# Dominance Constraints in Context Unification

Joachim Niehren[1] and Alexander Koller[2]

[1] Programming Systems Lab, Universität des Saarlandes, Saarbrücken, Germany, `www.ps.uni-sb.de/~niehren`
[2] Department of Computational Linguistics, Universität des Saarlandes, Saarbrücken, Germany, `www.coli.uni-sb.de/~koller`

**Abstract.** Tree descriptions based on dominance constraints are popular in several areas of computational linguistics including syntax, semantics, and discourse. Tree descriptions in the language of context unification have attracted some interest in unification and rewriting theory. Recently, dominance constraints and context unification have both been used in different underspecified approaches to the semantics of scope, parallelism, and their interaction. This raises the question whether both description languages are related. In this paper, we show for a first time that dominance constraints can be expressed in context unification. We also prove that dominance constraints extended with parallelism constraints are equal in expressive power to context unification.

**Keywords.** Computational linguistics, underspecification, tree descriptions, computational logics, unification theory.

## 1 Introduction

Logical tree descriptions are popular in many areas of computational linguistics and computer science. They are used to model data structures in logic programming, to reason with propositions and proofs in automated deduction, and to represent all kinds of syntactic or semantic structures in computational linguistics. In this paper, we investigate the relationship between tree descriptions based on dominance constraints and those in the language of context unification.

*Two Languages of Tree Descriptions. Dominance constraints* are popular for describing trees throughout computational linguistic. In syntax, they serve for deterministic parsing [MHF83] and to combine TAG and unification grammars [VS92]. In underspecified treatments of scope ambiguities, variants of dominance constraints appear somewhat implicitly in many places [Rey93,Bos96] and explicitly in two recent approaches [ENRX98,Mus98]. Even more recently, dominance constraints have been applied to discourse semantics [GW98], and they have been used to model information growth and partiality [MVK98].
In general, the problem of solving dominance constraints is NP-complete [KNT98]. Nevertheless, [DG99] describes an implementation of a dominance constraint solver which runs efficiently on practical examples from scope underspecification and discourse. This solver is implemented based on finite set constraints in

the Mozart System [Moz99], the most recent implementation of the concurrent constraint programming language Oz [Smo95].

*Context unification (CU)* was introduced in rewriting and unification theory [Com92,SS94]. CU can be considered as second-order linear unification [Lév96], which is a restriction of higher-order unification, or as an extension of string unification [SSS98]. The decidability question for CU is a prominent open problem [RTA98]. A decidable fragment of CU called *stratified unification* has been used to show the decidability of distributive unification [SS97] and for solving one-step rewriting constraints [NPR97a,NTT99]. It is shown in [SSS99] that context unification with two context variables – each of which may occur an arbitrary number of times – is decidable. The proof is by reduction to string unification, which is decidable according to Makanin's famous result [Mak77,Sch93].

*Tree Descriptions in Semantic Underspecification.* Recently, tree descriptions based on dominance constraints and context unification have been proposed for the same application to natural-language semantics [ENRX98,NPR97a,Kol99]. There, the goal was to find a uniform language providing underspecified representations for the semantics of scope, parallelism, anaphora, and their interactions (for a survey of semantic underspecification, see e.g. [vP96]). The common characteristic of both approaches is that they view the formulae of the semantic representation as trees and describe these trees. The role of dominance constraints in this context is to describe scope ambiguities; they are extended with constructions for describing parallelism and anaphoric and variable binding to obtain the Constraint Language over Lambda Structures (CLLS).

*Contribution.* If CU and CLLS are used for the same application, an immediate question is if there is a formal relationship between the two languages that says something about their relative expressive power.

In this paper, we show that the fragment of CLLS which provides dominance and parallelism constraints is equal in expressive power to context unification. We do this by giving satisfiability preserving, polynomial time encodings in both directions. The most interesting (and non-obvious) part of the construction is to encode dominance constraints in context unification. Once we know how to do that, the rest of this direction is easy. The inverse encoding can be deduced from a result in [NPR97a].

*Plan of the Paper.* In Section 2 we illustrate why encoding dominance constraints into context unification is nontrivial. In Section 3, we recall the fundamental definitions of trees and contexts. These definitions are used in Section 4, where we present dominance and parallelism constraints and briefly review a linguistic example. They are also used in Section 5, where we recall context unification, discuss first results on its expressive power, and give a linguistic example, too. Section 6 contains the encoding of dominance and parallelism constraints in CU, and Section 7 the inverse encodings. We conclude in Section 8.

## 2 What is the Problem?

It is not obvious to encode dominance constraints in context unification. The problem is that both languages describe trees from different perspectives. We now illustrate the difference by an example.

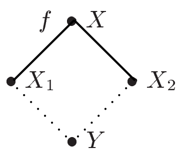
*Two Perspectives on Trees.* Dominance constraints (and CLLS as a whole) describe relations between the *nodes* of the same tree (or a more general $\lambda$-structure). In contrast, the language of CU models relations between different *trees* and *contexts.* In CU, one cannot speak directly about the nodes of a tree; but we shall use contexts to speak about occurrences of subtrees later in this paper.
The perspective taken when speaking about the nodes of the same tree is called *internal* in [BGMV93], in contrast to the *external* perspective where one relates several trees. Both views have a long tradition in logics. The internal view is taken in modal logic and in second-order monadic logic ($SnS$) [Rab69], whereas the external view is popular in unification theory [MM82,Col84,BS93] and for set constraints [HJ90,AW92,TDT00]. In feature logics [KR86,Smo92], both perspectives have been employed and compared [BS95,MN00].

*Dominance versus Subtree Constraints.* Dominance constraints contain node valued variables that we write as capital letters $X, Y, Z$. An atomic dominance constraint $X \lhd^* Y$ holds in a tree (structure) if the node denoted by $X$ is above (strictly or not) the node denoted by $Y$.
A first idea for encoding dominance constraints in CU is to replace each atomic dominance constraint by a *subtree constraint* [Ven87], which can be expressed in CU in a very simple way. Subtree constraints have tree valued variables for which we use lower case letters $x, y, z$. A subtree constraint $x \gg y$ says that the denotation of $y$ is a subtree of the denotation of $x$.
Although they look very similar, there is an important difference between dominance and subtree constraints: Dominance constraints can speak about *occurrences* of subtrees by specifying their root nodes, whereas subtree constraints can't.

*An Example.* Because of this difference, the naive encoding of dominance as subtree constraints does not preserve satisfiability. As an example, we consider the dominance constraint in (1) and the "corresponding" subtree constraint (2).

(1) $\quad X{:}f(X_1, X_2) \wedge X_1 \lhd^* Y \wedge X_2 \lhd^* Y \quad =$

(2) $\quad x{=}f(x_1, x_2) \;\wedge\; x_1 \gg y \;\wedge\; x_2 \gg y \quad \neq$



The dominance constraint (1) is depicted by the graph to the right. It describes trees in which the node denoted by $X$ is labeled with a binary function symbol $f$ and has two (distinct) children denoted by $X_1$ and $X_2$. Furthermore, it requires that there is a node, denoted by $Y$, which is below $X_1$ and $X_2$. This is impossible in a tree. Thus, (1) is unsatisfiable.

The subtree constraint (2) requires that $x$, $x_1$, $x_2$, and $y$ denote trees. The tree for $x$ has two direct subtrees denoted by $x_1$ and $x_2$, which in turn have a common subtree $y$ (not necessarily at the same position). The subtree constraint (2) is satisfiable; one solution is obtained by mapping $y$, $x_1$, and $x_2$ to the tree $a$, and $x$ to the tree $f(a, a)$. The two occurrences of $y$ in the subtree constraint (2) refer to different occurrences the tree $a$ in $f(a, a)$.

## 3 Trees and Contexts

Understanding the notions of *trees* and *contexts* is essential for this paper. We next define both notions and explain the views on them we will adopt.

