漸進的発話生成のための表層生成モデル*

坂庭克幸　乾健太郎　石澤宏明　徳永健伸　田中穂積†

1 Introduction

As many researchers have pointed out, task-oriented dialogues such as explanatory dialogues should be incremental and interactive (e.g. [3, 13]). These two requirements come from an important aspect of communication: the interlocutor can never be certain of the correct model of her counterpart's beliefs or knowledge. Dialogue systems designed by Cawsey[3], Moore[13], and Carletta[2]. are good examples of systems that address this issue.

Dialogue has another important facet: the grain size of primitive utterances (which we term *utterance units* in this paper). In dialogue, the overall information that is to be conveyed is usually decomposed into smaller chunks of information and conveyed individually by a sequence of utterance units. An utterance unit may be a sentence, or alternatively a phrase or even a word. This aspect has received little attention for existing explanatory dialogue systems. Most existing systems are designed under the assumption that the grain size of an utterance unit is no smaller than a clause, with phrase-sized utterance units interpreted as elliptical expressions (e.g. the three systems cited above). This is mainly because the linguistic theories underlying existing plan-based dialogue systems, such as Mann and Thompson's RST and Austin's speech act theory, assume that a speech act is realized by a single utterance unit, which is defined to be the size of a clause.

However, we have several good reasons to reconsider this conventional assumption. In dialogue, the hearer gets the chance to give feedback, such as a back channel cue, between the speaker's utterance units. In fact, it is reported that the speaker usually utilize particular prosodic patterns to let the the hearer know that the current utterance unit has finished, thereby prompting the hearer's feedback (e.g. [12]). This implies that the speaker to some extent can control the timing and frequency of the hearer's feedback. The smaller the grain size of an utterance unit, the more often the recipient can give feedback. Allowing the recipient to give frequent feedback would improve the efficiency of communication in settings of the following types:

- the information that is to be conveyed is complicated,
- the information provider has great difficulty in modelling what the recipient already knows,
- the channel is noisy.
- the recipient is required to understand the conveyed information correctly (e.g. in order to accomplish a given task),

Furthermore, it is reported that in Japanese dialogue, back channel cues tend to be given much more frequently than, for example, in English dialogue[4]. Thus, in designing Japanese dialogue systems, realization of fine-grained utterance units is further significant. Although several works for incremental generation have already been reported (e.g. [5–7, 18]), they did not directly address design issues for fine-grained incremental utterance generation in interactive settings .

Let us call *utterance planning* the generation subtask involving delimitation and organization of utterance units. As a minimum requirement for utterance planning, we need to take the following constraints into account:

The feedback prompting constraint: The speaker suspends an utterance, attaching a follow-up prompt at its end, if she needs to confirm the attainment of the intended goals.

The efficiency constraint: The speaker tends to try to include unit as content as possible in a single utterance unless she needs the hearer's feedback.

The realizability constraint: Each delimited content must be linguistically realizable (i.e. there must be some way of realizing each content).

The simplicity constraint: If the original content is too complex to realize as a single utterance unit, it should be subdivided.

The cohesiveness constraint: Neighboring utterance units should be locally cohesive.

The former two constraints are obviously related to the intentional structure of dialogue, whereas the latter three constraints additionally associate the utterance planning task with the grammar and lexicon. Thus, the main issue in utterance planning is what architecture would facilitate the process of effectively compromising between these constraints.

In what follows, we first review the state-of-the-art text generation model, and then discuss how to enhance it in order to implement a fine-grained incremental utterance generator that takes all the above constraints into consideration. Throughout this paper, we restrict our discussion to explanatory dialogue, assuming the system and user to be an explanation provider and explanation recipient, respectively.

