Bottom Up Parsing:
The LR(0) Algorithm

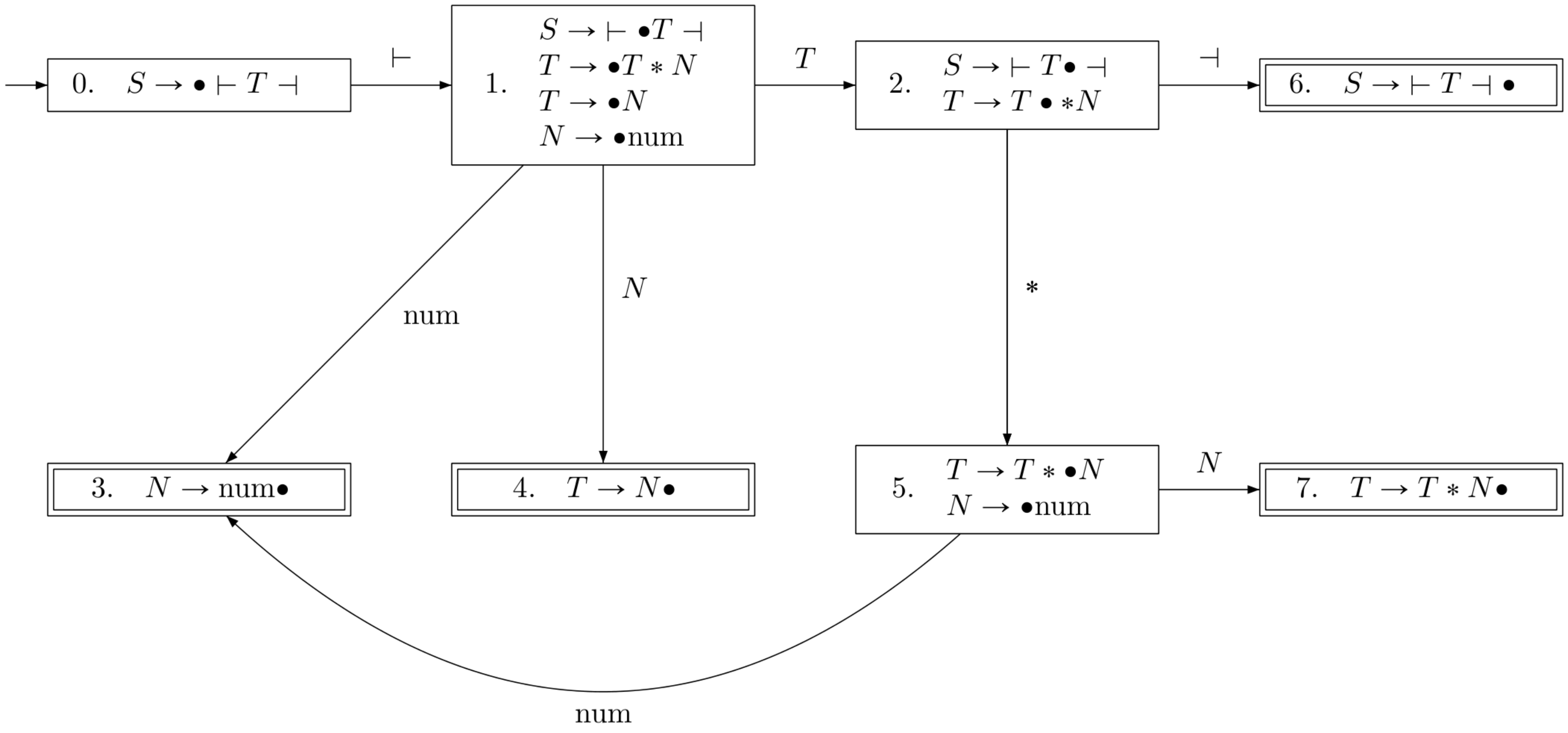
An LR(0) Parsing DFA

S → ⊢ T ⊣

T → T * N

T → N

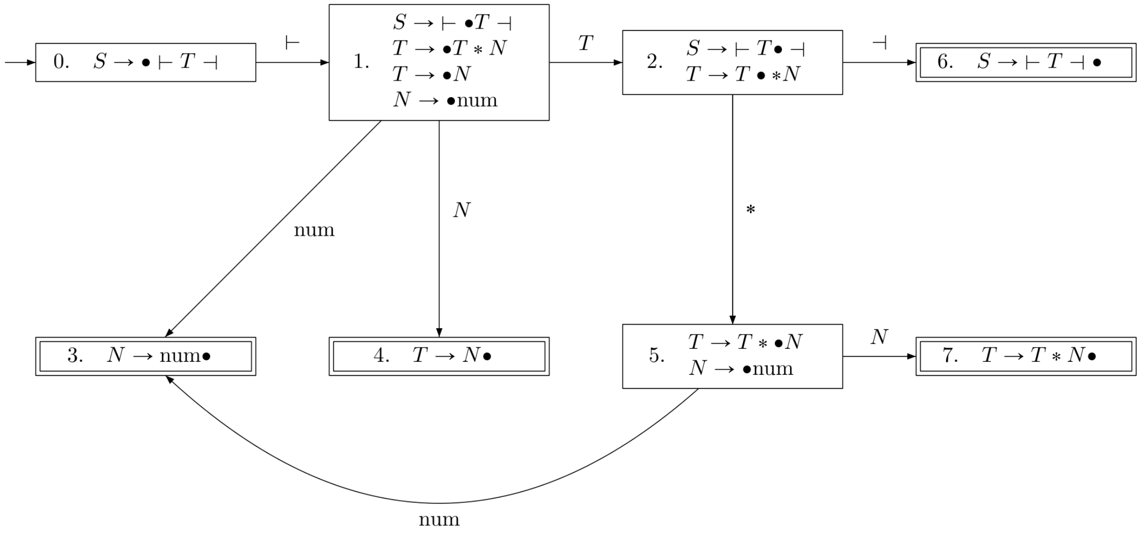
N → num



0. $S \to \bullet \vdash T \dashv$
$\vdash$
1. $S \to \vdash \bullet T \dashv$
$T \to \bullet T * N$
$T \to \bullet N$
$N \to \bullet \mathrm{num}$
T
2. $S \to \vdash T \bullet \dashv$
$T \to T \bullet * N$
$\dashv$
6. $S \to \vdash T \dashv \bullet$
num
N
*
3. $N \to \mathrm{num}\bullet$
4. $T \to N\bullet$
5. $T \to T * \bullet N$
$N \to \bullet \mathrm{num}$
N
7. $T \to T * N\bullet$
num

Revisiting Our Parsing Example...

S → ⊢ T ⊣
T → T * N
T → N
N → num



⊢ num * num * num ⊣	shift ⊢
⊢ num * num * num ⊣	shift num
⊢ **num** * num * num ⊣	reduce N → num
⊢ **N** * num * num ⊣	reduce T → N
⊢ **T** * num * num ⊣	shift *
⊢ **T** ***** num * num ⊣	shift num
⊢ **T** ***** **num** * num ⊣	reduce N → num
⊢ **T** ***** **N** * num ⊣	reduce T → T * N
⊢ **T** * num ⊣	shift *
⊢ **T** ***** num ⊣	shift num
⊢ **T** ***** **num** ⊣	reduce N → num
⊢ **T** ***** **N** ⊣	reduce T → T * N
⊢ **T** ⊣	shift ⊣
⊢ **T** **⊣**	reduce S → ⊢ T ⊣
S	

If we run the shifted portion of the sequence through the DFA, it tells us whether to shift or reduce, depending on whether we end up in an accepting or non-accepting state!!

LR(0) Algorithm: Basic Version

- We will refer to the portion of input that has been shifted/reduced so far as the "reduction sequence".
- We now have a way to implement the LR(0) algorithm:
    - Augment the grammar and construct the parsing DFA.
    - Repeat the following steps until ⊣ is shifted.
        - Run the current reduction sequence through the DFA.
        - If we end up in an accepting state, reduce by the rule contained in the state.
        - If we end up in a non-accepting state, shift a symbol from input.
    - A parsing error occurs if we get "stuck" in the DFA (no transition on a symbol).
    - If our grammar is augmented, we know the parse is successful as soon as we shift the final symbol ⊣.

Efficiency Concerns

- We won't do a full example of the algorithm just presented because it is actually a **quadratic time** algorithm.
- At every step, to decide whether to shift or reduce, we need to run the *entire* reduction sequence through the DFA.
- The length of the reduction sequence is potentially proportional to the length of the input, and the number of times we run it through the DFA is also proportional to the length of the input.
- Just like in LL(1) parsing, we will use a **stack** to deal with this inefficiency.

LR(0) Parsing with a Stack: The Idea

- Shifting a symbol makes the reduction sequence one symbol longer, which means after we shift, we get one state further in the DFA compared to before.
- So instead of running the whole sequence through the DFA every time, we keep track of the **current state**, and shifting makes us transition to the next state.
- What about reducing? We *remove* symbols (the RHS of the rule) from the reduction sequence, which is like *backtracking* in the DFA, then *add* the LHS, which is like going forward again.
- We use a stack to track the **history** of DFA states we passed through.

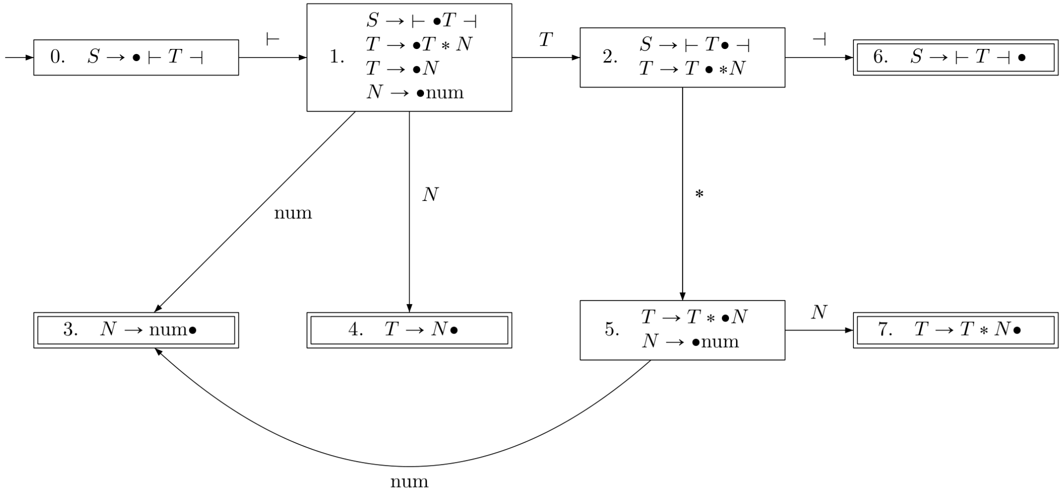
LR(0) Parsing with a Stack: The Details

- We maintain a **state stack** which is synchronized with our reduction sequence. If the reduction sequence contains *n* symbols, the state stack contains *n + 1* states.
- Initially, the state stack contains the **initial state** of the parsing DFA, and the reduction sequence is empty.
- **Shifting:**
    - Look at the top of the state stack (current state).
    - Take the transition on the next input symbol.
    - If there is no transition on the next input symbol, this is a **parsing error**.
    - Otherwise, push the new state to the state stack. Then, consume the next input symbol and push it to the reduction sequence.

LR(0) Parsing with a Stack: The Details

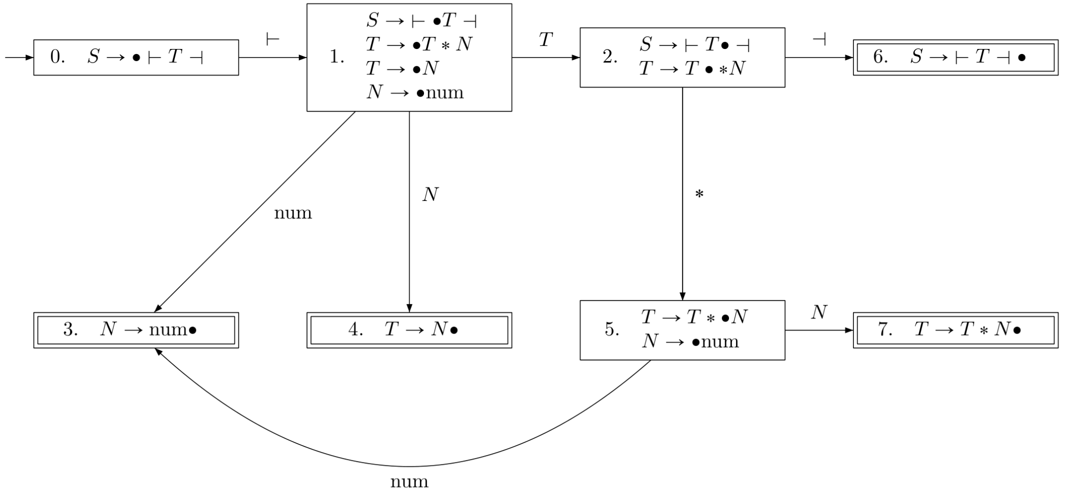
- We maintain a **state stack** which is synchronized with our reduction sequence. If the reduction sequence contains *n* symbols, the state stack contains *n + 1* states.
- Initially, the state stack contains the **initial state** of the parsing DFA, and the reduction sequence is empty.
- **Reducing by a rule A → α** where α is a string (the RHS of a rule):
  - Pop each symbols in **α** from the reduction sequence, and pop *the same number of states* from the state stack to keep them synchronized.
  - Look at the *current top of the state stack* (after popping). Take the transition from this state on **A** (the LHS of the rule). Push the new state to the state stack, and push **A** to the reduction sequence.

