An Algorithm for the Construction of Dependency Trees

Gerrit F. van der Hoeven

University of Twente, Department of Computer Science, Section SETI
P.O. Box 217, 7500 AE Enschede, The Netherlands
email: vdhoeven@cs.utwente.nl

Abstract

A casting system is a dictionary which contains information about words, and relations that can exist between words in sentences. A casting system allows the construction of dependency trees for sentences. They are trees which have words in roles at their nodes, and arcs which correspond to dependency relations. The trees are related to dependency trees in classical dependency syntax, but they are not the same. Formally, casting systems define a family of languages which is a proper subset of the contextfree languages. It is richer than the family of regular languages however. The interest in casting systems arose from an experiment in which it was investigated whether a dictionary of words and word-relations created by a group of experts on the basis of the analysis of a corpus of titles of scientific publications, would suffice to automatically produce reasonable but maybe superficial syntactical analyses of such titles. The results of the experiment were encouraging, but not clear enough to draw firm conclusions. A technical question which arose during the experiment, concerns the choice of a proper algorithm to construct the forest of dependency trees for a given sentence. It turns out that Earley's well-known algorithm for the parsing of contextfree languages can be adapted to construct dependency trees on the basis of a casting system. The adaptation is of cubic complexity. In fact one can show that contextfree grammars and dictionaries of words and word-relations like casting systems, both belong to a more general family of systems, which associate trees with sequences of tokens. Earley's algorithm cannot just be adapted to work for casting systems, but it can be generalized to work for the entire large family.

1 Associating trees with sentences

This paper is about formal systems which associate trees with sequences of symbols. Most of the contents of the paper deal with definitions, the formal properties of the systems defined, common generalizations of new and well-known systems, and finally parsing problems. First however, we will describe an experiment which gave rise to the formalisms we introduce here. The experiment is as follows.

A group of experts is given a set of titles of scientific publications in their field of expertise. As a first step, they are asked to give a structural analysis of the titles. More precisely: their task is to draw lines between related words in each of the titles of the corpus, in such a way that every title gets a tree structure.

The words of the title are the nodes of the tree, the arrows connecting mothers and daughters in the tree stand for: 'in some sense related'. Such an analysis applied to the title of this paper might yield a tree like the one in figure 1. There is one restriction concerning word order the experts must obey in drawing their trees. The restriction is, that if they relate word b to word a, then no word c which is at the other side of a than b is in the textual order, can be related to b.

The second step is, to ask the experts to motivate their tree constructions. The motivation must take a specific form. They are asked to give a name to the lines connecting mother- and daughter-words in their trees, in a consistent way for all titles. In this way they are supposed to make explicit which relations between words they

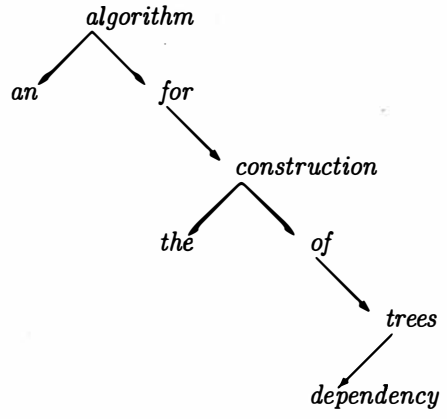

Figure 1: A tree for the title

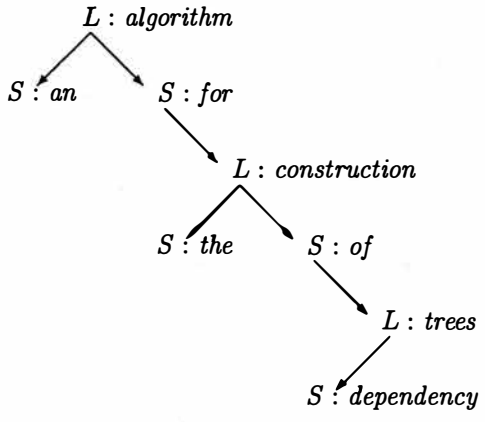

Figure 2: An attributed tree for the title

consider important. Moreover, they are asked to name the characteristics of the individual words in every tree. Thus it is made explicit what the properties of the individual words are that make them fit in a particular relationship to one another.

The final step would be, to redraw the trees in such a way that the relation names assigned to lines connecting mothers and daughters, are now assigned to the daughters, together with the characteristics of the daughter words.

