ビジュアル構文解析

Kazuhiro Ichikawa
@phenan

概要

- 様々な構文解析アルゴリズムを図解で紹介
  - ➢ 古典的な手法から最近のものまで
  - ➢ "なんとなく理解できる" のを目指す
- スライド
  - ➢ www.csg.ci.i.u-tokyo.ac.jp/~ichikawa/visual_parsing.pdf
  - ➢ 重いのでダウンロード推奨

紹介するアルゴリズム

演算子順位構文解析

- Shunting-yard algorithm
- Pratt parsing

上向き構文解析

- LR parsing
- GLR parsing
- Earley parsing
- CYK algorithm
- Valiant parsing

下向き構文解析

- LL(k) parsing
- Packrat parsing
- Packrat parsing with left recursion
- GLL parsing
- Adaptive LL(*) parsing

紹介するアルゴリズム

演算子順位構文解析

- Shunting-yard algorithm 中置記法を後置記法に直す
- Pratt parsing 再帰下降型演算子順位構文解析

上向き構文解析

- LR parsing 決定性有限オートマトン
- GLR parsing 非決定性有限オートマトン
- Earley parsing 動的計画法
- CYK algorithm 動的計画法
- Valiant parsing 行列積

下向き構文解析

- LL(k) parsing k文字先読み
- Packrat parsing メモ化再帰
- Packrat parsing with left recursion 強欲マッチ
- GLL parsing グラフ構造継続
- Adaptive LL(*) parsing 先読みオートマトンの動的構築

Operator precedence parsing

演算子順位構文解析

Shunting-yard algorithm [Dijkstra 1961]

- 最も簡単な上向き構文解析
- 数式の構文解析が得意

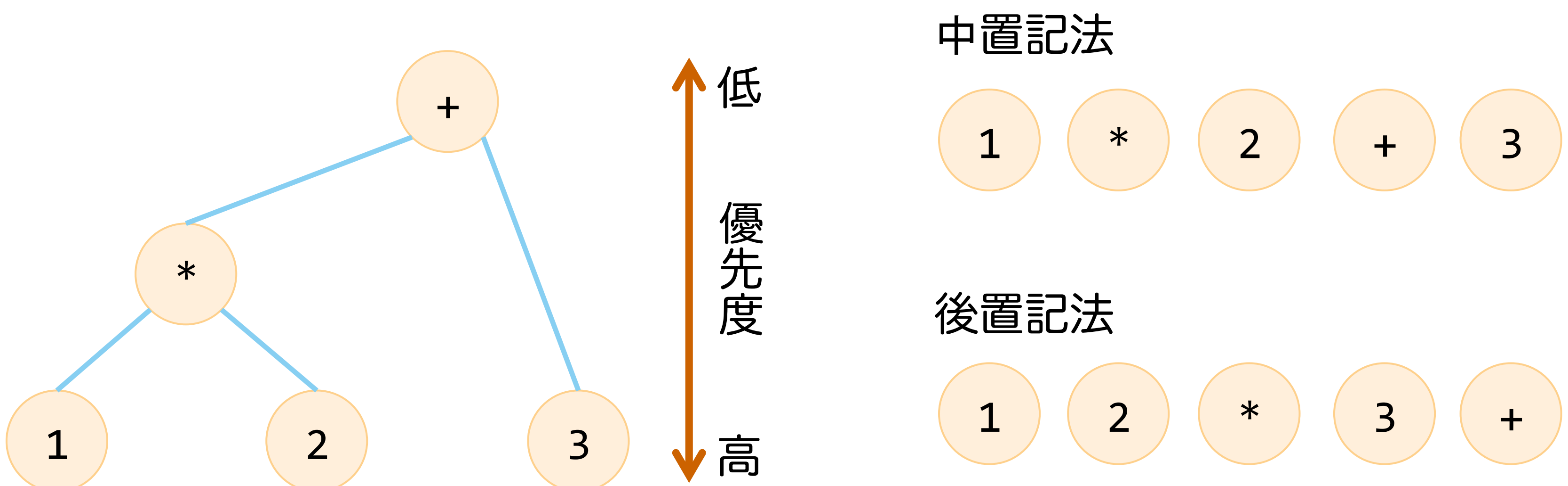
Pratt parsing [Pratt 1973]

- 下向きの演算子順位構文解析
- 手書きの再帰下降構文解析器の一部として利用される
- 数式の構文解析が得意

Shunting-yard algorithm [Dijkstra 1961]

アイデア

- 演算子優先度を使って後置記法に直す
- スタックを使い優先度順に並べ替える



	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0

入力

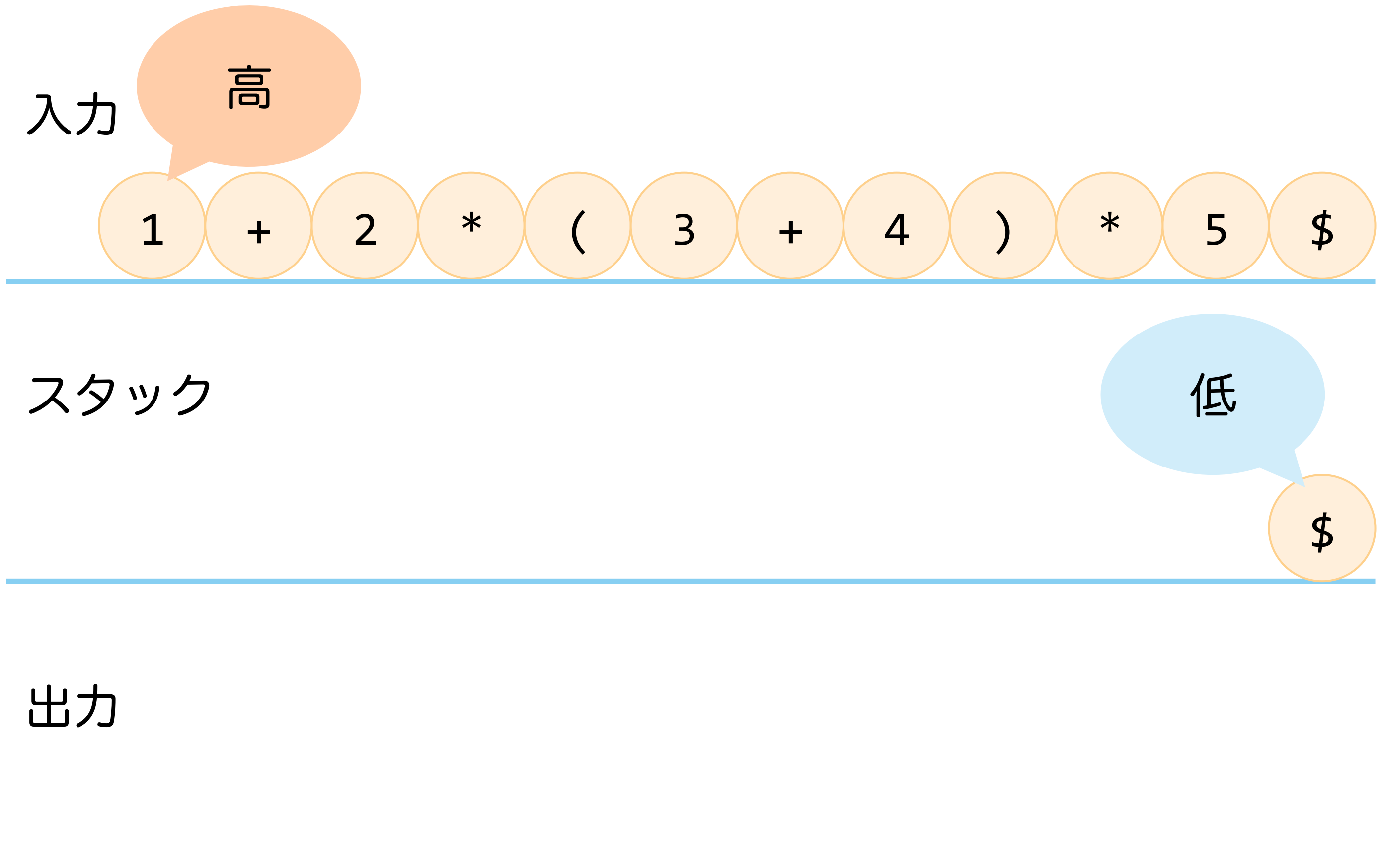
1 + 2 * (3 + 4) * 5 $

スタック

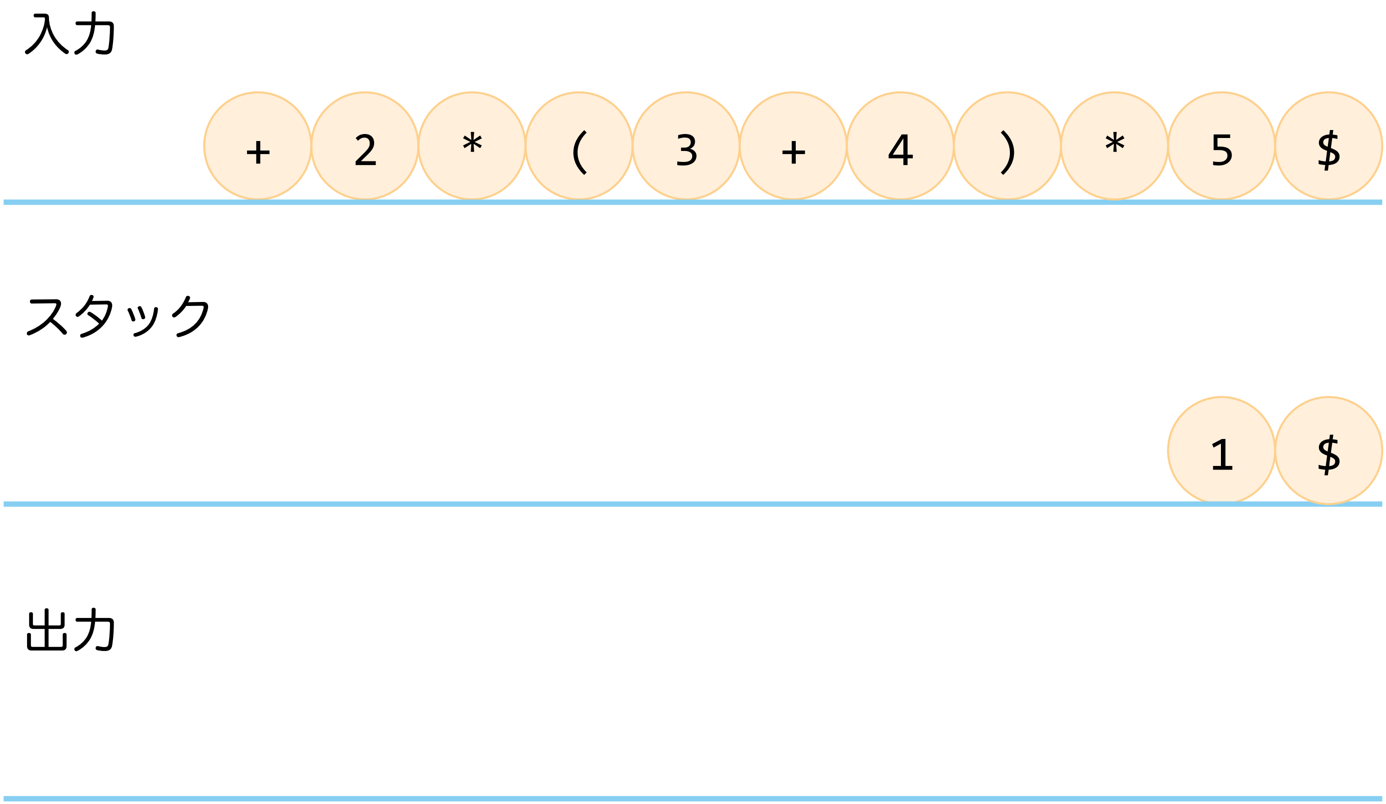
$

出力

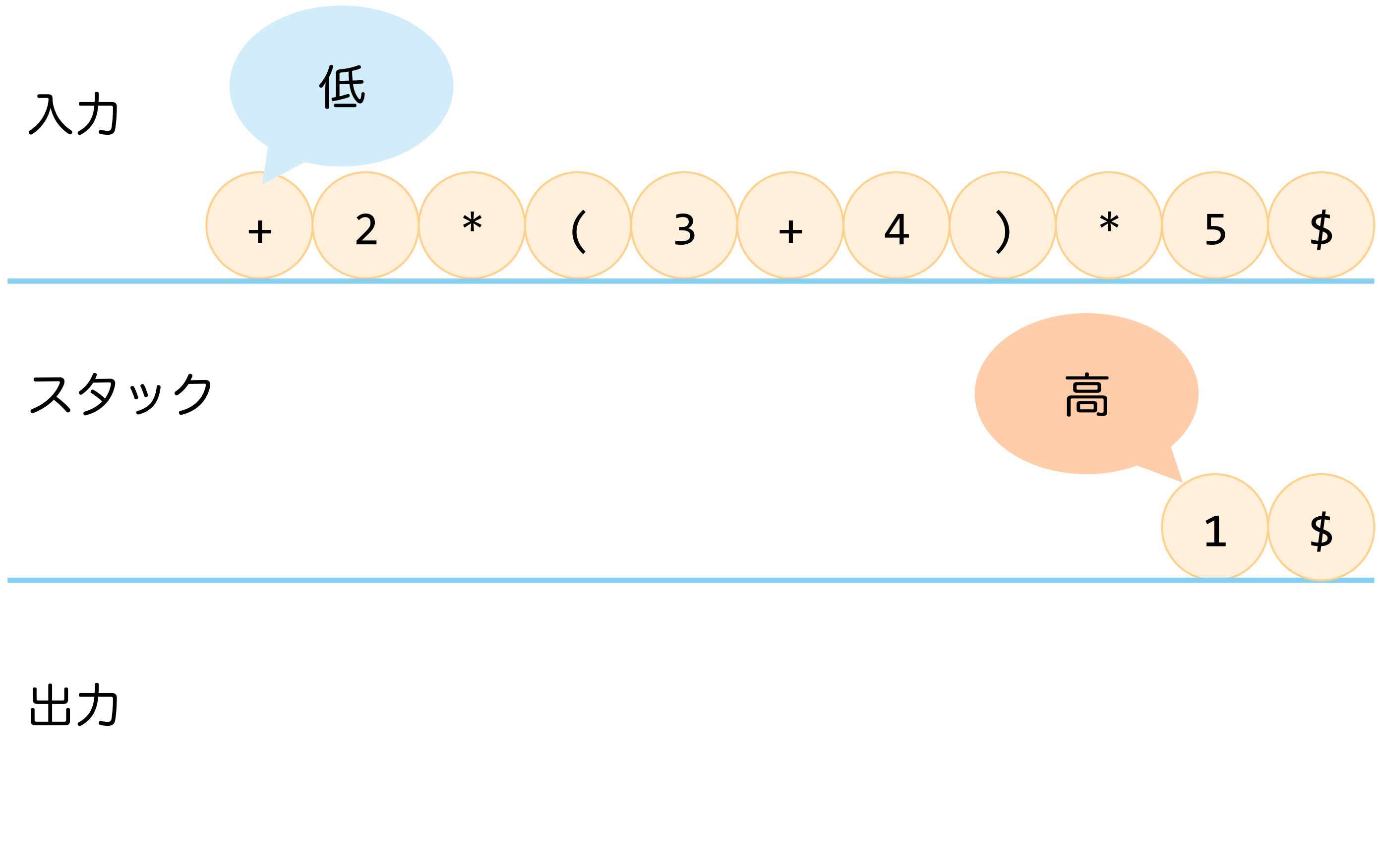
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



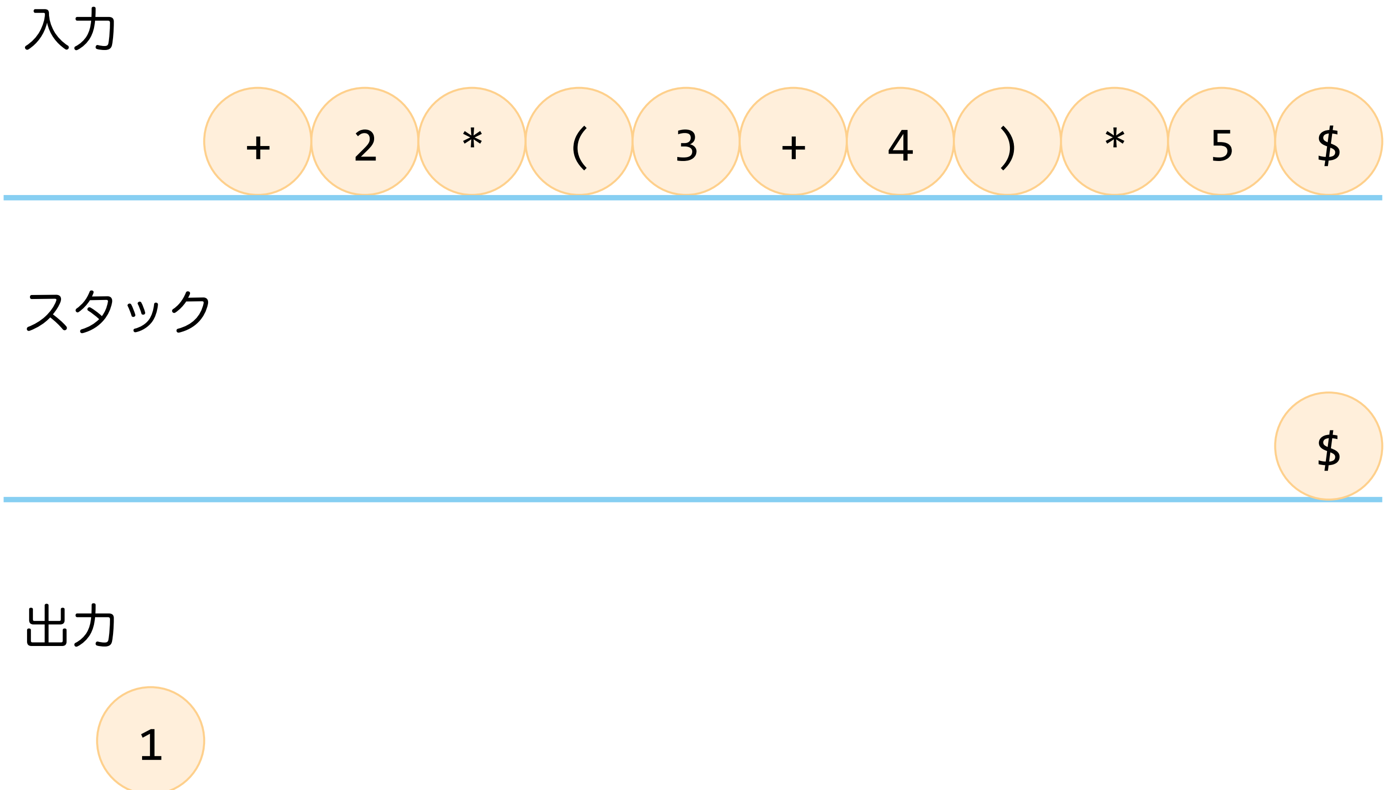
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



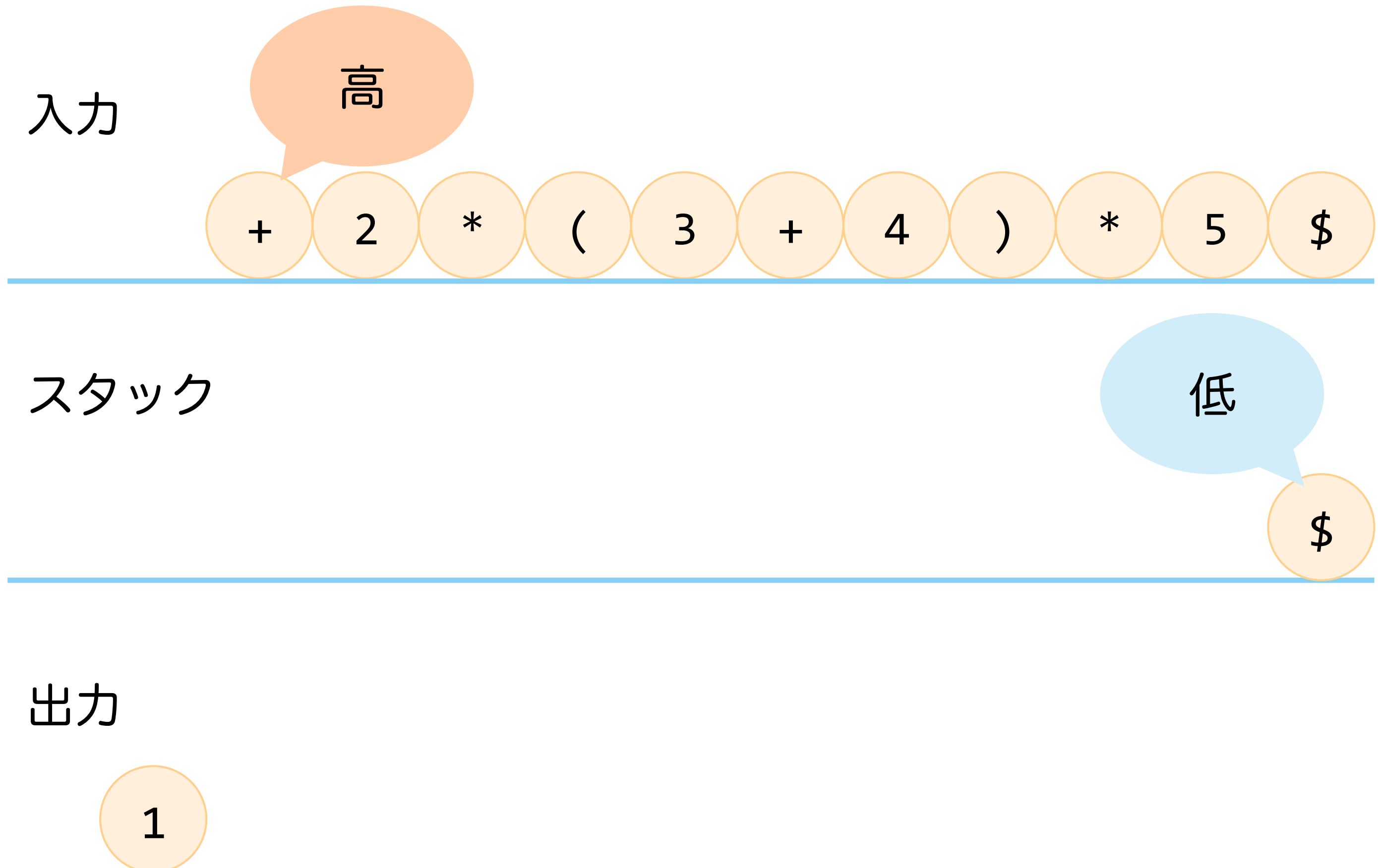
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



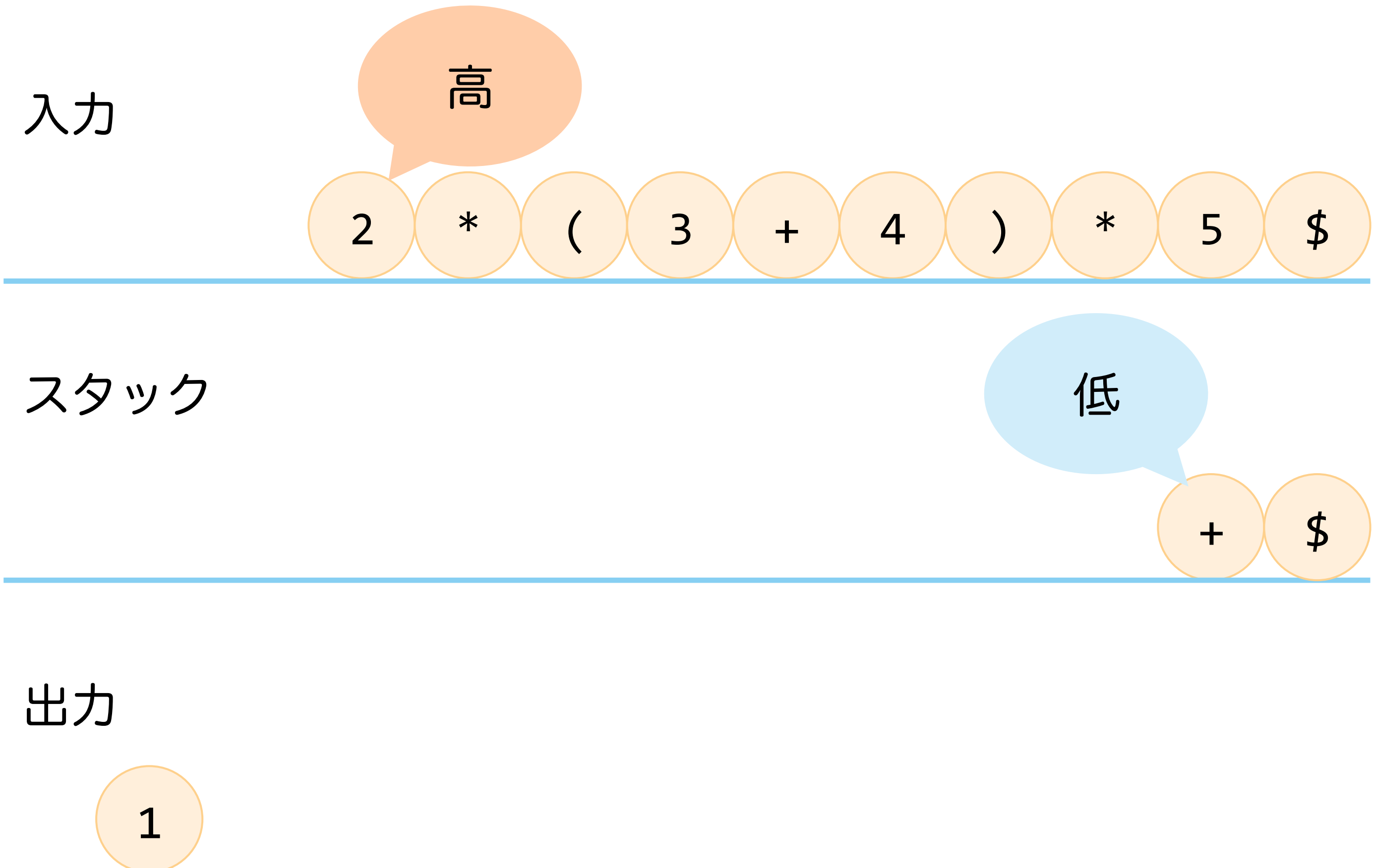
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



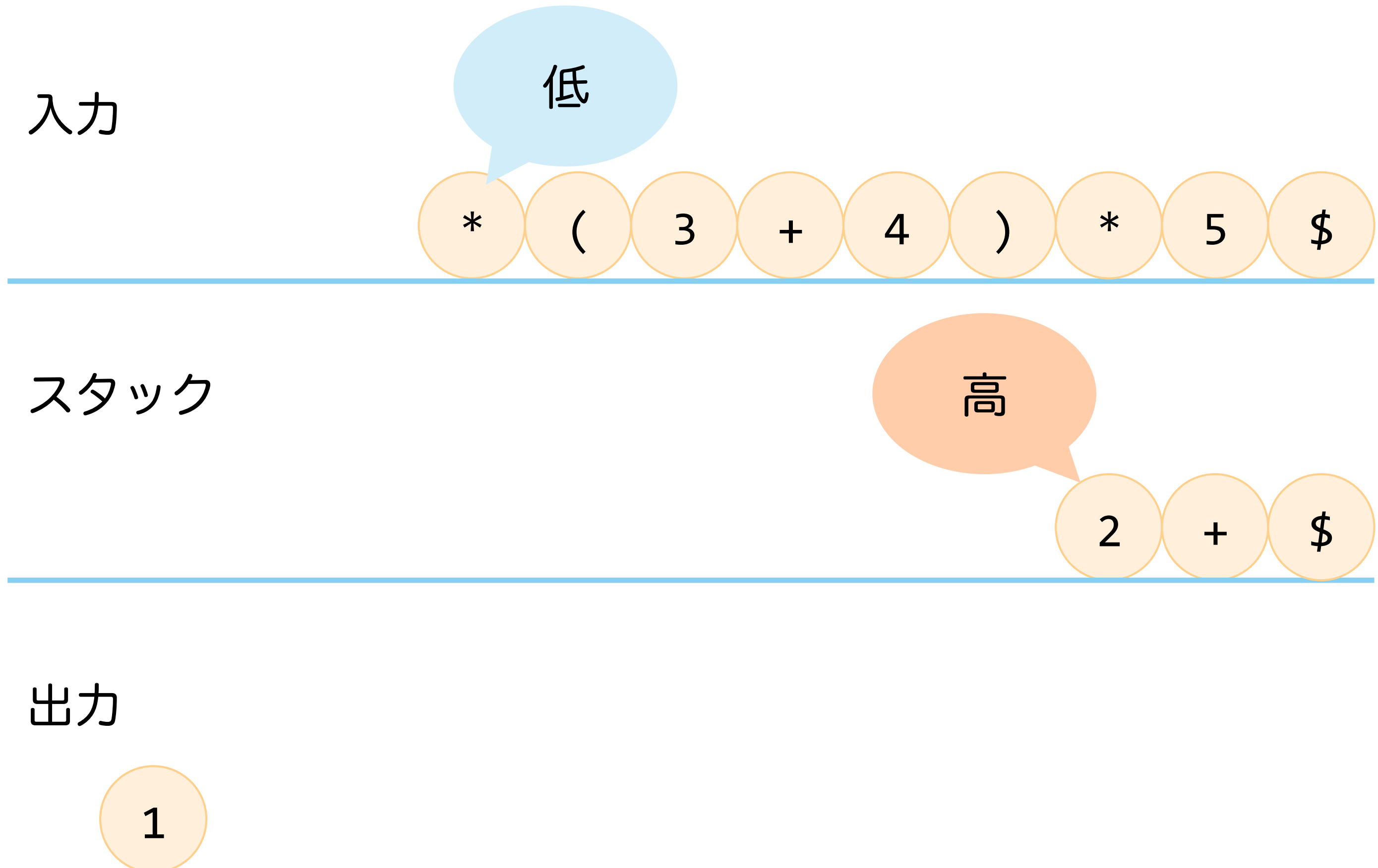
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



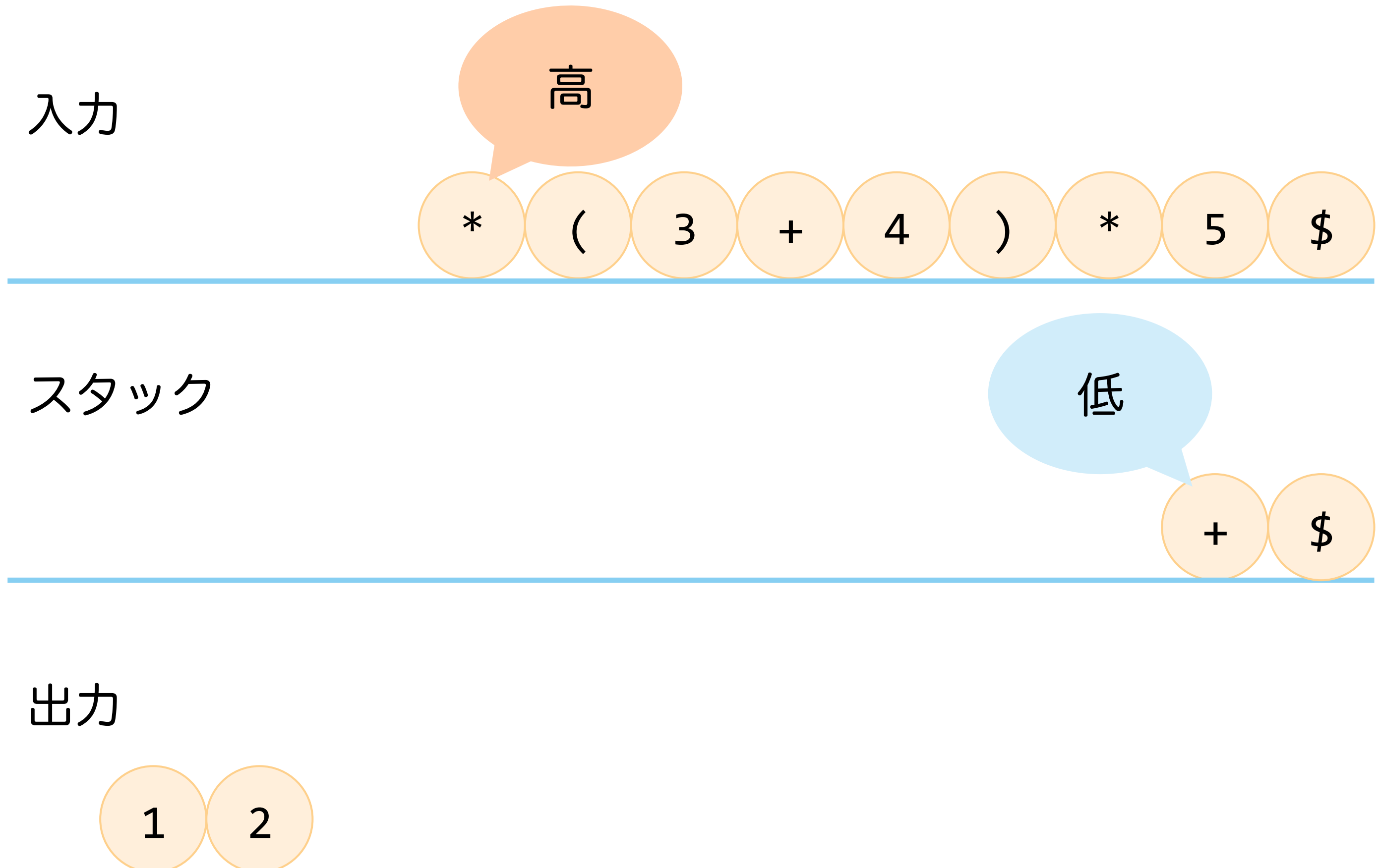
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



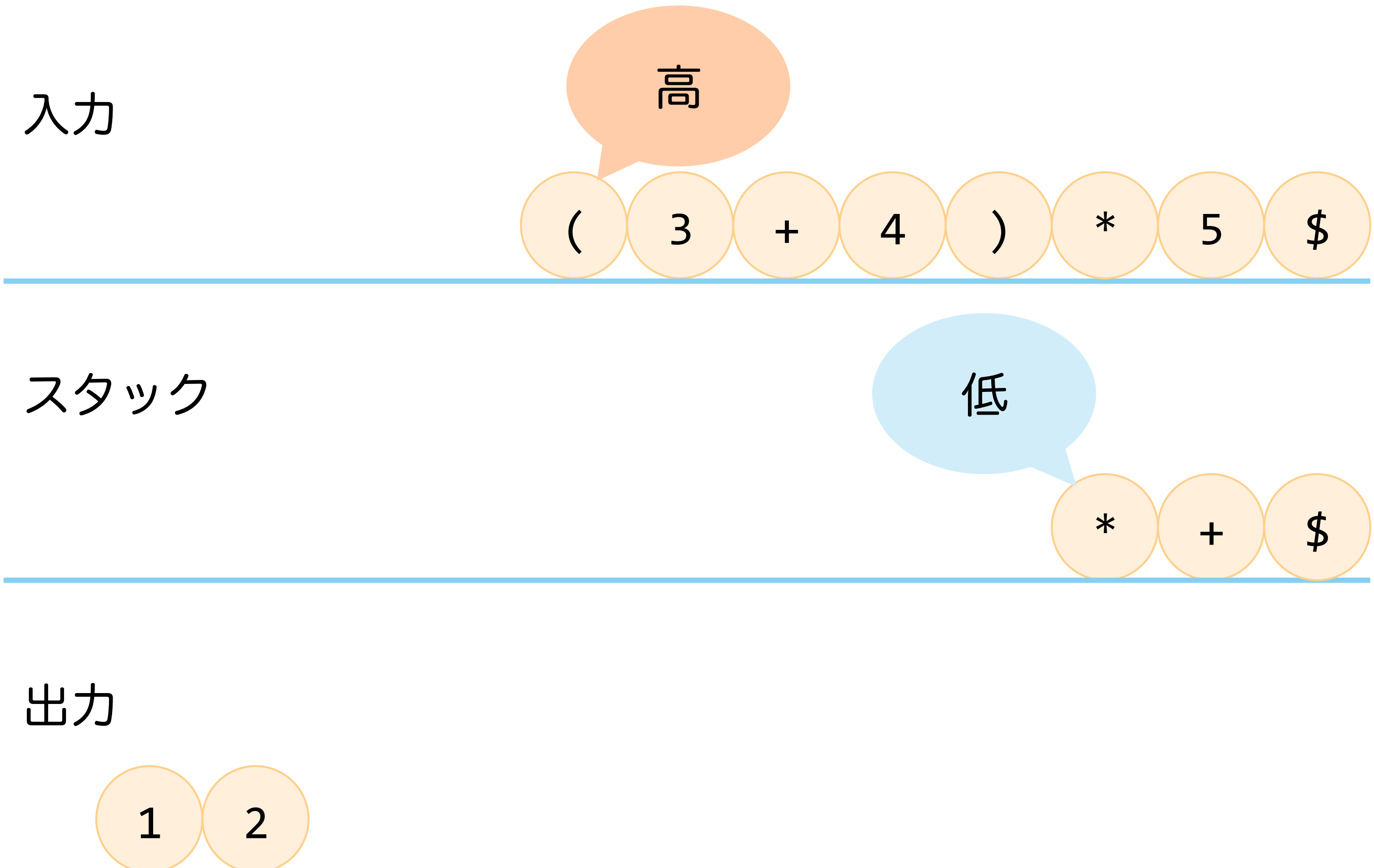
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



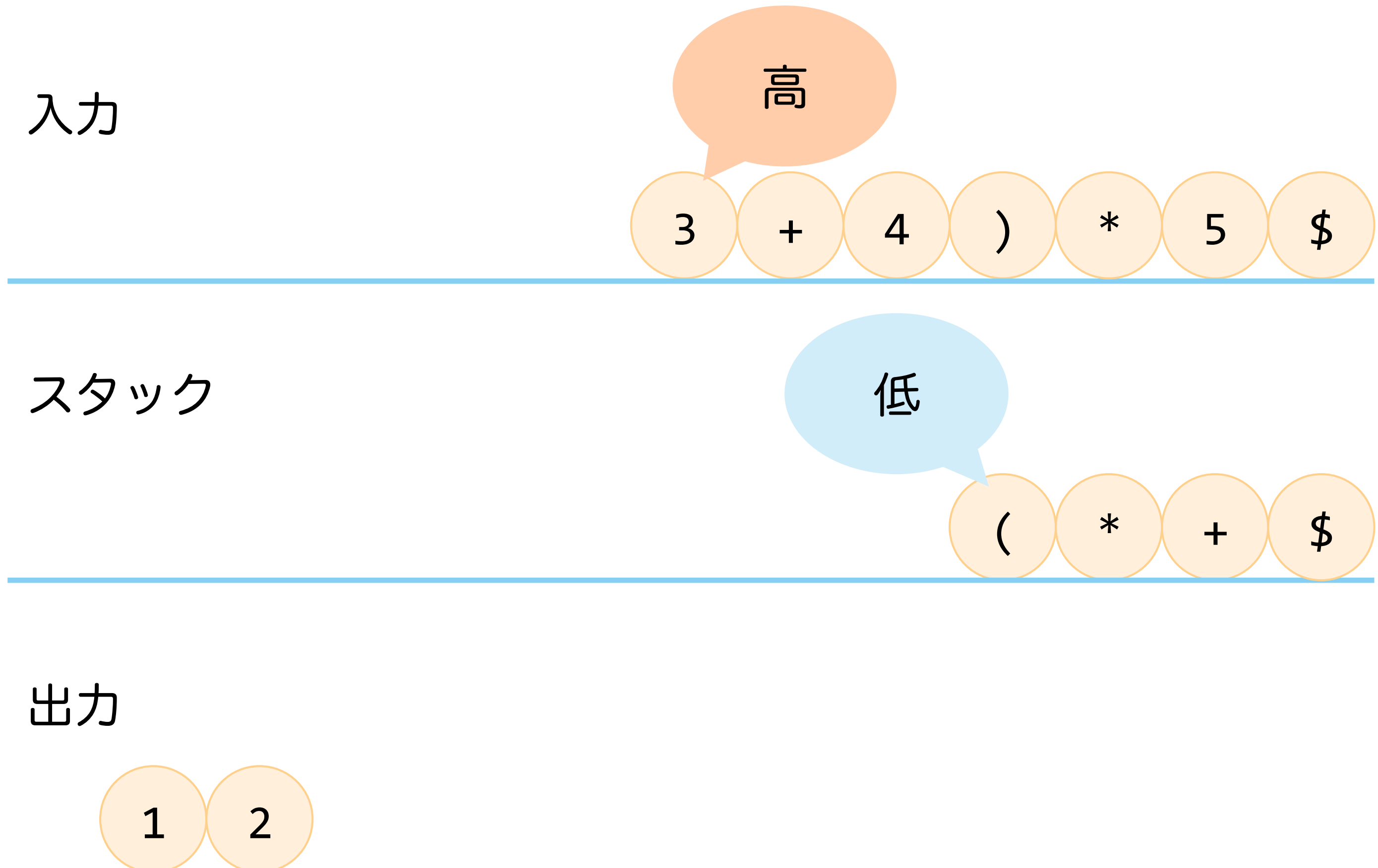
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



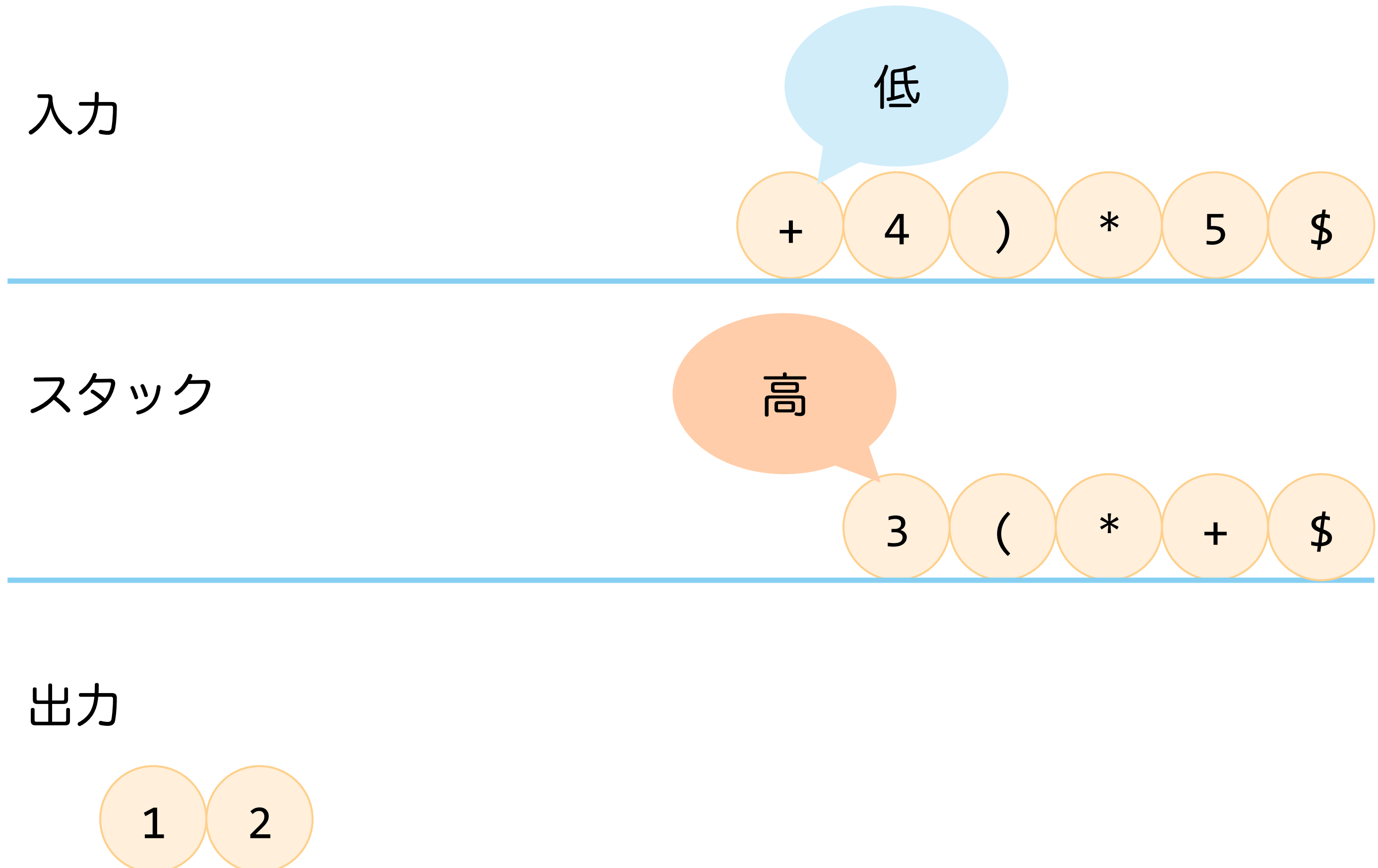
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



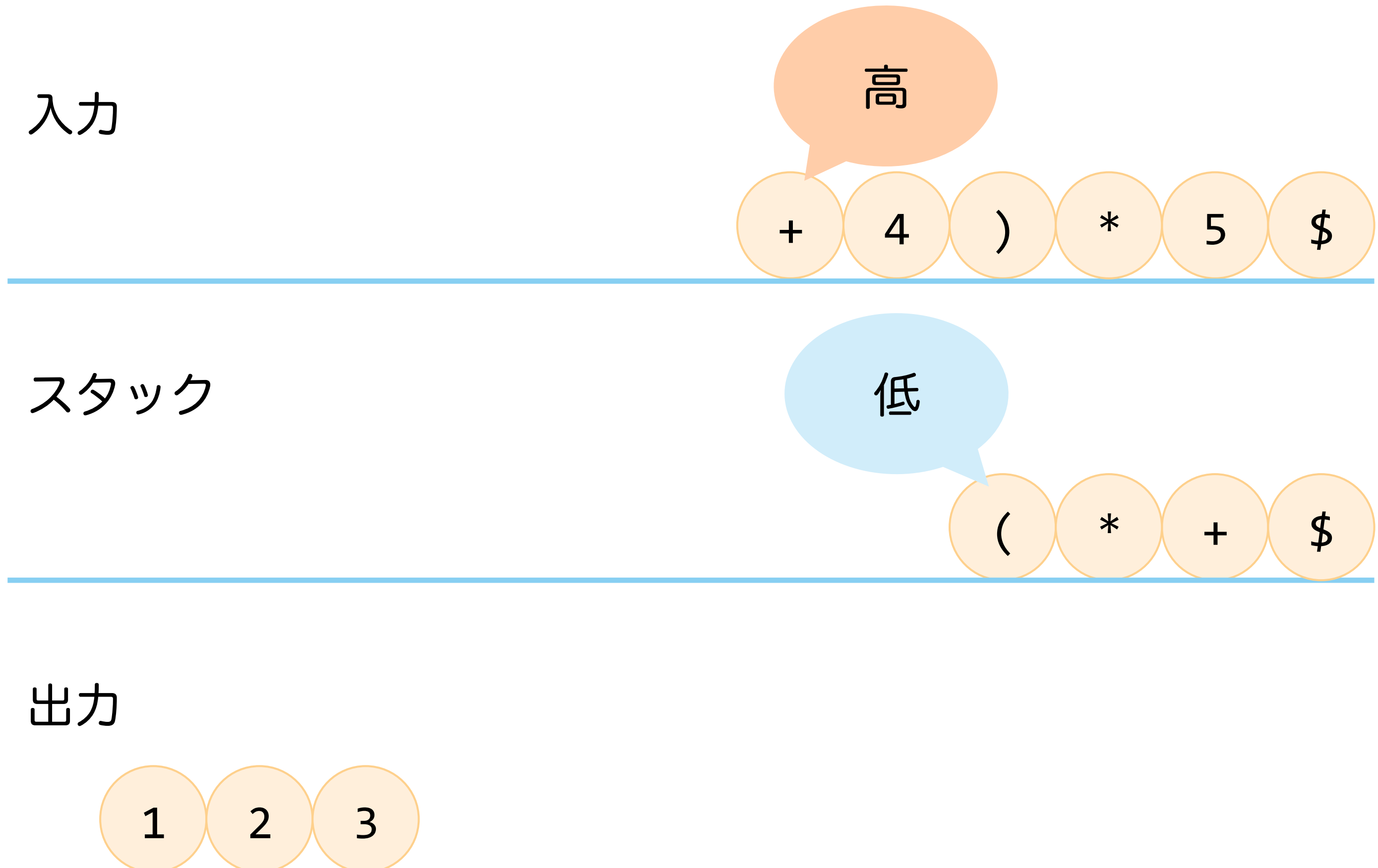
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



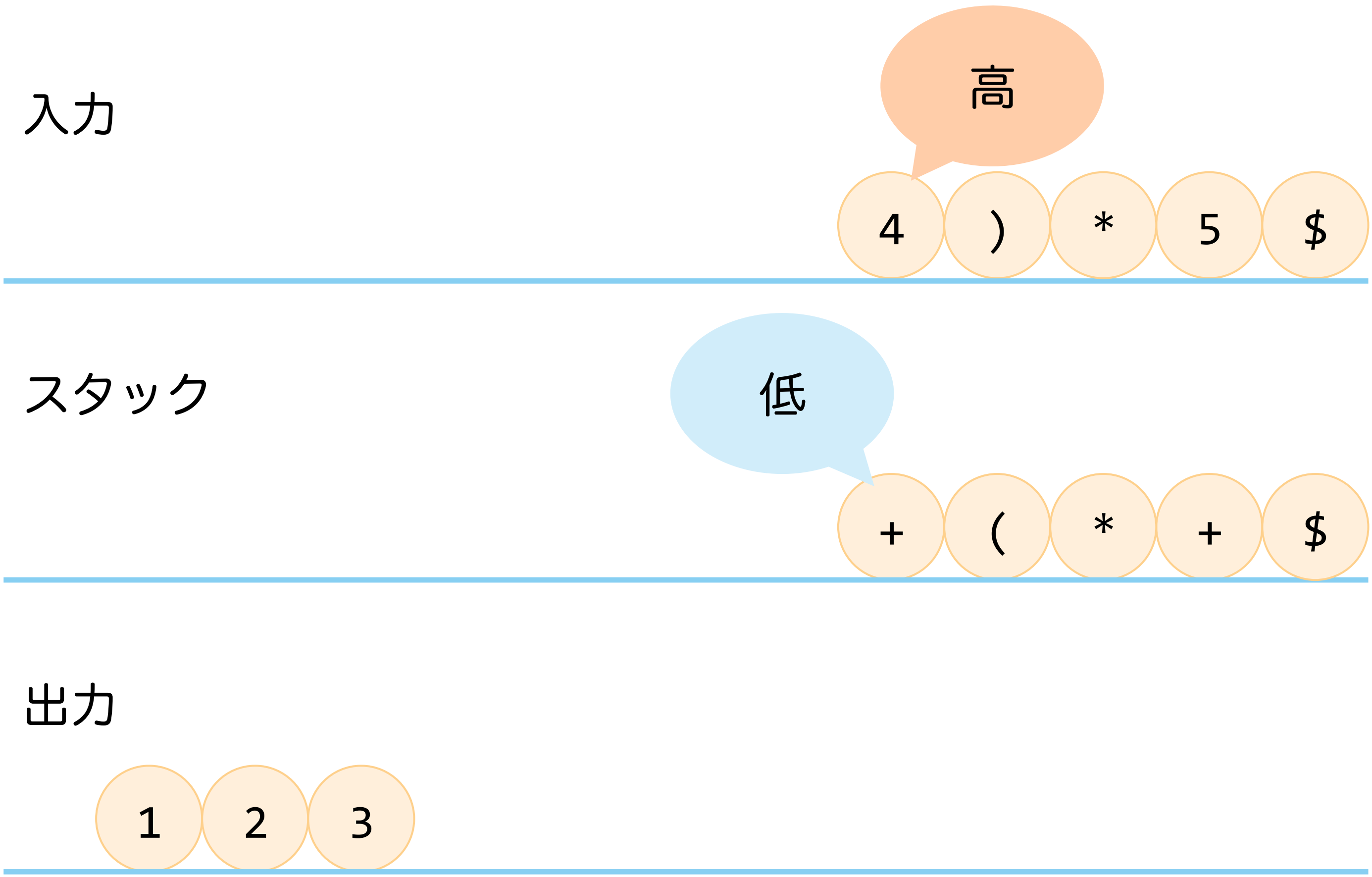
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



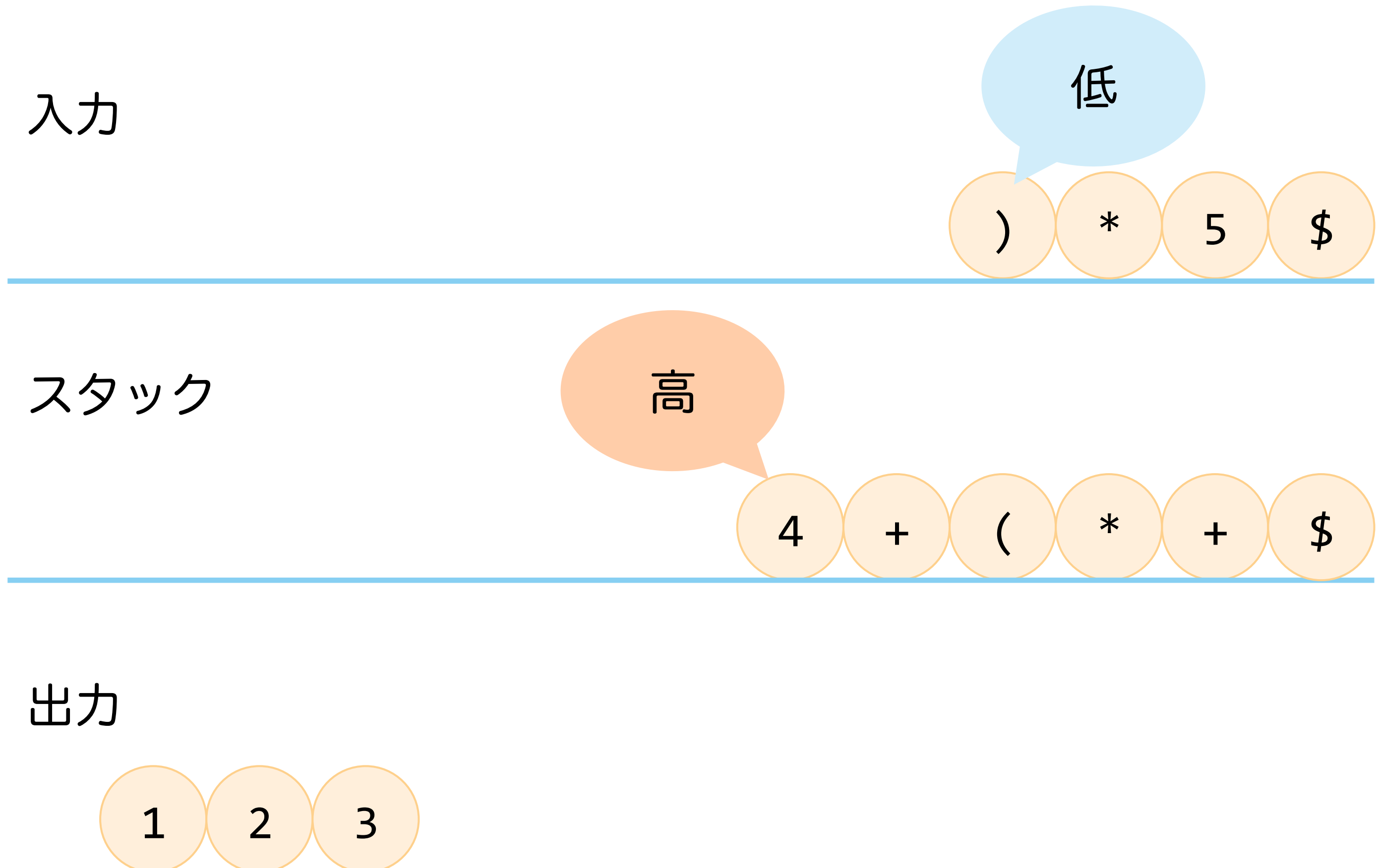
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



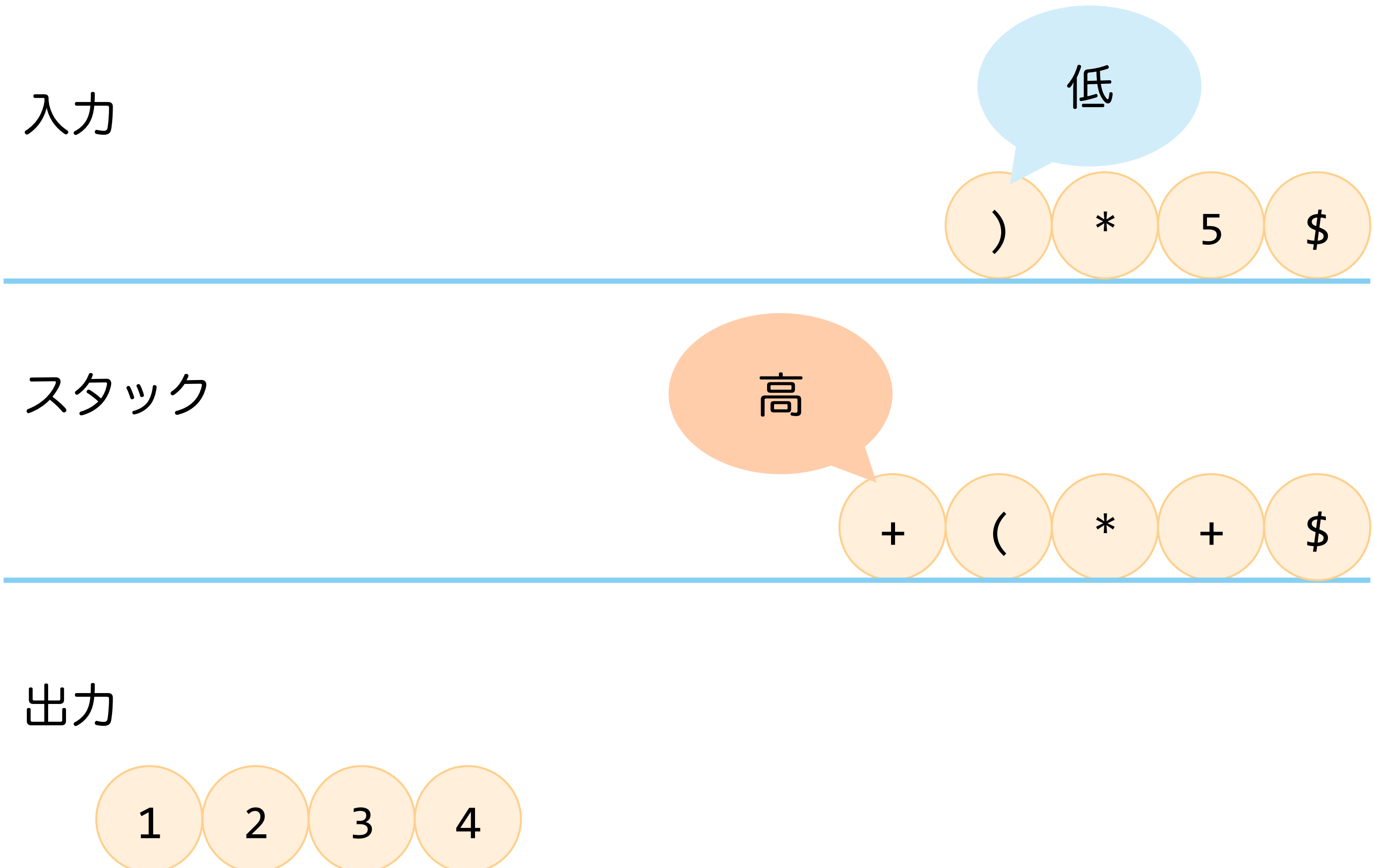
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



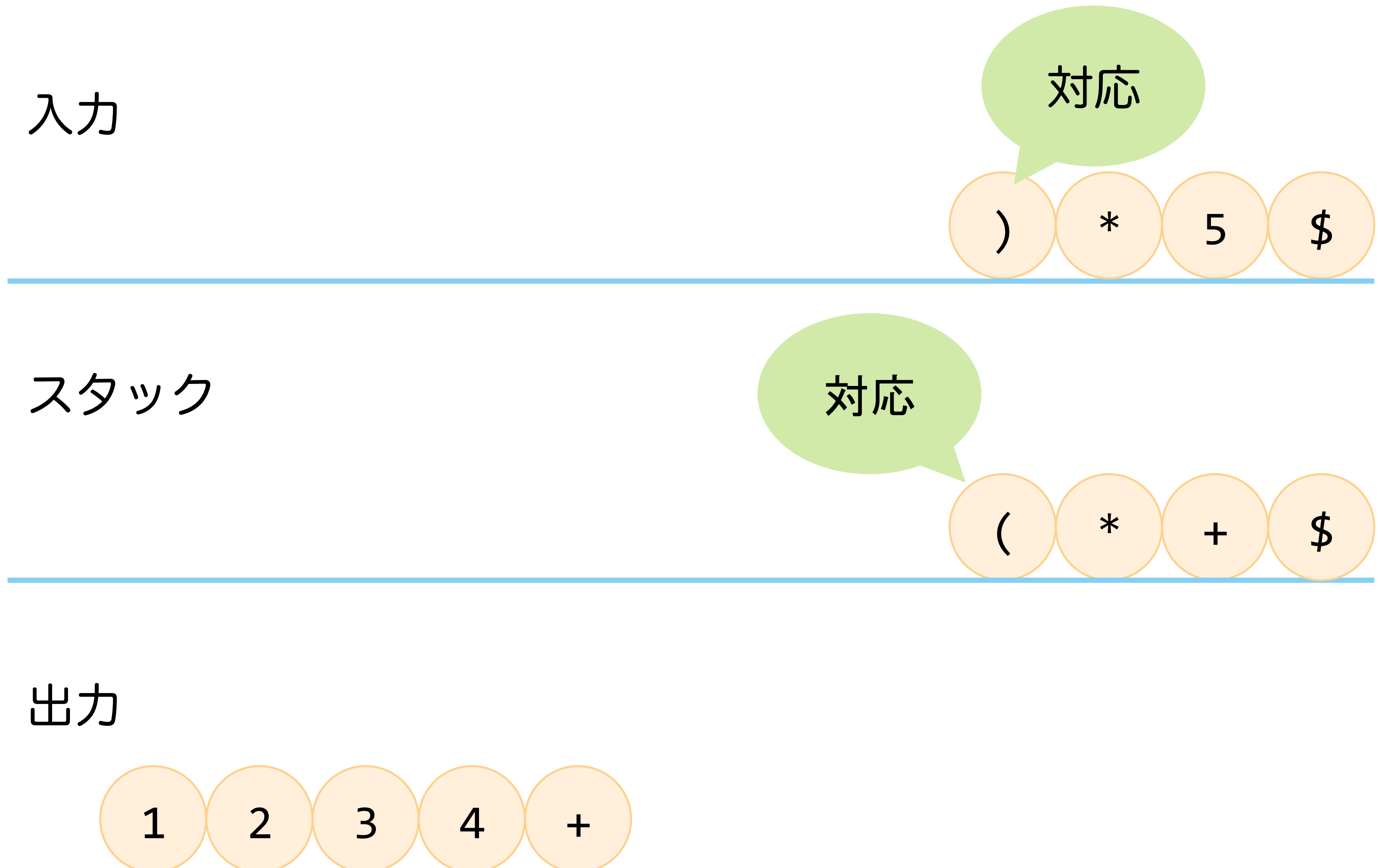
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



