Constraint Solving and Language Processing

International Workshop, CSLP 2004
Roskilde University, 1–3 September 2004

Proceedings

Edited by

Henning Christiansen
Peter Rossen Skadhauge
Jørgen Villadsen

Preface

The purpose of the workshop is to provide an overview of activities in the field of Constraint Solving with special emphasis on Natural Language Processing and to provide a forum for researchers to meet and exchange ideas.

Constraint Solving (CS), in particular Constraint Logic Programming (CLP), is a promising platform, perhaps the most promising present platform, for bringing forward the state of the art in language processing. The data subjected to processing via constraint solving may include written and spoken language, formal and semiformal language, and even general input data to multimodal and pervasive systems.

CLP and CS have been applied in projects for shallow and deep analysis and generation of language, and to different sorts of languages. The view of grammar expressed as a set of conditions simultaneously constraining and thus defining the set of possible utterances has influenced formal linguistic theory for more than a decade. CLP and CS provide flexibility of expression and potential for interleaving the different phases of language processing, including handling of pragmatic and semantic information, e.g. ontologies.

This volume contains papers accepted for the workshop based on an open call, contributions from the invited speakers, and abstract of a tuturial. We are very honoured that a selection of highly distinguished researchers in the field has accepted our invitation to talk at the workshop: Philippe Blache, Veronica Dahl, Denys Duchier, and Gerald Penn.

Following the workshop, an edited volume of selected and revised contributions will be produced for wider publication, possibly with other invited contributions in order to cover the field.

We want to thank the program committee, which is listed below, the invited speakers, and all researchers who submitted papers to the workshop. The workshop is supported by the CONTROL project, CONstraint based Tools for RObust Language processing, funded by the Danish Natural Science Research Council; CMOL, Center for Computational Modelling of Language at Copenhagen Business School; and Computer Science Section at Roskilde University, that also hosts the workshop.

The editors
Roskilde, August 2004

Organizers

Henning Christiansen, Roskilde University (chair)
Peter Skadhauge, Copenhagen Business School
Jørgen Villadsen, Roskilde University

Program committee

Troels Andreasen, Roskilde, Denmark
Philippe Blache, Aix-en-Provence, France
Henning Christiansen, Roskilde, Denmark (Chair)
Veronica Dahl, Simon Fraser University, Canada
Denys Duchier, LORIA, France
John Gallagher, Roskilde, Denmark
Claire Gardent, LORIA, France
Daniel Hardt, Copenhagen Business School, Denmark
Peter Juel Henrichsen, Copenhagen Business School, Denmark
Jørgen Fischer Nilsson, Technical University of Denmark
Kiril Simov, Bulgarian Academy of Science
Peter Skadhauge, Copenhagen Business School, Denmark
Jørgen Villadsen, Roskilde, Denmark

Contents

Tutorials

Contributed Short Presentation/Poster

Syntactic Structures as Constraint Graphs

Philippe Blache

LPL-CNRS, Université de Provence
29 Avenue Robert Schuman
13621 Aix-en-Provence, France
pb@lpl.univ-aix.fr

The representation of syntactic information usually makes use of tree-like structures. This is very useful, both for theoretical and computational reasons. However, if such structures are adequate for the representation of simple languages (typically the formal ones), they are not expressive enough for natural language and several difficulties in the processing of NL come from this aspect for different reasons. First, the idea that a complete and homogeneous syntactic structure can be associated to any input is false. In many cases, there is simply no possibility to do this, as illustrated in the example (1) taken from a spoken language corpus.

(1) *monday washing tuesday ironing wednesday rest*

This input is a succession of nouns, without relation given at the syntactic level. It would be very artificial to structure it into a tree. In this example, information making it possible the interpretation comes from the lexical level, eventually prosody, but not from syntax. Similar situations occurs frequently in spoken languages: phenomena such as hesitations, repairs, phatics, etc., have to be taken into account and represented, but trees fail to do this (non connectivity, crossing relations, etc.).

From a theoretical perspective, the conception of linguistic information in terms of hierarchized structures has deep consequences, especially concerning relations between different domains (phonetics, phonology, syntax, semantics, etc). In most of the cases, such relations are given in terms of correspondences between structures. For example, prosody-syntax interaction is explained by means of relations between syntactic trees and prosodic units. This means that both structures have to be built separately before being possible to implement such relations. This is the same problem in the syntax-semantics interface: the semantic structure is usually built started from the syntactic tree, as it is typically the case in Montague grammars. This option imposes a compositional conception of this interface in which, again, syntactic structure has to be built before. Such a conception does not fit with the fact that the interpretation of an utterance consists in bringing together pieces of information coming from different domains.

An alternative approach consists in making it possible to represent such spread and partial information by means of structures that are not necessarily homogeneous, stable and strictly hierarchized. Constraints can play in this perspective an interesting role. Everyone now agrees on the fact that linguistic information can be represented by means of constraints: all modern linguistic

theories make use at one moment or another of constraints. We propose to develop this idea in representing *all* linguistic information by means of constraints. In this way, it becomes possible to exploit constraints not only as filtering process making it possible for example to eliminate unwanted structures (as in OT for example), but as an actual system. A grammar becomes a constraint system and parsing a satisfaction process. What is interesting in this perspective is that hierarchized structures are no more needed. At the difference with HPSG, for example, in which almost all information is stipulated in terms mother/head connection, relations are expressed directly between different objects (features, categories, set of categories, etc.) whatever their function or their role in the structure.

This solution is proposed in the formalism of *Property Grammars*[1] (noted hereafter PG) in which all syntactic information is represented by means of different types of constraints (also called properties): linearity, exclusion, requirement, uniqueness, dependency and obligation. In our approach, the objects taken into consideration in the grammar are constructions. A construction, as proposed in the *Construction Grammar* theory of Fillmore, can be any kind of syntactic object: a category (e.g. Det, NP), a clause, a specific turn (e.g. interrogative, subject-auxiliary inversion), etc. Each construction is described by a set of properties that can come from different linguistic domains, implementing then directly interaction between them, without needing to build separately different structures. In this approach, a grammar, which is a set of constructions, consists then in a set of constraints. The basic mechanism consists for each construction to verify what are the satisfied and violated constraints. We obtain then a precise description, that we call *characterization*, for any kind of category, whatever the form of the input to be parsed.

A complete characterization is built in two stages. The first consists, starting from the set of categories corresponding to the input that is considered as an assignment, to go through the entire constraint system, evaluating all constraints that can be applied to this assignment. More precisely, different assignments are built corresponding to the possible subsets of categories. A characterization is associated to each assignment. The process consists then to check whether such assignments correspond to constructions described in the grammar. This consists in verifying that the characterization (i.e. the set of violated and satisfied constraints) is included in the description of the construction (i.e. the set of constraints describing it). When this condition is verified, then the construction is activated and the subsystem of constraints corresponding to the construction is to its turn evaluated. At this second stage, constraints specific to a construction can be then ev aluated and, when the construction corresponds to a category, it is instantiated.

The result of this process is a set of characterizations, in other words, a set of relations between different objects (at different levels). This structure corresponds to a network of constraints, or a graph, not necessarily connected. What is interesting is that all constraints are at the same level (even though it

[1] For a short presentation of this formalism, see [Blache & Prost 04] in this volume.

possible to weight them), which means that it is always possible to characterize an object or, in other words, all constraints can be evaluated independently from the others and without any priority order. This is a non-holistic way of representing information that makes it possible to merge constraints coming from different domains (which is the main interest of constructions).

An Abductive Treatment of Long Distance Dependencies in CHR

Veronica Dahl

Logic and Functional Programming Group
Department of Computing Science
Simon Fraser University
Burnaby, B.C., Canada
veronica@cs.sfu.ca

Abstract. We propose Abductive Concept Formation Grammars, a CHR methodology for treating long distance dependencies by abducing the missing elements while relating the constituents that must be related. We discuss our ideas both from a classical analysis point of view, and from a property-grammar based perspective. We exemplify them through relative clauses first, and next through the more challenging case of natural language coordination.

1 Introduction

One of the main challenges in computational linguistics is the relating of constituents that can be arbitrarily far apart within a sentence. For instance, to understand "logic, we love", we must understand that "logic" functions as the (dislocated) direct object of "love" (in the canonical, subject-verb-object order, we would say: we love logic). The dislocated phrase, "logic", can be arbitrarily far away from the point in which the direct object it represents would normally appear, cf. "logic, he thinks we love", "logic, they told me that he thinks we love", and so on.

Making sense of sentences with long distance dependencies, in which constituents must be related that may be separated by an arbitrarily long sequences of intervening words, is one of the most difficult computational linguistic problems. Not only must the relationship be established despite arbitrary distances between the relating elements, but this type of construction often involves "guessing" material that is left implicit. For instance, in "the workshop was interesting and the talks inspiring", we must understand the verb of the second conjoint to be "were" (i.e., the talks were inspiring), even though it does not appear explicitly.

The advantages of bottom-up approaches to NL processing aimed at flexibility have been demonstrated within declarative paradigms for instance for parsing incorrect input and for detecting and correcting grammatical errors [2,3,6,21]. Most of these approaches use constraint reasoning of some sort, sometimes

blended with abductive criteria. For the specific problem of coordination in natural language, datalog approaches with constraints placed upon word boundaries have shown to be particularly adequate [15,26].

In this article we introduce Distant Concept Formation Grammar rules, a CHR based methodology for treating long distance dependencies by abducing the missing elements while relating the constituents that must be related. We provide the necessary background, assuming some knowledge of CHR but relating them particularly to language processing. Our approach is discussed both from a classical analysis point of view, and from a property-grammar based perspective. We exemplify our ideas through relative clauses first, and next through the more challenging case of natural language coordination.

2 Background

2.1 CHR and parsing; CHRG

Parsing problems can be expressed in CHR [19] by explicitly manipulating input and output strings: A string to be analyzed such as *"the house collapsed"* is entered as a sequence of constraints
`{token(0,1,the), token(1,2,house), token(2,3,collapsed)}` that comprise an initial constraint store. The integer arguments represent word boundaries, and a grammar for this intended language can be expressed in CHR as follows.

```
token(X0,X1,the) ==> det(X0,X1,sing).
token(X0,X1,house) ==> n(X0,X1,sing).
token(X0,X1,collapsed) ==> v(X0,X1,sing).
n(X0,X1,Num), v(X1,X2,Num) ==> s(X0,X1,Num).
```

CHRGs [10,12] include an automatic insertion and handling of word boundaries. Their rules can be combined with rules of CHR and with Prolog, which is convenient for defining the behaviour of non-grammatical constraints. As well, CHRGs provide a straightforward implementation of assumptions [16,14] and have recently been shown to be very well suited for abductive logic programming [9]. They are in particular useful for datalog renditions of language: as shown in [6], the datalog part requires very little implementation machinery when using CHRGs: basically, grammar rules in CHRGs can be made to work either top-down or bottom-up according to the order in which their left and right hand sides are given.

CHRG includes also notation for gaps and for parallel matching, which we neither describe nor use in the present work.

CHR/CHRG applications to natural language processing include [1], which flexibly combines top-down and bottom-up computation in CHR, [6], which uses CHR to diagnose and correct grammatical errors, and [13], which implements property grammars using CHRG.

2.2 Extending CHR with Abduction and Assumptions

In [9], a *negation-free* abductive logic program is defined as one which has no application of negation, whose abductive predicates are distinguished by prefix exclamation marks (so, e.g., `p` and `!p` refer to different predicates), and whose integrity constraints are written as CHR propagation rules whose head atoms are abducibles and whose body atoms are abducibles or built-ins (or possibly `fail`).

In this approach, abducibles are viewed as constraints in the sense of CHR, the logic program is executed by the Prolog system and whenever an abducible is called it is added automatically by CHR to the constraint store and CHR will activate integrity constraints whenever relevant. The complete implementation in SICStus Prolog is provided by including the following lines in the start of a the program file.[1]

```
:- use_module(library(chr)).
:- op(500,fx,!).
handler abduction.
constraints ! /1.
```

Assumptions [16,14] can also interact with abduction. These are basically backtrackable assertions which can serve among other things to keep somewhat globally accessible information (an assertion can be used, or consumed, at any point during the continuation of the computation). The following version of append/3 exemplifies. The addition and subtraction signs are used respectively to indicate assumption and consumption:

```
append(X,Y,Z):- +global(Y), app(X,Z).

app([],Y):- -global(Y).
app([X|L],[X|Z]):- app(L,Z).
```

Here the assumption global/1 keeps the second argument in store until the recursive process hits the empty clause, at which point it is consumed to be incorporated at the end of the resulting list.

2.3 Concept Formation Grammars and our basic parsing methodology

In [7], we introduced the cognitive model of Concept Formation, which has been used for oral cancer diagnosis [8] and specialized into grammatical concept formation, with applications to property grammar parsing [13].

The grammatical counterpart of Concept Formation, namely Concept Formation Grammars, or CFGs, are extensions of CHRGs which dynamically handle properties between grammar constituents and their relaxation as statically defined by the user.

[1] The prefix exclamation mark is used as a "generic" abducible predicate.

An Example Let's first exemplify within the traditional framework of rewrite rules which implicitly define a parse tree. Whereas in CHRG we would write the following toy grammar [2]

```
[a] ::> determiner(singular).
[boy] ::>  noun(singular).
[boys] ::>  noun(plural).
[laughs] ::>  verb(singular).


determiner(Number), noun(Number), v(Number) ::>  sentence(Number).
```

to parse correct sentences such as "a boy laughs", in CFGs we can replace its last rule by the following CFG rules, so the system will accept also sentences which do not agree in number, while pointing out the error as a side effect of the parse:

```
determiner(Ndet), noun(Nn), v(Nv) =>
                  acceptable(prop(agreement,Ndet,Nn,Nv,N),_) |
    sentence(N).

prop(agreement,[Ndet,Nn,Nv,N],true):- Ndet=Nn,
   Nn=Nv, !,  N=Nv.
prop(agreement,[Ndet,Nn,Nv],mismatch).

relax(agreement).
```

The binary predicate "acceptable" is a primitive predicate to the Concept Formation system, which succeeds if its second argument has been bound to "true", or it has been bound to something else but the property has been declared by the user as relaxable. In the case of our example, the agreement property will appear in the list of violated properties automatically constructed as a result of the parse.

3 Treating Long distance dependencies: through assumptions, through abductive concept formation

Typically, treatments of long distance dependencies involve either enriching the linguistic representation using explicit rules of grammar (the grammatical approach) or adding special mechanisms to the parsing algorithms (the metagrammatical approach). The latter is more concise, in that for instance explicit rules for coordination are not needed in the grammar itself, but sentence coordination can be inferred from a metagrammatical component plus the user's grammar (see for instance Woods[27], Dahl and McCord[17], Haugeneder[20] and Milward[23]).

[2] terminal symbols are noted between brackets as in DCGs

3.1 Through Assumption Grammars

The traditional grammar rule approach can be exemplified by the following (assumption) grammar rules for relativization, where we omit all other arguments to focus on the information to be carried long distance, namely the variable X:

```
noun_phrase(X) --> det, noun(X), {+antecedent(X)}, relative, {!}.
noun_phrase(X) -->  {-antecedent(X)}.

relative --> relative_pronoun,  sentence.

sentence --> noun_phrase(X), verb_phrase(X).
```

The first rule uses an assumption to store the antecedent for a noun phrase that will go missing in the relative clause. The second rule picks up a missing noun phrase's representation through consuming the antecedent left by an overt noun phrase. The third rule defines a relative clause as a relative pronoun followed by a sentence. Note that any missing noun phrase inside that sentence can now be reconstructed from the assumption of its antecedent. The same rules serve therefore for relativizing on a subject (as in "the house that fell"), on an object (as in "the house that Jack built"), on an indirect object ("the student that Mary gave a book to"), etc.

3.2 Through Abductive Concept Formation

We now present our Abductive Concept Formation methodology through the example of abducing the missing noun phrase in a relative clause. Because in bottom-up mode we cannot deal with empty rules [3], we must pay the price of making separate rules for each type of relativization, but the resulting grammar does not need to use assumptions. All we need to do is to retrieve the end point of the previous constituent and reconstruct (abduce) the missing noun phrase at that point, e.g.:

```
relative_pronoun, verb_phrase(X):(P1,P2) :: >
                                          !noun_phrase(X):(P2,P2),
            relative.
relative_pronoun, noun_phrase, verb:(P1,P2) :: >
                                          !noun_phrase(Y):(P2,P2),
            relative.
```

These rules produce a relative clause while abducing its missing noun phrase. The fact that the abduced phrase is syntactically marked as abduced (as opposed to the previous example, in which noun phrases look the same whether they have

[3] notice that the second rule above is, grammatically speaking, empty, because its right hand side is not a grammar symbol but a Prolog test

been reconstructed or are explicit) makes it easier to reconstruct further implicit meanings for more sophisticated examples, as we shall see next. Note that the start and end points of the abduced noun phrase are the same, as befits a non-overt noun phrase.

4 A non trivial example: coordination in natural language

In this section we briefly present property based grammars [4,5], next we summarize our parsing methodology for them (see complete details in [13]) and we then extend it with our abductive concept formation, metagrammatical treatment of coordination.

4.1 Parsing Property Grammars

Property based Grammars [4,5] define any natural language in terms of a small number of properties: linear precedence (e.g. within a verb phrase, a transitive verb must precede the direct object); dependency (e.g., a determiner and a noun inside a noun phrase must agree in number), constituency (e.g. a verb phrase can contain a verb, a direct object,...), requirement (e.g. a singular noun in a noun phrase requires a determiner), exclusion (a superlative and an adjectival phrase cannot coexist in a noun phrase), obligation (e.g. a verb phrase must contain a verb), and unicity (e.g. a prepositional phrase contains only one preposition). The user defines a grammar through these properties instead of defining hierarchical rewrite rules as in Chomskyan based models. In addition, properties can be relaxed by the user in a simple modular way. For instance, we could declare "precedence" as relaxable, with the effect of allowing ill-formed sentences where precedence is not respected, while pointing out that they are ill-formed.

The result of a parse is, then, not a parse tree per se (although we do provide one, just for convenience, even in the case of ill-formed input), but a list of satisfied and a list of unsatisfied properties.

4.2 The basic parser

Our basic methodology relies upon a single rule which successively takes two categories (one of which is a phrase or a phrase head), checks the properties between them, and constructs a new category by extending the phrase with the other category, until no more categories can be inferred. Lists of satisfied and unsatisfied properties are created by the rule, using property inheritance (a detailed analysis of which can be seen in the original paper). Its form is described in Fig. 1.

This rule first tests that one of the two categories is of type `XP` (a phrase category) or obligatory (i.e., the head of an `XP`), and that the other category is an allowable constituent for that `XP`. It then successively tests each of the PG properties among those categories, incrementally building as it goes along the lists of satisfied and unsatisfied properties. Finally, it infers a new category of

```
cat(Start1,End1,Cat,Features1,Graph1,Sat1,Unsat1),
cat(End1,End2,Cat2,Features2,Graph2,Sat2,Unsat2) ==>
    xp_or_obli(Cat2,XP), ok_in(XP,Cat),
    precedence(XP,Start1,End1,End2,Cat,Cat2,Sat1,Unsat1,SP,UP),
    dependency(XP,Start1,End1,End2,Cat,Features1,Cat2, Features2,SP,UP,SD,UD),
    build_tree(XP,Graph1,Graph2,Graph,ImmDaughters),
    unicity(Start,End2,Cat,XP,ImmDaughters,SD,UD,SU,UU),
    requirement(Start,End2,Cat,XP,ImmDaughters,SU,UU,SR,UR),
    exclusion(Start,End2,Cat,XP,ImmDaughters,SR,UR,Sat,Unsat)
    | cat(Start1,End2,XP,Features2,Graph,Sat,Unsat).
```

Fig. 1. New Category Inference

type `XP` spanning both these categories, with the finally obtained `Sat` and `Unsat` lists as its characterization.

In practice, we need another rule symmetric to this one, in which the XP category appears before the category Cat which is to be incorporated into it.

4.3 Extending the parser for abducing implicit elements

Our Abductive Concept Formation methodology imposes two requirements on the user's grammar. First, semantics must be defined compositionally. Second, semantic material must be isolated into one specific argument, so that the rules for virtual coordination can easily identify and process it. For instance, if we use the second argument for semantics, a rule such as

```
name(X) -> np(X^Sem^Sem).
```

should be coded in CHR and CHRG as

```
CHR:   category(name,X,P0,P1) ==> constituent(np,X^Sem^Sem,P0,P1).
CHRG:  category(name,X) ::> constituent(np,X^Sem^Sem).
```

We assume that there are two (implicit or explicit) coordinating constituents, C1 and C2, surrounding the conjunction, which must in general be of the same category [4]. As in Dahl's previous work [15], we adopt the heuristics that closer scoped coordinations will be attempted before larger scoped ones. Thus in Woods' well-known example[27], *"John drove his car through and demolished a window"* , `"vp conj vp"` is tried before `"sent conj sent"` .

If both C1 and C2 are explicit, we simply postulate another constituent of the same category covering both, and conjoin their meanings. This is achieved through the rule:

```
constituent(C,Sem1,P0,P1),
constituent(conj,_,P1,P2),
constituent(C,Sem2,P2,P3) ==>
                        constituent(C,Sem,P0,P3),conj(Sem1,Sem2,Sem).
```

`Sem` can take either the form `and(Sem1,Sem2)` or a more complex form.

[4] This is a simplifying assumption: coordination can involve different categories as well, but in this paper we only address same category coordination

If either C1 or C2 is implicit, the CHRG engine will derive all possible partial analyses and stop with no complete sentence having been parsed. We can then resume the process after dynamically adding the following symmetric rules, in charge of completing the target in parallel with the source. Once the target has been completed, the above rule for coordinating complete constituents can take over.

```
% The second conjoint is incomplete
     constituent(C,Sem1,P0,P1),
     constituent(conj,_,P1,P2) ==>
                              complete(C,Sem1,range(P0,P1),Sem2,P3)
                         |    constituent(C,Sem2,P2,P3).
% The first conjoint is incomplete
     constituent(conj,_,P1,P2),
     constituent(C,Sem2,P2,P3) ==>
                              complete(C,Sem2,range(P2,P3),Sem1,P0).
                         |    constituent(C,Sem1,P0,P1).
```

complete/5 generates the features of a constituent of category C that is incomplete between the given points, using the source (the portion of text indicated by range/2) as parallel structure. The new constraint is placed in the constraint store as an abduced constraint, so that the rule for complete conjoint coordination can apply. This rule must therefore be modified to allow for one of the constituents to be abduced.

An Example For *Philippe likes salsa and Stephane tango*, we have the initial constraint store:
`{philippe(0,1),likes(1,2),salsa(2,3),and(3,4),stephane(4,5),tango(5,6)}`
to which the following "constraints" (in the sense of the CHR store) are successively added: `[ name(0,1),verb(1,2),noun(2,3),conj(3,4),name(4,5),noun(5,6), np(0,1),vp(1,3),np(2,3),np(4,5),np(5,6),s(0,3) ]`
At this point, since the analysis of the entire sentence is not complete, the dynamically added rules will compare the data to the left and right of the conjunction, noting the parallelisms present and absent:

```
 np(0,1) parallel to np(4,5)
 verb(1,2) parallel to ?
 np(2,3) parallel to np(5,6)
```

and postulate(abduce) a verb(5,5), with the same surface form ("likes") as the verb in the parallel structure. The addition of this element to the constraint store triggers the further inferences: `{vp(5,6), s(4,6)}`. This in turn will trigger the complete constituent coordination rule, resulting in `{s(0,6)}`

Top-down Prediction For cases in which the two parallel structures are different, the simple pairing of constituents in the source and target, as above, will not be enough. For example *"John drove a car through and demolished a window"*, for instance, we have the constraint store:

```
 { john(0,1), drove(1,2), a(2,3), car(3,4),
```

```
through(4,5), and(5,6), demolished(6,7),
a(7,8), window(8,9), name(0,1), verb(1,2),
det(2,3), noun(3,4), prep(4,5), conj(5,6),
verb(6,7), det(7,8), noun(8,9), np(0,1),
np(7,9), vp(6,9) }
```

In such cases we examine the grammar rules in top-down fashion to determine which ones would warrant the completion of a constituent that appears to one side of the conjunction but not to the other.

In our example, the candidate sources are: prep(4,5), verb(6,7) and vp(6,9). Postulating a missing prep at point 6 or a missing verb at point 5 does not yield success, so we abduce a vp ending in point 5 and use vp rules top-down to predict any missing constituents. The pp rule is eventually found, which rewrites pp into a preposition plus a noun phrase, so we can abduce a missing noun phrase between point 5 and itself, to be filled in by the parallel noun phrase in the source, namely *"a window"*(we must avoid requantification, so the window driven through and the one demolished must be the same window). This new noun phrase triggers in turn the abduction of a verb phrase between points 1 and 5, which in turn allows us to conjoin the two verb phrases, and complete the analysis.

Semantics We now turn our attention to grammars which construct meaning representations for the sentences analysed.

After having determined the parallel elements in source and target, we must void the source's meaning (using for instance higher-order unification) from those elements which are not shared with the target, and only then apply the resulting property on the representation of the target.

For our example, we must from the meaning representation of *"Philippe likes salsa"* reconstruct the more abstract property[λy.λx.likes(x,y)], which can then be applied on *"tango"* and *"stephane"* to yield likes(stephane,tango).

We bypass this need by keeping copies of the abstract properties as we go along. While the original properties get instantiated during the analysis (so that the meaning of *"likes"* in the context *"Philippe likes salsa"* becomes `likes(philippe,salsa))` , their copies do not. It is these uninstantiated copies that are used as meanings for the reconstructed targets.

Syntactic Considerations Some of the syntactic features carried around by typical grammars need a special treatment by the coordination rules: the conjunction of two singular noun phrases (e.g. *"the cat and the dog"*), for instance, should result in a plural conjoined noun phrase.

5 Concluding remarks

We have discussed abduction as an appropriate method for treating long distance dependencies, both within traditional, hierarchical approaches, and within

property-based kinds of grammars, so that even incomplete or incorrect input will yield interesting, if partial, results.

The present work was inspired by [15], where we provided a left-corner plus charting datalog approach. This approach recorded parse state constituents through linear assumptions to be consumed as the corresponding constituents materialize throughout the computation. Parsing state symbols corresponding to implicit structures remained as undischarged assumptions, rather than blocking the computation as they would if they were subgoals in a query. They could then be used to glean the meaning of elided structures, with the aid of parallel structures.

While being quite minimalistic in the amount of code required, this approach involved sophisticated process synchronization notions, as well as the use of linear affine implication. In the present work we showed how the use of abduction helps retain the advantages of [15] while drastically simplifying our methodology.

Various authors, e.g.[22] discuss an alternative approach to anaphoric dependencies in ellipsis, in which the dependence between missing elements in a target clause and explicit elements in a source clause does not follow from some uniform relation between the two clauses, but follows indirectly from independently motivated discourse principles governing pronominal reference. While containing linguistically deep discussions, the literature on this discourse-determined analysis also focusses on ellipsis resolution, while still leaving unresolved (to the best of our knowledge) the problem of automatically determining which are the parallel structures.

On another line of research, Steedman's CCGs [25] provide an elegant treatment of a wide range of syntactic phenomena, including coordination, which does not resort to the notions of movement and empty categories, instead using limited combinatory rules such as type raising and functional composition . However, these are also well known to increase the complexity of parsing, originating spurious ambiguity- that is, the production of many irrelevant syntactic analyses as well as the relevant ones. Extra work for getting rid of such ambiguity seems to be needed, e.g. as proposed in [24].

With this work we hope to stimulate further research into the uses of abduction in constraint-based parsing.

Acknowledgements

This research was made possible by the author's NSERC research grant.

References

1. Abdennadher, S. and Schtz, H. CHR: A Flexible Query Language. In International conference on Flexible Query Answering Systems, FQAS'98, LNCS, Springer, Roskilde, Denmark (1998)
2. Balsa, J., Dahl, V. and and Pereira Lopes, J. G. Datalog Grammars for Abductive Syntactic Error Diagnosis and Repair. In Proceedings of the Natural Language Understanding and Logic Programming Workshop, Lisbon (1995)

3. Blache, P. and Azulay, D. Parsing Ill-formed Inputs with Constraints Graphs. In A. Gelbukh (ed), Intelligent Text Processing and Computational Linguistics, LNCS, Springer, 220–229 (2002)
4. Bès G. and Blache, P. Propri/'et/'es et analyse d'un langage. In Proceedings of TALN'99 (1999)
5. Bès G., Blache, P., and Hagège, C. The 5P Paradigm. Research report, GRIL/LPL (1999)
6. Christiansen, H. and Dahl, V. Logic Grammars for Diagnosis and Repair. In Proceedings of 14th IEEE International Conference on Tools with Artificial Intelligence, Washington D.C., 307–314 (2002)
7. Dahl V. and Voll K. (2004) "Concept Formation Rules: an executable cognitive model of knowledge construction", in proceedings of *First International Workshop on Natural Language Understanding and Cognitive Sciences*, INSTICC Press.
8. Barranco-Mendoza, A., Persaoud, D.R. and Dahl, V. (2004) "A property-based model for lung cancer diagnosis", in proceedings of *8th Annual Int. Conf. on Computational Molecular Biology, RECOMB 2004*, San Diego, California (accepted poster).
9. Christiansen, H. and Dahl, V. Assumptions and Abduction in Prolog. In Proceedings MULTICPL'04 (Third International Workshop on Multiparadigm Constraint Programming Language, Saint-Malo, France (2004)
10. Christiansen, H. Logical Grammars Based on Constraint Handling Rules,(Poster abstract). In Proc. 18th International Conference on Logic Programming, Lecture Notes in Computer Science, 2401, Springer-Verlang, p. 481 (2002)
11. Christiansen, H. Abductive Language Interpretation as Bottom-up Deduction. In Proc. NLULP 2002, Natural Language Understanding and Logic Programming, Wintner, S. (ed.), Copenhagen, Denmark, 33–48 (2002)
12. Christiansen, H. CHR as Grammar Formalism, a First Report. In Sixth Annual Workshop of the ERCIM Working Group on Constraints, Prague (2001)
13. Dahl, V. and Blache, P. Directly Executable Constraint Based Grammars. In Proc. Journees Francophones de Programmation en Logique avec Contraintes, Angers, France (2004).
14. Dahl, V. and Tarau, P. Assumptive Logic Programming. In Proc. ASAI 2004, Cordoba, Argentina (2004).
15. Dahl, V. On Implicit Meanings. In Computational Logic: from Logic Programming into the Future. F. Sadri and T. Kakas (eds), Springer-Verlang (2002)
16. Dahl, V., Tarau, P., and Li, R. Assumption Grammars for Processing Natural Language. In Fourteenth International Conference on Logic Programming, MIT Press, 256–270 (1997)
17. Dahl, V. and McCord, M. Treating Coordination in Logic Grammars. In American Journal of Computational Linguistics 9, 69–91 (1983)
18. Darlymple, M., Shieber, S., and Pereira, F. Ellipsis and Higher-Order Unification. In Linguistics and Philosophy, 14(4), 399–452 (1991)
19. Fruhwirth, T. W. Theory and Practice of Constraint Handling Rules. In Journal of Logic Programming, 37, 95–138 (1998)
20. Haugeneder, H. A Computational Model for Processing Coordinate Structures: Parsing Coordination will-out Grammatical Specification. In ECAI 1992, 513–517 (1992)
21. Kakas, A.C., Michael, A., and Mourlas, C. ACLP: Abductive Constraint Logic Programming. The Journal of Logic Programming, Vol.44, pp. 129–177, 2000.
22. Kehler, A. and Shieber, S. Anaphoric Dependencies in Ellipsis. In Computational Linguistics, 23(3) (1997)

23. Milward, D. Non-Constituent Coordination: Theory and Practice In Proceedings of COLING 94 , Kyoto, Japan, 935-941 (1994)
24. Park, J.C and Cho, H.J. Informed Parsing for Coordination with Combinatory Categorial Grammar. COLING'00, pp. 593–599, (2000)
25. Steedman, M. Gapping as Constituent Coordination. In Linguistics and Philosophy (1990)
26. Voll, K., Yeh, T., and Dahl, V. An Assumptive Logic Programming Methodology for Parsing. In International Journal on Artificial Intelligence Tools (2001)
27. Woods, W. An Experimental Parsing System for Translation Network Grammars. In R. Rustin, editor, Natural Language Processing, Algorithmic Press, New York, 145–149 (1973)

Invited Talk
(no title provided at time of printing)

Denys Duchier

Équipe Calligramme, LORIA, Nancy, France
duchier@loria.fr

The Other Syntax

Gerald Penn

Department of Computer Science
University of Toronto
10 King's College Rd.
Toronto M5S 3G4
Ontario, Canada
gpenn@cs.toronto.edu

Abstract. The ultimate goal of linguistics is to explain the relationship between the form of sound (or text) and the substance of meaning. Almost 50 years ago now, Artificial Intelligence promised that computers were capable of mimicking and possibly even learning this correspondence in a way that would revolutionize the manner in which we interact with them. To put it mildly, we have not fulfilled that promise yet, and the reason we have not has at least something to do with where most of us have been looking for the answer: to a view of syntactic phrase structure that has, at best, a circuitous connection to meaning. Even within the Chomskyan linguistic tradition, there has been an acknowledgement that a "logical form" quite unlike syntactic structure must exist. But logical form is still a "form," a different syntax for talking *about* meaning in the absence of any kind of inference that could justify viewing it *as* meaning.

This talk will describe a research programme in progress that seeks to connect grammar, in the conventional sense that computational linguists use that term, with real semantic inference, and thus a real semantics. In our approach, this connection is made by viewing the syntactic, logical form, and semantic components of grammar as instances of common methods in constraint logic programming.

Gradience, Constructions and Constraint Systems

Philippe Blache[1] and Jean-Philippe Prost[2, 1]

[1] LPL-CNRS
Université de Provence
29 Avenue Robert Schuman
13621 Aix-en-Provence, France
pb@lpl.univ-aix.fr
[2] Centre for Language Technology
Macquarie University
Sydney NSW 2109, Australia
jpprost@ics.mq.edu.au

Abstract. One important question to be addressed by modern linguistics concerns the variability of linguistic constructions. Some of them are very regular, some others quite rare. Some are easy to explain, some others hard. And finally, some are canonical whereas some others are less grammatical. These different aspects have been addressed, usually separately, from a psycholinguistic perspective. Some elements of explanation are given in terms of sentence complexity and gradience (see for example [Gibson00], [Sorace04]). It is necessary to explain why some utterances can be interpretable more easily than some others, or why some are less acceptable than others.
Several linguistic theories address explicitly such questions, in particular from the perspective of dealing with ill-formed inputs. Some elements of answer can be found for example in the *optimality theory* (see [Prince93]), in the *model-theoretic syntax* approach (see [Pullum03]) or in *construction grammar* (see [Fillmore98], [Goldberg95]). One of the main challenges in these approaches, is the characterization of gradience in linguistic data. The basic idea consists in hierarchizing the linguistic information according to some "importance" criterion. However, such importance is difficult to define. In some approaches such as probabilistic grammars, it relies on frequency information (see [Keller03]): each rule is weighted according to its frequency acquired on treebanks. The syntactic constructions are specified according to the weights of the different rules. In some other approaches, explored in this paper, the idea is to propose some objective information relying on a symbolic representation.
This paper argues in favor of a fully constraint-based approach representing all kind of information by means of constraints. Such an approach makes it possible to quantify the information and proposes an ordering relation among the different utterances relying on the interpretation of satisfied and violated constraints.

1 Constructions

Several modern approaches in linguistics argue in favor of a contextual description of linguistic phenomena. This way of representing information is typically proposed in construction grammar (see [Fillmore98] or [Kay99]), in property grammars (see

[Blache00]), etc. In these approaches, a specific phenomenon is characterized by a convergence of different properties. For example, taking into account the syntactic level alone, a dislocated construction is roughly characterized by the realization of an NP, before or after a main clause, together with an anaphoric clitic inside this clause. Other properties can be added to this framework, for example concerning the lexical selection that can be specific to this construction. The important point is that a construction is specified by a given set of properties, some of them being relevant only for this construction. In other words, given the fact that, as proposed in property grammars, a property can be conceived as a constraint, a construction is defined by a constraint system: what makes sense is not a property taken separately from the others, but the interaction of the constraints. Contextuality is then implemented by means of such interaction: defining a construction consists in specifying on one hand a set of properties characterizing the construction and on the other hand some property subset that specifically entails some properties. For example, in the description of passive in French, an accusative pronoun in the VP has to agree with the subject (has to be reflexive).

(1) Je me le suis dit (*I myself it aux-1st tell*)
(2) *Je te le suis dit (*I you it aux-1st tell*)

Such an approach presents several advantages. First, it is possible to express constraints at very different granularity levels for the specification of a given construction. Moreover, different kinds of constraints, coming from different linguistic domains such as prosody, semantics, pragmatics, etc. can participate to the definition of a construction. It is one of the main arguments in favor of construction approaches. This aspect is illustrated in the following example (from [Mertens93]) illustrating a very specific construction in which only a little information is available at the syntactic level. In this case, prosody plays an important role in its interpretation:

(3) lundi lavage mardi repassage mercredi repos
monday washing tuesday ironing wednesday rest

Finally, a constraint can have in such perspective a relative importance. For some constructions, a constraint can be obligatory whereas the same constraint can be easily relaxed in some other cases.

2 Constraints

Classically, constraints are used in linguistic theories as a filtering process. This is typically the case with constraint grammars, but also with most recent constraint-based approaches such as HPSG (see [Sag99]) or Optimality (see [Prince93]). In HPSG for example, constraints are applied to a structure in order to verify its well-formedness. As a side effect, constraints can also implement feature values instantiation or propagation. The valence principle for example plays exactly this double role: ruling out the structures that don't satisfy the constraint and, in case of unification between a structure and a description in the valence list, instantiating some feature values of the structure. In this case, constraints are seen as structure descriptions, they don't implement information that can be possibly evaluated independently from these structures. This means that structures are first to be built before verifying their values and syntactic properties are expressed in terms of relations inside such hierarchized constructions.

Constraints are used in a completely different way in OT. They also constitute a filtering process, but the constraints belong to a system containing two main pieces of information: the basic information specified by the constraint itself, expressed in universal and imperative terms and a second-level (rather implicit) information expressed by ranking. In such system, the fact that a constraint is satisfied or not is in the end less important than its position in the ranking. Moreover, all constraints are stipulated taking into account the fact that other opposite constraints also belong to the system. This is a kind of negative way of using constraints that are in fact stipulated so as to be violated.

There is a common basis of these different uses of constraints. In all cases, they need to be interpreted into a system. In other words, they cannot be evaluated for themselves, but in reference to the entire system. This is what Pullum underlines as a holistic way of seeing syntactic information in which a syntactic property cannot be interpreted in itself. This is a limitation in the perspective of finding syntactic characterizations of unrestricted material: in many cases, especially when parsing spoken languages, syntactic information is sparse. For example in (3), the relation among the different elements is difficult to express at the syntactic level.

In such cases, information comes from the interaction of different linguistic domains, in particular morphology and prosody more than other kinds of information. And in such cases, classical approaches fail to build a description. There is also another drawback. In the case of OT for example, ranking makes it possible to order different candidates. Such ranking expresses a level of well-formedness, according to the grammar. However, there is no direct relation between well-formedness and more general notions such as understandability, acceptability, sentence complexity, etc. What is important to explain from a cognitive point of view is what kind of utterances are more easily interpretable and why.

We think that constraints can play an important role in this perspective provided that they are expressed in a non holistic manner. In such a perspective, each constraint must implement a syntactic property and be expressed independently from the others. Obviously, constraints have to interact, but they can always be evaluated. This characteristic is underlined by Pullum as being one of the interests of a model-theoretical approach in comparison with a deductive one: it is possible to give some information in all cases, whatever the input form.

2.1 Property Grammars

We briefly describe here a framework for such an approach called *property grammars* (see [Blache00]). In this approach, all information is represented by means of constraints (also called properties). These constraints are relations among categories expressed as a set of features. A category, at the difference with HPSG, doesn't contain hierarchical information among constituents, but only what is called in construction grammar *intrinsic* information. All constraints are expressed independently from the others and represent a specific kind of syntactic information:

- linear precedence, which is an order relation among constituents,
- subcategorization, which indicates the co-occurrence relations among categories or sets of categories,

- the impossibility of co-occurrence between categories,
- the impossibility for a category to be repeated,
- the minimal set of obligatory constituents (usually one single constituent) which is the head,
- the semantic relations among categories, in terms of dependency.

These different kinds of information correspond to different properties, respectively: *linearity, requirement, exclusion, unicity, obligation, dependency*. Such information can always be expressed in terms of relations among categories, as shown in the following examples:

- Linear precedence: *Det* $\prec$ *N* (a determiner precedes the noun)
- Dependency: *AP* $\rightsquigarrow$ *N* (an adjectival phrase depends on the noun)
- Requirement: *V[inf]* $\Rightarrow$ *to* (an infinitive comes with *to*)
- Exclusion: *most* $\not\Rightarrow$ *Adj[super]* (*most* can not modify an adjective that already is a superlative.)

Here is a more formal representation of such information :
let $\mathcal{K}$ be a set of categories, $\mathcal{A}$ be the ordered set of categories for a given input,
let $pos(\mathcal{C}, \mathcal{A})$ be a function that returns the position of $\mathcal{C}$ in $\mathcal{A}$,
let $card(\mathcal{C}, \mathcal{A})$ be a function that returns the number of elements of type $\mathcal{C}$ in $\mathcal{A}$,
let $\{\mathcal{C}_1, \mathcal{C}_2\} \in \mathcal{K}$,
let $comp(\mathcal{C}_1, \mathcal{C}_2)$ be a function that verifies the semantic compatibility of $\mathcal{C}_1$ and $\mathcal{C}_2$ and that completes the semantic structure of $\mathcal{C}_2$ with that of $\mathcal{C}_1$[3]

- LP: $\mathcal{C}_1 \prec \mathcal{C}_2$ holds in $\mathcal{A}$ iff $pos(\mathcal{C}_1, \mathcal{A}) < pos(\mathcal{C}_2, \mathcal{A})$
- Req: $\mathcal{C}_1 \Rightarrow \mathcal{C}_2$ holds in $\mathcal{A}$ iff $\mathcal{C}_1 \notin \mathcal{A}$ or $\mathcal{C}_2 \in \mathcal{A}$
- Excl: $\mathcal{C}_1 \not\Rightarrow \mathcal{C}_2$ holds in $\mathcal{A}$ iff $\{\mathcal{C}_1, \mathcal{C}_2\} \cap \mathcal{A} \neq \{\mathcal{C}_1, \mathcal{C}_2\}$
- Uniq: $Uniq(\mathcal{C}_1)$ holds in $\mathcal{A}$ iff $card(\mathcal{C}_1, \mathcal{A}) \leq 1$
- Oblig: $Oblig(\mathcal{C}_1)$ holds in $\mathcal{A}$ iff $card(C_1, \mathcal{A}) = 1$
- Dep: $\mathcal{C}_1 \rightsquigarrow \mathcal{C}_2$ holds in $\mathcal{A}$ iff $comp(\mathcal{C}_1, \mathcal{C}_2)$ holds

A grammar is in this perspective a set of constraints, and nothing else. In particular, there is neither ranking in the OT sense nor need of building any structure before being able to evaluate the properties (as in OT with the Gen function or in HPSG with the need of selecting first a hierarchized structure type). Parsing in property grammars consists in evaluating for a given input the entire set of constraints. The characterization of an input is then formed by the set of satisfied constraints and the set of violated ones. More precisely, the grammar contains for each category a subset of constraints. It is then possible to specify for each category as well as for the entire input a characterization (the set of evaluated constraints).

[3] The semantic completion follows some schemas such as *subject, complement* or *modifier*. These schema indicate what part of the semantic structure of the modified category must be completed with that of the dependent.

2.2 Property Grammars and Constraint Solving Problem

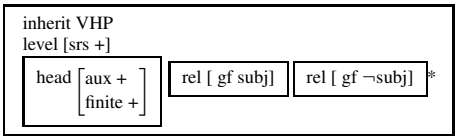
Unlike a more standard way of seeing a constraint solving problem which would consist in finding an assignment that satisfies the system of constraints, we start here with a partial assignment—corresponding to the utterance to be parsed—and we want to complete this assignment so as to satisfy the system of constraints—i.e. the grammar. In terms of output of the solving process we are not only interested in the qualitative aspect of the final assignment but equally in knowing which constraints are satisfied and which ones are not, both from a qualitative and a quantitative point of view. The knowledge of the assignment by itself is not enough. Actually, what we want to know is, given a system of constraints and a first partial assignment—representing the input to be analysed— a description of the final system and the complete assignment used to reach it. In a linguistic perspective it means that we want to know the lists of properties which are satisfied and violated by the input utterance—i.e. the characterisation—when the input is analysed in such way—i.e. for a given assignment.

2.3 Constructions and Constraints

A fully constraint-based view such as the one proposed by property grammars makes it possible to implement contextual fine-grained information. In the same way as categories are described by means of constraint subsets, other kind of objects such as constructions can also be specified in the same way. The notion of construction is of deep importance, especially in the perspective of finding gradient judgements: the importance of a constraint can vary from one construction to another. In the remainder of this section, we describe how constructions can be represented by means of constraints in the PG framework, taking the example of the subject-auxiliary inversion (SAI) construction in English (see [Fillmore98]). In this construction, there are strong linear constraints, especially concerning adverbs as shown in the following examples:

(4) Did you learn your lesson?
(5) Did you really learn your lesson?
(6) Didn't you learn your lesson?
(7) *Did really you learn your lesson?
(8) *Did not you learn your lesson?
(9) *Did you your lesson learn?

