Entailment and Disentailment of Order-Sorted Feature Constraints

Hassan Aït-Kaci and Andreas Podelski

Digital Equipment Corporation, Paris Research Laboratory
85, avenue Victor Hugo, 92500 Rueil-Malmaison, France

{hak,podelski}@prl.dec.com

Abstract. LIFE uses matching on order-sorted feature structures for passing arguments to functions. As opposed to unication which amounts to normalizing a conjunction of constraints, solving a matching problem consists of deciding whether a constraint (guard) or its negation are entailed by the context. We give a complete and consistent set of rules for entailment and disentailment of order-sorted feature constraints. These rules are directly usable for relative simplication, a general proof-theoretic method for proving guards in concurrent constraint logic languages using guarded rules.

1 Introduction

LIFE [5] extends the computational paradigm of Logic Programming in two essential ways:

- using a data structure richer than that provided by rst-order constructor terms; and,
- allowing interpretable functional expressions as *bona de* terms.

The rst extension is based on ψ-terms which are attributed partially-ordered sorts denoting sets of objects [1, 2]. In particular, ψ-terms generalize rst-order constructor terms in their rôle as data structures in that they are endowed with a unication operation denoting type intersection.

The second extension deals with building into the unication operation a means to reduce functional expressions using denitions of interpretable symbols over data patterns. The basic insight is that unication is no longer seen as an atomic operation by the resolution rule. Indeed, since unication amounts to normalizing a conjunction of equations, and since this normalization process commutes with resolution, these equations may be left in a normal form that is not a fully solved form. In particular, if an equation involves a functional expression whose arguments are not sufciently instantiated to match a *deniens* of the function in question, it is simply left untouched. Resolution may proceed until the arguments are *proven* to match a denition from the accumulated constraints in the context [3]. This simple idea turns out invaluable in practice.

This technique|delaying reduction and enforcing determinism by allowing only equivalence reductions|is called *residuation* [3]. It does not have to be limited to functions. Therefore, we explain it for the general case of relations. Intuitively, the arguments of a relation which are constrained by the guard are its input parameters and correspond to the arguments of a function. This has been used as an implicit control mechanism in general concurrent constraint logic programming schemes; *e.g.*, the logic of guarded Horn-clauses studied by Maher [11], Concurrent Constraint Programming (CCP) [12], and Kernel Andorra Prolog (KAP) [9]. These schemes are parameterized with respect to an abstract class of constraint systems. An incremental test for entailment and disentailment between constraints is needed for advanced control mechanisms such as delaying, coroutining, synchronization, committed choice, and deep constraint propagation. LIFE is formally an instance of this scheme, namely a CLP language using a constraint system based on order-sorted feature (OSF) structures [5]. It employs a related, but limited, suspension strategy to enforce deterministic functional application. Roughly, these systems are concurrent thanks to a new effective discipline for procedure parameter-passing that can be described as \call-by-constraint-entailment" (as opposed to Prolog's call-by-unication).

The most direct way to explain the issue is with an example. In LIFE, one can dene functions as usual; say:

$$\begin{aligned} &\mathit{fact}(0) &&\rightarrow 1.\\ &\mathit{fact}(N : \mathit{int}) &&\rightarrow N * \mathit{fact}(N-1). \end{aligned}$$

More interesting is the possibility to compute with partial information. For example:

$$\begin{aligned} &\mathit{minus}(\mathit{negint}) &&\rightarrow \mathit{posint}.\\ &\mathit{minus}(\mathit{posint}) &&\rightarrow \mathit{negint}.\\ &\mathit{minus}(\mathit{zero}) &&\rightarrow \mathit{zero}. \end{aligned}$$

Let us assume that the symbols *int*, *posint*, *negint*, and *zero* have been dened as sorts with the approximation ordering such that $\mathit{posint}, \mathit{zero}, \mathit{negint}$ are pairwise incompatible subsorts of the sort *int* (*i.e.*, $\mathit{posint} \wedge \mathit{zero} = -$, $\mathit{negint} \wedge \mathit{zero} = -$, $\mathit{posint} \wedge \mathit{negint} = -$). This is declared in LIFE as $\mathit{int} := \{\mathit{posint}; \mathit{zero}; \mathit{negint}\}$. Furthermore, we assume the sort denition $\mathit{posint} := \{\mathit{posodd}; \mathit{poseven}\}$; *i.e.*, *posodd* and *poseven* are subsorts of *posint* and mutually incompatible.

The LIFE query $Y = \mathit{minus}(X : \mathit{poseven})$? will return $Y = \mathit{negint}$. The sort *poseven* of the actual parameter is incompatible with the sort *negint* of the formal parameter of the rst rule dening the function *minus*. Therefore, that rule is skipped. The sort *poseven* is more specic than the sort *posint* of the formal parameter of the second rule. Hence, that rule is applicable and yields the result $Y = \mathit{negint}$.

The LIFE query $Y = \mathit{minus}(X : \mathit{string})$ will fail. Indeed, the sort *string* is incompatible with the sort of the formal parameter of every rule dening *minus*.

Thus, in order to determine which of the rules, if any, dening the function in a given functional expression will be applied, two tests are necessary:

- verify whether the actual parameter is more specic than or equal to the formal parameter;
- verify whether the actual parameter is at all compatible with the formal parameter.

What happens if both of these tests fail? For example, consider the query consisting of the conjunction:

$$Y = \mathit{minus}(X : \mathit{int}), X = \mathit{minus}(\mathit{zero})?$$

Like Prolog, LIFE follows a left-to-right resolution strategy and examines the equation $Y = minus(X : int)$ rst. However, both foregoing tests fail and deciding which rule to use among those dening *minus* is inconclusive. Indeed, the sort *int* of the actual parameter in that call is neither more specic than, nor incompatible with, the sort *negint* of the rst rule's formal parameter. Therefore, the function call will *residuate* on the variable X. This means that the functional evaluation is suspended pending more information on X. The second goal in the query is treated next. There, it is found that the actual parameter is incompatible with the rst two rules and is the same as the last rule's. This allows reduction and binds X to *zero*. At this point, X has been instantiated and therefore the residual equation pending on X can be reexamined. Again, as before, a redex is found for the last rule and yields $Y = zero$.

The two tests above can in fact be worded in a more general setting. Viewing data structures as constraints, \more specic" is simply a particular case of constraint entailment. We will say that a constraint *disentails* another whenever their conjunction is unsatisable; or, equivalently, whenever it entails its negation. In particular, rst-order matching is deciding entailment between constraints consisting of equations over rst-order terms. Similarly, deciding uniability of rst-order terms amounts to deciding \compatibility" in the sense used informally above.

The suspension/resumption mechanism illustrated in our example is repeated each time a residuated actual parameter becomes more instantiated from the context; *i.e.*, through solving other parts of the query. Therefore, it is most benecial for a practical algorithm testing entailment and disentailment to be incremental. This means that, upon resumption, the test for the instantiated actual parameter builds upon partial results obtained by the previous test. One outcome of the results presented in this paper is that it is possible to build such a test; namely, an algorithm deciding simultaneously two problems in an incremental manner|entailment and disentailment. The technique that we have devised to do that is called *relative simplication* of constraints [4].

We have organized this paper as follows. In Section 2, we review background on our OSF formalism. This is for the sake of staying self-contained since its technical notation and terminology is pervasive in this paper's presentation. In Section 3, we give rules for incrementally deciding entailment and disentailment of OSF constraints, and we make explicit the effectuality of relative simplication. In Section 4, we prove the termination of the rules. In Section 5, we show the correctness and completeness of these rules. Section 6 establishes the property of independence of negated OSF constraints. Finally, we conclude in Section 7.

2 OSF Formalism

We introduce briey the OSF formalism terminology and notation that we use. For a thorough investigation of these notions, the reader is referred to [5].

2.1 OSF algebras and OSF constraints

The building blocks of OSF algebras are sorts and features.

An *order-sorted feature signature* (or simply OSF signature) is a tuple $\langle \mathcal{S}, \leq, \wedge, \mathcal{F} \rangle$ such that:

- $\mathcal{S}$ is a set of *sorts* containing the sorts $\top$ and $-$;
- $\leq$ is a decidable partial order on $\mathcal{S}$ such that $-$ is the least and $\top$ is the greatest element;
- $\langle \mathcal{S}, \leq, \wedge \rangle$ is a lower semi-lattice ($s \wedge s'$ is called the greatest common subsort of sorts s and s');
- $\mathcal{F}$ is a set of *feature symbols*.

