Fragment Processing in the DELPHI System

David Stallard, Robert Bobrow

BBN Systems and Technologies
10 Moulton St. Cambridge, MA 02138

ABSTRACT

This paper presents the fallback understanding component of BBN's DELPHI NL sysystem. This component is invoked when the core DELPHI system is unable to understand an input. It incorporates both syntax- and frame-based fragment combination sub-components, in an attempt to provide a smoother path from accurate but fragile conventional parsers on the one hand to the robust but less accurate schema-based methods on the other. The frame-based sub-component is fully integrated with the DELPHI's core grammar and parser, and represents an advance over previous proposals.

The complete fallback understanding component, incorporating both sub-components, was used in the February 1992 NL and SLS evaluations of the DELPHI system and we report on its contribution to these results, and those of its two separate sub-components. For SLS, use of the frame-based sub-component alone resulted in a figure 39.2% Weighted Error—signifigantly lower than our lowest official score of 43.7% Weighted Error.

1. INTRODUCTION

We describe the fallback understanding component of the DELPHI Natural Language component of BBN's Spoken Language System. This component is invoked when DELPHI's regular chart-based unification grammar parser is unable to parse an input; it attempts to come up with a parse and semantic interpretation, or a semantic interpretation alone, based on a fragmentary analysis of the input.

The fallback understanding component consists of three separate stages, which are invoked successively. First, the Fragment Generator produces a sequence of fragmentary subparses from the chart state left over from the unsuccessful parse. Next, two different combination modules—the Syntactic Combiner and Frame Combiner - employ alternative and complementary strategies for combining these fragments.

The Syntactic Combiner uses extended grammar rules that can skip over intervening material to combine constituents in an attempt to re-construct a plausible parse of the input. This parse can be a clause or some other useful constituent such as an imperative VP. A semantic interpretation for this reconstructed parse is automatically provided through the action of the grammar rules.

The Frame Combiner is invoked when the Syntactic Combiner is unsuccessful. It utilizes a set of pragmatic slot-filling schemata that embody the goals that ATIS users most commonly have, such as finding a flight or fare that satisfies some set of constraints, or asking about ground transportation between an airport or a city. As such, it determines only a semantic interpretation and not a parse.

The intent of this multi-step approach to fallback processing is to provide a smoother path between the accuracy but fragility of regular parsing on the one hand, and the robustness but possible inaccuracy of schemata-based methods on the other.

The remainder of the paper is taken up with detailed description of each component. The next section, Section 2, describes the Fragment Generator. Section 3 describes the Syntactic Combiner and Section 4 the Frame Combiner. Finally Section 5 gives the February 1992 NL and SLS evaluation test results for these components, separate and combined, and our conclusions based on these results.

2. THE FRAGMENT GENERATOR

The core DELPHI system consists of a unification-based grammar, an agenda-driven chart parser, a discourse component and a question-answering back-end. DELPHI's grammar rules incorporate semantic constraint and interpretation components by associating with each element of the right-hand side a grammatical relation label which keys into an associated system of semantic rules. This feature means that any term which is inserted into the chart is guaranteed to be semantically well-formed and to be annotated with one or more semantic interpretations.

The fragment generator generates a set of such semantically annotated fragments from the chart state left over after an unsuccessful parse. The algorithm for generating fragments from the chart extracts the most probable terms associated with longest sub-strings of the input, using probabilities associated with the producing rules [2].

For example, the utterance:

I want a flight uhh that arrives in Boston let's say at 3 pm

is conventionally unparseable due to the interpositions "uhh" and "let's say". The Fragment Generator produces the following set of four fragments:

```
S[I want a flight]
NO-INTERP[uhh]
REL-S[that arrives in Boston]
NO-INTERP[let's say]
PP[at 3 pm]
```

3. THE SYNTACTIC COMBINER

The Syntactic Combiner uses a special set of grammar rules, called fragment rules, to combine these fragments into a single parse. These rules have the same form as rules of the regular DELPHI grammar and incorporate semantic constraints and interpretation rules in the same way. But the method for applying the fragment rules differs in that it allows them to combine constituents even when these constituents are separated by intervening portions of the input, or when they occur in a reversed order.