This construction is explained by the following SAI construction:



In terms of PG, this construction can be detailed as a set of such constraints:

1. V[aux] $\prec$ NP[subj]
2. NP[subj] $\prec$ V[$\neg$fin]
3. V[aux] $\Rightarrow$ NP[subj]
4. V[$\neg$fin] $\prec$ XP[$\neg$sub]
5. NP[subj] $\prec$ Adv[neg,$\neg$contraction]

6. NP[subj] $\prec$ Adv[$\neg$neg]
7. NP $\rightsquigarrow$ V
8. Adv $\rightsquigarrow$ V

This subset of constraints {1,2,4,7,8} represents the information of the SAI construction represented above and we can say that these two notations are equivalent. Adding new information simply consists in adding new constraints to the set describing the construction, at the same level. In this example, on top of these general constraints, it is necessary to specify some constraints. For example, the negative form has to be contracted here. This constraint is imperative in this construction, whereas it can be optional in other cases. This constitutes one important interest in the perspective of associating constraints with an "importance" degree: such degree may vary according to the construction. Using the terminology of [Sorace04], a constraint can be hard in some construction or soft in some others.

3 Gradience and Density

A fully constraint-based representation may also be helpful in identifying criteria for sentence complexity as well as acceptability. The idea is to make use of the information contained in characterizations in terms of satisfied and violated constraints. More precisely, some figures can be extracted from these characterizations illustrating the difference in the realization of a given category. For example, the ratio of satisfied/violated constraints is obviously of main interest.

(10) Quelles histoires Paul a-t-il écrites ?
What stories Paul did he write[fem-plu]? / What stories did Paul write?

(11) Quelles histoires Paul a-t-il écrit ?
What stories Paul did he write[masc-sing]? / What stories did Paul write?

(12) Quelles histoires a-t-il écrites Paul ?
What stories did he write[fem-plu] Paul? / What stories did he write Paul?

(13) Quelles histoires a-t-il Paul écrites
What stories did he Paul write[fem-plu]? / What stories did he Paul write?

These examples are given in order of (un)acceptability which corresponds in our hypothesis to a progressively greater number of violated constraints. Constraints are given here without taking into account specificities of the interrogative construction.

(11) NP[obj] $\rightsquigarrow$ VP[ppas]
(12) NP[subj] $\prec$ VP
(13) NP[subj] $\prec$ VP, V $\not\Leftrightarrow$ NP[subj]

Even without a precise evaluation of the consequence of constraint violations type by type, this first criterion can constitute an objective element of estimation for acceptability: unacceptability increases with the number of constraint violations (Keller's property of *Cumulativity*). This indication seems trivial, but directly comes from the possibility of representing separately the different types of syntactic information by means of properties. Such estimation is for example not possible with a phrase-structure representation and even difficult using classical constraint-based approaches such as HPSG.

However, it is necessary to have a finer-grained use of such information. In particular, the number of constraints may vary from one category to another. Some categories, such as adverbial phrases are very static and are described with a limited number of properties. At the opposite, the noun phrase, that can have many different forms, needs an important number of properties. It is then necessary to distinguish the number of constraint violation in these cases: violating a constraint for an AdvP has more consequences on acceptability than for the NP. Again, this indication is purely quantitative and doesn't take into account constraint type. It is probably the case that some constraints (for example exclusion) play a more important role on acceptability than dependency for example. However, when taking into consideration interpretability for example, a hard constraint such as unicity with respect to acceptability becomes soft for the interpretation, as shown in the following examples:

(14) Paul reads a book

(15) Paul reads reads a book

The second example is obviously unacceptable but perfectly understandable. We propose then a first stage in the identification of gradient criterion by means of purely quantitative aspects. This is the role played by the notion of *density*. This information indicates two figures: the number of satisfied properties with respect to the total number of properties that described the object and the same ratio for violated properties. We note respectively these figures as `dens_sat` and `dens_unsat` with the following definitions:

- `dens_sat` = nb of satisfied properties / total nb of properties
- `dens_unsat` = nb of unsatisfied properties / total nb of properties

To some extent the notion of density can be compared to the one of recall, which is used in evaluation.

Density in itself, at the difference with the ratio satisfied/violated properties, gives some indication about the quantity of information of a given object. In the case of a high density of satisfied properties, this means that an important number of syntactic characteristics contained in the grammar is realized in the object. In other words, we can say that this object contains, with respect to the grammar, important syntactic information. Reciprocally, a low density of satisfied properties can have different interpretations. In case of a high density of violated constraints, the object is clearly ill-formed and we can suspect a low probability for its acceptability. But it can also be the case that there is a low density for violated constraints. This situation indicated that the object contains little syntactic information. In the following example, extracted from a corpus analysis, the category, a sentence, is formed with a PP and a VP:

(16) En renforçant le projet, avançons vers le succès.

in reinforcing the project, let's go toward the success

Cat	`dens_sat`	`dens_unsat`	construction
S	0,5	0,125	PP; VP

Such a construction is not frequent, but some information can be given, according to the grammar. Concerning the violated properties, the non-null density comes from the fact that there is a dependency relation between the VP and a NP subject which is not realized. The level of satisfied properties density comes from the fact that even if the

properties involving PP and VP are satisfied, many properties describing S involve an NP. There is then a high number of properties for S that cannot be evaluated, explaining a low `dens_sat`.

These different ratios constitute then a first tool providing some indication on acceptability and interpretability. Acceptability primarily depends on the ratio of satisfied constraints with respect to the number of violated ones. Interpretability can be illustrated by the densities of satisfied and violated constraints. Low densities, as shown above, indicate a low level of syntactic information. More generally, there is a correlation between the quantity of information contained by an object and its interpretability: a high density of satisfied constraints comes with an easier interpretation. In case of low densities, it is necessary to obtain information from other domains such as prosody.

Using these different indications makes it possible to give information about any kind of input, without any restriction to well-formed ones. Moreover, it becomes possible to propose quantitative elements towards gradience in linguistic data concerning both acceptability and interpretability. Moreover, such elements of information give also some indication about domain interaction. For some utterances, it is necessary to extract information from the different linguistic domains such as morphology, syntax, prosody or pragmatics. In some other cases, the morpho-syntactic level alone contains enough information in order to make an utterance interpretable. In other words, there is a balance among the different domains. Each domain can be characterized with densities such as the one described here for syntax, the balance status being a function of the different densities. A high density of satisfied properties for one domain is an indication of a high level of information. The hypothesis stipulates that in this case, other domains can contain a low level of information without consequence on the interpretability. For example, for some construction, if there is a high density in syntax and semantics, then the density of prosody is not constrained and can take any value. Concretely, this means that intonation is not constrained anymore and can be realized in various ways. On the contrary, when syntactic and semantic densities are not heavy enough, then the prosody density has to be high and the intonation is less variable. This is the case in the example (3) for which prosody plays an important role in the interpretation.

4 Experiment

In the following, we give some indications from different French corpora, calculated from the output of a deterministic property grammar parser (see [Blache01] for a description of this parser). One important restriction is that, insofar as the parser used is deterministic, the number of violated constraints has been restricted to a minimal level. In particular, only the linearity, exclusion and unicity constraints have to be satisfied. The density of violated constraints is therefore not relevant for our discussion. We take then only into account the density of satisfied constraints. The aim of this experiment is to extract some figures from different data, for a given grammar and a given parser. It cannot be considered as a parser (or grammar) evaluation.

Three different corpora have been used: the first from the newspaper “Le Monde ’, with 15,420 words, the two others are transcribed spoken language corpora containing respectively 523 and 1,923 words. These corpora are very small, which is justified by

the difficulty in parsing such data. Moreover, they have been filtered: incomplete words for example have been eliminated. However, all repetitions are kept.

The first observation in this experiment is that, even if most of the categories have a null density (for the reasons explained above), there is a huge difference among the densities of satisfied constraints. The following table indicates for example some figures concerning the noun phrase in the written text corpus:

Density	*Const*	*Density*	*Const*
0.034483	PPro	0.310345	Det AP PP
0.068966	Clit	0.379310	Det N Rel
0.103448	N	0.413793	Det AP N
0.1724138	ProP AP	0.413793	Det N PP
0.206897	Det AP	0.517241	Det N Rel PP
0.241379	Det PP Rel	0.551724	Det N AP PP
0.275862	Det N	0.655172	Det N PP AP Rel

In these figures, one can remark that density doesn't grow systematically with grammaticality. For example, the two lowest densities correspond to grammatical constructions (personal pronoun and clitic). This comes from the fact that the noun phrase, which is the most frequent category, has many different constructions and needs a lot of constraints to describe them. In all cases, even when a given construction satisfied all its corresponding constraints, insofar as the total number of constraints for the NP is high, the density is necessarily low. Moreover, the realizations observed here only contain one category. The total number of satisfied properties is then by definition very low without having any consequence on the grammaticality (which should be indicated by the ratio satisfied/violated constraints). The same explanation is valid when comparing the realization /Det N/ with /Det N PP/. The first has a lower density whereas one should expect a high one for this basic construction. Frequency information plays then an important role in the use of the density notion. The following table indicates the mean density with respect to the frequency of the category in the different corpora:

Cat	*Frequency*	*Density*
S	0.069917582	0.4733535
AP	0.108379121	0.408556
AdvP	0.048139361	1
NP	0.302047952	0.204571
PP	0.1003996	0.31331
VP	0.218981019	0.341995
Circ	0.064360639	0.718518
Coord	0.071978022	0.4821425
Rel	0.015796703	0.3543475

We can see in this table that the most frequent categories (NP, VP and PP) are also those with the lowest mean density whereas the less frequent ones (Circ, AdvP and Coord) are associated with high densities. The arguments given above concerning the number of constituents, the number of properties and the number of different constructions can be used for explaining these differences.

This density parameter has then to be modulated with the frequency of the construction. In all cases, the progression of the density comes with an increasing quantity of

information. It is important to notice that the density notion is not directly useful in the identification of sentence complexity. For example, one can consider that a realization of a NP with a relative clause is more complex than a construction /Det N/. However, the first has a higher density than the second, for the reasons explained above. But from the interpretability point view, these aspects are disconnected. For example, a cleft construction, which is identified as being complex, is easily understandable because of the high number of constraints describing it. The following examples illustrate some density differences from a construction with few constraints to be satisfied and another containing more information:

	Example	*Density*
(16)	celui rouge *that red*	0,1724138
(17)	Le contenu de la future convention qui devrait permettre de régler les problèmes de fond *the content of the future convention that may allow to solve problems in depth*	0,6551724

5 Further Works

Intuitively, the notion of density could be refined by weighting the constraints according to their importance. The hard/soft discrimination ([Sorace04]), for instance, is not accounted at the moment by the density whereas we have seen previously that the constraints play roles of different importance when it comes to acceptability. Some sort of constraint ranking would also let us model the cumulativity and ganging up effects (i.e. when multiple violations of soft constraints could possibly be more unacceptable than a single violation of a hard constraint) described by [Sorace04].
Another object of further investigation concerns the use of weighted densities during the parsing process as an element of disambiguation. Indeed when faced with different possible assignments heuristics could be based on the measure of density for each possibility in order to rank the structures by preference. Subsequently a deterministic approach, of course, could also use this preferred structure to reduce the search space at different stages in the solving process.

6 Conclusion

The constraint-based approach proposed in this paper presents several advantages. In particular, all information is represented by means of constraints. Moreover, these constraints are at the same level (there is no ranking between them) and the grammar is formed by the constraint system. In contrast with other holistic approaches, each constraint can then be evaluated separately. Concretely, this means that syntactic information can be built for any input, whatever its form. Moreover, such information can be quantified and gives some indications about acceptability (high ratio satisfied/violated constraints) and interpretability (high density of satisfied constraints). One of the interests of this approach is that such information can be modulated. For example, it has

been shown that the importance of some constraints may vary from one construction (represented in our approach as a subset of constraints) to another.

From a cognitive point of view, the use of a fully constraint-based system may have some interest in terms of modeling. Constraints play the role of a filtering process: linguistic information does not consist in defining all and only the possible constructions, but in indicating for some construction what kind of information can be extracted. This means that in some cases (at least theoretically), little (or no) information can be extracted from one domain. But even in this case, such utterances can be treated.

Acknowledgements

We would like to acknowledge the support from an International Macquarie University Research Scholarship (iMURS) for JPP, from the CNRS and from a Macquarie University Research Development Grant (MURDG).

References

[Blache01] Blache P. & J-M. Balfourier (2001). "Property Grammars: a Flexible Constraint-Based Approach to Parsing", in proceedings of IWPT-2001.

[Blache00] Blache P. (2000). "Constraints, Linguistic Theories and Natural Language Processing", in *Natural Language Processing*, D. Christodoulakis (ed), Lecture Notes in Artificial Intelligence 1835, Springer-Verlag

[Croft03] Croft W. & D. Cruse (2003) *Cognitive Linguistics*, Cambridge University Press.

[Fillmore98] Fillmore C. (1998) "Inversion and Contructional Inheritance", in *Lexical and Constructional Aspects of Linguistic Explanation*, Stanford University.

[Gibson00] Gibson T. (2000) "Dependency locality theory: a distance-based theory of linguistic complexity", in Marantz & al. (eds), Image, Language and Brain , MIT Press.

[Goldberg95] Goldberg A. (1995) *Constructions: A Construction Grammar Approach to Argument Structure*, Chicago University Press.

[Kay99] Kay P. & C. Fillmore (1999) "Grammatical Constructions and Linguistic Generalizations: the *what's x doing y* construction", Language.

[Keller03] Keller F. (2003) "A probabilistic Parser as a Model of Global Processing Difficulty", in proceedings of *ACCSS-03*

[Langacker99] Langacker R. (1999), *Grammar and Conceptualization*, Walter de Gruyter.

[Mertens93] Mertens P. (1993) "Accentuation, intonation et morphosyntaxe", in Travaux de Linguistique 26

[Pollard94] Pollard C. & I. Sag (1994), *Head-driven Phrase Structure Grammars*, CSLI, Chicago University Press.

[Prince93] Prince A. & Smolensky P. (1993), Optimality Theory: Constraint Interaction in Generative Grammars, Technical Report RUCCS TR-2, Rutgers Center for Cognitive Science.

[Pullum03] Pullum G. & B. Scholz (2003), Model-Theoretic Syntax Foundations - Linguistic Aspects, ESSLLI lecture notes, Vienna University of Technology.

[Sag99] Sag I. & T. Wasow (1999), *Syntactic Theory. A Formal Introduction*, CSLI.

[Sorace04] Sorace A. & F. Keller (2004), *Gradience in Linguistic Data*, to appear, *Lingua*.

[Vasishth03] Vasishth S. (2003) "Quantifying Processing Difficulty in Human Sentence Parsing", in procedings of Eurocogsci-2003

Problems of Inducing Large Coverage Constraint-Based Dependency Grammar for Czech

Ondřej Bojar

Center for Computational Linguistics, MFF UK
Malostranské náměstí 25, CZ-118 00 Praha 1, Czech Republic
obo@cuni.cz

Abstract. This article describes an attempt to implement a constraint-based dependency grammar for Czech, a language with rich morphology and free word order, in the formalism Extensible Dependency Grammar (XDG). The grammar rules are automatically inferred from the Prague Dependency Treebank (PDT) and constrain dependency relations, modification frames and word order, including non-projectivity. Although these simple constraints are adequate from the linguistic point of view, their combination is still too weak and allows an exponential number of solutions for a sentence of n words.

1 Introduction

Czech is a thoroughly studied Slavonic language with extensive language data resources available. Traditionally, most of the research on Czech is performed within the framework of Functional Generative Description (FGD, [1]). This dependency-based formalism defines both surface syntactic (*analytic*) and deep syntactic (*tectogrammatical*, syntactico-semantic) level of language description. Language data sources for Czech include Prague Dependency Treebank (PDT, [2, 3]) and Czech valency lexicon (VALLEX, [4]). Specific properties of Slavonic languages and Czech in particular make the task of syntactic analysis significantly more difficult than parsing English. Available parsers of Czech ([5], [6] and an adapted version of [7]) are statistical, aimed at surface syntactic analysis and there is no simple way to extending them to include deep syntactic analysis. Up to now, no attempt has been made to approach large-coverage syntactic analysis with a constraint-based technology.

Extensible Dependency Grammar (XDG, [8]) is a promising relational framework aimed at multi-dimensional constraint-based analysis of languages. So far, only small scale grammars have been implemented in XDG. These grammars illustrated efficient and elegant treatment of various complex syntax and semantic phenomena in XDG [9, 10]. However, the grammars were always tailored to a few test sentences and constraints implemented in XDG never had to cope with syntactic ambiguity of a grammar inferred from a larger amount of data.

This paper describes a first experiment of inducing a large-coverage XDG grammar for Czech from PDT.[1]

[1] A more detailed description is given in [11].

1.1 Properties of Czech Language

Table 1 summarises some of the well known properties of Czech language[2]. Czech is an inflective language with rich morphology and relatively free word order allowing non-projective constructions. However, there are important word order phenomena restricting the freedom. One of the most prominent examples are clitics, i.e. pronouns and particles that occupy a very specific position within the whole clause. The position of clitics is very rigid and global within the sentence. Locally rigid is the structure of (non-recursive) prepositional phrases or coordination. Other elements, such as the predicate, subject, objects or other modifications may be nearly arbitrarily permuted. Such permutations correspond to the topic-focus articulation of the sentence. Formally, the topic-focus articulation is described at the deep syntactic level.

Moreover, like other languages with relatively free word order, Czech allows non-projective constructions (crossing dependencies). Only about 2% of edges in PDT are non-projective, but this is enough to make nearly a quarter (23.3%) of all the sentences non-projective.

The task of parsing languages with relatively free word order is much more difficult than parsing of English, for example, and new approaches still have to be searched for. Rich morphology is a factor that makes parsing more time and data demanding.

	Czech	English
Morphology	rich	limited
	$\geq$ 4,000 tags	50 used
	$\geq$ 1,400 actually seen	
Word order	free with rigid global phenomena	rigid
Known parsing results		
Edge accuracy	69.2–82.5%	91%
Sentence correctness	15.0–30.9%	43%

Table 1. Properties of Czech compared to English.

1.2 Overview of the Intended Multi-dimensional Czech Dependency Grammar

Figure 1 summarises data sources available for a Czech grammar induction. PDT contains surface syntactic (analytic, AT) as well as deep syntactic (tectogrammatical, TG) sentence annotations. The Czech valency lexicon is under

[2] Data by [5], [12], Zeman (http://ckl.mff.cuni.cz/~zeman/projekty/neproj), [13] and [14]. Consult [15] for measuring word order freeness.

development, and alternatively, the valency lexicon collected while annotating the tectogrammatical level of PDT could be used.

A grammar in the formalism of XDG could be inferred from these sources addressing the immediate dominance (ID), linear precedence (LP) and predicate-argument (PA) dimensions.

Only a part of this overall picture has been implemented so far. First, the correspondence between tectogrammatical and analytic levels is quite complicated, some nodes have to be deleted, some nodes have to be added. Second, the tectogrammatical valency information from Vallex is mostly useful only if a tectogrammatical structure is considered, only then the constraints addressing surface realization can be fully exploited. Therefore, in the first approach the current grammar implementation focuses only on ID an LP levels.

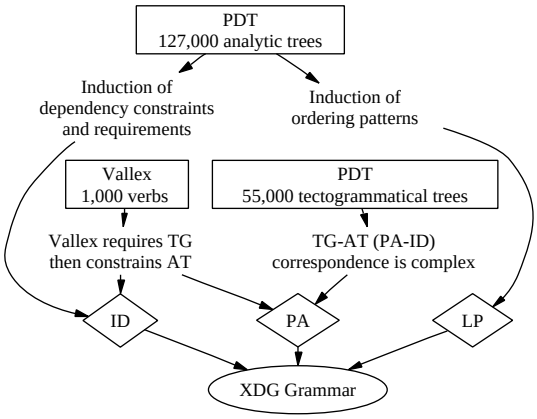


Fig. 1. Czech data sources available for XDG grammar.

2 Description of the Grammar Parts

The experimental XDG grammar induced from PDT utilizes basic principles that are linguistically motivated and traditionally used in many varieties of dependency grammars, including XDG. The current XDG grammar extracted from PDT consists of the following parts: ID Agreement, LP Direction, Simplified ID Frames and ID Look Right. For every part independently, the properties of individual lexical entries (with an arbitrary level of lexicalization) are collected from the training data. The contributions are then combined into XDG lexical entries and classes in a conjunction manner: when parsing, every input word must match one of the observed configurations in all the grammar parts.

For practical reasons (memory and time requirements), the grammar finally used in the XDG parser is restricted to the word forms of the test sentences only. Figure 2 summarizes the pipeline of grammar extraction and evaluation.

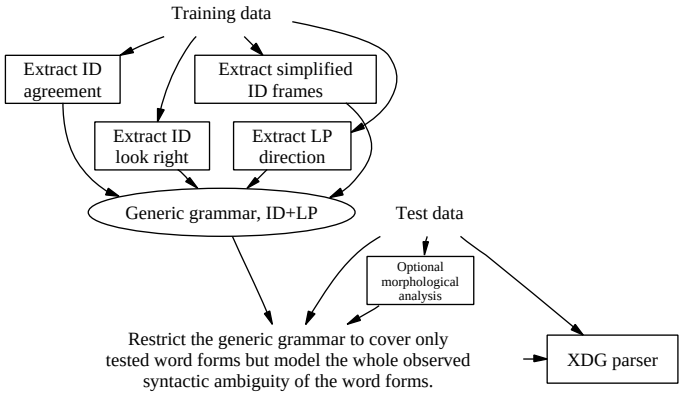


Fig. 2. XDG grammar parts and evaluation.

2.1 Grammar Non-lexicalized in General

XDG is designed as a lexicalized formalism, most syntactic information is expected to come from the lexicon. Conversely, to make the most use of this approach, the information in an XDG grammar should be as lexicalized as possible.

Despite the size of PDT (1.5 million tokens), there is not enough data to collect syntactic information for individual word forms and even lemmas.

All the grammar parts described below are therefore based on simplified morphological tags only (part and subpart of speech, case, number and gender). Table 2 justifies this simplification. Theoretically, full morphological tags could be used, but we would face sparse data problem if pairs (such as head-dependent pairs) or n-tuples of tags were examined.

After having observed	20,000	75,000 sentences
a new ... comes every		test sent.
lemma (i.e. word)	1.6	1.8 test sent.
full morphological	110	290 test sent.
simplified tag	280	870 test sent.

Table 2. Lack of training data in PDT for full lexicalization.

2.2 ID Agreement

The ID Agreement part of the grammar allows for a specific edge type between a father and a daughter. The edge type is cross checked in both directions: from the father and from the daughter.

Technically, the lexical entry of a father (with known morphological properties) contains a mapping from edge labels to morphological requirements on any possible daughter. If a daughter is connected via a particular edge label to this father, the daughter's morphology must match at least one of the requirements. Conversely, the daughter's lexical entry contains a mapping to restrict the morphology of the father.

This approach ensures grammatical agreement between the father and the daughter and also helps to reduce morphological ambiguity of nodes: For every node, only such morphological analyses remain allowed which fit the intersection of requirements of all the connected nodes. During parsing, the ambiguous morphology of the node is reduced step by step, as more and more edges are assigned.

2.3 LP Direction

The LP Edge Direction part describes simplified linear precedence rules and handles non-projectivity. In the original design of XDG grammars, motivated by German, the LP dimension is used to describe *topological fields* [16]. Unfortunately, the word order of Czech and other Slavonic languages does not exhibit similar word order restrictions in general. (To a very limited extent, one could think about three fields in a clause: preclitic, clitic and postclitic field.) However, there is often an important distinction between dependencies to the left and dependencies to the right. In this first attempt, the LP constraints of the grammar ensure only an acceptable direction (left/right) of an edge between a father and a daughter. The constraints do not model acceptability of different mutual orderings of several daughters of a father.

Technically, the checking of edge direction is implemented by means of topological fields, but these are extremely simplified. Every father at the LP dimension offers three fields: the left and right fields of unlimited cardinality[3] and the head field to contain only the father itself. The left field is offered for all the daughters to the left, the head field is used for the father itself and the right field is offered for all the daughters to the right. There is no restriction on mutual ordering of the left or right daughters, the only ensured thing is that every left daughter must precede the father and every right daughter must follow the father.

The LP edge direction is coupled with the ID label of the corresponding ID edge. Given a daughter connected to the father with an ID edge of a particular label, the corresponding LP edge is in certain cases allowed to have only the label

[3] In other words, unlimited number of outgoing LP edges can have the label LEFT and all edges labelled LEFT must be present first in the left-to-right ordering of nodes.

LEFT, in other cases only the label RIGHT but sometimes both of the labels (i.e. both directions) are allowed. As illustrated in Figure 3, under the preposition *about*, an (ID) edge labelled ATR can go to the right only, so the corresponding LP edge must have the label RIGHT. On the other hand, adverbs can be found both before and after the governing verb and therefore the verb *was* accepts outgoing (ID) edges labelled ADV both in the left and right fields.

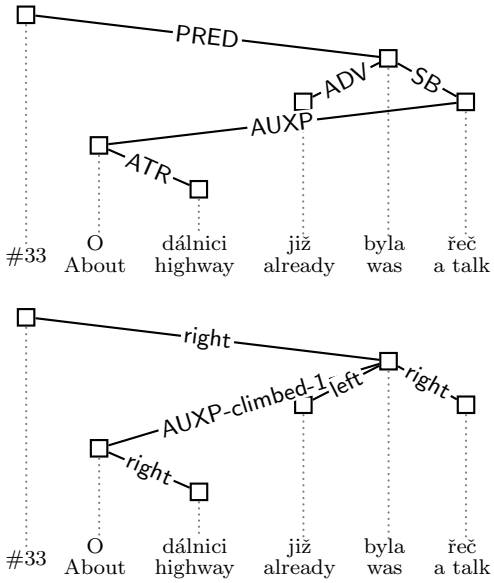


Fig. 3. LP dimension to handle edge direction and non-projectivity.

An intuitive approach to handle non-projectivities in Czech is to require projective analyses in general but allow for non-projective edges in specific observed cases.[4] My XDG grammar expresses this requirement in the LP tree only, the ID tree is allowed to be non-projective in general. The LP tree is required to be projective and the exceptions are handled by the so-called *climbing principle*. In order to obtain a projective LP tree from a non-projective one, the tree is "flattened" by climbing. For example, the AUXP edge is non-projective in the ID tree in Figure 3. Moving the corresponding LP edge one step up from the governor *talk* to the governor *was*, the LP edge becomes projective.

To distinguish LP edges that had to climb from LP edges directly corresponding to ID edges, a set of extra LP labels is introduced: AUXP-CLIMBED-1, ATR-CLIMBED-1... These additional LP labels encode also the ID label, because the

[4] Consult [17] for a more advanced approach to restricting non-projectivity.

syntactic role of the daughter is important with respect to allowing or denying the non-projective realization.

The nodes where a climbed edge may land (such as the word *was* in Figure 3) offer not just the left, head and right fields, but also the required amount of specific X-CLIMBED-Y edges. There is no restriction on mutual linear ordering of the LEFT/RIGHT and *-CLIMBED-* edges.[5]

This way, sentences are analyzed projectively in general, but specific known non-projectivities (based on the simplified morphological tag of the father and the daughter and the ID label of the non-projective edge) are modelled, too.

The current model still lacks restrictive power to control the clitic position. Similarly, coordination is not modelled properly yet, because the cardinality of left and right fields is unrestricted in general (for example, both members of a coordination are allowed to appear on the same side of the conjunction). More adequate handling of these phenomena remains open for further research.

2.4 Simplified ID Frames

One of the crucial principles restricting available sentence analyses in XDG is the valency principle: Every father node allows only specific combinations and cardinalities of outgoing (ID) edges.

The Simplified ID Valency Frames ensure that a word doesn't accept implausible combinations of modifiers. Rarely, they ensure that a word has all its "modification requirements" saturated, because most of the modifiers are deletable anyway.

Current approaches[6] aim at distinguishing *complements* vs. *adjuncts*, i.e. modifications that are typically required vs. optional. However, there is no use of this distinction, if *deletability* of modifications is taken into account (in real Czech sentences, complements are often omitted). Any consistent grammar must reflect this optionality of complements.

The restrictive power of valency frames in XDG should therefore come from *interdependencies* of modifications (e.g. if a secondary object or a specific type of adjunct was observed, a primary object must be present). The set of allowed combinations and cardinalities must be explicitly enumerated in the current XDG implementation. Future versions of this principle might accept a constraint network (for example a set of implications) of interdependencies.

To my knowledge, no published approach aims at discovering such interdependencies of particular modifications so far. On the other hand, there are too

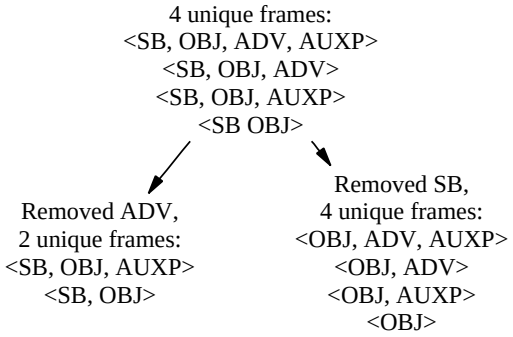
[5] This is due to technical limitations of the XDG parser: currently it is not possible to impose partial ordering on topological fields, only linear ordering is supported. Either the *-CLIMBED-* fields are not mentioned in the ordering at all, or one must specify a full linear ordering among them. It is not possible to express only that all *-CLIMBED-* fields precede the HEAD field without specifying the ordering among them.

[6] See [18] for comparison and references.

many unique frames observed under a given node type, so it is impossible to enumerate all of them.[7]

Therefore, I implemented a naive algorithm to infer simplified modification frames: this algorithm automatically simplifies treatment of adjuncts and stores the complexity of interdependencies of other modifications by enumerating them. As sketched in Figure 4, the set of observed modification frames of a specific word class can be simplified by removing different modification types. When an adverbial is removed under a verb, the set of modification frames shrinks to a half in size. When the subject is removed instead, the set does not shrink at all. This indicates that an adverbial has no effect on interdependencies of other modifications: an adverbial may be present or may not–half of the frames was observed with an adverbial, half of the frames had no adverbial.

Example: Observed under a verb:



⇒ ADV is more optional than SB.

Fig. 4. Identifying optional modifications in order to simplify the set of allowed modification frames.

This simplification is applied iteratively, until the number of unique frames is acceptable. The removed modifications are added to all the frames as optional.

A short example in Figure 5 illustrates the optionality order of modifications observed under infinite verbs (POS=V, SUBPOS=f). In a sample of 2,500 sentences, there were 727 occurrences of infinite verbs. Regardless the mutual order of modifications of the verbs but with respect to the number of modifications of a particular kind (i.e. representing the modification frame as a multiset, a bag

[7] Enumerating all seen modification frames would face a severe sparse data problem anyway as the number of unique modification frames steadily grows. In 81,000 sentences, there were 89,000 unique frames observed when describing the frames as lists of simplified tags of all the daughters of a node.

of modification labels), there were 132 unique frames observed. The order of optionality of different modification types is estimated by the described algorithm. The most optional modification (AUXP[8], a prepositional phrase) is torn off in the first step, reducing the set size from 132 to 95. Equally optional is an adverbial (ADV) because tearing off the adverbial alone would lead to the set size of 95, too. In the cumulative process, the set size after removing AUXP and ADV is 64. The third most optional modification is an object (OBJ) so we arrive at 46 core frames plus three modifications marked as optional. The resulting frames are shown in Figure 5, too. Each resulting frame contains some fixed members, strictly required at least once, and also the optional modifications with cardinality 0 to the highest observed cardinality of this particular modification. For instance, the first resulting frame {AUXP(0-3), ADV(0-3), OBJ(0-2)} contains no fixed members at all, AUXP and ADV are allowed at most 3 times and OBJ is allowed at most twice. Finally, the XDG grammar requires every finite verb in a sentence to satisfy at least one of the allowed modification frames, i.e. to have exactly the allowed number of outgoing ID edges of a particular kind, as the frame prescribes.

Unique observed modification frames: 132
Set sizes when removing specific modifiers:
AUXP(95), ADV(95), OBJ(96), AUXC(113), AUXV(119), AUXT(119), AUXX(122), COORD(123), SB(126), AUXZ(126), AUXR(126), ATVV(127), PNOM(128), AUXG(128), AUXY(129), APOS(129), EXD(130), COORD_PA(131), ATR(131), PRED_PA(132), EXD_PA(132)
Cumulative simplification:
132→(AUXP)→95→(ADV)→64→(OBJ)→46.
Resulting frames:
{AUXP(0-3), ADV(0-3), OBJ(0-2)}
{APOS(1), EXD(1), AUXP(0-3), ADV(0-3), OBJ(0-2)}
{APOS(1), AUXP(0-3), ADV(0-3), OBJ(0-2)}
{ATR(2), COORD(1), AUXP(0-3), ADV(0-3), OBJ(0-2)}
{ATVV(1), AUXC(1), AUXP(0-3), ADV(0-3), OBJ(0-2)}
{ATVV(1), AUXT(1), AUXP(0-3), ADV(0-3), OBJ(0-2)}
(... 46 resulting frames altogether)

Fig. 5. Simplifying modifications of infinite verbs.

It should be noted that the described solution is by no means a final one. The tasks of inducing modification frames and employing the frames to constrain syntactic analysis are very complex and deserve much deeper research.

[8] See [2] for explanation of the labels.

2.5 ID Look Right

The generally accepted idea of dependency analysis is that head-daughter dependencies model syntactic analysis best. [19] doubt this assumption and document that for German sister-sister dependencies (lexicalized case) are more informative.

Context		Neighbours		Sisters	
used	Head	Left	Right	Left	Right
Entropy	0.65	1.20	1.08	1.14	1.15

Table 3. Difficulty of predicting an edge label based on simplified tag of a node and a node from close context.

Table 3 gives an indication for Czech: if the structure was already assigned, choosing the edge label is easiest when looking at morphological properties of the node and its head (lowest entropy). Contrary to Dubey and Keller, Czech with a very strong tendency for grammatical agreement confirms the generally accepted view.

The ID Agreement principle is crucial in Czech and it is already employed in the grammar. Table 3 indicates also which context gives the second best hint: the right neighbour, i.e. the following word. Therefore, a new principle was added: ID Look Right: An incoming ID edge to a word must be allowed by the word class of its right neighbour.

The differences among sisters' and neighbours' contributions to the prediction of edge label are not very significant, so adding more constraints of this kind is still under consideration.

3 Results

To evaluate the grammar, only the first fixed point in constraint solving is searched. Given a sentence, the XDG parser propagates all relevant and applicable constraints to reduce the number of analyses and returns an underspecified solution: some nodes may have unambiguously found a governor, for some nodes, several structural assignments may still remain applicable. At the first fixed point, none of the constraints can be used to tell anything more[9].

[9] At fixed points, also called choice points, the constraint solver of the underlying system Mozart-Oz makes an arbitrary decision for one of the still underspecified variables and starts propagating constraints again. Other fixed points are reached and eventually a fully specified solution can be printed. Different solutions are obtained by making different decisions at the fixed points. The parser can be instructed to perform a complete search, but in our case there is no point in enumerating so many available solutions.

Two grammars were evaluated: first a version without the Look Right principle, second a version that included the new principle, too. The grammars were trained on sentences from the training part of PDT and evaluated on 1,800 to 2,000 unseen sentences from the standard evaluation part of PDT (devtest). The results are displayed in Table 4.

Note that the number of training sentences was relatively low (around 2 to 5% of PDT), which explains the relatively high number of unsolved sentences (around 10 to 20%). A wider coverage of the grammar is achieved by training on more data but immediately leads to significant growth of the number of solutions available. This problem with scalability can be solved only by providing the grammar with more constraints of various kinds. As indicated in the row Avg. ambiguity/node, a node has 8 to 9 possible governors (regardless the edge label). Compared with the average sentence length of 17.3 words, the grammar reduces the theoretically possible number of structural configurations to a half. At the first fixed point, the parser has enough information to establish only 3 to 5% of edges, an edge with a label can be assigned only to 2 to 4% of nodes. Out of the assigned structural edges, around 82% is correct, out of the assigned labelled edges, around 85% is correct. Based on other experiments, training on more data leads to a lower error rate but less edges securely established.

Contrary to our expectations, adding the new principle Look Right did not help the analysis. The average ambiguity per node became even higher. There were slightly more edges securely assigned, but the correctness of this assignment has dropped. One possible explanation comes from the word order freedom of Czech. The Look Right principle probably helps to establish rigid structures such as dates but leads to wrong decisions in general, because it faces a serious sparse data problem. A deeper analysis is necessary to confirm this explanation.

4 Discussion and Further Research

The presented results indicate several weak points in the described approach to constraint-based dependency parsing. All these points remain open for further research.

First, the current grammar relies on very few types of constraints. More constraints of different kinds have to be added to achieve both a better scalability of the grammar and a more effective propagation of the constraints.[10] The current grammar lacks especially such a kind of constraints that bind information together–the current constraints are too independent to achieve strong propagation. A related problem is the locality of the constraints. All the current constraints rely on a too local context. There are too many analyses available, because the local constraints are not powerful enough to check invariant properties of clauses or sentences as a whole.

[10] Similarly as [20] observed for English, purely syntactic constraints are too weak to analyse Czech. The deep syntactic level of PDT and the Czech valency lexicon provide a promising source of additional constraints.

Training sentences	2500	5000
	Unsolved sentences	
Without Look Right	21.1	11.9
With Look Right	25.6	15.4
	Avg. ambiguity/node	
Without Look Right	8.09	8.91
With Look Right	8.17	9.05
	Assigned structural edges	
Without Look Right	4.4	3.3
With Look Right	4.7	3.5
	Correct structural edges	
Without Look Right	82.3	82.5
With Look Right	81.9	81.0
	Assigned labelled edges	
Without Look Right	3.4	2.3
With Look Right	3.6	2.5
	Correctly labelled edges	
Without Look Right	85.9	85.9
With Look Right	85.0	83.5

Table 4. Results of underspecified solutions.

Second, there are several kinds of expressions that in fact have no dependency structure, such as names, dates and other multi-word expressions. Coordination should be handled specifically, too. The "dependency" analysis of such expressions in PDT reflects more the annotation guidelines than some linguistic motivation. Separate treatment of these expressions by means of a sub-grammar would definitely improve the overall accuracy. This expectation comes from my analysis of sources of structural ambiguity modelled by the grammar: given the set of all trees assigned by the grammar to a string of words, punctuation symbols, cardinals, adverbs and conjunctions (in this order) are the parts of speech that have most different governors.

Third, the tested version of XDG parser could not make any use of frequency information contained in PDT.[11] [21] attempt at guiding the XDG parser by frequency information which should help to find a plausible solution sooner, but the research is still in progress.[12]

[11] In an experiment, frequency information was used as a threshold to ignore rare edge assignments. The thresholding resulted in lower coverage *and* lower precision.

[12] A similar constraint-based dependency parsing by [22] inherently includes weight of constraints, but no directly comparable results were published so far. [23] report edge accuracy of 96.63% on a corpus of 200 sentences with average length 8.8 words, significantly less than in our data.

5 Conclusion

I described an experiment with constraint based dependency parsing of a language with rich morphology and relatively free word order. Although the constraints are linguistically adequate and serve well when employed on small-scale corpora, they face a serious problem when trained on large data sets. The constraints are too local and weak in order to restrict the number of available solutions.

6 Acknowledgement

I'm grateful to Ralph Debusmann for his explanatory and immediate implementation support of new features needed in the XDG parsing system for this experiment. The work could not have been performed without the support of Programming Systems Lab headed by Gert Smolka (Universität des Saarlandes) and without the insightful guidance by Geert-Jan Kruijff and Denys Duchier. This work has been partially supported by the Ministry of Education of the Czech Republic, project LN00A063.

References

1. Sgall, P., Hajičová, E., Panevová, J.: The Meaning of the Sentence and Its Semantic and Pragmatic Aspects. Academia/Reidel Publishing Company, Prague, Czech Republic/Dordrecht, Netherlands (1986)
2. Hajič, J., Panevová, J., Buráňová, E., Urešová, Z., Bémová, A.: A Manual for Analytic Layer Tagging of the Prague Dependency Treebank. Technical Report TR-2001-, ÚFAL MFF UK, Prague, Czech Republic (2001) English translation of the original Czech version.
3. Hajičová, E., Panevová, J., Sgall, P.: A Manual for Tectogrammatic Tagging of the Prague Dependency Treebank. Technical Report TR-2000-09, ÚFAL MFF UK, Prague, Czech Republic (2000) In Czech.
4. Žabokrtský, Z., Benešová, V., Lopatková, M., Skwarská, K.: Tektogramaticky anotovaný valenční slovník českých sloves. Technical Report TR-2002-15, ÚFAL/CKL, Prague, Czech Republic (2002)
5. Collins, M., Hajič, J., Brill, E., Ramshaw, L., Tillmann, C.: A Statistical Parser of Czech. In: Proceedings of 37th ACL Conference, University of Maryland, College Park, USA (1999) 505–512
6. Zeman, D.: Can Subcategorization Help a Statistical Parser? In: Proceedings of the 19th International Conference on Computational Linguistics (Coling 2002), Taibei, Tchaj-wan, Zhongyang Yanjiuyuan (Academia Sinica) (2002)
7. Charniak, E.: A maximum-entropy-inspired parser. In: Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2000), Seattle, Washington, USA (2000) 132–139 `http://www.cs.brown.edu/people/ec/papers/`.
8. Debusmann, R., Duchier, D., Koller, A., Kuhlmann, M., Smolka, G., Thater, S.: A relational syntax-semantics interface based on dependency grammar. In: Proceedings of COLING 2004, Geneva, Switzerland (2004)

9. Duchier, D., Debusmann, R.: Topological dependency trees: A constraint-based account of linear precedence. In: 39th Annual Meeting of the Association for Computational Linguistics (ACL 2001). (2001)
10. Debusmann, R., Duchier, D.: A meta-grammatical framework for dependency grammar (2003)
11. Bojar, O.: Czech Syntactic Analysis Constraint-Based, XDG: One Possible Start. Prague Bulletin of Mathematical Linguistics (2004)
12. Holan, T.: K syntaktické analýze českých(!) vět. In: MIS 2003, MATFYZPRESS (2003)
13. Yamada, H., Matsumoto, Y.: Statistical dependency analysis with support vector machines. In: Proceedings of the International Workshop on Parsing Technologies (IWPT 2003), Nancy, France (2003)
14. Bojar, O.: Towards Automatic Extraction of Verb Frames. Prague Bulletin of Mathematical Linguistics (2003) 101–120
15. Kruijff, G.J.M.: 3-phase grammar learning. In: Proceedings of the Workshop on Ideas and Strategies for Multilingual Grammar Development. (2003)
16. Bech, G.: Studien über das deutsche Verbum infinitum. (1955) 2nd unrevised edition published 1983 by Max Niemeyer Verlag, Tübingen (Linguistische Arbeiten 139).
17. Holan, T., Kuboň, V., Oliva, K., Plátek, M.: Two Useful Measures of Word Order Complexity. In Polguere, A., Kahane, S., eds.: Proceedings of the Coling '98 Workshop: Processing of Dependency-Based Grammars, Montreal, University of Montreal (1998)
18. Sarkar, A., Zeman, D.: Automatic Extraction of Subcategorization Frames for Czech. In: Proceedings of the 18th International Conference on Computational Linguistics (Coling 2000), Saarbrücken, Germany, Universität des Saarlandes (2000)
19. Dubey, A., Keller, F.: Probabilistic parsing for German using sister-head dependencies. In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, Sapporo (2003) 96–103
20. Harper, M.P., Helzerman, R.A., Zoltowski, C.B., Yeo, B.L., Chan, Y., Stewart, T., Pellom, B.L.: Implementation Issues in the Development of the PARSEC Parser. SOFTWARE - Practice and Experience **25** (1995) 831–862
21. Dienes, P., Koller, A., Kuhlmann, M.: Statistical a-star dependency parsing. In Duchier, D., ed.: Prospects and Advances of the Syntax/Semantics Interface, Nancy (2003) 85–89
22. Heinecke, J., Kunze, J., Menzel, W., Schöder, I.: Eliminative parsing with graded constraints. In: Proceedings of COLING-ACL Conference, Montreal, Canada (1998)
23. Foth, K., Menzel, W., Schröder, I.: Robust parsing with weighted constraints. Natural Language Engineering (2004) in press.

Metagrammar Redux

Benoit Crabbé and Denys Duchier

LORIA, Nancy, France

Abstract. In this paper we introduce a general framework for describing the lexicon of a lexicalised grammar by means of elementary descriptive fragments. The system described hereafter consists of two main components: a control device aimed at controlling how fragments are to be combined together in order to describe meaningful lexical descriptions and a composition system aimed at resolving how elementary descriptions are to be combined.

1 Introduction

This paper is concerned with the design of large scaled grammars for natural language. It presents an alternative language of grammatical representation to the classical languages used for this purpose such as PATR II.

The need for a new language is motivated by the development of strongly lexicalised grammars based on tree structures rather than feature structures, and by the observation that, for tree based formalisms, lexical management with lexical rules raises non trivial practical issues [1].

In this paper we revisit a framework – the metagrammar – designed in particular for the lexical representation of tree based syntactic systems. It is articulated around two central ideas: (1) a *core* grammar is described by elementary tree fragments and (2) these fragments are combined by means of a control language to produce an *expanded* grammar. Throughout the paper, we illustrate the features of the framework using Tree Adjoining Grammar (TAG)[2] as a target formalism.

The paper is structured as follows. First (Section 2) we introduce the key ideas underlying grammatical representation taking PATR II as an illustration. We then provide the motivations underlying the design of our grammatical representation framework. The core metagrammatical intuition: lexical representation by manipulating fragments made of tree descriptions is provided in (Section 3). The motivations concerning the set up of an appropriate tree representation language are provided in Section 4. The fragment manipulation language is then developped in section 5. Section 6 introduces further questions concerning global conditions on model admissibility. And finally, the computational treatment of our description language is detailed in Section 7.

2 Lexical organisation

In this section we introduce the issue and the main ideas concerning lexical organisation of tree based syntactic systems. We begin by investigating the core

ideas developped in PATR II then we highlight inadequacies of PATR II for representing the lexicon of tree based syntactic systems such as Tree Adjoining Grammar.

