Parsing with Tree Descriptions: a constraint–based approach*

Denys Duchier duchier@ps.uni-sb.de
Stefan Thater stth@coli.uni-sb.de

Abstract

In previous work [7, 8] it was shown that a constraint-based treatment of tree-descriptions results in a simple and tractable implementation. However, that approach treated only the conjunctive fragment. Therefore it could not be directly applied to parsing with tree description-based grammars where lexical ambiguity gives rise to disjunction.

In this paper, we extend the previous approach in two ways. First, we introduce the formalism of 'electrostatic tree descriptions', which combines dominance logic with a notion of polarities and permits a convenient characterization of admissible syntactic structures. Second, we extend this idea to disjunctive systems of descriptions sufficient to account for lexical ambiguities. Finally, we exhibit an encoding that turns parsing into a constraint satisfaction problem (CSP) solvable by constraint programming.

1 Introduction

Traditionally, syntax is about trees: phrase structure trees. In computational syntax however, it has variously been argued that *descriptions of trees* offer an alternative with significant advantages. [11] shows that tree descriptions support incremental syntactic processing; [16] uses them to provide a fully monotone treatment of unification-based Tree Adjoining Grammar; [12] finds them necessary for combining semantic with syntactic under-specification; and [13] advocates their use for a uniform treatment of modification and complementation in a TAG-like framework.

Common to all these proposals, is the idea that descriptions permit a certain level of under-specification necessary for an adequate and elegant treatment of the phenomena under consideration. The under-specification is in two guises. First, node variables are used rather than nodes. In particular, two distinct node variables may refer to one and the same node. Second, arbitrary dominance statements are allowed (as opposed to only strict dominance) and permit to under-specify the structural relation between two node variables and therefore, indirectly, between the tree nodes these variables denote.

*The research presented in this paper was funded by the DFG in SFB–378, Project C2 (LISA).

The tree descriptions used by computational linguists are expressions in a tree logic e.g. [2, 1]: they are logical formulae which, if consistent, are satisfied by one and usually many more models in that logic. These models are the syntactic trees the linguist is searching for.

In terms of processing however, tree descriptions are often treated as data-structures and combined together using various composition operations. Under such a view, parsing is dealt with by means of standard techniques e.g. an Earley style parsing algorithm in [13] and automaton-based parsing in [3]. The price to pay is that node variables are no longer variables: they are nodes which under certain circumstances can be merged.

Under the logical view on the other hand, variables are variables and parsing is a model-generation problem: given a logical formula ϕ, how can we generate the minimal models that satisfy it? As [4] remarks, naive tableau-based approaches such as [14, 4] can lead to combinatorial explosion in the face of disjunction inherent in the theory of trees. In fact, [9] shows that solving purely conjunctive descriptions is NP-hard. In spite of this theoretical result, practical algorithms are needed to enumerate solutions as efficiently as possible. In [7, 8] we described a constraint-based approach by encoding into finite set constraints. This technique resulted in a simple and efficient implementation in the concurrent constraint programming language Oz [10, 15].

In this paper, we build on the approach of [7, 8] and adapt it to parsing with description-based grammars. First, we introduce the formalism of 'electrostatic tree descriptions', which combines dominance logic with a notion of polarities and permits a convenient characterization of admissible syntactic structures. Second, we extend the formal framework with a restricted form of disjunction to account for lexical ambiguity, and introduce a computationally judicious notion of model. Third, we provide an encoding, in the axiomatic style, that turns a parsing problem into a constraint satisfaction problem (CSP) solvable by constraint programming.

Section 2 presents and motivates the tree logic used for writing the grammar; it then outlines a grammar fragment in that logic. Section 3 introduces the semantics of tree descriptions and develops a formal model of our parsing framework. In particular, Section 3.3 extends the semantics to disjunctive systems of descriptions. Section 4 turns to the computational aspect and develops a corresponding constraint model. We precisely define an encoding scheme that turns a problem expressed in our formal model into a CSP in our constraint model. Section 5 discusses preliminary results obtained with our prototype implementation and outlines directions of future development.

2 Tree Description Grammar

Like lexicalized tree adjoining grammar (LTAG) the basic component of a tree description grammar is a lexicon that maps words to fragments of phrase structure (elementary trees). The task of parsing is to combine these fragments into a single coherent tree.

In order to account for modification or movement phenomena it is often necessary to insert one fragment into the middle of another one. In TAG this is accomplished by the non-monotonic operation of adjunction: A node is split into an upper and a lower

part and a fragment is inserted into this hole. In our approach, we take up a different idea. Since we use descriptions of trees there is no need to split a node. We can already provide holes in the lexical fragments. Into these holes other fragments may be inserted incrementally and monotonically. (Fig 1) shows a small lexicon. Description α_5 contains a hole between the VP nodes which is bridged by a dominance link (the dotted line). Into this hole, we might insert, for example, the VP-modifier α_4.

Another benefit of using descriptions is that one obtains greater flexibility: we do not require that the two nodes related by a dominance link be assigned the same symbol (category), which allows alternative analyses for e.g. extraction phenomena. Unlike in TAG, where the filler-gap dependency is localized to the verb, we provide lexical entries for topicalized NPs. For example, the lexical entry α_1 for *Who* says that this tree can act as a filler of a sentence gap (we use ϵ to refer to the empty word).

In this sense, our approach is closely related to d-tree grammar (DTG) presented in [13]. However, DTG distinguishes two different tree combining operations called subsertion and sister-adjunction. This distinction is motivated by linguistic considerations: it mirrors the distinction between complementation and modification. Subsertion always corresponds to complementation and sister-adjunction to modification. From a formal perspective, this distinction is not necessary: modification phenomena can equally well be treated by the subsertion operation. From a practical point of view, the sister-adjunction operation has the disadvantage that it changes the arity of nodes and would make an implementation less efficient. Therefore, we do not provide an operation that corresponds to sister-adjunction.

