Part of CONTEXT'11:
The 7th International and Interdisciplinary Conference on
Modeling and Using Context 2011

6th International Workshop, CSLP 2011

Constraints and Language Processing

Karlsruhe, Germany, 27 September 2011

Proceedings

Editors:

Philippe Blache
Henning Christiansen
Verónica Dahl
Jørgen Villadsen

Preface

This collection constitutes the Proceedings of CSLP 2011, the 6th International Workshop on Constraints and Language Processing, which takes place together with CONTEXT '11, the 7th International and Interdisciplinary Conference on Modeling and Using Context 2011, on September 27, 2011.

The CSLP 2011 workshop addresses the question of constraints and language processing from an interdisciplinary perspective. Constraints are widely used in linguistics, computer science, and psychology. How they are used, however, varies widely according to the research domain: natural language processing, 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 experiment have been made, investigating the role of constraint systems for cognitive processes in comprehension and production, as well as addressing how they can be acquired. The years' collocation with the CONTEXT'11 conference underlines the application of constraints for context comprehension and discourse modelling.

Previous CSLP workshops were held in 2008 (with ESSLLI, in Hamburg, Germany), 2007 (with Context07, in Roskilde, Denmark), 2006 (with COLING-ACL, Sydney, Australia), 2005 (with ICLP, Sitges, Spain), and in 2004 (as a separate event, Roskilde, Denmark). After a pause of three years, CSLP 2011 aims to gather a new overview of the field and, once again, provide a forum for a dialogue across disciplines, creating a potential for new research cooperations.

We want to thank all authors who submitted their very interesting papers, the program committee for its in-depths reviews, and not least the organizers of CONTEXT 2011, especially its workshop chair, Robert J. Ross, and the leader of the local organization, Hedda R. Schmidtke, for hosting the workshop.

Roskilde
September 2011

Philippe Blache
Henning Christiansen
Verónica Dahl
Jørgen Villadsen

Program Committee

Philippe Blache, CNRS & Université de Provence, France
Henning Christiansen, Roskilde University, Denmark
Verónica Dahl, Simon Fraser University, Canada & Tarragona, Spain
Barbara Hemforth, Université Paris Descartes, France
Helen de Hoop, Radboud University Nijmegen, Netherlands
Denys Duchier, Université d'Orléans, France
M. Dolores Jimenez-Lopez, Universitat Rovira i Virgili, Spain
Detmar Meurer, Universität of Tübingen, Germany
Patrick McCrae, Hamburg University, Germany
Véronique Moriceau, Université Paris XI, France
Gerald Penn, Universities of Toronto and Trinity College, Canada
Kiril Simov, Bulgarian Academy of Sciences, Bulgaria
Jørgen Villadsen, Technical University of Denmark

Table of Contents

Evaluating Language Complexity in Context: New Parameters for a Constraint-Based Model

Philippe Blache

Laboratoire Parole et Langage
CNRS & Université de Provence
blache@lpl-aix.fr

Abstract. Language processing can be more or less difficult for human subjects. It is then important to identify what are the complexity factors not only to better understand cognitive mechanisms, but also in the perspective of more realistic NLP tools. The problem is that existing models have been experimented on very small examples, on written texts. We present in this paper new parameters adapted to the description of the processing difficulty in natural situations: spoken language, conversations, etc. These parameters account for the fact that information comes from different modalities and that the form of the input in such situations can be very variable, even ill-formed.
These parameters have been integrated into a constraint-based formalism, making it possible to quantify them directly. The result is a *computational difficulty model*, opening the way to new investigation towards a better understanding of language processing and acquisition.

Keywords: Difficulty, complexity, language processing, computational model, constraints, Property Grammars

1 Introduction

Language processing is a complex task, both for human and machines. However, we still do not know precisely what are the complexity factors explaining the fact that some phenomena are more difficult to process, to understand or to acquire than others. Unfortunately, language complexity studies rely until now on very partial models, usually elaborated and validated with highly controlled linguistic material. Very recently, some experiences have been done, trying to validate models on medium-size corpora (see for example [Demberg08]). One conclusion of such experiences is that a broad-coverage model has to bring together several complexity parameters in order to cover more phenomena.

The problem is that these approaches only deal with written material. We explore in this paper the possibility of elaborating a complexity model for language in natural situation (typically conversational speech). In this perspective, we propose first to specify new complexity parameters, adequate for spoken language. We present then a way to implement these parameters as well as classical ones into an constraint-based model. This approach renders possible to propose a quantification of complexity, opening the way to new kinds of experimentations.

Before exploring these question, preliminary definitions are necessary. In the following, we propose to distinguish subdivide complexity into two separate problems: language complexity and difficulty, specified as follows:

1. *Language complexity*: competence level
   - Concerns general properties of language
   - Depends on the system, brings together all parameters
2. *Difficulty*: performance level
   - Concerns concrete realizations (text, speech)
   - Two levels of difficulty
     - *Global Difficulty*: interpretation difficulty of a sentence or an utterance
     - *Local Difficulty*: processing difficulty of a word or a given construction

Generally speaking, complexity is made of these two notions. In the following, we only focus on difficulty. The difficulty model we propose is then part of a more general complexity model.

2 Different Complexity Models

Linguistic complexity have been first studied in linguistic typology, in order to compare different languages by means of several parameters. For example, [Parkvall08] proposes a set of criteria taken from the *World Atlas of Language Structures*[1] and assigns the criterion a numerical value according to an empirical complexity scale. Each language receive a complexity index which is a function of the criteria values. However, this method is not precise enough, the criteria being too general, leading to contradictory results (cf. [Nichols09]).

Psycholinguistics is the other domain that propose in-depth studies on this question. Psycholinguists try to identify and quantify complexity parameters in order to explain language processing and acquisition. These works usually rely on the analysis of two mechanisms for sentence processing that can be source of complexity: integration (insertion of a new element into an existing structure) and storage (number of incomplete dependencies). We present in the remaining of this section some classical approaches to this question.

The **Incomplete Dependency Hypothesis** (cf. [Gibson98]) evaluates complexity in terms of incomplete structures to store. Example (1b) bears an embedded constituent (the relative clause) separating the noun from the verb it complements. The example (1c), which has two such constructions, is considered to be too complex to be interpretable. At the opposite, (1d) which has the same lexical material and the same number of relatives is considered to be more simple because without any break between the nouns and the verbs they are subject of.

(1) a. The reporter disliked the editor.

[1] http://wals.info/

b. The reporter [who the senator attacked] disliked the editor.
c. #The reporter [who the senator [who John met] attacked] disliked the editor.
d. John met the senator [who attacked the reporter [who disliked the editor]].

Dependency Locality Theory is a more complete approach (see [Gibson00]). It describes the complexity in terms of referential objects occurring between two syntactic structures, the latter to be integrated to the former. DLT takes into account discourse referents, their integration and memory costs.

Discourse referents can have different impact in the processing according to their status. A referent can be "*new*" (typically introduced by a pronoun or a proper noun) or "*accessible*" (already given, but indirectly). New referents (noted *DRn*) require more processing costs.

- *Integration costs*: defined in DLT as the distance between a head and its governor. It is approximated by the number of *DRn* between these two heads,
- *Storage costs*: the minimal number of syntactic heads needed in order to build a complete well-formed sentence. For example, after a subject , a verb is required to form a sentence.

Some works (cf. [Vasishth03]) propose that the kinds of costs described in DLT can be counterbalanced by the **Activation** of a word. When a given word (or category) is more predictable thanks to the context, it becomes easier to integrate it in the structure. For example, the sentence (2b) is shown experimentally to be processed more rapidly than (2a), contrarily to what is predicted by the IDH or DLT:

(2) a. The rat the cat saw died.
b. The rat the cat briefly saw died.

In (2b), the presence of the adverb activates the first verb. Other experiments show that this is regularly the case with pre-modifiers that always activate the head. Comparable results are shown in [Hawkins10] that makes use of the number of unsatisfied properties or rules at a given point: structures with more properties satisfied early in the sentence are preferred.

Today, most experimental works in this domain use **Surprisal** (cf. [Hale01]) which relies on lexical and syntactic criteria. This parameter provides a possibility to identify locus of complexity into a sentence and gives an indication of the overall complexity of the sentence in summing these values. This technique, simple and efficient, can be enriched in including other features such as semantics (cf. [Landauer97], [Pynte08]).

Surprisal is however not fully adequate when dealing with natural speech for mainly two reasons: information is multimodal (typically results from prosody, syntax and gestures interaction) and utterances are often syntactically ill-formed or non-canonical (see [Blache06]).

3 Information Complexity in Natural Interaction

As seen above, language complexity can have consequences at different levels, in particular interpretation: a message is more or less complex to understand depending on several parameters. Among them, the amount of available information plays an important role. Our hypothesis is that lack of information renders a message difficult to understand whereas redundancy (which comes to a large amount of information) facilitates interpretation.

When dealing with language in natural situation, we know that information comes from different sources (verbal and non verbal): taking into account only one source comes to decrease the amount of information. The following example illustrates this phenomenon for an interacting situation between two speakers.

The first figure shows an abstract of the information coming from the morpho-syntactic analysis of a short utterance, made of two words. Without any other context, each word bears some information represented by a feature structure.

`[Speaker B]` "*him no*"

$$\left[\begin{array}{l}\textsc{phon}\ \textit{him} \\ \textsc{ss}\mid\textsc{cat}\left[\begin{array}{l}\textsc{hd}\ \textit{Pro} \\ \textsc{index}\ \boxed{1}\left[\begin{array}{l}\textsc{per}\ \textit{3} \\ \textsc{gen}\ \textit{masc} \\ \textsc{num}\ \textit{sing}\end{array}\right] \\ \textsc{case}\ \textit{dat}\end{array}\right]\end{array}\right]\left[\begin{array}{l}\textsc{phon}\ \textit{no} \\ \textsc{ss}\mid\textsc{cat}\left[\begin{array}{l}\textsc{hd}\ \textit{Adv} \\ \textsc{content}\mid\textsc{neg}\ +\end{array}\right]\end{array}\right]$$

The content of each word defines a set of possible discourse referents: *him* specifies a subset of possible referents (singular masculine). Se also know that this referent will occupy an object position in the syntactico-semantic relation. The adverb *no* brings the information that this relation will be negated.

The set of possible referents is very large, rendering the interpretation very ambiguous and the difficult:

- $him = \sum DR_{object}[masc, sing]$
- $no = \sum DR_{event}[negated]$

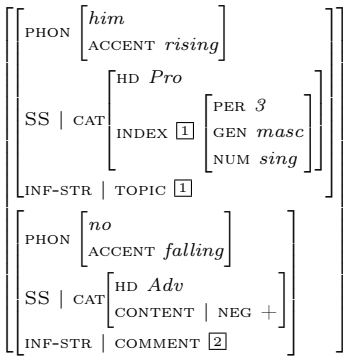
In this situation, no other information being given (in particular the semantic relation), no direct interpretation of the entire utterance is then possible.

The second figure illustrates the situation when prosodic information is accessible: a raising occurs with *him*, *no* being associated with a flat contour:

`[Speaker B]` "*him no*"
`LH*  LL`

The prosody is typical to a binary topic/comment construction: we tell something about *him*, this event being negated. One important information brought by prosody is that both items belong to a single construction, reducing the set of possible interpretations:

- $x \in DR_{him} \wedge y \in DR_{no} \wedge x \rightsquigarrow y$

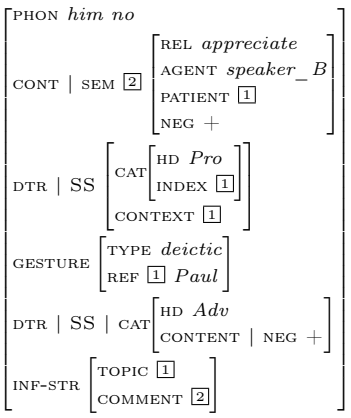


The last figure illustrates the case when the entire situation is accessible, two speakers interacting, speaker `A` showing by means of a deictic a person (let's caul him *Paul*) to speaker `B`, asking him a question (indicated by the final prosodic contour):

`[Speaker A]` "*do you appreciate him*"
`LH*`

`[Speaker B]` "*him no*"
`LH* LL`

These two utterances bring together information coming from lexicon and morpho-syntax, prosody, gestures and the context (the fact that *Paul* is in the scene), making it possible to build an interpretation. In this case, the set of possible referent is reduced to one element, *Paul*, thanks to the synchronization between a pronoun, a deictic gesture and a possible referent in the scene. The following structure represents the different pieces of information and their links indicated by means of coindexation.

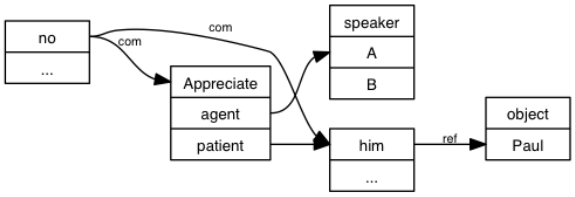


Each piece of information in this example works as a constraint reducing the set of possible referents, the final result being a unique referent, allowing unambiguous interpretation. This constraint-based process is exactly what we want to

implement in our model: as for problem solving, each new constraint facilitates the interpretation by reducing the search space. In terms of difficulty modelization, it is then necessary to integrate this aspect: more constraints means more information, more information comes to difficulty minimization.

This aspect becomes clear when representing the corresponding examples by means of graphs. In the situation where the transcription and the prosody are accessible, but not the context neither the gestures, the only semantic relation on top of which interpretation can be built is a topic/comment one:

At the opposite, when the entire context is accessible, a richer graph can be built, leading to a single interpretation as follows:



Such a representation shows a possibility to evaluate the quantity of information in terms of graph density. A high density semantic graph, bearing more information, is easier to interpret than a low density one.

4 Ill-formedness and Difficulty

Besides multimodality, a difficulty model of natural speech has to take into account ill-formedness and non canonical constructions, frequent in spoken language. Several studies have shown that constraint violations are subject to a cumulative effect (see for example [Keller00]): sentences can be ranked into a grammaticality scale according to the number of violated constraints (some constraints being more important than others). Sentences with low grammaticality ranking can also be judged as more difficult than others (this effect being also observed in language acquisition (see [Sorace05]). More recently, some studies have shown that constraint violation can be counterbalanced by other phenomena, among them the quantity of positive information available (see [Blache06]).

We propose to integrate these aspects to the difficulty model by evaluating precisely the different effects of constraint satisfaction/violation.

4.1 A Constraint-Based Representation

We briefly present in this section an approach making it possible to describe and quantify the effects mentioned above by means of constraints. In this approach

(see [Blache05]), all information can be represented by means of a constraint. They are of different types, implementing linear order, dependency, cooccurrence restrictions, arity, etc. The following table gives an example of constraints describing properties of different phrases in French:

Constraint type	Operational Semantics	*Example*
Constituency	Set of possible constituents	*Const(NP) = {Det, N, AP, PP, Rel, ...}*
Linear Precedence ($\prec$)	Linear order between constituents	$Det \prec N$
Cooccurrency ($\Rightarrow$)	Mandatory coooccurrencies	*V[ppass]* $\Rightarrow$ *aux[fin]* *{le, être[fin, acc$_i$]}* $\Rightarrow$ *reflexive[acc$_i$]*
Exclusion ($\otimes$)	Impossible coooccurrencies	*reflexive* $\otimes$ *lui* *lui* $\otimes$ *y*
Uniqueness	Constituents that cannot be repeated	*Uniqueness(NP) = {Det, N, PP, Rel, ...}*
Dependency ($\rightsquigarrow$)	Relations with the head	$NP \rightsquigarrow VP$
Adjacency ($\oplus$)	Constituents that have to be in an adjacent position	$AP \oplus N$

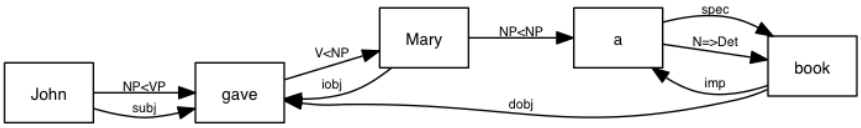
Table 1. Constraint types

In this approach, each linguistic information is represented by a constraint. Parsing a sentence comes to evaluate the set of constraints, the result of the parse being the set of satisfied constraints. In case of ill-formed or non-canonical constructions, some constraints can be violated. In such situation, the result of the parse is made of satisfied and violated constraints.

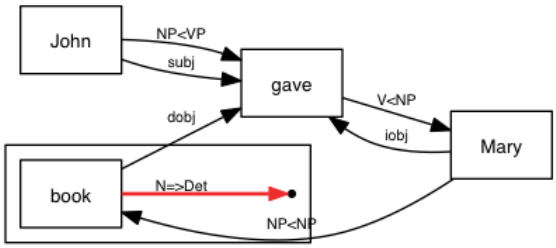
The interest of this approach is that it becomes possible to quantify the result of the parse thanks to different values (number of satisfied and violated constraints, their weights, etc.).

4.2 Compensation Effects

Constraints can be represented in terms of relations between the different constituents. The result of a parse is then equivalent to a graph. The following example shows a partial constraint graph for the sentence *"John gave Mary a book."* . This graph shows the main constraints (in particular linearity and dependency):

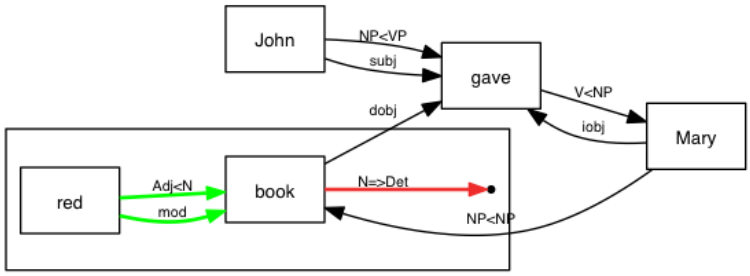


Let's take now the case where a constraint is violated such as in the example *"John gave Mary book."*. In this sentence, the lack of determiner generates a constraint violation: the obligation constraint between the common noun and the determiner is not satisfied. This situation is illustrated by the following graph, showing a NP containing one violated constraint:

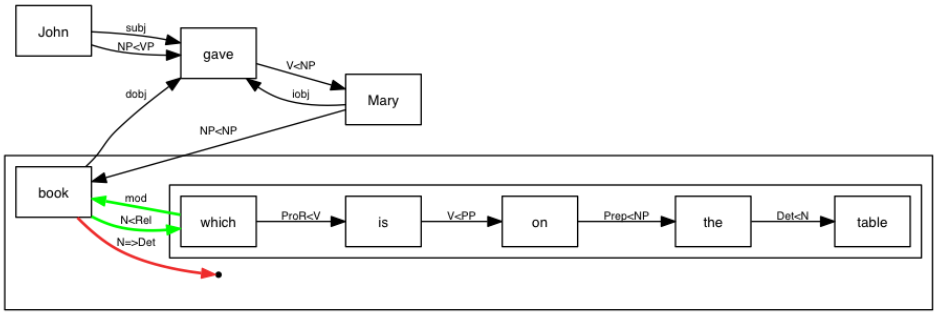


"John gave Mary book."

Let's analyze now the situation where an AP is realized before the noun in the second NP in the sentence *"John gave Mary red book."*. In this case, the obligation constraint is still violated, but new other constraints, describing the relation between the AP and the noun head are satisfied:



This sentence is considered by users as more acceptable than the previous. Such effect is predicted by the fact that new constraint are satisfied in the ill-formed NP, counterbalancing in a certain sense the effect of the violation. The same effect can be observed when other complements or adjuncts are realized such as in the following example, with a relative clause:



In these examples, new adjuncts bring more information in the NP both at the syntactic and the semantic levels. The NP correspond to a more specific referential object and then becomes more easily accessible. In a certain sense, adding more information facilitates the interpretation and partly compensate ill-formedness. This phenomenon is observed in many different situations, it comes to a positive cumulativity, besides the negative one: sentences with a large amount of satisfied constraints are more easily processed than others containing less information. Moreover, the amount of satisfied constraints can compensate constraint violation.