2 The three-layered model

Starting with what is called sentence planning (e.g. [10, 16, 17]) can be one promising approach to utterance planning, since the utterance planning task inherits most of its features from sentence planning. In general, sentence planning involves such processes as sentence delimitation, local discourse organization, phrase ordering, and choice of references, playing the role of bridging the gap between the content planning and sentence realization tasks. Such three-layered generation models have several advantages, among which the most significant is in the modularity of each layer. Content planning tends to be domain-dependent; for example, the plan library may be replaced depending on the do-

*A Model for Surface Generation of Fine-grained Incremental Utterances

†Sakaniwa, K., Inui, K., Ishizawa, H., Tokunaga, T., and Tanaka, H. Department of Computer Science, Tokyo Institute of Technology {sakaniwa, inui, ishizawa, take, tanaka}@cs.titech.ac.jp

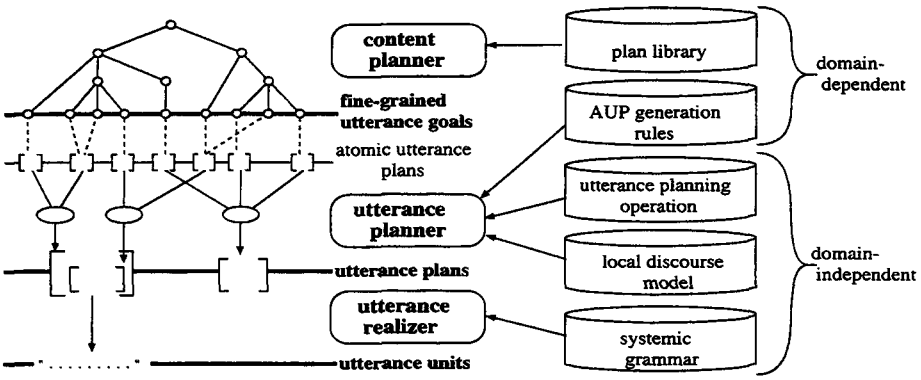

Figure 1: Three-layered incremental utterance generation model

main, also making the ontology of the content representation domain-dependent. In contrast, the sentence realizer can be expected to be designed as a domain-independent general module similarly to such existing sentence generators as Penman and its multilingual extension KPML[1]. By introducing the sentence planner as the bridge between these two modules, one can maintain the domain-independency of the sentence realizer.

However, since previous three-layered models were originally designed for monologic text generation, we need to enhance them in the following respects:

(a) In the previous models, consideration of the feedback prompting constraint is lacking. We need to newly take this into account.

(b) Due to the feedback prompting constraint, utterance units have varieties of grain-size (from a sequence of sentences to a single word). We need a mechanism for controlling the grain-size of utterance units over such a wide range.

In our enhanced model, we call the intermediate and linguistic modules the *utterance planner* and *utterance realizer*, respectively, to distinguish our model from previous models. Our model is illustrated in Figure 1. The content planner infers what to say, producing a set of *fine-grained utterance goals* (FUGs). Before content planning completes, the utterance planner starts to consume the FUGs, generating a sequence of *utterance plans* (UPs), which will then be handed to the utterance realizer[1]. Thus, these three modules theoretically work in parallel as in, for example, De Smedt and Kempen's incremental generation model[5] and Reithinger's POPEL[18].

In the following sections, we first discuss design issues for FUGs, and then describe the process of generating fine-grained utterances, focusing on the utterance planning process. Design issues for content planning are beyond the scope of this paper. We simply assume that the utterance planner is input a set of FUGs generated from the content planner, whether it is plan-based or schema-based, at each time of dialogue.

3 Fine-grained utterance goals

As illustrated in Figure 1, the content planner and utterance planner interact through a set of fine-grained utterance goals (FUGs). Since these utterance goals are more fine-grained than any conventional surface speech act, the utterance planner is able to infer what has been done and what is to be done on a certain fine-grained level. The utterance planner, on the other hand, is also able to generate fine-grained utterance plans corresponding to fine-grained utterance goals. Thus, with such an interface between content planning and utterance planning, the system acquires the competence to generate utterance units of various grain-sizes. This is an initial step to addressing issue (b) mentioned above.

