Lexicalisation as a Description Logic Inference Task

Claire Gardent, Evelyne Jacquey

Lexicalisation as a Description Logic Inference Task

Claire Gardent* and Evelyne Jacquey **

* *CNRS - LORIA, gardent@loria.fr*
Campus Scientifique BP 239, Vandoeuvre-les-Nancy, France
** *CNRS - ATILF, ejacquey@atilf.fr*
44 avenue de la libération, Nancy, France

Abstract

We argue that lexicalisation (i.e., the task of mapping conceptual input onto words), is intrisincally an inference task. We then present an algorithm which integrates lexicalisation as inference with surface realisation.

1 Introduction

In natural language generation, the *lexicalisation task* consists in finding, given some conceptual input (e.g., a set of facts describing a situation), a set of words which when combined into a grammatical sentence (or text) expresses this input. In this paper, we argue that lexicalisation is primarily an inference task which can be sketched as follows:

> Given an entity e, a set of facts ϕ_e about that entity and a subsumption based ontology capturing lexical meanings KB_{Lex}, lexicalising e involves finding the most specific concepts in KB_{Lex} of which given ϕ_e, e is an instance.

Assuming concepts in KB_{Lex} are linked to lexical entries in the grammar, lexicalisation then consists in retrieving from the lexicon those lexical entries that are linked to the retrieved concepts.

In what follows, we flesh out this idea. First, we motivate the inference based approach. Second, we present an algorithm integrating lexicalisation and surface realisation. Third, we discuss possible extensions and optimisations.

2 Description Logic

Before we begin, we need to introduce the logical framework used to perform reasoning namely, Description Logic. Description Logic (DL, [1]) provides a natural logical framework in which to reason about concepts. In this framework, concepts can be defined and ordered under subsumption whilst assertions can be made about properties of, and relations between, individuals. Moreover, inference services are available which support the retrieval both of all the concepts true of an individual and of all the individuals satisfying a given concept. In short, DLs support the definition and use of subsumption hierarchies i.e., precisely the type of hierarchies we will use to represent and classify word meanings. The particular language we assume has the following syntax.

$$C, D \rightarrow A|\top|\bot|\neg A \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$$

The semantics of this language is given below with Δ the domain of interpretation and I the interpretation function which assigns to every atomic concept A, a set $A^I \subseteq \Delta$ and to every atomic role R a binary relation $R^I \subseteq \Delta \times \Delta$.

$$\begin{aligned}
\top^I &= \Delta \\
\bot^I &= \emptyset \\
(\neg A)^I &= \Delta \backslash A^I \\
(C \sqcap D)^I &= C^I \cap D^I \\
(C \sqcup D)^I &= C^I \cup D^I \\
(\forall R.C)^I &= \{a \in \Delta \mid \forall b(a,b) \in R^I \rightarrow b \in C^I\} \\
(\exists R.C)^I &= \{a \in \Delta \mid \exists b \in C^I \wedge (a,b) \in R^I\}
\end{aligned}$$

3 Why inference?

To perform the lexicalisation task, discrimination nets were first proposed [10], then frames and nearest neighbor classification [13] and finally taxonomical knowledge bases [15].

The first two methods (discrimination nets and frames with nearest neighbour classification) were intrisincally procedural. They became obsolete once taxonomical reasoning developed which allowed capturing similar intuitions but in a more declarative manner (i.e., through subsumption relations instead of arbitrary procedures).

The third approach (Stede's approach, [15]) is closest to our proposal in that it is based on taxonomical reasoning. In this approach, the comparison between the meaning representation given as input and the meaning representations present in lexical entries is done by a matching procedure. Crucially, this matching procedure takes into account the subsumption relation (written $\sqsubseteq$) encoded in a taxonomy of concepts and relations implemented in the LOOM language (i.e., a language of the KL-ONE family nowadays usually

referred to as description logic). But it does so in a limited way. Indeed, the approach is based on the assumption that semantic representations are directed acyclic graphs and the matching procedure is a graph matching procedure which is defined as follows:

G subsumes S iff:
(i) $type(root_S) \sqsubseteq type(root_G)$
(ii) $\forall x, \forall R[R(root_G, x_G) \rightarrow \exists y[R(root_S, x_S)$ and the graph rooted in x_G subsumes the graph rooted in $x_S]$.