From this second representation, the original one can be easily reconstructed, since there is always only one line in the tree to which the relation component in the dressing of a daughter can belong. This final tree translation does not affect the structure of the trees nor does it contribute to the insight into their structure. It is relevant for technical purposes: now we have trees in which only the nodes have attributes, instead of both nodes and arcs.

A schematic representation of a final result tree (with just two simple attributes L and S, where a real tree would have more and more complex ones) is the tree in figure 2.

The outcome of such an experiment could be interesting for all sorts of reasons. Our interest is simply to use the trees, the word characteristics and the relations between words as indicated by the experts, to construct similar trees for titles of publications that the experts did not consider.

The basic idea for extrapolation of the results of the experiment is to abstract from the trees that are delivered, and to concentrate on *word profiles* that can be derived from the trees. A word profile is roughly a triple consisting of an attribute, and two sets of attributes. A profile for a word can be derived from a set of trees by first collecting all trees in which the word has the same attribute. Next the set of attributes assigned to daughters of that word in any of the trees are collected. Finally this set of daughter attributres is split in two, possibly overlapping, subsets. One has the attributes assigned to daughters which occur to the left of the given word, and the other has the attributes assigned to words which occur to the right. Note that a word can occur with different attributes, and that therefore a word can have more than one profile. The formal notion of a casting system introduced below, gives the precise elaboration of this idea.

Before we turn to the formal definitions and their properties, a few remarks are in order. The first remark concerns the experiment described above. It was never properly conducted. There have been experts drawing trees for titles of scientific publications, but they were the same group as the ones who used the resulting trees for the analysis of new titles. Although it was not known beforehand to which new titles the analysis method would be applied, the experts making the original analyses were clearly aware of the ways in which their results would be used. Their discussions therefore concentrated not so much on the actual analyses they made, but more on the generality of the relations between words and the characteristics of individual words they introduced. Moreover, the corpus of titles they considered was too small to draw any firm conclusions from the outcome of the experiment anyway. But the results were not discouraging.

The second remark concerns the kind of trees we consider and the notion of word profile. In shape, the trees are very much like dependency trees. What we ask the experts to do, could rightly be called dependency analysis. The syntactical claims in our approach however, are far from classical dependency syntax. In fact, we will present a system that is capable of assigning trees to well-formed utterances, but that will assign trees just as easily to many ill-formed utterances. The question what makes a sentence or phrase correct, let alone the explanation of correctness at any level of adequacy, does not interest us. What we want, is to have a tree shaped representation of an utterance which organizes the information in that utterance in a way that is both manageable and acceptable to a human reader or hearer of the utterance.

As for the word profiles, if one thinks of the attributes for words as semantic categories, and omits the left-of/right-of distinction, a word profile bears some resemblance to a case frame. In fact, it seems that the analysis we consider here could just as well be performed on the basis of a dictionary of case frames, as on the basis of a dictionary of word profiles that are derived from a corpus of handmade analyses.

1.1 Casting systems and dependency trees

A casting system is nothing but the formal description of a dictionary of word profiles, as introduced informally above. There is a slight change of terminology however. What we called 'words' above, are 'actors' in the formal representation, and what we called 'attributes', are now 'roles'. A casting system tells which actors can play which roles, and what supporting roles the actors in their roles expect to their left and to their right.

Strictly formal, a casting system is a seven tuple of sets, symbols and relations. It fixes a relation between sequences of 'actors' and dependency trees. It is a dictionary of words, word roles, and co-occurrence relations between words and roles.

Definition 1 A *casting system* Γ is a seven tuple with the following components:

- A, the *actor set* of Γ. A is a finite alphabet. Its elements are actors.
- P, the *set of roles* of Γ. P is a finite set.
- L, the set of *leading roles* of Γ. L is a subset of P.
- ι, the *invisible role* of Γ. ι is a distinguished element of P.
- $\Box : \Box$, the *can-be-played-by relation* of Γ. It relates roles and actors. If p is a role and

Actors	{ *an*, *algorithm*, *for*, *the*, *construction*, *dependency*, ... }
Roles	$\{L, S, \iota\}$
Leading roles	$\{L\}$
The invisible role	ι
Can-be-played-by	L : *algorithm*, L : *construction*, ...
	S : *an*, S : *for*, S : *the*, S : *dependency*, ...
Can-be-combined-left	$S\backslash L$: *algorithm*, $S\backslash L$: *construction*, ...
Can-be-combined-right	L : *algorithm* $/S$, S : *for* $/L$, ...
Combine-with-ι	S : *an* $/\iota$, S : *dependency* $/\iota$, ...
	$\iota\backslash S$: *an*, $\iota\backslash S$: *for*

Figure 3: A simple casting system

a is an actor then we write $p : a$ to express that p can be played by a.

