# Message Composition Based on Concepts and Goals

**Article** *in* International Journal of Speech Technology · December 2008

DOI: 10.1007/s10772-009-9050-8



**3 authors**, including:

Michael Zock
French National Centre for Scientific Research
**92** PUBLICATIONS **454** CITATIONS

SEE PROFILE

Int J Speech Technol (2008) 11: 181–193
DOI 10.1007/s10772-009-9050-8

# Message composition based on concepts and goals

**Michael Zock · Paul Sabatier · Line Jakubiec**

Received: 15 July 2009 / Accepted: 13 October 2009 / Published online: 19 November 2009


**Abstract** The goal of this paper is to deal with a problem hardly ever addressed in natural language generation, *conceptual input*. In order to be able to express something, one needs to have something to express to begin with: ideas, concepts and thoughts. The question is how to access thoughts and build their representation in form of messages. What are the building blocks? How to organize and index them in order to allow for quick and intuitive access later on?

It is generally believed that ideas precede expressions. Indeed, meanings, imprecise as they may be, tend to precede their expression in language. Yet, message creation is hardly ever a one-step process. Conceptual inputs are generally abstract and underspecified, which implies that they need to get refined later on, possibly even during the expression phase.

In this paper we investigate *interactive sentence generation*, the focus being on conceptual input, a major component of language generation. Our views will be illustrated via three systems: ILLICO, a system for analyzing/generating sentences and guiding their composition; SPB, a *multilingual phrase-book* with on the fly generated audio output and *Drill Tutor* (DT), an exercise generator. While ILLICO is a message-understanding system with a *message-completion* functionality, SPB and DT are *message-specification systems*. The user works quite early with fairly complete structures (sentences or patterns), making basically only local changes: replacing words in the case of SPB, and choosing them to instantiate pattern variables in the case of DT.

M. Zock (✉) · P. Sabatier · L. Jakubiec
Laboratoire d'Informatique Fondamentale de Marseille, CNRS & Aix-Marseille Université, 163, Avenue de Luminy, Case 901, 13288 Marseille Cedex 9, France
e-mail: michael.zock@lif.univ-mrs.fr

**Keywords** Conceptual input · Message composition · Message, completion · Interactive natural language generation

## 1 Introduction

The goal of this paper is to investigate a problem hardly ever addressed in the literature of natural language generation, *conceptual input*. Being one of the major sources of a generator, this is clearly an important element, especially if input is to be given by a human user. Unfortunately, little is known about issues such as:

- *Nature of the input and compositional rules*: words and sentences are linguistic means for expressing ideas, uttered to achieve a communicative (non-linguistic) goal. If we can agree that the inputs are messages and goals, we still need to clarify the nature of the building blocks (concepts, primitives), the rules of combining them (conceptual grammar), and the means to present them at the interface level: icons, semantic networks, ordinary words, or words having the status of concepts or primitives (interlingua), or any combination of them (hybrid approach);
- *Specificity of input*: inputs are rarely the specific words used in a concrete sentence. Depending on the situation, messages are encoded at various levels of abstraction (semantic networks, syntactic trees). While messages may be fully specified in the case of spontaneous discourse (dialogue), they certainly are much less specific at the early stages of text production, in particular if the message is very long, challenging our short-term memory capacities (Miller 1956). The question is: what are the core message

elements, typically present in a message,[1] and how close are they to language;
- *Moment of the input*: obviously input precedes output. Yet, things can be quite subtle. Not everything is necessarily fully specified or definitely encoded before initiating expression. During delivery (speaking or writing) we may change our thoughts. Realizing that our output does not fit our goal well, we may alter the initial message to a greater or lesser extent, adding or deleting details as a result of afterthoughts. In sum, conceptual input need not be fully specified prior to *expression*, it may also occur during expression or afterwards (feedback, corrections), language feeding back on thought;
- *Representation*: message elements and their combination can be represented in various ways: categorially or visually (graphs, trees, menus). Storage and representation are relevant both for the long-term memory and for the working memory (intermediate structure). For example, words can be stored permanently in a dictionary, or temporarily in a graph representing the message. Graphs are useful mnemonic devices as long as the message they represent is not too complex. They allow us to see and remember what he has been encoded or built so far, helping us to realize at the same time what is still lacking;
- *Planning unit*: atomistic (concepts) vs. holistic (messages, patterns); the planning units can be of various sizes. We tend to consider words as such units, but words can be broken down into elements (meaning, form, sound), just as sentences can be divided into smaller units (phrases, words). Yet, sentences and their underlying conceptual structure (patterns) can also be considered as tokens or units, in which case they don't need to be synthesized. Just like words in a dictionary, they only need to be accessed. In other words, they behave like lexicalized phrases;
- *Building strategies*: messages can be produced in various ways, step by step, i.e. incrementally, or holistically, in which case they are just retrieved. Processing can also be performed hierarchically, from top to bottom (a general idea is fleshed out with details) or from the bottom to the top (words are integrated into a larger structure). Input can be given at once (one-shot process) or in various steps, with backtrack (corrections, refinements) or not. Finally, processing can be done serially, i.e., left to right, or via a mixed approach.

Natural language generation is typically driven by input messages, i.e., meanings, and goals, the usual starting points. We present here our views on how to help people provide this kind of input. Our views will be illustrated via three systems: ILLICO, a system that was built in the 90s, and two more recent systems which are still in the development phase and have not been tested yet by users: SPB, a multilingual speaking phrase-book, and DT, a web-based exercise generator, called 'Drill Tutor'.

While all these systems have been created for assisting the text producer, their design or blueprints have been guided by different principles and goals. One function of ILLICO is to guide the user to compose a message. In order to do so it checks both completeness and well-formedness of the message according to the context. The goal of the other two systems is language learning. With SPB we pursue two goals: helping users survive in a new culture or language, by providing them with translation equivalences, and helping them discover the underlying principles for building such sentences. The goal of DT is to help people become fluent in a new language. All three systems address the issue of message planning, but they do so by starting from different assumptions and inputs: words, concepts, or goals. While ILLICO is a message and sentence-completion system, SPB and DT are message-specification systems.

