

PROFER: PREDICTIVE, ROBUST FINITE-STATE PARSING FOR SPOKEN LANGUAGE

Edward C. Kaiser, Michael Johnston and Peter A. Heeman

Center for Spoken Language Understanding
Oregon Graduate Institute
PO Box 91000 Portland OR 97291
kaiser@cse.ogi.edu

ABSTRACT

The natural language processing component of a speech understanding system is commonly a robust, semantic parser, implemented as either a chart-based transition network, or as a generalized left-right (GLR) parser. In contrast, we are developing a robust, semantic parser that is a single, predictive finite-state machine. Our approach is motivated by our belief that such a finite-state parser can ultimately provide an efficient vehicle for tightly integrating higher-level linguistic knowledge into speech recognition. We report on our development of this parser, with an example of its use, and a description of how it compares to both finite-state predictors and chart-based semantic parsers, while combining the elements of both.

1. INTRODUCTION

Parsing spontaneous speech is difficult. Users often say things that are not covered in the system's lexicon and grammar. Their speech may also contain noise, on-line verbal corrections, or other extragrammaticalities. The recognizer itself may make errors in decoding the speech. In the face of all this variability, typical rule-based parsers are too brittle. Because of this brittleness, rule-based grammars are generally not used for language modeling in speech recognition. Statistical n -gram language models, instead, have become the standard. They are more forgiving and therefore robust; but, they cannot capture the long-range dependencies represented in higher-level grammar definitions.

It is possible to build robust, rule-based parsers. The JANUS system for multi-lingual spontaneous speech translation has used various approaches to robust parsing [8]. At the transcription level a *skipping* version of Tomita's (GLR) parser has been used that includes probabilities on individual *reduce* actions (in the manner of Briscoe and Carroll [2]). And at the semantic level the PHOENIX system has been used for robust "concept-spotting" [12].

PHOENIX works by filling *slots* within a *case-frame*. The patterns which map to individual *slots* can occur in the input any number of times in any order. Out-of-grammar words that occur between *slots* can be skipped. Thus, partial parses (i.e., *frames* in which only some of the available *slots* have been filled) can be returned. This "concept-spotting" approach allows for considerable robustness in the face of spontaneous spoken input, and its usage is common in spoken language information systems [11].

Our system, PROFER (pronounced "proffer") — which stands for Predictive ROBust Finite-state parsER, is designed to be functionally equivalent to PHOENIX in terms of robust parsing. Figure 1 offers a simple example of robust "concept spotting." Us-

ing the CSLU toolkit's Rapid Application Developer (RAD) [7] we have built a small prototype system that extracts canonical representations of movie titles and times-of-day. This example uses a very simplistic language model in which every possible word is equally likely at every state in the grammar.

System Prompt:

What movie info would you like to know about?

User Query:

Uhhh lets see, is uhhh Gone With The ... no not that, is Titanic playing in the late afternoon uhhhm I mean in the early evening?

Recognizer Transcript:

with see in with you some with with k with with gone would see a know know with ahhh with a a this with with titanic k with with playing with in the we afternoon with umm umm evening of early evening

Case-frame Hierarchy (returned by PROFER):

```
[fsType:movie_query_type,
  title_fi:[titanic_c:[titanic]],
  time_fi:[early_evening_c:[early_evening]]]
```

Canonical Transcript (returned by PROFER):

titanic early evening

Figure 1: Robust parsing example.

In Figure 1 the *User Query* includes both online verbal corrections, and out-of-grammar vocabulary. The *Recognizer Transcript* of that *User Query* is rife with mis-recognitions, insertions and deletions — due to our overly-simplistic language model. Given all of this variability, PROFER still robustly extracts the intended meaning, as both a *Case-frame Hierarchy* and as a simple *Canonical Transcript*.

Both GLR parsers and chart-based parsers, like PHOENIX, can also be used to provide prediction sets for limiting the acoustic search space of a speech recognizer. However, the computational expense of using such parsers for language modeling can overload the decoder. For example, Murveit and Moore [5] abandoned the formation of prediction sets in their dynamic network generation system because it was too time-consuming. One way to reduce the computational expense of using these parsers as predictors is to transform them into finite-state acceptors (FSAs), as has been described by Pereira and Wright [6].

