Describing Lambda Terms in Context Unification

Joachim Niehren, Mateu Villaret

Describing Lambda Terms in Context Unification*

Joachim Niehren[1] and Mateu Villaret[2]

[1] INRIA Futurs, Lille, France.
[2] Universitat de Girona, Girona, Spain.

Abstract. The constraint language for lambda structures (CLLS) is a description language for lambda terms. CLLS provides parallelism constraints to talk about the tree structure of lambda terms, and lambda binding constraints to specify variable binding. Parallelism constraints alone have the same expressiveness as context unification. In this paper, we show that lambda binding constraints can also be expressed in context unification when permitting tree regular constraints.

Keywords: second-order unification, dominance constraints, computational linguistics, underspecified semantics.

1 Introduction

The *constraint language for lambda structures* (CLLS) is a first-order language for describing lambda terms [5, 6]. CLLS provides parallelism constraints [7] to talk about the tree structure of lambda terms, and lambda binding constraints to specify variable binding. In particular, CLLS models the interaction of parallelism and variable binding in lambda terms correctly.

CLLS supports parallelism constraints to model ellipsis in natural language. These subsume dominance constraints [13, 23, 1, 4], an excellent modeling language for scope underspecification [15]. CLLS features lambda binding constraints in order to analyze the interactions of scope and ellipsis, i.e., parallel lambda binding. Further ingredients of CLLS are anaphoric binding constraints, group parallelism and beta reduction constraints [2].

Parallelism constraints of CLLS alone have the same expressive power as the context equations in *context unification* (CU) [14, 3]. CU is a variant of *linear second-order unification* (LSOU) [9] which restricts second-order unification to unifiers with linear lambda-terms. LSOU and CU only differ in variable binding; this difference can be captured by imposing tree regular constraints [11].

The decidability of the satisfiability problems for CU, LSOU, and CLLS – with or without tree regular constraints – are prominent open questions. Decidability of CU was often conjectured (see e.g. [12]). This is because various restrictions [3, 9, 18–20] make CU decidable, while the analogous restrictions for second-order unification don't [10]. A decidable fragment of CLLS that can deal

* A previous version of the paper was presented at ICoS4. This research has been partially supported, on the one hand, by the Mostrare project of INRIA Futurs and the LIFL at the Universities of Lille 1 and 3, and on the other hand, by the spanish projects: TIN2004-07672-C03-01 and TIN2004-04343.

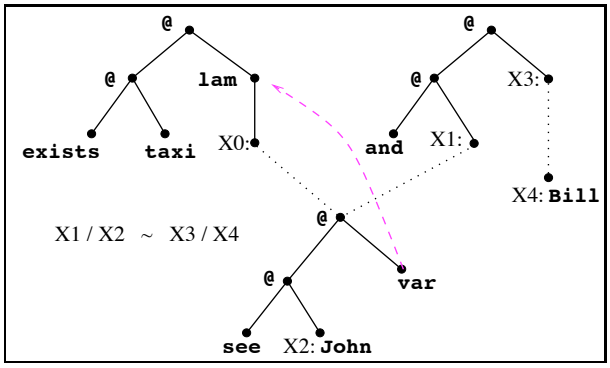

Fig. 1. The graph of a CLLS constraint

with most phenomena raised by scope underspecification and ellipsis was proposed recently [8]. Generally speaking, CLLS is superior in semantic modeling, while CU has advantages for studying expressiveness. This is why this paper investigates the expressiveness of CLLS by comparison to CU.

In this paper, we contribute the first comparison between CLLS and CU which accounts for lambda binding constraints. The motivating question is whether it is possible to describe the interaction of lambda binding and parallelism in CU. In other words: *can parallel lambda binding be expressed in CU similarly to CLLS?* Note that parallel lambda binding was one of the main motivations for introducing CLLS as an alternative to CU. We give a positive answer to the question but at the cost of tree regular constraints.

We show how to encode lambda binding and parallelism constraints in CU with tree regular constraints. Our encoding composes several steps, one of which exploits a recent variant of the famous relationship between monadic second-order logic (MSO) and tree automata [16, 22]. Another step relies on the non-intervenance property of lambda binding constraints, that we exhibit and prove for the first time in the present paper.

Plan. We illustrate parallel lambda binding in CLLS and argue its relevance for underspecified semantic modeling (Sec. 2). We recall tree structures, parallelism, lambda structures, and parallel lambda binding (Sec.3 and 4), the basic notions underlying CLLS, the constraint language for lambda structures (Sec. 5). We exhibit the non-intervenance property of parallel lambda binding in CLLS and prove it to hold (Sec. 6). Due to this property, we can encode parallel lambda binding by using MSO formulas (Sec. 7). Parallelism constraints with MSO formulas have the same expressiveness as CU with tree regular constraints (Sec. 8). We discuss the limitation of our approach with respect to group parallelism (Sec. 9) and conclude with some open questions.

2 Parallel Lambda Binding in Semantics

The prime application of CLLS is the modeling of underspecification in natural language semantics [5, 6]. It improves on previous approaches on analyzing that

were based on higher-order unification [21], LSOU [17], and CU. Here, we illustrate why parallel lambda binding is crucial to capture the interactions of scope and ellipsis.

We consider the sentence: *John saw a taxi and so did Bill.* This elliptic sentence has two possible meanings that can be represented in higher-order logic by the following Boolean valued lambda terms:

1. There exists a taxi t seen by John and Bill:

$$exists\ taxi\ \lambda t.(and\ (see\ john\ t)\ \underline{(see\ bill\ t)})$$

2. There exists a taxi t_1 seen by John and a taxi t_2 seen by Bill.

$$and\ (exists\ taxi\ \lambda t_1.(see\ john\ t_1))\ \underline{(exists\ taxi\ \lambda t_2.(see\ bill\ t_2))}$$

The ellipses requires the meanings of source and target to be parallel except for the contrasting elements *john* and *bill*. Note that the elided parallel segments, shown underlined, are different in the two readings.

This example illustrates parallel lambda binding: in the first reading, both occurrences of t are bound by the same lambda binder taking scope over both parallel segments, while in the second case, the corresponding variables t_1 and t_2 are bound by distinct lambda binders that correspond to each other in the parallel segments. The parallel parts become equal if we rename the variables t_1 and t_2 to t. But renaming needs to be done carefully without variable capture.

The CLLS constraint in Fig. 1 can be derived by a compositional semantics construction from a parse tree. The parallelism free part describes the meanings of source sentence, the conjunction, and the target parallel element. By means of dotted lines, that express dominance between nodes, it leaves underspecified where to place the fragment with the lambda binder $@(@(\mathsf{exists}, \mathsf{taxi}),\mathsf{lam}(X_0))$, either above the conjunction $@(@(\mathsf{and}, X_1), X_3)$ or below its first argument that starts at node X_1.

The parallelism constraint $X_1/X_2 \sim X_3/X_4$ expresses the parallelism requirement of the ellipses. The parallel segments X_1/X_2 and X_3/X_4 must have equal tree structure and parallel binding relations. The meaning of parallel lambda binding is formalized by CLLS's lambda structures (see Section 4).

The lambda terms of both readings satisfy the constraint in Fig. 1. Binding constraints are imposed by dashed edges from var to lam-labeled nodes, the lambda binding constraints. The single dashed edge in Fig. 1 is satisfied in both readings: in the first reading, it holds since t is bound by λt, in the second reading it holds since t_1 is bound by λt_1 and since t_2 is bound by λt_2. No variable names are used in CLLS; this avoids variable renaming and capturing during constraint resolution once and for all.