Another difference with traditional DTGs is that we decorate nodes with *polarities* in order to control the way in which tree descriptions may combine: A $-$ identifies a *hole*, which can be thought of as an open valency, and a $+$ identifies a *plug*. Neutral nodes are decorated with a 0 and are normally used for lexical nodes. Polarities offer a simple way to precisely characterize the desired models, i.e. the syntactic structures licensed by the grammar.

Successful parsing requires every hole to be plugged and every plug to be used. (Fig 2) illustrates the idea with a parse tree for the sentence *Who did Mary see*. The pairings of plugs with holes are indicated by dashed lines. As can be seen from this example, we assume that the root of each fragment has a positive polarity. Therefore, we must introduce an extra sentential root node with negative polarity in order to cancel the polarity of the topmost node.

2.1 Electrostatic Tree Descriptions

We now introduce a formal language for writing the elementary tree descriptions of our lexicon. We assume three infinite and disjoint sets $Vars^0$, $Vars^+$ and $Vars^-$ of variables (resp. neutral, positively and negatively charged variables) and a lattice $\mathcal{L}$ of labels. An electrostatic tree description is a formula of the form:

$$\phi \quad ::= \quad x\,R\,y \mid x : \langle y_1, \ldots, y_n \rangle \mid x \leftarrow \ell \mid \phi_1 \wedge \phi_2$$

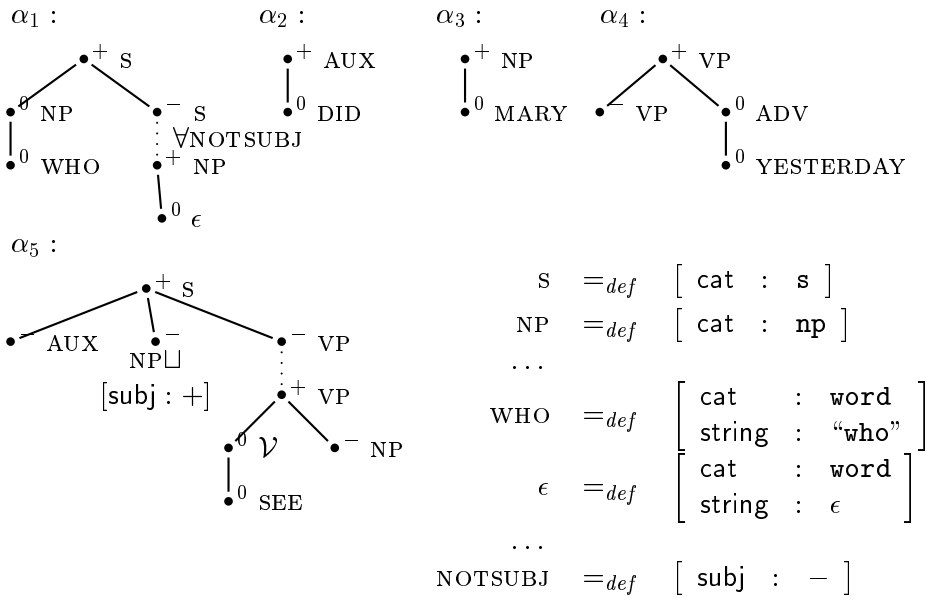

Figure 1: Lexicon entries for *who*, *did*, *Mary*, *see* and *yesterday*.

where R is any boolean combination of $=$, $\lhd^+$, $\rhd^+$, $\prec$, $\succ$ — where $\lhd^+$ represents strict dominance and $\prec$ precedence — and expresses the relative position of x and y. R may be formed according to the following grammar:

$$R \quad ::= \quad = \mid \lhd^+ \mid \rhd^+ \mid \prec \mid \succ \mid R_1 \cup R_2 \mid R_1 \cap R_2 \mid \neg R$$

$x : \langle y_1, \ldots, y_n \rangle$ states that x has the (y_i) as immediate daughters. Finally, we write $x \leftarrow \ell$ (for $\ell \in \mathcal{L}$) to indicate that x is labeled by ℓ. The purpose of ℓ is to encapsulate all the grammatical features, such as *category* or *agreement*. In this manner, they are abstracted out and we can study the grammatical framework independently of any specific commitment to these features. In this sense, our framework is parametrized by an arbitrary lattice $\mathcal{L}$ of e.g. AVMs.

In our examples we assume a product of lattices denoting AVMs with features cat, string and subj. Useful abbreviations, such as S and NP, are defined in (Fig 1), where we follow the standard convention that $\top$-valued[1] features are omitted. Thus, label S denotes the most general AVM with value s at feature cat.

The full language allows literals of the form $\forall(x..y) \leftarrow \ell$ and $\exists(x..y) \leftarrow \ell$. These are called *insertion constraints* which are similar to the selective and obligatory adjunction constraints in TAG: a universal insertion constraint $\forall(x..y) \leftarrow \ell$ requires that all nodes that lie on the path properly between x and y are assigned a label compatible with ℓ whereas an existential constraint $\exists(x..y) \leftarrow \ell$ requires the existence of such a node. For example, the universal insertion constraint in α_1 requires that no node on the path between the gap and the filler is marked as subject, which rules out certain ungrammatical

[1]We write $\top$ for the top, i.e. most general, element of a lattice.

