Ontology in Coq for a Guided Message Composition

Line Jakubiec

Ontology in Coq for a Guided Message Composition

Line Jakubiec-Jamet

Laboratoire d'Informatique Fondamentale de Marseille
Aix-Marseille Université, CNRS, LIF UMR 7279
163, Avenue de Luminy - Case 901
13288 Marseille Cedex 9 - France

Summary. Natural language generation is based on messages that represent meanings, and goals that are the usual starting points for communicate. How to help people to provide this conceptual input or, in other words, how to communicate thoughts to the computer? In order to express something, one needs to have something to express as an idea, a thought or a concept. The question is how to represent this. In 2009, Michael Zock, Paul Sabatier and Line Jakubiec-Jamet suggested the building of a resource composed of a *linguistically motivated ontology*, a *dictionary* and a *graph generator*. The ontology guides the user to choose among a set of concepts (or words) to build the message from; the dictionary provides knowledge of how to link the chosen elements to yield a message (compositional rules); the graph generator displays the output in visual form (message graph representing the user's input). While the goal of the ontology is to generate (or analyse) sentences and to guide message composition (what to say), the graph's function is to show at an intermediate level the result of the encoding process. The Illico system already proposes a way to help a user for generating (or analyzing) sentences and guiding their composition. Another system, the Drill Tutor, is an exercise generator whose goal is to help people to become fluent in a foreign language. It helps people (users have to make choices from the interface in order to build their messages) to produce a sentence expressing a message from an idea (or a concept) to its linguistic realization (or a correct sentence given in a foreign language). These two systems led us to consider the representation of the conceptual information into a symbolic language; this representation is encoded in a logic system in order to automatically check conceptual well-formedness of messages. This logic system is the Coq system used here only for its high level language. Coq is based on a typed λ-calculus. It is used for analysing conceptual input interpreted as types and also for specifying general definitions representing messages. These definitions are typed and they will be instanciated for type-checking the conceptual well-formedness of messages.

1 Introduction

Natural language generation is typically based on messages, i.e. meanings, and goals, the usual starting points. We present our views on how to help people to provide this kind of input. Guiding a user to compose sentences can be done in many ways, a lot depending on the user's knowledge state.[1] We will present here *Illico* (Pasero and Sabatier)[2], some of its shortcomings and our solution. Illico can be helpful in many ways. For example, it can analyze and synthesize expressions (words, phrases, clauses, 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). All the suggestions made by the system are possible expressions fitting into this place and being in line with the constraints at hand. To achieve this goal a powerful mechanism is used to process in parallel knowledge of different sorts (lexical, syntactic, semantic, conceptual and even contextual ones).

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 is powerful enough to allow 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.

---

[1] The problem we are dealing with here is search. Obviously, knowledge available at the onset (cognitive state) plays also 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 (knowledge available) and the distance between the latter and a given output (target word). Imagine that your target word were 'guy' while you've started search from any of the following inputs: 'cat' (synonyme), 'person' (more general term), or 'gars' (equivalent word in French). Obviously, the type of search and ease of access would not be the same. The nature and number of items among which to choose would be different in each case. The influence of formally similar, i.e. close words ('libreria' in Spanish vs. 'library' in English) is well known. Cognates tend to prime each other, a fact that depending on the circumstances can be helpful or sheer nuisance.

[2] 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.) you may want to take a look at `http://pageperso.lif.univ-mrs.fr/~paul.sabatier/ILLICO/illico.html`.

