of

the running example, the set of basic external stimuli, the set of basic agent behaviours, and the set of defined constants are defined using the following commands:

```
> let stimSet = setStimSet ["ips", "abor", "allo", "help", "noop", "D", "N"]
> let ckaSet = setCKASet ["ABOR", "ALLO", "HELP", "NOOP", "SENDABOR",
                          "SENDALLO", "SENDHELP", "SENDNOOP", "SINCE1",
                          "SINCE2", "SINCE3", "SINCE4", "COUNT1", "COUNT2",
                          "COUNT3", "COUNT4", "AVG1", "AVG2", "AVG3",
                          "AVG4", "RESET1", "RESET2", "RESET3", "RESET4",
                          "TICK", "DELTA", "LAST", "WAIT", "READ", "0", "1"]
> let constants = setConstants ["ABOR", "ALLO", "HELP", "NOOP", "true"]
```

Once these sets have been defined, a new system of communicating agents can be created using the `newSoCA` function. The following program fragment creates a new system of communicating agents named "Example" with the set of basic external stimuli represented by `stimSet`, the set of basic agent behaviours represented by `ckaSet`, and the set of defined constants represented by `constants`. The newly created system of communicating agents is represented by `soca`.

```
> let soca = newSoCA "Example" stimSet ckaSet constants
```

After a new system of communicating agents has been created, agents need to be added. Each agent to be added to the system needs to be specified by providing the stimulus-response specification and the concrete behaviour specification in a text file (see Appendix C.5 for the detailed agent specification files corresponding to the running example). Each agent specification is loaded for use in the system of communicating agents using the `load` function and is added to the system of communicating agents using the `addAgent` function. In the following program fragment, five agents called `agentC`, `agentS`, `agentP`, `agentQ`, and `agentR` are loaded and added to the system of communicating agents represented by `soca`.

```
> agentC <- load "AgentC.txt"
> agentS <- load "AgentS.txt"
> agentP <- load "AgentP.txt"
> agentQ <- load "AgentQ.txt"
> agentR <- load "AgentR.txt"
> soca <- addAgent agentC soca
> soca <- addAgent agentS soca
> soca <- addAgent agentP soca
> soca <- addAgent agentQ soca
> soca <- addAgent agentR soca
```

Once all of the agents are added to the system of communicating agents, each of their specifications need to be generated for use with the Maude term rewriting system. This is done automatically using the `generateMaudeSpec` function. Additionally, all of the agent Maude specifications need to be combined with the pre-defined Maude files for C^2KA to generate a Maude specification for the system of communicating agents. The Maude specification for the system of communicating agents is also generated automatically using the `generateMaudeSOCA` function. In the following program fragment the Maude specifications for the agents represented by `agentC`, `agentS`, `agentP`, `agentQ`, and `agentR` are generated with respect to the system of communicating agents represented by `soca`. Then, the Maude specification for the system of communicating agents represented by `soca` is generated.

```
> generateMaudeSpec soca agentC
> generateMaudeSpec soca agentS
> generateMaudeSpec soca agentP
> generateMaudeSpec soca agentQ
> generateMaudeSpec soca agentR
> generateMaudeSOCA soca
```

The resulting system of communicating agents that has been defined using the prototype tool can be displayed using the `printSoCA` function. The system of communicating agents for the running example represented by `soca` is displayed using the following program fragment.

```
> printSoCA soca
  SoCA             : Example
  External Stimuli: {abor, allo, help, ips, noop, 𝔡, 𝔫}
  Behaviours       : {ABOR, ALLO, AVG1, AVG2, AVG3, AVG4, COUNT1, COUNT2,
                      COUNT3, COUNT4, DELTA, HELP, LAST, NOOP, READ,
                      RESET1, RESET2, RESET3, RESET4, SENDABOR, SENDALLO,
                      SENDHELP, SENDNOOP, SINCE1, SINCE2, SINCE3, SINCE4,
                      TICK, WAIT, 0, 1}
  Named Constants : {ABOR, ALLO, HELP, NOOP, true}
  Agents           : {C, P, Q, R, S}
```

The prototype tool can be used to automate the computations required at the abstract behaviour specification level. Consider the example context given in Section 4.2.6. The prototype tool can be used to automate the computation of the abstract behaviour of the composed agent $(\mathsf{S}\,;\mathsf{P})$ specified as $ips \circ (\mathsf{S}\,;\mathsf{P})$. This can be done using the `maude` function. The `maude` function is an interface function to the Maude rewriting system. It takes a Maude term as a string input and returns the rewrite result. The following program fragment shows the computation of the abstract behaviour of the composed agent $(\mathsf{S}\,;\mathsf{P})$. Note that the prototype tool requires the explicit specification of the behaviours of agents S and P.

```
> print $ maude "NB((’ips), ((’SENDABOR + ’SENDALLO + ’SENDHELP +
                  ’SENDNOOP) ; ((’ABOR ; ’COUNT1) + (’ALLO ; ’COUNT2) +
                  (’HELP ; ’COUNT3) + (’NOOP ; ’COUNT4))))"
  "((((((SENDABOR ; ABOR) ; COUNT1) + ((SENDALLO ; ALLO) ; COUNT2)) +
   ((SENDHELP ; HELP) ; COUNT3)) + SENDNOOP) ; NOOP) ; COUNT4"
```

C.2 Computing Orbits, Stabilisers, and Fixed Points

Once a system of communicating agents has been defined, the prototype tool can be used to compute the orbits, strong orbits, stabilisers, and fixed points for agent behaviours. For instance, the examples given in Section 4.2.7 can all be computed automatically using the prototype tool. In what follows, the `printSet` function prints a set representation of the resulting orbits, strong orbits, and stabilisers.

The computation of the orbits of agent behaviours can be done using the `ckaOrbit` function. The `ckaOrbit` function implements Definition 4.2.4(1) and takes the set of basic external stimuli and an agent behaviour and returns the orbit of the given agent behaviour. Some examples of computing orbits using the prototype tool with respect to the running example are given below.

```
> printSet $ ckaOrbit stimSet "ABOR"
  {0, ABOR, ALLO, HELP, NOOP}
> printSet $ ckaOrbit stimSet "READ"
  {0, READ}
> printSet $ ckaOrbit stimSet "COUNT3"
  {0, COUNT1, COUNT2, COUNT3, COUNT4}
```

Strong orbits of agent behaviours can be computed using the `ckaStrongOrbit` function. The `ckaStrongOrbit` function implements Definition 4.2.4(2) and takes the set of basic external stimuli, the set of basic agent behaviours, and an agent behaviour, and returns the

strong orbit of the given agent behaviour. Considering the running example, a selection of examples of computing strong orbits using the prototype tool are given below.

```
> printSet $ ckaStrongOrbit stimSet ckaSet "ABOR"
  {ABOR, ALLO, HELP, NOOP}
> printSet $ ckaStrongOrbit stimSet ckaSet "READ"
  {READ}
> printSet $ ckaStrongOrbit stimSet ckaSet "COUNT3"
  {COUNT1, COUNT2, COUNT3, COUNT4}
```

Similarly, the stabilisers of agent behaviours are computed using the `ckaStab` function which provides an implementation of Definition 4.2.4(3). There is a technical issue that is evident when computing the stabiliser of an agent behaviours. This is due to that fact that the stabiliser is computed with respect to the set of external stimuli S which is an infinite set. Because of this, it is required that the computation of the stabilisers be restricted to a set of external stimuli that contains all external stimuli up to a prescribed maximum length of the composed stimuli. Due to this, the `ckaStab` function takes the set of basic external stimuli, an integer representing the maximum length of the composed stimuli in the set of stimuli for which the stabiliser is to be computed with respect to, and an agent behaviour, and returns the stabiliser of the given agent behaviour computed with respect to the set of external stimuli generated by the given set of basic stimuli, up to the given maximum length of the compositions. The following commands show some examples of computing stabilisers using the prototype tool with respect to the running example. Each example shows the computation of the stabiliser with respect to only the set of basic external stimuli (i.e., maximum composition length 1) and again with the maximum length of the compositions set to 2. It should be noted that the stabiliser obtained from setting the maximum composition length to 1 is a subset of the stabiliser obtained from setting the maximum composition length to 2.

```
> printSet $ ckaStab stimSet 1 "ABOR"
  {abor, ips, n}
> printSet $ ckaStab stimSet 2 "ABOR"
  {abor, abor ⊙ abor, abor ⊙ ips, allo ⊙ abor, help ⊙ abor, ips,
   ips ⊙ abor, ips ⊙ ips, noop ⊙ abor, n}
> printSet $ ckaStab stimSet 1 "READ"
  {abor, allo, help, ips, noop, n}
> printSet $ ckaStab stimSet 2 "READ"
  {abor, abor ⊙ abor, abor ⊙ allo, abor ⊙ help, abor ⊙ ips, abor ⊙ noop,
   allo, allo ⊙ abor, allo ⊙ allo, allo ⊙ help, allo ⊙ ips, allo ⊙ noop,
   help, help ⊙ abor, help ⊙ allo, help ⊙ help, help ⊙ ips, help ⊙ noop,
   ips, ips ⊙ abor, ips ⊙ allo, ips ⊙ help, ips ⊙ ips, ips ⊙ noop, noop,
   noop ⊙ abor, noop ⊙ allo, noop ⊙ help, noop ⊙ ips, noop ⊙ noop, n}
> printSet $ ckaStab stimSet 1 "COUNT3"
  {help, ips, n}
> printSet $ ckaStab stimSet 2 "COUNT3"
  {abor ⊙ help, allo ⊙ help, help, help ⊙ help, help ⊙ ips, ips,
   ips ⊙ help, ips ⊙ ips, noop ⊙ help, n}
```

Finally, the prototype tool can be used to verify whether a given agent behaviour is a fixed point. This can be done using the `ckaFixedPoint` function which implements Definition 4.2.4(4) and takes the set of basic external stimuli and an agent behaviour and verifies whether the given agent behaviour is a fixed point with respect to the next behaviour mapping ∘. The following commands show how the prototype tool can be used to verify whether an agent behaviour is a fixed point.

```
> print $ ckaFixedPoint stimSet "READ"
  True
```

```
> print $ ckaFixedPoint stimSet "TICK"
  True
> print $ ckaFixedPoint stimSet "ABOR"
  False
> print $ ckaFixedPoint stimSet "COUNT3"
  False
```

It is important to note that each of the above functions for computing orbits, strong orbits, stabilisers, and fixed points with respect to the next behaviour mapping $\circ$ have counterparts for computing orbits, strong orbits, stabilisers, and fixed points with respect to the next stimulus mapping λ. In the prototype tool, these functions are denoted by `stimOrbit`, `stimStrongOrbit`, `stimStab`, and `stimFixedPoint`, respectively. Each of these functions are used in a very similar way to their counterparts demonstrated above.

C.3 Verifying Stimuli-Connected Systems, Communication Fixed Points, and Universally Influential Agents

The prototype tool can be used to verify whether a given system of communicating agents is stimuli-connected. It can additionally verify whether a given agent in a system of communicating agents is a communication fixed point or whether it is universally influential. The `isStimConnected` function implements Definition 5.2.4 and is used to verify whether a given system of communicating agents is stimuli-connected. This function takes a system of communicating agents as input and returns whether the given system is stimuli-connected. The running example, represented by `soca`, can be checked for stimuli-connectedness using the following command:

```
> print $ isStimConnected soca
False
```

In order to determine whether a given system agent is a communication fixed point using the prototype tool, the `isCommFixedPoint` function is used. This function implements Definition 5.2.5. It takes a system of communicating agents and an agent as input, and returns whether the given agent is a communication fixed point in the given system of communicating agents. The following commands show how the prototype tool can be used to verify whether an agent is a communication fixed point.

```
> print $ isCommFixedPoint soca agentC
  True
> print $ isCommFixedPoint soca agentS
  False
> print $ isCommFixedPoint soca agentP
  False
> print $ isCommFixedPoint soca agentQ
  False
> print $ isCommFixedPoint soca agentR
  True
```

Similarly, the prototype tool can be used to determine whether a given system agent is universally influential. The `isUniversallyInfluential` function implements Definition 5.2.6, taking a system of communicating agents and an agent as input, and returning whether the given agent is universally influential in the given system of communicating agents. The following commands show how the prototype tool can be used to verify whether an agent is universally influential.

```
> print $ isUniversallyInfluential soca agentC
  False
> print $ isUniversallyInfluential soca agentS
  False
```

```
> print $ isUniversallyInfluential soca agentP
  False
> print $ isUniversallyInfluential soca agentQ
  False
> print $ isUniversallyInfluential soca agentR
  False
```

C.4 Verifying the Potential for Communication Condition

With consideration of the specification of the running example system of communicating agents of Section 4.1, the prototype tool can be used to automatically verify the potential for communication amongst the system agents.

