Class 11: SLR Parsing

SI 413 - Programming Languages and Implementation

Dr. Daniel S. Roche

United States Naval Academy

Fall 2011

Simple grammar from last lecture

$S \to E$
$E \to E + T$
$E \to T$
$T \to \texttt{n}$

LR items:

$S \to \bullet\ E$	$E \to E + T\ \bullet$
$S \to E\ \bullet$	$E \to \bullet\ T$
$E \to \bullet\ E + T$	$E \to T\ \bullet$
$E \to E\ \bullet + T$	$T \to \bullet\ \texttt{n}$
$E \to E + \bullet\ T$	$T \to \texttt{n}\ \bullet$

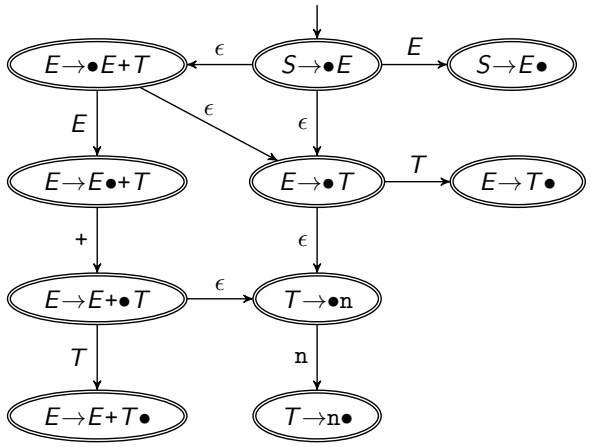
Pieces of the CFSM

The CSFM (Characteristic Finite State Machine) is a FA representing the *transitions* between the LR item "states".

There are two types of transitions:

- **Shift**: consume a terminal *or non-terminal* symbol and move the • to the right by one.
  Example: $T \to \bullet\texttt{n}$ —n→ $T \to \texttt{n}\bullet$
- **Closure**: If the • is to the left of a non-terminal, we have an ϵ-transition to any production of that non-terminal with the • all the way to the left.
  Example: $E \to E + \bullet\ T$ —ϵ→ $T \to \bullet\texttt{n}$

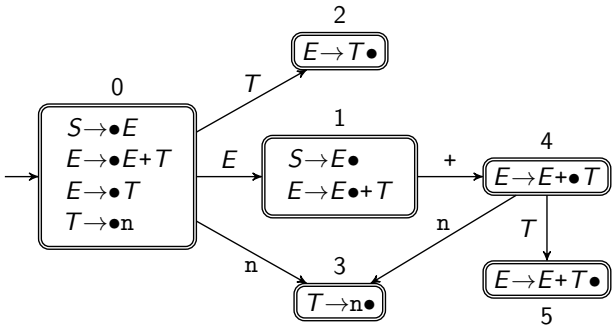
Nondeterministic CFSM for example grammar



CFSM Properties

- Observe that every state is accepting.
- This is an NDFA that accepts *valid stack contents*.
- The "trap states" correspond to a *reduce* operation:
  Replace r.h.s. on stack with the l.h.s. non-terminal.
- We can simulate an LR parse by following the CFSM on the current stack symbols AND un-parsed tokens, then starting over after every reduce operation changes the stack.
- We can turn this into a DFA just by combining states.

Deterministic CFSM for example grammar



- Every state is labelled with a number.
- Labels are pushed on the stack along with symbols.
- After a reduce, go back to the state label left at the top of the stack.

SLR

Parsing this way using a (deterministic) CFSM is called *SLR Parsing*.

Following an edge in the CFSM means shifting;
coming to a rule that ends in • means reducing.

SLR(k) means SLR with k tokens of look-ahead.
The previous grammar was SLR(0); i.e., no look-ahead required.

When might we need look-ahead?

Example Grammar 2

Consider the following grammar:

$S \rightarrow W\ W$
$W \rightarrow$ `a`
$W \rightarrow$ `ab`

Draw the CSFM for this grammar.
What is the problem?

Example Grammar 3

Consider the following grammar:

$S \rightarrow W$ `b`
$W \rightarrow$ `a`
$W \rightarrow X$ `a`
$X \rightarrow$ `a`

Draw the CSFM for this grammar.
What is the problem?

SLR(1)

SLR(1) parsers handle conflicts by using one token of look-ahead:

- If the next token is an outgoing edge label of that state, shift and move on.
- If the next token is in the *follow set* of a non-terminal *that we can reduce to*, then do that reduction.

Of course, there may still be conflicts, in which case the grammar is not SLR(1). More look-ahead may be needed.