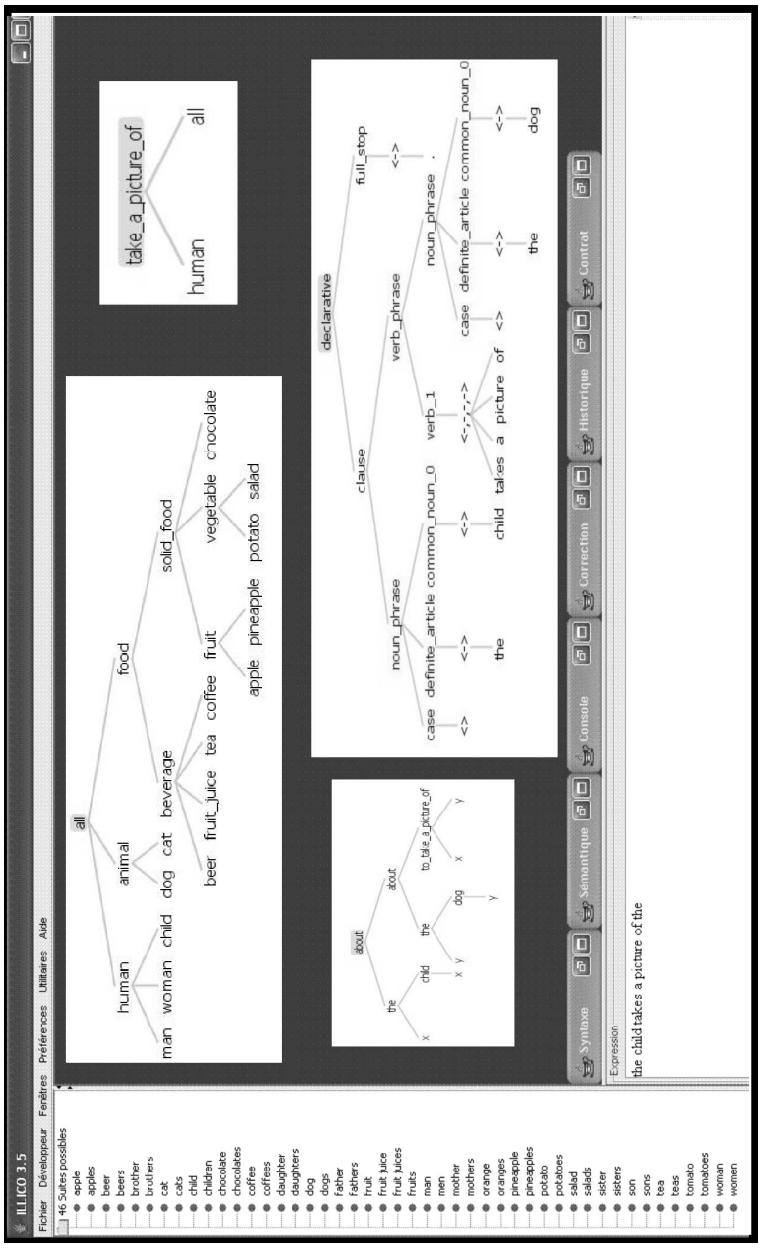

Fig. 1. Illico Screen Printing

If you want 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 ("The child takes a picture of the...") the user is waiting for suggestions to be made by the system. Offering possible continuations is but one way among others to assist a user in sentence composition. One can imagine richer kind of assistance where the user accesses various kinds of knowledge (linguistic, conceptual, etc.) to select then the one fitting best his communicative goals.

This aspect could concretely help a user in sentence composition to be fluent in a foreign language. Becoming fluent in a language requires not only learning words and methods for accessing them quickly, but also learning how to place them and how to make the necessary morphological adjustments. The Drill Tutor [ZL10] aims to help people to produce a sentence expressing some message. In the system, the user chooses the goal of communication (to introduce someone for example). The goals is tree-structured. The user can either drill down to a given goal by clicking on a goal name to expand the sub-goals until a pattern is displayed, or search for them via a term because goals are indexed. The Drill Tutor gets from the starting point, the goal, to its linguistic realization. The system presents sequentially:

1. a model: the user chooses one of the patterns proposed by the system (a pattern is a general sentence needing to be instanciated; for example, to introduce someone, one can use the model: *This is <title><name>* or *That is <title><name> from <origin>*);
2. the stimulus (chosen word): this step allows the instanciation of the chosen pattern. For example, for <title>, the user can use *Mr, Mrs, Dr., Professor*, and for <origin>, he can choose *Japan, Germany, France*;
3. the user's answer and the system's confirmation: the system has now all the information needed to create the sentence (or the exercise) expressing the conceptual input. Finally, a set of possible outputs representing the message is depicted on the screen and the user can express his message.

