LR(0) parsers

Given a context free grammar $G = (V, T, P, S)$ with:
V a finite set of variables/non-terminals,
T a finite set of terminals,
P a finite set of productions of the form $A \to \alpha$ with $A \in V$ and $\alpha \in (V \cup T)^*$,
$S \in V$ the starting symbol.

We can construct an LR(0) parse table as follows:

We define the set of items $I : \{A \to \alpha \cdot \beta | \alpha, \beta \in (V \cup T)^* \vee A \to \alpha\beta \in P\}$.
The dot indicates the contents of our parsing stack (left of the dot), and the expected input (right of the dot).

We also define the following functions in pseudo-code to aid in the construction of the parsing table:

$\mathit{Closure} : 2^I \to 2^I$
$\mathit{Closure}(\mathit{items}) \mapsto \mathit{result}$
  $\mathit{result} = \mathit{items}$
  $\mathit{changed} = \textsc{True}$
  while $\mathit{changed} == \textsc{True}$:
    $\mathit{changed} = \textsc{False}$
    for each item $A \to \alpha \cdot X\beta \in \mathit{result}$ with $X \in V$:
      for each production $X \to \gamma \in P$ with $X \to \cdot\gamma \notin \mathit{result}$:
        $\mathit{result} = \mathit{result} \cup \{X \to \cdot\gamma\}$
        $\mathit{changed} = \textsc{True}$

$\mathit{Goto} : 2^I \times (V \cup T) \to 2^I$
$\mathit{Closure}(\mathit{items}, \mathit{symbol}) \mapsto \mathit{result}$
  if $\mathit{symbol} == \$$:
    $\mathit{result} = \{\}$
  else:
    $\mathit{result} = \{\}$
    for each item $A \to \alpha \cdot X\beta \in \mathit{items}$ with $X \in V$:
      $\mathit{result} = \mathit{result} \cup \{A \to \alpha X \cdot \beta\}$
    $\mathit{result} = \mathit{Closure}(\mathit{result})$

We can then use the following algorithm to create a state diagram for the LR(0) parser:

$start = Closure(\{S' \to \cdot S\$\})$
$states = \{start\}$ $\quad 2^{2^I}$
$edges = \{\}$ $\quad (2^I \times (V \cup T)) \to 2^I$
$changed = \text{TRUE}$
while $changed == \text{TRUE}$:
  $changed = \text{FALSE}$
  for each $state \in states$
    for each item $A \to \alpha \cdot X\beta \in state$ with $X \in (T \cup V)$:
      $changed = \text{TRUE}$
      target $= Goto(state, X)$
      states $=$ states $\cup$ {target}
      edges $=$ edges $\cup \{(state, X) \to target\}$

Based on the generated state diagram, we can generate the parse table. The parse table is made up of states for the rows, and terminals and non-terminals for the columns. An action in the parse table can be either to shift (terminals), to goto (non-terminals), to reduce (finishing a grammar rule), or to accept.

We define the following actions:
SHIFT t $\quad t \in T$ consumes the current input character and puts it on the stack, then puts t on the stack.
REDUCE $A \to \gamma$ $\quad A \to \gamma \in P$ pops the current state (top of the stack) and the previous symbol off of the stack, then looks up a GOTO action for the new current state and A and pushes A and the state from the GOTO action on the stack.
GOTO s $\quad s \in states$ used by REDUCE actions to determine the next state to go to.
ACCEPT makes the parser accept the input string.

We can now say:

$$\text{TABLE}(state, X) = \begin{cases} \text{SHIFT}\, r & \text{if } X \in T, \exists! r : (state, X) \to r \in edges \\ \text{GOTO}\, s & \text{if } X \in V, \exists! r : (state, X) \to s \in edges \\ \text{REDUCE}\, A \to \gamma & \text{if } A \to \gamma \cdot \in state \\ \text{ACCEPT} & \text{if } S \to S \cdot \$ \in state \end{cases}$$

Where the value of *state* and of X during execution are taken from the top of the stack, and the current input character respectively.
If the value for a $(state, X)$ pair in TABLE can be multiple things (for example, a shift and a reduce, or a two reduces), then this indicates that the grammar G is not LR(0) compatible.