Perception, Concepts and Language: $\mathcal{RoAD}$ and IPaGe

Matthias Rehm and Karl Ulrich Goecke
Faculty of Linguistics and Literature
University of Bielefeld
{rehm, goecke}@coli.uni-bielefeld.de

Abstract

A two-level natural language generation system for *situation and action descriptions (SADs)* of a simulated assembly robot is presented. In the first step, multimodal information is used to obtain a *conceptual representation (CR)* of an event. The second step is the parallel, incremental surface realization of utterances from the robot's perspective based on the CR. Theoretical issues addressed are semantics of SADs and distribution of lexical and syntactic processing, leading to a natural type of incrementality in NLG.

1 Introduction

Following (Reiter, 1994), in the majority of systems language generation starts with *content determination*. This step is further subdivided in *deep content determination* where a decision takes place what information should be communicated to the hearer and *rhetorical planning* where this information is organized in a rhetorically coherent manner. However, the definition of deep content determination allows two differing interpretations: One perspective is that there already exists a set of representations of different contents in a specific representation format. The task is to select one of these representations to be communicated. This process is typically implemented using discourse structure information and other criteria (e.g. Grice's maximes, (Grice, 1975)).

The other perspective is more extensive. If, initially, no set of representations is available, deep content determination could also comprise the creation of this set. In many technical applications, this viewpoint does not really make sense. In the most widespread case - language generation out of a data base - the system *is* the set of representations. A possibly more complex case - language generation by agents[1] - seems to be no problem either: Take the world model and the plan representing the agent's environment and future actions, respectively and then select from these representations the content to be verbalized. Approaches of this kind are indeed applied in various scene description or robotic systems, e.g. (Novak, 1987), (Längle et al., 1995).

However, there are agent architectures not entertaining a coherent world model and not having an explicit plan at every level of action available. Such agent architectures have emerged within *behavior-oriented robotics* over the past fifteen years (e.g. (Brooks, 1986)). Perception and control processes are distributed over many modules and interact on a local rather than a global basis. Thus, reactive behavior is modeled.

Considering the problem of language generation , it becomes more natural to adopt the second perspective about what deep content determination should mean in this context. In this paper, we present an approach to natural language generation where in a first step, a set of possible utterances of an agent is constructed by the system $\mathcal{RoAD}$. For the lack of a better term, we call this process *conceptualization*. The intermediate structure is based on the Conceptual Semantics paradigm (e.g. (Jackendoff, 1990)). In a second step, what is classically thought of as language generation is accomplished with the system IPaGe (**I**ncremental **Pa**rallel **Ge**nerator). Here we propose a massively parallel approach that leads to a natural kind of either qualitative and quantitative incrementality (c.f. (Finkler, 1997)).

The paper is structured as follows: In section 2, some theoretical claims are clarified and the implementation of $\mathcal{RoAD}$ is illustrated with an example conceptualization. The corresponding CR is also used in section 3 where IPaGe is described. An outlook on directions for further research in section 4 concludes this paper.

2 Conceptualization

The example domain of our work is the artificial agent $\mathcal{CoRA}$ (c.f. (Milde et al., 1997)). It is a simulated assembly robot able to manipulate wooden toy parts and instructable by a human user. Due to its behavior-based control architecture, not every instructed action is carried out as expected. Thus, an explanation of ongoing actions by the robot is desir-

[1] For a discussion of this notion see e.g. (Franklin and Graesser, 1996).

able. We shall deal with a specific kind of utterances, namely *situation/action descriptions (SADs)*.

SADs are descriptions of the environment and the actions of an agent from its own perspective. In the chosen domain, possible SADs are: `Ich bewege mich zu dem blauen Würfel` (I am moving towards the blue cube) or `Ich lege den Würfel auf eine Leiste` (I am placing a cube on a connection bar).

A theoretical and an application-oriented issue are addressed with the investigation of SADs: First, we claim that internal and sensoric parameters of the agent play a decisive role in determining the semantics of SADs, especially of action verbs. Second, by generating SADs, we enable an interactor to understand what the agent is doing. This is particularly interesting if there is no direct visual contact between the agent and the interlocutor.

Sensor data (visual, telemetric, haptic) and internal states (activation of behavior generating modules (BM), joint values) of $\mathcal{CORA}$ serve as *selection criteria (SC)* of *interpretative schemata (ISM)* containing also conceptual units (Goecke and Milde, 1998). These units constitute the interface to the surface generation process. If the SC are present in the current sensoric pattern of the robot, an ISM may become active and the corresponding concept information is passed to surface generation (c.f. section 3). Some SC are temporally extended: A sensor value has to be in some interval for a certain amount of time to function as a trigger for an ISM.