An historical overview: PATR II Since the very first works in computational linguistics [3], lexical description roughly consists of specifying lexical entries together with a subcategorisation frame such as in PATR II:

```
love :
   <cat> = v
   <arg0 cat> = np
   <arg1 cat> = np
```

where we specify that the verb *love* takes two arguments: a *subject noun phrase* and an *object noun phrase*. This lexical entry, together with an appropriate grammar, is used to constrain the set of sentences in which *love* may be inserted. For instance this lexical entry is meant to express that *love* is used transitively as in *John loves mary* but not intransitively such as in *John loves* or *John loves to Mary*.

PATR II offers two devices to facilitate lexical description: templates and lexical rules. Templates are described by [3] as macros, and permit to easily state that that *love* and *write* are transitive verbs by writing:

```
love :
     transitiveVerb
write :
     transitiveVerb
transitiveVerb :
   <cat> = v
   <arg0 cat> = np
   <arg1 cat> = np
```

where `transitiveVerb` is a macro called in the descriptions of *love* and *write*. On the other hand, lexical rules are used to describe multiple variants of verbs. For instance, to express that a transitive verb such as love may be used in its active or passive variant we may add the following lexical rule to our lexicon:

```
passive :
   <out cat> = <in cat>
   <out arg1 cat> = <in arg0 cat>
   <out arg0 cat> = pp
```

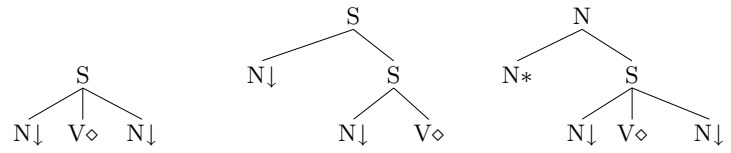
This rule says that a new lexical entry `out` is to be build from an initial lexical entry `in` where the category of `out` is identical to the category of `in`, the category of the object becomes the category of the subject and that the subject category now becomes prepositional phrase.

Lexical rules are meant to allow a dynamic expansion of related lexical variants. So for the verb *love* the application of the `passive` lexical rule to its base entry generates a new, derived, passive lexical entry meaning that both active and passive variants are licensed by the lexical entries.

Variants and improvements of this classical system have been (and are still) used for describing the lexicon in various syntactic frameworks such as LFG[4] or HPSG [5]. Whatever the differences, two leading ideas remain nowadays: lexical description aims both at factorising information (templates) and at expressing relationships between variants of a same lexical unit (lexical rules).

Tree Adjoining Grammar: a case study Tree adjoining grammar (TAG)[1] is a tree composition system aimed at describing natural language syntax [2] which strongly lexicalised. In other words, a tree adjoining grammar consists of a lexicon, the elementary trees, each of them being associated to a lexical unit, and two operations used for combining the lexical units: adjunction and substitution.

Following the conventions used in TAG implementations such as XTAG [6], we work with tree schematas (or templates) such as these[2]:



Jean voit Marie	Quelle fille Jean voit	(Jean) qui voit Marie
John sees Mary	Which girl John sees	(John) who sees Mary

(1)

where the appropriate lexical word is inserted dynamically by the parser as a child of the anchor (marked $\diamond$). The nodes depicted with an arrow ($\downarrow$) are the substitution nodes and those depicted with a star ($\star$) are the foot nodes.

Strikingly, works concerning lexical organisation of strongly lexicalised syntactic systems often try to provide alternative solutions to that of Shieber. The main reason is that the amount and the variety of lexical units is much more important, therefore the number of templates and lexical rules to be used is strongly increased. In the context of the development of large sized grammars, this situation requires the grammar writer to design complicated ordering schemes as it is illustrated by [1].

To overcome this, we take up an idea first introduced in [8] for Construction Grammar. Roughly speaking they describe the lexicon using a dynamic process: given a *core* lexicon manually described they build up an *expanded* lexicon by combining elementary *fragments* of information.

Besides strong lexicalisation, setting up a system representing a TAG lexicon raises another problem, that of the structures used. In Construction Grammar, [8] combine elementary fragments of information via feature structure unification. When working with TAG, however, one works with trees

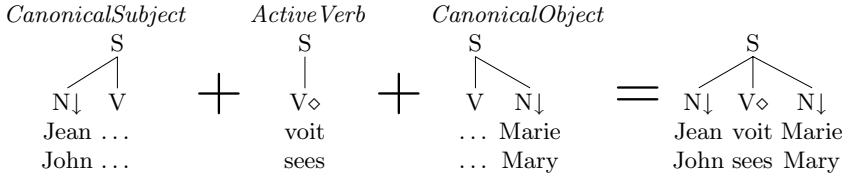
[1] Strictly speaking, we mean here Lexicalised Tree Adjoining Grammar (LTAG). Indeed, the system is usually used in its lexicalised version[6].

[2] The trees depicted in this paper are motivated by the French grammar of [7] who provides linguistic justifications in particular for the non utilisation of the VP category and the use of N at multiple bar levels instead of introducing the category NP in French.

3 Introduction to the framework

In this section we sketch the idea of describing the lexicon by controlling combinations of elementary fragment descriptions.

This idea stems from the following observation: the design of a TAG grammar consists of describing trees made of elementary pieces of information (hereafter: fragments). For instance the following tree is defined by combining a subtree representing a subject another subtree representing an object and finally a subtree representing the spine of the verbal tree:



Of course, we will also want convenient means of expressing variants of the above tree; for example, where, the subject instead of being realized in canonical position is realized as a questioned subject (*wh*) or a relative subject.

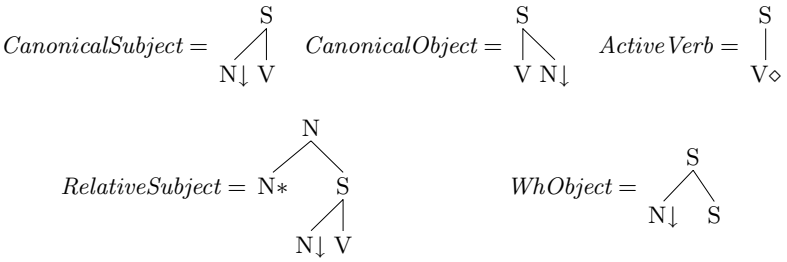
More generally while designing a grammar one wants to express general statements for describing sets of trees: for instance, a transitive verb is made of a subject, an object and a verbal active spine. In short we would like to write something like:

$$\mathit{TransitiveVerb} \quad = \quad \mathit{Subject} \ \wedge \ \mathit{ActiveVerb} \ \wedge \ \mathit{Object}$$

where *Subject* and *Object* are shortcuts for describing sets of variants:

$$\begin{aligned} \mathit{Subject} &= \mathit{CanonicalSubject} \ \vee \ \mathit{RelativeSubject} \\ \mathit{Object} &= \mathit{CanonicalObject} \ \vee \ \mathit{WhObject} \end{aligned}$$

and where *CanonicalSubject*, *WhSubject*... are defined as the actual fragments of the grammar:



Given the above definitions, a description such as *TransitiveVerb* is intended to describe the following tree schematas depicted in (1)[3]. That is each variant

[3] The combination of relative subject an a questioned object is rejected by the principle of extraction unicity (See section 6).

description of the subject embedded in the *Subject* clause is combined with each variant description of the object in the *Object* clause and the description in the *ActiveVerb* clause.

As it stands, the representation system we have introduced so far requires to set up two components: first we investigate which language to use for describing *tree fragments* and combining them (Section 4). Second we detail the language which controls how fragments are to be combined (Section 5).

4 A language for describing tree fragments

In this section, we consider two questions: (1) how to conveniently describe tree fragments, (2) how to flexibly constrain how such tree fragments maybe combined to form larger syntactic units. We first introduce a language of tree descriptions, and then show how it can be generalized to a family of formal languages parametrized by an arbitrary constraining decoration system that further limits how elements can be combined.

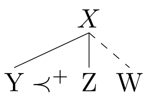
The base language L. Let $x, y, z \ldots$ be nodes variables. We write $\lhd$ for immediate dominance, $\lhd^*$ for its reflexive transitive closure (dominance), $\prec$ for immediate precedence (or adjacency) and $\prec^+$ for its transitive closure (strict precedence). We let ℓ range over a set of node labels generally intended to capture the notion of categories. A tree description ϕ has the following abstract syntax:

$$\phi \quad ::= \quad x \lhd y \;\mid\; x \lhd^* y \;\mid\; x \prec y \;\mid\; x \prec^+ y \;\mid\; x : \ell \;\mid\; \phi \wedge \phi \tag{2}$$

L-descriptions are, as expected, interpreted over first-order structures of finite, ordered, constructor trees. As usual, we limit our attention to minimal models.

Throughout the paper we use an intuitive graphical notation for representing tree descriptions. Though this notation is not sufficient to represent every expression of the language, it nonetheless generally suffices for the kind of trees typically used in natural language syntax. Thus, the description (D_0) on the left is graphically represented by the tree notation on the right:

$$D_0 = \begin{array}{l} x \lhd^* w \wedge x \lhd y \wedge x \lhd z \\ \wedge\; y \prec^+ z \wedge z \prec w \\ \wedge\; x : X \wedge y : Y \wedge z : Z \wedge w : W \end{array}$$

(D_0)  (3)

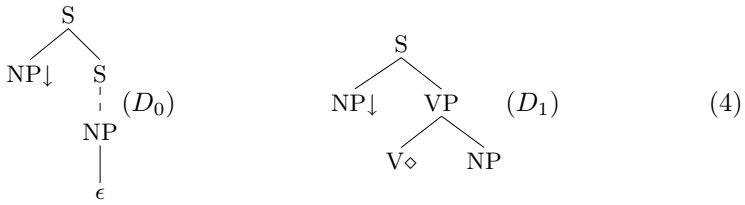
where immediate dominance is represented by a solid line, dominance by a dashed line, precedence by the symbol $\prec^+$ and adjacency is left unmarked.

A parametric family of languages. It is possible to more flexibly control how tree fragments maybe combined by adding annotations to nodes together with stipulations for how these annotations restrict admissible models and interpretations. In this manner, we arrive at the idea of a family of languages $L(C)$ parametrized by a *combination schema* C.

In the remainder of this section we discuss three instantiations of $L(C)$ that have been used for describing the lexicon of Tree Adjoining Grammars. The first

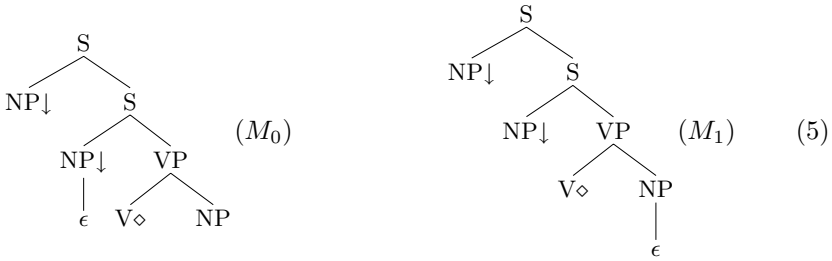
one, $L(\emptyset)$ is used by Xia [9], the second one $L(names)$ is used by Candito [10]. We show that neither $L(\emptyset)$ nor $L(names)$ are appropriate for describing the lexicon of a French TAG Grammar. We then introduce $L(colors)$ which we have used successfully for that purpose.

Language $L(\emptyset)$. This first instantiation of $L(C)$ is used by F. Xia[9]. This language does not use any combination constraint. The combination schema C is thus empty. Equipped with such a language we can independently describe fragments such as these[4]:



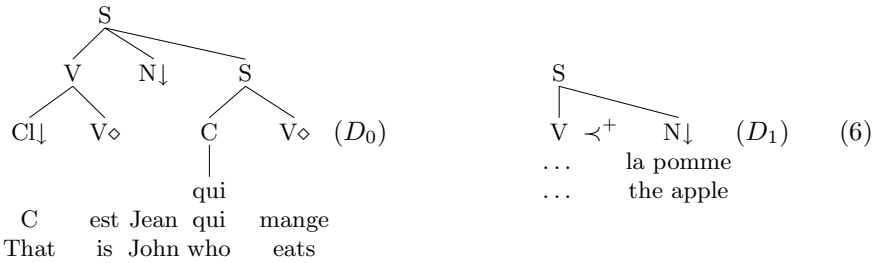
(4)

where D_0 describes a relative NP and D_1 a transitive construction. Their combination leads to the following two models:



(5)

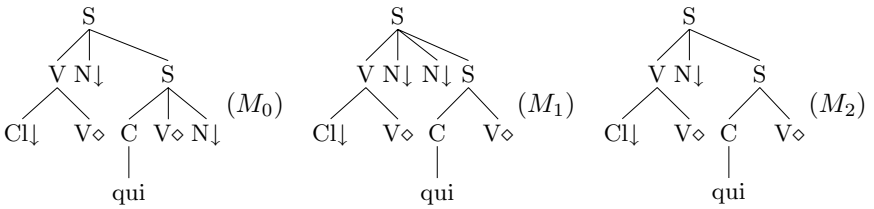
However this language faces an expressivity limit since, for the purpose of lexical organisation, linguists want to constrain combinations more precisely. For instance, in the French Grammar the following fragment composition is badly handled since:



(6)

[4] These fragments and the related models are those used by F. Xia in the context of the XTAG English Grammar.

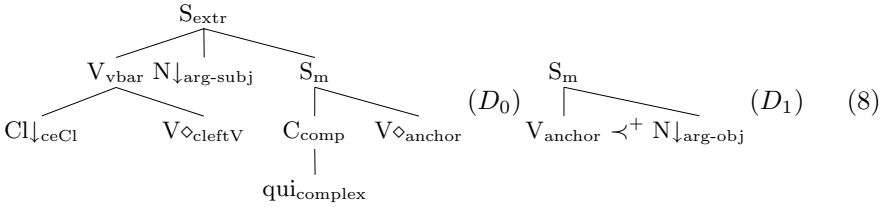
yields, among others, the following results:



(7)

where only M_0 is normally deemed linguistically valid.

Language $L(names)$. In her thesis, M.-H. Candito [10] introduces an instance of $L(C)$ that constrains combinations to avoid cases such as the one outlined above. The combination schema C is as follows: (1) a finite set of names where each node of a tree description is associated to such a name and (2) Two nodes sharing the same name are to be interpreted as denoting the same entity, hence when merging descriptions, only the nodes with the same names are merged. In other words, a model is valid if (1) every node has exactly one name and (2) there is at most one node with a given name[5].



(8)

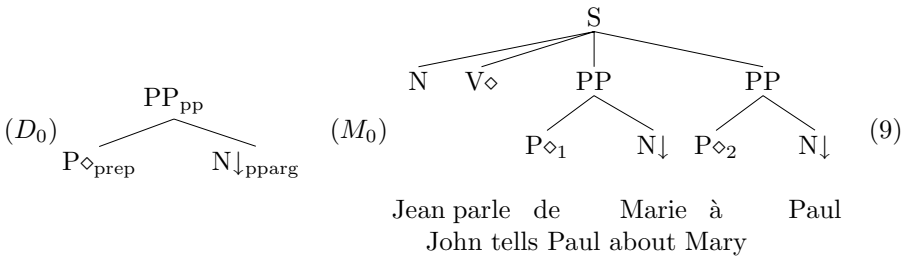
The model resulting from merging D_0 with D_1 is only M_0 depicted in (7). In such a case, $L(names)$ corrects the shortcomings of $L(\emptyset)$. However, during the development of a non trivial grammar using this language, it turned out that $L(names)$ was eventually unsatisfactory for two main reasons:

The first is practical and rather obvious: the grammar writer has to manage naming by hand, and must handle the issues arising from name collisions.

The second is more tricky: the grammar writer may need to use the same tree fragment more than once in the same description. For example, such an

[5] To be complete, M.-H. Candito uses additional operations to map multiple names on a single node. However this does not change the content of our actual discussion.

occasion arises in the case of a double PP complementation:



where one cannot use the fragment (D_0) more than once to yield M_0 since identical names must denote identically the same nodes.

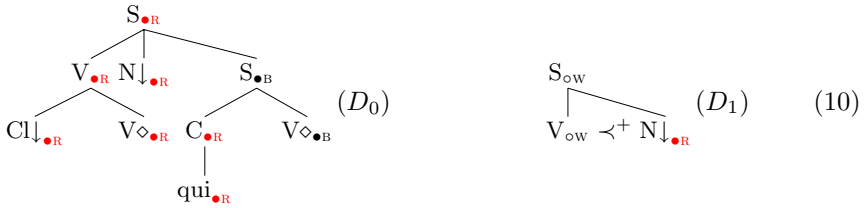
A Language with colored nodes $L(colors)$. We used this language in the development of a large scale French TAG patterned after the analysis of [7].

$L(colors)$ was designed to overcome the shortcomings of languages $L(\emptyset)$ and $L(names)$. We want (1) to be able to constrain more precisely the way fragments combine together than with language $L(\emptyset)$ (2) we want to eschew the explicit naming management of language $L(names)$.

To do this, the combination schema C used in $L(colors)$ decorates all nodes with colors: black ($\bullet_B$), white ($\circ_W$), red ($\bullet_R$) or failure ($\bot$). The additional condition on model admissibility is that each node must be either red or black.

When combining tree descriptions, nodes are merged and their colors combined. The table to the right specifies the result of combining two colors. For instance, combining a white node with a black node yields a black node; combining a white node with a red node is illegal and produces a failure. As a matter of illustration, the following color enriched descriptions yields only the desired model (M_0) for example number (7)[6]

	$\bullet_B$	$\bullet_R$	$\circ_W$	$\bot$
$\bullet_B$	$\bot$	$\bot$	$\bullet_B$	$\bot$
$\bullet_R$	$\bot$	$\bot$	$\bot$	$\bot$
$\circ_W$	$\bullet_B$	$\bot$	$\circ_W$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$



Intuitively the colors have a semantic similar to that of ressources and requirements systems such as Interaction Grammars [11]. A tree is well formed if it is saturated. The colors representing saturation are red or black the color representing non saturation is white and we have a color representing failure.

[6] We let the reader figure out how to express double PP complementation (9). It requires to use a description similar to (D_1) depicted here, patterned for describing a prepositional phrase though.

Though $L(colors)$ turned out to be satisfactory for designing a large scale French TAG, it might not be similarly adequate for other frameworks or languages.[7] However, alternative instances of $L(C)$ might be suitable. For example a combination schema based on polarities seems a very reasonable foundation for interaction grammars [11] and even for polarity based unification grammars [12].

5 Controlling fragment combinations

In Section 3 we identified a number of desirable requirements for a metagrammar language: (1) it should support disjunctions to make it easy to express diathesis (such as active, passive), (2) it should support conjunction so that complex descriptions can be assembled by combining several simpler ones, (3) it should support abstraction so that expressions can be named to facilitate reuse and avoid redundancy.

In this section, we introduce the language $\mathcal{L}_C$ to control how fragments can be combined in our proposed lexical representation framework, and show how $\mathcal{L}_C$ satisfies all the above requirements.

$$\text{Clause} ::= \text{Name} \rightarrow \text{Goal} \tag{11}$$

$$\text{Goal} ::= \text{Goal} \wedge \text{Goal} \mid \text{Goal} \vee \text{Goal} \mid \phi \mid \text{Name} \tag{12}$$

This language allows to manipulate fragment descriptions (ϕ), to express the composition of statements (Goal $\wedge$ Goal), to express nondeterministic choices (Goal$\vee$Goal), and finally to name complex statements for reuse (Name $\rightarrow$ Goal).

The main motivation for the control language is to support the combination and reuse of tree fragments. Instead of manipulating directly tree descriptions, the language allows to define abstractions over (possibly complex) statements. Thus, the clause:

$$\textit{CanonicalSubject} \rightarrow \begin{array}{c} \text{S} \\ \diagup \quad | \\ \text{N}\downarrow \quad \text{V} \end{array} \tag{13}$$

defines the abstraction *CanonicalSubject* to stand for a tree description which can be subsequently reused via this new name, while the clause:

$$\textit{TransitiveVerbActive} \rightarrow \textit{Subject} \wedge \textit{ActiveVerb} \wedge \textit{Object} \tag{14}$$

states that a lexical tree for a transitive verb is formed from the composition of the descriptions of a subject, of an object and of an active verb.

Disjunction is interpreted as an nondeterministic choice: each of the alternatives describes one of the ways in which the abstraction can be realized. As

[7] The current framework is not restricted to the specific case of Tree Adjoining Grammars. It should be straightforward to adapt it to other cases of tree based syntactic systems such as Interaction Grammars.

illustrated by lexical rules as used e.g. in PATR II [3], a system of lexical representation needs to be equipped with a way to express relationships between lexical items such as does a passive lexical rule relating an active and a passive lexical entry. In our approach, similar relations are expressed with disjunctions. Thus the following statement expresses the fact that various realisation of the subject are equivalent:

$$\begin{array}{rcl} \mathit{Subject} & \rightarrow & \mathit{CanonicalSubject} \\ & \vee & \mathit{WhSubject} \\ & \vee & \mathit{RelativeSubject} \\ & \vee & \mathit{CliticSubject} \end{array} \tag{15}$$

As surely has become evident, the language presented in this section has very much the flavor of a logic programming language. More precisely, it can be understood as an instance of the *Definite Clause Grammar* (DCG) paradigm. DCGs were originally conceived to express the production rules of context free grammars: they characterized the sentences of a language, i.e. all the possible ways words could be combined into grammatical sequences by concatenation. Here, instead of words, we have tree fragments, and instead of concatenation we have a composition operation. In other words, $\mathcal{L}_\mathcal{C}$ allows us to write the grammar of a tree grammar, which surely justifies the name *metagrammar*.

6 Principles of well-formedness

So far, the current system assumes that one can describe grammatical information by combining fragments of local information. There are however cases where the local fragments interact when realised together. To handle these interactions in an elegant way, the system allows to formulate additional global constraints on tree admissibility, called the principles.

Let us express in the control language the fact that a transitive verb is made of a subject, an object and a verb in the active form:

$$\mathit{TransitiveVerb} \rightarrow \mathit{Subject} \wedge \mathit{ActiveVerb} \wedge \mathit{Object} \tag{16}$$

$$\mathit{Subject} \rightarrow \mathit{CanonicalSubject} \vee \mathit{CliticSubject} \tag{17}$$

$$\mathit{Object} \rightarrow \mathit{CanonicalObject} \vee \mathit{CliticObject} \tag{18}$$

Clitic ordering According to the subject and object clauses, it is the case that among others, a description of a transitive verb is made of the composition of a clitic subject and a clitic object[8] whose definitions are as follows:

$$\mathit{CliticSubject} \rightarrow \begin{array}{ccc} & & \mathrm{V} \\ & \diagup & \diagdown \\ \mathrm{Cl}{\downarrow}[\mathrm{case} = \mathrm{nom}] & \prec^+ & \mathrm{V} \end{array} \qquad \mathit{CliticObject} \rightarrow \begin{array}{ccc} & & \mathrm{V} \\ & \diagup & \diagdown \\ \mathrm{Cl}{\downarrow}[\mathrm{case} = \mathrm{acc}] & \prec^+ & \mathrm{V} \end{array} \tag{19}$$

[8] In French, clitics are small non tonic pronominal particles realized in front of the verb which are ordered according to a fixed order. The problem of clitic ordering is a well known case of such an interaction. It was already described as problematic in the Generative litterature in the early 70's [13].

When realized together, none of the clitic descriptions say how these clitics are ordered relative to each other so that a merge of these two descriptions yields the following two models:

where M_1 is an undesirable solution in French.

French clitic ordering is thus rendered by a principle of tree well formedness: *sibling nodes of category Clitic have to be ordered according to the respective order of their ranking property.* So, if we take the case feature of descriptions (19) to be the ranking property, and that the order defined over the property constrains (inter alia) nominative to precede accusative then in every tree where both a nominative and an accusative clitic are realised, the principle ensures that only M_0 is a valid model.

Extraction unicity Another principle presented hereafter (Section 7) is that of extraction unicity. We assume that, in French, only one argument of a given predicate may be extracted[9]. Following this, the extraction principle is responsible for ruling out trees models where more than a node would be associated to the property of extraction.

Two other principles have actually been used in the implementation of the French Grammar: a principle for ensuring clitic unicity and a principle for expressing islands constraints[10]. The expression of an additional principle of functional unicity is currently under investigation.

7 A constraint satisfaction approach

As mentioned earlier, the control language $\mathcal{L}_C$ of Section 5 can be regarded as an instance of the *Definite Clause Grammar* (DCG) paradigm. While DCGs are most often used to describe sentences, i.e. sequences of words, here, we apply them to the description of formulae in language $L(colors)$, i.e. conjunctions of colored tree fragments.

A consequence of regarding a metagrammar, i.e. a program expressed in language $\mathcal{L}_C$, as a DCG is that it can be reduced to a logic program and executed as such using well-known techniques. What remains to be explained is how, from a conjunction of colored tree fragments, we derive all complete trees that can be formed by combining these fragments together.

For this task, we propose a constraint-based approach that builds upon and extends the treatment of dominance constraints of Duchier and Niehren [15]. We begin by generalizing slightly the language introduced in Section 4 to make

[9] Actually, cases of double extraction have been discovered in French, they are so rare and so unnatural that they are generally ruled out of the grammatical implementations.

[10] This principle is related to the way one formalises islands constraints in TAG [14].

it more directly amenable to the treatment described in [15], then we show how we can enumerate the minimal models of a description in that language by translating this description into a system of constraints involving set variables, and solving that instead.

Tree description language. In order to account for the idea that each node of a description is colored either red, black or white, we let x, y, z range over 3 disjoint sets of node variables: $V_{\mathrm{R}}, V_{\mathrm{B}}, V_{\mathrm{W}}$. We write $\lhd$ for immediate dominance, $\lhd^+$ for its transitive closure, i.e. strict dominance, $\prec$ for immediate precedence, and $\prec^+$ for its transitive closure, i.e. strict precedence. We let ℓ range over a set of node labels. A description ϕ has the following abstract syntax:

$$\phi \quad ::= \quad x\ R\ y \quad | \quad x \lhd y \quad | \quad x \prec y \quad | \quad x : \ell \quad | \quad \phi \wedge \phi \tag{20}$$

where $R \subseteq \{=, \lhd^+, \rhd^+, \prec^+, \succ^+\}$ is a set of relation symbols whose intended interpretation is disjunctive; thus $x\ \{=, \lhd^+\}\ y$ is more conventionally written $x \lhd^* y$.

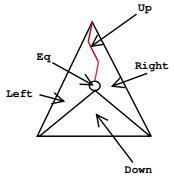
In [15], the abstract syntax permitted a literal of the form $x : \ell(x_1, \ldots, x_n)$ that combined (1) an assignment of the label ℓ to x, (2) immediate dominance literals $x \lhd x_i$, (3) immediate precedence literals $x_i \prec x_{i+1}$, (4) an arity constraint stipulating that x has exactly n children. Here we prefer a finer granularity and admit literals for immediate dominance and immediate precedence. For simplicity of presentation we omit an arity constraint literal.

Enumerating minimal models. We now describe how to convert a description into a constraint system that uses set constraints and such that the solutions of the latter are in bijection with the minimal models of the former. Such a constraint system can be realized and solved efficiently using the constraint programming support of Mozart/Oz. Our conversion follows very closely the presentation of [15].

The general intuition is that a literal $x\ R\ y$ should be represented by a membership constraint $y \in R(x)$ where $R(x)$ is a set variable denoting all the nodes that stand in R relationship with x. We write V^ϕ for the set of variables occurring in ϕ. Our encoding consists of 3 parts:

$$[\![\phi]\!] \quad = \quad \bigwedge_{x \in V^\phi} \mathsf{A}_1(x) \bigwedge_{x,y \in V^\phi} \mathsf{A}_2(x, y) \quad \wedge \quad \mathsf{B}[\![\phi]\!] \tag{21}$$

$\mathsf{A}_1(\cdot)$ introduces a node representation per variable, $\mathsf{A}_2(\cdot, \cdot)$ axiomatizes the treeness of the relations between these nodes, and $\mathsf{B}(\cdot)$ encodes the problem-specific restrictions imposed by ϕ.



7.1 Representation

When observed from a specific node x, the nodes of a solution tree (a model), and hence the variables which they interpret, are partitioned into 5 regions: the node denoted by x itself, all nodes below, all

nodes above, all nodes to the left, and all nodes to the right. The main idea is to introduce corresponding set variables Eq_x, Up_x, $Down_x$, $Left_x$, $Right_x$ to encode the sets of variables that are interpreted by nodes in the model which are respectively equal, above, below, left, and right of the node interpreting x. First, we state that x is one of the variables interpreted by the corresponding node in the model:

$$x \in Eq_x \tag{22}$$

Then, as explained above, we have the following fundamental partition equation:

$$V^\phi = Eq_x \uplus Up_x \uplus Down_x \uplus Left_x \uplus Right_x \tag{23}$$

where $\uplus$ denotes *disjoint union* We can (and in fact *must*, as proven in [15]) improve propagation by introducing shared intermediate results *Side*, *Eqdown*, *Equp*, *Eqdownleft*, *Eqdownright*.

$$Side_x = Left_x \uplus Right_x \qquad Eqdownleft_x = Eqdown_x \uplus Left_x \tag{24}$$

$$Eqdown_x = Eq_x \uplus Down_x \qquad Eqdownright_x = Eqdown_x \uplus Right_x \tag{25}$$

$$Equp_x = Eq_x \uplus Up_x \tag{26}$$

which must all be related to V^ϕ:

$$V^\phi = Eqdown_x \uplus Up_x \uplus Side_x \qquad V^\phi = Eqdownleft_x \uplus Up_x \uplus Right_x \tag{27}$$

$$V^\phi = Equp_x \uplus Down_x \uplus Side_x \qquad V^\phi = Eqdownright_x \uplus Down_x \uplus Left_x \tag{28}$$

We define $\mathsf{A}_1(x)$ as the conjunction of the constraints introduced above.

7.2 Welformedness

Posing $\mathbf{Rel} = \{=, \lhd^+, \rhd^+, \prec^+, \succ^+\}$, in a tree, the relationship that obtains between the nodes denoted by x and y must be one in **Rel**: the options are mutually exclusive. We introduce a variable C_{xy}, called a *choice variable*, to explicitly represent it and contribute a well-formedness clause $\mathsf{A}_3[\![x\ r\ y]\!]$ for each $r \in \mathbf{Rel}$.

$$\mathsf{A}_2(x,y) \quad = \quad C_{xy} \in \mathbf{Rel} \ \wedge \ \wedge\{\mathsf{A}_3[\![x\ r\ y]\!] \mid r \in \mathbf{Rel}\} \tag{29}$$

$$\mathsf{A}_3[\![x\ r\ y]\!] \quad \equiv \quad \mathsf{D}[\![x\ r\ y]\!] \wedge C_{xy} = r \quad \vee \quad C_{xy} \neq r \wedge \mathsf{D}[\![x\ \neg r\ y]\!] \tag{30}$$

For each $r \in \mathbf{Rel}$, it remains to define $\mathsf{D}[\![x\ r\ y]\!]$ and $\mathsf{D}[\![x\ \neg r\ y]\!]$ encoding respectively the relationships $x\,r\,y$ and $x\,\neg r\,y$ by set constraints on the representations of x and y.

$$\mathsf{D}[\![x = y]\!] = Eq_x = Eq_y \wedge Up_x = Up_y \wedge \ldots \tag{31}$$

$$\mathsf{D}[\![x\ \neg{=}\ y]\!] = Eq_x \parallel Eq_y \tag{32}$$

$$\mathsf{D}[\![x \lhd^+ y]\!] = Eqdown_y \subseteq Down_x \ \wedge \ Equp_x \subseteq Up_y \ \wedge \ Left_x \subseteq Left_y \ \wedge \ Right_x \subseteq Right_y \tag{33}$$

$$\mathsf{D}[\![x\ \neg{\lhd}^+ y]\!] = Eq_x \parallel Up_y \ \wedge \ Down_x \parallel Eq_y \tag{34}$$

$$\mathsf{D}[\![x \prec^+ y]\!] = Eqdownleft_x \subseteq Left_y \ \wedge \ Eqdownright_y \subseteq Right_x \tag{35}$$

$$\mathsf{D}[\![x\ \neg{\prec}^+ y]\!] = Eq_x \parallel Left_y \ \wedge \ Right_x \parallel Eq_y \tag{36}$$

where $\|$ represents disjointness.

7.3 Problem-specific constraints

The third part $\mathsf{B}[\![\phi]\!]$ of the translation forms the problem-specific constraints that further restrict the admissibility of well-formed solutions and only accepts those which are models of ϕ. The translation is given by case analysis following the abstract syntax of ϕ:

$$\mathsf{B}[\![\phi \wedge \phi']\!] \quad = \quad \mathsf{B}[\![\phi]\!] \wedge \mathsf{B}[\![\phi']\!] \tag{37}$$

A rather nice consequence of introducing 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 $C_{xy} \in \{=, \lhd^+\}$. More generally:

$$\mathsf{B}[\![x\, R\, y]\!] \quad = \quad C_{xy} \in R \tag{38}$$

A labeling literal $x : \ell$ simply restricts the label associated with variable x:

$$\mathsf{B}[\![x : \ell]\!] \quad = \quad \mathit{Label}_x = \ell \tag{39}$$

An immediate dominance literal $x \lhd y$ not only states that $x \lhd^+ y$ but also that there are no intervening nodes on the spine that connects the two nodes:

$$\mathsf{B}[\![x \lhd y]\!] \quad = \quad C_{xy} = \lhd^+ \ \wedge \ \mathit{Up}_y = \mathit{Equp}_x \tag{40}$$

An immediate precedence literal $x \prec y$ not only states that $x \prec^+ y$ but also that there are no intervening nodes horizontally between them:

$$\mathsf{B}[\![x \prec y]\!] = C_{xy} = \prec^+ \ \wedge \ \mathit{Eqdownleft}_x = \mathit{Left}_y \ \wedge \ \mathit{Right}_x = \mathit{Eqdownright}_y \tag{41}$$

7.4 Well-coloring

While distinguishing left and right is a small incremental improvement over [15], the treatment of colors is a rather more interesting extension. The main question is: which nodes can or must be identified with which other nodes? Red nodes cannot be identified with any other nodes. Black nodes may be identified with white nodes. Each white node must be identified with a black node. As a consequence, for every node, there is a unique red or black node with which it is identified. We introduce the (integer) variable RB_x to denote the red or black node with which x is identified.

For a red node, x is identified only with itself:

$$x \in V_{\mathrm{R}} \qquad \Rightarrow \qquad RB_x = x \ \wedge \ Eq_x = \{x\} \tag{42}$$

For a black node, the constraint is a little relaxed (it may also be indentified with white nodes):

$$x \in V_{\mathrm{B}} \qquad \Rightarrow \qquad RB_x = x \tag{43}$$

Posing $V_{\mathrm{B}}^{\phi} = V^{\phi} \cap V_{\mathrm{B}}$, each white node must be identified with a black node:

$$x \in V_{\mathrm{W}} \quad \Rightarrow \quad RB_x \in V_{\mathrm{B}}^{\phi} \tag{44}$$

Additionally, it is necessary to ensure that $RB_x = RB_y$ iff x and y have been identified. We can achieve this simply by modifiying the definition (32) of $\mathsf{D}[\![x \neg{=} y]\!]$ as follows:

$$\mathsf{D}[\![x \neg{=} y]\!] \quad = \quad Eq_x \parallel Eq_y \ \wedge \ RB_x \neq RB_y \tag{45}$$

7.5 Extraction Principle

As an illustration of how the framework presented so far can be extended with linguistically motivated principles to further constrain the admissible models, we describe now what we have dubbed the *extraction principle*.

The description language is (somehow) extended to make it possible to mark certain nodes of a description as representing an *extraction*. The extraction principle then makes the additional stipulation that, to be admissible, a model must contain at most one node marked as extracted.

Let V_{XTR}^{ϕ} be the subset of V^{ϕ} of those node variables marked as extracted. We introduce the new boolean variable $\mathit{Extracted}_x$ to indicate whether the node denoted by x is extracted:

$$\mathit{Extracted}_x \quad = \quad Eq_x \cap V_{\mathrm{XTR}}^{\phi} \neq \emptyset \tag{46}$$

Posing $V_{\mathrm{RB}}^{\phi} = V^{\phi} \cap (V_{\mathrm{R}} \cup V_{\mathrm{B}})$, and freely identifying the boolean values false and true respectively with the integers 0 and 1, the extraction principle can be enforced with the following constraint:

$$\sum_{x \in V_{\mathrm{RB}}^{\phi}} \mathit{Extracted}_x \quad < \quad 2 \tag{47}$$

8 Conclusion

This paper introduces a core abstract framework for representing grammatical information of tree based syntactic systems. Grammatical representation is organised around two central ideas: (1) the lexicon is described by means of elementary tree fragments that can be combined. (2) Fragment combinations are handled by a control language, which turns out to be an instance of a DCG.

The framework described here, generalises the TAG specific approaches of [9, 10]. We have provided a parametric family of languages for tree composition as well as constraints on tree well formedness.

Besides the non TAG specific tree composition language, it mostly differs from the TAG instanciations by (1) it introduces a control language allowing to express explicitly composition of fragments as well as variants of related lexical

entries. The two existing systems of [10] and [9] rely mostly on an algorithmic device for expressing variants, namely a crossing algorithm for [10], and an external module of lexical rules for [9].

The introduction of the control language (1) avoids to work with different modules and (2) introduces more flexibility in expressing variants that avoids to deal with "shadow" classes as it turns out to be the case in [10].

The framework presented here has been extensively tested against the development of a large sized French TAG based on [7]. This grammar covers most of the phenomenons related to the syntax of French verbs.

References

1. Prolo, C.: Generating the xtag english grammar using metarules. In: Proc. COLING 2002, Taiwan (2002)
2. Joshi, A.K., Schabès, Y.: Tree adjoining grammars. In Rozenberg, G., Salomaa, A., eds.: Handbook of Formal Languages. Springer Verlag, Berlin (1997)
3. Shieber, S.M.: The design of a computer language for linguistic information. In: Proceedings of the Tenth International Conference on Computational Linguistics, Stanford University, Stanford, California (1984) 362–366
4. Kaplan, R.M., Maxwell, J.T.: Lfg grammar writer's workbench. Technical report, Xerox PARC (1996)
5. Meurers, W.D.: On implementing an hpsg theory – aspects of the logical architecture, the formalization, and the implementation of head-driven phrase structure grammars. In: Erhard W. Hinrichs, W. Detmar Meurers, and Tsuneko Nakazawa: *Partial-VP and Split-NP Topicalization in German – An HPSG Analysis and its Implementation*. Arbeitspapiere des SFB 340 Nr. 58, Universität Tübingen (1994)
6. XTAG Research Group: A lexicalized tree adjoining grammar for english. Technical Report IRCS-01-03, IRCS, University of Pennsylvania (2001)
7. Abeillé, A.: Une grammaire d'arbres adjoints pour le français. Editions du CNRS, Paris (2002)
8. Koenig, J.P., Jurafsky, D.: Type underspecification and on-line type construction in the lexicon. In: Proceedings of WCCFL94. (1995)
9. Xia, F.: Automatic Grammar Generation from two Different Perspectives. PhD thesis, University of Pennsylvania (2001)
10. Candito, M.H.: Organisation Modulaire et Paramétrable de Grammaires Electroniques Lexicalisées. PhD thesis, Université de Paris 7 (1999)
11. Perrier, G.: Les grammaires d'interaction. Université Nancy 2 (2003) Habilitation à diriger des recherches.
12. Kahane, S.: Grammaires d'unification polarisées. In: Proc. TALN 2004, Fès (2004)
13. Perlmutter, D.: Surface structure constraints in syntax. Linguistic Inquiry **1** (1970) 187–255
14. Kroch, A., Joshi, A.K.: The linguistic relevance of tree adjoining grammar. Technical report, IRCS, Philadelphia (1985)
15. Duchier, D., Niehren, J.: Dominance constraints with set operators. In: Proceedings of the First International Conference on Computational Logic (CL2000). Volume 1861 of Lecture Notes in Computer Science., Springer (2000) 326–341

Multi-dimensional Graph Configuration for Natural Language Processing

Ralph Debusmann[1], Denys Duchier[2], and Marco Kuhlmann[1]

[1] Programming Systems Lab, Saarland University, Saarbrücken, Germany
{rade,kuhlmann}@ps.uni-sb.de
[2] Équipe Calligramme, LORIA, Nancy, France
duchier@loria.fr

Abstract. We introduce the new abstract notion of *multi-dimensional lexicalized graph configuration problems*, generalising over many important tasks in computational linguistics such as semantic assembly, surface realization and syntactic analysis, and integrating them. We present *Extensible Dependency Grammar* (XDG) as one instance of this notion.

1 Introduction

A variety of tasks in computational linguistics can be regarded as *configuration problems* (CPs) [1]. In this paper, we introduce the notion of *lexicalised, multi-dimensional* CPs, a particular class of configuration problems that both has a wide range of linguistic applications, and can be solved in a straightforward way using state-of-the-art constraint programming technology. Linguistic modeling based on multi-dimensional CPs brings two major benefits: complete modularity of the developed resources, and tight integration of the individual modules.

We first consider configuration problems where the input is a set of components, and the output is a valid assembly of these components that satisfies certain problem-specific constraints. We then broaden the perspective to permit ambiguity as to the choice of each component. We call a configuration problem *lexicalized* when, as is the case in typical linguistic applications, this ambiguity is stipulated by a lexicon. Finally, to permit modeling with multiple levels of linguistic description (e.g. syntax, linear precedence, predicate/argument structure ...), we introduce the notion of a multi-dimensional configuration problem where we must simultaneously solve several configuration problems that are not independent, but rather constrain each other.

We then provide an introduction to Extensible Dependency Grammar (XDG) which is a development environment for linguistic modeling embracing the approach of lexicalised, multi-dimensional CPs.

The methodology based on lexicalized, multi-dimensional configuration problems is attractive for several reasons: for linguistic modeling, it is possible, for each level of linguistic description, to design a dimension that is especially well suited to it. For constraint-based processing, an inference on any dimension may help disambiguate another.

2 Configuration Problems in NLP

In this section, we develop the point that many tasks in computational linguistics can be usefully regarded as instances of *configuration problems*. We illustrate this point with three representative examples for which we have developed constraint-based processing techniques: semantic assembly, surface realization, and syntax analysis.

2.1 Semantic Assembly

We first turn to the task of assembling the semantic representation of a sentence from the individual fragments of representation contributed by its words in the context of *scope ambiguity*. Consider the following sentence:

(1) Every researcher deals with a problem.

This sentence has two readings, which may be disambiguated by the following continuations:

(1a) ... Some of these problems may be unsolvable.
(1b) ... This problem is his funding.

If represented in terms of Montague-style semantics, the two readings could be rendered as follows:

$$\forall x\colon \mathsf{researcher}(x) \rightarrow \exists y\colon \mathsf{problem}(y) \wedge (\mathsf{deal\ with})(x, y) \qquad (1a)$$

$$\exists y\colon \mathsf{problem}(y) \wedge \forall x\colon \mathsf{researcher}(x) \rightarrow (\mathsf{deal\ with})(x, y) \qquad (1b)$$

Notice that both these terms are made up of exactly the same "material":

$$\forall x\colon (\mathsf{researcher}(x) \rightarrow \dots) \qquad \exists y\colon (\mathsf{problem}(y) \wedge \dots) \qquad (\mathsf{deal\ with})(x, y)$$

The only difference between the two is the relative ordering of these term fragments: in (1a), the universal quantifier takes scope over the existential quantifier; in (1b), it is the other way round. Formalisms for *scope underspecification* [2–4] aim for a compact representation of this kind of ambiguity: they can be used to describe the common parts of a set of readings, and to express constraints on how these fragments can be "plugged together".

The left half of Fig. 1 shows an underspecified graphical representation of the two readings of 1 in the formalism of *dominance constraints* [4]. The solid edges in the picture mark the term fragments that are shared among all readings. Two fragments can combine by "plugging" one into an open "hole" of the other. The dashed edges mark *dominance requirements*, where dominance corresponds to ancestorship. For instance, the fragments for "every researcher" and for "a problem" dominate the "deal with" fragment, i.e. both must be ancestors of the latter. With the given dominance requirements, exactly two configurations of the fragments are possible (shown schematically in the right half of Fig. 1); these two configurations correspond to the readings (1a) and (1b).

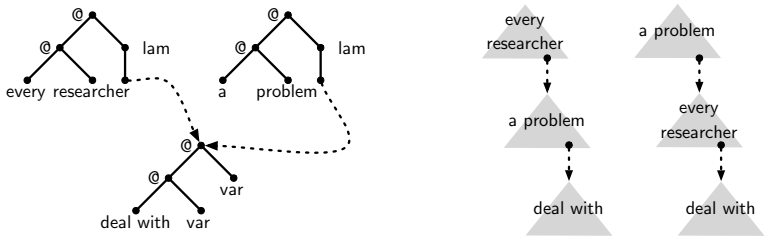


Fig. 1. A dominance constraint for the two readings of (1), and its two solutions

2.2 Surface realisation

Surface realisation is the sub-task of natural language generation that maps a semantic representation to a grammatical surface string. More specifically, for some given grammar, surface realisation takes as its input a bag of semantic descriptions, ϕ, and returns as its output a syntax tree containing the verbalisation of ϕ.

Here we discuss surface realisation for Tree Adjoining Grammar (TAG) [5]. One of the underlying design principles of many TAG grammars is *semantic minimality*: each lexical entry (*elementary tree*) of a TAG grammar corresponds to an atomic semantics. Surface realisation then can be reduced to the problem of *selecting* for each semantic atom a matching elementary tree, and *assembling* these trees into a *derivation tree* using the standard TAG operations that combine grammatical structures, namely substitution and adjunction [6].