## 2 Message- and Sentence-Completion

Guiding a user to compose sentences can be done in many ways, much depending on the user's knowledge state. The problem we are dealing with here is search. Obviously, knowledge available at the onset (cognitive state) plays a very important role in this kind of task, regardless of the goal (determine conceptual input, lexical access, etc.). Search strategies and relative ease of finding a given piece of information (concept, word) depend crucially on the *nature of the input* (available knowledge) and the *distance* between this input and a given output (target word). Imagine that your target word is 'guy', a colloquial term for 'person'. In principle you could start searching from any of the following inputs: 'person' (more general term), 'cat', 'fellow' (synonyms), or 'gars' (equivalent word in French). Obviously, the type of search and ease of access would not be the same.[2]

### 2.1 Brief Introduction of ILLICO

We briefly present ILLICO (Pasero and Sabatier 2007), focusing on one of its functionalities: guided composition. ILLICO can be helpful in many ways. For example, it can analyze and synthesize expressions (words, phrases, clauses,

[1] A rough first guess would be the concepts underlying the major syntactic categories (nouns, verbs, adjectives, adverbs): entities, actions, processes, qualities.

[2] The nature and number of items among which to choose would be different in each case. The influence of formally similar words is well known, the Spanish word 'libreria' being likely to evoke the English word 'library'. Cognates tend to prime each other, a fact that, depending on the circumstances, can be helpful or sheer nuisance.

sentences), as well as offer possible continuations for a sentence that has not been completed yet, whatever the reasons may be (lexical, syntactic, semantic, conceptual or contextual).[3] Suggestions made by the system are possible expressions in line with the available constraints. To achieve this goal a powerful mechanism is used, processing in parallel all the knowledge typically needed.

Written in Prolog, 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, semantic, conceptual and contextual) 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. The implemented strategy allows for analysis and synthesis to be simultaneously performed in a single pass. The same principle is used for guiding composition incrementally, i.e., by means of partial synthesis.

To compose a sentence step-by-step, from left to right, ILLICO automatically and dynamically offers at each step a list of candidates for continuing the sentence built so far. Figure 1 illustrates this mode. Having reached a certain point in the chain (in our case: "The child takes a picture of the…") the system suggests possible continuations, satisfying the syntactic, conceptual and pragmatic constraints, explicitly and implicitly contained in the sentence fragment composed so far. It should be noted that ILLICO's goal consists in getting the user to compose (only) sentences it can understand, that is, sentences it is able to parse (Pasero and Sabatier 1995).

Offering possible continuations is but one way among many to assist a user in sentence composition. One can imagine a richer kind of assistance where the user accesses various kinds of knowledge (linguistic, conceptual, etc.) to select the one fitting best his communicative goals.

### 2.2 The Problem of Unstructured Lists, or the Way to Speed Up Conceptual Input

The objectives of *sentence-completion systems* are different from those of conventional surface generators (Reiter and Dale 2000; Bateman and Zock 2003). Surface generators start from a *goal* and a set of *messages* (input) in order to produce the corresponding surface form (output). 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.

ILLICO performs analysis by synthesis (top-down strategy), and while it does not need any help from the user for analysis, it certainly does so for synthesis. Otherwise it would produce unreasonably large sets of possible continuations, most of which are completely outside of the users' scope, i.e. his or her intention. Figure 1 should give the reader a rough idea of how ILLICO works. We can see basically three frames (A, B, and C–F) with at the bottom (A), the output produced so far (generally an incomplete sentence, in our case: *the child takes a picture of the* …); at the very left (B), the candidates for the next slot (*apple*, *beer*, *brother*, …), and in the main frame (C–F), various kinds of representation, such as the system's underlying conceptual ontology (C), semantic (D), logic (E), and syntactic representation concerning the sentence still in the making (F).

Offering rich assistance during sentence composition was not the main goal of the designers of ILLICO. The production of possible continuations is only one functionality among others, though a very useful one.[4] It is easily implemented due to the power of Prolog's core mechanisms.

A system providing more sophisticated assistance would look different: (a) the nodes of the tree in screen A would be *categories* (types) rather than *instances* (words), the latter being shown only at the leaf-level; (b) frequency would be taken into account (the words' likelihood varying with the topic); (c) *governors* (e.g. nouns) would precede their *dependants* (e.g. determiners, adjectives); and (d) *variables* would be used rather than extensive lists of possible morphological *values*, etc. We hope that the user will share (or, at least, understand) our conceptual views and the implicit metalanguage used for displaying information.

The goal of all these features is to limit the alternatives (set size), as having to choose among (too) many candidates causes distraction, mental fatigue (consumption of time and energy) and forgetting (we don't remember anymore what we were looking for). Indeed, as the number of lexical candidates grows (as is typically the case at the beginning of a clause), grows the danger of getting "drowned", distracted or to "downslide". Likewise, with the distance between the governor and its dependant growing, increases the danger to produce something that, while linguistically correct, does not correspond anymore to what one had in mind (memory and divided attention having taken their toll). In order to avoid this, we suggest determining the governing elements first, and keeping the set of data from which to choose small. In other words, filtering and navigation become critical issues, and there are at least two ways to deal with them.

[3] For more details and references concerning ILLICO and its applications (natural language interfaces to knowledge bases, simultaneous composition of sentences in different languages, linguistic games for language learning, communication aid for disabled people, software for language rehabilitation, etc.) the reader may want to consider http://www.lif.univ-mrs.fr/illico.html.

[4] For example, it allows the testing of well-formedness and linguistic coverage of the application one is about to develop. Therefore we can check now whether all the expected continuations are produced, and whether the produced strings are correct.

