Proceedings of the 4th International Workshop on

Constraints and Language Processing (CSLP 2007)

Henning Christiansen
Jørgen Villadsen
(Editors)

AUGUST 2007 ROSKILDE UNIVERSITY COMPUTER SCIENCE RESEARCH REPORT #113

Computer Science
Roskilde University
P. O. Box 260
DK–4000 Roskilde
Denmark

Telephone:	+45 4674 3839
Telefax:	+45 4674 3072
Internet:	http://www.ruc.dk/dat/
E-mail:	datalogi@ruc.dk

ISSN 0109–9779

This research report constitutes the proceedings of the *4th International Workshop on Constraints and Language Processing (CSLP 2007)* which is held in conjunction with the 6th International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT 2007), Roskilde University, Denmark, August 2007.

Research reports are available electronically from:

http://www.ruc.dk/dat/

Constraints and Language Processing

CSLP@Context07

4th International Workshop, CSLP 2007
Roskilde, Denmark, 20 August 2007

Proceedings

Edited by

Henning Christiansen
Jørgen Villadsen

Preface

The present volume contains the proceedings of CSLP 2007, the 4th International Workshop on Constraints and Language Processing which takes place in Roskilde, Denmark, on 20 August, 2007.

CSLP 2007 is affiliated with Context07: Sixth International and Interdisciplinary Conference on Modeling and Using Context. We would like to thank the organizers of Context07, especially its workshop chair Stefan Schulz, for hosting the workshop.

Constraints are widely used in linguistics, computer science, and psychology. How they are used, however, varies widely according to the research domain: knowledge representation, cognitive modelling, problem solving mechanisms, etc. These different perspectives are complementary, each one adding a piece to the puzzle. For example, linguistics proposes in-depth descriptions implementing constraints in order to filter out structures by means of description languages, constraint ranking, etc. The constraint programming paradigm, on the other hand, shows that constraints have to be taken as a systematic whole and can thus play a role in building the structures (or can even replace structures). Finally, psycholinguistics investigates the role of constraint systems for cognitive processes in comprehension and production as well as addressing how they can be acquired.

The CSLP@Context07 workshop considers the role of constraints in the representation of language and the implementation of language processing. This theme should be interpreted inclusively: contributions from linguistics, computer science, psycholinguistics and related areas are welcome, and an interdisciplinary perspective is of particular interest.

The collocation with the Context07 conference underlines the application of constraints for context comprehension and discourse modelling. We are honoured to present our invited speaker, Annelies Braffort, who will talk on the modelling of spatio-temporal constraints in sign language processing. We will also express a special thanks to Barbara Hemforth who accepted to send us a late paper which emphasizes the importance of the psycholinguistic dimension in this research.

This volume contains papers accepted for the workshop based on an open call, and each paper has been reviewed by three or four members of the program committee. As editors, we would also like to thank the other members of the organization committee, Philippe Blache and Veronica Dahl, whose involvement has been important for the establishment of the forum around the CSLP workshops.

Revised papers from the 1st CSLP were published as Springer Lecture Notes in Artificial Intelligence (volume 3438).

We want to thank the program committee listed below, the invited speaker, and all researchers who submitted papers to the workshop and all participants in the CSLP workshops 2004 (Roskilde, Denmark), 2005 (Sitges, Spain; with ICLP), 2006 (Sydney, Australia; with COLING/ACL) and 2007, once again in Roskilde affiliated with CONTEXT. We expect to continue the CSLP series, which to us is a stimulating research forum concerning an important, interdisciplinary field, possibly collocating with central conferences for an increased exchange of knowledge. The workshop is supported by the CONTROL project, CONstraint based Tools for RObust Language processing, funded by the Danish Natural Science Research Council, and the Programming, Logic and Intelligent Systems Research Group & the Department of Communication, Business and Information Technologies, Roskilde University, Denmark.

Finally we dedicate this volume to Peter Rossen Skadhauge, who sadly and unexpectedly passed away last year. He has been a member of the program committee for the previous CSLP workshops and co-editor of the mentioned LNAI volume. We will miss Peter personally as well as for his contribution to the CSLP workshop series.

Roskilde, July 2007

Henning Christiansen
Jørgen Villadsen

Organizing Committee

Philippe Blache
Henning Christiansen, co-chair
Veronica Dahl
Jørgen Villadsen, co-chair

Program Committee

Philippe Blache (Provence University, France)
Henning Christiansen (Roskilde University, Denmark), co-chair
Veronica Dahl (Simon Fraser University, Canada)
Denys Duchier (University of Orléans, France)
John Gallagher (Roskilde University, Denmark)
Claire Gardent (University of Nancy, France)
Barbara Hemforth (Provence University, France)
Jerry Hobbs (University of Southern California, USA)
M. Dolores Jiménez-López (Tarragona, Spain)
Michael Johnston (AT&T, USA)
Lars Konieczny (Freiburg University, Germany)
Shalom Lappin (King's College, UK)
Detmar Meurers (Ohio State University, USA)
Véronique Moriceau (Université Paul Sabatier, Toulouse, France)
Gerald Penn (University of Toronto, Canada)
Kiril Simov (Bulgarian Academy of Sciences, Bulgaria)
Jørgen Villadsen (Technical University of Denmark), co-chair
Eric Villemonte de la Clergerie (INRIA, France)

Contents

Sign Language Processing: Modelling of spatio-temporal constraints

Annelies Braffort

LIMSI-CNRS, Campus d'Orsay, Bat 508,
90 403 Orsay cedex, France
annelies.braffort@limsi.fr

Abstract. Sign Language (SL) is the deaf community's first language. SL linguistic functioning is completely adapted to the visuo-gestural channel specificity: 1) Several articulators are simultaneously used (hands, arms, gaze, mimic, head, chest), each of them having one or several specific linguistic roles, and possibly interacting with other ones; 2) Iconic features have a key structural role in shaping signed language discourse; 3) SL show a heavy and consistent use of the "signing space", i.e. the portion of space in which the signs are performed. SL Processing systems should be respectful of SL specificities. Therefore, while modelling the signs of the language, i.e. at the lexical level, all the possible context-driven variations must be considered as they are fully part of the language. At the discourse level, sentence construction rules must also account for the extensive use of space of LSF. We present here some examples of our research on SL lexical and discourse modelling based on spatio-temporal constraints, and we explain how we plan to exploit these models in the context of SL generation by means of a signing avatar.

Keywords: Sign Language**,** lexical model, signing space, spatio-temporal rules, signing avatar.

1 Introduction

Sign Language (SL) is the deaf community's first language. For deaf persons to have ready access to information and communication technologies (ICTs), the latter must be usable in sign language. Such applications will be accepted by deaf users if they are reliable and respectful of SL. Before developing ICT applications, it is necessary to model these features.

LIMSI lab has begun studies on LSF modelling in 1992. Several models have been proposed in the context of automatic SL recognition, mainly on spatio-temporal structure modelling and on lexical modelling. The LIMSI's SL team has recently initiated a signing avatar project called ELSI[1] with the purpose of generating French Sign Language (LSF) and which integrated a number of these models.

This paper describes some of our investigations on SL modelling and the way we plan to exploit these models in the context of SL generation by means of a signing

1 **ELSI**: **E**lsi is **L**imsi's **SI**gner

avatar. Section 2 goes through the main characteristics of LSF and addresses the consequent problems posed for representation. Section 3 describes some of our investigations on LSF lexicon and sentence modelling. Section 4 presents the ELSI software platform as well as the methodology we chose for its development and ongoing evaluation. Section 5 states our progress and discusses some of the prospects of the ELSI platform.

2 SL functioning

The principle characteristic of LSF and of SL in general, is the use of the visuo-gestural channel. Thus, there is a compromise between articulation, visual interpretation and comprehensibility. Signs are articulated with a constant purpose of a economising in articulation, while retaining a meaningful message which can be easily interpreted by another person [8].

The number of body features involved in LSF communication allows for a lot of information at once: Sign Languages (SLs) do not only use hands, but also chest and shoulder movements, eye gaze, facial expression, head movements (§ 2.1.); linguistic studies of LSF, whose a major actor in France is Cuxac and his team [7], [8], show a heavy and consistent use of the "signing space", i.e. the portion of space in which the signs are performed (§ 2.2.); iconicity is also a relevant feature of both its lexicon and its grammar (§ 2.3.). All these characteristics allow a signer to “say”, but also to “show while saying”.

2.1 Simultaneous parameters

The meaning is conveyed by means of body articulators: hand and arm shape movement orientation and location, head shoulder and chest movements, but also gaze (direction, closing) and facial expression. These *parameters* occur simultaneously and are articulated in the space. Each of them can possess a syntactical function and a semantic value, allowing whole parts of sentence in vocal language to be expressed with a single sign.

Figure 1 shows an extract of a LSF story related to a cow looking at a horse while ruminated. This also illustrates one of the way a signer can “show while saying”, and the importance of the gaze: During this kind of utterance, the gaze is never directed to the addressee, but toward a specific point in the signing space, or on his hands or arms.

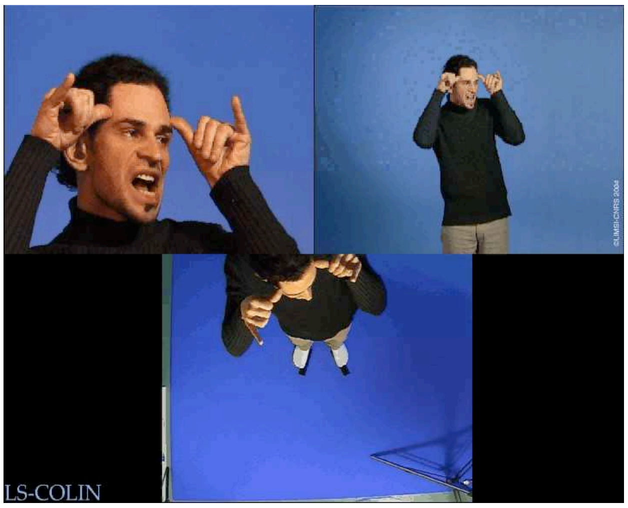


Fig. 1. Example of a LSF production involving the whole body articulators [4].

2.2 Signing space

One of the main properties of SL is the intensive use of the space located in front of the signer, called *signing space* [8], [21]. This space is responsible of the global discourse structure. The entities of the discourse (persons, objects, events…) are often located in this signing space. Then spatio-temporal structures are used at the sentence level. For example, spatial relations between entities are generally established without using *lexical signs*. This is the spatio-temporal organisation of specific signs named "proforms" which allows interpretation. An example is given in figure 2b, showing a bird (left hand proform showing the beak) located *on* (relative positions of the two hands) a fence composed of two horizontal bars (right hand proform).

The sign order is less important than their arrangement in space. But we can draw out some general principles. The larger and the more static objects are signed before the little or moving ones. The general rule is that one signs from the more general, the context, to the more detailed, the action.

2.3 Iconicity

An important property of SL is the iconicity. Because the language is highly contextual, a lot of signs are articulated in *highly iconic structures* (HISs). Cuxac has proposed a categorisation of these linguistic structures [7], [8]. He has distinguished three kinds of structures that he called *transfers*: The *size and shape transfer* (TTF), which is used to describe the shape of a person, an object or a place (Fig. 2a), the *situational transfer* (TS), which is used to show the displacement of a person or an object relatively to a stable locative reference (Fig. 1 and 2b), and the personal transfer (TP), where the signer "becomes" one of the person or object of the discourse (Fig. 2c). TS and TP can be combined in double transfers (DT), such as in figure 2d. Sometimes, some parts of lexical signs can be combined with a DT to form a semi transfer (SM) (Fig. 2e).

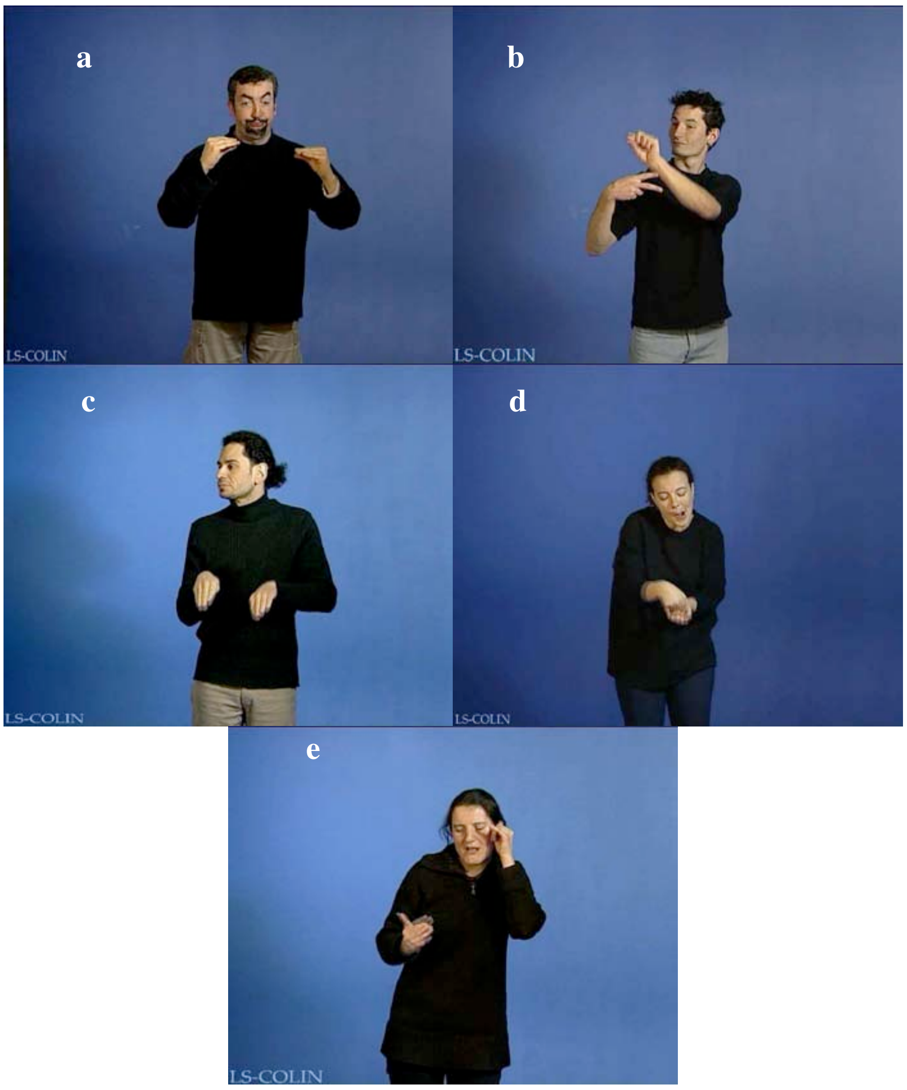


Fig. 2. Examples of transfers - a: A TTF, “spread pastry”; b: A TS, “a bird on a fence”; c: A TP, “a horse galloping”; d: A DT, “a ruminating cow”; e: A SM, “the cow (left hand) is waiting (right hand)” [4].

3 SL modelling: what to tackle?

Figure 3 give some examples of what an automatic generation system should be able to produce. These animations have been designed by our computer artist. The number of body features involved (hands, but also chest, shoulders, eye gaze, facial expression, head movements - Fig. 1a), the heavy and consistent use of the signing space (Fig. 1c and 1d), and iconicity of both lexicon and grammar (Fig. 1b) should be tackle in SL processing systems.

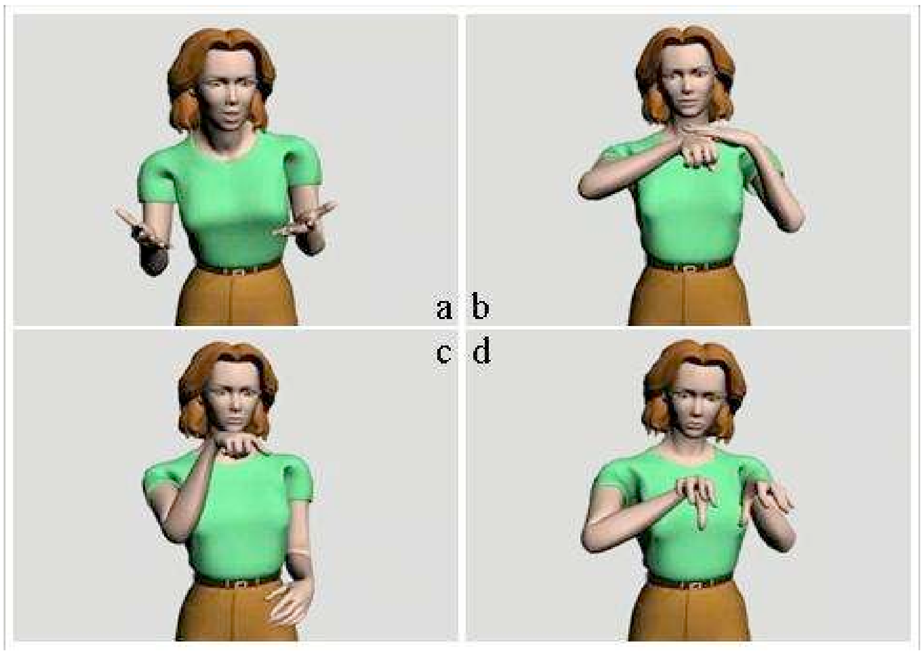


Fig. 3. Examples of LSF signs: (a) [WHAT?], (b) [BIRTH], (c) “here”, (d) [GROUP].

Formalising such features for automatic sign generation is not trivial. This section present some of our investigations both at the lexical (§ 3.1) and sentence (§ 3.2) levels.

3.1 Lexical modelling

The major current model suitable as an input to a computer generation platform is SigML [10], [19], which is basically a machine-readable form of HamNoSys. HamNoSys is virtually the only formal model for signs and has already been encapsulated in sign generation software (eSign, ViSiCast) [15], [16], [17]. It is based on Stokoe's 4-parameter model (hand shape, hand orientation, location and movement). But we see three limitations to these parameter models that we would like to avoid.

3.1.1 Parameter model limitations

Parameter models describe every sign with the same fixed set of parameters, each of which must be given a discrete value. However, we argue that not all signs require all parameters, and that not all the parameters that are needed can be given at the same time in the same way. The trouble when filling a parameter list with values is that all parameters inherit the same status. Yet often, some are crucial to the sign in that changing them would result in a loss of the whole sign, whereas others are only given so as to enable, say, a signing avatar to perform the target sign but could well be specified differently. Instead of over-specifying the orientation parameter, we suggest that the sign contents be constrained enough to define the target sign, but that whatever is not necessary be banned from its description.

Parameter models consider parameters separately. Each of them is assigned a distinct value, regardless of the other parameters. It means that these assignments could all be carried out simultaneously, in other words all at once and independently. Though, this does not account for inter-parameter dependencies. Moreover, two different parameters could well depend on a common non-parameter object, such as in [BUILDING] (Fig. 4a). The strong hand moves along and close to a line, say *L*. Its palm is constantly facing *L* and the weak hand's location and orientation is defined as being symmetric to those of the strong hand's, with respect to *L*. Both location and palm orientation of both hands depend on the same object *L*. Although it is obviously crucial to the sign as a great part of the description depends on it, *L* is no parameter in Stokoe's sense. It is why we call *L* a *non-parameter common dependency*. To account for all the cases mentioned above, we claim that any part of a sign description should even be allowed to make use of other parts of the same description. This way, internal dependencies become part of the description.

Above all, using C. Cuxac's theory of iconicity [8] as a framework for ours, it has become obvious that the many possible influences of discourse context on the signs that are used cannot be ignored in the process of devising a lexical description model. A great part of the beauty of SLs and their power in conciseness comes from the potential for signs to be altered according to the context in which they are used, thereby switching discourse from a conventional sign flow to HISs. For instance, the sole sign [BUILDING] can be used to sign the phrase "large building" in LSF, only the distances between the hands will be greater than the ones involved in the plain conventional [BUILDING] sign (plus the signer will probably also puff his cheeks and raise his elbows). Formalising such features for automatic sign generation is not trivial. An HIS can not only alter the location or the hand shape involved in a sign, but also a path, a direction, eye gaze, etc. Virtually, anything can be acted upon, and these actions being commonplace in SL, we claim a description model should allow signs to behave accordingly. Back to the example above, describing [BUILDING] without making the distance between the hands responsive to the contextual size weakens –if not destroys– the sign's re-usability.

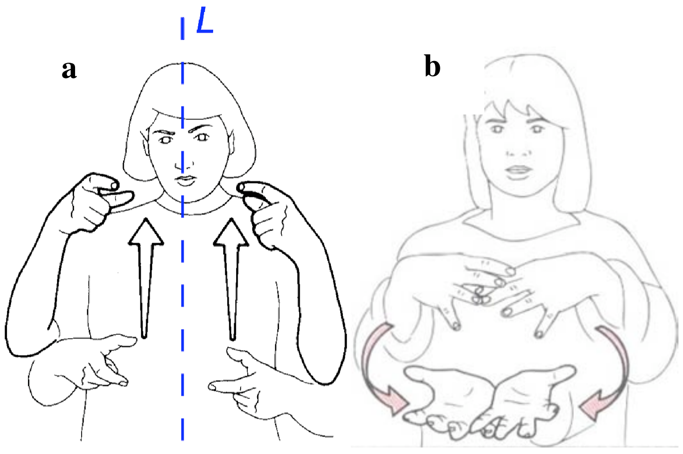


Fig. 4. a) [BUILDING], b) [BALL]. Pictures extracted from [23].

3.1.2 A lexical model based on geometric constraints

Our model uses *geometric constraints* [11, 12]. Signs are no more regarded as tuples of universal parameters like in Stokoe-based approaches [24], but rather as dynamic spatial geometric figures. A description may build any useful set of geometric objects like planes or points in space, and constrains body segments to describe positions and gestures. For instance, when hands are symmetric, we rather build the symmetry and not repeat parts of the description for both hands.

A statistical analysis of LSF [2] shows that geometric objects are commonplace in sign languages: most two-handed signs include a symmetry or a parallelism, a large number of signs sketch out or move along planes, lines or circles... Almost every sign makes people use geometric notions in spontaneous descriptions. Realising how essential geometry is in every sign, we no longer regard it as a universal list of placeholders but as a geometric construction in space. Like any spatial geometric figure, objects like points, planes, etc. are built step by step using a formal description language in an unambiguous way to form a desired figure. Specifying a sign requires that, from an empty space (the signing space), the necessary objects are built one by one, and then constrained as the description goes to create a dynamic figure, to which parts of the body then just have to be connected.

No over-specification

The first challenge above becomes easy: as we are free to build whatever we decide, there is no need to build superfluous objects. Starting in an empty space, we just have to build as much –and indeed as little– as required for the description. Nothing is added for the only sake of fitting into the model.

For example, to describe the sign [BALL], we will need to build different things:

- A symmetry plane through the centre of the ball;
- A starting point for the path of the strong hand, in this plane;
- A semi-circle path for the strong hand;

– A path for the weak hand, symmetric to the first path, across the symmetry plane.

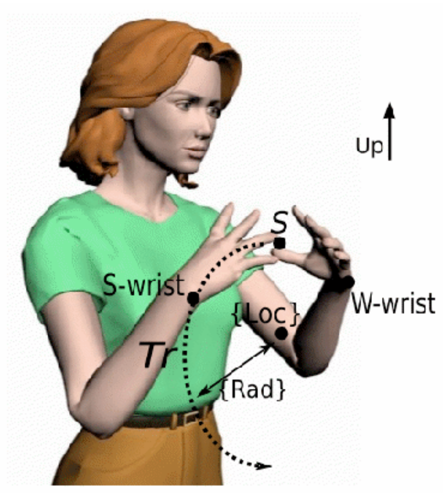


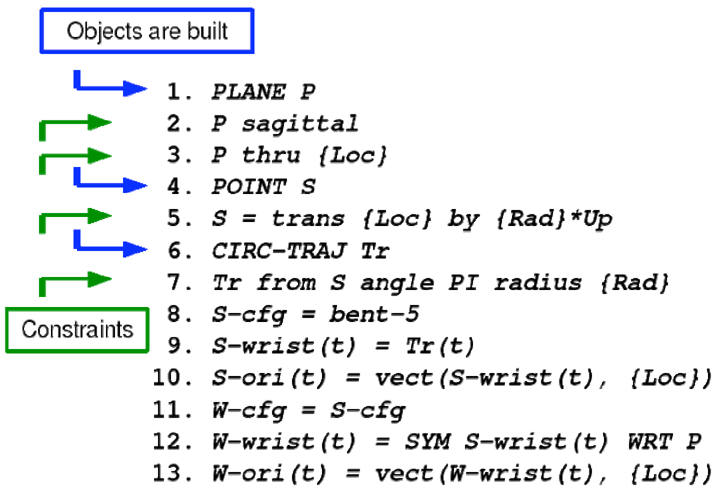
Fig. 5. Geometric objects for [BALL].

Figure 5 shows geometric objects present in sign [BALL], superimposed on a snapshot of the signing avatar under development in LIMSI [13]. Only the symmetry plane is missing, as it would simply have covered half the picture if it were inserted.

Reveal sign structures

This geometric approach also beats the second challenge. One object being built at a time, descriptions are iterative and sequential. Each object can well refer to one or several other objects already in place, like the starting point above is said to be in the plane defined in a first place. Allowing objects to rely on other objects this way makes the whole process account for internal object dependencies: when an object is specified with a reference to another (or others), the former is explicitly dependent on the latter. Thus, contrary to parametric models, a sign-dependent structure is visible in each description.

Figure 7 shows the type of language we actually plan to describe the signs with. It includes object building and different constraint statements, each on a separate line.



```
 1. PLANE P
 2. P sagittal
 3. P thru {Loc}
 4. POINT S
 5. S = trans {Loc} by {Rad}*Up
 6. CIRC-TRAJ Tr
 7. Tr from S angle PI radius {Rad}
 8. S-cfg = bent-5
 9. S-wrist(t) = Tr(t)
10. S-ori(t) = vect(S-wrist(t), {Loc})
11. W-cfg = S-cfg
12. W-wrist(t) = SYM S-wrist(t) WRT P
13. W-ori(t) = vect(W-wrist(t), {Loc})
```

Fig. 6. Description for [BALL], using the geometric model.

It consists of a list of lines, whose order cannot randomly be changed. If a line includes a reference to other objects, it has to come after the ones defining these objects. This happens for instance on line 7, where the path of the strong hand is constrained to start at point *S*, which itself is defined earlier in the description.

In a less direct sense of the term, we can infer that the path of the strong hand also depends on the ball's centre point *{Loc}* (see next session for a word on why *{Loc}* is the centre point): line 5 tells us that the starting point *S* depends on *{Loc}*. But, and as we just stated, the strong hand path depends on *S*. Hence transitively, we may say that the path of the strong hand depends on the centre of the ball. We can picture that well, as if we move the centre of the ball, the path taken by both hands (hence the strong hand) will necessarily change accordingly.

If we push this property further to a 3-step transitivity, we could infer from the symmetry between the hands that the path of the weak hand also depends on the ball's centre. We call these *indirect dependencies*.

Enable real productive sign units

It has always been intended that the model handle iconicity in sign descriptions. Enabling iconicity in signs can be done by extending the language with a new type of reference. Every time a value or an object is expected in the description, a call to a context element can be placed instead. For instance, instead of specifying an arbitrary distance between the hands' positions in the description for [BUILDING], we may refer to an external reference called size. This way, whenever the description is used to perform the sign in discourse (i.e. in context), it can be tagged with a size attribute, so that the distances are changed accordingly, with no extra rule about how to sign "big building" or "small building". This brings us to allow elements within the description to depend on values that are "outside" the lexeme itself. In fact, they are to be found in –or given by– the context when the description is used to utter an actual sign.

On figure 6, these external dependencies are noted in curly brackets. *{Loc}* illustrates a dependency on the location of the ball in the signing space; *{Rad}* represents the radius of the ball.

3.2 Sentence modelling

To date, most research on sign language views an utterance as a sequence of isolated, sometimes coarticulated signs. The models used (generation or recognition of finger spelling, signed English, etc.) may be far removed from real SL mechanisms.

A small number of projects, mainly generation-related, nevertheless credit some of the syntactico-semantic features of SL. Some work has been initiated with the ViSiCAST project to include use of proforms and signing space [22]. Classifier predicates, as generated by [18], are close to the situational transfers described by Cuxac. But we do not find anything which is really close to the richness of the different kinds of HISs reported in his book [8].

Because SLs employ visual-gestural channels to transmit and receive messages, their organisation hinges on pertinent use of space and on multilinear structures built of discrete or non discrete, iconic or non iconic units. Our models seek to credit the specific nature of sign language. Their utterance representations, designed to reflect SL specificities, therefore differ from those used for vocal languages.

