Set Constraints In Computational Linguistics — Solving Tree Descriptions —

Denys Duchier
University of the Saarland
duchier@ps.uni-sb.de

June 30, 1999

Computational linguistics has always been a priviledged application domain for constraint programming. While attention has traditionally focused on feature constraints, set constraints are now emerging as affording especially elegant and concise declarative formulations that naturally obtain very efficient operational semantics using constraint propagation.

In [DG99, DN99] we show how to reformulate, in terms of set constraints, the problem of finding minimal models of tree descriptions expressed in an extended version of dominance logic. In [Duc99] we provide an account of parsing, in the framework of dependency grammar, using set constraints, that is both a succinct declarative specification of the problem as a CSP and a very efficient implementation when regarded as a concurrent constraint program. Both applications can be viewed as highly combinatorial configuration problems and set constraints allow elegant declarative specifications that are also efficient constraint programs for enumerating solutions.

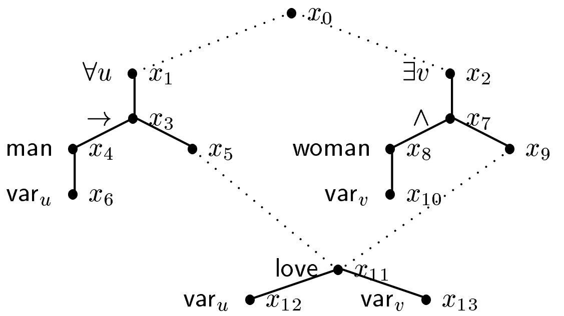
In this paper, we propose to describe our application of set constraints to the problem of finding solutions of tree descriptions. The sets that we consider are finite sets of non-negative integers, for which very efficient constraint programming support has been developed in the programming language Oz [Smo95, MC98] for the past 3 years.

1 Tree Descriptions in Linguistics

In computational linguistics it is frequently useful to manipulate descriptions of trees rather than trees themselves. [MHF83] introduced D-Theory which permited to speak about trees in terms of the *domination* relation rather than the parent relation. The formal framework was elaborated in [RVS92, BRVS95]. Since then, tree descriptions have found applications in many areas of computational linguistics: in [MHF83] for deterministic parsing, in [VS92, RVSW95] for tree-adjoining and D-tree grammars, in [Mus95] for underspecification in semantics and scope ambiguities, which [EKNR99] extended to parallelism and ellipsis, and in [GW98] for discourse representation. Yet little was known concerning the efficient computational treatment of tree descriptions.

Recently, two algorithms were proposed for solving dominance constraints: [KNT98] proposed a saturation-based method motivated by an application to underspecified semantics, while [DG99] described an encoding into set constraints for a treatment of discourse. We propose to describe the declarative axiomatization into set constraints first proposed in [DG99]. This axiomatization has a direct computational reading as a concurrent constraint program that implemements a very efficient solver.

Tree Descriptions with Dominance Constraints. Dominance constraints describe relations between the nodes of a tree. [KNT98] defines them as conjunctions of formulae $x \lhd^* y$ stating that a node x dominates a node y,[1] and $x : f(y_1 \dots y_n)$ stating that x is a node labeled with n-ary symbol f and a sequence of immediate daugthers $y_1, \dots y_n$. Dominance constraint have trees as models, i.e. every variable denotes a node of a tree. Figure 1 displays an example of a tree description with dominance constraints: it shows both a formula and a graphical depiction of its constraints.



$$
\begin{array}{l}
x_0 \lhd^* x_1 \wedge x_0 \lhd^* x_2 \wedge \\
x_1{:}\forall u(x_3) \wedge x_2 : \exists v(x_7) \wedge \\
x_3{:}\mathsf{imply}(x_4, x_5) \wedge x_4{:}\mathsf{man}(x_6) \wedge \\
x_7{:}\mathsf{and}(x_8, x_9) \wedge x_8{:}\mathsf{woman}(x_{10}) \wedge \\
x_6{:}\mathsf{var}_u \wedge x_{10}{:}\mathsf{var}_v \wedge \\
x_5 \lhd^* x_{11} \wedge x_9 \lhd^* x_{11} \wedge \\
x_{11}{:}\mathsf{love}(x_{12}, x_{13}) \wedge \\
x_{12}{:}\mathsf{var}_u \wedge x_{13}{:}\mathsf{var}_v
\end{array}
$$

