Polish Parsers, Step by Step

R. John M. Hughes
Computing Science
Chalmers University of Technology
SE-412 96 Gøteborg, Sweden
rjmh@cs.chalmers.se

S. Doaitse Swierstra
Computer Science
Utrecht University
The Netherlands
doaitse@cs.uu.nl

August 26, 2003

Overview

- Motivation
- Polish Expressions
- Recognisers
- Polish Parsing

Overview

- Motivation
- Polish Expressions
- Recognisers
- Polish Parsing
  - Basic parsers
  - Monadic Interface
  - Parsing ambiguous grammars

Overview

- Motivation
- Polish Expressions
- Recognisers
- Polish Parsing
  - Basic parsers
  - Monadic Interface
  - Parsing ambiguous grammars
- Conclusion

Standard parser combinators: Types

$$
\begin{array}{l}
\textbf{infixl}\ 3\ \texttt{<|>} \\
\textbf{infixl}\ 4\ \texttt{<*>}, \texttt{<*}, \texttt{<\$}, \texttt{<\$>}
\end{array}
$$

$$
\begin{array}{lcllll}
\textbf{type}\ \mathit{Parser}\ a & = & \mathit{String} \rightarrow [(a, \mathit{String})] & & & \\
(\texttt{<|>}) & :: & \mathit{Parser}\ a & \rightarrow \mathit{Parser}\ a & \rightarrow & \mathit{Parser}\ a \\
(\texttt{<*>}) & :: & \mathit{Parser}\ (b \rightarrow a) & \rightarrow \mathit{Parser}\ b & \rightarrow & \mathit{Parser}\ a \\
\mathit{pSucceed} & :: & a & & \rightarrow & \mathit{Parser}\ a \\
\mathit{pFail} & :: & & & & \mathit{Parser}\ a \\
\mathit{pSym} & :: & \mathit{Char} & & \rightarrow & \mathit{Parser}\ a
\end{array}
$$

List of Successes Implementation: backtracking

$$
\begin{array}{lll}
pSucceed\ \ v\ \ inp & = & [(v, inp)] \\
pFail & = & [\,] \\
(p\ \texttt{<|>}\ q)\ \ inp & = & p\ inp +\!\!+\ q\ inp \\
(p\ \texttt{<*>}\ q)\ \ inp & = & [(f\ \ a, rr) \quad | \quad (f, r) \leftarrow p\ inp \\
 & & \qquad\qquad\qquad\quad , \quad (a, rr) \leftarrow q\ r \\
 & &] \\
pSym\ \ a\ \ inp & = & \mathbf{case}\ inp\ \mathbf{of} \\
 & & \quad (s : ss) \rightarrow \mathbf{if}\ \ a \equiv s \\
 & & \qquad\qquad\qquad\quad \mathbf{then}\ [(s, ss)] \\
 & & \qquad\qquad\qquad\quad \mathbf{else}\ [\,] \\
 & & \quad [\,] \rightarrow [\,]
\end{array}
$$

Shortcomings of this implementation

- Backtracking keeps a reference to the complete input
  - can be alleviated by explicit extra annotations (e.g. *Parsec*
  - requires detailed analysis of the grammar
  - has to be done by skilled programmers
  - is error prone

Shortcomings of this implementation

- Backtracking keeps a reference to the complete input
  - can be alleviated by explicit extra annotations (e.g. *Parsec*
  - requires detailed analysis of the grammar
  - has to be done by skilled programmers
  - is error prone
- Result only becomes available after a complete parse has been found
  - very unfortunate for very lost inputs: e.g. processing a file
  - complete result is first constructed in memory

Shortcomings of this implementation

- Backtracking keeps a reference to the complete input
  - can be alleviated by explicit extra annotations (e.g. *Parsec*
  - requires detailed analysis of the grammar
  - has to be done by skilled programmers
  - is error prone
- Result only becomes available after a complete parse has been found
  - very unfortunate for very lost inputs: e.g. processing a file
  - complete result is first constructed in memory
- Ambiguous grammars will be very expensive
  - the part of the input after the ambiguous section is parsed over and over again with the same parsers