Beta Reduction Constraints

Manuel Bodirsky, Katrin Erk, Alexander Koller, Joachim Niehren

Beta Reduction Constraints

Manuel Bodirsky Katrin Erk*
Alexander Koller♯ ** Joachim Niehren**

Programming Systems Lab ♯ Dept. of Computational Linguistics
Universität des Saarlandes, Saarbrücken, Germany
www.ps.uni-sb.de/~{bodirsky,erk,koller,niehren}

Abstract. The constraint language for lambda structures (CLLS) can model lambda terms that are known only partially. In this paper, we introduce beta reduction constraints to describe beta reduction steps between partially known lambda terms. We show that beta reduction constraints can be expressed in an extension of CLLS by group parallelism. We then extend a known semi-decision procedure for CLLS to also deal with group parallelism and thus with beta-reduction constraints.

1 Introduction

The constraint language for lambda structures (CLLS) [7, 6, 8] can model λ-terms that are known only partially. The idea is to see a λ-term as a λ-structure: a tree decorated with binding edges. One can then describe a λ-term partially as one would describe a standard tree structure. CLLS provides dominance [13, 2, 5], parallelism [9] and binding constraints for this purpose.

This paper shows how to lift β-reduction to partial descriptions of λ-terms in CLLS. We define *beta reduction constraints*, which allow a declarative description of the result of a single β-reduction step. At first, this description is very implicit; it is made explicit by *solving* the constraints. To this end, we show how beta reduction constraints can be expressed as *group parallelism constraints*. Then we adapt a known semi-decision procedure for CLLS to also deal with group parallelism and thus with beta-reduction constraints.

Beta-reduction constraints lay the foundation for *underspecified beta reduction*, which is needed in the application of CLLS to *semantic underspecification* of natural language [15, 17, 14]. Given a CLLS constraint describing many lambda terms, the aim is to compute a compact description of all corresponding beta normal forms efficiently. In particular, we want to avoid enumerating and individually beta-reducing the described lambda terms. (Enumerating is neither efficient, nor is its result compact.) A recent proposal towards *underspecified beta reduction* is described by the authors in a follow-up paper [4].

* Supported by the DFG through the Graduiertenkolleg Kognition in Saarbrücken.
** Supported by the Collaborative Research Center (SFB) 378 of the DFG and the Procope project of the DAAD.

Solving beta reduction constraints is very much different from higher-order unification [10] in that CLLS constraints express α-equality rather than $\alpha\beta\eta$-equality. CLLS is closely linked to context unification [12, 16], and it can express sharing as in optimal lambda reduction [11] or calculi with explict substitutions [1] but can also describe several lambda terms at once.

Plan. We first recall the definition of CLLS restricted to dominance and λ-binding constraints (Sec. 2); then we go through two examples to give an idea of how one might lift β-reduction to partial descriptions (Sec. 3). We next define β-reduction constraints (Sec. 4). Then we define group parallelism constraints and show how they can express β-reduction constraints (Sec. 5). Finally, we present a sound and complete semi-decision procedure for CLLS with group parallelism (Sec. 6) and illustrate it with an example (Sec. 7).

2 CLLS with dominance and lambda binding constraints

We first introduce λ-structures and then a fragment of CLLS for their description. This fragment contains dominance and λ-binding constraints, but not parallelism and anaphoric binding constraints.

We assume a signature $\Sigma = \{f, g, \dots\}$ of function symbols, each equipped with an arity $\mathsf{ar}(f) \geq 0$. Symbols of arity 0 are constants, written as $a, b, \dots$

A tree θ is a ground term over Σ, e.g. $g(f(a, b))$. A *node* of a tree is identified with a *path* from the root to this node, expressed by a word over the naturals (excluding 0). ϵ is the empty path, and $\pi_1\pi_2$ the concatenation of π_1 and π_2. π is a *prefix* of a path π' if there is a (possibly empty) π'' s.t. $\pi\pi'' = \pi'$. The set of all nodes of a tree θ is defined as

$$D_\theta(f(\theta_1, \dots, \theta_n)) \;=\; \epsilon \cup \{i\pi \mid \pi \in D_\theta(\theta_i),\ 1 \leq i \leq n\}$$

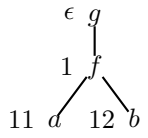


Fig. 1. Tree structure for $g(f(a, b))$.

A tree θ can be characterized uniquely by the set D_θ of its nodes and a labeling function $L_\theta : D_\theta \to \Sigma$.

Now we can consider λ-terms as pairs of a tree and a *binding function* that encodes variable binding. We assume that Σ contains the symbols var (arity 0, for variables), lam (arity 1, for abstraction), and @ (arity 2, for application), and quantifiers $\exists$ and $\forall$ (arity 1). The tree uses these symbols to reflect the structure of the λ-term and of first-order connectives. The binding function λ explicitly maps var-labeled nodes to binders. For example, Fig. 2 shows a representation of the term $\lambda x.f(x)$. Here $\lambda(12) = \epsilon$. Such a pair of a tree and a binding function is called a λ-structure.

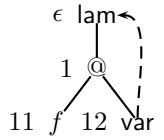


Fig. 2. The λ-structure of $\lambda x.f(x)$

Definition 1. *A* λ-structure τ *is a pair* (θ, λ) *of a tree* θ *and a total binding function* $\lambda : L_\theta^{-1}(\mathsf{var}) \to L_\theta^{-1}(\{\mathsf{lam}, \exists, \forall\})$ *such that* $\lambda(\pi)$ *is always a prefix of* π.

A λ-structure corresponds uniquely to a closed λ-term modulo α-renaming. We freely consider λ-structures as first-order structures with domain D_θ. As such, they define relations of labeling, binding, inverse binding, dominance, disjointness, and inequality of nodes. (Later we will add group parallelism and β-reduction relations.) The labeling relation $\pi{:}f(\pi_1,\ldots,\pi_n)$ holds in a λ-structure $\tau = (\theta,\lambda)$ if $L_\theta(\pi) = f$, $\mathsf{ar}(f) = n$ and $\pi_i = \pi i$ for all $1 \leq i \leq n$. *Dominance* $\lhd^*$ is the prefix relation between paths of D_θ; inequality $\neq$ is simply inequality of paths; *disjointness* $\pi \perp \pi'$ holds if neither $\pi \lhd^* \pi'$ nor $\pi' \lhd^* \pi$. We will also consider intersections, unions, and complements of these relations; for instance, *proper dominance* $\lhd^+$ is $\lhd^* \cap \neq$, and *equality* $=$ is $\lhd^* \cap \rhd^*$. The relation $\lambda^{-1}(\pi_0){=}\{\pi_1,\ldots,\pi_n\}$ states that $\pi_1,\ldots,\pi_n$, and only those nodes, are λ-bound by π_0. Note that an element of a set can be mentioned multiply, i.e. $\{\pi,\pi\} = \{\pi\}$.

Now we can define *dominance and binding* constraints to talk about λ-structures as follows; X, Y, Z are variables that denote nodes.

$$\begin{array}{rcll} \varphi,\psi & ::= & XRY \mid X{:}f(X_1,\ldots,X_n) \mid \varphi \wedge \psi \mid \mathbf{false} & (\mathsf{ar}(f) = n) \\ & \mid & \lambda(X){=}Y \mid \lambda^{-1}(X_0){=}\{X_1,\ldots,X_n\} & \\ R,R' & ::= & \lhd^* \mid \rhd^* \mid \perp \mid \neq \mid R{\cup}R' \mid R{\cap}R' & \end{array}$$

A constraint φ is a conjunction of *literals* (for dominance, labeling, etc). Set operators in relation descriptors R [5] are mainly needed for processing purposes. As above we also use $\lhd^+, =$ to abbreviate set operators. The one literal that has not appeared in the literature before is the *inverse binding literal* $\lambda^{-1}(X){=}\{X_1,\ldots,X_n\}$, which matches the inverse binding relation.

We will also use first-order formulas Φ built over constraints. We write $\mathcal{V}(\Phi)$ for the set of variables occurring in Φ. Given a pair (τ,σ) of a λ-structure τ and a variable assignment $\sigma : \mathcal{G} \to D_\tau$, for some set $\mathcal{G} \supseteq \mathcal{V}(\varphi)$, we can associate a truth value to Φ in the usual Tarskian sense. We say that (τ,σ) *satisfies* Φ iff Φ evaluates to true under (τ,σ). In this case, we write $(\tau,\sigma) \models \Phi$ and say that (τ,σ) is a *solution* of Φ. Φ is *satisfiable* iff it has a solution. Entailment $\Phi \models \Phi'$ means that all solutions of Φ are also solutions of Φ', equivalence $\Phi \models\!\!\models \Phi'$ is mutual entailment.

We draw constraints as graphs with the nodes representing variables. E.g. Fig. 3 is the graph of $\lambda^{-1}(X){=}\{X_1,X_2\} \wedge X \lhd^* X_1 \wedge X \lhd^* X_2$. Labels and solid lines indicate labeling literals, while dotted lines represent dominance. Dashed arrows indicate the binding relation; disjointness and inequality literals are not represented.

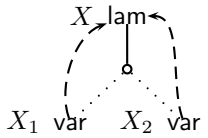


Fig. 3. A constraint graph

3 Examples

Before we begin with the formal investigation of beta reduction constraints, we first go through two examples which illustrate how beta-reduction can be lifted to descriptions of λ-structures in CLLS, and why the problem is nontrivial.

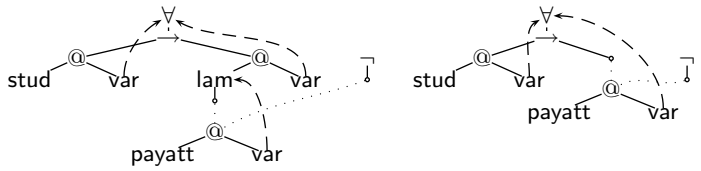


Fig. 4. Underspecified representations of 'Every student did not pay attention' before and after beta reduction.

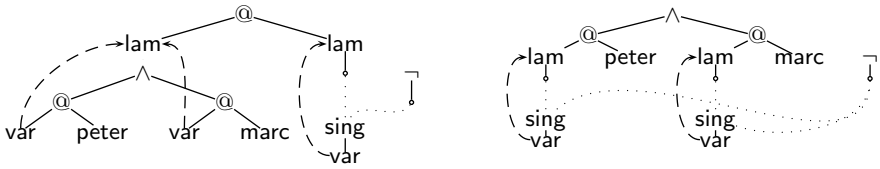


Fig. 5. Representation of 'Peter and Marc do not sing', wrong description of the reduct.

First, consider the left constraint in Fig. 4. The constraint contains just one redex, and it is easy to see how to obtain a description of the reduced formulas. We can essentially replace the bound variables with the argument description; the result is shown on the right-hand side of Fig. 4.

Incidentally, the left constraint in Fig. 4 is an underspecified description of the ambiguous sentence *Every student didn't pay attention*. Its two readings are given by the HOL formulas:

$$\forall x\ (\text{stud}\ x \to (\lambda y \neg(\text{payatt}\ y))\ x), \qquad \neg\forall x\ (\text{stud}\ x \to (\lambda y\ \text{payatt}\ y)\ x).$$

(These are the only models of the constraint that do not contain additional material not mentioned in the constraint. We ignore this aspect of "solution minimality" in this paper and always consider all solutions.)

The naive replacement approach, however, does not work in general. Fig. 5 shows an example (which describes the ambiguous sentence 'Peter and Marc do not sing.') This constraint also describes a β-redex, this time one where the binder binds two variables. Here it is no longer trivial to replace the bound variables by the argument description, as we do not know what belongs to the argument. There is no useful choice for the part of the constraint that should be duplicated; for example, if we decide not to duplicate the negation, we get the description on the right-hand side of Fig. 5, which lacks one solution. Describing the reduct using β-reduction constraints solves this problem; the description is correct even if it is not yet known which variables belong to the body and the argument of the redex.

4 Beta Reduction Constraints

In this section, we add the *β-reduction relation* to lambda structures and extend the constraint language by *β-reduction constraints*. The β-reduction relation on nodes of a lambda structure corresponds exactly to traditional beta reduction on lambda terms.

Stated in the unfolded notation for λ-terms we use to build the λ-structures (with application as an internal label @, etc.), β-reduction looks as follows:

$$C(@(\lambda x.B, A)) \quad \rightarrow_\beta \quad C(B[x/A]) \qquad\qquad x \text{ free for } A$$

We call the left-hand side the *reducing tree*, the right-hand side *the reduct* of the β-reduction. We call C the context, B the body, and A the argument of the reduction step.

Now an important notion throughout the paper are *tree segments*. Intuitively, a tree segment is a subtree which may itself be missing some subtrees (see Fig. 6). The context, body, and argument of a beta reduction step will all be tree segments.

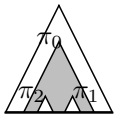


Fig. 6. The tree segment $\pi_0/\pi_1, \pi_2$.

Definition 2. *A* tree segment α *of a λ-structure τ is given by a tuple $\pi_0/\pi_1, \ldots, \pi_n$ of nodes in D_τ, such that $\tau \models \pi_0 \triangleleft^* \pi_i$ and $\tau \models \pi_i(\bot \cup =)\pi_j$ for $1 \leq i < j \leq n$. The node $r(\alpha) = \pi_0$ is called the* root*, and $hs(\alpha) = \pi_1, \ldots, \pi_n$ is the sequence of* holes *of α. If $n = 0$ we write $\alpha = \pi_0/$. The nodes* between *the root $r(\alpha)$ and the holes $hs(\alpha)$ are defined as*

$$\mathsf{b}(\alpha) =_{\mathsf{df}} \{\pi \in D_\tau \mid r(\alpha) \triangleleft^* \pi \wedge \bigwedge_{\pi' \in \{hs(\alpha)\}} \pi' \neg \triangleleft^+ \pi\}$$

To exempt the holes of the segment, we define $\mathsf{b}^-(\alpha) =_{\mathsf{df}} \mathsf{b}(\alpha) - \{hs(\alpha)\}$.

Definition 3. *A* correspondence function *between tree segments α, β in a lambda structure τ is a bijective mapping $c : \mathsf{b}(\alpha) \rightarrow \mathsf{b}(\beta)$ which satisfies for all nodes $\pi_1, \ldots, \pi_n$ of τ:*

1. *The roots correspond: $c(r(\alpha)) = r(\beta)$*
2. *The sequences of holes correspond:*

$$hs(\alpha) = \pi_1, \ldots, \pi_n \Leftrightarrow hs(\beta) = c(\pi_1), \ldots, c(\pi_n)$$

3. *Labels and children correspond within the proper segments. For $\pi \in \mathsf{b}^-(\alpha)$ and label f:*

$$\pi{:}f(\pi_1, \ldots, \pi_n) \Leftrightarrow c(\pi){:}f(c(\pi_1), \ldots c(\pi_n)).$$

We next define the β-reduction relation on λ-structures to be a relation between nodes in the *same* λ-structure. This allows us to see the β-reduction relation as a conservative extension of the existing λ-structures. The representations both of the reducing and reduced term are part of same big λ-structure—in Fig. 7, these are the subtrees rooted by $r(\gamma)$ and $r(\gamma')$ respectively.

A *redex* in a lambda structure is a sequence of segments (γ, β, α) that are connected by nodes π_0, π_1 with the following properties.

$$hs(\gamma) = \pi_0,\ \pi_0{:}@(\pi_1, r(\alpha)),\ \pi_1{:}\mathsf{lam}(r(\beta)), \text{ and } \lambda^{-1}(\pi_1) = \{hs(\beta)\}$$

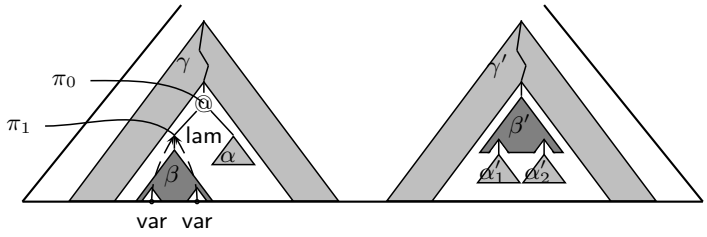


Fig. 7. The beta reduction relation for a binary redex.

We call a sequence of segments $(\gamma', \beta', \alpha'_1, \ldots, \alpha'_n)$ reductlike iff $hs(\gamma') = r(\beta')$, and $r(\alpha'_i)$ is the ith hole of β' for all $1 \leq i \leq n$.

Note that not every reductlike segment sequence is a potential reduct of a beta reduction, since we cannot enforce that there is no binder from the argument into the body (that would violate the freeness condition).

The lambda structure in Fig. 7 contains a redex (γ, β, α) and also its reduct $(\gamma', \beta', \alpha'_1, \alpha'_2)$. There, corresponding segments (γ to γ', β to β', α to both α'_1 and α'_2) have the same structure.

Definition 4 (Beta Reduction Relation). *Let τ be a λ-structure. Then*

$$(\gamma, \beta, \alpha) \rightarrow^{\beta} (\gamma', \beta', \alpha'_1, \ldots, \alpha'_n)$$

holds in τ iff first, (γ, β, α) form a redex and $(\gamma', \beta', \alpha'_1, \ldots, \alpha'_n)$ are reductlike. *Second, there are correspondence functions c_γ between γ, γ', c_β between β, β' and c^i_α between α, α'_i (for $1 \leq i \leq n$), such that for each δ, δ' among these segment pairs with correspondence function c between them and for each $\pi \in \mathsf{b}^-(\delta)$, the following conditions hold:*

1. *for a* var*-labeled node bound in the same segment, the correspondent is bound by the c-corresponding binder node.*

$$\lambda(\pi) \in \mathsf{b}^-(\delta) \Rightarrow \lambda(c(\pi)) = c(\lambda(\pi))$$

2. *for a* var*-labeled node bound in the context γ, the correspondent is bound by the c_γ-corresponding binder node.*

$$\lambda(\pi) \in \mathsf{b}^-(\gamma) \Rightarrow \lambda(c(\pi)) = c_\gamma(\lambda(\pi))$$

3. *for a* var*-labeled node bound above the reducing tree, the corresponding node is bound at the same place:*

$$\lambda(\pi) \notin \mathsf{b}(r(\gamma)/) \Rightarrow \lambda(c(\pi)) = \lambda(\pi)$$

The β-reduction relation on λ-structures models β-reduction on λ-terms faithfully. This even holds for λ-terms with global variables, although λ-structures can only model closed λ-terms. Global variables correspond to var-labeled nodes that are bound in the surrounding tree, i.e. above the root node

of the context of the redex. Rule 3 of Def. 4 thus ensures a proper treatment of global variables.