An OSF signature has the following interpretation. An *OSF algebra* over the signature $\langle \mathcal{S}, \leq, \wedge, \mathcal{F} \rangle$ is a structure:

$$\mathcal{A} = \langle\, D^{\mathcal{A}}\ ,\ \left(s^{\mathcal{A}}\right)_{s \in \mathcal{S}}\ ,\ \left(\ell^{\mathcal{A}}\right)_{\ell \in \mathcal{F}}\ \rangle$$

such that:

- $D^{\mathcal{A}}$ is a non-empty set, called the *domain* of $\mathcal{A}$ (or, universe);
- for each sort symbol s in $\mathcal{S}$, $s^{\mathcal{A}}$ is a subset of the domain; in particular, $\top^{\mathcal{A}} = D^{\mathcal{A}}$ and $-^{\mathcal{A}} = \emptyset$;
- the greatest lower bound (*GLB*) operation on the sorts is interpreted as the intersection; *i.e.*, $(s \wedge s')^{\mathcal{A}} = s^{\mathcal{A}} \cap s'^{\mathcal{A}}$ for two sorts s and s' in $\mathcal{S}$.
- for each feature ℓ in $\mathcal{F}$, $\ell^{\mathcal{A}}$ is a total unary function from the domain into the domain; *i.e.*, $\ell^{\mathcal{A}} : D^{\mathcal{A}} \mapsto D^{\mathcal{A}}$;

The notion of OSF algebra calls naturally for a corresponding notion of homomorphism preserving structure appropriately. Namely,

Denition 1 OSF Homomorphism. An OSF algebra *homomorphism* $\gamma : \mathcal{A} \mapsto \mathcal{B}$ between two OSF algebras $\mathcal{A}$ and $\mathcal{B}$ is a function $\gamma : D^{\mathcal{A}} \mapsto D^{\mathcal{B}}$ such that:

- $\gamma\big(\ell^{\mathcal{A}}(d)\big) = \ell^{\mathcal{B}}\big(\gamma(d)\big)$ for all $d \in D^{\mathcal{A}}$;
- $\gamma\big(s^{\mathcal{A}}\big) \subseteq s^{\mathcal{B}}$.

It is straightforward to verify that OSF algebras together with OSF homomorphisms form a category. We call this category OSF.

Let $\mathcal{V}$ be a countably innite set of variables.

Denition 2 OSF Constraint. An atomic OSF constraint is one of:

- $X : s$,
- $X \doteq X'$,
- $X.\ell \doteq X'$,

where X and X' are variables in $\mathcal{V}$, s is a sort in $\mathcal{S}$, and ℓ is a feature in $\mathcal{F}$. An OSF constraint is a conjunction of atomic OSF constraints.

The set $Var(\phi)$ of variables occurring in an OSF constraint ϕ is dened in the standard way. OSF constraints will always be considered equal if they are equal modulo the commutativity, associativity and idempotence of conjunction \&." Therefore, a constraint can also be formalized as the set consisting of its conjuncts. As usual, the empty conjunction corresponds to the propositional constant interpreted as *true*.

Let $\mathcal{A}$ be an OSF algebra. We call $Val(\mathcal{A}) = \{\alpha : \mathcal{V} \mapsto D^{\mathcal{A}}\}$ the set of all possible valuations in the interpretation $\mathcal{A}$. The semantics of OSF constraints is straightforward.

Given $\mathcal{A}$ is OSF algebra, an OSF constraint ϕ is *satisable* in $\mathcal{A}$, if there exists a valuation $\alpha : \mathcal{V} \mapsto D^{\mathcal{A}}$ such that $\mathcal{A}, \alpha \models \phi$, where:

$$
\begin{array}{lll}
\mathcal{A}, \alpha \models X : s & \text{iff} & \alpha(X) \in s^{\mathcal{A}}; \\
\mathcal{A}, \alpha \models X \doteq Y & \text{iff} & \alpha(X) = \alpha(Y); \\
\mathcal{A}, \alpha \models X.\ell \doteq Y & \text{iff} & \ell^{\mathcal{A}}(\alpha(X)) = \alpha(Y); \\
\mathcal{A}, \alpha \models \phi \;\&\; \phi' & \text{iff} & \mathcal{A}, \alpha \models \phi \text{ and } \mathcal{A}, \alpha \models \phi'.
\end{array}
$$

2.2 ψ-Terms

Denition 3 ψ-Term. A ψ-term ψ is an expression of the form:

$$X : s(\ell_1 \Rightarrow \psi_1, \ldots, \ell_n \Rightarrow \psi_n)$$

where

- X is a variable in $\mathcal{V}$ called the root of ψ;
- s is a sort different from $-$ in $\mathcal{S}$;
- $\ell_1, \ldots, \ell_n$ are pairwise different features in $\mathcal{F}$, $n \geq 0$;
- $\psi_1, \ldots, \psi_n$ are again ψ-terms; and,
- no variable Y occurring in ψ is the root variable of more than one non-trivial ψ-term (*i.e.*, different than $Y : \top$).

Note that the equation above includes $n = 0$ as a base case. That is, the simplest ψ-terms are of the form $X : s$.

We can associate to a ψ-term $\psi = X : s(\ell_1 \Rightarrow \psi_1, \ldots, \ell_n \Rightarrow \psi_n)$ the OSF constraint:

$$
\begin{array}{rl}
\phi(\psi) = & X : s \;\&\; X.\ell_1 \doteq Y_1 \;\&\; \ldots \;\&\; X.\ell_n \doteq Y_n \\
 & \qquad\quad \&\; \phi(\psi_1) \qquad \&\; \ldots \;\&\; \phi(\psi_n)
\end{array}
$$

where $Y_1, \ldots, Y_n$ are the roots of $\psi_1, \ldots, \psi_n$, respectively. We say that the OSF constraint $\phi(\psi)$ is obtained from *dissolving* the ψ-term ψ, and refer to the OSF constraint as the *dissolved ψ-term*. We will often deliberately confuse a ψ-term ψ with its dissolved form $\phi(\psi)$ and simply refer to $\phi(\psi)$ simply as ψ.

Given the interpretation $\mathcal{A}$, the *denotation* $[\![\psi]\!]^{\mathcal{A},\alpha}$ *under a valuation* $\alpha : \mathcal{V} \mapsto D^{\mathcal{A}}$ of a ψ-term ψ with root X is given as:

$$[\![\psi]\!]^{\mathcal{A},\alpha} = \{d \in D^{\mathcal{A}} \mid \alpha(X) = d,\ \mathcal{A}, \alpha \models \psi\}.$$

Note that this is either the singleton $\{\alpha(X)\}$ or the empty set.

The *type-as-set denotation* of a ψ-term ψ is dened as the set of domain elements:

$$[\![\psi]\!]^{\mathcal{A}} = \bigcup_{\alpha \in Val(\mathcal{A})} [\![\psi]\!]^{\mathcal{A},\alpha}.$$

This amounts to saying that:

$$
\begin{array}{rl}
[\![\psi]\!]^{\mathcal{A}} = \{d \in D^{\mathcal{A}} \mid & \text{there exists } \alpha \in Val(\mathcal{A}) \text{ such that} \\
 & \alpha(Z) = d, \text{ and } \mathcal{A}, \alpha \models \exists \mathcal{X}\ Z : \psi\}
\end{array}
$$

where Z is a new variable not occurring in ψ, $\mathcal{X} = Var(\psi)$, $Z : \psi$ stands for $Z \doteq X \,\&\, \psi$, and $X \in \mathcal{X}$ is ψ's root variable.

A ψ-term ψ with root X corresponds to a unique rooted graph g which is the direct translation of the constraint ψ together with an indication of the root. The nodes of g are exactly the variables of ψ. A node Z is labeled by the sort s if the conjunction ψ contains a non-trivial sort constraint $Z : s$, and by the sort $\top$, otherwise. For every feature constraint $Y.\ell \doteq Z$ the graph g has a directed edge (Y, Z) which is labeled by the feature ℓ. The root of g is the node X. Clearly, g is the natural graphical representation of ψ.