Each fragment rule is adjunction oriented, in the following form:

```
X → :head X, :other-relation C
```

The following is an example, from which unification features have been omitted:

```
VP → :head VP, :pp-comp PP
```

This rule says that an existing Verb Phrase fragment and an existing Prepositional Phrase fragment can be combined together to make a new Verb Phrase with the original VP as head and the original PP as pp complement, provided they satisfy the semantic constraints associated with ":head" and ":pp-comp".

The central operation of the Syntactic Combiner is adjunction. The example rule licenses the Syntactic Combiner to "adjoin" one fragment tree into another—that is to replace a node of the first tree with a new node whose head daughter is the old node and whose other daughter is second tree. An example, using the rule above, would be the combination of the two fragments:

```
PP[at 3 pm]

VP[arrives in Boston]
```

to make the new VP:

```
VP[arrives in Boston at 3 pm]
```

Note that the adjunction node does not have to be the top of the first fragment tree: it can be any non-terminal node, as in the following pair of fragments:

```
PP[at 3 pm]

S[NP[I]
  VP[want
     NP[NP[a flight]
        REL-S[that
              VP[arrives in Boston]]
```

The algorithm that applies these rules first scans right to left taking each successive fragment and looking for fragments to its left to adjoin the first fragment into. The search for an attachment point within a fragment is right-to-left, bottom-up first, and deterministic.

The reason for the directional priority is to enforce the preference of fragment rules that the sub-term of the adjunction be to the right of the head. The algorithm then reverses direction, attempting to adjoin any remaining fragments into other fragments on their right. It oscillates back and forth in this fashion until no more fragments can be combined.

At the end of this process the largest fragment (possibly now containing other fragments which it has absorbed) is returned as the reconstructed parse, subject to cut-off restrictions which we discuss below. More than one fragment is returned in the case of multiple clausal fragments, and the discourse module is invoked to construct the interpretation of the whole.

As a simple example, let us return to the example of the previous section:

I want a flight uhh that arrives in Boston let's say at 3 pm

which generates the fragments:

```
S[I want a flight]
NO-INTERP[uhh]
REL-S[that arrives in Boston]
NO-INTERP[let's say]
PP[at 3 pm]
```

The rules that enable combination of these fragments are:

```
VP→ :head VP, :pp-comp PP
NP→ :head NP, :rel-clause REL-S
```

The first rule above licenses the attachment of "at 3 pm" to "arrives" inside the existing REL-S structure "that arrives" and the second the attachment of the combined REL-S structure to the NP "a flight" inside the clause "I want a flight". After this combination, we are left with two fragments: a clause and an unanalyzable portion of the string. Since all grammar rules in DELPHI include a semantic interpretation component[3,4], a semantic interpretation of the clause is also available.

The other fragment rules combine NPs and their various modifiers and VPs and their NP complements:

```
NP→ :head NP, :pp-comp PP
NP→ :head NP, :post-nom NP
NP→ :head NP, :whiz-rel VP
VP→ :head VP, :direct-object NP
```

The Syntactic Combiner uses a cut-off (currently .8) for the ratio of the number of words included in the final reconstructed parse to the number of words of the original input to determine whether or not to accept the final analysis as plausible. The computation of this ratio is adjusted to ignore certain words that carry little meaning ("does" "me" "could" etc.) and to block intepretations which exclude other words which do tend to change the meaning ("first", "most" etc.).

4. THE FRAME COMBINER

4.1 Overview

The Frame Combiner seeks to combine together not fragments but the semantic interpretations of fragments, and does so based not on grammar rules but on pragmatic schemata which have various "slots" to fill. It works primarily with semantic intepretations of fragments at the NP and PP level. Its approach is similar in spirit to SRI's Template Matcher [1] but it differs from that work in a number of important ways.

Most importantly, it is fully integrated with a conventional NLU system including grammar and parser. This makes it possible for it to work from recursive tree fragment structures instead of sub-strings of input. As a result, the slot-filling process is not limited to simple phrases such as "to BWI" but can also handle more syntactically and semantically complex phrases such as "to the airport closest to Washington DC". All the complex modifier structure internal to NPs which a conventional parser normally uncovers can be incorporated into slot-filling.

Moreover, while the system does not use larger constituents such as VPs and clauses to fill slots directly, it does make use of a candidate NP or PP's occurence *inside* such a larger phrase to determine which slot the candidate should fill. This enables the Frame Combiner to cope with such cases as the PP "before 3 pm", which means entirely different things, and therefore constraints entirely different slots, depending on whether it modifies the verb "arrive" or "depart".