Figure 1: Under specified semantic representation of *“every man loves a woman”*

Solving Dominance Constraints. Since solving dominance constraints is NP-complete [KNT98], we cannot expect any polynomial algorithm for it. A naïve generate and test method would enumerate the alternatives given by the formula $\forall x \forall y : x \lhd^* y \vee y \lhd^* x \vee x \parallel y$. This is clearly not practical: in Figure 1 there are 14 variables and hence $3^{14*14} = 3^{196}$ possible relationships. Instead our technique takes full advantage of constraint propagation to realize effective model elimination.

Application to Semantic Underspecification. Semantic underspecification aims to represent possible meanings of a natural language utterance in an

[1] x is equal to, or and ancestor of y

underspecified manner. The goal is to avoid combinatorial explosion raised by semantic ambiguities during semantic processing.

A prototypical scope-ambiguous sentence is:

$$\textit{Every man loves a woman} \tag{1}$$

Its first reading is *"For every man u, there is a woman v whom u loves"*:

$$\forall u(\mathsf{man}(u) \rightarrow \exists v(\mathsf{woman}(v) \wedge \mathsf{love}(u,v))) \tag{2}$$

while its second reading is *"There is a woman v whom every man u loves"*:

$$\exists v(\mathsf{woman}(v) \wedge \forall u(\mathsf{man}(u) \rightarrow \mathsf{love}(u,v))) \tag{3}$$

One might find the second reading (3) less plausible at first sight. But once (1) is continued by (4) only this second reading remains valid.

$$\textit{Her name is Mary} \tag{4}$$

In (Fig 1), an underspecified representation of the semantics of example (1) is given in terms of dominance constraints. The readings of (1) regarded as syntactic representations of predicate logic formulae correspond to the minimal solutions of the constraints depicted in Fig 1.

2 Extended Dominance Constraints

We will consider the extended language of dominance constraints addressed by [DG99] and formally examined in [DN99]. Its formulae are given by the abstract syntax:

$$\begin{array}{rcl} \phi & ::= & x\, R\, y \;\mid\; x : f(y_1, \ldots, y_n) \\ R & ::= & = \mid \lhd^+ \mid \rhd^+ \mid \| \mid\; R \cup R \;\mid\; R \cap R \;\mid\; \neg R \end{array}$$

where $\lhd^+$ indicates strict dominance and $\|$ disjointness. The classical dominance relation is defined by $\lhd^* \equiv\; = \cup \lhd^+$. A model M of a description ϕ is a tree T together with an interpretation I mapping each variable in ϕ to a node in T.

When regarded from a specific node, a tree is divided into 5 regions: (1) the current node, (2) the nodes above, (3) the nodes below, (4) the nodes to the left, and (5) the nodes to the right. This is illustrated in (Fig 2).

For the purposes of this presentation, we will aggregate the set of nodes to the left and to the right, and call the result the *side set*. A similar treatment can trivially be developed that retains the distinction; such a treatment would support precedence constraints.

This partitioning of the tree T into 4 disjoint sets of nodes also partitions the variables of ϕ interpreted by T. This idea forms the foundation of our technique. We write $\mathsf{eq}(N)$ for the set of variables interpreted by node N, $\mathsf{up}(N)$, $\mathsf{down}(N)$ and $\mathsf{side}(N)$ for the set of variables interpreted by nodes strictly above, below or to the side of N.

