Parallelism and Tree Regular Constraints

Joachim Niehren, Mateu Villaret

Parallelism and Tree Regular Constraints

Joachim Niehren[1] and Mateu Villaret[2⋆]

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

Abstract. Parallelism constraints are logical descriptions of trees. Parallelism constraints subsume dominance constraints and are equal in expressive power to context unification. Parallelism constraints belong to the constraint language for lambda structures (CLLS) which serves for modeling natural language semantics. In this paper, we investigate the extension of parallelism constraints by tree regular constraints. This canonical extension is subsumed by the monadic second-order logic over parallelism constraints. We analyze the precise expressiveness of this extension on basis of a new relationship between tree automata and logic. Our result is relevant for classifying different extensions of parallelism constraints, as in CLLS. Finally, we prove that parallelism constraints and context unification remain equivalent when extended with tree regular constraints.

Keywords: computational logic, tree automata, unification.

1 Introduction

Parallelism constraints [11, 13] are logical descriptions of trees, i.e., of ground terms such as $f(f(a,b),a)$. Parallelism constraints constitute a purely conjunctive language. They can talk about the mother, ancestor, and parallelism relation of a tree:

1. The parallelism relation $\pi_1/\pi_2{\sim}\pi_3/\pi_4$ holds for nodes π_1, ..., π_4 of some tree (see Fig. 1) if π_1 is an ancestor of π_2, π_3 an ancestor of π_4, while tree segment between π_1 and π_2 is structurally equal to the segment between π_3 and π_4.
2. The labeling relation $\pi{:}f(\pi_1,\ldots,\pi_n)$ requires that node π is labeled with f and has the children π_1, ..., π_n in this order.

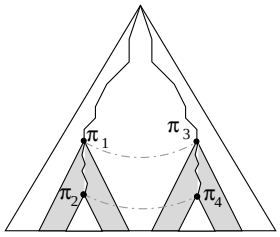


Fig. 1. Parallelism

Parallelism constraints subsume *dominance constraints* [18, 2] for which efficient satisfiability tests exists [1]. Dominance constraints are widely used throughout computational linguistics (see e.g. [21, 10]). They can express the ancestor relation $\pi_1{\lhd}^*\pi_2$ between nodes π_1 and π_2 of some tree (which is equivalent to $\pi_1/\pi_2{\sim}\pi_1/\pi_2$).

Parallelism constraints are equal in expressive power to the equational language of *context unification* (CU) [5, 22, 23, 17] as proved in [19]. Whether CU is decidable

⋆ This work has been partially supported by the SFB 378 of the DFG and the CICYT projects DENOC (BFM2000-1054-C02) and CADVIAL (TIC2001-2392-C03-01).

(and thus the satisfiability of parallelism constraints) is a prominent open problem in unification theory. So far, only fragments could be proved decidable.

Parallelism constraints belong to CLLS – the constraint language for lambda structures which serves for modeling natural language semantics [11, 12]. CLLS extends parallelism constraints in several directions: there are lambda binding and beta reduction constraints [3, 4], but also anaphoric binding and group parallelism constraints. These extensions are used in applications but their expressiveness has never been studied.

In this paper, we investigate the canonical extension P+R of parallelism with tree regular constraints. The formulas of P+R are conjunctions of parallelism constraints P with regular constraints R. Let $\mathcal{A}$ be a tree automaton:

– A tree regular restriction $\mathtt{tree}(\pi)\in\mathcal{L}(\mathcal{A})$ is valid in a tree with node π if the subtree rooted by π belongs to the language recognized by tree automaton $\mathcal{A}$.

The extended language P+R is sufficiently restricted so that processing methods for pure parallelism constraints still apply: Given an extended constraint of P+R, we can first enumerate the *minimal solved forms* (i.e., the most general unifiers) of the pure parallelism part by saturation [13] and then test all minimal solved forms for compatibility with the tree regular part. We obtain a semi-decision procedure for P+R; even if we could decide the satisfiability of parallelism constraints we might still have to check infinitely many minimal solved forms for compatibility.

The language P+R is obviously subsumed by the monadic second-order logic over parallelism constraints. But it is less clear to which precise logical fragment P+R corresponds. This is the question, we will answer in this paper.

The basic idea is to exploit the classical relationship between tree automata and the weak monadic second-order logic of the binary tree (WS2S) [25, 9], which states that regular constraints R and formulas of WS2S have the same expressive power. But unfortunately, this result cannot be lifted to extensions of parallelism constraints as the languages P+R and WS2S have different models. We propose to consider the monadic second-order logic over dominance constraints (SDom) instead of WS2S. Both languages talk about the ancestor relation of trees. But WS2S is interpreted over the infinite binary tree while SDom models ground terms.

We establish a new relationship between tree automata and the logic SDom on basis of the old techniques for WS2S: We show that tree regular constraints R are equal in expressiveness to formulas of SDom (Section 3). We then lift our new result to the respective extensions of parallelism constraints: We prove that the languages P+R and P+SDom can be inter-translated (Section 4). This answers the question raised above. Our result also shows that the languages SDom and WS2S have equal expressiveness. We thereby generalize and complement an earlier insight of Rogers [2, 15] who noticed that the first-order theory of dominance constraints can be expressed in WS2S.

Finally, we reconsider the relationship between CU and parallelism constraints (which have the same expressiveness [19]). We show that P+R is equal in expressive power to CU with tree regular constraints (Section 5). This language in turn is equivalent to linear second order unification (LSOU) with tree regular constraints [16]. It is open whether CU+R is decidable (even if we freely assume that CU is decidable). But the situation is better for the special case of string unification with regular constraints [24] which can be decided in PSPACE [8].

Our contributions are relevant for classifying extensions of parallelism constraint, as for instance provided by CLLS. A forthcoming paper [20] proves, for instance, that the monadic second-order dominance logic SDom can express lambda binding constraints. The results of this paper thus imply that the extension of parallelism with lambda binding constraints (as provided by CLLS) can be expressed in P+R and CU+R.

2 Parallelism Constraints

We assume a finite *signature* Σ of function symbols ranged over by f, g. Each function symbol comes with an arity $\mathsf{ar}(f) \geq 0$. We assume at least one constant $a \in \Sigma$, i.e. a function symbol of arity 0 and at least one binary function symbol.

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

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

The empty word ϵ is called the *root* of the tree, while $i\pi$ is node π of the i-th subtree. We freely identify a tree τ with the function $\tau : \mathsf{nodes}(\tau) \to \Sigma$ that maps every node of τ to its node label. For a tree τ equal to $f(\tau_1, \ldots, \tau_n)$ we set:

$$\tau(\pi) = f(\tau_1, \ldots, \tau_n)(\pi) = \begin{cases} f & \text{if } \pi = \epsilon \\ \tau_i(\pi') & \text{if } \pi = i\pi',\ 1 \leq i \leq n \end{cases}$$

If τ is a tree with node π then we write $\tau.\pi$ for the subtree of τ rooted by π. Furthermore, we write $\tau[\pi/\tau']$ for the tree obtained by replacing the subtree of τ at node π by τ'.

