Compiler Construction

Lecture 6: SLR(1) and LR(1)

Jeremy Yallop

jeremy.yallop@cl.cam.ac.uk

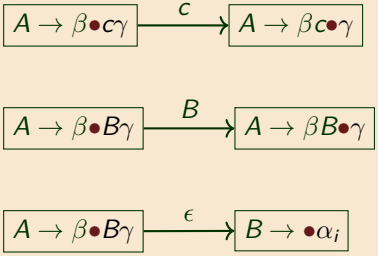
Lent 2023

Configurations $\$\alpha, w\$$ with **actions**:

$$\$\alpha, xw\$ \xrightarrow{\text{shift } x} \$\alpha x, w\$$$

$$\$\alpha\beta, w\$ \xrightarrow{\text{reduce } A\to\beta} \$\alpha A, w\$$$

Items $A \to \beta \bullet \gamma$ with **transitions**:



A **parsing algorithm**:

$c :=$ NextToken()

while true:

$\alpha :=$ the stack

if $A \to \beta \bullet c\gamma \in \delta_G(q_0, \alpha)$
then SHIFT c; $c :=$ NextToken()

if $A \to \beta \bullet \in \delta_G(q_0, \alpha)$
then REDUCE via $A \to \beta$

if $S \to \beta \bullet \in \delta_G(q_0, \alpha)$
then ACCEPT (if no more input)

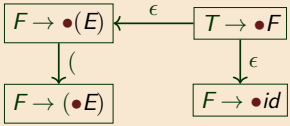
if none of the above
then ERROR

non-deterministic

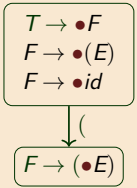
Making the algorithm deterministic

Two sources of **nondeterminism**:

1. The **NFA**



Solution: convert to a **DFA**



2. **Conflicts**

$\$\alpha E + T, x\$$ $\xrightarrow{\text{reduce } E \to E+T}$

$\$\alpha E + T, x\$$ $\xrightarrow{\text{reduce } E \to T}$

$\$\alpha E + T, x\$$ $\xrightarrow{\text{shift } x}$

Solution: make a **deterministic choice**

using the **input** (lookahead)

using the **grammar** (FIRST and FOLLOW)

Two approaches: SLR(1) and LR(1)

The DFA

The easy part: NFA → DFA

Grammar G_2'

$S \to E$
$E \to E + T \mid T$
$T \to T * F \mid F$
$F \to (E) \mid \text{id}$

$$\text{DFA start state} = \epsilon\text{−closure}(\{S \to \bullet E\}) = \begin{array}{l} S \to \bullet E \\ E \to \bullet E + T \\ E \to \bullet T \\ T \to \bullet T * F \\ T \to \bullet F \\ F \to \bullet (E) \\ F \to \bullet \text{id} \end{array}$$

Transition function:

$$\delta(I, X) = \epsilon\text{−closure}(\{A \to \alpha X \bullet \beta \mid A \to \alpha \bullet X \beta \in I\})$$

*(**NB**: this is just the powerset/subset construction for converting NFAs to DFAs.)*

Some DFA transitions for grammar G_2

Grammar G_2'

$S \to E$
$E \to E + T \mid T$
$T \to T * F \mid F$
$F \to (E) \mid \text{id}$

$F \to (\bullet E)$

Some DFA transitions for grammar G_2

Grammar G_2'

$S \to E$
$E \to E + T \mid T$
$T \to T * F \mid F$
$F \to (E) \mid \text{id}$

$F \to (\bullet E)$
$E \to \bullet E + T$
$E \to \bullet T$

Some DFA transitions for grammar G_2

Grammar G_2'

$S \to E$
$E \to E + T \mid T$
$T \to T * F \mid F$
$F \to (E) \mid \text{id}$

$F \to (\bullet E)$
$E \to \bullet E + T$
$E \to \bullet T$
$T \to \bullet T * F$
$T \to \bullet F$

Some DFA transitions for grammar G_2

Grammar G_2'

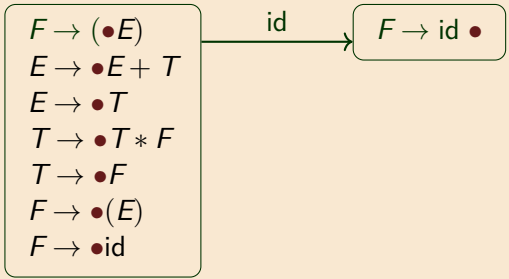
$S \to E$
$E \to E + T \mid T$
$T \to T * F \mid F$
$F \to (E) \mid \text{id}$

$F \to (\bullet E)$
$E \to \bullet E + T$
$E \to \bullet T$
$T \to \bullet T * F$
$T \to \bullet F$
$F \to \bullet (E)$
$F \to \bullet \text{id}$

Some DFA transitions for grammar G_2

Grammar G_2'

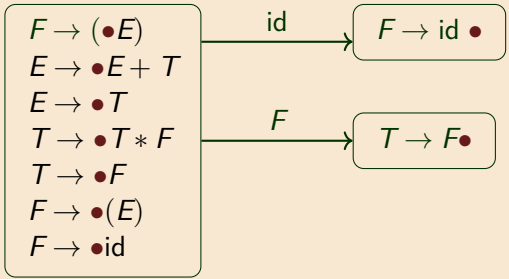
$S \to E$
$E \to E + T \mid T$
$T \to T * F \mid F$
$F \to (E) \mid \text{id}$



Some DFA transitions for grammar G_2

Grammar G_2'

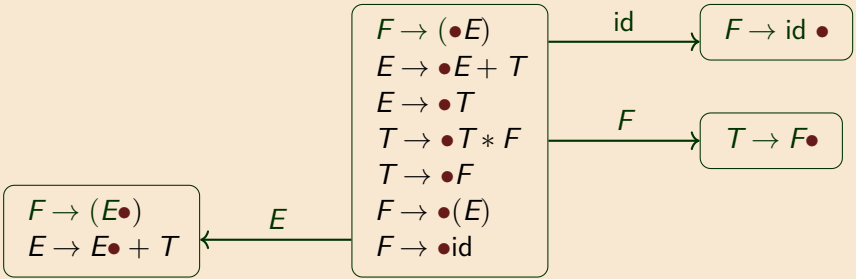
$S \to E$
$E \to E + T \mid T$
$T \to T * F \mid F$
$F \to (E) \mid \text{id}$



Some DFA transitions for grammar G_2

Grammar G_2'

