THE TICC: PARSING INTERESTING TEXT.

David Allport
School of Cognitive Sciences
University of Sussex,
Falmer,
Brighton BN1 9QN
davida%uk.ac.sussex.cvaxa@cs.ucl.ac.uk

ABSTRACT

This paper gives an overview of the natural language problems addressed in the Traffic Information Collator/Condenser (TICC) project, and describes in some detail the "interesting-corner parser" used in the TICC's Natural Language Summariser. The TICC is designed to take free text input describing local traffic incidents, and automatically output local traffic information broadcasts for motorists in appropriate geographical areas. The "interesting-corner parser" uses both syntactic and semantic information, represented as features in a unification-based grammar, to guide its bi-directional search for significant phrasal groups.

1. INTRODUCTION

The overall goal of the TICC project is to show the potential benefits of automatically broadcasting local traffic information. Our target system, dealing with traffic incidents in the Sussex area, is to be completed by September 1989. The project forms part of the Alvey Mobile Information Systems large-scale Demonstrator.

The Natural Language Summariser component of this system is being developed at Sussex University. Its function is to accept a series of free text messages describing traffic incidents, and to extract from these messages any information that might be relevant for broadcast to other motorists.

The Natural Language Summariser is designed to work in a restricted domain, and only needs to solve a subset of the problems of text understanding. The TICC's output messages are short and very simple assemblies of canned text, posing no significant natural language generation problems. Our main concern is that the messages should be useful to motorists, i.e that they be reliable indications of the state of the roads at the time they are broadcast.

Programs such as METEO [Chevalier et al. 1978] have demonstrated that in a restricted domain with a restricted sub-language, automatic information broadcasts can be useful. Programs such as FRUMP [De Jong 1979, De Jong 1982] have also demonstrated that expectation-driven analysers can often successfully capture the gist of free text. However, the top-down depth-first confirmation of expectations based on sketchy scripts, ignoring most of the input structure, can lead to serious misinterpretations [Riesbeck 82]. Our concern for accuracy of interpretation has led us to a processing strategy in which the Natural Language Summariser analyses the input text at a far greater level of detail than is given in the output messages, so the system "knows more" about the traffic incidents it is describing than it says in its broadcasts. Our parser uses both syntactic and semantic information to guide its search for phrases in the input that might be directly or indirectly relevant to motorists, and explores alternative possible interpretations bottom-up using an active chart [Earley 1970, Kay 1973].

This is an ongoing research project, and we do not claim to have solved all the problems involved in developing a successful system yet. The current paper considers the particular natural language problems we are addressing and describes the "interesting-corner parser" that has been implemented in the prototype system.

2. THE NATURAL LANGUAGE SUMMARISER'S TASK

2.1 INPUT: Our input data comes from the Sussex Police, who have a computer system for storing the text of incoming traffic messages from a variety of sources (eg. patrol cars, emergency services, motoring organisations). An example of the style of this text, derived from real input but with names etc. changed, is given in fig.1.

The series of messages dealing with a

single incident continues over a number of hours, depending on the severity of the incident, and the TICC can afford to spend up to one minute analysing an average length message. All aspects of the police management of the incident are described, and many of the messages are only indirectly relevant to motorists. For example, if one of the vehicles involved in an accident needs a total lift to remove it from the road, the likely delay time given in the broadcast message may be longer, although the need for the total lift will not itself be mentioned in the broadcast. Much of the input is completely uninteresting for the TICC's purposes, such as details of injuries sustained by people involved, or of which police units are dealing with the incident.

There is a great variety in prose style, from the "normal" to the highly telegraphic, but there is a strong tendency towards the abbreviated. It is a non-trivial task to correctly identify the lexical items in the text. Parts of the input string which are not recognised as entries in the Summariser's lexicon (or regular derivations from entries) may be of four types:

i) Names, numbers etc, which may be recognised as such from the context (e.g *pc* humphries *requests ..., ford cortina reg* ABC123).

ii) Other English words not in the lexicon, which cannot reliably be predicted to be proper names (e.g hovis *lorry b/dwn o/s bull's head ph).*

iii) Misspellings of items in the lexicon.

iv) Non-standard abbreviations of known words or phrases.