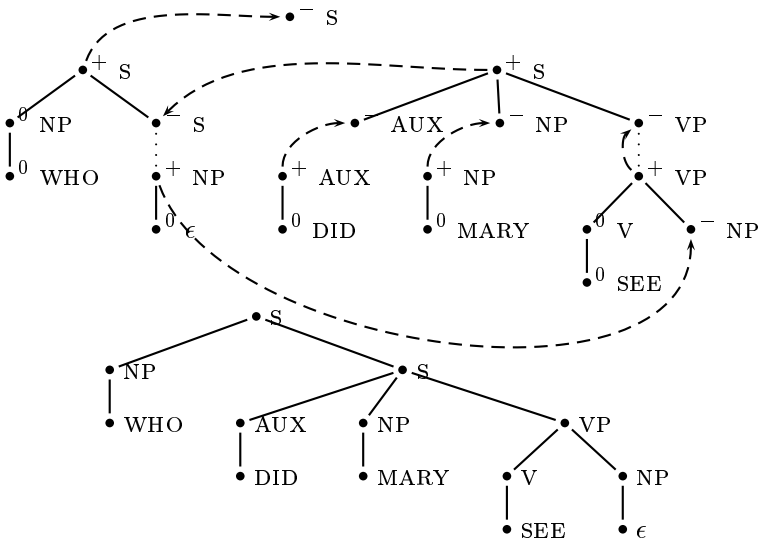

Figure 2: Who did Mary see?

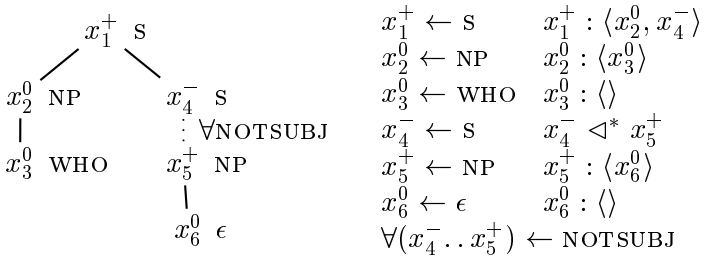

Figure 3: Lexical entry for topicalized *Who*.

constructions. In this paper, we shall not account for insertion constraints, although the treatment can be extended simply.

2.2 Lexicon, Lexical Entries

Let us first explain how to formulate tree descriptions in terms of electrostatic formulae. Take, for example, the description depicted in (Fig 3): Immediate dominance relations (solid lines) are formulated using $x : \langle y_1, \ldots, y_n \rangle$ whereas dominance relations (dotted lines) are expressed using $\lhd^*$, which is just an abbreviation for $= \cup \lhd^+$. Since immediate dominance relations always constrain the ordering of the daughters, there is no need to make explicit use of the precedence relations $\prec$ and $\succ$ within the descriptions of the lexical entries. However, they will become important when we combine lexical trees to larger trees.

The polarities decorating the variables in (Fig 3) should be viewed as part of their name. They visually indicate whether the variable comes from $Vars^+$, $Vars^-$, or $Vars^0$.

Since these sets are disjoint, it is not possible for two occurrences of a variable to be decorated with different polarities.

A lexical entry is an AVM (or triple) that relates a string s, a description ϕ and an anchor node variable x_0 occurring in ϕ:

$$\begin{bmatrix} \mathsf{string} & : & s \\ \mathsf{formula} & : & \phi(x_0) \\ \mathsf{anchor} & : & x_0 \end{bmatrix}$$

The lexicon is simply a set of lexical entries. We write attribute access in functional notation: for a lexical entry e, the functions $\mathsf{formula}(e)$, $\mathsf{anchor}(e)$ and $\mathsf{string}(e)$ return the values of the corresponding attributes of e.

2.3 Grammar Framework

A lexicalized tree description grammar is a triple $G = \langle \mathcal{L}, \mathcal{E}, \mathcal{S} \rangle$, where $\mathcal{L}$ is a lattice of labels, $\mathcal{E}$ is a set of lexical entries, and $\mathcal{S} \in \mathcal{L}$ is a start symbol.

Parsing. As long as a tree description grammar associates each word with exactly one description, it is straightforward to adapt the technique of [7] for parsing: building a parse tree for an input sentence $w_1 \ldots w_n$ can be done simply by solving the constraint:

$$\phi = \bigwedge_{i=1}^{n} \mathsf{formula}(e_i) \wedge \bigwedge_{i=2}^{n} \mathsf{anchor}(e_{i-1}) \prec \mathsf{anchor}(e_i) \wedge x_R^- \leftarrow \mathcal{S} \qquad (1)$$

where e_i is the lexicon entry for word w_i and $x_R^- \in \mathit{Vars}^-$ is the root node which we previously argued must be introduced in order to cancel the polarity of the topmost node. We assume that the formulae $\mathsf{formula}(e_i)$ do not share variables and that x_R^- is a fresh variable not occuring in any $\mathsf{formula}(e_i)$.

However, grammars for natural language typically assign more than one description to a word, and the approach developed in [7] must be extended to allow *disjunctions* of tree descriptions.

3 Formal Model

In this section, we develop the formal framework of our approach to parsing. The semantics of tree descriptions are given by interpretation over finite trees: we begin with definitions, where the intuition is to identify a node in a feature tree with the sequence of features that must be followed from the root to arrive at this node. In Section 3.2 we make precise what a model is, and in Section 3.3 we extend our account to disjunctive systems of descriptions.

3.1 Preliminary Definitions

Let F be a set of symbols called *features*. An *F-path* is a finite, possibly empty, sequence of features, i.e. a word of F^*. We will use the letter π for paths and f for features, and we write $\pi_1\pi_2$ for the concatenation of paths π_1 and π_2. We write ϵ for the empty path.

An *F-feature tree* T is a *prefix closed* set of F-paths, i.e. such that $\pi f \in T \;\Rightarrow\; \pi \in T$. The paths of a feature tree are also called *nodes*.