5 New Parameters for Difficulty Modelization

An integrated model of difficulty has to take into account various difficulty parameters. As seen above, sparseness and ambiguity play an important role in the modelization, they have been shown, through surprisal, to be good predictors of difficulty. However, example taken from natural situations (as presented in the previous section) show that information can be more or less precise, with direct consequence on interpretation difficulty.

Moreover, new difficulty models also need to be experimented against large amount of data. A first step in this end has been experimented in [Demberg08] which proposes to compare on a large corpus surprisal prediction with eye-tracking results. Such experiences requires robust methods able to deal with large corpora. Moreover, the study of difficulty in language processing has now to take into account natural language interaction. This means two requirements, to integrate to a model:

- necessity to take into account different sources of information, coming from different domains and modalities
- necessity to deal with non-canonical constructions

This general ideas can be formulated into new parameters to be integrated into the model:

1. **Information density** (noted ID)

- ID is the quantity of evaluable properties, measuring the graph density.
- ID has consequence on difficulty: high ID reduces search space and facilitates interpretation.
- ID is a global difficulty parameter (takes into account the entire utterance).

2. **Cumulativity**
   - Ill-formed constructions increase difficulty, the more constraints (or properties) are violated, the highest the difficulty is: there is a negative cumulativity effect.
   - The importance of linguistic properties present (or satisfied) in the utterance can compensate negativity: this is the positive cumulativity effect.
   - Cumulativity is a local difficulty parameter.
   - Cumulativity is a symbolic model for surprisal.

Information density implements multimodality in taking into account all kinds of properties, from any domain. On its side, cumulativity describes ill-formedness, but also compensation aspects occurring in natural situations. These parameters render quantifiable high-level information in a symbolic manner provided that the different linguistic properties can be represented and evaluated.

6 A Computational Model

We propose to integrate different parameters in our difficulty model. These parameters can be evaluated thanks to the constraint-based approach presented above. The idea consists in quantifying the different parameters on the basis of the set of evaluated constraints. The interest of such a set (also called characterization set), is that both density, cumulativity as well as grammaticality can be easily quantified.

We propose in this section a quantification method for different parameters relying on the analysis of the state of the constraint system after the parse. More precisely, we propose a method implementing each complexity parameter in terms of constraints: starting from the characterization set $\mathcal{C}$ (containing for a given sentence the set of satisfied and violated constraints), different procedures are presented in order to assign the corresponding parameter a value.

Incomplete Dependency: This parameter relies on the number of elements between the head and a complement. Two constraints identifies this situation: dependency and linearity. For a given category C_i, the incomplete dependency value is the number of dependency relations coming from categories preceded by C_i and going to C_i or categories that precede C_i. It can be defined as follows[2]:

$$ID[C_i] = |\{C_k \rightsquigarrow C_j \mid (C_j \preceq C_i) \wedge (C_i \prec C_k)\}| \quad (1)$$

[2] This simple definition has been suggested by an anonymous reviewer, as well as the operational semantics of the Dependency Locality rule (next paragraph).

Dependency Locality: Integration costs of DLT rely on the referents between a head and its dependent before it. The parameter value is the number of such referent. The following rule has a logical left-hand side and a side-effecting right-hand side, indicating that RHS has to be executed for every solution of the LHS in the characterization.

$$[(C_i \rightsquigarrow C_j) \wedge (C_i \prec C_j)] \wedge [\nexists k \mid (C_k \rightsquigarrow C_j) \wedge (C_k \prec C_i)] \Rightarrow DLT[j] \leftarrow |DRn_{[i,j]}| \tag{2}$$

$$\text{with } DRn_{[i,j]} = \{C_l[+ref] \mid i \leq l \leq j\}$$

This constraint identifies the leftmost dependent C_i of a head C_j. The DLT value is the cardinality of the set of the discourse reference between these two categories.

Activation: The activation degree of a category C_i can be evaluated as the weight of all the relations with other categories C_j that precedes it. The following function quantifies this activation in identifying for a given category C_i at a position i the number of constraints that takes C_i as an argument. It consists in traversing the set of constraints and selecting the constraints between two categories, C_i being the last.

$$Activation(C_i) = \sum Weight(\mathcal{P}_{C_i}) \text{ with } \mathcal{P}_{C_i} = \{P(x, C_i) \mid (x \sqcup C_j) \wedge (C_j \prec C_i)\} \tag{3}$$

The set of constraints activated for C_i is noted $\mathcal{P}_{C_i}$, a constraint is noted $P(x, C_i)$ with P representing the constraint type and x a constraint argument. When x unifies with a category preceding C_i and entering into a constraint relation with C_i, the corresponding constraint is added to $\mathcal{P}_{C_i}$. In order to take into account the relative importance of each constraints, their weights are summed. Finally, a category is highly activated when it is the target of a high number of constraints.

In our model, activation is a facilitation parameter, it is then inversely proportional to difficulty.

Information Density: The first parameter describing density relies on the number of constraints used in the parse, taking into account the number of categories. In terms of graph, this ratio represents the density of the graph:

$$Density = \frac{\texttt{nb of constraints}}{\texttt{nb of categories}} \tag{4}$$

In our hypothesis, a dense graph represents a sentence (or an utterance) for which a lot of constraints can be evaluated. Let's remind the fact that in a complete description, these constraints are properties coming from all the

different domains (prosody, syntax, semantics, but also discourse, pragmatics, gestures, etc.). High density means a lot of information. Moreover, having more constraints means the possibility to reduce the search space and then facilitate the interpretation. In conclusion, density is inversely proportional to difficulty.

Density has to be completed with another information about the theoretical maximal density. Each category is described in the grammar by a certain number of constraints. In some cases, very few constraints are necessary for a complete description (e.g. adverbial phrases). Inversely, some other constructions require a lot of constraints. It is then necessary to know whether a construction description uses all or only part of the possible constraints describing it in the grammar. We propose for this a completeness ratio indicating for a given constituent (or construction) the proportion of possible constraints used for its parse. We call this ration *Completeness*:

$$Completeness = \frac{\texttt{nb of evaluated constraints}}{\texttt{nb of relevant constraints}} \tag{5}$$

In the same way as density, completeness also facilitates interpretation in reducing ambiguity. It is inversely proportional to difficulty.

Cumulativity: As shown above, dealing with unrestricted linguistic material (in particular spoken language) requires to implement robust methods. In our constraint-based approach, constraint relaxation makes it possible to build a parse whatever the form of the input. However, it is necessary to evaluate the overall quality of the parse. For a given constituent, when all evaluated constraints are satisfied, its interpretation will be facilitated. Reciprocally, a lot of violated constraints increases difficulty. We propose to measure this aspect with the following ratio:

$$Satisfaction = \frac{\texttt{nb of satisfied constraints}}{\texttt{nb of evaluated constraints}} \tag{6}$$

Finally,it is necessary to take into account positive as well as negative cumulativity. Moreover, each constraint can have different importance, represented by different weights. We propose a *Quality* ratio accounting for these aspects. It relies on the sum of the weights of constraints that are satisfied in the parse (noted W^+) and the that of the violated ones (noted W^-). The following ratio proposed a normalized quantification of this information:

$$Quality = \frac{W^+ - W^-}{W^+ + W^-} \tag{7}$$

The table 2 recaps the relative contribution of each parameter to the difficulty model, some of them being proportional, some others inversely proportional to difficulty.

A general computational model of complexity consists in bringing together these different parameters and to complete them with other indications such as the size of the description (i.e. Kolmogorov complexity) or surprisal.

Parameter	Proportional	Inversely proportional
Incomplete Dependency	+	
Dependency Locality	+	
Activation		+
Density		+
Completeness		+
Satisfaction		+
Quality		+

Table 2. Contribution of Parameters to the Difficulty

7 Conclusion

Different difficulty models for language processing exist for written material. It is now necessary to propose models accounting for uses in natural situations (typically conversational speech). Such models have to account for specific phenomena: information is spread over different modalities, utterances can be ill-formed, interpretation can be rendered difficult due to ambiguity in reference resolution, etc.

We have analyzed in this paper some of these specificities and proposed new parameters. In particular, we have described the relations between information density and difficulty: a lot of information, despite the fact that they require more resources to be processed, facilitates the interpretation and then decreases difficulty. In the same way, the utterances to parse are often ill formed or non canonical, but not necessarily difficult to process. We have explained this fact by compensation mechanisms: violated constraints can be conterbalanced by the importance of the satisfied one.

Finally, the constraint-based representation we propose makes it possible to quantify directly these different parameters, opening the way to an integrated model of difficulty for language processing in natural context. This computational model renders possible new experimentations in language processing and acquisition: all kinds of utterances can be evaluated and their difficulty quantified. We have now to validate the model in verifying that it is a good predictor for human processing.

References

[Blache05] Blache P. (2005), "Property Grammars: A Fully Constraint-Based Theory", in *Constraint Solving and Language Processing*, H. Christiansen & al. (eds), LNAI 3438, Springer

[Blache06] Blache P. , B. Hemforth, & S. Rauzy (2006) "Acceptability prediction by means of grammaticality quantification", in proceedings of *COLING/ACL 2006*.

[Demberg07] Demberg V. & F. Keller (2007) "Eye-tracking evidence for integration cost effects in corpus data", in Proceedings of the 29th annual conference of the cognitive science society

[Demberg08] Demberg V. & F. Keller (2008) "Data from eye-tracking corpora as evidence for theories of syntactic processing complexity", in *Cognition*, 109

[Gibson98] Gibson E. (1998) "Linguistic complexity: Locality of syntactic dependencies", in *Cognition* vol. 68 (1).

[Gibson00] Gibson, E. (2000) "The dependency locality theory: A distance-based theory of linguistic complexity". In A. Marantz, Y. Miyashita, & W. O?Neil (Eds.), Image, language, brain: Papers from the first mind articulation project symposium, MIT Press.

[Hale01] Hale J. (2001) "A probabilistic Earley parser as a psycholinguistic model", in proceedings of ACL-2001

[Hawkins01] Hawkins J. (2001) "Why are categories adjacent", in Journal of Linguistics, 37.

[Hawkins10] Hawkins J. (2010) "Processing efficiency and complexity in typological patterns", in Oxford Handbook of Linguistic Typology, Oxford University Press.

[Kusters08] Kusters W. (2008) "Complexity in Linguistic Theory, Language Learning and Language Change", in *Language Complexity*, Miestamo et al. (eds), John Benjamins.

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

[Keller10] "Cognitively Plausible Models of Human Language Processing" in proceedings of ACL-2010

[Landauer97] Landauer, Thomas K. and Susan T. Dumais (1997) "A solution to Plato's problem: The latent se- mantic analysis theory of acquisition, induction and representation of knowledge", in *Psychological Review* 104(2)

[Lindstrom08] Lindstrňöm E. (2008) "Language complexity and interlinguistic difficulty", in *Language Complexity*, Miestamo et al. (eds), John Benjamins.

[Miestamo09] Miestamo M. (2009) "Implicational hierarchies and grammatical complexity", in *Language Complexity as an Evolving Variable*, Sampson et al. (eds), Oxford University Press.

[Nichols09] Nichols J. (2009) "Linguistic complexity: a comprehensive definition and survey", in *Language Complexity as an Evolving Variable*, Sampson et al. (eds), Oxford University Press.

[Parkvall08] Parkvall M. (2008) "The simplicity of creoles in a cross-linguistic perspective", in *Language Complexity*, Miestamo et al. (eds), John Benjamins.

[Pynte08] Pynte J., B. New and A. Kennedy (2008) "On-line contextual influences during reading normal text: A multiple-regression analysis", in *Vision Research* 48(21)

[Sorace05] Sorace A. & F. Keller (2005) "Gradience in Linguistic Data", in *Lingua*, 115.

[Vasishth03] Vasishth S. (2003) "Quantifying processing difficulty in human sentence parsing: The role of decay, activation, and similarity-based interference", in Proceedings of Eurocogsci 03: The European Cognitive Science Conference 2003

Approaching the Chinese Word Segmentation Problem with CHR Grammars

Henning Christiansen and Bo Li

Research group PLIS: Programming, Logic and Intelligent Systems
Department of Communication, Business and Information Technologies
Roskilde University, P.O.Box 260, DK-4000 Roskilde, Denmark
E-mail: {henning, bol}@ruc.dk

Abstract. Written Chinese text does not include separators between words, as do European languages using space characters, and this creates the Chinese Word Segmentation Problem: given a text in Chinese, divide it in a correct way into segments corresponding to words. Correctness means how a competent Chinese language user would do this. CHR Grammars (CHRG) is an implemented grammar system that allows highly flexible bottom-up analyses using rule-based constraint solving techniques. We demonstrate how different approaches to the problem can be expressed in CHRG in a highly concise way, and how different principles can complement each other in this paradigm. We do not claim to have provided any improvement with methods currently in use, our aims are a) to forward a way for further experimentation with solutions to the problem, and b) to show how CHRG gives rise to succinct and executable specifications of such methods. We present here some preliminary and promising experiments tested on simple examples.

1 Introduction

Chinese text is written without explicit separation between the different words and it is up to the reader to make this separation; however, periods are unambiguously delineated using the special character "。" which serves no other purpose. The Chinese language presents the same collection of problems as any other language with respect to automatic analysis, including syntactic (lexical, structural) and semantic ambiguities, unknown words, etc. Compared with European languages, written Chinese exposes further problematic issues, most notably the lack of word separators which is further emphasized by that fact that most single characters, seen in isolation, may form a word, and quite many pairs of two characters also form words. As in other languages, analysis is also made difficult by the fact that certain parts may be left out of a sentence; as opposed to most European languages, even the verb may be left out when obvious from the context, though mostly verbs corresponding to "have" or "be". Also, Chinese has almost no inflectional markers. These facts make it even more difficult to use syntactic constraints or cues to guide or validate a given segmentation.

The recent textbook [1] contains a good introduction to these difficulties also for non-Chinese speakers.

Good solutions to the Chinese Word Segmentation Problem, henceforth referred to as CWSP, are evidently in demand for different kinds of digital processing of Chinese text, ranging from simple type-setting (breaking text into lines), over web search engines (text indexing and query processing), to (front-ends for) different sorts of deep analysis. As Chinese is one of the most used languages on the Internet,[1] the problem has attracted much research attention; proceedings of the CIPS-SIGHAN Joint Conferences on Chinese Language Processing since 2002 and previous workshops provide a good overview of the history and state of the art of such methods [2]. Lexicon-based approaches, complemented by different heuristics and statistical-based methods are dominating; see, e.g., [1, 3] for overview. Controlled competitions between different Chinese word segmentation systems have been arranged together with the CIPS-SIGHAN conferences. The report from the 2010 competition [4] shows precision and recall figures up to around 0.95 for tests on selected corpora. This seems quite impressive and indicate only little room for improvement; on the other hand it is not obvious that the reported results will hold on arbitrary unseen (types of) text.

Some general systems for Internet search such as Google[2] and Baidu[3] use their own word segmentation algorithms which are not publicly available; [3] provides some tests and discussion of these approaches.

CHR Grammars [5] is a grammar formalism and implemented system that adds a grammar notation layer on top of the programming language of Constraint Handling Rules [6, 7], CHR, analogous to the way Definite Clause Grammars [8] are added on top of Prolog. CHR itself was introduced in early 1990es as a rule-based, logical programming language for writing constraint solvers for traditional constraint domains such as integer or real numbers in a declarative way, but has turned out to be a quite general and versatile forward-chaining reasoner suited for a variety of applications, including language processing through CHRG. CHR, often in the shape of CHRG, have been used for a variety of language processing tasks until now, but to our knowledge, not to Chinese before; see, e.g., [9–15]. In this paper we present the first, early experiments in approaching CWSP using CHRG, and our experience so far is very promising in that different principles for (partly) solving CWSP can be expressed in very elegant ways. This should be seen in contrast to the difficulties and comprehensive amount of detailed programming that would be necessary if similar experiments and prototype implementations were made using a traditional programming language such as Java or C.

In the following, we start giving a brief overview of CHRG in section 2. Section 3 shows how a lexicon can be represented in a CHRG and demonstrates it for

[1] Chinese is currently the secondly most used language on the Internet after English, and, based on the current growth rate, it may very well be the most used in a few years; consult, e.g., http://www.internetworldstats.com/stats7.htm.

[2] http://www.google.com

[3] http://www.baidu.com; the biggest Chinese web search engine.

a rudimentary CWSP method based on the so-called maximum match principle, and 4 shows how two simple CHRG rules can split a text into smaller portions, called maximum ambiguous segments, which can be analyzed separately. In section 5, we discuss further principles that we would like to experiment with in CHRG, and section 6 gives a short summary and a conclusion.

2 Brief Overview of CHR Grammars

We assume the terminology and basic concepts of CHR and Prolog to be known, but the following introduction of CHR Grammars may also provide sufficient insight to readers with a broader background. These grammars are implemented as an additional layer of syntax on top of a Prolog-based CHR system and are automatically compiled into CHR rules when loaded. The CHRG system and a comprehensive Users' Guide are available [16].

The basic idea is that grammar symbols (terminals and nonterminals) are represented as constraints, decorated with integer numbers, that refer to positions in the text and in this way maintain the usual sequential order. Rules work bottom up: when certain patterns of grammar symbols are observed in the store, a given rule may apply and add new grammar symbols. Consider the following example of a grammar given as its full source text.

```
:- chrg_symbols noun/0, verb/0, sentence/0.
[dogs] ::> noun.
[cats] ::> noun.
[hate] ::> verb.
noun, verb, noun ::> sentence.
end_of_CHRG_source.
```

Given the query

```
?- parse([dogs,hate,cats])
```

constraints corresponding to the three terminal symbols will be entered; the three "lexical" rules will apply and add new grammar symbols representing the recognition of two `noun`s and a `verb` in a suitable order such that the last rule can apply and report the recognition of a `sentence`. The answer is given as the final constraint store which will include the following constraint; notice that the positions (or boundaries) in the string, that were invisible in the grammar, are shown.

```
sentence(0,3).
```

The rules shown above are *propagation* rules, which add new grammar symbols to the existing ones; when the arrow in a rule is replaced by `<:>`, the rule becomes a *simplification* rule which will remove the symbols matched on the letfhand side; there is also a form of rules, called *simpagations*, that allow to remove only some of the matched symbols which will be shown below.

Notice that when simplification rules are used, the actual result of a parsing process may depend on the procedural semantics of the underlying CHR system (which rules are applied when), and a knowledge about this is needed for those who want to exploit the full power of CHRGs. However, these properties imply a natural handling of ambiguity: when propagation rules are used, all possible analysis are generated in the same constraint store, while simplification rules may be applied for pruning or sorting out among different solutions.

In some cases, it may be relevant to order the application of rules into phases such that firstly all rules of one kind apply as much as possible, and then a next sort of rules is allowed to apply. This can be done by embedding non-grammatical constraints in the head of grammar rule, declared as ordinary CHR constraints. We can illustrate this principle by a modification of the sample grammar above.

```
...
:- chr_constraint phase2/0.
{phase2}, noun, verb, noun ::> sentence.
```

Notice the special syntax with curly brackets, which is inspired by Definite Clause Grammars [8]. This means that, in this rule, the constraint `phase2` does not depend on positions in the text, but must be present for the rule to apply. The query for analysis should then be changed as follows.

```
?- parse([dogs,hate,cats]), phase2.
```

This means that first, the lexical rules will apply as long as possible (as they are not conditioned by the constraint `phase2`), and when they are finished, the `sentence` rule is allow to be tried. In this particular example, this technique will in fact not change the result, but we give examples below where it is essential.

CHRG rules allow an extensive collection of patterns on the lefthand side for how grammar symbols can be matched in the store: context-sensive matching, parallel matching, gaps, etc.; these facilities will be explained below when they are used in our examples. As in a Definite Clause Grammar [8], grammar symbols may be extended with additional arguments that may store arbitrary information of syntactic and semantic kinds.

CHRG runs under SICStus Prolog 4 [17] and SWI Prolog [18], which are both capable of handling UNICODE characters, so Chinese characters and text can be represented directly.