Abbreviations are not always of "canonical" form, and may be derived from complete words in three different ways, as follows:

i) Single morpheme roots: These usually have more than one possible abbreviated form and never include punctuation eg. *gge, grg* or *gar* for *garage.* But some words do have canonical abbreviations (eg *rd* for *road* and *st* for *street* (or *saint).*

ii) Multi-morpheme roots: These often take only the first letter from the first root morpheme, and then either part or all of the second morpheme. They occasionally include slash punctuation eg. *cway, c/way* for *carriageway, mcycle, m/c* for *motorcycle, o/s* for *outside* (or *offside),* and *ra* for *roundabout.*

iii) Sequences / phrases: Some sequences of words have canonical abbreviations (e.g *bbc* and not *britbrdcrp).* Canonical examples seen in Fig. 1. below include *rta* for *road traffic accident* and *oic* for *officer in charge.*

Non-canonical sequences may have a variety of abbreviations for each of the constituent words, and may or may not have slash or period punctuation, eg. *f/b* for *fire brigade, eamb* or *esamb* for *east (sussex) ambulance, hazchem* for *hazardous chemicals.*

The problem is compounded for the TICC by the fact that the input we receive is all in upper case, hence the even the convention of distinguishing proper name abbreviations by upper case is not applicable. In order to cope with these different types of input string, we need not only a "phrasal lexicon" as advocated by Becker [Becker 1975], but also an "abbreviation lexicon".

Time: 1634 *Location:* scaynes hill, haywards heath

1634	rta serious near top scaynes hill persons trapped rqst esamb f/b 1/2 mile south jw freshfield rd.
1638	fm pc 123 acc inv reqd poss black oic pc 456
1639	fire and amb en route
1642	req total lift for saloon car rota garage
1654	eamb now away from scene
1655	freshfield bodyshop on way
1657	fm pc 456 req rd closed n and s of hill st crnr
1658	req two traff units to assist re closures
1709	can we inform brighton 1234 tell mr fred smith will be late due to this rta
1715	local authority required loose paving stones
1723	fm pc 234 at st george's hosp. dr in charge having examined mr jones now feels this is not likely to be a black. driver of lorry has arrived, will probably be released after treatment for cuts. car 45 will be free from hosp in about 20 min

Fig. 1. An extract from an example (fictitious) incident log.

Our aim is to have a unified process for identifying idiomatic or fixed phrases and abbreviated sequences as in iii) above, so that for example *as soon as poss*, *asap* and *a.s.a.p.* are all identified as the same "lexical item". Work on this is, however, at a preliminary stage, and we have not yet found any general solution to the problem.

2.2 SUMMARISATION: Deriving a short broadcast for motorists from a long series of messages such as that in fig. 1 requires two main phases. First, the Natural Language Summariser must build up a picture of what is happening at the scene of the incident. Second, a Tactical Inferencer must decide what motorists should be told regarding the incident.

The Natural Language Summarising process also requires two phases. In the first phase a Message Analyser extracts interesting information from a single message. In the second phase an Event Recogniser puts together the information from a series of messages to build up a description of the incident as a whole, or rather those aspects of the incident relevant to other motorists (see fig 2. below).

The Message Analyser does not build a complete representation of the syntax and semantics of messages such as those at 1709 and 1723 in fig. 1 above, since they have no bearing on the progress of the traffic incident as far as other motorists are concerned. It just searches for phrases describing "interesting" events. These fall into two classes:

Primary Events: Such as vehicles blocking the road, substances spilling onto the road, all or part of the road being closed, diversions being put into operation, garage coming to remove vehicles from the road, services like fire brigade and county council removing hazards, etc.

The input messages rarely describe these events in full, so the Event Recogniser must infer, for example, that if the local council has been called out to remove debris from the road, that at some time earlier debris must have fallen on the road.

Secondary Events: These include requests that some of the primary events should happen, and people being informed that primary events have happened are happening or will happen.

We will not have any model of the beliefs of the various agents involved in incident handling. As far as the TICC Natural Language Summariser is concerned, the meaning of someone being informed that a primary event has happened is equivalent to the statement that it has happened. But the Tactical Inferencer will use its model of the typical progress of traffic incidents to predict the significance of the primary events for other motorists. For example, if a vehicle is stated to need a front suspended tow, then the Tactical Inferencer will predict that a certain amount of time will elapse before the vehicle is towed away.

