Who's Next? From Sentence Completion to Conceptually Guided Message Composition

Michael Zock, Paul Sabatier and Line Jakubiec-Jamet

(L.I.F - CNRS - UMR 6166)
Laboratoire d'Informatique Fondamentale de Marseille
Université de la Méditerranée Faculté des Sciences de Luminy
163, Avenue de Luminy - Case 901 13288 Marseille Cédex 9 - France

Abstract. Natural language generation is typically based on messages and goals. We present here our views on how to help people to provide this kind of input, i.e. how to communicate thoughts to the computer, so that it could produce the corresponding surface-forms (sentences). The resource we are building is composed of a *linguistically motivated ontology*, a *dictionary* and a *graph generator*, whose respective functions are (a) guiding the user to make his choices concerning the concepts/words to build the message from, (b) to provide knowledge of how to link the chosen elements to yield a message (compositional rules), and (c) the visual display of the output, i.e. message graph representing the user's input. Our starting point is *Illico*, a system developed for French. Yet, being designed for sentence completion rather than message construction, it tends to drown the user by providing too many options, a shortcoming that we try to overcome via the mentionned ontology combined with a tool checking conceptual well formedness. In order to make our goal feasable (allow users to express freely ideas of any sort), we will start by limiting ourselves to two small domains: soccer and textbooks designed for learning French.

1 Introduction

Illico is a generic software tool designed for analysis and synthesis of natural language. To this end various applications have been developed, in particular: natural language interfaces for knowledge bases, a communication aid in French and German for disabled people [1], software for language rehabilitation and education of autistic children [2], linguistic games for language learning [3], support for simultaneous composition of sentences in Arab and French [4]. For more details about the system and its applications you may want to take a look at: `http://www.lif-sud.univ-mrs.fr/~paulsab/ILLICO/illico.html`. *Illico* provides tools and formalisms to modularly encode lexical, syntactic and semantic (conceptual and contextual) knowledge to account at the various levels for the well formedness of natural language expressions. Given this knowledge, *Illico* is able to perform any of the following tasks :

- analyze expressions (words, phrases, clauses, sentences, etc.) in terms of well-formedness and display the different representations associated with each one of them at the various levels (lexical, syntactic and semantic);

Zock M., Sabatier P. and Jakubiec-Jamet L. (2007).
Who's Next? From Sentence Completion to Conceptually Guided Message Composition.
In *Proceedings of the 4th International Workshop on Natural Language Processing and Cognitive Science*, pages 38-46
DOI: 10.5220/0002431700380046

– detect during analysis lexically, syntactically or semantically unexpected words, to synthesize the expected ones in inflected or lemmatized form;
– guide the composition of expressions in case of error or upon the user's request;
– avoid dead ends during the process of synthesis and guided composition. By dead end, we mean that the system comes to a halt, without being able to propose any candidate to continue or end the expression built so far.

Illico's engine is based on a mechanism of coroutine processing. Both for analysis and synthesis, it checks and executes the different constraints (lexical, syntactic and semantic) as soon as the structures of the different representations on which they apply are built, the process being dynamic and taking place in parallel. Representations are processed non deterministically in a top-down manner. This allows the system to analyse and to synthesize expressions simultaneously, and to guide composition incrementally, i.e. by partial synthesis.

Illico stops the analysis as soon as it encounters an unexpected expression, be it for reasons of lexical, syntactic or semantic incongruences, to propose in exchange all possible expressions fitting into this place and being in agreement with the constraints at hand. The implemented top-down strategy is powerful enough to allow for analysis and synthesis to be performed in a single pass. The same principle is used for composition. Hence, a sentence is composed step-by-step, left to right, *Illico* offering at each step a list of candidates for continuing the sentence built so far. Figure 1 (next page) illustrates this mode. The user having reached a certain state is waiting for suggestions by the system.