3 Representing Lexicon in a CHR Grammar; Lexicon-Based Methods

The simplest way to represent a lexicon in a CHRG is a by sequence of short rules as those we indicated as lexical rules above. For our first experiments with CWSP, we represent a lexicon classifying words only. The following example rules provide the lexicon for examples to follow.

```
[中]    ::> word([中]).          % centre, middle
[中,华] ::> word([中,华]).       % Chinese (adjective), China
[华,人] ::> word([华,人]).       % Chinese (people)
[人]    ::> word([人]).          % people, human
[人,民] ::> word([人,民]).       % people
[国]    ::> word([国]).          % country
[国,中] ::> word([国,中]).       % high-school
[共,和] ::> word([共,和]).       % republic
[共,和,国] ::> word([共,和,国]). % republic, country
[中,华,人,民,共,和,国] ::> word([中,华,人,民,共,和,国]).   % People's-republic-of-China
[中,央] ::> word([中,央]).       % central
[政,府] ::> word([政,府]).       % government
[民,政] ::> word([民,政]).       % civil-administration
[中,央,人,民,政,府]   ::> word([中,央,人,民,政,府]).      % People's central government
```

Notice that the grammar contains two rather large words that look like compounds, but which will be included in any dictionary as words as they are known and fixed terms with fixed meanings.

The `word` grammar symbol may of course be extended with syntactic tags, but for now we will do with the simplest form as shown.

A simple Lexicon-Based Method for CWSP, Maximum Matching

A first naive idea for CWSP may be to generate all possible words from the input, followed by an assembly of all possible segmentations that happens to include the entire input, and then a final phase selecting a best segmentation according to some criteria. Obviously, this is of exponential or worse computational complexity and thus more efficient heuristics have been developed. One such heuristics is the maximum matching method, which has been used in both forward and backward versions; here we show the forward method (see [1] for background). The sentence is scanned from left to right, always picking the longest possible word; then the process continues this way until the entire string has been processed.[4]

Three CHRG rules are sufficient to implement this principle. The first one, which needs some explanation, will remove occurrences of words that are proper prefixes of other word occurrences.

```
!word(_) $$ word(_), ... <:> true.
```

The "`$$`" operator is CHRG's notation for parallel match: the rule applies whenever both of the indicated patterns match grammatical constraints in the store for the same range of positions (i.e., substring). The symbol "`...`" refers to a *gap* that may match any number of positions in the string, from zero and upwards, independently of whatever grammar symbols might be assiciated with those positions.[5] In other words, the pattern "`word(_), ...`" matches any substring that starts with a word. So when this is matched in parallel with a single

[4] In theory, this may fail to capture the entire string, but as most single characters can represent a word, this will be extremely rare.

[5] In fact, gaps themselves are not implemented by matching, but affect how its neighbouring grammar symbols are matched, putting restrictions on their word bound-

word, it can apply in exactly those cases where two words occur, one being a (not necessarily proper) prefix of the other. Finally, the exclamation mark in front of the first `word` indicates, in a rule having the `<:>` arrow (which otherwise signifies simplification), a simpagation rule. The meaning is that here only grammar symbols and constraints marked with "`!`" are kept in the store. The `true` on the righthand side stands for nothing, meaning that no new constraints or grammar symbols are added.

So when a string is entered, this rule will apply as many times as possible, each time a lexicon rule adds a new word, and thus keeping only longest words.

In a second phase, we compose a segmentation from left to right, starting from the word starting after position 0. The first rule applies an optional notation in "`:(0,_)`", which makes the word boundaries explicit, here used to indicate that this rule only applies for a leftmost word. The `compose` constraint is used as described above to control that these rules cannot be applied before all short words have been removed by the rule above.

```
{!compose}, word(W):(0,_) <:> segmentation(W).
{!compose}, segmentation(Ws), word(W) <:> segmentation(Ws/W).
```

Assuming the lexicon given above, we can query this program as follows, shown also with the answer found (with constraints removed that are not important for our discussion).

```
?- parse([中,华,人,民,共,和,国,中,央,人,民,政,府]), compose.
segmentation(0,13,[中,华,人,民,共,和,国]/[中,央,人,民,政,府])
```

Here the method actually produces the right segmentation, meaning "The Central People's Government of the People's Republic of China"; the "of" being implicit in the Chinese text. Notice that there is actually a word spanning over the split, namely the word for high-school. This example showed also the advantage of combining the maximum match principle with having common terms or idioms represented as entries in the lexicon.

We can show another example that demonstrates how maximum matching easily can go wrong, with a suggestion for a repair. We extend the lexicon with the following rules.

---

aries. In the example shown, it must hold that $r_1 \geq r_2$ for the rule to apply where r_1 and r_2 designate the right boundary of the first, resp., the second `word` in the rule head.

```
[明,确]    ::> word([明,确]).    % definitude, clearly
[确,实]    ::> word([确,实]).    % really, actually
[实,在]    ::> word([实,在]).    % honest, actually
[考,虑]    ::> word([考,虑]).    % consider
[将,来]    ::> word([将,来]).    % future
[将,来,的] ::> word([将,来,的]). % future-related
[李]       ::> word([李]).       % Li (family name)
[李,子]    ::> word([李,子]).    % plum
[明]       ::> word([明]).       % bright, Ming (given name)
[将]       ::> word([将]).       % will
[在]       ::> word([在]).       % at
[来]       ::> word([来]).       % come
[事]       ::> word([事]).       % thing
```

The sample sentence we want to check is "李明确实在考虑将来的事", which can be translated into English as "Li Ming is really considering the future things" corresponding to the correct segmentation

```
[李,明]/[确,实]/[在]/[考,虑]/[将,来,的]/[事].
```

Querying the maximum matching program as shown above for this sentence gives the segmentation

```
[李]/[明,确]/[实,在]/[考,虑]/[将,来,的]/[事]
```

that does not give sense to a Chinese reader. The problem is that the first two characters are not seen as a unit, but the first character is taken as a single word and thus the second and third second character are taken as a word, and so on. In the middle of the sentence, the program accidentally gets on the right track again and gets the remaining words right. Due to the high frequency of two-character words in Chinese, it is easy to produce quite long sentences where one wrong step in the beginning makes everything go wrong for the maximum matching method.

If instead, in the example above, the two characters for the personal name Li Ming are treated as one unit, everything would go right. This could suggest that a specialized algorithm for identifying personal names might be useful as an auxiliary for CWSP, as has been suggested among others by [19]. We can simulate such a facility by adding a rule for this specific name as follows.

```
[李,明] ::> word([李,明]).  % Li Ming (person name)
```

Finally, we mention that combinations of forward and backward maximum segmentation have been used, and in those regions where the two disagree, more advanced methods are applied; see, e.g., [20].

4 Identifying Maximum Ambiguous Segments

Another principle that may be used in algorithms for CWSP is to run a first phase, identifying the maximum ambiguous segments of a text. We have distilled

the principle from a variety of methods that apply similar principles; we have not been able to trace it back to a single source, but [1] may be consulted for an overview and [2] for detailed contributions.

An *ambiguous segment* is defined as a contiguous segment s in which

- any two contiguous characters are part of a word,
- there are at least two words that overlap, and
- the first and last character are each part of a word entirely within s.

For example, if *abcd* and *def* are words, then the substring *abcdef* will form an ambiguous segments, but not necessarily *cdef* or *abcdefg*. An ambiguous segment is *maximal*, a MAS, whenever it cannot be expanded in any direction to form a larger ambiguous segment. For example, if *abc*, *cde*, *def*, *defg* are words, then the substring *abcdefg* may form a maximal ambiguous segment.

In other words, if no unknown words occur in a text, the splits between the MASs will be definitive. Except in construed cases, the length of the MASs are reasonable, which means that we can apply more expensive methods subsequently within each MAS, perhaps even with exponential methods that enumerate and evaluate all possible segmentations.

Identifying these MASs can be done by very few CHR rules. For simplicity, we introduce a grammar symbol `maxap` which covers MASs as well as single words that can only be recognized as such. Assuming a lexicon defined as shown above, which identifies any possible word in the text, the following two CHRG rules and an additional Prolog predicate are sufficient to identify the `maxap`s.

```
word(W) ::> maxap.
maxap:R1, ... $$ ..., maxap:R2 <:> overlap(R1,R2) | maxap.

overlap((A1,B1),(A2,B2)):- A1 < B2, A2 < B1.
```

The second rule uses the auxiliary Prolog predicate `overlap` as a guard. The presence of a guard, between the arrow and the vertical bar, means that the rule can only apply in those cases where the guard is true. The `overlap` predicate tests, as its name indicates, whether the two segments in the string occupied by the two input `maxap`s do overlap. This grammar rule will gradually put together ambiguous segments and, via repeated applications, merge together so only maximum ones remain.

We can test this program for the previous example, “Li Ming is really ...” as follows.

```
?- parse([李,明,确,实,在,考,虑,将,来,的,事]).
...
maxap(0,5)
maxap(5,7)
maxap(7,10)
maxap(10,11)
```

This corresponds to splitting the sequence into the substrings

李明确实在，考虑，将来的，事，

which then can be separately analyzed.

5 Further extensions

Our main sources on CWSP research [1, 2] report also statistically based methods of different sorts, possibly combining with part-of-speech tagging. While part-of-speech tagging is straightforward to add via the lexicon rules, CHRG is currently not supported by machine-learning techniques to produce useful statistics. However, it is straightforward to integrate probabilities and other weighting schemes in a CHRG: each constituent has an associated weight, and when a rule applies, it calculates a new weight for the compound. Additional rules can be added that prune partial segmentations of low weight. Comprehensive statistics concerning ambiguity phenomena in Chinese text is reported by [21], which appears to be very useful for further research into CWSP.

Furthermore, we can extend the CHRG rules with particular knowledge about the Chinese language. For example, the sign "的" (pronounced "de") normally serves as a marker that converts a preceding noun into an adjective; in fact, most adjectives are constructed in this way from nouns which often have no direct equivalent in European languages, e.g., adjective "red" is constructed from a noun for "red things". Thus, what comes before "的" should preferably be a noun.[6] There are of course lots of such small pieces of knowledge that can be employed and should be employed, and we may hope that the modular rule-based nature of CHR can make it possible to add such principles in an incremental way, one by one.

We did not yet approach the out-of vocabulary (OOV) problem, but one direction to follow is to identify specific patterns in the way that a CHRG-based analyzer attempts to solve the problem with a limited dictionary, including which grammatical constraints that might be broken in this process. OOV words are often proper names and it is obvious that a module for recognizing proper names should be included. We have already referred to [19] that suggests an approach to recognize person names, and [1] lists several characteristics than may be applied in identifying also place names, transcription of foreign names, etc. We may also refer to an interesting approach to OOV in CWSP that incorporate web searches [22]. In [3] a method is suggested that involves web searches to evaluate alternative suggestions for segmentations which also may improve performance in case of OOV.

6 Conclusion

We have presented the first steps of experimentations using CHR Grammars to approach the Chinese Word Segmentation Problem. This problem has a high

[6] There are few additional usages of "的" (where it is pronounced "di"), but these are in special words that are expected to be included in any Chinese dictionary.

demand for efficient and precise solutions due to the high presence of the Chinese language on the Internet, as well as for Chinese language processing in general. It is also an interesting test case for the versatility of CHR Grammars. We have demonstrated very concise and succinct contributions to solutions in terms of CHR Grammar rules, which we take as a preliminary evidence for the suitability of CHR Grammars as an experimental platform for the Chinese Word Segmentation Problem.

We do not believe scaling to be a big issue for our approach:

- Periods in Chinese are always delimited in an unambiguous way, which puts an upper limit to the length of the texts that need to be treated as a hole.
- The straightforward lexicon-as-grammar-rules approach that we have applied here, which is perfect for small prototypes, does not scale well to full dictionaries. However, it is easy to get around this problem using an external dictionary, so that a text is entered into the CHR Grammar system as a character sequence (as shown here) together with constraints that represent all possible word occurrences in the text.

Then CHR Grammars' flexibility may be utilized to handle lots of special cases, perhaps ordered into layered phases, as demonstrated in our examples. An important next step is to incorporate methods for handling OOV words.

Acknowledgements. This work is supported by the project "Logic-statistic modelling and analysis of biological sequence data" funded by the NABIIT program under the Danish Strategic Research Council.

References

1. Wong, K.F., Li, W., Xu, R., Zhang, Z.S.: Introduction to Chinese Natural Language Processing. Morgan and Claypool publishers (2010)
2. ACL Anthology: A Digital Archive of Research Papers in Computational Linguistics: Special Interest Group on Chinese Language Processing (SIGHAN) Webarchive with all articles of CIPS-SIGHAN Joint Conference on Chinese Language Processing 2010, Proceedings of the nth SIGHAN Workshop on Chinese Language Processing, $n = 1, \ldots, 6$, 2002–2010, Second Chinese Language Processing Workshop. 2000. http://www.aclweb.org/anthology/sighan.html. *Link checked July 2011.*
3. Li, B.: Research on Chinese Word Segmentation and proposals for improvement. Master's thesis, Roskilde University, Computer Science Studies, Roskilde, Denmark (2011)
4. Zhao, H., Liu, Q.: The CIPS-SIGHAN CLP 2010 Chinese Word Segmentation Bakeoff. In: Proceedings of the Joint Conference on Chinese Language Processing, Association for Computational Linguistics (2010) 199–209
5. Christiansen, H.: CHR Grammars. Int'l Journal on Theory and Practice of Logic Programming **5**(4-5) (2005) 467–501
6. Frühwirth, T.W.: Theory and practice of Constraint Handling Rules. Journal of Logic Programming **37**(1-3) (1998) 95–138

7. Frühwirth, T.: Constraint Handling Rules. Cambridge University Press (August 2009)
8. 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
9. Christiansen, H., Dahl, V.: Logic grammars for diagnosis and repair. International Journal on Artificial Intelligence Tools **12**(3) (2003) 227–248
10. Bavarian, M., Dahl, V.: Constraint based methods for biological sequence analysis. Journal of Universal Computing Science **12**(11) (2006) 1500–1520
11. Dahl, V., Voll, K.D.: Concept formation rules: An executable cognitive model of knowledge construction. In Sharp, B., ed.: NLUCS, INSTICC Press (2004) 28–36
12. Dahl, V., Blache, P.: Extracting selected phrases through constraint satisfaction. In: Constraint Solving and Language Processing; Proceedings of the 2nd International Workshop. Volume 104 of Datalogiske skrifter, Roskilde University. (2005) 3–17
13. Christiansen, H., Have, C.T., Tveitane, K.: From use cases to UML class diagrams using logic grammars and constraints. In: RANLP '07: Proc. Intl. Conf. Recent Adv. Nat. Lang. Processing. (September 2007) 128–132
14. Dahl, V., Gu, B.: A CHRG analysis of ambiguity in biological texts. In: CSLP '07: Proc. 4th Intl. Workshop on Constraints and Language Processing. Volume 113 of Computer Science Research Report. (August 2007) 53–64 Extended Abstract.
15. Hecksher, T., Nielsen, S.T., Pigeon, A.: A CHRG model of the ancient Egyptian grammar. Unpublished student project report, Roskilde University, Denmark (December 2002)
16. Christiansen, H.: CHR Grammar web site; released 2002. http://www.ruc.dk/~henning/chrg (2002)
17. Swedish Institute of Computer Science: SICStus Prolog Website (checked 2012) http://www.sics.se/isl/sicstuswww/site/index.html.
18. SWI Prolog Organization: SWI Prolog Website (checked 2012) http://www.swi-prolog.org/.
19. Chen, Y., Jin, P., Li, W., Huang, C.R.: The Chinese persons name disambiguation evaluation: Exploration of personal name disambiguation in Chinese news. In: CIPS-SIGHAN Joint Conference on Chinese Language Processing 2010. (2010) Online proceedings, http://aclweb.org/anthology/W/W10/W10-4152.pdf.
20. Zhai, F.W., He, F.W., Zuo, W.L.: Chinese word segmentation based on dictionary and statistics. Journal of Chinese Computer Systems, **9**(1) (2009) (No page numbers given)
21. Qiao, W., Sun, M., Menzel, W.: Statistical properties of overlapping ambiguities in Chinese word segmentation and a strategy for their disambiguation. In Sojka, P., Horák, A., Kopecek, I., Pala, K., eds.: TSD. Volume 5246 of Lecture Notes in Computer Science., Springer (2008) 177–186
22. Qiao, W., Sun, M.: Incorporate web search technology to solve out-of-vocabulary words in Chinese word segmentation. In: Proceedings of 11th Pacific Asia Conference on Language, Information and Computation (PACLIC'2009). (2009) 454–463

Cross-framework Grammar Engineering using Constraint-driven Metagrammars

Denys Duchier, Yannick Parmentier, and Simon Petitjean

LIFO, Université d'Orléans, F-45067 Orléans Cedex 2, France,
firstname.lastname@univ-orleans.fr,
WWW home page: http://www.univ-orleans.fr/lifo/

Abstract. In this paper, we present an abstract constraint-driven formalism for grammar engineering called *eXtensible MetaGrammar* and show how to extend it to deal with cross-framework grammar engineering. As a case study, we focus on the design of tree-adjoining, lexical-functional, and property grammars (TAG / LFG / PG).
A particularly interesting feature of this formalism is that it allows to apply specific constraints on the linguistic structures being described.

Keywords: computational linguistics, formal grammar, metagrammar, constraint solving.

1 Introduction

Many grammatical frameworks have been proposed over the last decades to describe the syntax of natural language. Among the most widely used, one may cite Tree-Adjoining Grammar (TAG) [1], Lexical-Functional Grammar (LFG) [2], or Head-driven Phrase Structure Grammar (HPSG) [3]. These frameworks present theoretical and practical interests. From a theoretical point of view, they provide a formal device for the linguist to experiment with her/his theories. From a practical point of view, they make it possible to automatically process natural language in applications such as dialog systems, machine translation, *etc.* They differ in their expressivity and complexity. Some reveal themselves more adequate for the description of a given language than others. Still, for many of these frameworks, large resources (*i.e.*, grammars) have been designed, at first by hand, and later via dedicated tools (*e.g.*, integrated grammar environments such as XLE for LFG [4]). In this paper, we are concerned with this complex task of grammar engineering, keeping in mind the two above-mentioned theoretical and practical interests.

Several approaches have been proposed for a computer-aided grammar engineering, mainly to reduce the costs of grammar extension and maintenance. The main approaches are 1. the automatic acquisition from treebanks (see *e.g.*, [5] for LFG), 2. systems based on an abstract description of the grammar, either via transformation rules, also known as *metarules* (see *e.g*, [6] for TAG) or via a description language, sometimes called *metagrammar* (see *e.g.*, [7] for TAG). The

advantage of the description-based approach (and especially metagrammars[1]) over the automatic acquisition approach lies in the linguistic control it provides. Indeed, these descriptions capture linguistic generalizations and make it possible to reason about language at an abstract level. Describing language at an abstract level is not only interesting for structure sharing within a given framework, but also for information sharing between frameworks and / or languages.

This observation was already made by [9,10]. In their papers, the authors showed how to extend an existing metagrammar for TAG so that both a TAG and an LFG could be generated from it. They annotated TAG metagrammatical elementary units (so-called classes) with extra pieces of information, namely (i) LFG's functional descriptions and (ii) filtering information to distinguish common classes from classes specific to TAG or LFG. The metagrammar compilation then generated an extended TAG, from which LFG rules were extracted. To maximize the structure sharing between their TAG and LFG metagrammars, the authors defined classes containing tree fragments of depth one. These fragments were either combined to produce TAG trees or associated with functional descriptions to produce LFG rules. This cross-framework experiment was applied to the design of a French / English parallel metagrammar, producing both a TAG and a LFG. This work was still preliminary. Indeed (i) it concerned a limited metagrammar (the target TAG was composed of 550 trees, and the associated LFG of 140 rules) (ii) more importantly, there is no clear evidence whether a generalization to other frameworks and / or languages could be possible (metagrammar implementation choices, such as tree fragment depth, were not independent from the target frameworks).

