gefördert vom
Bundesministerium
für Forschung
und Technologie

Verbmobil
Verbundvorhaben

Default Handling in Incremental Generation

Karin Harbusch
Anne Kilger

DFKI

Gen-ichiro Kikui

ATR

VM-Report 22
Juli 1994

Juli 1994

Karin Harbusch
Anne Kilger

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken

Tel.: (0681) 302 - 5271|5255
Fax: (0681) 302 - 5341
e-mail: `harbusch|kilger@dfki.uni-sb.de`

Gen-ichiro Kikui

ATR Interpreting Telecomm. Research Labs*
2-2 Hikaridai, Seikacho, Soraku-gun
Kyoto 619-02, Japan

Tel.: (+81 7749) 5 1301
Fax: (+81 7749) 5 1308
e-mail: `kikui@itl.atr.co.jp`

Gehört zum Antragsabschnitt: 9 Spontansprachliche und inkrementelle Generierung

Das diesem Bericht zugrundeliegende Forschungsvorhaben wurde mit Mitteln des Bundesministers für Forschung und Technologie unter dem Förderkennzeichen 01 IV 101 K/1 gefördert. Die Verantwortung für den Inhalt dieser Arbeit liegt bei den Autoren.

Abstract

Natural language generation must work with insufficient input. Underspecifications can be caused by shortcomings of the component providing the input or by the preliminary state of incrementally given input. The paper[1] aims to escape from such dead–end situations by making assumptions. We discuss global aspects of default handling. Two problem classes for defaults in the incremental syntactic generator VM–GEN are presented to substantiate our discussion.

1 Motivation

Natural Language Generation, i.e., the process of building an adequate utterance for some given content, is by nature a decision–making problem (Appelt, 1985). Internal decisions are made on the basis of the specified input. Unfortunately, input information can be insufficient in two respects:

- If the input structure for generation is provided by another AI–system, *global problems in producing sufficient input information* for the generator may occur, e.g., because of translation mismatches in machine translation (Kameyama, 1991). In this case, the generator either has to use a default or formulate a request for clarification in order to be able to continue its processing, i.e., to produce an utterance. During simultaneous interpretation requests are rather unusual. Here defaults allow for a standalone handling of the problem. For example, problems during speech recognition of automatic interpretation can lead to results like "the (man/men) will come to the hotel tomorrow". If the system is not able to give a preference for one of the alternatives, e.g., by evaluating context information, the generator has to choose a probable number value on its own to complete verbalization.
- Furthermore, for incremental generation, the input information is produced and handed over step by step, so that it can be *temporarily incomplete* — although as a whole it may become sufficient. This behaviour of a generator is motivated by psycholinguistic observations which show that people start speaking before all necessary linguistic material has been chosen

*The author is currently at NTT Network Information Systems Laboratories (kikui@nttnly.ntt.jp).

[1]This paper will appear in the Proceedings of the 15th International Conference on Computational Linguistics (COLING–94), Kyoto, Japan, 1994.

(e.g., articulating a noun phrase before the dominating verb is selected). As a consequence of underspecification, incremental generation is essentially based on working with defaults. Elements are uttered before the processing or input consumption has been finished. (Kitano, 1990) gives an example for defaults in the context of simultaneous interpretation: In Japanese, negation is specified at the end of the sentence while in English, it has to be specified in front of the finite verb. Therefore, during Japanese–English translation, where analysis, transfer, and generation are performed in a parallel and incremental way, the system has to commit, e.g., positive value before knowing the actual polarity[2].

Generally speaking, *default handling* specifies how processing, i.e., further decision–making, can continue without sufficient input information. So, one can compare default handling with *advice to the system.* For reasons of uncertainty of assumptions, incremental systems with this facility must be able to repair the default decision when the assumption turns out to be wrong by information given later. Catching on to the above example, there can be a negation specifier given at the end of the Japanese input sentence which cannot be simply integrated into the output sentence because the finite verb has already been uttered. In this case, the output has to be repaired, e.g., by repeating parts of the utterance: "I will be able to meet you ...oops ...I won't be able to meet you at the hotel this evening."