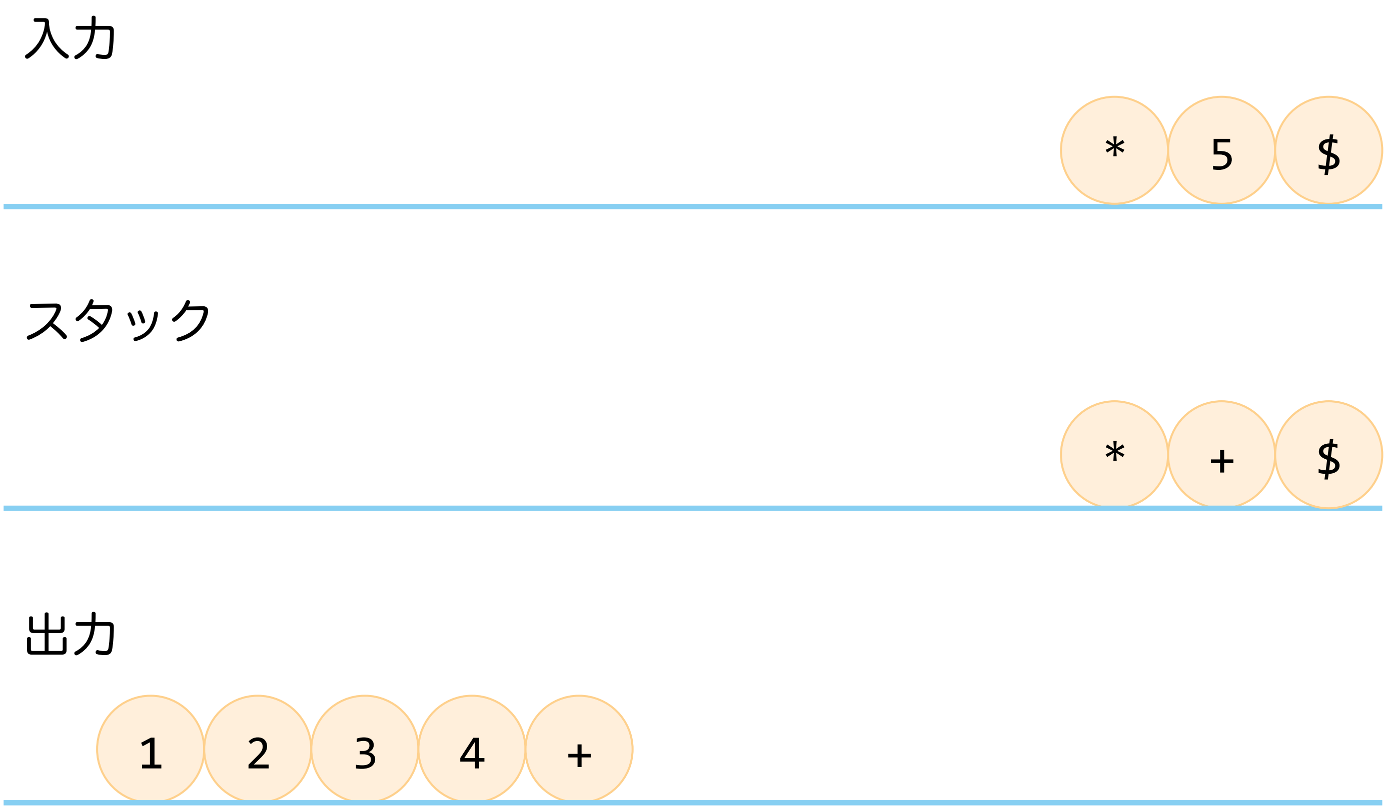
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



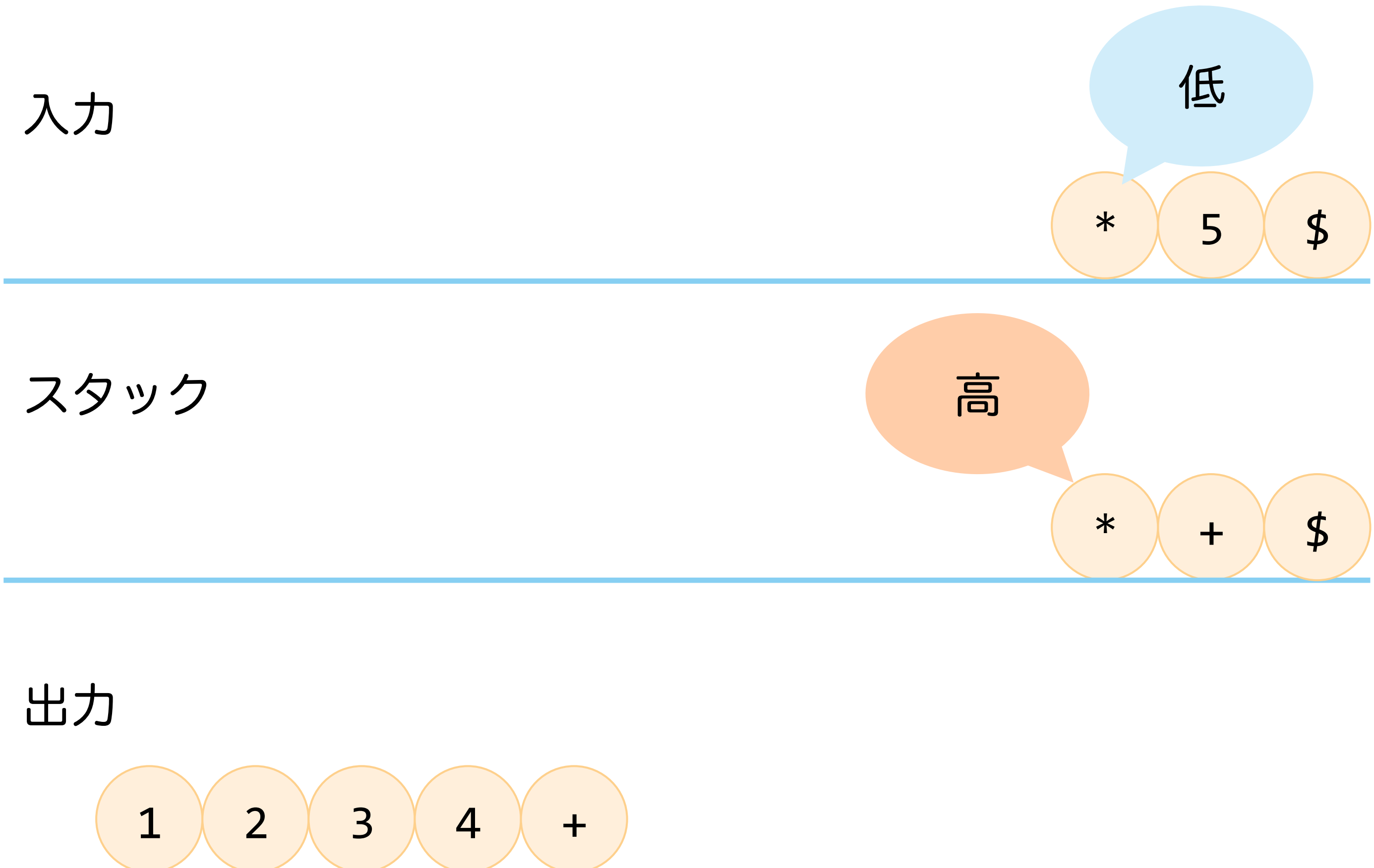
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



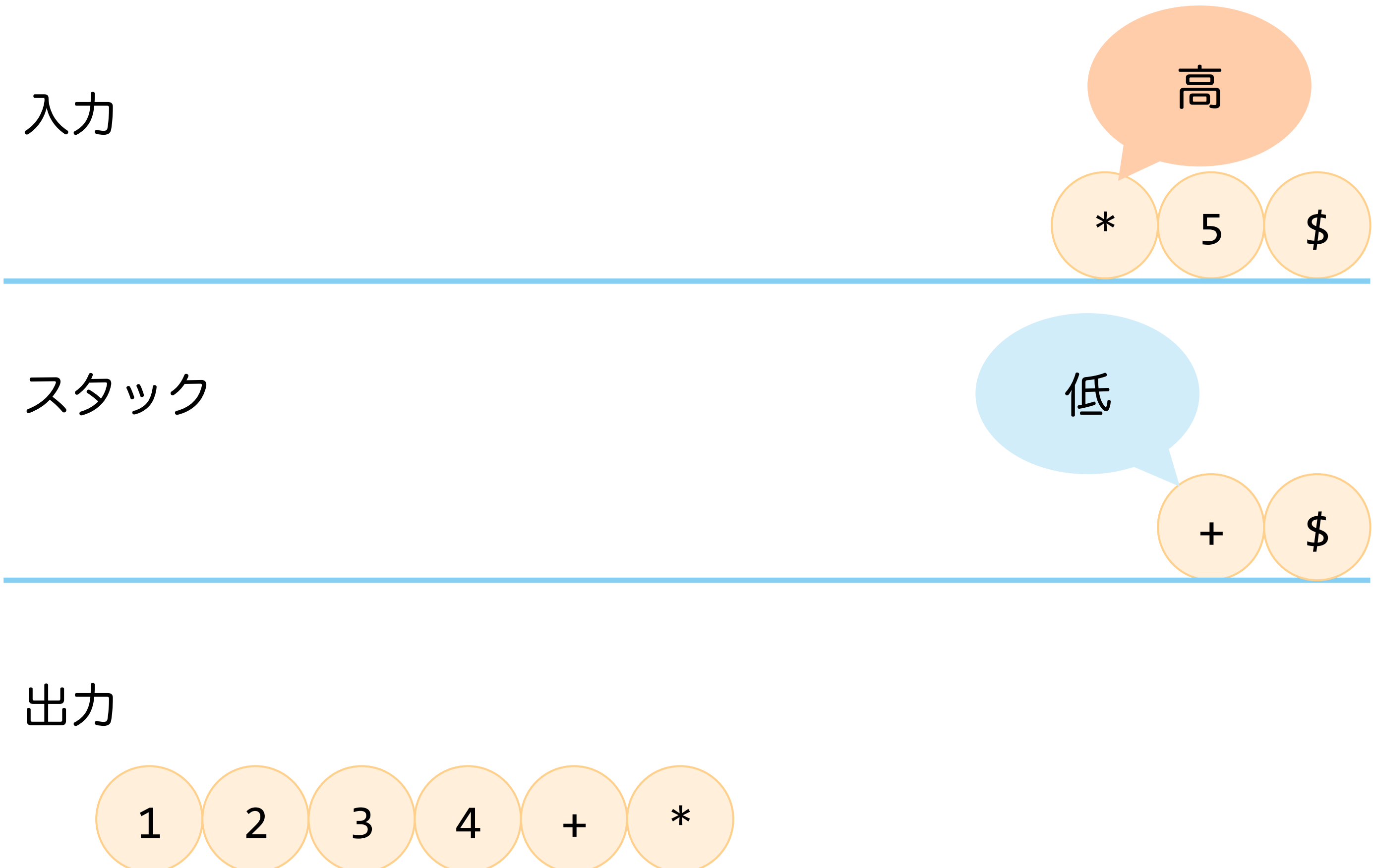
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



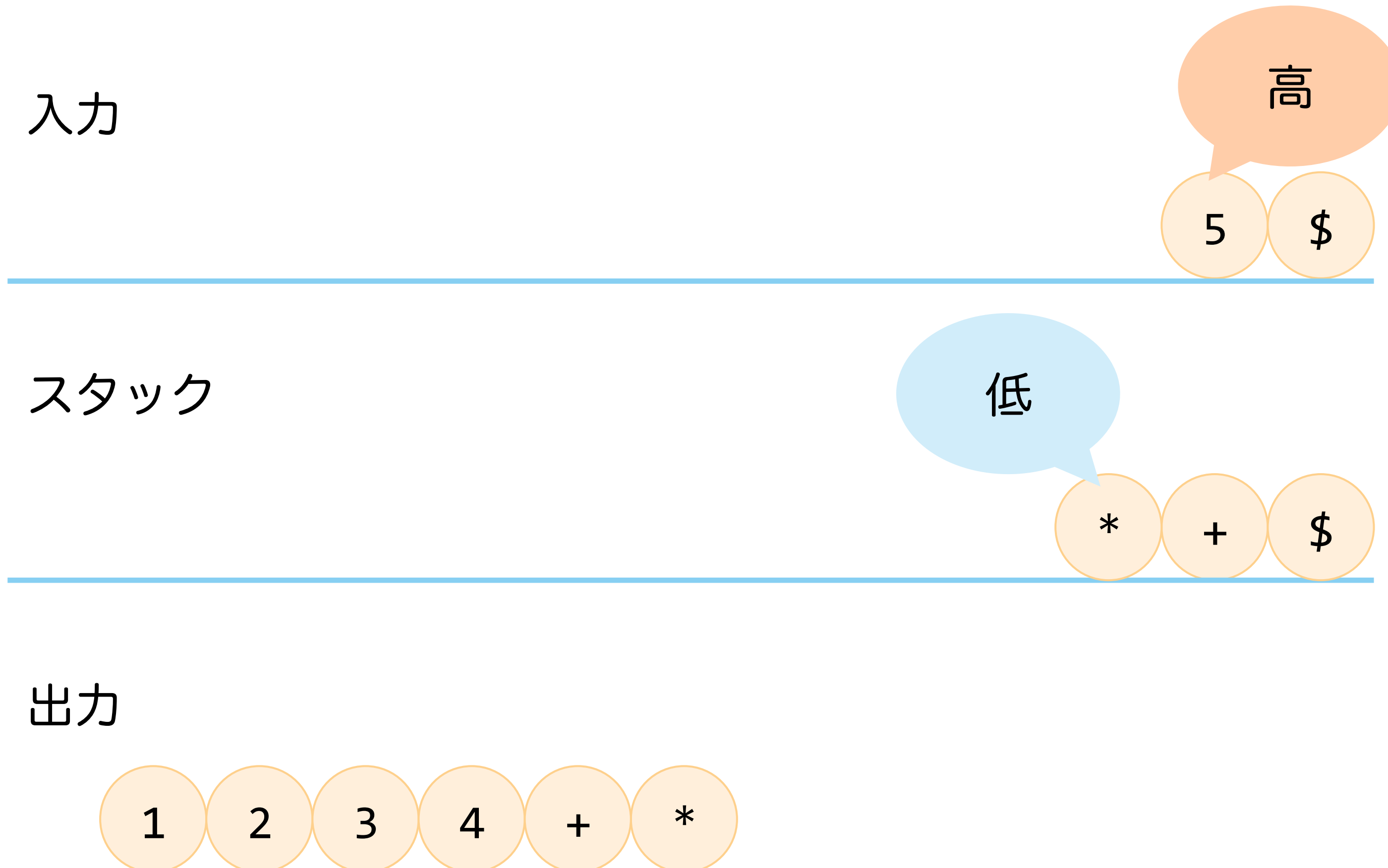
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



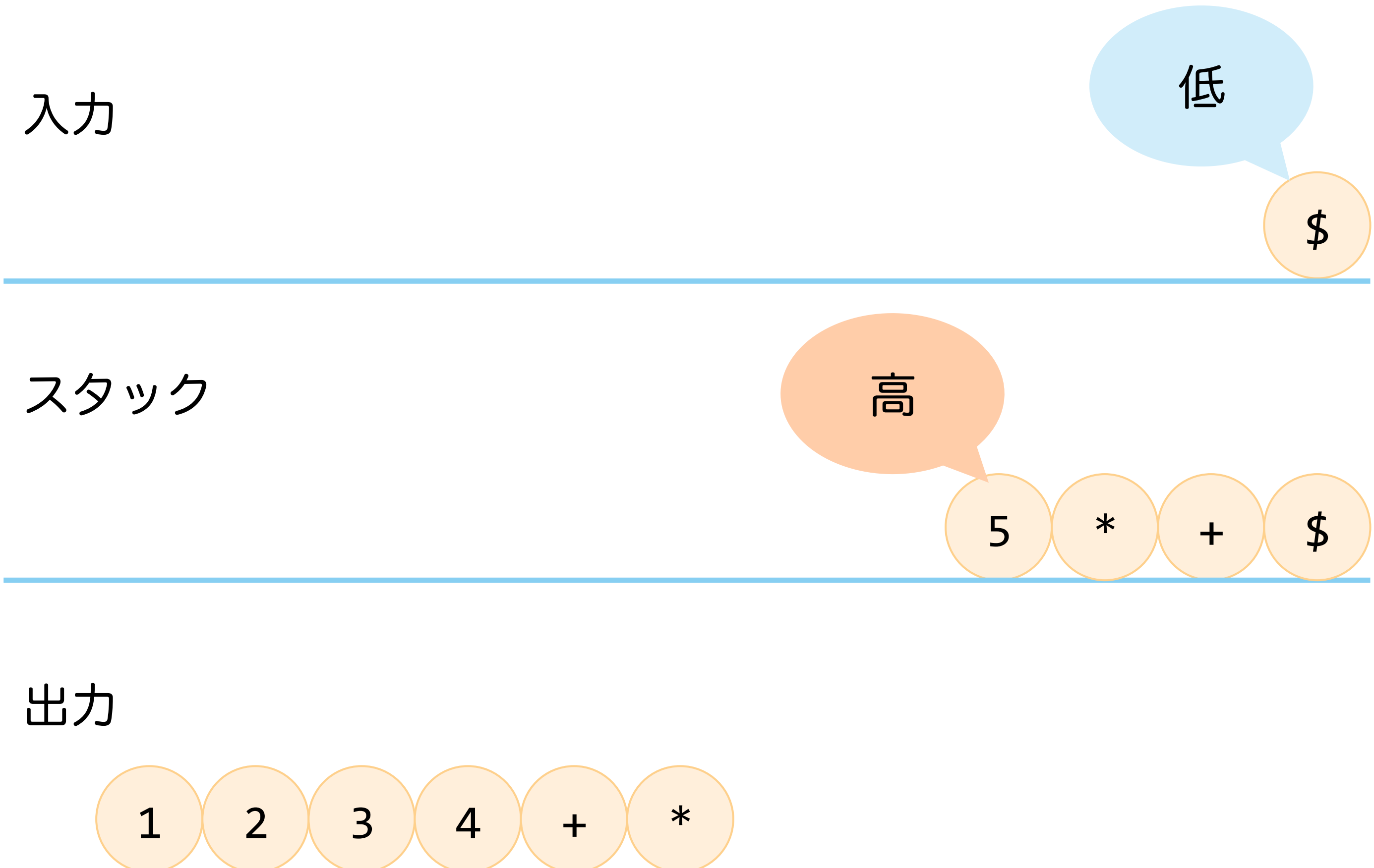
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



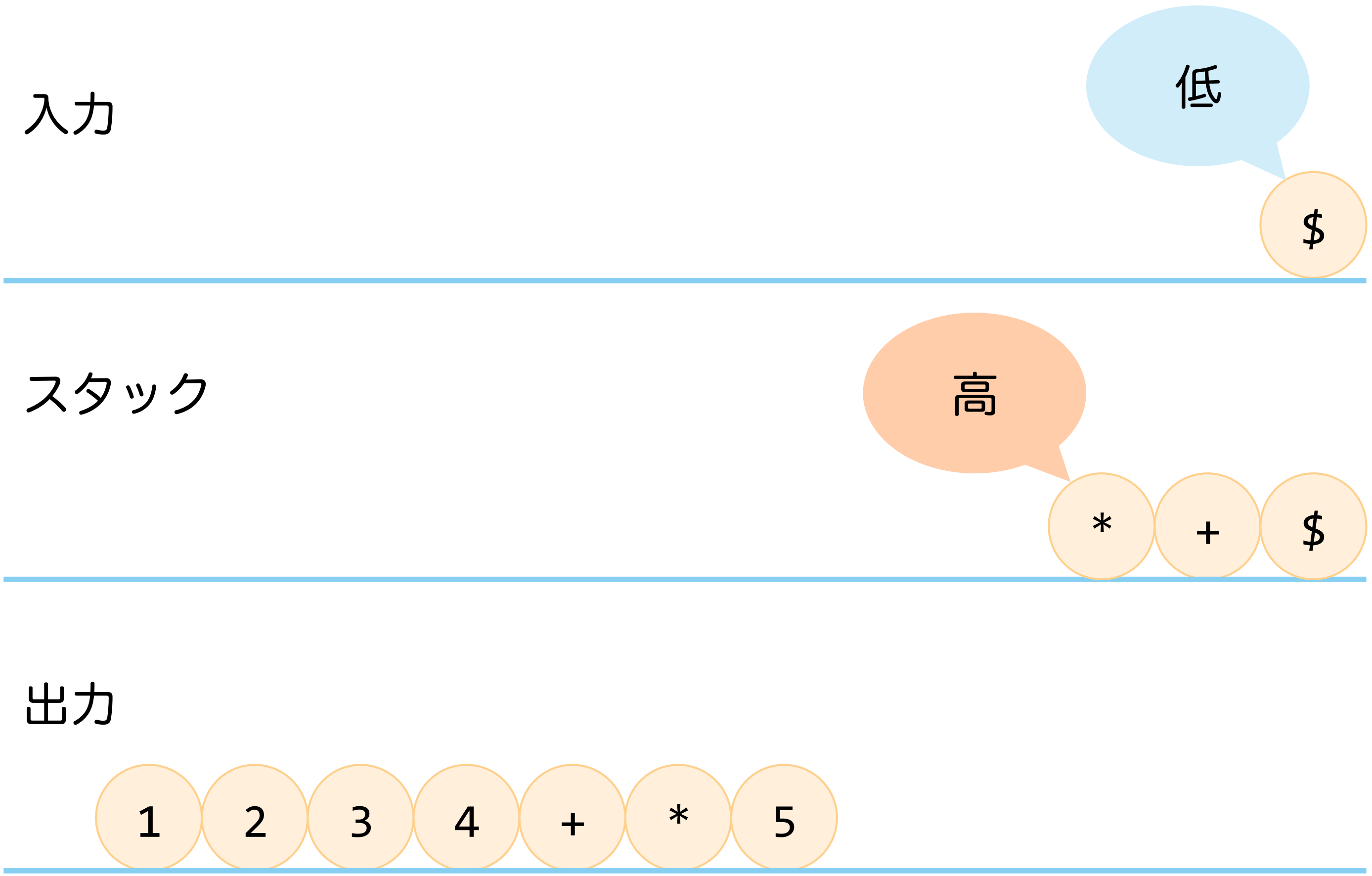
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



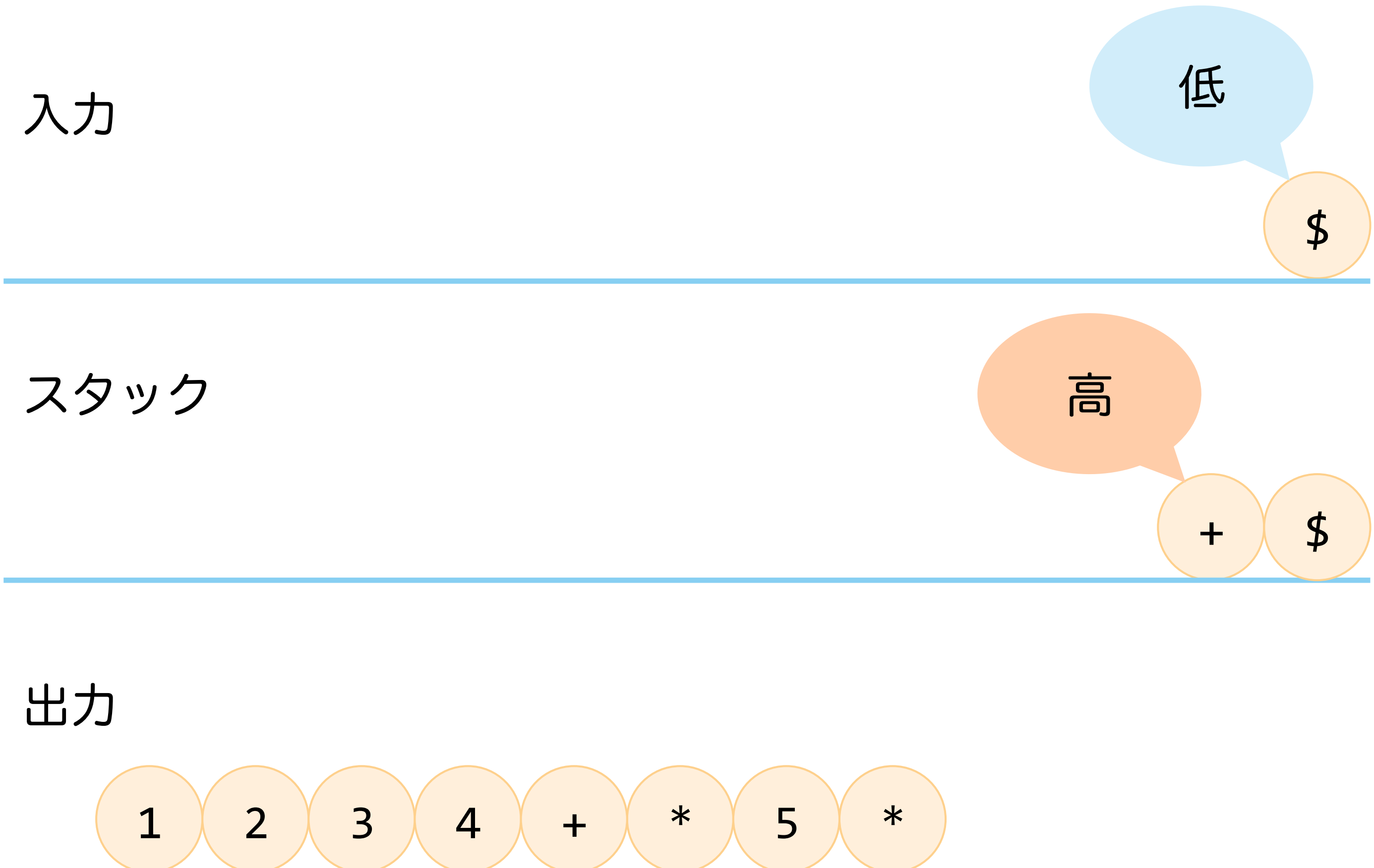
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



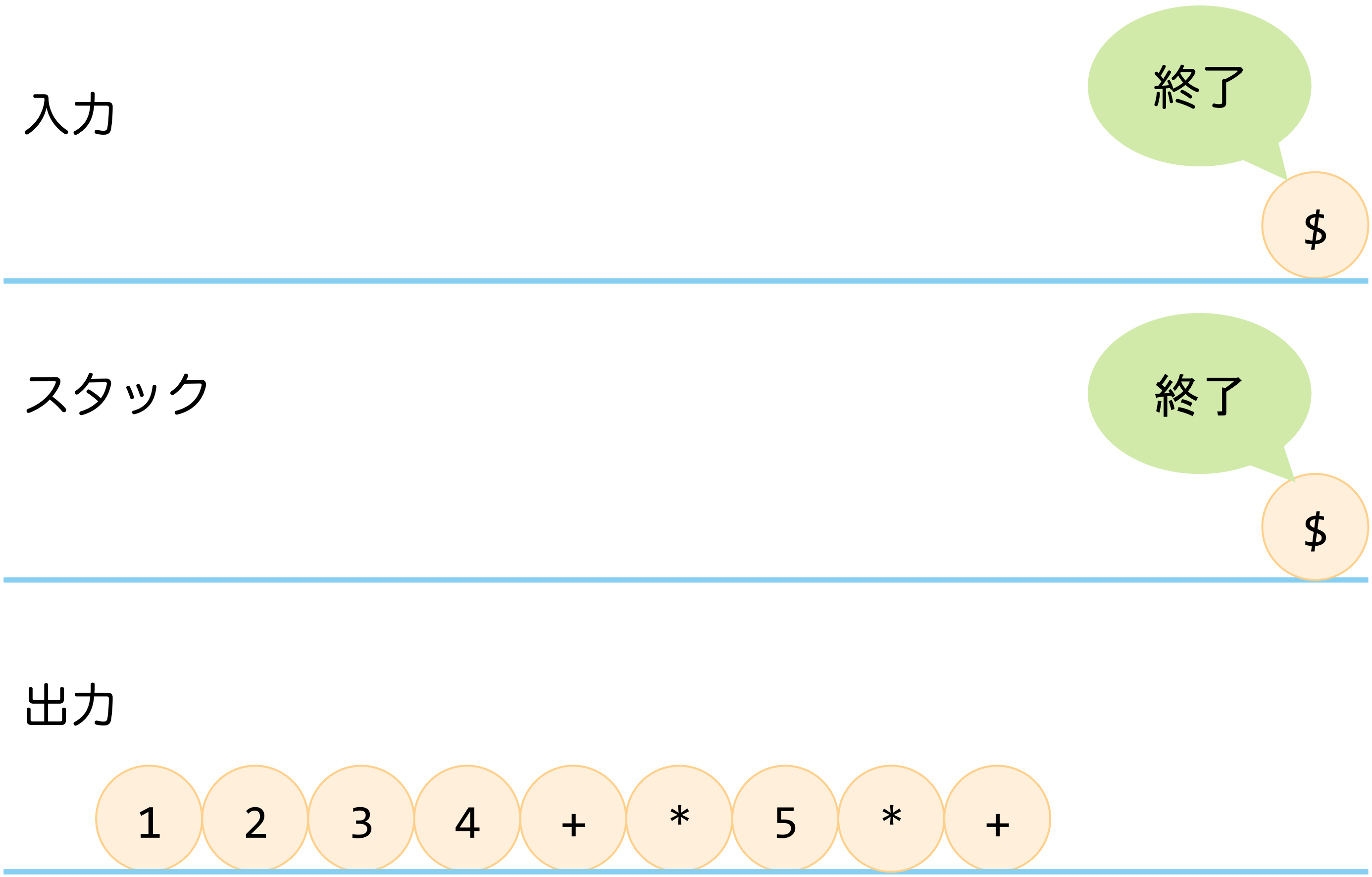
	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



	Num	+	*	(	)	$
input	7	2	4	6	1	0
stack	7	3	5	1		0



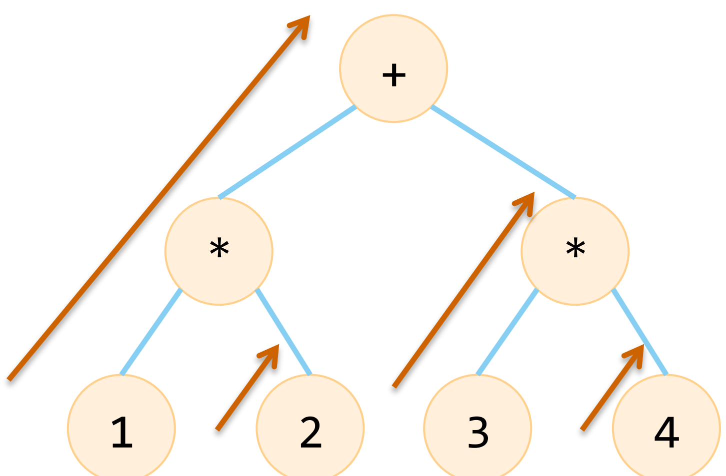
Shunting-yard algorithm [Dijkstra 1961]

- 2つの順位関数を利用する手法を紹介した
  - Dijkstra のオリジナルの論文では順位関数は1つ
- 順位行列を利用する方法もある
  - こちらの方がより強力
- 能力
  - 計算量 : O(n)
  - 言語 : Operator-precedence language
  - 文法 : Operator-precedence grammar

Pratt parsing [Pratt 1973]

