Permissive Subsorted Partial Logic in CASL

Maura Cerioli[1], Anne Haxthausen[2], Bernd Krieg-Brückner[3], Till Mossakowski[3]

[1] DISI, Via Dodecaneso 35, I-16146 Genova
[2] Dept. of Information Technology, Techn. University of Denmark, DK-2800 Lyngby
[3] BISS, Universität Bremen, P.O. Box 330440, D-28334 Bremen

Abstract. This paper presents a permissive subsorted partial logic used in the CoFI Algebraic Specification Language. In contrast to other order-sorted logics, subsorting is not modeled by set inclusions, but by injective embeddings allowing for more general models in which subtypes can have different data type representations. Furthermore, there are no restrictions like monotonicity, regularity or local filtration on signatures at all. Instead, the use of overloaded functions and predicates in formulae is required to be sufficiently disambiguated, such that all parses have the same semantics. An overload resolution algorithm is sketched.

1 Introduction

During the past decades a large number of algebraic specification languages have been developed. The presence of so many similar specification languages with no common framework hinders the dissemination and application of research results in algebraic specification. In particular, it makes it difficult to produce educational material, to re-use tools and to get algebraic methods adopted in industry. Therefore, in 1995, an initiative, CoFI[4], to design *a Common Framework for Algebraic Specification and Development* was started [Mos97b]. The goal of CoFI is to get a common agreement in the algebraic specification community about basic concepts, and to provide a family of specification languages at different levels, a development methodology and tool support. The family of specification languages will comprise a central, common language, called CASL[5], various restrictions of CASL, and various extensions of CASL (e.g. with facilities for particular programming paradigms). A design proposal [LD97] of CASL has already been completed by representatives of most algebraic specification language groups. Note that the concrete syntax used in this paper is a proposal and no decision about concrete syntax has yet been taken.

CASL provides constructs for writing structured requirement and design specifications as well as architectural specifications. Basic CASL specifications consist of declarations and axioms representing theories of a first-order logic in which predicates, total as well as partial functions, and subsorts are allowed. Predicate and function symbols may be overloaded.

[4] CoFI is an acronym for *Common Framework Initiative* and is pronounced like 'coffee'.
[5] CASL is an acronym for *CoFI Algebraic Specification Language* and is pronounced like 'castle'.

In this paper we will present the subsorting approach used in CASL. The main novelties of our approach compared with the usual order-sorted approach [GM92] are:

- Functions may be partial.
- Predicates are allowed as in EqLog [GM86].
- Projection functions from supersorts to subsorts are naturally defined as partial functions, instead of total retract functions [GM92].
- Subsorting is not modeled by set inclusions, but by injective embeddings allowing for more general models in which subtypes may have different data type representations.
- There are no requirements like monotonicity, regularity or local filtration imposed on signatures.

We dropped the requirements like monotonicity, regularity or local filtration in order to avoid problems with modularity (as described in [HN96] and [Mos96, Mos97a]) and to allow overloading of constants. Instead, the use of overloaded functions and predicates in formulae is required to be sufficiently disambiguated, such that all parses have the same semantics. This means a more complicated definition of well-formed formulae, and requires a parsing algorithm which is more complex than the simple bottom-up least sort parsing algorithm by Goguen and Meseguer [GM92]. However, in cases where the signatures satisfy the mentioned requirements, the complexity of the two algorithms is the same.

The idea of dropping regularity (but not local filtration) has been proposed by Goguen and Diaconescu [GD94], and the idea of dropping local filtration (but not regularity) has been proposed by Haxthausen [Hax97] for a subsorting approach with implicit, possibly non-injective coercions. However, it is new that both requirements are dropped at the same time, and it is new that a parsing (overload resolution) algorithm is sketched.

First, in section 2, some well-known definitions from many-sorted partial logic are given, and, in section 3, the underlying subsorted partial logic of CASL is defined in terms of the many-sorted partial logic. Then, in section 4, the CASL subsorting language is presented, and in section 5, an overload resolution algorithm and the possibility of re-using existing theorem provers are briefly described. Finally, in section 6 a discussion is given. A longer version of this paper with all technical details will appear in a forthcoming report.

2 Many-sorted Partial Logic

This section defines the notions of signatures, models, and sentences of many-sorted partial first-order logic. For a full treatment of the topic, see e.g. [CMR98].

2.1 Signatures

Definition 2.1 A *many-sorted signature* $\Sigma = (S, TF, PF, P)$ consists of:

- a set S of *sorts*
- two $S^*{\times}S$-sorted families $TF = (TF_{w,s})_{w \in S^*, s \in S}$ and $PF = (PF_{w,s})_{w \in S^*, s \in S}$ of *total function symbols* and *partial function symbols*, respectively, such that $TF_{w,s} \cap PF_{w,s} = \{\}$, for each $(w, s) \in S^*{\times}S$ (constants are treated as functions with no arguments)
- a family $P = (P_w)_{w \in S^*}$ of *predicate symbols*

For a function symbol $f \in TF_{w,s} \cup PF_{w,s}$, we call (w, s) its *profile*. For predicate symbols $p \in P_w$, we call w its *profile*.

Signature morphisms are defined as usual, with the speciality that a partial function symbol may be mapped to a total function symbol (but not vice versa).

Note that function and predicate symbols may be overloaded, occurring in more than one of the above sets. To ensure that there is no ambiguity in sentences, however, symbols are always qualified by profiles when used. In the CASL language considered in section 4, such qualifications may be omitted when these are unambiguously determined by the context.

2.2 Sentences

Let a many-sorted signature $\Sigma = (S, TF, PF, P)$ and an S-sorted family of variables $X = (X_s)_{s \in S}$ be given.

Definition 2.2 The sets $T_\Sigma(X)_s$ of *many-sorted Σ-terms* of sort s, $s \in S$, with variables in X are the least sets satisfying the following rules:

1. $x \in T_\Sigma(X)_s$, if $x \in X_s$
2. $f_{(w,s)}(t_1, \ldots, t_n) \in T_\Sigma(X)_s$, if $t_i \in T_\Sigma(X)_{s_i}, f \in TF_{w,s} \cup PF_{w,s}, w = s_1 \ldots s_n$

Note that each term has a unique sort.