Capturing in β-reduction on λ-terms in classical λ-calculus is usually avoided by a freeness condition. For instance, one cannot simply β-reduce $(\lambda x.\lambda y.x)y$ without renaming the bound ocurrence of y beforehand. Otherwise, the global variable y in the argument gets captured by the binder λy. The following proposition states that the analoguous problem can never arise with the β-reduction relation on λ-structures.

Proposition 5 (No Capturing). *Global variables in the argument are never captured by a λ-binder in the body: with the notation of Def. 4, this means that no* var*-labeled node in* $\mathsf{b}(\alpha'_i)$ *is bound by a* lam*-labeled node in* $\mathsf{b}^-(\beta')$.

Proof. Assume there exists a node π' in $\mathsf{b}(\alpha'_i)$ such that $\lambda(\pi') \in \mathsf{b}^-(\beta')$. There must be a corresponding var-labeled node π with $c^i_\alpha(\pi) = \pi'$, which is bound either in α, in γ or outside the reducing tree. In the first case property (1) leads to a contradiction, in the second case property (2), and in the third case (3).

The β-reduction relation conservatively extends λ-structures. We extend our constraint syntax similarly, by *β-reduction literals*, which are interpreted by the β-reduction relation. Let a *segment term* A, B, C be given by the following abstract syntax:

$$A, B, C =_{\mathsf{df}} X_0/X_1, \ldots, X_n$$

Then β-reduction literals have the following form:

$$(C, B, A) \to^\beta (C', B', A'_1, \ldots, A'_n)$$

5 Group Parallelism

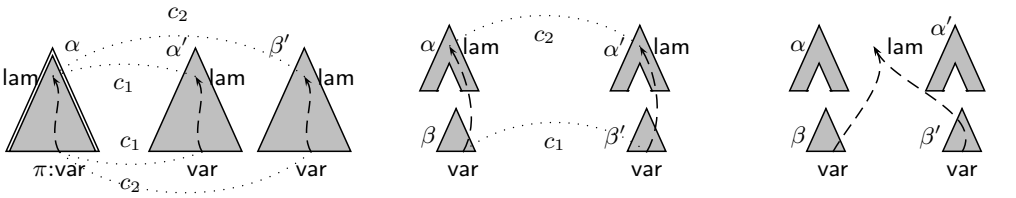
In this section, we extend dominance and binding constraints with *group parallelism constraints* (Def. 6), a generalization of the parallelism constraints found in CLLS [6, 9]. Then we show that CLLS with group parallelism can express β-reduction constraints (Thm. 7).

Group parallelism relates two *groups*, i.e. sequences of tree segments. It requires that corresponding entries in the two sequences must have the same tree structures, and binding in the two groups must be parallel. The following definition makes this precise; all conditions but the last are illustrated in Fig. 8.

Definition 6. *The* group parallelism relation $\sim$ *of a λ-structure τ is the greatest symmetric relation between groups of the same size such that*

$$(\alpha_1, \ldots, \alpha_n) \sim (\alpha'_1, \ldots, \alpha'_n)$$

implies there are correspondence functions $c_k : \mathsf{b}(\alpha_k) \to \mathsf{b}(\alpha'_k)$ for all $1 \le k \le n$ that satisfy the following properties for all $1 \le i, j \le n$ and $\pi \in \mathsf{b}^-(\alpha_i)$:



(same.seg) $(\alpha, \alpha) \sim (\alpha', \beta')$ (diff.seg) $(\alpha, \beta) \sim (\alpha', \beta')$ (outside) $(\alpha, \beta) \sim (\alpha', \beta')$

Fig. 8. Possible bindings in a group parallelism.

(same.seg) *for a* var*-labeled node bound in the same segment, the corresponding node is bound correspondingly:*

$$\lambda(\pi) \in \mathsf{b}^-(\alpha_i) \;\Rightarrow\; \lambda(c_i(\pi)) = c_i(\lambda(\pi))$$

(diff.seg) *for a* var*-labeled node bound outside α_i but inside α_j, the correspondent is bound at the corresponding place with respect to c_j:*

$$\lambda(\pi) \in \mathsf{b}^-(\alpha_j) \wedge \lambda(\pi) \notin \mathsf{b}^-(\alpha_i) \quad \Rightarrow\; \lambda(c_i(\pi)) = c_j(\lambda(\pi))$$

(outside) *corresponding* var*-labeled nodes with binders outside the group segments are bound by the same binder node:*

$$\lambda(\pi) \notin \cup_{k=1}^{n} \mathsf{b}^-(\alpha_k) \;\Rightarrow\; \lambda(c_i(\pi)) = \lambda(\pi)$$

(hang) *there are no hanging binders:*

$$\lambda^{-1}(\pi) \subseteq \cup_{k=1}^{n} \mathsf{b}^-(\alpha_k)$$

On the syntactic side, we extend CLLS by *group parallelism literals* that are interpreted by the group parallelism relation. Let $A_1, \ldots A_m, A'_1, \ldots, A'_m$ be segment terms, then group parallelism literals have the form

$$(A_1, \ldots, A_m) \quad \sim \quad (A'_1, \ldots, A'_m)$$

Group parallelism extends ordinary parallelism constraints [6, 9], which are simply the special case for groups of size one. This extension is proper; ordinary parallelism constraints cannot handle the case where a node is bound in a different segment of the group, as illustrated in the (diff.seg) part of Fig. 8. From the perspective of ordinary parallelism, the node is bound outside the parallel segment, and thus the (outside) condition applies, and the corresponding node must be bound by the *same* binder.

Another interesting observation in Fig. 8 is that the conditions (same.seg) and (diff.seg) must be mutually exclusive. If (diff.seg) was applicable in the leftmost case, it would enforce $\lambda(c_1(\pi)) = c_2(\lambda(\pi))$, which is clearly wrong.

Now we show how to encode beta reduction constraints in CLLS. First, we define the following formula to express that the segment term $A = X_0/X_1, \ldots, X_n$ indeed denotes a tree segment:

$$\mathsf{seg}(A) =_{\mathsf{df}} \wedge_{i=1}^{n} X_0 \triangleleft^* X_i \ \wedge \ \wedge_{1 \le i < j \le n} X_i (\perp \cup =) X_j$$

Using this, we can axiomatize a redex in CLLS. For segment terms $C = X'/X_0$, $B = X_3/X_4, \ldots, X_n$ and $A = X_2/$ we set:

$$\begin{aligned} \mathsf{redex}_{X_0,X_1}(C, B, A) \ =_{\mathsf{df}} \ & \mathsf{seg}(A) \ \wedge \ \mathsf{seg}(B) \ \wedge \ \mathsf{seg}(C) \\ & \wedge \ X_0{:}@(X_1, X_2) \wedge X_1{:}\mathsf{lam}(X_3) \\ & \wedge \ \lambda^{-1}(X_1) = \{X_4, \ldots, X_n\} \end{aligned}$$

Next, we define reduct-like groups. Let $C = X/X_0$, $B = X_0'/X_1', \ldots, X_n'$ and $A_i = X_i/$ for $1 \le i \le n$, then we define:

$$\begin{aligned} \mathsf{reductlike}(C, B, A_1, \ldots, A_n) \ =_{\mathsf{df}} \ & \mathsf{seg}(A_1) \wedge \cdots \wedge \mathsf{seg}(A_n) \wedge \mathsf{seg}(B) \wedge \mathsf{seg}(C) \\ & \wedge X_0{=}X_0' \wedge X_1{=}X_1' \wedge \cdots \wedge X_n{=}X_n' \end{aligned}$$

Theorem 7. *Beta reduction constraints can be expressed in CLLS with group parallelism via the following equivalence:*

$$\begin{aligned} (C, B, A) \to^{\beta} (C', B', A_1', \ldots, A_n') \quad \models \quad & \exists \, X_0, X_1 : \mathsf{redex}_{X_0,X_1}(C, B, A) \\ & \wedge \ (C, B, A, \ldots, A) \sim (C', B', A', \ldots, A') \\ & \wedge \ \mathsf{reductlike}(C', B', A_1', \ldots, A_n') \end{aligned}$$

Proof. We will check the two-side entailment separately, first from right to left. Let σ be a variable assignment into some λ-structure that solves the right hand side. Properties (same.seg), (diff.seg), and (outside) of group parallelism (Def. 6) then subsume the corresponding properties of β-reduction (Def. 4).

For the other direction, let (τ, σ') solve the beta-reduction literal on the left hand side. According to (Def. 4) there exists a redex (γ, β, α) in τ with nodes π_0, π_1 as in Sec. 4. Let σ be the variable assignment $\sigma'[\pi_0/X_0, \pi_1/X_1, \alpha/A, \beta/B, \gamma/C]$. It remains to check that (τ, σ) solves the group parallelism literal on the right hand side.

We consider the symmetric relation $\approx$ which relates the group $(\sigma(C), \sigma(B), \sigma(A), \ldots, \sigma(A))$ to $(\sigma(C'), \sigma(B'), \sigma(A_1'), \ldots, \sigma(A_n'))$ and conversely. We show that $\approx$ satisfies all conditions in the definition of group parallelism (Def. 6), which means that $\approx$ is subsumed by the group parallelism relation $\sim$.

First of all, both above groups satisfy condition (hang). This is clear for the group $(\sigma(C'), \sigma(B'), \sigma(A_1'), \ldots, \sigma(A_n'))$, which covers the complete subtree below $r(\sigma(C'))$. A similar argument applies to $(\sigma(C), \sigma(B), \sigma(A), \ldots, \sigma(A))$, which covers the whole tree below $r(\sigma(C))$ exept the @-labeled node π_0, the lam-labeled node π_1 and the var-labeled nodes $hs(\sigma(B))$. But these var-labeled nodes are bound by π_1.

By Def. 4 there exist correspondence functions c_γ between segments $\sigma(C)$, $\sigma(C')$, c_β between $\sigma(B)$, $\sigma(B')$ and c_α^i between $\sigma(A), \sigma(A_i')$ for $1 \le i \le n$. Since $\approx$ is symmetric, we have to check properties (same.seg), (diff.seg), and (outside) of group parallelism (Def. 6) for the correspondence functions and their inverse functions.

(D.clash.ineq) $X{=}Y \wedge X{\neq}Y \rightarrow \mathbf{false}$

(D.dom.trans) $X{\lhd^*}Y \wedge Y{\lhd^*}Z \rightarrow X{\lhd^*}Z$

(D.lab.ineq) $X{:}f(\ldots) \wedge Y{:}g(\ldots) \rightarrow X{\neq}Y$ where $f \neq g$

(D.lab.dom) $X{:}f(\ldots, Y, \ldots) \rightarrow X{\lhd^+}Y$

(D.distr.notDisj) $X{\lhd^*}Z \wedge Y{\lhd^*}Z \rightarrow X{\lhd^*}Y \vee Y{\lhd^*}X$

(D.distr.child) $X{\lhd^*}Y \wedge X{:}f(X_1, \ldots, X_m) \rightarrow Y{=}X \vee \bigvee_{i=1}^{m} X_i{\lhd^*}Y$

Fig. 9. Saturation rules for dominance constraints

We only show the particularly interesting property (diff.seg) for a correspondence function $(c_\alpha^i)^{-1}$ with $1 \leq i \leq n$. Let π' be a var-labeled node in $\mathsf{b}^-(\sigma(A_i'))$, and $\lambda(\pi') \notin \mathsf{b}^-(\sigma(A_i'))$. There are three cases: $\lambda(\pi') \in \mathsf{b}^-(\sigma(C'))$, or $\lambda(\pi') \in \mathsf{b}^-(\sigma(B'))$, or $\lambda(\pi') \in \mathsf{b}^-(\sigma(A_j'))$ for some $1 \leq j \leq n$. The second case is impossible by Proposition 5. The third case is impossible as the holes of the segment $\sigma(B')$ are disjoint or equal (Def. 2). We can thus concentrate on the first case. Let π be the corresponding node of π'. i.e. $c_\alpha^i(\pi) = \pi'$. The node π has to be var-labeled by Def 3. Properties (same.seg) and (outside) of Def. 4 yield $\lambda(\pi) \in \sigma(C)$ (some computation is needed here). Thus, Property 2 of Def. 4 can be applied. It implies $\lambda(\pi') = c_\gamma(\lambda(\pi))$, i.e. $c_\gamma^{-1}(\lambda(\pi')) = \lambda(c_\gamma^{-1}(\pi'))$ as required.

6 Solving group parallelism constraints

We now turn to a sound and complete semi-decision procedure for CLLS with group parallelism, which thus solves β-reduction constraints. To keep the presentation readable, we focus on the most relevant rules. The full procedure is given in [3]. An illustrative example follows in Section 7.

The procedure is obtained by extending an existing semi-decision procedure for CLLS [8] that is based on *saturation*. A constraint is freely identified with the *set* of its literals. Starting with a set of literals, more literals are added according to some *saturation rules*. Our saturation rules are implications of the form $\varphi_0 \rightarrow \vee_{i=1}^n \varphi_i$ for some $n \geq 1$. To write down rules more compactly, we will also use arbitrary positive existential formulas on the left hand side. These can be eliminated in a preprocessing step: $\exists$-quantified variables can be replaced by arbitrary variables, and disjunction is eliminated by explosion into several rules.

A saturation rule of the above form is applicable to a constraint φ if φ_0 is contained in φ, but none of the φ_i is. A rule $\varphi \rightarrow \Phi$ is *sound* if $\varphi \models \Phi$. Apart from that, we have saturation rules of the form $\varphi_0 \rightarrow \exists X \varphi_1$, which introduce fresh variables. Such a rule is applicable to φ if φ_0 is in φ, but φ_1, modulo renaming of X, is not. Given a set S of saturation rules, we call a constraint *saturated* (under S) if no further rule of S applies to it. We say that a constraint is in *S-solved form* if it is saturated under S and clash-free (i.e. it does not contain **false**).

Fig. 9 contains an (incomplete) set of saturation rules for dealing with dominance constraints (the constraints of Sec. 2 without binding). A more

(P.symm) $\overline{A} \sim \overline{B} \to \overline{B} \sim \overline{A}$

(P.init) $\overline{A} \sim \overline{B} \to \mathsf{seg}(A_i) \wedge \mathrm{co}(A_i, B_i)(X_i^j){=}Y_i^j$ where $1 \leq i \leq n$, $A_i = X_i^0/X_i^1, \ldots, X_i^{m_i}$, $B_i = X_i^0/Y_i^1, \ldots, Y_i^{m_i}$, and $0 \leq j \leq m_i$

(P.new) $\overline{A} \sim \overline{B} \wedge U \in \mathsf{b}(A_i) \to \exists V\ \mathrm{co}(A_i, B_i)(U){=}V$ where V fresh, $1 \leq i \leq n$

(P.copy.lab) $\bigwedge_{i=0}^{m} \mathrm{co}(A, B)(X_i){=}Y_i \wedge X_0{:}f(X_1, \ldots, X_m) \wedge X_0{\in}\mathsf{b}^-(A) \to Y_0{:}f(Y_1, \ldots, Y_m)$

(P.copy.dom) $U_1\ R\ U_2 \wedge \bigwedge_{i=1}^{2} \mathrm{co}(A, B)(U_i){=}V_i \to V_1\ R\ V_2$

(P.distr.eq) $\varphi \to X{=}Y \vee X{\neq}Y$ for $X, Y \in \mathcal{V}(\varphi)$

Fig. 10. Saturation rules, where $\overline{A} = A_1, \ldots, A_n$ and $\overline{B} = B_1, \ldots, B_n$

complete collection including a treatment of set operators can be found in [5, 8, 3]. To deal with parallelism, we first introduce some formulas that describe membership in (proper) segments and groups. Let $A = X_0/X_1, \ldots, X_n$.

$$\begin{aligned}
X{\in}\mathsf{b}(A) &=_{\mathsf{df}} X_0{\lhd^*}X \wedge \textstyle\bigwedge_{i=1}^{n} X(\lhd^*{\cup}\bot)X_i \\
X{\in}\mathsf{b}^-(A) &=_{\mathsf{df}} X{\in}\mathsf{b}(A) \wedge \textstyle\bigwedge_{i=1}^{n} X{\neq}X_i \\
X{\notin}\mathsf{b}^-(A) &=_{\mathsf{df}} X(\lhd^+{\cup}\bot)X_0 \vee \textstyle\bigvee_{i=1}^{n} X_i{\lhd^*}X \\
X{\in}\mathsf{b}(A_1, \ldots, A_m) &=_{\mathsf{df}} \textstyle\bigvee_{i=1}^{m} X{\in}\mathsf{b}(A_i) \\
X{\in}\mathsf{b}^-(A_1, \ldots, A_m) &=_{\mathsf{df}} \textstyle\bigvee_{i=1}^{m} X{\in}\mathsf{b}^-(A_i)
\end{aligned}$$

Note that the terms $\mathsf{b}(A)$, $\mathsf{b}^-(A)$, $\mathsf{b}(A_1, \ldots, A_m)$ are not given any formal meaning, even though it would be correct to interpret them as the corresponding sets of nodes.