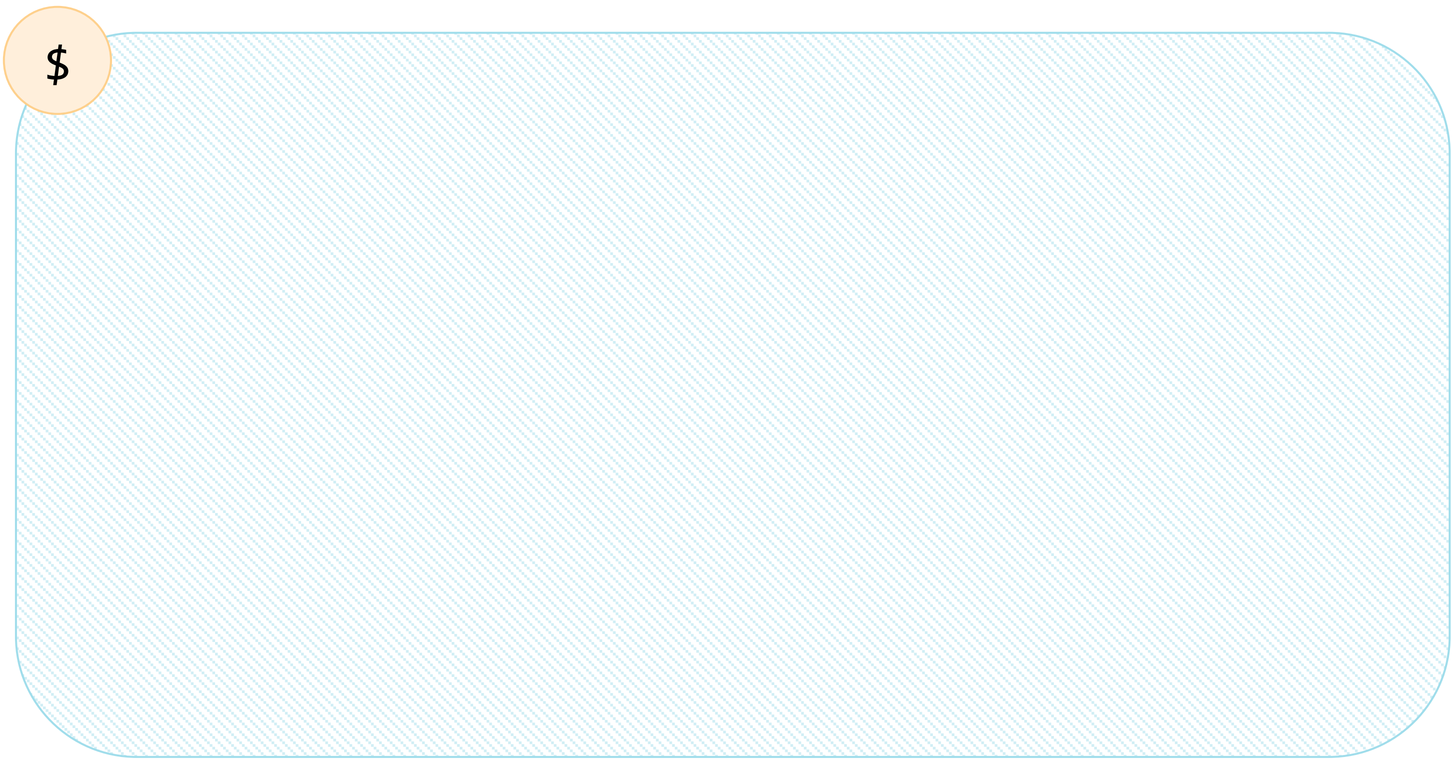
アイデア

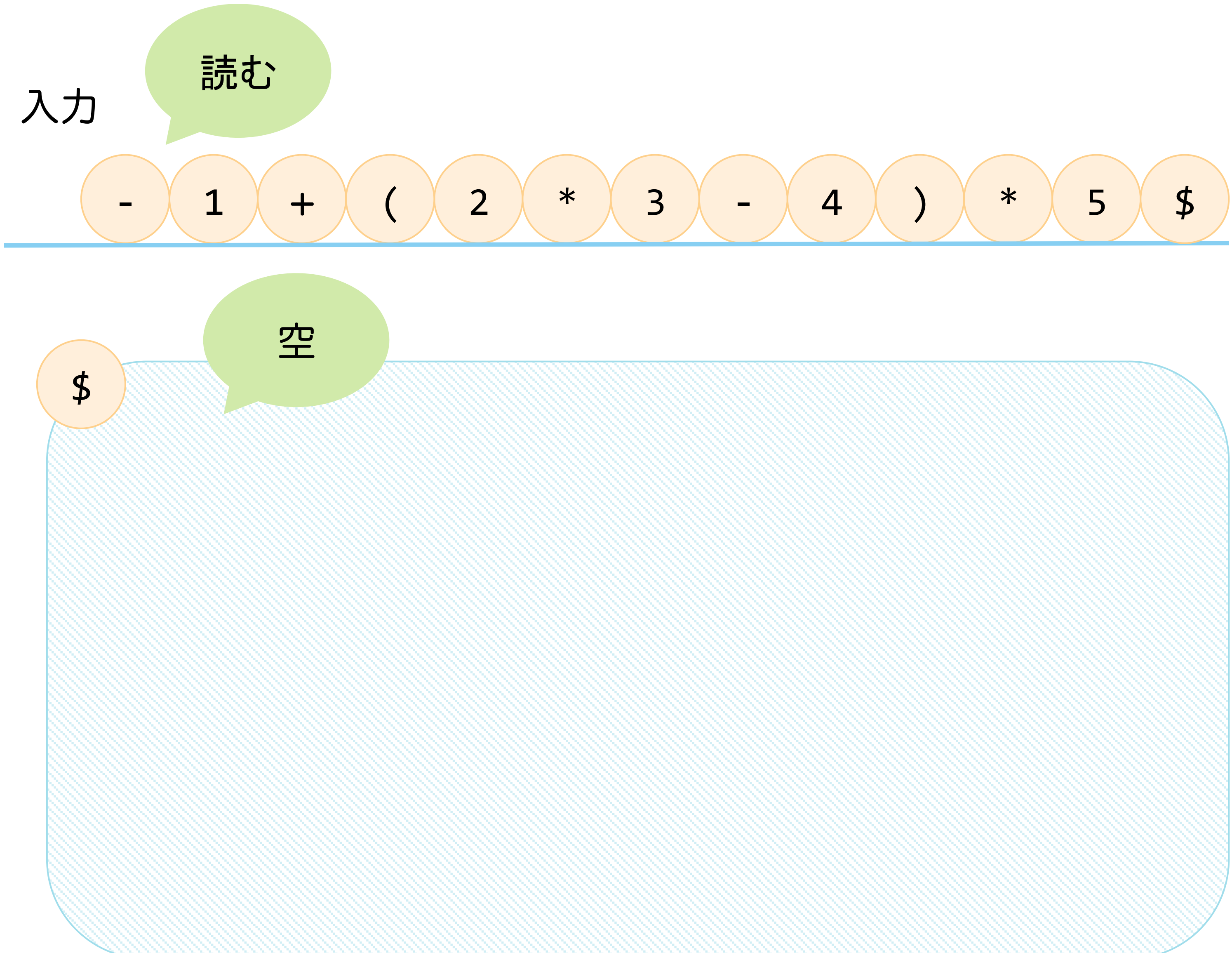
- 構文木の左下から上に成長させる
- 構文木の右の枝は再帰的に作る



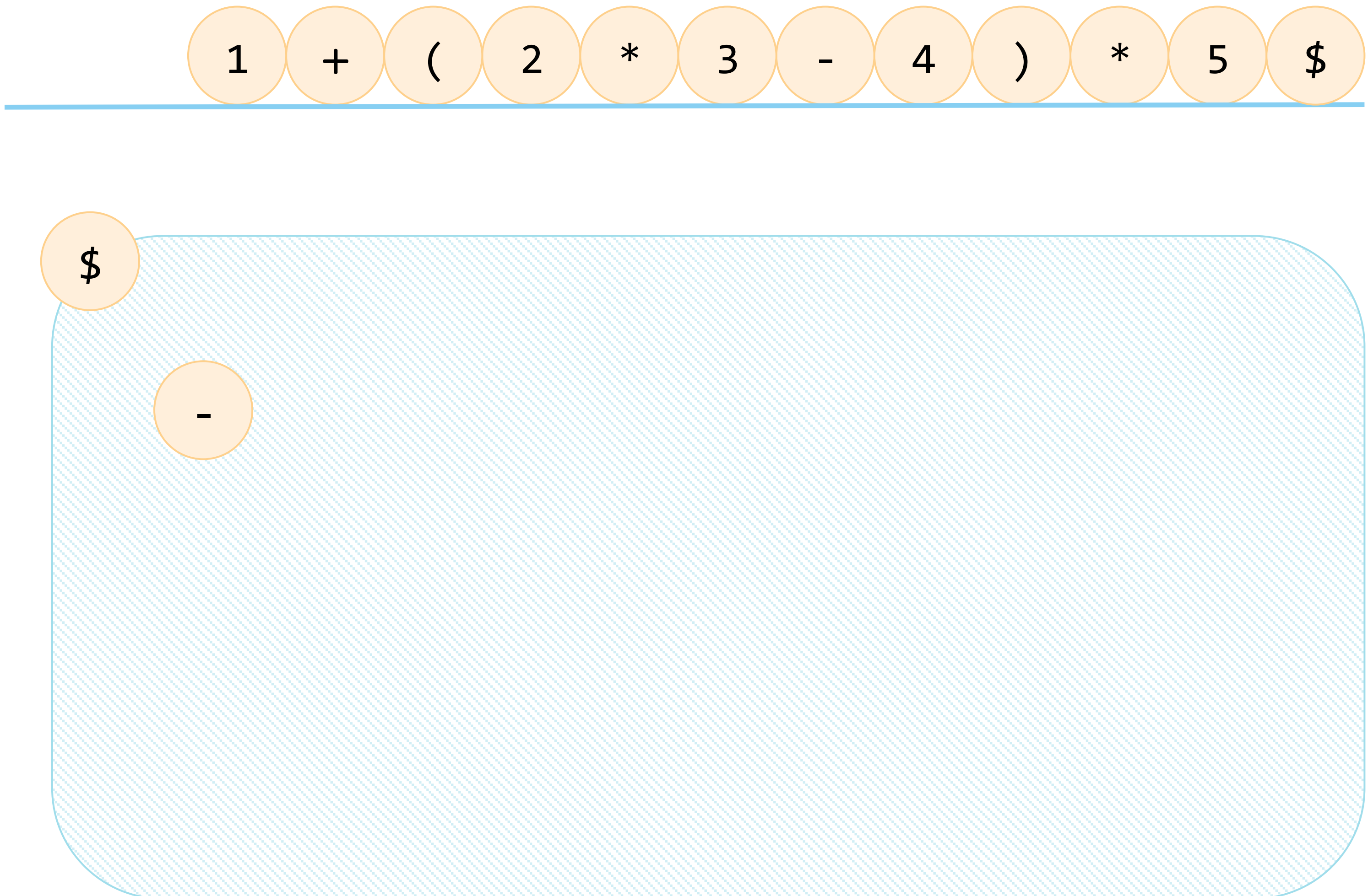
入力

- 1 + (2 * 3 - 4) * 5 $

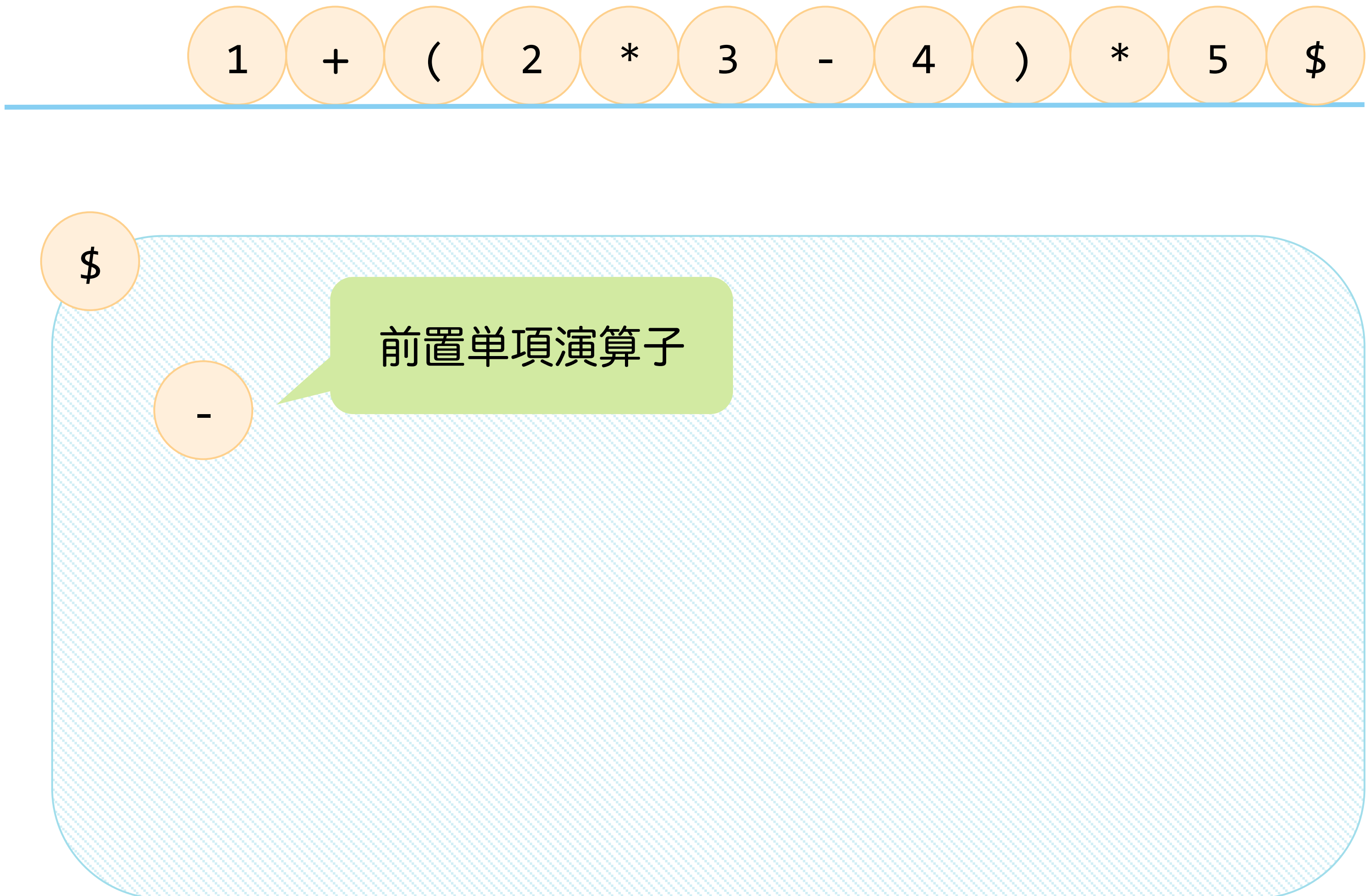



読む
入力
- 1 + (2 * 3 - 4) * 5 $
空
$

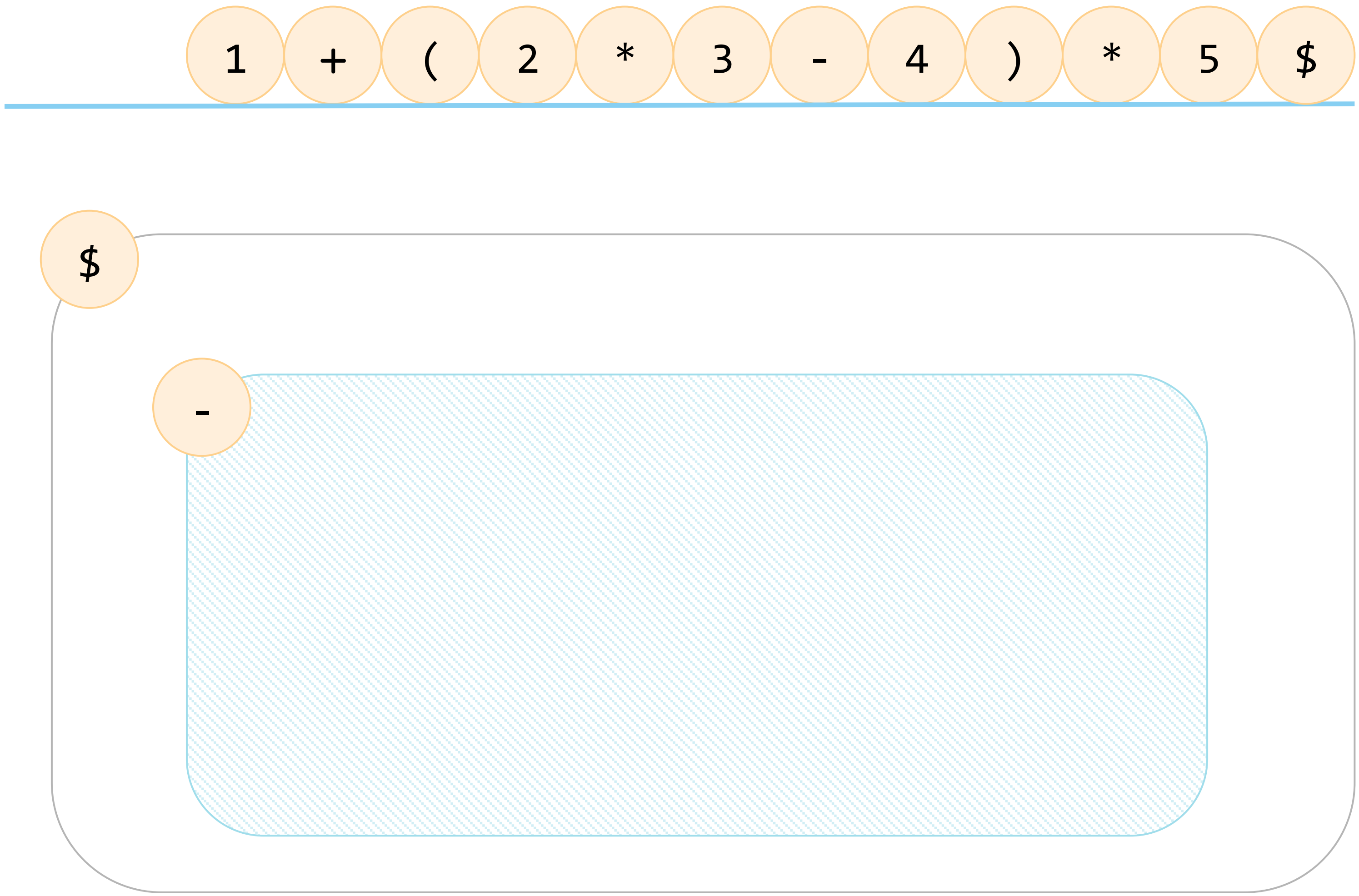
入力

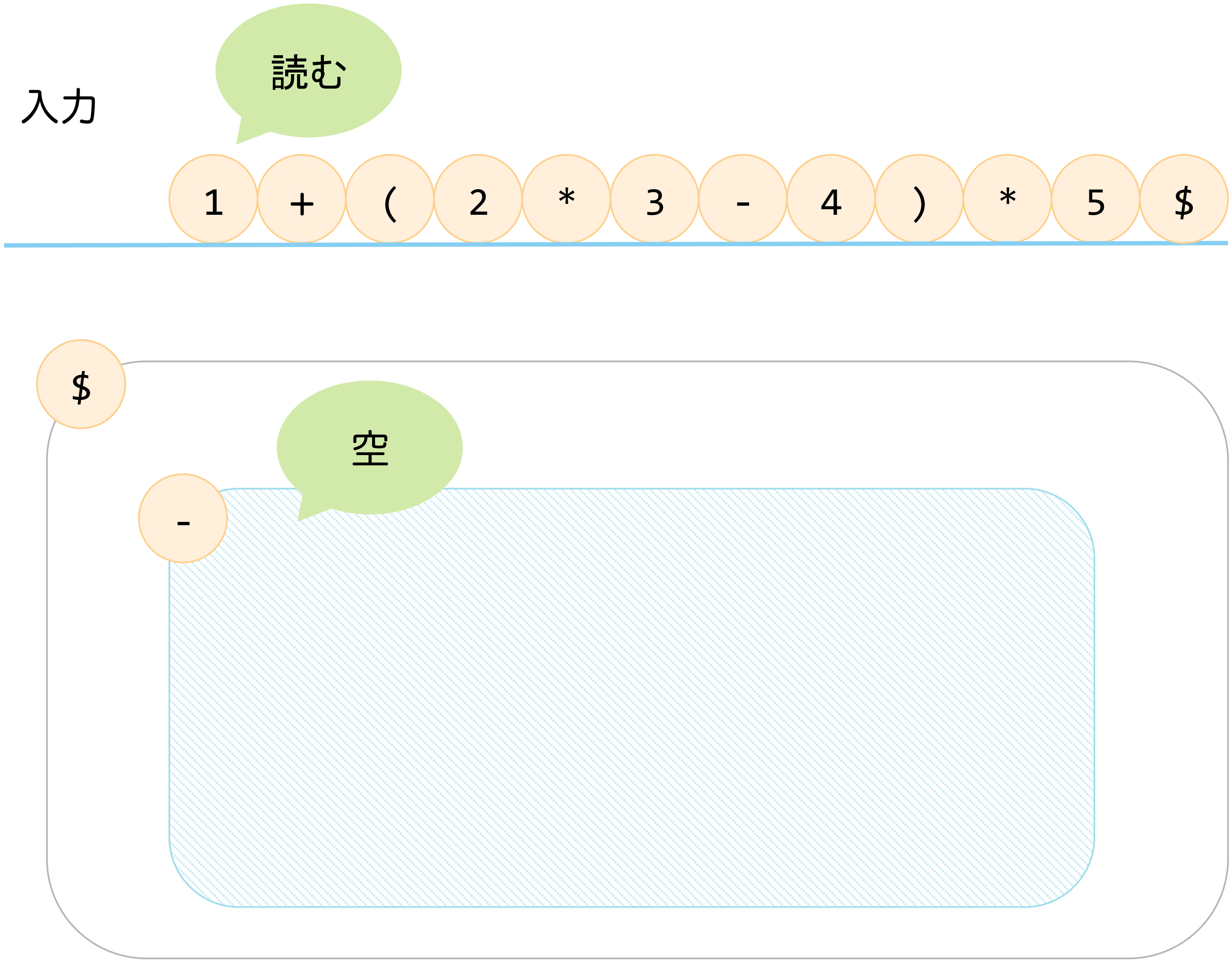


入力



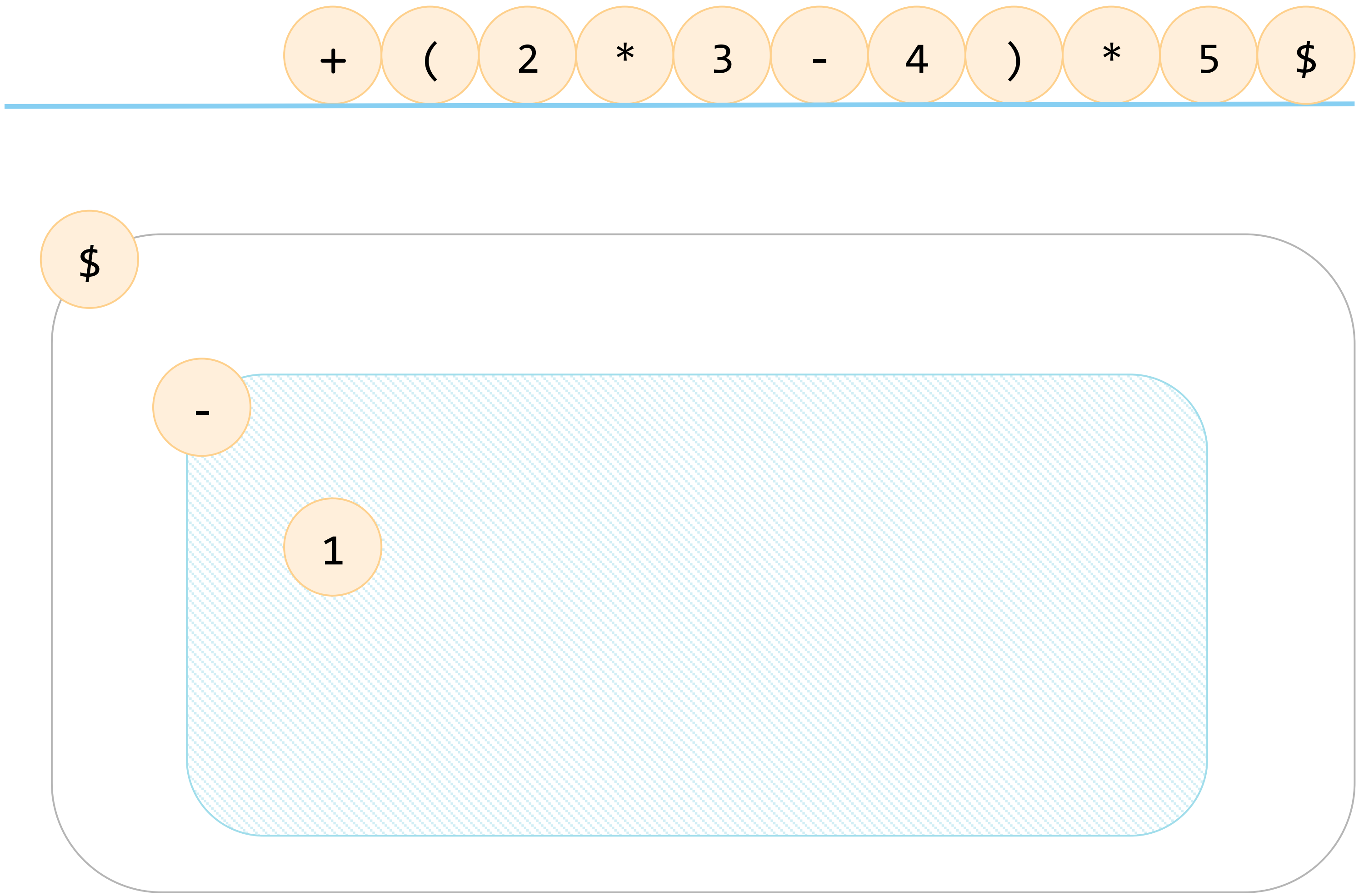
入力

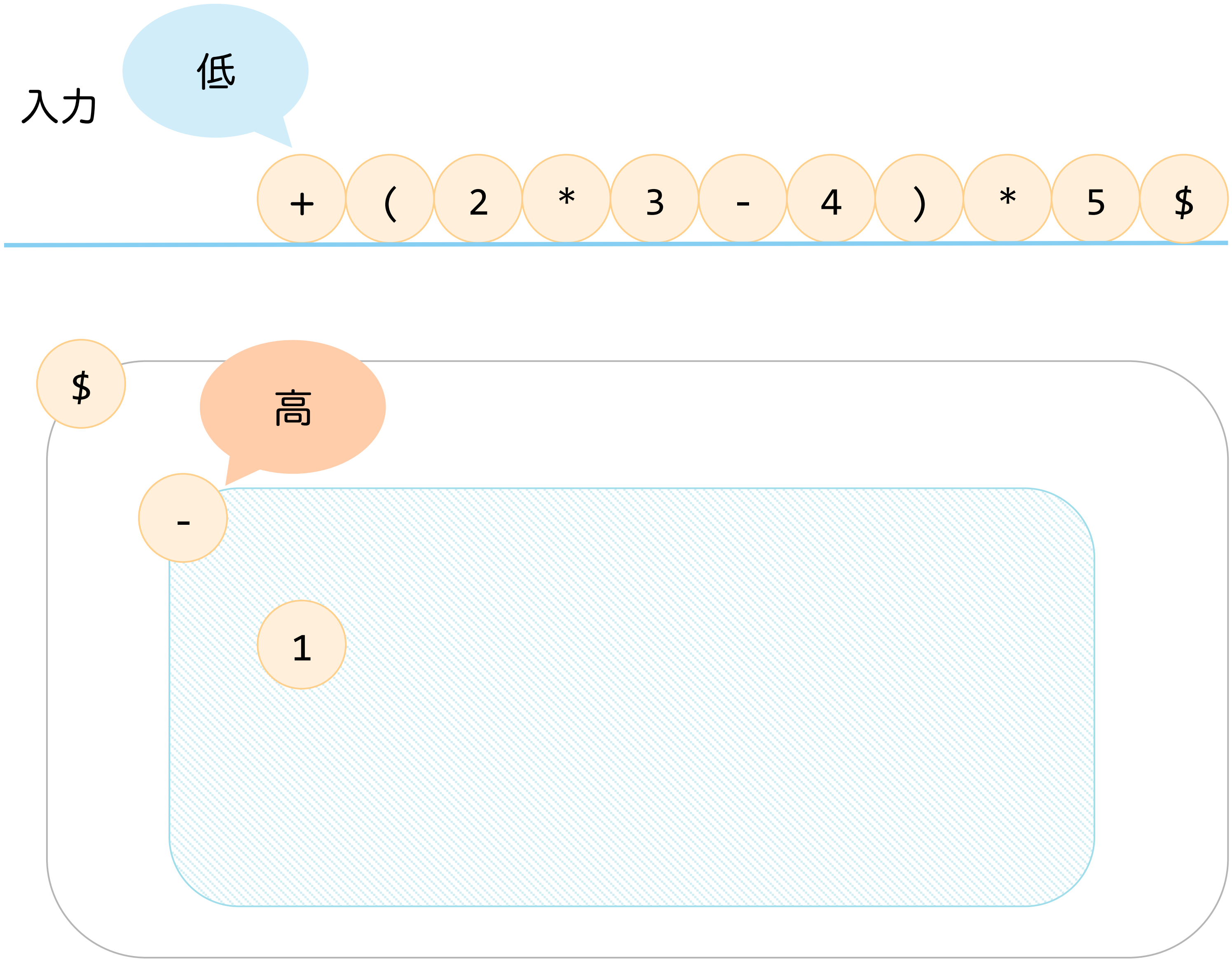




入力
読む
1
+
(
2
*
3
-
4
)
*
5
$
$
空
-

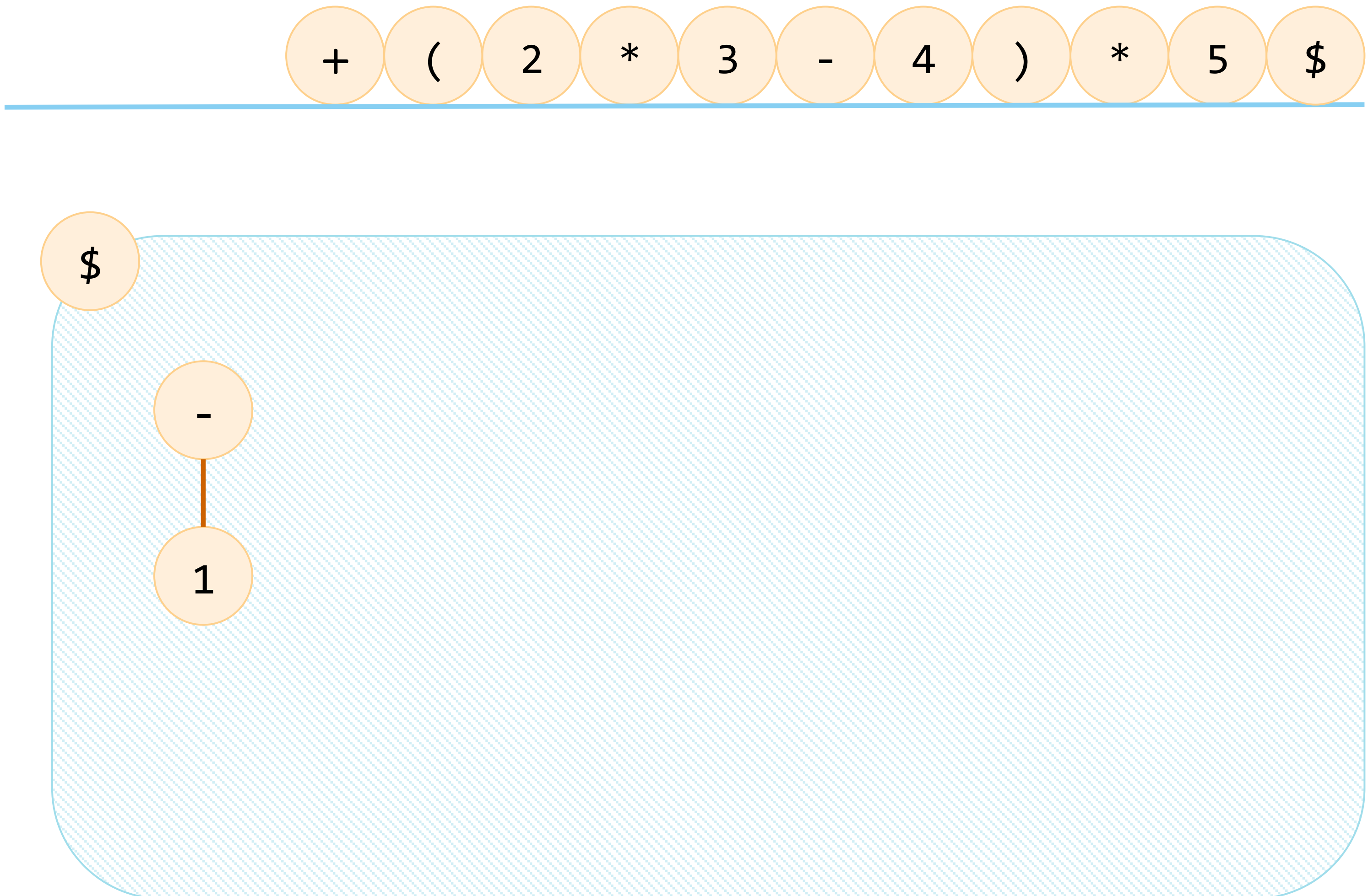
入力

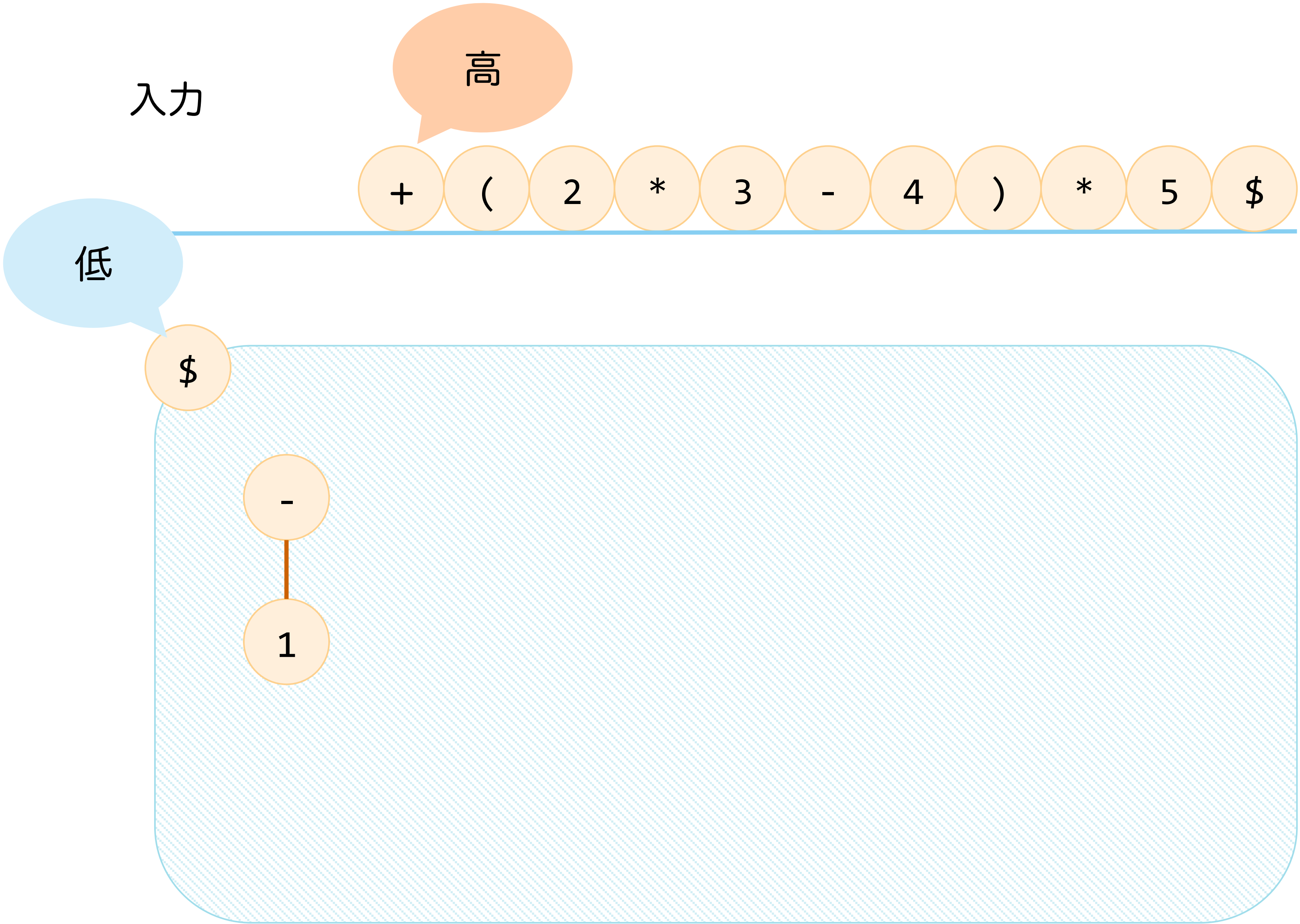




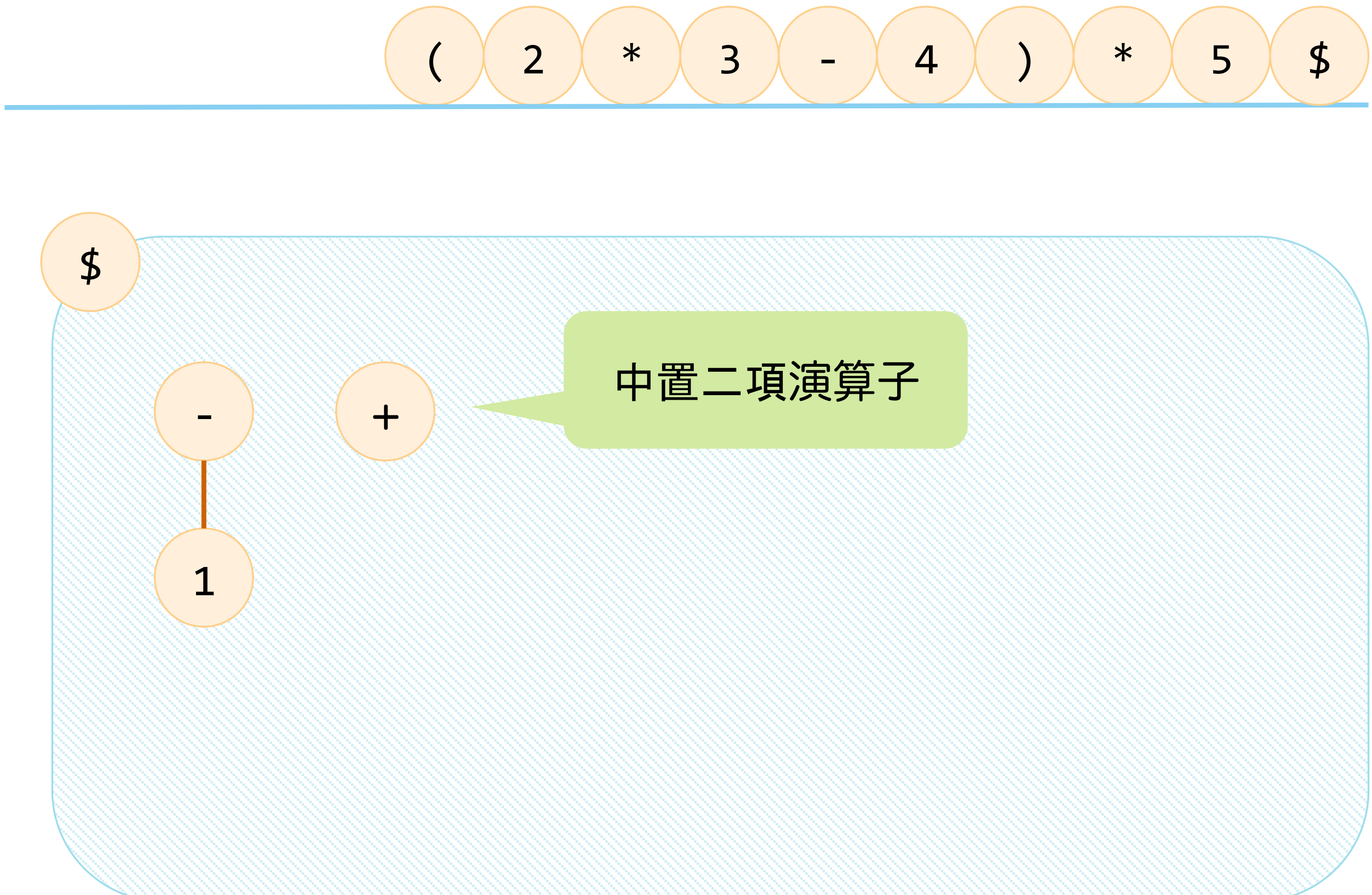
入力
低
+ (2 * 3 - 4) * 5 $
$
高
-
1

入力

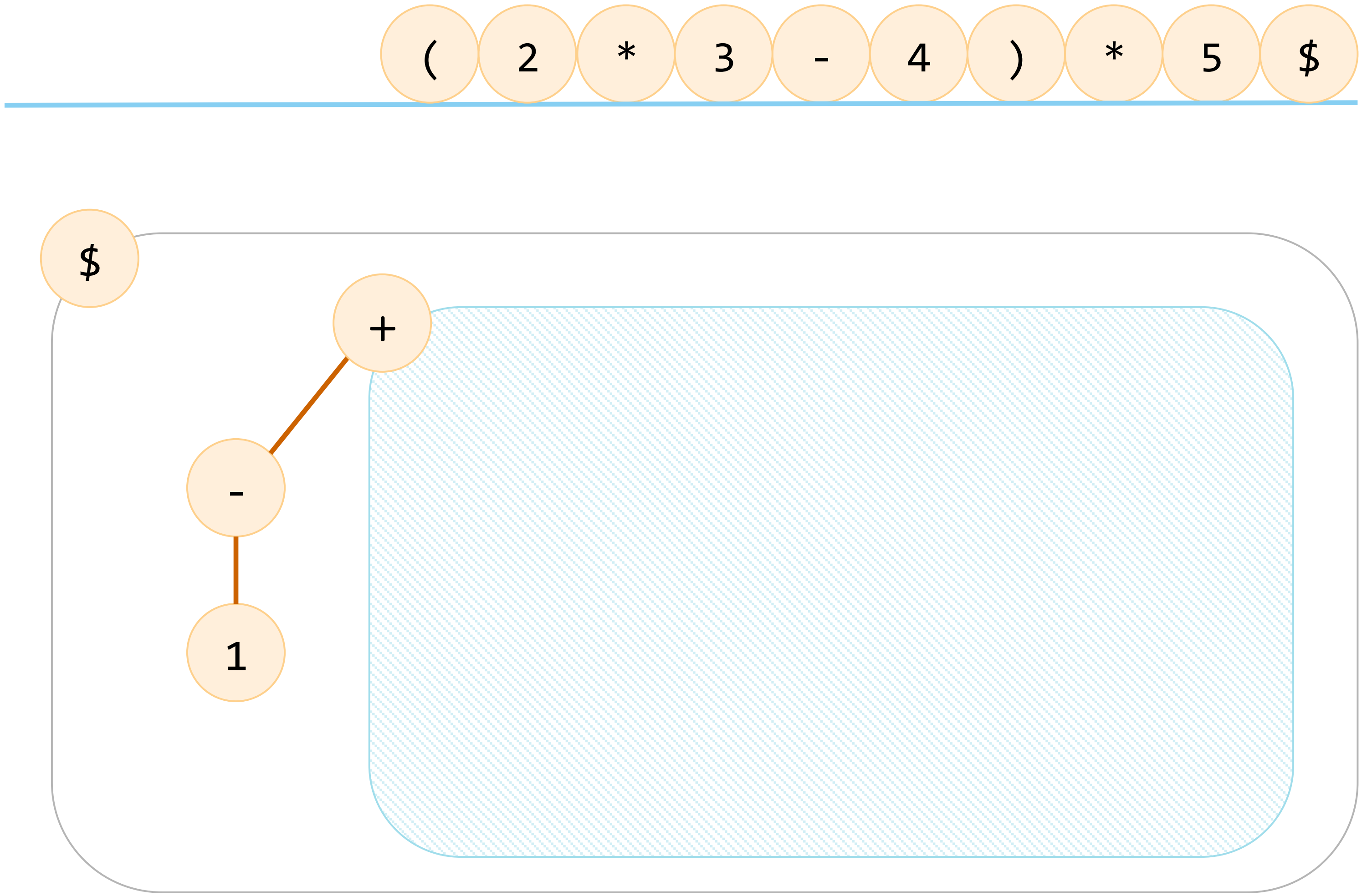


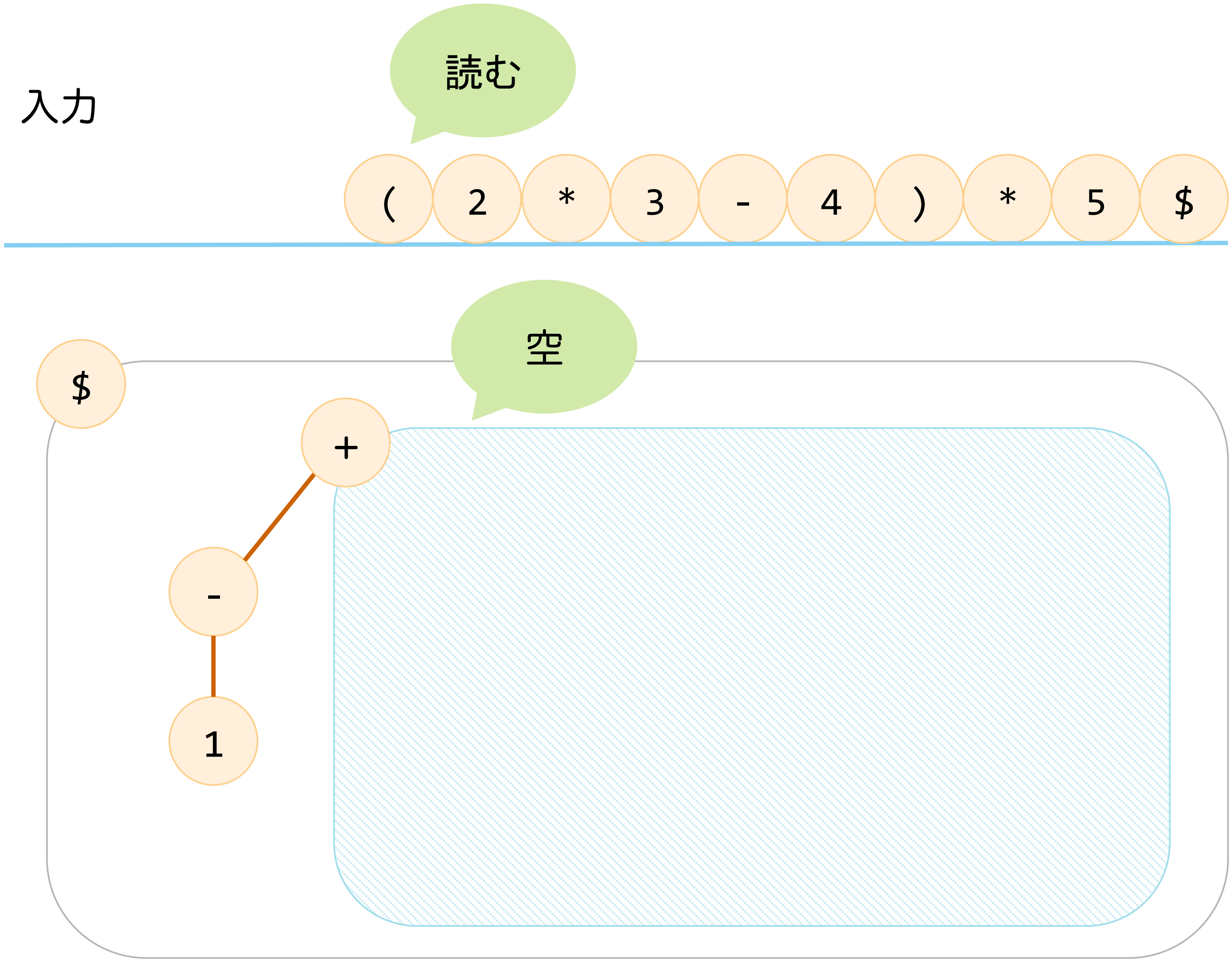

入力
高
+ (2 * 3 - 4) * 5 $
低
$
-
1

入力

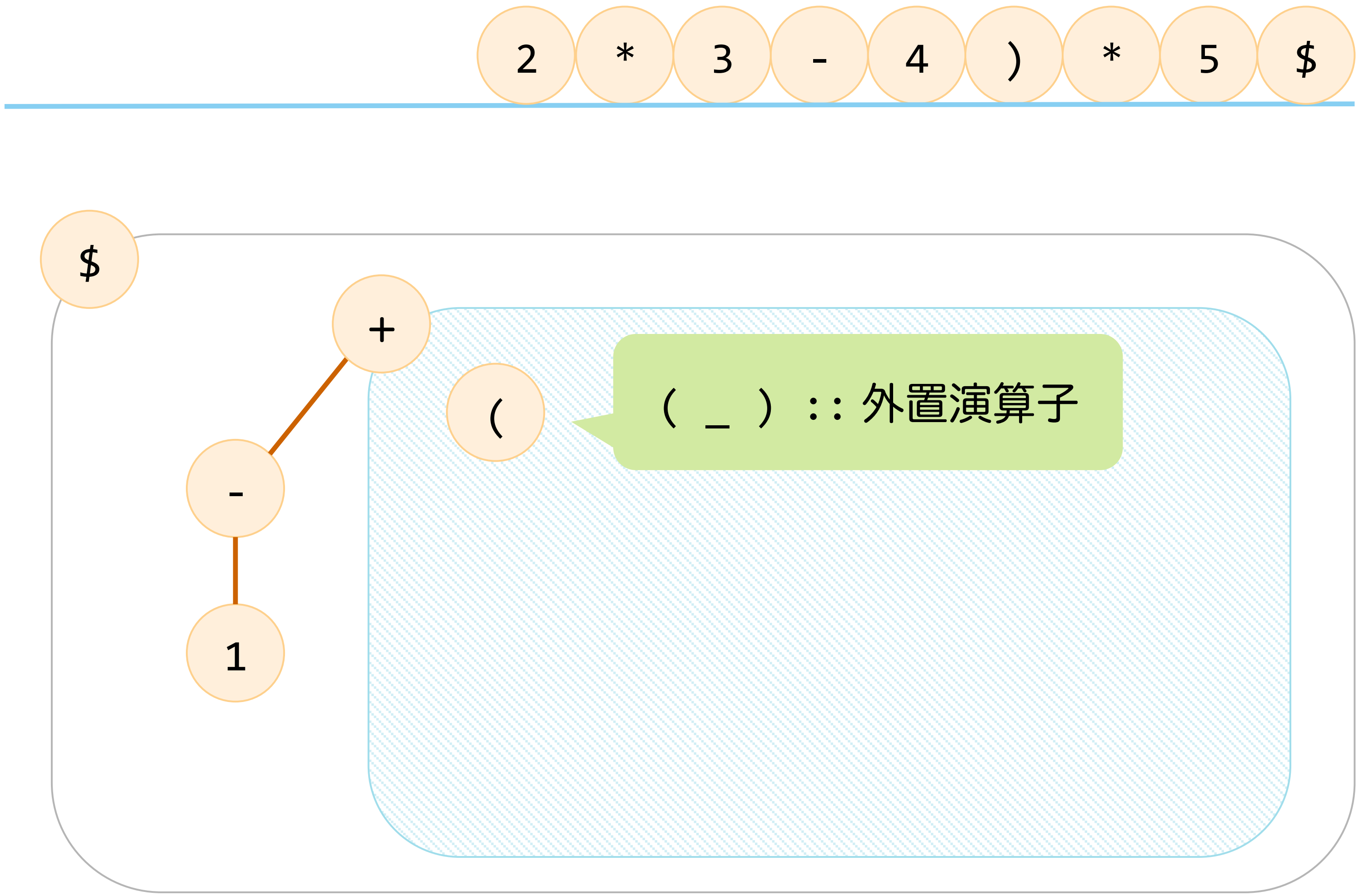


入力

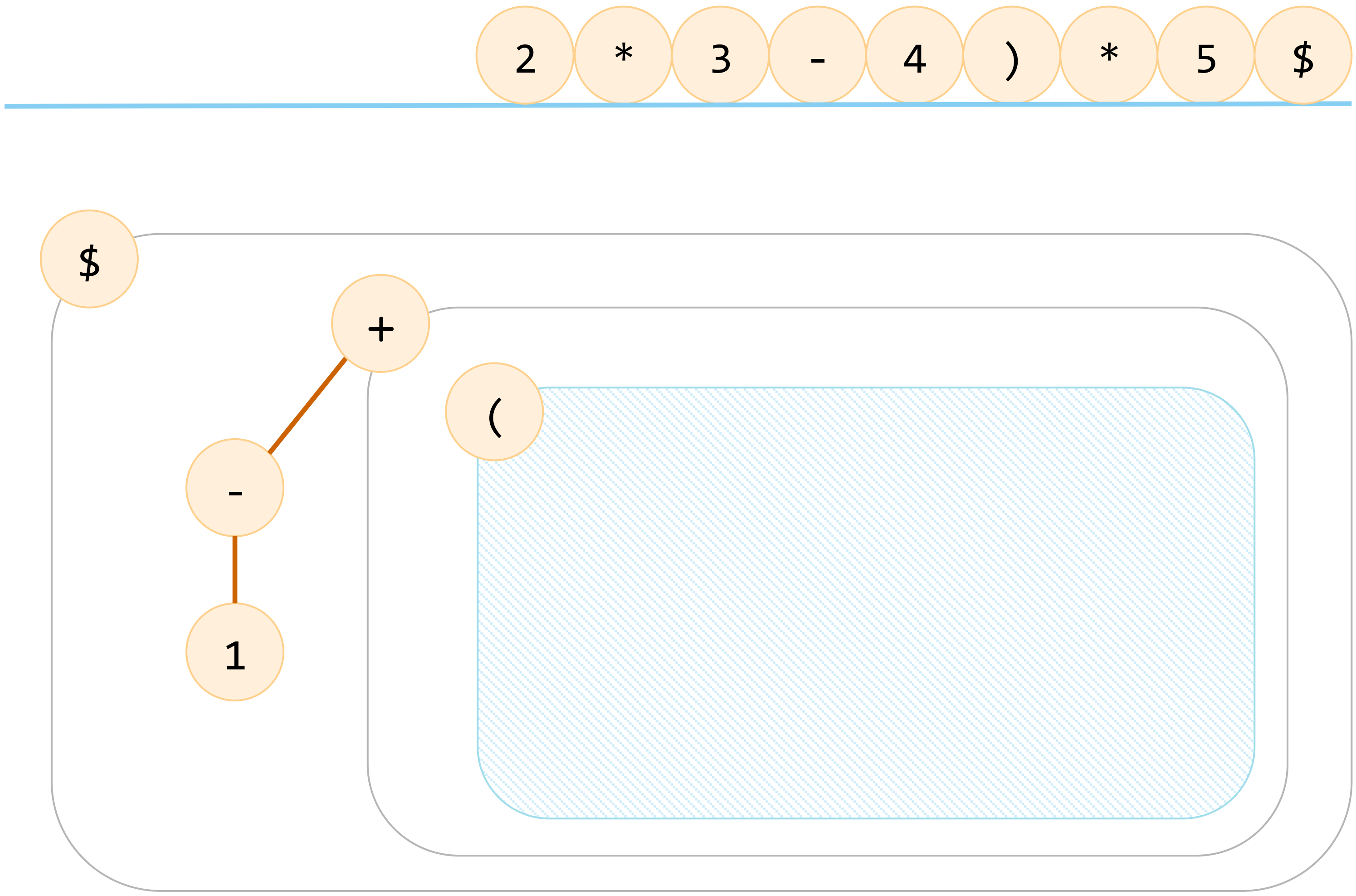



入力
読む
(2 * 3 - 4) * 5 $
空
$
+
-
1

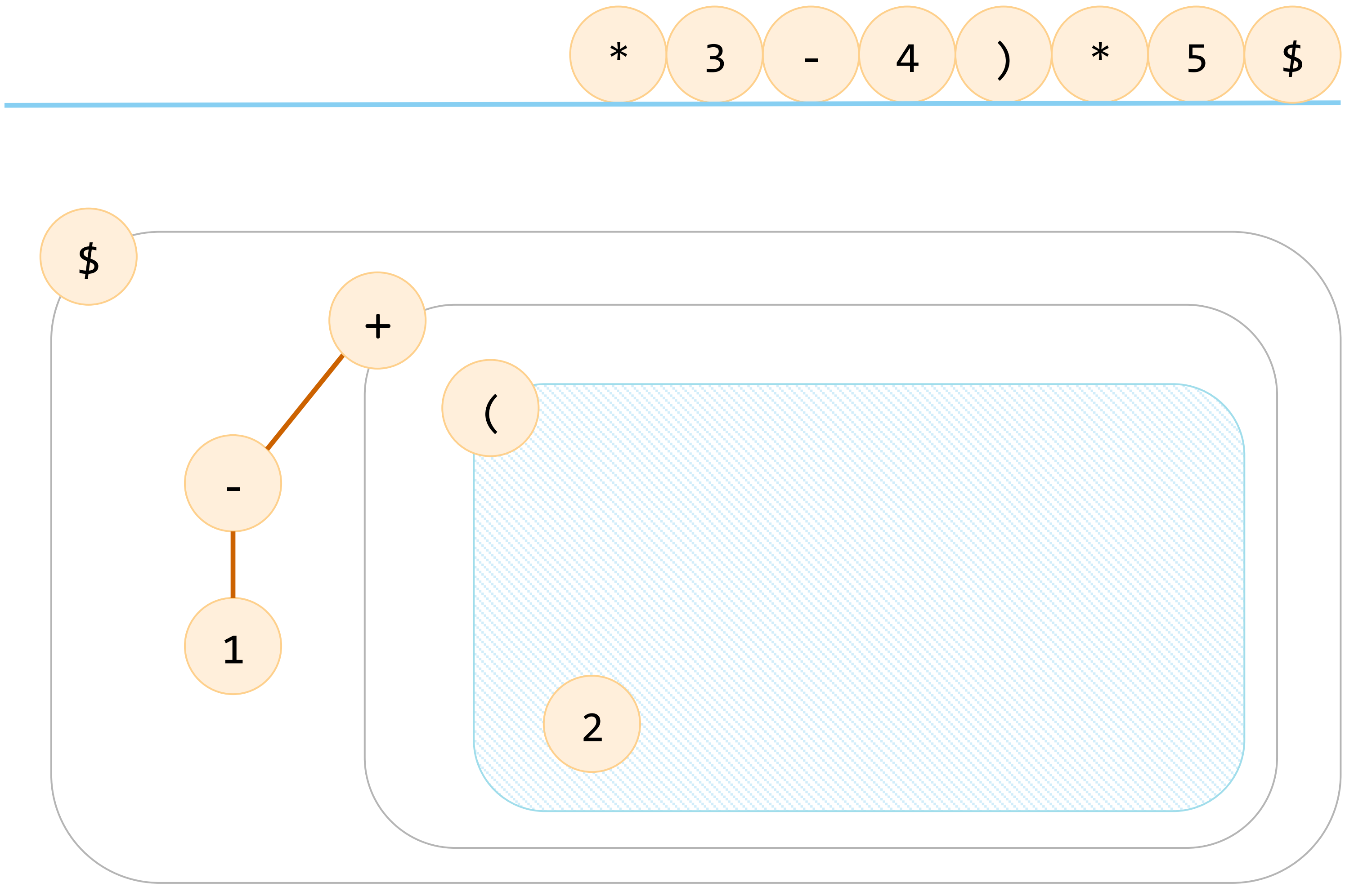
入力

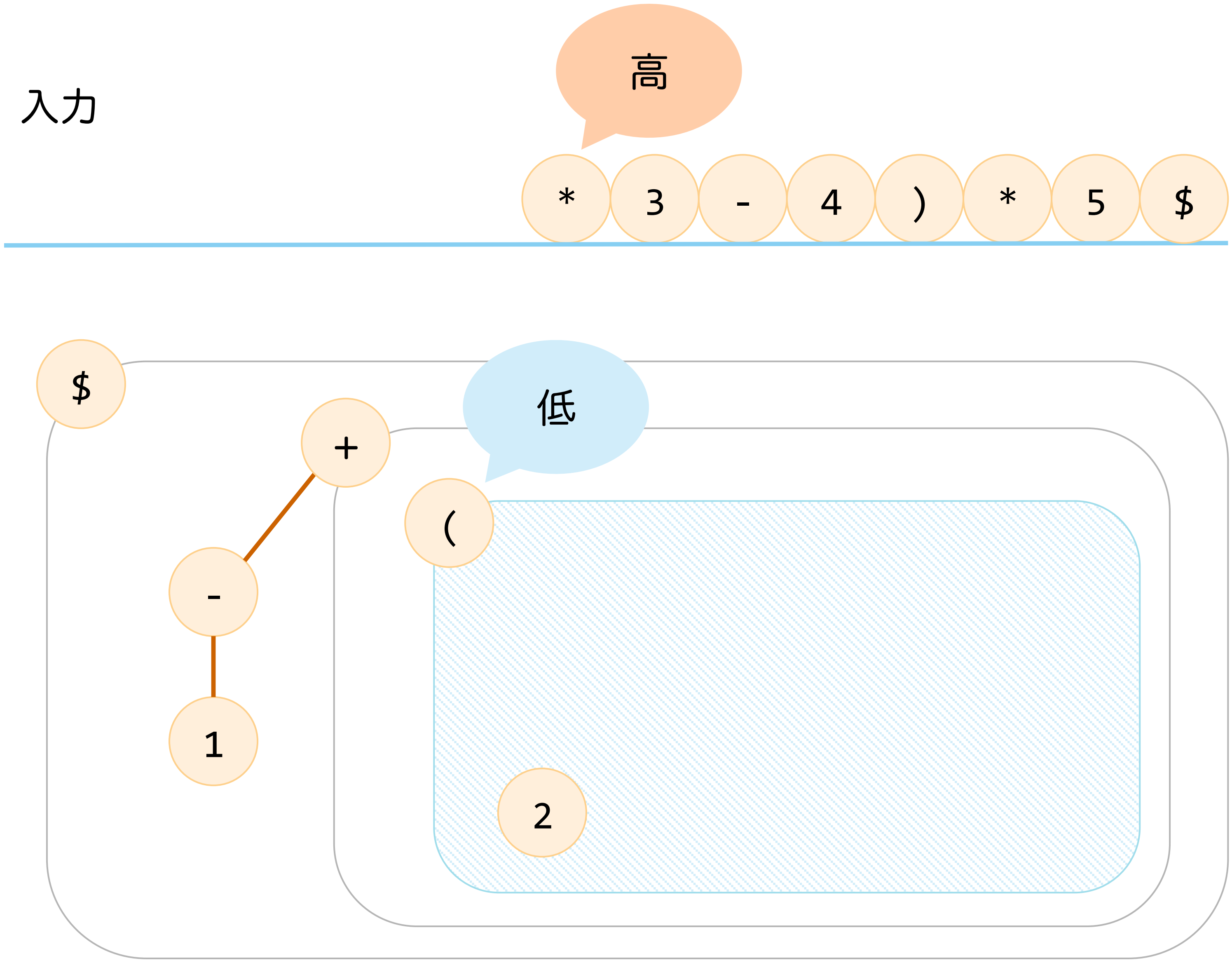


入力

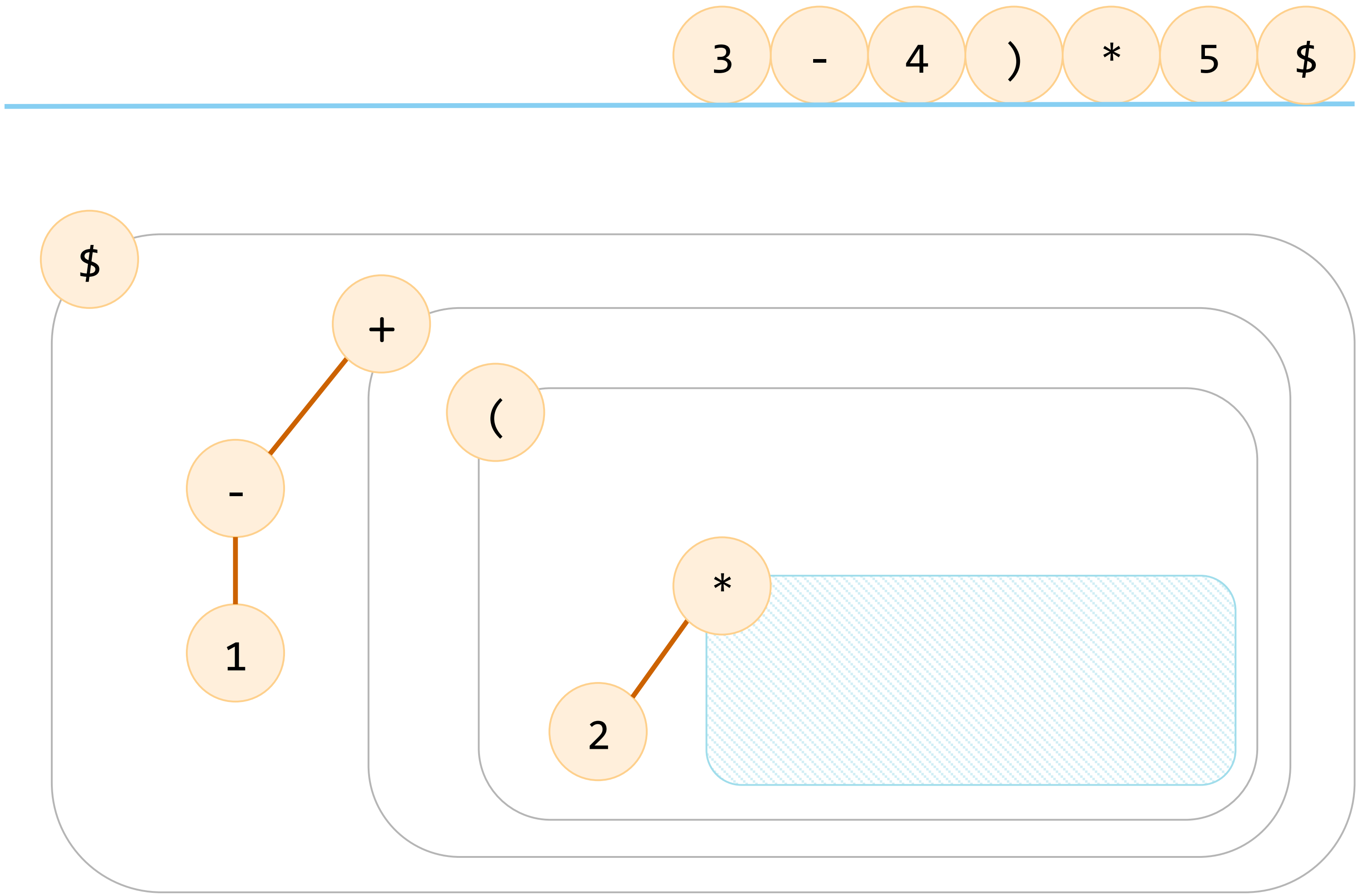


入力

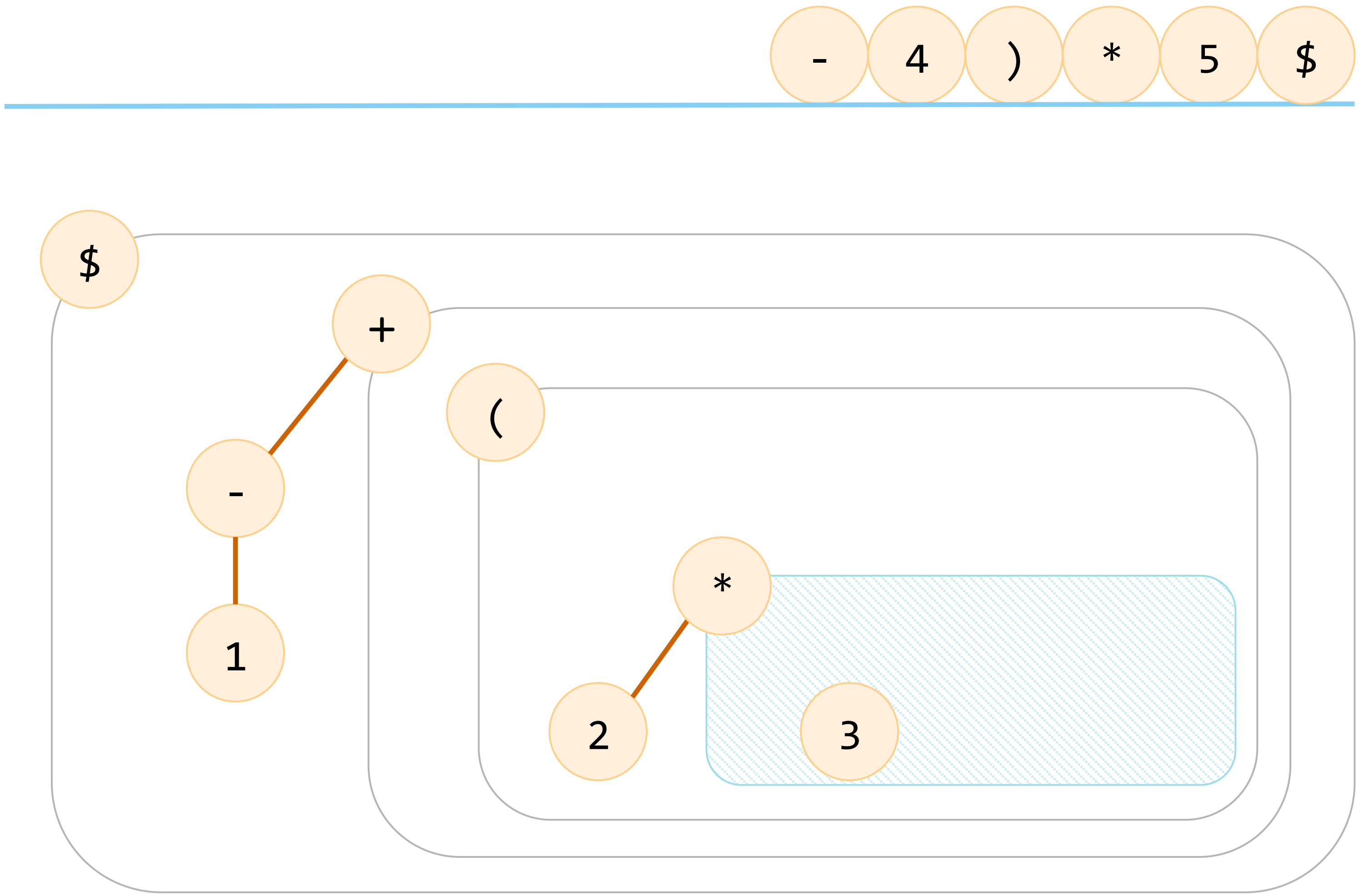



入力
高
* 3 - 4) * 5 $
低
$
+
-
1
(
2

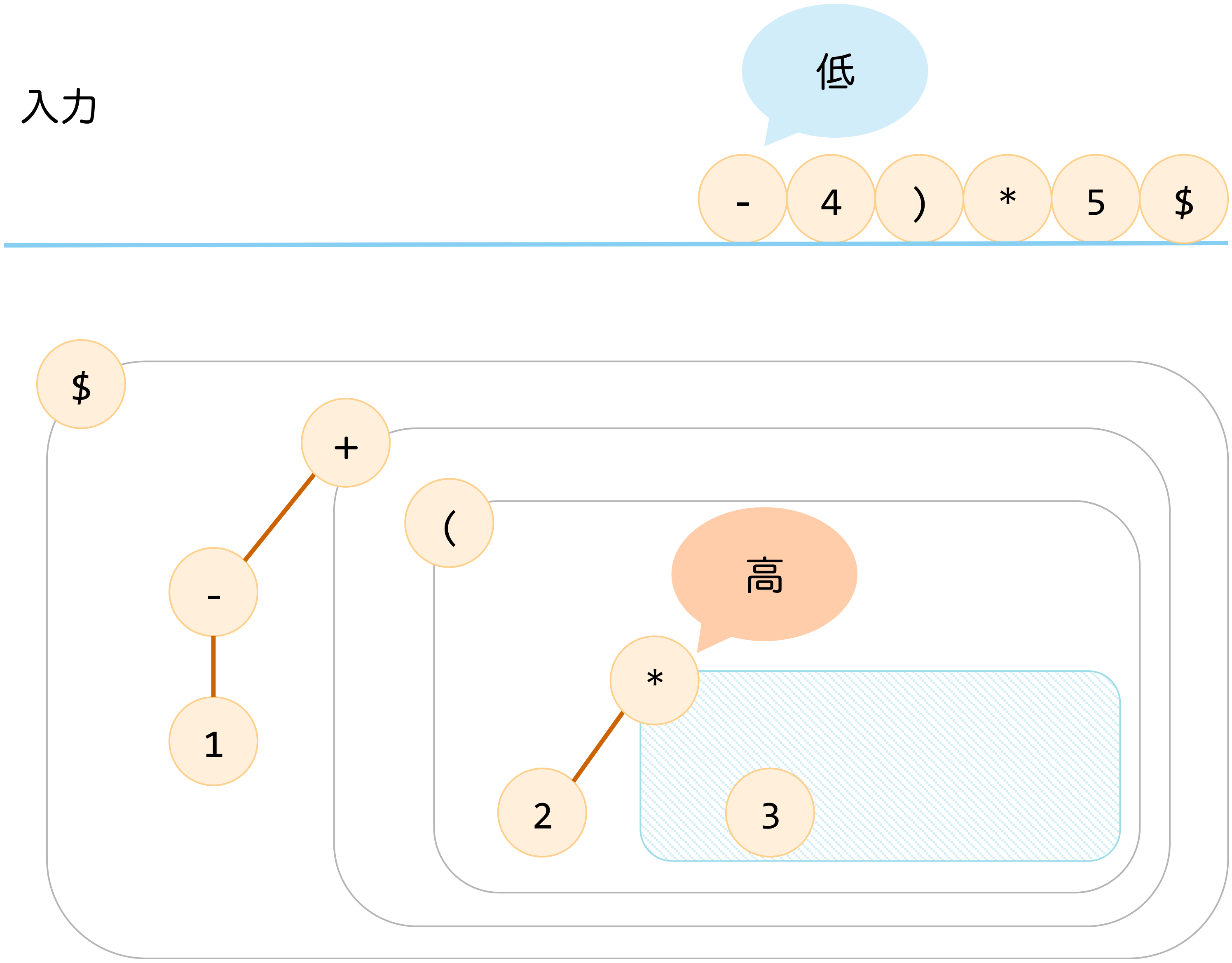
入力



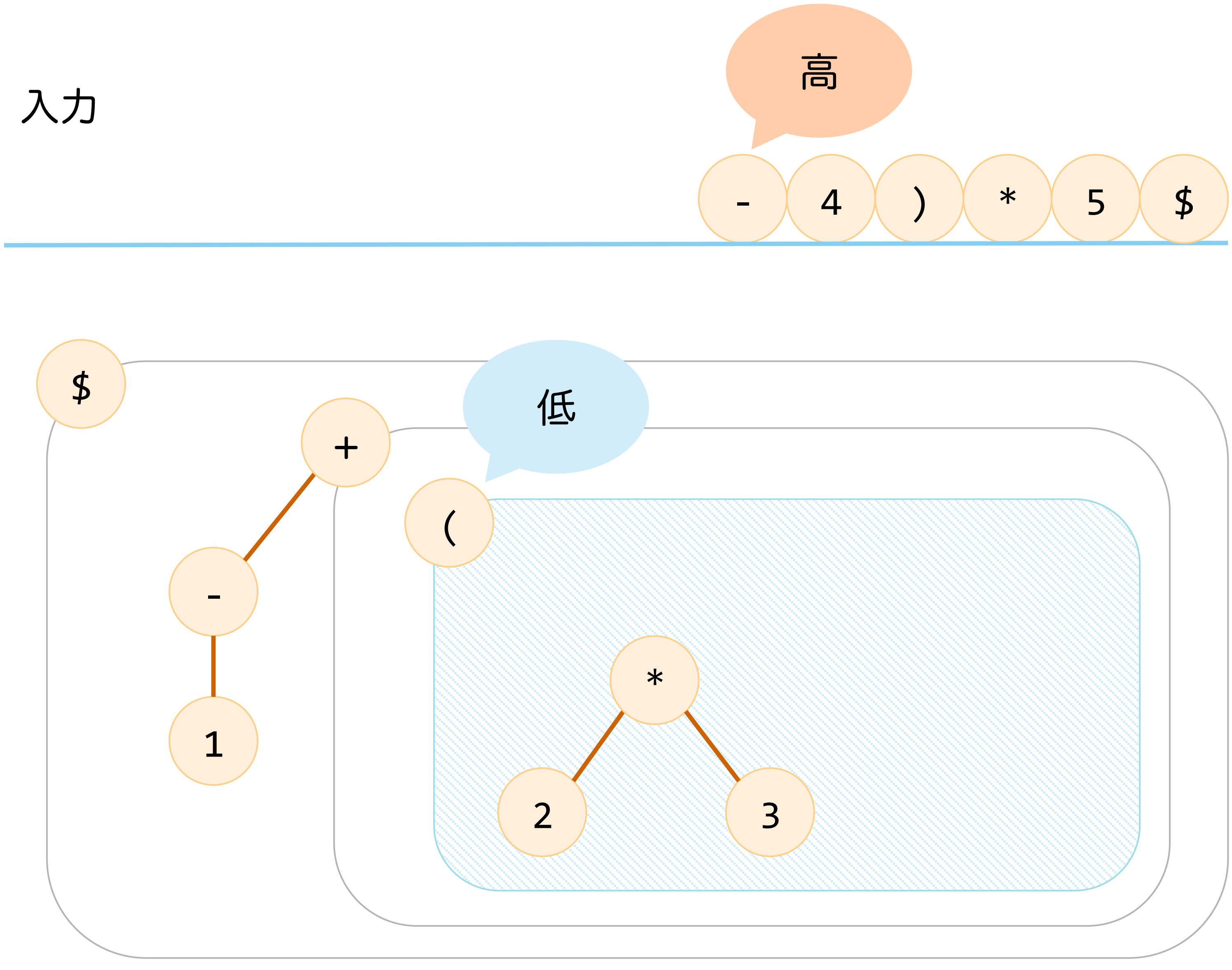
入力



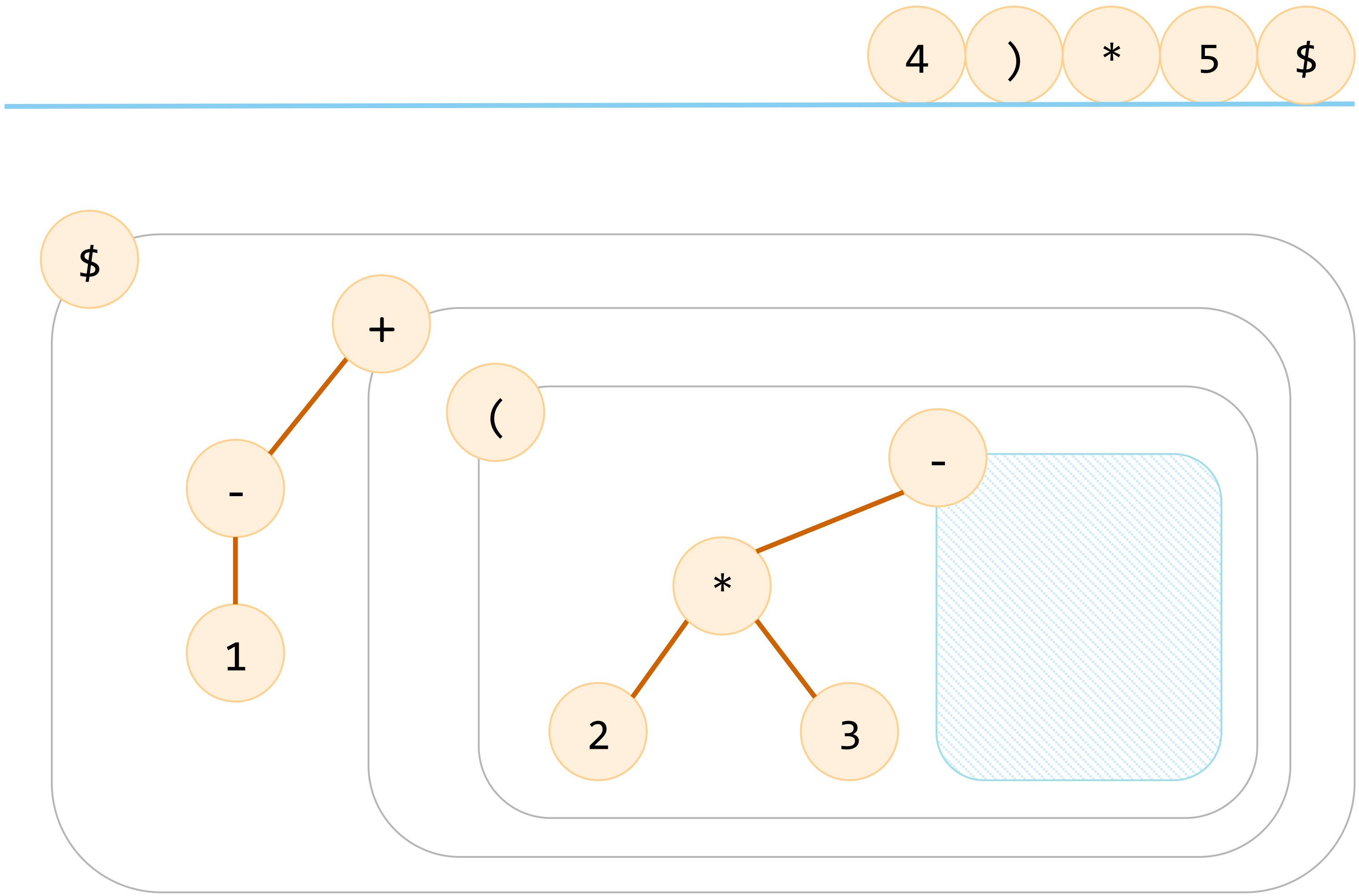
入力



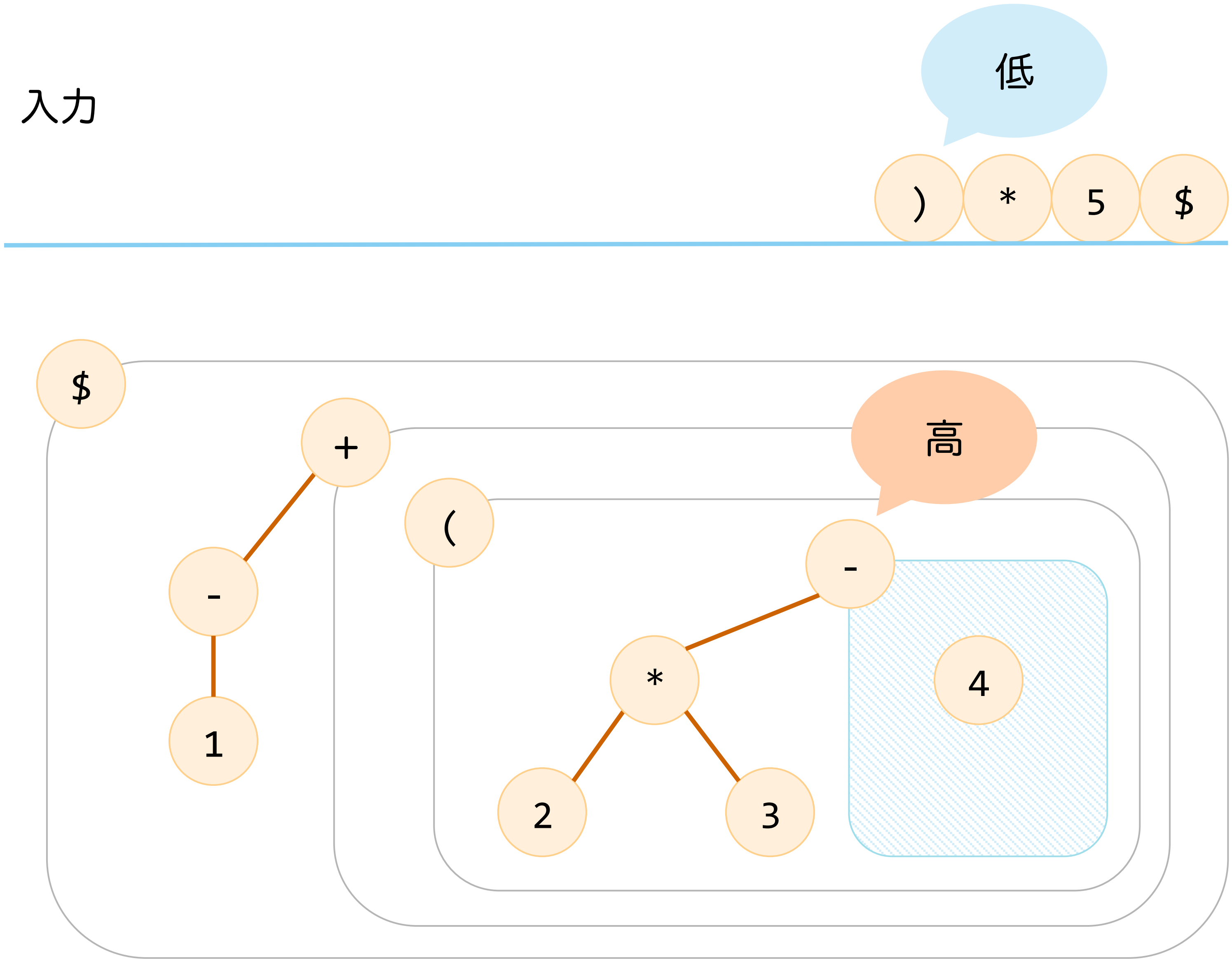