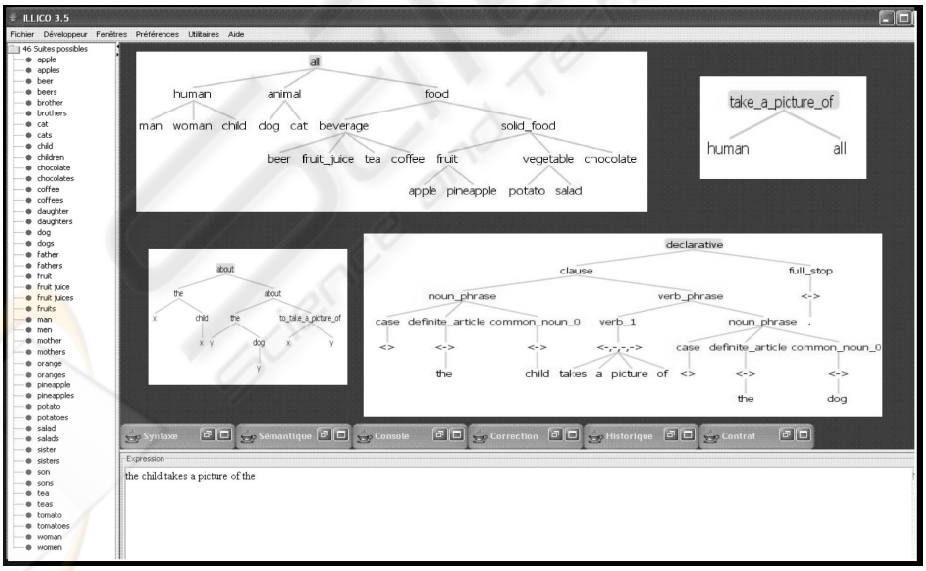

Fig. 1. Illico Screen Printing.

Guided composition of expressions (words, phrases, clauses, sentences, etc.) is one of the functionalities of *Illico*. But for the time being, the system is limited to sentence completion. We shall see in the remainder of this paper what else is needed in order to support guidance of message composition.

2 Limitation of Sentence Completion, or the Need of Controlling Conceptual Input

Sentence completion systems are different from conventional surface generators [5, 6] which start from a *goal* and a set of *messages* (input) in order to produce the corresponding surface form (output). Actually sentence completion systems are less than surface generators, which does not mean that they cannot be just as useful. Working quietly in the background, *Illico* tries to be pro-active, making reasonable guesses about what the author could say next. Hence, it supposes somehow, that the author knows at least to some extent what he is/was going to say[1]. It's main concern being to prevent the author to get blocked or stranded. To do so it offers a set of terms (words) which, pragmatically, semantically and linguistically speaking are possible continuations at this point in the chain. *Illico* works basically the following way. Given some input (typically one or several words) the system generates all possible consequences. To this end it relies on a formal grammar. Since the next item in the chain can only be one element out of many (the set, i.e. list of possible words), and since the system cannot guess the one the user has in mind, it will ask him. To keep the list reasonably small it relies not only on syntactic, but also on semantic and pragmatic knowledge, presenting only items that satisfy the specified constraints.

As one can see, unlike most parsers, *Illico* performs analysis by synthesis, and while it does not need any help from the user for analysis, it does so for synthesis, as, otherwise it would produce too many sentences, most of which do not correspond at all to the writers intention. Yet this is clearly not what we want. Figure 1 should give you a rough idea of how *Illico* works. You'll see basically three windows with, at the bottom, the output produced so far (generally an incomplete sentence, here: "the child takes a picture of the"); at the very left, the candidates for the next slot (apple, beer, brother, ...), and in the main frame, various kinds of representation, like the system's underlying ontology, pragmatic, semantic and syntactic information concerning the sentence in the making. While *Illico* has various nice features, like for example, the fact that you never get stuck, and that the produced outputs will always be well-formed at the various levels, it has also a few shortcomings, in particular with respect to the number of candidates displayed. Indeed, the candidate list (size of the set, or number of items from which to choose) could definitely be improved, (a) if the system presented at the top level *categories* (types) rather than *instances* (words), which should only be shown at the very end (leaf level of the ontology), (b) if frequency were taken into account (the likelihood of words varying with the topic), (c) if *governors* (eg. nouns) were presented before their *dependants* (eg. determiners, adjectifs), and (d) if *variables* were given rather than all possible morphological *values*.

[1] Of course, we can also assume, that the author does not even know that. But this is a bit of an extreme case.

Another problem linked to set size (i.e. great number of words from which to choose) is the fact that large sets tend to be distracting and to cause forgetting. Indeed, as the number of candidates grows (as is typically the case at the beginning of a clause), increases the danger to get drowned. Likewise, with the distance between the governor and its dependant increasing, grows the danger to end up producing something that, while in line with the language (Illico produces only well-formed sentences) does not correspond anymore to what one had in mind. Memory and divided attention having taken their toll. In order to avoid this, governing elements should always be determined first, and the set of data from which to choose should be kept small. In other words, filtering and navigation become a critical issue, and there are at least two ways to deal with them.