We illustrate this by means of an example. Assume that we want to realise the following (event-based) input semantics using some given TAG grammar G:

$$x = Peter, \quad see(e, x, y), \quad some(x), \quad fat(x), \quad rabbit(x)\,.$$

(Note that all atoms are considered to be ground.) In a first step, we need to choose for each of the semantic atoms an elementary tree from G that verbalises its semantics. A sample selection of trees is shown in the left half of Fig. 2. The dashed arrows in the figure indicate a way to compose the chosen elementary trees by means of substitution and ajunction. For example, the tree realising the semantic atom $fat(x)$ can adjoin into the root node (labelled with N) of the tree realising the semantics of $rabbit(x)$. The right half of Fig. 2 shows the resulting derivation tree for the sentence. In a post-processing step, this derivation tree can be transformed into a *derived tree,* whose yield is a possible realisation of the intended semantics.

2.3 Syntactic Analysis

Our final example is the parsing of dependency grammars. As we have seen, surface realisation can be reduced to the configuration of a labelled tree in which

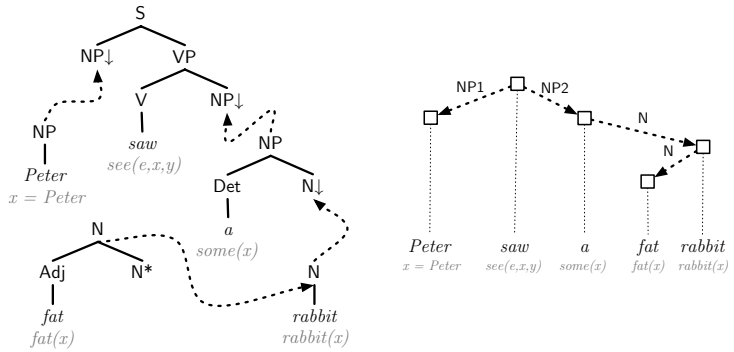
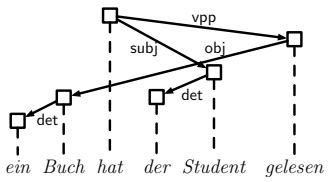


Fig. 2. Generation

the nodes are labelled with lexical entries (elementary trees), and the edges are labelled with sites for substitution and adjunction. It has often been noted that these derivation trees closely resemble dependency trees.

A *dependency tree* for a sentence s is a tree whose nodes are labelled with the words of s, and whose (directed) edges are labelled with antisymmetric *grammatical relations* (like *subject-of* or *adverbial-modifier-of*). Given an edge $u - \rho \rightarrow v$ in a dependency tree, u is called the *head* of u, and v is called the *dependent* of v. ρ is the grammatical relation between the two. A *dependency grammar* consists of a lexicon and a *valency* assignment for the lexical entries that specifies the grammatical relations a given entry must or may participate in as head and as dependent. A dependency tree is *licensed* by a given grammar if for every edge $u - \rho \rightarrow v$, the relation ρ is a grammatical relation licensed by both the lexical entries for u and v.



Lexical entry	*incoming*	*outgoing*
ein	{det}	{}
Buch	{subj, obj}	{det}
hat	{}	{subj, vpp}
der	{det}	{}
Student	{subj, obj}	{det}
gelesen	{vpp}	{obj}

Fig. 3. A dependency tree for the German sentence

The left half of Fig. 3 shows a dependency tree for the sentence

(2) Ein Buch hat der Student gelesen.

The right half of the figure shows a valency assignment with which this tree would be licensed: it specifies possible incoming edges and required outgoing edges. For example, the lexical entry for *Student* can act both as a subject and an object dependent of its head, and itself requires a dependent determiner.

3 Graph Configuration Problems

The problems described in the previous section have this in common: the task is always one where we are given a number of tree (or graph) fragments, and we must plug them together (under constraints) to obtain a complete assembly. We call this class of problems *graph configuration problems*.

For such problems, it is often convenient to represent the plugging of a fragment w into another w' by means of a directed labeled edge $w{-}\ell{\rightarrow}w'$ which makes explicit that a resource of type ℓ supplied by w' is being matched with a corresponding need in w.

We are thus led to the notion of (finite) labeled graphs. Consider given a finite set $\mathcal{L}$ of labels. A $\mathcal{L}$-labeled graph (V, E) consists of a set V of nodes and a set $E \subseteq V \times V \times \mathcal{L}$ of directed labeled edges between them. We can interpret each label ℓ as a function from node to set of nodes defined a follows:

$$\ell(w) = \{w' \mid w{-}\ell{\rightarrow}w' \in E\}$$

$\ell(w)$ represents the set of immediate successors of w that can be reached by traversing an edge labeled ℓ. Duchier [7] developed this set-based approach and showed e.g. how to formulate a constraint system that precisely characterizes all labeled trees which can be formed from the finite set of nodes V and the finite set of edge labels $\mathcal{L}$. This system is formulated in terms of finite set variables $\ell(w)$, $\mathsf{daughters}(w)$, $\mathsf{down}(w)$, $\mathsf{eqdown}(w)$ and roots:

$$\begin{aligned}
& V = \mathsf{roots} \uplus \uplus\{\mathsf{daughters}(w) \mid w \in V\} \\
\wedge\ & |\mathsf{roots}| = 1 \\
\wedge\ & \forall w \in V \\
& \quad (\forall \ell \in \mathcal{L} \quad \ell(w) \subseteq V) \\
& \wedge\ \mathsf{eqdown}(w) = \{w\} \uplus \mathsf{down}(w) \\
& \wedge\ \quad \mathsf{down}(w) = \cup\{\mathsf{eqdown}(w') \mid w' \in \mathsf{daughters}\} \\
& \wedge\ \quad \mathsf{daughters} = \uplus\{\ell(w) \mid \ell \in \mathcal{L}\}
\end{aligned} \tag{3}$$

By taking advantage of the constraint programming support available in a language such as Mozart/Oz, this formal characterization of finite labeled trees can be given straightforwardly a computational interpretation.

Using this approach as a foundation, we can encode graph configuration problems with additional constraints. For example, each node typically offers a specific subset of resources. Such a restriction can be enforced by posing constraints on the cardinality of $\ell(w)$ (for $\ell \in \mathcal{L}$), e.g. $|\ell(w)| = 0$ for a resource not offered by w, $|\ell(w)| = 1$ for a resource offered exactly once, etc... This is how we can model subcategorization in the application to dependency parsing.

4 Lexicalized Configuration Problems

The view presented sofar, where we are given a fixed number of fragments to be plugged together into a fully configured assembly, is not yet sufficient to capture realistic tasks. In the surface realization problem, there may be several alternative ways to verbalize the same semantic element. Similarly, in the syntax analysis problem, one word may have several readings, alternative subcategorization frames, alternative linearization constructions.

We generalize the previous view by replacing each fragment with a finite collection of fragments from which one must be selected. Since the mapping from nodes to collection of alternative fragments is often stipulated by a lexicon, we call such problems *lexicalized configuration problems*.

The challenge is now to adapt the constraint-based approach outlined earlier to gracefully handle lexical ambiguity.

4.1 Selection constraints

Let's consider again the dependency parsing application. As described in the previous section, for a given set V of words, we can (a) model the possible dependency trees as the solutions of constraint system (3), and (b) we can enforce subcategorization frames using cardinality constraints on ℓ-daughter sets $\ell(w)$.

If we now assume that w has k lexical entries, each one may stipulate a different cardinality constraint for $\ell(w)$. Clearly, in order to avoid combinatorial explosion, we do not want to try all possible combinations of selection for the given words. Instead, we would like to take advantage of constraint propagation to avoid having to make non-deterministic choices.

What we want is an underspecified representation of the lexical entry that is ultimately chosen in a form that integrates well in a constraint-based formulation. This is the purpose of the *selection constraint*:

$$X = \langle Y_1, \ldots, Y_n \rangle [I]$$

Its declarative semantics is $X = Y_I$. All of X, Y_i and I may be variables and propagation takes place in both directions: into X in a manner similar to constructive disjunction, and into I whenever a Y_k becomes incompatible with X (and thus k can be removed from the domain of I).

We have implemented support for selection constraints for integer variables (FD) and integer set variables (FS). This support can easily be lifted over feature structures as follows:

$$\left\langle \begin{bmatrix} \mathsf{f}_1 = v_1^1 \\ \vdots \\ \mathsf{f}_p = v_p^1 \end{bmatrix}, \ldots, \begin{bmatrix} \mathsf{f}_1 = v_1^k \\ \vdots \\ \mathsf{f}_p = v_p^k \end{bmatrix} \right\rangle [I] \quad = \quad \begin{bmatrix} \mathsf{f}_1 = \langle v_1^1, \ldots, v_1^k \rangle [I] \\ \vdots \\ \mathsf{f}_p = \langle v_p^1, \ldots, v_p^k \rangle [I] \end{bmatrix} \tag{4}$$

and, in this manner, can apply to complex lexical entries. Notice how features f_1 through f_p are constrained by concurrent selection constraints which are all covariant because they share the same selector I.

5 Multi-dimensional Configuration Problems

Sofar, we have informally introduced the notion of lexicalized graph configuration problems and suggested how they can be encoded into systems of contraints that are adequately supported by corresponding constraint technology.

However, when modeling language, we must contend with multiple descriptive levels (e.g. syntax, linear precedence, predicate/argument structure, information structure, etc. . .). Each is concerned with a different aspect of linguistic description, yet they are not independent: they interact and constrain each other.

Our notion of lexicalized configuration problems is an excellent tool for modeling each individual descriptive level, but, as we said, these levels are not independent. For this reason, we now propose a generalized approach where we have several configuration problems, all simultaneously constrained by the same lexical entries.

This is what we call a *multi-dimensional configuration problem*. For each descriptive level, there is one dedicated dimension. Every lexical entry now contains a sub-lexical entry for each dimension. In this manner, each lexical entry simultaneously constrains all dimensions. Furthermore, we also permit *inter-dimensional principles*, i.e. global constraints that relate one dimension with another. In this manner, they are not merely coupled through the lexicon, but also by linguistically motivated well-formedness relations.

Figure 4 illustrates the selection constraint operating simultaneously on 3 dimensions over 3-dimensional lexical entries.

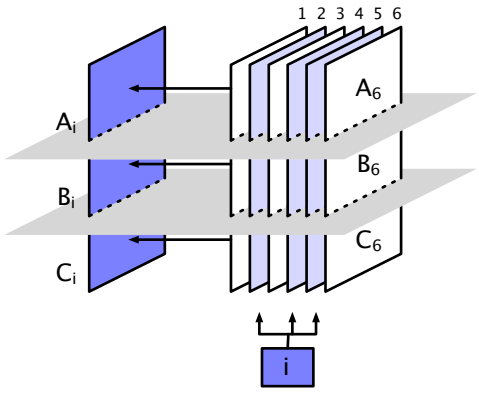


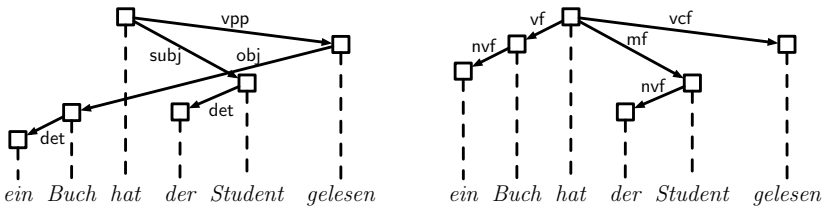
Fig. 4. Multi-dimensional selection

An important methodological motivation for the generalisation to multi-dimensional graph configuration problems is the desire to obtain a modular and scalable framework for modeling linguistic phenomena. We now illustrate the idea with the dependency analysis of word order phenomena in German. Consider again the parsing example sentence from Section 2:

(5) Ein Buch hat der Student gelesen.

It illustrates object topicalisation in German. Starting from the "canonically" ordered sentence *Der Student hat ein Buch gelesen,* it may be construed as the result of the fronting of *ein Buch* (the object of the verb), and a successive movement of *der Student* (the subject) to the now vacant object position – but a far more natural and perspicuous account is obtained when separating constituency and linear precedence, and describing word order variation as a relation between those two structures. Fig. 5 shows the analysis of the sentence using Topological Dependency Grammar (TDG, [8]) which dedicates one dimension to *immediate dominance* (ID) and another to *linear precendence* (LP).

Each dimension has its own set of well-formedness conditions (*principles*): both ID and LP structures are required to be trees, but LP structures must also be ordered and projective. Moreover, a *multi-dimensional principle* called *climbing* constrains the LP tree to be a flattening of the ID tree.



(a) Immediate Dominance (b) Linear Precedence

Fig. 5. Topicalisation

The parsing task of Topological Dependency Grammar is an example for a multi-dimensional graph configuration problem according to the characterisation above: to find the structures licensed by a given input, one has to find all triples $(V, T_{\mathsf{id}}, T_{\mathsf{lp}})$ in which T_{id} is a licensed ID tree, T_{lp} is a licensed LP tree, the two trees share the same node set V, and the climbing relation holds between them. Section 6 describes XDG which is an extended version of this model with additional dimensions for deep syntax, predicate/argument structure, and scope.

How does the multi-dimensional graph model relate to other approaches? With various *multi-stratal* formalisms, like Lexical Functional Grammar (LFG) [9] and Meaning-Text Theory (MTT) [10], it shares the idea of distinguishing several representational structures. In contrast to these formalisms, however, it does not presuppose a *layered* representation (where only adjacent layers can share information directly), nor does it assume that information can only flow in one direction: multi-dimensional principles can connect arbitrary dimensions. However, given that all dimensions share the same set of nodes, multi-dimensional

graphs also exhibit many of the benefits that arise through the tight integration of information as it is obtained in *mono-stratal* formalisms like HPSG [11].

6 Extensible Dependency Grammar

In this section, we introduce Extensible Dependency Grammar (XDG) [12], our flagship instance of a lexicalized multi-dimensional configuration problem. XDG is a generalization of TDG. It supports the use of arbitrary many dimensions of linguistics representation, and of arbitrary *principles*[3] regulating the well-formedness conditions. XDG was devised to be maximally general, and allow the grammar writer the freedom to decide how to split up linguistic modeling into separate dimensions.

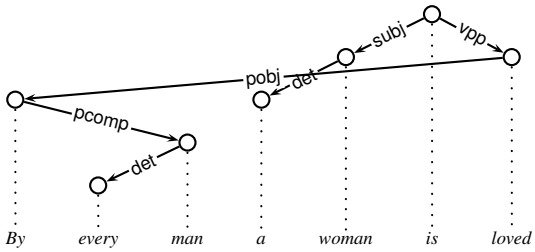
6.1 Example

As an illustrative example, we propose a five-dimensional grammar and illustrate it on the following example, an English passive construction paraphrasing the ubiquitous linguistic example "Every man loves a woman":

By every man, a woman is loved. (6)

In the following, we give a quick tour through the five dimensions of our example grammar to see what aspects of the linguistic analysis of this sentence they cover.

ID dimension The ID dimension (where ID stands for *immediate dominance*) was already introduced for TDG, and represents the linguistic aspect of *grammatical function*. We display the ID analysis of the example below:

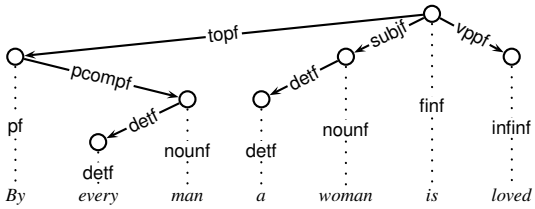


(7)

Here, "woman" is the subject (edge label subj) of the finite verb "is". "a" is the determiner (edge label det) of "woman". "loved" is the verbal past participle (vpp) of "is". Moreover, "by" is the prepositional object (pobj) of "loved", "man" the prepositional complement (pcomp) of "by", and finally "every" the determiner of "man".

[3] Currently instantiated from a library of parametric principles.

LP dimension The LP dimension (LP stands for *linear precedence*) represents the linguistic aspect of *word order*, and has also already been introduced in the preceding section for TDG. On the LP dimension, the edge labels are names for word positions[4]:

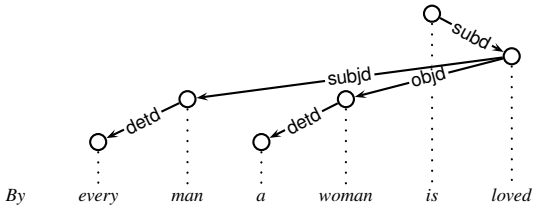


(8)

Here, the finite verb "is" is the root. "by" is in the topicalization position (topf), "woman" in the subject position (subjf), and "loved" in the verbal past participle position (vppf). Furthermore, "man" is in the prepositional complement position of "by" (pcompf), and "every" in the determiner position of "man" (detf). Similarly, "a" is in the determiner position of "woman" (detf).

On the LP dimension, we additionally annotate the nodes with node labels (displayed on the dotted projection edges connecting nodes and words). These are required for specifying the relative order of mothers and their daughters, e.g. that a determiner in the determiner position (edge label detf) of a noun must precede the noun (node label nounf).

DS dimension The DS dimension (DS stands for *deep syntax*) represents an intermediate structure between syntax and semantics. In this dimension, e.g. function words such as the preposition "by" or "to"-particles are not connected since they have no impact on the semantics. Also, constructions such as control, raising and passive are already resolved on the DS dimension, to enable a more seamless transition to semantics. Below is an example DS analysis:[5]:



(9)

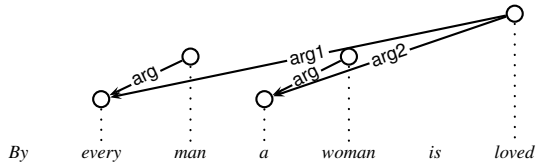
Here, "loved" is subordinated to "is" (edge label subd). "man" is the deep subject (subjd) of "loved", and "woman" the deep object (objd). "every" is the

[4] Following work on TDG, we adopt the convention to suffix LP edge labels with "f" for "field" to better distinguish them from ID edge labels.

[5] We adopt the convention to suffix DS edge labels with "d" for "deep" to better distinguish them from ID edge labels.

determiner of "man", and "a" the determiner of "woman". Notice that in this example, the relations of deep subject and deep object do not match the relations of subject and prepositional object on the ID dimension, due to the passive construction. Whereas in the ID analysis, "woman" is the subject of the auxiliary "is", and "by" the prepositional object of "loved", the DS analysis mirrors the underlying predicate-argument structure much more closely. Here, "woman" is the deep object of "loved", whereas "man" is its deep subject.

PA dimension The PA dimension (PA for *predicate-argument structure*) represents semantic predicate-argument structure, or, in terms of logic, *variable binding*. The idea is that quantifiers such as the determiners "every" and "a" introduce variables, which can then be bound by predicates (e.g. common nouns or verbs):

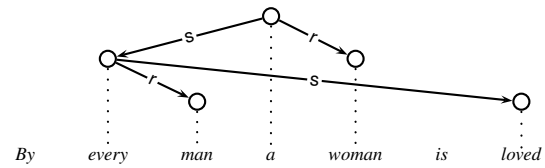


(10)

In the example analysis, think of the quantifier "every" as introducing variable x and "a" as introducing the variable y. Now, "man" binds the x, and "woman" the y. Finally, the x is the first argument of the predicate expressed by "loved", and y is the second argument. I.e. the PA analysis represents a flat semantics in the sense of [13], where the semantics of a sentence is represented by a multiset of first order predicates, and scopal relationships are completely omitted:

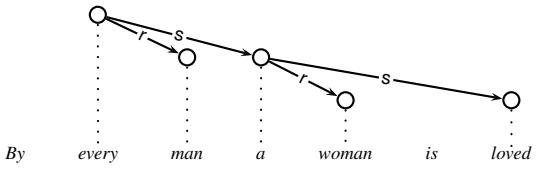
$$\{every(x), man(x), a(y), woman(y), love(x, y)\} \tag{11}$$

SC dimension The SC dimension (SC for *scope structure*) represents semantic scope structure. Contrary to [13], who advocates a semantic representation which completely omits scopal relationships, we follow of Minimal Recursion Semantics (MRS) [3] and have both: the PA dimension represents a flat semantics, and the SC dimension the scopal relationships. Here, e.g. quantifiers such as "every" and "a" have a restriction and a scope:



(12)

"a" has "woman" in its restriction (edge label r), and "every" in its scope (edge label s), and "every" has "man" in its restriction, and "loved" in its scope. Notice that this is only one of the two possible scope readings of the sentence — the "strong" reading where the existential quantifier outscopes the universal quantifier. The SC analysis representing the other, "weak" reading is depicted below:



(13)

6.2 Principles and the lexicon

XDG describes the well-formedness conditions of an analysis by the interaction of *principles* and the *lexicon*. The principles stipulate restrictions on one or more of the dimensions, and are controlled by the feature structures assigned to the nodes from the lexicon. The principles are drawn from an extensible *principle library*. The principle library already contains the necessary principles to model the syntax and semantics for large fragments of German and English, and smaller fragments of Arabic, Czech and Dutch. We present a representative subset of it below.

Tree principle. Dimension i must be a tree. In the example above, we use this principle on the ID, LP and SC dimensions.

DAG principle. Dimension i must be a directed acyclic graph. We use this principle on the DS and PA dimensions, which need not necessarily be trees.

Valency principle. For each node on dimension i, the incoming edges must be licensed by the in specification, and the outgoing edges by the out specification. This is a key principle in XDG, and used on all dimensions. It is also lexicalized (cf. the lexical entries for TDG in the preceding section).

Order principle. For each node v on dimension i, the order of the daughters depends on their edge labels. We use this principle on the LP dimension to constrain the order of the words in a sentence. We can use it e.g. to require that determiners ("a") precede nouns ("woman").

Projectivity principle. Dimension i must be a projective graph. We use this principle on the LP dimension to ensure that LP trees do not have crossing edges.

Climbing principle. The climbing principle is two-dimensional, and allows us to constrain the relation between two dimensions. It stipulates that dimension i must be flatter than dimension j. We use it to state that the LP dimension (8) must be flatter than the ID dimension (7).

Linking principle. The linking principle relates two dimensions i and j, and in particular allows us to specify how semantic arguments must be realized in the syntax. It is lexicalized, and we use it to stipulate e.g. that the first argument (arg1) of "loved" on the PA dimension (10) must be realized by the deep subject (subjd), and the second argument (arg2) by the deep object (objd) on the DS dimension (9).

Contra-dominance principle. The contra-dominance is also two-dimensional. We use it to constrain the relation between the two semantic dimensions PA and SC, in particular to stipulate that the semantic arguments of verbal predicates (on the PA dimension) must dominate them on the SC dimension. For instance, the first semantic argument of "loved" on the PA dimension (in (10), this is the determiner "every" of the NP "every man") must dominate (be an ancestor of) "loved" on the SC dimension (12) and (13).

6.3 Parsing and generation

Parsing and generation with XDG grammars is done using constraint solving by the *XDG solver*. XDG solving has a natural reading as a constraint satisfaction problem (CSP) on finite sets of integers, where well-formed analyses correspond to the solutions of the CSP [7]. We have implemented the XDG solver with the Mozart/Oz programming system [14], [15].

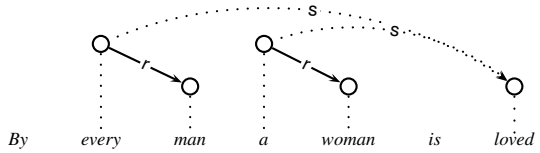
XDG solving operates on all dimensions concurrently. This means that the solver can infer information about one dimension from information on any other, e.g. by the multi-dimensional principles (climbing, linking, contra-dominance). For instance syntactic information can trigger inferences in semantics, and vice versa.

Because XDG allows us to write grammars with completely free word order, XDG solving is an NP-complete problem [6]. This means that the worst-case complexity of the solver is exponential at present. However, the behaviour of the solver on NL grammars is excellent in practice. Constraint propagation is both fast and very effective, and permits to enumerate solutions with few or no failures.

6.4 Underspecification

Similar to MRS and also CLLS, XDG supports the use of *underspecification*. An underspecified XDG analysis is a partial XDG dependency graph where not all of the edges are fully determined. We show an underspecified XDG analysis for the

SC dimension below:



(14)

In this analysis, the edges from "every" to "man" (labeled r) and from "a" to "woman" (also labeled r) are already determined, i.e. we know already that "man" is in the restriction of "every", and that "woman" is in the restriction of "a". However, the scopal relationship between the two quantifiers is yet unknown. Still, the XDG constraint solver has already inferred that both dominate the verb "loved" (as indicated by the dotted "dominance edge"). This partial analysis abstracts over both fully specified analyses (12) and (13) above.

Whereas in MRS and CLLS, only scopal relationships can be underspecified, XDG goes one step further and allows to underspecify *any* of its dimensions, i.e. not only the scopal but also the syntactic dimensions. This can be used e.g. to compactly represent PP-attachment ambiguities.

7 Conclusion

We proposed a notion of *lexicalized multi-dimensional configuration problems* as a metaphor and a practical constraint-based approach for a wide range of tasks in computational linguistics, including semantic assembly, surface realization and syntactic analysis, and how it can be used to integrate them. We then presented XDG as an instance of this approach, and showed how to use it to integratively handle syntax and semantics of natural language. We think that multi-dimensional lexicalized configuration problems can play an important role in the future, as an overarching framework for computational linguistics research, on the theoretical and on the algorithmic side. For instance, research on configuration problems for semantic assembly have already yielded highly efficient algorithms for satisfiability and enumeration of dominance constraints [16]. For surface realization, an efficient algorithm was presented in [6]. At the moment, we are working on making the integrated processing of syntax and semantics in XDG more efficient.

References

1. Mittal, S., Frayman, F.: Towards a generic model of configuration tasks. In: Proceedings of the International Joint Conference on Artificial Intelligence, Morgan Kaufmann (1989) 1395–1401
2. Bos, J.: Predicate logic unplugged. In: Proceedings of the 10th Amsterdam Colloquium. (1996) 133–143

3. Copestake, A., Flickinger, D., Pollard, C., Sag, I.: Minimal recursion semantics. an introduction. Journal of Language and Computation (2004) To appear.
4. Egg, M., Koller, A., Niehren, J.: The constraint language for lambda structures. Journal of Logic, Language, and Information (2001)
5. Abeillé, A., Rambow, O.: Tree Adjoining Grammar: An Overview. In: Tree Adjoining Grammars: Formalisms, Linguistic Analyses and Processing. CSLI Publications (2000)
6. Koller, A., Striegnitz, K.: Generation as dependency parsing. In: Proceedings of ACL 2002, Philadelphia/USA (2002)
7. Duchier, D.: Configuration of labeled trees under lexicalized constraints and principles. Research on Language and Computation **1** (2003) 307–336
8. Duchier, D., Debusmann, R.: Topological dependency trees: A constraint-based account of linear precedence. In: Proceedings of ACL 2001, Toulouse/FRA (2001)
9. Bresnan, J., Kaplan, R.: Lexical-functional grammar: A formal system for grammatical representation. In Bresnan, J., ed.: The Mental Representation of Grammatical Relations. The MIT Press, Cambridge/USA (1982) 173–281
10. Mel'čuk, I.: Dependency Syntax: Theory and Practice. State Univ. Press of New York, Albany/USA (1988)
11. Pollard, C., Sag, I.A.: Head-Driven Phrase Structure Grammar. University of Chicago Press, Chicago/USA (1994)
12. Debusmann, R., Duchier, D., Koller, A., Kuhlmann, M., Smolka, G., Thater, S.: A relational syntax-semantics interface based on dependency grammar (2004)
13. Trujillo, I.A.: Lexicalist Machine Translation of Spatial Prepositions. PhD thesis, University of Cambridge, Cambridge/USA (1995)
14. Smolka, G.: The Oz Programming Model. In van Leeuwen, J., ed.: Computer Science Today. Lecture Notes in Computer Science, vol. 1000. Springer-Verlag, Berlin (1995) 324–343
15. Mozart Consortium: The Mozart-Oz website (2004) http://www.mozart-oz.org/.
16. Fuchss, R., Koller, A., Niehren, J., Thater, S.: Minimal recursion semantics as dominance constraints: Translation, evaluation, and analysis. In: Proceedings of ACL 2004, Barcelona/ESP (2004)

An intuitive tool for constraint based grammars

Mathieu Estratat and Laurent Henocque

Université d'Aix-Marseille III,
Laboratoire des Sciences de l'Information et des Systèmes
Avenue Escadrille Normandie Niemen
13397 Marseille cedex 20, France
{mathieu.estratat, laurent.henocque}@lsis.org

Abstract. A lot of recent linguistic theories are feature-based and heavely rely upon the concept of constraint. Several authors have pointed out the similitude existing between the representation of the features on feature-based theories and the notions of objects or frames. Oriented object configuration allows us to deal with these modern grammars. We propose here a systematic translation of the concepts and constraints introduced by two linguistic formalisms : the very useful *HPSG* and the recent *property grammars* to configuration problems representing specific target languages. We assess the usefulness of these translations by studying first a part of the grammar for english proposed by the HPSG's authors then a natural language subset with lexical ambiguities, using property grammars.

1 Introduction

A lot of recent linguistic theories are feature-based and heavely rely upon the concept of constraint. Several authors have pointed out the similitude existing between the representation of the features on feature-based theories and the notions of objects or frames. Oriented object configuration[8] allows us to deal with these modern grammars. Configuration task consists in building (a simulation of) a complex product from components picked from a catalog of types. Neither the number nor the actual types of the required components are known beforehand. Components are subject to relations (this information is called "partonomic"), and their types are subject to inheritance (this is the taxonomic information). Constraints (also called well formedness rules) generically define all the valid products. A configurator expects as input a fragment of a target object structure, and expands it to a solution of the problem constraints, if any. This problem is undecidable in the general case. Such a program is well described using an object model (as illustrated by the figures 8 and 10), together with well formedness rules. Technically solving the associated enumeration problem can be made using various formalisms or technical approaches : extensions of the CSP paradigm [9, 5], knowledge based approaches [14], terminological logics [10], logic programming (using forward or backward chaining, and non standard semantics) [13], object-oriented approaches [8, 14]. Our experimentations were conducted using the object-oriented configurator Ilog JConfigurator [8].

More precisely, a configurator is a constraint-based solver. Technically, its constituents are a catalog of types of components and a set of constraints over this components. An oriented oject configurator like Ilog JConfigurator represents its catalog of types through an object model, containing classes, attributes and relations between classes. These relations can be inheritance, composition or also aggregate relations. The configurator's user has to represent a generic (object) model of the knowledge field he wants to implement. This model is not usually sufficient to represent all the relations between elements. For instance, to represent a PC, we have to state the model described in figure 1. This model represent one (or more) PC and its (theirs) components. We can see that a *PC* must have one and only one *Motherboard*[1], one and only one *Supply*, one and only one *Monitor*. But it can have to one to four *Disk(s)*[2]. The *Motherboard* can have one or two *Processor(s)* and one to four *Memory(ies)*. Any PC can be represented by an instance of this model. Some attributes need

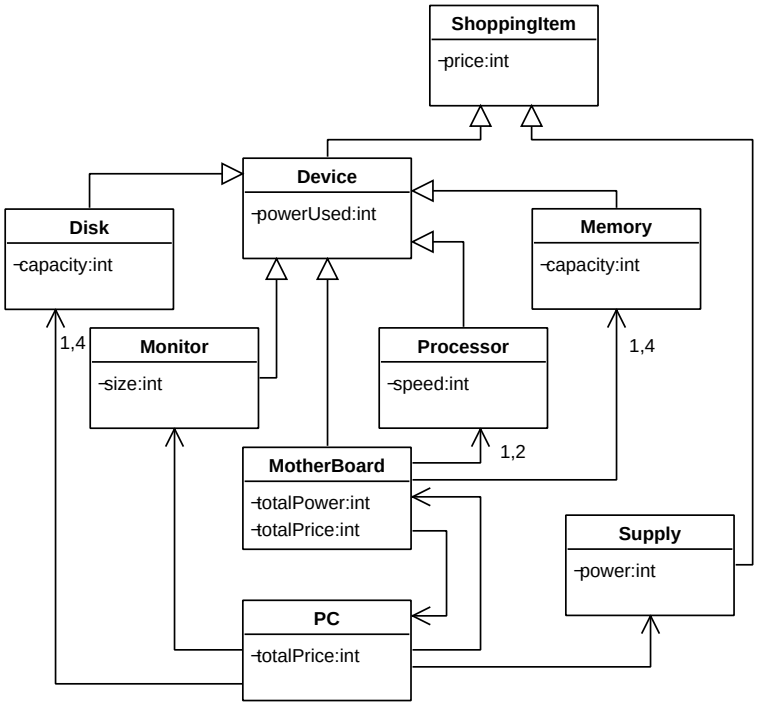


Fig. 1. A generic object model for a PC configuration

[1] Simple arrows with no label represent a relation of cardinality one
[2] Simple arrows with label X, Y represent a relation of cardinality X to Y

some other constraints to be instanciated correctly. For sample, the attribute *totalPrice* of an instance of the class *PC* is calculated with the constraint : *PC.totalPrice = SUM (MotherBoard.totalPrice, Supply.price, Monitor.price, Disk_1.price, Disk_2.price, Disk_3.price, Disk_4.price)*[3]

The configurator JConfigurator is implemented in Java. User draws the object model with a graphic tool incorporated and states constraints using an other incorporated tool or using a java program.

We want to show that a configurator can deal with some other constraint-based grammars. Hence, we propose a systematic translation of the concepts and constraints introduced by two linguistic formalisms : the very useful *HPSG*[11] and the recent *property grammars*[1] to configuration problems representing specific target languages. We assess the usefulness of these translations by studying first a part of the grammar for english proposed by the HPSG's authors then a natural language subset with lexical ambiguities, using property grammars. We have already presented the traduction of property grammars into configuration problem in [3, 4]. Section 2 describes a mapping from feature structures to object model. Section 3 presents the translation of a grammar based on HPSG into a configuration problem. Section 4 presents a mapping from a property grammar to a configuration problem. Section 5 shows an application of the previous translation to a subset of the french grammar proposed in [1] with ambiguous words. Section 6 concludes and presents ongoing and future research.

2 Feature structures as object model

A lot of recent theories are based on feature structures to represent constituents of the grammar and informations over syntax, semantic, phonology, and others. A feature structure is a set of $(attribute, value)$ pairs used to label a linguistic unit, as illustrated in figure 2(2), where *son* is a *masculine noun*, at the *singular, 3^{rd} pers.* This definition is recursive : a feature value can be another feature structure, or a set of features. Feature structures are often used in several linguistic theories. For example, GPSG[6], HPSG[11], Property grammars[1], Dependency Grammars[2] use feature structures. Several authors have pointed out the similitude existing between the representation of the features on feature-based theories and the notions of objects or frames. We will see how such structures can be represented with oriented object paradigm over the Unified Modeling Language(UML)[7].

Functionally, a feature can be mapped to a CSP variable, and a feature structure can be seen as a binding of values to an aggregate of feature variables. A feature value can be a constant from a specific domain (for instance an enumeration as $\{Singular, Plural\}$, or an integer as $\{1(st), 2(nd), 3(rd)\}$). A feature value can also be a (set of, list of) feature structure(s) (as *Agreement* in figure

[3] The usually dot notation is used to access to attributes of a class and also class linked to an other class. Disk_[1,4] represent the possible instance of the class Disk in relation with the class PC. Each of them could be present or not (only one is mandatory) and if others are not present the default value 0 is set

$$\begin{bmatrix} \textit{Cat:} & \text{N} \\ \textit{Phon:} & \text{list(String)} \\ \textit{Agreement:} & \begin{bmatrix} \textit{Gen :} & \{\text{masc,fem,neutral}\} \\ \textit{Num :} & \{\text{sing,plur}\} \\ \textit{Per :} & \{\text{1st,2nd,3rd }\} \end{bmatrix} \\ \textit{Case :} & \{\text{Common, proper}\} \end{bmatrix}$$

(1) - general feature structure for a noun.

$$\begin{bmatrix} \textit{Cat:} & \text{N} \\ \textit{Phon:} & \text{son} \\ \textit{Agreement:} & \begin{bmatrix} \textit{Gen :} & \text{masc} \\ \textit{Num :} & \text{sing} \\ \textit{Per :} & \text{3rd} \end{bmatrix} \\ \textit{Case :} & \text{Common} \end{bmatrix}$$

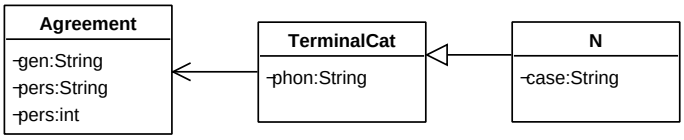
(2) - instance of (1) for the noun *son*.

Fig. 2. General feature structure for a noun and one of its instances (*son* as example)

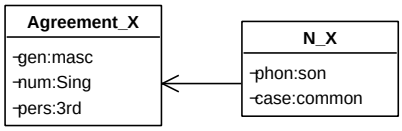
2). Hence standard finite domain CSP variables cannot be used to model features, and a notion of relations, or set variables must be used (as in [8, 14, 2]). It is worth pointing that feature structures are available as a language construct in Oz [12] and support feature constraints. Feature structures are aggregates well modeled using *classes* in an object model. A feature structure, following the studied formalism, naturally maps to a *class* in an object model, inserted in a class hierarchy involving *inheritance* (possibly multiple [11]). For instance, the category in figure 2 is translated into a class in an object model, as illustrated by figure 3. In this translation, the feature structure is represented by the class N . This class inherits from the class *TerminalCat*, representing all terminal categories. So N inherits the attribute *phon* (representing the lexical entry) from *TerminalCat* and the relation with class *Agreement* (representing the feature *agreement* in figure 2, each of its attributes represents one of the features of the composed feature agreement). For the noun *son*, the instance of the model represent perfectly this feature structure. We have seen that feature structures may be represented with an object model (which itself can be represented using an UML diagram). This UML diagram maps straightforwardly to the object model used in JConfigurator.

3 The HPSG structure and principles as a configuration problem

Feature structures are named *Signs* in HPSG. These signs map to all syntactic units as words or sentences. The *Principles* state the available values for some attributes of some coexistent signs.



(1) - generic object model representing feature structure of figure 2.



(2) - instance of (1).

Fig. 3. A generic object model for the category N and an instance of it

3.1 The signs

Signs are implemented with classes on the object model where composition and inheritance relations are explicits. Some signs are non-specific to linguistic as *list* and *set* (see figure 4 for sort hierarchy of these feature structures). Lists are

Partition of $list(\sigma)$: $nonempty\text{-}list(\sigma)$ ($nelist(\sigma)$), $empty\text{-}list$ ($elist$ or $\langle\ \rangle$)
Partition of $set(\sigma)$: $nonempty\text{-}set(\sigma)$ ($neset(\sigma)$), $empty\text{-}set$ ($eset$ or $|\ |$)

Fig. 4. Part of sort hierarchy presented in [11] used to represent list and set relations

translated using a specific class named *list* which contains an attribute *FIRST* (which has the *object* type) and is linked to itself with the relation REST of cardinality [0,1]. This cardinality means that an element of this list can have only one following element at most. The figure 5 presents the object model associated to this idea. The order of the elements remains the same. It is necessary to set a constraint specifying that all elements of a same list are of the same type. So we set the constraint as following:

$$\forall l \in list,\ l.REST.FIRST.sort = l.FIRST.sort$$

The empty list is represented using the value 0 for the cardinality of the relation. Sets are translated by specifying on the object model that this relation is of sort set. This relation is presented in figure 9, the star over the relation specifies that this is a set relation. The feature QSTORE is used in the grammar for

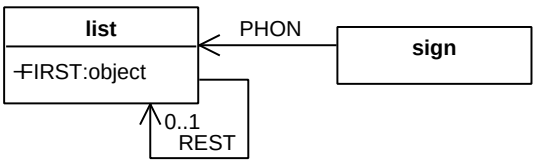


Fig. 5. Translation of list

english as we will see below. An empty set is represented using the value 0 for the cardinality of the relation . Figure 6 represents some feature structures declarations of the english grammar proposed in [11] while figure 7 represents some of its sort hierarchy. Informations contained in this two figures, allows us to define the object model presented in figure 8. Note that these elements are subparts of the grammar, limited in space for readibility. The grammar is too expensive to be represented totaly here. Nevertheless these elements permit us to explain how to translate elements of the HPSG theory on an object model. The sort hierarchy is translated with the inheritance relations on the object model. As example we can see in figure 7 that sign is partitioned into 2 sorts : *phrase* and *word*, in figure 8 a inheritance relation exists between classes *word* and *phrase* on one hand and *sign* on the other hand. The translation of the feature *PHON* of the feature structure sign is shown on figure 5. On it we see that the class *sign* is linked to the class *list* with the relation *PHON*. This relation allows an instance of the class *sign* to be linked to the first element of the list and to access easily to the following elements of this list using the relation *REST* of the class *list*. In order to specify the type of the elements in this list, we have to set the following constraint :

$$\forall s \in sign,\ s.PHON.FIRST.sort = string$$

For each feature which value is of list sort, a similary constraint, setting the sort of the list elements, has to be set. The translation of the feature *SYNSEM* of the feature structure sign is implemented directly using a relation on the object model (figure 6). This relation links the classes sign and synsem. The *QSTORE* feature has a set of *quantifier* as value. As seen previously, this relation of sort set is direcly translated into a relation on the model without using dedicated feature structure as in [11]. In the HPSG theory we also need a set of constraints, named principles, specifying the allowed arrangements of the features values. We are now presenting some of this principles and their translations.

3.2 Principles

A *headed phrase*, following the authors notations, is a phrase whose DAUGTHERS value is of sort *headed-struc*. As a start, we present a translation of the

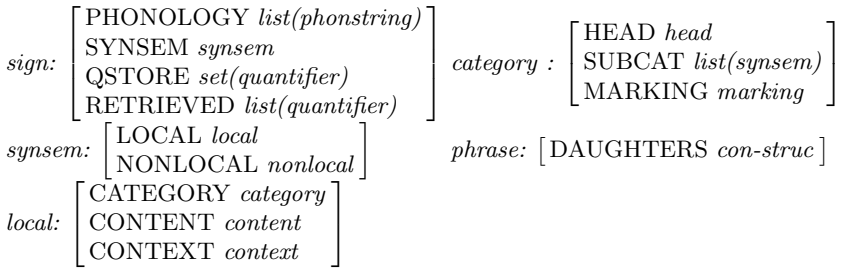


Fig. 6. Some feature structures presented in [11]

Partition of *sign*: *word*, *phrase*
Partition of *con-struc*: *headed-structure*(*head-struc*), *coordinate-structure*(*coord-struc*)

Fig. 7. Part of sort hierarchy presented in [11]

Head Feature Principle.
The Head Feature Principle [4] :
In a headed phrase (phrase whose DAUGHTERS value is of sort *headed-structure*), the value of SYNSEM|LOCAL|CATEGORY|HEAD and DAUGHTERS|HEAD-DTR|SYNSEM|LOCAL|CATEGORY|HEAD are *token-identical*.
Along with the partial object model presented in figure 8, we have to state the constraint, using the usually doted notation :

$$\forall p \in phrase\ p.SYNSEM.LOCAL.CATEGORY.HEAD =$$

$$p.DAUGHTERS.HEAD - DTR.SYNSEM.LOCAL.CATEGORY.HEAD$$

This constraint specifies that if two feature values are equals then only one instance of the class is generated. In fact, if the two values are equals we only need to generate one instance of the class and link it to the available instances. Some principles force a feature value to be a union of some other feature values (non necessary, and generally it is not the case, of the same feature structure). As example the *SUBCATEGORIZATION Principle*[5]. These principles are translated using a constraint of union, implemented to make set with other sets. As we have seen, the configuration paradigm seems to be really useful to implement a heavily linguistic theory as HPSG. We are now going to see how we can also translate a more recent theory, *the property grammars*[1], heavily relying upon constraints.

[4] State in [11] p.34
[5] State in [11] p.34 : In a headed phrase, the list value of DAUGHTERS|HEAD-DAUGHTER|SYNSEM|LOCAL|CATEGORY|SUBCAT is the concatenation of the list value of SYNSEM|LOCAL|CATEGORY|SUBCAT with the list consisting of the SYNSEM values (in order) of the elements of the list value of DAUGHTER|COMPLEMENT-DAUGTHER.

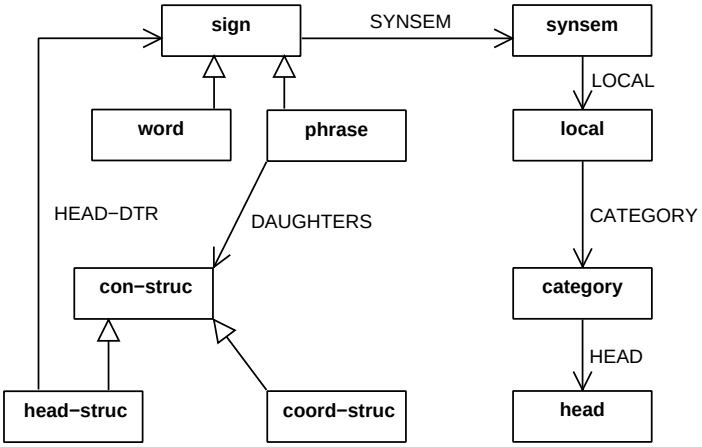


Fig. 8. A partial object model for an english grammar

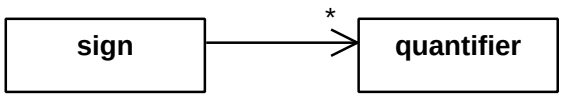


Fig. 9. Translation on the object model of a feature value of sort set

4 Property grammars

4.1 An other feature based theory

Property grammars, as the author says, heavily rely upon constraints. The syntactic units are represented with features structures named *categories*. These categories are translated into an object model. For each phrase like NP or VP, some well formedness rules are stated. This rules or constraints are called *properties* and are translated directly into the object model (using relations cardinalities) or through added constraints to the model. As it has been done for the HPSG paradigm in section 3.2.

