MORPHOLOGICAL PROCESSING IN THE NABU SYSTEM

Jonathan Slocum
Microelectronics and Computer
Technology Corporation (MCC)
3500 West Balcones Center Drive
Austin, Texas 78759

ABSTRACT

The Nabu morphological processor is designed to perform a number of different functions, of which five have so far been identified: analysis, guessing (about unknown words), synthesis, defaulting (proposing the most likely inflectional paradigm for a new base form), and coding (producing all possible inflectional paradigm variants for a new base form). Complete or very substantial analyzers have been produced for a number of diverse languages; other functions have been implemented as well. This paper discusses our design philosophy, as well as our technique and its implementation.

INTRODUCTION

Nabu is a multilingual Natural Language Processing system under development in the Human Interface Laboratory at MCC, for shareholder companies. Its morphological component is designed to perform a number of different functions. This has been used to produce a complete analyzer for Arabic; very substantial analyzers for English, French, German, and Spanish; and small collections of rules for Russian and Japanese. In addition, other functions have been implemented for several of these languages.

In this paper we discuss our philosophy, which constrained our design decisions; elaborate some specific functions a morphological component should support; survey some competing approaches; describe our technique, which provides the necessary functionality while meeting the other design constraints; and support our approach by characterizing our success in developing/testing processors for various combinations of language and function.

DESIGN PHILOSOPHY

Before we set about designing our morphological processor, we elaborated our philosophical commitments regarding an NLP system. These include: (1) multilingual application, (2) fault-tolerant, fail-soft behavior, (3) rule reversibility, (4) disparate functionality, (5) inherent parallelism, (6) grammatical clarity, and (7) rigorous testing.

MULTILINGUAL APPLICATION

The algorithms and their software instantiation must admit application to any human language likely to confront our system; these include the languages of major demographic and/or economic significance (and their relatives). Our selected representatives are English, French, German, Spanish, Russian, Arabic, Chinese, and Japanese.

FAULT-TOLERANT, FAIL-SOFT BEHAVIOR

Real-world NLP applications, whether for text or interactive dialog, will be confronted with numerous errors of various types. As far as users are concerned, guaranteed failure in the presence of any error is intolerable: a system must overcome simple mistakes without discernable problems. For example, insignificant typing and/or spelling mistakes should be ignored, as should minor morphological blunders. Users do not like to be asked for corrections of (seemingly) simple mistakes, and of course printed texts cannot be queried in any case. In the presence of more serious problems, performance should degrade only gradually. This is nothing more than a commitment to *understanding* the utterance, rather than *punishing* the user for errors in it. We contend that human-like fault-tolerant, fail-soft behavior must be incorporated in the fundamental design of a system: it cannot be tacked-on after system development. Creating an applied system for a "perfect" natural language that is hypothesized, but never observed, is misguided, aside from being wasteful.

RULE REVERSIBILITY

To the extent feasible and reasonable, the linguistic rules in an NLP system should be reversible -- useful for both analysis and synthesis. But it is not enough to have one, completely reversible grammar performing two functions. Indeed, reversibility may not be always desirable: in keeping with the commitment to fault-tolerant, fail-soft behavior, an analyzer should over-generate (accepting some incorrect

forms as input), while a synthesizer should never produce them as output. Since these two functions are, therefore, rather different processes, one must search for a means to distinguish the rules (linguistic descriptions) from the strategies (linguistic processes, called grammars) controlling their application: the former can be reversible, while the latter might not be.

DISPARATE FUNCTIONALITY

Analysis and synthesis constitute two obvious linguistic processes (grammars imposed upon rule sets). There are, however, even more processes than these of interest in a practical setting: that is, there may be a number of grammars built from a single set of linguistic rules, as we demonstrate below. Thus, a processor must admit the simultaneous instantiation of a number of grammars in order to be called general.

INHERENT PARALLELISM

Acceptable runtime performance in any significant real-world NLP setting is now understood to require implementation on parallel machines. Thus, grammars must be inherently suited to parallel execution, and such opportunities must somehow be representable in the formalism in which the grammars are expressed.

GRAMMATICAL CLARITY

In any real-world application, the number of linguistic rules and the complexities of grammars imposed upon them is considerable. Successful implementation and maintenance thus requires that the grammars be clearly and concisely stated, however powerful they may be. Not only must the rule formalism be relatively clean and simple, but also a grammar must be viewable at various levels of detail. Variable granularity in the presentation enhances the opportunity for comprehensibility.

