A constraint-based treatment of descriptions

Denys Duchier and Claire Gardent
University of the Saarland
denys.duchier@ps.uni-sb.de, claire@coli.uni-sb.de

Abstract

Both in computational linguistics and in formal semantics, descriptions have been used which are stated in terms of *dominance*. Yet the issue of how such descriptions are processed has been little explored. In this paper, we present a constraint-based treatment of descriptions and apply it to the description-based treatment of discourse advocated in [GW98a].

Keywords: Underspecified representations, Constraint programming, Discourse semantics

1 Introduction

Both in computational linguistics and in formal semantics, descriptions have been used which are stated in terms of *dominance*. In the domain of syntactic processing, [MHM83] used dominance to support deterministic parsing. In the domain of grammar formalisms, [VS92] has used dominance to maintain monotonicity within the framework of Feature-Based Tree Adjoining Grammar (FTAG, [VSJ88]). In formal semantics, [Mus97] used dominance to capture semantic underspecification and in particular, scope ambiguities while [ENRX98] extends the idea to parallelism and ellipsis. Finally, [GW98a] use dominance to underspecify the semantic representation of discourse.

Yet the issue of how such descriptions are processed has been little explored. On the other hand, although dominance-based descriptions have been used for a wide variety of applications, the particulars both of the descriptions used and of the models assumed vary from application to application. For instance, [VS92] assumes the models to be quasi-trees (i.e. sets of trees) whereas [GW98a] take them to be discourse feature structures (a special kind of DAGs) and [ENRX98] work with so-called λ-structures that is, tree structures extended by a binding and a linking relation. It is therefore particularly useful to develop algorithms for handling descriptions which are general enough to be parameterisable for one or another application.

In this paper, we propose an efficient and general constraint-based treatment of descriptions. We then specialise it and show that it can be used to implement the treatment of discourse advocated in [GW98a, BG98]. The paper is structured as follows. Section 2 introduces [GW98a, BG98]'s description-based treatment of discourse. In Section 3, a general treatment of descriptions is given with a tree-based semantics. This approach is then

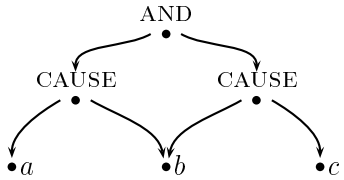
generalized to a DAG-based semantics in Section 4 so as to model [GW98a]'s Discourse Feature Structures.

2 Describing discourse meaning

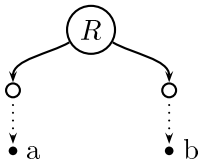
[GW98a] uses dominance to incrementally build a semantic representation for discourse. The motivation is the same as in [MHM83]: underspecifying the structure being built permits incremental, deterministic processing despite local (i.e. temporary) ambiguities. But what are the structures being described? [GW98a]'s working hypothesis is that the meaning of discourse can be represented by a special kind of DAG (Directed Acyclic Graph) namely, a Discourse Feature Structure. The meaning of a discourse is not just the conjunction of the meaning of its constituting sentences; it also includes "relational propositions" [MT86] that is, complex assertions which relate two or more discourse meanings. For instance, the meaning of:

(1) (a.) He was in a foul humour. (b.) He hadn't slept well that night. (c.)His electric blanket hadn't worked.

is not just the conjunctive assertion of (1a), (1b) and (1c). It also includes the inferential meaning that (1c) caused (1b), and that (1b) caused (1a). [GW98a]'s assumption is that this meaning can be represented by the following DAG:



Further, [GW98a] use descriptions as follows. Each time a discourse relation is inferred to hold between two discourse meanings, a description is licensed which increments the current description. Interpreting discourse then amounts to computing the dominance-minimal DFS satisfying the description. More specifically, whenever a discourse relation R is inferred to hold between two discourse meanings a and b, the following description is licensed:



Crucially, this description leaves the relation between the discourse relation R and its current arguments a and b underspecified. This is precisely the feature which permits both determinism and monotonicity. In what follows, we list the classes of ambiguity which [GW98b] handles. Section 4 will show how these problems are handled by our system and with which results.

Attachment ambiguity

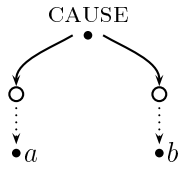
Attachment ambiguity stems from incrementality: at some point during processing, a discourse meaning is known to provide the current argument of a given discourse relation but it is unclear whether it will also be its final argument or only part of it. The following example illustrates this.

(2) a. The trains aren't running now.

b. The conductors' union called a strike last Sunday.

