Bottom-up Parsing

- 从左到右对 Input Terminal 扫描.
- 根据输入的 Input 选择相应的产生式模拟最右推导的逆(也可看成自下而上建立语法树，由叶索根).
- 无回溯（Non-backtracking）.

归约 (reduction)

- 是推导关系的逆.
- 最左归约是最右推导关系的逆.
- 将寻找文法符号串中的一个子串，该子串和某一个产生式 RHS 的文法符号串一致，用该产生式 LHS 的非终结符替换该串后得到的文法符号串满足归约关系：

$$\alpha\underline{\gamma}\beta \Leftarrow \alpha A\beta \text{ if } A \to \gamma \in P$$

Example

XL grammar

```
T = { ID,  PLUS, TIMES,
      LP,  RP};
N = {exp, term, fac};
S = exp;
P:
1/ exp -> exp PLUS term
2       |  term

3/ term -> term TIMES fac
4/       |  fac

5/ fac -> ID
6/      |  LP exp  RP
```

最左归约

$$
\begin{array}{lll}
 & \underline{\text{ID}}\ \text{PLUS ID TIMES ID} & \text{ID PLUS ID TIMES ID} \\
1 \underset{rm}{\Leftarrow} & \underline{\text{fac}}\ \text{PLUS ID TIMES ID} & \text{ID PLUS ID TIMES ID} \\
2 \underset{rm}{\Leftarrow} & \underline{\text{term}}\ \text{PLUS ID TIMES ID} & \text{ID PLUS ID TIMES ID} \\
3 \underset{rm}{\Leftarrow} & \text{exp PLUS}\ \underline{\text{ID}}\ \text{TIMES ID} & \text{ID PLUS ID TIMES ID} \\
4 \underset{rm}{\Leftarrow} & \text{exp PLUS}\ \underline{\text{fac}}\ \text{TIMES ID} & \text{ID PLUS ID ITMES ID} \\
5 \underset{rm}{\Leftarrow} & \text{exp PLUS term TIMES}\ \underline{\text{ID}} & \text{ID PLUS ID TIMES ID} \\
6 \underset{rm}{\Leftarrow} & \text{exp PLUS}\ \underline{\text{term TIMES fac}} & \text{ID PLUS ID TIMES ID} \\
7 \underset{rm}{\Leftarrow} & \underline{\text{exp PLUS term}} & \text{ID PLUS ID TIMES ID} \\
8 \underset{rm}{\Leftarrow} & \underline{\text{exp}} & \text{ID PLUS ID TIMES ID}
\end{array}
$$

问题

- 在第⑤如果用以下方式归约将出错：

$$
\begin{array}{lll}
5 & \underset{rm}{\Leftarrow} & \underline{\text{exp PLUS term}}\ \text{TIMES ID} \\
6' & \underset{rm}{\Leftarrow} & \underbrace{\text{exp TIMES ID}}
\end{array}
$$

Not match any RHS of productions!

Example

XL grammar

```
T = { ID,  PLUS, TIMES,
      LP,  RP};
N = {exp, term, fac};
S = exp;
P:
1/ exp -> exp PLUS term
2       |  term

3/ term -> term TIMES fac
4/       |  fac

5/ fac -> ID
6/      |  LP exp  RP
```

最左归约

		ID PLUS ID TIMES ID	ID PLUS ID TIMES ID
1	$\Leftarrow_{rm}$	fac PLUS ID TIMES ID	ID PLUS ID TIMES ID
2	$\Leftarrow_{rm}$	term PLUS ID TIMES ID	ID PLUS ID TIMES ID
3	$\Leftarrow_{rm}$	exp PLUS ID TIMES ID	ID PLUS ID TIMES ID
4	$\Leftarrow_{rm}$	exp PLUS fac TIMES ID	ID PLUS ID ITMES ID
5	$\Leftarrow_{rm}$	exp PLUS term TIMES ID	ID PLUS ID TIMES ID
6	$\Leftarrow_{rm}$	exp PLUS term TIMES fac	ID PLUS ID TIMES ID
7	$\Leftarrow_{rm}$	exp PLUS term	ID PLUS ID TIMES ID
8	$\Leftarrow_{rm}$	exp	ID PLUS ID TIMES ID

问题

- 在第⑤如果用以下方式归约将出错：

$$5 \quad \underset{rm}{\Leftarrow} \quad \text{exp PLUS term TIMES ID}$$

$$6' \quad \underset{rm}{\Leftarrow} \quad \underbrace{\text{exp TIMES ID}}_{\text{Not match any RHS of productions!}}$$

Example

XL grammar

```
T = { ID,  PLUS, TIMES,
      LP,  RP};
N = {exp, term, fac};
S = exp;
P:
1/ exp -> exp PLUS term
2       |  term

3/ term -> term TIMES fac
4/       |  fac

5/ fac -> ID
6/      |  LP exp  RP
```

最左归约

		ID PLUS ID TIMES ID	ID PLUS ID TIMES ID
1	$\Leftarrow_{rm}$	fac PLUS ID TIMES ID	ID PLUS ID TIMES ID
2	$\Leftarrow_{rm}$	term PLUS ID TIMES ID	ID PLUS ID TIMES ID
3	$\Leftarrow_{rm}$	exp PLUS ID TIMES ID	ID PLUS ID TIMES ID
4	$\Leftarrow_{rm}$	exp PLUS fac TIMES ID	ID PLUS ID ITMES ID
5	$\Leftarrow_{rm}$	exp PLUS term TIMES ID	ID PLUS ID TIMES ID
6	$\Leftarrow_{rm}$	exp PLUS term TIMES fac	ID PLUS ID TIMES ID
7	$\Leftarrow_{rm}$	exp PLUS term	ID PLUS ID TIMES ID
8	$\Leftarrow_{rm}$	exp	ID PLUS ID TIMES ID

问题

- 在第⑤如果用以下方式归约将出错：

$$5 \quad \Leftarrow_{rm} \quad \underline{\text{exp PLUS term}}\ \text{TIMES ID}$$

$$6' \quad \Leftarrow_{rm} \quad \underbrace{\text{exp TIMES ID}}$$

Not match any RHS of productions!

Example

XL grammar

```
T = { ID,  PLUS, TIMES,
      LP,  RP};
N = {exp, term, fac};
S = exp;
P:
1/ exp -> exp PLUS term
2       |  term

3/ term -> term TIMES fac
4/       |  fac

5/ fac -> ID
6/      |  LP exp  RP
```

最左归约

		ID PLUS ID TIMES ID	ID PLUS ID TIMES ID
1	$\Leftarrow_{rm}$	fac PLUS ID TIMES ID	ID PLUS ID TIMES ID
2	$\Leftarrow_{rm}$	term PLUS ID TIMES ID	ID PLUS ID TIMES ID
3	$\Leftarrow_{rm}$	exp PLUS ID TIMES ID	ID PLUS ID TIMES ID
4	$\Leftarrow_{rm}$	exp PLUS fac TIMES ID	ID PLUS ID ITMES ID
5	$\Leftarrow_{rm}$	exp PLUS term TIMES ID	ID PLUS ID TIMES ID
6	$\Leftarrow_{rm}$	exp PLUS term TIMES fac	ID PLUS ID TIMES ID
7	$\Leftarrow_{rm}$	exp PLUS term	ID PLUS ID TIMES ID
8	$\Leftarrow_{rm}$	exp	ID PLUS ID TIMES ID

问题

- 在第⑤如果用以下方式归约将出错：

$$5 \quad \Leftarrow_{rm} \quad \underline{\text{exp PLUS term}}\ \text{TIMES ID}$$

$$6' \quad \Leftarrow_{rm} \quad \underbrace{\text{exp TIMES ID}}$$

Not match any RHS of productions!

Example

XL grammar

```
T = { ID,  PLUS, TIMES,
      LP,  RP};
N = {exp, term, fac};
S = exp;
P:
1/ exp -> exp PLUS term
2       |  term

3/ term -> term TIMES fac
4/       |  fac

5/ fac -> ID
6/      |  LP exp  RP
```

最左归约

		ID PLUS ID TIMES ID	ID PLUS ID TIMES ID
1	$\overset{rm}{\Leftarrow}$	fac PLUS ID TIMES ID	ID PLUS ID TIMES ID
2	$\overset{rm}{\Leftarrow}$	term PLUS ID TIMES ID	ID PLUS ID TIMES ID
3	$\overset{rm}{\Leftarrow}$	exp PLUS ID TIMES ID	ID PLUS ID TIMES ID
4	$\overset{rm}{\Leftarrow}$	exp PLUS fac TIMES ID	ID PLUS ID ITMES ID
5	$\overset{rm}{\Leftarrow}$	exp PLUS term TIMES ID	ID PLUS ID TIMES ID
6	$\overset{rm}{\Leftarrow}$	exp PLUS term TIMES fac	ID PLUS ID TIMES ID
7	$\overset{rm}{\Leftarrow}$	exp PLUS term	ID PLUS ID TIMES ID
8	$\overset{rm}{\Leftarrow}$	exp	ID PLUS ID TIMES ID

问题

- 在第⑤如果用以下方式归约将出错：

$$5 \quad \overset{rm}{\Leftarrow} \quad \underline{\text{exp PLUS term}}\ \text{TIMES ID}$$

$$6' \quad \overset{rm}{\Leftarrow} \quad \underbrace{\text{exp TIMES ID}}$$

Not match any RHS of productions!

Example

XL grammar

```
T = { ID,  PLUS, TIMES,
      LP,  RP};
N = {exp, term, fac};
S = exp;
P:
1/ exp -> exp PLUS term
2       |  term

3/ term -> term TIMES fac
4/       |  fac

5/ fac -> ID
6/      |  LP exp  RP
```

最左归约

		ID PLUS ID TIMES ID	ID PLUS ID TIMES ID
1	$\Leftarrow_{rm}$	fac PLUS ID TIMES ID	ID PLUS ID TIMES ID
2	$\Leftarrow_{rm}$	term PLUS ID TIMES ID	ID PLUS ID TIMES ID
3	$\Leftarrow_{rm}$	exp PLUS ID TIMES ID	ID PLUS ID TIMES ID
4	$\Leftarrow_{rm}$	exp PLUS fac TIMES ID	ID PLUS ID ITMES ID
5	$\Leftarrow_{rm}$	exp PLUS term TIMES ID	ID PLUS ID TIMES ID
6	$\Leftarrow_{rm}$	exp PLUS term TIMES fac	ID PLUS ID TIMES ID
7	$\Leftarrow_{rm}$	exp PLUS term	ID PLUS ID TIMES ID
8	$\Leftarrow_{rm}$	exp	ID PLUS ID TIMES ID

问题

- 在第⑤如果用以下方式归约将出错：

$$5 \quad \underset{rm}{\Leftarrow} \quad \underbrace{\text{exp PLUS term}}\ \text{TIMES ID}$$

$$6' \quad \underset{rm}{\Leftarrow} \quad \underbrace{\text{exp TIMES ID}}$$

Not match any RHS of productions!

Example

XL grammar

```
T = { ID,  PLUS, TIMES,
      LP,  RP};
N = {exp, term, fac};
S = exp;
P:
1/ exp -> exp PLUS term
2       |  term

3/ term -> term TIMES fac
4/       |  fac

5/ fac -> ID
6/      |  LP exp  RP
```

最左归约

		ID PLUS ID TIMES ID	ID PLUS ID TIMES ID
1	$\Leftarrow_{rm}$	fac PLUS ID TIMES ID	ID PLUS ID TIMES ID
2	$\Leftarrow_{rm}$	term PLUS ID TIMES ID	ID PLUS ID TIMES ID
3	$\Leftarrow_{rm}$	exp PLUS ID TIMES ID	ID PLUS ID TIMES ID
4	$\Leftarrow_{rm}$	exp PLUS fac TIMES ID	ID PLUS ID ITMES ID
5	$\Leftarrow_{rm}$	exp PLUS term TIMES ID	ID PLUS ID TIMES ID
6	$\Leftarrow_{rm}$	exp PLUS term TIMES fac	ID PLUS ID TIMES ID
7	$\Leftarrow_{rm}$	exp PLUS term	ID PLUS ID TIMES ID
8	$\Leftarrow_{rm}$	exp	ID PLUS ID TIMES ID

问题

- 在第⑤如果用以下方式归约将出错：

$$5 \quad \underset{rm}{\Leftarrow} \quad \underbrace{\text{exp PLUS term}}\ \text{TIMES ID}$$

$$6' \quad \underset{rm}{\Leftarrow} \quad \underbrace{\text{exp TIMES ID}}$$

Not match any RHS of productions!

Example

XL grammar

```
T = { ID,  PLUS, TIMES,
      LP,  RP};
N = {exp, term, fac};
S = exp;
P:
1/ exp -> exp PLUS term
2       |  term

3/ term -> term TIMES fac
4/       |  fac

5/ fac -> ID
6/      |  LP exp  RP
```

最左归约

		ID PLUS ID TIMES ID	ID PLUS ID TIMES ID
1	$\Leftarrow_{rm}$	fac PLUS ID TIMES ID	ID PLUS ID TIMES ID
2	$\Leftarrow_{rm}$	term PLUS ID TIMES ID	ID PLUS ID TIMES ID
3	$\Leftarrow_{rm}$	exp PLUS ID TIMES ID	ID PLUS ID TIMES ID
4	$\Leftarrow_{rm}$	exp PLUS fac TIMES ID	ID PLUS ID ITMES ID
5	$\Leftarrow_{rm}$	exp PLUS term TIMES ID	ID PLUS ID TIMES ID
6	$\Leftarrow_{rm}$	exp PLUS term TIMES fac	ID PLUS ID TIMES ID
7	$\Leftarrow_{rm}$	exp PLUS term	ID PLUS ID TIMES ID
8	$\Leftarrow_{rm}$	exp	ID PLUS ID TIMES ID

问题

- 在第⑤如果用以下方式归约将出错：

$$5 \quad \Leftarrow_{rm} \quad \underline{\text{exp PLUS term}}\ \text{TIMES ID}$$

$$6' \quad \Leftarrow_{rm} \quad \underbrace{\text{exp TIMES ID}}$$

Not match any RHS of productions!

Example

XL grammar

```
T = { ID,  PLUS, TIMES,
      LP,  RP};
N = {exp, term, fac};
S = exp;
P:
1/ exp -> exp PLUS term
2       |  term

3/ term -> term TIMES fac
4/       |  fac

5/ fac -> ID
6/      |  LP exp  RP
```

最左归约

		ID PLUS ID TIMES ID	ID PLUS ID TIMES ID
1	$\Leftarrow_{rm}$	fac PLUS ID TIMES ID	ID PLUS ID TIMES ID
2	$\Leftarrow_{rm}$	term PLUS ID TIMES ID	ID PLUS ID TIMES ID
3	$\Leftarrow_{rm}$	exp PLUS ID TIMES ID	ID PLUS ID TIMES ID
4	$\Leftarrow_{rm}$	exp PLUS fac TIMES ID	ID PLUS ID ITMES ID
5	$\Leftarrow_{rm}$	exp PLUS term TIMES ID	ID PLUS ID TIMES ID
6	$\Leftarrow_{rm}$	exp PLUS term TIMES fac	ID PLUS ID TIMES ID
7	$\Leftarrow_{rm}$	exp PLUS term	ID PLUS ID TIMES ID
8	$\Leftarrow_{rm}$	exp	ID PLUS ID TIMES ID

问题

- 在第⑤如果用以下方式归约将出错：

$$5 \quad \underset{rm}{\Leftarrow} \quad \underline{\text{exp PLUS term}}\ \text{TIMES ID}$$

$$6' \quad \underset{rm}{\Leftarrow} \quad \underbrace{\text{exp TIMES ID}}$$

Not match any RHS of productions!

Example

XL grammar

```
T = { ID,  PLUS, TIMES,
      LP,  RP};
N = {exp, term, fac};
S = exp;
P:
1/ exp -> exp PLUS term
2       |  term

3/ term -> term TIMES fac
4/       |  fac

5/ fac -> ID
6/      |  LP exp  RP
```

最左归约

		ID PLUS ID TIMES ID	ID PLUS ID TIMES ID
1	$\underset{rm}{\Leftarrow}$	fac PLUS ID TIMES ID	ID PLUS ID TIMES ID
2	$\underset{rm}{\Leftarrow}$	term PLUS ID TIMES ID	ID PLUS ID TIMES ID
3	$\underset{rm}{\Leftarrow}$	exp PLUS ID TIMES ID	ID PLUS ID TIMES ID
4	$\underset{rm}{\Leftarrow}$	exp PLUS fac TIMES ID	ID PLUS ID ITMES ID
5	$\underset{rm}{\Leftarrow}$	exp PLUS term TIMES ID	ID PLUS ID TIMES ID
6	$\underset{rm}{\Leftarrow}$	exp PLUS term TIMES fac	ID PLUS ID TIMES ID
7	$\underset{rm}{\Leftarrow}$	exp PLUS term	ID PLUS ID TIMES ID
8	$\underset{rm}{\Leftarrow}$	exp	ID PLUS ID TIMES ID

问题

- 在第⑤如果用以下方式归约将出错：

$$5 \quad \underset{rm}{\Leftarrow} \quad \underbrace{\text{exp PLUS term}}\ \text{TIMES ID}$$

$$6' \quad \underset{rm}{\Leftarrow} \quad \underbrace{\text{exp TIMES ID}}$$

Not match any RHS of productions!

Example

XL grammar

```
T = { ID,  PLUS, TIMES,
      LP,  RP};
N = {exp, term, fac};
S = exp;
P:
1/ exp -> exp PLUS term
2       |  term

3/ term -> term TIMES fac
4/       |  fac

5/ fac -> ID
6/      |  LP exp  RP
```

最左归约

		ID PLUS ID TIMES ID	ID PLUS ID TIMES ID
1	$\Leftarrow_{rm}$	fac PLUS ID TIMES ID	ID PLUS ID TIMES ID
2	$\Leftarrow_{rm}$	term PLUS ID TIMES ID	ID PLUS ID TIMES ID
3	$\Leftarrow_{rm}$	exp PLUS ID TIMES ID	ID PLUS ID TIMES ID
4	$\Leftarrow_{rm}$	exp PLUS fac TIMES ID	ID PLUS ID ITMES ID
5	$\Leftarrow_{rm}$	exp PLUS term TIMES ID	ID PLUS ID TIMES ID
6	$\Leftarrow_{rm}$	exp PLUS term TIMES fac	ID PLUS ID TIMES ID
7	$\Leftarrow_{rm}$	exp PLUS term	ID PLUS ID TIMES ID
8	$\Leftarrow_{rm}$	exp	ID PLUS ID TIMES ID

问题

- 在第⑤如果用以下方式归约将出错：

$$5 \underset{rm}{\Leftarrow} \underline{\text{exp PLUS term}}\ \text{TIMES ID}$$

$$6' \underset{rm}{\Leftarrow} \underbrace{\text{exp TIMES ID}}$$

Not match any RHS of productions!

Example --- 归约与语法树的建立的关系.

最左归约：$\overset{*}{\underset{rm}{\Leftarrow}}$ ID PLUS ID TIMES ID

句柄

Example --- 归约与语法树的建立的关系.

最左归约：$\overset{*}{\underset{rm}{\Leftarrow}}$ fac PLUS ID TIMES ID

句柄

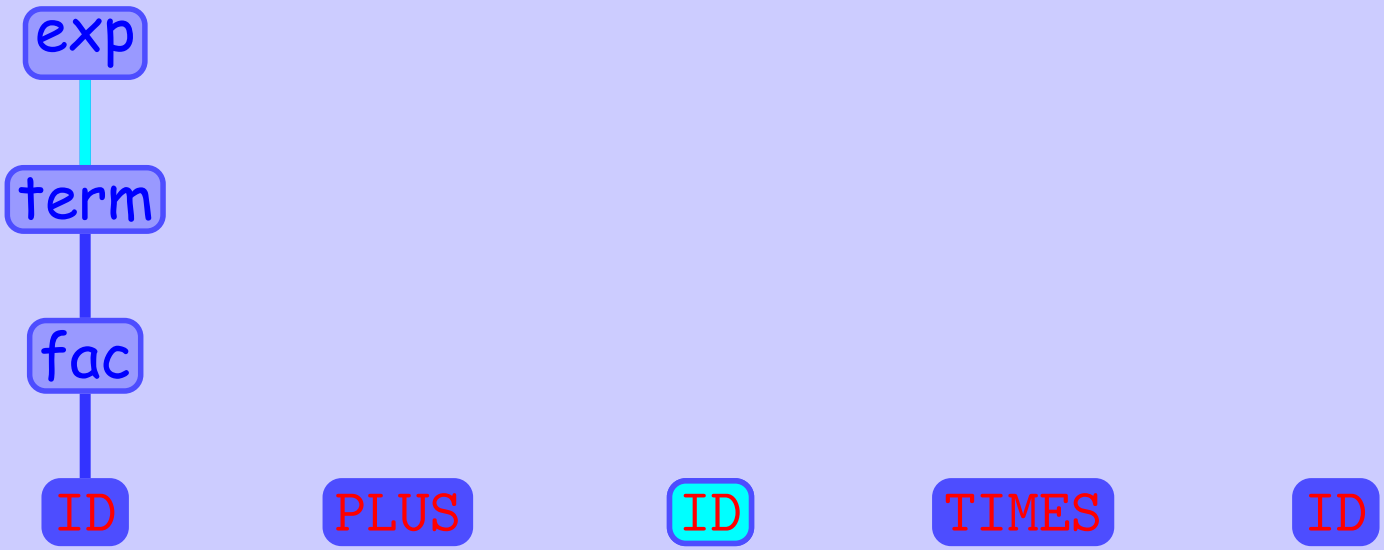
Example --- 归约与语法树的建立的关系.

最左归约：$\overset{*}{\underset{rm}{\Leftarrow}}$ term PLUS ID TIMES ID

句柄

Example --- 归约与语法树的建立的关系.

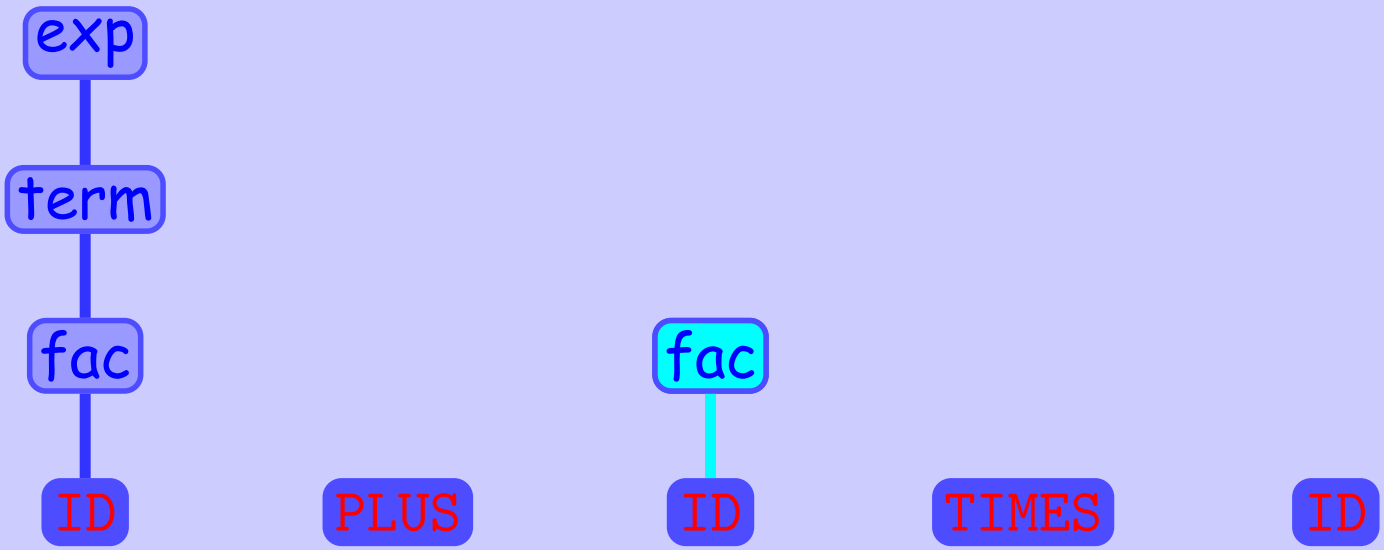
最左归约：$\overset{*}{\underset{rm}{\Leftarrow}}$ exp PLUS ID TIMES ID



句柄

Example --- 归约与语法树的建立的关系.

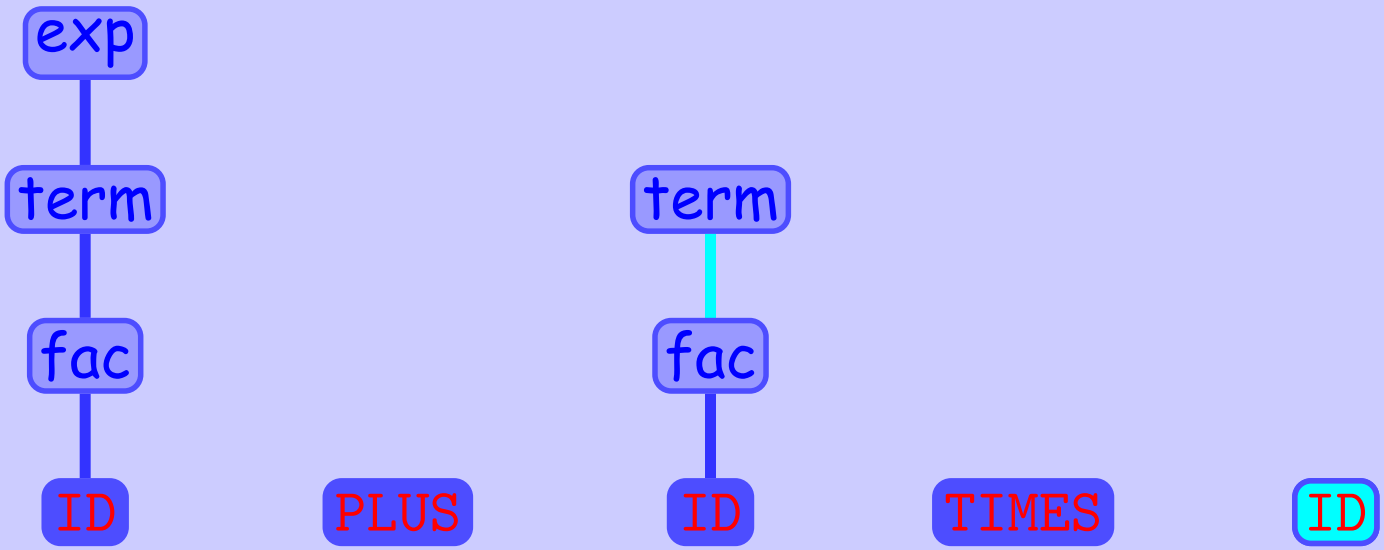
最左归约：$\overset{*}{\underset{rm}{\Leftarrow}}$ exp PLUS fac TIMES ID



句柄

Example --- 归约与语法树的建立的关系.

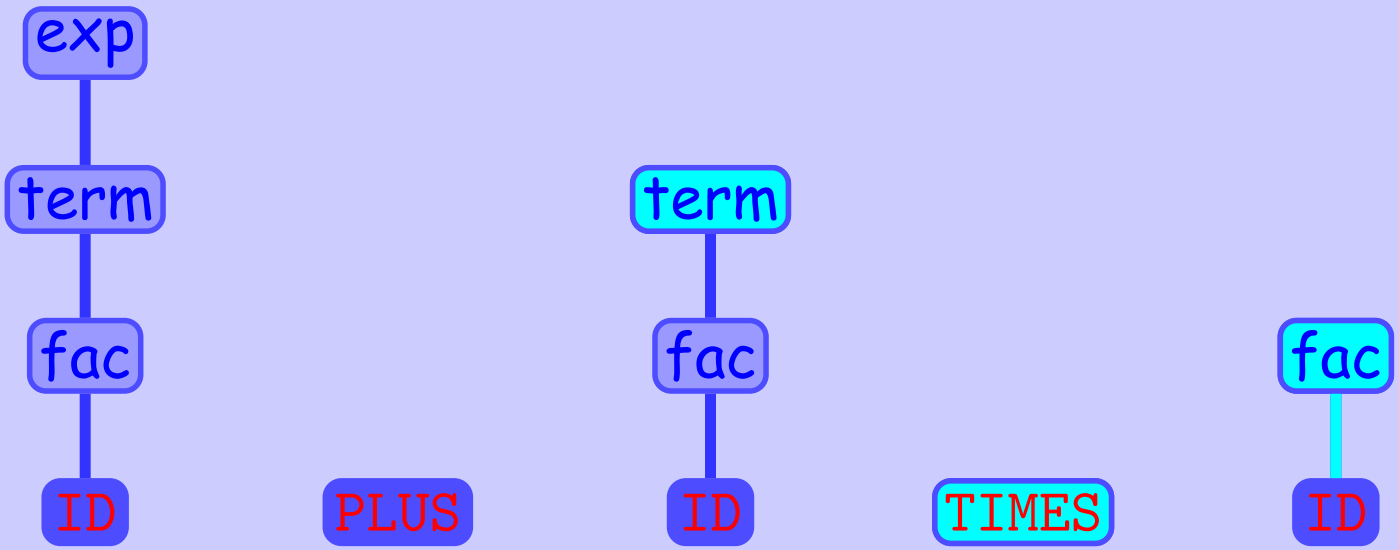
最左归约：$\overset{*}{\underset{rm}{\Leftarrow}}$ exp PLUS term TIMES ID



句柄

Example --- 归约与语法树的建立的关系.

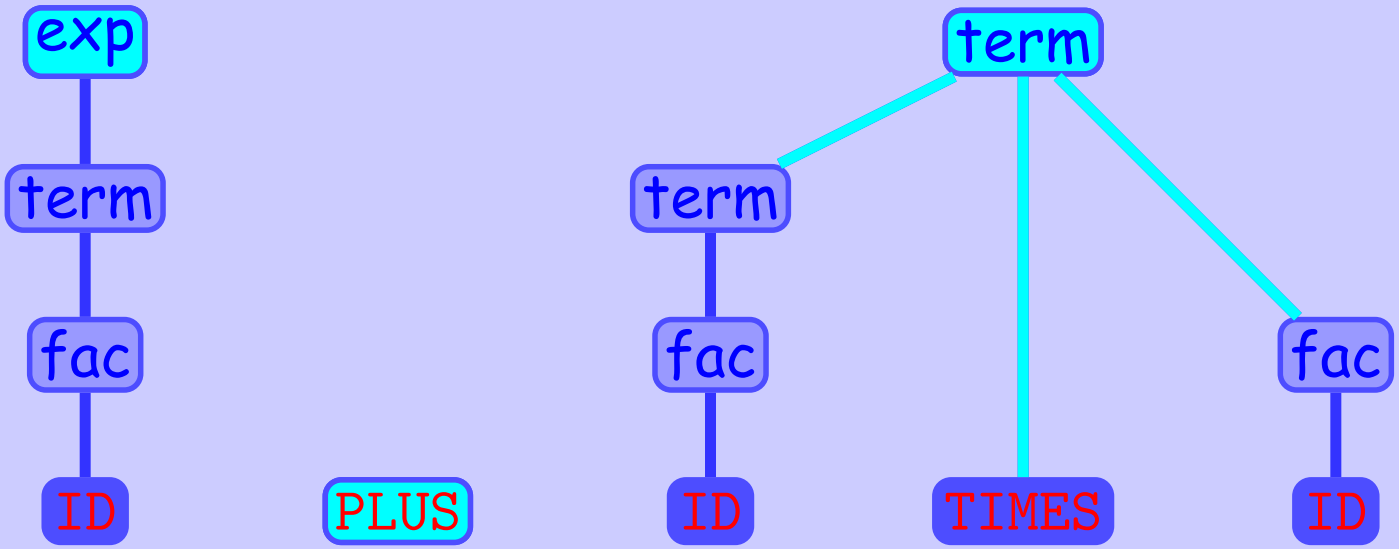
最左归约：$\overset{*}{\underset{rm}{\Leftarrow}}$ exp PLUS term TIMES fac



句柄

Example --- 归约与语法树的建立的关系.

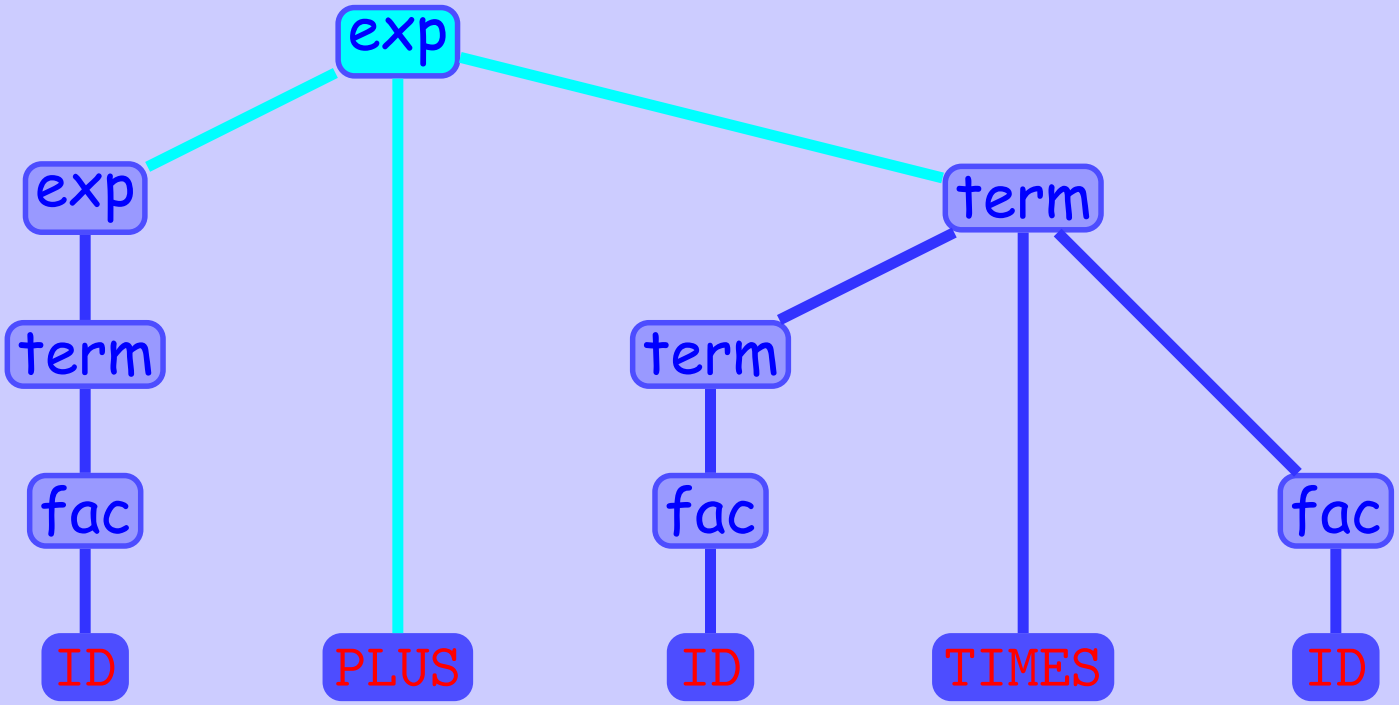
最左归约：$\overset{*}{\underset{rm}{\Leftarrow}}$ exp PLUS term



句柄

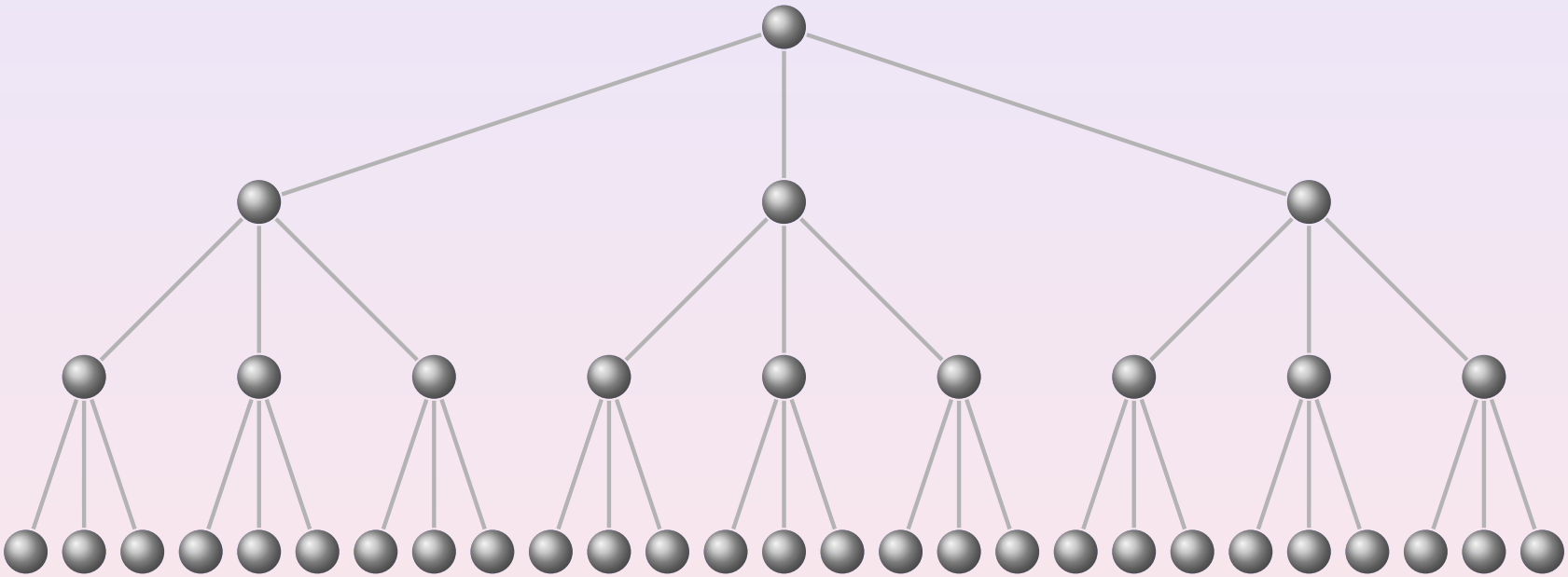
Example --- 归约与语法树的建立的关系.