The potential for direct communication and the potential for communication via external stimuli can be verified using the `dPFCviaStim` and `pfcViaStim` functions, respectively. The `dPFCviaStim` provides an implementation of Definition 5.2.1 and the `pfcViaStim` function implements Definition 5.2.3. Each of these functions take a system of communicating agents and two agents and verifies if there is the a potential for communication via external stimuli from the first agent to the second agent. These functions simply return a boolean value representing the satisfaction of the specified potential for communication. The commands for verifying the satisfaction of the potential for direct communication and the potential for communication via external stimuli between two agents in the running example system of communicating agents and corresponding to the examples given in Section 5.3 are shown below.

```
> print $ dPFCviaStim soca agentS agentP
  True
> print $ dPFCviaStim soca agentP agentR
  False
```

```
> print $ pfcViaStim soca agentS agentQ
  True
> print $ pfcViaStim soca agentS agentR
  False
```

Similarly, the potential for direct communication and the potential for communication via shared environments can be verified using the `dPFCviaEnv` and `pfcViaEnv` functions, respectively. The `dPFCviaEnv` provides an implementation of Definition 5.2.7 and the `pfcViaEnv` function implements Definition 5.2.8. When verifying the potential for direct communication and the potential for communication via shared environments with the prototype tool, the dependence relation R is generated as a definition-reference relation between program variables in the concrete behaviour specifications of agents. It is in this generation of the definition-reference relation that the prototype tool uses the set of defined constants for the given system of communicating agents so that they can be excluded from the relation. Some example commands for verifying the satisfaction of the potential for direct communication and the potential for communication via shared environments between two agents in the running example system of communicating agents and corresponding to the examples given in Section 5.3 are shown below.

```
> print $ dPFCviaEnv soca agentP agentR
  True
> print $ dPFCviaEnv soca agentC agentP
  False
> print $ pfcViaEnv soca agentC agentR
  True
> print $ pfcViaEnv soca agentQ agentP
  False
```

Finally, the prototype tool can be used to automatically verify the potential for communication between two agents in a system of communicating agents using the `pfc` function which provides an implementation of Definition 5.2.10. The `pfc` function takes a system of communicating agents and two agents and verifies if there is a potential for communication from the first agent to the second agent. It returns a pair where the first component represents the satisfaction of the potential for communication condition and the second component is a list of all of the possible communication paths with the first agent as the source and the second agent as the sink. In this way, the second component of the returned pair represents all of the possible communication paths or patterns of communication from the first agent to the second agent. The commands for verifying the satisfaction of the potential for communication between two agents in the running example system of communicating agents and corresponding to the examples given in Section 5.3 are shown below where the `printPaths` function prints and formats the list of all possible communication paths or patterns of communication.

```
> let (condSR, pathsSR) = pfc soca agentS agentR
> print $ condSR
  True
> printPaths $ pathsSR
  S  ->S  P  ->E  R
  S  ->S  P  ->S  Q  ->E  R
  S  ->S  Q  ->S  P  ->E  R
  S  ->S  Q  ->E  R

> let (condCS, pathsCS) = pfc soca agentC agentS
> print $ condCS
  False
```

```
> printPaths $ pathsCS

  .
```

From the results of the `pfc` function and the prototype tool, it is easy to see that $(\mathsf{S} \rightsquigarrow^{+} \mathsf{R})$ and that there are multiple communication paths or patterns of communication that achieve this. Therefore, the potential for communication condition is satisfied. Additionally, it is easy to see that $\neg(\mathsf{C} \rightsquigarrow^{+} \mathsf{S})$. Since the potential for communication condition between agent C and agent S is false, there is no potential for communication and therefore there are no potential communication paths or patterns of communication that are printed.

C.5 Agent Behaviour Specifications for the Prototype Tool

This section contains each of the detailed agent specification files for the running example described in Section 4.1. The specifications correspond to the stimulus-response specification for each agent as given in Tables 4.1 to 4.5, the abstract behaviour specifications as given in Figure 4.2, and the concrete behaviour specifications as given in Figures 4.3 to 4.7.

C.5.1 Behaviour Specification File for Agent C (`AgentC.txt`)

```
begin AGENT where

    C := TICK

end

begin NEXT_BEHAVIOUR where

    (ips,TICK)  = TICK
    (abor,TICK) = TICK
    (allo,TICK) = TICK
    (help,TICK) = TICK
    (noop,TICK) = TICK

end
```

```
begin NEXT_STIMULUS where

    (ips,TICK)  = N
    (abor,TICK) = N
    (allo,TICK) = N
    (help,TICK) = N
    (noop,TICK) = N

end

begin CONCRETE_BEHAVIOUR where

    C => [ time := time + 1 ]

end
```

C.5.2 Behaviour Specification File for Agent S (AgentS.txt)

```
begin AGENT where

    S := SENDABOR + SENDALLO + SENDHELP + SENDNOOP

end

begin NEXT_BEHAVIOUR where

    (abor,SENDABOR) = SENDABOR
    (abor,SENDALLO) = SENDALLO
    (abor,SENDHELP) = SENDHELP
    (abor,SENDNOOP) = SENDNOOP
    (allo,SENDABOR) = SENDABOR
    (allo,SENDALLO) = SENDALLO
    (allo,SENDHELP) = SENDHELP
    (allo,SENDNOOP) = SENDNOOP
    (help,SENDABOR) = SENDABOR
    (help,SENDALLO) = SENDALLO
    (help,SENDHELP) = SENDHELP
```

```
    (help,SENDNOOP) = SENDNOOP
    (noop,SENDABOR) = SENDABOR
    (noop,SENDALLO) = SENDALLO
    (noop,SENDHELP) = SENDHELP
    (noop,SENDNOOP) = SENDNOOP

end

begin NEXT_STIMULUS where

    (abor,SENDABOR) = abor
    (abor,SENDALLO) = allo
    (abor,SENDHELP) = help
    (abor,SENDNOOP) = noop
    (allo,SENDABOR) = abor
    (allo,SENDALLO) = allo
    (allo,SENDHELP) = help
    (allo,SENDNOOP) = noop
    (help,SENDABOR) = abor
    (help,SENDALLO) = allo
    (help,SENDHELP) = help
    (help,SENDNOOP) = noop
    (noop,SENDABOR) = abor
    (noop,SENDALLO) = allo
    (noop,SENDHELP) = help
    (noop,SENDNOOP) = noop

end

begin CONCRETE_BEHAVIOUR where

    S => [ if true -> x := ABOR
            | true -> x := ALLO
            | true -> x := HELP
            | true -> x := NOOP
           fi;
           send x ]

end
```

C.5.3 Behaviour Specification File for Agent P (AgentP.txt)

```
begin AGENT where

    P := (ABOR ; COUNT1) + (ALLO ; COUNT2) + (HELP ; COUNT3) + (NOOP ; COUNT4)

end

begin NEXT_BEHAVIOUR where

    (ips,ABOR)    = ABOR
    (ips,ALLO)    = ALLO
    (ips,HELP)    = HELP
    (ips,NOOP)    = NOOP
    (ips,COUNT1)  = COUNT1
    (ips,COUNT2)  = COUNT2
    (ips,COUNT3)  = COUNT3
    (ips,COUNT4)  = COUNT4
    (abor,ABOR)   = ABOR
    (abor,ALLO)   = ABOR
    (abor,HELP)   = ABOR
    (abor,NOOP)   = ABOR
    (abor,COUNT1) = COUNT1
    (abor,COUNT2) = COUNT1
    (abor,COUNT3) = COUNT1
    (abor,COUNT4) = COUNT1
    (allo,ABOR)   = ALLO
    (allo,ALLO)   = ALLO
    (allo,HELP)   = ALLO
    (allo,NOOP)   = ALLO
    (allo,COUNT1) = COUNT2
    (allo,COUNT2) = COUNT2
    (allo,COUNT3) = COUNT2
    (allo,COUNT4) = COUNT2
    (help,ABOR)   = HELP
    (help,ALLO)   = HELP
    (help,HELP)   = HELP
    (help,NOOP)   = HELP
```

```
    (help,COUNT1) = COUNT3
    (help,COUNT2) = COUNT3
    (help,COUNT3) = COUNT3
    (help,COUNT4) = COUNT3
    (noop,ABOR)   = NOOP
    (noop,ALLO)   = NOOP
    (noop,HELP)   = NOOP
    (noop,NOOP)   = NOOP
    (noop,COUNT1) = COUNT4
    (noop,COUNT2) = COUNT4
    (noop,COUNT3) = COUNT4
    (noop,COUNT4) = COUNT4

end

begin NEXT_STIMULUS where

    (ips,ABOR)    = N
    (ips,ALLO)    = N
    (ips,HELP)    = N
    (ips,NOOP)    = N
    (ips,COUNT1)  = N
    (ips,COUNT2)  = N
    (ips,COUNT3)  = N
    (ips,COUNT4)  = N
    (abor,ABOR)   = abor
    (abor,ALLO)   = abor
    (abor,HELP)   = abor
    (abor,NOOP)   = abor
    (abor,COUNT1) = abor
    (abor,COUNT2) = abor
    (abor,COUNT3) = abor
    (abor,COUNT4) = abor
    (allo,ABOR)   = allo
    (allo,ALLO)   = allo
    (allo,HELP)   = allo
    (allo,NOOP)   = allo
    (allo,COUNT1) = allo
```

```
    (allo,COUNT2) = allo
    (allo,COUNT3) = allo
    (allo,COUNT4) = allo
    (help,ABOR)   = help
    (help,ALLO)   = help
    (help,HELP)   = help
    (help,NOOP)   = help
    (help,COUNT1) = help
    (help,COUNT2) = help
    (help,COUNT3) = help
    (help,COUNT4) = help
    (noop,ABOR)   = noop
    (noop,ALLO)   = noop
    (noop,HELP)   = noop
    (noop,NOOP)   = noop
    (noop,COUNT1) = noop
    (noop,COUNT2) = noop
    (noop,COUNT3) = noop
    (noop,COUNT4) = noop

end

begin CONCRETE_BEHAVIOUR where

    P => [ receive x;
           if x >= ABOR -> cmd := 1 ; num1 := num1 + 1
            | x >= ALLO -> cmd := 2 ; num2 := num2 + 1
            | x >= HELP -> cmd := 3 ; num3 := num3 + 1
            | x >= NOOP -> cmd := 4 ; num4 := num4 + 1
           fi;
           send x ]
end
```

C.5.4 Behaviour Specification File for Agent Q (`AgentQ.txt`)

```
begin AGENT where

    Q := (DELTA ; LAST) + WAIT

end

begin NEXT_BEHAVIOUR where

    (ips,DELTA)  = WAIT
    (ips,LAST)   = WAIT
    (ips,WAIT)   = WAIT
    (abor,DELTA) = DELTA
    (abor,LAST)  = LAST
    (abor,WAIT)  = DELTA
    (allo,DELTA) = DELTA
    (allo,LAST)  = LAST
    (allo,WAIT)  = DELTA
    (help,DELTA) = DELTA
    (help,LAST)  = LAST
    (help,WAIT)  = DELTA
    (noop,DELTA) = DELTA
    (noop,LAST)  = LAST
    (noop,WAIT)  = DELTA

end

begin NEXT_STIMULUS where

    (ips,DELTA)  = N
    (ips,LAST)   = N
    (ips,WAIT)   = N
    (abor,DELTA) = abor
    (abor,LAST)  = abor
    (abor,WAIT)  = abor
    (allo,DELTA) = allo
    (allo,LAST)  = allo
```

```
    (allo,WAIT)  = allo
    (help,DELTA) = help
    (help,LAST)  = help
    (help,WAIT)  = help
    (noop,DELTA) = noop
    (noop,LAST)  = noop
    (noop,WAIT)  = noop

end

begin CONCRETE_BEHAVIOUR where

    Q => [ receive x;
           if  (x >= ABOR || x >= ALLO || x >= HELP || x >= NOOP) -> delta := time - last; last := time
            | ~(x >= ABOR || x >= ALLO || x >= HELP || x >= NOOP) -> skip
           fi;
           send x ]

end
```