Parallel lambda binding cannot be easily expressed in CU or LSOU where lambda binding constraints are not available. In CU for instance, the only known approach to express lambda binding is by adding variable names into function symbols lam_t and var_t. Different variables should be named apart, in order to avoid variable capture during constraint resolution. But this is not possible for variables such as t_1 and t_2 above: context equality would impose equal names.

3 Tree Structures and Parallelism

We assume a finite *signature* Σ of function symbols ranged over by f, g. Each function symbol has an arity $\mathsf{ar}(f) \geq 0$.

Finite Trees. A finite (rooted) tree τ over Σ is a ground term over Σ, i.e. $\tau ::= f(\tau_1, \ldots, \tau_n)$ where $n = \mathsf{ar}(f) \geq 0$ and $f \in \Sigma$. We identify a node of a tree with the word of positive integers π that addresses this node from the root:

$$\mathsf{nodes}_{f(\tau_1,\ldots,\tau_n)} = \{\epsilon\} \cup \{i\pi \mid 1 \leq i \leq n,\ \pi \in \mathsf{nodes}_{\tau_i}\}$$

The empty word ϵ is called the *root* of the tree, i.e. $\mathsf{root}(\tau) = \epsilon$, while a word $i\pi$ addresses the π node of the i-th subtree of τ. We freely identify a tree τ with the function $\tau : \mathsf{nodes}_\tau \to \Sigma$ that maps nodes to their label; for $\tau = f(\tau_1, \ldots, \tau_n)$:

$$\tau(\pi) = f(\tau_1, \ldots, \tau_n)(\pi) = \begin{cases} f & \text{if } \pi = \epsilon \\ \tau_i(\pi') & \text{if } \pi = i\pi' \end{cases}$$

If τ is a tree with $\pi \in \mathsf{nodes}_\tau$ then we write $\tau.\pi$ for the subtree of τ rooted by π, and $\tau[\pi/\tau']$ for the tree obtained by replacing the subtree of τ at node π by τ'.

Dominance and Parallelism. Let τ be a tree with $\pi, \pi', \pi_1, \ldots, \pi_n \in \mathsf{nodes}_\tau$. The relation $\pi{:}f(\pi_1, \ldots, \pi_n)$ holds for τ if node π is labeled by f in τ and has the children $\pi_1, \ldots, \pi_n$ in that order from left to right. This is if $\tau(\pi) = f$ and $\pi_1 = \pi 1$, ..., $\pi_n = \pi n$ where $n = \mathsf{ar}(f)$. The *dominance relation* $\pi \lhd^* \pi'$ holds for τ if π is an ancestor of π', i.e. if π is above π' in τ, i.e. if π is a prefix of π'. *Strict dominance* $\pi \lhd^+ \pi'$ holds for τ if $\pi \lhd^* \pi'$ but not $\pi{=}\pi'$ in τ. The *disjointness relation* $\pi \perp \pi'$ holds for τ if neither $\pi \lhd^* \pi'$ nor $\pi' \lhd^* \pi$ in τ.

Definition 1. A *segment* $\sigma = \pi/\pi_1, \ldots, \pi_n$ of a tree τ is a tuple of nodes $\pi, \pi_1, \ldots, \pi_n$ of τ such that π dominates all π_i and, all π_i with different index are pairwise disjoint. We call π the *root* of σ and $\pi_1, \ldots, \pi_n$ its *holes*. The *nodes of a segment* σ of a tree τ lie between the root and the holes of σ in τ:

$$\mathsf{nodes}_\tau(\pi/\pi_1, \ldots, \pi_n) = \{\pi' \in \mathsf{nodes}_\tau \mid \pi \lhd^* \pi' \text{ and not } \pi_i \lhd^+ \pi' \text{ for any } 1 \leq i \leq n\}$$

Segment nodes generalize tree nodes in that $\mathsf{nodes}_{\tau.\pi} = \mathsf{nodes}_\tau(\pi/)$ for all trees τ and $\pi \in \mathsf{nodes}(\tau)$. The labels of holes do not belong to segments. The *inner nodes of a segment* are those that are not holes:

$$\mathsf{nodes}^-_\tau(\sigma) = \mathsf{nodes}_\tau(\sigma) - \{\pi_1, \ldots, \pi_n\} \qquad \text{if } \sigma = \pi/\pi_1, \ldots, \pi_n$$

Definition 2. A *correspondence function* between segments σ_1 and σ_2 with the same number of holes of a tree τ is a function $c : \mathsf{nodes}_\tau(\sigma_1) \to \mathsf{nodes}_\tau(\sigma_2)$ that is one-to-one and onto and satisfies the following homomorphism conditions:

1. The root of σ_1 is mapped to the root of σ_2 and the sequence of holes of σ_1 is mapped to the sequence of holes of σ_2 in the same order.
2. The labels of inner nodes $\pi \in \mathsf{nodes}^-_\tau(\sigma_1)$ are preserved: $\tau(\pi) = \tau(c(\pi))$.
3. The children of inner nodes in $\pi \in \mathsf{nodes}^-_\tau(\sigma_1)$ are mapped to corresponding children in σ_2: for all $1 \leq i \leq \mathsf{ar}(\tau(\pi))$ it holds that $c(\pi i) = c(\pi)i$.

We call two segments σ_1 and σ_2 of a tree structure τ *(tree) parallel* and write $\sigma_1 \sim \sigma_2$ if and only if there exists a correspondence function between them.

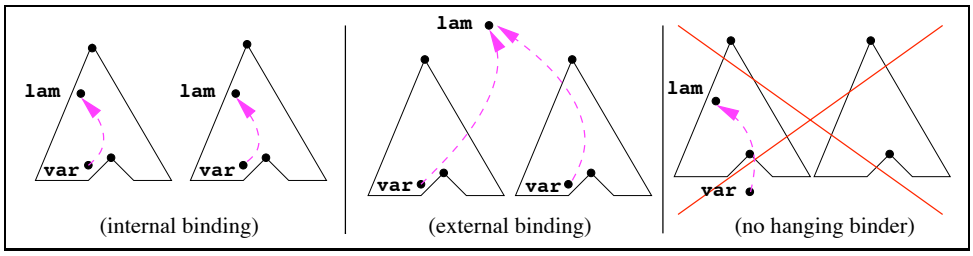

Fig. 2. Axioms of parallel lambda binding.

4 Lambda Structures and Parallel Lambda Binding

Lambda structures represent lambda terms uniquely modulo renaming of bound variables. They are tree structures extended by lambda binding edges. The signature Σ of lambda structures contains, at least, the symbols var (arity 0, for variables), lam (arity 1, for abstraction), and @ (arity 2, for application).

The tree uses these symbols to reflect the structure of the lambda term. The binding function λ maps var-labeled to lam-labeled nodes. For example, Fig. 3 shows the lambda structure of the term $\lambda x.\,(f\ x)$ which satisfies $\lambda(12) = \epsilon$.

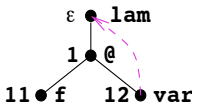

Fig. 3. $\lambda x.\,(f\ x)$

Definition 3. *A* lambda structure (τ, λ) *is a pair of a tree* τ *and a total binding function* $\lambda : \tau^{-1}(\mathsf{var}) \to \tau^{-1}(\mathsf{lam})$ *such that* $\lambda(\pi) \lhd^* \pi$ *for all* var*-nodes* π *in* τ.

We consider lambda structures as logical structures with the relations of tree structures, lambda binding $\lambda(\pi) = \pi'$, and its inverse relation. Inverse lambda binding $\lambda^{-1}(\pi_0)=\{\pi_1, \ldots, \pi_n\}$ states that π_0 binds $\pi_1, \ldots, \pi_n$ and no other nodes.