A many-sorted atomic Σ-formula with variables in X comprises (1) applications of qualified predicate symbols to terms of appropriate sorts, (2) existential equations between terms of the same sort, (3) strong equations between terms of the same sort, and (4) assertions about definedness of terms.

Definition 2.3 The set $AF_\Sigma(X)$ of *many-sorted atomic Σ-formulae* with variables in X is the least set satisfying the following rules:

1. $p_w(t_1, \ldots, t_n) \in AF_\Sigma(X)$, if $t_i \in T_\Sigma(X)_{s_i}, p \in P_w, w = s_1 \ldots s_n \in S^*$
2. $t \stackrel{!}{=} t' \in AF_\Sigma(X)$, if $t, t' \in T_\Sigma(X)_s, s \in S$
3. $t = t' \in AF_\Sigma(X)$, if $t, t' \in T_\Sigma(X)_s, s \in S$
4. $\partial\, t \in AF_\Sigma(X)$, if $t \in T_\Sigma(X)_s, s \in S$

Definition 2.4 *Many-sorted Σ-sentences* are the usual closed many-sorted first-order logic formulae, built using quantification (over sorted variables), logical connectives and atomic Σ-formulae.

2.3 Models

Definition 2.5 Given a many-sorted signature $\Sigma = (S, TF, PF, P)$, a *many-sorted Σ-model M* consists of:

- a carrier set s^M for each sort $s \in S$
- a partial function f^M from w^M to s^M for each function symbol $f \in TF_{w,s} \cup PF_{w,s}$, the function being total if $f \in TF_{w,s}$
- a predicate $p^M \subseteq w^M$ for each predicate symbol $p \in P_w$.

Notation. We write w^M for the Cartesian product $s_1^M \times ... \times s_n^M$, when $w = s_1 ... s_n$.

Definition 2.6 A *many-sorted Σ-homomorphism* $h : M \to N$ consists of a family of functions $(h_s : s^M \to s^N)_{s \in S}$ such that

- for all $f \in TF_{w,s} \cup PF_{w,s}$ and $(a_1, \ldots, a_n) \in w^M$ with $f^M(a_1, \ldots, a_n)$ defined,

$$h_s(f^M(a_1, \ldots, a_n)) = f^N((h_{s_1}(a_1), \ldots, h_{s_n}(a_n))$$

- for all $p \in P_w$ and $(a_1, \ldots, a_n) \in w^M$

$$(a_1, \ldots, a_n) \in p^M \text{ implies } (h_{s_1}(a_1), \ldots, h_{s_n}(a_n)) \in p^N$$

2.4 Satisfaction Relation

The satisfaction of a Σ-sentence by a Σ-model M is defined as usual in terms of the satisfaction of its constituent atomic formulae w.r.t. assignments of values to all the variables that occur in them, the value assigned to variables of sort s being in s^M. Variable assignments are total, but the value of a term w.r.t. a variable assignment may be undefined, due to the application of a partial function during the evaluation of the term. Note, however, that the satisfaction of sentences is two-valued.

The application of a predicate symbol p to a sequence of argument terms holds in M iff the values of all the terms are defined and give a tuple belonging to p^M. A definedness assertion concerning a term holds iff the value of the term is defined. An existential equation holds iff the values of both terms are defined and identical, whereas a strong equation holds also when the values of both terms are undefined; thus both notions of equation coincide for defined terms. The value of an occurrence of a variable in a term is that provided by the given variable assignment. The value of the application of a function symbol f to a sequence of argument terms is defined only if the values of all the argument terms are defined and give a tuple in the domain of definedness of f^M, and then it is the associated result value.

3 Subsorted Partial Logic

This section defines the notions of signatures, models, and sentences of subsorted partial first-order logic, leading to an institution $SubPFOL$ which is used in the semantics of CASL specifications.

The definitions are based on the many-sorted partial first-order logic given in section 2. Subsorted models, homomorphisms and sentences are defined to be certain many-sorted models, homomorphisms and sentences. This has the important consequence, that results like the existence of initial models carry directly over from the many-sorted case.

3.1 Signatures

The notion of subsorted signatures extends the notion of order-sorted signatures as given by Goguen and Meseguer [GM92], by allowing not only total function symbols, but also partial function symbols and predicate symbols:

Definition 3.1 A *subsorted signature* $\Sigma = (S, TF, PF, P, \leq_S)$ consists of a many-sorted signature (S, TF, PF, P) together with a reflexive transitive *subsort relation* $\leq_S$ on the set S of sorts.

Notation. The relation $\leq_S$ extends pointwise to sequences of sorts. We drop the subscript S when obvious from the context.

Note that the signatures are not required to be monotonic as in [GM92]. This decision was taken in order to allow constants to be overloaded and avoid problems with modularity. This is further discussed in section 6. However, whenever two qualified symbols are in one of the overloading relations defined below, their semantics are still required to be consistent, cf. the overloading axioms in section 3.3.

Definition 3.2 Let a subsorted signature $\Sigma = (S, TF, PF, P, \leq_S)$ be given, and let $f \in (TF_{w_1,s_1} \cup PF_{w_1,s_1}) \cap (TF_{w_2,s_2} \cup PF_{w_2,s_2})$ and $p \in P_{w_1} \cap P_{w_2}$.

Two qualified function symbols $f_{(w_1,s_1)}$ and $f_{(w_2,s_2)}$ are in *overloading relation* (written $f_{(w_1,s_1)} \sim_F f_{(w_2,s_2)}$) iff there exists a $w \in S^*$ and $s \in S$ such that $w \leq_S w_1, w_2$ and $s_1, s_2 \leq_S s$.

Two qualified predicate symbols p_{w_1} and p_{w_2} are in overloading relation (written $p_{w_1} \sim_P p_{w_2}$) iff there exists a $w \in S^*$ such that $w \leq_S w_1, w_2$.