C.5.5 Behaviour Specification File for Agent R (AgentR.txt)

```
begin AGENT where

    R := SINCE1 ; SINCE2 ; SINCE3; SINCE4 ; READ ; ((AVG1 ; RESET1) + (AVG2 ; RESET2) +
           (AVG3 ; RESET3) + (AVG4 ; RESET4))

end

begin NEXT_BEHAVIOUR where

    (ips,SINCE1)  = SINCE1
    (ips,SINCE2)  = SINCE2
    (ips,SINCE3)  = SINCE3
    (ips,SINCE4)  = SINCE4
    (ips,READ)    = READ
    (ips,AVG1)    = AVG1
    (ips,AVG2)    = AVG2
```

```
(ips,AVG3)    = AVG3
(ips,AVG4)    = AVG4
(ips,RESET1)  = RESET1
(ips,RESET2)  = RESET2
(ips,RESET3)  = RESET3
(ips,RESET4)  = RESET4
(abor,SINCE1) = SINCE1
(abor,SINCE2) = SINCE2
(abor,SINCE3) = SINCE3
(abor,SINCE4) = SINCE4
(abor,READ)   = READ
(abor,AVG1)   = AVG1
(abor,AVG2)   = AVG2
(abor,AVG3)   = AVG3
(abor,AVG4)   = AVG4
(abor,RESET1) = RESET1
(abor,RESET2) = RESET2
(abor,RESET3) = RESET3
(abor,RESET4) = RESET4
(allo,SINCE1) = SINCE1
(allo,SINCE2) = SINCE2
(allo,SINCE3) = SINCE3
(allo,SINCE4) = SINCE4
(allo,READ)   = READ
(allo,AVG1)   = AVG1
(allo,AVG2)   = AVG2
(allo,AVG3)   = AVG3
(allo,AVG4)   = AVG4
(allo,RESET1) = RESET1
(allo,RESET2) = RESET2
(allo,RESET3) = RESET3
(allo,RESET4) = RESET4
(help,SINCE1) = SINCE1
(help,SINCE2) = SINCE2
(help,SINCE3) = SINCE3
(help,SINCE4) = SINCE4
(help,READ)   = READ
(help,AVG1)   = AVG1
```

```
    (help,AVG2)   = AVG2
    (help,AVG3)   = AVG3
    (help,AVG4)   = AVG4
    (help,RESET1) = RESET1
    (help,RESET2) = RESET2
    (help,RESET3) = RESET3
    (help,RESET4) = RESET4
    (noop,SINCE1) = SINCE1
    (noop,SINCE2) = SINCE2
    (noop,SINCE3) = SINCE3
    (noop,SINCE4) = SINCE4
    (noop,READ)   = READ
    (noop,AVG1)   = AVG1
    (noop,AVG2)   = AVG2
    (noop,AVG3)   = AVG3
    (noop,AVG4)   = AVG4
    (noop,RESET1) = RESET1
    (noop,RESET2) = RESET2
    (noop,RESET3) = RESET3
    (noop,RESET4) = RESET4

end

begin NEXT_STIMULUS where

    (ips,SINCE1)  = N
    (ips,SINCE2)  = N
    (ips,SINCE3)  = N
    (ips,SINCE4)  = N
    (ips,READ)    = N
    (ips,AVG1)    = N
    (ips,AVG2)    = N
    (ips,AVG3)    = N
    (ips,AVG4)    = N
    (ips,RESET1)  = N
    (ips,RESET2)  = N
    (ips,RESET3)  = N
    (ips,RESET4)  = N
```

```
(abor,SINCE1) = N
(abor,SINCE2) = N
(abor,SINCE3) = N
(abor,SINCE4) = N
(abor,READ)   = N
(abor,AVG1)   = N
(abor,AVG2)   = N
(abor,AVG3)   = N
(abor,AVG4)   = N
(abor,RESET1) = N
(abor,RESET2) = N
(abor,RESET3) = N
(abor,RESET4) = N
(allo,SINCE1) = N
(allo,SINCE2) = N
(allo,SINCE3) = N
(allo,SINCE4) = N
(allo,READ)   = N
(allo,AVG1)   = N
(allo,AVG2)   = N
(allo,AVG3)   = N
(allo,AVG4)   = N
(allo,RESET1) = N
(allo,RESET2) = N
(allo,RESET3) = N
(allo,RESET4) = N
(help,SINCE1) = N
(help,SINCE2) = N
(help,SINCE3) = N
(help,SINCE4) = N
(help,READ)   = N
(help,AVG1)   = N
(help,AVG2)   = N
(help,AVG3)   = N
(help,AVG4)   = N
(help,RESET1) = N
(help,RESET2) = N
(help,RESET3) = N
```

```
    (help,RESET4) = N
    (noop,SINCE1) = N
    (noop,SINCE2) = N
    (noop,SINCE3) = N
    (noop,SINCE4) = N
    (noop,READ)   = N
    (noop,AVG1)   = N
    (noop,AVG2)   = N
    (noop,AVG3)   = N
    (noop,AVG4)   = N
    (noop,RESET1) = N
    (noop,RESET2) = N
    (noop,RESET3) = N
    (noop,RESET4) = N

end

begin CONCRETE_BEHAVIOUR where

    R => [ since1 := since1 + delta;
           since2 := since2 + delta;
           since3 := since3 + delta;
           since4 := since4 + delta;
           n := cmd;
           if n = 1 -> avg1 := since1 / num1; since1 := 0
            | n = 2 -> avg2 := since2 / num2; since2 := 0
            | n = 3 -> avg3 := since3 / num3; since3 := 0
            | n = 4 -> avg4 := since4 / num4; since4 := 0
           fi ]

end
```

Appendix D

Analysing Agent Knowledge Using the SPASS Theorem Prover

In what follows, a brief overview of the usage of the SPASS theorem prover [TST14] for specifying and analysing the agent knowledge in the running example system of communicating agents described in Section 4.1 is provided.

D.1 Verifying the Constraint on Communication Condition

With consideration of the specification of the running example system of communicating agents of Section 4.1, the SPASS theorem prover can be used to support the automation of the verification of the constraint on communication condition in a system of communicating agents.

The detailed specification of each agent knowledge base using the SPASS theorem prover is shown in Section D.2. In each SPASS specification, under the `list_of_special_formulae` section, there is an implementation of the condition for determining whether the given agent is a potential source of confidential information leakage (see Section 6.4). For instance, in each of the agent specifications for the running example system of communicating agents,

there is a conjecture corresponding to the formula $\exists(\mathsf{Z} \mid \mathsf{Z} \in N_O : \mathcal{N} \models \mathsf{ConfInfo}(\mathsf{Z}))$, denoted as `formula( exists( [z], ConfInfo( z ) ) )` in SPASS. In essence, this formula verifies whether the given agent knows any confidential information (i.e., any concept assertion that associates some object Z to the concept ConfInfo).

The SPASS theorem prover is run on each specification file to determine if the formula is satisfied with respect to the agent knowledge base specified in the given file. Running SPASS results in the final output `SPASS beiseite: Proof found` if the formula is valid or `SPASS beiseite: Completion found` if the formula is not valid [TST14]. An example of using the SPASS theorem prover to determine if agent S in the running example system of communicating agents is a potential source of confidential information leakage is given below.

```
> SPASS AgentS.dfg

-------------------------SPASS-START-----------------------------
Input Problem:
1[0:Inp] ||  -> Command(abor)*.
2[0:Inp] ||  -> Command(allo)*.
3[0:Inp] ||  -> Command(help)*.
4[0:Inp] ||  -> Command(noop)*.
5[0:Inp] ||  -> Enumeration(1)*.
6[0:Inp] ||  -> Enumeration(2)*.
7[0:Inp] ||  -> Enumeration(3)*.
8[0:Inp] ||  -> Enumeration(4)*.
9[0:Inp] ||  -> BitString(0)*.
10[0:Inp] ||  -> BitString(1)*.
11[0:Inp] ||  -> BitString(10)*.
12[0:Inp] ||  -> BitString(11)*.
13[0:Inp] ||  -> ConfInfo(1)*.
14[0:Inp] || ConfInfo(U)* -> .
15[0:Inp] ||  -> Variable(avg_1,0)*.
16[0:Inp] ||  -> Variable(avg_2,0)*.
17[0:Inp] ||  -> Variable(avg_3,0)*.
18[0:Inp] ||  -> Variable(avg_4,0)*.
```

```
19[0:Inp] ||  -> Cmd_To_Enum(abor,1)*.
20[0:Inp] ||  -> Cmd_To_Enum(allo,2)*.
21[0:Inp] ||  -> Cmd_To_Enum(help,3)*.
22[0:Inp] ||  -> Cmd_To_Enum(noop,4)*.
23[0:Inp] ||  -> Variable(time,0)*.
24[0:Inp] ||  -> Variable(delta,0)*.
25[0:Inp] || ConfInfo(U) -> BitString(U)*.
26[0:Inp] || Enum_To_Cmd(U,V)* -> Cmd_To_Enum(V,U).
27[0:Inp] || Cmd_To_Enum(U,V) -> Enum_To_Cmd(V,U)*.
 This is a first-order Horn problem without equality.
 This is a problem that has, if any, a finite domain model.
 There are no function symbols.
 This is a problem that contains sort information.
 The following monadic predicates have finite extensions: ConfInfo, Enumeration, Command.
 Axiom clauses: 26 Conjecture clauses: 1
 Inferences: IEmS=1 ISoR=1 IORe=1
 Reductions: RFMRR=1 RBMRR=1 RObv=1 RUnC=1 RTaut=1 RSST=1 RSSi=1 RFSub=1 RBSub=1 RCon=1
 Extras    : Input Saturation, Always Selection, No Splitting, Full Reduction, Ratio: 5,
      FuncWeight: 1, VarWeight: 1
 Precedence: command > Command > bitstring > BitString > enumeration > Enumeration >
      confinfo > ConfInfo > variable > Variable > cmd_to_enum > Enum_To_Cmd >
      Cmd_To_Enum > enum_to_cmd > abor > allo > help > noop > 0 > 1 > 10 > 11 > 2 >
      3 > 4 > cmd > delta > last > time > num_1 > num_2 > num_3 > num_4 > since_1 >
      since_2 > since_3 > since_4 > avg_1 > avg_2 > avg_3 > avg_4
 Ordering  : KBO
Processed Problem:

Worked Off Clauses:

Usable Clauses:
13[0:Inp] ||  -> ConfInfo(1)*.
8[0:Inp] ||  -> Enumeration(4)*.
7[0:Inp] ||  -> Enumeration(3)*.
6[0:Inp] ||  -> Enumeration(2)*.
5[0:Inp] ||  -> Enumeration(1)*.
4[0:Inp] ||  -> Command(noop)*.
3[0:Inp] ||  -> Command(help)*.
2[0:Inp] ||  -> Command(allo)*.
```

```
1[0:Inp] ||  -> Command(abor)*.
12[0:Inp] ||  -> BitString(11)*.
11[0:Inp] ||  -> BitString(10)*.
10[0:Inp] ||  -> BitString(1)*.
9[0:Inp] ||  -> BitString(0)*.
24[0:Inp] ||  -> Variable(delta,0)*.
23[0:Inp] ||  -> Variable(time,0)*.
22[0:Inp] ||  -> Cmd_To_Enum(noop,4)*.
21[0:Inp] ||  -> Cmd_To_Enum(help,3)*.
20[0:Inp] ||  -> Cmd_To_Enum(allo,2)*.
19[0:Inp] ||  -> Cmd_To_Enum(abor,1)*.
18[0:Inp] ||  -> Variable(avg_4,0)*.
17[0:Inp] ||  -> Variable(avg_3,0)*.
16[0:Inp] ||  -> Variable(avg_2,0)*.
15[0:Inp] ||  -> Variable(avg_1,0)*.
25[0:Inp] ConfInfo(U) ||  -> BitString(U)*.
27[0:Inp] || Cmd_To_Enum(U,V) -> Enum_To_Cmd(V,U)*.
26[0:Inp] || Enum_To_Cmd(U,V)* -> Cmd_To_Enum(V,U).
SPASS V 3.7
SPASS beiseite: Proof found.
Problem: AgentS.dfg
SPASS derived 0 clauses, backtracked 0 clauses, performed 0 splits and kept 24 clauses.
SPASS allocated 45899 KBytes.
SPASS spent 0:00:00.04 on the problem.
0:00:00.01 for the input.
0:00:00.01 for the FLOTTER CNF translation, of which
0:00:00.00 for the translation from EML to FOL.
0:00:00.00 for inferences.
0:00:00.00 for the backtracking.
0:00:00.00 for the reduction.
-------------------------SPASS-STOP-----------------------------
```

In the output from the SPASS theorem prover, it can be seen that the result is `SPASS beiseite: Proof found`, meaning that the formula is satisfied. This means that in the running example, agent S is a potential source for confidential information leakage. This confirms what was determined in Section 6.4.