Definition 4. Two segments σ_1, σ_2 of a lambda structure (τ, λ) are *(binding) parallel* $\sigma_1 \sim \sigma_2$ if they are tree parallel so that the correspondence function c between σ_1 and σ_2 satisfies the following axioms of parallel binding (see Fig. 2):

Internal binder. Internal lambda binder in parallel segments correspond: for all $\pi \in \mathsf{nodes}^-_\tau(\sigma_1)$ if $\lambda(\pi) \in \mathsf{nodes}^-_\tau(\sigma_1)$ then $\lambda(c(\pi)) = c(\lambda(\pi))$.

External binder. External lambda binder of corresponding var-nodes are equal: for all $\pi \in \mathsf{nodes}^-_\tau(\sigma_1)$ if $\lambda(\pi) \notin \mathsf{nodes}^-_\tau(\sigma_1)$ then $\lambda(c(\pi)) = \lambda(\pi)$.

No hanging binder. A var-node below a segment cannot be bound by a lam-node within: $\lambda^{-1}(\pi) \subseteq \mathsf{nodes}^-_\tau(\sigma_i)$ for all $i \in 1, 2$ and $\pi \in \mathsf{nodes}^-_\tau(\sigma_i)$.

Note that this definition overloads the notion of parallelism $\sigma_1 \sim \sigma_2$. For tree structures it means tree parallelism and for lambda structures binding parallelism (if not stated differently). The following basic property will be useful to prove Lemma 8 in Sec. 6.

Lemma 5. *Parallelism in lambda structures is symmetric: if $\sigma_1 \sim \sigma_2$ holds in a lambda structure then $\sigma_2 \sim \sigma_1$ holds as well.*

Proof. Suppose that σ_1 and σ_2 are parallel segments of a lambda structure (τ, λ) and that c is the correspondence function between them. By assumption, c satisfies the axioms of parallel binding. We have to show that the inverse correspondence function c^{-1} also satisfies these axioms.

Internal binder. Let $\pi, \lambda(\pi) \in \mathsf{nodes}^-_\tau(\sigma_2)$ and $\pi' = c^{-1}(\pi)$ be a node in $\mathsf{nodes}^-_\tau(\sigma_1)$. Since $\lambda(\pi')$ dominates π' there remain only two possibilities:

1. Case $\lambda(\pi') \in \mathsf{nodes}^-_\tau(\sigma_1)$. The **internal binder** axiom for c yields $c(\lambda(\pi')) = \lambda(c(\pi')) = \lambda(\pi)$. We can apply the inverse function c^{-1} on both sides and obtain $\lambda(c^{-1}(\pi)) = c^{-1}(\lambda(\pi))$ as required.
2. Case $\lambda(\pi') \notin \mathsf{nodes}^-_\tau(\sigma_1)$. The **external binder** axiom for c implies $\lambda(\pi') = \lambda(c(\pi')) = \lambda(\pi)$. If π' does not belong to the inner nodes of σ_2 then $\lambda(\pi')$ is a **hanging binder** which is not possible. In the same way, we can prove by induction that $(c^{-1})^n(\pi)$ must belong to the inner nodes of σ_2 for all $n \geq 1$. But this is also impossible as trees are finite.

External binder. Suppose that $\pi \in \mathsf{nodes}^-_\tau(\sigma_2)$ while $\lambda(\pi) \notin \mathsf{nodes}^-_\tau(\sigma_2)$. Let $\pi' = c^{-1}(\pi) \in \mathsf{nodes}^-_\tau(\sigma_1)$. Again, there are two possibilities:

1. Case $\lambda(\pi') \in \mathsf{nodes}^-_\tau(\sigma_1)$. The **internal binder** axiom for c yields $c(\lambda(\pi')) = \lambda(c(\pi')) = \lambda(\pi)$ which is impossible since $\lambda(\pi)$ does not belong to the image $\mathsf{nodes}^-_\tau(\sigma_2)$ of c.
2. Case $\lambda(\pi') \notin \mathsf{nodes}^-_\tau(\sigma_1)$. The **external binder** for c implies $\lambda(\pi') = \lambda(c(\pi')) = \lambda(\pi)$ as required.

No hanging binder. This axiom coincides for c and c^{-1}.

5 Constraint Languages

Given the model-theoretic notions of tree structures and lambda structures we can now define logical languages for their description in the usual Tarski'an manner.

We assume an infinite set X, Y, Z of *node variables* and define languages of tree descriptions in Figure 4. A *lambda binding constraint* μ is a conjunction of lambda binding and inverse lambda binding literals: $\lambda(X){=}Y$ means that the value of X is a var-node that is lambda bound by the value of Y, while $\lambda^{-1}(X){\subseteq}\{X_1, \ldots, X_m\}$ says that all var-nodes bound by the lam-node denoted by X are values of one of $X_1, \ldots, X_m$.

A *dominance constraint* is a conjunction of dominance $X{\lhd^*}Y$ and children-labeling literals $X{:}f(X_1, \ldots, X_n)$ that describe the respective relations in some tree structure. We will write $X{=}Y$ to abbreviate $X{\lhd^*}Y \ \wedge \ Y{\lhd^*}X$. Note that dominance constraints are subsumed by parallelism constraints by definition. A *first-order dominance formula* ν is built from dominance constraints and the usual first-order connectives: universal quantification, negation, and conjunction. These can also express existential quantification $\exists X.\nu$ and disjunction $\nu_1 \ \vee \ \nu_2$

Lambda binding constraints:

$$\mu \quad ::= \quad \lambda(X){=}Y \;\mid\; \lambda^{-1}(X){\subseteq}\{X_1,\ldots,X_m\} \;\mid\; \mu_1 \wedge \mu_2$$

First-order dominance formulas:

$$\nu \quad ::= \quad X{:}f(X_1,\ldots,X_n) \;\mid\; X{\lhd^*}Y \;\mid\; \forall X.\nu \;\mid\; \neg\nu \;\mid\; \nu_1 \wedge \nu_2$$

Parallelism constraints:

$$\phi \quad ::= \quad X{:}f(X_1,\ldots,X_n) \;\mid\; X{\lhd^*}Y \;\mid\; S_1{\sim}S_2 \;\mid\; \phi_1 \wedge \phi_2$$

Segment terms:

$$S \quad ::= \quad X/X_1,\ldots,X_m \qquad (m \geq 0)$$

Fig. 4. Logical languages for tree and lambda structures

that we will freely use. Furthermore, we will write $X{\neq}Y$ instead of $\neg X{=}Y$ and $X{\lhd^+}Y$ for $X{\lhd^*}Y \wedge X{\neq}Y$.

A *parallelism constraint* ϕ is a conjunction of children-labeling, dominance, and parallelism literals $S_1{\sim}S_2$. We use *segment terms* S of the form $X/X_1,\ldots,X_m$ to describe segments with m holes, given that the values of X and $X_1,\ldots,X_m$ satisfy the conditions imposed on the root and holes of segments (Definition 1). Note that a parallelism literal $S_1{\sim}S_2$ requires that the values of S_1 and S_2 are indeed segments.

