On Lambda Binding Constraints and Context Unification

Joachim Niehren[1] and Mateu Villaret[2]

[1] Programming Systems Lab, Universität des Saarlandes, Saarbrücken, Germany.
[2] IMA, Universitat de Girona, Campus de Montilivi, Girona, Spain.

Abstract. Lambda binding and parallelism constraints are the main ingredients of the constraint language for lambda structures. Parallelism constraints alone are known to have the same expressive power as the language of context unification. Lambda binding constraints however were never investigated in that framework. We show that lambda binding plus parallelism constraints can be expressed in context unification with tree regular constraints.

1 Introduction

The *constraint language for lambda structures* (CLLS) is a first-order language by which to talk about lambda terms [6, 7]. These are modeled as lambda structures, i.e. trees with additional lambda binding edges. CLLS can talk about the nodes of a lambda structure, children, ancestor, and lambda binding; it can also speak about the parallelism relation between segments of lambda structure, which relates those segments that have the same tree and lambda binding structure.

We have argued so far that parallelism and lambda binding constraints are the main ingredients of CLLS. Parallelism constraints [8] subsume dominance constraints [14, 2, 1] which by themselves have a wide range of applications in computational linguistics, in syntax, semantics, and discourse (see i.g. [20, 5]). The full language of CLLS was introduced and still serves as a uniform framework for modeling underspecification in natural language semantics.

Context unification (CU) [4] extends first-order unification by second-order variables that denote contexts. Contexts are trees with finitely many holes. The segments of CLLS can be seen as occurrences of contexts. CU treats contexts as functions that fill the holes of the context when applied to a sequence of trees. CU is closely related with linear second-order unification [9, 12]. Whether CU is decidable is open but was often conjectured (see e.g. [13]). This is because various restrictions [9, 17, 18, 4, 19] make it decidable, while the analogous restrictions applied to second-order unification don't [10, 11].

It was shown in [15] that parallelism constraints are equivalent in expressive power to the language of CU. Lambda binding constraints, however, were never taken into account in this framework, and this was for good reasons: On the one hand side, it would be quite surprising if lambda binding could be expressed by using only parallelism constraints and thus by CU. On the other hand side, it

seems unlikely that adding lambda binding to parallelism constraints could raise undecidability, so this extension seemed of low interest with respect to the open decidability question for CU.

In this paper, we show that lambda binding and parallelism constraints can be expressed by CU with tree regular constraints [12]. This result is far from obvious. Its proof requires several encoding steps, one of which is based on a general but recent relationship between dominance logic and tree automata [16]. Another step relies on a new special property of parallel lambda binder that this paper exhibits for a first time.

If one believes in the conjecture of [12] that adding tree regular constraints to CU does not raise undecidability, then our result shows that adding lambda binding to parallelism constraints does neither.

We proceed as follows: Section 2 recalls the notions of tree structures, segments, and parallelism. Section 3 continues with lambda structures, where parallelism is restricted to to deal with parallel lambda binding. Section 4 defines the languages of dominance, parallelism, and lambda binding constraints. In Section 5 we present the special property of parallel lambda binding that our encodings will rely on. Sections 6 and 7 show how to eliminate lambda binding constraints through naming of lambda binder, but at the cost of adding first-order dominance formulas. Section 8 then explains why parallelism constraints plus first-order dominance formulas can be expressed by CU with tree regular constraints. The final section, section 9, discusses applications and limitations of our approach.

2 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 of τ to their function symbol labeling. If $\tau = f(\tau_1, \ldots, \tau_n)$ then we set:

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

If τ is a tree with $\pi \in \mathsf{nodes}_\tau$ then we write $\tau.\pi$ for the subtree of τ rooted by π. Furthermore, we write $\tau[\pi/\tau']$ for the tree that is 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 *children-labeling relation* $\pi{:}f(\pi_1, \ldots, \pi_n)$ holds for τ if the 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* $\pi \triangleleft^* \pi'$ 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* $\pi \triangleleft^+ \pi'$ holds for τ if $\pi \triangleleft^* \pi'$ but not $\pi = \pi'$ in τ. The *disjointness relation* $\pi \perp \pi'$ holds for τ if neither $\pi \triangleleft^* \pi'$ nor $\pi' \triangleleft^* \pi$ hold for 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 are those nodes of τ that lie between the root of σ and its holes:

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

Note that our notion of segment nodes generalize the notion of tree nodes properly: $\mathsf{nodes}_{\tau.\pi} = \mathsf{nodes}_\tau(\pi/)$ holds for all trees τ and nodes $\pi \in \mathsf{nodes}(\tau)$. The labels and children of holes do not belong anymore to the segment. We therefore define the *inner nodes of a segment* to be all those nodes that are not holes, i.e.:

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

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. Tree parallelism can be characterized equivalently by saying that parallel segments are occurrences of the same context (see Proposition 13 below).