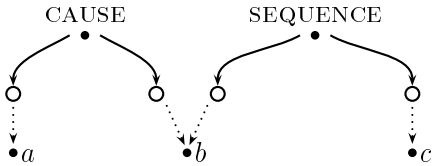
c. Then the signalmen's union walked out in sympathy.

At the end of (2b), the hearer infers a causal relation to hold between the meanings of (2a) and (2b) which is then revised to hold between (2a) and the sequence of (2b) and (2c). [GW98a]'s observation is that such revisions however, do not trigger a "repair feeling". In other words, they do not seem to lead to an increase in processing load and therefore are best modeled by a monotonic process. The use of dominance permits this. More specifically, the derivation for (2) proceeds as follows. As (2a–b) is processed, a causal relation is inferred to hold between (2a) and(2b) thereby licensing the following description:



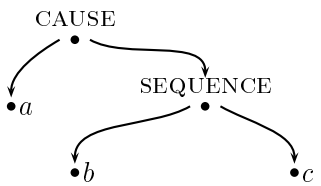
(*A*1)

Similarly, on processing (2c) the hearer infers a discourse relation of SEQUENCE to hold between the meaning b of (2b) and that c of (2c) thus licensing the following additions to the current description.



(*A*2)

Finally, the inferential component will identify (2b–c) as providing the right-hand argument of the CAUSE relation introduced by the first two clauses, which in [GW98a] is taken to license an equality constraint between the right-hand daughter of the CAUSE node and the SEQUENCE node. The only minimal structure satisfying (*A*2) and this re-entrancy constraint is the following tree:



(*A*3)

The point to note is that the initial description (*A*1) captures the local attachment ambiguity triggered by b without introducing non-determinism. Marcus's *deterministic hypothesis*

is satisfied: cases of local attachment ambiguity which require no conscious reanalysis are processed deterministically.

Relation ambiguity

The discourse relation holding between two discourse meanings may be be ambiguous. For instance, in

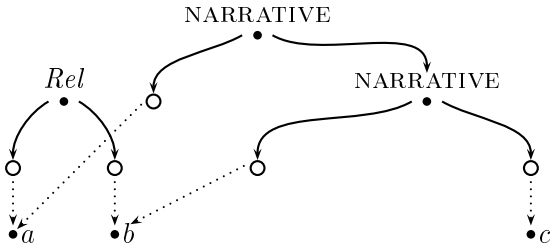
(3) a. Max fell

b. Jon pushed him.

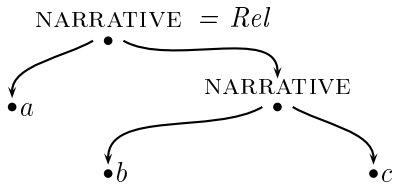
(3a) and (3b) may be related either by a causal or by a narrative (i.e. temporally sequential) relation. The first reading is obvious, the second can be illustrated by imagining the following continuation:

(4) c. and he rolled off the cliff.

To permit both possibilities, [GW98b] propose to represent the ambiguous discourse relation by a variable. Further discourse information might disambiguate this variable, which is then captured by adding the relevant information to the description. In specific, the final description for (4) is then:



for which the dominance-minimal satisfying DFS is:



As in [MHM83]'s treatment of coordination, distinct variables in the description may point to the same node in the structure. In this way, the variable discourse relation *Rel* can be disambiguated to NARRATIVE.

Scope ambiguity

Discourse exhibits cases of scope ambiguity much like sentences where the ambiguity stems from the respective scope of discourse relations. For instance, the discourse

(5) a. Sarah reads a novel

b. if she is bored

c. or she is unhappy.

is ambiguous between a reading (*if(a, or(b,c))*) where Sarah reads a novel if she is either bored or unhappy; and a reading (*or(if(a,b),c)*) where if Sarah does not read a novel when she is bored, then she is unhappy. This ambiguity is captured as in [Mus97] by leaving the structural relations between scope bearing elements underspecified. So for instance the description for examples such as (5) has the same structure as given in (A2), but no additional equality constraint is added. As a result, the description has not one but two satisfying trees, one for each possible meaning.

Embedded/Conjoined ambiguity

The conjoined/embedded ambiguity is specific to discourse. It is illustrated by the following example :

(6) a. Jon didn't come to work

b. The trains aren't running.

c. Buses aren't either.