2.3 Syntactic interpretations

Among all OSF algebras, there are those whose domain elements are concrete data structures. We call these *syntactic interpretations*. We will now present three important examples obtained directly from the syntactic expressions of ψ-terms. They turn out to be *canonical interpretations* for OSF constraints.[1]

The most immediate syntactic OSF interpretation is the OSF algebra Ψ of ψ-terms. The domain of Ψ is the set of all ψ-terms, up to graph representation. That is, we identify ψ-terms as values of Ψ if they are represented by the same graph. For example, the two ψ-terms $Y : s(\ell_1 \Rightarrow X : s', \ell_2 \Rightarrow X)$ and $Y : s(\ell_1 \Rightarrow X, \ell_2 \Rightarrow X : s')$ clearly correspond to the same object. Indeed, they have the same OSF graph representation.

A sort $s \in \mathcal{S}$ is interpreted as:

$$s^{\Psi} = \{\psi \in D^{\Psi} \mid Sort(Root(\psi)) \leq s\},$$

where $Sort(Root(\psi))$ is the root sort of the graph of ψ. A feature $\ell \in \mathcal{F}$ is interpreted as a function $\ell^{\Psi} : D^{\Psi} \mapsto D^{\Psi}$ as follows. Let ψ be a ψ-term and g its graph. If (X, Y) is the edge of g labeled by ℓ, then $\ell^{\Psi}(g)$ is the ψ-term represented by the maximally connected subgraph g' of g rooted at the node Y. That is, g' is obtained by removing all nodes and edges which are not reachable by a directed path from the node Y.

If X does not have the feature ℓ, *i.e.*, there is no outgoing edge from the root of g labeled ℓ, then ℓ^{Ψ} is the ψ-term $Z_{\ell,\psi} : \top$, for a new variable $Z_{\ell,\psi}$ uniquely determined by the feature ℓ and the ψ-term ψ.

For example, taking $\psi = X : \top(\ell_1 \Rightarrow Y : s, \ell_2 \Rightarrow X)$, we have $\ell_1^{\Psi}(\psi) = Y : s$, $\ell_2^{\Psi}(\psi) = \psi$, and $\ell_3^{\Psi}(\psi) = Z_{\ell_3,\psi} : \top$.

We obtain two other examples of OSF algebras when we factorize the ψ-term domain by further identifying values. The rst one identies two ψ-terms which are equal up to variable renaming. The obtained domain obviously spans an OSF algebra. We call this OSF algebra Ψ_0.

The second one is obtained from Ψ_0 by further identifying two ψ-terms if their (possibly innite) tree unfoldings are equal. A tree unfolding is obtained from a ψ-term by associating a unique node to every feature path. It is well known that a rooted directed graph represents a unique rational tree [8]. In our case, we obtain trees whose nodes are labeled by sorts and whose edges are labeled by features. We call these

[1] If an OSF constraint is satisable in some interpretation, then it is also satisable in all canonical interpretations.

(rational) OSF trees. It is again clear that the set of all OSF trees spans an OSF algebra $\mathcal{T}$.[2]

Formally, OSF algebras can also be introduced as logical structures, namely models providing interpretations for the sort symbols as unary predicates and the feature symbols as unary functions, which satisfy the *Sort Axiom* saying, for all sorts s and s',

$$X : s \mathbin{\&} X : s' \;\rightarrow\; X : s \wedge s'.$$

Furthermore, both Ψ_0 and $\mathcal{T}$ satisfy a *Constructibility Axiom* stating essentially the satisability of any OSF constraint ϕ coming from dissolving a ψ-term ψ. More precisely, if $\mathcal{X} = Var(\phi)$ and, for $i = 1, \ldots, n$, $X_i.\ell_i \doteq Y \notin \phi$ for any variable Y, and $Y_i \notin Var(\phi)$, and $X_i \in \mathcal{X}$, then this axiom states the validity of:

$$\forall Y_1. \ldots \forall Y_n.\ \exists \mathcal{X}.\ \phi \mathbin{\&} X_1.\ell_1 \doteq Y_1 \mathbin{\&} \ldots \mathbin{\&} X_n.\ell_n \doteq Y_n.$$

The constructibility axiom is a generalization of the axiom of functionality which is valid for rst-order terms. Namely, the axiom which guarantees that, given a constructor symbol f of rank n, an individual $X = f(Y_1, \ldots, Y_n)$ exists if individuals Y_i exist, $i = 1, \ldots, n$. Formally, taking $\phi = X : f$,

$$\forall Y_1. \ldots \forall Y_n.\ \exists X.\ X : f \mathbin{\&} X.1 \doteq Y_1 \mathbin{\&} \ldots \mathbin{\&} X.n \doteq Y_n.$$

The form we give for constructibility is indeed more general than plain functionality since it states the existence of something which is not valid for rst-order terms; *e.g.*, self-referential individuals. For example, $\exists X.\ X.\ell \doteq X$ is obtained as an instance of our axiom by taking $n = 0$ and $\phi = X.\ell \doteq X$.

2.4 OSF unication

We describe next how to determine whether an OSF constraint ϕ is consistent; *i.e.*, if it is satisable in some OSF algebra $\mathcal{A}$|and, therefore, in particular in Ψ. Unication of two ψ-terms reduces to this problem.

Denition 4 Solved OSF Constraints. An OSF constraint ϕ is called <u>*solved*</u> if for every variable X, ϕ contains:

- at most one sort constraint of the form $X : s$, with $- < s$;
- at most one feature constraint of the form $X.\ell \doteq Y$ for each ℓ; and,
- no other occurrence of the variable X if it contains the equality constraint $X \doteq Y$.

In [5], we show that an OSF constraint in solved form is always satisable. Now, by Denition 3, the OSF constraint obtained as the dissolved form of any ψ-term ψ is *de facto* in solved form.[3] Hence, such a constraint is always satisable. It is so, in particular, in the canonical interpretation Ψ with, interestingly enough, the valuation that assigns to each variable X in ψ the value in D^{Ψ} that is the very ψ-term rooted in X in ψ. For this reason, a ψ-term can also be seen as a variable substitution.

Given an OSF constraint ϕ, it can be normalized by choosing non-deterministically and applying any applicable rule among the transformations rules shown in Figure 1

Feature Decomposition:

$$\text{(B.1)}\ \frac{\psi\ \&\ U.\ell \doteq V\ \&\ U.\ell \doteq W}{\psi\ \&\ U.\ell \doteq V\ \&\ W \doteq V}$$

Sort Intersection:

$$\text{(B.2)}\ \frac{\psi\ \&\ U : s\ \&\ U : s'}{\psi\ \&\ U : s \wedge s'}$$

Variable Elimination:

$$\text{(B.3)}\ \frac{\psi\ \&\ U \doteq V}{\psi[V/U]\ \&\ U \doteq V} \qquad \textit{if } U \in \mathit{Var}(\psi) \textit{ and } U \neq V$$

Inconsistent Sort:

$$\text{(B.4)}\ \frac{\psi\ \&\ X : \bot}{\bot}$$

Variable Clean-up:

$$\text{(B.5)}\ \frac{\psi\ \&\ U \doteq U}{\psi}$$

Fig. 1. Basic simplication

until none applies. A rule transforms the numerator into the denominator. The expression $\phi[X/Y]$ stands for the formula obtained from ϕ after replacing all occurrences of Y by X.

Theorem 5 OSF Constraint Normalization. *The transformation system of Figure 1 is solution-preserving, nite terminating, and conuent (modulo variable renaming). Furthermore, it always yields a normal form that is either the* false *constraint* $-$ *or an OSF constraint in solved form.*

In our case, the constraint ϕ to be normalized will be of the form $\psi_1\ \&\ \psi_2\ \&\ X_1 \doteq X_2$; *i.e.*, the conjunction of the dissolved ψ-terms ψ_1 and ψ_2 together with an equation identifying their root variables X_1 and X_2. If ϕ normalizes to the *false* constraint, then the two ψ-terms are non-uniable. Otherwise, the resulting solved OSF constraint is a

[2] $\mathcal{T}$ is essentially the feature tree structure of [6] and [7, 13]. The difference lies in our using partially-ordered sorts and total, as opposed to partial, features.