In order to reduce the number of candidates from which to choose, one can filter out linguistically and conceptually irrelevant material. This strategy is generally used both by the speaker and the listener as long as optimal transmisssion of information, i.e. reasonable input/output are considered as a major means to achieve a given communication goal (default case). Hearing someone say : "I'd love to smoke a....", our mind or ears will be "tuned" to smokeable items (cigar, cigarette, pipe) rather than to any noun, no matter how correct all of them may be from a syntactic point of view. With regard to our problem (sentence completion or message composition) this means, that the list to be presented should be small and contain but "reasonable" items[2]. How can this be achieved without sacrifizing coverage? Indeed, even a filtered list can still be quite large. Imagine that you were to talk about people, food, or a day of the year, etc. The number of representatives of each category is far too big to allow for fast identification, if the candidates are presented extensively in an unstructured, or only alphabetically structured list. This can be avoided, and navigation can conserably be eased by categorizing items, presenting them as a conceptually structured tree (type hierarchy) rather than as a flat list. Instead of operating at the concrete level of instances (all days of the year) the user will now operate (navigate or choose) at a much higher level, using more abstract words (generic concepts, type names, hypernymes) like month, weekdays, hour, etc. Of course, ultimately he will have to choose one of the concrete instances, but having eliminated rapidly, i.e. via categories, most of the irrelevant data, he will now choose in a much smaller list. The gain is obvious in terms of storage (at the inferface level) and speed of navigation.

3 Incremental Building and Refining a Message Graph

To show what we have in mind, take a look at figure 2 (preceding page). It is inspired by SWIM, an ontology-driven interactive sentence generator [10]. Let's see how this is meant to work. Suppose you were to produce the underlying message of the following sentence "Zidane hammered the ball straight into the net". This would require several walks through the conceptual network, one for each major category (Zidane, hammer,

[2] This idea is somehow contained in Tesnière's notion of *valency* [7], in Schank's *conceptual dependancy* [8] and McCoy and Cheng's *discourse focus trees* [9].