最左归约：$\overset{*}{\underset{rm}{\Leftarrow}}$ exp

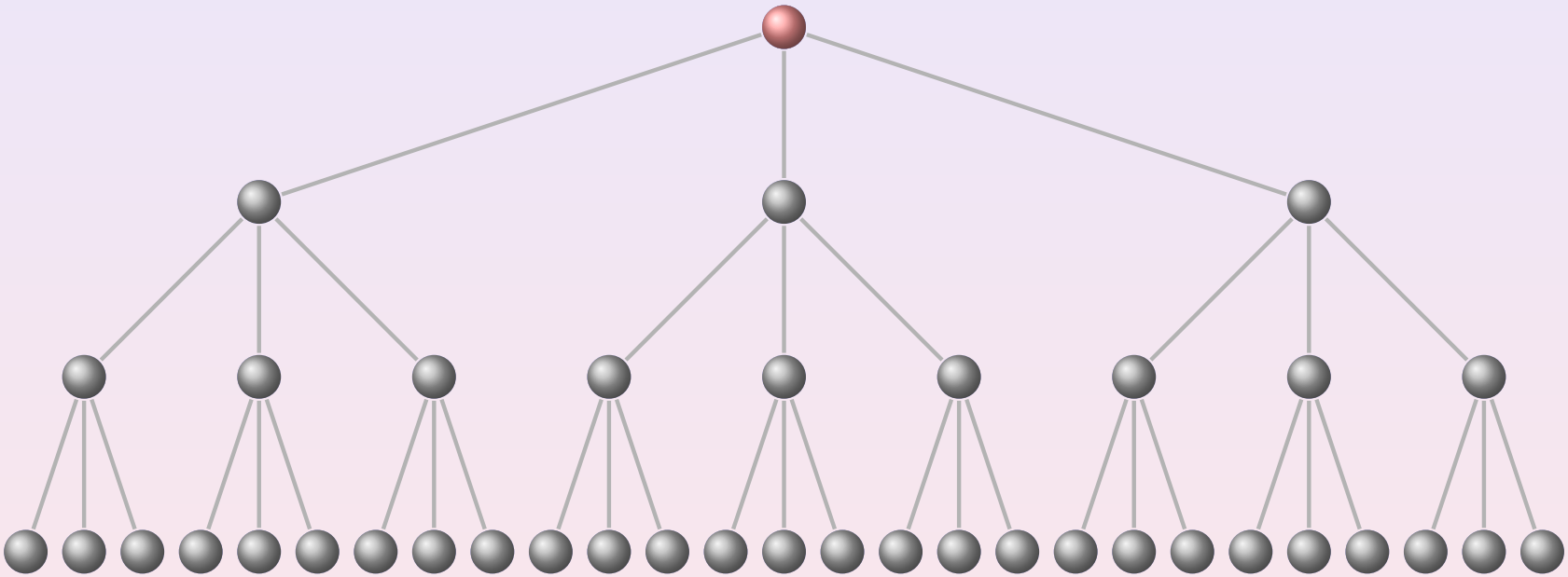


句柄

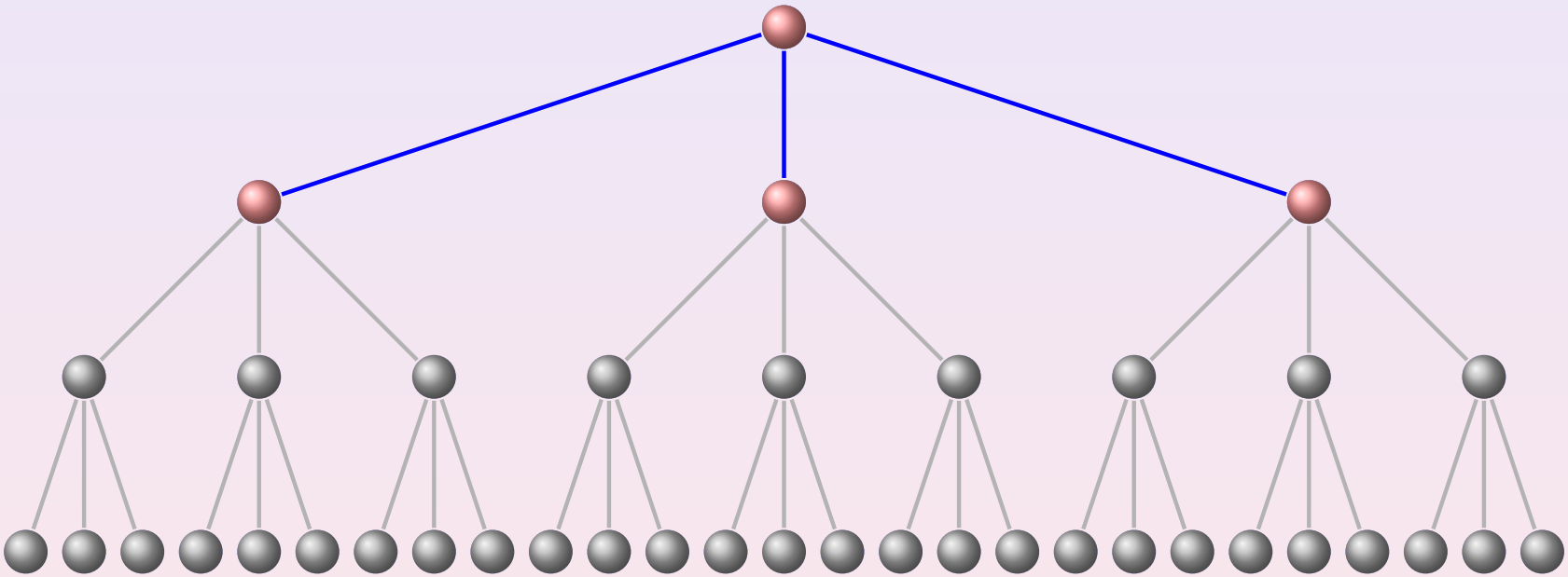
最右推导建树过程



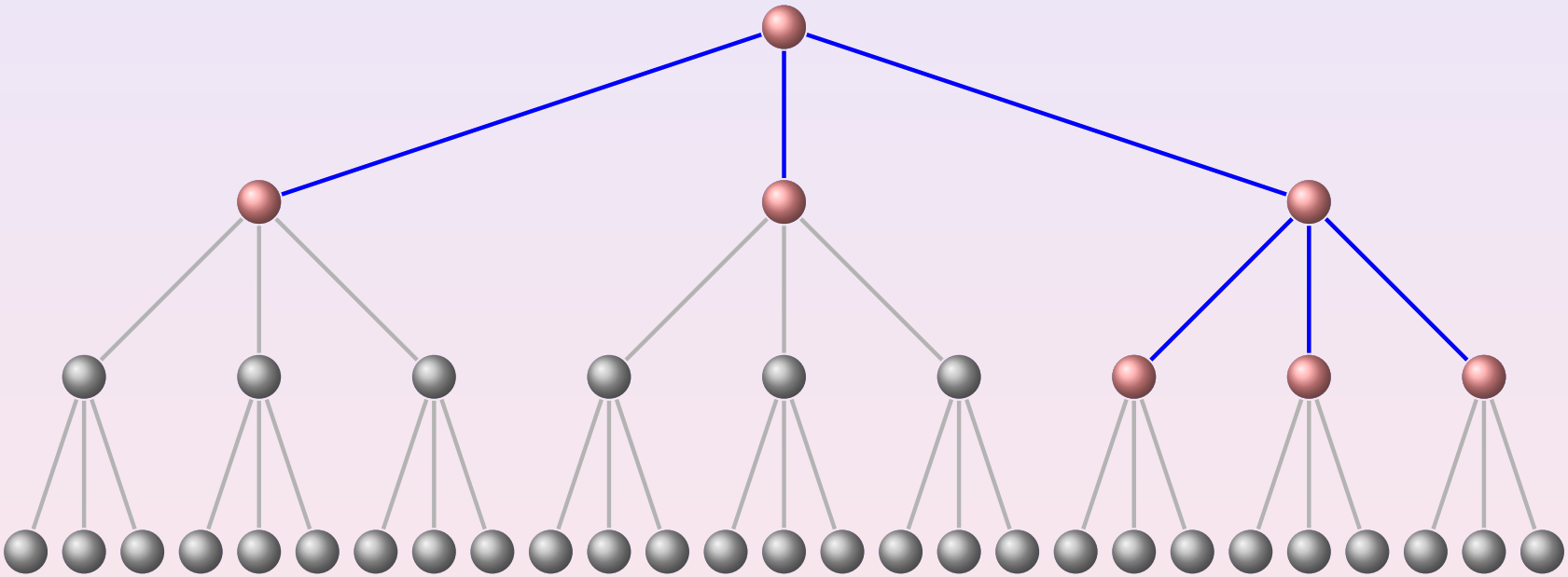
最右推导建树过程



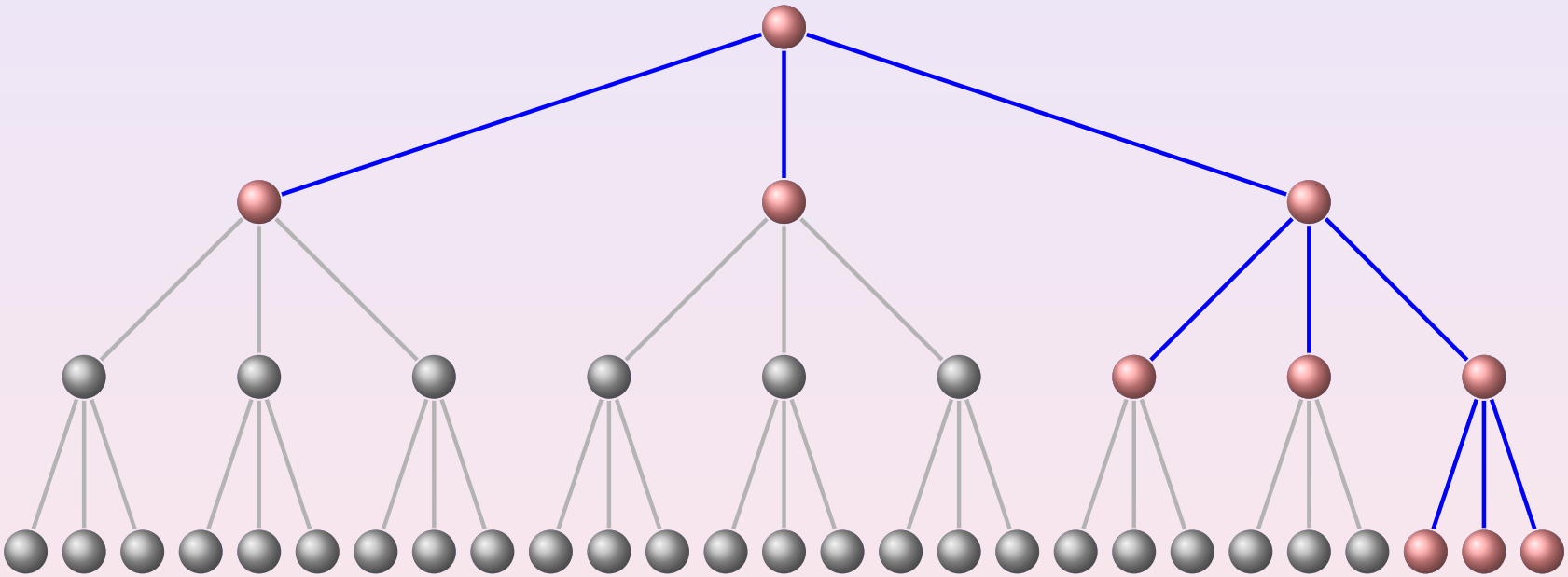
最右推导建树过程



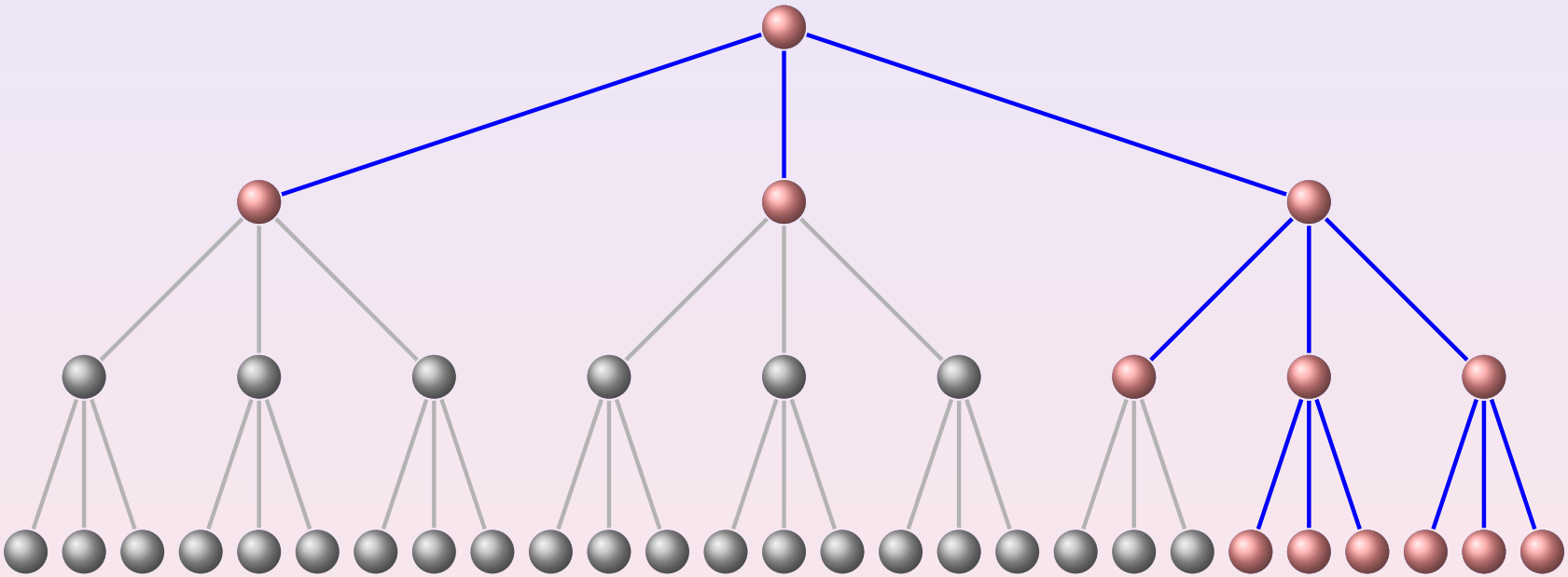
最右推导建树过程



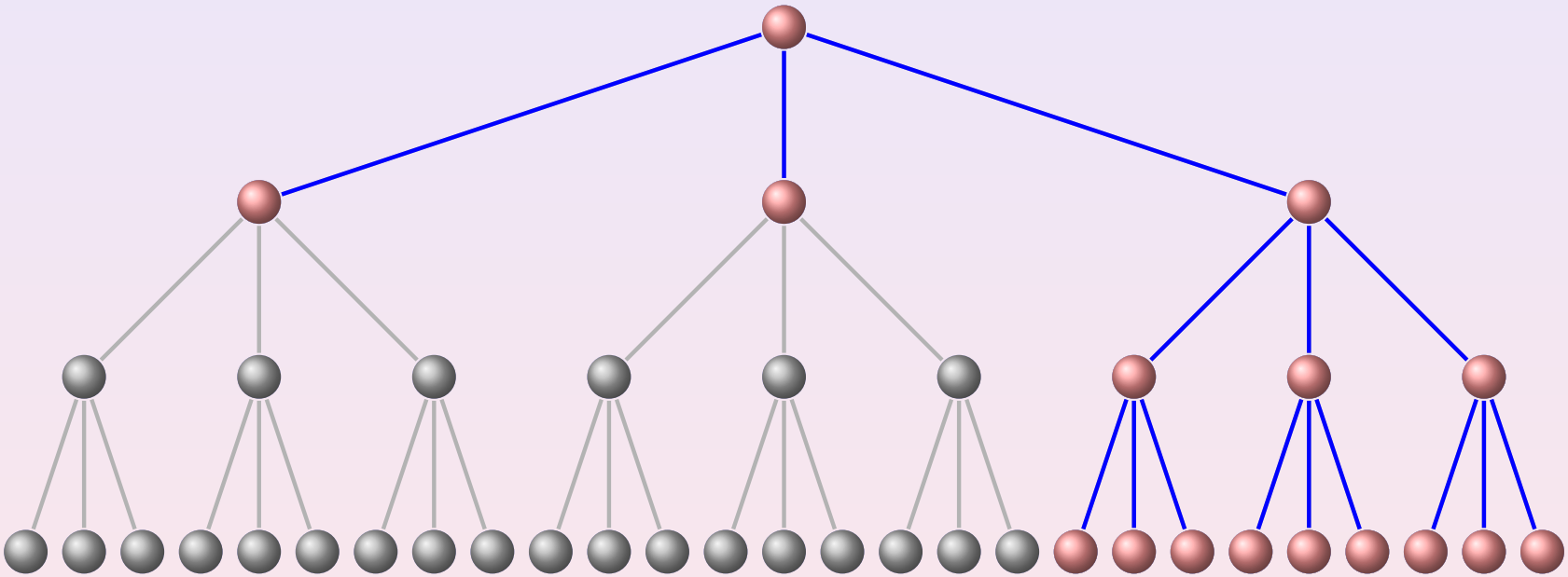
最右推导建树过程



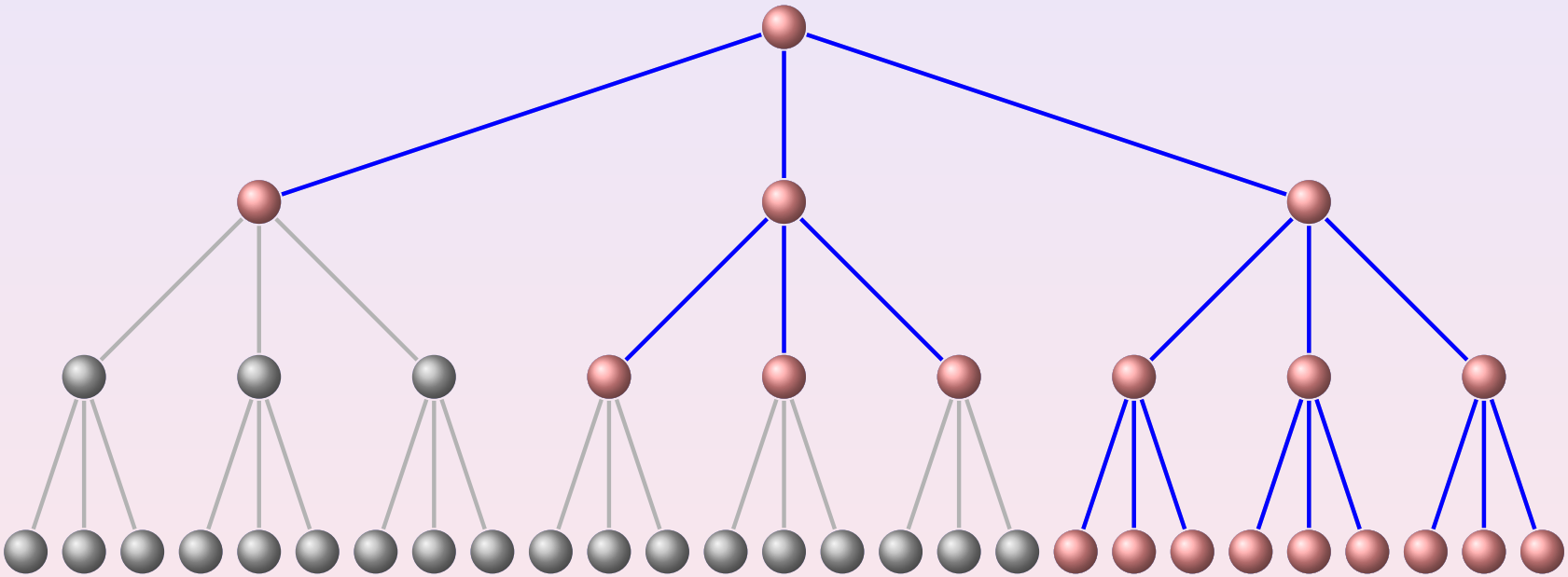
最右推导建树过程



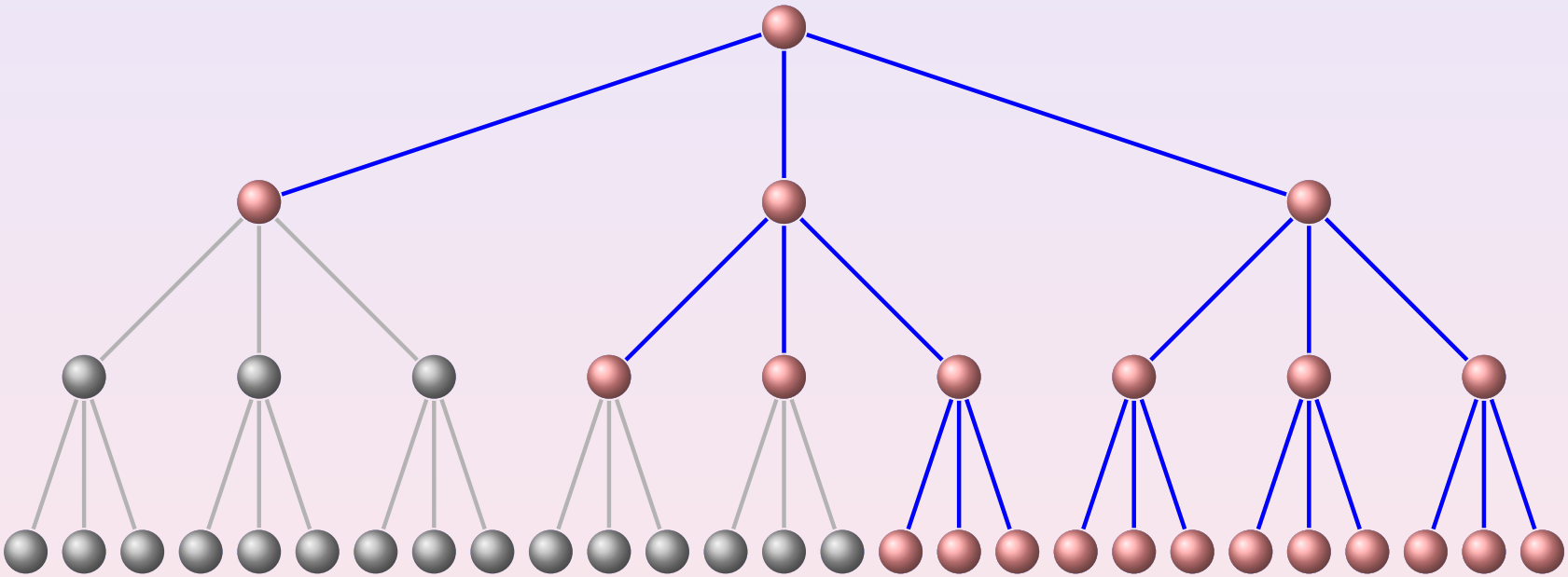
最右推导建树过程



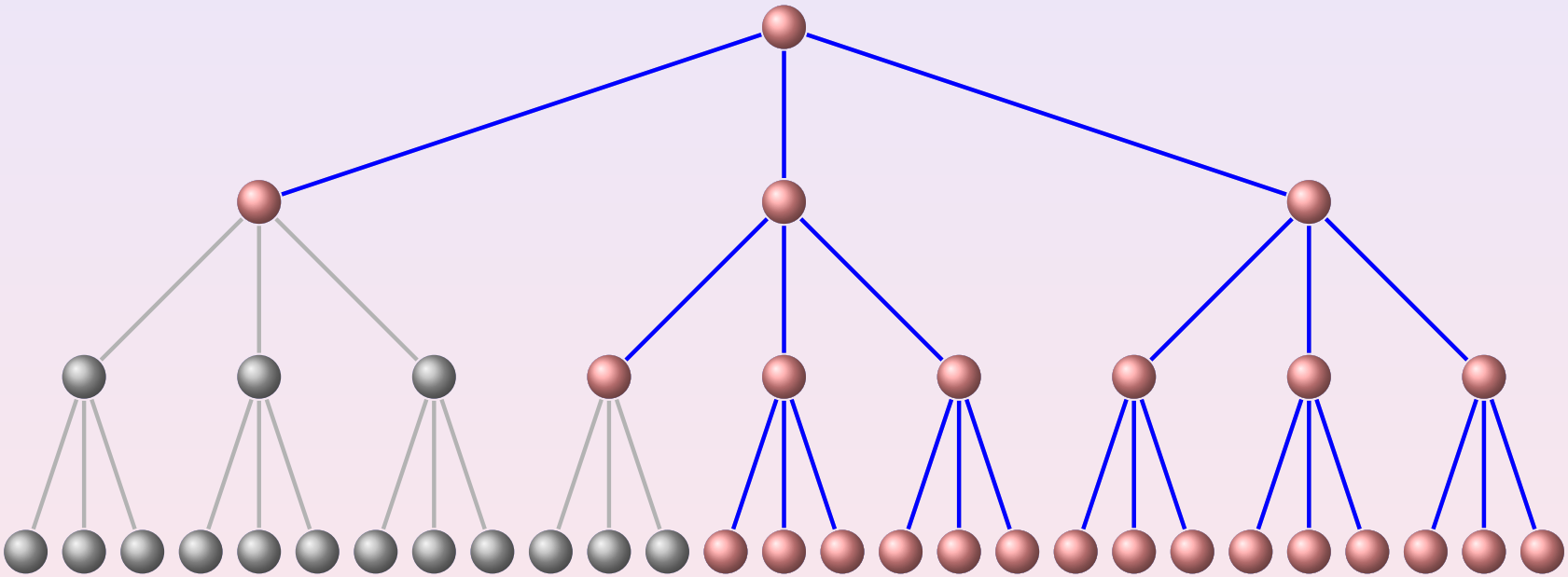
最右推导建树过程



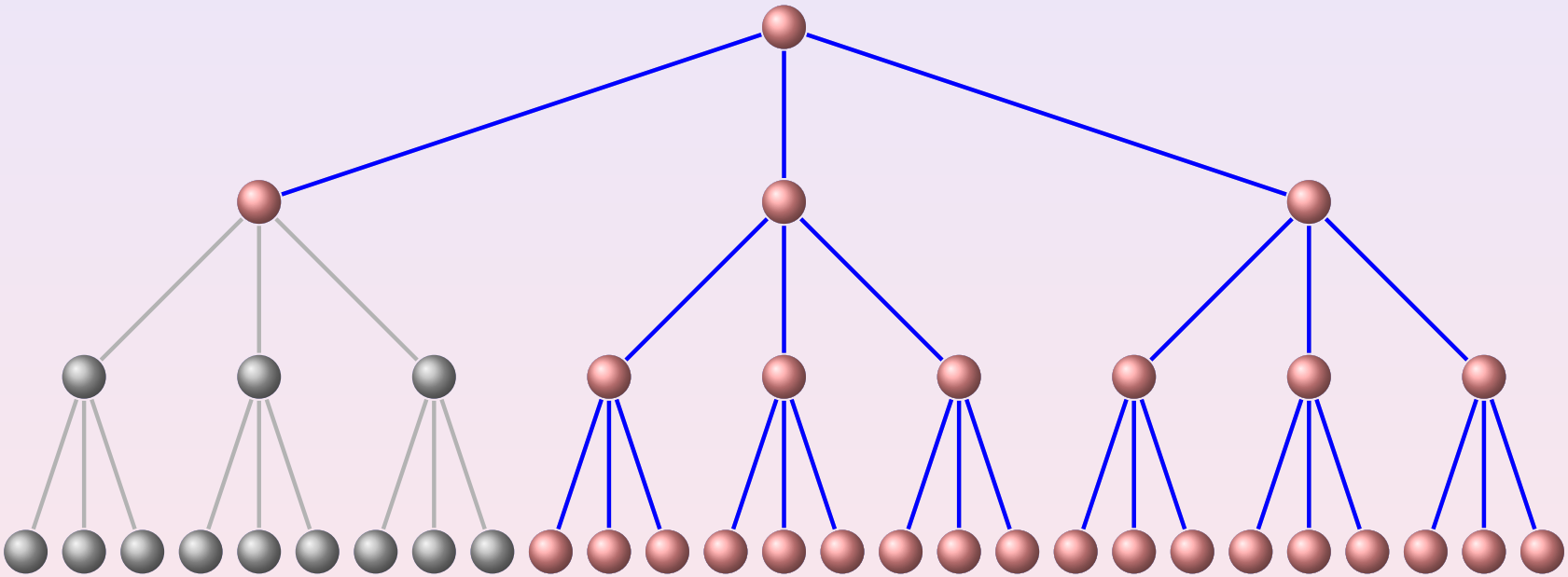
最右推导建树过程



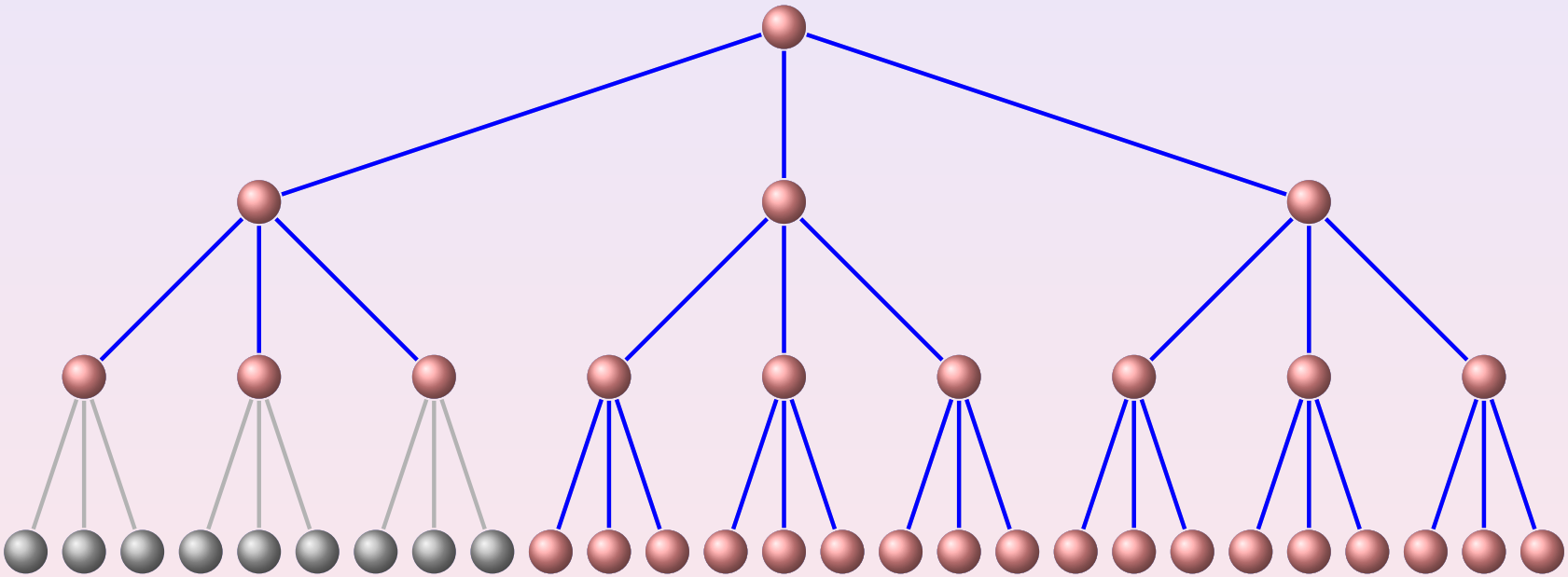
最右推导建树过程



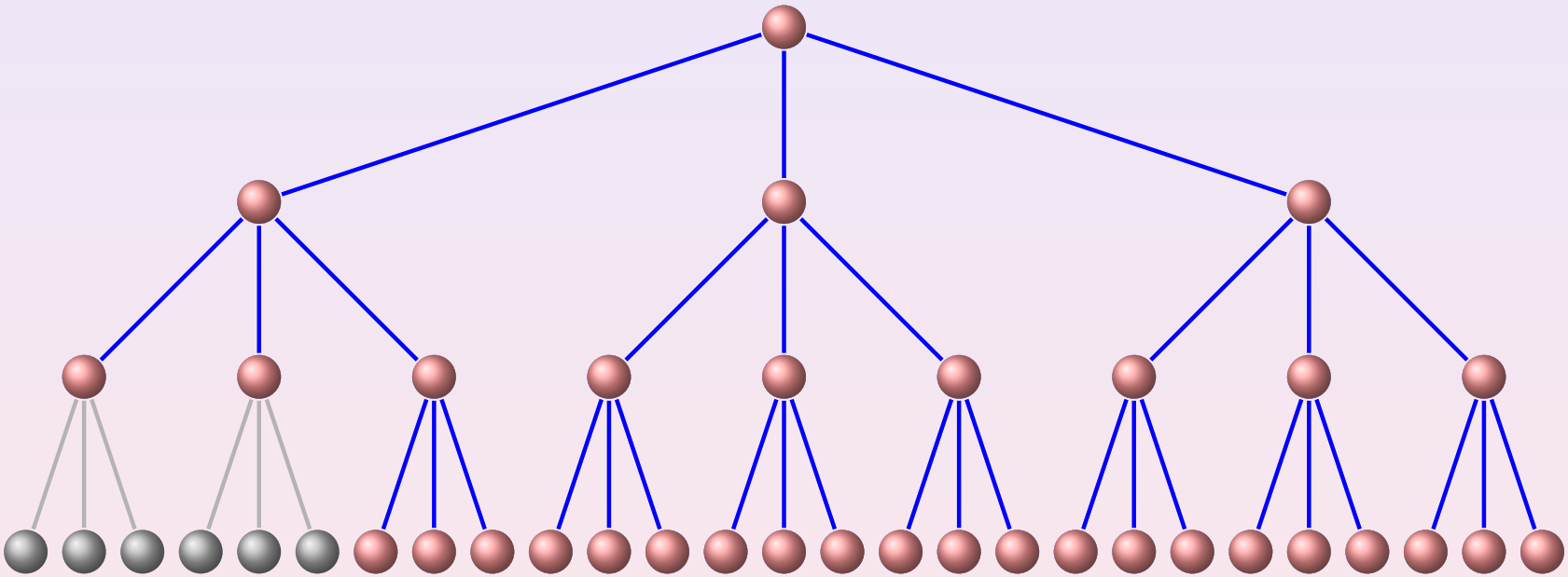
最右推导建树过程



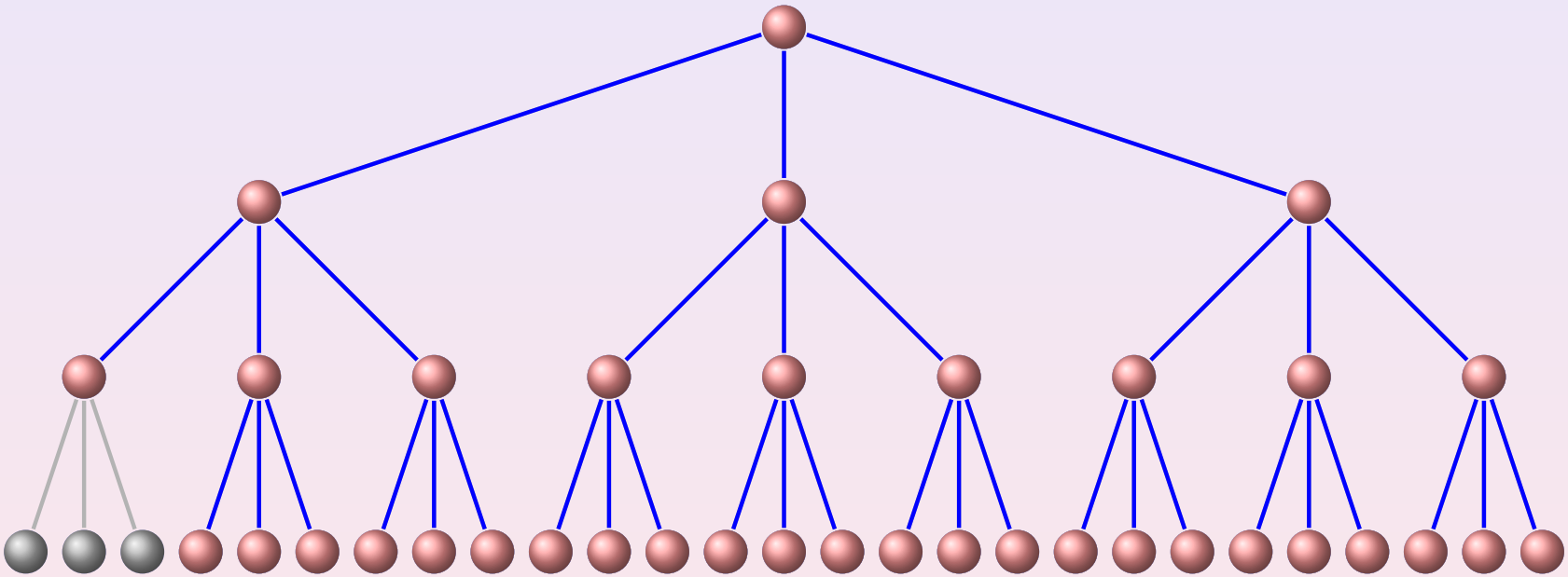
最右推导建树过程



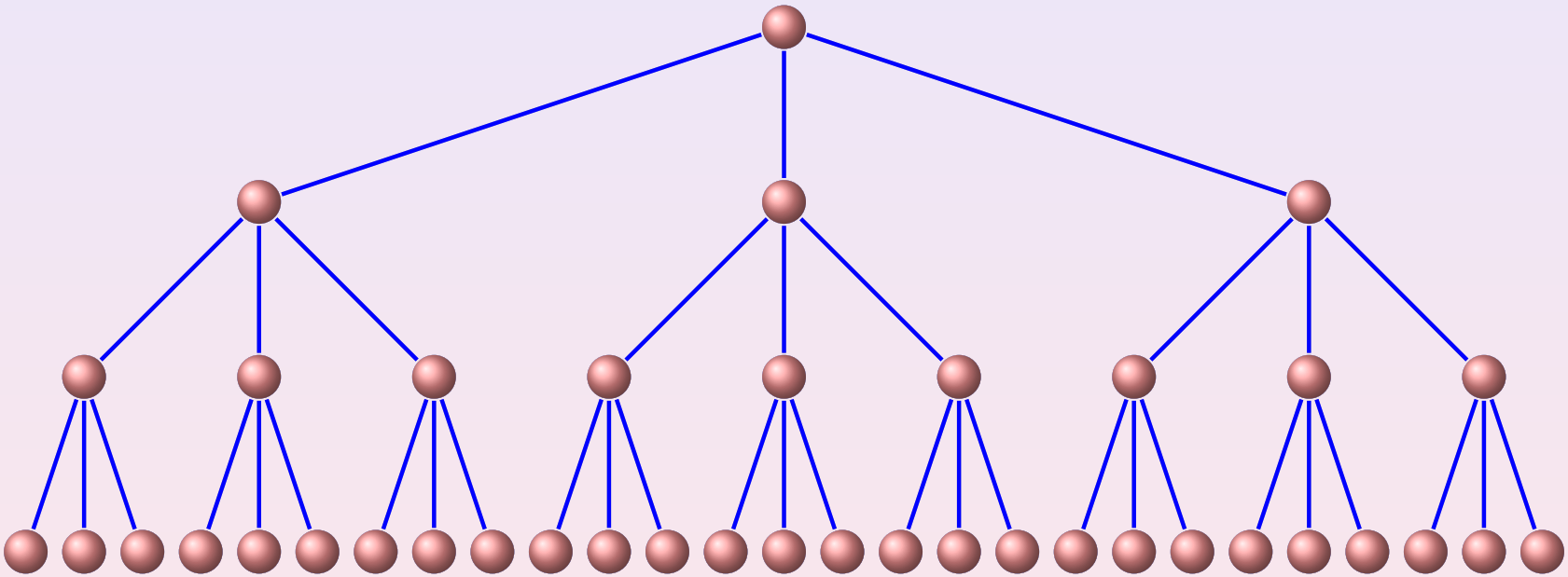
最右推导建树过程



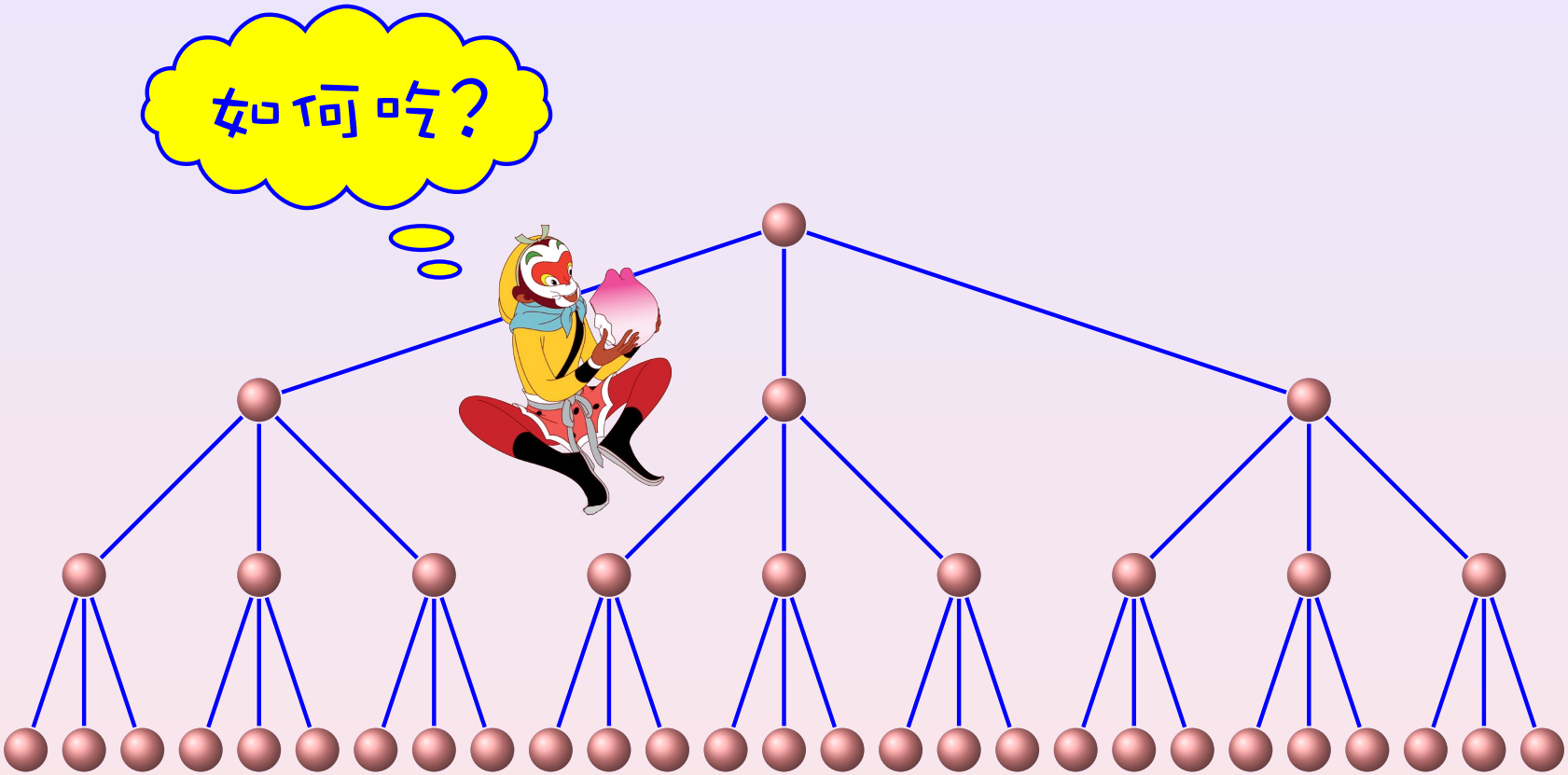
最右推导建树过程



最右推导建树过程

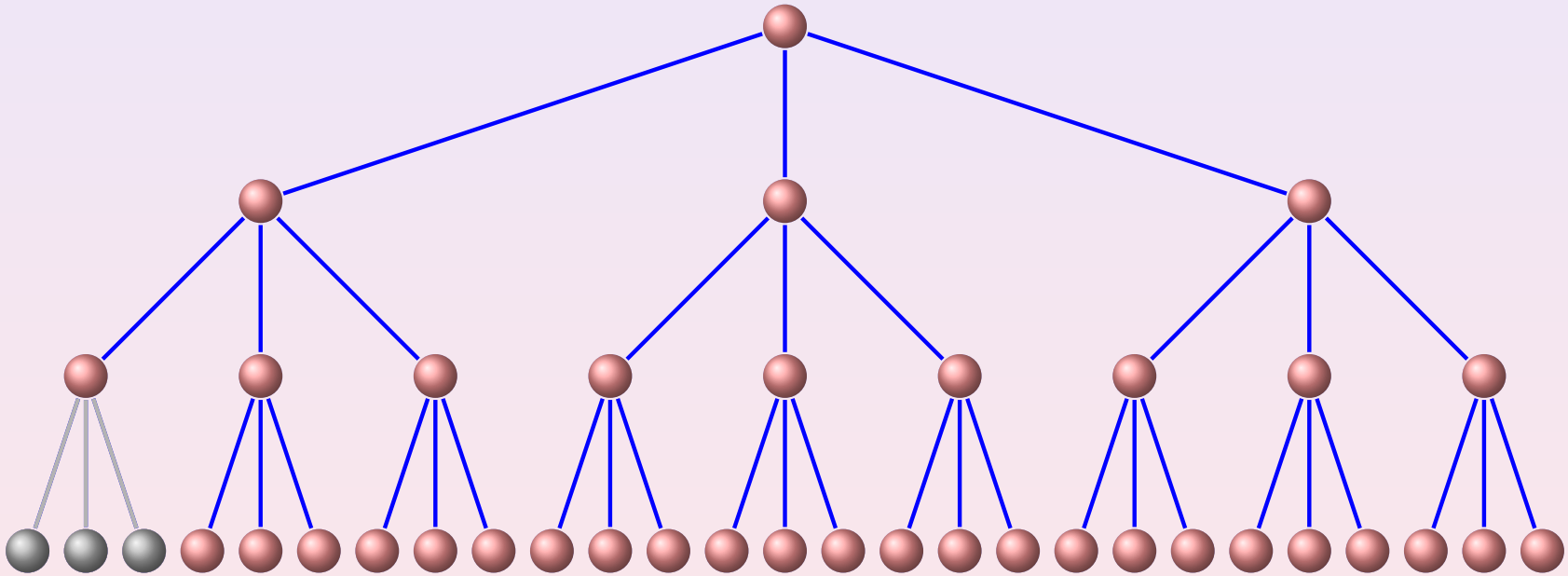