[3] More precisely, this is true if we forget superuous trivial sort constraints of the form $X : \top$.

conjunction of equality constraints and of the dissolved form of some ψ-term. This ψ-term is *the most general unier* of ψ_1 and ψ_2, up to variable renaming. We shall see that this ψ-term has two equivalent order-theoretic characterizations (*cf.*, Propositions 11 and 12).

2.5 OSF orderings and semantic transparency

In this section, we introduce the notion of *endomorphic approximation* which captures precisely and elegantly object inheritance. We also show how it relates to the logic and type views, capturing semantically the essence of constraint entailment.

Endomorphisms on a given OSF algebra $\mathcal{A}$, *i.e.*, homomorphisms from $\mathcal{A}$ to $\mathcal{A}$, induce a natural partial ordering.

Denition 6 Endomorphic Approximation. An approximation preorder $\sqsubseteq_{\mathcal{A}}$ is dened such that, for two elements d and e in $D^{\mathcal{A}}$, *d approximates e* if and only if e is an endomorphic image of d. Formally, $d \sqsubseteq_{\mathcal{A}} e$ iff $\gamma(d) = e$ for some endomorphism $\gamma : \mathcal{A} \mapsto \mathcal{A}$.

We shall omit subscripting $\sqsubseteq_{\mathcal{A}}$ and write $\sqsubseteq$ when $\mathcal{A} = \Psi$. Notice that this ordering on ψ-terms as values of the domain of Ψ translates into an information-theoretic approximation ordering on ψ-terms as types.

We note that endomorphisms on Ψ are graph homomorphisms with the additional sort-compatibility property. A node labeled with sort s is always mapped into a node labeled with s or a subsort of s. An edge labeled with a feature is mapped into an edge labeled with the same feature. Thus, endomorphic approximation captures exactly object-oriented class inheritance. Indeed, if an attribute is present in a class, then it is also present in a subclass with a sort that is the same or rened. Since features are total functions, this also takes care of introducing a new attribute in a subclass: it renes $\top$. Note also, that the restriction of γ to the set of nodes denes a variable binding; it corresponds to the notion of a matching substitution for rst-order terms.

The following fact was established in [5].

Proposition 7 ψ-Terms as Filters. *The denotation of a ψ-term in Ψ is the set of all ψ-terms it approximates; i.e.,*

$$[\![\psi]\!]^{\Psi} = \{\psi' \in D^{\Psi} \mid \psi \sqsubseteq \psi'\}.$$

The next ordering is the ordering on ψ-terms that expresses that one ψ-term is \more specic than" another one.

Denition 8 ψ-Term Subsumption. A ψ-term ψ is *subsumed by* a ψ-term ψ' if and only if the denotation of ψ is contained in that of ψ' in all interpretations. Formally,

$$\psi \leq \psi' \;\; \textit{iff} \;\; [\![\psi]\!]^{\mathcal{A}} \subseteq [\![\psi']\!]^{\mathcal{A}}$$

for all OSF algebras $\mathcal{A}$.

In fact, it is sufcient to limit the above statement to the OSF algebra Ψ only; *i.e.*, $[\![\psi]\!]^{\Psi} \subseteq [\![\psi']\!]^{\Psi}$.

The next and last ordering is a logical ordering on ψ-terms. We state it here in less general terms than in [5].

Denition 9 ψ-Term Entailment. A ψ-term ψ *entails* a ψ-term ψ' if and only if, as constraints, ψ implies the conjunction of ψ' and $X \doteq X'$; more precisely,

$$\psi \succeq \psi' \quad \textit{iff} \quad \models \ \psi \rightarrow \exists \mathcal{U} \ (X \doteq X' \ \& \ \psi')$$

where X, X' are the roots of ψ and ψ' and $\mathcal{U} = \mathit{Var}(\psi')$.

It is again sufcient to state the validity of the implication in the OSF algebra Ψ only (namely, using $\models_\Psi$). This is not true in the more general wording and holds here only because the constraints are obtained by dissolving ψ-terms and their root variables are bound together.

Proposition 10 Semantic Transparency of Orderings. *The following are equivalent:*

- $\psi \sqsubseteq \psi'$ — *ψ approximates ψ';*
- $\psi' \leq \psi$ — *ψ' is a subtype of ψ;*
- $\psi' \succeq \psi$ — *ψ entails ψ';*
- $[\![\psi]\!]^\Psi \subseteq [\![\psi']\!]^\Psi$ — *the set of ψ-terms ltered by ψ is contained in that ltered by ψ'.*

The following two propositions are straightforward. Let ψ_1 and ψ_2 be two ψ-terms with variables renamed apart; *i.e.*, such that $\mathit{Var}(\psi_1) \cap \mathit{Var}(\psi_2) = \emptyset$. Let X_1 and X_2 be their respective root variables. Let ϕ be the normal form of the OSF constraint $\psi_1 \ \& \ \psi_2 \ \& \ X_1 \doteq X_2$.

Proposition 11 ψ-Term Unication. *The normal form ϕ is the* false *constraint if and only if $[\![\psi_1]\!]^\mathcal{A} \cap [\![\psi_2]\!]^\mathcal{A} = \emptyset$, for all OSF algebras $\mathcal{A}$. Otherwise, ϕ is the conjunction of equality constraints and of the dissolved version of some ψ-term ψ. This ψ-term is the $\leq$-GLB of ψ_1 and ψ_2 up to variable renaming; i.e., $[\![\psi]\!]^\mathcal{A} = [\![\psi_1]\!]^\mathcal{A} \cap [\![\psi_2]\!]^\mathcal{A}$.*

Proposition 12 $\sqsubseteq$-LUB of two ψ-terms. *The ψ-term ψ above is approximated by both ψ_1 and ψ_2 and is the least ψ-term for $\sqsubseteq$ (i.e., approximating all other ones) with this property.*

3 Proving OSF Guards

In the following, we use ϕ as the *context* formula. It is assumed to be an *OSF*-constraint in solved form, although not necessarily coming from dissolving a single ψ-term. The variables in ϕ are *global*. We shall use $\mathcal{X}$ to designate the set of global variables $\mathit{Var}(\phi)$ and the letters $X, Y, Z, \ldots,$ for variables in $\mathcal{X}$. We use ψ, a dissolved ψ-term, as the *guard* formula. The variables in ψ are *local* to ψ; *i.e.*, $\mathit{Var}(\phi) \cap \mathit{Var}(\psi) = \emptyset$. We shall use $\mathcal{U}$ to designate the set of local variables $\mathit{Var}(\psi)$ and the letters $U, V, W, \ldots,$ for variables in $\mathcal{U}$. The letter U will always designate the root variable of ψ. We also refer to ϕ as the *actual* parameter, and to ψ as the *formal* parameter. By extension, we will often use the qualiers global/local, actual/formal, and context/guard, with all syntactic entities; *e.g.*, variables, formulae, constraints, or sorts.

We investigate a proof system which decides two problems simultaneously:

- the validity of $\forall \mathcal{X} \ \big(\ \phi \rightarrow \exists \mathcal{U} . \ (\psi \ \& \ U \doteq X) \ \big)$;
- the unsatisability of $\phi \ \& \ \psi \ \& \ U \doteq X$.

The rst test is called a test for *entailment* of the guard by the context, and the second, a test for *disentailment*. This second test is equivalent to testing the validity of the implication $\forall\mathcal{X}\ \big(\ \phi \rightarrow \neg\exists\mathcal{U}\,.\ (\psi\ \&\ U \doteq X)\ \big)$.

Since both tests amount to deciding whether the context implies the guard or its negation, all local variables are existentially quantied and all global variables are universally quantied.