In words: a graph rooted in $root_G$ subsumes another graph rooted in $root_S$ if the type of $root_G$ subsumes the type of $root_S$ in the ontology and for all edges coming out of $root_G$ with label R and end vertex y_G, there is an edge coming out of $root_S$ with label R and end vertex y_S such that the graph rooted in y_G subsumes the graph rooted in y_S.

For instance, the graph on the left handside of Figure 1 subsumes the graph on its left handside since:

- The type of x subsumes the type of x' (Move subsumes Move)
- The "object" edge leading out of x is matched by an "object" edge leading out of x' and
- the graph rooted in y subsumes the graph rooted in y' (TOP subsumes Liquid)

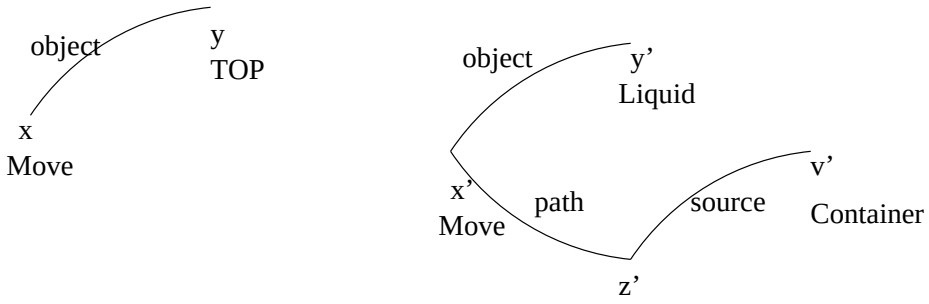

Fig. 1. Two graphs in a subsume relation

A lexical item is selected iff its semantic representation *Lex* subsumes the input meaning representation *Input*.

This approach captures the essence of the lexicalisation task namely, that the comparison between lexical semantics and semantic input must involve knowledge based subsumption i.e., subsumption with respect to some knowledge base (as opposed to e.g., subsumption between feature structures).

Nonetheless it fails to capture the more complex cases of knowledge based reasoning involved in lexicalisation. Here is a simple example illustrating this point. Suppose the semantic input is the following set of DL assertions.

(1) $i : \mathit{Human}, i : \mathit{Female}, (i, d_1) : \mathit{hasChild}, d_1 : \mathit{Human}, d_1 : \mathit{Female}$, $(d_1, d_2) : \mathit{hasChild}, d_2 : \mathit{Human}$

One natural way to lexicalise the description of the individual i is by using the word "grandmother". That is, rather than describing i as a female person whose child is a woman who herself has a child, then one can describe i as a grandmother – this is both more efficient and more natural.

However the lexical semantics of the noun "grandmother" does not say that a grandmother is a female person whose child is a woman who herself has a child. This would be both wrong (a grandmother need not be mother of a *woman* with child) and lexically inappropriate (whenever possible, a concept definition is usually kept short by involving defined rather than primitive concepts).

Instead, a standard dictionnary definition of the noun "grandmother" might state that a grandmother is the mother of a parent. Further a parent might be defined as either a mother or a father where a mother (father) is a woman (man) who has a child. And finally, a woman (man) can be defined as a female (male) human being.

In short, such cases are problematic for a matching based approach because the structures of the inputs are widely divergent: while the input describes an individual as a "woman with a woman child who has a human child", a natural definition of the concept instantiated by such an individual is as "a woman whose child is a parent".

Using description logic on the other hand, the inference that i is best described as a grandmother is straigthforward. The lexicographic intuitions described above can be encoded as given in Figure 1.

$$\mathit{GrandMother} \equiv \mathit{Woman} \sqcap \exists \mathit{hasChild}.\mathit{Parent}$$

$$\mathit{Parent} \equiv (\mathit{Father} \sqcup \mathit{Mother})$$

$$\mathit{Mother} \equiv \mathit{Woman} \sqcap \exists \mathit{hasChild}.\mathit{Human}$$

$$\mathit{Father} \equiv \mathit{Man} \sqcap \exists \mathit{hasChild}.\mathit{Human}$$

$$\mathit{Man} \equiv \mathit{Human} \sqcap \mathit{Male}$$

$$\mathit{Woman} \equiv \mathit{Human} \sqcap \mathit{Female}$$

$$\mathit{Male} \equiv \neg \mathit{Female}$$

Fig. 2. DL encoding of lexicographic definitions

Next, the retrieval facility made available by most DL reasoners and in particular by RACER [17] can be used to retrieve the concepts subsuming i given the input in (1) and the above ontology. Retrieval will return the following set:

(2) *Human,Female,Woman,Mother,Parent,GrandMother*

Finally by using the ontology to determine the subsumption relations between these concepts, it is possible to infer that *GrandMother* is the most

specific of these six concepts.

