15-411 – Lecture 8
Handout: Resolving Conflicts in a Parse Table

Jan Hoffmann

September 22, 2016

We can use shift-reduce parsing to parse ambiguous grammars, but we have different choices for parse tables. Consider the following grammar.

$$\begin{array}{lrcl} [\text{plus}] & E & \longrightarrow & E \texttt{ + } E \\ [\text{times}] & E & \longrightarrow & E \texttt{ * } E \\ [\text{number}] & E & \longrightarrow & num \\ [\text{parens}] & E & \longrightarrow & \texttt{(} E \texttt{)} \end{array}$$

We can see that the grammar has *four shift/reduce conflicts*, while all other actions (including errors) are uniquely determined. These conflicts arise when E + E or E * E is on the stack and either + or * is the first character in the remaining input string. It is called a shift/reduce conflict, because either a shift action or a reduce action could lead to a valid parse. Here, we have decided to resolve the conflicts by giving a precedence to the operators and declaring both of them to be left-associative.

It is also possible to have reduce/reduce conflicts, if more than one reduction could be applied in a given situation, but it does not happen in this grammar.

To resolve the conflicts, we select one of the available actions in the parse table. In this way, we define precedence and left/right associativity.

$\beta \setminus$ a	+	*	num	(	)	$
E + E	reduce(plus) (+ left assoc.)	shift (+ < *)	error	error	reduce(plus)	reduce(plus)
E * E	reduce(times) (+ < *)	reduce(times) (* left assoc.)	error	error	reduce(times)	reduce(times)
num	reduce(number)	reduce(number)	error	error	reduce(number)	reduce(number)
(E)	reduce(parens)	reduce(parens)	error	error	reduce(parens)	reduce(parens)
E +	error	error	shift	shift	error	error
E *	error	error	shift	shift	error	error
(E	shift	shift	error	error	shift	error
(	error	error	shift	shift	error	error
ϵ	error	error	shift	shift	error	error
ϵ E	shift	shift	error	error	error	accept(E)