Here, we chose to adopt a more generalized approach by designing an extensible metagrammatical language, that can handle an arbitrary number of distinct target frameworks. The linguist can thus use the same formalism to describe different frameworks and grammars. Nonetheless, if one wants to experiment with multi-formalism, *e.g.*, by designing a parallel TAG / LFG grammar, nothing prevents her/him from defining "universal" classes, which contain metagrammatical descriptions built on a common sublanguage. Rather than designing a new metagrammatical language from scratch, we propose to extend an existing formalism, namely *eXtensible MetaGrammar* (XMG) [11], which seems particularly adequate thanks to its modularity and extensibility.

The paper is organized as follows. In section 2, we briefly introduce TAG, as well as the redundancy issues raising while developing large TAG grammars (which motivated metagrammars). We then introduce the XMG metagrammatical language and show how it can be used to design TAG grammars. In section 3, we briefly introduce LFG and present an extension of XMG to describe LFG grammars. In section 4, we introduce Property Grammar (PG) [12], and present a second extension of XMG to generate PG grammars. In section 5, we will generalize over these two extensions, and define a layout for cross-framework grammar engineering. Finally, we conclude and give perspectives in section 6.

[1] In rule-based descriptions, one has to carefully define the ordering of the applications of rules [8], which makes it hard to design large grammars.

2 eXtensible Meta Grammar: generating Tree-Adjoining Grammars with a metagrammar

2.1 Tree-Adjoining Grammar

TAG[2] is a tree rewriting system, where elementary trees can be combined via two rewriting operations, namely *substitution* and *adjunction*. Substitution consists in replacing a leaf node labelled with ↓ with a tree whose root has the same syntactic category as this leaf node. Adjunction consists in replacing an internal node with a tree where both the root node and one of the leaf nodes (labelled with ⋆) have the same syntactic category as this internal node. As an illustration, consider Fig. 1 below. It shows (i) the substitution of the elementary tree associated with the noun *John* into the elementary tree associated with the verb *sleeps*, and (ii) the adjunction of the elementary tree associated with the adverb *deeply* into the tree associated with *sleeps*.

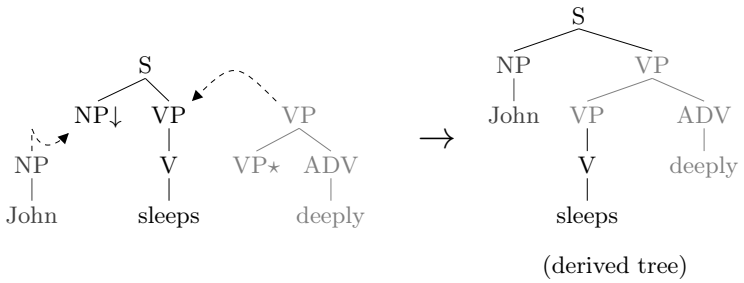


Fig. 1. Tree rewriting in TAG

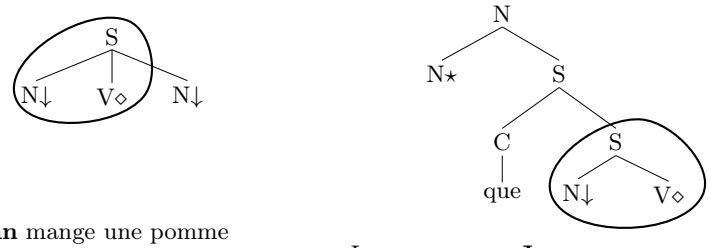
Basically, a real size TAG is made of thousands of elementary trees [14,15]. Due to TAG's extended domain of locality, many of these trees share common sub-trees, as for instance the relation between a canonical subject and its verb, as shown in Fig. 2. To deal with this redundancy, the metagrammar approach (in particular XMG) proposes to describe large TAG grammars in an abstract and factorized way.

2.2 eXtensible MetaGrammar (XMG)

XMG is a metagrammatical language inspired by logic programming. The idea behind XMG is that a metagrammar is a declarative logical specification of what a grammar is. This specification relies on the following three main concepts:

- several *dimensions* of language (*e.g.* syntax, semantics) can be described;
- for *each* of these dimensions, descriptions are made of non-deterministic combinations of elementary units;
- for *some* of these dimensions, descriptions must be solved to produce models.

[2] For a detailed introduction to TAG, see [13].



Jean mange une pomme
John eats an apple

La pomme que **Jean** mange
The apple that John eats

Fig. 2. Structural redundancy in TAG

XMG's extensibility comes from the concept of dimensions. These allow to describe an arbitrary number of types of linguistic structures. Non-determinism allows for factorization, and description solving for assembly and validation (*i.e.*, well-formedness of the description according to some target framework). Hereafter, we will first use XMG to describe TAG. Then, we will apply XMG's extensibility to the description of other frameworks, namely LFG and PG. Eventually, we will generalize over these applications.

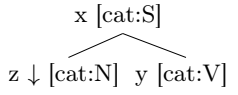
When describing TAG trees with XMG, one defines both (i) tree fragments and (ii) constraints that express how these fragments have to be combined to produce the grammar. Two languages are thus used: a *description language* $\mathcal{L}_D$ to specify fragments, and a *control language* $\mathcal{L}_C$ to specify combination constraints.

$\mathcal{L}_D$ is based on the precedence and dominance relations. Furthermore, since TAG allows for the labelling of syntactic nodes with feature structures, so does $\mathcal{L}_D$. A description in $\mathcal{L}_D$ is a formula built as follows:

$$Desc := x \to y \mid x \to^+ y \mid x \to^* y \mid x \prec y \mid x \prec^+ y \mid x[f{:}E] \mid x(p{:}E) \mid Desc \wedge Desc$$

where x, y refer to node variables, $\to$ (resp. $\prec$) to the dominance (resp. precedence) relation, and $+$ (resp. $*$) are used to denote the transitive (resp. reflexive and transitive) closure of this relation. The square brackets are used to associate a node variable with some feature structure. Parenthesis are used to associate a node variable with some property (such as the TAG $\star$ property seen in Fig. 1). Note that node variables are *by default* local to a description. If a node variable needs to be accessed from outside its description, it is possible to use some *export* mechanism. Once a variable is exported, it becomes accessible using a dot operator. For instance, to refer to the variable x in the description $Desc$, one writes $Desc.x$. Here is an illustration of a fragment description in XMG (on the right, one can see a minimal model of this description):

$$(x\ [cat : S] \to y\ [cat : V]\ ) \wedge (x \to z\ (mark : subst)\ [cat : N]\ ) \wedge (z \prec y)$$



$\mathcal{L}_C$ offers three mechanisms to handle fragments: *abstraction* via parameterized *classes* (association of a name and zero or more parameters with a content),

conjunction (accumulation of contents), and *disjunction* (non-deterministic accumulation of contents). A formula in $\mathcal{L}_C$ is built as follows:

$$\begin{array}{rcl} \mathit{Class} & := & \mathit{Name}[p_1, \ldots, p_n] \rightarrow \mathit{Content} \\ \mathit{Content} & := & \mathit{Desc} \mid \mathit{Name}[\ldots] \mid \mathit{Content} \vee \mathit{Content} \mid \mathit{Content} \wedge \mathit{Content} \end{array}$$

As an illustration of $\mathcal{L}_C$, let us consider different object realizations. One could for instance define the 4 fragments: (i) canonical subject, (ii) verbal morphology, (iii) canonical, and (iv) relativized object, and the following combinations, thus producing the two trees of Fig. 2:

$$\begin{array}{c} \mathit{Object} \rightarrow \mathit{CanObj} \vee \mathit{RelObj} \\ \mathit{Transitive} \rightarrow \mathit{CanSubj} \wedge \mathit{VerbMorph} \wedge \mathit{Object} \end{array}$$

Metagrammar compilation. To produce a grammar from an XMG metagrammar, we let the logical specification generate, in a non-deterministic way, descriptions. In other words, the combination constraints are processed to generate descriptions (one per dimension). For some dimensions, descriptions need to be solved to produce models. This is the case for TAG, a constraint-based tree description solver is thus used to compute trees [11]. Note this solver actually checks several types of constraints [16]: tree well-formedness constraints, TAG-related constraints (*e.g.*, unique node labelled $\star$), and language-related constraints (*e.g.*, uniqueness and order of clitics in French).

As one of the first ambitions of XMG is multi-formalism, dimensions are an efficient way to define different types of description language adapted to target frameworks. Let us see how to define dimensions for LFG and PG.

3 Generating Lexical-Functional Grammars with a metagrammar

3.1 Lexical-Functional Grammar

A lexical-functional grammar (LFG) consists of three main components: 1. context-free rules annotated with functional descriptions, 2. well-formedness principles, and 3. a lexicon. From these components, two main interconnected structures can be built[3]: a c(onstituent)-structure, and a f(unctional)-structure. The c-structure represents a syntactic tree, and the f-structure grammatical functions in the form of recursive attribute-value matrices. As an example of LFG, consider the Fig. 3 below. It contains a toy grammar and the c- and f-structures for the sentence "John loves Mary". In this example, one can see functional descriptions labelling context-free rules (see (1) and (2)). These descriptions are made of equations. For instance, in rule (1), the equation $(\uparrow SUBJ) = \downarrow$ constrains the $SUBJ$ feature of the functional description associated with the left-hand side of the context-free rule to *unify* with the functional description associated with

[3] This connection is often referred to as *functional projection* or *functional mapping.*

the first element of the right-hand side of the rule. In other words, these equations are unification constraints between attribute-value matrices. Nonetheless, these constraints may not provide enough control on the f-structures licensed by the grammar, LFG hence comes with three additional well-formedness principles (completeness, coherence and uniqueness) [2].

Toy grammar:

(1) S → NP VP
$\uparrow=\downarrow$ $(\uparrow SUBJ)=\downarrow$ $\uparrow=\downarrow$

(2) VP → V NP
$\uparrow=\downarrow$ $\uparrow=\downarrow$ $(\uparrow OBJ)=\downarrow$

(3) John NP, $(\uparrow PRED) =' JOHN', (\uparrow NUM) = SG, (\uparrow PRES) = 3$

(4) Mary NP, $(\uparrow PRED) =' MARY', (\uparrow NUM) = SG, (\uparrow PRES) = 3$

(5) loves V, $(\uparrow PRED) =' LOVE\langle(\uparrow SUBJ)\,(\uparrow OBJ)\rangle', (\uparrow TENSE) = PRESENT$

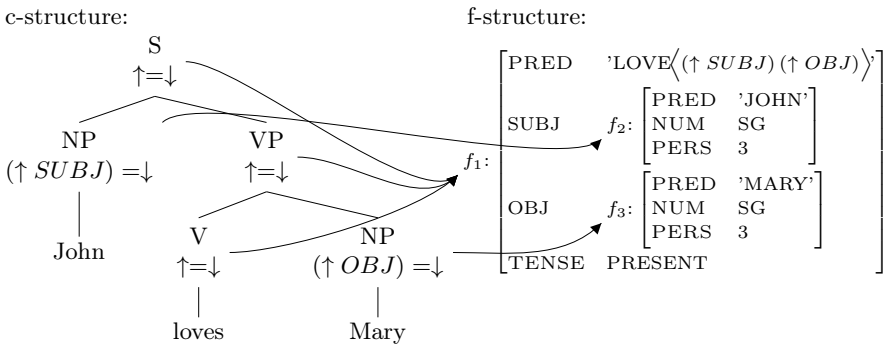


Fig. 3. LFG grammar and c-and f-structures for the sentence "John loves Mary"

3.2 Extending XMG for LFG

In the previous section, we defined the XMG language, and applied it to the description of TAG. Let us recall that one of the motivations of metagrammars in general (and of XMG in particular) is the redundancy which affects grammar extension and maintenance. In TAG, the redundancy is higher than in LFG. Still, as mentioned in [9], in LFG there are redundancies at different levels, namely within the rewriting rules, the functional equations and the lexicon. Thus, the metagrammar approach can prove helpful in this context. Let us now see what type of language could be used to describe LFG.[4]

To describe LFG at an abstract level, one needs to describe its elementary units, which are context-free rules annotated with functional descriptions (*e.g.*, equations) and lexical entries using attribute-value matrices. Context-free rules

[4] A specification language for LFG has been proposed by [17], but it corresponds more to a model-theoretic description of LFG than to a metagrammar.

can be seen as trees of depth one. Describing such structures can be done in XMG using a description language similar to the one for TAG, *i.e.*, using the $\rightarrow$ (dominance) and $\prec$ (precedence) relations. One can for instance define different context-free backbones according to the number of elements in the right-hand sides of the LFG rules. These backbones are encapsulated in parameterized XMG classes, where the parameters are used to assign a syntactic category to a given element of the context-free rule, such as in the class *BinaryRule* below.

$$\begin{array}{l} BinaryRule[A,B,C] \;\; \rightarrow \;\; (x[cat:A] \rightarrow y[cat:B]) \wedge (x \rightarrow z[cat:C]) \wedge (y \prec^{+} z) \\ \quad exports \;\; \langle x,y,z \rangle \end{array}$$

We also need to annotate the node variables x, y, z with functional descriptions. Let us see how these functional descriptions F_{Desc} are built:[5]

$$\begin{array}{rl} F_{desc} := & \exists(g\ FEAT) \mid \neg(g\ FEAT) \mid\ (g * FEAT) \mid\ (g\ FEAT)\ \ CONST\ \ VAL \mid \\ & F_{desc} \vee F_{desc}\ \mid\ (F_{desc})\ \mid\ F_{desc} \wedge F_{desc} \end{array}$$

where g refers to an attribute-value matrix, $FEAT$ to a feature, VAL to a (possibly complex) value, $CONST$ to a constraint operator ($=$ for unification, $=_c$ for constraining unification, $\in$ for set membership, $\neq$ for difference), (F_{Desc}) to optionality, and $*$ to LFG's functional uncertainty. Note that g can be complex, that is, it can correspond to a (relative – using $\uparrow$ and $\downarrow$ – or absolute) path pointing to a sub-attribute-value matrix.

To specify such functional descriptions, we can extend XMG in a straightforward manner, with a dedicated dimension and a dedicated description language $\mathcal{L}_{LFG}$ defined as follows:

$$\begin{array}{rcl} Desc_{LFG} & := & x \rightarrow y \ \mid\ x \prec y \ \mid\ x \prec^{+} y \ \mid\ x = y \ \mid\ x[f{:}E] \ \mid\ x\langle F_d \rangle \ \mid \\ & & Desc_{LFG} \ \wedge \ Desc_{LFG} \\ F_d & := & g \mid\ \exists g.f \mid\ g.f = v \mid\ g.f =_c v \mid\ g.f \in v \mid\ \neg F_d \mid\ F_d \vee F_d \mid \\ & & (F_d) \mid\ F_d \wedge F_d \\ g, h & := & \uparrow \mid \downarrow \mid\ h.f \mid\ f * i \end{array}$$

where g, h are variables denoting attribute-value matrices, f, i (atomic) feature names, v (possibly complex) values, and $\langle \ldots \rangle$ corresponds to LFG's functional mapping introduced above. With such a language, it now becomes possible to define an XMG metagrammar for our toy LFG as follows.[6]

$$\begin{array}{rl} Srule & \rightarrow br = BinaryRule[\texttt{S}, \texttt{NP}, \texttt{VP}] \wedge br.x\langle\uparrow{=}\downarrow\rangle \wedge br.y\langle(\uparrow \texttt{.SUBJ}) {=}\downarrow\rangle \\ & \quad \wedge\ br.z\langle\uparrow{=}\downarrow\rangle \\ VPrule & \rightarrow br = BinaryRule[\texttt{VP}, \texttt{V}, \texttt{NP}] \wedge br.x\langle\uparrow{=}\downarrow\rangle \wedge br.y\langle\uparrow{=}\downarrow\rangle \\ & \quad \wedge\ br.z\langle(\uparrow \texttt{.OBJ}) {=}\downarrow\rangle \end{array}$$

[5] We do not consider here additional LFG operators, which have been introduced in specific LFG environments, such as *shuffle*, *insert or ignore*, *etc.*

[6] Here, we do not describe the lexical entries, these can be defined using the same language as the LFG context-free rules, omitting the right-and-side.

In this toy example, the structure sharing is minimal. To illustrate what can be done, let us have a look at a slightly more complex example taken from [9]:

$$\begin{array}{lcccc} VP \rightarrow & V & (NP) & PP & (NP) \\ & \uparrow{=}\downarrow & (\uparrow OBJ) {=}\downarrow & (\uparrow SecondOBJ) {=}\downarrow & (\uparrow OBJ) {=}\downarrow \end{array}$$

Here, we have two possible positions for the NP node, either before or after the PP node. Such an situation can be described in XMG as follows:

$$\begin{array}{l} VPrule2 \rightarrow br = BinaryRule[\texttt{VP}, \texttt{V}, \texttt{PP}] \ \wedge \ u[cat : \texttt{NP}] \ \wedge \ br.y \prec^{+} u \\ \qquad\qquad \wedge \ br.y\langle\uparrow{=}\downarrow\rangle \ \wedge \ br.z\langle(\uparrow \texttt{.SecondOBJ}) {=}\downarrow\rangle \ \wedge \ u\langle(\uparrow \texttt{.OBJ}) {=}\downarrow\rangle \end{array}$$

Here, we do not specify the precedence between the NP and PP nodes. We simply specify that the NP node is preceded by the V node (denoted by y). When compiling this description with a solver such as the one for TAG, two solutions (LFG rules) will be computed. In other terms, the optionality can be expressed directly at the metagrammatical level, and the metagrammar compiler can directly apply LFG's uniqueness principle.

In other words, the metagrammar here not only allows for structure sharing via the (conjunctive or disjunctive) combination of parameterized classes, but it also allows to apply well-formedness principles to the described structures. In the example above with the two NP nodes, this well-formedness principle is checked on the constituent structure and indirectly impacts the functional structure (which is the structure concerned with these principles). If we see the functional structures as graphs and equations as constraints on these, one could imagine to develop a specific constraint solver. This would allow to turn the metagrammar compiler into an LFG parser, which would, while solving tree descriptions for the constituent structure, solve graph-labelling constraints for the functional structure.

Note that a similar approach of structure sharing within an LFG through combinations of elementary units has been proposed by [18]. In their paper, the authors describe how to share information between LFG structures by defining named descriptions, called templates. These templates can abstract over conjunction or disjunction of templates, they are thus comparable to our metagrammar classes. The main difference with our approach, is that nothing is said about an interpretation of these templates (they act in a macro-like fashion), while in XMG, one could apply some specific treatments (*e.g.* constraint solving) on the metagrammar classes.

4 Generating Property Grammars with a metagrammar

4.1 Property Grammar

Property Grammar (PG) [12] differs from TAG or LFG in so far as it does not belong to the generative syntax family, but to the model-theoretic syntax one. In PG, one defines the relations between syntactic constituents not in terms

of rewriting rules, but in terms of local constraints (the so-called *properties*).[7] The properties licensed by the framework rely on linguistic observations, such as linear precedence between constituents, coocurrency, mutual exclusion, *etc.*

Here, we will consider the following 6 properties, that constrain the relations between a constituent (*i.e.*, the node of a syntactic tree), with category A and its sub-constituents (*i.e.*, the daughter-nodes of A):[8]

Obligation	$A : \triangle B$	at least one B child
Uniqueness	$A : B!$	at most one B child
Linearity	$A : B \prec C$	B child precedes C child
Requirement	$A : B \Rightarrow C$	if a B child, then also a C child
Exclusion	$A : B \not\Rightarrow C$	B and C children are mutually exclusive
Constituency	$A : S$	children must have categories in S

In a real size PG, such as the French PG of [19], these properties are encapsulated (together with some syntactic features) within *linguistic constructions*, and the latter arranged in an inheritance hierarchy[9]. An extract of the hierarchy of [19] is presented in Fig. 4 (fragment corresponding to basic verbal constructions).