2.3 OUTPUT: Not every message input to the system will produce an update to the Event Recogniser's description of the incident, because the Message Analyser may fail to find a description of an interesting event. But even when the Event Recogniser passes a description of a traffic incident to the Tactical Inferencer, this will not necessarily result in a broadcast. For example, the Event Recogniser may recognise a series of messages as describing a traffic light failure incident. The Tactical Inferencer may decide to broadcast a message about this incident if it has occurred on a busy main road in the rush hour, but not if it has occurred late at night in a small village.

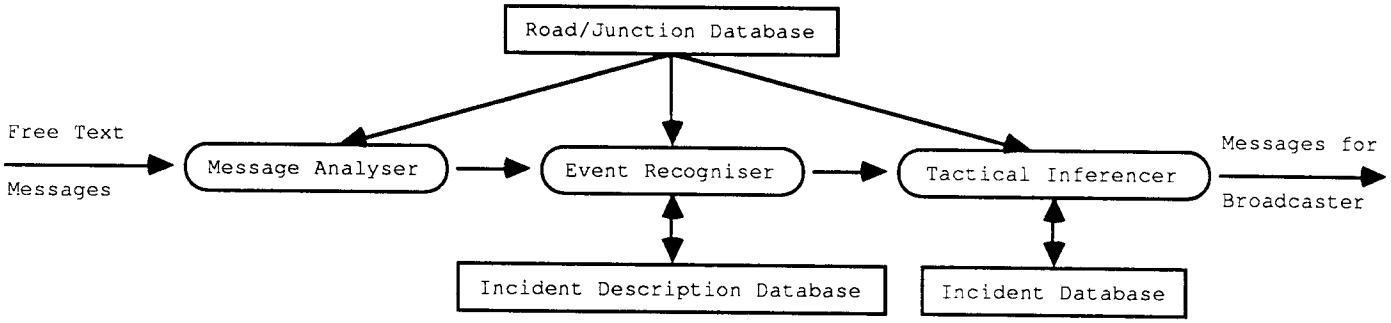

Fig. 2. Part of the TICC system, showing Message Analyser, Event Recogniser, and Tactical Inferencer.

The domain knowledge used in the the Tactical Inferencer is non-linguistic, and concerns inferences about the likely time delays for different types of incident, the geographical areas likely to be affected by a given incident, etc. The Transport and Road Research Laboratory, part of the Department of Transport, are assisting us in the development of rules for this part of the system.

There are other components of the TICC system which we do not detail in this paper, such as the graphical interface, via a map of the Sussex area, to a database of of current traffic incidents. Although the TICC is designed to send its messages to a dedicated broadcasting system, the actual broadcasting aspect of the project is the responsibility of RACAL research, one of our other Alvey collaborators. In our current prototype system, implemented on a Sun-3 workstation, broadcasts to local geographical areas in Sussex are simulated, and the Tactical Inferencer is extremely simple.

3 . INTERESTING CORNER PARSING

The parser that has been implemented for the Message Analyser searches bidirectionally for all syntactic parses associated with semantically interesting parts of the input. Before describing the search strategy in more detail, we need to clarify what a syntactic parse looks like in our grammar formalism, and how we specify what is semantically interesting.

3.1 THE GRAMMAR FORMALISM: We use a unification-based grammar formalism, with rules that look similar to context-free phrase-structure rules. Both immediate dominance and constituent ordering information are specified by the same rule, rather than by separate statements as in FUG [Kay 1985], LFG [Kaplan & Bresnan 1982] and GPSG [Gazdar et al 1985]. Feature-passing between categories in rules is done explicitly with logical variables, rather than by conventions such as the HFC and FFP in GPSG [Gazdar et al 1985]. Thus the rule format is most similar to that used in DCG's [Pereira & Warren 1980]. Categories in rules are feature/value trees, and at each level the value of a feature may itself be another feature/value tree. Feature values may be given logical names, and occurrences of feature values having the same logical name in a rule must unify.