To keep this section self contained let us quickly recall some model theoretic notions. We write $\mathsf{var}(\psi)$ for the set of free variables of a formula ψ of one of the above kinds. A *variable assignment* to the nodes of a tree τ is a total function $\alpha : V \to \mathsf{nodes}(\tau)$ where V is a finite subset of node variables. A *solution* of a formula ψ thus consists of a tree structure τ or a lambda structure (τ,λ) and a variable assignment $\alpha : V \to \mathsf{nodes}(\tau)$ such that $\mathsf{var}(\psi) \subseteq V$. Segment terms evaluate to tuples of nodes $\alpha(X/X_1,\ldots,X_n) = \alpha(X)/\alpha(X_1),\ldots,\alpha(X_n)$ which may or may not be segments. Apart from this, we require as usual that a formula evaluates to true in all solutions. We write $\tau,\alpha \models \psi$ if τ,α is a solution of ψ, and similarly $(\tau,\lambda),\alpha \models \psi$. A formula is *satisfiable* if it has a solution.

Sections 6 and 7 deal with the translation required in the following theorem.

Theorem 6. *Satisfiability of parallelism and lambda binding constraints $\phi \wedge \mu$ can be reduced in non-deterministic polynomial time to satisfiability of parallelism constraints with first-order dominance formulas $\phi' \wedge \nu$.*

Note that the signature is part of the input of the respective satisfiability problems. This means that a formula $\phi \wedge \mu$ over signature Σ can be translated to some formula $\phi' \wedge \nu$ over some other signature Σ'. Section 8 links the result of Theorem 6 to context unification plus tree regular constraints.

6 Non-Intervenance Property

The idea behind our translation is to eliminate lambda bindings from the lambda-binding and parallelism constraints, by naming the variable binders. This means

that we want to obtain similar parallelism constraints that use *named* labels lam_u and var_u, instead of *anonymous* labels lam and var and lambda-binding constraints.

In order to avoid undesired variable capture, we would like to associate different names to different lambda binders. But unfortunately we cannot always do so in the presence of parallelism: corresponding lam-nodes have to carry the same label lam_u and corresponding var-nodes the same label var_u.

Given that we cannot freely assign fresh names, we are faced with the danger of capturing and have to avoid it. The simplest idea would be to forbid trees where some node with label lam_u intervenes between any two other nodes with labels lam_u and var_u. This restriction can be easily expressed by a closed first-order dominance formula or could also be directly checked by a tree automaton in some tree regular constraint.

Unfortunately, the above restriction is too restrictive and thus not correct, as illustrated by the following example:

$$\mathsf{lam}_u(@(\mathsf{lam}_u(@(a, \mathsf{var}_u)), \mathsf{var}_u))$$

It can always happen that a corresponding lam_u takes place above of a binding lam_u-node, so that the binding lam_u intervenes between the corresponding lam_u-node and one of the var_u-nodes bound by it. Thus we need a refined *non-intervenance property* stating that no corresponding lam_u may intervene between a lam_u-node and one of the var_u-nodes it binds.

Example 7. The following parallelism constraint that is drawn in Fig. 5 is unsatisfiable:

$$X \lhd^+ Y \lhd^+ X' \ \wedge \ X/X' \sim Y/Y' \ \wedge \ Y \lhd^+ V \ \wedge \ \lambda(V) = X$$

This will be proved by Lemma 8. The problem is that lam-node Y must correspond to X but intervene between X and the var-node V that X binds.

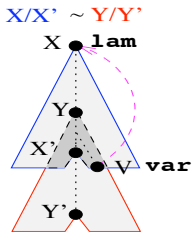

Fig. 5. Intervenance

Lemma 8. *Let (τ, λ) be a lambda structure with parallel segments σ and σ' that correspond via the correspondence function c. For all π with $\lambda(\pi) \in \mathsf{nodes}^-_\tau(\sigma)$ it is not the case that $\lambda(\pi) \lhd^+ c(\lambda(\pi)) \lhd^+ \pi$.*

Proof. We suppose that $\lambda(\pi) \in \mathsf{nodes}^-_\tau(\sigma)$ and $\lambda(\pi) \lhd^+ c(\lambda(\pi)) \lhd^+ \pi$ and derive a contradiction. The segments σ and σ' must overlap such that the root of σ dominates that of σ' properly.

$$\mathsf{root}(\sigma) \lhd^+ \mathsf{root}(\sigma')$$

Notice that π must belong to the inner nodes of segment σ, $\pi \in \mathsf{nodes}^-_\tau(\sigma)$, since otherwise $\lambda(\pi)$ would be a **hanging binder**.

$$\mathsf{intervene}_{\mathsf{lam}_u}(Y,X) \quad = \quad \exists Z \exists Z'.\ Y \lhd^+ Z \lhd^+ X \ \wedge\ Z{:}\mathsf{lam}_u(Z')$$
$$\mathsf{bind}_u(X,Y) \ = \ \exists Z\ (Y{:}\mathsf{lam}_u(Z) \ \wedge\ Z \lhd^* X \ \wedge\ X{:}\mathsf{var}_u) \ \wedge\ \neg \mathsf{intervene}_{\mathsf{lam}_u}(Y,X)$$

Fig. 6. Non-intervenance and lambda binding

Now suppose π does not belong to the inner nodes of the lower segment $\pi \notin \mathsf{nodes}^-_\tau(\sigma')$. First of all, notice that π must be dominated by a hole of σ' since otherwise $\pi \perp \mathsf{root}(\sigma')$ and the lemma would follow trivially.

We also know that $c(\lambda(\pi)) \lhd^+ \pi$ and belongs to the inner nodes of segments σ and σ', therefore we can apply axiom **internal binder** and we get that $c(\lambda(\pi)) = \lambda(c(\pi))$, to avoid **hanging binder** axiom violation, $c(\pi)$ must belong to the inner nodes of segments σ and σ', that corresponds to the next case.

Consider now the case that π also belongs to the inner nodes of the lower segment $\pi \in \mathsf{nodes}^-_\tau(\sigma')$. We prove the following property inductively and thus derive a contradiction: For all $\pi \in \mathsf{nodes}^-_\tau(\sigma) \cap \mathsf{nodes}^-_\tau(\sigma')$ it is impossible that:

$$\lambda(\pi) \lhd^+ c(\lambda(\pi)) \lhd^+ \pi.$$

The proof is by well-founded induction on the length of the word π.

1. Case $\mathsf{root}(\sigma') \lhd^* \lambda(\pi) \lhd^+ c(\lambda(\pi))$. Let $\pi' = c^{-1}(\pi)$ be an inner node of σ. The length of the word π' is properly smaller than the length of π. Since π' belongs to the inner nodes of σ, the axiom for **internal binder** can be applied to the correspondence function c yielding $c(\lambda(\pi')) = \lambda(c(\pi'))$ and thus $c(\lambda(\pi')) = \lambda(\pi)$. The node $\lambda(\pi')$ must properly dominate both $c(\lambda(\pi'))$ and π'. The address (length) of $c(\lambda(\pi'))$ is smaller than that of π', so that:

   $$\lambda(\pi') \lhd^+ c(\lambda(\pi')) \lhd^+ \pi'$$

   This is impossible as stated by induction hypothesis applied to π'.
2. Case $\lambda(\pi) \lhd^+ \mathsf{root}(\sigma') \lhd^* c(\lambda(\pi))$. Let $\pi' = c^{-1}(\pi)$ be an inner node of σ. Since π is externally bound outside of σ', the axiom for **external binder** applies to the inverse correspondence function c^{-1} by Lemma 5 and yields $\lambda(\pi') = \lambda(\pi)$. By now, π' is internally bound in σ. The axiom for **internal binder** applied to correspondence function c yields: $c(\lambda(\pi')) = \lambda(c(\pi'))$ which is $c(\lambda(\pi)) = \lambda(\pi)$. This clearly contradicts $\lambda(\pi) \lhd^+ c(\lambda(\pi))$.