We also want to be able to speak about correspondence functions. So we extend our constraint language by auxiliary correspondence literals

$$\varphi \quad ::= \quad \ldots \mid \mathrm{co}(A, B)(X){=}Y$$

where A and B are segment terms for segments with the same number of holes. Such a literal states that A and B are parallel within some group parallelism, that $X{\in}\mathsf{b}(A)$ and $Y{\in}\mathsf{b}(B)$, and that X corresponds to Y with respect to the correspondence function for A and B. We introduce two more formulas. Let $\overline{A} = (A_1, \ldots, A_n)$, $\overline{B} = (B_1, \ldots, B_n)$, and $1 \leq k \leq n$.

$$\begin{aligned}
\mathrm{co}^-(A, B)(X){=}Y &=_{\mathsf{df}} \mathrm{co}(A, B)(X){=}Y \wedge X{\in}\mathsf{b}^-(A) \\
\mathrm{co}_k^-(\overline{A}, \overline{B})(X){=}Y &=_{\mathsf{df}} \overline{A}{\sim}\overline{B} \wedge \mathrm{co}^-(A_k, B_k)(X){=}Y
\end{aligned}$$

The second lets us talk about correspondence functions for a group parallelism, picking out the k-th correspondence function. In that respect, $\mathrm{co}_k^-(\overline{A}, \overline{B})$ matches the c_k of Def. 6 (except that $\mathrm{co}_k^-(\overline{A}, \overline{B})(X){=}Y$ additionally demands $X{\in}\mathsf{b}^-(A_k)$ for convenience).

The main rules for handling parallelism are given in Fig. 10. A complete set can be found in [9, 8, 3]. The rules (P.init) and (P.new) introduce correspondence literals; between them, they state that each node in a parallel segment

(L.same.seg) $\lambda(U_1){=}U_2 \wedge \bigwedge_{i=1}^{2} \mathrm{co}_k^-(\overline{A},\overline{B})(U_i){=}V_i \rightarrow \lambda(V_1){=}V_2$

(L.diff.seg) $\lambda(U_1){=}U_2 \wedge \bigwedge_{i=1}^{2} \mathrm{co}_{k_i}^-(\overline{A},\overline{B})(U_i){=}V_i \wedge U_2{\notin}\mathsf{b}^-(A_{k_1}) \rightarrow \lambda(V_1){=}V_2$

(L.outside) $\lambda(U){=}Y \wedge \mathrm{co}_k^-(\overline{A},\overline{B})(U){=}V \wedge Y{\notin}\mathsf{b}^-(\overline{A}) \rightarrow \lambda(V){=}Y$

(L.hang) $\lambda(U_1){=}U_2 \wedge \overline{A}{\sim}\overline{B} \wedge U_2{\in}\mathsf{b}^-(\overline{A}) \rightarrow U_1{\in}\mathsf{b}^-(\overline{A})$

(L.distr.1) $\lambda(U_1){=}U_2 \wedge \overline{A}{\sim}\overline{B} \wedge U_1{\in}\mathsf{b}^-(\overline{A}) \rightarrow \mathsf{distr}_{\overline{A}}(U_2)$

(L.distr.2) $\lambda(U_1){=}U_2 \wedge \overline{A}{\sim}\overline{B} \wedge U_2 \in \mathsf{b}^-(\overline{A}) \rightarrow \mathsf{distr}_{\overline{A}}(U_1)$

(L.equal) $\lambda(X_1){=}X_2 \wedge \bigwedge_{i=1}^{2} X_i{=}Y_i \rightarrow \lambda(Y_1){=}Y_2$

(L.inverse) $\lambda^{-1}(X){=}S_1 \wedge \mathrm{co}_k^-(\overline{A},\overline{B})(X){=}Y \wedge \mathrm{co}^-(\overline{A},\overline{B})(S_1){=}S_2 \cup S_3 \wedge \bigwedge_{V\in S_2} \lambda(V){=}Y \wedge \bigwedge_{V\in S_3} \lambda(V){\neq}Y \rightarrow \lambda^{-1}(Y){=}S_2$

Fig. 11. Lambda binding rules for group parallelism

needs to have a correspondent. (P.init) states that in a correspondence function, root corresponds to root, and hole to hole, while (P.new) is responsible for all other nodes. (P.copy.dom) and (P.copy.lab) between them ascertain the structural isomorphism that Def. 3 demands for a correspondence function.

Fig. 11 shows saturation rules for the interaction of group parallelism and lambda binding. The first four rule schemata directly express the conditions of Def. 6. The rules (L.distr.gr.1) and (L.distr.gr.2) decide, loosely speaking, whether variables occurring in a binding literal belong to some segment of a group or not. This is necessary because we need to know which of the schemata (L.same.seg), (L.diff.seg), (L.outside) and (L.hang) is applicable. This is expressed by using the following formula, where $\overline{A} = (A_1, \ldots, A_n)$:

$$\mathsf{distr}_{\overline{A}}(U) =_{\mathsf{df}} \bigwedge_{i=1}^{n} (U{\in}\mathsf{b}^-(A_i) \vee U{\notin}\mathsf{b}^-(A_i).$$

Finally, (L.inverse) deals with the copying of λ^{-1} literals. This is necessary if we want to perform a second beta reduction step, where we need the λ^{-1} information again. The schema uses two more formulas. The first one is simple:

$$\lambda(X){\neq}Y =_{\mathsf{df}} \exists Z(\lambda(X){=}Z \wedge Z{\neq}Y)$$

The second formula collects, for a finite set S_1 of variables, all correspondents with respect to $\overline{A} \sim \overline{B}$. Let S_1, S_2 stand for finite sets of variables, and let $\overline{A} = A_1 \ldots, A_n$.

$$\begin{aligned}\mathrm{co}^-(\overline{A},\overline{B})(S_1){=}S_2 =_{\mathsf{df}} & \textstyle\bigwedge_{i=1}^{n} \bigwedge_{X\in S_1} (X{\notin}\mathsf{b}^-(A_i) \vee \bigvee_{Y\in S_2} \mathrm{co}_i^-(\overline{A},\overline{B})(X){=}Y) \\ \wedge & \textstyle\bigwedge_{Y\in S_2} \bigvee_{X\in S_1} \bigvee_{i=1}^{n} \mathrm{co}_i^-(\overline{A},\overline{B})(X){=}Y\end{aligned}$$

So (L.inverse) collects all correspondents of all variables bound by X; for each of these correspondents it must be known whether it is bound by Y or definitely bound by something else. Then we can determine $\lambda^{-1}(Y)$. The soundness of this rule is not obvious: is it really sufficient to look among the correspondents of $\lambda^{-1}(X)$ to compute $\lambda^{-1}(Y)$? The following proposition shows that it is.

Proposition 8 (Inverse lambda binding). *Suppose* $(\alpha_1, \ldots, \alpha_n) \sim (\alpha'_1, \ldots, \alpha'_n)$ *holds with correspondence functions* $c_1, \ldots, c_n$. *Then for all* $1 \leq k \leq n$ *and all* $\pi \in \mathsf{b}^-(\alpha_k)$,

$$\lambda^{-1}(c_k(\pi)) \subseteq \bigcup_{i=1}^{n} \{c_i(\pi') \mid \pi' \in \lambda^{-1}(\pi) \cap \mathsf{b}^-(\alpha_i)\}$$

Proof. Let $\omega \in \lambda^{-1}(c_k(\pi))$. The "no hanging binders" condition (hang) of Def. 6 is critical here: it enforces $\omega \in \bigcup_{i=1}^{n} \mathsf{b}^-(\alpha'_i)$. If $\omega \in \mathsf{b}^-(\alpha'_k)$, then there exists some $\pi' \in \mathsf{b}^-(\alpha_k)$ with $c_k(\pi') = \omega$. π' is var-labeled by Def. 3 and has a binder since λ is total. So we must have $\lambda(c_k(\pi')) = c_k(\lambda(\pi'))$ by condition (same.seg) of Def. 6. Now $\lambda(c_k(\pi')) = c_k(\pi)$ and c_k is a bijection, so $\pi' \in \lambda^{-1}(\pi)$. If, on the other hand, $\omega \notin \mathsf{b}^-(\alpha'_k)$ but $\omega \in \mathsf{b}^-(\alpha'_j)$, there is again a π' with $c_j(\pi') = \omega$, and $\lambda(c_j(\pi')) = c_k(\lambda(\pi'))$ by condition (diff.seg), so again $\pi' \in \lambda^{-1}(\pi)$.

The rules we have presented are part of a sound and complete semi-decision procedure for group parallelism constraints given in [3]. The omitted rules state additional properties of dominance constraints, ensure that correspondence functions are indeed bijective functions, and regulate the interaction between different correspondence functions.

Theorem 9. *There exists a saturation procedure GP which encompasses all instances of the rule schemata in Fig. 9, 10, and 11, such that each rule of GP is sound, and each GP-solved form of a constraint* φ *is satisfiable* **(soundness)**, *and for every solution* (τ, σ) *of* φ, *GP computes a GP-solved form of* φ *of which* (τ, σ) *is a solution* **(completeness)**.

Proving that *GP*-solved forms are satisfiable can be done by constructing a model and variable assignment explicitly. One then has to check that all literals are indeed satisfied, which requires a tedious case distinction. Proving completeness is nontrivial as well, but can be done along the lines of [9]. The proof is largely independent of the particularities of the rule system we employ.

7 The procedure in action

We illustrate the procedure of the previous section by solving the constraint in Fig. 12. It contains a non-linear lambda redex at (C, B, A) (similarly to Fig. 5) and a lambda binder at Y_1 which can either belong to the context C or argument A. The group parallelism constraint $(C, B, A, A) \sim (C', B', A', A'')$ describes a beta-reduction step for the redex (C, B, A).

A record of the solving steps is given in Fig. 13 and 14. We only comment on the main steps. In step (4), we have $Y_1 \triangleleft^* Z \wedge X_0 \triangleleft^* Z$, and as trees do not branch upwards, one of Y_1, X_0 must dominate the other. This step effectively guesses whether Y_1, Y_2 are in C or in A. With choice (5c), we make two copies of Y_1 and Y_2 each. This is because A is parallel both to A' and A'': because X_ℓ binds two

$(C, B, A, A) \sim (C', B', A', A'')$
with $C = X/X_0$,
$C' = X'/X'_0$,
$B = X_t/X_1, X_2$,
$B' = X'_0/X'_1, X'_2$,
$A = X_a/$,
$A' = X'_1/$, and $A'' = X'_2/$.
$\lambda^{-1}(X_\ell) = \{X_1, X_2\}$, $\lambda^{-1}(Y_1) = \{Z\}$.

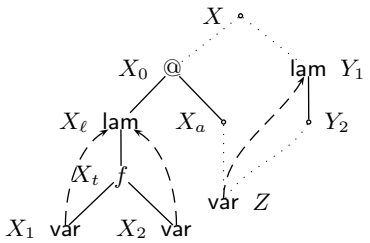


Fig. 12. A group parallelism constraint encoding a non-linear beta reduction step

variables, the argument is copied twice. On the other hand, with (7b) Y_1 and Y_2 are only copied once: they belong to the context C, which is parallel only to C'.

In Fig. 14, we continue case (5c) of Fig. 13, applying the lambda binding rules. All steps from (22) on prepare the determination of $\lambda^{-1}(Y'_1)$ and $\lambda^{-1}(Y''_1)$ in (25) and (26). We know $\lambda^{-1}(Y_1)$=$\{Z\}$. Steps (22) and (23) determine $S_2 \cup S_3$ to be $\{Y'_1, Y''_1\}$ for both (25) and (26). After (24) we know $\lambda(Z')\neq Y''_1$ and $\lambda(Z'')\neq Y'_1$, so we have all we need to infer the correct λ^{-1} information in the last steps.

8 Conclusion and future work

We have introduced *beta reduction constraints* and have presented a semi-decision procedure for processing them, in three steps: First, we have extended CLLS by group parallelism constraints. Second, we have expressed β-reduction constraints in this extension of CLLS. Third, we have lifted a known semi-decision procedure for CLLS to also deal with group parallelism constraints. It is an open question to what extent beta reduction constraints can conversely express parallelism.

This gives us a framework for investigating the more general problem of *underspecified beta reduction* [4]: How can we string together several reduction steps, as described by beta reduction constraints, until we arrive at descriptions of normal forms? In this broader setting, we can investigate properties such as confluence and termination on the underspecified level. Another problem, motivated by the application, is to modify the saturation procedure to perform as few case distinctions as possible during underspecified beta reduction. Finally, it will be interesting to find an efficient implementation of this operation, possibly employing concepts such as sharing and constraint programming.

References

1. M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.
2. R. Backofen, J. Rogers, and K. Vijay-Shanker. A first-order axiomatization of the theory of finite trees. *J. Logic, Language, and Information*, 4(1):5–39, 1995.

(1)	$Y_1 \neq X_0$	(D.lab.ineq)
(2)	$Y_1 \lhd^+ Y_2, X_0 \lhd^+ X_a$	(D.lab.dom)
(3)	$Y_1 \lhd^* Z, X_0 \lhd^* Z$	(D.dom.trans)
(4)	$X_0 \lhd^* Y_1 \ \vee \ Y_1 \lhd^* X_0$	(D.distr.notDisj)

(4a) $X_0 \lhd^* Y_1$:	(4b) $Y_1 \lhd^* X_0$:
(5) $X_0 = Y_1 \vee X_\ell \lhd^* Y_1 \vee X_a \lhd^* Y_1$ (D.distr.child)	(7) $Y_1 = X_0 \ \vee \ Y_2 \lhd^* X_0$ (D.distr.child)

(5a)	$X_0 = Y_1$:	(5b) $X_\ell \lhd^* Y_1$:	(7a)	$Y_1 = X_0$:	
...	both lead to **false**		(8)	**false**	(D.clash.ineq)
(5c)	$X_a \lhd^* Y_1$:		(7b)	$Y_2 \lhd^* X_0$:	
(9)	$\mathrm{co}(A, A')(X_a) = X'_1$	(P.init)			
(10)	$\mathrm{co}(A, A')(Y_1) = Y'_1$, $\mathrm{co}(A, A')(Y_2) = Y'_2$, $\mathrm{co}(A, A')(Z) = Z'$	(P.new)	(17)	$\mathrm{co}(C, C')(X) = X'$, $\mathrm{co}(C, C')(X_0) = X'_0$	(P.init)
(11)	$X'_1 \lhd^* Y'_1, Y'_2 \lhd^* Z'$	(P.copy.dom)	(18)	$\mathrm{co}(C, C')(Y_1) = Y'_1$, $\mathrm{co}(C, C')(Y_2) = Y'_2$	(P.new)
(12)	$Y'_1 : \mathsf{lam}(Y'_2)$	(P.copy.lab)	(19)	$X' \lhd^* Y'_1, Y'_2 \lhd^* X'_0$	(P.copy.dom)
(13)	$\mathrm{co}(A, A'')(X_a) = X'_2$	(P.init)	(20)	$Y'_1 : \mathsf{lam}(Y'_2)$	(P.copy.lab)
(14)	$\mathrm{co}(A, A'')(Y_1) = Y''_1$, $\mathrm{co}(A, A'')(Y_2) = Y''_2$, $\mathrm{co}(A, A'')(Z) = Z''$	(P.new)			
(15)	$X'_2 \lhd^* Y''_1, Y''_2 \lhd^* Z''$	(P.copy.dom)			
(16)	$Y''_1 : \mathsf{lam}(Y''_2)$	(P.copy.lab)			

Fig. 13. Solving the group parallelism constraint in Fig. 12

3. M. Bodirsky, K. Erk, A. Koller, and J. Niehren. Beta reduction constraints, 2001. Full version. `http://www.ps.uni-sb.de/Papers/abstracts/beta.html`.
4. M. Bodirsky, K. Erk, A. Koller, and J. Niehren. Underspecified beta reduction, 2001. Submitted. `http://www.ps.uni-sb.de/Papers/abstracts/usp-beta.html`.
5. D. Duchier and J. Niehren. Dominance constraints with set operators. In *First International Conference on Computational Logic (CL2000)*, LNCS, July 2000.
6. M. Egg, A. Koller, and J. Niehren. The Constraint Language for Lambda Structures. *Journal of Logic, Language, and Information*, 2001. To appear.
7. M. Egg, J. Niehren, P. Ruhrberg, and F. Xu. Constraints over lambda-structures in semantic underspecification. In *In Proceedings COLING/ACL'98*, 1998.
8. K. Erk, A. Koller, and J. Niehren. Processing underspecified semantic descriptions in CLLS. *Journal of Language & Computation*, 2001. To appear.
9. K. Erk and J. Niehren. Parallelism constraints. In *Int. Conference on Rewriting Techniques and Applications*, volume 1833 of *LNCS*, pages 110–126, 2000.
10. G. P. Huet. A unification algorithm for typed λ-calculus. *Theoretical Computer Science*, 1:27–57, 1975.
11. J. Lamping. An algorithm for optimal lambda calculus reduction. In *ACM Symposium on Principles of Programming Languages*, pages 16–30, 1990.
12. J. Lévy. Linear second order unification. In *7th Int. Conference on Rewriting Techniques and Applications*, volume 1103 of *LNCS*, pages 332–346, 1996.
13. M. P. Marcus, D. Hindle, and M. M. Fleck. D-theory: Talking about talking about trees. In *Proceedings of the 21st ACL*, pages 129–136, 1983.
14. M. Pinkal. Radical underspecification. In *Proc. 10th Amsterdam Colloquium*, 1996.
15. U. Reyle. Dealing with ambiguities by underspecification: construction, representation, and deduction. *Journal of Semantics*, 10:123–179, 1993.