The feature trees which constitute categories in our grammar may specify both syntactic and semantic features, so that we can write "syntactic" rules which also identify the semantic types of their constituents. For example, if we use the feature *sf* on categories to specify a tree of semantic features for that category, then the rule:

```
(1)  vp=(sf:VSF) --> v=(sf=(patient:P):VSF),
                      np=(sf:P)
```

says that a verb phrase may consist of a verb followed by a noun phrase, and that the semantic features on the noun phrase (labelled *P)* must unify with the semantic features specified as the value of the *patient* sub-feature of the verb's semantic features, and additionally that the semantic features on the whole verb phrase (labelled *VSF)* must unify with the (complete tree of) semantic features on the verb.

By adding domain-specific semantic feature information to lexical categories, we gain the power of domain-specific semantic grammars, which have been shown to be successful for handling ill-formed input in limited domains [Burton 1976]. But because we use unification by extension as the basic criterion for node admissability when we test for rules to licence local trees, we can also capture generalisations about syntactic categories that are not domain-specific. So for example if we had a verb-phrase rule such as (2) and a lexical entry as in (3):

```
(2)  vp --> v=(tr=trans), np

(3)  close  v=(tr=(trans),
                sf=(event_type=road_closure,
                    agent=service,
                    patient=roadlocation))
```

then the verb feature tree specified in (2) would unify with the verb feature tree in (3). Hence *close* can be treated both as a domain specific verb and as an instance of the general class of transitive verbs.

Using a feature-based semantic grammar therefore gives us a compact representation of both domain independent and domain-specific information in a single uniform formalism. Syntactic generalisations are captured by rules such as (2), and domain-specific subcategorisation information is expressed in feature-trees as in (3), which states that *close* has the semantic features of a road-closure event, expecting an agent with the semantic features of a service (eg police) and a patient with semantic features indicating (a part of) a road. As with all sub-languages, our

lexicon also includes domain-specific meanings for particular lexical items, eg. *black* meaning *fatal* (cp messages at 1638 and 1723 in fig. 1 above).

3.2 A GRAMMAR FOR THE TICC DOMAIN: Writing a grammar to give adequate coverage of the input that our system must handle is a lengthy task, which will continue over the next two years. However, analysis of a corpus of data from police logs of over one hundred incidents in the Sussex area, and trials with experimental grammars, have led us to adopt a style of grammar which we expect will remain constant as the grammar expands.

We do not attempt to map telegraphic forms onto "fully grammatical" English forms by some variant of constraint relaxation [Kwasny & Sondheimer 1981]. We simply have a grammar with fewer constraints. This is because it is not always easy to decide what is missing from an elliptical sentence, or which constraints should be relaxed. Consider for example the message at 1655 from fig. 1, repeated here:

(4) freshfield bodyshop on way

It is not at all clear what the "full" sentential form of this message ought to be, since it might also have been phrased as one of:

(5.1) freshfield bodyshop is on the way
(5.2) freshfield bodyshop is on its way
(5.3) freshfield bodyshop are on the way
(5.4) freshfield bodyshop are on their way

Each of the (5.1)-(5.4) must be allowed to be grammatical (and each might occur in our type of input), since noun phrases naming corporate entities can regularly be regarded as singular or plural (cp. *Ford Motors has announced massive profits ... vs. Ford Motors have announced massive profits).* But in each case the semantic representation that the Message Analyser must build only needs to represent the fact that the garage called freshfield bodyshop are going somewhere (which the Event Recogniser will expect to be the scene of the incident, in order to remove the damaged vehicle). Since the distinctions between the syntactic forms in these examples is irrelevant for our purposes, it would be a waste of the parser's effort to introduce search and inference problems in the attempt to map the syntax of (4) uniquely into the syntax of one or other of the forms in (5). Indeed it is more appropriate for our purposes to regard *on way* as a domain-specific idiomatic phrase, equivalent to *en route, enrte* etc (each of which occur in similar contexts).

In keeping with this approach to ill-formedness, our grammar contains many categories (ie feature-trees), that would not be recognised as syntactic categories in grammars for normal English, eg. we have special rules for phrases containing predicted unknowns such as names, car registration numbers, etc. Our parser is looking for phrases describing events rather than sentences, and we will not necessarily always assign a structure with a single "S" label spanning all the input message.

As we noted in 3.1 above, the lexical entries for words that suggest interesting events include trees of semantic features that specify expected fillers for various roles in these events. These feature trees provide selectional restrictions useful for guiding the parse, but do not themselves constitute the "semantics" of the lexical entries. The semantics are represented as first-order logical expressions in a separate field of the lexical entry, and representations of the meaning of phrases are built using semantic rules associated with each syntactic rule, as phrases are completed in the bottom-up parse.