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	


0. $S \to \bullet \vdash T \dashv$
⊢
1. $S \to \vdash \bullet T \dashv$ $T \to \bullet T * N$ $T \to \bullet N$ $N \to \bullet \text{num}$
T
2. $S \to \vdash T \bullet \dashv$ $T \to T \bullet * N$
⊣
6. $S \to \vdash T \dashv \bullet$
num
N
*
3. $N \to \text{num}\bullet$
4. $T \to N \bullet$
5. $T \to T * \bullet N$ $N \to \bullet \text{num}$
N
7. $T \to T * N \bullet$
num

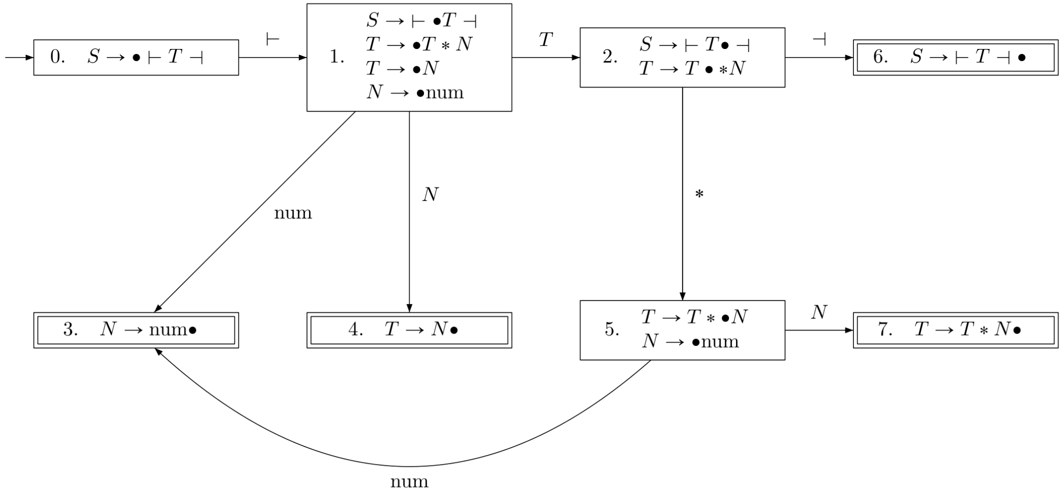
St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	


0. $S \to \bullet \vdash T \dashv$
$\vdash$
1. $S \to \vdash \bullet T \dashv$
$T \to \bullet T * N$
$T \to \bullet N$
$N \to \bullet \text{num}$
T
2. $S \to \vdash T \bullet \dashv$
$T \to T \bullet * N$
$\dashv$
6. $S \to \vdash T \dashv \bullet$
num
N
*
3. $N \to \text{num}\bullet$
4. $T \to N \bullet$
5. $T \to T * \bullet N$
$N \to \bullet \text{num}$
N
7. $T \to T * N \bullet$
num

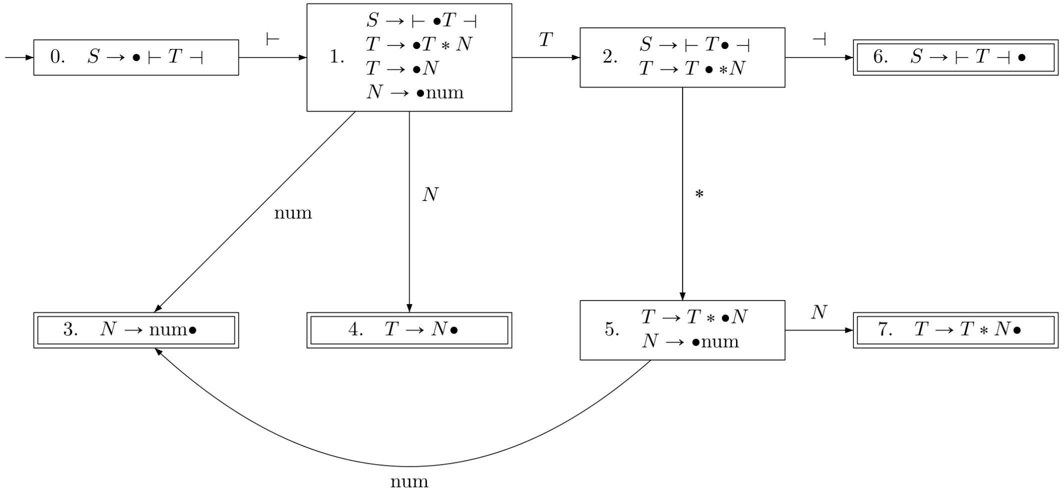
St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	


0. $S \to \bullet \vdash T \dashv$
$\vdash$
1. $S \to \vdash \bullet T \dashv$
$T \to \bullet T * N$
$T \to \bullet N$
$N \to \bullet \mathrm{num}$
T
2. $S \to \vdash T \bullet \dashv$
$T \to T \bullet * N$
$\dashv$
6. $S \to \vdash T \dashv \bullet$
num
N
$*$
3. $N \to \mathrm{num} \bullet$
4. $T \to N \bullet$
5. $T \to T * \bullet N$
$N \to \bullet \mathrm{num}$
N
7. $T \to T * N \bullet$
num

St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num



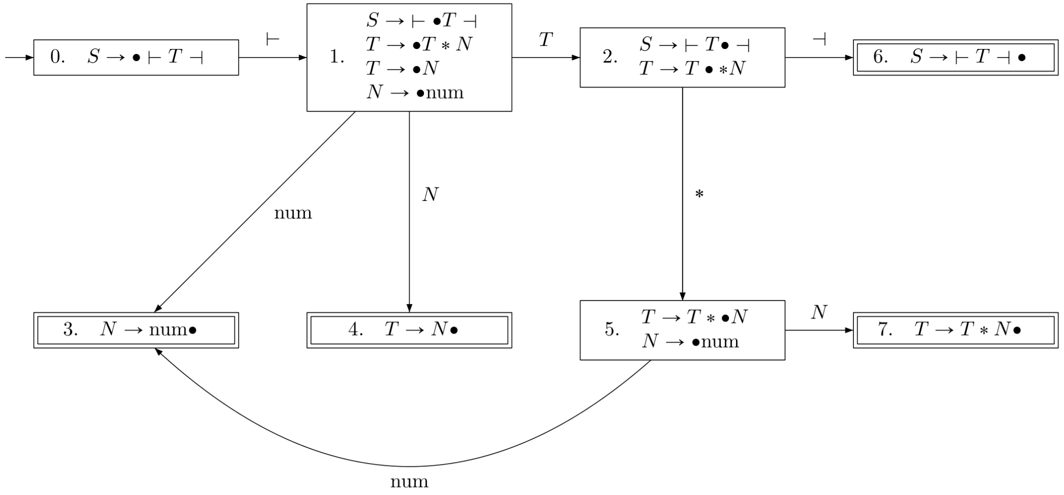
$0.\quad S \to \bullet \vdash T \dashv$
$\vdash$
$1.\quad S \to\ \vdash \bullet T \dashv$
$T \to \bullet T * N$
$T \to \bullet N$
$N \to \bullet \mathrm{num}$
T
$2.\quad S \to\ \vdash T \bullet \dashv$
$T \to T \bullet * N$
$\dashv$
$6.\quad S \to\ \vdash T \dashv \bullet$
num
N
$*$
$3.\quad N \to \mathrm{num}\bullet$
$4.\quad T \to N\bullet$
$5.\quad T \to T * \bullet N$
$N \to \bullet \mathrm{num}$
N
$7.\quad T \to T * N\bullet$
num

St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.

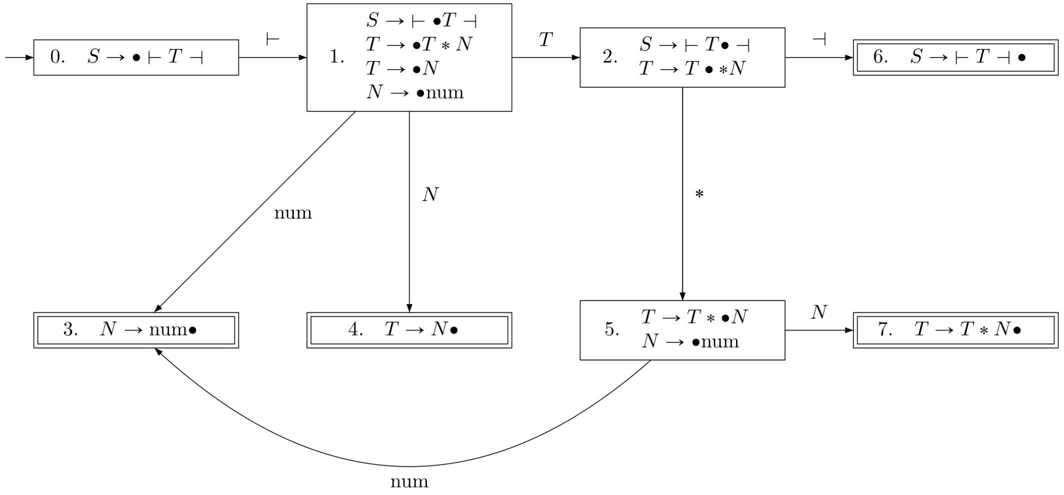


St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.



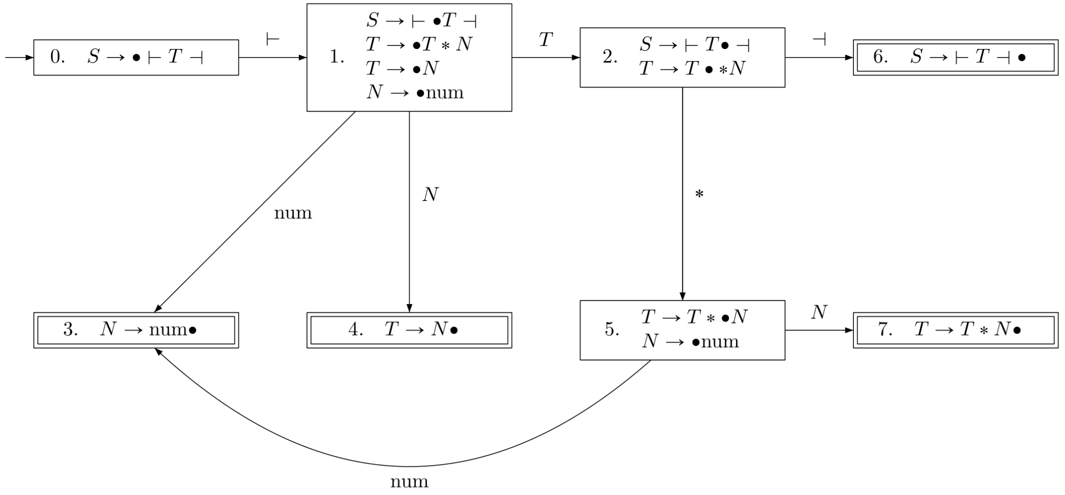
0. $S \to \bullet \vdash T \dashv$
1. $S \to \vdash \bullet T \dashv$
$T \to \bullet T * N$
$T \to \bullet N$
$N \to \bullet \mathrm{num}$
2. $S \to \vdash T \bullet \dashv$
$T \to T \bullet * N$
6. $S \to \vdash T \dashv \bullet$
3. $N \to \mathrm{num}\bullet$
4. $T \to N \bullet$
5. $T \to T * \bullet N$
$N \to \bullet \mathrm{num}$
7. $T \to T * N \bullet$
⊢
T
⊣
num
N
*
N
num

St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4
0 1 4	⊢ N	* num ⊣	

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.