	Continuing (5c)		
(21)	$\lambda(Z') = Y_1'$, $\lambda(Z'') = Y_1''$ (L.same.seg)		
(22)	$Z \not\in \mathsf{b}^-(B) \ \vee \ Z \in \mathsf{b}^-(B)$ (L.distr.2)		
(22a)	$Z \not\in \mathsf{b}^-(B)$	(22b)	$Z \not\in \mathsf{b}^-(B)$
(23)	$Z \not\in \mathsf{b}^-(C) \ \vee \ Z \in \mathsf{b}^-(C)$ (L.distr.2)	...	**false**
(23a)	$Z \not\in \mathsf{b}^-(C)$	(23b)	$Z \in \mathsf{b}^-(C)$
(24)	$Y_1' \neq Y_1'' \ \vee \ Y_1' = Y_1''$ (P.distr.eq)	...	**false**
(24a)	$Y_1' \neq Y_1''$	(24b)	$Y_1' = Y_1''$
(25)	$\lambda^{-1}(Z'') \neq Y_1'$	...	**false**
(26)	$\lambda^{-1}(Y_1'') = \{Z'\}$ (L.inverse)		
(27)	$\lambda^{-1}(Y_1'') = \{Z''\}$ (L.inverse)		

Fig. 14. Inverse Binding in case (5c)

16. M. Schmidt-Schauß and K. Schulz. Solvability of context equations with two context variables is decidable. In *Int. Conf. on Automated Deduction*, LNCS, 1999.
17. K. van Deemter and S. Peters. *Semantic Ambiguity and Underspecification*. CSLI Press, Stanford, 1996.

A The procedure *GP*

In this section, we list all rule schemata of the saturation procedure *GP* for group parallelism constraints.

We have said some words on saturation earlier, but have left things rather informal. We remedy that now. The *saturation rules* we work with are implications of the following form:

$$\varphi_0 \to \vee_{i=1}^n \exists V_i \varphi_i$$

where $n \geq 1$ and $\mathcal{V}(\varphi_i) - \mathcal{V}(\varphi_0) \subseteq V_i$ for all $1 \leq i \leq n$. A rule is called a *propagation rule* if $n = 1$ and a *distribution rule* otherwise.

A saturation algorithm S is a set of saturation algorithms. A *saturation step* $\to_S$ consists of one application of a rule in S:

$$\frac{\varphi' \subseteq \varphi \qquad \rho \in S}{\varphi \to_S \varphi \wedge \exists V_i \varphi_i} \text{ if } C_\rho(\varphi) \text{ where } \rho \text{ is } \varphi' \to \vee_{i=1}^n \exists V_i \varphi_i$$

Let $\mathcal{V}$ be the set of all node variables. Given a set V of variables and a constraint φ, we call a constraint $\zeta\varphi$ a *V-variant of φ* if $\zeta : V \to \mathcal{V}$ is some substitution of the variables in V. Then we let $C_{\varphi' \to \vee_{i=1}^n V_i \varphi_i}(\varphi)$ be true iff for all $1 \leq i \leq n$ and for all V_i-variants φ_i' of φ_i, we have $\varphi_i' \not\subseteq \varphi$. That is, we only apply a rule when it can add something new.

Let S be a set of saturation rules. As mentioned before, we call a constraint *saturated* (under S) if no further rule of S applies to it. We say that a constraint is in *S-solved form* if it is saturated under S and clash-free (i.e. it does not contain **false**).

The rules we present here only make use of part of the language of CLLS with group parallelism. We use $\lhd^*, \bot$ and $\neq$, but we do not process literals of the form $X(R \cup R')Y$ or $X(R \cap R')Y$. Instead, we interpret $\cup$ as disjunction and $\cap$ as conjunction. So XRY becomes another constraint abbreviation.

The saturation procedure GP that we present consists of four modules: The rule system D deals with constraints without parallelism and binding; rule system B states properties of λ-binding, system P is concerned with parallelism, and rule system L handles the interaction of parallelism and binding.

We first list the rule schemata of system D, which deals with *Dominance Constraints*, the sublanguage of group parallelism constraints which only comprises dominance, labeling, disjointness, and inequality literals.

Solving dominance constraints: rule system D

(D.clash.ineq) $X{=}Y \wedge X{\neq}Y \rightarrow \textbf{false}$

(D.clash.disj) $X \bot X \rightarrow \textbf{false}$

(D.dom.refl) $\varphi \rightarrow X \lhd^* X$ where $X \in \mathcal{V}(\varphi)$

(D.dom.trans) $X \lhd^* Y \wedge Y \lhd^* Z \rightarrow X \lhd^* Z$

(D.eq.decom) $X{:}f(\overline{X}) \wedge Y{:}f(\overline{Y}) \wedge X{=}Y \rightarrow \wedge_{i=1}^{n} X_i{=}Y_i$

(D.lab.ineq) $X{:}f(\ldots) \wedge Y{:}g(\ldots) \rightarrow X{\neq}Y$ where $f \neq g$

(D.lab.disj) $X{:}f(\ldots X_i, \ldots, X_j, \ldots) \rightarrow X_i \bot X_j$ for $1 \leq i < j \leq n$

(D.prop.disj) $X \bot Y \wedge X \lhd^* X' \wedge Y \lhd^* Y' \rightarrow Y' \bot X'$

(D.lab.dom) $X{:}f(\ldots, Y, \ldots) \rightarrow X \lhd^+ Y$

(D.distr.notDisj) $X \lhd^* Z \wedge Y \lhd^* Z \rightarrow X \lhd^* Y \vee Y \lhd^* X$

(D.distr.child) $X \lhd^* Y \wedge X{:}f(\overline{X}) \rightarrow Y{=}X \vee \bigvee_{i=1}^{n} X_i \lhd^* Y$

The rule system B is all we need to deal with λ-binding in the absence of parallelism. The rules state that λ is a function, that binders dominate their bound variables, that binders go from var-labeled nodes to nodes labeled lam, $\exists$ or $\forall$, and that a λ^{-1} literal specifies *all* variables bound by a certain λ-binder.

Properties of λ**-binding: rule system** B

(B.func) $\lambda(X){=}Y \wedge \lambda(U){=}V \wedge X{=}U \rightarrow Y{=}V$

(B.dom) $\lambda(X){=}Y \rightarrow Y \lhd^* X$

(B.var) $\lambda(X){=}Y \rightarrow X{:}var$

(B.lam) $\lambda(X){=}Y \rightarrow \exists Z\ (Y{:}lam(Z) \vee Y{:}\exists(Z) \vee Y{:}\forall(Z))$

(B.inv) $\lambda^{-1}(X){=}\{X_1, \ldots, X_n\} \rightarrow \bigwedge_{i=1}^{n} \lambda(X_i){=}X$

(B.all) $\lambda^{-1}(X){=}\{X_1, \ldots, X_n\} \wedge \lambda(Y){=}X \rightarrow \bigvee_{i=1}^{n} Y{=}X_i$

System P contains rules for handling parallelism, and rules that describe the interaction between different correspondence functions. We first list the rules that deal with parallelism. Let $A = X_0/X_1, \ldots, X_n$. As in Section 6, we use the formula

$$\mathsf{seg}(A) =_{\mathsf{df}} \wedge_{i=1}^{n} X_0 \triangleleft^* X_i \ \wedge \ \wedge_{1 \leq i < j \leq n} X_i (\bot \cup =) X_j,$$

furthermore the formulas

$$\begin{aligned} X \in \mathsf{b}(A) &=_{\mathsf{df}} X_0 \triangleleft^* X \wedge \textstyle\bigwedge_{i=1}^{n} X (\triangleleft^* \cup \bot) X_i \\ X \in \mathsf{b}^-(A) &=_{\mathsf{df}} X \in \mathsf{b}(A) \wedge \textstyle\bigwedge_{i=1}^{n} X \neq X_i \\ X \notin \mathsf{b}^-(A) &=_{\mathsf{df}} X (\triangleleft^+ \cup \bot) X_0 \vee \textstyle\bigvee_{i=1}^{n} X_i \triangleleft^* X \\ X \in \mathsf{b}(A_1, \ldots, A_m) &=_{\mathsf{df}} \textstyle\bigvee_{i=1}^{m} X \in \mathsf{b}(A_i) \\ X \in \mathsf{b}^-(A_1, \ldots, A_m) &=_{\mathsf{df}} \textstyle\bigvee_{i=1}^{m} X \in \mathsf{b}^-(A_i) \end{aligned}$$

as well as

$$\begin{aligned} \mathrm{co}^-(A, B)(X) = Y &=_{\mathsf{df}} \mathrm{co}(A, B)(X) = Y \wedge X \in \mathsf{b}^-(A) \\ \mathrm{co}^-_k(\overline{A}, \overline{B})(X) = Y &=_{\mathsf{df}} \overline{A} \sim \overline{B} \wedge \mathrm{co}^-(A_k, B_k)(X) = Y \end{aligned}$$

Let $\overline{A} = A_1, \ldots, A_n$ and $\overline{B} = B_1, \ldots, B_n$.

Properties of parallelism literals: rule system P

(P.symm) $\overline{A} \sim \overline{B} \rightarrow \overline{B} \sim \overline{A}$

(P.init) $\overline{A} \sim \overline{B} \rightarrow \mathsf{seg}(A_i) \wedge \mathrm{co}(A_i, B_i)(X_i^j) = Y_i^j$ where $1 \leq i \leq n$, $A_i = X_i^0/X_i^1, \ldots, X_i^{m_i}$, $B_i = X_i^0/Y_i^1, \ldots, Y_i^{m_i}$, and $0 \leq j \leq m_i$

(P.new) $\overline{A} \sim \overline{B} \wedge U \in \mathsf{b}(A_i) \rightarrow \exists V \ \mathrm{co}(A_i, B_i)(U) = V$ where V fresh, $1 \leq i \leq n$

(P.copy.lab) $\bigwedge_{i=0}^{m} \mathrm{co}(A, B)(X_i) = Y_i \wedge X_0 : f(X_1, \ldots, X_m) \wedge X_0 \in \mathsf{b}^-(A) \rightarrow Y_0 : f(Y_1, \ldots, Y_m)$

(P.copy.dom) $U_1 \, R \, U_2 \wedge \bigwedge_{i=1}^{2} \mathrm{co}(A, B)(U_i) = V_i \rightarrow V_1 \, R \, V_2$

(P.distr.eq) $\varphi \rightarrow X = Y \vee X \neq Y$ for $X, Y \in \mathcal{V}(\varphi)$

(P.distr.hole) $\overline{A} \sim \overline{B} \wedge X_i^0 \triangleleft^* X \rightarrow X \in \mathsf{b}(A_i) \vee \bigvee_{j=1}^{m_i} X_i^j \triangleleft^+ X$ where $1 \leq i \leq n$, $A_i = X_i^0/X_i^1, \ldots, X_i^{m_i}$

In section 6 we have introduced correspondence literals $\mathrm{co}(A, B)(U) = V$ as auxiliary literals to record correspondence. Actually, correspondence literals are just an abbreviation; what we really use are *path equalities*. Informally speaking, if a path equality $\mathrm{p}\left(\begin{smallmatrix} \pi_1 & \pi_3 \\ \pi_2 & \pi_4 \end{smallmatrix}\right)$ holds in a tree, that means that the path from π_1 to π_2, including the node labels passed on the way, is the same as the path from π_3 to π_4. More precisely, the path equality relation $\mathrm{p}\left(\begin{smallmatrix} \cdot & \cdot \\ \cdot & \cdot \end{smallmatrix}\right)$ on a tree θ is the greatest relation on node quadruples such that the following holds: $\mathrm{p}\left(\begin{smallmatrix} \pi_1 & \pi_3 \\ \pi_2 & \pi_4 \end{smallmatrix}\right)$ is true iff there exists a path π such that $\pi_2 = \pi_1 \pi$ and $\pi_4 = \pi_3 \pi$, and for each $\pi' \triangleleft^+ \pi$, $L_\theta(\pi_1 \pi') = L_\theta(\pi_3 \pi')$. At the same time, suppose we have parallel tree segments α and β and a node $\pi \in \mathsf{b}(\alpha)$; then if the path equality $\mathrm{p}\left(\begin{smallmatrix} r(\alpha) & r(\beta) \\ \pi & \pi' \end{smallmatrix}\right)$ holds, then

π in α *corresponds to* π' in β. That is,

$$\mathrm{co}(A_i, B_i)(U){=}V \text{ stands for } \overline{A}{\sim}\overline{B} \wedge \mathrm{p}\binom{X_0\ Y_0}{U\ V} \wedge U{\in}\mathsf{b}(A)$$

for $1 \leq i \leq |\overline{A}|$, and $A_i = X_0/X_1, \ldots, X_n$ and $B_i = Y_0/Y_1, \ldots, Y_n$.

The main idea about path equalities is that, as they possess a semantics of their own, they have properties that we can use, irrespective of which correspondence function the path equalities in question come from. This is exploited by the saturation rule schemata (P.path...), (P.trans...), and (P.diff...). These schemata ensure the correct interaction of correspondence functions. See [9] for a comprehensive treatment of this topic.

Properties of Path Equality Constraints

(P.path.symm) $\mathrm{p}\binom{X\ Y}{U\ V} \rightarrow \mathrm{p}\binom{Y\ X}{V\ U}$

(P.path.dom) $\mathrm{p}\binom{X\ Y}{U\ V} \rightarrow X{\lhd^*}U \wedge Y{\lhd^*}V$

(P.path.eq.1) $\mathrm{p}\binom{X_1\ X_3}{X_2\ X_4} \wedge \bigwedge_{i=1}^{4} X_i{=}Y_i \rightarrow \mathrm{p}\binom{Y_1\ Y_3}{Y_2\ Y_4}$

(P.path.eq.2) $\mathrm{p}\binom{X\ X}{U\ V} \rightarrow U{=}V$

(P.trans.h) $\mathrm{p}\binom{X\ Y}{U\ V} \wedge \mathrm{p}\binom{Y\ Z}{V\ W} \rightarrow \mathrm{p}\binom{X\ Z}{U\ W}$

(P.trans.v) $\mathrm{p}\binom{X_1\ Y_1}{X_2\ Y_2} \wedge \mathrm{p}\binom{X_2\ Y_2}{X_3\ Y_3} \rightarrow \mathrm{p}\binom{X_1\ Y_1}{X_3\ Y_3}$

(P.diff.1) $\mathrm{p}\binom{X_1\ Y_1}{X_2\ Y_2} \wedge \mathrm{p}\binom{X_1\ Y_1}{X_3\ Y_3} \wedge X_2{\lhd^*}X_3 \wedge Y_2{\lhd^*}Y_3 \rightarrow \mathrm{p}\binom{X_2\ Y_2}{X_3\ Y_3}$

(P.diff.2) $\mathrm{p}\binom{X_1\ Y_1}{X_3\ Y_3} \wedge \mathrm{p}\binom{X_2\ Y_2}{X_3\ Y_3} \wedge X_1{\lhd^*}X_2 \wedge Y_1{\lhd^*}Y_2 \rightarrow \mathrm{p}\binom{X_1\ Y_1}{X_2\ Y_2}$

Rule system L describes the interaction between parallelism and binding. We use the formula

$$\mathsf{distr}_{\overline{A}}(U) =_{\mathsf{df}} \bigwedge_{i=1}^{n} \big(U{\in}\mathsf{b}^-(A_i) \vee U{\notin}\mathsf{b}^-(A_i)\big).$$

in (L.distr.1) and (L.distr.2). (L.inverse) uses two more formulas, the first just being

$$\lambda(X){\neq}Y =_{\mathsf{df}} \exists Z(\lambda(X){=}Z \wedge Z{\neq}Y).$$

As explained in Section 6, the second formula collects, for a finite set S_1 of variables, all correspondents with respect to $\overline{A} \sim \overline{B}$. Let S_1, S_2 stand for finite sets of variables, and let $\overline{A} = A_1 \ldots, A_n$. Then

$$\begin{array}{c}\mathrm{co}^-(\overline{A},\overline{B})(S_1){=}S_2 =_{\mathsf{df}} \bigwedge_{i=1}^{n} \bigwedge_{X\in S_1} (X{\notin}\mathsf{b}^-(A_i) \vee \bigvee_{Y\in S_2} \mathrm{co}_i^-(\overline{A},\overline{B})(X){=}Y) \\ \wedge\ \bigwedge_{Y\in S_2} \bigvee_{X\in S_1} \bigvee_{i=1}^{n} \mathrm{co}_i^-(\overline{A},\overline{B})(X){=}Y\end{array}$$

Interaction between parallelism and binding: rule system L

(L.same.seg) $\lambda(U_1){=}U_2 \wedge \bigwedge_{i=1}^{2} \mathrm{co}_k^-(\overline{A},\overline{B})(U_i){=}V_i \rightarrow \lambda(V_1){=}V_2$

(L.diff.seg) $\lambda(U_1){=}U_2 \wedge \bigwedge_{i=1}^{2} \mathrm{co}_{k_i}^-(\overline{A},\overline{B})(U_i){=}V_i \wedge U_2{\notin}\mathsf{b}^-(A_{k_1}) \rightarrow \lambda(V_1){=}V_2$

(L.outside) $\lambda(U){=}Y \wedge \mathrm{co}_k^-(\overline{A},\overline{B})(U){=}V \wedge Y{\notin}\mathsf{b}^-(\overline{A}) \rightarrow \lambda(V){=}Y$

(L.hang) $\lambda(U_1){=}U_2 \wedge \overline{A}{\sim}\overline{B} \wedge U_2{\in}\mathsf{b}^-(\overline{A}) \rightarrow U_1{\in}\mathsf{b}^-(\overline{A})$

(L.distr.1) $\lambda(U_1){=}U_2 \wedge \overline{A}{\sim}\overline{B} \wedge U_1{\in}\mathsf{b}^-(\overline{A}) \rightarrow \mathsf{distr}_{\overline{A}}(U_2)$

(L.distr.2) $\lambda(U_1){=}U_2 \wedge \overline{A}{\sim}\overline{B} \wedge U_2 \in \mathsf{b}^-(\overline{A}) \rightarrow \mathsf{distr}_{\overline{A}}(U_1)$

(L.equal) $\lambda(X_1){=}X_2 \wedge \bigwedge_{i=1}^{2} X_i{=}Y_i \rightarrow \lambda(Y_1){=}Y_2$

(L.inverse) $\lambda^{-1}(X){=}S_1 \wedge \mathrm{co}_k^-(\overline{A},\overline{B})(X){=}Y \wedge \mathrm{co}^-(\overline{A},\overline{B})(S_1){=}S_2 \cup S_3 \wedge \bigwedge_{V\in S_2} \lambda(V){=}Y \wedge \bigwedge_{V\in S_3} \lambda(V){\neq}Y \rightarrow \lambda^{-1}(Y){=}S_2$

B Soundness and Completeness of *GP*

In this section, we show first the soundness, then the completeness of the procedure *GP* we have presented in the previous section.

