PROOFS OF RELATIONAL SEMIGROUPOIDS IN ISABELLE/ISAR

PROOFS OF RELATIONAL SEMIGROUPOIDS IN ISABELLE/ISAR

By

JINRONG HAN, M.Eng., B.Sc

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirement

for the Degree

Master of Science

McMaster University

MASTER OF SCIENCE (2008)
(Computer Science)

McMaster University
Hamilton, Ontario

TITLE: Proofs of Relational Semigroupoids in Isabelle/Isar

AUTHOR: Jinrong Han, M.Eng. (Xidian University), B.Sc. (Zhengzhou University)

SUPERVISOR: Dr. Wolfram Kahl

NUMBER OF PAGES: xi, 184

Abstract

The concept of relations is useful for applications in mathematics, logics and computer science. Once an application structure is identified as a model of a particular relation-algebraic theory, that theory becomes the preferred reasoning environment in this application area. Examples of applications in computer science are database, graph and games.

In [Kah03], Kahl proposed using the proof assistant Isabelle/Isar to provide a collection of theories for abstract relation-algebraic reasoning. In [DG04], De Guzman improved and populated the theories introduced by Kahl in [Kah03]. Finite maps or finite relations between infinite sets do not form a category since the necessary identities are infinite. In [Kah08], Kahl presented relation-algebraic extensions of semigroupoids where the operations that would produce infinite results in category have been replaced with their variants that preserve finiteness, but still satisfy useful algebraic laws.

In this thesis, we will build a framework by building a hierarchy of Isabelle/Isar theories to implement relational semigroupoid theories which are presented by Kahl in [Kah08], focusing on the following:

Since the difference between semigroupoids and categories are that no identities are assumed in semigroupoids, category theories in [DG04] will be transferred into our semigroupoid theories by modifying definitions, reformulating theorems, adding theorems to help reprove theorems involving identities in their proofs.

New theorems and new theories will be added to implement subidentity and range and their properties. Then new theorems and new theories about restricted residual and standard residual and their properties will be developed. In [Kah08], Kahl proposed that in ordered semigroupoids with domain and range, if standard residuals exist, then restricted residuals exist too and can be calculated via standard residuals. A new theory will be built to prove this.

Acknowledgements

I would first like to thank my supervisor Dr.Wolfram Kahl for his valuable help, guidance, and constructive criticism in the creation of this thesis.

Thanks are also due to my defense committee, Dr. Alan Wassyng and Dr. Ryszard Janicki. Their comments helped me clarify some ideas.

Finally, I am grateful to my parents and my husband for their love and encouragement.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The concept of relation algebra is useful for applications in mathematics, logic and computer science. Example applications in computer science are program semantics [BKS97] [SS93], graph [SS93] [Kah01], databases [BKS97] [SS93], and fuzzy relations [Win01]. Once an application structure is identified as a model of a particular relation-algebraic theory, that theory will become the preferred reasoning environment in this application area.

Finite maps or finite relations are used frequently in computing science, especially in programming languages. Surprisingly, the mathematical foundations for dealing with finite relations are not well-established. In section 1.3 of [Kah08], Kahl identified three problems with relation-algebraic treatment of finite relations:

- No complement (negation): this is not a big problem, but perhaps can be worked around by using difference.
- No identities: many relational properties are normally defined using identities.
- No residuals: standard residuals do not exist.

Finite maps or finite relations between infinite sets do not form a category, since the necessary identities are not finite. Identities are not assumed in semigroupoids. In [Kah08], Kahl proposed relation-algebraic extensions of semigroupoids where the operations that would produce infinite results in category have been replaced with their variants that preserved finiteness, but still satisfy useful algebraic laws. The resulting theories allow calculational reasoning in the relation-algebraic style with only minor sacrifices. Kahl introduced the concept of restricted residuals in semigroupoids since standard residuals do not generally exist in the semigroupoids of finite relations between arbitrary sets.

In this thesis, we will develop the framework of the interesting semigroupoids and other related weaker theories presented in [Kah08], especially restricted residual and

standard residual and their properties. In this framework, we will systematically organize algebraic structures and providing readable formal proofs both for machines and humans, using 2008 Isabelle/Isar.

1.2 Related work

Since relation-algebraic reasoning typically follows a very calculational style, and, due to the expressive power of its constructs and rules, also proceeds in relatively formal steps, computer support for this kind of reasoning appears to be quite feasible. In [Kah03], Kahl proposed using the proof assistant Isabelle/Isar to provide a collection of theories for abstract relation-algebraic reasoning. In [DG04], De Guzman improved and populated the theories introduced by Kahl in [Kah03], from categories via allegories up to heterogeneous relation algebras using Isabelle/Isar.

The difference of semigroupoids and categories is that no identities are assumed in semigroupoids. Basic category theories in appendix B of De Guzman's thesis [DG04] can be transformed into our semigroupoid system by modifying definitions, reformulating theorems, deleting theorems which are closely related to identities and adding new theorems to help reprove many theorems involving identities in their proofs, in order to adapt them to our system.

1.3 Our approach

We first provide a framework for abstract weaker semigroupoids. Then we define ordered semigroupoids with range and domain operators and also give their properties. And finally we provide some applications based on it. Specifically, we define restricted residuals and standard residuals and give their properties.

This thesis is organized as follows:

- In chapter 2, our chosen theorem prover Isabelle/Isar is discussed.
- Chapter 3 gives a quick look at how the theories are organized as well as their dependencies.
- Chapter 4 focuses on explaining in detail the bottom theories of the hierarchy and presents the important decisions made in implementing these theories

in Isabelle/Isar. Subidentities are also defined and discussed in this chapter, because many theorems need to be proved via subidentity laws.

- Chapter 5 introduces theories involving domain and range. Range theories are new theories which are dual theories of domain theories.

- Chapter 6 presents the new theories of our hierarchy and how we implemented them by using subidentity laws and the properties of range and domain. These new theories involve standard residuals and restricted residuals and their properties. The theorem that in ordered semigroupoids with domain and range, if standard residual exists, restricted residual exists too and it can be calculated via standard residual, will also be proved in the chapter.

- Chapter 7 covers possible future work of the system and conclusion.

Some theories, axioms and lemmas are referred in the above chapters without showing their implementations in Isabelle/Isar. We provide appendix B for the interested readers for further reading.

Chapter 2

Isabelle/Isar

Isabelle/Isar were adopted to provide computer-aided proof assistance for research and abstract relation-algebra reasoning by Kahl in [Kah03] and by De Guzman in [DG04] in the past. In order to make our research and reasoning persistent in Isabelle/Isar and make further applications based on them easily implemented, we continue to use Isabelle/Isar for our proposed objectives.

In this chapter, we briefly illustrate Isabelle/Isar proving system. For other provers, there are some discussions about IMPS, PVS in the second chapter of [DG04].

2.1 Isabelle/HOL

Isabelle [NPW08] is a generic system for implementing logical formalisms. It allows mathematical formulas to be expressed in a formal language and provides tools for proving those formulas in a logical calculus. Isabelle is developed at University of Cambridge (Larry Paulson) and Technische Universität München.

Isabelle can be viewed from two main perspectives. On the one hand it may serve as a generic framework for rapid prototyping of deductive systems. On the other hand, major existing logics like Isabelle/HOL provide a theorem proving environment ready to use for sizable applications. Isabelle/HOL [NPW08] is the specialization of Isabelle for HOL, which abbreviates Higher-Order Logic.

2.2 Isabelle/Record

A record [NPW08] of Isabelle/HOL covers a collection of fields. Each field has a specified type, which may be polymorphic. The field names are part of the record type, and the order of the fields is significant. Every record structure has an implicit pseudo-field, *more*. When a fixed record value is expressed using just its standard fields, the value of *more* is implicitly set to (), the empty tuple.

Each record declaration introduces a number of derived operations to refer collectively to a record's fields and to convert between fixed record types.

We are interested in the extensibility of records, which allow us build a new record by extending existing one.

2.3 Isar Overview

Isar stands for Intelligible semi-automated reasoning. The Isar subsystem [NPW08] is an extension of Isabelle. It hides the implementation language almost completely. Isar proofs [Nip07] are an extension of the apply-style proofs – tactic-style reasoning in the Isabelle/HOL. Isar supports a calculational style of reasoning and allows us to provide structured proofs which are presented like mathematical proofs and are understandable for both humans and machines.

By integrating Isabelle/Isar with Proof General – a generic (X)Emacs interface for interactive proof assistants, we arrive at a reasonable environment for *live proof document editing*. For presentation of the final outcome, Isabelle/Isar provides an integrated document preparation system [Wen08] based on current PDF/LaTeX hypertext technology. Thus Isabelle/Isar proof documents may be both browsed on the WWW, and printed on paper in high quality.

2.3.1 A glimpse of Isar

An Isar proof can be either compound (proof – qed) or atomic (by). This is a typical proof skeleton [Nip07]:

```
proof
  assume "the-assm"
  have ". . . "          — intermediate result
   ⋮
  have ". . . "          — intermediate result
  show "the-concl"
```

qed

The following are main commands of Isar proof language [Wen08] [DG04]:

- Primitive commands: *lemma*, *proof*, *fix*, *then*, *have*, *show*, *qed*, *note*.
- Derived commands: *also*, *finally*, *moreover*, *ultimately*, *with*, *hence*, *thus*, etc.
- Automatic commands: *simp*, *auto*, *best*, *etc.*
- Other commands: *this*, *rule*, *intro*, *elim*, *unfold*, etc.

Derived proof commands are used for calculational style of reasoning. Commands such as *assumption, rule, intro*, and *unfold* are often used to solve or simplify the current goal by using existing facts, rules, theorems or definitions.

The above automatic commands are very useful in our implementations. *simp* uses the simplifier which applies theorems with *simp* attribute automatically. *auto* uses the simplifier and the classical reasoning, and *best* uses standard Isabelle inference and best-first search. Isabelle allow users to tell these reasoners to add or delete specific rules. Theorems with *simp* attribute or *intro* attribute will be applied automatically. These automatic reasoners help users prove theorems easier and make proofs shorter.

For other commands, how they are used is referred to [NPW08] and [Nip07].

2.3.2 Isar locales

Locale [KWP99] [Bal07], an extension of the Isabelle proof assistant, aims to support modular reasoning. Locales are based on contexts. The logic view of a context can be seen as a formula schema

$$\bigwedge x_1 \ldots x_n.\, [A_1; \ldots; A_m] \Rightarrow \cdots$$

where variables$x_1 \ldots x_n$ are called parameters,the premises$A_1; \ldots; A_m$ are assumptions. A formula $\mathbf{F}$ is a theorem in the context if it is a conclusion

$$\bigwedge x_1 \ldots x_n.\, [A_1; \ldots; A_m] \Rightarrow \mathbf{F}.$$

Isabelle/Isar's notion of context [Bal07] goes beyond this logical view. Its context record is in a consecutive order. Contexts also contain syntax information for parameters and for terms depending on them.

Locales can be seen as persistent contexts. In a simplest form, a locale declaration [Bal07] consists of a sequence of context elements declaring parameters (keyword fixes) and assumptions (keyword assumes). One aspect of locales which we are interested in are local expressions. Locale expressions provide an effective way of constructing complex specifications from simple ones. A new locale [Bal07] can be achieved through *import* existing locales. That is, a locale can be defined by adding operations and properties to existing locales. Algebraic structures are commonly defined in this way. Hence, a locale hierarchy can be easily obtained in a specific application.

2.4 Isabelle Theory

A theory [NPW08] in Isabelle is a collection of types, functions, and theorems, much like a module in a programming language. The general format [NPW08] of a theory T is:

theory T
imports $B_1 \cdots B_n$
begin
declarations, definitions, and proofs
end

where:

- $B_1 \cdots B_n$ are the names of existing theories that T is based on. $B_1 \cdots B_n$ are the direct **parent theories** of T. Everything defined in the parent theories (and their parents, recursively) is automatically visible. To avoid name clashes, identifiers can be qualified by theory names as in $T.f$ and $B.f$. Each theory T must reside in a theory file named T.thy.
- *declarations* and *definitions* represent the newly introduced concepts (types, functions, etc).
- *proofs* are proofs about the newly introduced concepts.

Chapter 3

Theory Organization Overview

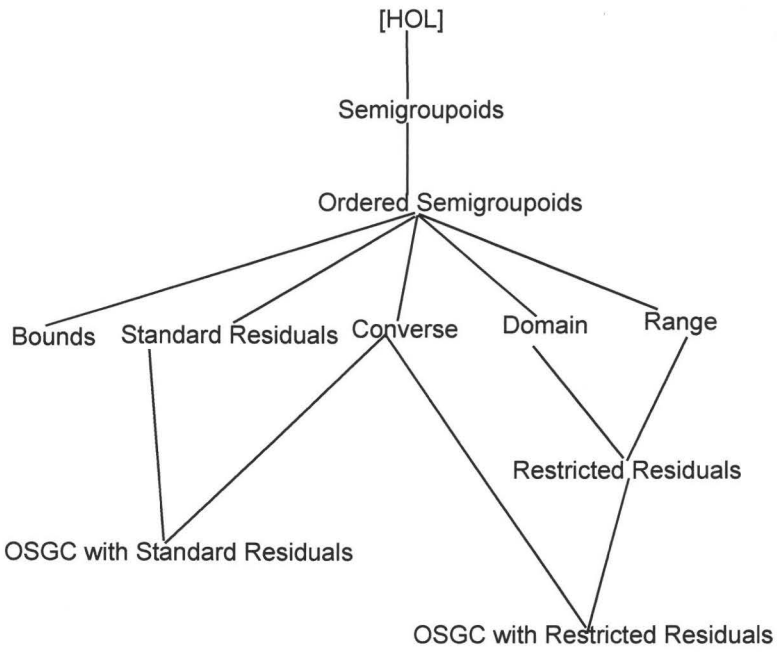

Figure 3.1: A Simplified View of Theory Organization

In this chapter, we give an overview of the organization of our theories. These theories are commonly defined by adding some definitions, declarations and theorems based on new concepts to existing theories. For example, theory *Main* in logic HOL is extended to our first theory *Semi*, theory *Semi* is then extended to theory *OrdSemi*. A theory hierarchy obtained through these extensions is shown in appendix A.

This approach allows us to reason about weak theories (i.e., theories with fewer symbols and less axioms) which mainly are located at the top of the hierarchy. This

way also allows those theories with complicated structure to be built step by step through *import* of theories with simple structure. We also benefit from reusing theories. It helps us systematically add a large number of new theories.

To effectively present the hierarchy, we are going to group our theories into several subgroups to give a simplified view (see Figure 3.1). In each subgroup, theories are closely built to provide solid support for our higher-level applications. Each subgroup is also a subhierarchy. The theories of each subhierarchy are closely related to a topic in [Kah08]. Semigroupoids and the concept of restricted residuals had not been available before [Kah08]. Our goal is developing a framework of semigroupoid theories, which is able to implement converse, domain, range, standard residuals if they exists, and restricted residuals for finite relations, reasoning their properties and then achieving the relationship [Kah08] between unrestricted residuals and restricted residuals under som assumptions in our implementations.

- **Semigroupoid** [Kah08] is what we use as the foundation of our framework since we are working in the setting of the relation-algebraic extensions for finite relations between infinite types and no identity can be assumed. Composition morphisms as well as homsets are introduced here. Basic concepts such as epi, mono and parallel, special objects such as terminal and initial, are also included in the theory of semigroupoids.

- **Ordered Semigroupoid** [Kah08] extends semigroupoids by adding an inclusion operator and properties which are based on the new operator. Here we provide transitivity and monotonicity rules for reasoning with inclusion. Subidentities which is defined through inclusion operator and several related important theorems are also introduced. We also present some lemmas which are useful for showing properties of residuals. All these are in theory OrdSemi which is short of Ordered Semigroupoids. The theories of semigroupoid and ordered semigroupoids are developed by transforming the theories of category and ordered category in [DG04]. We extended the theory of semigroupoids to the theory of properties of operators on homsets and the theory of bounds.

- **Ordered Semigroupoids with Domain and Range** add domain and range operators and related properties to ordered semigroupoids. We define domain operator in the way which is defined in [DG04]. The definition of domain is based on [DMS03]. The theories of range are dual theories of domain. This part contains the following theories: *PreDomSemi, MonPreDom-*

Semi, DomSemi, PreRanSemi, MonPreRanSemi and *RanSemi.* We prepare these theories for the implementation of restricted residuals.

- **Restricted Residuals and Standard Residuals** : this is an exciting part of our implementations. Standard residuals are actually the regular residuals. We also simply call them residuals which include two parts: left residuals and right residuals. Theories about standard residuals are OrdSemiRes, LResSemi, RResSemi, ResSemi and ResOSGC. Restricted residuals [Kah08] are defined through domain and range operators. Restricted residuals also contain two parts: left residuals and right residuals. The theories involving Restricted residuals include *OrdSemiRestrRes, RestrLResSemi, RestrRResSemi, RestrResSemi, RestrResAndRes* and RestrResOSGC.

We build a theory by taking at least one of the following steps:

- Establishing new structures for abstract relational algebraic by defining Isabelle report or Isabelle locale.
- Defining new concepts such as subidentities, parellel morphisms by using **constdefs** in Isabelle.
- Defining and proving lemmas about the properties of the structures of abstract relational algebra.
- Defining auxiliary lemmas explicitly which will be very helpful to prove other lemmas.

In the following chapters, we will proceed by discussing the theories in Figure 3.1 as well as mark our contribution in relation to [Kah08] and [DG04].

Chapter 4

Transformation of theories from categories to semigroupoids

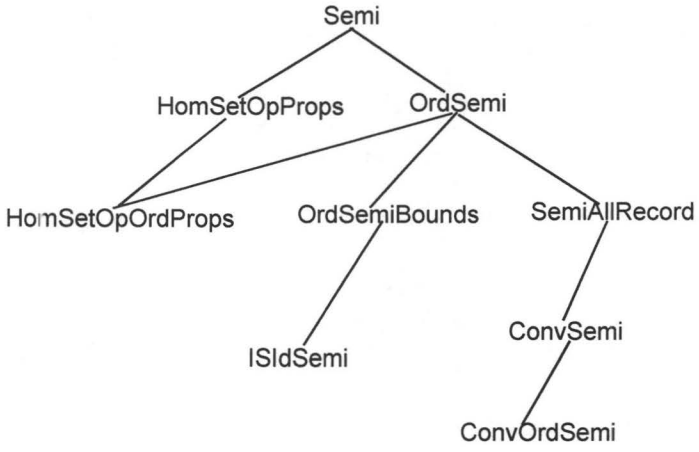

Figure 4.1: Hierarchy of the theories this chapter involves

In this chapter, we focus on three aspects:

- Explain how we build *Semi* and *OrdSemi* and other theories, by modifying definitions and reformulating theorems of theory *Cat* and *OrdCat* and other theories in appendix B of [DG04]. Many definition and relational properties are defined using identities in category system. We modified them to ensure them adapt to our system. Here *Semi* and *OrdSemi* represent semigroupoids [Kah08] and ordered semigroupoids [Kah08] respectively. Similarly, *Cat* and *OrdCat* represent categories and ordered categories.
- Explain the new parts we add and why we add them. Many relational properties

are proved using identities in category system. New theorems are added to help reprove many theorems involving identities in their proofs.

- Present important decisions we made in implementing these theories.

Semi theory is based on the predefined *Main* theory of Isabelle/HOL. It is at the bottom of our theories in Figure 3.1. *OrdSemi* theory is an extension of *Semi*. Other theories are extensions of *Semi* directly or indirectly.

4.1 Semigroupoids (*New Definition*)

Theory *Semi* is built in the way that theory *Cat* in appendix B of [DG04] is built except that *Semi* has no identity-related content of *Cat* because no identities are assumed in Semigroupoids. For theorems in *Cat* which are proved by taking advantage of the properties of identities, we reprove them in *Semi* by using new approaches.

Now we proceed by explaining the important definitions in Theory *Semi*. Understanding the definitions of *Semi* is very helpful to understand other theories.

4.1.1 Basic definitions

In Isabelle/Isar, we build theory *Semi* which is based on theory *Main* of Isabelle/HOL that is an extension of all the basic predefined theories such as *Int*, *Set*, *List* and *Map*.

theory *Semi*

imports *Main*

A semigroupoid (O, M, src, trg, $\odot$) [Kah08] is a graph, where:

- O is a set of objects as vertices
- M is a set of morphisms as edges
- src and trg are functions such that for every morphism f from object A to object B, we have src f = A and trg f = B

- $\odot$ is a binary composition operator. The composition of two morphisms f and g is defined *iff* trg f = src g, and so src($f \odot g$) = src f and trg($f \odot g$) = trg g. The composition operator is associative.

The above structure is defined by using records in Isabelle:

record ('o, 'm) Semigroupoid =

isObj :: 'o $\Rightarrow$ bool	(Obj$\imath$ - [1000] 999)
isMor :: 'm $\Rightarrow$ bool	(Mor$\imath$ - [1000] 999)
cmp :: 'm $\Rightarrow$ 'm $\Rightarrow$ 'm	(**infixr** $\odot\imath$ 200)
Ssrc :: 'm $\Rightarrow$ 'o	(src$\imath$ - [1000] 999)
Strg :: 'm $\Rightarrow$ 'o	(trg$\imath$ - [1000] 999)

Where:

- isObj checks whether an element of type 'o is in the semigroupoid
- isMor checks whether an element of type 'm is in the semigroupoid
- cmp is the composition of two morphisms in the semigroupoid
- Ssrc and Strg give the source object and target object respectively

Type variables 'o and 'm prior to our record name Semigroupoid indicates that the fields of Semigroupoid may involve these named type variables. Each field of Semigroupoid has a specified polymorphic type. Each field also is provided with a syntax to the user for convenience. For example, the type of field isObj is 'o $\Rightarrow$ bool, its syntax is (Obj$\imath$ - [1000] 999) . The higher the number the higher the priority. The annotation **infixr** means that the symbol $\odot$ can be used as an infix operator that associates to the right. The token $\imath$ [DG04] in a syntax annotation applies to the structural parameters.

We write

$$\mathrm{a} \leftrightarrow \mathrm{b} = \{R \mid R \in M, \mathrm{src}\ R = \mathrm{a}, \mathrm{trg}\ R = \mathrm{b}\}$$

for the set of morphisms from object a to object b of a semigoupoid S = (O, M, src, trg, $\odot$). It is called the homset [BW99] from a to b. In Isabelle/Isar, the definition of homset is in the form $\{x.\ Px\}$ through keyword **constdefs** which is used to introduce new concepts. The line following **constdefs** gives the type of homset.

constdefs

homset :: ('o, 'm, 'r) Semigroupoid-scheme $\Rightarrow$ 'o $\Rightarrow$ 'o $\Rightarrow$ 'm set (**infixr** $\leftrightarrow_\imath$ 300)

homset s a b == {f . isObj s a $\wedge$ isObj s b $\wedge$ isMor s f $\wedge$ Ssrc s f = a $\wedge$ Strg s f = b}

('o, 'm, 'r) Semigroupoid-scheme comprises all possible extensions to the fields of Semigroupoid record. The type of the pseudo-field *more* of the record Semigroupoid is made explicit by providing type parameter 'r of Semigroupoid-scheme.

In Isabelle, the definition of a concept is usually followed by a user-level lemma for readability. And then other lemmas about the properties of the new concept follow the user-level lemma. The user-level lemma of homset is:

lemma (**in** Semigroupoid) hs-def:

a $\leftrightarrow$ b = {f . Obj a $\wedge$ Obj b $\wedge$ Mor f $\wedge$ src f = a $\wedge$ trg f = b}

by (unfold homset-def, simp)

Semigroupoid locale is the first locale in our implementation which starts with the keyword **locale**, followed by the keyword **fixes** which fixes the structure the locale will be working on. We declare other locales in our implementation by *import* this locale without using **fixes**. The axioms are listed through the commands **assumes**. Then we use these axioms to obtain the derivation of properties of our structures.

locale Semigroupoid =

fixes C :: ('o, 'm, 'r) Semigroupoid-scheme (**structure**)

assumes src-defined[intro?,simp]: Mor f $\Longrightarrow$ Obj (src f)

assumes trg-defined[intro?,simp]: Mor f $\Longrightarrow$ Obj (trg f)

assumes cmp-defined[intro?,simp]: $[\![$ Mor f; Mor g; trg f = src g $]\!]$ $\Longrightarrow$ Mor (f $\odot$ g)

assumes cmp-src[simp]: ⟦ Mor f; Mor g; trg f = src g ⟧ ⟹ src (f ⊙ g) = src f

assumes cmp-trg[simp]: ⟦ Mor f; Mor g; trg f = src g ⟧ ⟹ trg (f ⊙ g) = trg g

4.1.2 Theorems

The format of a lemma in our implementation is :

lemma (**in** *locale*)*lemma-name*:
lemma-statement
proof

Here **lemma** and **in** are keywords of Isabelle/Isar. *locale* indicates in which locale the lemma is defined. For example, the above lemma *hs-def* is defined in locale Semigroupoid. *hs-def* is visible in the scope of the locale Semigroupoid. It is also visible in the scope of other locales which are directly or indirectly extensions of the locale Semigroupoid. The rule applies to other lemmas as well. The proving process *proof* of a lemma follows *lemma-statement.*

In order to flexibly use the axioms in a locale and the definitions of new concepts, we need to supply all kinds of variations of axioms and the user-level lemmas of new concepts in Isabelle/Isar. Then we can directly or indirectly use these definitions and axioms in different settings to achieve the result we are interesting in.

For example, the auxiliary lemmas *homset,homset0* of theory *Semi* are variations of the user-level lemma *hs-def.* We prove these variations by using lemma *hs-def* through adding it to the simplifier. And then these variations can be added to the simplifier or as an introduction rule to inform Isabelle/Isar how to use them to prove other theorems. Here homset0 is added as an introduction rule.

lemma (**in** Semigroupoid) homset:

fixes a :: 'o **and** b :: 'o **and** f :: 'm

assumes a[simp]: Obj a

assumes b[simp]: Obj b

assumes f[simp]: Mor f

assumes src[simp]: src f = a

assumes trg[simp]: trg f = b

shows f : a ↔ b

by (simp add: hs-def)

lemma (**in** Semigroupoid) homset0[intro?]:

assumes f [intro,simp]: Mor f

assumes [intro,simp]: src f = a

assumes [intro,simp]: trg f = b

shows f : a ↔ b

proof −

from f **have** Obj (src f) **by** (rule src-defined)

also from f **have** Obj (trg f) **by** (rule trg-defined)

ultimately show ?thesis **by** (simp add : hs-def)

qed

In the theory of semigroupoids, we reformulate the concepts of special morphisms and parallel morphisms and the related theorems of the theory *Cat* in appendix B of [DG04].

The first theory *Semi* which serves as the basis of our hierarchy provides the basic semigroupoid record, semigroupoid locale and some important concepts for easily building other theories in our implementation.

4.2 Ordered Semigroupoids (OSG) (*New Definitions, New Theorems*)

theory OrdSemi

imports Semi

Theory *OrdSemi* is built by extending the existing theory *Semi* through adding a new operator and the axioms for the new operator. Our higher level applications are then based on *OrdSemi*.

In this theory, we redefine concepts and reformulate theorems of the theory *OrdCat* in [DG04]. In ordered semigroupoids, we define subidentities without using identities, and give the properties of subidentities in our own way, based on the need of higher level implementations.

4.2.1 Basic definitions and theorems

In this part, we define *OrderedSemigroupoid* OrderedSemigroupoid and *OrderedSemigroupoid* locale by using *Semigroupoid* report and *Semigroupoid* locale in the theory of semigroupoids.

record ('o, 'm) OrderedSemigroupoid = ('o, 'm) Semigroupoid +

incl :: 'm ⇒ 'm ⇒ bool (**infixr** $\sqsubseteq_i$ 50)

locale OrderedSemigroupoid = Semigroupoid OS +

assumes incl-refl[intro,simp]: Mor f ⟹ f ⊑ f

assumes incl-trans[trans]:

⟦ f ⊑ g; g ⊑ h; f : a ↔ b; g : a ↔ b; h : a ↔ b ⟧ ⟹ f ⊑ h

assumes incl-antisym[trans]:

⟦ f ⊑ g; g ⊑ f; f : a ↔ b; g : a ↔ b ⟧ ⟹ f = g

assumes comp-incl-mon[intro,simp]:

⟦ f ⊑ f'; g ⊑ g'; f : a ↔ b; f' : a ↔ b; g : b ↔ c; g' : b ↔ c ⟧

⟹ (f ⊙ g) ⊑ (f' ⊙ g')

We add an ordering "⊑" on each homset of our structure to get a new record *OrderedSemigroupoid*. The ordering is defined only between morphisms with the same sources and targets,that is, it is a partial ordering. The axioms that resulted from adding this new symbol are listed in the locale *OrderedSemigroupoid*.

In the above locale declaration, the parameter **OS** is bound inside the locale *OrderedSemigroupoid*. This parameter (invisibly) accurs as argument to the locale *OrderedSemigroupoid*. Therefore, Locale *OrderedSemigroupoid* has the extensible record type ('o, 'm, 'r)OrderedSemigroupoid-scheme as its type. (see page 22 of [DG04])

In the theory of ordered semigroupoids, we reformulate most theorems of the theory *OrdCat* in appendix B of [DG04], based on the axioms defined in the above locale.

4.2.2 Subidentities

Subidentities are defined via idendities in category system [DG04]. We can not define them in the way in our semigroupoid system.

```
''OC-isSId s R == if isMor s R & (Csrc s R = Ctrg s R)
                  then incl s R (CId s (Csrc s R))
                  else arbitrary''
```

The concept of subidentities (see section 2.3 of [Kah08]) is introduced in the theory of ordered semigroupoids. We say a morphism R from a to a is a subidentity if and only if for all objects b and all morphisms F: $a \rightarrow b$ and G: $b \rightarrow a$, we have $R \odot F \sqsubseteq F$ and $G \odot R \sqsubseteq G$. In Isabelle/Isar, this is written as:

constdefs

OS-isSId :: ('o, 'm, 'r) OrderedSemigroupoid-scheme $\Rightarrow$ 'm $\Rightarrow$ bool (isSId$_i$ - [1000] 999)

OS-isSId s R == if (isMor s R & Ssrc s R = Strg s R)

then (let a = Ssrc s R in

(∀b.

(∀f. f : homset s a b ⟶incl s (cmp s R f) f) &

(∀g. g : homset s b a ⟶ incl s (cmp s g R) g)

))

else arbitrary

Notice how the " *if* **A** then **B** else *arbitrary*" construct is used in the definition of a new concept. The expression *arbitrary* is used for dealing with partial functions.

The following are some theorems which are achieved from the definition of subidentities. One of them is *isSId-def* which is a user-level lemma of the above definition for readability. *isSId-right*, *isSId-left*, *isSId* and *isSId-intro1* are four variations of *isSId-def*. We inform Isabelle/Isar by adding some of them to the simplifier or providing some as introduce rules.

Lemma *isSId-def* is actually a translation of the above definition of subidentities. When a new concept is defined, we normally provide such a user-level lemma for future use.

lemma (**in** OrderedSemigroupoid) isSId-def:

R : a $\leftrightarrow$ a $\Longrightarrow$ isSId R = ($\forall$ b.

($\forall$f. f : a $\leftrightarrow$ b$\longrightarrow$ (R $\odot$ f) $\sqsubseteq$ f) &

($\forall$g. g : b $\leftrightarrow$ a $\longrightarrow$ (g $\odot$ R) $\sqsubseteq$ g))

by (unfold OS-isSId-def, simp add: Let-def)

Based on the definition of subidentities, if R is a subidentity on a, it must be a left subidentity. Lemma *isSId-left* is introduced to show this.

lemma (**in** OrderedSemigroupoid) isSId-left[intro?, simp]:

assumes R-t: R : a $\leftrightarrow$ a

assumes R: isSId R

assumes f: f : a $\leftrightarrow$ b

shows (R $\odot$ f) $\sqsubseteq$ f

proof −

from R-t R f **show** ?thesis **by** (unfold OS-isSId-def, simp)

qed

If R is a subidentity on a, it must be a right subidentity. Lemma *isSId-right* is provided for this. The above lemma *isSId-left* and the following lemma *isSId-right* are very useful for proving inclusion-related theorems in our higher level implementation.

lemma (**in** OrderedSemigroupoid) isSId-right[intro?, simp]:

assumes R-t: R : a ↔ a

assumes R: isSId R

assumes g: g : b ↔ a

shows (g ⊙ R) ⊑ g

proof −

from R-t R g **show** ?thesis **by** (unfold OS-isSId-def, simp)

qed

If R is a subidentity on a, it must be a left subidentity as well as a right subidentity.

lemma (**in** OrderedSemigroupoid) isSId:

assumes R-t: R : a ↔ a

assumes R: isSId R

assumes f: f : a ↔ b

assumes g: g : b ↔ a

shows (R ⊙ f) ⊑ f & (g ⊙ R) ⊑ g

proof −

from R-t R f g **show** ?thesis **by** (unfold OS-isSId-def, simp)

qed

Lemma *isSId-intro1* shows a way to prove that a morphism R is a subidendity. In the lemma, the sufficient condition of *isSId* R is that R is a left subidentity as well

as a right subidentity.

lemma (**in** OrderedSemigroupoid) isSId-intro1:

assumes R-t: R: a↔a

assumes i1: ⋀b f. f : a ↔ b⟹ (R ⊙ f) ⊑ f

assumes i2: ⋀b g. g : b ↔ a ⟹ (g ⊙ R) ⊑ g

shows isSId R

proof −

from R-t i1 i2 **show** isSId R **by**(subst isSId-def, auto)

qed

In our further applications, we usually know that a subidentity J on an object a includes morphism R, and need to prove that morphism R is also a subidentity on an object a. For this situation, the following theorem *isSId-intro2* is provided. The process of directly proving *isSId-intro2* is very complicated. In order to make its proving easier and make its proof shorter for readability, two lemmas, *isSId-intro2-right* and *isSId-intro2-left*, are provided to support its proving.

isSId-intro2-right show R is a right subidentity if J is a subidentity and it includes R.

lemma (**in** OrderedSemigroupoid) isSId-intro2-right[intro]:

assumes j-t[intro, simp]: j : a ↔ a

assumes j[intro, simp]: isSId j

assumes R-t[intro, simp]: R : a ↔ a

assumes Rj[intro, simp]: R ⊑ j

assumes f[intro,simp]: f: a↔b

shows (R ⊙ f) ⊑ f

proof −

let ?g = j ⊙ f

have ?g: a↔b **by** auto

moreover have bf1:(R ⊙ f) ⊑ ?g & (?g ⊑ f) **by** auto

moreover have bf2 : (((R ⊙ f) ⊑ ?g) & (?g ⊑ f)) ⟹ ((R ⊙ f) ⊑ f) **by** (rule-tac incl-trans, auto)

ultimately show ?thesis **by** auto

qed

isSId-intro2-left proves R is also a left subidentity if J is a subidentity and it includes R.

lemma (**in** OrderedSemigroupoid) isSId-intro2-left[intro]:

assumes j-t[intro, simp]: j : a ↔ a

assumes j[intro, simp]: isSId j

assumes R-t[intro, simp]: R : a ↔ a

assumes Rj[intro, simp]: R ⊑ j

assumes g[intro,simp]: g: b↔a

shows (g ⊙ R) ⊑ g

proof −

let ?f = g ⊙ j

have ?f: b↔a **by** auto

moreover have bf1:(g ⊙ R) ⊑ ?f & (?f ⊑ g) **by** auto

moreover have bf2 : (((g ⊙ R) ⊑ ?f) & (?f ⊑ g)) ⟹ ((g ⊙ R) ⊑ g) **by** (rule-tac incl-trans, auto)

ultimately show ?thesis **by** auto

qed

The conclusions of the above two lemmas provide the sufficient condition of *isSId R*,

based on lemma *isSId-intro1*. Therefore, with the assumptions of *isSId j* and $R \sqsubseteq j$, lemma *isSId-intro2* easily reaches that R is a subidendity by applying those two lemmas.

lemma (in OrderedSemigroupoid) isSId-intro2:

assumes j-t[intro, simp]: j : a ↔ a

assumes j[intro, simp]: isSId j

assumes R-t[intro, simp]: R : a ↔ a

assumes Rj[intro, simp]: R ⊑ j

shows isSId R

by (subst isSId-def, auto)

The following defines the set of subidentities on an object *a*. Following the definition, we give four theorems: *SId-def,SId-intro*, *SId-homset* and *SId*. Please read the corresponding part of appendix B for detailed information about these theorems. This part is developed for further domain and range applications.

constdefs

OS-SId :: ('o, 'm, 'r) OrderedSemigroupoid-scheme ⇒ 'o ⇒ 'm set (SIdι - [1000] 999)

OS-SId s a == Collect (λ m . m : homset s a a & OS-isSId s m)

In our higher level implementations, the properties of subidentites are used a lot. Theorems proved via idendities in category system, are proved via subidentities in our system.

4.3 Other theories (*New theorem*)

We also buid some other theories like *HomsetOpProps*, *HomsetOpOrdProps*, *OrdSemiBounds*, *SemiAllRecords*, *ISIdSemi*, *ConvSemi* and *ConvOrdSemi*. Here, *HomsetOpProps*, an direct extension of *Semi*, is a theory of properties of operations on homsets. *OrdSemiBounds* and *SemiAllRecords* are direct extensions of *OrdSemi*, and

ISIdSemi is directly based on *OrdSemiBounds*. In *OrdSemiBounds*, upper and lower bounds and top and bottom morphisms are introduced. Idempotent subidentities and their properties are introduced in *ISIdSemi*. *ConvOrdSemi* is the theory of ordered semigroupoids with converse (OSGC). These theories are achieved by transforming the correspoinding parts in appendix B of [DG04] into our semigroupoids system. For example, *OrdSemiBounds* is based on *OrdCatBounds* and *SemiAllRecords* is based on *AllRecord*.

We provide these theories for the implementations of range, domain, standard residuals and restricted residuals. These theories also provide solid support for the future implementations of semi-Allegories [Kah08] and Kleene semigroupoids [Kah08]. Interested readers are referred to our appendix B for these theories. Here we choose to discuss what we add.

We add a new theorem *isid-issid* in *ISIdSemi* which presents that an idempotent subidentity on object a is also a subidentity on object a. At the same time we give it *simp* and *intro* in Isabelle/Isar by adding *intro* and *simp* following the theorem name. Therefore, for our higher level applications, idempotent subidentitis automatically have all the properties of subidentities.

lemma (**in** OrderedSemigroupoid) isid-issid[intro, simp]:

assumes [intro,simp]: R : ISId a

shows isSId R

proof −

have isISId R **by** (rule ISId, best)

moreover have R : a ↔ a **by** (simp)

ultimately show ?thesis **by** auto

qed

A record of SemiAllegory is defined in *SemiAllRecords*. We present this record in a separate theory in order to make it easier to have theories that do not need these components, but components higher up in the record hierarchy. We add a field *All-rang* in the record for convenience. So we can proceed to develop the theories of range operation.

record ('o, 'm) SemiAllegory = ('o, 'm) OrderedSemigroupoid +

meet :: 'm ⇒ 'm ⇒ 'm (**infixr** ⊓ι 70)

conv :: 'm ⇒ 'm (-⌣ι [1000] 999)

All-dom :: 'm ⇒ 'm (domι - [1000] 999)

All-rang :: 'm ⇒ 'm (rangι - [1000] 999)

Chapter 5

Domain and Range

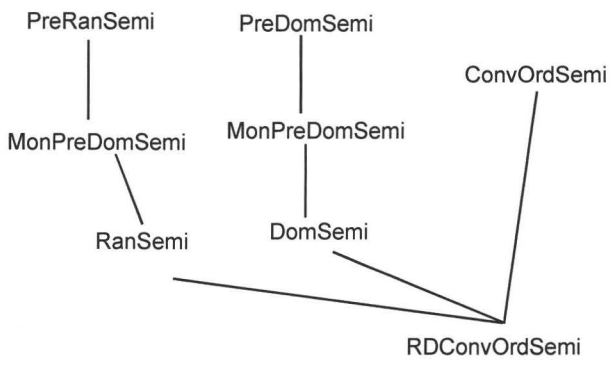

Figure 5.1: Dependency graph of theories involving domain and range

The theories about domain and range are prepared for defining the concept of restricted residuals. Domain theories are built in ordered semigroupoids by formulating the corresponding theories in ordered categories in appendix B of [DG04]. Range is dual part of domain. In ordered semigroupoids with converse, range and domain can be defined in terms of each other. This has been proved in the new theory *RDConvOrdSemi*.

In this part, we focus on three aspects:

- How domain and range theories are developed;
- New theorems which are added in order to reprove theorems involving identities in their proofs, and theorems which are reproved;
- New theorems in the new theory *RDConvOrdSemi*which is prepared for higher level application.

5.1 OSG with Domain

We build the theory of ordered semigroupoids with domain through a subhierarchy of (*PreDomSemi*, *MonPreDomSemi* and *DomSemi*), which can be see from figure 5.1. At the bottom of the subhierarchy, *PreDomSemi* gets the symbol of domain operator from theory *SemiAllRecord* and gets the properties of multiplicatively-idempotent subidentities from theory *ISIdSemi*. Therefore, we introduce *PreDomSemi* by:

theory PreDomSemi

imports SemiAllRecord ISIdSemi

The three axioms that characterize the domain operator are introduced in the definition of locale *PreDomSemi*.

locale PreDomSemi = OrderedSemigroupoid PDS +

assumes dom-ISId[intro,simp]: R : a $\leftrightarrow$ b $\Longrightarrow$ dom R : ISId a

assumes dom-self[intro,simp]: R : a $\leftrightarrow$ b $\Longrightarrow$ R $\sqsubseteq$ dom R $\odot$ R

assumes dom-ISId-cmp[intro,simp]: $[\![$ R : a $\leftrightarrow$ b; P : ISId a $]\!]$ $\Longrightarrow$ dom(P $\odot$ R) $\sqsubseteq$ P

we proceed by giving the derived properties of the domain operator such as *llp1*,*llp2*, *llp*, *dom-isISId-eq* and *dom-decomp* which satisfies a decomposition law. Especially we figure out new approach to reprove the following theorems in ordered semigroupoids which are proved by using the properties of identities in ordered categories in appendix B of [DG04].

lemma (**in** PreDomSemi) isid-isbot:

assumes [intro, simp]: isBot W

assumes [intro, simp]: W : a $\leftrightarrow$ a

shows W : ISId a

proof −

have isSId W

proof (rule-tac j=dom W **in** isSId-intro2, auto)
show W $\sqsubseteq$ dom W **by** (rule isBot, auto)
qed
moreover have W $\odot$ W = W
proof −
have W $\odot$ W $\sqsubseteq$ W
proof −
have W $\sqsubseteq$ dom W **by** (rule isBot, auto)
hence W $\odot$ W $\sqsubseteq$ W $\odot$ dom W **by** (rule comp-incl-mon2, best+)
also have ... $\sqsubseteq$ W **by** (auto)
finally show ?thesis **by** best+
qed
moreover have W $\sqsubseteq$ W $\odot$ W **by** (rule isBot, auto)
ultimately show ?thesis **by** (rule incl-antisym, best+)
qed
ultimately have isISId W **by** (rule-tac isISId-intro, best+)
thus ?thesis **by** (rule-tac ISId-intro, best+)
qed

The new theorem *isSId-intro2*, which is introduced in theory *OrdSemi*, is first used for proving the above theorem *isid-isbot*, in order to avoid identities.

In order to reprove some lemmas such as *dom-strict1* and *dom-left-inv* using the properties of subidentities, instead of proving them via identities, new theorem *isSId-dom* is introduced in the theory. At the same time, we add *isSId-dom* to the simplifier and also inform Isabelle/Isar to use it as an introduction rule.

lemma (**in** PreDomSemi)isSId-dom[intro, simp]:

assumes [intro]: W: a $\leftrightarrow$ b

shows isSId (dom W)

proof −

have dom W : ISId a **by** (rule dom-ISId, auto)

thus ?thesis **by** auto

qed

The new theorem *isSId-dom* and theorem *isSId-left* are applied via "**by** auto" in the line of "**moreover have** dom R $\odot$ R $\sqsubseteq$ R **by** auto" in the following proof of lemma *dom-strict1*.

lemma (**in** PreDomSemi) dom-strict1:

assumes [intro]: isBot (dom R)

assumes cmp-isidbot: isBot (dom R $\odot$ R)

assumes [intro,simp]: R : a $\leftrightarrow$ b

shows isBot R

proof −

have isBot (dom R $\odot$ R) **by** (rule cmp-isidbot)

also have dom R $\odot$ R = R

proof −

have R $\sqsubseteq$ dom R $\odot$ R **by** (rule-tac llp [THEN iffD1], best+)

moreover have dom R $\odot$ R $\sqsubseteq$ R **by** auto

ultimately show ?thesis **by** (rule-tac incl-antisym, best+)

qed

finally show ?thesis **by** auto

qed

Theorem *isSId-dom* and theorem *isSId-left* are automatically applied via "**by** auto" in the line of "**moreover have** ... $\sqsubseteq$ R **by** auto" in the proof of lemma *dom-left-inv*.

lemma (**in** PreDomSemi) dom-left-inv:

assumes [intro]: R : a $\leftrightarrow$ b

shows R = dom R $\odot$ R

proof −

have R $\sqsubseteq$ dom R $\odot$ R **by** (rule dom-self, auto)

moreover have ... $\sqsubseteq$ R **by** auto

ultimately show ?thesis **by** (rule incl-antisym, best+)

qed

For ordered Semigroupoids with monotonic predomain (*MonPreDomSemi*) and ordered semigroupoids with domain (*DomSemi*), we simply build them by reformulating theory *MonPreDomCat* and theory *DomCat* in appendix B of [DG04]. *MonPreDomSemi* is an extension of *PreDomSemi* by introducing monotonicity of the domain operator and *DomSemi* is developed by adding rule *dom-local* to theory *MonPreDomSemi*. So far our theory ordered semigroupoids with domain (*DomSemi*) is built. Interested readers are referred to our appendix B for detailed information about these theories.