We say that two profiles of a symbol are in overloading relation if the corresponding qualified symbols are in overloading relation.

Note that two profiles of an overloaded constant declared with two different sorts are in the overloading relation iff the two sorts have a common supersort.

Definition 3.3 A *signature morphism* $\sigma : \Sigma \to \Sigma'$ is a many-sorted signature morphism that preserves the subsort relation and the overloading relations.

For modularity aspects, it is important that signatures can be combined. This is guaranteed by the following proposition.

Proposition 3.4 The category of subsorted CASL signatures and signature morphisms is cocomplete.

A proof of this is given in [Mos97a].

Definition 3.5 With each subsorted signature $\Sigma = (S, TF, PF, P, \leq_S)$ we associate a many-sorted signature $\Sigma^{\#}$, which is the extension of the underlying many-sorted signature (S, TF, PF, P) with

- a total *injection* function symbol $\mathtt{inj}_{(s,s')}$, for each pair of sorts $s \leq_S s'$
- a partial *projection* function symbol $\mathtt{pr}_{(s',s)}$, for each pair of sorts $s \leq_S s'$
- a unary *membership* predicate symbol $\in^s_{s'}$, for each pair of sorts $s \leq_S s'$

We assume that the symbols used for injection, projection and membership are not used otherwise in Σ.

3.2 Sentences

Definition 3.6 *Subsorted Σ-sentences* are ordinary many-sorted $\Sigma^{\#}$-sentences.

Note that in the sentences, injections from subsorts to supersorts must be explicit. In the CASL language considered in section 4 these are left implicit and must unambiguously be determined by the context.

Since terms are fully disambiguated, they have a unique sort. This implies that, at this institution level, there is no need at all to require signatures to be regular and locally filtered. However, for the parsing of CASL terms it makes a difference whether signatures satisfy these requirements or not.

3.3 Models

Definition 3.7 Subsorted Σ-models are ordinary many-sorted $\Sigma^{\#}$-models satisfying the following set of axioms $J(\Sigma)$ (where the variables are all universally quantified):

$\mathtt{inj}_{(s,s)}(x) \stackrel{!}{=} x$ [identity]
$\mathtt{inj}_{(s,s')}(x) \stackrel{!}{=} \mathtt{inj}_{(s,s')}(y) \Rightarrow x \stackrel{!}{=} y$ for $s \leq_S s'$ [embedding-injectivity]
$\mathtt{inj}_{(s',s'')}(\mathtt{inj}_{(s,s')}(x)) \stackrel{!}{=} \mathtt{inj}_{(s,s'')}(x)$ for $s \leq_S s' \leq_S s''$ [transitivity]
$\mathtt{pr}_{(s',s)}(\mathtt{inj}_{(s,s')}(x)) \stackrel{!}{=} x$ for $s \leq_S s'$ [projection]
$\mathtt{pr}_{(s',s)}(x) \stackrel{!}{=} \mathtt{pr}_{(s',s)}(y) \Rightarrow x \stackrel{!}{=} y$ for $s \leq_S s'$ [projection-injectivity]
$\in^s_{s'}(x) \Leftrightarrow \partial\ \mathtt{pr}_{(s',s)}(x)$ for $s \leq_S s'$ [membership]
$\mathtt{inj}_{(s',s)}(f_{(w',s')}(\mathtt{inj}_{(s_1,s'_1)}(x_1), \ldots, \mathtt{inj}_{(s_n,s'_n)}(x_n))) =$
  $\mathtt{inj}_{(s'',s)}(f_{(w'',s'')}(\mathtt{inj}_{(s_1,s''_1)}(x_1), \ldots, \mathtt{inj}_{(s_n,s''_n)}(x_n)))$
  for $w = s_1 \times \ldots \times s_n$, $w' = s'_1 \times \ldots \times s'_n$, $w'' = s''_1 \times \ldots \times s''_n$,
  $w \leq_S w', w'', s', s'' \leq_S s$, $f \in (TF_{w',s'} \cup PF_{w',s'}) \cap (TF_{w'',s''} \cup PF_{w'',s''})$
  [function-overloading]
$p_{w'}(\mathtt{inj}_{(s_1,s'_1)}(x_1), \ldots, \mathtt{inj}_{(s_n,s'_n)}(x_n)) \Leftrightarrow p_{w''}(\mathtt{inj}_{(s_1,s''_1)}(x_1), \ldots, \mathtt{inj}_{(s_n,s''_n)}(x_n))$
  for $w = s_1 \times \ldots \times s_n$, $w' = s'_1 \times \ldots \times s'_n$, $w'' = s''_1 \times \ldots \times s''_n$, $w \leq_S w', w''$,
  $p \in P_{w'} \cap P_{w''}$ [predicate-overloading]

Definition 3.8 Σ-homomorphisms are $\Sigma^{\#}$-homomorphisms.

The following result directly follows from the corresponding result for partial first-order logic (PFOL) stated in [Rei87]:

Proposition 3.9 Consider the restriction of $SubPFOL$ to universally quantified positive conditional axioms. In this restriction, all theories have initial models and along all theory morphisms there exist free extensions.

3.4 Injections Versus Inclusions

In CASL, subsorting is modeled by (implicit) injections (i.e. injective embedding functions between the corresponding carriers), and not by set inclusions as in usual order-sorted algebra [GM92]. This extra generality means that it is possible to specify a sort s_1 to be a subsort of another sort s_2 without requiring that their implementations use the same data representation (i.e. there are models in which the carrier set of s_1 is not a subset of the carrier set of s_2), as it is often the case in imperative programming languages (for efficiency reasons). The injection functions are used to convert the data representation when passing from a subtype to a supertype. In cases where it is desirable not to have the cost of converting data representation, a model with subset inclusions should be used.

For locally upwards filtered signatures, it is possible to pass from a model with injections to an isomorphic model with true subset inclusions, cf. Goguen and Meseguer [GM92]. We could have obtained the same result for non-filtered signatures, if we in Definition 3.7 had added axioms expressing that different compositions of injections and projections to the same term should give the same result, when well-defined. However, as illustrated by the following example, we want to allow more general models for which this is not always the case.