RIGOROUS TESTING

In order to become practical, a system must admit -- and have undergone -- rigorous testing. It is not enough to develop a micro-implementation, then claim that the system can be scaled up to become a real application (presumably to be tested on end-users). More than just a philosophical commitment to the idea of testing, this requires that the system actually be fast enough *during the development phase* that thorough testing (of grammars, etc.) can take place at that time. If its speed is glacial during the development phase, a complex system cannot be completed, and its practicality will never be shown.

MORPHOLOGICAL PROCESSES

We have, so far, identified five interesting morphological processes: analysis, guessing, synthesis, defaulting, and coding. The first two concern comprehension; the other three, production.

ANALYSIS

Morphological analysis is a relatively well-understood notion -- which is not to say that there is agreement concerning what the result should be, or how it should be produced. But, generally speaking, analysis is agreed to involve the decomposition of a surface-form string (usually in English called a *word*) into its constituent base-form morphemes and their functional attachments; this may be finer-grained, as when each morpheme is associated with lexical or other linguistic information, and indeed the process is usually understood to imply access to a stored lexicon of morphemes, in order to correctly identify those contained in the string. Analysis is assumed to perform this decomposition according to a set of language-specific strategies (i.e., a grammar) limiting the possible decompositions.

GUESSING

In keeping with a commitment to fault-tolerant, fail-soft behavior, a system must, e.g., deal with unknown words in an uttterance by making plausible guesses regarding their morphological, lexical, syntactic, and even semantic properties. A morphological **guessing grammar**, presumably operating after the analysis grammar has failed, must embody heuristic strategies, and these may well differ from those of the analyzer, even though the rule stock upon which they draw might be identical. For example, while the analyzer must, sooner or later, entertain all possible decomposition hypotheses, the guesser might best be constrained to produce only the "most likely/plausible" hypotheses.

SYNTHESIS

Synthesis, like analysis, is a relatively well-understood notion, albeit characterized by debate concerning the details. Generally speaking, synthesis is the composition of a surface-form string encoding the information contained in one or more base-form morphemes having known functional attachments. Synthesis, like analysis, is assumed to perform this composition as dictated by a grammar. Note again that, in practice, this grammar cannot be the simple inverse of the one controlling analysis: a synthesizer should be prohibited from making mistakes that are tolerated (if, indeed, even noticed) by an analyzer.

DEFAULTING

Guessing is relevant to end-users, dealing as it does with unknown words in an input utterance. Developers, on the other hand, faced with teaching the system [how to synthesize only] the correct surface forms of words being defined, can make use of additional functions, such as a **defaulting grammar**. Given a root or stem form and part of speech, a lexical defaulter can propose the most likely inflectional paradigm for a word. This is better than requiring the lexicographer to type in the variants, or manually encode the paradigm in some other fashion: greater human reliability is experienced when validating good guesses and correcting a few wrong ones than when entering everything from scratch.

CODING

When the lexical defaulter guesses incorrectly, a **coding grammar** could render all potential inflectional paradigms (as permitted by the language), from which the lexicographer could select the correct one(s) for the newly-defined word. This is desirable, because greater human reliability is experienced in selecting from among possible inflections than in producing all and only the correct variants.

SURVEY

One of the more popular frameworks for morphological processing is the two-level model developed by Koskenniemi [1983], modified and extended by Karttunen [1983]. Two-level models are fully reversible, performing both analysis and synthesis, and correspond to finite-state machines, hence they appear to enjoy some computational advantages. However, there are both theoretical and practical problems. It appears that the model is not sufficiently powerful to account for some human languages (e.g., Icelandic, which exhibits recursive vowel harmony). The model can be decidedly inelegant in some respects (e.g., in handling alternations such as the English *nominate/nominee* by positing a "lexical entry" *nomin*). There is a very substantial time penalty involved in compiling two-level grammars (it may be measured in hours), which impedes rapid debugging and testing. Finally, the "advantages" of reversibility are arguable, for reasons discussed above and below.