Our claim is that computerised modelling must includes signing space models as a basis for language representation. In [2], [3], signing space modelling is used to interpret utterances containing proforms and directional verbs in an automatic sign recognition system. More recently, signing space modelling has been proposed in the context of FSL analysis using image processing techniques [9], [20]. For generation, a semantico-cognitive formalisation has been proposed in [5] in order to represent some iconic properties related to iconicity within a signing space modelling scheme.

3.2.1 Signing space modelling as an extended scene graph

We use signing space representation as the discourse structure representation. This representation can have more or less fine levels of granularity, as needed. It can be a simple "spatial memory" containing a history of relevant locations in space [2], [3], with a list of associated entities; or, in more complex cases, a set of information on inter-entity relationships and their semantico-cognitive characteristics [5].

Our representation is made up of appropriate entities and a three-dimensional Euclidian space structured as an *extended 3d scene graph*.

The signing scene is made up of entities with specified locations, and the relationships between them. Some of the locations are predefined and generic, such as those used to represent time (past: straight, horizontal backward line away from the signer; present: signer location; and future: straight, horizontal forward line away from the signer).

All entities and their relationships are represented as a graph with as many nodes as there are entities in the discourse. The nodes contain information about shape, size, location, orientation and semantico-cognitive kind of entity, in accordance with the principles described above. The arcs of this graph contain information on spatial relationships.

3.2.2 Spatio-temporal representation using qualitative temporal properties and spatial constraints

We have associated this representation with a set of methods for manipulating entities and their relationships, to allow updates of the signing scene, and to manage a set of sentence generation rules.

To date, only a limited number of rules has been devised. These rules are concerned with use of proforms to express spatial relationships between entities. The proposed spatio-temporal structures are based on the assumption that the natural order of SL production is Localiser-Localised entity [8]. Such structures are expressed in a formal language that is currently being developed. This language facilitates expression of qualitative temporal properties with more or less severe spatial constraints. It is based on Allen's interval logic [1] and enriched with data types and operators for manipulating sign components and spatial data.

The examples given below illustrate representation of a "proform structure". This kind of structure is intensively used in SL discourse and dialogues. It allows entities to be spatialised in the signing space. This type of structure is made up of *lexical signs* that designate entities, denoted as [SIGN], *proforms* for spatialising entities at a given location in the signing space, denoted as PF(sign, place), and a *gaze* unit, denoted as GAZ(place), which serves to "instantiate" or "reactivate" a location in the signing space, specifically before a proform is placed there. Once the entity is signed, a brief gaze is directed toward the future location of the proform; and the proform is produced at that location. In the corresponding sequence of events, we observe that certain phenomena take place in parallel. These can be represented in a partition-type scheme in which time elapses from left to right (Fig. 7).

- The first tier of this diagram represents the direction of the signer's gaze. Unless his eyes are closed, his gaze is always directed toward some location (light grey area). The dark grey area represents the point in time at which his gaze is forced to focus on a point P of the signing space (GAZ(P)).
- The second tier represents the moment in time at which the standard sign [S] representing the entity is produced. This occurs at the start of the sequence.
- Tier 3 depicts the proform PF(S, P), whose configuration is selected according to kind of entity S and to the context, and is located at point P.

The duration of each event is based on statistical values derived from video corpus analysis. Note that the signer's gaze may focus on point P shortly before the end of S and turn away from this point shortly before the end of the proform. The constraints implemented here are flexible, as shown by different shades of grey on the "Gaze" tier of figure 7.

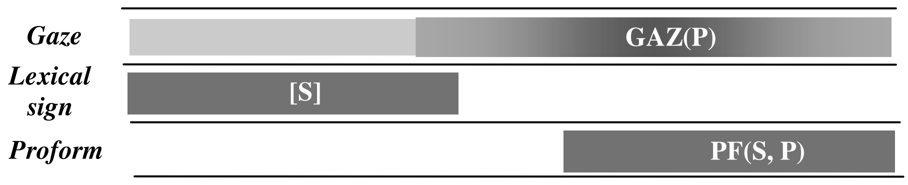


Fig. 7. Partition scheme for representing a proform structure.

This structure can be formalised as follows, using Allen's temporal relation ships:

```
POINT P ;
INTERVAL TGAZE, TSS, TPF ;
(TSS < TGAZE) v (TSS m TGAZE) v (TSS o TGAZE) ;
(TPF e TGAZE) v (TGAZE = TPF) v (TGAZE o TPF) ;
TGAZE.direction = Vect(eyes, P) ;
TPF.location = P ;
TPF.handshape = Select(TSS.proformList) ;
```

Meaning of Allen relationships:

```
v : or
< : precedes
m : immediately precedes
o : partially overlaps
e : completely overlaps at end
```

The first two lines declare a 3d point in space and three temporal intervals. The next two lines describe the temporal relationships between these intervals, using Allen's relationships. They are followed by two further lines expressing the spatial constraints within the intervals. The last line associates with the proform a list of possible proforms for the entity represented by S. A proform is selected from this list only when the context is adequately specified.

4 SL generation & animation software on the way...

An avatar animation platform is presently under development at LIMSI. The lexical signs involved in a sign utterance are first ordered, then performed according to context and output to a video. Figure 8 sketches out the structure of the ELSI sign production software.

Sentence generation (M1) is based on a model of the signing space (K2) and uses spatio-temporal structures (K1). The sign generation module (M2) uses descriptions of the wanted signs (K3).

The animation engine module (M3) is taking as input a formal XML representing the sentence production. Its role is to animate ELSI in a separate window. It uses a hierarchical description of the avatar (K4) and a bank of animated signs performed by our computer artist.

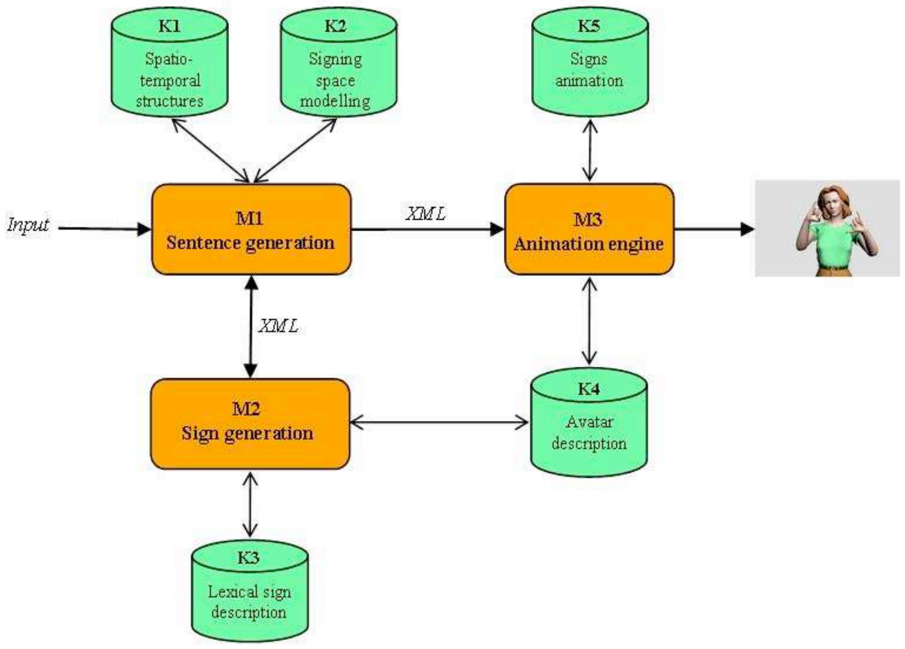


Fig. 8. Structure of the ELSI animation platform.

Producing an LSF clause consists in producing a sequence of LSF signs that suits the LSF grammar defined by means of the spatio-temporal structures representations. We use a simple *model of the signing space* for now. It contains the signed units and their respective locations, orientations and sizes. For the moment being, it allows production of isolated *gap clauses* with predefined format and use of both manual and non-manual signs. Production is carried out by signing the units back to back. We chose an incremental approach so that each module can be tested independently and every step forward guaranteed the reliability of its basis:

1. Design a *virtual character* (K4) and *predefined units* (K5) and check the fluidity of a single animation;
2. Try *sequences* of those units and check the fluidity of the interpolated transitions (M3);
3. Replace a predefined unit by a *lexical description* processed by M2;
4. Add context values to the input of M2;
5. Add more complex spatio-temporal structures representations in K1;
6. Extend the representation power of the signing space modelling (K2)…

All the modules and knowledge bases are not yet completely implemented. We are currently at the step 2: evaluation of the automatic coarticulation process, that will need a specific study based on LSF corpus analysis in order to design dedicated collision detection and a planning process.

5. Conclusion, current progress and prospects

This paper describes our approach to constructing the models and formalisms required to develop SL applications for ICTs. Deaf-friendly SL models must reflect the specificities of the language – use of space to support discursive concepts, iconicity as the structuring principle and simultaneous contribution of various body parts – shoulders, chest, face and eyes – that supplement hand movements.

A good evaluation of the models will currently be their integration with the platform of virtual signation under development in LIMSI. It will then be advisable to estimate acceptability, with the linguistic direction, of the generated signs. Also we will be interested particularly in the precision of the articulations of the avatar's skeleton, the fluidity of his movements and the quality of the transitions between the productions.

Another promising perspective of our work is related to the LS-Script project [14], whose objective is to build a written form of LSF. This study focuses, among other aspects of signs, on geometric features contained in them. Even more importantly, it includes discussions about what is "core" to a sign and what can be seen as "peripheral". It is vital indeed, with the goal of offering a writing system usable for taking quick notes, to be aware of what must end up on the paper and what elements may be left out.

Finally, high level models of the signing space, along with a representation of the FSL lexicon should be shareable for SL analysis and synthesis domains and thus afford synergies that already seem promising [6].

References

1. Allen JF. (1990) Towards a general theory of action and time. In Allen J., Hendler J., Tate A. (eds.) Readings in planning, Kaufmann, San Mateo, pp. 464-479
2. Braffort A. (1996). Reconnaissance et Compréhension de gestes, application à la langue des signes. PhD thesis, Université Paris-XI Orsay.
3. Braffort A. (1996). ARGo: An Architecture for Sign Language Recognition and Interpretation. In: Proceedings of Gesture Workshop on Progress in Gestural Interaction. Pages: 17 - 30 ISBN:3-540-76094-6
4. Braffort A., Choisier A., Collet C. and Lejeune F. (2004). « Presentation of three French Sign Language Corpora ». In: In: Gesture in human-computer interaction and simulation, LNCS/LNAI vol 2915/2004.
5. Braffort A., Lejeune F. (2006), "Spatialised semantic relations in French Sign Language: toward a computational modelling". In: Gesture in human-computer interaction and simulation, LNCS/LNAI vol 3881/2006.
6. Braffort A. and Dalle P. (to appear), Sign Language Application: preliminary modelling. In: UAIS Journal, Special Issue on Emerging Technologies for Deaf Accessibility in the Information Society, Springer.
7. Cuxac, C. (1999). French sign language: proposition of a structural explanation by iconicity. In A. Braffort, R. Gherbi, S. Gibet, J. Richardson and D. Teil, (Eds.), Lecture Notes in Artificial Intelligence 1739, Springer: Berlin, pp. 165-184
8. Cuxac C. (2000), « La Langue des Signes Française (LSF) – Les voies de l'iconicité ». In: Faits de Langues vol 15-16, Ophrys.

9. Dalle P., Lenseigne B. (2005). Vision-based sign language processing using a predictive approach and linguistic knowledge. In: IAPR conference on Machine Vision Applications – MVA Tsukuba Science City, Japon,. IAPR, pp. 510-513
10. Elliott, R., Glauert, J. R. W., Jennings, V. and Kennaway, J. R (2004), "SiGML Notation and SiGMLSigning Software System"., Workshop on the Representation and Processing of Sign Languages, Proceedings of LREC 2004, Portugal.
11. Filhol M., Braffort A. (2006), "A sequential approach to lexical sign description", Second Workshop on the Representation and Processing of Sign Languages: Lexicographic matters and didactic scenarios, LREC 2006, Italy.
12. Filhol M., Braffort A. (2006), "Sign Description - How geometry and graphing serve linguistic issues", TISLR 2006, Brasil.
13. Filhol M., Braffort A. and Bolot L. (2007). "Signing Anatar: Say hello to Elsi!. In: Gesture in human-computer interaction and simulation, LNCS/LNAI vol to appear/2007.
14. Garcia B., Boutet D., Braffort A. Dalle P. (2005), "Sign language in graphical form: methodology, modeling and representations for gestural communication". In: Interacting Bodies, ISGS 2005, France.
15. Hanke T. (1989). HamNoSys - an introductory guide. Signum Press, Hamburg.
16. Hanke T. et al (2002), ViSiCAST deliverable D5-1: interface definitions, ViSiCAST project report, http://www.visicast.co.uk.
17. Hanke T. (2004). HamNoSys for representing sign language data in language resources and language processing contexts. In: Streiter, Oliver / Vettori, Chiara (eds): LREC 2004, Workshop proceedings: Representation and processing of sign languages. Paris : ELRA - pp. 1-6
18. Huenerfauth M. (2006), Generating American Sign Language Classifier Predicates For English-To-ASL Machine Translation, Doctoral dissertation, University of Pennsylvania
19. Kennaway R. (2004), "Experience with and Requirements for a Gesture Description Language for Synthetic Animation". In: Gesture-Based Communication in Human-Computer Interaction, LNCS/LNAI vol 2915/2004, Selected Revised Papers of GW'03.
20. Lenseigne B., Dalle P.(2005) Using signing space as a representation for sign language processing, in Gibet S., Courty N., Kamp JF. (eds) Lecture Notes In Artificial Intelligence 3881. Springer, Berlin Heilderberg, pp 25-36
21. Liddell S. (2003). Grammar, gesture and meaning in American Sign Language. Cambridge Univ. Press, Cambridge.
22. Marshall I., Safar E. (2004), « Sign Language Generation in an ALE HPSG, in Proc. Of HPSG04 conference, Bergium.
23. Moody, B. Moody B. (1986). La langue des signes. Dictionnaire bilingue élémentaire. vol.2, IVT, Paris.
24. Stokoe W. (1960). "Sign Language Structure: An Outline of the Visual Communication System of the American Deaf". Studies in Linguistics, NY.

Linguistic Context Solving with Membranes. A Preview

Gemma Bel-Enguix and M. Dolores Jiménez-López

Research Group on Mathematical Linguistics (GRLMC)
Rovira i Virgili University
Pl. Imperial Tárraco, 1, 43005 Tarragona, Spain
gemma.bel@urv.cat
mariadolores.jimenez@urv.cat

Abstract. Membrane Systems [24] and Brane Calculi [9] provide a good framework for modelling context in linguistics. Several context-based linguistic branches are not easily formalizable and implementable, due to difficulty of dealing with environments and actual circumstances. We claim that the combination of the bio-inspired models based on the behaviour of cells and some tools taken from constraint grammars and constraint solving can help to define a new formal approach to "contextual" linguistics.

1 Introduction

The new idea introduced in this paper comes from (or is based on) the conjunction of three theories: *Membrane Systems, Brane Calculi and Constraint Solving.*

Membrane Systems (MS) were introduced by Păun [24] as a powerful generation device based in the behaviour of cellular membranes, and developed during almost a decade by an increasing number of papers. Membrane computing can be included in the area of natural computing. Despite its microbiological inspiration, the model has to be described as a mathematical and formal computational device.

On the other hand, Brane Calculi (BC) [9] are a new type of calculi that have been especially conceived to model the behaviour of biological membranes. Brane calculi is closely related to BioAmbients [26] and Mobile Ambients [8].

It seems, then, that membrane computing (MC) and brane calculi share the same biological referent and they could be understood almost as the same formal device. Indeed, they are closely connected and some interaction between both fields has been performed [7, 10]. However, there are some interesting differences between MC and BC:

– MC is mainly focused on the elements inside the membranes. In fact, the first formalizations of MC did not take into account what happened with membranes, but only what they generated. Later, the idea of dynamic systems, with operations in membranes, was introduced, but the main focus of the theory is still what the system produces. Nevertheless, BC is more concerned in the behaviour of membranes, in the interaction between them, forgetting what they have inside.

– MC is a formal model only primarily inspired in a biological entity, which does not care about the biological coherence of the rules. BC tries to formalize biological operations, and is closer to systems biology, even if it makes use of algebra.
– MC is, by definition, a maximal parallel system, whereas BC performs, in each computational step, only one instruction.

Whereas membrane systems are focused in the computational power of membranes and Brane calculi highlight the biological referent, we go one step further: *the use of membranes in linguistics and context management*. The main features we are taking into account are the following:

– For us, membranes are not cellular objects or generative mechanisms, but *contexts*: environments with specific configurations that can modify elements and processes inside them.
– We are interested in interactions between membranes, like dynamic membrane systems or brane calculi. But we also take into account the final productions, like classical membrane systems do.
– Membranes can help to start an approach to context solving, and also to those parts of linguistics which have the context as a main component. The different resolution of syntactic and semantic operations in different environment is still an open problem both for linguistics and for the implementation of linguistic theories. Context is also a difficulty for linguistic computational simulation. By context solving we define the attempt to find the correct instance of a linguistic problem depending on the context where it is placed and the surrounding contexts it has.
– This work could be said to be closer to brane calculi because of the final goal, management and resolution of context. However mathematical formalisms are directly taken form membrane systems with the contribution of several operations based on BC to describe interaction among membranes.

For linguistics, the main feature and advantage of membranes over the other generative methods is that the membranes can be understood as contexts, providing a suitable framework for the formalization of semantics, pragmatics and interaction between different agents or contexts. Contexts may be different words, persons, social groups, historical periods, languages. They can accept, reject, or produce changes in elements they have inside. Membranes can be also defined as constraint stores, which determine different worlds.

From here, we think constraint solving [18] can be also a model for context solving with membranes. Although constraint solving is far away from our simple mechanism to deal with contexts, we think they can provide some tools for the formalization of operations with contexts, as suggested by [13].

The paper is organized as follows. In section 2, we present a definition of membrane systems and the definition of context solving membranes systems.

Section 3 provides some operations taken from brane calculi. Finally, section 5 shows how some processes of context resolution can be modelled by membranes.

2 P systems: Generalities

Membrane systems provide a powerful framework for formalizing any kind of *interaction*, both among agents and among agents and the environment. An important idea in membrane systems is that generation is made by evolution, when the configuration of membranes undergoes some modifications, given by certain rules. Therefore, most of evolving systems can be formalized by means of membranes.

Membrane systems, as a computational model based in biology, consist of multisets of objects which are placed in the compartments defined by the membrane structure –a hierarchical arrangement of membranes, all of them placed in a main membrane called the *skin membrane*– that delimits the system from its environment.

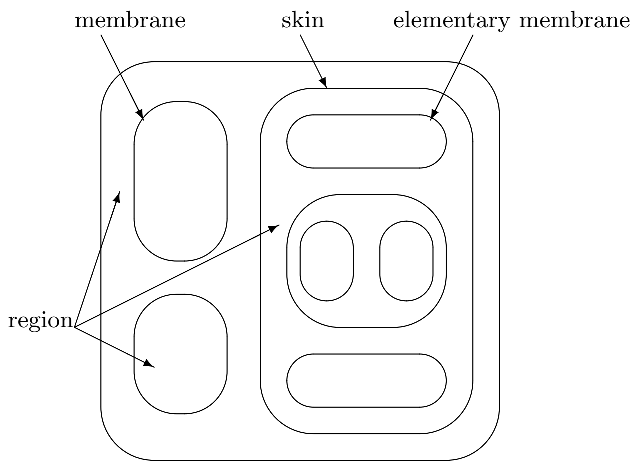


Fig. 1. A Membrane Structure.

Each membrane identifies a *region*, the space between it and all the directly inner membranes, if any exists. Objects evolve by means of reaction rules also associated with the compartments, and applied in a maximally parallel, non-deterministic manner. Objects can pass through membranes, membranes can change their permeability, dissolve and divide.

Definitions of membrane systems can be found in [24] and [25]. In general, membrane systems are a distributed parallel computational model based in the concept of *membrane structure*. Such structure is represented by a Venn diagram where all the sets, *membranes*, are inside a unique *skin membrane*. A membrane without any membrane inside is called *elementary membrane*. Every membrane delimits a *region*. *Objects*, placed in these regions, are able to evolve travelling to other membranes or being transformed in different objects. Unlike other similar computational devices – as Grammar Systems and NEPs – that consist of two different types of configuration states – evolution and communication –, Membrane Systems have only *evolution rules*. Therefore, the rules in membrane systems regulate both: changes in the system and communication between membranes.

Figure 1 shows the graphical intuitive idea on how a membrane structure is. Membranes are usually represented by the sign [ ], and they are labelled with a number between 1 and the number n of membranes in the system. For example, the structure in Figure 1 has the shape $[\ [\ ]_2\ \ [\ ]_3\ \ [\ [\ ]_5\ [\ [\ ]_8\ [\ ]_9\ ]_6\ [\ ]_7\ ]_4\ ]_1$.

Definition 1 *A membrane system Π is defined as a construct:*

$$\Pi = (V, \mu, w_1, ...w_n, (R_1, \rho_1), ..., (R_n, \rho_n), i_o),$$

where:

- *V is an alphabet; its elements are objects;*
- *μ is a membrane structure of degree n;*
- *w_i, $1 \leq i \leq n$, are strings from V^* representing multisets over V associated with the regions 1, 2,...n of μ;*
- *R_i, $1 \leq i \leq n$, are finite sets of evolution rules over V associated with the regions 1, 2,...n of μ; ρ is a partial order relation over R_i, $1 \leq i \leq n$, specifying a priority relation between rules of R_i.*
- *i_o is a number between 1 and n, which specifies the output membrane of Π.*

The *degree of a system* is the number of membranes it has (cf. [24]). It is denoted by *degree*(μ).

2.1 Relations between Membranes

The way the membranes are related to the others is important in the moment they have to interact, and also in the configuration of the communication we are going to deal with later. There are mainly tree types of relations: *nesting*, *sibling* and *command*.

1. Nesting. Given two membranes M_1, M_2, it is said M_2 to be nested in M_1 when it is inside M_1. The outer membrane M_1 is called *parent membrane* and the inner membrane M_2 is called *nested membrane*. It is denoted $M_2 \subset M_1$: $[_1\ [_2\ ]_2\ ]_1$. The notation $\subset M_1$ refers to every membrane nested in M_1.

Nesting is a strict order, which satisfies:

a) It is not reflexive, by definition.
b) It is asymmetric. If $M_n \subset M_m$, then it is not the case that $M_m \subset M_n$.
c) It is transitive. If $M_n \subset M_m$ and $M_m \subset M_j$, then $M_n \subset M_j$.

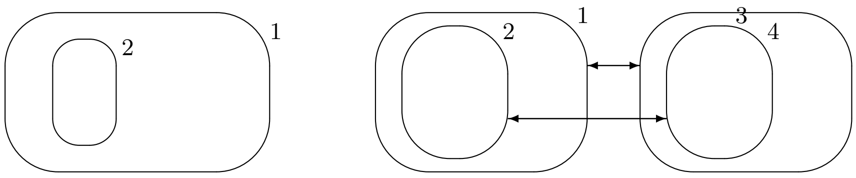


Fig. 2. Nesting and Sibling.

2. Sibling. Two membranes M_n, M_m are related by sibling, if they satisfy:

i. they are adjacent or nested in adjacent membranes, and
ii. they have the same depth.

Sibling is denoted $M_n \approx M_m$. Namely, in a membrane system denoted as $[_0\ [_1\ [_2\ ]_2\ ]_1\ [_3\ [_4\ ]_4\ ]_3\ ]_0$, $M_1 \approx M_3$ and $M_2 \approx M_4$. The notation $\approx M_n$ refers to every sibling membrane for M_n.
Sibling satisfies the following properties:

– It is reflexive, by definition. $M_n \approx M_n$.
– It is symmetric. If $M_n \approx M_m$, then $M_m \approx M_n$.
– It is transitive. If $M_n \approx M_m$, and $M_m \approx M_i$, then $M_n \approx M_i$.

3. Command. Given two membranes M_n, M_m, M_n commands M_m iff:

i. they are not nested,
ii. both are nested in a membrane M_j,
iii. $deg(M_n \subset M_j) = 1$, $deg(M_m \subset M_j) > 1$

Command is denoted $M_n \lhd M_m$. In the system $[_0\ [_1\ [_2\ ]_2\ ]_1\ [_3\ [_4\ ]_4\ ]_3\ ]_0$, M_1 commands M_4 and M_3 commands M_2. The notation $\rhd M_n$ refers to every membrane commanded by M_n.

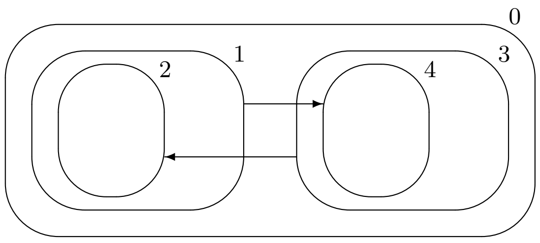


Fig. 3. Command.

Command is a strict order:

a) It is not reflexive, by definition. It is not $M_n \lhd M_n$.

b) It is asymetric. If $M_n \lhd M_m$, then it is not the case that $M_m \lhd M_n$.
c) It is transitive. If $M_n \lhd M_m$ and $M_m \lhd M_j$, then $M_n \lhd M_j$.

For two given membranes M_s, M_z, if $M_s \approx M_z$, then $M_s \lhd (\subset M_z)$, and $M_z \lhd (\subset M_s)$. This means that, if two membranes are related by sibling, then each one of them commands the membranes nested inside the other.

For two given membranes M_s, M_z, if $M_s \lhd M_z$, then $M_s \lhd (\subset M_z)$. This is, if a membrane M_s commands a membrane M_z, it also commands every membrane nested in it.

The relationships of adjacency and nesting will be very important in the development of this paper.

3 Brane Calculi: Operations with Membranes

Operations are directly inspired in the ones of brane calculi, with several differences: a) BC introduces a dinstinction between bitonality and monotonality (see [9]). We dismiss such distinction because of the lack of lingusitic relevance. In linguistic membranes, every operation is monotonal. From here, b) there is only one level in membrane extraction. And, finally, c) not every brane interaction has a correspondent interaction in membrane systems. Eight operations are considered here: *pino, frith, endo, exo, froth, mate, mito, drip.* From these, *froth* and *frith* are complementary. The same can be said for *endo* and *exo. Mate* is complementary with *mito* and with *drip.*

Endo $[p]_m[q]_n \Rightarrow_{endo\ m} [[p]_m q]_n$

Exo $[[p]_m q]_n \Rightarrow_{exo} [p]_m[q]_n$

Pino $[\ p]_m \Rightarrow_{pino} [[]_n p]_m$

Frith $[\ p]_m \Rightarrow_{frith}$

Mate $[p]_m[q]_n \Rightarrow_{mate} [pq]_i$

Mito $[pq]_i \Rightarrow_{mito} [p]_m[q]_n$

Drip $[pq]_i \Rightarrow_{drip} [pq]_m[pq]_n$

Froth $[\ p]_m \Rightarrow_{froth\ p} [[p]_n]$

4 Context Solving Membrane Systems

We introduce in this section a new type of membrane systems, *Context Solving Membrane Systems*, specially defined for dealing with semantics and context-based linguistics. In them, membranes are contexts, this is, constraint stores. Objects in membranes are features, understood as first order predicates.

The rules are of three types, related with conjunction, disjunction or implication. It is important to keep in mind that membranes are not sets; they do not behave as sets. Disjunction is related to membrane generation, conjunction to addition of objects (constraints) in membranes, and implication to membrane nesting.

The main logical formula can be modelled by membrane systems. If we understand a membrane as a constraints store, *conjunction* adds elements (constraints) to the membrane or produces the fusion of two membranes (MATE), while *disjunctive* and *implicative* relations create new membranes. *Disjunction* crates a sibling membrane (MITO), while *implication* creates nesting (ENDO).

Definition 2 *A Context Solving Membrane System $C\Pi$ is defined as a construct:*

$$C\Pi = (V, \mu, C, \rho),$$

where:

- *V is an alphabet; its elements are objects, for example first order predicates;*
- *μ is a membrane structure of degree n;*
- *C is a set of logical connectors ($\wedge$, $\vee$ $\rightarrow$).*
- *ρ is a set of evolution rules for μ, which represent operations with membranes, given by the connectors on C: $\wedge$/Mate; $\vee$/Mito; $\rightarrow$/Endo.*

The three basic rules – Mate, Mito, Endo – given by basic operators, can be formalized as follows:

1. $[f(x)] \wedge [g(x)] \Rightarrow [f(x)\ g(x)]$: Mate
2. $[f(x) \vee g(x)] \Rightarrow [f(x)][g(x)]$: Mito
3. $[f(x)] \rightarrow [g(x)] \Rightarrow [[f(x)]g(x)]$: Endo

A graphical representation is shown in Figure 4.

