LR(1)-items

$[A \to \alpha \cdot \beta, \Delta]$

Where Δ is called **lookahead-set**

The closure of LR(1)-items refines $\mathrm{closure}_0(\_)$

The closure of $\{[A \to \alpha \cdot B\beta, \Delta]\}$ propagates the symbols following B to those items that are added to the set to close wrt B

The goal is to refine follow-sets: compute them locally rather than globally

Why taking sets of lookaheads?

Consider the grammar

$$\begin{array}{lcl} S & \to & Ab \\ A & \to & Ba \mid a \\ B & \to & Ac \mid c \end{array}$$

- When we accept a word we expect to see \$ in the input buffer hence we initially take $[S' \to \cdot S, \$]$
- What is the closure of $\{[S' \to \cdot S, \$]\}$?
- S is followed by \$ in $[S' \to \cdot S, \$]$ and $S \to Ab$ hence take $[S \to \cdot Ab, \$]$
- A is followed by b in $[S \to \cdot Ab, \$]$ and $A \to Ba \mid a$ hence take $[A \to \cdot Ba, b]$ and $[A \to \cdot a, b]$
- B is followed by a in $[A \to \cdot Ba, b]$ and $B \to Ac \mid c$ hence take $[B \to \cdot Ac, a]$ and $[B \to \cdot c, a]$
- A is followed by c in $[B \to \cdot Ac, a]$ hence take also $[A \to \cdot Ba, c]$ and $[A \to \cdot a, c]$

Why taking sets of lookaheads?

For grammar

$$\begin{array}{rcl} S & \rightarrow & Ab \\ A & \rightarrow & Ba \mid a \\ B & \rightarrow & Ac \mid c \end{array}$$

- Write $[A \rightarrow \cdot Ba, b]$ and $[A \rightarrow \cdot Ba, c]$ as the single LR(1)-item $[A \rightarrow \cdot Ba, \{b, c\}]$
- Write $[A \rightarrow \cdot a, b]$ and $[A \rightarrow \cdot a, c]$ as the single LR(1)-item $[A \rightarrow \cdot a, \{b, c\}]$

$\mathrm{closure}_1(P)$

function $\mathrm{closure}_1(P)$
  tag every item in P as unmarked ;
  while *there is an unmarked item I in P* **do**
    mark I ;
    if *I has the form* $[A \rightarrow \alpha \cdot B\beta, \Delta]$ **then**
      $\Delta_1 \longleftarrow \bigcup_{d \in \Delta} \mathrm{first}(\beta d)$;
      foreach $B \rightarrow \gamma \in \mathcal{P}'$ **do**
        if *there is no item in P with first component* $B \rightarrow \cdot\gamma$ **then**
          add $[B \rightarrow \cdot\gamma, \Delta_1]$ as an unmarked item to P ;
        else
          if $([B \rightarrow \cdot\gamma, \Gamma] \in P$ *and* $\Delta_1 \not\subseteq \Gamma)$ **then**
            update $[B \rightarrow \cdot\gamma, \Gamma]$ to $[B \rightarrow \cdot\gamma, \Gamma \cup \Delta_1]$ in P ;
            tag $[B \rightarrow \cdot\gamma, \Gamma \cup \Delta_1]$ as unmarked ;
  return P ;

Closure of sets of LR(1)-items

Given the set of LR(1)-items P, $\mathrm{closure}_1(P)$ is the smallest set of items, with smallest lookahead-sets, that satisfies the following equation

$$
\begin{aligned}
&\mathrm{closure}_1(P) = P \cup \\
&\quad \{[B \to \cdot\gamma, \Gamma] \text{ such that} \\
&\qquad\quad [A \to \alpha \cdot B\beta, \Delta] \in \mathrm{closure}_1(P) \text{ and} \\
&\qquad\quad B \to \gamma \in \mathcal{P}' \text{ and } \mathrm{first}(\beta\Delta) \subseteq \Gamma\}
\end{aligned}
$$

Where $\mathrm{first}(\beta\Delta) = \bigcup_{d \in \Delta} \mathrm{first}(\beta d)$

For example, if $\Delta = \{d_1, d_2\}$
then $\mathrm{first}(\beta\Delta) = \mathrm{first}(\beta d_1) \cup \mathrm{first}(\beta d_2)$

Fixed point computation: initialize $\mathrm{closure}_1(P)$ as P, then add items as needed, up to reaching the fixed point

$\mathrm{closure}_1(P)$
EXAMPLE

Take

$$
\begin{aligned}
S &\to aAd \mid bBd \mid aBe \mid bAe \\
A &\to c \\
B &\to c
\end{aligned}
$$

Compute $\mathrm{closure}_1(\{[S' \to \cdot S, \{\$\}]\})$

$\mathrm{closure}_1(P)$
EXAMPLE

$$\mathrm{closure}_1(P) = P \cup \{[B \to \cdot\gamma, \Gamma] \text{ such that } [A \to \alpha \cdot B\beta, \Delta] \in \mathrm{closure}_1(P) \text{ and } B \to \gamma \in \mathcal{P}' \text{ and } \mathrm{first}(\beta\Delta) \subseteq \Gamma\}$$