Depending on intonation and situational knowledge, (6) is ambiguous between a reading (*and(cause(a,b), and(b,c))*) where the cause for Jon not coming to work is that the trains aren't running – and furthermore we learn that buses aren't running either; and a reading (*cause(a,and(b,c))*) in which the cause for Jon not coming to work is the conjunction of (6b) and (6c). These two readings are captured by a description whose shape is identical to (A2). The difference is that instead of identifying the right-hand daughter of the first discourse relation with the node labelled by the second discourse relation (SEQUENCE in A2), the left-hand argument of the second discourse relation (AND) is identified with the node labelled with b – this captures the fact that b is the final left-hand argument of AND. The description has two satisfying models: one, tree-like, which captures the embedded reading, and the other, graph-like, which represents the conjoined reading.

Garden-path ambiguity

Finally, there are ambiguities which lead the hearer to garden-path that is, to experience conscious processing difficulties. Thus in:

(7) a. Suppose the subject is dangerous.

b. If you touch it,

c. it will blow up.

d. Then we should call the police.

there seems to be a preference for inferring that (7b-c) provides the second argument of the CONDITION relation marked by *suppose*. On hearing (7d), the hearer then realises her mistake and re-analyses (7b-c) as elaborating on (7a). [GW98b] observes that this type of ambiguity is in stark contrast with the previous ones in that it cannot be captured by a single description. Either (7b-c) provides the right-hand argument of the CONDITION relation or it functions as a modifier of its left-hand argument. In the first case, the description licensed by (7b-c) must be dominated by the *right-hand* daughter of the CONDITION node whereas in the second case, it should be dominated by its *left-hand* daughter. The two statements are incompatible (there is no structure satisfying both descriptions) and therefore choosing one over the other may lead to subsequent failure. This is precisely what happens with (7): if (7b-c) is assumed to provide the right-hand argument of the CONDITION relation, subsequent processing leads to a description which has no satisfying structure. "Re-analysis" then takes place: to obtain an appropriate description of the meaning of (7), the initial description must be undone and another constructed which this time, will be satisfiable.

3 A constraint-based treatment of descriptions

In this section, we present a constraint-based treatment of descriptions for the semantic domain of first-order constructor trees. We introduce an intuitive notion of models, called D-trees, which are related to quasi-trees [RVS92], yet differ from them in that (1) they are based on strict dominance and (2) any node may be labelled with a constructor of a given arity. In the next section, this approach is extended to DAGs and applied to [GW98a]'s treatment of discourse semantics.

3.1 The language

Descriptions can be expressed in terms of *dominance constraints*. In this section we present such a formulation, provide an intuitive explanation of what it means to find solutions of a description problem, and derive a purely declarative, yet very efficient, constraint-based implementation.

The constraint-based technique devised by the first author has since been carefully studied by [KNT98]. We follow here their formalization according to which a description ϕ is an arbitrary conjunction of dominance and labeling constraints:

$$\phi \quad ::= \quad \phi \wedge \phi' \mid x \lhd\star y \mid x : f(x_1 \ldots x_n)$$

where x, y, x_i are taken from a set of variables and f is an n-ary function symbol from a given signature.

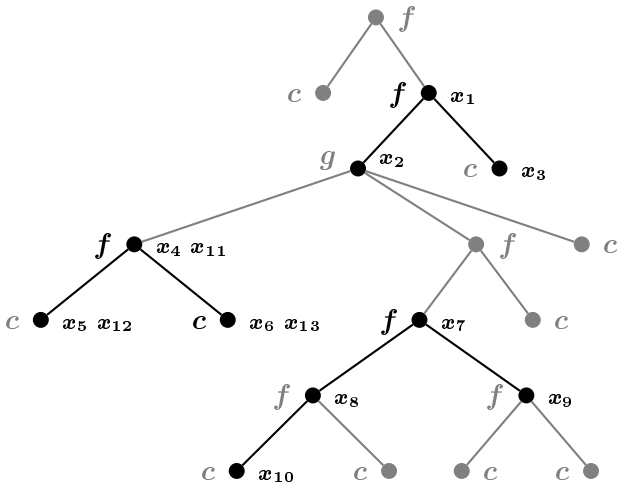
The semantics are given by interpretation over finite tree structures. A solution to ϕ consists of a finite tree T and an interpretation I that maps each variable in ϕ to a node in T. $x \lhd\star y$ means that, in the solution tree T, $I(x)$ must dominate $I(y)$, whereas the labeling constraint $x : f(x_1 \ldots x_n)$ means that $I(x) = f(I(x_1) \ldots I(x_n))$.

Complexity result. [KNT98] show that the satisfiability of propositional logic formulae can be reduced to the satisfiability of dominance constraints over the signature $\{f : 2,\ \mathbf{true} : 0,\ \mathbf{false} : 0\}$, thus establishing the NP-hardness result.