5.2 OSG with Range (*New theories*)

An ordered semigroupoid with domain also has range. In ordered semigroupoids with converse and domain, range can be defined in terms of domain and converse: rang R = conv(dom (conv R)) [Kah08] – note that subidentities are not necessary symmetric. In Semi-Allegories [Kah08], range can be further defined as rang R = dom (conv R) because all subidentities are symmetric. In our further implementations, range need work in ordered semigroupoids without converse. This means the above definitions are not available.

In this section, range is defined analogously to domain. The theory of ordered semigroupoids with range is built through a hierarchy of *PreRanSemi*, *MonPreRanSemi* and *RanSemi*, which can be see from figure 5.1.

Theory *PreRanSemi* gets the symbol of range operator from theory *SemiAllRecord* and gets the properties of multiplicatively-idempotent subidentities from theory *ISIdSemi*. Therefore, we introduce *PreRanSemi* by:

theory PreRanSemi

imports SemiAllRecord ISIdSemi

The three axioms that characterize the range operator are introduced in the definition of locale *PreRanSemi*.

locale PreRanSemi = OrderedSemigroupoid PRS+

assumes ran-ISId[intro,simp]: R : a $\leftrightarrow$ b $\Longrightarrow$ rang R : ISId b

assumes ran-self[intro,simp]: R : a $\leftrightarrow$ b $\Longrightarrow$ R $\sqsubseteq$ R $\odot$ rang R

assumes ran-ISId-cmp[intro,simp]: $[\![$ R : a $\leftrightarrow$ b; P : ISId b $]\!]$ $\Longrightarrow$ rang (R $\odot$ P) $\sqsubseteq$ P

Here axioms *ran-ISId*, *ran-self* and *ran-ISId-cmp* are dual to *dom-ISId*, *dom-self* and *dom-ISId-cmp* of locale *PreDomSemi* respectively.

Then we proceed by giving the derived properties of range, which are dual parts of the properties of domain. For example, the corresponding part of "lemma (in PreDomSemi) dom-left-inv" is "lemma (in PreRanSemi) ran-right-inv":

lemma (**in** PreRanSemi) ran-right-inv:

assumes [intro]: R : a $\leftrightarrow$ b

shows R = R $\odot$ rang R

proof −

have R $\sqsubseteq$ R $\odot$ rang R **by** (rule ran-self, auto)

moreover have ... $\sqsubseteq$ R **by** auto

ultimately show ?thesis **by** (rule incl-antisym, best+)

qed

In short, domain operator is a left invariant, analogously range operator is a right invariant. The dual parts of all the lemmas of theory *PreDomSemi* are presented in the theory *PreRanSemi*, such as *llp*, *ran-strict1*, *ran-strict2*, *ran-isISId-eq*, *isSId-ran*, *isid-isbot* and *ran-decomp*.

For ordered Semigroupoids with monotonic prerange (*MonPreRanSemi*) and ordered semigroupoids with domain (*RanSemi*), we simply build them in the way we develop *PreRanSemi*. *MonPreRanSemi* is an extension of *PreRanSemi* by introducing monotonicity of the range operator and *RanSemi* is developed by adding rule *ran-local* to theory *MonPreRanSemi*. So far our theory ordered semigroupoids with range (*DomSemi*) is built. Interested readers are referred to our appendix B for detailed information about these theories.

5.3 OSGC with Range and Domain (*New Theory, New Theorems*)

OSGC is short of ordered semigroupoids with converse operator. We provide important theorems involving domain, range and converse in the theory of OSGC with range and domain operators — *RDConvOrdSemi* theory. These theorems will be used to prove the properties of restricted residuals which will be introduced in the next chapter.

In the section, the most important theorems we intend to prove are *RDCOS-ran* and *RDCOS-dom*.

lemma (**in** RDConvOrdSemi) RDCOS-ran:

assumes [intro]:R : a $\leftrightarrow$ b

shows rang R = (dom($R^\smile$))$^\smile$

Lemma *RDCOS-ran* shows that, in OSGC with domain, range can be defined as $rang\ R = (dom(R^\smile))^\smile$.

lemma (**in** RDConvOrdSemi) RDCOS-dom:

assumes [intro]:R : a $\leftrightarrow$ b

shows dom R = (rang($R^\smile$))$^\smile$

Dually, lemma *RDCOS-ran* shows that, in OSGC with range, domain can be defined as *dom R* = $(rang(R^\smile))^\smile$.

In order to prove lemma *RDCOS-ran*, i.e. *rang R* = $(dom(R^\smile))^\smile$, based on *iff* rule in Isabelle/Isar, we provide theorem *RDCOS-incl1* (i.e. $(dom(R^\smile))^\smile \sqsubseteq rang\ R$) and theorem *RDCOS-incl3* (i.e. $rang\ R \sqsubseteq (dom(R^\smile))^\smile$) first.

lemma (**in** RDConvOrdSemi)RDCOS-incl1:

assumes [intro]:R : a $\leftrightarrow$ b

shows (dom($R^\smile$))$^\smile$ $\sqsubseteq$ rang R

proof −

have R $\sqsubseteq$ R $\odot$ rang R **by** auto

have $R^\smile$ $\sqsubseteq$ (R $\odot$ rang R)$^\smile$ **by** auto

have d: $R^\smile$ $\sqsubseteq$ (rang R)$^\smile$ $\odot$ $R^\smile$ **by** (rule-tac conv-cmp [THEN subst], auto)

from d **have** dom($R^\smile$) $\sqsubseteq$ dom((rang R)$^\smile$ $\odot$ $R^\smile$) **by** auto

moreover have ... $\sqsubseteq$ (rang R)$^\smile$ **by** auto

ultimately have dom($R^\smile$) $\sqsubseteq$ (rang R)$^\smile$ **by** (rule incl-trans, auto)

hence dd: (dom($R^\smile$))$^\smile$ $\sqsubseteq$ ((rang R)$^\smile$)$^\smile$ **by** auto

from dd **show** ?thesis **by** (rule-tac conv-idem [THEN subst], auto)

qed

lemma (**in** RDConvOrdSemi)RDCOS-incl3:

assumes [intro]:R : a $\leftrightarrow$ b

shows rang R $\sqsubseteq$ (dom($R^\smile$))$^\smile$

proof −

let ?R′ = R$^\smile$

have (?R′)$^\smile$ = R **by** auto

moreover have (rang((?R′)$^\smile$))$^\smile$ ⊑ dom(?R′) **by** (rule ResOSGC-incl2, auto)

ultimately have r: (rang R)$^\smile$ ⊑ dom(R$^\smile$) **by** auto

from r **have** rr: ((rang R)$^\smile$)$^\smile$ ⊑ (dom(R$^\smile$))$^\smile$ **by** auto

from rr **show** ?thesis **by** (rule-tac conv-idem [THEN subst], auto)

qed

Dually, we provide theorem *RDCOS-incl2* (i.e. $(rang(R^\smile))^\smile \sqsubseteq dom\ R$) and theorem *RDCOS-incl4* (i.e. $dom\ R \sqsubseteq (rang(R^\smile))^\smile$) for proving lemma *RDCOS-dom*, i.e. $dom\ R = (rang(R^\smile))^\smile$.

lemma (**in** RDConvOrdSemi)ResOSGC-incl2:

assumes [intro]:R : a ↔ b

shows (rang(R$^\smile$))$^\smile$ ⊑ dom R

proof −

have R ⊑ dom R ⊙ R **by** auto

have R$^\smile$ ⊑ (dom R ⊙ R)$^\smile$ **by** auto

have d: R$^\smile$ ⊑ R$^\smile$ ⊙ (dom R)$^\smile$ **by** (rule-tac conv-cmp [THEN subst], auto)

from d **have** rang(R$^\smile$) ⊑ rang(R$^\smile$ ⊙ (dom R)$^\smile$) **by** auto

moreover have ... ⊑ (dom R)$^\smile$ **by** auto

ultimately have rang(R$^\smile$) ⊑ (dom R)$^\smile$ **by** (rule incl-trans, auto)

hence dd: (rang(R$^\smile$))$^\smile$ ⊑ ((dom R)$^\smile$)$^\smile$ **by** auto

from dd **show** ?thesis **by** (rule-tac conv-idem [THEN subst], auto)

qed

lemma (**in** RDConvOrdSemi)RDCOS-incl4:

assumes [intro]:R : a $\leftrightarrow$ b

shows dom R $\sqsubseteq$ (rang(R$^\smile$))$^\smile$

proof −

let ?R′ = R$^\smile$

have (?R′)$^\smile$ = R **by** auto

moreover have (dom((?R′)$^\smile$))$^\smile$ $\sqsubseteq$ rang(?R′) **by** (rule RDCOS-incl1, auto)

ultimately have r: (dom R)$^\smile$ $\sqsubseteq$ rang(R$^\smile$) **by** auto

from r **have** rr: ((dom R)$^\smile$)$^\smile$ $\sqsubseteq$ (rang(R$^\smile$))$^\smile$ **by** auto

from rr **show** ?thesis **by** (rule-tac conv-idem [THEN subst], auto)

qed

After the above four lemmas are proved, obviously *RDCOS-ran* theorem holds via *RDCOS-incl1* and *RDCOS-incl3*. Similarly, *RDCOS-ran* holds via *RDCOS-incl2* and *RDCOS-incl4*, too.

In the proofs of *RDCOS-incl1* and *RDCOS-incl2*, the following rules have been used. For $R : a \leftrightarrow b$,

- $(rang\ R)^\smile$ is a subidentity on b and $(dom\ R)^\smile$ is a subidentity on a .
- $(rang\ R)^\smile$: $ISId\ b$, and $(dom\ R)^\smile$: $ISId\ a$.

In the current *RDConvOrdSemi* theory, see the following lemmas we have proved before proving these four lemmas for achieving the above rules.

- *RDCOS-convRan-left* and *RDCOS-convRan-right*
  $\Leftrightarrow$ $(rang\ R)^\smile$ is a subidentity on b
- *RDCOS-convRan-left*, *RDCOS-convRan-right* and *RDCOS-convRan-I*
  $\Leftrightarrow$ *RDCOS-convRan-isISId* which proves $(rang\ R)^\smile$: $ISId\ b$.
- *RDCOS-convDom-left* and *RDCOS-convDom-right*
  $\Leftrightarrow$ $(Dom\ R)^\smile$ is a subidentity on a

- *RDCOS-convDom-left*, *RDCOS-convDom-right* and *RDCOS-convDom-I* $\Leftrightarrow$ *RDCOS-convDom-isISId* which proves $(Dom\ R)^\smile$: *ISId a*.

At the end of *RDConvOrdSemi* theory, we introduce another two new theorems which are based on *RDCOS-ran* and *RDCOS-dom*, to support proving the property of restricted residuals in the new theory *RestrResOSGC* which will be presented in the next chapter.

lemma (**in** RDConvOrdSemi)RDCOS-incl1:

assumes [intro]:R : a $\leftrightarrow$ b

shows $(\text{dom}(\text{R}^\smile))^\smile \sqsubseteq$ rang R

lemma (**in** RDConvOrdSemi)ResOSGC-incl2:

assumes [intro]:R : a $\leftrightarrow$ b

shows $(\text{rang}(\text{R}^\smile))^\smile \sqsubseteq$ dom R

See *RDConvOrdSemi* theory in appendix B for their proofs.

The theories of this chapter are provided for our higher level applications — ordered semigroupoids with restricted residuals.

Chapter 6

Standard residuals and Restricted Residuals

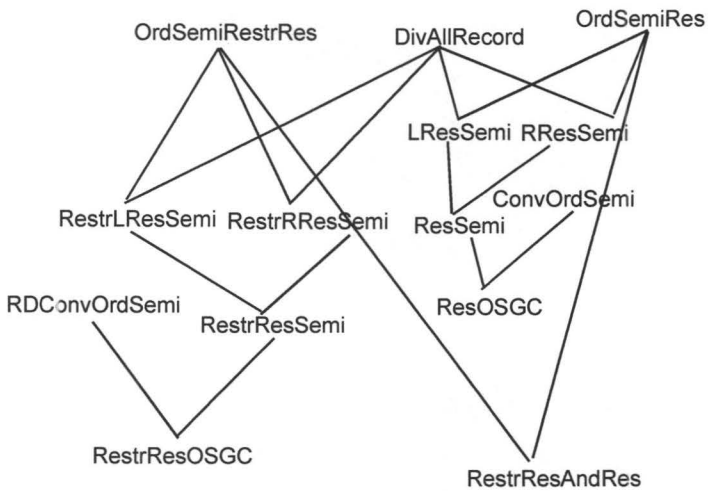

Figure 6.1: Theory Dependency of restricted residuals and standard residuals

This chapter focuses on theories standard residuals and restricted residuals and their properties. First we give the definition of standard residuals and their properties based on the corresponding part in appendix B of [DG04]. Standard residuals are also called residuals in this chapter. For semigroupoids of finite relations between arbitrary sets, standard residuals do not generally exist. In [Kah08], Kahl introduced the concept of restricted residuals. Following standard residuals, we give the definition and properties of restricted residuals in Isabelle/Isar based on section 5 of [Kah08].

A number of new properties of standard residuals and restricted residuals will be added in this chapter, based on chapter 4 of [FK98]. Also, the theorem [Kah08] that in ordered semigroupoids with domain and range, if standard residuals exist,

restricted residuals also exist and can be calculated via standard residuals, will be proved.

6.1 Standard Residuals

In this section, we first introduce the theory of ordered semigroupoids with predicates for standard residuals, *OrdSemiRes*, on which theory *LResSemi* and theory *RResSemi* are based. The theory *OrdSemiRes* extends the theory *OrdSemi* by the definitions of *haveLeftRes* and *haveRightRes* predicates. The following are the derived user-friendly versions of these predicates.

lemma (**in** OrderedSemigroupoid) haveLeftRes-def:

$[\![S : a \leftrightarrow b; R : c \leftrightarrow b; L : a \leftrightarrow c]\!] \Longrightarrow$

haveLeftRes S R L = $(\forall\ X \in a \leftrightarrow c\ .\ (X \odot R \sqsubseteq S) = (X \sqsubseteq L))$

lemma (**in** OrderedSemigroupoid) haveRightRes-def:

$[\![S : a \leftrightarrow b; L : a \leftrightarrow c; R : c \leftrightarrow b]\!] \Longrightarrow$

haveRightRes S L R = $(\forall\ X \in c \leftrightarrow b\ .\ (L \odot X \sqsubseteq S) = (X \sqsubseteq R))$

For concrete relations, we have

- The standard left-residual $R \leftharpoonup S$
  $(x, y) \in (R \leftharpoonup S) \iff \forall z.\ (y, z) \in R \Rightarrow (x, z) \in S$
- The standard right-residual $S \rightharpoonup L$
  $(y, z) \in (S \rightharpoonup L) \iff \forall x.\ (x, y) \in L \Rightarrow (x, z) \in S$

Standard residuals provide standard methods to translate predicate logic formulas involving universal quantifications into relation-algebraic formulas.

The theory of semigroupoids with standard residuals, *ResSemi*, is based on the theories of standard left residuals and standard right residuals *LResSemi* and *RResSemi*, respectively. The symbols of standard left-residual and standard right-residual are

defined in the theory *DivAllRecord*. Due to symmetry, only the theory *LResSemi* is discussed here.

The theory of left residuated semigroupoids, *LResSemi*, takes the standard left residual symbol "$\leftharpoonup$" from the theory *DivAllRecord* and gets its behaviors from the theories *OrdSemiRes*. Therefore, *LResSemi* is an extension of *DivAllRecord* and *OrdSemiRes*.

theory LResSemi

imports OrdSemiRes DivAllRecord

Two axioms for the newly added symbol are introduced in the definition of the locale *LResSemi*.

locale LResSemi = OrderedSemigroupoid LRS +

assumes leftRes-homset[intro,simp]: $[\![R : c \leftrightarrow b; S : a \leftrightarrow b]\!] \Longrightarrow (R \leftharpoonup S) : a \leftrightarrow c$

assumes leftRes[intro,simp]: $[\![R : c \leftrightarrow b; S : a \leftrightarrow b]\!] \Longrightarrow$ haveLeftRes S R (R $\leftharpoonup$ S)

The following properties of standard left residuals are derived in Isabelle/Isar. Their proofs are omitted here.

lemma (**in** LResSemi) lres-src:

assumes [simp]: trg R = trg S

assumes [simp]: Mor R

assumes [simp]: Mor S

shows src (R $\leftharpoonup$ S) = src S

lemma (**in** LResSemi) lres-trg:

assumes [simp]: trg R = trg S

assumes [simp]: Mor R

assumes [simp]: Mor S

shows trg (R $\leftharpoonup$ S) = src R

lemma (**in** LResSemi) lres:

assumes [intro]: R : c $\leftrightarrow$ b

assumes [intro]: S : a $\leftrightarrow$ b

assumes [intro]: X : a $\leftrightarrow$ c

shows ((X $\odot$ R) $\sqsubseteq$ S) = (X $\sqsubseteq$ (R $\leftharpoonup$ S))

Two theorems based on the above theorem *lres* are also provided for convenience.

lemmas (**in** LResSemi) lres1 = lres [THEN iffD1]

lemmas (**in** LResSemi) lres2 = lres [THEN iffD2]

The following are another two properties of standard left residuals in Isabelle/Isar. Their proofs are also omitted here.

lemma (**in** LResSemi) incl-lres:

assumes [intro]: T : a $\leftrightarrow$ c

assumes [intro]: R : c $\leftrightarrow$ b

shows T $\sqsubseteq$ (R $\leftharpoonup$ (T $\odot$ R))

lemma (**in** LResSemi) lrescmp-incl:

assumes [intro]: R : c $\leftrightarrow$ b

assumes [intro]: S : a $\leftrightarrow$ b

shows ((R $\leftharpoonup$ S) $\odot$ R) $\sqsubseteq$ S

Now we merge the theories of standard left-residual and standard right-residual, *LResSemi* and *RResSemi*, to give the theory of residuated semigroupoids, *ResSemi*.

theory ResSemi

imports LResSemi RResSemi

begin

locale ResSemi = LResSemi RS + RResSemi RS

end

We'll use theory *ResSemi* together with theory *RestrResSemi*, which will be introduced in the next section, to build theory *RestrResAndRes* in which restricted residuals is defined via standard residuals under the assumption that standard residuals exist.

6.2 Restricted Residuals (*New theories*)

We build the theories of Restricted Residuals in the way that standard Residuals related theories are developed.

In this section, we first introduce the theory of ordered semigroupoids with predicates for restricted residuals, *OrdSemiRestrRes*, on which theory *RestrLResSemi* and theory *RestrRResSemi* are based. The theory *OrdSemiRestrRes* extends theory *DomSemi* and theory *RanSemi*, which are introduced in chapter 5, by the definitions of *haveRestrLeftRes* and *haveRestrRightRes* predicates. We define *haveRestrLeftRes* and *haveRestrRightRes* predicates based on the definitions of restricted left-residual and restricted right-residual in [Kah08]. The following are the derived user-friendly versions of these predicates.

lemma (**in** OrdSemiRestrRes) haveRestrLeftRes-def:

$[\![S : a \leftrightarrow b; R : c \leftrightarrow b; L : a \leftrightarrow c]\!] \Longrightarrow$

haveRestrLeftRes S R L = ($\forall$ X $\in$ a $\leftrightarrow$ c . ((X $\odot$ R $\sqsubseteq$ S) & (rang X $\sqsubseteq$ dom R)) = (X $\sqsubseteq$ L))

lemma (**in** OrdSemiRestrRes) haveRestrRightRes-def:

$[\![S : a \leftrightarrow b; L : a \leftrightarrow c; R : c \leftrightarrow b]\!] \Longrightarrow$

haveRestrRightRes S L R = ($\forall$ X $\in$ c $\leftrightarrow$ b . ((L $\odot$ X $\sqsubseteq$ S) & (dom X $\sqsubseteq$ rang L)) = (X $\sqsubseteq$ R))

For concrete relations, we have

- $(x, y) \in (R \vdash S) \iff$
  $\forall z.\ (y, z) \in R \Rightarrow (x, z) \in S$ and $\exists z.\ (y, z) \in R \wedge (x, z) \in S$

- $(y, z) \in (S \dashv L) \iff$
  $\forall x.\ (x, y) \in L \Rightarrow (x, z) \in S$ and $\exists z.\ (x, y) \in L \wedge (x, z) \in S$

The theory of semigroupoids with restricted residuals, *RestrResSemi*, is based on the theories of restricted left residuals and restricted right residuals *RestrLResSemi* and *RestrRResSemi*, respectively. The symbols of restricted left-residual and restricted right-residual are defined in the theory *DivAllRecord*. Due to symmetry, only the theory *RestrLResSemi* is introduced here.

The theory of restricted left-residual, *RestrLResSemi*, takes the restricted left-residual symbol "$\vdash$" from theory *DivAllRecord* and gets its behaviors from theory *OrdSemiRestrRes*. Therefore, theory *RestrLResSemi* is an extension of *DivAllRecord* and *OrdSemiRestrRes*.

theory RestrLResSemi

imports OrdSemiRestrRes DivAllRecord

The two axioms *restrleftRes-homset* and *restrleftRes* for the newly added symbol are introduced in the definition of the locale *RestrLResSemi*.

locale RestrLResSemi = OrdSemiRestrRes RLRS +

assumes restrleftRes-homset[intro,simp]: $[\![R : c \leftrightarrow b; S : a \leftrightarrow b]\!] \Longrightarrow (R \vdash S) : a \leftrightarrow c$

assumes restrleftRes[intro,simp]: $[\![R : c \leftrightarrow b; S : a \leftrightarrow b]\!] \Longrightarrow$ haveLeftRes S R (R $\vdash$ S)

The following properties of restricted left-residual are derived in Isabelle/Isar. Their proofs are omitted here.

lemma (**in** RestrLResSemi) restr-lres-src:

assumes [simp]: trg R = trg S

assumes [simp]: Mor R

assumes [simp]: Mor S

shows src (R ⊢ S) = src S

lemma (**in** RestrLResSemi) restr-lres-trg:

assumes [simp]: trg R = trg S

assumes [simp]: Mor R

assumes [simp]: Mor S

shows trg (R ⊢ S) = src R

lemma (**in** RestrLResSemi) restr-lres:

assumes [intro]: R : c ↔ b

assumes [intro]: S : a ↔ b

assumes [intro]: X : a ↔ c

shows ((X ⊙ R ⊑ S) & (rang X ⊑ dom R)) = (X ⊑ (R ⊢ S))

Theorem *restr-lres1* and theorem *restr-lres2* are provided for convinence.

lemmas (**in** RestrLResSemi) restr-lres1 = restr-lres [THEN iffD1]

lemmas (**in** RestrLResSemi) restr-lres2 = restr-lres [THEN iffD2]

We also introduce another two properties of restricted left-residual in Isabelle/Isar. Their proofs are also omitted here.

lemma (**in** RestrLResSemi) incl-restr-lres:

assumes [intro]: T : a ↔ c

assumes [intro]: R : c $\leftrightarrow$ b

assumes TR[intro]: rang T $\sqsubseteq$ dom R

shows T $\sqsubseteq$ R $\vdash$ (T $\odot$ R)

lemma (**in** RestrLResSemi) restr-lrescmp-incl:

assumes [intro]: R : c $\leftrightarrow$ b

assumes [intro]: S : a $\leftrightarrow$ b

shows (R $\vdash$ S) $\odot$ R $\sqsubseteq$ S

Finally, we merge the theories of restricted left-residual and restricted right-residual, *RestrLResSemi* and *RestrRResSemi*, to give the theory of semigroupoids with restricted residuals, *ResSemi*.

theory RestrResSemi

imports RestrLResSemi RestrRResSemi

begin

locale RestrResSemi = RestrLResSemi RRS + RestrRResSemi RRS

end

Theory *ResSemi* works together with theory *RestrResSemi* providing solid support for our top theory *RestrResAndRes* which is presented in the next section.

6.3 New Theorems Added for Standard Residuals and Restricted Residuals (*New Theorems*)

In this section, we'll discuss the new properties we provide for standard residuals and restricted residuals. Some of them hold for standard residuals and restricted residuals both. Others are slightly different.

Two new auxiliary lemmas *restr-lres-incl1* and *restr-lres-incl2* are added in the theory of the restricted left-residual — theory *RestrLResSemi*, to prove :

- $X \sqsubseteq R \vdash S \implies rang\ X \sqsubseteq dom\ R$
- $X \sqsubseteq R \vdash S \implies X \odot R \sqsubseteq S$

The following is their proofs.

lemma (**in** RestrLResSemi) restr-lres-incl1[intro]:
assumes [intro]: X : a ↔ c
assumes [intro]: R : c ↔ b
assumes [intro]: S: a ↔ b
assumes TR[intro]: X ⊑ R ⊢ S
shows rang X ⊑ dom R
proof −
from TR **have** ss: (X ⊙ R ⊑ S) ∧ (rang X ⊑ dom R)
by (rule-tac restr-lres [THEN iffD2], auto)
from ss **show** ?thesis **by** auto
qed

lemma (**in** RestrLResSemi) restr-lres-incl2[intro]:
assumes [intro]: X : a ↔ c
assumes [intro]: R : c ↔ b
assumes [intro]: S: a ↔ b
assumes XR[intro]: X ⊑ R ⊢ S
shows (X ⊙ R ⊑ S)
proof −
from XR **have** ss: (X ⊙ R ⊑ S) ∧ (rang X ⊑ dom R)
by (rule-tac restr-lres [THEN iffD2], auto)

from ss **show** ?thesis **by** auto

qed

The above two theorems give two direct conclusions from the auxiliary lemmas *restr-lres* which translates the definition of the restricted left-residual. Dually, new auxiliary lemmas *restr-rres-incl1* and *restr-rres-incl2* are added in the theory of the restricted right-residual — *RestrRResSemi*.

Another two new theorems *restr-lres-incl-new* and *restr-rres-incl-new* are introduced in the theory *RestrLResSemi* and the theoryRestrRResSemi respectively. The following is *restr-lres-incl-new*. See *restr-rres-incl-new* in appendix B.

lemma (**in** RestrLResSemi) restr-lres-incl-new:

assumes [intro]: R : c $\leftrightarrow$ b

assumes [intro]: S : a $\leftrightarrow$ b

shows rang (R $\vdash$ S) $\sqsubseteq$ dom R

proof −

let ?X =R $\vdash$ S

have [intro]: ?X : a $\leftrightarrow$ c **by** auto

moreover have [intro]: ?X $\sqsubseteq$ (R $\vdash$ S) **by** (rule incl-refl, best)

have (?X $\odot$ R $\sqsubseteq$ S) & (rang ?X $\sqsubseteq$ dom R)

by (rule restr-lres [THEN iffD2], auto)

thus ?thesis **by** (best+)

qed

6.4 Restricted Residuals and Standard Residuals (*New Theory, New Theorems*)

For semigroupoids of finite relations between arbitrary sets, standard residuals do not generally exist. In [Kah08], Kahl provided a theorem to propose that in ordered

semigroupoids with domain and range, if standard residuals exist, restricted residuals also exist and can be calculated via standard residuals. This has been proved in the new theory *RestrResAndRes*, which is an extension of theory *OrdSemiRestrRes* and theory *OrdSemiRes*.

theory RestrResAndRes

imports OrdSemiRestrRes OrdSemiRes

The theorem Kahl proposed in [Kah08] can be translated into two parts:

- haveLeftRes S R L $\Rightarrow$ haveRestrLeftRes S R L'
  where: L' == L $\odot$ dom R
- haveRightRes S L R $\Rightarrow$ haveRestrRightRes S L R'
  where: R' == rang L $\odot$ R

which are translated into the following two lemmas in our implementation.

lemma (**in** OrdSemiRestrRes) RestrRightRes-RightRes:

assumes [simp,intro]: S: a $\leftrightarrow$ b

assumes [simp,intro]: R : c $\leftrightarrow$ b

assumes [simp,intro]: L : a $\leftrightarrow$ c

assumes [intro]:haveRightRes S L R

shows haveRestrRightRes S L (rang L $\odot$ R)

lemma (**in** OrdSemiRestrRes) RestrleftRes-LeftRes:

assumes [intro]: S : a $\leftrightarrow$ b

assumes [intro]: R : c $\leftrightarrow$ b

assumes [intro]: L : a $\leftrightarrow$ c

assumes [intro]:haveLeftRes S R L

shows haveRestrLeftRes S R (L $\odot$ dom R)

The proofs of these two lemmas are pretty long. In order to effectively present how we prove them, a proof overview of the former lemma is provided here.

haveRightRes S L R $\Rightarrow$ haveRestrRightRes S L (rang L $\odot$ R)
$\Leftrightarrow$ — standard res. def, restr. res. def
$(L \odot X \sqsubseteq S) = (X \sqsubseteq R) \Rightarrow (L \odot X \sqsubseteq S \wedge \text{dom } X \sqsubseteq \text{rang } L) = (X \sqsubseteq \text{rang } L \odot R)$
$\Leftrightarrow$ —three subgoals
$(X \sqsubseteq R \wedge \text{dom } X \sqsubseteq \text{rang } L) \Rightarrow (X \sqsubseteq \text{rang } L \odot R)$
— dom-self: $X \sqsubseteq \text{dom } X \odot X$
$((L \odot X \sqsubseteq S) = (X \sqsubseteq R) \wedge X \sqsubseteq \text{rang } L \odot R) \Rightarrow L \odot X \sqsubseteq S$
— rang L is a subidentity
$((L \odot X \sqsubseteq S) = (X \sqsubseteq R) \wedge X \sqsubseteq \text{rang } L \odot R) \Rightarrow \text{dom } X \sqsubseteq \text{rang } L$
— rule *dom-ISId-cm*, rule *dom-incl-mon*

The proof of lemma *RestrRightRes-RightRes* follows.

lemma (**in** OrdSemiRestrRes) RestrRightRes-RightRes:
assumes [simp,intro]: S: a $\leftrightarrow$ b
assumes [simp,intro]: R : c $\leftrightarrow$ b
assumes [simp,intro]: L : a $\leftrightarrow$ c
assumes [intro]:haveRightRes S L R
shows haveRestrRightRes S L (rang L $\odot$ R)
proof(rule haveRestrRightRes-def [THEN sym, THEN iffD1],best+)
show i:$\forall$ X$\in$c $\leftrightarrow$ b. ((L $\odot$ X $\sqsubseteq$ S $\wedge$ dom X $\sqsubseteq$ rang L) = (X $\sqsubseteq$ rang L $\odot$ R))
proof (intro strip)
fix X
assume[intro]: X : c $\leftrightarrow$ b
show (L $\odot$ X $\sqsubseteq$ S $\wedge$ dom X $\sqsubseteq$ rang L) = (X $\sqsubseteq$ rang L $\odot$ R)
proof −
have R:(L $\odot$ X $\sqsubseteq$ S) = (X $\sqsubseteq$ R) **by** (rule haveRightRes, best+)

have L ⊙ X ⊑ S ∧ dom X ⊑ rang L ⟹ X ⊑ rang L ⊙ R

proof −

have X: X ⊑ dom X ⊙ X **by** auto

from R **have** (L ⊙ X ⊑ S ∧ dom X ⊑ rang L)

= (dom X ⊑ rang L ∧ X ⊑ R) **by** auto

also have ... ⟹ (dom X ⊙ X ⊑ rang L ⊙ R) **by** auto

from X **have** (dom X ⊙ X ⊑ rang L ⊙ R) ⟹ (X ⊑ rang L ⊙ R)

by (rule incl-trans, auto)

ultimately show L ⊙ X ⊑ S ∧ dom X ⊑ rang L ⟹ X ⊑ rang L ⊙ R

by best+

qed

moreover have X ⊑ rang L ⊙ R ⟹ L ⊙ X ⊑ S ∧ dom X ⊑ rang L

proof(rule conjI)

show X ⊑ rang L ⊙ R ⟹ L ⊙ X ⊑ S

proof −

have [intro]: rang L ⊙ R ⊑ R **by** auto

have [intro]:X ⊑ rang L ⊙ R ⟹ X ⊑ R

by (rule-tac incl-trans, auto)

from R **have**[intro]: X ⊑ R ⟹ L ⊙ X ⊑ S **by** auto

show X ⊑ rang L ⊙ R ⟹ L ⊙ X ⊑ S **by** auto

qed

next show X ⊑ rang L ⊙ R ⟹ dom X ⊑ rang L

proof −

have dom (rang L ⊙ R) ⊑ rang L **by** auto

also have[intro]: X ⊑ rang L ⊙ R ⟹ dom X ⊑ dom (rang L ⊙ R)

by (rule dom-incl-mon, auto)

have[intro]: dom X $\sqsubseteq$ dom (rang L $\odot$ R) $\Longrightarrow$dom X $\sqsubseteq$ rang L

by (rule-tac incl-trans, auto)

show X $\sqsubseteq$ rang L $\odot$ R $\Longrightarrow$ dom X $\sqsubseteq$ rang L **by** auto

qed

qed

ultimately show (L $\odot$ X $\sqsubseteq$ S $\wedge$ dom X $\sqsubseteq$ rang L) = (X $\sqsubseteq$ rang L $\odot$ R)

by (rule iffI, best+)

qed

qed

qed

Due to symmetry, lemma *RestrLeftRes-LeftRes* is proved in the similar way. See theory *RestrResAndRes* in appendix B for its proof.

6.5 New Properties Added (*New Theory, New Theorems*)

We found that properties of standard residuals in chapter 4 in [FK98], specifically, the following propositions

- Propositions 4.1
  $(S \rightharpoonup Q) = (Q^{\smile} \leftharpoondown S^{\smile})^{\smile}$

- Propositions 4.4(i)
  $(R \leftharpoondown S) \odot (T \leftharpoondown R) \sqsubseteq (T \leftharpoondown S)$;
  $(S \rightharpoonup L) \odot (U \rightharpoonup S) \sqsubseteq (U \rightharpoonup L)$

- Propositions 4.4(iii)
  If $S \sqsubseteq S$, $R' \sqsubseteq R$ and $Q' \sqsubseteq Q$, then
  $(R \leftharpoondown S) \sqsubseteq (R' \leftharpoondown S')$ and $(S \rightharpoonup Q) \sqsubseteq (S' \rightharpoonup Q')$

- Propositions 4.5(i)
  $(F \odot (S \leftharpoonup R)) \sqsubseteq (S \leftharpoonup (F \odot R))$;
  $((Q \rightharpoonup U) \odot T) \sqsubseteq ((Q \odot T) \rightharpoonup U)$

hold for standard residuals in ordered semigroupoids. These properties have been formalized in semigroupoid setting in

- **lemma** (**in** ResOSGC) resOSGC-eq:

  assumes [intro]:Q : a $\leftrightarrow$ b

  assumes [intro]:S : a $\leftrightarrow$ c

  shows $(S \rightharpoonup Q) = (Q^{\smile} \leftharpoonup S^{\smile})^{\smile}$

  Here, ResOSGC, which is the locale declared in the new theory of OSGC with standard residuals.

- **lemma** (**in** LResSemi) lrescom-incl-fs:

  assumes [intro]:S : a $\leftrightarrow$ c

  assumes [intro]:R : b $\leftrightarrow$ c

  assumes [intro]:T : d $\leftrightarrow$ c

  shows $(R \leftharpoonup S) \odot (T \leftharpoonup R) \sqsubseteq (T \leftharpoonup S)$

  lemma (**in** RResSemi) rrescom-incl-fs:

  assumes [intro]:S : a $\leftrightarrow$ c

  assumes [intro]:L : a $\leftrightarrow$ b

  assumes [intro]:U : a $\leftrightarrow$ d

  shows $(S \rightharpoonup L) \odot (U \rightharpoonup S) \sqsubseteq (U \rightharpoonup L)$

- **lemma** (**in** LResSemi) lrescom-incl-ohk:

  assumes [intro]: S : a $\leftrightarrow$ c

  assumes [intro]: S′: a $\leftrightarrow$ c

  assumes [intro]: R : b $\leftrightarrow$ c

  assumes [intro]: R′: b $\leftrightarrow$ c

assumes S[intro]: $S \sqsubseteq S'$
assumes R[intro]: $R' \sqsubseteq R$
shows $(R \leftharpoonup S) \sqsubseteq (R' \leftharpoonup S')$

lemma (**in** RResSemi) rrescom-incl-ohk:
assumes [intro]: $S : a \leftrightarrow c$
assumes [intro]: S': $a \leftrightarrow c$
assumes [intro]: $Q : a \leftrightarrow b$
assumes [intro]: Q': $a \leftrightarrow b$
assumes S[intro]: $S \sqsubseteq S'$
assumes Q[intro]: $Q' \sqsubseteq Q$
shows $(S \rightharpoonup Q) \sqsubseteq (S' \rightharpoonup Q')$

- **lemma** (**in** LResSemi) lrescom-incl-ex:
assumes [intro]:F : $a \leftrightarrow b$
assumes [intro]:R : $b \leftrightarrow c$
assumes [intro]:S : $d \leftrightarrow c$
shows $(F \odot (S \leftharpoonup R)) \sqsubseteq (S \leftharpoonup (F \odot R))$

lemma (**in** RResSemi) rrescom-incl-ex:
assumes [intro]:U : $a \leftrightarrow b$
assumes [intro]:Q : $a \leftrightarrow c$
assumes [intro]:T : $c \leftrightarrow d$
shows $((Q \rightharpoonup U) \odot T) \sqsubseteq ((Q \odot T) \rightharpoonup U)$

respectively.

Among the above propositions, proposition 4.1 and proposition 4.4(i) also hold for restricted residuals. They have been proved in

- **lemma** (**in** RestrResOSGC) restr-resOSGC-eq:

  assumes [intro]:Q : a $\leftrightarrow$ b

  assumes [intro]:S : a $\leftrightarrow$ c

  shows (S $\dashv$ Q) = (Q$^\smile$ $\vdash$ S$^\smile$)$^\smile$

  Here, RestrResOSGC is the locale declared in the new theory of OSGC with restricted residuals. We prove this lemma by using the newly introduced two lemmas *RDCOS-domRan1* and *RDCOS-domRan2* in chapter 5.

- **lemma** (**in** RestrLResSemi) restr-lrescom-incl-fs:

  assumes [intro]:S : a $\leftrightarrow$ c

  assumes [intro]:R : b $\leftrightarrow$ c

  shows (R $\vdash$ S) $\odot$ (T $\vdash$ R) $\sqsubseteq$ (T $\vdash$ S)

  lemma (**in** RestrRResSemi) restr-rrescom-incl-fs:

  assumes [intro]:S : a $\leftrightarrow$ c

  assumes [intro]:L : a $\leftrightarrow$ b

  assumes [intro]:U : a $\leftrightarrow$ d

  shows (S $\dashv$ L) $\odot$ (U $\dashv$ S) $\sqsubseteq$ (U $\dashv$ L)

respectively.

For proposition 4.4 (iii), in ordered semigroupoids with domain and range, when R' $\sqsubset$ R, *dom* R' $\sqsubset$ *dom* R holds obviously. Therefore, when R' $\sqsubset$ R,

$\forall\, X.\ (rang\ X = dom\ R \text{ and } X \sqsubseteq (R \leftharpoonup S)) \longrightarrow dom\ R' \sqsubset rang\ X\ .$

Hence, for such X, $X \sqsubseteq (R' \leftharpoonup S')$ does not hold.

Based on the above analysis, proposition 4.4 (iii) does not hold for restricted left-residual. Dually, it also does not hold for restricted right-residual.

In proposition 4.4 (iii), if $R' \sqsubseteq$ R and $Q' \sqsubseteq$ Q are replaced with $R' =$ R and $Q' =$ Q, it holds for restricted residuals. This has been proved in the following two lemmas for restricted left-residual and restricted right-residual, respectively.

lemma (**in** RestrLResSemi) restr-lrescom-incl-ohk:

assumes [intro]: S : a $\leftrightarrow$ c

assumes [intro]: S': a $\leftrightarrow$ c

assumes [intro]: R : b $\leftrightarrow$ c

assumes S[intro]: S $\sqsubseteq$ S'

shows (R $\vdash$ S) $\sqsubseteq$ (R $\vdash$ S')

lemma (**in** RestrRResSemi) restr-rrescom-incl-ohk:

assumes [intro]: S : a $\leftrightarrow$ c

assumes [intro]: S': a $\leftrightarrow$ c

assumes [intro]: Q : a $\leftrightarrow$ b

assumes S[intro]: S $\sqsubseteq$ S'

shows (S $\dashv$ Q) $\sqsubseteq$ (S' $\dashv$ Q)

For proposition 4.5 (i), with the assumptions *rang* $F \sqsubseteq$ *dom* R and *dom* $T \sqsubseteq$ *rang* Q, it holds for restricted right-residual. This has been proved in the following two lemmas for restricted left-residual and restricted right-residual, respectively.

lemma (**in** RestrLResSemi) restr-lrescom-incl-ex:

assumes [intro]:F : a $\leftrightarrow$ b

assumes [intro]:R : b $\leftrightarrow$ c

assumes [intro]:S : d $\leftrightarrow$ c

assumes FR: rang F $\sqsubseteq$ dom R

shows (F $\odot$ (S $\vdash$ R)) $\sqsubseteq$ (S $\vdash$ (F $\odot$ R))

lemma (**in** RestrRResSemi) Restr-rrescom-incl-ex:

assumes [intro]:U : a $\leftrightarrow$ b

assumes [intro]:Q : a $\leftrightarrow$ c

assumes [intro]:T : c $\leftrightarrow$ d

assumes TQ: dom T $\sqsubseteq$ rang Q

shows ((Q $\dashv$ U) $\odot$ T) $\sqsubseteq$ ((Q $\odot$ T) $\dashv$ U)

See appendix B for the proofs of the above newly introduced lemmas about the properties of standard residuals and restricted residuals.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The work completed for this thesis was to develop a framework by building a hierarchy of Isabelle/Isar theories to implement relational semigroupoid theories first presented by Kahl in [Kah08].

Basic category theories in appendix B of De Guzman's thesis [DG04] have been transformed into our semigroupoid theories such as *Semi*, *OrdSemi*,*OrdSemiBounds*, *SemiAllRecord*, *ISIdSemi*, *PreDomSemi* and *ResSemi*, by modifying definitions, reformulating theorems, deleting theorems which are closely related to identities and adding new theorems to help reprove many theorems involving identities in their proofs, in order to adapt them to our system (about 77 pages).

New definitions, new theorems and new theories are added to implement the theory of restricted residuals and its properties, as well as the properties of standard residual (about 46 pages).

- Three new theories *PreRanSemi*, *MonPreRanSemi*, *RanSemi* are added for range operator.
- A new theory *RDConvOrdSemi* is added to prove a number of theorems involving range, domain and converse for providing properties for the newly introduced concept — restricted residuals.
- A number of new properties have been introduced in the new theories, *RestrLResSemi*, *RestrRResSemi* and *RestrResOSGC*, for restricted residuals as well as in *LResSemi*, *RResSemi* and *RestrResOSGC* for standard residuals.
- New theory *RestrResAndRes* is provided to prove that in ordered semigroupoids with domain and range, if standard residual exists, restricted residual exists too.

7.2 Future Work

The following are a number of possible future extensions of our system.

- Implement semi-allegory theories [Kah08] which involves meet, converse and domain operators. Converse and domain operators, and their properties have been provided in the current system.
- Implement Kleene semigroupoid theories [Kah08].
- Define the restricted symmetric quotient [Kah08] in ordered semigroupoids with converse, based on the existing restricted left-residual and restricted right-residual theories in the system, and provide its properties.

Since our work is done by deploying a hierarchy of Isabelle/Isar theories and the hierarchy is based on Isabelle/Isar locales and Isabelle records, the above work can be easily implemented by extending our Isabelle records and Isabelle/Isar locales and "*import*" our theories.

Bibliography

[Bal07] Clemens Ballarin. Tutorial to locales and locale interpretation. Technical report tum-i0723, Technische Universitat München, 2007.

[BKS97] Chris Brink, Wolfram Kahl, and Gunther Schmidt, editors. *Relational Methods in Computer Science.* Advances in Computing Science. Springer, Wien, New York, 1997.

[BW99] Michael Barr and Charles Wells. *Category Theory for Computing Science.* Centre de recherches mathématiques (CRM), Université de Montréal, 3rd edition edition, 1999.

[DG04] Millie Rhoss De Guzman. Relation-algebraic proofs in Isabelle/Isar. Master's thesis, Department of Computing and Software, McMaster University, 2004.

[DMS03] J. Desharnais, B. Mölle, and G. Struth. Kleene algebra with domain. Technical Report 2003-7 92-92, Universität Augsburg,University of Augsbur, 2003.

[FK98] Hitoshi Furusawa and Wolfram Kahl. A study on symmetric quotients. Technical Report 1998-06, Fakultät für Informatik, Universität der Bundeswehr München, December 1998.

[Kah01] Wolfram Kahl. A relation-algebraic approach to graph structure transformation, 2001. Habil. Thesis, Fakultät für Informatik, Univ. der Bundeswehr München, Techn. Bericht 2002-03.

[Kah03] Wolfram Kahl. Calculational relation-algebraic proofs in Isabelle/Isar. In Rudolf Berghammer and Bernhard Möller, editors, *Relational and Kleene-Algebraic Methods in Computer Science*, LNCS 3051, pages 178–190. Springer, 2003.

[Kah08] Wolfram Kahl. Relational semigroupoids: Abstract relation-algebraic interfaces for finite relations between infinite types. *J. Log. Algebr. Program.*, 76(1):60–89, 2008.