- Init: $\mathrm{closure}_1(\{[S' \to \cdot S, \{\$\}]\}) = \{[S' \to \cdot S, \{\$\}]\}$
- Here:
  - B in the equation is instantiated by S
  - β in the equation is instantiated by ϵ
  - Δ in the equation is instantiated by $\{\$\}$
- Hence add to the set the LR(0)-items for the closure wrt S, all paired with the lookahead-set $\{\$\}$

Construction of LR(1)-automaton

Construct the automaton by populating a collection of states while definining the transition function

$P_0 = \mathrm{closure}_1(\{[S' \to \cdot S, \{\$\}]\})$

If an already collected state P contains an item of the form $[A \to \alpha \cdot Y\beta, \Delta]$

Then there is a transition from P to a state Q which contains the item $[A \to \alpha Y \cdot \beta, \Delta]$

And, since Q contains $[A \to \alpha Y \cdot \beta, \Delta]$, it also contains all the items in $\mathrm{closure}_1(\{[A \to \alpha Y \cdot \beta, \Delta]\})$

Construction of LR(1)-automata

initialize the collection $\mathcal{Q}$ to contain $P_0 = \mathrm{closure}_1(\{[S' \to \cdot S, \{\$\}]\})$;
tag P_0 as unmarked;
while *there is an unmarked state P in $\mathcal{Q}$* **do**
  mark P ;
  foreach *Y on the right side of the marker in some item of P* **do**
    Compute in Tmp the kernel-set of the Y-target of P;
    if *$\mathcal{Q}$ already contains a state Q whose kernel is Tmp* **then**
      Let Q be the Y-target of P;
    else
      Add $\mathrm{closure}_1(Tmp)$ to $\mathcal{Q}$ as an unmarked state;
      Let $\mathrm{closure}_1(Tmp)$ be the Y-target of P;

Construction of LR(1)-automata

EXAMPLE

Construct the LR(1)-automaton for

$$\begin{array}{lcl} S & \to & aAd \mid bBd \mid aBe \mid bAe \\ A & \to & c \\ B & \to & c \end{array}$$

Construction of LR(1)-automata

EXAMPLE

$$\begin{array}{lcl} S & \to & aAd \mid bBd \mid aBe \mid bAe \\ A & \to & c \\ B & \to & c \end{array}$$

(Omitting the enclosing "[ ]" for LR(1)-items)

0:

$S' \to \cdot S, \{\$\}$
$S \to \cdot aAd, \{\$\}$ $S \to \cdot bBd, \{\$\}$ $S \to \cdot aBe, \{\$\}$ $S \to \cdot bAe, \{\$\}$

$1 = \tau(0, S)$:

$S' \to S\, \cdot, \{\$\}$

$2 = \tau(0, a)$:

$S \to a \cdot Ad, \{\$\}$ $S \to a \cdot Be, \{\$\}$
$A \to \cdot c, \{d\}$ $B \to \cdot c, \{e\}$

Construction of LR(1)-automata

EXAMPLE

$$\begin{array}{lcl} S & \to & aAd \mid bBd \mid aBe \mid bAe \\ A & \to & c \\ B & \to & c \end{array}$$

$3 = \tau(0, b)$:

$S \to b \cdot Bd, \{\$\}$ $S \to b \cdot Ae, \{\$\}$
$A \to \cdot c, \{e\}$ $B \to \cdot c, \{d\}$

$4 = \tau(2, A)$:

$S \to aA \cdot d, \{\$\}$

$5 = \tau(2, B)$:

$S \to aB \cdot e, \{\$\}$

$6 = \tau(2, c)$:

$A \to c\, \cdot, \{d\}$ $B \to c\, \cdot, \{e\}$

$7 = \tau(3, B)$:

$S \to bB \cdot d, \{\$\}$

Construction of LR(1)-automata

EXAMPLE

$$\begin{array}{lcl} S & \rightarrow & aAd \mid bBd \mid aBe \mid bAe \\ A & \rightarrow & c \\ B & \rightarrow & c \end{array}$$

$8 = \tau(3, A)$:

$\boxed{S \rightarrow bA \cdot e, \{\$\}}$

$9 = \tau(3, c)$:

$\boxed{\begin{array}{l} A \rightarrow c \cdot, \{e\} \\ B \rightarrow c \cdot, \{d\} \end{array}}$

$10 = \tau(4, d)$:

$\boxed{S \rightarrow aAd \cdot, \{\$\}}$

$11 = \tau(5, e)$:

$\boxed{S \rightarrow aBe \cdot, \{\$\}}$

$12 = \tau(7, d)$:

$\boxed{S \rightarrow bBd \cdot, \{\$\}}$

$13 = \tau(8, e)$:

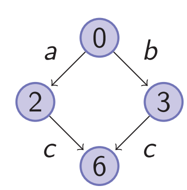
$\boxed{S \rightarrow bAe \cdot, \{\$\}}$

Construction of LR(1)-automata

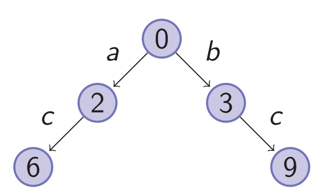
EXAMPLE

$$\begin{array}{lcl} S & \rightarrow & aAd \mid bBd \mid aBe \mid bAe \\ A & \rightarrow & c \\ B & \rightarrow & c \end{array}$$