3.2 A Tree-based Semantics

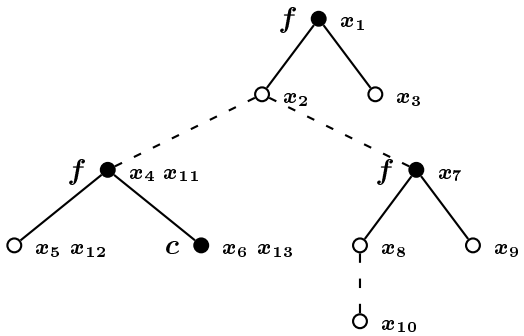
With semantics based on finite trees, a solution to a description ϕ is a tree T and an interpretation I that maps each variable of ϕ to a node in T. This has the disadvantage that every tree T' which contains T as a subtree is also a solution of ϕ. Thus, there are infinitely many solutions to ϕ.

Consider the example below: a description ϕ is listed in the box and a solution tree T is displayed next to it. The interpretation I is represented by listing next to a node the variables of ϕ which I maps to it.



$x_1 : f(x_2\ x_3)$
$x_2 \lhd\star x_4$
$x_4 \lhd\star x_5$
$x_4 \lhd\star x_6$
$x_1 \lhd\star x_7$
$x_7 : f(x_8\ x_9)$
$x_1 \lhd\star x_{10}$
$x_2 \lhd\star x_{11}$
$x_{11} : f(x_{12}\ x_{13})$
$x_{13} : c$

There is much in T which is not required to model ϕ. In the picture, all superfluous information is shown in gray. If we remove all the gray parts, we are left with a much simpler *tree shape*:



The tree shape contains two kinds of nodes: (1) **closed nodes** are labeled and colored black; from them emanate solid edges representing *immediate dominance*; (2) **open nodes**

are not labeled and hollow; from them emanate dotted edges representing *strict dominance* to an arbitrary distance.

The formal notion of a tree shape is called a D-tree [RVSW95] and allows us to define a more economical semantics by interpretation over finite D-tree structures on a given signature. A D-tree T is defined by the inductive rule:

$$T \quad ::= \quad f(T_1 \dots T_n) \mid \{T_1 \dots T_m\}$$

The first alternative stands for a closed node and the second for an open node. The semantics of D-trees are given by interpretation over finite trees relative to the same signature. If I is such an interpretation, we have:

$$I(f(T_1 \dots T_n)) = f(I(T_1) \dots I(T_n))$$

Open nodes have somewhat less direct semantics: $I(\{T_1 \dots T_m\})$ must be a tree which has $I(T_1)$, ..., $I(T_m)$ as subtrees. These notions can be made precise and formal, but the exercise lies beyond the scope of this article.

Obviously all trees are also D-trees: so what have we gained? The answer is that we can now define the notion of a *minimal D-tree model*. First, we notice that there is a partial order of specialization on D-trees: T_1 is said to be an instance of T_2 if, by erasing some information, like we did with the gray bits in the earlier picture, we can transform T_1 into T_2. Again, this could be made formal.

If we have a D-tree model of ϕ, all its instances are also models of ϕ, but not all its generalizations. There exists a unique one which is the most general: this we call a minimal D-tree model of ϕ.

In a minimal D-tree model of ϕ, all nodes interpret at least one variable in ϕ. Since there are finitely many of them, ϕ has finitely many D-tree models.

It may be helpful to draw an analogy between minimal D-tree models and *most general unifiers*. Much like a most general unifier instantiates two terms only as far as necessary to make them equal, a minimal D-tree model explicitates only as much of the tree shape as is necessary to model ϕ.

We have stated (without proof) that for every tree model of ϕ there is a unique minimal D-tree model of which it is an instance. What about the reverse direction? Is it the case that for every D-tree model of ϕ there exists a finite tree which is a ground instance of that model?

Obviously the signature must contain at least one constant c (nullary function symbol), else we could not construct a leaf node.[1] If the signature contains at least one binary function symbol[2] f, then we can replace any open node $\{\}$ by c, $\{T_1\}$ by $f(T_1\ c)$, and $\{T_1\ T_2 \dots\}$ by $f(T_1\ f(T_2 \dots))$. By repeating this procedure, we obtain a ground instance.

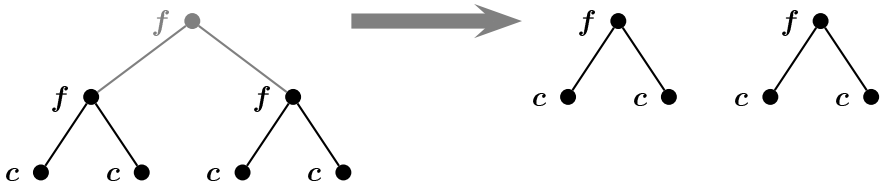
Thus, under the reasonable assumption that the signature contains at least one nullary function c and one n-ary function f with $n \geq 2$, every D-tree model of ϕ denotes a non-empty family of ground tree models of ϕ.