[KWP99] Florian Kammüller, Markus Wenzel, and Lawrence C. Paulson. Locales — a sectioning concept for Isabelle. In Y. Bertot, G. Dowek, A. Hirschowitz,

C. Paulin, and L. Théry, editors, *Theorem Proving in Higher-Order Logics, 12th International Conference, TPHOLs'99*, volume 1690 of *LNCS*, pages 149–165. Springer, 1999.

[Nip07] Tobias Nipkow. A tutorial introduction to structured Isar proofs. Technical report, Institut für Informatik, TU München, 2007.

[NPW08] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*. Springer, 2008.

[SS93] Gunther Schmidt and Thomas Ströhlein. *Relations and Graphs, Discrete Mathematics for Computer Scientists*. EATCS-Monographs on Theoretical Computer Science. Springer, 1993.

[Wen08] Markus Wenzel. *The Isabelle/Isar Reference Manual*. TU München, 2008.

[Win01] M. Winter. A new algebraic approach to L-fuzzy relations convenient to study crispness. *Information Sciences*, 139(3-4):233–252, 2001.

Appendix A

Theory dependency Graph

In chapter 3, we gave an overview of our theory hierarchy. Here we provide a detailed graph of our theory collection. We also provide makers to classify our contribution on theories in relation to [Kah08] and [DG04].

- (*New definitions*) marks theories in which new definitions were provided. If the corresponding concepts exist in categories, it has been defined in a completely new way in our implementation based on our need.
- (*New theorems*) marks theories in which new theorems were added for helping reprove theorems during the process of transforming theories from category system to our system ,or for supporting further implementations, or for providing new properties.
- *New theories* marks new theories we added to the hierarchy (i.e., these theories are not present in [DG04] and [Kah03]). They are different from those theories which have been built by transforming theories from category system into semigroupoid system.

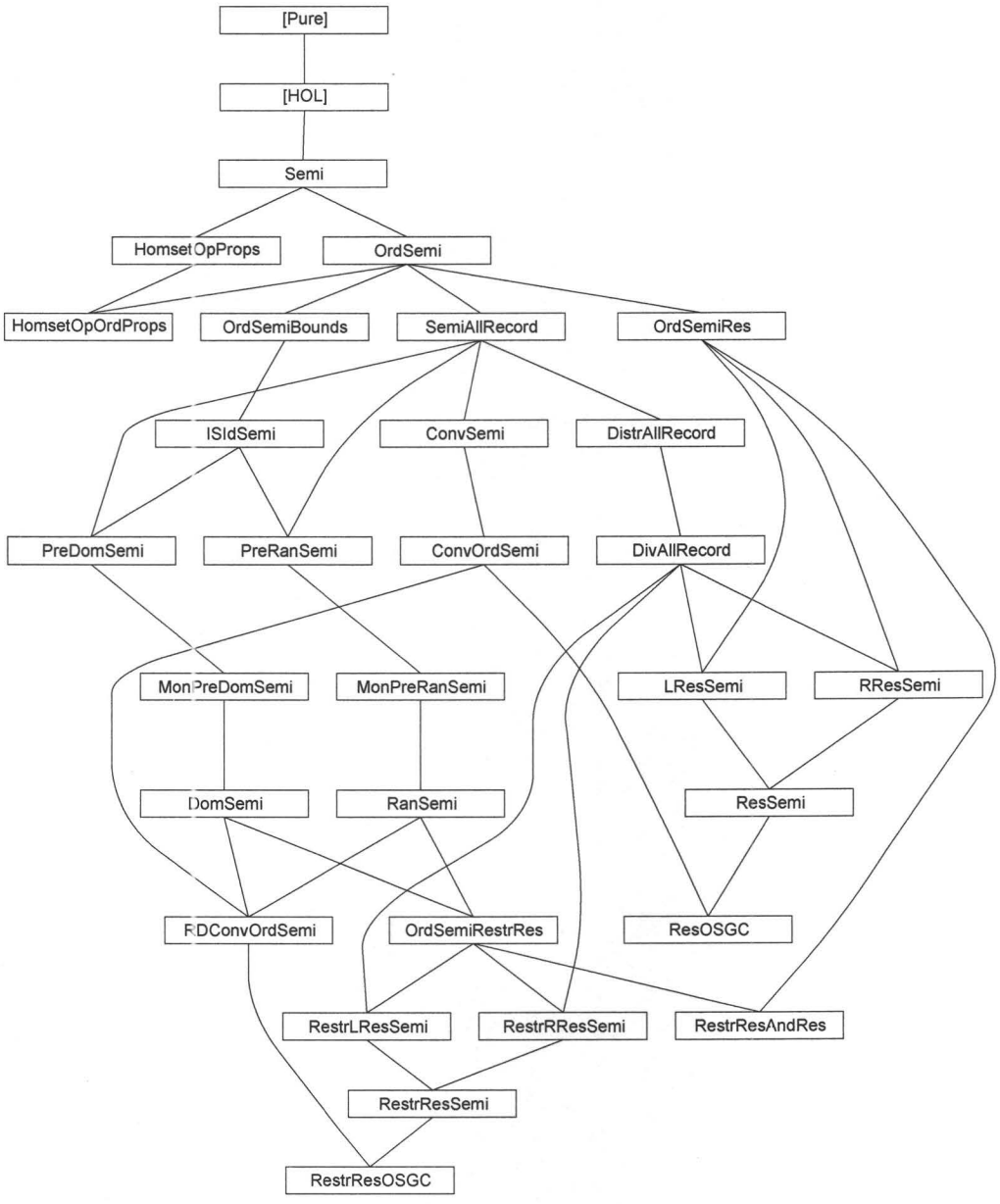

Figure A.1: Theory Dependency Graph

Appendix B

Proofs of Relational Semigroupoids in Isabelle/Isar

B.1 Semigroupoids

theory *Semi*
imports *Main*
begin

B.1.1 Basic Definitions

record $('o, 'm)$ *Semigroupoid* =
 isObj :: $'o \Rightarrow$ *bool* (Obj_1 - [*1000*] *999*)
 isMor :: $'m \Rightarrow$ *bool* (Mor_1 - [*1000*] *999*)
 cmp :: $'m \Rightarrow 'm \Rightarrow 'm$ (**infixr** $\odot_1$ *200*)
 Ssrc :: $'m \Rightarrow 'o$ (src_1 - [*1000*] *999*)
 Strg :: $'m \Rightarrow 'o$ (trg_1 - [*1000*] *999*)

constdefs
 homset :: $('o, 'm, 'r)$ *Semigroupoid-scheme* $\Rightarrow 'o \Rightarrow 'o \Rightarrow 'm$ *set* (**infixr** $\leftrightarrow_1$ *300*)
 homset s a b == $\{f$. *isObj s a* $\wedge$ *isObj s b* $\wedge$ *isMor s f* $\wedge$ *Ssrc s f* $= a \wedge$ *Strg s f* $= b\}$

locale *Semigroupoid* =
 fixes *C* :: $('o, 'm, 'r)$ *Semigroupoid-scheme* (**structure**)
 assumes *src-defined*[*intro?*,*simp*]: *Mor f* $\Longrightarrow$ *Obj* (*src f*)
 assumes *trg-defined*[*intro?*,*simp*]: *Mor f* $\Longrightarrow$ *Obj* (*trg f*)
 assumes *cmp-defined*[*intro?*,*simp*]: $[\![$ *Mor f*; *Mor g*; *trg f* $=$ *src g* $]\!] \Longrightarrow$ *Mor* $(f \odot g)$
 assumes *cmp-src*[*simp*]: $[\![$ *Mor f*; *Mor g*; *trg f* $=$ *src g* $]\!] \Longrightarrow$ *src* $(f \odot g)$ $=$ *src f*
 assumes *cmp-trg*[*simp*]: $[\![$ *Mor f*; *Mor g*; *trg f* $=$ *src g* $]\!] \Longrightarrow$ *trg* $(f \odot g)$ $=$ *trg g*

— All user-level laws use homset premises!

 assumes *cmp-assoc*[*simp*]: $[\![f : a \leftrightarrow b; g : b \leftrightarrow c; h : c \leftrightarrow d]\!]$
 $\Longrightarrow (f \odot g) \odot h = f \odot (g \odot h)$

B.1.2 Auxiliary Lemmas

lemmas (**in** *Semigroupoid*) *cmp-assoc-sym* $=$ *cmp-assoc* [*THEN sym*]

lemma (**in** *Semigroupoid*) *hs-def*:
 $a \leftrightarrow b = \{f$. *Obj* $a \wedge$ *Obj* $b \wedge$ *Mor* $f \wedge$ *src* $f = a \wedge$ *trg* $f = b\}$
by (*unfold homset-def*, *simp*)

For each constant definition, we will provide a user level lemma for readability.

lemma (**in** *Semigroupoid*) *homset*:
 fixes $a :: 'o$ **and** $b :: 'o$ **and** $f :: 'm$
 assumes *a*[*simp*]: *Obj a*
 assumes *b*[*simp*]: *Obj b*
 assumes *f*[*simp*]: *Mor f*
 assumes *src*[*simp*]: *src f* $= a$
 assumes *trg*[*simp*]: *trg f* $= b$
 shows $f : a \leftrightarrow b$
by (*simp add*: *hs-def*)

lemma (**in** *Semigroupoid*) *homset0*[*intro?*]:
 assumes *f* [*intro*,*simp*]: *Mor f*
 assumes [*intro*,*simp*]: *src f* $= a$
 assumes [*intro*,*simp*]: *trg f* $= b$
 shows $f : a \leftrightarrow b$
proof −
 from *f* **have** *Obj* (*src f*) **by** (*rule src-defined*)
 also from *f* **have** *Obj* (*trg f*) **by** (*rule trg-defined*)
 ultimately show *?thesis* **by** (*simp add* : *hs-def*)
qed

lemma (**in** *Semigroupoid*) *homset1*[*intro?*]:
 assumes *m*: *Mor f*
 shows $f : src\ f \leftrightarrow trg\ f$
proof (*rule homset0*)
 note *m*
next
 from *m* **show** *n*: *src f* $=$ *src f* **by** *simp*
next
 from *m* **show** *nn*:*trg f* $=$ *trg f* **by** *simp*
next

from m **show** $Mor\ f$ **by** *simp*
qed

lemma (**in** *Semigroupoid*) *homset-expand*[*dest?*]:
$f : a \mapsto b \Longrightarrow Obj\ a \wedge Obj\ b \wedge Mor\ f \wedge src\ f = a \wedge trg\ f = b$
by (*simp add*: *hs-def*)

lemma (**in** *Semigroupoid*) *homset-def1*:
$f : a \mapsto b = (Mor\ f \wedge src\ f = a \wedge trg\ f = b)$
proof (*rule iffI*)
assume $f : a \mapsto b$
thus $Mor\ f \wedge src\ f = a \wedge trg\ f = b$ **by** (*simp add*: *homset-expand*)
next
assume $Mor\ f \wedge src\ f = a \wedge trg\ f = b$
thus $f : a \mapsto b$ **by** (*simp add*: *homset0*)
qed

lemma (**in** *Semigroupoid*) *homset-srcObj*[*intro?*,*simp*]: $f : a \mapsto b \Longrightarrow Obj\ a$
by (*drule homset-expand*, *simp*)

lemma (**in** *Semigroupoid*) *homset-trgObj*[*intro?*,*simp*]: $f : a \mapsto b \Longrightarrow Obj\ b$
by (*drule homset-expand*, *simp*)

lemma (**in** *Semigroupoid*) *homset-Mor*[*intro*,*simp*]: $f : a \mapsto b \Longrightarrow Mor\ f$
by (*drule homset-expand*, *simp*)

lemma (**in** *Semigroupoid*) *homset-src*[*simp*]: $f : a \mapsto b \Longrightarrow src\ f = a$
by (*drule homset-expand*, *simp*)

lemma (**in** *Semigroupoid*) *homset-trg*[*simp*]: $f : a \mapsto b \Longrightarrow trg\ f = b$
by (*drule homset-expand*, *simp*)

lemma (**in** *Semigroupoid*) *cmp-homset*[*intro!*,*simp*]:
assumes f [*intro*,*simp*]: $f : a \mapsto b$
assumes g [*intro*,*simp*]: $g : b \mapsto c$
shows $(f \odot g) : a \mapsto c$
proof −
have [*intro*,*simp*]:$trg\ f = src\ g$
proof −

from f **have** $trg\ f = b$ **by** (*rule homset-trg*)
also from g **have** $src\ g = b$ **by** (*rule homset-src*)
ultimately show *?thesis* **by** *simp*
qed
have [*intro*,*simp*]: $Mor\ f$ **by** (*rule homset-Mor*,*best+*)
have [*intro*,*simp*]: $Mor\ g$ **by** (*rule homset-Mor*,*best+*)
have [*intro*,*simp*]: $Mor\ (f \odot g)$ **by** (*rule cmp-defined*, *simp-all*)
have $src\ (f \odot g) = a$
proof −
have $src\ (f \odot g) = src\ f$ **by** (*rule cmp-src*, *simp-all*)
also have $\ldots = a$ **by** (*rule homset-src*, *auto*)
ultimately show *?thesis* **by** *simp*
qed
also have $trg\ (f \odot g) = c$
proof −
have $trg\ (f \odot g) = trg\ g$ **by** (*rule cmp-trg*, *simp-all*)
also have $\ldots = c$ **by** (*rule homset-trg*, *auto*)
ultimately show *?thesis* **by** *simp*
qed
ultimately show *?thesis* **by** (*rule-tac homset0*, *auto*)
qed

The following two lemmas are moved here from LResSemi.thy and RResSemi.thy respectively. Because they are not actually related to standards residuals in their statements and their proofs. They in the theory can be share by all the theories which are entensions of semigoupoids.

lemma (**in** *Semigroupoid*) *target-eq*[*simp*]:
assumes *t*: $trg\ R = trg\ S$
assumes [*intro*]: $Mor\ R$
assumes [*intro*]: $Mor\ S$
shows $S : src\ S \mapsto trg\ R$
proof −
have $S : src\ S \mapsto trg\ S$ **by** (*rule homset1*, *best*)
also have $trg\ S = trg\ R$ **by** (*rule t* [*THEN sym*])
finally show *?thesis* .
qed

lemma (**in** *Semigroupoid*) *source-eq*[*simp*]:
assumes *s*: $src\ L = src\ S$
assumes [*intro*,*simp*]: $Mor\ L$

assumes [*intro,simp*]: *Mor S*
shows *S* ∈ *src L* ↔ *trg S*
proof −
have *S* : *src S* ↔ *trg S* **by** (*rule homset1, best*)
also have *src S* = *src L* **by** (*rule s* [*THEN sym*])
finally show *?thesis* .
qed

B.1.3 Derived Properties

In semigroupoids, idendities do not generally exist. Here we give the definition of idendites under some assumptions. The properities of idendities are not provided because idendities do not generally exist and we don't use them in our proofs.

constdefs
Semi-isId:: ('*o*, '*m*, '*r*) *Semigroupoid-scheme* ⇒'*m* ⇒ *bool* (*isId*₁ - [*1000*] *999*)
Semi-isId s i == *if* (*isMor s i* & *Ssrc s i* = *Strg s i*)
then (*let a* = *Ssrc s i in*
(∀ *b f g*.
f : *homset s a b* ⟶
g : *homset s b a* ⟶
(*cmp s i f* = *f* & *cmp s g i* = *g*)))
else arbitrary

lemma (**in** *Semigroupoid*) *isId-def*[*simp*]:
i : *a* ↔ *a* ⟹ *isId i* = (∀ *b f g*.
f : *a* ↔ *b* ⟶ *g* : *b* ↔ *a* ⟶ *i* ⊙ *f* = *f* & *g* ⊙ *i* = *g*)
by (*unfold Semi-isId-def,simp add*: *Let-def*)

lemma (**in** *Semigroupoid*) *homset-inj*:
assumes *i1*: *f* : *a* ↔ *b*
assumes *i2*: *f* : *c* ↔ *d*
shows *a* = *c* ∧ *b* = *d*
proof (*rule conjI*)
from *i1* **have** *src f* = *a* **by** (*rule homset-src*)
moreover from *i2* **have** *src f* = *c* **by** (*rule homset-src*)
ultimately show *a* = *c* **by** *simp*
next

from *i1* **have** *trg f* = *b* **by** (*rule homset-trg*)
moreover from *i2* **have** *trg f* = *d* **by** (*rule homset-trg*)
ultimately show *b* = *d* **by** *simp*
qed

B.1.4 Parallel Morphisms

constdefs
Semi-parallel :: ('*o*, '*m*, '*r*) *Semigroupoid-scheme* ⇒ '*m* ⇒ '*m* ⇒ *bool* (**infix** ∥₁ *200*)
Semi-parallel s f g == (*isMor s f* & *isMor s g* & *Ssrc s f* = *Ssrc s g* & *Strg s f* = *Strg s g*)

lemma (**in** *Semigroupoid*) *parallel-def*:
(*f* ∥ *g*) = (*Mor f* & *Mor g* & *src f* = *src g* & *trg f* = *trg g*)
by (*unfold Semi-parallel-def*, *simp*)

lemma (**in** *Semigroupoid*) *parallel-intro*[*intro*]:
assumes *T-f*[*intro*]: *f* : *a* ↔ *b*
assumes *T-g*[*intro*]: *g* : *a* ↔ *b*
shows *f* ∥ *g*
proof (*subst parallel-def*)
show *Mor f* ∧ *Mor g* ∧ *src f* = *src g* ∧ *trg f* = *trg g*
proof (*subst homset-src*)
from *T-f* **have** *f*: *Mor f* **by** (*simp*)
from *T-g* **have** *g*: *Mor g* **by** *simp*
from *T-f* **have** *f1*: *a* = *src f* & *b* = *trg f* **by** (*simp*)
from *T-g* **have** *g1*:*a* = *src g* & *b* = *trg g* **by** *simp*
from *f1 g1* **have** *fg*: *src f* = *src g* & *trg f* = *trg g* **by** *simp*
from *f g fg* **show** *?thesis* **by** *auto*
next
from *T-f* **have** *f2*: *src f* = *a* **by** (*rule-tac homset-src*, *simp*)
from *T-f f2* **show** *f*: *src f* ↔*b* **by** (*auto*)
qed
qed

lemma (**in** *Semigroupoid*) *parallel-intro-new*:
assumes *f-t*[*intro!,simp*]: *f* : *a* ↔ *b*
assumes *g-t*[*intro!,simp*]: *g* : *a* ↔ *b*
shows *f* ∥ *g*

proof (*subst parallel-def*, *intro conjI*)
show *Mor f* **by** (*rule homset-Mor*, *best+*)
next
show *Mor g* **by** (*rule homset-Mor*, *best+*)
next
have *src f* = *a* **by** (*rule homset-src*, *best+*)
moreover have *src g* = *a* **by** (*rule homset-src*, *best+*)
ultimately show *src f* = *src g* **by** *simp*
next
have *trg f* = *b* **by** (*rule homset-trg*, *best+*)
moreover have *trg g* = *b* **by** (*rule homset-trg*, *best+*)
ultimately show *trg f* = *trg g* **by** *simp*
qed

lemma (**in** *Semigroupoid*) *parallel-symmetric*:
assumes *l*[*intro!*]: *f* ∥ *g*
shows *g* ∥ *f*
proof (*rule parallel-def* [*THEN iffD2*])
from *l* **have** *Mor f* ∧ *Mor g* ∧ *src f* = *src g* ∧ *trg f* = *trg g*
by (*rule parallel-def* [*THEN iffD1*])
thus *Mor g* ∧ *Mor f* ∧ *src g* = *src f* ∧ *trg g* = *trg f* **by** *simp*
qed

Lemma *homset-eq* is used to prove a subgoal of Lemma *parallel*.

lemma (**in** *Semigroupoid*) *homset-eq*:
⟦ *Mor f*; *Mor g*; *src f* = *src g*; *trg f* = *trg g* ⟧ ⟹ *g* : *src f* ↔ *trg f*
by (*rule homset-def1* [*THEN iffD2*], *simp*)

lemma (**in** *Semigroupoid*) *parallel*[*dest?*]:
assumes *i1*: *f* ∥ *g*
shows ∃ *a b* . *f* : *a* ↔ *b* ∧ *g* : *a* ↔ *b*
apply (*rule-tac x*=*Ssrc C f* **in** *exI*)
apply (*rule-tac x*=*Strg C f* **in** *exI*)
proof (*rule conjI*)
from *i1* **have** *Mor f* ∧ *Mor g* ∧ *src f* = *src g* ∧ *trg f* = *trg g*
by (*rule parallel-def* [*THEN iffD1*])
from *this* **have** *Mor f* **by** *auto*
from *this* **show** *f* : *src f* ↔ *trg f* **by** (*simp add*: *homset1*)
next
from *i1* **have** *i2*: *Mor f* ∧ *Mor g* ∧ *src f* = *src g* ∧ *trg f* = *trg g*

by (*rule parallel-def* [*THEN iffD1*])
from *i2* **have** *Mor f* **by** *auto*
moreover from *i2* **have** *Mor g* **by** *auto*
moreover from *i2* **have** *src f* = *src g* **by** *auto*
moreover from *i2* **have** *trg f* = *trg g* **by** *auto*
ultimately show *g* : *src f* ↔ *trg f* **by** (*rule homset-eq*)
qed

lemma (**in** *Semigroupoid*) *parallel-1*[*dest?*,*intro?*]:
assumes *i1*: *f* ∥ *g*
assumes *i2*: *f* : *a* ↔ *b*
shows *g* : *a* ↔ *b*
proof (*rule homset-def1* [*THEN iffD2*], *intro conjI*)
from *i1* **have** *Mor f* ∧ *Mor g* ∧ *src f* = *src g* ∧ *trg f* = *trg g*
by (*rule parallel-def* [*THEN iffD1*])
thus *Mor g* **by** *simp*
next
from *i1* **have** *Mor f* ∧ *Mor g* ∧ *src f* = *src g* ∧ *trg f* = *trg g*
by (*rule parallel-def* [*THEN iffD1*])
then have *src f* = *src g* **by** *auto*
moreover from *i2* **have** *src f* = *a* **by** (*rule homset-src*)
ultimately show *src g* = *a* **by** *simp*
next
from *i1* **have** *Mor f* ∧ *Mor g* ∧ *src f* = *src g* ∧ *trg f* = *trg g*
by (*rule parallel-def* [*THEN iffD1*])
then have *trg f* = *trg g* **by** *simp*
moreover from *i2* **have** *trg f* = *b* **by** (*rule homset-trg*)
ultimately show *trg g* = *b* **by** *simp*
qed

lemma (**in** *Semigroupoid*) *parallel-2*[*dest?*,*intro?*]:
assumes *i1*: *f* ∥ *g*
assumes *i2*: *g* : *a* ↔ *b*
shows *f* : *a* ↔ *b*
proof −
from *i1* **have** *g* ∥ *f* **by** (*rule parallel-symmetric*)
with *i2* **show** *?thesis* **by** (*rule-tac parallel-1*)
qed

B.1.5 Special Morphisms

constdefs
Semi-isEpi :: ('o, 'm, 'r) *Semigroupoid-scheme* ⇒ 'm ⇒ *bool* (*isEpi*ı - [*1000*] *999*)
Semi-isEpi s f == *if isMor s f*
then (*let a* = *Ssrc s f*; *b* = *Strg s f in*
(∀ *c g h* .
g : *homset s b c* ⟶
h : *homset s b c* ⟶
(*cmp s f g* = *cmp s f h*) = (*g* = *h*)))
else arbitrary

lemma (**in** *Semigroupoid*) *epi-def*:
f : *a* ↔ *b* ⟹ *isEpi f* = (∀ *c g h* .
g : *b* ↔ *c* ⟶ *h* : *b* ↔ *c* ⟶ (*f* ⊙ *g* = *f* ⊙ *h*) = (*g* = *h*))
by (*unfold Semi-isEpi-def*, *simp add*: *Let-def*)

constdefs
Semi-isMono :: ('o, 'm, 'r) *Semigroupoid-scheme* ⇒ 'm ⇒ *bool* (*isMono*ı - [*1000*] *999*)
Semi-isMono s f == *if isMor s f*
then (*let b* = *Ssrc s f*; *c* = *Strg s f in*
(∀ *a g h* .
g : *homset s a b* ⟶
h : *homset s a b* ⟶
(*cmp s g f* = *cmp s h f*) = (*g* = *h*)))
else arbitrary

lemma (**in** *Semigroupoid*) *mono-def*:
f : *b* ↔ *c* ⟹ *isMono f* = (∀ *a g h* .
g : *a* ↔ *b* ⟶ *h* : *a* ↔ *b* ⟶ (*g* ⊙ *f* = *h* ⊙ *f*) = (*g* = *h*))
by (*unfold Semi-isMono-def*, *simp add*: *Let-def*)

B.1.6 Special Objects

constdefs
Semi-terminal :: ('o, 'm, 'r) *Semigroupoid-scheme* ⇒ 'o ⇒ *bool* (*terminal*ı - [*1000*] *999*)
Semi-terminal s t == *if isObj s t*
then (∀ *a* . (∃! *f* . *f* : *homset s a t*))
else arbitrary

lemma (**in** *Semigroupoid*) *terminal-def*:

Obj t ⟹ (*terminal t* = (∀ *a* . (∃! *f* . *f* : *a* ↔ *t*)))
by (*rule iffI*, *unfold Semi-terminal-def*, *simp-all*)

constdefs
Semi-initial :: ('o, 'm, 'r) *Semigroupoid-scheme* ⇒ 'o ⇒ *bool* (*initial*ı - [*1000*] *999*)
Semi-initial s i == *isObj s i* & (∀ *a* . (∃! *f* . *f* : *homset s i a*))

lemma (**in** *Semigroupoid*) *initial-def*:
Obj i ⟹ *initial i* = (∀ *a* . (∃! *f* . *f* : *i* ↔ *a*))
by (*rule iffI*, *unfold Semi-initial-def*, *simp-all*)

end

B.2 Properties of Operators on Homsets

theory *HomsetOpProps*
imports *Semi*
begin

B.2.1 Idempotence

constdefs
homset-idempotent :: ('o, 'm, 'r) *Semigroupoid-scheme* ⇒ ('m ⇒ 'm ⇒ 'm) ⇒ *bool*
(*hsIdempotent*ı - [*1000*] *999*)
homset-idempotent s f == *ALL m* . *isMor s m* ⟶ (*f m m* = *m*)

lemma (**in** *Semigroupoid*) *hsIdempotent-def*:
hsIdempotent f = (∀ *m*. *Mor m* ⟶ (*f m m* = *m*))
by (*rule iffI*, *unfold homset-idempotent-def*, *assumption*+)

lemma (**in** *Semigroupoid*) *hsIdempotent-intro*[*intro*]:
⟦ ⋀ *m* . ⟦ *Mor m* ⟧ ⟹ (*f m m* = *m*) ⟧ ⟹ *hsIdempotent f*
by (*subst hsIdempotent-def*, *simp*)

lemma (**in** *Semigroupoid*) *hsIdempotent-intro-new*:
assumes *i1*: (⋀ *m* . ⟦ *Mor m* ⟧ ⟹ (*f m m* = *m*))
shows *hsIdempotent f*
proof (*subst hsIdempotent-def*, *intro strip*)

from *i1* **show** $\bigwedge m.\ Mor\ m \Longrightarrow f\ m\ m = m$ **by** *simp*
qed

lemma (in *Semigroupoid*) *hsIdempotent-intro2[intro]*:
$[\![\bigwedge a\ b\ m\ .\ [\![m : a \mapsto b]\!] \Longrightarrow f\ m\ m = m]\!] \Longrightarrow hsIdempotent\ f$
apply (*rule hsIdempotent-intro-new*)
apply (*drule homset1*)
apply (*simp*)
done

lemma (in *Semigroupoid*) *hsIdempotent-intro2-new*:
assumes *i1*: $\bigwedge a\ b\ m\ .\ [\![m : a \mapsto b]\!] \Longrightarrow f\ m\ m = m$
shows *hsIdempotent f*
proof (*rule hsIdempotent-intro-new*, *drule homset1*)
from *i1* **show** $\bigwedge m.\ m \in src\ m \mapsto trg\ m \Longrightarrow f\ m\ m = m$ **by** *simp*
qed

lemma (in *Semigroupoid*) *hsIdempotent[simp]*:
$[\![hsIdempotent\ f;\ m : a \mapsto b]\!] \Longrightarrow f\ m\ m = m$
by (*drule hsIdempotent-def* [*THEN iffD1*], *simp*)

lemma (in *Semigroupoid*) *hsIdempotent-new*:
assumes *i1*: *hsIdempotent f*
assumes *i2*: $m : a \mapsto b$
shows $f\ m\ m = m$
proof −
from *i1* **have** $\forall m.\ Mor\ m \longrightarrow f\ m\ m = m$ **by** (*rule hsIdempotent-def* [*THEN iffD1*])
with *i2* **show** *?thesis* **by** *simp*
qed

B.2.2 Commutativity

Need to redefine some *constdefs* without using the concept of parallel morphisms. This will enable us to provide more structured proofs. We will follow definitions of *OS-isLBound* and *OS-isUBound* in OrdSemiBounds.thy

constdefs
homset-commutative :: $('o, 'm, 'r)$ *Semigroupoid-scheme* $\Rightarrow$ $('m \Rightarrow 'm \Rightarrow 'm) \Rightarrow$ *bool*
(*hsCommutative*₁ - [*1000*] *999*)
homset-commutative s f == $\forall\ m\ n\ .$ *Semi-parallel s m n* $\longrightarrow$ $(f\ m\ n = f\ n\ m)$


```
"homset_commutative s f == ALL m n . Semi_parallel s m n
                           \<longrightarrow> (f m n = f n m)"
```

The commented definition given above uses the parallel definition to illustrate that m and n have the same homset. The problem with this definition is that proofs of lemmas about commutativity are not well- structured.

Hence we provide a simpler definition of hsCommutative without the use of parallel. This lemma looks just like the definition provided above but with the expansion of parallel.

```
"homset_commutative s f == (ALL m n .(let a = Ssrc s m;
 b = Strg s m; c = Ssrc s n; d = Strg s n in isMor s m
\<and> isMor s n \<and> a = c \<and> b = d \<longrightarrow>
(f m n = f n m) ))"
```

The following definition is slightly shorter, more readable, but is harder to show than the one above. Problem is the presence implication instead of conjuctions. For now, I will stick with the expansion parallel as described above.

```
"homset_commutative s f == (ALL m n . isMor s m
\<longrightarrow> (let a = Ssrc s m; b = Strg s m in
n : homset s a b \<longrightarrow> (f m n = f n m) ))"
```

But adapting the second definition from above for *hsCommutative* did not improve the structure of the proof. Hence, we stick with the old definition.

lemma (in *Semigroupoid*) *hsCommutative-def*:
hsCommutative f = $(\forall\ m\ n.\ m \parallel n \longrightarrow (f\ m\ n = f\ n\ m))$
by (*rule iffI*, *unfold homset-commutative-def*, *assumption*+)

lemma (in *Semigroupoid*) *hsCommutative-intro[intro]*:
$[\![\bigwedge m\ n\ .\ [\![m \parallel n]\!] \Longrightarrow (f\ m\ n = f\ n\ m)]\!] \Longrightarrow hsCommutative\ f$
apply (*rule hsCommutative-def* [*THEN iffD2*])
apply *auto*
done

lemma (in *Semigroupoid*) *hsCommutative-intro-new*:
assumes *i1*: $\bigwedge m\ n\ .\ [\![\ m \parallel n\ ]\!] \Longrightarrow (f\ m\ n = f\ n\ m)$
shows *hsCommutative f*
proof (*subst hsCommutative-def*, *auto*)
from *i1* **show** $\bigwedge m\ n.\ m \parallel n \Longrightarrow f\ m\ n = f\ n\ m$ **by** *auto*
qed

lemma (in *Semigroupoid*) *hsCommutative-intro2*[*intro*]:
$[\![\bigwedge a\ b\ m\ n\ .\ [\![\ m : a \leftrightarrow b;\ n : a \leftrightarrow b\ ]\!] \Longrightarrow f\ m\ n = f\ n\ m\ ]\!] \Longrightarrow$ *hsCommutative f*
apply (*rule hsCommutative-intro-new*)
apply (*drule parallel*)
apply (*erule exE*)+
apply (*erule conjE*)
apply *simp*
done

lemma (in *Semigroupoid*) *hsCommutative-intro2-new*:
assumes *comm*: $\bigwedge a\ b\ m\ n\ .\ [\![\ m : a \leftrightarrow b;\ n : a \leftrightarrow b\ ]\!] \Longrightarrow f\ m\ n = f\ n\ m$
shows *hsCommutative f*
proof (*rule hsCommutative-intro-new*, *drule parallel*)
from *comm* **show** $\bigwedge m\ n.\ \exists a\ b.\ m \in a \leftrightarrow b \wedge n \in a \leftrightarrow b \Longrightarrow f\ m\ n = f\ n\ m$ **by** *auto*
qed

the proof above is better compare to the following:

lemma (in *Semigroupoid*) *hsCommutative-intro2-old*:
assumes *i1*[*intro*, *simp*]: $\bigwedge a\ b\ m\ n\ .\ [\![\ m : a \leftrightarrow b;\ n : a \leftrightarrow b\ ]\!] \Longrightarrow f\ m\ n = f\ n\ m$
shows *hsCommutative f*
proof (*subst hsCommutative-def*, *intro strip*, *drule parallel*, (*erule exE*)+, *auto*)
have $\bigwedge m\ n\ a\ b.\ m \in a \leftrightarrow b \wedge n \in a \leftrightarrow b \Longrightarrow f\ m\ n = f\ n\ m$ **by** *auto*
qed

The new proof is using *hsCommutative-intro-new* instead of *hsCommutative-def*. This change improved the proof; we had to apply two methods back instead of five methods.

lemma (in *Semigroupoid*) *hsCommutative*:
$[\![$ *hsCommutative f*; $m : a \leftrightarrow b$; $n : a \leftrightarrow b\ ]\!] \Longrightarrow f\ m\ n = f\ n\ m$
by (*drule hsCommutative-def* [*THEN iffD1*], *auto*)

lemma (in *Semigroupoid*) *hsCommutative-new*:
assumes *i1*: *hsCommutative f*
assumes *m-t*[*intro*]: $m : a \leftrightarrow b$
assumes *n-t*[*intro*]: $n : a \leftrightarrow b$
shows $f\ m\ n = f\ n\ m$
proof −
from *i1* **have** $\forall\ m\ n.\ m \parallel n \longrightarrow (f\ m\ n = f\ n\ m)$ **by** (*rule hsCommutative-def* [*THEN iffD1*])
with *m-t n-t* **show** *?thesis* **by** *auto*
qed

B.2.3 Associativity

Looking at the proofs of the derived lemmas below for associativity of homsets, we see that the proof for *intro2-new* is not well-structured. We will try to give another definition of *homset-associative* that does not involve parallel. This new definition with the expansion of parallel is suppose to be simpler which would make *hsAssoc-intro2-new* proof nicer.

constdefs
homset-associative :: ('o, 'm, 'r) *Semigroupoid-scheme* ⇒ ('m ⇒ 'm ⇒ 'm) ⇒ *bool*
(*hsAssoc*ı - [*1000*] *999*)
homset-associative s f == *ALL m n p . isMor s m & isMor s n & isMor s p*
& (let a = Ssrc s n; b = Strg s n in
Ssrc s m = a & Strg s m = b & a = Ssrc s p & b = Strg s p)
⟶ (f (f m n) p = f m (f n p))

lemma (in *Semigroupoid*) *hsAssoc-def*:
hsAssoc f = (∀ *m n p. Mor m* ∧ *Mor n* ∧ *Mor p* ∧ *src m = src n* ∧ *trg m = trg n*
∧ *src n = src p* ∧ *trg n = trg p* ⟶ *(f (f m n) p = f m (f n p)))*
by (*rule iffI*, *unfold homset-associative-def*, *auto simp add*: *Let-def*)

lemma (in *Semigroupoid*) *hsAssoc-intro*[*intro*]:
$[\![\bigwedge m\ n\ p\ .\ [\![$ *Mor m*; *Mor n*; *Mor p*; *src m = src n*; *trg m = trg n*;
src n = src p; *trg n = trg p* $]\!] \Longrightarrow$ *(f (f m n) p = f m (f n p))* $]\!]$
$\Longrightarrow$ *hsAssoc f*
apply (*subst hsAssoc-def*)
apply (*intro strip*)
apply (*erule conjE*)+
apply *simp*

done

lemma (**in** *Semigroupoid*) *hsAssoc-intro-new*:
assumes *i1*[*intro*, *simp*]: $\bigwedge m\ n\ p\ .\ [\![\ Mor\ m;\ Mor\ n;\ Mor\ p;\ src\ m = src\ n;\ trg\ m = trg\ n;$
$src\ n = src\ p;\ trg\ n = trg\ p\ ]\!] \Longrightarrow (f\ (f\ m\ n)\ p = f\ m\ (f\ n\ p))$
shows *hsAssoc f*
by (*subst hsAssoc-def*, *intro strip*, (*erule conjE*)+, *auto*)

lemma (**in** *Semigroupoid*) *hsAssoc-intro2*[*intro*]:
assumes *i4*: $\bigwedge a\ b\ m\ n\ p\ .\ [\![\ m : a \mapsto b;\ n : a \mapsto b;\ p : a \mapsto b\ ]\!] \Longrightarrow$
$f\ (f\ m\ n)\ p = f\ m\ (f\ n\ p)$
shows *hsAssoc f*
proof (*rule hsAssoc-intro*)
fix *m* **and** *n* **and** *p*
assume *m*[*intro*, *simp*]: *Mor m*
assume *n*[*intro*, *simp*]: *Mor n*
assume *p*[*intro*, *simp*]: *Mor p*
assume *s-mn* [*intro*, *simp*]: *src m* = *src n*
assume *s-np* [*intro*, *simp*]: *src n* = *src p*
assume *t-mn* [*intro*, *simp*]: *trg m* = *trg n*
assume *t-np* [*intro*, *simp*]: *trg n* = *trg p*
note *i4*
moreover have $m : src\ p \mapsto trg\ p$ **by** (*rule-tac homset0*, *auto*)
moreover have $n : src\ p \mapsto trg\ p$ **by** (*rule-tac homset0*, *best*+)
moreover have $p : src\ p \mapsto trg\ p$ **by** (*rule-tac homset0*, *auto*)
ultimately show $f\ (f\ m\ n)\ p = f\ m\ (f\ n\ p)$ **by** *best*+
qed

lemma (**in** *Semigroupoid*) *hsAssoc*[*simp*]:
$[\![\ hsAssoc\ f;\ m : a \mapsto b;\ n : a \mapsto b;\ p : a \mapsto b\ ]\!] \Longrightarrow f\ (f\ m\ n)\ p = f\ m\ (f\ n\ p)$
by (*drule hsAssoc-def* [*THEN iffD1*], *auto*)

lemma (**in** *Semigroupoid*) *hsAssoc-new*:
assumes *i1*: *hsAssoc f*
assumes *m-t*: $m : a \mapsto b$
assumes *n-t*: $n : a \mapsto b$
assumes *p-t*: $p : a \mapsto b$

shows $f\ (f\ m\ n)\ p = f\ m\ (f\ n\ p)$
proof −
from *i1* **have** $\forall\ m\ n\ p.\ Mor\ m \wedge Mor\ n \wedge Mor\ p \wedge src\ m = src\ n \wedge trg\ m = trg\ n$
$\wedge\ src\ n = src\ p \wedge trg\ n = trg\ p \longrightarrow f\ (f\ m\ n)\ p = f\ m\ (f\ n\ p)$
by (*rule hsAssoc-def* [*THEN iffD1*])
with *m-t n-t p-t* **show** *?thesis* **by** *auto*
qed

To make the proof for *hsAssoc-intro2-new* well-structured, we expanded the parallel definition in the underlying *homset-associative-def*.

B.2.4 Totality

constdefs
homset-totalBinOp :: ('*o*, '*m*, '*r*) *Semigroupoid-scheme* ⇒ ('*m* ⇒ '*m* ⇒ '*m*) ⇒ *bool*
(*hsBinTotal*ı - [*1000*] *999*)
homset-totalBinOp s f == *ALL m n* . *Semi-parallel s m n* ⟶ *Semi-parallel s m* (*f m n*)

lemma (**in** *Semigroupoid*) *hsBinTotal-def*:
$hsBinTotal\ f = (\forall\ m\ n.\ m \parallel n \longrightarrow m \parallel (f\ m\ n))$
by (*rule iffI*, *unfold homset-totalBinOp-def*, *assumption*+)

lemma (**in** *Semigroupoid*) *hsBinTotal-intro*[*intro*]:
$[\![\ \bigwedge m\ n\ .\ [\![\ m \parallel n\ ]\!] \Longrightarrow m \parallel (f\ m\ n)\ ]\!] \Longrightarrow hsBinTotal\ f$
by (*subst hsBinTotal-def*, (*rule allI*)+, *rule impI*, *simp*)

lemma (**in** *Semigroupoid*) *hsBinTotal-intro-new*:
assumes *i1*: $\bigwedge m\ n\ .\ [\![\ m \parallel n\ ]\!] \Longrightarrow m \parallel (f\ m\ n)$
shows *hsBinTotal f*
proof (*subst hsBinTotal-def*, (*rule allI*)+, *rule impI*)
from *i1* **have** *?thesis* **by** (*auto*)
qed

lemma (**in** *Semigroupoid*) *hsBinTotal-intro2*[*intro*]:
$[\![\ \bigwedge a\ b\ m\ n\ .\ [\![\ m : a \mapsto b;\ n : a \mapsto b\ ]\!] \Longrightarrow f\ m\ n : a \mapsto b\ ]\!] \Longrightarrow hsBinTotal\ f$
apply (*rule hsBinTotal-intro*)
apply (*drule parallel*)
apply (*erule exE*, *erule exE*, *erule conjE*)
apply (*rule parallel-intro*, *assumption*, *simp*)

done

lemma (**in** *Semigroupoid*) *hsBinTotal-intro2-new*:
assumes *i1*: $\bigwedge a\ b\ m\ n\ .\ [\![\ m : a \leftrightarrow b;\ n : a \leftrightarrow b\ ]\!] \Longrightarrow f\ m\ n : a \leftrightarrow b$
shows *hsBinTotal f*
proof (*rule hsBinTotal-intro, drule parallel*)
from *i1* **show** $\bigwedge m\ n.\ \exists a\ b.\ m \in a \leftrightarrow b \wedge n \in a \leftrightarrow b \Longrightarrow m \parallel f\ m\ n$ **by** *auto*
qed

lemma (**in** *Semigroupoid*) *hsBinTotal*[*intro?*,*simp*]:
$[\![\ hsBinTotal\ f;\ m : a \leftrightarrow b;\ n : a \leftrightarrow b\ ]\!] \Longrightarrow f\ m\ n : a \leftrightarrow b$
apply (*drule hsBinTotal-def* [*THEN iffD1*])
apply (*frule-tac f*=*m* **and** *g*=*n* **in** *parallel-intro, assumption*)
apply (*drule-tac x*=*m* **in** *spec*)
apply (*drule-tac x*=*n* **in** *spec*)
apply *simp*
apply (*erule parallel-1, assumption*)
done

lemma (**in** *Semigroupoid*) *hsBinTotal-new*:
assumes *i1*: *hsBinTotal f*
assumes *m-t*: $m : a \leftrightarrow b$
assumes *n-t*: $n : a \leftrightarrow b$
shows $f\ m\ n : a \leftrightarrow b$
proof −
from *i1* **have** $\forall\ m\ n.\ m \parallel n \longrightarrow m \parallel (f\ m\ n)$ **by** (*rule hsBinTotal-def* [*THEN iffD1*])
with *m-t n-t* **have** *i2*: $m \parallel (f\ m\ n)$ **by** *auto*
from *i2 m-t* **show** *?thesis* **by** (*rule parallel-1*)
qed

The Isar proof is better than the tactic proof

The use of *with m-t* instead of *from i2 m-t* above will not work, apparently because the order of the assumptions matters.

lemma (**in** *Semigroupoid*) *hsBinTotal-Mor*[*intro?*,*simp*]:
$[\![\ hsBinTotal\ f;\ m : a \leftrightarrow b;\ n : a \leftrightarrow b\ ]\!] \Longrightarrow Mor\ (f\ m\ n)$
by (*drule hsBinTotal, auto*)

lemma (**in** *Semigroupoid*) *hsBinTotal-Mor-new*:
assumes *i1*: *hsBinTotal f*
assumes *m-t*: $m : a \leftrightarrow b$
assumes *n-t*: $n : a \leftrightarrow b$
shows *Mor* $(f\ m\ n)$
proof −
from *i1 m-t n-t* **have** $f\ m\ n : a \leftrightarrow b$ **by** (*rule hsBinTotal*)
thus *?thesis* **by** (*rule homset-Mor*)
qed

lemma (**in** *Semigroupoid*) *hsBinTotal-src*[*simp*]:
$[\![\ hsBinTotal\ f;\ m : a \leftrightarrow b;\ n : a \leftrightarrow b\ ]\!] \Longrightarrow src\ (f\ m\ n) = a$
by (*drule hsBinTotal, auto*)

lemma (**in** *Semigroupoid*) *hsBinTotal-src-new*:
assumes *i1*: *hsBinTotal f*
assumes *m-t*: $m : a \leftrightarrow b$
assumes *n-t*: $n : a \leftrightarrow b$
shows $src\ (f\ m\ n) = a$
proof −
from *i1 m-t n-t* **have** $f\ m\ n : a \leftrightarrow b$ **by** (*rule hsBinTotal*)
thus *?thesis* **by** (*rule homset-src*)
qed

lemma (**in** *Semigroupoid*) *hsBinTotal-trg*[*simp*]:
$[\![\ hsBinTotal\ f;\ m : a \leftrightarrow b;\ n : a \leftrightarrow b\ ]\!] \Longrightarrow trg\ (f\ m\ n) = b$
by (*drule hsBinTotal, auto*)

lemma (**in** *Semigroupoid*) *hsBinTotal-trg-new*:
assumes *i1*: *hsBinTotal f*
assumes *m-t*: $m : a \leftrightarrow b$
assumes *n-t*: $n : a \leftrightarrow b$
shows $trg\ (f\ m\ n) = b$
proof −
from *i1 m-t n-t* **have** $f\ m\ n : a \leftrightarrow b$ **by** (*rule hsBinTotal*)
thus *?thesis* **by** (*rule homset-trg*)
qed

B.2.5 Collections of Properties

constdefs
homset-AC :: $('o, 'm, 'r)$ *Semigroupoid-scheme* $\Rightarrow$ $('m \Rightarrow 'm \Rightarrow 'm) \Rightarrow$ *bool*

(*hsAC*$_1$ - [*1000*] *999*)
homset-AC s f == *homset-associative s f* & *homset-commutative s f*

lemma (**in** *Semigroupoid*) *hsAC-def*: *hsAC f* = (*hsAssoc f* ∧ *hsCommutative f*)
by (*rule iffI*, *unfold homset-AC-def*, *assumption*+)

lemma (**in** *Semigroupoid*) *hsAC-intro*[*intro*]:
⟦ *hsAssoc f*; *hsCommutative f* ⟧ ⟹ *hsAC f*
by (*subst hsAC-def*, *rule conjI*)

lemma (**in** *Semigroupoid*) *hsAC-intro-new*:
assumes *i1*: *hsAssoc f*
assumes *i2*: *hsCommutative f*
shows *hsAC f*
proof (*subst hsAC-def*)
from *i1 i2* **show** *hsAssoc f* ∧ *hsCommutative f* ..
qed

lemma (**in** *Semigroupoid*) *hsAC-A*[*simp*]:
⟦ *hsAC f* ⟧ ⟹ *hsAssoc f*
by (*drule hsAC-def* [*THEN iffD1*], *erule conjE*)

lemma (**in** *Semigroupoid*) *hsAC-A-new*:
assumes *i1*: *hsAC f*
shows *hsAssoc f*
proof −
from *i1* **have** *hsAssoc f* ∧ *hsCommutative f* **by** (*rule hsAC-def* [*THEN iffD1*])
thus *?thesis* ..
qed

lemma (**in** *Semigroupoid*) *hsAC-C*[*simp*]:
⟦ *hsAC f* ⟧ ⟹ *hsCommutative f*
by (*drule hsAC-def* [*THEN iffD1*], *erule conjE*)

lemma (**in** *Semigroupoid*) *hsAC-C-new*:
assumes *i1*: *hsAC f*
shows *hsCommutative f*
proof −
from *i1* **have** *hsAssoc f* ∧ *hsCommutative f* **by** (*rule hsAC-def* [*THEN iffD1*])
thus *?thesis* ..

qed

constdefs
homset-ACI :: ('*o*, '*m*, '*r*) *Semigroupoid-scheme* ⇒ ('*m* ⇒ '*m* ⇒ '*m*) ⇒ *bool*
(*hsACI*$_1$ - [*1000*] *999*)
homset-ACI s f == *homset-AC s f* & *homset-idempotent s f*

lemma (**in** *Semigroupoid*) *hsACI-def*: *hsACI f* = (*hsAC f* ∧ *hsIdempotent f*)
by (*rule iffI*, *unfold homset-ACI-def*, *assumption*+)

lemma (**in** *Semigroupoid*) *hsACI-intro*[*intro*]:
⟦ *hsAC f*; *hsIdempotent f* ⟧ ⟹ *hsACI f*
by (*subst hsACI-def*, *simp*)

lemma (**in** *Semigroupoid*) *hsACI-intro-new*:
assumes *i1*: *hsAC f*
assumes *i2*: *hsIdempotent f*
shows *hsACI f*
proof −
from *i1 i2* **show** *?thesis* **by** (*subst hsACI-def*, *rule conjI*)
qed

lemma (**in** *Semigroupoid*) *hsACI-AC*[*simp*]:
⟦ *hsACI f* ⟧ ⟹ *hsAC f*
by (*drule hsACI-def* [*THEN iffD1*], *erule conjE*)

lemma (**in** *Semigroupoid*) *hsACI-AC-new*:
assumes *i1*: *hsACI f*
shows *hsAC f*
proof −
from *i1* **have** *hsAC f* ∧ *hsIdempotent f* **by** (*rule hsACI-def* [*THEN iffD1*])
thus *?thesis* ..
qed

lemma (**in** *Semigroupoid*) *hsACI-I*[*simp*]:
⟦ *hsACI f* ⟧ ⟹ *hsIdempotent f*
by (*drule hsACI-def* [*THEN iffD1*], *erule conjE*)

lemma (**in** *Semigroupoid*) *hsACI-I-new*:
assumes *i1*: *hsACI f*

shows *hsIdempotent f*
proof −
from *i1* **have** *hsAC f* ∧ *hsIdempotent f* **by** (*rule hsACI-def* [*THEN iffD1*])
thus *?thesis* ..
qed

constdefs
homset-TACI :: ('*o*, '*m*, '*r*) *Semigroupoid-scheme* ⇒ ('*m* ⇒ '*m* ⇒ '*m*) ⇒ *bool*
(*hsTACI*₁ - [*1000*] *999*)
homset-TACI s f == *homset-ACI s f* & *homset-totalBinOp s f*

lemma (**in** *Semigroupoid*) *hsTACI-def*: *hsTACI f* = (*hsACI f* ∧ *hsBinTotal f*)
by (*rule iffI*, *unfold homset-TACI-def*, *assumption*+)

lemma (**in** *Semigroupoid*) *hsTACI-intro*[*intro*]:
⟦ *hsBinTotal f*; *hsACI f* ⟧ ⟹ *hsTACI f*
by (*subst hsTACI-def*, *rule conjI*)

lemma (**in** *Semigroupoid*) *hsTACI-intro-new*:
assumes *i1*: *hsBinTotal f*
assumes *i2*: *hsACI f*
shows *hsTACI f*
proof −
from *i2 i1* **show** *?thesis* **by** (*subst hsTACI-def*, *rule conjI*)
qed

Note that in the above proof, "from i1 i2" fails.