If there exists a partial order $\preceq$ over F, a *left-saturated* F-feature tree is defined as an F-feature tree such that $\pi f_1 \in T \;\Rightarrow\; \pi f_2 \in T$ $(\forall f_2 \preceq f_1)$. In a left saturated feature tree T, we can define the sets of nodes that are equal, below, above, to the left, and to the right of a node π as follows:

$$\mathrm{eq}(\pi) = \{\pi\} \tag{2}$$
$$\mathrm{down}(\pi) = \{\pi\pi' \in T \mid \pi' \neq \epsilon\} \tag{3}$$
$$\mathrm{up}(\pi) = \{\pi_1 \in T \mid \exists \pi_2 \neq \epsilon \;\; \pi = \pi_1\pi_2\} \tag{4}$$
$$\mathrm{left}(\pi) = \{\pi' \in T \mid \exists \pi_1, \pi_2, \pi_2', f_i, f_j \;\; f_i \prec f_j \;\; \pi = \pi_1 f_j \pi_2 \;\; \pi' = \pi_1 f_i \pi_2'\} \tag{5}$$
$$\mathrm{right}(\pi) = \{\pi' \in T \mid \exists \pi_1, \pi_2, \pi_2', f_i, f_j \;\; f_i \succ f_j \;\; \pi = \pi_1 f_j \pi_2 \;\; \pi' = \pi_1 f_i \pi_2'\} \tag{6}$$

where $f_i \prec f_j \;\equiv f_i \preceq f_j \wedge f_i \neq f_j$.

An *ordered tree* is simply a left-saturated feature tree, where the features are positive integers. In an ordered tree T, we write $\pi : \langle \pi_1, \ldots, \pi_n \rangle$ to abbreviate the fact that $\pi_i \in T$ and $\pi_i = \pi i \;\; (1 \leq i \leq n)$ and $\pi(n+1) \notin T$.

3.2 Models

$$M \models \phi_1 \wedge \phi_2 \;\equiv\; M \models \phi_1 \;\wedge\; M \models \phi_2 \tag{7}$$
$$M \models x \; R_1 \cap R_2 \; y \;\equiv\; M \models x \; R_1 \; y \;\wedge\; M \models x \; R_2 \; y \tag{8}$$
$$M \models x \; R_1 \cup R_2 \; y \;\equiv\; M \models x \; R_1 \; y \;\vee\; M \models x \; R_2 \; y \tag{9}$$
$$M \models x \; \neg R \; y \;\equiv\; M \models x \; (= \cup \lhd^+ \cup \rhd^+ \cup \prec \cup \succ \setminus R) \; y \tag{10}^\dagger$$
$$M \models x = y \;\equiv\; I(x) = I(y) \tag{11}$$
$$M \models x \lhd^+ y \;\equiv\; I(x) \in \mathrm{up}(I(y)) \tag{12}$$
$$M \models x \rhd^+ y \;\equiv\; I(x) \in \mathrm{down}(I(y)) \tag{13}$$
$$M \models x \prec y \;\equiv\; I(x) \in \mathrm{left}(I(y)) \tag{14}$$
$$M \models x \succ y \;\equiv\; I(x) \in \mathrm{right}(I(y)) \tag{15}$$
$$M \models x : \langle y_1, \ldots, y_n \rangle \;\equiv\; I(x) : \langle I(y_1), \ldots, I(y_n) \rangle \tag{16}$$
$$M \models x \leftarrow \ell \;\equiv\; L(I(x)) \sqsubseteq \ell \tag{17}$$

Figure 4: Conditions for being a model of a description

[†] infix '\' is a difference operator, e.g. $= \cup \lhd^+ \cup \rhd^+ \cup \prec \cup \succ \;\setminus\; \lhd^+ \cup \succ \;\; =_{\mathrm{def}} \;\; = \cup \rhd^+ \cup \prec$

A model $M = \langle T, L, I \rangle$ of an electrostatic tree description ϕ consists of an ordered tree T, a labeling function L mapping nodes of T to elements of $\mathcal{L}$, and an interpretation I mapping each variable of ϕ to a node in T. M is a model if it satisfies conditions (7–17) in Figure 4, where $\ell_1 \sqsubseteq \ell_2$ states that ℓ_1 is a specialization of ℓ_2 in lattice $\mathcal{L}$.

We will distinguish two important classes of models: *free* and *saturated*. In a free model, a node of T interprets at most 1 variable. In a saturated model, a node of T interprets either no variable, precisely one neutral variable, or precisely two variables, 1 positive and 1 negative: every charged variable is *mated* with an anti-variable. We write $M \models_{\text{F}} \phi$ for a free model and $M \models_{\text{S}} \phi$ for a saturated model.

In an elementary tree description, the positive and negative charges indicate the *plugs* and *holes*. They constrain the way in which elementary trees may be plugged together to form larger syntactic constructions. Saturated models will serve to characterize completed parse trees.

3.3 Disjunctive Systems of Descriptions

We now demonstrate how the treatment developed for the conjunctive fragment can be extended to handle electrostatic tree descriptions with a restricted form of disjunction. We only aim to account for disjunctions arising from lexical ambiguity: for any given word, we may need to choose, in the lexicon, one of several possible lexical entries.

We consider the following idealization of the problem: we are given n words, and each must choose 1 out of m descriptions. We say that a disjunctive system of electrostatic tree descriptions is given by a matrix (ϕ_{ij}) and has the semantics:

$$\bigwedge_{i=1}^{i=n} \bigvee_{j=1}^{j=m} \phi_{ij} \tag{18}$$

The definition of a model can be extended to disjunctive matrices of descriptions by adding a function S which maps the index of a row to an index of a *selected* column:

$$\langle M, S \rangle \models (\phi_{ij}) \quad \equiv \quad M \models_{\text{S}} \bigwedge_{1 \leq i \leq n} \phi_{iS(i)} \tag{19}$$

where $\phi_{iS(i)}$ is called the selected disjunct of row i.