裁枝过程 --- 吃桃子的法则



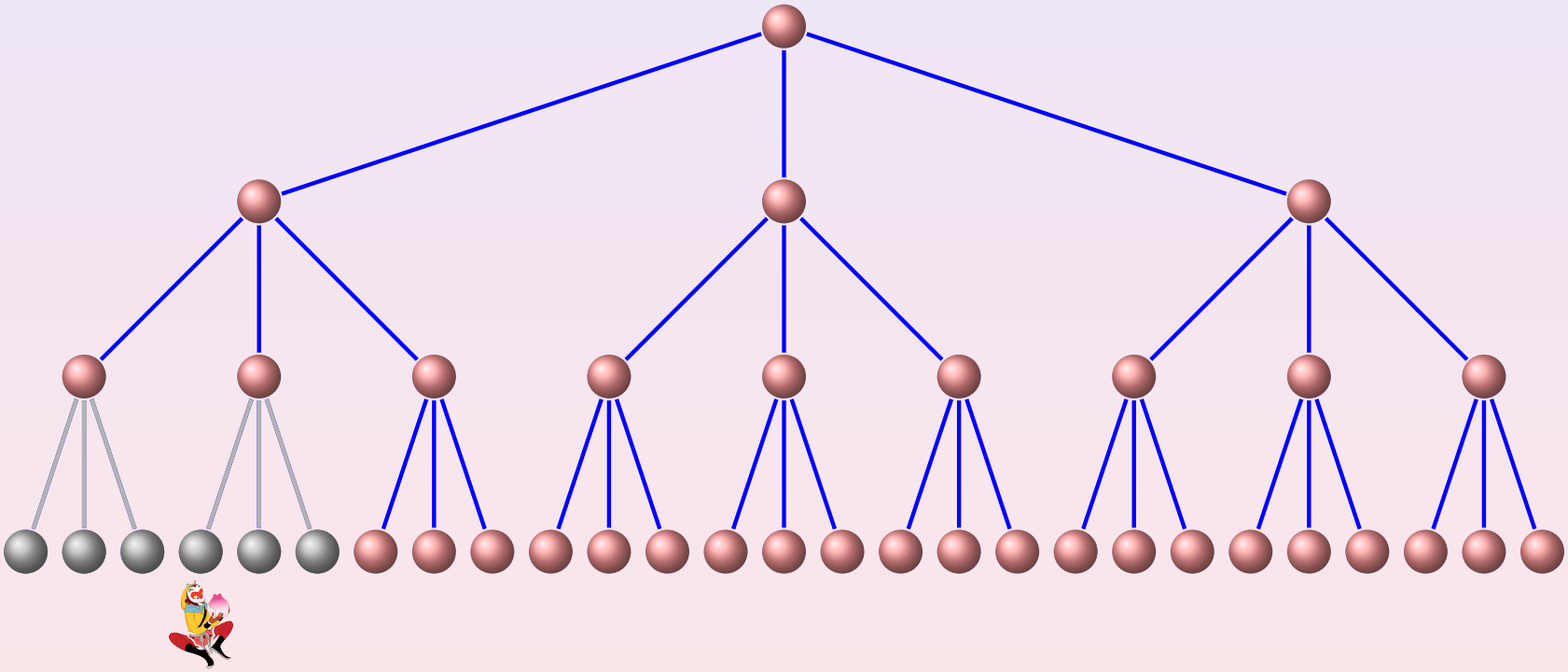
裁枝过程 --- 吃桃子的法则

1/ 同属一个父结点的叶结点要一同吃掉.
2/ 如果左边还有 1/, 就不能吃右边.



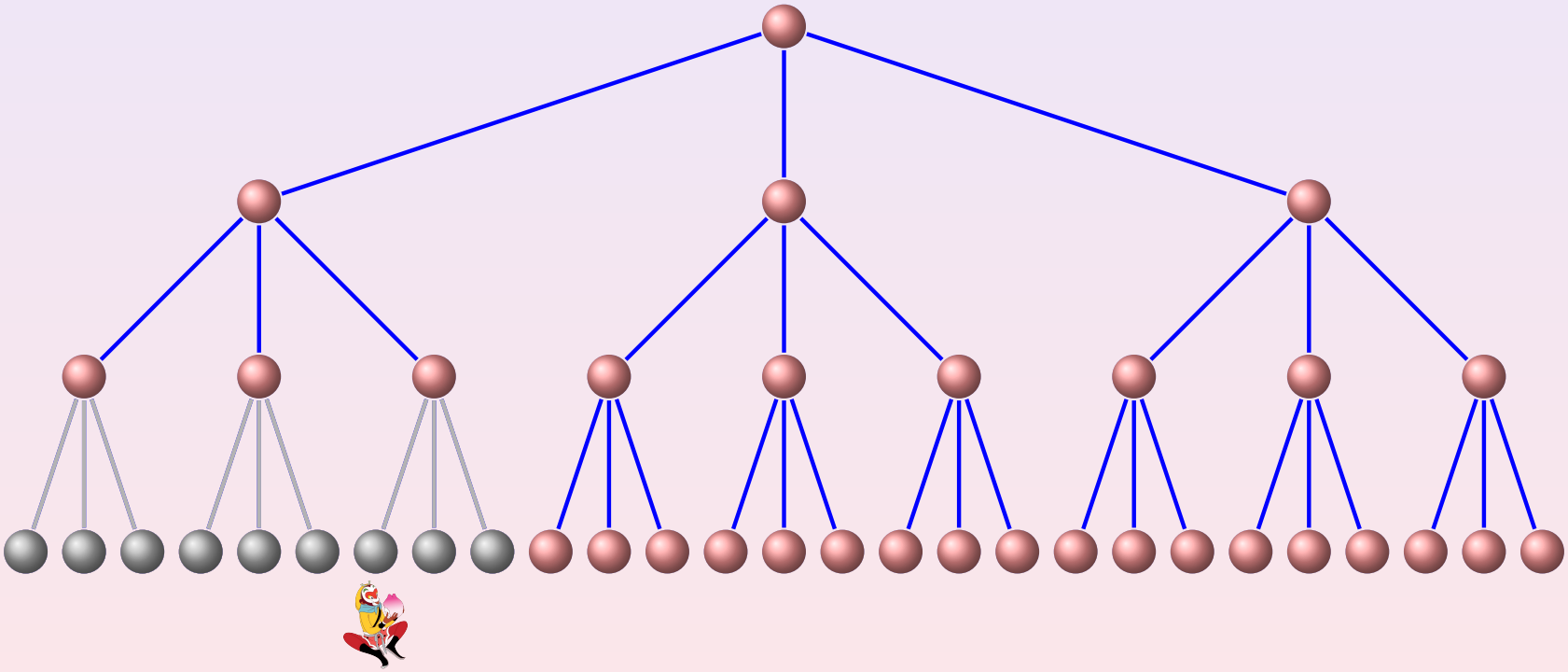
裁枝过程 --- 吃桃子的法则

1/ 同属一个父结点的叶结点要一同吃掉.
2/ 如果左边还有 1/, 就不能吃右边.



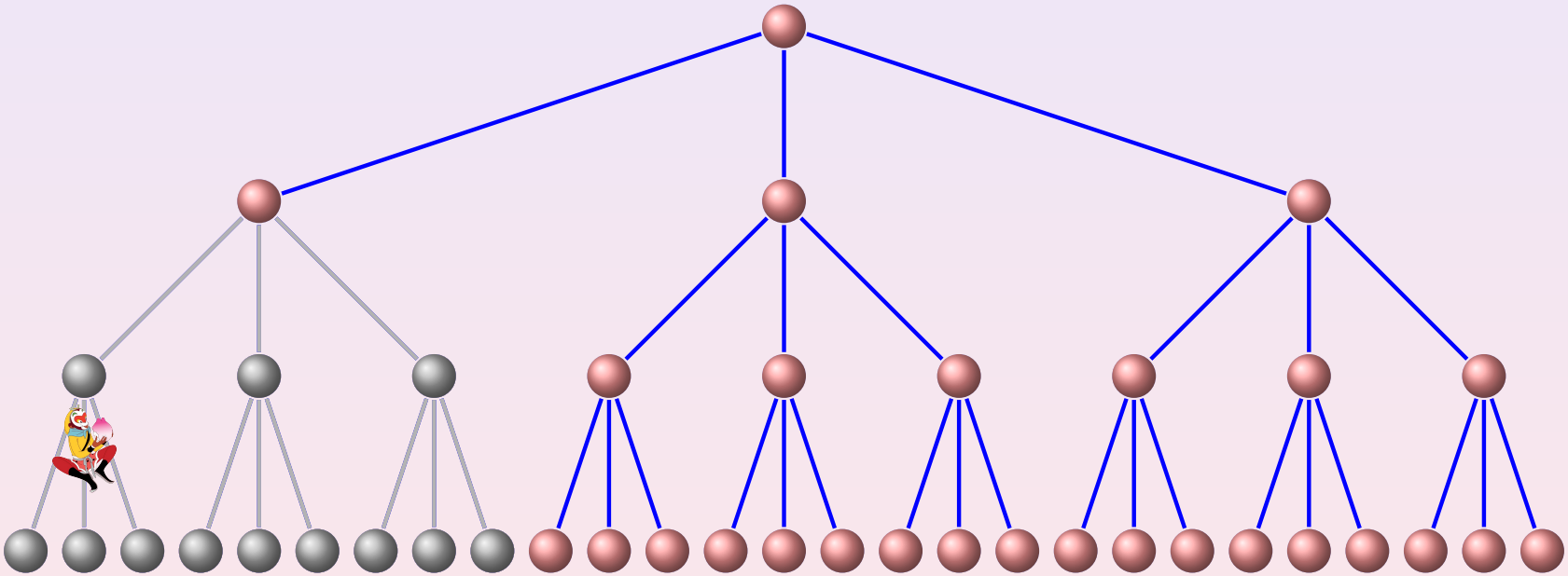
裁枝过程 --- 吃桃子的法则

1/ 同属一个父结点的叶结点要一同吃掉.
2/ 如果左边还有 1/, 就不能吃右边.



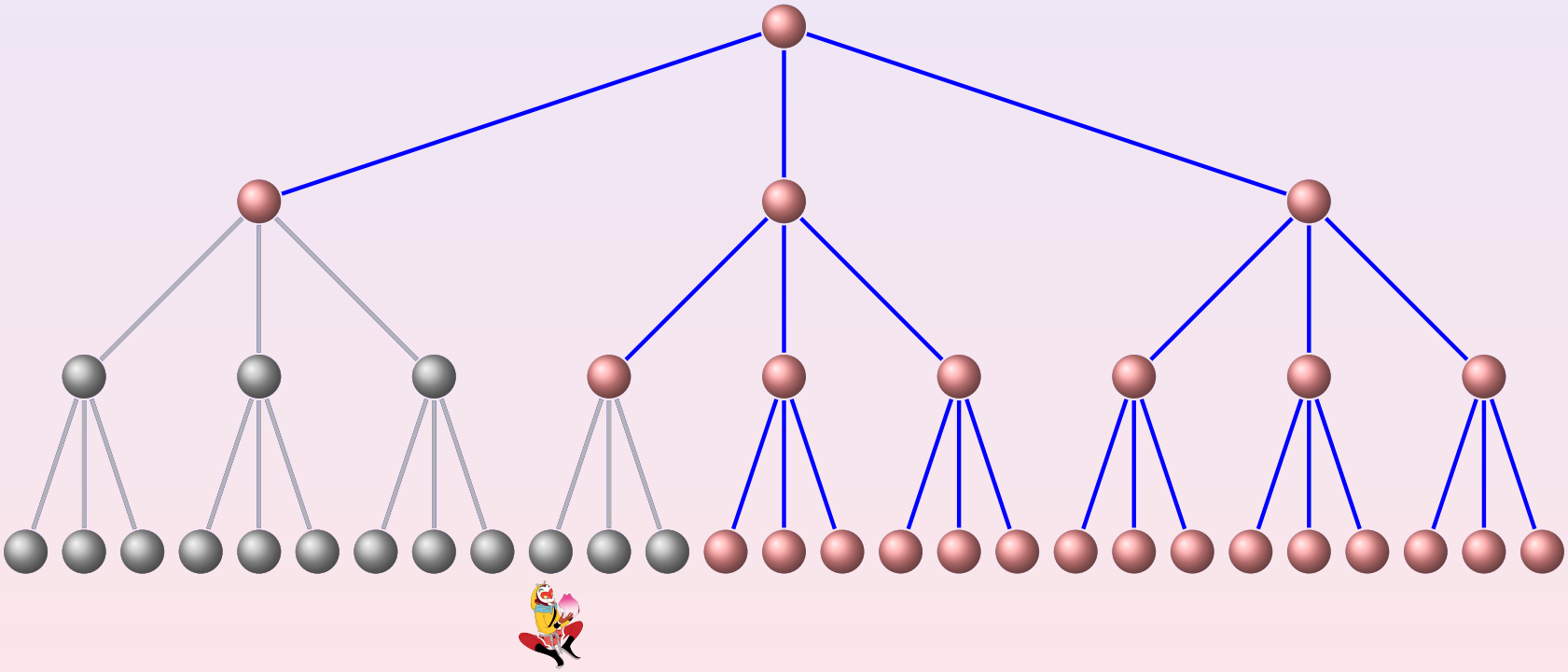
裁枝过程 --- 吃桃子的法则

1/ 同属一个父结点的叶结点要一同吃掉.
2/ 如果左边还有 1/, 就不能吃右边.



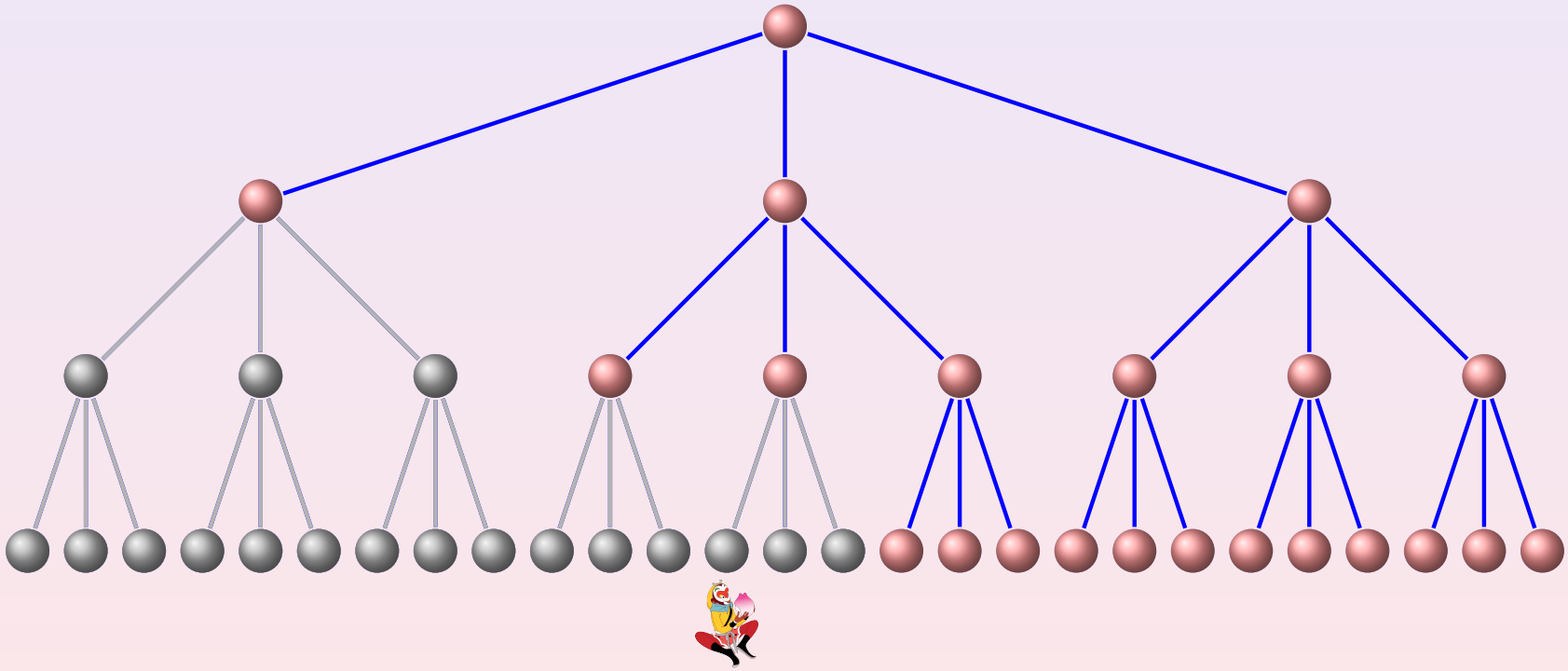
裁枝过程 --- 吃桃子的法则

1/ 同属一个父结点的叶结点要一同吃掉.
2/ 如果左边还有 1/, 就不能吃右边.



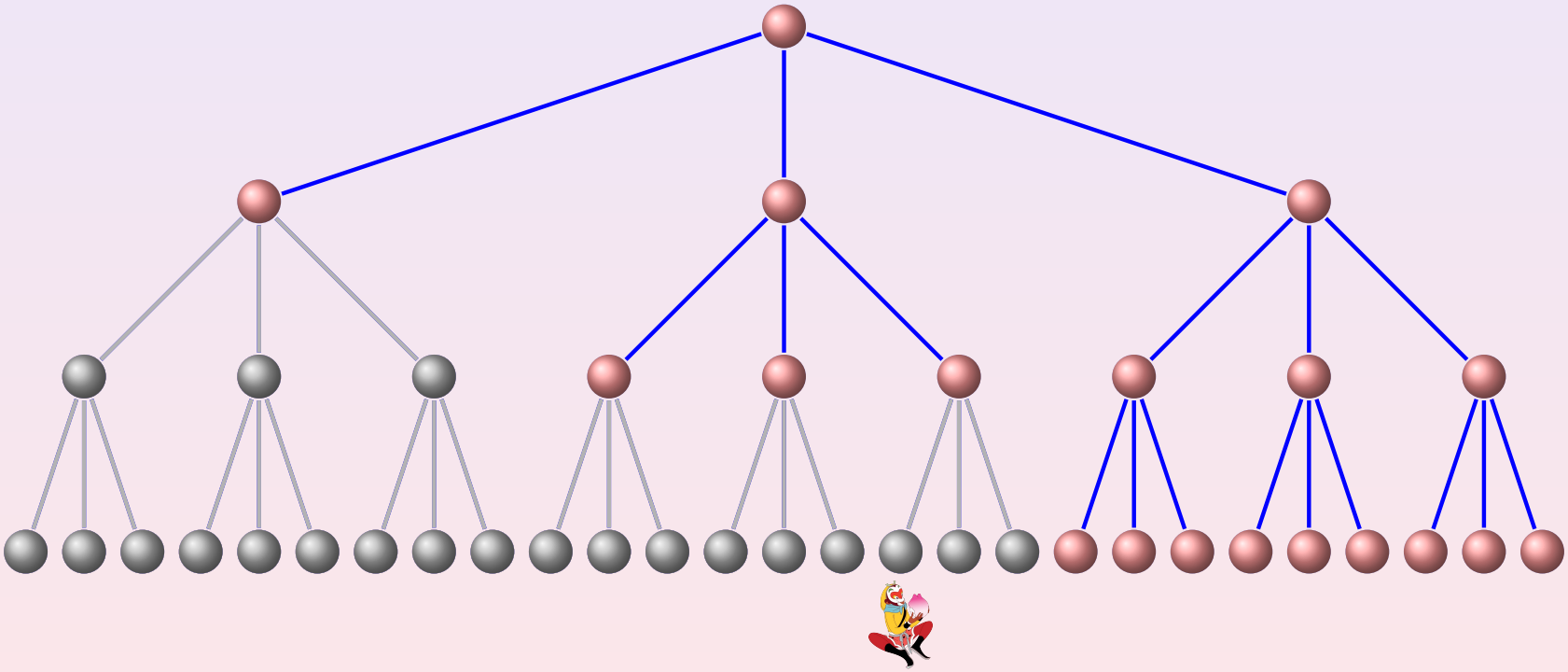
裁枝过程 --- 吃桃子的法则

1/ 同属一个父结点的叶结点要一同吃掉.
2/ 如果左边还有 1/, 就不能吃右边.



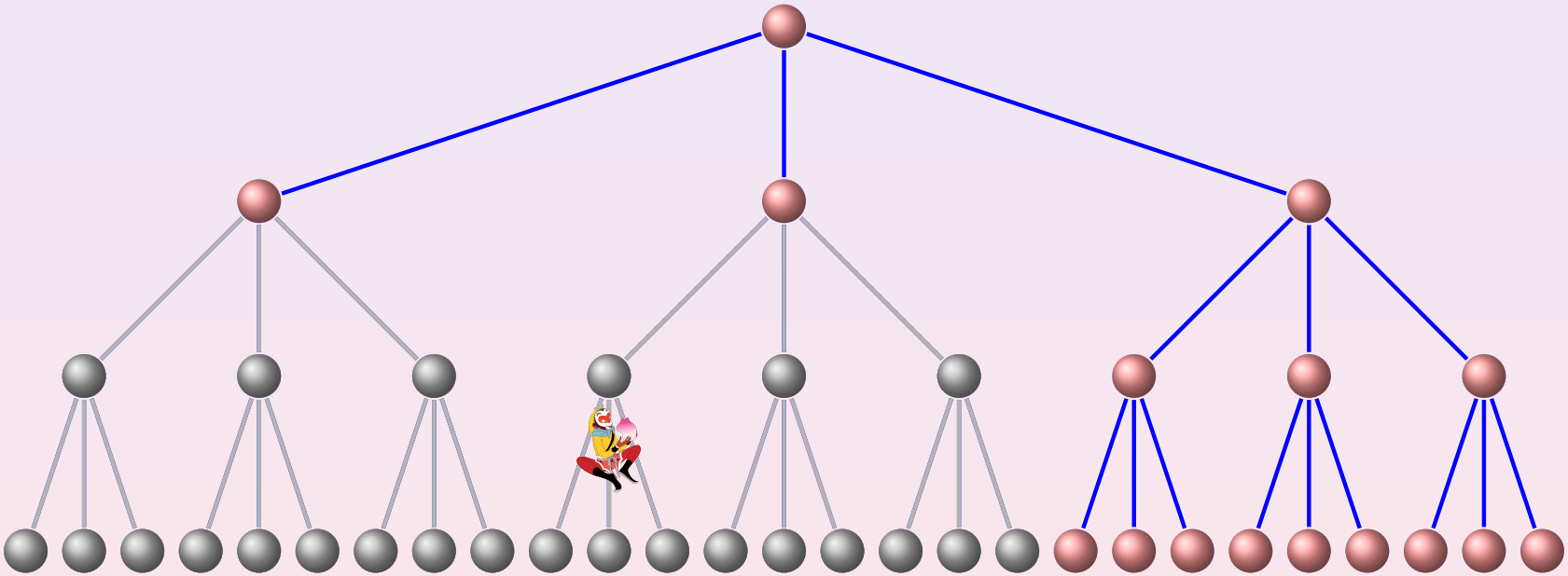
裁枝过程 --- 吃桃子的法则

1/ 同属一个父结点的叶结点要一同吃掉.
2/ 如果左边还有 1/, 就不能吃右边.



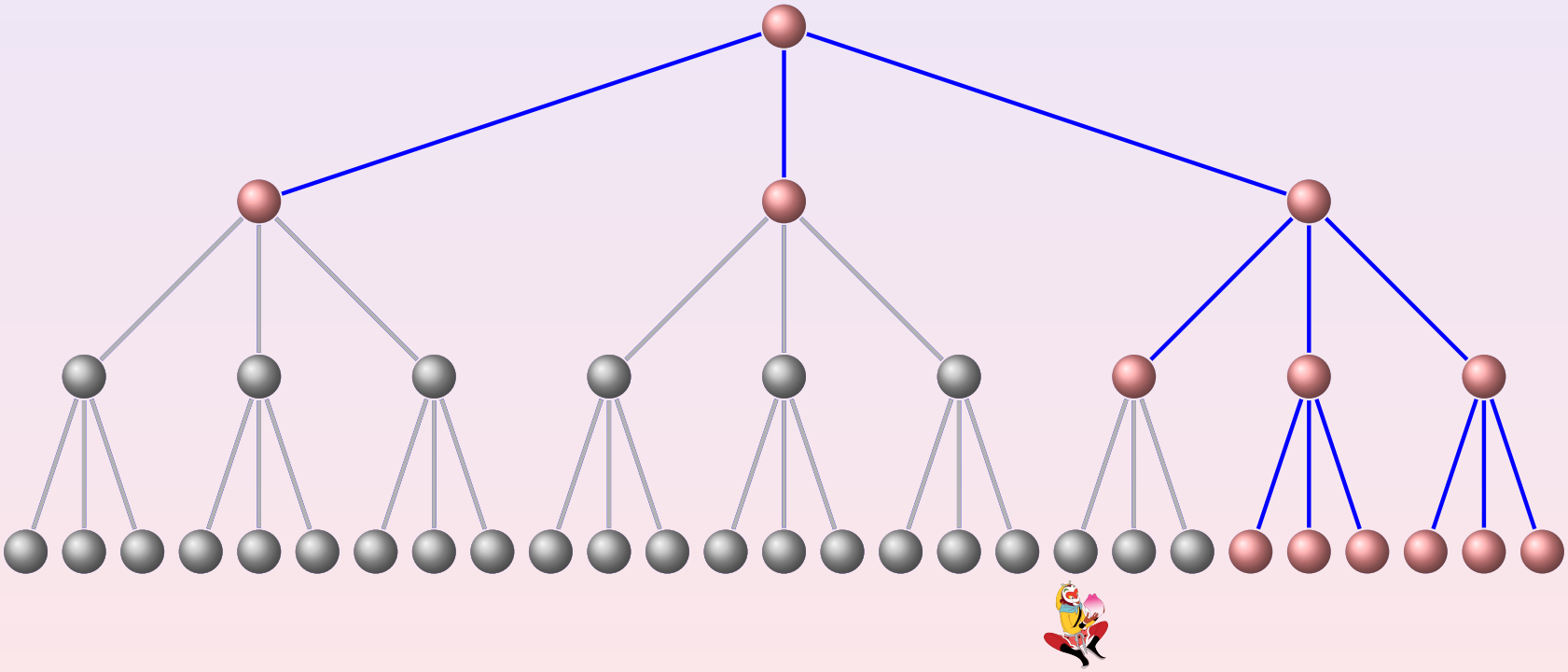
裁枝过程 --- 吃桃子的法则

1/ 同属一个父结点的叶结点要一同吃掉.
2/ 如果左边还有 1/, 就不能吃右边.