Example 3.10 Consider the following specification

> **sorts** $Digit, Byte, Nat, Character$
> $\quad Digit, Byte < Nat$
> $\quad Digit, Byte < Character$

An intended model A would interpret $Digit$ with $Digit^A = \{0, ..., 9\}$ and $Byte$ with $Byte^A = \{0, ..., 255\}$, both obviously included in the natural numbers $Nat^A = \mathbb{N}$. Then, $Character$ is interpreted with $Character^A = \{0, ..., 255\}$. Since characters are already represented by bytes, the subsort embedding of $Byte$ to $Character$ is interpreted with the identity (if we had 16bit-characters, it would be a true inclusion). However, the interpretation of the subsort embedding of $Digit$ in $Character$ maps a digit to its ASCII code (we could also choose some other code here).

A digit can be mapped to a byte in two ways: either inject it into $Character$, and project it onto $Byte$, or inject it into $Natural$ and project it onto $Byte$; the results are different. This is not possible in a model where subsort embeddings are set inclusions, whichs shows that A is not isomorphic to any such model.

3.5 Satisfaction Relation

Since subsorted Σ-models and Σ-sentences are just certain many-sorted $\Sigma^{\#}$-models and $\Sigma^{\#}$-sentences, the notion of satisfaction for the subsorted case follows directly from the notion of satisfaction for the many-sorted case.

4 CASL Subsorting Language

In this section we present the CASL subsorting language. We use an ad-hoc concrete syntax, as the concrete syntax for CASL has not yet been decided, and we give the semantics in terms of signatures, sentences and models of the underlying institution of subsorted partial logic defined in section 3. A description of the full CASL language is given in the design proposal [LD97].

4.1 Basic Specifications

A *basic* CASL *specification* consists of a set of declarations and a set of axioms. In section 4.7 some examples of CASL specifications are given.

A specification is *well-formed* if the declarations (see section 4.2) determine a subsorted signature Σ and a possible empty set of subsorted Σ-sentences $E1$. Furthermore, the set of axioms (see section 4.3) must determine a set of subsorted Σ-sentences $E2$. The *semantics* of the specification is then the class of subsorted Σ-models satisfying $E1$ and $E2$.

CASL also provides facilities for writing structured specifications and architectural specifications, but these are not covered in this paper as they are institution independent and hence independent of the subsorting in basic specifications.

4.2 Declarations

A CASL declaration can be

- a declaration of a sort, s,
- a declaration of a total function symbol, $f : w \rightarrow s$ or $f : s$,
- a declaration of a partial function symbol, $f : w \stackrel{?}{\rightarrow} s$ or $f :^{?} s$,
- a declaration of a predicate symbol, $p : \wp(w)$, or,
- an assertion of a subsort relationship, $s_1 < s_2$

and contributes in the obvious way to the elements S, TF, PF, P and $\leq_S$ of a subsorted signature $(S, TF, PF, P, \leq_S)$.

Furthermore, a declaration can be a *predicative sort definition* of the form:

$$\textbf{sorts}\ s = \{x : s' \bullet\ \phi[x]\}$$

where s' is an existing sort and $\phi[x]$ is a formula. It declares the new sort s to be a subsort of the sort s', asserting that the values of the subsort s are precisely the projections of those values x from the supersort s' for which the given formula $\phi[x]$ holds. More formally, the predicative sort definition is a shorthand for the following declarations and axiom:

sorts s; $s < s'$
$\forall x : s' \bullet \phi[x] \Leftrightarrow x \in s$

4.3 Axioms

CASL *axioms*[6] are CASL *formulae* built from atomic CASL formulae using quantification (over sorted variables) and the usual logical connectives of first-order logic.

An *atomic* CASL *formula* can have the following forms:

$p(t_1 \dots t_n)$	(application of a predicate p to terms $t_1 \dots t_n$)
$(p : \wp(w))(t_1 \dots t_n)$	(application of a qualified predicate to terms $t_1 \dots t_n$)
$t \stackrel{!}{=} t'$	(existential equation between terms t and t')
$t = t'$	(strong equation between terms t and t')
$t \in s$	(the assertion that a term t is within a subsort s)
$\partial\, t$	(the assertion that a term t is defined)

A CASL *term* can have one of the following forms:

x	(variable)
f	(constant)
$f(t_1, \dots, t_n)$	(application of a function f to terms $t_1 \dots t_n$)
$(f : w \rightarrow s)(t_1, \dots, t_n)$	(application of a qualified total function to terms)
$(f : w \stackrel{?}{\rightarrow} s)(t_1, \dots, t_n)$	(application of a qualified partial function to terms)
$t : s$	(sort disambiguation: force term t to have sort s)
$t \downarrow s$	(casting a term t to a subsort s)

Sort disambiguations and the qualification of function and predicate symbols with their types are used to avoid unintended interpretations of overloaded function and predicate symbols.

In section 4.4, we introduce the rules for *well-sortedness* of CASL formulae. These rules allow implicit embeddings in term formation (i.e. terms of a subsort can be used whenever terms of a supersort are expected). Well-sorted CASL formulae can be *parsed/expanded* to subsorted sentences of the underlying institution $SubPFOL$ of subsorted partial logic by qualifying function and predicate symbols with their profiles, inserting implicit injection functions, replacing casts with corresponding applications of projection functions and removing sort disambiguations. Due to the overloading of function and predicate symbols and implicit injections, it turns out that a CASL formula may have several possible expansions. The expansion relation is defined in section 4.5. Finally, in section 4.6, we define a CASL formula to be *well-formed*, if it is well-sorted and can be sufficiently disambiguated (i.e. all its expansions have the same semantics).

A well-formed CASL formula then *determines* all the subsorted sentences it can be expanded into.

[6] In the full CASL language, it is also possible to state sort generation constraints.

4.4 Well-sorted Terms and Formulae

The well-sortedness of a CASL formula is defined wrt. a subsorted signature $\Sigma = (S, TF, PF, P, \leq_S)$ (determined from the declarations in the specification in which the formula occur).