入力



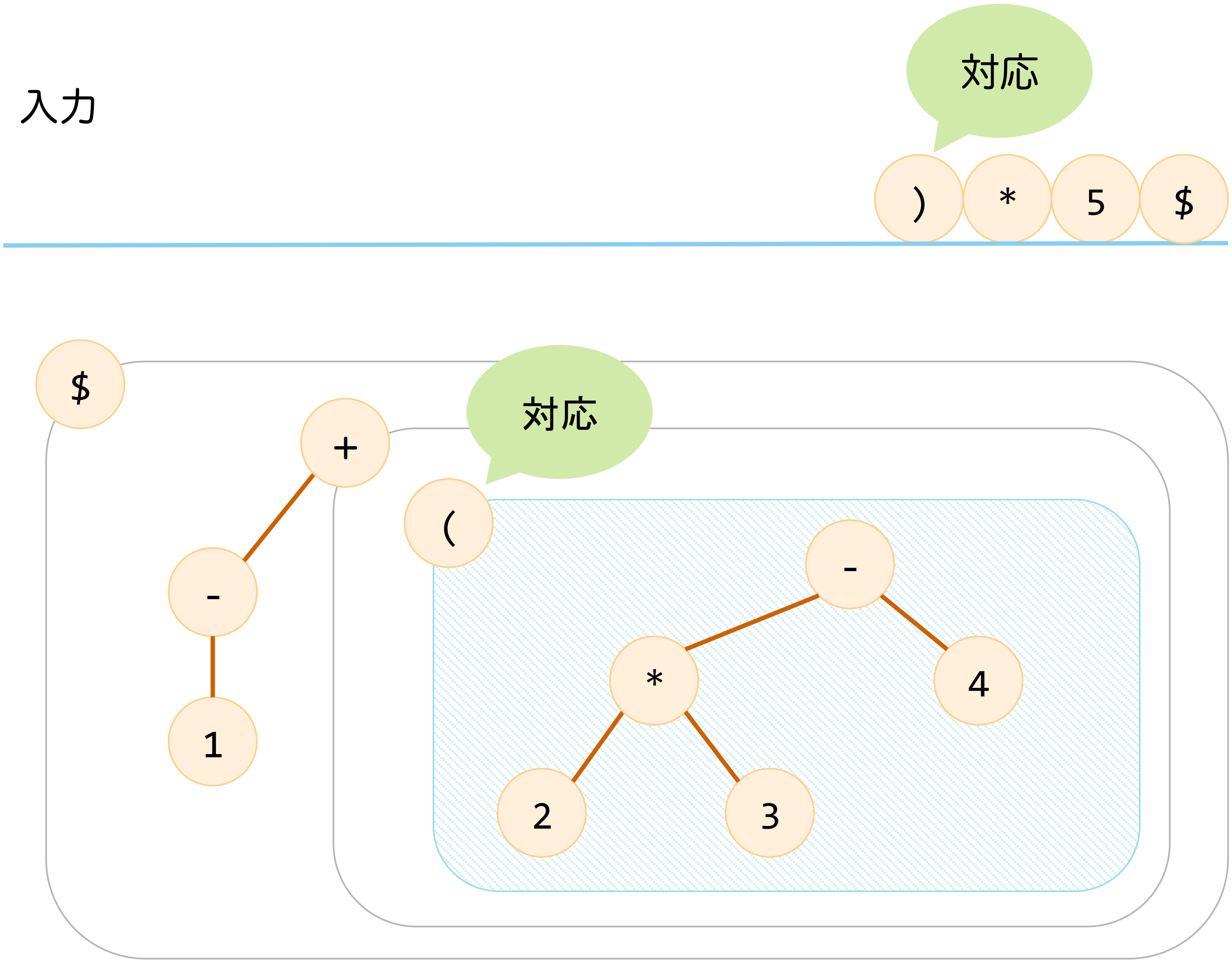
入力



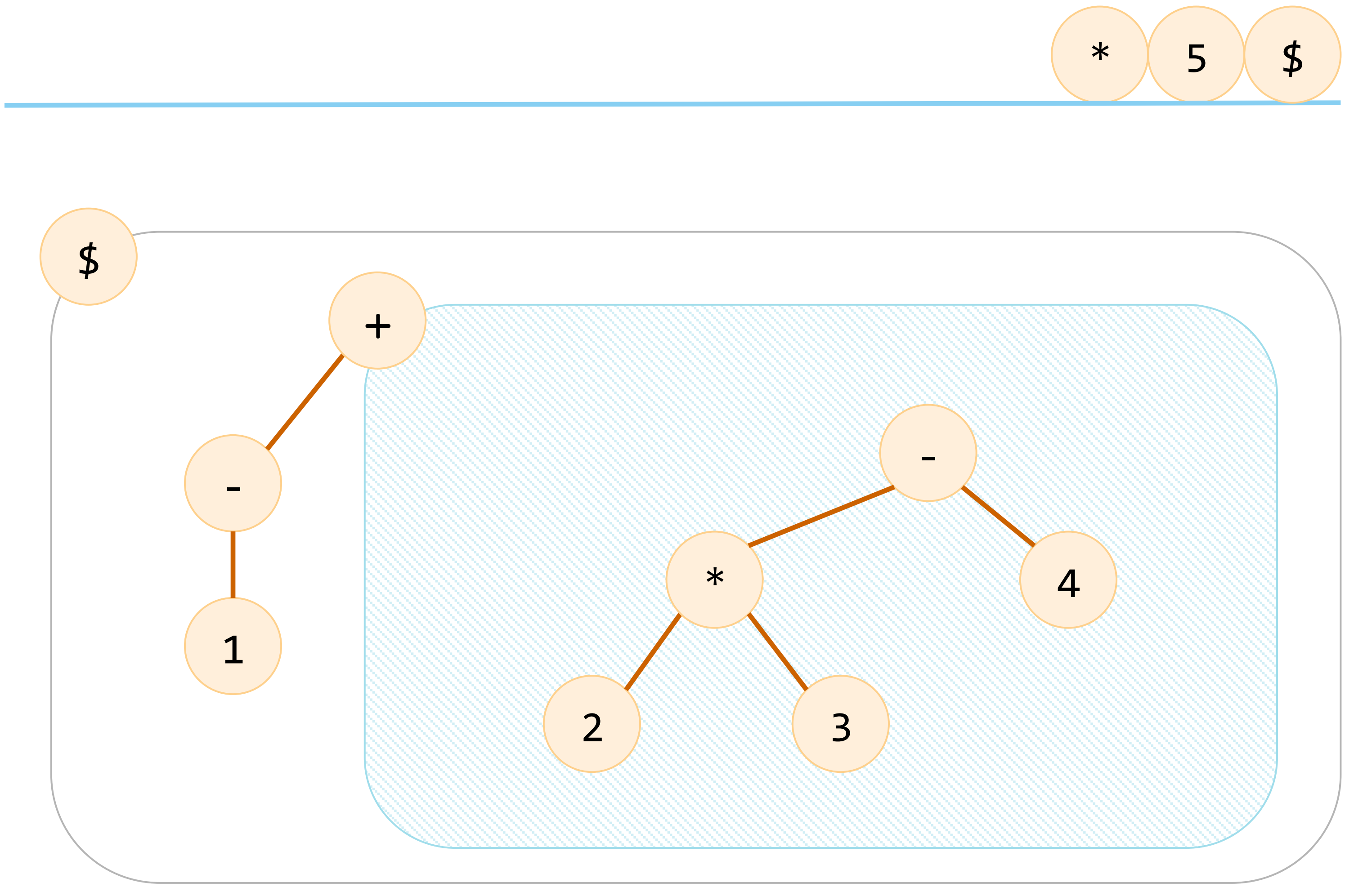
入力

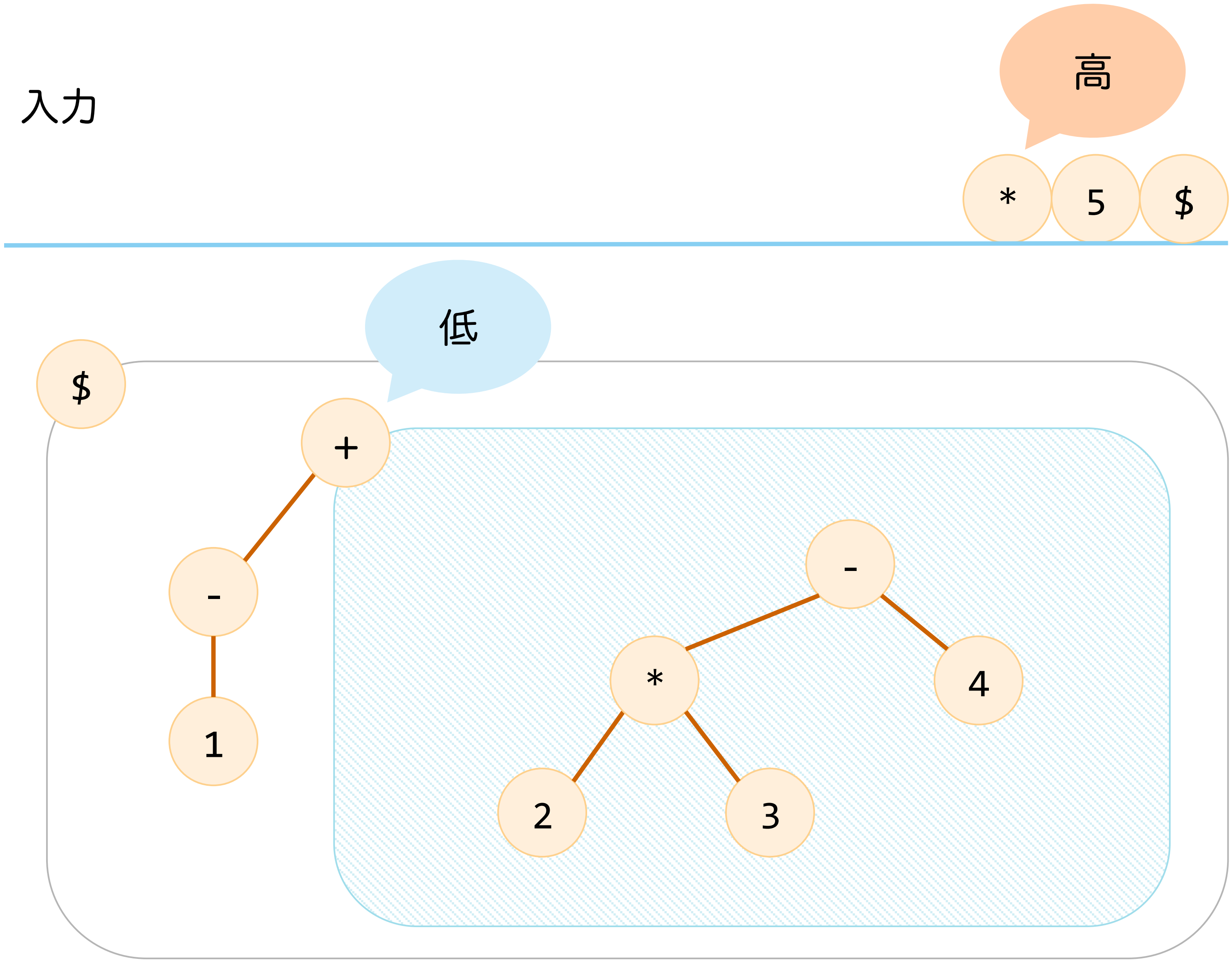


入力

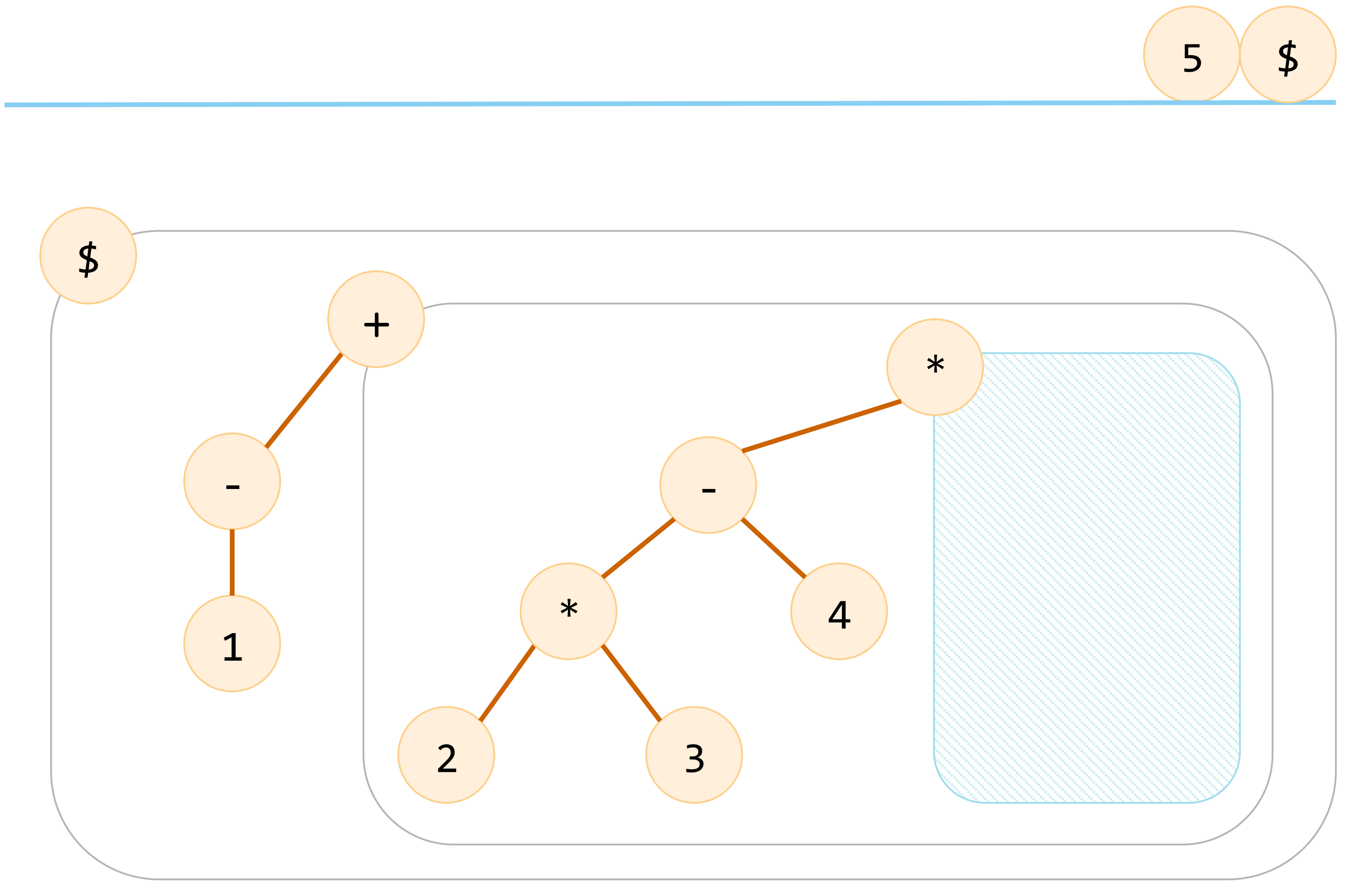


入力

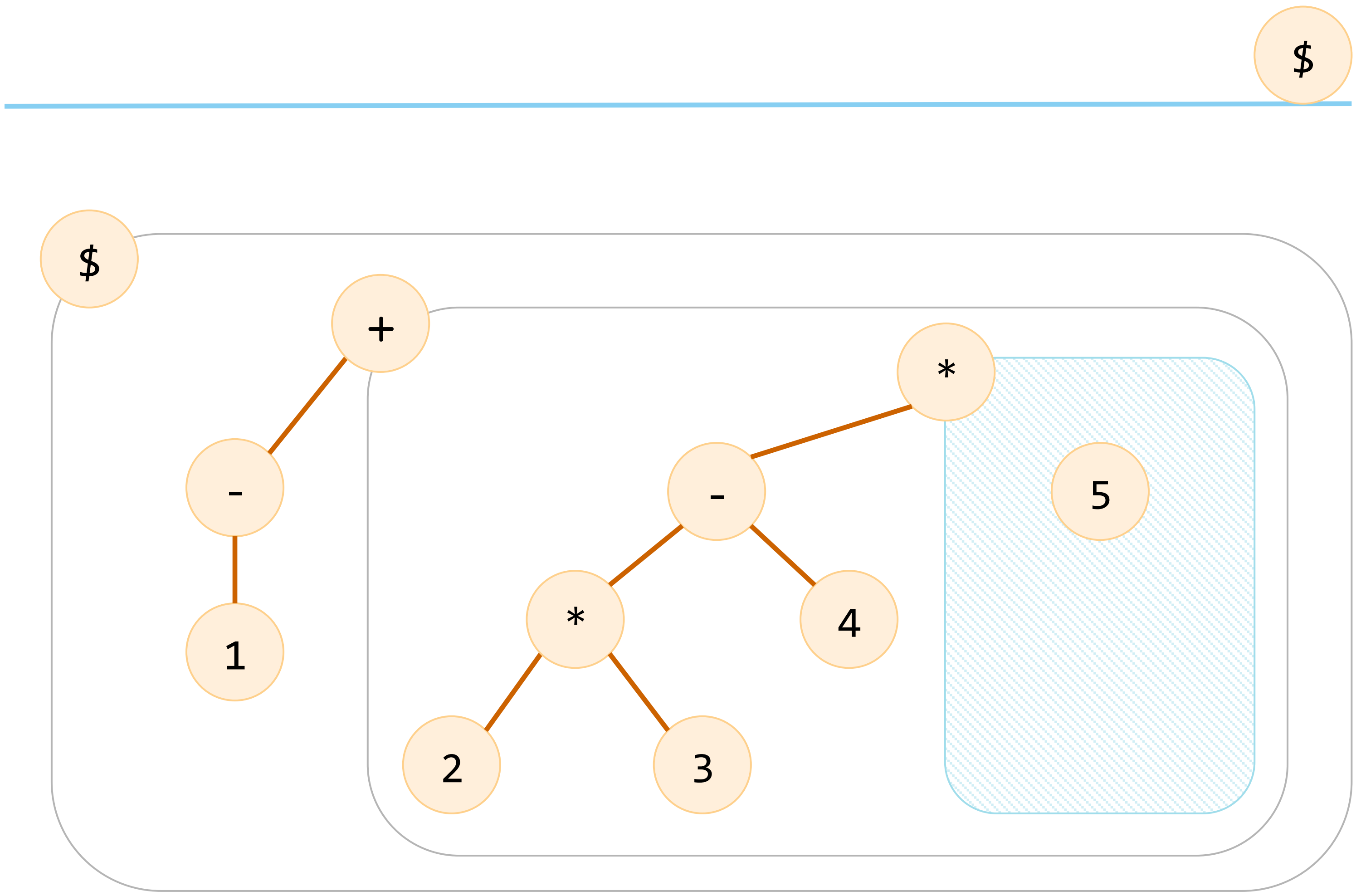



入力
高
*
5
$
低
$
+
-
1
-
*
4
2
3

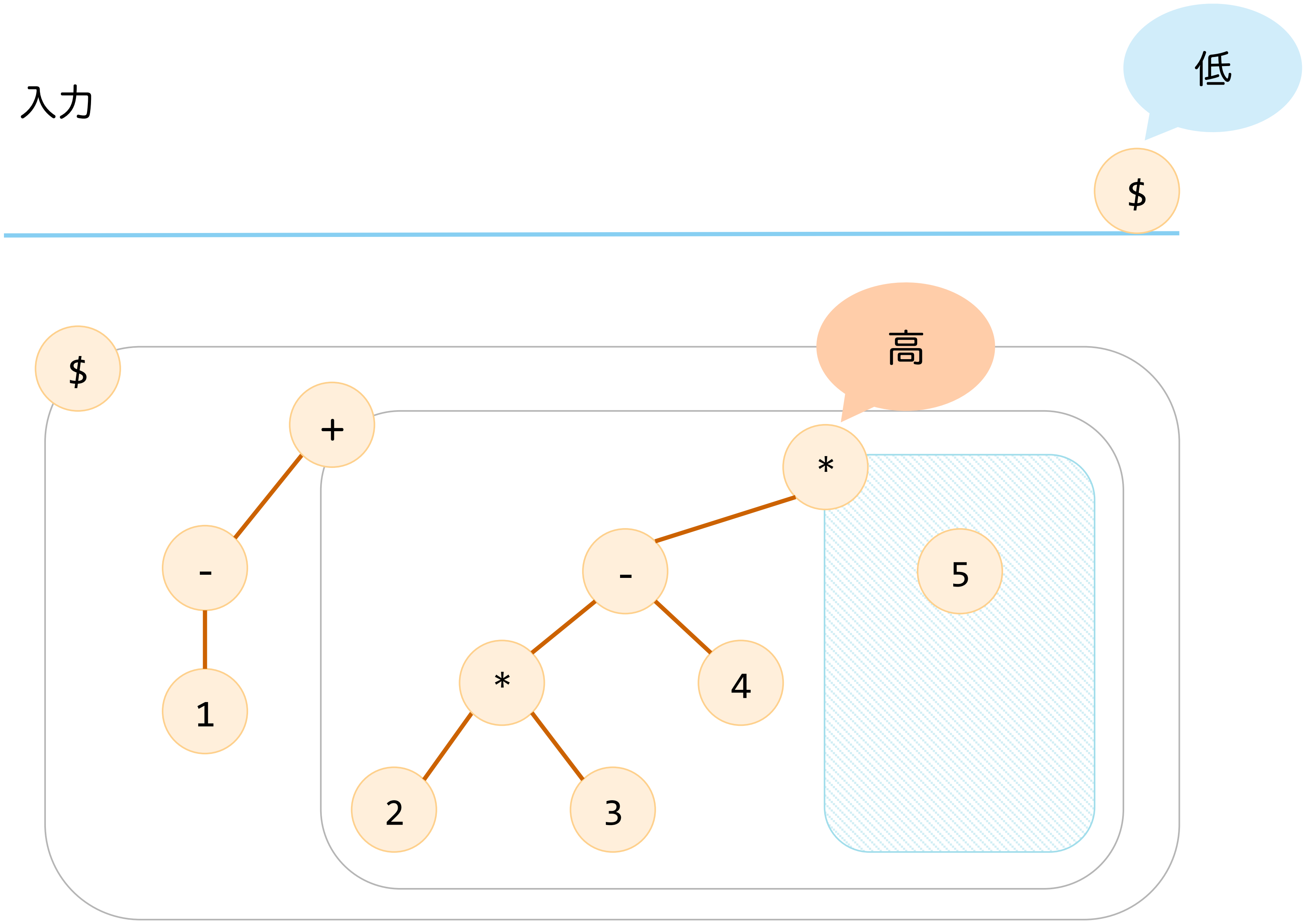
入力



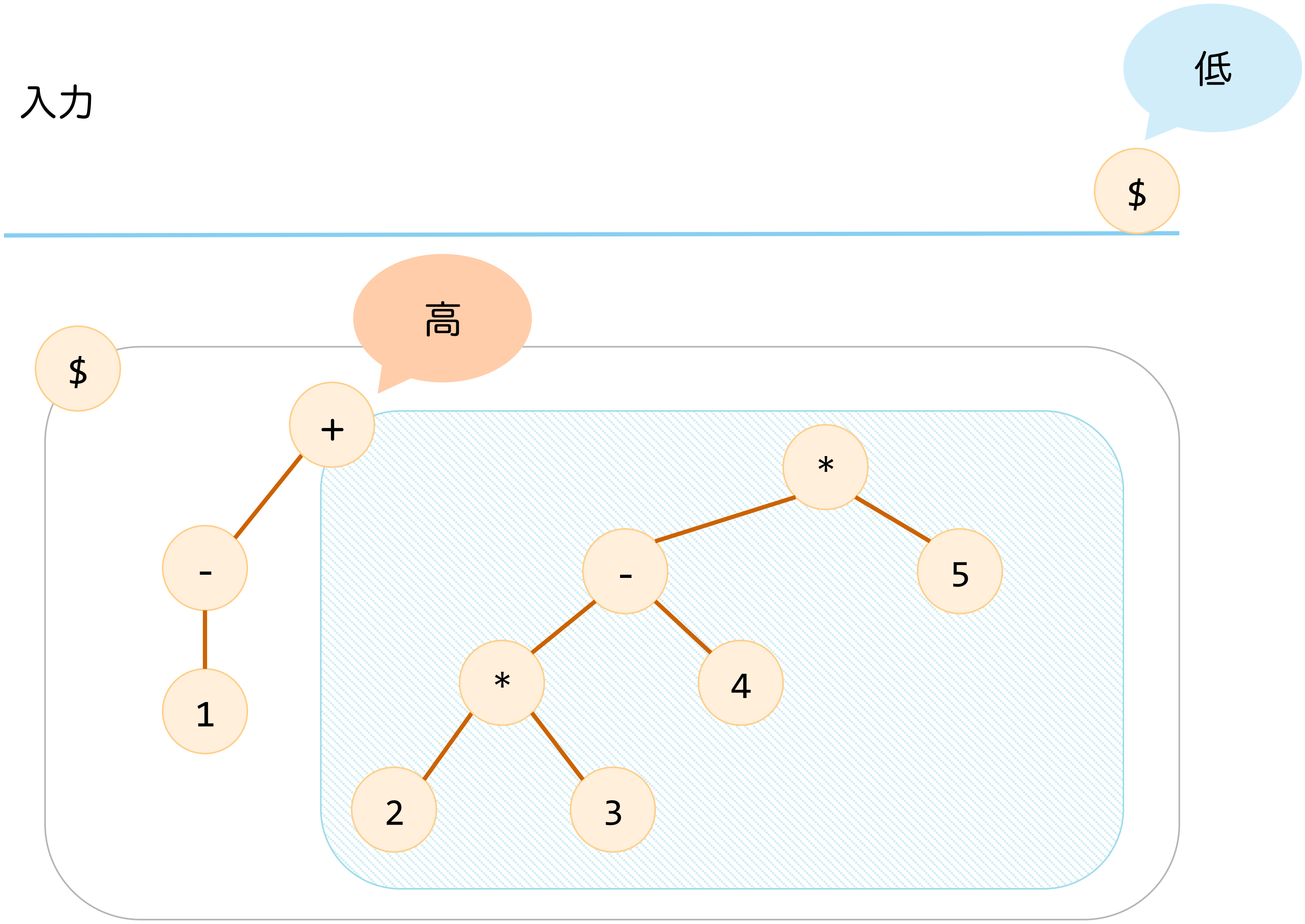
入力



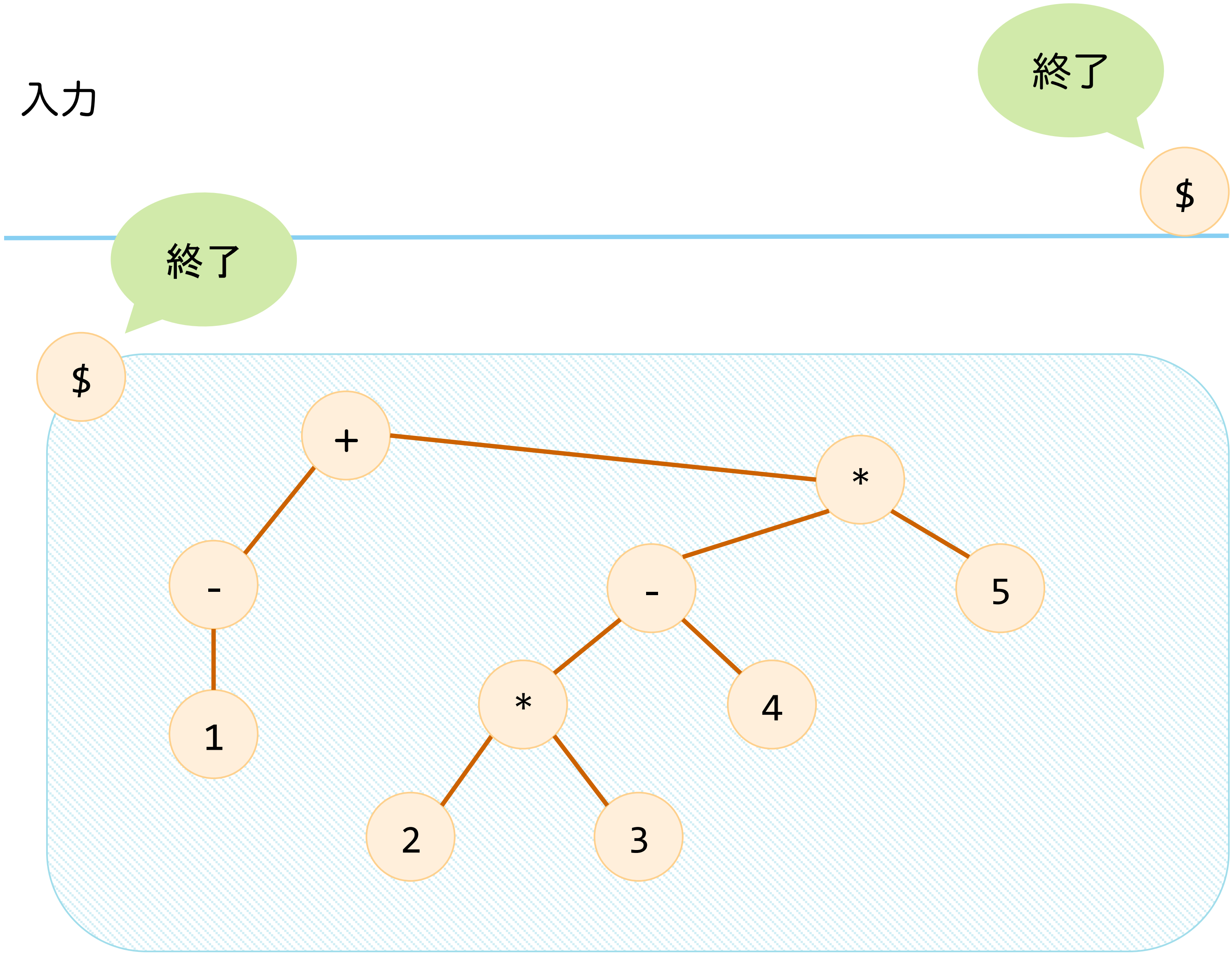
入力



入力



入力



Pratt parsing [Pratt 1973]

- 手書きの構文解析器の一部としてよく用いられる
  - 再帰関数による実装と相性が良い
- 演算子順位構文解析の手法は他にも存在する
  - Precedence climbing method [Richardsら 1979]
- 能力
  - 計算量 : O(n)
  - 言語 : Operator-precedence language
  - 文法 : Operator-precedence grammar

Bottom-up parsing

上向き構文解析

LR parsing [Knuth 1965]

- 実用に十分な表現力を持ち、かつ高速
- 多くのパーサジェネレータで利用されている

GLR parsing [Tomita 1985]

- Earley parsing の最適化版
- LR parsing の自然な拡張でもある

Earley parsing [Earley 1968]

- CYK algorithm を任意の CFG に拡張したもの

CYK algorithm [Cocke 1969, Younger 1967, Kasami 1965]

- 様々な CFG の構文解析手法の理論的なベース
- Chomsky 標準形のみを対象とする

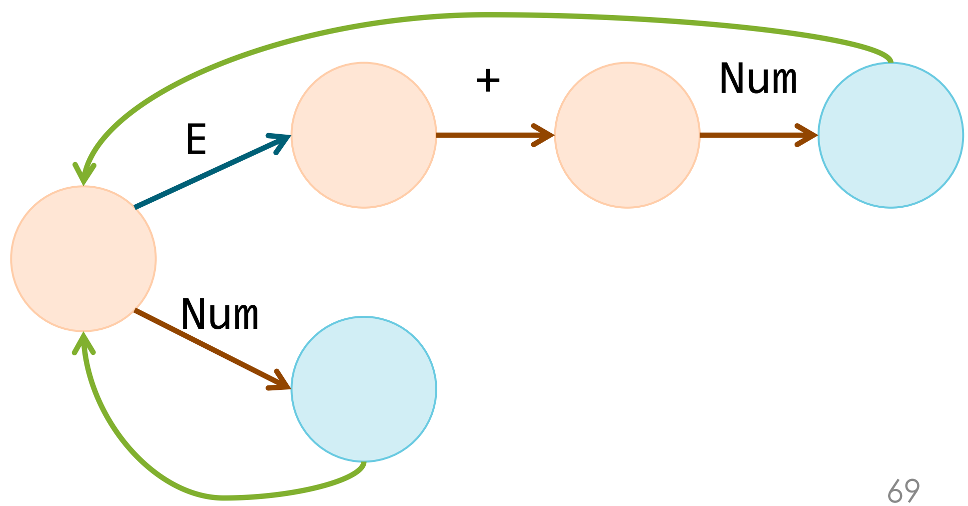
Valiant parsing [Valiant 1975]

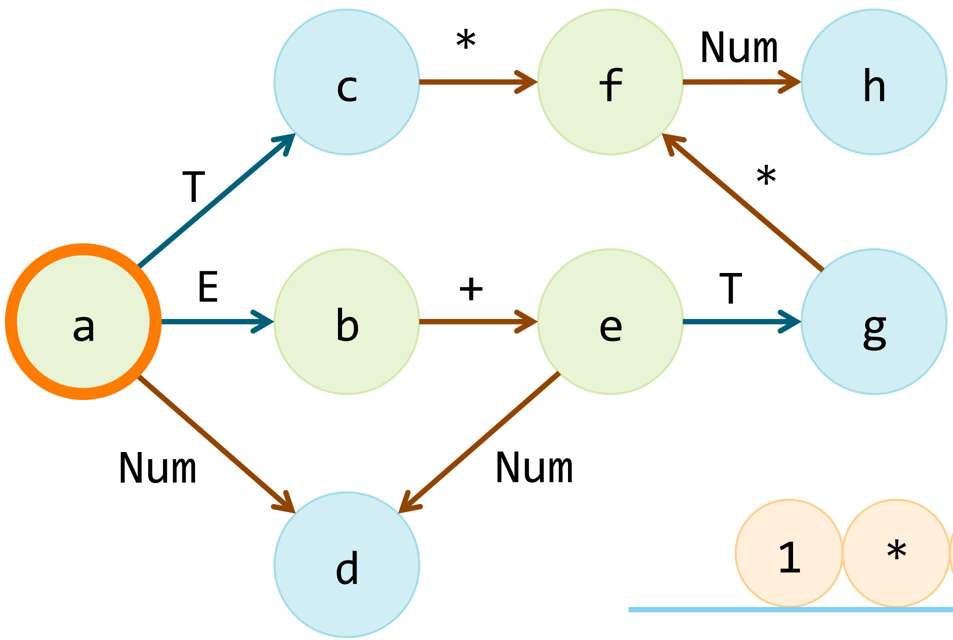
- CFG の構文解析は行列積の計算量で計算できることを示した

LR parsing [Knuth 1965]

アイデア

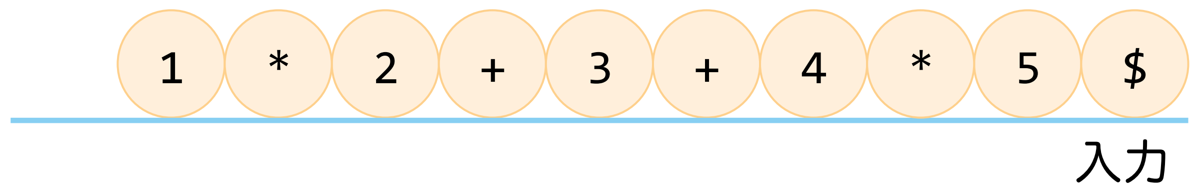
- 決定性オートマトンを作る
- shift & reduce
  - ➢ shift : 終端記号を読んで遷移
  - ➢ reduce : 構文木を組み立てて非終端記号で遷移