- $\square\backslash\square : \square$, the *can-be-combined-left relation* of Γ. It relates roles with actors and roles. If p and q are roles, and a is an actor, then we write $q\backslash p : a$ to express that a in role p can play together with any actor in role q to its left.

- $\square : \square/\square$, the *can-be-combined-right relation* of Γ. The counterpart of the previous relation in the following sense: we write $p : a/q$ to express that a in role p can play together with any actor in role q to its right.

It should be obvious that the can-be-played-by, can-be-combined-left and can-be-combined-right relations give us the ingredients of a word profile. The special status of the set of leading roles is, that it contains the roles that can appear at the root of a well-formed tree. The invisible role is important in the 'combine' relations. Possibility of combination with the invisible role indicates that an actor can occur without support of other roles, i.e. without daughters in a dependency tree.

An example of a small casting system, corresponding to the dependency tree shown earlier for the title of this paper, is in figure 3.

A casting system is just the rules of the game. The rules can be derived from a given set of trees. But the game is the inverse: to associate trees with sequences of actors. That is what the following definition is about.

Definition 2 Let Γ be a casting system with actor set A, and let u be a string of actors.

A *casting tree* or a *dependency tree* for u w.r.t. Γ is a directed graph T. The nodes of T are pairs (p, α), with p a role of Γ, and α an occurrence of an actor of Γ in u. The graph T has the following properties:

- it is a tree,
- the role of every node can be played by the actor of the node;
- if (q, β) is a successor of (p, α) and the occurrence β is to the left of the occurrence α, then $q\backslash p : a$, where a is the actor of which α is an occurrence; if β is to the right of α, then $p : a/q$;
- if node (p, α) has no successors (q, β) with β to the left of α, then $\iota\backslash p : a$; if there are no successors (q, β) with β to the right of α, then $p : a/\iota$;
- with every node there is a segment v of u, which consists of the actors in the node and in its descendants. In particular, the root node corresponds to the entire sequence u.

A casting system has an associated formal language. The existence of a dependency tree w.r.t. the casting system determines whether or not a string of actors belongs to the formal language.

Definition 3 Let Γ be a casting system, with actor set A. The *language associated with* Γ is the set of actor sequences in A^* which have a dependency tree w.r.t. Γ. We call this language $\mathcal{L}_\Gamma$.

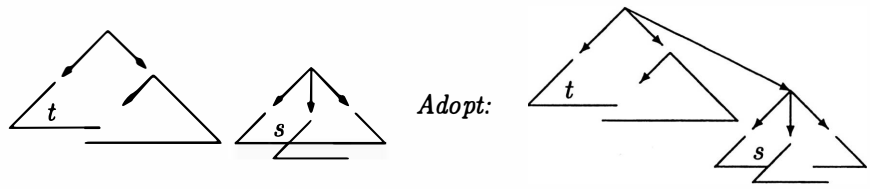

Figure 4: Where t adopts s

The family of casting languages has some peculiar properties. We mention the following facts without proof.

Fact 1 If Γ is a casting system, $\mathcal{L}_\Gamma$ is the associated language, a and b are actors, and ab is a string in $\mathcal{L}_\Gamma$, then ab^n or a^nb is in $\mathcal{L}_\Gamma$ for every $n > 0$. More general, in every string in $\mathcal{L}_\Gamma$ with two or more actors, there is at least one actor which can be repeated arbitrarily often, and the resulting string will again be in $\mathcal{L}_\Gamma$.

Fact 2 If Γ is a casting system, and $\mathcal{L}_\Gamma$ is the associated language, then $\mathcal{L}_\Gamma$ is contextfree.

Fact 3 There are regular languages $\mathcal{L}$, for which no casting system Γ exists such that $\mathcal{L} = \mathcal{L}_\Gamma$.

Fact 4 There are casting systems Γ which have an associated language $\mathcal{L}_\Gamma$ which is not regular.

Fact 1 shows that casting systems are not the proper systems to distinguish between ill- and well-formed phrases of a natural language. Fact 3 is an immediate consequence of fact 1. From facts 2 to 4 we see that casting languages do not have a proper place in the Chomsky hierarchy, but are 'somewhere in between regular and contextfree'.

There are quite a number of open problems concerning casting systems. E.g. is it decidable whether two casting systems have the same associated language or not?