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 [ZA09]).

2 Limitation of sentence completion or the need of controlling conceptual input

The objectives of *sentence completion systems* are different from those of conventional surface generators [RD00, BZ03]. The latter 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[3].

Illico performs analysis by synthesis (top-down strategy) and, while it does not need any help from the user for analysis, it does so for synthesis, as, otherwise, it would produce unreasonably large sets of possible continuations, most of which do not even correspond to the users' intention. 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.

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

A system providing more sofisticated assistance system would look quite a bit differently: (a) the nodes of the tree would be *categories* (types) rather than *instances* (words), the latter being shown only at the leave-level; (b) frequency would be taken into account (the words' likelihood varying with the topic); (c) *governors* (eg. nouns) would precede their *dependants* (eg. determiners, adjectives), and (d) *variables* would be used rather than extensive lists of possible morphological *values*, etc.

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) 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, does not correspond anymore to what one had in mind. Memory and divided attention have taken their toll. In order to avoid this, we suggest to determine the governing elements first and to keep the set of data from which to choose 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 transmission of information, i.e. reasonable input/output are considered as a

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

[4] For example, it allows the testing of well-formedness and linguistic coverage of the application one is about to develop. This being so, we can check now whether all the produced continuations are expected and none is missing.

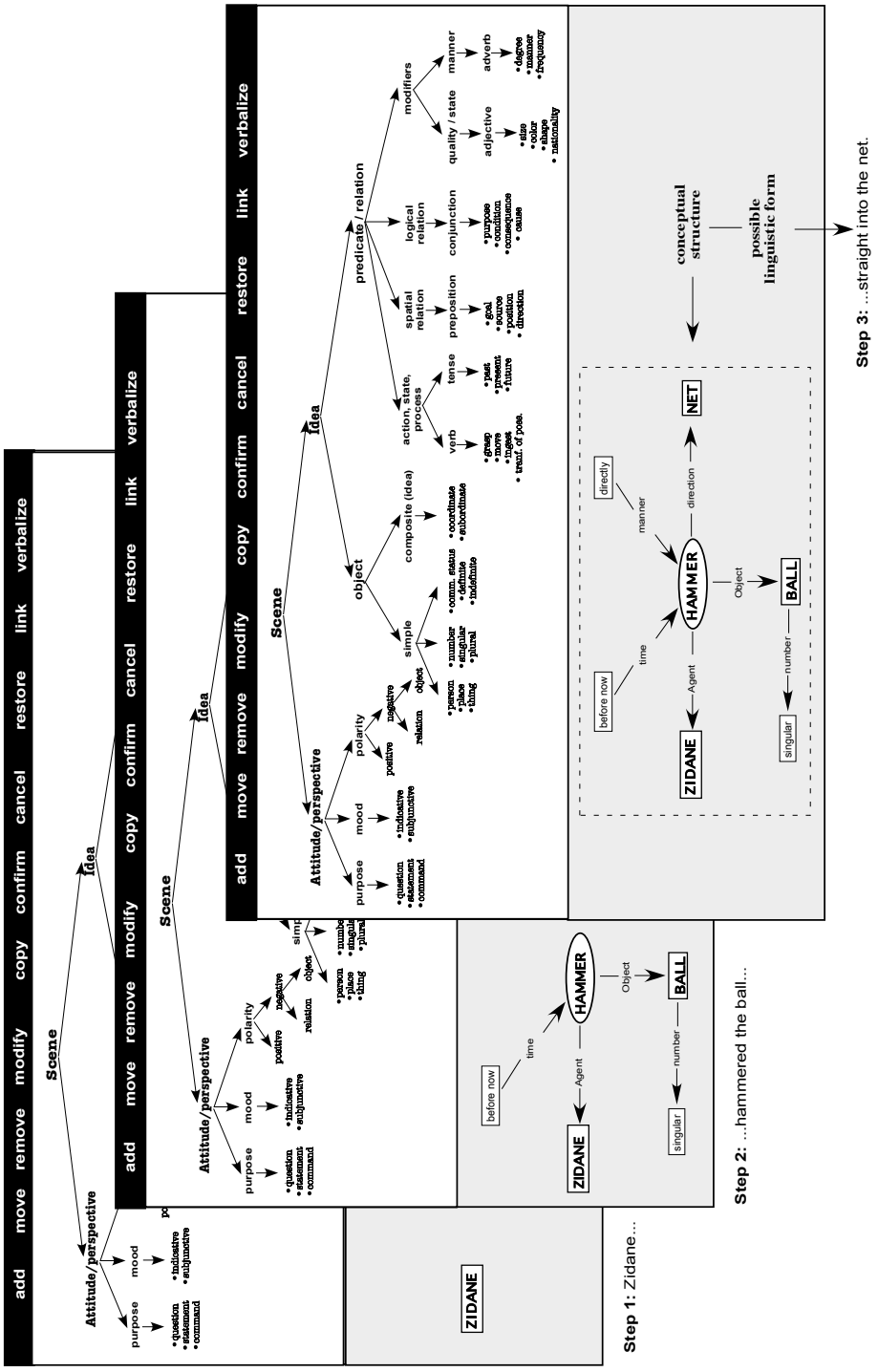

Fig. 2. An Interactive Graph Editor for Incremental Message Composition

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[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 structured list. This 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, hyperonymes) 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 from a much smaller list. The gain is obvious in terms of storage (at the interface 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. It is reminiscient of SWIM, an ontology-driven interactive sentence generator [Zoc91]. 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, ball, straight, net)[6]. 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

[5] This idea is somehow contained in Tesnière's notion of *valency* [Tes59], in Schank's *conceptual dependancy* [Sch75] and McCoy and Cheng's *discourse focus trees* [MC91].

[6] The upper part shows the conceptual 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 have ignored the attitude or speech-act node in the lower part of our figure.

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. In spite of 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, incrementally displays the message structure in the making.

How to use the resource? Obviously, as the ontology or conceptual tree grows, access time increases, 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 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; (3) allow access via the words' initial letters, or, even better, (4) via associatively linked concepts/words.[7].

[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, 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