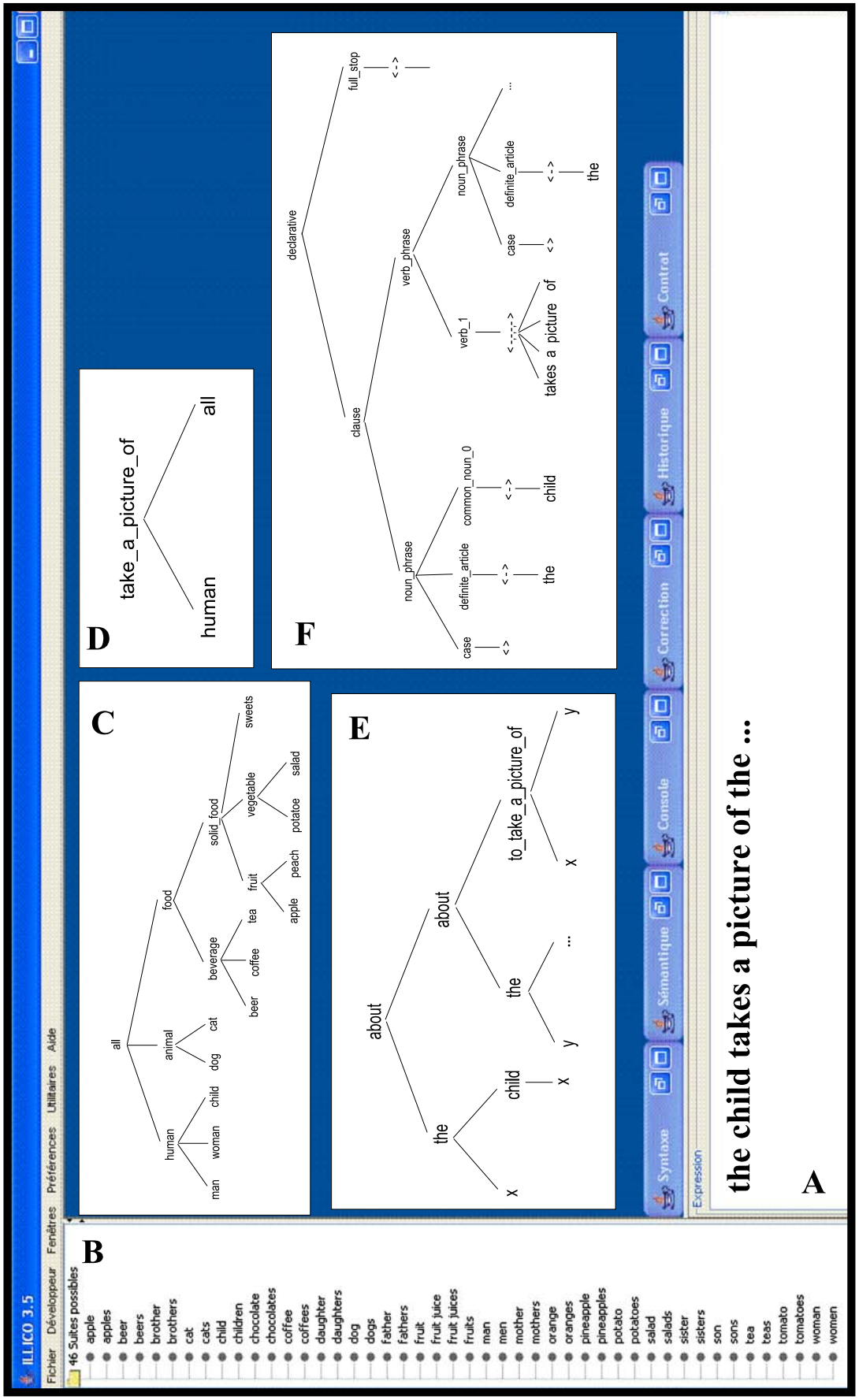


**Fig. 1** Screenshots of ILLICO in action: the pictures (**A**–**F**) show respectively the string built so far (**A**), possible lexical candidates (**B**), part of the conceptual ontology (**C**), the message to be expressed (**D**), the logical form (**E**), and the parse-tree of the complete sentence (**F**)