The *relative-simplication* system for OSF constraints is given by the rules in Figures 2, 3, and 4. An OSF constraint ψ simplies to ψ' relatively to ϕ by a

Feature Decomposition:

$$\text{(F.1)}\ \frac{\psi\ \&\ U.\ell \doteq V\ \&\ U.\ell \doteq W}{\psi\ \&\ U.\ell \doteq V\ \&\ W \doteq V}$$

Relative Feature Decomposition:

$$\text{(F.2)}\ \frac{\psi\ \&\ U \doteq X\ \&\ U.\ell \doteq V}{\psi\ \&\ U \doteq X\ \&\ V \doteq Y} \qquad \textit{if } X.\ell \doteq Y \in \phi$$

Relative Feature Equality:

$$\text{(F.3)}\ \frac{\psi\ \&\ U \doteq X_1\ \&\ U \doteq X_2\ \&\ V \doteq Y_1}{\psi\ \&\ U \doteq X_1\ \&\ U \doteq X_2\ \&\ V \doteq Y_1\ \&\ V \doteq Y_2} \qquad \begin{array}{l}\textit{if } X_1.\ell \doteq Y_1 \in \phi, X_2.\ell \doteq Y_2 \in \phi \\ \textit{and } V \doteq Y_2 \notin \psi\end{array}$$

Variable Introduction:

$$\text{(F.4)}\ \frac{\psi\ \&\ U \doteq X_1\ \&\ U \doteq X_2}{\psi\ \&\ U \doteq X_1\ \&\ U \doteq X_2\ \&\ V \doteq Y_1\ \&\ V \doteq Y_2} \qquad \begin{array}{l}\textit{if } X_1.\ell \doteq Y_1 \in \phi, X_2.\ell \doteq Y_2 \in \phi \\ \textit{and } Y_1 \notin \mathit{Var}(\psi) \textit{ and } Y_2 \notin \mathit{Var}(\psi) \\ \textit{where } V \textit{ is a new variable}\end{array}$$

Fig. 2. Simplication relatively to ϕ: Features

simplication rule ρ if $\frac{\psi}{\psi'}$ is an instance of ρ and the applicability condition (on ϕ and on ψ) is satised. We say that ψ simplies to ψ' relatively to ϕ if it does so in a nite number of steps.

The relative-simplication system preserves an important invariant property: *a global variable never appears on the left of a variable equality constraint in the formula being simplied*. Thus, an equality $U \doteq X$ is a *directed* relation binding the local variable U to the global variable X. Furthermore, a global variable is never eliminated by a local one, or *vice versa*.

Sort Intersection:

$$\text{(S.1)}\ \frac{\psi\ \&\ U : s\ \&\ U : s'}{\psi\ \&\ U : s \wedge s'}$$

Sort Containment:

$$\text{(S.2)}\ \frac{\psi\ \&\ U \doteq X\ \&\ U : s}{\psi\ \&\ U \doteq X} \qquad \textit{if } X : s' \in \phi \textit{, and } s' \leq s$$

Sort Renement:

$$\text{(S.3)}\ \frac{\psi\ \&\ U \doteq X\ \&\ U : s}{\psi\ \&\ U \doteq X\ \&\ U : s \wedge s'} \qquad \textit{if } X : s' \in \phi \textit{, and } s \wedge s' < s$$

Relative Sort Intersection:

$$\text{(S.4)}\ \frac{\psi\ \&\ U \doteq X\ \&\ U \doteq X'}{\psi\ \&\ U \doteq X\ \&\ U \doteq X'\ \&\ U : s \wedge s'} \qquad \begin{array}{l}\textit{if } X : s \in \phi, X' : s' \in \phi,\\ s \wedge s' < s, s \wedge s' < s',\\ \textit{and } U : s'' \notin \psi \textit{, for any sort } s''\end{array}$$

Sort Inconsistency:

$$\text{(S.5)}\ \frac{\psi\ \&\ U : -}{-}$$

Fig. 3. Simplication relatively to ϕ: Sorts

A set of bindings $U_i \doteq X_i$, $i = 1, \ldots, n$ is a *functional binding* if all the variables U_i are mutually distinct.

The effectuality of the relative-simplication system is summed up in the following statement:

> **Effectuality of Relative-Simplication** *The solved OSF constraint ϕ entails (resp., disentails) the OSF constraint $\exists U.\ (U \doteq X\ \&\ \psi)$ if and only if the normal form ψ' of $\psi\ \&\ U \doteq X$ relatively to ϕ is a conjunction of equations making up a functional binding (resp., is the* false *constraint $\psi' = -$).*

There are two technical remarks to be made. Firstly, observe that in our formulation of the entailment/disentailment problem, the implication contains *only one* equality $U \doteq X$ binding *only one* global variable. However, this is not a restriction. Equations $U_1 \doteq X_1, \ldots, U_n \doteq X_n$ can be equivalently replaced by adding $X_1 \doteq X.1\ \&\ \ldots\ \&\ X_n \doteq$

Relative Variable Elimination:

$$\text{(E.1)}\ \frac{\psi\ \&\ U \doteq X\ \&\ V \doteq X}{\psi[U/V]\ \&\ U \doteq X\ \&\ V \doteq X}\quad \textit{if } V \in \mathit{Var}(\psi),\ V \doteq X \notin \psi, \textit{ and } U \neq V$$

Equation Entailment:

$$\text{(E.2)}\ \frac{\psi\ \&\ U \doteq X\ \&\ U \doteq Y}{\psi\ \&\ U \doteq X}\qquad \textit{if } X = Y \textit{ or if } X \doteq Y \in \phi.$$

Fig. 4. Simplication relatively to ϕ: Equations

$X.n$ to the context ϕ and $U_1 \doteq U.1\ \&\ \ldots\ \&\ U_n \doteq U.n\ \&\ U \doteq X$ to ψ, where X and U are new. That is, one obtains the conjunction of one equality $U \doteq X$ and a guard which, again, is a dissolved ψ-term.

Secondly, the fact that ψ is a dissolved ψ-term rooted in U ensures that the test of entailment of $\psi\ \&\ U \doteq X$ by ϕ does not depend on whether the implication holds in *all* OSF interpretations, or only in Ψ, or $\mathcal{T}$. This is not necessarily so if U is not the root of ψ. Indeed, let us assume that U is *not* the root of ψ; for example, take ψ to be $V.\ell \doteq U$. Clearly, while $\forall X\ \big(\top \rightarrow \exists U \exists V\ (\psi\ \&\ U \doteq X)\big)$ holds in Ψ and $\mathcal{T}$, it does not hold in all OSF algebras where it is not guaranteed that every element is the ℓ-image of some other element. In Ψ (and $\mathcal{T}$), this is the case since any element X is the ℓ-image of at least one element; namely, $\top(\ell \Rightarrow X)$.

Effectuality of relative-simplication is the central result of this section. We now proceed through the technical details aimed at establishing its claim in the form of two theorems: Theorem 22 and Theorem 24.

4 Termination of relative simplication

For the purpose of showing that the relative simplication rules always terminate, we introduce an additional set of rules shown in Figure 5 extending basic simplication. These rules are *not* meant to be used in the effective operation of basic simplication, but only serve in our proof argument. The idea is that relative simplication of a guard ψ relatively to a context ϕ can be \simulated" by normalizing the formula $\phi\ \&\ \psi\ \&\ U \doteq X$ using basic simplication (Figure 1) together with the rules of Figure 5. It is not a real simulation, however, as Rules (B.1){(B.5) have for side effect to destroy the context. The point is that one application of a relative simplication rule can be made to correspond to at least one application of one of Rules (B.1){(B.5), (X.1){(X.3). Since this latter system can be shown to terminate, then so can relative simplication.

Rules (X.1){(X.3) perform essentially the same work as Rules (B.1) and (B.2) except that they do no erase parts of the formula. In Rule (X.1), we denote by $\sim_{\doteq}$

the reexive, symmetric and transitive closure of $\doteq$ (that is, the equivalence relation on the variables occurring in the constraint which is generated by the $\doteq$-pairs between variables in the constraint).