In general, the SPASS theorem prover can be run for each agent knowledge base specification for a given system of communicating agents. If any of the results show that there exists a potential source for confidential information leakage then the constraint on communication condition is satisfied for the given system of communicating agents.

D.2 Agent Knowledge Specifications for the SPASS Theorem Prover

This section contains each of the detailed agent knowledge specification files for the running example described in Section 4.3.2. The specifications correspond to the knowledge bases specified using the description logic $\mathcal{ALB}$ in Figures 4.10 to 4.14 in Section 4.3.1.

D.2.1 Knowledge Specification File for Agent C (AgentC.dfg)

```
begin_problem(AgentC).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FILE DESCRIPTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
list_of_descriptions.

    name({* AgentC.dfg *}).
    author({* Jason Jaskolka *}).
    status(unknown).
    description
    ({*  Initial Knowledge Base for Agent C.  *}).

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SYMBOL LIST
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
list_of_symbols.

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % OBJECTS
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    functions
    [
        (abor,0),
        (allo,0),
        (help,0),
        (noop,0),
        (00,0),
        (01,0),
        (10,0),
        (11,0),
        (0,0),
        (1,0),
        (2,0),
        (3,0),
        (4,0),
        (cmd,0),
        (delta,0),
        (last,0),
        (time,0),
        (num_1,0),
        (num_2,0),
        (num_3,0),
        (num_4,0),
        (since_1,0),
        (since_2,0),
        (since_3,0),
        (since_4,0),
        (avg_1,0),
        (avg_2,0),
        (avg_3,0),
        (avg_4,0)
    ].
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONCEPTS AND ROLES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
predicates
[
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % CONCEPTS
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        (command,0),            (Command,1),
        (bitstring,0),          (BitString,1),
        (enumeration,0),        (Enumeration,1),
        (confinfo,0),           (ConfInfo,1),
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % ROLES
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        (variable,0),           (Variable,2),
        (cmd_to_enum,0),        (Cmd_To_Enum,2),
        (enum_to_cmd,0),        (Enum_To_Cmd,2)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
].
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TRANSLATIONS
%     Concept/Role Names  : lowercase
%     Individual Types    : Capitalised
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
translpairs
[
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % CONCEPTS
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        (command,Command),
        (bitstring,BitString),
        (enumeration,Enumeration),
        (confinfo,ConfInfo),
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
        % ROLES
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            (variable,Variable),
            (cmd_to_enum,Cmd_To_Enum),
            (enum_to_cmd,Enum_To_Cmd)
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ].
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% KNOWLEDGE BASE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
list_of_special_formulae(axioms, DL).

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % TBox
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % EMPTY
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % ABox
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        formula( Variable( time, 0 ) ).
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%--------------------------------------------------------------------------%
% CONJECTURES (GOALS)
%--------------------------------------------------------------------------%
list_of_special_formulae(conjectures, DL).

        formula( exists( [z], ConfInfo( z ) ) ).

end_of_list.
%--------------------------------------------------------------------------%

end_problem.
```

D.2.2 Knowledge Specification File for Agent S (AgentS.dfg)

```
begin_problem(AgentS).

%--------------------------------------------------------------------------%
% FILE DESCRIPTION
%--------------------------------------------------------------------------%
list_of_descriptions.

    name({* AgentS.dfg *}).
    author({* Jason Jaskolka *}).
    status(unknown).
    description
    ({*  Initial Knowledge Base for Agent S.  *}).

end_of_list.
%--------------------------------------------------------------------------%

%--------------------------------------------------------------------------%
% SYMBOL LIST
%--------------------------------------------------------------------------%
list_of_symbols.

    %----------------------------------------------------------------------%
    % OBJECTS
    %----------------------------------------------------------------------%
```

```
functions
[
    (abor,0),
    (allo,0),
    (help,0),
    (noop,0),
    (00,0),
    (01,0),
    (10,0),
    (11,0),
    (0,0),
    (1,0),
    (2,0),
    (3,0),
    (4,0),
    (cmd,0),
    (delta,0),
    (last,0),
    (time,0),
    (num_1,0),
    (num_2,0),
    (num_3,0),
    (num_4,0),
    (since_1,0),
    (since_2,0),
    (since_3,0),
    (since_4,0),
    (avg_1,0),
    (avg_2,0),
    (avg_3,0),
    (avg_4,0)
].
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONCEPTS AND ROLES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
predicates
[
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % CONCEPTS
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        (command,0),            (Command,1),
        (bitstring,0),          (BitString,1),
        (enumeration,0),        (Enumeration,1),
        (confinfo,0),           (ConfInfo,1),
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % ROLES
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        (variable,0),           (Variable,2),
        (cmd_to_enum,0),        (Cmd_To_Enum,2),
        (enum_to_cmd,0),        (Enum_To_Cmd,2)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
].
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TRANSLATIONS
%     Concept/Role Names  : lowercase
%     Individual Types    : Capitalised
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
translpairs
[
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % CONCEPTS
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        (command,Command),
        (bitstring,BitString),
        (enumeration,Enumeration),
        (confinfo,ConfInfo),
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
        % ROLES
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            (variable,Variable),
            (cmd_to_enum,Cmd_To_Enum),
            (enum_to_cmd,Enum_To_Cmd)
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ].
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% KNOWLEDGE BASE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
list_of_special_formulae(axioms, DL).

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % TBox
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % All confidential information are bit-strings
        concept_formula( implies( confinfo, bitstring ) ).

        % Enum_To_Cmd is the inverse mapping of Cmd_To_Enum
        role_formula( equiv( enum_to_cmd, conv( cmd_to_enum ) ) ).
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % ABox
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        formula( Command( abor ) ).
        formula( Command( allo ) ).
        formula( Command( help ) ).
        formula( Command( noop ) ).

        formula( Enumeration( 1 ) ).
        formula( Enumeration( 2 ) ).
        formula( Enumeration( 3 ) ).
```

```
        formula( Enumeration( 4 ) ).

        formula( BitString( 00 ) ).
        formula( BitString( 01 ) ).
        formula( BitString( 10 ) ).
        formula( BitString( 11 ) ).

        formula( Variable( avg_1, 0 ) ).
        formula( Variable( avg_2, 0 ) ).
        formula( Variable( avg_3, 0 ) ).
        formula( Variable( avg_4, 0 ) ).

        formula( Cmd_To_Enum( abor, 1 ) ).
        formula( Cmd_To_Enum( allo, 2 ) ).
        formula( Cmd_To_Enum( help, 3 ) ).
        formula( Cmd_To_Enum( noop, 4 ) ).

        formula( Variable( time, 0 ) ).
        formula( Variable( delta, 0 ) ).

        formula( ConfInfo( 01 ) ).
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONJECTURES (GOALS)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
list_of_special_formulae(conjectures, DL).

        formula( exists( [z], ConfInfo( z ) ) ).

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end_problem.
```

D.2.3 Knowledge Specification File for Agent P (AgentP.dfg)

```
begin_problem(AgentP).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FILE DESCRIPTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
list_of_descriptions.



    name({* AgentP.dfg *}).
    author({* Jason Jaskolka *}).
    status(unknown).
    description
    ({*  Initial Knowledge Base for Agent P.  *}).

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SYMBOL LIST
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
list_of_symbols.

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % OBJECTS
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    functions
    [
        (abor,0),
        (allo,0),
        (help,0),
        (noop,0),
        (00,0),
        (01,0),
        (10,0),
        (11,0),
        (0,0),
```

```
    (1,0),
    (2,0),
    (3,0),
    (4,0),
    (cmd,0),
    (delta,0),
    (last,0),
    (time,0),
    (num_1,0),
    (num_2,0),
    (num_3,0),
    (num_4,0),
    (since_1,0),
    (since_2,0),
    (since_3,0),
    (since_4,0),
    (avg_1,0),
    (avg_2,0),
    (avg_3,0),
    (avg_4,0)
].
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONCEPTS AND ROLES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
predicates
[
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % CONCEPTS
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        (command,0),            (Command,1),
        (bitstring,0),          (BitString,1),
        (enumeration,0),        (Enumeration,1),
        (confinfo,0),           (ConfInfo,1),
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % ROLES
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
            (variable,0),           (Variable,2),
            (cmd_to_enum,0),        (Cmd_To_Enum,2),
            (enum_to_cmd,0),        (Enum_To_Cmd,2)
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ].
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % TRANSLATIONS
    %     Concept/Role Names  : lowercase
    %     Individual Types    : Capitalised
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    translpairs
    [
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % CONCEPTS
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            (command,Command),
            (bitstring,BitString),
            (enumeration,Enumeration),
            (confinfo,ConfInfo),
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % ROLES
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            (variable,Variable),
            (cmd_to_enum,Cmd_To_Enum),
            (enum_to_cmd,Enum_To_Cmd)
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ].
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% KNOWLEDGE BASE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
list_of_special_formulae(axioms, DL).

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % TBox
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Enum_To_Cmd is the inverse mapping of Cmd_To_Enum
        role_formula( equiv( enum_to_cmd, conv( cmd_to_enum ) ) ).
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % ABox
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        formula( Command( abor ) ).
        formula( Command( allo ) ).
        formula( Command( help ) ).
        formula( Command( noop ) ).

        formula( Enumeration( 1 ) ).
        formula( Enumeration( 2 ) ).
        formula( Enumeration( 3 ) ).
        formula( Enumeration( 4 ) ).

        formula( BitString( 00 ) ).
        formula( BitString( 01 ) ).
        formula( BitString( 10 ) ).
        formula( BitString( 11 ) ).

        formula( Variable( num_1, 0 ) ).
        formula( Variable( num_2, 0 ) ).
        formula( Variable( num_3, 0 ) ).
        formula( Variable( num_4, 0 ) ).

        formula( Variable( avg_1, 0 ) ).
        formula( Variable( avg_2, 0 ) ).
        formula( Variable( avg_3, 0 ) ).
        formula( Variable( avg_4, 0 ) ).
```

```
        formula( Cmd_To_Enum( abor, 1 ) ).
        formula( Cmd_To_Enum( allo, 2 ) ).
        formula( Cmd_To_Enum( help, 3 ) ).
        formula( Cmd_To_Enum( noop, 4 ) ).

        formula( Variable( time, 0 ) ).
        formula( Variable( delta, 0 ) ).

        formula( Variable( cmd, 0 ) ).
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONJECTURES (GOALS)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
list_of_special_formulae(conjectures, DL).

        formula( exists( [z], ConfInfo( z ) ) ).

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end_problem.
```