Affix-stripping models are commonly employed, especially for English [Slocum, 1981]. The IBM system [Byrd, 1983] also uses affix rules in a strictly word-based model of morphology, but the rules are unordered and thus only marginally may be said to constitute a grammar; certainly, morphosyntactic behavior is not highlighted. These two systems were developed for English analysis only; they are not reversible in any sense, and have not been tested on other languages. A serendipitous situation exists, in that they have a certain degree of fault-tolerance built in, though this was not a goal of the design/implementation process.

In order to elucidate morphosyntax, some approaches use a simple (e.g., two-level, or similar) model to account for orthographic behavior, and a "grammar of words" to analyze morpheme sequences syntactically (e.g., [Bear, 1986], [Russell et al., 1986]). It does not seem that these approaches, as described, lend themselves to any sort of reversal, or other form of rule-sharing; furthermore, only analysis is mentioned as an application.

Even simpler models have employed allomorphic segmentation; morphosyntax may be approximated by a longest-match heuristic [Pounder and Kommenda, 1986] or defined by a finite-state grammar [Bennett and Slocum, 1985]. The required entry of every allomorphic variant of every word is both a theoretical and a practical disadvantage, but runtime processing is speeded up as a positive consequence due to total neglect of spelling changes. Fault-tolerant behavior is not built-in, but can be almost trivially added (whereas, in many other models, it would be difficult to incorporate). The systems cited here are used for analysis only.

No previous models of morphology seem to have been used for anything but analysis, and occasionally synthesis; the other three functions mentioned above, and others that might exist, are neglected. Although the author has discussed other functionality in earlier work [Slocum and Bennett, 1982], even there the morphological processors used for analysis, synthesis, and defaulting/coding were distinct, being implemented by entirely different software modules, and shared only the dictionary entries.

THE NABU TECHNIQUE

In Nabu, rules are separate from the strategies imposed upon them. Rules may be thought of as declarative in nature; they are organized in an inheritance hierarchy. The "grammars of words" imposed upon them, however, may be thought of as procedures -- actually, dataflow networks.

RULE HIERARCHY

The structure of the rule hierarchy is determined by linguists, purely for their own convenience, and implies no runtime behavior of any kind. That is, the rule hierarchy is purely static and declarative in nature. The one constraint is that, at the top level, collections of rules are distinguished by the language they belong to -- i.e., the first division is by language. Typically, though not necessarily, the second-

level division imposed by the lingusts is that of category: the part-of-speech of the surface-forms for which the rule subset is relevant. For languages that distinguish inflection from derivation, our linguists have generally found it convenient to divide rules at the third level in terms of these two classes. Other than the top-level language division, however, the structure of the hierarchy is entirely at the discretion of the responsible linguists. Consequently, we have observed that different linguists -- each responsible for the morphological rules (and grammars) of entire languages -- prefer and employ different organizational principles.

Rules may also be thought of as declarative in nature -- though, in fact, for the purposes of maintenance they embody certain procedural behaviors such as a self-test facility. A morphological rule is an equation between one letter-string+feature-set, which we call a **glosseme**, and another. One side of the equation describes what might be called the "surface" side of a glosseme, as it represents an encoding of information nearer to (but not necessarily at!) the surface-string level: all this really means is that relatively more information is expressed in the *string* part than in the *feature* part. The other side, in turn, describes what might be called the "base" glosseme, as it represents an encoding of information closer to (but not necessarily at!) the base-form level, with relatively *less* information expressed in the string part than in the feature part. It is important to note that *the information content of the two sides is the same* -- only the *form* of the information changes. This is why we classify a rule as an **equation**, and this also admits rule reversibility in the implementation.

As a trivial example of such an equation, consider the English inflectional rule

["+s" ()] = ["+" (NOUN PL)].

The "surface" side, on the left, describes a glosseme string whose last character is the letter *s* [by means of the pattern "+s"] and places no constraints on the glosseme's feature set [by means of the empty list ()]. The "base" side, on the right, describes an equivalent glosseme whose string lacks the final letter *s* [by means of the pattern "+"] and constrains the feature set [by means of the two features (NOUN PL)]. Executed in one direction, this rule removes an *s* and adds the two features (NOUN PL); reversed, the rule removes the two features (NOUN PL) and adds the morpheme *s*. Obviously, this English rule conveys the notion that a NOUN may be inflected for PLural by means of the [bound] morpheme *s* at its right end.