To explore how human interlocutors produce fine-grained utterances and what kinds of FUGs are required, we analyzed several dialogues (roughly eighty minutes long in total) from the NTT Japanese route navigation dialogue corpus[14][2] and the Chiba University MapTask dialogue corpus[3]. Through this analysis, we extracted several types of FUGs, some of which are as follows:

- ideational FUGs:
  - refer to an instance:
    `g_refer(`*instance*`)`
  - describe the class of an instance:
    `g_type(`*instance*`,`*class*`)`
  - describe an attribute of an instance:
    `g_attrib(`*instance*`,`*attribute*`,`*value*`)`
- interpersonal FUGs:
  - confirm the attainment of a FUG:
    `g_confirm(`*fug*`)`
  - describe the intentional relation between FUGs:
    `g_subgoal(`fug_1`,`fug_2`)`
  - express the attitude to an event instance:
    `g_attitude(`*attitude*`,`*instance*`)`

[1] There also should be a fourth module, the utterance articulator, below the utterance realizer. Despite noticing the importance of utterance articulation, we do not discuss it in this paper.

[2] A collection of dialogues wherein the explanation provider explains to the explanation recipient the route from a certain place to her/his house.

[3] We analyzed a sort of "beta version" of the corpus (http://www.l.chiba-u.ac.jp/MapTask.html).

- textual FUGs:
  - attain fug_1 before fug_2:
    `g_order(`fug_1`,`fug_2`)`
  - topicalize an instance:
    `g_topic(`*instance*`)`

The interpersonal FUG `g_confirm(`*fug*`)` is introduced in order to address issue (a) mentioned above. This type of goals are employed to represent the content planner's request to confirm the attainment of an FUG *fug*. Note that, since the choice of linguistic means of prompting the hearer's feedback requires linguistic knowledge, it should not be a matter for content planning. In our model, the content planner only generates abstract utterance goals such as `g_confirm(`*fug*`)`, entrusting the task of generating feedback prompts to the utterance planner.

Figure 2 shows an example set of FUGs from which an utterance unit appearing in the NTT dialogue corpus is generated.

```
ref1:g_refer(act1)
typ1:g_type(act1,turn)
ref2:g_refer(plc1)
typ2:g_type(plc1,cross)
ref3:g_refer(crd1)
typ3:g_type(crd1,next)
att1:g_attrib(act1,place,plc1)
att2:g_attrib(plc1,cardinal,crd1)
cnf1:g_confirm(ref2)
atd1:g_attitude(request,act1)
```

Figure 2: An example set of FUGs

4 Utterance Planning

The utterance planner is responsible for utterance delimitation, phrase ordering and lexical choice. These task are performed through the following two types of operations:

- mapping FUGs to *atomic utterance plans* (AUPs)
- aggregating AUPs to produce a sequence of UPs

4.1 Mapping from FUGs to AUPs

AUPs are atomic fragments of UPs. The knowledge for mapping from FUGs to AUPs is described as a set of production rules (*AUP generation rules*). Each AUP generation rule is defined as a production rule of the form:

$$\begin{array}{l}\{G_1{:}FUG_1, G_2{:}FUG_2, \ldots\} \\ \quad \langle Cond_1, Cond_2, \ldots\rangle\end{array} \Rightarrow \begin{array}{l}\{AUP_1, AUP_2, \ldots\} \\ \langle Constr_1, Constr_2, \ldots\rangle\end{array}$$

where the left hand side consists of the source FUGs ($G_i{:}FUG_i$, where G_i is the identifier of FUG_i) and the applicability conditions ($Cond_j$) of the rule, and the right hand side consists of the target AUPs and the additional constraints ($Constr_i$) on those AUPs (*UP constraints*). Each AUP is represented as a type feature structure for the reason we mention in Section 4.2.

The following is an example of an AUP generation rule, which is applicable to the FUG `att1` in Figure 2:

$$\{G{:}\texttt{g\_attrib}(X,\texttt{place},Y)\}\,\langle\rangle \Rightarrow$$

$$\left\{\boxed{2}\left[\begin{array}{l}\text{inst:}X \\ \text{sem:}\left[\text{place:}\boxed{1}\right]\end{array}\right], \boxed{3}\left[\begin{array}{l}\text{inst:}Y \\ \text{sem:}\boxed{1}\end{array}\right]\right\}$$

$$\langle \boxed{2} \succ \boxed{3} \rangle$$

A symbol beginning with a capital letter denotes a logical variable, and the symbol $\succ$ denotes the whole-part relation between AUPs, i.e. $aup_i \succ aup_j$ means that aup_j is the value of an attribute included in aup_i.