Paths *ac* and *bc* in the LR(0)-automaton



Paths *ac* and *bc* in the LR(1)-automaton



LR(1) parsing tables

LR(1) parsing tables are obtained by taking:

- Characteristic automaton: LR(1)-automaton
- Lookahead function: for every $[A \to \beta\cdot, \Delta] \in P$, $\mathcal{LA}(P, A \to \beta) = \Delta$

$\mathcal{G}$ **is LR(1)** iff its LR(1) parsing table has no conflict

LR(1)-parsing

EXAMPLE

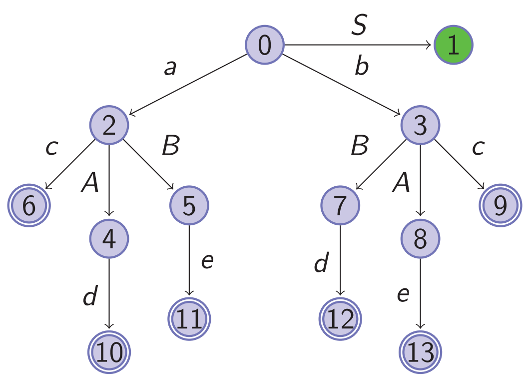
$$
\begin{array}{lcl}
S & \to & aAd \mid bBd \mid aBe \mid bAe \\
A & \to & c \\
B & \to & c
\end{array}
$$

6:

$$\boxed{\begin{array}{l} A \to c\cdot, \{d\} \\ B \to c\cdot, \{e\} \end{array}}$$

9:

$$\boxed{\begin{array}{l} A \to c\cdot, \{e\} \\ B \to c\cdot, \{d\} \end{array}}$$



LR(1)-parsing

TRAINING

Construct the LR(1)-automaton for

$$\begin{array}{lcl} S & \rightarrow & AaB \mid b \\ A & \rightarrow & BcBaA \mid \epsilon \\ B & \rightarrow & \epsilon \end{array}$$

0:

$S' \rightarrow \cdot S, \{\$\}$
$S \rightarrow \cdot AaB, \{\$\}$
$S \rightarrow \cdot b, \{\$\}$
$A \rightarrow \cdot BcBaA, \{a\}$
$A \rightarrow \cdot, \{a\}$
$B \rightarrow \cdot, \{c\}$

$1 = \tau(0, S)$:

$S' \rightarrow S \cdot, \{\$\}$

$2 = \tau(0, A)$:

$S \rightarrow A \cdot aB, \{\$\}$

LR(1)-parsing

TRAINING

$$\begin{array}{lcl} S & \rightarrow & AaB \mid b \\ A & \rightarrow & BcBaA \mid \epsilon \\ B & \rightarrow & \epsilon \end{array}$$

$3 = \tau(0, b)$:

$S \rightarrow b \cdot, \{\$\}$

$4 = \tau(0, B)$:

$A \rightarrow B \cdot cBaA, \{a\}$

$5 = \tau(2, a)$:

$S \rightarrow Aa \cdot B, \{\$\}$
$B \rightarrow \cdot, \{\$\}$

$6 = \tau(4, c)$:

$A \rightarrow Bc \cdot BaA, \{a\}$
$B \rightarrow \cdot, \{a\}$

$7 = \tau(5, B)$:

$S \rightarrow AaB \cdot, \{\$\}$

$8 = \tau(6, B)$:

$A \rightarrow BcB \cdot aA, \{a\}$

LR(1)-parsing

TRAINING

$$
\begin{array}{lcl}
S & \rightarrow & AaB \mid b \\
A & \rightarrow & BcBaA \mid \epsilon \\
B & \rightarrow & \epsilon
\end{array}
$$

$9 = \tau(8, a)$:

$$\boxed{A \rightarrow BcBa \cdot A, \{a\}}$$

$$\boxed{\begin{array}{l} A \rightarrow \cdot BcBaA, \{a\} \\ A \rightarrow \cdot, \{a\} \\ B \rightarrow \cdot, \{c\} \end{array}}$$

$\tau(9, B) = 4$

$10 = \tau(9, A)$:

$$\boxed{A \rightarrow BcBaA \cdot, \{a\}}$$

LR(1) parsing

CASE STUDY

Construct the LR(1) parsing table for

$$
\begin{array}{lcl}
S & \rightarrow & L = R \mid R \\
L & \rightarrow & *R \mid id \\
R & \rightarrow & L
\end{array}
$$

LR(1) parsing

CASE STUDY

$$\begin{array}{lcl} S & \rightarrow & L = R \mid R \\ L & \rightarrow & *R \mid id \\ R & \rightarrow & L \end{array}$$

1 = $\tau(0, S)$:

$S' \rightarrow S \cdot, \{\$\}$

0:

$S' \rightarrow \cdot S, \{\$\}$

$S \rightarrow \cdot L = R, \{\$\}$
$S \rightarrow \cdot R, \{\$\}$
$L \rightarrow \cdot * R, \{=, \$\}$
$L \rightarrow \cdot id, \{=, \$\}$
$R \rightarrow \cdot L, \{\$\}$

2 = $\tau(0, L)$:

$S \rightarrow L \cdot = R, \{\$\}$
$R \rightarrow L \cdot, \{\$\}$

3 = $\tau(0, R)$:

$S \rightarrow R \cdot, \{\$\}$

LR(1) parsing

CASE STUDY

$$\begin{array}{lcl} S & \rightarrow & L = R \mid R \\ L & \rightarrow & *R \mid id \\ R & \rightarrow & L \end{array}$$

4 = $\tau(0, *)$:

$L \rightarrow * \cdot R, \{=, \$\}$

$R \rightarrow \cdot L, \{=, \$\}$
$L \rightarrow \cdot * R, \{=, \$\}$
$L \rightarrow \cdot id, \{=, \$\}$

6 = $\tau(2, =)$:

$S \rightarrow L = \cdot R, \{\$\}$

$R \rightarrow \cdot L, \{\$\}$
$L \rightarrow \cdot * R, \{\$\}$
$L \rightarrow \cdot id, \{\$\}$

5 = $\tau(0, id)$:

$L \rightarrow id \cdot, \{=, \$\}$

7 = $\tau(4, R)$:

$L \rightarrow * R \cdot, \{=, \$\}$

LR(1) parsing

CASE STUDY

$$
\begin{array}{lcl}
S & \rightarrow & L = R \mid R \\
L & \rightarrow & *R \mid id \\
R & \rightarrow & L
\end{array}
$$

$8 = \tau(4, L)$:

$$\boxed{R \rightarrow L\cdot, \{=, \$\}}$$

$9 = \tau(6, R)$:

$$\boxed{S \rightarrow L = R\cdot, \{\$\}}$$

$10 = \tau(6, L)$:

$$\boxed{R \rightarrow L\cdot, \{\$\}}$$

$11 = \tau(6, *)$:

$$\boxed{L \rightarrow * \cdot R, \{\$\}}$$

$$\boxed{\begin{array}{l} R \rightarrow \cdot L, \{\$\} \\ L \rightarrow \cdot *R, \{\$\} \\ L \rightarrow \cdot id, \{\$\} \end{array}}$$

$12 = \tau(6, id)$:

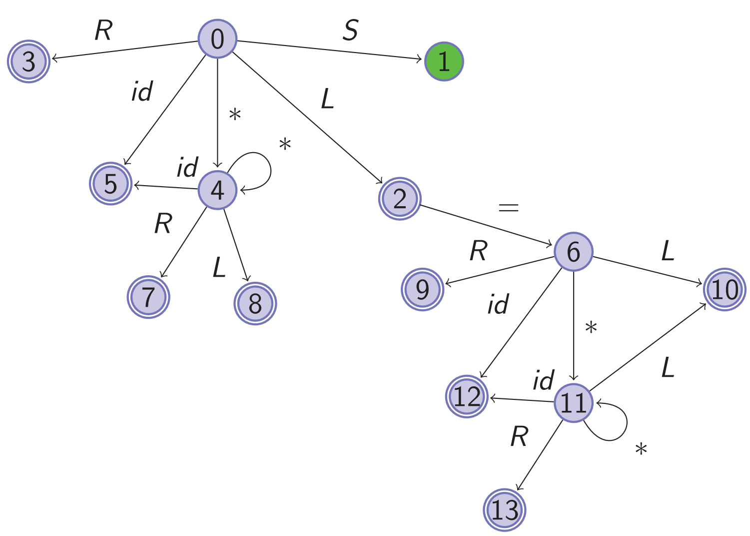
$$\boxed{L \rightarrow id\cdot, \{\$\}}$$

$13 = \tau(11, R)$:

$$\boxed{L \rightarrow *R\cdot, \{\$\}}$$

LR(1) parsing

CASE STUDY: LR(1)-AUTOMATON



LR(1) parsing
CASE STUDY

$$
\begin{array}{lcl}
S & \rightarrow & L = R \mid R \\
L & \rightarrow & *R \mid id \\
R & \rightarrow & L
\end{array}
$$

Is it LR(1)?

Yes, it is

LR(1) parsing
CASE STUDY

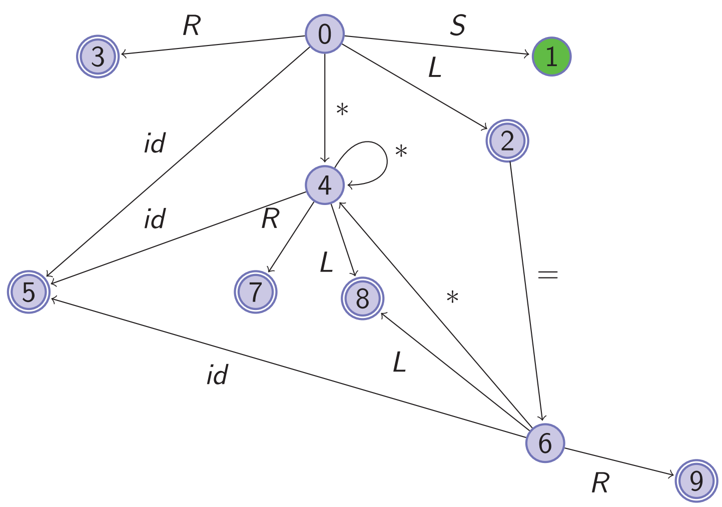
$$
\begin{array}{lcl}
S & \rightarrow & L = R \mid R \\
L & \rightarrow & *R \mid id \\
R & \rightarrow & L
\end{array}
$$

Is it SLR(1)?