The plus sign (+) in a pattern is used to indicate whether prefixation or suffixation of the glosseme string is being described, depending on whether the pattern precedes or follows it; or a pattern may describe an entire glosseme string via omission of the sign. In a pattern, alphabetic case is important: a lower-case letter signifies a constant, and must be matched by the same letter (in any case) in the glosseme; an upper-case letter signifies a restricted *variable*, and must be matched by some letter from the set over which it is defined. Thus, for example, in English rules we use the letter *V* to stand for Vowel; *C*, for Consonant; and *G*, for Gemminating consonant. (Of course, the variable restrictions are entirely arbitrary as far as Nabu is concerned. Each linguist defines sets of variables and their match restrictions according to taste and the language at hand.)

If the same variable appears more than once in a pattern, it is required to match the same letter in the glosseme: the equation

["+GGing" ()] = ["+G" (VERB PRPL)]

thus describes doubling of the last consonant in an English verb, before suffixation of the present participial morpheme. Another facility is required for convenience in describing alternation. In German, for example, certain morphological operations involve the umlauting of vowels; thus, an unmarked vowel on the "base" side may require replacement by an umlauted vowel on the "surface" side. If only one vowel behaved this way, this would be no problem: the corresponding letters would simply appear in their places in the patterns. But there are three vowels that behave like this in German (*a*, *o*, and *u*) in the identical context. In order to eliminate the need for tripling the size of the rule base otherwise required to describe this phenomenon, we provide the linguists with a means for pairing-up *variables*, so that a character matching one may be replaced by the corresponding character in the other's set. Many other languages exhibit this kind of alternation, making this a useful technique.

A character in a pattern string matches one and only one character in a glosseme string. Generally speaking, the characters appearing in a pattern string are those of the language being described, as one would expect. Some languages, however, lack a case distinction -- Arabic, for example -- rendering the variable notation problematic. In this situation, upper-case letters from the Roman alphabet are used to represent variables in rules.

Given this framework, creating a bidirectional rule interpreter is relatively straightfor-

ward. Rule execution is a matter of matching according to one side of the equation and (if that is successful) transforming according to the other side. So long as every rule is truly an equation (and only a human linguist can decide this, except in trivial cases) then every rule is reversible -- that is, can be used for comprehension as well as production, because the interpreter can transform a rule's output back into the original input. In Nabu, as noted earler, there are currently two comprehension processes (analysis and guessing) and three production processes (synthesis, defaulting, and coding). But neither collections of rules nor their hierarchical structure describes such functionality.

CONTROL GRAPHS

A morphological grammar is an execution strategy imposed upon a body of morphological rules. Bearing in mind that morphological rules can apply to the outputs of other rules (consider the derivational process in English, as for example when the word *derivational* is constructed from *derive* + *ation* + *al*), and that such compounding is not free, but linguistically constrained, it is obvious that the compounding constraints -- as well as the individual rules themselves -- must be accounted for. In Nabu, an execution strategy is represented as a dataflow network, which we loosely term a **control graph**.

A control graph is a collection of nodes connected by directed arcs. Each node is composed of a bundle of morphological rules, which may be applied when input reaches that node, and whose output may be passed across arcs to other nodes. There will be one or more designated **start nodes**, where input appears and processing begins (conceptually, in parallel, if there are multiple start nodes). From each start node, there will be a path to one or more designated **terminal nodes**, where processing ends and the graph's output is emitted. The path may be of arbitrary length; start nodes may themselves be terminal nodes. In analysis, encountering a terminal node entails dictionary look-up; in synthesis, output of a surface-form.

There are two types of arcs, **Success** and **Failure**; the former are arcs across which successfully-applied rules will pass their output, and the latter are arcs across which the original input to a node is passed, should no rule inside it be successful. A successful rule is one whose input side ("surface" or "base," depending on whether the graph is engaged in comprehension or production) matches the input glosseme, and whose output side represents the same information as was present in the input, only reformulated.

Conceptually, the rules inside a node may be executed in **series** or in **parallel**. The linguist controls this by setting a flag in each node; thus, some nodes may fire rules in parallel, while others fire their rules serially. (In serial nodes, rule execution terminates as soon as one rule succeeds; there is *no backup*.) In either case, all success arcs are traversed in parallel, or else all failure arcs are traversed in parallel, depending on whether any rule(s) succeeded -- meaning *all possible* morphological analyses are produced for later consideration.