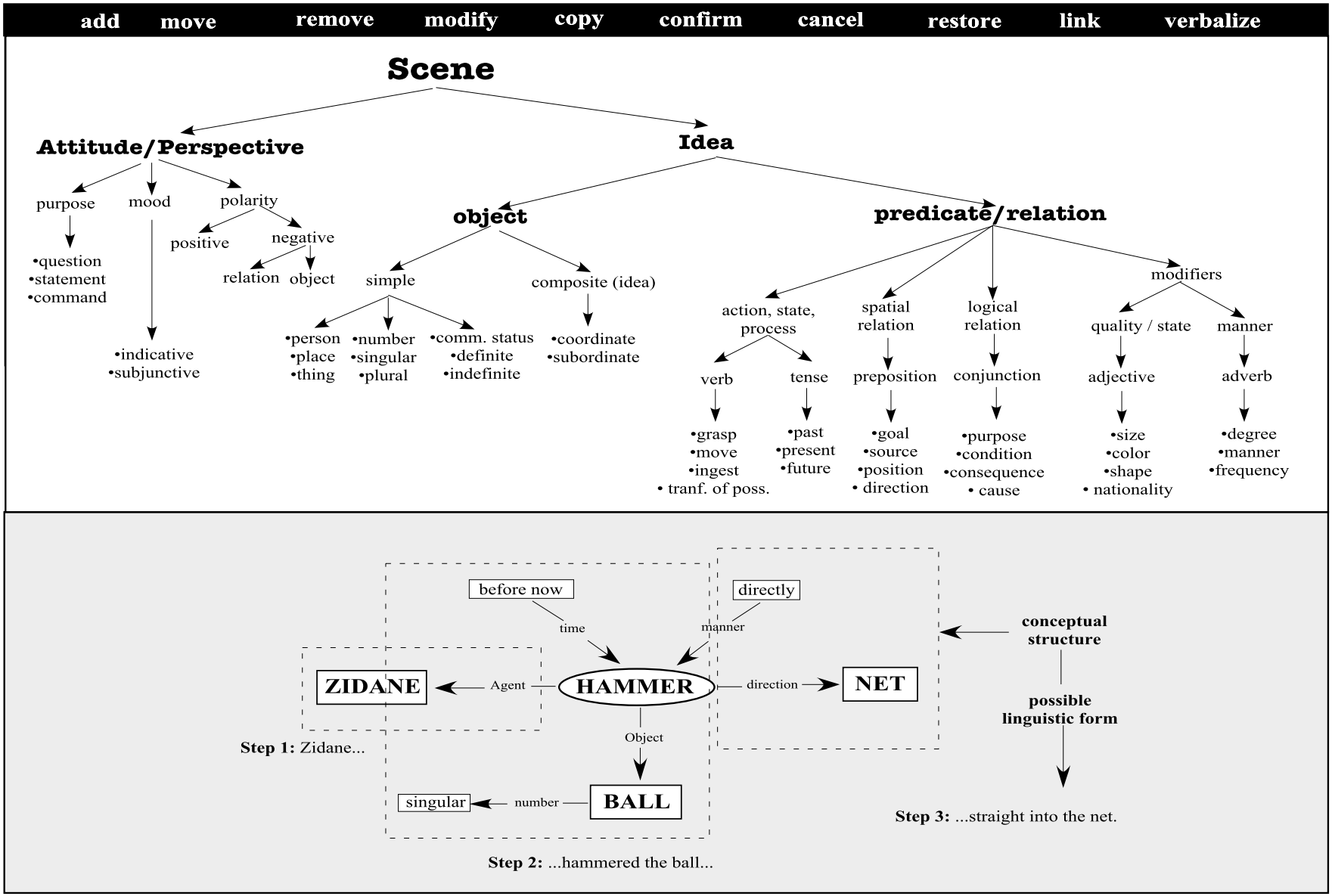


**Fig. 2** An interactive graph editor for authoring, i.e., incremental message composition. The figure illustrates the way how the underlying representation (graph) of the sentence 'Zidane hammered the ball straight into the net' is built in three steps. On *top* of the figure (here to the *left*) you see the conceptual building blocks (ontology), and at the *bottom* (here, to the *right*), the current state(s) of the incrementally built graph. To simplify matters, we've ignored in the graph the speaker's attitude (the speech-act representation)

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 transmission 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 just 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, structured by types or categories (apple, pear → fruit) and contain but "reasonable" items, that is, items in line with the constraints of the string produced so far.[5] How can this be achieved without sacrificing 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 ordered list.

This challenge can be avoided, and navigation can considerably 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, hypernyms) like month, weekdays, hour, etc. Ultimately s/he will have to choose one of the concrete instances, but having eliminated rapidly, i.e., via categories, most of the irrelevant data, s/he will choose from a much smaller list. The gain is obvious in terms of storage (at the interface level) and speed of navigation.

### 2.3 Incremental Building and Refining of a Message Graph

To show what we have in mind, take a look at Fig. 2. It is inspired by SWIM, an ontology-driven interactive sentence generator (Zock 1991).[6] How is this 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, ball, straight, net). The path for 'Zidane' (step 1) would be "scene/idea/objects/entity/person", while the one for the 'shooting-act' (step 2) would be "scene/idea/relation/action/verb". Concerning this yet-to-be-built resource, various questions arise with respect to the components (nature), their composition 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 first and last elements can be seen in Fig. 2. The *ontology* is needed for guiding the user to make their 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, 'Zidane' and 'ball', to a verb like *hammer*. 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 adjectives), 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 parallel to the input displays incrementally the message structure in the making.

How should the resource be used? Obviously, as the ontology or conceptual tree grows, increases also access time, or more precisely, the number of elements from which to choose in order 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. Having a multilingual dictionary allows for doing so in the users' mother tongue or any of the languages known by the resource, which would considerably speed up conceptual input. 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 anyone would start at the root level, to reach eventually the level of the desired object. Most people would immediately start from the hyperonym or base level;

[5] This idea is contained to some extent in Tesnière's *valency theory* (Tesnière 1959), in Schank's *conceptual dependancy* (Schank 1975) and in McCoy and Cheng's notion of *discourse focus trees* (McCoy and Cheng 1991).

[6] For a somewhat similar approach see Power et al. (1998).

(3) allow access via the words' initial letters, or, even better, (4) via associatively linked concepts/words.[7]

Finally, 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 the user must choose later on. However, this is not necessarily a shortcoming, quite to the contrary. It is a good thing, 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 (e.g., during surface generation), as the words occurring in the message graph might not be syntactically compatible with each other. Hence, we will be blocked, facing a problem of expressibility (Meteer 1992; Nicolov and Mellish 2000).

### 2.4 Conceptual, Computational and Psychological Issues

Building the kind of editor illustrated in Fig. 2 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 (Sabatier 1997). Kicktionary, developed by Schmidt (2007), is a domain-specific trilingual (English, German, and French) lexical resource of the language of soccer (http://www.kicktionary.de/Introduction.html). It is based on Fillmore's Frame Semantics (Baker et al. 1998) and uses WordNet style semantic relations (Fellbaum 1998) as an additional layer to structure the conceptual level.
- *Language specificity*: there are good reasons to believe that the conceptual tree will be language dependent. Think of Spanish or Russian where verb-form depends on *aspect*, that is, on the speaker's choice of considering an action as completed, i.e perfective, or not, yielding two, morphologically speaking, entirely different lemmas (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 multiply embedded relative-clause. Also, rather than frightening the user by showing him the entire tree as in Fig. 2, we will show only the useful parts (don't drown the user), for example, the children nodes for a selected node.
- *The limits of symbolic representation*: as shown elsewhere for *time* (Ligozat and Zock 1992) and *space* (Briffault and Zock 1994), 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 basically two issues at stake. Which categories to put into the tree and where to place each one of them. Indeed, there are various problematic points in Fig. 2. Where shall we put negation? Shall we factor it out, or put it at every node where it is needed?

There are several other issues, relevant and important for natural language generation in general, and interactive language generation (our case) in particular. We have mentioned some of them already in the introduction:

- *Representation*: in what terms to encode the message (concepts vs. words) and at what level of abstraction (general vs. specific)?;
- *Size of the planning unit*: concepts vs. messages;
- *Processing strategy*: is planning done as a one-shot process or is it performed in various steps, i.e., incrementally, possibly allowing for corrections?;
- *Direction*: is planning done left to right or top to bottom?;
- *Processing order*: Do thoughts precede language, and if so, is this always the case?