4.2 Object model constraints for properties

Properties define both object model relations and constraints, adjoined to the object model built from a given property grammar. We use uppercase symbols to denote categories (e.g. $S, A, B, C \ldots$). We also use the following notations : when an anonymous relation exists between two categories S and A, we denote

as $s.A$ the set of As linked to a given S instance s, and as $|s.A|$ their number. For simplicity, and wherever possible, we will use the notation $\forall SF(S)$ (where F is a formula involving the symbol S) rather than $\forall s \in SF(s)$. Class attributes are denoted using standard dotted notation (as e.g $a.begin$ that represents the *begin* attribute for object a). I_A denotes the set of all available indexes for the category A.

- **Constituents** :
  $Const(S) = \{A_m\}_{m \in I_A}$ specifies that an S may only contain elements from $\{A_m\}$. This property is described by using relations between S and all $\{A_m\}$, as shown in the object models presented in figure 10.

- **Heads** :
  The $Heads(S) = \{A_m\}_{m \in I_A}$ property lists the possible heads of the category S. The head's element is unique, and mandatory. For example, $Heads(NP) = \{N, Adj\}$. The word "door" is the *head* in the NP : "the door". The $Head$ relation is a subset of $Const$. Such properties are implemented using relations as for constituency, plus adequate cardinality constraints.

- **Unicity** :
  The property $Unic(S) = \{A_m\}_{m \in I_A}$ specifies that an instance of the category S can have at most one instance of each $A_m, m \in I_A$ as a constituent. *Unicity* can be accounted for using cardinality constraints as e.g : $\forall S|\{x : S.Const \mid x \in A_m\}| \leq 1$ which for simplicity in the sequel, we shall note $|S.A_m| \leq 1$. For instance, in an NP, the determiner Det is unique.

- **Requirement** :
  $\{A_m\}_{m \in I_A} \Rightarrow_S \{\{B_n\}_{n \in I_B}, \{C_o\}_{o \in I_C}\}$ means that any occurrence of all A_m implies that all the categories of either $\{B_n\}$ or $\{C_o\}$ are represented as constituents. As an example, in a noun phrase, if a common name is present, then so must a determiner ("door" does not form a valid noun phrase, whereas "the door" does). This property maps to the constraint

$$\forall S(\forall m \in I_A \ |S.A_m| \geq 1) \Rightarrow$$
$$((\forall n \in I_B \ |S.B_n| \geq 1) \vee (\forall o \in I_C \ |S.C_o| \geq 1))$$

- **Exclusion** :
  The property $\{A_m\}_{m \in I_A} \nLeftrightarrow \{B_n\}_{n \in I_B}$ declares that two category groups mutually exclude each other, which can be implemented by the constraint :

$$\forall S, \begin{cases} (\forall m \in I_A \ |S.A_m| \geq 1) \Rightarrow (\forall n \in I_B \ |S.B_n| = 0) \\ \wedge \\ (\forall n \in I_B \ |S.B_n| \geq 1) \Rightarrow (\forall m \in I_A \ |S.A_m| = 0) \end{cases}$$

  For example, a N and a Pro can't cooccur in a NP. (Note that in the formulation of these constraints, $\Rightarrow$ denotes logical implication, and not the

requirement property.)

– **Linearity** :
The property $\{A_m\}_{m\in I_A} \prec_S \{B_n\}_{n\in I_B}$ specifies that any occurrence of an $\{A_m\}_{m\in I_A}$ precedes any occurrence of an $\{B_n\}_{n\in I_B}$. For example, in a NP a Det must precede an N (if present). Implementing this property induces the insertion in the representation of categories in the object model of two integer attributes *begin* and *end* that respectively denote the position of the first and last word in the category. This property is translated as the constraint:

$$\forall S\ \forall m \in I_A\ \forall n \in I_B,$$
$$max(\{i \in S.A_m \bullet i.end\}) \leq min(\{i \in S.B_n \bullet i.begin\})$$

– **Dependency** :
This property states specific relations between distant categories, in relation with text semantics (so as to denote for instance the link existing between a pronoun and its referent in a previous sentence). For instance, in a verb phrase, there is a dependency between the subject noun phrase and the verb. This property is adequately modeled using a relation.

Properties can therefore be translated as independent constraints. It is however often possible to factor several properties within a single modeling construct, most often a relation and its multiplicity. For instance, constituency and unicity can be grouped together in some models where one relation is used for each possible constituent (we made this choice in the forthcoming example, in figure 10).

5 Parsing a lexically ambiguous natural language

Figure 10 represents a fragment of the constrained object model for a subset of french, where constituency and unicity are made explicit. The figure 11 illustrates some well formedness constraints. In the figure 12 we define a small example lexicon.
The language accepted by this constrained object model is made of phrases constructed around a subject, a verb and a complement, where both the subject and the verb are mandatory, and both the subject and the complement are noun phrases.
Both constraints are stated straightforwardly within the object model via relations and their cardinalities (as can be seen in the figure 11). However, more constraints are required, like the constraints stating that a Head is a constituent, or the constraints ruling the value of the *begin* and *end* attributes in syntagms.

The program runs as it : Its root object (the object from which the research is launched) is state to be an instance of *Sentence*. From this root, the solver tries to complete the relations and attributes for each component (in respect

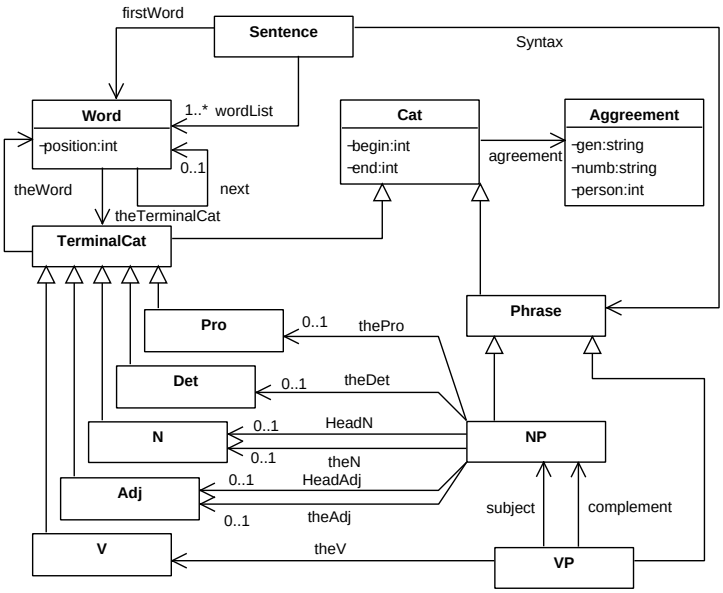


Fig. 10. Object model used to parse our language

with the adjunct constraints). Moreover, for each word it generates the number of instances of *terminalCat* associated to this word. For instance, for the french determiner/pronoun "La", the program will create an object *word* named *La*, and two *terminalCat*, *LaDet* and *LaPro*. Each *terminalCat* is linked to a word, for example, *LaDet* will be linked to the word *La*.

5.1 Experimental results

We tested the object model with sentences involving variable levels of lexical ambiguity, as from the lexicon listed in figure 12.
Sentence (1), "la porte ferme mal" (*the door doesn't close well*) is fully ambiguous. In this example, "la" can be a *pronoun* (i.e. *it* as in "give **it** to me !") or

$Head : |NP.headN| + |NP.headAdj| = 1;$
$Linearity : Det < N;\ Det < Adj;$
$Exclusion : (|NP.N| >= 1) \Rightarrow (|NP.Pro| = 0)$ and
$(|NP.Pro >= 1) \Rightarrow (|NP.N| = 0);$
$Requirement : (|NP.N| = 1) \Rightarrow (|NP.det| = 1)$

Fig. 11. Some NP constraints

WORD	CAT	GEN	NUM	PERS
ferme	N	fem	sing	3
ferme	Adj	-	sing	-
ferme	V	-	sing	1,3
la	Det	fem	sing	3
la	Pro	fem	sing	3
mal	N	masc	sing	3
mal	Adj	-	-	-
porte	N	fem	sing	3
porte	V	-	sing	1,3

Fig. 12. A lexicon fragment

a *determiner*(like *the* in "**the** door"), "porte" can be a *verb* (i.e. *to carry*) or a noun (*door*), "ferme" can be a *verb* (*to close*), a *noun* (*farm*) or an adjective (*firm*) and "mal" can be an *adjective* (*badly*) or a *noun* (*pain*). Our program produces a labeling for each word and the corresponding syntax tree (figure 13).

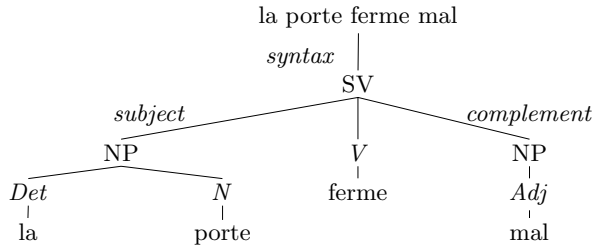


Fig. 13. Syntax tree for the french sentence "la porte ferme mal"

Sentence (2) is "la porte bleue possède trois vitres jaunes" (*the blue door has three yellow windows*). Here "bleue" and "jaunes" are *adjectives*, "vitres" is a *noun* and "trois" is a *determiner*. The last sentence (3), "le moniteur est bleu" (*the monitor is blue*) involves no ambiguous word. The table 1 presents results obtained for the sentences (1), (2) and (3). These results show that the

Table 1. Experimental results for french phrases

p	$\#fails$	$\#cp$	$\#csts$	$\#vars$	$\#secs$
(1)	4	40	399	220	0.55 s
(2)	3	50	442	238	0.56 s
(3)	1	35	357	194	0.52 s

correct labeling is obtained after very few backtracks ($fails$). The number of fails depends on the number of ambiguous words in the sentence. The execution time also depends on the sentence length. Explanation of these three samples :

- (1) : The solver badly associates the *Verb* of the sentence : it states *porte* as being the verb. Hence it cannot create a complement to this verb because "ferme mal" is not a *NP*, and the program backtracks on the instanciation $SV.comp = SN1$. As there is not anymore available instanciation, it backtracks on the instanciation $porteV.theWord = porte$ and also backtracks on the instanciation $SV.theV = porteV$. At this time, the verb is now stated to $fermeV$ and the program goes on and instantiates all possible variables. The solver has as heuristic to try to state the more little value first. As a consequence it first states the value 0 for the cardinality of the relation $SN.noyauAdj$, but this leads to a fail so it backtracks again and changes the value to 1. The remaining variables are instantiated with no more fail.
- (2) : The three fails, as for (1), are dued to the instanciation of $SV.theV$ to $porteV$. Like in (1) three backtracks are needed.
- (3) : The remaining value is due to the minimalisation inherents heuristic. The solver first tries to set the value of the cardinality of the relation $SN.noyauAdj$ to 0.

We have not already implemented an analyser with sufficient efficiency to compare our results to those of other approches. We have only shown a translation of linguistic theories to a configuration problem. This work was the first part of a most important project to deal with syntax and semantic on a same object model.

6 Conclusion

We showed that data structures and inner mechanism (the principles) of a useful linguistic theory as HPSG can be translated into a configuration problem, solved with a general solver. We also showed that an other linguistic theory can be translated into a configuration problem and we presented (an embryo of) a parser, which is available to disambiguate sentences that can be used over it. Other grammars could be translated into a configuration problem. For example the *dependency grammars*[2] is clearly described by the author as a configuration problem. Our intuition leads us to think that where there is feature structures there is object model and where there is constraints on these feature structures, there is a configuration problem. A configurator seems to be an intuitive tool for linguistic theories based on constraints. First step of our work was to represent grammar concepts in term of a configuration problem. The second will be to upgrade this simple parser into a more general one, and by this way, using configurators to deal with syntax and semantic on a same object model, as we have seen that configurators are used to deal with semantic of some knowledge field. They are used to represent a world which can be describe with a description language (PCs can be represented as a configuration problem and we can describe

PCs with description language). Ongoing research involves the implementation of a parser for a natural language subset of french dealing with the semantics of three dimensional scene descriptions.

References

1. P. Blache, *Les Grammaires de Propriétés : des contraintes pour le traitement automatique des langues naturelles*, Hermès Sciences, 2001.
2. Denys Duchier, 'Axiomatizing dependency parsing using set constraints', in *Sixth Meeting on Mathematics of Language, Orlando, Florida*, pp. 115–126, (1999).
3. Mathieu Estratat and Laurent Henocque, 'Application des programmes de contraintes orientés objet à l'analyse du langage naturel', *Traitement Automatique du Langage Naturel, TALN 2004*, 163–172, (2004).
4. Mathieu Estratat and Laurent Henocque, 'Parsing languages with a configurator', *European Conference on Artificial Intelligence, ECAI 2004*, (2004). To appear.
5. G. Fleischanderl, G. Friedrich, A. Haselbck, H. Schreiner, and M. Stumptner, 'Configuring large-scale systems with generative constraint satisfaction', *IEEE Intelligent Systems - Special issue on Configuration*, **13(7)**, (1998).
6. G. Gazdar, E. Klein, G.K. Pullum, and I.A. Sag, *Generalized Phrase Structure Grammar*, Blackwell, Oxford, 1985.
7. Object Management Group, 'Uml v. 1.5 specification', *OMG*, (2003).
8. D. Mailharro, 'A classification and constraint based framework for configuration', *AI-EDAM : Special issue on Configuration*, **12(4)**, 383 – 397, (1998).
9. Sanjay Mittal and Brian Falkenhainer, 'Dynamic constraint satisfaction problems', in *Proceedings of AAAI-90*, pp. 25–32, Boston, MA, (1990).
10. B. Nebel, 'Reasoning and revision in hybrid representation systems', *Lecture Notes in Artificial Intelligence*, **422**, (1990).
11. C. Pollard and I.A. Sag, *Head-Driven Phrase Structure Grammar*, The University of Chicago Press, Chicago, 1994.
12. Gert Smolka and Ralf Treinen, 'Records for logic programming', *The Journal of Logic Programming*, **18**(3), 229–258, (April 1994).
13. Timo Soininen, Ilkka Niemela, Juha Tiihonen, and Reijo Sulonen, 'Representing configuration knowledge with weight constraint rules', in *Proceedings of the AAAI Spring Symp. on Answer Set Programming: Towards Efficient and Scalable Knowledge*, pp. 195–201, (March 2001).
14. Markus Stumptner, 'An overview of knowledge-based configuration', *AI Communications*, **10(2)**, 111–125, (June 1997).

A broad-coverage parser for German based on defeasible constraints

Kilian A. Foth, Michael Daum, and Wolfgang Menzel

Natural Language Systems Group, University of Hamburg

Abstract. We present a parser for German that achieves a competitive accuracy on unrestricted input while maintaining a coverage of 100%. By writing well-formedness rules as declarative, defeasible constraints that integrate different sources of linguistic knowledge, very high robustness is achieved against all sorts of language error.

1 Introduction

Although most linguists agree that natural language to a large degree follows well-defined rules, it has proven exceedingly difficult to actually define these rules well enough that a computer could reliably assign the intuitively correct structure to natural language input. Therefore, almost all current approaches to parsing constitute compromises of some kind:

1. Often the goal of full syntactic analysis is abandoned in favour of various kinds of *shallow parsing*, which is all that is needed for many applications. Instead of a full syntax tree, e.g., only the boundaries of major constituents [Schmid and Schulte im Walde2000] or topological fields [Becker and Frank2002] are computed.
2. Instead of casting the language description into linguistically motivated rules, a probability model is induced automatically from a large amount of past utterances, which is then used to maximize the similarity to previously seen structures [Collins1999]. This approach is robust and efficient, but relies on a large amount of existing data, and the resulting model is difficult to comprehend and extend.
3. Formalisms that do perform deep analysis by following explicit rules typically often have to be restricted to particular subsets of linguistic constructions or to particular domains. Also, their robustness and coverage (the performance for ungrammatical and extragrammatical input, respectively) is often rather low on realistic data.

We present a parsing system that tries to avoid all three compromises to the extent possible at the current time. Instead of a derivation gram-

mar, we employ a declarative formalism in which well-formedness conditions are expressed as explicit constraints (which take the form of logical formulas) on word-to-word dependency structures. Since there is no limit to the complexity of these formulas, *every* conceivable well-formedness condition can be expressed as a constraint. Although the formal power of this approach would theoretically make the parsing problem intractable, we have found that approximative solution methods yield good results in practice.

All grammar rules are ordered by a preference measure that distinguishes e.g. less important rules of style from more important fundamental grammar rules. This not only ensures robust behaviour against all kinds of deviant input, but also allows us to integrate the information from external shallow parsers, such as a part-of-speech tagger, without becoming dependent on their results.

The output of the system consists of labelled dependency trees rather than constituent-based structures. Although this is mainly a consequence of casting the parsing problem in the form of a constraint optimization problem, it also has benefits for the processing of languages like German with a relatively free word order.

2 The WCDG parsing system

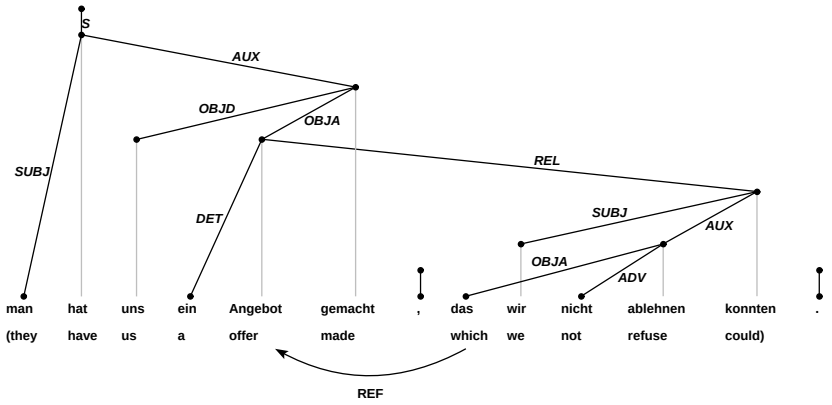


Fig. 1. Dependency analysis of a German sentence.

Weighted constraint dependency grammar (WCDG) [Schröder2002] is an extension of the CDG formalism first described by [Maruyama1990].

It describes the structure of natural language as a set of labelled subordinations: each word is either subordinated to exactly one other word or considered the root of the syntax tree (also called a NIL subordination). See Figure 1 for an example. Each subordination is annotated with one of a fixed set of labels such as 'subject', 'direct object' or 'indirect object', but no larger groups of words carry labels. This means that there is no direct equivalent to the constituents postulated by phrase structure grammar.

Since there are no constituents, there are also no generative rules along the lines of 'S → NP VP'; these are often problematic for languages with free or semi-free word order since they mingle dominance principles with linear precedence rules. Instead, the grammar rules take the form of declarative constraints about permissible subordinations. These constraints can reference the position, reading and lexical features of the concerned word forms, as well as features of neighbouring dependency edges. Every subordination that is not forbidden by any constraint is considered valid. The goal of the parser is to select a set of subordinations that satisfies all constraints of the grammar.[1]

As an example of a constraint, consider the rule that normal nouns of German require a determiner of some kind, either an article or a nominal modifier, unless they are mass nouns. This rule can be formulated as a constraint as follows:[2]

```
{X:SYN} : 'missing determiner' : 0.2 :
  X@cat = NN
  ->
  exists(X@mass_noun) |
  has(X@id, DET) |
  has(X@id, GMOD);
```

It states that for each subordination on the syntax level (`SYN`), a word with the category 'normal noun' (`NN`) must either bear the feature 'mass noun' or be modified by a determiner (label `DET`) or a genitive modifier (label `GMOD`).

Each constraint bears a score between `0.0` and `1.0` that indicates its importance relative to other constraints. The acceptability of an analysis is defined as the product of the scores of all instances of constraints

[1] Categorial and morphological ambiguity must also be resolved for each word in a sentence. While often glossed over, this task is quite difficult in languages with a rich morphology; the average German adjective form has 10 morphosyntactic variants that are relevant for agreement.

[2] This constraint is considerably simplified for exposition purposes, since the rule actually has many systematic exceptions.

that it violates. This means that constraints with a score of `0.0` must be satisfied if at all possible, since violating them would yield the worst possible score.[3] Note that the score of the determiner constraint is `0.2`, which means that missing determiners are considered wrong but not totally unacceptable. In fact, many other constraints are considered more important than the determiner rule.

The selection of real-valued weights for all constraints of a grammar is a difficult problem because of the virtually unlimited number of possibilities. A score for a new-written constraint can be calculated by counting how often it holds and fails on an annotated corpus, or by measuring the overall performance when different weights are assigned to it. (This method usually shows very little variation; better results are achieved when only sentences that actually contain the described phenomenon are analysed.) In general the exact score of a rule is less important than the relative score of two rules that might interact [Foth2004].

By assigning values other than `0.0` to all rules that might conceivably be broken, a prescriptive grammar can easily be converted to one that is robust against all kinds of language errors, while still preferring conforming over deviant structures wherever possible. In fact, a coverage of 100% can be guaranteed with an appropriately written grammar. This is achieved by giving those constraints which could forbid *all* structures for a given utterance a higher value.

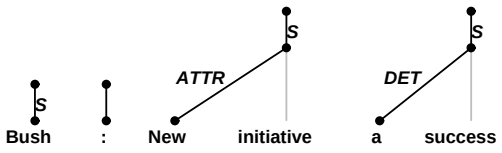


Fig. 2. Analysis of an elliptical utterance.

One easy way of ensuring this property is to allow surplus words to form the roots of isolated subtrees. This is possible because WCDG by itself does not enforce a unique NIL subordination. Note that in some cases a set of fragments actually is the most faithful rendition that is possible in a dependency formalism. Consider the typical elliptical headline *"Bush: New initiative a success"*, where the verbs "said" and "is" have

[3] However, the parser will still return such a structure if no analysis with a positive score can be found.

been elided. This should actually be modelled as a forest of three tree fragments (cf. Figure 2). Assigning a heavy penalty to fragments that are not finite verbs ensures that they will be avoided unless absolutely necessary. *Partial parsing* can thus be achieved simply by not constraining the number of NIL subordinations.

Since there are no generative rules, WCDG does not have a derivation component. Instead, parsing a sentence defines a multidimensional optimization problem that can be solved by all known methods of constraint optimization. Where the problem instance is too large to be solved via complete search, good success has been achieved with transformation-based heuristic solution methods [Daum and Menzel2002] [Foth and Menzel2000] that approximate the acceptability model defined by the constraints. In a large-scale grammar with many defeasible constraints, different analyses of the same sentence usually carry at least slightly different scores, so that full disambiguation is achieved. If different analyses happen to have the same score, the one that the transformation process finds first is usually selected as the output of the parser. By default, analysis terminates when it has repaired, or failed to repair, all important constraint violations (those whose score lies below a configurable threshold).

3 A comprehensive grammar of German

We have developed a grammar for the WCDG formalism that is intended to cover the entire realm of modern German. As an example of our dependency representation, consider again the analysis of the sentence *"Man hat uns ein Angebot gemacht, das wir nicht ablehnen konnten." (They made us an offer we could not refuse.)* in Figure 1. Note that there are two instances of *nonprojective* structure in this sentence, i.e. the vertical projection lines must cross dependency edges in two places no matter how the tree is drawn. WCDG does not require its structures to be projective, although constraints to that effect can easily be written and then have a considerable disambiguating effect. We can thus represent phenomena such as German auxiliary groups or extraposed relative sentences faithfully by writing constraints that enforce projectivity in general, but make exceptions for edges with these particular labels.

The referential relationship between the relative pronoun 'das' and its antecedent 'Angebot' cannot be represented on the syntax level, since both words already function as direct objects. Instead, it is represented by an additional edge which connects the two words on an extrasyntactical

'reference' level. The agreement of gender and number that must hold between the two words can thus be checked directly.[4]

The grammar currently consists of about 700 handwritten constraints, although much information is lexicalized, in particular valence information of verbs and prepositions. An extensive lexicon of full forms is used that includes all closed-class forms of German. Among the open-class forms, around 6,000 verb stems and 25,000 noun stems are covered; compound forms can be deduced from their base forms on the fly to deal with the various types of German compounds. As a last resort, lexicon templates provide skeleton entries for totally unknown words; these only contain the syntactic category and underspecified values for case, number etc. In the experiment reported here, 721 of 16649 tokens had to be hypothesized from such templates, such as personal names and foreign-language material.[5]

So far we have considered phenomena from about 28,000 sentences from various text types, such as newspaper text, appointment scheduling dialogues, classical literature, law texts, and online newscasts. Although obscure extragrammatical constructions are still occasionally encountered, the great majority of new input is covered accurately, and we do not expect major changes to the existing rules to become necessary. Many problem cases involve constructions that are probably impossible to represent accurately in a word-based dependency tree, such as complicated conjunctions or heavy ellipsis.

We estimate the overall effort for our grammar of German at about 5 work-years. A large part of this work, however, consisted in creating manual syntax annotations for testing the predictions of the constraint model; this corresponds to the effort needed to create the treebank that a stochastical parser is then trained on. Another very time-consuming task was to collect and classify open-class lexicon entries with respect e.g. to their inflection and valence features. While helpful for disambiguation in many cases, this information is not strictly necessary for running the parser.

[4] Although other pronouns, and possibly nouns, can refer to other words, these relationships are usually outside the scope of a sentence-based parser because they transcend sentence boundaries. Therefore, this grammar describes reference edges *only* for the relative pronouns.

[5] In theory, an unknown word could be of any open class, and thus introduce very great ambiguity; but most of these alternatives are usually discarded by the POS-tag preprocessing.

4 Evaluation

Exact solution of the optimization problem posed by a large constraint grammar is often computationally infeasible for long input sentences, so that incomplete solution methods must usually be employed. Therefore, in addition to *model errors*, where the linguistic intuition contradicts the model defined by the actually implemented grammar, *search errors* can occur, where the parser output in turn differs from the modelled optimum because the heuristic optimization algorithm terminated in a local peak.

In the case of the earlier example sentence, no problems occur: the grammar assigns the highest score to the desired analysis shown in Figure 1, and the solution algorithm actually finds it. In general, this is not always the case, particularly for long sentences. Although all individual cases we have investigated suggest that the accuracy of the model established by our grammar is higher than the actual performance of the parser (i.e., search errors decrease rather than increase the overall accuracy), this is not of practical use because the better solutions cannot be efficiently computed. Therefore, all results reported in this paper use the strictest possible measure: only the accuracy of the computed structures with respect to the intended (annotated) ones is reported.

Since our parser always returns exactly one complete syntax structure, recall and precision are both identical to the percentage of correct versus incorrect dependency edges. According to this measure, our parser typically computes between 80% and 90% of correct dependency attachments for written German. This is near the performance of the Collins parser for English [Collins1999], but somewhat below the results of [Tapanainen and Järvinen1997], who report precisions above 95% (also for English), but do not always attach all words. More relevant are the results described by [Dubey and Keller2003], who analysed 968 sentences (with at most 40 words) from the NEGRA corpus of German newspaper text. Reimplementing Collins' parser for German, they only achieved a labelled precision and recall of 66.0%/67.9%. This suggests that German syntax is considerably more difficult to analyse for this kind of parser than English. [Dubey and Keller2003] improved the parsing model to 70.9%/71.3% by considering sister-head dependencies instead of head-head dependencies.

For purposes of comparison we analysed the same section of the NEGRA treebank with our own parser. We first translated the phrase treebank to dependencies automatically, with only few edges corrected to conform to our annotation guidelines [Daum et al.2004]. The accuracy of

parsing is then measured by counting the number of correctly computed dependency edges. Input is first annotated with part-of-speech tags by the statistical trigram tagger TnT, and some typical errors of the trigram model are automatically corrected by about 20 handwritten rules. The resulting scores are integrated into the constraint grammar with a constraint that disprefers word forms with improbable categories.

We employ a heuristic transformation-based solution method that first constructs a complete syntax tree in linear time, and then tries to transform it so as to remove errors that were diagnosed by violated constraints. Three passes are made over each sentence; in the first phase, only subordinations between nearby words are allowed, and the resulting partial trees are recombined by their roots in the second phase. A third pass then tries to repair any remaining errors; only this phase investigates the entire space of possible subordinations. We have found that the phase-based approach yields better results than tackling the full optimization problem to begin with [Foth and Menzel2003].

Table 1 gives the results; altogether 89.0% of all dependency edges are attached correctly by the parser (87.0% if we also demand the correct edge label). Since the previous work reported constituent-based precision and recall figures, a direct comparison is not possible, but it seems safe to say that parsing accuracy is somewhat higher in our system. A more telling comparison would be achieved by transforming the phrase structure results of the sister-head-dependency model into dependency structures with the same tool used for transforming the treebank, and measuring the edge accuracy directly.

Sentence length correlates with a slowly increasing number of parsing errors; to a certain degree this simply reflects the greater number of plausible but wrong subordinations in long sentences. However, the incompleteness of the solution algorithm also contributes to this: as the search space increases, more alternatives are overlooked, so that the method effectively becomes less and less complete.

Corpus	# of sent.	edges	lab. edges
all sentences	1000	89.0%	87.0%
<60 words	998	89.1%	87.1%
<40 words	963	89.7%	87.7%
<20 words	628	92.3%	90.1%
<10 words	300	93.4%	91.0%

Table 1. Parsing results for NEGRA sentences.

5 Further experiments

A WCDG provides remarkable flexibility with respect to aspects of parsing behaviour. In particular, although the exact solution of the optimization problem that it poses would theoretically require exponential runtime, good results can also be achieved with much less resources. In fact, an arbitrary time limit can be set for the parsing process because the algorithm exhibits the *anytime property*: processing can be interrupted at any time if necessary, and the current analysis returned. This allows a trade-off to be made between parsing time and quality of results, since improvements to the score of an analysis generally coincide with a better accuracy.

For the results in Table 1, we allowed the parser to spend up to 600 seconds on each sentence (on 1533-MHz Athlon PC processors), and it actually terminated within 68 seconds on the average. If the time limit is lowered, accuracy slowly decreases, as shown in Table 2.

Time limit	Actual time	edges	lab. edges
600 seconds	68.0s	89.0%	87.0%
400 seconds	59.3s	88.7%	86.8%
200 seconds	44.9s	88.2%	86.2%
100 seconds	31.8s	87.1%	85.0%
50 seconds	21.6s	84.6%	82.3%

Table 2. Parsing results with reduced runtime limit.

We also investigated the generality of our model of German, originally developed to parse the text of online technical newscasts, by parsing a variety of other text types. Table 3 gives results of analysing various corpora under the same conditions as those in Table 1. In general, performance is measurably affected by text type as well as sentence length, but the accuracy is comparable to that on the NEGRA corpus. Classical literature was notably more difficult to parse, since it employs many constructions considered marked or extragrammatical by our current model of German. Transcriptions of spontaneous speech also pose some problems, since they contain many more sentence fragments than written text and no disambiguating punctuation.

A particular advantage of a language model composed of many cooperating constraints is that any single rule is not vital to the operation of

Text type	sentences	avg. length	edges	lab. edges
Trivial literature	9547	14 words	93.1%	91.1%
Law text	1145	19 words	88.8%	86.7%
Verbmobil dialogues	1316	8 words	90.3%	86.3%
Online news	1894	23 words	89.8%	88.1%
Serious literature	68	34 words	78.0%	75.4%

Table 3. Parsing results on different text types.

the grammar. In fact, since constraints forbid rather than license particular structures, a missing rule can never cause parsing to fail completely; at most it can increase the search space for a given sentence. This means that a grammar can be applied and tested even while it is still under construction, so that a realistic grammar for an entire language can be developed step by step.

To demonstrate this robustness against omitted rules, we deliberately disabled entire groups of constraints of the grammar, one group at a time, and repeated the comparison experiment of Section 4. Table 4 shows the results. For each group of constraints the general purpose is given as well as the informal version of an example constraint. The importance of each constraint group is given as the ratio between the structural correctness achieved with and without the constraints from this group.

Class	Purpose	Example	Importance
agree	rection and agreement	subjects have nominative case	1.02
cat	category cooccurrence	prepositions do not modify each other	1.13
dist	locality principles	prefer the shorter of two attachments	1.01
exist	valency	finite verbs must have subjects	1.04
init	hard constraints	appositions are nominals	3.70
lexical	word-specific rules	“entweder” requires following “oder”	1.02
order	word-order	determiners precede their regents	1.11
pos	POS tagger integration	prefer the predicted category	1.77
pref	default assumptions	assume nominative case by default	1.00
proj	projectivity	disprefer nonprojective coordinations	1.09
punc	punctuation	subclauses are marked with commas	1.03
root	NIL subordinations	only verbs should be tree roots	1.72
sort	sortal restrictions	“sein” takes only local predicatives	1.00
uniq	label cooccurrence	there can be only one determiner	1.00
zone	crossing of marker words	conjunctions must be leftmost dependents	1.00

Table 4. Measuring the effect of different constraint classes.

As expected, the most important type of constraints (in that their omission degrades performance the most) is the "init" group, into which most hard constraints fall. Parsing here suffers from a vastly increased initial ambiguity that often prevents the parser from finding the correct solution even if it is still theoretically optimal. POS preprocessing turns out to be very important as well, again because it quickly closes huge but implausible search spaces. Rules that disprefer multiple NIL subordinations effectively prevent the parser from taking the 'easy' solution and treating words that it cannot attach as additional roots. Each other group of constraints has comparatively little effect on its own.

6 Related Work

Except for [Dubey and Keller2003] there seems to be no other attempt to evaluate a syntactic parser of German on a gold standard annotation using the common PARSEVAL methodology. Accordingly, there is no standard setting available so far which could facilitate a direct parser comparison, similar to the established Penn Treebank scenario for English. However, there have been evaluation efforts for more shallow types of syntactic information, e.g. chunks or topological fields.

Parsing into topological fields like Vorfeld, Mittelfeld, and Nachfeld usually is considered a somewhat easier task than the construction of a full-fledged constituency analysis, because it clearly profits from the easily identifiable position of the finite verb in a German sentence and avoids any decision about the assignment of syntactic functions to the constituents. [Braun2003] presents a rule-based approach to topological parsing of German and reports a coverage of 93.0%, an ambiguity rate of 1.08, and a precision/recall of 86.7% and 87.3% respectively on a test set of 400 sentences. [Becker and Frank2002] trained a stochastic topological parser on structures extracted from the NEGRA-Treebank and tested it on 1058 randomly selected sentences with a maximum length of 40 words. They achieved a labelled precision/recall of 93.4% and 92.1%.

Chunking, i.e. the segmentation of a sentence into pieces of particular type (usually NPs and PPs), is an approach to shallow sentence processing sufficient for many practical purposes. [Brants1999] used cascaded Hidden Markov Models to chunk sentences into a hierarchy of structurally embedded NPs and PPs with a maximum tree depth of nine. He achieved a precision/recall of 91.4% and 84.8% in a cross evaluation on the 17,000 sentences of the NEGRA-Treebank. [Schmid and Schulte im Walde2000] trained a probabilistic context-free grammar on unlabelled data and used

it to segment sentences into NP chunks. They evaluated the system on 378 sentences from newspaper text and obtained a precision/recall of 93% and 92% if only the range of a chunk is considered, which decreased to 84% and 83% if also the case of an NP has to be determined.

Special evaluation methodologies have been designed for several parsers of German. Among them is the one used to evaluate the quality improvement during the development of the Gramotron parser [Beil et al.2002]. The method is partly motivated by lexical semantic clustering, the ultimate goal of the project, although the authors admit that a direct evaluation of head-to-head dependencies as proposed by [Lin1995] would be more appropriate. Besides avoiding overtraining by monitoring the cross-entropy on held-out data, a linguistic evaluation was carried out focussing on selected clauses types. It is based on a randomly sampled subcorpus of 500 clauses and measured the quality of determining noun chunks and subcategorization types. The precision of chunking reached 98% if only the range was considered and decreased to 92% if the chunk type was also included. The subcategorization type of verbs was computed with a precision between 63% and 73% depending on the kind of clauses under consideration.

[Langer2001] evaluated a GPSG parser with the capability of additional processing rules (of unlimited generative power) on 62,073 sentences from German newspaper text. Of the sentences with 60 words or less, 34.5% could be analysed, with an average of 78 solutions per sentence. On a self-selected test corpus with an average length of 6 words, coverage rose to 83%. Only an indirect measure of accuracy is given: in 80.8% of the newspaper sentences, giving the parser access to partially annotated target structures (chunks and some POS information) did not change its output.

[Neumann and Flickinger2002] evaluated their HPSG-parser based on the DOP-model with 1000 sentences from the Verbmobil-domain. They measured coverage (70.4% if combined), and the runtime performance of the system.

[Wauschkuhn1995] developed a two step parser computing a clause level structure which clause-internally is kept completely flat. The system is evaluated on 1097 newspaper sentences only with respect to its coverage (86.7% if run on manually tagged input and 65.4% for automatically tagged text) and the achieved degree of disambiguation (76.4% for manually and 57.1% for automatically tagged input).

7 Conclusions

A competitive parsing system for unrestricted German input has been described that is largely independent of domain and text type. Total coverage is ensured by means of defeasible, graded constraints rather than hard grammar rules. The method of using individual constraints for different grammatical principles turns out to be useful for ongoing grammar development, as the parser can be run normally from the beginning and aspects of the working language can be modelled one at a time.

We find that using a powerful formalism that allows all conceivable rules to be actually used allows for better parsing than using a tractable but less expressive one: even a theoretically intractable formalism is useful in practice if its model can be approximated well enough. While casting syntax analysis as a multidimensional optimization problem leads to high requirements of processing time, this inefficiency is somewhat mitigated by the anytime property of the transformational optimizer that allows the user to limit the desired processing time.

A greater concern is certainly the great amount of expert knowledge needed to create a complete constraint grammar. We propose that for deep analysis of German syntax, the greater precision achievable through handwritten rules justifies the extra effort compared to approaches based purely on machine learning. Although automatic extraction of constraints from annotated corpora remains a goal of further research, so far we have achieved greater success by hand-selecting and scoring all grammar rules, since this allows the grammar writer to make conscious decisions about the relative importance of different principles.

An online demonstration of our system can be accessed at `http://nats-www.informatik.uni-hamburg.de/Papa/ParserDemo`.

The source code of the parser and the grammar is available under the General Public License at `http://nats-www.informatik.uni-hamburg.de/download`.

References

[Becker and Frank2002] Markus Becker and Anette Frank. 2002. A stochastic topological parser for German. In *Proc. 19th Int. Conf. on Computational Linguistics, Coling-2002*, Taipeh, Taiwan.

[Beil et al.2002] Franz Beil, Detlef Prescher, Helmut Schmid, and Sabine Schulte im Walde. 2002. Evaluation of the Gramotron parser for German. In *Proceedings of the LREC Workshop: Beyond PARSEVAL*, Las Palmas, Gran Canaria.

[Brants1999] Thorsten Brants. 1999. Cascaded markov models. In *Proc. 9th Conf. of the European Chapter of the ACL, EACL-1999*, Bergen, Norway.

[Braun2003] Christian Braun. 2003. Parsing German text for syntactico-semantic structures. In *Prospects and Advances in the Syntax/Semantics Interface, Proc. of the Lorraine-Saarland Workshop*, Nancy.

[Collins1999] Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, Philadephia, PA.

[Daum and Menzel2002] Michael Daum and Wolfgang Menzel. 2002. Parsing natural language using guided local search. In *Proc. 15th European Conference on Artificial Intelligence, ECAI-2002*, pages 435–439, Lyon, France.

[Daum et al.2004] Michael Daum, Kilian Foth, and Wolfgang Menzel. 2004. Automatic transformation of phrase treebanks to dependency trees. In *Proc. 4th international Language Resources and Evaluation Conference*, Lisbon, Portugal.

[Dubey and Keller2003] Amit Dubey and Frank Keller. 2003. Probabilistic parsing for German using sister-head dependencies. In *Proc. 41st Annual Meeting of the Association of Computational Linguistics, ACL-2003*, Sapporo, Japan.

[Foth and Menzel2000] Kilian Foth and Wolfgang Menzel. 2000. A transformation-based parsing technique with anytime properties. In *Proc. International Workshop on Parsing Technologies (IWPT-2000)*, pages 89–100, Trento, Italy.

[Foth and Menzel2003] Kilian Foth and Wolfgang Menzel. 2003. Subtree parsing to speed up deep analysis. In *Proc. 8th Int. Workshop on Parsing Techniques*, pages 91–102, Nancy, France.

[Foth2004] Kilian A. Foth. 2004. Writing weighted constraints for large dependency grammars. In *Proc. Recent Advances in Dependency Grammars, COLING-Workshop 2004*, Geneva, Switzerland.

[Langer2001] Hagen Langer. 2001. *Parsing-Experimente: praxisorientierte Untersuchungen zur automatischen Analyse des Deutschen*. Peter Lang, Frankfurt am Main.

[Lin1995] Dekang Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *IJCAI*, pages 1420–1427.

[Maruyama1990] Hiroshi Maruyama. 1990. Constraint dependency grammar. Technical Report RT0044, IBM Research, Tokyo Research Laboratory.

[Neumann and Flickinger2002] Günter Neumann and Dan Flickinger. 2002. HPSG-DOP: Data-oriented parsing with HPSG. In *Proc. of the 9th Int. Conf. on HPSG, HPSG-2002*, Seoul, South Korea.

[Schmid and Schulte im Walde2000] Helmut Schmid and Sabine Schulte im Walde. 2000. Robust German noun chunking with a probabilistic context-free grammar. In *Proc. 18th Int. Conf. on Computational Linguistics, Coling-2000*, Saarbrücken, Germany.

[Schröder2002] Ingo Schröder. 2002. *Natural Language Parsing with Graded Constraints*. PhD thesis, Hamburg University, Hamburg, Germany.

[Tapanainen and Järvinen1997] Pasi Tapanainen and Timo Järvinen. 1997. A non-projective dependency parser. In *Proc 5th Conference on Applied Natural Language Processing*, Washington, D.C.

[Wauschkuhn1995] Oliver Wauschkuhn. 1995. The influence of tagging on the results of partial parsing in German corpora. In *Proc. 4th Int. Workshop on Parsing Technologies, IWPT-1995*, pages 260–270, Prague/Karlovy Vary, Czech Republic.

The Role of Animacy Information in Human Sentence Processing Captured in Four Conflicting Constraints

Monique Lamers & Helen de Hoop

Department of Linguistics, Radboud University Nijmegen[1]
{H.deHoop, M.Lamers} @ let.kun.nl

Abstract. To formalize and analyze the role of animacy information in on-line sentence comprehension, results of several on-line studies are compared and analyzed according to a new model of incremental optimization of interpretation. This model makes use of violable ranked constraints. To analyze the use of animacy information a set of four constraints is needed, namely Case, Selection, Precedence, and Prominence. It is shown that the pattern of constraint violations of these four constraints provide sufficient information to reflect the on-line effects of language comprehension studies in which animacy information played a crucial role. More specifically, the evaluation of sentences in which either case information or animacy information in combination with the selection restrictions of the verb were used, showed that the model can account for the ambiguity resolution with both sorts of information. The model was also successfully applied to the on-line processing of a more complex object relative structure in English.

1 Introduction

The different sorts of information that become available incrementally during natural language comprehension are very diverse varying from morphosyntactic information (e.g. case marking, word order, number) to semantic/conceptual information (e.g. animacy, specificity). In contrast to the morphological marking of case and number, which incorporates both semantic and syntactic information, animacy is merely semantic (or conceptual) in nature. It plays a role of utmost importance in fulfilling the selectional criteria of the verb, and as such, it is quintessential for the incremental interpretation of a sentence.

To investigate the role of animacy information in sentence comprehension a newly developed model of incremental optimization of interpretation is used [7]. This model uses a set of ranked interpretive constraints that are violable on a word-by-word basis. These constraints are derived by analyzing characteristics of Dutch and English that are relevant to the processing of animacy information.

[1] The research reported here was supported by the Netherlands Organisation of Scientific Research (grants #220-70-003 to the PIONIER project *Case Cross-linguistically* and #051-02-070 to the Cognition project *Conflicts in Interpretation*), which is gratefully acknowledged.

This paper presents a first attempt to map the patterns of constraint violations onto the on-line effects found in studies in which the use of animacy information was investigated. Thus, the model of incremental optimization of interpretation that is adapted in this paper can be shown to predict the kind of processes elicited on-line on the basis of the pattern of constraint violations. By applying this model that is based on principles of the time insensitive model of Optimality Theoretic Semantics [6] to on-line human sentence processing studies, we bridge the gap between theoretical linguistic models and natural language processing.

2 Four Constraints on Interpretation: Case, Precedence, Prominence, and Selection

In languages with a case system such as German, morphosyntactic information can be used to interpret the syntactic and semantic relations. In German, not only personal pronouns, but also articles and adjectives are overtly case marked. If a sentence starts with an accusative case marked NP, as is illustrated in (1), it will not only be identified as the object of the sentence, but it also triggers the expectation of a suitable subject and predicate. Thus case information helps to determine the interpretation.

1. Den Zaun hat der Junge zerbrochen.
   [the fence]$_{ACC,3rd,sg}$ has$_{3rd,sg}$ [the boy]$_{NOM3rd,sg}$ broken
   "The fence, the boy broke."

There are, however, also languages with a poor case system such as Dutch and English. In these languages, case is only visible on pronouns. Hence, the eventual interpretation of many sentences is dependent on other sorts of information. Even in languages with poor case marking there are hardly any sentences that are difficult to interpret. In English, this is not all that surprising because of the strict word order that constrains the interpretation in such a way that the first argument in a main clause is almost always the subject of the sentence.[2] Because Dutch has a relatively free word order and no case marking on full noun phrases, many sentences are ambiguous. In these cases, in Dutch as well as in German, in which due to morphological poverty of certain phrases the same kind of ambiguities occur, the interpretation will be based on the strong preference for the canonical word order with its concomitant (subject precedes object) reading. This is also known as word order freezing (cf. [11], [20]). In a sentence such as in (2), there is a strong preference for interpreting the first NP as the subject. In other words, in the absence of conflicting information, the preferred interpretation is one in which the subject precedes the object (cf. [4], [5], [12]).

2. De patiënt heeft de arts bezocht.
   [the patient]$_{SUBJ/OBJ}$ has [the doctor]$_{SUBJ/OBJ}$ visited
   "The patient visited the doctor."