B.1 Soundness

We call a single saturation rule $\varphi \rightarrow \Phi$ *sound* iff $\varphi \models \Phi$. We call a saturation procedure $\mathcal{S}$ *sound* iff each rule in $\mathcal{S}$ is sound and each $\mathcal{S}$-solved form of a constraint is satisfiable.

The soundness of all rule schemata except (L.inverse) is obvious. The soundness of (L.inverse) is shown by Proposition 8.

In the rest of this section, we show that from every constraint in *GP*-solved form, a solution can be read off. We proceed in two steps. First, we show that a special class of *GP*-solved forms, called "simple", are satisfiable. Then we lift the result to arbitrary *GP*-solved forms.

We only regard *generated* constraints, where each path equality either establishes a correspondence for some pair A_i, B_i of a group parallelism literal $(A_1,\ldots,A_n) \sim (B_1,\ldots,B_n)$ or is the result of combining several such correspondence statements by a (P.trans...) or (P.diff...) rule.

Definition 10. Let φ be a constraint.

A path equality $\mathrm{p}(\begin{smallmatrix}U_1 & V_1\\ U_2 & V_2\end{smallmatrix}) \in \varphi$ is *correspondence-generated in* φ iff there exists some group parallelism literal $(A_1,\ldots,A_n) \sim (B_1,\ldots,B_n)$ and some $i \in \{1,\ldots,n\}$ such that on the one hand, $A_i = U_1/\ldots$ and $B_i = V_1/\ldots$, and on the other hand either $U_2{\in}\mathsf{b}(A_i)$ or $V_2{\in}\mathsf{b}(B_i)$ is in φ.

Let $CP(\varphi)$ be the set of correspondence-generated path equalities in φ, and let φ_0 be φ without all its path equalities, then a path equality is *generated in* φ iff it is in the saturation of $CP(\varphi) \cup \varphi_0$ under the instances of (P.trans.h), (P.trans.v), (P.diff.1), (P.diff.2).

φ is called *generated* iff each of its parallelism literals is.

Every *GP*-solved form of a parallelism constraint is generated, so we can safely restrict our attention to generated constraints:

Lemma 11. *Let φ be a constraint without path equalities, and let $\varphi \rightarrow^*_{GP} \varphi'$ with φ' in GP-solved form. Then φ' is generated.*

Proof. Let $\varphi_1, \ldots, \varphi_\ell$ be a sequence of constraints such that $\varphi_1 = \varphi$, $\varphi_\ell = \varphi'$, and $\varphi_i \rightarrow_{GP} \varphi_{i+1}$ for $1 \leq i \leq \ell - 1$. We show by induction on i that

1. each $\mathrm{p}(\begin{smallmatrix} X & Y \\ U & V \end{smallmatrix}) \in \varphi_i$ is generated in φ',
2. alongside with $\mathrm{p}(\begin{smallmatrix} Y & X \\ V & U \end{smallmatrix})$ and every $\mathrm{p}(\begin{smallmatrix} X' & Y' \\ U' & V' \end{smallmatrix})$ with $X'{=}X$, $U'{=}U$, $Y'{=}Y$, $V'{=}V \in \varphi'$.

φ_1 contains no path qualities. So let $\varphi_i \rightarrow_{\{\rho\}} \varphi_{i+1}$, where ρ is an instance of (P.init), (P.path.symm), (P.path.eq.1), (P.new), (P.trans.h), (P.trans.v), (P.diff.1), or (P.diff.2).

Suppose ρ is an instance of (P.init). Then the lhs of ρ is some group parallelism literal $(A_1, \ldots, A_n) \sim (B_1, \ldots, B_n)$ of φ, and there exists some $k \in \{1, \ldots, n\}$ such that $A_k = X_0/X_1, \ldots, X_m$, $B_k = Y_0/Y_1, \ldots, Y_m$, and φ_{i+1} contains one path equality $\mathrm{p}(\begin{smallmatrix} X_0 & Y_0 \\ X_j & Y_j \end{smallmatrix})$ (with $j \in \{1, \ldots, m\}$) that is not in φ_i. As ρ has also inferred $\mathsf{seg}(A_k)$, we have $X_j{\in}\mathsf{b}(A_k)$ in φ_{i+1}. So $\mathrm{p}(\begin{smallmatrix} X_0 & Y_0 \\ X_j & Y_j \end{smallmatrix})$ is correspondence-generated in φ'. Condition 2 from above holds by closure of φ' under (P.path.symm), (P.path.eq.1), (D.dom.refl) and (D.dom.trans).

Suppose ρ is an instance of (P.new). Then the lhs of ρ has the form $(A_1, \ldots, A_n) \sim (B_1, \ldots, B_n) \wedge U{\in}\mathsf{b}(A_k)$ for some $k \in \{1, \ldots, n\}$. Let $A_k = X_0/X_1, \ldots, X_m$ and $B_k = Y_0/Y_1, \ldots, Y_m$, then φ_{i+1} contains one path equality $\mathrm{p}(\begin{smallmatrix} X_0 & Y_0 \\ U & V \end{smallmatrix})$ that is not in φ_i. Then $\mathrm{p}(\begin{smallmatrix} X_0 & Y_0 \\ U & V \end{smallmatrix})$ is correspondence-generated in φ' by definition. Condition 2 holds by closure of φ' under (P.path.symm), (P.path.eq.1), (D.dom.trans) and (D.prop.disj).

If ρ is an instance of (P.trans.h), (P.trans.v), (P.diff.1) or (P.diff.2), and φ_{i+1} has the form $\varphi_i \wedge \mathrm{p}(\begin{smallmatrix} X & Y \\ U & V \end{smallmatrix})$, then $\mathrm{p}(\begin{smallmatrix} X & Y \\ U & V \end{smallmatrix})$ is generated by definition. Concerning condition 2, we just consider the case of (p.trans.h), the others are analogous. Suppose ρ has the form $\mathrm{p}(\begin{smallmatrix} X & Z \\ U & W \end{smallmatrix}) \wedge \mathrm{p}(\begin{smallmatrix} Z & Y \\ W & V \end{smallmatrix}) \rightarrow \mathrm{p}(\begin{smallmatrix} X & Y \\ U & V \end{smallmatrix})$. Then $\mathrm{p}(\begin{smallmatrix} Z & X \\ W & U \end{smallmatrix}), \mathrm{p}(\begin{smallmatrix} Y & Z \\ V & W \end{smallmatrix})$ are in φ' by closure under (P.trans.h) and generated by the inductive hypothesis. So $\mathrm{p}(\begin{smallmatrix} Y & X \\ V & U \end{smallmatrix}) \in \varphi'$ is generated in φ' as well. The case of a literal $\mathrm{p}(\begin{smallmatrix} X' & Y' \\ U' & V' \end{smallmatrix})$ where $X'{=}X, U'{=}U, Y'{=}Y, V'{=}V \in \varphi'$ is analogous.

If ρ is an instance of (P.path.symm) or (P.path.eq.1) and φ_{i+1} has the form $\varphi_i \wedge \mathrm{p}(\begin{smallmatrix} X & Y \\ U & V \end{smallmatrix})$, then $\mathrm{p}(\begin{smallmatrix} X & Y \\ U & V \end{smallmatrix})$ is generated in φ' because of inductive hypothesis 2.

B.2 Soundness: Simple constraints

Definition 12. A constraint φ is called *simple* if all its variables are labeled and there exists some *root variable* $Z \in \mathcal{V}(\varphi)$ such that $Z{\triangleleft^*}X$ is in φ for all $X \in \mathcal{V}(\varphi)$.

Proposition 13. *A simple generated constraint in GP-solved form is satisfiable.*

Proof. Let φ be a simple generated constraint in *GP*-solved form, and let φ_{dom} be the maximal subset of φ that contains no group parallelism literals, no path equalities, and no lambda binding literals. φ_{dom} is tree-shaped since each of its variables is labeled. [9] gives an inductive construction of a solution (τ_{dom}, σ) for φ_{dom}: Suppose Z is a root variable in φ. Since all variables in φ are labeled, there

is a variable Z' and a term $f(Z_1, \ldots, Z_n)$ such that $Z{=}Z'$ and $Z'{:}f(Z_1, \ldots, Z_n)$ are in φ. Let

$$V = \{X \in \mathcal{V}(\varphi) \mid Z{=}X \in \varphi\} \text{ and } V_i = \{X \in \mathcal{V}(\varphi) \mid Z_i \triangleleft^* X \in \varphi\}$$

for all $1 \leq i \leq n$. Then $\mathcal{V}(\varphi) = V \cup V_1 \cup \ldots \cup V_n$. For a set $W \subseteq \mathcal{V}(\varphi)$ we define $\varphi_{|W}$ as the conjunction of all literals $\psi \in \varphi$ with $\mathcal{V}(\psi) \subseteq W$. Then

$$\varphi \models\!\!\models \varphi' \quad \text{holds where} \quad \varphi' := \varphi_{|V} \wedge Z{:}f(Z_1, \ldots, Z_n) \wedge \bigwedge_{i=1}^{n} \varphi_{|V_i}$$

because φ is in D-solved form: Each literal in φ is entailed by φ'. Next note that all $\varphi_{|V_i}$ are simple D-solved forms. By the inductive hypothesis there exist solutions $(\tau_i, \sigma_i) \models \varphi_{|V_i}$ for all $1 \leq i \leq n$. If $\tau_i = (\theta_i, \lambda_i)$, then the function λ_i is undefined everywhere. Let λ' be a function with empty domain, let $\theta_{dom} = f(\theta_1, \ldots, \theta_n)$, and let $\tau_{dom} = (\theta_{dom}, \lambda')$. Then (τ_{dom}, σ) is a solution of φ if $\sigma_{|V_i} = \sigma_i$ and $\sigma(X) = \sigma(Z)$ is the root node of θ_{dom} for all $X \in V$. Note that by this construction, if $\pi \in D_{\theta_{dom}}$, then there exists some $X \in \mathcal{V}(\varphi)$ with $\sigma(X) = \pi$.

Lambda binding. Now let φ_{base} be φ_{dom} plus all lambda binding constraints in φ. We construct a solution for φ_{base}. First for the tree: we have to make sure that every var-labeled node possesses a binder. Suppose φ_{base} contains ℓ var-labeled variables $U_1, \ldots, U_\ell$ without a binder. Then we construct the new tree θ by adding ℓ lam-labeled nodes "above" θ_{dom}: let $\theta = \underbrace{\mathsf{lam}(\ldots \mathsf{lam}}_{\ell \text{ times}}(\theta_{dom}) \ldots)$. Now we construct the λ function of the solution. We define $\lambda : \sigma(\{X \in \mathcal{V}(\varphi) \mid X{:}\mathsf{var} \in \varphi\}) \to \sigma(\{Y \in \mathcal{V}(\varphi) \mid \exists Z.Y{:}\mathsf{lam}(Z) \in \varphi\}) \cup \{1^i \mid 0 \leq i \leq \ell - 1\}$ as follows: for all $X, Y \in \mathcal{V}(\varphi)$, $\lambda(\sigma(X)) = \sigma(Y)$ iff $\lambda(X){=}Y \in \varphi$; furthermore, for $1 \leq i \leq \ell$, $\lambda(\sigma(U_i)) = 1^{i-1}$.

The function λ is well-defined: φ is saturated under (B.func) and by the construction given above, if $X{=}U$ is not in φ, then $\sigma(X) \neq \sigma(U)$. Furthermore, $U_1, \ldots, U_\ell$ do not have a binder in $\mathcal{V}(\varphi)$.

If $\sigma(X)$ is in the domain of λ, then X is labeled var by closure of φ under (B.var). If $\sigma(Y)$ is in the range of λ, then Y is labeled lam by closure of φ under (B.lam) and by the construction of θ from θ_{dom}. As (τ_{dom}, σ) is a solution of φ_{dom}, the domain of λ is $L_\theta^{-1}(\mathsf{var})$, and its range is a subset of $L_\theta^{-1}(\mathsf{lam})$. The function λ is total on $L_\theta^{-1}(\mathsf{var})$ because φ is simple and because we have added binders for $\sigma(U_1), \ldots, \sigma(U_\ell)$.

Each var-labeled node is dominated by its binder because φ is saturated under (B.dom).

Let $\tau = (\theta, \lambda)$. Then (τ, σ) is a solution of φ_{base} because (τ_{dom}, σ) is a solution of φ_{dom}, and for every literal $\lambda(X){=}Y \in \varphi$, we have $\lambda(\sigma(X)) = \sigma(Y)$ in the lambda structure by the construction of the λ function.

In a last step, we move from φ_{base} to φ, showing that all path equality literals and all group parallelism literals of φ are satisfied in (τ, σ).

Path equality literals. Let

$$\varphi' = \varphi_{dom} \cup \{\overline{A} \sim \overline{B} \in \varphi\} \cup \\ \{\mathrm{p}\binom{X\ Y}{U\ V} \in \varphi \mid \exists f, m, i, X_1, \ldots, X_m.\ X{:}f(X_1, \ldots, X_m), X_i{=}U \in \varphi\}.$$

We can restrict our consideration to the path equalities in φ', i.e. $\varphi \models\!\!\!\models \varphi'$: $\varphi \models \varphi'$ since $\varphi' \subseteq \varphi$. The opposite direction, $\varphi' \models \varphi$, can be seen from the fact that (P.path.eq.1), (P.trans.h), (P.trans.v), (P.diff.1) and (P.diff.2) are sound.

Now it remains to show that each path equality in φ' is satisfied by (τ, σ). For any $\mathrm{p}\binom{X\ Y}{U\ V} \in \varphi'$, there exist literals $X{:}f(X_1, \ldots, X_\ell), X_j{=}U \in \varphi$. Suppose we can show that in that case, there are $Y_1, \ldots, Y_\ell$ such that $Y{:}f(Y_1, \ldots, Y_\ell), Y_j{=}V \in \varphi$. Then it is easy to show that $\mathrm{p}\binom{\sigma(X)\ \sigma(Y)}{\sigma(U)\ \sigma(V)}$ holds in τ: by the way we have constructed θ above, the subtree θ_X of θ with root $\sigma(X)$ is labeled f, as is the subtree θ_Y of θ with root $\sigma(Y)$, and the path from $\sigma(X)$ to $\sigma(X_i) = \sigma(U)$ in θ_X is i, as is the path from $\sigma(Y)$ to $\sigma(Y_i) = \sigma(V)$ in θ_Y.

It remains to show that for any $\mathrm{p}\binom{X\ Y}{U\ V} \in \varphi'$, if $X{:}f(X_1, \ldots, X_\ell), X_j{=}U \in \varphi$, then there exist $Y_1, \ldots, Y_\ell$ such that $Y{:}f(Y_1, \ldots, Y_\ell), Y_j{=}V \in \varphi$. We show a stronger claim: for any $\mathrm{p}\binom{X\ Y}{U\ V} \in \varphi$, we get the same sequence of labels on the path from X to U as on the path from Y to V.

Let $\mathrm{p}\binom{X\ Y}{U\ V}$ be a path equality literal in φ. As φ is simple, there exist $X_0, \ldots, X_m \in \mathcal{V}(\varphi)$ for some $m \geq 0$ such that $X_0{=}X, X_m{=}U \in \varphi$ and for all $0 \leq i \leq m-1$, $X_i{:}f_i(X_1^i, \ldots, X_{\ell_i}^i) \in \varphi$ for some $X_1^i, \ldots, X_{\ell_i}^i \in \mathcal{V}(\varphi)$ and $f_i \in \theta$ of arity ℓ_i, and $X_{j_i}^i{=}X_{i+1} \in \varphi$ for some $j_i \in \{1, \ldots, \ell_i\}$. Note that m and the f_i, $1 \leq i \leq m$, are unique as φ is clash-free and closed under (D.Distr.NotDisj), (D.Distr.Child) and (D.Lab.Ineq). We show the following: for all $0 \leq i \leq m$, $\mathrm{p}\binom{X\ Y}{X_i\ Y_i} \in \varphi$ for some $Y_i \in \mathcal{V}(\varphi)$ in such a way that for $0 \leq i \leq m-1$, $Y_i{:}f_i(Y_1^i, \ldots, Y_{\ell_i}^i) \in \varphi$ for some $Y_1^i, \ldots, Y_{\ell_i}^i \in \mathcal{V}(\varphi)$, and $Y_{j_i}^i{=}Y_{i+1} \in \varphi$ (for the same j_i as in $X_{j_i}^i$ above). We use induction on the length of a proof of generatedness for $\mathrm{p}\binom{X\ Y}{U\ V}$.

Suppose $\mathrm{p}\binom{X\ Y}{U\ V}$ is correspondence-generated. That means that there exists some group parallelism literal $(A_1, \ldots, A_n) \sim (B_1, \ldots, B_n)$ and some $k \in \{1, \ldots, n\}$ with $A_k = X/\ldots$ and $B_k = Y\ldots$ such that $U{\in}\mathsf{b}(A_k)$ is in φ. Then $V{\in}\mathsf{b}(B_k)$ is in φ by (P.copy.dom). We proceed by induction on m.

Suppose $m = 0$. Then $X{=}U \in \varphi$ by closure under (D.dom.trans). Also, $Y{=}V \in \varphi$ by closure under (P.copy.dom).