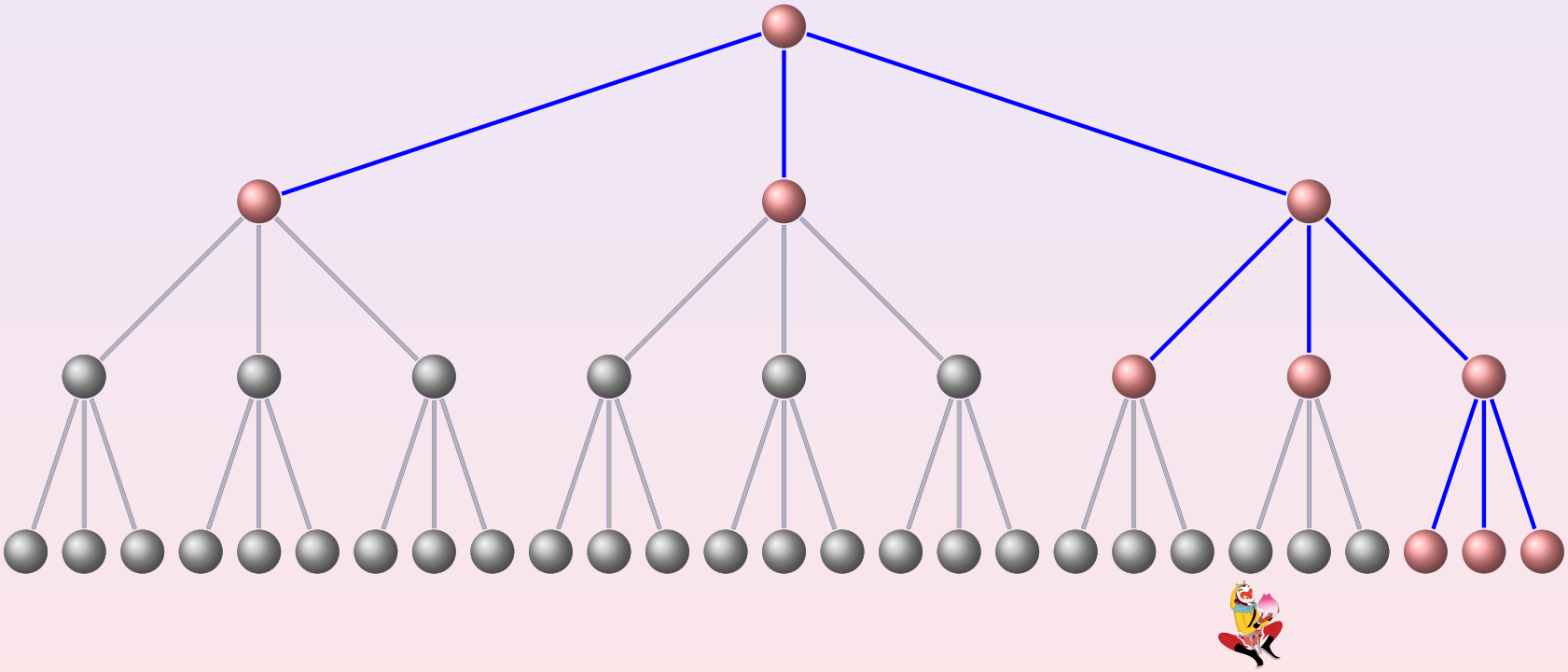
裁枝过程 --- 吃桃子的法则

1/ 同属一个父结点的叶结点要一同吃掉.
2/ 如果左边还有 1/, 就不能吃右边.



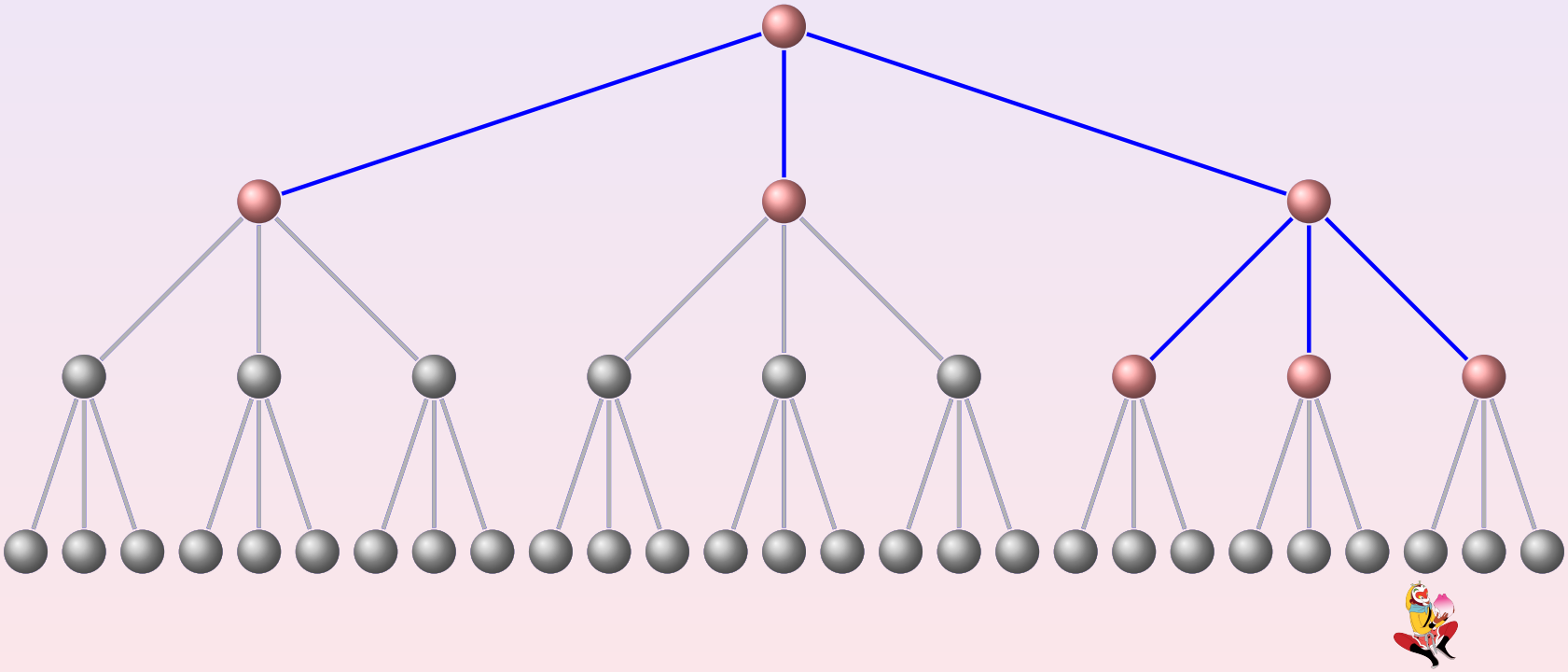
裁枝过程 --- 吃桃子的法则

1/ 同属一个父结点的叶结点要一同吃掉.
2/ 如果左边还有 1/, 就不能吃右边.



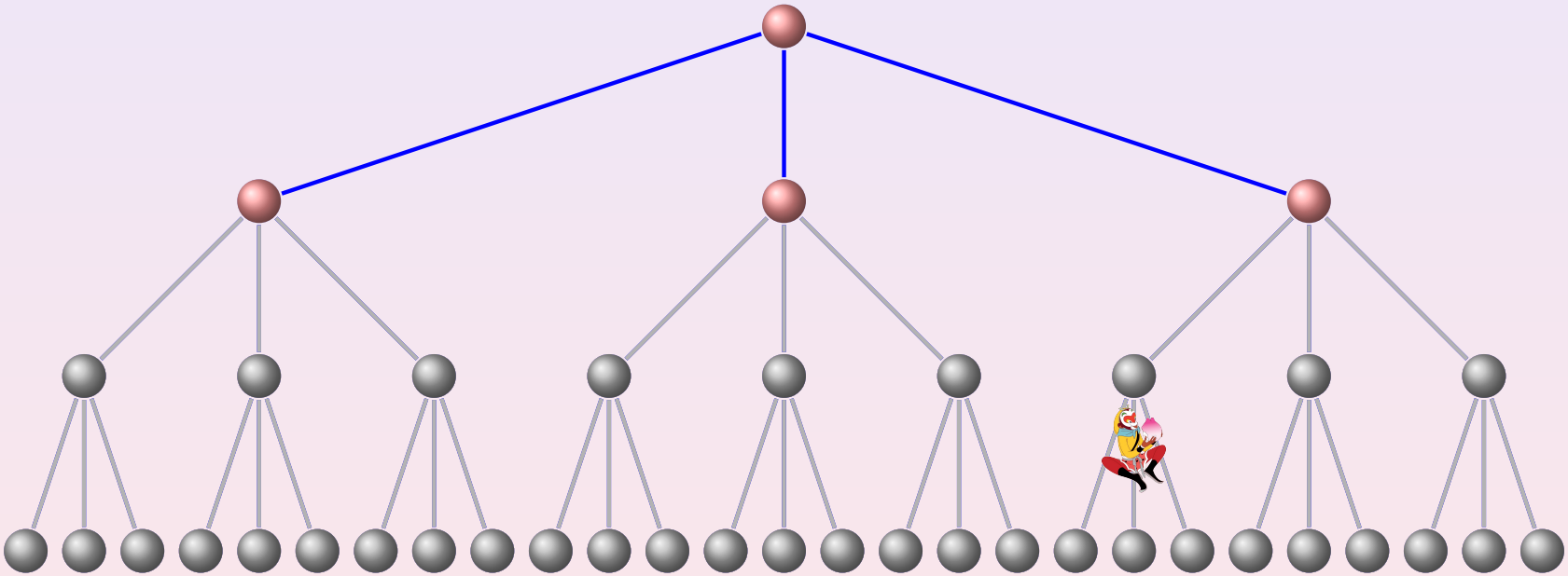
裁枝过程 --- 吃桃子的法则

1/ 同属一个父结点的叶结点要一同吃掉.
2/ 如果左边还有 1/, 就不能吃右边.



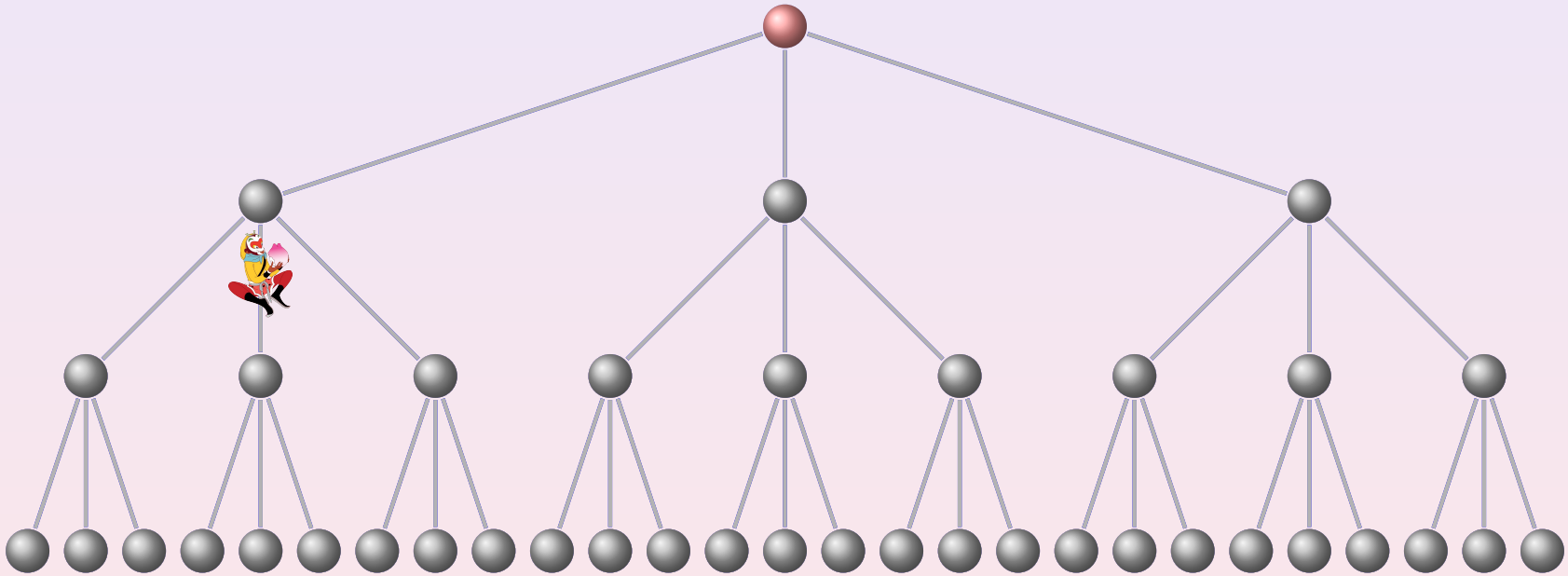
裁枝过程 --- 吃桃子的法则

1/ 同属一个父结点的叶结点要一同吃掉.
2/ 如果左边还有 1/, 就不能吃右边.



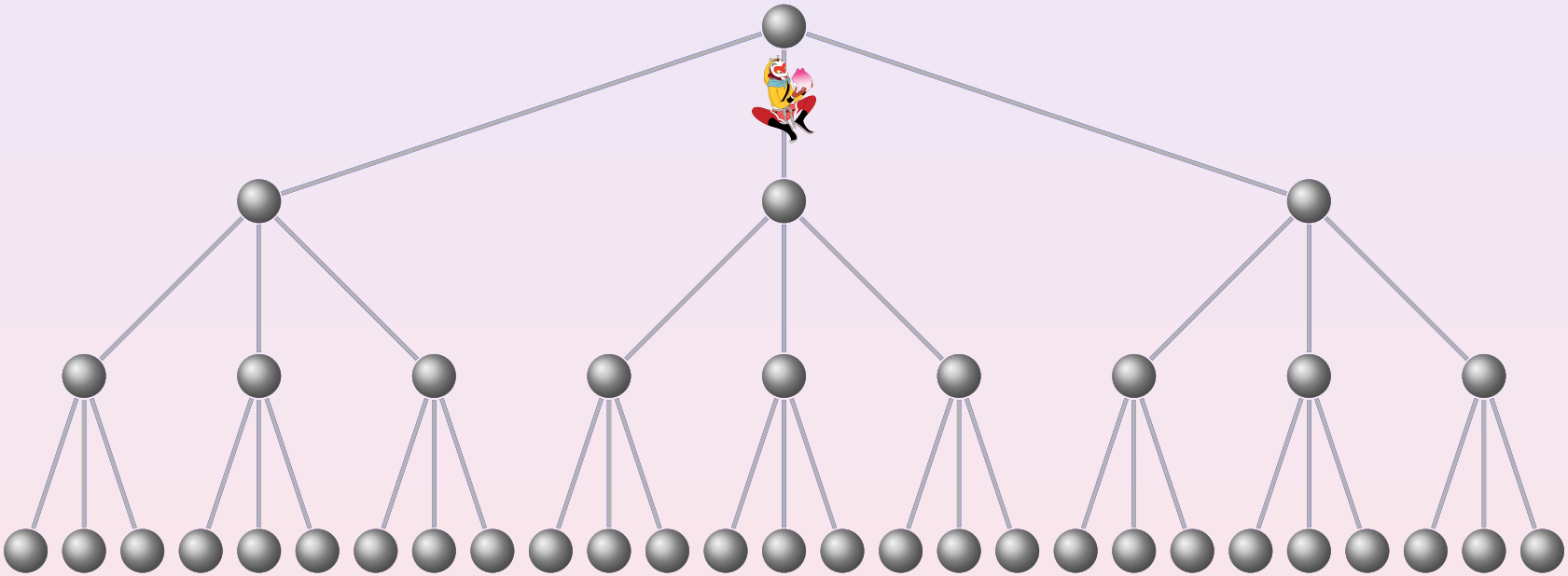
裁枝过程 --- 吃桃子的法则

1/ 同属一个父结点的叶结点要一同吃掉.
2/ 如果左边还有 1/, 就不能吃右边.



裁枝过程 --- 吃桃子的法则

1/ 同属一个父结点的叶结点要一同吃掉.
2/ 如果左边还有 1/, 就不能吃右边.



现实的困难

- 树的情况未知，看不到树的完整结构，只能看到已吃完的树干和当前的树叶.
- 在吃完一组叶结点后如何动作，向前或向上，且不能有丝毫的错误 (无回溯).

解决问题的思路

- 保留已吃过的树干作为下次吃法的判断依据.

相关性质

- 每组儿子一定是产生式右边的文法符号串，吃过一组后应知道其父亲.
- 已吃过的树干从左到右排列一定是对树的``垂直 + 水平''遍历所经历的叶结点序列.

最左归约的性质

- 被归约的对象一定是某一产生式 RHS 的文法符号串 α，将 α 看成是 $T \cup N$ 为字母表的一个正则表达式，$\beta \in (T \cup N)^*$ 是要归约的对象，则在 β 中寻找 α 子串可以用自动机解决.
- 被归约对象之后一定是终结符号串：

推导过程：

$$\begin{aligned} A &\underset{rm}{\overset{*}{\Rightarrow}} \alpha B w \\ &\underset{rm}{\Rightarrow} \alpha\beta w \end{aligned}$$

归约过程：

$$\begin{aligned} &\alpha\beta w \\ \underset{rm}{\overset{*}{\Leftarrow}} \quad &\alpha B w \end{aligned}$$

- 不确定性：
  - exp PLUS term TIMES ID 可以归约的对象有：

1) exp PLUS term TIMES <u>ID</u>
2) exp PLUS <u>term</u> TIMES ID
3) <u>exp PLUS term</u> TIMES ID

但是只有 (1) 是唯一一个正确的.

句柄

- 正确的被归约的对象称为句柄 (handle)，即它能保证被归约后一定还保持着最右句型. 例子

  1) $\text{exp}\ \text{PLUS}\ \text{term}\ \text{TIMES}\ \underline{\text{ID}} \underset{\text{rm}}{\Leftarrow} \text{exp}\ \text{PLUS}\ \text{term}\ \text{TIMES}\ \underline{\text{fac}}$
  2) $\text{exp}\ \text{PLUS}\ \underline{\text{term}}\ \text{TIMES}\ \text{ID} \underset{\text{rm}}{\Leftarrow} \text{exp}\ \text{PLUS}\ \underline{\text{exp}}\ \text{TIMES}\ \text{ID}$
  3) $\underline{\text{exp}\ \text{PLUS}\ \text{term}}\ \text{TIMES}\ \text{ID} \underset{\text{rm}}{\Leftarrow} \underline{\text{exp}}\ \text{TIMES}\ \text{ID}$

  只有 (1) 是句柄.
- 如果文法是没有二义性的，则一个最右句型的句柄是唯一的.
- 寻找句柄是解决自底向上分型的关键.

page-break

移进 - 归约（shift-reduce）分析器的设计

移进 - 归约

stack	action	rest of scan
$ exp PLUS term	shift	TIMES ID $
$ exp PLUS term TIMES	shift	ID $
$ exp PLUS term TIMES ID	reduce	$
$ exp PLUS term TIMES fac	reduce	$
$ exp PLUS term	reduce	$
$ exp	success	$

例子

移进 - 归约（shift-reduce）分析器的设计

移进 - 归约

stack	action	rest of scan
$ exp PLUS term	shift	TIMES ID $
$ exp PLUS term TIMES	shift	ID $
$ exp PLUS term TIMES ID	reduce	$
$ exp PLUS term TIMES fac	reduce	$
$ exp PLUS term	reduce	$
$ exp	success	$

例子

移进 - 归约（shift-reduce）分析器的设计

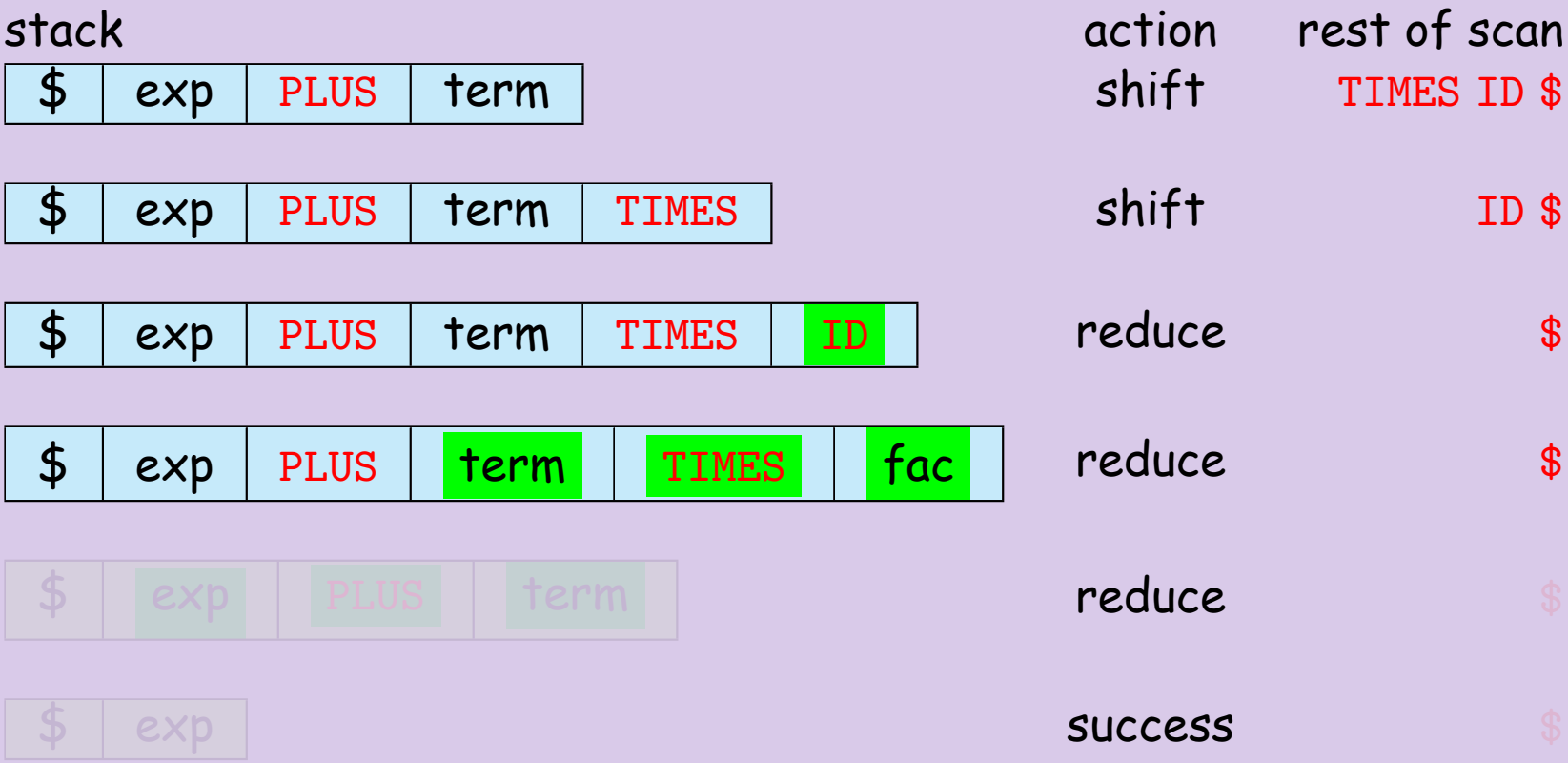
移进 - 归约

stack	action	rest of scan
$ exp PLUS term	shift	TIMES ID $
$ exp PLUS term TIMES	shift	ID $
$ exp PLUS term TIMES ID	reduce	$
$ exp PLUS term TIMES fac	reduce	$
$ exp PLUS term	reduce	$
$ exp	success	$

例子

移进 - 归约（shift-reduce）分析器的设计

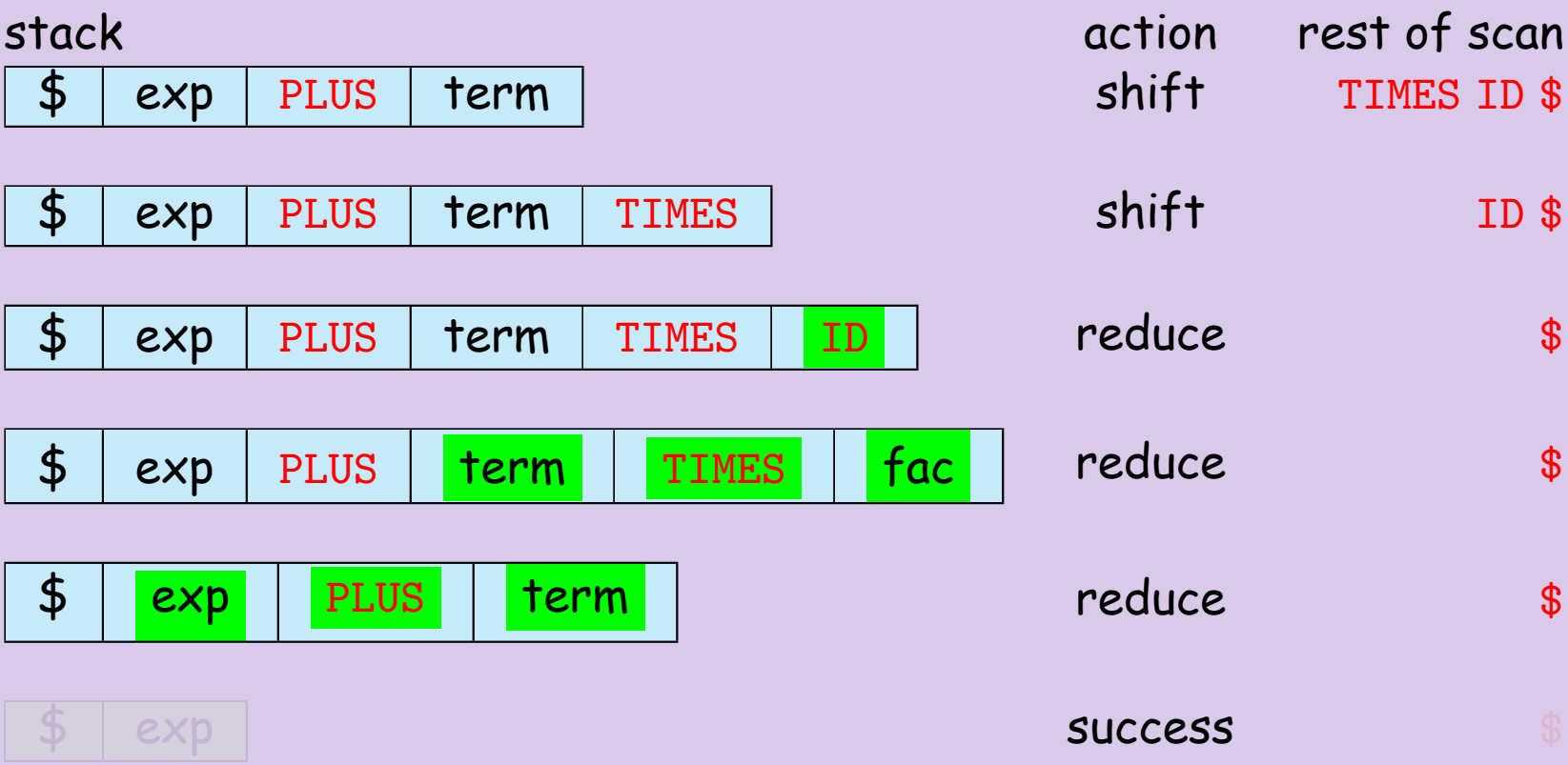
移进 - 归约



例子

移进 - 归约（shift-reduce）分析器的设计

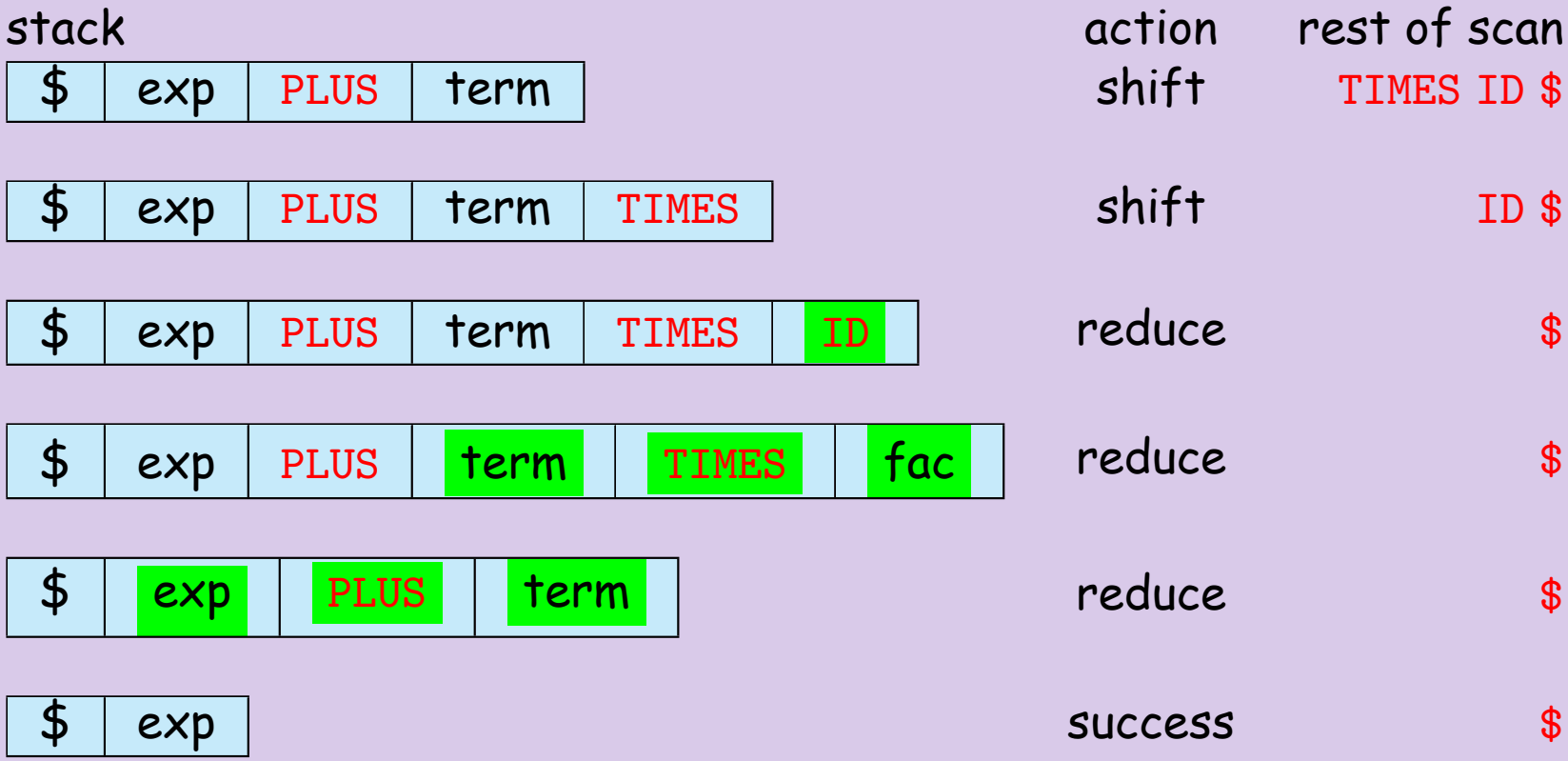
移进 - 归约



例子

移进 - 归约（shift-reduce）分析器的设计

移进 - 归约



例子

移进 - 归约（shift-reduce）分析器的设计.

分析栈

- 保留已经归约的句型.
- 自左向右扫描.
- "stack" + "rest of scan" = 最右句型.
- 句柄总是在栈顶形成.
- 无回溯.

问题

- 在归约无误的前提下，句柄是否能保证总是在栈顶形成.
- 如何不回看栈中元素，仅根据栈顶的状态和当前的输入就能够正确地做出移进或归约的操作.

句柄一定在栈顶 (1/2)

句柄一定在栈顶

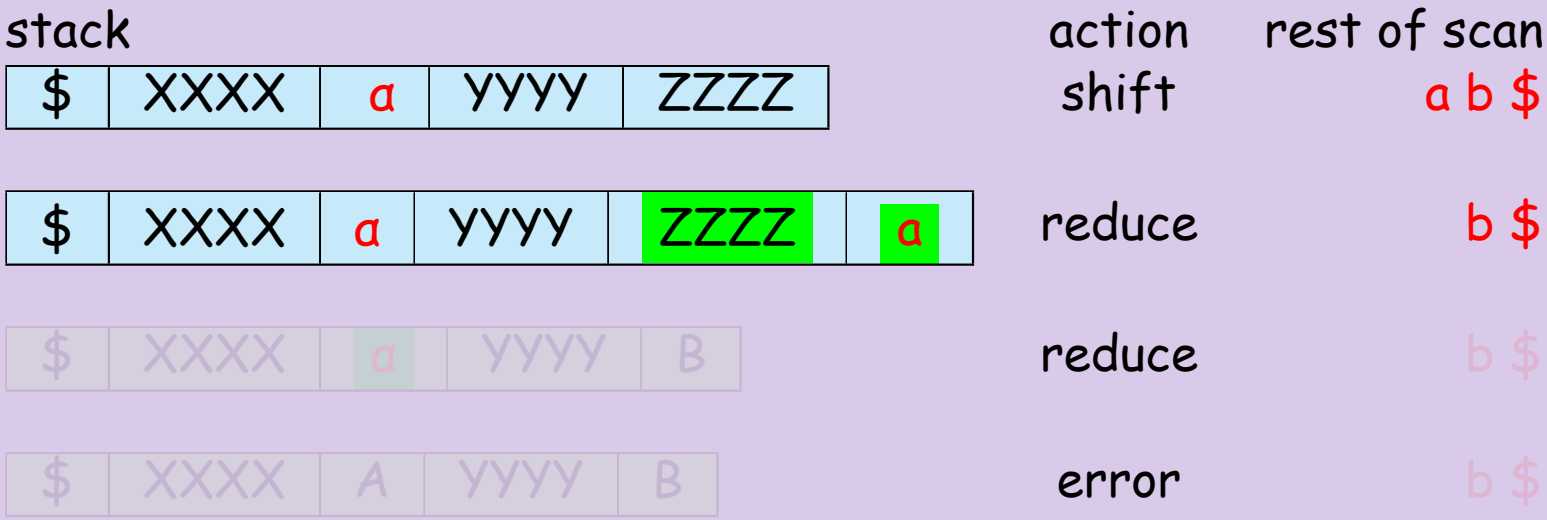
stack					action	rest of scan
$	XXXX	a	YYYY	ZZZZ	shift	a b $
$	XXXX	a	YYYY	ZZZZ	a	reduce	b $
$	XXXX	a	YYYY	B	reduce	b $
$	XXXX	A	YYYY	B	error	b $

这与最右推导矛盾：

$$XXXX\ A\ YYYY\ B\ w \underset{rm}{\Rightarrow} XXXX\ A\ YYYY\ ZZZZa\ w \underset{rm}{\Rightarrow} XXXX\ a\ YYYY\ ZZZZa\ w$$

句柄一定在栈顶 (1/2)

句柄一定在栈顶



这与最右推导矛盾：

$$XXXX\ A\ YYYY\ B\ w \Rightarrow_{rm} XXXX\ A\ YYYY\ ZZZZa\ w \Rightarrow_{rm} XXXX\ \alpha\ YYYY\ ZZZZa\ w$$

句柄一定在栈顶 (1/2)

句柄一定在栈顶

stack	action	rest of scan
$ XXXX a YYYY ZZZZ	shift	a b $
$ XXXX a YYYY ZZZZ a	reduce	b $
$ XXXX a YYYY B	reduce	b $
$ XXXX A YYYY B	error	b $

这与最右推导矛盾：

$$XXXX\ A\ YYYY\ B\ w \Rightarrow_{rm} XXXX\ A\ YYYY\ ZZZZa\ w \Rightarrow_{rm} XXXX\ a\ YYYY\ ZZZZa\ w$$

句柄一定在栈顶 (1/2)

句柄一定在栈顶

stack	action	rest of scan
$ XXXX a YYYY ZZZZ	shift	a b $
$ XXXX a YYYY ZZZZ a	reduce	b $
$ XXXX a YYYY B	reduce	b $
$ XXXX A YYYY B	error	b $

这与最右推导矛盾：

$$\begin{aligned} & XXXX\ A\ YYYY\ B\ w \\ \Rightarrow_{rm} & XXXX\ A\ YYYY\ ZZZZa\ w \\ \Rightarrow_{rm} & XXXX\ a\ YYYY\ ZZZZa\ w \end{aligned}$$

句柄一定在栈顶 (1/2)

句柄一定在栈顶

stack	action	rest of scan
$ XXXX a YYYY ZZZZ	shift	a b $
$ XXXX a YYYY ZZZZ a	reduce	b $
$ XXXX a YYYY B	reduce	b $
$ XXXX A YYYY B	error	b $

这与最右推导矛盾：

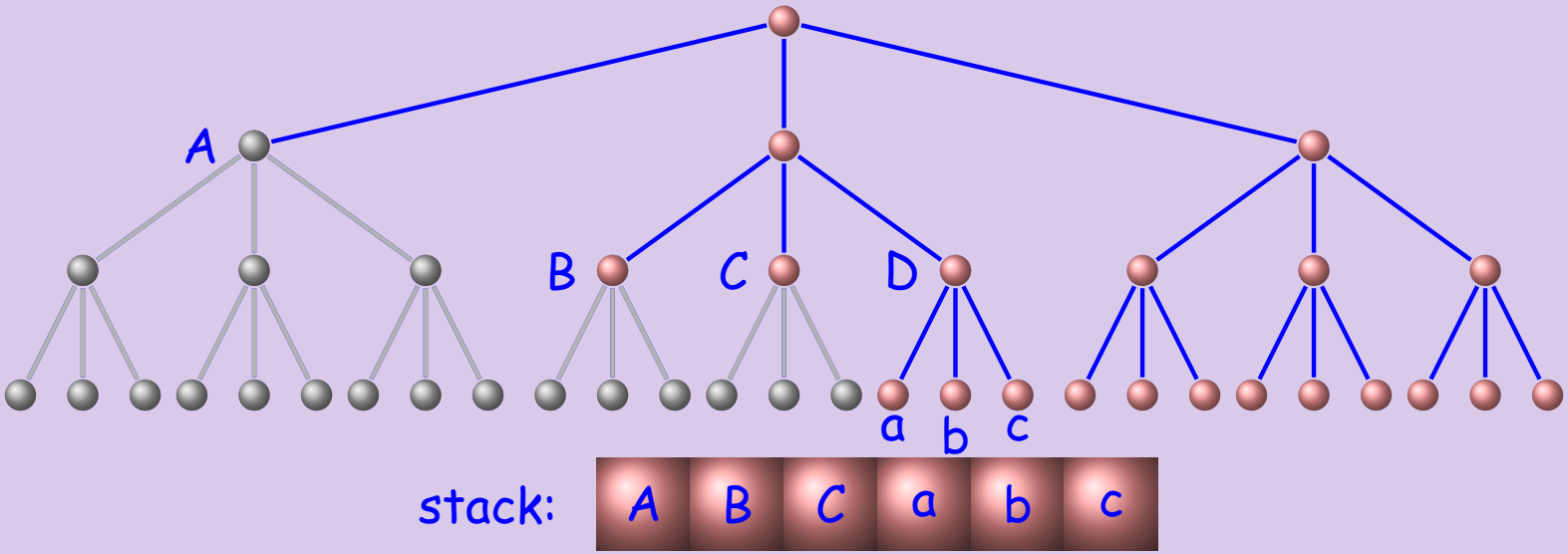
$$\begin{aligned} & XXXX\ A\ YYYY\ B\ w \\ \Rightarrow_{rm} & XXXX\ A\ YYYY\ ZZZZa\ w \\ \Rightarrow_{rm} & XXXX\ a\ YYYY\ ZZZZa\ w \end{aligned}$$

句柄一定在栈顶 (2/2)

原因

- 栈内的元素一定是已经归约到不能再归约的对象，否则，由于是句柄，在新的移进前还要归约.
- 句柄的最后一个文法符号一定是栈顶元素.
- 最右归约过程是自底向上从左到右的裁枝过程.