We will touch upon these points here only very briefly. For more details see Zock (1996), where a worked out example is giving, showing how language may feedback on

[7] Suppose you were looking for the word *mocha* (target word: $t_w$), yet the only token coming to your mind were *computer* (source word: $s_w$). Taking this latter as starting point, the system would show all the connected words, for example, *Java, Perl, Scala, Prolog* (programing languages), *mouse, printer* (hardware), *Mac, PC* (type of machines), etc. querying the user to decide on the direction of search by choosing one of these words. After all, s/he knows best which of them comes closest to the $t_w$. Having started from the $s_w$ 'computer', and knowing that the $t_w$ is neither some *kind of software* nor a *type of computer*, s/he would probably choose *Java*, which is not only a *programming language* but also an *island*. Taking this latter as the new starting point s/he might choose *coffee* (since s/he is looking for some kind of *beverage*, possibly made from an ingredient produced in Java, coffee), and finally *mocha*, a type of *beverage* made from these beans. Of course, the word *Java* might just as well trigger *Kawa* which not only rhymes with the $s_w$, but also evokes *Kawa Igen*, a Javanese volcano, or familiar word of *coffee* in French. For more details, see Zock and Schwab (2008).

thought, the lexicon acting as interface between language (output) and thought (input). Suppose you wanted to produce the following sentence: *When the old man saw the little boy drowning in the river, he went to his canoe in order to rescue him.* Obviously, before producing such a sentence its content must be planned and represented somehow, but how is this done? There are several good reasons to believe that this sentence has not been planned entirely in advance, neither from left to right, nor in a single pass. There are both psychological and linguistic reasons supporting these claims.

*Psychological reasons*: The sentence is simply too long for a speaker to hold all its information in short-term-memory. It is highly unlikely that the speaker has all this information available at the onset of verbalization. The need for planning, that is, the need to look ahead and to plan in general terms, increases with the length of the sentence and with the number and type of embeddings (for example, center embedded sentences). There is also good evidence in the speech error literature supporting the claim that people plan in abstract terms. False starts or repairs, like "I 've turned on the *stove* switch, I mean the *heater* switch" suggest that the *temperature increasing device* has been present in the speakers mind, yet at an abstract level (Levelt 1989; Fromkin 1993).

*Linguistic reasons*: as is well known, the order of words does not necessarily parallel the order of thought. For example, the generation of the first word of the sentence here above, the temporal adverbial "when", requires knowledge of the fact that there is another event taking place. Yet, this information appears fairly late in the sentence.

## 3 Two Alternatives: Message Specification Systems

Message completion as described for ILLICO is a special case of conceptual input, as the input is provided fairly late in the process, namely during the formulation phase. Actually, in the systems described here below we don't compose messages, at least not from scratch. What we do is complete an existing sentence fragment. The most frequent cases are probably situations where the author knows beforehand, perhaps only in crude terms, what he wants to convey. This is particularly true in spontaneous discourse where sentences are short, but to a lesser degree for writing, which needs to be planned at various levels of abstraction (macro-plan, micro-plan). We will describe here two other systems (SPB and DT), providing new challenges, but also new possibilities for helping people to provide conceptual input. They have been developed separately, but they will be used in conjunction. Since both systems are still in the prototype phase, they have not been evaluated yet.

### 3.1 Context and Problem

To be able to communicate in a foreign language is vital in our globalized world, yet this is a complex task which needs to be learned. We will restrict ourselves here to speaking, i.e., translation of ideas into sentences and speech. More precisely, we will be concerned with language learning: Asian (for the time being Japanese, but later also Chinese) and Indo-European languages (for the time being English and French). The scope will be the *survival-level*, i.e., acquisition of the basic words and structures to be able to survive in a new culture and to become relatively autonomous in this new environment.[8] In sum, we'd like to help someone learn the basic stock of phrases and expressions that are generally taught in the classroom, or that are acquired via a phrase book, the latter being structured by tasks a tourist is likely to perform: ask for information in public places (post office, airport), do shopping, etc. Yet, we would like to go one step further and help the learner not only learn by heart the form of a stock of phrases, but also (or, more importantly) the underlying principles (structures) to build similar sentences. To reach this kind of open-ended generativeness we propose an electronic version of two well-known methods, a multilingual phrase-book and pattern drills. There are two good reasons for this:

- If speaking in one's mother tongue is a daunting task, to do so in a foreign language can be overwhelming. In order to gain control, it is good to divide the task and automate part of it. Yet, this requires practice. Without the latter we will not only forget, do things in the wrong order (what should be done when?) and end up being incapable of delivering the expected result in time;
- Different people have different needs, and needs tend to change. This being so, we propose an open system, allowing the user to tailor it to his or her liking.

In sum, we propose the building of an exercise generator to help people develop basic communication skills and become fluent in a foreign language. The goal is to assist the *memorization* of words and sentence patterns, and help people acquire the needed skills, *automatisms* to access and to instantiate them, such as to be proficient enough to participate in a simple conversation.

### 3.2 A Multi-lingual Phrasebook Able to Speak

To achieve the above-mentioned goal we have started building a multilingual phrasebook (SPB) to convert meanings into speech. The idea is to provide a web-based service helping people reach the survival level in the new language.[9]

[8] For a similar goal, but with a quite different method, see Boitet et al. (2007).

[9] For a similar goal, but with a quite different method, see Boitet et al. (2007).

More precisely, we aim at building an open, generic, customizable multilingual phrase-book allowing not only to see and hear the translation of a given sentence in various languages (currently Japanese, English, French), but also to learn the language and the underlying principles for producing fluently such kind of sentences.

The core element is a multi-lingual data-base composed of sentences occurring in specific discourse situations, that is, sentences, typically used to achieve specific communicative goals: in a shop or in a restaurant, at the bank, train station, police station, etc. The base has been built for the three languages. It is open and meant to grow. The current base is still quite small, containing only about 10,000 sentences and their translation into the other two languages, but we expect to have about 40,000 sentences per language by the end of 2010. To allow the user to find quickly the information he is looking for, we have started to structure the database in terms of domains and scenarios and we have added indexing terms (lemmata, goals). Hence, navigation (input) can be done via any of these categories (and their combination): the system will display all the sentences containing a given word, sentences being part of a scenario or realizing a specific goal. The system is open in the sense that the user can add sentences or terms for indexing them.

In addition to the database, the system needs two other elements: a multi-lingual dictionary and a speech synthesizer. Only the latter exists at present.

The dictionary should contain not only translation equivalences, but also relational information concerning the (nature of the) links between words. In particular it should contain hypernyms. One of the many benefits of this kind of information is, that it allows replacement of a specific term (word instance) by a more general term (dog—animal). This being so, we can now go beyond the information given and replace concrete data (instances: words and sentences) with more abstract forms (hypernyms, patterns). In other words, with this method we can convert a sentence-database into a pattern library. This is a very valuable asset, as patterns can now be used to extract other instances or example sentences in a given corpus (Hearst 1998), growing the database.[10] Patterns can also be used to guide and constrain conceptual input, especially if they are linked to goals (see below 'Drill Tutor'). For example, 'Didn't you know that *person* love *sweet*?', allowing the creation of a sentence like 'Didn't you know that kids love chocolate?'.