Combining these rules, we obtain several types of reactions: a) simple reactions, reductions, b) composition.

Simple Reactions Simple reactions show how relationship among elements in a membrane, can start several processes in this membrane. We enumerate the easiest ones:

4. $\frac{[f(x)g(x)]}{f(x)\wedge g(x)} \Rightarrow [f(x)\ g(x)]$: Mate
5. $\frac{[f(x)g(x)]}{f(x)\vee g(x)} \Rightarrow [f(x)]\ [g(x)]$: Mito
6. $\frac{[f(x)g(x)]}{f(x)\rightarrow g(x)} \Rightarrow [[f(x)]\ g(x)]$ Endo

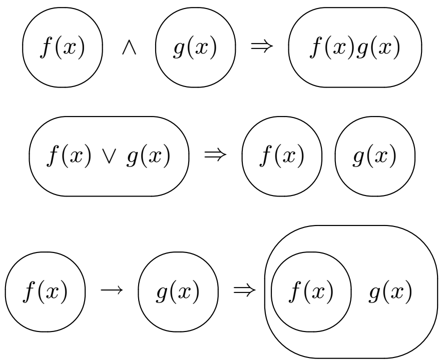


Fig. 4. Mito, Mate, Endo.

Reductions Reductions are reflexive fusions:

7. $[f(x)] \wedge [f(x)] \Rightarrow [f(x)]$
8. $[f(x)] \vee [f(x)] \Rightarrow [f(x)]$
9. $[f(x)] \rightarrow [f(x)] \Rightarrow [f(x)]$

Composition Composition refers to the solution of operations among two or more membrane systems with $deg > 1$, – among three or more simple membranes. In them, more than one simple operation is performed. We enumerate five simple cases:

10. $([f(x)] \rightarrow [g(x)]) \vee ([g(x)] \rightarrow [h(x)]) \Rightarrow [[f(x)][h(x)]]g(x)]$: ENDO + Reduction + MITO
11. $([f(x)] \rightarrow [g(x)]) \wedge ([g(x)] \rightarrow [h(x)]) \Rightarrow [[f(x)g(x)]h(x)]$: ENDO + Reduction + MATE
12. $([f(x)] \wedge [g(x)]) \wedge ([f(x)] \wedge [h(x)]) \Rightarrow [f(x)g(x)h(x)]$: MATE
13. $([f(x)] \rightarrow [g(x)]) \wedge ([g(x)] \rightarrow [h(x)]) \Rightarrow [[[f(x)]g(x)]h(x)]$: ENDO + Reduction + ENDO
14. $([f(x)] \rightarrow [g(x)]) \wedge [g(x)h(x)] \Rightarrow [f(x)[g(x)h(x)]]$: ENDO + Reduction

5 Lexical Semantics with Membrane Context Solvers

In this section, we introduce a quite simple formalization of lexical semantics considering membranes and the basic rules explained above. First, we define by means of simple rules, the three main types of semantic relationship: *polysemy, synonymy* and *hyponymy.* Later, we give in the same way two of the main types of semantic change according to [15]: *broadening* and *narrowing.*

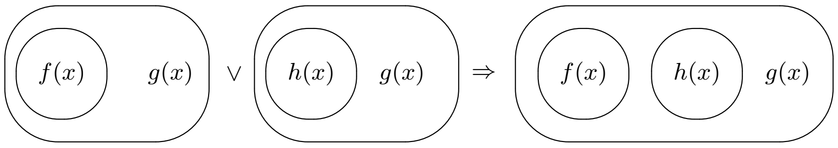


Fig. 5. Compositon 10: ENDO + Reduction + MITO

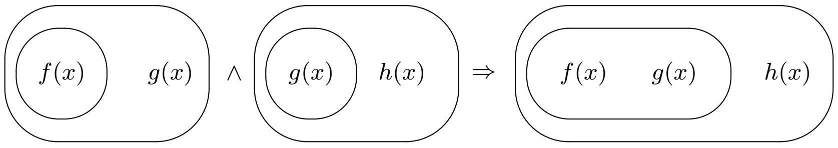


Fig. 6. Composition 11: ENDO + Reduction + MATE

5.1 Semantic Relationships

Three Brane Calculi operations are considered for defining the three main semantic relationships: DRIP, MATE, ENDO. Drip refers to the context division which causes the multiplicity in the meaning of a lexical item; the fusion process entailed by mate merges meanings and contexts which have, after the operation, the same linguistic referent; on the other hand, implication (or inclusion), establishes a membership feature in words. Here, the definition and membrane modelling for every relationship is provided.

- **Polysemy: Drip**.
  - **Definition:** Polysemy is a matter of one lexeme having several interrelated meanings.
  - **Membrane Modelling:** Polysemy is generated by Drip operations, when a membrane produces a copy of itself generating a new context for its productions.
  - **Example:** Let w_1 be a membrane with the element "light" insight. By the Drip rule $[light]_1 \rightarrow [light]_1\ [light]_2$, multiple contexts can be created for the same word, with different meanings.
- **Synonymy: Mate**.
  - **Definition:** Can be explained in a way inverse to polysemy. There is a fusion of contexts with some common element, producing very similar lexical items.
  - **Membrane Modelling:** Mate operations merge context causing synonymy.

  - **Example:** Having two membranes w_1 and w_2, with "thrifty" $\in w_1$, "mean" $\in w_2$, we obtain a new membrane including both terms by $[thrifty]_1 \wedge [mean]_2$.

– **Hyponymy: Endo**.

  - **Definition:** According to [21], hyponymy can be defined as follows: an expression A is a hyponym of an expression B iff the meaning of B is part of the meaning of A and A is a subordinate of B. In addition to the meaning of B, the meaning of A must contain further specifications, rendering the meaning of A, the hyponym, more specific than the meaning of B. If A is a hyponym of B, B is called a *hyperonym* of A.
  - **Membrane Modelling:** Hyponymy can be explained by the *nesting* relationship. Every membrane can be explained as a constraint store of traits of a lexical item, so that by including one membrane into another one we account for the relation between words that are in a hyponymy relationship.
  - **Example:** Let us consider that membranes w_1, w_2, and w_3 representing respectively the words "animal", "dog" and "terrier", by establishing a nesting relation between them ($w_3 \subset w_2 \subset w_1$) we are easily accounting for the hyponymy relation.
    Following the properties we have explained before, it can be said that being $w_3 \subset w_2 \subset w_1$ and being x a constraint of the world w, it can be said that, if $\forall x(x \in w_3 \rightarrow x \in w_2 \rightarrow x \in w_1)$. However $\exists x(x \in w_1 \wedge x \in w_2 \wedge x \in w_3)$. In the same way, it is obvious that $\exists x(x \in w_3 \wedge x \notin w_2)$.

5.2 Semantic Change

Hyponymy relationships can evolve and change, mainly in two directions. First of all, some of the membranes can accept some constraint during the evolution process, in a way that the membrane is equivalent to its parent one. The next step is the dissolution of the parent membrane. This is a process called *broadening* in lexical semantics. The opposite process is the one called *narrowing*. By it, a membrane loose a constraint and, in the process, it becomes equivalent to the nested membrane in it. The next step is the dissolution of the nested membrane.

– **Broadening, extension or generalization: Froth**.

  - **Definition:** The term *broadening* is used to refer to a change in meaning that results in a word acquiring additional meanings to those that it originally had, while still retaining those original meanings as part of the new meaning. So, it refers to the increase of the number of contexts in which a word can be used, paradoxically reducing the amount of information conveyed about each one.
  - **Membrane Modelling:** Membranes can model broadening by deleting constraints to the membrane, i.e. generating a new parent membrane to the existing one:

* There is a membrane system μ, being w_1 and w_2 membranes of that system, in a nested relationship such as $w_2 \subset w_1$. Both membranes are defined as a store of constraints $w_1 = f(x) \wedge g(x) \wedge h(x)$, $w_2 = f(x) \wedge g(x)$ The relationship between both domains is the following: $\forall x(f(x) \in w_1 \rightarrow f(x) \in w_2$, and $\exists x(f(x) \in w_2 \wedge f(x) \notin w_1)$.
* By applying the Froth operation, we obtain: $w_1 = f(x) \wedge g(x) \wedge h(x) \rightarrow w_i = f(x) \wedge g(x)$, with the broadening of the membrane. And by reduction, they MATE.

- **Example:** Figure 7 shows a graphical example of the extension of the word “dog”.

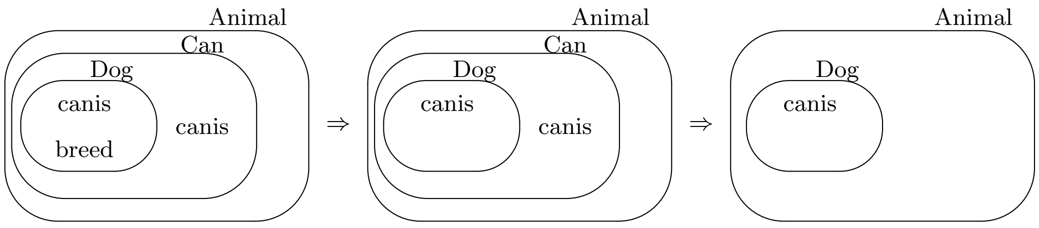


Fig. 7. Example of Broadening: Dog.

– **Narrowing, restriction or specialization: Pino**.

- **Definition:** Semantic narrowing is the exact opposite of the previous kind of change. We say that *narrowing* takes place when a word comes to refer to only part of the original meaning. The restriction of meaning paradoxically involves an increase in the information conveyed, since a restricted form is applicable to fewer situations, but tells more about each one.
- **Membrane Modelling:** Membranes can model narrowing by adding constraints to the membrane, i.e. generating a new nested membrane to the existing one:

  * There is a membrane system μ, being w_1 and w_2 membranes of that system, in a nested relationship such as $w_2 \subset w_1$. Both membranes are defined as a store of constraints $w_1 = f(x) \wedge g(x) \wedge h(x)$, $w_2 = f(x) \wedge g(x)$ The relationship between both domains is the following: $\forall x(f(x) \in w_1 \rightarrow f(x) \in w_2$, and $\exists x(f(x) \in w_2 \wedge f(x) \notin w_1)$.
  * By applying the Pino operation, we obtain: $w_2 = f(x) \wedge g(x) \rightarrow w_i = f(x) \wedge g(x) \wedge h(x)$, with the narrowing of the membrane.

- **Example:** Figure 8 shows a graphical example of the extension of the word “meat”.

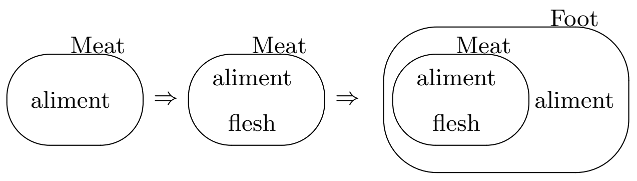


Fig. 8. Example of Restriction: Meat.

6 Final Remarks

The paper starts from a quite intuitive idea, the use of constraints for dealing with context in linguistics. This work has as a goal the implementation of contexts in natural language processing, and the ideas introduced here have to be further developed in order to have some application.

We suggest that, by means of the membrane-based approach to linguistics, some traditional difficulties like ambiguity and the formalization of pragmatics could be partially covered in a computational-friendly way. But membranes are a computational method introduced to generate formal languages, and their mathematical tools to formalize reasoning are quite limited. Constraint handling rules can help to this task. We think constraint solving provides some convenient formalization for inference that can be adapted to the membrane computing framework. Finally, perhaps using logical-based reasoning, the computational approach to context can become more robust as well as more consistent with current theories on formal linguistics.

The ideas introduced in this paper are still seminal, and a better study is needed in order to demonstrate such formalism can be useful for description and implementation of linguistic contexts. If it works, even if partially, and can be implemented, several applications for semantics and pragmatics can be designed.

References

1. Aitchison, J. (1991), *Language Change: Progress or Decay?*, Cambridge, Cambridge University Press.
2. Bel Enguix, G. (2003), Preliminaries about some Possible Applications of P Systems in Linguistics, in Păun, Gh.; Rozenberg, G.; Salomaa, A. & Zandron, C., (eds.), *Membrane Computing*, LNCS 2597, Springer, Berlin: 74–89.
3. Bel Enguix, G. & Gramatovici, R. (2004), Parsing with Active P Automata, in Martín-Vide, C., Mauri, G., Păun, Gh., Rozenberg, G., & Salomaa, A. (eds), *Membrane Computing*, Berlin, Springer.
4. Bel Enguix, G. & Jiménez López, M.D. (2007), Dynamic Meaning Membrane Systems: an Application to the Description of Semantic Change, *Fundamenta Informaticae*, 76(3): 219–237.

5. Bel Enguix, G. & Jiménez López, M.D. (2005), Linguistic Membrane Systems and Applications, in Ciobanu, G., Păun, Gh. & Pérez Jiménez, M.J. (eds.), *Applications of Membrane Computing*, Springer, Berlin: 347–388.
6. Bunt, Harry C. (1990), *DIT-Dynamic Interpretation in Text and Dialogue*, ITK Research Report, no. 15, Tilburg University, The Netherlands.
7. Busi, N. & Gorrieri, R. (2005), On the computational power of Brane Calculi. *Proc. Third Workshop on Computational Methods in Systems Biology (CMSB05)*, Edinburgh: 106-117.
8. L. Cardelli, A.D. Gordon (2000), Mobile Ambients. *Theoretical Computer Science*, 240 (1): 177-213.
9. Cardelli, L. (2005), Brane Calculi. Interactions of Biological Membranes, *Proc. Computational Methods in System Biology (CMSB 2004), Paris, France, May 2004. Revised Selected Papers*, LNCS 3082, Berlin, Springer: 257-280.
10. Cardelli, L. & Paun, G. (2006), A universality result for a (Mem)Brane Calculus based on mate/drip operations, *J. Found. Comput. Sci.* 17(1): 49–68.
11. Christiansen, H. (2001), *CHR as Grammar Formalism. A first report*, Annual Workshop of the ERCIM Working group on Constraints. Available at CoRR: http://arXiv.org/abs/cs.PL/0106059.
12. Christiansen, H. (2005), *CHR Grammars*, in *International Journal on Theory and Practice of Logic Programming*, vol. 4+5, special issue on Constraint Handling Rules: 467–501.
13. Cristiansen, H. & Dahl, V. (2005), Meaning in Context, in Dey, A., Kokinov, B., Leake, D., Turner, R. (eds.), Lecture Notes in Artificial Intelligence 3554: 97–111.
14. Croft, W. (2000), *Explaining Language Change. An Evolutionary Approach*, Longman, Singapore.
15. Crowley, T. (1992), *An Introduction to Historical Linguistics*, Oxford University Press.
16. Cruse, D.A. (1986), *Lexical Semantics*, Cambridge University Press.
17. Debusmann, R., Duchier, D. & Kuhlmann, M. (2004), Multi-dimensional Graph Con.guration for Natural Language Processing, in Christiansen, H., Skadhauge, P.R. & Villadsen, J. (2004), *Constraint Solving and Language Processing. International Workshop, CSLP 2004. Proceedings*, Roskilde, Roskilde University: 59–73.
18. Frühwirth, T. (1994), Theory and Practice of Constraint Handling Rules, *Journal of Logic Programming* 37, 1-3: 95–138.
19. Groenendijk, J. & Stokhof, M. (1991), Dynamic Predicate Logic, *Linguistics and Philosophy*, 14: 39–100.
20. Hobbs, J.R., Stickel, M., Martin, P. & Edwards, D. (1993), Interpretation as abduction, *Artificial Intelligence* 63, 1-2: 69–142.
21. Löbner, S. (2002), *Understanding Semantics*, Arnold, London.
22. Lyons, J. (1995), *Linguistic Semantics. An Introduction*, Cambridge University Press.
23. McMahon, A.M.S. (1996), *Understanding Language Change*, Cambridge University Press.
24. Păun, Gh. (2000), Computing with Membranes, *Journal of Computer and System Sciences*, 61(1): 108-143.
25. Păun, Gh. (2002), *Membrane Computing. An Introduction*, Springer, Berlin.
26. A. Regev, E. M. Panina, W. Silverman, L. Cardelli, E. Shapiro (2004), BioAmbients: An Abstraction for Biological Compartments. Theoretical Computer Science 325 (1): 141-167.
27. Sihler, A. L. (2000), *Language History. An Introduction*, John Benjamins, Amsterdam.

“*Model Theoretic Syntax is not Generative Enumerative Syntax with constraints*”: under what condition?

Philippe Blache

Laboratoire Parole et Langage
Aix-Marseille Universités & CNRS
pb@lpl.univ-aix.fr

Abstract. The generative conception of grammars relies on the derivation process which, to its turn, depends on a hierarchical representation of syntax. We show in this paper how a fully constraint-based approach, avoiding such restriction, can constitute an alternative to generativity and form the basis of a framework for a model-theoretic conception of grammar.

1 Introduction

There are two different approaches in logic: one is *syntactic* and only uses the form of the formulae in order to demonstrate a theorem, the second is *semantic* and relies on formulae interpretation. The same distinction also holds in linguistics. A first conception relies on the study of the input formedness. In this case, the problem consists in finding a structure adequate to the input. Grammatical information is then represented by means of a set a rules. An alternative conception, instead of relying on the input form, focuses on its characteristics. Following [Pullum & Scholz 01], we will call these approaches respectively generative enumerative syntax (GES) and model-theoretic syntax (MTS) The first approach is that of generative theories, it has been extensively experimented. The latter still remains marginal. One of the reasons is that generativity has been for years almost the unique view for formal syntax and it is difficult to move from this conception to a different one. In particular, one of the problem comes from the fact that all approaches, even those in the second perspective, still rely on a hierarchical (tree-like) representation of syntactic information.

This paper describes this problem and proposes an MTS framework moving from a classical tree domain towards a graph domain for the representation of syntactic information. We show how constraints can be an answer to this problem: first, they can represent all kinds of syntactic information and second, they constitute a system, all constraints being at the same level (none being to be evaluated before others).

The paper starts with a description of the main characteristics of these different conceptions of syntax. In the second section, we focus on the specific problems coming from the hierarchical conception of syntax, showing how it can constitute a severe limitation for linguistic description. The third section proposes an overview of an MTS framework, called *Property Grammars*, following this requirements. We precise formally the status of the constraints we use, and how in this approach a syntactic description comes to a graph. We explain in particular how it is possible to take advantage of such a representation in order to shift from the classical tree domain to a graph one, and in what sense this shift constitutes a solution to the MTS problem.

2 Proof vs. Model Theory in Syntax

The generative conception of syntax relies on a particular relation between grammar and language: a specific mechanism, derivation, makes it possible to generate a language from a grammar. This basic mechanism can be completed with other devices (transformations, moves, feature propagation, etc.) but in all cases constitute the core of all generative approaches. In such case, grammaticality consists in finding a set of derivations between the start symbol of the grammar and the sentence to be parsed. As a side effect, a derivation step coming to a local tree, it is possible to build a syntactic structure, represented by a tree. It is then possible to reduce in a certain sense the question of grammaticality to the possibility of building a tree. This reminder seems to be trivial, but it is important to mesure its consequences. The first is that grammaticality is reduced, as it has been noticed in [Chomsky75], to a boolean value: true when a tree can be built, false otherwise. This is a very restrictive view of grammaticality, as it also has been noticed in [Chomsky75] (without proposing a solution), which forbids a finer conception, capable of representing in particular a grammaticality scale (also called *gradience*, see [Keller00] or [Aarts07]).

This generative conception of syntax is characterized as being enumerative (see [Pullum & Scholz 01] in the sense that derivation can be seen as an enumeration process, generating all possible structures and selecting them by means of extra constraints (as it is typically the case in the Optimality Theory, see [Prince93]).

Model Theoretic Syntax proposes an alternative view ([Blackburn & al. 93], [Cornell & Rogers 00], [Pullum & Scholz 01]). In this conception, a grammar is a set of assessments, the problem consists in finding a model into a domain.

>From a logical perspective, generative approaches rely on a *syntactic* conception in the sense that parsing consists in applying rules depending on the form of the structures generated at each step. For example, a nonterminal is replaced with a set of constituents. On the opposite, model-theoretic approaches rely on a *semantic* view in which parsing is based on the interpretation (the truth values) of the statements of the grammar. A grammar in MTS is a set of statements or, formally speaking, formulae. Each formula describes a linguistic property; its interpretation consists in finding whether this statement is true or false for a given set of values (the universe of the theory in logical terms). When a set of values satisfies all assessments of the grammar (in other words when the

interpretation of all the formulae for this set of values is true), then this set is said to be a model.

As far as syntax is concerned, formulae indicate relations between categories or, more precisely, between descriptions of categories. These descriptions correspond to the specification of a variable associated with several properties: they can be seen as formulae. For example, given $\mathcal{K}$ a set of categories, a description of a nominative noun comes to the formula:

(1) $\exists x[cat(x, N) \wedge gend(x, masc)]$

A category can be described by a more or less precise description, according to the number of conjuncts. A grammatical statement is a more complex formula, adding to the categories descriptions other relations. For example, a statement indicating that a determiner is unique within a noun phrase comes to the formula:

(2) $[cat(x, Det) \wedge cat(y, Det) \rightarrow x \approx y]$

Concretely, when parsing a given input, a set of categories is instantiated, making it possible to interpret all the atomic predicates corresponding to the features (category, gender, number, etc.), making it possible to interpret to their turn the complex predicates formed by the grammatical statements. In this perspective, we say that an instantiated category is a value and finding a model consists in finding a set of values satisfying all the grammatical statements. For example, the set of words "*the book*" makes it possible to instantiate two categories with labels *Det* and *N* (these labels representing the conjunction of features). Intuitively, we can say that the set of values *{Det, N}* is a model for the *NP*.

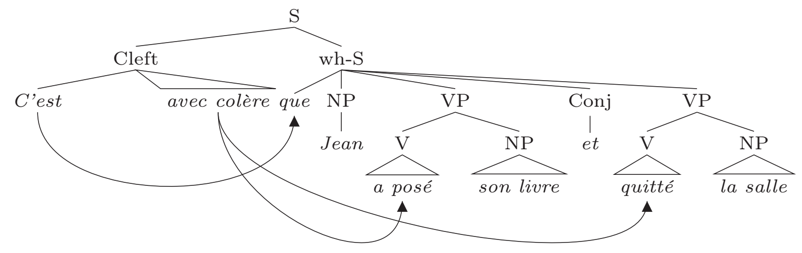
Finding a model is then completely different than deriving a structure. As underlined by [Pullum & Scholz 01], instead of enumerating a set of expressions, an MTS grammar simply states conditions on these expressions.

3 Generativity and hierarchical structures

Geoffrey Pullum, during a lecture at ESSLLI in 2003, explained that "*Model Theoretic Syntax is not Generative Enumerative Syntax with constraints*". In other words, constraints are not to be considered only as a control device (in the DCG sense for example) but have to be part of the theory. Some theories (in particular HPSG) try to integrate this aspect. But it remains an issue both for theoretical and technical reasons. The problem comes in particular from the fact usually, dominance relation plays a specific role in the representation of syntactic information: dominance structures have first to be built before verifying other kinds of constraints. This is a problem when no such hierarchical relations can be identified. Moreover, we know since GPSG that dominance constitutes only a part of syntactic information to be represented in phrase-structure approaches, not necessarily to be considered as more important than others.

Syntactic information is usually defined, especially in generative approaches, over tree domains. This is due to the central role played by the notion of dominance, and more precisely by the relation existing between the head and its direct ancestor. In theories like HPSG (see [Sag al. 03]), even though no rules are used (they are replaced with abstract schemata), this hierarchical organization remains at the core of the system. As a consequence, constraints in HPSG can be evaluated provided that a tree can be built: features can be propagated and categories can be instantiated only when the hierarchical skeleton is known. This means that one type of information, dominance, plays a specific role in the syntactic description.

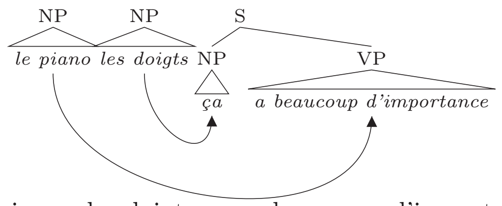
However, in many cases, a representation in terms of tree is not adapted or even not possible. The following example illustrate this situation. It present the case of a cleft element adjunct of two coordinated verbs.



C'est avec colère que Jean a posé son livre et quitté la salle
It is with anger that Jean put his book and left the room

Arrows in this figure shows in what sense the tree fails in representing the distribution of the cleft element onto the conjuncts. Moreover, there also exist other kinds of relations, for example the obligatory cooccurrence in French between "*c'est*" and "*que*".

The second example, presented in the following structure, illustrates the fact that in many cases, it is not possible to specify clearly what kind of syntactic relation exists between different parts of the structure:



Le piano les doigts ça a beaucoup d'importance
The piano the fingers it has a lot of immportance

This example illustrate a multiple detachment construction. In this case, detached element are not directly connected by classical syntactic relations to the rest of the structure: the two relations indicated by arrows are dependencies at the discourse level (plus an anaphoric relation).

Many other examples can be given, illustrating this problem: it is not always possible to give a connected structure on the basis of syntactic relations. Moreover, when adding other kinds of relations, the structure is not anymore a tree. This conception has direct consequences on the notion of grammaticalness. First, building a tree being a pre-requisite, nothing can be said about the input when this operation fails. This is the main problem with generative approaches that can only indicate whether or not an input is grammatical, but do not explain the existence of levels of grammaticality (the gradience phenomenon, see [Keller00], [Pullum & Scholz 01]).

A second consequence concerns the nature of linguistic information, that is typically spread over different domains (prosody, syntax, pragmatics, and related domains such as gestures, etc.). An input, in order to be interpreted, does not necessarily need to receive a complete syntactic structure. The interpretation rather consists in bringing together pieces of information coming from these different domains. This means that interpreting an input requires to take into account all the different domains and their interaction, rather than building a structure for each of them and then calculating their interface. In this perspective, no specific relation plays a more important role than others. This is also true within domains: as for syntax, the different properties presented in the previous section has to be evaluated independently from the others.

4 A constraint-based MTS framework: Property Grammars

A seminal idea in GPSG (see [Gazdar & al. 85]) was to dissociate the representation of different types of syntactic information: dominance and linear precedence (forming the ID/LP formalism), but also some other kinds of information stipulated in terms of cooccurrence restriction. This proposal is not only interesting in terms of syntactic knowledge representation (making it possible to factorize rules, for example), but also theoretically. Remind that one of the main differences between GES and MTS frameworks lies in the relation between grammar and language: MTS approaches try to characterize an input starting from available information, with no need to "overanalyze", to re-build (or infer) information that is not accessible from the input. For example, GES techniques have to build a connex and ordered structure, representing the generation of the input. On the opposite, nothing in MTS imposes to build a structure covering the input, which makes it possible for example to deal with partial or heterogeneous information. *Property Grammars* (see [Blache05]) systematizes the GPSG proposal in specifying these different types. More precisely, they propose to represent separately the following properties:

- *Constituency*: set of all the possible elements of a construction
- *Uniqueness*: constituents that cannot be repeated within a construction
- *Linearity*: linear order

– *Obligation*: set of obligatory constituents, one of them (exclusively to the others) being realized.

– *Requirement*: obligatory cooccurrence between constituents within a construction

– *Exclusion*: impossible cooccurrence between constituents within a construction

This list is not closed and other types of information can be added. For example, dependency (syntactico-semantic relation between a governor and a complement), or adjacency (juxtaposition of two elements). We focus in this paper on the 6 basic relations indicated above. These relations makes it possible to represent most of the syntactic information. We call these relations "properties", they can also be considered as constraints on the structure.

We adopt in the remaining of this paper the following notations: x, y (lower case) represent individual variables; X, Y (upper case) are set variables. We note $C(x)$ the set of individual variables in the domain assigned to the category C (see [Backofen & al. 95] for more precise definitions). We use the set of binary predicates for immediate domination ($\lhd$), linear precedence ($\prec$) and equality ($\approx$).

Let us now define more precisely the different properties. The first one (constituency) implements the classical immediate dominance relation. The others can be defined as follow:

– $Const(A, B) : (\forall x, y)[(A(x) \wedge B(y) \rightarrow x \lhd y]$
This is the classical definition of constituency, represented by the dominance relation: a category B is constituent of A stipulates that there is a dominance relation between the corresponding nodes.