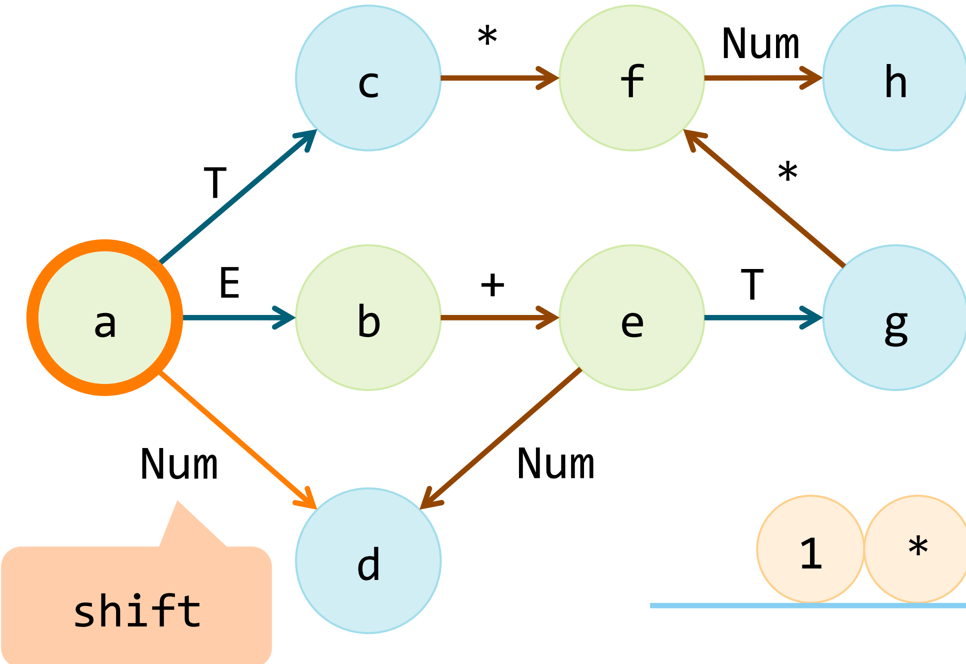




文法

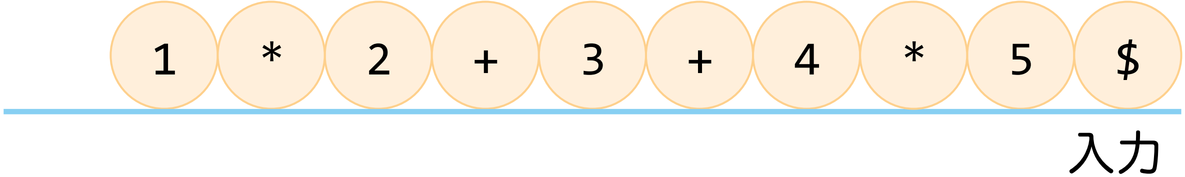
E → E + T
E → T
T → T * Num
T → Num

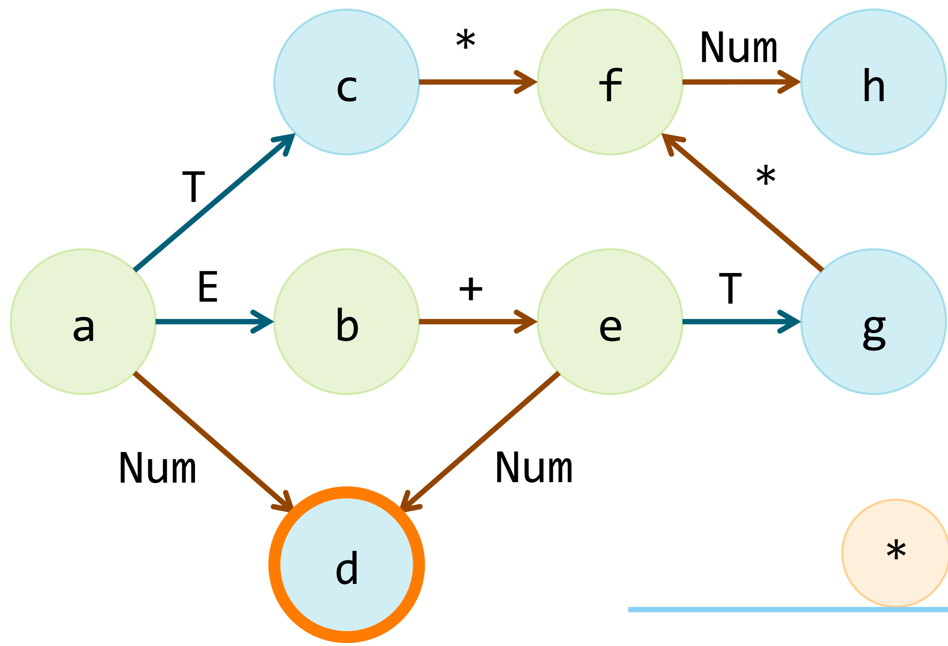




文法

E → E + T
E → T
T → T * Num
T → Num





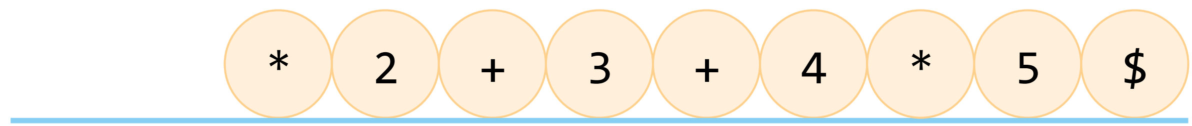
文法

E → E + T

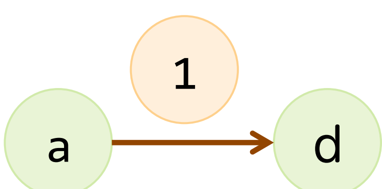
E → T

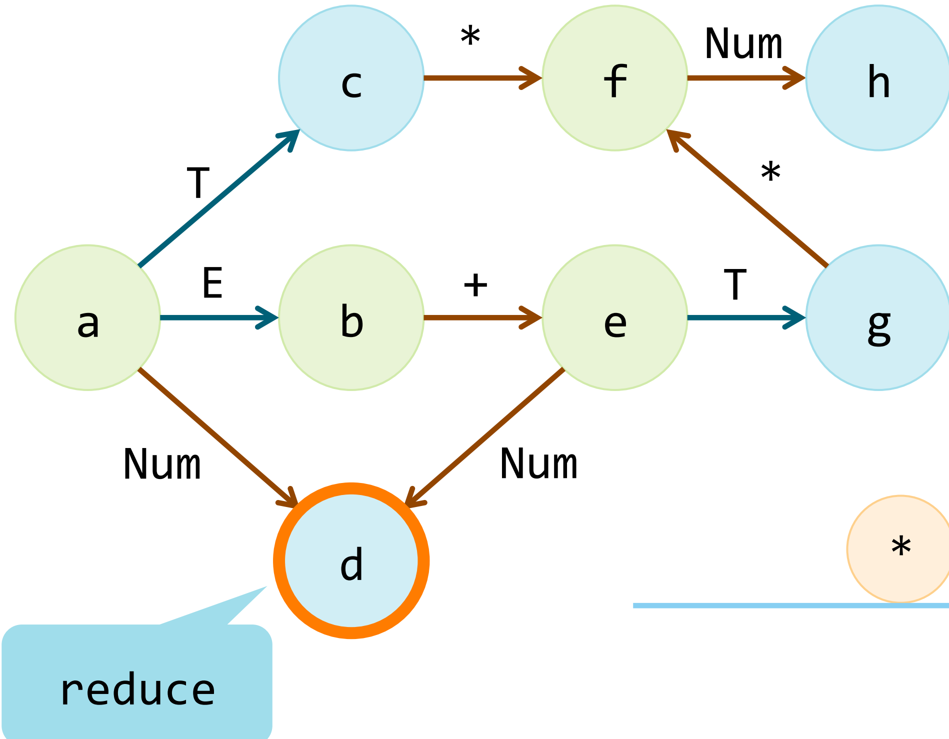
T → T * Num

T → Num



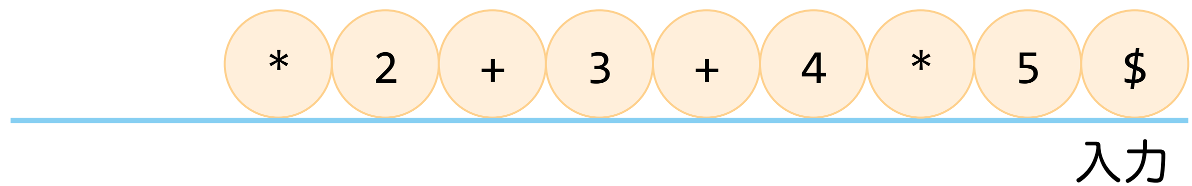
入力

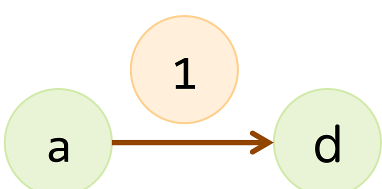


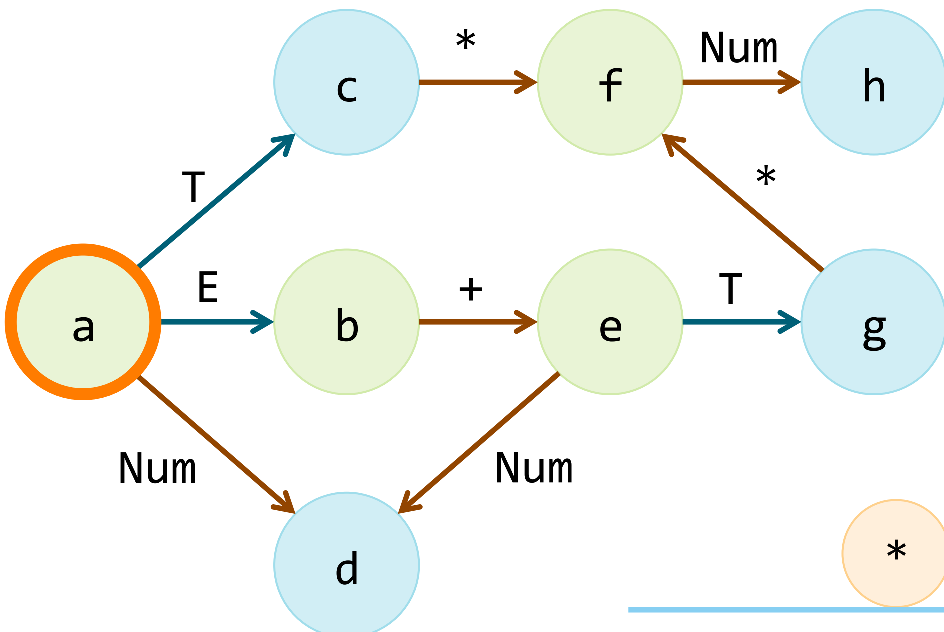


文法

E → E + T
E → T
T → T * Num
T → Num







文法

E → E + T

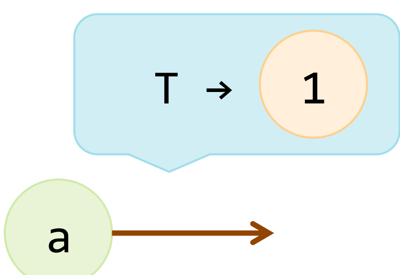
E → T

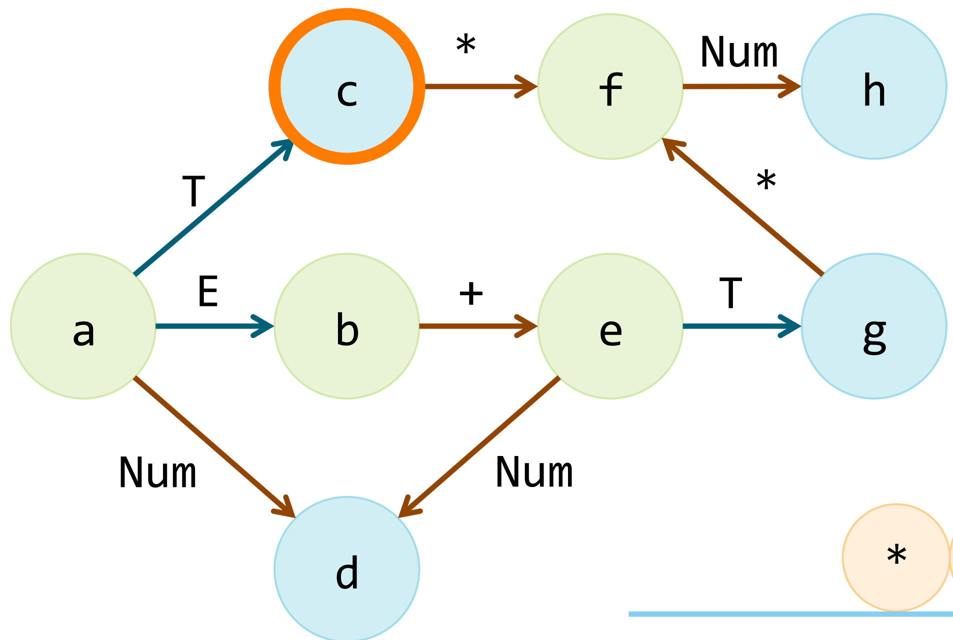
T → T * Num

☆ T → Num

* 2 + 3 + 4 * 5 $

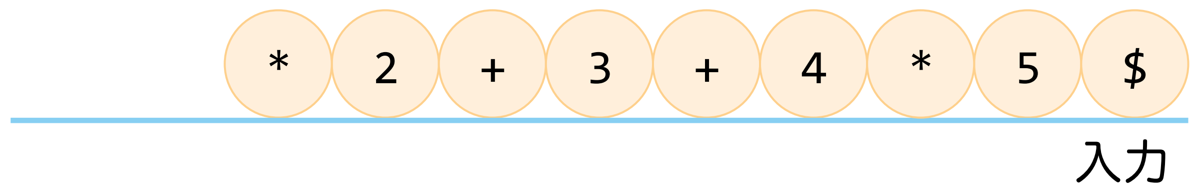
入力

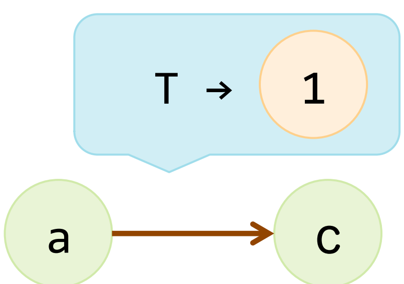


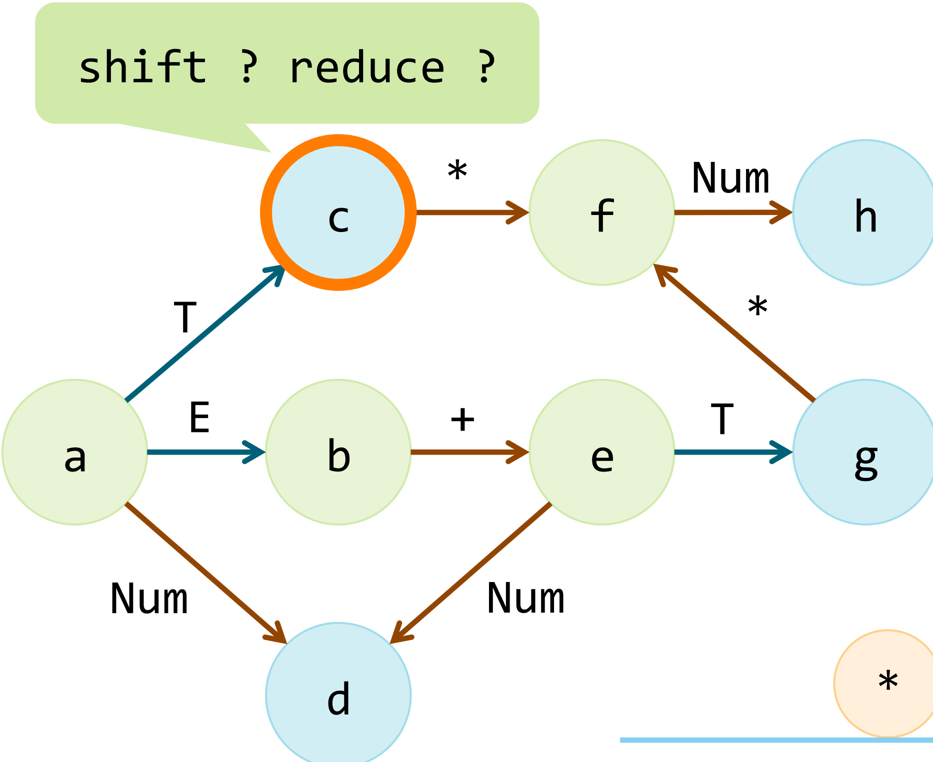


文法

E → E + T
E → T
T → T * Num
☆ T → Num

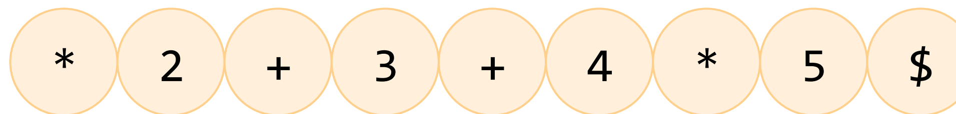




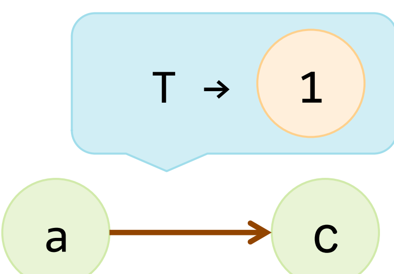


文法

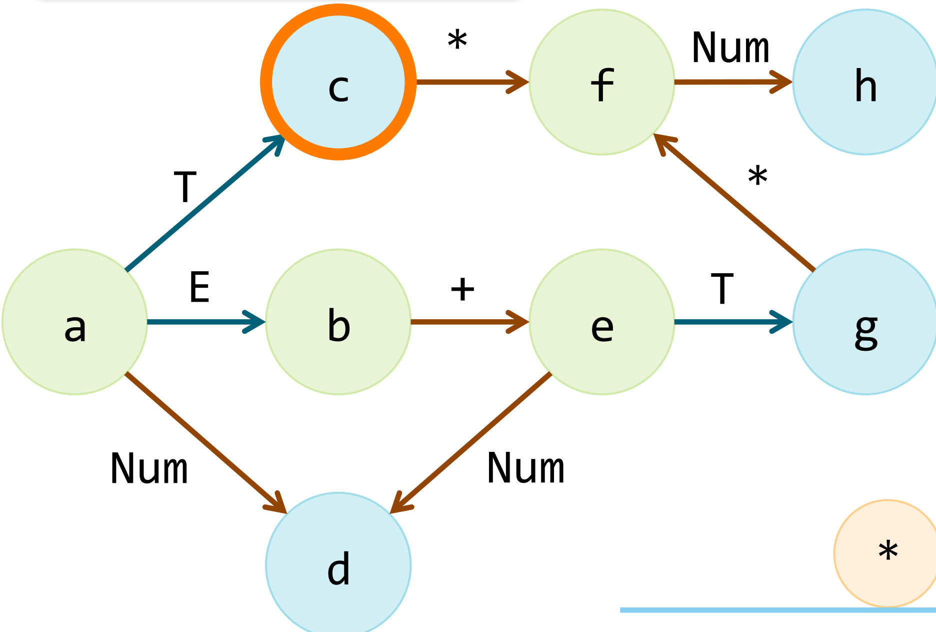
E → E + T
E → T
T → T * Num
T → Num



入力

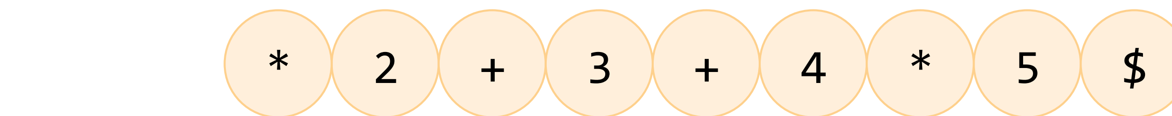


look-ahead

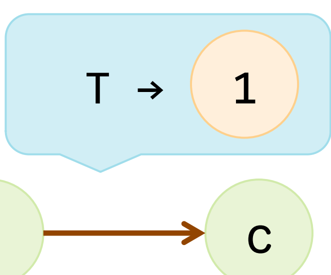


文法

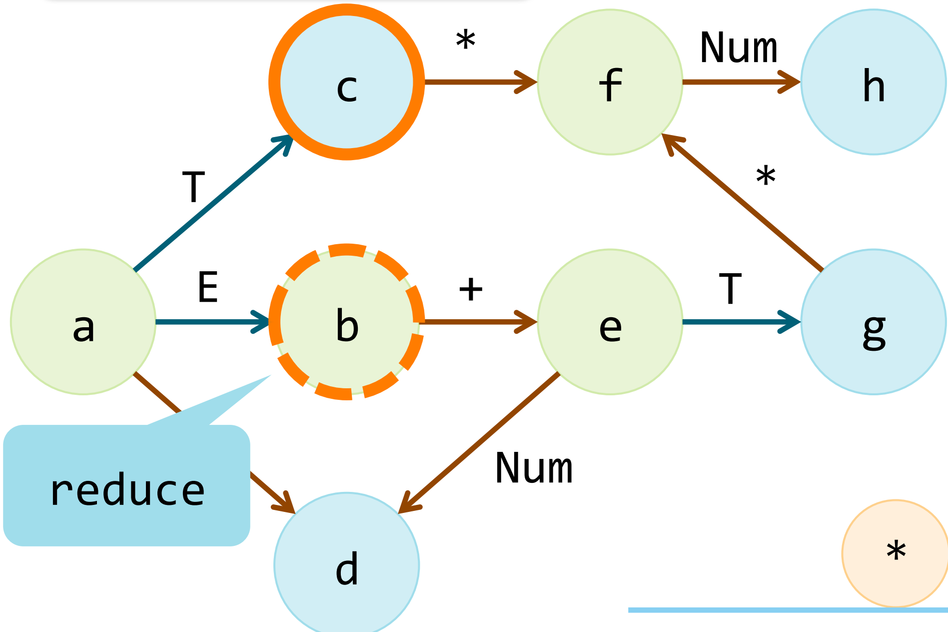
E → E + T
E → T
T → T * Num
T → Num



入力



look-ahead



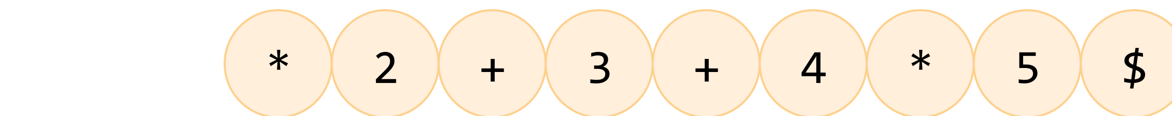
文法

E → E + T

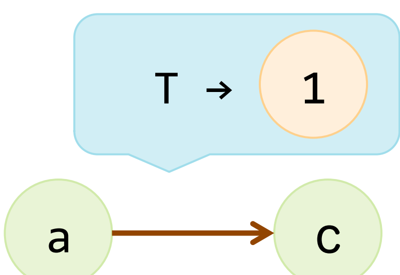
☆ E → T

T → T * Num

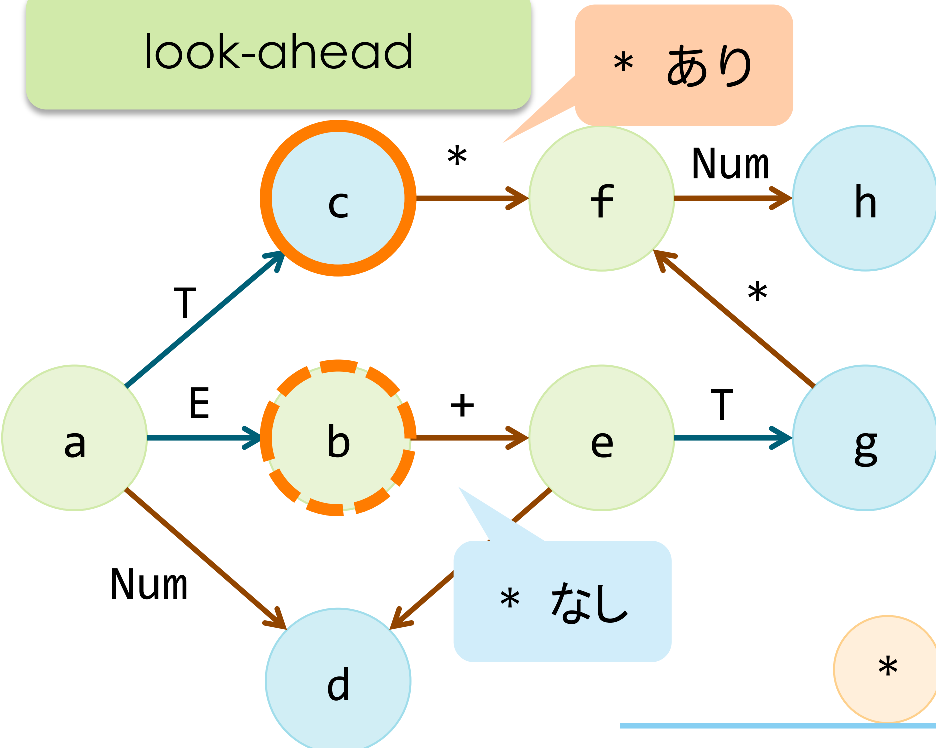
T → Num



入力



look-ahead



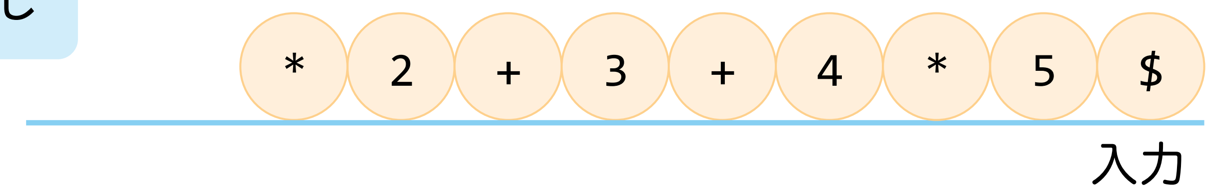
文法

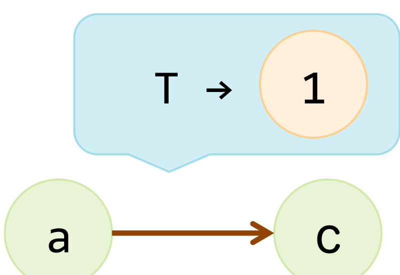
E → E + T

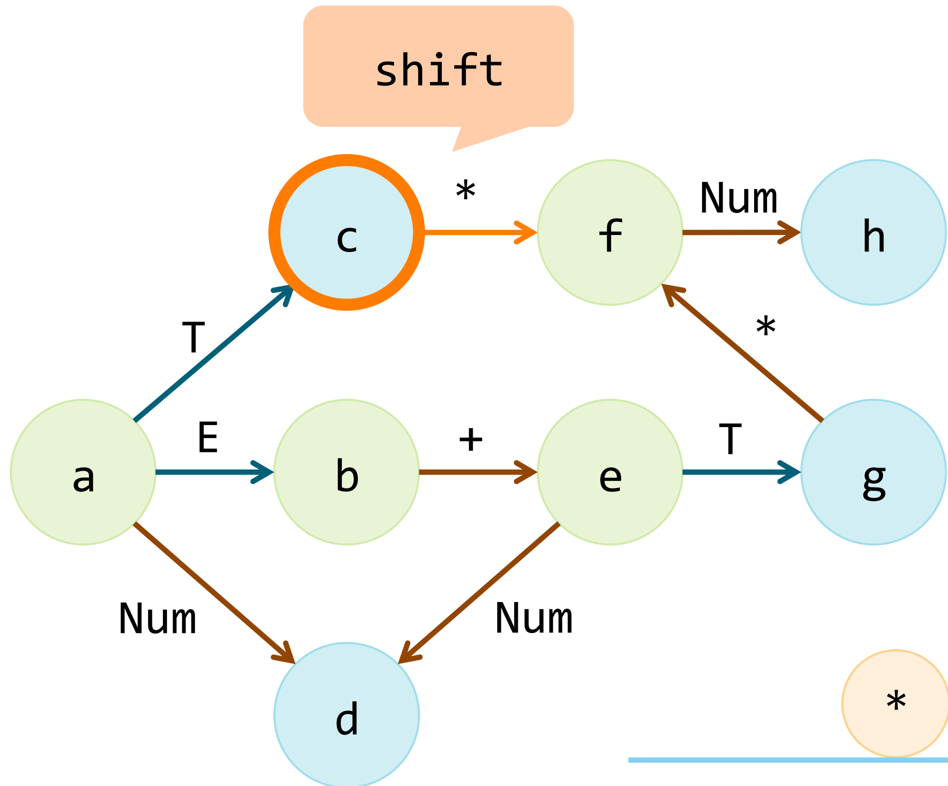
E → T

T → T * Num

T → Num

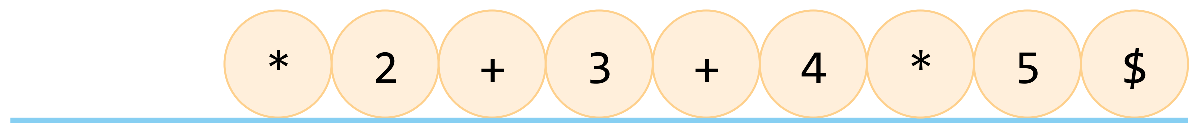




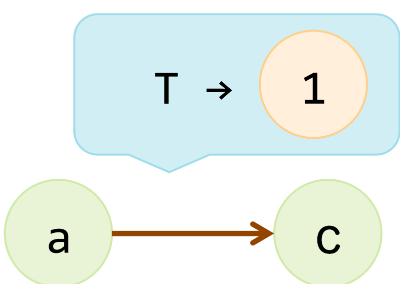


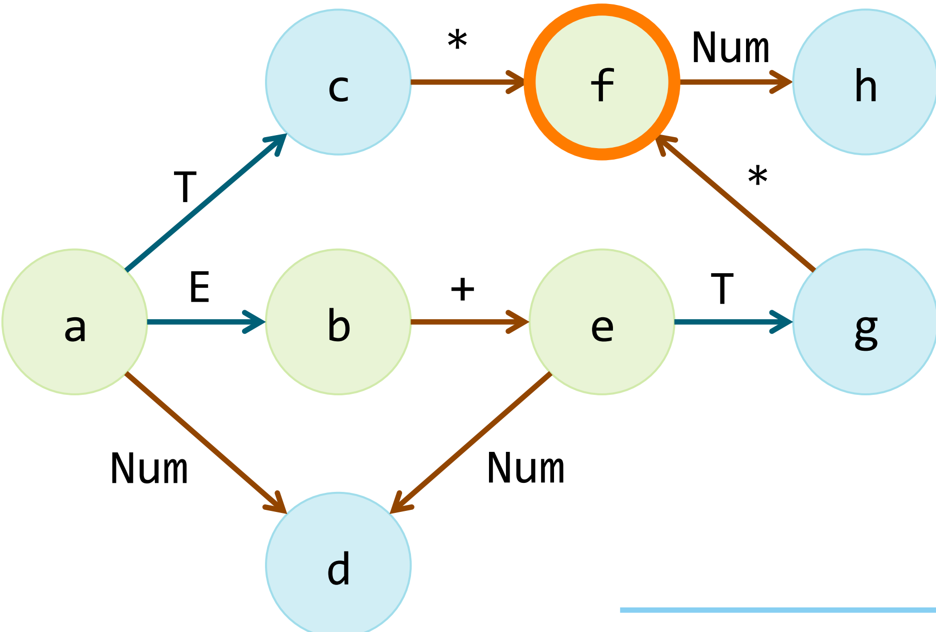
文法

E → E + T
E → T
T → T * Num
T → Num



入力





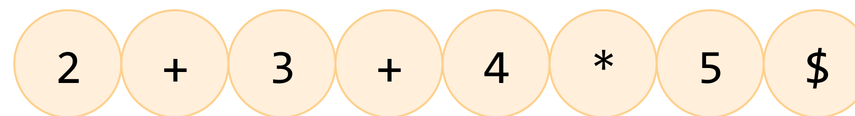
文法

$E \rightarrow E + T$

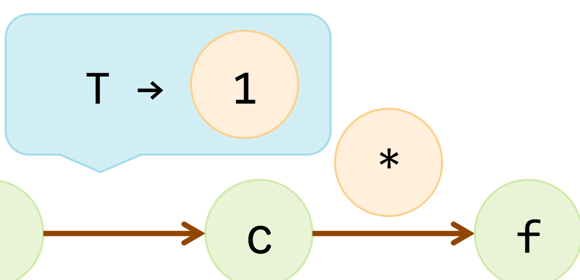
$E \rightarrow T$

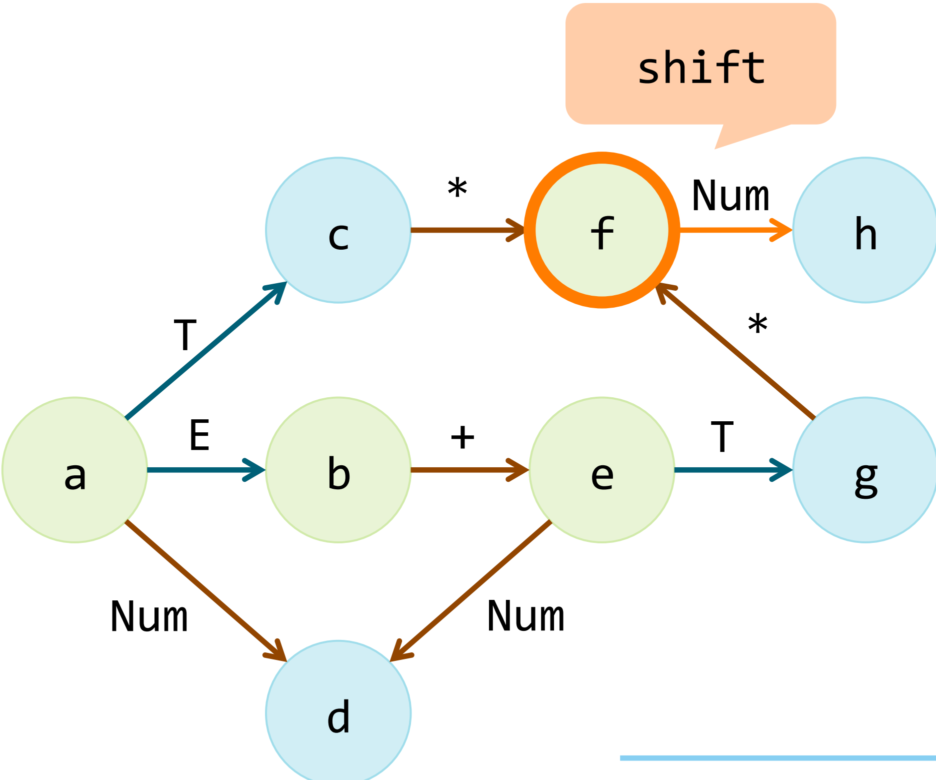
$T \rightarrow T * Num$

$T \rightarrow Num$



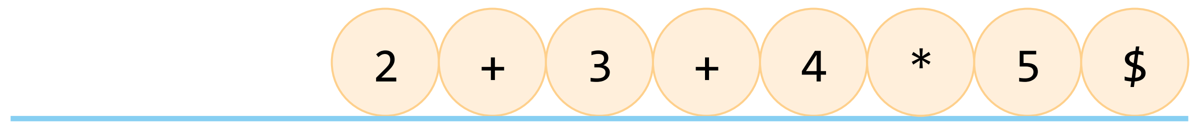
入力



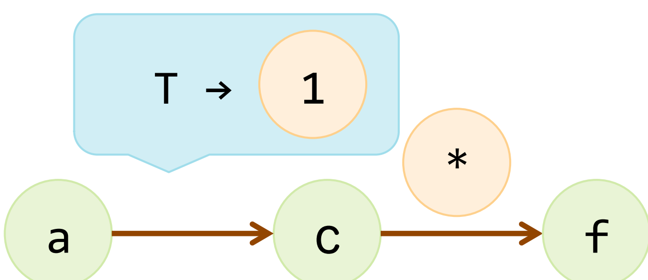


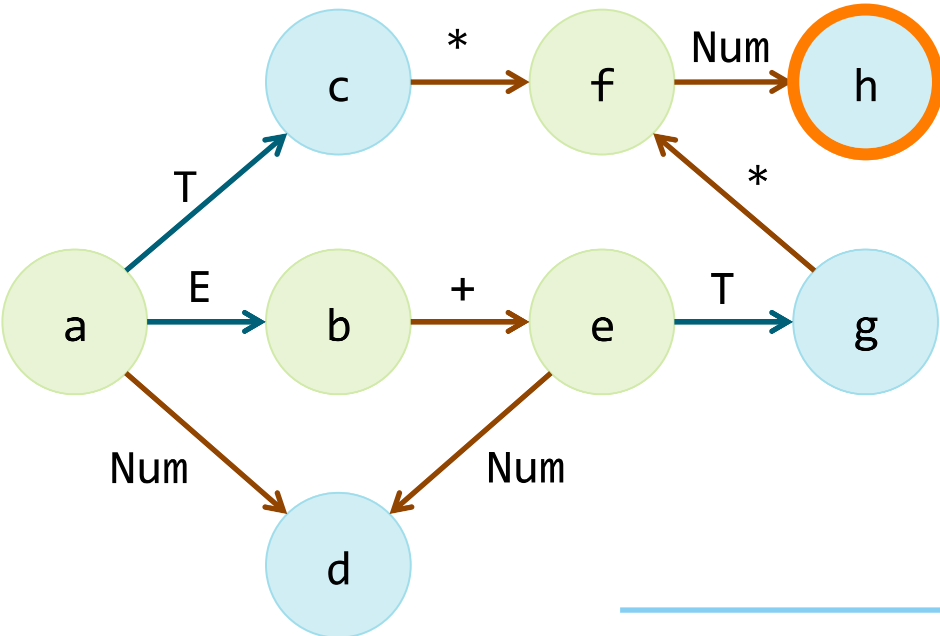
文法

E → E + T
E → T
T → T * Num
T → Num



入力





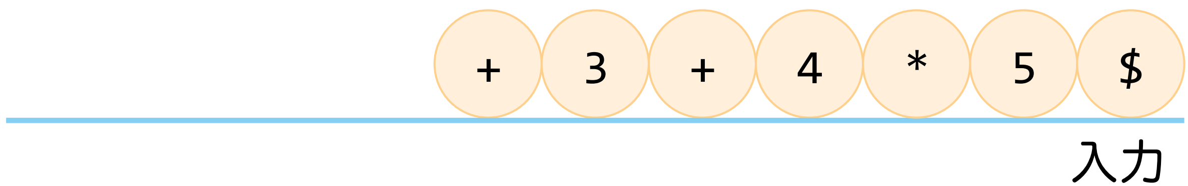
文法

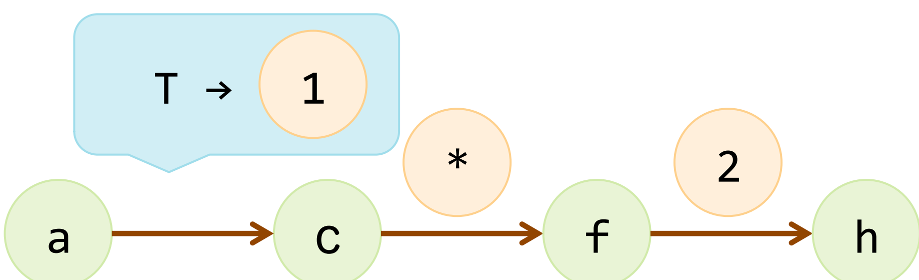
E → E + T

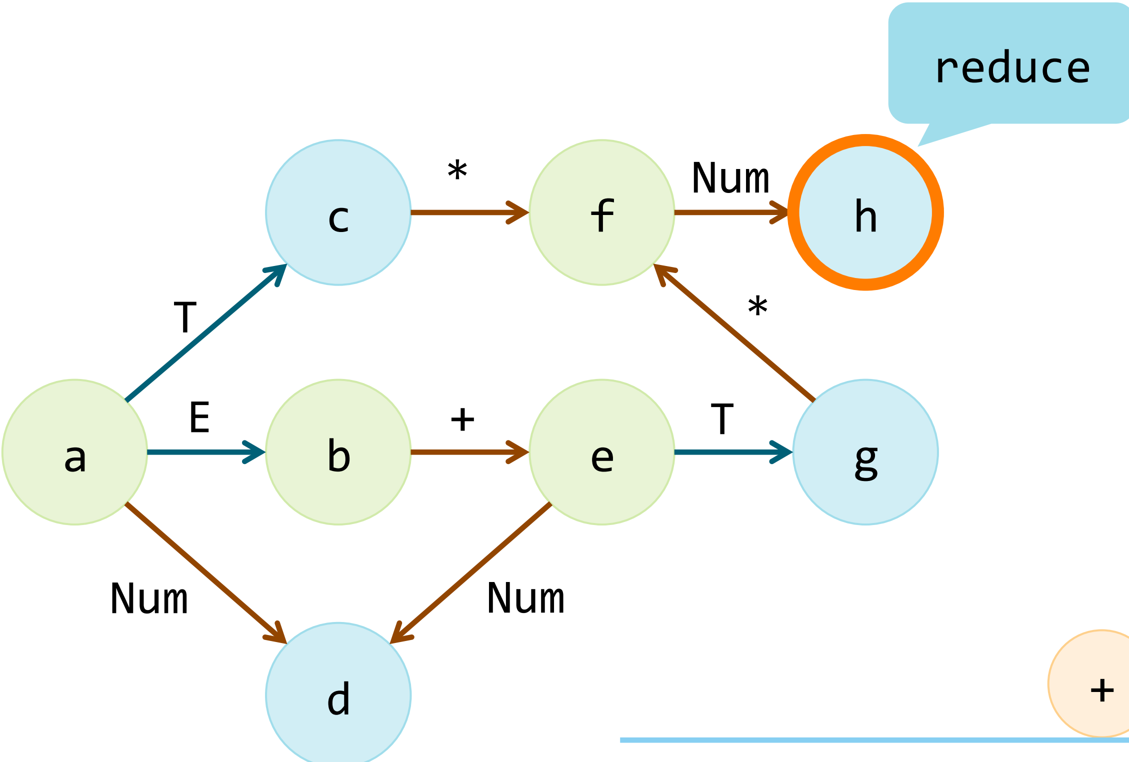
E → T

T → T * Num

T → Num







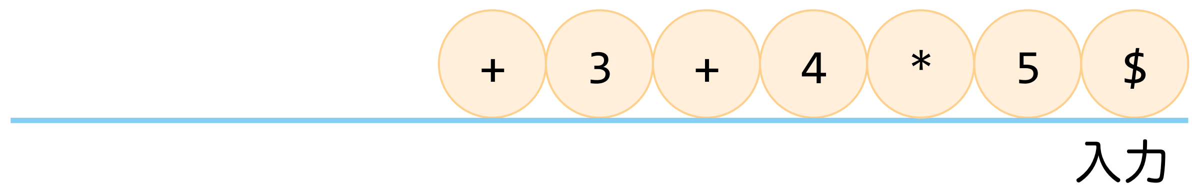
文法

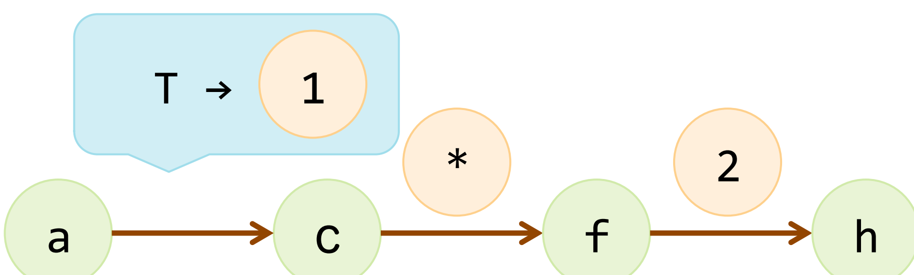
E → E + T

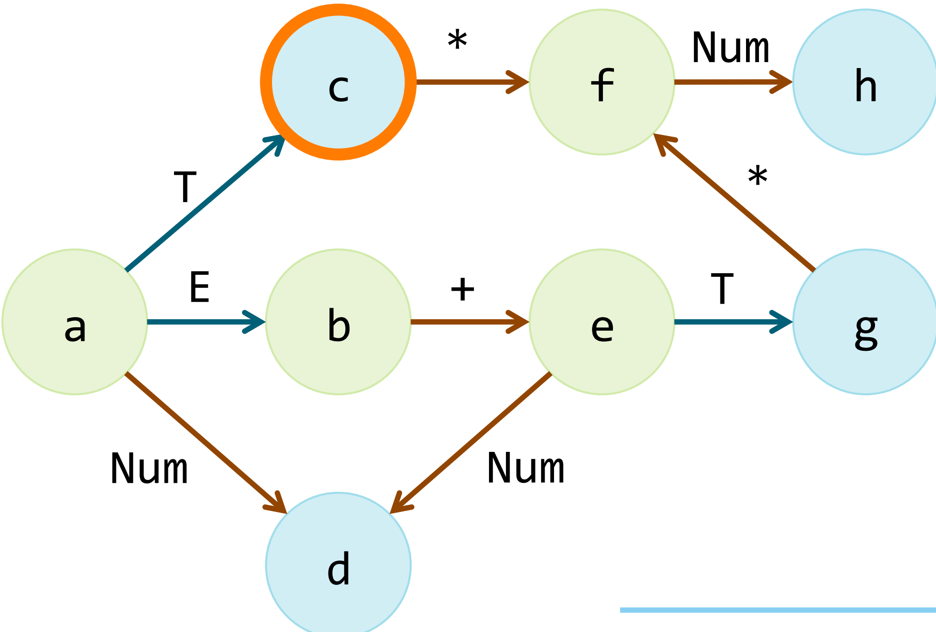
E → T

T → T * Num

T → Num







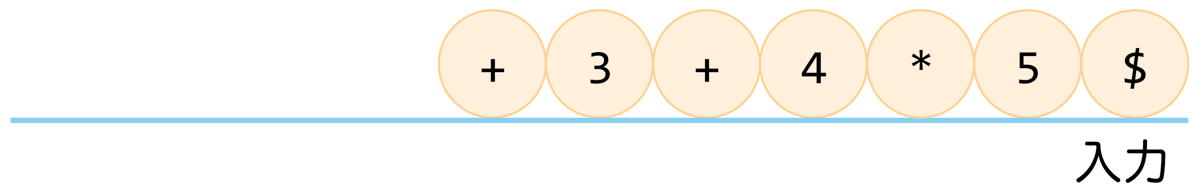
文法

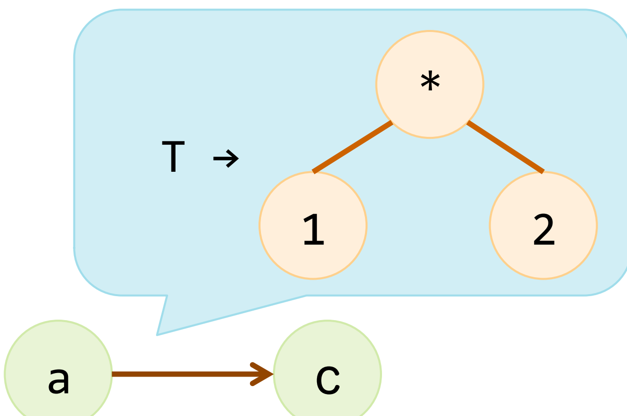
E → E + T

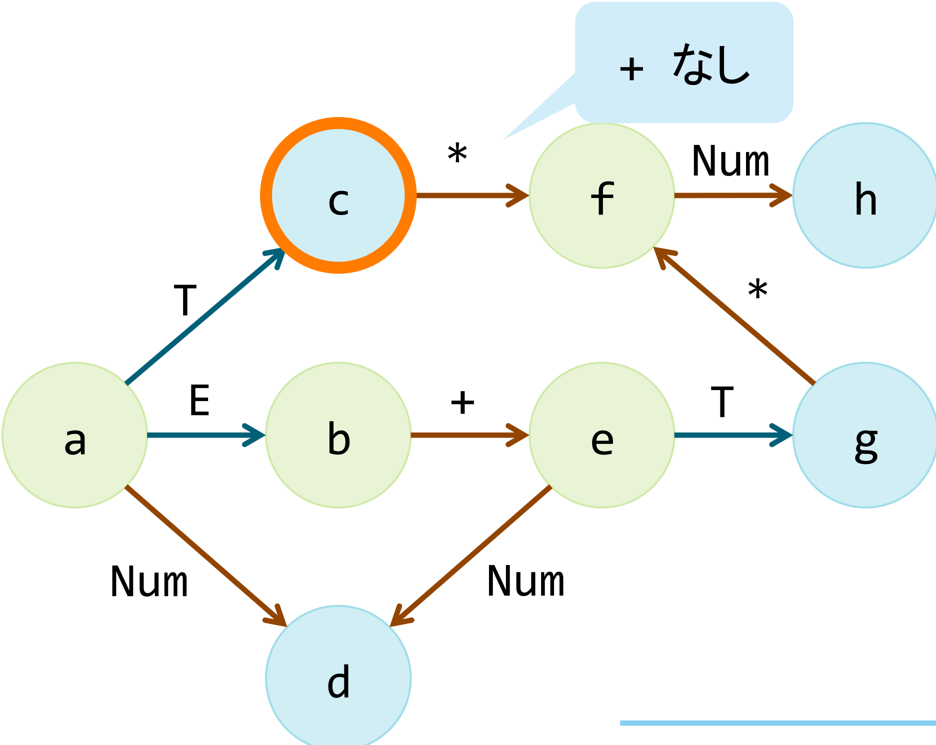
E → T

T → T * Num

T → Num

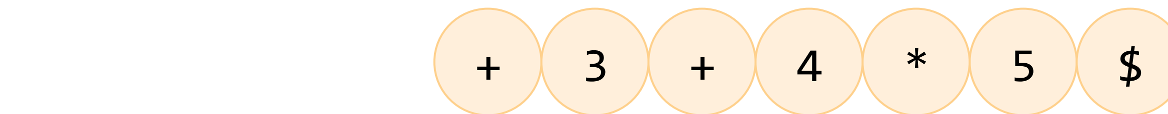




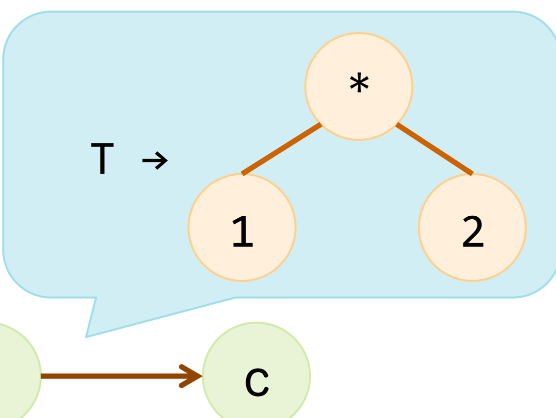


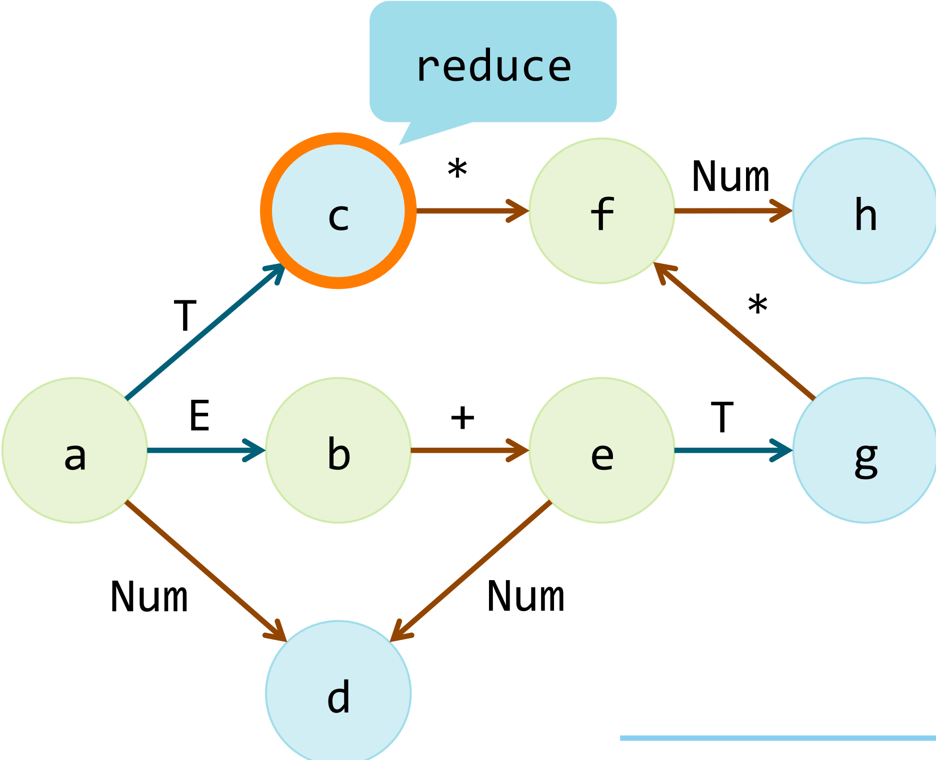
文法

E → E + T
E → T
T → T * Num
T → Num



入力



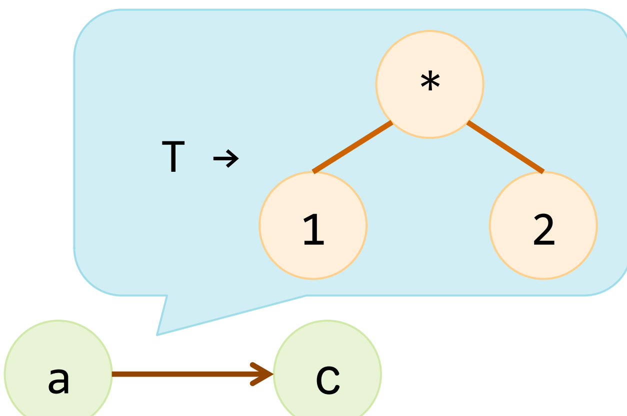


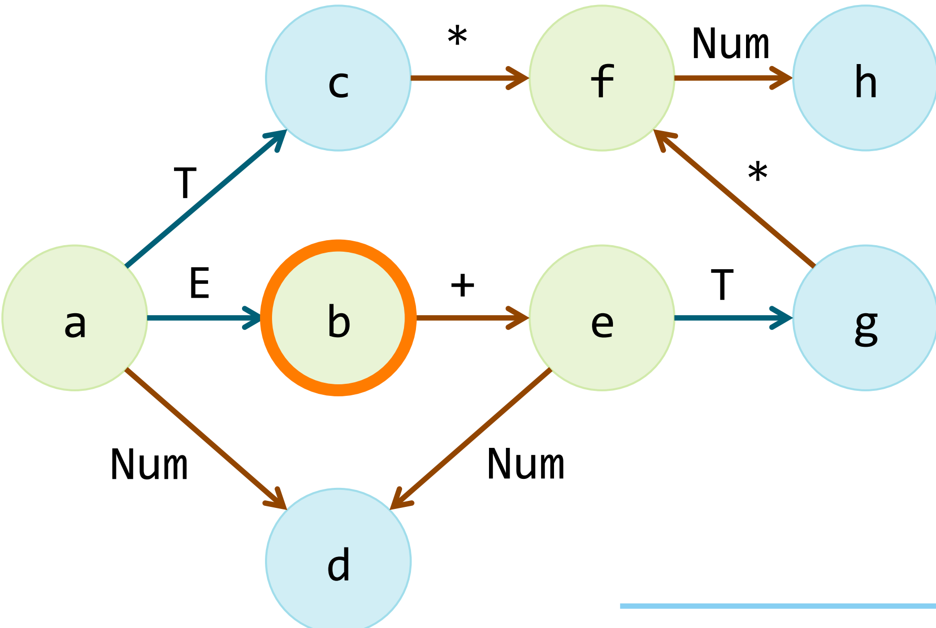
文法

E → E + T
☆ E → T
T → T * Num
T → Num

+ 3 + 4 * 5 $

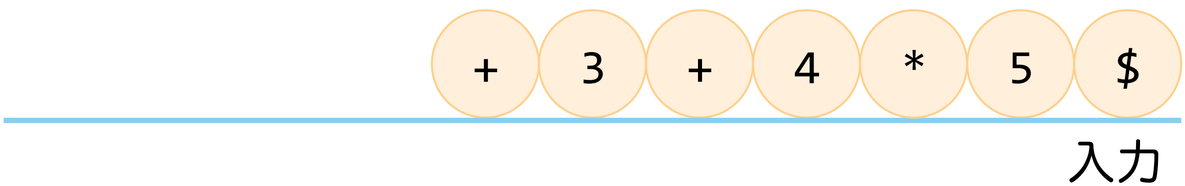
入力

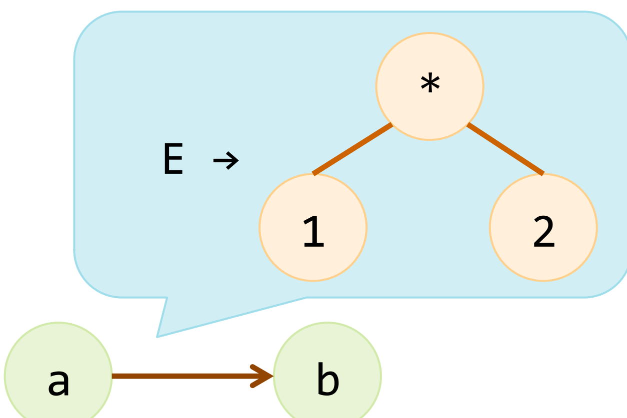


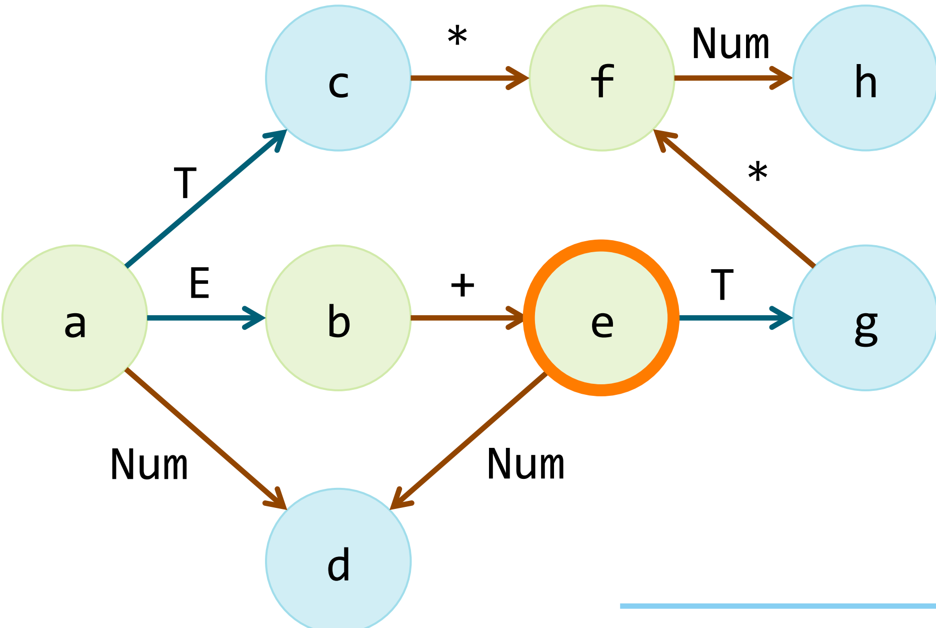


文法

E → E + T
E → T
T → T * Num
T → Num

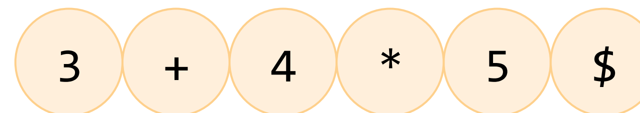




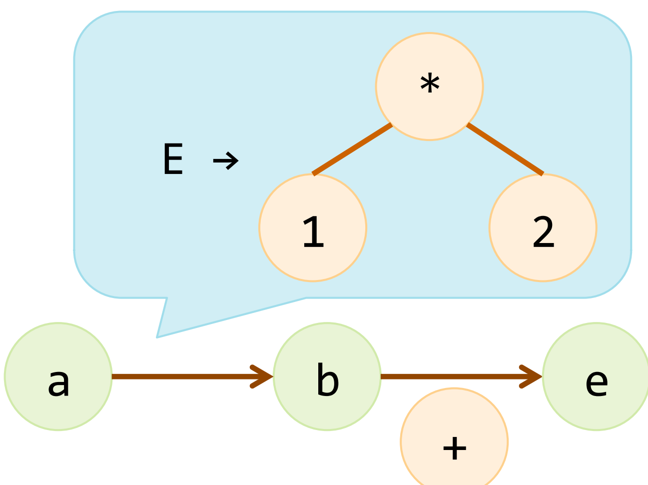


文法

E → E + T
E → T
T → T * Num
T → Num

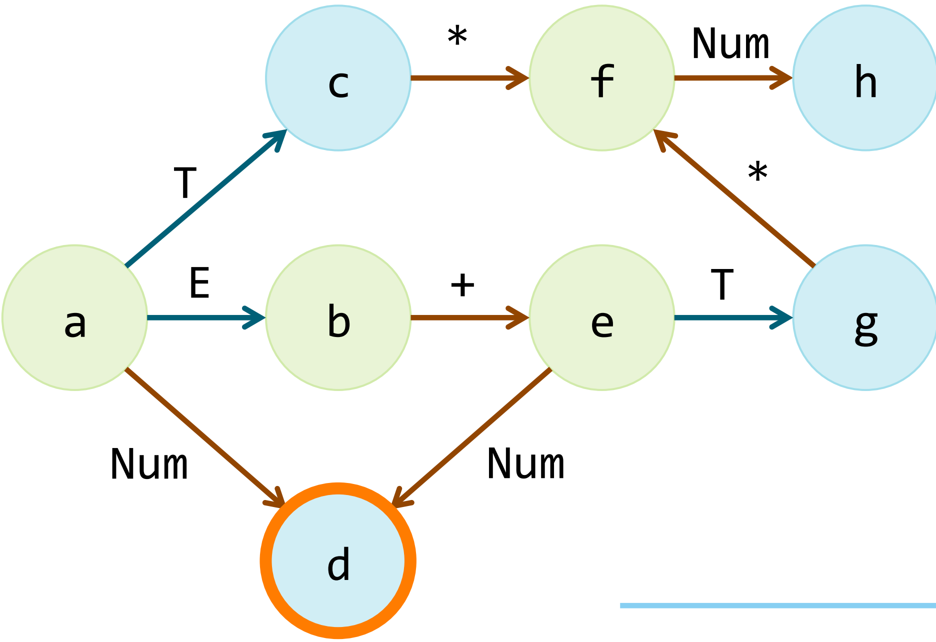


入力

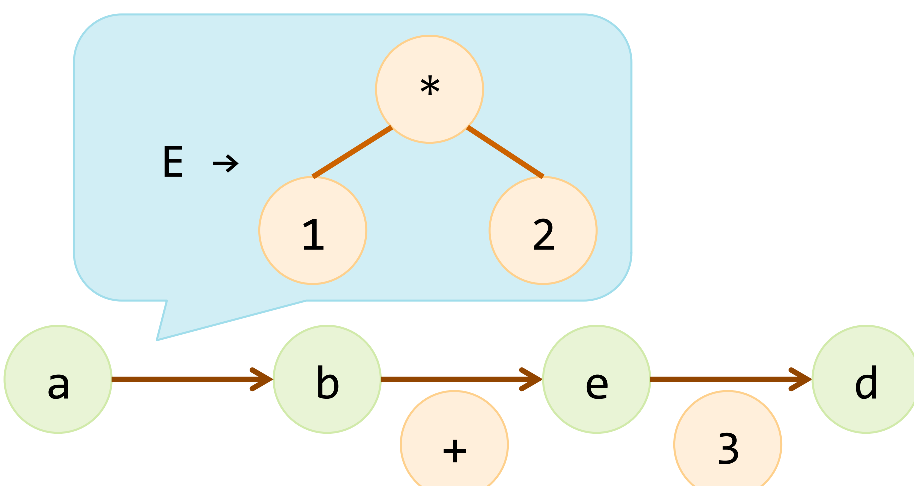


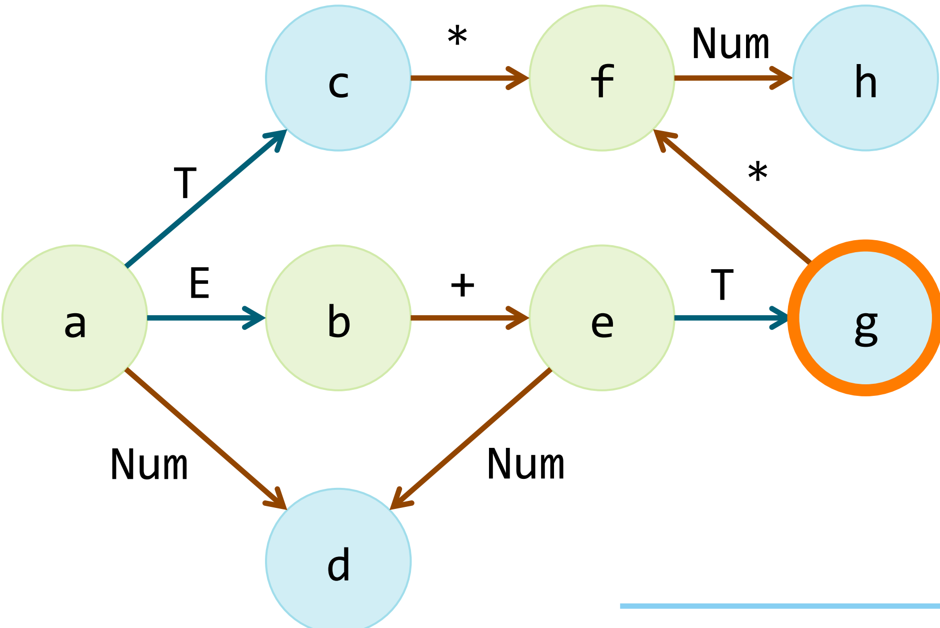
文法

E → E + T
E → T
T → T * Num
T → Num



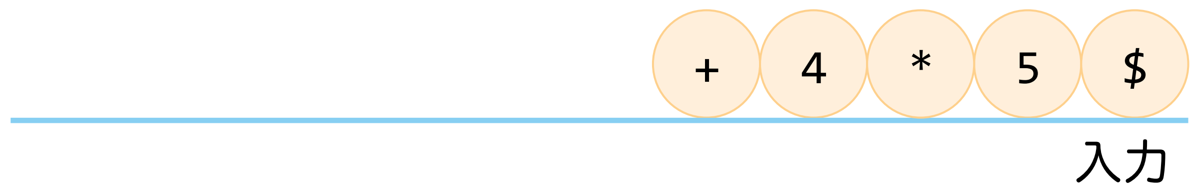
入力

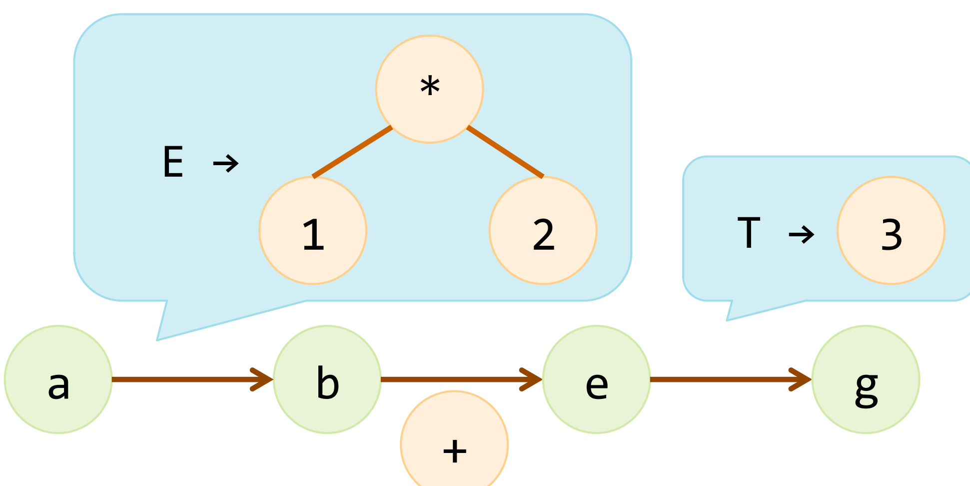




文法

E → E + T
E → T
T → T * Num
T → Num





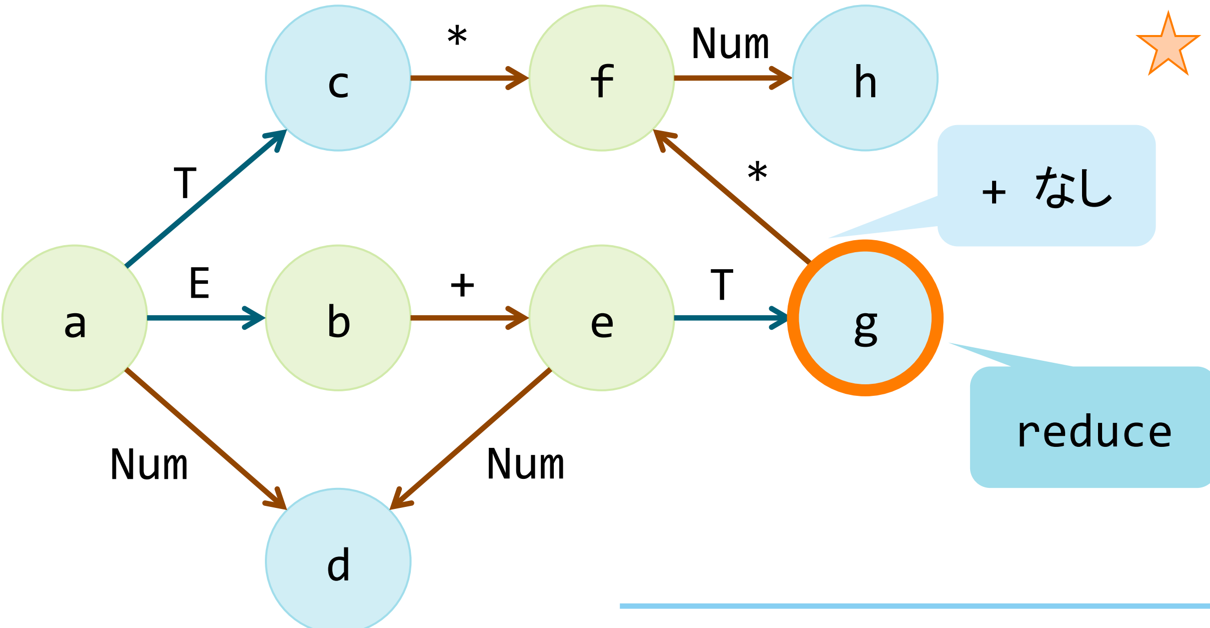
文法

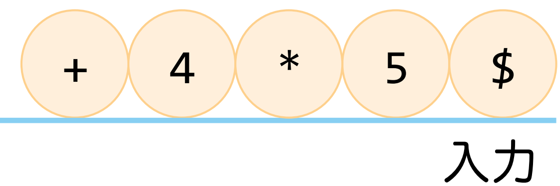
☆ E → E + T

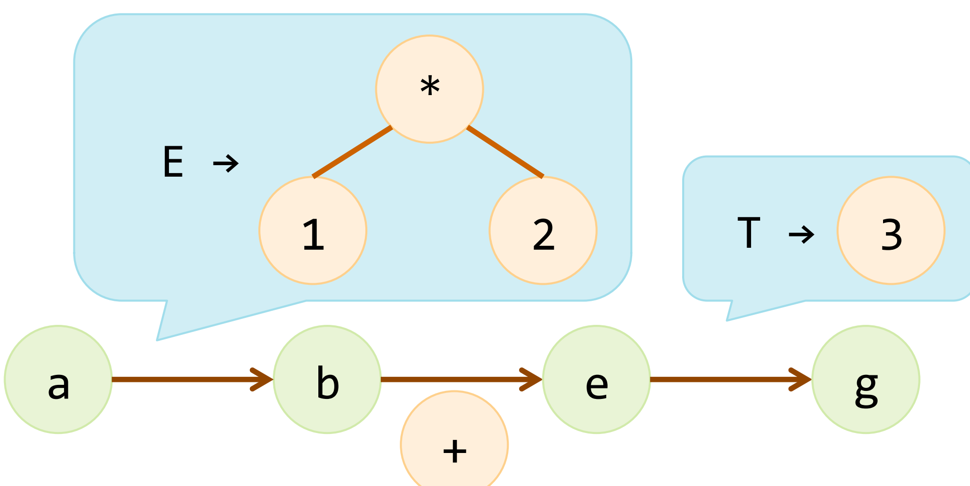
E → T

T → T * Num

T → Num

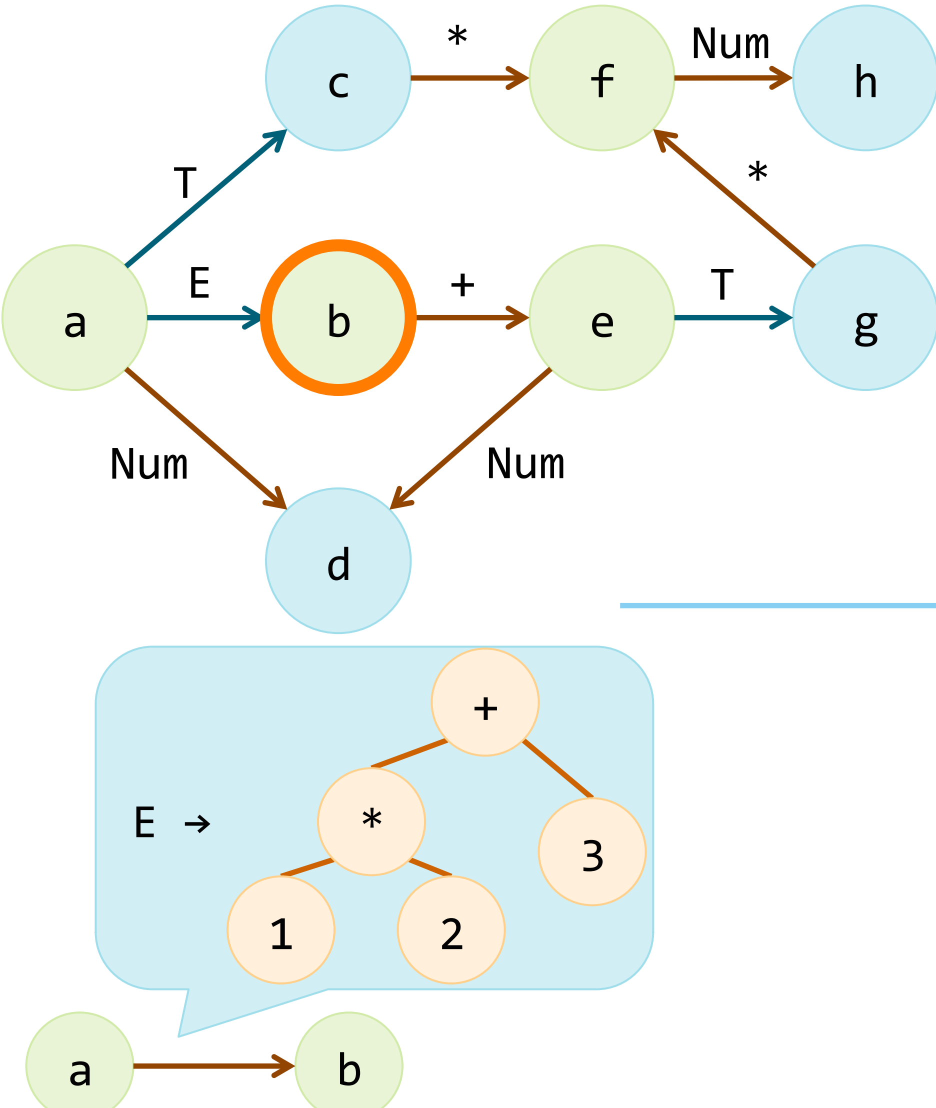


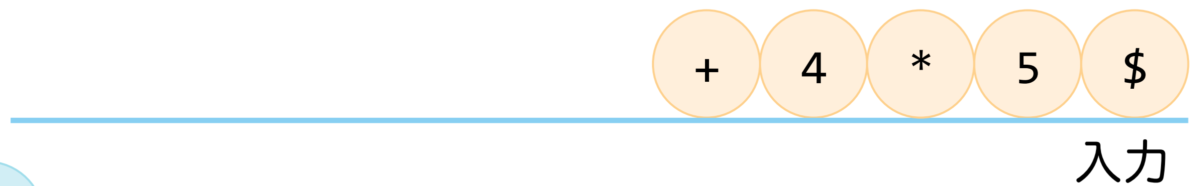


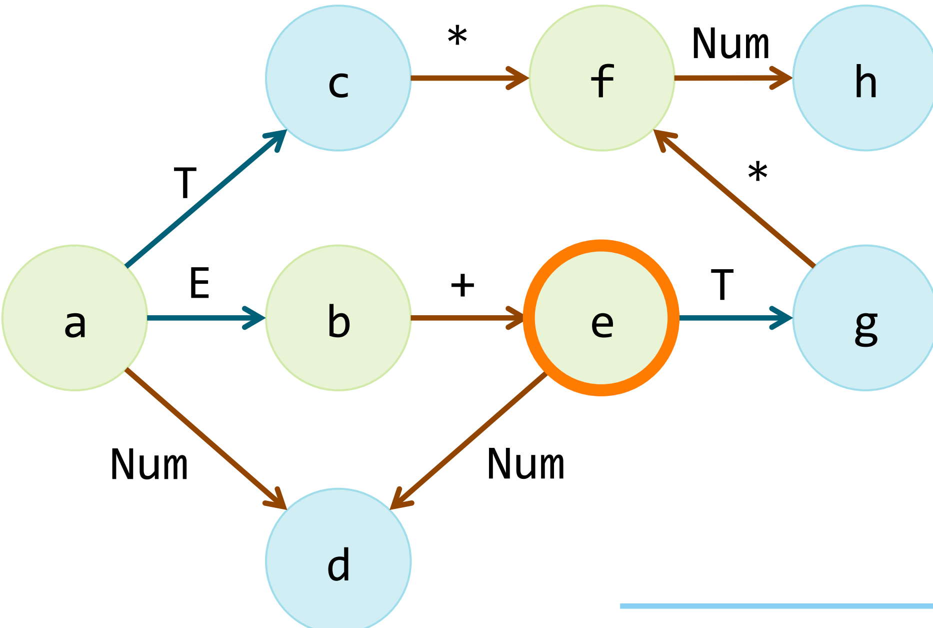


文法

E → E + T
E → T
T → T * Num
T → Num



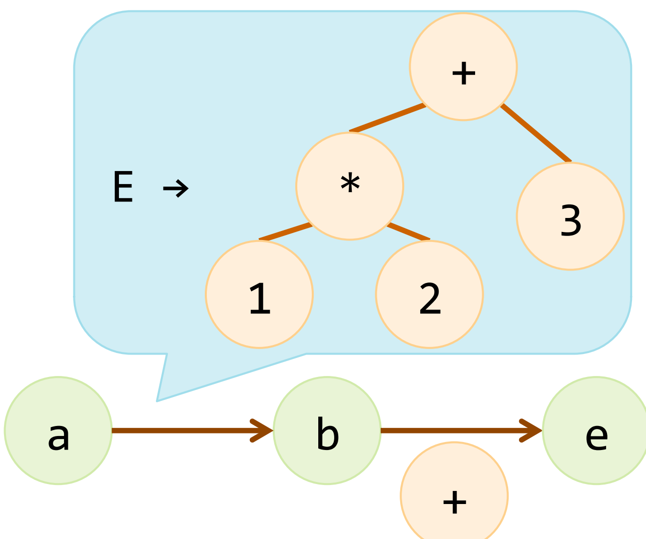


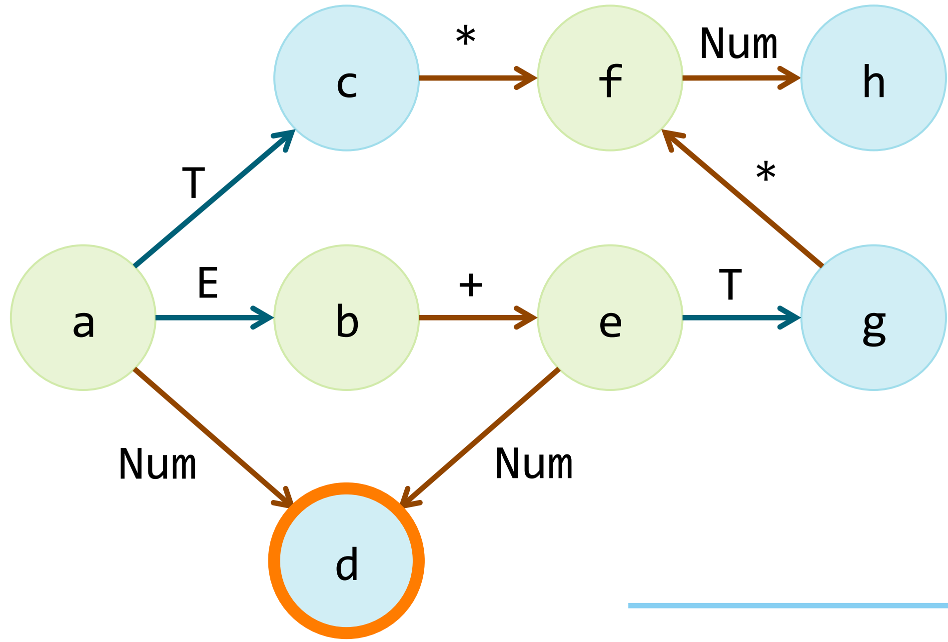


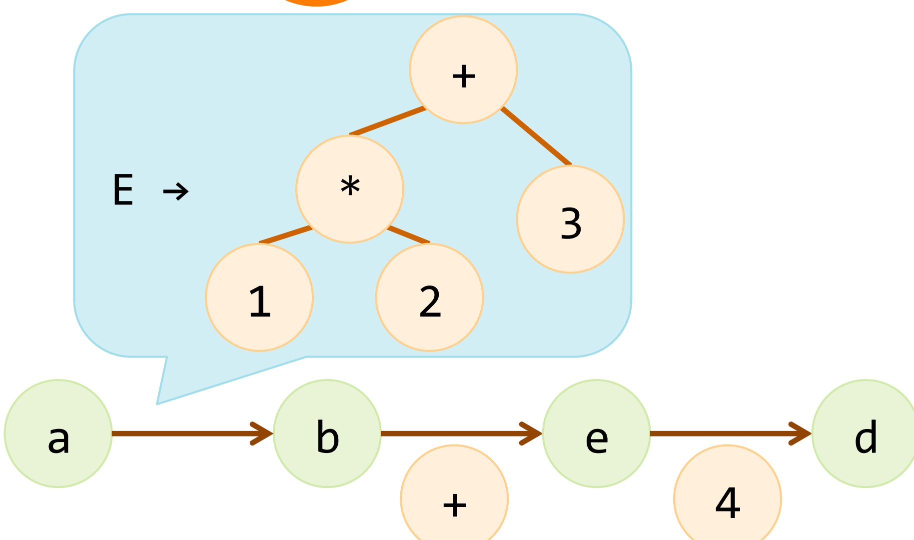
文法

E → E + T
E → T
T → T * Num
T → Num

入力







文法

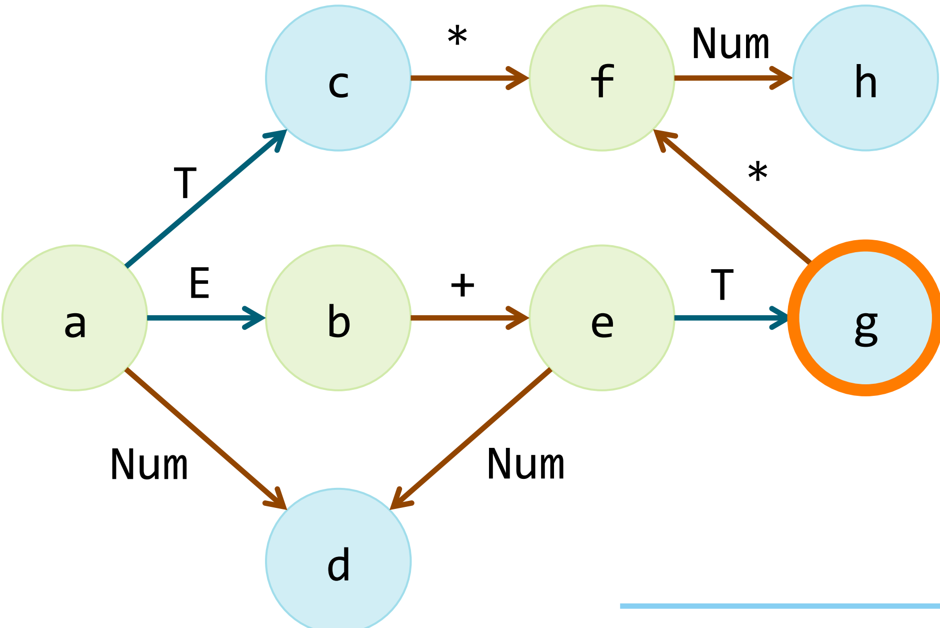
E → E + T

E → T

T → T * Num

☆ T → Num

入力



文法

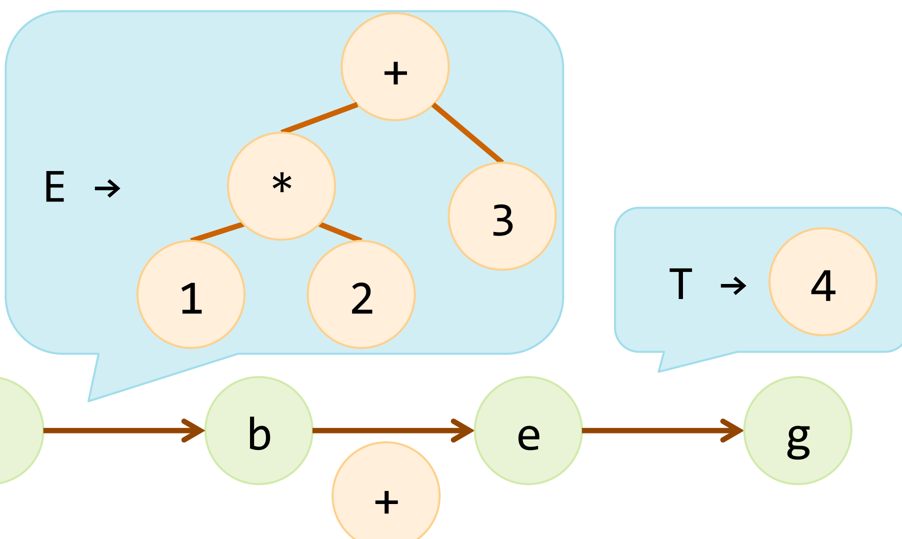
E → E + T

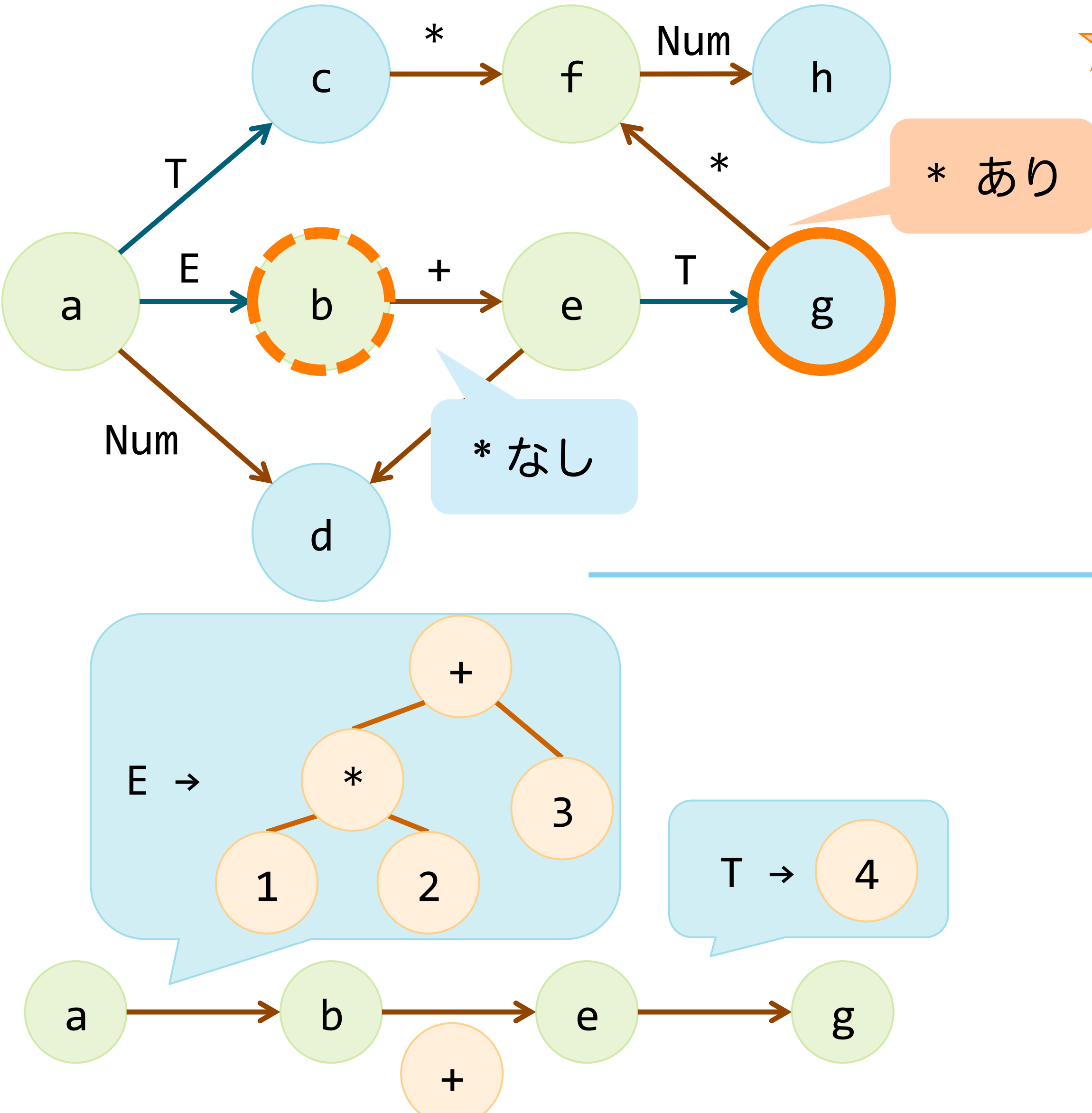
E → T

T → T * Num

T → Num

入力





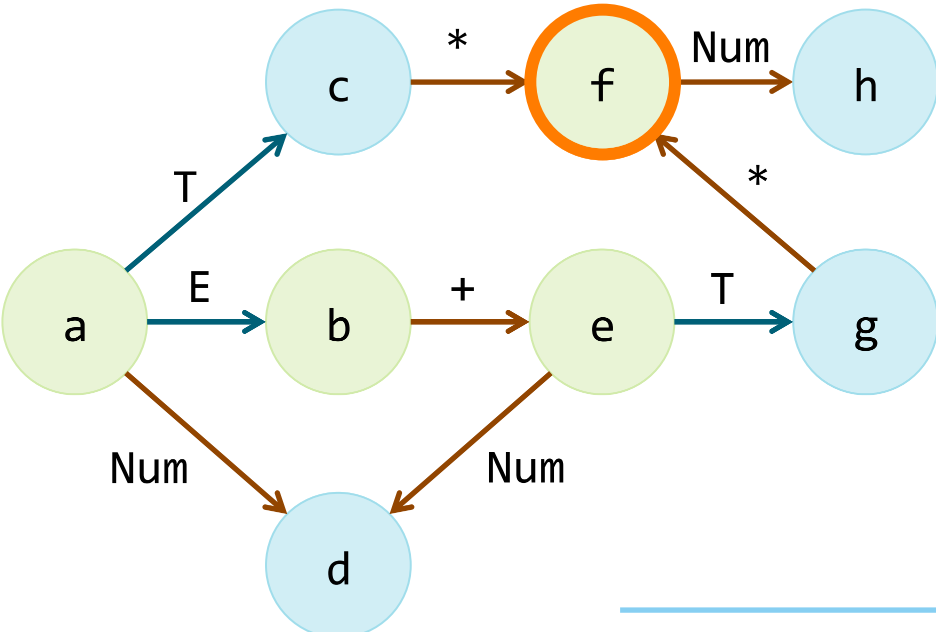
文法

☆ E → E + T

E → T

T → T * Num

T → Num



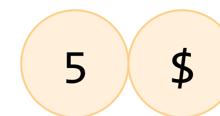
文法

E → E + T

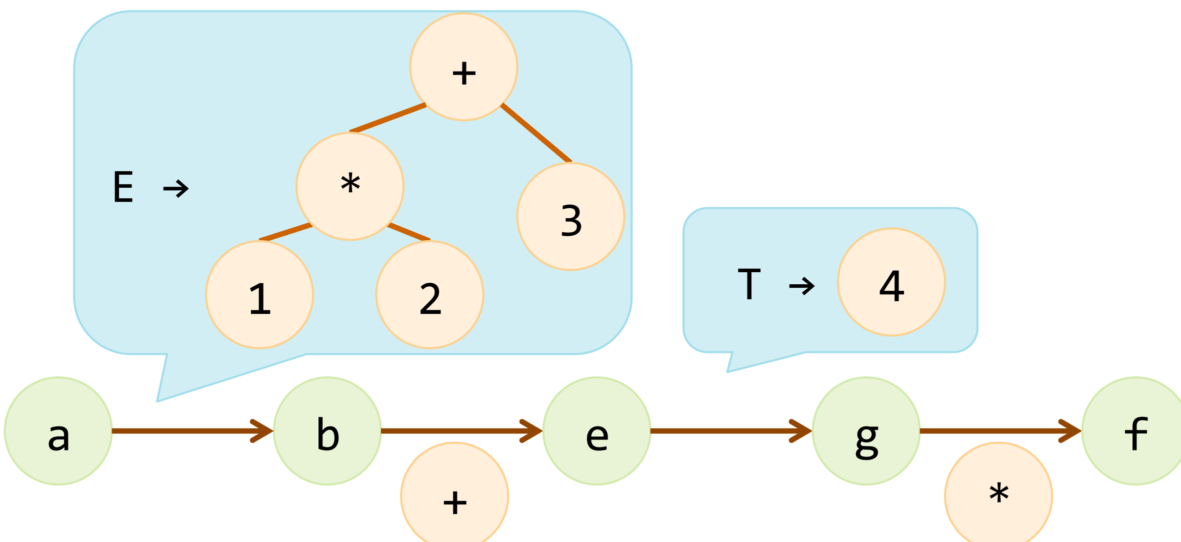
E → T

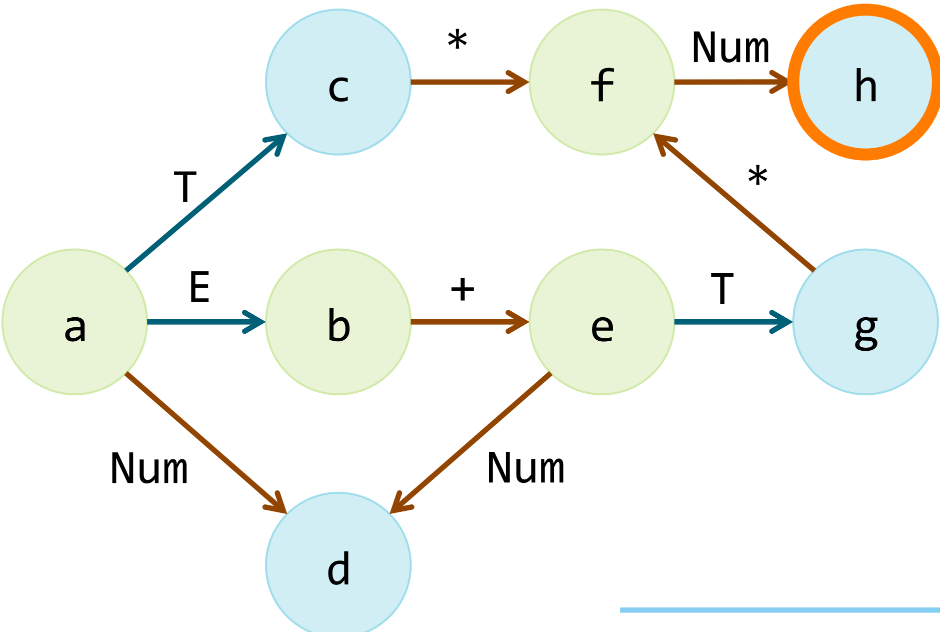
T → T * Num

T → Num



入力





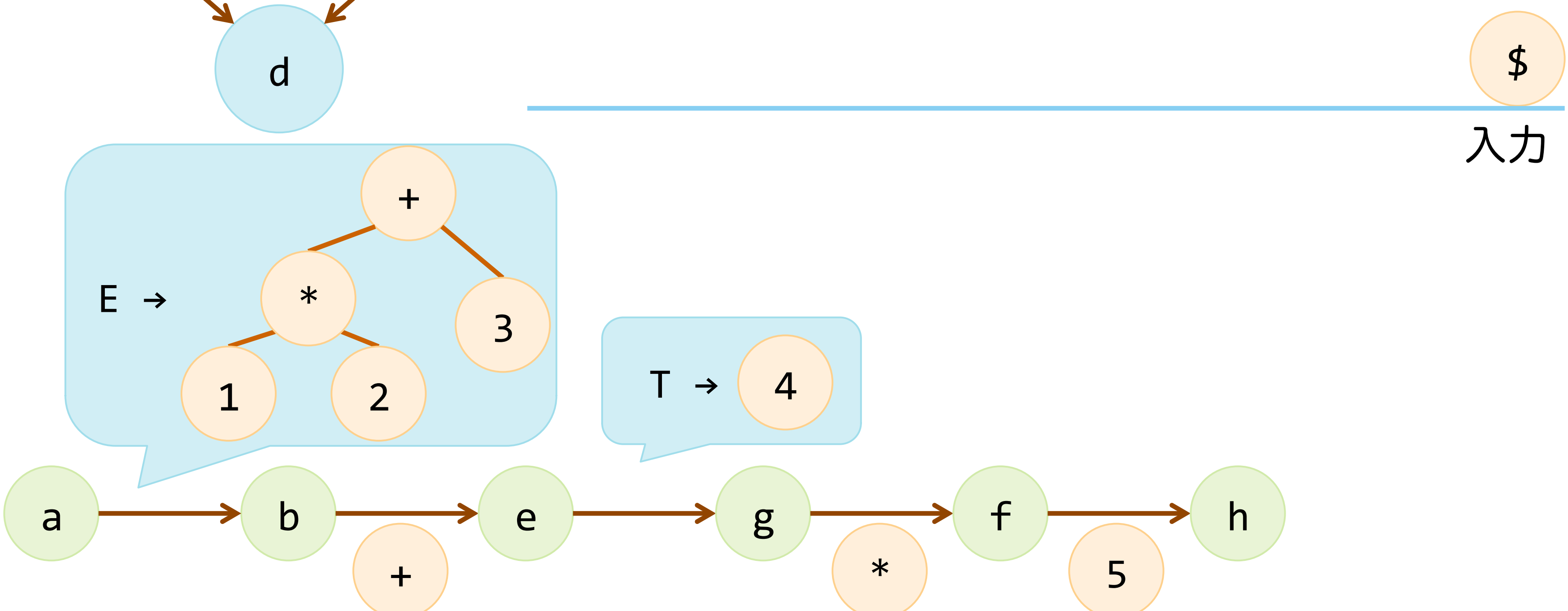
文法

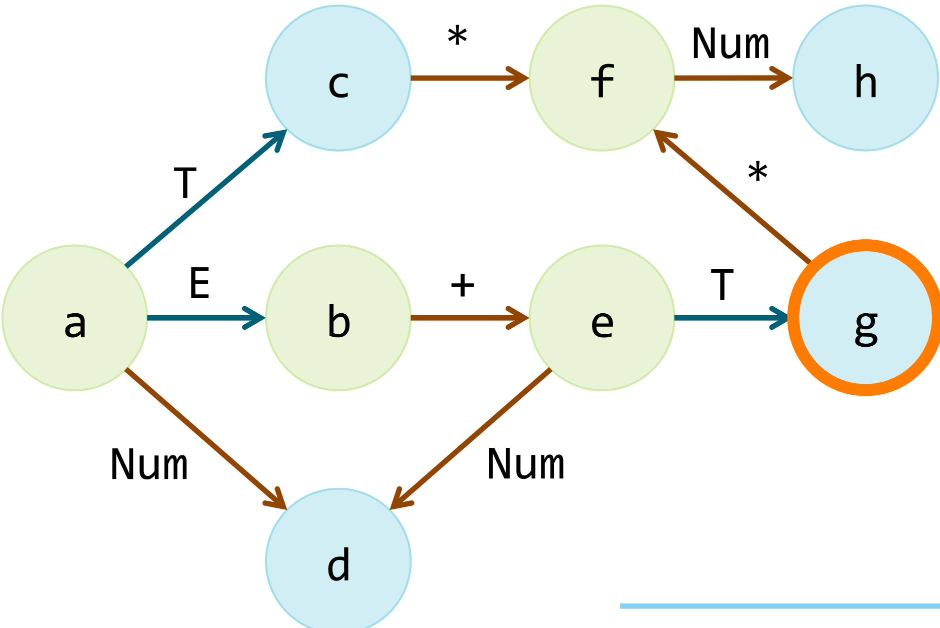
E → E + T

E → T

T → T * Num

T → Num

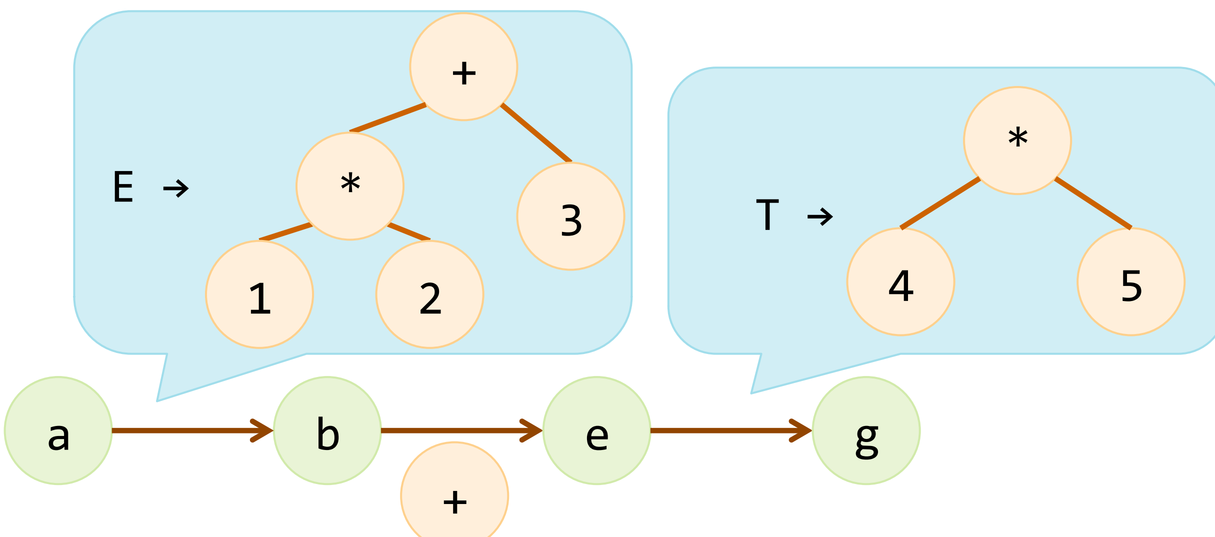


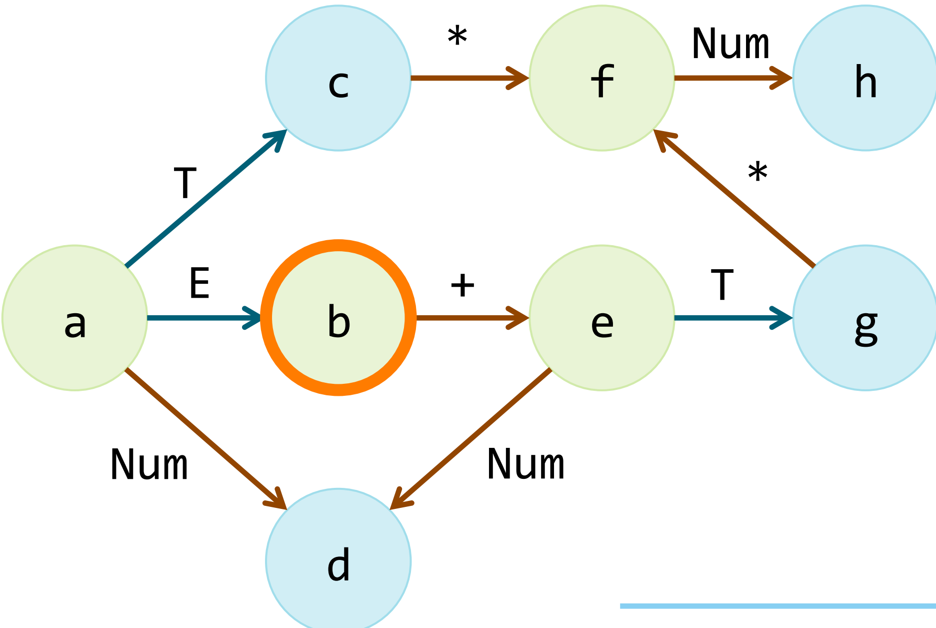


文法

- E → E + T
- E → T
- T → T * Num
- T → Num

入力

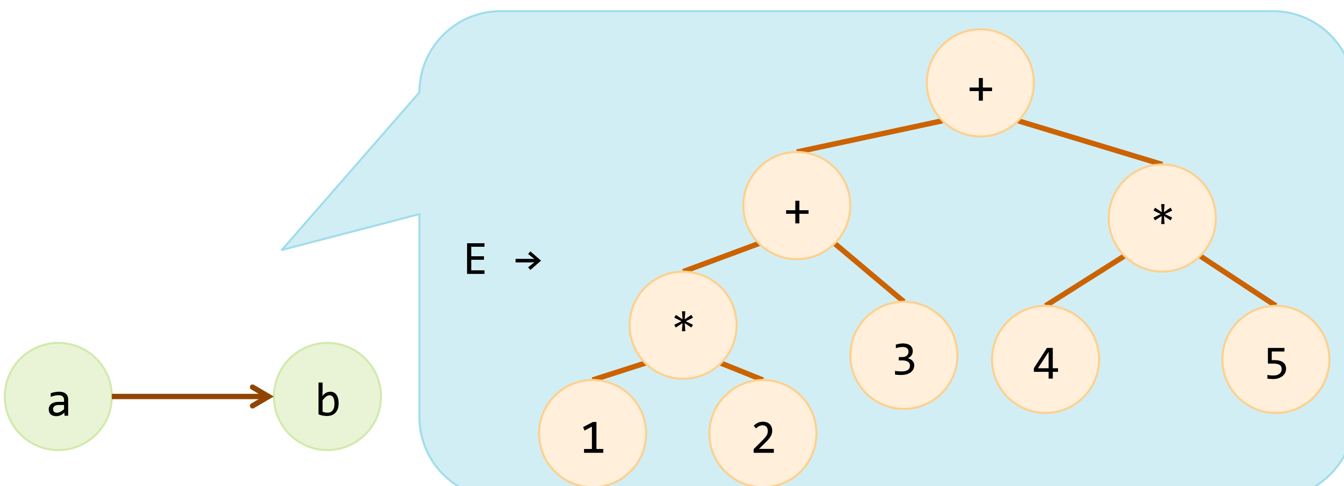




文法

E → E + T
E → T
T → T * Num
T → Num

入力



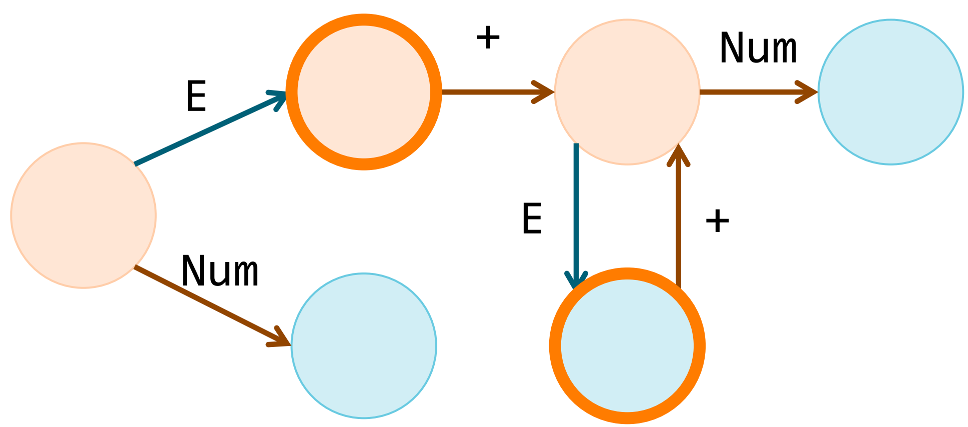
LR parsing [Knuth 1965]