No, it is not

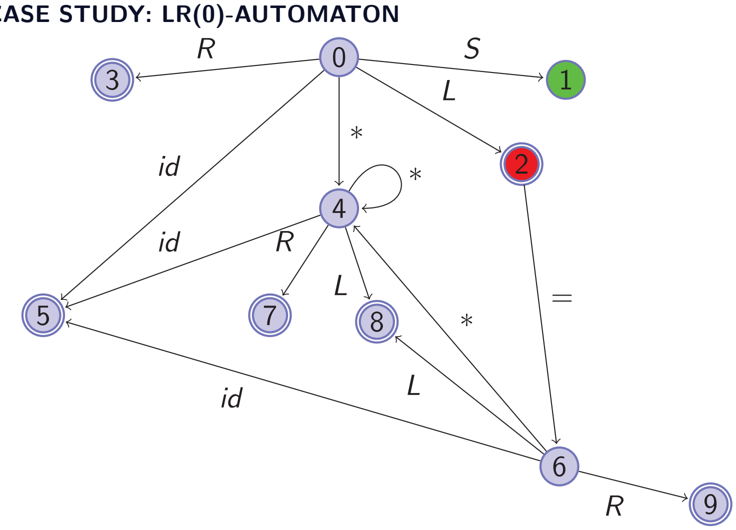
LR(1) parsing

CASE STUDY: LR(0)-AUTOMATON



LR(1) parsing

CASE STUDY: LR(0)-AUTOMATON



There is an s/r conflict in $(2, =)$ where the reduce is for $R \to L$

LR(1) parsing

CASE STUDY

$$
\begin{array}{lcl}
S & \rightarrow & L = R \mid R \\
L & \rightarrow & *R \mid id \\
R & \rightarrow & L
\end{array}
$$

State 2 in the LR(0)-automaton

$$\boxed{\begin{array}{l} S \rightarrow L \cdot = R \\ R \rightarrow L \cdot \end{array}}$$

State 2 in the LR(1)-automaton

$$\boxed{\begin{array}{l} S \rightarrow L \cdot = R, \{\$\} \\ R \rightarrow L \cdot, \{\$\} \end{array}}$$

Calling for the reduce of $R \rightarrow L\cdot$ for both \$ and $=$

Calling for the reduce of $R \rightarrow L\cdot$ only for \$

Why this difference?

LR(1) parsing

CASE STUDY

$$
\begin{array}{lcl}
S & \rightarrow & L = R \mid R \\
L & \rightarrow & *R \mid id \\
R & \rightarrow & L
\end{array}
$$

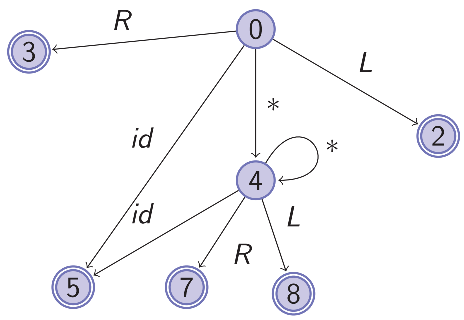
Suppose we are parsing the string "$w_1 = w_2$"

By the first production of the grammar, w_1 must be reduced to L

From L we can derive words in the set $\{*^n id \mid n \geq 0\}$ and only the derivations of words in $\{*^n id \mid n > 0\}$ involve R

LR(1) parsing

CASE STUDY

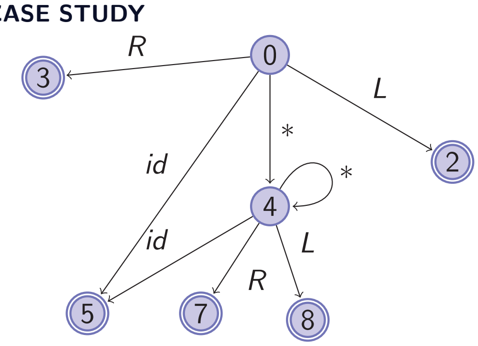


If the parsed string is "$id{=}w_2$", then we move to state 5

Then reduce $L \to id$, and move to state 2

LR(1) parsing

CASE STUDY

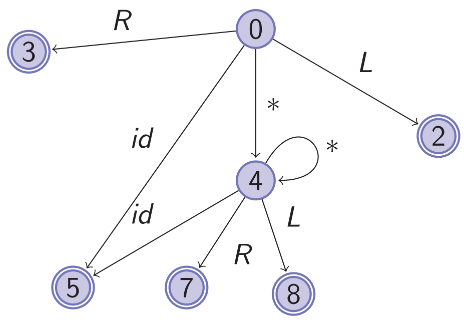


If the parsed string is "$*w_1'{=}w_2$", then we move to state 4 and remain there if we read some more $*$

Upon reading *id* we move to state 5, reduce $L \to id$, and move to state 8, **not** to state 2