The task of parsing is thus to pick one description in each row of matrix (ϕ_{ij}) and to find a saturated model for the conjunction of these descriptions, i.e. to plug these elementary trees together so as to leave no unfilled hole and no unused plug.

If we further assume (a) that no two descriptions in (ϕ_{ij}) have any variable in common, and (b) that each ϕ_{ij} has at least one free model M_{ij}, then each non-selected ϕ_{ij} can be given this free model M_{ij}. Both conditions can easily be met in the application to parsing. Condition (a) may be achieved by appropriately renaming variables when looking up a lexical entry. Condition (b) merely requires a description to be "tree shaped."

We can now define the notion of a matrix model $\langle (M_{ij}), S\rangle$ for (ϕ_{ij}):

$$\begin{array}{lcll} \langle (M_{ij}), S\rangle \models (\phi_{ij}) & \equiv & M_{ij} \models_{\mathrm{F}} \phi_{ij} & j \neq S(i) \\ & \wedge & M_{1S(1)} = \cdots = M_{nS(n)} & (= M) \\ & \wedge & M \models_{\mathrm{S}} \bigwedge\limits_{1\leq i\leq n} \phi_{iS(i)} & \end{array} \tag{20}$$

In the following, we will assume that conditions (a) and (b) are met, and we will restrict our attention to the search for matrix models. The advantage over simply looking for a model that satisfies (19) is that we can obtain stronger propagation: now the constraints in all the descriptions ϕ_{ij} must be satisfied ... but not necessarily in the same model.

This is the trick! We have transformed a disjunctive problem into a conjunctive formulation: all constraints must be satisfied ... but not necessarily in the same model: the disjunctive aspect has been shifted to how we partition the models.

4 Constraint Model

We now develop a constraint model for the formal framework of Section 3: we introduce variables for the quantities and mappings it mentions, and formulate constraints on them that precisely capture the conditions stipulated by the formal model. Following our presentation in [8, 6], we describe the technique by means of an encoding scheme $[\![(\phi_{ij})]\!]$ which turns a matrix of descriptions (ϕ_{ij}) into a CSP:

$$[\![(\phi_{ij})]\!] = [\![(\phi_{ij})]\!]_0 \wedge [\![(\phi_{ij})]\!]_1 \tag{21}$$

$[\![(\phi_{ij})]\!]_0$ produces the well-formedness constraints that ensure that solutions represent minimal matrix models on tree domains, and $[\![(\phi_{ij})]\!]_1$ forms the additional constraints required for them to be models of (ϕ_{ij}).

4.1 Representation

We introduce variable $S(i) \in [1..m]$ to indicate the column selected in row i, $\mathsf{dom}(M_{ij})$ for the set of variables which is the domain of model M_{ij}, and $\mathsf{node}(x)$ for the representation of the interpretation $I(x)$ of x.

For each variable x, we introduce the variables $\mathsf{eq}(x)$, $\mathsf{down}(x)$, $\mathsf{up}(x)$, $\mathsf{left}(x)$, $\mathsf{right}(x)$, $\mathsf{eqdown}(x)$, $\mathsf{equp}(x)$, $\mathsf{side}(x)$, $\mathsf{daughters}(x)$, $\mathsf{mother}(x)$, $\mathsf{label}(x)$, and define $\mathsf{node}(x)$ as the 11-tuple of these variables.

4.2 Well-Formedness Constraints

Selection Constraint. In [5], we introduced the selection constraint $X = \langle V_1, \ldots, V_n\rangle[I]$ to indicate that X is equated with the Ith element in sequence $\langle V_1, \ldots, V_n\rangle$.[2] This can be very efficiently implemented as a constraint that additionally provides constructive disjunction semantics (lifting of information common to all remaining alternatives).

[2] X, V_1, ..., V_n, and I are all variables.

We write V for the domain of our saturated model, i.e. the variables in all selected descriptions:

$$V = V_1 \cup \cdots \cup V_n \tag{22}$$
$$V_i = \langle \mathsf{vars}(\phi_{i1}), \ldots, \mathsf{vars}(\phi_{im}) \rangle [S(i)] \tag{23}$$

We write B_{ij} for the boolean indicating whether ϕ_{ij} is selected, and make the classical identification of false with 0 and true with 1. We write $\mathsf{dom}(M_{ij})$ for the domain of model M_{ij}. Since we require models to be minimal, $\mathsf{dom}(M_{ij})$ is either V if ϕ_{ij} is selected or just $\mathsf{vars}(\phi_{ij})$ otherwise:

$$B_{ij} \equiv S(i) = j \tag{24}$$
$$\mathsf{dom}(M_{ij}) = \langle \mathsf{vars}(\phi_{ij}), V \rangle [B_{ij} + 1] \tag{25}$$

For each variable x in ϕ_{ij}, we pose, for notational convenience:

$$\mathsf{dom}(x) = \mathsf{dom}(M_{ij}) \tag{26}$$
$$\mathsf{sel}(x) = B_{ij} \tag{27}$$

Further, $\mathsf{eq}(x)$, $\mathsf{up}(x)$, $\mathsf{down}(x)$, $\mathsf{left}(x)$, $\mathsf{right}(x)$ stand for the sets of variables whose interpretations are in the corresponding positions relative to the interpretation of x, and write $\mathsf{mates}(x)$ for the other variables with the same interpretation:

$$\mathsf{eq}(x) = \{x\} \uplus \mathsf{mates}(x) \tag{28}$$
$$\mathsf{dom}(x) = \mathsf{eqdown}(x) \uplus \mathsf{up}(x) \uplus \mathsf{side}(x) \tag{29}$$
$$= \mathsf{equp}(x) \uplus \mathsf{down}(x) \uplus \mathsf{side}(x) \tag{30}$$