We will not go into formal properties of casting systems here, nor will we further pursue the question of their suitability for the description or the processing of natural language. The second half of this paper deals with the parsing problem casting systems pose, and the solution to that problem which is found in a common generalization of casting systems and contextfree grammars.

2 Parsing: the construction of a dependency tree for a given sentence

In this section we discuss the problem of constructing dependency trees for sentences on the basis of a casting system Γ. But what we shall do is *not* to present a parsing algorithm for casting languages. Our approach is to consider the association of analysis trees with sequences of symbols in general terms, independent of whether the associated trees are dependency trees or e.g. contextfree parse trees. First we will show that such a general approach, of which dependency trees and contextfree parse trees are both an instance, indeed exists. Then we consider the parsing problem for the generalized notion of tree association. We conclude that the generalized notion has such characteristics that it allows an Earley-like parsing strategy. It follows that dependency trees can be constructed by an Earley-like algorithm. To obtain the actual Earley algorithm for contextfree languages from the generalized version, optimizations are needed which are typical for the contextfree case. We shall not go into these optimizations.

The kernel of the generalized notion of tree association, is the notion of FB-system. Strictly formal, an FB-system is a seven tuple of sets and relations. It fixes a set of colored trees, and a relation between colored trees and sequences of symbols.

A *colored tree* is a tree with a mapping from its nodes into a set of colors. The set of colors is just an arbitrary finite set.

In the sequel we shall work with three basic tree forming operations. They are: *Single*, *Adopt*, and *Recolor*, and they are defined as follows:

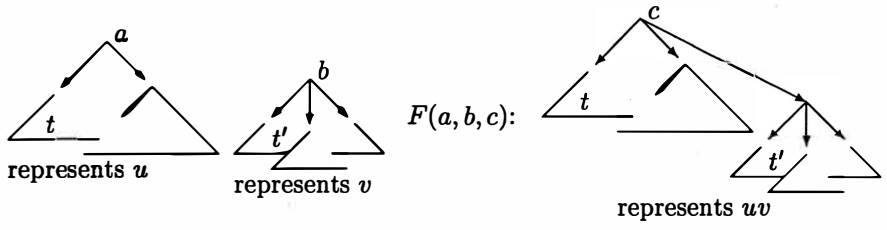

Figure 5: Forward adoption

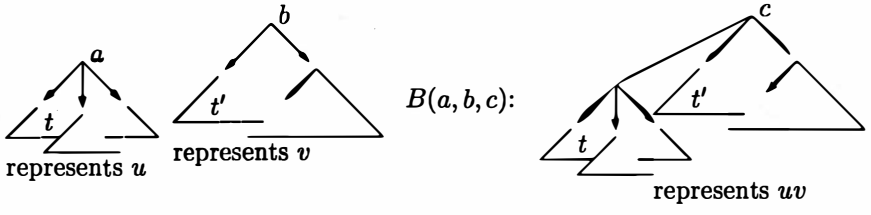

Figure 6: Backward adoption

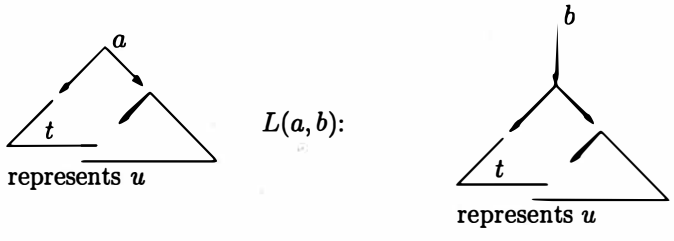

Figure 7: Lift

Single takes a color, and yields a tree consisting of just a single node, which has this color.

Adopt takes two trees, which it turns into one by making the root of the second tree a daughter of the root of the first one (cf. Figure 4).

Recolor finally, takes a tree and a color and 'changes' the root of the given tree to have the given color.

We shall use the phrase: the *color of a tree* to mean the color of the root of the tree. We shall denote this color of t by $\gamma(t)$.

With these preliminaries we are now able to give a precise definition of an FB-system, and to give an interpretation to this formal definition.

Definition 4 An *FB-system* is a seven tuple Φ with the following components:

- C, the *alphabet of colors* of Φ.
- C_R, subset of C with the *admissible root colors.*
- S, the *alphabet of symbols* of Φ. S and C are disjoint.
- R, the *representation relation* of Φ, a relation on $C \times S$. $R(c, s)$ indicates that the color c 'represents' the symbol s. A color which represents a symbol is a *terminal color.*
- F, the *forward relation* of Φ, a relation on $C \times C \times C$. $F(c_0, c_1, c)$ indicates that a tree with color c_0 can adopt one with color c_1, to form a new tree of which the root is (re)colored by c. If c is terminal, then so is c_0. If both are terminal, then they represent the same symbols.