7 Elimination of Lambda Binding Constraints

We now give a translation that eliminates lambda binding literals while preserving satisfiability. The procedure is highly non-deterministic and introduces first-order dominance formulas to express consistent naming of bound variables.

We impose the non-intervenance property of Lemma 8 when expressing the lambda binding predicate $\mathsf{bind}_u(X,Y)$ in Fig. 6. This is defined by using the predicate $\mathsf{intervene}_{\mathsf{lam}_u}(Y,X)$, that we express via first-order dominance formulas that some lam_u-node intervenes between X and Y.

Guessing Correspondence Classes. Corresponding lam and var nodes clearly have to carry the same node labels. But we have to be a little more careful since we may have several correspondence functions for several pairs of parallel segments. We say that two nodes are in the same correspondence class for a given set of correspondence functions $\{c_1, \ldots, c_n\}$ if they belong to the symmetric, reflexive, transitive closure of the common graph of these functions.

Consider for instance tree-structure τ of Fig 7, and correspondence functions c_1 and c_2 defined by $c_1(11) = 12$ and $c_2(12) = 2$. Then, $\mathcal{C}_{\tau,\{c_1,c_2\}} = \{(11,11),(11,12),(11,2),(12,11),(12,12),(12,2),(2,11),(2,12),(2,2)\}$ is the symmetric, reflexive and transitive closure of $\{c_1, c_2\}$ in τ.

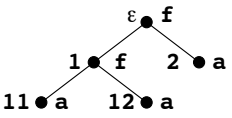

Fig. 7. $f(f(a,a),a)$

Given a parallelism and lambda binding constraint $\phi \wedge \mu$ we consider the set of correspondence functions for pairs of segments that are required to be parallel in ϕ. But how can we know whether two variables of $\phi \wedge \mu$ will denote nodes in the same correspondence class? We want to guess an equivalence relation e between variables of $\phi \wedge \mu$ depending on a solution τ, α for $\phi \wedge \mu$, such that for any two variables X and Y of $\phi \wedge \mu$, $(X,Y) \in e \iff (\alpha(X), \alpha(Y)) \in \mathcal{C}_{\tau,\{c_1,\ldots,c_n\}}$, where $\{c_1, \ldots, c_n\}$ are the correspondence functions for pairs of segments that are required to be parallel in ϕ. We cannot do it a priori, but we simply guess it as there are only finitely many possibilities for the finitely many variables.

Translation. We want to guess one of the possible partitions into correspondence classes for variables of ϕ. Instead, we simply guess an equivalence relation on the variables of ϕ, and as our proofs will show, we don't have to express that equivalent variables denote values in the same correspondence class. Let

$$\mathsf{equ}_\phi = \{e \mid e \subseteq \mathsf{vars}(\phi) \times \mathsf{vars}(\phi) \text{ equivalence relation}\}$$

be the set of possible equivalence relations on the variables of ϕ. We write $e(X)$ for the equivalence class of some variable $X \in \mathsf{vars}(\phi)$ with respect to e, but consider equivalence classes of distinct equivalence relations to be distinct. Let

$$\mathsf{names}_e = \{e(X) \mid X \in \mathsf{vars}(\phi)\}$$

be the set names of e which contains all equivalences classes of e. Note that names_e is finite for all $e \in \mathsf{equ}_\phi$, and that names_e and $\mathsf{names}_{e'}$ are disjoint for distinct equivalence classes e and e'.

We now fix a constraint $\Phi = \phi \wedge \mu$ and guess an equivalence relation $e \in \mathsf{equ}_\phi$ that determines the translation $[\ .\ ]_e$ presented in Fig. 8. This translation maps to a parallelism constraint plus first order dominance formulas $\phi' \wedge \nu$ over the following signature Σ_ϕ which extends Σ with finitely many symbols:

$$\Sigma_\phi = \Sigma \uplus \{\mathsf{lam}_u, \mathsf{var}_u \mid u \in \mathsf{names}_e,\ e \in \mathsf{equ}(\phi)\}$$

The literal $\lambda(X) = Y$ is translated to $\mathsf{bind}_{e(Y)}(X,Y)$ as explained before. This ensures that all corresponding nodes in e are translated with the same name

$$
\begin{aligned}
[\lambda(X){=}Y]_e &= \mathsf{bind}_{e(Y)}(X,Y)\\
[\lambda^{-1}(Y){\subseteq}\{X_1,\ldots,X_n\}]_e &= \forall X.\mathsf{bind}_{e(Y)}(X,Y) \rightarrow \vee_{i=1}^{n} X{=}X_i\\
[Y{:}\mathsf{lam}(Z)]_e &= Y{:}\mathsf{lam}_{e(Y)}(Z)\\
[X{:}\mathsf{var}]_e &= \exists Y.\bigvee_{\{Z \mid Z:\mathsf{lam}(Z')\in\phi\}} \mathsf{bind}_{e(Z)}(X,Y)\\
[Y{:}f(Y_1\ldots,Y_n)]_e &= Y{:}f(Y_1\ldots,Y_n) \qquad \text{if } f \notin \{\mathsf{lam},\mathsf{var}\}\\
[X{\lhd^*}Y]_e &= X{\lhd^*}Y\\
[S_1 \sim S_2]_e &= S_1 \sim S_2 \ \wedge\ \mathsf{external{-}binder}_e(S_1,S_2) \ \wedge\\
&\quad\ \mathsf{no{-}hang{-}binder}_e(S_1) \ \wedge\ \mathsf{no{-}hang{-}binder}_e(S_2)\\
[\Phi_1 \ \wedge\ \Phi_2]_e &= [\Phi_1]_e \ \wedge\ [\Phi_2]_e
\end{aligned}
$$

Fig. 8. Naming variable binder for correspondence classes e

$$
\begin{aligned}
&\mathsf{inside}(X,\ Y/Y_1,\ldots,Y_n) \quad = \quad Y{\lhd^*}X \ \wedge\ (\bigvee_{i\in\{1..n\}} X{\lhd^+}Y_i)\\
&\mathsf{root}(X,\ Y/Y_1,\ldots,Y_n) \quad = \quad X{=}Y\\
&\mathsf{no{-}hang{-}binder}_e(S) \quad = \quad \bigwedge_{u\in\mathsf{names}_e} \mathsf{no{-}hang{-}binder}_u(S)\\
&\mathsf{no{-}hang{-}binder}_u(S) \quad = \quad \neg(\exists Y \exists Z.\ \mathsf{bind}_u(Y,Z) \ \wedge\ \neg\mathsf{inside}(Y,S) \ \wedge\ \mathsf{inside}(Z,S))\\
&\mathsf{external{-}binder}_e(S_1,S_2) \quad = \quad \bigwedge_{u\in\mathsf{names}_e} \mathsf{external{-}binder}_u(S_1,S_2)\\
&\mathsf{external{-}binder}_u(S_1,S_2) \quad =\\
&\qquad \forall Z_1 \forall Z_2 \forall Y.\ (\mathsf{bind}_u(Z_1,Z_2) \ \wedge\ \mathsf{inside}(Z_1,S_1) \ \wedge\ \neg\mathsf{inside}(Z_2,S_1) \ \wedge\ \mathsf{root}(Y,S_2))\\
&\qquad\qquad \rightarrow (Z_2{\lhd^*}Y \ \wedge\ \neg\mathsf{intervene}_{\mathsf{lam}_u}(Z_2,Y))\\
&\mathsf{no{-}free{-}var}_e \quad = \quad \bigwedge_{u\in\mathsf{names}_e} \forall X.\ X{:}\mathsf{var}_u \rightarrow (\ \exists Y \exists Z.\ Y{:}\mathsf{lam}_u(Z) \ \wedge\ Y{\lhd^*}X\ )
\end{aligned}
$$

Fig. 9. Auxiliary predicates

$e(Y)$. The axioms about **external binding** and **no hanging binder** are stated by first-order dominance formulas in the translation of parallelism literals. The first-order formulas are defined in Fig. 9. Note that the axiom of **internal binding** will always be satisfied without extra requirements.