Extended Feature Decomposition:

$$\text{(X.1)}\ \frac{\psi \ \&\ U.\ell \doteq U' \ \&\ U.\ell \doteq U''}{\psi \ \&\ U.\ell \doteq U' \ \&\ U.\ell \doteq U'' \ \&\ U'' \doteq U'} \quad \textit{if } U' \not\sim_{\doteq} U''$$

Extended Sort Intersection 1:

$$\text{(X.2)}\ \frac{\psi \ \&\ U : s \ \&\ U : s'}{\psi \ \&\ U : s \ \&\ U : s \wedge s'} \qquad \begin{array}{l}\textit{if } s \wedge s' < s'' \textit{ for any } s'' \\ \textit{such that } U : s'' \in \psi\end{array}$$

Extended Sort Intersection 2:

$$\text{(X.3)}\ \frac{\psi \ \&\ U : s \ \&\ U : s'}{\psi \ \&\ U : s \ \&\ U : s' \ \&\ U : s \wedge s'} \qquad \begin{array}{l}\textit{if } s \wedge s' < s'' \textit{ for any } s'' \\ \textit{such that } U : s'' \in \psi\end{array}$$

Fig. 5. Rules extending basic simplication

Lemma 13. *The extended basic-simplication rules (B.1){(B.5), (X.1){(X.3) dene equivalence transformations; furthermore, they are terminating.*

Proof. The rst statement is clear. The proof of the second statement is an extension of the termination proof of the basic simplication rules (B.1){(B.5) from [5]: (X.1) can be applied only a nite number of times, since the number of equivalence classes partitioning the nite set of variables occurring in the constraint which is to be simplied decreases by 1 with each application. (X.2) and (X.3) can be applied only a nite number of times, since they can be applied at most once for every sort occurring in the constraint which is to be simplied.

Lemma 14. *Let $\psi \ \&\ U \doteq X$ simplify to ψ' relatively to ϕ by a relative-simplication step not using Rule (F.4). Then, $\phi \ \&\ \psi \ \&\ X \doteq U$ simplies to $\phi' \ \&\ \psi''$ by at most one extended basic-simplication step and a nite number of variable elimination (B.3), where ψ' and ψ'' are equal up to variable renaming.*

Proof. It can be seen that each relative simplication rule, except for (F.4), corresponds to one or several extended basic-simplication rules. Rules (F.1){(F.3) correspond to Rules (B.1) and (X.1). Rules (S.1){(S.4) correspond to Rules (B.2), (X.2) and (X.3).

Rules (E.1){(E.2) correspond to Rule (B.3). This, and the fact that extended basic-simplication rules are equivalence transformations, allow us to conclude.

Lemma 15. *Let ψ simplify to ψ' of the form ψ & $U_1 \doteq X_1$ & $U_1 \doteq X_2$ by an application of Rule (F.4) relatively to ϕ. Then, ψ & $U_1 \doteq X_1$ simplies to the same constraint ψ' by an application of Rule (F.3) relatively to ϕ.*

Proposition 16. *The relative-simplication rules are terminating.*

Proof. This is proved by induction on n, using Lemma 14 and Lemma 15. For every relative-simplication chain ψ_1 & $U_1 \doteq X_1, \ldots, \psi_n$ & $U_n \doteq X_n$ relatively to ϕ, there exists an extended-basic simplication chain of length $n + k$, where $k \geq 0$. This chain starts with the basic constraint ϕ & ψ & $X_1 \doteq U_1$ & $X \doteq U$, where $X \doteq U$ stands for the equations we have added so that each global variable X is bound to some local variable U (which, if necessary, is chosen new).

Since, according to Lemma 13, extended-basic-simplication chains are nite, so are relative-simplication chains.

5 Correctness and completeness

We rst note another consequence of the lemmata of the last section. Let $\mathcal{V}$ stand for the new local variables introduced by Rule (F.4).

Proposition 17. *Let ψ & $U \doteq X$ simplify to ψ' relatively to ϕ. Then, ϕ & ψ & $U \doteq X$ and $\exists\mathcal{V}.\ (\phi$ & $\psi')$ are equivalent.*

Proof. Let us rst assume that ψ & $U \doteq X$ simplies to ψ' relatively to ϕ, not using Rule (F.4). Then, ϕ & ψ & $U \doteq X$ and ϕ & ψ' are equivalent by Lemma 13 and Lemma 14. Let ψ & $U \doteq X$ simplify to ψ & $U \doteq X$ & $V \doteq X_1$ & $V \doteq X_2$ relatively to ϕ, by an application of Rule (F.4). Clearly, ϕ & ψ & $U \doteq X$ and ϕ & $\exists V.\ (\psi$ & $U \doteq X$ & $V \doteq X_1)$ are equivalent. Thus, with Lemma 15, we can apply the rst part of the proof on ψ & $U \doteq X$ & $V \doteq X_1$.

The next corollary states a property which is important for showing that relative simplication can be used for proving entailment, the *invariance property*.

Corollary 18 Invariance of Relative-Simplication. *If ψ & $U \doteq X$ simplies to ψ' relatively to ϕ, then $\exists\mathcal{U}.\ (\phi$ & ψ & $U \doteq X)$ and $\exists\mathcal{U}\exists\mathcal{V}.\ (\phi$ & $\psi')$ are equivalent.*

It is helpful to list systematically the normal-form properties of the relative-simplication system.

Proposition 19. *The constraint ψ is in normal form relatively to ϕ iff the following conditions are satised:*

- *ψ is in solved-form;*
- *a global variable X may occur in ψ only in the form $\_ \doteq X$;*
- *if $X \doteq \_ \in \phi$, then X does not occur in ψ;*

- *if* $V \doteq X \in \psi$, *and* $\_ \doteq X.\ell \in \phi$, *then* $\_ \doteq V.\ell \notin \psi$;
- *if* $V \doteq X \in \psi$, *and* $X : s \in \phi$, *and* $V : s' \in \psi$, *then* $s' < s$;
- *if* $\{V \doteq X, V \doteq Y\} \subseteq \psi$, *and* $\{X' \doteq X.\ell, Y' \doteq Y.\ell\} \subseteq \phi$, *then* $\{W \doteq X', W \doteq Y'\} \subseteq \psi$, *for some variable* W;
- *if* $\{V \doteq X, V \doteq Y\} \subseteq \psi$, *and* $\{X : s_1, Y : s_2\} \subseteq \phi$, *then* $V : s \in \psi$ *for some sort* s *such that* $s \leq s_1$ *and* $s \leq s_2$.

Proof. By inspection of the relative-simplication rules.

Proposition 20. *Let* ψ' *be a normal form of* ψ & $U \doteq X$ *relatively to* ϕ. *Let* ϕ' *be the constraint obtained from* ϕ *eliminating all redundancies according to the rules of Figure 6, and removing bindings* $V \doteq \_$ *of new variables introduced by (F.4). Then, the constraint* ϕ' & ψ' *is a solved-form of the constraint* ϕ & ψ & $U \doteq X$, *up to variable renaming.*

Redundant Sort Elimination:

$$(\mathrm{R.1})\ \frac{\phi \ \&\ X : s}{\phi} \qquad \textit{if } U \doteq X \in \psi, \textit{ and } U : s' \in \psi \textit{ for some } s' \leq s$$

Redundant Feature Elimination:

$$(\mathrm{R.2})\ \frac{\phi \ \&\ X_1' \doteq X_1.\ell \ \&\ X_2' \doteq X_2.\ell}{\phi \ \&\ X_1' \doteq X_1.\ell} \ \textit{if } U \doteq X_1 \in \psi, U \doteq X_2 \in \psi$$

Entailed Sort Redundancy Elimination:

$$(\mathrm{R.3})\ \frac{\phi \ \&\ X_1 : s \ \&\ X_2 : s}{\phi \ \&\ X_1 : s} \qquad \textit{if } U \doteq X_1 \in \psi, U \doteq X_2 \in \psi$$

Fig. 6. Redundancy elimination rules

Proof. According to Proposition 17, ϕ & ψ & $U \doteq X$ is equivalent to $\exists \mathcal{V}.\ \phi$ & ψ', where $\mathcal{V}$ stands for the new variables. According to the last three conditions of Proposition 19, Rules (R.1), (R.2) or (R.3) perform equivalence transformations. Thus, if applications of these rules modify ϕ' to ϕ'', then ϕ' & ψ' is equivalent to ϕ'' & ψ'.

According to the rst four conditions of Proposition 19, ϕ'' & ψ' is in solved-form up to variable eliminations via Rule (B.3). More precisely, these variable eliminations are applications of Rule (B.3) using new equations of the form $V \doteq X$ introduced by Rule (F.4). They produce possibly equations of the form $X \doteq Y$ between global variables; then, further variable eliminations consist of applications of Rule (B.3) using these new equations. As a last step, these new equations are removed in order to obtain a constraint which is exactly equivalent to ϕ & ψ & $U \doteq X$, and not just up to existential quantication of new variables.