[2] As can be seen in the English translation of example (1), it is possible that the object precedes the subject in English. This, however, can only be the case as a form of topicalisation.

Most experiments that found evidence for the subject before object preference dealt with sentences where the subject and the object both were animate used in combination with agentive transitive verbs. However, several off-line and on-line experiments in Dutch and English showed that besides case-marking and word order, animacy information is an important source of information for the comprehension process (cf. [9],[14],[15],[19]). For instance, McDonald [15] compared the validity of three cues (word order, noun animacy and case inflection on pronouns) in choosing the controller (agent) of transitive sentences and relative clauses. Experiments in which subjects had to assign the agent role after listening to the sentence showed that, in Dutch, animacy was a better cue than word order, whereas, in English, the reverse was the case. MacWhinney, Bates and Kliegl [13] found that in German, too, animacy was a better cue than word order. Hence, there seems to be coherence between word order freedom and the role of animacy information in sentence processing in different languages.

Where word order preferences might help to identify the subject of a sentence, animacy information provides us with information about the potential control over the action expressed by the verb, as well as the prominence relation between the arguments. That is, given the fact that for many verbs it is the subject (which in most sentences is the first argument of the sentence) that is in control of the action, it is expected that given an animate and inanimate NP in a sentence, it is more likely that the animate NP is the subject and the inanimate NP is the object of the sentence. In other words, the first argument outranks the second argument in control or prominence, thus in most cases, in animacy. Note, however that in sentence processing the information becomes available incrementally. Therefore, the animacy of a possible second NP is not yet available at the moment the first NP is being processed. Moreover, it might be unclear whether a transitive or intransitive verb comes in. If animacy information of the initial NP is used as soon as it becomes available it cannot be used in relation to the animacy of the other argument(s). What can be taken into account is whether the argument that is being processed is animate and is potentially in control and prominent.

As we have seen in example (1) it is very well possible that word order preference is not followed. This also holds for the expectation of the prominence relationship between the arguments and the control characteristics of the subject. Although not as common as an animate subject, it is not only possible that the subject of a sentence is inanimate, it might also be that given an animate and inanimate NP, the inanimate NP has to be the subject of the sentence and the animate NP the object. It is the verb that imposes these selection restrictions onto the arguments, as is illustrated in (3a,b). Verbs such as *please* take an experiencer (animate) object, while verbs as *like* take an experiencer (animate) subject.

3. a. The holiday pleased the boy.
   b. The boy liked the holiday.

So far, four possible constraints that play an important role in language processing follow from the above discussion. The first constraint is based on morphological case-marking, the second on a general preference of word order, whereas the other two

constraints are more directly related to the role of animacy information. In (4) the self-explanatory constraint CASE is defined.

4. CASE: the subject is nominative case marked, the object is accusative case marked

In (5) the word order constraint PRECEDENCE is defined which is based on a strong preference for the subject-before-object word order. If this constraint is satisfied, it leads directly to the interpretation in which the subject precedes the object.

5. PRECEDENCE: the subject (linearly) precedes the object

Note that PRECEDENCE, as it is formulated above, can be considered an instantiation of a linearity constraint as proposed within the framework of Property Grammars [2]. However, the use of constraints in Property Grammars (PG) radically differs from our use of constraints. In PG constraints are used to encode syntactic information and although constraint evaluation takes place on the basis of grammatical as well as ungrammatical inputs, a grammatical input should not violate any constraints. In our Optimality Theoretic model of interpretation, the input is the form (being it grammatical or not) and the output (that is built up incrementally) is the (optimal) interpretation of that form. The constraints are potentially conflicting, which is one of the basic characteristics of Optimality Theory [16], and as a consequence optimal outputs often violate numerous constraints (in order to satisfy stronger ones).

The third constraint, PROMINENCE, concerns the potential control characteristics of the subject and the prominence relationship between the subject and the object. As it is defined in (6), it is expected that the subject of a sentence has potential control over the action expressed in the sentence. Additionally, we assume that in case of a transitive relationship the subject outranks the object in PROMINENCE. In terms of animacy, it is expected that the subject is animate, and when a second argument is present, the constraint is clearly satisfied if the object is inanimate.

6. PROMINENCE: a) the subject is animate and thus has potential control;
b) the subject outranks the object in prominence

The fourth constraint is called SELECTION and reflects the inherent semantic relation between the verb and the subject or the object. This constraint comes about if a verb always selects an inanimate NP as the object and/or an animate NP as the subject.

7. SELECTION: the verb selects an animate object, and/or an animate subject

At this point, note that in sentence (3a) where the initial NP is the inanimate subject, the word order constraint PRECEDENCE is satisfied, but PROMINENCE is violated. This violation of PROMINENCE is directly related to the satisfaction of SELECTION. In sentence (1) on the other hand, PRECEDENCE is violated, but PROMINENCE is satisfied. Here, the satisfaction of PROMINENCE goes hand in hand with the satisfaction of CASE. The interplay of the different constraints will be further explained in the following section.

3 The Model of Incremental Optimization of Interpretation of Animacy Information

In this section, the constraints will be ranked so that they can be used in the model of incremental optimization of interpretation. The ranking of the constraints is established by principles taken from the theoretical perspective of Optimality Theory (cf. [16]) and Optimality Theoretic Semantics (cf. [6]).

As already pointed out above, within the framework of Optimality Theory constraints are violable rules that are potentially conflicting. These constraints are reflections of linguistic regularities. A constraint is never violated without any reason, but only in order to satisfy another, stronger constraint. This basic characteristic of Optimality Theory (OT) originates from its predecessor, Harmonic Grammar, a linguistic theory that uses the connectionist well-formedness measure Harmony to model linguistic well-formedness [18].

As the name of our model indicates, we assume that during human sentence processing, the optimal interpretation of a sentence (form) is being built up incrementally. Hence, we assume the process of optimization itself to be incremental. Optimality Theoretic Semantics [6] gives a straightforward tool for analyzing processing in this way. OT semantics [6] take as a point of departure free generation of interpretations in combination with the parallel evaluation of violable constraints. The integration of pragmatic, semantic, and syntactic information in a system of ranked constraints is proposed to correctly derive the optimal interpretations for inputs that contain of utterances, i.e., forms in context. Thus, in OT semantics, the direction of optimization is from form to meaning, that is, it is optimization from the hearer's point of view. To use this approach for our purpose of analyzing experimental results of processing requires an incremental approach to optimization. That is, the process of optimization of interpretation proceeds while the information comes in word-by-word, or constituent-by-constituent.

Before incremental optimization of interpretation can be used to analyze the role of animacy information in sentence processing the four relevant constraints that were defined above, have to be ranked. We determine the ranking by examining the optimal output interpretation in case of a conflict between constraints. Such a situation was illustrated in example (3a) in which an inanimate NP is interpreted as the subject of the sentence. This indicates that SELECTION must outrank PROMINENCE, since the optimal reading will be an animate object reading, despite the fact that this leads to an inevitable violation of PROMINENCE. If the animate object precedes the inanimate subject, there is not only a conflict between SELECTION and PROMINENCE, but also between SELECTION and PRECEDENCE. This is illustrated by the Dutch example in (8).

8. De jongen beviel de vakantie.
   the boy pleased the holiday
   "The holiday pleased the boy"

The optimal reading for the sentence in (8) is the one in which the initial NP is the object of the sentence. Thus, PRECEDENCE is violated to satisfy SELECTION; hence

SELECTION not only outranks PROMINENCE, but also PRECEDENCE. This leaves us with the issue of ranking PRECEDENCE and PROMINENCE. Because of much processing evidence that in case of ambiguity, a sentence-initial NP is interpreted as the subject even if it is inanimate, we assume that PRECEDENCE outranks PROMINENCE. Finally, we assume that CASE is the strongest of the four constraints. This can be derived from the fact that we obtain a sometimes pragmatically odd yet optimal interpretation when SELECTION must be violated in order to satisfy CASE, as in a German sentence such as *Der Zaun hat den Jungen zerbrochen* glossed as 'the fence$_{NOM}$ has the boy$_{ACC}$ broken'. In this sentence, despite the fact that transitive *break* normally selects an animate subject, the inanimate NP *der Zaun* 'the fence' is interpreted as the subject because of its nominative case. That is, the sentence can only mean that the fence broke the boy (see [17] for more evidence on the important role of case in human sentence processing). In (9) the ranking of the four constraints is given.

9. CASE >> SELECTION >> PRECEDENCE >> PROMINENCE

4 Applying Incremental Optimization of Interpretation to the On-line Use of Animacy in Dutch Sentence Comprehension

Having established the ranking of the constraints, incremental optimization of interpretation can be used to evaluate word-by-word sentences used in on-line studies in which animacy information was manipulated. We will apply incremental optimization to three reading studies in which event related brain potentials (ERPs) were measured. ERPs are small changes in spontaneous activity of the brain that occur in response to certain sensory events. Because of the multidimensionality of the signal, differences in the effects are related to differences in the nature of the involved processes [8]. This makes these studies extremely suitable as test cases for the application of incremental optimization of interpretation.[3]

In the first two studies animacy information is used to resolve subject-object ambiguities in Dutch, as was reported by Lamers [9], [10]. Additionally, object relative clauses taken from a study of Weckerly and Kutas [19] will be evaluated.

Lamers [9], [10] investigated sentences such as given in (10), (11) and (12) in two ERP studies.

10. De oude vrouw in the straat verzorgde hij …
    The old woman in-the-street took-care-of he …
    "He took care of the old woman in the street …"

11. Het oude park in the straat verzorgde hij …
    The old park in-the street took-care-of he …
    "He took care of the old park in the street …"

[3] For a similar application of incremental optimization of interpretation to the processing of sentences in which the distinguishability of two arguments used in a transitive relation was addressed, the reader is referred to de Hoop & Lamers [7].

12. De oude vrouw in the straat verzorgde hem …
The old woman in the street took-care-of him …
"The old woman in the street took care of him …"

Notice that sentence (11) is disambiguated by the animacy information of the initial NP because of the selection restrictions of the verb. That is, the verb selects an animate subject and as a consequence, when the verb comes in, the sentence-initial NP can (no longer) be interpreted as the subject of the sentence. The sentences in (10) and (12) are clearly disambiguated at the time the case-marked pronoun is encountered (which is unambiguously nominative in (10) and accusative in (12)). Lamers reports early and late positivities at the verb for sentence (11) starting with the inanimate NP in comparison to sentence (10) as well as at the nominative case marked pronoun in sentence (10) in comparison to the accusative case marked pronoun in (12). The effects were interpreted as structure building problems, possibly involving reassignment of syntactic functions and thematic roles. Strikingly, the effects are similar although different sorts of information are used for disambiguation (the verbal selection criteria and the case-marking of the pronouns, respectively). We claim that this similarity can be explained within the incremental optimization model. The evaluation of these sentences against the set of constraints will show that the pattern of constraint violations is the same in the two different contexts. In the left panel of Table 1 a schematic overview is given of the constraint violations pattern of the optimal interpretation of the incoming words of sentence (10)[4]. In the right panel the evaluation of the object-initial sentence (11) is given. As can be seen, PROMINENCE is violated as soon as the initial inanimate NP becomes available in order to satisfy the higher ranked constraint PRECEDENCE. In that stage, the optimal interpretation is the subject-before-object reading. Up to the verb SELECTION cannot play a role in the parsing process, since no relevant information is available. At the verb, it becomes clear that the subject has to be animate. Hence, the optimal interpretation of the initial inanimate NP changes from subject to object. As a consequence, PRECEDENCE is violated, but PROMINENCE is satisfied. It is at this point in the sentence that Lamers found the significant ERP effects (early and late positivities) for sentence (11) compared to (10) (Table 1).

[4] In contrast to tableaux normally used to present constraint satisfaction patterns in Optimality Theory, in this paper a table shows only the pattern of constraint violations of **the optimal interpretation at time t**. Obviously, what is the optimal interpretation at time t may vary through time. For example, the optimal interpretation of the sentence in (12) will be the subject-initial interpretation **until** the verb comes in. Then the optimal interpretation switches to the object-initial interpretation, due to the fact that Selection is ranked above Precedence.

Table 1. An overview of the pattern of violations of CASE, SELECTION (SEl), PRECEDENCE (PREC), and PROMINENCE (PROM) for the sentences from the examples (10) and (11) up to the verb. In this table as in all the follwing tables an ✔ indicates that at this word the optimal interpretation satisfies the constraint, whereas an * indicates a violation of the constraint; Pos. = positivity; the crucial words are in bold; grey columns indicate the point in the sentence in which ERP differences were reported.

	De oude vrouw…	verzorgde…	Het oude park…	verzorgde…
CASE				
SEL		✔		✔
PREC	✔	✔	✔	*
PROM	✔	✔	✔	✔
ERP				early , late Pos.

Table 2 is a schematic overview of the constraint violations patterns of the crucial words of sentences (12) (left panel) and (10) (right panel). The ERP effect that is illustrated was found at the moment the case marked pronoun was encountered.

Table 2. An overview of the pattern of violations of CASE, SELECTION (SEl), PRECEDENCE (PREC), and PROMINENCE (PROM) for the sentences (12) and (10) until the disambiguating case-marked pronoun.

	De oude vrouw…	verzorgde…	hem…	De oude vrouw…	verzorgde…	hij…
CASE			✔			✔
SEL		✔	✔		✔	✔
PREC	✔	✔	✔	✔	✔	*
PROM	✔	✔	✔	✔	✔	✔
ERP						early, late Pos

Let us now compare the pattern of constraint violations found at the nominative case marked pronoun to the pattern observed at the verb of the sentence starting with the inanimate NP (Table 2 versus Table 1). At the nominative case marked pronoun information becomes available such that the object-initial interpretation overrules the subject-initial interpretation, which was optimal until that point. At that time, PRECEDENCE must be violated (the object precedes the subject), whereas PROMINENCE is still satisfied, given that a pronoun is more prominent in a discourse prominence hierarchy than a full NP (cf. [1], [3], [11]). The resulting pattern is basically the same pattern as the one created at the verb of the inanimate condition (11). Thus, the similarity of the constraint violation patterns indeed reflects the similarity in ERP effects reported in the on-line studies. This indicates that the four constraints of the incremental optimization of interpretation model successfully

capture the role of animacy information in subject-object ambiguity resolution in Dutch.

5 Extension of the Approach to English

Although English has a strict word order, relative clauses can either be subject initial or object initial. Weckerly and Kutas [19] investigated the influence of the animacy on the processing of object relative clauses with a local structural ambiguity. The sentences used in this study differed in the word order of an animate and inanimate NP, as exemplified in (13) and (14)

13. The novelist that the movie inspired praised the director …

14. The movie that the novelist praised inspired the director …

In the initial NP, the subject of the main clause is either animate (as in 13) or inanimate (as in 14). Hence, there is already a difference in the pattern of constraint violations at the first argument, as is illustrated in Table 3. Since it concerns a violation of the lower ranked constraint in order to satisfy PRECEDENCE, it is not expected that this violation pattern will lead to strong on-line effects. Nevertheless, Weckerly and Kutas do report an enhanced negative shift for the sentence with an initial inanimate NP, indicating that the processing of an initial inanimate NP is more costly than an animate NP.

At the word *that* it is clear that a relative clause has to be processed, although it is not yet known whether it concerns a subject or an object relative clause. Since *that* refers to the initial NP, the pattern of constraint violations is the same as was found at the previous word.

As soon as the determiner comes in, it becomes obvious that an object relative clause is being processed (because the incoming NP must be the subject of the relative clause). We assume that at this point in the sentence the comprehension process is mainly concerned with integrating incoming information in the relative clause, while the main clause information is stored in verbal working memory. Thus, focusing at the relative clause, at *that* PRECEDENCE is violated in both sentences. Because both sentences have the same structure, this violation will not be reflected in the ERP-waveform. If, however, a comparison would have been possible between subject relative clauses and object relative clauses we expect this violation to be reflected as an early and late positivity corroborating the findings of Lamers [9], [10]. Since subject relative sentences were not part of the study of Weckerly and Kutas [19], such a comparison is unfortunately not possible.

In sentence (13) the object refers to an animate NP, and thus PROMINENCE is violated (Table 3, left panel). In (14) the object is inanimate and thus, it is still very well possible that the incoming subject of the relative clause outranks the object in prominence and control. Since there is a difference in the pattern of constraint

violations between the two sentences, differences in ERP waveforms are expected. However, at the determiner of the incoming subject of the relative clause, no differences in ERP-waveforms were reported. This might be due to the fact that a determiner is a high frequent closed class word, which has to be followed by the semantically more important noun, the subject of the relative clause.

Table 3. An overview of the violations of SELECTION (SEl), PRECEDENCE (PREC), and PROMINENCE (PROM) for the sentences (13) and (14). We have left out the constraint CASE, because it does not play a role here. Neg.= negativity; LAN= left anterior negativity.

	The	novelist	that	the	movie	inspired	praised	The	movie	that	the	novelist	praised	inspired
SEL						✔	✔						✔	✔
PREC	✔	✔	✔	*	*	*	✔	✔	✔	✔	*	*	*	✔
PROM	✔	✔	✔	*	*	*	✔	✔	*	*	✔	✔	✔	*
ERP					N400	late Pos.	LAN late Pos.		Neg. shift	Neg. shift				

In sentence (13) the subject of the relative clause is inanimate. Consequently, PROMINENCE is violated. In (14), however, the subject is animate and outranks the inanimate object in prominence. The comparison of the two ERP waveforms showed an N400 for the inanimate NP.[5] Weckerly and Kutas explain this N400 in terms of the violation of the expectancy of an animate noun. Since animacy is clearly involved in the violation of PROMINENCE, the explanation of Weckerly and Kutas corresponds with the constraint violation pattern (in accordance with the claims made in [7]).

The subject of the relative clause is followed by the verb. In sentence (13) the verb *inspire* comes in, which selects an animate object. Hence, SELECTION is satisfied at the cost of a violation of PROMINENCE. *Praise*, in (14), can only be used in sentences with an animate subject, and thus SELECTION and PROMINENCE can both be satisfied. A late positivity (P600) was found at the verb that violates SELECTION. Weckerly and Kutas argue that this positivity is caused by more difficult processing probably involving problems in thematic role assignment. This explanation corroborates our discussion above that the so-called experiencer verbs assigning the role of experiencer to the (necessarily animate) object of the sentence, and not the more frequently used roles of theme or patient (cf. [9]) inherently combine the satisfaction of SELECTION with a violation of PROMINENCE.

More puzzling is the ERP-effects found at the verb of the main clause. As can be seen in Table 3, ERP-effects are found at the verb that does not cause any constraint violation. This contrasts with the correspondence between ERP-effects and constraint violations in every other word position of the sentences. A possible explanation for this discrepancy between the incremental optimization of interpretation analysis and the ERP-effects might be that the complexity of the structures at issue in this study is not accounted for by just the three constraints used for our present purposes. The

[5] The N400 is normally taken as an indication of problems in the comprehension process of semantic information, possibly including ease of lexical access and lexical/discourse integration processes (cf.[8],[21]).

constraints that were used in the model of incremental optimization are directly derived from rules and mechanisms of the language faculty. It would be highly unlikely to assume that the effects found in the ERP-waveforms are only reflections of language specific processes. As a matter of fact, Weckerly and Kutas explain the finding of a left anterior negativity in terms of differences in memory load caused by the difficulties of the preceding object relative clause with the inanimate subject. A similar reasoning is given for the late positivity, which is interpreted as a reflection of the complexity of the object relative clause. Further research is necessary to determine whether the pattern of constraint violations, such as the one found for the sentences with an inanimate subject (Table 3, left panel) is an indication for high demands on working memory and general processing complexity.

Nevertheless, we think that finding a correspondence between constraint violations and ERP-effects on almost every word position indicates that the model of incremental optimization of interpretation can also successfully be used for the analysis of the human processing of complex structures.

6 Conclusion

If an argument is processed, animacy information becomes available. In this paper we have shown that a newly developed model of incremental optimization of interpretation can analyze the role of animacy information in language comprehension. In this model four violable constraints were defined, namely Case, Selection, Precedence, and Prominence. By applying the constraints incrementally in a word-by-word fashion, patterns of constraint violations come about that largely correspond to the differences in ERP waveforms found in the relevant ERP studies.

We conclude that the incremental optimization model provides a useful tool in psycholinguistic research that bridges the gap between theoretical linguistic models and natural language processing.

References

1. Aissen, J.: Differential Object Marking: Iconicity vs. Economy. Natural Language and Linguistic Theory 21 (2003) 435-483
2. Blache, P.: Constraints, Linguistic Theories and Natural Language Processing. In: Christodoulakis, D. (ed.): Lecture Notes in Artificial Intelligence. Springer (2000)
3. Comrie, B.: Language Universals and Linguistic Typology. University of Chicago Press, Chicago (1989)
4. Frazier, L.: Resolution of Syntactic Category Ambiguities: Eye Movements in Parsing Lexically Ambiguous Sentences. Journal of Memory and Language 26 (1987) 505
5. Gibson, E.: Linguistic complexity: Locality of syntactic dependencies. Cognition 68 (1998) 1-76
6. Hendriks, P., de Hoop, H.: Optimality Theoretic Semantics. Linguistics and Philosophy 24 (2001) 1-32

7. de Hoop, H., Lamers M.: Incremental distinguishability of subject and object. In: Kulikov, L., Malchukov, A., de Swart, P. (eds.): Case, Valency, and Transitivity. Benjamins, Amsterdam (to appear)
8. Kutas, M., Van Petten, C.K.: Psycholinguistics electrified. In: Gernsbacher, M.A. (ed.): Handbook of psycholinguistics. Academic Press, New York (1994) 83-143
9. Lamers, M.J.A.: Sentence processing: using syntactic, semantic, and thematic information. PhD dissertation University of Groningen (2001)
10. Lamers M.J.A.: Resolving subject-object ambiguities with and without case: evidence from ERPs. In: Amberber, M., de Hoop, H. (eds.): Competition and variation in natural languages: the case for case. Elsevier (to appear)
11. Lee, H.: Parallel Optimization in Case Systems. In: Butt, M., King, T. (eds.): Nominals: Inside and Out. CSLI, Stanford (2003)
12. MacWhinney, B.: Models of the emergence of language. Annual review of psychology 49 (1998) 199
13. MacWhinney, B., Bates, E., Kliegl, R.: Cue Validity and Sentence Interpretation in English, German, and Italian. Journal of verbal learning and verbal behavior 23 (1984) 127
14. Mak, W.M., Vonk, W., Schriefers, H.J.: The influence of animacy on relative clause processing. Journal of Memory and Language 47 (2002) 50-68
15. McDonald J.: Sentence interpretation processes: The influence of conflicting cues. Journal of Memory and Language 26 (1987) 100-117
16. Prince, A., Smolensky, P.: Optimality: From neural networks to universal grammar. Science 275 (1997) 1604-1610
17. Schlesewsky, M., Bornkessel, I.: On incremental interpretation: Degrees of meaning accessed during sentence comprehension. Lingua 114 (2004) 1213-1234
18. Smolensky, P.: Information Processing in Dynamical Systems: Foundations of Harmony Theory. In: Rumelhart, D.E., McClelland, J. *et al.*: Parallel Distributed Processing. Explorations in the microstructure of Cognition, Vol. 1. MIT Press, Cambridge, MA (1986)
19. Weckerly, J., Kutas, M.: An electrophysiological analysis of animacy effects in the processing of object relative sentences. Psychophysiology 36 (1999) 559-570
20. Zeevat, H.: Freezing and Marking. Manuscript University of Amsterdam, Amsterdam (2004)
21. Garnsey, S.M. (ed.): Event-related brain potentials in the study of language. Special issue of Language and Cognitive Processes (1993)

An Exploratory Application of Constraint Optimization in Mozart to Probabilistic Natural Language Processing

Irene Langkilde-Geary

Brigham Young University, Provo UT 84602, USA,
irenelg@cs.byu.edu

Abstract. This paper describes an exploratory implementation in Mozart applying constraint optimization to basic subproblems of parsing and generation. Optimization is performed on the probability of a sentence using dependency-style syntactic representation, which is computed using an adaptation of the English Penn Treebank as data. The same program solves both parsing and generation subproblems, providing the flexibility of a general architecture combined with practical efficiency. We show results on a sample sentence that is a classic in natural language processing.

1 Introduction

Although many people have long recognized that natural language can be quite naturally characterized using constraints, the question of how to organize a computer system to effectively deal with interdependent constraints in the face of exponential complexity continues to be an active area of research. This issue of system architecture has probably been most extensively addressed in the context of generation, where the problem is quite prominent [1]. In a recent attempt to generalize across the architectures of 19 representative generator systems, Cahill et al. proposed a whiteboard as a reference design [2,3]. They argued that a whiteboard was general enough to subsume the variety of architectures surveyed and could serve as a standard to facilite the sharing and reuse of processing resources.

A whiteboard architecture divides processing into knowledge-centric functional modules (such as lexicalization, aggregation, rhetorical structuring, referring expression generation, ordering, segmentation, and coherence) and models communication between modules with a single repository of data called a *whiteboard*. The data stored in a whiteboard is typed and relational, and added cumulatively by extending or copying existing data, but not changing or removing anything. A whiteboard thus generalizes further what had previously been considered the most general architecture for generation, namely a blackboard architecture, in which changes to the data repository are destructive, not cumulative.

Although there does not seem to be much debate on the theoretical adequacy of a whiteboard as a general-purpose architecture, existing large-scale practical systems to-date have actually used something simpler (typically a pipeline) for the sake of efficiency. The new development of concurrent constraint programming, however, seems to offer the potential to combine both theoretical adequacy and practical efficiency in the same system. A concurrent constraint program (CCP) is a flexible, general, and principled framework for managing convoluted dependencies and combinatorial complexity such as that which exists in natural language processing. A concurrent constraint program's behavior can be analogized to that of a whiteboard architecture even though the specific representation and processing details may differ from those proposed by [4].

In this paper we present an exploratory implementation of a finite domain constraint optimization program in the Mozart programming system to solve some basic subproblems of both parsing and generation. Our motivation is determining how well this basic program might serve as a suitable foundation for a whiteboard-like system capable eventually of handling the full range of generation tasks with reasonable run-time efficiency.

We are interested in using a single program to solve both parsing and generation problems for more than theoretical or academic reasons. We find that parsing can often be a subproblem of general-purpose generation. Our ongoing work in machine translation frequently requires parsing of phrasal fragments that must be massaged to fit into a larger whole. In other applications of generation it is often helpful to combine processing of canned text and templates with traditional generation from abstract representations of meaning [5,6,7]. One potential way to seamlessly process all three kinds of input is by integrating parsing and generation.

A whiteboard-style architecture without a predefined processing sequence could clearly facilitate a tighter integration of the two tasks. A constraint program seems to be a natural fit for this problem. If natural language is represented declaratively as set of variables associated with each word, then parsing and generation are simply inverse (almost) divisions of variables into "given" versus "to be solved for". This declarative framework makes it easy to blur the distinction between parsing and generation and address hybrid problems with nearly arbitrary combinations of known versus unknown variables, resulting in an NLP system with wider applicability.

To see how well a CCP might model a whiteboard architecture and solve hybrid problems we perform experiments with different combinations of given versus unknown variables. The paper is organized as follows: we first describe how we formulate our exploratory subproblem of parsing/generation as a concurrent constraint program (Section 2). Then in Section 3 we discuss how the program performs on a sample input sentence with different combinations of known and unknown variables. Finally, Section 4 concludes with a discussion of related work and future directions.

2 Problem Formulation

In our formulation of the problem we use a dependency-style syntax and represent a sentence as a group of words where each word is associated with a set of linguistically-motivated features. We define a probability distribution over a subset of the features, optimizing for the most probable assignment of values to features given a few hard constraints. We obtain the probabilities from an adaptation of the English Penn Treebank. Note that our constraint optimization approach to the problem does not require or involve an explicit grammar (or lexicon) at all.

2.1 Variables

In this exploratory implementation, we restrict our attention to just a handful of features per word plus some needed auxiliary variables. Table 1 summarizes the main features associated with each word. They are node ID, head ID, functional role, group type, direction, relative position, and absolute position. To keep things simple at this stage and because we don't yet have a morphological component implemented, we do not directly use the actual words of the sentence, only the features listed.

The **ID** is an arbitrary number associated with a word, and is used together with the **HeadID** feature to represent the dependency structure of a sentence. We define the actual value of the ID to have no linguistic significance, and assume that it is either given in the input or internally assigned during initialization. Each word has exactly one head, except the top word in the sentence. The number 0 is reserved as the value of the HeadID feature when there is no head.

The **role** feature represents the functional relationship of a child with respect to its head. We currently distinguish 23 possible values, listed in Table 2. The roles are derived from the original annotation on the Penn Treebank and/or motivated by probabilistic modeling needs. In some cases, the role feature correlates with the part-of-speech of a word, for example: determiner and right-punc. In other cases, it identifies prominent parts of a clause, such as the subject or the function of an auxiliary verb. The role "role-marker" designates prepositions and complementizers, which we treat as left dependents (not heads) in

Feature	Value
ID	a word id
HeadID	id of head word
Role	syntactic function, see Table 2
Group type (GT)	clause, np, other
Direction (DIR)	+/- from head
Relative position (RP)	tree distance from head
Absolute position (AP)	distance from start of sent

Table 1. Word features

constituents typically labelled in constituency grammar as PP and SBAR. The Top role identifies the top of the sentence as a whole. "Adjunct" serves as a catch-all miscellaneous role.

Adjunct	Determiner	LGS-adjunct	Polarity	Role-marker	Top
Aspect	Emphatic	Modal	Pre-determiner	Subject	Topic
Closely-related	Junction-marker	Object	Predicate	Taxis	Voice
Dative	Left-punc	Particle	Right-punc	To-infinitive	

Table 2. Roles

The **group-type** (gt) feature is a generalization of constituent-style non-terminal labels associated with the head word of the constituent. We distinguish just three coarse categories: clause, noun phrase (np), and other—where clauses include SBAR phrases, and noun phrases include PPs. The **direction** (dir) feature indicates whether a child comes to the left or right of its head. It is partially redundant with the relative position feature, but useful as a generalization of it. The **relative position** (rp) indicates that a word is the nth dependent attached to one side of a head with the sign indicating which side. The value used for n is actually offset by 1000 to keep the domain positive in the implementation. Finally, **absolute position** (ap) designates the linear order of words in a sentence, with the first word of the sentence assigned position 1.

2.2 Constraints

Besides domain constraints for each variable we use the following non-basic constraints that define the relationships between the features of words, the tree structure of a sentence and the probabilistic score of the sentence as a whole. Auxiliary feature variables used include NumLeftCs (number of left children), and NumRightCs. The notation `==>` designates a logical implication, and `<==>` a logical equivalence. The dot notation `A.f` refers to feature `f` of node `A`. Node names "Head" and "Child" refer to confirmed head and child nodes, respectively.

Single Node Constraints

```
Dir= '-'  <==>  RP < 1000
Dir= '+'  <==>  RP >= 1000
HeadID=0 <==> RP=1000
HeadID=0  ==> Dir= '+'
```

Head-Child Constraints

```
Head.ap>Child.ap  <==>  Child.dir= '-'
Head.ap<Child.ap  <==>  Child.dir= '+'
A.numLeftCs= Number of B nodes for which A.id=B.headID AND B.dir= '-'.
A.numRightCs= Number of B nodes for which A.id=B.headID AND B.dir= '+'.
```

Relationship between the number of children on a side and the set of valid relative position values:

```
A.id=B.headID AND B.dir= '-'  ==>  B.rp >= 1000-A.numLeftCs
A.id=B.headID AND B.dir= '+'  ==>  B.rp =< 1000+A.numRightCs
```

Other Definitional Constraints

```
ForAll AP, AllDistinct(AP)
```

Rule out impossible heads based on absolute position:

```
A.dir= '-' AND A.ap > B.ap  ==>  A.headID <> B.id}
A.dir= '+' AND A.ap < B.ap  ==>  A.headID <> B.id}
|A.ap - B.ap| < A.rp  ==>  A.headID <> B.id}
```

Among siblings, relative positions are unique. Also, smaller absolute positions correspond to smaller relative positions and conversely, larger absolute positions to larger relative positions:

```
A.id <> B.id AND A.headID=B.headID ==>
    [A.rp <> B.rp AND [[A.rp<B.rp AND A.ap<B.ap] OR
    [A.rp>B.rp AND A.ap>B.ap]]]
```

No head cycles:
Let `NoHeadCycle(Child,A)` be defined recursively as `Child.id <> A.headID; if A.headID>0 then NoHeadCycle(Child,A.headID)` then `NoHeadCycle(Child,Child)`

Structural Constraint The structural constraint for a sentence uses both projectivity and probabilistic information to define well-formedness, disallowing structures likely to be misinterpreted. It can be summarized as follows: for any node A positioned between a node Child and its head Head, it must be more more likely that Child attaches to Head than to A (or otherwise A would likely be misinterpreted as the the head of Child by a reader), *and* Head must be an ancestor of A.
More precisely, let `IsAncestor(A,Child)` be a predicate that returns true if A is an Ancestor of Child and returns false otherwise. Also let `JointHCLikelihood(Head,Child)` be a function that computes the joint probability of Head and Child. Then the constraint is:

```
Let Left=Min(Child.ap, Head.ap), Right=Max(Child.ap, Head.ap) then
Left < A.ap < Right ==>
  JointHCLikelihood(Head,Child)> JointHCLikelihood(A,Child) AND
  IsAncestor(Head,A)
```

This well-formedness constraint still needs to be tested against a corpus of actual sentences, and would likely benefit from some refinements, not to mention extensions for non-projectivity. The JointHCLikelihood function also needs further refinement. Currently we just define it over the role and group-type features of both head and child, which was adequate for the experiment in this paper.

Feature Scores and Sentence Cost With each of the features *grouptype*, *position direction*, *relative position*, and *role* of every node we associate a conditional log probability score. This score is equal to 0 (the log of 1) if the value of the feature is already given in the input. Otherwise it is equal to logprob(Feature—Parents). "Parents" in this context refers to statistical parent, which often, but not necessarily always, includes features associated with the structural head of a node. To comply with Mozart's implementation constraints, the log probabilities are converted to negative log likelihoods, multiplied by 10000, rounded, and then converted to integers. These feature scores are combined to compute a likelihood score for a whole sentence. The score for the whole sentence is interpreted as a cost to be minimized, and is the main constraint used to determine the optimal solution for an input.

The potential parent features of each feature f in a node have been provisionally defined as shown in Table 3. To provide flexibility in the order in which features can be determined, the parents of a feature are adjusted at runtime according to the relative order in which they are determined. So the features actually used as parents of f are a subset of those in the table. Whether a parent feature is used for conditioning depends on whether the feature is associated with the head of the node or with the node itself.

Feature	*Interdependencies*
group type	head: group type; self: role, position direction, relative position
role	head: role; self: group type, position direction, relative position
position direction	self: role, group type
relative position	self: role, group type; head: group type

Table 3. Statistical Feature Dependencies

The features associated with the head are always used if the ID of the head is known (which it is in the generation task but not the parsing task). The features associated with the node itself are used only if they are determined before f itself is. The table is defined symmetrically so that for each parent feature p of f, f is also listed as one of the parents of p. Thus, this procedure amounts to dynamic variation in the way a node's joint probability is decomposed, but since all the possible decompositions are mathematically equivalent, the node's score remains consistently defined.

For example, suppose that for a particular node its role feature is given in the input (as it is in our experiments described later), and that the three other statistical features are determined in this order: position direction, relative position, and group type. Then the score for that node is computed as

```
NodeScore = FeatScore(role) + FeatScore(pd) + FeatScore(rp) + FeatScore(gt)
          = 0 + logprob(pd|role) + logprob(rp | role, pd, h_gt)
              + logprob (gt | h_gt, role, pd, rp)
```

2.3 Distribution Strategy and Search Algorithm

In Mozart the distribution strategy is independent from the search strategy. In other words, the shape of the search space is specified separately from the order in which subparts of the space are searched. For distribution, we define a procedure that selects from among the variables associated with statistical scores the variable with the

greatest number of suspensions. When those are all determined it then selects absolute position or head id variables based on smallest current domain size. The reason for this two-stage method is that absolute position and head id can be almost completely determined from the other features. Delaying them until last reduces the search space considerably. (Treating them the same as the others results in "out of virtual memory errors".)

Distribution on domain values of variables that are associated with probabilistic scores is done in order of increasing conditional likelihood of the values, given the same context that is used for calculating feature scores. In other words, the most likely value is searched first. Values that haven't been seen in the given context are pruned, as long as there was at least one seen value for that context. The values of head id variables are ordered to try first the nearer nodes (in terms of absolute position). Absolute position is distributed naively.

For search we use the built-in branch-and-bound algorithm provided by the Explorer program. The solution ordering constraint used to prune the search space is simply that the total cost of a new solution must be better than or equal to the cost of the previous best solution.

3 Experiments

We performed our experiments using the classic sentence "Time flies like an arrow." The complete solution for this sentence with respect to the features we are using is shown in Table 4. Note that there is no element of our program that is tied/hardwired to this particular input.

Tok	Role	Group type	Absolute position	Relative position	Position direction	ID	Head id
time	subject	np	1	-1	-	4	1
flies	top	clause	2	0	+	1	0
like	role-marker	other	3	-2	-	6	3
an	determiner	other	4	-1	-	5	3
arrow	adjunct	np	5	1	+	3	1
.	right-punc	other	6	2	+	2	1

Table 4. Solution Sentence

Table 5 summarizes the experiments we performed on the sample sentence as well as the results. The first four experiments represent subproblems of parsing, and the last four are subproblems of generation. Each line of the table has a different combination of given versus unknown variables, organized in order of increasing difficulty within each type of subproblem.

The time figure is the one reported by Mozart's Explorer, but excludes garbage collection and time spent on database queries. (Our implementation includes three levels of caching for database queries, and the actual real time varied from less than a second to less than half a minute for the same input depending on the contents of the cache.) The third column reports the size of the search tree in terms of search nodes.

Given VS. Unknown	Time	Search Tree Size	Depth	Correct
role, ap, dir, rp VS. head	20ms	12	6	yes
role, ap, dir, rp VS. head, gt	30ms	28	10	yes
role, ap, dir VS. rp, head, gt	210ms	209	15	yes
role, ap VS. dir, rp, head, gt	320ms	259	17	yes
role, head, dir, rp VS. ap	10ms	3	2	yes
role, head, dir, rp VS. ap, gt	40ms	21	6	yes
role, head, dir VS. rp, ap, gt	130ms	71	9	yes
role, head VS. dir, rp, ap, gt	420ms	194	11	yes

Table 5. Exploratory Experiments

The Depth column is the depth of the search tree. The last column shows that our program always arrived at the correct solution. This fact is our most interesting result.

These experiments also help indicate how well the constraint program scales as the complexity of the problem is increased. From the time and search tree sizes, the program seems to scale adequately enough to merit further exploration of the CCP framework, especially since our implementation can quite probably be tuned to improve these figures.

To give more of the flavor of how our approach works, Table 6 shows the relative likelihood of the top values in the domains of each of the probabilistic variables given just the role. The values are listed in order of decreasing likelihood, so the most likely one is first. A value that does not appear in the table was not among the top seven values for that feature. The flag points out cases where the first value listed was not the correct answer for our sample sentence. Note that during runtime the actual conditioning features used can result in a different relative ordering than what is shown in the table.

4 Future Directions and Related Work

The approach described in this paper is still quite preliminary. Work remains to add a fuller linguistic representation, analyze the true statistical relationships between features, and refine various parts of our formulation. In particular, the propagation in our current representation is weaker than it could be. An approach using selection constraints as in [8] would be better. Our structural well-formedness constraint is also a prominent and very interesting area for further exploration. We wish to extend it to handle non-projectivity using probabilistic information. Our distribution and search strategies could also benefit from further study, as well as (of course) our overall linguistic model in terms of feature/value sets. Finally, an empirical evaluation against a test corpus of sentences is also needed. However, the initial results are very encouraging: we produce the correct result in roughly real time on a sample sentence using less than 400 lines of Mozart code (excluding comments).

Note that probabilities are applied in three different ways in our formulation: as a hard constraint in defining legal dependency structures; as a soft constraint on the cost of a sentence—used both to find the optimal solution and to prune the search space; and thirdly, implicitly as part of the search strategy when choosing which value to try first for probabilistic variables. This goes beyond any other work we know of in leveraging probabilistic information. The work of [9] is the most closely related in this

Feature	Word/Role	Values	Flag
position direction	time/subject	'-', '+'	
	flies/top	'+'	
	like/role-marker	'-'	
	an/determiner	'-'	
	arrow/adjunct	'-', '+'	*
	./right-punc	'-'	
relative position	time/subject	-1, -2, -3, +1, -4, -5, +2	
	flies/top	0	
	like/role-marker	-2, -1, -3, -4, -5, -6, -7	
	an/determiner	-1, -2, -3, -4, -5, -6, -7	
	arrow/adjunct	-1, +1, -2, +2, -3, +3, -4	*
	./right-punc	+2, +1, +3, +4, +5, +6, +7	
group type	time/subject	np, other, clause	
	flies/top	clause, np, other	
	like/role-marker	other, np, clause	
	an/determiner	other	
	arrow/adjunct	other, np, clause	*
	./right-punc	other	

Table 6. Relative likelihood of probabilistic domain values

aspect: it describes using an A* heuristic for search in a constraint program, although the implementation does not seem to have been done yet. Also related is the work on graded constraints in [10].

Since we use the same program to solve subproblems of both parsing and generation, our work is similar to research on bi-directional grammars. In contrast to that research, however, our formulation of the problem does not involve an explicit grammar (or lexicon) at all. Instead it relies on a database of annotated sentences derived from the English Penn Treebank [11,12]. From this database, our program can readily obtain frequency information for a variety of syntactic patterns, and use this information to solve for unknown feature values. The benefit of our approach is avoidance of the laborious process of developing linguistic resources manually.

A constraint programming formulation of the parsing and generation problem makes the relationship between the two inverse faces of NLP very clear. It even makes it possible to blend the two types of problems in ways that are more useful in practice, making an NLP system overall more versatile than traditional parsers and generators. Doing so requires a flexible architecture that does not predetermine the order in which processing is done. Our initial work in this paper gives us hope that concurrent constraint programming can be the foundation for an efficient architecture with this flexibility.

5 Acknowledgements

Much thanks goes to the anonymous reviewers who provided many helpful comments.

References

1. Smedt, K.D., Horacek, H., Zock, M.: Architectures for natural language generation. In: Trends in Natural Language Generation. Springer, Berlin (1996)
2. Cahill, L., Doran, C., Evans, R., Mellish, C., Paiva, D., Reape, M., Scott, D., Tipper, N.: In search of a reference architecture for NLG systems. In: Proc. EWNLG. (1999)
3. Cahill, L., Doran, C., Evans, R., Kibble, R., Mellish, C., Paiva, D., Reape, M., Scott, D., Tipper, N.: Enabling resource sharing in language generation: an abstract reference architecture. In: Proc. LREC. (2000)
4. Cahill, L., Evans, R., Mellish, C., Paiva, D., Reape, M., Scott, D.: The RAGS reference manual. Technical report, Univ. of Brighton and Univ. of Edinburgh (2002)
5. Reiter, E.: NLG vs. templates. In: Proc. ENLGW '95. (1995)
6. Busemann, S., Horacek, H.: A flexible shallow approach to text generation. In: Proc. INLG Workshop. (1998)
7. Pianta, E., Tovena, L.M.: Mixing representation levels: The hybrid approach to automatic text generation. In: Proc. of the AISB Workshop on "Reference Architectures and Data Standards for NLP". (1999)
8. Duchier, D.: Configuration of labeled trees under lexicalized constraints and principles. Journal of Research on Language and Computation (2003)
9. Dienes, P., Koller, A., Kuhlmann, M.: Statistical a-star dependency parsing. In: Prospects and Advances in the Syntax/Semantics Interface. (2003)
10. Schroder, I.: Natural Language Parsing with Graded Constraints. PhD thesis, University of Hamburg (2002)
11. Marcus, M., Santorini, B., Marcinkiewicz, M.: Building a large annotated corpus of english: the Penn treebank. Computational Linguistics **19** (1993)
12. Marcus, M., Kim, G., Marcinkiewicz, M., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., Schasberger, B.: The Penn treebank: Annotating predicate argument structure. In: ARPA Human Language Technology Workshop. (1994)

A Constraint-Based Model for Preposition Choice in Natural Language Generation

Véronique Moriceau and Patrick Saint-Dizier

Institut de Recherches en Informatique de Toulouse, IRIT,
118 route de Narbonne, 31062 Toulouse Cedex, France.
moriceau@irit.fr,stdizier@irit.fr

Abstract. In this paper, we show how a constraint-based approach influences the modelling of preposition lexicalization in natural language generation. We concentrate on the linguistic description, which is the most challenging. The CSP procedures themselves are then rather straightforward. Preposition choice depends on the verb and its requirements, on the one hand, and the characteristics of the NP the preposition heads, on the other hand. These dependencies may induce somewhat contradictory expectations. A Constraint-based approach introduces a declarative model of this complex relation, allowing to identify all the possible lexical choices which can be predicted from lexical descriptions.

1 Introduction

With the development of WEBCOOP (Benamara et al. 2003), a system that produces cooperative answers to queries, the production of accurate natural language (NL) responses becomes essential. This implies great care in the taking into account of both user parameters (e.g. the terms used in the query) and linguistic constraints to produce a response which is accurate in its contents and as fluid and as well-formed as possible from a language point of view.

1.1 WebCoop output forms

The outputs of the WEBCOOP reasoning component are logical formula that encode conceptual representations of a fine-grained level. These representations are true decompositional semantics representations; they are language independent, allowing thus a quite large freedom to the NL generator, so that it can take into account a variety of parameters, such as: user prefered terms, homogeneity of style, complexity of structures, etc. Representations are a conjunction of several types of predicates:

- those that type objects or variables (for example, *flight(X), city(paris)*),
- those that denote relations (such as prepositions: *from(loc,X,Y)*) and
- those that describe events (*visit(E,X,Y)*).