- B, the *backward relation* of Φ, a relation on $S \times S \times S$. $B(c_0, c_1, c)$ indicates the same as $F(c_0, c_1, c)$, except that now c_1 adopts c_0. If c is terminal, then so is c_1. If both are terminal, then they represent the same symbols.

- L, the *lift relation* of Φ, a relation on $S \times S$. $L(c_0, c_1)$ indicates that a tree with color c_0 can be adopted by a single node tree with color c_1. c_1 can not be a terminal color.

FB-systems are named after their characteristic forward- and backward-relations. An FB-system is just a formalism which fixes the rules of the game. The game is to build colored trees, and to associate such trees with sequences of symbols. The next definition (with pictures) tells us how to interpret the contents of an FB-system.

Definition 5 The set of *admissible trees* T_Φ over an FB-system Φ is a set of colored trees. Every tree in T_Φ is an *analysis tree* for a sequence of symbols.

The set T_Φ and the analysis tree relation are inductively defined by the following four clauses:

1. If s is a symbol and the color c represents s, then $Single(c)$ is an admissible tree, it is an analysis tree for s.

2. If t with $\gamma(t) = a$ is an analysis tree for u, and t' with $\gamma(t') = b$ is an analysis tree for v, and $F(a, b, c)$ holds, then $Recolor(Adopt(t, t'), c)$ is an admissible tree, it is an analysis tree for uv (forward adoption, cf. figure 5).

3. If t with $\gamma(t) = a$ is an analysis tree for u, and t' with $\gamma(t') = b$ is an analysis tree for v, and $B(a, b, c)$ holds, then $Recolor(Adopt(t', t), c)$ is an admissible tree, it is an analysis tree for uv (backward adoption, cf. figure 6).

4. If t with $\gamma(t) = a$ is an analysis tree for u, and $L(a, b)$ holds, then $Adopt(Single(b), t)$ is an admissible tree, it is also an analysis tree for u (lift, cf. figure 7).

2.1 A contextfree grammar as an FB-system

To illustrate the concept of FB-system and the association of analysis trees with sequences of symbols, we will present a simple contextfree grammar as an FB-system, and show how the parse tree of a simple sentence can be obtained as an analysis tree according to the foregoing definition.

Terminals	$\{d, n, p\}$
Non-terminals	$\{ NP, PP \}$
Start symbol	NP
Rules	$NP \rightarrow d\ n$, $NP \rightarrow NP\ PP$, $PP \rightarrow p\ NP$
Colors	$\{d \rightarrow d, n \rightarrow n, p \rightarrow p, NP \rightarrow d\ n, NP \rightarrow NP\ PP, PP \rightarrow p\ NP,$ $NP \rightarrow d, NP \rightarrow NP\ PP \rightarrow p\}$
Root colors	$\{ NP \rightarrow d\ n, NP \rightarrow NP\ PP \}$
Symbols	$\{d, n, p\}$
Represent	$R(d \rightarrow d, d), R(n \rightarrow n, n), R(p \rightarrow p, p)$
Forward	(F1) $F(NP \rightarrow d, n \rightarrow n, NP \rightarrow d\ n)$
	(F2) $F(NP \rightarrow NP, PP \rightarrow p\ NP, NP \rightarrow NP\ PP)$
	(F3) $F(PP \rightarrow p, NP \rightarrow d\ n, PP \rightarrow p\ NP)$
	(F4) $F(PP \rightarrow p, NP \rightarrow NP\ PP, PP \rightarrow p\ NP)$
Backward	empty
Lift	(L1) $L(NP \rightarrow d\ n, NP \rightarrow NP)$ (L2) $L(NP \rightarrow NP\ PP, NP \rightarrow NP)$
	(L3) $L(d \rightarrow d, NP \rightarrow d)$ (L4) $L(p \rightarrow p, PP \rightarrow p)$