In the following sections, we argue for the appropriateness of *processing–conforming* default handling. Basically, the processing–conforming mode makes the overall system *homogeneous* because the combination of default–caused processing and input–licensed processing requires no specific description. The homogeneity becomes especially helpful in the case where the input verifies the default assumption rendering unnecessary any recomputation. For the opposite case where the default must be withdrawn we have to mark all defaults. Even more homogeneity is introduced to an incremental system if the default descriptions are given *in terms of input specifications.* This representation allows for easy checking the coincidence between a chosen default and input given later.

The content of this paper can be summarized as follows. Section 2 provides a general description for defaults in generation emphasizing the specific requirements in an incremental system. After identifying the conditions under which defaults are triggered (section 2.1), the application of a default

[2]Alternatively, the system could use the dialogue context to infer a negation value +/-.

(section 2.2) and the definition of its description (section 2.3) is outlined. The crucial case of removing defaults not coinciding with newly arriving input in an incremental system is discussed in section 2.4.

In section 3, this mechanism is applied to the incremental sentence generator VM–GEN. In the beginning of the section, the basic design of the system is outlined. Later on, default handling is included and exemplified for two general cases.

In the final section we summarize the main results of the paper. Furthermore, we discuss how default handling can be adapted to *multilingual generation*, as required by the speech–to–speech translation system VERBMOBIL (Block et al., 1992).

2 General Discussion of Defaults

In the literature of non–incremental generation, the need for defaults is hardly ever taken into account. The common point of view restricts the input to be sufficient for generation (see, e.g., the *Text Structure* by (Meteer, 1990) for a syntactic generator). In incremental generation, most authors agree on the necessity of using defaults (see, e.g., (De Smedt, 1990; Kitano, 1990; Ward, 1991)). Nevertheless, they do not in sufficient depth answer the question of how to guide the processes of default handling and repair within a generator. This problem is the starting–point for the following considerations.

We assume that generation is a decision–making process with the aim of producing a plausible utterance based on given information. As mentioned in section 1, there are cases where this process stops (caused by underspecification of the input) before finishing its output.

We define a module named *default handler* which tries to resume the process by giving advice to it, i.e., by making assumptions about the missing input specification. With respect to this task it is discussed

1. in which situations defaults are applied (see section 2.1),
2. how default handling is integrated into a system (see section 2.2),
3. how the knowledge for default handling is described (see section 2.3), and
4. how assumptions are cancelled when they turn out to be inconsistent with newly arriving input (see section 2.4).

In incremental generation, as mentioned in section 1, interleaved input consumption and output production causes specific default situations. An incremental processing scheme allows for an increase of efficiency and flexibility, e.g., by making the analysis and generation processes of a system for simultaneous interpretation overlap in time. There are two competing goals of incremental generation for spoken output, that must be taken into account when estimating the usefulness of defaults:

Fluency: Long hesitations should be avoided during the production of an utterance, in order to be *acceptable* to the hearer[3].

Reliability: Errors in an utterance may cause misunderstanding. In most cases, errors should be recovered by appropriate self–corrections[4]. Excessive use of self–corrections or erroneous expressions should be avoided because they decrease *intelligibility* of the utterance.

Obviously there is a trade–off between fluency and reliability: maximal reliability requires 'secure' decisions and therefore leads to output delay. On the other hand, maximal fluency necessitates the use of assumptions and repair, respectively.

2.1 When to Trigger Default Handling

We define as *default situation* the situation where a generation system has not yet finished the utterance but at the same time has consumed all given input and is not able to continue processing. In non–incremental generation, this corresponds to the fact that the input lacks necessary information, because the entire input is assumed to be given at one time (e.g., the undecidable number value of the example described in section 1). Thus, default handling should be triggered immediately.