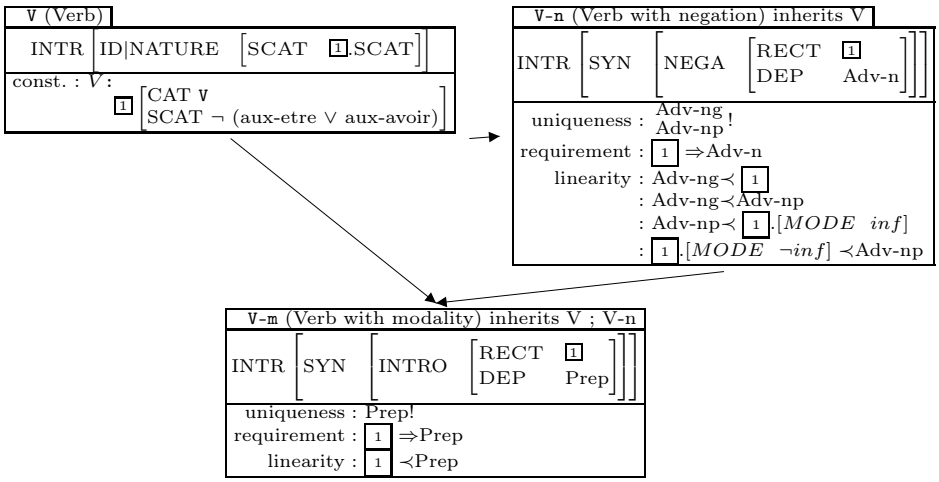


Fig. 4. Fragment of a PG for French (basic verbal constructions)

Let us for instance have a closer look at the properties of the `V-n` construction of Fig. 4. It says that in French, for verbs with a negation, this negation is made

[7] An interesting characteristic of these constraints is that they can be independently violated, and thus provide a way to characterize agrammatical sentences.

[8] Here, we omit lexical properties, such as $\mathsf{cat}(apple) = \mathtt{N}$.

[9] Note that this hierarchy is a disjunctive inheritance hierarchy, *i.e.*, when there is multiple inheritance, the subclass inherits *one* of its super-classes.

of an adverb *ne* (labelled with the category *Adv-ng*) and / or an adverb *pas* (or a related adverb such as *guère*, labelled with the category *Adv-np*). These adverbs, if they exist, are unique (*uniqueness* property), and linearly ordered (*linearity* property). When the verb is an infinitive, it comes after these adverbs (*e.g., ne pas donner (not to give)* versus *je ne donne pas (I do not give)*).

4.2 Extending XMG for PG

In order to describe PG, we need to extend the XMG formalism with linguistic constructions. These will be encapsulated within XMG's classes. As for LFG, we extend XMG with a dedicated dimension and a dedicated description language $\mathcal{L}_{PG}$. Formulas in $\mathcal{L}_{PG}$ are built as follows:

$$\begin{aligned} Desc_{PG} &:= x \mid x = y \mid x \neq y \mid [f{:}E] \mid \{P\} \mid Desc_{PG} \wedge Desc_{PG} \\ P &:= A : \triangle B \mid A : B! \mid A : B \prec C \mid A : B \Rightarrow C \mid A : B \not\Leftrightarrow C \mid A : B \end{aligned}$$

where x, y correspond to unification variables, $=$ to unification, $\neq$ to unification failure, E to some (possibly complex) expression to be associated with the feature f, and $\{P\}$ to a set of properties. Note that E and P may share unification variables. With this language, it is now possible to define the above `V`, `V-n` and `V-m` constructions as follows:

$$\begin{aligned} Vclass \rightarrow\ & [\mathsf{INTR} : [\mathsf{ID}|\mathsf{NATURE} : [\mathsf{CAT} : X.\mathsf{SCAT}]]] \wedge (V : X) \\ & \wedge (X = [\mathsf{CAT} : V, \mathsf{SCAT} : Y]) \wedge (Y \neq \texttt{aux-etre}) \wedge (Y \neq \texttt{aux-avoir}) \\ V{-}n \rightarrow\ & Vclass \wedge [\mathsf{INTR}{:}[\mathsf{SYN}{:}[\mathsf{NEGA}{:}[\mathsf{RECT}{:}X, \mathsf{DEP}{:}\texttt{Adv-n}]]]] \\ & \wedge (V : \texttt{Adv-ng!}) \wedge (V : \texttt{Adv-np!}) \wedge (V : X \Rightarrow \texttt{Adv-n}) \\ & \wedge (V : \texttt{Adv-ng} \prec X) \wedge (V : \texttt{Adv-ng} \prec \texttt{Adv-np}) \\ & \wedge (V : \texttt{Adv-ng} \prec Y) \wedge (V : Z \prec \texttt{Adv-np}) \\ & \wedge (Y = \texttt{inf}) \wedge (Y = X.\mathsf{mode}) \wedge \neg(Z = \texttt{inf}) \wedge (Z = X.\mathsf{mode}) \\ V{-}m \rightarrow\ & (Vclass \vee V{-}n) \wedge [\mathsf{INTR}{:}[\mathsf{SYN}{:}[\mathsf{INTRO}{:}[\mathsf{RECT}{:}X, \mathsf{DEP}{:}\texttt{Prep}]]]] \\ & \wedge (V : \texttt{Prep!}) \wedge (V : X \Rightarrow \texttt{Prep}) \wedge (V : X \prec \texttt{Prep}) \end{aligned}$$

Note that the disjunction operator from XMG's control language $\mathcal{L}_C$ allows us to represent [19]'s disjunctive inheritance. Also, compared with TAG and LFG, there is relatively few redundancy in PG, for redundancy is already dealt with directly at the grammar level, by organizing the constructions within an inheritance hierarchy based on linguistic motivations.

As for LFG, the metagrammar could be extended so that it solves the properties contained in the final classes, according to a sentence to parse. This could be done by adding a specific constraint solver such as that of [20] as a post-processor of the metagrammar compilation.

5 Towards an extensible metagrammatical formalism

We have seen two extensions of the XMG formalism to describe not only TAG grammars, but also LFG and PG ones, these rely on the following concepts:

- The metagrammar describes a grammar by means of conjunctive and / or disjunctive combinations of elementary units (using a combination language $\mathcal{L}_C$).
- The elementary units of the (meta)grammar depend on the target framework, and are expressed using dedicated description languages ($\mathcal{L}_D$,$\mathcal{L}_{LFG}$,$\mathcal{L}_{PG}$).

When compiling a metagrammar, the compiler executes the logic program underlying $\mathcal{L}_C$ (*i.e.*, unfolds the combination rules) while storing the elementary units of $\mathcal{L}_{D|LFG|PG}$ in dedicated accumulators. The resulting accumulated descriptions may need some additional post-processing (*e.g.*, tree description solving for TAG). Thus, to extend XMG into a cross-framework grammar engineering environment, one needs (i) to design dedicated description languages, and (ii) to develop the corresponding pre/post-processing modules (*e.g.*, metagrammar parsing / description solving).

A first version of XMG (XMG 1) was developed in Oz-Mozart.[10] It implements the language described in section 2, and supports tree-based formalisms, namely TAG and Interaction Grammar [21]. It has been used to design various large tree grammars for French, English and German.[11] The implementation of a new version of XMG (XMG 2) has started in 2010, in Prolog (with bindings to the Gecode Constraint Programming C++ library)[12], with the goal of supporting cross-framework grammar engineering as presented here.

6 Conclusion and perspectives

In this paper, we presented a metagrammatical formalism for cross-framework grammar engineering. This formalism offers a collection of description languages, making it possible to describe different types of linguistic structures (TAG's syntactic trees, LFG's functional descriptions, PG's linguistic constructions), these structures being combined either conjunctively or disjunctively via a common control language. The formalism also applies specific constraints on some of these structures to ensure their well-formedness (*e.g.*, rank principle for TAG).

Using a formalism that can describe several types of grammar frameworks offers new insights in grammar comparison and sharing. This sharing appears naturally when designing parallel grammars, but appears also when designing distinct grammars (*e.g.*, reuse of the combinations of elementary units).

The implementation of the formalism introduced here is a work in progress. We aim to provide the linguist with an extensible formalism, offering a rich collection of predefined description languages; each one with a library of principles, and constraint solvers to effect specific assembly, filtering, and verifications on the grammatical structures described by the metagrammar.

[10] See `http://sourcesup.cru.fr/xmg` and `http://www.mozart-oz.org`.

[11] These are available on line, see `http://sourcesup.cru.fr/projects/xmg` (repository METAGRAMMARS) and `http://www.sfs.uni-tuebingen.de/emmy/res-en.html`.

[12] See `https://launchpad.net/xmg` and `http://www.gecode.org`.

References

1. Joshi, A.K., Levy, L.S., Takahashi, M.: Tree adjunct grammars. Journal of the Computer and System Sciences **10** (1975) 136–163
2. Bresnan, J.: The passive in lexical theory. In Bresnan, J., ed.: The Mental Representation of Grammatical Relations. The MIT Press, Cambridge, MA (1982)
3. Pollard, C., Sag, I.: Head-Driven Phrase Structure Grammar. University of Chicago Press, Stanford : CSLI Publications, Chicago (1994)
4. King, T.H., Dipper, S., Frank, A., Kuhn, J., Maxwell, J.: Ambiguity management in grammar writing. In: Proceedings of the Workshop on Linguistic Theory and Grammar Implementation, ESSLLI 2000, Birmingham, Great-Britain (2000)
5. Cahill, A.: Parsing with Automatically Acquired, Wide-Coverage, Robust, Probabilistic LFG Approximations. PhD thesis, Dublin City University (2004)
6. Becker, T.: Patterns in Metarules for TAG. In: Tree Adjoining Grammars. Formalisms, Linguistic Analysis and Processing. CSLI, Stanford (2000)
7. Candito, M.: A Principle-Based Hierarchical Representation of LTAGs. In: Proceedings of COLING 96, Copenhagen, Denmark (1996)
8. Prolo, C.: Systematic grammar development in the XTAG project. In: Proceedings of COLING'02, Taipei, Taiwan (2002)
9. Clément, L., Kinyon, A.: Generating parallel multilingual LFG-TAG grammars from a MetaGrammar. In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL), Sapporo, Japan (2003)
10. Clément, L., Kinyon, A.: Generating LFGs with a MetaGrammar. In: Proceedings of LFG-03, Saratoga Springs, United States of America (2003)
11. Duchier, D., Le Roux, J., Parmentier, Y.: The Metagrammar Compiler: An NLP Application with a Multi-paradigm Architecture. In: Proceedings of the 2nd Oz-Mozart Conference, MOZ 2004, Charleroi, Belgium (2004)
12. Blache, P.: Constraints, Linguistic Theories and Natural Language Processing. Lecture Notes in Artificial Intelligence Vol. 1835. Springer-Verlag (2000)
13. Joshi, A.K., Schabès, Y.: Tree adjoining grammars. In Rozenberg, G., Salomaa, A., eds.: Handbook of Formal Languages. Springer Verlag, Berlin (1997)
14. XTAG Research Group: A lexicalized tree adjoining grammar for english. Technical Report IRCS-01-03, IRCS, University of Pennsylvania (2001)
15. Crabbé, B.: Représentation informatique de grammaires fortement lexicalisées : Application à la grammaire d'arbres adjoints. PhD thesis, Université Nancy 2 (2005)
16. Le Roux, J., Crabbé, B., Parmentier, Y.: A constraint driven metagrammar. In: The Eighth International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+8), Sydney, Australia (2006)
17. Blackburn, P., Gardent, C.: A Specification Language for Lexical Functional Grammars. In: Proceedings of EACL'95, Dublin, Ireland (1995)
18. Dalrymple, M., Kaplan, R., King, T.H.: Lexical structures as generalizations over descriptions. In: Proceedings of LFG 04, Christchurch, New Zealand (2004)
19. Guénot, M.L.: Éléments de grammaire du français pour une théorie descriptive et formelle de la langue. PhD thesis, Université de Provence (2006)
20. Duchier, D., Dao, T.B.H., Parmentier, Y., Lesaint, W.: Property Grammar Parsing Seen as a Constraint Optimization Problem. In: Proceedings of the 15th International Conference on Formal Grammar (FG 2010), Copenhagen, Denmark (2010)
21. Perrier, G.: Interaction Grammars. In: Proceedings of COLING 2000, Saarbrücken, Germany (2000)

Situated Propositions with Constraints and Restricted Parameters

Roussanka Loukanova

Abstract. The paper revises major situation-theoretical objects with respect to potential applications, incl. systems with logic programming. It introduces models of situated, partial, and parametric information. The system of constrained objects is defined by mutual recursion. The main contribution is the distinction between situated propositions and situated factuality of the verified propositions.

1 Introduction

In 80's, see Barwise and Perry [2], introduced Situation Theory with the ideas that partiality, factual content and situatedness are crucial features of the meaning concepts that involve mental states, incl. attitudes. Situation Theory models partiality and the inherent relational and situational nature of information, in general, not only linguistic, by diverging from the traditional possible-world semantics and type theoretic settings. One of the most distinguished applications of situation theory has been situation semantics for computational analysis of human language. One of the first practical systems with the ideas of situation theory and situation semantics is Head-driven Phrase Structure Grammar (HPSG), see Pollard and Sag [14], and Loukanova [11]. From model-theoretic point, the meta-theory of situation theory is set theory. This means that situation theory has a complex, hierarchical system of abstract objects, which are set-theoretic constructs, see Barwise and Perry [2] and Devlin [4]. Furthermore, the more powerful versions of situation theory are distinguished by representing circular pieces information, which are non-well-founded. The typical examples of such circularity involves situations that carry information about mutual belief and common knowledge shared by different agents. Such information units can be represented in situation theory by objects that do not conform with the classic axiom of foundation supporting cumulative hierarchy of sets. To accommodate such non-well-founded circularity, situation theory uses hypersets that are based on a version of non-well-founded set theory of Aczel [1]. Aczel's non-well-founded set theory replaces the foundation axiom, *FA*, of the standard axiom system *ZFC* of axiomatic set theory, with an axiom of anti-foundation, *AFA*, which was motivated by modeling non-well-founded situations in theory of processes. Applications of situation theory, for which non-well-founded objects and sets are not needed, use versions of situation theory based on standard *ZFC* of set theory. For a recent application of situation theory in linguistics, to semantic analysis of questions, see Ginzburg and Sag [6], and Lambalgen and Hamm [7]. A set-theoretic modeling of situation theory as an axiomatic system is Seligman

and Moss [16]. The model-theoretic objects in this paper presume variants of such information structures handling partial, underspecified, and parametric information. Of particular interest are applications based on logic programming, in areas that require relational structures with partiality. The presented formal system is especially useful for modeling context and resource situations that provide information about restricted parameters, see [8, 10, 9].

Partiality, underspecification, restricted parameters. Computerized information systems call for reliable, faithful representation of information. This requires theory of information which is based on information models and information inferences that do not distort information, especially when it is partial. Partiality can appear in various ways. For example, some objects have as components partially defined functions or relations. Some of these partial functions and relations may or may not be extended over some of the objects that are not in their domains, regardless of circumstances. In other cases, information is partial by missing pieces of information and components, which can be added by updates or depending on the context of usage. Parametric information is a very important kind of information, where information structure is available, but various components present as parameters that are either totally unrestricted (which is rarely the case), or vary within a broader type of objects, or are restricted to vary within a narrow domain restricted by various conditions. Naturally, such conditions are expressed by propositional constraints. Situation theory is a information theory that targets namely such goals: representation of information, which is relational and partial; it handles partially defined objects, parametric and otherwise underspecified information that are restricted to satisfy constraints and vary depending on context.

The major topic of this paper are situational objects, such as situated propositions and parameters, that can be constrained. Situation theory takes the stand that information is dependent on situations. Objects can be situation independent, but typically, propositions and their component objects are dependent on situations, and that dependence can be on more than one situation. The innovative element, contributed by the paper, is the distinction between a situated proposition, which is a semantic object that can be true or false, and the semantic object that expresses factuality of the verified proposition, by a situation that supports the informational content of the proposition. Section 2 provides an introduction to a version of situation theory, and Section 3 defines the notions of propositions and restricted parameters with constraints.

2 Fundamental Situation Theoretical Notions

This section introduces situation theoretical notions and objects that are fundamental for representation of information. Informally, the basic informational pieces, called *infons*, are composite objects carrying information about relations and objects filling the arguments of the relations, at certain time and space locations. Recursively, infons can be primitive or complex. Infons are the ground, informational content of the situated propositions (introduced in Section 3). In-

fons are facts when supported by actual situations, e.g., in real or virtual worlds, theoretical models, or computerized models. Situation theory takes some set-theoretic objects as its basic objects. These basic objects then are used in the construction of more complex situation theoretic objects.

Primitive individuals. A collection (typically, a set) $\mathcal{A}_{IND}$ is designated as the set of primitive *individuals* of the situation theory: $\mathcal{A}_{IND} = \{a, b, c, \ldots\}$ The objects in $\mathcal{A}_{IND}$ are set-theoretic objects, but they are considered as primitives, not as complex situation-theoretic constructions. In various versions of situation theory, designated for specific applications, some of the individuals in $\mathcal{A}_{IND}$ may be parts of other individuals in $\mathcal{A}_{IND}$, and as such can be in respective *part-of* relations. **Space-time locations.** Simplified versions of situation theory use a collection (typically, a set) $\mathcal{A}_{LOC}$ of space-time points and regions units: $\mathcal{A}_{LOC} = \{l, l_0, l_1, \ldots\}$. The collection $\mathcal{A}_{LOC}$ is endorsed with relations of time precedence $\prec$, time overlapping $\circ$, space overlapping @, and inclusion $\subseteq_t$, $\subseteq_s$, $\subseteq$, between locations. In some versions of situation theory, the space-tile locations can be given by complex objects. E.g., a simple option (equivalent to the above) is that space-time locations are pairs of two components, one for space locations, and one for time points or periods. **Primitive relations.** Significantly, situation theory has a collection (typically, a set) $\mathcal{A}_{REL}$ of abstract, primitive objects that are relations: $\mathcal{A}_{REL} = \{r_0, r_1, \ldots, \}$ The elements of $\mathcal{A}_{REL}$ are abstract representatives of real or virtual relations. For example, if situation theory is used to model real world situations, these are abstract representatives of properties of objects and relations between objects. **Primitive types.** A collection (typically, a relatively small set) of objects, which are called *primitive* or *basic* types: $B_{TYPE} = \{IND, LOC, REL, TYPE, POL, PAR, ARG, INFON, SIT, PROP, \models\}$, where *IND* is the type of individuals; *LOC*: of space-time locations; *REL* of relations; *TYPE*: of basic and complex types; *PAR*: of parameters; *POL*: of two polarity objects, e.g., presented by the natural numbers 0 and 1; *ARG*: of abstract argument roles (primitive and complex); *INFON*: of situation-theoretical objects that are basic or complex information units, defined later; *PROP*: of abstract objects that are propositions; *SIT*: of situations; $\models$ is a type called "supports". **Primitive parameters (indeterminates).** We consider situation theory that has a collection (a set) of primitive parameters, for each of the basic types IND, LOC, REL, POL, SIT: $\mathcal{P}_{IND} = \{\dot{a}, \dot{b}, \dot{c}, \ldots\}$, $\mathcal{P}_{LOC} = \{\dot{l}, \dot{l_0}, \dot{l_1}, \ldots\}$, $\mathcal{P}_{REL} = \{\dot{r}, \dot{r_0}, \dot{r_1}, \ldots\}$, $\mathcal{P}_{POL} = \{\dot{i}, \dot{i_0}, \dot{i_1}, \ldots\}$, $\mathcal{P}_{SIT} = \{\dot{s}, \dot{s_0}, \dot{s_1}, \ldots\}$. Informally, given a type T (primitive or complex) and an object Θ, we write[1]: $T\colon \Theta$ iff Θ is of type T. **Primitive argument roles with and without appropriateness constraints.** Each relation and each type is associated with a set of argument roles. We consider situation theory that has a collection (a set) of primitive objects designated as *primitive argument roles.*: $\mathcal{A}_{ARG} = \{arg_1, \ldots, arg_n, \ldots\}$ Since each primitive relation has its own argument roles, it may be more acceptable to model them by abstract primitive objects that are uniquely distinctive for each primitive relation and type.