3.3 THE SEARCH STRATEGY: Interesting-corner parsing is basically an adaptation of bottom-up chart parsing to allow island-driving through the input string, whilst still parsing each individual rule unidirectionally. This gives a maximally efficient parse for our goal of reliably extracting from the input all and only the information that is relevant to other motorists. This form of expectation-driven parsing differs from that used in earlier script-based systems such as MARGIE [Schank 1975], ELI [Riesbeck 1978] and FRUMP in four ways:

First, the interesting-corner parser uses an active chart to consider bottom-up all interesting interpretations that might be given to an input message, rather than proceeding left to right and filtering out later (right) candidate interpretations on the basis of earlier (left) context.

Second, if there are no interesting lexical items in the input string, or if the only interesting items occur at the (right) end of the input, there is no attempt to match all the leftmost items to a series of candidate scripts or frames using top-down expectations.

Third, the expectations themselves are expressed declaratively in feature trees that form part of the lexical categories, which control the search via standard unification with declarative rules, where previous systems used procedural "requests" in the lexicon.

Fourth, our parser builds an explicit syntactic tree for the input, albeit including semantic features, rather than by building a semantic representation "directly".

The interesting-corner parser checks the semantic features on every lexical item in the input to see if they are interesting, but this is a far faster operation than testing many times whether a series of lexical items matches the expectations from a top-down script. This does assume that the parser can identify what the lexical items are, which is problematic as we noted in section 2.1 above. But as we shall see, the interesting-corner parser does use predictions about the presence of lexical items with particular features in its search, and hence is in no worse a position than a strictly top-down parser as regards matching expectations to ill-formed lexical items.

3.3.1 UNIDIRECTIONAL ISLAND-DRIVING: Island-driving is useful for text where one needs to start from clearly identifiable (and in our case, semantically interesting) parts of the input and extend the analysis from there to include other parts. But parsing rules bi-directionally is inherently inefficient. Consider, for example, a chart parse of the input string *a b* given a single rule:

c -> a b.

A standard bottom-up left-to-right active chart parse of this input would create three nodes (1 *a* 2 *b* 3) two active edges (an empty one at node 1 and one from nodes 1 to 2) and one inactive edge (from node 1 to 3).

But a bi-directional parse, allowing the rule to be indexed at any point, would build a total of 7 active edges (one empty one at each node, and 2 pairs with one constituent found, built in different directions, ie 5 distinct edges). It would also build the same inactive edge in two different directions. For a rule with three daughters, a bidirectional parse produces 14 active edges (9 of which are distinct) and again 2 inactive edges.

This redundancy in structure-building can be removed by incorporating constituents into rules unidirectionally whilst still parsing the text bidirectionally. We do this by indexing each rule on either left-most or right-most daughter, and parsing in a unique direction away from the indexed daughter. In order to preserve completeness in the search, the chart must contain lists of active and inactive edges for each direction of expansion, although the same structure can be shared in the inactive edge-lists for both directions. The fundamental rule of edge-combination must be augmented so that when an inactive edge is added to the chart, it combines with any appropriate active edges at both of its ends. This process might be called "indexed-corner parsing", in that it effectively combines left-corner parsing and right-corner parsing, and the direction of parse at any stage simply depends upon how the individual grammar rules are indexed.

The interesting-corner parser implements an indexed-corner chart parser, with the addition of an agenda control mechanism and an indexing principle for grammar rules.

3.3.2 AGENDA CONTROL: The insertion of edges into the agenda is constrained by the value of a "control-feature", which specifies where to look in the feature-trees that constitute our categories in order to find the semantically "interesting" features. In our examples (1) and (2) above, this control-feature is named *sf*. When a normal bottom-up chart parse begins, all lexical items are tested to see whether they can spawn higher edges. But in the interesting-corner parse, higher edges are only spawned from lexical items that have a control-feature specification which unifies with a pre-defined initial value of the control feature. Thus by assigning (sf=event_type) to be the initial value of the control feature, we ensure that only those edges are entered into the agenda that have semantic feature trees that are extensions of this tree (eg the semantic feature tree for *close* in (3) above). This effectively means that parsing must begin from words that suggest some kind of interesting event. Note that the initial active edges may be proposed from any point in the input string, and their direction of expansion from that point is determined by the indexing on the rules.