In incremental generation, however, the system may get a new piece of information later on that enables it to continue processing (e.g., the specification of a negation value + as outlined in the example in section 1). Therefore, possible alternatives are either to wait for the next input or to trigger default handling. The former violates the fluency goal, the latter may violate the reliability goal. We propose the explicit use of *time–limits for delay intervals*[5].

[3] Humans often fill such pauses with *fillers* like "er" or "what shall I say".

[4] Sometimes, correction is unnecessary if (the speaker believes that) the hearer can infer the intended utterance from erroneous speech.

[5] An explicit parameter expressing the desired degree of fluency influences the time–limits.

Furthermore, the *certainty of a default* is described by a value. As soon as a default situation is identified, the certainty of the default is checked to see whether it exceeds a predefined threshold that determines the degree of fluency/reliability[6].

Each application of a default decreases the global certainty of the system's state. Consequently, there should be a *limit for the maximal number of defaults* applicable to the same sentence.

2.2 How to Integrate Default Handling

Basically, there are two strategies to integrate default handling into ongoing processing.

Defaults may be handled in a way that differs from the 'normal' processing of the system, e.g., as short–cuts. One advantage can be an efficient handling of defaults. Furthermore, the designer of the default component is completely free in deciding about the realization of defaults in the system. A disadvantage is the difficulty of providing consistency between default–caused and input–licensed processing.

Alternatively, the ongoing processing can deal with the default values in an ordinary manner (processing–conforming default handling). This may be less efficient but guarantees consistency during processing, especially in case of a replacement by an input–licensed value. For incremental generation, the system has to provide repair facilities in any case. So, they can also be used for *non–monotonic* modifications of default–caused results. We take this option in order to make the overall system *homogeneous*.

2.3 How to Describe Defaults

The knowledge source that is used for default handling should provide the most plausible actions for a default situation. We represent the knowledge as a set of heuristic rules called *default descriptions*. A default description defines a set of operations that should be carried out in a certain situation where the generation process can not be continued. A default description has the following form:

[6]The basis for assigning certainty values to defaults should be a corpus study that can be used to find statistical evidence for various features with alternative values (like number, voice, ..., see, e.g., (Bock and Warren, 1985)).

$$\begin{bmatrix} default & \Rightarrow & default & ; & certainty \\ preconditions & & body & & value \end{bmatrix}$$

The set of *default preconditions* defines tests that are applied to the given situation in order to find out whether the corresponding default body can be activated. They include tests for the existence of particular information, tests for the structure under creation and tests for the state of processing.

The *default body* describes how to continue processing with defaults in an adequate way. For incremental systems, we propose to express the body as a specification of input increments. An important prerequisite is that the size of increments is defined flexibly enough to cope with varying amounts of information. Obviously, an important advantage of this approach is homogeneity of the overall system. Especially, the homogeneous representation of default–caused and input–licensed structures is the easiest and most direct way to test coincidences or contradictions between default–specified and input–caused values. In section 3, this approach is outlined by different examples. For non–incremental systems, an operational approach is preferable since there is no way to consume additional input increments, presupposing that the input has been considered as a whole before a default situation occurs[7].

If several default preconditions are applicable, the *certainty values* for default descriptions are examined to find which provides the system with the most plausible action.

The individual default descriptions should take into account the global constraints for processing stated in the knowledge sources of the system. For example, the assumption of nominative case for a German NP complement can regularly be made only once for the same verb. For reasons of homogeneity, the default description should at least be compatible with the specifications of the knowledge used for basic processing. In order to guarantee consistency, default descriptions should merely contain what is orthogonal to the basic knowledge sources.

[7]The difference between incremental and non–incremental generation becomes smaller, if we assume that defaults in a non–incremental system can be triggered after the system has only considered parts of its input information. In this case, the input considered after default handling becomes comparable to later increments.

2.4 How to Cancel Defaults

The repair of false assumptions is a crucial point of default handling in the context of incremental processing because the default information does not remain locally but can cause further decisions of the system. Contrarily, for non–incremental input there will be no value given that can contradict default values.