The goal of our research in developing PROFER is to combine the tractability, speed and predictiveness of finite-state approaches to language modeling, like Pereira and Wright's FSAs, with the ro-

business and “concept-spotting” capabilities of a system like PHOENIX. In other words, we are creating a finite-state parser capable both of producing sequential prediction sets, and also of incrementally building a robust, *case-frame* representation of concepts extracted from the input.

In Section 2 we will describe Pereira and Wright’s method for approximating CFGs as FSAs. It involves some of the same preliminary steps taken by a GLR parser, and so will be instructive. Then we will take a high-level look at the PHOENIX system (Section 3). Next we will show how PROFER combines the elements of both approaches (Section 4), and conclude with some discussion of our future work (Section 5).

2. PREDICTIVE FINITE-STATE ACCEPTORS

Pereira and Wright use the following steps in their method:

- **Translate** the defining feature-constrained, phrase-structure grammar into a CFG (not shown).
- From the CFG create the LR(0) characteristic machine (Figure 2), using standard LR techniques [1].

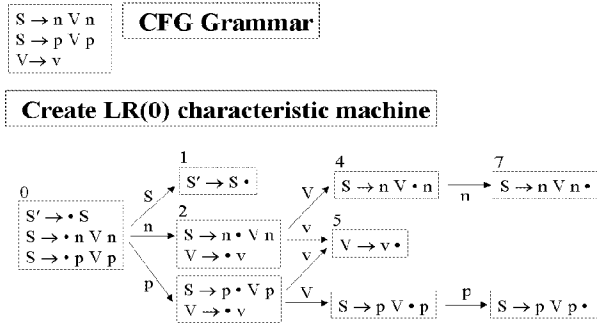


Figure 2: LR(0) machine creation.

- **Flatten** the LR(0) characteristic machine into a preliminary finite-state acceptor (PFSA) (Figure 3). That is, from the item-sets, shift-transitions, and logical reduction paths in the LR(0) machine, create the states and arcs of the PFSA.

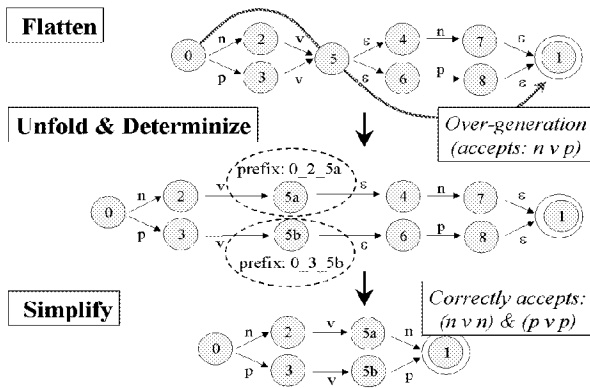


Figure 3: The FSA approximation process.

- To reduce the occurrence of over-generation in the PFSA (exemplified by the heavy-lined path in the **Flattening** step of Figure 3), **unfold** its states into new states that are sets of stack-prefixes arriving at the same location in the PFSA.
- **Determine** the group of new FSA states by collapsing the stack-prefixes into equivalence sets (see the **Unfold & Determine** step of Figure 3).
- **Simplify** the resulting FSA into the final approximation by removing any non-terminal arcs, and removing any null-arcs along with the states they lead into (see the **Simplify** step of Figure 3).

The resulting FSA can be used for the purpose of sequentially producing prediction sets; that is, FSAs can provide a higher-level, grammatical language model for speech recognition, and have been used for that purpose in various limited task domains [6].

Pereira and Wright point out that it has been shown that no possible algorithm exists for converting all CFGs that describe regular languages into FSAs [6]. This is not a problem for their method, because some over-generation in the prediction sets is acceptable. However, for the purposes of robustly generating *case-frame* parse-trees, as PROFER does, inexact finite-state representations of the grammar could allow the parser to spuriously find concepts that did not exist in the input.