[1] An alternative notation resembles classic type systems, such as Montague's IL: for any given type T and any object Θ, $\Theta\colon T$ iff Θ is of type T.

Definition 1 (Assignment of primitive argument roles). *A set of argument roles is assigned to each of the primitive relations and each of the primitive types by a function Args with the domain and range of which are such that:* $Dom(Args) = \mathcal{A}_{REL} \cup B_{TYPE}$, *and* $Range(Args) \subseteq \mathcal{A}_{ARG}$.

For example, we can associate relations, such as *smile*, *read*, *give*, respectively denoted by the lexemes smile, read, give, etc., with arguments roles:[2] $Args(smile) = \{smiler\}$, $Args(read) = \{reader, read\text{-}ed\}$, $Args(give) = \{giver, receiver, given\}$. Another option is to use a common set of shared primitive objects for argument roles: $\mathcal{A}_{ARG} = \{arg_1, \ldots, arg_n, \ldots\}$. We can use as many argument roles as needed, e.g.: $Args(give) = \{arg_1, arg_2, arg_3\}$ Note that there is no implicit order over the argument roles. Which role is for what in a relation depends on the actual modeling[3] of the relations and their arguments in the abstract theoretic constructions. Similarly to relations, each type is associated with a set of argument roles. If a type T has a single argument role, we call it a *unary type*, or a *property type*. In particular, *IND*, *LOC*, *POL*, *PAR*, *TYPE*, are unary types, each with one argument role, that can be declared as filled only by elements of the corresponding sets:

$$IND\colon \xi, \text{ for each } \xi \in \mathcal{A}_{IND} \cup \mathcal{P}_{IND} \tag{1a}$$
$$LOC\colon \xi, \text{ for each } \xi \in \mathcal{A}_{LOC} \cup \mathcal{P}_{LOC} \tag{1b}$$
$$REL\colon \xi, \text{ for each } \xi \in \mathcal{A}_{REL} \cup \mathcal{P}_{REL} \tag{1c}$$
$$\text{and for each complex relation } \xi \text{ (introduced later)}$$
$$POL\colon \xi, \text{ for each } \xi \in \{0, 1\} \cup \mathcal{P}_{POL} \tag{1d}$$
$$PAR\colon \xi, \text{ for each } \xi \in \mathcal{P}_{IND} \cup \mathcal{P}_{LOC} \cup \mathcal{P}_{REL} \cup \mathcal{P}_{POL} \cup \mathcal{P}_{SIT} \tag{1e}$$
$$\text{and for each complex parameter } \xi \text{ (introduced later)}$$
$$TYPE\colon \xi, \text{ for each } \xi \in B_{TYPE} \tag{1f}$$
$$\text{and for each complex type } \xi \text{ (introduced later)}$$

The argument roles of both relations and types can be associated with types as constraints for their appropriate filling.

Definition 2 (Argument roles with appropriateness constraints). *A set of argument roles is assigned to each of the primitive relations and to each of the primitive types by a function Args with its domain and range of values such that:* $Dom(Args) = (\mathcal{A}_{REL} \cup B_{TYPE})$, *and* $Range(Args) \subseteq (\mathcal{A}_{ARG} \times TYPE)$, *so that for any primitive relation and any type* $\gamma \in \mathcal{A}_{REL} \cup B_{TYPE}$ *with n-arguments:* $Args(\gamma) = \{\langle arg_{i_1}, T_{i_1}\rangle, \ldots, \langle arg_{i_n}, T_{i_n}\rangle\}$, *where* $T_1, \ldots, T_n$ *are sets of types (basic or complex), which are specific for* γ *and are called* basic appropriateness constraints of the argument roles *of* γ.

[2] In what follows, we shall follow a practice of naming the argument role of the object that is read, by the "misspelled" notations *read-ed* and *readed*.

[3] Another option, "intermediate" between the above two, is to accept a relatively small set of common, abstract roles, which are similar to those in Θ-theory of Government and Binding Theory (GBT).

Often, we shall use the notation: $Args(\gamma) = \{T_{i_1} : arg_{i_1}, \ldots, T_{i_n} : arg_{i_n}\}$. For example, $Args(give) = \{IND : giver, IND : receiver, IND : given\}$. For applications, more complex constraints for appropriateness of arguments can be added and expressed by complex types, $T_{i_1}, \ldots, T_{i_n}$, such that $TYPE : T_{i_1}, \ldots, TYPE : T_{i_n}$. For any relation or type (which can be primitive or complex), the objects that fill its argument roles are restricted to satisfy the constraints associated with the roles.

Definition 3 (Argument filling). *For any given relation $\gamma \in \mathcal{R}_{REL}$ and for any given type $\gamma \in \mathcal{T}_{type}$ associated with the set of argument roles $Args(\gamma) = \{T_{i_1} : arg_{i_1}, \ldots, T_{i_n} : arg_{i_n}\}$, an* argument filling *for γ is any total function θ with $\mathrm{Dom}(\gamma) = \{arg_{i_1}, \ldots, arg_{i_n}\}$, which is set-theoretically defined by a set of ordered pairs $\theta = \{\langle arg_{i_1}, \xi_1\rangle \ldots, \langle arg_{i_n}, \xi_n\rangle\}$, so that its values, $\theta(arg_{i_1}) = \xi_1$, $\ldots$, $\theta(arg_{i_n}) = \xi_n$, satisfy the appropriateness constraints of the argument roles of γ: $T_{i_1} : \xi_1, \ldots, T_{i_n} : \xi_n$.*

Infons, State of Affairs (soas), Situations. Next, I shall give a mutually recursive definition of several sets of situational objects:

- the set $\mathcal{I}_{INF}$, the elements of which are called infons, and are basic or complex information units;
- the set $\mathcal{R}_{REL}$ of all primitive and complex relations (complex relations are defined later): $\mathcal{A}_{REL} \subset \mathcal{R}_{REL}$;
- the set $\mathcal{T}_{TYPE}$ of all primitive and complex types: $B_{TYPE} \subset \mathcal{T}_{TYPE}$;
- the collection $\mathcal{S}_{SIT}$ of situations.

The basic informational units are identified by a unique relation, an assignment of its argument roles and a corresponding negative or positive polarity.

Definition 4 (Infons). *The set $\mathcal{I}_{\mathrm{INF}}$ of all infons:*

1. Basic infon *is every tuple $\langle \gamma, \theta, \tau, i\rangle$, where $\gamma \in \mathcal{R}_{REL}$ is a relation (primitive or complex), $LOC : \tau$, $POL : i$, and θ is an argument filling for γ, i.e.: $\theta = \{\langle arg_{i_1}, \xi_1\rangle, \ldots, \langle arg_{i_n}, \xi_n\rangle\}$, for some situation-theoretical objects $\xi_1, \ldots, \xi_n$ satisfying the appropriateness constraints of γ.*
2. *Let $\mathcal{BI}_{\mathrm{INF}}$ be the set of all basic infons. $\mathcal{BI}_{\mathrm{INF}} \subset \mathcal{I}_{\mathrm{INF}}$.*
3. Complex infons *(for representation of conjunctive and disjunctive information) are formed by the operators (i.e., primitive relations, for which locations are irrelevant) conjunction and disjunction:*
   For any infons $\sigma_1, \sigma_2 \in \mathcal{I}_{\mathrm{INF}}$, $\langle \wedge, \sigma_1, \sigma_2\rangle \in \mathcal{I}_{\mathrm{INF}}$ and $\langle \vee, \sigma_1, \sigma_2\rangle \in \mathcal{I}_{\mathrm{INF}}$.

In this paper, we adopt the traditional linear notations of the basic infons:

$$\ll \gamma, arg_{i_1} : \xi_1, \ldots, arg_{i_n} : \xi_n, LOC : \tau; i \gg \tag{2a}$$

$$\ll \gamma, \xi_1, \ldots, \xi_n, \tau; i \gg \tag{2b}$$

The notation (2a) does not assume any order over the argument roles of γ. On the other hand, in case that γ has more than one argument roles, the notation (2b) makes sense only by having some agreement about an order over the argument

roles of γ. Complex infons constructed by Definition 4.3 represent conjunctive and disjunctive pieces of information. hey are denoted usually by $(\sigma_1 \wedge \sigma_2)$ and $(\sigma_1 \vee \sigma_2)$, sometimes without parentheses, without causing confusion.

Definition 5 (States of affairs, events, situations).

1. State of affairs (soa) *is any set of infons that have the same location component.*
2. *An* event (course of event, coa) *is any set of infons.*
3. *A* situation *is any set of infons.*

Infons, states of affairs, and situations, in which some of the argument roles, including the space-time location and polarity components, are filled by parameters, are called, respectively, *parametric infons*, *parametric soas*, and *parametric situations*. For example:

$$\ll read, reader: a, readed: b, l; 1 \gg \tag{3a}$$

$$\ll read, reader: \dot{a}, readed: \dot{b}, \dot{l}; 1 \gg \tag{3b}$$

$$\ll read, reader: a, readed: \dot{b}, \dot{l}; 1 \gg \tag{3c}$$

$$\ll read, a, b, l; \dot{i} \gg \tag{3d}$$

3 Situated Propositions, Constraints and Parameters

In the considered version of situation theory, I add a specialized primitive type $PROP \in B_{TYPE}$, with two argument roles: a type $\mathbb{T} \in \mathcal{T}_{TYPE}$, and an appropriate argument filling θ for $\mathbb{T}$. I will use the type $PROP$ for constructing abstract objects (set-theoretic tuples) to model the abstract notion of a proposition, which states that some given objects are of some given type, in the following way:

Definition 6 (Propositions). Proposition *is any tuple* $\langle PROP, \mathbb{T}, \theta \rangle$ *where* $\mathbb{T} \in \mathcal{T}_{TYPE}$ *is a type that is associated with a set of argument roles*

$$Args(\mathbb{T}) = \{T_{i_1}: arg_{i_1}, \ldots, T_{i_n}: arg_{i_n}\} \tag{4}$$

and θ *is an argument filling for* $\mathbb{T}$*, i.e.:* $\theta = \{\langle arg_{i_1}, \xi_1 \rangle, \ldots, \langle arg_{i_n}, \xi_n \rangle\}$*, for some objects* $\xi_1, \ldots, \xi_n$ *that satisfy the appropriateness constraints of* $\mathbb{T}$*:*

$$T_{i_1}: \xi_1, \ldots, T_{i_n}: \xi_n. \tag{5}$$

We use the notation $(\mathbb{T}, \theta)$ for $\langle PROP, \mathbb{T}, \theta \rangle$. When $\langle PROP, \mathbb{T}, \theta \rangle$ is true (see later), we say that the objects $\xi_1, \ldots, \xi_n$ are of type $\mathbb{T}$ with respect to the argument role filling θ, and we write $\mathbb{T}: \theta$, or $\mathbb{T}: \xi_1, \ldots, \xi_n$, in case it is clear which roles are filled by which objects. I.e., propositions are the result of filling up the argument roles of a type with appropriate objects. Now, we shall concentrate on a special kind of propositions defined by the following Definition 7.

Definition 7 (Situated propositions).

1. *The type "support", $\models$, has two argument roles, one that can be filled by any object that is of the type SIT of situations, and the other can be filled by any object that is of the type INF of inforns:*
$$Args(\models) = \{\langle arg_{sit}, SIT\rangle, \langle arg_{infon}, INF\rangle\}, \tag{6}$$
i.e.,
$$Args(\models) = \{SIT\colon arg_{sit}, INF\colon arg_{infon}\}. \tag{7}$$
2. Situated proposition *is any proposition $\langle PROP, \models, s, \sigma\rangle$, where $s \in \mathcal{P}_{SIT}$ and $\sigma \in \mathcal{I}_{INF}$.*
3. *Often, we shall use the notation $(s \models \sigma)$ and say "the proposition that σ holds in s" or "the proposition that the situation s supports the infon σ".*

$$(s \models \ll read, reader\colon x, readed\colon b, Loc\colon l; 1 \gg \wedge \tag{8a}$$
$$\ll book, arg\colon b, Loc\colon l; 1 \gg) \tag{8b}$$

Situation theory uses an abstraction operator, which recalls the λ-abstraction in functional λ-calculi, but, in situation theory, abstraction operator is different and does not define functions. In situation theory, the abstraction operator results in complex types, with abstract argument roles.

Definition 8 (Complex appropriateness constraints and complex types). *Let Θ be a given proposition and $\{\xi_1, \ldots, \xi_n\}$ be a set of parameters that occur in Θ. Let, for each $i \in \{1, \ldots, n\}$, T_i be the union of all the appropriateness constraints of all the argument roles that occur in Θ and ξ_i fills up.*

Then the object $\lambda\{\xi_1, \ldots, \xi_n\}\,\Theta \in \mathcal{T}_{TYPE}$, i.e., $\lambda\{\xi_1, \ldots, \xi_n\}\Theta$ is a complex type*, with abstract argument roles denoted by $[\xi_1], \ldots, [\xi_n]$ and corresponding* appropriateness constraints *associated in the following way:*
$$Args(\lambda\{\xi_1, \ldots, \xi_n\}\Theta) = \{T_1\colon [\xi_1], \ldots, T_n\colon [\xi_n]\}. \tag{9}$$

The type $\lambda\{\xi_1, \ldots, \xi_n\}\Theta$ is alternatively denoted by
$$[\xi_1, \ldots, \xi_n/\Theta(\xi)] \quad \text{or} \quad [T_1\colon \xi_1, \ldots, T_n\colon \xi_n/\Theta(\xi)]. \tag{10}$$

Example 1. For example, (11a) is the type of situations and locations where the specific individual a walks; (11b) is the type of individuals that walk in a specific situation s and a specific location l; (11d) is the type of individuals that read a specific book b, in a specific situation s and a specific location l; (11d) is the type of situations, locations and individuals, where the individual reads a specific book b:

$$\lambda\dot{s}, \dot{l}\,(\dot{s} \models \ll walk, walker\colon a, Loc\colon \dot{l}; 1 \gg) \tag{11a}$$
$$\lambda x\,(s \models \ll walk, walker\colon x, Loc\colon l; 1 \gg) \tag{11b}$$
$$\lambda x\,(s \models \ll read, reader\colon x, readed\colon b, Loc\colon l; 1 \gg \wedge \tag{11c}$$
$$\ll book, arg\colon b, Loc\colon l; 1 \gg)$$
$$\lambda\dot{s}, \dot{l}, x\,(\dot{s} \models \ll read, reader\colon x, readed\colon b, Loc\colon \dot{l}; 1 \gg \wedge \tag{11d}$$
$$\ll book, arg\colon b, Loc\colon \dot{l}; 1 \gg)$$

Notation. Given object α and a set of appropriateness constraints T, we write $T\colon \alpha$ just in case α satisfy all the constraints in T.

Property 1. Let Θ be a given proposition and $\{\xi_1, \ldots, \xi_n\}$ be a set of parameters that occur in Θ. Let, for each $i \in \{1, \ldots, n\}$, T_i be the union of all the appropriateness constraints of all the argument roles that occur in Θ and ξ_i fills up. Given that $\alpha_1, \ldots, \alpha_n$ are objects that satisfy appropriateness constraints $T_1\colon \alpha_1, \ldots, T_n\colon \alpha_n$, we have:

1. by Definition 8, $\lambda\{\xi_1, \ldots, \xi_n\}\,\Theta \in \mathcal{T}_{TYPE}$ is a *type*, with argument roles $Args(\lambda\{\xi_1, \ldots, \xi_n\}\Theta) = \{T_1\colon [\xi_1], \ldots, T_n\colon [\xi_n]\}$.
2. Let θ be the total function that is set-theoretically defined by the set of ordered pairs $\theta = \{\langle [\xi_1], \alpha_1\rangle \ldots, \langle [\xi_n], \alpha_n\rangle\}$,
   (a) by Definition 3, θ is an argument filling for the type $\lambda\{\xi_1, \ldots, \xi_n\}\,\Theta$.
   (b) by Definition 6, $(\lambda\{\xi_1, \ldots, \xi_n\}\,\Theta\colon \theta)$ is a proposition that the objects of the filling θ are of the complex type $\lambda\{\xi_1, \ldots, \xi_n\}\,\Theta$, i.e.:

$$(\lambda\{\xi_1, \ldots, \xi_n\}\,\Theta\colon \theta) \equiv \langle PROP, \lambda\{\xi_1, \ldots, \xi_n\}\,\Theta, \theta\rangle$$

□

Abstractions over individuals in propositions result in *complex types of individuals*. In general, for any given proposition Θ and a parameter ξ for an individual, i.e., $IND\colon \xi$, which occurs in Θ, the complex type $\lambda\{\xi_1\}\,\Theta \in \mathcal{T}_{TYPE}$; is the type of the individuals for which the proposition $\Theta(\xi_1)$ is true.

Definition 9 (Denotational truth assignment for complex propositions). *Let s be either a situation or a parameter for a situation, and σ be an infon (possibly parametric). The proposition $(s \models \sigma)$ is true iff there is some assignment c of objects to the parameters in $(s \models \sigma)$, such that $c(s) \models \sigma(c)$.*

Definition 10 (Denotational truth assignment for complex propositions). *Given a complex type $\lambda\{\xi_1, \ldots, \xi_n\}\,\Theta \in \mathcal{T}_{TYPE}$ and an argument filling for it,*

$$\theta = \{\langle [\xi_1], \alpha_1\rangle \ldots, \langle [\xi_n], \alpha_n\rangle\},$$

Then, the proposition $\langle PROP, \lambda\{\xi_1, \ldots, \xi_n\}\,\Theta, \{\langle [\xi_1], \alpha_1\rangle \ldots, \langle [\xi_n], \alpha_n\rangle\}\rangle$ is true iff the proposition $\Theta[\xi_1 :\equiv \alpha_1, \ldots, \xi_n :\equiv \alpha_n]$ is true where $\Theta[\xi_1 :\equiv \alpha_1, \ldots, \xi_n :\equiv \alpha_n]$ is obtained from Θ by simultaneous replacement of all occurrences of ξ_1 with α_1, …, ξ_n with α_n.

A particular case of the Definition 10 yields:

Property 2. Given a parameter $PAR\colon \xi$, the proposition $([\xi/\Theta] : \alpha)$ is true iff the proposition $\Theta[\xi :\equiv \alpha]$ is true.

However, neither the types of individuals, nor propositions $([\xi/\Theta] : \alpha)$ represent per se individuals α constrained to satisfy the proposition Θ. But these complex types are used in the definition of restricted parameters, which stand for (e.g., by denoting) individuals constrained to satisfy conditions. The following is part of the mutual recursion that defines the system of situational objects:

Definition 11 (Restricted parameters). *Let $\Theta(\xi)$ be a proposition, and T be the set of all the appropriateness constraints of all the argument roles in $\Theta(\xi)$ that are filled by ξ*[4]*.*

1. *If $\tau : x$ is a parameter of type τ, and τ is compatible with the set T of constraints, then $x^{\lambda\xi\,\Theta(\xi)}$ is also a parameter of type τ, which is called parameter restricted by $\lambda\xi\,\Theta(\xi)$.*
   With the alternative denotation of the complex type $[\xi/\Theta(\xi)]$, the restricted parameter is denoted by $x^{[\xi/\Theta(\xi)]}$.
2. *An assignment of situation-theoretic objects to the parameters of $\Theta(\xi)$ defines a substitution function c over a situation theoretic object $\gamma(x^{[\xi/\Theta(\xi)]})$ iff the proposition $c(\Theta(x))$ is true.*

The restrictions over parameters have a pressuppositional effect on the parameter assignment and argument fillings: The restricted parameters are the central objects in the formal definition of general contexts, and in particular, of linguistic contexts, discourse situations, and resource situations, see Loukanova [10, 9].