裁枝



如何判断句柄已经在栈顶形成.

活前缀 (viable prefix)

- 最右句型中句柄之前 (含句柄) 的文法符号串.
- 最右句型的一个前缀, 该前缀不含有句柄以后的文法符号.
- 移进 - 归约分析栈中的符号串.

Example

Sentential form : exp PLUS termTIMES ID w

Viable prefix {
- exp
- exp PLUS
- exp PLUS term
- exp PLUS termTIMES
- exp PLUS termTIMES ID

活前缀的性质

- 任一活前缀都可以在其后增加终结符而形成一个右句型.
- 活前缀中出现在前的文法符号的辈分一定不低于后面的.
- 活前缀中的任一文法符号一定是最右推导中从未被展开的符号.
- 如果分析器能识别活前缀并且能预测形成活前缀的所有可能的最右推导就能进行无回溯的分析.

Example

Sentential form : exp PLUS termTIMES ID w

可以形成最右句型的 w:

$$w = \begin{cases} \$ \\ \text{PLUS ID} \\ \text{TIMES ID} \\ \text{PLUS LPID RP} \\ \cdots \end{cases}$$

在以下讨论中用+, *, (和)分别表示终结符PLUS, TIMES, LP 和 RP.

自动机所接受的文法符号串：活前缀.

相关性质

- 活前缀是语法树由父亲到其最左儿子垂直向下遍历或具有相同父亲结点从最左端开始自左向右水平遍历所经过的节点序列.
- 自动机从开始状态出发的任何一条路经与上述遍历一一对应.
- 即某一文法符号串是活前缀，当且仅当，它是自动机从开始状态到某一个或多个状态所经历的边的序列.
- 形成某一活前缀的所有可能的最右推导与自动机在接受该活前缀后所到达的状态集一致.
- 自动机从开始状态出发到达任何一个状态，其状态对应的项目一定能最终形成句柄.
- 该自动机称为识别活前缀的自动机(简称前缀自动机).

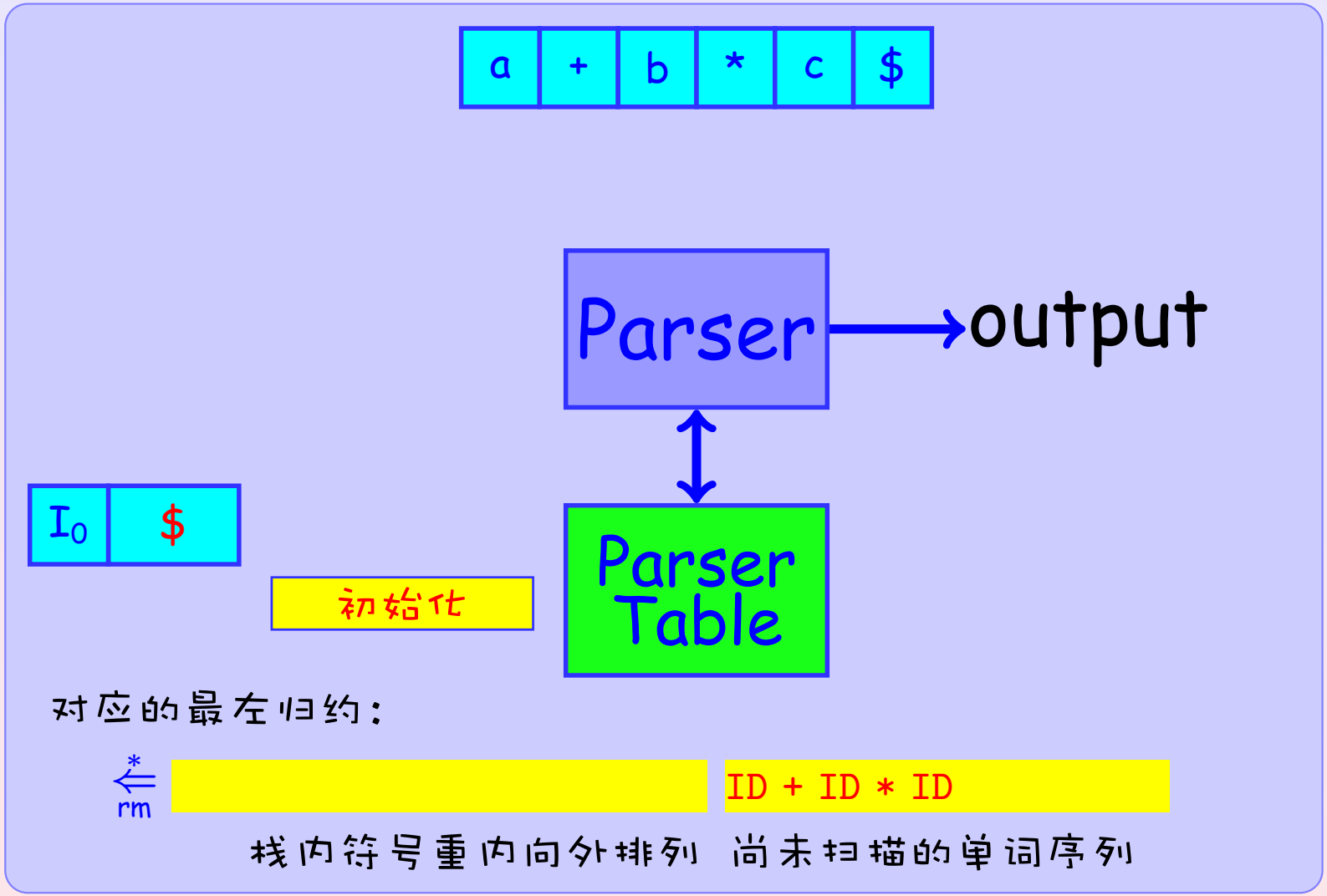
活前缀与有效项目

定义与定理

- 设 $\alpha\beta_1$ 是一个活前缀，设 $A \to \beta_1 \bullet \beta_2$ 是一个以 β_1 为前缀的项目，称 $A \to \beta_1 \bullet \beta_2$ 是 $\alpha\beta_1$ 的一个有效项目 iff $S' \underset{rm}{\overset{*}{\Rightarrow}} \alpha\beta_1\beta_2 w$.
- 表示活前缀 $\alpha\beta_1$ 中的后缀 β_1 可以形成 $A \to \beta_1\beta_2$ 的句柄.
- $\beta_1 \neq \varepsilon$，表示当前活前缀中还没有句柄中的一个符号.
- $\beta_2 = \varepsilon$，表示句柄已经形成.
- 所有的有效项目集合涵盖了从文法开始符号最右推出该活前缀的所有可能.
- 任何一个活前缀的有效项目就是自动机从开始状态出发在接受该活前缀后所能到达的状态对应的项目.
- 分析就是根据状态对应项目和当前的输入选择唯一的移进 - 归约操作.

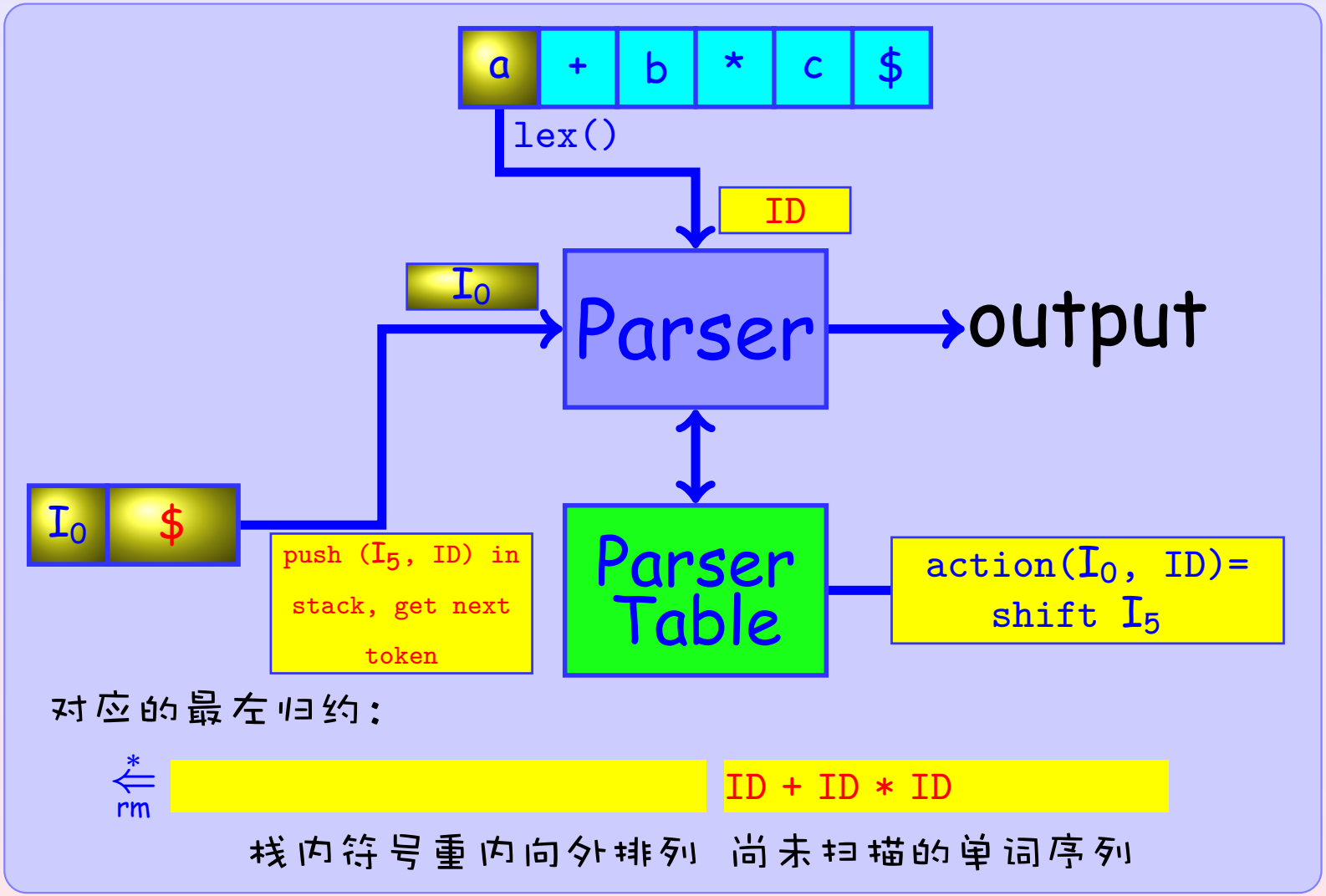
page-break

LR 分析法动态演示

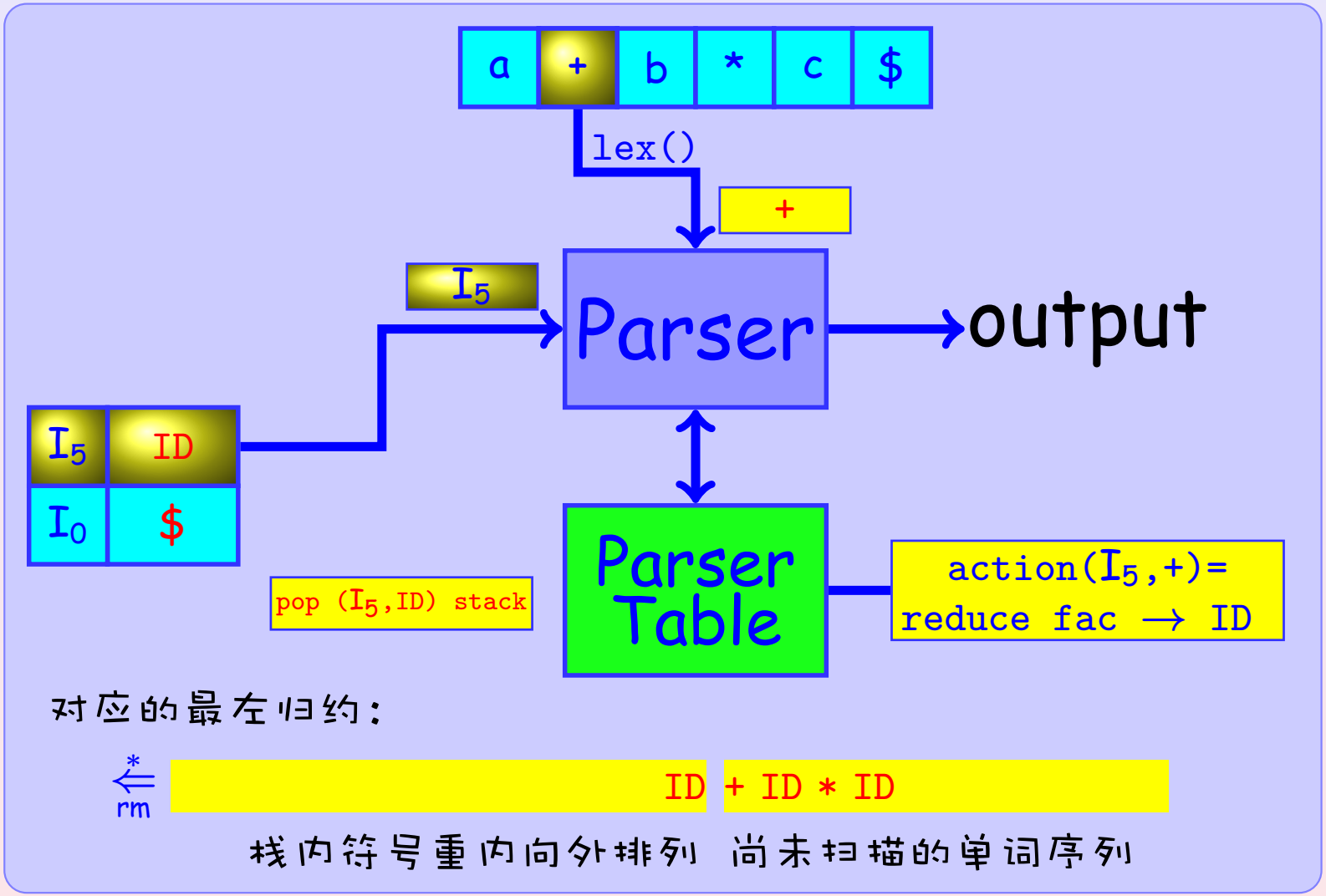


移进归约

LR 分析法动态演示



LR 分析法动态演示

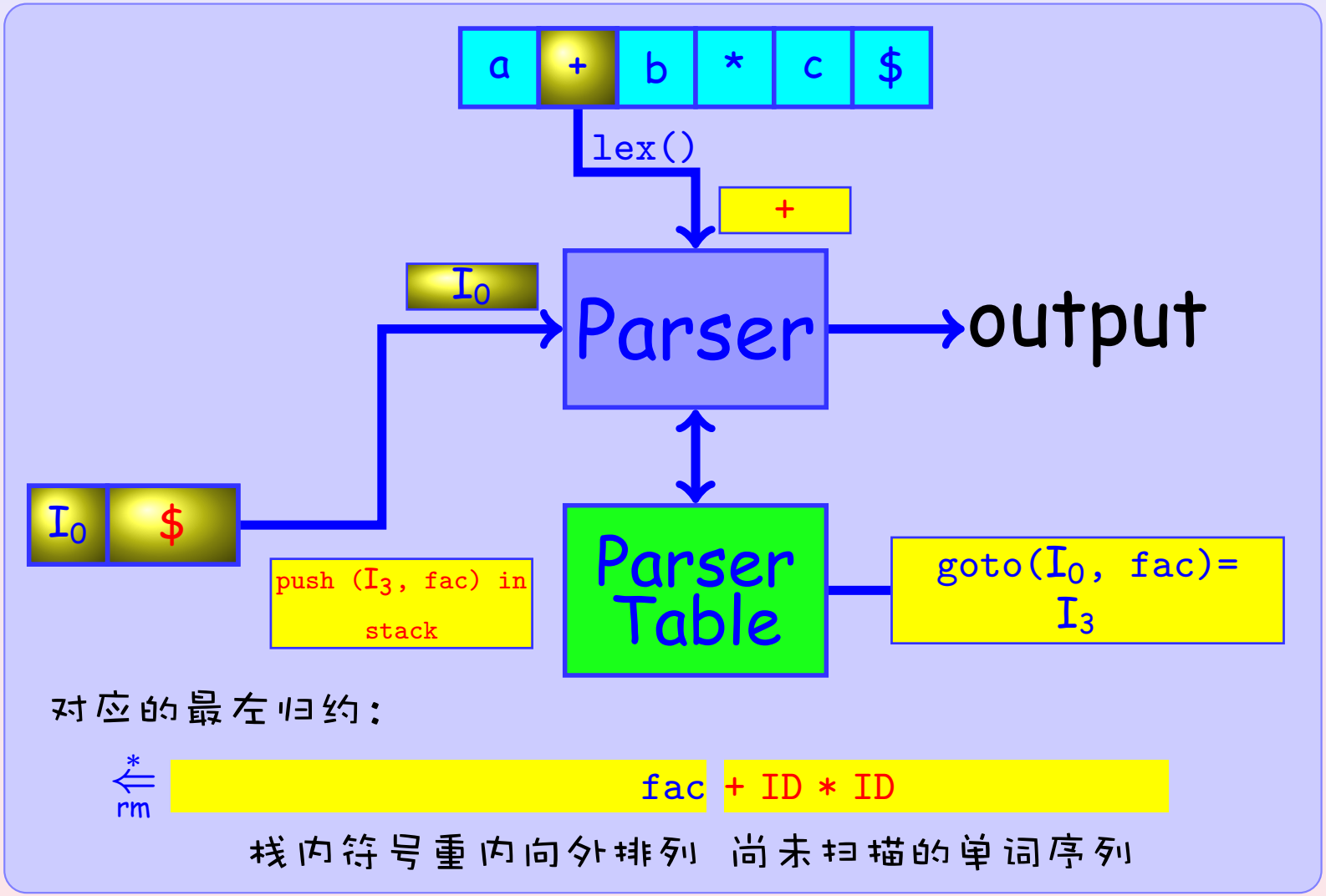


对应的最左归约：

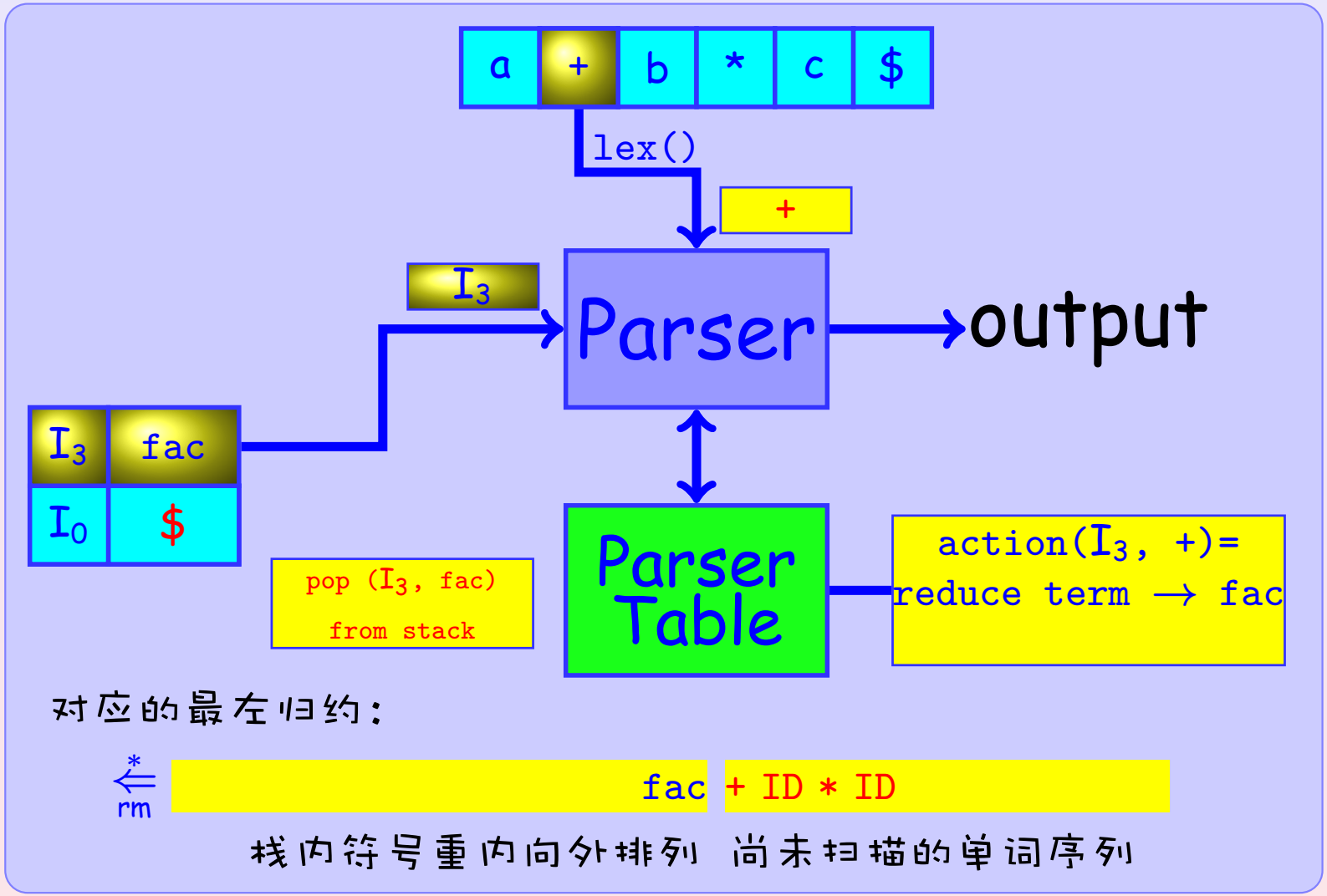
$\overset{*}{\Leftarrow}_{rm}$ ID + ID * ID