In sum, it is possible to infer that i is an instance of the *GrandMother* concept but the task is a full blown, logical inference task whose solving is usually performed using tableaux based methods rather than a simple structural matching algorithm.

To recap, Stede's approach falls short of accounting for the lexicalisation data because it compares input and lexical semantics by means of a structural algorithm (much like the structural reasoning algorithm used for DL systems without negation); and because, to support lexicalisation, a more complex type of reasoning is needed which treats concept definitions not as data structures but as logical formulae.

In the following section, we show how an inference based lexicalisation procedure can be integrated within the realisation phase of generation.

4 A realisation algorithm with lexicalising lookup

Natural language generation falls into three main parts: planning what to say (macroplanning), planning how to say it (microplanning) and saying it (surface realisation). As argued in [18,16], surface realisation and microplanning tasks (e.g., referring expression generation, lexicalisation and aggregation) interact in intricate ways thus suggesting an integrated rather than a pipeline architecture as has often been advocated (cf. for example [14]).

The algorithm we present here assumes macroplanning and integrates tasks from both microplanning and surface realisation. Thus, the algorithm defined in Figure (3,4) assumes as input from the macroplanner a set of facts and a set of entities (event, individuals etc.) to be described. Next referring expressions are handled which contributes to expand the input. Specifically, we assume that for each discourse old entity, a set of facts is computed by an algorithm such as e.g., [7] which uniquely identifies this entity. This set of facts is then added to the input. Once referring expressions have been handled, a chart based generation algorithm is used which both lexicalises the entities to be described and constructs a syntactically and morphologically correct tree on the basis of the selected lexical entries.

5 A detailed example

To illustrate the workings of the algorithm defined in Figures (3,4), we now run through a simple example. Suppose the input and the knowledge base are as given in figure 5.

In the first step, the algorithm retrieves the most specific concepts of which the objects to be described ($\{h, g, e\}$) are instances namely, *GrandMother*, *Kind*, and *Means* for g; *Hat* for h; and *Running* for e.

Next the lexical entries are retrieved which are linked to these concepts. In the process, the semantics indices contained by these lexical entries are

Input:
Sit: a set of facts (represented as an A-Box)
I: a set of individuals to be described
KB: a DL ontology
G: A TAG grammar where each lexical entry L consists of:
- a tree T_L whose nodes are decorated with feature structures
- the list I_L of indices introduced by the lexical entry
- a DL formula ϕ_L representing the meaning conveyed by the entry

T: a table linking concepts and lexical entries

Initialisation:
for all i in I **do**
Construct the set $C^i = \{C \mid i : C \text{ and } \forall C' \neq C (i : C' \rightarrow C' \sqsubseteq C)\}$
for all pairs (C, L) with $C \in C^i$ and L a lexical entry linked to C **do**
Add the item $< I_L,\ L >$ to the agenda, with I_L the index list of L
end for
if i is discourse old **then**
Add the entry $< [i],\ L_{the} >$ to the agenda, with L_{the} the lexical entry for the definite determiner "the"
else if i is discourse new **then**
Add the entry $< [i],\ L_a >$ to the agenda, with L_a the lexical entry for the indefinite determiner "a"
end if
end for

Processing the agenda:
while The agenda is not empty **do**
Retrieve first item I in the agenda
if I is in chart **then**
Skip
else
Add I to chart
end if
Combine I with items currently in the chart and add resulting items to the agenda
end while

Checking success:
Successful items are items $< I_{L_i},\ L_i >$ such that
1. $I_{L_i} = I$ {all individuals to be described are described}
2. For all $i \in I_{L_i}$, the set of atomic concepts of which i_{L_i} is an instance given the semantics ϕ_{L_i} of $L_i u$ is the same as the set of atomic concepts of which i_{L_i} is an instance given the input *Sit*.