– $Uniq(A) : (\forall x, y)[A(x) \wedge A(y) \rightarrow x \approx y]$
If one node of category A is realized, there cannot exists other nodes with the same category A. Uniqueness stipulates constituents that cannot be repeated in a given construction.

– $Prec(A, B) : (\forall x, y)[(A(x) \wedge B(y) \rightarrow y \not\prec x)]$
This is the linear precedence relation as proposed in GPSG. If the nodes x and y are realized, then y cannot precedes x

– $Oblig(A) : (\exists x)(\forall y)[A(x) \wedge A(y) \rightarrow x \approx y]$
There exists a node x of category A and there is no other node y of the same category. An obligatory category is realized exactly once.

– $Req(A, B) : (\forall x, y)[A(x) \rightarrow B(y)]$
If a node x of category A is realized, a node y of category B has too. This relation implements cooccurrence, in the same way as GPSG does.

– $Excl(A, B) : (\forall x)(\nexists y)[A(x) \wedge B(y)]$
When x exists, there cannot exist a sibling y. This is the exclusion relation between two constituents.

What is interesting in this representation of syntactic information is that all relations are represented independently form each others. They all are assessment in the MTS sense, and they can be evaluated separately (which fits well with the non-holistic view of grammatical information proposed by Pullum). In other words there is no need to assign the dominance relation a specific role: this is one information among others, what is meaningful is the interaction between these relations. More precisely, a set of categories can lead to a well-formed structure when all these assessments are satisfied, altogether. We do not need first to build a structure relying on dominance and then to verify other kind of information represented by the rest of the relations. In other words, in this approach, "*MTS is not GES with constraints*" ([Pullum & Scholz 03]).

Concretely, when taking into consideration a set of categories (an assignment), building the syntactic structure comes to evaluate the constraint system for this specific assignment. The result of the evaluation indicates whether or not the assignment corresponds to a well-formed list of constituents. For example, given two nodes x and y, if they only verify a precedence relation, nothing else can be said. But when several other properties such that requirement, uniqueness, constituency are also satisfied, the assignment $\{x, y\}$ becomes a model for an upper-level category. For example, if we have x and y such that $Det(x)$ and $N(y)$, this assignment verifies precedence, uniqueness, constituency and requirement properties. This set of properties makes it possible to characterize a *NP*. At the opposite, if we take x and y such that $Det(x)$ and $Adv(y)$, no constraint involving both constituents belong to the system: they do not constitue a model, and no new category can be infer.

In terms of representation, at the difference with classical approaches, syntactic information is not represented by means of a tree (see [Huddleston & Pullum 02]), but with a *directed labelled graph.* Nodes are categories and edges represent constraints over the categories: dominance, precedence, requirement, etc. A non-lexical category is described by a set of constraints, that are relations between its constituents. It is possible to take under consideration only one type of property (in other words one type of relation): this comes to extract a subgraph from the total one. For example, one can consider only constituency properties. In this case, the corresponding subgraph of dominance relations is (generally) a tree. But what is needed to describe precisely an input is the entire set of relations.

In the following, we represent the properties with the set of relations noted $\Rightarrow$ (requirement), $\otimes$ (exclusion), $\circ$ (uniqueness), $\lhd$ (constituency), $\uparrow$ (obligation), $\prec$ (precedence). A *Property Grammar* graph (noted *PG-graph*) is a tuple of the form:

$$G = \langle W, \Rightarrow, \otimes, \circ, \lhd, \uparrow, \prec, \theta \rangle$$

in which W is the set of nodes, θ the set of terminal nodes. A model is a pair $\langle G, V \rangle$ where V is a function from W to *Pow(W).* We describe in the next section the use of such graphs.

5 Grammaticalness and constraints

Classically, syntactic information is usually represented in terms of decorated ordered trees (see [Blackburn & al. 93], [Blackburn & Meyer-Viol 94]). In this approach, tree admissibility relies on a distinction between dominance relation (that gives the structure) and other constraints on the tree such as precedence, cooccurrence restriction, etc. In our view, all relations has to be at the same level. In other words, dominance does not play a specific role: cooccurrence restriction for example can be expressed and evaluated independently from dominance. This means that each property represents a relation between nodes, dominance being one of them. When taking into consideration the entire set of relations, the structure is then a graph, not a tree. More precisely, each property specifies a set of relations between nodes: precedence relations, cooccurrence relations, dominance relations, etc. It can be the case that the dominance subset of relations (a subgraph of the graph of relations), is a tree, but this can be considered as a side effect. No constraint for example stipulates a connexity restriction on the dominance subgraph.

In *PG*, a grammar is then conceived as a constraint system, corresponding to a set of properties as defined above. Parsing an input consists in finding a model satisfying all the properties (or more precisely, the properties involving the categories of an assignment). In this case, the input is said to be grammatical, its description being the set of such properties. However, it is also possible to find models that satisfy partially the system. This means that some constraints can be violated. If so, the input is not grammatical, but the set of satisfied and violated properties still constitute a good description. We call such set a *characterization.* This notion replaces that of grammaticalness (which is a particular case of characterization in which no property is violated).

The following example (figure 1b) illustrates the case of an assignment *A={NP, Det, Adj, N}*. All properties are satisfied, each relation forms an labeled edge, the set of relations being a graph. A phrase is characterized when it is connected to a graph of properties.

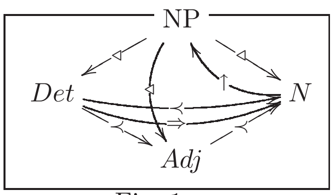


Fig. 1a

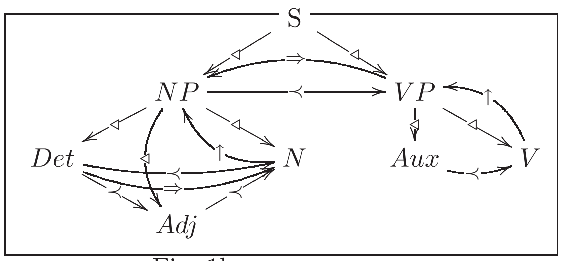
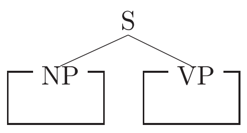
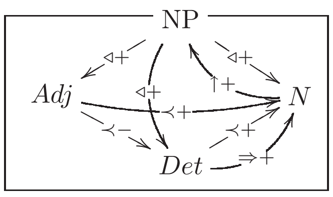


Fig. 1b

Figure 1b shows a more complete graph, corresponding to an entire sentence. Again, no relation in this graph plays a specific role. The information comes from the fact that this set of categories are linked by several relations. The set of relations forms a description: it tells us that linearity, requirement, obligation, constituency properties are satisfied, they characterize an *S*. Theoretically, each node can be connected to any other node. Nothing forbids for example to represent a relation of some semantic type between the adjective and the verb nodes. By another way, when taking from this graph constituency relations only, we obtain a dominance tree:



Finally, insofar as a property can be satisfied or violated in a characterization, we have to label relations with their type and their interpretation (true or false, represented $+$ or -). The following example presents a graph for the assignment *A={NP, Adj, Det, N}*, in which the determiner has been realized after the adjective.



In this graph, all constraints but the precedence between *Det* and *Adj* have been satisfied, the corresponding relations being labeled with $+$.

6 Explaining levels of grammaticality

As a side effect, representing information in this way also constitues a possibility to rank the inputs according to a grammaticalness evaluation. We present in this section how to use characterizations in order to quantify such information. The idea (see [Blache & al. 06]) consists in analyzing the form of the graph and its density, taking into account the interpretation of the relations. More precisely the method consists in calculating an index from the cardinality of P^+ and P^-, (respectively the set of satisfied and violated properties). Let's call N^+ and N^- the cardinality of these sets. The first indication that can be obtained is the ratio of satisfied properties with respect to the total number of evaluated properties E. This index is called the *Satisfaction ratio*, calculated as $SR = \frac{N^+}{E}$.

Going further, it is also possible to give an account of the coverage of the assignment by means of the ratio of evaluated properties with respect to the total number of properties T describing the category in the grammar. This coefficient is called *Completeness coefficient*: $CC = \frac{E}{T}$.

A *Precision Index* can to its turn be proposed, integrating these two previous information: $PI = \frac{SR+CC}{2}$.

Finally, a general index can be proposed, taking into consideration the different indexes of all the constituents. For example, a phrase containing only well-formed constituents has to be assigned a higher value than one containing ill-formed ones. This is done by means of the *Grammaticalness Index*, d being the number of embedded constructions C_i: if $d = 0$ then $GI = PI$, else $GI = PI \times \frac{\sum_{i=1}^{d} GI(C_i)}{d}$.

In reality, this different figures need to be balanced with other kind of information. For example, we can take into consideration the relative importance of constraint types in weighting them. Also, the influence of SR and CC over the global index can be modified by means of coefficients.

This possibility of giving a quantified estimation of grammaticalness directly comes from the possibility of representing syntactic information in a fully contraint-based manner, that has been made possible thanks to the MTS view of grammar.

7 Conclusion

The representation of syntactic information by means of constraints, as described in this paper, has several advantages. First, it makes it possible to implement the entire MTS programme in which derivation does not play anymore a role. The shift from generative to model-based conception of syntax becomes then concrete: constraint satisfaction completely replaces derivation. This evolution becomes possible provided that we abandon a strict hierarchical representation of syntax in which dominance plays a central role.

As a consequence, such fully constraint-based approach offers the possibility to replace ordered trees domain with that of constraint graphs. This is not only a matter of representation, but has deep consequences on theory itself: different types of information is represented by different relations, all of them being at the same level.

The *Property Grammar* framework described in this paper represents the possibility of an actual MTS implementation in which constraints are not only a control layer over the structure, but represent the structure itself: MTS is not GES plus constraints, provided that dominance is not represented separately from other information.

References

[Aarts07] Aarts. B (2007) *Syntactic Gradience. The nature of Grammatical Indeterminacy*, Oxford University Press.

[Blackburn & al. 93] Blackburn P., C. Gardent & W. Meyer-Viol (1993) “Talking About Trees” in proceedings of *EACL*

[Blackburn & Meyer-Viol 94] Blackburn P. & W. Meyer-Viol (1994) “Linguistics, Logic and Finite Trees” in *Bulletin of the IGPL*, 2.

[Backofen & al. 95] Backofen R., J. Rogers, and K. Vijay-Shanker (1995) “A First-Order Axiomatization of the Theory of Finite Trees” in *Journal of Logic, Language, and Information*, 4:1

[Blache05] Blache P. (2005) “Property Grammars: A Fully Constraint-Based Theory", in H. Christiansen & al. (eds), *Constraint Solving and NLP*, Lecture Notes in Computer Science, Springer.

[Blache & al. 06] Blache P., B. Hemforth & S. Rauzy (2006), “Acceptability Prediction by Means of Grammaticality Quantification”, in proceedings of *COLING-ACL 06*

[Chomsky75] Chomsky N.. (1975) *The Logical Structure of Linguistic Theory*, Plenum Press

[Cornell & Rogers 00] Cornell T. & J. Rogers (2000) “Model Theoretic Syntax”, in L. Lai-Shen Cheng & R. Sybesma (eds), *The Glot International State of the Article Book I*, Holland Academic Graphics

[Gazdar & al. 85] Gazdar G., E. Klein, G. Pullum & I. Sag (1985) *Generalized Phrase Structure Grammars*, Blackwell

[Huddleston & Pullum 02] Huddleston R. G. Pullum (2002) *The Cambridge Grammar of the English Language*, Cambridge University Press.

[Keller00] Keller F. (2000) *Gradience in Grammar. Experimental and Computational Aspects of Degrees of Grammaticality*, Phd Thesis, University of Edinburgh.

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

[Pullum & Scholz 01] Pullum G. & B. Scholz (2001) “On the distinction between model-theoretic and generative-enumerative syntactic frameworks”, in processdings of the conference on *Logical Aspects of Computational Linguistics*, Springer

[Pullum & Scholz 03] Pullum G. & B. Scholz (2003) “Foundations of Model-Theoretic Syntax”, Introductory course of the *ESSLLI-03*

[Rogers97] Rogers J. (1997) “Grammarless Phrase Structure Grammar”, in *Linguistics and Philosophy*, 20

[Sag al. 03] Sag I., T. Wasow & E. Bender (2003) *Syntactic Theory. A Formal Introduction*, CSLI.

Reasoning about Use Cases using Logic Grammars and Constraints

Henning Christiansen, Christian Theil Have, Knut Tveitane

Roskilde University and IT University of Copenhagen

Abstract. We consider automated transition from Use Cases in a restricted natural language syntax into UML models, by trying to capture the semantics of the natural language and map it into building blocks of the object oriented programming paradigm. Syntax and semantic analysis is done in a framework of Definite Clause Grammars extended with Constraint Handling Rules, which generalizes previous approaches with a direct way to express domain knowledge utilized in the interpretation process as well as stating explicit rules for pronoun resolution.

1 Introduction

Definite Clause grammars [1] complemented with Constraint Handling Rules [2] (CHR) provides a potential for abductive discourse analysis as suggested by [3, 4]: knowledge about the semantic and syntactic context, represented as constraints, is added gradually to the constraint store, and a rule-based constraint solver can provide an incremental processing of this knowledge, including resolving various issues at different levels, such as anaphora analysis and other potential ambiguities. For a better processing of anaphora we extend "assumptions" of earlier proposals with time stamps.

We apply these principles in an automatic and interactive system which translates a restricted, but naturally appearing, use case language into software models. The software models take the form of class and activity diagrams in the UML notation [5].

The system maintains an up-to-date diagrammatical presentation of the current use case text in a window on the user's screen, cf. Fig. 4. This is intended to encourage an iterative mode of working, so as soon as the user adds a new or modifies an existing sentence, the consequences in terms of the object model is displayed immediately. This can aid the user in the process of understanding the current domain, including identifying possible misconceptions at an early stage. In case of unknown or ambiguous sentence constructions, the system may issue a warning and advise the user in rephrasing the sentence. Possible applications include requirement engineering, brainstorming, prototyping and as a tool for teaching UML.

The paper is structured as follows. Section 2 reviews related work; section 3 provides an overview of the system and section 4 goes into detail with the specific NLP methods developed. A conclusion and proposal for future directions is provided by section 5.

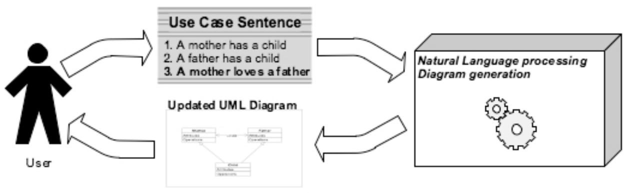


Fig. 1. The system is used iteratively and interactively - each new sentence causes the UML diagram to be updated.

2 Related Work

2.1 Similar Systems

Several authors have attempted to automate translation from specifications in natural language to code or diagrams. We have not seen this related directly to Use Cases and UML, which is our approach.

The examined systems use either essentially formal language with a "natural" appearance or opportunistic parsing. Formal languages are essentially designed to to allow unambiguous interpretation whereas opportunistic parsing takes a statistical approach to ambiguity, interpreting it as the most likely hypothesis wrt. a corpus of statistical material. Our approach falls into the first category and is based on a subset english which can be interpreted unambiguously.

Attempto Controlled English [6] is a system that translates specification texts in a formal language of declarative sentences into first order logic discourse representation structures and optionally into Prolog. ACE supports anaphora resolution and simple variable references. It accomplishes the translation using "a top-down parser using a unification-based phase structure grammar" [6].

Sowa has defined a similar, but simpler, specification language called "Common Logic Controlled English" [7] which translates directly into first order logic (and vice versa). CLCE can be parsed using a context-free grammar and a couple of lookup tables. Anaphora is not allowed. The Metafor system [8] use opportunistic techniques and the semantically enriched lexicon *ConceptNet* [9] to derive and discover relations between classes and translate English sentences into code in Lisp or Python. The supported input language supports in an impressing way a variety of narrative stances, past and present tense and anaphorical and indirect references, but it also allows for much ambiguity and it is not clear how this is handled. The authors [10] have coined the term *"programmatic semantics"* to describe the transliteration process: "Programmatic semantics is a mapping between natural linguistic structures and basic programming language structures" [10]. We have adopted this terminology.

Examples of other approaches using NLP for requirement analysis are described by [11–15].

2.2 Use Cases and Unified Modelling Language

Using problem description sentence analysis to derive program designs is by no means a new idea. Abbott [16] introduced a (manual) methodology for object-oriented program design that derives candidate classes, objects and operators from the syntactical elements of English sentences.

Booch was inspired by Abbotts method, which became an integral part of Booch's "object-oriented design" process, where an informal problem description is formalized through definition of objects and their attributes and operations.

At the OOPSLA conference in 1987, Jacobsen introduced the concept of Use Cases [17], which resembled Booch's problem descriptions. Use cases model the actors of a system and the flow of events between them. They describe *what* a system does without specifying how [18], which is useful for "gaining an understanding of the problem" and "identifying candidate objects" [19]. Rumbaugh et. all [20] decribed the OMT notation including the "Class Association Diagram", the precursor of the UML class diagram. All these things came together when Booch, Jacobson and Rumbaugh later defined the first draft for a "Unified Modeling Language" [21].

The UML User Guide [18] does not provide any in-depth guidance on how to write use-cases. However semi-formal approaches aiming simplify the language in which the use cases are written have been suggested by [22–25].

2.3 NLP using Logic Grammars

As shown in previous work [26–28, 4], CHR provides a straightforward implementation of abduction, and here we follow the principle of discourse analysis as abduction, introduced by [29] and now widely accepted: the meaning of a discourse is taken to be the set of "hidden" facts over which the discourse is faithfully created; in our application, these facts represent the inherent class relationships and other properties of the involved actors and objects.

The NLP methods developed through the present application follows the tradition of logic programming based grammars, but extends previous work in different ways. Assumption Grammars [30] (AG) provide mechanisms that may cope with pronouns references. Inspired by the work of [28, 4] that realizes AGs in CHR, we have extended such "assumptions" with time stamps to make it possible to specify detailed scope and preference rules in CHR, which otherwise is a shortcoming of AGs. Creation of data and knowledge bases from text using logic grammars have been pursued by a variety of authors, e.g., [31, 6]. The model of "Meaning in Context" formalized by [3], which is based on CHR shows how domain knowledge utilized in the interpretation process can be expressed directly in CHR; the domain for our application is, thus, general use case modeling.

3 System Overview

3.1 Architecture

The system is implemented in Prolog using a combination of Definite Clause Grammars and Constraint Handling Rules. Input sentences are entered and processed individually by the user. Each sentence is processed using our combined DCG and CHR grammar, explained in section 4, and a representation of its meaning is added to a constraint store. The contents of the constraint store is extracted as code in the GraphViz language using a second DCG. The code is rendered as a UML class diagram using a GraphViz engine, and displayed to the user. Figure 2 illustrates this architecture.

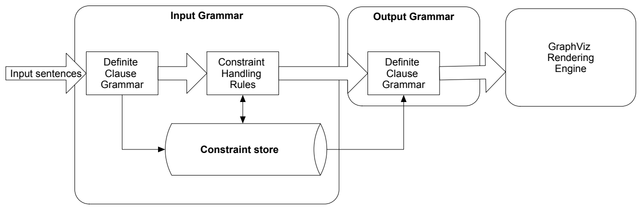


Fig. 2. Architecture Overview. The thick lines illustrate program flow and the thin lines illustrate the use of the constraint store.

3.2 Supported Language Constructs

The subset of natural English supported by our system needs to be sufficiently expressive as to cover the important entities and relations in an object oriented system description. While conforming with a basic formal syntax, it should also allow for a sufficient flexibility to maintain a certain degree of the flow and style of natural language. We have tried to find a balance between these; a natural and expressive, but formal language.

In the following, we describe the supported sentence types; an example is given for each and at the end, in Fig. 3, the diagram produced by our system is shown. The grammar includes compund sentences, conjunctions, and repetions which are more or less standard and not discussed in detail, but illustrated in the examples.

Method Sentences The most basic sentence consist of a noun phrase followed by a simple verb phrase. The verb phrase can contain an intransitive or transitive verb. We will first consider verbs that imply an action to be performed by or on the subject of the sentence.

Programmatic Semantics The subject of the sentence is a noun. This noun maps to a *class definition* in the object oriented programming paradigm. The verb maps to a *method* of the class represented by the subject. If the verb is transitive, the object is another noun (that defines another class) and this class serves as *argument* to the method.

Example: *"The professor teaches. A student reads, writes projects and takes exams."*

Property Sentences Property sentences that imply an ownership or containment relation are syntactically similar to the transitive sentences above, but uses specific verbs such as "have" indicating a different programmatic semantics. The object of a property sentence may be plural, and can be quantified. The quantification may be exact; a numeral or a number, but can also be a linguistic quantifier such as "some", "most" or "every".

Programmatic Semantics Property sentences associate properties expressed by their object with the class(es) indicated by their subject. Properties in plural form map to multi-valued properties. There are different approaches for representing these in object oriented programming languages. We have tried not to limit the flexibility, by maintaining as precise information as possible about the cardinality of the property. When exact number is given, this is preserved and a quantifier such as "some" is mapped into an undetermined cardinality denoted as "`n`". When alternatives are given for the same property, the different cardinalities are aggregated into an interval; the details are spelled out in section 4.2 below.

Example: *"A professor has an office. The university has five study lines."*

Inheritance and Instantiation Sentences Sentences formed with the verb "to be" relate single objects or classes to other classes.

Programmatic Semantics In *"A student is a (kind of) person"*, "person" is a *superclass* of "student", and "student" a subclass of "person"; the markers *kind of/type of* are reserved for sub/superclass relationships. Multiple inheritance, where the subject is "a kind of" more than one class is also possible.

Another construction is a sentence like "John is a student". The subject noun phrase here is a proper noun, and the significance of this sentence is that there is a concrete, named entity (John) of the class "student". In object oriented terminology, John is an *object* of class student. After an individual has been introduced with a designated class membership, it can be used as a *prototype* member of that class C in sentence forms above. Consider a sentence such as "John reads". It looks straightforward at first glance, but its programmatic semantics is bit more complex. "John" maps to an object, "reads" to a method, but objects don't define methods (or properties). The sentence must be understood to mean that the method belongs to the *class* the object is an instance of. However, if John has been explicitly declared member of two different classes (i.e., not implicitly via superclass relationships), “John reads” is judged ambiguous and rejected. Prototypes have also natural usages as arguments in basic sentences that introduces method In a suitable context, “Mary interviews Peter” carries the same programmatic semantics as “Teachers interview students.

Example: *‘Students are a kind of persons and professors are a kind of persons.”*

Pronouns For use case text, we need to require that pronouns can be resolved in a unique way which is intelligible to the user; at the same time we should also, for the acceptance of the tool, allow a variety of natural patterns. As is well-known, pronoun (and anaphora) resolution is one of the most difficult tasks in computational linguistics, cf. [32], and we have decided to use a simple heuristic, and reject any sentence for which it does not apply. Resolving, say, “he” considers the most recent occurrence of a male object which is found in a previous sentence S; however, if S contains two candidates, the pronoun application is claimed ambiguous. This excludes usages such as “Peter and Paul ... . He ...” and “Women are persons. They have two legs.”

Example: *“The professor writes papers and he supervises students.”*

Temporal Sentences Most sentences in a use case will have an event-oriented nature, since the purpose of use cases is to describe what a system does. Temporal sentences are in essence method sentences. The verb implies an action, which gives rise to an event from a temporal perspective and a method from a structural perspective.

The word *when* is used to indicate the beginning of an event flow. Continuations of events in the following sentences are indicated using *next* or *then.*

Conjunctions of event sentences like ”the professor thinks and he mumbles” indicate a *fork*; the events in the sentences occur independently and their order is not given. If the next sentence is a continuation, it will *join* the event flow.

Example: *“When a worm sees an apple, the worm enters the apple. When a boy sees an apple, he eats it. Next he burps and he throws the core.”*

3.3 Current Status

Incrementality is simulated by parsing the entire text and drawing new diagram when a period which is added or changed. Only a rudimentary user interface exists at the moment, but the current prototype qualifies as proof of concept for our ideas.

4 NLP Methods Applied

In the following, we assume a basic knowledge of logic programming and DCG, and explain the CHR specifics that are used.

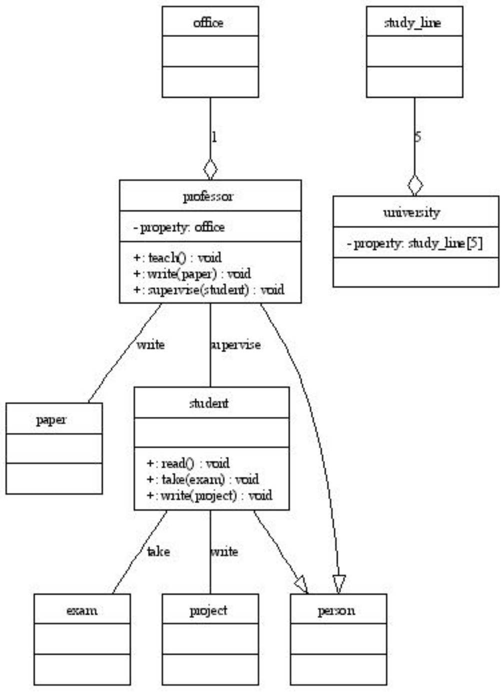


Fig. 3. UML class diagram generated as result of the collected sample sentences above, which contains examples of the three kinds of class relationships; association, aggregation and inheritance. Associations are labeled with corresponding verbs and aggregations are labeled with cardinalities.

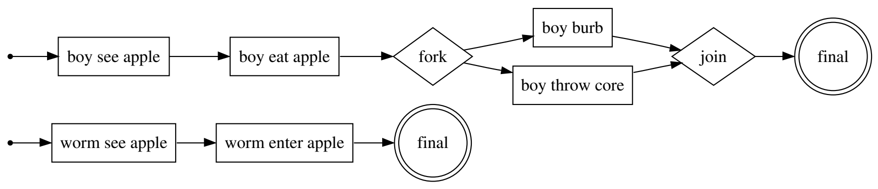


Fig. 4. UML Activity diagram generated from the temporal sentences example.

4.1 Overall Implementation Principles

We consider sentences that describe class hierarchy, e.g., "A dog is an animal". The following grammar rule gives the overall structure.

```
sentence -->
   fc_noun_phrase(Number, _, subj, IdSub),
   subord_verb(Number,_),
   fc_noun_phrase(_, Number, obj, IdObj),
   {extends(IdSub,IdObj)}.
```

Notice that it produces no explicit output via arguments, but updates the constraint store by the calls to constraints, i.e., extends by abductive reasoning the constraint store with those facts that seem to be the reason why the sentence can be stated. In this example, the rule depends on `extends(dog,animal)` and, since `extends` is declared as a constraints (and thus distinguished from ordinary Prolog predicates), it will be added to the constraint store if it was not there already. The grammar rules for noun phrases may, in a similar way, create the facts `class(dog)` and `class(animal)`. Thus the analysis of this sample sentence produces a store of three constraints that can be converted in a straightforward way into the input language of Graphviz, which in turn produces a class diagram showing that the class of dogs is a subclass of animals.

The second attribute of noun phrases is called CollectiveNumber; for "a cat and a dog" it evaluates to `sing` and for "cats and dogs" to `plur`. Taking the collective number for the subject as the number for the object allows "A dog is an animal and a pet" and excludes * "A dog is animals and pets". Noun phrases are divided into different categories with particular restrictions; for example, `fc_noun_phrase` (for fully specified class) used above is a category which forbids pronouns and quantified expressions like "two cats" in this particular sort of sentences. Similar categories are defined for `indiv_noun_phrase` referring to particular objects ("She, Peter, and Paul"), `rc_noun_phrase` for restricted class ("Mary and the boys" assuming that Mary is a prototype for some class), and `q_noun_phrase` for quantified expressions such as "four legs and a tail".

Noun phrases generate a representation of their contents, using a plus to combine conjunctive phrases. Here are some examples, assuming that Mary is a prototype woman and that "she" refers to Mary.

`fc_`	cats and dogs	`cat+dog`
`indiv_`	her, Peter, and Paul	`mary+peter+paul`
`rc_`	Mary and the boys	`woman+boy`
`q_`	a tail and some legs	`tail:1+legs:n`

Special codes are produced in case of errors within noun phrases such as unresolved or ambiguously used pronouns, or an attempt to use a named individual without a unique class as prototype. The following CHR rules unroll constraints with composite arguments into individual constraints.

```
extends(A+B,C) <=> extends(A,C), extends(B,C).
extends(A,B+C) <=> extends(A,B), extends(A,C).
```

These are so-called simplification rules, triggered each time an `extends` constraint with a plus in one of its arguments appear in the store. They delete the constraint(s) matched by the left-hand side, the head, and add those on the right, the body.