A final difference is that the Frame Combiner attempts to determine the actual items of information that the user wants to have presented to him—that is, what slots in the frame are being asked about, as opposed to filled or constrained. This last has practical importance within the context of the ATIS task domain because it enables only what is asked about to be displayed to the user. Formerly it was sufficient simply to provide the entire extension of a suitably frame as the answer, but given the MIN/MAX scoring procedure, such a tactic is likely to result in numerous wrong answers.

The basic operation of the Frame Combiner is to input a sequence of semantically annotated fragment trees and to output a logical form as a proposed interpretation of the utterance. As intermediate steps it generates alternative sets of attribute-value "triples" and filters these according to plausibility criteria before generating a final interpretation from the chosen set. We next describe each of these steps.

4.2 Representational Triples

As intermediate output, the frame combiner first produces a set of attribute-value triples with the following structure:

```
<OPERATOR, ATTRIBUTE, VALUE>
```

The ATTRIBUTE is a single or multi-valued function. The VALUE is an element or set of elements from this function's range. The OPERATOR is a binary relation over elements of the range. In the following example:

```
<EQUAL, ORIGIN-CITY, BOSTON>
```

The operator is the relation EQUAL. The attribute in this example is the function ORIGIN-CITY, whose domain is the class FLIGHT and whose range is the class CITY. The value in the example is the individual city BOSTON.

Other typical operators are relations like TIME-BEFORE and GREATER-THAN. There is a special operator, HAS-PROPERTY, which is combined with a truth-valued (i.e. one-place predicate) attribute and the value TRUE for adjectival meanings like "non-stop".

Currently there are three classes which can serve as the domain of an attribute—FLIGHT, FARE and GROUND-TRANSPORTATION. We refer to these as the "core" classes

of the ATIS task. These core classes are associated, respectively, with the distinguished attributes FLIGHT-OF, FARE-OF and TRANS-OF, which we term the "explicit" attribute of the core class. Explicit attributes are neccesary to incorporate well-formed,parsed NP fragments whose semantic type is one of the core classes, such as "the USAir flight from Boston to Denver", without having to break them up into their component modifiers. Explicit attributes are always combined with the EQUAL operator and an element of the domain, and effectively correspond to the identity function for the domain.

An attribute-value triple can be formally viewed as a specification of a subset of the domain of the attribute of the triple. While they have a clear relationship to the notion of a template or frame, they are perhaps better thought of as disembodied "slot-constraints". Note in particular that a set of such triples is a more flexible representation than a single template in that it can uniformly combine triples whose attributes have different domains. This is important when the question itself concerns more than domain—such as both FLIGHTs and FAREs.

4.3 Generating Triples

Triples are produced from fragment trees using a recursive-descent algorithm that applies a set of pattern rules that match against fragment trees and their attached semantic interpretations. Rules can produce disjunctions of triples in case of ambiguity. The rules primarily match against NP and PP constituents, associating the semantic intepretation of the NP constituents with the value element of a triple. The algorithm mainly recurses through other types of constituents, though it does note and pass down certain items of information associated with them, such as the head-predicate of a VP.

Rules consist of a syntactic pattern component followed by optional extra constraints and an attribute assignment component. For example the rule:

```
(PP :pp FROM :object)
→
(SORT :object CITY)
(RESTRICT-SLOT EQUAL
               (:OR ORIGIN-CITY
                    TRANS-TO-CITY)
               :object)
```

applies to PPs whose preposition is "from". It requires that the NP object of the PP be of the semantic class CITY. It restricts either the ORIGIN-CITY or TRANS-TO-CITY attributes to be EQUAL to the semantics of the NP object. When applied to the fragment:

```
[PP from
    [NP boston]]
```

it generates the following two alternative triples:

```
<EQUAL, ORIGIN-CITY, BOSTON>

<EQUAL, TRANS-TO-CITY, BOSTON>
```

corresponding to the two alternatives possible in a situation where "from Boston" is uttered: either the user wants to fly from Boston to some different city or he wants to get from Boston to its airport.

