Problem Set 3

Erlend Paulsen Skaaden

Task 1: Bottom/Up Parsing Tables

We are considering the following grammar:

$$\begin{aligned} E &\rightarrow E + T | T \\ T &\rightarrow T * F | F \\ F &\rightarrow x \end{aligned}$$

Subtask 1: Augmentation and LR(0) automaton

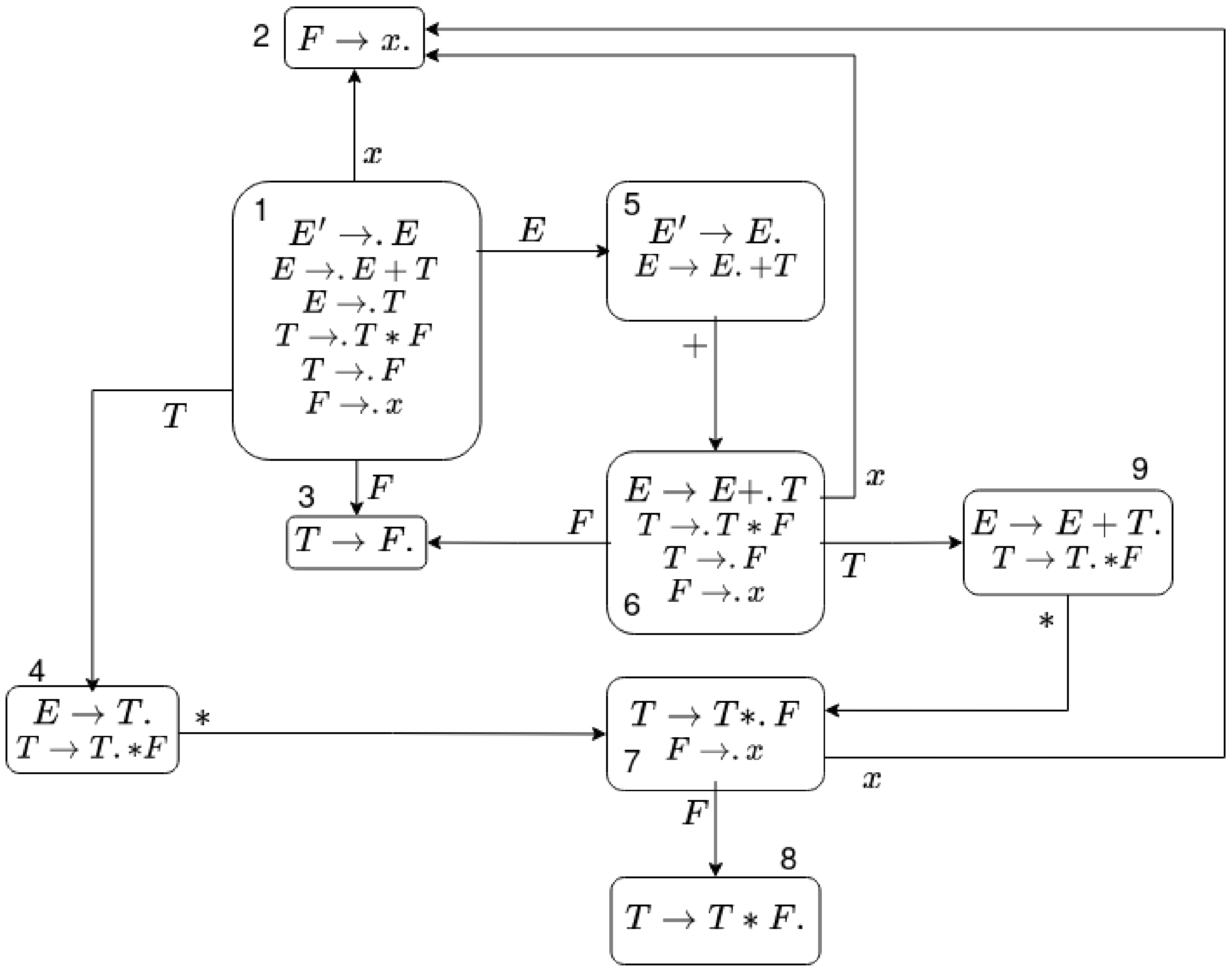
First, we'll have to augment the grammar so that we can uniquely tell if we are just starting or finishing. We then get the grammar:

$$\begin{aligned} E' &\rightarrow E \\ E &\rightarrow E + T | T \\ T &\rightarrow T * F | F \\ F &\rightarrow x \end{aligned}$$