Additional to sensor data and internal values, ISM themselves can be part of the SC of other ISM. The latter ones are then said to have a higher degree of complexity then the former ones. ISM may be subsumed by other ISM or may bear a part/whole-relation to other ISM. An ISM is subsumed by another one if its SC constitute a proper subset of the SC of the subsuming ISM, including time restrictions. For example, the ISM MOVE, detecting a movement of the robot that is not specified with respect to direction or velocity, is subsumed by MOVE-TO-OBJECT which, in addition, recognizes a goal of the movement, namely some object. Part/whole-relations of ISM exist if a "higher" ISM combines others to identify more complex actions. In this case, the "lower" ISM not necessarily has to be active over the whole time interval that the higher one covers.

Thus, ISM form a hierarchy (see fig. 1). ISM on the lowest level (SEE, MOVE, BUMP, ...) are *basic* in the sense that they only contain sensor data as SC. *Complex* ISM in levels 1 to 4 integrate sensor data as well as other ISM.

When an ISM becomes active, the corresponding *conceptual representation (CR)* is a possible candidate to be passed to the surface generation component. As it is possible that several ISM are active

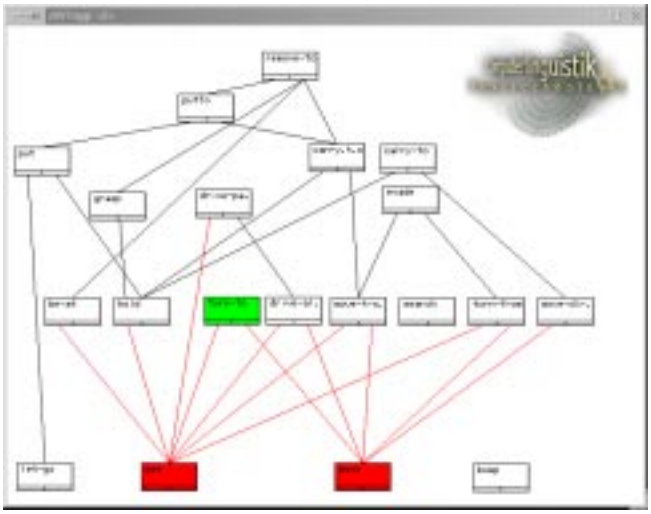

Figure 1: Screen depiction of the ISM-hierarchy. Active ISM are coloured; edges mark subsumption or part/whole-relations.

at the same time, a selection has to take place. At present, the only criterium relevant for selection is the position of the corresponding ISM within the hierarchy. Thus, only the CR of the highest active ISM is going to be verbalized at the given time.

CRs contain information about objects, their properties and about events the agent is involved in. They follow the Conceptual Semantics by Jackendoff (Jackendoff, 1990). In the next section, an example shows the representations used in $\mathcal{RoAD}$ in more detail.

2.1 Example

In the following, the conceptualization of the SAD `Ich drehe mich zu dem blauen Würfel` (I am turning to the blue cube) is going to illustrate the processing mechanisms described in the previous section.

Suppose a situation where visual and movement information is provided by the sensors of the robot. Among other things, the ISM SEE checks for the existance of values in the interface for either object type (`type`), object color (`color`) or object position within the visual field (`x-g und y-g`). A possible configuration is

```
type:  c      color:  b
x-g:  102      y-g:  99
width:  83     height:  200
```

The SC of SEE is a disjunction of several conditions. If any of these conditions is met, the ISM becomes active: `x-g` $>$ `0` | `y-g` $>$ `0` | `width`

> 0 | height > 0. The comparison between the attribute-value pairs in the interface and the SC of SEE shows that some relevant parameters are indeed present. Thus, SEE becomes active.

The underspecified CR of SEE is EVENT: see, AGENT: i, OBJECT: OBJ, (COLOR: COL). EVENT and AGENT are instantiated with default values. The associated transition rules specify the remaining conceptual parameters:

$$\text{type OBJ} \longrightarrow \text{OBJ}$$
$$\text{color COL} \longrightarrow \text{COL}$$

Consequently, the complete CR of SEE is EVENT: see, AGENT: i, OBJECT: c, COLOR: b[2]

On the basis of the sensor data down: -8 and velocity: 0.441355, denoting a downward movement with a certain velocity, the ISM MOVE becomes active at the same time.