We assume a signature $\Sigma$ of *function symbols* ranged over by $f, g$, each of which is equipped with a fixed arity $\mathsf{ar}(f) \geq 0$. Constants, ranged over by $a, b$, are function symbols with arity 0. We assume that $\Sigma$ contains at least two function symbols, one of which is not constant. Note that we do not restrict our signature to be finite.

*Trees.* A (finite constructor) *tree* $\tau$ is a ground term constructed from function symbols in $\Sigma$. For instance, $f(f(a, b), c)$ is a tree whose root node is labeled with $f$ and which has three leaves labeled by $a, b, c$.

An equivalent definition of trees, which makes the nodes and node labels of the tree explicit, is based on *tree domains.* Let $\mathbb{N}$ be the set of natural numbers $n \geq 1$ and $\mathbb{N}^*$ the set of words over natural numbers. $\epsilon$ is the *empty word*, and the *concatenation* of two words $\pi$ and $\pi'$ is written by juxtaposition $\pi\pi'$. A path $\pi'$ is a *prefix* of $\pi$ if there is a $\pi''$ such that $\pi'\pi'' = \pi$.

A *tree domain* $D$ is a nonempty prefixed-closed subset of $\mathbb{N}^*$. That is, $D$ contains *paths* which are words of positive integers; they can intuitively be identified with the nodes of the tree. A *labeling function* is a function $L : D \to \Sigma$ defined on a tree domain $D$ which satisfies for all $\pi \in D$ and $k \in \mathbb{N}$: $\pi k \in D$ iff $1 \leq k \leq \mathsf{ar}(L(\pi))$. A tree, then, is a pair $(D, L)$ of a tree domain and a labeling function.

The two definitions of trees can be connected by associating with each tree $\tau$ a tree domain $D_\tau$ and a labeling function $L_\tau : D_\tau \to \Sigma$ as follows:

$$
\begin{aligned}
D_{f(\tau_1,\dots,\tau_n)} &= \{\epsilon\} \cup \{k\pi \mid 1 \leq k \leq n,\ \pi \in D_{\tau_k}\} \\
L_{f(\tau_1,\dots,\tau_n)}(\pi) &= \begin{cases} f & \text{if } \pi = \epsilon \\ L_{\tau_k}(\pi') & \text{if } \pi = k\pi',\ 1 \leq k \leq n,\ \pi' \in D_{\tau_k} \end{cases}
\end{aligned}
$$

For instance, the tree $\tau = f(g(a), b)$ has the tree domain $D_\tau = \{\epsilon, 1, 11, 2\}$ and the labeling function $L_\tau$ with $L_\tau(\epsilon) = f$, $L_\tau(1) = g$, $L_\tau(11) = a$, and $L_\tau(2) = b$.

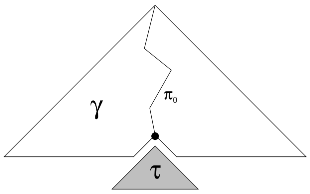



**Fig. 1.** A context $\gamma$ with hole $\pi_0$.

**Lemma 1.** *For every finite tree domain $D$ and labeling function $L : D \to \Sigma$ there exists a unique tree $\tau$ such that $D_\tau = D$ and $L_\tau = L$.*

Whenever $\tau$ is a tree and $\pi$ a path in $D_\tau$, we define the *subtree* $\tau.\pi$ of $\tau$ at $\pi$ as the unique tree with the following properties (otherwise $\tau.\pi$ is undefined):

$$\begin{aligned} D_{\tau.\pi} &= \{\pi' \mid \pi\pi' \in D_\tau\} \\ L_{\tau.\pi}(\pi') &= L_\tau(\pi\pi') \qquad \text{for all } \pi\pi' \in D_\tau \end{aligned}$$

**Lemma 2.** *For all trees $\tau$ and paths $\pi \in D_\tau$ if $f = L_\tau(\pi)$ and $\mathsf{ar}(f) = n$ then $\tau.\pi = f(\tau.(\pi 1), \ldots, \tau.(\pi n))$ .*

*Contexts.* Intuitively, a context is a tree with a hole. More formally, we introduce a special symbol $\bullet$ that we call *hole marker* and assign it the arity $\mathsf{ar}(\bullet) = 0$. A *context* $\gamma$ is a ground term over $\Sigma \cup \{\bullet\}$ which contains exactly one occurrence of the hole marker. For instance, $f(a, f(\bullet, b))$ is a context, but $f(\bullet, f(\bullet, b))$ isn't. We shall use the letter $\tau$ for trees over $\Sigma$ and the letter $\gamma$ for contexts (i.e. special trees over $\Sigma \cup \{\bullet\}$).
The *hole* of a context $\gamma$ is the occurrence of the hole marker in $\gamma$. More precisely, the hole is the unique path $\pi_0 \in D_\gamma$ such that $L_\gamma(\pi_0) = \bullet$. Fig. 1 shows a context with hole $\pi_0$.
We will freely consider contexts as functions that map trees to trees. Application $\gamma[\tau]$ of a context $\gamma$ to a tree $\tau$ is defined by

$$\gamma[\tau] = \gamma[\tau/\bullet]$$

That is, $\gamma[\tau]$ is the result of substituting the hole marker $\bullet$ in $\gamma$ by $\tau$. The context $\bullet$ corresponds to the identity function on trees. This illustrates that the hole marker can be seen as a $\lambda$-bound variable (rather than a constant or a free variable). Concatenation $\gamma \circ \gamma'$ of contexts seen as functions can be defined as $\gamma[\gamma'/\bullet]$.

**Lemma 3.** *For a context $\gamma$ with hole $\pi$ and all trees $\tau$, it holds that $\gamma[\tau].\pi = \tau$.*

*Contexts in Trees.* Since contexts are ground terms over a special signature, we have already defined subtree selection for contexts. If $\gamma$ is a context and $\pi \in D_\gamma$ then $\gamma.\pi$ is either a tree over $\Sigma$ or a context. It is a context iff $\pi$ is a prefix (proper or not) of the hole of $\gamma$.
Given a tree $\tau$ and a path $\pi \in D_\tau$, we write the context obtained by replacing the subtree of $\tau$ at $\pi$ with the symbol $\bullet$ as $\tau^\bullet\pi$. More precisely, $\tau^\bullet\pi$ is defined as the context with domain $D_{\tau^\bullet\pi} = \{\pi' \mid \pi \text{ not a proper prefix of } \pi'\}$ and the labeling function which assigns $L_{\tau^\bullet\pi}(\pi') = L_\tau(\pi')$ for all $\pi' \in D_{\tau^\bullet\pi} \setminus \{\pi\}$ and $L_{\tau^\bullet\pi}(\pi) = \bullet$.

**Lemma 4.** *For all $\tau$ and $\pi \in D_\tau$ it holds that $\tau^\bullet\pi[\tau.\pi] = \tau$.*

Given a prefix $\pi_1$ of $\pi_2$ and a tree $\tau$ with $\pi_2 \in D_\tau$, we define $\tau^{\pi_1}_{\pi_2}$ to be the *context of $\tau$ between $\pi_1$ and $\pi_2$*:

$$\tau^{\pi_1}_{\pi_2} = (\tau^\bullet\pi_2).\pi_1 = (\tau.\pi_1)^\bullet\pi \qquad \text{where } \pi_1\pi = \pi_2.$$

# 4 Dominance and Parallelism Constraints

We now present the language of dominance and parallelism constraints which is a fragment of the constraint language over $\lambda$-structures CLLS [ENRX98]. CLLS also has constructs for dealing with variable binding and anaphora, but we ignore these for the purpose of this paper.
Our notion of dominance constraints differs slightly from the one used e.g. by Vijay-Shanker [VS92]; these languages are mostly based on feature trees as common in computational linguistics, whereas our trees are *constructor trees*.