Instead of "rule conjI" "auto" will also work."

lemma (**in** *Semigroupoid*) *hsTACI-ACI*[*simp*]:
⟦ *hsTACI f* ⟧ ⟹ *hsACI f*
by (*drule hsTACI-def* [*THEN iffD1*], *erule conjE*)

lemma (**in** *Semigroupoid*) *hsTACI-ACI-new*:
assumes *i1*: *hsTACI f*
shows *hsACI f*
proof −
from *i1* **have** *hsACI f* ∧ *hsBinTotal f* **by** (*rule hsTACI-def* [*THEN iffD1*])
thus *?thesis* ..
qed

lemma (**in** *Semigroupoid*) *hsTACI-T*[*simp*]:
⟦ *hsTACI f* ⟧ ⟹ *hsBinTotal f*
by (*drule hsTACI-def* [*THEN iffD1*], *erule conjE*)

lemma (**in** *Semigroupoid*) *hsTACI-T-new*:
assumes *i1*: *hsTACI f*
shows *hsBinTotal f*
proof −
from *i1* **have** *hsACI f* ∧ *hsBinTotal f* **by** (*rule hsTACI-def* [*THEN iffD1*])
thus *?thesis* ..
qed

B.2.6 Self-Distributivity

lemma (**in** *Semigroupoid*) *hsSelfDistr1*[*simp*]:
assumes *TACI*[*simp*]: *hsTACI f*
assumes *T-m*[*intro,simp*]: *m* : *a* ↔ *b*
assumes *T-n*[*intro,simp*]: *n* : *a* ↔ *b*
assumes *T-p*[*intro,simp*]: *p* : *a* ↔ *b*
shows *f m* (*f n p*) = *f* (*f m n*) (*f m p*)
proof −
have *f m* (*f n p*) = *f* (*f m m*) (*f n p*)
by (*subst hsIdempotent* [*of f m a b*, *THEN sym*], *auto*)

also have ... = *f m* (*f m* (*f n p*))
by (*rule-tac hsAssoc*, *auto*)
also have ... = *f m* (*f* (*f m n*) *p*)
by (*subst hsAssoc* [*of f*, *THEN sym*], *auto*)
also have ... = *f* (*f m* (*f m n*)) *p*
by (*rule-tac hsAssoc* [*THEN sym*], *auto*)
also have ... = *f* (*f* (*f m n*) *m*) *p*
by (*subst hsCommutative* [*of f*], *auto*)
also have ... = *f* (*f m n*) (*f m p*)
by (*rule-tac hsAssoc*, *auto*)
finally show *?thesis* .
qed

lemma (**in** *Semigroupoid*) *hsSelfDistr2*[*simp*]:

assumes *TACI*[*simp*]: *hsTACI f*
assumes *T-m*[*intro,simp*]: $m : a \leftrightarrow b$
assumes *T-n*[*intro,simp*]: $n : a \leftrightarrow b$
assumes *T-p*[*intro,simp*]: $p : a \leftrightarrow b$
shows $f\ (f\ n\ p)\ m = f\ (f\ n\ m)\ (f\ p\ m)$
proof −
have $f\ (f\ n\ p)\ m = f\ (f\ n\ p)\ (f\ m\ m)$
by (*subst hsIdempotent* [*of f m a b*, *THEN sym*], *auto*)
also have $\ldots = f\ (f\ (f\ n\ p)\ m)\ m$
by (*rule-tac hsAssoc* [*THEN sym*], *auto*)
also have $\ldots = f\ (f\ n\ (f\ p\ m))\ m$
by (*subst hsAssoc* [*of f*], *auto*)
also have $\ldots = f\ n\ (f\ (f\ p\ m)\ m)$
by (*rule-tac hsAssoc*, *auto*)
also have $\ldots = f\ n\ (f\ m\ (f\ p\ m))$
by (*subst hsCommutative* [*of f*], *auto*)
also have $\ldots = f\ (f\ n\ m)\ (f\ p\ m)$
by (*rule-tac hsAssoc* [*THEN sym*], *auto*)
finally show *?thesis* .
qed

lemmas (**in** *Semigroupoid*) *hsSelfDistr* = *hsSelfDistr1 hsSelfDistr2*

end

B.3 Ordered Semigroupoids: Inclusion Relation

theory *OrdSemi*
imports *Semi*
begin

record $('o, 'm)$ *OrderedSemigroupoid* = $('o, 'm)$ *Semigroupoid* +
incl :: $'m \Rightarrow 'm \Rightarrow$ *bool* (**infixr** $\sqsubseteq_1$ *50*)

locale *OrderedSemigroupoid* = *Semigroupoid OS* +
assumes *incl-refl*[*intro,simp*]: *Mor f* $\Longrightarrow f \sqsubseteq f$
assumes *incl-trans*[*trans*]:
$[\![f \sqsubseteq g;\ g \sqsubseteq h;\ f : a \leftrightarrow b;\ g : a \leftrightarrow b;\ h : a \leftrightarrow b\ ]\!] \Longrightarrow f \sqsubseteq h$

assumes *incl-antisym*[*trans*]:
$[\![f \sqsubseteq g;\ g \sqsubseteq f;\ f : a \leftrightarrow b;\ g : a \leftrightarrow b\ ]\!] \Longrightarrow f = g$
assumes *comp-incl-mon*[*intro,simp*]:
$[\![f \sqsubseteq f';\ g \sqsubseteq g';\ f : a \leftrightarrow b;\ f' : a \leftrightarrow b;\ g : b \leftrightarrow c;\ g' : b \leftrightarrow c\ ]\!]$
$\Longrightarrow (f \odot g) \sqsubseteq (f' \odot g')$

The next two lemmas below are derived from the axiom *comp-incl-mon*.

lemma (**in** *OrderedSemigroupoid*) *comp-incl-mon1*[*intro,simp*]:
assumes *i*: $f \sqsubseteq f'$
assumes *f*: $f : a \leftrightarrow b$
assumes *g*: $g : b \leftrightarrow c$
assumes *f'*: $f' : a \leftrightarrow b$
shows $(f \odot g) \sqsubseteq (f' \odot g)$
proof −
have *gg*:$g \sqsubseteq g$ **by** (*rule-tac incl-refl*, *rule homset-Mor*, *rule g*)
from *i f g f' gg* **show** *?thesis* **by** (*rule-tac comp-incl-mon*, *auto*)
qed

lemma (**in** *OrderedSemigroupoid*) *comp-incl-mon1-new*:
assumes *i*: $f \sqsubseteq f'$
assumes [*intro*]: $f : a \leftrightarrow b$
assumes [*intro*]: $g : b \leftrightarrow c$
assumes [*intro*]: $f' : a \leftrightarrow b$
shows $(f \odot g) \sqsubseteq (f' \odot g)$
proof −
have $g \sqsubseteq g$ **by** (*rule-tac incl-refl*, *auto*)
with *i* **show** *?thesis* **by** (*rule-tac comp-incl-mon*, *auto*)
qed

The *intro* attributes save us from naming the typing assumptions and auto suffices rather than naming the rules *homset-Mor* and *g*.

lemma (**in** *OrderedSemigroupoid*) *comp-incl-mon2*[*intro,simp*]:
assumes *i*: $g \sqsubseteq g'$
assumes *f-t*[*intro*]: $f : a \leftrightarrow b$
assumes *g-t*[*intro*]: $g : b \leftrightarrow c$
assumes *f'-t*[*intro*]: $g' : b \leftrightarrow c$
shows $(f \odot g) \sqsubseteq (f \odot g')$
proof −
have $f \sqsubseteq f$ **by** (*rule-tac incl-refl*, *auto*)

with *i* **show** *?thesis* **by** (*rule-tac comp-incl-mon, auto*)
qed

B.3.1 Transitivity Rules for Calculational Reasoning

lemma (in *OrderedSemigroupoid*) *incl-monotonicity*:
assumes *i*: $f \sqsubseteq g$
assumes *F-mon*: $(\bigwedge u\ v\ .\ [\![\ u \sqsubseteq v;\ u : a \leftrightarrow b;\ v : a \leftrightarrow b\ ]\!] \Longrightarrow F\ u \sqsubseteq F\ v)$
assumes *f-t*[*intro*]: $f : a \leftrightarrow b$
assumes *g-t*[*intro*]: $g : a \leftrightarrow b$
assumes *F-t*[*intro*]: $\bigwedge u\ .\ u : (a \leftrightarrow b) \Longrightarrow F\ u : (A \leftrightarrow B)$
shows $F\ f \sqsubseteq F\ g$
proof −
from *i F-mon* **show** *?thesis* **by** *auto*
qed

lemma (in *OrderedSemigroupoid*) *incl-mon*:
assumes *eq*: $h = F\ f$
assumes *i*: $f \sqsubseteq g$
assumes *F-mon*: $(\bigwedge u\ v\ .\ [\![\ u \sqsubseteq v;\ u : a \leftrightarrow b;\ v : a \leftrightarrow b\ ]\!] \Longrightarrow F\ u \sqsubseteq F\ v)$
assumes *f-t*[*intro*]: $f : a \leftrightarrow b$
assumes *g-t*[*intro*]: $g : a \leftrightarrow b$
assumes *h-t*[*intro*]: $h : A \leftrightarrow B$
assumes *F-t*[*intro*]: $\bigwedge u\ .\ u : (a \leftrightarrow b) \Longrightarrow F\ u : (A \leftrightarrow B)$
shows $h \sqsubseteq F\ g$
proof −
note *eq*
also from *i F-mon* **have** $F\ f \sqsubseteq F\ g$ **by** *auto*
finally show *?thesis* .
qed

lemma (in *OrderedSemigroupoid*) *mon-incl*:
assumes *i*: $f \sqsubseteq g$
assumes *eq*: $F\ g = h$
assumes *F-mon*: $(\bigwedge u\ v\ .\ [\![\ u \sqsubseteq v;\ u : a \leftrightarrow b;\ v : a \leftrightarrow b\ ]\!] \Longrightarrow F\ u \sqsubseteq F\ v)$
assumes *f-t*[*intro*]: $f : a \leftrightarrow b$
assumes *g-t*[*intro*]: $g : a \leftrightarrow b$
assumes *h-t*[*intro*]: $h : A \leftrightarrow B$
assumes *F-t*[*intro*]: $\bigwedge u\ .\ u : (a \leftrightarrow b) \Longrightarrow F\ u : (A \leftrightarrow B)$
shows $F\ f \sqsubseteq h$

proof −
from *i F-mon* **have** $F\ f \sqsubseteq F\ g$ **by** *auto*
also from *eq* **have** $\ldots = h$.
finally show *?thesis* .
qed

lemma (in *OrderedSemigroupoid*) *subst-incl*:
assumes *eq*: $f = g$
assumes *i*: $F\ g \sqsubseteq h$
assumes *f-t*[*intro*]: $f : a \leftrightarrow b$
assumes *g-t*[*intro*]: $g : a \leftrightarrow b$
assumes *h-t*[*intro*]: $h : A \leftrightarrow B$
assumes *F-t*[*intro*]: $\bigwedge u\ .\ u : (a \leftrightarrow b) \Longrightarrow F\ u : (A \leftrightarrow B)$
shows $F\ f \sqsubseteq h$
proof −
from *eq* **have** $F\ f = F\ g$ **by** *auto*
also from *i* **have** $\ldots \sqsubseteq h$.
finally show *?thesis* .
qed

lemma (in *OrderedSemigroupoid*) *incl-subst*:
assumes *eq*: $f = g$
assumes *i* : $h \sqsubseteq F\ f$
assumes *f-t*[*intro*]: $f : a \leftrightarrow b$
assumes *g-t*[*intro*]: $g : a \leftrightarrow b$
assumes *h-t*[*intro*]: $h : A \leftrightarrow B$
assumes *F-t*[*intro*]: $\bigwedge u\ .\ u : (a \leftrightarrow b) \Longrightarrow F\ u : (A \leftrightarrow B)$
shows $h \sqsubseteq F\ g$
proof −
note *i*
also from *eq* **have** $F\ f = F\ g$ **by** *auto*
finally show *?thesis* .
qed

lemma (in *OrderedSemigroupoid*) *subst-incl-new*:
$[\![\ f = g;\ F\ g \sqsubseteq h;\ f : a \leftrightarrow b;\ g : a \leftrightarrow b;\ h : A \leftrightarrow B;$
$\quad (\bigwedge u\ .\ u : (a \leftrightarrow b) \Longrightarrow F\ u : (A \leftrightarrow B))\ ]\!] \Longrightarrow F\ f \sqsubseteq h$
by (*drule mon-incl, simp-all*)

lemma (in *OrderedSemigroupoid*) *incl-mon-trans*:

$[\![h \sqsubseteq F f; f \sqsubseteq g; (\bigwedge u\ v\ .\ [\![u \sqsubseteq v; u : a \leftrightarrow b; v : a \leftrightarrow b\ ]\!] \Longrightarrow F u \sqsubseteq F v);$
$f : a \leftrightarrow b; g : a \leftrightarrow b; h : c \leftrightarrow d;$
$(\bigwedge u\ .\ u : (a \leftrightarrow b) \Longrightarrow F u : (c \leftrightarrow d))$
$]\!] \Longrightarrow h \sqsubseteq F g$
by (*rule-tac incl-trans, assumption, simp-all, simp-all*)

lemma (in *OrderedSemigroupoid*) *incl-mon-trans-new*:
assumes *i1*: $h \sqsubseteq F f$
assumes *i2*: $f \sqsubseteq g$
assumes *F-Mon*: $\bigwedge u\ v\ .\ [\![u \sqsubseteq v; u : a \leftrightarrow b; v : a \leftrightarrow b\ ]\!] \Longrightarrow F u \sqsubseteq F v$
assumes *f-t*[*intro*]: $f : a \leftrightarrow b$
assumes *g-t*[*intro*]: $g : a \leftrightarrow b$
assumes *h-t*[*intro*]: $h : c \leftrightarrow d$
assumes *F-t*[*intro*]: $\bigwedge u\ .\ u : (a \leftrightarrow b) \Longrightarrow F u : (c \leftrightarrow d)$
shows $h \sqsubseteq F g$
proof −
note *i1*
also from *i2 F-Mon* **have** $F f \sqsubseteq F g$ **by** *best*
finally show *?thesis* **by** *best*
qed

lemma (in *OrderedSemigroupoid*) *incl-incl-trans*:
assumes *i1*: $f \sqsubseteq g$
assumes *i2*: $h \sqsubseteq k$
assumes *eq*: $F g = G h$
assumes *F-mon*: $\bigwedge u\ v\ .\ [\![u \sqsubseteq v; u : a \leftrightarrow b; v : a \leftrightarrow b\ ]\!] \Longrightarrow F u \sqsubseteq F v$
assumes *G-mon*: $\bigwedge x\ y\ .\ [\![x \sqsubseteq y; x : c \leftrightarrow d; y : c \leftrightarrow d\ ]\!] \Longrightarrow G x \sqsubseteq G y$
assumes *f-t*[*intro*]: $f : a \leftrightarrow b$
assumes *g-t*[*intro*]: $g : a \leftrightarrow b$
assumes *h-t*[*intro*]: $h : c \leftrightarrow d$
assumes *k-t*[*intro*]: $k : c \leftrightarrow d$
assumes *F-t*[*intro*]: $\bigwedge u\ .\ u : (a \leftrightarrow b) \Longrightarrow F u : (A \leftrightarrow B)$
assumes *G-t*[*intro*]: $\bigwedge x\ .\ x : (c \leftrightarrow d) \Longrightarrow G x : (A \leftrightarrow B)$
shows $F f \sqsubseteq G k$
proof (*rule-tac g*=$F g$ **in** *incl-trans*)
from *i1 F-mon* **show** $F f \sqsubseteq F g$ **by** *auto*
next
from *eq i2 G-mon* **show** $F g \sqsubseteq G k$ **by** *auto*
next
show $F f \in A \leftrightarrow B$ **by** *auto*

next
show $F g \in A \leftrightarrow B$ **by** *auto*
next
show $G k \in A \leftrightarrow B$ **by** *auto*
qed

B.3.2 Properties of Automorphisms

constdefs
OS-transitive :: ('*o*, '*m*, '*r*) *OrderedSemigroupoid-scheme* ⇒ '*m* ⇒ *bool*
(*transitive*ı - [*1000*] *999*)
OS-transitive s R == *if isMor s R* & (*Ssrc s R* = *Strg s R*)
then incl s (*cmp s R R*) *R*
else arbitrary

lemma (in *OrderedSemigroupoid*) *transitive-def*:
$R : a \leftrightarrow a \Longrightarrow$ *transitive* $R = (R \odot R \sqsubseteq R)$
by (*unfold OS-transitive-def, simp*)

lemma (in *OrderedSemigroupoid*) *transitive-expand*:
assumes *n*: *transitive R*
assumes *m*: $R : a \leftrightarrow a$
shows $R \odot R \sqsubseteq R$
proof −
from *n m* **show** *?thesis* **by** (*rule-tac transitive-def* [*THEN iffD1*])
qed

lemma (in *OrderedSemigroupoid*) *transitive-intro*[*intro*]:
assumes *n*: $R \odot R \sqsubseteq R$
assumes *m*: $R : a \leftrightarrow a$
shows *transitive R*
proof −
from *n m* **show** *?thesis* **by** (*rule-tac transitive-def* [*THEN iffD2*])
qed

The following lemmas are useful for showing properties of residuals.

lemma (in *OrderedSemigroupoid*) *indirect-ineq'*:
$[\![R : a \leftrightarrow b; S : a \leftrightarrow b\ ]\!] \Longrightarrow ((R \sqsubseteq S) = (\forall\ C\ .C : a \leftrightarrow b \longrightarrow C \sqsubseteq R \longrightarrow C \sqsubseteq S))$
apply (*rule iffI*)
apply (*intro strip*)

apply (*rule-tac incl-trans*)
apply *auto*
done

lemma (**in** *OrderedSemigroupoid*) *indirect-ineq*:
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $S : a \leftrightarrow b$
shows $(R \sqsubseteq S) = (\forall\ C\ .C : a \leftrightarrow b \longrightarrow C \sqsubseteq R \longrightarrow C \sqsubseteq S)$
proof −
have $(R \sqsubseteq S) \Longrightarrow (\forall\ C\ .C : a \leftrightarrow b \longrightarrow C \sqsubseteq R \longrightarrow C \sqsubseteq S)$
proof −
assume *incl*: $R \sqsubseteq S$
show $\forall C.\ C \in a \leftrightarrow b \longrightarrow C \sqsubseteq R \longrightarrow C \sqsubseteq S$
proof (*intro strip*)
fix C
assume [*intro*]: $C \in a \leftrightarrow b$
assume [*intro*]: $C \sqsubseteq R$
also from *incl* **have** $R \sqsubseteq S$ **by** *simp*
finally show $C \sqsubseteq S$ **by** *best+*
qed
qed
moreover have $(\forall\ C\ .C : a \leftrightarrow b \longrightarrow C \sqsubseteq R \longrightarrow C \sqsubseteq S) \Longrightarrow (R \sqsubseteq S)$ **by**(*auto*)
ultimately show *?thesis* **by** (*rule iffI*, *best+*)
qed

lemmas (**in** *OrderedSemigroupoid*) *indir-ineq1* = *indirect-ineq* [*THEN iffD1*]
lemmas (**in** *OrderedSemigroupoid*) *indir-ineq2* = *indirect-ineq* [*THEN iffD2*]

lemma (**in** *OrderedSemigroupoid*) *indirect-equality*:
$[\![\ R : a \leftrightarrow b; S : a \leftrightarrow b\ ]\!] \Longrightarrow (R = S) = (\forall\ C\ .C : a \leftrightarrow b \longrightarrow ((C \sqsubseteq R) = (C \sqsubseteq S)))$
apply (*rule iffI*)
apply (*intro strip*)
apply *auto*
apply (*rule incl-antisym* [*THEN sym*])
apply (*rule-tac indirect-ineq* [*THEN iffD2*])
apply *auto*
done

lemmas (**in** *OrderedSemigroupoid*) *indir-eq1* = *indirect-equality* [*THEN iffD1*]
lemmas (**in** *OrderedSemigroupoid*) *indir-eq2* = *indirect-equality* [*THEN iffD2*]

B.3.3 Subidentities

constdefs
OS-isSId :: $('o, 'm, 'r)$ *OrderedSemigroupoid-scheme* $\Rightarrow$ $'m \Rightarrow$ *bool* (*isSId*$_1$ - [*1000*] *999*)

OS-isSId s R == *if* (*isMor s R* & *Ssrc s R* = *Strg s R*)
then (*let a* = *Ssrc s R in*
$(\forall b.$
$(\forall f.\ f : homset\ s\ a\ b \longrightarrow incl\ s\ (cmp\ s\ R\ f)\ f)$ &
$(\forall g.\ g : homset\ s\ b\ a \longrightarrow incl\ s\ (cmp\ s\ g\ R)\ g)$
$))$
else arbitrary

lemma (**in** *OrderedSemigroupoid*) *isSId-def*:
$R : a \leftrightarrow a \Longrightarrow isSId\ R = (\forall\ b.$
$(\forall f.\ f : a \leftrightarrow b \longrightarrow (\ R \odot f) \sqsubseteq f)$ &
$(\forall g.\ g : b \leftrightarrow a \longrightarrow (g \odot R) \sqsubseteq g))$
by (*unfold OS-isSId-def*, *simp add*: *Let-def*)

lemma (**in** *OrderedSemigroupoid*) *isSId-left*[*intro*, *simp*]:
assumes *R-t*: $R : a \leftrightarrow a$
assumes *R*: *isSId R*
assumes *f*: $f : a \leftrightarrow b$
shows $(\ R \odot f) \sqsubseteq f$
proof −
from *R-t R f* **show** *?thesis* **by** (*unfold OS-isSId-def*, *simp*)
qed

lemma (**in** *OrderedSemigroupoid*) *isSId-right*[*intro*, *simp*]:
assumes *R-t*: $R : a \leftrightarrow a$
assumes *R*: *isSId R*
assumes *g*: $g : b \leftrightarrow a$
shows $(\ g \odot R) \sqsubseteq g$
proof −
from *R-t R g* **show** *?thesis* **by** (*unfold OS-isSId-def*, *simp*)
qed

lemma (**in** *OrderedSemigroupoid*) *isSId*:
assumes *R-t*: $R : a \leftrightarrow a$
assumes *R*: *isSId R*

assumes *f*: $f : a \leftrightarrow b$
assumes *g*: $g : b \leftrightarrow a$
shows $(R \odot f) \sqsubseteq f$ & $(g \odot R) \sqsubseteq g$
proof −
from *R-t R f g* **show** *?thesis* **by** (*unfold OS-isSId-def*, *simp*)
qed

lemma (in *OrderedSemigroupoid*) *isSId-intro1*:
assumes *R-t*: R: $a \leftrightarrow a$
assumes *i1*: $\bigwedge b\ f.\ f : a \leftrightarrow b \Longrightarrow (R \odot f) \sqsubseteq f$
assumes *i2*: $\bigwedge b\ g.\ g : b \leftrightarrow a \Longrightarrow (g \odot R) \sqsubseteq g$
shows *isSId R*
proof −
from *R-t i1 i2* **show** *isSId R* **by**(*subst isSId-def*, *auto*)
qed

lemma (in *OrderedSemigroupoid*) *isSId-intro2-right*[*intro*]:
assumes *j-t*[*intro*, *simp*]: $j : a \leftrightarrow a$
assumes *j*[*intro*, *simp*]: *isSId j*
assumes *R-t*[*intro*, *simp*]: $R : a \leftrightarrow a$
assumes *Rj*[*intro*, *simp*]: $R \sqsubseteq j$
assumes *f*[*intro*,*simp*]: *f*: $a \leftrightarrow b$
shows $(R \odot f) \sqsubseteq f$
proof −
let $?g = j \odot f$
have *?g*: $a \leftrightarrow b$ **by** *auto*
moreover have *bf1*:$(R \odot f) \sqsubseteq ?g$ & $(?g \sqsubseteq f)$ **by** *auto*
moreover have *bf2* : $(((R \odot f) \sqsubseteq ?g)\ \&\ (?g \sqsubseteq f)) \Longrightarrow ((R \odot f) \sqsubseteq f)$ **by** (*rule-tac incl-trans*, *auto*)
ultimately show *?thesis* **by** *auto*
qed

lemma (in *OrderedSemigroupoid*) *isSId-intro2-left*[*intro*]:
assumes *j-t*[*intro*, *simp*]: $j : a \leftrightarrow a$
assumes *j*[*intro*, *simp*]: *isSId j*
assumes *R-t*[*intro*, *simp*]: $R : a \leftrightarrow a$
assumes *Rj*[*intro*, *simp*]: $R \sqsubseteq j$
assumes *g*[*intro*,*simp*]: *g*: $b \leftrightarrow a$
shows $(g \odot R) \sqsubseteq g$

proof −
let $?f = g \odot j$
have *?f*: $b \leftrightarrow a$ **by** *auto*
moreover have *bf1*:$(g \odot R) \sqsubseteq ?f$ & $(?f \sqsubseteq g)$ **by** *auto*
moreover have *bf2* : $(((g \odot R) \sqsubseteq ?f)\ \&\ (?f \sqsubseteq g)) \Longrightarrow ((g \odot R) \sqsubseteq g)$ **by** (*rule-tac incl-trans*, *auto*)
ultimately show *?thesis* **by** *auto*
qed

lemma (in *OrderedSemigroupoid*) *isSId-intro2*:
assumes *j-t*[*intro*, *simp*]: $j : a \leftrightarrow a$
assumes *j*[*intro*, *simp*]: *isSId j*
assumes *R-t*[*intro*, *simp*]: $R : a \leftrightarrow a$
assumes *Rj*[*intro*, *simp*]: $R \sqsubseteq j$
shows *isSId R*
by (*subst isSId-def*, *auto*)

constdefs
OS-SId :: ('*o*, '*m*, '*r*) *OrderedSemigroupoid-scheme* ⇒ '*o* ⇒ '*m set* (*SId*$_1$ - [*1000*] *999*)
OS-SId s a == *Collect* (λ *m* . *m* : *homset s a a* & *OS-isSId s m*)

lemma (in *OrderedSemigroupoid*) *SId-def*:
$SId\ a = \{\ m\ .\ m : a \leftrightarrow a\ \&\ isSId\ m\ \}$
by (*unfold OS-SId-def*, *simp*)

lemma (in *OrderedSemigroupoid*) *SId-intro*[*intro*]:
$[\![\ R : a \leftrightarrow a;\ isSId\ R\ ]\!] \Longrightarrow R : SId\ a$
by (*unfold OS-SId-def*, *simp*)

lemma (in *OrderedSemigroupoid*) *SId-homset*[*intro*,*simp*]:
$R : SId\ a \Longrightarrow R : a \leftrightarrow a$
by (*unfold OS-SId-def*, *simp*)

lemma (in *OrderedSemigroupoid*) *SId*[*intro?*,*simp*]:
$R : SId\ a \Longrightarrow isSId\ R$
by (*unfold OS-SId-def*, *simp*)

end

B.4 Ordering Properties of Operators on Homsets

theory *HomsetOpOrdProps*
imports *OrdSemi HomsetOpProps*
begin

B.4.1 Monotonicity of Unary Operators

constdefs
homset-Mon :: $('o, 'm, 'r)$ *OrderedSemigroupoid-scheme* $\Rightarrow$ $('m \Rightarrow 'm) \Rightarrow$ *bool*
(*hsMon*$_1$ - [*1000*] *999*)
homset-Mon s f == *ALL m n . incl s m n* $\longrightarrow$ *incl s* (*f m*) (*f n*)

lemma (**in** *OrderedSemigroupoid*) *hsMon-def*: *hsMon f* = $(\forall\ m\ n.\ m \sqsubseteq n \longrightarrow f\ m \sqsubseteq f\ n)$
by (*unfold homset-Mon-def*, *simp*)

lemma (**in** *OrderedSemigroupoid*) *hsMon-intro*[*intro?*]:
$[\![\bigwedge m\ n\ .\ [\![m \sqsubseteq n]\!] \Longrightarrow f\ m \sqsubseteq f\ n]\!] \Longrightarrow$ *hsMon f*
by (*subst hsMon-def*, *intro strip*, *simp*)

lemma (**in** *OrderedSemigroupoid*) *hsMon-intro-new*:
assumes *i1*: $\bigwedge m\ n\ .\ [\![m \sqsubseteq n]\!] \Longrightarrow f\ m \sqsubseteq f\ n$
shows *hsMon f*
proof (*subst hsMon-def*, *intro strip*)
from *i1* **show** $\bigwedge m\ n.\ m \sqsubseteq n \Longrightarrow f\ m \sqsubseteq f\ n$ **by** *simp*
qed

lemma (**in** *OrderedSemigroupoid*) *hsMon*[*intro?*, *simp*]:
$[\![$ *hsMon f*; $m \sqsubseteq n]\!] \Longrightarrow f\ m \sqsubseteq f\ n$
by (*drule hsMon-def* [*THEN iffD1*], *auto*)

lemma (**in** *OrderedSemigroupoid*) *hsMon-new*:
assumes *i1*: *hsMon f*
assumes *i2*: $m \sqsubseteq n$
shows $f\ m \sqsubseteq f\ n$
proof −
from *i1* **have** $(\forall\ m\ n.\ m \sqsubseteq n \longrightarrow f\ m \sqsubseteq f\ n)$ **by** (*rule hsMon-def* [*THEN iffD1*])
with *i2* **show** *?thesis* **by** *auto*
qed

B.4.2 Monotonicity of Binary Operators

To improve the structure of the proofs of

hsBinMon1-intro-new and *hsBinMon2-intro-new*, we have to rewrite the underlying definition of *homset-BinMon1* by expanding the parallel definition.

constdefs
homset-BinMon1 :: $('o, 'm, 'r)$ *OrderedSemigroupoid-scheme* $\Rightarrow$ $('m \Rightarrow 'm \Rightarrow 'm) \Rightarrow$ *bool*
(*hsBinMon*$_1$ - [*1000*] *999*)

homset-BinMon1 s f == *ALL m n p . isMor s m & isMor s n & isMor s p &*
(*let a* = *Ssrc s m*; *b* = *Strg s m in*
a = *Ssrc s n & b* = *Strg s n &*
a = *Ssrc s p & b* = *Strg s p*) *&*
incl s m n $\longrightarrow$ *incl s* (*f m p*) (*f n p*)

The following is the original definition of *homset-BinMon1*.

```
"homset_BinMon1 s f == ALL m n p . Semi_parallel s m n
                       \<longrightarrow> Semi_parallel s m p
                       \<longrightarrow> incl s m n
                       \<longrightarrow> incl s (f m p) (f n p)" *)

lemma (in OrderedSemigroupoid) hsBinMon1_def:
 "hsBinMon1 f = (\<forall> m n p. m \<parallel> n \<longrightarrow>
m \<parallel> p \<longrightarrow> m \<sqsubseteq> n
\<longrightarrow> f m p \<sqsubseteq> f n p)"
by (unfold homset_BinMon1_def, simp)
```

lemma (**in** *OrderedSemigroupoid*) *hsBinMon1-def*:
hsBinMon1 f = $(\forall\ m\ n\ p\ .$ *Mor m* $\wedge$ *Mor n* $\wedge$ *Mor p* $\wedge$ *src m* = *src n* $\wedge$ *trg m* = *trg n* $\wedge$
src m = *src p* $\wedge$ *trg m* = *trg p* $\wedge\ m \sqsubseteq n \longrightarrow f\ m\ p \sqsubseteq f\ n\ p)$
by (*unfold homset-BinMon1-def*, *simp add*: *Let-def*)

lemma (**in** *OrderedSemigroupoid*) *hsBinMon1-intro-0*[*intro?*]:
$[\![\bigwedge m\ n\ p\ .\ [\![m \sqsubseteq n$; *Mor m*; *Mor n*; *Mor p*; *src m* = *src n*;
trg m = *trg n*; *src m* = *src p*; *trg m* = *trg p* $]\!]$

$\Longrightarrow f\ m\ p \sqsubseteq f\ n\ p\rrbracket \Longrightarrow$ *hsBinMon1 f*
by (*subst hsBinMon1-def*, *intro strip*, *auto*)

lemma (**in** *OrderedSemigroupoid*) *hsBinMon1-intro-0-new*:
assumes *i1*: $\bigwedge m\ n\ p\ .\ [\![\ m \sqsubseteq n;\ Mor\ m;\ Mor\ n;\ Mor\ p;\ src\ m = src\ n;$
$trg\ m = trg\ n;\ src\ m = src\ p;\ trg\ m = trg\ p\ ]\!] \Longrightarrow f\ m\ p \sqsubseteq f\ n\ p$
shows *hsBinMon1 f*
proof (*subst hsBinMon1-def*, *intro strip*, *auto*)
from *i1* **show** $\bigwedge m\ n\ p.$
$[\![Mor\ m;\ Mor\ n;\ Mor\ p;\ src\ p = src\ n;\ trg\ p = trg\ n;\ src\ m = src\ n;$
$trg\ m = trg\ n;\ m \sqsubseteq n]\!] \Longrightarrow f\ m\ p \sqsubseteq f\ n\ p$ **by** *simp*
qed

lemma (**in** *OrderedSemigroupoid*) *hsBinMon1-intro*[*intro?*]:
assumes *i1*: $\bigwedge a\ b\ m\ n\ p\ .\ [\![\ m \sqsubseteq n;\ m : a \leftrightarrow b;\ n : a \leftrightarrow b;\ p : a \leftrightarrow b\ ]\!]$
$\Longrightarrow f\ m\ p \sqsubseteq f\ n\ p$
shows *hsBinMon1 f*
proof (*rule hsBinMon1-intro-0*)
fix *m* **and** *n* **and** *p* — This makes Isabelle accept the "show"!
assume *incl*[*intro*, *simp*]: $m \sqsubseteq n$
assume [*intro*]: *Mor m*
assume [*intro*]: *Mor n*
assume [*intro*]: *Mor p*
assume [*intro*,*simp*]: *src m* = *src n*
assume [*intro*,*simp*]: *trg m* = *trg n*
assume *s-mp*: *src m* = *src p*
assume *t-mp*: *trg m* = *trg p*
note *i1* — More elegant than the next line. See also *rule-tac i1* in the next lemma
moreover have $m \sqsubseteq n$ **by** *simp*
moreover have m : *src m* $\leftrightarrow$ *trg m* **by** (*rule-tac homset1*, *best+*)
moreover have n : *src m* $\leftrightarrow$ *trg m* **by** (*rule-tac homset0*, *auto*)
moreover from *s-mp t-mp* **have** p : *src m* $\leftrightarrow$ *trg m* **by** (*rule-tac homset0*, *auto*)
ultimately show $f\ m\ p \sqsubseteq f\ n\ p$ **by** *best+*
qed

lemma (**in** *OrderedSemigroupoid*) *hsBinMon1-intro-new*[*intro?*]:
fixes *f* — makes no difference
assumes *i1*: $\bigwedge a\ b\ m\ n\ p\ .\ [\![\ m \sqsubseteq n;\ m : a \leftrightarrow b;\ n : a \leftrightarrow b;\ p : a \leftrightarrow b\ ]\!]$
$\Longrightarrow f\ m\ p \sqsubseteq f\ n\ p$
shows *hsBinMon1 f*

proof (*rule hsBinMon1-intro-0*)
fix *a* **and** *b* **and** *m* **and** *n* **and** *p*
assume [*simp*, *intro*]: $m \sqsubseteq n$
assume [*simp*, *intro*]: *Mor m*
assume [*simp*, *intro*]: *Mor n*
assume [*simp*, *intro*]: *Mor p*
assume *sn*[*simp*, *intro*]: *src m* = *src n* — these "diverging equations" need special treatment below
assume *tn*[*simp*, *intro*]: *trg m* = *trg n*
assume *sp*[*simp*, *intro*]: *src m* = *src p*
assume *tp*[*simp*, *intro*]: *trg m* = *trg p*
assume [*simp*, *intro*]: *a* = *src p*
assume [*simp*, *intro*]: *b* = *trg p*
have [*simp*, *intro*]: *src n* = *src p* **by** (*subst sn* [*THEN sym*], *rule sp*)
have [*simp*, *intro*]: *trg n* = *trg p* **by** (*subst tn* [*THEN sym*], *rule tp*)
have [*simp*, *intro*]: *Obj a* **by** *auto*
have [*simp*, *intro*]: *Obj b* **by** *auto*
have [*simp*, *intro*]: $m : a \leftrightarrow b$ **by** (*rule homset*, *auto*)
have [*simp*, *intro*]: $n : a \leftrightarrow b$ **by** (*rule homset*, *auto*)
have [*simp*, *intro*]: $p : a \leftrightarrow b$ **by** (*rule homset*, *auto*)
show $f\ m\ p \sqsubseteq f\ n\ p$ **by** (*rule i1*, *auto*) — *from i1* didn't work.
— How does *rule i1* work? How does it differ from *from i1*
qed

lemma (**in** *OrderedSemigroupoid*) *hsBinMon1*[*intro?*, *simp*]:
$[\![\ hsBinMon1\ f;\ m \sqsubseteq n;\ m : a \leftrightarrow b;\ n : a \leftrightarrow b;\ p : a \leftrightarrow b\ ]\!] \Longrightarrow f\ m\ p \sqsubseteq f\ n\ p$
by (*drule hsBinMon1-def* [*THEN iffD1*], *auto*)

lemma (**in** *OrderedSemigroupoid*) *hsBinMon1-new*:
assumes *i1*: *hsBinMon1 f*
assumes *i2*: $m \sqsubseteq n$
assumes *m-t*: $m : a \leftrightarrow b$
assumes *n-t*: $n : a \leftrightarrow b$
assumes *p-t*: $p : a \leftrightarrow b$
shows $f\ m\ p \sqsubseteq f\ n\ p$
proof −
from *i1* **have** $\forall\ m\ n\ p.\ \ Mor\ m \wedge Mor\ n \wedge Mor\ p \wedge src\ m = src\ n \wedge trg\ m = trg\ n \wedge$
$src\ m = src\ p \wedge trg\ m = trg\ p \wedge m \sqsubseteq n \longrightarrow f\ m\ p \sqsubseteq f\ n\ p$
by (*rule hsBinMon1-def* [*THEN iffD1*])
with *i2 m-t n-t p-t* **show** *?thesis* **by** *simp*

qed

constdefs
homset-BinMon2 :: ('o, 'm, 'r) *OrderedSemigroupoid-scheme* ⇒ ('m ⇒ 'm ⇒ 'm) ⇒ *bool*
(*hsBinMon2*₁ - [1000] 999)
homset-BinMon2 s f == *ALL m n p . isMor s m & isMor s n & isMor s p*
& (*let a* = *Ssrc s m*; *b* = *Strg s m in*
a = *Ssrc s n & b* = *Strg s n*
& *a* = *Ssrc s p & b* = *Strg s p*)
& *incl s m n* ⟶ *incl s* (*f p m*) (*f p n*)

lemma (in *OrderedSemigroupoid*) *hsBinMon2-def*:
hsBinMon2 f = (∀ *m n p . Mor m* ∧ *Mor n* ∧ *Mor p* ∧ *src m* = *src n*
∧ *trg m* = *trg n* ∧ *src m* = *src p* ∧ *trg m* = *trg p*
∧ *m* ⊑ *n* ⟶ *f p m* ⊑ *f p n*)
by (*unfold homset-BinMon2-def*, *simp add*: *Let-def*)

lemma (in *OrderedSemigroupoid*) *hsBinMon2-intro-0*[*intro?*]:
⟦ ⋀ *m n p . ⟦ m* ⊑ *n*; *Mor m*; *Mor n*; *Mor p*; *src m* = *src n*; *trg m* = *trg n*;
src m = *src p*; *trg m* = *trg p* ⟧ ⟹ *f p m* ⊑ *f p n*⟧ ⟹ *hsBinMon2 f*
by (*subst hsBinMon2-def*, *intro strip*, *auto*)

