A RATIONAL RECONSTRUCTION OF THE PROTEUS SENTENCE PLANNER

Graeme Ritchie

Department of Artificial Intelligence
University of Edinburgh, Hope Park Square
Edinburgh EH8 9NW

ABSTRACT

A revised and more structured version of Davey's discourse generation program has been implemented, which constructs the underlying forms for sentences and clauses by using rules which annotate and segment the initial sequence of events in various ways.

1. The Proteus Program

The text generation program designed and implemented by Davey (1974,1978) achieved a high level of fluency in the generation of small paragraphs of English describing events in a limited domain (games of "tic-tac-toe"/"noughts-and-crosses"). Although that work was completed ten years ago, the performance is still impressive by current standards. The program could play a game of "noughts-and-crosses" with a user, then produce a fluent summary of what had happened during the game(whether or not the game was complete). For example:

The game began with your taking a corner, and I took the middle of an adjacent edge. If you had taken the corner opposite the one which you had just taken, you would have threatened me, but you took the one adjacent to the square which I had just taken. The game hasn't finished yet.

As well as heuristics for actually playing a game, the program contained rules for text generation, which could be regarded as having the following components (this is not a decomposition used by Davey, but an organisation imposed here in order to clarify the processing):

(a) Sentence planner
(b) Description constructor
(c) Systems network

The third (syntactic) component, is a major part of the original Proteus program, and Davey included a very detailed systemic grammar (in the style of Hudson (1971)) for the area of English he was concerned with; consequently the written accounts (Davey (1974,1978)) deal mainly with these grammatical aspects. However, much of the fluency of the discourses produced by Proteus seems to derive from the crucial computations performed by components (a) and (b), since the syntactic system is largely set up to convert deep representations into surface tokens, without too much regard for global contextual factors. Unfortunately, the written accounts give only a rough informal outline of how these components operated. A completely revised version of Proteus has been implemented in Prolog on a DEC System 10, and this paper describes the working of its sentence planner. The system outlined below is not an exact replication of Davey's program, but is a "rational reconstruction"; that is, an attempt to present a slightly cleaner, more general method, based on Davey's ideas and performing the same specific task as Proteus. Paradoxically, this cleaning up process may lead to minor losses of fluency, where particular effects were gained in Proteus by slightly ad hoc measures.

2. The Sentence Planner

The module which creates the overall clausal structure of each sentence works on a list of numbers representing the course of a game (complete or unfinished), where each square is represented by a number between 1 and 9. The processing carried out by the sentence planner can be seen as occurring in three logical phases:

1. move annotation
2. sentence segmentation
3. case-frame linking

Although these stages are logically distinct, they need not occur wholly in temporal sequence. However, the abstract model is clearer if viewed in separate stages.

2.1. Move Annotation

The system has a set of heuristic rules which enable it to play noughts-and-crosses to a reasonable standard. (A non-optimal set of rules helps to introduce some variety into the play). It uses these move-generating rules to work through the history of the game, computing at each position which move it would have made for that situation and which move-generating rule gives rise to the move actually made at that point. This allows it to mark the actual move in the given history with certain tactical details, using the implicit assumption that whoever made the moves had the same knowledge of the game as the system itself does. The five move-generators are totally ordered to reflect a "priority" or "significance" with

This research was supported by SERC grants GR/B/9874.6 and GR/C/8845.1.

respect to the game, and each move-generator is labelled with one of three categories - "defensive" (e.g. blocking the third square in an opponent's near-complete line), "offensive" (e.g. creating a near-complete line, which thus threatens the opponent) or "neutral" (e.g. taking a square to start the game). In addition to basic organisational entries (square taken, name of player, pointer to preceding move, pointer to following move), the annotation of the moves contains the following information:

(a) generating heuristic(s) - there is a list, in priority order, of the heuristics which could have given rise to that move.

(b) tactically equivalent alternatives - for each heuristic listed in (a), there is a list of the other squares which could also have resulted from that heuristic.

(c) lines involved - for each square mentioned in the various entries, there is a note of which lines (if any) were (or would have been) tactically involved in that move.

(d) better move - if there is a higher priority heuristic that would give rise to a different choice of square, an annotated description of that "better" move is attached.

For example, the game described by the discourse in Section 1 above would initially be just a sequence of square-numbers, together with the name of the first player:

```
user 1 2 3
```

After annotation, the third move (square 3) would have the following information attached:

```
square : 3
heuristics/alternatives : take [9 8 7 6 5 4]
better move :
   square : 9 (1 5 9)
   heuristics/alternatives :
      threaten [7 (1 4 7) 5 (1 5 9) 4 (1 4 7)]
```

2.2 Sentence Segmentation

The sentence segmentation process involves grouping the annotated moves into clusters so that each cluster contains an appropriate amount of information for one sentence. This uses the following guidelines, in the following order, to determine the number of moves within a sentence:

1. If there is just one move left in the sequence, that must be a single sentence.

2. If there are just two moves left, they form a single sentence.

3. If a move is a "mistake" (i.e. there is a tactically better alternative) then start a new sentence to describe it. This is quite a dominant principle, in that the system will perform "look-ahead" of two (actual) moves in the annotated chain to check if there is a mistake looming up.