To take a simple example, consider the word *derivations*, and assume that neither it nor the singular form *derivation* is in the dictionary. An English analyzer graph might have multiple start nodes, one of which is intended (by the linguist) for inflected nouns. The input glosseme

["derivations" ()]

is thus passed to the node PLURAL-NOUN, which contains, among others, the rule

["+s" ()] = ["+" (NOUN PL)].

The suffix pattern "+s" matches the glosseme string, and no features are required to be present in the glosseme, thus this rule succeeds and produces the output glosseme

["derivation" (NOUN PL)].

If PLURAL-NOUN has been marked as a terminal node (in addition to being a start node), then dictionary look-up will take place. If so, by our assumption above it fails. Our hypothetical graph contains a Success arc from PLURAL-NOUN to DEVERBAL-NOUN, which contains, among others, the rule

["+ation" (NOUN)] = ["+e" (VERB (DERIVE NOUN +ION))].

When (and if) this rule fires, it would match the suffix (*ation*) in the glosseme string *derivation*, and the feature (NOUN), and therefore transform that glosseme into

["derive" (VERB (DERIVE NOUN +ION) PL)].

Note the *e* restoration: rules can in principle remove and add any letter sequence, hence affix alternation is handled in a straightforward man-

ner. If DEVERBAL-NOUN has been marked as a terminal node, then dictionary look-up will take place: the VERB entry *derive* is retrieved. The glosseme feature list, in addition to indicating the main-entry ("derive") and lexical category (VERB) to look for in the dictionary, contains sufficient information to allow transformation of the stored entry for *derive* into a representation of its surface-form realization (*derivations*), in terms of syntactic category (NOUN), subcategorization features (PL), and semantics (the meaning of *derive*, transformed by +ION).

There remains only one problem to account for: that of compositional vs. non-compositional readings. Words with strictly non-compositional readings (e.g., *fruitful*, which is not computable from *fruit* and *ful*) simply must be stored in the dictionary; this is not a contentious claim. Words with strictly compositional readings (e.g., *derivation*, which is computable from *derive* and *ation*) may or may not be stored in the dictionary: this is an efficiency consideration, based on the usual time vs. space trade-off, and is also not subject to significant debate -- at least, not with respect to theoretical implications.

The problem arises for cases where both compositional and non-compositional readings exist for the same word (e.g., *largely*). In such situations, the non-compositional reading must of course be stored, but it would be nice if this did not *require* storage of the compositional reading as well. In Nabu, we solve this by means of a DECOMPOSABLE flag that must appear within the non-compositional definition of a word, in case that word also has a compositional reading which is to be computed. During the course of morphological analysis, performing successful dictionary look-up (at a "terminal" node) will affect subsequent processing: if the DECOMPOSABLE flag is noted, then the glosseme just submitted to dictionary look-up will *also* be passed across any success arcs leaving that node, for further analysis. In the absence of a DECOMPOSABLE flag, successful dictionary look-up marks a truly terminal case, and the results are returned (possibly along with readings from other paths in the graph) forthwith.

The operations of other types of control graphs are analogous to those of the analyzer. The principles are the same, with small exceptions (some noted above), and so are not exemplified here.

PROCESSORS IMPLEMENTED

In addition to the rule and graph interpreters per se, delivered to MCC shareholders in mid-1985, Nabu includes a variety of tools supporting the development, testing, maintenance, and multilingual documentation of morphological rule hierarchies and grammars. These tools (not described in this paper, for reasons of space) have been used to create a great many rules and several morphological processors. Non-terminals included, the English morph-rule hierarchy numbers 626 nodes; French, 434; German, 493; Spanish, 1395; and Arabic, 882. In addition to these mature rule hierarchies, some preliminary work has been done on the Russian and Japanese rule hierarchies.

ANALYZERS

The English analyzer is complete with respect to inflection; it has been successfully tested on, among other things, the entire collection of inflectional variants presented in Webster's Seventh New Collegiate Dictionary (ca. 42,500 nouns, 8,750 verbs, and 13,250 adjectives). It also accounts for the great bulk of English derivation, as determined by various word frequency lists, and is undergoing gradual, evolutionary extension to the missing (low-frequency) affixes and their combinations. A first version of this grammar was delivered to MCC shareholders in mid-1985, followed by upgrades in 1986 and 1987. The current analyzer numbers 20 nodes and 60 arcs.