Definition 4.1 A CASL formula is *well-sorted* wrt. Σ, if each of its constituent quantifications uses S-sorted variables, and each of its constituent atomic formulae φ is well-sorted wrt. Σ and X, where X is an S-sorted family of variables determined in the obvious way from those quantifications which enclose φ.

Before defining which atomic formulae are well-sorted, we first define the sets of well-sorted terms.

Definition 4.2 The set $W_\Sigma(X)_s$ of *well-sorted* CASL terms of sort s wrt. Σ and X is the least set satisfying the following rules:

1. $x \in W_\Sigma(X)_s$, if $x \in X_s$
2. $f \in W_\Sigma(X)_s$, if $f \in TF_{w,s} \cup PF_{w,s}$, $w = \lambda$
3. $f(t_1, \ldots, t_n) \in W_\Sigma(X)_s$, if $t_i \in W_\Sigma(X)_{s_i}$, and $f \in TF_{w,s} \cup PF_{w,s}$, $w = s_1 \ldots s_n$
4. $(f : w \to s)(t_1, \ldots, t_n) \in W_\Sigma(X)_s$, if $t_i \in W_\Sigma(X)_{s_i}$, $f \in TF_{w,s}$, $w = s_1 \ldots s_n$
5. $(f : w \stackrel{?}{\to} s)(t_1, \ldots, t_n) \in W_\Sigma(X)_s$, if $t_i \in W_\Sigma(X)_{s_i}$, $f \in PF_{w,s}$, $w = s_1 \ldots s_n$
6. $t : s \in W_\Sigma(X)_s$, if $t \in W_\Sigma(X)_s$
7. $t \downarrow s' \in W_\Sigma(X)_{s'}$, if $t \in W_\Sigma(X)_s$ and $s' \leq_S s$
8. $t \in W_\Sigma(X)_{s'}$, if $t \in W_\Sigma(X)_s$ and $s \leq_S s'$

The set of *well-sorted* CASL terms, wrt. Σ and X is the union of the sorted sets defined above: $W_\Sigma(X) = \cup_{s \in S} W_\Sigma(X)_s$.

Note that, due to function overloading and implicit injections, a CASL term may have several sorts.

Definition 4.3 The set $A_\Sigma(X)$ of *well-sorted* atomic CASL formulae wrt. Σ and X is the least set satisfying the following rules:

1. $p(t_1, \ldots, t_n) \in A_\Sigma(X)$, if $t_i \in W_\Sigma(X)_{s_i}$, $p \in P_{s_1 \ldots s_n}$
2. $(p : \wp(w))(t_1, \ldots, t_n) \in A_\Sigma(X)$, if $t_i \in W_\Sigma(X)_{s_i}$, $p \in P_w$, $w = s_1 \ldots s_n$
3. $t \stackrel{!}{=} t' \in A_\Sigma(X)$, if $t, t' \in W_\Sigma(X)_s$
4. $t = t' \in A_\Sigma(X)$, if $t, t' \in W_\Sigma(X)_s$
5. $\partial\, t \in A_\Sigma(X)$, if $t \in W_\Sigma(X)$
6. $t \in s \in A_\Sigma(X)$, if $t \in W_\Sigma(X)_{s'}$, $s \in S$, $s \leq_S s'$

4.5 Expansion of Terms and Formulae

In this section we define the *expansion* relation, $\leadsto$, between well-sorted terms or formulae of the CASL language and terms or formulae of the underlying institution $SubPFOL$ of subsorted partial logic.

The expansion relation for formulae and terms is defined wrt. a subsorted signature $\Sigma = (S, TF, PF, P, \leq_S)$, and for atomic formulae and terms also wrt. an S-sorted set of variables X.

Definition 4.4 The expansion relation, $\rightsquigarrow \subseteq W_\Sigma(X) \times T_{\Sigma\#}(X)$, from well-sorted CASL terms into subsorted Σ-terms is inductively defined by the following rules in which $t \rightsquigarrow_s t'$ is a shorthand for $t \rightsquigarrow t' \wedge t' \in T_{\Sigma\#}(X)_s$:

$$1.\ \frac{}{x \rightsquigarrow x}\ x \in X \qquad 2.\ \frac{}{f \rightsquigarrow f_{(w,s)}()}\ f \in TF_{w,s} \cup PF_{w,s}\ , w = \lambda$$

$$3.\ \frac{t_i \rightsquigarrow_{s_i} t'_i}{f(t_1,\ldots,t_n) \rightsquigarrow f_{(w,s)}(t'_1,\ldots,t'_n)}\ f \in TF_{w,s} \cup PF_{w,s}\ , w = s_1 \ldots s_n$$

$$4.\ \frac{t_i \rightsquigarrow_{s_i} t'_i}{(f : w \to s)(t_1,\ldots,t_n) \rightsquigarrow f_{(w,s)}(t'_1,\ldots,t'_n)}\ f \in TF_{w,s}\ , w = s_1 \ldots s_n$$

$$5.\ \frac{t_i \rightsquigarrow_{s_i} t'_i}{(f : w \xrightarrow{?} s)(t_1,\ldots,t_n) \rightsquigarrow f_{(w,s)}(t'_1,\ldots,t'_n)}\ f \in PF_{w,s}\ , w = s_1 \ldots s_n$$

$$6.\ \frac{t \rightsquigarrow_s t'}{t : s \rightsquigarrow t'} \qquad 7.\ \frac{t \rightsquigarrow_{s'} t'}{t \downarrow s \rightsquigarrow \mathtt{pr}_{(s',s)}(t')}\ s \leq_S s'$$

$$8.\ \frac{t \rightsquigarrow_s t'}{t \rightsquigarrow \mathtt{inj}_{(s,s')}(t')}\ s \leq_S s'$$