[1] The domain of interpretation would be empty.

[2] or n-ary, with $n \geq 2$, in which case we can arbitrarily set $n-2$ arguments to c and obtain the equivalent of a binary function symbol.

Thus we have established that to enumerate the minimal D-tree models is both complete and sound with respect to the original tree-based semantics. It also has the very desirable properties of being finite and systematic (no repeats).

One last remark concerns a technical infelicity in the notion of minimal D-tree model. A D-tree model of ϕ must consist of a set of D-trees: typically the set is a singleton, but not necessarily as the example below illustrates:



3.3 A constraint-based Approach

Our goal is now to devise a computationally effective method to enumerate the minimal D-tree models of a description ϕ. We are going to do this by representing the interpretation of a variable x of ϕ as an underspecified term N^x in a constraint programming language. This is what Prolog programmers have been doing for years when representing underspecified trees using terms containing variables. The search procedure will simply look for all possible arrangements of these representations into D-trees that model ϕ.

The idea is that every constraint in ϕ involving x and y will be translated into a constraint involving terms N^x and N^y. For this strategy to be effective, the translated constraints must involve quantities which are local to N^x and N^y. The most important constraint is that of dominance, and, in this respect, it is crucial to notice that every two nodes N^x and N^y of a D-tree always stand in one of four mutually exclusive relationships:

$N^x = N^y$	they are equal
$N^x \lhd\!+ N^y$	N^x strictly dominates N^y
$N^y \lhd\!+ N^x$	N^y strictly dominates N^x
$N^x \perp N^y$	they are unrelated

For every N^x we can keep track of all variables whose interpretation is equal to, strictly dominates, is strictly dominated, or is unrelated to that of x.

In earlier pictures, we represented an interpretation I by decorating each node of a D-tree with the set of variables of ϕ that I maps to it. Now, we simply extend this decoration with the set of variables that are above, below, and unrelated.

We assume a constraint language that supports *set variables*. Most readers will be familiar with the idea of *finite domain* variable, i.e. a variable that denotes an integer and can be constrained (e.g. $x < y$). Set variables are just the same idea but ranging over sets. Our implementation language (Oz, [Smo95]) supports only finite sets of integers, whose constraint theory has an extremely efficient implementation. However, the formulation which we present here does not depend on this restriction.

The idea is that for every node N^x, we make explicit the set of variables whose interpretations are equal, above, below, and unrelated to N^x. We represent N^x by the feature

structure below:

$$\left[\begin{array}{lcl} = & : & N^x_{=} \\ \lhd + & : & N^x_{\lhd +} \\ +\rhd & : & N^x_{+\rhd} \\ \bot & : & N^x_{\bot} \end{array}\right]$$

we write N^x_α to notate feature α of term N^x. Here, $N^x_=$ is the set of variables whose interpretations are equal to that of x. In earlier pictures, we represented this set by listing the variables to the right of the node. $N^x_{\lhd +}$ is the set of variables whose interpretations are strictly dominated by the interpretation of x, and $N^x_\bot$ are these variables whose interpretations neither dominate nor are dominated by the interpretation of x: we say that their interpretations are unrelated.

Since the interpretations of any two variables must stand in one of the mutually exclusive relationships described earlier, the sets $N^x_=$, $N^x_{\lhd +}$, $N^x_{+\rhd}$ and $N^x_\bot$ must form a partition of the set $\mathcal{V}$ of variables of ϕ:

$$\mathcal{V} = N_= \uplus N_{\lhd +} \uplus N_{+\rhd} \uplus N_\bot$$

where $\uplus$ denotes disjoint union. Furthermore, x must be in the set of variables mapped to N^x:

$$x \in N^x_=$$

For convenience, we define $N_{\lhd\star} = N_= \cup N_{\lhd +}$ and $N_{\star\rhd} = N_= \cup N_{+\rhd}$. The dominance constraint $x \lhd\star y$ translates to the constraint $N^x \lhd\star N^y$:

$$N^x \lhd\star N^y \quad \equiv \quad \left\{\begin{array}{l} N^x_{\lhd\star} \supseteq N^y_{\lhd\star} \\ N^x_{\star\rhd} \subseteq N^y_{\star\rhd} \\ N^x_\bot \subseteq N^y_\bot \end{array}\right.$$

To account for the labeling constraint $x : f(x_1 \ldots x_n)$, we equip nodes with the additional feature ℓ. The labeling constraint translates as follows:

$$\begin{array}{lcl} N^x_\ell & = & f(N^{x_1} \ldots N^{x_n}) \\ N^x_{\lhd +} & = & N^{x_1}_{\lhd\star} \uplus \ldots \uplus N^{x_n}_{\lhd\star} \\ N^x_{\star\rhd} & = & N^{x_1}_{+\rhd} = \ldots = N^{x_n}_{+\rhd} \end{array}$$

The first line serves two purposes: it guarantees that two equal nodes (1) must be labeled with the same constructor, (2) must have (pairwise) equal daughters.