As mentioned earlier, a complete Arabic morphological analyzer exists; so far as we are aware, it accounts for all morphological phenomena in the language -- no mean feat, for a language in which a single root form could in theory be realized as over 200,000 surface forms, and in which morphemes are frequently discontinuous (i.e., cannot be described by simple affixation models) [Aristar, 1987]. This 371-node, 1133-arc analyzer was delivered to MCC shareholders in mid-1986, and may represent the first complete analyzer ever produced for Arabic.

The French and German analyzers are complete with respect to inflection (highly irregular forms, like *sein* in German, naturally excepted). The former numbers 71 nodes and 121 arcs; the latter, 54 nodes and 79 arcs. The 19-node, 17-arc Spanish analyzer is nearly complete with respect to inflection; adjectives remain a temporary exception. With respect to verbs, for example, it has been tested on an extensive list of conjugated verbs [Noble and Lacasa, 1980], comprising over 6,000 surface forms, and in the first such test it was 97% accurate.

GUESSERS

A 76-node, 116-arc German guessing graph has been implemented and tested. It is still experimental, and incomplete, but it does go beyond inflection to account for some derivational processes. Our Arabic guesser is actually the Arabic analyzer graph: such strategy sharing is not always appropriate, as discussed above, but it would seem to be so for languages that,

like Arabic, are morphologically very rich, admitting as a consequence very strong predictions.

DEFAULTER

A 21-node, 23-arc English defaulting graph exists. It seems to be complete (insofar as such a processor might be), in that it constitutes a seemingly adequate component of a dictionary entry coding tool.

CONCLUSIONS

Morphological grammars in Nabu are able to account for all compositional readings of arbitrarily-complex surface-forms in a wide range of languages. Furthermore, the formalism and development environment are reasonably comfortable. These claims are supported by our implementation and large-scale testing of several diverse grammars.

For philosophical reasons, we are opposed to the idea that grammars (as opposed to individual rules) must be reversible: even if it were not for the need of five-fold rather than merely dual functionality, the need for fault-tolerance in a practical system, without consequent fault-exhibition, argues for separate analysis and synthesis grammars. We also point out that, in our implementations, the [non-reversible] control graphs tend to be much smaller in size than the hierarchies of [reversible] rules, hence the storage penalty for "redundancy" is inconsequential.

REFERENCES

Aristar, A., "Unification and the Computational Analysis of Arabic," Computers and Translation 2, 2, (April-June) 1987, pp. 67-75.

Bear, J., "A Morphological Recognizer with Syntactic and Phonological Rules," Proceedings of COLING86, Bonn, 1986, pp. 272-276.

Bennett, W.S., and J. Slocum, "The LRC Machine Translation System," Computational Linguistics 11 (2-3), 1985, pp. 111-121.

Byrd, R.J., "Word Formation in Natural Language Processing Systems," Proceedings of the 8th IJCAI, Karlsruhe, 1983, pp. 704-706.

Karttunen, L., "Kimmo - A General Morphological Processor," Texas Linguistic Forum 22, 1983, pp. 165-186.

Koskenniemi, K., "Two-level Model for Morphological Analysis," Proceedings of the 8th IJCAI, Karlsruhe, 1983, pp. 683-685.

Noble, J., and J. Lacasa. Handbook of Spanish Verbs. Iowa State University Press, Ames, Iowa, 1980.

Pounder, A., and M. Kommenda, "Morphological Analysis for a German Text-to-Speech System," Proceedings of COLING86, Bonn, 1986, pp. 263-268.

Russell, G.J., S.G. Pulman, G.D. Ritchie, and A.W. Black, "A Dictionary and Morphological Analyser for English," Proceedings of COLING86, Bonn, 1986, pp. 277-279.

Slocum, J., "An English Affix Analyzer with Intermediate Dictionary Look-up," Working Paper LRC-81-1, Linguistics Research Center, University of Texas, February 1981.

Slocum, J., and W.S. Bennett, "The LRC Machine Translation System: An Application of State-of-the-Art Text and Natural Language Processing Techniques to the Translation of Technical Manuals," Working Paper LRC-82-1, Linguistics Research Center, University of Texas, July 1982.