As an illustration, the output of WEBCOOP that says that *all hotels in Cannes have a swimmingpool* is:
$hotel(X) \wedge in(loc, X, cannes) \wedge equipment - of(X, Y) \wedge swimmingpool(Y)$
Within such a decompositional approach, preposition choice is, in general, really difficult to handle. The choice of the preposition *in* in the above example is not straightforward. Most prepositions are indeed heavily polysemic, making lexical choices highly contextual. Elaborating constraints to model lexical choice performances realized by humans requires a detailed analysis of the parameters at stake and how they interact. This is the main aim of this contribution.

1.2 Facets of lexicalization

Lexicalisation is the operation that associates a word or an expression to a concept. It is a major parameter in response production (Reiter and Dale, 1997), (a good synthesis can be found in (Cahill, 1999)). Lexicalisation is often decomposed into two different stages: **lexical choice**, which occurs during content determination, where the lexical term chosen, which may still be underspecified, is dependent on reasoning procedures, the knowledge base contents, and grammatical constraints; and **lexical variation** which is the choice of a particular word or form among possible synonyms or paraphrases. Lexical variation occurs in the surface realizer, and may have some pragmatic connotations, for example an implicit evaluation via the language level chosen (e.g. argotic entails low evaluation). In this paper, we are mainly concerned with lexical choice. Some simple forms of lexical variation occur when two or more prepositions have exactly the same behavior w.r.t. to the lexical description, which is rather unfrequent.

1.3 Scope of the investigation

In this paper, we propose a declarative model that handles the complex relation verb-preposition-NP and a correlate, the relation deverbal noun-preposition-NP, which has a quite similar behavior. We concentrate on the linguistic description, which is by far the most challenging. This modelling and the subsequent NL performances are obviously heavily dependent on the accuracy and the adequacy of the linguistic descriptions provided, in particular lexical, including a number of usage variations like metaphors, which abound in preposition uses. The power of our model mainly lies in the way linguistic descriptions are organized and processed, in a way that avoids making too early and definitive choices. Thus, in our approach, CSP brings a new way (1) to organize and formulate lexical structures so that they can be adequtely used, and (2) then to state co-occurence constraints between sets of potential candidates.

We investigate PP constructions which are compositional. We also consider verb constructions where the verb incorporates the preposition, as, e.g. *climb* which incorporates the preposition *up* by default. This verb requires an explicit preposition otherwise, as in *climb down*. We consider that in this type of construction the preposition is both part of the verb (as a particle) and of the head of the localization NP that follows. We exclude true verb-particle constructions,

however not common in Romance languages, but frequent in Germanic languages (e.g. move up, clear up), and fixed forms (e.g. boil down), which are best treated as single units.

Our study being carried out on French, most of our examples are in French with approximate English glosses.

2 The conflictual relation verb-preposition-NP

2.1 A motivational example

Let us first illustrate the problem by a simple example. Linguistic notions involved being quite complex, these will just be presented here, without any deep analysis and justifications. A more detailed analysis is given gradually in the next sections.

The formula:
$person(jean) \wedge sea(X) \wedge to(loc, jean, X)$,
which is a simplified form of the Lexical Conceptual Structure (Jackendoff 1990) used in WEBCOOP, conveys the idea of someone (jean) going to somewhere (the sea). If we want to construct a proposition out of this formula, then the expression R = $to(loc, jean, X)$ is selected as the only predicate that can be realized as a verb. This form is quite abstract, it characterizes a priori any verb from the class 'movement verb to a certain location', movement being induced from the conceptual domain 'loc'. Verbs of this class are, for example: *aller, marcher, se déplacer, monter, descendre, pousser*, etc. (go, walk, move, ascend, descend, push), most of which incorporate manners and/or means. The same expression, R also originates the production of the preposition required by the verb for its object, of type fixed localization (Cannesson et al. 2002). The two arguments required by the verb are a priori quite neutral, typed 'human' for *jean* and 'place' for *sea*.

The choice of the preposition depends on the verb, but also on more pragmatic factors. Let us examine the above example in more depth. Since there are no explicit manners or means in the formula, let us select a verb which is as neutral as possible. In the lexicon, a verb such as *aller* (go) basically requires a preposition of type 'fixed position' since *aller* already incorporates the notion of movement, but not all such prepositions are acceptable, e.g. *à* is acceptable but not *pour*. In contrast, a verb like *partir* accepts both. *Marcher* would rather select *sur, dans, à coté de* as fixed positions, characterizing the space in which movement occurs. This situation is not proper to movement verbs, in (Mari and Saint-Dizier 2003), we show that the same kind of problems are raised by prepositions denoting instrumentality.

2.2 Stability over deverbals

Note that, although there is a quite large stability, prepositions acceptable for the relation verb-preposition-NP are not systematically acceptable for the relation

deverbal noun-preposition-NP although the semantics of the phrase is the same. For example, the preposition *pour* is not acceptable in ** l'avion vole pour Paris* (the aircraft flies to Paris) whereas it is acceptable in *le vol pour Paris est complet* (the flight to Paris is full) - the semantics of both phrases conveyed by the preposition semantic representation being the same.

Conversely, the preposition *à* (to) is acceptable in *le vol va à Paris* (the flight goes to Paris) whereas it is not in ** le vol à Paris est complet* (the flight to Paris if full). In fact, the verb *aller* (go) incorporates a notion of destination which explains why the preposition *pour*, denoting a notion of trajectory towards a goal, is not acceptable because, among other considerations, of the redundancy it introduces. The preposition *à* is more neutral in this respect and can be combined with *aller*. In the case of the relation deverbal noun-preposition-NP, the preposition used must denote a destination this is why only *le vol pour Paris* is acceptable. Here *vol* simply denotes the event of a flight, it does weakly incorporate a notion of trajectory, but this notion needs some further lexicalization, as also shown in: *a climb up the hill*.

2.3 Derived uses

In addition, prepositions that indicate a direction, which can be interpreted (as a kind of metonymy) as denoting an area relatively well delimited in the far, are also acceptable: *aller vers la mer* (go towards the sea). We call this phenomenon semantic coercion. Finally, a number of movement verbs also accept a more complex object structure, of type trajectory, including source and vias, but this structure often needs to be fully instantiated.

Note that depending on the object type and on the type of movement induced by the verb, other fixed position prepositions are possible: *monter sur la chaise* (to go on the chair), *aller sous le pont* (to go under the bridge). The choice of the most prototypical preposition (which does not exclude others) crucially depends on geometrical parameters of the object, or on prototypical uses. Finally, pragmatic factors may interfere (pragmatic coercion). For example if Jean is far from the sea, a preposition denoting a destination, interpreted as defining an area, is possible: *marcher vers la mer* (walk towards the sea), whereas it is not so appropriate if Jean is close to the sea.

Movement verbs, in particular, are subject to a large number of metaphorical uses on their object PP: *rentrer dans un projet (to enter into a project), passer sur un problème (to pass over a problem), monter dans les sondages (to rise in polls)*. In that case, the preposition choice heavily depends on the metaphorical interpretation of the object. For example, a project as an activity to be in, similarly to a place, or polls as a ladder to climb. We treat metaphors as a type coercion operation that changes the type of the object NP into a type appropriate for the preposition, that translates the meaning of the PP (Moriceau et al. 2003). In this paper, we consider the result of such type coercion operations, assuming they are specified in the system, in conjunction with the lexicon.

2.4 Cocomposition

A final point in the interaction verb-preposition-NP is the mutual compositional influence of the verb on the PP and vice-versa, called 'co-composition' in the generative lexicon (Pustejovsky 1995). This phenomenon remains compositional in its essence even if it requires non monotonic treatments of semantic representations. A case such as *aller contre un principe* (to go against a principle) preserves the idea of progression of the verb while *contre* overrides the fixed position naturally expected as the PP of the verb, to introduce a complex meaning which can be schematized as Opposition to a fixed, abstract position (here, a principle). Co-composition is still quite an open research field, essentially linguistic, which we will not discuss here.

3 Constraint domains

In this section, we present in more depth an analysis of the verb-preposition-NP complex relation, modelling constraints imposed by each protagonist in terms of domains of potential realizations. The basic idea is to produce sets of structures in which each element includes all the parameters necessary for expressing constraints and for making appropriate lexicalizations. In section 4, we show how constraints expressed as equations can be stated, so that a priori all potential acceptable preposition lexicalizations can be selected. Lexicalization decisions are thus delayed as much as possible avoiding useless backtracking and also allowing for the taking into account of long-distance dependencies.

Starting from semantic representations, let us first investigate the domains associated with each element in the formula, that corresponds to the following constructs to be determined and lexicalized: the verb (or the deverbal), the preposition and the NP. In section 4, we show how they interact.

3.1 Verbs

Our description of French verbs (Saint-Dizier 1998) heavily relies on the conceptual categories defined in WordNet. implemented a three level classification which seems to be sufficiently accurate to deal with preposition choice, as far as the verb is concerned. For example, for movement verbs, we have the following hierarchy:

1. 'local movement' (dance, vibrate)
2. 'movement': specified arrival, specified departure, trajectory (go)
3. 'medium involved': land, water, air/space, other types of elements (swim, skate)
4. 'movement realized by exerting a certain force': willingly or unwillingly (push)
5. 'movement with a certain direction': upwards, downwards, forward, backward, horizontal (rise, ascend, climb)
6. 'movement accompanied' (carry)

7. 'beginning/stopping/resuming/amplifying a movement' (stop, accelerate)
8. 'activity related to a movement' (drive).

We have a total of 292 verb classes organized around 17 main families borrowed from WordNet (e.g. verbs of body care, possession, communication, change, etc.). Each lower level class is associated with a by-default subcategorization frame (corresponding to the prototypical verb of the class) that describes the structure of the verb arguments, in particular the type of the preposition and of the argument normally expected. Each class also receives a by-default LCS representation, which can then be adapted to each verb, to capture some sense variations or specializations. For example, the movement verb with specified arrival subclass is represented as follows:

```
[ class: movement with specified arrival
prototypical verb: aller (go)
subcat: [X: NP(human), Y: PP(localization)]
lcs: to(loc, X,Y) ]
```

As defined in (Cannesson et al. 2001), `to` is a primitive that potentially covers a number of surface realizations.

Verbs often occur in different classes, even for a given sense: within a generative approach like ours, this characterizes the different facets the verb may take, as well as some of its frequent metaphoric uses. In fact, some very frequent metaphoric uses may get the status of a specific sense, with its own autonomy.

Verbs that incorporate a preposition by default (e.g. *climb, enter*) are marked as such in their subcategorization frame, informally:

```
enter: [NP, PP(prep: type: 'inside', incorporated]
```

resulting, in by-default situations, in the non-surface realization of the preposition. This phenomenon is implemented as a verb local constraint, and does not alter our model, since it is a lexical constraint.

From an NL generation point of view, starting from a semantic representation such as $to(loc, jean, X)$ we get a set of verb lexicalizations (or equivalent deverbal nouns) or a more intentional specification such as a verb class. This is obtained via unification or subsumption of the representations in individual verb lexical entries or in the semantic representation specified at verb class level with the semantic representation R. The domain induced is set of tuples verb(A,B,C,D) that meet the subsumption constraint. Domain is the following:

$$Domain(verb(R)) = \{\cup\, verb(VClass, RR, scat(Type-Prep,\ Type-PP),\ Lexicalization) \wedge subsumes(RR, R)\}$$

where, in the verb lexical entry, RR is the verb semantic representation in the formula subsumed by R (subsumption in LCS is defined in (Saint-Dizier 2001)), which originates the construction of the domain. VClass is the verb class, Type-PP is the type normally expected for the object PP, and scat contains the subcategorization information. We have, in our example, the abstract form R = $to(loc, Y, X)$ and the set of verbs of the class 'movement, specified arrival', associated with their subcategorization frame, which is constructed via unification

(see above example). The same situation occurs with the corresponding deverbal nouns.

3.2 Nouns and NPs

The head noun of the NP must meet in some way with the verb subcategorization frame expectations. The domain of the head noun is its semantic type (and all its descendent, via subsumption in the domain ontology). Derived types may come from possible metaphorical and metonymic uses, implemented in our framework (Moriceau et al. 2003) via type coercion rules.

Type coercion Type coercion rules are modelled globally by means of constrained rewriting rules. They implement various forms of metaphors and metonymies. Let Σ be the set of such coercion rules σ, functions of the verb class and of the semantic types expected for the object PP:
$derived - type = \sigma(verb(VClass, Type - PP))$,
and let TC be the transitive closure of this operation on the verb at stake (i.e. the set of all elements which can be derived). TC is finite and is defined via recursive enumeration. In our approach, derived types are not so numerous: there is no recursive application of type coercion, resulting in a quite small and stereotyped set of derived types.

For example, a frequently encountered metaphorical construction such as *entrer dans un projet* (to enter in a project), where project is typed as 'epistemic artefact' requires a type coercion rule of the form:
σ_i: epistemic artefact = σ(verb(enter,localization)).
Coercion rule is here restricted to the subclass of 'enter' verbs, since the metaphor is not fully regular for all movement verbs with specified arrival. This means that an accurate description of metaphors involves a large number of rules if one wants to avoid overgeneration, a major problem in NL generation.

Then, if R1 is the semantic representation of the noun, the domain of the NP is therefore a set of triples of the form:
$Domain(NP(R1)) = \{\cup\ noun(R1, head - noun - type, Lexicalization),$
$noun(\_, derived - type, Lexicalization)\}$

3.3 Prepositions

Most prepositions are highly polysemic and are often involved in a large number of derived, unexpected or metaphorical uses. Analyzing and representing the semantics of prepositions and generating appropriate prepositions in natural language generation is a rather delicate task, but of much importance for any application that requires even a simple form of understanding. Spatial and temporal prepositions have received a relatively in-depth study for a number of languages (e.g. (Verkuyl et al. 1992)). The semantics of the other types of prepositions describing manner, instrument (Mari and Saint-Dizier 2003), amount or

accompaniement remain largely unexplored. In this section, for readability purposes, we introduce some background on preposition semantics, mainly on a general classification and semantic description we carried out a few years ago (Cannesson et al. 2002). Work is on French, but a number of elements are stable over a variety of languages, in particular Romance languages.

Identifying preposition senses Although prepositions have a number of idiosyncratic usages (probably much less in French than in English), most senses are relatively generic and can be characterized using relatively well-known and consensual abstract labels, as shown in (Cannesson et al. 2002) for French. To illustrate this point, let us consider the case of *par*. *Par* has the following 6 senses, which seem to be all approximately at the same level of abstraction:

- distribution: *il gagne 1500 Euros par mois* (he earns 1500 Euros per month),
- causality: as in passives but also e.g. in *par mauvais temps, je ne ne sors pas* (by bad weather I don't go out),
- origin: *je le sais par des amis* (I know it from friends),
- via: *je passe par ce chemin* (I go via this path),
- tool or means: *je voyage par le train* (I travel by train),
- 'approximate' value: *nous marchons par 3500m d'altitude* (we hike at an altitude of 3500m).

In a second stage, we have provided a conceptual representation of these senses, based on the Lexical Conceptual Structure (LCS, Jackendoff 90), which is based on a language of primitives, viewed as linguistic macros, which can be interpreted in various frameworks, such as the Euclidean geometry. The LCS also introduces a decompositional approach to meaning which allows for the development of an accurate and abstract theory of lexicalization, in particular for prepositions, which it represents particularly well.

A general typology for prepositions Here is a first classification proposal for French prepositions (see also (Cannesson et al. 2002)). We have identified three levels of decomposition which are quite regular and have an internal stability: family, facet and modality. Only the two first levels are introduced here. Labels for semantic classes are intuitive and quite often correspond to thematic role names (examples are direct translations from French), these are the labels specified in verb subcategorization frames:

- **Localization** with facets: **source, destination, via/passage, fixed position**. Destination may be decomposed into destination reached or not (possibly vague), but this is often contextual. Fixed position can either be vague (*he is about 50 years old*) or precise. From an ontological point of view, all of theses senses can, a priori, apply to spatial, temporal or abstract arguments.
- **Quantity** with facets: **numerical or referencial quantity, frequency and iterativity, proportion or ratio**. Quantity can be either precise (*temperature is 5 degrees above 0*) or vague. Frequency and iterativity: *he comes several times per week.*

– **Manner** with facets: **attitudes, means (instrument or abstract), imitation or analogy**. Imitation: *he walks like a robot; he behaves according to the law.*
– **Accompaniement** with facets: **adjunction, simultaneity of events, inclusion, exclusion**. Adjunction : *flat with terrace / steak with French fries/ tea with milk*, exclusion : *they all came except Paul.*
– **Choice and exchange** with facets: **exchange, choice or alternative, substitution**. Substitution : *sign for your child*, choice: *among all my friends, he is the funniest one.*
– **Causality** with facets **causes, goals and intentions**. Cause: *the rock fell under the action of frost.*
– **Opposition** with two ontological distinctions: physical opposition and psychological or epistemic opposition. Opposition: *to act contrary to one's interests.*
– **Ordering** with facets: **priority, subordination, hierarchy, ranking, degree of importance**. Ranking : *at school, she is ahead of me.*
– Minor groups: **About, in spite of, comparison**. The terms given here are abstract, and cover several prepositions. About: *a book about dinosaurs.*

Each of the subsenses described above is associated with a number of preposition senses, clearly distinct from other senses. Here is a brief description of the Ordering class:

Fig. 1 - prepositions of Ordering family

facet	prepositions
Priority	après (*after*), avant (*before*)
Subordination	sous (*under*), sur (*above*)
Hierarchy	devant (*in front of*), derrière (*behind*) avant (*before*), après (*after*)
Ranking	devant (*in front of*), derrière (*behind*)
Degree of importance	à côté de, auprès de (*close to, near*), par rapport à, pour, vis-à-vis de (*compared to*)

A general conceptual semantics for prepositions Each preposition facet or modality has a unique representation. For example, 2 major senses of the preposition *avec* (with) are:

– **accompaniement**, represented as, in the simplified LCS representation we developed:
with(*loc*, *I*, *J*),
where 'loc' indicates a physical accompaniement (*I go to the movies with Maria*), while 'psy' instead of 'loc' would metaphorically indicate a psychological accompaniement (*Maria investigated the problem with Joana*).

– **instrument**, represented as:
$by-means-of(manner,\ I,\ J)$
(*they opened the door with a knife*). This is, in fact, a generic representation for most preposition senses introducing instruments, a more refined analysis can be found in (Mari et al. 2003).

In our framework, we defined 65 primitives encoding 170 preposition senses, which seem to cover most senses found in standard texts over a number of languages. Language being essentially generative and creative, this obviously does not exclude other senses, for which, most probably, a few additional primitives, or the composition of already defined ones, will be necessary.

Semantic and pragmatic coercion Semantic and pragmatic coercion on the type of preposition expected by the verb can be modelled as follows. Let Υ be the set of such coercion rules τ, functions of the verb class and of the semantic class of the preposition:
$derived-subclass = \tau(verb(VClass, Type-Prep))$,
For example, *parler sur le vague* (litt. to talk on vagueness) has a PP characterizing a topic of conversation. The default preposition in French, specified in the lexical entry of *parler* is *de* (about). The preposition *sur* introduces the idea of a general discussion, we have then the following pragmatic coercion rule:
$'sur' = \tau(verb(talk,'about'))$.
We limit here preposition classes to a precise preposition, to avoid potential overgeneration.

From a more formal point of view and more generally, let TT be the transitive closure of this operation on the verb at stake (i.e. the set of all subclasses which can be derived). TT is finite and is defined via recursive enumeration from coercion rules and lexical descriptions. TT defines the semantic and pragmatic expansion of the verb-preposition compound.

Back to our example, the preposition is primarily induced by the semantic representation R2 = $to(loc, Y, X)$. More generally, the domain of potential lexicalizations is characterized by a set of triples as follows:

$$Domain(prep(R2)) = \{\cup\ (prep(Subclass, RestrNP, RR2, Lexicalization) \wedge subsumes(RR2, R2),\ prep(derived-subclass, \_, \_, Lexicalization)\ \}$$

where, in the lexical entry 'prep', Subclass designates the modality or the facet level as exemplified above, RestrNP are the selectional restrictions imposed on the NP headed by the preposition, RR2 is the semantic representation which must be subsumed by R2, and Lexicalization is the surface realization(s) of RR2 (this implements lexical variation, when there is a strict lexical equivalence). The second set of triples prep is the transitive closure (produced by recursive enumeration) of all potential type coercions given the verb identified or its class.

4 Preposition Lexicalization as a CSP

In the previous section, we show how domains of lexical entries and associated lexicalizations can be constructed, via unification, subsumption and type, semantic and pragmatic coercion. The challenge is now to express and to manage the constraints of the triple verb-preposition-NP, so that all possible preposition lexicalizations can be made explicit, allowing for the generation of the VP.

4.1 Constraints between domains as equations

Let us first recall the 3 domains defined above:
$Domain(verb(R)) \ = \ \{\cup\ verb(VClass, RR, scat(Type - Prep,\ Type - PP),$
$\quad Lexicalization) \wedge subsumes(RR, R)\}$
$Domain(np(R1)) \ = \ \{\cup\ noun(R1, head - noun - type, Lexicalization),$
$\quad noun(\_, derived - type, Lexicalization)\}$
$Domain(prep(R2)) \ = \ \{\cup\ (prep(Subclass, RestrNP, RR2, Lexicalization)$
$\quad \wedge subsumes(RR2, R2)),\ prep(derived - subclass, \_, \_, Lexicalization)\ \}$

From these specifications, it is possible to state constraints under the form of equations, from which pairs:
(verb lexicalization, preposition lexicalization)
can be produced, assuming the noun has a fixed type, and a limited number of lexicalizations which cannot a priori be revised. We have the following equations:

1. Type-PP subsumes (a) head-noun-type or (b) an element s in the derived types via type coercion. In more formal terms:
   $subsumes(Type - PP, head - noun - type) \vee\ \exists \sigma \in \Sigma,$
   $\quad s = \sigma(Type - PP) \wedge subsumes(Type - PP, s)).$
2. RestrNP subsumes head-noun-type or s:
   $subsumes(Restr - NP, head - noun - type) \vee subsumes(Restr - NP, s).$
3. Type-Prep subsumes (a) Subclass in direct usages or (b) a derived Subclass via semantic or pragmatic coercion:
   $subsumes(Type - Prep, SubClass) \vee\ \exists \tau \in \Upsilon,$
   $\quad t = \tau(Type - Prep) \wedge subsumes(Type - Prep, t)).$

The satisfaction of these equations produces the lexicalization set composed of pairs (verb, preposition). Lexical choice can operate on these pairs to select one of these, possibly e.g. on the basis of user preferences. Preposition incorporation into verbs is also handled at this level.

As explained in the introduction, provided restrictions in various lexical entries are adequately described, and that type, semantic and pragmatic coercion rules are appropriate, our system does not overgenerate. Overgeneration is solely due to incomplete or inadequate linguistic descriptions. A priori, our system is sound and complete and produces all the admissible solutions, w.r.t. lexical specifications and coercion operations.

4.2 A direct illustration

Let us illustrate the satisfaction of these constraints by a simple example:
$flight(5564) \wedge to(loc, 5564, Paris) \wedge city(Paris)$,
where *to(loc,5564,Paris)* is at the same time R (the semantic representation of the verb) and R2 (the semantic representation of the preposition); *city(Paris)* is R1 (the semantic representation of the noun of the PP).
The three domains are:

$$\begin{aligned} Domain(verb(R)) \;=\; \{\; & verb(movement-verb-to-destination, to(loc, X, Y), \\ & \quad scat(fixed-position,\ fixed-position), aller), \\ & verb(movement-verb-to-destination, to(loc, X, Y), \\ & \quad scat(fixed{-}position,\ fixed{-}position), monter), \\ & verb(movement-verb-to-destination, to(loc, X, Y), \\ & \quad scat(fixed{-}position,\ fixed{-}position), arriver), \\ & ... \\ & \wedge\ subsumes(to(loc, X, Y), to(loc, 5564, Paris))\} \end{aligned}$$

$$\begin{aligned} Domain(np(R1)) \;=\; & \{\; noun(city(Paris), city, Paris)\} \\ Domain(prep(R2)) \;=\; & \{\; prep(fixed{-}position, fixed{-}position, to(loc, X, Y), à), \\ & prep(destination, fixed-position, to(loc, X, Y), pour), \\ & prep(destination, fixed-position, to(loc, X, Y), en), \\ & \wedge\ subsumes(to(loc, X, Y), to(loc, 5564, Paris))\ \} \end{aligned}$$

Let us apply the constraints:

1. $subsumes(Type-PP, head-noun-type)$
   $\rightarrow\ subsumes(fixed-position, city)$
2. $subsumes(Restr-PP, head-noun-type)$
   $\rightarrow\ subsumes(fixed-position, city)$
3. $subsumes(Type-Prep, SubClass)$
   $\rightarrow\ subsumes(fixed-position, fixed-position)$

The solution domain is: {*(aller,à),(monter,à),(arriver,à), ...*} .

With respect to the lexical descriptions, these three solutions are acceptable, although *monter à* is les natural for a flight, than, e.g. for a car.

4.3 More complex situations

Our model deals with composition (or co-compositional) forms. It naturally discards constructions which do not meet the selection and possibly coercion constraints. For example, let $person(jean) \wedge to(loc, jean, Y) \wedge wall(Y)$ be the semantic representation of the metaphorical semi-fixed form: *Jean part dans le mur* (*Jean goes in the wall = john fails (in an enterprise), as a car going in a wall is going to be broken)*).
In this example, *partir* is a movement verb with destination specified, *dans* can

be acceptable provided the NP has an inside into which the subject can go, which is not the case for *mur* (wall).

The three domains are (without type coercion):

$$\begin{aligned}
Domain(verb(R)) \;=\; \{\; & verb(movement{-}verb{-}to{-}destination, to(loc, X, Y), \\
& \qquad scat(destination,\; fixed-position), partir), \\
& verb(movement{-}verb{-}to{-}destination, to(loc, X, Y), \\
& \qquad scat(destination,\; fixed-position), courir), \\
& \ldots \\
& \wedge\; subsumes(to(loc, X, Y), to(loc, jean, Y))\} \\
Domain(np(R1)) \;=\; \{\; & noun(wall(Y), compact-object, mur)\} \\
Domain(prep(R2)) \;=\; \{\; & prep(destination, fixed-position, to(loc, X, Y), \\
& \qquad vers), \\
& prep(destination, fixed-position, to(loc, X, Y), pour), \\
& prep(destination, fixed{-}position, to(loc, X, Y), a\ destination\ de), \\
& \ldots \\
& \wedge\; subsumes(from(loc, X, Y), to(loc, jean, Y))\;\}
\end{aligned}$$

Let us now apply the equations between domains:

1. $subsumes(Type-PP, head-noun-type)$
   $\rightarrow\; subsumes(fixed-position, object)$
2. $subsumes(Restr-NP, head-noun-type)$
   $\rightarrow\; subsumes(fixed-position, object)$
3. $subsumes(Type-Prep, SubClass)$
   $\rightarrow\; subsumes(destination, destination)$

The solution domain is:
{*(partir, vers), (partir, pour), (partir, à destination de), ...*} .
This solution domain does not contain the expected metaphoric solution *partir dans le mur* because of the type of wall, which is not an object with an inside. This is due to the fact that the preposition used *dans* (*in*) does not belong to the preposition class *destination*.

If we now add the domain engendered by the corresponding type coercion rule:
$'dans' = \tau(verb(partir, destination))$,
assuming *destination* is the type of preposition normally expected by the verb, the solution domain now includes the form (partir, dans), *mur* (wall) being correctly typed as a fixed position. In fact, this metaphor applies to most movement verbs with specified destination, each of them making more explicit a manner of going in a wall.

Let us also note that while (partir, vers), (partir, pour) can be combined with a number of metaphors in abstract domains, (partir, à destination de) essentially expects an object NP of type physical location.

Finally, this approach allows us to treat the verb-preposition-NP constraints in a way independent from their syntactic realization. For example, the above example, with a left extraposition of the PP (due to a stressed focus) and a

nested proposition:
C'est pour Paris que part ce vol (this is for Paris that this flight leaves)
is treated in exactly the same way. This allows us to describe lexical choice of the pair verb-preposition at the highest level of abstraction, independently, a priori, of its syntactic realizations.

5 Conclusion

In this paper, we presented a model that describes in a declarative way the complex interactions between the verb, the preposition and the object NP. We focussed on the lexicalization of the preposition, which is a major difficulty in language generation.

Based on the notion of domain, we introduced equations that manage these interactions, and which, after resolution, propose a set of lexicalizations for the pair verb-preposition. We considered regular cases as well as derived ones, in particular a number of metaphors, modelled by means of type, semantic and pragmatic coercion rules.

We proposed in this paper a declarative model which has a priori completeness and soundness properties. These properties, as for e.g. grammars, entirely depends on the quality of the linguistic descriptions and restrictions, without which the system is just an empty shell.

In terms of implementation, this work is being integrated into a larger set of tools for language generation (including, among others: NP, PP, proposition and syntactic alternation generation). Such a set of tools, based on a decompositional representation of meaning, is of much interest and importance in language generation, where there are very few generic tools available. This work will be used and validated in the WEBCOOP project, where short statements with a large number of PPs need to be generated.

Implementations are under study. A simple solution is to manage set intersections, sets being viewed as finite domains. It is possible, furthermore, to define some of those sets (in particular those related to coercion) a priori, for each verb, by means, e.g. of partial evaluation techniques. We would like, on top of these set intersection constraints to be able to introduce linguistic preferences, be they general or related to a user. However, these types of heuristics can only operate after construction of the set of admissible lexicalizations. It would be of much interest to have them interact as early as possible in the resolution process in order to limit complexity.

At a linguistic level, this study involves only knowledge from lexical descriptions. It is abstract and not committed a priori to any grammatical form. This means that nothing prevents it from being integrated, with, obviously, some customization, into a linguistic theory such as HPSG, PP (via the projection principle), or LFG, and possibly TAGs, since these theories are centered around

lexical descriptions.

Acknowledgements. This project is partly supported by the CNRS TCAN program, we are grateful to its members for their advices. We also thank anonymous reviewers which helped improved this work.

References

1. F. Benamara and P. Saint-Dizier. WEBCOOP: a Cooperative Question-Answering System on the Web. EACL project notes, Budapest, Hungary, April 2003.
2. E.Cannesson, P. Saint-Dizier. Defining and Representing preposition Senses: a preliminary analysis. In ACL02-WSD, Philadelphia, July 2002.
3. L. Cahill. Lexicalisation in applied NLG systems, Research report, ITRI-99-04, 1999.
4. R. Jackendoff. Semantic Structures. MIT Press, 1990.
5. V. Moriceau and P. Saint-Dizier. A Conceptual Treatment of Metaphors for NLP. ICON, Mysore, India, December 2003.
6. J. Pustejovsky. The Generative Lexicon. MIT Press, 1995.
7. E. Reiter, R. Dale. *Building Applied Natural Language Generation Systems*, Journal of Natural Language Engineering, volume 3, number 1, pp:57-87, 1997.
8. P. Saint-Dizier. Alternations and Verb Semantic Classes for French. In Predicatives Forms for NL and LKB, P.Saint-Dizier (ed), Kluwer Academic, 1998.
9. P. Saint-Dizier and G. Vazquez. A Compositional Framework for Prepositions. IWCS4, Springer, lecture notes, Tilburg, 2001.
10. H. Verkuyl and J. Zwarts. Time and Space in Conceptual and Logical Semantics: the notion of Path, Linguitics 30: 483-511, 1992.
11. A. Mari and P. Saint-Dizier. A Conceptual Semantics for Prepositions Denoting Instrumentality. ACL-SIGSEM Workshop: The Linguistic Dimensions of Preposition and their Use in Computational Linguistics Formalisms and Applications, Toulouse, France, 2003.

Rapid Software Prototyping of an Arabic Morphological Analyzer in CLP

Hamza Zidoum

Department of Computer Science, SQU University,
PO BOX 36, Al Khod PC 123, Oman
zidoum@squ.edu.om

Abstract. This paper presents an Arabic Morphological Analyzer and its implementation in ECL^{i}PS^e a constraint logic programming language. The Morphological Analyzer (MA) represents a component of an architecture, which can process unrestricted text from a source such as Internet. The morphological analyzer uses a constraint-based model to represent the morphological rules for verbs and nouns, a matching algorithm to isolate the affixes and the root of a given word-form, and a linguistic knowledge base consisting in lists of markers. The morphological rules fall into two categories: the regular morphological rules of the Arabic grammar and the exception rules that represent the language exceptions. ECL^{i}PS^e is particularly suitable for a rapid prototyping of a morphological analyzer for Arabic thanks to its double reasoning: symbolic reasoning expresses the logic properties of the problem (linguistic rules) and facilitates the implementation of the linguistic knowledge base and heuristics, while constraint satisfaction reasoning on finite domains uses constraint propagation to keep the search space manageable while stemming the tokens.

1 Introduction

Morphology is the study of meaningful units in language (*morphemes*) and how they combine to form words. Hence, morphological analysis module is inherent to the architecture of any system that is intended to allow a user to query a collection of documents, process them, and extract salient information, as for instance, systems that handle Arabic texts and retrieve information expressed in Arabic language over Internet [11].

In this paper we present the implementation of a prototype Arabic morphological analyzer intended to be a component of an architecture that can process unrestricted text, within a Constraint Logic Programming (CLP) framework [12],[13],[14]. CLP is particularly suitable for the implementation of such a system thanks to its double reasoning: symbolic reasoning expresses the logic properties of the problem and facilitates the implementation of a the linguistic knowledge base, and heuristics, while constraint satisfaction reasoning on finite domains uses constraint propagation to keep the search space manageable. Constraints present an overwhelming advantage: *declarativity*. Constraints describe *what* the solution is and leave the answer to

question *how* to solve them to the underlying solvers. A typical constraint-based system has a two-level architecture consisting of (1) a programming module i.e. symbolic reasoning module that expresses the logic properties of the problem, and (2) the constraint module provides a computational domain such as reals, booleans, sets, finite domains, etc… and a reasoning about the properties of the constraints such that satisfiability, and reduction, algorithms known as *solvers*. Constraints thus, reduce the gap between the high level description of a problem and the code implemented.

The objective is to process a text in order to facilitate its usage by a wide range of further applications e.g.; text summary, translation, etc. The system is intended to process unrestricted text. Hence, the criteria of robustness and efficiency are critical, and highly desirable. To fulfill these criteria, we made the following choices:

1. Avoid the use of a dictionary as is it the case of classical morphological analyzers. Indeed, the coverage of such tool is limited to a given domain and cannot cope with unrestricted texts from dynamic information sources such as Internet.
2. Deal with *unvoweled* texts since most Arabic texts available on Internet are written in modern Arabic that usually doesn't use diacritical marks.

In order to implement the morphological analyzer, we used the *contextual exploration method* [1]. It consists of scanning a given linguistic marker and its context (surrounding tokens in a given text) looking for linguistic clues that guide the system to make the suitable decision. In our case, the method scans an input token and tries to find the required affixes in order to associate the root-form and the corresponding morpho-syntactic information.
Arabic is known to be a highly inflexional language; its famous pattern model using the CV (Consonant, Vowel) analysis has widely been used to build computational morphological models [2, 3, 4]. During the last decade, an increasing interest has been noticed to implement Arabic morphological analyzers [5, 6, 7, 8, 9, 10]. Almost all systems developed, in the industry as well as in the research, make use of a dictionary to perform morphological analysis. For instance, In France, the Xerox Centre developed an Arabic morphological analyzer [7] using the finite-state technique. The system uses an Arabic dictionary containing 4930 roots that are combined with patterns (an average of 18 pattern for every root). This system analyses words that may include full diacritics, partial diacritics, or no diacritics; and if no diacritics are present it returns all the possible analyses of the word.

The remaining sections of this paper are organized as follows. Section two describes the system design and its architecture. Section three is dedicated to the description of regular rules. The matching algorithm is described in section 4. Finally, section five concludes the paper with future directions to extend this work.

2 System Design and Implementation

Since the intended system is concerned with the morphology of verbs and particles only, we prefer the part of speech type classification in which an Arabic word can be a verb, a noun or a particle (Fig.1).

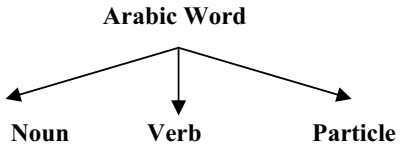


Fig.1. A Classification of Arabic words according to the Part of speech

One possible way of viewing the structure of a token - among several possible ones - and a convenient one is:
The main part of the token is the *stem* (whether noun, verb or particle). It is the inner part surrounded (possibly) by:
The prefix part positioned before the main part.
The suffix part positioned after the main part.
In the remaining paper, we concentrate on automating verbs morphology and leave particles, and nouns morphology for further work.

The following example shows the analysis of an Arabic sentence. This is similar to what the system should do in principle. The analyzed text is verse 186 of the second chapter of the Holy Qur'an.

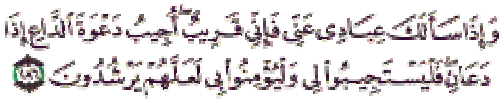


Table1 shows a segmentation of each word in the previous verse. The segmentation follows the affixes and body/stem schemes discussed in this chapter. Notice that the example shows some nouns for explanation purposes but the system deals with verbs and particles only. The segmentations process shown in the table is not the single task the system is designed to do; extracting useful information from these segments is another important function of the system. These functions do not depend on and make no use of the diacritical marks appearing in the example.

Token	*Prefix* 1,2,3	Body	Root	Measure	Suffix 1,2,3
و		و	و		
إذا		إذا	إذا		
سألك		سأل	سأل	فعل	,ك
عبادي		عباد	عبد	فعال	ي
عني		عن	عن		ي
فإني	ف	إن	إن		ي
قريب		قريب	قرب	فعيل	
أجيب	أ, ,	جيب	جيب	أفعل	
دعوة		دعوة	دعو	فعلة	
الداع	ال	داع	دعو	فاع	
إذا		إذا	إذا		
دعان		دعا	دعو	فعل	,ن
فليستجيبوا	ي,ل ,ف	استجيب	جيب	استفعل	,,وا
لي	ل	ي	ي		
وليؤمنوا	ي,ل ,و	ؤمن	أمن	فعل	,,وا
بي	ب	ي	ي		
لعلهم		لعل	لعل		هم
يرشدون	ي, ,	رشد	رشد	فعل	ون

Table 1. Segmentation/Stemming of Words Example

The morphological analyzer finds all word root forms and associates the *morpho-syntactic* information to the tokens. Within the text representation, a token includes the following fields.

1. *Name,* which contains the name of the token. Every token is assigned a name that allows the system to identify it.
2. *Value*, which is the word-form as it appears in the text before any processing.
3. *Root,* stores the root of the word that is computed during morphological analysis.
4. *Category, Tense, Number, Gender and Person* are the same fields as the regular-rules.
5. *Rule applied,* stores the identifier of the rule that has been fired to analyze the token.
6. *Sentence*, a reference to the object "Sentence" containing the token in the text. This is a relationship that holds between the token and its corresponding sentence.
7. *Order*, stores the rank of the token in sequence from the beginning of the text. It is used to compare the sequential order of tokens in the text.
8. *Positions,* correspond to the offset positions of the token in the text. It is used to highlight a relevant token when displaying the results to the user.
9. *Format*, is the associated format (boldface, italics…) applied to a token in the text.

The morphological analyzer (Fig.2) includes two kinds of rules: *regular morphologi-*

cal rules and *exception rules* that represent the morphological exceptions. Lists of exceptions contain all the markers that do not fall under the regular rules category. When analyzing input tokens, the matching algorithm attempts to match between the affixes of the token with a regular rule. If it does not succeed, it attempts to apply an exception rule by looking into the exception lists.

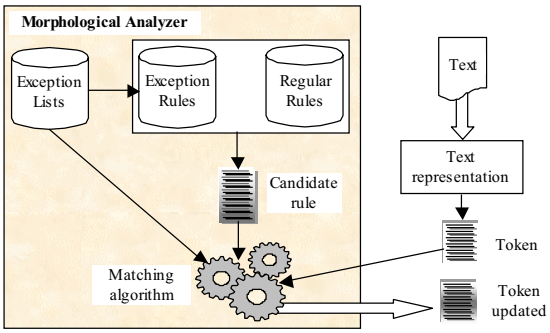


Fig. 2. The morphological analyzer architecture.

In view of a rapid-prototyping software implementation strategy, only regular rules specifications have been considered for implementation. After a test phase, the implementation of the exceptions is underway. We used Constraint Logic Programming language ECL^{i}PS^e [14] for building the Arabic morphological analyzer to benefit from several advantages. CLP is particularly suited for rapid-prototyping software implementation because it provides a high-level of abstraction. The reason lies in the neat separation between the declarative model (how to express the problem to be solved) and the operational model (how the problem is actually solved). A typical constraint-based system has a two-level architecture consisting of a programming module i.e. symbolic reasoning module expresses the logic properties of the problem and facilitates the implementation of the linguistic knowledge base, and heuristics; while constraint satisfaction reasoning on (finite) domains uses constraint propagation to keep the search space manageable [12], [13, [14].

3 Regular rules

Regular Arabic verb, and noun forms have a fixed pattern of the form "*prefix+root+suffix*", thus they can be implemented as automatic procedures since the identification of affixes is enough to extract the root form and associate the morpho-syntactic information.
A regular rule models a spelling rule for adding affixes. The structure of *regular-rule* consists of nine fields that can be grouped into three classes: i) *Name* and *Class* identify the object in the system, ii) *Prefix* and *Suffix* store the prefix and suffix that are

attached to a given token iii) *Category*, *Tense*, *Number*, *Gender*, and *Person* store the morpho-syntactic information inferred from a token. For instance, consider the Arabic word "يكتبون" (in the active mode: 'they write') that is composed of the three following morphemes: the root of the verb that is "كتب" (/ktb/, notion of writing), the prefix "يـ" that denotes both the present tense and the third person, and the suffix "ون" that denotes the masculine and the plural. The rule that analyses this word is represented in (fig. 3 (a)). In (fig. 3 (b)) the token is shown before matching, and in (fig. 3 (c)) the token attributes are updated:

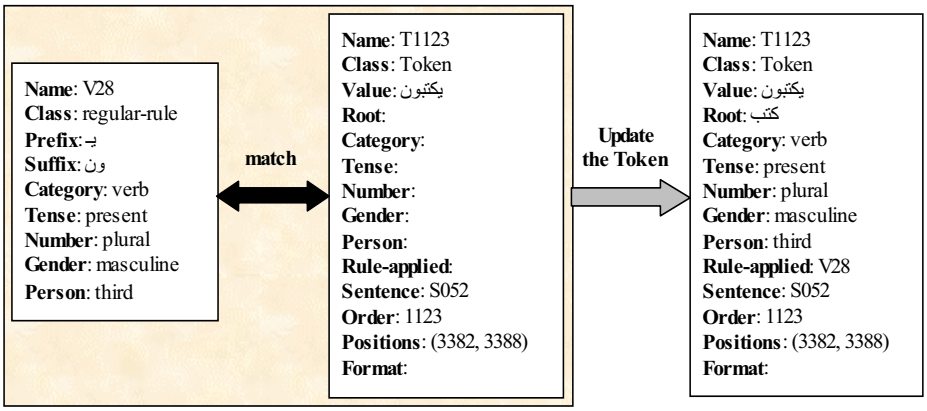


Fig. 3. Matching regular rules

The structure of the *regular-rule* class is detailed below:

1. *Name*: identifies uniquely a rule
2. *Class*: is the class of the object
3. *Prefix*: a sequence of characters at the beginning of a token
4. *Suffix*: a sequence of characters at the end of a token
5. *Category*: the part of speech to which the token belongs to. It can hold two possible values: verb or noun
6. *Tense*: the tense associated to the token in case of a verb
7. *Number*: the cardinality of the token consisting of singular, dual or plural
8. *Gender*: the gender associated to the token consisting of either masculine or feminine
9. *Person*: valid only for verbs, it represents either the first, second or third person.

Thus, the rules are represented as predicates, which implement the attributes as discussed above:

```
  regularRule(Pref, Suff, Cat, Tens, Num, Gend, Per, Name,
  Class):-
       Pref= "y",
       Suff= "wn",
       Cat = verb,
       Tens = present,
       Num = plural,
       Gend = masculin,
       Per = third,
       Name = v28,
       Class = regular_rule.
```

4 Matching Algorithm

The extracted tokens from the source text are represented through the following facts base which distinctive arguments are the value of the token itself and its corresponding root classification inferred by the algorithm

```
  token(name, value, root, cat, tense, number, gender, person,
format, ruleApplied, sentence, order, position).
```

The aim of token-to-rule matching algorithm implemented in predicate `tokenToRule/1` (shown below) is to fetch the rule that extracts the root of a given token `t`, and consequently, associates the morpho-syntax information to the token. The rule is identified if it matches a given pair of suffix and prefix. Thus, first the affixes are extracted from the token's value `v` through `getSuffix/3` and `getPrefix/3`, then the matching operation is performed by `regularRule/8`.

```
tokenToRule(T):-
          T = token(_, V,_,_,_,_,_,_,_, Rule,_,_,_),
          tokenToRule (V, Rule, Pref, Suff, 3, 1).

tokenToRule(V, Rule, Pref, Suff, I, J) :-
          I #< 0,
          Rule = null.
tokenToRule(V, Rule, Pref, Suff, I, J) :-
          length(V)-I-J #=< 1,
          Rule = null.

tokenToRule(V, Rule, Pref, Suff, I, J):-
          getPrefix(T, I, Pref),
          I #= I -1,
          I #>= 0,
          length(v)-I-J #> 1,
          tokenToRule(V, Rule, Pref, Suff, I, J).
```

```
tokenToRule(V, Rule, Pref, Suff, I, J):-
          j #< 0,
          Rule = null.
tokenToRule(V, Rule, Prefix, Suff, I, J):-
    length(V)-I-I #=< 1,
    Rule = null.
tokenToRule(V, Rule, Pref, Suff, I, J):-
    getSuffix(V, J, Suff),
    regularRule(Pref, Suff, Cat, Tens, Num,
                Gend, Pers, Name, Clas),
    Rule = Name,
    J #= J - 1,
    tokenToRule(J, Pref, Suff, I, J).
```

Note that the length of a prefix for regular rules is at most one character, and the length of a suffix is limited to three characters. The matching algorithm gives the priority to the longest affixes first. Hence, the predicate for affix extraction are respectively initialized to 1, and 3 and are gradually decremented as many time as no rule matches and the root still contains at least 2 characters (constraint `length(V)-I-J>1`) due to the fact that for regular words no root is less than two characters long:

```
getPrefix(V, I, Pref):-
      getPrefix1(V, I, [], Pref).
getPrefix1(V, 0, Pref, Pref).
getPrefix1(V, I, Pref, _):-

      I1 #= I-1
      getPrefix1([C|V1], I1, [C| Pref], _).
getSuffix(V, I, Suff):-
      reverse(V, V1),
      getSuffix(V1, I, [],Suff).
getSuffix1(V, 0, Suff, Suff1):-
      reverse(Suff, Suff1).
getSuffix1(V, I, Suff, _):-

     I1 #= I-1
      getSuffix1([C|V1], I1, [C|Suff1], _).
```

5 Conclusion

A morphological analyzer is one of the essential components in any natural language processing architecture. The morphological analyzer presented in this paper is implemented within CLP framework and is composed of three main components: a linguistic knowledge base comprising the regular and irregular morphological rules of the Arabic grammar, a set of linguistic lists of markers containing the exceptions handled by the irregular rules, and a matching algorithm that matches the tokens to the rules. The complete implementation of the system is underway. In a first phase, we have considered only the regular rules for implementation. Defining a strategy to match the regular and irregular rules and the extension of the linguistic lists of markers are the future directions of this project.