## 4.1 Tree Structures

We first define *tree structures*, logical structures representing trees. Tree structures fix the interpretation of a set of predicate symbols. Based on tree structures, we will define the syntax and semantics of our constraint language in the usual Tarskian style.
We associate with every tree $\tau$ a logical structure $\mathcal{M}^\tau$, the *tree structure of* $\tau$. The domain of the *tree structure* $\mathcal{M}^\tau$ coincides with the tree domain of $\tau$. Furthermore, $\mathcal{M}^\tau$ provides interpretations for the binary relation symbol $\lhd^*$, a 4-ary relation symbol $./.{\sim}./.$, and a relation symbol $:f$ of arity $\mathsf{ar}(f)+1$ for every function symbol $f \in \Sigma$. We use the same symbols for relations and relation symbols; there shouldn't be any danger of confusion. For instance, we write $\pi\lhd^*\pi'$ in order to say that the relation $\lhd^*$ holds for the pair $(\pi, \pi')$, whereas $X\lhd^*X'$ is an atomic constraint built from the relation symbol $\lhd^*$ and variables $X, X'$. A relation symbol is generally interpreted by the relation of the same name.
If $f \in \Sigma$ and $\mathsf{ar}(f) = n$, then the *labeling relation* $\pi{:}f(\pi_1, \dots, \pi_n)$ is true in $\mathcal{M}^\tau$ iff $L_\tau(\pi) = f$ and $\pi_i = \pi i$ for all $1 \leq i \leq n$. The *dominance relation* $\pi\lhd^*\pi'$ is

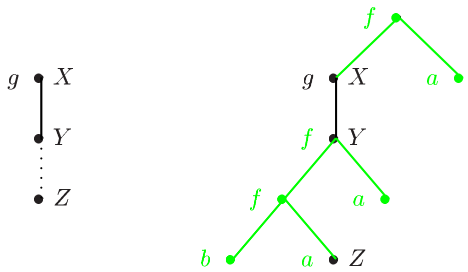



**Fig. 2.** The dominance constraint $X{:}g(Y) \wedge Y \lhd^* Z$ and of one of its solutions.

true in $\mathcal{M}^\tau$ iff $\pi, \pi' \in D_\tau$ and $\pi$ is a prefix of $\pi'$. Finally, the *parallelism relation* $\pi_1/\pi_1'{\sim}\pi_2/\pi_2'$ holds if the contexts $\tau_{\pi_1}^{\pi_1'}$ and $\tau_{\pi_2}^{\pi_2'}$ exist and coincide:

$$\pi_1/\pi_1'{\sim}\pi_2/\pi_2' \text{ holds in } \mathcal{M}^\tau \text{ iff } \pi_1 \lhd^* \pi_1',\ \pi_2 \lhd^* \pi_2', \text{ and } \tau_{\pi_1'}^{\pi_1} = \tau_{\pi_2'}^{\pi_2}.$$

Intuitively, this means the subtrees of $\tau$ below $\pi_1$ and $\pi_2$ have the same structure, except for the subtrees below $\pi_1'$ and $\pi_2'$, which may be different.

### 4.2 The Constraint Language

We assume an infinite set of *node variables* $X, Y, Z$. A *parallelism constraint* $\varphi$ is given by the following abstract syntax:

$$\varphi ::= X \lhd^* Y \mid X{:}f(X_1, \ldots, X_n) \mid X/X'{\sim}Y/Y' \mid \varphi \wedge \varphi'$$

A parallelism constraint is a conjunction of atomic constraints for the dominance, labeling, and parallelism relations. A *dominance constraint* is parallelism constraint without atomic constraints $X/X'{\sim}Y/Y'$ for parallelism.

The semantics of parallelism constraints is given by interpretation over arbitrary tree structures $\mathcal{M}^\tau$. A *solution* of a parallelism constraint $\varphi$ consists of a tree structure $\mathcal{M}^\tau$ and a variable assignment $\alpha$ into its domain that satisfies all atomic constraints in $\varphi$. We write $(\mathcal{M}^\tau, \alpha) \models \varphi$ if $(\mathcal{M}^\tau, \alpha)$ is a solution of $\varphi$.

Note that the constraint $X{:}a \wedge Y{:}a$ has solutions where $X$ and $Y$ denote distinct nodes both of which are labeled with $a$.

We often display a dominance constraint and its solutions graphically. For instance, the constraint $X{:}g(Y) \wedge Y \lhd^* Z$ and one of its solutions are displayed in Fig. 2. Note that additional material (printed in light gray) has been filled into the space between the nodes denoted by $Y$ and $Z$ and above $X$. The dominance constraint does not say anything about these regions.

The careful reader might have noticed that dominance constraints can be expressed by parallelism constraints without atomic constraints for dominance.

**Lemma 5.** *The equivalence $X \lhd^* Y \leftrightarrow X/Y{\sim}X/Y$ is valid in all tree structures.*

$$
\begin{aligned}
&X_1{:}\forall u(X_3) \wedge X_3{:}{\rightarrow}(X_4, X_5) \wedge \\
&X_4{:}\mathsf{man}(X_6) \wedge X_6{:}\mathsf{var}_u \wedge \\
&X_2{:}\exists v(X_7) \wedge X_7{:} \wedge (X_8, X_9) \wedge \\
&X_8{:}\mathsf{woman}(X_{10}) \wedge X_{10}{:}\mathsf{var}_v \wedge \\
&X_5 \lhd^* X_{11} \wedge X_9 \lhd^* X_{11} \wedge \\
&X_{11}{:}\mathsf{love}(X_{12}, X_{13}) \wedge \\
&X_{12}{:}\mathsf{var}_u \wedge X_{13}{:}\mathsf{var}_v
\end{aligned}
$$

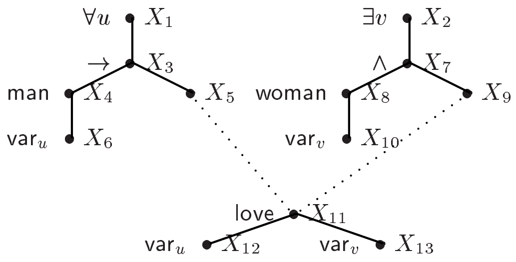



**Fig. 3.** An underspecified representation of the meaning of Example 3.

### 4.3 Application to Semantic Underspecification

As examples for the linguistic application of dominance and parallelism constraints, we briefly review a scope ambiguity and a very simple VP ellipsis. For the first example, consider the sentence (3), which is a classical scope ambiguity.

(3) Every man likes a woman.

The readings of this sentence can be represented by the predicate logic formulae in (4) and (5).

(4) $\forall u.(man(u) \rightarrow \exists v.(woman(v) \wedge love(u, v)))$
(5) $\exists v.(woman(v) \wedge \forall u.(man(u) \rightarrow love(u, v)))$

A compact underspecified representation of both readings is given by the dominance constraint in Fig. 3. The semantic representation of the sentence is considered as a tree, which is then described by a dominance constraint.
Ellipses can be modeled with parallelism constraints expressing that the trees corresponding to the semantics of source and target sentences must be the same except for the respective parallel elements. For instance, the semantics of (6) can be described by (7).

(6) John sleeps. Mary does too.
(7) $X{:}\mathsf{sleep}(X') \wedge X'{:}\mathsf{john} \wedge Y'{:}\mathsf{mary} \wedge X/X' {\sim} Y/Y'$

We cannot go into this in more detail here and refer the reader to [ENRX98] for an in-depth discussion (in particular on the interaction of scope and ellipses).

## 5 Context Unification

Context unification is the problem of solving equations between tree valued terms in the two-sorted algebra $\mathcal{TC}$ of trees and contexts. We first introduce

$$
\begin{array}{ll}
 & x \mapsto g(f(f(b,a),a)), \\
x{=}g(y) & y \mapsto f(f(b,a),a), \\
\wedge\ y{=}C(z) & C \mapsto f(f(b,\bullet),a), \\
 & z \mapsto a.
\end{array}
$$

**Fig. 4.** The equation system $x{=}g(y) \wedge y{=}C(z)$ and one of its solutions.

equations between tree-valued terms and then show that they can also express equations between context-valued terms. Finally, we sketch an application to semantic underspecification.