Of course, there are numerous phrasebooks, but they are all closed and hard-wired. For example, unlike commercial systems, we do not store audio-files, sound is generated on the fly. The Japanese colleagues with whom we do this research have a speech synthesizer for the three languages mentioned here above.

Phrasebooks are interesting for learners for several reasons:

- they allow quick building of a huge translation memory: a set of phrases and their equivalence in the target language;
- words appear immediately in the context of a sentence, which reveals their usage, fosters fluency and increases their memorizability: co-occurring words strengthen each others links, increasing the likelihood of evoking each other later on (priming);
- sentences are, at least implicitly linked to a domain or goal.

Meaning is conveyed directly or in several steps by choosing and possibly refining any of the existing sentences. Access can be performed via the above-mentioned methods (words, goals, domains). Meanings can also be changed locally, by replacing one of the words of the chosen sentence. Since sentences are aligned translations, access can be performed via any of the languages for which there are sentences in the base, hence also via the mother tongue, provided that it is part of the existing languages.

Of course there are limitations. We can express only ideas that fit into the patterns abstracted away via the sentences of our domain. Put differently, we can instantiate only patterns that are part of the database, or have become part of it. Still, this can be quite a bit more than the initial set of sentences, and, given the scope or our goal, this is not too much of a problem. We want to reach the survival level and not cover everything that can be conceived and expressed in a given language.

### 3.3 The Drill Tutor

The goal of the drill tutor is to help people to become fluent in a foreign language, i.e., produce at a 'normal' rate, that is, without too many hesitations, a sentence expressing some message. To this end we have chosen a set of basic patterns which we indexed in terms of goals.

Figure 3 illustrates the way how the student gets from the starting point, a *goal* (frame A), to its *linguistic realization*, the endpoint (frame D) by using the Drill Tutor. Having reached frame C, the system presents sequentially a model,[11] the stimulus (chosen word), followed by the student's answer and the system's confirmation/information

[10] For example, a pattern like 'the major export good of *country-x* is *object-y*, might yield sentences containing 'perfume', 'wine' and 'cheese' as lexical instances in the case of France, and 'cigars' and 'rhum' in the case of Cuba.

[11] The latter is basically composed of a sentence (step 1), a stimulus (the lexical value of the variable: B1, B2, etc.), and the new sentence based on the model and the stimulus (frame D).

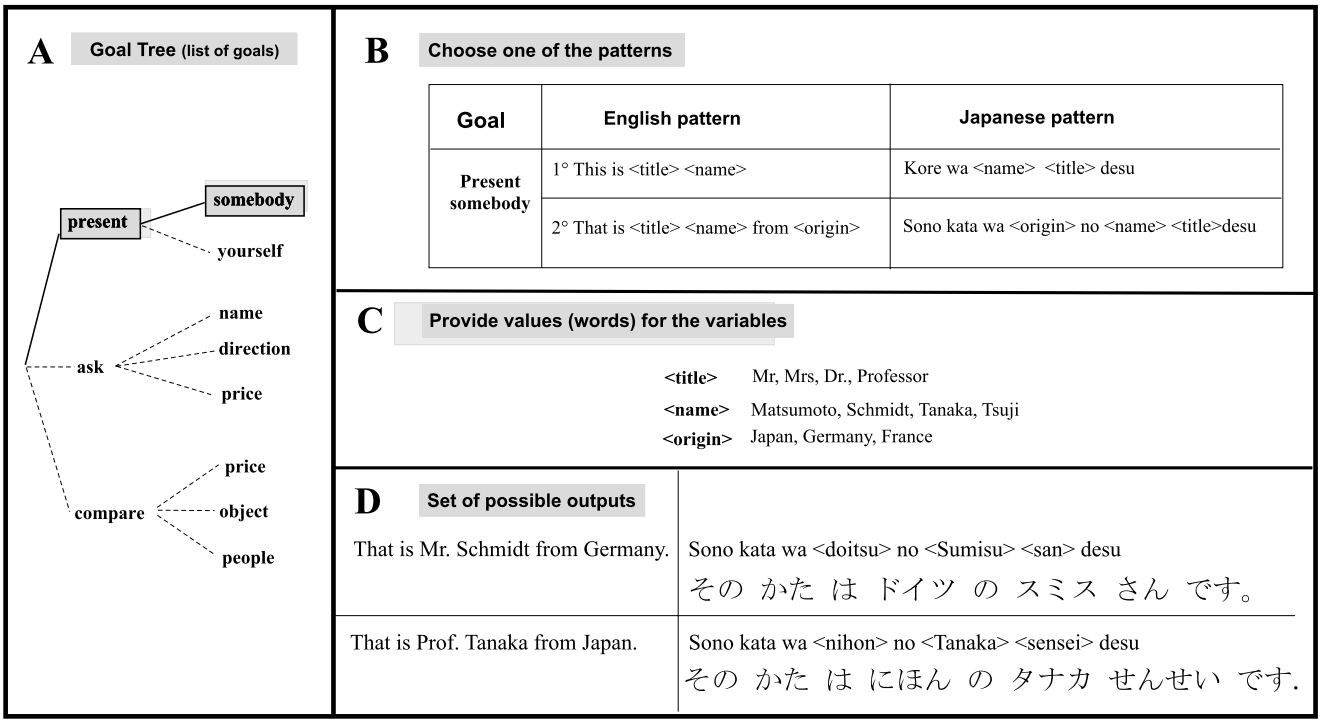


**Fig. 3** The basic interaction process between Drill Tutor and the student. The process is initiated via the choice of a goal (*introduce somebody*, step 1, frame **A**) to which the system answers with a set of patterns (frame **B**). The user chooses one of them (step 2: B1 vs. B2), signalling then the specific lexical values with which s/he would like the pattern to be instantiated (frame **C**). The system has now all the information needed to create the exercise or the sentence expressing the conceptual input (frame **D**)

(normally also a sentence, implying that the student's answer is correct if the two sentences match and incorrect otherwise). The process continues until the student has done all the exercises, or until s/he decides to stop. Note that, if the values of the variables <title>, <name>, <origin> were (Professor, Tsuji, Japan) rather than (Mr, Smith, Germany), then the outputs would vary, of course, accordingly.