The tree structure is guaranteed by requiring that every pair of nodes N^x and N^y stand in one of the four mutually exclusive relationships described earlier:

$$N^x = N^y \quad \vee \quad N^x \lhd + N^y \quad \vee \quad N^x +\rhd N^y \quad \vee \quad N^x \bot N^y \qquad (\alpha)$$

These alternatives can be translated according to the following definitions:

$$\begin{array}{lcl} N^x = N^y & \equiv & \mathrm{unify}(N^x, N^y) \\ N^x \neq N^y & \equiv & N^x_= \cap N^y_= = \emptyset \\ N^x \lhd + N^y & \equiv & N^x \lhd\star N^y \quad \wedge \quad N^x \neq N^y \\ N^x \bot N^y & \equiv & N^x_{\lhd\star} \subseteq N^y_\bot \quad \wedge \quad N^y_{\lhd\star} \subseteq N^x_\bot \end{array}$$

The first alternative involves unification of set variables: when the element domain consists of terms, this is a very expensive AC-unification problem; when the elements are integers, it is a very efficient or-like operation. We assume the latter: in our encoding, every variable is represented by a distinct integer.

Note that the only source of disjunction comes from formula schema (α): thus, simply by arbitrarily choosing the relationship in which every two nodes stand, we obtain a purely conjunctive formula. Inferential closure (constraint propagation) either derives a contradiction or constructs a minimal D-tree model (a solved form).

3.4 Efficient Implementation

So far, we have specified declaratively what all the constraints should be. Every solution of this constraint system represents a D-tree model of ϕ. We now explain how to efficiently enumerate all and only the minimal D-tree models.

In a minimal D-tree model, the relationship in which any two nodes N^x and N^y stand is determined. We can make this choice explicit in (α) by introducing a finite domain variable $C^{x,y} \in \{1..4\}$:

$$\begin{array}{lllll} & C^{x,y} = 1 & \wedge & N^x = N^y \\ \vee & C^{x,y} = 2 & \wedge & N^x \lhd + N^y \\ \vee & C^{x,y} = 3 & \wedge & N^x + \rhd N^y \\ \vee & C^{x,y} = 4 & \wedge & N^x \perp N^y \end{array}$$

This does not change the solutions since the alternatives are mutually exclusive. Choosing $C^{x,y}$ selects precisely one alternative; the others become inconsistent. In every D-tree model, $C^{x,y}$ must be determined. By making no further choices, we enumerate only minimal D-tree models. Thus, the problem of searching for a solution to the initial dominance description has been reduced to the search for a consistent assignment to the *choice* variables $C^{x,y}$ $\forall x, y \in \mathcal{V}$.

Computationally, we are going to make these choices in an incremental sequence. As soon as one choice has been decided, the constraint of the selected alternative strengthens the partial description of the solution. Often, the currently known constraints are sufficient to eliminate all but one of the alternatives in a disjunction. It is desirable to notice this as soon as possible and add the remaining alternative to the global description.

The programming language Oz makes this possible. Disjunctions are implemented as concurrent agents that speculatively investigate their alternatives.[3] When only a single consistent alternative remains, it is automatically *committed* (chosen).

A disadvantage of the 4-way disjunction (α) is that it can only contribute to the global constraints when it has been fully decided. A more effective formulation separates the alternatives into 4 concurrent agents:

$$\begin{array}{lll} C^{x,y} = 1 \wedge N^x = N^y & \vee & C^{x,y} \neq 1 \wedge N^x \neq N^y \\ C^{x,y} = 2 \wedge N^x \lhd + N^y & \vee & C^{x,y} \neq 2 \wedge N^x \not\lhd + N^y \\ C^{x,y} = 3 \wedge N^y \lhd + N^x & \vee & C^{x,y} \neq 3 \wedge N^y \not\lhd + N^x \\ C^{x,y} = 4 \wedge N^x \perp N^y & \vee & C^{x,y} \neq 4 \wedge N^x \not\perp N^y \end{array}$$

[3] This generalizes the idea of *deep guards*.