The basic ISM SEE and MOVE serve as SC for the complex ISM TURN-TO which identifies the turning of the robot towards an object. Furthermore, TURN-TO has some additional SC. The complete set is as follows:

$$\text{MOVE \& SEE}$$
$$\text{\& ((cont ($\rightarrow$ 100) x-g) \& (cont ($\rightarrow$ 100) y-g)) for 5 cycles}$$

Thus, if MOVE and SEE are active for five cycles and, in addition, the object in the visual field is moving to the center of vision, TURN-TO is activated.

MOVE and SEE make available their CRs to TURN-TO. By this means, previously unspecified parameters in TURN-TO's CR can be spelled out via the transition rule

$$\text{see(OBJ, COL)} \longrightarrow \text{OBJECT, COLOR}$$

resulting in the fully instantiated CR EVENT: turn, AGENT: i, PATH: to, OBJECT: c, COLOR: b.

If TURN-TO turns out to be the highest active ISM at a given time, it is selected for surface generation.

3 Generating the utterance

After conceptualizing the agent's current action at the level of ISMs (content determination) it has to be decided how the corresponding CR can be articulated in a natural language utterance. A fundamental division in lexical and syntactical processing along with an incremental and parallel processing behaviour are the crucial features of the proposed architecture for surface realization. Such a processing behaviour is facilitated by the use of CRs as interface between $\mathcal{RoAD}$ and IPaGe.

[2] c stands for the conceptual entity 'cube', b for 'blue'.

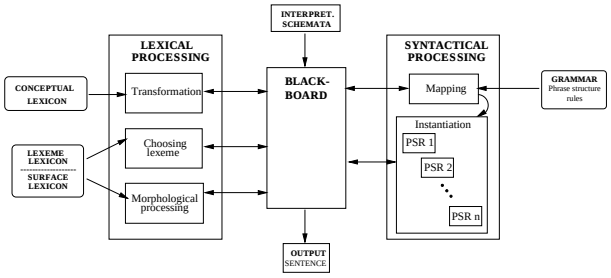

Figure 2: The architecture of the generation system. Conceptual, lexical, and syntactical information is stored centrally on the blackboard. All processes can obtain this information if needed.

3.1 Blackboard architecture

The blackboard is the central storing device of the system. All processes write their results on the blackboard and obtain their input structures there. In such an architecture parallel processing can be achieved in a convenient fashion.

3.2 Parallel processing

Parallel processing can be found on various levels of abstraction. The fundamental division runs between lexical and syntactical processing (see fig. 2). When a (part of a) CR originating from the ISM is written on the blackboard, processing of this structure starts simultaneously in both components. Using a conceptual lexicon as described by Jackendoff a transformation process constitutes the first step on the lexical side (Jackendoff, 1990). On the syntactical side, a process based on the type-phrase-correspondence, also described by Jackendoff, starts processing in this component. The type-phrase-correspondence constrains the choice of possible phrases to realize a structure with a given conceptual type in an utterance.

Different processes inside these two components work simultaneously on different CRs. On the lexical side, the three subprocesses transformation, choosing of lexeme and morphological processing can be identified. On the syntactical side the mapping of conceptual types on possible phrase structure rules and the instantiation of these rules take place.

3.3 Incremental processing

In natural language generation, incremental processing is a central issue. According to Finkler's definition two fundamental types can be identified: *quantitative* and *qualitative* incremental processing (Finkler, 1997).

Quantitative incremental processing means that the input to a process can be divided into differ-

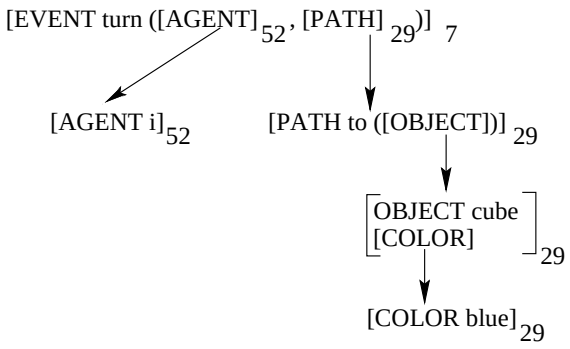

Figure 3: A complex CR with its subparts which can serve as input increments to the generation process. Parts of the same structure are coindexed. The relevant features for generation from CRs are: the type, the head, and the number and types of arguments. Thus, `[PATH to([OBJECT])]` is a possible increment.

ent parts and that processing of these parts is possible. Qualitative incremental processing on the other hand denotes the possibility to obtain a first, suboptimal result and to elaborate it when new information is available.