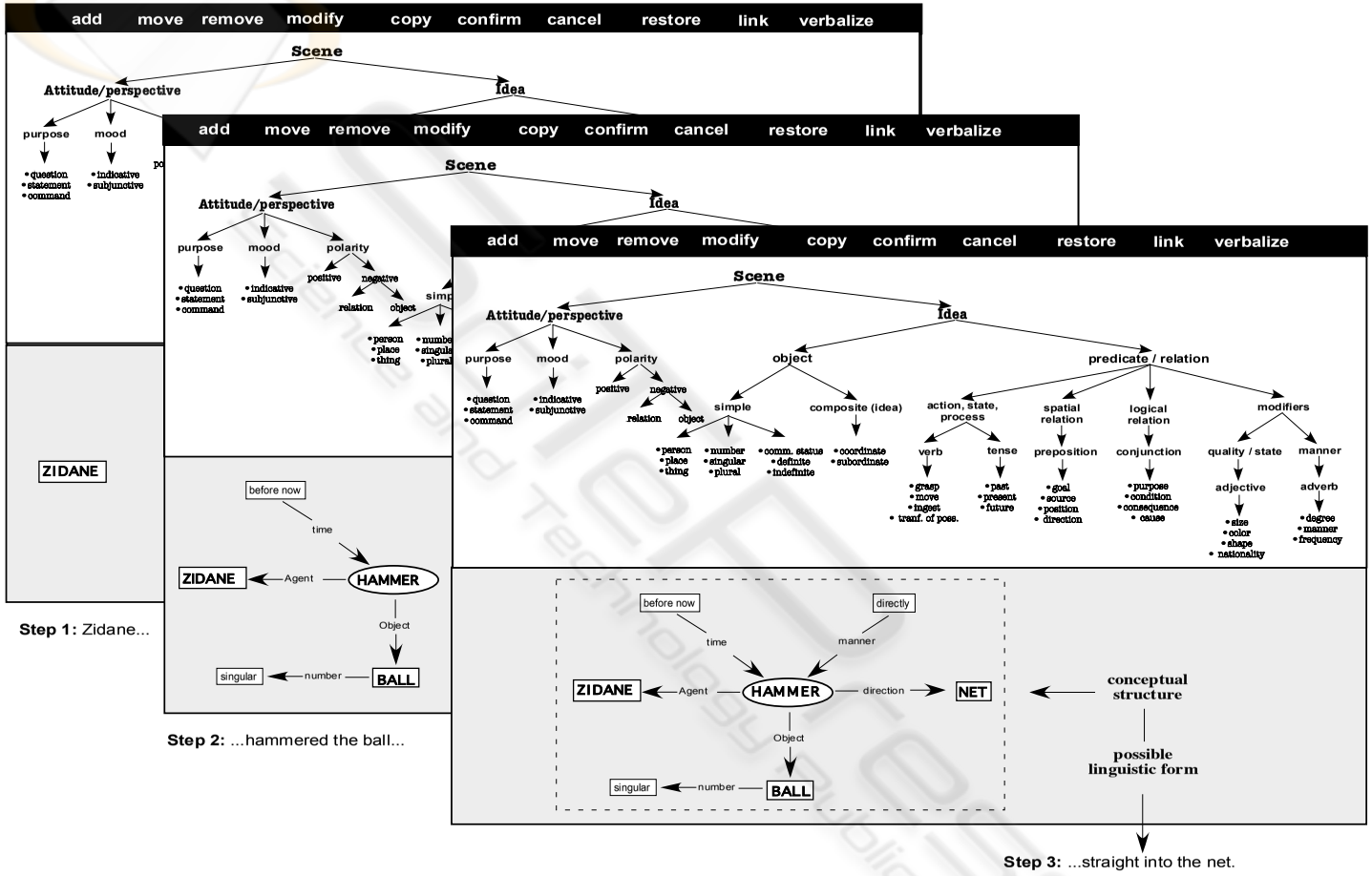

Fig. 2. An Interactive Graph Editor for Incremental Message Composition.

ball, straight, net)[3]. The path for "Zidane" would be "scene/idea/objects/entity/person", while the one for the shooting-act would be "scene/idea/relation/action/verb". Concerning this yet-to-be-built resource, various questions arise concerning the components (nature), their building and usage. The three problems are somehow related.

What are the components? In order to allow for the interactive building of the message graph we will need three components: a *linguistically motivated ontology*, a *dictionary* and a *graph generator*. The *ontology* is needed for guiding the user to make his choices concerning the elements to build the message from: concepts/words. The fact that the user has chosen a set of building blocks (concepts, i.e. class names, or words) does not mean that we have a message. At this point we have only a set of elements which still need to be connected to form a coherent whole (conceptual structure or message graph). To this end the system might need additional information and knowledge. Part of this can be put into the *dictionary*. Hence, *nouns* can be specified in terms of subcategorial information (animate, human, etc.), *verbs* in terms of case-frames and roles, etc. These kinds of restrictions should allow the connection of the proper arguments, for example, baby and milk, to a verb like *drink*. The argument connected via the link *agent* is necessarily *animate*, the information being stated in the lexicon. If despite all this, the system still cannot build the graph (suppose the user had given only the equivalent of two nouns and two adjectifs), it will engage in a clarification dialogue, asking the user to specify, which attribute qualifies which object. Once all objects (nodes) are linked, the result still needs to be displayed. This is accomplished via the *graph generator*, which parallely to the input displays incrementally the message structure in the making.

How to use the resource? Obviously, as the ontology or conceptual tree grows, increases access time, or more precisely, the number of elements from which to choose to reach the terminal level (words). In addition, the metalanguage (class names) will become more and more idiosyncratic. Both of these consequences are shortcomings which should definitely be avoided. This could be done in several ways: (1) allow the message to be input in the user's mother tongue. Of course, this poses other problems: lexical ambiguity, structural mismatches between the source and the target language; (2) start navigation at any level. Indeed, when producing a sentence like, "give me a cigarette", hardly nobody would start at the root level, to reach eventually the level of the desired object. Most people would immediately start from the hypernym or base level; (3) allow access via the words' initial letters, or, even better, (4) via associatively linked concepts/words: *apple* yielding *computers* which may prompt then *Java* (both a computer language and an island), *coffee* (product of Java) and finally *mokka*, the target word. For more details, see [11].

Last, but not least, there is no good reason to have the user give all the details necessary to reach the leaf-, i.e. word-level. He could stop anywhere in the hierarchy, providing details later on. This being so, he can combine breadth-first and depth-first strategies, depending on his knowledge states and needs. Obviously, the less specific the input, the larger the number of words from which we must choose later on. Yet,

[3] The upper part shows the conceptal building blocks, structured as a tree, and the lower part contains the result of the choices made so far, that is, the message built up to this point. To simplify matters we've ignored the attitude or speech-act node in the lower part of our figure.

this is not necessarily a shortcoming, quite to the contrary. It is a quality, since users can now decide whether they want to concentrate first on the big picture (general structure or frame of the idea) or rather on the low level details (which specific words to use). Full lexical specification is probably not even wanted, as it is not only tiresome as soon as the ontology grows (imagine the time it might take just to produce the conceptual equivalent to a message like a beer, please), but also it may pose problems later on (surface generation), as the words occuring in the message graph might not be syntactically compatible with each other. Hence, we will be blocked, facing a problem of expressibility.