### 5.1 Syntax and Semantics of CU

The *algebra of trees and contexts* $\mathcal{TC}$ over $\Sigma$ is a two-sorted algebra whose domains are the set of trees and the set of contexts over $\Sigma$. The operations provided by $\mathcal{TC}$ are tree construction and functional application of contexts to trees. Each function symbol $f \in \Sigma$ is interpreted as an $\mathsf{ar}(f)$-ary tree constructor, which maps a tuple $(\tau_1, \ldots, \tau_n)$ of trees to the tree $f(\tau_1, \ldots, \tau_n)$. The application $\gamma[\tau]$ of a context $\gamma$ to a tree $\tau$ has already been defined.
For both sorts of $\mathcal{TC}$, we assume an infinite set of variables: *tree variables* $x, y, z$ and *context variables* $C$. A *tree-valued term* $t$ is built from tree variables, applications of function symbols in $\Sigma$, and application of context variables.

$$t \ ::= \ x \mid f(t_1, \ldots, t_n) \mid C(t) \qquad (\mathsf{ar}(f) = n)$$

In particular, every tree is a tree-valued term.
A *variable assignment* into $\mathcal{TC}$ is a function $\beta$ that assign trees to tree variables and contexts to context variables. Variable assignments can be lifted homomorphically to tree-valued terms:

$$\begin{aligned}
\beta(f(t_1, \ldots, t_n)) &= f(\beta(t_1), \ldots, \beta(t_n)) \\
\beta(C(t)) &= \beta(C)[\beta(t)].
\end{aligned}$$

A variable assignment $\beta$ into $\mathcal{TC}$ is a *solution* of an equation system (i.e. a conjunction of equations between terms) if $\beta(t) = \beta(t')$ holds for all equations $t = t'$ in this system. *Context unification* is the problem of solving such equation systems over $\mathcal{TC}$. An example for a solution of the equation system $x{=}g(y) \wedge y{=}C(z)$ is given in Fig. 4. The similarity between Figures 2 and 4 is intended.

### 5.2 Properties of Contexts

The following three lemmas are quite simple, but will facilitate a lot of later work.

**Lemma 6.** *Two contexts $\gamma$ and $\gamma'$ are equal iff their holes are the same and there is a tree $\tau$ such that $\gamma[\tau] = \gamma'[\tau]$.*

Note that the existence of a single tree $\tau$ such that $\gamma[\tau] = \gamma'[\tau]$ is sufficient.

*Proof.* The "$\Rightarrow$" direction is trivial. For the other direction, all we have to prove is that the domains of the contexts $\gamma$ and $\gamma'$ are equal; this immediately implies equality of the labeling functions, since for every $\pi \in D_\gamma$ except for the (common) hole, $L_\gamma(\pi) = L_{\gamma[\tau]}(\pi) = L_{\gamma'[\tau]}(\pi) = L_{\gamma'}(\pi)$.
Let's say that the common hole of $\gamma$ and $\gamma'$ is $\pi_0$. Then $\gamma[\tau] = \gamma'[\tau]$ implies the following equalities:

$$D_\gamma \cup \pi_0\ D_\tau = D_{\gamma[\tau]} = D_{\gamma'[\tau]} = D_{\gamma'} \cup \pi_0\ D_\tau .$$

As the unions on both sides are between sets whose respective intersection is $\{\pi_0\}$, it follows that $D_\gamma = D_{\gamma'}$. □

We next express a correspondence between nodes and their contexts.

**Lemma 7.** *Let $\tau$ be a tree and $\pi_1$ a prefix of $\pi_2$ with $\pi_1, \pi_2 \in D_\tau$. Then $\tau^{\pi_1}_{\pi_2}$ is the unique context such that* $\tau.\pi_1 = \tau^{\pi_1}_{\pi_2}[\tau.\pi_2]$ .

*Proof.* From Lemma 4 it follows that $\tau.\pi_1 = \tau^{\pi_1}_{\pi_2}[\tau.\pi_2]$. The uniqueness of $\tau^{\pi_1}_{\pi_2}$ follows from Lemma 6. □

**Lemma 8.** *Let $\pi_1$ be a prefix of $\pi_2$, $\pi_2$ a prefix of $\pi_3$, and $\tau$ a tree whose domain contains $\pi_1$, $\pi_2$, and $\pi_3$. Then* $\tau^{\pi_1}_{\pi_2} \circ \tau^{\pi_2}_{\pi_3} = \tau^{\pi_1}_{\pi_3}$.

*Proof.* Straightforward. □

### 5.3 Equations between Context-Valued Terms

In the construction in the next section, it will be convenient to use equations between context-valued terms, such as $C = C_1 \circ C_2$. This notation emphasizes the functional character of contexts. In this section, we show that these equations can in fact be expressed by equations between tree-valued terms. A *context-valued term* $u$ has the following abstract syntax:

$$u ::= C \mid \bullet \mid f(t_1, \ldots, t_i, u, t_{i+1} \ldots, t_n) \mid u \circ u'$$

We conservatively extend $\mathcal{TC}$ by concatenation $\gamma \circ \gamma'$ of contexts and lift variable assignments $\beta$ to context-valued terms as follows. As above, we define that $\beta$ is a *solution* of an equation $u{=}u'$ iff it maps $u$ and $u'$ to the same context.

$$\begin{aligned} \beta(\bullet) &= \bullet \\ \beta(f(t_1, \ldots, u, \ldots, t_n)) &= f(\beta(t_1), \ldots, \beta(u), \ldots, \beta(t_n)) \\ \beta(u \circ u') &= \beta(u) \circ \beta(u') \end{aligned}$$

Now we can define syntactic insertion $u[t]$ of tree-valued into context-valued terms in the obvious way. This produces tree-valued terms with the property $\beta(u[t]) = \beta(u)[\beta(t)]$. With this operation, we can express each equation between context-valued terms as a conjunction of equations between tree-valued terms.

**Proposition 1 (Equations between context valued terms).** *Let $\tau_1$ and $\tau_2$ be two different trees. Then the following equivalence holds:*

$$u{=}u' \quad \leftrightarrow \quad u[\tau_1]{=}u'[\tau_1] \wedge u[\tau_2]{=}u'[\tau_2].$$

Note that our restriction on the signature in Section 3 implies that two different trees really exist.

*Proof.* The direction from left to right is trivial. For the right-to-left direction, we show that the contexts $\gamma$ and $\gamma'$ denoted by $u$ and $u'$ must be equal. To this end, we only need to show that $\gamma$ and $\gamma'$ have the same hole; then their equality follows from Lemma 6.
Let's say that $\pi$ and $\pi'$ are the holes of $\gamma$ and $\gamma'$, respectively. The path $\pi$ cannot be a proper prefix of $\pi'$ or vice versa. Otherwise, $\gamma[\tau_1] = \gamma'[\tau_1]$ would not be satisfied. Since $\pi' \in D_{\gamma'[\tau_1]}$, either $\pi' \in D_\gamma$, or $\pi$ is a proper prefix of $\pi'$. But $\pi$ is no prefix (proper or not) of $\pi'$, so $\pi' \in D_\gamma$. As $\pi'$ and $\pi$ are not a prefix of each other, it follows that $\gamma[\tau_1].\pi'{=}\gamma[\tau_2].\pi'$. Hence, by Lemma 3,

$$\begin{aligned}\gamma[\tau_1].\pi' &= \gamma'[\tau_1].\pi' = \tau_1\\ \gamma[\tau_2].\pi' &= \gamma'[\tau_2].\pi' = \tau_2.\end{aligned}$$

So in contradiction to our assumptions, we have derived that $\tau_1 = \tau_2$. □

### 5.4 Application to Semantic Underspecification

It is quite simple to express a scope ambiguity by using equations between context-valued terms. An underspecified representation of the meaning of Example 3 is given below.

$$\begin{aligned}x_\top &= C_1(\mathsf{love}(\mathsf{var}_u, \mathsf{var}_v))\\ C_1 &= C_2(\forall u(\to(\mathsf{man}(\mathsf{var}_u), C_3)))\\ C_1 &= C_4(\exists v(\wedge(\mathsf{woman}(\mathsf{var}_v), C_5)))\end{aligned}$$

The semantics of the whole sentence is represented by the tree denoted by $x_\top$ in solutions of the above equations. The first equation states that the semantic description contains a description of the semantics of the verb love. The context of the verb semantics is denoted by $C_1$. The second equation requires that a quantifier every man is placed within the context denoted by $C_1$, i.e. above the verb. The third equation states that another quantifier a woman has also be placed above the verb.