Let τ be a tree with nodes $\pi, \pi', \pi_1, \ldots, \pi_n$. The *labeling relation* $\pi{:}f(\pi_1, \ldots, \pi_n)$ holds in τ if π is labeled by f in τ and has the sequence of children $\pi_1, \ldots, \pi_n$ in that order from the left to the right. This is if $\tau(\pi) = f$ and $\pi_1 = \pi 1$, …, $\pi_n = \pi n$ where $n = \mathsf{ar}(f)$.

The *dominance relation* $\pi \triangleleft^* \pi'$ is valid in τ if π is an ancestor of π', i.e. if π is above π' in τ, resp. if π is a prefix of π'. *Strict dominance* $\pi \triangleleft^+ \pi'$ holds in τ if $\pi \triangleleft^* \pi'$ but not $\pi{=}\pi'$ in τ. *Disjointness* $\pi \perp \pi'$ is valid in τ if neither $\pi \triangleleft^* \pi'$ nor $\pi' \triangleleft^* \pi$ in τ.

We now define the parallelism relation. We consider more general tree segments than in the introduction (Fig. 1) where several holes are permitted (see Fig. 2).

Definition 1. *A* segment σ *of a tree* τ *is a tuple of nodes in* τ *– written as* $\pi/\pi_1, \ldots, \pi_n$ *– where* π *dominates all* π_i *which in turn are pairwise disjoint. We call* π *the root of the segment and* $\pi_1, \ldots, \pi_n$ *its holes. The segment* $\pi/$ *is the segment with 0 holes.*

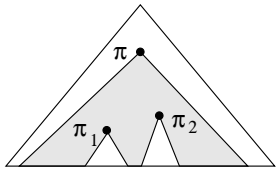


Fig. 2. Segment $\pi/\pi_1, \pi_2$

A segment can be seen as an occurrence of a context: Let $\{\bullet_1, \ldots, \bullet_n, \ldots\}$ be an infinite set of *hole markers*. A *context* γ with n holes over Σ is a tree over Σ and hole

markers $\{\bullet_1, \ldots, \bullet_n\}$ such that each of the hole markers occurs exactly once in γ. For instance, $f(\bullet_2, g(\bullet_1))$ is a context with two holes. Every segment σ of a tree τ with n holes defines a unique context with n holes:

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

The substitutions $[\pi_i/\bullet_i]$ remove the subtrees below the segment holes. The order in which the substitutions are performed does not matter since all holes π_i of a segment are pairwise disjoint. Note also, that the root π of a segment is never removed from τ since it dominates all holes π_i.

Definition 2. Parallelism $\sigma_1 \sim \sigma_2$ *is valid in a tree* τ *if the segments* σ_1 *and* σ_2 *of* τ *are occurrences of the same context, i.e. iff* $\mathsf{context}_\tau(\sigma_1) = \mathsf{context}_\tau(\sigma_2)$.

We now define the purely conjunctive language of *parallelism constraints*. We assume an infinite set $\mathcal{V}_{\mathrm{node}}$ of *node variables* X, Y, Z.

$$\begin{aligned} P &::= X{:}f(X_1, \ldots, X_n) \mid S_1 \sim S_2 \mid P_1 \ \wedge \ P_2 \\ S &::= X/X_1, \ldots, X_m \end{aligned}$$

A parallelism constraint P is a conjunction of labeling and parallelism literals. They are interpreted over the respective relations of some tree in the usual Tarski'an manner. We use *segment terms* S of the form $X/X_1, \ldots, X_m$ to describe segments with m holes, given that the values of X and $X_1, \ldots, X_m$ satisfy the conditions imposed on the root and holes of segments in Definition 2. A parallelism literal $S_1 \sim S_2$ requires that S_1 and S_2 denote segments.

Note that dominance literals $X \triangleleft^* Y$ can also be expressed even though they are not directly part of the language. This follows from Definition 2 which forces roots of segments to dominate holes so that the following equivalence $X \triangleleft^* Y \leftrightarrow X/Y \sim X/Y$ gets valid.

Parallelism constraints are useful to model the meaning of natural language ellipses [11]. Here they avoid the over-generation problem of the previous approach based on higher-order unification [7]. Consider the sentence simplistic example: *Peter sings and so does Bill*. The meaning of this sentence is represented by the formula:

$$\mathsf{and}(\mathsf{sing}(\mathsf{peter}), \mathsf{sing}(\mathsf{bill}))$$

which is a tree. A simplified compositional semantics could construct the following tree description from the syntactic structure of the sentence. Node X_1 stands for the semantics of the source clause *Peter sings*, Y_0 for the semantics of the target clause *so does Bill*. The semantics of the complete sentence starts at node Z:

$Z{:}\mathsf{and}(X_0, Y_0) \wedge$	conjunction of source and target
$X_0 \triangleleft^* X_1 \ \wedge \ X_1{:}\mathsf{sing}(X_2) \ \wedge \ X_2{:}\mathsf{peter} \wedge$	source clause
$Y_0 \triangleleft^* Y_1 \ \wedge \ Y_1{:}\mathsf{bill} \wedge$	target clause
$X_0/X_2 \sim Y_0/Y_1$	ellipses description

The parallelism literal $X_0/X_2 \sim Y_0/Y_1$ states that the semantics of the source clause without peter is equal to the semantics of the target clause up to bill. In the given solution, the terms X_0/X_2 and Y_0/Y_1 denote the two occurrences of the context $\mathsf{sing}(\bullet_1)$.

For a less trivial example consider the sentence *peter sings a song and so does bill*. It has two readings (there is a song that both sing, or both sing different songs). It is possible and appropriate to represent both readings with a single constraint.[3]

To keep this section self-contained let us quickly recall some model theoretic notions. We write $\mathsf{var}(P)$ for the set of free variables of a constraint P. A *variable assignment* to the nodes of a tree τ is a total function $\alpha : V \to \mathsf{nodes}(\tau)$ where V is a finite subset of node variables. A *solution* of a constraint P thus consists of a tree τ and a variable assignment $\alpha : V \to \mathsf{nodes}(\tau)$ such that $\mathsf{var}(P) \subseteq V$. As usual, we require that all literals of a constraint P are validated by every solution τ, α of P. We write $\tau, \alpha \models P$ if τ, α is a solution of P. A formula P is *valid in a tree* τ if $\tau, \alpha \models P$ holds for all α whose domain subsumes $\mathsf{var}(P)$. We write $\alpha_{|V'}$ for the restriction of a variable assignment $\alpha : V \to \mathsf{nodes}(\tau)$ to the variables in V'.