4.2 Expressing Knowledge About Use Case Modeling

CHR can be used to express knowledge about the domain in question. We can illustrate this by the way we aggregate the constraints emerging from different statements about the same property. We use `property(car,wheels:4)` to express that a car has four wheels and `property(car,doors:(2..5))` to say that it has between 2 and 5 doors. Consider the following CHR rule which is part of the implementation.

```
property(C,P:N), property(C,P:M) <=>
   q_count(N), q_count(M), q_less_eq(N,M)
   | property(C,P:(N..M)).
```

It applies when the store contains two `property` constraints for the same class and property, provided the guard is true. The guard is an optional part between the arrow and the vertical bar which here refers to Prolog predicates written specifically for the purpose, so that, say, `q_count(5)`, `q_count(n)`, and `q_less_eq(2,n)` are true. So "Peter has a dog. Paul has five dogs" yields `property(man,dog:1)` and `property(man,dog:5)` which by the rule above get replaced by `property(man,dog:(1..5))`. The following rule combines different intervals for multiplicity into one.

```
property(C,P:(N1..M1)),property(C,P:(N2..M2)) <=>
   q_min(N1,N2,N), q_max(M1,M2,M),
   property(C,P:(N..M)).
```

So `property(c,a:(1..7))` and `property(c, a:(5..n))` are converted into `property(c, a:(1..n))`. In total, five rules are used to manage multiplicity of properties.

4.3 Pronoun Resolution

Here we sketch briefly the approach inspired by the assumption principle of [30] but extended with a time stamp (here, sentence number) to realize the indicated principle. When, say, "Peter" is mentioned in sentence no. 7, a constraint `referent(sing,masc,peter,7)` is emitted, and an occurence of "him" in sentence no. 10 gives rise to `expect_referent(sing, masc,X)`; the following rule attempts to bind `X` to the suitable value.

```
sentence_no(Now), referent(No,G,Id,T) \
                      expect_referent(No,G,X) <=>
   T < Now,
   \+ ( find_constraint( referent(No,G,_,TMoreRecent),_),
        T < TMoreRecent, TMoreRecent\==Now)

    | ( find_constraint( referent(No,G,Id1,T), _),
        Id1\=Id
         -> X = error:pronoun:ambiguous(No,G,Now)
          ; X=Id ).
```

The rule is a so-called simpagation rule which, when applied, keeps the constraints before "\" in the store and removes the remaining ones up until the arrow. CHR does not allow negations in the head, so the test that time `T` designates the most recent relevant `referent` (i.e., there is no other such with a more recent time stamp) is done in less elegant way in the guard. The body tests for ambiguity and generates a special code which is converted into an error message elsewhere. Finally, the following rule catches unresolved pronouns if, e.g., the whole text start with "He".

```
sentence_no(Now) \ expect_referent(No,G,X) <=>
    X=error:pronoun:unresolved(No,G,Now).
```

The following grammar rule for using pronouns shows how the implemented `expect_referent` constraint can be used.

```
indiv_simple_noun_phrase(Num,Case,G,Id) -->
   pronoun(Num,Case,G),
   {expect_referent(Num,G,Id)}.
```

This example illustrates how relatively complicated contextual dependencies in logic grammars can be modeled in a fairly concise way using CHR. The use of prototypical individuals for classes is realized in a similar way. In "Mary is the boss. She manages the employees.", the pronoun is resolved to `mary`, and a call to a constraint `expect_class(···mary···)` locates the class `boss` (i.e., if it is unique, otherwise an error code) and the constraint `method(boss,manage,employees)` is created.

4.4 From Constraint Stores to UML Diagrams

The program "Graphviz" is used to create the graphical representation of an UML diagram given an input in graph drawing language. A DCG is defined to generate the input for the GraphViz input language after a use case has been processed. This grammar references the constraint store and generates a phrase as long as possible, thereby converting constraints into phrases to be given as input to GraphViz. If, for example, the constraint store contains `class(man)` and `method(man,walk)`, the nonterminal `class_node` generates the phrase `man[ label = "{man||: walk(): void\l}"]`. This is straightforward and not described further.

5 Conclusions and Future Work

We presented a system for analyzing a restricted natural language for use case writing, based on Definite Clause Grammars extended with Constraint Handling Rules. Our grammar captures candidate domain classes and their relations and visualize these using an UML class diagram. The syntax of the language is simple but expressive enough to model a given domain. The language seems natural and expressive but avoids inherently ambiguous sentence elements such as adverbs and adjectives.

By extending our grammar with Constraint Handling Rules, we are able to handle pronoun resolution with ambiguity detection, prototypical references (e.g. names) and allow the user to express knowledge about the domain, such as multiplicity, using simple prototypical sentences.

We have shown that the system has potential for extraction of dynamic information. We are currently working on extending language so more complex activity stories, including conditions and loops, can be expressed.

At this stage the implemented system qualifies only as proof of concept for our ideas. More work needs to be done in order to apply the tool in a realistic setting. It would be interesting to evaluate the system applied to large real world problems and to study how users would respond to the various usage scenarios.

Acknowledgement: This work is supported by the CONTROL project, funded by Danish Natural Science Research Council.

References

1. Pereira, F.C.N., Warren, D.H.D.: Definite clause grammars for language analysis—A survey of the formalism and a comparison with augmented transition networks. Artificial Intelligence **13**(3) (1980) 231–278
2. Frühwirth, T.: Theory and practice of constraint handling rules, special issue on constraint logic programming. Journal of Logic Programming **37**(1–3) (October 1998) 95–138
3. Christiansen, H., Dahl, V.: Meaning in Context. In Dey, A., Kokinov, B., Leake, D., Turner, R., eds.: Proceedings of Fifth International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT-05). Volume 3554 of Lecture Notes in Artificial Intelligence. (2005) 97–111
4. Christiansen, H., Dahl, V.: HYPROLOG: A new logic programming language with assumptions and abduction. In Gabbrielli, M., Gupta, G., eds.: ICLP. Volume 3668 of Lecture Notes in Computer Science., Springer (2005) 159–173
5. Object Management Group: Unified Modeling Language (UML), version 2.0. Object Management Group, Framingham, Massachusetts. (October 2004)
6. Fuchs, N.E.: Attempto controlled english. In: WLP. (2000) 211–218
7. Sowa, J.F.: *Common Logic Controlled English* (2004) Draft, http://www.jfsowa.com/clce/specs.htm.
8. Liu, H., Lieberman, H.: Toward a programmatic semantics of natural language. In: VL/HCC, IEEE Computer Society (2004) 281–282
9. Liu, H., Singh, P.: Conceptnet: A practical commonsense reasoning toolkit (May 02 2004)

10. Liu, Hugo, Lieberman, Henry: Programmatic semantics for natural language interfaces. In: Proceedings of ACM CHI 2005 Conference on Human Factors in Computing Systems. Volume 2 of Late breaking results: short papers. (2005) 1597–1600
11. Drazan, J., Mencl, V.: Improved processing of textual use cases: Deriving behavior specifications. In: Proceedings of SOFSEM 2007. Volume 4362 of Lecture Notes in Computer Science., Springer (2007)
12. Fantechi, A., Gnesi, S., Lami, G., Maccari, A.: Application of linguistic techniques for use case analysis. In: RE, IEEE Computer Society (2002) 157–164
13. Harmain, H.M., Gaizauskas, R.J.: Cm-builder: A natural language-based case tool for object-oriented analysis. Autom. Softw. Eng. **10**(2) (2003) 157–181
14. Kiyavitskaya, N., Zeni, N., Mich, L., Mylopoulos, J.: Experimenting with linguistic tools for conceptual modelling: Quality of the models and critical features. In Meziane, F., Métais, E., eds.: NLDB. Volume 3136 of Lecture Notes in Computer Science., Springer (2004) 135–146
15. Estratat, M., Henocque, L.: An intuitive tool for constraint based grammars. Volume 3438., Springer (2004)
16. Abbott, R.J.: Program design by informal English descriptions. Communications of the ACM **26**(11) (1983) 882–894
17. Jacobson, I.: Object oriented development in an industrial environment. In: OOPSLA. (1987) 183–191
18. Booch, G., Jacobson, I., Rumbaugh, J.: The Unified Modeling Language User Guide. Addison-Wesley (1999)
19. Berard, E.V.: Be Careful With 'Use Cases'. The Object Agency, Inc. (August 1998)
20. Loomis, M.E.S., Shah, A.V., Rumbaugh, J.E.: An object modeling technique for conceptual design. In Bézivin, J., Hullot, J.M., Cointe, P., Lieberman, H., eds.: ECOOP '87, European Conference on Object-Oriented Programming. Volume 276 of Lecture Notes in Computer Science., Springer-Verlag (1987) 192–202
21. Booch, G., Rumbaugh, J.: Unified Method for Object-Oriented Development Version 1.0. Rational Software Corporation (1997)
22. Cockburn, A.: Structuring use cases with goals. Journal of Object-Oriented Programming (SeptemberOctober 1997)
23. Achour, C.B.: Guiding scenario authoring. In: EJC. (1998) 152–171
24. Cox, K., Phalp, K.: Replicating the CREWS use case authoring guidelines experiment. Empirical Software Engineering **5**(3) (2000) 245–267
25. Karl Cox, K.P., Shepperd, M.: Comparing use case writing guidelines. In: Seventh International Workshop on Requirements Engineering (RE'01). (June 2001)
26. Abdennadher, S., Christiansen, H.: An experimental CLP platform for integrity constraints and abduction. In: Proceedings of FQAS2000, Flexible Query Answering Systems: Advances in Soft Computing series, Physica-Verlag (Springer) (2000) 141–152
27. Christiansen, H.: A constraint-based bottom-up counterpart to definite clause grammars. In Nicolov, N., Bontcheva, K., Angelova, G., Mitkov, R., eds.: RANLP. Volume 260 of Current Issues in Linguistic Theory (CILT)., John Benjamins, Amsterdam/Philadelphia (2004) 227–236
28. Christiansen, H.: CHR Grammars. Int'l Journal on Theory and Practice of Logic Programming **5**(4-5) (2005) 467–501
29. Hobbs, J.R., Stickel, M.E., Appelt, D.E., Martin, P.A.: Interpretation as abduction. Artif. Intell. **63**(1-2) (1993) 69–142
30. Dahl, V., Tarau, P., Li, R.: Assumption grammars for processing natural language. In: ICLP. (1997) 256–270

31. Tarau, P., Bosschere, K.D., Dahl, V., Rochefort, S.: Logimoo: An extensible multi-user virtual world with natural language control. J. Log. Program. **38**(3) (1999) 331–353
32. Mitkov, R.: Anaphora Resolution. Longman (Pearson Education) (2002)

A CHRG Analysis of ambiguity in Biological Texts

(Extended Abstract)

Veronica Dahl and Baohua Gu

Logic and Functional Programming Group,
School of Computing Science, Simon Fraser University,
Burnaby, B.C. V5A 1S6 Canada
{veronica,bgu}@cs.sfu.ca

Abstract. We propose a methodology for analyzing human language sentences which can efficiently choose between alternative readings springing from the interaction between coordination and preposition phrase attachment. We present a proof-of-concept in terms of an extremely succinct CHRG [3] analyzer for interpreting biological text titles. Our method uses expert knowledge on semantic types and compatibilities among them, as well as contextual facilities of CHRG to gain an overall view of the sentence components involved in disambiguation.

1 Introduction

This work was inspired by our efforts to automatically extract concepts from biological text, where one of the main challenges faced is the amount of information succinctly packed into titles. Typically, biological named entities appear as acronyms or in condensed versions, and the amount of information is maximized by heavy use of coordination within noun phrases. As well, they have a tendency to contain several prepositional phrases with no clear indication of what antecedent they should attach to. For example, given the title sentence of a Medline abstract: "*IL-2 gene expression and NF-kappa B activation through CD28 requires reactive oxygen production by 5-lipoxygenase*", it is hard to see whether the activation through *CD28* refers only to *NF-kappa B* or to both *IL-2 gene* and *NF-kappa B.*

The ambiguity involved in titles containing even one instance of coordination or preposition phrase attachment is a challenge, and when two or more coexist in the same title, the number of possible interpretations explodes, making it extremely difficult for a naive automated system to cope with.

However, it is not unusual to find, within the text or in related knowledge bases such as biological dictionaries and ontologies (e.g., the GENIA Ontology [8]), short descriptions of what the entities' names refer to, or at the very least their semantic types (e.g., protein molecule, DNA family or group). The short descriptions are often contained in simple constructs named appositions (e.g., as in "*Grf40, a novel Grb2 family member*""), and their semantic types can very often be found directly in available taxonomies (e.g., the GENIA corpus [7]), or inferred from other related knowledge bases (e.g., [8]).

We have found that by extracting the semantic class to which each named entity refers, and by considering the relationships the sentence involves it in, we can discard many of the ambiguities that originate in the coordination constructions where they intervene. In this paper we propose an analysis of ambiguity in compact text in general, while focusing on biological texts' titles in particular for ease of demonstration and exemplification.

2 Analysis of the Ambiguities Most Commonly Present in Biological Titles

2.1 An Example

The typical features of titles and similar corpora are:

a) the entities are referred to through abbreviations or acronyms;
b) coordination is quite common;
c) prepositional phrase attachment ambiguities are very common too;

The following sentence, taken from the GENIA corpus [7], illustrates:
IL-2 gene expression and NF-kappa B activation through CD28 requires reactive oxygen production by 5-lipoxygenase.

In this sentence, coordination interacts with preposition phrase attachment with highly ambiguous results: the sentence could mean either (note that each reading is the conjunction of two simpler sentences, noted with a) and b) below):

- Reading (1):
  1a. IL-2 gene expression through CD28 requires reactive oxygen production by 5-lipoxygenase.
  1b. NF-kappa B activation through CD28 requires reactive oxygen production by 5-lipoxygenase.
- Reading (2):
  2a. IL-2 gene expression requires reactive oxygen production by 5-lipoxygenase.
  2b. NF-kappa B activation through CD28 requires reactive oxygen production by 5-lipoxygenase.

At first glance, deciding among these possible readings appears an un-surmountable task. However, if we simply retrieve the semantic types or classes to which each entity belongs, and note whether such classes can meaningfully appear as arguments of the relationships in which they are involved, many of the possible readings fade away. Any remaining ones will in general be those that are also ambiguous for a human expert in the biological notions involved.

2.2 Our methodology

To analyze sentences such as the above, we first consult the GENIA corpus, a repository of annotations for every biological named entity in biological text in order to attach each abbreviation or acronym to a) its full name and b) its semantic type. In those cases where this information is not present, we look for an apposition either in the text that accompanies the title we are analyzing, or in related texts.

In the previous section's example, a lookup for *IL-2 gene* in the GENIA corpus yields the name *IL-2 gene* and the semantic type *DNA domain or region.* On the other hand, it could be that an entity is not annotated as a biological entity in the GENIA corpus, but we find it within the text or within some other corpus, in an apposition which can point us to the type. Even when an entity is found in the GENIA corpus, consultation of the appositions which further define it may be useful. For instance, from "*Grf40 , A novel Grb2 family member, is involved in T cell signaling through interaction with SLP-76 and LAT.*", we can infer that the protein molecule *GRf40* further belongs to the subfamily *Grb2.*

Our problem now reduces to encode which semantic types make sense in each argument of each of the biological relationships we most commonly encounter in biological texts. This information has to be constructed from an expert's knowledge. The Appendix shows a prototype implementation of our methodology as a first step in demonstrating its usefulness.

3 Exploiting ontological information

One of the files we consult in our system is the GENIA ontology, which expresses subtype relationships in the format exemplified by:

```
subtype("Natural_source",    "Source").
```

In another file we have the expert knowledge about compatibility between biomedical concepts, expressed in our system as constants or as types, e.g.

```
compatible(IL-2_gene_expression','Protein_molecule').
```

Such information will be consulted from our grammar rules for disambiguation. For instance, the example in 2.1 is easily taken care of by just one CHRG rule, namely

```
np(A), prep(P), np(B) /- (verb(_); prep(_); eos(_)) <:>
  subtype(A,'BioProcess'), subtype(B,'BioEntity'), compatible(A,B)
  | np(A+P+B).
```

This rule will create a noun phrase from two noun phrases (represented A and B) joined by a preposition (P), provided that the second np is flanked by either a verb, another preposition, or and end of sentence character, and provided that A's type and B's type are compatible (as well as being a subtype of, respectively, 'BioProcess' and 'BioEntity'). The ontology is consulted, of course, when checking the desired subtype relationships.

4 Criticality of CHRG for our methodology

The use of CHRG is crucial to our approach, since

a) it allows us to work bottom-up, thereby heading more directly to the right analysis.

b) it allows us to put together into the same rule information coming from heterogeneous sources. For instance, we mine ontology information from the Genia Ontology [8], and can consult or effect transformations of that information that suit disambiguation purposes. Left hand sides of CHRG rules do not care from which file (among the ones having been read) the information that needs to be put together comes from, so this gives a great degree of modularity and allows us to incorporate information from heterogenous sources.

c) in the case of the text,or even of some titles containing appositions that define a given term. If we consider for instance the sentence: "*Overexpression of DR-nm23, a protein encoded by a member of the nm23 gene family, inhibits granulocyte differentiation and induces apoptosis in 32Dc13 myeloid cells*", we can glean some definitional information for *DR-nm23.* The level of granularity with which we want to take advantage of this feature will vary according to our purposes. We might be content for instance with noting only that it is a protein, in which case the rest of the sentence can be ignored (CHRG includes a facility for disregarding intermediate strings which will not be analyzed) or that it is a protein and is encoded by a member of the *nm23* gene family, or that it is a protein, is encoded by a member of the *nm23* gene family, and induces apoptosis, and so on. Likewise, the information that *nm23* is a gene can be gleaned from the same sentence. In other words, we can implement a specialized CHRG which only looks at appositions within a text, disregards the rest of the text, and decides how to usefully exploit the information in the apposition: will it be used to consult the type hierarchy, to expand it, or just to add a definition into the database we are working with?

d) the use of CHRGs allows for a straightforward coexistence with CHR [6] rules, and even for the same symbols to be considered both as grammar symbols or as constraints. This is exemplified by the coexistence of the grammar rule described in Section 3 with the CHR rule:

```
np(X), np(Y), compatible(A,B) ==>
            subtype(X,A), subtype(Y,B) | compatible(X,Y).
```

which extends the user's definitions of compatibility by considering that if the user has described A and B as being compatible, and the parser has discovered X and Y as noun phrases, where X is a subtype of A and Y is a subtype of B, X and Y are also compatible. This inferred compatibility information can then be used by the grammar rule in Section 3, since it is now in the constraint store.

We do not know of any other system which so seamlessly would allow us to combine grammar and program rules for similar interactions needed.

As well, the facility of CHRG for looking at context allows us to implement the above described methodology with extreme conciseness. The full prototype program takes only one page and is included in full in the Appendix.

It is to be noted that as a side effect of disambiguation, our implementation completes the meaning of coordinated sentences which do not overtly contain all the conjuncts. Previous work for reconstructing elided meanings within coordinated phrases in natural language typically take more machinery for their implementation, e.g. computing parallelism in discourse, or further tools such as assumptions and Datalog grammars [4] [5].

Let us exemplify with the same sample sentence, taken from a real life title, which we showed in Section 1. Semantic types are described through binary predicates of the form `type(Entity,Class)`, e.g. from the program in the Appendix we can see that the semantic type of "*IL-2_gene-expression*" is "*bioprocess*". As well, our expert knowledge base includes information on compatibilities , from which we can know for instance that "*IL-2_gene-expression* is compatible with *CD28*.

The rules that analyze conjoined noun phrases consult such information and take appropriate action by conjoining only those components that are compatible in type, and likewise appropriately attaching any prepositional phrases. Thus, in the above example, the second reading is simply not accessible from the grammar rules given, since they fail to satisfy the compatibility condition.

5 Discussion

We have proposed a CHRG methodology to disambiguate multiple readings of sentences in biological text, on the basis of compatibilities between semantic types, which are calculated on the fly by consulting the GENIA ontology and dynamically extending user defined, basic relationships on compatibilities. Our parsing technique integrates semantics at the lexical level, exploiting an ontology for the application domain (biological texts).

We mix grammar rules and CHR proper rules to allow productive interaction between domain constraints and grammatical constraints. As explained in Section 4, this makes it easier to express our problem in directly executable terms.

Our approach uses includes expert knowledge on semantic types of named entities and on compatibility between entities based on their semantic types. Appositions can provide further information about an entity of interest, as we also saw. Some appositions may even throw light upon relationships between two entities.

We have shown that this approach allows us to very succinctly express within the grammar rules the conditions under which alternative readings originating in preposition attachment plus coordinating ambiguities should be chosen.

We have exemplified our methodology for the particular problem of PP-attachment in coordinate constructions, and tested it with a first running prototype which is shown in the Appendix. These first results show that much simpler machinery can be arrived at within our methodology than was previously the case in related work on coordination, including our own work with Datalog grammars and assumptions [4] and even CHR [5]. Part of this is due to the restriction of our domain to a well-investigated domain for which online corpora and ontologies exist, namely the biological domain, but as also pointed out, a bigger part is due to the use of CHRG rules which focus on the relevant context seen overall, checks on types and their compatibilities, and uses this information to decide how to form meaning from the meanings of the involved parts.

However, we yet have to combine the advantages obtained in the present work with other long distance dependency work around CHR [6], for a uniform, more encompassing treatment, perhaps along the lines suggested in citeDahl-2004. Our present focus on titles allows us to get away with "just" allowing coordination among the possible long-distance dependency phenomena.

With this work we hope to stimulate further research into the subject.

Acknowledgements This work is supported by the CONTROL project, funded by Danish Natural Science Research Council, and by Veronica Dahl's NSERC Discovery Grant.

References

[1] Aguilar Solis, D. and Dahl, V.: Coordination revisited: a CHR approach. In Proc. Iberamia '04, Mexico.

[2] CHRG User Manual. http://akira.ruc.dk/ henning/chrg/

[3] Henning Christiansen: CHR grammars. Theory and Practice of Logic Programming, 5(4-5): 467-501 (2005)

[4] Dahl, V.: On Implicit Meanings. In: Computational Logic: From Logic Programming into the Future . F. Sadri and T. Kakas (eds). (invited contribution), volume in honour of Bob Kowalski, Springer-Verlag, 2002.

[5] Dahl, V.: Treating Long-Distance Dependencies through Constraint Reasoning. In Proc. of the 3rd International Workshop on Multiparadigm Constraint Programming Languages, 2004.

[6] Frhwirth, T.: Theory and Practice of Constraint Handling Rules, Special Issue on Constraint Logic Programming (P. Stuckey and K. Marriot, Eds.), Journal of Logic Programming, Vol 37(1-3), pp 95-138, October 1998.

[7] GENIA Corpus: http://www-tsujii.is.s.u-tokyo.ac.jp/ genia/topics/Corpus/

[8] GENIA Ontology: http://www-tsujii.is.s.u-tokyo.ac.jp/ genia/topics/Corpus/genia-ontology. html

Appendix A: the prototype CHRG implementation for title disambiguation

```
% the CHR rules and CHR grammar rules used for disambiguation


:- compile('chrg.txt').

handler         simple_coordination_solver.

constraints     np/1, compatible/2, subtype/2.

grammar_symbols sentence/1, subj/1, verb/1, obj/1,
                np/1, conj/1, prep/1, eos/1,
                compatible/2, subtype/2.

% to induce new subtype relations

np(A), subtype(A,B), subtype(B,C) ==> subtype(A,C).

% to induce new compatible relations

np(X), np(Y), compatible(A,B) ==>
             subtype(X,A), subtype(Y,B) | compatible(X,Y).
np(X), compatible(A,B) ==> subtype(X,A) | compatible(X,B).
np(Y), compatible(A,B) ==> subtype(Y,B) | compatible(A,Y).

%% grammar rules to group a np with a following preposition phrase

np(A), prep(P), np(B), conj(K), np(C) <:> np(A+P+B), conj(K), np(A+P+C).

np(A), prep(P), np(B) /- (verb(_); prep(_); eos(_)) <:>
    subtype(A,'BioProcess'), subtype(B,'BioEntity'), compatible(A,B)
  | np(A+P+B).

np(A), conj(K), np(B+P+C) <:>
    subtype(A,'BioProcess'), subtype(C,'BioEntity'), compatible(A,C)
  | np(A+P+C), conj(K), np(B+P+C).

%% rules to classify noun phrases as subjects or objects of verbs

np(A)   /-   verb(_)  ::> subj(A).
verb(_)  -\  np(A)    ::>  obj(A).

%% rules to handle coordinations

  np(A), conj(_), subj(B)  ::> subj(A), subj(B).
 obj(A), conj(_),   np(B)  ::>  obj(A),  obj(B).

%% to identify a complete sentence
```

```
subj(A), verb(V),  obj(B) ::> sentence(s(A,V,B)).

sentence(A), conj(_), sentence(B) <:> sentence(A+B).

%% to solve incomplete sentences

subj(A), verb(V)   /- conj(_), sentence(s(_,_,B))
                               ::> sentence(s(A,V,B)).
sentence(s(A,_,_)) -\ conj(_), verb(V), obj(B)
                               ::> sentence(s(A,V,B)).
subj(A)            -\ conj(_), sentence(s(_,V,B))
                               ::> sentence(s(A,V,B)).
sentence(s(A,V,_)) -\ conj(_),  obj(B)
                               ::> sentence(s(A,V,B)).
subj(A), verb(V1), conj(_), verb(V2), obj(B)
                               ::> sentence(s(A,V1,B)),
                                   sentence(s(A,V2,B)).

% include example sentence, ontology, concepts, and domain knowledge

:- include('test_example.txt').     % sentences for testing
:- include('genia_ontology.txt').   % the GENIA Ontology
:- include('genia_concepts.txt').   % annotations from GENIA corpus
:- include('compatibility.txt').    % compatibility between concepts

end_of_CHRG_source.
```

(N.B. for the referees: you can download the example1.txt and other title phrases or title sentences of the GENIA corpus we are considering here from www.cs.sfu.ca/ bgu/personal/CSLP2007)

Appendix B: the Auxiliary Files

```
% the content of file "genia_ontology.txt"

subtype('BioEntity',  'BioConcept').
subtype('BioProcess', 'BioConcept').
subtype('Source',    'BioEntity').
subtype('Substance', 'BioEntity').

subtype('Natural_source',    'Source').
subtype('Artificial_source', 'Source').

subtype('Organism',       'Natural_source').
subtype('Body_part',      'Natural_source').
subtype('Tissue',         'Natural_source').
subtype('Cell_type',      'Natural_source').
subtype('Cell_component', 'Natural_source').

subtype('Other_artificial_source', 'Artificial_source').
subtype('Cell_line',               'Artificial_source').

subtype('Multi_cell', 'Organism').
subtype('Mono_cell',  'Organism').
subtype('Virus',      'Organism').

subtype('Compound', 'Substance').
subtype('Atom',     'Substance').

subtype('Organic',   'Compound').
subtype('Inorganic', 'Compound').

subtype('Amino_acid',             'Organic').
subtype('Nucleic_acid',           'Organic').
subtype('Lipid',                  'Organic').
subtype('Carbohydrate',           'Organic').
subtype('Other_organic_compound', 'Organic').

subtype('Protein',            'Amino_acid').
subtype('Peptide',            'Amino_acid').
subtype('Amino_acid_monomer', 'Amino_acid').

subtype('DNA',           'Nucleic_acid').
subtype('RNA',           'Nucleic_acid').
subtype('Nucleotide',    'Nucleic_acid').
subtype('Polynucleotide', 'Nucleic_acid').

subtype('Protein_family_or_group',  'Protein').
subtype('Protein_complex',          'Protein').
subtype('Protein_molecule',         'Protein').
subtype('Protein_subunit',          'Protein').
subtype('Protein_substructure',     'Protein').
subtype('Protein_domain_or_region', 'Protein').
subtype('Protein_ETC',              'Protein').

subtype('DNA_family_or_group',  'DNA').
subtype('DNA_molecule',         'DNA').
subtype('DNA_domain_or_region', 'DNA').
subtype('DNA_substructure',     'DNA').
subtype('DNA_ETC',              'DNA').

subtype('RNA_family_or_group',  'RNA').
subtype('RNA_molecule',         'RNA').
subtype('RNA_domain_or_region', 'RNA').
subtype('RNA_substructure',     'RNA').
subtype('RNA_ETC',              'RNA').

% the content of file "genia_concepts.txt"
% sample domain knowledge about the types of bioconcepts

subtype('IL-2_gene_expression',       'BioProcess').
subtype('NF-kappa-B_activation',      'BioProcess').
subtype('reactive_oxygen_production', 'BioProcess').

subtype('CD28',           'Protein_molecule').
subtype('5-lipoxygenase', 'Protein_molecule').

% the content of file "compatibility.txt"
% sample domain knowledge about compatibility between bioconcepts

compatible('IL-2_gene_expression',       'Protein_molecule').
compatible('NF-kappa-B_activation',      'Protein').
compatible('reactive_oxygen_production', 'Protein_molecule').
```

```
% the content of file "test_example.txt"
% a sample sentence to disambiguate
% we assume that base NPs have been identified beforehand

s1 :- X = ['IL-2_gene_expression', and, 'NF-kappa-B_activation',   \
           through, 'CD28', requires, 'reactive_oxygen_production', \
           by, '5-lipoxygenase', '.'], parse(X).

['IL-2_gene_expression']       <:> np('IL-2_gene_expression').
[and]                          <:> conj(and).
['NF-kappa-B_activation']      <:> np('NF-kappa-B_activation').
[through]                      <:> prep(through).
['CD28']                       <:> np('CD28').
[requires]                     <:> verb(require).
['reactive_oxygen_production'] <:> np('reactive_oxygen_production').
[by]                           <:> prep(by).
['5-lipoxygenase']             <:> np('5-lipoxygenase').
['.']                          <:> eos('.').
```

Appendix C: the Execution of Testing Sentences

```
% the execution results of the testing sentence on SICSTUS 3.8.4

| ?- s1.
 <0> IL-2_gene_expression <1> and <2> NF-kappa-B_activation   \
 <3> through <4> CD28 <5> requires <6> reactive_oxygen_production   \
 <7> by <8> 5-lipoxygenase <9> . <10>

all(0,10),
begin(-1,0),
end(10,11),
subtype('NF-kappa-B_activation','BioProcess'),
subtype('CD28','BioEntity'),
compatible('NF-kappa-B_activation','CD28'),
subtype('IL-2_gene_expression','BioProcess'),
subtype('CD28','BioEntity'),
compatible('IL-2_gene_expression','CD28'),
subtype(reactive_oxygen_production,'BioProcess'),
subtype('5-lipoxygenase','BioEntity'),
compatible(reactive_oxygen_production,'5-lipoxygenase'),
verb(5,6,require),
np(0,5,'IL-2_gene_expression'+through+'CD28'),
subj(0,5,'IL-2_gene_expression'+through+'CD28'),
conj(0,5,and),
np(0,5,'NF-kappa-B_activation'+through+'CD28'),
subj(0,5,'NF-kappa-B_activation'+through+'CD28'),
obj(6,7,reactive_oxygen_production),
sentence(0,7,s('NF-kappa-B_activation'+through+'CD28',require, \
                reactive_oxygen_production)),
sentence(0,7,s('IL-2_gene_expression'+through+'CD28',require, \
                reactive_oxygen_production)),
eos(9,10,'.'),
np(6,9,reactive_oxygen_production+by+'5-lipoxygenase'),
obj(6,9,reactive_oxygen_production+by+'5-lipoxygenase'),
sentence(0,9,s('NF-kappa-B_activation'+through+'CD28',require,  \
                reactive_oxygen_production+by+'5-lipoxygenase')),
sentence(0,9,s('IL-2_gene_expression'+through+'CD28',require,   \
                reactive_oxygen_production+by+'5-lipoxygenase')) ?

yes
| ?-
```

Applying Constraints derived from the Context in the process of Incremental Sortal Specification of German *ung*-Nominalizations

Kristina Spranger and Ulrich Heid

IMS, Universität Stuttgart, 70 174 Stuttgart, Germany

Abstract. Many German nominalizations with the affix *-ung* are sortally ambiguous. Within a sentence, lexico-semantic and/or syntactic phenomena may support disambiguation. The sortal interpretation of a nominalization may vary depending on the underlying syntactic analysis of one and the same, syntactically ambiguous sentence.
We model the process of sortal disambiguation as a constraint-based incremental process. The process is incremental as it evaluates in subsequent steps constraints from increasingly larger context windows.