Now suppose $m > 0$. Suppose that $X_{m-1}{:}f(X_1^{m-1}, \ldots, X_\ell^{m-1}), X_j^{m-1}{=}U$ are in φ. As $\mathrm{p}\binom{X\ Y}{U\ V}$ is correspondence-generated, we must have $X_{m-1}{\in}\mathsf{b}(A_k) \in \varphi$ as well as $X_i^{m-1}{\in}\mathsf{b}(A_k) \in \varphi$ for $1 \leq i \leq \ell$. Then we must have $X_{m-1}{\in}\mathsf{b}^-(A_k)$ by closure of φ under (D.lab.dom) and the clash-freeness of φ. Suppose further that $\mathrm{co}(A_k, B_k)(X_{m-1}){=}Y_{m-1}$. Then $Y_{m-1}{\in}\mathsf{b}^-(B_k)$ by the definition of $\mathrm{co}(A_k, B_k)$ and by closure under (P.copy.dom). We must have $\mathrm{co}(A_k, B_k)(X_i^{m-1}){=}Y_i^{m-1}$ for $1 \leq i \leq \ell$ and $Y_{m-1}{:}f(Y_1^{m-1}, \ldots, Y_\ell^{m-1})$ in φ by closure under (P.copy.lab), (D.eq.decom), and (P.path.eq.1), which makes sure that each variable $\in\mathsf{b}(A_k)$ has only one correspondent $\in\mathsf{b}(B_k)$. The constraint φ contains $\mathrm{p}\binom{X_0\ \ Y_0}{X_{m-1}\ Y_{m-1}}$

and $\mathrm{p}(\begin{smallmatrix} X_0 & Y_0 \\ X_i^{m-1} & Y_i^{m-1} \end{smallmatrix})$ for $1 \leq i \leq \ell$ by the definition of $\mathrm{co}(A_k, B_k)$. So by closure of φ under (P.path.eq.1), if $U{=}X_j^{m-1}$, then $\mathrm{p}(\begin{smallmatrix} X & Y \\ X_j^{m-1} & V \end{smallmatrix}) \in \varphi$; by the definition of $\mathrm{co}(A_k, B_k)$, $\mathrm{p}(\begin{smallmatrix} X & Y \\ X_j^{m-1} & Y_j^{m-1} \end{smallmatrix}) \in \varphi$; and by closure under (P.trans.h), $\mathrm{p}(\begin{smallmatrix} Y & Y \\ V & Y_j^{m-1} \end{smallmatrix}) \in \varphi$. Hence by closure under (P.path.eq.2), $V{=}Y_j^{m-1}$ is in φ.

Now suppose $\mathrm{p}(\begin{smallmatrix} X & Y \\ U & V \end{smallmatrix})$ is generated but not correspondence-generated, i.e. there exists an instance ρ of (P.trans.h), (P.trans.v), (P.diff.1) or (P.diff.2) with rhs $\mathrm{p}(\begin{smallmatrix} X & Y \\ U & V \end{smallmatrix})$ such that all path equality literals in the lhs of ρ are generated.

Suppose ρ is an instance of (P.trans.h) and the lhs of ρ is $\mathrm{p}(\begin{smallmatrix} X & Z \\ U & W \end{smallmatrix}) \wedge \mathrm{p}(\begin{smallmatrix} Z & Y \\ W & V \end{smallmatrix})$. If $X{=}U \in \varphi$ then $Z{=}W \in \varphi$ and thus also $Y{=}V \in \varphi$ by the inductive hypothesis. So suppose $X{=}U \notin \varphi$, and suppose we have sequences $X{=}X_0, \ldots, X_{m_1}{=}U$ and $Z{=}Z_0, \ldots, Z_{m_2}{=}W$. Then by the inductive hypothesis, we must have $m_1{=}m_2$.

Now suppose ρ is an instance of (P.diff.2) and the lhs of ρ is $\mathrm{p}(\begin{smallmatrix} X & Y \\ U' & V' \end{smallmatrix}) \wedge \mathrm{p}(\begin{smallmatrix} U & V \\ U' & V' \end{smallmatrix}) \wedge X \triangleleft^* U \wedge Y \triangleleft^* V$. If $X{=}U' \in \varphi$, then $X{=}U \in \varphi$ by (D.dom.trans), and by the inductive hypothesis $Y{=}V'$ is in φ, thus $Y{=}V$ is in φ by (D.dom.trans). If $U{=}U' \in \varphi$, then $V{=}V' \in \varphi$ by the inductive hypothesis, and $\mathrm{p}(\begin{smallmatrix} X & Y \\ U & V \end{smallmatrix}) \in \varphi$ even without application of ρ. Suppose otherwise, and let $X{=}X_0, \ldots, X_{m_1}{=}U'$ and $U{=}U_0, \ldots, U_{m_2}{=}U'$. By closure under (D.lab.dom), (D.dom.trans) and (D.distr.notDisj), there exists a minimal $i \in \{0, \ldots, m_1\}$ with $U_0 \triangleleft^* X_i \in \varphi$. φ is simple, so by (D.distr.child), we must have $X_i{=}U_0 \in \varphi$, i.e. we can choose the sequence $X_0, \ldots, X_{m_1}$ such that it equals $X_0, \ldots, X_{i-1}, U_0, \ldots, U_{m_2}$. But then the inductive hypotheses already holds for $\mathrm{p}(\begin{smallmatrix} X & Y \\ U & V \end{smallmatrix})$ and the sequence $X = X_0, \ldots, X_{i-1}, U_0{=}U$. The cases of ρ being an instance of (P.trans.v) or (P.diff.1) are analogous.

Group parallelism literals. Let $(A_1, \ldots, A_n) \sim (B_1, \ldots, B_n) \in \varphi$. We first consider each pair A_k, B_k on its own (for $1 \leq k \leq n$). Let $A_k = X_0/X_1, \ldots, X_m$ and $B_k = Y_0/Y_1, \ldots, Y_m$. Then $\sigma(X_0) \triangleleft^* \sigma(X_i), \sigma(Y_0) \triangleleft^* \sigma(Y_i)$ hold in (τ, σ) for all $0 \leq i \leq m$ because (τ_{dom}, σ) is a model of φ_{dom}, and $X_0 \triangleleft^* X_i, Y_0 \triangleleft^* Y_i \in \varphi_{dom}$ by closure of φ under (P.init). So $\gamma_1 = \sigma(X_0)/\sigma(X_1), \ldots, \sigma(X_m)$ and $\gamma_2 = \sigma(Y_0)/\sigma(Y_1), \ldots, \sigma(Y_m)$ are segments of θ. We define a function $c^k : \mathsf{b}(\gamma_1) \to \mathsf{b}(\gamma_2)$ by

$$c^k(\sigma(X)){=}\sigma(Y) \text{ iff } X \in \mathsf{b}(A_k), \mathrm{p}(\begin{smallmatrix} X_0 & Y_0 \\ X & Y \end{smallmatrix}) \in \varphi.$$

c^k is well-defined because if $\mathrm{p}(\begin{smallmatrix} X_0 & Y_0 \\ X & Y \end{smallmatrix}), \mathrm{p}(\begin{smallmatrix} X_0 & Y_0 \\ X & Z \end{smallmatrix}) \in \varphi$, then by closure under (P.trans.h), (P.path.symm), (P.path.eq.1) also $Y{=}Z \in \varphi$.

$|\mathbf{hs}(\gamma_1)| = |\mathbf{hs}(\gamma_2)|$ by the definition of group parallelism (and by closure under (P.init)).

c^k's domain is $\mathsf{b}(\gamma_1)$, and its range is $\mathsf{b}(\gamma_2)$: first we show that the domain of c^k is a subset of $\mathsf{b}(\gamma_1)$. Let $X{\in}\mathsf{b}(A_k)$ be in φ. As (τ_{dom}, σ) is a solution of φ_{dom}, $r(\gamma_1) \triangleleft^* \sigma(X)$ and for all $\pi \in hs(\gamma_1)$, either $\sigma(X) \triangleleft^* \pi$ or $\sigma(X) \perp \pi$ must hold in τ_{dom}. So $\sigma(X) \in \mathsf{b}(\gamma_1)$, i.e. for every $X \in \mathcal{V}(\varphi)$ for which $X{\in}\mathsf{b}(A_k)$ is in φ, $\sigma(X)$ is in the domain of c^k.

Conversely, $\mathsf{b}(\gamma_1)$ is a subset of the domain of c^k: let $\pi \in \mathsf{b}(\gamma_1)$. In moving from θ_{dom} to θ, we have added nodes only *above* $\mathsf{b}(\gamma_1)$. So, as noted above for

θ_{dom}, there exists an X with $\sigma(X) = \pi$. We need to show that $X{\in}\mathsf{b}(A_k)$ is in φ. φ possesses a root variable, call it R, and we have $R{\lhd^*}X_0, R{\lhd^*}X$ in φ. Let R' be a $\lhd^+$-maximal variable such that $R'{\lhd^*}X_0, R'{\lhd^*}X \in \varphi$ (i.e. there is no R'' such that $R'{\lhd^+}R''$ and R'' dominates both X_0 and X). If $R'{=}X \in \varphi$, then $X{\lhd^*}X_0$ by closure under (D.dom.trans), and φ must contain $X{=}X_0$ by (P.distr.eq) because $r(\gamma_1){\lhd^*}\pi$. If $R'{=}R'', R'{:}f(Z_1, \ldots, Z_m) \in \varphi$, then we cannot have $Z_i{\lhd^*}X_0, Z_j{\lhd^*}X \in \varphi$ for $1 \leq i \neq j \leq m$, since then $X{\perp}X_0 \in \varphi$ by closure under (D.dom.trans), (D.prop.distr), which would mean $r(\gamma_1){\perp}\pi$ in τ_{dom} because τ_{dom} is a model of φ_{dom}. We cannot have $Z_i{\lhd^*}X_0, Z_i{\lhd^*}X \in \varphi$ for some $i \in \{1, \ldots, m\}$ since we have chosen R' to be maximal. The only remaining possibility is $R'{=}X_0 \in \varphi$ and $Z_i{\lhd^*}X \in \varphi$ for some $i \in \{1, \ldots, m\}$. In any case, $X_0{\lhd^*}X \in \varphi$. By closure under (P.distr.hole), we must have chosen either $X{\lhd^*}Z$ or $X{\perp}Z$ for each hole Z of A_k, otherwise we would have $\sigma(Z){\lhd^+}\pi$ in τ_{dom} for the Z concerned.

By an analogous argument, one can see that the range of c^k is $\mathsf{b}(\gamma_2)$.

$\mathbf{c^k}$ is a bijection: c^k is one-to-one (injective) because if $\mathrm{p}(\begin{smallmatrix} X_0 & Y_0 \\ X & Z \end{smallmatrix}), \mathrm{p}(\begin{smallmatrix} X_0 & Y_0 \\ Y & Z \end{smallmatrix}) \in \varphi$ for $X{\in}\mathsf{b}(A_k), Y{\in}\mathsf{b}(A_k)$ in φ, then $X{=}Y \in \varphi$ by closure under (P.copy.dom). It is onto (surjective) by closure under (P.new).

$\mathbf{c^k}$ is structure-preserving: Suppose $\psi_0 \in \mathsf{b}^-(\gamma_1)$, and $\tau \models \psi_0{:}f(\psi_1, \ldots, \psi_m)$. Then as mentioned above, there exists a $U_0 \in \mathcal{V}(\varphi)$ with $\sigma(U_0) = \psi_0$ and $U_0{\in}\mathsf{b}(A_k)$ in φ. Actually, φ even contains $U_0{\in}\mathsf{b}^-(A_k)$ by the closure of φ under (P.distr.hole) and (P.distr.eq) and the fact that τ_{dom} is a model of φ_{dom}. As φ is simple, U_0 must be labeled: φ must contain $U_0{=}U_0', U_0'{:}g(U_1, \ldots, U_\ell)$ for some $U_0', U_1, \ldots, U_\ell$. In fact, we must have $f = g$, $m = \ell$, and $\sigma(U_i) = \psi_i$ for $1 \leq i \leq m$. As $U_0{\in}\mathsf{b}^-(A_k)$ is in φ, φ also contains $U_i{\in}\mathsf{b}(A_k)$ for $1 \leq i \leq m$. By closure under (P.new), φ contains $\mathrm{p}(\begin{smallmatrix} r(A) & r(B) \\ U_i & V_i \end{smallmatrix})$, $0 \leq i \leq m$, for some $V_0, \ldots, V_m$. By closure under (P.path.eq.1) and (P.copy.lab), it contains $V_0{:}f(V_1, \ldots, V_m)$. By the construction of c^k, we have $c^k(\psi_i) = c(\sigma(U_i)) = \sigma(V_i)$ for $0 \leq i \leq m$, and as τ_{dom} is a model of φ_{dom}, we must have $(\tau_{dom}, \sigma) \models \sigma(V_0){:}f(\sigma(V_1), \ldots, \sigma(V_m)) = c(\psi_0){:}f(c(\psi_1), \ldots, c(\psi_m))$. The opposite direction, starting from $(\tau, \sigma) \models c(\psi_0){:}f(c(\psi_1), \ldots, c(\psi_m))$, is proved by an analogous argument.

The remaining conditions pertain to the interaction of different correspondence functions with respect to binders. For $1 \leq k \leq n$, let $A_k = X_0^k/X_1^k, \ldots, X_{n_k}^k$ and $B_k = Y_0^k/Y_1^k, \ldots, Y_{n_k}^k$, and let $\sigma(A_k) = \gamma_1^k$, $\sigma(B_k) = \gamma_2^k$.

Conditions (same.seg), (diff.seg): Suppose $\psi_1 \in \mathsf{b}^-(\gamma_1^j)$ and $\psi_2 \in \mathsf{b}^-(\gamma_1^k)$ (for $1 \leq j, k \leq n$), and $\lambda(\psi_1) = \psi_2$ in τ. Then, as shown above, there exist $U_1, U_2 \in \mathcal{V}(\varphi)$ with $U_1{\in}\mathsf{b}^-(A_j), U_2{\in}\mathsf{b}^-(A_k)$ in φ such that $\sigma(U_i) = \psi_i$, $i = 1, 2$. By the way we have constructed the λ function within τ, we must have $\lambda(U_1){=}U_2 \in \varphi$. By closure of φ under (P.new) and (P.copy.dom) there exist V_1, V_2 with $V_1{\in}\mathsf{b}^-(B_j)$, $V_2{\in}\mathsf{b}^-(B_k)$ in φ such that $\mathrm{p}(\begin{smallmatrix} X_0^j & Y_0^j \\ U_1 & V_1 \end{smallmatrix})$, $\mathrm{p}(\begin{smallmatrix} X_0^k & Y_0^k \\ U_2 & V_2 \end{smallmatrix}) \in \varphi$. If $j = k$, then by closure of φ under (L.same.seg), we have $\lambda(V_1){=}V_2 \in \varphi$. Now suppose that $\psi_2 \notin \mathsf{b}^-(\gamma_1^j)$. By closure of φ under (L.distr.1), either

$U_2 \in \mathsf{b}^-(A_j)$ is in φ or $U_2 \notin \mathsf{b}^-(A_j)$ is in φ, and as (τ_{dom}, σ) is a solution of φ_{dom}, we must have $U_2 \notin \mathsf{b}^-(A_j)$. Then by closure of φ under (L.diff.seg), we have $\lambda(V_1)=V_2 \in \varphi$. Hence, $\lambda(\sigma(V_1)) = \lambda(c^j(\psi_1)) = \sigma(V_2) = c^k(\lambda(\psi_1))$.
For nodes $\psi_1 \in \mathsf{b}^-(\gamma_2^j)$, $\psi_2 \in \mathsf{b}^-(\gamma_2^k)$ with $\lambda(\psi_1) = \psi_2$, the argument is analogous.

Condition (outside): Let $\psi \in \mathsf{b}^-(\gamma_1^k)$ be in the domain of λ, with $\lambda(\psi) \notin \bigcup_{i=1}^n \mathsf{b}^-(\gamma_1^i)$. Then there exists a U with $U \in \mathsf{b}^-(A_k)$ with $\sigma(U) = \psi$, and by the construction of the function λ there exists some $Z \in \mathcal{V}(\varphi)$ such that $\lambda(U)=Z \in \varphi$, and $\sigma(Z) = \lambda(\psi)$. By closure of φ under (L.distr.1), either $Z \in \mathsf{b}^-(A_j)$ is in φ for some $j \in \{1, \ldots, n\}$, or $Z \notin \mathsf{b}^-(A_j)$ is in φ for all $1 \leq j \leq n$. As (τ_{dom}, σ) is a solution of φ_{dom}, the latter must be the case. Let $c^k(\psi) = \psi'$. By closure of φ under (P.new) and (P.copy.dom) there exists some V with $V \in \mathsf{b}^-(A_k)$ such that $\mathrm{p}(\begin{smallmatrix} X_0^k & Y_0^k \\ U & V \end{smallmatrix}) \in \varphi$, so $\sigma(V) = \psi'$ by the construction of c^k. By closure of φ under (L.outside), we must have $\lambda(V)=Z \in \varphi$, so $\lambda(\psi') = \lambda(c^k(\psi)) = \sigma(Z) = \lambda(\psi)$ by the construction of the function λ.
The case of $\psi \in \mathsf{b}^-(\gamma_2^k)$ is again analogous.

Condition (hang) Let $\psi \in \mathsf{b}^-(\gamma_1^k)$ be in the range of λ. Then there exists an U with $U \in \mathsf{b}^-(A_k)$ with $\sigma(U) = \psi$. For every ψ' with $\lambda(\psi')=\psi'$ there exists a $V \in \mathcal{V}(\varphi)$ with $\sigma(V) = \psi'$. By the construction of the λ function, we must have $\lambda(V)=U \in \varphi$. By closure of φ under (L.hang), there is some $j \in \{1, \ldots, n\}$ such that $V \in \mathsf{b}^-(A_j)$ is in φ. As (τ_{dom}, σ) is a solution of φ_{dom}, we must have $\psi' = \sigma(V) \in \mathsf{b}^-(\gamma_1^j)$.
The case of $\psi \in \mathsf{b}^-(\gamma_2^k)$ is of course analogous.