D.2.4 Knowledge Specification File for Agent Q (AgentQ.dfg)

```
begin_problem(AgentQ).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FILE DESCRIPTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
list_of_descriptions.

    name({* AgentQ.dfg *}).
    author({* Jason Jaskolka *}).
    status(unknown).
    description
    ({*  Initial Knowledge Base for Agent Q.  *}).

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SYMBOL LIST
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

list_of_symbols.

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % OBJECTS
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    functions
    [
        (abor,0),
        (allo,0),
        (help,0),
        (noop,0),
        (00,0),
        (01,0),
        (10,0),
        (11,0),
        (0,0),
```

```
    (1,0),
    (2,0),
    (3,0),
    (4,0),
    (cmd,0),
    (delta,0),
    (last,0),
    (time,0),
    (num_1,0),
    (num_2,0),
    (num_3,0),
    (num_4,0),
    (since_1,0),
    (since_2,0),
    (since_3,0),
    (since_4,0),
    (avg_1,0),
    (avg_2,0),
    (avg_3,0),
    (avg_4,0)
].
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONCEPTS AND ROLES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
predicates
[
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % CONCEPTS
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        (command,0),            (Command,1),
        (bitstring,0),          (BitString,1),
        (enumeration,0),        (Enumeration,1),
        (confinfo,0),           (ConfInfo,1),
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % ROLES
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
            (variable,0),           (Variable,2),
            (cmd_to_enum,0),        (Cmd_To_Enum,2),
            (enum_to_cmd,0),        (Enum_To_Cmd,2)
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ].
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % TRANSLATIONS
    %     Concept/Role Names  : lowercase
    %     Individual Types    : Capitalised
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    translpairs
    [
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % CONCEPTS
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            (command,Command),
            (bitstring,BitString),
            (enumeration,Enumeration),
            (confinfo,ConfInfo),
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % ROLES
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            (variable,Variable),
            (cmd_to_enum,Cmd_To_Enum),
            (enum_to_cmd,Enum_To_Cmd)
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ].
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% KNOWLEDGE BASE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
list_of_special_formulae(axioms, DL).

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % TBox
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % EMPTY
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % ABox
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        formula( Command( abor ) ).
        formula( Command( allo ) ).
        formula( Command( help ) ).
        formula( Command( noop ) ).

        formula( Variable( num_1, 0 ) ).
        formula( Variable( num_2, 0 ) ).
        formula( Variable( num_3, 0 ) ).
        formula( Variable( num_4, 0 ) ).

        formula( Variable( avg_1, 0 ) ).
        formula( Variable( avg_2, 0 ) ).
        formula( Variable( avg_3, 0 ) ).
        formula( Variable( avg_4, 0 ) ).

        formula( Variable( time, 0 ) ).
        formula( Variable( delta, 0 ) ).

        formula( Variable( cmd, 0 ) ).

        formula( Variable( last, 0 ) ).
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%--------------------------------------------------------------------------%
% CONJECTURES (GOALS)
%--------------------------------------------------------------------------%
list_of_special_formulae(conjectures, DL).

        formula( exists( [z], ConfInfo( z ) ) ).

end_of_list.
%--------------------------------------------------------------------------%

end_problem.
```

D.2.5 Knowledge Specification File for Agent R (AgentR.dfg)

```
begin_problem(AgentR).

%--------------------------------------------------------------------------%
% FILE DESCRIPTION
%--------------------------------------------------------------------------%
list_of_descriptions.

    name({* AgentR.dfg *}).
    author({* Jason Jaskolka *}).
    status(unknown).
    description
    ({*  Initial Knowledge Base for Agent R.  *}).

end_of_list.
%--------------------------------------------------------------------------%

%--------------------------------------------------------------------------%
% SYMBOL LIST
%--------------------------------------------------------------------------%
list_of_symbols.

    %----------------------------------------------------------------------%
    % OBJECTS
    %----------------------------------------------------------------------%
```

```
functions
[
    (abor,0),
    (allo,0),
    (help,0),
    (noop,0),
    (00,0),
    (01,0),
    (10,0),
    (11,0),
    (0,0),
    (1,0),
    (2,0),
    (3,0),
    (4,0),
    (cmd,0),
    (delta,0),
    (last,0),
    (time,0),
    (num_1,0),
    (num_2,0),
    (num_3,0),
    (num_4,0),
    (since_1,0),
    (since_2,0),
    (since_3,0),
    (since_4,0),
    (avg_1,0),
    (avg_2,0),
    (avg_3,0),
    (avg_4,0)
].
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONCEPTS AND ROLES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
predicates
[
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % CONCEPTS
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        (command,0),            (Command,1),
        (bitstring,0),          (BitString,1),
        (enumeration,0),        (Enumeration,1),
        (confinfo,0),           (ConfInfo,1),
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % ROLES
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        (variable,0),           (Variable,2),
        (cmd_to_enum,0),        (Cmd_To_Enum,2),
        (enum_to_cmd,0),        (Enum_To_Cmd,2)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
].
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TRANSLATIONS
%     Concept/Role Names  : lowercase
%     Individual Types    : Capitalised
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

translpairs
[
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % CONCEPTS
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        (command,Command),
        (bitstring,BitString),
        (enumeration,Enumeration),
        (confinfo,ConfInfo),
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % ROLES
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        (variable,Variable),
```

```
            (cmd_to_enum,Cmd_To_Enum),
            (enum_to_cmd,Enum_To_Cmd)
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ].
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% KNOWLEDGE BASE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
list_of_special_formulae(axioms, DL).

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % TBox
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Enum_To_Cmd is the inverse mapping of Cmd_To_Enum
        role_formula( equiv( enum_to_cmd, conv( cmd_to_enum ) ) ).
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % ABox
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        formula( Command( abor ) ).
        formula( Command( allo ) ).
        formula( Command( help ) ).
        formula( Command( noop ) ).

        formula( Enumeration( 1 ) ).
        formula( Enumeration( 2 ) ).
        formula( Enumeration( 3 ) ).
        formula( Enumeration( 4 ) ).

        formula( BitString( 00 ) ).
        formula( BitString( 01 ) ).
        formula( BitString( 10 ) ).
```

```
        formula( BitString( 11 ) ).

        formula( Variable( num_1, 0 ) ).
        formula( Variable( num_2, 0 ) ).
        formula( Variable( num_3, 0 ) ).
        formula( Variable( num_4, 0 ) ).

        formula( Variable( since_1, 0 ) ).
        formula( Variable( since_2, 0 ) ).
        formula( Variable( since_3, 0 ) ).
        formula( Variable( since_4, 0 ) ).

        formula( Variable( avg_1, 0 ) ).
        formula( Variable( avg_2, 0 ) ).
        formula( Variable( avg_3, 0 ) ).
        formula( Variable( avg_4, 0 ) ).

        formula( Cmd_To_Enum( abor, 1 ) ).
        formula( Cmd_To_Enum( allo, 2 ) ).
        formula( Cmd_To_Enum( help, 3 ) ).
        formula( Cmd_To_Enum( noop, 4 ) ).

        formula( Variable( time, 0 ) ).
        formula( Variable( delta, 0 ) ).

        formula( Variable( cmd, 0 ) ).
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONJECTURES (GOALS)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
list_of_special_formulae(conjectures, DL).

        formula( exists( [z], ConfInfo( z ) ) ).

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end_problem.
```

D.2.6 Evolved Knowledge Specification File for Agent R (EvolvedAgentR.dfg)

```
begin_problem(EvolvedAgentR).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FILE DESCRIPTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
list_of_descriptions.

    name({* EvolvedAgentR.dfg *}).
    author({* Jason Jaskolka *}).
    status(unknown).
    description
    ({*  Evolved Knowledge Base for Agent R resulting from the manual
         simulation of its amended concrete behaviour.  *}).

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SYMBOL LIST
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
list_of_symbols.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% OBJECTS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
functions
[
    (abor,0),
    (allo,0),
    (help,0),
    (noop,0),
    (00,0),
    (01,0),
    (10,0),
    (11,0),
    (0,0),
    (1,0),
    (2,0),
    (3,0),
    (4,0),
    (cmd,0),
    (delta,0),
    (last,0),
    (time,0),
    (num_1,0),
    (num_2,0),
    (num_3,0),
    (num_4,0),
    (since_1,0),
    (since_2,0),
    (since_3,0),
    (since_4,0),
    (avg_1,0),
    (avg_2,0),
    (avg_3,0),
    (avg_4,0),
    (x,0),
    (y,0)
].
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONCEPTS AND ROLES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
predicates
[
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % CONCEPTS
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        (command,0),            (Command,1),
        (bitstring,0),          (BitString,1),
        (enumeration,0),        (Enumeration,1),
        (confinfo,0),           (ConfInfo,1),
        (confvar,0),            (ConfVar,1),
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % ROLES
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        (variable,0),           (Variable,2),
        (cmd_to_enum,0),        (Cmd_To_Enum,2),
        (enum_to_cmd,0),        (Enum_To_Cmd,2)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
].
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TRANSLATIONS
%     Concept/Role Names  : lowercase
%     Individual Types    : Capitalised
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
translpairs
[
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % CONCEPTS
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        (command,Command),
        (bitstring,BitString),
        (enumeration,Enumeration),
        (confinfo,ConfInfo),
        (confvar,ConfVar),
```

```
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % ROLES
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            (variable,Variable),
            (cmd_to_enum,Cmd_To_Enum),
            (enum_to_cmd,Enum_To_Cmd)
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ].
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% KNOWLEDGE BASE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
list_of_special_formulae(axioms, DL).

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % TBox
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Enum_To_Cmd is the inverse mapping of Cmd_To_Enum
        role_formula( equiv( enum_to_cmd, conv( cmd_to_enum ) ) ).

        % All values in variables storing confidential information represent
        % confidential information
        concept_formula( implies( and ( some ( conv( variable ), confvar ),
            bitstring ), confinfo ) ).
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % ABox
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        formula( Command( abor ) ).
        formula( Command( allo ) ).
        formula( Command( help ) ).
        formula( Command( noop ) ).
```

```
    formula( Enumeration( 1 ) ).
    formula( Enumeration( 2 ) ).
    formula( Enumeration( 3 ) ).
    formula( Enumeration( 4 ) ).

    formula( BitString( 00 ) ).
    formula( BitString( 01 ) ).
    formula( BitString( 10 ) ).
    formula( BitString( 11 ) ).

    formula( Variable( num_1, 0 ) ).
    formula( Variable( num_2, 1 ) ).
    formula( Variable( num_3, 0 ) ).
    formula( Variable( num_4, 0 ) ).

    formula( Variable( since_1, 3 ) ).
    formula( Variable( since_2, 0 ) ).
    formula( Variable( since_3, 3 ) ).
    formula( Variable( since_4, 3 ) ).

    formula( Variable( avg_1, 0 ) ).
    formula( Variable( avg_2, 3 ) ).
    formula( Variable( avg_3, 0 ) ).
    formula( Variable( avg_4, 0 ) ).

    formula( Cmd_To_Enum( abor, 1 ) ).
    formula( Cmd_To_Enum( allo, 2 ) ).
    formula( Cmd_To_Enum( help, 3 ) ).
    formula( Cmd_To_Enum( noop, 4 ) ).

    formula( Variable( time, 3 ) ).
    formula( Variable( delta, 3 ) ).
    formula( Variable( cmd, 2 ) ).

    formula( Variable( x, allo ) ).
    formula( Variable( y, 01 ) ).
    formula( ConfVar(y) ).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONJECTURES (GOALS)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
list_of_special_formulae(conjectures, DL).

        formula( exists( [z], ConfInfo( z ) ) ).

end_of_list.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end_problem.
```

Bibliography

[AA11] S.A. Ahmadzadeh and G.B. Agnew. Covert channels in multiple access protocols. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM'11, pages 404–405, Toronto, ON, Canada, 2011.

[AFW06] K. Alghathbar, C. Farkas, and D. Wijesekera. Securing UML information flow using FlowUML. *Journal of Research and Practice in Information Technology*, 38(1):111–120, February 2006.

[AGM85] C.E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic*, 50(2):510–530, 1985.

[AR80] G.R. Andrews and R.P. Reitman. An axiomatic approach to information flow in programs. *ACM Transactions on Programming Languages and Systems*, 2(1):56–76, January 1980.

[Ber07] H. Berghel. Hiding data, forensics, and anti-forensics. *Communications of the ACM*, 50(4):15–20, April 2007.

[BGC05a] V. Berk, A. Giani, and G. Cybenko. Covert channel detection using process query systems. In *Proceedings of the 2nd Annual Conference for Network Flow Analysis*, FLOCON 2005, September 2005.