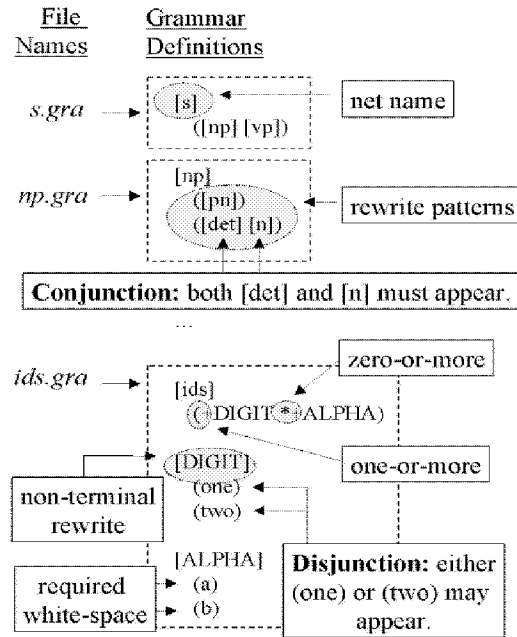


Figure 4: PROFER’s grammar definition formalism.

The *case-frame* grammar definitions accepted by both PHOENIX and PROFER are notational means for expressing regular languages, with the restriction that no *rewrite pattern* can be empty. So, an exact FS translation of any *case-frame* definition is always possible (see [3]). Figure 4 illustrates the basic elements of PROFER’s (and PHOENIX’s) grammar notation.

3. PHOENIX: A ROBUST SEMANTIC PARSER

Figure 5 illustrates the basic outline of the PHOENIX system. PHOENIX goes immediately from a grammar definition written in its own *case-frame* style, as illustrated in Figure 4, to individual transition networks (TNs) as shown in step 1 of Figure 5.

There is no intervening translation into the LR(0) characteristic machine, as in Pereira and Wright's algorithm. As each TN is constructed its non-terminals are re-written, based on the *rewrite patterns* in its definition file (see Figure 4). At the end of this process all non-terminals arcs are removed from each TN's representation (see step 2 in Figure 5).

During run-time (see step 3 in Figure 5), when an arc traversed by a TN is encountered the chart is searched for a previous occurrence of that TN in that position. If no previous occurrence is found then that TN is descended into recursively and processing is continued at the lower level. When an arc leading to the final state of that TN has been traversed, the position of the completed TN is charted, and the parse returns to the previous level of recursion (see the gray arrow in step 3 of Figure 5).

Each chart entry extends a branch within the overall parse-tree. When all possible chart entries have been made, the resulting paths are scored heuristically. The path or paths which account for the largest number of input words in the fewest number of nets, slots and frames receive the highest score.

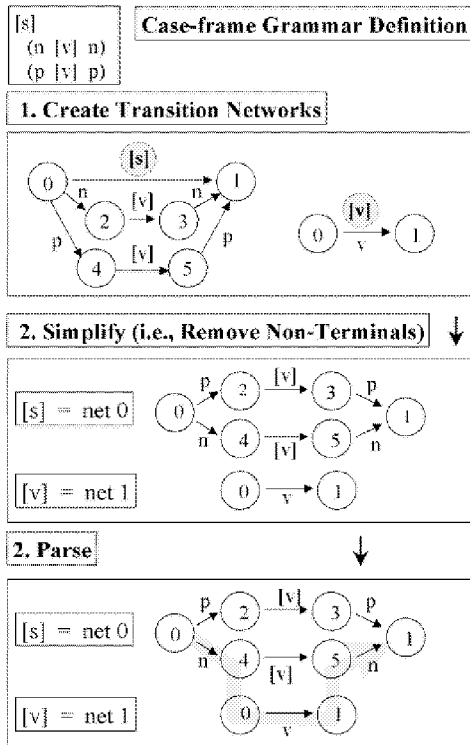


Figure 5: PHOENIX system outline.