As soon as one alternative becomes inconsistent, its negation is added to the global store. With this encoding, the standard *first-fail* search strategy appears to give rather good results.

4 Application to discourse

We now show how the constraint-based approach described above can be extended to model [GW98a]'s treatment of discourse. First, we tailor the above framework to a DAG-based semantics as advocated in [BG98]. Then, we describe its use for implementing [GW98a]'s account.

4.1 A DAG-based Semantics

We can develop a similar framework with semantics based on finite DAGs. It can be obtained by weakening the previous formulation (trees are a subset of dags), and we shall be looking for *minimal D-dag models* defined analogously to minimal D-tree models.

Due to the requirement of acyclicity, it is still the case that:

$$\mathcal{V} = N_{=} \uplus N_{\lhd +} \uplus N_{+\rhd} \uplus N_{\bot}$$

Thus, both our first equation and our search strategy remain unchanged. However, the translation of $N^x \lhd\star N^y$ looses the 3rd constraint since nothing can be concluded concerning the sets $N^x_\bot$ and $N^y_\bot$:

$$N^x \lhd\star N^y \quad \equiv \quad \begin{cases} N^x_{\lhd\star} \supseteq N^y_{\lhd\star} \\ N^x_{\star\rhd} \subseteq N^y_{\star\rhd} \end{cases}$$

Finally, for the labeling constraint $x : f(x_1 \ldots x_n)$, it is no longer the case that $N^{x_1} \ldots N^{x_n}$ must be disjoint subtrees. However our linguistic application stipulates that no sibling may dominate another. Thus the translation becomes:

$$\begin{array}{lcl} N^x_\ell & = & f(N^{x_1} \ldots N^{x_n}) \\ N^x_{\lhd +} & = & N^{x_1}_{\lhd\star} \cup \ldots \cup N^{x_n}_{\lhd\star} \\ \emptyset & = & N^{x_i}_{=} \cap N^{x_j}_{\star\rhd} \quad \forall i \neq j \in 1..n \end{array}$$

Nothing else need change. Results of completeness, soundness, finiteness and systematicity still hold.

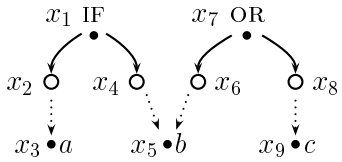
4.2 Treatment of Discourse Descriptions

Discourse descriptions are given dag-based semantics as outlined in the previous section. To account for certain phenomena, we have found it convenient to decompose the labeling constraint $x : f(x_1 \ldots x_n)$ into two simpler constraints: one that specifies the constructor and one that specifies the daughters. A description is now defined by the inductive rule:

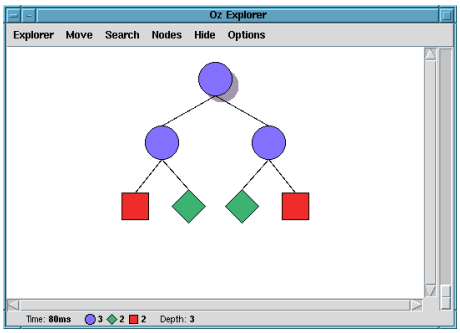
$$\phi \quad ::= \quad \phi \wedge \phi' \mid x \lhd\star y \mid x : f \mid x : \langle x_1 \ldots x_n \rangle$$

If I is an interpretation of ϕ, $x \lhd\star y$ means that $I(x)$ must dominate $I(y)$, $x : f$ means that $I(x)$ must be labeled with constructor f, and $x : \langle x_1 \dots x_n \rangle$ means that $I(x)$ is labeled with an n-ary constructor and has daughters $I(x_1)$ through $I(x_n)$. In the following, we write $x : f\langle x_1 \dots x_n \rangle$ to abbreviate the common case $x : f \wedge x : \langle x_1 \dots x_n \rangle$

Consider the 'scope ambiguity' example on page 4: (a) Sarah reads a novel (b) if she is bored (c) or she is unhappy. The description is displayed in the box below.