[BGC05b] V. Berk, A. Giani, and G. Cybenko. Detection of covert channel encoding in network packet delays. Technical Report TR2005-536, Dartmouth College, Hanover, NH, U.S.A., August 2005.

[Bis02] M. Bishop. *Computer Security: Art and Science*. Addison Wesley, Boston, MA, U.S.A., November 2002.

[BK84] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Control*, 60(1-3):109–137, 1984.

[BL76] D.E. Bell and L.J. LaPadula. Secure computer system: Unified exposition and Multics interpretation. Technical Report ESD-TR-75-306, The MITRE Corporation, March 1976.

[BMNP03] F. Baader, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[BN89] D.F.C. Brewer and M.J. Nash. The Chinese wall security policy. In *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, pages 206–214, May 1989.

[BR05] R. Bidou and F. Raynal. Covert channels, November 2005.

[Bro94] R. Browne. Mode security: An infrastructure for covert channel suppression. In *Proceedings of the 1994 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 39–55, Los Almitos, CA, U.S.A., 1994.

[CBS04] S. Cabuk, C.E. Brodley, and C. Shields. IP covert timing channels: Design and detection. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, CCS '04, pages 178–187. ACM, 2004.

[CDE+03] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. The Maude 2.0 System. In R. Nieuwenhuis, editor, *Rewriting Techniques and Applications*, volume 2706 of *Lecture Notes in Computer Science*, pages 76–87. Springer Berlin/Heidelberg, 2003.

[CDL02] D. Calvanese, G. De Giacomo, and M. Lenzerini. Description logics for information integration. In A.C. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond*, volume 2408 of *Lecture Notes in Computer Science*, pages 41–60. Springer Berlin/Heidelberg, 2002.

[CE82] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In D. Kozen, editor, *Logics of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer Berlin/Heidelberg, 1982.

[CKC05] E. Cole, R.L. Krutz, and J.W. Conley. *Network Security Bible*. Wiley, Indianapolis, IN, U.S.A., 2005.

[CKNZ10] D. Calvanese, E. Kharlamov, W. Nutt, and D. Zheleznyakov. Evolution of DL-Lite knowledge bases. In *Proceedings of the 9th International Semantic Web Conference on The Semantic Web - Volume Part I*, ISWC'10, pages 112–128, 2010.

[Coh81] P.M. Cohn. *Universal Algebra*, volume 6 of *Mathematics and Its Applications*. Springer Netherlands, 1981.

[Com93] Communications Security Establishment Canada. *Canadian Trusted Computer Product Evaluation Criteria (CTCPEC)*. Communications Security Establishment Canada, Ottawa, ON, Canada, 1993.

[Com09] Common Criteria Recognition Arrangement. *Common Criteria for Information Technology Security Evaluation (CC)*. Number CCMB-2009-07. Common Criteria Recognition Arrangement, July 2009.

[CR06] E. Cole and S. Ring. *Insider Threat: Protecting the Enterprise From Sabotage, Spying, and Theft*. Syngress, 2006.

[CS96] R. Cleaveland and S.A. Smolka. Strategic directions in concurrency research. *ACM Computing Surveys*, 28(4):607–625, December 1996.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.

[DD77] D.E. Denning and P.J. Denning. Certification of programs for secure information flow. *Communications of the ACM*, 20(7):504–513, July 1977.

[Den76] D.E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, May 1976.

[Dij75] E.W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM*, 18(8):453–457, August 1975.

[DLN+92] F.M. Donini, M. Lenzerini, D. Nardi, A. Schaerf, and W. Nutt. Adding epistemic operators to concept languages. In *Proceedings of the 3rd Internation Conference on Principles of Knowledge Representation and Reasoning*, KR '92, pages 342–353, 1992.

[DoD85] U.S.A. Department of Defense. *Trusted Computer System Evaluation Criteria (TCSEC)*. Number DoD 5200.28-STD in Defense Department Rainbow Series (Orange Book). Department of Defense/National Computer Security Center, Fort George G. Meade, MD, U.S.A., December 1985.

[DoHS09] U.S.A. Department of Homeland Security. A roadmap for cybersecurity research. Department of Homeland Security Science and Technology Directorate, Washington, DC, U.S.A., November 2009.

[DoTI91] U.K. Department of Trade & Industry. *Information Technology Security Evaluation Criteria (ITSEC), COM(90) 314*. Department of Trade & Industry, London, UK, June 1991.

[DP97] A. Darwiche and J. Pearl. On the logic of iterated belief revision. *Artificial Intelligence*, 89(1-2):1–29, January 1997.

[EH86] E.A. Emerson and J.Y. Halpern. "sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, January 1986.

[FG94] R. Focardi and R. Gorrieri. A classification of security properties for process algebras. *Journal of Computer Security*, 3(1):5–33, November 1994.

[FGM03] R. Focardi, R. Gorrieri, and F. Martinelli. Real-time information flow analysis. *IEEE Journal on Selected Areas in Communications*, 21(1):20–35, 2003.

[FH94] N. Friedman and J.Y. Halpern. A knowledge-based framework for belief change. Part II: Revision and update. In *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, KR '94, pages 190–201, 1994.

[GGLT02] J. Giffin, R. Greenstadt, P. Litwack, and R. Tibbetts. Covert messaging through TCP timestamps. In *Proceedings of the Privacy Enhancing Technologies Workshop*, PET, pages 194–208, April 2002.

[GGT10] A. Grusho, N. Grusho, and E. Timonina. Problems of modeling in the analysis of covert channels. In I. Kotenko and V. Skormin, editors, *Proceedings*

of the 5th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security, volume 6258 of *Lecture Notes in Computer Science*, page 6258. Springer Berlin/Heidelberg, 2010.

[GH99] J.R. Giles and B. Hajek. The jamming game for timing channels. In *Proceedings of the 1999 Information Theory and Networking Workshop*, page 35, Metsovo , Greece, 1999.

[GH02] J. Giles and B. Hajek. An information-theoretic and game-theoretic study of timing channels. *IEEE Transactions on Information Theory*, 48(9):2455–2477, September 2002.

[GKT05] A. Grusho, A. Kniazev, and E. Timonina. Detection of illegal information flow. In V. Gorodetsky, I. Kotenko, and V. Skormin, editors, *Proceedings of the 3rd International Workshop on Mathematical Methods, Models, and Architectures for Computer Networked Security*, volume 3685 of *Lecture Notes in Computer Science*, pages 235–244. Springer Berlin/Heidelberg, 2005.

[GM82] J.A. Goguen and J. Meseguer. Security policies and security models. In *Proceedings of the 1982 Symposium on Security and Privacy*, pages 11–20, New York, NY, U.S.A., 1982.

[Gol11] D. Gollmann. *Computer Security*. John Wiley & Sons, 2011.

[Gra91] J.W. Gray III. Toward a mathematical foundation for information flow security. In *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 21–34, Oakland, CA, U.S.A., 1991.

[GW07] S. Gianvecchio and H. Wang. Detecting covert timing channels: An entropy-based approach. In *Proceedings of the 14th ACM Conference on Computer and*

Communications Security, CCS '07, pages 307–316, New York, NY, U.S.A., 2007. ACM.

[HKMY87] J.T. Haigh, R.A. Kemmerer, J. McHugh, and W.D. Young. An experience using two covert channel analysis techniques on a real system design. *IEEE Transactions on Software Engineering*, SE-13(2):157–168, February 1987.

[HMSW09a] C.A.R. Hoare, B. Möller, G. Struth, and I. Wehrman. Concurrent Kleene algebra. In M. Bravetti and G. Zavattaro, editors, *CONCUR 2009 - Concurrency Theory*, volume 5710 of *Lecture Notes in Computer Science*, pages 399–414. Springer Berlin/Heidelberg, 2009.

[HMSW09b] C.A.R. Hoare, B. Möller, G. Struth, and I. Wehrman. Foundations of concurrent Kleene algebra. In R. Berghammer, A. Jaoua, and B. Möller, editors, *Relations and Kleene Algebra in Computer Science*, volume 5827 of *Lecture Notes in Computer Science*, pages 166–186. Springer Berlin/Heidelberg, 2009.

[HMSW10] C.A.R. Hoare, B. Möller, G. Struth, and I. Wehrman. Concurrent Kleene algebra and its foundations. Technical Report CS-10-04, University of Sheffield, Department of Computer Science, Sheffield, UK, August 2010. Available: http://www.dcs.shef.ac.uk/~georg/ka/.

[HMSW11] C.A.R. Hoare, B. Möller, G. Struth, and I. Wehrman. Concurrent Kleene algebra and its foundations. *Journal of Logic and Algebraic Programming*, 80(6):266–296, 2011.

[Hoa69] C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, October 1969.

[Hoa78a] C.A.R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, August 1978.

[Hoa78b] C.A.R. Hoare. Some properties of predicate transformers. *Journal of the ACM*, 25(3):461–480, July 1978.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[Hol04] W.M.L. Holcombe. *Algebraic Automata Theory*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2004.

[HPRW08] R. Hähnle, J. Pan, P. Rümmer, and D. Walter. Integration of a security type system into a program logic. *Theoretical Computer Science*, 402(2–3):172–189, 2008.

[HR10] L. Hélouët and A. Roumy. Covert channel detection using information theory. In K. Chatzikokolakis and V. Cortier, editors, *Proceedings of 8th International Workshop on Security Issues in Concurrency*, SecCo 2010, pages 34–51, August 2010.

[HRLS06] K. Hristova, T. Rothamel, Y.A. Liu, and S.D. Stoller. Efficient type inference for secure information flow. In *Proceedings of the 2006 Workshop on Programming Languages and Analysis for Security*, PLAS '06, pages 85–94, New York, NY, U.S.A., October 2006.

[HS96] T.G. Handel and M.T. Sandford II. Hiding data in the OSI network model. In *Proceedings of the First International Workshop on Information Hiding*, volume 1174 of *Lecture Notes in Computer Science*, pages 23–38, London, UK, 1996. Springer-Verlag.

[HS00] U. Hustadt and R.A. Schmidt. Issues of decidability for description logics in the framework of resolution. In R. Caferra and G. Salzer, editors, *Automated Deduction in Classical and Non-Classical Logics*, volume 1761 of *Lecture Notes in Computer Science*, pages 191–205. Springer Berlin/Heidelberg, 2000.

[HSG04] U. Hustadt, R.A. Schmidt, and L. Georgieva. A survey of decidable first-order fragments and description logics. *Journal of Relational Methods in Computer Science*, 1:251–276, 2004.

[Hu91] W.-M. Hu. Reducing timing channels with fuzzy time. In *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 8–20, Oakland, CA, U.S.A., May 1991. IEEE Computer Society.

[Hu92] W.-M. Hu. Lattice scheduling and covert channels. In *Proceedings of the 1992 IEEE Symposium on Security and Privacy*, pages 52–61, Oakland, CA, U.S.A., 1992.

[Hun74] T. W. Hungerford. *Algebra*, volume 73 of *Graduate Texts in Mathematics*. Springer-Verlag, 1974.

[HW93] U. Hebisch and H.J. Weinert. *Semirings: Algebraic Theory and Applications in Computer Science*, volume 5 of *Series in Algebra*. World Scientific, 1993.

[HW11] C.A.R. Hoare and J. Wickerson. Unifying models of data flow. In M. Broy, C. Leuxner, and C.A.R. Hoare, editors, *Proceedings of the 2010 Marktoberdorf Summer School on Software and Systems Safety*, pages 211–230. IOS Press, August 2011.

[HZD05] L. Hélouët, M. Zeitoun, and A. Degorre. Scenarios and covert channels: Another game... *Electronic Notes in Theoretical Computer Science*, 119:93–116, 2005.

[HZJ03] L. Hélouët, M. Zeitoun, and C. Jard. Covert channels detection in protocols using scenarios. In *Proceedings of Security Protocols Verification*, SPV'03, pages 21–25, 2003.

[Jac90] J. Jacob. Separability and the detection of hidden channels. *Information Processing Letters*, 34(1):27–29, February 1990.

[Jas10] J. Jaskolka. Modeling, analysis, and detection of information leakage via protocol-based covert channels. Master's thesis, McMaster University, Hamilton, ON, Canada, September 2010.