3 Tree Regular Constraints

We next introduce tree regular constraints and show how to express them in logics. A *tree regular constraints* R has the form:

$$R ::= \mathsf{tree}(X) \in \mathcal{L}(\mathcal{A}) \mid R_1 \ \wedge \ R_2$$

Interpreted over a tree τ, the term $\mathsf{tree}(X)$ denotes the subtree of τ rooted by X, while $\mathcal{L}(\mathcal{A})$ stands for the tree language accepted by tree automaton $\mathcal{A}$ over Σ.

But which properties of trees can be expressed by tree regular constraints? Can we express, for instance, a first-order dominance formula which requires that no f labeled node intervenes between nodes X and Y? Such formulas are needed in an application of CLLS [14].

3.1 Monadic Second-Order Dominance Logic

We next define the *monadic second-order dominance logic* (SDom) to be the monadic second-order logic over dominance constraints, i.e. of ground terms. Note that monadic second-order logics were already investigated for many other graph structures (e.g. [6]).

We assume an infinite set $\mathcal{V}_{\mathrm{set}}$ of *monadic second-order variables* A, B that denote sets of nodes. The formulas D of SDom have the form:

$$D ::= X \triangleleft^* Y \mid X{:}f(X_1, \ldots, X_n) \mid X \in A \mid D \ \wedge \ D' \mid \neg D \mid \exists X.\ D \mid \exists A.\ D$$

Beyond of conjunctions of dominance and labeling literals, there are membership constraints, existential quantification over nodes and sets, negation, and thus universal quantification.

The logic SDom is interpreted over ground terms. Every ground term τ now defines a two sorted domain: $\mathsf{domain}_\tau = \mathsf{nodes}(\tau) \uplus 2^{\mathsf{nodes}(\tau)}$. Variables assignments to a tree τ are functions $\alpha : V \to \mathsf{domain}_\tau$ defined on a finite set $V \subseteq \mathcal{V}_{\mathrm{node}} \uplus \mathcal{V}_{\mathrm{set}}$ which

[3] One can use dominance constraints to leave the scope of the quantifier *a song* underspecified so that parallelism constraints correctly model its interaction with the ellipses.

map node variables to nodes and set variables to sets of nodes, i.e. for all $X, A \in V$: $\alpha(X) \in \mathsf{nodes}(\tau)$ and $\alpha(A) \in 2^{\mathsf{nodes}(\tau)}$.

The language SDom is closely related to the weak monadic second-order logic of the complete binary tree (WS2S) [25, 9]. This was first noticed by Rogers in 1995 [2]. The models of SDom are ground terms while the only model of WS2S is the infinite binary tree. The later is simpler in that all its nodes have first and second successors (children). This allows to found WS2S on the two successor functions while SDom must rely on the labeling relation.

Still, one can encode all ground terms in the infinite binary tree and thereby encode SDom into WS2S. This was used in [15] to encode the first-order theory of dominance constraints in WS2S [15]. The current section generalizes and complements these earlier result.

Proposition 1. *Every tree regular constraint R is equivalent to some formula D in the monadic second-order dominance logic over the same signature.*

Proof. Let $\mathcal{A}$ be a tree automaton and X a node variable. We show how to express $\mathsf{tree}(X) \in \mathcal{L}(\mathcal{A})$ through an equivalent formula D of SDom. Let Q be the set of states of $\mathcal{A}$ and Q_{fin} the set of its final states. We consider all states $q \in Q$ as second-order variables, whose set value contains all those nodes Y such that $\mathsf{tree}(Y)$ has a run into state q in $\mathcal{A}$. We then require that the value of $\mathsf{tree}(X)$ has a run into a final state, i.e. that $\mathsf{tree}(X) \in q$ for some final state $q \in Q_{\mathsf{fin}}$.

$$D = \exists Q.(\bigvee_{q \in Q_{\mathsf{fin}}} X \in q \ \wedge \bigwedge_{q \in Q} \forall Y.\ (Y \in q \leftrightarrow \mathsf{step}_{\mathcal{A}}(Y, q)))$$

where $\mathsf{step}_{\mathcal{A}}(Y, q)$ step means that there is a single automaton step proving that the value of $\mathsf{tree}(Y)$ has a run into q.

$$\mathsf{step}_{\mathcal{A}}(Y, q) = \bigvee_{f(q_1,\ldots,q_n) \to q \in \mathcal{A}} \exists Y_1 \ldots \exists Y_n.\ (Y{:}f(Y_1, \ldots, Y_n) \wedge Y_1 \in q_1 \ \wedge \ \ldots \ \wedge \ Y_n \in q_n)$$

Note that all states of $\mathcal{A}$ may belong to the set of free set variables of formula $\mathsf{step}_{\mathcal{A}}(Y, q)$ so that the values of all sets $q \in Q$ are defined by mutual recursion.

The converse of the above proposition is wrong. For instance, one cannot express $X \triangleleft^* Y$ equivalently by tree regular constraints R since satisfiable tree regular constraints can always be satisfied such that all variables denote disjoint nodes. Nevertheless, a weakened converse modulo satisfaction equivalence still holds.

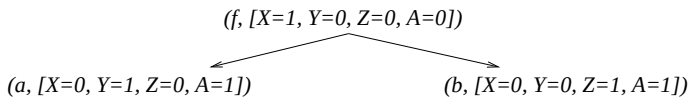
Theorem 1. *Every tree regular constraint R is satisfaction equivalent to some formula D of the monadic second-order dominance logic over the same signature, and vice versa.*

This theorem establishes a bidirectional relationship between dominance logics and tree automata. The one direction is already proved (Proposition 1). The proof of the other direction relies on standard encoding techniques known from WS2S. For every formula of SDom, we have to construct a tree automaton that recognizes all its solutions converted into some tree format (Corollary 1 below). This format is obtained by encoding information about the values of node variables into extended node labels of some extended signature.

3.2 Extending Node Labels

This trick is to encode a solution pair τ, α into a single tree which looks like the tree τ except that it contains all information about the variable assignment α in extended node labels. Given a formula D of SDom, one can then recognize all encoded solutions of D by a tree automaton.

We first illustrate the encoding of pairs τ, α at an example. Let τ be the tree $f(a, b)$ with nodes $\mathsf{nodes}(\tau) = \{\epsilon, 1, 2\}$ and α be the variable assignment given by $\alpha(X) = \epsilon$, $\alpha(Y) = 1$, $\alpha(Z) = 2$, and $\alpha(A) = \{1, 2\}$. We then encode τ, α by the following tree with extended node labels:



In the general case, we encode pairs $\tau, \alpha : V \to \Sigma$ into trees over the signature of extended labels Σ_V:

$$\Sigma_V = \{(f, c) \mid f \in \Sigma, c : V \to \{0, 1\}\}$$

The second components of extended labels (f, c) are finite characteristic functions with domain V. The arity of a label (f, c) in Σ_V is equal to the arity of f. As in the preceding example, we will use the record notation $[Z_1{=}B_1, \ldots, Z_n{=}B_n]$ to represent the finite characteristic function $c : \{Z_1, \ldots, Z_n\} \to \{0, 1\}$ with $c(Z_1) = B_1$, …, $c(Z_n) = B_n$.