栈内符号重内向外排列　尚未扫描的单词序列

LR 分析法动态演示

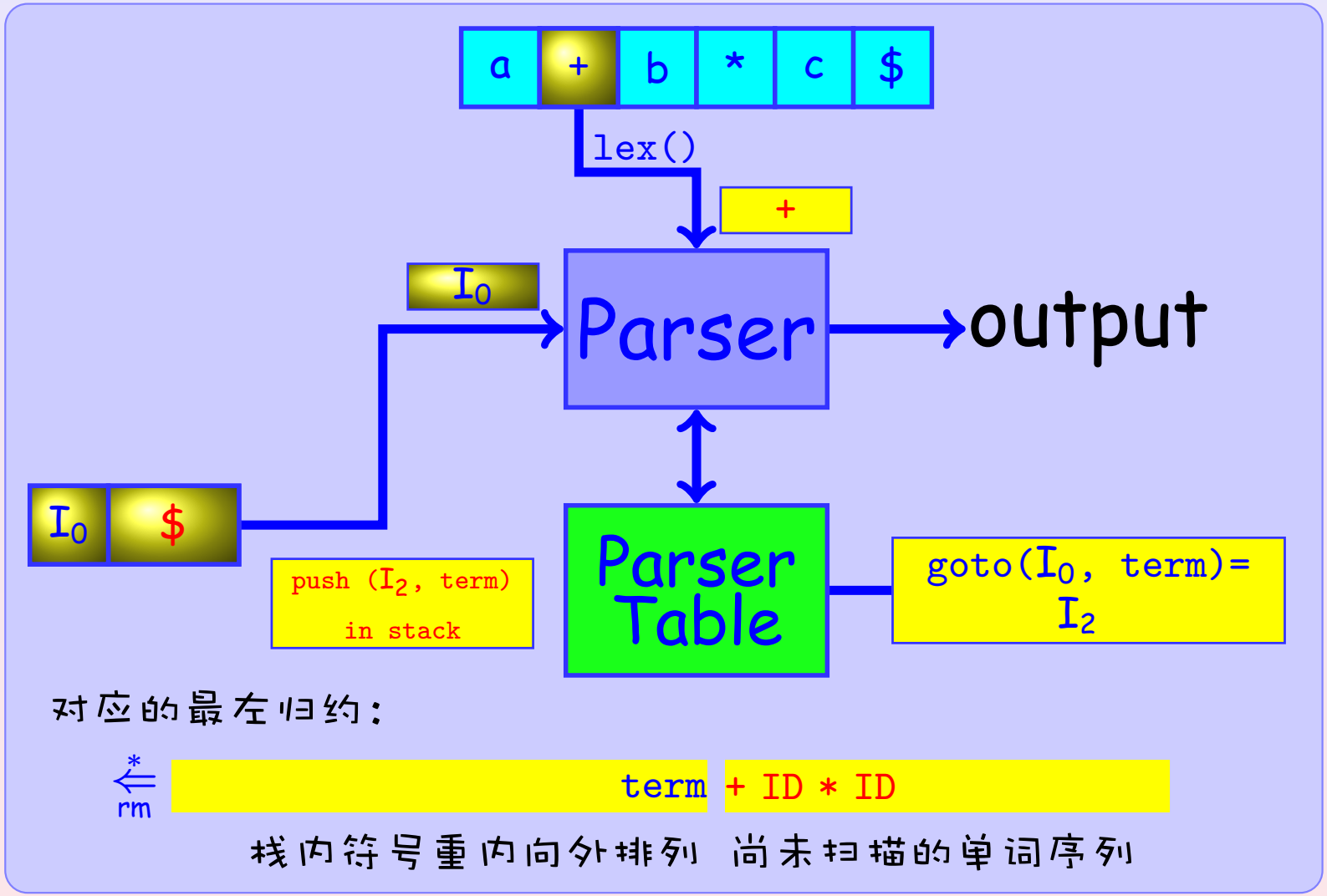


LR 分析法动态演示



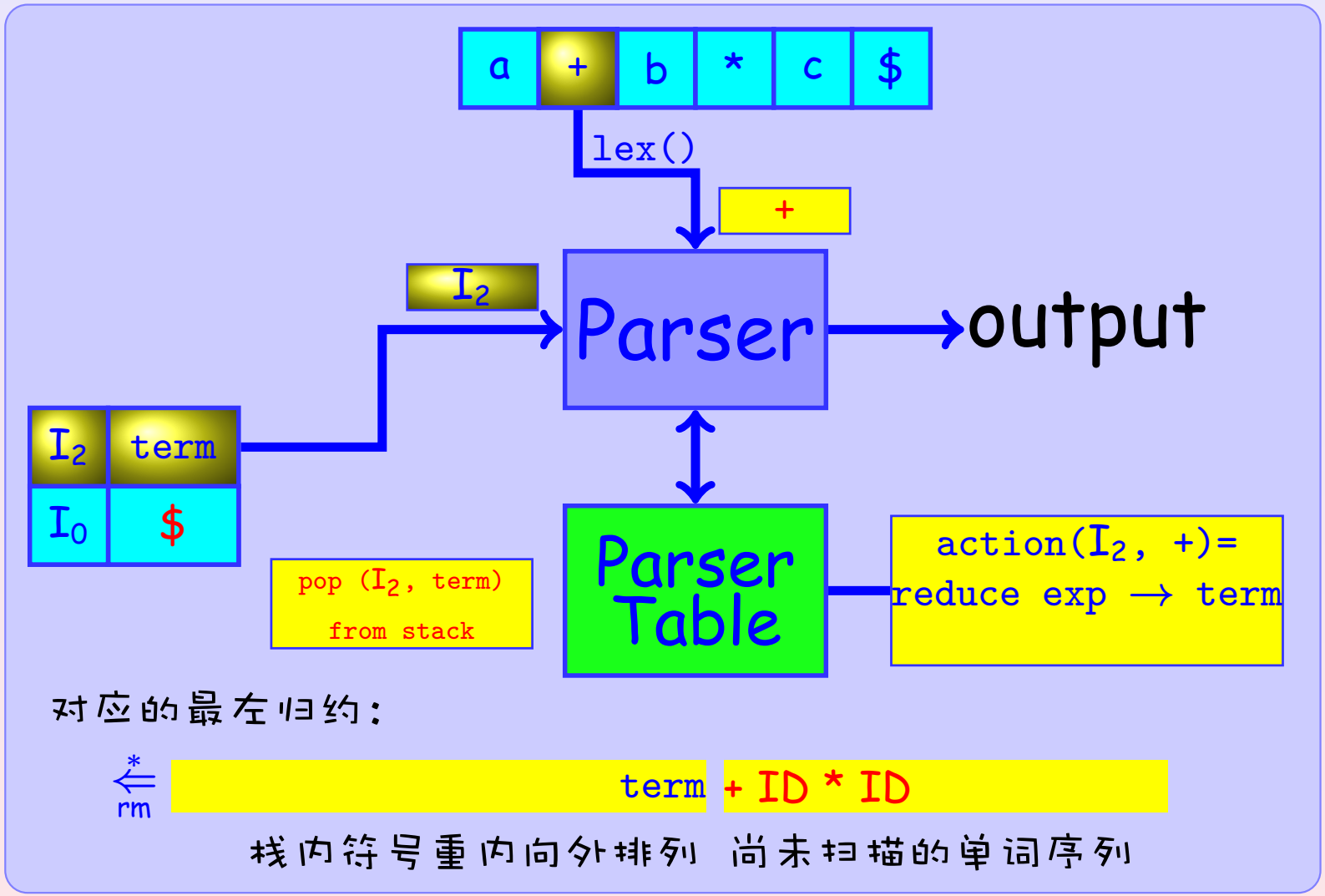
移进归约

LR 分析法动态演示

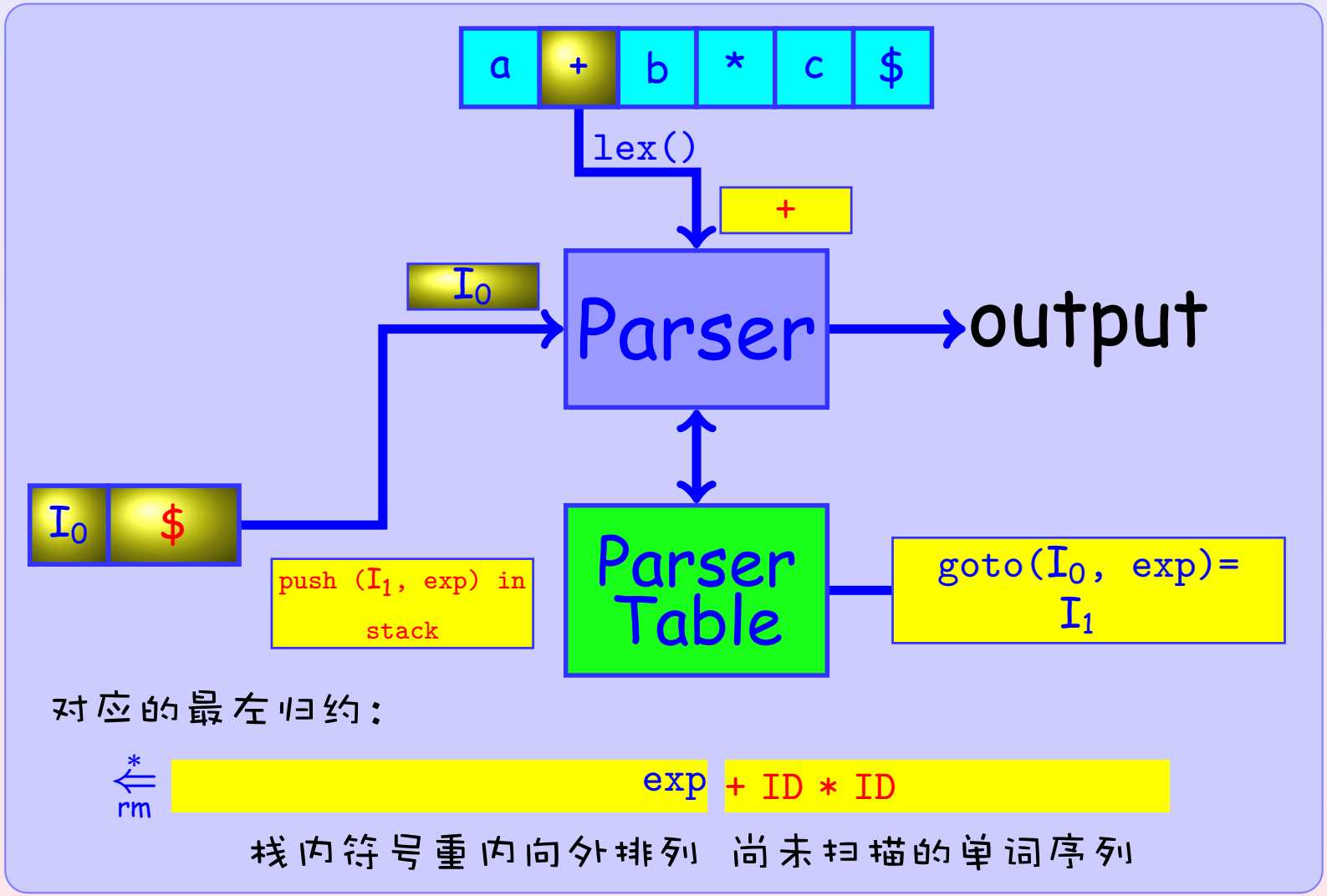


移进归约

LR 分析法动态演示

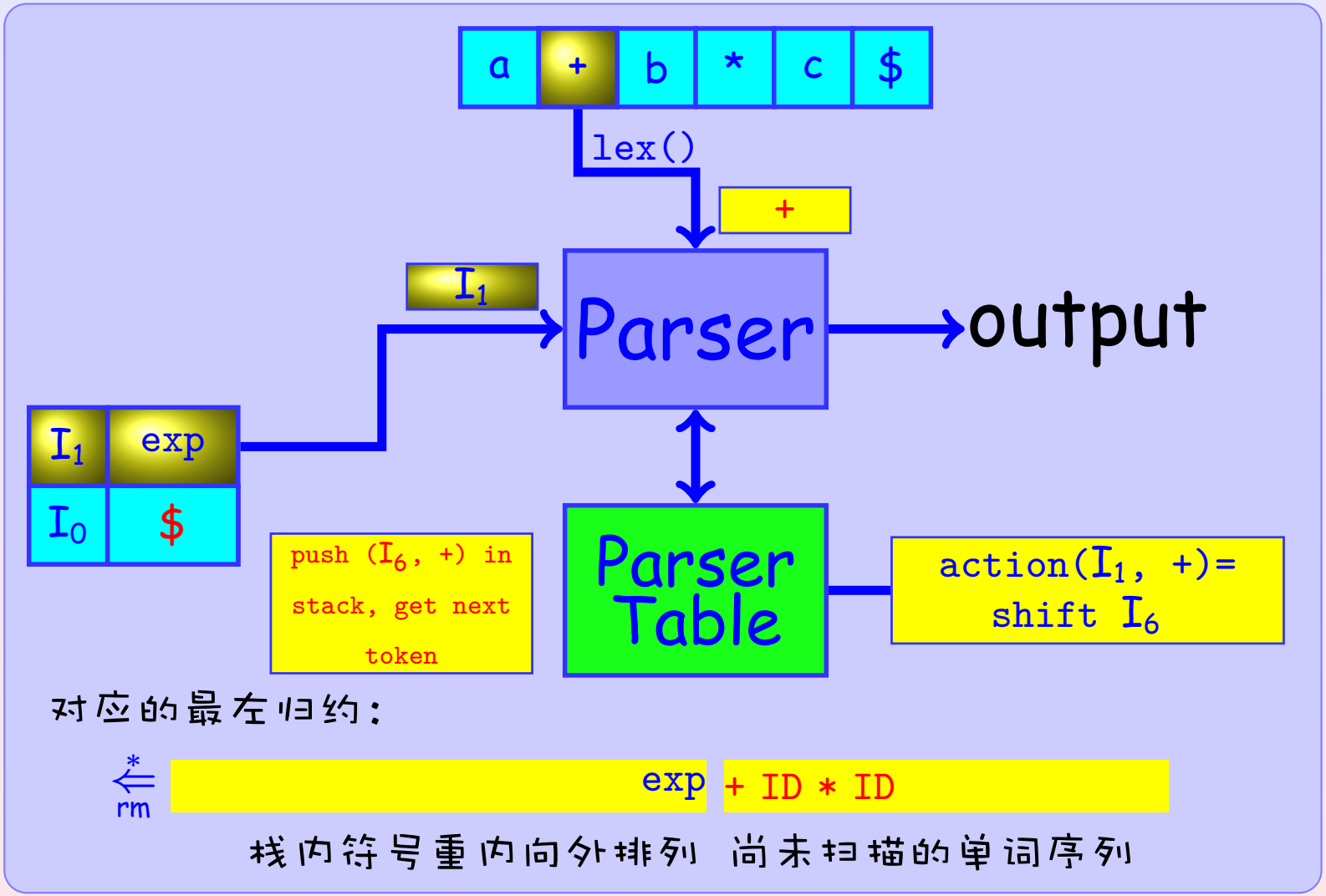


LR 分析法动态演示

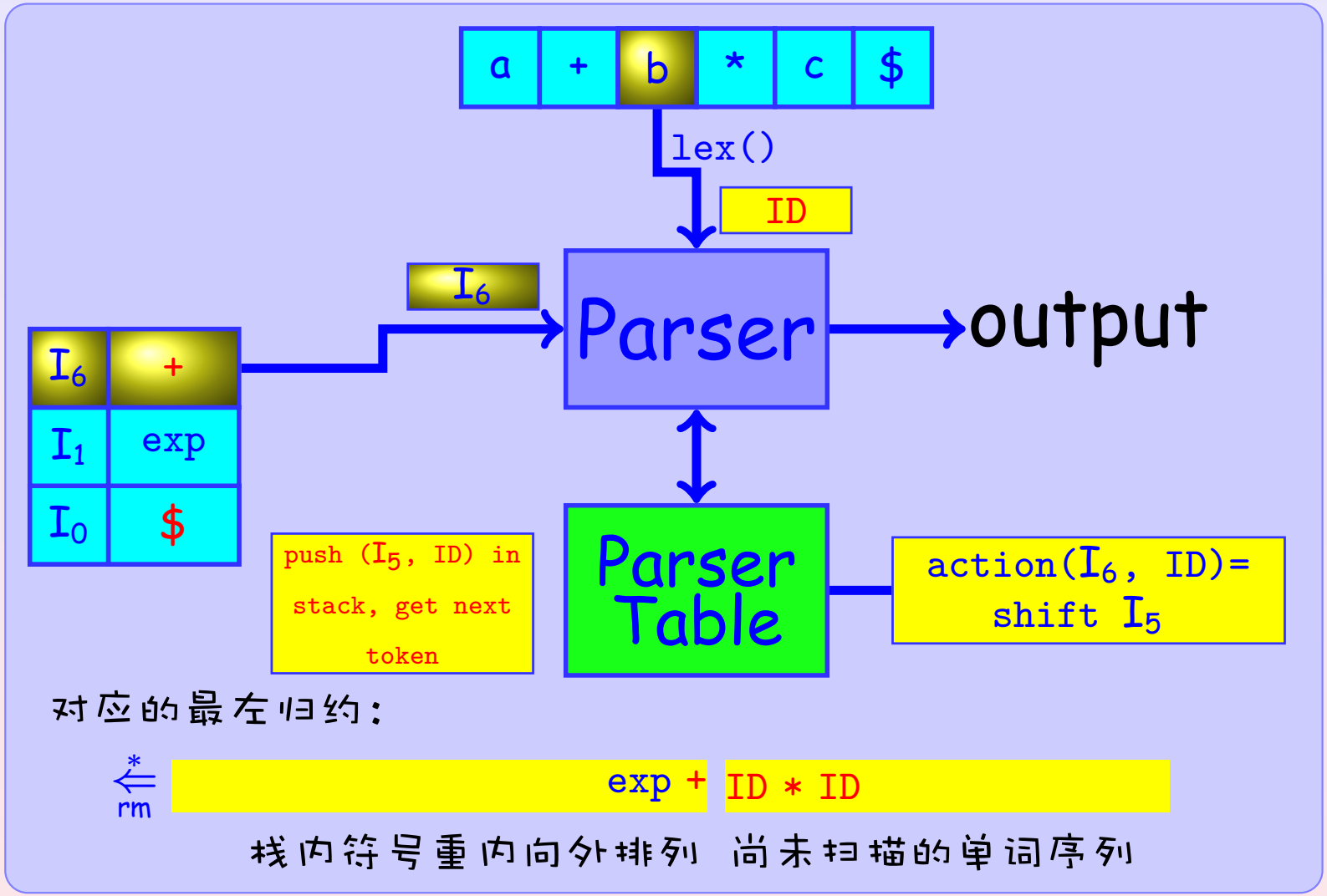


移进归约

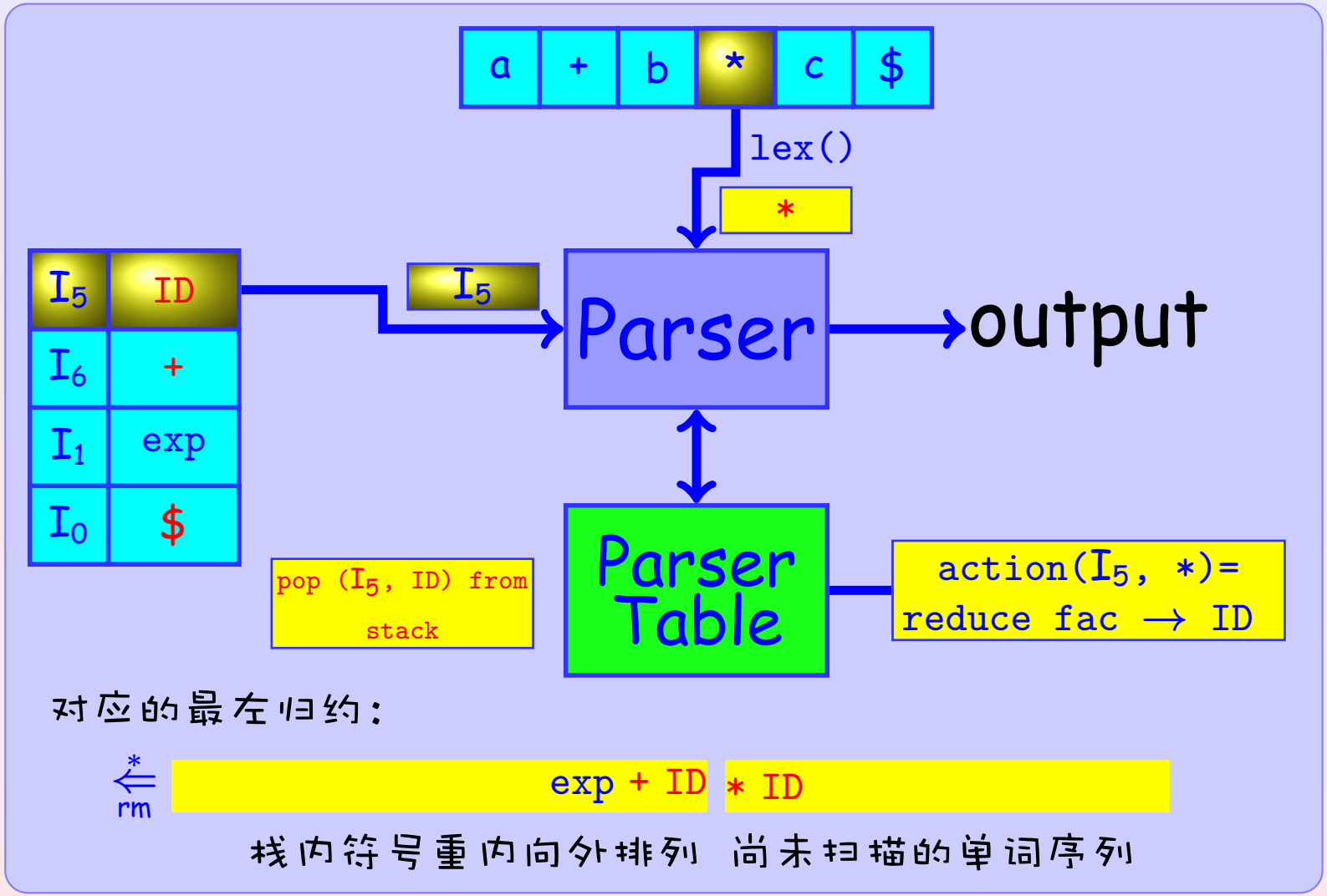
LR 分析法动态演示



LR 分析法动态演示



LR 分析法动态演示

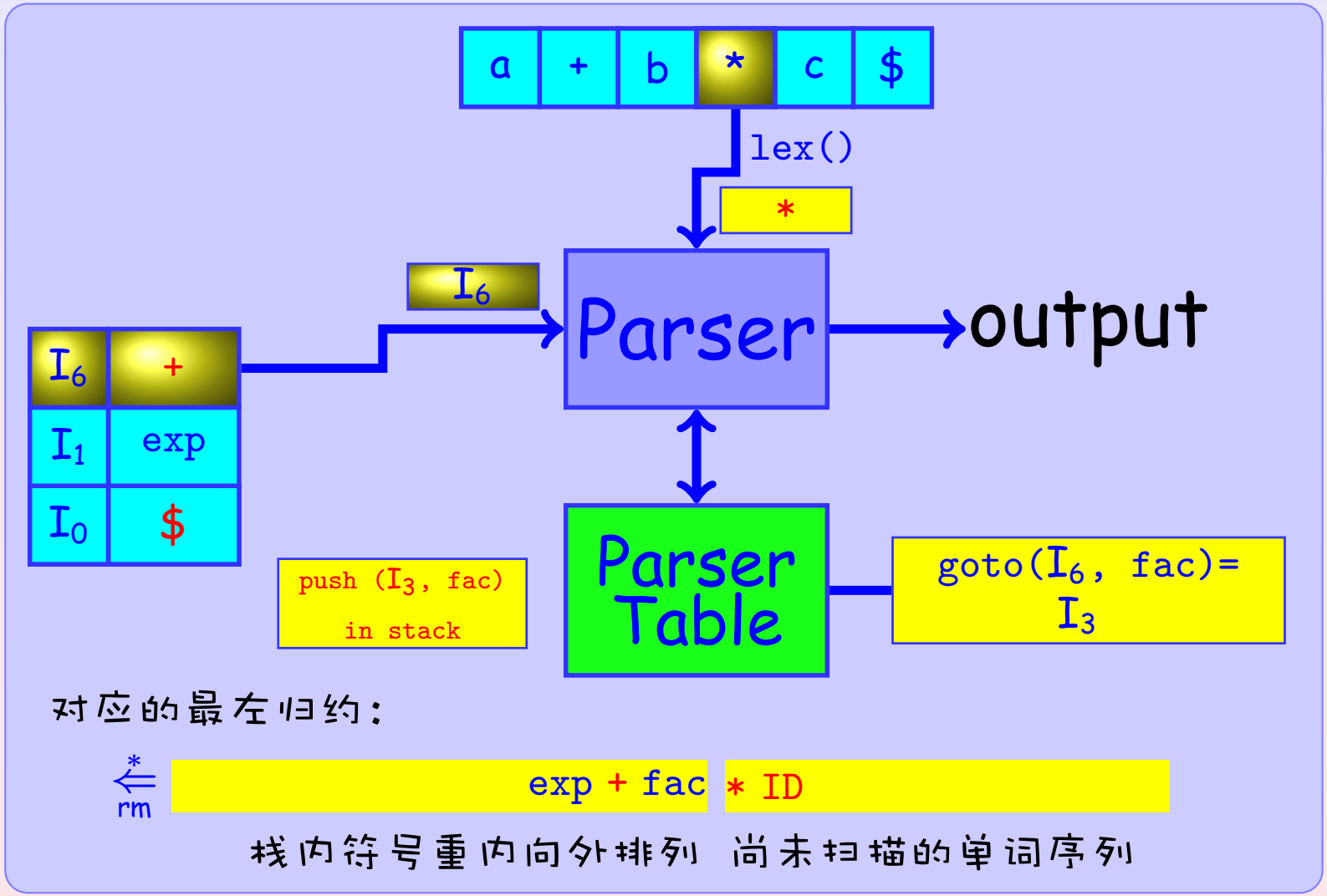


对应的最左归约：

$\overset{*}{\underset{rm}{\Leftarrow}}$ exp + ID * ID

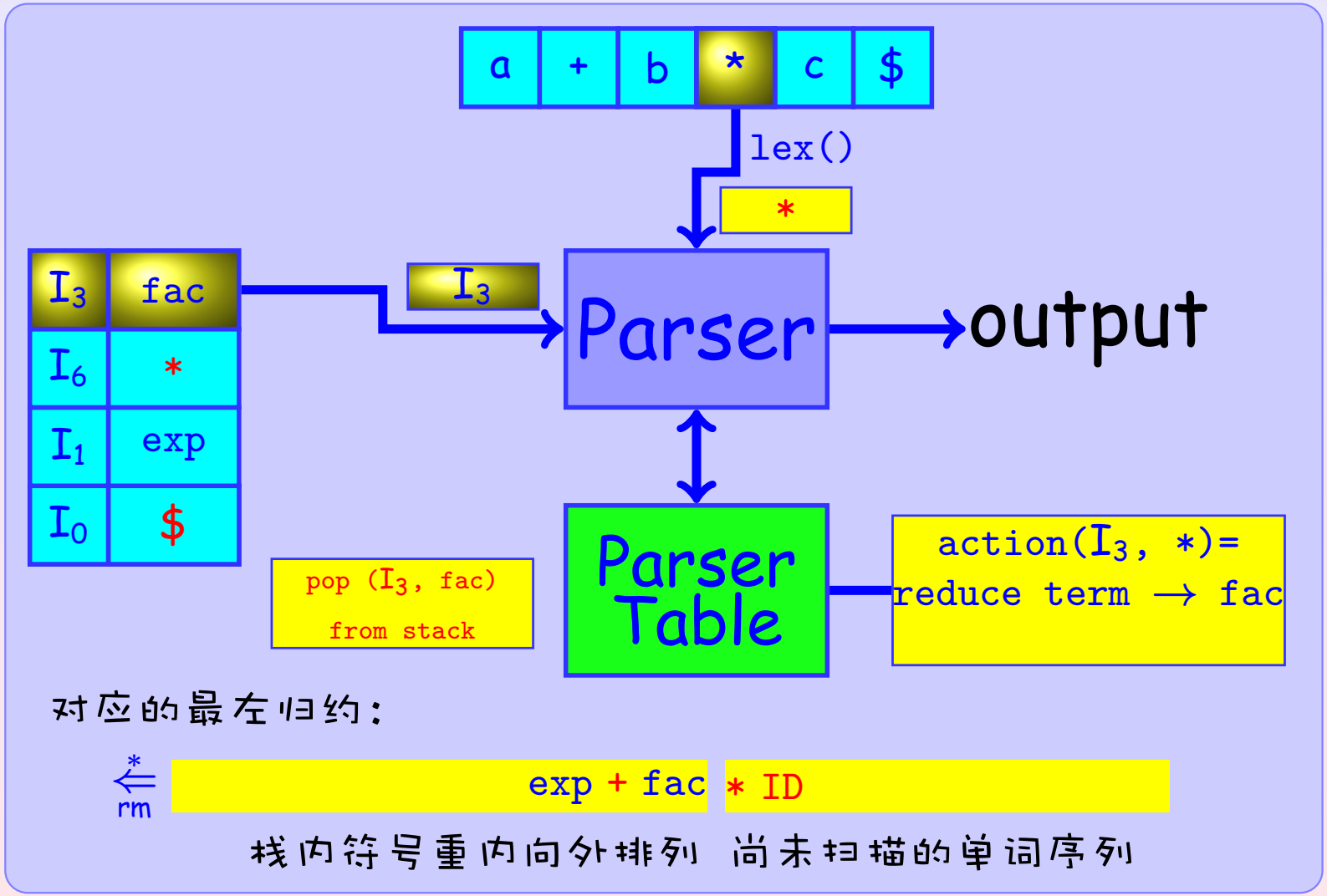
栈内符号重内向外排列　尚未扫描的单词序列

LR 分析法动态演示



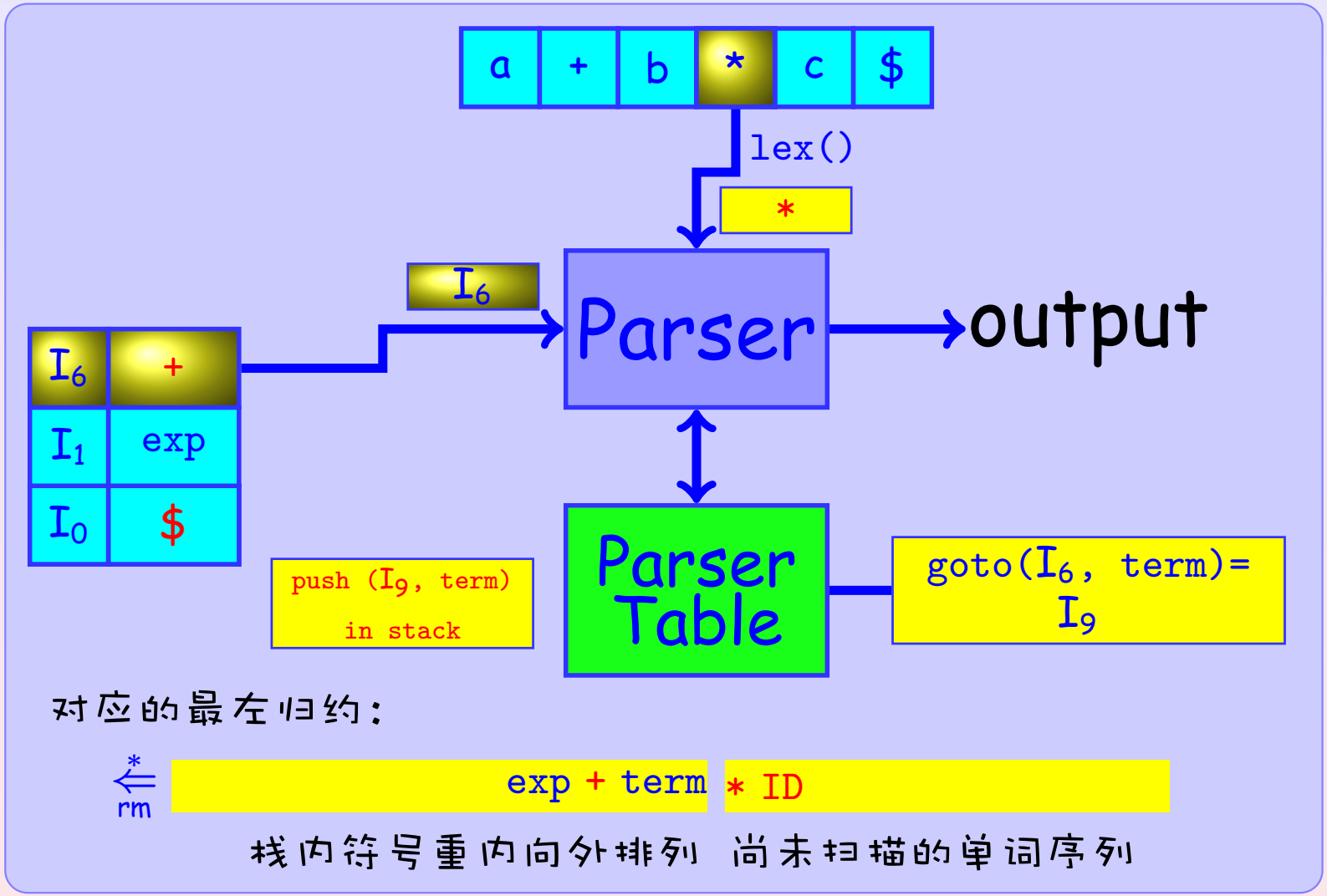
移进归约

LR 分析法动态演示



移进归约

LR 分析法动态演示

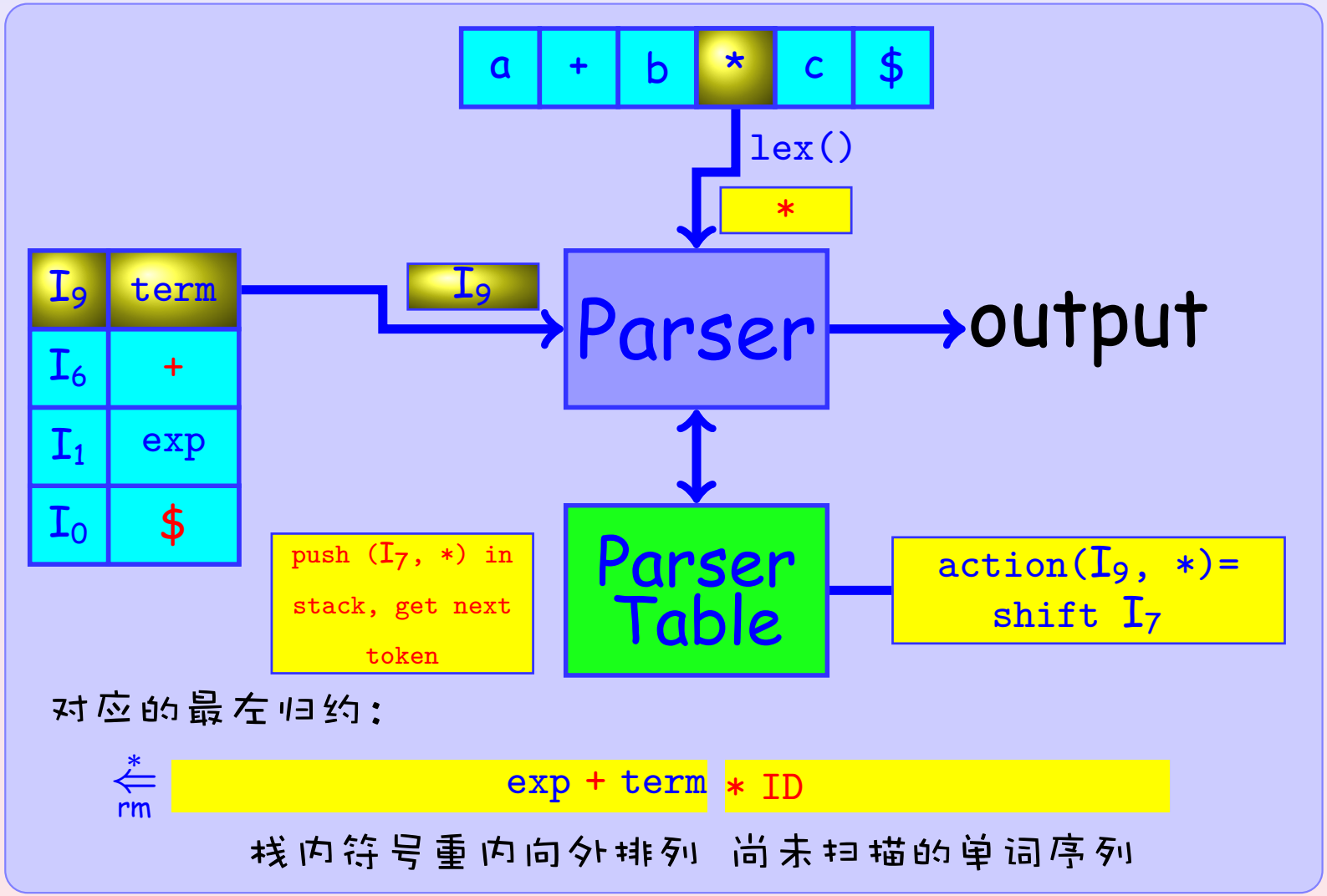


对应的最左归约：

$\overset{*}{\underset{rm}{\Leftarrow}}$ exp + term * ID

栈内符号重内向外排列　尚未扫描的单词序列

LR 分析法动态演示

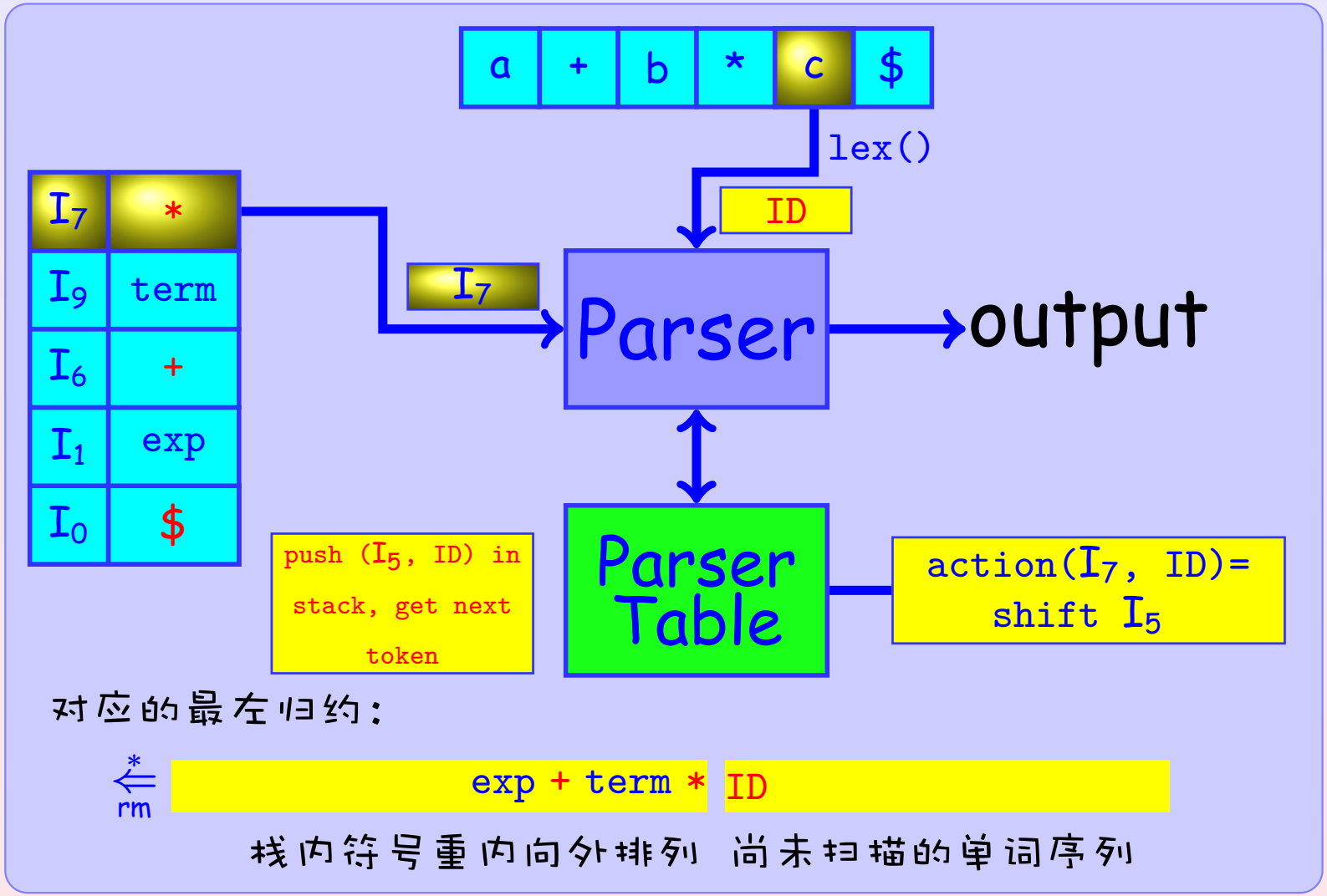


对应的最左归约：

$\overset{*}{\underset{rm}{\Leftarrow}}$ exp + term * ID

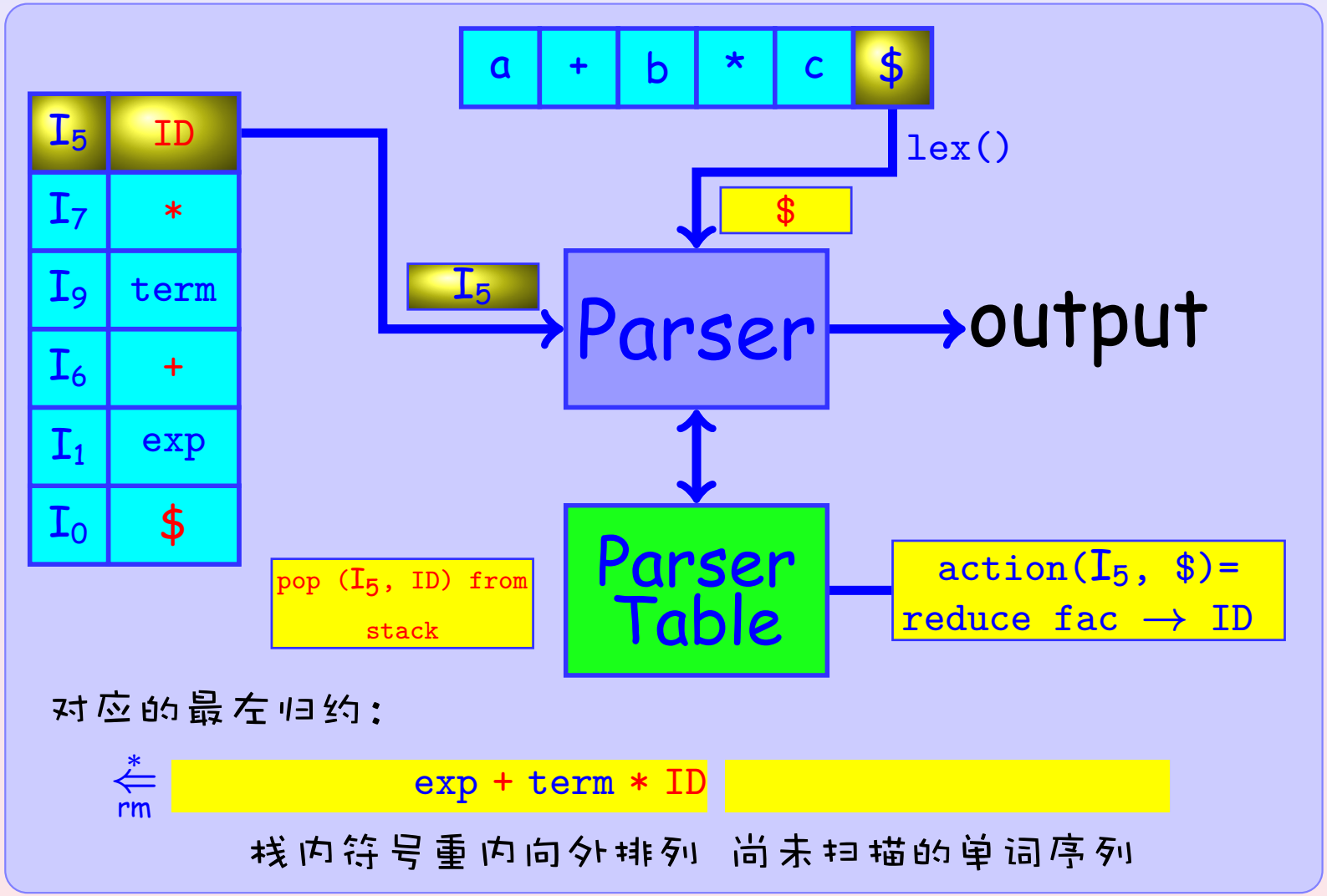
栈内符号重内向外排列　尚未扫描的单词序列

LR 分析法动态演示



移进归约

LR 分析法动态演示

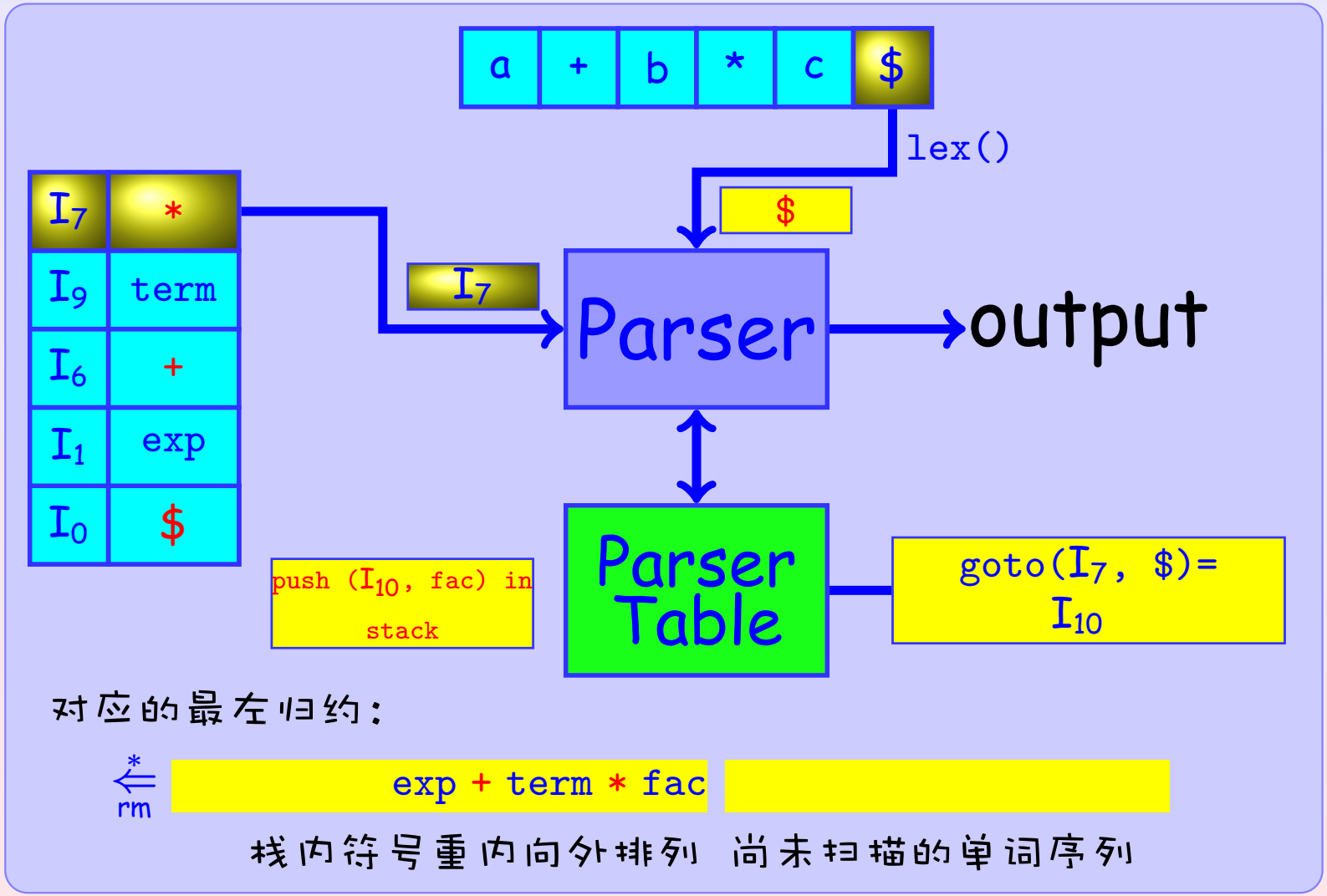


对应的最左归约：

$\overset{*}{\underset{rm}{\Leftarrow}}$ exp + term * ID

栈内符号重内向外排列　尚未扫描的单词序列

LR 分析法动态演示



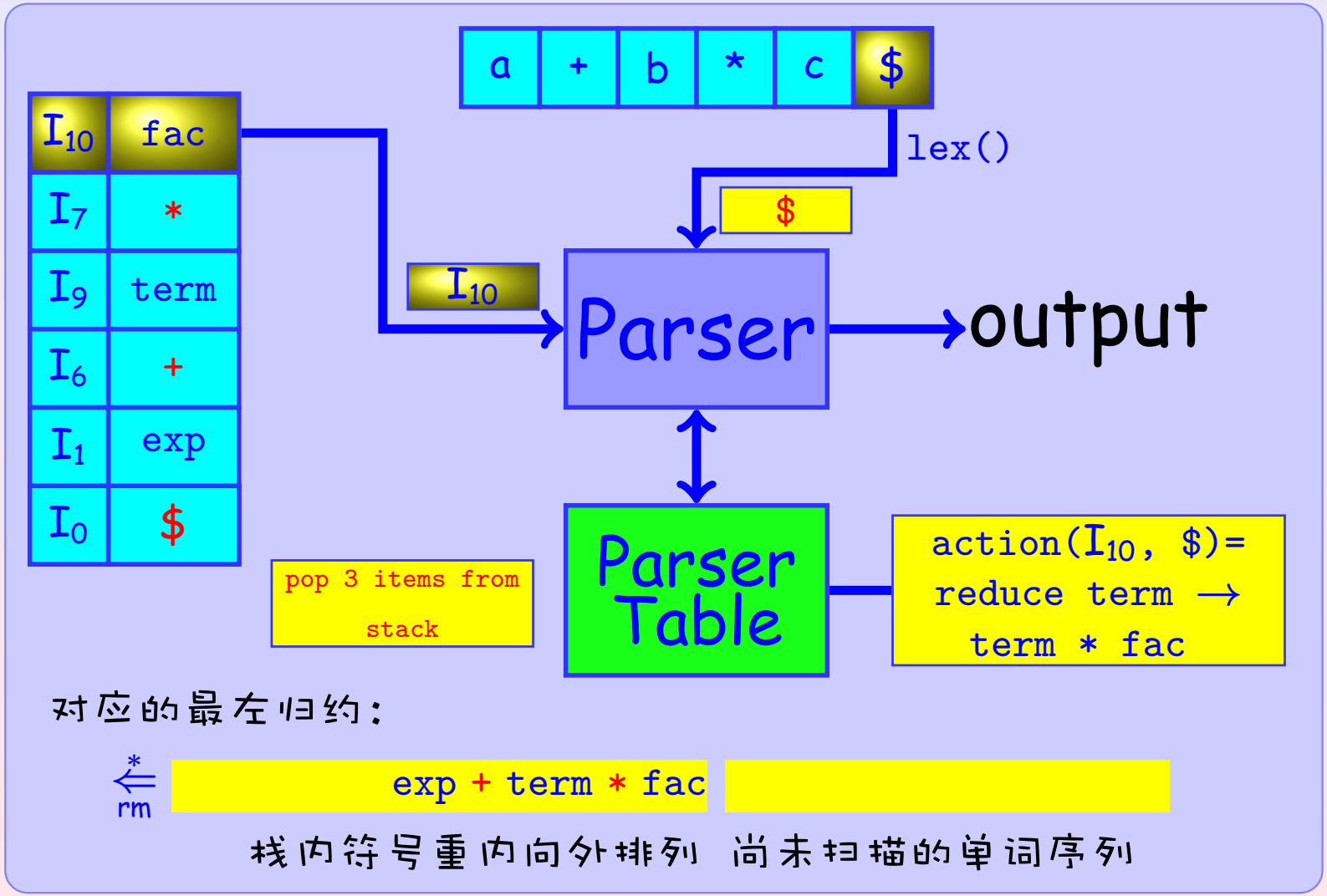
对应的最左归约：

$\overset{*}{\underset{rm}{\Leftarrow}}$ exp + term * fac

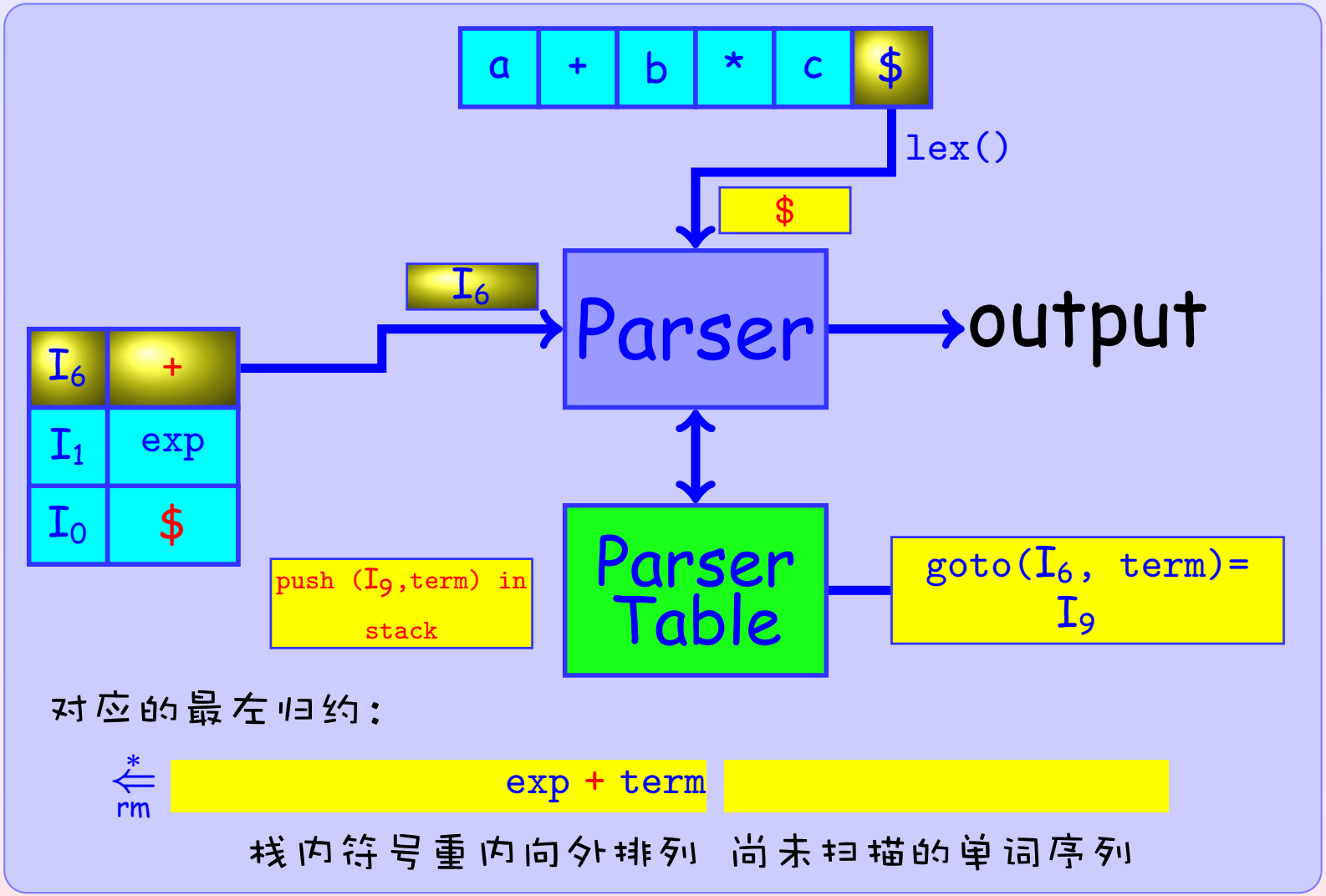
栈内符号重内向外排列　尚未扫描的单词序列

移进归约

LR 分析法动态演示

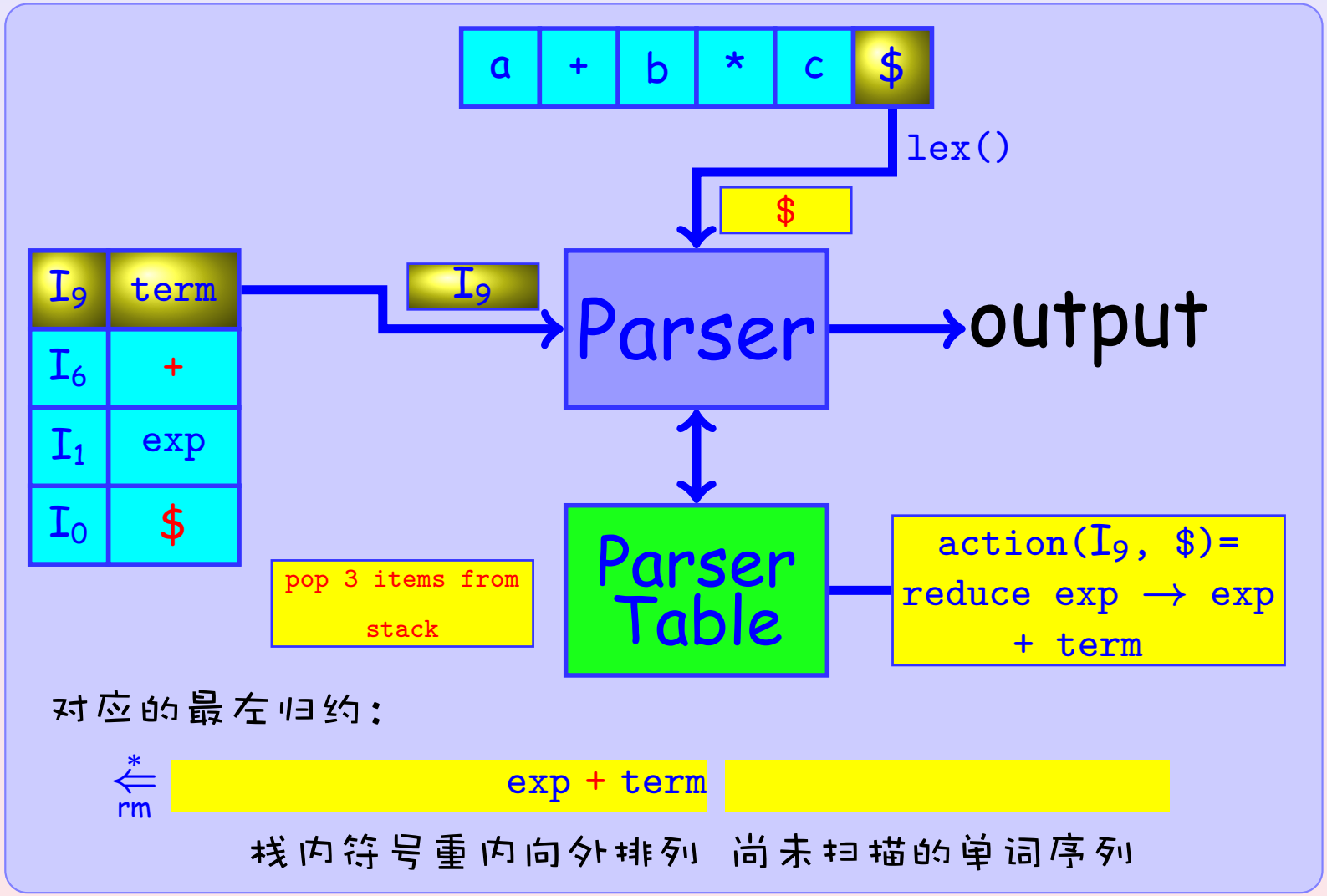


LR 分析法动态演示

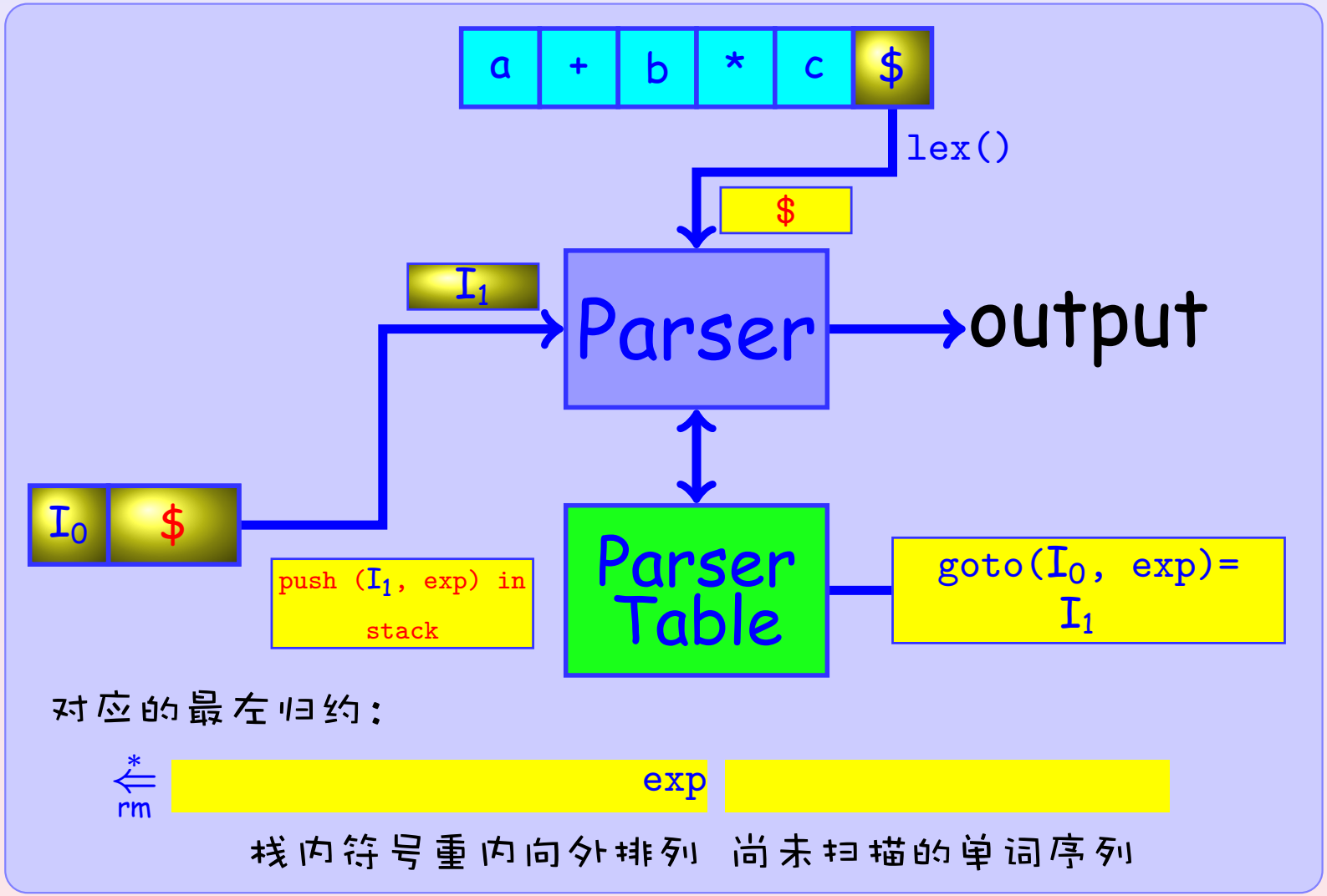


移进归约

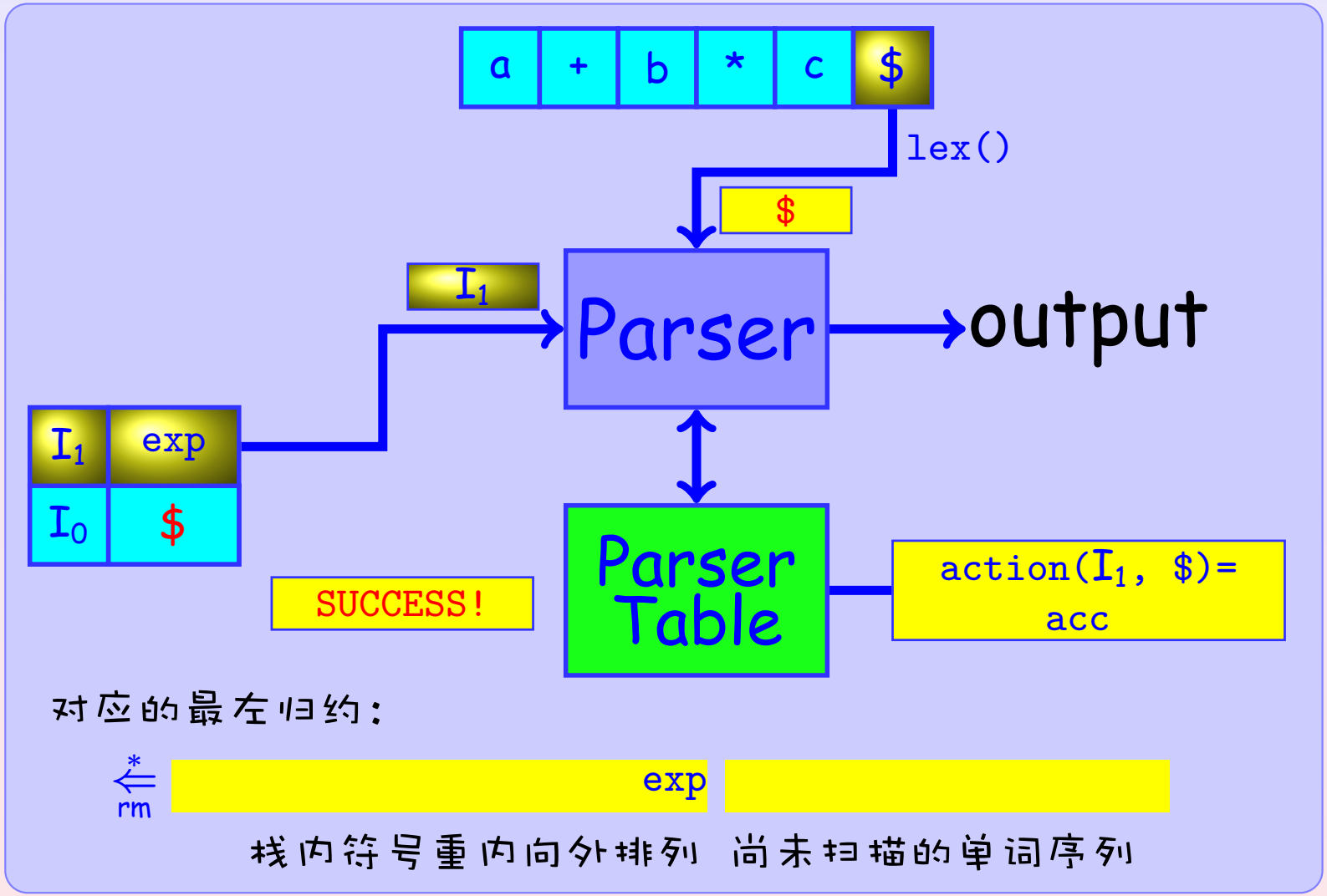
LR 分析法动态演示



LR 分析法动态演示



LR 分析法动态演示



XL 语言的 SLR 分析表

XL 语言文法

1/ exp → exp+term　2/ exp → term　3/ term → term* fac
4/ term → fac　5/ fac →ID　6/ fac → (exp)

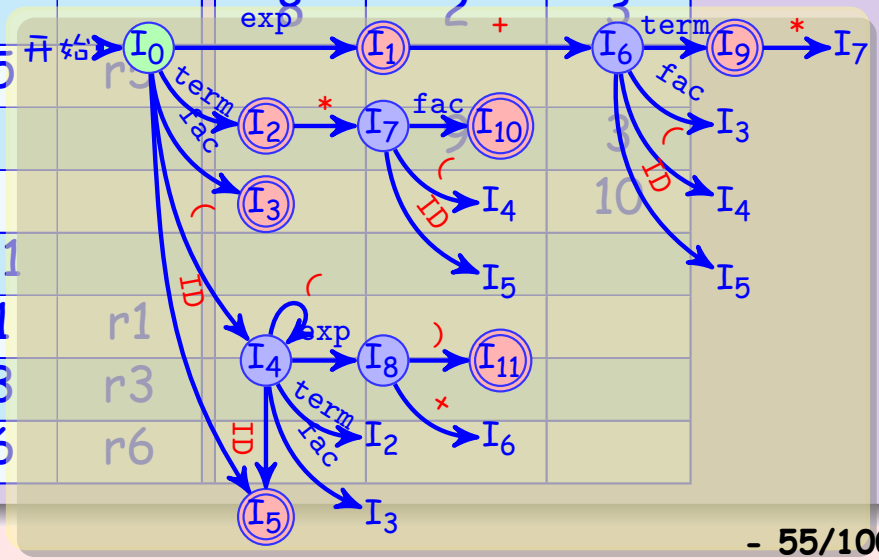
	action						goto		
状态	ID	+	*	(	)	$	exp	term	fac
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r5	r5		r5	r5			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r6	r6		r6	r6			

XL 语言的 SLR 分析表

XL 语言文法

1/ exp → exp+term　　2/ exp → term　　3/ term → term* fac
4/ term → fac　　5/ fac →ID　　6/ fac → (exp)

	action						goto		
状态	ID	+	*	(	)	$	exp	term	fac
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r5	r5		r5	r5			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r6	r6		r6	r6			

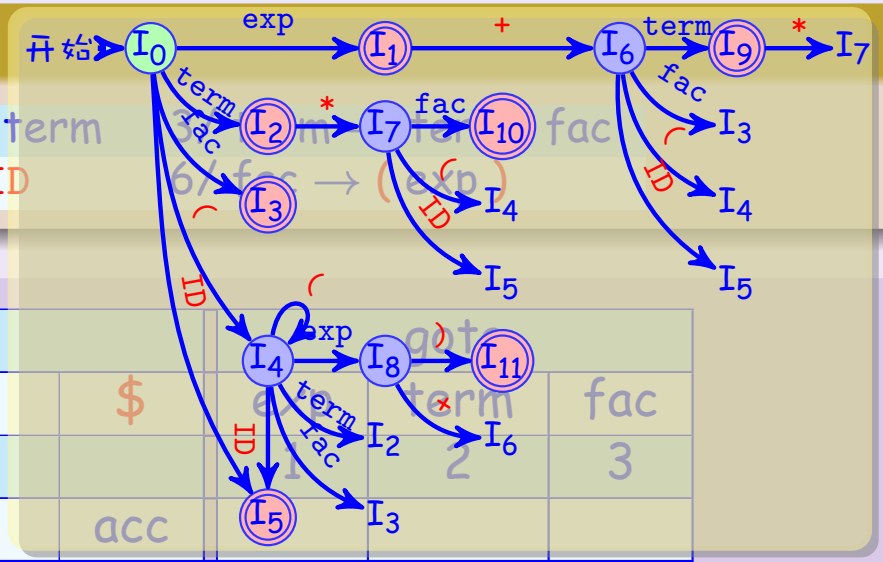


XL 语言的 SLR 分析表

XL 语言文法

1/ exp → exp+term　　2/ exp → term

4/ term → fac　　5/ fac → ID



	action						goto		
状态	ID	+	*	(	)	$	exp	term	fac
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r5	r5		r5	r5			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r6	r6		r6	r6			

分析过程

stack	symbols	input	action
0		ID+ID*ID$	shift
05	ID	+ID*ID$	reduce by fac→ID
03	fac	+ID*ID$	reduce by term→fac
02	term	+ID*ID$	reduce by exp→term
01	exp	+ID*ID$	shift
016	exp+	ID*ID$	shift
0165	exp+ID	*ID$	reduce by fac→ID
0163	exp+fac	*ID$	reduce by term→fac
0169	exp+term	*ID$	shift
01697	exp+term*	ID$	shift
016975	exp+term*ID	$	reduce by fac→ID
0169710	exp+term*fac	$	reduce by term→term*fac
0169	exp+term	$	reduce by exp→exp*term
01	exp	$	accept

分析过程

stack	symbols	input	action
0		ID+ID*ID$	shift
05	ID	+ID*ID$	reduce by fac→ID
03	fac	+ID*ID$	reduce by term→fac
02	term	+ID*ID$	reduce by exp→term
01	exp	+ID*ID$	shift
016	exp+	ID*ID$	shift
0165	exp+ID	*ID$	reduce by fac→ID
0163			
0169			
01697			
01697			
01697			m*fac
0169			term
01			

	action						goto		
状态	ID	+	*	(	)	$	exp	term	fac
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r5	r5		r5	r5			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r6	r6		r6	r6			

分析过程

stack	symbols	input	action
0		ID+ID*ID$	shift
05	ID	+ID*ID$	reduce by fac→ID
03	fac	+ID*ID$	reduce by term→fac
02	term	+ID*ID$	reduce by exp→term
01	exp	+ID*ID$	shift
016	exp+	ID*ID$	shift
0165	exp+ID	*ID$	reduce by fac→ID
0163			
0169			
01697			
01697			
01697			m*fac
0169			term
01			

	action						goto		
状态	ID	+	*	(	)	$	exp	term	fac
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r5	r5		r5	r5			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r6	r6		r6	r6			

分析过程

stack	symbols	input	action
0		ID+ID*ID$	shift
05	ID	+ID*ID$	reduce by fac→ID
03	fac	+ID*ID$	reduce by term→fac
02	term	+ID*ID$	reduce by exp→term
01	exp	+ID*ID$	shift
016	exp+	ID*ID$	shift
0165	exp+ID	*ID$	reduce by fac→ID
0163			
0169			
01697			
01697			
01697			m*fac
0169			term
01			

	action						goto		
状态	ID	+	*	(	)	$	exp	term	fac
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r5	r5		r5	r5			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r6	r6		r6	r6			

分析过程

stack	symbols	input	action
0		ID+ID*ID$	shift
05	ID	+ID*ID$	reduce by fac→ID
03	fac	+ID*ID$	reduce by term→fac
02	term	+ID*ID$	reduce by exp→term
01	exp	+ID*ID$	shift
016	exp+	ID*ID$	shift
0165	exp+ID	*ID$	reduce by fac→ID
0163			
0169			
01697			
01697			
01697			
0169			
01			

	action						goto		
状态	ID	+	*	(	)	$	exp	term	fac
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r5	r5		r5	r5			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r6	r6		r6	r6			

分析过程

stack	symbols	input	action
0		ID+ID*ID$	shift
05	ID	+ID*ID$	reduce by fac→ID
03	fac	+ID*ID$	reduce by term→fac
02	term	+ID*ID$	reduce by exp→term
01	exp	+ID*ID$	shift
016	exp+	ID*ID$	shift
0165	exp+ID	*ID$	reduce by fac→ID
0163			
0169			
01697			