3 Lambda Structures and Parallel Lambda Binding

Lambda structures represent lambda terms uniquely modulo consistent renaming of bound variables. They can be seen as tree structures that are decorated with lambda binding edges.

The signature Σ of lambda structures contains 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 and of first-order connectives. The binding function λ explicitly maps var-labeled to lam-labeled nodes.

For example, Fig. 1 shows a representation of the term $\lambda x.f(x)$. Here $\lambda(12) = \epsilon$.

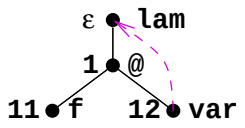

Fig. 1. The lambda structure of $\lambda x.f(x)$

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* $\lambda(\pi) \lhd^* \pi$ *holds for all* var*-nodes* π *in* τ*.*

We freely consider lambda structures logical structures that beside of the relations of tree structures have relations for lambda binding and inverse lambda binding. The inverse lambda binding relation $\lambda^{-1}(\pi_0)$=$\{\pi_1,...,\pi_n\}$ states that π_0 binds $\pi_1, \ldots, \pi_n$, and only those nodes are lambda-bound by π_0.

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:

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. Which of the two notions of parallelism is spoken about should always become clear from the respective context.

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. Suppose that $\pi, \lambda(\pi) \in \mathsf{nodes}_\tau^-(\sigma_2)$. Let $\pi' = c^{-1}(\pi)$ 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** 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 proof 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 assumed finite.

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. 2. Logical languages for tree and lambda structures

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}.

4 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 2. 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 to X_m.

A *dominance constraint* is a conjuction 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 expres existential quantification $\exists X.\nu$ and disjunction $\nu_1 \vee \nu_2$ 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 assigment* 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 $\alpha(X/X_1,...,X_n) = \alpha(X)/\alpha(X_1),...,\alpha(X_n)$ which may or may not be segments. Apart from this, we require as usual that a formula evaluates to true in all solution. We write $\tau,\alpha \models \psi$ if τ,α is a solution of ψ, and in similarly $(\tau,\lambda),\alpha \models \psi$. A formula is *satisfiable* if it has a solution.

Theorem 6. *The satisfiablity problem of parallelism plus lambda binding constraints $\phi \wedge \mu$ can be reduced in polynomial time to the satisfiability problem of parallelism constraints plus first-order dominance formulas $\phi' \wedge \nu$.*

Note that the signature is part of the input of the respective satisfiablity 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'. Sections 5-7 deals with proving this theorem. Section 8 links the result of Theorem 6 to context unification plus tree regular constraints.

5 Non-Intervenance Property

We now present a new property of parallelism in lambda structures that will be fundamental to our elimination of lambda binding constraints. It says that a corresponding lam-node cannot intervene between the lam-node and one of the var-nodes it binds. This statement is obvious for non-overlapping parallel segments but non obvious otherwise.

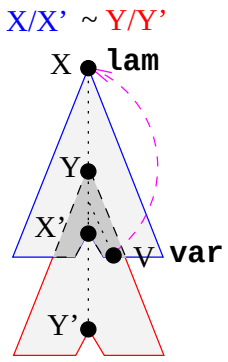

Fig. 3. Intervenance

Example 7. The parallelism constraint drawn in Fig. 3 is unsatisfiable as will be proved by Lemma 8 below. The problem is that lam-node Y corresponds to X but must intervene between X and one of its bound var-nodes, i.e. V.

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

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 properly such that root of σ dominates that of σ'.

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

We can assume that π belongs to the inner nodes of both segments. Notice that $\pi \in \mathsf{nodes}^-_\tau(\sigma)$ holds since otherwise $\lambda(\pi)$ would be a **hanging binder**. W.l.o.g. π also belongs to the inner nodes of the lower segment $\pi \in \mathsf{nodes}^-_\tau(\sigma')$ since otherwise $\pi \bot \mathsf{root}(\sigma')$ and the lemma would follow trivially.