LR(1) parsing

CASE STUDY

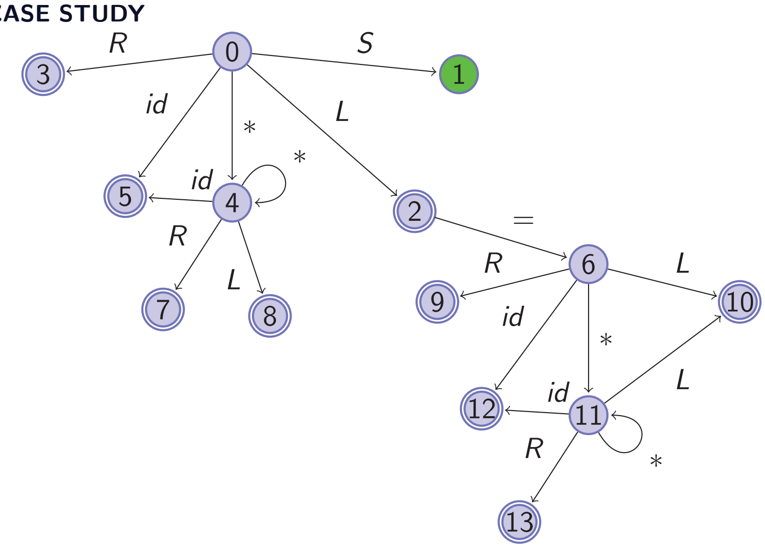


SLR(1) parsing does not capture the difference between state 2 and state 8

LR(1) does it (at a price)

LR(1) parsing

CASE STUDY



Some states of the LR(1)-automaton have the same LR(0)-projection (i.e. look the same if all lookahead-sets are forgotten)

LR(1) parsing

CASE STUDY

Kernel of 4:

$\boxed{L \rightarrow * \cdot R, \{=, \$\}}$

Kernel of 11:

$\boxed{L \rightarrow * \cdot R, \{\$\}}$

Kernel of 5:

$\boxed{L \rightarrow id\cdot, \{=, \$\}}$

Kernel of 12:

$\boxed{L \rightarrow id\cdot, \{\$\}}$

Kernel of 7:

$\boxed{L \rightarrow *R\cdot, \{=, \$\}}$

Kernel of 13:

$\boxed{L \rightarrow *R\cdot, \{\$\}}$

Kernel of 8:

$\boxed{R \rightarrow L\cdot, \{=, \$\}}$

Kernel of 10:

$\boxed{R \rightarrow L\cdot, \{\$\}}$

LRm(1)-automaton

Construct the LRm(1)-automaton $\mathcal{AM}$ from the LR(1)-automaton $\mathcal{A}$ ("m" for "merged")

States: Merge in one single state of $\mathcal{AM}$ all the items in the states $P_1, \ldots, P_n$ of $\mathcal{A}$ which have the same LR(0)-projection

Transitions:

- If
  - The state P of $\mathcal{A}$ has a Y-transition to Q and
  - P has been merged in $\langle P_1, \ldots, P_n \rangle$ and
  - Q has been merged in $\langle Q_1, \ldots, Q_m \rangle$
- Then there is a Y-transition in $\mathcal{AM}$ from $\langle P_1, \ldots, P_n \rangle$ to $\langle Q_1, \ldots, Q_m \rangle$

LRm(1)-automaton

EXAMPLE

From

4:

$L \to * \cdot R, \{=, \$\}$
$R \to \cdot L, \{=, \$\}$ $L \to \cdot * R, \{=, \$\}$ $L \to \cdot id, \{=, \$\}$

11:

$L \to * \cdot R, \{\$\}$
$R \to \cdot L, \{\$\}$ $L \to \cdot * R, \{\$\}$ $L \to \cdot id, \{\$\}$

Construct the state

4&11

$L \to * \cdot R, \{=, \$\},\ L \to * \cdot R, \{\$\}$
$R \to \cdot L, \{=, \$\},\ R \to \cdot L, \{\$\}$ $L \to \cdot * R, \{=, \$\},\ L \to \cdot * R, \{\$\}$ $L \to \cdot id, \{=, \$\},\ L \to \cdot id, \{\$\}$

LRm(1)-automaton

EXAMPLE

From

5:

$L \to id \cdot, \{=, \$\}$

12:

$L \to id \cdot, \{\$\}$

Construct the state

5&12

$L \to id \cdot, \{=, \$\},\ L \to id \cdot, \{\$\}$

LRm(1)-automaton
EXAMPLE

In the LR(1)-automaton there are a transition labelled by *id* from state 4 to state 5 (and also a transition labelled by *id* from state 11 to state 12)

States 4 and 11 are merged together to form 4&11, and 5 and 12 are merged together to form 5&12

Then in the LRm(1)-automaton set a transition labelled by *id* from state 4&11 to state 5&12

LRm(1)-automaton
OBSERVATIONS

By construction, the states of the LRm(1)-automaton can contain more items with the same LR(0)-projection

The LRm(1)-automaton has the same number of states and the same layout as the LR(0)-automaton

LALR(1) parsing tables

LALR(1) parsing tables are obtained by taking:

- Characteristic automaton: LRm(1)-automaton
- Lookahead function: $\mathcal{LA}(P, A \to \beta) = \bigcup_{[A \to \beta \cdot, \Delta_j]} \Delta_j$

$\mathcal{G}$ **is LALR(1)** iff its LALR(1) parsing table has no conflict

LALR(1) parsing

CASE STUDY

$$
\begin{array}{lcl}
S & \to & L = R \mid R \\
L & \to & *R \mid id \\
R & \to & L
\end{array}
$$

Is it LALR(1)?