$x_1 : \text{IF}\langle x_2, x_4 \rangle$
$x_2 \lhd\star x_3$
$x_3 : a\langle\rangle$
$x_4 \lhd\star x_5$
$x_5 : b\langle\rangle$
$x_7 : \text{OR}\langle x_6, x_8 \rangle$
$x_6 \lhd\star x_5$
$x_8 \lhd\star x_9$
$x_9 : c\langle\rangle$

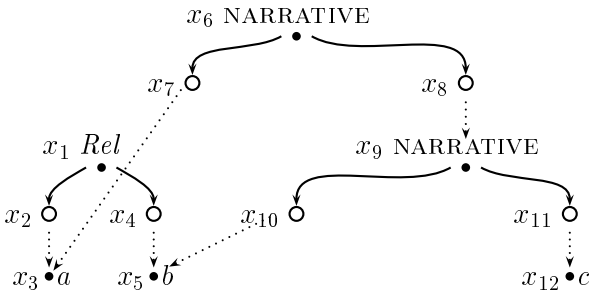


Additionally to the description, our linguistic application makes the following stipulations:

- there is a unique root: the interpretation of one variable dominates all the others
- a variable which is not assigned a label in the description must be identified with one that is

The corresponding search tree is displayed in the window on the right: the blue circles represent choice nodes, the red squares failed nodes, and the green diamonds are the two solutions. In this example, the full search tree is obtained in 80ms.

The ability to state constructor and daughter constraints separately allows us to expression the 'relation ambiguity' description of page 4 as follows:



$x_1 : \langle x_2, x_4 \rangle$
$x_2 \lhd\star x_3$
$x_3 : a\langle\rangle$
$x_4 \lhd\star x_5$
$x_5 : b\langle\rangle$
$x_6 : \text{NARRATIVE}\langle x_7, x_8 \rangle$
$x_7 \lhd\star x_3$
$x_8 \lhd\star x_9$
$x_9 : \text{NARRATIVE}\langle x_{10}, x_{11} \rangle$
$x_{10} \lhd\star x_5$
$x_{11} \lhd\star x_{12}$
$x_{12} : c\langle\rangle$

x_1 is simply given the daughter constraint $x_1 : \langle x_2, x_4 \rangle$, but its label remains unspecified.

5 Conclusion

In computational linguistics and formal semantics, descriptions of trees and DAGs stated in terms of dominance have gained popularity. In this paper, we have contributed a constraint-based framework for processing such descriptions.

Our approach is purely declarative and formulated in terms of constraints between sets of variables which straightforwardly capture what it means for two nodes to be part of a tree (or graph) structure, or to stand in a more specific dominance relationship. By taking advantage of modern constraint programming technology, this transparent declarative encoding, when further equipped with a search strategy, also constitutes an implementation.

For the tree-based semantics, experimental results indicate that even the simplistic *first-fail* enumeration of the choice variables gives excellent performance. The NP-hardness result, however, shows that we cannot expect miracles. From the computational point of view, constraint programming offers two potential advantages: (1) an efficient implementation of inference through constraint propagation, and (2) an efficient treatment of value disjunction through *finite domain* and *finite set* variables. It is the job of the search strategy to take effective advantage of the framework. In the weaker semantic domains of DAGs, its importance becomes again critical.

A methodological bonus of our declarative approach is that the design of an appropriate search strategy can be investigated independently: the search strategy is merely a parameter of our system. We are currently exploring the effectiveness of more specialized search trategies.

Finally, our treatement of descriptions is stated in general terms using D-trees as the basic models. As we showed, the approach can easily be tailored to graphs. But it can also be tailored to minimal trees or λ-structure. Thus it offers a general framework in which the various description-based approaches mentioned in the introduction can be implemented, combined and compared.

References

[BG98] Patrick Blackburn and Claire Gardent. A specification language for discourse. In *Proceedings of LACL'98 (Logical Aspects of Computational Linguistics)*, Grenoble, France, 1998.

[ENRX98] M. Egg, J. Niehren, P. Ruhrberg, and F. Xu. Constraints over lambda-structures in semantic underspecification. In *Proceedings of ACL/COLING '98*, Montreal, Canada, 1998.

[GW98a] Claire Gardent and Bonnie Webber. Describing discourse semantics. In *Proceedings of the 4th TAG+ Workshop*, University of Pennsylvania, Philadelphia, 1998.

[GW98b] Claire Gardent and Bonnie Webber. Incremental discourse processing. In preparation, 1998.

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

[MHM83] 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.

[MT86] William Mann and Sandra Thompson. Relational propositions in discourse. *Discourse Processes*, 9:57–90, 1986.

[Mus97] Reinhard Muskens. Order-independence and underspecification. University of Tilburg, 1997.

[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 Jan van Leeuwen, editor, *Computer Science Today*, volume 1000 of *LNCS*, pages 324–343. Springer Verlag, 1995.

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

[VSJ88] K. Vijay-Shanker and Aravind Joshi. Feature based tags. In *Proceedings of the 12th International Conference of the Association for Computational Linguistics*, pages 573–577, Budapest, 1988.