01697			
01697			
0169			
01			

	action						goto		
状态	ID	+	*	(	)	$	exp	term	fac
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r5	r5		r5	r5			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r6	r6		r6	r6			

分析过程

stack	symbols	input	action
0		ID+ID*ID$	shift
05	ID	+ID*ID$	reduce by fac→ID
03	fac	+ID*ID$	reduce by term→fac
02	term	+ID*ID$	reduce by exp→term
01	exp	+ID*ID$	shift
016	exp+	ID*ID$	shift
0165	exp+ID	*ID$	reduce by fac→ID
0163			
0169			
01697			
01697			
01697			
0169			
01			

	action						goto		
状态	ID	+	*	(	)	$	exp	term	fac
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r5	r5		r5	r5			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r6	r6		r6	r6			

分析过程

stack	symbols	input	action
0		ID+ID*ID$	shift
05	ID	+ID*ID$	reduce by fac→ID
03	fac	+ID*ID$	reduce by term→fac
02	term	+ID*ID$	reduce by exp→term
01	exp	+ID*ID$	shift
016	exp+	ID*ID$	shift
0165	exp+ID	*ID$	reduce by fac→ID

	action						goto		
状态	ID	+	*	(	)	$	exp	term	fac
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r5	r5		r5	r5			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r6	r6		r6	r6			

分析过程

	action						goto		
状态	ID	+	*	(	)	$	exp	term	fac
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r5	r5		r5	r5			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r6	r6		r6	r6			

stack			
0			
05			
03			
02			
01			
016			
0165			
0163	exp+fac	*ID$	reduce by term→fac
0169	exp+term	*ID$	shift
01697	exp+term*	ID$	shift
016975	exp+term*ID	$	reduce by fac→ID
0169710	exp+term*fac	$	reduce by term→term*fac
0169	exp+term	$	reduce by exp→exp*term
01	exp	$	accept

分析过程

	action						goto		
状态	ID	+	*	(	)	$	exp	term	fac
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r5	r5		r5	r5			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r6	r6		r6	r6			

stack			
0			
05			
03			
02			
01			
016			
0165			
0163	exp+fac	*ID$	reduce by term→fac
0169	exp+term	*ID$	shift
01697	exp+term*	ID$	shift
016975	exp+term*ID	$	reduce by fac→ID
0169710	exp+term*fac	$	reduce by term→term*fac
0169	exp+term	$	reduce by exp→exp*term
01	exp	$	accept

分析过程

	action						goto		
状态	ID	+	*	(	)	$	exp	term	fac
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r5	r5		r5	r5			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r6	r6		r6	r6			

stack			
0			
05			
03			
02			
01			
016			
0165			
0163	exp+fac	*ID$	reduce by term→fac
0169	exp+term	*ID$	shift
01697	exp+term*	ID$	shift
016975	exp+term*ID	$	reduce by fac→ID
0169710	exp+term*fac	$	reduce by term→term*fac
0169	exp+term	$	reduce by exp→exp*term
01	exp	$	accept

分析过程

	action						goto		
状态	ID	+	*	(	)	$	exp	term	fac
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r5	r5		r5	r5			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r6	r6		r6	r6			

stack			
0			
05			
03			
02			
01			
016			
0165			
0163	exp+fac	*ID$	reduce by term→fac
0169	exp+term	*ID$	shift
01697	exp+term*	ID$	shift
016975	exp+term*ID	$	reduce by fac→ID
0169710	exp+term*fac	$	reduce by term→term*fac
0169	exp+term	$	reduce by exp→exp*term
01	exp	$	accept

分析过程

状态	action						goto		
	ID	+	*	(	)	$	exp	term	fac
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r5	r5		r5	r5			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r6	r6		r6	r6			

stack			
0			
05			
03			
02			
01			
016			
0165			
0163	exp+fac	*ID$	reduce by term→fac
0169	exp+term	*ID$	shift
01697	exp+term*	ID$	shift
016975	exp+term*ID	$	reduce by fac→ID
0169710	exp+term*fac	$	reduce by term→term*fac
0169	exp+term	$	reduce by exp→exp*term
01	exp	$	accept

分析过程

状态	action						goto		
	ID	+	*	(	)	$	exp	term	fac
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r5	r5		r5	r5			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r6	r6		r6	r6			

stack			
0			
05			
03			
02			
01			
016			
0165			
0163	exp+fac	*ID$	reduce by term→fac
0169	exp+term	*ID$	shift
01697	exp+term*	ID$	shift
016975	exp+term*ID	$	reduce by fac→ID
0169710	exp+term*fac	$	reduce by term→term*fac
0169	exp+term	$	reduce by exp→exp*term
01	exp	$	accept

分析过程

状态	action						goto		
	ID	+	*	(	)	$	exp	term	fac
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r5	r5		r5	r5			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r6	r6		r6	r6			

stack			
0			
05			
03			
02			
01			
016			
0165			
0163	exp+fac	*ID$	reduce by term→fac
0169	exp+term	*ID$	shift
01697	exp+term*	ID$	shift
016975	exp+term*ID	$	reduce by fac→ID
0169710	exp+term*fac	$	reduce by term→term*fac
0169	exp+term	$	reduce by exp→exp*term
01	exp	$	accept

分析过程

stack	symbols	input	action
0		ID+ID*ID$	shift
05	ID	+ID*ID$	reduce by fac→ID
03	fac	+ID*ID$	reduce by term→fac
02	term	+ID*ID$	reduce by exp→term
01	exp	+ID*ID$	shift
016	exp+	ID*ID$	shift
0165	exp+ID	*ID$	reduce by fac→ID
0163	exp+fac	*ID$	reduce by term→fac
0169	exp+term	*ID$	shift
01697	exp+term*	ID$	shift
016975	exp+term*ID	$	reduce by fac→ID
0169710	exp+term*fac	$	reduce by term→term*fac
0169	exp+term	$	reduce by exp→exp*term
01	exp	$	accept

page-break

LALR(LookAhead LR) 文法.

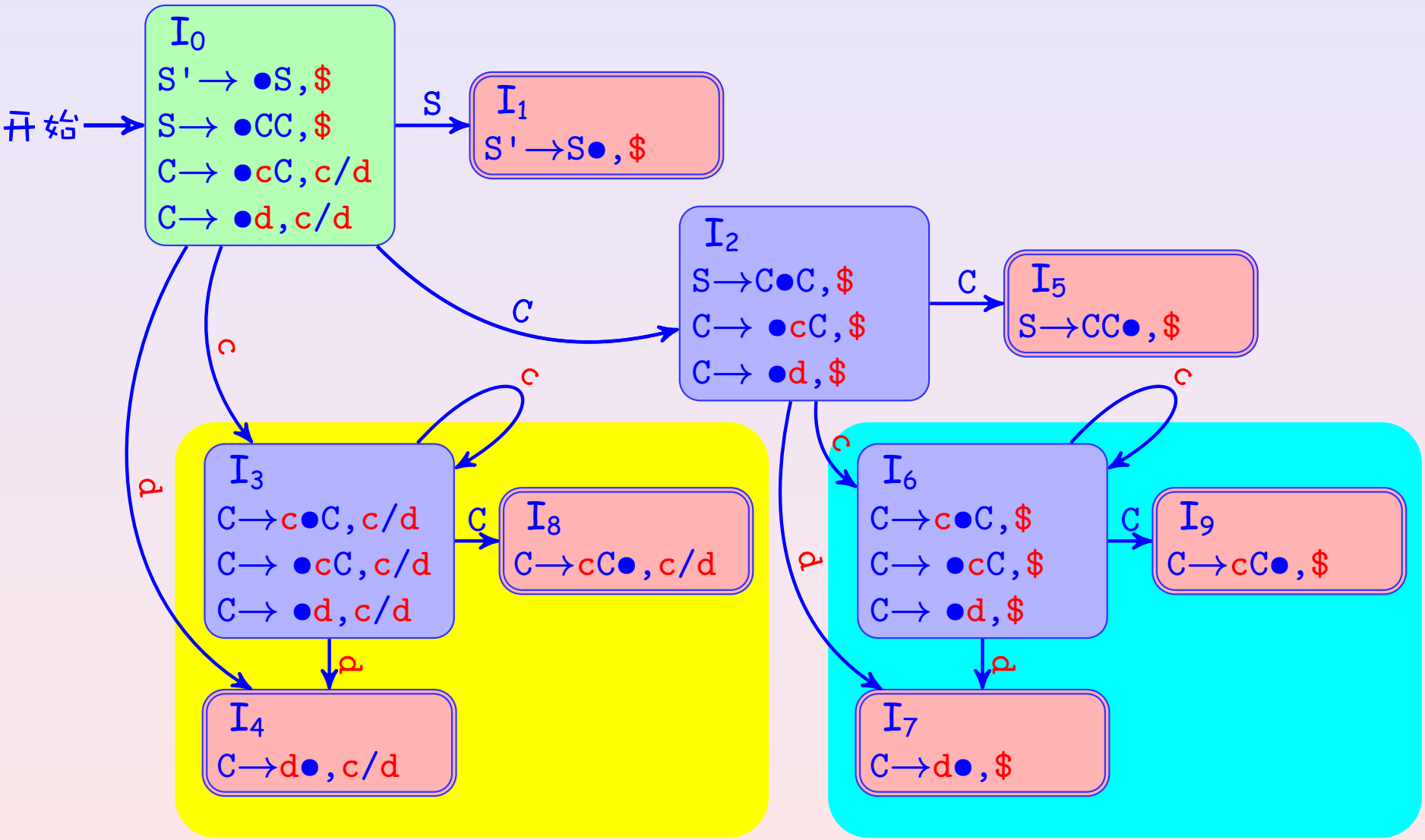
解决 LR(1) 分析器状态过大问题的一个实际解决方案

- LR(1) 规范项目集构造中 ε-closure 求出的项目集的第一分量和 LR(0) 的一致.
- LR(1) DFA 的 Dtrans 函数仅和项目集的第一分量有关，因此，它和 LR(0) 的也是一致的.
- 因此, LR(1) 的规范项目集是 LR(0) 的项目集加上不同的后随符号.