The interpersonal FUG `confirm(`*fug*`)`, which is employed to deal with the feedback prompting constraint, is normally mapped to some linguistic/prosodic specifications by way of the UP constraint c_cut(*aup*) in the manner exemplified in the next AUP generation rule:

$$\{G{:}\texttt{g\_confirm}(G1)\}\,\langle G1{:}\texttt{g\_refer}(X)\rangle \Rightarrow$$

$$\left\{\boxed{2}\left[\begin{array}{l}\text{inst:}G \\ \text{sem:}\left[\begin{array}{l}\text{declarative} \\ \text{\& pointing} \\ \text{\& continuing-type} \\ \text{goal :}\boxed{1}\end{array}\right]\end{array}\right], \boxed{3}\left[\begin{array}{l}\text{inst:}X \\ \text{sem:}\boxed{1}\end{array}\right]\right\}$$

$$\langle \text{c\_cut}(\boxed{2}), \text{c\_order}(\boxed{2},\boxed{3}), \boxed{2} \succ \boxed{3} \rangle$$

c_cut(*aup*) denotes the constraint that a feedback prompt should be put immediately after *aup* is executed. c_order(aup_i,aup_j) denotes the constraint that aup_i should be executed before aup_j.

The role of AUP generation rules is to bridge the gap between the domain-dependent ontology of the domain resources and the domain-independent ontology of the linguistic resources. That is, we develop AUP generation rules depending on the domain, whereas maintaining the domain-independency of the ontology of the target AUPs then facilitates developing the rest of the linguistic resources domain-independently (see Figure 1). Since each AUP generation rule is associated with only a limited number of FUG(s), this task tends to be rather simple, which facilitates the development and maintenance of AUP generation rules themselves. Furthermore, this simplicity will not diminish the diversity of utterance plans that the system can generate, because utterance plans are expected to emerge from diverse choices in applying AUP generation rules and combining AUPs, which are both executed throughout the aggregation process described below.

4.2 Aggregation of AUPs

Since our utterance goals are fine-grained, AUPs also tend to be fragmental, and need to be aggregated together again due to the efficiency constraint mentioned in Section 1. In order to facilitate this aggregation process, we represent AUPs (and UPs) as typed feature structures, and use the unification operation to merge AUPs. This unification-based processing facilitates the aggregation operation in the following respects:

- The monotonicity of unification allows us to describe AUP generation rules purely declaratively, which facilitates development and maintenance.
- As we mention below, since the ontology of AUPs is defined in a systemic fashion, the unifiability of AUPs guarantees the realizability of the resultant UP.
- Each AUP has the particular attribute inst, which specifies the instance associated with the AUP. Thus, by unifying AUPs, the system can correctly aggregate the semantic specifications for each instance.

Due to the efficiency and simplicity constraints, the utterance planner determines the method of applying AUP generation rules and combining AUPs according to a set of general heuristics, some of which are:

- An AUP generation rule whose target AUPs can be merged with the other AUPs is preferred (the efficiency constraint).
- A combination of AUPs that generates a smaller number of UPs is preferred (the efficiency constraint).
- Aggregation operations that would produce any embedded structure whose depth are greater than a certain threshold is prohibited (the simplicity constraint)

We also take the realizability constraint into account. Since arbitrary aggregation of AUPs might produce a linguistically ill-formed UP, the utterance planner is required to check the linguistic realizability of each resultant UP. However, it is obvious that a naive generate-and-test methodology would make the whole process prohibitively expensive. We need some efficient mechanism to examine the realizability of any given UPs. To solve this problem, we define the ontology of AUPs in a systemic fashion, i.e. we typologize the typed features of AUPs (and UPs) according to system networks[8]. The systemic typology is linguistically motivated, and explicitly describes the inconsistency between features. With such a typology, the unifiability of AUPs guarantees the realizability of the merged UP, which prevents the system from merging linguistically inconsistent AUPs.