Fig. 3. Lexicalisation and Realisation Algorithm

$< i,\ L_i >$ combines with $< j,\ L_j >$ to yield $< k,\ L_k >$ iff
- $i \cap j = \emptyset$ {Items must have an index in common}
- L_i and L_j can be combined either by substitution or by adjunction
- $k = i \cup j$
- L_k is the result of combining L_i with L_j

Fig. 4. Combining Items Procedure

Input	Knowledge Base
$\mathrm{I} = \{g, h, e\}$,	*Human* $\sqsubseteq$ *Object*, *Kind*
g:Woman, *g:Kind*,	*GrandMother* $\equiv$ *Woman* $\sqcap$ $\exists$*hasChild.Parent*
(g, d_1)*:HasChild*,	*Mother* $\equiv$ *Woman* $\sqcap$ $\exists$*hasChild.Human*
(d_1, d_2)*:HasChild*,	*Father* $\equiv$ *Man* $\sqcap$ $\exists$*hasChild.Human*
d_1*:Human*, d_2*:Human*	*Man* $\equiv$ *Human* $\sqcap$ *Male*
(g, h)*:with*, h*:Hat*,	*Male* $\equiv$ $\neg$*Female*
e*:Run*, (e, g)*:agent*	*Running* $\equiv$ *Run* $\sqcap$ $\exists$*agent.Human*
	Parent $\equiv$ *(Father* $\sqcup$ *Mother)*
	Woman $\equiv$ *Human* $\sqcap$ *Female*
	Means $\equiv$ $\exists$*with.Object*

Fig. 5. Example input

instantiated. Assuming a minimal grammar for the sake of simplicity [1], the algorithm might at this stage retrieve the lexical entries indicated in Figure 6.

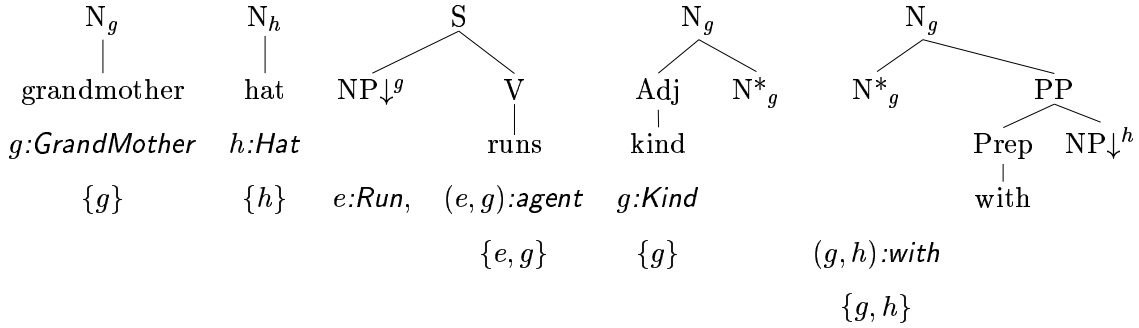

Fig. 6. Selected lexical entries

Assuming g is discourse old and h discourse new [2], the lexical entries for "the" and "a" are then added to the agenda with the appropriate instantiations. The agenda is then processed which will yield among other the tree and semantic representation depicted in Figure 7.

The semantic representation associated with this tree involves the same individuals $\{g, h, e\}$ as the input and for each of these individuals the exact

[1] The specific type of TAG assumed here is described in more details in [8] and assumes that each tree is associated with a semantic representation sharing variables with nodes in the tree. This sharing implements the syntax/semantics interface: it specifies which syntactic constituent provides a value for which semantic argument.

[2] As is usual (cf. in particular the SPUD generator [16]), we assume here that a record of dialog history is kept so that in particular the discourse status of entities can be checked.

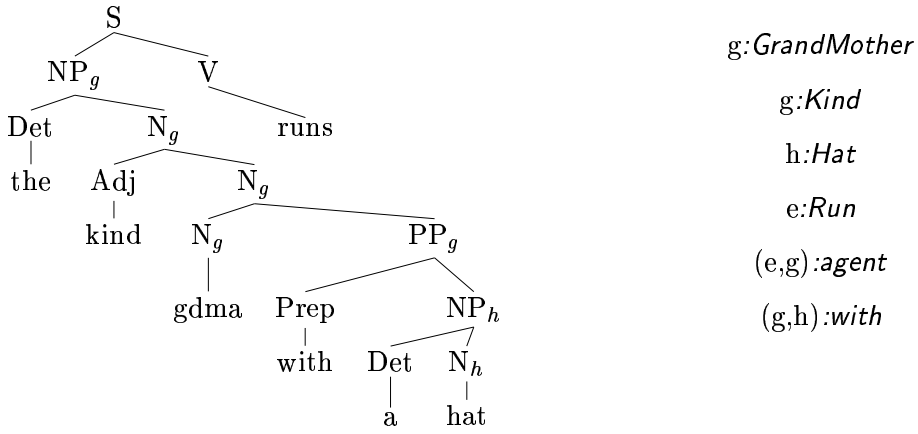

Fig. 7. Successfull item

same set of properties can be inferred from the knowledge base as from the input situation. More precisely, the set of most specific concepts of which $\{g, h, e\}$ are instances is exactly the same as for the input: the most specific concepts of which g is an instance are *GrandMother,Kind* and *Means*; of h, *Hat* and of e, *Running*. Hence the derived tree in Figure 7 is successful and sentence (3) is generated.