## 6 Parallelism Constraints into Context Unification

In this section, we encode parallelism constraints (and thus dominance constraints) into context unification. More precisely, we show that for every parallelism constraint $\varphi$, there is an equation system $[\![\varphi]\!]$ in the language of context

$$
\begin{array}{rcl}
[\![X \lhd^* Y]\!]_p & = & \exists C\,(C_X \circ C = C_Y) \qquad\qquad (C \text{ fresh}) \\[6pt]
[\![X{:}f(X_1,\ldots,X_n)]\!]_p & = & \bigwedge_{1\le i\le n} C_{X_i} = C_X \circ f(x_1,\ldots,\bullet,\ldots,x_n) \quad \text{if } (n \ge 1) \\
[\![X{:}a]\!]_p & = & x{=}a \\
[\![X/X'{\sim}Y/Y']\!]_p & = & \exists C\,(C_{X'}{=}C_X \circ C \wedge C_{Y'}{=}C_Y \circ C) \qquad (C \text{ fresh}) \\
[\![\varphi_1 \wedge \varphi_2]\!]_p & = & [\![\varphi_1]\!]_p \wedge [\![\varphi_2]\!]_p
\end{array}
$$

**Fig. 5.** Pre-encoding of dominance and parallelism constraints.

unification with the same solutions (up to a simple correspondence). We freely use equations between context-valued terms, which is safe according to Proposition 1.

We will proceed as follows: First, we define the encoding and consider some examples. Second, we lift the encoding to the first-order theory of parallelism constraints and prove its correctness.

For the proof, we will relate every solution $(\mathcal{M}^\tau, \alpha)$ of a parallelism constraint to a variable assignment $[\![\mathcal{M}^\tau, \alpha]\!]$ into $\mathcal{TC}$ which solves the encoded constraint. With this terminology, the key result (Proposition 3) of our correctness proof (which makes the term "have the same solutions" precise) can be stated like this: For an arbitrary dominance constraint $\varphi$ and its encoding $[\![\varphi]\!]$ as a CU equation system, the following equivalence holds.

$$(\mathcal{M}^\tau, \alpha) \models \varphi \Leftrightarrow \mathcal{TC}, [\![\mathcal{M}^\tau, \alpha]\!] \models [\![\varphi]\!]$$

As illustrated in Section 2, the main obstacle that we must overcome in our encoding of dominance constraints is to provide the power to talk about *occurrences* of subtrees. The central idea is to talk about nodes (occurrences of subtrees) by talking about their contexts. For instance, the two occurrences of $a$ in the term $f(a,a)$ can be specified by the contexts represented by $f(a,\bullet)$ and $f(\bullet,a)$ respectively.

### 6.1 The Encoding

Let us define the encoding of a parallelism constraint $\varphi$. We associate with every variable $X$ appearing in a $\varphi$ a context variable $C_X$ (whose purpose it is to denote the context starting at the root of the tree and whose hole is the node denoted by $X$) and a tree variable $x$ (whose purpose it is to denote the tree below $X$). In addition, we introduce a new tree variable $x_\top$ that we want to denote the entire tree. To ensure that these new variables interact correctly, we impose the following constraint, $\mathrm{Root}(\varphi)$, where $\mathcal{FV}(\varphi)$ are the free variables of $\varphi$:

$$\mathrm{Root}(\varphi) = \bigwedge_{X \in \mathcal{FV}(\varphi)} x_\top{=}C_X(x)$$

In addition, we define a pre-encoding $[\![\cdot]\!]_p$ as in Figure 5. The complete encoding $[\![\cdot]\!]$ is obtained as

$$[\![\varphi]\!] = [\![\varphi]\!]_p \wedge \mathrm{Root}(\varphi).$$

An atomic dominance constraint $X \triangleleft^* X'$ is pre-encoded by $\exists C(C_{X'}{=}C_X \circ C)$, which expresses that the context of the node $X$ can be enlarged by adding more material below its hole to obtain the context $X'$. An atomic parallelism constraint $X/X'{\sim}Y/Y'$ is pre-encoded by $\exists C(C_X \circ C = C_{X'} \wedge C_Y \circ C = C_{Y'})$, which expresses that the context of the node $X$ can be enlarged to the context $X'$ by adding the same material as for enlarging the context of $Y$ to that of $Y'$. The pre-encoding of $X{:}f(X_1, \dots, X_n)$ requires for all $1 \leq i \leq n$ that the context above $X_i$ is the context above $X$, enlarged with $f(x_1, \dots, \bullet, \dots, x_n)$, where the hole is at position $i$. For a nullary labeling constraint $X{:}a$, the pre-encoding requires $x = a$.

**Proposition 2 (Encoding Parallelism Constraints).** *A parallelism constraint $\varphi$ is satisfiable iff its encoding Root($\varphi$) $\wedge$ $[\![\varphi]\!]_p$ is a satisfiable equation system of context unification.*

*Proof.* The proposition will be a simple consequence of Theorem 1, the analogous result for first-order formulae. □

### 6.2 Examples

Before we turn to the first-order case, let us consider some examples. First, we reconsider Example (1) from Section 2. When we tried to encode this dominance constraint as a subtree constraint (2), we lost unsatisfiability. However, our new encoding works just fine. (8) shows the pre-encoding of the example; we have left the Root formula away, as it is not necessary for the unsatisfiability in this case.

(1) $X : f(X_1, X_2) \wedge X_1 \triangleleft^* Y \wedge X_2 \triangleleft^* Y$
(8) $C_{X_1}{=}C_X \circ f(\bullet, x_2) \wedge C_{X_2}{=}C_X \circ f(x_1, \bullet) \wedge C_{X_1} \circ C{=}C_Y \wedge C_{X_2} \circ C'{=}C_Y$

We can see that (8) is unsatisfiable in the following way. As $C_{X_1} \circ C = C_Y$ and $C_{X_2} \circ C'{=}C_Y$, $C_{X_1} \circ C{=}C_{X_2} \circ C'$. In this equation, we can substitute $C_{X_1}$ by $C_X \circ f(\bullet, x_2)$ and $C_{X_2}$ by $C_X \circ f(x_1, \bullet)$ and obtain $f(\bullet, x_2) \circ C{=}f(x_1, \bullet) \circ C'$, which is clearly unsatisfiable because the holes are different on both sides.

Another example will serve to show that the Root formula is really necessary to obtain the correct results. (10) is the (complete) encoding of the (unsatisfiable) dominance constraint (9) ($a$ and $b$ are different constants):

(9) $X{:}a \wedge Y{:}b \wedge X \triangleleft^* Y$
(10) $x_\top{=}C_X(x) \wedge x_\top{=}C_Y(y) \quad \wedge \quad x{=}a \wedge y{=}b \wedge C_X \circ C = C_Y$

The pre-encoding alone (i.e. the last three conjuncts) is satisfiable; together with the Root formula, it isn't. $x_\top{=}C_X(x) \wedge x_\top{=}C_Y(y)$ implies $C_X(x){=}C_Y(y)$, which, when combined with $C_X \circ C = C_Y$, yields $x{=}C(y)$. When using $x{=}a \wedge y{=}b$ as a substitution, we obtain $a{=}C(b)$, which is not satisfiable.

$$
\begin{array}{rcll}
[\![\varphi]\!] & = & [\![\varphi]\!]_p & (\varphi \text{ atomic}) \\[1em]
[\![\Phi_1 \wedge \Phi_2]\!] & = & [\![\Phi_1]\!] \wedge [\![\Phi_2]\!] & \\
[\![\neg\Phi]\!] & = & \neg[\![\Phi]\!] & \\
[\![\exists X.\Phi]\!] & = & \exists C_X \exists x.(x_\top{=}C_X(x) \wedge [\![\Phi]\!]) &
\end{array}
$$