4 Conceptual and Computational Issues

Building the kind of editor we have in mind is not a trivial issue and various problems need be addressed and solved :

- **coverage** : obviously, the bigger the coverage, the more complex the task. For practical reasons we shall start with a small domain (soccer), as we can rely already on a good set of resources both in terms of the ontology and the corresponding dictionary [12]. Kicktionary, developped by Thomas Schmidt (`http://www.kicktionary.de/Introduction.html`), is a domain-specific trilingual (English, German, and French) lexical resource of the language of soccer. It is based on Fillmore's Frame Semantics [13] and uses WordNet style semantic relations as an additional layer to structure the conceptual level.
- **language specificity** : there are good reasons to believe that the conceptual tree will be language dependant. Think of Spanish or Russian where verb-form depends on whether an action is completed or not (aspect), yielding two, morphologically speaking, entirely different root-forms (ser/estar, meaning *to be* in Spanish, or "uchodits" vs "uitsi" *to walk* in Russian).
- **ergonomic aspects (readability)** : the graph's readability will become an issue as soon as messages grow big. Imagine the underlying graph of a multiple embedded relative-clause. Also, rather than frightening the user by showing him the entire tree of figure 2, we intend to show only the useful (don't drown the user), for example, the children nodes for a given choice.
- **the limits of symbolic representation** : as shown elsewhere for *time* [14] and *space* [15], symbolic representations can be quite cumbersome. Just think of gradient phenomena like colors or sounds, which are much easier represented analogically (for example, in terms of a color-wheel), than categorially.
- **the problem of metalanguage** : we will discourage the user if, learning the target language is only possible by learning yet another (meta) language.
- **the conceptual tree** : there are bascially two issues at stake: which categories put into the tree and where to place them. Indeed, there are various problematic points in figure 2. Where shall we put negation ? Shall we factor it out, or put it at every node where it is needed ?

5 Checking Conceptual Well-formedness

Obviously, messages must be complete and well-formed, and this is something we would like to check in our future component. Suppose you were to make a comparison, then you must (at least at the beginning) mention the two items to be compared (completeness), and the items must be comparable (well formedness). In other words, having chosen some predicate, a certain number of specific variables or arguments are activated, waiting for instantiation. Yet, arguments are not only of a specific kind, playing a given role, they also have specific constraints which need to be satisfied.

While checking well-formedness for single words does not make sense (apart from spell checking, which is not our concern here), it does make sense to check the compatibility and well-formedness of the combination of concepts or words, to see whether they produce an acceptable conceptual structure. In the case of our example, figure 2, this means that, having received as input something like to shoot (or, to hammer), we know that there is someone, performing this action, with a specific target in mind (the goal), and that the action can only be performed in so many ways (manner). While not all of this information is mandatory, some of it is (agent, object, target), and there are definitely certain constraints on the various arguments (the agent must be animate, the object, some kind of sphere, typically used in soccer games, etc.). Being formalized and stored in a conceptual dictionary, this information can now be used by our system to check the well formedness of a given structure and its compatibility with the user's input. The conceptual information being organized hierarchically, we use Coq [16], that is, a typed λ-calculus providing a framework with a rich type system. Hence, the dictionary entries might look as follows :

```
Parameter hammer : Agent -> Object -> Target -> Prop.
Parameter Zidane : human.
Parameter ball: soccer_instrument.
Parameter net: soccer_equipment.
```

$Prop$ stands in Coq for the type of proposition ; roles ($Agent$, $Object$, $Target$) and features ($human$, $soccer\_instrument$, $soccer\_equipment$) are generic types. We must also express conceptual constraints for the various types, for example, $Agents$ must be $animate$. Coq uses the subtype principle (inheritance graph) to check that all constraints are satisfied, defining $human$, $soccer\_instrument$ and $soccer\_equipment$ respectively as subtypes of $Agent$, $Object$ and $Target$. When all the constraints are satisfied, we have what it takes to represent the semantics of a sentence, which in our case would be something like, "there is an agent who did something in a specific way, by using some instrument." In other words : there is a person, an object and a target (let us call them respectively a, b and c), the three arguments being linked via an action that is being performed in a specific way. This general idea can be described as follows :

```
Message definition := exists a, exists b, exists c,
     is_someone a /\ is_something b /\ is_manner c /\ relation a b c.
```