We have to ensure that all var_u-nodes in solutions of translated constraints will be bound by some lam_u-node. Let $\mathsf{no{-}free{-}var}_e$ be as defined in Fig. 9. We then define the complete translation $[\Phi]$ by:

$$[\Phi] = \bigvee_{e\in\mathsf{equ}_\phi} [\Phi]_e \ \wedge\ \mathsf{no{-}free{-}var}_e$$

We want to prove that our translation preserves satisfiability. We split the proof into the following two Lemmas:

Lemma 9. *Let Φ be a conjunction of a parallelism and lambda binding constraint and $e \in \mathsf{equ}(\Phi)$ an equivalence relation on $\mathsf{vars}(\Phi)$. If $[\Phi]_e \wedge \mathsf{no{-}free{-}var}_e$ is satisfiable then Φ is satisfiable.*

Let τ be a tree structure and $\alpha : \mathsf{vars} \rightarrow \mathsf{nodes}_\tau$ an assignment with

$$\tau, \alpha \models [\Phi]_e \ \wedge\ \mathsf{no{-}free{-}var}_e$$

We now define a lambda structure $(p(\tau), \lambda)$ of signature Σ by projecting labels away. The nodes of $p(\tau)$ are the nodes of τ. Let projection $\mathsf{proj} : \Sigma_\phi \rightarrow \Sigma$ be

the identity function except that $\mathsf{proj}(\mathsf{lam}_u) = \mathsf{lam}$ and $\mathsf{proj}(\mathsf{var}_u) = \mathsf{var}$ for any $u \in \mathsf{names}_e$. The labels of $p(\tau)$ satisfy for all $\pi \in \mathsf{nodes}_\tau$:

$$p(\tau)(\pi) = \mathsf{proj}(\tau(\pi))$$

We define the lambda binding function $\lambda : p(\tau)^{-1}(\mathsf{var}) \to p(\tau)^{-1}(\mathsf{lam})$ as follows: Let π be a node such that $p(\tau)(\pi) = \mathsf{var}$. There exists a unique name u such that $\tau(\pi) = \mathsf{var}_u$. We define $\lambda(\pi)$ to be the lowest ancestor of π that is labeled by lam_u. This is the unique node in $p(\tau)$ that satisfies $\mathsf{bind}_u(\pi, \lambda(\pi))$. It exists since we required $\tau, \alpha \models \mathsf{no{-}free{-}var}_e$.

It remains to prove that $p(\tau), \lambda, \alpha$ is indeed a solution of Φ, i.e. whether $(p(\tau), \lambda), \alpha$ satisfies all literals of Φ.

- $X \lhd^* Y$ in Φ: The dominance relation of τ coincides with that of $p(\tau)$. Since $\tau, \alpha \models X \lhd^* Y$ it follows that $(p(\tau), \lambda), \alpha \models X \lhd^* Y$.
- $X{:}f(X_1, \ldots, X_n)$ in Φ where $f \notin \{\mathsf{lam}, \mathsf{var}\}$. The labeling relation of τ coincides with that of $p(\tau)$, so there is no difference again.
- $X{:}\mathsf{var}$ in Φ: Notice that $\mathsf{bind}_{e(Y)}(X, Y)$ enforces X to be a $\mathsf{var}_{\mathsf{e(Y)}}$-labeled node in $[\Phi]_e$, which implies $(p(\tau), \lambda), \alpha \models X{:}\mathsf{var}$ by the definition of p.
- $X{:}\mathsf{lam}(Z)$ in Φ: Now, the literal $X{:}\mathsf{lam}_{e(X)}(Z)$ belongs to $[\Phi]_e$. Thus, $\tau, \alpha \models X{:}\mathsf{lam}_{e(X)}(Z)$ which implies $(p(\tau), \lambda), \alpha \models X{:}\mathsf{lam}(Z)$ by the definition of p.
- $\lambda(X){=}Y$ in Φ: Let $\tau, \alpha \models [\lambda(X){=}Y]_e$. By definition of the translation $[\lambda(X){=}Y]_e$ this means that $\tau, \alpha \models \mathsf{bind}_{e(Y)}(X, Y)$. In particular, it follows that $\alpha(Y)$ is the lowest $\mathsf{lam}_{e(Y)}$-labeled ancestor of the $\mathsf{var}_{e(Y)}$-labeled node $\alpha(X)$. The definition of the lambda-binding relation of $p(\tau)$ yields $(p(\tau), \lambda), \alpha \models \lambda(X){=}Y$ as required.
- $\lambda^{-1}(Y) \subseteq \{X_1, \ldots, X_n\}$ in Φ: the proof for this literal follows straightforward using similar arguments as for the previous one.

Consider at last, $S_1 {\sim} S_2$ in Φ: This is the most complicated case. If τ, α satisfies this literal then clearly, $(p(\tau), \lambda), \alpha$ satisfies the correspondence conditions for all labeling and children relations. We have to verify that $(p(\tau), \lambda)$ also satisfies the conditions of parallel binding. Let $c : \mathsf{nodes}^-_\tau(\alpha(S_1)) \to \mathsf{nodes}^-_\tau(\alpha(S_2))$ be the correspondence function between $\alpha(S_1)$ and $\alpha(S_2)$ which exists since $\tau, \alpha \models [\Phi]_e$.

Internal binder. Let $\lambda(\pi_1){=}\pi_2$ for some $\pi_1, \pi_2 \in \mathsf{nodes}^-_\tau(\alpha(S_1))$. By definition of λ, there exists a name u such that $\tau(\pi_1) = \mathsf{var}_u$ and π_2 is the lowest node above π_1 with $\tau(\pi_2) = \mathsf{lam}_u$. Since the labels of the nodes on the path between π_1 and π_2 are equal to the labels of the nodes of the corresponding path from $c(\pi_1)$ to $c(\pi_2)$ it follows that $\tau(c(\pi_1)) = \mathsf{var}_u$, $\tau(c(\pi_2)) = \mathsf{lam}_u$ and that no node in between is labeled with lam_u. Thus, $\lambda(c(\pi_1)) = c(\pi_2)$.

External binder. Suppose that $\lambda(\pi_1){=}\pi_2$ for two nodes $\pi_1 \in \mathsf{nodes}^-_\tau(\alpha(S_1))$ and $\pi_2 \notin \mathsf{nodes}^-_\tau(\alpha(S_1))$. There exists a name u such that $\tau(\pi_1) = \mathsf{var}_u$ and π_2 is the lowest ancestor of π_1 with $\tau(\pi_2) = \mathsf{lam}_u$. By correspondence, it follows that $\tau(c(\pi_1)) = \mathsf{var}_u$ and that no lam_u-node lies on the path form the root of segment $\alpha(S_2)$ to $c(\pi_1)$. The predicate $\mathsf{external{-}binder}_u(S_1, S_2)$ requires that π_2 dominates that root of $\alpha(S_2)$ and that no lam_u-node intervenes on the path from π_2 to that root. Thus, π_2 is the lowest ancestor of $c(\pi_1)$ that satisfies $\tau(\pi_2) = \mathsf{lam}_u$, i.e. $\lambda(c(\pi_1)) = \pi_2$.