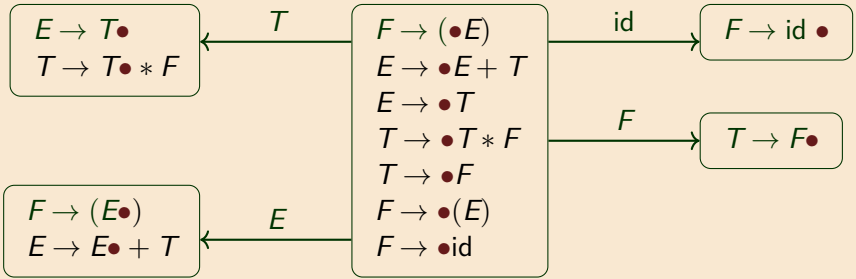
$S \to E$
$E \to E + T \mid T$
$T \to T * F \mid F$
$F \to (E) \mid \text{id}$



Some DFA transitions for grammar G_2

Grammar G_2'

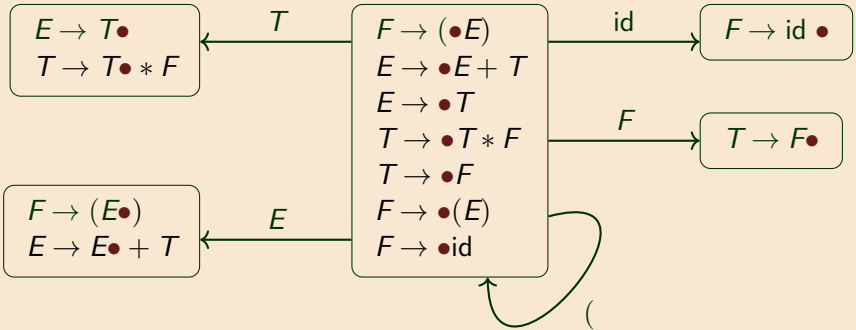
$S \to E$
$E \to E + T \mid T$
$T \to T * F \mid F$
$F \to (E) \mid \text{id}$



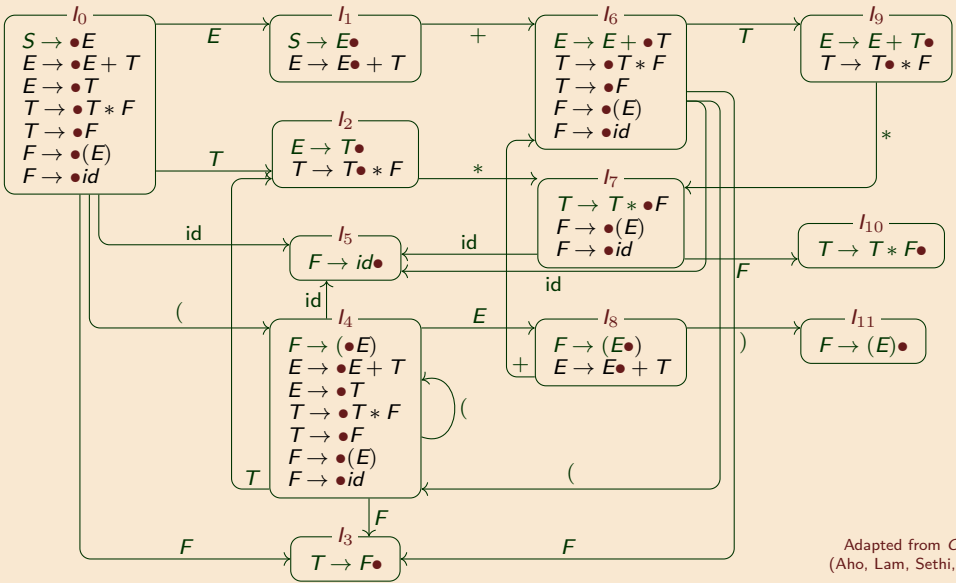
Some DFA transitions for grammar G_2

Grammar G_2'

$S \to E$
$E \to E + T \mid T$
$T \to T * F \mid F$
$F \to (E) \mid \text{id}$



Full DFA for the stack language of G_2



Adapted from *Compilers* (Aho, Lam, Sethi, Ullman)

SLR(1)

The state I_2 has a **shift/reduce conflict**.

I_2

$E \to T\bullet$

$T \to T\bullet * F$

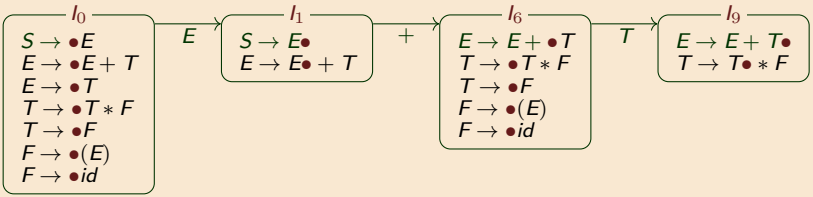
The **S**imple **LR(1)** approach resolves the conflict using the next token c:

shift if $c = *$

reduce with $E \to T$ only if $c \in \text{FOLLOW}(E) = \{(, +, \$\}$.

Recall: when the stack contains a, the parser is in state $\delta(I_0, a)$. For example,

$$\delta(I_0, E + T) \quad = \quad I_9$$



Let I be the current state and c the next token. Then:

1. When $A \to \beta \bullet c\gamma \in I$ then **shift** t onto stack
2. When $A \to \beta \bullet \in I$ and $c \in \text{FOLLOW}(A)$ then **reduce** with $A \to \beta$

Replay parsing of $(x + y)$ using SLR(1) actions

stack,	input	state	action	reason
$,	$(x + y)$$	I_0	shift (	

Replay parsing of $(x + y)$ using SLR(1) actions

stack,	input	state	action	reason
$\$,$	$(x+y)\$$	I_0	shift (	$F \to \bullet(E) \in \delta(I_0, \epsilon) = I_0$

Replay parsing of $(x + y)$ using SLR(1) actions

stack,	input	state	action	reason
$\$,$	$(x+y)\$$	I_0	shift (	$F \to \bullet(E) \in \delta(I_0, \epsilon) = I_0$
$\$($,	$x+y)\$$	I_4	shift x	$F \to \bullet\mathsf{id} \in \delta(I_0, () = I_4$

Replay parsing of $(x+y)$ using SLR(1) actions

stack,	input	state	action	reason
$\$,$	$(x+y)\$$	I_0	shift (	$F \to \bullet(E) \in \delta(I_0, \epsilon) = I_0$
$\$($,	$x+y)\$$	I_4	shift x	$F \to \bullet\mathsf{id} \in \delta(I_0, () = I_4$
$\$(x$,	$+y)\$$	I_5	reduce $F \to \mathsf{id}$	"+" $\in \textsc{follow}(F)$