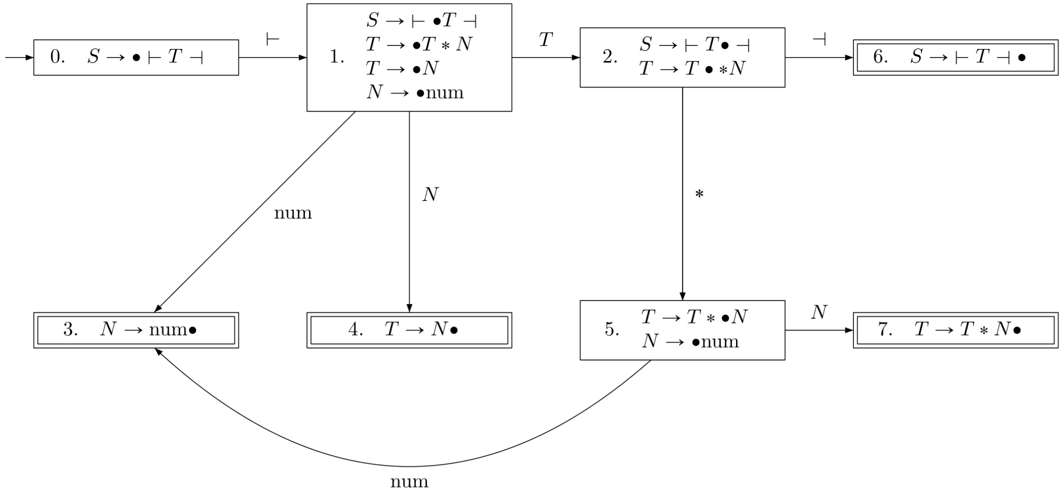
0. $S \to \bullet \vdash T \dashv$
$\vdash$
1. $S \to \vdash \bullet T \dashv$
$T \to \bullet T * N$
$T \to \bullet N$
$N \to \bullet \mathrm{num}$
T
2. $S \to \vdash T \bullet \dashv$
$T \to T \bullet * N$
$\dashv$
6. $S \to \vdash T \dashv \bullet$
num
N
$*$
3. $N \to \mathrm{num} \bullet$
4. $T \to N \bullet$
5. $T \to T * \bullet N$
$N \to \bullet \mathrm{num}$
N
7. $T \to T * N \bullet$
num

St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4
0 1 4	⊢ N	* num ⊣	Reduce T → N

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.



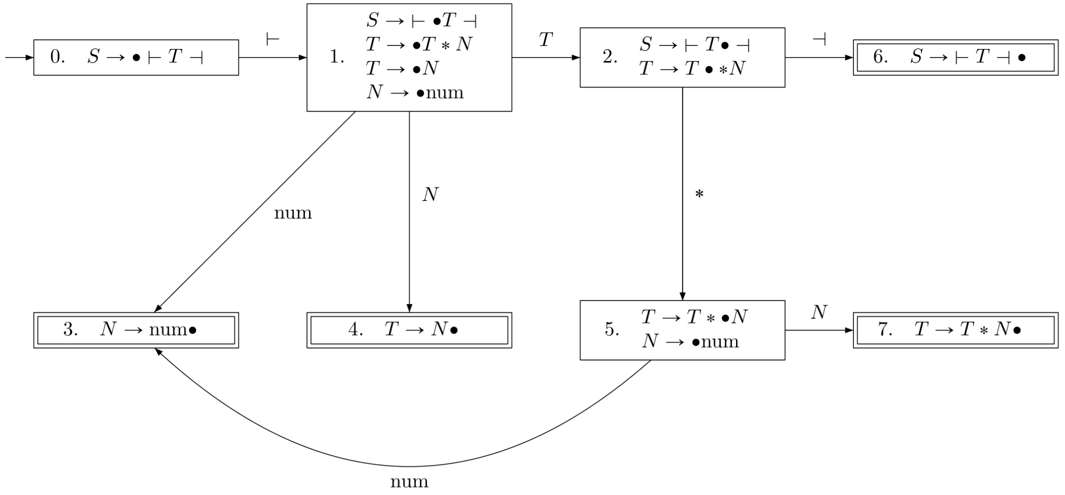
0. $S \to \bullet \vdash T \dashv$
$\vdash$
1. $S \to \vdash \bullet T \dashv$
$T \to \bullet T * N$
$T \to \bullet N$
$N \to \bullet \mathrm{num}$
T
2. $S \to \vdash T \bullet \dashv$
$T \to T \bullet * N$
$\dashv$
6. $S \to \vdash T \dashv \bullet$
num
N
*
3. $N \to \mathrm{num}\bullet$
4. $T \to N \bullet$
5. $T \to T * \bullet N$
$N \to \bullet \mathrm{num}$
N
7. $T \to T * N \bullet$
num

St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4
0 1 4	⊢ N	* num ⊣	Reduce T → N
0 1	⊢	* num ⊣	Pop N / 4

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.



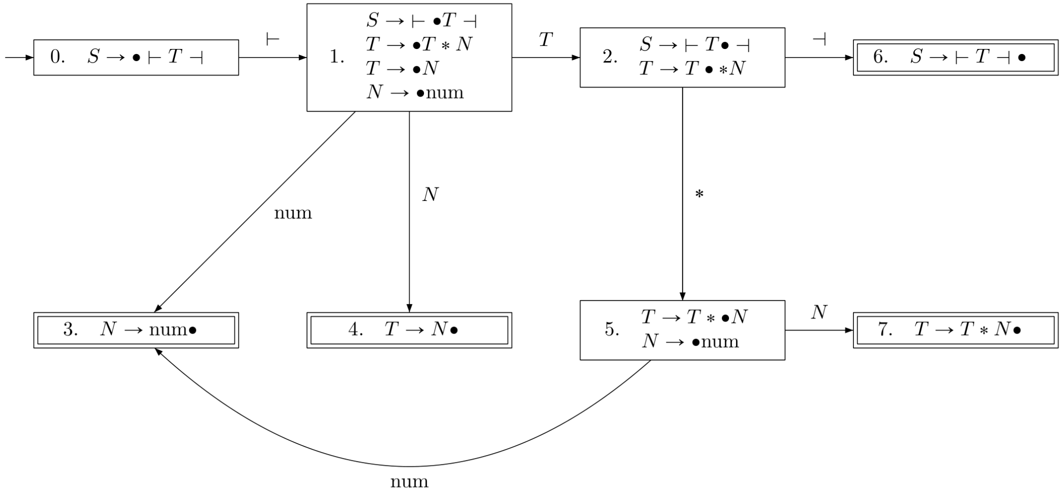
0. $S \to \bullet \vdash T \dashv$
1. $S \to \vdash \bullet T \dashv$
$T \to \bullet T * N$
$T \to \bullet N$
$N \to \bullet \mathrm{num}$
2. $S \to \vdash T \bullet \dashv$
$T \to T \bullet * N$
6. $S \to \vdash T \dashv \bullet$
3. $N \to \mathrm{num} \bullet$
4. $T \to N \bullet$
5. $T \to T * \bullet N$
$N \to \bullet \mathrm{num}$
7. $T \to T * N \bullet$
⊢
T
⊣
num
N
*
N
num

St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4
0 1 4	⊢ N	* num ⊣	Reduce T → N
0 1	⊢	* num ⊣	Pop N / 4
0 1 2	⊢ T	* num ⊣	Push T / 2

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.

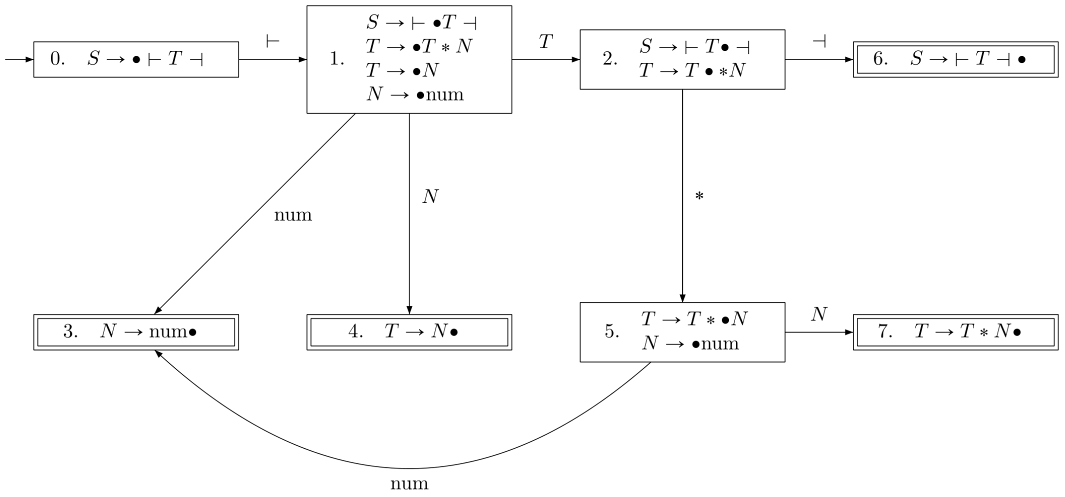

0. $S \to \bullet \vdash T \dashv$
⊢
1. $S \to \vdash \bullet T \dashv$ $T \to \bullet T * N$ $T \to \bullet N$ $N \to \bullet \text{num}$
T
2. $S \to \vdash T \bullet \dashv$ $T \to T \bullet * N$
⊣
6. $S \to \vdash T \dashv \bullet$
num
N
*
3. $N \to \text{num}\bullet$
4. $T \to N\bullet$
5. $T \to T * \bullet N$ $N \to \bullet \text{num}$
N
7. $T \to T * N\bullet$
num

St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4
0 1 4	⊢ N	* num ⊣	Reduce T → N
0 1	⊢	* num ⊣	Pop N / 4
0 1 2	⊢ T	* num ⊣	Push T / 2
0 1 2	⊢ T	* num ⊣	

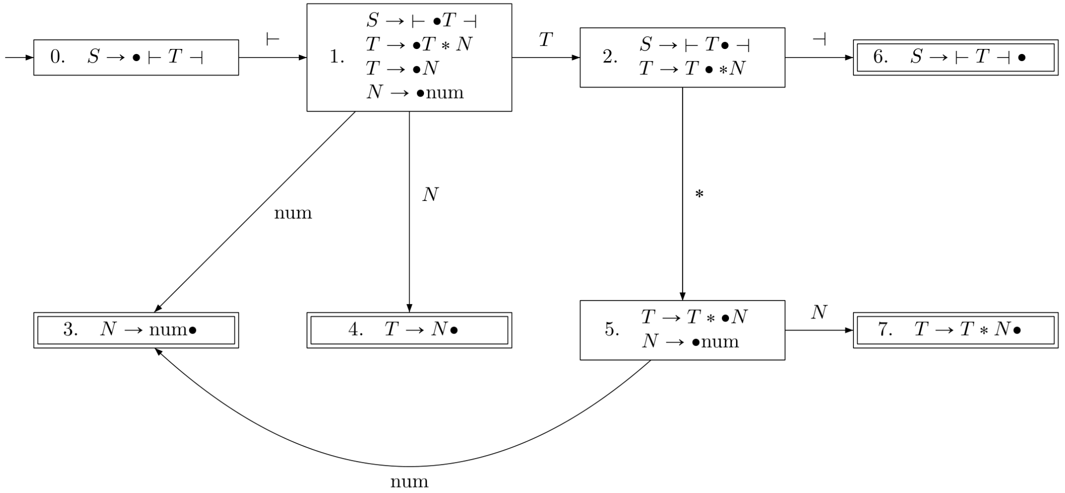


0. $S \to \bullet \vdash T \dashv$
$\vdash$
1. $S \to \vdash \bullet T \dashv$
$T \to \bullet T * N$
$T \to \bullet N$
$N \to \bullet \mathrm{num}$
T
2. $S \to \vdash T \bullet \dashv$
$T \to T \bullet * N$
$\dashv$
6. $S \to \vdash T \dashv \bullet$
num
N
$*$
3. $N \to \mathrm{num} \bullet$
4. $T \to N \bullet$
5. $T \to T * \bullet N$
$N \to \bullet \mathrm{num}$
N
7. $T \to T * N \bullet$
num

St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4
0 1 4	⊢ N	* num ⊣	Reduce T → N
0 1	⊢	* num ⊣	Pop N / 4
0 1 2	⊢ T	* num ⊣	Push T / 2
0 1 2	⊢ T	* num ⊣	Shift * / 5
0 1 2 5	⊢ T *	num ⊣	

On exams, don't write the orange Push/Pop rows.
These represent intermediate steps of a Reduce.