[JK11a] M.V. Jadhav and S.L. Kattimani. Effective detection mechanism for TCP based hybrid covert channels in secure communication. In *Proceedings of the 2011 International Conference on Emerging Trends in Electrical and Computer Technology*, ICETECT 2011, pages 1123–1128, March 2011.

[JK11b] J. Jaskolka and R. Khedri. Exploring covert channels. In *Proceedings of the 44th Hawaii International Conference on System Sciences*, HICSS-44, pages 1–10, Koloa, Kauai, HI, U.S.A., January 2011.

[JK14a] J. Jaskolka and R. Khedri. A formulation of the potential for communication condition using C^2KA. In A. Peron and C. Piazza, editors, *Proceedings of the 5th International Symposium on Games, Automata, Logics and Formal Verification*, volume 161 of *Electronic Proceedings in Theoretical Computer Science*, pages 161–174, Verona, Italy, September 2014. Open Publishing Association.

[JK14b] J. Jaskolka and R. Khedri. Mitigating covert channels based on analysis of the potential for communication. *Theoretical Computer Science*, (40 pages), (*Submitted*, 2014).

[JKS11] J. Jaskolka, R. Khedri, and K.E. Sabri. A formal test for detecting information leakage via covert channels. In *Proceedings of the 7th Annual Cyber Security and Information Intelligence Research Workshop*, CSIIRW7, pages 1–4, Oak Ridge, TN, U.S.A., October 2011.

[JKS14] J. Jaskolka, R. Khedri, and K.E. Sabri. Investigative support for information confidentiality part I: Detecting confidential information leakage via protocol-based covert channels. In *Proceedings of the 9th International Conference on Future Networks and Communications*, volume 34 of *Procedia Computer Science, FNC 2014 and MobiSPC 2014*, pages 276–285, Niagara Falls, ON, Canada, August 2014.

[JKZ12] J. Jaskolka, R. Khedri, and Q. Zhang. On the necessary conditions for covert channel existence: A state-of-the-art survey. *Procedia Computer Science*, 10:458–465, August 2012. Proceedings of the 3rd International Conference on Ambient Systems, Networks and Technologies, ANT 2012.

[JKZ13] J. Jaskolka, R. Khedri, and Q. Zhang. Foundations of communicating concurrent Kleene algebra. Technical Report CAS-13-07-RK, McMaster University, Hamilton, ON, Canada, November 2013. Available: `http://www.cas.mcmaster.ca/cas/0template1.php?601`.

[JKZ14] J. Jaskolka, R. Khedri, and Q. Zhang. Endowing concurrent Kleene algebra with communication actions. In P. Höfner, P. Jipsen, W. Kahl, and M.E. Müller, editors, *Proceedings of the 14th International Conference on Relational and Algebraic Methods in Computer Science*, volume 8428 of *Lecture Notes in Computer Science*, pages 19–36. Springer International Publishing Switzerland, 2014.

[JLY10] D. Johnson, P. Lutz, and B. Yuan. Behavior-based covert channel in cyberspace. In *Proceedings of the 4th International ISKE Conference on Intelligent Decision Making Systems*, pages 311–318, 2010.

[Jon81] C.B. Jones. *Development Methods for Computer Programs Including a Notion*

of Interference. PhD thesis, Oxford University, June 1981. Programming Research Group, Technical Monograph PRG-25.

[JSS07] T. Jaeger, R. Sailer, and Y. Sreenivasan. Managing the risk of covert information flows in virtual machine systems. In *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies*, SACMAT '07, pages 81–90, Sophia Antipolis, France, 2007.

[Kas14] Kaspersky Lab. IT security risks survey 2014: A business approach to managing data security threats. Kaspersky Lab, 2014.

[KC09] N. Kiyavash and T. Coleman. Covert timing channels codes for communication over interactive traffic. In *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP 2009, pages 1485–1488, Piscataway, NJ, U.S.A., 2009.

[Kel76] R.M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 19(7):371–384, July 1976.

[Kem83] R.A. Kemmerer. Shared resource matrix methodology: An approach to identifying storage and timing channels. *ACM Transactions on Computer Systems*, 1(3):256–277, August 1983.

[KKM00] M. Kilp, U. Knauer, and A.V. Mikhalev. *Monoids, Acts and Categories with Applications to Wreath Products and Graphs: A Handbook for Students and Researchers*, volume 29 of *De Gruyter Expositions in Mathematics Series*. Walter de Gruyter, 2000.

[KL51] S. Kullback and R.A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[KM93] M.H. Kang and I.S. Moskowitz. A pump for rapid, reliable, secure communication. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 119–129, Fairfax, VA, U.S.A., 1993.

[Kob05] N. Kobayashi. Type-based information flow analysis for the π-calculus. *Acta Informatica*, 42(4):291–347, 2005.

[Koz93] Dexter Kozen. On action algebras. In *Logic and Information Flow*, pages 78–88. MIT Press, 1993.

[Koz97] D. Kozen. *Automata and Computability*. Undergraduate Texts in Computer Science. Springer, 1997.

[KP91] R.A. Kemmerer and P.A. Porras. Covert flow trees: A visual approach to analyzing covert storage channels. *IEEE Transactions on Software Engineering*, 17(11):1166–1185, November 1991.

[Kri63] S. Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16:83–94, 1963.

[KS07] J. Kohlas and R. Stärk. Information algebras and consequence operators. *Logica Universalis*, 1(1):139–165, January 2007.

[Lam73] B.W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, October 1973.

[LG98] A.A. Letichevsky and D. Gilbert. A general theory of action languages. *Cybernetics and Systems Analysis*, 34:12–30, 1998.

[LHD10] X. Liu, J. Hao, and Y. Dai. An approach to analyze covert channel based on finite state machine. In *Proceedings of the 2nd International Conference on Multimedia Information Networking and Security*, MINES 2010, pages 438–442, Los Alamitos, CA, U.S.A., 2010.

[LMST+04] R. Lanotte, A. Maggiolo-Schettini, S. Tini, A. Troina, and E. Tronci. Automatic covert channel analysis of a multilevel secure component. In J. Lopez, S. Qing, and E. Okamoto, editors, *Proceedings of the 6th International Conference on Information and Communications Security*, volume 3269 of *Lecture Notes in Computer Science*, pages 249–261. Springer Berlin/Heidelberg, 2004.

[Lon14] H.W. Longfellow. Paul Revere's Ride. In C.W. Eliot, editor, *English Poetry III: From Tennyson to Whitman*, volume 42 of *The Harvard Classics*. P.F. Collier & Son, 1909–1914.

[Low02] G. Lowe. Quantifying information flow. In *Proceedings of the 15th IEEE Computer Security Foundations Workshop*, CSFW-15, pages 18–31, Los Alamitos, CA, U.S.A., 2002. IEEE Computer Society.

[LPRR02] S.A. Linton, G. Pfeiffer, E.F. Robertson, and N. Ruškuc. Computing transformation semigroups. *Journal of Symbolic Computation*, 33(2):145–162, 2002.

[LSWH09] J. Luo, Z. Shi, M. Wang, and H. Huang. Multi-agent cooperation: A description logic view. In D. Lukose and Z. Shi, editors, *Multi-Agent Systems for Society*, volume 4078 of *Lecture Notes in Computer Science*, pages 365–379. Springer Berlin/Heidelberg, 2009.

[LT97] J. Lobo and G. Trajcevski. Minimal and consistent evolution of knowledge bases. *Journal of Applied Non-Classical Logics*, 7(1-2):117–146, 1997.

[Maz87] A. Mazurkiewicz. Trace theory. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *Lecture Notes in Computer Science*, pages 279–324. Springer Berlin/Heidelberg, 1987.

[McH95] J. McHugh. *Handbook for the Computer Security Certification of Trusted*

Systems, chapter 8: Covert Channel Analysis. Naval Research Laboratory, Washington, DC, U.S.A., October 1995.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[Mil87] J.K. Millen. Covert channel capacity. In *Proceedings of the 1987 Symposium on Security and Privacy*, pages 60–66, Oakland, CA, U.S.A., 1987. IEEE Computer Society.

[Mil89a] J.K. Millen. Finite-state noiseless covert channels. In *Proceedings of the Computer Security Foundations Workshop II*, pages 81–86, Washington, DC, U.S.A., 1989. IEEE Computer Society.

[Mil89b] R. Milner. *Communication and Concurrency*. Prentice-Hall International Series in Computer Science. Prentice Hall, 1989.

[Mil90] J.K. Millen. Hookup security for synchronous machines. In *Proceedings of the Computer Security Foundations Workshop III*, pages 84–90, June 1990.

[Mil99] J. Millen. 20 years of covert channel modeling and analysis. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, pages 113–114, Los Alamitos, CA, U.S.A., 1999.

[MK94] I.S. Moskowitz and M.H. Kang. Covert channels — here to stay? In *Computer Assurance*, COMPASS '94 Safety, Reliability, Fault Tolerance, Concurrency and Real Time, Security, pages 235–243, Gaithersburg, MD , U.S.A., June 1994. IEEE Computer Society.

[ML10] T. Murray and G. Lowe. Analysing the information flow properties of object-capability patterns. In P. Degano and J. Guttman, editors, *Formal Aspects in*

Security and Trust, volume 5983 of *Lecture Notes in Computer Science*, pages 81–95. Springer Berlin/Heidelberg, 2010.

[MMA06] K. Martin, I.S. Moskowitz, and G. Allwein. Algebraic information theory for binary channels. *Electronic Notes in Theoretical Computer Science*, 158:289–306, 2006.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes part I. *Information and Computation*, 100(1):1–40, September 1992.

[MQ91] J.A. McDermid and S. Qi. A formal model of security dependency for analysis and testing of secure systems. In *Proceedings of the Computer Security Foundations Workshop IV*, pages 188–200, Los Alamitos, CA, U.S.A., 1991.

[MSM91] P.M. Melliar-Smith and L.E. Moser. Protection against covert storage and timing channels. In *Proceedings of the 4th IEEE Computer Security Foundations Workshop*, CSFW '91, pages 209–214, Franconia, NH, U.S.A., June 1991. IEEE Computer Society.

[Mur07] S.J. Murdoch. Covert channel vulnerabilities in anonymity systems. Technical Report UCAM-CL-TR-706, University of Cambridge, Cambridge, UK, December 2007.

[MvdH04] J.-J. Ch. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Science*, volume 41 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2004.

[NCSC93] U.S.A. National Computer Security Center. *A Guide to Understanding Covert Channel Analysis of Trusted Systems*. Number NCSC-TG-030 in NSA/NCSC Rainbow Series (Light Pink Book). National Security Agency/National Computer Security Center, Fort George G. Meade, MD, U.S.A., November 1993.

[NW06] N. Nagatou and T. Watanabe. Run-time detection of covert channels. In *Proceedings of the 1st International Conference on Availability, Reliability and Security*, ARES 2006, pages 577–584, Vienna, Austria, 2006. IEEE Computer Society.

[OOS+97] N. Ogurtsov, H. Orman, R. Schroeppel, S. O'Malley, and O. Spatscheck. Experimental results of covert channel limitation in one-way communication systems. In *Proceedings of the 1997 Symposium on Network and Distributed System Security*, pages 2–15, Los Alamitos, CA, U.S.A., 1997. IEEE Computer Society.

[PAK99] F.A.P. Petitcolas, R.J. Anderson, and M.G. Kuhn. Information hiding — a survey. *Proceedings of the IEEE, Special Issue on Protection of Multimedia Content*, 87(7):1062–1078, July 1999.

[Pet62] C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für instrumentelle Mathematik, Bonn, Germany, 1962. English translation available as: *Communication with Automata*, Technical Report RADC-TR-65-377, volume 1, supplement 1, Applied Data Research, Princeton, NJ, U.S.A., 1966.

[PHGB10] C.W. Probst, J. Hunker, D. Gollmann, and M. Bishop. *Insider Threats In Cyber Security*, volume 49 of *Advances In Information Security*. Springer, 2010.

[PK91] P.A. Porras and R.A. Kemmerer. Covert flow trees: A technique for identifying and analyzing covert storage channels. In *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 36–51, Los Alamitos, CA, U.S.A., 1991. IEEE Computer Society.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57, 1977.