It should be noted that *conceptual input* takes place in three-steps: at a global level the learner chooses the pattern via a *goal* (step 1), next s/he provides *lexical values* for the pattern's variables (step 2), to refine this global message by specifying *morphological values*: number, tense, etc. (step 3). This approach is much more economical for storing and accessing patterns, than storing a pattern for every morphological variant. This approach also allows faster authoring, i.e., message building, than long-winded navigation through a *conceptual ontology*. For more details see Zock and Afantenos (2008); Zock and Afantenos (2009).

## 4 Checking Conceptual Well-Formedness

Obviously, messages must be complete and well-formed, and this is something which needs to be checked. The problem of well-formedness is important, not only in systems where a message is built from scratch or from incomplete sentence fragments (ILLICO), but also in message-specification systems.[12] 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. Arguments are, however, 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.

To illustrate this further, let's take up again the sentence illustrated in Fig. 2, *Zidane hammered the ball straight into the net*. 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

[12]Of course, conceptual well-formedness, i.e. meaningfulness, does not guarantee communicative adequacy. In other words, it does not assure that the message makes sense in the context of a conversion. To achieve this goal additional mechanisms are needed.

(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 idea according to which types allow well-formedness checking of mathematical objects is well-known. We use it them for a different domain (messages and sentences), because they allow the checking of the adequacy of the elements used to build or complete a message. Having a rigorous representation, we can reason about objects not only to check the well-formedness of the user's input, but also its soundness.

To test this hypothesis we rely on the Coq proof-assistant (Coq Development Team 2008) as it allows us to:

- take advantage of its type system and its powerful representation mechanisms: polymorphism, dependent types, higher-order logic...;
- propose natural and general specification;
- check automatically the well-formedness of the user's input.

The Coq system provides a formal language to specify mathematical definitions and prove them. The Coq language implements a higher-order typed λ-calculus, the calculus of constructions. Its logic is constructive and relies on the *Curry-Howard isomorphism*. Each Coq proposition is of type $Prop$ and describes a predicate. There are also objects of type $Set$, but they are not used in the context of this work.

Coq allows an hierarchical organization of types via the coercion mechanism. In other words, it contains a mechanism to represent conceptual information in the form of a tree of concept types. We use coercions here to inject terms implicitly from one type into another, which can be viewed as a subtyping mechanism. Given this facility a user may apply an object (which is not a function, but can be coerced to a function) to the coercion, and more generally, consider that a term of type $A$ is of type $B$, provided that there is a declared coercion between the two.

For example, in Fig. 2 we see that a *Scene* contains both an *Attitude_Perspective* (speech-act, if you prefer) and an *Idea* (core part of the message). This is expressed in Coq as follows:

```
Coercion Attitude_Perspective_is_Scene :
         Attitude_Perspective >-> Scene.
Coercion Idea_is_Scene : Idea >-> Scene.
```

where `Attitude_Perspective`, `Idea` and `Scene` are declared as parameters of type *Prop*. These coercions declare the construction of the conceptual type `Scene` that can be seen as the composition of an `Idea` and an `Attitude_Perspective`.

The coercions used for this study are described by an inheritance graph in Coq. Moreover, Coq detects ambiguous paths during the creation of the tree, and it checks the *uniform inheritance condition* according to which at most one path must be declared between two nodes. The relevant part of the inheritance graph for our example is:

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

These four parameters describe the variables used in our example of Fig. 2. `Prop` stands in Coq for the type of proposition. Roles (`Agent`, `Object`, `Target`) and features (`human`, `soccer_instrument`, `soccer_equipment`) are generic types. To express conceptual constraints such as, *Agents* must be *animate*, Coq uses the subtype principle in order to check that all constraints are satisfied, defining `human`, `soccer_instrument` and `soccer_equipment` respectively as subtypes of `Agent`, `Object` and `Target`.

When all constraints are satisfied, the semantics of a sentence can be represented, which yields in our case "there is an agent who did something in a specific way, by using some instrument". In other words: there is a person $p$, an object $o$ and a target $t$ that are linked via an action performed in a specific way. The user message can be defined generically and typed as follows:

```
Parameter is_someone    : Agent -> Prop.
Parameter is_something : Object -> Prop.
Parameter is_manner     : Target -> Prop.
Parameter relation      : Agent ->
                          Object ->
                          Target -> Prop.
Definition message      := exists p,
                           exists o,
                           exists t,
 is_someone p/\is_something o/\
 is_manner t/\relation p o t.
```

This definition is a λ-expression taking as parameters the following variables (type names are referred to via their initial capital)

$$(\lambda s : A \to P)(\lambda o : O \to P)(\lambda t : T \to P)$$
$$\times(\lambda r : A \to O \to T \to P)$$

Hence, 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 -> Prop`), `is_ball` (of type `soccer_instrument -> Prop`), `is_net` (of type `soccer_equipment -> 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.

More generally speaking, checking the conceptual well-formedness and consistency of the messages amounts basically to type-checking the composite elements of the message.

## 5 Conclusion and Perspectives

The goal of this paper has been to deal with a problem hardly ever addressed in the literature on natural language generation, conceptual input. In order to express something, one needs to have something to express (idea, thought, concept) to begin with (input, meaning). The question is how to represent this something. What are the building blocks and how shall we organize and index them to allow for quick and intuitive access later on?

Dealing with interactive sentence generation, we suggested building a *linguistically motivated ontology* combined with a dictionary and a graph generator. While the goal of the ontology is to guide message composition (what to say), the graph's function is to show at an intermediate level the result of the encoding process. This reduces memory load, allowing at the same time the checking of well formedness. Does the message-graph really encode the authors intention?

Of course, there are many ontologies. Unfortunately, we cannot draw on any of them directly, as they have not been built for message composition.