Making $E' \rightarrow E$ as our base state, we get the closure:

$$\begin{aligned} E' &\rightarrow .E \\ E &\rightarrow .E + T \\ E &\rightarrow .T \\ T &\rightarrow .T * F \\ T &\rightarrow .F \\ F &\rightarrow .x \end{aligned}$$

This is every rule we have, and I will number them I through VI, in the order they are as above. We do this for every shift, and end up with the following table:

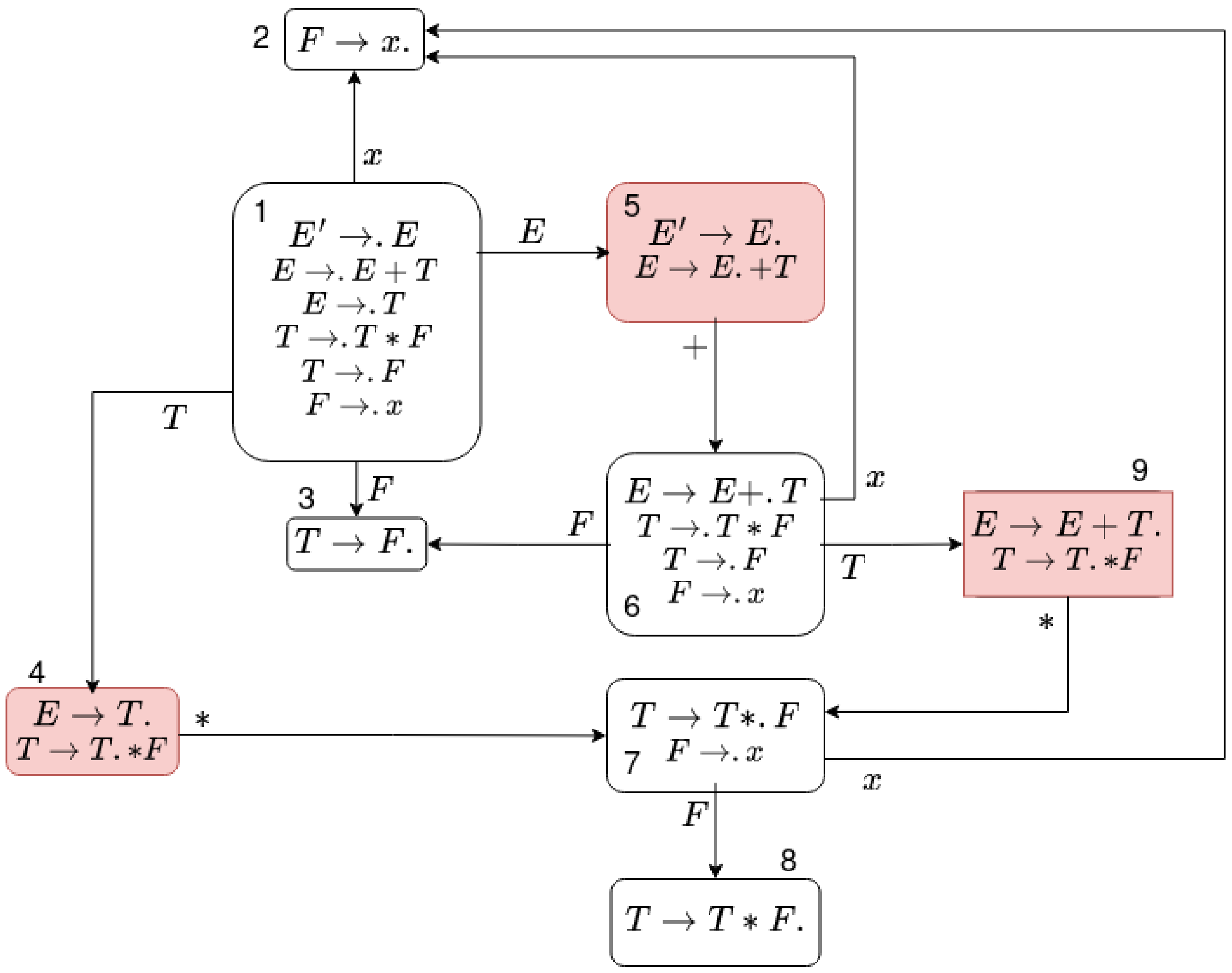


2 $F \to x.$
1 $E' \to .E$ $E \to .E + T$ $E \to .T$ $T \to .T * F$ $T \to .F$ $F \to .x$
5 $E' \to E.$ $E \to E.+T$
3 $T \to F.$
6 $E \to E+.T$ $T \to .T * F$ $T \to .F$ $F \to .x$
9 $E \to E + T.$ $T \to T.*F$
4 $E \to T.$ $T \to T.*F$
7 $T \to T*.F$ $F \to .x$
8 $T \to T * F.$
x
E
T
F
$+$
x
F
T
$*$
$*$
x
F

Figure 1: LR(0) automata for the grammar

Subtask 2: Shift/Reduce conflicts

See the image below. State 4 and 5, highlighted in red have shift/reduce conflicts. For state 4, we can reduce the T to and E, but also shift the $*$. For state 5 we can reduce E to E', but we can also shift $+$.



2 $F \to x.$
x
1 $E' \to .E$ $E \to .E+T$ $E \to .T$ $T \to .T*F$ $T \to .F$ $F \to .x$
E
5 $E' \to E.$ $E \to E.+T$
$+$
T
3 $T \to F.$
F
F
$E \to E+.T$ $T \to .T*F$ $T \to .F$ $F \to .x$ 6
x
T
9 $E \to E+T.$ $T \to T.*F$
$*$
4 $E \to T.$ $T \to T.*F$
$*$
$T \to T*.F$ 7 $F \to .x$
x
F
8 $T \to T*F.$

Figure 2: LR(0) automata with shift/reduce conflicts highlighted

Subtask 3: SLR Parsable

This is the parsing table for a LR(0) parser. We can more clearly see the shift/reduce conflicts in state 4, 5 and 9.

State	x	$*$	$+$	\$	E	T	F