References

1. J-P. Desclès, Langages applicatifs, Langues naturelles et Cognition, Hermès, Paris, 1990.
2. A. Arrajihi, *The Application of morphology*, Dar Al Maarefa Al Jameeya, Alexandria, 1973. (in Arabic)
3. F. Qabawah, *Morphology of nouns and verbs*, Al Maaref Edition, 2nd edition, Beyruth, 1994. (in Arabic)
4. G. A. Kiraz, "Arabic Computational Morphology in the West.", In *Proceedings of the 6th International Conference and Exhibition on Multi-lingual Computing*, Cambridge, 1998.
5. B. Saliba and A. Al Dannan, "Automatic Morphological Analysis of Arabic: A study of Content Word Analysis", In *Proceedings of the Kuwait Computer Conference*, Kuwait, March 3-5, 1989.
6. M. Smets, "Paradigmatic Treatment of Arabic Morphology", In *Workshop on Computational Approaches to Semitic Languages COLING-ACL98*, August 16, Montreal, 1998.
7. K. Beesley, "Arabic Morphological Analysis on the Internet", In *Proceedings of the International Conference on Multi-Lingual Computing (Arabic & English)*, Cambridge G.B., 17-18 April, 1998.
8. R. Alshalabi and M. Evens, "A Computational Morphology System for Arabic", In *Workshop on Computational Approaches to Semitic Languages COLING-ACL98*, Montreal.
9. R. Zajac, and M. Casper, "The temple Web Translator", 1997 Available at: http://www.crl.nmsu.edu/Research/Projects/tide/papers/twt.aaai97.html
10. T. A. El-Sadany, and M. A. Hashish, "An Arabic Morphological System", In *IBM Systems J.*, Vol. 28, No. 4, 600-612, 1989.
11. J. Berri, H. Zidoum., Y. Attif, "Web-based Arabic Morphological Analyser", A. Gelbukh (Ed): CICLing 2001, pp.389-400, 2001, Springer-Verlag, 2001.
12. K. Marriott and P. Stuckey, "Programming with constraints: An Introduction", MIT Press, 1998.
13. P. Codognet, and D Diaz, "Compiling Constraints in clp(FD)", Journal of Logic Programming, 1996:27:1-199.
14. A M Cheadle, W Harvey, A J Sadler, J Schimpf, K Shen and M G Wallace.*ECLiPSe: An Introduction*. IC-Parc, Imperial College London, Technical Report IC-Parc-03-1, 2003.

A tutorial on CHR Grammar

Henning Christiansen

Roskilde University, Computer Science Dept.
P.O.Box 260, DK-4000 Roskilde, Denmark
E-mail: henning@ruc.dk

Abstract. Constraint Handling Rules (CHR) are a declarative language for writing constraint solvers which is available as an extension of Prolog and CHR Grammars (CHRG) provide a grammar formalism added on top of CHR, analogously to the way Definite Clause Grammars (DCG) are added on top of Prolog. CHRG inherits the underlying execution model which immediately provides robust bottom-up parsers that evaluate all possible parses given by an ambiguous grammar in parallel. Compared with DCG, CHRG provides a high degree of flexibility and expressibility including a sort of context-sensitive rules, a straightforward implementation of abduction. The system includes different tools for optimization inherited from CHR so that most grammars can perform with a reasonable speed. This tutorial gives a short introduction to CHRG and its applications, mainly in terms of examples.

1 Introduction

The language of Constraint Handling Rules (CHR) was introduced originally by Th. Frühwirth [1] as a declarative language for writing constraint solvers for constraint logic programming, such as finite domain solvers and constraint programming over real and rational numbers. The language consists of rewrite rules over constraint stores: If a set if constraints matched by the head of a given rule is identified in the constraint store, the rule may fire resulting in the addition of new constraints to the store, possibly removing those matched by the head (depending of the sort of rule applied); CHR rules may include a guard to determine whether or not a rule can apply and the procedural semantics is committed-choice so that no backtracking is possible.

The fundamental assumption underlying the introduction of a grammar notation on top of CHR, is that languages analysis is a process of constraint solving, involving many different conceptual layers (lexical, morphologic, syntactic, pragmatic, etc.) possibly executing in parallel. In the following we present the system of CHR Grammars, or CHRG for short, mainly in terms of examples.

CHRG supports parsing by an encoding of an input string by means of word boundaries (given by integer numbers) attached to each syntactic node which is represented as a CHR constraint. Grammar rules are translated in a straightforward way into CHR rules that identify sequences of tokens by comparing word boundaries. This encoding makes it possible also to represent context-sensitive

rules and a notion of gaps shown in the following. CHRG includes the full power of the underlying CHR system so that parsing can integrate with arbitrary constraint solvers written in CHR. Abduction in language analysis, for example, is realized by having grammar rules to cast of "abducibles" into the constraint store, further processed by integrity constraints written directly as CHR rules.

Applications of CHRG until now have concerned mainly syntactic aspects, scratching the surface of semantic and contextual analysis; current research is trying to produce a model in which the pragmatic meaning of sentences are produced right away as a product of language analysis (i.e., avoiding the construction of huge parameterized terms representing de-contextualized semantics).

A comprehensive treatment of CHRG is given in [2] that also gives an overview of related work; the system is available at the CHRG web site [3]. We do not consider the implementation of CHRG here; see [2]. In the following, we introduce the main features of the system.

2 An introductory example

The following example shows a source text as it is given to the CHRG system (with a few trivial reminiscences of the underlying CHR system retouched away).

```
grammar_symbols np/0, verb/0, sentence/0.

np, verb, np ::> sentence.
[peter] ::> np.
[mary] ::> np.
[likes] ::> verb.
```

As it appears, the notation is inspired by DCG so, for example, terminal strings are indicated by square bracket. However, the role of head and body (lhs resp. rhs) is switched in order to indicate the bottom-up nature of the system and to mimmic the underlying CHR syntax. The arrow `::>` indicates a propagation rule meaning that the rules only add new grammar nodes to the store without removing anything (analogous to CHR's notion of the same name). So-called simplification rules are also available, indicated by `<:>`, which remove the nodes matched by the head. A list of tokens can be analyzed by a predicate `parse` that translates it into a set of constraints with explicit word boundaries and which will start the (compiled) grammar rules to apply until no more applications are possible. Consider for example the following dialogue with the system.

```
?- parse([peter,likes,mary]).
<0> peter <1> likes <2> mary <3>
np(0,1),
verb(1,2),
np(2,3),
sentence(0,3),
token(0,1,peter),
```

```
token(1,2,likes),
token(2,3,mary) ?
```

The systems echoes the string with indication of boundaries for easy evaluation of the result given as the final constraint store which, in this example, shows all grammar nodes constructed during parsing. (It is, of course, possible, to suppress details so that only the most essential information is given as answer).

3 Context-sensitive rules and an example of coordination

The head of a rule may include both a left and right context; the general shape of a head is as follows

left-context `-\` *core* `/-` *right-context*

where the three indicated components are strings of grammar symbols (context parts may be empty in which case the corresponding marker is left out). The meaning is that the rule can apply, recognizing the core to be whatever is specified by the body of that rule, provided the strings of grammar symbols indicated by the context parts are observed immediately to the left and right of the core.

In the following excerpt of a grammar for simple sentences, each grammar node has an attribute (analogous to DCG) that gives a symbolic representation of the sentence analyzed.

We consider simple coordinating sentences such as *"Peter likes and Mary detests spinach"* where the incomplete first sentence inherits the object from the following complete sentence.

```
sub(A), verb(V), obj(B) ::> sent(s(A,V,B)).
subj(A), verb(V) /- [and], sent(s(_,_,B)) ::> sent(s(A,V,B)).
```

The first rule is to be understood in the usual way that a complete `sub-verb-obj` sequence can be reduced to a `sent` node. The second rule is an example of a context-sensitive rule: It applies to a `subj-verb` sequence only when followed by terminal symbol "`and`" and another `sent` node, and in this case the incomplete sentence takes its obbject, matched by variable `B`, from this following `sent` node.

4 Simplification rules and context parts for disambiguation

Context parts can be used for a multitude of purposes, in [4], for example, for resolving ambiguities of the highly compact and syntactically ambiguous language of hieroglyph inscriptions. Here we show a grammar that uses left context parts to resolve ambiguities in an otherwise highly ambiguous grammar arithmetic expressions (computer scientists will recognize the standard follow-items in the context parts, cf. [5]). This grammar consists of simplification rules so that any symbol, once matched in a rule application, cannot give rise to other rule applications.

```
e(E1),[+],e(E2) /- Rc <:> e(plus(E1,E2))
   where Rc = (['+'];[')'];[eof]).

e(E1),[*],e(E2) /- Rc <:> e(times(E1,E2))
   where Rc = ([*];[+];[')'];[eof]).

e(E1),[^],e(E2) /- [X] <:> X \= ^ | e(exp(E1,E2)).

['('], e(E), [')'] <:> e(E).

[N] <:> integer(N) | e(N).
```

Notice also that some rules include a guard separated from the body by a vertical bar, and some auxiliary notation: semicolon in context means different alternatives and `where` indicates straightforward textual substitution.

5 An application of gaps for a simplified microbiologic example

Context parts as well as the core of the head of a rule may contain gaps. The symbol "`...`" indicates a string of arbitrary length and it can be given extra arguments to indicate restrictions on the length of acceptable strings, so that "`5...10`" can match any string of length between 5 and 10.

Gaps may be useful in analyzing chemical formulas for proteins in order to predict properties of their physical shape. A loop, for example, may be indicated by a certain subtring that appears later in reversed form. The following grammar rule will recognize any such loop where the "gluing" substrings are of length exactly 6 and the loop itself of length between 10 and 20.

```
[A1,A2,A3,A4,A5,A6], 10...20, [A6,A5,A4,A3,A2,A1] ::> loop.
```

In practice, more rules are needed in order to extend the "glue" as far as possible, and the match should be made between a substring and a reversed version of complementary symbols (representing opposite electrical charges).

6 Abduction and assumptions

The CHRG notation makes it possible for a grammar rule to work with arbitrary atomic hypotheses (CHR constraints) in addition to grammar symbols. Curly brackets can be used in both head and body of a rule to indicate such hypothesis that should not be confused with grammar symbols. So for example

```
{h(1)}, [a] ::> b, {h(2)}.
```

indicates that terminal symbol "`a`" can be recognized to be of category "`b`" provided that hypothesis "`h(1)`" is present in the store, and in which case an

additional hypothesis "`h(2)`" is created. This provides a potentiality for grammars to interact with a semantic (or pragmatic) context[1] for the given discourse. The behaviour of the analyzer may depend on the world knowledge recognized so far ("`h(1)`") and may add new world knowledge ("`h(2)`"). In addition, general knowledge of the world can be expressed as rules of CHR (that can be mixed with CHRG), so for example

```
h(1), h(2) ==> h(3).
```

indicated that if "`h(1)`, `h(2)`" is known, then we have also "`h(3)`".

The CHRG system is prepared for two kinds of applications of such extra-grammatical hypotheses, abduction and assumptions in the style of [6]. More details and examples can be found in [2]; the embedding of abduction in the system in this straightforward fashion is based on a technical trick observed in [7].

The newest version of the CHRG system includes facilities to explore all possible abductive explanations of a given discourse in parallel (which is problematic in the system as it is described above since it may mix up alternative hypothesis sets).

7 Perspectives

A constraint-based grammar formalism has been presented which provides a flexible medium for expressing a variety of linguistic phenomena and which is based on the computational paradigm of Constraint Handling Rules.

The formalism (and implemented system) seems well suited for experiments with language analysis that integrates different layers of analysis. However, experience is still limited to toy examples but the system is freely available on the word wide web for anyone who may be inspired to make experiments.

Acknowledgements

This research is supported in part by CONTROL project funded by the Danish Natural Science Research Council, the OntoQuery funded by the Danish Research Councils, and the IT-University of Copenhagen.

References

1. Frühwirth, T.: Theory and practice of constraint handling rules, special issue on constraint logic programming. Journal of Logic Programming **37** (1998) 95–138
2. Christiansen, H.: CHR Grammars. Int'l Journal on Theory and Practice of Logic Programming (2005) To appear.

[1] Notice that this usage of "context" is unrelated to the syntactic contexts considered above

3. Christiansen, H.: CHR Grammar web site; released 2002. http://www.ruc.dk/~henning/chrg (2002)
4. Hecksher, T., Nielsen, S.T.B., Pigeon, A.: A CHRG model of the ancient Egyptian grammar. Unpublished student project report, Roskilde University, Denmark (2002)
5. Aho, A., Sethi, R., Ullman, J.: Compilers. Principles, techniques, and tools. Addison-Wesley (1986)
6. Dahl, V., Tarau, P., Li, R.: Assumption grammars for processing natural language. In Naish, L., ed.: Proceedings of the 14th International Conference on Logic Programming, Cambridge, MIT Press (1997) 256–270
7. Christiansen, H.: Abductive language interpretation as bottom-up deduction. In Wintner, S., ed.: Natural Language Understanding and Logic Programming. Volume 92 of Datalogiske Skrifter., Roskilde, Denmark (2002) 33–47

Representing Act-Topic-based Dialogue Phenomena

Hans Dybkjær[1] and Laila Dybkjær[2]

[1]Prolog Development Center A/S (PDC),
H. J. Holst Vej 3C-5C, 2605 Brøndby, Denmark, dybkjaer@pdc.dk
[2]Nislab, University of Southern Denmark (SDU),
Campusvej 55, 5230 Odense M, Denmark, laila@nis.sdu.dk

Abstract. We examine phenomena in spoken human computer dialogues and suggest possible formalisations. The work is a step towards automating spoken dialogue systems assessment.

Keywords. Spoken dialogue, act-topic, structure analysis, transactions.

1 Introduction

Dialogue smoothness and transaction success rate are important SDS usability evaluation criteria but are costly to measure manually. Automating the measurement process would be of great benefit to the SDS community.

We suggest a two-step approach to the automatic annotation of act-topics and, eventually, of transactions. *First*, basic, context-independent act-topic annotation is added to all system and user utterances (Figure 1, left). Basic acts include `inform`, `accept`, `reject`. *Second*, basic acts are combined into composite acts and then further combined into transaction segments tagged with success or failure (Figure 1, right).

This paper investigates the second step: what is needed to automate act-topic based transaction structure annotation of dialogues between users and spoken dialogue systems (SDSs), such as a frequently asked questions (FAQ) system [Dybkjær and Dybkjær 2004], a flight ticket reservation system [Bernsen et al. 1998], and a train timetable information system [Aust et al. 1995]. As the formal vehicle for deriving composite acts we use rewrite rules with unification and supplementary constraints.

```
.s: .inform {N.employee, N.leave
…}
    "You can … choose between:
     'employee' 'on leave' …"
.u: .inform {N.student}
    "I'm a student"
.s: .inform {N.menu}
    "Did you ask for – Main menu?"
.u: .inform {N.student}
    "Student"
.s: .inform {V.student, V.su …}
    "If you are a student and
     receive SU, you may …"
```

```
.s: .success {N.student}
  <- success2
  .u: .request {N.student}
    <- sequenceRequest
    .u: request {N.student}
      <- request2
      .s: inform {N.employee, N.leave …}
      .u: inform {N.student}
    .u: request {N.student}
      <- request2
      .s: inform {N.menu}
      .u: inform {N.student}
  .s: .inform {V.student, V.su …}
```

Figure 1: Example dialogue, annotated with acts and topics.

2 Dialogues, turns, and moves

A *dialogue* is a sequence of *moves* where each move corresponds to one act and a set of topics for one speaker (Figure 2). An *utterance* is a sequence of moves of one speaker. A *turn* is an utterance that further satisfies that if other moves occur at the ends, then these moves belong to other speakers. In Figure 2, e.g., the example has one user utterance that is also a turn, and there are two system moves that may make one or two utterances and precisely one turn.

```
dialogue = move*
move = who : act topics ["text"]
who = .u | .s
act = .identifier
topics = { topic* }
topic = distinction.identifier
distinction = T | N | V
```

Example:

```
.s: .pause {}
.s: .inform {T.more}
    "Do you want more?"
.u: .accept {}
    "yes"
```

Figure 2: Formal structure of a dialogue.

3 Dialogue structure via rewrite rules

Rewrite rules define an acyclic graph which may be seen as the dialogue structure. Each rule takes a move pattern and produces a new sequence of moves. A *pattern* is a list of utterances but may contain variables for *who*, *act*, *topics*, and *topic*, cf. Figure 3.

```
rule =  rule identifier
          move* <- move*
          [ where condition* ]
        end rule
move M = who : act topics
who W = varVal
act A = varVal
topics Ts = { topic* } | varVal
topic T = varVal
varVal = [Type]var | [Type]val
var = _identifier
val = .identifier
condition = varVal operator varVal
operator = =|!= |in|not-in | <
```

Example:

```
rule select1
  _y: .select Ts_a
    <-
    _x: .inform Ts_a
    _y: .accept {}
  where
    _x != _y
end rule
```

Result when applied to example (Figure 2):

```
.s: .pause {}
.u: .select {T.more}
  <- select1
  .s: .inform {T.more}
      "Do you want more?"
  .u: .accept {}
      "yes"
```

Figure 3: Formal structure of rules and their application.

4 Analysing dialogue phenomena

The minimum expressiveness of the above rules would allow for one speaker, one act, one topic, and no constraints. The table below explains and exemplifies different needed extensions to the minimum expressiveness based on various dialogue phenomena. Rule references are to Appendix A.

Different topics

Allowing several topics enables detection of two moves including the same topic.

See `rule segment`

Different acts

```
x: Are you going to Copenhagen?
y: Yes
```

Allowing *different acts* instead of only one, enables distinction among certain patterns, such as a select pattern which may consist of an inform act telling about an option followed by an accept act of this option.

See `rule select0`

Differentiating speakers

```
s: Are you going to Copenhagen?
u: Yes
```

To distinguish that moves are by different speakers, an *inequality operator* is needed.

See `rule select1`

More topics in a move

```
x: Do you want to know about when the money is paid, transfer of money,
   or payment in general?
y: Payment in general.
```

A select rule matching this example must express that speaker `y` states one of the topics offered by speaker `x`. A member operator "`T in Ts`" is added as a constraint.

See `rule select2`

More topics in different moves

```
.s: .inform {T.paymentWhen, T.moneyTransfer, T.paymentGeneral}
    "Do you want to know about when the money is paid, transfer of money,
     or payment in general?"
.u: .inform {T.paymentGeneral, T.friend}
    "Payment in general, my friend."
```

Here we need to express that the same topic occurs in two different lists. We do so by allowing variables to be introduced in the constraints, too, and not only in the pattern.

See `rule select2a`

Rejecting topics

```
x: Do you want to know about when the money is paid, transfer of money,
   or payment in general?
y: My employer is bankrupt.
```

Symmetrically to selecting a topic a speaker may reject the offered topics by requesting a new topic, so we add the "`T notT-in Ts`"operator.

See `rule request`

Distinguishing names and values

```
.u: .inform {N.phone}
    "Your phone number?"
.s: .inform {N.phone}
    "Phone number"
```

```
.u: .inform {N.phone}
    "Your phone number?"
.s: .inform {V.phone}
    "Phone 48204910"
```

In order to distinguish the analyses of these two examples, we must distinguish topic names (N) and values (V). By a topic *name* we understand the mentioning of a topic, e.g. in terms of a user requesting information about a certain topic. By a topic *value* we understand details about a certain topic, e.g. the system informing about a topic name selected by the user.

See `rules select2, answer, success1 and success2`

Patterns across turns not using all the turn moves

```
0 .u: .inform {V.aalborg, V.tomorrow}
      "I want to go to Aalborg tomorrow."
1 .s: .inform {V.aalborg, V.aarhus}
      "Did you say to Aalborg or to Aarhus?"
2 .u: .inform {V.aalborg}
      "To Aalborg."
3 .s: .inform {V.tomorrow}
4 .s: .inform {V.aalborg}
      "Are you leaving tomorrow for Aalborg?"
5 .u: .accept {}
      "Yes."
```

The dialogue fragment above contains a success in selecting travel destination and date. The moves 1+2 may be reduced to `select {V.aalborg}` which potentially enables us to apply the `success1` rule. However, the subsequent utterance is annotated with the moves in a wrong order for this. There is no inherent reason why the order of moves within an utterance or turn should be important (at the level of analysis we do), and the equivalent formulation of move 4 above "`Aalborg. Are you leaving tomorrow?`" would naturally have made the annotator list the moves in the reverse order.

So we will allow patterns to match moves within turns in any order, leaving unused moves if the turns are at the ends of the pattern, otherwise dropping the unused moves.

Ontological relations

```
.s: .inform {N.travel, N.from}
    "Where does the travel start?"
.u: .inform {V.place}
    "Copenhagen"
```

When annotating each move independently of the context it becomes ambiguous what a topic value refers to. E.g. the place “Copenhagen” may be departure or destination city. To be able to automatically relate the question and the reply in such situations we need to introduce the *sub-topic relation* `T1 < T2`.

See rule `selectSub1`

Meta-communication and multi-level rule applications

In Figure 1 it seems fair to count a success and no failures, and to count one meta-exchange which is negative for the smoothness. If the two requests had been divided by several exchanges or even a full transaction on another topic, it is less obvious that the two requests should be counted as one, leading to one success. Part of a solution to handling meta-communication involves the division of rules into several sets that are applied successively. This reflects that naturally one would first do simple rewrites like detecting request and select, then handle meta-communication, then transactions, and finally issues like summarising feedback.

See rule `sequenceRequest`

Summarising feedback

```
.s: .inform {V.from, V.to, V.hour}
    "Es gibt die folgende Verbindung: Mit der S-Bahn Abfahrt in Berlin
    Hauptbahnhof um fuenfzehn Uhr vierundzwanzig Ankunft in Berlin-Zoo um
    fuenfzehn Uhr einundvierzig dort weiter mit … dort weiter mit Inter-
    city sechs fuenf zwei Abfahrt um zwanzig Uhr einundfuenfzig Ankunft in
    Darmstadt Hauptbahnhof um einundzwanzig Uhr neun"
```

We call such information *summarising feedback*. It is fairly common in information and booking systems. Often systems implementing this have "one call – one task" dialogues, and a simple measure of transaction success is to call it a success if the system reaches this state, and otherwise a failure. However, at least on two points this is problematic:

- The user may disagree in the summarisation, claiming something to be wrong.
- It provides no information on dialogue smoothness up until this point.

By instead using rules like those in Appendix A and assuming the `reject3` rule is applied before the `successSummary` rule, we may get a success/failure annotation that deals with the above two bullet points (`reject3` will block `successSummary`).
See rule `reject3` and `successSummary`

5 Conclusion

By applying subsequent levels of act-topic rewrite rules we can analyse a set of dialogue phenomena occurring in typical SDSs, eventually leading to automatic detection of non-smoothness and of transaction successes and failures.

Compared to other work, the two key distinguishing features of our analysis are *automation* and *act-topic structures*. Many other papers discuss how to find acts and/or topics [Heeman et al. 1998, Jurafsky et al. 1997], often based on statistical methods, but are not concerned with the further structural analysis of the dialogue structure. The probably most dominant discourse structure theory, RST (Rhetorical Structure Theory, [Mann and Thomson 1987]), is not aimed at computational analysis.

Much work remains to be done. There are unanalysed issues regarding in particular smoothness and summarising feedback, common to which is that they concern phenomena distributed over large parts of the dialogue instead of being locally (and continuously) defined. Other issues also needing further analysis include task dependence of rules, summaries not including all information, information stated in disguise, and inexact matches. The rules must be tested on larger sets of dialogues of different type. An automatic act-topic annotation parser must be made in order to achieve full automation.

Note that our approach only considers structure. For instance, the correctness of summarising feedback is not considered.

References

[Aust et al. 1995] Harald Aust, Martin Oerder, Frank Seide, and Volker Stenbiss: The Philips Automatic Train Timetable Information System. *Speech Communication*, 17, 249-262, 1995.

[Bernsen et al. 1998] Niels Ole Bernsen, Hans Dybkjær and Laila Dybkjær: *Designing Interactive Speech Systems. From First Ideas to User Testing.* Springer Verlag 1998.

[Dybkjær and Dybkjær 2004] Hans Dybkjær and Laila Dybkjær: Modeling Complex Spoken Dialog. *Computer*, IEEE, August 2004, 32-40.

[Heeman et al. 1998] Peter A. Heeman, Donna Byron, and James F. Allen: Identifying Discourse Markers in Spoken Dialog. *AAAI Spring Symposium on Applying Machine Learning and Discourse Processing*, Stanford, March 1998.

[Jurafsky et al. 1997] Daniel Jurafsky, Rebecca Bates, Noah Coccaro, Rachel Martin, Marie Meteer, Klaus Ries, Elizabeth Shriberg, Andreas Stolcke, Paul Taylor, and Carol van Ess-Dykema: *Automatic Detection of Discourse Structure for Speech Recognition and Understanding*. Proc. of IEEE Workshop on Speech Recognition and Understanding, 1997, 88-95.

[Mann and Thompson 1987] William C. Mann and Sandra A. Thompson: Rhetorical Structure Theory: Description and Construction of Text Structures. In Gerard Kempen (ed.): *Natural Language Generation. New results in artificial intelligence, psychology and linguistics.* NATO ASI series E 135, Chapter 7. The Netherlands, Martinus Nijhoff Publishers, 1987.

Appendix A Example rules

```
rule segment
  _y: .any {T_a}
    <-
    _x: .any {T_a}
    _y: .any {T_a}
end rule
```

```
rule select0
  _y: .select {T_a}
    <-
    _x: .inform {T_a}
    _y: .accept {}
end rule
```

```
rule select1
  _y: .select {T_a}
    <-
    _x: .inform {T_a}
    _y: .accept {}
  where
    _x != y
end rule
```

```
rule select2
  _y: .select {T_b}
    <-
    _x: .inform Ts_a
    _y: .inform {T_b}
  where
    _x != y
    _b in _a
end rule
```

```
rule select2a
  _y: .select {T_c}
    <-
    _x: .inform Ts_a
    _y: .inform Ts_b
  where
    _c in _a
    _c in _b
    _x != y
end rule
```

```
rule request
  _y: .request {T_b}
    <-
    _x: .inform Ts_a
    _y: .inform {T_b}
  where
    _x != y
    _b not-in _a
end rule
```

```
rule answer
  _y: .request {N_b}
  _x: .inform Vs_a
    <-
    _y: .inform {N_b}
    _x: .inform {V_b}
  where
    _x != _y
end rule
```

```
rule success1
  _y: .success {N_b}
    <-
    _x: .select {N_b}
    _y: .inform Vs_a
  where
    _b in Vs_a
    _x != _y
end rule
```

```
rule success2
  _y: .success {N_b}
    <-
    _x: .request {N_b}
    _y: .inform Vs_a
  where
    _b in Vs_a
    _x != _y
end rule
```

```
rule request2
  _y: .request {V_b}
    <-
    _x: .inform Ts_a
    _y: .inform {V_b}
  where
    _x != y
    _b not-in _a
end rule
```

```
from-place < place
rule selectSub1
  _y: .select {N_a}
  <-
    _x: .inform {N_a}
    _y: .inform {T_b}
  where
    _a < _b
    _x != _y
end rule
```

```
rule sequenceRequest
  _y: .request {T_a}
    <-
    _y: .request {T_b}
    _y: .request {T_b}
end rule
```

```
Rule reject3
  _u: .reject {V.toPlace,
               V.fromPlace,
               V.departureTime}
    <-
    _s: .inform {V.toPlace,
                 V.fromPlace,
                 V.departureTime}
    _u: .reject
end rule
```

```
rule successSummary
  _y: .success {N.travel}
    <-
    _u: .success {V.departureTime}
    _u: .success {V.fromPlace}
    _u: .success {V.toPlace}
    _s: .inform {V.toPlace,
                 V.fromPlace,
                 V.departureTime}
end rule
```

Multi-dimensional Type Theory: Rules, Categories, and Combinators for Syntax and Semantics

Jørgen Villadsen

Computer Science, Roskilde University
Building 42.1, DK-4000 Roskilde, Denmark

jv@ruc.dk

Abstract. We investigate the possibility of modelling the syntax and semantics of natural language by constraints, or rules, imposed by the multi-dimensional type theory Nabla. The only multiplicity we explicitly consider is two, namely one dimension for the syntax and one dimension for the semantics, but the general perspective is important. For example, issues of pragmatics could be handled as additional dimensions.
One of the main problems addressed is the rather complicated repertoire of operations that exists besides the notion of categories in traditional Montague grammar. For the syntax we use a categorial grammar along the lines of Lambek. For the semantics we use so-called lexical and logical combinators inspired by work in natural logic. Nabla provides a concise interpretation and a sequent calculus as the basis for implementations.

... Lambek originally presented his type logic as a calculus of *syntactic* types. Semantic interpretation of categorial deductions along the lines of the Curry-Howard correspondence was put on the categorial agenda in J. van Benthem (1983) *The semantics of variety in categorial grammar*, Report 83-29*, Simon Fraser University, Canada. This contribution made it clear how the categorial type logics realize Montagues Universal Grammar program — in fact, how they improve on Montagues own execution of that program in offering an integrated account of the composition of linguistic meaning *and* form. Montagues adoption of a categorial syntax does not go far beyond notation: he was not interested in offering a principled theory of allowable 'syntactic operations' going with the category formalism.

* Revised version in [1]

M. Moortgat (1997) *Categorial Type Logics*, in J. van Benthem & A. ter Meulen (eds.) *Handbook of Logic and Language*, Elsevier.

Full paper in Computing Research Repository http://arXiv.org (Computation and Language). This research was partly sponsored by the IT University of Copenhagen and the CONTROL project: CONstraint based Tools for RObust Language processing http://control.ruc.dk

1 Introduction

We investigate the possibility of modelling the syntax and semantics of natural language by constraints, or rules, imposed by the multi-dimensional type theory Nabla [14]. The only multiplicity we explicitly consider here is two, namely one dimension for the syntax and one dimension for the semantics, but we find the general perspective to be important. For example, issues of pragmatics could be handled as additional dimensions by taking into account direct references to language users and, possibly, other elements of the situation in which expressions are used. We note that it is possible to combine many dimensions into a single dimension using Cartesian products. Hence there is no theoretical difference between a one-dimensional type theory and a multi-dimensional type theory. However, we think that in practice the gain can be substantial.

Nabla is a linguistic system based on categorial grammars [1] and with so-called lexical and logical combinators [15] inspired by work in natural logic [12]. The original goal was to provide a framework in which to do reasoning involving propositional attitudes like knowledge and beliefs [16,17].

2 The Rules

We define a multi-dimensional type theory with the two dimensions: syntax and semantics. We use a kind of the so-called Lambek calculus with the two type constructors / and \, which are right- and left-looking functors [7,8,10].

We assume a set of basic types $\mathcal{T}_0$, where $\bullet \in \mathcal{T}_0$ is interpreted as truth values. The set of types $\mathcal{T}$ is the smallest set of expressions containing $\mathcal{T}_0$ such that if $A, B \in \mathcal{T}$ then $A/B, B\backslash A \in \mathcal{T}$.

A structure consists of a vocabulary and a set of bases $\mathcal{S} \equiv \langle \mathcal{V}, \mathcal{B} \rangle$, where $\mathcal{V}$ is finite and $\mathcal{B}(A) \neq \emptyset$ for all $A \in \mathcal{T}_0$.

We define three auxiliary functions on types (the first for the syntactic dimension and the second for the semantic dimension; symbol $\doteq$ is used for such "mathematical" definitions, in contrast with $\equiv$ for literal definitions):

$$\lceil A \rceil \doteq \mathcal{V}^+, A \in \mathcal{T}$$

$$\lfloor A \rfloor \doteq \mathcal{B}(A), A \in \mathcal{T}_0$$
$$\lfloor A/B \rfloor \doteq \lfloor B\backslash A \rfloor \doteq \lfloor B \rfloor \to \lfloor A \rfloor$$

$$|A| \doteq \lceil A \rceil \times \lfloor A \rfloor$$

By $\mathcal{V}^+$ we mean the set of (non-empty) sequences of elements from $\mathcal{V}$ (such sequences correspond to strings and for the sake of simplicity we call them strings).

The universe is $\bigcup_{A \in \mathcal{T}} |A|$ (which depends only on the structure $\mathcal{S}$).

With respect to $\mathcal{S}$ we extend a basic type interpretation $[\![A]\!] \subseteq |A|$ $(A \in \mathcal{T}_0)$ to a type interpretation $[\![A]\!] \subseteq |A|$ $(A \in \mathcal{T})$ as follows (the concatenation of the strings x and x' is written $x \hat{} x'$):

$$[\![A/B]\!] \doteq \{\ \langle x, y \rangle \mid \text{ for all } x', y', \text{ if } \langle x', y' \rangle \in [\![B]\!] \text{ then } \langle x \hat{} x', y y' \rangle \in [\![A]\!]\ \}$$
$$[\![B\backslash A]\!] \doteq \{\ \langle x, y \rangle \mid \text{ for all } x', y', \text{ if } \langle x', y' \rangle \in [\![B]\!] \text{ then } \langle x' \hat{} x, y y' \rangle \in [\![A]\!]\ \}$$

We use a so-called sequent calculus [11] with an explicit semantic dimension and an implicit syntactic dimension. The implicit syntactic dimension means that the antecedents form a sequence rather than a set and that the syntactic component for the succedent is the concatenation of the strings for the antecedents. It should be observed that all rules work unrestricted on the semantic component from the premises to the conclusion. We refer to the resulting sequent calculus as the Nabla calculus.

We use Γ (and Δ) for sequences of categories $A_1 \dots A_n$ ($n > 0$). The rules have sequents of the form $\Gamma \succ A$. The sequent means that if $a_1, \dots, a_n$ are strings of categories $A_1, \dots, A_n$, respectively, then the string that consists of the concatenation of the strings $a_1, \dots, a_n$ is a string of category A. Hence the sequent $A \succ A$ is valid for any category A.

Rules are displayed starting with the conclusion and the premises indented below. There are two rules for / (a left and a right rule) and two rules for \ too. The left rules specify how to introduce a / or a \ at the left side of the sequent symbol $\succ$, and vice versa for the right rules (observe that the introduction is in the conclusion and not in the premises). The reason why we display the rules in this way is that sequents tend to get very long, often as long as a whole line, and hence the more usual tree format would be problematic. Also the conclusion is usually longer than each of the premises.

We note that only the right rule of λ (where $\alpha \rightsquigarrow_\lambda \alpha'$ is λ-conversion) is possible, since only variables are allowed on the left side of the sequent symbol.

$$x : A \succ x : A \qquad =$$

$$\frac{\Delta \succ \alpha' : A}{\Delta \succ \alpha : A} \quad \alpha \rightsquigarrow_\lambda \alpha' \qquad \lambda$$

$$\frac{\Delta[\Gamma] \succ \beta[x \mapsto \alpha] : B \qquad \Gamma \succ \alpha : A \qquad \Delta[x : A] \succ \beta : B}{} \qquad \text{Cut}$$

$$\frac{\Delta[\Gamma \;\; z : B\backslash A] \succ \gamma[x \mapsto (z\ \beta)] : C \qquad \Gamma \succ \beta : B \qquad \Delta[x : A] \succ \gamma : C}{} \qquad \backslash\text{L}$$

$$\frac{\Gamma \succ \lambda y \alpha : B\backslash A \qquad y : B \;\; \Gamma \succ \alpha : A}{} \qquad \backslash\text{R}$$

$$\frac{\Delta[z : A/B \;\; \Gamma] \succ \gamma[x \mapsto (z\ \beta)] : C \qquad \Gamma \succ \beta : B \qquad \Delta[x : A] \succ \gamma : C}{} \qquad /\text{L}$$

$$\frac{\Gamma \succ \lambda y \alpha : A/B \qquad \Gamma \;\; y : B \succ \alpha : A}{} \qquad /\text{R}$$

3 The Categories

As basic categories for the lexicon we have N, G, S and the top category $\bullet$ corresponding to the whole argument (do not confuse the basic category N with the constant N for 'Nick' and so on). Roughly we have that N corresponds to "names" (proper nouns), G corresponds to "groups" (common nouns) and S to "sentences" (discourses). Consider the following lexical category assignments:

John Nick Gloria Victoria : N
run dance smile : $N\backslash S$
find love : $(N\backslash S)/N$
man woman thief unicorn : G
popular quick : G/G
be : $(N\backslash S)/N$
be : $(N\backslash S)/(G/G)$
a every : $(S/(N\backslash S))/G \quad ((S/N)\backslash S)/G$
not : $(N\backslash S)/(N\backslash S)$
nix : S/S
and or : $S\backslash(S/S)$
and or : $(N\backslash S)\backslash((N\backslash S)/(N\backslash S)) \quad (G/G)\backslash((G/G)/(G/G))$
ok : S
also : $S\backslash(S/S)$
so : $S\backslash(\bullet/S)$

4 The Combinators

We introduce the following so-called logical combinators [13]:

$\dot{\mathbf{Q}} \equiv \lambda xy(x = y)$	Equality
$\dot{\mathbf{N}} \equiv \lambda a(\neg a)$	Negation
$\dot{\mathbf{C}} \equiv \lambda ab(a \wedge b)$	Conjunction
$\dot{\mathbf{D}} \equiv \lambda ab(a \vee b)$	Disjunction
$\dot{\mathbf{O}} \equiv \lambda tu\exists x(tx \wedge ux)$	Overlap
$\dot{\mathbf{I}} \equiv \lambda tu\forall x(tx \Rightarrow ux)$	Inclusion
$\dot{\mathbf{T}} \equiv \top$	Triviality
$\dot{\mathbf{P}} \equiv \lambda ab(a \Rightarrow b)$	Preservation

After having introduced the logical combinators we introduce the so-called lexical combinators. There is one or more combinator for each token in the vocabulary, for example the combinator **John** for the token John, **be** and **be′** for be and so on (tokens and combinators are always spelled exactly the same way except for the ′ (possibly repeated) at the end).

In order to display the lexicon more compactly we introduce two place-holders (or "holes") for combinators and constants, respectively. $\bigcirc$ is place-holder for logical combinators (if any) and $\circ$ is place-holder for (ordinary and predicate) constant (if any); the combinators and constants to be inserted are shown after the | as in the following lexicon:

$$\begin{array}{l}
\textbf{John Nick Gloria Victoria} \;\equiv\; \circ \mid J\ N\ G\ V \\
\textbf{run dance smile} \;\equiv\; \lambda x(\circ x) \mid R\ D\ S \\
\textbf{find love} \;\equiv\; \lambda yx(\circ xy) \mid F\ L \\
\textbf{man woman thief unicorn} \;\equiv\; \lambda x(\circ x) \mid M\ W\ T\ U \\
\textbf{popular quick} \;\equiv\; \lambda tx(\bigcirc(\circ x)(tx)) \mid \dot{\mathbf{C}} \mid P\ Q \\
\textbf{be} \;\equiv\; \lambda yx(\bigcirc xy) \mid \dot{\mathbf{Q}} \\
\textbf{be}' \;\equiv\; \lambda fx(f\lambda y(\bigcirc xy)x) \mid \dot{\mathbf{Q}} \\
\textbf{a every} \;\equiv\; \lambda tu(\bigcirc tu) \mid \dot{\mathbf{O}}\ \dot{\mathbf{I}} \\
\textbf{not} \;\equiv\; \lambda tx(\bigcirc(tx)) \mid \mathbf{N} \\
\textbf{nix} \;\equiv\; \lambda a(\bigcirc a) \mid \mathbf{N} \\
\textbf{and or} \;\equiv\; \lambda ab(\bigcirc ab) \mid \mathbf{C}\ \mathbf{D} \\
\textbf{and}'\ \textbf{or}' \;\equiv\; \lambda tux(\bigcirc(tx)(ux)) \mid \mathbf{C}\ \mathbf{D} \\
\textbf{ok} \;\equiv\; \bigcirc \mid \dot{\mathbf{T}} \\
\textbf{also} \;\equiv\; \lambda ab(\bigcirc ab) \mid \dot{\mathbf{C}} \\
\textbf{so} \;\equiv\; \lambda ab(\bigcirc ab) \mid \dot{\mathbf{P}}
\end{array}$$

5 Examples: Syntax and Semantics

Consider the tiny argument (where $\surd$ indicates that the argument is correct):

$$\frac{\text{John is a popular man.}}{\text{John is popular.}} \quad \surd$$

The lexical category assignments to tokens give us the following string / formula association using the sequent calculus:

John be a popular man so John be popular

$\rightsquigarrow$ **so** (**a** (**popular man**) λx(**be** x **John**)) (**be**′ **popular John**)

$\rightsquigarrow$ $\dot{\mathbf{P}}$ ($\dot{\mathbf{O}}$ $\lambda x(\dot{\mathbf{C}}\ (Px)\ (Mx))\ \lambda x(\dot{\mathbf{Q}}Jx))\ (\dot{\mathbf{C}}\ (PJ)\ (\dot{\mathbf{Q}}JJ))$

$\rightsquigarrow$ $PJ \wedge MJ \Rightarrow PJ$

It is really an impressive undertaking, since not only does the order of the combinators not match the order of the tokens, but there is also no immediate clue in the string on how to get the structure of the formula right ("the parentheses").

As expected the resulting formula is valid.

6 Conclusions and Future Work

The multi-dimensional type theory Nabla provides a concise interpretation and a sequent calculus as the basis for implementations. Of course other calculi are possible for the same interpretation. The plans for future work include:

- Investigations of further type constructions for a larger natural language coverage, cf. the treatment of propositional attitudes in [16,17] which also replaces the classical logic with a paraconsistent logic.
- Implementations using constraint solving technologies, cf. recent work on glue semantics [3], XDG (Extensible Dependency Grammar) [5], CHRG (Constraint Handling Rules Grammar) [2], and categorial grammars [4,6,9].

References

1. W. Buszkowski, W. Marciszewski, and J. van Benthem, editors. *Categorial Grammar*. John Benjamins Publishing Company, 1988.
2. H. Christiansen. Logical grammars based on constraint handling rules. In P. J. Stuckey, editor, *18th International Conference on Logic Programming*, page 481. Springer-Verlag, 2002. LNCS 2401.
3. M. Dalrymple, editor. *Semantics and Syntax in Lexical Functional Grammar: The Resource Logic Approach*. MIT Press, 1999.
4. P. de Groote. Towards abstract categorial grammars. In *39th Annual Meeting of the Association for Computational Linguistics*, pages 148–155, 2001.
5. R. Debusmann, D. Duchier, A. Koller, M. Kuhlmann, G. Smolka, and S. Thater. A relational syntax-semantics interface based on dependency grammar. In *20th International Conference on Computational Linguistics*, 2004.
6. M. Kuhlmann. Towards a constraint parser for categorial type logics. Master's thesis, Division of Informatics, University of Edinburgh, 2002.
7. J. Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170, 1958. Reprinted in [1].
8. M. Moortgat. *Categorial Investigations — Logical and Linguistic Aspects of the Lambek Calculus*. Foris Publications, 1988.
9. R. Moot. Grail: An interactive parser for categorial grammars. In R. Delmonte, editor, *VEXTAL*, pages 255–261. Venice International University, 1999.
10. G. Morrill. *Type Logical Grammar*. Kluwer Academic Publishers, 1994.
11. D. Prawitz. *Natural Deduction*. Almqvist & Wiksell, 1965.
12. V. Sánchez. *Studies on Natural Logic and Categorial Grammar*. PhD thesis, University of Amsterdam, 1991.
13. S. Stenlund. *Combinators, λ-terms and Proof Theory*. D. Reidel, 1972.
14. J. Villadsen. *Nabla: A Linguistic System based on Multi-dimensional Type Theory*. PhD thesis, Department of Computer Science, Technical University of Denmark, 1995.
15. J. Villadsen. Using lexical and logical combinators in natural language semantics. *Consciousness Research Abstracts*, pages 51–52, 1997.
16. J. Villadsen. Combinators for paraconsistent attitudes. In P. de Groote et al., editors, *Logical Aspects of Computational Linguistics*, pages 261–278. Springer-Verlag, 2001. LNCS 2099.
17. J. Villadsen. Paraconsistent assertions. In J. Denzinger et al., editors, *Multiagent System Technologies*. Springer-Verlag, 2004. To appear in LNCS 3187, 15 pages.