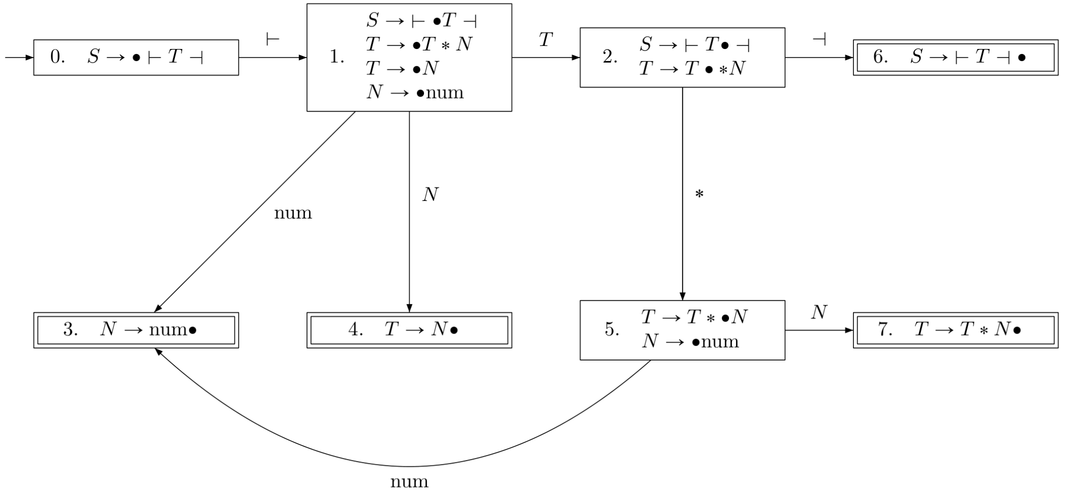
0. $S \to \bullet \vdash T \dashv$
$\vdash$
1. $S \to \vdash \bullet T \dashv$, $T \to \bullet T * N$, $T \to \bullet N$, $N \to \bullet\mathrm{num}$
T
2. $S \to \vdash T \bullet \dashv$, $T \to T \bullet * N$
$\dashv$
6. $S \to \vdash T \dashv \bullet$
num
N
$*$
3. $N \to \mathrm{num}\bullet$
4. $T \to N \bullet$
5. $T \to T * \bullet N$, $N \to \bullet\mathrm{num}$
N
7. $T \to T * N \bullet$
num

St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4
0 1 4	⊢ N	* num ⊣	Reduce T → N
0 1	⊢	* num ⊣	Pop N / 4
0 1 2	⊢ T	* num ⊣	Push T / 2
0 1 2	⊢ T	* num ⊣	Shift * / 5
0 1 2 5	⊢ T *	num ⊣	Shift num / 3
0 1 2 5 3	⊢ T * num	⊣	

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.

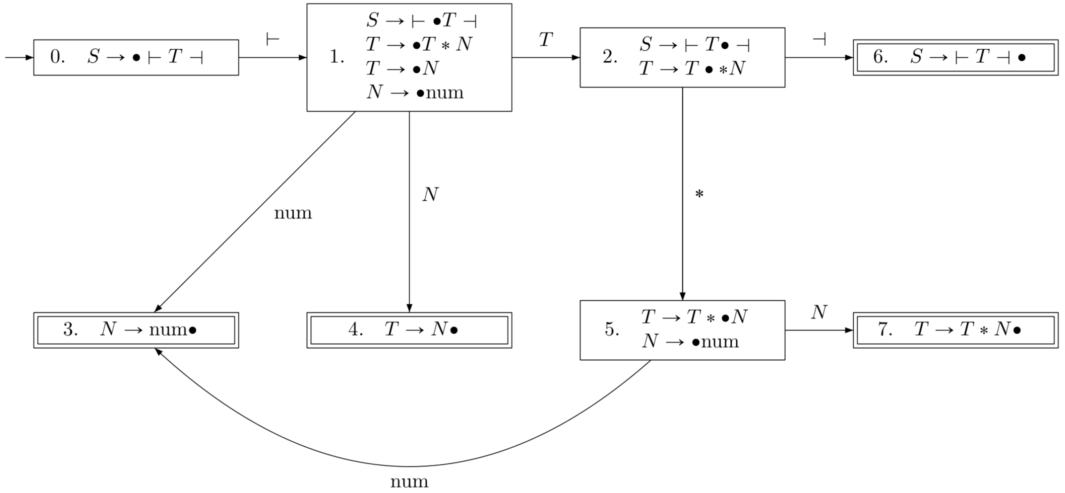


St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4
0 1 4	⊢ N	* num ⊣	Reduce T → N
0 1	⊢	* num ⊣	Pop N / 4
0 1 2	⊢ T	* num ⊣	Push T / 2
0 1 2	⊢ T	* num ⊣	Shift * / 5
0 1 2 5	⊢ T *	num ⊣	Shift num / 3
0 1 2 5 3	⊢ T * num	⊣	Reduce N → num

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.

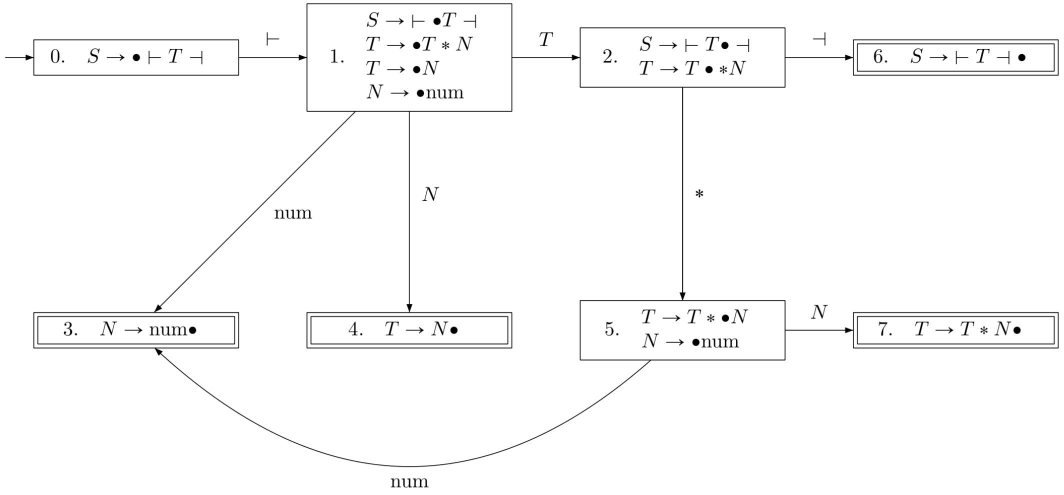

0. $S \to \bullet \vdash T \dashv$
⊢
1. $S \to \vdash \bullet T \dashv$
$T \to \bullet T * N$
$T \to \bullet N$
$N \to \bullet\mathrm{num}$
T
2. $S \to \vdash T \bullet \dashv$
$T \to T \bullet * N$
⊣
6. $S \to \vdash T \dashv \bullet$
num
N
*
3. $N \to \mathrm{num}\bullet$
4. $T \to N \bullet$
5. $T \to T * \bullet N$
$N \to \bullet\mathrm{num}$
N
7. $T \to T * N \bullet$
num

St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4
0 1 4	⊢ N	* num ⊣	Reduce T → N
0 1	⊢	* num ⊣	Pop N / 4
0 1 2	⊢ T	* num ⊣	Push T / 2
0 1 2	⊢ T	* num ⊣	Shift * / 5
0 1 2 5	⊢ T *	num ⊣	Shift num / 3
0 1 2 5 3	⊢ T * num	⊣	Reduce N → num
0 1 2 5	⊢ T *	⊣	Pop num / 3

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.



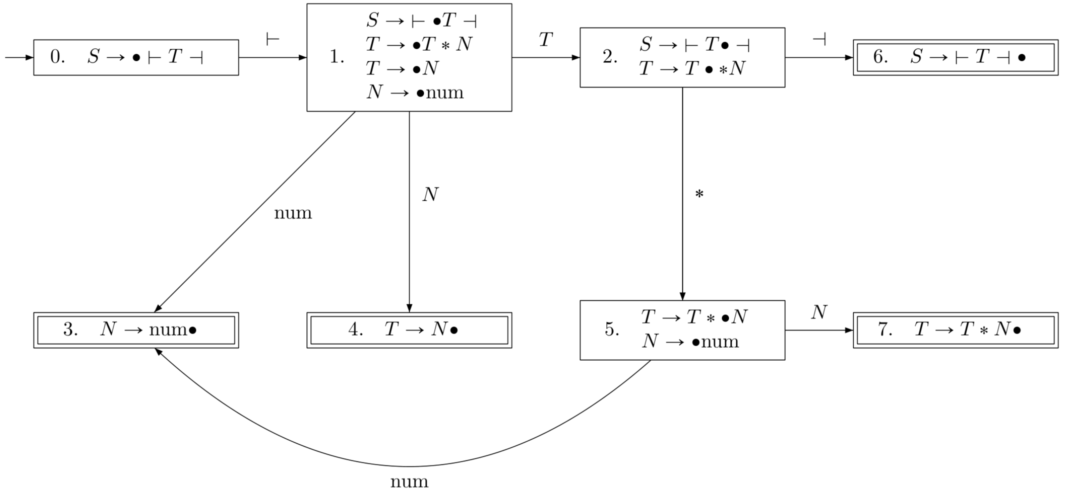
0. $S \to \bullet \vdash T \dashv$
$\vdash$
1. $S \to \vdash \bullet T \dashv$
$T \to \bullet T * N$
$T \to \bullet N$
$N \to \bullet \text{num}$
T
2. $S \to \vdash T \bullet \dashv$
$T \to T \bullet * N$
$\dashv$
6. $S \to \vdash T \dashv \bullet$
num
N
$*$
3. $N \to \text{num}\bullet$
4. $T \to N \bullet$
5. $T \to T * \bullet N$
$N \to \bullet \text{num}$
N
7. $T \to T * N \bullet$
num

St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4
0 1 4	⊢ N	* num ⊣	Reduce T → N
0 1	⊢	* num ⊣	Pop N / 4
0 1 2	⊢ T	* num ⊣	Push T / 2
0 1 2	⊢ T	* num ⊣	Shift * / 5
0 1 2 5	⊢ T *	num ⊣	Shift num / 3
0 1 2 5 3	⊢ T * num	⊣	Reduce N → num
0 1 2 5	⊢ T *	⊣	Pop num / 3
0 1 2 5 7	⊢ T * N	⊣	Push N / 7

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.

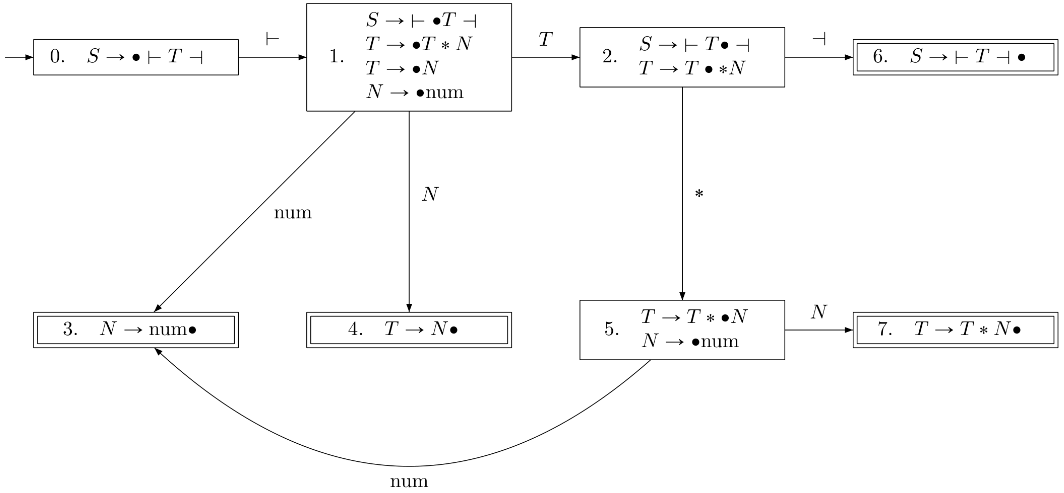

0. $S \to \bullet \vdash T \dashv$
⊢
1. $S \to \vdash \bullet T \dashv$
$T \to \bullet T * N$
$T \to \bullet N$
$N \to \bullet \text{num}$
T
2. $S \to \vdash T \bullet \dashv$
$T \to T \bullet * N$
⊣
6. $S \to \vdash T \dashv \bullet$
num
N
*
3. $N \to \text{num}\bullet$
4. $T \to N \bullet$
5. $T \to T * \bullet N$
$N \to \bullet \text{num}$
N
7. $T \to T * N \bullet$
num

St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4
0 1 4	⊢ N	* num ⊣	Reduce T → N
0 1	⊢	* num ⊣	Pop N / 4
0 1 2	⊢ T	* num ⊣	Push T / 2
0 1 2	⊢ T	* num ⊣	Shift * / 5
0 1 2 5	⊢ T *	num ⊣	Shift num / 3
0 1 2 5 3	⊢ T * num	⊣	Reduce N → num
0 1 2 5	⊢ T *	⊣	Pop num / 3
0 1 2 5 7	⊢ T * N	⊣	Push N / 7
0 1 2 5 7	⊢ T * N	⊣	

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.