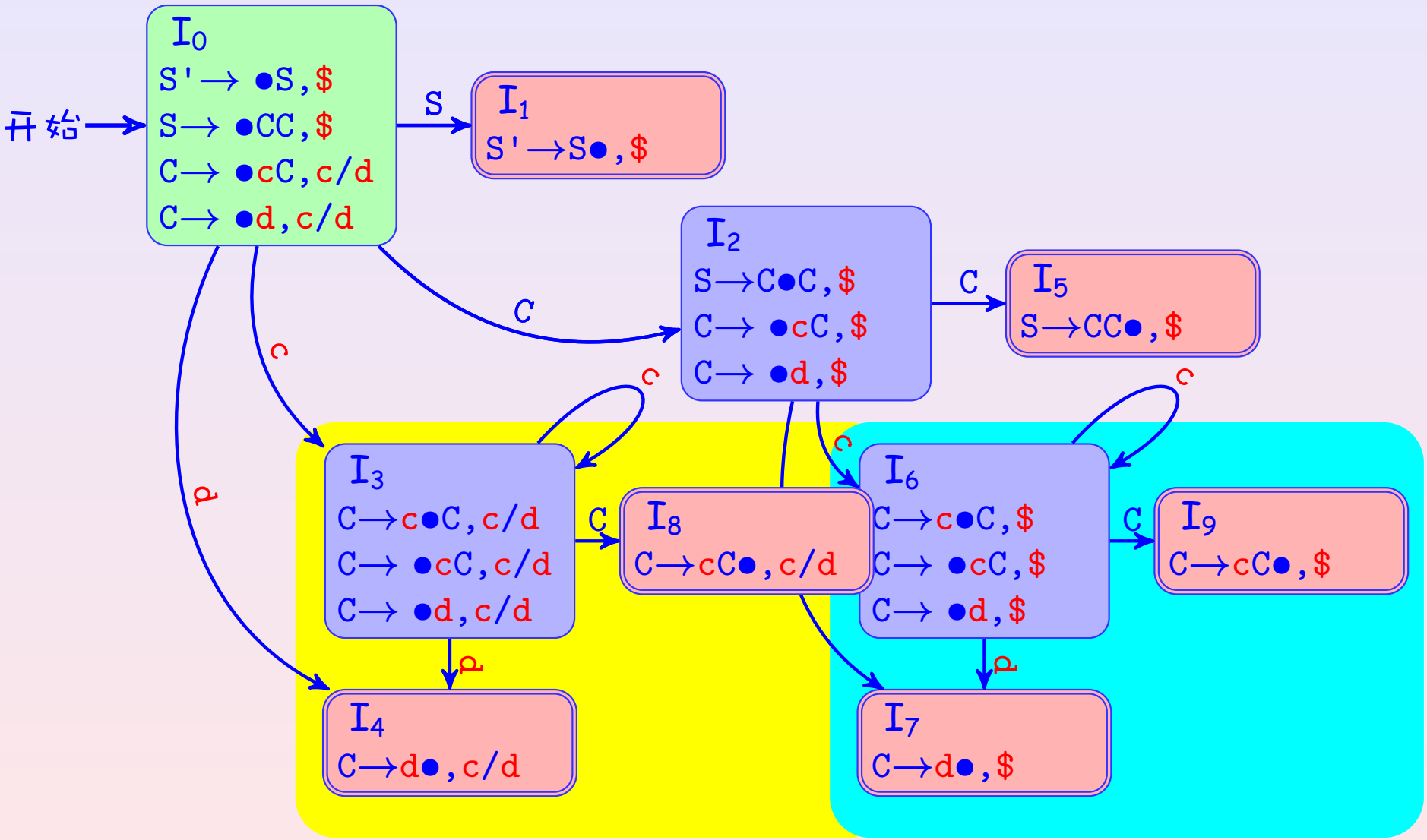
核 (core)

- LR(1) 规范项目集第一分量的集合称为该项目集的核.
- 每个核一定是规范 LR(0) 项目集.
- 设状态 I 和状态 J 对应的规范项目集是相同的核，
  则 Dtrans(I, X) 和 Dtrans(J, X) 也有相同的核，即核对状态转移函数是相容的.
- 合并 LR(1) 的同核项目所得到的自动机称为LALR 自动机，该自动机的状态转换函数和 LR(0) 的一致，唯一不同的是项目集增加了后随符号.

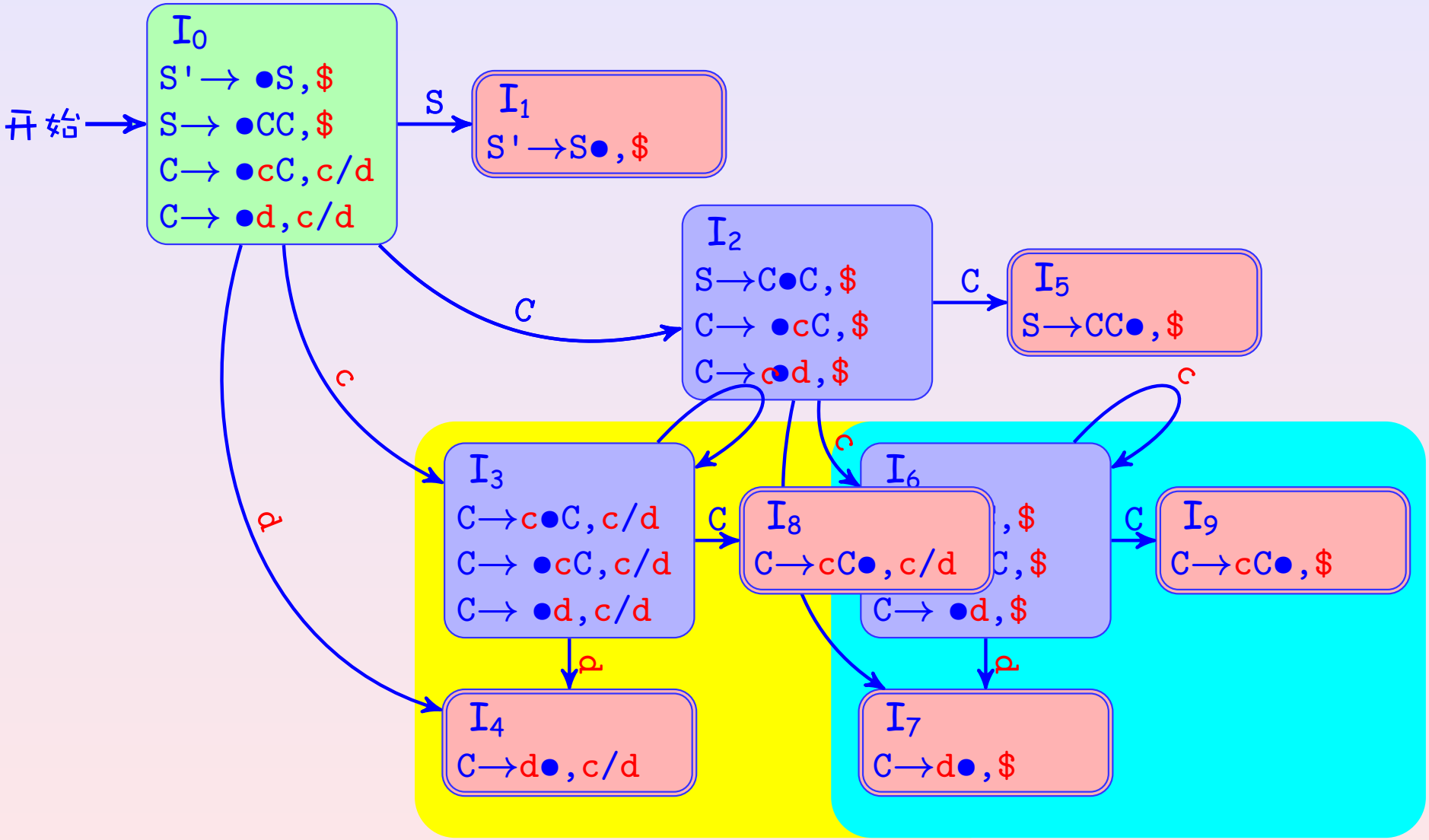
基于 LALR 项目集的前缀自动机的构造



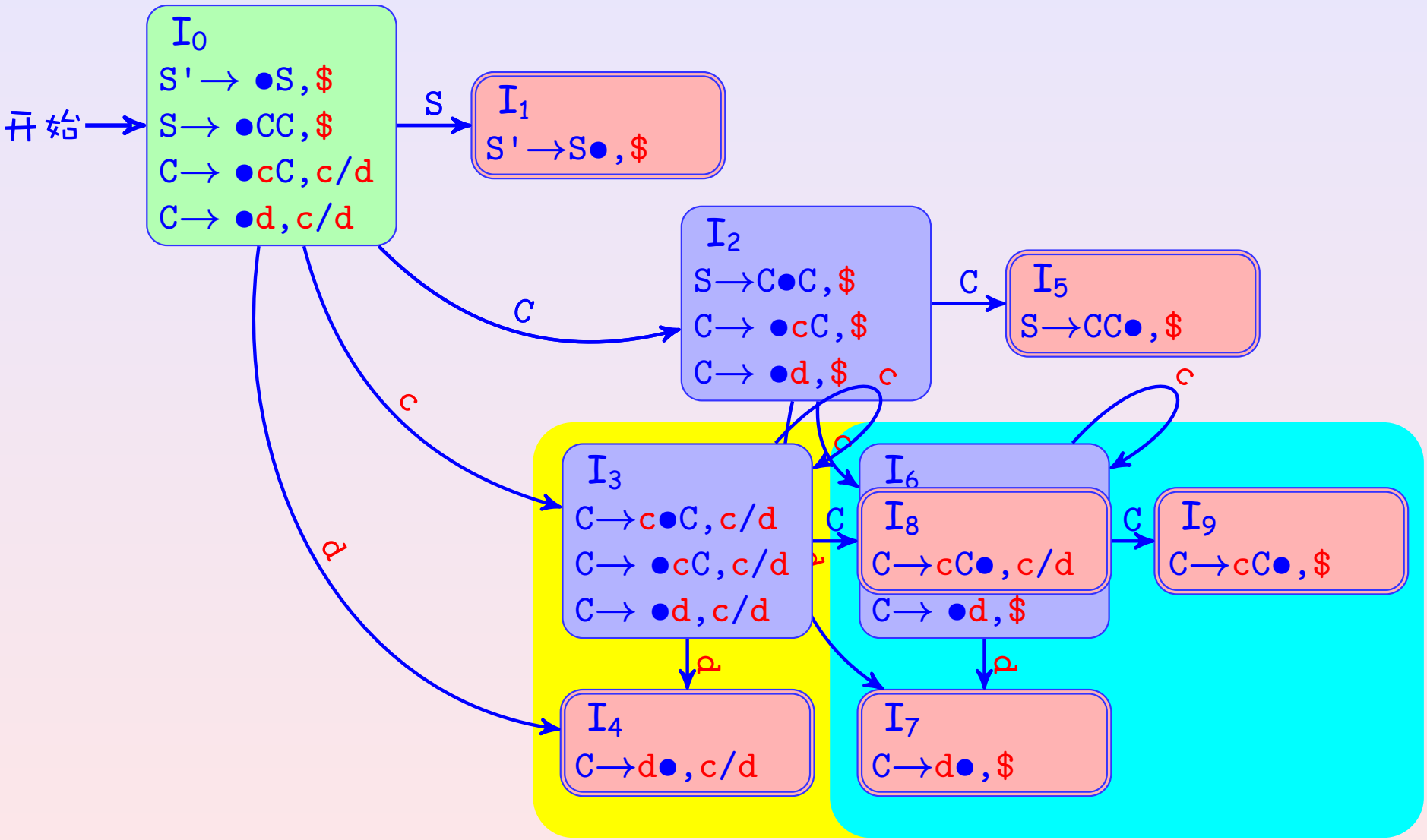
基于 LALR 项目集的前缀自动机的构造



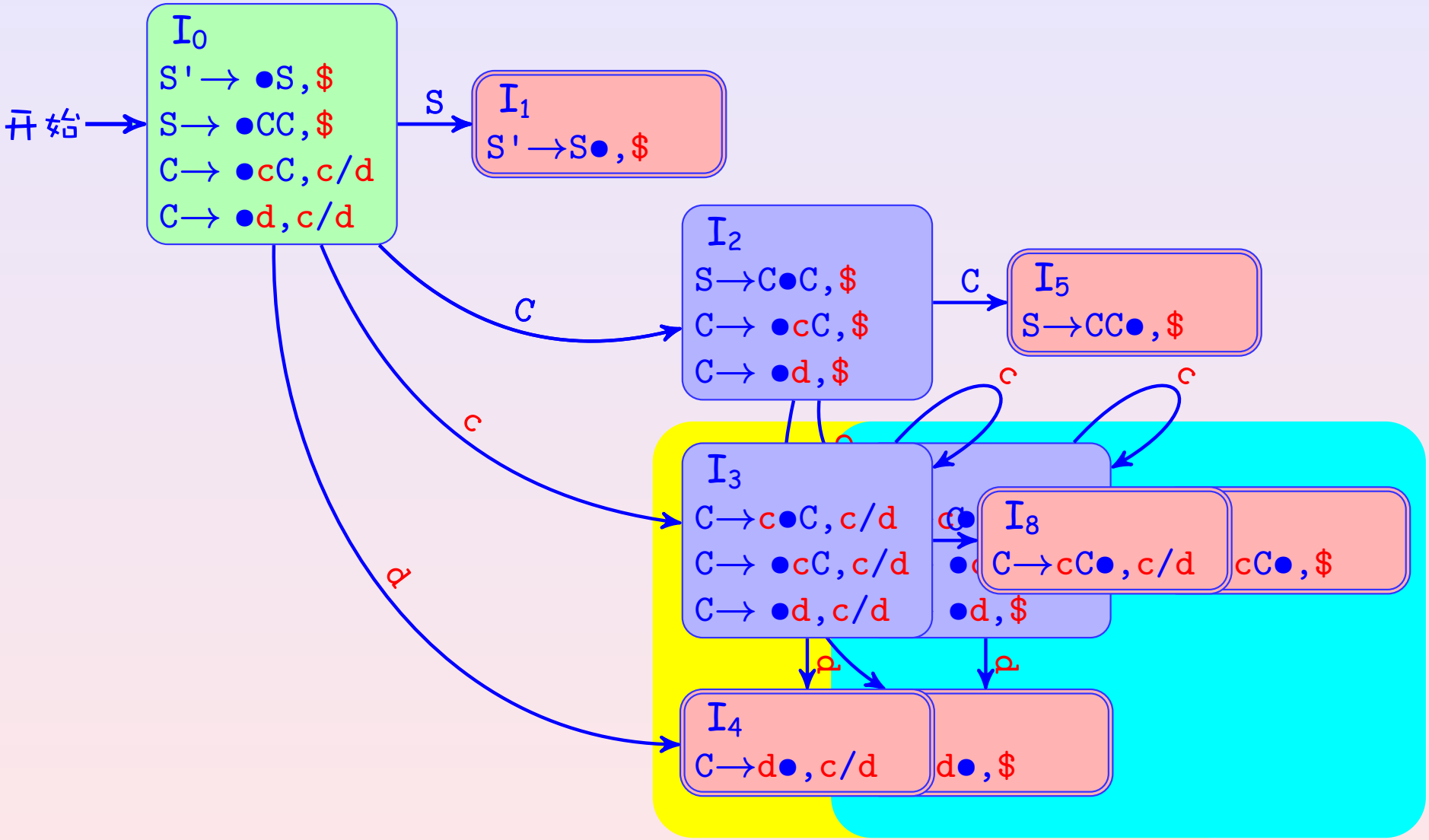
基于 LALR 项目集的前缀自动机的构造



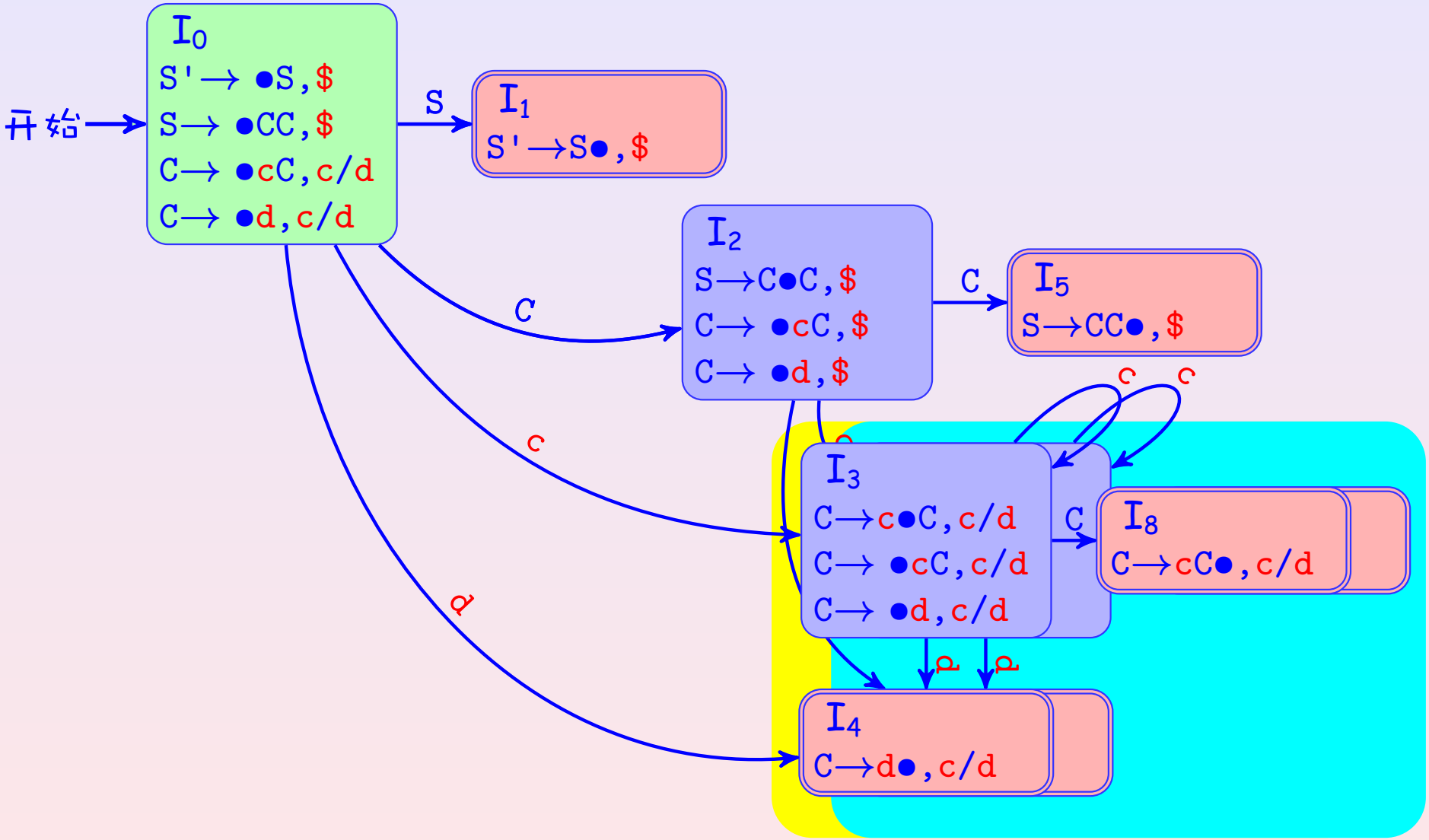
基于 LALR 项目集的前缀自动机的构造



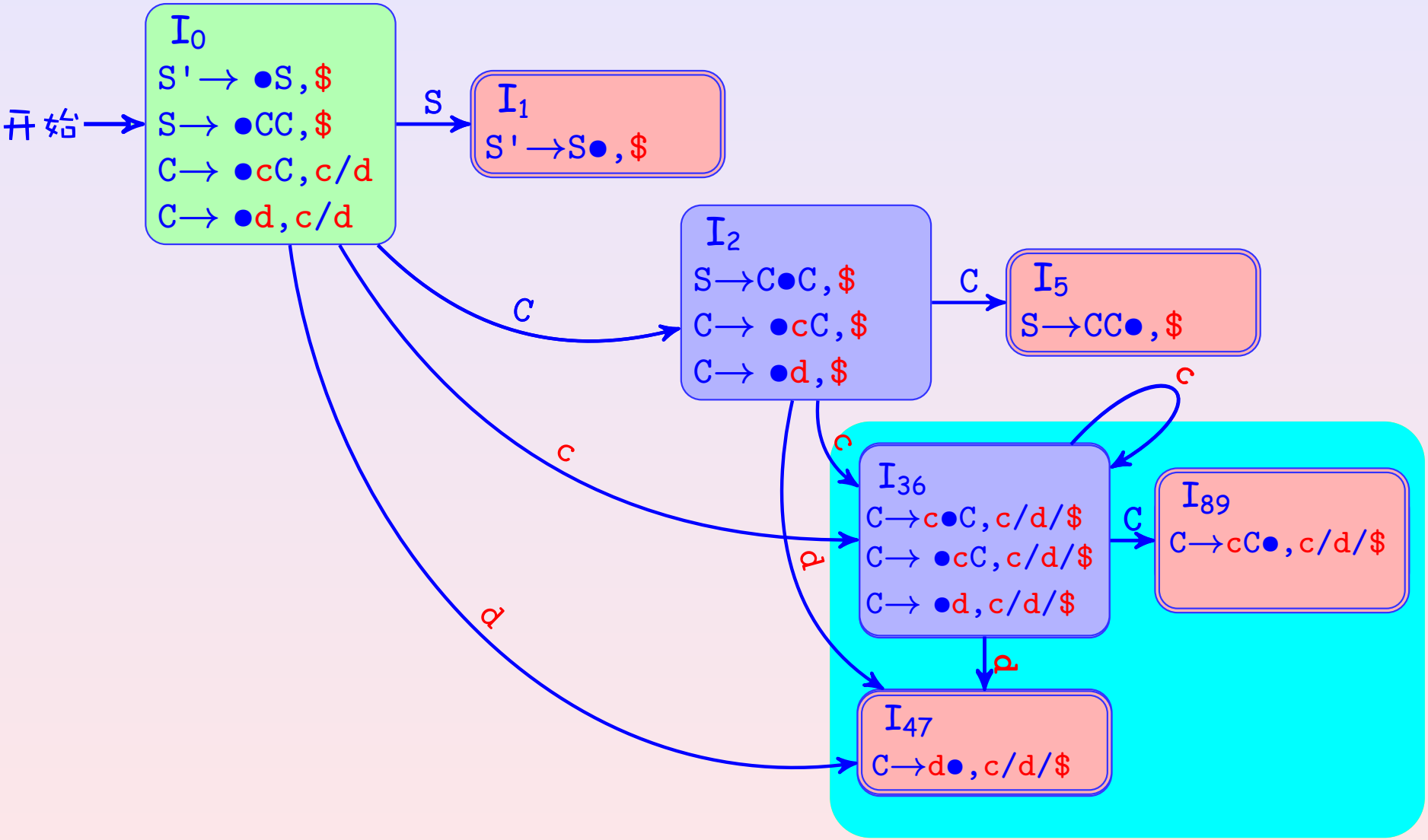
基于 LALR 项目集的前缀自动机的构造



基于 LALR 项目集的前缀自动机的构造



基于 LALR 项目集的前缀自动机的构造



LALR(1) 分析表

文法 G

1/ $S \to CC$ 2/ $C \to cC$ 3/ $C \to d$

	action			goto	
状态	c	d	$	S	C
0	s3	s4		1	2
1			acc		
2	s6	s7			5
36	s36	s47			89
47	r3	r3	r3		
5			r1		
89	r2	r2	r2		

page-break

二义性的处理

二义性的处理

- 二义文法一定不是 LR 文法，LR 文法一定不是二义文法.
- 二义文法比其对应的无二义性文法表达更简洁，自然，如：表达式文法.
- 二义性表现为 LR 分析表中的 S-R 和 R-R 冲突.
- 在保证不破坏识别能力的前提下，通过对冲突项选择适当的操作解消二义性.

二义表达式文法 SLR 分析表

二义表达式文法

1/ E → E+E　2/ E → E*E　3/ E → (E)　4/ E → id

	Action						Goto
状态	id	+	*	(	)	$	E
0	s3			s2			1
1		s4	s5			acc	
2	s3			s2			6
3		r4	r4		r4	r4	
4	s3			s2			8
5	s3			s2			8
6		s4	s5		s9		
7		r1 s4	r1 s5		r1	r1	
8		r2 s4	r2 s5		r2	r2	
9		r3	r2		r3	r3	

表达式文法

- expression : $E \rightarrow E+E \mid E*E \mid (E) \mid id$
- 无层次结构.
- LR(0) 规范项目集 $I_7 = \{E \rightarrow E+E\bullet, E \rightarrow E \bullet +E, E \rightarrow E \bullet *E\}$, 面对 ``+'' 和 ``*'' 有移进 - 归约冲突.
- $action[I_7, +]$ = reduce $E \rightarrow E+E$, 表示 ``+'' 左结合.
- $action[I_7, *]$ = shift J, 表示 ``*'' 运算的优先级别比 ``+'' 要高.
- LR(0) 规范项目集 $I_8 = \{E \rightarrow E*E\bullet, E \rightarrow E \bullet +E, E \rightarrow E \bullet *E\}$, 面对 ``+'' 和 ``*'' 有移进 - 归约冲突.
- $action[I_8, +]$ = reduce $E \rightarrow E*E$, 表示 ``*'' 运算的优先级别比 ``+'' 要高.
- $action[I_8, *]$ = reduce $E \rightarrow E*E$, 表示 ``*'' 左结合.

二义表达式文法 SLR 分析表.

二义表达式文法

1/ E → E+E　2/ E → E*E　3/ E → (E)　4/ E → id

	Action						Goto
状态	id	+	*	(	)	$	E
0	s3			s2			1
1		s4	s5			acc	
2	s3			s2			6
3		r4	r4		r4	r4	
4	s3			s2			7
5	s3			s2			8
6		s4	s5		s9		
7		r1	s5		r1	r1	
8		r2	r2		r2	r2	
9		r3	r2		r3	r3	

含有一元减的表达式文法.

- 表达式文法: $E \to E\text{-}E \mid E() \mid \text{-}E \mid \text{id}$.
- 无层次结构.
- LR(0) 规范项目集 $I = \{E \to \text{-}E\bullet, E \to E \bullet \text{-}E, E \to E \bullet ()\}$, 面对 ``-'' 和 ``('' 有移进 - 归约冲突.
- action[I, -] = reduce $E \to \text{-}E$, 表示一元 ``-'' 的优先级别比二元 ``-'' 的高.
- action[I, (] = shift J, 表示后缀运算 ``()'' 运算的优先级别比一元 ``-'' 要高.

if-then-else 结构

- if-then-else 文法：$S \to iSeS \mid iS \mid a$.
- LR(0) 规范项目集：$I = \{S \to iS \bullet eS, S \to iS\bullet\}$，面对 ``e'' 有移进 - 归约冲突.
- action[I, e] = shift J，表示 ``e'' 将挂靠最近的 ``i''.
- action[I, e] = reduce $S \to iS$，将破坏分析器的识别能力，如：``iaea'' 在移进 ``ia'' 之后到达状态 I，使用归约操作后，再也无法移进 ``e''.

出错处理

- 分析表对应的 Action 空白项表示出错，
- 一般处理方法：
  - 如果出错状态含有项目 $A \to \alpha \bullet \beta$，表示希望形成形如 ``$\alpha\beta$'' 的句柄，但是当前的错误输入使得分析器不能移进，此时可采用类似 LL 分析的方法，跳过当前的输入直到非终结符 A 的同步符号出现，用 $A \to \alpha\beta$ 归约继续分析.
  - 如果出错状态含有项目 $A \to \alpha\bullet$，表示当前的输入不是 A 的 Follow 集元素，处理方法，归约该项目，转入和前者一样的处理.

二义表达式文法出错分析 --- error1.

- Input: ``id+ *id'', 分析器在移进 ``id+'' 进入状态 $I_4 = \{E \to E+\bullet E,\ E \to \bullet E+E,\ E \to \bullet E*E,\ E \to \bullet(E),\ E \to \bullet id\}$, 面对 ``*'' 出错, 此时的状态希望移进一个表达式成分 ``id'' 或 ``(''.
- 报错: 缺少一个运算量.
- 恢复方法: 移进一个虚拟的 ``id'' 到下一个状态 $J = \{E \to id\bullet\}$.
- 相关的状态: 有移进 ``id'' 的状态.

二义表达式文法出错分析 --- error2.

- Input: ``id+ )id'', 分析器在移进 ``id+'' 进入状态 $I_4 = \{E \to E+ \bullet E,\ E \to \bullet E+E,\ E \to \bullet E*E,\ E \to \bullet(E),\ E \to \bullet id\}$, 面对 ``)'' 出错, 此时的项目集中不含有括号项目, 表示没有对应的左括号.
- 报错: 不平衡出现的右括号.
- 恢复方法: skip ``)'', read next token .
- 相关的输入: ``)''.

二义表达式文法出错分析 --- error3.

- Input: ``id+ id id'', 分析器在移进 ``id+'' 进入状态 $I_3 = \{E \to id\bullet\}$, 面对 ``id'' 出错, 状态 I_3 仅能在面对 $Follow(E) = \{+, *,), \$\}$ 的元素才能归约, 即缺少一个运算符.
- 报错: missing an operator.
- 恢复方法:
  - reduce $E \to id$;
  - goto state $I_7 = \{E \to E+E\bullet, E \to E\bullet +E, E \to E\bullet *E\}$;
  - reduce $E \to E+E$;
  - goto state $I_1 = \{E' \to E\bullet, E \to E\bullet +E, E \to E\bullet *E\}$;
  - 移进虚拟运算符号 ``+'', 转移状态到 $I_4 = \{E \to E+\bullet E, E \to \bullet E+E, E \to \bullet E*E, E \to \bullet(E), E \to \bullet id\}$
- 有归约项目的状态面对任何输入都执行归约操作.

二义表达式文法出错分析 --- error4.

- Input: ``(id+ id'', 分析器在移进 ``(id+id'' 进入状态 $I_6 = \{E \to E\bullet *E,\ E \to E\bullet +E,\ E \to (E\bullet)\}$, 面对 ``$'' 出错, 状态 $I_6$ 希望移进一个 ``)'' 形成平衡括号对.
- 报错: missing a right parenthesis.
- 恢复方法:
  - 移进一个虚拟的 ``)'' 到 state $I_9 = \{E \to (E)\bullet\}$;;
  - reduce $E \to (E)$;
  - goto state $I_1 = \{E' \to E\bullet,\ E \to E\bullet +E,\ E \to E\bullet *E\}$.
- 相关状态: 6.
- error4 总是在面对 $ 时发生.

二义表达式文法 SLR 分析表.

二义表达式文法

1/ E → E+E　2/ E → E*E　3/ E → (E)　4/ E → id

	Action						Goto
状态	id	+	*	(	)	$	E
0	s3			s2			1
1		s4	s5			acc	
2	s3			s2			6
3		r4	r4		r4	r4	
4	s3			s2			7
5	s3			s2			8
6		s4	s5		s9		
7		r1	s5		r1	r1	
8		r2	r2		r2	r2	
9		r3	r2		r3	r3	

二义表达式文法 SLR 分析表.

分析表的出错处理

1/ E → E+E　2/ E → E*E　3/ E → (E)　4/ E → id

	Action						Goto
状态	id	+	*	(	)	$	E
0	s3	e1	e1	s2	e2	e1	1
1	e3	s4	s5	e3	e2	acc	
2	s3	e1	e1	s2	e2	e1	6
3	r4	r4	r4	r4	r4	r4	
4	s3	e1	e1	s2	e2	e1	7
5	s3	e1	e1	s2	e2	e1	8
6	e3	s4	s5	e3	s9	e4	
7	r1	r1	s5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r2	r3	r3	r3	

分析过程

stack	symbols	input	action
0		id+)$	shift
03	id	+)$	reduce by E→id
01	E	+)$	shift
014	E+	)$	e2: 多余右括号，skip
014	E+	$	e1: 缺少运算量
			压id入栈到状态 3
0143	E+id	$	reduce by E→id
0147	E+E	$	reduce by E→E+E
01	E	$	accept

分析过程

stack	symbols	input	action
0		id+)$	shift
03	id	+)$	reduce by E→id
01	E	+)$	shift
014	E+	)$	e2: 多余右括号，skip

	Action						Goto
状态	id	+	*	(	)	$	E
0	s3	e1	e1	s2	e2	e1	1
1	e3	s4	s5	e3	e2	acc	
2	s3	e1	e1	s2	e2	e1	6
3	r4	r4	r4	r4	r4	r4	
4	s3	e1	e1	s2	e2	e1	7
5	s3	e1	e1	s2	e2	e1	8
6	e3	s4	s5	e3	s9	e4	
7	r1	r1	s5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r2	r3	r3	r3	

分析过程

stack	symbols	input	action
0		id+)$	shift
03	id	+)$	reduce by E→id
01	E	+)$	shift
014	E+	)$	e2: 多余右括号，skip

	Action						Goto
状态	id	+	*	(	)	$	E
0	s3	e1	e1	s2	e2	e1	1
1	e3	s4	s5	e3	e2	acc	
2	s3	e1	e1	s2	e2	e1	6
3	r4	r4	r4	r4	r4	r4	
4	s3	e1	e1	s2	e2	e1	7
5	s3	e1	e1	s2	e2	e1	8
6	e3	s4	s5	e3	s9	e4	
7	r1	r1	s5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r2	r3	r3	r3	

分析过程

stack	symbols	input	action
0		id+)$	shift
03	id	+)$	reduce by E→id
01	E	+)$	shift
014	E+	)$	e2: 多余右括号，skip

	Action						Goto
状态	id	+	*	(	)	$	E
0	s3	e1	e1	s2	e2	e1	1
1	e3	s4	s5	e3	e2	acc	
2	s3	e1	e1	s2	e2	e1	6
3	r4	r4	r4	r4	r4	r4	
4	s3	e1	e1	s2	e2	e1	7
5	s3	e1	e1	s2	e2	e1	8
6	e3	s4	s5	e3	s9	e4	
7	r1	r1	s5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r2	r3	r3	r3	

分析过程

stack	symbols	input	action
0		id+)$	shift
03	id	+)$	reduce by E→id
01	E	+)$	shift
014	E+	)$	e2: 多余右括号, skip

	Action						Goto
状态	id	+	*	(	)	$	E
0	s3	e1	e1	s2	e2	e1	1
1	e3	s4	s5	e3	e2	acc	
2	s3	e1	e1	s2	e2	e1	6
3	r4	r4	r4	r4	r4	r4	
4	s3	e1	e1	s2	e2	e1	7
5	s3	e1	e1	s2	e2	e1	8
6	e3	s4	s5	e3	s9	e4	
7	r1	r1	s5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r2	r3	r3	r3	

分析过程

	Action						Goto
状态	id	+	*	(	)	$	E
0	s3	e1	e1	s2	e2	e1	1
1	e3	s4	s5	e3	e2	acc	
2	s3	e1	e1	s2	e2	e1	6
3	r4	r4	r4	r4	r4	r4	
4	s3	e1	e1	s2	e2	e1	7
5	s3	e1	e1	s2	e2	e1	8
6	e3	s4	s5	e3	s9	e4	
7	r1	r1	s5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r2	r3	r3	r3	

st			xip
0			
03			
01			
01			
014	E+	$	e1: 缺少运算量
			压id入栈到状态 3
0143	E+id	$	reduce by E→id
0147	E+E	$	reduce by E→E+E
01	E	$	accept

分析过程

	Action						Goto
状态	id	+	*	(	)	$	E
0	s3	e1	e1	s2	e2	e1	1
1	e3	s4	s5	e3	e2	acc	
2	s3	e1	e1	s2	e2	e1	6
3	r4	r4	r4	r4	r4	r4	
4	s3	e1	e1	s2	e2	e1	7
5	s3	e1	e1	s2	e2	e1	8
6	e3	s4	s5	e3	s9	e4	
7	r1	r1	s5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r2	r3	r3	r3	

st			
0			
03			
01			
01			kip
014	E+	$	e1: 缺少运算量
			压id入栈到状态 3
0143	E+id	$	reduce by E→id
0147	E+E	$	reduce by E→E+E
01	E	$	accept

分析过程

	Action						Goto
状态	id	+	*	(	)	$	E
0	s3	e1	e1	s2	e2	e1	1
1	e3	s4	s5	e3	e2	acc	
2	s3	e1	e1	s2	e2	e1	6
3	r4	r4	r4	r4	r4	r4	
4	s3	e1	e1	s2	e2	e1	7
5	s3	e1	e1	s2	e2	e1	8
6	e3	s4	s5	e3	s9	e4	
7	r1	r1	s5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r2	r3	r3	r3	

st			kip
0			
03			
01			
01			
014	E+	$	e1: 缺少运算量 压id入栈到状态 3
0143	E+id	$	reduce by E→id
0147	E+E	$	reduce by E→E+E
01	E	$	accept

分析过程

	Action						Goto
状态	id	+	*	(	)	$	E
0	s3	e1	e1	s2	e2	e1	1
1	e3	s4	s5	e3	e2	acc	
2	s3	e1	e1	s2	e2	e1	6
3	r4	r4	r4	r4	r4	r4	
4	s3	e1	e1	s2	e2	e1	7
5	s3	e1	e1	s2	e2	e1	8
6	e3	s4	s5	e3	s9	e4	
7	r1	r1	s5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r2	r3	r3	r3	

st			
0			
03			
01			
01			kip
014	E+	$	e1: 缺少运算量
			压id入栈到状态 3
0143	E+id	$	reduce by E→id
0147	E+E	$	reduce by E→E+E
01	E	$	accept

分析过程

	Action						Goto
状态	id	+	*	(	)	$	E
0	s3	e1	e1	s2	e2	e1	1
1	e3	s4	s5	e3	e2	acc	
2	s3	e1	e1	s2	e2	e1	6
3	r4	r4	r4	r4	r4	r4	
4	s3	e1	e1	s2	e2	e1	7
5	s3	e1	e1	s2	e2	e1	8
6	e3	s4	s5	e3	s9	e4	
7	r1	r1	s5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r2	r3	r3	r3	

st			
0			
03			
01			
01			kip
014	E+	$	e1: 缺少运算量
			压id入栈到状态 3
0143	E+id	$	reduce by E→id
0147	E+E	$	reduce by E→E+E
01	E	$	accept

相关工具

自下而上的语法分析器生成工具

将含有附注的上下文无关文法转换以某种程序设计语言识别为输出的识别该文法语法分析器源程序.

Examples

1. yacc(Yet Another Compiler-Compiler): 1975 年由贝尔实验室 Mike Lesk & Eric Schmidt 开发, UNIX 标准实用工具 (utility);
2. byacc: Berkeley YACC: Robert Corbett, 1989 年, yacc compatible, in Free BSD distribution, DOS version in my CD-ROM;
3. bison: Robert Corbett & Richard Stallmen, 1988 年, yacc compatible, in Linux distribution, 最新版本: 2.4. 支持 GLR(Generalized LR) 文法, http://www.gnu.org/software/bison/.
4. CUP: LALR Parser Generator in Java, current version 0.11a, 对应的词法分析器生成工具为: JFLex(http://jflex.de/), http://www2.cs.tum.edu/projects/cup/.
5. A. Holub LRpars: See CD-ROM, 支持动态显示分析过程.