Figure 8: The example grammar and the corresponding FB-system

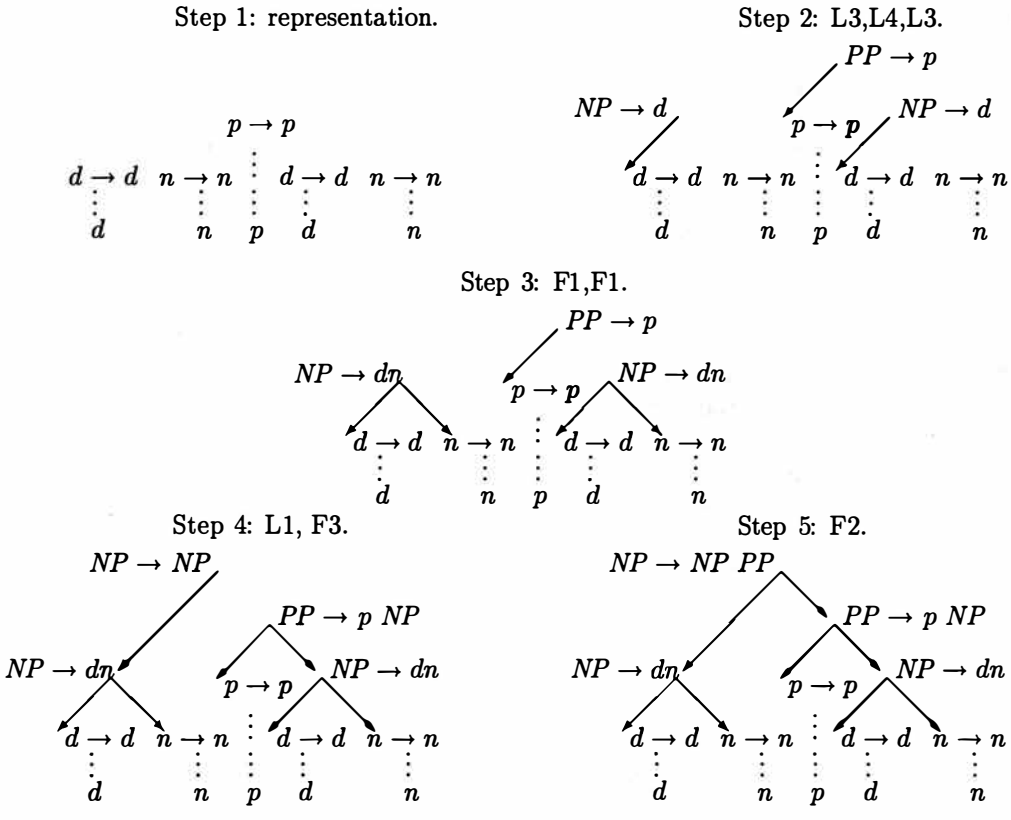

Figure 9: The construction of an analysis tree for *dnpdn*

The simple grammar we consider is shown in the top half of figure 8. The corresponding FB-system is shown in the bottom half of the same figure.

Its set of colors contains an element $x \to x$, for every terminal x of the grammar. It contains also all production rules of the grammar. Finally it contains the 'partial' rules $x \to u$, which are such that $x \to uv$ is a rule of the grammar (both u and v not empty).

The set of root colors of the FB-system contains the colors that are production rules for the start symbol.

Its symbols are the terminals of the grammar.

The representation relation R contains three pairs, one for every symbol.

The idea behind the forward relation F and the lift relation L of the system is, that an analysis tree will always have a color indicating a partially recognized production. The F rules state how to extend partial recognition, the L rules tell how a terminal or a completely recognized non-terminal can be the leftmost symbol in a partially recognized other non-terminal.

The five pictures in figure 9 show the step-by-step construction of an analysis tree for the sequence *dnpdn*. The analysis tree is a parse tree. The steps are:

1. single node trees representing the symbols,
2. three admissible lifts (L3,L4,L3),
3. two admissible forward adoptions (twice F1),
4. an admissible lift (L1), and a forward adoption (F3),
5. an admissible forward adoption (F2).

Actors	$\{d, n, p\}$
Roles	$\{D, N, P, \iota\}$
Leading roles	$\{N\}$
The invisible role	ι
Can-be-played-by	$D : d,\ N : n,\ P : p$
Can-be-combined-left	$D\backslash N : n,\ \iota\backslash D : d,\ \iota\backslash P : p$
Can-be-combined-right	$D : d/\iota,\ N : n/P,\ N : n/\iota,\ P : p/N$
Colors	$\{D : d, \bullet N : n, N : n, P : p\bullet, P : p\}$
Root colors	$\{N : n\}$
Symbols	$\{d, n, p\}$
Represent	$R(D : d, d), R(\bullet N : n, n), R(P : p\bullet, p)$
Forward	(F1) $F(\bullet N : n, P : p, \bullet N : n)$,
	(F2) $F(N : n, P : p, N : n)$,
	(F3) $F(P : p\bullet, N : n, P : p)$,
	(F4) $F(P : p, N : n, P : p)$
Backward	(B1) $B(D : d, \bullet N : n, N : n)$,
	(B2) $B(D : d, N : n, N : n)$
Lift	empty

Figure 10: The example casting- and the corresponding FB-system

2.2 A casting system as an FB-system

In a similar way as the contextfree grammar above, we can present a casting system as an FB-system, and show how the analysis tree construction yields a dependency tree. We take the casting system of the top half of figure 10 as an example.