Definition 12 (Coherency and Compatibility). *A situation s (parametric or not) is* coherent *iff*

1. *no infon and its dual are supported by s, i.e., it is not the case that both*

$$s \models \ll \gamma, \xi_1, \ldots, \xi_n, \tau; 0 \gg \tag{12a}$$

$$s \models \ll \gamma, \xi_1, \ldots, \xi_n, \tau; 1 \gg \tag{12b}$$

2. *Identity of indiscernibles: for any situation-theoretical objects ξ_1 and ξ_2,*

$$\text{if } s \models \ll \mathit{same}, \xi_1, \xi_2, \tau; 0 \gg \text{ then } \xi_1 = \xi_2 \tag{13}$$

3. *no violation of Indiscernibility of identicals, i.e.,*

$$s \models \ll \mathit{same}, \xi, \xi, \tau; 0 \gg, \text{for no object } \xi \tag{14}$$

4. *Two propositions $(s_1 \models \sigma_1)$ and $(s_2 \models \sigma_2)$ are compatible iff there is a common assignment of objects to the parameters that occur in them, such that $s_1 \models \sigma_1$ and $s_2 \models \sigma_2$*
5. *Let Θ_1 and Θ_2 be situated propositions. Then the types $\lambda\xi_1\,\Theta_1$ and $\lambda\xi_2\,\Theta_2$ are compatible iff the propositions Θ_1 and Θ_2 are compatible.*

4 Conclusions and Outlook

Situation theory models information that is typically situated, i.e., informational units are provided by world parts that we call situations. Technically, most versions of situation theory are many sorted model theories of structured

[4] I.e., ξ is a parameter that satisfies all the constraints in T: $T : \xi$. Then $\lambda\xi\,\Theta(\xi)$ is a type.

information. It has many applications, in different stages of development, and is promising for new ones with the demands of contemporary technologies for handling partial situations, states, context and agents. I conclude with topics that are in the focus of upcoming work.

Underspecification. Initially, some grammar formalisms took the stand that all ambiguities in human language expressions should be casted out explicitly, either syntactically, as in Montague's PTQ (see Montague [17]) or by some other means, so that the process of parsing would render all alternative readings. With the advance of language processing, it is already clear that in most cases, except for some special purposes, it is not necessary for syntactic analyses to render all the multiple "readings" (interpretations) straight away, especially in the absence of sufficient information. However, language analysis is more useful and adequate when it provides syntactic structure with corresponding rendering into semantic representation. Such an analysis should provide all semantic information that is explicitly carried by the analysed expression, which is the basis for obtaining possible interpretations, when relevant context information becomes available. This leads to two related tasks: underspecified semantic representation and modeling contexts.

Modeling different kinds of semantic ambiguities is one of the major ideas of situation theory, as well as modeling partiality of information, parametric information, parameters with constraints, context dependency of information, modeling context, and deriving specific interpretations from the parametric meaning in given contexts. This makes situation theory a natural model theory for semantics of various languages, especially of human languages. Situation semantics of human language was the first such an application of situation theory. However, both situation theory and situation semantics are in need of further development to meet the advances of technologies. The topic of this paper is an initiation in this direction. Situation theory supports partial and parametric theoretic objects (presented in this paper), which can model context (see Barwise and Perry [2], Devlin [4], and [8, 10, 9]), which is necessary for an adequate theory of meaning. Currently, development of situation theory as a general theory of information, including computational semantics, is largely open work.

Relational vs. functional structures. Relational models, like situation theory, are more direct and natural for applications where the domains of objects are relational and include multiargument predication. Something more, relational models are actually necessary with respect to adequateness, when the domains include partial relations between objects. This is so because there may not be faithful modeling of partial relations with functional encoding like the currying in classic model-theoretic logics, with possible worlds, such as those in the class of Gallin [5] and Montague [17] On the other hand, the seminal works of Gallin and Montague continue to be important, firstly for the fundamental techniques they introduced for computational semantics. Secondly, there are type-theoretic models and applications, which are more naturally based on developments of functional type-logic systems, because the domains they represent are originally and naturally consisting of functional objects. This is why

works on rectifying and extending the class of Gallin and Montague formal systems are important for theoretical foundations of computation and semantics, and for applications.

A very important current development of theory of computational systems, with potentials for various applications, is type theory of recursion by Moschovakis [12, 13].

Functional type theory is necessary for applications and computer systems that are based on functional domains and functional programming. At the contemporary stage of advancements in computer and other technologies, it is important that there are computerized systems that are capable of handling either both functional and relational structures, or are linked to systems that transfer between two modes. This is especially important for systems that involve human language processing, either by being solely human language processing system, or by using such as a part of its functionality.

There are language processing systems for grammar development, which are based on relational formalisms, in particular for semantic representation. For example, LKB grammar system[5] uses unification-based linguistic formalization and can accommodate semantic representation that is either functional or strictly relational. LKB can be used for writing CBLG of human languages, incl. categorial grammars. LKB comes with possibility for semantic representation in grammatical structures, by using Minimal Recursion Semantics (MRS), see Copestake et al. [3]. MRS is a version of situation semantics. In grammars written by LKB, when the grammar rules and lexicon include semantic representations in MRS, by parsing a language expression A, LKB outputs the feature structure analysis of A, which includes semantic representation of A in MRS. In addition, the feature structure of A renders a translation of the MRS representation into a Prolog term. This makes LKB a potential system for linking it with other systems that handle Prolog. On the other hand, there are grammar systems, that are in the category of functional approaches. For example, GF Grammatical Framework, see Ranta [15], is functional language for grammar writing, based on categorial grammar formalism, with dependent types that are naturally functional. A formal language, e.g., a version of the languages of recursion by Moschovakis [13], can lend naturally for semantic representation in GF grammars. Suitable versions of situation theory, may be employed in GF as well, by development standing at Moschovakis recursion.

Information exchange and transfer. Computerized systems, both in sciences and in daily applications, rely on representation of information, which is usually structured depending on the domain of application. Such practical systems, as well as theoretical models in sciences, or interdisciplinary developments of computational systems in sciences, vary with respect to natural and artificial languages that are used in them for information representation. Automatic exchange and transfer of information within and between computational systems should be without loss and distortion of information. This poses a renewed need of new approach to theory of information that is language independent. This pa-

[5] See <http://www.delph-in.net/lkb/>.

per has introduced a preview of situation theory, which is a relational approach to modeling partial information.

References

1. P. Aczel. *Non-well-founded Sets*, volume 14 of *CSLI Lecture Notes*. CSLI Publications, Stanford, California, 1988.
2. J. Barwise and J. Perry. *Situations and Attitudes*. Cambridge, MA:MIT press, 1983.
3. A. Copestake, D. Flickinger, C. Pollard, and I. Sag. Minimal recursion semantics: an introduction. *Research on Language and Computation*, 3:281–332, 2005.
4. K. Devlin. Situation theory and situation semantics. In D. Gabbay and J. Woods, editors, *Handbook of the History of Logic*, volume 7, pages 601–664. Elsevier, 2008.
5. D. Gallin. *Intensional and Higher-Order Modal Logic*. North-Holland, 1975.
6. J. Ginzburg and I. A. Sag. *Interrogative Investigations: The Form, Meaning, and Use of English Interrogatives*. CSLI Publications, Stanford, California, 2000.
7. M. V. Lambalgen and F. Hamm. *The Proper Treatment Of Events*. Wiley-Blackwell, Oxford, 2004.
8. R. Loukanova. Russellian and strawsonian definite descriptions in situation semantics. In A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 2004 of *Lecture Notes in Computer Science*, pages 69–79. Springer Berlin / Heidelberg, 2001.
9. R. Loukanova. Generalized quantification in situation semantics. In A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 2276 of *Lecture Notes in Computer Science*, pages 46–57. Springer Berlin / Heidelberg, 2002.
10. R. Loukanova. Quantification and intensionality in situation semantics. In A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 2276 of *Lecture Notes in Computer Science*, pages 32–45. Springer Berlin / Heidelberg, 2002.
11. R. Loukanova. An approach to functional formal models of constraint-based lexicalist grammar (CBLG). *Fundamenta Informaticae. Journal of European Association for Theoretical Computer Science (EATCS)*, to appear.
12. Y. N. Moschovakis. Sense and denotation as algorithm and value. In J. Oikkonen and J. Vaananen, editors, *Lecture Notes in Logic*, number 2 in Lecture Notes in Logic, pages 210–249. Springer, 1994.
13. Y. N. Moschovakis. A logical calculus of meaning and synonymy. *Linguistics and Philosophy*, 29:27–89, 2006.
14. C. Pollard and I. A. Sag. *Information-Based Syntax and Semantics, Part I*. Number 13 in CSLI Lecture Notes. CSLI Publications, 1987.
15. A. Ranta. *Grammatical Framework: Programming with Multilingual Grammars*. CSLI Publications, Stanford, 2011. ISBN-10: 1-57586-626-9 (Paper), 1-57586-627-7 (Cloth).
16. J. Seligman and L. S. Moss. Situation theory. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 239–307. Elsevier, Amsterdam, 1996.
17. R. H. Thomason, editor. *Formal Philosophy: Selected Papers of Richard Montague*, ed. Richmond Thomason. Yale University Press, New Haven, Connecticut, 1974. Edited, with an introduction, by Richmond H. Thomason.

Pairing Model-Theoretic Syntax and Semantic Network for Writing Assistance

Jean-Philippe Prost and Mathieu Lafourcade

LIRMM – 161, rue Ada – 34095 Montpellier Cedex 5 – France
{Prost,Lafourcade}@lirmm.fr

Abstract. In this paper we investigate the possibility of a syntax–semantics inferface between a framework for Model-Theoretic Syntax on one hand and a semantic network on the other hand. We focus on exploring the ability of such a pairing to solve a collection of grammar checking problems, with an emphasis on cases of missing words. We discuss a solution where constraint violations are interpreted as grammar errors and yield the re-generation of new candidate parses (partially unrealised) through tree operations. Follows a surface realisation phase, where missing words are filled through semantic network exploration.

1 Introduction

Model-Theoretic Syntax (MTS) refers to a family of frameworks for formal syntax, which is grounded in the Model Theory — unlike Generative-Enumerative Syntax (GES), such as the Chomskyan syntax, which originates from Proof Theory. In very general terms, an MTS framework considers that a (given) syntax structure is a model of the grammar $\mathcal{G}$, seen as a set of independent logical statements, if and only if it meets every statement in $\mathcal{G}$. For what we are interested in in this paper, it is important to note that: (i) the two problems of model generation and model recognition are kept separate, and (ii) every grammar statement may be evaluated independently from the rest of the grammar.

Many of the differences between the two families with respect to linguistic knowledge representation have been discussed by Pullum and Scholz ([1]). The ability of MTS, unlike GES, to represent linguistic information about quasi-expressions[1], that is, utterances which present grammar irregularities, is the main point of interest here. As we are going to show it, that linguistic knowledge which we have about the syntax of quasi-expressions directly serves the purpose of error detection and correction.

Meanwhile, a drawback of MTS for phrase structure grammar is a lack of syntax–semantics interface, which, beyond compositionality, would take advantage of the fine-grained information made available next to the phrase structure. To the best of our knowledge the work from Dahl and Gu ([2]) is the only exception. It extends Property Grammar (Blache [3]) with semantic predicates, which further constrain the specification of categories as any other assertion.

[1] Sometimes also referred to as *non-canonical input* in the literature.

Those semantic predicates essentially introduce different kinds of semantic relationships among lexical items. Those are mostly domain-specific, derived from a biomedical ontology. A semantic form is built through the parsing process by compositionality. Meanwhile, there seems to be no attempt to take advantage of the linguistic knowledge embodied in every property satisfied or violated by the parse tree.

In this paper, we address the same problem of the syntax–semantics interface for Property Grammar (PG) but from a different angle; we investigate the possibility to pair an MTS framework with a semantic network. We explore, especially, how such a pairing can serve the purpose of grammar error correction, and focus on cases of missing words. We show how, beyond compositionality, the constraints can interact with the phrase structure to build a more detailed semantic representation than with the phrase structure alone. Starting from an approximated parse for a quasi-expression missing a word, the process operates specific tree transformations according to the detected error, in order to generate a new set of candidate corrected trees. The characterisation of those candidate models let us build the messages with which the semantic network is then queried, in search for the missing word.

In section 2 we present briefly the theoretical background we are working with; in section 3 we detail how we adapt that theoretical framework to the semantic roles required by the semantic network; then in section 4 we present how an approximated parse is turned into a candidate corrected parse by regeneration; finally, section 5 describes how to explore a semantic network in order to fill missing words.

2 Property Grammar

The framework we are using for knowledge representation is Property Grammar (Blache [3])[2] (PG), for which a model-theoretical semantics was axiomatised by Duchier et al. ([5]). Intuitively, a PG grammar decomposes what would be rewrite rules of a generative grammar into atomic syntactic properties — a *property* being represented as a boolean constraint. Take, for instance, the rewrite rule $\texttt{NP} \rightarrow \texttt{D N}$. That rule implicitly informs[3] on different properties (for French): (1) NP has a D child; (2) the D child is unique; (3) NP has an N child; (4) the N child is unique; (5) the D child precedes the N child; (6) the N child requires the D child. PG defines a set of axioms, each axiom corresponding to a constraint type. The properties above are then specified in the grammar as the following constraints: (1) $\texttt{NP} : \triangle\texttt{D}$; (2) $\texttt{NP} : \texttt{D}!$; (3) $\texttt{NP} : \triangle\texttt{N}$; (4) $\texttt{NP} : \texttt{N}!$; (5) $\texttt{NP} : \texttt{D} \prec \texttt{N}$; (6) $\texttt{NP} : \texttt{N} \Rightarrow \texttt{D}$. A PG grammar is traditionally presented as a collection of Categories (or Constructions), each of them being specified by a set of constraints. Table 1 shows an example of categories.

[2] Property Grammars closely follows from the 5P Paradigm introduced by Bès and Blache ([4]).

[3] The rule is assumed to be the only one for NP.

These constraints can be independently verified, hence independently satisfied or violated. The parsing problem is, thus, a Constraint Satisfaction Problem (CSP), where the grammar is the constraint system to be satisfied. In the Model-Theoretic (MT) axiomatisation the class of models we are working with is made up of trees labelled with categories, whose surface realisations are the sentences σ of the language. A syntax tree of realisation the expression (i.e well-formed sentence) σ is a *strong* model for the PG grammar $\mathcal{G}$ iff it satisfies every pertinent constraint[4] in $\mathcal{G}$.

The loose semantics also allows for constraints to be relaxed. Informally, a syntax tree of realisation the quasi-expression (i.e. ill-sentence) σ is a *loose* model for $\mathcal{G}$ iff it maximises the proportion of satisfied constraints in $\mathcal{G}$ with respect to the total number of pertinent ones (i.e. evaluated) for a given category. Such a loose model is called an *approximated parse*.

The set of all the satisfied constraints and all the violated ones for a model $\mathcal{M}$ is called the model's *characterisation*. It provides fine-grained information about every node in the model, which complements usefully the sole phrase structure. The violated constraints, especially, naturally point out grammar errors.

On the downside, although the formal model is quite elegant in practice the size of the search space makes the parsing problem blow up exponentially. As emphasized by Duchier et al. ([6]), who implemented a parser as a genuine CSP based on the MT axiomatisation, the practical issue is not so much finding the best model (for which existing constraint programming techniques are quite efficient) as proving its optimality. The reason for that comes from the need to explore the entire class of models in order to address the decision problem. Yet, such an implementation does actually keep separate the generation of models from their checking, which opens the door to different, and more efficient, perspectives.

One of them is to reduce the search space to the subset of models, which are statistically the most significant. It can easily be achieved by any stochastic robust parser, or even a combination of them. That subset can even be completed with models for which the statistical significance is unknown, but which were generated by symbolic parser according to linguistic judgements. The Sygfran parser (Chauché [7]), for instance, is one of the latter. Provided such a small subset as search space the combinatorial explosion is avoided. Each model can, then, easily be completed with its characterisation. Incidently, it is interesting to notice that such an architecture makes it possible to overcome an important decision problem met by statistical robust parsing with respect to the grammaticality of sentence, and despite the fact that a parse tree is generated for it. That problem makes authors such as Wagner et al. ([8]) or Wong and Dras ([9]), among others, say that those robust parsers are *too* robust. Once the parses are characterised the decision about the grammaticality of a sentence is straightforward.

[4] As defined by Duchier et al. ([5]), a *pertinent constraint* is an instance of a property, which verifies the *pertinence predicate* $\mathcal{P}_\tau$. Intuitively, an instance of a property (or *constraint*) is pertinent at a node if the node's category and those of its children are in use in the property's definition; the constraint is otherwise trivially satisfied.

Another incentive is that it makes it a framework for comparing different parsers' outcomes over the same corpus. This is particularly interesting when it comes to quasi-expressions, for which there exists neither a linguistic theory nor an empirical consensus as to what an approximated parse should be. Further works should consider running such a comparison and addressing the question of parse quality, more especially on a corpus with a large number of ill-sentences.

3 Functional and Semantic Roles

The *Dependency* property in PG is the (only) property meant to model a semantic relationship between two constituents. According to the definition given in Blache ([10]), the dependeny property[5] `c0` : `c1` $\rightsquigarrow$ `c2` holds for the parent constituent of category `c0` between two children constituents of categories `c1` and `c2` if those two children constituents are *semantically compatible*. Intuitively, the role of that property in the grammar (still in (Blache [10])) is first to specify the existence of a predicative structure for the governor, and second to constrain the argument structure through the constituents feature structures.

That rather permissive definition is unclear as how the *semantic compatibility* between categories must be checked. The lack of conditional satisfaction makes it a property which can not be violated: either it holds true for two constituents, or it is non-pertinent, but it never fails. The rationale in Blache's proposal for such a semantics is to include in the characterisation of a sentence pieces of information regarding the existence of dependency relationships among constituents. VanRullen ([11]) takes a different perspective and defines for the Dependency property a semantics where the dependency relation between two categories is conditioned by feature agreement (e.g. in gender, number, person, ...) through feature unification.

In (Duchier et al. [6]) the Dependency property is not axiomatised due to that lack of conditional satisfaction. Yet, the idea to combine phrase structure and dependency structure within the same information structure is elegant and quite convenient for interfacing syntax and semantics. Therefore we follow Blache's proposal on the Dependency property, which we modify slightly in order to be able to specify functional roles within a sentence and the way they propagate through the phrase structure. The property is conditioned either by the sole existence of the governor and the modifier nodes, as in the original proposal, or by feature values. Unlike in Blache, roles are feature values as opposed to features. Predicate subcategorisation schemes, especially, may serve to specify different arguments. The schemes for the verb *plaider* (*to plead*), given in Figure 1, come from the LexSchem resource (Messiant [12]). Each argument category and role is then propagated in the phrase structure through Dependency properties. This way, it is possible to use the Dependency property to infer semantic roles from functional ones. In Table 1, for instance, the `VP` category specifies a Dependency between a `V` and an `NP`, where the `NP`'s semantic role is `PAT` (patient) whenever

[5] We use the notation introduced by Duchier et al. ([5]).

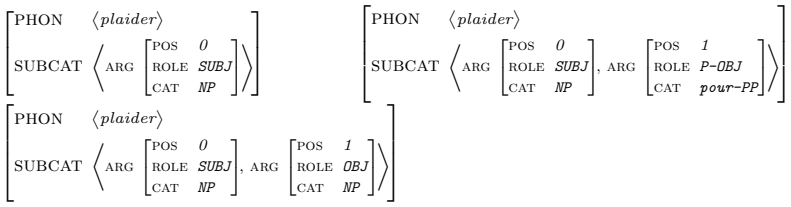


Fig. 1. Subcategorisation schemes for the verb *plaider* (*to plead*).

the V expects an object. Note that this should be refined whenever possible depending on the domain of values available for the semantic roles. A domain, for instance, which would make a distinction between a *patient* and a *recipient* role would benefit from a Dependency property where a P-OBJ (*for-object*) is turned into a recipient role.

4 Re-generation and Completion Message

Given a quasi-expression, an approximated parse tree for it, and its characterisation, is it possible to automatically infer candidate corrected syntax trees? To address that question we consider that every constraint failure in the characterisation can be interpreted positively either as a tree transformation operation or as an operation over feature structures. Those operations can be seen as grammar corrections.