We now prove the following property inductively and thus derive a contradiction: For all nodes $\pi \in \mathsf{nodes}^-_\tau(\sigma) \cap \mathsf{nodes}^-_\tau(\sigma')$ 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 $\lambda(\pi) \lhd^+ c(\lambda(\pi))$.

6 Elimination of Lambda Binding Constraints

We now give a translation that eliminates lambda binding literals in the presence of parallelism constraints while preserving satisfiability. The procedure is highly non-deterministic and will introduce first-order dominance formulas to express consistent binding.

6.1 Consistent Naming of Variable Binders

The idea behind our translation is to eliminate lambda binding by naming variable binder. This means that we want to use *named* labels lam_u and var_u, instead of *anonymous* labels lam and var.

In order to avoid variable capturing, we would like to introduce different names u, v, w for different binder $\mathsf{lam}_u, \mathsf{lam}_v, \mathsf{lam}_w$. But unfortunately we cannot

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

Fig. 4. Non-intervenance and lambda binding

always do so in the presence of parallelism: corresponding lam-nodes have to carry 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 were 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. The problem is that a corresponding lam-node can occur above of the lam-node it corresponds to. This can lead to the unwanted situation, 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.

Lemma 8 says that no corresponding lam-node can intervene between the lam-node it corresponds to and one of var-nodes it binds. This indicates a way out of that trouble: we have to require non-intervenance only when expressing lambda binding but not in general.

We define the binary predicate $\mathsf{intervene}_{\mathsf{lam}_u}(Y,X)$ through the the first-order dominance formula in Fig. 4, which holds in a tree structure τ if and only if some lam_u-node intervenes between the values of X and Y.

We can now express $\lambda(X){=}Y$ with binder name u through the predicate $\mathsf{bind}_u(X,Y)$ that is also given in Fig. 4. It requires X to be labeled with var_u, Y with lam_u, and no other lam_u-node to intervene between X and Y.

6.2 Guessing Correspondence Classes

We have said so far that corresponding lam and var nodes 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 set of correspondence function $\{c_1, \ldots, c_n\}$, if they are in the symmetric, reflexive, transitive closure of these functions.

Given a parallelism and lambda binding constraint $\phi \wedge \mu$ we will be interested in the set of correspondence functions for pair 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 cannot do it a priori, but we

simply guess it as there are only finitely many possibilities for the finitely many variables.

6.3 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. 5. 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 $e(Y)$. 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. 6. 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. 6. We then define the complete translation $[\Phi]$ by:

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

Proposition 9. *A parallelism and lambda binding constraint $\phi \wedge \mu$ is satisfiable if and only if its translation $[\phi \ \wedge \ \mu]$ is.*

7 Correctness and Completeness

We want to prove that our translation preserves satisfiability. We split the proof into Lemmas 10 and 11.

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

Fig. 5. Naming variable binder for correspondence classes e

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

Fig. 6. Auxiliary predicates

Lemma 10. *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.*

Proof. 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 $\lambda : p(\tau)^{-1}(\mathsf{var}) \rightarrow p(\tau)^{-1}(\mathsf{lam})$ 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. We distinguish all possible kinds of literals:

1. $X \lhd^* Y$ in Φ: The dominance relation of τ coincides with that of $p(\tau)$. Since $\tau, \alpha \models X \lhd^* Y$ it follows that $(p(\tau), \lambda), \alpha \models X \lhd^* Y$.
2. $X{:}f(X_1, \ldots, X_n)$ in Φ where $f \notin \{\mathsf{lam}, \mathsf{var}\}$. The children relation of τ coincides with that of $p(\tau)$, so there is no difference again.
3. $X{:}\mathsf{var}$ in Φ: Note that $X{:}\mathsf{var}$ is equivalent to $\exists Y.\lambda(X) = Y$.
4. $X{:}\mathsf{lam}(Z)$ 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 proj.
5. $\lambda(X){=}Y$ 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.
6. $\lambda^{-1}(Y) \subseteq \{X_1, \ldots, X_n\}$ in Φ: similar arguments than in previous case can be argued.
7. $S_1 {\sim} S_2$ 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 11. *If Φ has a solution whose correspondence classes induce the equivalence relation e then $[\Phi]_e \wedge \mathsf{no{-}free{-}var}_e$ is satisfiable.*

Proof. 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 $\lambda(X){=}Y$ in Φ (the corresponding translation of $\lambda^{-1}(Y) \subseteq \{X_1, \ldots, X_n\}$ in Φ, is 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 $\alpha(Y) \triangleleft^+ \alpha(X)$. 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 $\alpha(Y) \triangleleft^+ \pi \triangleleft^+ \alpha(X)$ 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 $\pi_2 \triangleleft^* \mathsf{root}(\alpha(S_2))$ 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 exteranlly 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, assume that 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.

