A Verifiable, Executable SLR Parser Generator

Aditi Barthwal
Supervisor: Dr. Michael Norrish

ESOP '09
WHY?

- Provide a mechanised theory of parsing
- Parsers as external proof oracles
- Part of bigger piece of work, that of providing mechanised language theory in HOL
SLR Parsing

- Simple Left-to-right Rightmost derivation (SLR)
- Bottom-up parser
- Parser has a stack and an input. The first $k$ tokens of the input are the lookahead (one in this case).

Possible actions
- **Shift**: move the first input token to the top of the stack
- **Reduce**: Choose a grammar rule $X \rightarrow ABC$; pop $C, B, A$ from the top of the stack; push $X$ on to the stack

- Uses a DFA to parse the input stream

- Example, derivation of $ab$ w.r.t. $G$: $S \rightarrow AB, A \rightarrow a, B \rightarrow b$
  - $ab \leftarrow Ab \leftarrow AB \leftarrow S$
An example

- **Initial grammar**
  - $E \rightarrow E + T \mid T$
  - $T \rightarrow n \mid (E)$

- **Augmented grammar**
  - $S \rightarrow E$
  - $E \rightarrow E + T \mid T$
  - $T \rightarrow n \mid (E)$
DFA for the grammar

- $S \rightarrow \cdot E$
- $E \rightarrow \cdot E + T$
- $E \rightarrow \cdot T$
- $T \rightarrow \cdot n$
- $T \rightarrow \cdot (E)$

Initial grammar
- $E \rightarrow E + T | T$
- $T \rightarrow n | (E)$

Augmented grammar
- $S \rightarrow E$
- $E \rightarrow E + T | T$
- $T \rightarrow n | (E)$
DFA for the grammar

Initial grammar
- $E \rightarrow E + T \mid T$
- $T \rightarrow n \mid (E)$

Augmented grammar
- $S \rightarrow E$
- $E \rightarrow E + T \mid T$
- $T \rightarrow n \mid (E)$

Graph with transitions:
- $S \rightarrow E$ (6)
- $E \rightarrow E + T$ (8)
- $T \rightarrow n$ (2)
- $T \rightarrow (E)$ (7)
- $E \rightarrow T\ast$ (1)
- $T \rightarrow (E)\ast$ (3)
- $E \rightarrow E + T\ast$ (9)
HOL Implementation

Augmented grammar

SLR (check DFA)

SLR parser

Generator

Parser

Tokens

Parse token

Tokens in language

SOME tree

NONE

Reject

NONE
Implementation

- `lrparse :: grammar → symbol
  → symbol → symbol list
  → ptree option`
- `sgoto :: grammar → state → symbol
  → state`
- `moveDot :: state → symbol → state`
- `closure :: grammar → state → state`
- `reduce :: grammar → state → symbol
  → rule list`
- `followSet :: grammar → symbol
  → symbol set`
Properties of Interest

- **Soundness**
  If a DFA can be constructed for some grammar \( g \) and the parser is able to parse token stream \( t \) into a parse tree, then \( t \) belongs in the language of \( g \).

- **Completeness**
  If a DFA can be constructed for some grammar \( g \) and token stream \( t \) belongs in the language of \( g \), then the parser can parse \( t \) into a parse tree.
Strategy Framework - Soundness

- Soundness
  - Show that stack has invariant that any tree on it is valid with respect to the grammar

- $S \rightarrow A \rightarrow a \quad \text{w.r.t} \quad G: \quad S \rightarrow A, \ A \rightarrow a$, i.e. $a \in L(G)$

State = Stack + Tree

```
<table>
<thead>
<tr>
<th>Shift</th>
<th>Reduce</th>
<th>Reduce</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>A</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>A</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>a</td>
<td>a</td>
</tr>
</tbody>
</table>
```

Diagram:

- $S$ (Start)
- $A$ (Non-terminal)
- $a$ (Terminal)

Shift: $a \rightarrow A$
Reduce: $A \rightarrow a$
Reduce: $S \rightarrow A, A \rightarrow a$
Strategy for Completeness Proof

- Completeness
  - Finite progression when working backwards through rightmost derivation

  - Number of machine steps = length of derivation + # shift steps
Complexity of Completeness