4. If a move is a combined attack and defence, give it a sentence to itself.

5. If this move is an attack, and the next move successfully thwarts that attack, then put these two moves into a sentence on their own.

6. Put the next three moves in a sentence. (No more than three moves may occur in a single sentence structure).

As well as segmenting the moves, this module attaches to each move a tag indicating its overall tactical relationship to the preceding moves. This is a gross summary of some of the tactical information provided by the annotator, and encodes much of the information needed by the next stage (case-frame linking). There are four tag-values used - "consequence" (the move is a result of the preceding one), "thwart" (the move prevents an attack by the preceding one), "mistake" (the move is a failure to make the best possible move), and "null" (an all-purpose default).

2.3 Case-frame Linking

Once the moves have been annotated, grouped and tagged, their descriptions can be constructed and linked together, to form the internal structure of the sentence. In this process, various case-frame structures are computed from the information attached to each move, and are placed in order, linked by various relationships. There may be, within a sentence, several descriptions associated with a single move, since it is possible for more than one aspect of a move to be mentioned. In each case-frame structure, the other roles will contain suitable fillers - e.g. the square taken (for a "take" description), or the other player (for a "threat") - which are computable from the annotations. Each such case-frame description will eventually give rise to a full tensed clause. In addition, some of these case-frames will have, embedded within them on the "method" case-role, further simple case-frames which will eventually give rise to adjuncts to the tensed clause in the form of verb phrases (e.g. "...by taking a corner.."). Hence the linking process involves selecting those descriptive structures (from the annotations) which are to be expressed linguistically, formulating these as filled case-frame's, and labelling the relationships between these descriptions. Relationships between case-frame descriptions are indicated by attaching to each case-frame a "link" symbol indicating its relation to the surrounding discourse (either within that sentence, or across the preceding sentence boundary). This process is non-deterministic in the sense that there are usually several equally good ways of expressing a given move or sequence of moves within a sentence. The program contains rules for all such possibilities, and works through all the possible combinations using a simple depth-first search. The case-frame construction also determines the clausal structure of the sentence, in that the nesting or conjoining of clauses is fixed at this stage. The clausal structure does not allow recursive levels - there are, for example, no verbs with sentential complements. The case-frame construction and tagging depends on the links inserted by the sentence-segmenter, together with three items of information from the annotations on the moves - whether the move has two aspects, defensive and offensive; any "better" move that has been attached; and whether the tactic-name uniquely defines, within the context, which square must have been taken. The case-frame construction and linking proceeds according to certain guidelines:

1. if the move is a "mistake", indicate that by describing both the better move and the actual move.

2. if a move has two possible descriptions, one "offensive" and the other "defensive", describe both aspects.

3. if a move has two possible descriptions which have the same classification within the set {neutral, offensive, defensive}, then choose the most significant (as determined by the priority ordering of tactics).

4. if two consecutive (actual) moves are such that the second one prevents an attack made by the first, then select the tactics corresponding to these aspects to describe them.

5. if there are no "offensive" or "defensive" aspects listed, use the simple "take" form.

The following rule is also applied to all moves described: if the square taken is not uniquely determined by the tactic-name, and the tactic-name is not "take", then create a "take" case-frame describing the move, and either make it into a separate conjoined clause (if the move has a sentence to itself) or attach it to the main case-frame as the "method".

Since the aim of the current project is to use this discourse domain as a "back-end" for experimenting with functional unification grammar (Kay (1979)), the sentence planner has to produce "fuctional descriptions" to indicate the underlying grammatical form for each sentence. The linked case-frames are therefore reformulated into functional descriptions, with the links attached to the front of each clause determining two aspects of the syntactic structure - the lexical item (if any) to be used as "binder" or "connective" at the front of the clause (again, a non-deterministic choice), and the grammatical features (e.g. modality, aspect) to be added to the clause in addition to those default settings programmed into the system. The ten possible "links", with their possible surface realisations are:

hypothetical	-
altho	although
condante	if
condconse	-
sequence	-
external-contrast	however
internal-contrast	but
conjunction	and
internal-result	and so
external-result	consequently as a result

In addition, the first four of the above links cause the clause to have perfect aspect, "hypothetical" and "altho" cause the presence of the modality "can", and "condconse" results in the modality "will". (Notice that "could" is regarded as the past tense of "can", and "would" as the past tense of "will").

3. Possible Generalisations

After establishing a suitably implementation independent description of the processing necessary to achieve the behaviour of Proteus, the next step should be to try to extract some general notion of how to describe a sequence of events. The domain used here (tic-tac-toe) has the unusually convenient feature that there is a basic canonical form for representing (in a relatively neutral, primitive form) what the sequence of events was. That is, the original list of moves is a non-grammatical representation of the world events to be described. It is not realistic to make such an assumption in general, so a more abstract model may have to take up the planning process at a slightly later stage, when moves already have some form of "descriptions".

REFERENCES

Davey, Anthony (1974) The Formalisation of Discourse Production. Ph.D. Thesis, University of Edinburgh.

Davey, Anthony (1978) Discourse Production. Edinburgh: Edinburgh University Press.

Hudson, Richard (1971) English Complex Sentences. Amsterdam: North Holland.

Kay, Martin (1979) Functional Grammar. Pp.142-158 in Proceedings of the Fifth Annual Meeting of the Berkeley Linguistics Society. Berkeley, CA: University of California.