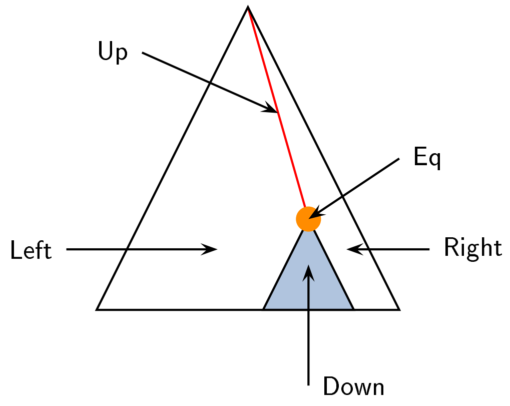


Figure 2: Each node sees a different partition of the tree

3 Solving Descriptions Using Set Constraints

As we described in [DG99], the problem of enumerating the minimal models of a tree description ϕ can be encoded as a constraint satisfaction problem on sets. Following our presentation in [DN99], we describe the technique by means of an encoding scheme $[\![\phi]\!]$ which turns a tree description ϕ into a CSP using set constraints:

$$[\![\phi]\!] = [\![\phi]\!]_0 \wedge [\![\phi]\!]_1 \tag{5}$$

$[\![\phi]\!]_0$ produces the well-formedness constraints that ensure that the models are trees, and $[\![\phi]\!]_1$ forms the additional constraints required for these trees to be models of ϕ.

3.1 Representation And Well-formedness Constraints

For each variable x in ϕ, we represent its interpretation N^x by a term:

$$N^x \quad =_{\text{def}} \quad \begin{bmatrix} \mathsf{eq} & : & N^x_{\mathsf{eq}} \\ \mathsf{up} & : & N^x_{\mathsf{up}} \\ \mathsf{down} & : & N^x_{\mathsf{down}} \\ \mathsf{side} & : & N^x_{\mathsf{side}} \\ \mathsf{eqdown} & : & N^x_{\mathsf{eqdown}} \\ \mathsf{equp} & : & N^x_{\mathsf{equp}} \\ \mathsf{label} & : & N^x_{\mathsf{label}} \\ \mathsf{parent} & : & N^x_{\mathsf{parent}} \end{bmatrix}$$

Where, N^x_{eq}, N^x_{up}, N^x_{down}, N^x_{side}, N^x_{eqdown}, N^x_{equp} are sets, and N^x_{label}, N^x_{parent} are terms refering to other nodes. The sets N^x_{eq}, N^x_{up}, N^x_{down}, N^x_{side} partition the set

$\mathcal{V}$ of variables of ϕ according to the position of their respective interpretations with respect to N^x:

$$\mathcal{V} = N^x_{\mathsf{eq}} \uplus N^x_{\mathsf{down}} \uplus N^x_{\mathsf{up}} \uplus N^x_{\mathsf{side}} \tag{6}$$

In [DN99], we improve constraint propagation by introducing N^x_{eqdown} and N^x_{equp} as explicit intermediate results:

$$\mathcal{V} = N^x_{\mathsf{eqdown}} \uplus N^x_{\mathsf{up}} \uplus N^x_{\mathsf{side}} \tag{7}$$

$$\mathcal{V} = N^x_{\mathsf{equp}} \uplus N^x_{\mathsf{down}} \uplus N^x_{\mathsf{side}} \tag{8}$$

N^x_{eqdown} and N^x_{equp} are defined by

$$N^x_{\mathsf{eqdown}} = N^x_{\mathsf{eq}} \uplus N^x_{\mathsf{down}} \tag{9}$$

$$N^x_{\mathsf{equp}} = N^x_{\mathsf{eq}} \uplus N^x_{\mathsf{up}} \tag{10}$$

Importantly, since x is interpreted by N^x it must be in N^x_{eq}:

$$x \in N^x_{\mathsf{eq}} \tag{11}$$

Models of descriptions are trees: therefore, we must ensure that the interpretations (N^{x_i}) of the variables (x_i) of ϕ are arranged in a tree shape. This is realized by stating a well-formedness constraint. It rests on the following observation: in a tree, two nodes N^x and N^y must stand in one of 4 mutually exclusive relationships: N^x is equal to N^y ($N^x = N^y$), N^x strictly dominates N^y ($N^x \lhd^+ N^y$), N^y strictly dominates N^x ($N^x \rhd^+ N^y$), or they occur in disjoint subtrees ($N^x \parallel N^y$).