No hanging binder. Let S be either of the segment terms S_1 or S_2. Suppose that $\lambda(\pi_1)=\pi_2$ for some nodes $\pi_1 \notin \mathsf{nodes}^-_\tau(S)$ and $\pi_2 \in \mathsf{nodes}^-_\tau(S)$. There exists a name $u \in \mathsf{names}_e$ such that $\tau(\pi_1) = \mathsf{var}_u$ and π_2 is the lowest ancestor of π_1 with $\tau(\pi_2) = \mathsf{lam}_u$. This contradicts that τ, α solves $\mathsf{no{-}hang{-}binder}_u(S)$ as required by $[S_1{\sim}S_2]_e$.

Lemma 10. *If Φ has a solution whose correspondence classes induce the equivalence relation e then $[\Phi]_e \wedge \mathsf{no{-}free{-}var}_e$ is satisfiable.*

Let Φ be a conjunction of a parallelism and lambda binding constraint over signature Σ and $(\tau, \lambda), \alpha$ a solution of it. Let $\{c_1, \ldots, c_n\}$ be the correspondence functions for the parallel segments $\alpha(S) \sim \alpha(S')$ where $S{\sim}S'$ belongs to ϕ. Let $c \subseteq \mathsf{nodes}_\tau \times \mathsf{nodes}_\tau$ be the reflexive, symmetric, and transitive closure of $\{c_1, \ldots, c_n\}$, and $e \in \mathsf{equ}(\Phi)$ be the relation $\{(X, Y) \mid (\alpha(X), \alpha(Y)) \in c\}$.

We define $\mathsf{tree}_e(\tau, \lambda)$ as a tree over the extended signature Σ_ϕ whose nodes are those of τ and whose labeling function satisfies for all $\pi \in \mathsf{nodes}_\tau$ that:

$$\mathsf{tree}_e(\tau,\lambda)(\pi) = \begin{cases} \mathsf{lam}_{e(X)} & \text{if } (\pi, \alpha(X)) \in c, \tau(\pi) = \mathsf{lam},\ X \in \mathsf{vars}(\Phi) \\ \mathsf{var}_{e(X)} & \text{if } (\lambda(\pi), \alpha(X)) \in c, X \in \mathsf{vars}(\Phi) \\ \tau(\pi) & \text{otherwise} \end{cases}$$

We now prove that $\mathsf{tree}_e(\tau, \lambda), \alpha$ solves $[\Phi]_e$, i.e. all of its conjuncts. This can be easily verified for dominance, labeling, and parallelism literals in $[\Phi]_e$. Notice in particular that corresponding lam-nodes in τ are assigned the same labels in $\mathsf{tree}_e(\tau, \lambda)$. Next, we consider the first-order formulas introduced in the translation of lambda binding and parallelism literals.

1. Case $\mathsf{bind}_{e(Y)}(X, Y)$ in $[\Phi]_e$. This requires either $\lambda(X){=}Y$ or $\lambda^{-1}(Y) \subseteq \{X_1, \ldots, X_n\}$ or $X{:}\mathsf{var}$ in Φ. Let's consider the first case, the corresponding cases of $\lambda^{-1}(Y) \subseteq \{X_1, \ldots, X_n\}$ in Φ, and of $X{:}\mathsf{var}$ in Φ are quite similar. It then clearly holds that $\mathsf{tree}_e(\tau, \lambda)(\alpha(X)) = \mathsf{var}_{e(Y)}$ and $\mathsf{tree}_e(\tau, \lambda)(\alpha(Y)) = \mathsf{lam}_{e(Y)}$. Furthermore $\alpha(Y) \lhd^+ \alpha(X)$. It remains to show for $\mathsf{tree}_e(\tau, \lambda)$ that no $\mathsf{lam}_{e(Y)}$-node intervenes between $\alpha(X)$ and $\alpha(Y)$. We do this by contradiction. Suppose there exists π such that $\alpha(Y) \lhd^+ \pi \lhd^+ \alpha(X)$ and $\mathsf{tree}_e(\tau, \lambda)(\pi) = \mathsf{lam}_{e(Y)}$. By definition of $\mathsf{tree}_e(\tau, \lambda)$ there exists Z such that $(\pi, \alpha(Z)) \in c$ and $e(Y) = e(Z)$. Hence $(\alpha(Y), \alpha(Z)) \in c$ and thus $(\pi, \alpha(Y)) \in c$. But this is impossible by the non-intervenance property shown in Lemma 8: no lam-node such as π that corresponds to $\alpha(Y)$ intervene between $\alpha(Y)$ and the var-node $\alpha(X)$ bound by it.
2. Case $\mathsf{external{-}binder}_u(S_1, S_2)$ in $[\Phi]_e$ where $S_1{\sim}S_2$ in Φ and $u \in \mathsf{names}_e$. By contradiction. Suppose that there exist $\pi_1 \in \mathsf{nodes}_\tau(\alpha(S_1))$, $\pi_2 \notin \mathsf{nodes}_\tau(\alpha(S_1))$ such that $\mathsf{tree}_e(\tau, \lambda)(\pi_1) = \mathsf{var}_u$ and π_2 is the lowest ancestor of π_1 with $\mathsf{tree}_e(\tau, \lambda)(\pi_2) = \mathsf{lam}_u$. Furthermore, assume either not $\pi_2 \lhd^* \mathsf{root}(\alpha(S_2))$ or $\mathsf{intervene}_{\mathsf{lam}_u}(\pi_2, \mathsf{root}(\alpha(S_2)))$. The first choice is impossible since the binding axioms were violated otherwise. (The correspondent of an externally bound node must be bound externally). Let π_1' be the correspondent of π_1 with respect to the parallel segment $\alpha(S_1){\sim}\alpha(S_2)$. By Lemma 8 we know

that no lam-node corresponding to π_2 can intervene between π_2 and π' and thus between π_2 and $\mathsf{root}(S_2)$. This also contradicts the second choice: $\mathsf{intervene}_{\mathsf{lam}_u}(\pi_2, \mathsf{root}(\alpha(S_2)))$.

3. $\mathsf{no{-}hang{-}binder}_e(S)$ in $[\Phi]_e$ where S is either S_1 or S_2 and $S_1 \sim S_2$ in Φ. Let's proceed by contradiction. If it is not satisfied by $\mathsf{tree}_e(\tau, \lambda), \alpha$, then there must exist a name $u \in \mathsf{names}_\phi$ and two nodes π_1, π_2 such that $\mathsf{tree}_e(\tau, \lambda)(\pi_1) = \mathsf{lam}_u$ and $\mathsf{tree}_e(\tau, \lambda)(\pi_2) = \mathsf{var}_u$, even more $\pi_1 \in \mathsf{nodes}_\tau(\alpha(S))$, $\pi_2 \notin \mathsf{nodes}_\tau(\alpha(S))$ and there not exists a third node π_3 between π_1 and π_2. Then, by Lemma 8, π_1 can not be a corresponding node of the lambda binding node of π_2, therefore, by definition of $\mathsf{tree}_e(\tau, \lambda)$ $\lambda(\pi_1) = \pi_2 \in \lambda$, but this is impossible because $(\tau, \lambda), \alpha$ must satisfy the **no hanging binder** condition.
4. Finally, we prove that $\mathsf{tree}_e(\tau, \lambda), \alpha$ satisfies $\mathsf{no{-}free{-}var}_e$. This is simple.