As demonstrated in [8], explicitly introducing these intermediate results affords greater propagation:

$$\mathsf{eqdown}(x) = \mathsf{eq}(x) \uplus \mathsf{down}(x) \tag{31}$$
$$\mathsf{equp}(x) = \mathsf{eq}(x) \uplus \mathsf{up}(x) \tag{32}$$
$$\mathsf{side}(x) = \mathsf{left}(x) \uplus \mathsf{right}(x) \tag{33}$$

We write $\mathcal{V}$ for the set of all variables in the matrix:

$$\mathcal{V} = \biguplus_{i,j} \mathsf{vars}(\phi_{ij}) \tag{34}$$

We define $\mathcal{V}^0 = \mathcal{V} \cap \mathit{Vars}^0$, $\mathcal{V}^+ = \mathcal{V} \cap \mathit{Vars}^+$ and $\mathcal{V}^- = \mathcal{V} \cap \mathit{Vars}^-$ for respectively the neutral, positive and negative variables in the matrix. A neutral variable has no mate (35). A charged variable has at most one mate: it has one mate iff its description is selected (36):

$$(x \in \mathcal{V}^0) \qquad |\mathsf{mates}(x)| = 0 \tag{35}$$
$$(x \notin \mathcal{V}^0) \qquad |\mathsf{mates}(x)| = \mathsf{sel}(x) \tag{36}$$

$$C^{xy} = 1 \ \wedge \ [\![x = y]\!]_3 \quad \vee \quad C^{xy} \neq 1 \ \wedge \ [\![x \neq y]\!]_3 \tag{40}$$
$$C^{xy} = 2 \ \wedge \ [\![x \lhd^+ y]\!]_3 \quad \vee \quad C^{xy} \neq 2 \ \wedge \ [\![x \neg\lhd^+ y]\!]_3 \tag{41}$$
$$C^{xy} = 3 \ \wedge \ [\![x \rhd^+ y]\!]_3 \quad \vee \quad C^{xy} \neq 3 \ \wedge \ [\![x \neg\rhd^+ y]\!]_3 \tag{42}$$
$$C^{xy} = 4 \ \wedge \ [\![x \prec y]\!]_3 \quad \vee \quad C^{xy} \neq 4 \ \wedge \ [\![x \neg\prec y]\!]_3 \tag{43}$$
$$C^{xy} = 5 \ \wedge \ [\![x \succ y]\!]_3 \quad \vee \quad C^{xy} \neq 5 \ \wedge \ [\![x \neg\succ y]\!]_3 \tag{44}$$
$$C^{xy} = 6 \ \wedge \ [\![x \perp y]\!]_3 \quad \vee \quad C^{xy} \neq 6 \ \wedge \ [\![x \neg\perp y]\!]_3 \tag{45}$$

Figure 5: Treeness clauses

Each positive (resp. negative) variable can only be mated with a negative (resp. positive) variable.

$$(x \in \mathcal{V}^+) \qquad \mathsf{mates}(x) \subseteq \mathcal{V}^- \tag{37}$$
$$(x \in \mathcal{V}^-) \qquad \mathsf{mates}(x) \subseteq \mathcal{V}^+ \tag{38}$$

We define $V^+ = V \cap \mathit{Vars}^+$ and $V^- = V \cap \mathit{Vars}^-$ for respectively the positive and negative variables in the saturated model. The mates of positive (resp. negative) variables form a partition of the negative (resp. positive) variables in the saturated model, and therefore in the matrix model since variables have no mates in the free models of (M_{ij}).

$$V^- = \biguplus_{x \in \mathcal{V}^+} \mathsf{mates}(x) \qquad V^+ = \biguplus_{x \in \mathcal{V}^-} \mathsf{mates}(x) \tag{39}$$

Treeness Constraint. Two nodes π_1 and π_2 in an ordered tree must stand in one of 5 mutually exclusive relationships: $\pi_1 = \pi_2$, $\pi_1 \lhd^+ \pi_2$, $\pi_1 \rhd^+ \pi_2$, $\pi_1 \prec \pi_2$, $\pi_1 \succ \pi_2$. Thus, for any two variables x and y, either they are interpreted by the same model and their interpretations stand in one of these relationships, or they are in distinct models. We introduce variable $C^{xy} \in \{1, 2, 3, 4, 5, 6\}$ to explicitly represent this choice, and axiomatize the options with the 6 clauses of (Fig 5). The translation $[\![\psi]\!]_3$ is given in (Fig 6).

In logic programming, disjunction is given the operational semantics of a choice point. For the clauses in (Fig 5), that would be inappropriate and would lead to disastrous performance. Instead, we assume that each clause can be implemented as one constraint: in Oz [10, 15], this is expressed using the **or** ... [] ... **end** combinator.