1 Introduction

In this article, we present work towards the automatic interpretation of German nominalizations with the affix *-ung*, such as *Lieferung* (*delivery*) or *Messung* (*measurement*). Many such *-ung*-nominalizations are ambiguous with respect to their sortal interpretation (cf. [3] - who lean heavily on [8] and [7] - for the notion of sortal ambiguity). In section 2, a more detailed discussion on sortal ambiguity as regards German *-ung*-nominalizations is given.
We are working towards a system for data extraction from corpus text that is able to carry out sortal disambiguation. Given the productivity of the *-ung*-formation process in German (cf. [4] and [12]) and the high frequency of *-ung*-nominalizations in text (cf. [6] or [10]), this ability is relevant, among others, for question answering or high quality information extraction[1].
In this work, we analyze the influence of the context of an *ung*-nominalization on its sortal interpretation. Relevant contextual phenomena include lexical combination partners of the nominalization and/or the surrounding syntactic structures. As we rely on parsed corpus data, we have no discourse representations available that go beyond the sentence level. Thus, we have to limit the interpretation process to the sentence context, even though the disambiguation of some *-ung*-nominalizations would require a larger context.
In a preliminary case study (see section 3), we have identified some of the contextual phenomena which constrain the sortal interpretation of *Messung*. From a

[1] A more detailed discussion on the relevance of this ability in natural language processing systems is given in [11].

descriptive perspective, such phenomena serve as “indicators” of sortal readings. We model the process of sortal interpretation as a process of incremental specification where the context of a nominalization is used for its sortal interpretation. Knowledge about the reading indicators is explicitly formulated as constraints that are applied to a given nominalization. In section 4, we explain the specification process in detail.
In the process of sortal disambiguation, the order in which different constraints are applied is crucial for the interpretation of a nominalization at the sentence level. Obviously, the order in which the constraints are applied depends on the syntactic analysis of the targeted sentence, and syntactic ambiguity leads to multiple syntactic analyses. In section 5, we demonstrate that the sortal interpretation depends on the underlying syntactic analysis.
We conclude in section 6, addressing relevant aspects of a planned underspecified representation to make allowance for the effect of syntactic ambiguity and pointing to some more directions of future work.

2 Ambiguous German Nominalizations with *-ung*

German verb nominalizations with *-ung* are up to three-fold ambiguous concerning their sortal interpretation. They may have an event reading, a (result) state reading, and an object reading[2].

2.1 The Sortal Interpretation of *ung*-Nominalizations

According to [3] the primary distinction is the distinction between (i) eventualities and (ii) objects:

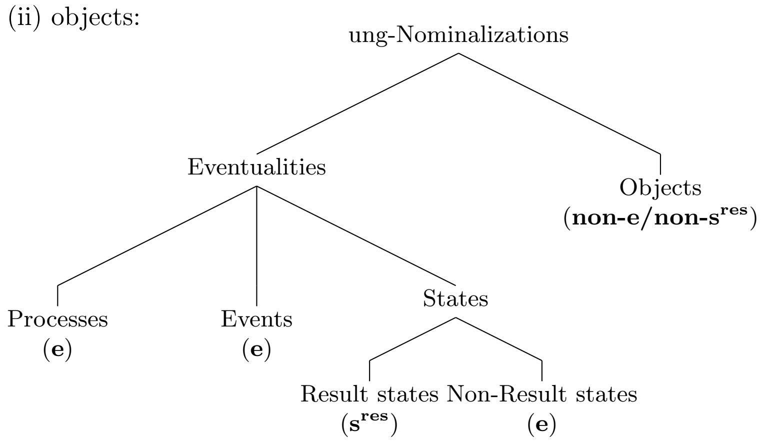


Fig. 1. The Sortal Interpretation of German *-ung*-Nominalizations

[2] For a more detailed discussion cf. [3], and [13] and [5] whose works are based on the theory developed in [3].

Eventualities Ehrich and Rapp subsume processes, events, and states under the concept of eventualities taken over from [2].
Events refer to telic actions whereas processes refer to atelic actions[3]. According to [9] processes as well as events can be seen as event complexes that are an association of a goal event, or “culmination” with a “preparatory phase” by which it is accomplished and a “consequent state” which ensues.
States (result states as well as non-result states) refer to eventualities that do not have a dynamic preparatory phase. Result states (e.g. *Absperrung* (*roadblock*)), in contrast to non-result states (e.g. *Bewunderung* (*admiration*)), are caused by a preceding event. Therefore, we distinguish between result states and other eventualities (including non-result states).
In the following, processes, events and non-result states are referred to by **e**, result states are referred to by $\mathbf{s}^{res}$.

Objects Objects refer to physical as well as abstract objects. They are referred to by **non-e/non-s**^{res}.

2.2 Distributional Tests

Except for non-result states and objects all classes of *-ung*-nominalizations (cf. figure 1) refer to some phase in the event complex as it is described by Moens and Steedman (cf. [9]): result states refer to the post-culmination phase, and events and processes refer to the whole event complex. Thus, it is especially challenging to keep them apart. To this end, Ehrich and Rapp propose a number of distributional tests:

1. Only eventualities allow to refer to phases of the events (a) and can be combined with process modifying predicates (b):
   (a) Die Verfolgung des Täters / Die Absperrung des Geländes
   The pursuit the perpetrator / The cordon the area
   beginnt / **hört auf** / **wird unterbrochen**.
   starts / stops / is interrupted.
   ‘The pursuit of the perpetrator / The cordon of the area starts / stops / is interrupted.’
   (b) die **umständliche** / **vorsichtige** Verfolgung des Täters /
   the awkward / cautious pursuit the perpetrator /
   Absperrung des Geländes
   cordon the area
   ‘the awkward / cautious pursuit of the perpetrator / cordon of the area.’
2. Result states can be combined with stative predicates (a) and with predicates of perceptibility (b) (summed up as “static predicates”):

[3] According to [15] events are “accomplishments” and “achievements”, and processes are “activities”.

(a) die **bestehende** Absperrung des Geländes
the existing cordon the area
'the existing cordon of the area'

(b) die **vorgefundene** / **kartographisch registrierte** Absperrung des Geländes
the found / cartographically registered cordon the area
'the cordon of the area found / cartographically registered'

3. Duration predicates can only occur together with processes and result states:
   - die **tagelange** Verfolgung des Täters / Absperrung des Geländes
     the lasting for days pursuit the perpetrator / cordon the area
     'the pursuit of the perpetrator / cordon of the area lasting for days'
4. Events can go together with time frame predicates (a) and they allow to refer to the incremental progression of the event (b):

(a) die **in zwei Tagen** erfolgte Absperrung des Geländes
the in two days accomplished cordon the area
'the cordon of the area accomplished in two days'

(b) die **allmähliche** Absperrung des Geländes
the gradual cordon the area
'the cordon of the area completed step by step'

The distributional tests show that event nominalizations and result state nominalizations are distributed complementarily.

3 The *-ung*-Nominalization *Messung*: A Case Study

The nominalization *Messung* (*measurement*) is two-fold ambiguous: it allows for an event interpretation (**e**), and for an object interpretation (**non-e**)[4].

3.1 Sortal Readings of *Messung*

The event reading of *Messung* refers to the process of measuring. Sentence (1) is a typical context for *Messung* as an event.

(1) die **Messung** des Erdumfangs durch Eratosthenes
the measuring the circumference of the earth by Eratosthenes
'the measuring of the circumference of the earth by Eratosthenes'

The object reading refers to the result of a measuring process, i.e. to data or figures. Sentence (2) is a context for *Messung* as an object.

[4] For the sake of convenience, we do without **non-s**^{res} since there is no result state interpretation of *Messung*.

(2) Die **Messungen** liegen unter dem zulässigen Grenzwert von 250 ppm.
The measurements lie under the acceptable critical value of 250 ppm.
'The measurements are lower than the maximum permissible value of 250 ppm.'

3.2 Disambiguating Reading Indicators from the Context

To decide about the sortal interpretation of an *-ung*-nominalization, humans seem to use lexico-semantic and syntactic reading indicators from the context. Many lexical indicators are combinatory constraints of lexico-semantic nature, ranging from preferences for general (ontological) classes, over selection restrictions, to lexeme-specific combinations. Some such indicators have been used by [3] to formulate their distributional tests (cf. section 2.2). We list more such indicators for event and object readings of *Messung* in tables 1 and 2. These indicators have been derived from a manual analysis of circa 400 sentences newspaper text.

Type	Examples
Reference to Event Phase	nominalization as subject: *Messung geht weiter*
	nominalization as object: *Messung aufnehmen, fortsetzen, abschliessen*
Duration predicates	adjectives: *fortlaufende, kontinuierliche Messungen*
	temporal PPs: *während der Messung*
Selection Restriction of Verbs of Order	*Messung anordnen, vorschreiben, veranlassen*
Lexical Collocations	support verbs: *Messung findet statt*, *Messungen durchführen*
Local/Temporal Adjuncts	*Messungen an Strassen*, *Messungen im Sommer*

Table 1. Event Indicators

Type	Examples
Static Predicates	*Messungen liegen vor*
Value Indicating Verbs	*M. liegt bei* <`value`>
Use with Proving Verbs	*Messung beweist/zeigt, dass*; *jmd. zieht aus der Messung den Schluss, dass*

Table 2. Object Indicators

In a given sentence, the lexical indicators may appear in different syntactic structures. For example, a support verb which has the nominalization as its object may also come as a prenominal participle or in a relative clause. Moreover, roughly synonymous indicators may belong to different word classes.

4 Incremental Sortal Specification in Context Using Context-Derived Constraints

Taking the contextual phenomena we have identified to constrain the sortal interpretation of *Messung* as a starting point, we will show in the following how the knowledge about these reading indicators is explicitly formulated as constraints, and how these constraints are used in an incremental process of sortal disambiguation.

4.1 Competing Reading Indicators

In many cases there is more than one indicator in a sentence, and not all indicators present in a given sentence support the same sortal reading. Sentence (3), for example, contains two indicators: one for the object reading and one for the event reading.

(3) Die Geologen beschreiben Messungen [**auf den Seychellen**]$_e$, [die
The geologists describe measurements on the Seychelles, that
Anzeichen des Klimawandels **zeigen**]$_{non-e}$.
indications the climate change show.
'The geologists describe measurements on the Seychelles that show indications of the climate change.'

auf den Seychellen is an indicator for the event reading: it is a local adjunct (cf. table 1). The relative clause *die Anzeichen des Klimawandels zeigen* with *zeigen* as predicate is an indicator for the object reading: *zeigen* belongs to the class of "proving verbs" (cf. table 2).
Nevertheless, the nominalization does not (necessarily) remain sortally ambiguous at the sentence level. The human reader is perfectly able to interpret the nominalization as an event or as an object - at the latest when he considers a larger context window than one sentence. Obviously, there are cases where human readers are not able to disambiguate the sortal interpretation of the nominalization (cf. sentence 4). However, these seem to be cases where it is not relevant for the comprehension of the text.

(4) Die Schiffahrt profitiert von den aktuellen Messungen über
The navigation benefits from the current measurements over
Windgeschwindigkeit und Wellenhöhen.
wind speed and wave heights.
'The navigation benefits from the current measurements of wind speed and wave heights.'

On the other hand, there are examples that show that some contextual phenomena enforce a certain interpretation which leads to a misinterpretation of a sentence such as sentence 5.

(5) Da die Messungen vor den Fenstern der vom Lärm
Since the measurements in front of the windows the of the noise
Betroffenen vorgenommen wurden, hat ihnen die Gemeinde
persons concerned carried out were, has them the municipality
Schallschutzfenster angeboten.
soundproof windows offered.

'Since the measurements were carried out in front of the windows of the persons concerned of the noise, the municipality offered them soundproof windows.'

4.2 The Incremental Process of Sortal Specification

What determines the sortal interpretation at the sentence level is the underlying syntactic reading. Due to syntactic ambiguity we get most often more than one syntactic reading. Different syntactic readings may lead to different sortal interpretations of one and the same sentence.
The sortal interpretation depends on the syntactic reading insofar as the indicators of a given sentence may appear in different places of the according syntax tree. Since indicators of different sortal readings may be present in one and the same sentence, we assume that the interpretation process works strictly incrementally, i.e. the indicators "enter" the context used for interpretation one after the other. Depending on what context window is considered the sortal interpretation may vary; if the context window grows the sortal interpretation may change.
The interpretation that is accessible for a context larger than one sentence is the one at the sentence level. So, in order to come up with the sortal interpretation of the nominalization at the sentence level we start with the nominalization in the null context and walk up the syntax tree. The considered context grows step by step walking through the syntax tree.
The sortal interpretation of the nominalization is "defeasible" as long as there is a larger context that is relevant for the interpretation process; in its current context, the sortal interpretation is "indefeasible". As regards our concept of "defeasible" and "indefeasible" sortal interpretations, we lean on Alshawi and Crouch's concept of "believed" vs. "unbelieved" in their monotonic semantic interpretation (cf. [1]). The sortal interpretation of a nominalization is explicitly called into question until there is no larger context relevant for the interpretation process. That means, the sortal interpretation of a nominalization actually is a disjunction of all possible sortal types with the "contextually active" sortal type "underlined". Therefore, the interpretation process is a monotonic process nevertheless.
The core idea of this specification process is that the reading indicators that enter the context while it grows incrementally introduce constraints that can be applied to a nominalization in its current context. The specification process follows the algorithm given below:

- The "bare" *-ung*-nominalization (i.e. the nominalization in its null context) which, obviously, is sortally ambiguous gets the sortal type $<\texttt{e} \overset{+}{\cup} \texttt{non-e}>$[5].
- Then, all sibling nodes are considered: before a sibling node is added to the "active" context[6], it is checked whether it dominates an indicator.
- If so, the indicator introduces a constraint over the interpretation of the *-ung*-nominalization in its current context.
- The constraint is applied, and the sibling node is added to the context of the nominalization.
- The procedure is repeated until the sentence node is reached.

The Main Constraint and a Type Conversion Function Supposed:

- $\texttt{U} = \{$x, x is a *-ung*-nominalization$\}$
- $\texttt{m} \in \texttt{U}$
- $\texttt{ung-sort} = \{\text{e}, \text{s}^{res}, \text{non-e/s}^{res}\}$
- $\alpha, \beta \in \texttt{ung-sort}$

We define a constraint $\text{C}_{<\alpha\overset{+}{\cup}\beta,\alpha>}$ that has the following two properties:

1. $\text{C}_{<\alpha\overset{+}{\cup}\beta,\alpha>}(\text{m}_{\alpha\overset{+}{\cup}\beta}) = \text{m}_\alpha$
2. $\text{C}_{<\alpha\overset{+}{\cup}\beta,\alpha>}(\text{m}_\beta) = \text{m}_\alpha$

In order that the constraint be applicable to m_β we define a type conversion function τ:

- $\tau(m_\beta) = \text{m}_{\alpha\overset{+}{\cup}\beta}$

5 Four Syntactic Readings and their Corresponding Sortal Readings

In order to demonstrate how the specification algorithm works, we will analyze sentence (3): *Die Geologen beschreiben Messungen auf den Seychellen, die Anzeichen des Klimawandels zeigen.*

5.1 Four Syntactic Readings

Sentence (3) has at least four syntactic readings since there are two possible attachment points for the prepositional phrase *auf den Seychellen* and two for the relative clause *die Anzeichen des Klimawandels zeigen*. These four most obvious syntactic analyses are given below (our syntax trees exclusively reflect the dominance relations and we abstract away from linear surface order):

1. *Die Geologen* is the subject, *Messungen* is the direct object of *beschreiben*, the relative clause is attached to *den Seychellen*, and the emerging complex prepositional phrase is an adjunct of *beschreiben* (cf. figure 2).

[5] $<\texttt{e} \overset{+}{\cup} \texttt{non-e}>$ reads event or object.
[6] "Active" context is used in the sense of "active" edges in chart parsing.

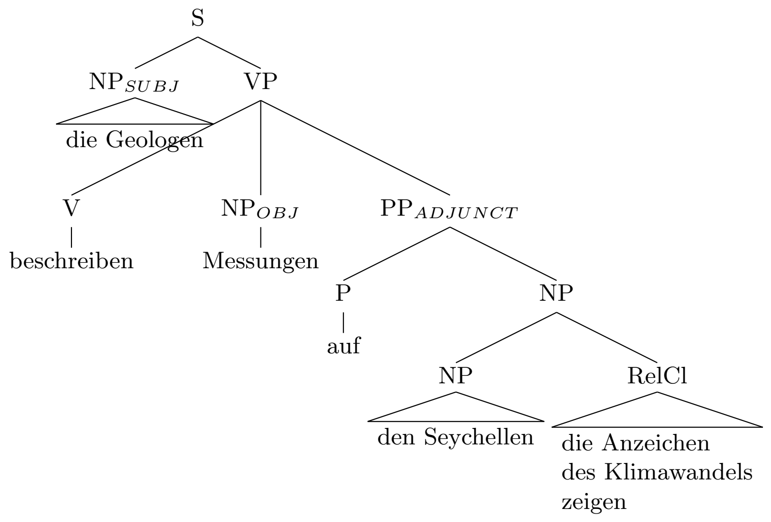


Fig. 2. Reading (1) of Sentence (3)

2. *Die Geologen* is the subject, *Messungen* is modified by the relative clause, and the resulting complex noun phrase is the direct object of *beschreiben.* The prepositional phrase is an adjunct of *beschreiben* (cf. figure 3).
3. *Messungen* is modified by the prepositional phrase. The emerging complex noun phrase is modified by the relative clause and constitutes the direct object of *beschreiben*; there is no verbal adjunct (cf. figure 4).
4. The relative clause is attached to *den Seychellen*, and the resulting complex prepositional phrase is attached to *Messung*. The emerging complex noun phrase is the direct object of *beschreiben*; there is no verbal adjunct in this syntactic analysis (cf. figure 5).

5.2 The Incremental Sortal Specification of Sentence (3)

In the following, we will show that depending on which syntactic reading we choose, we may end up with different sortal interpretations. To this end, we will analyze in detail reading (3) (cf. figure 4). In all other cases, we only present the result of the specification process.

The Reading Indicators and the Constraints they Introduce In sentence (3), we find two competing reading indicators (cf. 4.1) which introduce the following constraints[7]:

[7] Again, for the sake of convenience we do without `non-s`^{res}.

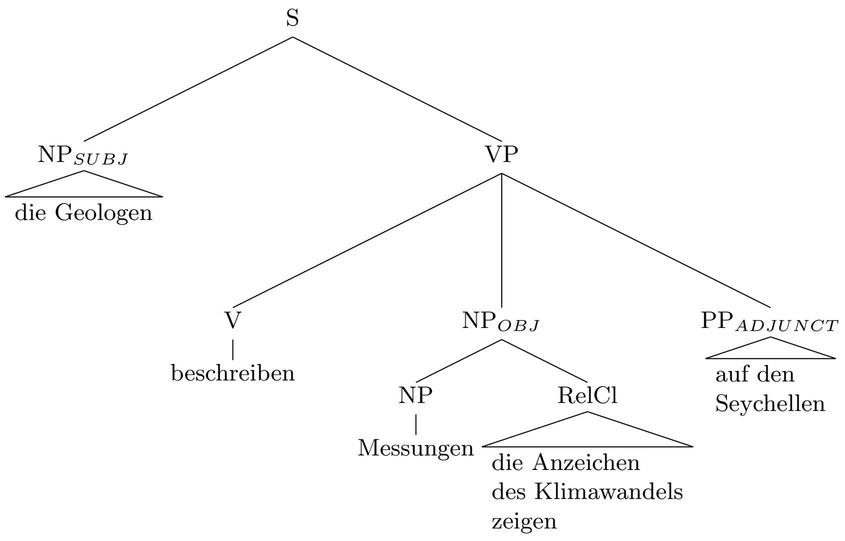


Fig. 3. Reading (2) of Sentence (3)

1. The local PP *auf den Seychellen* introduces a constraint that yields a linguistic object of the event-type <`e`>: $C_{<e \overset{+}{\cup} non-e,e>}$.
2. The relative clause with the predicate *zeigen* introduces a constraint that yields a linguistic object of the object-type <`non-e`>: $C_{<e \overset{+}{\cup} non-e,non-e>}$.

The Sortal Interpretation of Reading (3)

1. *Messungen* in the null context is considered; it is assigned the type $< e \overset{+}{\cup} non - e, e >$.
2. All sibling nodes of *Messungen* are considered: i.e. the prepositional phrase *auf den Seychellen*.
3. Does the PP-node dominate an indicator ?
   Yes: The indicator introduces a constraint that yields a linguistic object of the event-type: $C_{<e \overset{+}{\cup} non-e,e>}$.
4. The constraint is applied to *Messungen*, the active context grows, and *Messungen* in its "new" active context is of the type <`e`>: [*Messungen auf den Seychellen*]$_{<\texttt{e}>}$.
5. All sibling nodes of *Messungen auf den Seychellen* are considered: i.e. the relative clause *die Anzeichen des Klimawandels zeigen*.
6. Does the RelCl-node dominate an indicator ?
   Yes: The indicator introduces a constraint that yields a linguistic object of the object-type: $C_{<e \overset{+}{\cup} non-e,non-e>}$.

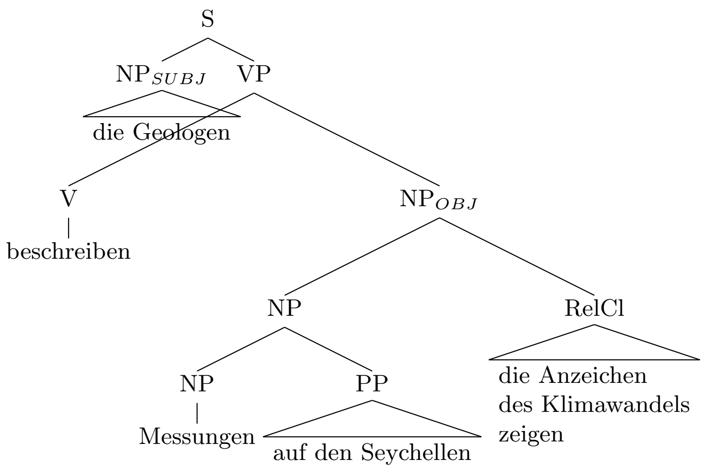


Fig. 4. Reading (3) of Sentence (3)

7. The constraint should be applied to *Messungen auf den Seychellen*, but this noun phrase is of the wrong type: it should be of the type $< e \overset{+}{\cup} non - e, e >$, but is of the type $< e >$.
8. The type conversion function τ is applied: $\tau(\mathrm{NP}_{<e>}) = \mathrm{NP}_{<e \overset{+}{\cup} non - e>}$.
9. Now, the constraint is applied to *Messungen auf den Seyhellen*, the active context grows, and *Messungen auf den Seychellen* in its "new" active context is of the type <`non-e`>: [*Messungen auf den Seychellen, die Anzeichen des Klimawandels zeigen*]$_{<\texttt{non-e}>}$.
10. The predicate *beschreiben* as well as the subject *die Geologen* do not introduce constraints into the context. We reach the sentence level and the interpretation process is finished.

⇒ If the underlying syntactic analysis is reading (3), the sortal interpretation is the obejct-interpretation.

The Sortal Interpretations of Readings (1), (2), and (4) Following the same algorithm, we end up with an ambiguous sortal interpretation of *Messungen* in case of reading (1), reading (2) leads to an event-interpretation, and reading (4) enforces an object-interpretation.

6 Conclusions and Future Work

We have shown that the sortal interpretation of *-ung*-nominalizations is highly context-dependent: the sentence context introduces indicators which can trigger a sortal reading. The interpretation is also dependent on syntactic ambiguity, as different syntactic readings of a sentence may give rise to different sortal interpretations.
We specified an algorithm for the incremental sortal specification of *-ung*-nominalizations which uses context-derived constraints in order to determine the sortal interpretation at the sentence level.