For all active edges proposed from lexical items that were initially recognised to be interesting, the parser checks the list of edges sought for "interesting" categories (ie. those with values for the control-feature *sf*). If

there are any, it searches, in the direction of expansion for the current active edge, for any lexical items that have a semantic feature-tree which unifies with the new specification of what is "interesting".

For example, if the rule given in (1) above is indexed on the left daughter, and an active edge is proposed starting from an inactive edge representing the lexical item *close* defined as in (3) above, then via the logical name P the features on the noun-phrase being sought become instantiated to (sf=roadlocation). The parser then looks rightwards in the input string for any lexical items having semantic feature trees that are extensions of this new tree. If it finds any, it predicts more active edges from there, and so forth.

Fig. 3 below illustrates the numerical order in which the interesting-corner parser incorporates nodes into the parse tree for a very simple "sentence" (in our grammar we allow sentences with deleted auxiliaries), but with the details of the feature trees omitted for legibility.

Extension unification allows one of the structures to be unified (the target) to be an extension of the other (the pattern), but not vice-versa. This means that it is more restricted than graph unification, and hence can be implemented more efficiently. It is less restricted than term unification, and hence less efficient at parse-time, but it does allow the grammar and lexicon to be far more compact than they would be with term-unification in the absence of a grammar pre-processor. However, using extension unification as the basic operation does also mean that that the unification of logical variables in rules is not order-independent, and hence we need an indexing principle to determine the direction in which particular rules should be parsed.

3.3.3 THE INDEXING PRINCIPLE: Our general principle for indexing rules is that we must parse from categories that specify general information (ie. that have small feature-trees) to those that specify particular modifications of that general information (ie. that provide extensions to the smaller trees by unification). This usually means that we parse from syntactic heads to complements, eg indexing sentences on the vp (cf. HPSG [Proudian & Pollard 1985]).

In our example rule (1), we index on the verb, because its expectations specify the general semantic type of the object, and the semantic feature tree of the noun-phrase will specify a sub-type of this general type, and therefore will be an extension of the verb's patient semantic feature tree. In the example shown in fig 3, the semantic tree of the np built at node 4 is:

```
(sf=(roadlocation=(name=huntingdon,
                   rtitle=lane)))
```

which unifies by extension with the feature tree (sf=roadlocation)), and this as we saw above became the expected semantic tree for the noun-phrase when rule (1) unified with the verb in (3).

Finally, rules for categories that have expected unknowns as daughters are always indexed on the known categories, even if these are not the grammatical head (eg we index on the policeman's title for rules handling *sgt smith, insp brown* etc. and on the known title of a road for cases like *huntingdon lane, markworthy avenue etc.*

3.3.4 EXTENSIONS TO THE CURRENT SYSTEM: There are many aspects of the TICC's Natural Language Summarisation not dealt with in this paper, such as the semantic rules used in the Message Analyser and

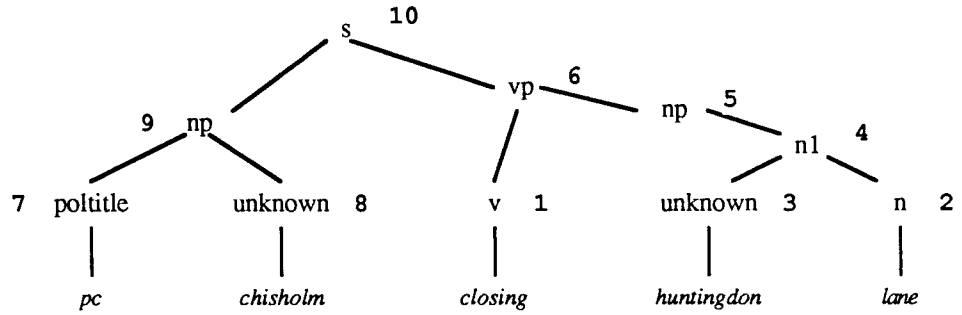

Fig. 3. Showing the order in which the interesting-corner parser constructs a parse tree, starting with the most interesting words.

the Event Recogniser. There are also many inadequacies in the current implementation of the Message Analyser, eg in its handling of abbreviations/phrases, and in the handling of input that is "ill-formed" even with respect to our relatively unconstrained grammar.