**Fig. 6.** Encoding closed first-order formulas over parallelism constraints.

### 6.3 Encoding First-Order Formulae

In Fig. 6, the encoding of parallelism constraints is lifted to first-order formulae $\Phi$. If we restrict ourselves to closed first-order formulae, an explicit Root formula is no longer needed; its components are distributed among the encodings of existential quantifiers. If we write $\tilde{\exists}\Phi$ for the existential closure of a formula $\Phi$, then it holds for all dominance constraints $\varphi$ that:

$$\tilde{\exists}\ (\text{Root}(\varphi) \wedge [\![\varphi]\!]_p) = [\![\tilde{\exists}\varphi]\!]$$

Hence, the correctness of the encoding $\text{Root}(\varphi) \wedge [\![\varphi]\!]_p$ claimed in Prop. 2 follows from the correctness of the encoding of first-order sentences.
Now let us turn to the proof of the first-order case. First, we formulate the correspondence $[\![\cdot,\cdot]\!]_V$ we announced above. This function maps pairs of tree structures $\mathcal{M}^\tau$ and variable assignments mapping the variables in $V$ to the domain of $\tau$ to variable assignments into $\mathcal{TC}$. The goal is that if the arguments satisfy a given dominance constraint, the result will satisfy its encoding.

$$
\begin{array}{lll}
[\![\mathcal{M}^\tau,\alpha]\!]_V(x_\top) & = \tau & \\
[\![\mathcal{M}^\tau,\alpha]\!]_V(x) & = \tau.\alpha(X) & \text{for all } x \text{ such that } X \in V \\
[\![\mathcal{M}^\tau,\alpha]\!]_V(C_X) & = \tau^\epsilon_{\alpha(X)} & \text{for all } C_X \text{ such that } X \in V.
\end{array}
$$

With this definition, the following proposition holds.

**Proposition 3.** *Let $\mathcal{M}^\tau$ be a tree structure, $\alpha$ a variable assignment, and $\Phi$ a first-order formula over the parallelism constraints. Then $\Phi$ is satisfied by $(\mathcal{M}^\tau,\alpha)$ iff $[\![\Phi]\!]$ is satisfied by $[\![\mathcal{M}^\tau,\alpha]\!]_{\mathcal{FV}(\Phi)}$.*

*Proof.* We prove the proposition by structural induction. First, we show that it is true for the atomic constraints; towards the end of the proof, we conduct the induction steps. Throughout the proof, we write $\beta = [\![\mathcal{M}^\tau,\alpha]\!]_{\mathcal{FV}(\Phi)}$ for brevity.

– $X \triangleleft^* Y$. The treatment of $X/X' {\sim} Y/Y'$ is analogous.
  "⇒" Assume that $(\mathcal{M}^\tau,\alpha)$ satisfies $X \triangleleft^* Y$; we show that $\beta$ satisfies the encoding $\exists C(C_X \circ C{=}C_Y)$. Our assumption means that $\alpha(X)$ is a prefix

of $\alpha(Y)$. Hence, we can construct a variable assignment $\beta'$ that is like $\beta$, but assigns $\tau^{\alpha(X)}_{\alpha(Y)}$ to $C$. By Lemma 8, $\beta'$ solves $C_X \circ C{=}C_Y$ and, thus, $\beta$ is a solution of $[\![X \lhd^* Y]\!]$.

"⇐" Assume that $\exists C(C_X \circ C{=}C_Y)$ is satisfied by $\beta$. Then there must be a context $\gamma$ such that $\beta(C_X) \circ \gamma = \beta(C_Y)$; hence, $\alpha(X)$ must be a prefix of $\alpha(Y)$, and $(\mathcal{M}^\tau, \alpha)$ satisfies $X \lhd^* Y$.

– $X{:}f(X_1, \dots, X_n)$, where $n \geq 1$

"⇒" Assume that $(\mathcal{M}^\tau, \alpha)$ satisfies $X{:}f(X_1, \dots, X_n)$; we assume $1 \leq i \leq n$ and conclude that $\beta$ satisfies all equations $C_{X_i}{=}C_X \circ f(x_1, \dots, \bullet, \dots, x_n)$ where the hole marker $\bullet$ is at position $i$.

Let $u$ be the context-valued term $C_X \circ f(x_1, \dots, \bullet, \dots, x_n)$. We first show that the holes of $\beta(u)$ and $\beta(C_{X_i})$ are the same, and then that their values on $\beta(x_i)$ are equal. (Here we need $n \geq 1$, as $x_i$ would not exist otherwise.) From Lemma 6, we can then conclude $\beta(u) = \beta(C_{X_i})$.

The hole of $\beta(C_{X_i}) = \tau^\epsilon_{\alpha(X_i)}$ is $\alpha(X_i)$, and that of $\beta(u)$ is $\alpha(X)i$. Since $(\mathcal{M}^\tau, \alpha)$ is a solution of $X{:}f(X_1, \dots, X_n)$, we have $\alpha(X)i = \alpha(X_i)$, and hence the holes are equal.

We already noticed that $\alpha(X)i = \alpha(X_i)$ for all $1 \leq i \leq n$. Lemma 2 implies that

$$\begin{aligned}\beta(x) = \tau.\alpha(X) &= f(\tau.\alpha(X)1, \dots, \tau.\alpha(X)n) \\ &= f(\tau.\alpha(X_1), \dots, \tau.\alpha(X_n)) \\ &= f(\beta(x_1), \dots, \beta(x_n))\end{aligned}$$

Based on this equation and Lemma 7, we are now in the position to prove $\beta(u)(\beta(x_i)) = \beta(C_{X_i})(\beta(x_i))$ (and thus $\beta(u) = \beta(C_{X_i})$ as required):

$$\begin{array}{lllllll}\beta(u)(\beta(x_i)) & = & \tau^\epsilon_{\alpha(X)}[f(\beta(x_1), \dots, \beta(x_n)] & = & \tau^\epsilon_{\alpha(X)}[\beta(x)] & = & \tau \\ \beta(C_{X_i})(\beta(x_i)) & = & \tau^\epsilon_{\alpha(X_i)}[\beta(x_i)] & = & \tau & & \end{array}$$

"⇐" Assume that $\beta$ solves the equation $C_X = C_{X_i} \circ f(x_1, \dots, \bullet, \dots, x_n)$ for some $1 \leq i \leq n$, where the hole $\bullet$ is at position $i$. Lemma 7 yields

$$\begin{aligned}&\tau = \tau^\epsilon_{\alpha(X)}[\tau.\alpha(X)] = \beta(C_X)[\beta(x)] \\ &\tau = \tau^\epsilon_{\alpha(X_i)}[\tau.\alpha(X_i)] = \beta(C_{X_i})[\beta(x_i)] = \beta(C_X)[f(\beta(x_1), \dots, \beta(x_n))]\end{aligned}$$

Since context functions are one-to-one and $\beta(C_X)$ is a context function, these equations imply $\beta(X) = f(\beta(x_1), \dots, \beta(x_n))$. This is equivalent to

$$\tau.\alpha(X) = f(\tau.\alpha(X_1), \dots, \tau.\alpha(X_n)),$$

which in turn means that $(\mathcal{M}^\tau, \alpha)$ solves $X{:}f(X_1, \dots, X_n)$.

– $X{:}a$

"⇒" Assume that $(\mathcal{M}^\tau, \alpha)$ satisfies $X{:}a$. Since $\mathsf{ar}(a) = 0$, it follows that $\tau.\alpha(X) = a$, so $\beta$ solves $x{=}a$.

"⇐" Assume that $\beta$ satisfies $x{=}a$; then $\beta(x) = \tau.\alpha(X) = a$ and, hence, $(\mathcal{M}^\tau, \alpha)$ solves $X{:}a$.