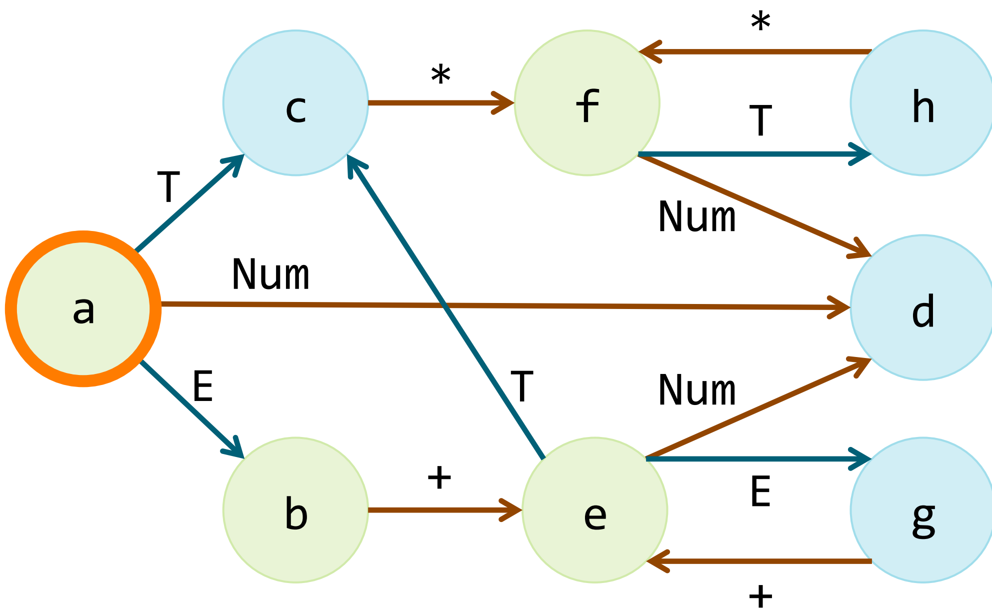
- 通常は LR 構文解析表を利用する
- 構文解析表の作り方により受理文法が異なる
  - SLR(1), LALR(1) [DeRemer 1969, 1971] など
  - 紹介した手法は SLR(1) に相当
  - 解説: www.slideshare.net/ichikaz3/lr-parsing
- 能力
  - 計算量 : O(n)
  - 言語 : Deterministic context-free language
  - 文法 : LR(1), SLR(1), LALR(1), ...

GLR parsing [Tomita 1985]

アイデア

- 非決定性オートマトンを作る
- 異なる遷移で同一状態になったものをまとめる
  - ➢ グラフ構造スタック(GSS)
  - ➢ Parse forest : 子の構文木を直接持たず、ラベルで持つ