We encode a pair $\tau, \alpha : V \to \mathsf{nodes}(\tau)$ through the α-extension $\mathsf{ext}_\alpha(\tau)$. The trees $\mathsf{ext}_\alpha(\tau)$ and τ have the same set of nodes; a node π of $\mathsf{ext}_\alpha(\tau)$ is given the label (f, c) if and only if the same node of τ is given the label f and for all $X, A \in V$:

$$\pi = \alpha(X) \text{ iff } c(X) = 1 \qquad \text{and} \qquad \pi \in \alpha(A) \text{ iff } c(A) = 1$$

We illustrate the encoding at the example of non-intervenance which is crucial for applications [14, 20]. We present a tree automaton which accepts trees where no f-labeled node intervences properly between X and Y:

$$\neg\, \exists Z.\ (X \lhd^+ Z \ \wedge\ Z \lhd^+ Y \ \wedge\ \exists Z_1 \exists Z_2.\ Z{:}f(Z_1, Z_2)\ )$$

Since automata are closed under complementation, it is sufficient to construct an automaton positive intervenance. The signature is Σ_V where $V = \{X, Y\}$. The acceptance state is $q_{\mathsf{above}(X)}$. For all $g \in \Sigma$ we have the following rules:

$$\begin{array}{ll}
(g, [X{=}0, Y{=}0])(q_{\mathsf{below}(Y)}, \ldots, q_{\mathsf{below}(Y)}) & \to q_{\mathsf{below}(Y)} \\
(g, [X{=}0, Y{=}1])(q_{\mathsf{below}(Y)}, \ldots, q_{\mathsf{below}(Y)}) & \to q_{\mathsf{above}(Y)} \\
(g, [X{=}0, Y{=}0])(\ldots, q_{\mathsf{above}(Y)}, \ldots) & \to q_{\mathsf{above}(Y)} \text{ if } f \neq g \\
(f, [X{=}0, Y{=}0])(\ldots, q_{\mathsf{above}(Y)}, \ldots) & \to q_{\mathsf{above}_f} \\
(g, [X{=}0, Y{=}0])(\ldots, q_{\mathsf{above}_f}, \ldots) & \to q_{\mathsf{above}_f} \\
(g, [X{=}1, Y{=}0])(\ldots, q_{\mathsf{above}_f}, \ldots) & \to q_{\mathsf{above}(X)} \\
(g, [X{=}0, Y{=}0])(\ldots, q_{\mathsf{above}(X)}, \ldots) & \to q_{\mathsf{above}(X)}
\end{array}$$

The state $q_{\mathsf{below}(Y)}$ recognizes all trees where $Y{=}0$ for all nodes. The state $\mathsf{above}(Y)$ recognizes trees containing a node where $Y{=}1$; above_f recognizes trees which contain a proper f-labeled ancestor of some node with $Y{=}1$. Finally, $q_{\mathsf{above}(X)}$ accepts all trees where $X{=}1$ occurs properly above an f-ancestor of where $Y{=}1$.

We also need to check that $X{=}1$ and $Y{=}1$ are seen at most once in a node label. This can be done by intersection with another tree automaton.

3.3 Constructing Tree Automata

We now construct tree automata for general formulas of SDom. The following lemma will be useful. We omit its simple proof.

Lemma 1. *If* $\mathsf{ext}_{\alpha_1}(\tau_1) = \mathsf{ext}_{\alpha_2}(\tau_2)$ *then* $\alpha_1 = \alpha_2$ *and* $\tau_1 = \tau_2$.

Proposition 2. *For all second-order dominance formulas D and finite sets V of variables there exists a tree automaton $\mathcal{A}$ over the signature Σ_V which accepts those trees over Σ_V that encode tree-assignment-pairs $\tau, \alpha_{|V}$ such that $\tau, \alpha \models D$:*

$$\mathcal{L}(\mathcal{A}) = \{\mathsf{ext}_{\alpha_{|V}}(\tau) \mid \tau, \alpha \models D\}$$

Proof. We can assume without loss of generality that $\mathsf{var}(D) \subseteq V$. Otherwise, we can apply the proposition to $D' =_{\mathrm{df}} \exists \mathsf{var}(D) - V.\ D$ which satisfies $\mathsf{var}(D') = V$ since $\mathsf{var}(D) - (\mathsf{var}(D) - V) \subseteq V$. The automaton $\mathcal{A}$ for D' recognizes the required language $\mathcal{L}(\mathcal{A}) = \{\mathsf{ext}_{\alpha_{|V}}(\tau) \mid \tau, \alpha \models D'\} = \{\mathsf{ext}_{\alpha_{|V}}(\tau) \mid \tau, \alpha \models D\}$. Let a V-extension of a tree τ be some α-extension of τ with $\alpha : V \to \mathsf{domain}_\tau$. We next construct an automaton $\mathcal{A}_{\mathsf{ext}_V}$ which only accepts those trees over Σ_V that are V extensions of some tree in Σ. This automaton has to check for every first-order variable $X \in V$ and acceptable trees τ that there exists exactly one node in τ whose characteristic function maps X to 1. The automaton $\mathcal{A}_\emptyset$ accepts all trees of Σ. For the general case, let $V_1 \subseteq V$ where V_1 is the set of first order variables we define $\mathcal{A}_{\mathsf{ext}_V} = \bigcap_{X \in V_1} \mathcal{A}_{\mathsf{ext}_{\{X\}}}$. It only remains to define the automata $\mathcal{A}_{\mathsf{ext}_{\{X\}}}$ for singleton sets $\{X\}$. Let $V = \{Z_1, \ldots, Z_n\}$. The rules are:

$$\begin{array}{ll}
(f, [\ldots, X{=}0, \ldots])(q_{\mathsf{none}}, \ldots, q_{\mathsf{none}}) & \to q_{\mathsf{none}} \\
(f, [\ldots, X{=}1, \ldots])(q_{\mathsf{none}}, \ldots, q_{\mathsf{none}}) & \to q_{\mathsf{once}} \\
(f, [\ldots, X{=}0, \ldots])(q_{\mathsf{none}}, \ldots, q_{\mathsf{none}}, q_{\mathsf{once}}, q_{\mathsf{none}}, \ldots, q_{\mathsf{none}}) & \to q_{\mathsf{once}}
\end{array}$$

The automaton counts how often $X{=}1$ was seen. It starts with q_{none} and increments to q_{once} when the first occurrences comes, and rejects starting from the second occurrence. The only finial state of $\mathcal{A}_{\mathsf{ext}_{\{X\}}}$ is q_{once}.

We next construct automata $\mathcal{A}_D$ over the signature Σ_V that check the validity of D. The proposition is then always satisfied with $\mathcal{A} = \mathcal{A}_D \cap \mathcal{A}_{\mathsf{ext}_V}$. The construction is by induction on formulas D.