Replay parsing of $(x+y)$ using SLR(1) actions

stack,	input	state	action	reason
$\$,$	$(x+y)\$$	I_0	shift (	$F \to \bullet(E) \in \delta(I_0, \epsilon) = I_0$
$\$($,	$x+y)\$$	I_4	shift x	$F \to \bullet\mathsf{id} \in \delta(I_0, () = I_4$
$\$(x$,	$+y)\$$	I_5	reduce $F \to \mathsf{id}$	"+" $\in \textsc{follow}(F)$
$\$(F$,	$+y)\$$	I_3	reduce $T \to F$	"+" $\in \textsc{follow}(T)$

Replay parsing of $(x + y)$ using SLR(1) actions

stack,	input	state	action	reason
$\$,$	$(x+y)\$$	I_0	shift (	$F \to \bullet(E) \in \delta(I_0, \epsilon) = I_0$
$\$($,	$x+y)\$$	I_4	shift x	$F \to \bullet\mathsf{id} \in \delta(I_0, () = I_4$
$\$(x$,	$+y)\$$	I_5	reduce $F \to \mathsf{id}$	"+" $\in \textsc{follow}(F)$
$\$(F$,	$+y)\$$	I_3	reduce $T \to F$	"+" $\in \textsc{follow}(T)$
$\$(T$,	$+y)\$$	I_2	reduce $E \to T$	"+" $\in \textsc{follow}(E)$

Replay parsing of $(x + y)$ using SLR(1) actions

stack,	input	state	action	reason
$\$,$	$(x+y)\$$	I_0	shift (	$F \to \bullet(E) \in \delta(I_0, \epsilon) = I_0$
$\$($,	$x+y)\$$	I_4	shift x	$F \to \bullet\mathsf{id} \in \delta(I_0, () = I_4$
$\$(x$,	$+y)\$$	I_5	reduce $F \to \mathsf{id}$	"+" $\in$ FOLLOW(F)
$\$(F$,	$+y)\$$	I_3	reduce $T \to F$	"+" $\in$ FOLLOW(T)
$\$(T$,	$+y)\$$	I_2	reduce $E \to T$	"+" $\in$ FOLLOW(E)
$\$(E$,	$+y)\$$	I_8	shift +	$E \to E\bullet + T \in \delta(I_0, \mathtt{E}) = I_8$

Replay parsing of $(x + y)$ using SLR(1) actions

stack,	input	state	action	reason
$\$,$	$(x+y)\$$	I_0	shift (	$F \to \bullet(E) \in \delta(I_0, \epsilon) = I_0$
$\$($,	$x+y)\$$	I_4	shift x	$F \to \bullet\mathsf{id} \in \delta(I_0, \texttt{(}) = I_4$
$\$(x$,	$+y)\$$	I_5	reduce $F \to \mathsf{id}$	$\texttt{"+"} \in \textsc{follow}(F)$
$\$(F$,	$+y)\$$	I_3	reduce $T \to F$	$\texttt{"+"} \in \textsc{follow}(T)$
$\$(T$,	$+y)\$$	I_2	reduce $E \to T$	$\texttt{"+"} \in \textsc{follow}(E)$
$\$(E$,	$+y)\$$	I_8	shift $+$	$E \to E\bullet + T \in \delta(I_0, \texttt{(E)}) = I_8$
$\$(E+$,	$y)\$$	I_6	shift y	$F \to \bullet\mathsf{id} \in \delta(I_0, \texttt{(E+)}) = I_6$

Replay parsing of $(x + y)$ using SLR(1) actions

stack,	input	state	action	reason
$\$,$	$(x+y)\$$	I_0	shift (	$F \to \bullet(E) \in \delta(I_0, \epsilon) = I_0$
$\$($,	$x+y)\$$	I_4	shift x	$F \to \bullet\mathsf{id} \in \delta(I_0, ()) = I_4$
$\$(x,$	$+y)\$$	I_5	reduce $F \to \mathsf{id}$	$\text{"+"} \in \textsc{follow}(F)$
$\$(F,$	$+y)\$$	I_3	reduce $T \to F$	$\text{"+"} \in \textsc{follow}(T)$
$\$(T,$	$+y)\$$	I_2	reduce $E \to T$	$\text{"+"} \in \textsc{follow}(E)$
$\$(E,$	$+y)\$$	I_8	shift +	$E \to E\bullet + T \in \delta(I_0, \mathtt{(E)}) = I_8$
$\$(E+,$	$y)\$$	I_6	shift y	$F \to \bullet\mathsf{id} \in \delta(I_0, \mathtt{(E+)}) = I_6$
$\$(E+y,$	$)\$$	I_5	reduce $F \to \mathsf{id}$	$\text{")"} \in \textsc{follow}(F)$

Replay parsing of $(x + y)$ using SLR(1) actions

stack,	input	state	action	reason
$\$,$	$(x+y)\$$	I_0	shift (	$F \to \bullet(E) \in \delta(I_0, \epsilon) = I_0$
$\$($,	$x+y)\$$	I_4	shift x	$F \to \bullet\mathsf{id} \in \delta(I_0, () = I_4$
$\$(x,$	$+y)\$$	I_5	reduce $F \to \mathsf{id}$	"+" $\in$ FOLLOW(F)
$\$(F,$	$+y)\$$	I_3	reduce $T \to F$	"+" $\in$ FOLLOW(T)
$\$(T,$	$+y)\$$	I_2	reduce $E \to T$	"+" $\in$ FOLLOW(E)
$\$(E,$	$+y)\$$	I_8	shift +	$E \to E\bullet + T \in \delta(I_0, \mathtt{E}) = I_8$
$\$(E+,$	$y)\$$	I_6	shift y	$F \to \bullet\mathsf{id} \in \delta(I_0, \mathtt{E+}) = I_6$
$\$(E+y,$	$)\$$	I_5	reduce $F \to \mathsf{id}$	")" $\in$ FOLLOW(F)
$\$(E+F,$	$)\$$	I_3	reduce $T \to F$	")" $\in$ FOLLOW(T)