文法

E → E + E

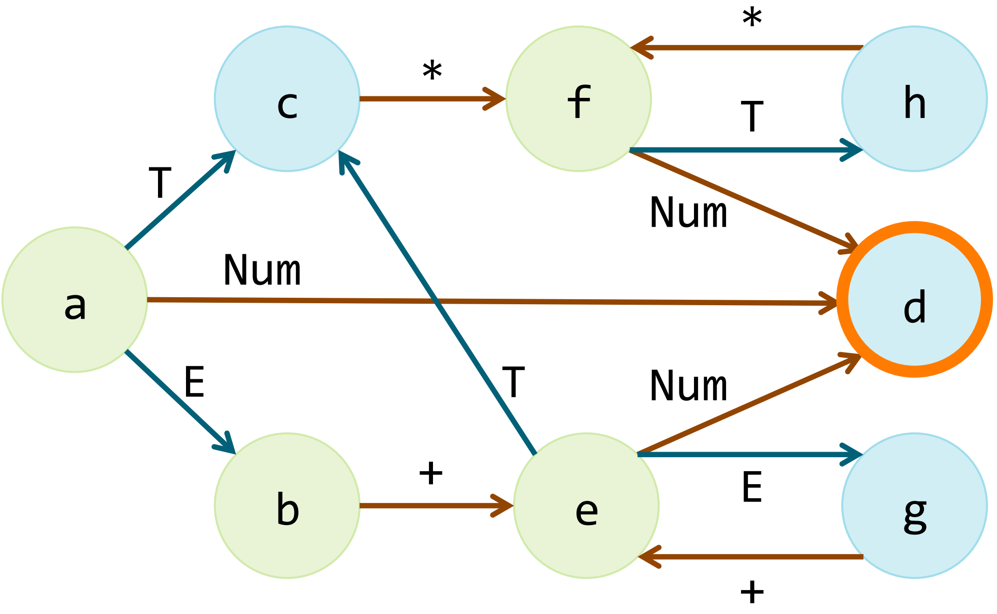
E → T

T → T * T

T → Num

1 + 2 * 3 * 4 + 5 $

入力



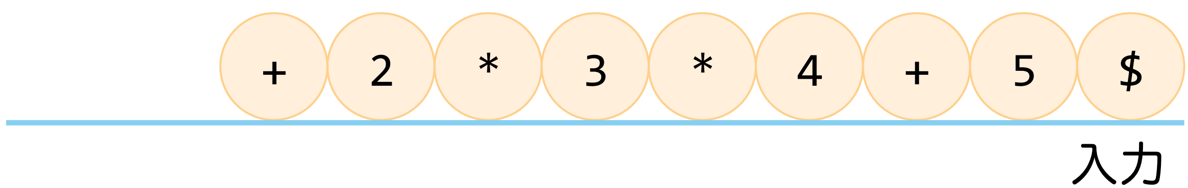
文法

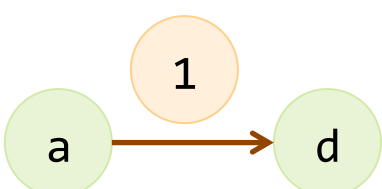
E → E + E

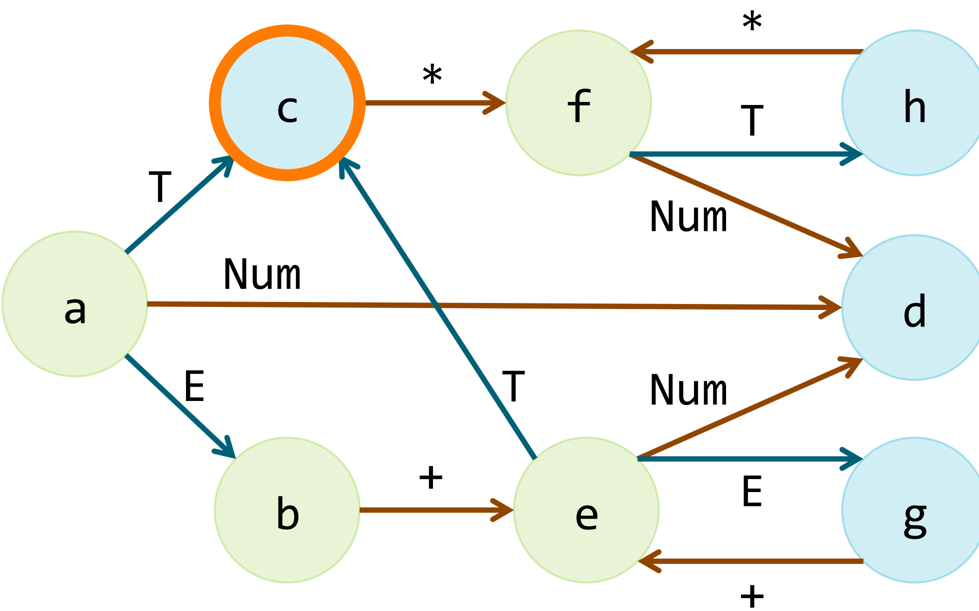
E → T

T → T * T

T → Num







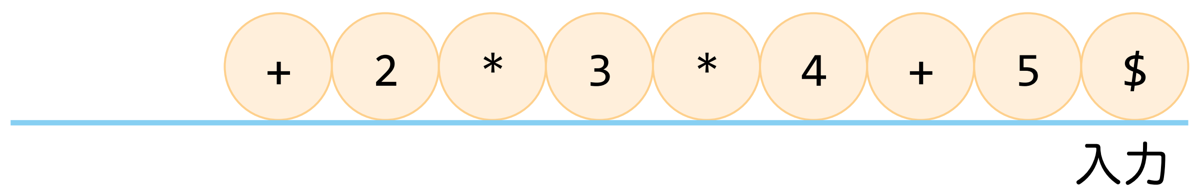
文法

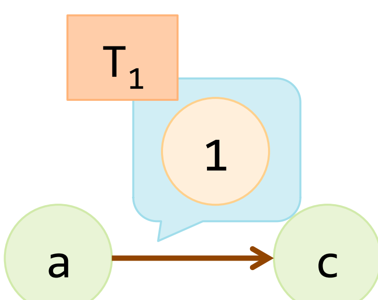
E → E + E

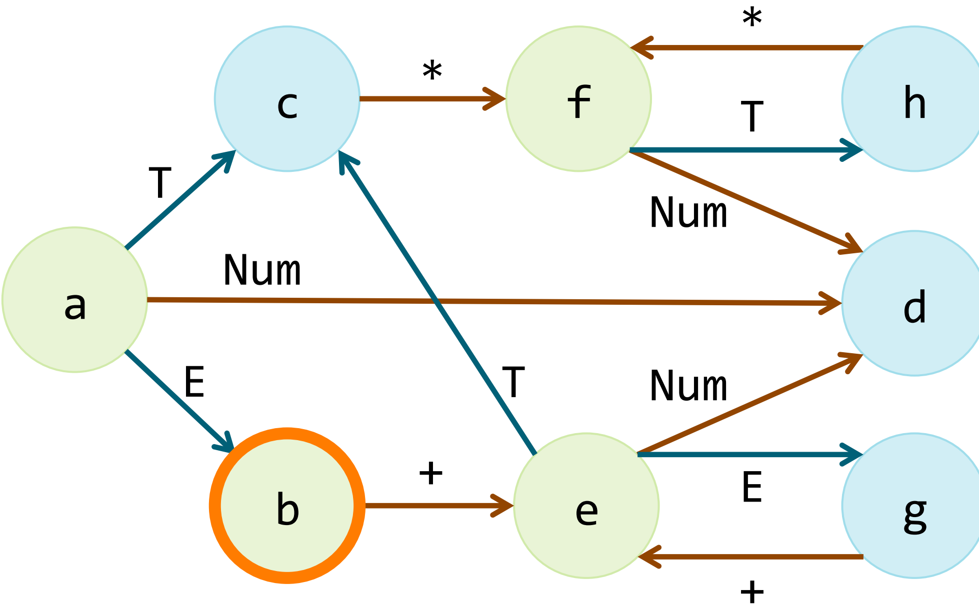
E → T

T → T * T

T → Num







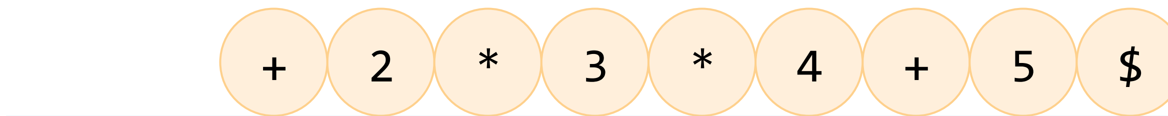
文法

$E \rightarrow E + E$

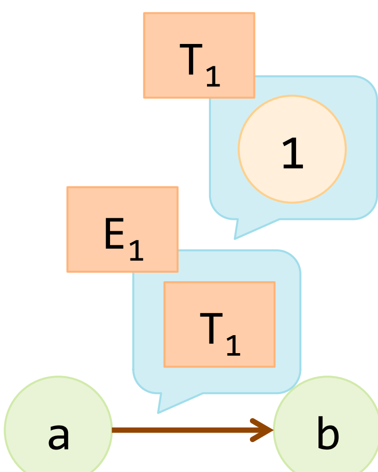
$E \rightarrow T$

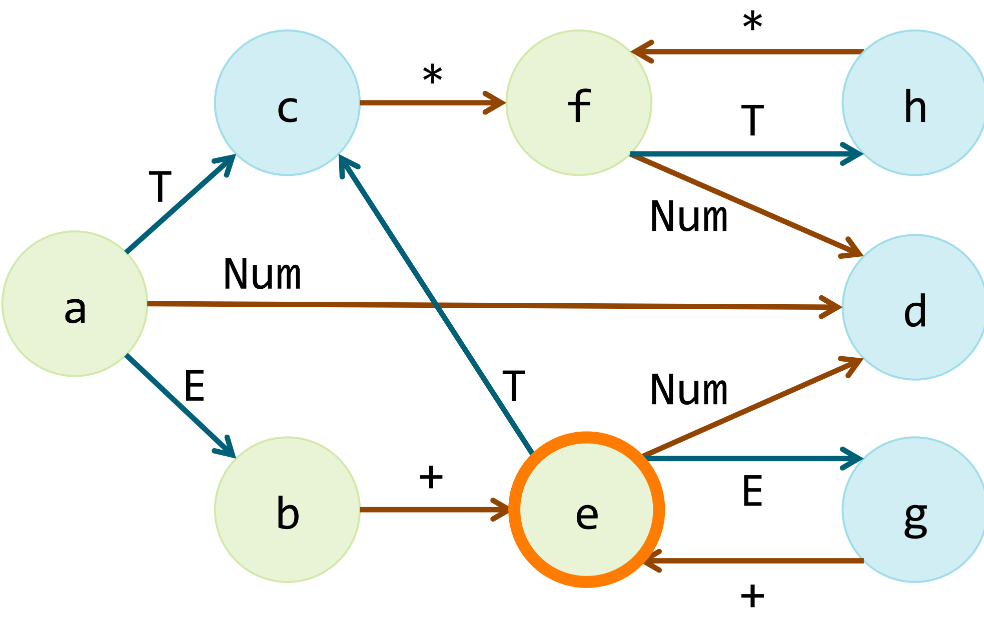
$T \rightarrow T * T$

$T \rightarrow Num$



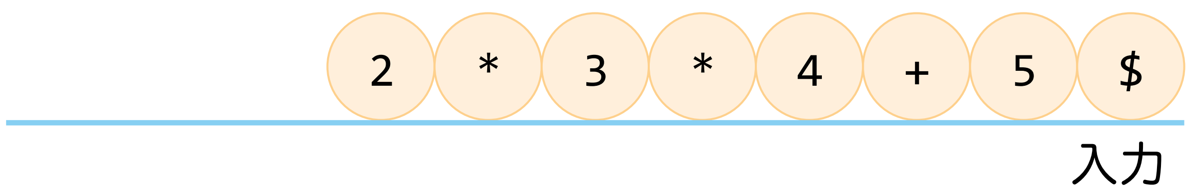
入力

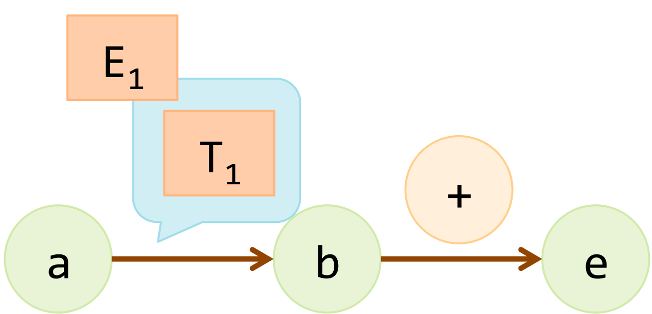


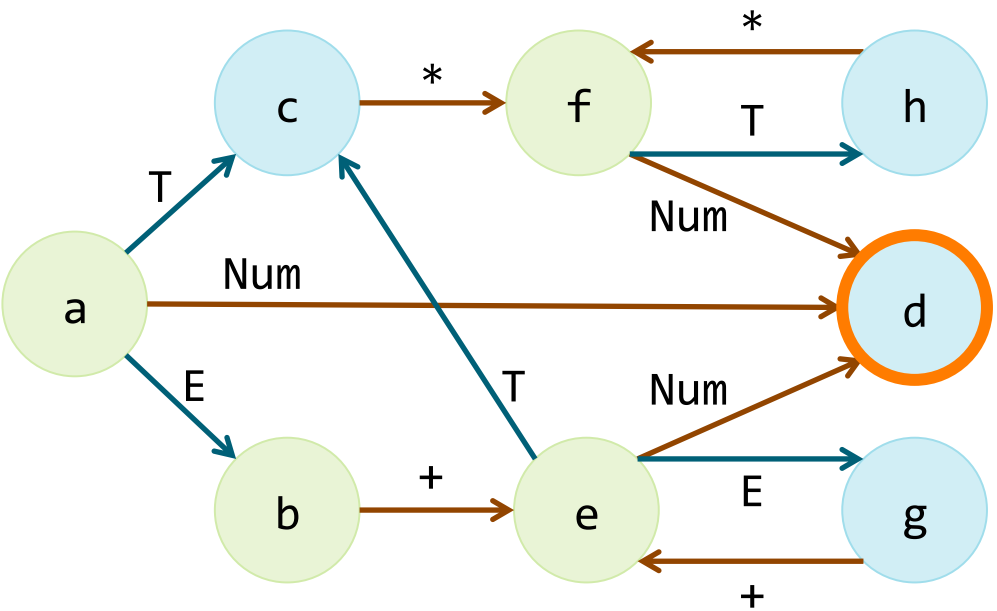


文法

E → E + E
E → T
T → T * T
T → Num

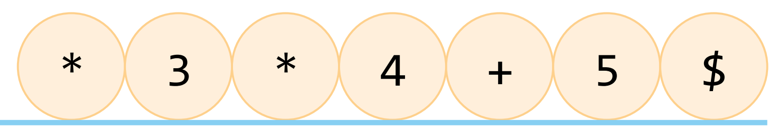




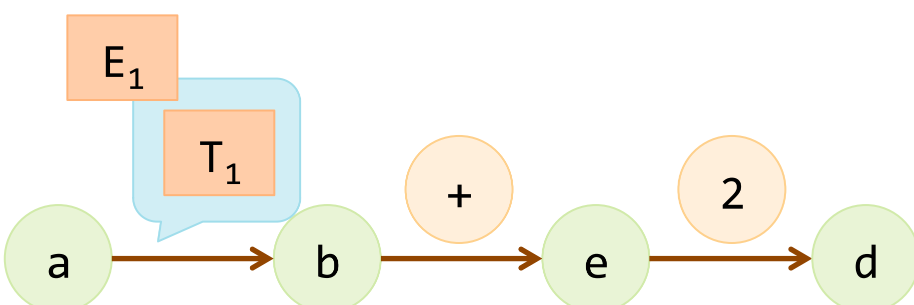


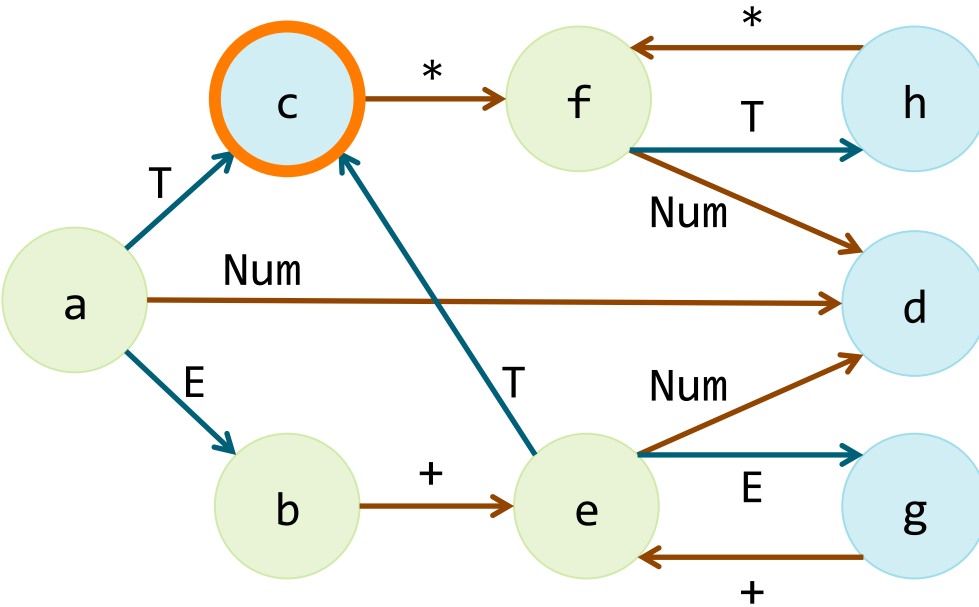
文法

E → E + E
E → T
T → T * T
T → Num



入力





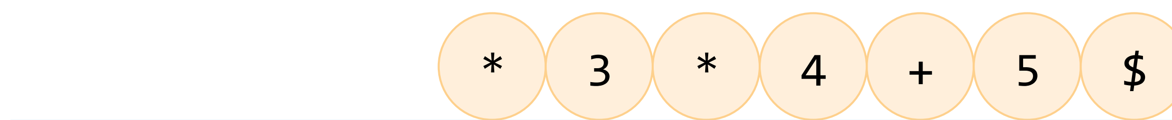
文法

E → E + E

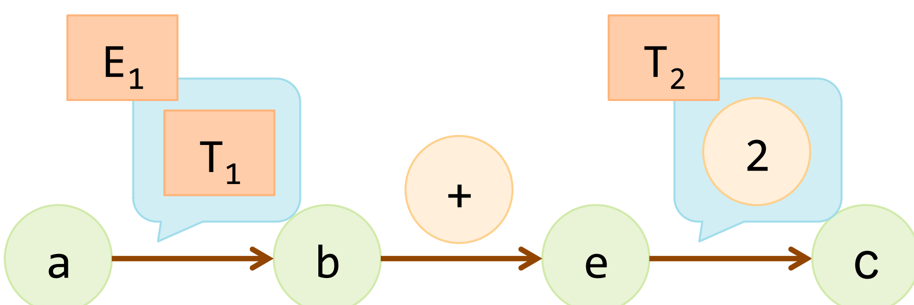
E → T

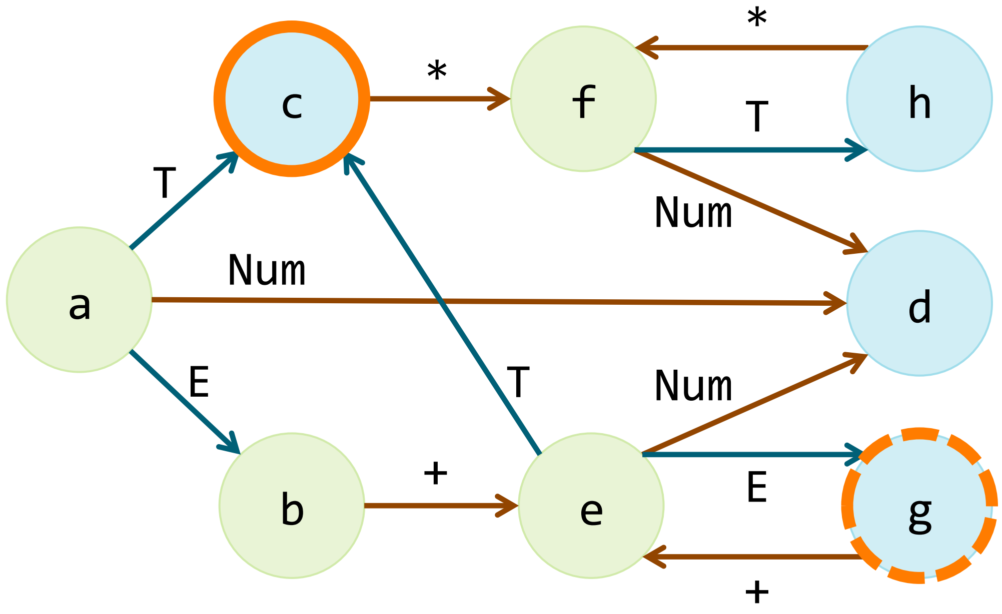
T → T * T

T → Num



入力





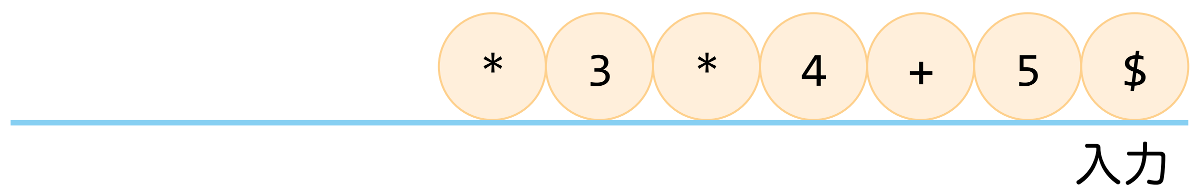
文法

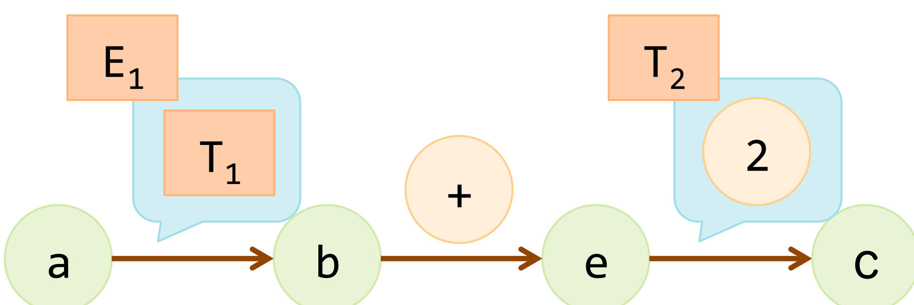
E → E + E

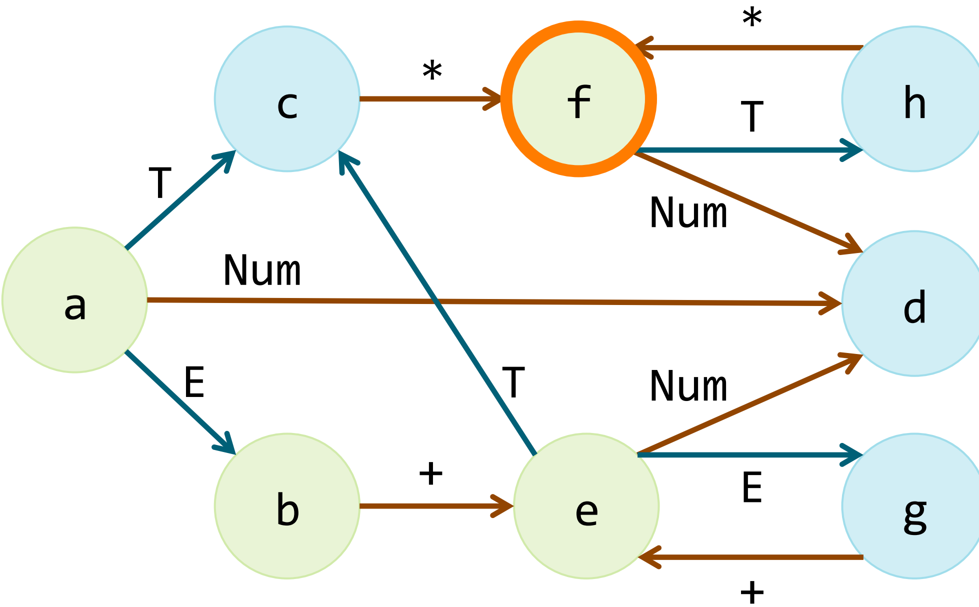
☆ E → T

T → T * T

T → Num







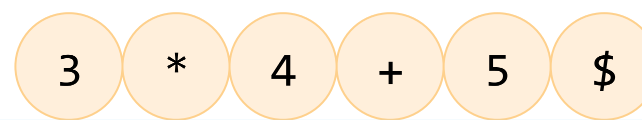
文法

E → E + E

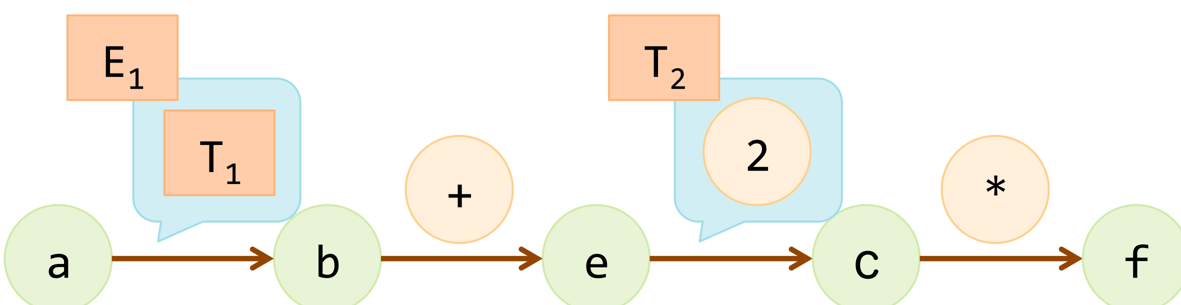
E → T

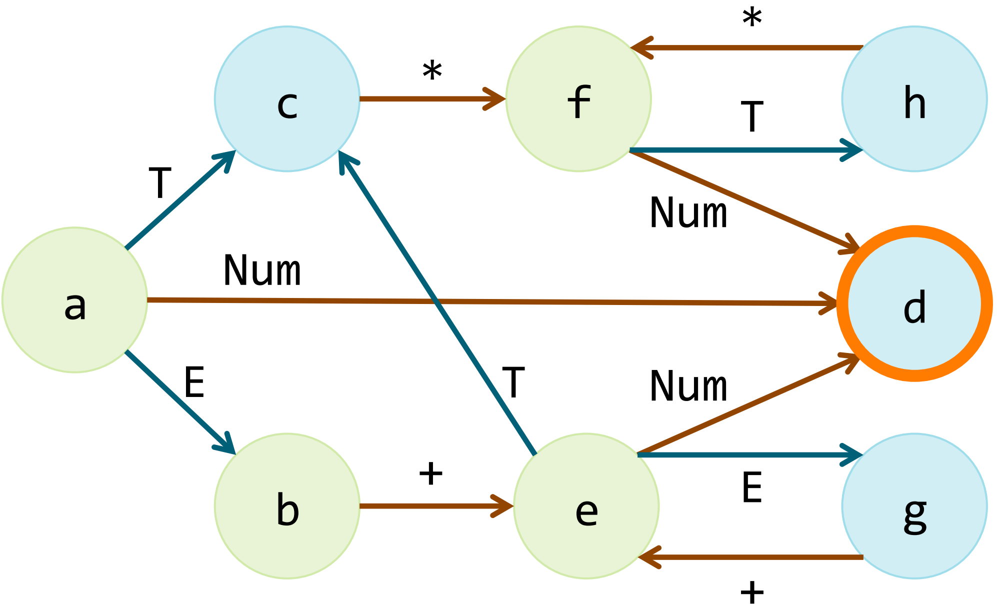
T → T * T

T → Num



入力





文法

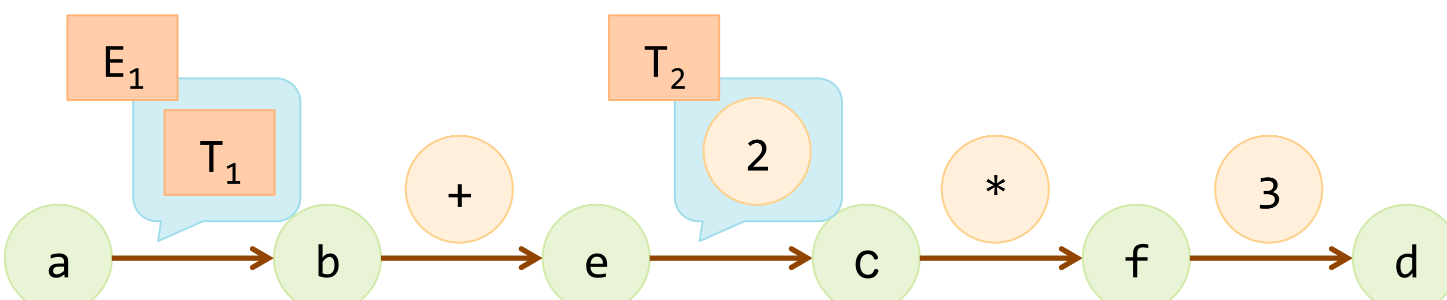
E → E + E

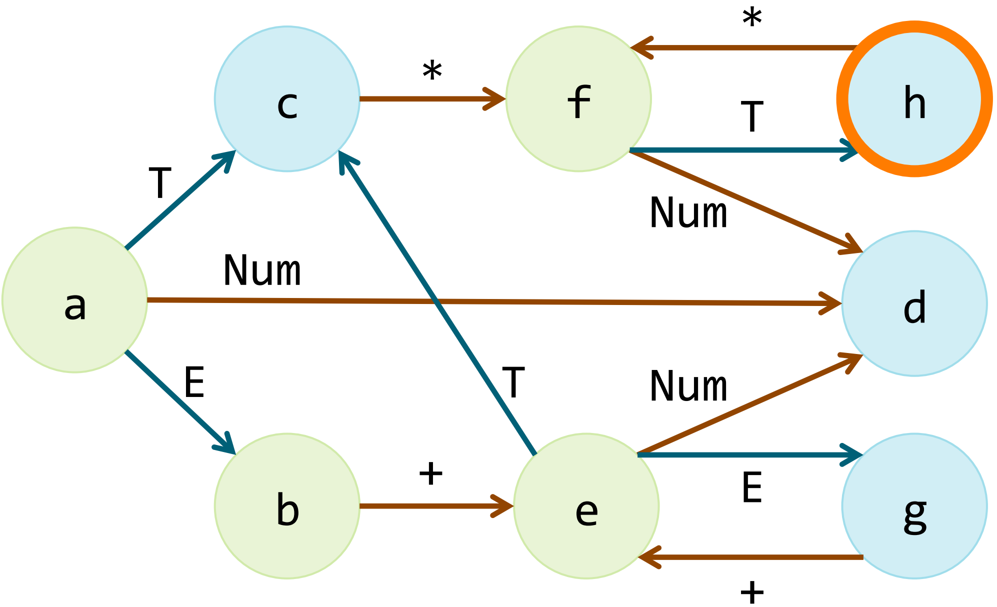
E → T

T → T * T

☆ T → Num

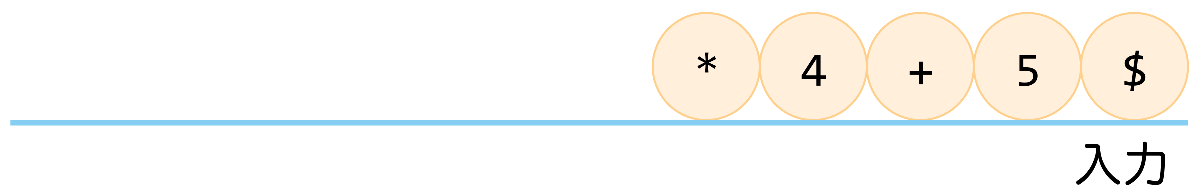
入力

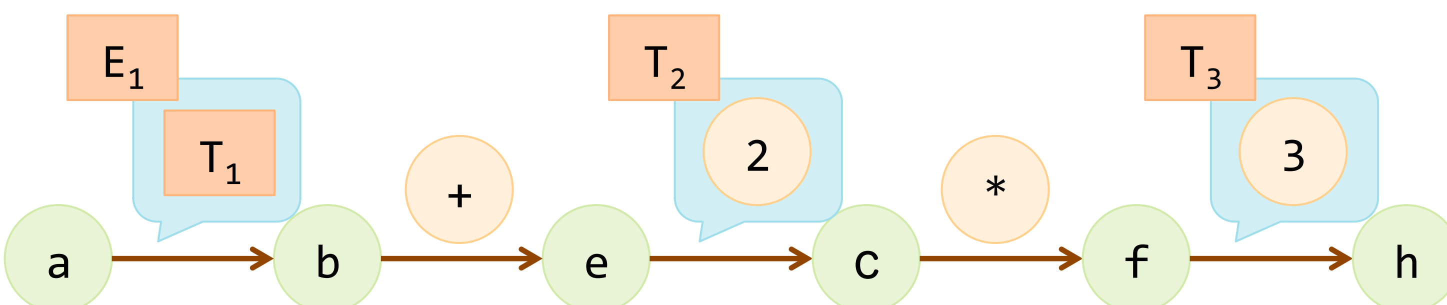


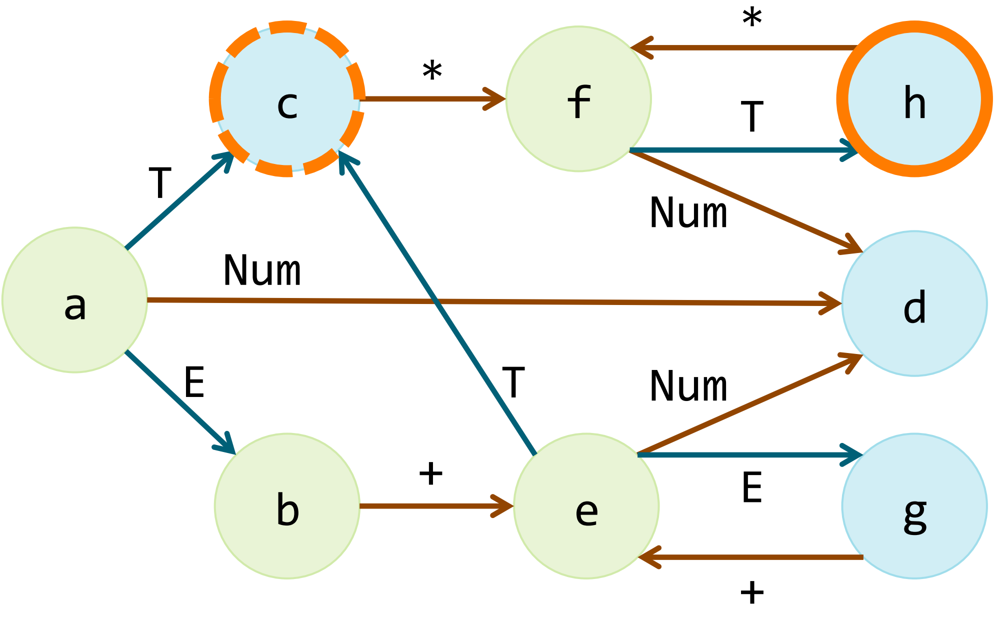


文法

E → E + E
E → T
T → T * T
T → Num







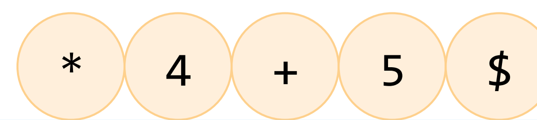
文法

E → E + E

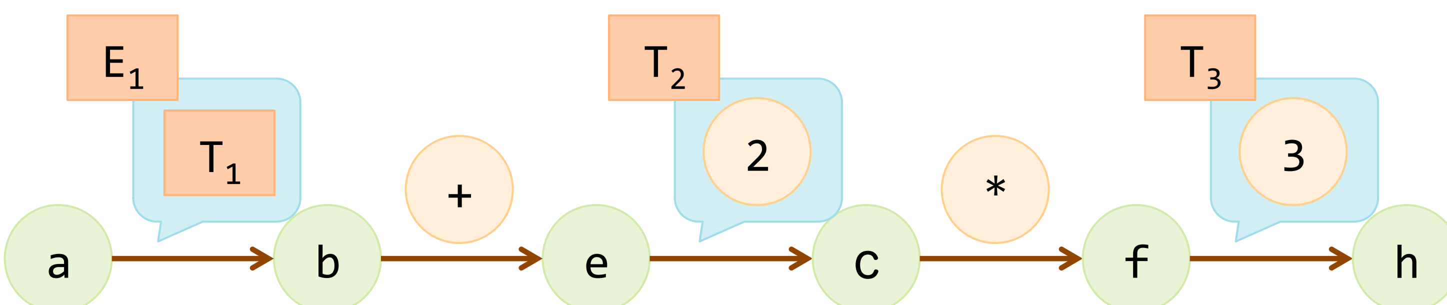
E → T

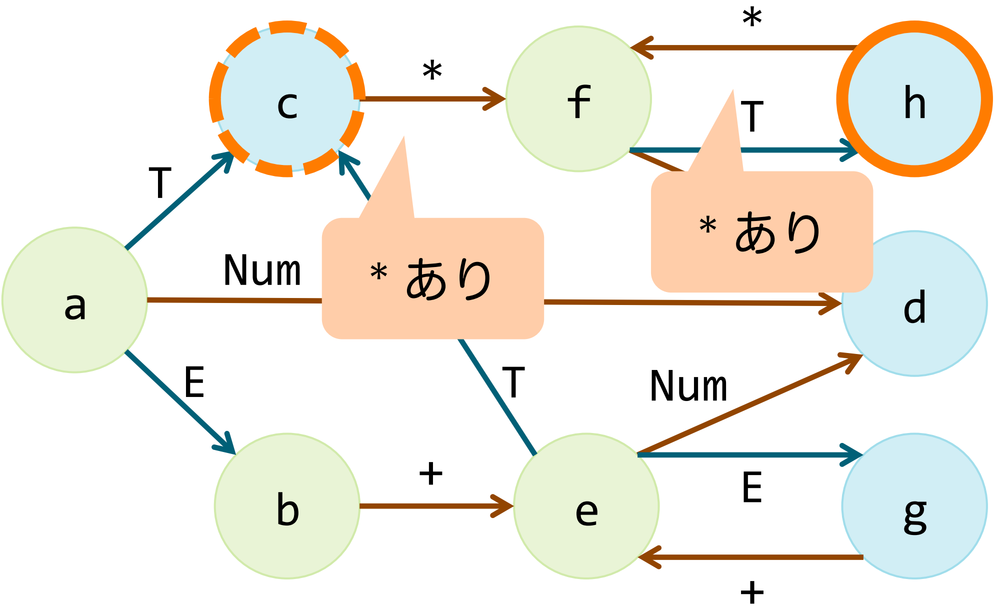
T → T * T

T → Num



入力





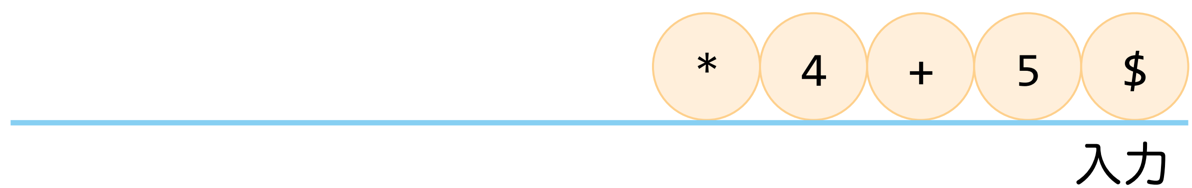
文法

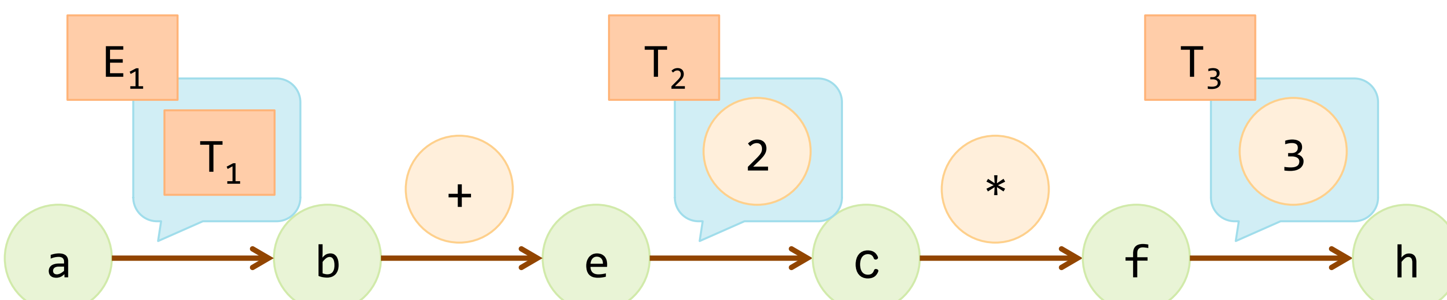
E → E + E

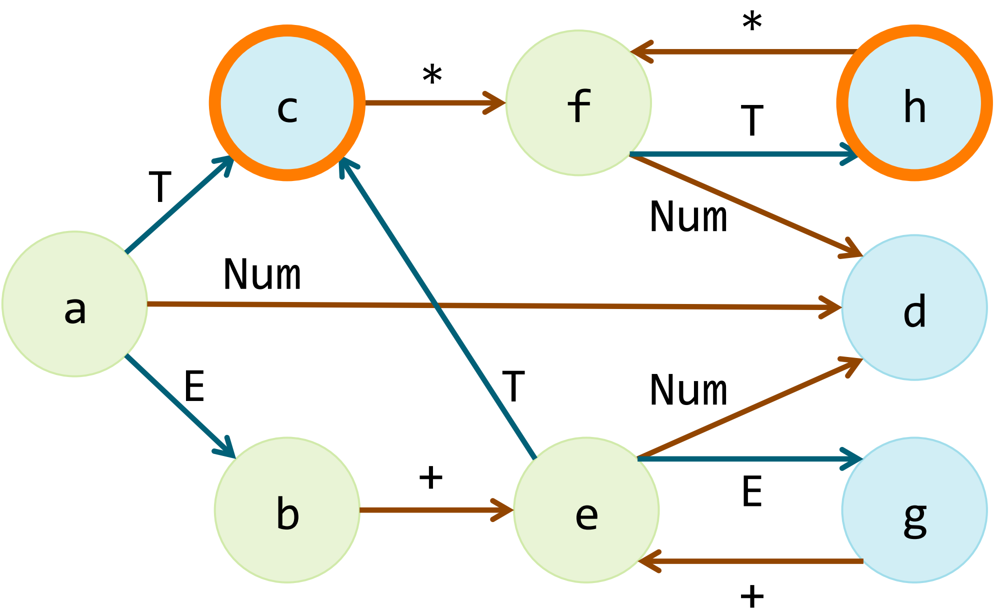
E → T

☆ T → T * T

T → Num







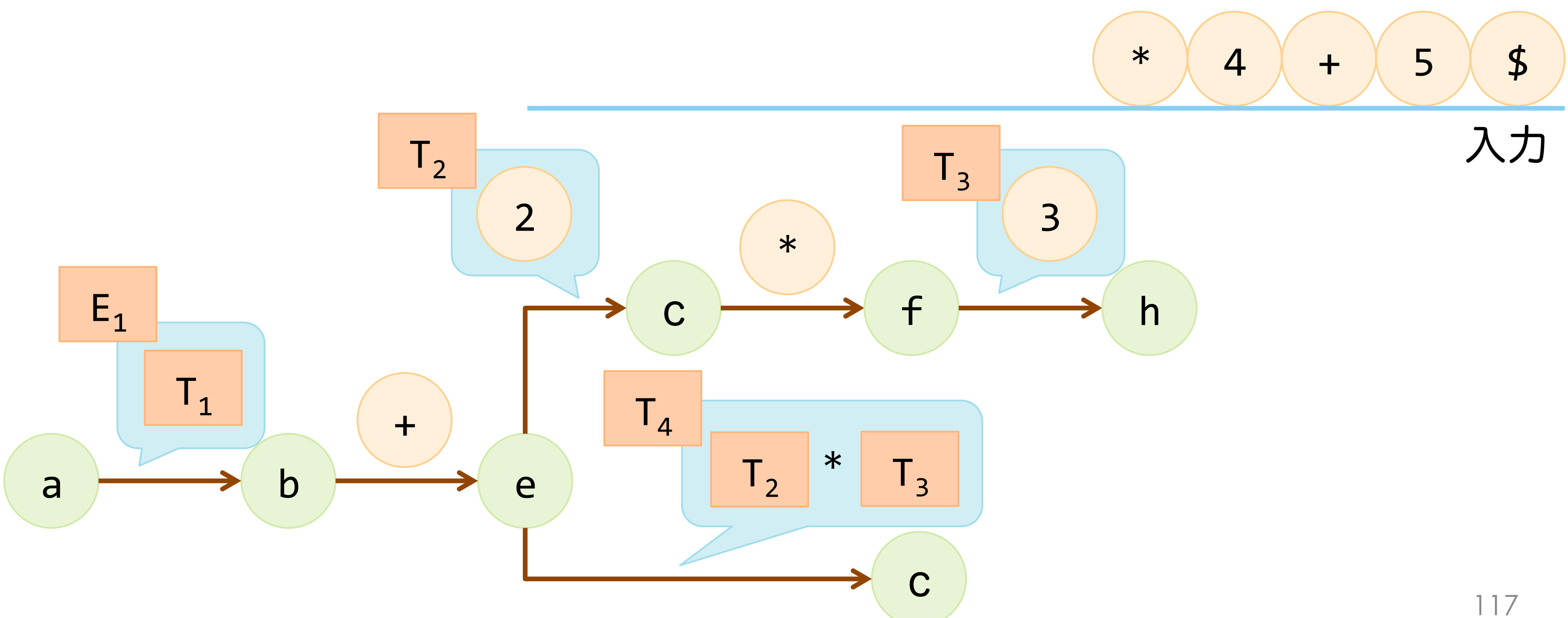
文法

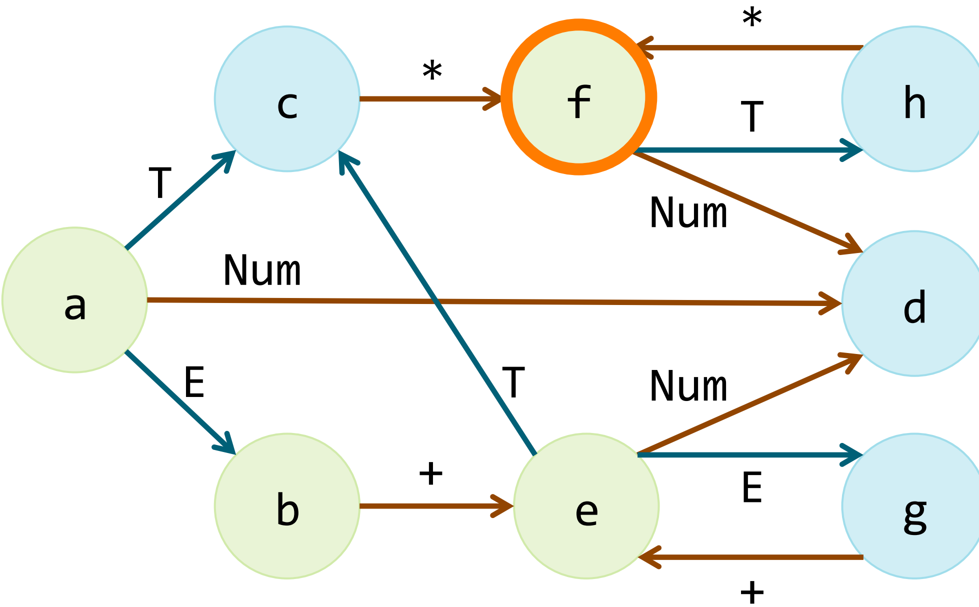
E → E + E

E → T

☆ T → T * T

T → Num





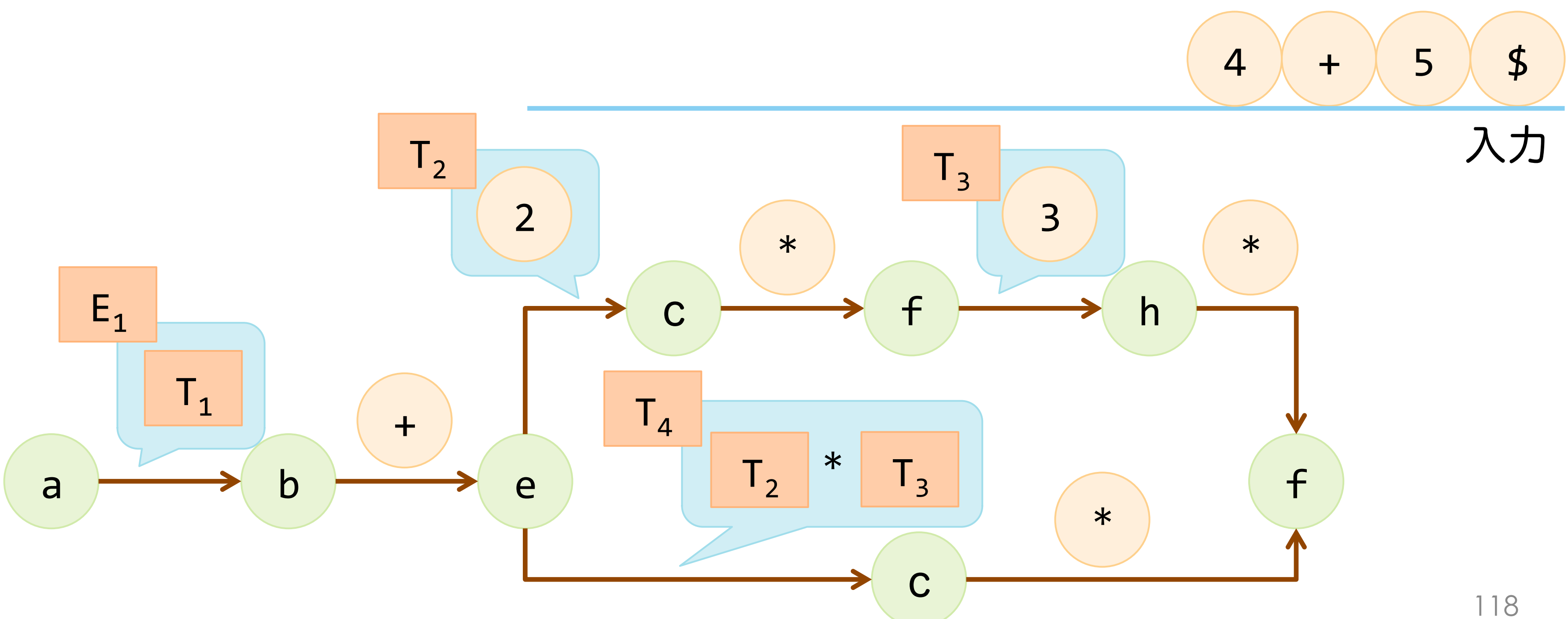
文法

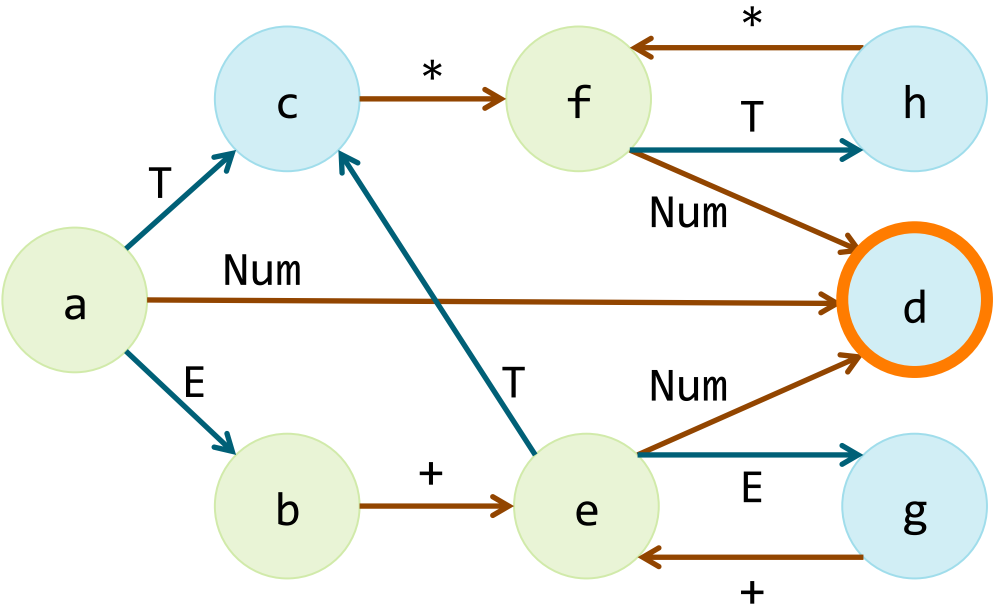
E → E + E

E → T

T → T * T

T → Num





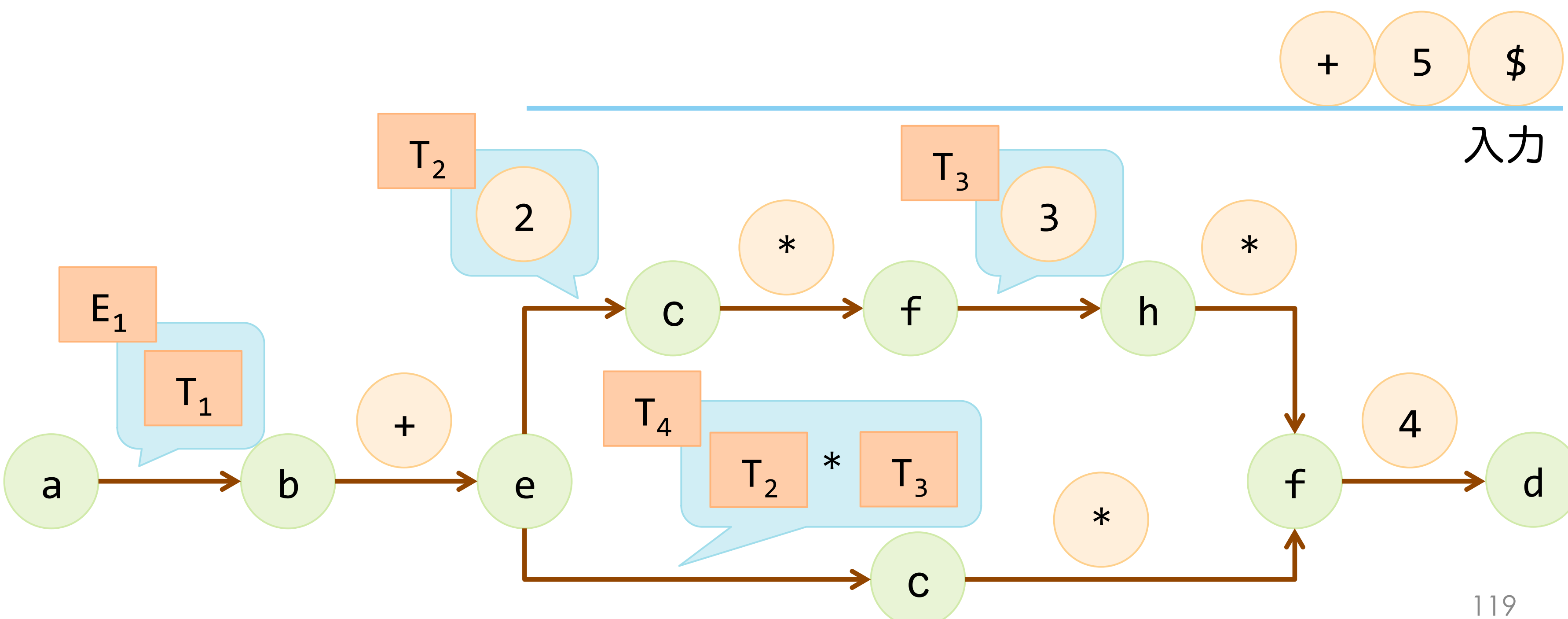
文法

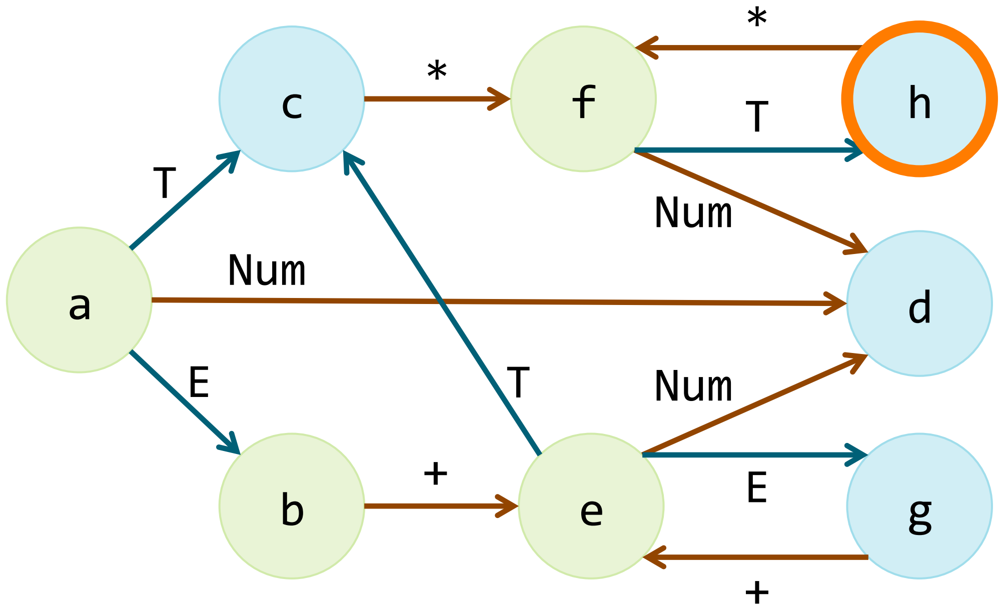
E → E + E

E → T

T → T * T

★ T → Num





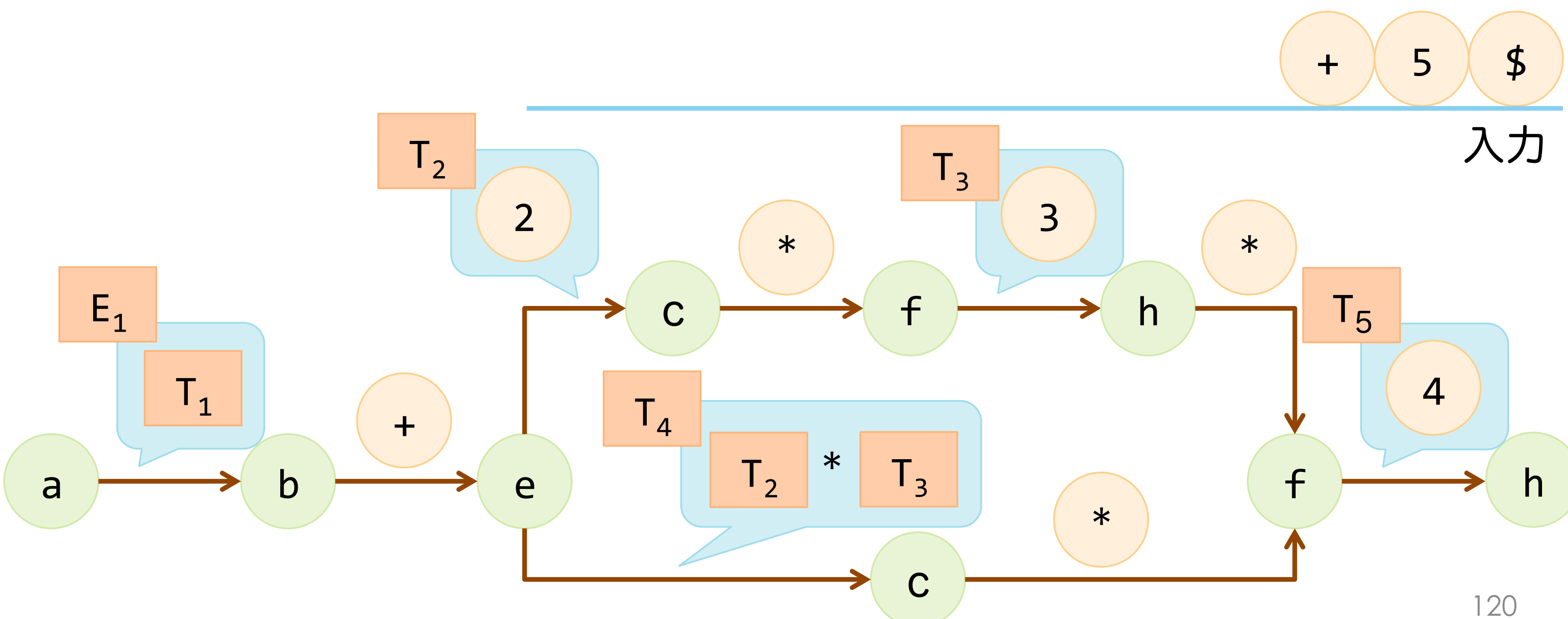
文法

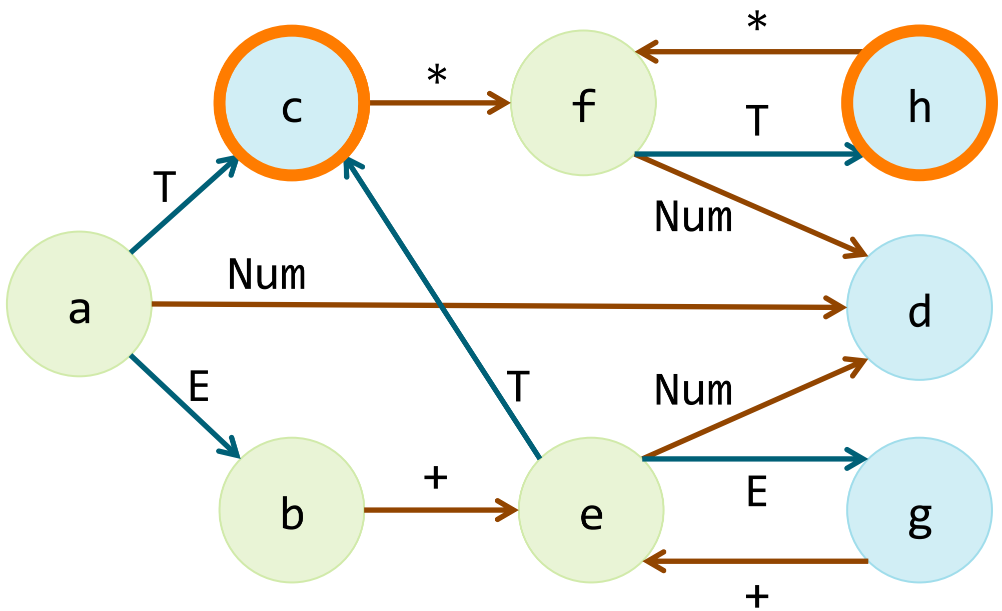
E → E + E

E → T

☆ T → T * T

T → Num





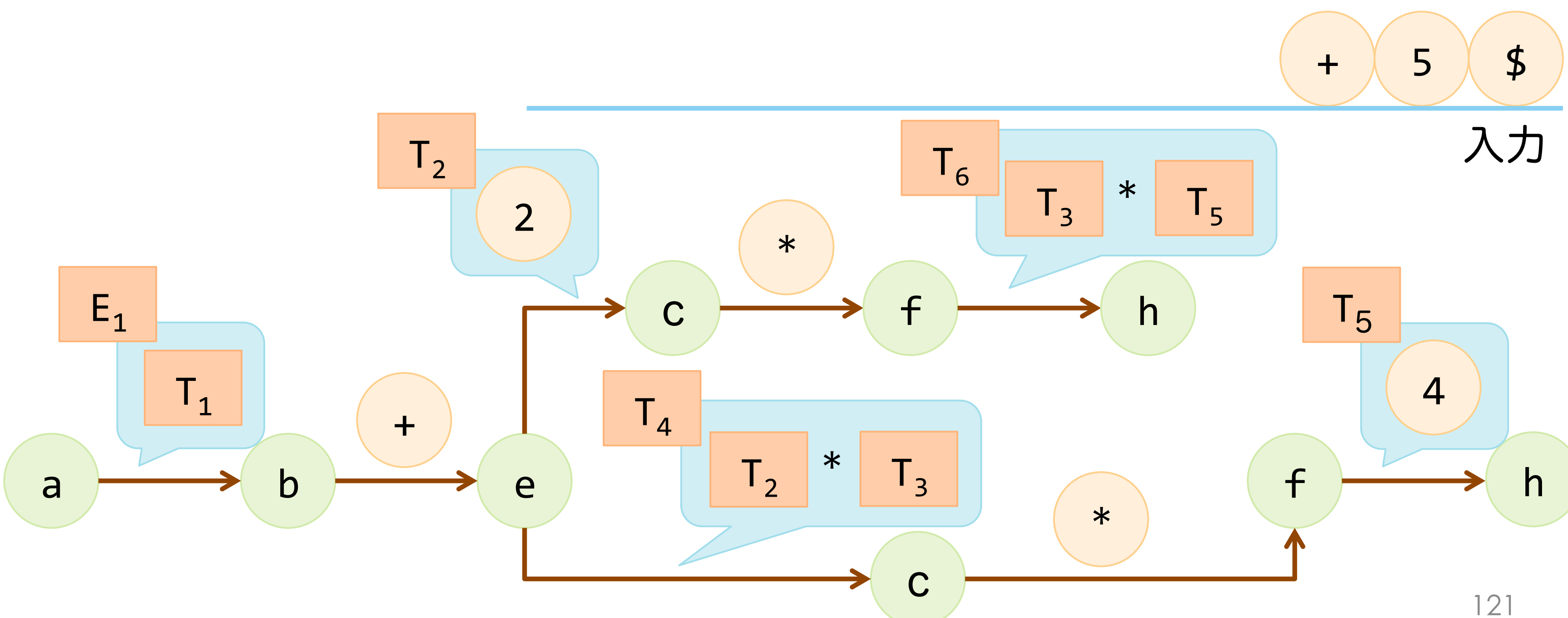
文法

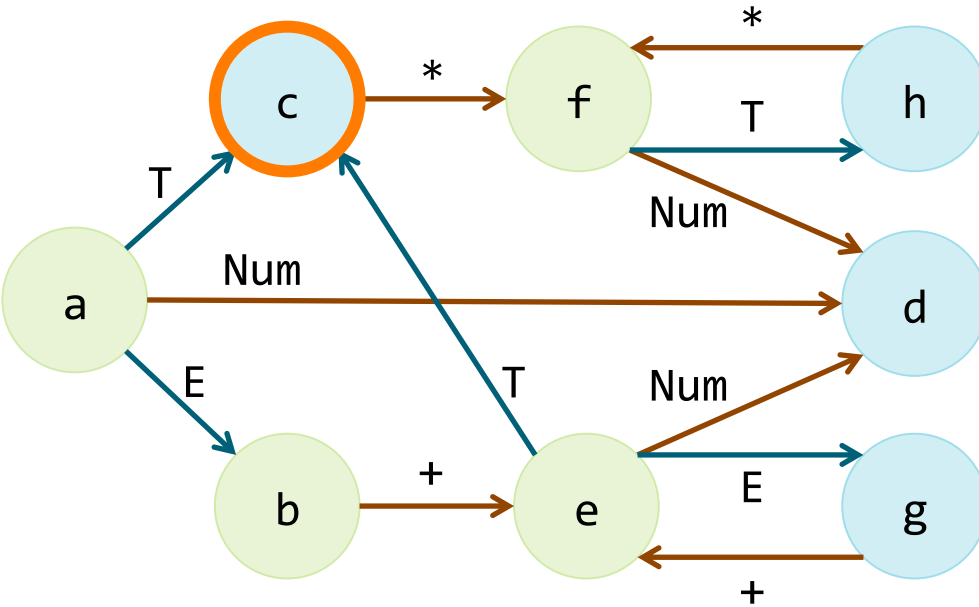
E → E + E

E → T

T → T * T

T → Num





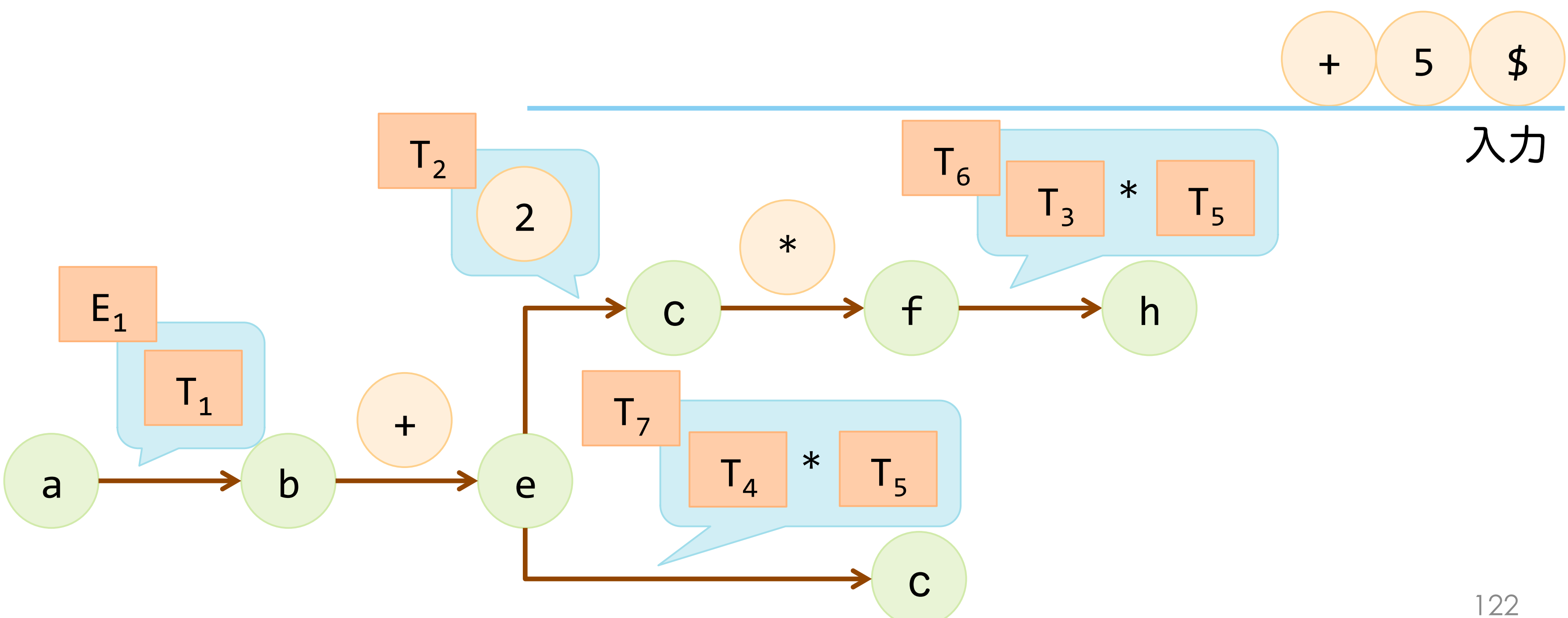
文法

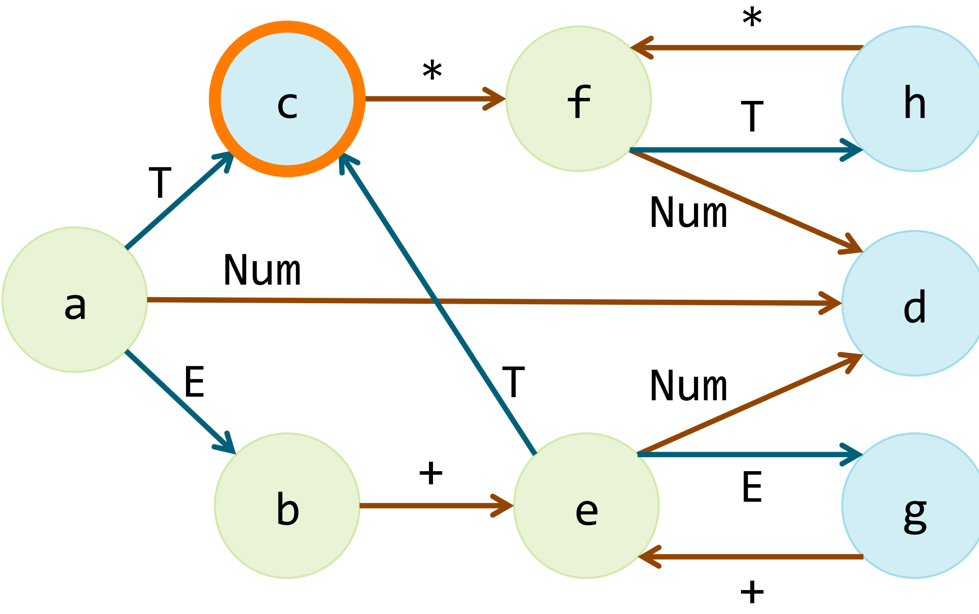
E → E + E

E → T

T → T * T

T → Num





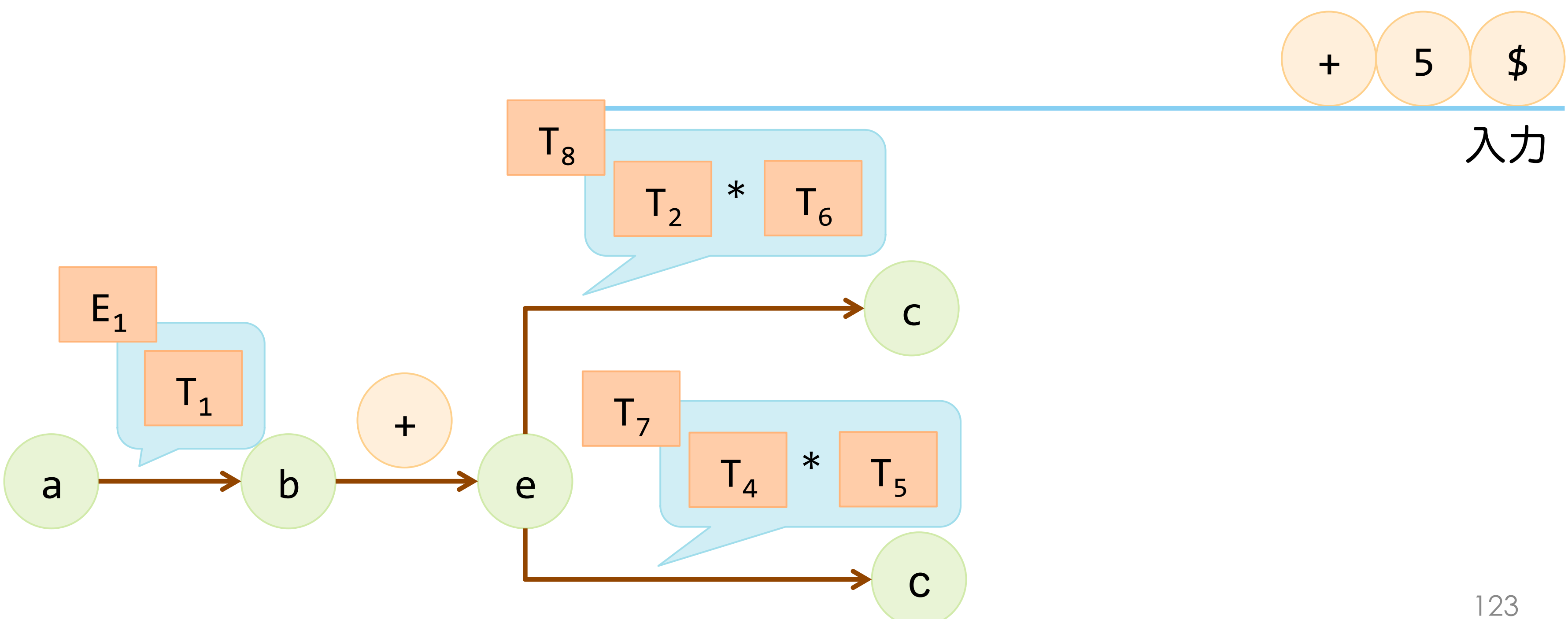
文法

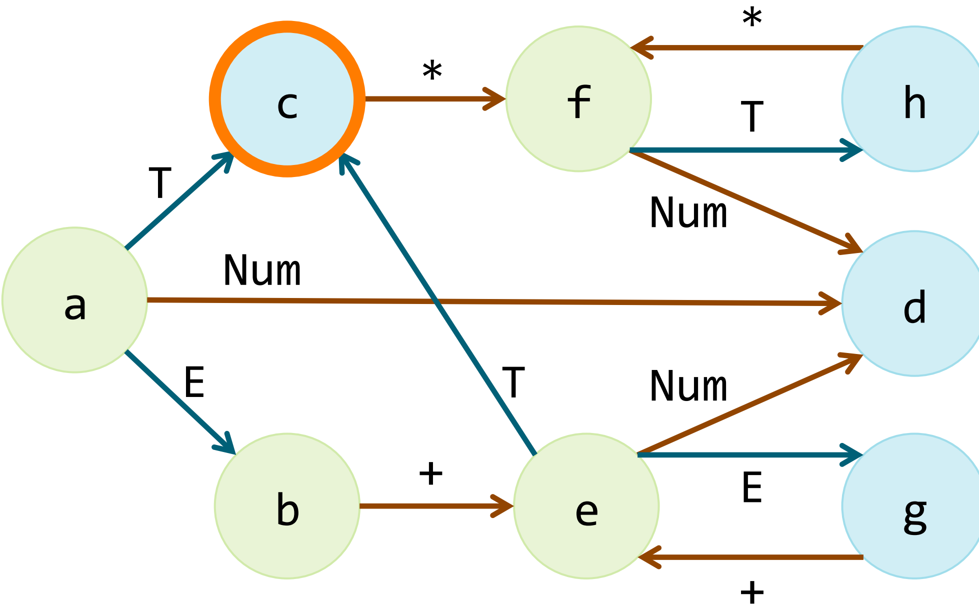
E → E + E

E → T

☆ T → T * T

T → Num





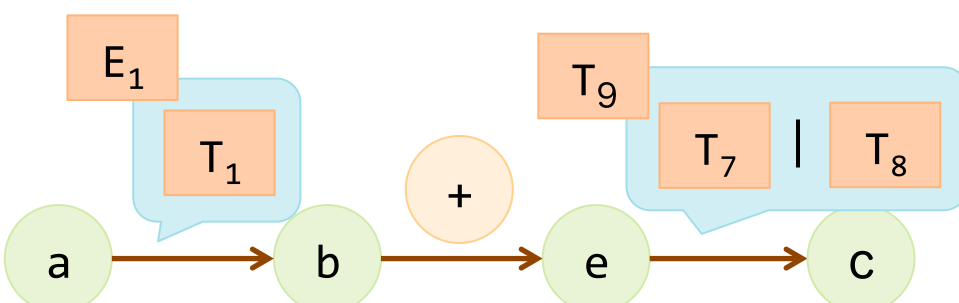
文法

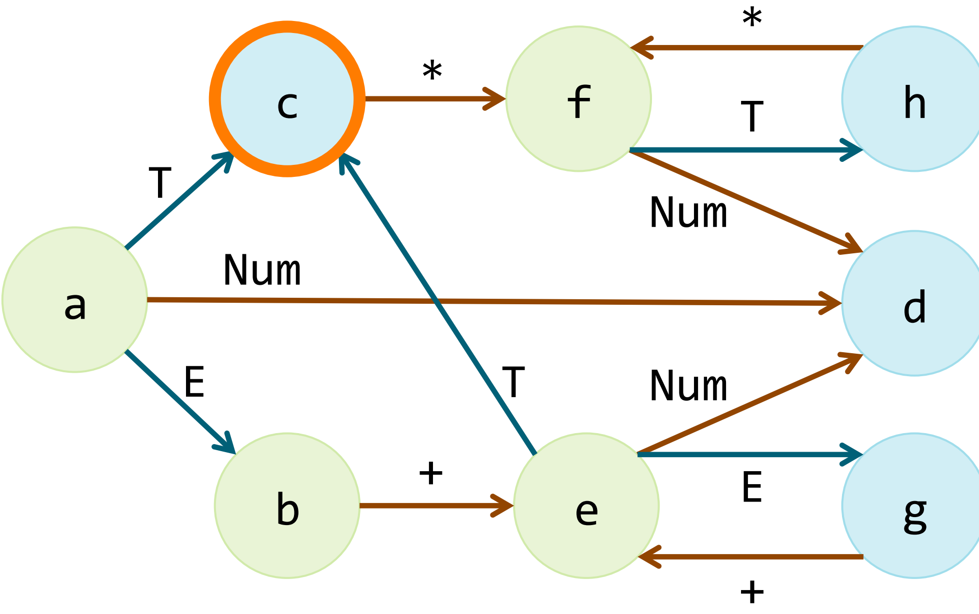
$E \rightarrow E + E$

$E \rightarrow T$

$T \rightarrow T * T$

$T \rightarrow Num$





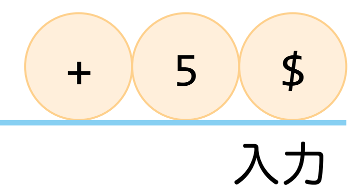
文法

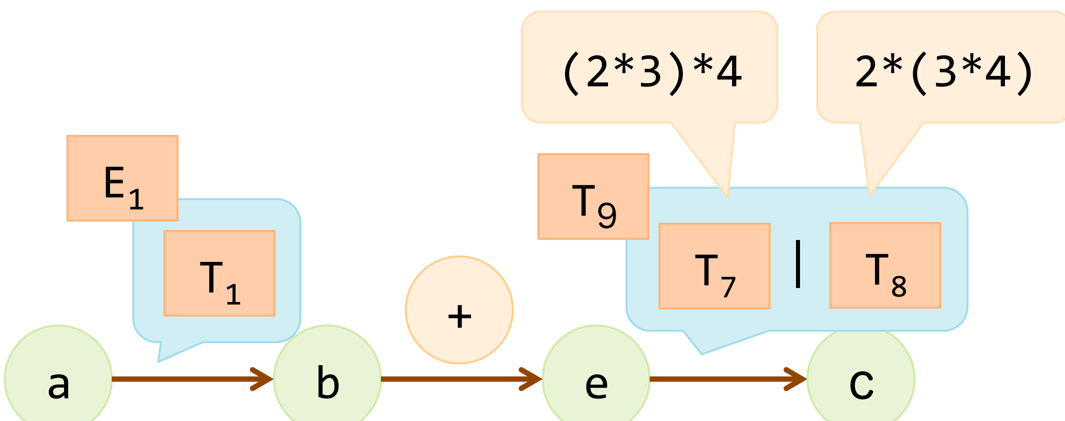
$$E \rightarrow E + E$$

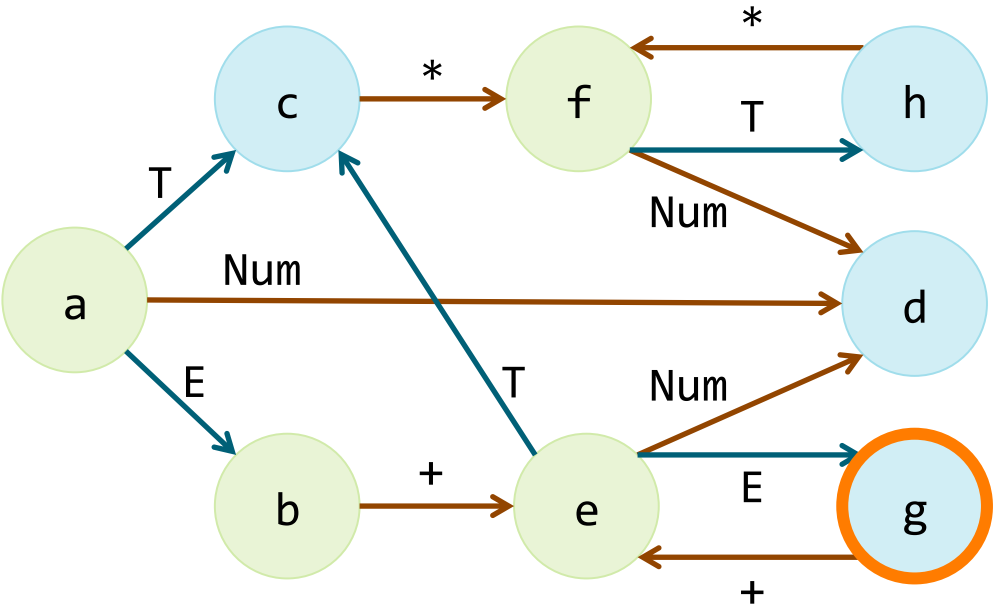
★ $$E \rightarrow T$$

$$T \rightarrow T * T$$

$$T \rightarrow Num$$

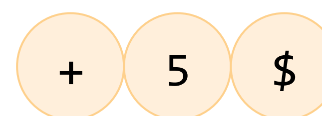




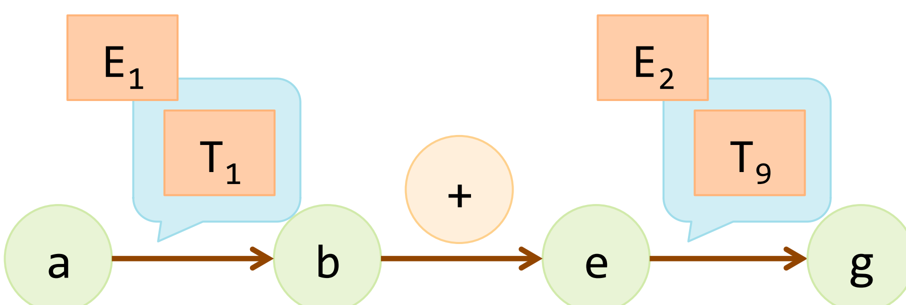


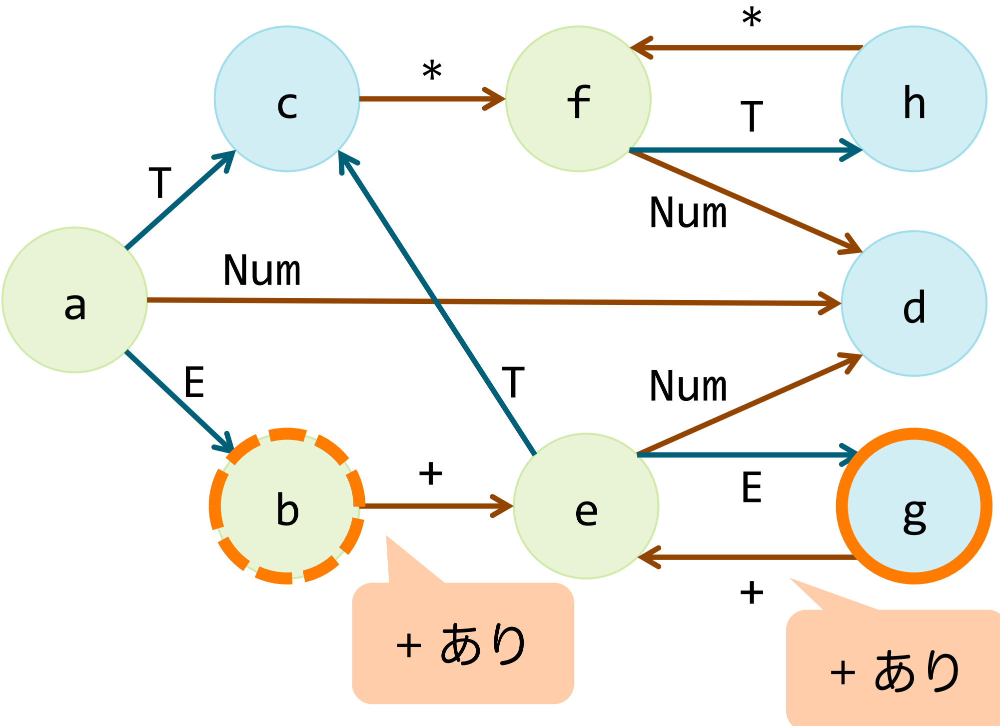
文法

- ☆ E → E + E
- E → T
- T → T * T
- T → Num



入力





文法

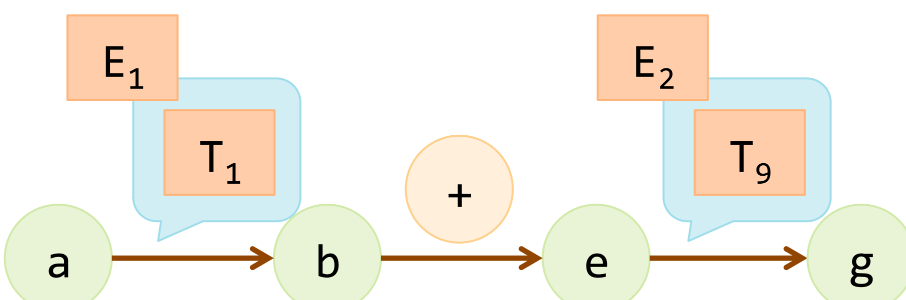
☆ E → E + E

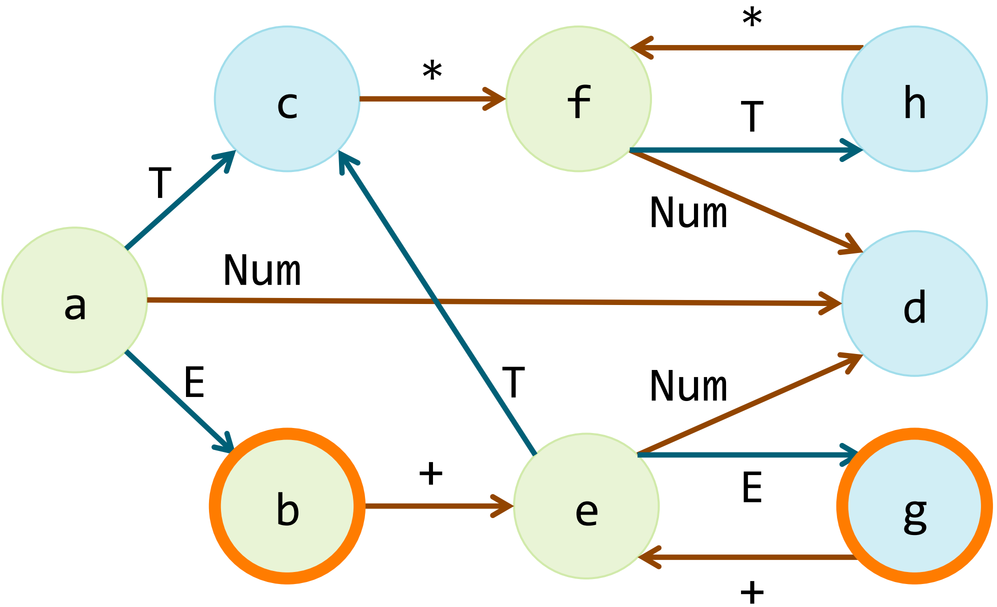
E → T

T → T * T

T → Num

入力





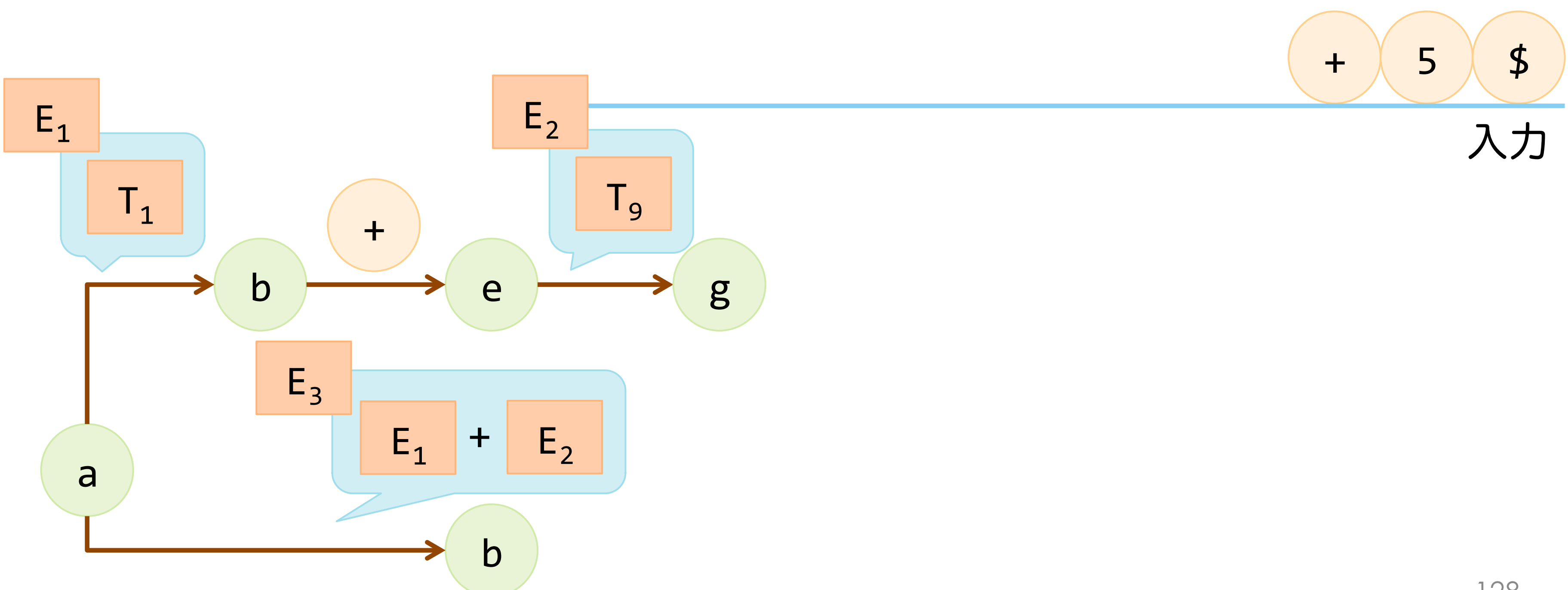
文法

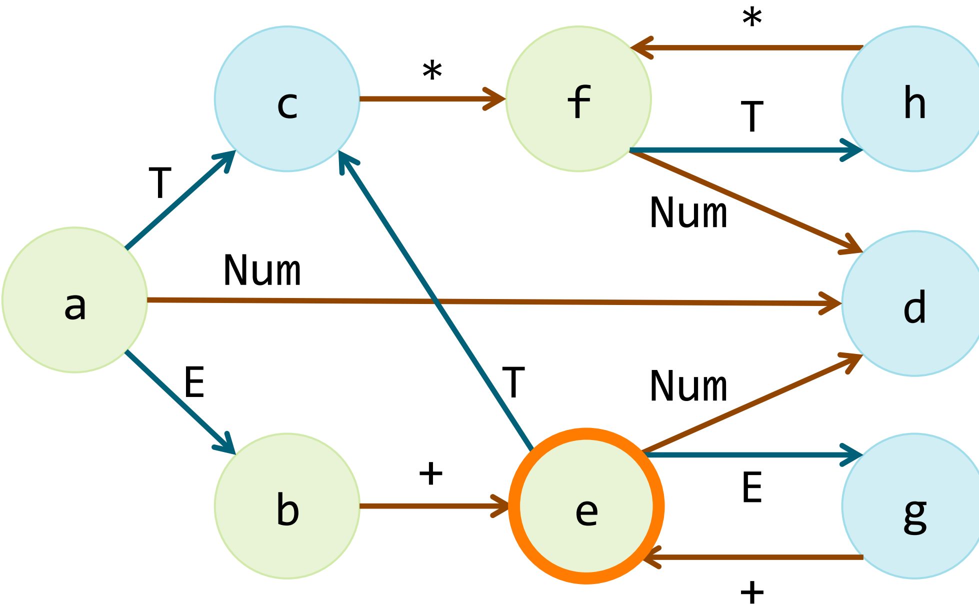
☆ E → E + E

E → T

T → T * T

T → Num





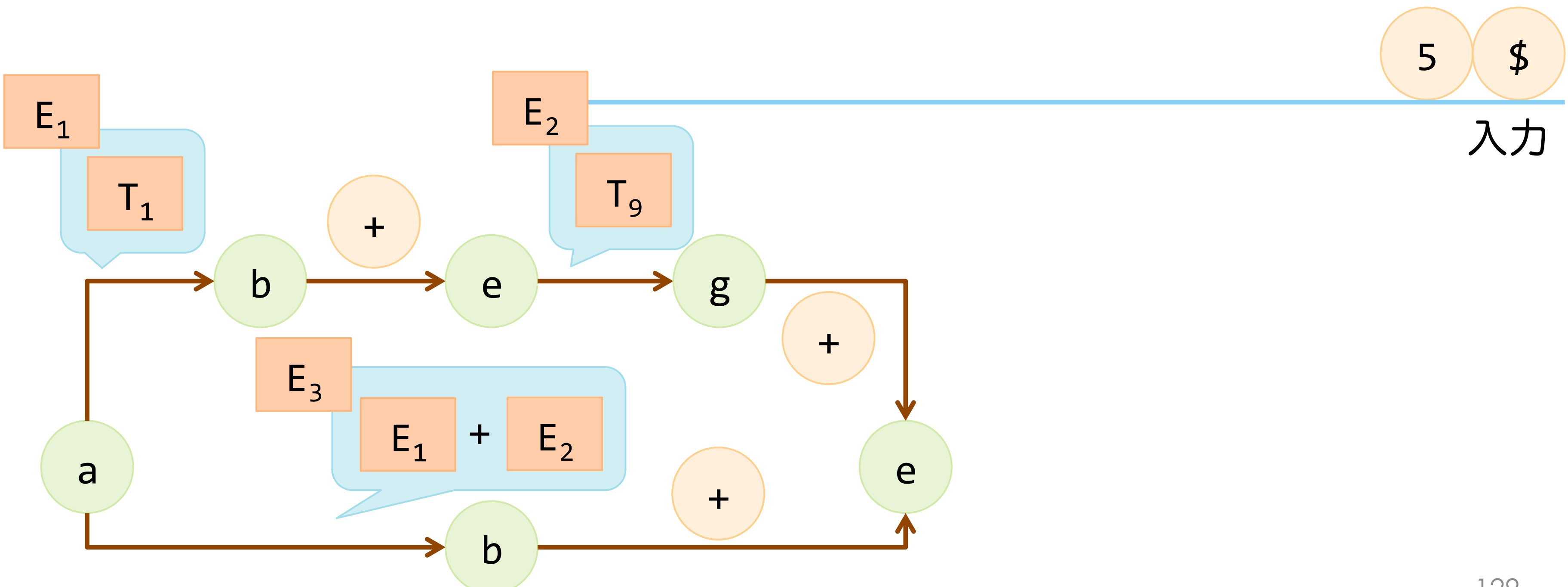
文法

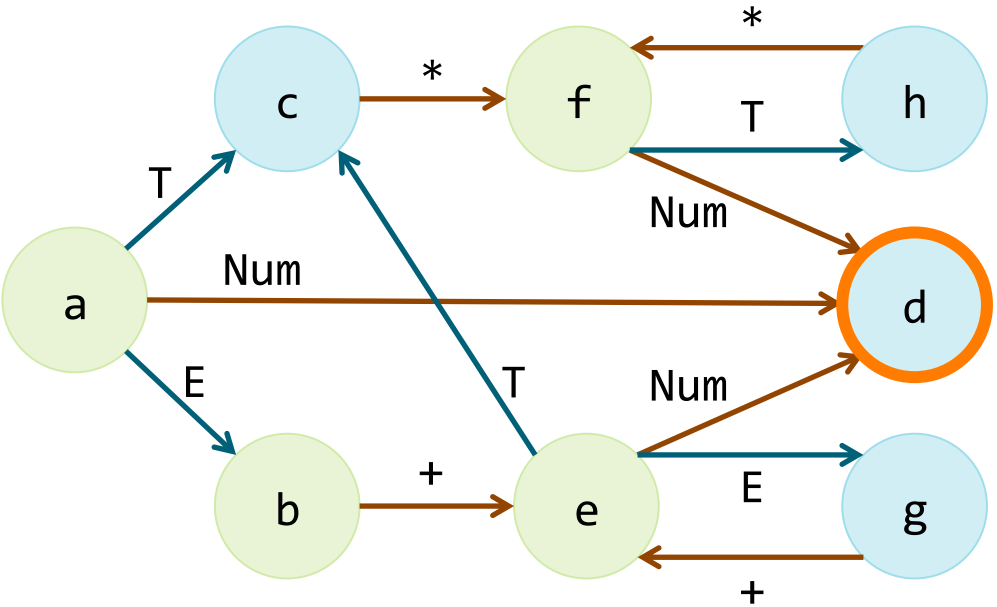
$E \rightarrow E + E$

$E \rightarrow T$

$T \rightarrow T * T$

$T \rightarrow \mathrm{Num}$





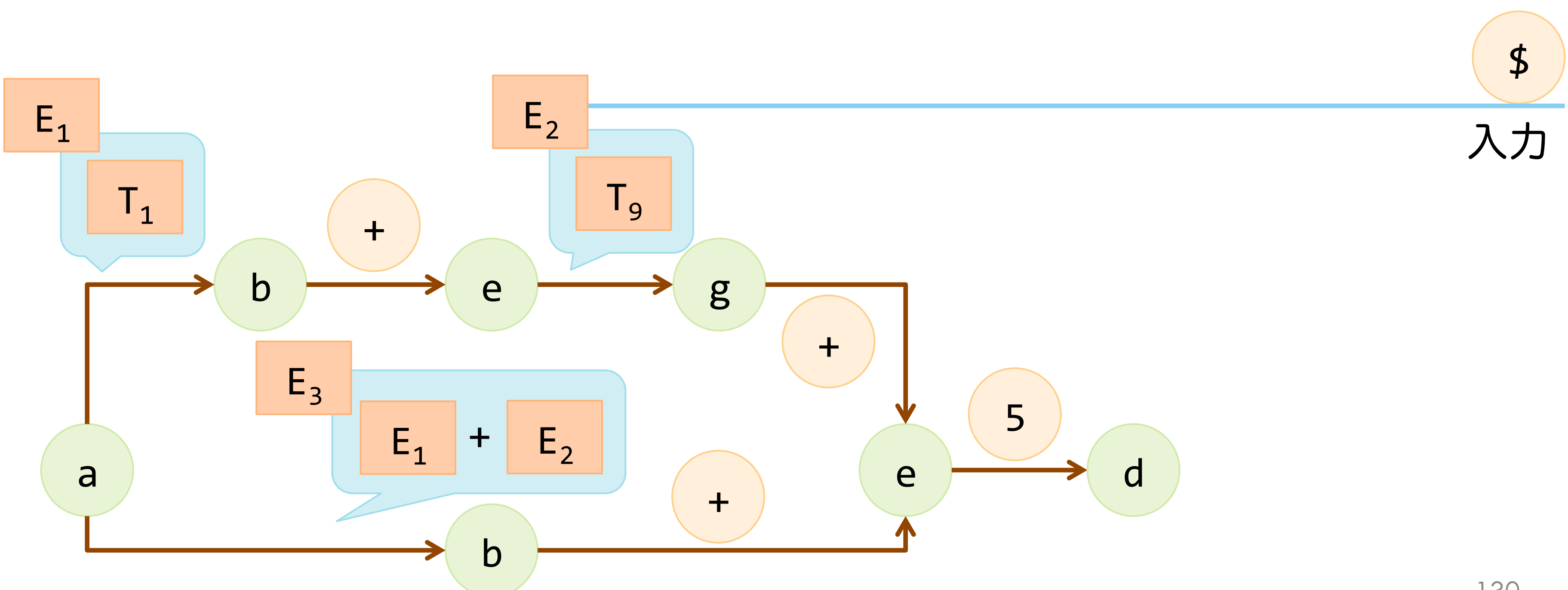
文法

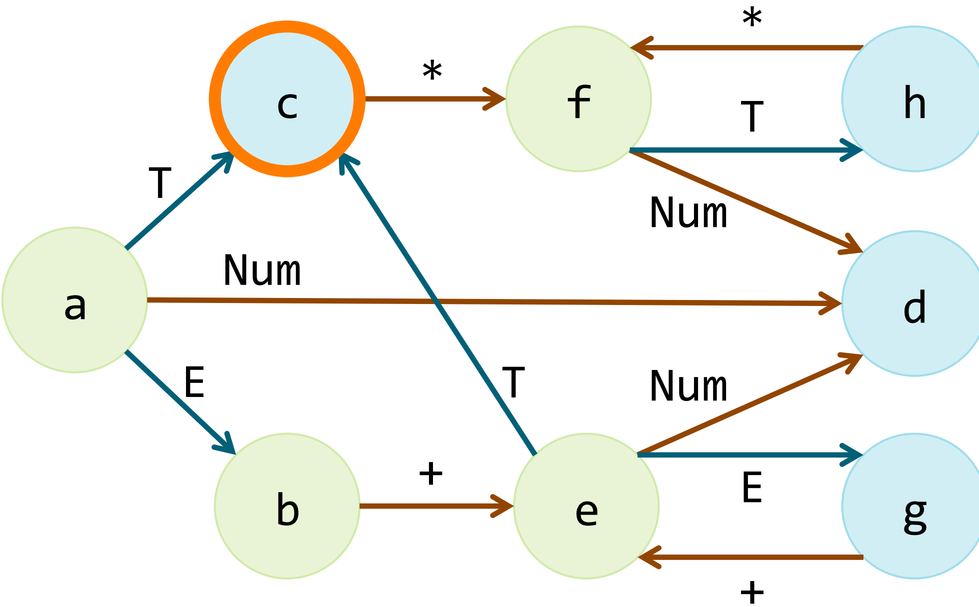
E → E + E

E → T

T → T * T

☆ T → Num





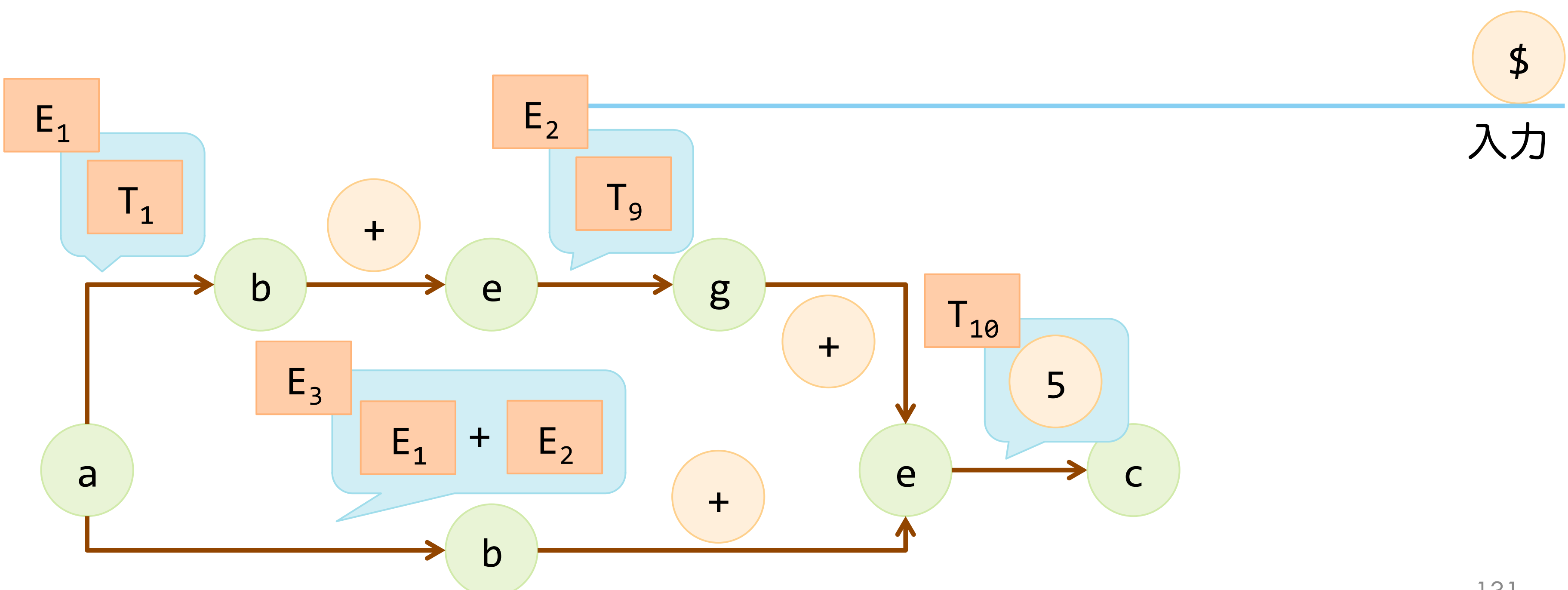
文法

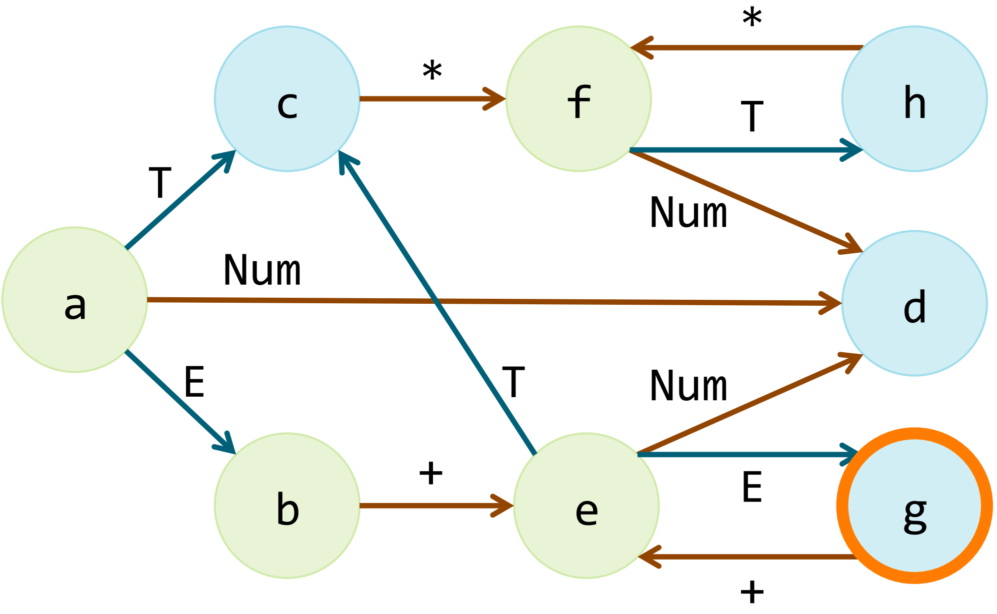
$E \rightarrow E + E$

$E \rightarrow T$

$T \rightarrow T * T$

$T \rightarrow Num$





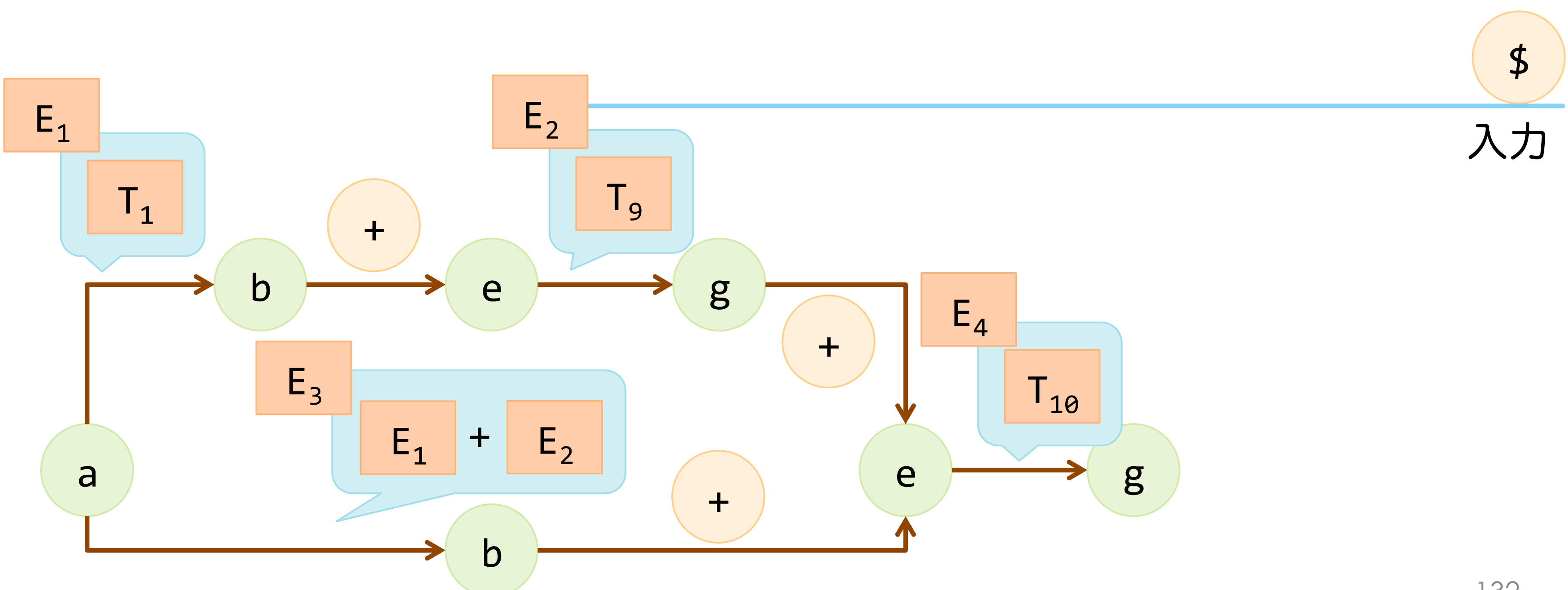
文法

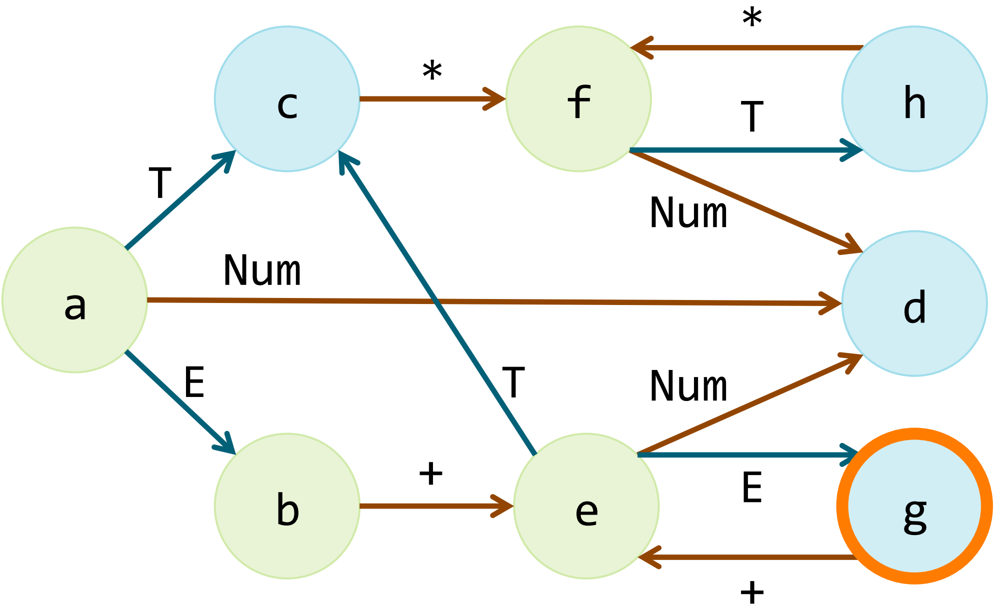
$E \rightarrow E + E$

$E \rightarrow T$

$T \rightarrow T * T$

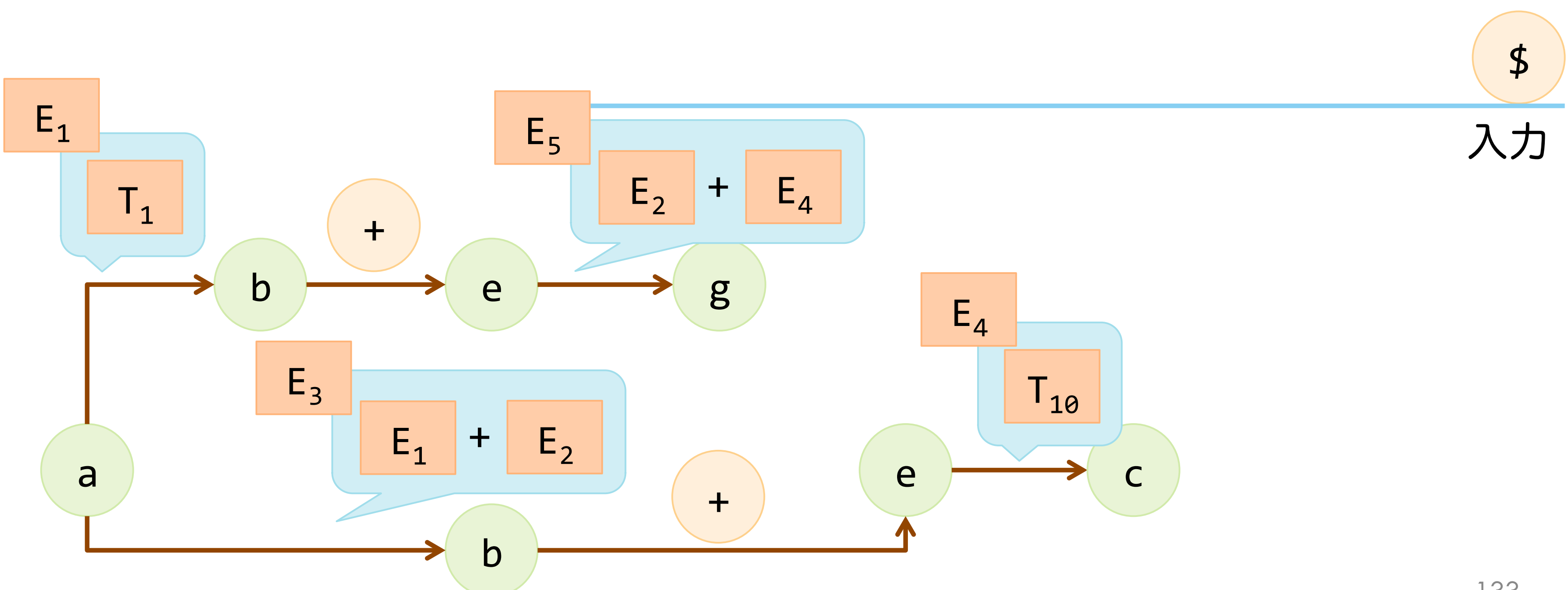
$T \rightarrow Num$

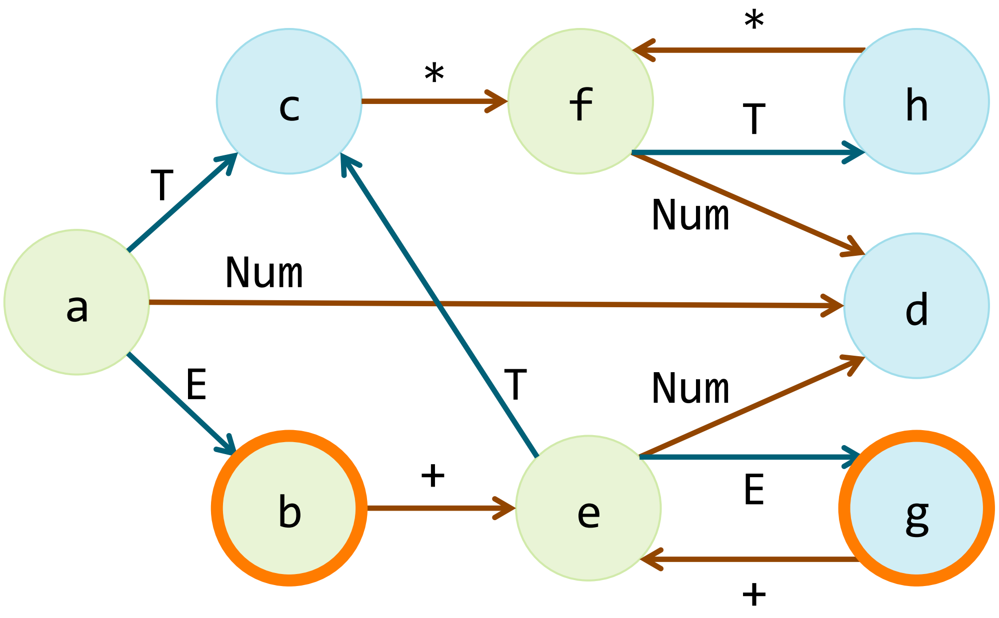




文法

E → E + E
E → T
T → T * T
T → Num





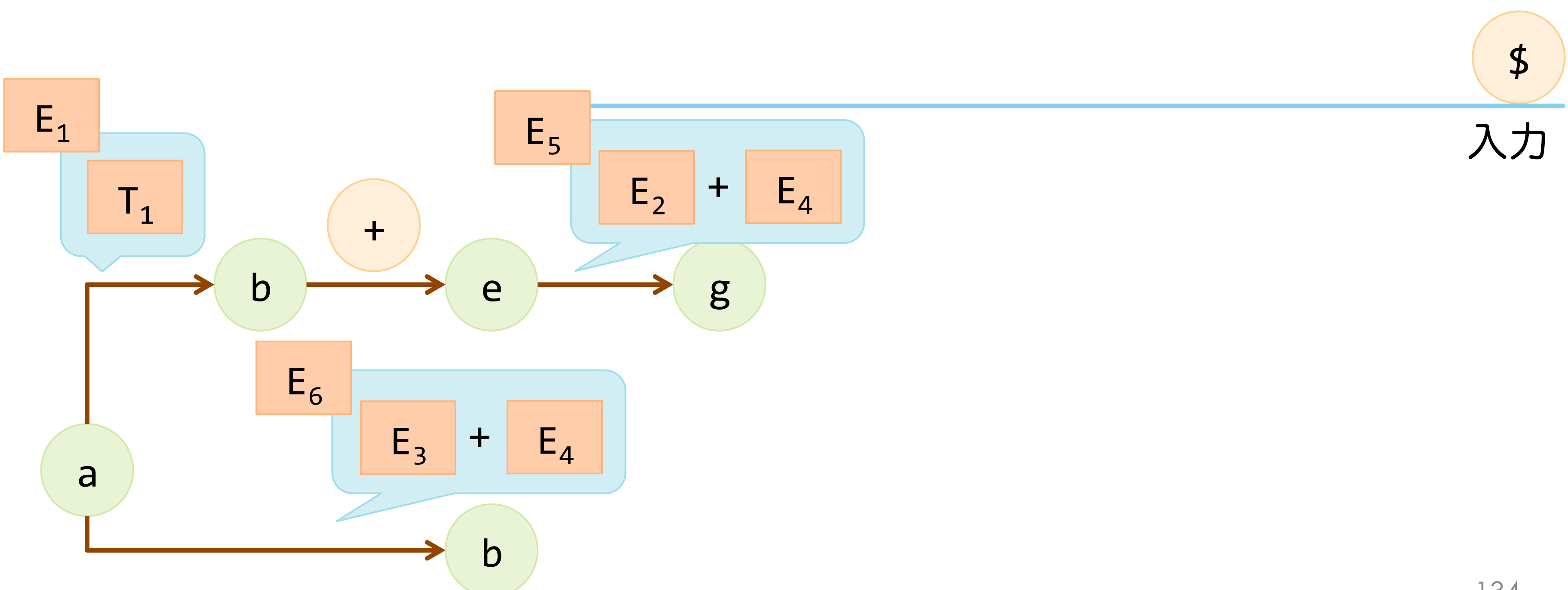
文法

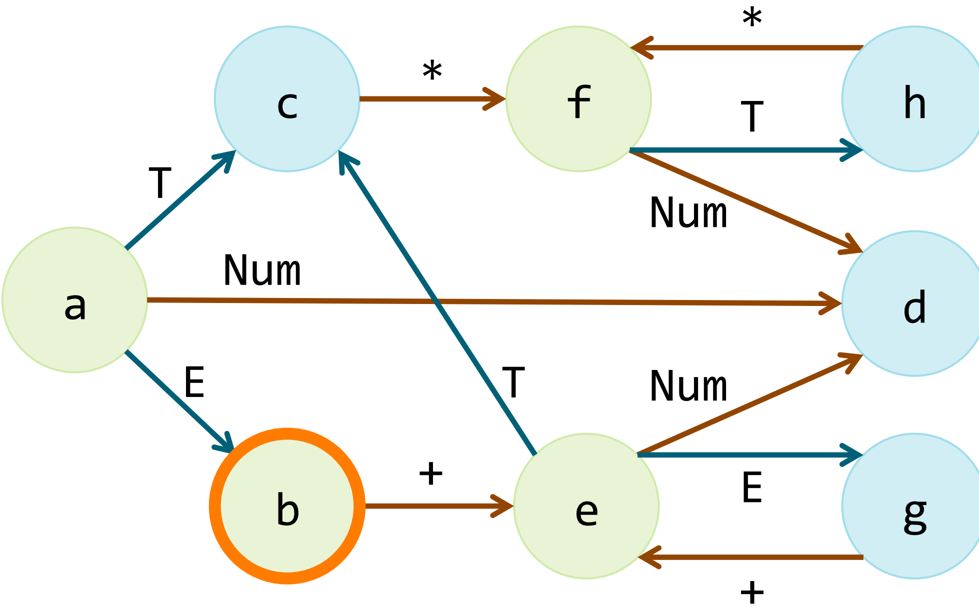
★ E → E + E

E → T

T → T * T

T → Num





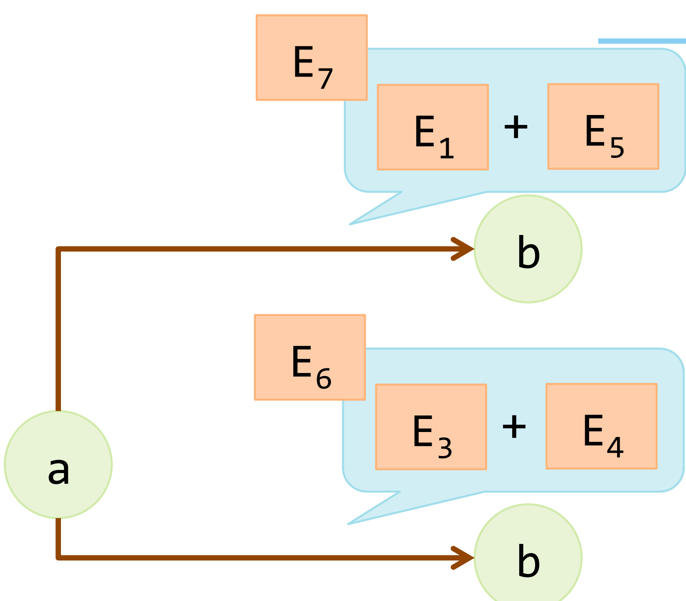
文法

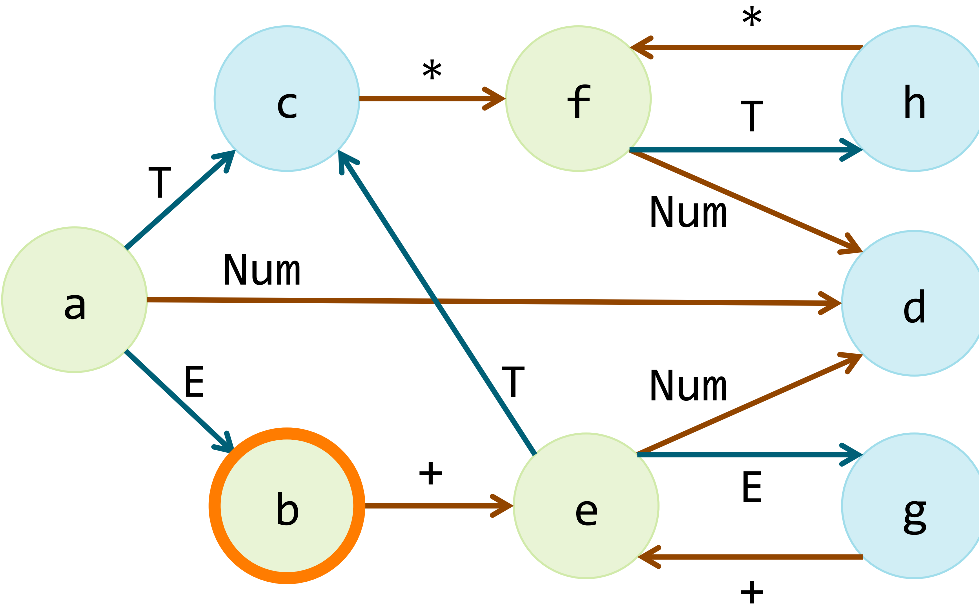
☆ E → E + E

E → T

T → T * T

T → Num

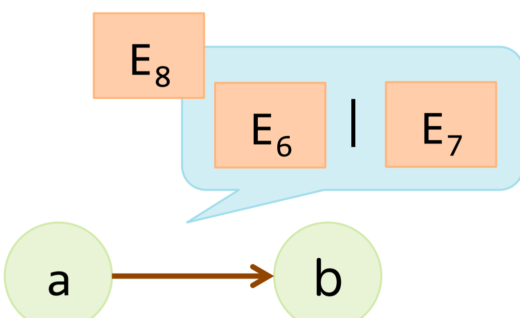




文法

E → E + E
E → T
T → T * T
T → Num

入力



GLR parsing [Tomita 1985]

- 構文解析表の作り方は LR parsing と同様
  - ただし同一箇所に複数規則を許す
- ほぼ決定的な文法に対して O(n) で動作
- 能力
  - 計算量：$O(n^{p+1})$（p=文法の最長の右辺の長さ)
  - 言語：Context-free language
  - 文法：Context-free grammar

Earley parsing [Earley 1968]

アイデア

- 動的計画法
- 入力位置を進めながら構文規則上の位置を記録する
  - ➢ 構文木を記憶すると指数爆発するため親エントリを記憶

1 + 2 + 3

```
E → Num ·     {1}
E → E · + E {1}
S → E ·       {1}
```

```
E → Num ·     {3}
E → E · + E {3,2+3,(1+2)+3,1+(2+3)}
E → E + E · {2+3,(1+2)+3,1+(2+3)}
S → E ·       {(1+2)+3,1+(2+3)}
```

文法

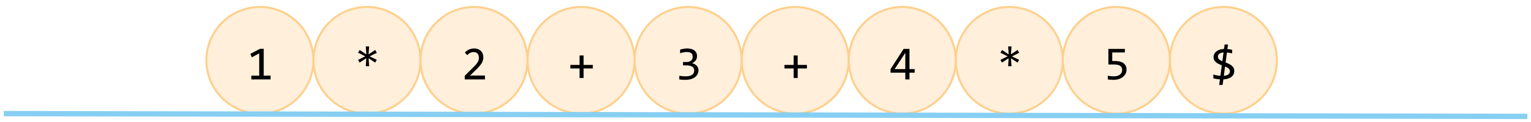
E → E + E

E → T

T → T * T

T → Num

入力



文法

E → E + E

E → T

T → T * T

T → Num

入力

1	*	2	+	3	+	4	*	5	$

a_{00}: E → ・ E + E
b_{00}: E → ・ T
c_{00}: T → ・ T * T
d_{00}: T → ・ Num

文法

E → E + E
E → T
T → T * T
T → Num

入力

1 * 2 + 3 + 4 * 5 $

a_{00}: E → ・ E + E
b_{00}: E → ・ T
c_{00}: T → ・ T * T
d_{00}: T → ・ Num

文法上の位置

0文字目から0文字目まで

文法

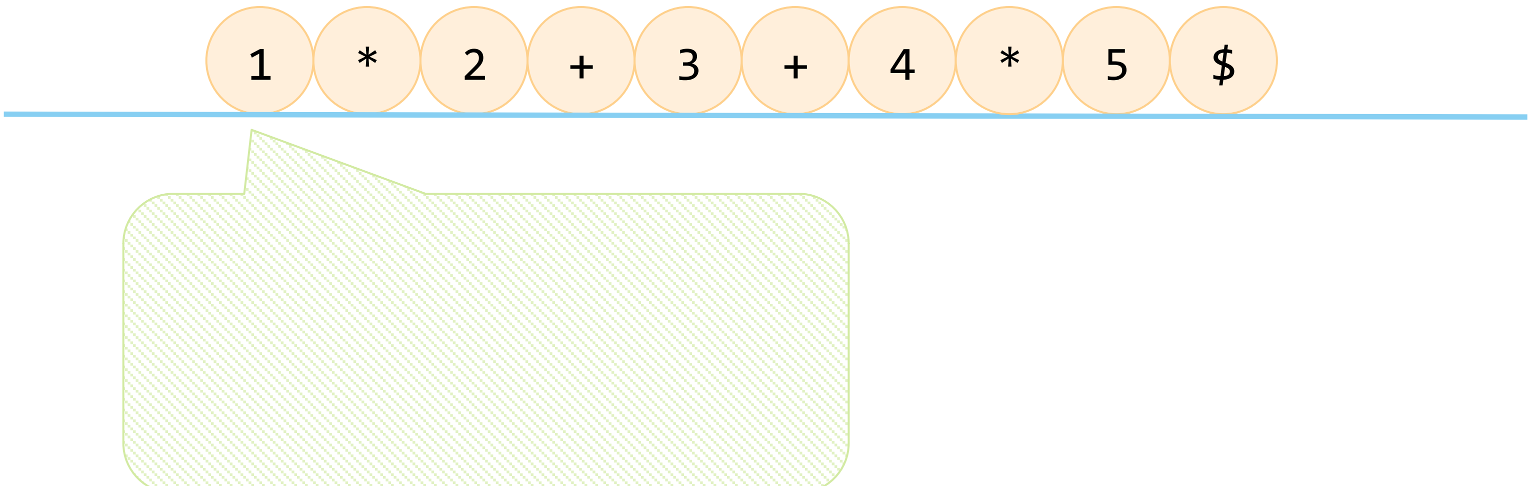
E → E + E

E → T

T → T * T

T → Num

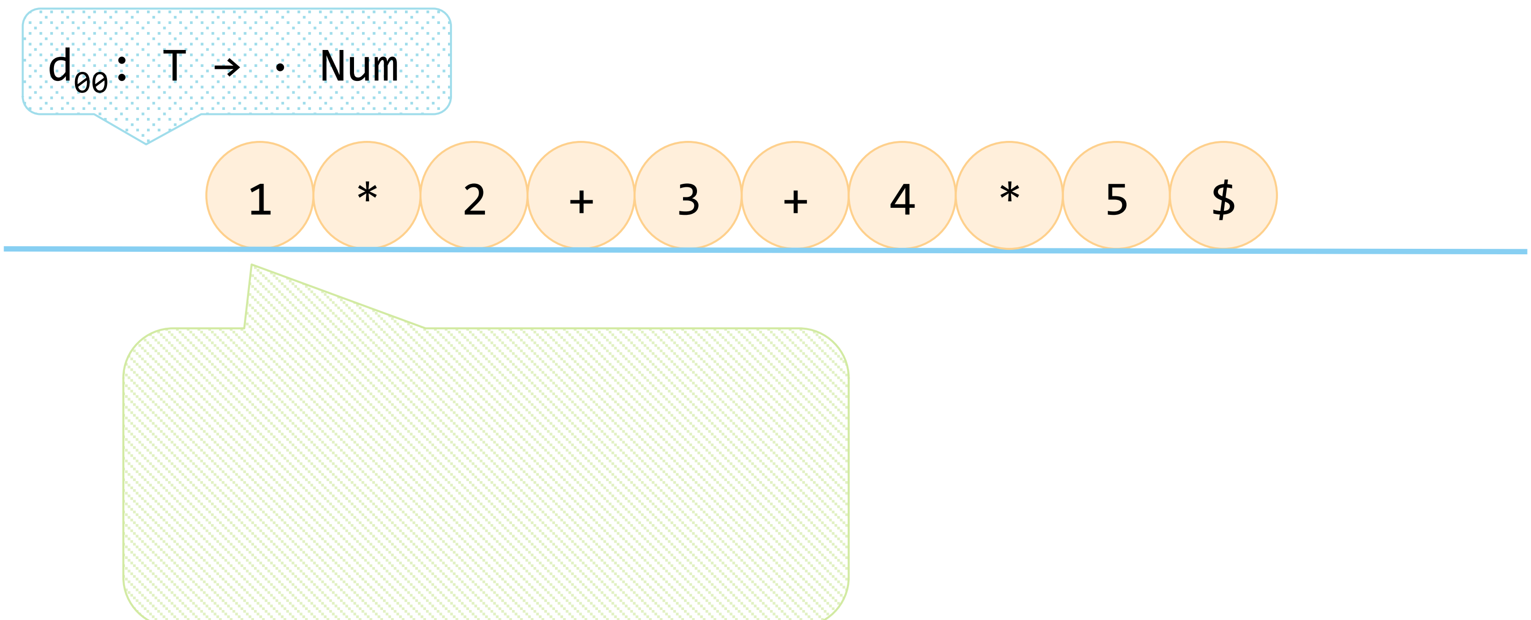
入力



文法

```
E → E + E
E → T
T → T * T
T → Num
```

入力



文法

```
E → E + E
E → T
T → T * T
T → Num
```

入力

d_{00}: T → ・ Num

1 * 2 + 3 + 4 * 5 $

a_{01}: T → Num ・ [d_{00}]

scan (≒ shift)

文法

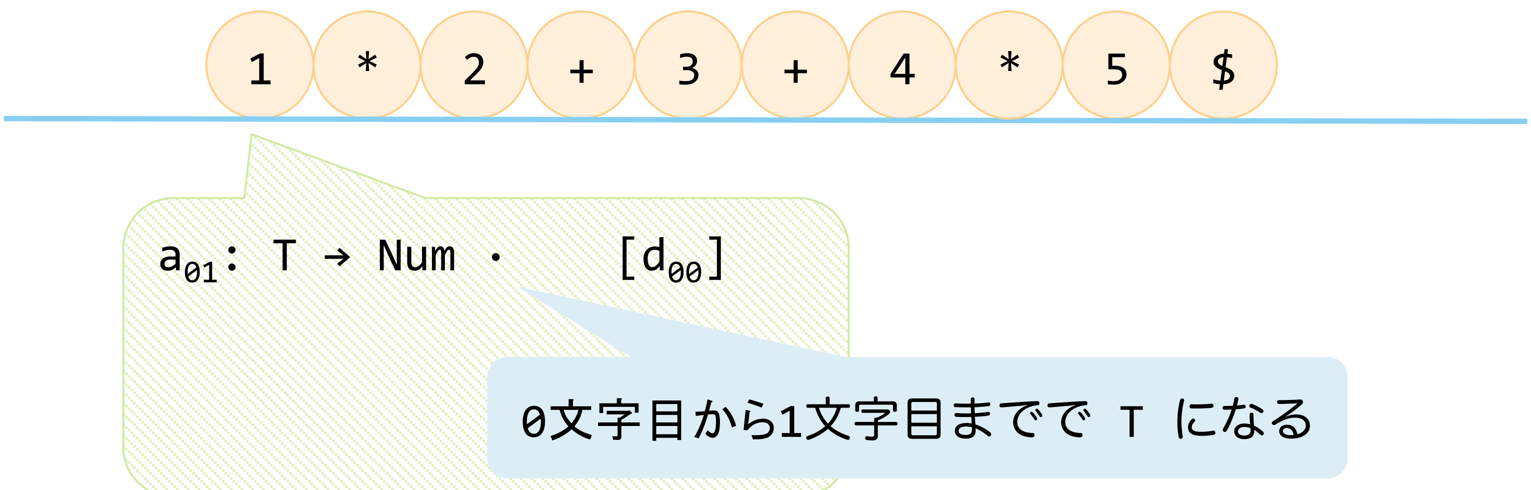
E → E + E

E → T

T → T * T

T → Num

入力



文法

E → E + E

E → T

T → T * T

T → Num

c_{00}: T → · T * T

入力

1 * 2 + 3 + 4 * 5 $

a_{01}: T → Num · [d_{00}]

文法

E → E + E

E → T

T → T * T

T → Num

c_{00}: T → · T * T

入力

1 * 2 + 3 + 4 * 5 $

a_{01}: T → Num · [d_{00}]

b_{01}: T → T · * T [c_{00}]

complete (≒ reduce)

文法

E → E + E

E → T

T → T * T

T → Num

b_{00}: E → ・ T

入力

1 * 2 + 3 + 4 * 5 $

a_{01}: T → Num ・ [d_{00}]

b_{01}: T → T ・ * T [c_{00}]

文法

E → E + E
E → T
T → T * T
T → Num

b_{00}: E → · T

入力

1	*	2	+	3	+	4	*	5	$

a_{01}: T → Num · [d_{00}]
b_{01}: T → T · * T [c_{00}]
c_{01}: E → T · [b_{00}]

complete

文法

E → E + E

E → T

T → T * T

T → Num

a_{00}: E → · E + E

入力

1 * 2 + 3 + 4 * 5 $

a_{01}: T → Num · [d_{00}]

b_{01}: T → T · * T [c_{00}]

c_{01}: E → T · [b_{00}]

文法

E → E + E
E → T
T → T * T
T → Num

a_{00}: E → · E + E

入力

1 * 2 + 3 + 4 * 5 $

a_{01}: T → Num · [d_{00}]
b_{01}: T → T · * T [c_{00}]
c_{01}: E → T · [b_{00}]
d_{01}: E → E · + E [a_{00}]

complete

文法

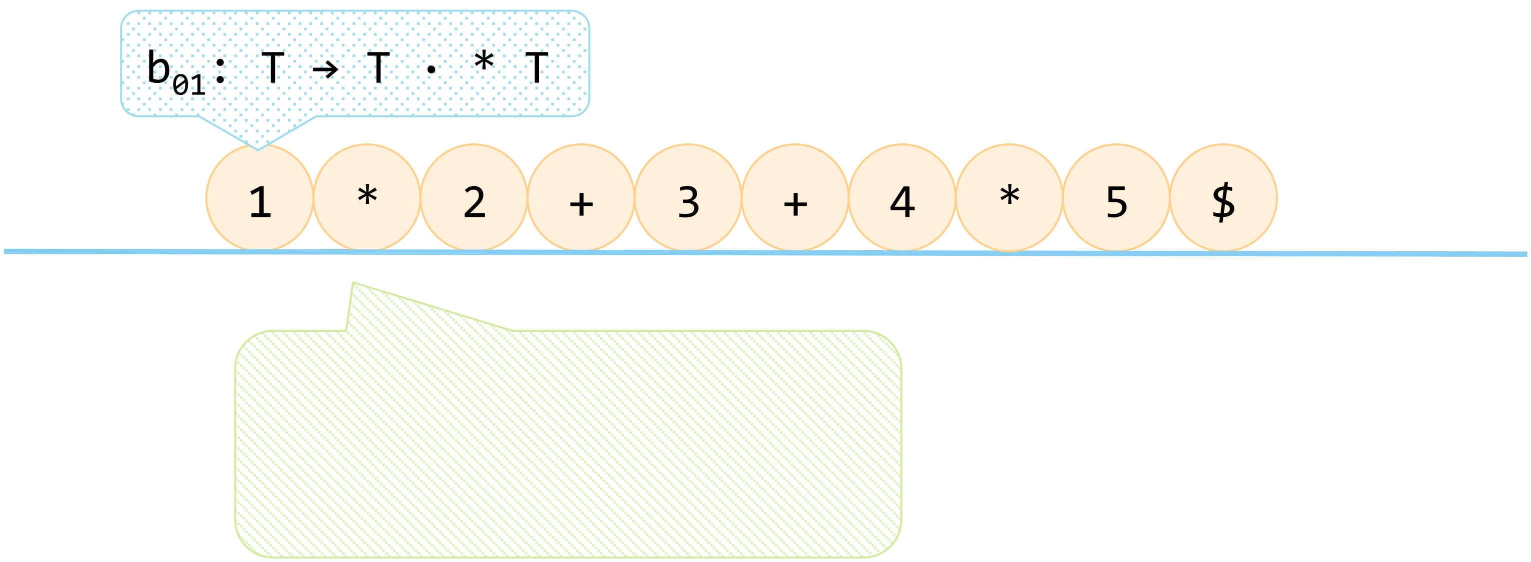
E → E + E

E → T

T → T * T

T → Num

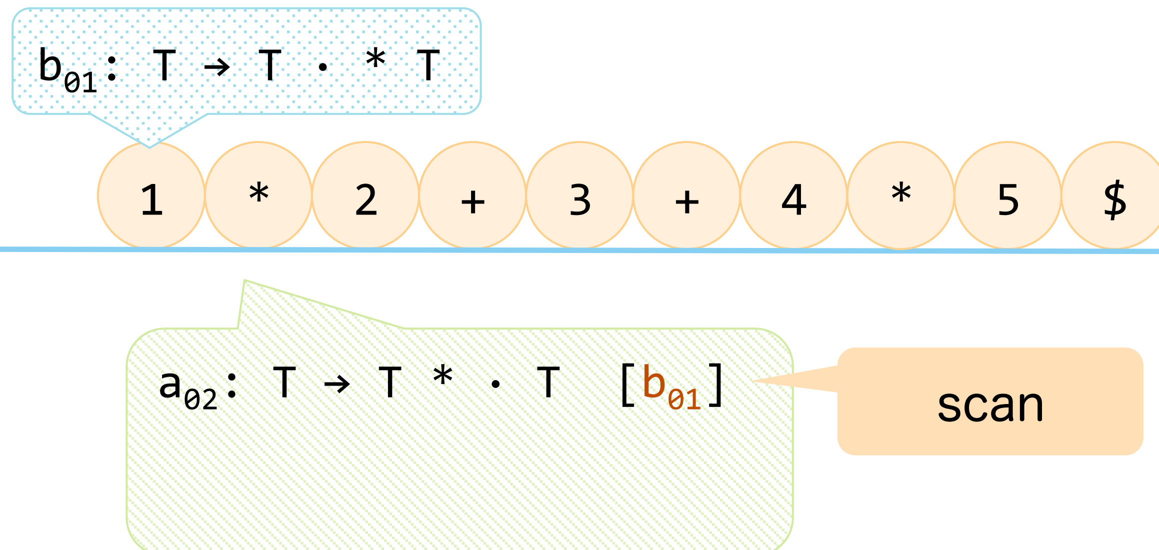
入力



文法

E → E + E
E → T
T → T * T
T → Num

入力



文法

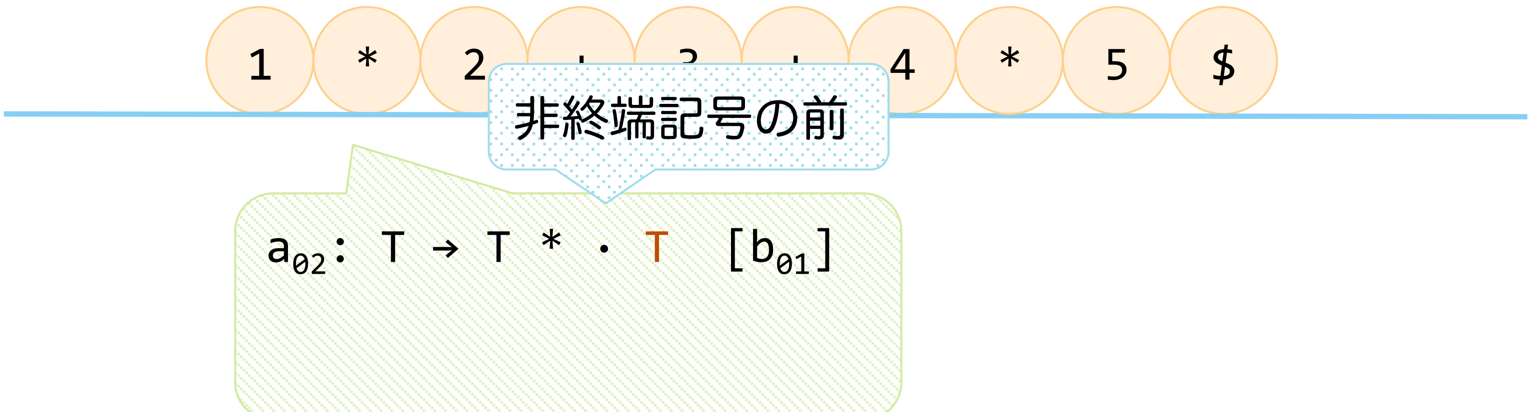
E → E + E

E → T

T → T * T

T → Num

入力



文法

E → E + E

E → T

T → T * T

T → Num

入力

1 * 2 + 3 + 4 * 5 $

a_{02}: T → T * · T [b_{01}]

b_{22}: T → · T * T [a_{02}]

c_{22}: T → · Num [a_{02}]

predict

文法

E → E + E
E → T
T → T * T
T → Num

scan → complete → predict

入力

1	*	2	+	3	+	4	*	5	$

a_{02}: T → T * · T [b_{01}]
b_{22}: T → · T * T [a_{02}]
c_{22}: T → · Num [a_{02}]

文法

E → E + E
E → T
T → T * T
T → Num

scan → complete → predict

入力

c_{22}: T → · Num

1 * 2 + 3 + 4 * 5 $

a_{23}: T → Num · [c_{22}]

scan

文法

E → E + E
E → T
T → T * T
T → Num

scan → complete → predict

a_{02}: T → T * · T
b_{22}: T → · T * T

入力

1 * 2 + 3 + 4 * 5 $

a_{23}: T → Num · [c_{22}]
b_{23}: T → T · * T [b_{22}]
c_{03}: T → T * T · [a_{02}]

complete

文法

$E \to E + E$
$E \to T$
$T \to T * T$
$T \to Num$

scan → complete → predict

a_{00}: $E \to \cdot\ E + E$