lemma (in *OrderedSemigroupoid*) *hsBinMon2-intro*[*intro?*]:
assumes *i1*: ⋀ *a b m n p . ⟦ m* ⊑ *n*; *m* : *a* ↔ *b*; *n* : *a* ↔ *b*; *p* : *a* ↔ *b* ⟧
⟹ *f p m* ⊑ *f p n*
shows *hsBinMon2 f*
proof (*rule hsBinMon2-intro-0*)
fix *m* **and** *n* **and** *p*
assume *incl*[*intro*, *simp*]: *m* ⊑ *n*
assume [*intro*, *simp*]: *Mor m*
assume [*intro*, *simp*]: *Mor n*
assume [*intro*, *simp*]: *Mor p*
assume [*intro*, *simp*]: *src m* = *src n*
assume [*intro*, *simp*]: *trg m* = *trg n*
assume *s-mp*[*intro*, *simp*]: *src m* = *src p*
assume *t-mp*[*intro*, *simp*]: *trg m* = *trg p*
from *i1* **have** ⋀ *a b m n p . ⟦ m* ⊑ *n*; *m* : *a* ↔ *b*; *n* : *a* ↔ *b*; *p* : *a* ↔ *b* ⟧
⟹ *f p m* ⊑ *f p n* **by** *simp*
moreover have *m* ⊑ *n* **by** *simp*
moreover have *m* : *src m* ↔ *trg m* **by** (*rule homset1*, *best+*)

moreover have *n* : *src m* ↔ *trg m* **by** (*rule homset0*, *auto*)
moreover from *s-mp t-mp* **have** *p* : *src m* ↔ *trg m* **by** (*rule-tac homset0*, *auto*)
ultimately show *f p m* ⊑ *f p n* **by** *best+*
qed

lemma (in *OrderedSemigroupoid*) *hsBinMon2*[*intro?*, *simp*]:
⟦ *hsBinMon2 f*; *m* ⊑ *n*; *m* : *a* ↔ *b*; *n* : *a* ↔ *b*; *p* : *a* ↔ *b* ⟧ ⟹ *f p m* ⊑ *f p n*
by (*drule hsBinMon2-def* [*THEN iffD1*], *auto*)

lemma (in *OrderedSemigroupoid*) *hsBinMon2-new*:
assumes *i1*: *hsBinMon2 f*
assumes *i2*: *m* ⊑ *n*
assumes *m-t*: *m* : *a* ↔ *b*
assumes *n-t*: *n* : *a* ↔ *b*
assumes *p-t*: *p* : *a* ↔ *b*
shows *f p m* ⊑ *f p n*
proof −
from *i1* **have** (∀ *m n p . Mor m* ∧ *Mor n* ∧ *Mor p* ∧ *src m* = *src n*
∧ *trg m* = *trg n* ∧ *src m* = *src p* ∧ *trg m* = *trg p*
∧ *m* ⊑ *n* ⟶ *f p m* ⊑ *f p n*) **by** (*rule hsBinMon2-def* [*THEN iffD1*])
with *i2 m-t n-t p-t* **show** *?thesis* **by** *simp*
qed

lemma (in *OrderedSemigroupoid*) *hsBinMon2-from-1-and-Commut-0*:
assumes *m*[*intro*,*simp*]: *hsBinMon1 f*
assumes *c*[*intro*,*simp*]: *hsCommutative f*
assumes *i*[*intro*,*simp*]: *m* ⊑ *n*
assumes *T-m*[*intro*,*simp*]: *m* : *a* ↔ *b*
assumes *T-n*[*intro*,*simp*]: *n* : *a* ↔ *b*
assumes *T-p*[*intro*,*simp*]: *p* : *a* ↔ *b*
shows *f p m* ⊑ *f p n*
proof −
have *f p m* = *f m p* **by** (*rule-tac hsCommutative* [*of f*], *auto*)
also have ... ⊑ *f n p* **by** (*rule-tac hsBinMon1* [*of f*], *auto*)
also have ... = *f p n* **by** (*rule-tac hsCommutative* [*of f*], *auto*)
finally show *?thesis* .
qed

lemma (in *OrderedSemigroupoid*) *hsBinMon2-from-1-and-Commut*:
assumes *m*[*intro*,*simp*]: *hsBinMon1 f*

assumes *c*[*intro,simp*]: *hsCommutative f*
shows *hsBinMon2 f*
apply (*rule hsBinMon2-intro* [*of f*])
apply (*rule hsBinMon2-from-1-and-Commut-0* [*of f*])
apply *auto*
done

lemma (**in** *OrderedSemigroupoid*) *hsBinMon1-from-2-and-Commut-0*:
assumes *m*[*intro,simp*]: *hsBinMon2 f*
assumes *c*[*intro,simp*]: *hsCommutative f*
assumes *i*[*intro,simp*]: $m \sqsubseteq n$
assumes *T-m*[*intro,simp*]: $m : a \leftrightarrow b$
assumes *T-n*[*intro,simp*]: $n : a \leftrightarrow b$
assumes *T-p*[*intro,simp*]: $p : a \leftrightarrow b$
shows $f\ m\ p \sqsubseteq f\ n\ p$
proof −
have $f\ m\ p = f\ p\ m$ **by** (*rule-tac hsCommutative* [*of f*], *auto*)
also have $\ldots \sqsubseteq f\ p\ n$ **by** (*rule-tac hsBinMon2* [*of f*], *auto*)
also have $\ldots = f\ n\ p$ **by** (*rule-tac hsCommutative* [*of f*], *auto*)
finally show *?thesis* .
qed

lemma (**in** *OrderedSemigroupoid*) *hsBinMon1-from-2-and-Commut*:
assumes *m*[*intro,simp*]: *hsBinMon2 f*
assumes *c*[*intro,simp*]: *hsCommutative f*
shows *hsBinMon1 f*
apply (*rule hsBinMon1-intro* [*of f*])
apply (*rule hsBinMon1-from-2-and-Commut-0* [*of f*])
apply *auto*
done

constdefs
homset-BinMon :: ('*o*, '*m*, '*r*) *OrderedSemigroupoid-scheme* ⇒ ('*m* ⇒ '*m* ⇒ '*m*) ⇒ *bool*
(*hsBinMon*₁ - [*1000*] *999*)
homset-BinMon s f == *homset-BinMon1 s f* & *homset-BinMon2 s f*

lemma (**in** *OrderedSemigroupoid*) *hsBinMon-def*:
hsBinMon f = (*hsBinMon1 f* ∧ *hsBinMon2 f*)
by (*rule iffI*, *unfold homset-BinMon-def*, *assumption*+)

lemma (**in** *OrderedSemigroupoid*) *hsBinMon-def-new*:
hsBinMon f = (*hsBinMon1 f* ∧ *hsBinMon2 f*)
proof −
have *hsBinMon f* ⟹ (*hsBinMon1 f* ∧ *hsBinMon2 f*)
proof −
assume *i1*: *hsBinMon f*
hence *hsBinMon1 f* **by** (*unfold homset-BinMon-def*, *simp*)
moreover from *i1* **have** *hsBinMon2 f* **by** (*unfold homset-BinMon-def*, *simp*)
ultimately show *?thesis* ..
qed
moreover have *hsBinMon1 f* ∧ *hsBinMon2 f* ⟹ *hsBinMon f*
proof −
assume *hsBinMon1 f* ∧ *hsBinMon2 f*
thus *?thesis* **by** (*rule hsBinMon-def* [*THEN iffD2*])
qed
ultimately show *?thesis* **by** (*rule iffI*)
qed

lemma (**in** *OrderedSemigroupoid*) *hsBinMon-intro-0*[*intro?*]:
⟦ *hsBinMon1 f*; *hsBinMon2 f* ⟧ ⟹ *hsBinMon f*
by (*subst homset-BinMon-def*, *rule conjI*)

lemma (**in** *OrderedSemigroupoid*) *hsBinMon-intro*[*intro?*]:
⟦ ⋀ *a b m n p q* . ⟦ $m \sqsubseteq n$; $p \sqsubseteq q$; $m : a \leftrightarrow b$; $n : a \leftrightarrow b$; $p : a \leftrightarrow b$; $q : a \leftrightarrow b$ ⟧ ⟹ $f\ m\ p \sqsubseteq f\ n\ q$⟧ ⟹ *hsBinMon f*
apply (*unfold homset-BinMon-def*, *rule conjI*)
apply (*rule hsBinMon1-intro*)
apply (*subgoal-tac incl OS p p*, *simp*, *simp*)
apply (*rule hsBinMon2-intro*)
apply (*subgoal-tac incl OS p p*, *simp*, *simp*)
done

lemma (**in** *OrderedSemigroupoid*) *hsBinMon-1*[*intro?*, *simp*]:
⟦ *hsBinMon f* ⟧ ⟹ *hsBinMon1 f*
by (*drule hsBinMon-def* [*THEN iffD1*], *erule conjE*)

lemma (**in** *OrderedSemigroupoid*) *hsBinMon-2*[*intro?*, *simp*]:
⟦ *hsBinMon f* ⟧ ⟹ *hsBinMon2 f*
by (*drule hsBinMon-def* [*THEN iffD1*], *erule conjE*)

lemma (in *OrderedSemigroupoid*) *hsBinMon*[*intro?*, *simp*]:
assumes *m*[*intro*,*simp*]: *hsBinMon f*
assumes *i*[*intro*,*simp*]: $m \sqsubseteq n$
assumes *j*[*intro*,*simp*]: $p \sqsubseteq q$
assumes *T-m*[*intro*,*simp*]: $m : a \leftrightarrow b$
assumes *T-n*[*intro*,*simp*]: $n : a \leftrightarrow b$
assumes *T-p*[*intro*,*simp*]: $p : a \leftrightarrow b$
assumes *T-q*[*intro*,*simp*]: $q : a \leftrightarrow b$
assumes *T-1*[*intro*,*simp*]: $f\ m\ p : a \leftrightarrow b$
assumes *T-2*[*intro*,*simp*]: $f\ n\ p : a \leftrightarrow b$
assumes *T-2*[*intro*,*simp*]: $f\ n\ q : a \leftrightarrow b$
shows $f\ m\ p \sqsubseteq f\ n\ q$
proof −
have $f\ m\ p \sqsubseteq f\ n\ p$ **by** (*rule-tac hsBinMon1* [*of f*], *auto*)
also have ... $\sqsubseteq f\ n\ q$ **by** (*rule-tac hsBinMon2* [*of f*], *auto*)
finally show *?thesis* **by** *best+*
qed

end

B.5 Structure Record for SemiAllegories

theory *SemiAllRecord*
imports *OrdSemi*
begin

We present this record in a separate theory in order to make it easier to have theories that do not need these components, but components higher up in the record hierarchy.

record ('*o*, '*m*) *SemiAllegory* = ('*o*, '*m*) *OrderedSemigroupoid* +
meet :: '*m* ⇒ '*m* ⇒ '*m* (**infixr** ⊓ 70)
conv :: '*m* ⇒ '*m* (-⌣ [1000] 999)
All-dom :: '*m* ⇒ '*m* (dom - [1000] 999)
All-rang :: '*m* ⇒ '*m* (rang - [1000] 999)
end

B.6 Semigroupoids with Converse

theory *ConvSemi*
imports *SemiAllRecord*
begin

B.6.1 Definitions

locale *ConvSemi* = *Semigroupoid C* +
assumes *conv-homset*[*intro*,*simp*]: $R : a \leftrightarrow b \Longrightarrow R^\smile : b \leftrightarrow a$
assumes *conv-idem*[*simp*, *intro*]: *Mor* $R \Longrightarrow (R^\smile)^\smile = R$
assumes *conv-cmp*[*simp*, *intro*]: $[\![R : a \leftrightarrow b;\ S : b \leftrightarrow c]\!] \Longrightarrow (R \odot S)^\smile = S^\smile \odot R^\smile$

B.6.2 Auxiliary Lemmas

lemma (in *ConvSemi*) *conv-defined*[*simp*]: *Mor* $R \Longrightarrow$ *Mor* $(R^\smile)$
by (*drule homset1*, *auto*)

lemma (in *ConvSemi*) *conv-src*[*simp*]: *Mor* $R \Longrightarrow$ *src* $(R^\smile)$ = *trg* R
by (*drule homset1*, *drule conv-homset*, *simp*)

lemma (in *ConvSemi*) *conv-trg*[*simp*]: *Mor* $R \Longrightarrow$ *trg* $(R^\smile)$ = *src* R
by (*drule homset1*, *drule conv-homset*, *simp*)

lemma (in *ConvSemi*) *conv-equality*[*simp*]:
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $S : a \leftrightarrow b$
shows $(R^\smile = S^\smile) = (R = S)$
proof (*rule iffI*)
assume [*simp*]: $R^\smile = S^\smile$
have $R = (R^\smile)^\smile$ **by** (*rule conv-idem* [*THEN sym*], *auto*)
also have ... $= (S^\smile)^\smile$ **by** *auto*
also have ... $= S$ **by** (*rule conv-idem*, *auto*)
finally show $R = S$.
next
assume [*simp*]: $R = S$
show $R^\smile = S^\smile$ **by** *simp*
qed

lemma (in *ConvSemi*) *conv-equality-1*:
assumes [*intro*]: $R : a \leftrightarrow b$

assumes [*intro*]: $S : a \leftrightarrow b$
shows $(R^\smile = S) = (R = S^\smile)$
proof (*rule iffI*)
assume [*simp*]: $R^\smile = S$
have $R = (R^\smile)^\smile$ **by** (*rule conv-idem* [*THEN sym*], *auto*)
also have $\ldots = S^\smile$ **by** *auto*
finally show $R = S^\smile$.
next
assume [*simp*]: $R = S^\smile$
have $R^\smile = (S^\smile)^\smile$ **by** *simp*
also have $\ldots = S$ **by** (*rule conv-idem*, *auto*)
finally show $R^\smile = S$.
qed

end

B.7 Ordered Semigroupoids with Converse

theory *ConvOrdSemi*
imports *ConvSemi*
begin

B.7.1 Definitions

locale *ConvOrdSemi* = *OrderedSemigroupoid OS* + *ConvSemi OS* +
assumes *conv-mon*[*intro*,*simp*]: $[\![R \sqsubseteq S; R : a \leftrightarrow b; S : a \leftrightarrow b]\!] \Longrightarrow R^\smile \sqsubseteq S^\smile$

B.7.2 Auxiliary Lemmas

lemma (**in** *ConvOrdSemi*) *conv-incl*[*intro*]:
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $S : a \leftrightarrow b$
shows $(R^\smile \sqsubseteq S^\smile) = (R \sqsubseteq S)$
proof (*rule iffI*)
assume [*simp*]: $R^\smile \sqsubseteq S^\smile$
have $R = (R^\smile)^\smile$ **by** (*rule conv-idem* [*THEN sym*], *auto*)
also have $\ldots \sqsubseteq (S^\smile)^\smile$ **by** (*rule conv-mon*, *auto*)
also have $\ldots = S$ **by** (*rule conv-idem*, *auto*)
finally show $R \sqsubseteq S$.

next
assume [*simp*]: $R \sqsubseteq S$
show $R^\smile \sqsubseteq S^\smile$ **by** (*rule conv-mon*, *auto*)
qed

lemma (**in** *ConvOrdSemi*) *conv-idem1*[*intro*]:
assumes [*intro*]: $R : a \leftrightarrow b$
shows $(R^\smile)^\smile \sqsubseteq R$
proof −
have $(R^\smile)^\smile \sqsubseteq (R^\smile)^\smile$ **by** (*rule incl-refl*, *best*)
moreover have $\ldots = R$ **by** (*rule conv-idem*, *best*)
ultimately show *?thesis* **by** *auto*
qed

lemma (**in** *ConvOrdSemi*) *conv-idem2*[*intro*]:
assumes [*intro*]: $R : a \leftrightarrow b$
shows $R \sqsubseteq (R^\smile)^\smile$
proof −
have $R \sqsubseteq R$ **by** (*rule incl-refl*, *best*)
moreover have $\ldots = (R^\smile)^\smile$ **by** (*rule conv-idem* [*THEN sym*], *best*)
ultimately show *?thesis* **by** *auto*
qed

B.7.3 Properties of Homogeneous Relations

constdefs
All-symmetric :: ('*o*, '*m*, '*r*)*SemiAllegory-scheme* ⇒ '*m* ⇒ *bool*
(*symmetric*₁ - [*1000*] *999*)
All-symmetric s R == *if isMor s R* & (*Ssrc s R* = *Strg s R*)
then conv s R = *R*
else arbitrary

lemma (**in** *ConvOrdSemi*) *symmetric-def*: $R : a \leftrightarrow a \Longrightarrow$ *symmetric* $R = (R^\smile = R)$
by (*unfold All-symmetric-def*, *auto*)

lemma (**in** *ConvOrdSemi*) *symmetric-expand*:
assumes *n*: *symmetric R*
assumes *nn*: $R : a \leftrightarrow a$
shows $R^\smile = R$
proof −

from *n nn* **show** *?thesis* **by** (*rule-tac symmetric-def* [*THEN iffD1*])
qed

lemma (**in** *ConvOrdSemi*) *symmetric-intro*[*intro*]:
assumes *n*: $R^\smile = R$
assumes *nn*[*intro*]: $R : a \mapsto a$
shows *symmetric R*
proof −
from *n nn* **show** *?thesis* **by** (*rule-tac symmetric-def* [*THEN iffD2*], *auto*)
qed

end

B.8 Bounds in Ordered Semigroupoids

theory *OrdSemiBounds*
imports *OrdSemi*
begin

B.8.1 Lower Bounds

constdefs
OS-isLBound :: ('*o*, '*m*, '*r*) *OrderedSemigroupoid-scheme* ⇒ '*m* ⇒ '*m* ⇒ '*m* ⇒ *bool*
(*isLBound*ı - - - [*1000,1000,1000*] *999*)
OS-isLBound struct R S M ==
isMor struct M &
(*let a* = *Ssrc struct M*; *b* = *Strg struct M in*
R : *homset struct a b* &
S : *homset struct a b* &
incl struct M R & *incl struct M S*)

lemma (**in** *OrderedSemigroupoid*) *isLBound-def*:
isLBound R S M = (*Mor M* ∧ (*let a* = *src M*; *b* = *trg M*
in R : $a \mapsto b$ ∧ *S* : $a \mapsto b$ ∧ $M \sqsubseteq R$ ∧ $M \sqsubseteq S$))
by (*unfold OS-isLBound-def*, *auto*)

lemma (**in** *OrderedSemigroupoid*) *isLBound-def-new*:
isLBound R S M = (*Mor M* ∧ *R* : *src M* ↦ *trg M* ∧ *S* : *src M* ↦ *trg M* ∧ $M \sqsubseteq R$ ∧ $M \sqsubseteq S$)

by (*unfold OS-isLBound-def*, *simp add*: *Let-def*)

It is nice to have *isLbound-def* without the *let-def* construct – we need not add *simp add*: *Let-def* to *auto*.

lemma (**in** *OrderedSemigroupoid*) *isLBound-expand*[*elim?*]:
assumes *l*: *isLBound R S M*
assumes [*intro*]: $R : a \mapsto b$ — unnecessary, but hygienic
assumes [*intro*]: $S : a \mapsto b$
assumes [*intro*]: $M : a \mapsto b$
shows $M \sqsubseteq R$ & $M \sqsubseteq S$
apply (*insert l*)
apply (*drule isLBound-def-new* [*THEN iffD1*])
apply (*simp add*: *Let-def*)
done

lemma (**in** *OrderedSemigroupoid*) *isLBound-expand-new*:
assumes *l*: *isLBound R S M*
assumes [*intro*]: $R : a \mapsto b$ — unnecessary, but hygienic
assumes [*intro*]: $S : a \mapsto b$
assumes [*intro*]: $M : a \mapsto b$
shows $M \sqsubseteq R$ & $M \sqsubseteq S$
proof (*rule conjI*)
from *l* **have** *Mor M* ∧ *R* : *src M* ↦ *trg M* ∧ *S* : *src M* ↦ *trg M* ∧ $M \sqsubseteq R$ ∧ $M \sqsubseteq S$
by (*rule isLBound-def-new* [*THEN iffD1*])
thus $M \sqsubseteq R$ **by** *simp*
next
from *l* **have** *Mor M* ∧ *R* : *src M* ↦ *trg M* ∧ *S* : *src M* ↦ *trg M* ∧ $M \sqsubseteq R$ ∧ $M \sqsubseteq S$
by (*rule isLBound-def-new* [*THEN iffD1*])
thus $M \sqsubseteq S$ **by** *simp*
qed

lemma (**in** *OrderedSemigroupoid*) *isLBound-1*[*elim?*]:
assumes *l*: *isLBound R S M*
assumes [*intro*]: $R : a \mapsto b$
assumes [*intro*]: $S : a \mapsto b$
assumes [*intro*]: $M : a \mapsto b$
shows $M \sqsubseteq R$
proof −
from *l* **have** $M \sqsubseteq R$ & $M \sqsubseteq S$ **by** (*rule isLBound-expand*, *best*+)
thus *?thesis* ..

qed

lemma (**in** *OrderedSemigroupoid*) *isLBound-2[elim?]*:
assumes *l*: *isLBound R S M*
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $M : a \leftrightarrow b$
shows $M \sqsubseteq S$
proof −
from *l* **have** $M \sqsubseteq R \;\&\; M \sqsubseteq S$ **by** (*rule isLBound-expand*, *best+*)
thus *?thesis* **..**
qed

lemma (**in** *OrderedSemigroupoid*) *isLBound-contract*:
assumes *l*: $M \sqsubseteq R \;\&\; M \sqsubseteq S$
assumes *R-t*[*intro*]: $R : a \leftrightarrow b$
assumes *S-t*[*intro*]: $S : a \leftrightarrow b$
assumes *M-t*[*intro*]: $M : a \leftrightarrow b$
shows *isLBound R S M*
proof (*subst isLBound-def*, *simp add*: *l Let-def*, *rule conjI*)
show *Mor M* **by** *auto*
next
from *M-t R-t S-t* **show** $R \in src\ M \leftrightarrow trg\ M \wedge S \in src\ M \leftrightarrow trg\ M$ **by** *auto*
qed

lemma (**in** *OrderedSemigroupoid*) *isLBound-contract-new*:
assumes *l*: $M \sqsubseteq R \;\&\; M \sqsubseteq S$
assumes *R-t*[*intro*]: $R : a \leftrightarrow b$
assumes *S-t*[*intro*]: $S : a \leftrightarrow b$
assumes *M-t*[*intro*]: $M : a \leftrightarrow b$
shows *isLBound R S M*
proof (*subst isLBound-def-new*, *rule conjI*)
show *Mor M* **by** *auto*
next
from *M-t R-t S-t l* **show** $R \in src\ M \leftrightarrow trg\ M \wedge S \in src\ M \leftrightarrow trg\ M \wedge M \sqsubseteq R \wedge M \sqsubseteq S$
by *auto*
qed

lemma (**in** *OrderedSemigroupoid*) *isLBound-intro[intro?]*:

assumes *l1*: $M \sqsubseteq R$
assumes *l2*: $M \sqsubseteq S$
assumes *R-t*[*intro*]: $R : a \leftrightarrow b$
assumes *S-t*[*intro*]: $S : a \leftrightarrow b$
assumes *M-t*[*intro*]: $M : a \leftrightarrow b$
shows *isLBound R S M*
proof −
from *l1 l2* **have** $M \sqsubseteq R \;\&\; M \sqsubseteq S$ **by** (*rule-tac conjI*, *best+*)
thus *?thesis* **by** (*rule-tac isLBound-contract*, *best+*)
qed

lemma (**in** *OrderedSemigroupoid*) *incl-isLBound[intro?]*:
assumes *l*: *isLBound R S M*
assumes *q*: $Q \sqsubseteq M$
assumes *R-t*[*intro*]: $R : a \leftrightarrow b$
assumes *S-t*[*intro*]: $S : a \leftrightarrow b$
assumes *M-t*[*intro*]: $M : a \leftrightarrow b$
assumes *Q-t*[*intro*]: $Q : a \leftrightarrow b$
shows *isLBound R S Q*
proof −
from *l* **have** $M \sqsubseteq R$ **by** (*rule isLBound-1*, *best+*)
with *q* **have** *r*: $Q \sqsubseteq R$ **by** (*rule incl-trans*, *auto*)
from *l* **have** $M \sqsubseteq S$ **by** (*rule isLBound-2*, *best+*)
with *q* **have** *s*: $Q \sqsubseteq S$ **by** (*rule incl-trans*, *auto*)
from *r s* **show** *?thesis* **by** (*rule isLBound-intro*, *auto*)
qed

lemma (**in** *OrderedSemigroupoid*) *incl-isLBound-new*:
assumes *l*: *isLBound R S M*
assumes *q*: $Q \sqsubseteq M$
assumes *R-t*[*intro*]: $R : a \leftrightarrow b$
assumes *S-t*[*intro*]: $S : a \leftrightarrow b$
assumes *M-t*[*intro*]: $M : a \leftrightarrow b$
assumes *Q-t*[*intro*]: $Q : a \leftrightarrow b$
shows *isLBound R S Q*
proof −
from *l* **have** $M \sqsubseteq R$ **by** (*rule isLBound-1*, *best+*)
with *q* **have** $Q \sqsubseteq R$ **by** (*rule incl-trans*, *auto*)
moreover from *l* **have** $M \sqsubseteq S$ **by** (*rule isLBound-2*, *best+*)
with *q* **have** $Q \sqsubseteq S$ **by** (*rule incl-trans*, *auto*)

ultimately show *?thesis* **by** (*rule isLBound-intro*, *auto*)
qed

constdefs
OS-isMeet :: ('*o*, '*m*, '*r*) *OrderedSemigroupoid-scheme* ⇒ '*m* ⇒ '*m* ⇒ '*m* ⇒ *bool*
(*isMeet*₁ - - - [*1000*,*1000*,*1000*] *999*)
OS-isMeet struct R S M == *OS-isLBound struct R S M* &
(*ALL L* : *homset struct* (*Ssrc struct M*) (*Strg struct M*) .
OS-isLBound struct R S L ⟶ *incl struct L M*)

lemma (**in** *OrderedSemigroupoid*) *isMeet-def*[*iff?*]:
assumes *R-t*[*intro*]: *R* : *a* ↔ *b*
assumes *S-t*[*intro*]: *S* : *a* ↔ *b*
assumes *M-t*[*intro*]: *M* : *a* ↔ *b*
shows *isMeet R S M* = (*isLBound R S M* & (*ALL L* : *a* ↔ *b* . *isLBound R S L* ⟶ *L* ⊑ *M*))
proof −
from *M-t* **have** [*simp*]: *src M* = *a* **by** *simp*
from *M-t* **have** [*simp*]: *trg M* = *b* **by** *simp*
show *?thesis* **by** (*unfold OS-isMeet-def*, *simp*)
qed

lemma (**in** *OrderedSemigroupoid*) *isMeet-expand*[*elim?*]:
assumes *m*: *isMeet R S M*
assumes *R-t*[*intro*]: *R* : *a* ↔ *b*
assumes *S-t*[*intro*]: *S* : *a* ↔ *b*
assumes *M-t*[*intro*]: *M* : *a* ↔ *b*
shows *isLBound R S M* & (*ALL L* : *a* ↔ *b* . *isLBound R S L* ⟶ *L* ⊑ *M*)
proof −
from *m R-t S-t M-t* **show** *?thesis* **by** (*rule-tac isMeet-def* [*THEN iffD1*])
qed

lemma (**in** *OrderedSemigroupoid*) *isMeet-isLBound*[*elim?*]:
assumes *m*: *isMeet R S M*
assumes *R-t*[*intro*]: *R* : *a* ↔ *b*
assumes *S-t*[*intro*]: *S* : *a* ↔ *b*
assumes *M-t*[*intro*]: *M* : *a* ↔ *b*
shows *isLBound R S M*

proof −
from *m* **have** *isLBound R S M* & (*ALL L* : *a* ↔ *b* . *isLBound R S L* ⟶ *L* ⊑ *M*)
by (*rule-tac isMeet-expand*, *best+*)
thus *?thesis* **by** *simp*
qed

lemma (**in** *OrderedSemigroupoid*) *isMeet-LBound*[*elim?*]:
assumes *m*: *isMeet R S M*
assumes *l*: *isLBound R S L*
assumes *R-t*[*intro*]: *R* : *a* ↔ *b*
assumes *S-t*[*intro*]: *S* : *a* ↔ *b*
assumes *M-t*[*intro*]: *M* : *a* ↔ *b*
assumes *L-t*[*intro*]: *L* : *a* ↔ *b*
shows *L* ⊑ *M*
proof −
from *m* **have** *isLBound R S M* & (*ALL L* : *a* ↔ *b* . *isLBound R S L* ⟶ *L* ⊑ *M*)
by (*rule-tac isMeet-expand*, *best+*)
then have *ALL L* : *a* ↔ *b* . *isLBound R S L* ⟶ *L* ⊑ *M* **by** *simp*
with *l L-t* **show** *?thesis* **by** *simp*
qed

lemma (**in** *OrderedSemigroupoid*) *isMeet-contract*:
assumes *l*: *isLBound R S M* & (*ALL L* : *a* ↔ *b* . *isLBound R S L* ⟶ *L* ⊑ *M*)
assumes *R-t*[*intro*]: *R* : *a* ↔ *b*
assumes *S-t*[*intro*]: *S* : *a* ↔ *b*
assumes *M-t*[*intro*]: *M* : *a* ↔ *b*
shows *isMeet R S M*
proof −
from *l R-t S-t M-t* **show** *?thesis* **by** (*rule-tac isMeet-def* [*THEN iffD2*])
qed

lemma (**in** *OrderedSemigroupoid*) *isMeet-intro*[*intro?*]:
assumes *b*: *isLBound R S M*
assumes *l*: *ALL L* : *a* ↔ *b* . *isLBound R S L* ⟶ *L* ⊑ *M*
assumes *R-t*[*intro*]: *R* : *a* ↔ *b*
assumes *S-t*[*intro*]: *S* : *a* ↔ *b*
assumes *M-t*[*intro*]: *M* : *a* ↔ *b*
shows *isMeet R S M*
proof −
from *b l* **have** *isLBound R S M* & (*ALL L* : *a* ↔ *b* . *isLBound R S L* ⟶ *L* ⊑ *M*)

by (*rule conjI*)
thus *?thesis* **by** (*rule-tac isMeet-contract, best+*)
qed

B.8.2 Upper Bounds

constdefs
OS-isUBound :: ('*o*, '*m*, '*r*)*OrderedSemigroupoid-scheme* ⇒ '*m* ⇒ '*m* ⇒ '*m* ⇒ *bool*
(*isUBound*₁ - - - [*1000,1000,1000*] *999*)
OS-isUBound struct R S J == *isMor struct J* &
(*let a* = *Ssrc struct J*; *b* = *Strg struct J in*
R : *homset struct a b* & *S* : *homset struct a b* &
incl struct R J & *incl struct S J*)

lemma (**in** *OrderedSemigroupoid*) *isUBound-def*:
isUBound R S J = (*Mor J* ∧ *R* : *src J* ↔ *trg J* ∧ *S* : *src J* ↔ *trg J* ∧ *R* ⊑ *J* & *S* ⊑ *J*)
by (*unfold OS-isUBound-def, simp add: Let-def*)

lemma (**in** *OrderedSemigroupoid*) *isUBound-expand*:
assumes *u*: *isUBound R S J*
assumes *R-t*[*intro*]: *R* : *a* ↔ *b* — unnecessary, but hygienic
assumes *S-t*[*intro*]: *S* : *a* ↔ *b*
assumes *M-t*[*intro*]: *J* : *a* ↔ *b*
shows *R* ⊑ *J* & *S* ⊑ *J*
by (*insert u, drule isUBound-def* [*THEN iffD1*], *simp*)

lemma (**in** *OrderedSemigroupoid*) *isUBound-expand-new*:
assumes *u*: *isUBound R S J*
assumes *R-t*[*intro*]: *R* : *a* ↔ *b* — unnecessary, but hygienic
assumes *S-t*[*intro*]: *S* : *a* ↔ *b*
assumes *M-t*[*intro*]: *J* : *a* ↔ *b*
shows *R* ⊑ *J* & *S* ⊑ *J*
proof
from *u* **have** (*Mor J* ∧ *R* : *src J* ↔ *trg J* ∧ *S* : *src J* ↔ *trg J* ∧ *R* ⊑ *J* & *S* ⊑ *J*)
by (*rule isUBound-def* [*THEN iffD1*])
thus *R* ⊑ *J* **by** *simp*
next
from *u* **have** (*Mor J* ∧ *R* : *src J* ↔ *trg J* ∧ *S* : *src J* ↔ *trg J* ∧ *R* ⊑ *J* & *S* ⊑ *J*)
by (*rule isUBound-def* [*THEN iffD1*])
thus *S* ⊑ *J* **by** *simp*

qed

lemma (**in** *OrderedSemigroupoid*) *isUBound-1*:
assumes *u*: *isUBound R S M*
assumes *R-t*[*intro*]: *R* : *a* ↔ *b*
assumes *S-t*[*intro*]: *S* : *a* ↔ *b*
assumes *M-t*[*intro*]: *M* : *a* ↔ *b*
shows *R* ⊑ *M*
proof −
from *u* **have** *R* ⊑ *M* & *S* ⊑ *M* **by** (*rule-tac isUBound-expand, best+*)
thus *?thesis* ..
qed

lemma (**in** *OrderedSemigroupoid*) *isUBound-2*:
assumes *u*: *isUBound R S M*
assumes *R-t*[*intro*]: *R* : *a* ↔ *b*
assumes *S-t*[*intro*]: *S* : *a* ↔ *b*
assumes *M-t*[*intro*]: *M* : *a* ↔ *b*
shows *S* ⊑ *M*
proof −
from *u* **have** *R* ⊑ *M* & *S* ⊑ *M* **by** (*rule isUBound-expand, best+*)
thus *?thesis* ..
qed

lemma (**in** *OrderedSemigroupoid*) *isUBound-contract*:
assumes *u*: *R* ⊑ *J* & *S* ⊑ *J*
assumes *R-t*[*intro*]: *R* : *a* ↔ *b*
assumes *S-t*[*intro*]: *S* : *a* ↔ *b*
assumes *J-t*[*intro*]: *J* : *a* ↔ *b*
shows *isUBound R S J*
proof (*subst isUBound-def, simp add: u Let-def, rule conjI*)
from *J-t* **show** *Mor J* **by** *simp*
next
from *J-t R-t S-t* **show** *R* ∈ *src J* ↔ *trg J* ∧ *S* ∈ *src J* ↔ *trg J* **by** *simp*
qed

lemma (**in** *OrderedSemigroupoid*) *isUBound-contract-new*:
assumes *u*: *R* ⊑ *J* & *S* ⊑ *J*
assumes *R-t*[*intro*]: *R* : *a* ↔ *b*
assumes *S-t*[*intro*]: *S* : *a* ↔ *b*

assumes *J-t*[*intro*]: $J : a \leftrightarrow b$
shows *isUBound R S J*
proof (*subst isUBound-def*, *rule conjI*)
from *J-t* **show** *Mor J* **by** *simp*
next
from *u J-t R-t S-t* **show** $R \in src\ J \leftrightarrow trg\ J \wedge S \in src\ J \leftrightarrow trg\ J \wedge R \sqsubseteq J\ \&\ S \sqsubseteq J$ **by** *simp*
qed

lemma (**in** *OrderedSemigroupoid*) *isUBound-intro*[*intro?*]:
assumes *u1*: $R \sqsubseteq M$
assumes *u2*: $S \sqsubseteq M$
assumes *R-t*[*intro*]: $R : a \leftrightarrow b$
assumes *S-t*[*intro*]: $S : a \leftrightarrow b$
assumes *M-t*[*intro*]: $M : a \leftrightarrow b$
shows *isUBound R S M*
proof −
from *u1 u2* **have** $R \sqsubseteq M\ \&\ S \sqsubseteq M$ **by** (*rule conjI*)
thus *?thesis* **by** (*rule isUBound-contract*, *best+*)
qed

constdefs
OS-isJoin :: ('*o*, '*m*, '*r*) *OrderedSemigroupoid-scheme* ⇒ '*m* ⇒ '*m* ⇒ '*m* ⇒ *bool*
(*isJoin* - - - [*1000*,*1000*,*1000*] *999*)
OS-isJoin struct R S J == *OS-isUBound struct R S J* &
(*ALL U*: *homset struct* (*Ssrc struct J*) (*Strg struct J*) .
OS-isUBound struct R S U ⟶ *incl struct J U*)

lemma (**in** *OrderedSemigroupoid*) *isJoin-def*:
assumes *R-t*[*intro*]: $R : a \leftrightarrow b$
assumes *S-t*[*intro*]: $S : a \leftrightarrow b$
assumes *J-t*[*intro*]: $J : a \leftrightarrow b$
shows *isJoin R S J* = (*isUBound R S J* ∧ (*ALL U* : $a \leftrightarrow b$. *isUBound R S U* ⟶ $J \sqsubseteq U$))
proof −
from *J-t* **have** *src J* = *a* **by** *simp*
moreover from *J-t* **have** *trg J* = *b* **by** *simp*
ultimately show *?thesis* **by** (*unfold OS-isJoin-def*, *simp*)
qed

lemma (**in** *OrderedSemigroupoid*) *isJoin-expand*[*elim?*]:
assumes *j*: *isJoin R S J*
assumes *R-t*[*intro*]: $R : a \leftrightarrow b$
assumes *S-t*[*intro*]: $S : a \leftrightarrow b$
assumes *J-t*[*intro*]: $J : a \leftrightarrow b$
shows *isUBound R S J* ∧ (*ALL U* : $a \leftrightarrow b$. *isUBound R S U* ⟶ $J \sqsubseteq U$)
proof −
from *j R-t S-t J-t* **show** *?thesis* **by** (*rule-tac isJoin-def* [*THEN iffD1*])
qed

lemma (**in** *OrderedSemigroupoid*) *isJoin-isUBound*[*elim?*]:
assumes *j*: *isJoin R S J*
assumes *R-t*[*intro*]: $R : a \leftrightarrow b$
assumes *S-t*[*intro*]: $S : a \leftrightarrow b$
assumes *J-t*[*intro*]: $J : a \leftrightarrow b$
shows *isUBound R S J*
proof −
from *j* **have** *isUBound R S J* ∧ (*ALL U* : $a \leftrightarrow b$. *isUBound R S U* ⟶ $J \sqsubseteq U$)
by (*rule isJoin-expand*, *best+*)
thus *?thesis* **by** *simp*
qed

lemma (**in** *OrderedSemigroupoid*) *isJoin-UBound*[*elim?*]:
assumes *j*: *isJoin R S J*
assumes *u*: *isUBound R S U*
assumes *R-t*[*intro*]: $R : a \leftrightarrow b$
assumes *S-t*[*intro*]: $S : a \leftrightarrow b$
assumes *J-t*[*intro*]: $J : a \leftrightarrow b$
assumes *U-t*[*intro*]: $U : a \leftrightarrow b$
shows $J \sqsubseteq U$
proof −
from *j* **have** (*isUBound R S J* ∧ (*ALL U* : $a \leftrightarrow b$. *isUBound R S U* ⟶ $J \sqsubseteq U$))
by (*rule isJoin-expand*, *auto*)
hence *ALL U* : $a \leftrightarrow b$. *isUBound R S U* ⟶ $J \sqsubseteq U$ **by** *simp*
with *u U-t* **show** *?thesis* **by** *simp*
qed

lemma (**in** *OrderedSemigroupoid*) *isJoin-contract*:
assumes *u*: *isUBound R S J* & (*ALL U* : $a \leftrightarrow b$. *isUBound R S U* ⟶ $J \sqsubseteq U$)
assumes *R-t*[*intro*]: $R : a \leftrightarrow b$

assumes *S-t[intro]*: $S : a \leftrightarrow b$
assumes *j-t[intro]*: $J : a \leftrightarrow b$
shows *isJoin R S J*
proof −
from *u* **show** *?thesis* **by** (*rule-tac isJoin-def* [*THEN iffD2*], *best+*)
qed

lemma (**in** *OrderedSemigroupoid*) *isJoin-intro[intro?]*:
assumes *b*: *isUBound R S J*
assumes *u*: $ALL\ U : a \leftrightarrow b\ .\ isUBound\ R\ S\ U \longrightarrow J \sqsubseteq U$
assumes *R-t[intro]*: $R : a \leftrightarrow b$
assumes *S-t[intro]*: $S : a \leftrightarrow b$
assumes *J-t[intro]*: $J : a \leftrightarrow b$
shows *isJoin R S J*
proof −
from *b u* **have** $isUBound\ R\ S\ J\ \&\ (ALL\ U : a \leftrightarrow b\ .\ isUBound\ R\ S\ U \longrightarrow J \sqsubseteq U)$
by (*rule conjI*)
thus *?thesis* **by** (*rule isJoin-contract*, *auto*)
qed

B.8.3 Predicate for Greatest Element

constdefs
OS-isTop :: ('*o*, '*m*, '*r*) *OrderedSemigroupoid-scheme* ⇒ '*m* ⇒ *bool* (*isTop*$_1$ - [*1000*] *999*)
OS-isTop s R == *isMor s R* & (*ALL S . S* : *homset s* (*Ssrc s R*) (*Strg s R*) ⟶ *incl s S R*)

lemma (**in** *OrderedSemigroupoid*) *isTop-def*:
$R : a \leftrightarrow b \Longrightarrow isTop\ R = (ALL\ S\ .\ S : a \leftrightarrow b \longrightarrow S \sqsubseteq R)$
by (*unfold OS-isTop-def*, *simp*)

lemma (**in** *OrderedSemigroupoid*) *isTop-expand*:
assumes *t*: *isTop R*
assumes *R-t[intro]*: $R : a \leftrightarrow b$
shows $ALL\ S\ .\ S : a \leftrightarrow b \longrightarrow S \sqsubseteq R$
proof −
from *t R-t* **show** *?thesis* **by** (*rule-tac isTop-def* [*THEN iffD1*])
qed

lemma (**in** *OrderedSemigroupoid*) *isTop-contract[intro?]*:

assumes *t*: $ALL\ S\ .\ S : a \leftrightarrow b \longrightarrow S \sqsubseteq R$
assumes *R-t[intro]*: $R : a \leftrightarrow b$
shows *isTop R*
proof −
from *t R-t* **show** *?thesis* **by** (*rule-tac isTop-def* [*THEN iffD2*])
qed

lemma (**in** *OrderedSemigroupoid*) *isTop*:
assumes *t*: *isTop R*
assumes *R-t[intro]*: $R : a \leftrightarrow b$
assumes *S-t[intro]*: $S : a \leftrightarrow b$
shows $S \sqsubseteq R$
proof −
from *t* **have** $ALL\ S\ .\ S : a \leftrightarrow b \longrightarrow S \sqsubseteq R$ **by** (*rule isTop-expand*, *best+*)
thus *?thesis* **by** *auto*
qed

B.8.4 Predicate fo Least Element

constdefs
OS-isBot :: ('*o*, '*m*, '*r*) *OrderedSemigroupoid-scheme* ⇒ '*m* ⇒ *bool* (*isBot*$_1$ - [*1000*] *999*)
OS-isBot s R == *isMor s R* & (*ALL S . S* : *homset s* (*Ssrc s R*) (*Strg s R*) ⟶ *incl s R S*)

lemma (**in** *OrderedSemigroupoid*) *isBot-def*:
$R : a \leftrightarrow b \Longrightarrow isBot\ R = (ALL\ S\ .\ S : a \leftrightarrow b \longrightarrow R \sqsubseteq S)$
by (*unfold OS-isBot-def*, *simp*)

lemma (**in** *OrderedSemigroupoid*) *isBot-expand*:
assumes *t*: *isBot R*
assumes *R-t[intro]*: $R : a \leftrightarrow b$
shows $ALL\ S\ .\ S : a \leftrightarrow b \longrightarrow R \sqsubseteq S$
proof −
from *t R-t* **show** *?thesis* **by** (*rule-tac isBot-def* [*THEN iffD1*])
qed

lemma (**in** *OrderedSemigroupoid*) *isBot-contract[intro?]*:
assumes *t*: $ALL\ S\ .\ S : a \leftrightarrow b \longrightarrow R \sqsubseteq S$
assumes *R-t[intro]*: $R : a \leftrightarrow b$
shows *isBot R*

proof −
from *t R-t* **show** *?thesis* **by** (*rule-tac isBot-def* [*THEN iffD2*])
qed

lemma (**in** *OrderedSemigroupoid*) *isBot*:
assumes *t*: *isBot R*
assumes *R-t*[*intro*]: $R : a \leftrightarrow b$
assumes *S-t*[*intro*]: $S : a \leftrightarrow b$
shows $R \sqsubseteq S$
proof −
from *t* **have** *ALL S* . $S : a \leftrightarrow b \longrightarrow R \sqsubseteq S$ **by** (*rule isBot-expand*, *best+*)
thus *?thesis* **by** *auto*
qed

lemma (**in** *OrderedSemigroupoid*) *isbot-incl-isbot*:
assumes [*intro*,*simp*]: *isBot P*
assumes [*intro*,*simp*]: $P : a \leftrightarrow b$
assumes [*intro*,*simp*]: $R : a \leftrightarrow b$
assumes *incl*: $R \sqsubseteq P$
shows *isBot R*
proof −
have *ALL S* . $S : a \leftrightarrow b \longrightarrow R \sqsubseteq S$
proof (*intro strip*)
have *ALL S* . $S : a \leftrightarrow b \longrightarrow P \sqsubseteq S$ **by** (*rule isBot-expand*, *auto*)
fix *S*
assume [*intro*,*simp*]: *S*: $a \leftrightarrow b$
from *incl* **have** $R \sqsubseteq P$ **by** *simp*
also have $P \sqsubseteq S$ **by** (*rule-tac isBot*, *best+*)
finally show $R \sqsubseteq S$ **by** *best+*
qed
thus *?thesis* **by** (*rule isBot-contract*, *best*)
qed

end

B.9 Idempotent Subidentities

theory *ISIdSemi*
imports *OrdSemiBounds*

begin

Multiplicatively idempotent subidentities can serve as tests in KAT and as domain elements in KAD [Desharnais-Moeller-Struth-2003].