Corollary 21. *If the normal form of ψ & $U \doteq X$ relatively to ϕ is not $-$, then ϕ & ψ & $U \doteq X$ is satisable.*

Proof. In [5] we showed that a constraint is satisable if and only if it has a solved-form; that is, its basic normal form is different from $-$. The statement then follows from Proposition 20.

Theorem 22 Disentailment. *Let ψ' be a normal form of ψ & $U \doteq X$ relatively to ϕ. Then, ϕ disentails $\exists\mathcal{U}.\ (\psi$ & $U \doteq X)$ if and only if $\psi' = -$.*

Proof. If $\psi' = -$, then $\forall\mathcal{X}\ (\phi \rightarrow \neg\exists\mathcal{U}\exists\mathcal{V}.\ \psi')$ is valid. From Corollary 18, it follows that $\forall\mathcal{X}\ (\phi \rightarrow \neg\exists\mathcal{U}.\ \psi$ & $U \doteq X)$ is valid, too. If $\psi' \neq -$, then Corollary 21 can be applied.

Proposition 23. *If the normal form ψ' of ψ & $U \doteq X$ relatively to ϕ is not a conjunction of equations representing a functional binding, then ϕ & $\neg\exists\mathcal{U}.\ (\psi$ & $U \doteq X)$ is satisable.*

Proof. The assumption on the form of ψ' means that one of the three following cases is true, for some $V \in Var(\psi')$ bound to some $X \in Var(\phi)$; *i.e.*, $V \doteq X \in \psi'$.

1. ψ' contains a sort constraint on V; say, $V : s$; or,
2. ψ' contains two equations on V; say, $V \doteq X$ & $V \doteq Y$; or,
3. ψ' contains a feature constraint on V, say, $V.\ell \doteq W$.

For each case, we can nd a constraint ϕ' such that ϕ & ϕ' is satisable and disentails ψ'. Then, ϕ & ϕ' also disentails $\exists\mathcal{U}.\ (\psi$ & $U \doteq X)$; *i.e.*, ϕ & $\phi' \rightarrow \neg\exists U.\ (\psi$ & $U \doteq X)$ is valid. Clearly, this is sufcient to show that ϕ & $\neg\exists\mathcal{U}.\ (\psi$ & $U \doteq X)$ is satisable.

(1) $V : s \in \psi'$; then, according to the third condition of Proposition 19, ϕ contains either no sort constraint on X or one of the form $X : s'$ where $s < s'$. Thus, we set $\phi' = X : s''$, in the rst case, for some sort s'' incompatible with s; *i.e.*, such that $s \wedge s'' = -$. In the second case, we choose s'' such that $s \wedge s'' = -$ and $s'' \leq s'$.

(2) $V \doteq X$ & $V \doteq Y \in \psi'$; then, either $V : s \in \psi'$ and we are in Case (2), or, according to the last condition of Proposition 19, at most one of X and Y is sorted in ϕ. If $Y : s \in \phi$, we set $\phi' = X : s'$ for some sort s' such that $s \wedge s' = -$. If none of X and Y is sorted in ϕ, we set $\phi' = Y : s$ & $X : s'$ for some sorts s, s' such that $s \wedge s' = -$.

(3) $V.\ell_1 \doteq V_1 \in \psi'$; then, ϕ contains no feature constraint $X.\ell_1 \doteq \_$, according to the fourth condition of Proposition 19. Without loss of generality, we can assume that ψ does

not contain redundant conjuncts.[4] There exists a sort s such that ψ contains a conjunct of the form: $V.\ell_1 \doteq V_1$ & $V_1.\ell_2 \doteq V_2$ & $\dots$ & $V_{n-1}.\ell_n \doteq V_n$ & $V_n : s$, for some $n \geq 1$. Thus, we set $\phi' = X.\ell_1 \doteq X_1$ & $X_1.\ell_2 \doteq X_2$ & $\dots$ & $X_{n-1}.\ell_n \doteq X_n$ & $X_n : s'$, for some new variables $X_1, \dots, X_n$ and some sort s' such that $s \wedge s' = -$.

Theorem 24 Entailment. *Let ψ' be a normal form of ψ relatively to ϕ. Then, ϕ entails $\exists\mathcal{U}.\ (\psi\ \&\ U \doteq X)$ if and only if ψ' is a functional binding. Moreover, $\phi\ \&\ \psi'$ is a solved OSF constraint.*

Proof. If ψ' is a conjunction of equations representing a functional binding, then $\exists\mathcal{U}\exists\mathcal{V}.\ \psi'$ is valid; thus, so is $\phi \rightarrow \exists U\exists\mathcal{V}.\ \psi'$. By invariance of relative simplication (Corollary 18), it follows that $\phi \rightarrow \exists\mathcal{U}.\ \psi$ is valid, too.

If ψ' has a different form then, either $\psi' = -$, or ψ' contains conjuncts that are not a functional binding. The fact that $\phi \rightarrow \exists\mathcal{U}.\ \psi$ is not valid is trivial in the rst case. In the other case, since the context ϕ is always assumed in solved form and, thus, satisable, then it follows from Proposition 23.

Corollary 25. *Let ψ' be the relative-simplication normal form of $\psi\ \&\ U \doteq X$ relatively to ϕ. Then, the context entails the guard if and only if the conjunction $\phi\ \&\ \psi'$ is the solved-form of the conjunction $\phi\ \&\ \psi\ \&\ U \doteq X$.*

Proof. This is an immediate consequence of Theorem 24 and Proposition 20.

6 Independence

The following theorem states that the OSF constraint system has the independence property [10]. It is well-known that in any constraint system with this property it is possible to solve constraints which are conjunctions of constraints and negated constraints by testing entailment. Namely, $\phi\ \&\ \neg\exists\mathcal{U}_1\psi_1\ \&\ \dots\neg\exists\mathcal{U}_n\psi_n$ is satisable if and only if ϕ does not entail $\exists\mathcal{U}_i.\ \psi_i$, for every $i = 1, \dots, n$. Here $\exists\mathcal{U}_i$ abbreviates the existential quantication of variables in $Var(\psi_i) - Var(\phi)$.

Clearly, ϕ entails $\exists\mathcal{U}_i.\ \psi_i$ if and only if ϕ entails $\exists\mathcal{U}_i\exists U_i.\ \psi_i[U_i/X_i]\ \&\ U_i \doteq X_i$, where we introduce a new variable U_i for every $X_i \in Var(\phi) \cap Var(\psi_i)$. Hence, given that the independence property holds, we can use the relative-simplication algorithm in order to check satisability of conjunctions of positive and negative OSF constraints.

For the formulation of the theorem, let us make a few assumptions that do not incur any loss of generality. First, we assume that $\mathcal{U}_i = Var(\psi_i)$, $U_i \in \mathcal{U}_i$, and $Var(\phi) \cap Var(\psi_i) = \emptyset$. Second, since they correspond to different existential quantication scopes, we will assume $\mathcal{U}_i \cap \mathcal{U}_j = \emptyset$ for $i \neq j$. Finally, we again assume that ψ_i does not contain redundant constraints (*cf.*, Footnote 4).

[4] That is, we assume that every variable in ψ has at least one sort constraint and that redundant constraints in ψ are removed. A redundant constraint in ψ is one of the form $X.\ell \doteq Y\ \&\ Y : \top$ where Y does not occur elsewhere in ψ. Since we interpret features as total functions, this is not a proper restriction: redundant constraints can be moved into the functional expression or the body of the guarded clause without changing the declarative or the operational semantics. On the other hand, if this assumption is fullled, then the entailment of $\psi\ \&\ U \doteq X$ by ϕ does not depend on whether features are interpreted as total or partial functions.

Theorem 26 Independence. *A constraint ϕ entails the disjunction of the constraints $\exists\mathcal{U}_i.\,(\psi_i \;\&\; U_i \doteq X_i)$, for $i = 1, \ldots, k$, if and only if it entails one of them.*

Proof. The *if*-direction is trivial. It is sufcient to show that if $\phi \;\&\; \neg\exists\mathcal{U}_i.\,(\psi_i \;\&\; U_i \doteq X_i)$ is satisable for every i, then $\phi \;\&\; \bigwedge_{i=1,\ldots,k} \neg\exists\mathcal{U}_i.\,(\psi_i \;\&\; U_i \doteq X)$ is satisable.