Of the complex cases, negation and conjunction are trivial. Existential quantification is more interesting:

- $\exists X.\Phi$
  "$\Rightarrow$" We assume that $(\mathcal{M}^\tau, \alpha)$ satisfies $\exists X.\Phi$; so there is a path $\pi$ such that $(\mathcal{M}^\tau, \alpha[\pi/X])$ solves $\Phi$. By induction hypothesis, $[\![\mathcal{M}^\tau, \alpha[\pi/X]]\!]_{\mathcal{FV}(\Phi)}$ satisfies $[\![\Phi]\!]$. On the free variables of $\Phi$, this variable assignment agrees with $[\![\mathcal{M}^\tau, \alpha]\!]_{\mathcal{FV}(\exists X\Phi)}[\tau.\pi/X, \tau_\pi^\epsilon/C_X]$, and the latter variable assignment satisfies $x_\top = C_X(x)$ as well. Thus $[\![\mathcal{M}^\tau, \alpha]\!]_{\mathcal{FV}(\exists X\Phi)}$ solves $[\![\exists X\Phi]\!]$.
  "$\Leftarrow$" We assume that $\beta = [\![\mathcal{M}^\tau, \alpha]\!]_{\mathcal{FV}(\exists X\Phi)}$ solves $\exists C_X \exists x([\![\Phi]\!] \wedge x_\top {=} C_X(x))$. There is a $\pi$ such that $\beta[\tau.\pi/x, \tau_\pi^\epsilon]$ solves $[\![\Phi]\!]$. Since $\beta[\tau.\pi/x, \tau_\pi^\epsilon/C_X]$ is equal to $[\![\mathcal{M}^\tau, \alpha[\pi/X]]\!]_{\mathcal{FV}(\Phi)}$ on all free variables of $[\![\Phi]\!]$, it follows from the induction hypothesis that $(\mathcal{M}^\tau, \alpha[\pi/X])$ solves $\Phi$. Hence $(\mathcal{M}^\tau, \alpha)$ solves $\exists X\Phi$. $\square$

**Corollary 1 (Encoding First-Order Formulae).** *A closed first-order formula $\Phi$ over dominance and parallelism constraints is satisfied by a pair $(\mathcal{M}^\tau, \alpha)$ iff there is a variable assignment $\beta$ into $\mathcal{TC}$ that solves $[\![\Phi]\!]$ such that $\beta(x_\top) = \tau$.*

## 7 Context Unification into Parallelism Constraints

We finally show how to express equations of context unification by parallelism constraints. This is not obvious but it follows from a result of [NPR97a] which shows that CU has the same expressive power as *equality up-to constraints.* Equality up-to constraints can be translated to parallelism constraints plus *similarity constraints.* Finally, one can get rid of similarity constraints by a neat trick.

An equality up-to constraint is a conjunction of atomic constraints of the following form, which are interpreted in the algebra $\mathcal{TC}$.

$$\psi \ ::= \ x/x'{=}y/y' \mid x{=}f(x_1, \ldots, x_n) \mid \psi \wedge \psi'$$

An atomic equality up-to constraint $x/x'{=}y/y'$ is satisfied by a variable assignment $\beta$ into $\mathcal{TC}$ if there is a context $\gamma$ such that $\beta(x) = \gamma[\beta(x')]$ and $\beta(y) = \gamma[\beta(y')]$. Intuitively, this is the case iff the trees denoted by $x$ and $y$ are *equal, up to* an occurrence of $x'$ in $x$ and of $y'$ in $y$ respectively. Equality up-to constraints are equivalent to context unification:

**Proposition 4 (Equality up-to Constraints and CU [NPR97a]).** *For every equation system of context unification, there is a satisfaction equivalent equality up-to constraint, and vice versa.*

With this result, it remains to encode equality up-to constraints into parallelism constraints. This would be simple if parallelism constraints could express similarity constraints. A similarity constraint has the form $X{\sim}Y$ and is interpreted by the *similarity relation.* A similarity relationship $\pi{\sim}\pi'$ holds for two nodes if $\tau.\pi = \tau.\pi'$ (i.e. if the s ubtrees below $\pi$ and $\pi'$ are the same).

| | | |
|---|---|---|
| $[\![x/x'=y/y']\!]^{-1}$ | $= X/X''{\sim}Y/Y'' \wedge X'{\sim}X'' \wedge Y'{\sim}Y''$ | $(X'', Y''$ fresh$)$ |
| $[\![x=f(x_1,\ldots,x_n)]\!]^{-1}$ | $= X{:}f(X_1',\ldots,X_n') \wedge \bigwedge_{i=1}^n X_i{\sim}X_1'$ | $(X_1',\ldots,X_n'$ fresh$)$ |

**Fig. 7.** Encoding equality up-to into parallelism and similarity constraints.

**Lemma 9.** *If the signature $\Sigma$ contains a single constant, then parallelism constraints can express similarity constraints.*

*Proof.* Let $a$ be the unique constant of $\Sigma$. Every finite tree must contain a node labeled with $a$; so the following equivalence holds for all tree models:

$$X{\sim}Y \;\leftrightarrow\; \exists Z \exists Z' (Z{:}a \wedge Z'{:}a \wedge X/Z{\sim}Y/Z') \qquad \square$$

If the number of constants in $\Sigma$ is finite, we can express $X{\sim}Y$ by a finite disjunction; but this would not lead to a polynomial time transformation. But there is a neat trick to work around which even applies for infinitely many constants.

**Lemma 10.** *For every signature $\Sigma$, there exists a signature $\Sigma'$ with a single constant such that parallelism and similarity constraints over $\Sigma$ can be translated in linear time into satisfiability equivalent constraints of the same kind over $\Sigma'$.*

*Proof.* For any signature $\Sigma$, let $\Sigma'$ be the signature consisting of all non-constant symbols of $\Sigma$, plus the constants of $\Sigma$ considered as unary function symbols, plus a new constant $a$. We transform each parallelism constraint $\varphi$ into a constraint $\varphi'$ by replacing every constraint $X{:}b$ by $\exists Y (X{:}b(Y) \wedge Y{:}a)$. Now it is easy to see that $\varphi$ is satisfiable over $\Sigma$ iff $\varphi'$ is satisfiable over $\Sigma'$. $\square$

**Theorem 1 (Parallelism Constraints = Context Unification).** *For every parallelism constraint $\varphi$, there is a satisfiability equivalent equation system of context unification, and vice versa.*

*Proof.* The correctness of an encoding of parallelism constraints into CU is stated in Proposition 2.
For the converse, we first express CU by equality up-to constraints according to Proposition 4. Second, we encode equality up-to constraints by parallelism and similarity constraints. This is quite easy; an encoding $[\![\psi]\!]^{-1}$ is defined in Figure 7. In order to encode $\psi$, we assume a node variable $X$ for every tree variable $x$ occurring in $\psi$. The variable $X$ is supposed to denote the root node of an occurrence of $x$ in the solution of the encoding of $\varphi$. It is obvious that $[\![\cdot]\!]^{-1}$ preserves satisfiability. The encoding of $x/x'=y/y'$ expresses that somewhere below the nodes $X$ and $Y$, there are nodes $X''$ and $Y''$ the trees below which look just like the trees below the nodes $X'$ and $Y'$, and the contexts between $X$ and $X''$

and $Y$ and $Y''$ are equal. (Note that this is a weaker condition than parallelism itself; it does not say anything about the locations of the nodes denoted by $X'$ and $Y'$.) The encoding of equation $x{=}f(x_1, \dots, x_n)$ works similarly: It expresses that $X$ is labeled with $f$ and that its subtrees look just like the subtrees below the $X_1, \dots, X_n$.

Third, we switch to a signature with a single constant which we can do according to Lemma 10. We can now express all similarity constraints by parallelism constraints (Lemma 9) which completes the proof. □

## 8 Conclusion

The main result of this paper is that context unification has the same expressive power as parallelism constraints. Parallelism constraints subsume dominance constraints. The most involved part was to embed dominance constraints into CU. The inverse direction from CU to parallelism constraints proceeds via a deviation through equality up-to constraints, which have the same expressivess as CU as well.