As a first step of repair, inconsistencies between input–provided and default–caused values are identified by simply matching the values. Then effects of the respective defaults are withdrawn introducing the input–provided values into the system. Generally, a decision during generation influences other decisions all over the system. Thus the effect of a default body may be propagated through the entire system (e.g., choosing a construction of main clause with causal subordinate clause influences the choice of syntactic realizations).

Roughly speaking, withdrawing a default assumption can be realized by *backtracking* to the earlier state of the system where the default had been introduced or by *non–monotonic changes* to the current state of the system. The disadvantage of backtracking is that partial results are thrown away which could be reused during further processing. Non–monotonic changes preserve these results. In this framework, cancelling defaults requires the system to identify which results are caused by default handling. *Dependency links* between the immediate result of a default body and results of the influenced decisions allow for this identification. The disadvantage of non–monotonic changes is the complexity of computation, e.g., supported by a truth maintenance system. When designing an incremental system, simple backtracking is ruled out because the part of the sentence uttered cannot be withdrawn after it has been perceived by the addressee of the message[8].

So, we end up with a processing–conforming default handler for generation realizing repair by non–monotonic changes.

3 Examples of Defaults in VM–GEN

The adaptation of our general discussion of default handling to the system VM–GEN not only provides concrete examples for the reader but also shows

[8]If some phrases influenced by defaults have already been verbalized, the effect of verbalization can be cancelled by using repair words like “oops” or “sorry” when starting the modified utterance.

that a homogeneous combination of default handling, regular processing, and utterance repair is possible.

The syntactic generator VM–GEN is a further development of TAG–GEN (Kilger, 1994) within the framework of VERBMOBIL, a speech–to–speech translation system. Its usefulness for simultaneous interpretation results from its *incremental and parallel style of processing*. VM–GEN is able to consume input increments of varying size. These increments describe lexical items or semantic relations between them. Single input increments are handed over to objects of a distributed parallel system, each of which tries to verbalize the structure that results from the corresponding input increment. VM–GEN uses an extension of *Tree Adjoining Grammars* (TAGs, cf. (Joshi, 1985)) as its syntactic representation formalism that is not only adequate for the description of natural language but also supports incremental generation (Kilger and Finkler, 1994).

In the following, we introduce examples for default processing triggered during the German *inflection process* in VM–GEN to substantiate the global statements made in section 2. Inflection uses some syntactic properties of an element to compute its morphological form. This information has partly to be specified in the input (e.g., the number for a noun) and is partly inherited from other elements (e.g., the number for a verb or the case for a noun). The two reasons for missing information necessitate different methods of treatment which nevertheless both can uniformly be integrated into regular processing.

If information of the first type is missing (e.g., because of problems during analysis, see section 1), an assumption can be made *locally* by simulating the respective part of the input. The default for missing number information in VM–GEN would look as follows:

$$\left[\begin{array}{l} (cat(OBJ){=}N) \\ (number(OBJ){=}NIL) \end{array} \Rightarrow \begin{array}{c} (ENTITY\ \ OBJ \\ (number\ \ sg)) \end{array} ; 0.8 \right]^{9}$$

The set of default preconditions is applied to all[10] objects (OBJ) of VM–GEN in order to test the kind of underspecification ('number' in the example). The

[9]'ENTITY' introduces information about a lexical item. For reasons of incrementality, there may be several ENTITY–packages specified for the same item which are composed to receive the global information. For certainty values, we use values between 0 and 1, where 1 means high reliability.

[10]In the actual implementation we preselect candidates with missing values for reasons of efficiency.

default body introduces a new value (sg) by creating an input increment as a default. The test for coincidence with the input–licensed value is realized by a comparison in the objects of VM–GEN. There is a unique association of input increments and objects of VM–GEN (OBJ is used as identifier) that allows for translating an input modification into a modification of the state of the respective object. In case of contradictions the default and all default–caused decisions are revised[11] (see below).