Definition 4.5 The expansion relation, $\rightsquigarrow \subseteq A_\Sigma(X) \times AF_\Sigma(X)$, from well-sorted atomic CASL formulae into atomic subsorted Σ-formulae is inductively defined by the following rules:

$$1.\ \frac{t_i \rightsquigarrow_{s_i} t'_i}{p(t_1,\ldots,t_n) \rightsquigarrow p_w(t'_1,\ldots,t'_n)}\ p \in P_w\ , w = s_1 \ldots s_n$$

$$2.\ \frac{t_i \rightsquigarrow_{s_i} t'_i}{(p : \wp(w))(t_1,\ldots,t_n) \rightsquigarrow p_w(t'_1,\ldots,t'_n)}\ p \in P_w\ , w = s_1 \ldots s_n$$

$$3.\ \frac{t_1 \rightsquigarrow_s t'_1 \wedge t_2 \rightsquigarrow_s t'_2}{t_1 \stackrel{!}{=} t_2 \rightsquigarrow t'_1 \stackrel{!}{=} t'_2} \qquad 4.\ \frac{t_1 \rightsquigarrow_s t'_1 \wedge t_2 \rightsquigarrow_s t'_2}{t_1 = t_2 \rightsquigarrow t'_1 = t'_2}$$

$$5.\ \frac{t \rightsquigarrow_s t'}{\partial\, t \rightsquigarrow \partial\, t'} \qquad 6.\ \frac{t \rightsquigarrow_{s'} t'}{t \in s \rightsquigarrow \in^s_{s'}(t')}\ s \leq_S s'$$

Well-sorted CASL formulae expand into subsorted Σ-formulae in the obvious way by expanding their constituent atomic CASL formulae.

4.6 Equivalent Expansions and Well-formedness of Formulae

In this section, first we define an equivalence relation on subsorted formulae, and then define what it means for a CASL formula to be *well-formed.*

Definition 4.6 Two subsorted Σ-formulae ψ and ψ' are *equivalent* if the formulae are identical up to replacement of equivalent atomic formulae.

Definition 4.7 Two subsorted atomic Σ-formulae ψ and ψ' are *equivalent* (written $\psi \sim \psi'$), if they are in the least equivalence $\sim$ on $AF_\Sigma(X)$ generated by the following rules:

1. $\psi \sim \psi'$, if ψ and ψ' are identical up to replacement of equivalent terms
2. $p_{w'}(\mathtt{inj}_{(s_1,s_1')}(t_1),\ldots,\mathtt{inj}_{(s_n,s_n')}(t_n)) \sim p_{w''}(\mathtt{inj}_{(s_1,s_1'')}(t_1),\ldots,\mathtt{inj}_{(s_n,s_n'')}(t_n))$ for $w = s_1\ldots s_n, w' = s_1'\ldots s_n', w'' = s_1''\ldots s_n'', w \leq_S w', w''$, $p \in P_{w'} \cap P_{w''}$
3. $\mathtt{inj}_{(s,s')}(t_1) = \mathtt{inj}_{(s,s')}(t_2) \sim t_1 = t_2$ for $s \leq_S s'$
4. $\mathtt{inj}_{(s,s')}(t_1) \stackrel{!}{=} \mathtt{inj}_{(s,s')}(t_2) \sim t_1 \stackrel{!}{=} t_2$ for $s \leq_S s'$
5. $\partial\ \mathtt{inj}_{(s,s')}(t) \sim \partial\ \mathtt{inj}_{(s,s'')}(t)$ for $s \leq_S s', s''$
6. $\dfrac{\mathtt{pr}_{(s',s)}(t') \sim \mathtt{pr}_{(s'',s)}(t'')}{\in^s_{s'}(t') \sim \in^s_{s''}(t'')}\ s \leq_S s', s''$

Definition 4.8 Two subsorted Σ-terms t and t' are *equivalent* (written $t \sim t'$), if they are in the least congruence $\sim$ (wrt. the function symbols in $\Sigma^\#$) on $T_{\Sigma^\#}(X)$ generated by the following axioms:

1. $\mathtt{inj}_{(s,s)}(t) \sim t$
2. $\mathtt{inj}_{(s',s'')}(\mathtt{inj}_{(s,s')}(t)) \sim \mathtt{inj}_{(s,s'')}(t)$ for $s \leq_S s' \leq_S s''$
3. $\mathtt{pr}_{(s',s)}(\mathtt{inj}_{(s,s')}(t)) \sim t$ for $s \leq_S s'$
4. $\mathtt{inj}_{(s',s)}(f_{(w',s')}(\mathtt{inj}_{(s_1,s_1')}(t_1),\ldots,\mathtt{inj}_{(s_n,s_n')}(t_n))) \sim$
   $\mathtt{inj}_{(s'',s)}(f_{(w'',s'')}(\mathtt{inj}_{(s_1,s_1'')}(t_1),\ldots,\mathtt{inj}_{(s_n,s_n'')}(t_n)))$
   for $w = s_1\ldots s_n, w' = s_1'\ldots s_n', w'' = s_1''\ldots s_n'', w \leq_S w', w'', s', s'' \leq_S s$ and $f \in (TF_{w',s'} \cup PF_{w',s'}) \cap (TF_{w'',s''} \cup PF_{w'',s''})$

The following theorem states that two equivalent expansions are satisfied by the same models.

Theorem 4.9 Let ψ_1 and ψ_2 be subsorted Σ-sentences, which are expansions of a CASL formula φ, and let m be a subsorted Σ-model. If ψ_1 and ψ_2 are equivalent, then m satisfies ψ_1 iff m satisfies ψ_2.

Definition 4.10 A CASL formula is *well-formed* wrt. Σ iff it is well-sorted wrt. Σ and all its expansions are equivalent.

From definition 4.10 and theorem 4.9 it follows that all the expansions of a well-formed CASL formula are satisfied by the same models. Therefore, we can say that a well-formed formula *determines* any of its expansions.