Rules have a slightly more complicated form when they involve an important feature of the Frame Combiner's triple-generation process: its use of syntactic structure and context. For example, in the ATIS domain the PP "at 3 pm" means something very different when attached to a verb like "arrive". This phenomenon tends to pose a problem for conventional non-integrated template matching system, as has been noted in earlier work [1].

In the Frame Combiner this is handled by passing down the predicate representing a verb's meaning as an extra argument to the recursive descent algorithm. If a constituent was attached to a VP with a particular meaning, the slot-filling process knows this when it reaches the constituent. Slot-filling rules can be written in such a way as to behave differently depending on whether the constituent under consideration is in the context of a particular verbal predicate.

For example, in order to deal with the above phenomenon, the following rule applies to PP fragments where the NP :OBJECT is of type TIME-OF-DAY, and :PREP is any preposition from which a temporal relation can be derived. This temporal relation restricts whatever slot is determined appropriate by the ATTRIBUTES component of the rule:

```
(PP :PP :PREP :OBJECT)
→
(SORT :OBJECT TIME-OF-DAY)
(TEMPORAL-RELATION :PREP :REL)
(RESTRICT-SLOT
   :REL
   (ATTRIBUTES
     (CONTEXT ARRIVE ARRIVAL-TIME)
     (DEFAULT DEPARTURE-TIME)
     (GENERAL ARRIVAL-TIME
              DEPARTURE-TIME))
   :OBJECT)
```

The ATTRIBUTES expression delivers zero or more attributes as a disjunction the specific attributes depending

upon which of its evidence clauses is satisfied. CONTEXT evidence is the strongest. NON-LOCAL evidence is next, and it includes situations where a particular verb is merely present elsewhere in the input, without dominating the constituent. DEFAULT evidence is the assignment preferred whereas GENERAL evidence is all the assignments allowed.

4.4 Filtering Sets of Triples

When all fragments have been analyzed through recursive descent, the system takes the cartesian product of all disjunctive interpretations to obtain the set of all alternative sets of triples. These are then filtered to leave only the most plausible sets of triples.

There are several criteria for plausibility. The most obvious is that two or more triples on the same attribute not specify contradictory values for the attribute. Another is that a set not contain any two triples with clashing attribute domains. For example, in the ATIS task one never sees queries that combine flights and ground transportation (even though such are certainly expressible, e.g. "Show me USAir flights to airports that have limousine service"). Thus FLIGHT and GROUND-TRANSPORTATION are clashing domains. On the other hand, queries concerning both flights and fares do frequently occur ("Show flights to Boston and their fares") so FLIGHT and FARE are not clashing domains.

Another criterion is that the set of triples have the commonly seen linguistic form for the domain. Thus, while "the airport" and "the city" are plausible fillers for TRANS-TO-AIRPORT and TRANS-TO-CITY in the GROUND-TRANSPORTATION domain they are much less plausible fillers for FLIGHT domain attributes, simply because proper noun fillers are fare more common for these.

Criteria such as non-clashing domains are hard criteria, and therefore any triple set which violates them is discarded. Other criteria, like the plausiblity of linguistic domain, are softer, and the system merely prefers not to violate them.

If there is more than one plausible set of triples, the Frame Combiner will, depending on switch setting, either give up or appeal to extrasentential discourse to resolve the ambiguity (much as the core DELPHI system will do).

4.5 Choosing the Information to Display

At each turn in dialogue, any system performing an information retrieval task, such as ATIS, is essentially required to display a set of objects. This holds for WH questions ("which flights..."), imperatives ("show me"), and existential yes-no questions ("are there any flights..."). On this perspective, the different sets of objects and relationships between them are one part of the meaning of the query, and are represented by the sets of triples. The other part of the meaning is the question of which of these sets to display. We refer to as the "topic" of the query.

To choose one (or more) of the triples as the topic means to display its value set, as it relates to all other value sets of the other triples. Several different heuristics are used, and are ranked in priority. Each is tried in succession until a topic is chosen.

Most obvious is whether the filler of the triple is a WH noun phrase. If it is, it definitely must be the topic.

Next are any "priority" domains that are not normally used merely to constrain other sets. An example is GROUND-TRANSPORTATION—the typical ATIS user does not ask to see cities that have a particular type of ground transportation—the user wants to see the ground transportation itself.