Making an assumption can be influenced by *global* constraints. An example, which is well studied in psycholinguistics, is the utterance of a noun before the verb has been chosen. If, e.g., the noun "Besucher" (English: "visitor") is known to be the agent of an action, it may be uttered as subject in the first position of the sentence by default. This treatment presupposes the choice of a 'dummy' verb, which at least subcategorizes a subject and has active voice[12]. The use of a dummy verb and an underspecified verbal structure the NP is integrated into allows for a simple global test that rules out the same case value assignment to different NP complements as it is required for most of the German verbs. This rule is represented in the grammar as a part of the description of subcategorization frames for verbs. For reasons of homogeneity we use the information stored in the syntactic knowledge sources of VM–GEN for expressing syntactic constraints during default handling as well. The advantage of this approach is, that processing is continued in a consistent way, which eases the introduction of the input–licensed value. One default for choosing a missing case–value is specified as follows:

$$\left[\begin{array}{lcl} & & (ENTITY\ OBJ' \\ (cat(OBJ){=}N) & & \quad (CAT\ v) \\ (case(OBJ){=}NIL) & & \quad (VOICE\ active)) \\ (function(OBJ){=}agent) & \Rightarrow & (RELATION\ REL \\ (lemma(head(OBJ)){=}NIL) & & \quad (HEAD\ OBJ') \\ & & \quad (MODIFIER\ OBJ)) \end{array} ; 0.8 \right]^{13}$$

The default preconditions of the rule characterize a situation where an object (OBJ) contains no information about the case but identifies the input category as 'N' for noun. Furthermore, the semantic function of the object is specified as 'agent' but no verb defined yet (lemma(head(OBJ))=NIL) in

[11] For ongoing work on repair in VM–GEN see (Finkler, 1994).

[12] This kind of expansion is called "provisional upward expansion" by (De Smedt, 1990).

[13] 'RELATION' introduces the specification of a relation between two lexical items which are identified by the names of their objects.

the head object. That is why, the N–object cannot inherit a case value and also does not know whether it is allowed to occupy the front position in the utterance.

Evaluating the default body, the system creates a V–object OBJ'. On the basis of the input information in (ENTITY OBJ' ...) it chooses a minimal syntactic structure from the inheritance net of the grammar, that just desribes a verb category without concrete filler (a dummy verb) plus a subject complement and active voice for the verbal phrase. Now, the N–structure is combined with the V–structure of the introduced V–object as during normal processing. Therefore, the case value can be inherited. Additionally, the first position can be assigned to the subject which can be uttered now.

The basic VM–GEN module provides repair strategies in order to allow for the specification of additions, modifications and deletions of input increments, i.e., to model a flexible input interface. Three features of the system are basically used for repair: First, input increments are uniquely associated with objects of VM–GEN, so that input modifications can be translated into modifications of the objects' states. Second, each modification of an object's state makes it compare new and old information. In case of a difference, the modified parts are sent to all concerned objects. Third, the dependency relations that determine the communication links between objects allow for a hierarchical organization of the objects, which is the basis for synchronizing repair.

A repair must be triggered in the example described above if, e.g., a verb with voice passive is actually specified. In this case, the mapping of the semantic role 'agent' to the syntactic function 'subject' is revised. The agent now has to be realized as part of a "von"–phrase, e.g. "dieser Termin wird *von dem Besucher* gewünscht." (word–for–word: "this date is whished *by the visitor* (dative object)"). Furthermore, the object checks whether the previously uttered part of the sentence includes some of the revised material (i.e., whether the object itself has participated in uttering). If this is the case, it sends an error message up to the uppermost object of the hierarchy that actually is engaged in uttering. This object is able to synchronize global repair. Up to now, we just realized a simple repair strategy that consists of repeating the concerned parts of the utterance, e.g. "der Besucher ... äh ... dieser Termin wird *von dem Besucher* gewünscht".