8 Context Unification with Tree Regular Constraints

We have shown so far how to express lambda binding and parallelism constraints $\phi \wedge \mu$ by parallelism constraints with first-order dominance formulas $\phi \wedge \nu$. We can now apply Theorem 11 of [16]: parallelism constraints plus first-order dominance formulas $\phi \wedge \nu$ can be expressed in context unification plus tree regular constraints. The same result also holds with monadic second-order dominance formulas, but we don't need them here.

The encoding of dominance formulas through tree automata applies similar tricks as encoding the weak monadic logic WS2S into tree automata. This result can be lifted with another encoding trick to translate parallelism constraints plus dominance formulas to parallelism constraints with tree regular constraints of the form $\mathsf{tree}(X) \in \mathcal{L}(\mathcal{A})$ where $\mathcal{L}(\mathcal{A})$ is the regular language of some tree automaton $\mathcal{A}$ and $\mathsf{tree}(X)$ denotes the tree rooted by node X. Finally, one can translate parallelism plus tree regular constraints to context unification with tree regular constraints by extending on [15].

Theorem 12. *Every conjunction of parallelism and lambda binding constraints is satisfaction equivalent to some context unification problem with tree regular constraints, and this problem can be computed in polynomial time.*

As explained above, this theorem is a corollary to Theorem 6 of the present paper and Theorem 11 of [16]. But to apply the later, we have first to show that the definition of parallelism used there is equivalent to ours (Prop. 13 below).

Let $\{\bullet_1, \ldots, \bullet_n, \ldots\}$ be an infinite set of *hole markers*. A *context* γ over Σ is a tree over Σ and the set of hole markers, but such that each hole occurs at most once in the context. For instance, $f(\bullet_2, g(\bullet_1))$ is a context with two holes. A segment can be identified with an occurrence of a context: For every segment σ of a tree τ with n holes we uniquely define a context $\mathsf{context}_\tau(\sigma)$ with hole markers $\bullet_1, \ldots, \bullet_n$ as follows:

$$\mathsf{context}_\tau(\pi/\pi_1, \ldots, \pi_n) = (\tau[\pi_1/\bullet_1] \ldots [\pi_n/\bullet_n]).\pi$$

The order in which the substitutions are performed does not matter since all holes π_i of a segment are pairwise disjoint. Note also, that the root π is never removed from τ while substituting, since it must dominate all holes π_i.

Proposition 13. *A tree parallelism relation $\sigma_1 \sim \sigma_2$ holds between two segments of a tree τ if and only if both segments are occurrences of the same context, i.e.* $\mathsf{context}_\tau(\sigma_1) = \mathsf{context}_\tau(\sigma_2)$.

9 Applications and Limitations

It is proposed in [3] to extend CLLS by group parallelism in order to deal with beta reduction constraints. The question is now whether beta reduciton constraints can be expressed in context unification with tree regular constraints.

The simplest form of beta reduction constraints $\mathsf{beta}_n(X, Y)$ says that the lambda substructure rooted by node X beta reduces in a single step with a redex that binds at most n variables to the substructure rooted by node Y. This constraint is equivalent to say that the tree below X has the form $S_1(@(\mathsf{lam}(S_2(\mathsf{var}, \ldots, \mathsf{var}), S_3))$ where at most n occurrences of var-nodes are bound by the lam-node, while the tree below Y is of the shape $S_1(S_2(S_3, \ldots, S_3))$. As long as one ignores the precise rules of parallel binding, this can be easily expressed by lambda binding and parallelism constraints.