Replay parsing of $(x + y)$ using SLR(1) actions

stack,	input	state	action	reason
$\$,$	$(x+y)\$$	I_0	shift (	$F \to \bullet(E) \in \delta(I_0, \epsilon) = I_0$
$\$($,	$x+y)\$$	I_4	shift x	$F \to \bullet\mathsf{id} \in \delta(I_0, ()) = I_4$
$\$(x,$	$+y)\$$	I_5	reduce $F \to \mathsf{id}$	$"+" \in \textsc{follow}(F)$
$\$(F,$	$+y)\$$	I_3	reduce $T \to F$	$"+" \in \textsc{follow}(T)$
$\$(T,$	$+y)\$$	I_2	reduce $E \to T$	$"+" \in \textsc{follow}(E)$
$\$(E,$	$+y)\$$	I_8	shift $+$	$E \to E\bullet + T \in \delta(I_0, \mathtt{(E)}) = I_8$
$\$(E+,$	$y)\$$	I_6	shift y	$F \to \bullet\mathsf{id} \in \delta(I_0, \mathtt{(E+)}) = I_6$
$\$(E+y,$	$)\$$	I_5	reduce $F \to \mathsf{id}$	$")" \in \textsc{follow}(F)$
$\$(E+F,$	$)\$$	I_3	reduce $T \to F$	$")" \in \textsc{follow}(T)$
$\$(E+T,$	$)\$$	I_9	reduce $E \to E+T$	$")" \in \textsc{follow}(E)$

Replay parsing of $(x + y)$ using SLR(1) actions

stack,	input	state	action	reason
$\$,$	$(x+y)\$$	I_0	shift (	$F \to \bullet(E) \in \delta(I_0, \epsilon) = I_0$
$\$($,	$x+y)\$$	I_4	shift x	$F \to \bullet\mathsf{id} \in \delta(I_0, ()) = I_4$
$\$(x,$	$+y)\$$	I_5	reduce $F \to \mathsf{id}$	"+" $\in \textsc{follow}(F)$
$\$(F,$	$+y)\$$	I_3	reduce $T \to F$	"+" $\in \textsc{follow}(T)$
$\$(T,$	$+y)\$$	I_2	reduce $E \to T$	"+" $\in \textsc{follow}(E)$
$\$(E,$	$+y)\$$	I_8	shift +	$E \to E\bullet + T \in \delta(I_0, \mathtt{(E)}) = I_8$
$\$(E+,$	$y)\$$	I_6	shift y	$F \to \bullet\mathsf{id} \in \delta(I_0, \mathtt{(E+)}) = I_6$
$\$(E+y,$	$)\$$	I_5	reduce $F \to \mathsf{id}$	")" $\in \textsc{follow}(F)$
$\$(E+F,$	$)\$$	I_3	reduce $T \to F$	")" $\in \textsc{follow}(T)$
$\$(E+T,$	$)\$$	I_9	reduce $E \to E+T$	")" $\in \textsc{follow}(E)$
$\$(E,$	$)\$$	I_8	shift)	$E \to (E\bullet) \in \delta(I_0, \mathtt{(E)}) = I_8$

Replay parsing of $(x+y)$ using SLR(1) actions

stack,	input	state	action	reason
$\$,$	$(x+y)\$$	I_0	shift (	$F \to \bullet(E) \in \delta(I_0, \epsilon) = I_0$
$\$(,$	$x+y)\$$	I_4	shift x	$F \to \bullet\mathsf{id} \in \delta(I_0, () = I_4$
$\$(x,$	$+y)\$$	I_5	reduce $F \to \mathsf{id}$	$"+" \in \textsc{follow}(F)$
$\$(F,$	$+y)\$$	I_3	reduce $T \to F$	$"+" \in \textsc{follow}(T)$
$\$(T,$	$+y)\$$	I_2	reduce $E \to T$	$"+" \in \textsc{follow}(E)$
$\$(E,$	$+y)\$$	I_8	shift +	$E \to E\bullet + T \in \delta(I_0, \mathtt{(E)}) = I_8$
$\$(E+,$	$y)\$$	I_6	shift y	$F \to \bullet\mathsf{id} \in \delta(I_0, \mathtt{(E+)}) = I_6$
$\$(E+y,$	$)\$$	I_5	reduce $F \to \mathsf{id}$	$")" \in \textsc{follow}(F)$
$\$(E+F,$	$)\$$	I_3	reduce $T \to F$	$")" \in \textsc{follow}(T)$
$\$(E+T,$	$)\$$	I_9	reduce $E \to E+T$	$")" \in \textsc{follow}(E)$
$\$(E,$	$)\$$	I_8	shift)	$E \to (E\bullet) \in \delta(I_0, \mathtt{(E)}) = I_8$
$\$(E),$	$\$$	I_{11}	reduce $F \to (E)$	$"\$" \in \textsc{follow}(F)$

Replay parsing of $(x + y)$ using SLR(1) actions

stack,	input	state	action	reason
$\$,$	$(x+y)\$$	I_0	shift (	$F \to \bullet(E) \in \delta(I_0, \epsilon) = I_0$
$\$($,	$x+y)\$$	I_4	shift x	$F \to \bullet\mathsf{id} \in \delta(I_0, () = I_4$
$\$(x,$	$+y)\$$	I_5	reduce $F \to \mathsf{id}$	$"+" \in \textsc{follow}(F)$
$\$(F,$	$+y)\$$	I_3	reduce $T \to F$	$"+" \in \textsc{follow}(T)$
$\$(T,$	$+y)\$$	I_2	reduce $E \to T$	$"+" \in \textsc{follow}(E)$
$\$(E,$	$+y)\$$	I_8	shift +	$E \to E\bullet + T \in \delta(I_0, \mathtt{(E)}) = I_8$
$\$(E+,$	$y)\$$	I_6	shift y	$F \to \bullet\mathsf{id} \in \delta(I_0, \mathtt{(E+)}) = I_6$
$\$(E+y,$	$)\$$	I_5	reduce $F \to \mathsf{id}$	$")" \in \textsc{follow}(F)$
$\$(E+F,$	$)\$$	I_3	reduce $T \to F$	$")" \in \textsc{follow}(T)$
$\$(E+T,$	$)\$$	I_9	reduce $E \to E+T$	$")" \in \textsc{follow}(E)$
$\$(E,$	$)\$$	I_8	shift)	$E \to (E\bullet) \in \delta(I_0, \mathtt{(E)}) = I_8$
$\$(E),$	$\$$	I_{11}	reduce $F \to (E)$	$"\$" \in \textsc{follow}(F)$
$\$F,$	$\$$	I_3	reduce $T \to F$	$"\$" \in \textsc{follow}(T)$