Hence, in order to produce the global message *Zidane hammered the ball straight into the net*, we must instantiate the composite propositions respectively by is_Zidane of type $human \rightarrow Prop$, is_ball of type $soccer\_instrument \rightarrow Prop$, is_net of type $soccer\_equipment \rightarrow Prop$. $Hammer$ is already declared. Once this is done, the parameters $Zidane$, $ball$ and net can be applied to produce the desired result, the system type-checking the compatibility of the involved parameters. In other words, checking the conceptual well-formedness and consistency of the messages amounts basically to type-checking the elements of which the message is composed.

6 Conclusion and Perspectives

We have presented and discussed a practical solution to one of the shortcomings of an existing system. To improve its conceptual component and to allow for conceptually guided message composition we suggested to build a *linguistically*

motivated ontology combined with a dictionary and graph generator. While there are many ontologies, we cannot draw on them directly, because they were simply not built for message composition. Strangely enough, a lot of relevant work has been produced outside the area of language generation, yet it has been largely ignored. Many great minds have devoted their energy to design an interlingua or universal language, Ramon Lullus, Leibniz, Descartes or bishop Wilkins, to name just those. We will certainly take a closer look at this work, as in spite of its age, it might contain highly useful information for our future work.

References

1. Guenthner, F., Thielmann, K., Pasero, R., Sabatier, P.: Communications aids for als patients. In: Proceedings of the Third International Conference on Computers for Handicapped Persons. (1992) 303–307
2. Godbert, E., Mouret, P., Pasero, R., M, M.R.: A software for language rehabilitation and education of autistic-like children. In ACL, ed.: Proceedings of the Workshop on Natural Language Processing for Communication Aids. (1997) 59–64
3. Pasero, R., Sabatier, P.: Linguistic games for language learning: A special use of the illico library. Computer Assisted Language Learning **11** (1998) 561–585 Published by *Swets & Zeitlinger*.
4. Boïde, O., Pasero, R., Sabatier, P.: Un système de composition simultanée de phrases en arabe et en français. In: Proceedings of the Arabic Translation and Localisation Symposium, ATLAS. (1999) 103–109
5. Reiter, E., Dale, R.: Building Natural Language Generation Systems. Cambridge University Press (2000)
6. Bateman, J., Zock, M.: Natural language generation. In Mitkov, R., ed.: Oxford Handbook of Computational Linguistics. Oxford University Press (2003) 284–304
7. Tesnière, L.: Éléments de syntaxe structurale. Klincksieck, Paris (1959)
8. Schank, R.: Conceptual dependency theory. In Schank, R.C., ed.: Conceptual Information Processing. North-Holland and Elsevier, Amsterdam and New York (1975) 22–82
9. McCoy, K., Cheng, J.: Focus of attention: Constraining what can be said next. In Paris, C., Swartout, W., Mann, W., eds.: Natural Language Generation in Artificial Intelligence and Computational Linguistics. Kluwer Academic Publisher, Boston (1991) 103–124
10. Zock, M.: Swim or sink: the problem of communicating thought. In Swartz, M., Yazdani, M., eds.: Intelligent Tutoring Systems for Foreign Language Learning. Springer, New York (1991) 235–247
11. Ferret, O., Zock, M.: Enhancing electronic dictionaries with an index based on associations. In: ACL '06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL, Morristown, NJ, USA, Association for Computational Linguistics (2006) 281–288
12. Sabatier, P.: Un lexique-grammaire du football. Lingvisticæ Investigationes **XXI** (1997) 163–197
13. Baker, C.F., Fillmore, C.J., Lowe, J.B.: The berkeley framenet project. In: COLING/ACL-98, Montreal (1998) 86–90
14. Ligozat, G., Zock, M.: How to visualize time, tense and aspect. In: Proceedings of COLING '92, Nantes (1992) 475–482
15. Briffault, X., Zock, M.: What do we mean when we say to the left or to the right? how to learn about space by building and exploring a microworld? In: 6th International Conference on ARTIFICIAL INTELLIGENCE: Methodology, Systems, Applications, Sofia (1994) 363–371
16. Coq Development Team (Logical Project): The Coq Proof Assistant. Reference manual. INRIA (2004-2006)