4. PROFER: COMBINING PREDICTION AND ROBUST PARSING

PROFER translates from its grammar definition files (cf. Figure 4) immediately into a series of finite-state machines (FSMs), just as PHOENIX does, with no intervening translation into the LR(0) machine (see 1 & 2 in Figure 5). Figure 6 illustrates the next step of PROFER's compilation, which does not occur in the PHOENIX system. We refer to this step as "contextualization," and it is very similar (in terms of outcome) to the steps of FSA construction illustrated in Figure 3.

PHOENIX uses the identifying numbers associated with each separate FSM during run-time to label each edge as it is added to a path in the chart. In a GLR parser these net identifiers would be encoded in the stack state at each position of the parse. In PROFER, as in Pereira and Wright's FSAs, they become part of the names of the finite-state machine's individual states (as illustrated in Figure 6). Thus, state names in PROFER hold much in common with the equivalent stack prefix configurations that would occur in a parallel GLR parse.

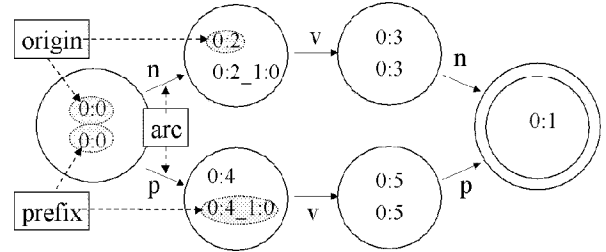


Figure 6: Contextualization in PROFER.

In PROFER, *origins* (Figure 6) are a superset of the states in the final FSM. Each *origin* contains two substructure trees. The first facilitates efficient compilation. It holds the full hierarchy of *prefixes*, *arcs* and *terminals*, and is illustrated in Figure 6. Conceptually this first substructure is similar to an item-set (cf. Figure 2) in the characteristic machine of a Left-Right parser [1]. The second facilitates run-time parsing and the formation of prediction sets. It holds top-level groupings of all the left-corner terminals associated with each *origin*. The terminals in each grouping are cross-indexed into their respective positions within the first substructure.

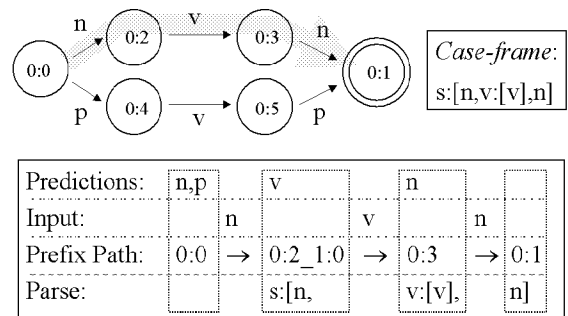


Figure 7: Building the case-frame parse tree.

Parsing begins by extending every left-corner contextual-group

that occurs in the first origin (see Figure 7). It then proceeds as an alternation between hash-table lookups, and the “unioning” together of contextual-groups from all possible next states into a new prediction set.

Parse-tree information is carried forward through a system of linked tokens that preserve the sequence of *prefixes*. This is shown both in context on the **Prefix Path** line of Figure 7, and then separately in Figure 8, which is a print-out of the last two token nodes of the finished path represented in Figure 7. The last token in Figure 8 illustrates the major fields included in each token node structure. The fields in gray boxes correspond to those on the **Prefix Path** line in Figure 7. Overall the fields in the token node structure provide the information needed to construct the finished *Case-frame* representation shown in Figure 7.

```

...
Prefix(oi): 0:2
Prefix(opi): 0:2 1:0
fSelfLoopPattern(0) fSelfLoopTerm(0) term(v)
Prefix(noi): 0:3

token[position(2),tableOffset(3)]
poi(1) popi(0) oi(3) opi(0) opai(0)
Prefix(poi): 0:2
Prefix(popi): 0:2 1:0
fSelfLoopPattern(0) fSelfLoopTerm(0) term(v)
Prefix(oi): 0:3
Prefix(opi): 0:3
fSelfLoopPattern(0) fSelfLoopTerm(0) term(n)
Prefix(noi): 0:1

```

Figure 8: Finished path token nodes.