4.7 Examples of CASL Specifications

Example 1
This example demonstrates the interplay between subsorting and partiality:

NAT =
sorts
nat, pos
$pos < nat$
$even = \{n : nat \bullet iseven(n)\}$
ops
$0 : nat$
$succ : nat \rightarrow pos$
$pred : nat \xrightarrow{?} nat$
$pred : pos \rightarrow nat$
$iseven : \wp(nat)$
$\forall n : nat \bullet$
$\neg \partial\ pred(0) \wedge$
$pred(succ(n)) = n \wedge$
$iseven(0) \wedge$
$(iseven(succ(n)) \Leftrightarrow \neg\, iseven(n))$

Now the formula $\partial\ pred(pred(succ(succ(0)))$ can be parsed with the first *pred* being partial, while the second *pred* can have any of the two profiles. But since the two profiles are in the overloading relation, this ambiguity is harmless. If we omit the partial *pred* from the signature, the formula becomes ill-formed, since no retracts are inserted automatically. We then have to insert an explicit projection:

$$\partial\ pred(\ pred(succ(succ(0))) \downarrow pos\ )$$

to make the formula well-formed.

Example 2
This example (inspired by [GD94]) shows that in CASL it is possible to have multiple representations (for example, points are represented by both cartesian and polar coordinates), to switch between representations without invoking explicit conversion functions and even to have overloaded functions on different representations.

$\vdots$
sorts
$Point, Cart, Polar$
$Cart, Polar < Point$
ops
$atan : Float \rightarrow Angle$
$origin : Cart$
$coord : Float \times Float \rightarrow Cart$
$origin : Polar$
$coord : PosFloat \times Angle \rightarrow Polar$
$\vdots$
$\forall x : PosFloat;\ y : Float\ \bullet coord(x, y) = coord(sqrt(x^2 + y^2), atan(y/x))$

Above, it is assumed that $PosFloat$ is a subsort of $Float$, and, $sqrt$, $\_^2$ and $+$ have been declared with sufficient types such that $sqrt(x^2 + y^2)$ has sort $PosFloat$. Furthermore, it is assumed that $Angle$ and $Float$ do not have a common subsort. This has the effect that the two profiles of $coord$ are not in the overloading relation and thus can coexist as different functions. Otherwise, one would have to use two different names, say $coordPolar$ and $coordCart$.

5 Tools

The design of CASL has been finished only recently, so it is clear that tools have to be developed yet. In this section, we briefly describe an overload resolution algorithm and the possibility to re-use existing theorem provers. Both together comprise a minimal set of tools which make the "in-the-small" part of CASL amenable to machine support. This will also be the basis for a shallow encoding of the "in-the-small" part of CASL into the HOL/Isabelle system, cf. the embedding of Z into HOL [KSW96].

5.1 Overload Resolution

Overload resolution for an arbitrary closed CASL formula proceeds inductively over the structure of the formula, while an environment with the sorts of the variables in the current scope is carried around. Thus it suffices to design an overload resolution algorithm for atomic formulae, which receives an atomic formula and an environment as input and which returns either an expansion of the atomic formula or the message "not resolvable".

There is a simple, naive overload resolution algorithm: just collect the set of *all* expansions of a given atomic formula, and check that they are all equivalent. But this bears a lot of redundancy. The "least sort parse" algorithm for regular signatures described in [GM92], p. 252, just picks *one* expansion (the one with least profile) and forgets about all the other ones. Of course, regularity is defined in a such way precisely that this algorithm works. Since CASL does not impose any condition on signatures, we cannot expect to get such a simple algorithm here. (But we can expect that our algorithm behaves as simple as the least sort parse if the signature *happens* to be regular and locally filtered.) Instead, we use an algorithm which inductively computes the set of *minimal* expansions of a formula, and then checks that this set is non-empty and all of its elements are equivalent[7]. If so, the algorithm returns an arbitrary element of the set, otherwise it returns "not resolvable". A detailed description of this algorithm can be found in [MKKB97].

5.2 Theorem proving

The traditional way to proceed would be to build a calculus and theorem prover for $SubPFOL$, the underlying institution of $CASL$, from scratch. A calculus for

[7] In order to make the equivalence check efficient, the expansions are divided into equivalence classes in each inductive step.

SubPFOL can be found in [CMR97]. But this approach has the disadvantages that much work has to be redone and that for $PFOL$ (partial first-order logic with *total assignments*) there can exist only calculi (like that in [Bur82]) that do not keep the simple substitution rule of FOL (total first-order logic).

It is already clear now that good theorem provers for $SubPFOL$ are available. This is because it is possible to translate a $SubPFOL$-theory to a first-order (FOL) theory and then just use any theorem proving tool for first-order logic. The translation is done in two steps: the first step is to translate subsorted partial first-order logic to partial first-order logic, as was indicated in section 3. The second step is the translation of partial first-order logic to total first-order logic, as described in [CMR98]. Both translation steps are particular cases of the *borrowing* technique proposed in [CM97]. The details are worked out in [MKKB97].

Note that the translation to FOL allows one to use (among others) the usual substitution- and resolution-based theorem provers for FOL. In general, such theorem provers can be used directly only for partial logics with *partial* variable valuations, see also [Sco79].

6 Discussion

In this paper we have presented the subsorting approach of the CASL language. The main novelty is that there are no requirements like monotonicity, regularity or local filtration imposed on subsorted signatures.

The main argument against these requirements is that they give complications with modularity. In order to be able to combine specifications and to use the pushout-approach to parameterization, the signature category has to be cocomplete. To guarantee cocompleteness of the signature category, one either has to restrict the category of signatures in a rather complicated way [HN96] or to confuse the user with unexpected identifications of sorts and extra sorts in the colimit [Mos96, Mos97a].

A second argument against imposing monotonicity (or regularity, which entails monotonicity) is that monotonicity rules out overloading of constants, which we consider very useful. Many many-sorted frameworks with overloading already allow them, and these frameworks should be easily embeddable into CASL.

It turns out that the notions (of models) and results (about initiality and freeness) remain as simple as in the many-sorted case (and can even be directly carried over!). The only price for dropping the requirements is a more complicated definition of well-formed atomic formulae than in the regular case. However, our overload resolution algorithm sketched in section 5 shows that this does not add too much complexity, and in the case that a signature happens to be regular and locally filtered, even the least sort parse algorithm is simulated.