1	s2				g5	g4	g3
2	rVI	rVI	rVI	rVI			
3	rV	rV	rV	rV			
4	rIII	s7, rIII	rIII	rIII			
5	rI	rI	s6, rI	a			
6	s2					g9	g3
7	s2						g8
8	rIV	rIV	rIV	rIV			
9	rII	s7, rII	rII	rII			

Figure 3: LR(0) parsing table

We need to determine if the grammar is SLR parsable, by buffering the next token only if this token is only a part of the construct of the non-terminal that follows the non-terminal we are currently producing. This is the follow-set for the non-terminals. We only care about the follow-set if the state is indicating a reduction. We therefore need the follow-set of F, T, E and E'.

- $F \rightarrow x.$ (state 2)
  $FOLLOW(F) = \{+, *, \$\}$
- $T \rightarrow F.$ (state 3)
  $FOLLOW(T) = \{+, *, \$\}$
- $E \rightarrow T.$ (state 4)
  $FOLLOW(E) = \{+, \$\}$
- $E' \rightarrow E.$ (state 5)
  $FOLLOW(E') = \{\$\}$
- $T \rightarrow T * F.$ (state 8)
  We have already found $FOLLOW(T)$.
- $E \rightarrow E + T.$ (state 9)
  We have already found $FOLLOW(E)$.

We now reduce only if the terminal is in the follow-set of the rule. This gives the SLR parsing table:

State	x	$*$	$+$	\$	E	T	F
1	s2				g5	g4	g3
2		rVI	rVI	rVI			
3		rV	rV	rV			
4		s7	rIII	rIII			
5			s6	a (rI)			
6	s2					g9	g3
7	s2						g8
8		rIV	rIV	rIV			
9		s7	rII	rII			

Figure 4: LR(0) parsing table

We can see the there no longer are any conflicts, and the SLR parser will be able to decide what to do in all states for all inputs. This means that the language is SLR parsable.

Task 2: Implementation

The implementation details and comments are found in the code repository.