Replay parsing of $(x+y)$ using SLR(1) actions

stack,	input	state	action	reason
$\$,$	$(x+y)\$$	I_0	shift (	$F \to \bullet(E) \in \delta(I_0, \epsilon) = I_0$
$\$($,	$x+y)\$$	I_4	shift x	$F \to \bullet\mathsf{id} \in \delta(I_0, () = I_4$
$\$(x,$	$+y)\$$	I_5	reduce $F \to \mathsf{id}$	$"+" \in \textsc{follow}(F)$
$\$(F,$	$+y)\$$	I_3	reduce $T \to F$	$"+" \in \textsc{follow}(T)$
$\$(T,$	$+y)\$$	I_2	reduce $E \to T$	$"+" \in \textsc{follow}(E)$
$\$(E,$	$+y)\$$	I_8	shift +	$E \to E\bullet + T \in \delta(I_0, \mathtt{(E)}) = I_8$
$\$(E+,$	$y)\$$	I_6	shift y	$F \to \bullet\mathsf{id} \in \delta(I_0, \mathtt{(E+)}) = I_6$
$\$(E+y,$	$)\$$	I_5	reduce $F \to \mathsf{id}$	$")" \in \textsc{follow}(F)$
$\$(E+F,$	$)\$$	I_3	reduce $T \to F$	$")" \in \textsc{follow}(T)$
$\$(E+T,$	$)\$$	I_9	reduce $E \to E+T$	$")" \in \textsc{follow}(E)$
$\$(E,$	$)\$$	I_8	shift)	$E \to (E\bullet) \in \delta(I_0, \mathtt{(E)}) = I_8$
$\$(E),$	$\$$	I_{11}	reduce $F \to (E)$	$"\$" \in \textsc{follow}(F)$
$\$F,$	$\$$	I_3	reduce $T \to F$	$"\$" \in \textsc{follow}(T)$
$\$T,$	$\$$	I_2	reduce $F \to E$	$"\$" \in \textsc{follow}(E)$

Replay parsing of $(x + y)$ using SLR(1) actions

stack,	input	state	action	reason
$\$,$	$(x+y)\$$	I_0	shift (	$F \to \bullet(E) \in \delta(I_0, \epsilon) = I_0$
$\$($,	$x+y)\$$	I_4	shift x	$F \to \bullet\mathsf{id} \in \delta(I_0, () = I_4$
$\$(x$,	$+y)\$$	I_5	reduce $F \to \mathsf{id}$	"+" $\in \textsc{follow}(F)$
$\$(F$,	$+y)\$$	I_3	reduce $T \to F$	"+" $\in \textsc{follow}(T)$
$\$(T$,	$+y)\$$	I_2	reduce $E \to T$	"+" $\in \textsc{follow}(E)$
$\$(E$,	$+y)\$$	I_8	shift +	$E \to E\bullet + T \in \delta(I_0, \texttt{(E)} = I_8$
$\$(E+$,	$y)\$$	I_6	shift y	$F \to \bullet\mathsf{id} \in \delta(I_0, \texttt{(E+)} = I_6$
$\$(E+y$,	$)\$$	I_5	reduce $F \to \mathsf{id}$	")" $\in \textsc{follow}(F)$
$\$(E+F$,	$)\$$	I_3	reduce $T \to F$	")" $\in \textsc{follow}(T)$
$\$(E+T$,	$)\$$	I_9	reduce $E \to E+T$	")" $\in \textsc{follow}(E)$
$\$(E$,	$)\$$	I_8	shift)	$E \to (E\bullet) \in \delta(I_0, \texttt{(E)} = I_8$
$\$(E)$,	$\$$	I_{11}	reduce $F \to (E)$	"$") $\in \textsc{follow}(F)$
$\$F$,	$\$$	I_3	reduce $T \to F$	"$") $\in \textsc{follow}(T)$
$\$T$,	$\$$	I_2	reduce $F \to E$	"$") $\in \textsc{follow}(E)$
$\$E$,	$\$$	I_1	reduce $S \to E$	"$") $\in \textsc{follow}(S)$

Parsing with states

Idea: don't restart the DFA at every step

Previous approach

Maintain a **stack of symbols**

$\$ \; (\; E \; + \; \text{id}$

At each step:

Use the **full stack** to find items

New approach

Maintain a **stack of states**

$0 \; 4 \; 1 \; 6 \; 5$

At each step:

Use the **top of the stack** to find actions

LR parsing with DFA states on the stack

```
tok := NextToken()
while true:
  state := TopStackState()
  if ACTION[state, tok] = SHIFT state
    then push state
         tok := NextToken()
  else if ACTION[state, tok] = REDUCE A → β
      then pop |β| states
           push GOTO[TopStackState(), A]
  else if ACTION[state, tok] = ACCEPT
    then accept and exit
  else ERROR
```

Constructing ACTION and GOTO for SLR(1)

If $A \to \alpha \bullet a\beta \in I_i$ and $\delta(I_i, a) = I_j$
then $\text{ACTION}[i, a] = \text{SHIFT } j$.

If $A \to \alpha \bullet \in I_i$ and $A \neq S$
then for all $a \in \text{FOLLOW}(A)$,
$\text{ACTION}[i, a] = \text{REDUCE } A \to \alpha$

If $S \to \alpha \bullet \in I_i$
then $\text{ACTION}[i, \$] = \text{ACCEPT}$

If $\delta(I_i, A) = I_j$
then $\text{GOTO}[i, A] = j$

(ACTION resolves conflicts; GOTO records nonterminal transitions $I_i \xrightarrow{A} I_j$)

ACTION and GOTO for G_2'

State	Action						Goto		
	id	+	*	(	)	$	*E*	*T*	*F*
0	*s*5			*s*4			1	2	3
1		*s*6				*acc*			
2		*r*2	*s*7		*r*2	*r*2			
3		*r*4	*r*4		*r*4	*r*4			
4	*s*5			*s*4			8	2	3
5		*r*6	*r*6		*r*6	*r*6			
6	*s*5			*s*4				9	3
7	*s*5			*s*4					10
8		*s*6			*s*11				
9		*r*1	*s*7		*r*1	*r*1			
10		*r*3	*r*3		*r*3	*r*3			
11		*r*5	*r*5		*r*5	*r*5			

Adapted from *Compilers* (Aho, Lam, Sethi, Ullman)

Example parse

Stack	Symbols	Input	Action
0		id $*$ id $+$ id\$	shift

Stack	Symbols	Input	Action	
0		id $*$ id $+$ id\$	shift	
$0\ 5$	id	$*$ id $+$ id\$	reduce $F \to$ id	$\text{GOTO}[0, F] = 3$

Adapted from *Compilers*
(Aho, Lam, Sethi, Ullman)

Example parse

Stack	Symbols	Input	Action	
0		id $*$ id $+$ id\$	shift	
0 5	id	$*$ id $+$ id\$	reduce $F \to$ id	$\textsc{goto}[0, F] = 3$
0 3	F	$*$ id $+$ id\$	reduce $T \to F$	$\textsc{goto}[0, T] = 2$