The idea to drop monotonicity and regularity is not new, it stems from Goguen and Diaconescu [GD94]. They point out that the result about existence of initial models also holds for non-regular signatures. However, they still assume local filtration, and they do not propose how overload resolution should be done.

Local filtration (in combination with regularity) is required by Goguen and Meseguer [GM92] in order to ensure that equational satisfaction is closed under isomorphism. This is necessary, because they define equations $t_1 = t_2$ to be well-formed if just the least sorts of t_1 and t_2 are in the same connected component. (Note, that this means that the least sorts are not required to have a common upper bound). This liberal definition has the advantage that well-formedness of $t_1 = t_2$ and $t_2 = t_3$ ensures well-formedness of $t_1 = t_3$.

The idea to drop local filtration has been proposed by Haxthausen [Hax97] for a subsorting approach with implicit, possibly non-injective coercions. However, regularity was still assumed. In order to ensure that equational satisfaction was still closed under isomorphism, it was proposed to use explicitly sorted equations $t_1 =_s t_2$, which were well-formed only if the least sorts of the terms had the sort s as a common upper bound. The price for this was that transitivity of well-formedness of equations only holds for equations over the same sort.

The same idea is adopted in CASL. However, here it was possible to leave the sort implicit. The following example illustrates how transitivity of well-formedness of equations in CASL only holds for equations over the same sort:

sorts $s, t, u;\ t < s;\ t < u$
ops $a : s;\ b : t;\ c : u$
$a = b;\ b = c$

$a = b$ is well-formed with sort s, $b = c$ is well-formed with sort u, but $a = c$ is ill-formed. We can repair this by considering $a \downarrow t = c \downarrow t$, which is well-formed, but has a semantics different from $a = c$ in [GM92]. Note that the problem of satisfaction not being closed under model isomorphism does not arise in CASL.

The overloading relations lead to identification of related functions and predicates in the semantics. This implements the "same name – same meaning" principle that is pervasive in the CASL language (cf. e.g. unions of specifications). We have deliberately excluded strong overloading in the sense of [GD94]. An alternative notion of signature would provide the overloading relations explicitly instead of deriving them from the other information in the signature, thus allowing strong overloading. In [GD94] it is argued that this may be used, for example, to model overwriting in the object oriented paradigm. We are not convinced that this suffices really to model object orientation, so both object orientation and strong overloading are left for extensions of CASL.

Acknowledgements The authors would like to thank all the participants of CoFI, and in particular Peter Mosses and Olaf Owe, for their contributions of ideas for the subsorting approach.

References

[Bur82] P. Burmeister. Partial algebras — survey of a unifying approach towards a two-valued model theory for partial algebras. *Algebra Universalis*, 15:306–358, 1982.

[CM97] M. Cerioli and J. Meseguer. May I borrow your logic? (transporting logical structures along maps). *Theoretical Computer Science*, 173:311–347, 1997.

[CMR98] M. Cerioli, T. Mossakowski, and H. Reichel. From total equational to partial first order logic. In E. Astesiano, H.-J. Kreowski, and B. Krieg-Brückner, editors, *Algebraic Foundations of Systems Specifications*[8]. To appear, 1998.

[GD94] Joseph Goguen and Răzvan Diaconescu. An Oxford survey of order sorted algebra. *Mathematical Structures in Computer Science*, 4(3):363–392, September 1994.

[GM86] J. Goguen and J. Meseguer. Eqlog: Equality, types, and generic modules for logic programming. In Douglas DeGroot and Gary Lindstrom, editors, *Functional and Logic Programming*, pages 295–363. Prentice-Hall, 1986.

[GM92] J. A. Goguen and J. Meseguer. Order-sorted algebra I: equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 105:217–273, 1992.

[Hax97] A.E. Haxthausen. Order-sorted algebraic specifications with higher-order functions. *Theoretical Computer Science*, 183:157–185, 1997.

[HN96] A.E. Haxthausen and F. Nickl. Pushouts of order-sorted algebraic specifications. In *Proceedings of AMAST'96*, volume 1101 of *Lecture Notes in Computer Science*, pages 132–147. Springer-Verlag, 1996.

[KSW96] Kolyang, T. Santen, and B. Wolff. A structure preserving encoding of Z in Isabelle/HOL. In *Proc. 1996 International Conference on Theorem Proving in Higher Order Logic (Turku)*, volume 1125 of *Lecture Notes in Computer Science*, pages 283–298. Springer-Verlag, 1996.

[LD97] The CoFI Task Group on Language Design. CASL – the CoFI Algebraic Specification Language – Design Proposal[9]. 1997.

[MKKB97] T. Mossakowski, Kolyang, and B. Krieg-Brückner. Static semantic analysis of CASL. Talk at the 12th Workshop on Algebraic Development Techniques, Tarquinia. Paper has been submitted, 1997.

[Mos96] T. Mossakowski. *Representations, hierarchies and graphs of institutions.* PhD thesis, Bremen University, 1996.

[Mos97a] T. Mossakowski. Colimits of order-sorted specifications revisited. Talk at the 12th Workshop on Algebraic Development Techniques, Tarquinia. Paper has been submitted, 1997.

[Mos97b] P. Mosses. CoFI: The common framework initiative for algebraic specification and development. In M. Bidoit and M. Dauchet, editors, *TAPSOFT'97*, volume 1214 of *Lecture Notes in Computer Science*, pages 115–137. Springer-Verlag, 1997.

[Rei87] H. Reichel. *Initial Computability, Algebraic Specifications and Partial Algebras.* Oxford Science Publications, 1987.

[Sco79] D. S. Scott. Identity and existence in intuitionistic logic. In M.P. Fourman, C.J. Mulvey, and D.S. Scott, editors, *Application of Sheaves*, volume 753 of *Lecture Notes in Mathematics*, pages 660–696. Springer-Verlag, 1979.

This article was processed using the LaTeX macro package with LLNCS style

[8] Available at http://www.informatik.uni-bremen.de/~kreo/ifip-WG1.3/ifip_chapters/chapters.html

[9] Available at http://www.brics.dk/Projects/CoFI/CASL/Documents/Proposal