Extending the proof technique of Proposition 23, we will nd a constraint ϕ' such that $\phi \;\&\; \phi'$ is satisable and disentails ψ_i', for all $i = 1, \ldots, k$. As a consequence, $\phi \;\&\; \phi'$ also disentails $\exists\mathcal{U}_i.\,(\psi_i \;\&\; U_i \doteq X_i)$. That is, $\phi \;\&\; \phi' \rightarrow \neg\exists\mathcal{U}_i.\,(\psi_i \;\&\; U_i \doteq X_i)$ is valid. Clearly, this shows that $\phi \;\&\; \bigwedge_{i=1,\ldots,k} \neg\exists\mathcal{U}_i.\,\psi_i \;\&\; U_i \doteq X$ is satisable.

According to Theorem 24, if $\phi \;\&\; \neg\exists\mathcal{U}_i.\,(\psi_i \;\&\; U_i \doteq X_i)$ is satisable, then ψ_i', the normal form of $\psi_i \;\&\; U_i \doteq X_i$ relatively to ϕ is not a conjunction of equations representing a functional binding.

Thus, one of the three following cases is true, for some $V_i \in Var(\psi_i')$ bound to some $X_i \in Var(\phi)$; *i.e.*, $V_i \doteq X_i \in \psi_i'$:

1. ψ_i' contains a sort constraint on V_i; say, $V_i : s_i$; or,
2. ψ_i' contains two equations on V_i; say, $V_i \doteq X_i \;\&\; V_i \doteq Y_i$;
3. ψ_i' contains a feature constraint on V_i, say, $V_i.\ell_i \doteq W_i$.

(1) If $V_i : s_i \in \psi_i'$, then ϕ contains either no sort constraint on X_i or one of the form $X_i : s_i'$ where $s_i < s_i'$, according to the third condition of Proposition 19. Let $U_{i_j} \doteq X_i$, for $i_j = 1, \ldots, m$, be the family of all equations occurring in the disjuncts binding a local variable U_{i_j} to that same global variable X_i. We add to ϕ the sort constraint $X_i : s_i''$ where s_i'' is some sort which is incompatible with those in the sort constraints $U_{i_j} : s_{i_j}$, and, in case $X_i : s_i' \in \phi$, is furthermore a subsort of s_i', $s_i'' \leq s_i'$.

(2) If $V_i \doteq X_i \;\&\; V_i \doteq Y_i \in \psi_i'$, and $V_i : s_i \notin \psi_i'$ (otherwise we are in Case (2)), then we add to ϕ' the conjuncts $X_i.\ell_i \doteq Z_i \;\&\; Z_i \in s \;\&\; Y_i.\ell_i \doteq Z_i' \;\&\; Z_i' \in s'$. Here s and s' are two incompatible sorts, and the ℓ_i's are pairwise different features which do not occur in ϕ and ψ_i, for $i = 1, \ldots, k$.

(3) Finally, we consider the set I of all indices i, $i = 1, \ldots, k$, for which Case (3), but neither Case (1) nor Case (2) applies. Thus, for $i \in I$, ψ_i' contains a feature constraint of the form $V_i.\ell_i \doteq V_i^1$. According to our assumption this constraint is not a redundant conjunct; *i.e.*, there exists a sort s_i such that ψ_i contains, in fact, a conjunct of the form:

$$V_i.\ell_i \doteq V_i^1 \;\&\; V_i^1.\ell_i^2 \doteq V_i^2 \;\&\; \ldots \;\&\; V_i^{n-1}.\ell_i^n \doteq V_i^n \;\&\; V_i^n : s_i,$$

for some $n \geq 1$. We add to ϕ' the conjunct:

$$X_i.\ell_i^1 \doteq X_i^1 \;\&\; X_i^1.\ell_i^2 \doteq X_i^2 \;\&\; \ldots \;\&\; X_i^{n-1}.\ell_i^n \doteq X_i^n \;\&\; X_i^n : s_i',$$

for some new variables $X_i^1, \ldots, X_i^n$ and for some sort s_i' incompatible with s_i.

If there are several disjuncts ψ_{i_j}' with exactly the same chain of feature constraints starting in a variable bound to the same global variable, then s_i' must be chosen to be incompatible with the sorts in all of these chains. More precisely, if, for $i_j = 1, \ldots, m$, the disjunct ψ_{i_j}' contains the conjunct:

$$V_{i_j}.\ell_i \doteq V_{i_j}^1 \;\&\; V_{i_j}^1.\ell_i^2 \doteq V_{i_j}^2 \;\&\; \ldots \;\&\; V_{i_j}^{n-1}.\ell_i^n \doteq V_{i_j}^n \;\&\; V_{i_j}^n : s_{i_j},$$

then s_i' is chosen as some sort such that $s_{i_j} \wedge s_i' = -$ for all i_j, $i_j = 1, \ldots, m$.

7 Conclusion

We have overviewed in detail a complete and correct system for deciding entailment and disentailment of constraints over order-sorted feature structures. One motivation for this system is parameter-passing for functions in LIFE, but it is general and relevant to all concurrent constraint languages. We used a technique of relative simplication [4] which amounts to normalizing a constraint in the context of another. This yields an incremental system with the additional benet of enjoying independence of negated constraints.

Further work extending this should be to generalize our scheme to so-called deep guards over OSF structures whereby guards are not limited to plain OSF constraints but may also contain relational atoms dened by clauses. This is particularly relevant to LIFE in order to explain matching over objects with attached relational constraints. This study in currently under way and will be reported soon.

References

1. Hassan Aït-Kaci. An algebraic semantics approach to the effective resolution of type equations. *Theoretical Computer Science*, 45:293{351 (1986).
2. Hassan Aït-Kaci and Roger Nasr. LOGIN: A logic programming language with built-in inheritance. *Journal of Logic Programming*, 3:185{215 (1986).
3. Hassan Aït-Kaci and Roger Nasr. Integrating logic and functional programming. *Lisp and Symbolic Computation*, 2:51{89 (1989).
4. Hassan Aït-Kaci and Andreas Podelski. Functions as passive constraints in LIFE. PRL Research Report 13, Digital Equipment Corporation, Paris Research Laboratory, Rueil-Malmaison, France (June 1991). (Revised, November 1992).
5. Hassan Aït-Kaci and Andreas Podelski. Towards a meaning of LIFE. PRL Research Report 11, Digital Equipment Corporation, Paris Research Laboratory, Rueil-Malmaison, France (1991). (Revised, October 1992; to appear in the *Journal of Logic Programming*).
6. Hassan Aït-Kaci, Andreas Podelski, and Gert Smolka. A feature-based constraint system for logic programming with entailment. In *Proceedings of the 5th International Conference on Fifth Generation Computer Systems*, pages 1012{1022, Tokyo, Japan (June 1992). ICOT. (Full paper to appear in *Theoretical Computer Science*).
7. Rolf Backofen and Gert Smolka. A complete and decidable feature theory. DFKI Research Report RR-30-92, German Research Center for Articial Intelligence, Saarbrücken, Germany (1992).
8. Bruno Courcelle. Fundamental properties of innite trees. *Theoretical Computer Science*, 25:95{169 (1983).
9. Seif Haridi and Sverker Janson. Kernel Andorra Prolog and its computation model. In David H. D. Warren and Peter Szeredi, editors, *Logic Programming, Proceedings of the 7th International Conference*, pages 31{46, Cambridge, MA (1990). MIT Press.
10. Jean-Louis Lassez, Michael Maher, and Kimball Mariott. Unication revisited. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, chapter 15, pages 587{625. Morgan-Kaufmann, Los Altos, CA (1988).
11. Michael Maher. Logic semantics for a class of committed-choice programs. In Jean-Louis Lassez, editor, *Logic Programming, Proceedings of the Fourth International Conference*, pages 858{876, Cambridge, MA (1987). MIT Press.

12. Vijay Saraswat and Martin Rinard. Concurrent constraint programming. In *Proceedings of the 7th Annual ACM Symposium on Principles of Programming Languages*, pages 232{245. ACM (January 1990).
13. Gert Smolka and Ralf Treinen. Records for logic programming. In Krzysztof Apt, editor, *Logic Programming, Proceedings of the Joint International Conference and Symposium on Logic Programming*, pages 240{254, Cambridge, MA (1992). MIT Press.