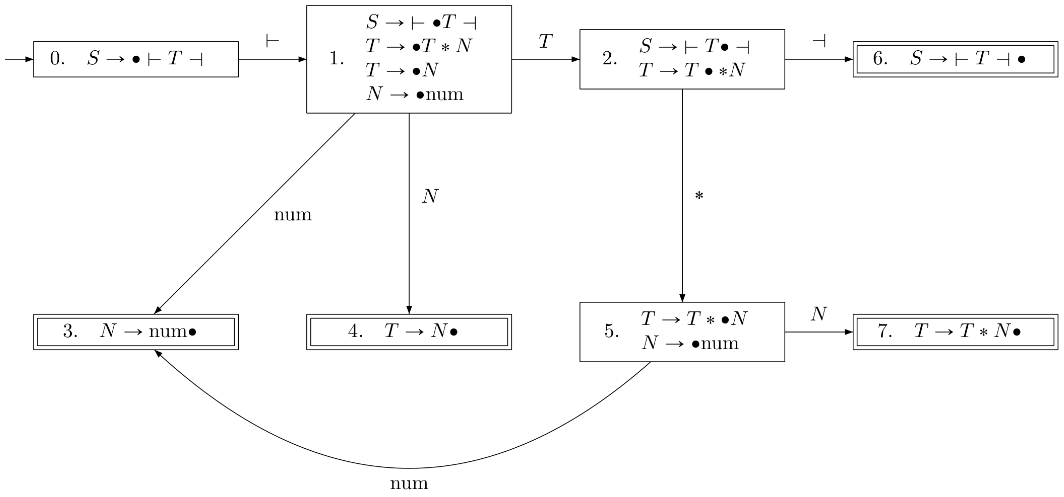

0. $S \to \bullet \vdash T \dashv$
$\vdash$
1. $S \to \vdash \bullet T \dashv$
$T \to \bullet T * N$
$T \to \bullet N$
$N \to \bullet \text{num}$
T
2. $S \to \vdash T \bullet \dashv$
$T \to T \bullet * N$
$\dashv$
6. $S \to \vdash T \dashv \bullet$
num
N
$*$
3. $N \to \text{num} \bullet$
4. $T \to N \bullet$
5. $T \to T * \bullet N$
$N \to \bullet \text{num}$
N
7. $T \to T * N \bullet$
num

St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4
0 1 4	⊢ N	* num ⊣	Reduce T → N
0 1	⊢	* num ⊣	Pop N / 4
0 1 2	⊢ T	* num ⊣	Push T / 2
0 1 2	⊢ T	* num ⊣	Shift * / 5
0 1 2 5	⊢ T *	num ⊣	Shift num / 3
0 1 2 5 3	⊢ T * num	⊣	Reduce N → num
0 1 2 5	⊢ T *	⊣	Pop num / 3
0 1 2 5 7	⊢ T * N	⊣	Push N / 7
0 1 2 5 7	⊢ T * N	⊣	Reduce T → T * N

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.

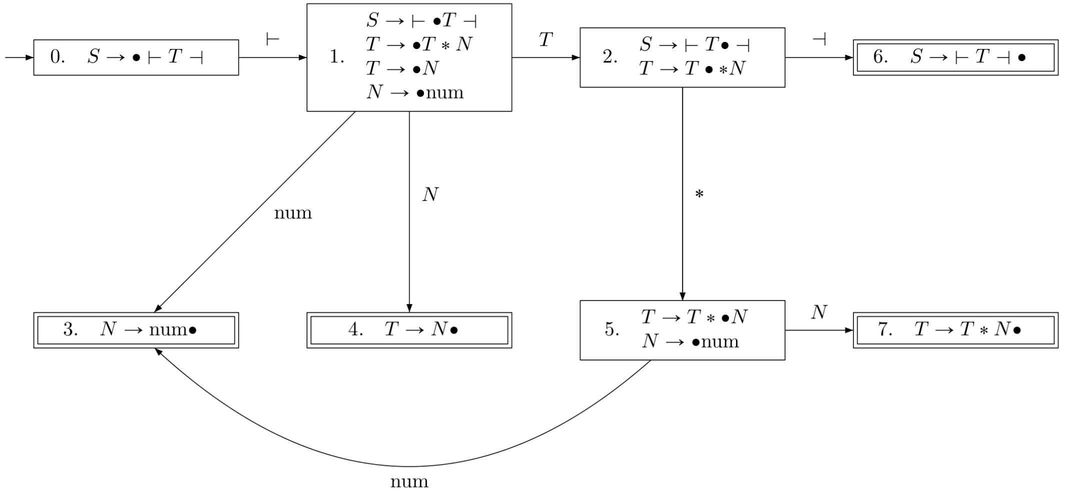


St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4
0 1 4	⊢ N	* num ⊣	Reduce T → N
0 1	⊢	* num ⊣	Pop N / 4
0 1 2	⊢ T	* num ⊣	Push T / 2
0 1 2	⊢ T	* num ⊣	Shift * / 5
0 1 2 5	⊢ T *	num ⊣	Shift num / 3
0 1 2 5 3	⊢ T * num	⊣	Reduce N → num
0 1 2 5	⊢ T *	⊣	Pop num / 3
0 1 2 5 7	⊢ T * N	⊣	Push N / 7
0 1 2 5 7	⊢ T * N	⊣	Reduce T → T * N
0 1 2 5	⊢ T *	⊣	Pop N / 7

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.

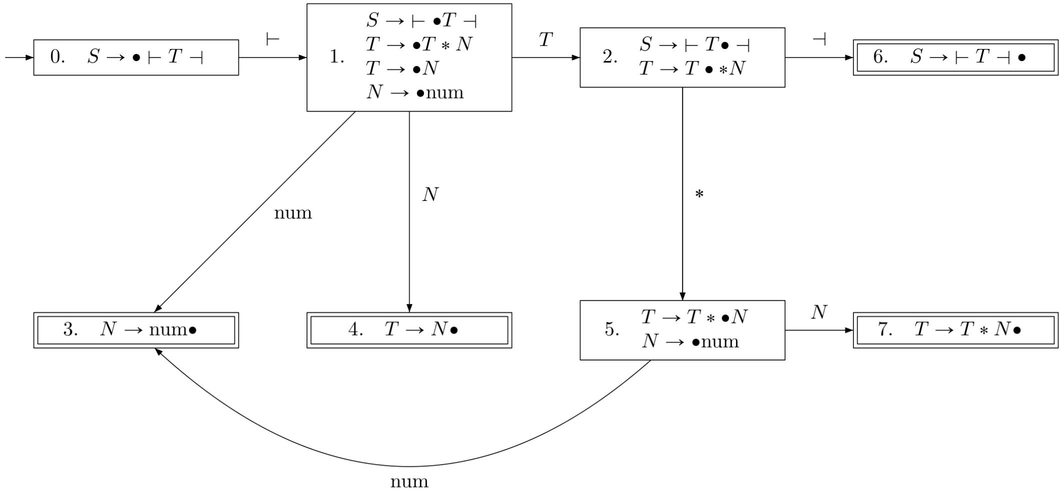

0. $S \to \bullet \vdash T \dashv$
$\vdash$
1. $S \to \vdash \bullet T \dashv$
$T \to \bullet T * N$
$T \to \bullet N$
$N \to \bullet \text{num}$
T
2. $S \to \vdash T \bullet \dashv$
$T \to T \bullet * N$
$\dashv$
6. $S \to \vdash T \dashv \bullet$
num
N
$*$
3. $N \to \text{num}\bullet$
4. $T \to N\bullet$
5. $T \to T * \bullet N$
$N \to \bullet \text{num}$
N
7. $T \to T * N\bullet$
num

St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4
0 1 4	⊢ N	* num ⊣	Reduce T → N
0 1	⊢	* num ⊣	Pop N / 4
0 1 2	⊢ T	* num ⊣	Push T / 2
0 1 2	⊢ T	* num ⊣	Shift * / 5
0 1 2 5	⊢ T *	num ⊣	Shift num / 3
0 1 2 5 3	⊢ T * num	⊣	Reduce N → num
0 1 2 5	⊢ T *	⊣	Pop num / 3
0 1 2 5 7	⊢ T * N	⊣	Push N / 7
0 1 2 5 7	⊢ T * N	⊣	Reduce T → T * N
0 1 2 5	⊢ T *	⊣	Pop N / 7
0 1 2	⊢ T	⊣	Pop * / 5

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.

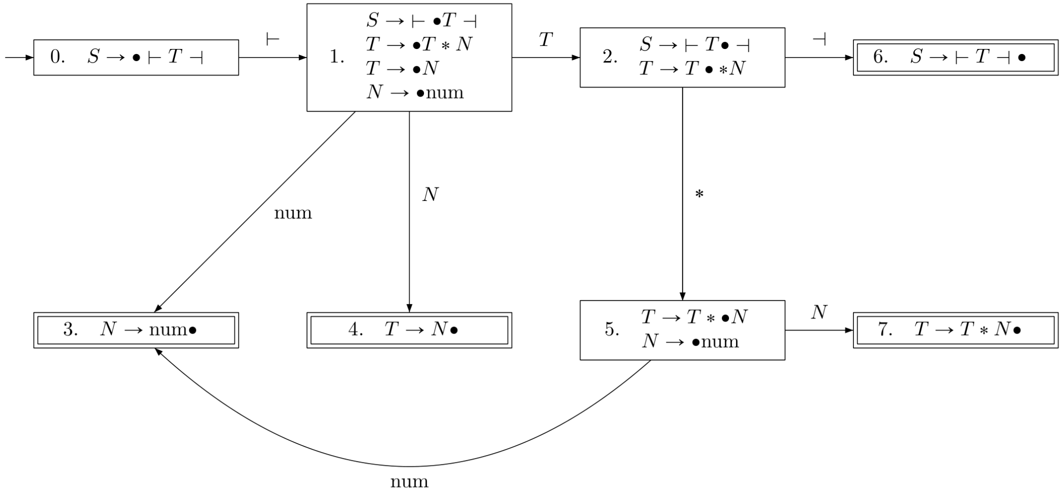


St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4
0 1 4	⊢ N	* num ⊣	Reduce T → N
0 1	⊢	* num ⊣	Pop N / 4
0 1 2	⊢ T	* num ⊣	Push T / 2
0 1 2	⊢ T	* num ⊣	Shift * / 5
0 1 2 5	⊢ T *	num ⊣	Shift num / 3
0 1 2 5 3	⊢ T * num	⊣	Reduce N → num
0 1 2 5	⊢ T *	⊣	Pop num / 3
0 1 2 5 7	⊢ T * N	⊣	Push N / 7
0 1 2 5 7	⊢ T * N	⊣	Reduce T → T * N
0 1 2 5	⊢ T *	⊣	Pop N / 7
0 1 2	⊢ T	⊣	Pop * / 5
0 1	⊢	⊣	Pop T / 2

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.

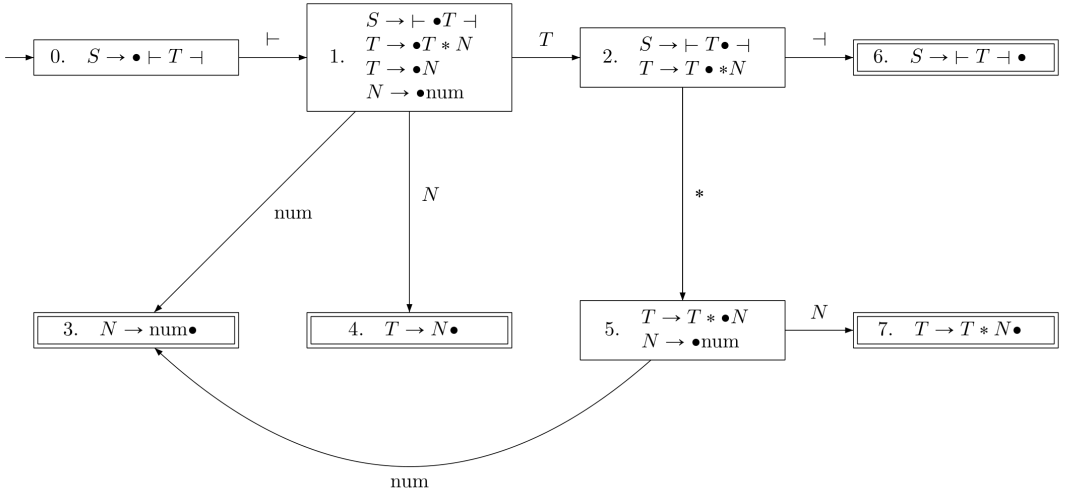


St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4
0 1 4	⊢ N	* num ⊣	Reduce T → N
0 1	⊢	* num ⊣	Pop N / 4
0 1 2	⊢ T	* num ⊣	Push T / 2
0 1 2	⊢ T	* num ⊣	Shift * / 5
0 1 2 5	⊢ T *	num ⊣	Shift num / 3
0 1 2 5 3	⊢ T * num	⊣	Reduce N → num
0 1 2 5	⊢ T *	⊣	Pop num / 3
0 1 2 5 7	⊢ T * N	⊣	Push N / 7
0 1 2 5 7	⊢ T * N	⊣	Reduce T → T * N
0 1 2 5	⊢ T *	⊣	Pop N / 7
0 1 2	⊢ T	⊣	Pop * / 5
0 1	⊢	⊣	Pop T / 2
0 1 2	⊢ T	⊣	Push T / 2

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.