Yes, in fact there is no conflict in state 2 of the LR(1)-automaton, and the LRm(1)-automaton keeps state 2 as it is in the LR(1)-automaton

LALR(1) parsing

TRAINING

Construct the LALR(1) parsing table for

$$\begin{array}{lcl} S & \rightarrow & AaB \mid b \\ A & \rightarrow & BcBaA \mid \epsilon \\ B & \rightarrow & \epsilon \end{array}$$

Another way of getting LALR(1) tables

Compute a **symbolic automaton**: an automaton with LR(1)-like items, but where lookahead-sets are kept symbolic

Use variables as lookahead-sets of kernel items and populate a system of equations over these variables

The target of transitions is decided in the LR(0)-way, by only looking at the first projection of the kernel items of the target

When the target of a transition is an already installed state, the contributions to the lookahead-sets of the target items is recorded in the of equations of the variables associated with its kernel items

After the construction of the symbolic automaton, resolve the system of equations to instantiate the lookahead-sets of reducing items

Another way of getting LALR(1) tables

We now run the construction of the symbolic automaton for the grammar that was used as case study for introducing LALR(1) parsing

At each step of the construction we compare the obtained state of the symbolic automaton with a corresponding state of the LR(1)-automaton

Keep in mind that the following behaviour

> *When the target of a transition is an already installed state, the contributions to the lookahead-sets of the target items is recorded in the of equations of the variables associated with its kernel items*

plays the role of merging states with the same first projection as we do with the LRm(1)-automaton

Symbolic automaton
EXAMPLE

Computation of the symbolic automaton for

$$
\begin{array}{lcl}
S & \rightarrow & L = R \mid R \\
L & \rightarrow & *R \mid id \\
R & \rightarrow & L
\end{array}
$$

Symbolic automaton
EXAMPLE

0:		0:
$S' \to \cdot S, \{x_0\}$		$S' \to \cdot S, \{\$\}$
$S \to \cdot L = R, \{x_0\}$ $S \to \cdot R, \{x_0\}$ $L \to \cdot * R, \{=, x_0\}$ $L \to \cdot id, \{=, x_0\}$ $R \to \cdot L, \{x_0\}$	instead of	$S \to \cdot L = R, \{\$\}$ $S \to \cdot R, \{\$\}$ $L \to \cdot * R, \{=, \$\}$ $L \to \cdot id, \{=, \$\}$ $R \to \cdot L, \{\$\}$

Add the equation

- $x_0 = \{\$\}$

Symbolic automaton
EXAMPLE

Compute $\tau(0, S)$ after $[S' \to \cdot S, \{x_0\}]$

1:		1:
$S' \to S\ \cdot, \{x_1\}$	instead of	$S' \to S\ \cdot, \{\$\}$

Add the equation

- $x_1 = \{x_0\}$

Symbolic automaton
EXAMPLE

Compute $\tau(0, L)$ after $[S \to \cdot L = R, \{x_0\}]$ and $[R \to \cdot L, \{x_0\}]$

2:		2:
$S \to L \cdot = R, \{x_2\}$ $R \to L\cdot, \{x_3\}$	instead of	$S \to L \cdot = R, \{\$\}$ $R \to L\cdot, \{\$\}$

Add equations
- $x_2 = \{x_0\}$
- $x_3 = \{x_0\}$

Symbolic automaton
EXAMPLE

Compute $\tau(0, R)$ after $[S \to \cdot R, \{x_0\}]$

3:		3:
$S \to R\cdot, \{x_4\}$	instead of	$S \to R\cdot, \{\$\}$

Add the equation
- $x_4 = \{x_0\}$

Symbolic automaton
EXAMPLE

Compute $\tau(0, *)$ after $[L \to \cdot * R, \{=, x_0\}]$

4:		4:
$L \to * \cdot R, \{x_5\}$		$L \to * \cdot R, \{=, \$\}$
$R \to \cdot L, \{x_5\}$ $L \to \cdot * R, \{x_5\}$ $L \to \cdot id, \{x_5\}$	instead of	$R \to \cdot L, \{=, \$\}$ $L \to \cdot * R, \{=, \$\}$ $L \to \cdot id, \{=, \$\}$

Add the equation

- $x_5 = \{=, x_0\}$

Symbolic automaton
EXAMPLE

Compute $\tau(0, id)$ after $[L \to \cdot id, \{=, x_0\}]$

5:		5:
$L \to id \cdot, \{x_6\}$	instead of	$L \to id \cdot, \{=, \$\}$

Add the equation

- $x_6 = \{=, x_0\}$

Symbolic automaton

EXAMPLE

Compute $\tau(2, =)$ after $[S \to L \cdot = R, \{x_2\}]$

6:		6:
$S \to L = \cdot R, \{x_7\}$	instead of	$S \to L = \cdot R, \{\$\}$
$R \to \cdot L, \{x_7\}$ $L \to \cdot * R, \{x_7\}$ $L \to \cdot id, \{x_7\}$		$R \to \cdot L, \{\$\}$ $L \to \cdot * R, \{\$\}$ $L \to \cdot id, \{\$\}$