1. Case $D = X{=}Y$. We construct the following automaton that checks whether $X{=}1$ and $Y{=}1$ occur simultaneously at some same. The only final states q_{equal} of $\mathcal{A}_D$ indicates this case. The state q_{all} can reached without restrictions.

$$\begin{array}{ll}
(f, [\ldots, X{=}0, \ldots, Y{=}0, \ldots])(q_{\mathsf{all}}, \ldots, q_{\mathsf{all}}) & \to q_{\mathsf{all}} \\
(f, [\ldots, X{=}1, \ldots, Y{=}1, \ldots])(q_{\mathsf{all}}, \ldots, q_{\mathsf{all}}) & \to q_{\mathsf{equal}} \\
(f, [\ldots, X{=}0, \ldots, Y{=}0, \ldots])(\ldots, q_{\mathsf{equal}}, \ldots) & \to q_{\mathsf{equal}}
\end{array}$$

2. Case $D = X \lhd^+ Y$. We construct the following automaton that checks whether $Y{=}1$ is seen properly below $X{=}1$. The final state of $\mathcal{A}_D$ is $q_{\mathsf{above}(X)}$.

$$\begin{array}{ll}
(f, [\ldots, X{=}0, \ldots, Y{=}0, \ldots])(q_{\mathsf{all}}, \ldots, q_{\mathsf{all}}) & \to q_{\mathsf{all}} \\
(f, [\ldots, Y{=}1, \ldots])(q_{\mathsf{all}}, \ldots, q_{\mathsf{all}}) & \to q_{\mathsf{above}(\mathsf{Y})} \\
(f, [\ldots, X{=}0, \ldots])(\ldots, q_{\mathsf{above}(\mathsf{Y})}, \ldots) & \to q_{\mathsf{above}(\mathsf{Y})} \\
(f, [\ldots, X{=}1, \ldots])(\ldots, q_{\mathsf{above}(\mathsf{Y})}, \ldots) & \to \mathsf{above}(X) \\
(f, [\ldots])(\ldots, q_{\mathsf{above}(X)}, \ldots) & \to q_{\mathsf{above}(X)}
\end{array}$$

3. Case $D = X \lhd^* Y$. Tree automata are closed under union, so let $\mathcal{A}_D{=}\mathcal{A}_{X \lhd^+ Y} \cup A_{X=Y}$.
4. Case $D = \exists X.D'$. We can assume without loss of generality that $X \notin V$. Let $\mathcal{A}_{D'}$ be the automaton for D' but over the extended signature $\Sigma_{V \uplus \{X\}}$. We call a tree τ over Σ_V an X-projection of a tree τ' over $\Sigma_{V \uplus \{X\}}$ if τ is obtained from τ' by restricting all characteristic functions in node labels of τ' to V. We can easily define the automaton $\mathcal{A}_D$ such that it accepts all X-projections of trees in $\mathcal{L}(\mathcal{A}_{D'})$.
5. Other cases: The constructions for labeling $X{:}g(X_1, \ldots, X_n)$ and membership literals $X \in A$ are obvious. Conjunctions $D_1 \wedge D_2$ and negation $\neg D$ can be reduced to complementation and intersection of tree automata. Second-order quantification $\exists A.D$ can be encoded as in the first-order case.

We can now prove that tree regular constraints can indeed express second order monadic dominance formulas modulo satisfaction equivalence (but not equivalence). This completes the proof of Theorem 1.

Corollary 1. *For every monadic second-order formula D of SDom there exists a satisfaction equivalent tree regular constraint R over the same signature.*

Proof. We can assume w.l.o.g. that D is closed. Let $V = \emptyset$ and let $\mathcal{A}$ be a tree automaton according to Proposition 2 that satisfies: $\mathcal{L}(\mathcal{A}) = \{\tau \mid \tau \models D\}$. We don't need any variable assignment to interpret D since D is closed. Let X, Y be fresh variables. The following conditional equivalence is valid in all trees:

$$\forall Y.\ X \lhd^* Y \ \to\ (D \ \leftrightarrow\ \mathsf{tree}(X) \in \mathcal{L}(\mathcal{A}))$$

If a tree τ, α satisfies the assumption $\forall Y.\ X \lhd^* Y$ then $\alpha(X)$ must be the root of τ. In this case, $\mathsf{tree}(\alpha(X)) \in \mathcal{L}(\mathcal{A})$ is equal to $\tau \in \mathcal{L}(\mathcal{A})$ which is $\tau \models D$. Next note that the assumption $\forall Y.\ X \lhd^* Y$ can be met while solving $\mathsf{tree}(X) \in \mathcal{L}(\mathcal{A})$ resp. D. Thus, D is satisfaction equivalent to tree regular constraint $\mathsf{tree}(X) \in \mathcal{L}(\mathcal{A})$.

4 Extensions of Parallelism Constraints

Our next goal is to lift Theorem 1 to extensions of parallelism constraints. This means that we want to reduce satisfiability of a conjunction $P \wedge R$ to the satisfiability of some conjunctions $P' \wedge D$ and vice versa.

Theorem 2. *The satisfiability problems of parallelism plus tree regular constraints $P \wedge R$ resp. parallelism constraints plus monadic second-order dominance formulas $P' \wedge D$ are equal modulo non-deterministic polynomial time transformations.*

Note that the signatures are part of the input of both satisfiability problems, i.e. the satisfaction equivalent formulas need not be defined over the same signature.

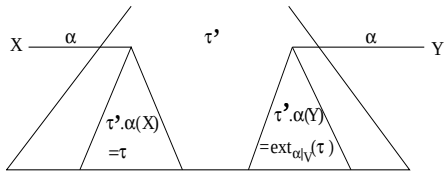
The one direction still follows immediately from Proposition 1 (which is modulo equivalence). But we cannot directly apply Theorem 1 to prove the converse. This weakness is due to the notion of satisfaction equivalence used there in contrast to ordinary equivalence.

Proposition 3. *Every conjunction $P \wedge D$ of a parallelism constraint with a formula of SDom is satisfaction equivalent to some formula $\bigvee_{i=1}^{n} P_i \wedge R_i$ with parallelism plus tree regular constraints.*

The proof captures the rest of this section. The idea is to describe a solution τ, α of $P \wedge D$ by talking about a large tree that contains τ and $\mathsf{ext}_\alpha(\tau)$ simultaneously. The translation keeps the parallelism constraint P in order to describe τ while it expresses the dominance formula D through a tree regular constraint about $\mathsf{ext}_\alpha(\tau)$. The intended relationship between τ and $\mathsf{ext}_\alpha(\tau)$ is enforced by additional parallelism constraints (Lemma 2).