"Unconstrained" triples are another likely topic. A triple is "unconstrained" if its filler is a bare common nominal, such as "airline", and its attribute is a total function. Since every FLIGHT has an AIRLINE, the user is most unlikely to be imposing the vacuous constraint that the flight is on some airline (even though this is again expressible). Rather, the user is much more likley to be interested in seeing the airline of the flight.

4.6 Generating the Final Interpretation

The Frame Combiner generates a final logical form from a chosen set of triples by first associating a variable with each triple filler ("value" slot) and a variable with each of the core classes present, in the set, whether through explicit attributes on the class or implicitly as the domain of another attributes. It generates a matrix formula in which all the attributes present are binary relations and the generated variables are the arguments to these binary relations. Quantificational structure, corresponding to the fillers of triples, is then generated. The quantifiers for topic triples are treated as though they were WH quantifiers, and appropriate display commands generated.

5. RESULTS AND DISCUSSION

This system was run with the DELPHI NL system in the February 1992 official evaluation. Using the same constant executable Lisp image ("disksave") run for the official results, the test was run using a number of different switch settings, and scored with the version of the NIST comparator used for the official results. The switch conditions were: no fallback processing at all (which is simply the core DELPHI system), Syntactic Combiner only, Frame Combiner only, and both Syntactic Combiner and Frame Combiner working together (which was the condition used in the official results). The figures for NL only are reported in Table 1.

	%T	%F	%NA	%WE
no fallback	69.3	7.4	23.3	38.1
syn only	73.1	8.9	18.0	35.8
frame only	78.3	9.6	12.1	31.3
both(official)	76.7	10.6	12.7	33.9

Table 1: NL Results

Note the frame-only condition is actually better than result officially reported, in which both fallback sub-components were used.

For the SLS test, the output of BBN's BYBLOS N-best recognizer was used, with N = 5. The core DELPHI system (without fragments) was first tested against the five theories. If an intepretation was found for one of them, it was returned. Otherwise, the fallback methods were applied.

Results for three of the four conditions are seen in Table 2 (results for the no-fallback = core DELPHI condition were unavailable as of this writing). The figure for the combination of both fragment modules (the configuration used in the official test) reflects an slight downward adjustment from the original value of 43.7 that corrects a purely procedural error committed during our running the test (the file that specifies "todays's date" for each query was not loaded, leading to a small increase in the number of wrong answers). This problem was fixed in obtaining the results in Table 2. As in the regular NL test, the SLS results show an noticeable improvement over the official results when the Frame Combiner is used alone.

These results tend to undercut a central premise of our original strategy: namely that using both fragment combination methods together would improve the result over the use of either alone. Our tentative hypothesis is that the Syntactic Combiner, when failing and passing to the Frame Combiner the best results of its combination attempt, is passing wrongly combined fragments which mislead the Frame Combiner.

On the other hand, these results do show the utility of the Frame Combiner when used alone. For NL only, it reduced the No Answer rate by 11.2 percentage points, and Weighted Error by 4.2 percentage points. For SLS, it reduced the Weighted Error from the adjusted official value of 43.2% to the signifigantly lower value of 39.2%.

	%T	%F	%NA	%WE
no fallback	65.2	10.5	24.3	45.6
syn only	68.9	11.5	19.7	42.6
frame only	73.8	13.0	13.2	39.2
both(official)	71.8	15.4	12.8	43.7
both(adjusted)	71.9	15.1	13.0	43.2

Table 2: SLS Results

ACKNOWLEDGEMENTS

The work reported here was supported by the Advanced Research Projects Agency and was monitored by the Office of Naval Research under Contract No. N00014-89-C-0008. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

REFERENCES

1. Jackson, E., Appelt, D., Bear, J., Moore, R. and Podlozny, A. *A Template Matcher for Robust NL Interpretation*, in Proceedings Speech and Natural Language Workshop February 1991
2. Bobrow, R.J. *Statistical Agenda Parsing* in Proceedings Speech and Natural Language Workshop February 1991
3. Bobrow, R.J., Ingria, R. and Stallard, D. *Syntactic/Semantic Coupling in the DELPHI System* to appear in Proceedings Speech and Natural Language Workshop February 1992
4. Bobrow, Ingria, R. and Stallard, D. *The Mapping Unit Approach to Subcategorization* in Proceedings Speech and Natural Language Workshop February 1991