However, work is currently in progress on these problems, and we believe that the basic mechanisms of interesting-corner parsing are sufficiently powerful to enable us to achieve a practical solution, whilst being sufficiently general to ensure that such a solution will be theoretically interesting.

4 . CONCLUSION

The automatic production of traffic broadcasts, given the type of free text we have described in this paper, poses many difficult problems. In many ways our overall approach to these problems follows in a long tradition of semantically driven systems, but the processing style of our Message Analyser is much closer to that used in contemporary syntax-driven systems. We make explicit use of rules in a unification-based grammatical formalism that express both semantic and syntactic information declaratively, and our interesting-corner parser provides a search of the input messages that is both thorough and efficient.

We believe that complete understanding of free text messages is well beyond the state of the art in computational linguistics, but that we can nevertheless develop the TICC's Natural Language Summariser to have sufficient partial understanding to be practically useful.

REFERENCES

Becker, J.D. (1975) "The Phrasal Lexicon", in R. C. Schank and B. L. Nash-Webber (eds.), *Proceedings of the Workshop on Theoretical Issues in Natural Language Processing*. Cambridge, Mass., Bolt, Beranek and Newman, pp. 70-73.

Burton, R. (1976) "Semantic Grammar: an Engineering Technique for Constructing Natural Language Understanding Systems", Technical Report 3453, Cambridge, Mass., Bolt, Beranek and Newman.

Chevalier, M., Dansereau, J., and Poulin, G. (1978) "TAUM-METEO: Description Du Systeme.", Montreal, Groupe TAUM, Universite de Montreal.

DeJong, G.F. (1979) "Skimming Stories in Real Time", Doctoral Thesis, New Haven, Yale University.

DeJong, G.F. (1982) "An Overview of the FRUMP System", in Wendy G. Lehnert and Martin H. Ringle (eds.), *Strategies for Natural Language Processing*. Hillsdale, Erlbaum, pp. 149-176.

Earley, J. (1970) "An Efficient Context-free Parsing Algorithm", *Communications of the ACM*. vol. 6, no. 8, pp. 451-455.

Gazdar, G., Klein, E., Pullum, G., and Sag, I. (1985) *Generalised Phrase Structure Grammar*. Oxford, Blackwell.

Kaplan, R., and Bresnan, J. (1982) "Lexical Functional Grammar: a Formal System for Grammatical Representation", in Joan Bresnan (ed.), *The Mental Representation of Grammatical Relations*. Cambridge MA, MIT Press, pp. 173-281.

Kay, M. (1973) "The MIND System", in Randall Rustin (ed.), *Natural Language Processing*. New York, Algorithmics Press, pp. 155-188.

Kay, M. (1985) "Parsing in Functional Unification Grammar", in David R. Dowty, Lauri Karttunen and Arnold M. Zwicky (eds.), *Natural Language Parsing*. Cambridge, Cambridge University Press, pp. 251-278.

Kwasny, S.C., and Sondheimer, N.K. (1981) "Relaxation Theories for Parsing Ill-formed Input", *American Journal of Computational Linguistics*. vol. 7, no. 2, pp. 99-108.

Pereira, F.C.N., and Warren, D.H.D. (1980) "Definite Clause Grammars for Language Analysis - a Survey of the Formalism and a Comparison with Augmented Transition Networks", *Artificial Intelligence*. vol. 13, no. 3, pp. 231-278.

Proudian, D., and Pollard, C.J. (1985) "Parsing Head-driven Phrase Structure Grammar", *ACL Proceedings, 23rd Annual Meeting*. pp. 167-171.

Riesbeck, C.K. (1978) "An Expectation-driven Production System for Natural Language Understanding", in Donald A. Waterman and Rick Hayes-Roth (eds.), *Pattern-directed Inference Systems*. New York, Academic Press, pp. 399-414.

Riesbeck, C.K. (1982) "Realistic Language Comprehension", in Wendy G. Lehnert and Martin H. Ringle (eds.), *Strategies for Natural Language Processing*. Hillsdale, Erlbaum, pp. 37-54.

Schank, R.C. (1975) *Conceptual Information Processing*. Amsterdam, North-Holland.