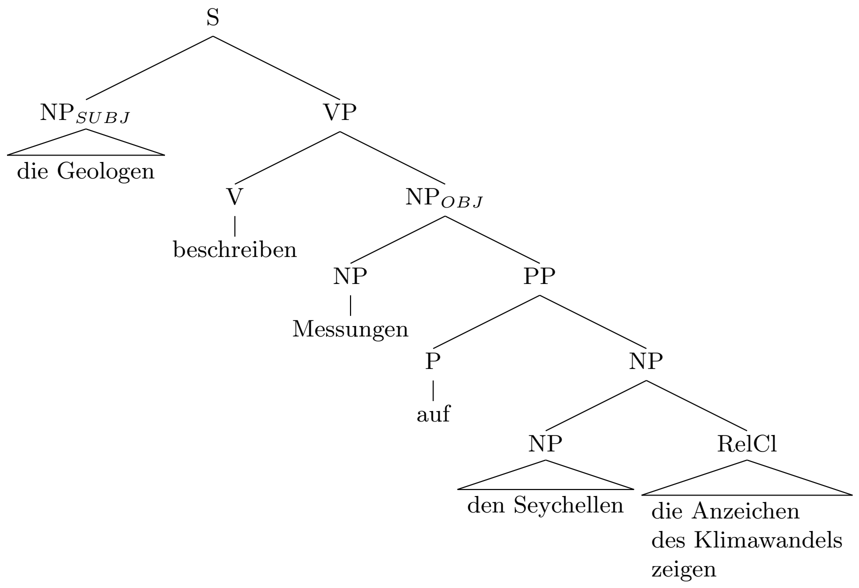


Fig. 5. Reading (4) of Sentence (3)

For an implementation in the framework of data extraction from corpus text, we will assess which syntax formalisms and parsing grammars provide adequate input for the specification process.
To be able to provide all possible syntactic readings and the pertaining sortal interpretations, we are developing an underspecified representation that should assemble all possible syntactic readings (cf. [14] for ideas that possibly can be adopted).

References

1. Alshawi, Hiyan, Crouch, Richard S.: Monotonic Semantic Interpretation. Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics (1992) 32–38
2. Bach, Emmon: The Algebra of Events. Linguistics and Philosophy **9(1)** (1986) 5–16
3. Ehrich, Veronika, Rapp, Irene: Sortale Bedeutung und Argumentstruktur: ung-Nominalisierungen im Deutschen. Zeitschrift für Sprachwissenschaft **19(2)** (2000) 245–303
4. Esau, Helmut: Some facts about German nominalization. Neophilologus **55(1)** (1971) 150–156
5. von Heusinger, Klaus: The Interface of Lexical Semantics and Conceptual Structure: Deverbal and Denominal Nominalizations. In: Nominalisierung. Zimmermann, I. and Lang, E. (eds.). Zentrum für Allgemeine Sprachwissenschaft, Berlin (2002) 109–124
6. Knobloch, Clemens: Zwischen Satz-Nominalisierung und Nennderivation: *-ung*-Nomina im Deutschen. Sprachwissenschaft **27(3)** (2002) 333–362

7. Lakoff, George: Linguistics and Natural Logic. In: Approaches to Natural Language. Davidson, Donald/Harman, Gilbert (eds.), Reidel, Dordrecht/Boston (1972) 545–665
8. McCawley, James D.: Lexical Insertion in a Grammar without Deep Structure. Papers from the 4th Regional Meeting of the Chicago Linguistic Society (1968) 71–80
9. Moens, Marc, Steedman, Mark: Temporal Ontology and Temporal Reference. Computational Linguistics **14(2)** (1988) 15–28
10. Osswald, Rainer: On Result Nominalization in German. Proceedings of Sinn und Bedeutung 9 (2005) 256–270
11. Reckman, Hilke, Cremers, Crit: Deep parsing semantic interpretation of nominalizations and their expressed and unexpressed arguments. Leiden Working Papers in Linguistics **4(1)** (2007) 40–55
12. Scheffler, Tatjana: Nominalization in German. Unpublished Manuscript, University of Pennsylvania (2005)
13. Shin, Soo-Song: On the event structure of *-ung* nominals in German. Linguistics **39(2)** (2001) 297–319
14. Spranger, Kristina: Combining Deterministic Processing with Ambiguity-Awareness – The Case of Quantifying Noun Groups in German. PhD Thesis, University of Stuttgart (2006)
15. Vendler, Zeno: Facts and Events. In: Linguistics in Philosophy, Cornell University Press, Ithaca (1967) 122-146

Constraint-based Analysis of Discourse Structure

Antoine Widlöcher and Patrice Enjalbert

Laboratoire GREYC, CNRS UMR 6072, Université de Caen, France
{Antoine.Widlocher,Patrice.Enjalbert}@info.unicaen.fr

Abstract. Recent works in the NLP community show an increasing interest for the analysis of discourse structure, leading to a variety of models and applications. We claim that a constraint-based approach allows a high level, abstract, description of various discourse structures, together with the design of an operative framework for generic analysis methods. The CDML formalism (Constraint-based Discourse Modelling Language) presented in this paper is devoted to these tasks.

1 Introduction

The notion of constraint has proved very useful in different areas of NLP, both as a unifying theoretical framework, and as a common ground for various constraint-based formalisms such as GPSG, HPSG, or Blache's Property Grammars [1]. In the present paper we want to address the question of application of the constraint-based methodology at discourse level. Indeed, recent works in the NLP community show an increasing interest for the analysis of discourse structure. Motivations may be purely linguistic, NLP tools being useful for corpus study and experimentation of linguistic theories. But there is also a wide range of applications where automated discourse analysis can help: thematic indexation, especially at inner document level, automated summarisation, document browsing, etc.

The approaches of discourse structure analysis are many. Several of them are oriented towards the description of so called "discourse relations" between sentences or clauses, of which they attempt to give organised inventories, together with criteria allowing to establish those links: RST [2], SDRT [3], LDM [4], D-LTAG [5] are such well established theories. Others consider texts at a coarser grain and focus on its segmentation in "homogeneous" blocks: typical examples are provided by the so-called text-tiling technique, following Hearst [6] and Teufels' argumentative zoning [7]. In both case one tries to capture the overall organisation of a text. Another major, ubiquitous question concerns anaphora resolution, i.e. coreferential links between (mostly nominal) expressions.

At present each approach develops its own description format and (possibly) analysis procedures. At first sight this may look unavoidable, because precisely of the diversity of linguistic phenomena under consideration. However, we cannot either forget the deep interactions between them and the fact that, at the end, the different kinds of structures all contribute to the reader's or hearer's perception of the coherence of a text, and finally to its understanding. This remark immediately raises the problem of defining some common format where several (if not all) kinds of structures could be described and processed. This format would be a useful device for comparing various approaches and taking benefit of the complementary analyses they each offer.

In order to make some steps in that direction, we present a formalism named CDML [8], which can be described as a "fully Constraint-based Discourse structure Modelling Language". Indeed, we claim that the notion of *constraints*, already widely used at sentential level, provides an adequate general framework for discourse structure description and analysis. The main reason is that it provides just the right level of abstraction to cope with different kinds of structures, and hence to combine different analyses, in a unified way. Moreover, constraint-based formalisms are highly declarative - which is good for description - and altogether allows efficient computation methods - good for processing.

The paper is organised as follows. In section 2, we go more deeply in the specifications of what can be a generic approach of discourse analysis. Section 3 outlines the syntax and intuitive semantics of CDML, examplified in section 4, while section 5 gives the basis of a formal semantics. In section 6 we present the current implementation and first experimental results, and finally we conclude with further research perspectives.

2 Elements for a generic approach

Let us consider a bit more precisely the what "discourse structure" is, focusing on approaches leading - or potentially leading - to automated processing.

A major notion is that of *discourse relation*. This can be understood in a rather semantic way as synonymous of "rhetoric relations": narration, explanation, elaboration... Rhetorical Structure Theory (RST) [2] and Segmented Discourse Representation Theory (SDRT) [3] are important theories in this area. Others like the Linguistic Discourse Model (LDM) [4] or D-LTAG [5] on contrary tend to extend syntactic relations at a supra-sentential level. In both cases, authors take for granted that propositions (or other phrasal units) are the basic constituents of discourse which have to be related. Their approach is "ascending": starting form basic units, one tries to establish a "discourse tree" (or "graph"). Connectives, cue phrases, as well as syntactic structure and semantic information or even world knowledge can be invoked as responsible for the various links.

Other approaches centrally focus on *text segmentation* in blocks of a certain amplitude, covering several sentences. In *thematic segmentation* the problem is to part the text into thematically homogeneous segments, and (possibly) to give a representation of what these segments are about. Following Hearst [6], such works are often based on the notion of lexical cohesion [9] and use statistical, Information Retrieval inspired, methods to detect break points between lexically homogeneous segments. However certain inventories of cue-phrases or other specific patterns (such as frame introducers, see below) can also help. Another significant approach is Teufel's *argumentative zoning* [7], where different text zones correspond to different moves in the rhetorical intentions of authors. In both cases, the segments are assumed to partition the whole text, while in Charolles' *discourse framing theory* [10], only certain specific segments, introduced by by detached adverbials, are identified (see section 4.1 below). All these approaches are rather "top-down", using "surface" indices or global effects rather than progressive composition of elementary units.

Another kind of structures consists in various possible *chaining between distant expressions*. A major example is the notion of co-reference chain constituted by a series of expressions that refer to the same object or event in the discourse domain ("anaphora resolution"). A variety of criteria operate, using morphological, lexical, syntactic, semantic information, and possibly also world knowledge. Other quite different kinds of chaining exist such as lexical (or semantic) chaining, which relate series of terms which are *semantically bound* together.

It is readily seen that these different kinds of structures are irreducible to one another. Nevertheless they are also deeply linked together, since they are different forms of discourse organisation that contribute to the reader's understanding of a same text, to produce its cohesion and coherence. The notion of "texture" (Halliday) nicely evokes this intertwining. Moreover a specific analysis often needs to take in consideration other dimensions of discourse organisation. For instance referential and semantic chaining contribute to thematic segmentation; discourse framing plays an important role in theme identification; argumentative relations often link whole segments rather than mere propositions so that text segmentation and rhetoric analysis can be co-dependant; and so on.

At present, each approach develops its own description format and (possibly) analysis procedures. However, if no "universal discourse grammar" is probably to be expected, it looks quite relevant to design formal models that can mix several kinds of analysis. In order to take in account these dependencies, first. But also because such unifying models could be beneficial, at a theoretical level, to compare different approaches and finally to produce a better understanding of "discourse structure". Let us analyse more closely the requirements for such a generic approach, that lead us to propose a constraint-based formalism.

Segments and relations. We observed that some approaches are rather segment-oriented and other relation-oriented, but both are relevant of discourse description. Moreover, segments themselves should not be considered as "one block" units, but subject to hierarchical decomposition.

Granularity. Some approaches (RST, LDM...) focus on sentential or phrasal text units, but others consider a coarser grain like paragraphs or *ad-hoc* segments. Moreover the same discursive phenomena may appear at different levels. The formalism should allow changes or underspecification of granularity.

Clues. As illustrated above, the different theories make use of a great diversity of methods in order to detect discourse structures including explicit connectors or characteristic cue phrases, as well as syntactic or positional criteria. One of the most challenging problem for a general approach of discourse structure analysis is to deal with this variety of linguistic clues.

Sequentiality and linearity. A sequential (word by word) analysis of text is often not relevant; relations between distant, non consecutive, expressions must be considered, as it is already the fact at sentential level. Moreover, the linear order itself may even be irrelevant, and some more global (or synthetic) mechanisms are also very important, for example in lexical cohesion phenomena.

"Soft" structures and computing. Another major feature of discourse structure is its *soft* nature. For example, two readers will not in general fully agree on the delimitation of thematic segments, and indeed transitions are often progressive. Moreover, a given discourse structure is hardly the unequivocal product of a simple, clearly identifiable, criterion, but rather of a set of indices, some being concordant and other discordant, producing *in fine* a dominant impression.

From this enumeration (which should certainly be refined) the immediate conclusion draws that no "specific algorithm" can take in charge all aspects of discourse structure. We need some more abstract, high level, and "soft" specification tool. This tool could rely on specific algorithms for specific limited tasks, but cannot reduce to one. Without being any "magic wand" we think that the constraint paradigm is a good candidate to fulfill those requirements. CDML, to be presented now, is an attempt in this direction.

3 CDML syntax and intended semantics

3.1 Discourse structure and its representation

In CDML the description of discourse structure relies on three basic notions:

Discourses Units (DU) correspond to well delimited text zones, at arbitrary levels of granularity: for example thematic segments or "à la Teufel" argumentative zones as well as simple clauses.

Discourse Relations (DR) are relations between DU's. They can be defined by a set of constraints on the constitutive DU's. Examples are rhetorical relations, as in RST or SDRT, or hierarchical constituency relations.

Discourses Schemes (DS) are high-level patterns defined by constraints on both DU's and DR's. A typical example would be an enumerative structure consisting of an initialisation and a sequence of items.

The term "discourse element" is a generic for DU, DR and DS. Different collections of units, relations and schemes have to be considered according to the linguistic theories (or practical applications) in view. The aim of CDML grammars is to provide a high-level, synthetic, framework allowing their formal definition.

A CDML grammar takes as *input* a discourse which may have been annotated from different points of view: morphological tagging, syntactic analysis... The grammar expresses constraints on available elements and annotations. As *output* it provides a new discourse representation in which described discourse elements are annotated.

Before entering the description of CDML grammars, we must say how discourse elements are represented. The notion of discourse schemes being not mature enough, from now on, we will only consider DU's and DR's. The delicate point concerns the representation of text segments. A first remark is that a text is not necessarily seen as a sequence of *words*. Often, we want to abstract from this "first layer", and consider it as a sequence of sentences, for example, or even at higher grain, as a succession of "parts" if we focus on its overall organisation. We can even want to mix different levels, considering for example certain phrasal and certain sentential units. We therefore introduced the notion of *extended token* (or simply *token* for short) to refer to any kind of contiguous linguistic expression, considered as a whole, hiding its constituents. The user can define what will be considered as "tokens" for a given analysis: it is part of the notion of "analysis perspective" described below.

The second important notion is that of *segment boundaries*. Boundaries of a segment are "brackets" surrounding that segment, in order to mark its beginning and end. Consider for example the notion of *frame introducer* (cf. section 4.1), defined as a prepositional phrase at initial position in a sentence. The simplest way to express this position criterion is to say that the left boundary of the PP coincides with the left boundary of a sentence.

Discourse input is then seen as a sequence of *text objects* (TO) consisting in (extended) tokens or boundaries. TO's corresponds to the atomic "events" thrown by textual flow. Their size may vary as well as their linguistic status: morpho-syntactic units, syntagmatic elements, propositions... They can also overlap or be embedded. They represent all available information on discourse content and may arise from any preliminary analyses.

We can associate TO's with pieces of information of any kind (syntactic nature or function, semantic values...) by means of (recursive) *feature structures* (FS). We use the following sequential notation for FS's:

```
    {a:{b:X, c:{d:Y, e:Z}}}
```

where a, b... are feature names and X, Y feature values.

A DU is then represented as a sequence of contiguous TO's, marked by boundaries denoting its opening and ending positions in text. With any DU a FS is associated, representing its "semantics", the "visible" part possibly involved in constraints for the definition of structures of higher level. DR's are undirected binary relations between DU's. They also are associated with a FS.

3.2 CDML grammars

Grammar. A CDML grammar consists in a set of rules. Each rule is devoted to the description of a particular element of discourse. Every rule is typed according to the kind of element it is concerned with.

Rule. The general form of a rule is:

```
RuleType [RuleName] fs [Dependencies]:
    [Perspective]
    Constraints
```

`RuleType` may be `Unit` or `Relation`; `RuleName` is the name of the rule. `Perspective` defines a view on the discourse structure, allowing some operations of selection and abstraction; `Constraints` is a list of constraint calls, belonging to a set of available ones according to the rule type (*i.e.* to the targeted discourse element). `fs` is a feature structure, representing information to be associated with the detected element. If the rule is *satisfied*, a discourse element is generated, which can, in turn, be considered by other rules in order to apply constraints of a higher order. Dependency relations between rules can be specified if necessary.

Constraint call. A constraint call has the following form:

```
    constraint-name(arg-1:val-1, arg-2:val-2...)
```

where `arg-1`, `arg-2`... are argument names, and `val-1`, `val-2`... are parameters. The order is meaningless and the values are implicitly typed (as boolean, integer, string, FS). When a constraint possesses a FS argument `p`, the intended meaning is generally to select one or several objects matching `p`. For example, the rule:

```
Unit void:
    start(pattern:{type:"sentence"})
```

specifies a discourse unit beginning with an element whose FS unifies with the indicated pattern, *i.e.* an element typed as "sentence".

We distinguish two kinds of unifications: *standard*, noted $\sim$, and *strong*, noted $\approx$ (pattern matching). Standard unification only imposes compatibility of two FS's, while the strong one considers the first FS as a model for the other. Hence, $\{a : b\} \sim \{c : d\}$ but $\{a : b\} \not\approx \{c : d\}$. The comparison between a "pattern" argument of a constraint and the FS of a TO works with strong unification. Patterns can include variables. Variable sharing is authorised. It allows transmission of information, both for compatibility checking (between variables shared by several constraints in the same rule) and information synthesis (in the output FS of the rule). For example:

```
Unit segment {firstSentenceLength:$a}:
    start(pattern:{type:"sentence", size:$a})
    end(pattern:{type:"sentence", size:$a})
```

selects text segments beginning and ending by sentences of the same "size", and return this value in the `$a` variable.

Meta-constraint. We distinguish *simple* and *meta constraints.* A simple constraint expresses a first order relation between its arguments. A meta-constraint expresses a relation between an object o and a set S. Intuitively, S will be a set of already selected candidates, and o will be any of them having a "special position" in S. For example the rule:

```
Unit frame {type:"frame", sub-type:"temporal"}:
    ... Constraints ...
    longest()
```

only keeps units with maximal length among all those satisfying `Constraints`. Meta-constraints can be seen as the CDML analogue of optimisation constraints in numerical constraint systems: in both cases, the problem is to compare different candidates and to select one, or several, which are in some sense "prefered".

Relation. A discourse relation (DR) is defined by constraints on the two DU's it relates, called its *targets*. The following grammar captures relations between elements known as subject and verb, ignoring order and distance between them:

```
Relation {type:"subject-verb"}:
    target(pattern:{type:"subject"})
    target(pattern:{type:"verb"})
```

Comparator. Constraints may use *comparators* as parameters in order to specify complex relations between text objects. For example, the *homogeneity* constraint is satisfied if all conditions defined by its comparator are satisfied. The following grammar, which will be further explained in section 4.1, defines units called "discourse frames", in which all verbs must have the same tense.

```
Unit frame {type:"frame", sub-type:"temporal"}:
    ... homogeneity(comparator:scope) ...

Comparator scope ({type:"verb"} as $v1, {type:"verb"} as $v2):
    $v1/tense = $v2/tense
```

Variables (`$v1, $v2`) represent TO's matching `{type:"verb"}`. Expressions like `$v1/tense` express pathes within FS's.

Perspective. Every rule may include what we call its *analysis perspective*, defined by a set of directives which specify a particular view on textual data. The *Maximal Relevant Unit* (MRU) directive restricts the search space in such a way that all objects considered by a rule should belong to the same MRU. For example, since the subject-verb relation is inner to a sentence, we can write:

```
Relation {type: "subject-verb"}:
@mru:['sentence']
    target(pattern:{type:"subject"})
    target(pattern:{type:"verb"})
```

It is also possible to filter some elements of the textual flow, by ignoring unwanted elements or by listing exhaustively accepted ones. An illustration is provided in section 4.2. Finally, we can transform a discourse unit into a single token. Its constituents become then invisible. It is the case of subordinate clauses in the following example, so that only the tense of the verb of the main clauses is captured.

```
Unit {type: "present-sentence"}:
@tokens:['subordinate-clause']
    contains(pattern:{type:"verb", tense:"present"})
```

4 Examples of CDML grammars

4.1 Temporal discourse frames.

The discourse framing theory [10] describes a specific mode of discourse organisation, identifying textual segments called *discourse frames*. These frames are homogeneous according to a semantic criterion given in a detached, sentence-initial expression called *discourse frame introducer*. We focus here on temporal frames. In the following example[1], the phrase *From 1965 to 1985* is an introducer which constrains the temporal localisation in a large subsequent text span.

From 1965 to 1985 the number of high-school students has increased by 70%, but at different rythms and intensities depending on academies and departments. Lower in South-West and Massif Central, moderate in Brittany and Paris, the rise has been considerable in Mid-West and Alsace. [...] Also occurs the schooling duration increase which was more important in departments where, in the middle of the 60's, study continuation after primary school was far from being systematic. [...]

We may assume that temporal expressions have been detected and given a semantic analysis, by means of local grammars. The most challenging problem is then to detect their scope *i.e.* the end of the frames. Different clues can be invoked [11], of which three will be considered here: semantical coherence of temporal expressions within the frame, tense homogeneity, and structural constraints, like the fact that frames contains complete sentences and cannot span over several paragraphs. This is expressed by the following CDML grammar:

```
Unit frame {type:"frame", sub-type:"temporal"}:
    start(pattern:{type : "introducer"})
    end(pattern:{type : "sentence"})
    not absolutePresence(pattern:{type:"introducer"}, amount:2)
    homogeneity(comparator : scope)
    longest()

Comparator scope ({type : "verb"} as $v1, {type : "verb"} as $v2):
    $v1/tense = $v2/tense

Comparator scope ({type : "introducer"} as $i, {type : "temporal"} as $t):
    (($i/start >= $t/start) and ($i/start <= $t/end))
  or
    (($i/end >= $t/start) and ($i/end <= $t/end))
```

We are looking for a textual unit starting with a discourse object identified as an introducer and ending with a sentence. It cannot contain two introducers (the second would open a new frame), and is homogeneous w.r.t. two criterions, expressed by the two *signatures* of the "scope" comparator: 1) two verbs must have the same tense and 2) every temporal expression must denote a period that overlap with the period of the introducer. Finally, among all admissible candidates, we keep the longest segment. Textual units satisfying these constraints are marked with the type "frame" and sub-type "temporal".

[1] Excerpt from: Hérin, R. and Rouault, R., *Atlas de la france scolaire de la maternelle au lycée.* Our translation.

4.2 Contrast and structural parallelism.

The following example focuses on constrast effect resulting from structural parallelism. Figure 1[2] illustrates such a configuration: similar (or parallel) structures introducing three textual segments result in contrastive relations between these segments. We can characterise these introducing structures as composed of "non consecutive n-grams", whose components belong to some set of "relevant words" (2-grams of verbs in our example, such as ("résister", "être") which appears three times). The repetition of such n-grams (2, 4, 6 on our example) results in relations of parallelism between them. Textual segments such as 1, 3 and 5, introduced by such parallel n-grams become in consequence important components of discourse structure, supporting a contrastive relation whose effect constitues a well known rhetorical device. The following grammar detects such a structure.

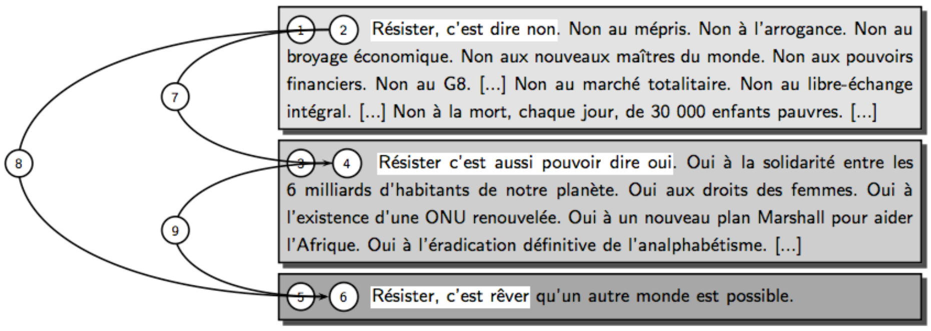

1
2
Résister, c'est dire non. Non au mépris. Non à l'arrogance. Non au broyage économique. Non aux nouveaux maîtres du monde. Non aux pouvoirs financiers. Non au G8. [...] Non au marché totalitaire. Non au libre-échange intégral. [...] Non à la mort, chaque jour, de 30 000 enfants pauvres. [...]
3
4
Résister c'est aussi pouvoir dire oui. Oui à la solidarité entre les 6 milliards d'habitants de notre planète. Oui aux droits des femmes. Oui à l'existence d'une ONU renouvelée. Oui à un nouveau plan Marshall pour aider l'Afrique. Oui à l'éradication définitive de l'analphabétisme. [...]
5
6
Résister, c'est rêver qu'un autre monde est possible.
7
8
9

Fig. 1. Contrast and structural parallelism

```
Unit nGram {type:"nGram", length:2, elements:{lemma-1:$l1, lemma-2:$l2}}:
@mru:['sentence']
@accept:['relevantWord']
    consecution(patterns:[{type:"relevantWord",lemma:$l1}, {type:"relevantWord",lemma:$l2}])

Relation parallelism {type:"parallelism"} requires nGram:
@mru:['document']
    leftTarget(pattern:{type:"nGram",elements:{lemma-1:$l1, lemma-2:$l2}})
    rightTarget(pattern:{type:"nGram",elements:{lemma-1:$l1, lemma-2:$l2}})

Unit structuringNGram {type:"structuringNGram"} requires parallelism, nGram:
@mru:['document']
    is(pattern:{type:"nGram"})
    inRelation(pattern:{type:"parallelism"})

Unit segment {type: "segment"} requires structuringNGram:
    start(pattern: {type:"structuringNGram"})
    end(pattern: {type:"sentence"})
    not absolutePresence(pattern: {type:"structuringNGram"}, amount:2)
    longest()
```

[2] Excerpt from *Le Monde Diplomatique.*

"Plain" words (verbs, nouns...) are supposed to be already annotated as `relevantWords` and represented by a FS providing their lemma. In order to describe (non consecutive) n-grams, we only consider these relevant words (`@accept:['relevantWord']`). Only sequences of such words present in the same sentence are taken into account (`@mru:['sentence']`). FS's associated with n-grams provide information about the lemmatized forms of their components. The parallelism relation whose distance can be maximal (`@mru:['document']`) links n-grams whose components share the same lemma (as required by unification). All n-grams occurring in a parallelism relation are considered as `structuringNGrams`. Segments are defined as units composed of full sentences, introduced by a structuring element, and which do not contain any other structuring element.

5 Formal semantics of CDML grammars

The semantics of CDML grammars is defined in an incremental, constructive, way. At a given stage, we have a set of discourse elements (DU's and DR's), whose projection on text is given as a sequence of TO's. This sequence contains all necessary information on existing elements. We assume an initial state of discourse, represented by an initial TO sequence. Each rule will be applied in turn, according to some specified order. It looks in the current discourse state for objects that can be gathered and related, according to its set of constraints, in order to identify new elements (units or relations). If it succeeds, these elements are created (and materialised by new TO's), resulting in a new discourse state.

In fact, a bit more precisely, each rule, before applying its constraints, builds its own view of the discourse state. This is the role of the *Perspective* part of the rules (cf. section 3.2). In particular the `@mru` directives may cut the text in successive blocks, to be treated one by one in turn. However, for sake of simplicity, we will omit this question in order to focus on constraints themselves.

Definitions and notations: A *text object* (TO) is a *token* or a *boundary*. Each object o has an associate feature structure (FS), noted $fs(o)$. $\mathbb{F}$ denotes the set of feature sets. TO's can be gathered in sequences (TOS). The resulting order relation is noted $\leq$. Boundaries are organised as couples of *opening* and *ending* ones, in this order, with the same FS. A text T is a particular TOS which represents a state of discourse.

5.1 Constraints: units

Definition: A *candidate unit* u is a subsequence of T. It can also be seen as an interval in T, identified by its lower and upper bounds, *i.e.* its first and last TO's. Let $U(T)$ be the set of unit candidate for the text T. Hence:

$$U(T) = \{[a,b] \mid a,b \in T, a \leq b\}$$

where $[a,b] = \{c \mid a \leq c \leq b\}$. We note $s(u)$ and $e(u)$ resp. the first and last element, so that, for instance, $fs(s(u))$ will denote the FS associated with the beginning TO of u.

Remember that a CDML *constraint call* has the form:

```
    constraint-name(arg-1:val-1, arg-2:val-2...)
```

We consider first the case of simple constraints. With each constraint-name a *constraint* is associated:

$$Const(unit\colon u, arg_1\colon val_1, arg_2\colon val_2...)$$

$Const$ is a computable relation relating arguments of the appropriate types. We keep the "argument naming" convention. Observe the introduction of an extra argument u of type *unit*. This argument is implicit in the syntax and made explicit in the semantics.

Let us consider some examples of constraint definitions. The $Start$ constraint checks that a unit candidate u begins with a TO matching (by strong unification) a given pattern $f \in \mathbb{F}$ according to the following condition :

$$Start(unit\colon u, pattern\colon f) \equiv (fs(s(u)) \approx f)$$

The $AbsolutePresence$ constraint (AP for short) checks that the number of elements of a unit u matching a given pattern f is greater or equal to some given integer q:

$$\text{let } S = \{i \mid s(u) \leq i \leq e(u), fs(i) \approx f\}$$
$$AP(unit\colon u, pattern\colon f, amount : q) \equiv |S| \geq q$$

where $|S|$ denotes the cardinality of S.