Thus, for every two variables x and y of ϕ, we introduce a choice variable $C^{xy} \in \{1, 2, 3, 4\}$ to explicitly represent this choice. The well-formedness condition is expressed by the following 4 clauses:

$$N^x = N^x \wedge C^{xy} = 1 \;\vee\; C^{xy} \neq 1 \wedge N^x \neq N^y \tag{12}$$

$$N^x \lhd^+ N^y \wedge C^{xy} = 2 \;\vee\; C^{xy} \neq 2 \wedge N^x \neg\lhd^+ N^y \tag{13}$$

$$N^y \lhd^+ N^x \wedge C^{xy} = 3 \;\vee\; C^{xy} \neq 3 \wedge N^y \neg\lhd^+ N^x \tag{14}$$

$$N^x \parallel N^y \wedge C^{xy} = 4 \;\vee\; C^{xy} \neq 4 \wedge N^x \neg\parallel N^y \tag{15}$$

In logic programming, disjunction is given the operational semantics of a choice point. That would be here completely inappropriate and would lead to very poor performance. Instead, we take advantage of the fact that, in concurrent constraint programming, disjunction can be implemented by a concurrent agent that continuously and speculatively investigates all alternatives. It discards alternatives that become inconsistent. When only one alternative remains, the agent *commits*, i.e. it is replaced by this alternative. Precise semantics can be found in [Smo95].

The abstract constraints used in clauses (12), (13), (14) and (15) are defined as follows:

$$
\begin{aligned}
N^x = N^y \ &\equiv \ \text{this is just unification} && (16)\\
N^x \neq N^y \ &\equiv \ N^x_{\mathsf{eq}} \parallel N^y_{\mathsf{eq}} && (17)\\
N^x \lhd^+ N^y \ &\equiv \ N^x_{\mathsf{down}} \supseteq N^y_{\mathsf{eqdown}} \wedge N^x_{\mathsf{equp}} \subseteq N^y_{\mathsf{up}} \wedge N^x_{\mathsf{side}} \subseteq N^y_{\mathsf{side}} && (18)\\
N^x \neg\lhd^+ N^y \ &\equiv \ N^x_{\mathsf{eq}} \parallel N^y_{\mathsf{up}} \wedge N^x_{\mathsf{down}} \parallel N^y_{\mathsf{eq}} && (19)\\
N^x \parallel N^x \ &\equiv \ N^x_{\mathsf{eqdown}} \subseteq N^y_{\mathsf{side}} \wedge N^y_{\mathsf{eqdown}} \subseteq N^x_{\mathsf{side}} && (20)\\
N^x \neg\parallel N^y \ &\equiv \ N^x_{\mathsf{eq}} \parallel N^y_{\mathsf{side}} \wedge N^y_{\mathsf{eq}} \parallel N^x_{\mathsf{side}} && (21)
\end{aligned}
$$

We can now state the translation scheme for the well-formedness constraint:

$$
[\![\phi]\!]_0 = \bigwedge_{x \in \mathcal{V}} (7, 8, 9, 10, 11)_x \bigwedge_{x,y \in \mathcal{V}} (12, 13, 14, 15)_{xy} \tag{22}
$$

3.2 Translation of Problem Specific Constraints

We now explicate how $[\![\phi]\!]_1$ forms the additional problem specific constraints that further limit the admissibility of well-formed solutions. The encoding is given by clauses (23,24,32).

$$
[\![\phi \wedge \phi']\!]_1 = [\![\phi]\!]_1 \wedge [\![\phi']\!]_1 \tag{23}
$$

A very nice consequence of the introduction of choice variables C^{xy} is that any dominance constraint $x \; R \; y$ can be translated as a restriction on the possible values of C^{xy}. For example, $x \lhd^* y$ can be encoded as $C^{xy} \in \{1, 2\}$. More generally:

$$
[\![x \; R \; y]\!]_1 = C^{xy} \in [\![R]\!]_2 \tag{24}
$$

where $[\![R]\!]_2$ turns an extended dominance relationship into a set of possible values for the choice variable.