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, this is not necessarily a shortcoming, quite 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 occurring in the message graph might not be syntactically compatible with each other. Hence, we will be blocked, facing a problem of expressibility [Met92].

4 Conceptual, computational and psychological 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 [Sab97]. 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 [BFL98] and uses WordNet style semantic relations [Fel98] 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 *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 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.

s_w, but also evokes *Kawa Igen*, a javanese volcano, or familiar word of *coffee* in French. For more details, see [ZS08].

- **the limits of symbolic representation**: as shown elsewhere for *time* [LZ92] and *space* [BZ94], 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 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?

There are several other issues that we have hardly touched upon, yet they are all relevant for natural language generation, in particular for interactive language generation which is our case:

1. in what terms to encode the message (concepts vs. words),
2. at what level of abstraction (general vs. specific);
3. size of the planning unit (concepts vs. messages);
4. processing strategy: is planning done as a one-shot process or is it performed in various steps, i.e. incrementally?;
5. direction: is planning done left to right or top to bottom?
6. 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 [Zoc96]). Let's 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.

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 of planning, that is, the need to look ahead and to plan in general terms, increases with sentence length and with the number and type of embeddings (for example, center embedded sentences). There is also good evidence in the speech error literature for 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 [Lev89, Fro93].

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.

5 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 [8]. 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, lets 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 (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 users'input.

The idea according to which types allow well-formedness checking of mathematical objects is well-known. We use them 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 users'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...;

[8] 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.

- propose natural and general specification;
- check automatically the well-formedness of the users'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):

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

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 $human \to Prop$), *is_ball* (of type $soccer\_instrument \to Prop$), *is_net* (of type $soccer\_equipment \to Prop$). $Hammer$ 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.

Another approach implemented in Coq allows the formalization of general patterns used for representing sentences. Then, these patterns are instanciated and a semantic analysis of simple sentences can been performed. This analysis relies on a hierarchy of types for type-checking the conceptual well-formedness of sentences in the same spirit of this paper[9]. The motivation for using Coq is to define general representations in order to have a more economical way for storing and analysing sentences that are built according patterns (for more details see [JJ12]).