We first introduce formulas $\mathsf{ext}_V(X, Y)$ for finite sets V of variables. The free variables of $\mathsf{ext}_V(X, Y)$ are those in $V \cup \{X, Y\}$. A pair τ', α satisfies $\mathsf{ext}_V(X, Y)$ if the tree below $\alpha(Y)$ in τ' is the $\alpha_{|V}$ extension of the tree below $\alpha(X)$ in τ', i.e.:

$\tau', \alpha \models \mathsf{ext}_V(X, Y)$
iff
$\tau'.\alpha(Y) = \mathsf{ext}_{\alpha_{|V}}(\tau'.\alpha(X))$



Every solution τ', α of $\mathsf{ext}_V(X, Y)$ indeed contains occurrences of $\tau = \tau'.\alpha(X)$ and its extension $\mathsf{ext}_{\alpha_{|V}}(\tau) = \tau'.\alpha(Y)$ simultaneously. Note that $\alpha_{|V}$ must map to nodes of τ by definition of extensions, while the unrestricted assignment α may map to arbitrary nodes of τ'.

From now on, let us identify the labels f and (f, c) where c is the constant 0-valued function. Through this identification, we turn Σ into a subset of Σ_V. This has an important consequence: if V contains only first-order variables then the trees τ and $\mathsf{ext}_\alpha(\tau)$ have the same structure with finitely many exceptions: for all $Z \in V$ the node $\alpha(Z)$ below $\alpha(X)$ and its correspondent below $\alpha(Y)$ carry distinct labels. The number of exceptions is bounded by size of V. This property would fails if we permitted second-order variables in V: a single second-order variable $A \in V$ where $\alpha(Y)$ contains all nodes of τ makes all corresponding node labels of τ and $\mathsf{ext}_{\alpha_{|V}}(\tau)$ distinct.

Lemma 2. *Let V be a set of first-order variables. Every formula $\mathsf{ext}_V(X, Y)$ is equivalent to some positive existentially quantified formula $\bigvee_{l=1}^{n} \exists Z_1^l \ldots \exists Z_{k_l}^l P_l$.*

Proof. We construct a formula E of the above form by induction on the since of V. If $V = \emptyset$ then we set $E =_{\mathrm{df}} X/{\sim}Y/$. Otherwise, we guess node labels for all variables in V and all relationships between them: properly above, properly below the i-th children, equal, or disjoint. These are $O(|V|^2 * M)$ guesses where M is the maximal arity of

function symbols in Σ. We then translate deterministically for all possible choices. Let $X_1, \ldots, X_n$ be some maximal set of top-most situated variables that take distinct values (according to our guesses). We define:

$$E =_{\mathrm{df}} \exists Y_1, \ldots, \exists Y_n.\ X/X_1, \ldots, X_n \sim Y/Y_1, \ldots, Y_n \ \wedge \ \bigwedge_{i=1}^{n} E_i$$

The formulas E_i are still to be defined. Let $c_i : V \to \{0, 1\}$ be the function that map all variables to 1 that take the same value as X_i and all others to 0 (according to our guesses). Let f_i be the guessed node label of arity n_i for the variable X_i and V_i^j be the set of variables lying below the j-th child of X_i. We then define:

$$\begin{aligned} E_i =_{\mathrm{df}} \ & \exists X_1^1 \ldots \exists X_n^{n_i}.\ X_i{:}f_i(X_i^1, \ldots, X_i^{n_i}) \wedge \\ & \exists Y_1^1 \ldots \exists Y_n^{n_i}.\ Y_i{:}(f_i, c_i)(Y_i^1, \ldots, Y_i^{n_i}) \ \wedge \ \textstyle\bigwedge_{j=1}^{n_i} \mathsf{ext}_{V_i^j}(X_i^j, Y_i^j) \end{aligned} \qquad \square$$

Proof (of Proposition 3). We consider a formula $P \ \wedge \ D$ were D does not contain free second-order variables w.l.o.g. Otherwise, we can produce a satisfaction equivalent formula of the same form by existential quantification.

Let X be a fresh variable and $V = \mathsf{var}(P \wedge D) \cup \{X\}$ a set of first-order variables. We next define a formula E that we will prove satisfaction equivalent to $P \ \wedge \ D$:

$$E =_{\mathrm{df}} P \ \wedge \ \exists Y.\ \mathsf{ext}_V(X, Y) \ \wedge \ \mathsf{tree}(Y) \in \{\mathsf{ext}_{\alpha_{|V}}(\tau) \mid \tau, \alpha \models D\}$$

First note that E can be rewritten into a satisfaction equivalent disjunction of the required form $\bigvee_{i=1}^{n} P_i \wedge R_i$. We can express $\mathsf{ext}_V(X, Y)$ by a disjunction of parallelism constraints up to satisfaction equivalence (Lemma 2) and state the membership condition on $\mathsf{tree}(Y)$ by a tree regular constraint (Proposition 2).

It remains to show that E is satisfaction equivalent to $P \wedge D$. For the one direction, suppose $\tau', \alpha' \models E$. We show that $\tau'.\alpha'(X), \alpha'_{|V} \models P \ \wedge \ D$. First note that $\alpha'_{|V}$ maps to nodes below $\alpha'(X)$ since $\tau', \alpha' \models \mathsf{ext}_V(X, Y)$. Second note that $\alpha'_{|V}$ can interpret all variables of $P \ \wedge \ D$ by definition of V. Third, we show that $\tau'.\alpha'(X), \alpha'_{|V}$ solves P: By assumption, $\tau', \alpha' \models E$ and thus $\tau', \alpha'_{|V} \models P$. But since P contains parallelism literals only, we can restrict this solution to the subtree of τ' to which $\alpha'_{|V}$ maps; thus: $\tau'.\alpha'(X), \alpha'_{|V} \models P$. Forth, we show that $\tau'.\alpha'(X), \alpha'_{|V}$ solves D. Since τ', α' satisfies the membership restriction on $\mathsf{tree}(Y)$ there exists a solution $\tau, \alpha \models D$ such that:

$$\tau'.\alpha'(Y) = \mathsf{ext}_{\alpha_{|V}}(\tau)$$

Since $\tau', \alpha' \models \mathsf{ext}_V(X, Y)$ we also know $\tau'.\alpha'(Y) = \mathsf{ext}_{\alpha'_{|V}}(\tau'.\alpha'(X))$. The previous two equations combine into $\mathsf{ext}_{\alpha_{|V}}(\tau) = \mathsf{ext}_{\alpha'_{|V}}(\tau'.\alpha'(X))$ such that the uniqueness Lemma 1 yields $\alpha'_{|V} = \alpha_{|V}$ and $\tau'.\alpha'(X) = \tau$. From $\tau, \alpha \models D$, we get $\tau, \alpha_{|V} \models D$, and hence, $\tau'.\alpha'(X), \alpha'_{|V} \models D$.

For the other direction, we assume that $P \wedge D$ is satisfiable and construct a solution of E. Let τ, α be a solution of $P \ \wedge \ D$. We define $\tau' = f(\tau, \mathsf{ext}_{\alpha_{|V}}(\tau), \ldots)$ where f is some function symbol of arity at least 2. (The children of τ starting from position 3 can be chosen arbitrarily.) Let π_1 be the first child of the root of τ'. It then holds that $\tau', \alpha[X \mapsto \pi_1] \models E$ whereby the existentially quantified variable Y can be mapped to the second child of τ'. $\square$

5 Relation to Context Unification

Parallelism constraints and context unification have the same expressiveness [19]. We now show that this result can be lifted when extending both languages with tree regular constraints.