The feedback prompting constraint is handled as follows. Given a UP with one or more c_cut constraint(s), the utterance realizer first realizes the utterance corresponding to the whole UP, and then articulates only its first subpart delimited by the c_cut constraint(s). What is to noted here is that the whole UP needs to be handed to the utterance realizer to specify the context of the articulated subpart. This context information and the local discourse model (focus model, old/new information, etc.) are employed to realize cohesive utterance units (the cohesiveness constraint).

5 Conclusion

The utterance planner and realizer are fully implemented on the CUF[4] purely monotonic typed unification system, where the systemic grammar is implemented in a manner analogous to that proposed by Henschel[9]. The resources are experimentally proven to be sufficient to generate several tens of patterns of utterance units, covering the dialogues we analyzed.

The future directions should include the development of the content planner that can generate FUGs according to the user's feedback (e.g. [11]).

Acknowledgements

The authors would like to thank the NTT Information and Communication Systems Labs and the MapTask dialogue project group of Chiba University for allowing us to use their dialogue corpora. They would also like to thank Dr. KATO Tsuneaki (NTT Information and Communication Systems Labs) for his suggestive comments and Mr. Timothy Baldwin for his help in writing this paper.

[4] http://www.ims.uni-stuttgart.de/cuf/dir.html

References

[1] J. Bateman. KPML: the KOMET-Penman multilingual linguistic resource development environment. In *Proceedings of the Fifth European Workshop on Natural Language Generation*, 1995.

[2] J. Carletta. *Risk-taking and recovery in task-oriented dialogue.* Ph.D. thesis, University of Edinburgh, 1992.

[3] A. Cawsey. *Explanation and interaction.* The MIT Press, 1992.

[4] P. Clancy. Written and spoken style in Japanese narratives. In T. Deborah, editor, *Spoken and Written Language*, pp. 55–76. 1982.

[5] K. de Smedt and G. Kempen. Incremental sentence production, self-correction and coordination. In G. Kempen, editor, *Natural Language Generation*, chapter 23, pp. 365–376. Martinus Nijhoff, 1987.

[6] K. de Smedt and G. Kempen. Segment grammar: a formalism for incremental sentence generation. In Paris, et al. [15], chapter 13, pp. 329–349.

[7] K. Dohsaka and A. Shimazu. A computational model of incremental utterance production in task-oriented dialogues. In *Proceedings of the 16th International Conference on Computational Linguistics*, pp. 304–309, 1996.

[8] M. A. K. Halliday. *An Introduction to Functional Grammar.* Edward Arnold, 1994.

[9] R. Henschel. Traversing the labyrinth of feature logics for a declarative implementation of large scale systemic grammars. In *Proceedings of the CLNLP'95*, 1995.

[10] E. H. Hovy and L. Wanner. Managing sentence planning requirements. In *Proceedings of ECAI'96 Workshop: Gap and Bridge: New Directions in Planning and Natural Language Generation*, 1996.

[11] K. Inui, A. Sugiyama, T. Tokunaga, and H. Tanaka. Fine-grained incremental and interactive elaboration in explanatory dialogue. In *Proceedings of ECAI'96 Workshop: Gap and Bridge: New Directions in Planning and Natural Language Generation*, 1996.

[12] H. Koiso, Y. Horiuchi, S. Tsuchiya, and K. Ichikawa. The accoustic properties of "subutterance units" and their relevance to the corresponding follow-up interjections in Japanese. 1995.

[13] J. D. Moore. *Participating in Explanatory Dialogues.* MIT Press, 1995.

[14] Y. Nakano (Ishikawa). Communicative mode dependent contribution from the recipient in information providing dialogue. In *Proceedings of the International Conference on Spoken Language Processing*, pp. 959–962, 1994.

[15] C. L. Paris, W. R. Swartout, and W. C. Mann, editors. *Natural Language Generation in Artificial Intelligence and Computational Linguistics.* Kluwer Academic Publishers, 1991.

[16] S. Prevost. An information structural approach to spoken language generation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, 1996.

[17] O. Rambow and T. Korelsky. Applied text generation. In *Proceedings of the Conference on Applied Natural Language Processing*, pp. 40–47, 1992.

[18] N. Reithinger. POPEL — a parallel and incremental natural language generation system. In Paris, et al. [15], chapter 7, pp. 179–199.