As we have seen, different applications may require different strategies for providing input. In Illico it was driven via an ontology, taking place fairly late. Part of the message was known and expressed, thus, constraining further inputs. In the case of SPB, conceptual input consisted mainly in searching (for existing sentences or patterns) and performing local changes. Rather than starting from scratch, data are accommodated. Given the fact that we have a translation memory, input can be given in any language (mother tongue) we are confortable with, provided that it is part of the translation memory. If there is a translation between two sentences, any element is likely to evoke its equivalent and the sentence in which it occurs in the target language. Obviously, this is a nice feature, as it allows not only for natural input, but also to speed up the authoring process. In the case of DT, conceptual input is distributed over time, specification taking place in three steps: first via the choice of a goal, yielding an abstract, global structure or sentence pattern (steps 1), then via the variables' concrete lexical- and morphological values (steps 2 and 3). In the case of DT, input is clearly underspecified at the earliest stage. Messages are gradually refined: starting from a fairly general idea, i.e., sentence pattern, one proceeds gradually to the specifics: lexical and morphological values. This seems a nice feature with respect to managing memory constraints.

Many ideas presented here are somehow half-baked, needing maturation, but as mentioned earlier on, conceptual input is an area in Natural Language Generation where more work is badly needed.

**Acknowledgements** We would like to thank Guy Lapalme for the time spent on reading, commenting and correcting our work. Sincere thanks also to Nicolas Nicolov and Lih-Juang for proofreading the final version. Their work helped a lot improving the final document. Finally, we would like to express our gratitude to Stergos Afantenos for implementing DT, and to Sadaoki Furui and Josafa de Jesus Aguiar Pontes from TITECH (Tokyo) for their contributions within the Speaking-Phrasebook project.

## References

Baker, C. F., Fillmore, C. J., & Lowe, J. B. (1998). The Berkeley FrameNet project. In *COLING/ACL-98*. Montreal (pp. 86–90).

Bateman, J., & Zock, M. (2003). Natural language generation. In R. Mitkov (Ed.), *Oxford handbook of computational linguistics* (pp. 284–304). London: Oxford University Press, Chap. 15.

Boitet, C., Bhattacharyya, P., Blanc, E., Meena, S., Boudhh, S., Fafiotte, G., Falaise, A., & Vacchani, V. (2007). Building Hindi-French-English-UNL resources for SurviTra-CIFLI, a linguistic survival system under construction. In *Seventh international symposium on natural language processing*. Pattayah (p. 7). Kasetsart University.

Briffault, X., & Zock, M. (1994). 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 (pp. 363–371).

Coq Development Team (2006–2008). *The Coq proof assistant. Reference manual*. INRIA.

Fellbaum, C. (1998). *WordNet: an electronic lexical database and some of its applications*. Cambridge: MIT Press.

Fromkin, V. (1993). Speech production. In J. Berko-Gleason & N. B. Ratner (Eds.), *Psycholinguistics*. Fort Worth: Harcourt, Brace, Jovanovich.

Hearst, M. (1998). Automated discovery of Word-Net relations. In C. Fellbaum (Ed.), *WordNet an electronic lexical database* (pp. 131–152). Cambridge: MIT Press.

Levelt, W. (1989). *Speaking: from intention to articulation*. Cambridge: MIT Press.

Ligozat, G., & Zock, M. (1992). How to visualize time, tense and aspect. In *Proceedings of COLING '92*. Nantes (pp. 475–482).

McCoy, K., & Cheng, J. (1991). Focus of attention: constraining what can be said next. In C. Paris, W. Swartout, & W. Mann (Eds.), *Natural language generation in artificial intelligence and computational linguistics* (pp. 103–124). Boston: Kluwer Academic.

Meteer, M. W. (1992). *Expressibility and the problem of efficient text planning*. London: Pinter.

Miller, G. A. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, *63*, 81–97.

Nicolov, N., & Mellish, C. (2000). PROTECTOR: efficient generation with lexicalized grammars. In *Recent advances in natural language processing* (pp. 221–243). Amsterdam: Benjamins. Current issues in linguistic theory (CILT 189).

Pasero, R., & Sabatier, P. (1995). Guided sentences composition: some problems, solutions and applications. In *Proceedings of the 5th international workshop on natural language understanding and logic programming (NLULP)*. Lisbon (pp. 97–110). Lisbon University.

Pasero, R., & Sabatier, P. (2007). ILLICO: présentation & principes, connaissances et formalismes & guide d'utilisation. Technical report, Laboratoire d'Informatique Fondamentale (LIF), Marseille, http://www.lif.univ-mrs.fr/illico.html.

Power, R., Scott, D., & Evans, R. (1998). What you see is what you meant: direct knowledge editings with natural language feedback. In H. Prade (Ed.), *13th European conference on artificial intelligence (ECAI'98)* (pp. 677–681). Chichester: Wiley.

Reiter, E., & Dale, R. (2000). *Building natural language generation systems*. Cambridge: Cambridge University Press.

Sabatier, P. (1997). Un lexique-grammaire du football. *Lingvisticæ Investigationes*, *21*(1), 163–197.

Schank, R. (1975). Conceptual dependency theory. In R. C. Schank (Ed.), *Conceptual information processing* (pp. 22–82). Amsterdam/New York: North-Holland/Elsevier.

Schmidt, T. (2007). Multilingual FrameNets. In *The Kicktionary—a multilingual lexical resource of football language*. New York: de Gruyter.

Tesnière, L. (1959). *Éléments de syntaxe structurale*. Paris: Klincksieck.

Zock, M. (1991). Swim or sink: the problem of communicating thought. In M. Swartz & M. Yazdani (Eds.), *Intelligent tutoring systems for foreign language learning* (pp. 235–247). New York: Springer.

Zock, M. (1996). The power of words in message planning. In *International conference on computational linguistics*. Copenhagen (pp. 990–995).

Zock, M., & Afantenos, S. (2008). Verbal fluency, or how to stay on top of the wave. In B. Sharp & M. Zock (Eds.), *NLPCS*. Barcelona (pp. 159–164). INSTICC Press.

Zock, M., & Afantenos, S. (2009). Using e-Learning to achieve fluency in foreign languages. In A. Tzanavari & N. Tsapatsoulis (Eds.), *Affective, interactive and cognitive methods for e-learning design: creating an optimal education experience*. Hershey: IGI Global.

Zock, M., & Schwab, D. (2008). Lexical access based on underspecified input. In *Proceedings of the workshop on cognitive aspects of the lexicon (COGALEX 2008)*. Manchester, United Kingdom (pp. 9–17). Coling 2008.