(3) The kind grandmother with a hat runs.

6 Paraphrases

The same meaning can be expressed by distinct natural language expressions across languages (through translations) and within a language (through paraphrases). For instance, the content:

(4) e:*Go* j:*John* (e, j):*agent* (e, p):*means* p:*Plane* (e, r):*dest* r:*Rome*,

is realised both by the english paraphrases (5a-b) and by the french sentence (5c).

(5) a. John flies to Rome.
b. John goes to Rome by plane.
c. John va à Rome en avion.

As it is formulated in Figure 4 however, the algorithm presented here will only generate (5a) because of the restriction that only the *most specific concepts* be considered (assuming the objects to be described are e, j and r then given the ontology in Figure 8, *Fly,John,Dest,Rome* and the corresponding lexical entries will be selected but neither *Go* nor *Plane*) .

To allow for paraphrases, we can relax the constraint regulating the selection of the relevant lexical concepts and allow all lexical entries to be selected which are linked to a concept subsuming any of the entities occurring in the

input. For the above example, this would mean that besides the entries mentioned above, the entries linked to the concepts *Means* and *plane* would also be selected thus allowing for the generation of (5b-c).

Since the success condition ensures that the generated text is semantically equivalent to the input, relaxing lexical selection in this way will not lead to ill formed output (i.e., output that is either less or more specified than the input). However, it is clear that such a "lenient" lexical look up will, by multiplying the number of items in the initial agenda, tremendously increase the combinatorics. In the following section, we therefore sketch two possible optimisations for our algorithm.

$$Go \equiv Event \sqcap \exists agent.\top \sqcap \exists dest.Location$$

$$Fly \equiv Go \sqcap \exists means.AirTransport$$

$$Plane \sqsubseteq AirTransport$$

$$Rome \sqsubseteq Location$$

$$Dest \equiv \exists dest.Location$$

$$Means \equiv \exists with.Object$$

Fig. 8. Fly Ontology

7 Optimisation

Chart generation is known to be exponential for intersective modification [4,3,11] even when chart parsing is polynomial. [4] proposes to solve the problem by treating intersective modification in a separate phase after all possible edges that do not involve potentially recursive intersective modification have been constructed. In this way, worst case complexity is polynomial provided modifier order is constrained by the grammar.

Because the grammar we use is a TAG, such an optimisation is straightforward to introduce in our algorithm. Indeed TAG is based on a fundamental distinction between basic recursive units (called "auxiliary trees") and non recursive units (so called "elementary trees"). Thus to ensure that recursive modification takes place only after all non recursive operations have, it suffices to keep two agendas: one for auxiliary trees and the other for elementary trees and to process the "elementary trees agenda" before the "auxiliary trees agenda". In this way, it can be ensured that a complete syntactic skeleton is built before modifiers are treated.

Further, because TAG auxiliary trees correspond to different linguistic notions (modification through adjectives and PPs but also introduction of completives leading to long distance dependencies), this auxiliary trees agenda could be further subdivided into several sub agendas thus ensuring that modifier trees be treated after all others.

Another possible optimisation draws on the work of [2]. In a parsing approach based on a grammar with "polarities", [2] show that preprocessing sequences of lexical items can dramatically increase performance sometimes decreasing the number of sequences to be parsed from several thousands to a few units. In [2], the strategy consists in filtering out as ungrammatical any sequence of lexical items whose polarity is not neutral, the idea being that polarities (plus and minuses on feature values) express needs and resources and must therefore cancel each other out.

A similar technique could be used here which would consist in filtering out of the chart only those sequences that are "semantically connected" roughly, sequences of lexical items whose semantic indices are "compatible". The list of indices (I_L in our algorithm) of each lexical entry would be ordered to reflect the linear order of their realising constituents (for instance, the "with" tree would have the index list $[g, h]$ to reflect the fact that g is realised by an N that precedes the NP realising h) and connected sequences would be defined using finite state techniques. Exactly what a connected sequence is remains to be defined however. The interaction with the previous optimisation (elementary trees before auxiliary trees) also needs to be worked out.