Add the equation

- $x_7 = \{x_2\}$

Symbolic automaton

EXAMPLE

Compute $\tau(4, R)$ after $[L \to * \cdot R, \{x_5\}]$

7:		7:
$L \to * R \cdot, \{x_8\}$	instead of	$L \to * R \cdot, \{=, \$\}$

Add the equation

- $x_8 = \{x_5\}$

Symbolic automaton
EXAMPLE

Compute $\tau(4, L)$ after $[R \to \cdot L, \{x_5\}]$

8: $\boxed{R \to L\cdot, \{x_9\}}$ instead of 8: $\boxed{R \to L\cdot, \{=, \$\}}$

Add the equation

- $x_9 = \{x_5\}$

Symbolic automaton
EXAMPLE

Compute $\tau(4, *)$ after $[L \to \cdot * R, \{x_5\}]$

The target is

4

$L \to * \cdot R, \{x_5\}$
$R \to \cdot L, \{x_5\}$ $L \to \cdot * R, \{x_5\}$ $L \to \cdot id, \{x_5\}$

Update equation $x_5 = \{=, x_0\}$ to

- $x_5 = \{=, x_0\} \cup \{x_5\}$

Symbolic automaton
EXAMPLE

Compute $\tau(4, id)$ after $[L \to \cdot id, \{x_5\}]$

The target is

$$\begin{array}{c} 5 \\ \boxed{L \to id\cdot, \{x_6\}} \end{array}$$

Update equation $x_6 = \{=, x_0\}$ to

- $x_6 = \{=, x_0\} \cup \{x_5\}$

Symbolic automaton
EXAMPLE

Compute $\tau(6, R)$ after $[S \to L = \cdot R, \{x_7\}]$

$$\begin{array}{l} 9: \\ \boxed{S \to L = R\cdot, \{x_{10}\}} \end{array} \quad \text{instead of} \quad \begin{array}{l} 9: \\ \boxed{S \to L = R\cdot, \{\$\}} \end{array}$$

Add the equation

- $x_{10} = \{x_7\}$

Symbolic automaton
EXAMPLE

Compute $\tau(6, L)$ after $[R \to \cdot L, \{x_7\}]$

The target is

8
$R \to L\cdot, \{x_9\}$

Update equation $x_9 = \{x_5\}$ to

- $x_9 = \{x_5\} \cup \{x_7\}$

FORMAL LANGUAGES AND COMPILERS Paola Quaglia, 2022

Symbolic automaton
EXAMPLE

Compute $\tau(6, *)$ after $[L \to \cdot * R, \{x_7\}]$

The target is

4
$L \to * \cdot R, \{x_5\}$
$R \to \cdot L, \{x_5\}$
$L \to \cdot * R, \{x_5\}$
$L \to \cdot id, \{x_5\}$

Update equation $x_5 = \{=, x_0\} \cup \{x_5\}$ to

- $x_5 = \{=, x_0\} \cup \{x_5\} \cup \{x_7\}$

Symbolic automaton
EXAMPLE

Compute $\tau(6, id)$ after $[L \rightarrow \cdot id, \{x_7\}]$

The target is

$$\overset{5}{\boxed{L \rightarrow id\cdot, \{x_6\}}}$$

Update equation $x_6 = \{=, x_0\} \cup \{x_5\}$ to

- $x_6 = \{=, x_0\} \cup \{x_5\} \cup \{x_7\}$

Symbolic automaton

The construction of the symbolic automaton is over

Now collect all the equations, simplify the system, and resolve

Symbolic automaton

EQUATIONS

$$
\begin{array}{lcl}
x_0 & = & \{\$\} \\
x_1 & = & \{x_0\} \\
x_2 & = & \{x_0\} \\
x_3 & = & \{x_0\} \\
x_4 & = & \{x_0\} \\
x_5 & = & \{=, x_0\} \cup \{x_5\} \cup \{x_7\} \\
x_6 & = & \{=, x_0\} \cup \{x_5\} \cup \{x_7\} \\
x_7 & = & \{x_2\} \\
x_8 & = & \{x_5\} \\
x_9 & = & \{x_5\} \cup \{x_7\} \\
x_{10} & = & \{x_7\}
\end{array}
$$

$x_0, x_1, x_2, x_3, x_4, x_7, x_{10} = \{\$\}$

$x_5, x_6, x_8, x_9 = \{=, \$\}$

Symbolic automaton

In the overall:

An automaton of the same size as the LR(0)-automaton

The closure applied to kernel items is done in the LR(1)-way, but only once for all the states that would be merged in the LRm(1)-automaton

The lookahead function is computed after the values computed for the variables

For example, $x_6 = \{=, \$\}$ then

$$[L \to id\cdot, \{x_6\}]$$

stands for

$$[L \to id\cdot, \{=, \$\}]$$

Symbolic automaton

Now try with

$$S \to aSb \mid ab$$

And with

$$\begin{array}{lcl} S & \to & AaB \mid b \\ A & \to & BcBaA \mid \epsilon \\ B & \to & \epsilon \end{array}$$