The token paths within the overall parse-tree are shared in a manner that is conceptually similar to the graph-structured stack (GSS) mechanism of a GLR parser [10]. A chart-based or GLR parser must walk each extension during run-time to collect the set of possible next terms. In our construction that collection work is done during compilation, leaving only the “unioning” operation to be done at run-time.

Various researchers [9, 4] have written extensively on modifications to Tomita’s GLR algorithm for better managing of the “ancestor” information, which is equivalent to the stack history information that Pereira and Wright describe as being “crucial” to the workings of their algorithm [6]. In PROFER’s design we have settled on a method that uses complete hierarchical locations to name each *origin*, *prefix* and *arc*, thus efficiently encoding their stack histories. Moreover, these encoded name strings also function as hash-table keys, providing fast table lookups for indexing during run-time.

5. CONCLUSION AND FUTURE WORK

We have outlined our method for constructing a predictive, robust finite-state parser, and contrasted it with both FSAs and the PHOENIX system, a robust chart-based semantic parser. We have also demonstrated the use of this parser within a small prototype system, built within the CSLU toolkit’s RAD environment.

Although our work on PROFER is still in its early stages, we have shown that such an approach is viable, both as a stand-alone semantic parser, and as a stand-alone finite-state predictor. In the

future we hope to show that such an approach will also be viable for tightly integrating higher-level linguistic constraints into the speech recognition process.

6. ACKNOWLEDGMENTS

The first author was funded by the Intel Research Council, the NSF, and the CSLU member consortium. We also wish to thank Ron Cole, Jim Larsen, and Wayne Ward for valuable discussions and support.

7. REFERENCES

- [1] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, 1988.
- [2] Ted Briscoe and John Carroll. Generalized probabilistic lr parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics*, 19(1):25–59, 1993.
- [3] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
- [4] J. R. Kipps. GLR Parsing in Time $O(n^3)$. In Masaru Tomita, editor, *Generalized LR Parsing*, pages 43–59. Kluwer Academic Publishers., 1991.
- [5] Hy Murveit and Robert Moore. Integrating natural language constraints into hmm-based speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing, Vol. 1*, pages 573–576, Albuquerque, April 1990.
- [6] Fernando C. N. Pereira and Rebecca N. Wright. Finite-state approximations of phrase-structure grammars. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing*, pages 149–173. The MIT Press, 1997.
- [7] J. Schalkwyk, L. D. Colton, and M. Fanty. The cs-lu-sh toolkit for automatic speech recognition: Technical report no. cs-lu-011-96, August 1996.
- [8] B. Suhm, L. Levin, N. Coccaro, J. Carbonell, K. Horiguchi, R. Isotani, A. Lavie, L. Mayfield, C. P. Rose, Van Ess-Dykema C., and A. Waibel. Speech-language integration in a multi-lingual speech translation system. In Paul McKevitt, editor, *Proceedings of the AAAI-94 Workshop on the Integration of Natural Language and Speech Processing*, pages 92–99, Seattle, Washington, 1994.
- [9] H. Tanaka, Hiu Li, and T. Tokunaga. Incorporation of phoneme-context-dependence into lr table through constraint propagation method. *Journal of Japanese Society for Artificial Intelligence*, 11(2):246–54, March 1996.
- [10] Masaru Tomita. *Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems*. Boston, Massachusetts: Kluwer Academic Publishers, 1986.
- [11] Gertjan Van Noord, Gosse Bouma, Rob Koeling, and Mark-Jan Nederhof. Robust grammatical analysis for spoken dialogue systems. *Natural Language Engineering*, 4(1):1–48, 1998.
- [12] W. Ward and S. Young. Flexible use of semantic constraints in speech recognition. In *Proceedings of ICASSP '93. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, MN, USA, IEEE, New York, NY, USA, 1993.