B.9.1 Definition: Multiplicatively Idempotent Subidentities

constdefs
OS-isISId :: ('*o*, '*m*, '*r*) *OrderedSemigroupoid-scheme* ⇒ '*m* ⇒ *bool* (*isISId*$_1$ - [*1000*] *999*)
OS-isISId s R == *if isMor s R* & (*Ssrc s R* = *Strg s R*)
then OS-isSId s R & *cmp s R R* = *R*
else arbitrary

lemma (**in** *OrderedSemigroupoid*) *isISId-def*:
$R : a \leftrightarrow a \Longrightarrow$ *isISId R* = (*isSId R* & $R \odot R = R$)
by (*unfold OS-isISId-def*, *simp*)

lemma (**in** *OrderedSemigroupoid*) *isISId-intro*[*intro*]:
⟦ $R \odot R = R$; *isSId R*; $R : a \leftrightarrow a$ ⟧ ⟹ *isISId R*
by (*rule-tac isISId-def* [*THEN iffD2*], *auto*)

lemma (**in** *OrderedSemigroupoid*) *isISId-0*:
⟦ *isISId R*; $R : a \leftrightarrow a$ ⟧ ⟹ *isSId R* & $R \odot R = R$
by (*rule-tac isISId-def* [*THEN iffD1*])

lemma (**in** *OrderedSemigroupoid*) *isISId-1*[*intro*,*simp*]:
⟦ *isISId R*; $R : a \leftrightarrow a$ ⟧ ⟹ *isSId R*
by (*drule-tac isISId-0*, *auto*)

lemma (**in** *OrderedSemigroupoid*) *isISId-2*[*simp*]:
⟦ *isISId R*; $R : a \leftrightarrow a$ ⟧ ⟹ $R \odot R = R$
by (*drule-tac isISId-0*, *auto*)

B.9.2 Definition: Set of Multiplicatively Idempotent Subidentities

constdefs
OS-ISId :: ('*o*, '*m*, '*r*) *OrderedSemigroupoid-scheme* ⇒ '*o* ⇒ '*m set* (*ISId*$_1$ - [*1000*] *999*)
OS-ISId s a == *Collect* (λ *m* . *m* : *homset s a a* & *OS-isISId s m*)

lemma (in *OrderedSemigroupoid*) *ISId-def*:
ISId $a = \{\ m\ .\ m : a \mapsto a\ \&\ isISId\ m\ \}$
by (*unfold OS-ISId-def*, *simp*)

lemma (in *OrderedSemigroupoid*) *ISId-intro*[*intro*]:
$[\![\ R : a \mapsto a;\ isISId\ R\ ]\!] \Longrightarrow R : ISId\ a$
by (*unfold OS-ISId-def*, *simp*)

lemma (in *OrderedSemigroupoid*) *ISId-homset*[*intro*,*simp*]:
$R : ISId\ a \Longrightarrow R : a \mapsto a$
by (*unfold OS-ISId-def*, *simp*)

lemma (in *OrderedSemigroupoid*) *ISId*[*intro?*,*simp*]:
$R : ISId\ a \Longrightarrow isISId\ R$
by (*unfold OS-ISId-def*, *simp*)

This lemma is added to help proving those lemmas which were proved via Id

lemma (in *OrderedSemigroupoid*) *isid-issid*[*intro*, *simp*]:
assumes [*intro*,*simp*]: $R : ISId\ a$
shows *isSId R*
proof −
have *isISId R* **by** (*rule ISId*, *best*)
moreover have $R : a \mapsto a$ **by** (*simp*)
ultimately show *?thesis* **by** *auto*
qed

B.9.3 Some Useful Properties

The following lemmas are going to be used in dealing with preimage and image operations later as noted in [J. Desharnais, et. al.]

The following lemma maybe useful so that I do not do it for each proof.

lemma (in *OrderedSemigroupoid*) *isid-incl-eq1*:
assumes *incl*: $P \odot R \sqsubseteq R \mathbin{⨾} Q$
assumes [*intro*, *simp*]: $P : ISId\ a$
assumes [*intro*, *simp*]: $Q : ISId\ b$
assumes [*intro*, *simp*]: $R : a \mapsto b$
shows $P \odot R = P \odot R \odot Q$

proof −
have $P \odot R \sqsubseteq P \odot R \odot Q$
proof −
have $P : a \mapsto a$ **by** (*rule ISId-homset*, *best*)
moreover have *isISId P* **by** (*rule ISId*, *best*)
ultimately have $P = P \odot P$ **by** (*rule-tac isISId-2* [*THEN sym*], *best*)
hence $P \odot R = (P \odot P) \odot R$ **by** *auto*
also have ... $= P \odot P \odot R$ **by** (*rule cmp-assoc*, *best+*)
also from *incl* **have** ... $\sqsubseteq P \odot R \odot Q$ **by** (*rule comp-incl-mon2*, *best+*)
finally show *?thesis* **.**
qed
moreover have $P \odot R \odot Q \sqsubseteq P \odot R$
proof −
have $P \odot R \odot Q = (P \odot R) \odot Q$ **by** (*rule cmp-assoc-sym*, *best+*)
also have ... $\sqsubseteq P \odot R$ **by** *auto*
show *?thesis* **by** *auto*
qed
ultimately show *?thesis* **by**(*rule incl-antisym*, *best+*)
qed

lemma (in *OrderedSemigroupoid*) *isid-incl-eq2*:
assumes *eq*: $P \odot R = P \odot R \odot Q$
assumes [*intro*]: $P : ISId\ a$
assumes [*intro*]: $Q : ISId\ b$
assumes [*intro*]: $R : a \mapsto b$
shows $P \odot R \sqsubseteq R \mathbin{⨾} Q$
proof −
from *eq* **have** $P \odot R = P \odot (R \odot Q)$ **by** *simp*
also have ... $\sqsubseteq (R \odot Q)$ **by** (*auto*)
finally show *?thesis* **by** (*auto*)
qed

lemma (in *OrderedSemigroupoid*) *isid-incl-eq*:
assumes [*intro*]: $P : ISId\ a$
assumes [*intro*]: $Q : ISId\ b$
assumes [*intro*]: $R : a \mapsto b$
shows $(P \odot R \sqsubseteq R \odot Q) = (P \odot R = P \odot R \odot Q)$
proof −
have $P \odot R \sqsubseteq R \odot Q \Longrightarrow P \odot R = P \odot R \odot Q$ **by** (*rule isid-incl-eq1*, *auto*)
moreover have $P \odot R = P \odot R \odot Q \Longrightarrow P \odot R \sqsubseteq R \odot Q$ **by** (*rule isid-incl-eq2*, *auto*)

ultimately show *?thesis* **by** (*rule iffI*, *best*)
qed

lemma (**in** *OrderedSemigroupoid*) *incl-isid-eq1a*:
assumes *incl*: $R \odot P \sqsubseteq Q \odot R$
assumes [*intro*]: P : *ISId* b
assumes [*intro*]: Q : *ISId* a
assumes [*intro*]: $R : a \leftrightarrow b$
shows $R \odot P \sqsubseteq Q \odot R \odot P$
proof −
have [*intro*, *simp*]: *isISId* P **by** (*rule ISId*, *best*)
have $P : b \leftrightarrow b$ **by** (*rule ISId-homset*, *best*)
hence $P = P \odot P$ **by** (*rule-tac isISId-2* [*THEN sym*], *best*)
hence $R \odot P = R \odot P \odot P$ **by** *simp*
hence $R \cdot P = (R \cdot P) \cdot P$ **by** (*subst cmp-assoc*, *best+*)
also from *incl* **have** $\ldots \sqsubseteq (Q \odot R) \odot P$ **by** (*rule comp-incl-mon1*, *best+*)
finally show *?thesis* **by** (*subst cmp-assoc-sym*, *best+*)
qed

lemma (**in** *OrderedSemigroupoid*) *incl-isid-eq1b*:
assumes *incl*: $R \odot P \sqsubseteq Q \odot R$
assumes [*intro*]: P : *ISId* b
assumes [*intro*]: Q : *ISId* a
assumes [*intro*]: $R : a \leftrightarrow b$
shows $Q \odot R \odot P \sqsubseteq R \odot P$
proof −
have [*intro*, *simp*]: *isISId* Q **by** (*rule ISId*, *best*)
have $Q \odot R \odot P \sqsubseteq Q \odot (R \odot P)$ **by** (*auto*)
thus *?thesis* **by** (*best*)
qed

lemma (**in** *OrderedSemigroupoid*) *incl-isid-eq1*:
assumes *incl*: $R \odot P \sqsubseteq Q \odot R$
assumes [*intro*]: P : *ISId* b
assumes [*intro*]: Q : *ISId* a
assumes [*intro*]: $R : a \leftrightarrow b$
shows $R \odot P = Q \odot R \odot P$
proof −
from *incl* **have** $R \odot P \sqsubseteq Q \odot R \odot P$ **by** (*rule incl-isid-eq1a*, *best+*)
moreover from *incl* **have** $Q \cdot R \cdot P \sqsubseteq R \cdot P$ **by** (*rule incl-isid-eq1b*, *best+*)

ultimately show *?thesis* **by**(*rule incl-antisym*, *best+*)
qed

lemma (**in** *OrderedSemigroupoid*) *incl-isid-eq2*:
assumes *eq*: $R \odot P = Q \odot R \mathbin{:} P$
assumes [*intro*]: P : *ISId* b
assumes [*intro*]: Q : *ISId* a
assumes [*intro*]: $R : a \leftrightarrow b$
shows $R \mathbin{:} P \sqsubseteq Q \odot R$
proof −
have [*intro*, *simp*]: *isISId* P **by** (*rule ISId*, *best*)
from *eq* **have** $R \mathbin{:} P = Q \odot (R \odot P)$ **by** *simp*
hence $R \odot P = (Q \odot R) \odot P$ **by** (*subst cmp-assoc*, *best+*)
also have $\ldots \sqsubseteq (Q \odot R)$ **by** (*best+*)
finally show *?thesis* **by** (*best*)
qed

lemma (**in** *OrderedSemigroupoid*) *incl-isid-eq*:
assumes [*intro*]: P : *ISId* b
assumes [*intro*]: Q : *ISId* a
assumes [*intro*]: $R : a \leftrightarrow b$
shows $(R \odot P = Q \odot R \odot P) = (R \odot P \sqsubseteq Q \odot R)$
proof −
have $R \odot P = Q \odot R \odot P \Longrightarrow R \odot P \sqsubseteq Q \mathbin{:} R$ **by** (*rule incl-isid-eq2*, *auto*)
moreover have $R \odot P \sqsubseteq Q \odot R \Longrightarrow R \odot P = Q \odot R \odot P$ **by** (*rule incl-isid-eq1*, *auto*)
ultimately show *?thesis* **by** (*rule iffI*, *best*)
qed

end

B.10 Ordered Semigroupoids with Predomain

theory *PreDomSemi*
imports *SemiAllRecord ISIdSemi*
begin

B.10.1 Definitions

Desharnais introduced three equivalent axiomatizations of the domain operations on the class of Test- semiring, one of these involves the complement operator. Since we only have multiplicatively idempotent subidentities, we can only use those axiomatizations that do not involve the complement operator. We will introduce one axiomatization then show that the other can be derive from that.

locale *PreDomSemi = OrderedSemigroupoid PDS +*
assumes *dom-ISId[intro,simp]*: $R : a \leftrightarrow b \Longrightarrow dom\ R : ISId\ a$
assumes *dom-self[intro,simp]*: $R : a \leftrightarrow b \Longrightarrow R \sqsubseteq dom\ R \odot R$
assumes *dom-ISId-cmp[intro,simp]*: $[\![R : a \leftrightarrow b; P : ISId\ a]\!] \Longrightarrow dom(P \odot R) \sqsubseteq P$

The axiom *dom-homset* can be written as a lemma using *dom-ISId* and *ISId-homset* as W.K. pointed out.

lemma (in *PreDomSemi*) *dom-homset[intro,simp]*:
assumes *[intro]*: $R : a \leftrightarrow b$
shows $dom\ R : a \leftrightarrow a$
proof −
have $dom\ R : ISId\ a$ **by** *best*
thus *?thesis* **by** (*rule ISId-homset*)
qed

lemma (in *PreDomSemi*) *llp1*:
assumes *r [intro]*: $R : a \leftrightarrow b$
assumes *p [intro]*: $P : ISId\ a$
assumes *i*: $dom\ R \sqsubseteq P$
shows $R \sqsubseteq P \odot R$
proof −
have $R \sqsubseteq dom\ R \odot R$ **by** (*rule dom-self*, *auto*)
also from *i* **have** $\ldots \sqsubseteq P \odot R$ **by** *auto*
finally show *?thesis* **by** *best+*
qed

lemma (in *PreDomSemi*) *llp2*:
assumes *nn1 [intro]*: $R : a \leftrightarrow b$
assumes *nn2 [intro]*: $P : ISId\ a$
assumes *incl-isid*: $R \sqsubseteq P \odot R$
shows *nn3*:$dom\ R \sqsubseteq P$
proof −

have $R = P \odot R$
proof −
from *incl-isid* **have** $R \sqsubseteq P \odot R$ **by** *simp*
moreover have $P \odot R \sqsubseteq R$
proof −
have *[intro]*: *isISId P* **by** (*rule ISId*, *best*)
show $P \odot R \sqsubseteq R$ **by** (*rule isSId-left*, *best+*)
qed
ultimately show *?thesis* **by** (*rule incl-antisym*, *best+*)
qed
also have $dom(P \odot R) \sqsubseteq P$ **by** (*rule dom-ISId-cmp*, *best+*)
finally show *?thesis* .
qed

We derive *llp* from *dom-self* and *dom-ISId-cmp*. This will be useful later in calculating properties of the domain operator.

lemma (in *PreDomSemi*) *llp*:
assumes *r[intro]*: $R : a \leftrightarrow b$
assumes *p[intro]*: $P : ISId\ a$
shows $(dom\ R \sqsubseteq P) = (R \sqsubseteq P \odot R)$
proof −
from *r p* **have** $dom\ R \sqsubseteq P \Longrightarrow R \sqsubseteq P \odot R$ **by** (*rule llp1*)
moreover from *r p* **have** $R \sqsubseteq P \odot R \Longrightarrow dom\ R \sqsubseteq P$ **by** (*rule llp2*)
ultimately show *?thesis* **by** (*rule iffI*, *best*)
qed

lemma (in *PreDomSemi*)*isSId-dom[intro, simp]*:
assumes *[intro]*: W: $a \leftrightarrow b$
shows *isSId* $(dom\ W)$
proof −
have $dom\ W : ISId\ a$ **by** (*rule dom-ISId*, *auto*)
thus *?thesis* **by** *auto*
qed

lemma (in *PreDomSemi*) *isid-isbot*:
assumes *[intro, simp]*: *isBot W*
assumes *[intro, simp]*: $W : a \leftrightarrow a$

shows $W : ISId\ a$
proof −
have *isSId W*
proof (*rule-tac j=dom W* **in** *isSId-intro2*, *auto*)
show $W \sqsubseteq dom\ W$ **by** (*rule isBot*, *auto*)
qed
moreover have $W \odot W = W$
proof −
have $W \odot W \sqsubseteq W$
proof −
have $W \sqsubseteq dom\ W$ **by** (*rule isBot*, *auto*)
hence $W \odot W \sqsubseteq W \odot dom\ W$ **by** (*rule comp-incl-mon2*, *best+*)
also have $\ldots \sqsubseteq W$ **by** (*auto*)
finally show *?thesis* **by** *best+*
qed
moreover have $W \sqsubseteq W \odot W$ **by** (*rule isBot*, *auto*)
ultimately show *?thesis* **by** (*rule incl-antisym*, *best+*)
qed
ultimately have *isISId W* **by** (*rule-tac isISId-intro*, *best+*)
thus *?thesis* **by** (*rule-tac ISId-intro*, *best+*)
qed

B.10.2 Algebraic Properties of the domain operator

The predomain operator is fully strict.

lemma (**in** *PreDomSemi*) *dom-strict1*:
assumes [*intro*]: *isBot* (*dom R*)
assumes *cmp-isidbot*: $isBot\ (dom\ R \odot R)$
assumes [*intro,simp*]: $R : a \leftrightarrow b$
shows *isBot R*
proof −
have $isBot\ (dom\ R \odot R)$ **by** (*rule cmp-isidbot*)
also have $dom\ R \odot R = R$
proof −
have $R \sqsubseteq dom\ R \odot R$ **by** (*rule-tac llp* [*THEN iffD1*], *best+*)
moreover have $dom\ R \odot R \sqsubseteq R$ **by** *auto*
ultimately show *?thesis* **by** (*rule-tac incl-antisym*, *best+*)
qed
finally show *?thesis* **by** *auto*

qed

B.10.3 Algebraic Properties of the domain operator

The predomain operator is fully strict.

lemma (**in** *PreDomSemi*) *dom-strict2*:
assumes *cmp-isidbot*: $isBot\ (P \odot R)$
assumes [*intro*]: *isBot P*
assumes [*intro*]: *isBot R*
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $P : ISId\ a$
shows $isBot\ (dom\ R)$
proof −
have $isBot\ (P \odot R)$ **by** (*rule cmp-isidbot*)
hence $R \sqsubseteq P \odot R$ **by** (*rule-tac isBot*, *best+*)
hence $dom\ R \sqsubseteq P$ **by** (*rule-tac llp* [*THEN iffD2*], *best+*)
thus *?thesis* **by** (*rule-tac isbot-incl-isbot*, *auto*)
qed

lemma (**in** *PreDomSemi*) *dom-strict*:
assumes [*intro*]: $isBot\ (dom\ R \odot R)$
assumes [*intro*]: $isBot\ (P \odot R)$
assumes [*intro*]: *isBot P*
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $P : ISId\ a$
shows $isBot\ (dom\ R) = isBot\ R$
proof −
have $isBot\ (dom\ R) \Longrightarrow isBot\ R$ **by** (*rule-tac dom-strict1*, *best+*)
moreover have $isBot\ R \Longrightarrow isBot\ (dom\ R)$ **by** (*rule-tac P=P* **in** *dom-strict2*, *auto*)
ultimately show *?thesis* **by** (*rule iff1*, *best+*)
qed

The predomain operator is an identity on multiplicatively idempotent subidentities.

lemma (**in** *PreDomSemi*) *dom-isISId-eq*:
assumes [*intro*]: $P : ISId\ a$
shows $dom\ P = P$
proof (*rule incl-antisym*, *auto*)
show $dom\ P \sqsubseteq P$

proof −
have *dom* $(P \odot P) \sqsubseteq P$ **by** (*rule-tac dom-ISId-cmp*, *best+*)
also have $P \odot P = P$
proof (*rule isISId-2*, *auto*)
show *isISId P* **by**(*rule-tac ISId*, *auto*)
qed
finally show *?thesis* .
qed
next
show $P \sqsubseteq$ *dom P*
proof −
have [*intro*, *simp*]: *isISId P* **by** (*rule-tac ISId*, *auto*)
have $P \sqsubseteq$ *dom P* $\odot$ *P* **by** (*rule dom-self*, *best*)
also have ... $\sqsubseteq$ *dom P* **by** (*best+*)
finally show *?thesis* **by** *best+*
qed
qed

The predomain operator is idempotent.

lemma (**in** *PreDomSemi*) *dom-idemp*:
assumes [*intro*]: $R : a \leftrightarrow b$
shows *dom* (*dom R*) = *dom R*
by (*rule dom-isISId-eq*, *best+*)

The predomain operator is a left invariant.

lemma (**in** *PreDomSemi*) *dom-left-inv*:
assumes [*intro*]: $R : a \leftrightarrow b$
shows $R =$ *dom R* $\odot$ *R*
proof −
have $R \sqsubseteq$ *dom R* $\odot$ *R* **by** (*rule dom-self*, *auto*)
moreover have ... $\sqsubseteq R$ **by** *auto*
ultimately show *?thesis* **by** (*rule incl-antisym*, *best+*)
qed

The predomain operator satisfies a decomposition law.

lemma (**in** *PreDomSemi*) *dom-decomp*:
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $S : b \leftrightarrow c$
shows *dom* $(R \odot S) \sqsubseteq$ *dom* $(R \odot$ *dom S*)

proof −
have $R \odot S \sqsubseteq$ *dom* $(R \odot$ (*dom S*))$\odot R \odot S$
proof −
have $R \odot S \sqsubseteq$ (*dom* $(R \mathbin{;}$ *dom S*) $\odot$ $(R \odot$ (*dom S*))) $\odot S$
proof −
have $R \odot S \sqsubseteq R \odot$ *dom S* $\mathbin{;} S$ **by** (*rule comp-incl-mon2*, *rule dom-self*, *best+*)
hence $R \odot S \sqsubseteq (R \odot$ *dom S*) $\odot S$ **by** (*subst cmp-assoc*, *best+*)
also have ... $\sqsubseteq$ (*dom* $(R \odot$ *dom S*) $\odot$ $(R \odot$ (*dom S*))) $\odot S$
by (*rule-tac comp-incl-mon1*, *rule dom-self*, *best+*)
also show *?thesis* **by** (*rule-tac calculation*, *best+*)
qed
hence $R \odot S \sqsubseteq$ *dom* $(R \odot$ *dom S*) $\odot$ $(R \mathbin{;}$ (*dom S*)) $\odot S$ **by** (*subst cmp-assoc-sym*, *best+*)
hence $R \odot S \sqsubseteq$ *dom* $(R \odot$ *dom S*) $\mathbin{;} R \odot$ (*dom S*) $\mathbin{;} S$ **by** (*subst cmp-assoc-sym*, *auto*)
also have (*dom S*) $\cdot S = S$ **by** (*rule dom-left-inv* [*THEN sym*], *auto*)
finally show *?thesis* .
qed
thus *?thesis* **by** (*rule-tac llp* [*THEN iffD2*], *best+*)
qed

B.10.4 Preimage Operator

constdefs
preimage :: ('*o*, '*m*, '*r*) *SemiAllegory-scheme* ⇒ '*m* ⇒ '*m* ⇒ '*m* (δ_1 - - [*1000*,*1000*] *999*)
preimage s R P == *if isMor s R* & *OS-isISId s P*
then All-dom s (*cmp s R P*)
else arbitrary

lemma (**in** *PreDomSemi*) *preimage-def*:
⟦ $R : a \leftrightarrow b$; P : *ISId b* ⟧ ⟹ δ *R P* = *dom* $(R \mathbin{;} P)$
by (*unfold preimage-def*,*auto*)

lemma (**in** *PreDomSemi*) *preimage-homset*[*intro*,*simp*]:
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: P : *ISId b*
shows δ *R P* : $a \leftrightarrow a$
proof −
have δ *R P* = *dom* $(R \odot P)$ **by** (*rule preimage-def*, *best+*)
also have ... : $a \leftrightarrow a$ **by** (*rule dom-homset*, *best*)
finally show *?thesis* .

qed

lemma (**in** *PreDomSemi*) *preimage-ISId*:
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $P : ISId\ b$
shows $\delta\ R\ P : ISId\ a$
proof −
have $\delta\ R\ P = dom\ (R \odot P)$ **by** (*rule preimage-def*, *best*+)
also have $\ldots : ISId\ a$ **by** (*rule dom-ISId*, *best*)
finally show *?thesis* .
qed

Lemma preimage connects the preimage operator with llp.

lemma (**in** *PreDomSemi*) *preimage1*:
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $P : ISId\ b$
assumes [*intro*]: $Q : ISId\ a$
assumes *preimage-incl*: $\delta\ R\ P \sqsubseteq Q$
shows $R \odot P \sqsubseteq Q \odot R$
proof −
have [*intro,simp*]: *isISId P* **by** (*rule-tac ISId*, *auto*)
have $\delta\ R\ P = dom\ (R \cdot P)$ **by** (*rule preimage-def*, *auto*)
with *preimage-incl* **have** [*intro*]: $\ldots \sqsubseteq Q$ **by** *simp*
have $R \odot P \sqsubseteq Q \odot (R \odot P)$ **by** (*rule-tac llp* [*THEN iffD1*], *best*+)
also have $\ldots \sqsubseteq Q \odot R$ **by** *auto*
finally show *?thesis* **by** *best*+
qed

lemma (**in** *PreDomSemi*) *preimage2*:
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $P : ISId\ b$
assumes [*intro*]: $Q : ISId\ a$
assumes *incl*: $R \odot P \sqsubseteq Q \odot R$
shows $\delta\ R\ P \sqsubseteq Q$
proof −
have [*intro,simp*]: *isISId P* **by** (*rule-tac ISId*, *auto*)
from *incl* **have** [*intro*]: $R \odot P \sqsubseteq Q \odot (R \odot P)$ **by** (*rule-tac incl-isid-eq1a*, *best*+)
have $dom\ (R \odot P) \sqsubseteq Q$ **by** (*rule-tac llp* [*THEN iffD2*], *best*+)
also have $dom\ (R \odot P) = \delta\ R\ P$ **by** (*rule preimage-def* [*THEN sym*], *best*+)
finally show *?thesis* .

qed

lemma (**in** *PreDomSemi*) *preimage*:
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $P : ISId\ b$
assumes [*intro*]: $Q : ISId\ a$
shows $(\delta\ R\ P \sqsubseteq Q) = (R \odot P \sqsubseteq Q \odot R)$
proof −
have $\delta\ R\ P \sqsubseteq Q \Longrightarrow R \odot P \sqsubseteq Q \odot R$ **by** (*rule preimage1*, *best*+)
moreover have $R \odot P \sqsubseteq Q \odot R \Longrightarrow \delta\ R\ P \sqsubseteq Q$ **by** (*rule preimage2*, *best*+)
ultimately show *?thesis* **by** (*rule iffI*, *best*+)
qed

end

B.11 Ordered Semigroupoids with Monotonic PreDomain

theory *MonPreDomSemi*
imports *PreDomSemi*
begin

B.11.1 Definition

We introduce monotonicity of the domain operator.

locale *MonPreDomSemi* = *PreDomSemi MPDS* +
assumes *dom-incl-mon*[*intro*]: $[\![\ R : a \leftrightarrow b;\ S : a \leftrightarrow b;\ R \sqsubseteq S\ ]\!] \Longrightarrow dom\ R \sqsubseteq dom\ S$

end

B.12 Ordered Semigroupoids with Domain

theory *DomSemi*
imports *MonPreDomSemi*
begin

locale *DomSemi* = *MonPreDomSemi DS* +
assumes *dom-local*[*intro*, *simp*]:⟦$R : a \leftrightarrow b$; $S : b \leftrightarrow c$⟧ ⟹ $dom(R \odot dom\ S) \sqsubseteq dom(R \odot S)$

lemma (**in** *DomSemi*) *dom-cmp*:
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $S : b \leftrightarrow c$
shows $dom(R \odot dom\ S) = dom(R \odot S)$
proof −
have $dom(R \odot dom\ S) \sqsubseteq dom(R \odot S)$ **by** *auto*
moreover have $dom(R \odot S) \sqsubseteq dom(R \odot dom\ S)$ **by** (*rule dom-decomp*, *auto*)
ultimately show *?thesis* **by** (*rule incl-antisym*, *auto*)
qed

end

B.13 Ordered Semigroupoid with Prerange

theory *PreRanSemi*
imports *SemiAllRecord ISIdSemi*
begin

B.13.1 Definitions

Desharnais introduced three equivalent axiomatizations of the Range operations on the class of Test- semiring, one of these involves the complement operator. Since we only have multiplicatively idempotent subidentities, we can only use those axiomatizations that do not involve the complement operator. We will introduce one axiomatization then show that the other can be derive from that.

locale *PreRanSemi* = *OrderedSemigroupoid PRS+*
assumes *ran-ISId*[*intro*,*simp*]: $R : a \leftrightarrow b \Longrightarrow rang\ R : ISId\ b$
assumes *ran-self*[*intro*,*simp*]: $R : a \leftrightarrow b \Longrightarrow R \sqsubseteq R \odot rang\ R$
assumes *ran-ISId-cmp*[*intro*,*simp*]: ⟦ $R : a \leftrightarrow b$; $P : ISId\ b$ ⟧ ⟹ $rang\ (R \odot P) \sqsubseteq P$

The axiom *ran-homset* can be written as a lemma using *ran-ISId* and *ISId-homset* as W.K. pointed out.

lemma (**in** *PreRanSemi*) *ran-homset*[*intro*,*simp*]:

assumes [*intro*]: $R : a \leftrightarrow b$
shows $rang\ R : b \leftrightarrow b$
proof −
have $rang\ R : ISId\ b$ **by** *best*
thus *?thesis* **by** (*rule ISId-homset*)
qed

lemma (**in** *PreRanSemi*) *llp1*:
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $P : ISId\ b$
assumes *i*: $rang\ R \sqsubseteq P$
shows $R \sqsubseteq R \odot P$
proof −
have $R \sqsubseteq R \odot rang\ R$ **by** (*rule ran-self*, *best*)
also from *i* **have** $\ldots \sqsubseteq R \odot P$ **by** *auto*
finally show *?thesis* **by** *best+*
qed

lemma (**in** *PreRanSemi*) *llp2*:
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $P : ISId\ b$
assumes *incl-isid*: $R \sqsubseteq R \odot P$
shows $rang\ R \sqsubseteq P$
proof −
have $R = R \odot P$
proof −
from *incl-isid* **have** $R \sqsubseteq R \odot P$ **by** *simp*
moreover have $R \odot P \sqsubseteq R$
proof −
have [*intro*]: *isISId P* **by** (*rule ISId*, *best*)
show $R \odot P \sqsubseteq R$ **by** (*rule isSId-right*, *best+*)
qed
ultimately show *?thesis* **by** (*rule incl-antisym*, *best+*)
qed
also have $rang(R \odot P) \sqsubseteq P$ **by** (*rule ran-ISId-cmp*, *best+*)
finally show *?thesis* .
qed

We derive *llp* from *dom-self* and *dom-ISId-cmp*. This will be useful later in calculating properties of the range operator.

lemma (in *PreRanSemi*) *llp*:
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: P : *ISId* b
shows $(rang\ R \sqsubseteq P) = (R \sqsubseteq R \odot P)$
proof −
have $rang\ R \sqsubseteq P \Longrightarrow R \sqsubseteq R \odot P$ **by** (*rule llp1*, *best+*)
moreover have $R \sqsubseteq R \odot P \Longrightarrow rang\ R \sqsubseteq P$ **by** (*rule llp2*, *best+*)
ultimately show *?thesis* **by** (*rule iffI*, *best+*)
qed

lemma (in *PreRanSemi*)*isSId-ran*[*intro*, *simp*]:
assumes [*intro*]: W: $a \leftrightarrow b$
shows *isSId* (*rang* W)
proof −
have *rang* W : *ISId* b **by** (*rule ran-ISId*, *auto*)
thus *?thesis* **by** *auto*
qed

lemma (in *PreRanSemi*) *isid-isbot*:
assumes [*intro*, *simp*]: *isBot* W
assumes [*intro*, *simp*]: $W : a \leftrightarrow a$
shows W : *ISId* a
proof −
have *isSId* W
proof (*rule-tac* j=*rang* W **in** *isSId-intro2*, *auto*)
show $W \sqsubseteq rang\ W$ **by** (*rule isBot*, *auto*)
qed
moreover have $W \odot W = W$
proof −
have $W \odot W \sqsubseteq W$
proof −
have $W \sqsubseteq rang\ W$ **by** (*rule isBot*, *auto*)
hence $W \odot W \sqsubseteq W \odot rang\ W$ **by** (*rule comp-incl-mon2*, *best+*)
also have ... $\sqsubseteq W$ **by** (*auto*)
finally show *?thesis* **by** *auto*
qed
moreover have $W \sqsubseteq W \odot W$ **by** (*rule isBot*, *auto*)
ultimately show *?thesis* **by** (*rule incl-antisym*, *best+*)
qed

ultimately have *isISId* W **by** (*rule-tac isISId-intro*, *best+*)
thus *?thesis* **by** (*rule-tac ISId-intro*, *best+*)
qed

B.13.2 Algebraic Properties of the range operator

The prerange operator is fully strict.

lemma (in *PreRanSemi*) *ran-strict1*:
assumes [*intro*]: *isBot* (*rang* R)
assumes *cmp-isidbot*: *isBot* ($R \odot$ *rang* R)
assumes [*intro*,*simp*]: $R : a \leftrightarrow b$
shows *isBot* R
proof −
have *isBot* ($R \odot$ *rang* R) **by** (*rule cmp-isidbot*)
also have $R \odot rang\ R = R$
proof −
have $R \sqsubseteq R \odot rang\ R$ **by** (*rule-tac llp* [*THEN iffD1*], *best+*)
moreover have $R \odot rang\ R \sqsubseteq R$ **by** *auto*
ultimately show *?thesis* **by** (*rule-tac incl-antisym*, *best+*)
qed
finally show *?thesis* .
qed

lemma (in *PreRanSemi*) *ran-strict2*:
assumes *cmp-isidbot*: *isBot* ($P \odot R$)
assumes [*intro*]: *isBot* P
assumes [*intro*]: *isBot* R
assumes [*intro*]: $P : a \leftrightarrow b$
assumes [*intro*]: R : *ISId* b
shows *isBot* (*rang* P)
proof −
have *isBot* ($P \odot R$) **by** (*rule cmp-isidbot*)
hence $P \sqsubseteq P \odot R$ **by** (*rule-tac isBot*, *best+*)
hence *rang* $P \sqsubseteq R$ **by** (*rule-tac llp* [*THEN iffD2*], *best+*)
thus *?thesis* **by** (*rule-tac isbot-incl-isbot*, *auto*)
qed

lemma (in *PreRanSemi*) *ran-strict*:
assumes [*intro*]: *isBot* ($R \odot$ *rang* R)

assumes [*intro*]: *isBot* (*R* ⊙ *P*)
assumes [*intro*]: *isBot P*
assumes [*intro*]: *R* : *a* ↔ *b*
assumes [*intro*]: *P* : *ISId b*
shows *isBot* (*rang R*) = *isBot R*
proof −
have *isBot* (*rang R*) ⟹ *isBot R* **by** (*rule-tac ran-strict1*, *best+*)
moreover have *isBot R* ⟹ *isBot* (*rang R*) **by** (*rule-tac R=P* **in** *ran-strict2*, *auto*)
ultimately show *?thesis* **by** (*rule iffI*, *best+*)
qed

The prerange operator is an identity on multiplicatively idempotent subidentities.

lemma (**in** *PreRanSemi*) *ran-isISId-eq*:
assumes [*intro*]: *P* : *ISId a*
shows *rang P* = *P*
proof −
have [*intro,simp*]: *P* : *a* ↔ *a* **by** (*rule ISId-homset*, *auto*)
have *rang P* ⊑ *P*
proof −
have *rang* (*P* ⊙ *P*) ⊑ *P* **by** (*rule-tac ran-ISId-cmp*, *best+*)
also have *P* ⊙ *P* = *P*
proof (*rule isISId-2*,*auto*)
show *isISId P* **by** (*rule-tac ISId*, *auto*)
qed
finally show *?thesis* .
qed
moreover have *P* ⊑ *rang P*
proof −
have [*intro*, *simp*]: *isISId P* **by** (*rule-tac ISId*, *auto*)
have *P* ⊑ *P* ⊙ *rang P* **by** (*rule ran-self*, *best*)
also have ... ⊑ *rang P* **by** (*best+*)
finally show *?thesis* **by** *best+*
qed
ultimately show *?thesis* **by** (*rule incl-antisym*, *best+*)
qed

The prerange operator is idempotent.

lemma (**in** *PreRanSemi*) *ran-idemp*:
assumes [*intro*]: *R* : *a* ↔ *b*

shows *rang* (*rang R*) = *rang R*
by (*rule ran-isISId-eq*, *best+*)

The prerange operator is a left invariant.

lemma (**in** *PreRanSemi*) *ran-right-inv*:
assumes [*intro*]: *R* : *a* ↔ *b*
shows *R* = *R* ⊙ *rang R*
proof −
have *R* ⊑ *R* ⊙ *rang R* **by** (*rule ran-self*, *auto*)
moreover have ... ⊑ *R* **by** *auto*
ultimately show *?thesis* **by** (*rule incl-antisym*, *best+*)
qed

The prerange operator satisfies a decomposition law.

lemma (**in** *PreRanSemi*) *ran-decomp*:
assumes [*intro*, *simp*]: *R* : *a* ↔ *b*
assumes [*intro*, *simp*]: *S* : *b* ↔ *c*
shows *rang* (*R* ⊙ *S*) ⊑ *rang* ((*rang R*) ⊙ *S*)
proof −
have *R* ⊙ *S* ⊑ (*R* ⊙ *S*) ⊙ *rang* ((*rang R*) ⊙ *S*)
proof −
have *R* ⊙ *S* ⊑ *R* ⊙ (((*rang R*) ⊙ *S*) ⊙ *rang* ((*rang R*) ⊙ *S*))
proof −
have [*simp*]: *R* ⊙ *S* ⊑ (*R* ⊙ *rang R*) ⊙ *S* **by** (*rule comp-incl-mon1*, *rule ran-self*, *best+*)
moreover have [*simp*]: ... = *R* ⊙ ((*rang R*) ⊙ *S*) **by** (*rule cmp-assoc*, *best+*)
moreover have [*simp*]: ... ⊑ *R* ⊙ (((*rang R*) ⨾ *S*) ⨾ *rang*((*rang R*) ⊙ *S*))
by (*rule-tac comp-incl-mon2*, *rule ran-self*, *best+*)
ultimately show *?thesis* **by** (*rule-tac incl-trans*, *auto*)
qed
hence *R* ⨾ *S* ⊑ (*R* ⊙ ((*rang R*) ⊙ *S*)) ⊙ *rang* ((*rang R*) ⊙ *S*) **by** (*subst cmp-assoc*, *best+*)
hence *R* ⨾ *S* ⊑ ((*R* ⊙ (*rang R*)) ⊙ *S*) ⊙ *rang* ((*rang R*) ⊙ *S*) **by** (*subst cmp-assoc*, *best+*)
also have *R* ⊙ *rang R* = *R* **by** (*rule ran-right-inv* [*THEN sym*], *auto*)
ultimately show *?thesis* **by** *auto*
qed
thus *?thesis* **by** (*rule-tac llp* [*THEN iffD2*], *best+*)
qed

B.13.3 postimage Operator

constdefs
postimage :: $('o, 'm, 'r)$ *SemiAllegory-scheme* $\Rightarrow$ $'m \Rightarrow 'm \Rightarrow 'm$ $(\xi_1$ - - *[1000,1000] 999)*
postimage s P R == if isMor s R & OS-isISId s P
then All-rang s (cmp s P R)
else arbitrary

lemma (in *PreRanSemi*) *postimage-def*:
$[\![R : a \leftrightarrow b; P : ISId\ a]\!] \Longrightarrow \xi\ P\ R = rang(P \odot R)$
by (*unfold postimage-def, auto*)

lemma (in *PreRanSemi*) *postimage-homset*[*intro,simp*]:
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $P : ISId\ a$
shows $\xi\ P\ R : b \leftrightarrow b$
proof −
have $\xi\ P\ R = rang\ (P \odot R)$ **by** (*rule postimage-def, best+*)
also have ... $: b \leftrightarrow b$ **by** (*rule ran-homset, best*)
finally show *?thesis* .
qed

lemma (in *PreRanSemi*) *postimage-ISId*:
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $P : ISId\ a$
shows $\xi\ P\ R : ISId\ b$
proof −
have $\xi\ P\ R = rang\ (P \odot R)$ **by** (*rule postimage-def, best+*)
also have ... $: ISId\ b$ **by** (*rule ran-ISId, best*)
finally show *?thesis* .
qed

Lemma postimage connects the postimage operator with llp.

lemma (in *PreRanSemi*) *postimage1*:
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $P : ISId\ a$
assumes [*intro*]: $Q : ISId\ b$
assumes *postimage-incl*: $\xi\ P\ R \sqsubseteq Q$
shows $P \odot R \sqsubseteq R \odot Q$

proof −
have [*intro,simp*]: *isISId P* **by** (*rule-tac ISId, auto*)
have $\xi\ P\ R = rang\ (P \odot R)$ **by** (*rule postimage-def, auto*)
with *postimage-incl* **have** [*intro*]: ... $\sqsubseteq Q$ **by** *simp*
have $P \odot R \sqsubseteq (P \odot R) \odot Q$ **by** (*rule-tac llp* [*THEN iffD1*], *best+*)
also have ... $\sqsubseteq R \odot Q$ **by** *auto*
finally show *?thesis* **by** *best+*
qed

lemma (in *PreRanSemi*) *postimage2*:
assumes [*intro, simp*]: $R : a \leftrightarrow b$
assumes [*intro, simp*]: $P : ISId\ a$
assumes [*intro, simp*]: $Q : ISId\ b$
assumes *incl*: $P \odot R \sqsubseteq R \odot Q$
shows $\xi\ P\ R \sqsubseteq Q$
proof −
have [*intro,simp*]: *isISId Q* **by** (*rule-tac ISId, auto*)
from *incl* **have** [*intro*]: $P \odot R = P \odot R \odot Q$ **by** (*rule isid-incl-eq1, best+*)
hence $P \odot R \sqsubseteq P \odot R \odot Q$ **by** *auto*
hence $P \odot R \sqsubseteq (P \odot R) \odot Q$ **by** (*subst cmp-assoc, auto*)
hence $rang\ (P \odot R) \sqsubseteq Q$ **by** (*rule-tac llp* [*THEN iffD2*], *best+*)
also have $rang(P \odot R) = \xi\ P\ R$ **by** (*subst postimage-def* [*THEN sym*], *auto*)
finally show *?thesis* .
qed

lemma (in *PreRanSemi*) *postimage*:
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $P : ISId\ a$
assumes [*intro*]: $Q : ISId\ b$
shows $(\xi\ P\ R \sqsubseteq Q) = (P \odot R \sqsubseteq R \odot Q)$
proof −
have $\xi\ P\ R \sqsubseteq Q \Longrightarrow P \odot R \sqsubseteq R \odot Q$ **by** (*rule postimage1, best+*)
moreover have $P \odot R \sqsubseteq R \odot Q \Longrightarrow \xi\ P\ R \sqsubseteq Q$ **by** (*rule postimage2, best+*)
ultimately show *?thesis* **by** (*rule iffI, best+*)
qed

end

B.14 Ordered Semigroupoid with Monotonic PreRange

theory *MonPreRanSemi*
imports *PreRanSemi*
begin

B.14.1 Definition

We introduce monotonicity of the domain operator.

locale *MonPreRanSemi* = *PreRanSemi MPRS* +
assumes *ran-incl-mon*[*intro*]: ⟦ $R : a \leftrightarrow b$; $S : a \leftrightarrow b$; $R \sqsubseteq S$ ⟧ ⟹ *rang* $R \sqsubseteq$ *rang* S

end

B.15 Ordered Semigroupoids with Range

theory *RanSemi*
imports *MonPreRanSemi*
begin

locale *RanSemi* = *MonPreRanSemi RS* +
assumes *ran-local*[*intro*, *simp*]:⟦$R : a \leftrightarrow b$; $S : b \leftrightarrow c$⟧ ⟹ *rang* (*rang* R $\odot$ S) $\sqsubseteq$ *rang* (R $\odot$ S)

lemma (**in** *RanSemi*) *ran-cmp*:
assumes [*intro*]: $R : a \leftrightarrow b$
assumes [*intro*]: $S : b \leftrightarrow c$
shows *rang* (*rang* $R \odot S$) = *rang* ($R \odot S$)
proof −
have *rang* (*rang* $R \odot S$) $\sqsubseteq$ *rang* ($R \odot S$) **by** *auto*
moreover have *rang* ($R \odot S$) $\sqsubseteq$ *rang* (*rang* $R \odot S$) **by** (*rule ran-decomp*, *auto*)
ultimately show *?thesis* **by** (*rule incl-antisym*, *auto*)
qed

end

B.16 Structure Record for Distributive Allegories

theory *DistrAllRecord*
imports *SemiAllRecord*
begin

record ('*o*, '*m*) *DistrAll* = ('*o*, '*m*) *SemiAllegory* +
join :: '*m* ⇒ '*m* ⇒ '*m* (**infixr** ⊔ı *60*)
bot :: '*o* ⇒ '*o* ⇒ '*m* (⊥ı - - [*1000*,*1000*] *999*)

end

B.17 Structure Record for Division Allegories: Residuals

theory *DivAllRecord*
imports *DistrAllRecord*
begin

We include both restricted residuals and standard residuals in the record in order to make it easier for theories without converse to use residuals.