4 Discussion

This paper proposes a processing–conforming default handler for generation realizing repair by non–monotonic changes. We provide the system with default descriptions. The set of default preconditions expresses possible reasons for dead–end situations. A default is triggered, if the preconditions match the current situation and the certainty value of the default exceeds the predefined threshold. The default body is expressed in terms of the missing input specification in order to make the system work homogeneously. We have verified the advantages of processing–conforming default handling by implementing a default handler for VM–GEN.

As future work, we will extend the default preconditions towards handling complex contextual information. We will apply default handling to microplanning and lexical choice within VERBMOBIL. With respect to a sophisticated output, we aim to combine VM–GEN with a flexible repair component.

The system VM–GEN is used in the VERBMOBIL scenario for multilingual generation (English, German, and Japanese). We mean by multilinguality that the same processing is applied for different languages. In the underlying knowledge sources language–specific constraints are defined. Default handling can be easily adapted to the requirements of multilingual generation by using language–specific default–descriptions.

For all knowledge sources the question arises how knowledge can be shared. We intend to use *core knowledge sources* for representing common phenomena. The core set of default descriptions for English and German, e.g., contains the description of a reaction to a missing number value for a noun. We aim to develop an efficient storing mechanism using a hierarchy of locally intersecting core descriptions.

References

[Appelt, 1985] Appelt, D. (1985). *Planning English Sentences.* Cambridge, MA: Cambridge University Press.

[Block et al., 1992] Block, H.–U., Bosch, P., Engelkamp, J., v. Hahn, W., Hauenschild, C., Höge, H., Rohrer, C., Tillmann, H., G., Wahlster, W., Waibel, A. (1992). *Wissenschaftliche Ziele und Netzpläne für das*

VERBMOBIL–Projekt. Technical report, German Research Center for Artificial Intelligence (DFKI GmbH), Saarbrücken, Germany, 1992.

[Bock & Warren, 1985] Bock, J., and Warren, R. (1985). *Conceptual accessibility and syntactic structure in sentence formulation. Cognition, 21*, 47-67, 1985.

[de Smedt, 1990] De Smedt, K. (1990). *Incremental Sentence Generation: a Computer Model of Grammatical Encoding.* PhD thesis, Nijmegen Institute for Cognition Research and Information Technology, Nijmegen, NICI TR No 90–01.

[Finkler, 1994] Finkler, W. (1994). *Performing Self–Corrections During Incremental Natural Language Generation.* Document, German Research Center for Artificial Intelligence (DFKI GmbH), Saarbrücken, Germany, 1994. to appear.

[Joshi, 1985] Joshi, A. (1985). *An Introduction to TAGs.* Technical Report MS-CIS-86-64, LINC-LAB-31, Department of Computer and Information Science, Moore School, University of Pennsylvania.

[Kameyama et al., 1991] Kameyama, M., Ochitani, R., and Peters, S. (1991). *Resolving Translation Mismatches With Information Flow.* 29th Annual Meeting of the Association for Computational Linguistics, Berkeley, CA, pp. 193-200.

[Kilger & Finkler, 1994] Kilger, A. and Finkler, W. (1994). *TAG–based Incremental Generation.* Technical report, German Research Center for Artificial Intelligence (DFKI GmbH), Saarbrücken, Germany. to appear.

[Kilger, 1994] Kilger, A. (1994). *Using UTAGs for Incremental and Parallel Generation.* Computational Intelligence. to appear.

[Kitano, 1990] Kitano, H. (1990). *Incremental Sentence Production with a Parallel Marker-Passing Algorithm.* 13th International Conference on Computational Linguistics, Helsinki, Finland, pp. 217 - 222.

[Meteer, 1990] Meteer, M. (1990). *The "Generation Gap": the Problem of Expressibility in Text Planning.* Department of Computer and Information Science, University of Massachusetts, Amherst, MA, BBN Report No. 7347.

[Ward, 1991] Ward, N. (1991). *A Flexible, Parallel Model of Natural Language Generation*. PhD thesis, Computer Science Division (EECS), University of California, Berkeley, CA, Report No. UCB/CSD 91/629.