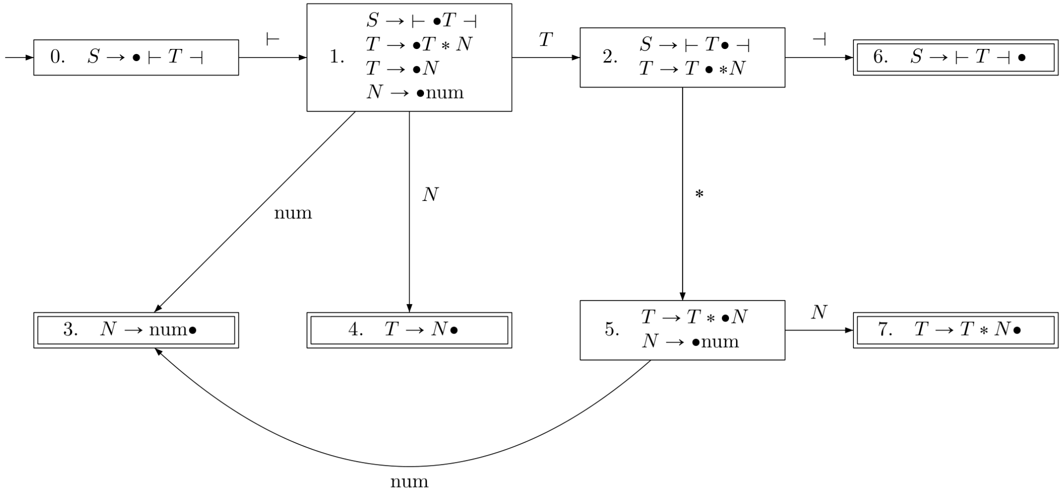
0. $S \to \bullet \vdash T \dashv$
⊢
1. $S \to \vdash \bullet T \dashv$
$T \to \bullet T * N$
$T \to \bullet N$
$N \to \bullet \text{num}$
T
2. $S \to \vdash T \bullet \dashv$
$T \to T \bullet * N$
⊣
6. $S \to \vdash T \dashv \bullet$
num
N
*
3. $N \to \text{num}\bullet$
4. $T \to N \bullet$
5. $T \to T * \bullet N$
$N \to \bullet \text{num}$
N
7. $T \to T * N \bullet$
num

St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4
0 1 4	⊢ N	* num ⊣	Reduce T → N
0 1	⊢	* num ⊣	Pop N / 4
0 1 2	⊢ T	* num ⊣	Push T / 2
0 1 2	⊢ T	* num ⊣	Shift * / 5
0 1 2 5	⊢ T *	num ⊣	Shift num / 3
0 1 2 5 3	⊢ T * num	⊣	Reduce N → num
0 1 2 5	⊢ T *	⊣	Pop num / 3
0 1 2 5 7	⊢ T * N	⊣	Push N / 7
0 1 2 5 7	⊢ T * N	⊣	Reduce T → T * N
0 1 2 5	⊢ T *	⊣	Pop N / 7
0 1 2	⊢ T	⊣	Pop * / 5
0 1	⊢	⊣	Pop T / 2
0 1 2	⊢ T	⊣	Push T / 2
0 1 2	⊢ T	⊣	

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.

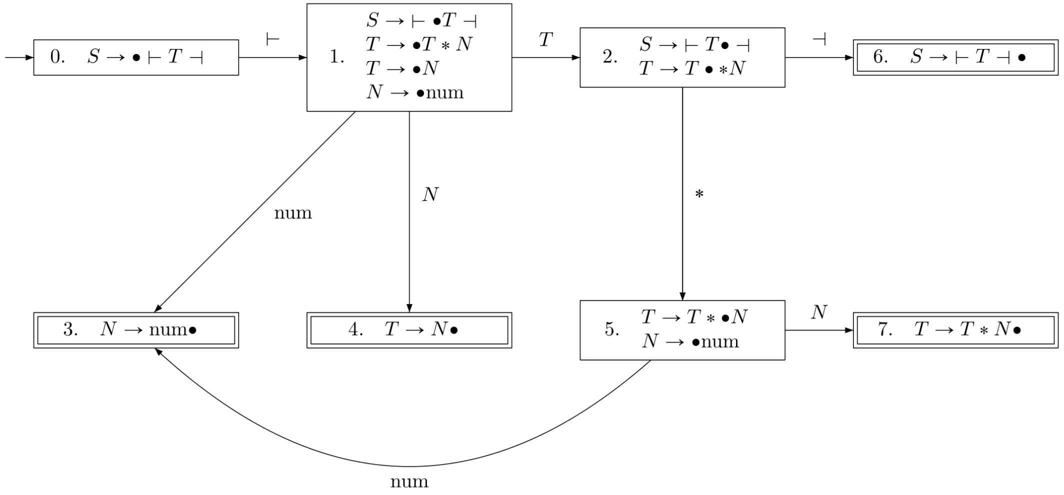

0. $S \to \bullet \vdash T \dashv$
$\vdash$
1. $S \to \vdash \bullet T \dashv$
$T \to \bullet T * N$
$T \to \bullet N$
$N \to \bullet \text{num}$
T
2. $S \to \vdash T \bullet \dashv$
$T \to T \bullet * N$
$\dashv$
6. $S \to \vdash T \dashv \bullet$
num
N
$*$
3. $N \to \text{num}\bullet$
4. $T \to N\bullet$
5. $T \to T * \bullet N$
$N \to \bullet \text{num}$
N
7. $T \to T * N\bullet$
num

St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

State Stack	Reduction Sequence	Unread Input	Action (Shift/Reduce)
0		⊢ num * num ⊣	Shift ⊢ / 1
0 1	⊢	num * num ⊣	Shift num / 3
0 1 3	⊢ num	* num ⊣	Reduce N → num
0 1	⊢	* num ⊣	Pop num / 3
0 1 4	⊢ N	* num ⊣	Push N / 4
0 1 4	⊢ N	* num ⊣	Reduce T → N
0 1	⊢	* num ⊣	Pop N / 4
0 1 2	⊢ T	* num ⊣	Push T / 2
0 1 2	⊢ T	* num ⊣	Shift * / 5
0 1 2 5	⊢ T *	num ⊣	Shift num / 3
0 1 2 5 3	⊢ T * num	⊣	Reduce N → num
0 1 2 5	⊢ T *	⊣	Pop num / 3
0 1 2 5 7	⊢ T * N	⊣	Push N / 7
0 1 2 5 7	⊢ T * N	⊣	Reduce T → T * N
0 1 2 5	⊢ T *	⊣	Pop N / 7
0 1 2	⊢ T	⊣	Pop * / 5
0 1	⊢	⊣	Pop T / 2
0 1 2	⊢ T	⊣	Push T / 2
0 1 2	⊢ T	⊣	Shift ⊣ / 6

On exams, don't write the orange Push/Pop rows.

These represent intermediate steps of a Reduce.



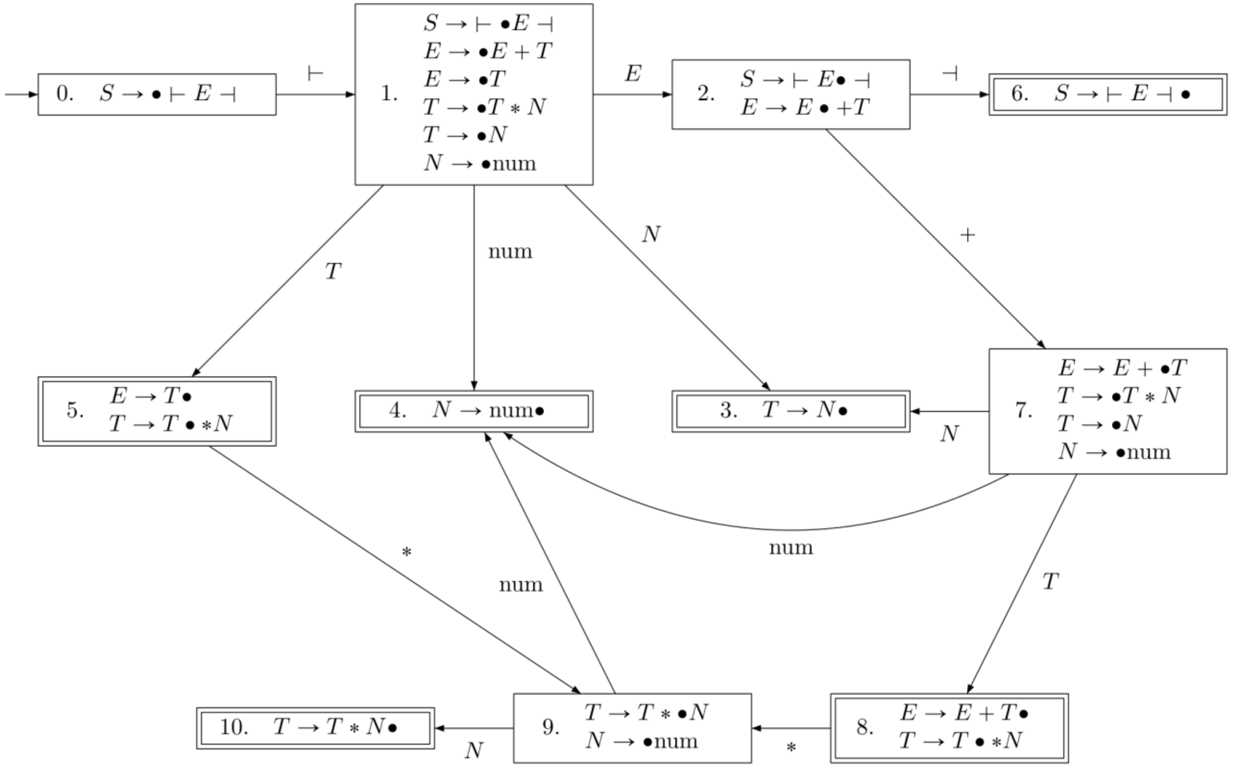
0. $S \to \bullet \vdash T \dashv$
⊢
1. $S \to \vdash \bullet T \dashv$
$T \to \bullet T * N$
$T \to \bullet N$
$N \to \bullet \text{num}$
T
2. $S \to \vdash T \bullet \dashv$
$T \to T \bullet * N$
⊣
6. $S \to \vdash T \dashv \bullet$
num
N
*
3. $N \to \text{num} \bullet$
4. $T \to N \bullet$
5. $T \to T * \bullet N$
$N \to \bullet \text{num}$
N
7. $T \to T * N \bullet$
num

St.	⊢	⊣	T	N	*	num
0	Shift / 1					
1			Shift / 2	Shift / 4		Shift / 3
2		Shift / 6			Shift / 5	
3	Reduce by N → num					
4	Reduce by T → N					
5				Shift / 7		Shift / 3
6	Reduce by S → ⊢ T ⊣					
7	Reduce by T → T * N					

Parsing Conflicts

Here is the LR(0) DFA for a more complicated grammar:

S → ⊢ E ⊣
E → E + T
E → T
T → T * N
T → N
N → num

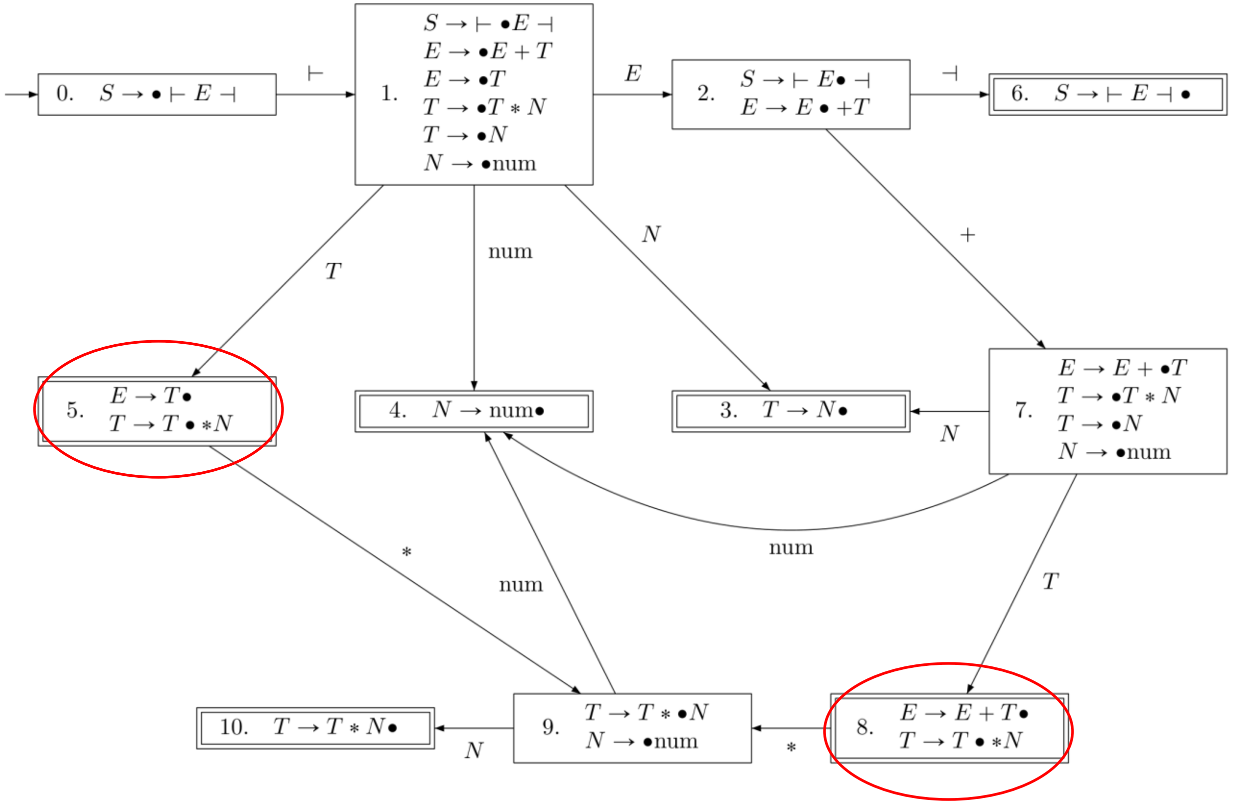


Parsing Conflicts

These accepting states have two items in them.