B.3 Soundness: Non-simple constraints

Now suppose we have a generated non-simple constraint φ in *GP*-solved form. We want to show that there is an *extension* $\varphi \wedge \varphi'$ of φ such that $\varphi \wedge \varphi'$ is in *GP*-solved form as well as simple. We proceed by successively labeling unlabeled variables $X \in \varphi$. Take for instance the constraint in Fig. 15. The main idea is to make all variables minimally dominated by X into X's children, i.e. all variables V with $X \triangleleft^+ V$ such that there is no intervening W with $X \triangleleft^+ W \triangleleft^+ V$. So in the constraint in Fig. 15, Y, Z, U are minimally dominated. However, we choose only one of Z, U as we have $Z{=}U$. As there are two prospective child variables left, we would like to label X by some function symbol of arity 2, extending the constraint, for instance, by $X{:}f(Y, Z)$. If there is no symbol of suitable arity in θ, we can always simulate it by a constant symbol and a symbol of arity ≥ 2. We formalize this as follows: Given a constraint φ we define an ordering $\prec_\varphi$ on its variables such that $X \prec_\varphi Y$ holds iff $X \triangleleft^* Y \in \varphi$ but not $Y \triangleleft^* X \in \varphi$.

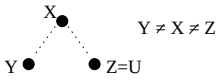


Fig. 15. Extension

Definition 14. Let φ be a dominance constraint and $X \in \mathcal{V}(\varphi)$ unlabeled. Then we define the set $\mathrm{con}_\varphi(X)$ of variables *connected to* X *in* φ as follows:

$$\mathrm{con}_\varphi(X) = \{Y \in \mathcal{V}(\varphi) \mid Y \text{ minimal with } X \prec_\varphi Y\}$$

For the constraint in Fig. 15, $\mathrm{con}_\varphi(X) = \{Y, Z, U\}$.

Definition 15. We call $V \subseteq \mathcal{V}(\varphi)$ a *φ-disjointness* set if for any two distinct variables $Y_1, Y_2 \in V$, $Y_1{=}Y_2 \notin \varphi$.

The idea is that all variables in a φ-disjointness set can safely be placed at disjoint positions in at least one of the trees solving φ. In our example, only one of Z, U can be in any φ- disjointness set.

We have to make sure that we preserve solvedness during extension. For example, when adding $X{:}f(Y, Z)$ to the constraint in Fig. 15, we also add $Y \perp Z$ so as not to make (D.lab.disj) applicable. Specifically, we have to be careful when labeling a variable like X_1 in Fig. 16 (where grey arcs stand for path equality literals): We have $X_1{\in}\mathsf{b}(X_1/X_2)$, and when we add $X_1{:}g(X)$ for some unary g, we also have to add $Y_1{:}g(X')$, otherwise (P.copy.lab) would be applicable. We formalize this in the notion of the *copy set* of a labeling literal $X{:}f(X_1, \ldots, X_n)$.

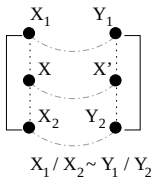


Fig. 16. Extension and parallelism

Definition 16. Let φ be a constraint with X, $X_1, \ldots, X_n$, Y, $Y_1, \ldots, Y_n \in \mathcal{V}(\varphi)$ and let f be a function symbol of arity n. Then we define $\hookrightarrow_\varphi$ by

$$X{:}f(X_1, \ldots, X_n) \hookrightarrow_\varphi Y{:}f(Y_1, \ldots, Y_n)$$

iff there exists some $(A_1, \ldots, A_m) \sim (B_1, \ldots, B_m) \in \varphi$ such that $X{\in}\mathsf{b}^-(A_k)$ and $X_1, \ldots, X_n{\in}\mathsf{b}(A_k)$ are in φ for some $k \in \{1, \ldots, m\}$, and $\mathrm{co}(A_k, B_k)(X){=}Y$, $\mathrm{co}(A_k, B_k)(X_i){=}Y_i$ are in $\varphi$for $1 \leq i \leq n$.

Furthermore,

$$\mathrm{copy}_\varphi\big(X{:}f(X_1, \ldots, X_n)\big) := \{Y{:}f(Y_1, \ldots, Y_n) \mid X{:}f(X_1, \ldots, X_n) \hookrightarrow_\varphi^* Y{:}f(Y_1, \ldots, Y_n)\}$$

where as usual $\hookrightarrow_\varphi^*$ is the reflexive and transitive closure of $\hookrightarrow_\varphi$.

We show some properties of the concepts we have just introduced.

Lemma 17. *Let φ be in GP-solved form, and let $X \in \mathcal{V}(\varphi)$. If V is a maximal φ-disjointness set within $con_\varphi(X)$, then for all $Y \in con_\varphi(X)$ there exists some $Z \in V$ such that $Y{=}Z \in \varphi$.*

Proof. If $Y{=}Z \notin \varphi$ for all $Z \in V$, then $\{Y\} \cup V$ is a disjointness set; thus $Y \in V$ by the maximality of V.

Lemma 18. *Let φ be a constraint in GP-solved form, and let $Y{:}f(Y_1, \ldots, Y_n) \in \mathrm{copy}_\varphi\big(X{:}f(X_1, \ldots, X_n)\big)$.*

- *If X is unlabeled in φ, then so is Y.*
- *If $\{X_1, \ldots, X_n\} \subseteq con_\varphi(X)$, then $\{Y_1, \ldots, Y_n\} \subseteq con_\varphi(Y)$.*

– *If* $\{X_1, \ldots, X_n\}$ *is a maximal* φ*-disjointness set in* $con_\varphi(X)$*, then* $\{Y_1, \ldots, Y_n\}$ *is a maximal* φ*-disjointness set in* $con_\varphi(Y)$*.*

Proof. By well-founded induction on the strict partial order $\subset$ on the set $\{\mathcal{S} \mid \{X{:}f(X_1, \ldots, X_n)\} \subseteq \mathcal{S} \subseteq \mathrm{copy}_\varphi\big(X{:}f(X_1, \ldots, X_n)\big)\}$.

The case of $\mathcal{S} = \{X{:}f(X_1, \ldots, X_n)\}$ is trivial. So suppose $\mathcal{S}$ is not a singleton. Then it has the form $\mathcal{S}' \cup \{Y{:}f(Y_1, \ldots, Y_n)\}$ and there exists $Z{:}f(Z_1, \ldots, Z_n) \in \mathcal{S}'$ with $Z{:}f(Z_1, \ldots, Z_n) \hookrightarrow_\varphi Y{:}f(Y_1, \ldots, Y_n)$ (because $X{:}f(X_1, \ldots, X_n) \in \mathcal{S}$, so if there were no such $Z{:}f(Z_1, \ldots, Z_n) \in \mathcal{S}'$, then $\mathcal{S} \not\subseteq \mathrm{copy}_\varphi\big(X{:}f(X_1, \ldots, X_n)\big)$). Let $(A_1, \ldots, A_m) \sim (B_1, \ldots, B_m) \in \varphi$ and $k \in \{1, \ldots, m\}$ such that $Z{\in}\mathsf{b}^-(A_k)$ and $Z_i{\in}\mathsf{b}(A_k)$ are in φ for $1 \leq i \leq n$. Let $\mathrm{co}(A_k, B_k)(Z){=}Y$ and $\mathrm{co}(A_k, B_k)(Z_i){=}Y_i$ be in φ for $1 \leq i \leq n$. Then $Y{\in}\mathsf{b}^-(A_k)$ and $Y_i{\in}\mathsf{b}(A)_k$ are in φ for $1 \leq i \leq n$ by closure of φ under (P.copy.dom).

– Suppose Z is unlabeled. Then Y must be unlabeled too, as any labeling literal would have been copied by (P.copy.lab).
– Suppose $\{Z_1, \ldots, Z_n\} \subseteq \mathrm{con}_\varphi(Z)$. Then by closure under (P.copy.dom), $Y \lhd^* Y_i \in \varphi$ but $Y_i \lhd^* Y \notin \varphi$ for $1 \leq i \leq n$. Assume that Y_i is not minimal with $Y \prec_\varphi Y_i$, i.e. there exists some W with $Y \prec_\varphi W \prec_\varphi Y_i$. Then $W{\in}\mathsf{b}(B_k)$ is in φ by closure under (D.dom.trans), (D.prop.disj), (P.distr.hole). So by (P.new), there exists some W' with $W'{\in}\mathsf{b}(A_k)$ and $\mathrm{co}(A_k, B_k)(W'){=}W$. But then $Z \lhd^* W' \lhd^* Z_i \in \varphi$ by (P.copy.dom), but neither $W' \lhd^* Z$ nor $Z_i \lhd^* W'$ is in φ, so Z_i is not minimal either, a contradiction.
– Suppose $\{Z_1, \ldots, Z_n\}$ is a maximal φ-disjointness set in $\mathrm{con}_\varphi(Z)$. Assume that $\{Y_i, Y_j\}$ is not a disjointness set for some $1 \leq i \neq j \leq n$. So $Y_i{=}Y_j \in \varphi$. But then by (P.copy.dom), $Z_i{=}Z_j \in \varphi$, a contradiction.
  Assume $\{Y_1, \ldots, Y_n\}$ is not maximal, i.e. there exists some $Y' \notin \{Y_1, \ldots, Y_n\}$ such that $\{Y_1, \ldots, Y_n, Y'\} \subseteq \mathrm{con}_\varphi(Y)$ is a disjointness set. Let $A_k = U_0/U_1, \ldots, U_\ell$ and $B_k = V_0/V_1, \ldots, V_\ell$. We must have $V_0 \lhd^* Y'$ by (D.dom.trans) and either $Y' \lhd^* V_i$ or $Y' \perp V_i$ or $V_i \lhd^+ Y'$ is in φ for all $1 \leq i \leq \ell$ by (P.distr.hole). But if $V_i \lhd^+ Y'$ for some $i \in \{1, \ldots, \ell\}$, then $Y' \notin \mathrm{con}_\varphi(Y)$ because $Y{=}V_i \notin \varphi$. So $Y'{\in}\mathsf{b}(B_k)$ is in φ. By closure under (P.new) and (P.copy.dom), there exists a Z' with $Z'{\in}\mathsf{b}(A_k)$ such that $\mathrm{co}(A_k, B_k)(Z'){=}Y'$ is in φ. By closure under (P.copy.dom), we have $Z' \in \mathrm{con}_\varphi(Z)$. Z' cannot be in $\{Z_1, \ldots, Z_n\}$: If $Z'{=}Z_i \in \varphi$ for some $i \in \{1, \ldots, n\}$, then $\mathrm{p}(\begin{smallmatrix} U_0 & V_0 \\ Z_i & Y' \end{smallmatrix}), \mathrm{p}(\begin{smallmatrix} U_0 & V_0 \\ Z_i & Y_i \end{smallmatrix}) \in \varphi$ by (P.path.eq.1), so $Y'{=}Y_i \in \varphi$ by (P.path.eq.2). Hence, $\{Z_1, \ldots, Z_n, Z'\}$ is a φ-disjointness set in $\mathrm{con}_\varphi(Z)$ that is bigger than $\{Z_1, \ldots, Z_n\}$, a contradiction.

Now we have all the necessary tools to state the main lemma of our soundness proof. It describes one extension step in which a previously unlabeled variable is labeled.

Proposition 19. *Every GP-solved form* φ *with an unlabeled variable* X *can be extended to a GP-solved form in which* X *is labeled.*

Proof. Let $\{X_1, \ldots, X_n\}$ be a maximal φ-disjointness set in $\mathrm{con}_\varphi(X)$. Let f be a function symbol in Σ of arity n.[1] W.l.o.g. we assume that Σ contains at least one constant apart from var. Then we can assume that we do not add a function symbol lam or var while extending φ. Now we define the extension $\mathsf{ext}(\varphi)$ of $\varphi \wedge X{:}f(X_1, \ldots, X_n)$ as

$$\mathsf{ext}(\varphi) := \varphi \wedge \bigwedge_{\substack{Y:f(Y_1,\ldots,Y_n)\in \\ \mathrm{copy}_\varphi\left(X:f(X_1,\ldots,X_n)\right)}} \Big(Y{:}f(Y_1, \ldots, Y_n) \wedge \textstyle\bigwedge_{i=1}^{n} Y{\neq}Y_i \wedge \bigwedge_{\substack{Y_i \lhd^* U, Y_j \lhd^* V \in \varphi, \\ 1 \leq i \neq j \leq n}} U{\perp}V \wedge \bigwedge_{\substack{Z:g(\ldots)\in\varphi, \\ g\neq f \ \vee\ \mathsf{ar}(g)\neq\mathsf{ar}(f)}} Z{\neq}Y\Big)$$

Note that X is labeled in $\mathsf{ext}(\varphi)$ since $X{=}X \in \varphi$ by (D.dom.refl). We consider each rule of D in turn and show that it is not applicable to $\mathsf{ext}(\varphi)$.

(D.clash.ineq): $\mathsf{ext}(\varphi)$ contains no new dominance literals. If a new inequality literal $Y{\neq}Y_i$ were to make (D.clash.ineq) applicable, then φ must contain $Y{=}Y_i$, but $Y{:}f(Y_1, \ldots, Y_n) \in \mathrm{copy}_\varphi\big(X{:}f(X_1, \ldots, X_n)\big)$, so $Y_i \in \mathrm{con}_\varphi(Y)$ by Lemma 18.
If a new inequality $Z{\neq}Y$ were to make the clash rule applicable, then $Z{:}g(\ldots)$ and $Y{=}Z$ must be in φ, but by lemma 18, Y is unlabeled because X is.

(D.clash.disj): The only new disjointness literals in $\mathsf{ext}(\varphi)$ have the form $U{\perp}V$ for $Y_i{\lhd^*}U, Y_j{\lhd^*}V$ in φ with $i \neq j$. Assume $U{=}V$ is in φ. Then by (D.distr.notDisj), either $Y_i{\lhd^*}Y_j$ or $Y_j{\lhd^*}Y_i$ must be in φ. But $\{X_i, X_j\}$ is a disjointness set, and so, by Lemma 18, is $\{Y_i, Y_j\}$.

(D.dom.refl): No new variables have been added.

(D.dom.trans), (D.distr.notDisj): No new dominance literals have been added.

(D.eq.decom): Suppose $Y{:}f(Y_1, \ldots, Y_n) \in \mathrm{copy}_\varphi\big(X{:}f(X_1, \ldots, X_n)\big)$ and $Y{=}Z$ is in φ. Then Y and Z must be unlabeled by Lemma 18, so for (D.eq.decom) to be applicable, both $Y{:}f(Y_1, \ldots, Y_n)$ and $Z{:}f(Z_1, \ldots, Z_n)$ must be in $\mathsf{ext}(\varphi) - \varphi$, which means that $Z{:}f(Z_1, \ldots, Z_n) \in \mathrm{copy}_\varphi\big(X{:}f(X_1, \ldots, X_n)\big)$, too.
If $\mathrm{copy}_\varphi\big(X{:}f(X_1, \ldots, X_n)\big)$ is a singleton, then we must have $X_i{=}Y_i{=}Z_i$ for $1 \leq i \leq n$. So suppose otherwise. We first prove the following auxiliary claim:
> Let $U{:}f(U_1, \ldots, U_n) \in \mathrm{copy}_\varphi\big(X{:}f(X_1, \ldots, X_n)\big)$. Then $\mathrm{p}\left(\begin{smallmatrix} X & U \\ X_i & U_i \end{smallmatrix}\right) \in \varphi$ for $1 \leq i \leq n$.

We use induction on the length of a $\hookrightarrow_\varphi$ sequence starting in $X{:}f(X_1, \ldots, X_n)$ and ending in $U{:}f(U_1, \ldots, U_n)$, and we start with a sequence of length 0. As $\mathrm{copy}_\varphi\big(X{:}f(X_1, \ldots, X_n)\big)$ is not a singleton, there exists some $(A_1, \ldots, A_m) \sim (B_1, \ldots, B_m) \in \varphi$ and some $k \in \{1, \ldots, m\}$ such that $X{\in}\mathsf{b}(A_k)$ and $X_i{\in}\mathsf{b}(A_k)$ are in φ for $1 \leq i \leq n$. Let $A_k = W_0/W_1, \ldots, W_\ell$ and $B_k = W'_0/W'_1, \ldots, W'_\ell$. For $1 \leq i \leq n$, the following holds: by closure under (P.new), there exist X', X'_i such that $\mathrm{p}\left(\begin{smallmatrix} W_0 & W'_0 \\ X & X' \end{smallmatrix}\right) \in \varphi$

[1] If there exists no suitable f, it is simulated using other function symbols.

as well as $\mathrm{p}(\begin{smallmatrix} W_0 & W_0' \\ X_i & X_i' \end{smallmatrix}) \in \varphi$; by (P.path.symm), $\mathrm{p}(\begin{smallmatrix} W_0' & W_0 \\ X' & X \end{smallmatrix}), \mathrm{p}(\begin{smallmatrix} W_0' & W_0 \\ X_i' & X_i \end{smallmatrix}) \in \varphi$, so by (P.trans.h), $\mathrm{p}(\begin{smallmatrix} W_0 & W_0 \\ X & X \end{smallmatrix}), \mathrm{p}(\begin{smallmatrix} W_0 & W_0 \\ X_i & X_i \end{smallmatrix}) \in \varphi$; as $X \triangleleft^* X_i \in \varphi$, closure under (P.diff.1) yields $\mathrm{p}(\begin{smallmatrix} X & X \\ X_i & X_i \end{smallmatrix}) \in \varphi$.