The corresponding FB-system is in the second half of the same figure.

Its set of colors contains all possible role–actor pairs according to the can-be-played-by relation. Role–actor combinations which cannot do without support, i.e. which cannot be combined with the invisible role, also appear 'dotted' in the color set. The dot marks the side at which support is obligatory.

There is one admissible root color, corresponding to the leading role.

The symbols are, as before, the actors of the casting system.

The representation relation R contains three pairs, one for every symbol.

F has four triples, B has two, and L is the empty relation, no color can be lifted to another.

Note that F and B correspond to the can-be-combined relations. Note also that a dot disappears in recoloring whenever a dotted color adopts a color at the side of the dot.

Note finally that the absence of Lift here and the importance of Lift in the case of contextfree grammars reflect the fact that every node in a dependency tree represents a symbol, whereas in parse trees the internal nodes are representatives of constituents.

The four pictures in figure 11 show the step-by-step construction of an analysis tree for the sequence *dnpdn*. The resulting analysis tree is a dependency tree for the sequence. The steps in its construction are:

1. single node trees, representing the individual symbols,
2. two admissible backward adoptions (twice B1),
3. a forward adoption (F3),
4. a forward adoption (F2).

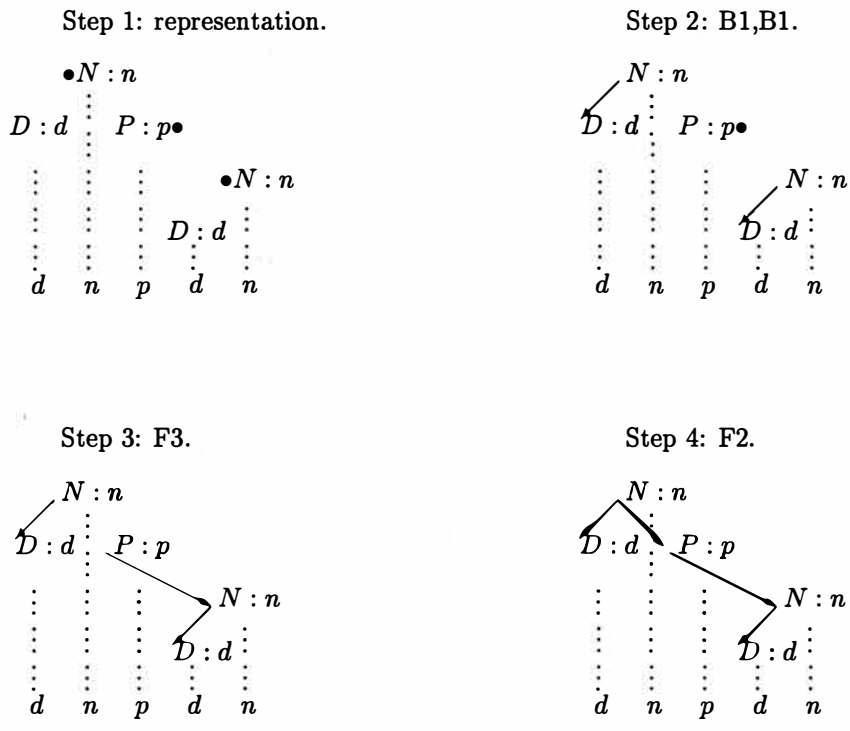

Figure 11: The construction of an analysis tree for *dnpdn*

2.3 Recognizing sequences which have an analysis tree

The two examples are of course not a proof of the fact that every contextfree grammar and every casting system can be represented as an FB-system. Such a proof can be given, it is in fact not difficult. But it is tedious, and we shall not present it here. Hopefully, the examples are enough to suggest the general techniques applicable for translating the one formalism into the other.