8 Conclusion

We have argued for an approach that views lexicalisation as a DL inference task, and for an architecture in which lexicalisation substitutes knowledge based subsumption for the unification based impoverished lexical look up implemented in surface realisers such as e.g., [4,9].

One feature of the approach is that it allows a clean separation between the definition and organisation of lexical concepts on the one hand and the semantic information that has to be present in a grammar lexical entry on the other hand. Roughly, the semantic information contained in lexical entries reduces to basic predicative (e.g., *grandmother(x)* for the word "grandmother") and interface (e.g., theta-grid information for verbs) information. Finer grained lexical semantics information (i.e., roughly lexicographic definition) is contained in the ontology. This differs from approaches such as VerbNet [12] – where the fine grained lexical semantic information is contained in lexical entries – in two respects. First, it supports the factorisation (through the ontological hierarchy) of information. Second, it directly permits reasoning with lexical concepts. Future work will concentrate on implementing, testing and optimising the algorithm defined in Figure 4 and on developing a grammar and a DL lexical ontology using existing resources such as WordNet [6], VerbNet and FrameNet [5]. Clearly, specifying such an ontology is accutely difficult. It is an important and necessary task however, if lexical reasoning is to be made possible at all, for any type of approach to lexicalisation of course but also for natural language processing in general.

References

[1] Baader, F., D. Calvanese, D. H. D. McGuiness and P. Patel-Schneider, "The Description Logic Handbook: Theory, Implementation and Applications," Cambridge, 2003.

[2] Bonfante, G., B. Guillaume and G. Perrier, *Analyse syntaxique lectrostatique* (2003), submitted for publication to Traitement Automatique des Langues.

[3] Brew, C., *Letting the cat out of the bag: Generation shake and bake*, in: *14th International Conference in Computational Linguistics*, 1992.

[4] Carroll, J., A. Copestake, D. Flickinger and V. Poznanski, *An efficient chart generator for (semi-)lexicalist grammar*, in: *Proceedings of the 7th European Workshop on Natural Language Generation (EWNLG'99)*, Toulouse, 1999.

[5] C.Johnson, C. Fillmore, M. P. C. Baker, M. Ellsworth and J. Ruppenhofer, *Framenet: Theory and practice*, Technical report, Berkeley (2002).

[6] Fellbaum, C., editor, "WordNet: An Electronic Lexical Database," MIT Press, 1998.

[7] Gardent, C., *Generating minimal definite descriptions*, in: *Proceedings of the 40th ACL*, Philadelphia, USA, 2002.

[8] Gardent, C. and L. Kallmeyer, *Semantic construction in FTAG*, in: *Proceedings of the 10th EACL*, Budapest, 2003.

[9] Gardent, C. and S. Thater, *Generating with a grammar based on tree descriptions: a constraint-based approach*, in: *Proceedings of the 39th ACL*, Toulouse, France, 2001.

[10] Goldman, N., *Conceptual generation*, in: *Conceptual Information Processing*, North Holland, Amsterdam, 1975 .

[11] Kay, M., *Chart generation*, in: *34th Annual Meeting of the Association for Computational Linguistics*, 1996.

[12] Kipper, K., H. T. Dang and M. Palmer, *Class based construction of a verb lexicon*, in: *Proceedings of AAAI-2000 Seventeenth National Conference on Artificial Intelligence*, Austin TX, 2000.

[13] Nirenburg, S. and I. Nirenburg, *A framework for lexical selection in natural language generation*, in: *Proceedings of the 12th International Conference on Computational Linguistics (COLING)*, Budapest, 1988.

[14] Reiter, E. and R. Dale, "Building Natural Language Systems," Cambridge, 2000.

[15] Stede, M., "Lexical Semantics and Knowledge Representation in Multilingual Sentence Generation," Ph.D. thesis, University of Toronto (1996).

[16] Stone, M., C. Doran, B. Webber, T. Bleam and M. Palmer, *Microplanning with communicative intentions: The SPUD system* (2001), working Report RuCCS TR 65. In submission.

[17] Volker Haarslev, R. M., *Description of the RACER system and its applications*, in: *Proceedings International Workshop on Description Logics (DL-2001*, Stanford, USA, 2001.

[18] Wanner, L. and E. Hovy, *The HEALTHDOC sentence planner*, in: *Seventh International Workshop on Natural Language Generation*, 1996, pp. 1–10.