$$
\begin{aligned}
[\![=]\!]_2 &= \{1\} && (25)\\
[\![\lhd^+]\!]_2 &= \{2\} && (26)\\
[\![\rhd^+]\!]_2 &= \{3\} && (27)\\
[\![\parallel]\!]_2 &= \{4\} && (28)\\
[\![R \cup R']\!]_2 &= [\![R]\!]_2 \cup [\![R']\!]_2 && (29)\\
[\![R \cap R']\!]_2 &= [\![R]\!]_2 \cap [\![R']\!]_2 && (30)\\
[\![\neg R]\!]_2 &= \{1, 2, 3, 4\} \setminus [\![R]\!]_2 && (31)
\end{aligned}
$$

Finally, the labeling constraint $x : f(y_1, \ldots, y_n)$ requires a more complicated treatment. It names the parent x, the constructor f, and the immediate daughters y_1 through y_n.

$$[\![x : f(y_1, \ldots, y_n)]\!]_1 = \begin{array}{ll} & N^x_{\mathsf{label}} = f(N^{y_1}, \ldots, N^{y_n}) \\ \wedge_{i=1}^{i=n} & N^{y_i}_{\mathsf{parent}} = N^x \\ \wedge & N^x_{\mathsf{down}} = N^{y_1}_{\mathsf{eqdown}} \uplus \ldots \uplus N^{y_n}_{\mathsf{eqdown}} \\ \wedge_{i=1}^{i=n} & N^{y_i}_{\mathsf{up}} = N^x_{\mathsf{equp}} \end{array} \tag{32}$$

3.3 Searching for Solutions of the CSP

Given the translation scheme above, the models of ϕ can be found by enumerating the assignments to the choice variables $(C^{xy})_{x,y\in\mathcal{V}}$ consistent with $[\![\phi]\!]$. Thus, the problem of finding minimal models of a tree description ϕ is reduced to that of applying a *labeling strategy* to the choice variables. In practice, we have used *first-fail* with very good results. [DN99] suggests a better informed and more economical alternative.

References

[BRVS95] 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.

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

[DN99] Denys Duchier and Joachim Niehren. Solving dominance constraints with finite set constraint programming. Technical report, Universität des Saarlandes, Programming Systems Lab, 1999. `http://www.ps.uni-sb.de/Papers/abstracts/DomCP99.html`.

[Duc99] Denys Duchier. Axiomatizing dependency parsing using set constraints. In *Sixth Meeting on Mathematics of Language (MOL6)*, Orlando, Florida, July 1999. `http://www.ps.uni-sb.de/~duchier/drafts/mol6.ps.gz`.

[EKNR99] Markus Egg, Alexander Koller, Joachim Niehren, and Peter Ruhrberg. Constraints over lambda structures, antecedent contained deletion, and quantifier identities. Submitted. `http://www.coli.uni-sb.de/~koller/papers/acd.html`, 1999.

[GW98] Claire Gardent and Bonnie Webber. Describing discourse semantics. In *Proceedings of the 4th TAG+ Workshop*, Philadelphia, 1998. University of Pennsylvania.

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

[MC98] The Mozart Consortium. The Mozart Programming System, 1998. http://www.mozart-oz.org/.

[MHF83] 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.

[Mus95] R.A. Muskens. Order-Independence and Underspecification. In J. Groenendijk, editor, *Ellipsis, Underspecification, Events and More in Dynamic Semantics*. DYANA Deliverable R.2.2.C, 1995. `http://www.ims.uni-stuttgart.de/ftp/pub/papers/DYANA2/95copy/R2.2.C/Muskens.ps.gz`.

[RVS92] James Rogers and K. Vijay-Shanker. Reasoning with descriptions of trees. In *Proc. ACL*, 1992.

[RVSW95] Owen Rambow, K. Vijay-Shanker, and David Weir. D-tree grammars. In *Proceedings of ACL'95*, pages 151–158, MIT, Cambridge, 1995.

[Smo95] Gert Smolka. The Oz Programming Model. In *Computer Science Today*, volume 1000 of *LNCS*, pages 324–343, 1995.

[VS92] K. Vijay-Shanker. Using descriptions of trees in a tree adjoining grammar. *Computational Linguistics*, 18:481–518, 1992.