We first recall the definition of context unification with tree regular constraints. The version of context unification we use is quite rich but can be reduced to the standard version.

Context unification is equation solving in the algebra of contexts where contexts may have one or arbitrary many holes. We consider contexts γ with n holes as n-ary functions on trees:

$$\gamma(\tau_1, \ldots, \tau_n) = \gamma[\bullet_1/\tau_1] \ldots [\bullet_n/\tau_n]$$

Contexts of arity 0 can be identified with trees. We assume a set of *context variables* F, G with arities $\mathsf{ar}(F) \geq 0$ which contains infinitely many variables for all arities. The arity of F determines the number of holes of the value of F. We next define *context terms* t over Σ where $f \in \Sigma$, $\mathsf{ar}(f) = n$, and $\mathsf{ar}(F) = m$.

$$t ::= f(t_1, \ldots, t_n) \mid F(t_1, \ldots, t_m) \mid \bullet_i$$

An n-ary context term is a context term with hole markers $\bullet_1, \ldots, \bullet_n$ each of which occurs exactly once. An n-ary context term denotes a context with n holes. A *context equation* is a pair $t_1 = t_2$ between n-ary context terms.

Context unification is the problem of solving finite conjunctions of context equations. For instance, the context equation $F(x, b){=}f(a, G(b))$ is solved by the variable assignment β with $\beta(F) = f(\bullet_1, \bullet_2)$, $\beta(G) = \bullet_1$ and $\beta(x) = a$. The problem can be freely restricted in several ways: It is sufficient to have a single equation and context variables of arity 1 only.

A *tree variable* x is a context variable of arity 0. The extension of context unification with tree regular constraints allows for membership literals $x \in \mathcal{L}(\mathcal{A})$ to be added to equation sets where $\mathcal{A}$ is a tree automaton over Σ.

Theorem 3. *The extensions of parallelism constraints and context unification with tree regular constraints are equivalent modulo polynomial time reductions.*

As shown by Levy and Villaret there is a third equivalent problem which is linear second-order unification (LSOU) with tree regular constraints [16].

The proof of Theorem 3 is non-trivial but can be obtained by extending the proof in [19]. We show both implications independently. We first translate CU+R to P+R. Here, we simplify the argument of [19]. Suppose w.l.o.g that we are given a single equation $t_1 = t_2$ and a single tree regular constraint $x \in \mathcal{L}(\mathcal{A})$. We first introduce fresh node variables for all subterm positions in the equation $t_1 = t_2$. We then collect parallelims, labeling, and membership literals in four steps.

1. We collect labeling literals for all subterms in $t_1 = t_2$ that have the form $f(s_1, \ldots, s_n)$. Let X be the node variable for the position of such a subterm and $X_1, \ldots, X_n$ the node variables for the positions of the subterms $s_1, \ldots, s_n$. We then add the labeling literal:
$$X{:}f(X_1, \ldots, X_n)$$

2. We collect parallelism literals for all context variables occuring in $t_1 = t_2$. So let $F(s_1, \ldots, s_n)$ be an occurences of some context variable F in $t_1 = t_2$, X be the node variable of this occurence and $X_1, \ldots, X_n$ the node variables of the subterms s_1,..., s_n. Let $F(s'_1, \ldots, s'_n)$ be a second possibly equal occurences of same context variable F in $t_1 = t_2$, Y be the node variable of this occurence and $Y_1, \ldots, Y_n$ the node variables of the subterms $s'_1, \ldots, s'_n$. We then add the parallelism literal:
$$X/X_1, \ldots, X_n \sim Y/Y_1, \ldots, Y_n$$
3. Suppose that x occurs in the equation $t_1 = t_2$ at some position with node variable X. We then add:
$$\mathsf{tree}(X) \in \mathcal{L}(\mathcal{A})$$
4. We ensure that both sides of the equation $t_1 = t_2$ denote equal values. Let X_1 and X_2 be the node variables of the subterm positions of t_1 and t_2. We then add the parallelims literal:
$$X_1/ \sim X_2/$$

Example 1. For instance, the context equation $F(f(x)) = f(F(a))$ with regular constraint: $x \in \mathcal{L}(\mathcal{A})$

We first introduce node variables for all subterm positions. The above constraint is then translated as follows where the lines contain the literals of the subsequent steps:

$$\begin{array}{ccccccc} X_0 & X_1 & X_2 & & Y_0 & Y_1 & Y_2 \\ \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow & \downarrow \\ F(& f(& x\,)) & = & f(& F(& a\,)) \end{array}$$

1. $X_1{:}f(X_2) \wedge X_2{:}a \wedge Y_0{:}f(Y_1) \wedge Y_2{:}a \wedge$
2. $X_0/X_1 \sim Y_1/Y_2 \wedge$
3. $\mathsf{tree}(X_2) \in \mathcal{L}(\mathcal{A}) \wedge$
4. $X_0/ \sim Y_0/$

In step 2 of this example we have freely omitted parallelism literals between equal segment terms: $X_0/X_1 \sim X_0/X_1$ and $Y_1/Y_2 \sim Y_1/Y_2$. These literal enforce dominance relations $X_0 \triangleleft^* X_1$ and $Y_1 \triangleleft^* Y_2$ that are entailed by $X_0/X_1 \sim Y_1/Y_2$ anyway.

Lemma 3. *A context equation with tree regular constraints $t_1 = t_2 \wedge x \in \mathcal{L}(\mathcal{A})$ is satisfiable if and only if its translation is.*

We give an inverse reduction which maps P+R to CU+R. The difficulty of this reduction is raised by the different views on trees: While parallelism constraints talk about nodes and segments, context unification deals with trees and contexts. So how can we speak about the nodes of a tree in context unification? The idea is that we speak about the context between the root of the tree and this node.