Suppose $V{:}f(V_1, \ldots, V_n) \in \mathrm{copy}_\varphi\big(X{:}f(X_1, \ldots, X_n)\big)$ with $\mathrm{p}(\begin{smallmatrix} X & V \\ X_i & V_i \end{smallmatrix}) \in \varphi$ for $1 \leq i \leq n$, and $V{:}f(V_1, \ldots, V_n) \hookrightarrow_\varphi U{:}f(U_1, \ldots, U_n)$. Then φ contains some $(A_1, \ldots, A_m) \sim (B_1, \ldots, B_m)$ and there exists some $k \imath n \{1, \ldots, m\}$ such that $V \in \mathsf{b}(B_k)$ and $V_i \in \mathsf{b}(B_k)$ for $1 \leq i \leq n$ and $\mathrm{co}(A_k, B_k)(V){=}U$, $\mathrm{co}(A_k, B_k)(V_i){=}U_i$ for $1 \leq i \leq n$. Then by closure of φ under (P.diff.1), $\mathrm{p}(\begin{smallmatrix} V & U \\ V_i & U_i \end{smallmatrix}) \in \varphi$ for $1 \leq i \leq n$, and so, by (P.trans.H), is $\mathrm{p}(\begin{smallmatrix} X & U \\ X_i & U_i \end{smallmatrix})$. This concludes the proof of the auxiliary claim.

By the auxiliary claim, $\mathrm{p}(\begin{smallmatrix} X & Y \\ X_i & Y_i \end{smallmatrix}), \mathrm{p}(\begin{smallmatrix} X & Z \\ X_i & Z_i \end{smallmatrix}) \in \varphi$. By closure under (P.path.symm) and (P.trans.h), φ contains $\mathrm{p}(\begin{smallmatrix} Y & Z \\ Y_i & Z_i \end{smallmatrix})$, and as $Y{=}Z \in \varphi$, $\mathrm{p}(\begin{smallmatrix} Z & Z \\ Y_i & Z_i \end{smallmatrix}) \in \varphi$ by (P.path.eq.1). Thus by (P.path.eq.2), $Y_i{=}Z_i \in \varphi$ already (all for $1 \leq i \leq n$).

(D.lab.ineq): Suppose $Y{:}f(Y_1, \ldots, Y_n) \in \mathsf{ext}(\varphi) - \varphi$. Then $Z{\neq}Y$ is in $\mathsf{ext}(\varphi)$ by definition for all Z labeled anything but f.

(D.lab.disj): Suppose $Y{:}f(Y_1, \ldots, Y_n) \in \mathsf{ext}(\varphi) - \varphi$. Since $Y_i \triangleleft^* Y_i, Y_j \triangleleft^* Y_j \in \varphi$ for $1 \leq i \leq n$ by closure under (D.dom.refl), $Y_i \perp Y_j$ is in $\mathsf{ext}(\varphi)$ by definition.

(D.prop.disj): Suppose $Y{:}f(Y_1, \ldots, Y_n) \in \mathrm{copy}_\varphi\big(X{:}f(X_1, \ldots, X_n)\big)$ and $U \perp V \in \mathsf{ext}(\varphi) - \varphi$ for some $Y_i \triangleleft^* U$, $Y_j \triangleleft^* V$, $j \neq i$. If $U \triangleleft^* U'$ and $V \triangleleft^* V'$ are in φ, then we also have $Y_i \triangleleft^* U', Y_j \triangleleft^* V' \in \varphi$ by closure under (D.dom.trans), so $U' \perp V'$ is in $\mathsf{ext}(\varphi)$.

(D.lab.dom): Suppose $Y{:}f(Y_1, \ldots, Y_n) \in \mathsf{ext}(\varphi) - \varphi$. We have $Y \triangleleft^* Y_i \in \varphi$ by Lemma 18. $Y{\neq}Y_i \in \mathsf{ext}(\varphi)$ by definition.

(D.distr.child): Suppose $Y{:}f(Y_1, \ldots, Y_n) \in \mathsf{ext}(\varphi) - \varphi$ and $Y \triangleleft^* Z \in \varphi$.

If $Z \triangleleft^* Y \in \varphi$, then (D.distr.child) is not applicable in $\mathsf{ext}(\varphi)$. Otherwise $Y \prec_\varphi Z$. If Z is minimal with $Y \prec_\varphi Z$, then $Z \in \mathrm{con}_\varphi(Y)$, and as $\{Y_1, \ldots, Y_n\}$ is a maximal φ-disjointness set in $\mathrm{con}_\varphi(Y)$, we have $Z{=}Y_i \in \varphi$ for some $i \in \{1, \ldots, n\}$. If Z is not minimal, there exists some $Y' \in \mathrm{con}_\varphi(Y)$ such that $Y' \triangleleft^* Z$ is in φ. But then again, $Y_i{=}Y'$ for some $i \in \{1, \ldots, n\}$, so $Y_i \triangleleft^* Z$.

(B.. . .) As mentioned above, we assume w.l.o.g. that we have not added any var or lam labels. Also, we have not added any binding literals or λ^{-1} literals. Thus, none of these rules are applicable.

(P.symm), (P.init), (P.path.symm), (P.path.dom), (P.path.eq.1), (P.path.eq.2), (P.distr.hole): No new dominance, parallelism, or path equality literals have been added.

(P.new): We have not added any new group parallelism or dominance literals. If there is some $(A_1, \ldots, A_n) \sim (B_1, \ldots, B_n) \in \varphi$ such that $A_k = X/\ldots$ for some $k \in \{1, \ldots, n\}$ and $X \triangleleft^* U \in \varphi$, then by the closure of φ under (P.distr.hole) we have already decided whether $U \in \mathsf{b}(A_k)$ or not.

(P.copy.dom): Any dominance literal in $\mathsf{ext}(\varphi)$ is in φ already, so the case of $R = \triangleleft^*$ does not apply.

- We consider the case of R being $\perp$. Let $U \perp V$ be in $\mathsf{ext}(\varphi) - \varphi$, where for some $Y{:}f(Y_1, \ldots, Y_n) \in \mathrm{copy}_\varphi\big(X{:}f(X_1, \ldots, X_n)\big)$ and some $1 \leq i \neq$

$j \leq n$, $Y_i \lhd^* U, Y_j \lhd^* V \in \varphi$. (This means that $\{Y_1, \ldots, Y_n\} \neq \emptyset$.) Suppose φ contains a group parallelism literal $(A_1, \ldots, A_m) \sim (B_1, \ldots, B_m)$ such that $U \in \mathsf{b}(A_k), V \in \mathsf{b}(A_k)$ are in φ for some $k \in \{1, \ldots, m\}$. Let $A_k = W_0/W_1, \ldots, W_\ell$ and $B_k = W_0'/W_1', \ldots, W_\ell'$. We have $W_0 \lhd^* U \in \varphi$ by the definition of $U \in \mathsf{b}(A_k)$, and by closure under (D.dom.trans), $Y \lhd^* U \in \varphi$. Hence by (D.distr.notDisj), φ contains either $Y \lhd^* W_0$ or $W_0 \lhd^* Y$.

If φ contains $Y \lhd^* W_0$ but not $Y = W_0$, then $Y \prec_\varphi W_0$. $\{Y_1, \ldots, Y_n\}$ is a maximal φ-disjointness set in $\mathrm{con}_\varphi(Y)$ by Lemma 18. So if $W_0 \in \mathrm{con}_\varphi(Y)$, then by Lemma 17, $W_0 = Y_j$ is in φ for some $j \in \{1, \ldots, n\}$. If W_0 is not minimal with $Y \prec_\varphi W_0$, then there exists some $Y' \in \mathrm{con}_\varphi(Y)$ such that $Y' \prec_\varphi W_0$. Again by Lemma 17, φ contains $Y' = Y_j$ for some $j \in \{1, \ldots, n\}$ and hence, by closure under (D.dom.trans), $Y_k \lhd^* W_0 \in \varphi$. But then we cannot have both $W_0 \lhd^* U$ and $W_0 \lhd^* V$ in φ since at least one of $Y_i \perp Y_k$ and $Y_j \perp Y_k$ is in φ, and φ is clash-free. So (D.distr.notDisj) must have made the choice $W_0 \lhd^* Y \in \varphi$.

φ is closed under (P.distr.hole), but the choice made cannot be $W_i \lhd^* Y$ for any $i \in \{1, \ldots, \ell\}$, since $Y \lhd^+ U, Y \lhd^+ V \in \varphi$ by closure under (D.dom.trans), (D.lab.dom), (P.distr.eq) and on the other hand $U \in \mathsf{b}(A_k)$ and $V \in \mathsf{b}(A_k)$. So for all $1 \leq i \leq \ell$, either $Y \lhd^+ W_i \in \varphi$ by (P.distr.hole) and (P.distr.eq), or $Y \perp W_i \in \varphi$ by (P.distr.hole). In the first case, (P.distr.hole) must have chosen either $Y_j \perp W_i$ or $Y_j \lhd^* W_i$ for each $1 \leq j \leq n$ because all the Y_j are minimal with $Y \prec_\varphi Y_j$. In the second case, we have $Y_j \perp W_i \in \varphi$ for $1 \leq j \leq n$ by closure under (D.prop.disj). In both cases, $Y \in \mathsf{b}^-(A_k)$ and $Y_i \in \mathsf{b}(A_k)$ are in φ for $1 \leq i \leq n$. By closure of φ under (P.new), there are $Z, Z_1, \ldots, Z_n$ such that $\mathrm{p}\left(\begin{smallmatrix} W_0 & W_0' \\ Y & Z \end{smallmatrix}\right) \in \varphi$ and $\mathrm{p}\left(\begin{smallmatrix} W_0 & W_0' \\ Y_i & Z_i \end{smallmatrix}\right) \in \varphi$ for $1 \leq i \leq n$. Since $Y \in \mathsf{b}^-(A_k)$, $Z{:}f(Z_1, \ldots, Z_n) \in \mathrm{copy}_\varphi\big(X{:}f(X_1, \ldots, X_n)\big)$. By closure under (P.new), there exist U', V' such that $\mathrm{co}(A_k, B_k)(U) = U', \mathrm{co}(A_k, B_k)(V) = V' \in \varphi$. And by closure under (P.copy.dom), $Z_i \lhd^* U', Z_j \lhd^* V' \in \varphi$, so $U' \perp V' \in \mathsf{ext}(\varphi)$ by definition.

– It remains to consider the case of R being $\neq$. Let $Y{:}f(Y_1, \ldots, Y_n) \in \mathrm{copy}_\varphi\big(X{:}f(X_1, \ldots, X_n)\big)$.

Suppose $Y \neq Y_i \in \mathsf{ext}(\varphi) - \varphi$ for some $i \in \{1, \ldots, n\}$. (Again, $\{Y_1, \ldots, Y_n\} \neq \emptyset$.) Suppose further that $(A_1, \ldots, A_m) \sim (B_1, \ldots, B_m) \in \varphi$ with $Y \in \mathsf{b}(A_k)$ and $Y_i \in \mathsf{b}(A_k)$ for some $k \in \{1, \ldots, m\}$. By closure under (P.new), there exist Z, Z_i such that $\mathrm{co}(A_k, B_k)(Y) = Z, \mathrm{co}(A_k, B_k)(Y_i) = Z_i \in \varphi$.

We must have $Y \in \mathsf{b}^-(A_k)$ by closure under (P.distr.hole), (P.distr.eq) and the fact that $Y_i \in \mathsf{b}(A_k)$. So $Y_j \in \mathsf{b}(A_k)$ for all $1 \leq j \leq n$ by closure under (P.distr.hole).

So there are $Z_1, \ldots, Z_n$ such that $\mathrm{co}(A_k, B_k)(Y_j) = Z_j \in \varphi$ for $1 \leq j \leq n$. $Y \in \mathsf{b}^-(A_k)$, so $Z{:}f(Z_1, \ldots, Z_n) \in \mathrm{copy}_\varphi\big(X{:}f(X_1, \ldots, X_n)\big)$. Hence, $Z \neq Z_i$ is in $\mathsf{ext}(\varphi)$ by definition.

Now suppose $Z \neq Y \in \mathsf{ext}(\varphi) - \varphi$, where $Z{:}g(\ldots)$ is in φ for some g with either $g \neq f$ or $\mathsf{ar}(g) \neq \mathsf{ar}(f)$. Suppose further that $(A_1, \ldots, A_m) \sim$

$(B_1, \ldots, B_m) \in \varphi$ with $Y{\in}\mathsf{b}(A_k), Z{\in}\mathsf{b}(A_k)$ in φ for some $k \in \{1, \ldots, m\}$. By closure under (P.distr.eq), we have either $Z{=}Y \in \varphi$ or $Z{\neq}Y \in \varphi$. $Z{=}Y \in \varphi$ is impossible since Y is unlabeled by Lemma 18. So $Z{\neq}Y$ must be in φ already.

(P.copy.lab): Let $Y{:}f(Y_1, \ldots, Y_n) \in \mathrm{copy}_\varphi\big(X{:}f(X_1, \ldots, X_n)\big)$ with $Y{:}f(Y_1, \ldots, Y_n) \in \mathsf{ext}(\varphi) - \varphi$. Suppose $(A_1, \ldots, A_m) \sim (B_1, \ldots, B_m) \in \varphi$ and suppose there exists some $k \in \{1, \ldots, m\}$ with $Y{\in}\mathsf{b}(A_k)$ and $Y_i{\in}\mathsf{b}(A_k)$ in φ for $1 \leq i \leq n$. Then there exist $Z, Z_1, \ldots, Z_n$ such that $\mathrm{co}(A_k, B_k)(Y){=}Z \in \varphi$ and $\mathrm{co}(A_k, B_k)(Y_i){=}Z_i \in \varphi$ for $1 \leq i \leq n$.
By closure under (P.distr.eq), either $Y{\in}\mathsf{b}^-(A_k)$ or there exists some hole W of A_k such that $Y{=}W \in \varphi$. In the first case, $Z{:}f(Z_1, \ldots, Z_n) \in \mathrm{copy}_\varphi\big(X{:}f(X_1, \ldots, X_n)\big)$, so the labeling literal $Z{:}f(Z_1, \ldots Z_n)$ is in $\mathsf{ext}(\varphi)$. If $Y{=}W \in \varphi$ for some hole W of A_k, then (P.copy.lab) is not applicable since it does not copy the label of the exception.

(P.distr.eq): No new variables have been added.

(P.trans.h), (P.trans.v), (P.diff.1), (P.diff.2): Now new path equalities have been added.

(L.distr.1), (L.distr.2): We have not introduced any new group parallelism literals or lambda binding literals. Also, we have not introduced new dominance literals. So if we have a segment term $A = X/\ldots$ such that $X{\triangleleft^*}U \in \varphi$ already, then by the closure of φ under (P.distr.hole) and (P.distr.eq) it is already decided in φ whether $U{\in}\mathsf{b}^-(A)$ or not.

(L.same.seg), (L.diff.seg), (L.outside): We have not added any lambda binding literals. It remains to show that we have not recently acquired new information on whether a binder or a bound variable is situated in a group. Suppose $(A_1, \ldots, A_m) \sim (B_1, \ldots, B_m) \in \varphi$, $\lambda(U){=}Y \in \varphi$, and $U{\in}\mathsf{b}^-(A_k)$ in $\mathsf{ext}(\varphi)$. Then $U{\in}\mathsf{b}^-(A_k)$ is in φ already, as pointed out above, and by closure of φ under (L.distr.1), there either exists a $\ell \in \{1, \ldots, m\}$ such that $Y{\in}\mathsf{b}^-(A_\ell)$, or $Y{\notin}\mathsf{b}^-(A_\ell)$ for all $1 \leq \ell \leq m$. So (L.diff.seg), or (L.outside), has been applied to U and Y in φ already.

(L.hang): We have not added any new parallelism literals, group parallelism literals or lambda binding literals. Also, we have not introduced new dominance literals. So if there is a segment term $A = X/\ldots$ with $XA{\triangleleft^*}U_2 \in \varphi$, then by closure under (P.distr.hole) and (P.distr.eq) it is already decided in φ whether $U_2{\in}\mathsf{b}^-(A)$ or not.

(L.inverse): We have not added any λ^{-1}, group parallelism, dominance, or path equality literals, and, as pointed out before, if $X{\in}\mathsf{b}^-(A)$ in $\mathsf{ext}(\varphi)$, then $X{\in}\mathsf{b}^-(A)$ is in φ already.
We turn to $\mathrm{co}^-(\overline{A}, \overline{B})(S_1)$. For each $Z \in S_1$ and each $1 \leq i \leq |\overline{A}|$, (L.distr.2) has decided whether $Z{\in}\mathsf{b}^-(A_i)$ or $Z{\notin}\mathsf{b}^-(A_i)$. If $X{\in}\mathsf{b}^-(A_i)$, then its correspondent is already known in φ by closure under (P.new).
For each $V \in S_2 \cup S_3$, there exists a literal $\lambda(V){=}U$ for some $U \in \mathcal{V}(\varphi)$, either by (L.same.seg) or by (L.diff.seg). This holds because $Y{\in}\mathsf{b}^-(A_k)$ for some $1 \leq i \leq |\overline{A}|$ and because φ is closed under (L.distr.1). By closure of φ under (P.distr.eq), either $U{=}Y$ or $U{\neq}Y \in \varphi$, so the question of whether $V \in S_2$ or $V \in S_3$ is settled in φ already.

Lemma 20. *Every generated GP-solved form can be extended to a simple generated GP-solved form.*

Proof. The construction in lemma 19 preserves generatedness, as it does not add any new path equalities.

Theorem 21 (Soundness). *A generated constraint in GP-solved form is satisfiable.*

Proof. By lemmas 13 and 20.

B.4 Completeness

[9] introduces a sub-procedure of *GP* and shows that it is complete for *parallelism constraints*, which are like group parallelism constraints except that they only have groups of size one and do not comprise binding constraints. Completeness means that the procedure computes, for every solution (τ, σ) of a constraint φ, a solved form from which a sub-λ-structure of τ can be read off. The only difficulty in the completeness proof are the rules that introduce fresh variables. However, the only rule in *GP* that is not present in [9] and introduces fresh variables is (B.lam). It can only introduce one fresh variable per lam-labeled variable. So the proof from [9] directly carries over to group parallelism constraints and the procedure *GP*.

Theorem 22 (Completeness). *For every solution (τ, σ) of φ, GP computes a GP-solved form of φ of which (τ, σ) is a solution.*