Let us turn now toward *meta*-constraints. Their intended semantics is to select one (or more) units in a set U. Hence, the semantics of a meta constraint call:

```
    meta-constraint-name(arg-1:val-1, arg-2:val-2...)
```

is a constraint with the following signature:

$$MConst(unitSet\colon U, unit\colon u, arg_1\colon val_1, arg_2\colon val_2...)$$

For example, the *Longest* constraint is a meta constraint that, in a set of candidates U, only keeps the longest for inclusion order.

$$Longest(unitSet\colon U, unit\colon u) \equiv \big|\{v \mid v \in U, u \subsetneq v\}\big| = 0$$

Consider now the *set* of constraint calls defining a (unit) rule. For sake of simplicity, we assume first that it contains at most one meta constraint μ and that no variable appears in the parameters of constraint calls. By the above process, we get a set of constraints:

$$C = \{c_1(unit\colon u, \overline{p_1}), ..., c_n(unit\colon u, \overline{p_n}), \mu(unitSet\colon U, unit\colon u, \overline{p_\mu})\}$$

where $\mu(unitSet\colon U, unit\colon u, \overline{p}) \equiv True$ if there is no meta constraint at all.

A *solution of* C *for a text* T is the set S defined by:

$$S_0 = \{u \mid u \in U(T), \forall i \in [1,n], c_i(unit\colon u, \overline{p_i}) \text{ is satisfied}\}$$
$$S = \{u \mid u \in S_0, \mu(unitSet\colon S_0, unit\colon u, \overline{p_\mu}) \text{ is satisfied}\}$$

If there is more than one meta constraint, we separate the set of constraint calls in successive parts consisting of 0 or more simple constraints, followed by one meta constraint. We apply the previous process to the first part; then to the second, replacing $U(T)$ by S; and so on: the set of solution at one stage is the range of units considered for the following one.

In the general case, variables appear in the parameters. We assume then that the associated constraints compute a *binding* σ for which satisfiability is ensured. Writing $(t\ \sigma)$ as usual for the instance of a term t by σ, the definition of a solution becomes:

$$S_0 = \{(u, \sigma) \mid u \in U(T), \forall i \in [1, n], c_i(unit\colon u, (\overline{p_i}\ \sigma)) \text{ is satisfied})$$
$$S = \{(u, \sigma) \mid u \in S_0, \mu(unitSet\colon S_0, unit\colon u, (\overline{p_\mu}\ \sigma)) \text{ is satisfied}\}$$

with σ being the empty binding in the no-variable case.

5.2 Constraints: relations

Definitions: A *candidate relation* r links two discourse units. We note $l(r)$ and $r(r)$ resp. the left and right *targets* of r. These units are two TOS's (text object sequences), that is to say two intervals in a text T. Let a_1, b_1 (and a_2, b_2) be the opening and ending boudaries of the first unit (resp. the second unit). Let $R(T)$ be the set of relation candidates for the text T:

$$R(T) = \{([a_1, b_1], [a_2, b_2]) \mid a_1, a_2, b_1, b_2 \in T, a_1 \leq b1 \leq a_2 \leq b_2\}$$

Each constraint-name is associated with a *constraint Const*:

$$Const(relation\colon r, arg_1\colon val_1, arg_2\colon val_2 ...)$$

where r ranges in $R(T)$. Let us consider an example of constraint definition. The $Target$ constraint checks that a relation candidate r links at last one unit whose FS matches a given pattern $f \in \mathbb{F}$. Knowing that a unit's FS is in practice supported by both its boundaries (cf. *infra*), we get:

$$Target(relation\colon r, pattern\colon f) \equiv (fs(s(l(r))) \approx f) \vee (fs(s(r(r))) \approx f)$$

Meta-constraints on relations may also be defined. Their intended semantics is to select one (or more) relation in a set of relations. The semantics of a relation constraints *set* is defined in the same way as in the unit case.

5.3 Rules

Consider a rule (without *Perspective* part):

```
RuleType RuleName fs:
    Constraints
```

Its semantics is defined as the transformation of a discourse state, represented by a text T, into a new one, T', in which new elements of the corresponding type (unit or relation) have been added.

Let S be the solution set of the `Constraints`. S consists in a set of couples (u, σ) (resp. (r, σ)) of a unit (resp. relation) candidate u (resp. r) and a variable binding σ. For each element of S, add to T a discourse element based on u (or r) with feature structure obtained by instanciating fs with σ, namely: $(fs\ \sigma)$. In the unit case, this is done by inserting pair (i, j) of opening and ending boundaries around u in T, $(fs\ \sigma)$ being their common feature structure. T' is the resulting sequence of TO's.

5.4 Grammars

Remember again that some rule may need the application of other ones as prerequesite:

```
    RuleType RuleName fs requires rule2, rule3:
```

Assuming that no loop is created in the *requires* relation, some order compatible with this relation can be defined on the set of rules. For example we can take the (only) order which is also compatible with the writing order of the rules in the grammar. Given an input text T, applying each rule in turn, in this order, produces a new text T'. The so defined function is the semantics of the grammar.

6 Implementation and results

The LinguaStream platform. A first implementation of a a CDML analyzer comes as a component for the LinguaStream platform[3] and takes advantage of its principles [12]. This generic NLP platform, allows complex processing streams to be designed, by means of successive components of various types and levels: part-of-speech, syntax, semantics... TO's on which CDML expresses constraints can be any objects produced by such aforegoing components.

Constraint satisfaction and optimization. Working at discourse level makes it necessary to take important complexity problems into consideration. Indeed, search space initialization and parsing may result in time and memory consuming operations. Different elements of our approach intend to address this problem. First of all, the MRU mechanism presented above makes it possible to optimize the in-memory discourse representation sent to the constraint solver. Other perspective directives (`accept`, `reject` and `tokens`) reduce this representation to really relevant objects for a given rule. In addition, grammar rules are organized in a dependance tree which makes it possible for independant rules to work on a same discourse representation.

Furthermore, we saw that constraint satisfaction consists in search space filtering, which can be costly. However, we can use some strategies in order to reduce its initial state. For example, in the case of discourse units, insofar as the search space consists in a set of TOS, a naive (but sometimes unavoidable) generation of its initial state results in $(n(n + 1)/2)$ candidates, where n is the

[3] http://www.linguastream.org.

number of TO's. In order to be more efficient, the system looks for a special, strongly selective, constraint (such as *Start* or *Target*) and applies it first, in order to generate the search space to which other constraints are applied next.

Results. The *meta*-model status of the proposed formalism makes it quite difficult to evaluate from a quantitative point of view. However, *specific* models, expressed using CDML, may. For example, we proceeded to an evaluation of the discourse framing analyzer presented above [13]. Results obtained using CDML are very similar to those obtained with a dedicated *ad hoc* software component. Time needed for scope analysis is about one minute, for a corpus of 65,000 words, on a standard workstation. Moreover, the linguistic modelisation time is significantly improved thanks to the declarative paradigm of CDML grammars.

7 Discussion and perspectives

It should be clear now what the main distinctive feature of our model is. Namely that, while current approaches focus on a specific vision of "what discourse structure is" with strong theoretical *a priori's* (or on some specific phenomena such as anaphora, discourses frames, etc.), our main objective was to find a unified framework allowing to describe - and process - a wide range of discursive structures. Our hypothesis is that a constraint-based approach is fully adapted to this goal and the CDML language was designed in this spirit.

We think that our first experiments (as illustrated by the few examples given in the paper) give credit to this hypothesis. They clearly show that it is possible in CDML to combine the detection of both segments and relations, exploiting a variety of indices, at different levels of granularity, abstracting from sequentiality and linearity of text when necessary. All of these were important requirements we assigned to our undertaking, as sketched out in section 2. Moreover, while one could have feared or expected to pay these "good news" relative to expressivity by "bad news" on the efficiency side, it appears that the system remains quite manageable, at least for research purposes, in order to perform experiments, projecting discourse models on corpora.

However these first applications, besides being still modest in size and number, concern a certain type of discourse structures, which we can characterise in a few words: they favour structures at a rather *high or medium level of granularity*, which can be discovered in a *top-down strategy*, exploiting combinations of scattered *linguistic markers*. This contrast on one side with more global phenomena as lexical cohesion (*text-tiling* and the like) and on the other with ascending approaches, where syntactic sentence structure play an important part (the various theories of "discourse relations": SDRT, D-LTAG...). This choice is not aimless. In particular, ascending approaches postpone discourse analysis to previous, difficult, sentential analyses, discarding certain "global effects" which can be captured at a least cost. On the other size, methods based on statistical techniques only detect very rough structures.

Nevertheless the question arises whether CDML can take in charge these approaches, and there are some hints that it should be feasible. Concerning lexical cohesion, it could be tackled thanks to a new constraint, expressing some kind of "homogeneity" between the sets of lexical descriptors of consecutive text segments. Turning to "discourse relations" based theories, the constraint paradigm seems quite relevant inasmuch as some of those theories extend the syntax "outside the sentence" and that constraints have proved their value in syntax description. The passage from specific algorithm to static declarative description certainly looks possible and beneficial. We could also invoke the "rule-based approach" of [4] where "each rule is conditioned by a set of constraints", expressing lexical, syntactic or semantic information, each having a specific "weight".

Exploring such extensions of the range of CDML applications is a major aspect of our present and future research. The other one addressing the question of efficiency of the constraint resolution techniques.

References

1. Blache, P.: Property grammars: A fully constraint-based theory. In Christiansen, H., Skadhauge, P.R., Villadsen, J., eds.: Constraint Solving and Language Processing. Number LNAI 3438, Springer (2005) 1–16
2. Mann, W.C., Thompson, S.A.: Rhetorical Structure Theory: A theory of Text Organization. Technical Report ISI-RS-87-190, ISI: Information Sciences Institute, Marina del Rey, CA (1987)
3. Asher, N.: Reference to Abstract Objects in Discourse. Kluwer (1993)
4. Polanyi, L., van den Berg, M., Culy, C., Thione, G., Ahn, D.: A rule based approach to discourse parsing. In: Proceeding of SIGDIAL 2004, Boston (2004)
5. Webber, B.: D-LTAG: extending lexicalizer tag to discourse. Cognitive Science (28) (2004) 751–779
6. Hearst, M.: Multi-paragraph segmentation of expository text. In: Proceedings of the 32nd. Annual Meeting of the Association for Computational Linguistics, New Mexico State University, Las Cruces, New Mexico (1994) 9–16
7. Teufel, S.: Argumentative Zoning: Information Extraction from Scientific Articles. PhD thesis, University of Edinburgh (1999)
8. Widlöcher, A.: Analyse par contraintes de l'organisation du discours. In Mertens, P., Fairon, C., Dister, A., Watrin, P., eds.: Actes de TALN 2006, Leuven, Belgique, Presses Universitaires de Louvain (2006) 367–376
9. Halliday, M.A.K., Hasan, R.: Cohesion in English. Longman, London (1976)
10. Charolles, M.: L'encadrement du discours : Univers, champs, domaines et espaces. Cahier de Recherche Linguistique **6** (1997)
11. Bilhaut, F., Ho-Dac, M., Borillo, A., Charnois, T., Enjalbert, P., Le Draoulec, A., Mathet, Y., Miguet, H., Péry-Woodley, M.P., Sarda, L.: Indexation discursive pour la navigation intradocumentaire : cadres temporels et spatiaux dans l'information géographique. In: Proceedings of TALN 2003, Batz-sur-Mer, France (2003) 315–320
12. Bilhaut, F., Widlöcher, A.: LinguaStream: An Integrated Environment for Computational Linguistics Experimentation. In: Proceedings of the 11th Conference of the European Chapter of the Association of Computational Linguistics (EACL) (Companion Volume), Trento, Italie (2006) 95–98
13. Ferrari, S., Bilhaut, F., Widlöcher, A., Laignelet, M.: Une plate-forme logicielle et une démarche pour la validation de ressources linguistiques sur corpus : application à l'évaluation de la détection automatique de cadres temporels. In: Actes des 4èmes Journées de Linguistique de Corpus, Lorient, France (2005) To be published.

Making sense out of nonsense: The acceptability of repairs

Barbara Hemforth*, Joël Pynte*, Emmanuel Bellengier**

* LPNC, CNRS, Université Paris Descartes
**LPL, CNRS, Université Aix en Provence

Abstract

Acceptability judgments are very often taken to be a central empirical resource of grammaticality. However, it has been shown that grammaticality has to be considered as graded (e.g., Blache, Hemforth, & Rauzy, 2006; Bresnan, Cueni, Nikitina, & Baayen, 2005; Keller, 2000), such that not all constructions that used to be considered as ungrammatical are judged alike in every possible context. In this paper, we will present two experiments on disfluency–based repairs in French. We will show that non-linguistic constraints such as the current content of an individuals working memory play an important role for acceptability judgments.

In our first experiment, applying a speeded end of the sentences acceptability task with speeded auditory presentation we found that in French versions of sentences like (1,2) the inconsistent continuation is much more acceptable when a compatible but repaired constituent has been mentioned before, thus still influencing sentence interpretation.

(1)
I will go to the baker, uh no, the butcher on my way home. I need meat /bread.
(2)
I will go to the butcher on my way home. I need meat /bread

In our second experiment, we show that this effect is not due to the negation or the simple fact that the repaired constituent has been mentioned, since in French versions of sentences like (3), the inconsistent continuation is no more acceptable than in sentences like (2).

(3)
I will not go to the baker but to the butcher on my way home. I need meat /bread.

Introduction

The human sentence processing system has to be extremely robust because it does not only have to cope with highly standardized and edited to correct input, but very often also with deficient input caused by various non-linguistic situational factors. In this paper, we will look at the comprehension of disfluent utterances. We will see that disfluencies lead to changes in acceptability of sentences which are semantically incorrect.

Disfluencies are highly frequent in natural language production. They include editing terms such as ***uh*** and ***um*** as well as repeats ("I – uh - I wouldn't", e.g. Clark & Wasow, 1998) and revisions. Typically, in spoken language, disfluencies can be found in about six out of 100 words (Fox Tree, 1995). In the corpus used by Levelt (1983), 25 % of the annotated disfluencies were repairs as the structures under investigation here. Of these, 62 % had editing expressions like Dutch versions of "I mean" or "that is" or mostly (30 % of all repairs) the Dutch version of "uh". Since disfluencies in general and repairs in particular are so frequent, listeners have to find ways to process them, they have to detect the disfluency, see what the problem is, and edit out the part of speech to be repaired to arrive at the intended utterance.

However, recently it has been claimed that this process of editing out may not always work perfectly well. Lau and Ferreira (2005, see also Bailey & Ferreira, 2003, Ferreira, Lau, & Bailey, 2004) claim that the reparandum (the to be repaired constituent) in repetitions and corrections introduces lexical content and local syntactic structure which are not fully overwritten by the correction. They looked at a disfluency involving the repair of a verb (like chosen vs. selected) in sentences like (4 a, b).

(4)

a. The little girl chosen-uh selected for the role celebrated with her parents and friends.
b. The little girl picked-uh selected for the role celebrated with her parents and friends.

Sentences like these with verbs like "selected" which are ambiguous between a main verb and a past participle reading usually lead to comprehension difficulty (e.g., increased reading times), on the disambiguating prepositional phrase ("for the role"), in particular when the verb is biased for a simple main verb imperfect reading. This garden-path disappeared when the ambiguous verb was preceded by an unambiguous past participle ("chosen").

Although verb replacement is not very frequent (0,7 % of all repairs), it is highly interesting, because verbs immediately project syntactic structure that can be used for further syntactic integration. The structure projected by the replaced verb form (e.g. the past participle "chosen") apparently influences further processing, in that it reduces the garden-path in sentences like (4a) but not in (4b).

In our experiments, we investigated the influence of more frequent repairs on listeners' on-line processing of speech: repairs involving NPs as in (1). For French, corpus studies on radio interviews (Boula de Mareüil et al., 2005) show that revisions very often include a preposition (from 22%; to 7%) or a determiner (30 %). This means that at least 30% of the revisions include noun phrases, probably more, since a large part of the prepositions heads a prepositional phrase including a noun phrase.

This makes noun phrase revisions a fairly frequent phenomenon in spoken language. Another important difference between verb revisions and NP revisions is that the latter at least in our study do not involve the projection and possibly revision of syntactic structure. Since the NPs in our experiments did not have these far reaching syntactic effects, it was quite possible that the repaired NP does not show any influence on later processing.

Experiment 1: Repairs

In our first experiment, our basic hypothesis follows Bailey and Ferreira in claiming that the NP reparandum, even though it is different with respect to structure as well as frequency of occurrence, is not fully deleted just as the verbal reparandum. If so, it should affect the final interpretation of the sentence containing the replacement.

Material and Procedure

Participants. 56 native French undergraduate students, from the University of Aix-en-Provence and the University of Compiègne participated in this experiment in exchange for credits in a psychology (Aix) or linguistics (Compiègne) class.

Material. We presented participants with synthesized sentences using Elan SaySo ™ (Elan Speech). All utterances were produced at 22 kHz, using the female voice. Two sentences were synthesized for each item. The second sentence was speeded up by 30%, using a speed tag that does not change basic parameters like frequency. The synthesized materials were inspected by all authors and judged as highly natural.

We constructed 16 items each in four conditions as in examples (5a, b) and (6a, b).

(5) Je dois aller chez le boucher sur le chemin du retour. J'ai besoin de a. viande / b. bread.

I will have to go to the butcher on my way home. I need some a. meat/ b. bread.

(6) Je dois aller chez le boulanger, euh non le boucher sur le chemin du retour. J'ai besoin de a. viande /b. pain.

I will have to go the baker uh no the butcher on my way home. I need some a. meat/ b. bread.

In half of our sentences we included disfluencies in form of an NP replacement. The second experimental factor was the consistency of the last word of the second sentence with the contents of the first sentence.

The object of the 2nd sentence made this sentence either consistent (5a, 6a) or inconsistent (5b, 6b) as a continuation of the first sentence. The inconsistent continuations were however consistent with the reparandum. For half of the participants, the reparans and the reparandum were exchanged to control for plausibility effects.

Procedure. Participants were told that they would hear sentences generated by a computer. For each experimental item, a visual signal indicated that a sentence would be played. Once the sentence was complete, the participant had to judge its grammaticality. Judgments were automatically recorded by the experimental software (PERCEVAL André & al. 2003). Each experimental item was presented in one of the four experimental conditions across participants. Four randomized lists were prepared including 32 fillers half of which with disfluencies and 8 training sentences. The filler sentences varied in syntactic and semantic acceptability.

Predictions: If the reparandum still affects sentence interpretation, it should influence acceptability judgments in that sentences with an inconsistent continuation, that is however consistent with the reparandum; should be judged as more acceptable than sentences without a replacement. We therefore predict an interaction of the experimental factors (repair vs. no repair, and consistent vs. inconsistent).

Results

These predictions were confirmed in the acceptability judgements (see Fig. 1). Inconsistent sentences were judged less acceptable than consistent sentences (F1(1,55) = 111,04; $p < 0.001$; F2(1,15) = 129,17; $p < 0,001$). Sentences with repairs were more acceptable than sentences without repairs (F1(1, 55)=10,07; $p<0.01$; F2(1,15)=18,54; p<0;01). However, this main effect resulted from a reliable interaction between our two experimental factors: Participants judged sentences like (6b) as being more acceptable than (5b) ((F1(1,55) = 9.40; $p < 0.05$; F2(1,15) = 8.14; p < 0;05).

Discussion

Obviously, the reparandum affects the interpretation of the sentence in that continuations which are compatible with the reparandum but not with the reparans are still fairly acceptable. However, this effect may not be due to the disfluency but simply to the fact that the reparandum has been mentioned in the first sentence and is therefore available for a reinterpretation. Given that comprehenders like to make sense of whatever linguistic input is given to them, just mentioning the “baker” may make it easier to integrate “bread” although that information should have been deleted from the relevant sentential context. In order to test this possibility, we ran a second experiment.

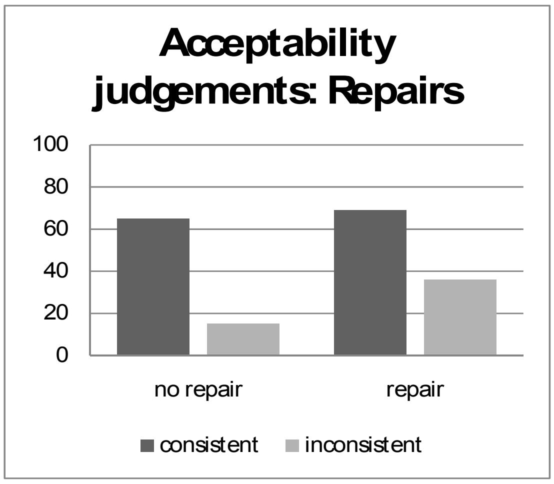


Fig. 1.

Experiment 2

In this experiment, we replaced the repairs from Experiment 1 with negations in order to test whether the persistence effect observed in Experiment 1 is due to the disfluency or to a more general effect of negations or even just an effect of the mentioning of the to be corrected NP..

(7) Je ne dois pas aller chez le boulanger mais chez le boucher sur le chemin du retour. J'ai besoin de a.viande, b.pain.
I will not have to go to the baker but to the butcher on my way home. I need some a. meat/ b. bread

Participants. 28 undergraduates of the University of Compiègne participated in this experiment.

Procedure and materials. We replaced the disfluencies by negations like (7a,b) for the 16 items used in Experiment 1. Otherwise, procedures as well as fillers were identical to those of Experiment 1.

Results

The factor “plausibility” was highly significant in this experiment (see Fig. 2) as in Experiment 1 (F1(1,27) = 230,46, p < 001; F2(1,15) = 210,55, p < 0.001). Negated sentences were slightly less acceptable than non-negated sentences (F1(1,27) = 4,46, p < 0.05; F2(1,15) = 4,24, p < 0.05). However, there was no reliable interaction of plausibility and negation (ps > .20).

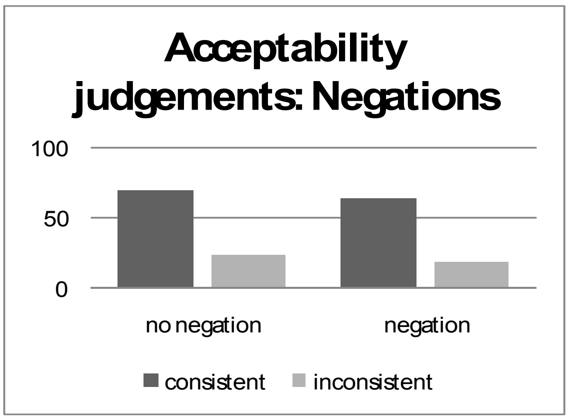


Fig. 2.

Experiment 2 shows that the “lingering” effect is not simply a consequence of the to be corrected NP having been mentioned, and thus not just an effect of priming. It is apparently specific for the processing of repairs.

Discussion

Disfluencies obviously do have a very specific effect on sentence processing. The fact that a non-negated discourse entity has been processed and stored in working memory at some point leaves a trace that can later be re-used to integrate linguistic material that should be judged as inconsistent with the sentence. An important differences between the disfluencies and the negations is that the to be corrected NP has at some point been positively integrated as part of the discourse model whereas this is not the case for the simple negations.

As Fodor and Inoue (1995) state:

"When a syntactic analysis is revised, it is essential that any semantic interpretation based on the initial incorrect analysis be erased. But it appears that the human sentence comprehension routines are not perfect; once a chunk of semantic representation is constructed and stored in working memory, it tends to persist unless obliterated by a new one overlaid on it."

How to account for the disfluency effects?

Obviously a parser that relies only on what a linguistic operation (such as a repair) is supposed to do cannot fully account for actual human performance data. To account for these effects, not only linguistic, but also cognitive constraints have to be taken into account. One of the general strategies is Fodor and Inoue's (2000) strategy "attach anyway" or even more generally "process anyway", a very general tendency to make sense of whatever input the human sentence processor is confronted with. This is in most cases a very useful strategy, since we are so often confronted with only approximately correct input.

Konieczny, Hemforth, Scheepers, and Strube (1996), for example, show how human comprehenders cope with input from a supposedly foreign speaker: In these experiments on German word order ambiguities, participants tended to interpret sentences with unconventional object<subject ordering as subject<object sentences, in particular, when they assumed that these sentences were produced by a non-native speaker (in this case an Irish student with a clearly noticeable accent). Taking the assumed imperfections of a foreign speaker into account, they also accepted ungrammatical sentences with two nominative marked subject NPs or two accusative marked objects more easily (mostly interpreting them as canonical subject<object sentences).

Obviously, the human sentence processor takes linguistic information - even if it is fully unambiguous - as more or less soft constraints depending on the context of the utterance that may be overridden in the interest of making sense out of nonsense.

References

Altmann, G.T.M., & Kamide, Y. (1999). Incremental interpretation at verbs: Restricting the domain of subsequent reference. Cognition, 73, 247-264.

André, C., Ghio A., Cavé C., & Teston B. (2003). "PERCEVAL: a Computer-Driven System for Experimentation on Auditory and Visual Perception", Proceedings of XVth ICPhS, Barcelone, Espagne, p. 1421-1424.

Bailey, K.G.B, & Ferreira, F. (2003). Disfluencies influence syntactic parsing. Journal of Memory and Language, 49, 183-200.

Bailey, K., & Ferreira, F. (2007). The processing of filled pause disfluencies in the visual world. In R. P. G. van Gompel, M. H. Fischer, W. S. Murray and R. L. Hill, Eye Movements: A Window on Mind and Brain, Elsevier LTD, 486-500.

Philippe Blache, Barbara Hemforth & Stéphane Rauzy (2006), "Acceptability Prediction by Means of Grammaticality Quantification" , in proceedings of COLING-ACL 06.

Boula de Mareüil, P., Habert, B., Bénard, F., Adda-Decker, M., Barras, C., Adda, G., & Paroubek. P. (2005). A quantitative study of disfluencies in French broadcast interviews. In Proceedings of Disfluency In Spontaneous Speech (DISS) Workshop, Aix-en-Provence, September 2005.

Brennan, S.E., & Schober, M.F. (2001). How listeners compensate for disfluencies in spontaneous speech. Journal of Memory and Language, 44:274-296.

Bresnan, J., Cueni, A., Nikitina, T., & Baayen, H. 2007. Predicting the Dative Alternation. In Cognitive Foundations of Interpretation, ed. by G. Boume, I. Kraemer, and J. Zwarts. Amsterdam: Royal Netherlands Academy of Science, pp. 69-94.Clark, H. H., & Wasow, T. (1998). Repeating words in spontaneous speech. Cognitive Psychology , 37, 201-242.

Corley, M., & Hartsuiker, R.J. (2003). Hesitation in speech can. . . um. . . help a listener understand. In Proceedings of CogSci2003, 2003.

Ferreira, F., Lau, E.F., & Bailey, K.G.D. (2004). Disfluencies, parsing, and tree-adjoining grammars. Cognitive Science, 721-749.

Fodor, J. D., & Inoue, A. (1994). The diagnosis and cure of garden-paths. Journal of Psycholinguistic Research, 23, 407-434.

Fodor, J. D. and A. Inoue (2000) “Garden Path Re-analysis: Attach (Anyway) and Revision as Last Resort,” in M. de Vincenzi and V. Lombardo, eds., Cross-linguistic Perspective on Language Processing, Kluwer, Dordrecht.

Fox Tree, J. E. (1995). The effects of false starts and repetitions on the processing of subsequent words in spontaneous speech. Journal of Memory and Language, 34, 709–738.

Keller F. (2000) Gradience in Grammar. Experimental and Computational Aspects of Degrees of Grammaticality, Phd Thesis, University of Edinburgh.

Konieczny, L., Hemforth, B., Scheepers, C. & Strube, G. (1996). Reanalysen vs. interne Reparaturen beim Sprachverstehen. In C. Habel, S. Kanngießer & G. Rickheit (Eds.), Perspektiven der kognitiven Linguistik: Modelle und Methoden (pp. 161-183). Opladen: Westdeutscher Verlag, 161-184.

Lau, E., & Ferreira, F. (2005). Lingering effects of disfluent material on comprehension of garden path sentences, Language and Cognitive Processes, 2005, 20 (5), 633–666.

Levelt, W. J. M. (1983). Monitoring and self-repair in speech. Cognition, 14(1):41-104.

Smith, V. L., & Clark, H. H. (1993). On the course of answering questions. Journal of Memory and Language, 32, 25–38.

Tanenhaus, M.K., Spivey-Knowlton, M.J., Eberhard, K.M. & Sedivy, J.E. (l995). Integration of visual and linguistic information in spoken language comprehension. Science, 268, 1632-1634.

van Dyke, J., & Lewis, R. (2003). Distinguishing effects of structure and decay on attachment and repair: A cue-based parsing account of recovery from misanalyzed ambiguities. Journal of Memory and Language, 49 (2003) 285–31.