We can now state precisely the well-formedness constraint $[\![(\phi_{ij})]\!]_0$:

$$\begin{aligned} [\![(\phi_{ij})]\!]_0 = \quad & (22, 39) \\ \bigwedge_{i,j} \quad & (24, 25) \\ \bigwedge_{x \in \mathcal{V}} \quad & (28\text{–}33, 35\text{–}38) \\ \bigwedge_{x,y \in \mathcal{V}} \quad & (40\text{–}45) \end{aligned} \tag{57}$$

$$
\begin{aligned}
[\![x = y]\!]_3 &= \mathsf{node}(x) = \mathsf{node}(y) && (46)\\
[\![x \neq y]\!]_3 &= \mathsf{eq}(x) \parallel \mathsf{eq}(y) && (47)\\
[\![x \lhd^+ y]\!]_3 &= \mathsf{equp}(x) \subseteq \mathsf{up}(y) \ \wedge\ \mathsf{down}(x) \supseteq \mathsf{eqdown}(y) && (48)\\
[\![x \rhd^+ y]\!]_3 &= [\![y \lhd^+ x]\!] && (49)\\
[\![x \neg\lhd^+ y]\!]_3 &= \mathsf{eq}(x) \parallel \mathsf{up}(y) \ \wedge\ \mathsf{eq}(y) \parallel \mathsf{down}(x) && (50)\\
[\![x \neg\rhd^+ y]\!]_3 &= \mathsf{eq}(x) \parallel \mathsf{down}(y) \ \wedge\ \mathsf{eq}(y) \parallel \mathsf{up}(x) && (51)\\
[\![x \prec y]\!]_3 &= \mathsf{eqdown}(x) \subseteq \mathsf{left}(y) \ \wedge\ \mathsf{eqdown}(y) \subseteq \mathsf{right}(x) && (52)\\
[\![x \succ y]\!]_3 &= [\![y \prec x]\!] && (53)\\
[\![x \neg\succ y]\!]_3 &= [\![y \neg\prec x]\!] && (54)\\
[\![x \perp y]\!]_3 &= \mathsf{dom}(x) \parallel \mathsf{dom}(y) && (55)\\
[\![x \neg\perp y]\!]_3 &= \mathsf{dom}(x) = \mathsf{dom}(y) && (56)
\end{aligned}
$$

Figure 6: translation $[\![\psi]\!]_3$

4.3 Problem specific constraints

We now explicate how $[\![(\phi_{ij})]\!]_1$ forms the additional problem specific constraints that further limit the admissibility of well-formed solutions. The encoding $[\![\ ]\!]_1$ is given by clauses (58–62).

$$
\begin{aligned}
[\![(\phi_{ij})]\!]_1 &= \bigwedge_{i,j} [\![\phi_{ij}]\!]_1 && (58)\\
[\![\phi_1 \wedge \phi_2]\!]_1 &= [\![\phi_1]\!]_1 \ \wedge\ [\![\phi_2]\!]_1 && (59)
\end{aligned}
$$

A 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:

$$
[\![x\ R\ y]\!]_1 \quad = \quad C^{xy} \in [\![R]\!]_2 \qquad (60)
$$

where $[\![R]\!]_2$ turns an extended dominance relationship into a set of possible values for the choice variable (see Fig 7, where we also allow $x \perp y$ to indicate that x and y are interpreted by different models). For the labeling constraint $x \leftarrow \ell$, we assume an appropriate encoding $[\![\ell]\!]_4$ such that unification of $[\![\ell]\!]_4$ and $[\![\ell']\!]_4$ returns their most general common specialization $[\![\ell \sqcap \ell']\!]_4$.

$$
[\![x \leftarrow \ell]\!]_1 \quad = \quad \mathsf{label}(x) = [\![\ell]\!]_4 \qquad (61)
$$

Finally, the immediate dominance constraint $x : \langle y_1, \ldots, y_n \rangle$ requires a more complicated treatment:

$$
\begin{array}{rcl}
[\![x : \langle y_1, \ldots, y_n \rangle]\!]_1 & = & \mathsf{daughters}(x) = \langle \mathsf{node}(y_1), \ldots, \mathsf{node}(y_n) \rangle \\
& \wedge & \mathsf{down}(x) = \mathsf{eqdown}(y_1) \uplus \cdots \uplus \mathsf{eqdown}(y_n) \\
& \wedge & \mathsf{equp}(x) = \mathsf{up}(y_1) = \cdots = \mathsf{up}(y_n) \\
& \bigwedge_{1 \le i \le n} & \mathsf{mother}(y_i) = \mathsf{node}(x) \\
& \bigwedge_{1 \le i \le n} & \mathsf{left}(y_i) = \mathsf{left}(x) \uplus \biguplus_{1 \le j < i} \mathsf{eqdown}(y_j) \\
& \bigwedge_{1 \le i \le n} & \mathsf{right}(y_i) = \mathsf{right}(x) \uplus \biguplus_{i < j \le n} \mathsf{eqdown}(y_j)
\end{array}
\tag{62}
$$

$$
[\![R_1 \cap R_2]\!]_2 = [\![R_1]\!]_2 \cap [\![R_2]\!]_2 \qquad [\![=]\!]_2 = \{1\} \qquad [\![\prec]\!]_2 = \{4\} \tag{63}
$$

$$
[\![R_1 \cup R_2]\!]_2 = [\![R_1]\!] \cup [\![R_2]\!]_2 \qquad [\![\lhd^+]\!]_2 = \{2\} \qquad [\![\succ]\!]_2 = \{5\} \tag{64}
$$

$$
[\![\neg R]\!]_2 = \{1,2,3,4,5,6\} \setminus [\![R]\!]_2 \qquad [\![\rhd^+]\!]_2 = \{3\} \qquad [\![\bot]\!]_2 = \{6\} \tag{65}
$$

Figure 7: Relationship encoding $[\![R]\!]_2$

4.4 Solving the CSP

The minimal models of (ϕ_{ij}) can be found by enumerating the assignments to the selection variables $(S(i))_i$ and the choice variables $(C^{xy})_{x,y \in \mathcal{V}}$ consistent with $[\![(\phi_{ij})]\!]$. In practice, we have used a *first-fail* labeling strategy.