Re-generation Take the example from Figure 2, where the constituent NP2 violates the constraint NP : N ⇒ D (*within an NP constituent an N requires a D*). In order for that constraint to be satisfied it is sufficient to transform the NP2

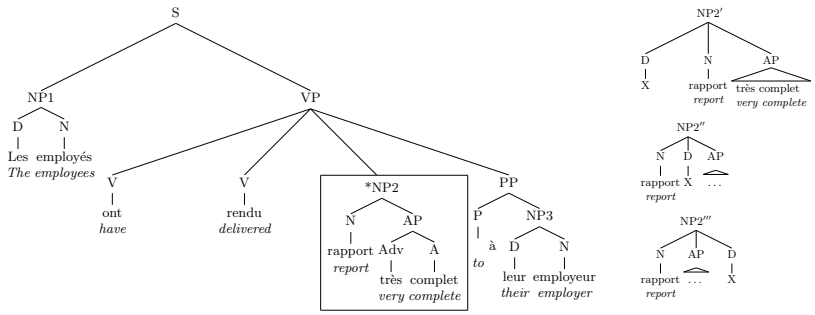


Fig. 2. Approximated parse for quasi-expression, and re-generated sub-trees

node by insertion of a daughter node labelled with the category D. This operation generates three sub-trees, illustrated in Figure 2. Three new candidate trees are then obtained by replacement of the NP2 node by each of the three sub-trees.

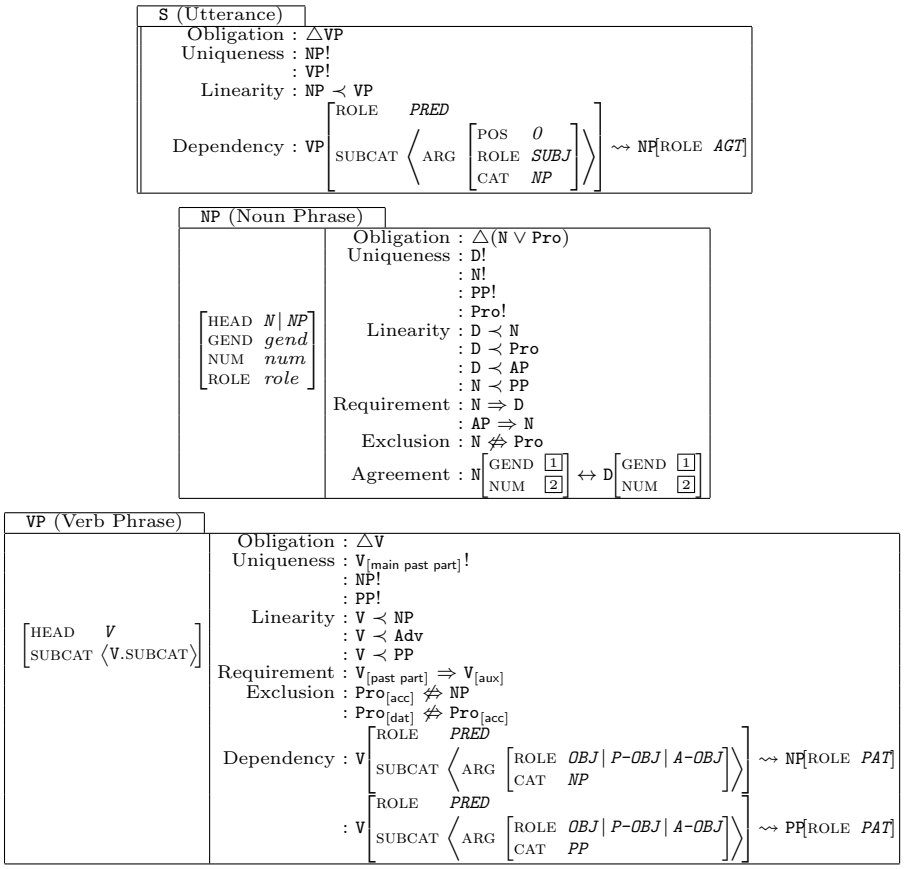


Table 1. Example property grammar for French

The multiple results are not a problem in that case, since they can easily be disambiguated through a new check of the grammar constraint system, that is, through the characterisation of each of the three new models. The sub-tree NP2′ is the only one meeting all the grammar constraints.

The sets of candidate sub-trees are generated using the following tree operations, where τ is a tree, and c, c_1, c_2 are node labels (i.e. categories):

- *Node insertion*, denoted by $\tau \downarrow c$
- *Node deletion*, denoted by $\tau \nmid c$
- *Node permutation*, denoted by $c_1 \overset{\mathcal{T}}{\leftrightarrow} c_2$

After generalisation, every PG property corresponds to a transformation tree operation:

Property	*Violated instances*	*Tree operation*
Requirement	$\mathcal{I}_\tau[\![c_0 : c_1 \Rightarrow s_2]\!]$	$\tau \downarrow s_2$
Obligation	$\mathcal{I}_\tau[\![c_0 : \triangle c_1]\!]$	$\tau \downarrow c_1$
Linearity	$\mathcal{I}_\tau[\![c_0 : c_1 \prec c_2]\!]$	$c_1 \overset{\tau}{\leftrightarrow} c_2$
Uniqueness	$\mathcal{I}_\tau[\![c_0 : c_1!]\!]$	$\tau \nmid c_1$
Exclusion	$\mathcal{I}_\tau[\![c_0 : c_1 \not\Rightarrow c_2]\!]$	$\tau \nmid c_1 \cup \tau \nmid c_2$

Completion Message After re-generation and re-characterisation partial phrase structures are obtained, which contain underspecified nodes, including empty leaves (denoted by `X` in our figures). Those correspond to lexical items to be found through exploration of the semantic network. In the following, to illustrate the process we narrow down the scope of investigation and focus on cases of missing predicate. Figure 3 illustrates a candidate re-generated model for the quasi-expression **L'avocat le dossier de son client* (**The lawyer his client's file*). In order to complete the `NP2`'s surface realisation the exploration process of the

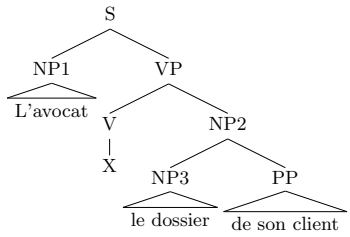


Fig. 3. Candidate re-generated model for quasi-expression

semantic network is fed with messages about the semantic relationships the missing predicate is involved in. Those messages take the form of triples $\langle a, \texttt{:R}, b\rangle$, where `:R` denotes an oriented semantic relation, and a and b its ordered elements. Examples of relations in use in the *rezoJDMFR* network (Lafourcade and Joubert [13]) are Agent, Patient, Instrument, Succession, ..., though at this stage we limit ourselves to the Agent and Patient ones. The messages are built from gathering the relevant information in the model's characterisation.

Following up with the example sentence from Figure 3, we know from `VP` that `NP2` is in a Patient relationship with the predicate `V`, which gives us a value for R in message ♯1. `NP2` takes its head from `NP3` by propagation, which takes its own from the noun *dossier* (*file*). That gives us a value for a in message ♯1. The last message member is instantiated with the wildcard `X`, hence the message ♯1: $\langle$`X`, `:PAT`, *dossier*$\rangle$. As for the members of message ♯2, they come

from the knowledge in `S` that `NP1` is in an Agent relationship with `VP`, which by inheritance let us instantiate b with the value *avocat* (*lawyer*). That gives us, for message ♯2, $\langle avocat, \texttt{:AGT}, \texttt{X} \rangle$.

In the end we get the following list of messages:
$\{\langle \texttt{X}, \texttt{:PAT}, dossier \rangle, \langle avocat, \texttt{:AGT}, \texttt{X} \rangle\}$.

5 Propagation

For our experiments, we make use of a lexical network that has been constructed by means of a (serious) game available on the net : JeuxDeMots. A propagation algorithm combining constraints and this network is presented.

5.1 A lexical network...

The structure of the lexical network we are building and using is composed of nodes and links between nodes, as it was initially introduced in the end of 1960s by Collins and Quillian ([14]) (developed by Sowa ([15])), used in the small worlds by Gaume et al. ([16]), and more recently clarified by Polguère ([17]). A node of the network refers to a term (or a multiple word expression), usually in its canonical form (lemma). The links between nodes are typed and are interpreted as a possible relation holding between the two terms. Some of these relations correspond to lexical functions, some of which have been made explicit by Mel'cuk ([18]) and Polguère ([17]). It would have been desirable the network to contain all those lexical functions, but considering the principle of our software JeuxDeMots, it is not reasonably feasible. Indeed, some of these lexical functions are too much specialized; for example, Mel'cuk et al. ([18]) make the distinction between the Conversive, Antonym and Contrastive functions. They also consider refinements, with lexical functions characterized as *wider* or *more narrow*. JeuxDeMots being intended for users who are *simple Internet users*, and not necessarily experts in linguistics, such functions could have been badly interpreted by them. Furthermore, some of these functions are too poorly lexicalized, that is, very few terms possess occurrences of such relations; it is for example the case of the functions of Metaphor or Functioning with difficulty. More formally, a lexical network is a graph structure composed of nodes (vertices) and links.

- A node is a 3-tuple : `<name, type, weight>`
- A link is a 4-tuple : `<start-node, type, end-node, weight>`

The name is simply the string holding the term. The type is an encoding referring to the information holding by the node. For instance a node can be a term or a Part of Speech (POS) like :Noun, :Verb. The link type refer to the relation considered. A node weight is interpreted as a value referring to the frequency of usage of the term. The weight of a relation, similarly, refers to the strength of the relation.

JeuxDeMots possesses a predetermined list of relation types, and for now the players cannot add new types. Relation types fall into several categories:

– Lexical relations: synonymy, antonymy, expression, lexical family These types of relations are about vocabulary.
– Ontological relations: generic (hyperonymy), specific (hyponymy), part of (meronymy), whole of (holonymy) ... It is about relations concerning knowledge in objects of the world.
– Associative relations: free association, associated feeling, meaning It is rather about subjective and global knowledge; some of them can be considered as phrasal associations.
– Predicative relations: typical agent, typical patient ... They are about types of relation associated with a verb and the values of its arguments (in a very wide sense). Those are similar (if not identical) to semantic roles, which are of primary interest for us in this article.

The types of relation implemented in JeuxDeMots are thus of several natures, partially according to a distinction made by Schwab and Lafourcade ([19]): some of them are part of knowledge of the world (hyperonymy / hyponymy, for example), others concern linguistic knowledge (synonymy, antonymy, expression or lexical family, for example). Most players do not make this distinction which remains often vague for them. Here, the word *relation* has to be understood as an occurrence of relation, and not as a type of relation. Let us note that between two same terms, several relations of different types can exist.

5.2 ... a game for building it...

To ensure a system leading to quality and consistency of the base, it was decided that the validation of the relations anonymously given by a player should be made by other players, also anonymously. Practically, a relation is considered valid if it is given by at least one pair of players. This process of validation is similar to the one used by von Ahn et al. ([20]) for the indexation of images or more recently by Lieberman et al.([21]) to collect common sense knowledge. As far as we know, this was never done in the field of the lexical networks. In Natural Language Processing, some other Web-based systems exist, such as Open Mind Word Expert (Mihalcea and Chklovski [22]) that aims to create large sense tagged corpora with the help of Web users, or SemKey (Marchetti et al. [23]) that exploits WordNet and Wikipedia in order to disambiguate lexical forms to refer to a concept, thus identifying a semantic keyword.

A game takes place between two players, in an asynchronous way, based on the concordance of their propositions. When a first player (A) begins a game, an instruction concerning a type of competence (synonyms, opposite, domains, ...) is displayed, as well as a term T randomly picked in a base of terms. This player A has then a limited time to answer by giving propositions which, to his mind, correspond to the instruction applied to the term T. The number of propositions which he can make is limited inducing players not just type anything as fast as possible, but to have to choose amongst all answers he can think of. The same

term, along the same instruction, is later proposed to another player B; the process is then identical. To increase the playful aspect, for any common answer in the propositions of both players, they receive a given number of points. The calculation of this number of points (as explained by Lafourcade and Joubert ([13])) is crafted to induce both precision and recall in the feeding of the database. At the end of a game, propositions made by the two players are showed, as well as the intersection between these terms and the number of points they win.

According to the JeuxDeMots Web site, at the time of the writing of this paper, the lexical network contains more than 1100000 relations linking more than 230000 terms. Around 900000 games (with a mean of 1 minute per game) have been played corresponding to approximately 13000 hours (about 550 days) of cumulative play.

5.3 ... and a propagation algorithm

We devise a quite simple algorithm taking a set of constraints for input. A constraint here takes a form that is similar to an unweighted relation, that is to say a 3-tuple : `<start-node, type, end-node>`. One or both node can be a free variable with or without information of their POS. This form is directly mappable to relations in the lexical network. With the previous example, we have :

- `<X, :AGT, avocat>` and `<X, :AGT, dossier>`

Our algorithm is inspired by Page et al. ([24]) and partially by the LexRank algorithm (Bouklit and Lafourcade [25]). The simple idea is to activate nodes of the lexical network that are involved in the constraints, and then to propagate activations along the network. As in practice, the entire network would be too large to be tractable, we reduce it only to the subset containing the first neighbours (at distance 1). All links are copied with the original weight when belonging to one of the constraints, or with weight equal to 1 otherwise. This propagation approach has been proven be convergent with a power iteration computing method. To be effective, the main hypothesis is that the set weight linking a node to its neighbour is considered as representing a probability distribution.

In the above example, an activation of 1 is inited to the node *avocat* and *dossier*. From *avocat*, all neighbours are copied with a link with a default value of 1, but those linked by a `:AGT` link. The process is similar for the node *dossier*. The spreading of the activation to a neighbour is done according to the ratio between this specific link weight. More precisely, the activation propagation from a node A to B is equal to the activation of A times the ratio of the weight of the link between A and B with the sum of the link weights from A). The output of the algorithm is *a set of activated terms*. In the above example, two terms are mostly activated: *plaider* and *étudier*. Furthermore, the proper specific meaning of *avocat>justice* (as *lawyer* and not *avocado*) is activated.

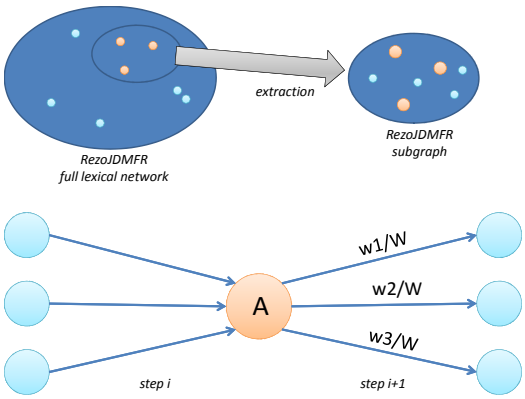


Fig. 4. Extraction of the lexical network subgraph and propagation of activation along nodes according to link weights.

6 Conclusion

The near absence of syntax–semantics interface for the Property Grammar framework is a serious impediment to its use for deep processing. Meanwhile, as a constituency-based MTS framework PG offers formal properties, which make it especially well-suited to address problems such as grammar error correction. In this paper we have started exploring the possibility to address the problem of the syntax–semantics interface through the pairing with a semantic network. We have shown that the linguistic knowledge contained within PG constraints, together with a deep phrase structure, allows for the construction of detailed semantic information about a—possibly ill—sentence. Such semantic information could not be gathered by compositional means only. We have also shown how to use that information to explore a semantic network and find suitable lexical items to complete a sentence missing words, with an emphasis on missing predicates.

References

1. Pullum, G., Scholz, B.: On the Distinction Between Model-Theoretic and Generative-Enumerative Syntactic Frameworks. In de Groote, P., Morrill, G., Rétoré, C., eds.: Logical Aspects of Computational Linguistics: 4th International Conference. Number 2099 in LNAI, Springer Verlag (2001) 17–43
2. Dahl, V., Gu, B.: On semantically constrained property grammars. In: Constraints and Language Processing. (2008) 20
3. Blache, P.: Les Grammaires de Propriétés : des contraintes pour le traitement automatique des langues naturelles. Hermès Sciences (2001)
4. Bès, G., Blache, P.: Propriétés et analyse d'un langage. In: Proc. of the 1999 Conf. on Traitement Automatique du Langage Naturel (TALN'99). (1999)

5. Duchier, D. and Prost, J.-P. and Dao, T.-B.-H.: A model-theoretic framework for grammaticality judgements. In: Proc. of FG'09. Volume 5591 of Lecture Notes in Artificial Intelligence., Springer (2009)
6. Duchier, D., Dao, T.B.H., Parmentier, Y., Lesaint, W.: Property Grammar Parsing Seen as a Constraint Optimization Problem. In: Proceedings of the 15th Intl Conference on Formal Grammar (FG 2010). (2010)
7. Chauché, J.: Un outil multidimensionnel de l'analyse du discours. In: Proc. of the 10th Int'l Conf. on Computational Linguistics and 22nd annual meeting on Association for Computational Linguistics, ACL (1984) 11–15
8. Wagner, J., Foster, J., van Genabith, J.: Judging grammaticality: Experiments in sentence classification. CALICO Journal **26**(3) (2009) 474–490
9. Wong, S., Dras, M.: Parser Features for Sentence Grammaticality Classification. In: Australasian Language Technology Association Workshop 2010. (2011) 67
10. Blache, P.: Property Grammars: A Fully Constraint-based Theory. In Christiansen, H., Skadhauge, P.R., Villadsen, J., eds.: Constraint Solving and Language Processing. Volume 3438 of LNAI. Springer (2005)
11. VanRullen, T.: Vers une analyse syntaxique à granularité variable. PhD thesis, Université de Provence, Informatique (2005)
12. Messiant, C.: A Subcategorization Acquisition System for French Verbs. In: Proc. of the ACL-08: HLT Student Research Workshop, ACL (2008) 55–60
13. Lafourcade, M., Joubert, A.: Détermination des sens d'usage dans un réseau lexical construit à l'aide d'un jeu en ligne. In: Proc. of the Conférence sur le Traitement Automatique des Langues Naturelles (TALN'08). (2008) 189–199
14. Collins, A., Quillian, M.R.: Retrieval time from semantic memory. Journal of verbal learning and verbal behaviour **8**(2) (1969) 240–248
15. Sowa, J.: Semantic networks. Encyclopedia of Artificial Intelligence. edited by S.C. Shapiro, Wiley, New York (1992)
16. Gaume, B., Duvignau, K., Vanhove, M.: Semantic associations and confluences in paradigmatic networks. In Vanhove, M., ed.: Typologie des rapprochements sémantiques. Benjamins (2007)
17. Polguère, A.: Structural properties of lexical systems: Monolingual and multilingual perspectives. In: Proc. of the Workshop on Multilingual Language Resources and Interoperability (COLING/ACL 2006). (2006) 50–59
18. Mel'cuk, I.A., Clas, A., Polguère, A.: Introduction à la lexicologie explicative et combinatoire. Ed. Duculot AUPELF-UREF (1995)
19. Schwab, D., Lafourcade, M.: Modelling, detection and exploitation of lexical functions for analysis. ECTI Journal **2** (2009) 97–108
20. von Ahn, L., Dabbish, L.: Labelling images with a computer game. In: Proc. of ACM Conf. on Human Factors in Computing Systems (CHI). (2004) 319–326
21. Lieberman, H., Smith, D.A., Teeters, A.: Common consensus: a web-based game for collecting commonsense goals. In: Proc. of Int'l Conf. on Intelligent User Interfaces (IUI'07). (2007)
22. Mihalcea, R., Chklovski, T.: Open mind word expert: Creating large annotated data collections with web users' help. In: Proc. of the EACL 2003 Workshop on Linguistically Annotated Corpora (LINC 2003). (2003)
23. Marchetti, A., Tesconi, M., Ronzano, F., Rosella, M., Minutoli, S.: Semkey: A semantic collaborative tagging system. In: Proc. of SemKey: A Semantic Collaborative Tagging System. (2007)
24. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking : Bringing Order to the Web. Technical report, Stanford University (1998)

25. Bouklit, M., Lafourcade, M.: Propagation de signatures lexicales dans le graphe du web. In: Proc. of RFIA'2006, Tours, France. (2006) 9