Adapted from *Compilers*
(Aho, Lam, Sethi, Ullman)

Example parse

Stack	Symbols	Input	Action	
0		id $*$ id $+$ id\$	shift	
0 5	id	$*$ id $+$ id\$	reduce $F \to$ id	$\textsc{goto}[0, F] = 3$
0 3	F	$*$ id $+$ id\$	reduce $T \to F$	$\textsc{goto}[0, T] = 2$
0 2	T	$*$ id $+$ id\$	shift	

Adapted from *Compilers*
(Aho, Lam, Sethi, Ullman)

Example parse

Stack	Symbols	Input	Action	
0		id $*$ id $+$ id\$	shift	
0 5	id	$*$ id $+$ id\$	reduce $F \to$ id	$\text{GOTO}[0, F] = 3$
0 3	F	$*$ id $+$ id\$	reduce $T \to F$	$\text{GOTO}[0, T] = 2$
0 2	T	$*$ id $+$ id\$	shift	
0 2 7	$T*$	id $+$ id\$	shift	

Example parse

Stack	Symbols	Input	Action	
0		id $*$ id $+$ id\$	shift	
0 5	id	$*$ id $+$ id\$	reduce $F \to$ id	$\text{GOTO}[0, F] = 3$
0 3	F	$*$ id $+$ id\$	reduce $T \to F$	$\text{GOTO}[0, T] = 2$
0 2	T	$*$ id $+$ id\$	shift	
0 2 7	$T*$	id $+$ id\$	shift	
0 2 7 5	$T*$ id	$+$ id\$	reduce $F \to$ id	$\text{GOTO}[7, F] = 10$

Adapted from *Compilers*
(Aho, Lam, Sethi, Ullman)

Example parse

Stack	Symbols	Input	Action	
0		id $*$ id $+$ id\$	shift	
0 5	id	$*$ id $+$ id\$	reduce $F \to$ id	$\text{GOTO}[0, F] = 3$
0 3	F	$*$ id $+$ id\$	reduce $T \to F$	$\text{GOTO}[0, T] = 2$
0 2	T	$*$ id $+$ id\$	shift	
0 2 7	$T*$	id $+$ id\$	shift	
0 2 7 5	$T*$ id	$+$ id\$	reduce $F \to$ id	$\text{GOTO}[7, F] = 10$
0 2 7 10	$T*F$	$+$ id\$	reduce $T \to T*F$	$\text{GOTO}[0, T] = 2$

Adapted from *Compilers*
(Aho, Lam, Sethi, Ullman)

Example parse

Stack	Symbols	Input	Action	
0		id $*$ id $+$ id\$	shift	
0 5	id	$*$ id $+$ id\$	reduce $F \to$ id	GOTO$[0, F] = 3$
0 3	F	$*$ id $+$ id\$	reduce $T \to F$	GOTO$[0, T] = 2$
0 2	T	$*$ id $+$ id\$	shift	
0 2 7	$T*$	id $+$ id\$	shift	
0 2 7 5	$T*$ id	$+$ id\$	reduce $F \to$ id	GOTO$[7, F] = 10$
0 2 7 10	$T*F$	$+$ id\$	reduce $T \to T*F$	GOTO$[0, T] = 2$
0 2	T	$+$ id\$	reduce $E \to T$	GOTO$[0, E] = 1$

Adapted from *Compilers*
(Aho, Lam, Sethi, Ullman)

Example parse

Stack	Symbols	Input	Action	
0		id $*$ id $+$ id\$	shift	
0 5	id	$*$ id $+$ id\$	reduce $F \to$ id	GOTO$[0, F] = 3$
0 3	F	$*$ id $+$ id\$	reduce $T \to F$	GOTO$[0, T] = 2$
0 2	T	$*$ id $+$ id\$	shift	
0 2 7	$T*$	id $+$ id\$	shift	
0 2 7 5	$T*$ id	$+$ id\$	reduce $F \to$ id	GOTO$[7, F] = 10$
0 2 7 10	$T*F$	$+$ id\$	reduce $T \to T*F$	GOTO$[0, T] = 2$
0 2	T	$+$ id\$	reduce $E \to T$	GOTO$[0, E] = 1$
0 1	E	$+$ id\$	shift	

Adapted from *Compilers* (Aho, Lam, Sethi, Ullman)

Example parse

Stack	Symbols	Input	Action	
0		id $*$ id $+$ id\$	shift	
0 5	id	$*$ id $+$ id\$	reduce $F \to$ id	GOTO$[0, F] = 3$
0 3	F	$*$ id $+$ id\$	reduce $T \to F$	GOTO$[0, T] = 2$
0 2	T	$*$ id $+$ id\$	shift	
0 2 7	$T*$	id $+$ id\$	shift	
0 2 7 5	$T*$ id	$+$ id\$	reduce $F \to$ id	GOTO$[7, F] = 10$
0 2 7 10	$T*F$	$+$ id\$	reduce $T \to T*F$	GOTO$[0, T] = 2$
0 2	T	$+$ id\$	reduce $E \to T$	GOTO$[0, E] = 1$
0 1	E	$+$ id\$	shift	
0 1 6	$E+$	id\$	shift	

Adapted from *Compilers* (Aho, Lam, Sethi, Ullman)

Example parse

Stack	Symbols	Input	Action	
0		id $*$ id $+$ id\$	shift	
0 5	id	$*$ id $+$ id\$	reduce $F \to$ id	$\text{GOTO}[0, F] = 3$
0 3	F	$*$ id $+$ id\$	reduce $T \to F$	$\text{GOTO}[0, T] = 2$
0 2	T	$*$ id $+$ id\$	shift	
0 2 7	$T*$	id $+$ id\$	shift	
0 2 7 5	$T*$ id	$+$ id\$	reduce $F \to$ id	$\text{GOTO}[7, F] = 10$
0 2 7 10	$T*F$	$+$ id\$	reduce $T \to T*F$	$\text{GOTO}[0, T] = 2$
0 2	T	$+$ id\$	reduce $E \to T$	$\text{GOTO}[0, E] = 1$
0 1	E	$+$ id\$	shift	
0 1 6	$E+$	id\$	shift	
0 1 6 5	$E+$ id	\$	reduce $F \to$ id	$\text{GOTO}[6, F] = 3$

Adapted from *Compilers* (Aho, Lam, Sethi, Ullman)