4.5 Encoding Parsing

We explain now how the task of parsing a sentence $s_1 \ \ldots \ s_n$ can be turned into a matrix problem as shown in (66), plus some additional constraints.

$$
\begin{pmatrix}
x_R^- \leftarrow S & \mathbf{true} & \ldots & \mathbf{true} \\
\phi_{11} & \cdots & \cdots & \phi_{1m} \\
 & & \vdots & \\
\phi_{n1} & \cdots & \cdots & \phi_{nm}
\end{pmatrix}
\tag{66}
$$

Row 1 corresponds to the extra root variable that cancels the polarity of the top node on the parse tree: only the 1st column is relevant; others are simply filled with distinct instances of the trivial tree, $\mathbf{true} \ = \ x^0 \leftarrow \top$, where $\top$ is the top of lattice $\mathcal{L}$. Constraint (67) is added to ensure that only the real root variable is selected:

$$
S(1) = 1 \tag{67}
$$

Row $i+1$ corresponds to word s_i. We assume a function $\mathsf{Lex}(s)$ mapping a word to a set of lexical entries. Row $i+1$ consists of all $\mathsf{formula}(e_{ij})$ for $e_{ij} \in \mathsf{Lex}(s_i)$. The row may be padded on the right with instances of **true**, and constraint (68) is added:

$$1 \leq S(i+1) \leq |\mathsf{Lex}(s_i)| \tag{68}$$

Finally precedence constraints must be imposed between anchor variables for distinct words.

$$\left.\begin{array}{l} x = \mathsf{anchor}(e_{i_1 j_1}) \\ y = \mathsf{anchor}(e_{i_2 j_2}) \\ i_1 < i_2 \end{array}\right\} \Rightarrow C^{xy} \in [\![\prec \cup \bot]\!]_2 \tag{69}$$

The CSP for the parsing problem is given by the encoding $[\![\ ]\!]$ of the matrix above, together with all additional constraints stipulated by (67–69).

5 Preliminary Results And Future Work

The ideas described in the preceding sections have been implemented in the research prototype LINDA in the concurrent constraint programming language Oz [15, 10], which supports constraints over finite domains and finite sets of integers. The primary components of LINDA are a grammar for a small fragment of English and a parser for tree description grammars.

The grammar covers just a small fragment of English including topicalization and relative clauses. The degree of lexical ambiguity is less than four, i.e. there are at most four lexical entries associated with a word in the lexicon. We plan to extend the coverage of the grammar, in particular we are interested in certain coordination phenomena.

Experimental results indicate that the parser performs well for unambiguous grammars: propagation is fast and yields shallow search trees. For ambiguous grammars like the one mentioned above the search trees are also shallow, but propagation is much more expensive. Thus our prototype parser serves as a proof of concept, but is not yet efficient enough for practical parsing.

The efficiency of our implementation is primarily affected by the following two considerations: (a) in order to account for lexical ambiguity we must introduce a form of disjunction which makes inference noticeably weaker than in the purely conjunctive fragment. Since propagation accomplishes less, effective search depends more on a good distribution strategy for the choice variables C^{xy}. Up to now, we have used a simple first-fail strategy, but we plan to work on more "clever" strategies. (b) the treeness constraints (see Figure 5) impose a quadratic number of constraints, which are implemented in LINDA by a quadratic number of propagators. In order to improve efficiency, we plan to develop specialized constraint technology, in particular we intend to replace the quadratic number of individual propagators by a single global propagator. This has been done successfully for the conjunctive fragment described in [7, 8].

However, it remains to be seen whether such improvements will suffice to allow our approach to scale to realistic grammars and longer sentences. A systematic evaluation has yet to be done.

6 Conclusion

In this paper we have shown that the constraint-based treatment of tree descriptions presented in [7, 8] can be adapted for parsing with tree-descriptions. First, we introduced a grammar framework for tree description grammars and a tree logic used for writing the tree descriptions. Second, we specified the semantics for this tree logic and extended it to accommodate a restricted form of disjunction sufficient to handle lexical ambiguity. Third, we presented a constraint model where we explain how to encode parsing problems into CSPs. Finally, we described preliminary results obtained with our LINDA prototype and outlined directions for future development.

Acknowledgments: we are grateful to Claire Gardent for initiating and co-supervising this project and for getting the present paper started.

References

[1] Ralf Backofen, James Rogers, and K. Vijay-Shankar. A first-order axiomatisation of the theory of finite trees. *Journal of Logic, Language and Information*, 1995.

[2] P. Blackburn, C. Gardent, and W. Meyer-Viol. Talking about trees. In *Proceedings of EACL'93*, Utrecht, 1993.

[3] J. Carroll, N. Nicolov, O. Shaumyan, M. Smets, and D. Weir. The LexSys Project. In *Proceedings of the 4th TAG+ workshop*, Philadelphia, 1998.

[4] Thomas L. Cornell. *Description Theory, Licensing Theory and Principle-Based Grammard.* PhD thesis, UCLA, 1992.

[5] Denys Duchier. Axiomatizing dependency parsing using set constraints. In *Sixth Meeting on Mathematics of Language (MOL6)*, Orlando, Florida, July 1999.

[6] Denys Duchier. Set constraints in computational linguistics – solving tree descriptions. In *Workshop on Declarative Programming with Sets (DPS'99)*, Paris, September 1999.

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

[8] Denys Duchier and Joachim Niehren. Solving dominance constraints with finite set constraint programming, 1999.

[9] 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.

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

[11] M.P.Marcus, D. Hindle, and M.M.Fleck. Talking about talking about trees. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, Cambridge, MA, 1983.

[12] R.A. Muskens and E. Krahmer. Description Theory, LTAGs and Underspecified Semantics. In *Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks*, pages 112–115, Philadelphia, PA, 1998. Institute for Research in Cognitive Science.

[13] Owen Rambow, K. Vijay-Shanker, and David Weir. D–Tree Grammars. In *Proceedings of ACL'95*, 1995.

[14] James Rogers and K. Vijay-Shanker. Reasoning with descriptions of trees, 1992.

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

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