We now encode an extended parallelism constraint $E = P \wedge R$ with the set of node variable $V = \mathsf{vars}(E)$. Let x_{all} be a tree variable which is supposed to denote a model of E. For every node variable $X \in V$ let F_X be a unary context variable, denoting the context from the root of x_{all} to node X, and a first-order variable x denoting the tree below X in x_{all}. We express these relationships through the context equations e_V:
$$e_V \quad =_{\mathrm{df}} \quad \bigwedge_{X \in V} x_{all} = F_X(x)$$

$$
\begin{array}{rll}
\lfloor X : f(X_1,\ldots,X_n)\rfloor & =_{\mathrm{df}} & F_{X_1}(\bullet_1){=}F_X(f(\bullet_1,x_2,\ldots,x_n)) \\
& & \wedge\ldots\wedge\ F_{X_n}(\bullet_1){=}F_X(f(x_1,\ldots,x_{n-1},\bullet_1)) \\
\lfloor X : a\rfloor & =_{\mathrm{df}} & x{=}a \\
\lfloor X/X_1,\ldots,X_n \sim Y/Y_1,\ldots,Y_n\rfloor & =_{\mathrm{df}} & \exists F(F_{X_1}(\bullet_1){=}F_X(F(\bullet_1,x_2,\ldots,x_n))\wedge \\
& & \quad F_{Y_1}(\bullet_1){=}F_Y(F(\bullet_1,y_2,\ldots,y_n)\wedge \\
& & \quad\quad \wedge\ \ldots\ \wedge \\
& & \quad F_{X_n}(\bullet_1){=}F_X(F(x_1,x_2,\ldots,\bullet_1))\wedge \\
& & \quad F_{Y_n}(\bullet_1){=}F_Y(F(y_1,y_2,\ldots,\bullet_1))) \qquad (F \text{ fresh}) \\
\lfloor E_1 \wedge E_2\rfloor & =_{\mathrm{df}} & \lfloor E_1\rfloor \wedge \lfloor E_2\rfloor \\
\lfloor \mathsf{tree}(X) \in \mathcal{L}(\mathcal{A})\rfloor & =_{\mathrm{df}} & x \in \mathcal{L}(\mathcal{A})
\end{array}
$$

Fig. 3. Reduction of P+R to CU+R

The translations $\lfloor E\rfloor$ of the literals of E is given in Figure 3.

Lemma 4. *An extended parallelism constraint $P \wedge R$ with variable set V is satisfiable if and only the system of context equation $e_V \wedge \lfloor P \wedge R\rfloor$ is.*

Conclusion

We have presented a new relationship between tree regular constraints and the second-order monadic dominance logic. We have lifted this relationship to the respective extensions of parallelism constraints, P+R and P+SDom. We have also proved that CU with tree regular constraints is equivalent to parallelism and tree regular constraint. To summarize, the following four languages have equivalent satisfiability problems (modulo non-elementary time reductions):

$$\text{P + SDom = P + R = CU + R = LSOU + R}$$

The first three equations are contributed by the present paper while the last equation was proved before [16]. Our result is relevant for classifying different extensions of parallelism constraints, as in the constraint language for lambda structures (CLLS). For instance, we will show in a forthcoming paper [20] that parallelism constraints plus lambda binding constraints of CLLS can be expressed in P+SDom and thus in all equivalent languages.

References

1. Ernst Althaus, Denys Duchier, Alexander Koller, Kurt Mehlhorn, Joachim Niehren, and Sven Thiel. An efficient algorithm for the configuration problem of dominance graphs. In *12th ACM-SIAM Symposium on Discrete Algorithms*, pages 815–824, 2001.
2. R. Backofen, J. Rogers, and K. Vijay-Shanker. A first-order axiomatization of the theory of finite trees. *Journal of Logic, Language, and Information*, 4:5–39, 1995.
3. Manuel Bodirsky, Katrin Erk, Alexander Koller, and Joachim Niehren. Beta reduction constraints. In *RTA'01*, volume 2051 of *LNCS*, pages 31–46, 2001.

4. Manuel Bodirsky, Katrin Erk, Alexander Koller, and Joachim Niehren. Underspecified beta reduction. In *ACL'01*, pages 74–81, 2001.
5. Hubert Comon. Completion of rewrite systems with membership constraints. *Symbolic Computation*, 25(4):397–453, 1998. Extends on a paper at ICALP'92.
6. Bruno Courcelle. The monadic second-order logic of graphs XIII: Graph drawings with edge crossings. *Computational Intelligence*, 244(1-2):63–94, 2000.
7. Mary Dalrymple, Stuart Shieber, and Fernando Pereira. Ellipsis and higher-order unification. *Linguistics & Philosophy*, 14:399–452, 1991.
8. Volker Diekert, Claudio Guterrez, and Christian Hagenah. The existential theory of equations with rational constraints in free groups is pspace-complete. In *STACS 2001*, volume 2010 of *LNCS*, pages 170–182, 2001.
9. John Doner. Tree acceptors and some of their applications. *Journal of Computer System Science*, 4:406– 451, 1970. Received December 1967, Revised May 1970.
10. Denys Duchier and Claire Gardent. Tree descriptions, constraints and incrementality. In *Computing Meaning*, volume 77 of *Studies In Linguistics And Philosophy*, pages 205–227. Kluwer Academic Publishers, 2001.
11. Markus Egg, Alexander Koller, and Joachim Niehren. The constraint language for lambda structures. *Logic, Language, and Information*, 10:457–485, 2001.
12. Katrin Erk, Alexander Koller, and Joachim Niehren. Processing underspecified semantic representations in the constraint language for lambda structures. *Journal of Language and Computation*, 2002. To appear.
13. Katrin Erk and Joachim Niehren. Parallelism constraints. In *RTA'00*, volume 1833 of *LNCS*, pages 110–126, 2000.
14. Alexander Koller and Joachim Niehren. On underspecified processing of dynamic semantics. In *18th Int. Conf. on Computational Linguistics*, pages 460–466, July 2000.
15. Alexander Koller, Joachim Niehren, and Ralf Treinen. Dominance constraints: Algorithms and complexity. In *LACL'98*, volume 2014 of *LNAI*, 2001.
16. Jordi Levy and Mateu Villaret. Linear second-order unification and context unification with tree-regular constraints. In *RTA'00*, volume 1833 of *LNCS*, pages 156–171, 2000.
17. Jordi Levy and Mateu Villaret. Context unification and traversal equations. In *RTA'01*, volume 2051 of *LNCS*, pages 167–184, 2001.
18. Mitchell P. Marcus, Donald Hindle, and Margaret M. Fleck. D-theory: Talking about talking about trees. In *Proceedings of the 21st ACL*, pages 129–136, 1983.
19. Joachim Niehren and Alexander Koller. Dominance constraints in context unification. In *Logical Aspects of Computational Linguistics (1998)*, volume 2014 of *LNAI*, 2001.
20. Joachim Niehren and Mateu Villaret. On lambda binding, parallelism constraints and context unification, 2002. Available at `http://www.ps.uni-sb.de/Papers`.
21. J. Rogers and K. Vijay-Shanker. Obtaining trees from their descriptions: An application to tree-adjoining grammars. *Computational Intelligence*, 10:401–421, 1994.
22. M. Schmidt-Schauß. A decision algorithm for stratified context unification. Technical Report F-rep.-12, FB Informatik, J.W. Goethe Universität Frankfurt, 1999.
23. Manfred Schmidt-Schauß and Klaus U. Schulz. Solvability of context equations with two context variables is decidable. In *CADE-16*, LNAI, pages 67–81, 1999.
24. Klaus U. Schulz. Makanin's algorithm for word equations- two improvements and a generalization. In *Proceedings of the First International Workshop of Word Equations and Related Topics*, volume 572 of *LNCS*, Tübingen, Germany, 1992.
25. J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81, 1967.