These are the best fits in X-Symbol – they have the same direction as the real residual lines.

record ('*o*, '*m*) *DivAll* = ('*o*, '*m*) *DistrAll* +
rightRes :: '*m* ⇒ '*m* ⇒ '*m* (**infixr** →ı *200*)
leftRes :: '*m* ⇒ '*m* ⇒ '*m* (**infixr** ←ı *200*)
restrrightRes :: '*m* ⇒ '*m* ⇒ '*m* (**infixr** ⊣ı *200*)
restrleftRes :: '*m* ⇒ '*m* ⇒ '*m* (**infixr** ⊢ı *200*)
end

B.18 Standard Residuals in Ordered Semigroupoids

theory *OrdSemiRes*
imports *OrdSemi*
begin

B.18.1 Left Residuals

constdefs
OS-haveLeftRes :: ('o, 'm, 'r) *OrderedSemigroupoid-scheme* ⇒ 'm ⇒ 'm ⇒ 'm ⇒ *bool*
(*haveLeftRes*₁ - - - [*1000,1000,1000*] *999*)
OS-haveLeftRes s S R L == *if isMor s S* & *isMor s R* & (*Strg s S* = *Strg s R*)
& *isMor s L* & (*Ssrc s L* = *Ssrc s S*) & (*Strg s L* = *Ssrc s R*)
then (*ALL X .Semi-parallel s X L* ⟶
(*incl s* (*cmp s X R*) *S*) = *incl s X L*)
else arbitrary

lemma (**in** *OrderedSemigroupoid*) *haveLeftRes-def*:
⟦ *S* : *a* ↔ *b*; *R* : *c* ↔ *b*; *L* : *a* ↔ *c*⟧ ⟹
haveLeftRes S R L = (∀ *X* ∈ *a* ↔ *c* . (*X* ⊙ *R* ⊑ *S*) = (*X* ⊑ *L*))
apply (*unfold OS-haveLeftRes-def*, *simp*)
apply (*rule iffI*)
apply (*intro strip*)
apply (*drule-tac x=X* **in** *spec*)
apply (*drule-tac f=X* **and** *g=L* **in** *parallel-intro*, *simp*, *drule mp*, *assumption*)
apply (*simp-all*)
apply (*intro strip*)
apply (*drule-tac f=X* **and** *g=L* **in** *parallel-2*, *assumption*)
apply *simp*
done

lemma (**in** *OrderedSemigroupoid*) *haveLeftRes-intro*:
assumes *i*[*intro*, *simp*]: ∀ *X* ∈ *a* ↔ *c* .(*X* ⊙ *R* ⊑ *S*) = (*X* ⊑ *L*)
assumes [*intro*]: *S* : *a* ↔ *b*
assumes [*intro*]: *R* : *c* ↔ *b*
assumes [*intro*]: *L* : *a* ↔ *c*
shows *haveLeftRes S R L*
proof (*rule haveLeftRes-def* [*THEN sym*, *THEN iffD1*], *best+*)
from *i* **show** ∀*X*∈*a* ↔ *c*. (*X* ⊙ *R* ⊑ *S*) = (*X* ⊑ *L*) **by** *simp*
qed

lemma (**in** *OrderedSemigroupoid*) *haveLeftRes*:
assumes *res*: *haveLeftRes S R L*
assumes [*intro*]: *S* : *a* ↔ *b*
assumes [*intro*]: *R* : *c* ↔ *b*

assumes [*intro*]: *L* : *a* ↔ *c*
assumes [*intro*]: *X* : *a* ↔ *c*
shows (*X* ⊙ *R* ⊑ *S*) = (*X* ⊑ *L*)
proof −
from *res* **have** ∀ *X* ∈ *a* ↔ *c* .(*X* ⊙ *R* ⊑ *S*) = (*X* ⊑ *L*)
by (*rule-tac haveLeftRes-def* [*THEN iffD1*], *auto*)
thus *?thesis* by *auto*
qed

lemma (**in** *OrderedSemigroupoid*) *haveLeftRes-res-intro*:
assumes [*simp*]: *X* ⊙ *R* ⊑ *S*
assumes [*intro*]: *haveLeftRes S R L*
assumes [*intro*]: *S* :*a* ↔ *b*
assumes [*intro*]: *R* : *c* ↔ *b*
assumes [*intro*]: *L* : *a* ↔ *c*
assumes [*intro*]: *X* : *a* ↔ *c*
shows *X* ⊑ *L*
by (*rule-tac haveLeftRes* [*THEN iffD1*], *auto*)

lemma (**in** *OrderedSemigroupoid*) *haveLeftRes-res-elim*:
assumes [*intro*]: *X* ⊑ *L*
assumes [*intro*]: *haveLeftRes S R L*
assumes [*intro*]: *S* : *a* ↔ *b*
assumes [*intro*]: *R* : *c* ↔ *b*
assumes [*intro*]: *L* : *a* ↔ *c*
assumes [*intro*]: *X* : *a* ↔ *c*
shows *X* ⊙ *R* ⊑ *S*
by (*rule-tac haveLeftRes* [*THEN iffD2*], *auto*)

B.18.2 Right Residuals

constdefs
OS-haveRightRes :: ('o, 'm, 'r) *OrderedSemigroupoid-scheme* ⇒ 'm ⇒ 'm ⇒ 'm ⇒ *bool*
(*haveRightRes*₁ - - - [*1000,1000,1000*] *999*)
OS-haveRightRes s S L R == *if isMor s S* & *isMor s L* & (*Ssrc s S* = *Ssrc s L*)
& *isMor s R* & (*Ssrc s R* = *Strg s L*) & (*Strg s R* = *Strg s S*)
then (*ALL X . Semi-parallel s X R* ⟶
(*incl s* (*cmp s L X*) *S*) = *incl s X R*)
else arbitrary

lemma (in *OrderedSemigroupoid*) *haveRightRes-def*:
⟦ $S : a \leftrightarrow b$; $L : a \leftrightarrow c$; $R : c \leftrightarrow b$ ⟧ ⟹
haveRightRes S L R = (∀ $X \in c \leftrightarrow b$. $(L \odot X \sqsubseteq S) = (X \sqsubseteq R)$)
apply (*unfold OS-haveRightRes-def*, *simp*)
apply (*rule iffI*)
apply (*intro strip*)
apply (*drule-tac x=X* **in** *spec*)
apply (*drule-tac f=X* **and** *g=R* **in** *parallel-intro*, *simp*, *drule mp*, *assumption*)
apply (*simp-all*)
apply (*intro strip*)
apply (*drule-tac f=X* **and** *g=R* **in** *parallel-2*, *assumption*)
apply *simp*
done

lemma (in *OrderedSemigroupoid*) *haveRightRes-intro*:
assumes *i* [*intro*, *simp*]: ∀ $X \in c \leftrightarrow b$. $(L \odot X \sqsubseteq S) = (X \sqsubseteq R)$
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $R : c \leftrightarrow b$
shows *haveRightRes S L R*
proof (*rule haveRightRes-def* [*THEN sym*, *THEN iffD1*], *best+*)
from *i* **show** $\forall X{\in}c \leftrightarrow b.\ (L \odot X \sqsubseteq S) = (X \sqsubseteq R)$ **by** *simp*
qed

lemma (in *OrderedSemigroupoid*) *haveRightRes*:
assumes *res*: *haveRightRes S L R*
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $X : c \leftrightarrow b$
shows $(L \odot X \sqsubseteq S) = (X \sqsubseteq R)$
proof −
from *res* **have** ∀ $X \in c \leftrightarrow b$. $(L \odot X \sqsubseteq S) = (X \sqsubseteq R)$
by (*rule-tac haveRightRes-def* [*THEN iffD1*], *auto*)
thus *?thesis* **by** *auto*
qed

lemma (in *OrderedSemigroupoid*) *haveRightRes-res-intro*:
assumes [*intro*]: $L \odot X \sqsubseteq S$
assumes [*intro*]: *haveRightRes S L R*

assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $X : c \leftrightarrow b$
shows $X \sqsubseteq R$
by (*rule-tac haveRightRes* [*THEN iffD1*], *auto*)

lemma (in *OrderedSemigroupoid*) *haveRightRes-res-elim*:
assumes [*simp*]: $X \sqsubseteq R$
assumes [*intro*]: *haveRightRes S L R*
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $X : c \leftrightarrow b$
shows $L \odot X \sqsubseteq S$
by (*rule-tac haveRightRes* [*THEN iffD2*], *auto*)

end

B.19 Semigroupoids with standard Left Residuals

theory *LResSemi*
imports *OrdSemiRes DivAllRecord*
begin

B.19.1 Definitions

locale *LResSemi* = *OrderedSemigroupoid LRS* +
assumes *leftRes-homset*[*intro*,*simp*]: ⟦ $R : c \leftrightarrow b$; $S : a \leftrightarrow b$ ⟧ ⟹ $(R \leftharpoonup S) : a \leftrightarrow c$
assumes *leftRes*[*intro*,*simp*]: ⟦ $R : c \leftrightarrow b$; $S : a \leftrightarrow b$ ⟧ ⟹ *haveLeftRes S R* $(R \leftharpoonup S)$

B.19.2 Auxiliary Lemmas

lemma (in *LResSemi*) *leftRes-src*[*simp*]:
assumes [*intro*]: $R : c \leftrightarrow b$

assumes [*intro*]: $S : a \leftrightarrow b$
shows *src* $(R \leftarrow S) = a$
proof −
have $(R \leftarrow S) : a \leftrightarrow c$ **by** (*rule leftRes-homset*, *auto*)
thus *?thesis* **by** (*rule homset-src*)
qed

lemma (**in** *LResSemi*) *leftRes-trg*[*simp*]:
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $S : a \leftrightarrow b$
shows *trg* $(R \leftarrow S) = c$
proof −
have $(R \leftarrow S) : a \leftrightarrow c$ **by** (*rule leftRes-homset*, *auto*)
thus *?thesis* **by** (*rule homset-trg*)
qed

lemma (**in** *LResSemi*) *leftRes-defined*[*simp*]:
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $S : a \leftrightarrow b$
shows *Mor* $(R \leftarrow S)$
proof (*rule homset-Mor*)
show $(R \leftarrow S) : a \leftrightarrow c$ **by** (*rule leftRes-homset*, *auto*)
qed

lemma (**in** *LResSemi*) *lres-src*[*simp*]:
assumes [*simp*]: *trg* R = *trg* S
assumes [*simp*]: *Mor* R
assumes [*simp*]: *Mor* S
shows *src* $(R \leftarrow S)$ = *src* S
proof −
have $R \leftarrow S$: *src* $S \leftrightarrow$ *src* R **by** (*rule leftRes-homset*, *rule homset1*, *auto*)
thus *?thesis* **by** (*rule homset-src*)
qed

lemma (**in** *LResSemi*) *lres-trg*[*simp*]:
assumes [*simp*]: *trg* R = *trg* S
assumes [*simp*]: *Mor* R
assumes [*simp*]: *Mor* S
shows *trg* $(R \leftarrow S)$ = *src* R
proof −

have $R \leftarrow S$: *src* $S \leftrightarrow$ *src* R **by** (*rule leftRes-homset*, *rule homset1*, *auto*)
thus *?thesis* **by** (*rule homset-trg*)
qed

lemma (**in** *LResSemi*) *lres*[*simp*]:
assumes [*intro*]: $R : c \leftrightarrow b$
~~**assumes** [*intro*]: $S : a \leftrightarrow b$~~
assumes [*intro*]: $X : a \leftrightarrow c$
shows $((X \odot R) \sqsubseteq S) = (X \sqsubseteq (R \leftarrow S))$
proof −
have *haveLeftRes S R* $(R \leftarrow S)$ **by** (*rule leftRes*, *best*+)
thus *?thesis* **by** (*rule haveLeftRes*, *best*+)
qed

lemmas (**in** *LResSemi*) *lres1* = *lres* [*THEN iffD1*]
lemmas (**in** *LResSemi*) *lres2* = *lres* [*THEN iffD2*]

B.19.3 Equivalent axiomatization

lemma (**in** *LResSemi*) *incl-lres*[*intro*]:
assumes [*intro*]: $T : a \leftrightarrow c$
assumes [*intro*]: $R : c \leftrightarrow b$
shows $T \sqsubseteq (R \leftarrow (T \odot R))$
proof −
have $(\forall\ X\ .\ X : a \leftrightarrow c \longrightarrow X \sqsubseteq T \longrightarrow X \sqsubseteq R \leftarrow (T \odot R))$
proof (*intro strip*)
fix X
assume [*intro*]:$X \in a \leftrightarrow c$
assume XT[*simp*]: $X \sqsubseteq T$
hence [*intro*]: $(X \odot R \sqsubseteq T \cdot R)$ **by** (*rule-tac comp-incl-mon1*, *best*+)
thus $X \sqsubseteq (R \leftarrow (T \odot R))$ **by** (*rule-tac lres1*, *best*+)
qed
thus *?thesis* **by** (*rule-tac indir-ineq2*, *auto*)
qed

lemma (**in** *LResSemi*) *lrescmp-incl*:
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $S : a \leftrightarrow b$
shows $((R \leftarrow S) \odot R) \sqsubseteq S$
proof −

let *?X* $= R \leftarrow S$
have [*intro*]: *?X* : $a \leftrightarrow c$ **by** *auto*
moreover have [*intro*]: *?X* $\sqsubseteq (R \leftarrow S)$ **by** (*rule incl-refl*, *best*)
have (*?X* $\odot R \sqsubseteq S$) **by** (*rule lres* [*THEN iffD2*], *auto*)
thus *?thesis* **by** (*best+*)
qed

——Residual properties in Furusawa-Kahl-1998——

Proposition 4.4 (i)

lemma (**in** *LResSemi*) *lrescom-incl-fs*:
assumes [*intro*]:$S : a \leftrightarrow c$
assumes [*intro*]:$R : b \leftrightarrow c$
assumes [*intro*]:$T : d \leftrightarrow c$
shows $(R \leftarrow S) \cdot (T \leftarrow R) \sqsubseteq (T \leftarrow S)$
proof −
have [*intro*] : $(T \leftarrow R) \odot T \sqsubseteq R$ **by** (*rule lrescmp-incl*, *auto*)
have [*intro*] : $(R \leftarrow S) \odot R \sqsubseteq S$ **by** (*rule lrescmp-incl*, *auto*)
have $((R \leftarrow S) \odot (T \leftarrow R)) \odot T = (R \leftarrow S) \odot ((T \leftarrow R) \odot T)$ **by** (*rule cmp-assoc*, *auto*)
moreover have ... $\sqsubseteq (R \leftarrow S) \odot R$ **by** *auto*
moreover have ... $\sqsubseteq S$ **by** *auto*
ultimately have *l* : $((R \leftarrow S) \odot (T \leftarrow R)) \odot T \sqsubseteq S$ **by** (*rule-tac incl-trans*, *auto*)
from *l* **show** $((R \leftarrow S) \odot (T \leftarrow R)) \sqsubseteq (T \leftarrow S)$**by** (*rule-tac lres* [*THEN iffD1*], *auto*)
qed

Proposition 4.4 (iii)

lemma (**in** *LResSemi*) *lrescom-incl-ohk*:
assumes [*intro*]: $S : a \leftrightarrow c$
assumes [*intro*]: S': $a \leftrightarrow c$
assumes [*intro*]: $R : b \leftrightarrow c$
assumes [*intro*]: R': $b \leftrightarrow c$
assumes S[*intro*]: $S \sqsubseteq S'$
assumes R[*intro*]: $R' \sqsubseteq R$
shows $(R \leftarrow S) \sqsubseteq (R' \leftarrow S')$
proof (*rule-tac indir-ineq2*, *best+*)
show $\forall C.\ C \in a \leftrightarrow b \longrightarrow C \sqsubseteq (R \leftarrow S) \longrightarrow C \sqsubseteq (R' \leftarrow S')$
proof (*intro strip*)
fix *C*

assume [*intro*]: $C : a \leftrightarrow b$
assume C[*intro*]: $C \sqsubseteq (R \leftarrow S)$
show $C \sqsubseteq (R' \leftarrow S')$
proof −
have $C \odot R' \sqsubseteq C \cdot R$ **by** *auto*
moreover have ... $\sqsubseteq S$ **by** (*rule-tac lres2*, *auto*)
ultimately have $C \cdot R' \sqsubseteq S$ **by** (*rule incl-trans*, *auto*)
moreover have ... $\sqsubseteq S'$ **by** *auto*
ultimately have *l*: $C \odot R' \sqsubseteq S'$ **by** (*rule incl-trans*, *auto*)
from *l* **show** *?thesis* **by**(*rule-tac lres1*, *auto*)
qed
qed
qed

Proposition 4.5 (i)

lemma (**in** *LResSemi*) *lrescom-incl-ex*:
assumes [*intro*]:$F : a \leftrightarrow b$
assumes [*intro*]:$R : b \leftrightarrow c$
assumes [*intro*]:$S : d \leftrightarrow c$
shows $(F \odot (S \leftarrow R)) \sqsubseteq (S \leftarrow (F \odot R))$
proof −
have[*intro*]: $F \sqsubseteq (R \leftarrow (F \cdot R))$ **by** *auto*
have $(F \odot (S \leftarrow R)) \sqsubseteq ((R \leftarrow (F \cdot R)) \odot (S \leftarrow R))$ **by** *auto*
moreover have ... $\sqsubseteq (S \leftarrow (F \odot R))$ **by**(*rule lrescom-incl-fs*, *auto*)
ultimately show *?thesis* **by** (*rule incl-trans*, *auto*)
qed

end

B.20 Semigroupoids with standard Right Residuals

theory *RResSemi*
imports *OrdSemiRes DivAllRecord*
begin

B.20.1 Definitions

locale *RResSemi* = *OrderedSemigroupoid RRS* +

assumes *rightRes-homset*[*intro*,*simp*]: $[\![L : a \leftrightarrow c;\ S : a \leftrightarrow b]\!] \Longrightarrow (S \rightharpoonup L) : c \leftrightarrow b$
assumes *rightRes*[*intro*,*simp*]: $[\![L : a \leftrightarrow c;\ S : a \leftrightarrow b]\!] \Longrightarrow$ *haveRightRes* $S\ L\ (S \rightharpoonup L)$

B.20.2 Auxiliary Lemmas

lemma (in *RResSemi*) *rightRes-src*[*simp*]:
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $S : a \leftrightarrow b$
shows *src* $(S \rightharpoonup L) = c$
proof −
have $(S \rightharpoonup L) : c \leftrightarrow b$ **by** (*rule rightRes-homset*, *auto*)
thus *?thesis* **by** (*rule homset-src*)
qed

lemma (in *RResSemi*) *rightRes-trg*[*simp*]:
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $S : a \leftrightarrow b$
shows *trg* $(S \rightharpoonup L) = b$
proof −
have $(S \rightharpoonup L) : c \leftrightarrow b$ **by** (*rule rightRes-homset*, *auto*)
thus *?thesis* **by** (*rule homset-trg*)
qed

lemma (in *RResSemi*) *rightRes-defined*[*simp*]:
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $S : a \leftrightarrow b$
shows *Mor* $(S \rightharpoonup L)$
proof (*rule homset-Mor*)
show $S \rightharpoonup L : c \leftrightarrow b$ **by** (*rule rightRes-homset*, *auto*)
qed

lemma (in *RResSemi*) *rres-src*:
assumes [*simp*]: *src* L = *src* S
assumes [*simp*]: *Mor* L
assumes [*simp*]: *Mor* S
shows *src* $(S \rightharpoonup L)$ = *trg* L
proof −
have $S \rightharpoonup L$: *trg* $L \leftrightarrow$ *trg* S **by** (*rule rightRes-homset*, *rule homset1*, *auto*)
thus *?thesis* **by** (*rule homset-src*)
qed

lemma (in *RResSemi*) *rres-trg*:
assumes [*simp*]: *src* L = *src* S
assumes [*simp*]: *Mor* L
assumes [*simp*]: *Mor* S
shows *trg* $(S \rightharpoonup L)$ = *trg* S
proof −
have $S \rightharpoonup L$: *trg* $L \leftrightarrow$ *trg* S **by** (*rule rightRes-homset*, *rule homset1*, *auto*)
thus *?thesis* **by** (*rule homset-trg*)
qed

lemma (in *RResSemi*) *rres*:
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $X : c \leftrightarrow b$
shows $(L \odot X \sqsubseteq S) = (X \sqsubseteq (S \rightharpoonup L))$
proof −
have *haveRightRes* $S\ L\ (S \rightharpoonup L)$ **by** (*rule rightRes*, *best*+)
thus *?thesis* **by** (*rule haveRightRes*, *best*+)
qed

lemmas (in *RResSemi*) *rres1* = *rres* [*THEN iffD1*]
lemmas (in *RResSemi*) *rres2* = *rres* [*THEN iffD2*]

B.20.3 Equivalent axiomatization

lemma (in *RResSemi*) *incl-rres*[*intro*]:
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $T : c \leftrightarrow b$
shows $T \sqsubseteq ((L \odot T) \rightharpoonup L)$
proof −
have $(\forall\ X\ .\ X : c \leftrightarrow b \longrightarrow X \sqsubseteq T \longrightarrow X \sqsubseteq (L \odot T) \rightharpoonup L)$
proof (*intro strip*)
fix X
assume [*intro*]: $X : c \leftrightarrow b$
assume [*simp*]: $X \sqsubseteq T$
hence [*intro*]: $L \odot X \sqsubseteq L \odot T$ **by** (*rule-tac comp-incl-mon2*, *best*+)
thus $X \sqsubseteq (L \odot T) \rightharpoonup L$ **by** (*rule-tac rres1*, *best*+)
qed
thus *?thesis* **by** (*rule-tac indir-ineq2*, *auto*)

qed

lemma (in *RResSemi*) *rrescmp-incl*:
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $L : a \leftrightarrow c$
shows $L \odot (S \rightharpoonup L) \sqsubseteq S$
proof −
let $?X = S \rightharpoonup L$
have [*intro*]: $?X : c \leftrightarrow b$ **by** *auto*
moreover have [*intro*]: $?X \sqsubseteq S \rightharpoonup L$ **by** (*rule incl-refl*, *best*)
moreover have $(L \odot ?X \sqsubseteq S)$ **by** (*rule rres* [*THEN iffD2*], *auto*)
thus *?thesis* **by** (*best+*)
qed

——Residual properties in Furusawa-Kahl-1998——

Proposition 4.4 (i)

lemma (in *RResSemi*) *rrescom-incl-fs*:
assumes [*intro*]: $S : a \leftrightarrow c$
assumes [*intro*]: $L : a \leftrightarrow b$
assumes [*intro*]: $U : a \leftrightarrow d$
shows $(S \rightharpoonup L) \odot (U \rightharpoonup S) \sqsubseteq (U \rightharpoonup L)$
proof −
have [*intro*] : $L \odot (S \rightharpoonup L) \sqsubseteq S$ **by** (*rule rrescmp-incl*, *auto*)
have [*intro*] : $S \odot (U \rightharpoonup S) \sqsubseteq U$ **by** (*rule rrescmp-incl*, *auto*)
have $L \odot ((S \rightharpoonup L) \odot (U \rightharpoonup S)) = (L \odot (S \rightharpoonup L)) \odot (U \rightharpoonup S)$ **by** (*rule cmp-assoc-sym*, *auto*)
moreover have ... $\sqsubseteq S \odot (U \rightharpoonup S)$ **by** *auto*
moreover have ... $\sqsubseteq U$ **by** *auto*
ultimately have $r : L \odot ((S \rightharpoonup L) \odot (U \rightharpoonup S)) \sqsubseteq U$ **by** (*rule-tac incl-trans*, *auto*)
from *r* **show** *?thesis* **by** (*rule-tac rres* [*THEN iffD1*], *auto*)
qed

Proposition 4.4 (iii)

lemma (in *RResSemi*) *rrescom-incl-ohk*:
assumes [*intro*]: $S : a \leftrightarrow c$
assumes [*intro*]: $S' : a \leftrightarrow c$
assumes [*intro*]: $Q : a \leftrightarrow b$
assumes [*intro*]: $Q' : a \leftrightarrow b$

assumes S[*intro*]: $S \sqsubseteq S'$
assumes Q[*intro*]: $Q' \sqsubseteq Q$
shows $(S \rightharpoonup Q) \sqsubseteq (S' \rightharpoonup Q')$
proof (*rule-tac indir-ineq2*, *best+*)
show $\forall C.\ C \in b \leftrightarrow c \longrightarrow C \sqsubseteq (S \rightharpoonup Q) \longrightarrow C \sqsubseteq (S' \rightharpoonup Q')$
proof (*intro strip*)
fix *C*
assume [*intro*]: $C : b \leftrightarrow c$
assume C[*intro*]: $C \sqsubseteq (S \rightharpoonup Q)$
show $C \sqsubseteq (S' \rightharpoonup Q')$
proof −
have $Q' \odot C \sqsubseteq Q \odot C$ **by** *auto*
moreover have ... $\sqsubseteq S$ **by** (*rule-tac rres2*, *auto*)
ultimately have $Q' \odot C \sqsubseteq S$ **by** (*rule incl-trans*, *auto*)
moreover have ... $\sqsubseteq S'$ **by** *auto*
ultimately have *l*: $Q' \odot C \sqsubseteq S'$ **by** (*rule incl-trans*, *auto*)
from *l* **show** *?thesis* **by**(*rule-tac rres1*, *auto*)
qed
qed
qed

Proposition 4.5 (i)

lemma (in *RResSemi*) *rrescom-incl-ex*:
assumes [*intro*]: $U : a \leftrightarrow b$
assumes [*intro*]: $Q : a \leftrightarrow c$
assumes [*intro*]: $T : c \leftrightarrow d$
shows $((Q \rightharpoonup U) \mathbin{:} T) \sqsubseteq ((Q \odot T) \rightharpoonup U)$
proof −
have[*intro*]: $T \sqsubseteq ((Q \odot T) \rightharpoonup Q)$ **by** *auto*
have $((Q \rightharpoonup U) \odot T) \sqsubseteq ((Q \rightharpoonup U) \odot ((Q \mathbin{:} T) \rightharpoonup Q))$ **by** *auto*
moreover have ... $\sqsubseteq ((Q \odot T) \rightharpoonup U)$ **by**(*rule rrescom-incl-fs*, *auto*)
ultimately show *?thesis* **by** (*rule incl-trans*, *auto*)
qed

end

B.21 Semigroupoids with Standard Residuals

theory *ResSemi*

imports *LResSemi RResSemi*
begin

B.21.1 Definitions

locale *ResSemi = LResSemi RS + RResSemi RS*

end

B.22 Restricted Residuals in Ordered Semigroupoids

theory *OrdSemiRestrRes*
imports *RanSemi DomSemi*
begin

B.22.1 Basic Definitions

locale *OrdSemiRestrRes = RanSemi OSR + DomSemi OSR*

B.22.2 Restricted Left Residuals

constdefs
OS-haveRestrLeftRes :: $('o, 'm, 'r)$ *SemiAllegory-scheme* $\Rightarrow$ $'m \Rightarrow 'm \Rightarrow 'm \Rightarrow$ *bool*
(*haveRestrLeftRes* - - - [*1000,1000,1000*] *999*)
OS-haveRestrLeftRes s S R L == *if isMor s S & isMor s R & (Strg s S = Strg s R)*
& isMor s L & (Ssrc s L = Ssrc s S) & (Strg s L = Ssrc s R)
then (ALL X .Semi-parallel s X L $\longrightarrow$
(incl s (cmp s X R) S & incl s (All-rang s X) (All-dom s R)) = incl s X L)
else arbitrary

lemma (**in** *OrdSemiRestrRes*) *haveRestrLeftRes-def*:
$[\![S : a \leftrightarrow b; R : c \leftrightarrow b; L : a \leftrightarrow c]\!] \Longrightarrow$
haveRestrLeftRes S R L = $(\forall\ X \in a \leftrightarrow c\ .\ ((X \odot R \sqsubseteq S)$ & $(rang\ X \sqsubseteq dom\ R)) = (X \sqsubseteq L))$
apply (*unfold OS-haveRestrLeftRes-def, simp*)
apply (*rule iffI*)

apply (*intro strip*)
apply (*drule-tac x=X* **in** *spec*)
apply (*drule-tac f=X* **and** *g=L* **in** *parallel-intro, simp, drule mp, assumption*)
apply (*simp-all*)
apply (*intro strip*)
apply (*drule-tac f=X* **and** *g=L* **in** *parallel-2, assumption*)
apply *simp*
done

lemma (**in** *OrdSemiRestrRes*) *haveRestrLeftRes-intro*:
assumes *i*[*intro, simp*]: $\forall\ X \in a \leftrightarrow c\ .((X \odot R \sqsubseteq S)$ & $(rang\ X \sqsubseteq dom\ R)) = (X \sqsubseteq L)$
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $L : a \leftrightarrow c$
shows *haveRestrLeftRes S R L*
proof (*rule haveRestrLeftRes-def* [*THEN sym, THEN iffD1*], *best+*)
from *i* **show** $\forall X \in a \leftrightarrow c.\ (X \odot R \sqsubseteq S \wedge rang\ X \sqsubseteq dom\ R) = (X \sqsubseteq L)$ **by** *simp*
qed

lemma (**in** *OrdSemiRestrRes*) *haveRestrLeftRes*:
assumes *res*: *haveRestrLeftRes S R L*
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $X : a \leftrightarrow c$
shows $((X \odot R \sqsubseteq S)$ & $(rang\ X \sqsubseteq dom\ R)) = (X \sqsubseteq L)$
proof −
from *res* **have** $\forall\ X \in a \leftrightarrow c\ .((X \odot R \sqsubseteq S)$ & $(rang\ X \sqsubseteq dom\ R)) = (X \sqsubseteq L)$
by (*rule-tac haveRestrLeftRes-def* [*THEN iffD1*], *auto*)
thus *?thesis* **by** *auto*
qed

lemma (**in** *OrdSemiRestrRes*) *haveRestrLeftRes-res-intro*:
assumes [*simp*]: $X \odot R \sqsubseteq S$
assumes [*simp*]: $rang\ X \sqsubseteq dom\ R$
assumes [*intro*]: *haveRestrLeftRes S R L*
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $L : a \leftrightarrow c$

assumes [*intro*]: $X : a \leftrightarrow c$
shows $X \sqsubseteq L$
by (*rule-tac haveRestrLeftRes* [*THEN iffD1*], *auto*)

lemma (in *OrdSemiRestrRes*) *haveRestrLeftRes-res-elim*:
assumes [*intro*]: $X \sqsubseteq L$
assumes [*intro*]: *haveRestrLeftRes S R L*
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $X : a \leftrightarrow c$
shows $X \odot R \sqsubseteq S$ & *rang* $X \sqsubseteq$ *dom* R
by (*rule-tac haveRestrLeftRes* [*THEN iffD2*], *auto*)

B.22.3 Restricted Right Residuals

constdefs
OS-haveRestrRightRes :: ('*o*, '*m*, '*r*) *SemiAllegory-scheme* ⇒ '*m* ⇒ '*m* ⇒ '*m* ⇒ *bool*
(*haveRestrRightRes*₁ - - - [*1000,1000,1000*] *999*)
OS-haveRestrRightRes s S L R == *if isMor s S* & *isMor s L* & (*Ssrc s S* = *Ssrc s L*)
& *isMor s R* & (*Ssrc s R* = *Strg s L*) & (*Strg s R* = *Strg s S*)
then (*ALL X . Semi-parallel s X R* ⟶
(*incl s* (*cmp s L X*) *S* & *incl s* (*All-dom s X*) (*All-rang s L*)) = *incl s X R*)
else arbitrary

lemma (in *OrdSemiRestrRes*) *haveRestrRightRes-def*:
⟦ $S : a \leftrightarrow b$; $L : a \leftrightarrow c$; $R : c \leftrightarrow b$ ⟧ ⟹
haveRestrRightRes S L R = (∀ $X \in c \leftrightarrow b$. (($L \odot X \sqsubseteq S$) & (*dom* $X \sqsubseteq$ *rang* L)) = ($X \sqsubseteq R$))
apply (*unfold OS-haveRestrRightRes-def*, *simp*)
apply (*rule iffI*)
apply (*intro strip*)
apply (*drule-tac x*=*X* **in** *spec*)
apply (*drule-tac f*=*X* **and** *g*=*R* **in** *parallel-intro*, *simp*, *drule mp*, *assumption*)
apply (*simp-all*)
apply (*intro strip*)
apply (*drule-tac f*=*X* **and** *g*=*R* **in** *parallel-2*, *assumption*)
apply *simp*
done

lemma (in *OrdSemiRestrRes*) *haveRestrRightRes-intro*:
assumes *i* [*intro*, *simp*]: ∀ $X \in c \leftrightarrow b$. (($L \odot X \sqsubseteq S$) & (*dom* $X \sqsubseteq$ *rang* L)) = ($X \sqsubseteq R$)
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $R : c \leftrightarrow b$
shows *haveRestrRightRes S L R*
proof (*rule haveRestrRightRes-def* [*THEN sym*, *THEN iffD1*], *best*+)
from *i* **show** ∀ $X \in c \leftrightarrow b$. ($L \odot X \sqsubseteq S \wedge$ *dom* $X \sqsubseteq$ *rang* L) = ($X \sqsubseteq R$) **by** *simp*
qed

lemma (in *OrdSemiRestrRes*) *haveRestrRightRes*:
assumes *res*: *haveRestrRightRes S L R*
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $X : c \leftrightarrow b$
shows (($L \odot X \sqsubseteq S$) & (*dom* $X \sqsubseteq$ *rang* L)) = ($X \sqsubseteq R$)
proof −
from *res* **have** ∀ $X \in c \leftrightarrow b$. (($L \odot X \sqsubseteq S$) & (*dom* $X \sqsubseteq$ *rang* L)) = ($X \sqsubseteq R$)
by (*rule-tac haveRestrRightRes-def* [*THEN iffD1*], *auto*)
thus *?thesis* **by** *auto*
qed

lemma (in *OrdSemiRestrRes*) *haveRestrRightRes-res-intro*:
assumes [*intro*]: $L \odot X \sqsubseteq S$
assumes [*intro*]: *dom* $X \sqsubseteq$ *rang* L
assumes [*intro*]: *haveRestrRightRes S L R*
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $X : c \leftrightarrow b$
shows $X \sqsubseteq R$
by (*rule-tac haveRestrRightRes* [*THEN iffD1*], *auto*)

lemma (in *OrdSemiRestrRes*) *haveRestrRightRes-res-elim*:
assumes [*simp*]: $X \sqsubseteq R$
assumes [*intro*]: *haveRestrRightRes S L R*
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $L : a \leftrightarrow c$

assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $X : c \leftrightarrow b$
shows $(L \odot X \sqsubseteq S)$ & $(dom\ X \sqsubseteq rang\ L)$
by (*rule-tac haveRestrRightRes* [*THEN iffD2*], *auto*)

end

B.23 Semigroupoids with Restricted Left Residuals

theory *RestrLResSemi*
imports *OrdSemiRestrRes DivAllRecord*
begin

B.23.1 Definitions

locale *RestrLResSemi* = *OrdSemiRestrRes RLRS* +
assumes *restrleftRes-homset*[*intro,simp*]: $[\![R : c \leftrightarrow b; S : a \leftrightarrow b]\!] \Longrightarrow (R \vdash S) : a \leftrightarrow c$
assumes *restrleftRes*[*intro,simp*]: $[\![R : c \leftrightarrow b; S : a \leftrightarrow b]\!] \Longrightarrow haveLeftRes\ S\ R\ (R \vdash S)$

B.23.2 Auxiliary Lemmas

lemma (**in** *RestrLResSemi*) *restr-leftRes-src*[*simp*]:
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $S : a \leftrightarrow b$
shows $src\ (R \vdash S) = a$
proof −
have $(R \vdash S) : a \leftrightarrow c$ **by** (*rule restrleftRes-homset*, *auto*)
thus *?thesis* **by** (*rule homset-src*)
qed

lemma (**in** *RestrLResSemi*) *restr-leftRes-trg*[*simp*]:
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $S : a \leftrightarrow b$
shows $trg\ (R \vdash S) = c$
proof −

have $(R \vdash S) : a \leftrightarrow c$ **by** (*rule restrleftRes-homset*, *auto*)
thus *?thesis* **by** (*rule homset-trg*)
qed

lemma (**in** *RestrLResSemi*) *restrleftRes-defined*[*simp*]:
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $S : a \leftrightarrow b$
shows $Mor\ (R \vdash S)$
proof (*rule homset-Mor*)
show $(R \vdash S) : a \leftrightarrow c$ **by** (*rule restrleftRes-homset*, *auto*)
qed

lemma (**in** *RestrLResSemi*) *restr-lres-src*[*simp*]:
assumes [*simp*]: $trg\ R = trg\ S$
assumes [*simp*]: $Mor\ R$
assumes [*simp*]: $Mor\ S$
shows $src\ (R \vdash S) = src\ S$
proof −
have $R \vdash S : src\ S \leftrightarrow src\ R$ **by** (*rule restrleftRes-homset*, *rule homset1*, *auto*)
thus *?thesis* **by** (*rule homset-src*)
qed

lemma (**in** *RestrLResSemi*) *restr-lres-trg*[*simp*]:
assumes [*simp*]: $trg\ R = trg\ S$
assumes [*simp*]: $Mor\ R$
assumes [*simp*]: $Mor\ S$
shows $trg\ (R \vdash S) = src\ R$
proof −
have $R \vdash S : src\ S \leftrightarrow src\ R$ **by** (*rule restrleftRes-homset*, *rule homset1*, *auto*)
thus *?thesis* **by** (*rule homset-trg*)
qed

lemma (**in** *RestrLResSemi*) *restr-lres*:
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $X : a \leftrightarrow c$
shows $((X \odot R \sqsubseteq S) \wedge (rang\ X \sqsubseteq dom\ R)) = (X \sqsubseteq (R \vdash S))$
proof −
have *haveLeftRes S R* $(R \vdash S)$ **by** (*rule restrleftRes*, *best+*)
thus *?thesis* **by** (*rule-tac haveRestrLeftRes*, *best+*)

qed

lemmas (in *RestrLResSemi*) *restr-lres1* = *restr-lres* [*THEN iffD1*]
lemmas (in *RestrLResSemi*) *restr-lres2* = *restr-lres* [*THEN iffD2*]

Add the following two auxiliary lemmas.

lemma (in *RestrLResSemi*) *restr-lres-incl1*[*intro*]:
assumes [*intro*]: $X : a \leftrightarrow c$
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: S: $a \leftrightarrow b$
assumes *TR*[*intro*]: $X \sqsubseteq R \vdash S$
shows *rang* $X \sqsubseteq$ *dom* R
proof −
from *TR* **have** *ss*: $(X \odot R \sqsubseteq S) \wedge (\mathit{rang}\ X \sqsubseteq \mathit{dom}\ R)$
by (*rule-tac restr-lres* [*THEN iffD2*], *auto*)
from *ss* **show** *?thesis* **by** *auto*
qed

lemma (in *RestrLResSemi*) *restr-lres-incl2*[*intro*]:
assumes [*intro*]: $X : a \leftrightarrow c$
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: S: $a \leftrightarrow b$
assumes *XR*[*intro*]: $X \sqsubseteq R \vdash S$
shows $(X \odot R \sqsubseteq S)$
proof −
from *XR* **have** *ss*: $(X \odot R \sqsubseteq S) \wedge (\mathit{rang}\ X \sqsubseteq \mathit{dom}\ R)$
by (*rule-tac restr-lres* [*THEN iffD2*], *auto*)
from *ss* **show** *?thesis* **by** *auto*
qed

B.23.3 Equivalent axiomatization

lemma (in *RestrLResSemi*) *incl-restr-lres*[*intro*]:
assumes [*intro*]: $T : a \leftrightarrow c$
assumes [*intro*]: $R : c \leftrightarrow b$
assumes *TR*[*intro*]: *rang* $T \sqsubseteq$ *dom* R
shows $T \sqsubseteq R \vdash (T \odot R)$
proof −
have $(\forall\ X\ .\ X : a \leftrightarrow c \longrightarrow X \sqsubseteq T \longrightarrow\ X \sqsubseteq R \vdash (T \odot R))$
proof (*intro strip*)

fix X
assume [*intro*]:$X \in a \leftrightarrow c$
assume XT[*simp*]: $X \sqsubseteq T$
hence [*intro*]: $(X \odot R \sqsubseteq T \odot R)$
by (*rule-tac comp-incl-mon1*, *best*+)
hence XT[*intro*]: *rang* $X \sqsubseteq$ *rang* T **by** *auto*
from XT *TR* **have** *rang* $X \sqsubseteq$ *dom* R
by (*rule-tac incl-trans*, *auto*)
thus $X \sqsubseteq (R \vdash (T \odot R))$ **by** (*rule-tac restr-lres1*, *best*+)
qed
thus *?thesis* **by** (*rule-tac indir-ineq2*, *auto*)
qed

lemma (in *RestrLResSemi*) *restr-lrescmp-incl*:
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $S : a \leftrightarrow b$
shows $(R \vdash S) \odot R \sqsubseteq S$
proof −
let *?X* $= R \vdash S$
have [*intro*]: *?X* $: a \leftrightarrow c$ **by** *auto*
moreover have [*intro*]: *?X* $\sqsubseteq (R \vdash S)$ **by** (*rule incl-refl*, *best*)
have (*?X* $\odot R \sqsubseteq S$) & (*rang ?X* $\sqsubseteq$ *dom* R)
by (*rule restr-lres* [*THEN iffD2*], *auto*)
thus *?thesis* **by** (*best*+)
qed

Add the following new properties.

lemma (in *RestrLResSemi*) *restr-lres-incl-new*:
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $S : a \leftrightarrow b$
shows *rang* $(R \vdash S) \sqsubseteq$ *dom* R
proof −
let *?X* $= R \vdash S$
have [*intro*]: *?X* $: a \leftrightarrow c$ **by** *auto*
moreover have [*intro*]: *?X* $\sqsubseteq (R \vdash S)$ **by** (*rule incl-refl*, *best*)
have (*?X* $\odot R \sqsubseteq S$) & (*rang ?X* $\sqsubseteq$ *dom* R)
by (*rule restr-lres* [*THEN iffD2*], *auto*)
thus *?thesis* **by** (*best*+)
qed

—-Residual properties in Furusawa-Kahl-1998—-

Proposition 4.4 (i)

lemma (in *RestrLResSemi*) *restr-lrescom-incl-fs*:
assumes [*intro*]:$S : a \leftrightarrow c$
assumes [*intro*]:$R : b \leftrightarrow c$
assumes [*intro*]:$T : d \leftrightarrow c$
shows $(R \vdash S) \odot (T \vdash R) \sqsubseteq (T \vdash S)$
proof (*rule restr-lres* [*THEN iffD1*], *best*)
show $S : a \leftrightarrow c$ **by** *auto*
next
show $(R \vdash S) \odot T \vdash R \in a \leftrightarrow d$ **by** *auto*
next
show $((R \vdash S) \odot T \vdash R) \odot T \sqsubseteq S \wedge rang\ ((R \vdash S) \odot T \vdash R) \sqsubseteq dom\ T$
proof −
have[*intro*]: $(T \vdash R) \odot T \sqsubseteq R$ **by** (*rule restr-lrescmp-incl*, *auto*)
have[*intro*]: $(R \vdash S) \odot R \sqsubseteq S$ **by** (*rule restr-lrescmp-incl*, *auto*)
have $((R \vdash S) \cdot (T \vdash R)) \cdot T = (R \vdash S) \cdot ((T \vdash R) \cdot T)$
by (*rule cmp-assoc*, *auto*)
moreover have ... $\sqsubseteq (R \vdash S) \odot R$ **by** *auto*
moreover have ... $\sqsubseteq S$ **by** *auto*
ultimately have $l : ((R \vdash S) \cdot (T \vdash R)) \cdot T \sqsubseteq S$
by (*rule-tac incl-trans*, *auto*)
have[*intro*]: $rang\ (T \vdash R) \sqsubseteq dom\ T$
by (*rule restr-lres-incl-new*, *auto*)
have $rang((R \vdash S) \cdot (T \vdash R)) \sqsubseteq rang(\ rang(R \vdash S) \cdot (T \vdash R)\ )$
by (*rule ran-decomp*, *auto*)
moreover have ... $\sqsubseteq rang\ (T \vdash R)$ **by** *best+*
ultimately have $rang((R \vdash S) \odot (T \vdash R)) \sqsubseteq rang\ (T \vdash R)$
by (*rule-tac incl-trans*, *auto*)
moreover have ... $\sqsubseteq dom\ T$ **by** *best+*
ultimately have *ll*: $rang\ ((R \vdash S) \odot T \vdash R) \sqsubseteq dom\ T$
by (*rule-tac incl-trans*, *auto*)
from *l ll* **show** *?thesis* **by** *auto*
qed
qed

Proposition 4.4 (iii) holds for restricted residuals if R'=R and Q'=Q.

lemma (in *RestrLResSemi*) *restr-lrescom-incl-ohk*:

assumes [*intro*]: $S : a \leftrightarrow c$
assumes [*intro*]: S': $a \leftrightarrow c$
assumes [*intro*]: $R : b \leftrightarrow c$
assumes S[*intro*]: $S \sqsubseteq S'$
shows $(R \vdash S) \sqsubseteq (R \vdash S')$
proof (*rule-tac indir-ineq2*)
show $R \vdash S \subset a \leftrightarrow b$ **by** *auto*
next
show $R \vdash S' \in a \leftrightarrow b$ **by** *auto*
next
show $\forall C.\ C \in a \leftrightarrow b \longrightarrow C \sqsubseteq (R \vdash S) \longrightarrow C \sqsubseteq (R \vdash S')$
proof (*intro strip*)
fix C
assume [*intro*]: $C : a \leftrightarrow b$
assume C[*intro*]: $C \sqsubseteq (R \vdash S)$
show $C \sqsubseteq (R \vdash S')$
proof −
from C **have** $(C \odot R \sqsubseteq S)$ **by** (*rule-tac restr-lres-incl2*, *auto*)
moreover have ... $\sqsubseteq S'$ **by** *auto*
ultimately have *r1*: $C \odot R \sqsubseteq S'$ **by** (*rule incl-trans*, *auto*)
from C **have** *r2*: $rang\ C \sqsubseteq dom\ R$ **by** *auto*
from *r1 r2* **show** *?thesis* **by**(*rule-tac restr-lres1*, *auto*)
qed
qed
qed

Proposition 4.5 (i) holds for restricted residuals when the following FR assumption is added.