The problem is that var-nodes of the function body S_2 might possibly be bound in the context S_1. But then, S_1 would have hanging binders. The problem can be fixed by redefining hanging binder with respect to groups of segments instead with respect to individual segments, i.e. one introduces *group parallelism* $(S_1, \ldots, S_n) \sim (S'_1, \ldots, S'_n)$ which is like a conjunction of parallelism literals $\wedge_{i=1}^{n} S_i \sim S'_i$ but such that hanging binder 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 to also deal with group parallelism. The problem is that group parallelism does not satisfy the non-intervenance property as stated for ordinary parallelism in Lemma 8. This is illustrated in Fig. 7 where the group parallelism constraint:

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

holds. Even though the lam-node X_2 corresponds to X_1, it intervenes between X_1 and one of its bound var-nodes X_6.

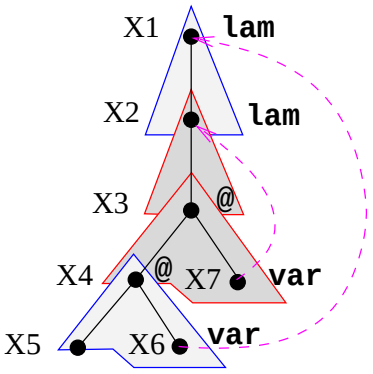

Fig. 7. Group Parallelism

Nevertheless we can still express restricted beta reduction constraints in context unification with tree regular constraints. We can describe a finite sequence of beta-reduction steps by group parallelism between disjoint groups – which satisfy the non-intervenance property – if we require that the terms in the sequence are all placed in disjoint positions of the surrounding lambda structure.

Another interesting question is, which binary relations on tree structures can be expressed by tree regular constraints in the way that we express the lambda binding relation. For instance, one might ask whether one can express the anaphoric binding relation of CLLS. This question is non-obvious so that we have to leave it open.

References

1. Ernst Althaus, Denys Duchier, Alexander Koller, Kurt Mehlhorn, Joachim Niehren, and Sven Thiel. An efficient algorithm for the configuration problem of dominance graphs. In *12th ACM-SIAM Symposium on Discrete Algorithms*, pages 815–824, 2001.
2. 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.
3. Manuel Bodirsky, Katrin Erk, Alexander Koller, and Joachim Niehren. Beta reduction constraints. In *RTA'01*, volume 2051 of *LNCS*, pages 31–46, 2001.
4. Hubert Comon. Completion of rewrite systems with membership constraints. *Symbolic Computation*, 25(4):397–453, 1998. Extends on a paper at ICALP'92.
5. Denys Duchier and Claire Gardent. Tree descriptions, constraints and incrementality. In *Computing Meaning, Volume 2*, volume 77 of *Studies In Linguistics And Philosophy*, pages 205–227. December 2001.
6. Markus Egg, Alexander Koller, and Joachim Niehren. The constraint language for lambda structures. *Journal of Logic, Language and Information*, 2001.
7. Katrin Erk, Alexander Koller, and Joachim Niehren. Processing underspecified semantic representations in the constraint language for lambda structures. *Journal of Logic, Language and Information*, 2000.
8. Katrin Erk and Joachim Niehren. Parallelism constraints. In *RTA'00*, volume 1833 of *LNCS*, pages 110–126, 2000.
9. Jordi Levy. Linear second-order unification. In *RTA'96*, volume 1103 of *LNCS*, pages 332–346, 1996.
10. Jordi Levy and Margus Veanes. On unification problems in restricted second-order languages. In *CSL'98*, Brno, Czech Republic, 1998.
11. Jordi Levy and Margus Veanes. On the undecidability of second-order unification. *Information and Computation*, 159:125–150, 2000.
12. Jordi Levy and Mateu Villaret. Linear second-order unification and context unification with tree-regular constraints. In *RTA'00*, volume 1833 of *LNCS*, pages 156–171, 2000.
13. Jordi Levy and Mateu Villaret. Context unification and traversal equations. In *RTA'01*, volume 2051 of *LNCS*, pages 167–184, 2001.
14. 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.
15. Joachim Niehren and Alexander Koller. Dominance constraints in context unification. In *LACL'98*, volume 2014 of *LNAI*, 2001.
16. Joachim Niehren and Mateu Villaret. The monadic second-order dominance logic and parallelism constraints, January 2002. Submitted. Available at `http://www.ps.uni-sb.de/Papers`.
17. Manfred Schmidt-Schauß. An algorithm for distributive unification. In *RTA'96*, volume 1103 of *LNCS*, pages 287–301, 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. K. Vijay-Shanker. Using descriptions of trees in a tree adjoining grammar. *Computational Linguistics*, 18:481–518, 1992.