[Pon14] Ponemon Institute. 2014 cost of data breach study: Global analysis. Ponemon Institute Research Report, May 2014.

[PP05] E. Pacuit and R. Parikh. The logic of communication graphs. In J. Leite, A. Omicini, P. Torroni, and P. Yolum, editors, *Proceedings of the 2nd International Workshop on Declarative Agent Languages and Technologies II*, volume 3476 of *Lecture Notes in Computer Science*, pages 256–269. Springer Berlin/Heidelberg, 2005.

[PP07] E. Pacuit and R. Parikh. Reasoning about communication graphs. In J. van Benthem, B. Löwe, and D. Gabbay, editors, *Interactive Logic: Games and Social Software*, volume 1, 2007.

[Pra86] V. Pratt. Modeling concurrency with partial orders. *International Journal of Parallel Programming*, 15(1):33–71, February 1986.

[Pra91] V. Pratt. Action logic and pure induction. In J. Eijck, editor, *Logics in AI*, volume 478 of *Lecture Notes in Computer Science*, pages 97–120. Springer Berlin/Heidelberg, 1991.

[PSCS07] A. Patel, M. Shah, R. Chandramouli, and K.P. Subbalakshmi. Covert channel forensics on the internet: Issues, approaches, and experiences. *International Journal of Network Security*, 5(1):41–50, July 2007.

[RMMG01] P. Ryan, J. McLean, J. Millen, and V. Gligor. Non-interference: Who needs it? In *Proceedings of the 14th IEEE Workshop on Computer Security Foundation*, pages 237–238, Washington, DC, U.S.A., 2001. IEEE Computer Society.

[Rus82] J. Rushby. Proof of separability: A verification technique for a class of security kernels. In M. Dezani-Ciancaglini and U. Montanari, editors, *International*

Symposium on Programming, volume 137 of *Lecture Notes in Computer Science*, pages 352–367. Springer Berlin/Heidelberg, 1982.

[SAIL07] A. Shaffer, M. Auguston, C. Irvine, and T. Levin. Toward a security domain model for static analysis and verification of information systems. In *Proceedings of the 7th OOPSLA Workshop on Domain-Specific Modeling*, DSM '07, pages 160–171, Montreal, QC, Canada, October 2007.

[SAIL08] A.B. Shaffer, M. Auguston, C.E. Irvine, and T.E. Levin. A security domain model to assess software for exploitable covert channels. In *Proceedings of the ACM SIGPLAN 3rd Workshop on Programming Languages and Analysis for Security*, PLAS '08, pages 45–56, Tucson, AZ, U.S.A., 2008.

[Sal09] M. Salaün. Practical overview of a Xen covert channel. *Journal in Computer Virology*, 6(4):317–328, 2009.

[Sar06] B. Sartin. Anti-forensics — distorting the evidence. *Computer Fraud and Security*, 2006(5):4–6, May 2006.

[SC99] S. Shieh and A.L.P. Chen. Estimating and measuring covert channel bandwidth in multilevel secure operating systems. *Journal of Information Science and Engineering*, 15(1):91–106, 1999.

[Sco07] C. Scott. Network covert channels: Review of current state and analysis of viability of the use of X.509 certificates for covert communications. Technical Report RHUL-MA-2008-11, Royal Holloway, University of London, January 2007.

[SGG07] A. Silberschatz, P.B. Galvin, and G. Gagne. *Operating System Concepts*. Wiley, seventh edition, 2007.

[Sha48] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948.

[Sid03] K.A. Siddiqi. Covert channels over TCP/IP and protocol steganography. Technical Report 2003-03-0044, Lahore University of Management Sciences, 2003.

[Sim84] G.J. Simmons. The prisoners' problem and the subliminal channel. In *Advances in Cryptology*, CRYPTO '83, pages 51–67, New York, NY, U.S.A., 1984.

[Sim85] G.J. Simmons. The subliminal channel and digital signatures. In T. Beth, N. Cot, and I. Ingemarsson, editors, *Advances in Cryptology*, volume 209 of *Lecture Notes in Computer Science*, pages 364–378. Springer Berlin/Heidelberg, 1985.

[SK06] M. Smeets and M. Koot. Research report: Covert channels. Master's thesis, University of Amsterdam, Amsterdam, Netherlands, February 2006.

[SKJ09a] K.E. Sabri, R. Khedri, and J. Jaskolka. Automated verification of information flow in agent-based systems. Technical Report CAS-09-01-RK, McMaster University, Hamilton, ON, Canada, January 2009.

[SKJ09b] K.E. Sabri, R. Khedri, and J. Jaskolka. Verification of information flow in agent-based systems. In G. Babin, P. Kropf, and M. Weiss, editors, *Proceedings of the 4th International MCETECH Conference on e-Technologies*, volume 26 of *Lecture Notes in Business Information Processing*, pages 252–266. Springer Berlin/Heidelberg, May 2009.

[SQ07] J. Shen and S. Qing. A dynamic information flow model of secure systems. In *Proceedings of the 2nd ACM Symposium on Information, Computer and*

Communications Security, ASIACCS '07, pages 341–343, Singapore, 2007. ACM.

[SSM03] T. Sohn, J. Seo, and J. Moon. A study on the covert channel detection of TCP/IP header using support vector machine. In S. Qing, D. Gollmann, and J. Zhou, editors, *Information and Communications Security*, volume 2836 of *Lecture Notes in Computer Science*, pages 313–324. Springer Berlin/Heidelberg, 2003.

[Ste10] B. Steinberg. A theory of transformation monoids: Combinatorics and representation theory. *The Electronic Journal of Combinatorics*, 17(1), 2010.

[SWBS09] S.H. Sellke, C.-C. Wang, S. Bagchi, and N. Shroff. TCP/IP timing channels: Theory to implementation. In *Proceedings of the 28th IEEE Conference on Computer Communications*, INFOCOM 2009, pages 2204–2212, April 2009.

[TA05a] E. Tumoian and M. Anikeev. Detecting NUSHU covert channels using neural networks. Technical report, Taganrog State University of Radio Engineering, 2005.

[TA05b] E. Tumoian and M. Anikeev. Network based detection of passive covert channels in TCP/IP. In *Proceedings of the 30th IEEE Conference on Local Computer Networks*, pages 802–807, Sydney, Australia, 2005.

[TG88] C.-R. Tsai and V.D. Gligor. A bandwidth computation model for covert storage channels and its applications. In *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, pages 108–121, Washington, DC, U.S.A., 1988.

[TGC87] C.-R. Tsai, V.D. Gligor, and C.S. Chandersekaran. A formal method for the identification of covert storage channels in source code. *IEEE Symposium on Security and Privacy*, page 74, 1987.

[TGC90] C.-R. Tsai, V.D. Gligor, and C.S. Chandersekaran. On the identification of covert storage channels in secure systems. *IEEE Transactions on Software Engineering*, 16(6):569–580, June 1990.

[TK08] H.F. Tipton and M. Krause. *Information Security Management Handbook*, volume 2. Auerbach, 2008.

[TO95] E. Teniente and A. Olivé. Updating knowledge bases while maintaining their consistency. *The VLDB Journal*, 4(2):193–241, April 1995.

[Tro93] J.T. Trostle. Modelling a fuzzy time system. In *Proceedings of the 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 82–89, Los Alamitos, CA, U.S.A., 1993.

[TST14] The SPASS Team. Spass: An automated theorem prover for first-order logic with equality. Available: `http://www.spass-prover.org/index.html` (Accessed: May 29, 2014), May 2014.

[Var90] V. Varadharajan. Petri net based modelling of information flow security requirements. In *In Proceedings of the Computer Security Foundations Workshop III*, pages 51–61, June 1990.

[vDvdHK03] H.P. van Ditmarsch, W. van der Hoek, and B.P. Kooi. Concurrent dynamic epistemic logic. Technical Report OUCS-2003-01, University of Otago, 2003.

[vDvdHK07] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*, volume 337 of *Synthese Library*. Springer, 2007.

[VIS96] D. Volpano, C. Irvine, and G. Smith. A sound type system for secure flow analysis. *Journal of Computer Security*, 4(2-3):167–187, 1996.

[VS97] D. Volpano and G. Smith. Eliminating covert flows with minimum typings. In *Proceedings of the 10th Computer Security Foundations Workshop*, pages 156–168, Los Alamitos, CA, U.S.A., 1997.

[Wag05] D. Wagner. Re: Suggestions for the passing of passphrases. Available: `http://www.derkeiler.com/Newsgroups/sci.crypt/2005-06/0622.html` (Accessed: October 23, 2014), June 2005.

[Wan11] Z. Wang. *Ontology Evolution in Description Logics*. PhD thesis, Griffith University, South East Queensland, Australia, January 2011.

[Wat30] J.B. Watson. *Behaviorism*. University of Chicago Press, 1930.

[Win87] G. Winskel. Event structures. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *Lecture Notes in Computer Science*, pages 325–392. Springer Berlin/Heidelberg, 1987.

[WJ06] C.-D. Wang and S. Ju. The dilemma of covert channels searching. In D. Won and S. Kim, editors, *Information Security and Cryptology*, volume 3935 of *Lecture Notes in Computer Science*, pages 169–174. Springer Berlin/Heidelberg, 2006.

[WJG+04] C.-D. Wang, S. Ju, D. Guo, Z. Yang, and W.-Y. Zheng. Research on the methods of search and elimination in covert channels. In M. Li, X.-H. Sun, Q.-N. Deng, and J. Ni, editors, *Grid and Cooperative Computing*, volume 3032 of *Lecture Notes in Computer Science*, pages 988–991. Springer Berlin/Heidelberg, 2004.

[WL05a] Z. Wang and R.B. Lee. Capacity estimation of non-synchronous covert channels. In *Proceedings of the 25th IEEE International Conference on Distributed*

Computing Systems Workshops, pages 170–176. IEEE Computer Society, June 2005.

[WL05b] Z. Wang and R.B. Lee. New constructive approach to covert channel modeling and channel capacity estimation. In J. Zhou, J. Lopez, R.H. Deng, and F. Bao, editors, *Proceedings of 8th International Conference on Information Security*, volume 3650 of *Lecture Notes in Computer Science*, pages 498–505. Springer Berlin/Heidelberg, 2005.

[Wra91] J.C. Wray. An analysis of covert timing channels. In *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 2–7, Los Alamitos, CA, U.S.A., 1991.

[WWTP10] Z. Wang, K. Wang, R. Topor, and J.Z. Pan. Forgetting for knowledge bases in DL-Lite. *Annals of Mathematics and Artificial Intelligence*, 58(1-2):117–151, 2010.

[ZAB07a] S. Zander, G. Armitage, and P. Branch. Covert channels and countermeasures in computer network protocols. *IEEE Communications Magazine*, 45(12):136–142, December 2007.

[ZAB07b] S. Zander, G. Armitage, and P. Branch. A survey of covert channels and countermeasures in computer network protocols. *IEEE Communications Surveys Tutorials*, 9(3):44–57, 2007.

[ZLD10] J. Zhai, G. Liu, and Y. Dai. A covert channel detection algorithm based on TCP markov model. In *Proceedings of the 2nd International Conference on Multimedia Information Networking and Security*, MINES 2010, pages 893–897, Los Alamitos, CA, U.S.A., 2010.

[ZLSN05] X. Zou, Q. Li, S. Sun, and X. Niu. The research on information hiding based

on command sequence of FTP protocol. In R. Khosla, R. Howlett, and L. Jain, editors, *Proceedings of the 9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems*, volume 3683 of *Lecture Notes in Computer Science*, pages 1079–1085. Springer Berlin/Heidelberg, 2005.

[ZS10] H. Zhao and Y.Q. Shi. A phase-space reconstruction approach to detect covert channels in TCP/IP protocols. In *Proceedings of the 2010 IEEE International Workshop on Information Forensics and Security*, WIFS 2010, pages 1–6, Piscataway, NJ, U.S.A., 2010.

[ZZ09] Y. Zhang and Y. Zhou. Knowledge forgetting: Properties and applications. *Artificial Intelligence*, 173(16–17):1525–1537, 2009.

Index