lemma (in *RestrLResSemi*) *restr-lrescom-incl-ex*:
assumes [*intro*]:$F : a \leftrightarrow b$
assumes [*intro*]:$R : b \leftrightarrow c$
assumes [*intro*]:$S : d \leftrightarrow c$
assumes *FR*: $rang\ F \sqsubseteq dom\ R$
shows $(F \odot (S \vdash R)) \sqsubseteq (S \vdash (F \odot R))$
proof −
have[*intro*]: $F \sqsubseteq (R \vdash (F \odot R))$ **by** *auto*
have $(F \odot (S \vdash R)) \sqsubseteq ((R \vdash (F \odot R)) \odot (S \vdash R))$ **by** *auto*
moreover have ... $\sqsubseteq (S \vdash (F \odot R))$
by(*rule restr-lrescom-incl-fs*, *auto*)
ultimately show *?thesis* **by** (*rule incl-trans*, *auto*)

qed

end

B.24 Semigroupoids with Restricted Right Residuals

theory *RestrRResSemi*
imports *OrdSemiRestrRes DivAllRecord*
begin

B.24.1 Definitions

locale *RestrRResSemi* = *OrdSemiRestrRes OSRR* +
assumes *restrrightRes-homset*[*intro*,*simp*]: ⟦ $L : a \leftrightarrow c$; $S : a \leftrightarrow b$ ⟧
$\Longrightarrow (S \dashv L) : c \leftrightarrow b$
assumes *restrrightRes*[*intro*,*simp*]: ⟦ $L : a \leftrightarrow c$; $S : a \leftrightarrow b$ ⟧
$\Longrightarrow$ *haveRestrRightRes* $S\ L\ (S \dashv L)$

B.24.2 Auxiliary Lemmas

lemma (**in** *RestrRResSemi*) *restrrightRes-src*[*simp*]:
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $S : a \leftrightarrow b$
shows *src* $(S \dashv L) = c$
proof −
have $(S \dashv L) : c \leftrightarrow b$ **by** (*rule restrrightRes-homset*, *auto*)
thus *?thesis* **by** (*rule homset-src*)
qed

lemma (**in** *RestrRResSemi*) *restrrightRes-trg*[*simp*]:
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $S : a \leftrightarrow b$
shows *trg* $(S \dashv L) = b$
proof −
have $(S \dashv L) : c \leftrightarrow b$ **by** (*rule restrrightRes-homset*, *auto*)
thus *?thesis* **by** (*rule homset-trg*)
qed

lemma (**in** *RestrRResSemi*) *restrrightRes-defined*[*simp*]:
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $S : a \leftrightarrow b$
shows *Mor* $(S \dashv L)$
proof (*rule homset-Mor*)
show $S \dashv L : c \leftrightarrow b$ **by** (*rule restrrightRes-homset*, *auto*)
qed

lemma (**in** *RestrRResSemi*) *restr-rres-src*[*simp*]:
assumes [*simp*]: *src* L = *src* S
assumes [*simp*]: *Mor* L
assumes [*simp*]: *Mor* S
shows *src* $(S \dashv L)$ = *trg* L
proof −
have $S \dashv L$: *trg* $L \leftrightarrow$ *trg* S
by (*rule restrrightRes-homset*, *rule homset1*, *auto*)
thus *?thesis* **by** (*rule homset-src*)
qed

lemma (**in** *RestrRResSemi*) *restr-rres-trg*[*simp*]:
assumes [*simp*]: *src* L = *src* S
assumes [*simp*]: *Mor* L
assumes [*simp*]: *Mor* S
shows *trg* $(S \dashv L)$ = *trg* S
proof −
have $S \dashv L$: *trg* $L \leftrightarrow$ *trg* S
by (*rule restrrightRes-homset*, *rule homset1*, *auto*)
thus *?thesis* **by** (*rule homset-trg*)
qed

lemma (**in** *RestrRResSemi*) *restr-rres*:
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $X : c \leftrightarrow b$
shows $((L \odot X \sqsubseteq S) \wedge (dom\ X \sqsubseteq rang\ L)) = (X \sqsubseteq (S \dashv L))$
proof −
have *haveRestrRightRes* $S\ L\ (S \dashv L)$ **by** (*rule restrrightRes*, *best*+)
thus *?thesis* **by** (*rule haveRestrRightRes*, *best*+)
qed

lemmas (in *RestrRResSemi*) *restr-rres1 = restr-rres* [*THEN iffD1*]
lemmas (in *RestrRResSemi*) *restr-rres2 = restr-rres* [*THEN iffD2*]

Added the following two auxiliary lemmas

lemma (in *RestrRResSemi*) *restr-rres-incl1*[*intro*]:
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $X : c \leftrightarrow b$
assumes [*intro*]: S: $a \leftrightarrow b$
assumes XL[*intro*]: $X \sqsubseteq S \dashv L$
shows $dom\ X \sqsubseteq rang\ L$
proof −
from XL **have** *ss*: $(L \odot X \sqsubseteq S) \wedge (dom\ X \sqsubseteq rang\ L)$
by (*rule-tac restr-rres* [*THEN iffD2*], *auto*)
from *ss* **show** *?thesis* **by** *auto*
qed

lemma (in *RestrRResSemi*) *restr-rres-incl2*[*intro*]:
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $X : c \leftrightarrow b$
assumes [*intro*]: S: $a \leftrightarrow b$
assumes XL[*intro*]: $X \sqsubseteq S \dashv L$
shows $(L \odot X \sqsubseteq S)$
proof −
from XL **have** *ss*: $(L \odot X \sqsubseteq S) \wedge (dom\ X \sqsubseteq rang\ L)$
by (*rule-tac restr-rres* [*THEN iffD2*], *auto*)
from *ss* **show** *?thesis* **by** *auto*
qed

B.24.3 Equivalent axiomatization

lemma (in *RestrRResSemi*) *incl-restr-rres*[*intro*]:
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]: $T : c \leftrightarrow b$
assumes LT[*intro*]: $dom\ T \sqsubseteq rang\ L$
shows $T \sqsubseteq ((L \odot T) \dashv L)$
proof −
have $(\forall\ X\ .\ X : c \leftrightarrow b \longrightarrow X \sqsubseteq T \longrightarrow X \sqsubseteq (L \odot T) \dashv L)$
proof (*intro strip*)
fix X

assume [*intro*]: $X : c \leftrightarrow b$
assume [*simp*]: $X \sqsubseteq T$
hence [*intro*]: $L \odot X \sqsubseteq L \odot T$
by (*rule-tac comp-incl-mon2*, *best+*)
hence XT[*intro*]: $dom\ X \sqsubseteq dom\ T$ **by** *auto*
from XT LT **have** $dom\ X \sqsubseteq rang\ L$ **by** (*rule-tac incl-trans*, *auto*)
thus $X \sqsubseteq (L \odot T) \dashv L$ **by** (*rule-tac restr-rres1*, *best+*)
qed
thus *?thesis* **by** (*rule-tac indir-ineq2*, *auto*)
qed

lemma (in *RestrRResSemi*) *restr-rrescmp-incl*:
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $L : a \leftrightarrow c$
shows $L \odot (S \dashv L) \sqsubseteq S$
proof −
let *?X*=$S \dashv L$
have [*intro*]: $?X : c \leftrightarrow b$ **by** *auto*
moreover have [*intro*]: $?X \sqsubseteq S \dashv L$ **by** (*rule incl-refl*, *best*)
moreover have $(L \odot ?X \sqsubseteq S)$ & $(dom\ ?X \sqsubseteq rang\ L)$
by (*rule restr-rres* [*THEN iffD2*], *auto*)
thus *?thesis* **by** (*best+*)
qed

Add the following new properties.

lemma (in *RestrRResSemi*) *restr-rres-incl-new*:
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $L : a \leftrightarrow c$
shows $dom\ (S \dashv L) \sqsubseteq rang\ L$
proof −
let *?X*=$S \dashv L$
have [*intro*]: $?X : c \leftrightarrow b$ **by** *auto*
moreover have [*intro*]: $?X \sqsubseteq S \dashv L$ **by** (*rule incl-refl*, *best*)
moreover have $(L \odot ?X \sqsubseteq S)$ & $(dom\ ?X \sqsubseteq rang\ L)$
by (*rule restr-rres* [*THEN iffD2*], *auto*)
thus *?thesis* **by** (*best+*)
qed

——-Residual properties in Furusawa-Kahl-1998——

Proposition 4.4 (i) holds for restricted residuals

lemma (in *RestrRResSemi*) *restr-rrescom-incl-fs*:
assumes [*intro*]:$S : a \leftrightarrow c$
assumes [*intro*]:$L : a \leftrightarrow b$
assumes [*intro*]:$U : a \leftrightarrow d$
shows $(S \dashv L) \odot (U \dashv S) \sqsubseteq (U \dashv L)$
proof −
have[*intro*] : $L \odot (S \dashv L) \sqsubseteq S$ **by** (*rule restr-rrescmp-incl, auto*)
have[*intro*] : $S \odot (U \dashv S) \sqsubseteq U$ **by** (*rule restr-rrescmp-incl, auto*)
have $L \cdot\!\!\odot ((S \dashv L) \cdot\!\!\odot (U \dashv S)) = (L \cdot\!\!\odot (S \dashv L)) \cdot (U \dashv S)$
by (*rule cmp-assoc-sym, auto*)
moreover have ... $\sqsubseteq S \odot (U \dashv S)$ **by** *auto*
moreover have ... $\sqsubseteq U$ **by** *auto*
ultimately have $r : L \cdot\!\!\odot ((S \dashv L) \cdot\!\!\odot (U \dashv S)) \sqsubseteq U$
by (*rule-tac incl-trans, auto*)
from r **show** $(S \dashv L) \odot (U \dashv S) \sqsubseteq (U \dashv L)$
proof (*rule-tac restr-rres* [*THEN iffD1*], *auto*)
show *dom* $((S \dashv L) \mathbin{:} U \dashv S) \sqsubseteq$ *rang* L
proof −
have[*intro*]: *dom*$(S \dashv L) \sqsubseteq$ *rang* L
by (*rule restr-rres-incl-new, auto*)
have *dom*$((S \dashv L) \odot (U \dashv S)) \sqsubseteq$ *dom*$((S \dashv L) \odot$ *dom*$(U \dashv S))$
by (*rule dom-decomp, auto*)
moreover have ... $\sqsubseteq$ *dom* $(S \dashv L)$ **by** *best+*
ultimately have *dom*$((S \dashv L) \odot (U \dashv S)) \sqsubseteq$ *dom* $(S \dashv L)$
by (*rule-tac incl-trans, auto*)
moreover have ... $\sqsubseteq$ *rang* L **by** *best+*
ultimately show *dom*$((S \dashv L) \odot (U \dashv S)) \sqsubseteq$ *rang* L
by (*rule-tac incl-trans, auto*)
qed
qed
qed

Proposition 4.4 (iii) holds for restricted residuals if R'=R and Q'=Q.

lemma (in *RestrRResSemi*) *restr-rrescom-incl-ohk*:
assumes [*intro*]: $S : a \leftrightarrow c$
assumes [*intro*]: S': $a \leftrightarrow c$
assumes [*intro*]: $Q : a \leftrightarrow b$
assumes S[*intro*]: $S \sqsubseteq S'$

shows $(S \dashv Q) \sqsubseteq (S' \dashv Q)$
proof (*rule-tac indir-ineq2, best+*)
show $\forall C.\ C \in b \leftrightarrow c \longrightarrow C \sqsubseteq (S \dashv Q) \longrightarrow C \sqsubseteq (S' \dashv Q)$
proof (*intro strip*)
fix C
assume [*intro*]: $C : b \leftrightarrow c$
assume C[*intro*]: $C \sqsubseteq (S \dashv Q)$
show $C \sqsubseteq (S' \dashv Q)$
proof −
have $Q \odot C \sqsubseteq S$ **by** *auto*
moreover have ... $\sqsubseteq S'$ **by** *auto*
ultimately have l: $Q \odot C \sqsubseteq S'$ **by** (*rule incl-trans, auto*)
have *dom* $C \sqsubseteq$ *rang* Q **by** *auto*
from l **show** *?thesis* **by**(*rule-tac restr-rres1, auto*)
qed
qed
qed

Proposition 4.5 (i) holds for restricted residuals when the following TQ assumption is added.

lemma (in *RestrRResSemi*) *Restr-rrescom-incl-ex*:
assumes [*intro*]:$U : a \leftrightarrow b$
assumes [*intro*]:$Q : a \leftrightarrow c$
assumes [*intro*]:$T : c \leftrightarrow d$
assumes *TQ*: *dom* $T \sqsubseteq$ *rang* Q
shows $((Q \dashv U) \odot T) \sqsubseteq ((Q \odot T) \dashv U)$
proof −
from *TQ* **have**[*intro*]: $T \sqsubseteq ((Q \odot T) \dashv Q)$ **by** *auto*
have $((Q \dashv U) \odot T) \sqsubseteq ((Q \dashv U) \mathbin{:} ((Q \odot T) \dashv Q))$ **by** *auto*
moreover have ... $\sqsubseteq ((Q \cdot\!\!\odot T) \dashv U)$ **by**(*rule restr-rrescom-incl-fs, auto*)
ultimately show *?thesis* **by** (*rule incl-trans, auto*)
qed

end

B.25 Semigroupoids with restricted residuals

theory *RestrResSemi*
imports *RestrLResSemi RestrRResSemi*

begin

locale *RestrResSemi = RestrLResSemi RRS + RestrRResSemi RRS*

end

B.26 Restricted Residuals and Standard Residuals.

theory *RestrResAndRes*
imports *OrdSemiRestrRes OrdSemiRes*
begin

B.26.1 Theorems

lemma (in *OrdSemiRestrRes*) *RestrRightRes-RightRes*:
assumes [*simp,intro*]: S: $a \leftrightarrow b$
assumes [*simp,intro*]: $R : c \leftrightarrow b$
assumes [*simp,intro*]: $L : a \leftrightarrow c$
assumes [*intro*]:*haveRightRes S L R*
shows *haveRestrRightRes S L* (*rang L* $\odot$ *R*)
proof(*rule haveRestrRightRes-def* [*THEN sym, THEN iffD1*],*best+*)
show *i*:$\forall X{\in}c \leftrightarrow b.$ $((L \odot X \sqsubseteq S \wedge dom\ X \sqsubseteq rang\ L) = (X \sqsubseteq rang\ L \odot R))$
proof (*intro strip*)
fix X
assume[*intro*]: $X : c \leftrightarrow b$
show $(L \odot X \sqsubseteq S \wedge dom\ X \sqsubseteq rang\ L) = (X \sqsubseteq rang\ L \odot R)$
proof −
have *R*:$(L \ddot{\cdot} X \sqsubseteq S) = (X \sqsubseteq R)$ **by** (*rule haveRightRes, best+*)
have $L \ddot{\cdot} X \sqsubseteq S \wedge dom\ X \sqsubseteq rang\ L \Longrightarrow X \sqsubseteq rang\ L \cdot R$
proof −
have *X*: $X \sqsubseteq dom\ X \odot X$ **by** *auto*
from *R* **have** $(L \odot X \sqsubseteq S \wedge dom\ X \sqsubseteq rang\ L)$
$= (dom\ X \sqsubseteq rang\ L \wedge X \sqsubseteq R)$ **by** *auto*
also have ... $\Longrightarrow (dom\ X \odot X \sqsubseteq rang\ L \odot R)$ **by** *auto*
from *X* **have** $(dom\ X \odot X \sqsubseteq rang\ L \odot R) \Longrightarrow (X \sqsubseteq rang\ L \ddot{\cdot} R)$
by (*rule incl-trans, auto*)
ultimately show $L \odot X \sqsubseteq S \wedge dom\ X \sqsubseteq rang\ L \Longrightarrow X \sqsubseteq rang\ L \odot R$
by *best+*
qed

moreover have $X \sqsubseteq rang\ L \odot R \Longrightarrow L \odot X \sqsubseteq S \wedge dom\ X \sqsubseteq rang\ L$
proof(*rule conjI*)
show $X \sqsubseteq rang\ L \odot R \Longrightarrow L \ddot{\cdot} X \sqsubseteq S$
proof −
have [*intro*]: $rang\ L \odot R \sqsubseteq R$ **by** *auto*
have [*intro*]:$X \sqsubseteq rang\ L \odot R \Longrightarrow X \sqsubseteq R$
by (*rule tac incl-trans, auto*)
from *R* **have**[*intro*]: $X \sqsubseteq R \Longrightarrow L \odot X \sqsubseteq S$ **by** *auto*
show $X \sqsubseteq rang\ L \odot R \Longrightarrow L \odot X \sqsubseteq S$ **by** *auto*
qed
next show $X \sqsubseteq rang\ L \odot R \Longrightarrow dom\ X \sqsubseteq rang\ L$
proof −
have $dom\ (rang\ L \odot R) \sqsubseteq rang\ L$ **by** *auto*
also have[*intro*]: $X \sqsubseteq rang\ L \odot R \Longrightarrow dom\ X \sqsubseteq dom\ (rang\ L \odot R)$
by (*rule dom-incl-mon, auto*)
have[*intro*]: $dom\ X \sqsubseteq dom\ (rang\ L \odot R) \Longrightarrow dom\ X \sqsubseteq rang\ L$
by (*rule-tac incl-trans, auto*)
show $X \sqsubseteq rang\ L \odot R \Longrightarrow dom\ X \sqsubseteq rang\ L$ **by** *auto*
qed
qed
ultimately show $(L \odot X \sqsubseteq S \wedge dom\ X \sqsubseteq rang\ L) = (X \sqsubseteq rang\ L \odot R)$
by (*rule iffI, best+*)
qed
qed
qed

lemma (in *OrdSemiRestrRes*) *RestrleftRes-LeftRes*:
assumes [*intro*]: $S : a \leftrightarrow b$
assumes [*intro*]: $R : c \leftrightarrow b$
assumes [*intro*]: $L : a \leftrightarrow c$
assumes [*intro*]:*haveLeftRes S R L*
shows *haveRestrLeftRes S R* (*L* $\odot$ *dom R*)
proof (*rule haveRestrLeftRes-def* [*THEN sym, THEN iffD1*],*best+*)
show $\forall X{\in}a \leftrightarrow c.\ (X \cdot R \sqsubseteq S \wedge rang\ X \sqsubseteq dom\ R) = (X \sqsubseteq (L \ddot{\cdot} dom\ R))$
proof (*intro strip*)
fix X
assume[*intro*]: $X : a \leftrightarrow c$
show $((X \ddot{\cdot} R \sqsubseteq S) \wedge (rang\ X \sqsubseteq dom\ R)) = (X \sqsubseteq (L \ddot{\cdot} dom\ R))$

```
  proof −
   have L:(X ⊙ R ⊑ S) = (X ⊑ L)
     by (rule haveLeftRes, best+)
   have X ⊙ R ⊑ S ∧ rang X ⊑ dom R ⟹ X ⊑ L ⊙ dom R
   proof −
    have X: X ⊑ (X ⊙ rang X) by auto
    from L have (X ⊙ R ⊑ S ∧ rang X ⊑ dom R)
            = (X ⊑ L ∧ rang X ⊑ dom R) by auto
    also have (X ⊑ L ∧ rang X ⊑ dom R)⟹
                  ((X ⊙ rang X) ⊑ (L ⊙ dom R)) by auto
    from X have ((X ⊙ rang X) ⊑ (L ⊙ dom R)) ⟹ X ⊑ (L ⊙ dom R)
                          by (rule-tac incl-trans, auto)
    ultimately show X ⊙ R ⊑ S ∧ rang X ⊑ dom R
                  ⟹ X ⊑ L ⊙ dom R by auto
   qed
   moreover have X ⊑ L ⊙ dom R ⟹ X ⊙ R ⊑ S ∧ rang X ⊑ dom R
   proof (rule conjI)
     show X ⊑ L ⊙ dom R ⟹ X ⊙ R ⊑ S
     proof −
      have [intro]: L ⊙ dom R ⊑ L by auto
      have [intro]:X ⊑ L ⊙ dom R ⟹ X ⊑ L
                          by (rule-tac incl-trans, auto)
      from L have[intro]: X ⊑ L ⟹ X ⊙ R ⊑ S by auto
      show X ⊑ L ⊙ dom R ⟹ X ⊙ R ⊑ S by auto
     qed
    next show X ⊑ L ⊙ dom R ⟹ rang X ⊑ dom R
     proof −
      have rang( L ⊙ dom R) ⊑ dom R by auto
      also have[intro]: X ⊑ L ⊙ dom R ⟹
       rang X ⊑ rang (L ⊙ dom R) by (rule ran-incl-mon, auto)
      have[intro]: rang X ⊑ rang( L ⊙ dom R) ⟹ rang X ⊑ dom R
                          by (rule-tac incl-trans, auto)
      show X ⊑ L ⊙ dom R ⟹ rang X ⊑ dom R by auto
     qed
    qed
   ultimately show ((X ⊙ R ⊑ S) ∧ (rang X ⊑ dom R))
          = (X ⊑ (L ⊙ dom R)) by (rule iffI, best+)
  qed
 qed
qed
```



```
end
```

B.27 OSGC with Standard Residuals

```
theory ResOSGC
imports ResSemi ConvOrdSemi
begin
```

B.27.1 Definitions

```
locale ResOSGC = ResSemi ROSGC + ConvOrdSemi ROSGC
```

B.27.2 Theorems

—Residual property (Proposition 4.1) in Furusawa-Kahl-1998—

```
lemma (in ResOSGC) resOSGC-eq:
 assumes [intro]:Q : a ↔ b
 assumes [intro]:S : a ↔ c
 shows (S ⇀ Q) = (Q˘ ↼ S˘)˘
proof (rule indirect-equality [THEN iffD2], best+)
 show ∀ C. C ∈ b ↔ c ⟶ (C ⊑ S ⇀ Q) = (C ⊑ (Q˘ ↼ S˘)˘)
 proof (intro strip)
  fix C
  assume [intro]:C : b ↔ c
  show (C ⊑ S ⇀ Q) = (C ⊑ (Q˘ ↼ S˘)˘)
  proof −
   have C ⊑ S ⇀ Q ⟹ C ⊑ (Q˘ ↼ S˘)˘
   proof −
    assume c: C ⊑ S ⇀ Q
    show C ⊑ (Q˘ ↼ S˘)˘
    proof −
     from c have [intro]: (Q ⊙ C ⊑ S ) by (rule-tac rres2, auto)
     have ((Q ⊙ C)˘ ⊑ S˘) by (auto)
     have c1: (C˘ ⊙ Q˘) ⊑ S˘ by (rule conv-cmp [THEN subst],auto)
     from c1 have c2: C˘ ⊑ (Q˘ ↼ S˘) by (rule-tac lres1, auto)
```

```
    from c2 have c3:(C⌣)⌣ ⊑ (Q⌣ ↼ S⌣)⌣ by auto
    from c3 show  C ⊑ (Q⌣ ↼ S⌣)⌣ by (rule-tac conv-idem [THEN subst],auto)
   qed
  qed
  moreover have C ⊑ (Q⌣ ↼ S⌣)⌣ ⟹ C ⊑ S ⇁ Q
  proof −
   assume C: C ⊑ (Q⌣ ↼ S⌣)⌣
   show C ⊑ S ⇁ Q
   proof −
    from C have C1: (C⌣)⌣ ⊑ (Q⌣ ↼ S⌣)⌣
          by (rule-tac conv-idem [THEN sym, THEN subst], auto)
    from C1 have C2: C⌣ ⊑ (Q⌣ ↼ S⌣)
                by (rule-tac conv-incl [THEN iffD1],  auto)
    from C2 have C3: (C⌣ ⨾ Q⌣) ⊑ S⌣ by (rule-tac lres2, auto)
    from C3 have [intro]: (Q ⨾ C)⌣ ⊑ S⌣
           by (rule-tac conv-cmp [THEN sym, THEN subst],auto)
    have [intro]: (Q ⨾ C) ⊑ S
                 by (rule-tac conv-incl [THEN iffD1],auto)
    show C ⊑ S ⇁ Q by (rule-tac rres1, auto)
   qed
  qed
  ultimately show ?thesis by(rule iffI, best+)
 qed
qed
qed


end
```

B.28 OSGC with Domain and Range Operators

```
theory RDConvOrdSemi
imports DomSemi RanSemi ConvOrdSemi
begin
```

Add the theory to provide some properties of OSGC with Domain and Range Operators. The properties in the theory are used to support the properties of restricted residuals in RestrResOSGC theory.

B.28.1 Definitions

locale *RDConvOrdSemi* = *DomSemi RDCOS* + *RanSemi RDCOS* + *ConvOrdSemi RDCOS*

B.28.2 Theorems

conv(rang R) is subidentity.

```
lemma (in RDConvOrdSemi )RDCOS-convRan-left:
 assumes [intro]: R : a ↔ b
 assumes [intro]: f : b ↔ c
 shows (rang R)⌣ ⨾ f ⊑ f
 proof −
  have [intro]: f⌣ ⨾ rang R ⊑ f⌣ by auto
  have (f⌣ ⨾ rang R)⌣ ⊑ (f⌣)⌣ by auto
  have (rang R)⌣ ⨾ (f⌣)⌣ ⊑ (f⌣)⌣ by (rule conv-cmp [THEN subst],auto)
  hence L:  (rang R)⌣ ⨾ f ⊑ f by (rule-tac conv-idem [THEN subst], best)
  from L show ?thesis by auto
 qed

lemma (in RDConvOrdSemi )RDCOS-convRan-right:
 assumes [intro]: R : a ↔ b
 assumes [intro]: f : c ↔ b
 shows f ⨾ (rang R)⌣ ⊑ f
 proof −
  have [intro]: rang R ⨾ f⌣ ⊑ f⌣ by auto
  have (rang R ⨾ f⌣)⌣ ⊑ (f⌣)⌣ by auto
  have (f⌣)⌣ ⨾ (rang R)⌣  ⊑ (f⌣)⌣ by (rule conv-cmp [THEN subst],auto)
  hence L:  f ⨾ (rang R)⌣  ⊑ f by (rule-tac conv-idem [THEN subst], best)
  from L show ?thesis by auto
 qed
```

conv(rang R) : ISId b

```
lemma (in RDConvOrdSemi )RDCOS-convRan-I:
 assumes [intro]: R : a ↔ b
 shows (rang R)⌣ ⨾ (rang R)⌣ = (rang R)⌣
 proof −
  have i: (rang R ⨾ rang R) = rang R
  proof (rule isISId-2)
```

```
  show rang R ∈ b ↔ b by auto
 next
  show isISId (rang R) by (rule ISId,auto)
qed
from i have ii: (rang R ⊙ rang R)⌣ = (rang R)⌣
          by (rule-tac conv-equality [THEN iffD2], auto)
from ii show ?thesis by (rule-tac conv-cmp [THEN subst],auto)
qed
```

conv(dom R) is subidentity.

```
lemma (in RDConvOrdSemi ) RDCOS-convRan-isISId[simp, intro]:
assumes [intro]:R : a ↔ b
shows (rang R)⌣: ISId b
proof (rule ISId-intro)
 show (rang R)⌣ ∈ b ↔ b by auto
next
 show isISId ((rang R)⌣)
 proof (rule isISId-intro)
  show (rang R)⌣ ∈ b ↔ b by auto
 next
  show isSId ((rang R)⌣)
  proof (rule isSId-def [THEN iffD2])
   have[intro]: (rang R)⌣ ∈ b ↔ b by auto
   show ∀ c.(∀ f. f ∈ b ↔ c ⟶ (rang R)⌣ ⊙ f ⊑ f)
         ∧ (∀ g. g ∈ c ↔ b ⟶ g ⊙ (rang R)⌣ ⊑ g)
    proof(intro strip)
     fix c
     show (∀ f. f ∈ b ↔ c ⟶ (rang R)⌣ ⊙ f ⊑ f)
          ∧ (∀ g. g ∈ c ↔ b ⟶ g ⊙ (rang R)⌣ ⊑ g)
      proof(rule conjI)
       show ∀ f. f ∈ b ↔ c ⟶ (rang R)⌣ ⊙ f ⊑ f
        proof (intro strip)
         fix f
         assume [intro]: f ∈ b ↔ c
         show (rang R)⌣ ⊙ f ⊑ f by (rule RDCOS-convRan-left, auto)
        qed
       next
        show ∀ g. g ∈ c ↔ b ⟶ g ⊙ (rang R)⌣ ⊑ g
         proof (intro strip)
          fix g
```



```
          assume [intro]: g ∈ c ↔ b
          show g ⊙ (rang R)⌣ ⊑ g by (rule RDCOS-convRan-right, auto)
         qed
       qed
     qed
     next
     show (rang R)⌣ ∈ b ↔ b by auto
   qed
   next
    show (rang R)⌣ ⊙ (rang R)⌣ = (rang R)⌣
          by (rule RDCOS-convRan-I, auto)
 qed
qed

lemma (in RDConvOrdSemi )RDCOS-convDom-left:
 assumes [intro]: R : a ↔ b
 assumes [intro]: f : a ↔ c
 shows (dom R)⌣ ⊙ f ⊑ f
 proof –
  have [intro]: f⌣ ⊙ dom R ⊑ f⌣ by auto
  have (f⌣ ⊙ dom R)⌣ ⊑ (f⌣)⌣ by auto
  have (dom R)⌣ ⊙ (f⌣)⌣ ⊑ (f⌣)⌣ by (rule conv-cmp [THEN subst],auto)
  hence L: (dom R)⌣ ⊙ f ⊑ f by (rule-tac conv-idem [THEN subst], best)
  from L show ?thesis by auto
 qed

lemma (in RDConvOrdSemi )RDCOS-convDom-right:
 assumes [intro]: R : a ↔ b
 assumes [intro]: f : c ↔ a
 shows f ⊙ (dom R)⌣ ⊑ f
 proof –
  have [intro]: dom R ⊙ f⌣ ⊑ f⌣ by auto
  have (dom R ⊙ f⌣)⌣ ⊑ (f⌣)⌣ by auto
  have (f⌣)⌣ ⊙ (dom R)⌣ ⊑ (f⌣)⌣ by (rule conv-cmp [THEN subst],auto)
  hence L: f ⊙ (dom R)⌣ ⊑ f by (rule-tac conv-idem [THEN subst], best)
  from L show ?thesis by auto
 qed
```

conv(rang R) : ISId a

```
lemma (in RDConvOrdSemi )RDCOS-convDom-I:
```

```
 assumes [intro]: R : a ↔ b
 shows (dom R)˘ ⊙ (dom R)˘ = (dom R)˘
 proof −
  have i: (dom R ⊙ dom R) = dom R
  proof (rule isISId-2)
   show dom R ∈ a ↔ a by auto
  next
   show isISId (dom R) by (rule ISId,auto)
  qed
  from i have ii: (dom R ⊙ dom R)˘ = (dom R)˘
              by (rule-tac conv-equality [THEN iffD2], auto)
  from ii show ?thesis by (rule-tac conv-cmp [THEN subst],auto)
qed


lemma (in RDConvOrdSemi ) RDCOS-convDom-isISId[simp, intro]:
 assumes [intro]:R : a ↔ b
 shows (dom R)˘: ISId a
 proof (rule ISId-intro)
  show (dom R)˘ ∈ a ↔ a by auto
  next
  show isISId ((dom R)˘)
  proof (rule isISId-intro)
   show (dom R)˘ ∈ a ↔ a by auto
  next
   show isSId ((dom R)˘)
   proof (rule isSId-def [THEN iffD2])
    have[intro]: (dom R)˘ ∈ a ↔ a by auto
    show ∀ c. (∀ f. f ∈ a ↔ c ⟶ (dom R)˘ ⊙ f ⊑ f)
         ∧ (∀ g. g ∈ c ↔ a ⟶ g ⊙ (dom R)˘ ⊑ g)
    proof(intro strip)
     fix c
     show (∀ f. f ∈ a ↔ c ⟶ (dom R)˘ ⊙ f ⊑ f)
          ∧ (∀ g. g ∈ c ↔ a ⟶ g ⊙ (dom R)˘ ⊑ g)
     proof(rule conjI)
      show ∀ f. f ∈ a ↔ c ⟶ (dom R)˘ ⊙ f ⊑ f
      proof (intro strip)
       fix f
       assume [intro]: f ∈ a ↔ c
       show (dom R)˘ ⊙ f ⊑ f by (rule RDCOS-convDom-left, auto)
```



```
      qed
     next
      show ∀ g. g ∈ c ↔ a ⟶ g ⊙ (dom R)˘ ⊑ g
       proof (intro strip)
        fix g
        assume [intro]: g ∈ c ↔ a
        show g ⊙ (dom R)˘ ⊑ g by (rule RDCOS-convDom-right, auto)
       qed
     qed
    qed
    next
    show (dom R)˘ ∈ a ↔ a by auto
   qed
   next
    show (dom R)˘ ⊙ (dom R)˘ = (dom R)˘
         by (rule RDCOS-convDom-I, auto)
 qed
 qedlemma (in RDConvOrdSemi)RDCOS-incl1:
 assumes [intro]:R : a ↔ b
 shows (dom(R˘))˘ ⊑ rang R
proof −
 have R ⊑ R ⊙ rang R by auto
 have R˘ ⊑ (R ⊙ rang R)˘ by auto
 have d: R˘ ⊑ (rang R)˘ ⊙ R˘ by (rule-tac conv-cmp [THEN subst], auto)
 from d have dom(R˘) ⊑ dom((rang R)˘ ⊙ R˘) by auto
 moreover have ... ⊑ (rang R)˘ by auto
 ultimately have dom(R˘) ⊑ (rang R)˘ by (rule incl-trans, auto)
 hence dd: (dom(R˘))˘ ⊑ ((rang R)˘)˘ by auto
 from dd show ?thesis by (rule-tac conv-idem [THEN subst], auto)
qed

lemma (in RDConvOrdSemi)ResOSGC-incl2:
 assumes [intro]:R : a ↔ b
 shows (rang(R˘))˘ ⊑ dom R
proof −
 have R ⊑ dom R · R by auto
 have R˘ ⊑ (dom R ⊙ R)˘ by auto
 have d: R˘ ⊑ R˘ ⊙ (dom R)˘ by (rule-tac conv-cmp [THEN subst], auto)
 from d have rang(R˘) ⊑ rang(R˘ ⊙ (dom R)˘) by auto
 moreover have ... ⊑ (dom R)˘ by auto
```

ultimately have $rang(R^\smile) \sqsubseteq (dom\ R)^\smile$ **by** (*rule incl-trans*, *auto*)
hence *dd*: $(rang(R^\smile))^\smile \sqsubseteq ((dom\ R)^\smile)^\smile$ **by** *auto*
from *dd* **show** *?thesis* **by** (*rule-tac conv-idem* [*THEN subst*], *auto*)
qed

lemma (**in** *RDConvOrdSemi*)*RDCOS-incl3*:
assumes [*intro*]:$R : a \leftrightarrow b$
shows $rang\ R \sqsubseteq (dom(R^\smile))^\smile$
proof −
let $?R' = R^\smile$
have $(?R')^\smile = R$ **by** *auto*
moreover have $(rang((?R')^\smile))^\smile \sqsubseteq dom(?R')$ **by** (*rule ResOSGC-incl2*, *auto*)
ultimately have *r*: $(rang\ R)^\smile \sqsubseteq dom(R^\smile)$ **by** *auto*
from *r* **have** *rr*: $((rang\ R)^\smile)^\smile \sqsubseteq (dom(R^\smile))^\smile$ **by** *auto*
from *rr* **show** *?thesis* **by** (*rule-tac conv-idem* [*THEN subst*], *auto*)
qed

lemma (**in** *RDConvOrdSemi*)*RDCOS-incl4*:
assumes [*intro*]:$R : a \leftrightarrow b$
shows $dom\ R \sqsubseteq (rang(R^\smile))^\smile$
proof −
let $?R' = R^\smile$
have $(?R')^\smile = R$ **by** *auto*
moreover have $(dom((?R')^\smile))^\smile \sqsubseteq rang(?R')$ **by** (*rule RDCOS-incl1*, *auto*)
ultimately have *r*: $(dom\ R)^\smile \sqsubseteq rang(R^\smile)$ **by** *auto*
from *r* **have** *rr*: $((dom\ R)^\smile)^\smile \sqsubseteq (rang(R^\smile))^\smile$ **by** *auto*
from *rr* **show** *?thesis* **by** (*rule-tac conv-idem* [*THEN subst*], *auto*)
qed

lemma (**in** *RDConvOrdSemi*) *RDCOS-ran*:
assumes [*intro*]:$R : a \leftrightarrow b$
shows $rang\ R = (dom(R^\smile))^\smile$
proof (*rule indirect-equality* [*THEN iffD2*], *best+*)
show $\forall C.\ C \in b \leftrightarrow b \longrightarrow (C \sqsubseteq rang\ R) = (C \sqsubseteq (dom\ (R^\smile))^\smile)$
proof (*intro strip*)
fix *C*
assume [*intro*]: $C \in b \leftrightarrow b$

show $(C \sqsubseteq rang\ R) = (C \sqsubseteq (dom\ (R^\smile))^\smile)$
proof −
have $C \sqsubseteq rang\ R \Longrightarrow C \sqsubseteq (dom\ (R^\smile))^\smile$
proof −
assume *cr*: $C \sqsubseteq rang\ R$
show $C \sqsubseteq (dom\ (R^\smile))^\smile$
proof −
have *RR*:$rang\ R \sqsubseteq (dom(R^\smile))^\smile$ **by** (*rule RDCOS-incl3*, *auto*)
from *cr RR* **show** *?thesis* **by**(*rule incl-trans*, *auto*)
qed
qed
moreover have $C \sqsubseteq (dom\ (R^\smile))^\smile \Longrightarrow C \sqsubseteq rang\ R$
proof −
assume *cr*: $C \sqsubseteq (dom\ (R^\smile))^\smile$
show $C \sqsubseteq rang\ R$
proof −
have *RR* :$(dom(R^\smile))^\smile \sqsubseteq rang\ R$ **by** (*rule RDCOS-incl1*, *auto*)
from *cr RR* **show** *?thesis* **by**(*rule incl-trans*, *auto*)
qed
qed
ultimately show *?thesis* **by** (*rule iffI*, *best+*)
qed
qed
qed

lemma (**in** *RDConvOrdSemi*) *RDCOS-dom*:
assumes [*intro*]:$R : a \leftrightarrow b$
shows $dom\ R = (rang(R^\smile))^\smile$
proof −
let $?R' = R^\smile$
have $(?R')^\smile = R$ **by** *auto*
moreover have $rang\ ?R' = (dom(?R'^\smile))^\smile$ **by** (*rule RDCOS-ran*, *auto*)
ultimately have *r1*: $rang(R^\smile) = (dom\ R)^\smile$ **by** *auto*
from *r1* **have** *r2*: $(rang(R^\smile))^\smile = ((dom\ R)^\smile)^\smile$ **by** *auto*
from *r2* **have** *r3*: $(rang(R^\smile))^\smile = dom\ R$
by (*rule-tac conv-idem* [*THEN subst*], *auto*)
from *r3* **show** $dom\ R = (rang(R^\smile))^\smile$ **by** (*rule-tac sym*, *auto*)
qed

The following two theorems are proved based on rang R = conv(dom(conv R)); dom R = conv(rang(conv R). They are used to support proving the property of restricted residuals in RestrResOSGC theory.

lemma (in *RDConvOrdSemi*) *RDCOS-domRan1*:
assumes [*intro*]: $Q : a \leftrightarrow b$
assumes [*intro*]: $C : b \leftrightarrow c$
assumes *CQ*: $dom\ C \sqsubseteq rang\ Q$
shows $rang\ (C^\smile) \sqsubseteq dom\ (Q^\smile)$
proof −
from *CQ* **have** *CQ1*:$dom\ C \sqsubseteq (dom(Q^\smile))^\smile$
by (*rule-tac RDCOS-ran* [*THEN subst*], *auto*)
from *CQ1* **have** *CQ2*:$(rang(C^\smile))^\smile \sqsubseteq (dom(Q^\smile))^\smile$
by(*rule-tac RDCOS-dom* [*THEN subst*], *auto*)
from *CQ2* **show** $rang(C^\smile) \sqsubseteq dom(Q^\smile)$
by (*rule-tac conv-incl* [*THEN iffD1*], *auto*)
qed

lemma (in *RDConvOrdSemi*) *RDCOS-domRan2*:
assumes [*intro*]: $Q : a \leftrightarrow b$
assumes [*intro*]: $C : b \leftrightarrow c$
assumes *CQ*: $rang\ (C^\smile) \sqsubseteq dom\ (Q^\smile)$
shows $dom\ C \sqsubseteq rang\ Q$
proof −
let $?Q' = Q^\smile$
have [*intro*]: $(?Q')^\smile = Q$ **by** *auto*
from *CQ* **have** $rang\ (C^\smile) \sqsubseteq dom\ (?Q')$ **by** *auto*
also have $\ldots = (rang(?Q'^\smile))^\smile$ **by**(*rule RDCOS-dom*, *auto*)
ultimately have *CQ1*: $rang\ (C^\smile) \sqsubseteq (rang(?Q'^\smile))^\smile$ **by** *auto*
moreover have $\ldots = (rang\ Q)^\smile$ **by** (*rule-tac subst*, *auto*)
ultimately have *CQ2*: $rang(C^\smile) \sqsubseteq (rang\ Q)^\smile$ **by**(*rule-tac subst*, *auto*)
have *CQ3*: $(rang(C^\smile))^\smile = dom\ C$
by (*rule-tac RDCOS-dom* [*THEN sym*], *auto*)
from *CQ3* **have** *CQ4*: $((rang(C^\smile))^\smile)^\smile = (dom\ C)^\smile$
by (*rule-tac conv-equality* [*THEN iffD2*], *auto*)
from *CQ4* **have** *CQ5*:$rang(C^\smile) = (dom\ C)^\smile$
by (*rule-tac conv-idem* [*THEN subst*], *auto*)
from *CQ2* **have** *CQ6*: $(dom\ C)^\smile \sqsubseteq (rang\ Q)^\smile$
by (*rule-tac CQ5* [*THEN subst*], *auto*)
from *CQ6* **show** *?thesis* **by** (*rule-tac conv-incl* [*THEN iffD1*], *auto*)

qed

end

B.29 OSGC with Restricted Residuals

theory *RestrResOSGC*
imports *RestrResSemi RDConvOrdSemi*
begin

B.29.1 Definitions

locale *RestrResOSGC* = *RestrResSemi RROSGC* + *RDConvOrdSemi RROSGC*

B.29.2 Theorems

Residual property (Proposition 4.1) in Furusawa-Kahl-1998 holds for restricted residuals

lemma (in *RestrResOSGC*) *restr-resOSGC-eq*:
assumes [*intro*]:$Q : a \leftrightarrow b$
assumes [*intro*]:$S : a \leftrightarrow c$
shows $(S \dashv Q) = (Q^\smile \vdash S^\smile)^\smile$
proof (*rule indirect-equality* [*THEN iffD2*])
show $\forall C.\ C \in b \leftrightarrow c \longrightarrow (C \sqsubseteq S \dashv Q) = (C \sqsubseteq (Q^\smile \vdash S^\smile)^\smile)$
proof (*intro strip*)
fix *C*
assume [*intro*]:$C : b \leftrightarrow c$
show $(C \sqsubseteq S \dashv Q) = (C \sqsubseteq (Q^\smile \vdash S^\smile)^\smile)$
proof −
have $C \sqsubseteq S \dashv Q \Longrightarrow C \sqsubseteq (Q^\smile \vdash S^\smile)^\smile$
proof −
assume *c*: $C \sqsubseteq S \dashv Q$
show $C \sqsubseteq (Q^\smile \vdash S^\smile)^\smile$
proof −
from *c* **have** [*intro*]: $(Q \odot C \sqsubseteq S)$ **by** (*rule-tac restr-rres-incl2*, *auto*)
from *c* **have** [*intro*]: $dom\ C \sqsubseteq rang\ Q$**by** (*rule-tac restr-rres-incl1*, *auto*)
have $(Q \odot C)^\smile \sqsubseteq S^\smile$ **by** (*auto*)
have $(C^\smile \odot Q^\smile) \sqsubseteq S^\smile$ **by** (*rule conv-cmp* [*THEN subst*],*auto*)

```
    also have rang (C⌣) ⊑ dom (Q⌣) by (rule RDCOS-domRan1, auto)
    ultimately have c2: C⌣ ⊑ (Q⌣ ⊢ S⌣) by (rule-tac restr-lres1, auto)
    from c2 have c3:(C⌣)⌣ ⊑ (Q⌣ ⊢ S⌣)⌣ by auto
    from c3 show C ⊑ (Q⌣ ⊢ S⌣)⌣ by (rule-tac conv-idem [THEN subst],auto)
   qed
  qed
  moreover have C ⊑ (Q⌣ ⊢ S⌣)⌣ ⟹ C ⊑ S ⊣ Q
  proof −
   assume C: C ⊑ (Q⌣ ⊢ S⌣)⌣
   show C ⊑ S ⊣ Q
   proof −
    from C have C1: (C⌣)⌣ ⊑ (Q⌣ ⊢ S⌣)⌣
              by (rule-tac conv-idem [THEN subst], auto)
    from C1 have C2: C⌣ ⊑ (Q⌣ ⊢ S⌣)
              by (rule-tac conv-incl [THEN iffD1], auto)
    from C2 have [intro]: (C⌣ ⊙ Q⌣) ⊑ S⌣ by auto
    from C2 have [intro]: rang(C⌣) ⊑ dom(Q⌣) by auto
    have (Q ⊙ C)⌣ ⊑ S⌣
         by (rule-tac conv-cmp [THEN sym, THEN subst],auto)
    also have [intro]: (Q ⊙ C) ⊑ S
              by (rule-tac conv-incl [THEN iffD1],auto)
    moreover have dom C ⊑ rang Q by (rule RDCOS-domRan2, auto)
    ultimately show C ⊑ S ⊣ Q by (rule-tac restr-rres1, auto)
   qed
  qed
  ultimately show ?thesis by(rule iffI, best+)
 qed
qed
next
 show S ⊣ Q ∈ b ↔ c by auto
next
 show (Q⌣ ⊢ S⌣)⌣ ∈ b ↔ c by auto
qed


end
```