The correspondence between CU and CLLS has two important consequences. For one, it allows us to transfer complexity and decidability results. For the time being, however, the decidability of either language is unknown. Conversely, the satisfiability problem of dominance constraints is shown NP-complete in [KNT98]. Of course, NP-hardness for several fragments of CU was well known before.

The other consequence is that CU can be easily expressed by parallelism constraints in CLLS [ENRX98] which explains why the linguistic application given for CU in [NPR97b] carries over to CLLS. Furthermore, this application of CU is clarified. In earlier papers, scope ambiguities could be described in CU but only in a somewhat intransparent fashion. In the light of the results presented, it becomes clear that the equations used previously were really just encodings of dominance and parallelism constraints.

**Acknowledgments** We are deeply indebted to Peter Ruhrberg, a former colleague of ours who conjectured the presented relationship long before CLLS was found. It is a pleasure to thank all members (student or not) of the CHORUS project. The research reported here was supported by the SFB 378 at the Universität des Saarlandes and the Esprit Working Group CCL II (EP 22457).

## 9 References

[AW92] Alexander Aiken and E.L. Wimmers. Solving Systems of Set Constraints. In *International Conference on Logic in Computer Science*, pages 329–340, June 1992.

[BGMV93] P. Blackburn, C. Gardent, and W. Meyer-Viol. Talking about trees. In *European Chapter of the Association of Comp. Linguistics*, 1993.

[Bos96] Johan Bos. Predicate logic unplugged. In *Proc. of the 10th Amsterdam Colloquium*, pages 133–143, 1996.

[BS93] F. Baader and J. Siekmann. Unification theory. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*. Oxford University Press, 1993.

[BS95] Rolf Backofen and Gert Smolka. A complete and recursive feature theory. *Journal of Theoretical Computer Science*, 146(1–2):243–268, July 1995.

[Col84] Alain Colmerauer. Equations and inequations on finite and infinite trees. In *Proceedings of the 2nd International Conference on Fifth Generation Computer Systems*, pages 85–99, 1984.

[Com92] Hubert Comon. Completion of rewrite systems with membership constraints. In *Coll. on Automata, Languages and Programming*, volume 623 of *LNCS*, 1992.

[DG99] Denys Duchier and Claire Gardent. A constraint-based treatment of descriptions. In *Proceedings of IWCS-3*, Tilburg, 1999.

[ENRX98] Markus Egg, Joachim Niehren, Peter Ruhrberg, and Feiyu Xu. Constraints over lambda-structures in semantic underspecification. In *Proc. of COLING/ACL*, pages 253–359, 1998.

[GW98] Claire Gardent and Bonnie Webber. Describing discourse semantics. In *Proc. of the TAG+ Workshop*, Philadelphia, 1998.

[HJ90] Nevin Heintze and Joxan Jaffar. A decision procedure for a class of set constraints. In *International Conference on Logic in Computer Science*, pages 42–51, 1990.

[KNT98] Alexander Koller, Joachim Niehren, and Ralf Treinen. Dominance constraints: Algorithms and complexity. In *Proceedings of the Third Conference on Logical Aspects of Computational Linguistics*, Grenoble, 1998.

[Kol99] Alexander Koller. Constraint languages for semantic underspecification. Master's thesis, Universität des Saarlandes, Saarbrücken, 1999. `http://www.coli.uni-sb.de/~koller/papers/da.html`.

[KR86] Robert T. Kasper and William C. Rounds. A logical semantics for feature structures. In *Annual Meeting of the Association of Comp. Linguistics*, pages 257–265, 1986.

[Lév96] Jordi Lévy. Linear second order unification. In *International Conference on Rewriting Techniques and Applications*, volume 1103 of *LNCS*, pages 332–346, 1996.

[Mak77] G.S. Makanin. The problem of solvability of equations in a free semigroup. *Soviet Akad. Nauk SSSR*, 223(2), 1977.

[MHF83] M. P. Marcus, D. Hindle, and M. M. Fleck. D–theory: Talking about talking about trees. In *Proc. of the Annual Meeting of the ACL*, 1983.

[MM82] Alberton Martelli and Ugo Montanari. An efficient unification algorithm. *ACM TOPLAS*, 4(2):258–282, 1982.

[MN00] Martin Müller and Joachim Niehren. Ordering constraints over feature trees expressed in second-order monadic logic. *Information and Computation*, 2000. Special Issue on RTA, Tsukuba, Japan, March 1998. To appear.

[Moz99] Mozart Consortium: Universität des Saarlandes, DFKI, SFB 378, SICS, Université de Louvain. The Mozart System of Oz. Freely available at `www.mozart-oz.org`, January 1999.

[Mus98] Reinhard Muskens. Underspecified semantics. CLAUS Report 95, Univ. des Saarlandes, Saarbrücken, 1998.

[MVK98] Wilfried Meyer-Viol and Ruth Kempson. Sequential construction of logical forms. In *Proceedings of the Third Conference on Logical Aspects of Computational Linguistics*, Grenoble, France, 1998.

[NPR97a] Joachim Niehren, Manfred Pinkal, and Peter Ruhrberg. On equality up-to constraints over finite trees, context unification and one-step rewriting. In *Proc. of the CADE*, volume 1249 of *LNCS*, pages 34–48, 1997.

[NPR97b] Joachim Niehren, Manfred Pinkal, and Peter Ruhrberg. A uniform approach to underspecification and parallelism. In *Annual Meeting of the Association of Comp. Linguistics*, pages 410–417, 1997.

[NTT99] Joachim Niehren, Ralf Treinen, and Sophie Tison. On rewrite constraints and context unification. Technical report, Universität des Saarlandes, Programming Systems Lab, 1999. Submitted. Available at `http://www.ps.uni-sb.de/Papers/abstracts/rewrite-context`.

[Rab69] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[Rey93] Uwe Reyle. Dealing with ambiguities by underspecification: construction, representation, and deduction. *Journal of Semantics*, 10:123–179, 1993.

[RTA98] Decidability of context unification. The RTA list of open problems, number 90, `http://www.lri.fr/~rtaloop/`, 1998.

[Sch93] Klaus U. Schulz. Word unification and transformation of generalized equations. *Information and Computation*, 11:149–184, 1993.

[Smo92] Gert Smolka. Feature constraint logics for unification grammars. *Journal of Logic Programming*, 12:51–87, 1992.

[Smo95] Gert Smolka. The Oz programming model. In Jan van Leeuwen, editor, *Computer Science Today*, Lecture Notes in Computer Science, vol. 1000, pages 324–343. Springer-Verlag, 1995.

[SS94] Manfred Schmidt-Schauß. Unification of stratified second-order terms. Technical Report 12/94, J. W. Goethe Universität, Frankfurt, 1994.

[SS97] Manfred Schmidt-Schauß. A unification algorithm for distributivity and a multiplicative unit. *J. of Symbolic Computation*, 22(3):315–344, 1997.

[SSS98] Manfred Schmidt-Schauß and Klaus Schulz. On the exponent of periodicity of minimal solutions of context equations. In *International Conference on Rewriting Techniques and Applications*, volume 1379 of *LNCS*, 1998.

[SSS99] Manfred Schmidt-Schauß and Klaus Schulz. Solvability of context equations with two context variables is decidable. In *Proc. of the CADE*, LNCS, 1999.

[TDT00] Jean-Marc Talbot, Phillipe Devienne, and Sophie Tison. Generalized definite set constraints. *Constraints, an International Journal, Special Issue on CP'97*, January 2000.

[Ven87] K. N. Venkataraman. Decidability of the purely existential fragment of the theory of term algebra. *Journal of the ACM*, 34(2):492–510, 1987.

[vP96] Kees van Deemter and Stanley Peters. *Semantic Ambiguity and Underspecification*. CSLI, Stanford, 1996.

[VS92] K. Vijay-Shanker. Using descriptions of trees in a tree adjoining grammar. *Computational Linguistics*, 18(4):481–517, 1992.