The goal of the introduction of FB-systems was to come to a uniform approach to parsing. It is parsing we shall now concentrate on. That is to say, the actual problem of parsing is to construct an analysis tree, if any, or even better, all analysis trees for a given sequence of symbols w.r.t. a given FB-system. What we present here is a strategy for recognition. Strictly speaking, the algorithm we present (not in full algorithmic detail), is capable only of deciding whether a given sequence has an associated analysis tree or not, and it does not produce the trees. But it is a well-known technique, and a minor adaptation to the algorithm, to keep track of the ways in which items are combined during recognition. Such additional information is sufficient to produce all analysis trees.

The central notion in the recognition algorithm is the notion of an item.

Definition 6 An *item* for a given FB-system Φ is an element of the Cartesian product $N \times C$, in which N is the set of positive natural numbers and C is the color set of Φ.

The overall structure of the recognizer is as follows.

The recognizer R w.r.t. Φ is an algorithm which expects an input sequence $(a_1, \ldots, a_n)$ which consists of symbols a_i of Φ.

As a result it produces a sequence of item sets $(I_0, \ldots, I_n)$.

I_0 is empty.

Every next I_{k+1} is computed from the symbol a_{k+1} and the preceding initial segment $(I_0, \ldots, I_k)$ of the result sequence.

The computation of I_{k+1} will yield a set of items (m, c) in which $m \leq k+1$.

The interpretation of $(m, c) \in I_k$ is: there is an analysis tree t for the segment $(a_m, \ldots, a_k)$, with $\gamma(t) = c$.

Recognition is expressed as: the last item set in the sequence produced, i.e. I_n, contains an item $(1, c)$ in which c is an admissible root color of Φ.

3mm

To be precise: the construction of the next item set I_{k+1} from the previous ones $(I_0, \ldots, I_k)$ and the next symbol a_{k+1} proceeds as follows:

It starts with the set

$K = \{(k+1, c) | R(c, a_{k+1})\}$

Of this set, the completion is constructed. The completion of K is the smallest set J satisfying:

- $K \subseteq J$
- If $(j+1, d) \in J$ and $(i, c) \in I_j$ and $F(c, d, c')$, then $(i, c') \in J$
- If $(j+1, c) \in J$ and $(i, d) \in I_j$ and $B(d, c, c')$, then $(i, c') \in J$
- If $(j+1, c) \in J$ and $L(c, c')$ then $(j+1, c') \in J$.

This completion is I_{k+1}.

For the complexity of the construction of the completion, the following is relevant:

every set I_j has a number of elements bounded by $C \times j$, where C is the number of colors,

to construct I_{k+1}, all sets I_j, $j < k+1$, must be traversed, every item in these previous sets must be matched against at most C items already in I_{k+1}, and for every item newly constructed at most C lifts must be added.

It follows that the number of steps in the construction of I_{k+1} is bounded by C^2k^2.

The recognition algorithm is of cubic complexity.

3 Conclusions

The creation of dependency trees for utterances on the basis of a dictionary of word profiles, i.e. a casting system, derived from the handmade analysis of a restricted set of utterances, is an interesting approach to structural analysis. To assess the full merits of the approach, further research is necessary however.

The parsing problem for the construction for dependency trees is in many respects the same as that for contextfree derivation trees. In fact, the general notion of FB-system seems to cover all methods of associating trees with sequences which are local, i.e. all methods where the well-formedness of the associated tree is determined by restrictions on the structure of the nodes, and not on the tree as a whole. An Earley-like algorithm of cubic complexity applies to every association of trees to sequences on the basis of such a general FB-system.

References

[1] Earley, J. (1970), "An efficient contextfree parsing algorithm". *Communications of the ACM* 13, 90–102.

[2] Filmore, C. (1968), "The case for case". In: Bach, E. & R. Harms, (Eds.): *Universals in Linguistic Theory*, 1–68. New York : Holt, Reinhart and Winston.

[3] Hoeven, G.F. van der (1992), "An experiment in the syntactical analysis of English noun phrases". *Memoranda Informatica* 92-24, 34pp. Enschede, The Netherlands : University of Twente, Department of Computer Science.

[4] Hoeven, G.F. van der (1992), "An algorithm for the construction of dependency trees", *Memoranda Informatica* 92-39, 29pp. Enschede, The Netherlands : University of Twente, Department of Computer Science.

[5] Hopcroft, J.E. – J.D. Ullman (1979), *Introduction to Automata Theory, Languages and Computation.* Reading, Massachusetts: Addison-Wesley.

[6] Schubert, K. (1987), *Metataxis—Contrastive Dependency Syntax for Machine Translation.* Dordrecht, Providence R.I. : Foris Publications