IPaGe realizes both kinds of incremental processing. Parts of CRs correspond to possible phrase structures that constitute the utterance, i.e. type-phrase-correspondence. Thus quantitative incremental processing can be achieved in a natural way. An arbitrarily complex part of a CR can serve as an increment.

Qualitative incremental processing is accomplished on the level of instantiating phrase structure rules. A phrase can always be realized by several rules of differing complexity. For example, a noun phrase can be realized in one of the following ways: as a noun, as a determiner and a noun, as an adjective and a noun, etc. All rules for a given CR are started as independent processes trying to instantiate their right hand sides. The result of a successful process is written on the blackboard and all processes with equal or less complexity are stopped. Processes of higher complexity can try further to instantiate their phrase structure rules. In case one of these processes succeeds, the former result is overwritten by the more complex one. Depending on the utterance time of the given phrase a more or less complex result is achieved.

3.4 Example

The example introduced in section 2.1 is continued here. Output of the ISM and thus input to the generation process is the CR depicted in figure 3. This structure will be realized in an utterance like `Ich drehe mich zu dem Würfel` (I am turning to the cube) bzw. `Zu dem blauen Quader drehe ich mich` (To the blue cuboid I am turning). The generation process is exemplified by the processing of the `PATH`-structure.

3.4.1 Transformation

Every CR carries enough information to initiate different processes simultaneously. At the moment an input structure is supplied, it triggers processing inside the lexical and the syntactical component: the transformation and the mapping process.

As CRs describe meaning by a structural mechanism, the same head can have different meanings in different structural constellations. Transformation - a disambiguation process - is implemented as a lookup process in a conceptual lexicon. The entries in this lexicon are sorted by different keys. The first key is the type of the CR. Input increments to the lexical processing component are typed CRs. Thus a type specific distribution of processing seems natural here.

By the `PATH`-structure the `PATH`-specific transformation process is triggered. The lookup process will yield a so-called intermediate structure already with some syntactic information such as category information: `[PRED to, CAT prep, ARG 29]`$_{29}$. `[OBJECT cube]` will yield: `[PRED block, CAT n]`$_{29}$.

3.4.2 Mapping

`PATH`- and `OBJECT`-structure initiate the mapping process simultaneously to the transformation process. During mapping the type-phrase-correspondence as described by (Jackendoff, 1990) is used. A given type can be expressed in an utterance only by a restricted set of possible phrases. A mapping of the types of the input structures to the relevant phrase structure rules takes place. These rules are then started as independent threads.

In German, structures with type `PATH` are nearly always realized as prepositional phrases. The `PATH`-specific mapping process will thus start PP-rules, e.g. `PP`$_{29}$ → `PREP NP` or `PP`$_{29}$ → `PREP ADV`. The `OBJECT`-structure will trigger the `OBJECT`-specific mapping thread which will start NP-rules, e.g. `NP`$_{29}$ → `N`, `NP`$_{29}$ → `DET N`, or `NP`$_{29}$ → `DET ADJ N`.

3.4.3 Instantiation

All threads started during the mapping process constitute the instantiation module. These threads correspond to phrase structure rules and try to substitute the right hand side non-terminals with inflected words. If one rule is fulfilled, the result will be written on the blackboard. All rules for the same constituent which are more complex will continue processing. If such a more complex rule is fulfilled, it will overwrite the result on the blackboard by a more

complex, i.e. more elaborate one. Which one will be uttered depends on the time the output process reaches the corresponding phrase and on the time contraint given by an *urgency* parameter. A low urgency corresponds to more processing time whereas a high urgency denotes the need for a fast result.

Let's have a closer look on the $\texttt{NP}_{29}$-rules. If `Würfel` is supplied by the lexical processing component, all `N`-constituents will be substituted by this word: (i) $\texttt{NP}_{29} \rightarrow$ `Würfel`, (ii) $\texttt{NP}_{29} \rightarrow$ `DET Würfel`, (iii) $\texttt{NP}_{29} \rightarrow$ `DET ADJ Würfel`. Rule (i) is complete and will be written on the blackboard. If the output process reaches the noun phrase at that moment of time `Würfel` (cube) will be used as noun phrase. Otherwise the more complex threads can try to fulfill their rules. Next, e.g., the determiner reaches the blackboard. Rule (ii) will be complete and will overwrite the former result: $\texttt{NP}_{29} \rightarrow$ `dem Würfel`. Now, the uttered noun phrase will be `dem Würfel` (the cube).