Example parse

Stack	Symbols	Input	Action	
0		id $*$ id $+$ id\$	shift	
0 5	id	$*$ id $+$ id\$	reduce $F \to$ id	$\text{GOTO}[0, F] = 3$
0 3	F	$*$ id $+$ id\$	reduce $T \to F$	$\text{GOTO}[0, T] = 2$
0 2	T	$*$ id $+$ id\$	shift	
0 2 7	$T*$	id $+$ id\$	shift	
0 2 7 5	$T*$ id	$+$ id\$	reduce $F \to$ id	$\text{GOTO}[7, F] = 10$
0 2 7 10	$T*F$	$+$ id\$	reduce $T \to T*F$	$\text{GOTO}[0, T] = 2$
0 2	T	$+$ id\$	reduce $E \to T$	$\text{GOTO}[0, E] = 1$
0 1	E	$+$ id\$	shift	
0 1 6	$E+$	id\$	shift	
0 1 6 5	$E+$ id	\$	reduce $F \to$ id	$\text{GOTO}[6, F] = 3$
0 1 6 3	$E+F$	\$	reduce $T \to F$	$\text{GOTO}[6, T] = 9$

Adapted from *Compilers* (Aho, Lam, Sethi, Ullman)

Example parse

Stack	Symbols	Input	Action	
0		id $*$ id $+$ id\$	shift	
0 5	id	$*$ id $+$ id\$	reduce $F \to$ id	GOTO$[0, F] = 3$
0 3	F	$*$ id $+$ id\$	reduce $T \to F$	GOTO$[0, T] = 2$
0 2	T	$*$ id $+$ id\$	shift	
0 2 7	$T*$	id $+$ id\$	shift	
0 2 7 5	$T*$ id	$+$ id\$	reduce $F \to$ id	GOTO$[7, F] = 10$
0 2 7 10	$T*F$	$+$ id\$	reduce $T \to T*F$	GOTO$[0, T] = 2$
0 2	T	$+$ id\$	reduce $E \to T$	GOTO$[0, E] = 1$
0 1	E	$+$ id\$	shift	
0 1 6	$E+$	id\$	shift	
0 1 6 5	$E+$ id	\$	reduce $F \to$ id	GOTO$[6, F] = 3$
0 1 6 3	$E+F$	\$	reduce $T \to F$	GOTO$[6, T] = 9$
0 1 6 9	$E+T$	\$	reduce $E \to E+T$	GOTO$[0, E] = 1$

Adapted from *Compilers* (Aho, Lam, Sethi, Ullman)

Example parse

Stack	Symbols	Input	Action	
0		id $*$ id $+$ id\$	shift	
0 5	id	$*$ id $+$ id\$	reduce $F \to$ id	GOTO$[0, F] = 3$
0 3	F	$*$ id $+$ id\$	reduce $T \to F$	GOTO$[0, T] = 2$
0 2	T	$*$ id $+$ id\$	shift	
0 2 7	$T*$	id $+$ id\$	shift	
0 2 7 5	$T*$ id	$+$ id\$	reduce $F \to$ id	GOTO$[7, F] = 10$
0 2 7 10	$T*F$	$+$ id\$	reduce $T \to T*F$	GOTO$[0, T] = 2$
0 2	T	$+$ id\$	reduce $E \to T$	GOTO$[0, E] = 1$
0 1	E	$+$ id\$	shift	
0 1 6	$E+$	id\$	shift	
0 1 6 5	$E+$ id	\$	reduce $F \to$ id	GOTO$[6, F] = 3$
0 1 6 3	$E+F$	\$	reduce $T \to F$	GOTO$[6, T] = 9$
0 1 6 9	$E+T$	\$	reduce $E \to E+T$	GOTO$[0, E] = 1$
0 1	E	\$	accept	

Adapted from *Compilers* (Aho, Lam, Sethi, Ullman)

The limits of SLR(1)

A new example grammar (for *assignment expressions*):

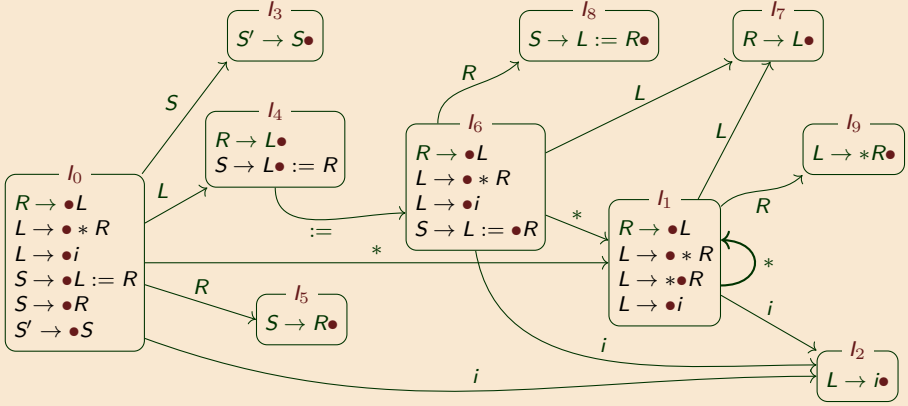
$$G_4 = \langle N_4, T_4, P_4, S' \rangle$$

$$N_4 = \{S', S, L, R\}$$

$$T_4 = \{*, :=, id\}$$

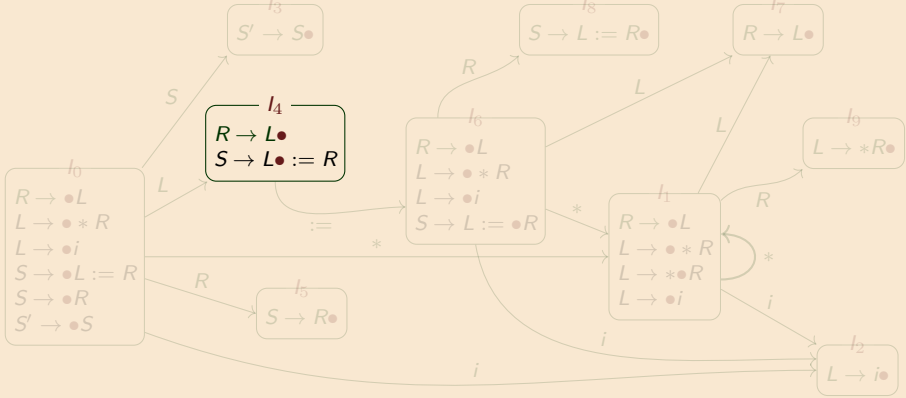
$$\begin{aligned} P_4: \quad & S' \to S\ \$ \\ & S \to L := R \mid R \\ & L \to *R \mid \mathrm{id} \\ & R \to L \end{aligned}$$

LR(0) DFA for grammar G_4



LR(0) DFA for grammar G_4

Ambiguity
In state 4 there is a shift/reduce conflict between $S \to L\bullet := R$ and $R \to L\bullet$



SLR(1) cannot resolve this conflict.