- Proof
  - Count steps to get to next sentential form

- Problem
  - Need to know any other steps are not possible

- Solution
  - SLR-ness of grammar = 'if other derivations do exist, grammar is not SLR'

- Corollary
  - SLR grammars are unambiguous
**Executability**

- **Nullable**
  
  \[
  \text{nullable } g \; sl = \text{RTC } (\text{derives } g) \; sl \; []
  \]

- **Executable nullable**
  
  \[
  \text{nullableML } g \; sn \; [] = T \land \\
  \text{nullableML } g \; sn \; (TS \; x \; :: \; t) = F \land \\
  \text{nullableML } g \; sn \; (NTS \; n \; :: \; t) = \\
  \hspace{1cm} \text{if MEM (NTS } n \text{) } sn \text{ then } F \\
  \hspace{1cm} \text{else EXISTS (nullableML } g \; (NTS \; n \; :: \; sn)) \\
  \hspace{1.5cm} (\text{getRhs } n \; (\text{rules } g)) \land \\
  \hspace{3cm} \text{nullableML } g \; sn \; t
  \]

- \(N \rightarrow N \rightarrow []\), ERROR
- \(N \rightarrow []\), OK
Equivalence Proof

- $\forall g \; sn \; l. \text{nullableML} \; g \; sn \; l \implies \text{nullable} \; g \; l$

- $\forall g \; sn \; l. \text{nullable} \; g \; l \implies (sn=[]) \implies \text{nullableML} \; g \; sn \; l$

SABBC  then  S     A     B     B     B     B     C
     .     .     .     .     .     .     .
     .     .     .     .     .     .     .
[]     []     []     []     []     []     []
And in case you thought it was easy!

Note: All derivations referred to below are rightmost derivations.
The NFA $M$ defined above has the property that $E(q, \gamma)$ contains $A \rightarrow a \bullet b$, iff $A \rightarrow a \bullet b$ is valid for $\gamma$.
We must show that each item $A \rightarrow a \bullet b$ contained in $E(q, \gamma)$ is valid for $\gamma$.

If: Suppose $A \rightarrow a \bullet b$ is valid for $\gamma$. Then

$S \Rightarrow \gamma 1 A w \Rightarrow \gamma 1 a b w$,

where $\gamma 1 = \gamma$. If we can show that $E(q, \gamma 1)$ contains $A \rightarrow a \bullet b$, then by rule (3) we know that $E(q, \gamma)$ contains $A \rightarrow a \bullet b$. We therefore prove by induction on the length of above derivation that $E(q, \gamma 1)$ contains $A \rightarrow a \bullet b$.

The basis, one step, follows from rule (1). For the induction, consider the step in $S \Rightarrow \gamma 1 A w$ in which the explicitly shown $A$ was introduced. That is, write $S \Rightarrow \gamma 1 A w$ as

$S \Rightarrow \gamma 2 B x \Rightarrow \gamma 2 \gamma 3 A y 4 x \Rightarrow \gamma 2 \gamma 3 A y x$,

where $\gamma 2 \gamma 3 = \gamma 1$ and $y x = w$. Then by the inductive hypothesis applied to the derivation $S \Rightarrow \gamma 2 B x \Rightarrow \gamma 2 \gamma 3 A y 4 x$,

we know that $B \rightarrow \gamma 2 \gamma 3 A y 4$ is in $E(q, \gamma 2)$. By rule (3), $B \rightarrow \gamma 3 \bullet A y 4$ is in $E(q, \gamma 2 \gamma 3)$, and by rule (2) $A \rightarrow a \bullet b$ is in $E(q, \gamma 2 \gamma 3)$. Since $\gamma 2 \gamma 3 = \gamma 1$, we have proved the inductive hypothesis.

Rule (1) $E(q, \epsilon) = \{ S \rightarrow a \mid S \rightarrow a$ is a production $\}$
Rule (2) $E(A \rightarrow a \bullet B, \epsilon) = \{ B \rightarrow \gamma \mid B \rightarrow \gamma$ is a production $\}$
Rule (3) $E(A \rightarrow a \bullet X, \epsilon) = \{ A \rightarrow a X \bullet b \}$
Possible Improvements

- An efficient algorithm for computing DFA states
- Decidability of the parser
  - Terminates on all inputs, not just elements of language