3.4.4 Choosing Lexemes

The inflected words are provided by the lexical processing component. After the disambiguation of the input structures during the transformation process possible lexemes[3] have to be chosen that will constitute the utterance. This process is once more a lookup in a lexicon. The first key to this lookup is the syntactic category which is given in the intermediate structures. Thus a category-specific distribution of processes is realized here.

Many concepts can be paraphrased by a number of words. For example, the concept of `cube` can be expressed in German by words like `Würfel, Quader, Block, Klotz, Quadrat,` etc. (cube, cuboid, block, ???, square[4]). One of these has to be chosen. This is mainly done randomly but the process takes into account preference values that guarantee that more unusual words will not be chosen as often as frequent words.

$\texttt{[PRED block, CAT n]}_{29}$ will trigger the N-specific choosing process. It is most likely that the result of this process will be $\texttt{[STEM Würfel, CAT n]}_{29}$. $\texttt{[PRED zu, CAT prep, ARG 29]}_{29}$ will yield $\texttt{[STEM zu, CAT prep, SUBCAT np}_{29}\texttt{]}_{29}$. A noun phrase is subcategorized, as the type of the argument is `OBJECT`.

3.4.5 Morphological Processing

The chosen lexemes have to be inflected. Again, this morphological processing consists at the moment of a lookup process. All possible word forms are listed along with the needed agreement information.

Thus, $\texttt{[STEM Würfel, CAT n]}_{29}$ is changed into $\texttt{[SURF Würfel, CAT n]}_{29}$ and $\texttt{[STEM zu, CAT prep]}_{29}$ into $\texttt{[SURF zu, CAT prep]}_{29}$ which are used during the instantiation process.

4 Conclusion

The systems described have been implemented in Java and tested in the above-mentioned context. Currently, the conceptualization is a pure bottom-up mechanism. No deliberative information like a partner model or intentions is taken into account. A discourse model could improve the selection mechanism for CRs significantly. In IPAGE, it could also be used for the treatment of ellipses or anaphora.

References

R. A. Brooks. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14–23.

W. Finkler. 1997. *Automatische Selbstkorrektur bei der inkrementellen Generierung gesprochener Sprache unter Realzeitbedingungen.* Infix.

S. Franklin and A. Graesser. 1996. Is It an Agent, or Just a Program?: A Taxonomy for Autonomous Agents. In J. P. Müller, M. J. Woolridge, and N. R. Jennings, editors, *Intelligent Agents III. Agent Theories, Architectures, and Languages.*, pages 21–35, Berlin. Springer.

K. U. Goecke and J.-T. Milde. 1998. Situations- und Aktionsbeschreibungen durch einen teilautonomen Montageroboter. In *Computers, Linguistics, and Phonetics between Language and Speech.*, volume 1, pages 331–335, Frankfurt a. M. Peter Lang.

H. P. Grice. 1975. Logic and conversation. In P. Cole and J. L. Morgan, editors, *Syntax and Semantics*, volume 3, pages 41–58. Academic Press, New York.

R. Jackendoff. 1990. *Semantic Structures.* MIT Press, Cambridge, MA.

T. Längle, T.C. Lüth, E. Stopp, G. Herzog, and G. Kamstrup. 1995. KANTRA – A Natural Language Interface for Intelligent Robots. Bericht 114, Universität des Saarlandes, 3. SFB 314.

W. J. M. Levelt. 1989. *Speaking: From Intention to Articulation.* MIT Press, Cambridge, MA.

J.-T. Milde, S. Strippgen, and K. Peters. 1997. Situated communication with robots. In *1st Int. Workshop on Human-Computer Conversation.*

H.-J. Novak. 1987. *Textgenerierung aus visuellen Daten : Beschreibungen von Straßenszenen.* Springer, Berlin.

E. Reiter. 1994. Has a Consensus NL Generation Architecture Appeared, and is it Psycholinguistically Plausible? In *Proc. of the 7th Int. Workshop on Natural Language Generation*, pages 163–170.

[3] Lexemes are understood here as the general form of a word, i.e. not inflected. This usage is not to be confused with the notions of *lemma* and *lexeme* as introduced by Levelt (Levelt, 1989).

[4] Not correct but found in actual language data.