Only one item is reducible, which means there is only choice of rule we would **reduce** by.

However, the presence of the other rule means **shifting** is also a potential option.

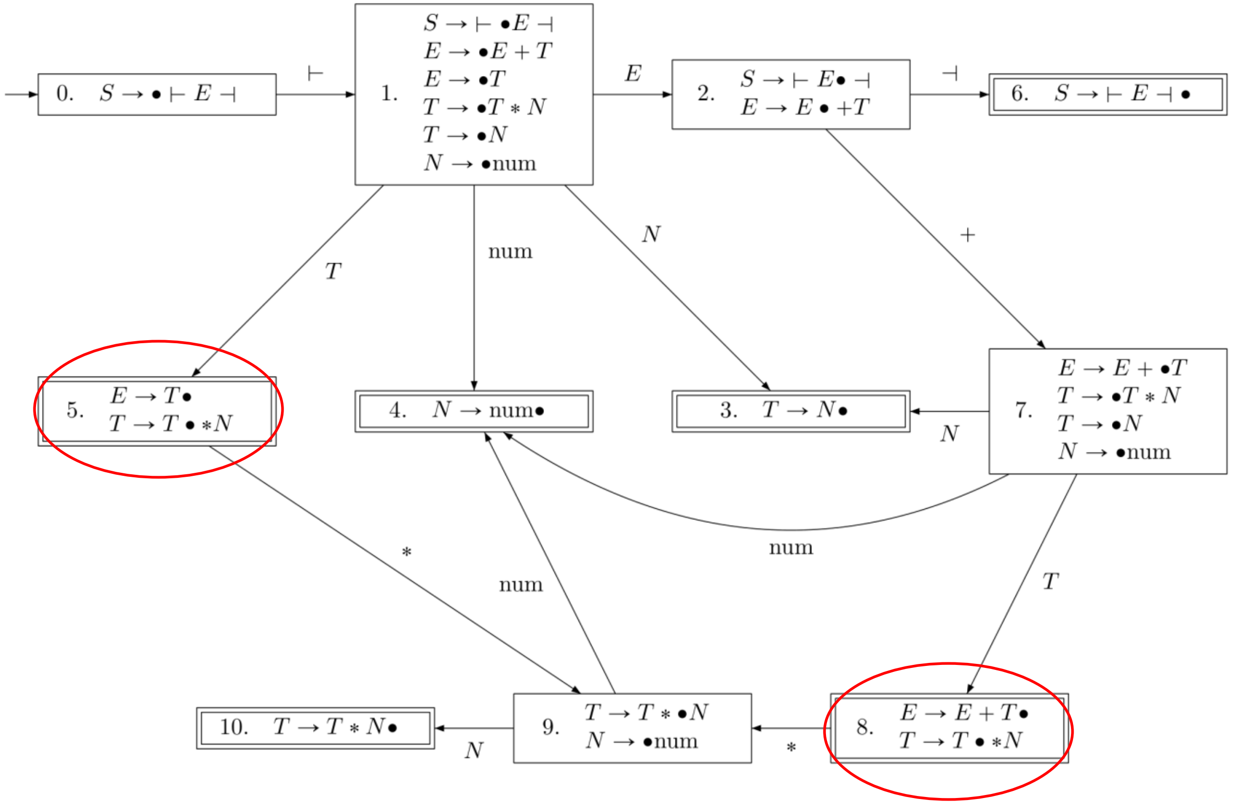


Shift-Reduce Conflicts

Parsing Conflicts

Let's consider a partial parse where the reduction sequence so far is: **⊢ T**

- If the remaining input is just:
  ⊣
  Then we need to reduce.
- But if the remaining input is:
  * num ⊣
  Then we need to shift!
- Without **lookahead** we can't tell what to do in this situation.

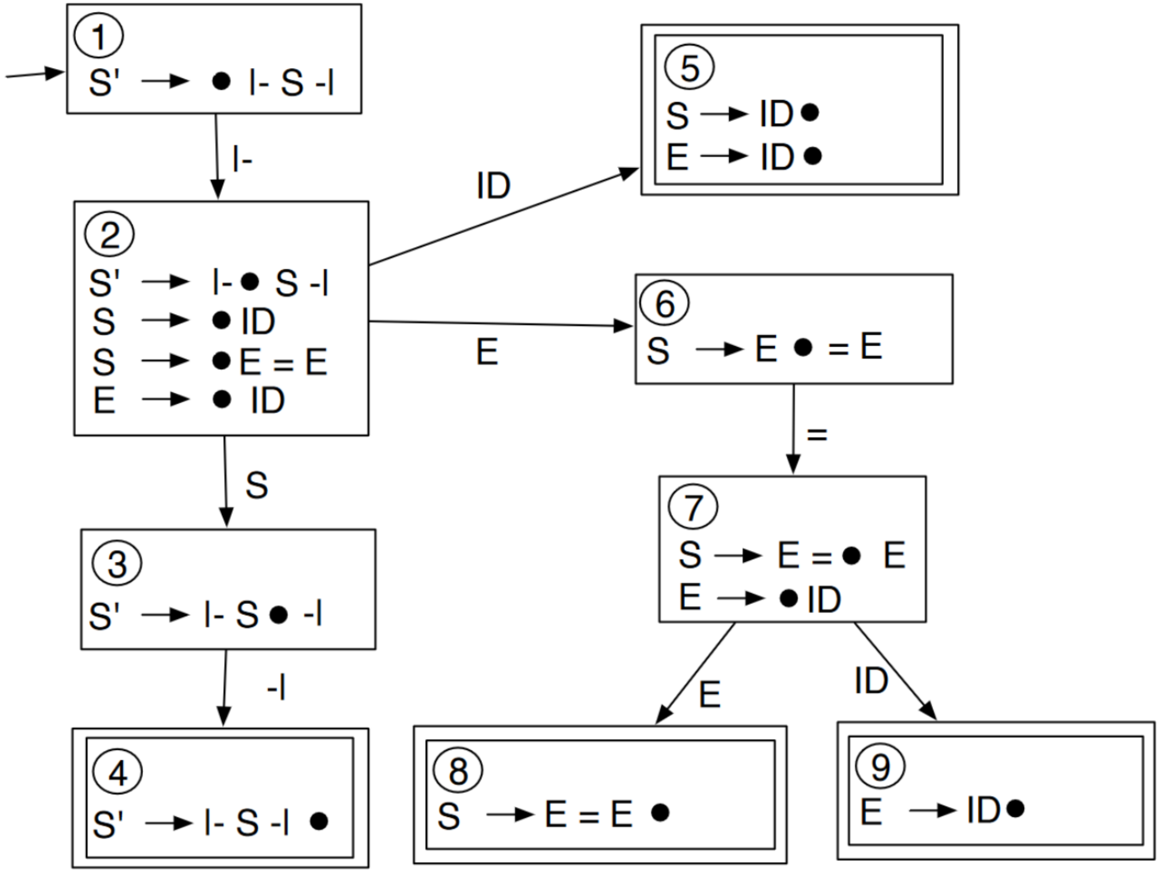


Shift-Reduce Conflicts

Parsing Conflicts

Reduce-Reduce Conflicts are also possible, but not in the previous DFA. Consider the parsing DFA for this grammar:

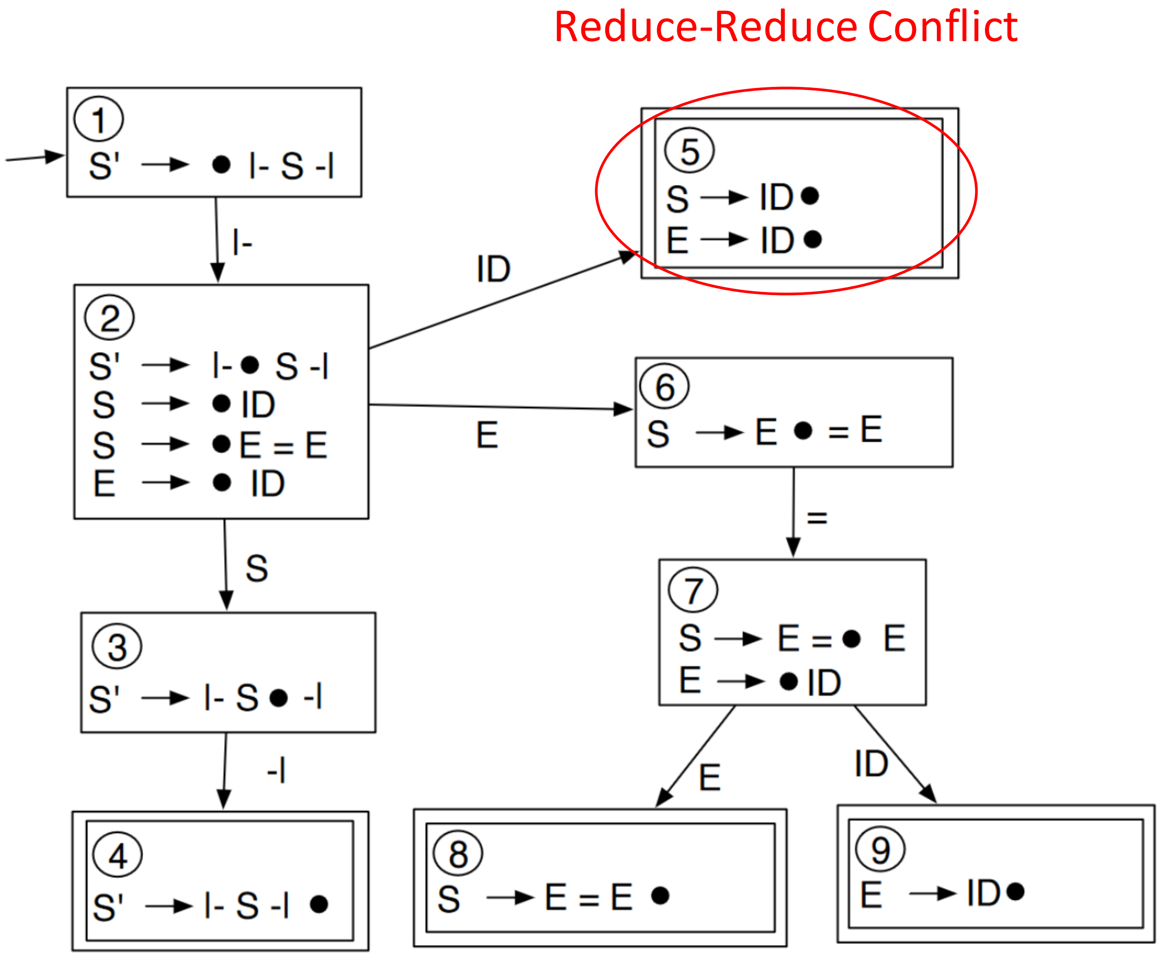
S' → ⊢ S ⊣
S → ID
S → E = E
E → ID



Parsing Conflicts

Reduce-Reduce Conflicts are also possible, but not in the previous DFA. Consider the parsing DFA for this grammar:

S' → ⊢ S ⊣
S → ID
S → E = E
E → ID

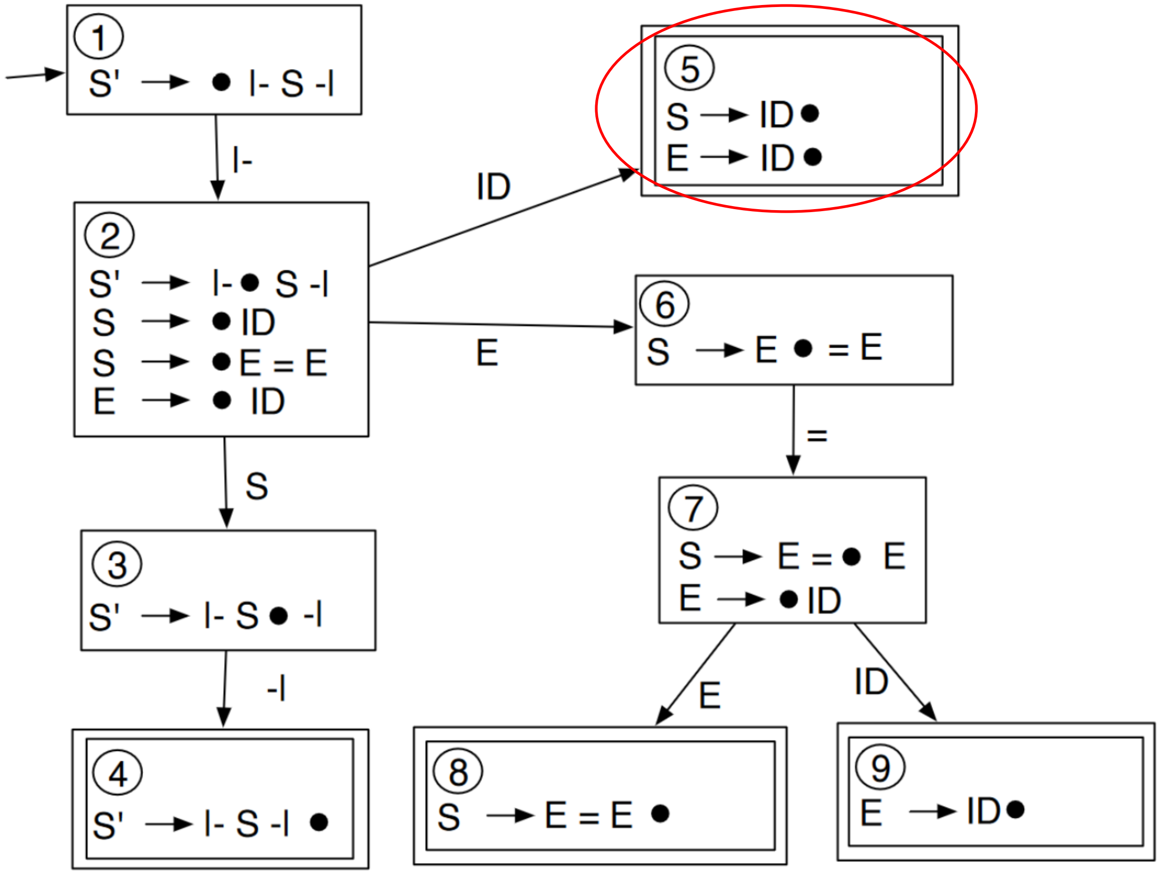


Parsing Conflicts

Again, consider a partial parse where the reduction sequence is:
⊢ ID

- If the unread input is:
  ⊣
  We need to reduce by S → ID.
- If the unread input is
  = ID ⊣
  We need to reduce by E → ID
  so we can eventually match the RHS of S → E = E !!
- Two choices of how to reduce.

Reduce-Reduce Conflict

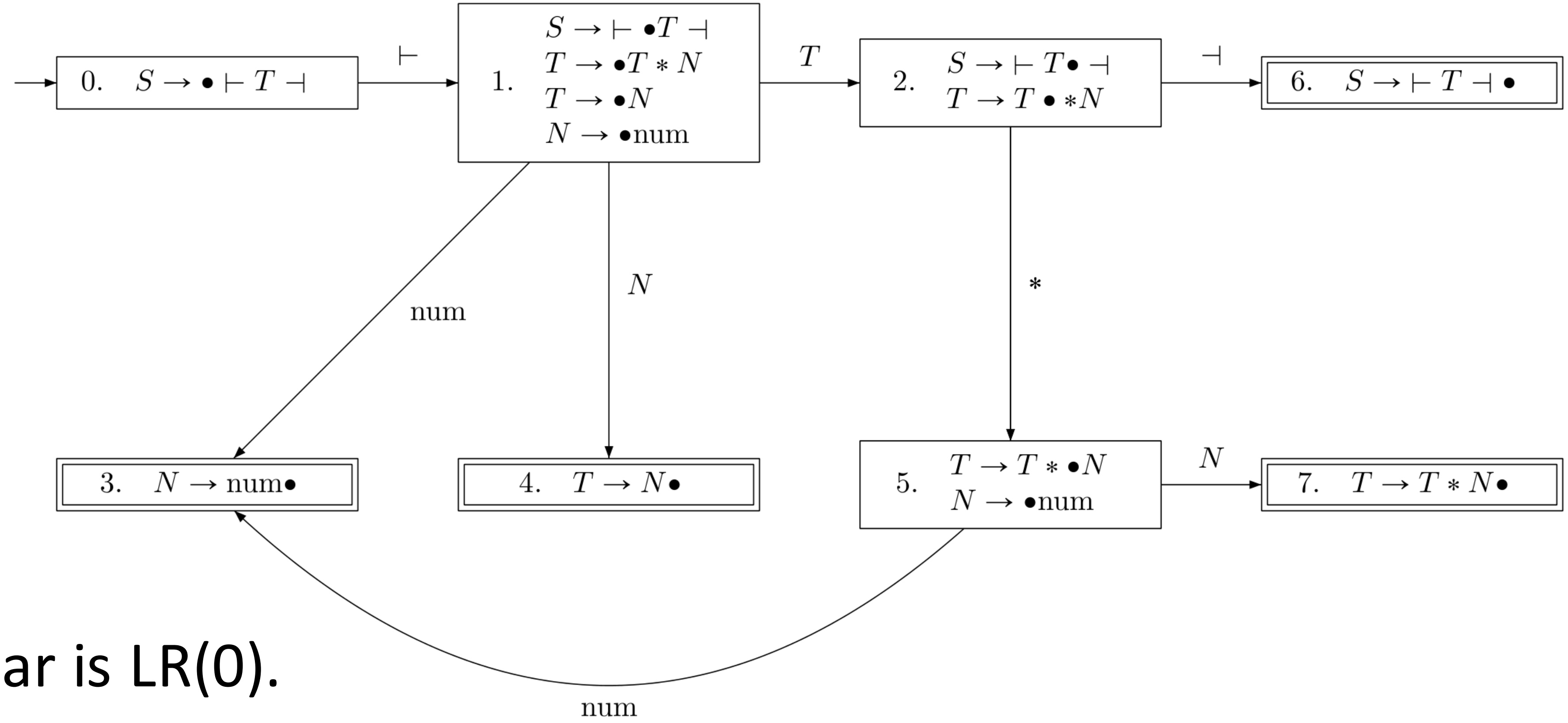


Conflicts and LR(0) Grammars

- In LL(1) parsing, if the Predict table gave us multiple choices for what rule to apply, we simply said "this grammar is not LL(1)" and decided we will not use LL(1) to parse it.
- Similarly, we define an **LR(0) grammar** to be one where the LR(0) DFA has no conflicts, and therefore, we can use the LR(0) algorithm to parse it without having to decide between multiple options.
- Technically we could parse non-LR(0) grammars using some kind of backtracking method, but this would not be efficient (not linear time).

Conflicts and LR(0) Grammars

S → ⊢ T ⊣
T → T * N
T → N
N → num



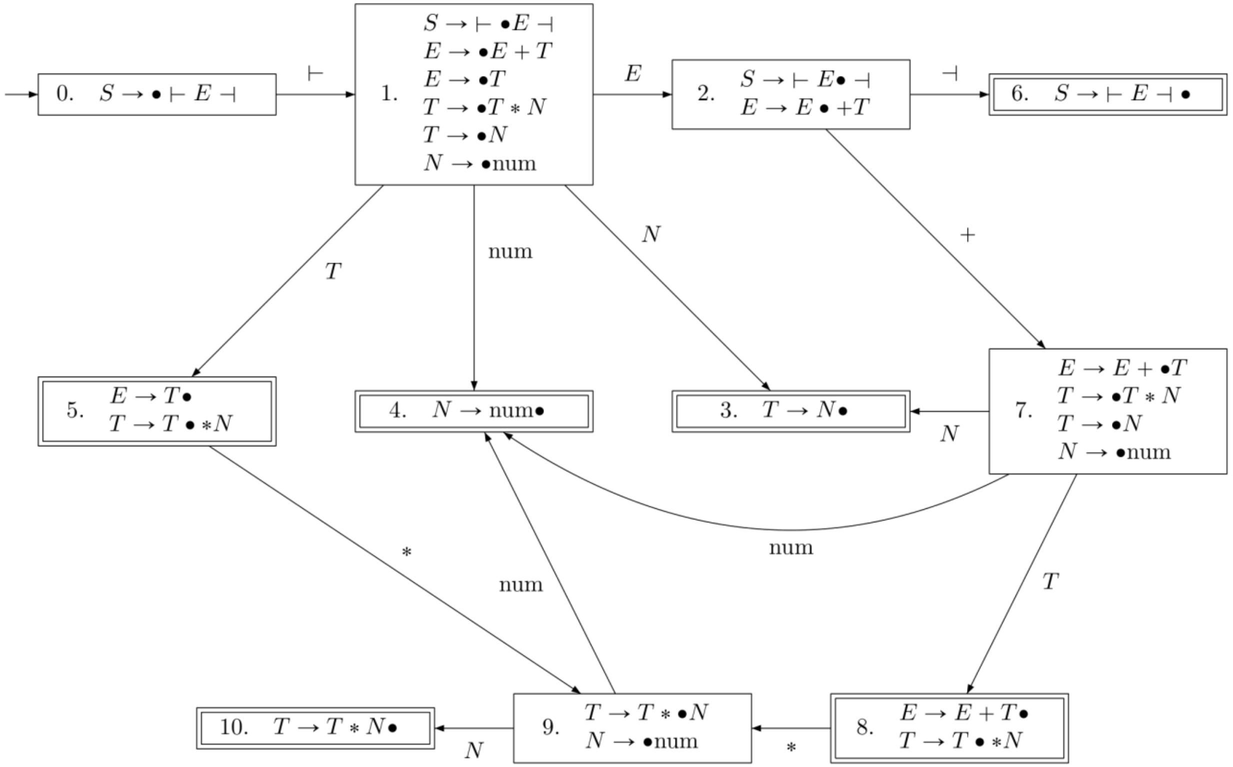
0. $S \to \bullet \vdash T \dashv$
$\vdash$
1. $S \to \vdash \bullet T \dashv$
$T \to \bullet T * N$
$T \to \bullet N$
$N \to \bullet \mathrm{num}$
T
2. $S \to \vdash T \bullet \dashv$
$T \to T \bullet * N$
$\dashv$
6. $S \to \vdash T \dashv \bullet$
num
N
$*$
3. $N \to \mathrm{num} \bullet$
4. $T \to N \bullet$
5. $T \to T * \bullet N$
$N \to \bullet \mathrm{num}$
N
7. $T \to T * N \bullet$
num

This grammar is LR(0).

Conflicts and LR(0) Grammars

S → ⊢ E ⊣
E → E + T
E → T
T → T * N
T → N
N → num

This grammar is not LR(0). Why is this a problem?



0. $S \to \bullet \vdash E \dashv$
$\vdash$
1. $S \to \vdash \bullet E \dashv$, $E \to \bullet E + T$, $E \to \bullet T$, $T \to \bullet T * N$, $T \to \bullet N$, $N \to \bullet\text{num}$
E
2. $S \to \vdash E \bullet \dashv$, $E \to E \bullet + T$
$\dashv$
6. $S \to \vdash E \dashv \bullet$
T
num
N
$+$
5. $E \to T \bullet$, $T \to T \bullet * N$
4. $N \to \text{num}\bullet$
3. $T \to N \bullet$
7. $E \to E + \bullet T$, $T \to \bullet T * N$, $T \to \bullet N$, $N \to \bullet\text{num}$
N
num
$*$
num
T
10. $T \to T * N \bullet$
9. $T \to T * \bullet N$, $N \to \bullet\text{num}$
8. $E \to E + T \bullet$, $T \to T \bullet * N$
N
$*$

Beyond LR(0): Adding Lookahead

- The non-LR(0) grammar on the previous slide describes expressions with addition and multiplication in the proper precedence order.
- So, LR(0) cannot parse our standard unambiguous grammar for arithmetic expressions.
- This is bad because the WLP4 programming language that we want to parse supports math.
- Should we now look at **LR(1) parsing**?
- We'll use the LR(1) parsing *algorithm*, but LR(1) *DFAs* are too complex.
- We'll use **SLR(1)** (Simple LR(1)) DFAs instead.