[9] Actually I gratefully acknowledge Michael from many fruitful discussions about this approach. He always has been very attentive to others'works and our collaboration is due to him.

6 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 have suggested the building of a *linguistically motivated ontology* combined with a dictionary and graph generator. While the goal of the ontology is to generate (or analyse) sentences and 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 author's 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.

Michael Zock also worked on another message-specification system (the SPB system), a multi-lingual phrasebook designed to convert meanings into speech. In 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[10].

In the case of Drill Tutor, 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 Drill Tutor, 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, conceptual input is an area in Natural Language Generation where more work is badly needed.

[10] For a similar goal, but with a quite different method, see [CPE+07]

References

[BFL98] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The berkeley framenet project. In *COLING/ACL-98*, pages 86–90, Montreal, 1998.

[BZ94] Xavier Briffault and Michael Zock. 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*, pages 363–371, Sofia, 1994.

[BZ03] John Bateman and Michael Zock. Natural language generation. In R. Mitkov, editor, *Oxford Handbook of Computational Linguistics*, chapter 15, pages 284–304. Oxford University Press, 2003.

[CPE+07] C.Boitet, P.Bhattacharyya, E.Blanc, S. Meena, S.Boudhh, G.Fafiotte, A.Falaise, and V.Vacchani. Building Hindi-French-English-UNL resources for SurviTra-CIFLI, a linguistic survival system under construction. In *Seventh international symposium on natural language processing*, 2007.

[Fel98] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database and some of its Applications*. MIT Press, 1998.

[Fro93] Victoria Fromkin. Speech production. In Jean Berko-Gleason and N. Bernstein Ratner, editors, *Psycholinguistics*. 1993.

[JJ12] Line Jakubiec-Jamet. A Case-study for the Semantic Analysis of Sentences in Coq. Research report, LIF, 2012.

[Lev89] William Levelt. *Speaking : From Intention to Articulation*. MIT Press, Cambridge, MA, 1989.

[LZ92] Grard Ligozat and Michael Zock. How to visualize time, tense and aspect. In *Proceedings of COLING '92*, pages 475–482, Nantes, 1992.

[MC91] Kathleen McCoy and J. Cheng. Focus of attention: Constraining what can be said next. In Ccile Paris, William Swartout, and William Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pages 103–124. Kluwer Academic Publisher, Boston, 1991.

[Met92] Marie W. Meteer. *Expressibility and the Problem of Efficient Text Planning*. Pinter, London, 1992.

[RD00] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, 2000.

[Sab97] Paul Sabatier. Un lexique-grammaire du football. *Lingvistic Investigationes*, XXI(1):163–197, 1997.

[Sch75] Roger Schank. Conceptual dependency theory. In R. C. Schank, editor, *Conceptual Information Processing*, pages 22–82. North-Holland and Elsevier, Amsterdam and New York, 1975.

[Tes59] Lucien Tesnière. *Éléments de syntaxe structurale*. Klincksieck, Paris, 1959.

[ZA09] Michael Zock and Stergos Afantenos. Using e-learning to achieve fluency in foreign languages. In N. Tsapatsoulis (Eds.) A. Tzanavari, editor, *Affective, interactive and cognitive methods for e-learning design: creating an optimal education experience*. Hershey: IGI Global, 2009.

[ZL10] Michael Zock and Guy Lapalme. A Generic Tool for Creating and Using Multilingual Phrasebooks. In *Natural Language Processing and Cognitive Science, Funchal*, 2010.

[Zoc91] Michael Zock. Swim or sink: the problem of communicating thought. In M. Swartz and M. Yazdani, editors, *Intelligent Tutoring Systems for Foreign Language Learning*, pages 235–247. Springer, New York, 1991.

[Zoc96] Michael Zock. The power of words in message planning. In *International Conference on Computational Linguistics, Copenhagen*, 1996.

[ZS08] Michael Zock and Didier Schwab. Lexical access based on underspecified input. In *Proceedings of the Workshop on Cognitive Aspects of the Lexicon (COGALEX 2008)*, pages 9–17, Manchester, United Kingdom, August 2008. Coling 2008.