Proposition 11. *A parallelism and lambda binding constraint $\phi \wedge \mu$ is satisfiable if and only if its translation $[\phi \wedge \mu]$ is.*

8 Context Unification with Tree Regular Constraints

CU is the problem of solving context equations over the algebra of trees and contexts. Let the *hole marker* $\bullet$ be a new symbol. A *context* γ over Σ is a tree over $\Sigma \cup \{\bullet\}$ such that the hole maker occurs at most once. For instance, $C = f(\bullet, a)$ is a context. The application of context C to a tree t over Σ, noted $f(\bullet, a)(t)$ is the tree $f(t, a)$ obtained from C by replacing the hole marker $\bullet$ by t.

In CU we may have first-order variables x denoting trees over Σ and context variables C that denote contexts. The following context equations express the CLLS constraint in Fig 1 except for lambda binding:

$x = C(see@john@var)$	C is the context of the verb in readings x
$C = C_1(exists@taxi@lam(C_2))$	C contains the quantifier
$C = C_3(and@C_4@C_5(bill))$	and the conjunction
$C_5 = C_2(see@\bullet@var)$	bill and john do the same

We can enrich context equations by imposing tree regular constraints where $\mathcal{A}$ is a tree-automaton over Σ.

$$x \in L(\mathcal{A})$$

Tree regular constraints can express MSO formulas over dominance constraints, even in the presence of parallelism constraints.

Theorem 12 (Theorem 11 of [16]). *Conjunctions of parallelism constraints with MSO dominance formulas have the same expressiveness as parallelism constraints with tree regular constraints.*

Finally, one can translate parallelism constraints to CU according to [14]. The proof of this paper can be easily extended to tree regular constraints:

Proposition 13 (Extension of [14]). *Parallelism with tree regular constraints have the same expressiveness as CU with tree regular constraints.*

Theorem 14. *Conjunctions of parallelism and lambda binding constraints are satisfaction equivalent to CU equations with tree regular constraints.*

This is a corollary to Theorems 6 and 12 and Proposition 13.

9 Limitations

It is proposed in [2] to extend CLLS by group parallelism in order to deal with beta reduction constraints. The question is now whether *group parallelism* can be expressed in context unification with tree regular constraints. This is a relation between groups of segments $(S_1, \ldots, S_n) \sim (S'_1, \ldots, S'_n)$ that behaves like a conjunction of parallelism literals $\wedge_{i=1}^{n} S_i \sim S'_i$ but such that hanging binders are defined with respect to groups of segments $(S_1, \ldots, S_n)$ resp. $(S'_1, \ldots, S'_n)$.

Unfortunately, we cannot extend the encodings of the present paper. The problem is that group parallelism does not satisfy the non-intervenance property as stated for ordinary parallelism in Lemma 8. Indeed, it is not always possible to name variables consistently in the presence of group parallelism, so that corresponding binder of parallel groups are named alike. In other words, binding parallelism cannot be reduced to tree parallelism by naming binders. This is illustrated by the lambda structure in Fig. 10 which satisfies the group parallelism constraint:

$$(X_1/X_2, X_4/X_5) \sim (X_2/X_3, X_3/X_4)$$

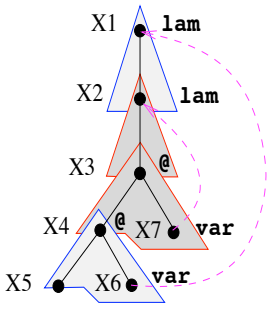

Fig. 10. Group Parallelism

Even though the lam-node X_2 corresponds to X_1, X_2 intervenes between X_1 and its bound var-node X_6. We thus cannot name these corresponding nodes alike.

10 Conclusion and Future Work

We have shown that the lambda-binding constraints of CLLS can be expressed in CU with tree regular constraints. The proof relies on the non-intervenance property of parallel lambda binding in CLLS that we establish. We leave it open whether all of CLLS can be translated into CU, in particular group parallelism, beta reduction, or anaphoric binding constraints. Another open question is how to characterize the decidable well-nested fragment of parallelism constraints [8] in a decidable fragment of CU.

References

1. R. Backofen, J. Rogers, and K. Vijay-Shanker. A first-order axiomatization of the theory of finite trees. *Journal of Logic, Language, and Information*, 4:5–39, 1995.
2. Manuel Bodirsky, Katrin Erk, Alexander Koller, and Joachim Niehren. Underspecified beta reduction. In *ACL*, pages 74–81, 2001.
3. Hubert Comon. Completion of rewrite systems with membership constraints. *Symbolic Computation*, 25(4):397–453, 1998. Extends on a paper at ICALP'92.
4. Denys Duchier and Claire Gardent. Tree descriptions, constraints and incrementality. In *Computing Meaning*, Linguistics and Philosophy, pages 205–227. 2001.
5. Markus Egg, Alexander Koller, and Joachim Niehren. The constraint language for lambda structures. *Logic, Language, and Information*, 10:457–485, 2001.
6. Katrin Erk, Alexander Koller, and Joachim Niehren. Processing underspecified semantic representations in the constraint language for lambda structures. *Journal of Research on Language and Computation*, 1(1):127–169, 2002.
7. Katrin Erk and Joachim Niehren. Parallelism constraints. In *RTA'00*, volume 1833 of *LNCS*, pages 110–126, 2000.
8. Katrin Erk and Joachim Niehren. Well-nested parallelism constraints for ellipsis resolution. In *EACL*, pages 115–122, 2003.
9. Jordi Levy. Linear second-order unification. In *RTA*, volume 1103 of *LNCS*, pages 332–346, 1996.
10. Jordi Levy and Margus Veanes. On the undecidability of second-order unification. *Information and Computation*, 159:125–150, 2000.
11. Jordi Levy and Mateu Villaret. Linear second-order unification and context unification with tree-regular constraints. In *RTA*, pages 156–171, 2000.
12. Jordi Levy and Mateu Villaret. Context unification and traversal equations. In *RTA'01*, volume 2051 of *LNCS*, pages 167–184, 2001.
13. Mitchell P. Marcus, Donald Hindle, and Margaret M. Fleck. D-theory: Talking about talking about trees. In *Proceedings of the 21st ACL*, pages 129–136, 1983.
14. Joachim Niehren and Alexander Koller. Dominance constraints in context unification. In *3rd LACL'98 (Grenoble, France)*, volume 2014 of *LNAI*, 2001.
15. Joachim Niehren and Stefan Thater. Bridging the gap between underspecification formalisms: Minimal recursion semantics as dominance constraints. In *41st Meeting of the Association of Computational Linguistics*, pages 367–374, July 2003.
16. Joachim Niehren and Mateu Villaret. Parallelism and tree regular constraints. In *LPAR'02*, volume 2514 of *LNAI*, pages 311–326, 2002.
17. Manfred Pinkal. Radical underspecification. In *Proceedings of the 10th Amsterdam Colloquium*, pages 587–606, 1996.
18. Manfred Schmidt-Schauß. A decision algorithm for distributive unification. *Theoretical Computer Science*, 208:111–148, 1998.
19. Manfred Schmidt-Schauß and Klaus U. Schulz. Solvability of context equations with two context variables is decidable. In *CADE-16*, LNAI, pages 67–81, 1999.
20. Manfred Schmidt-Schauß and Klaus U. Schulz. Decidability of bounded higher-order unification. In *Computer Science Logic*, volume 2471 of *LNAI*, 2002.
21. Stuart Shieber, Fernando Pereira, and Mary Dalrymple. Interaction of scope and ellipsis. *Linguistics & Philosophy*, 19:527–552, 1996.
22. J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81, 1967.
23. K. Vijay-Shanker. Using descriptions of trees in a tree adjoining grammar. *Computational Linguistics*, 18:481–518, 1992.