Suppose we see := in the input in state I_4. Then:

shifting is valid

Since $S \to L\bullet := R \in I_4$,
and $\delta(I_4, ":=") = I_6$,

$\text{ACTION}[4, ":="] = \text{shift } 6$

reducing is valid

Since $R \to L\bullet \in I_4$
and $":=" \in \text{FOLLOW}(R)$,

$\text{ACTION}[4, ":="] = \text{reduce } R \to L$

LR(1)

With SLR(1) there may be

shift-reduce conflicts **reduce-reduce** conflicts

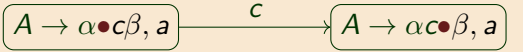
when ACTION and GOTO are not uniquely defined.

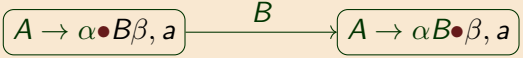
Options: fix the grammar, or use a more powerful parsing technique.

LR(1) parsing extends LR(0) items with an explicit look-ahead token a:

$$\overbrace{\underbrace{A \to \alpha \bullet \beta}_{\text{LR(0) item}}, \underbrace{a}_{\text{lookahead token}}}^{\text{LR(1) item}}$$

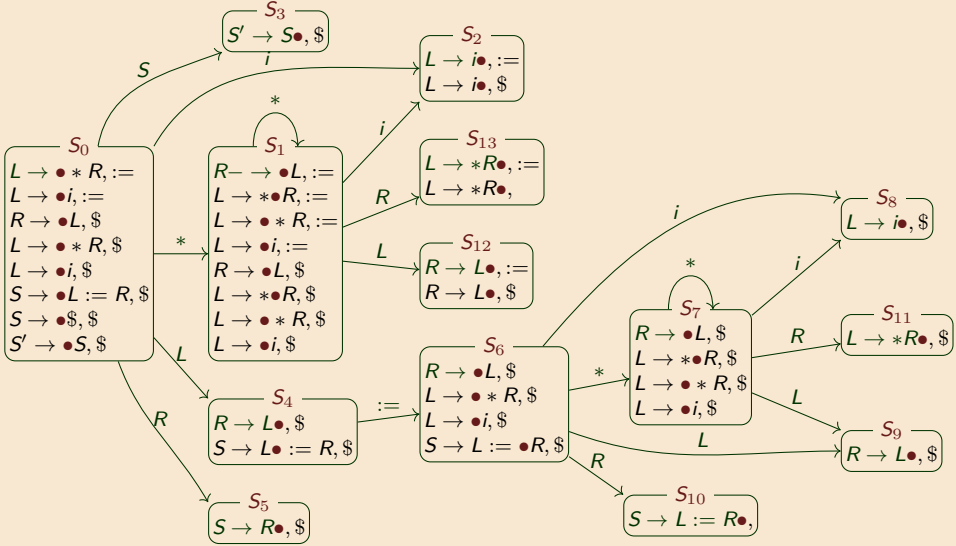
Define an NFA with LR (1) items as states



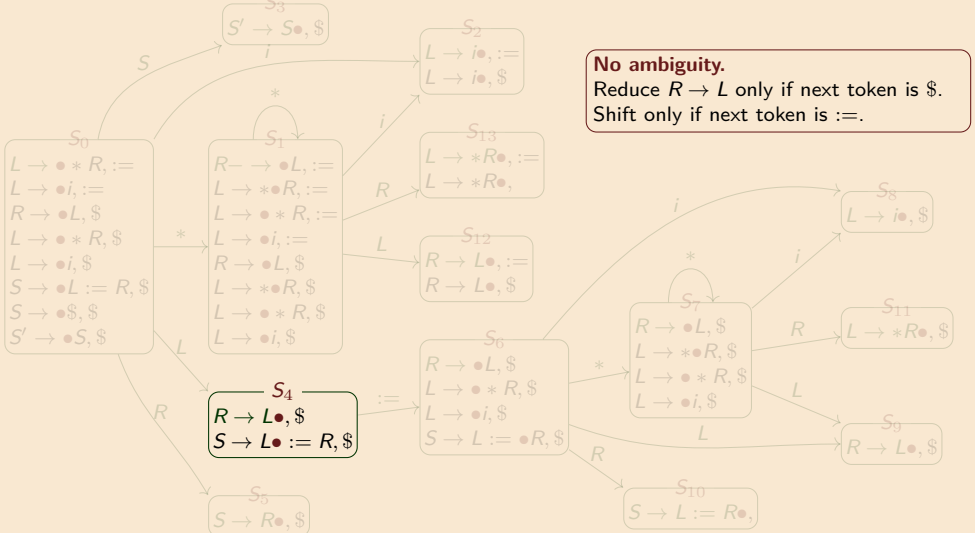


For each $b \in \textsc{first}(\beta a)$:

LR(1) DFA for grammar G_4



LR(1) DFA for grammar G_4



No ambiguity.
Reduce $R \to L$ only if next token is \$.
Shift only if next token is $:=$.

Constructing ACTION and GOTO for LR(1)

If $[A \to \alpha \bullet a\beta, a] \in I_i$ and $\delta(I_i, a) = I_j$
then $\text{ACTION}[i, a] = \text{SHIFT}\ \ j$.

If $[A \to \alpha \bullet, b] \in I_i$ and $A \neq S$,
then $\text{ACTION}[i, b] = \text{REDUCE}\ A \to \alpha$

If $[S \to \alpha \bullet, \$] \in I_i$
then $\text{ACTION}[i, \$] = \text{ACCEPT}$

If $\delta(I_i, A) = I_j$
then $\text{GOTO}[i, A] = j$.

Key change from SLR(1): use **lookahead** (i.e.FIRST), not FOLLOW, to select REDUCE actions

SLR(1)

If $[A \to \alpha\bullet] \in I_i$ and $A \neq S$
then for all $a \in \text{FOLLOW}(A)$,
$\text{ACTION}[i, a] =$ reduce $A \to \alpha$

LR(1)

If $[A \to \alpha\bullet, b] \in I_i$ and $A \neq S$
then
$\text{ACTION}[i, b] =$ reduce $A \to \alpha$

NB: b used *only* for reductions, not for shifts

LR(1) more powerful than SLR(1)

LR(1) DFA may have a very large number of states.
LALR[1] optimises DFA, collapsing states (but can give strange error messages)

[1] Implemented in yacc; not covered here

Next time: translation