b_{00}: $E \to \cdot\ T$
c_{00}: $T \to \cdot\ T * T$

入力

1 * 2 + 3 + 4 * 5 $

a_{23}:	$T \to Num\ \cdot$		[c_{22}]
b_{23}:	$T \to T\ \cdot\ * T$		[b_{22}]
c_{03}:	$T \to T * T\ \cdot$		[a_{02}]
d_{03}:	$T \to T\ \cdot\ * T$		[c_{00}]
e_{03}:	$E \to T\ \cdot$		[b_{00}]
f_{03}:	$E \to E\ \cdot\ + E$		[a_{00}]

complete

文法

E → E + E
E → T
T → T * T
T → Num

scan → complete → predict

f_{03}: E → E · + E

入力

1	*	2	+	3	+	4	*	5	$

a_{04}: E → E + · E [f_{03}]

scan

文法

E → E + E
E → T
T → T * T
T → Num

scan → complete → predict

入力

1 * 2 + 3 + 4 * 5 $

a_{04}: E → E + · E [f_{03}]
b_{44}: E → · E + E [a_{04}]
c_{44}: E → · T [a_{04}]
d_{44}: T → · T * T [c_{44}]
e_{44}: T → · Num [c_{44}]

predict

文法

E → E + E
E → T
T → T * T
T → Num

scan → complete → predict

e_{44}: T → · Num

入力

1 * 2 + 3 + 4 * 5 $

a_{45}: T → Num · [e_{44}]

scan

文法

E → E + E
E → T
T → T * T
T → Num

scan → complete → predict

a_{04}: E → E + · E
b_{44}: E → · E + E
c_{44}: E → · T
d_{44}: T → · T * T

入力

1 * 2 + 3 + 4 * 5 $

a_{45}: T → Num · [e_{44}]
b_{45}: T → T · * T [d_{44}]
c_{45}: E → T · [c_{44}]
d_{45}: E → E · + E [b_{44}]
e_{05}: E → E + E · [a_{04}]

complete

文法

E → E + E
E → T
T → T * T
T → Num

scan → complete → predict

入力

a_{00}: E → · E + E

1 * 2 + 3 + 4 * 5 $

a_{45}:	T →	Num ·		[e_{44}]
b_{45}:	T →	T · * T		[d_{44}]
c_{45}:	E →	T ·		[c_{44}]
d_{45}:	E →	E · + E		[b_{44}]
e_{05}:	E →	E + E ·		[a_{04}]
f_{05}:	E →	E · + E		[a_{00}]

complete

文法

E → E + E
E → T
T → T * T
T → Num

scan → complete → predict

d_{45}: E → E · + E
f_{05}: E → E · + E

入力

1 * 2 + 3 + 4 * 5 $

a_{46}: E → E + · E [d_{45}]
b_{06}: E → E + · E [f_{05}]

scan

文法

E → E + E
E → T
T → T * T
T → Num

scan → complete → predict

入力

1 * 2 + 3 + 4 * 5 $

a_{46}: E → E + · E [d_{45}]
b_{06}: E → E + · E [f_{05}]
c_{66}: E → · E + E [a_{46}]
d_{66}: E → · T [a_{46}]
e_{66}: T → · T * T [d_{66}]
f_{66}: T → · Num [d_{66}]

predict

文法

E → E + E
E → T
T → T * T
T → Num

scan → complete → predict

f_{66}: T → · Num

入力

1	*	2	+	3	+	4	*	5	$

scan

a_{67}: T → Num ·   [f_{66}]

文法

E → E + E
E → T
T → T * T
T → Num

scan → complete → predict

a_{46}: E → E + · E
b_{06}: E → E + · E
c_{66}: E → · E + E
d_{66}: E → · T
e_{66}: T → · T * T

入力

1 * 2 + 3 + 4 * 5 $

complete

a_{67}: T → Num · [f_{66}]
b_{67}: T → T · * T [e_{66}]
c_{67}: E → T · [d_{66}]
b_{67}: E → E · + E [c_{66}]
e_{07}: E → E + E · [b_{06}]
f_{47}: E → E + E · [a_{46}]

文法

E → E + E
E → T
T → T * T
T → Num

scan → complete → predict

a_{04}: E → E + · E
b_{44}: E → · E + E

入力

1 * 2 + 3 + 4 * 5 $

a_{67}: T → Num · [f_{66}]
b_{67}: T → T · * T [e_{66}]
c_{67}: E → T · [d_{66}]
b_{67}: E → E · + E [c_{66}]
e_{07}: E → E + E · [b_{06}]
f_{47}: E → E + E · [a_{46}]
g_{47}: E → E · + E [b_{44}]
h_{07}: E → E + E · [a_{04}]

complete

文法

E → E + E
E → T
T → T * T
T → Num

scan → complete → predict

入力

a_{00}: E → ・ E + E

1 * 2 + 3 + 4 * 5 $

a_{67}: T → Num ・ [f_{66}]
b_{67}: T → T ・ * T [e_{66}]
c_{67}: E → T ・ [d_{66}]
b_{67}: E → E ・ + E [c_{66}]
e_{07}: E → E + E ・ [b_{06}]
f_{47}: E → E + E ・ [a_{46}]
g_{47}: E → E ・ + E [b_{44}]
h_{07}: E → E + E ・ [a_{04}]
i_{07}: E → E ・ + E [a_{00}]

complete

文法

E → E + E
E → T
T → T * T
T → Num

scan → complete → predict

b_{67}: T → T ・ * T

入力

1 * 2 + 3 + 4 * 5 $

scan

a_{68}: T → T * ・ T [b_{67}]

文法

```
E → E + E
E → T
T → T * T
T → Num
```

scan → complete → predict

入力

1	*	2	+	3	+	4	*	5	$

predict

a_{68}: T → T * · T [b_{67}]
b_{88}: T → · T * T [a_{68}]
c_{88}: T → · Num [a_{68}]

文法

E → E + E
E → T
T → T * T
T → Num

scan → complete → predict

c_{88}: T → · Num

入力

1 * 2 + 3 + 4 * 5 $

scan

a_{89}: T → Num ·   [c_{88}]

文法

E → E + E
E → T
T → T * T
T → Num

scan → complete → predict

a_{68}: T → T * · T
b_{88}: T → · T * T

入力

1 * 2 + 3 + 4 * 5 $

complete

a_{89}: T → Num · [c_{88}]
b_{89}: T → T · * T [b_{88}]
c_{69}: T → T * T · [a_{68}]

文法

$E \rightarrow E + E$

$E \rightarrow T$

$T \rightarrow T * T$

$T \rightarrow Num$

scan → complete → predict

a_{46}: $E \rightarrow E + \cdot E$
b_{06}: $E \rightarrow E + \cdot E$
c_{66}: $E \rightarrow \cdot E + E$
d_{66}: $E \rightarrow \cdot T$
e_{66}: $T \rightarrow \cdot T * T$

入力

1	*	2	+	3	+	4	*	5	$

complete

a_{89}: $T \rightarrow Num \cdot$ $[c_{88}]$
b_{89}: $T \rightarrow T \cdot * T$ $[b_{88}]$
c_{69}: $T \rightarrow T * T \cdot$ $[a_{68}]$
d_{69}: $T \rightarrow T \cdot * T$ $[e_{66}]$
e_{69}: $E \rightarrow T \cdot$ $[d_{66}]$
f_{69}: $E \rightarrow E \cdot + E$ $[c_{66}]$
g_{09}: $E \rightarrow E + E \cdot$ $[b_{06}]$
h_{49}: $E \rightarrow E + E \cdot$ $[a_{46}]$

文法

E → E + E

E → T

T → T * T

T → Num

scan → complete → predict

a_{04}: E → E + · E

b_{44}: E → · E + E

入力

1 * 2 + 3 + 4 * 5 $

a_{89}: T → Num · [c_{88}]

b_{89}: T → T · * T [b_{88}]

c_{69}: T → T * T · [a_{68}]

d_{69}: T → T · * T [e_{66}]

e_{69}: E → T · [d_{66}]

f_{69}: E → E · + E [c_{66}]

g_{09}: E → E + E · [b_{06}]

h_{49}: E → E + E · [a_{46}]

i_{49}: E → E · + E [b_{44}]

j_{09}: E → E + E · [a_{04}]

complete

文法

$E \to E + E$

$E \to T$

$T \to T * T$

$T \to \mathrm{Num}$

scan → complete → predict

入力

a_{00}: $E \to \cdot\ E + E$

1 * 2 + 3 + 4 * 5 $

a_{89}: $T \to \mathrm{Num}\ \cdot$	$[c_{88}]$	g_{09}: $E \to E + E\ \cdot$	$[b_{06}]$
b_{89}: $T \to T\ \cdot * T$	$[b_{88}]$	h_{49}: $E \to E + E\ \cdot$	$[a_{46}]$
c_{69}: $T \to T * T\ \cdot$	$[a_{68}]$	i_{49}: $E \to E\ \cdot + E$	$[b_{44}]$
d_{69}: $T \to T\ \cdot * T$	$[e_{66}]$	j_{09}: $E \to E + E\ \cdot$	$[a_{04}]$
e_{69}: $E \to T\ \cdot$	$[d_{66}]$	k_{09}: $E \to E\ \cdot + E$	$[a_{00}]$
f_{69}: $E \to E\ \cdot + E$	$[c_{66}]$		

complete

文法

E → E + E
E → T
T → T * T
T → Num

scan → complete → predict

入力

1	*	2	+	3	+	4	*	5	$

a_{89}:	T → Num ·	[c_{88}]
b_{89}:	T → T · * T	[b_{88}]
c_{69}:	T → T * T ·	[a_{68}]
d_{69}:	T → T · * T	[e_{66}]
e_{69}:	E → T ·	[d_{66}]
f_{69}:	E → E · + E	[c_{66}]
g_{09}:	E → E + E ·	[b_{06}]
h_{49}:	E → E + E ·	[a_{46}]
i_{49}:	E → E · + E	[b_{44}]
j_{09}:	E → E + E ·	[a_{04}]
k_{09}:	E → E · + E	[a_{00}]

Earley parsing [Earley 1968]

- 原理的には LR parsing と同じ
  - scan = shift, complete = reduce
  - predict は LR ではオートマトン生成時に行っている
- メモ化した再帰下降構文解析とほぼ同じ [Norvig 1991]
- 能力
  - 計算量 : $O(n^3)$
  - 言語 : Context-free language
  - 文法 : Context-free grammar

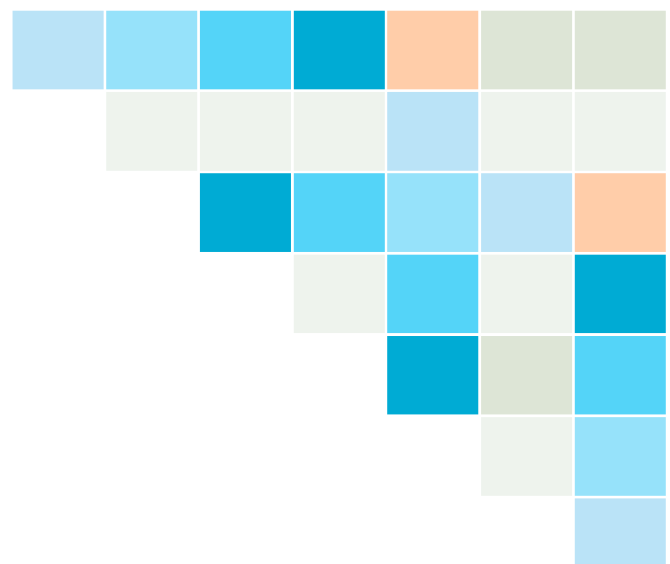
CYK algorithm [Cocke 1969, Younger 1967, Kasami 1965]

アイデア

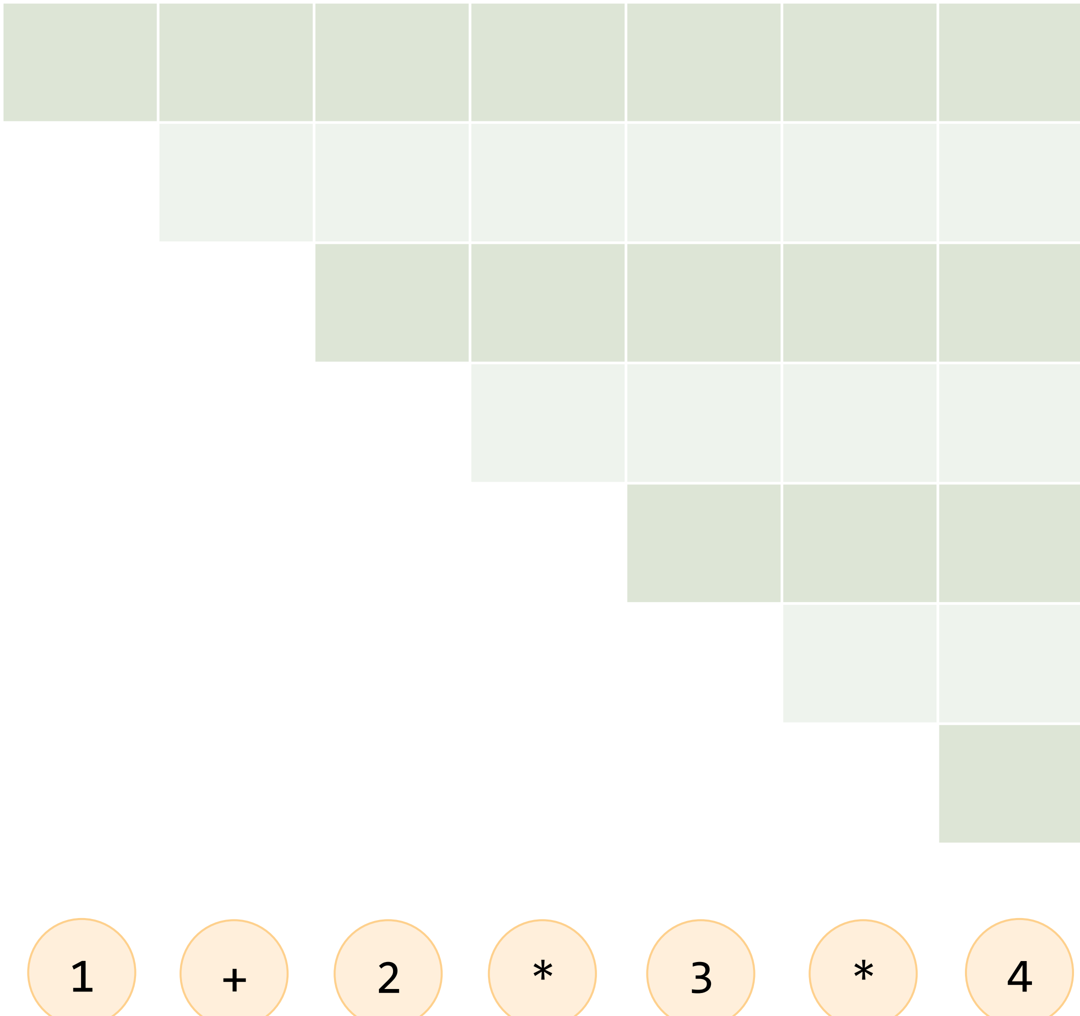
- 動的計画法
- n ~ n+k 番目のあり得る構文木を列挙していく
- Chomsky標準形の文法が対象

[Chomsky 標準形]
以下の形式の規則しか持たない：

- A → B C
- A → a
- S → ε



入力



文法

$S \to E\ E_1$

$E \to E\ E_1$

$S \to T\ T_1$

$E \to T\ T_1$

$T \to T\ T_1$

$E_1 \to N_1\ E$

$T_1 \to N_2\ T$

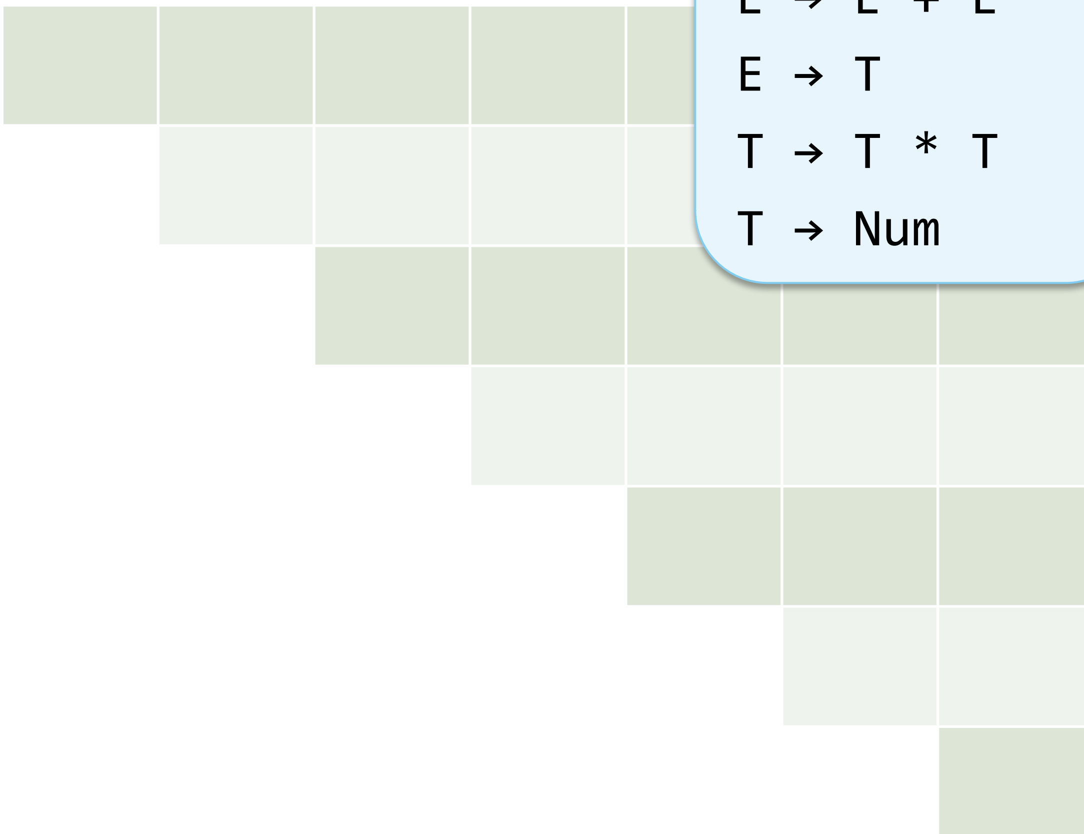
$S \to \mathrm{Num}$

$E \to \mathrm{Num}$

$T \to \mathrm{Num}$

$N_1 \to +$

$N_2 \to *$



$S \rightarrow E$

$E \rightarrow E + E$

$E \rightarrow T$

$T \rightarrow T * T$

$T \rightarrow Num$

文法

$S \rightarrow E\ E_1$

$E \rightarrow E\ E_1$

$S \rightarrow T\ T_1$

$E \rightarrow T\ T_1$

$T \rightarrow T\ T_1$

$E_1 \rightarrow N_1\ E$

$T_1 \rightarrow N_2\ T$

$S \rightarrow Num$

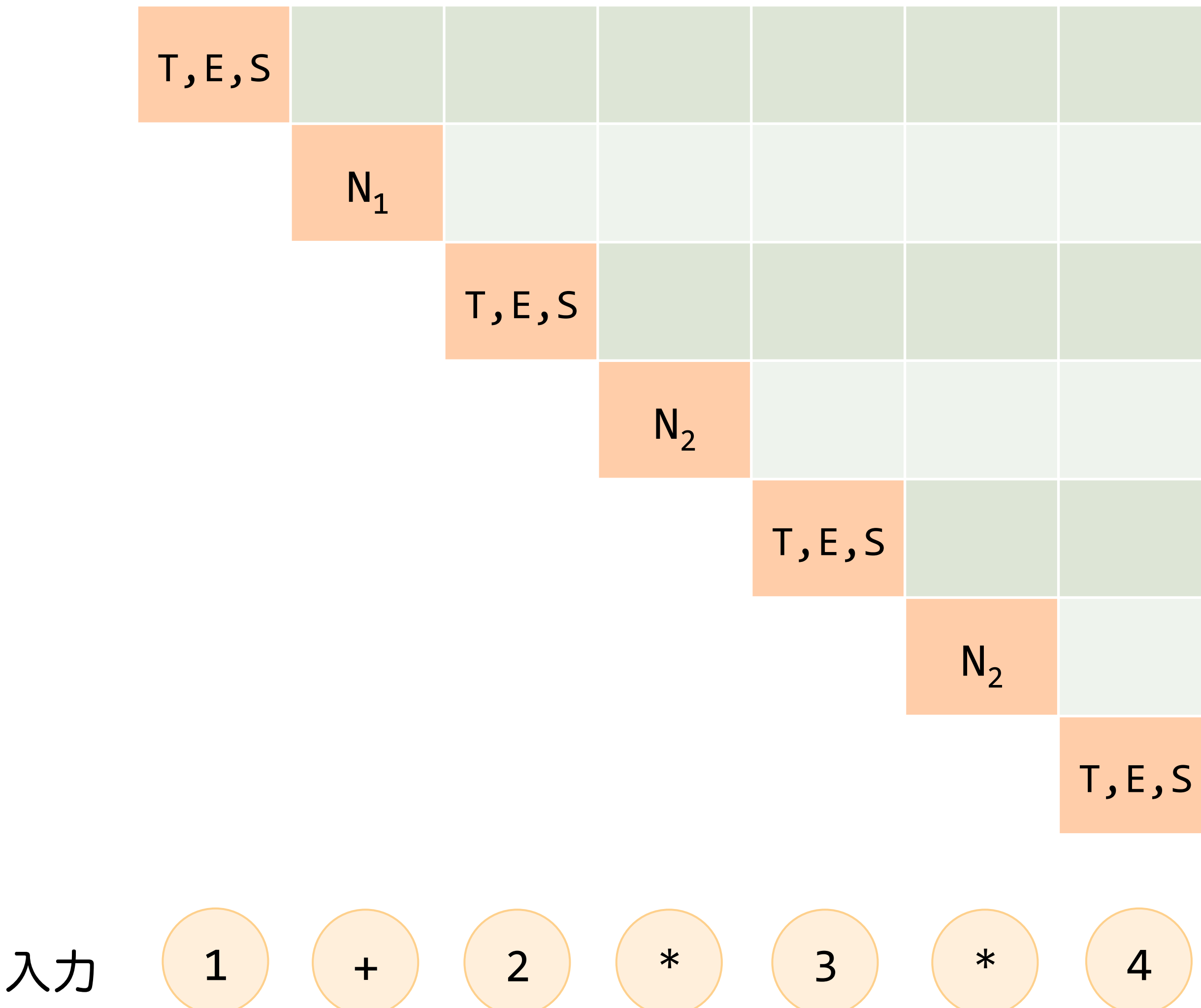
$E \rightarrow Num$

$T \rightarrow Num$

$N_1 \rightarrow +$

$N_2 \rightarrow *$

入力 1 + 2 * 3 * 4



文法

$S \to E\ E_1$

$E \to E\ E_1$

$S \to T\ T_1$

$E \to T\ T_1$

$T \to T\ T_1$

$E_1 \to N_1\ E$

$T_1 \to N_2\ T$

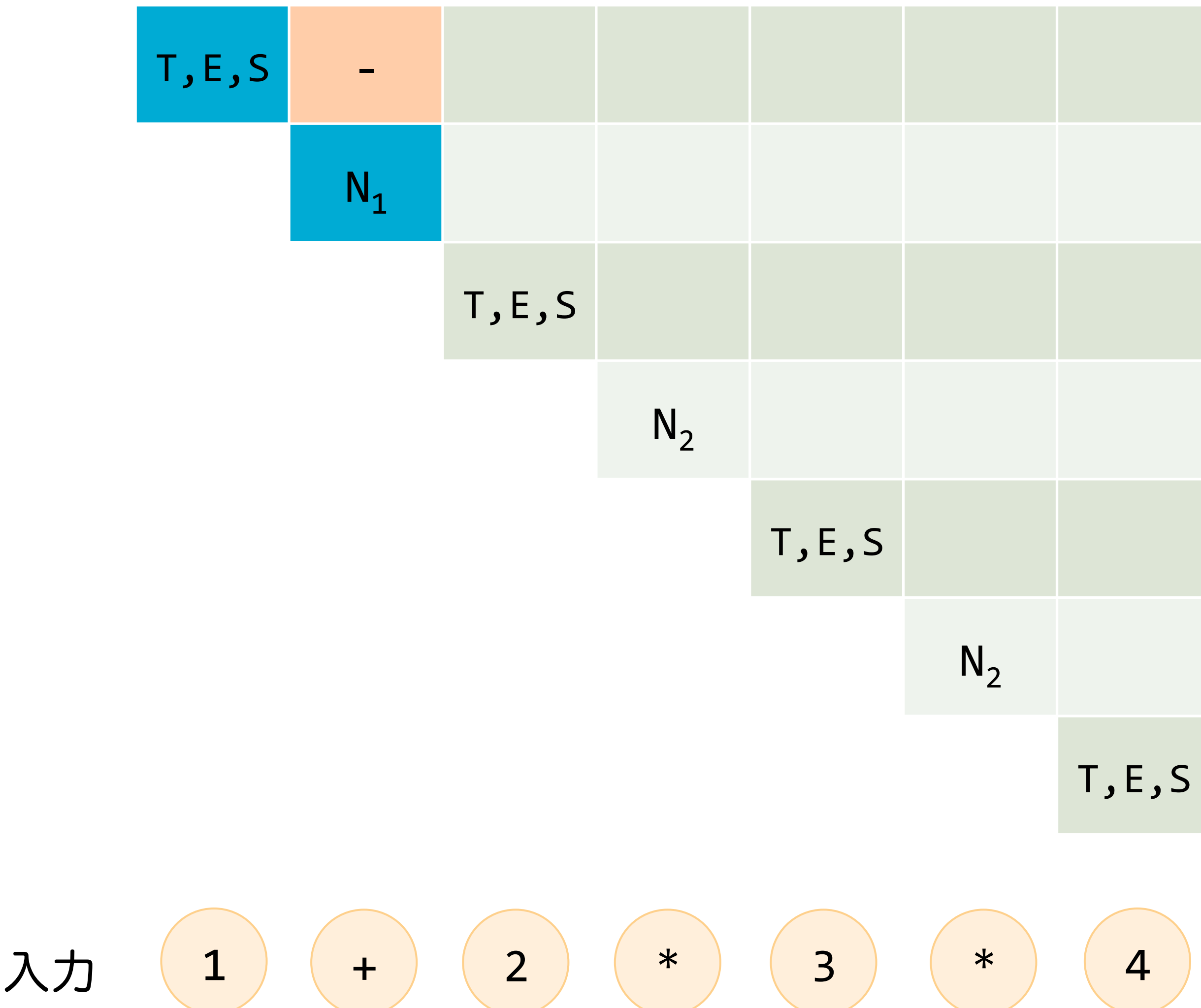
$S \to \mathrm{Num}$

$E \to \mathrm{Num}$

$T \to \mathrm{Num}$

$N_1 \to +$

$N_2 \to *$



文法

$S \rightarrow E\ E_1$

$E \rightarrow E\ E_1$

$S \rightarrow T\ T_1$

$E \rightarrow T\ T_1$

$T \rightarrow T\ T_1$

$E_1 \rightarrow N_1\ E$

$T_1 \rightarrow N_2\ T$

$S \rightarrow \mathrm{Num}$

$E \rightarrow \mathrm{Num}$

$T \rightarrow \mathrm{Num}$

$N_1 \rightarrow +$

$N_2 \rightarrow *$

T,E,S	-					
	N_1	E_1				
		T,E,S				
			N_2			
				T,E,S		
					N_2	
						T,E,S

入力 1 + 2 * 3 * 4

文法

$S \to E\ E_1$

$E \to E\ E_1$

$S \to T\ T_1$

$E \to T\ T_1$

$T \to T\ T_1$

$E_1 \to N_1\ E$

$T_1 \to N_2\ T$

$S \to Num$

$E \to Num$

$T \to Num$

$N_1 \to +$

$N_2 \to *$

T,E,S	-					
	N_1	E_1				
		T,E,S	-			
			N_2			
				T,E,S		
					N_2	
						T,E,S

入力 1 + 2 * 3 * 4

文法

$S \rightarrow E\ E_1$

$E \rightarrow E\ E_1$

$S \rightarrow T\ T_1$

$E \rightarrow T\ T_1$

$T \rightarrow T\ T_1$

$E_1 \rightarrow N_1\ E$

$T_1 \rightarrow N_2\ T$

$S \rightarrow \mathrm{Num}$

$E \rightarrow \mathrm{Num}$

$T \rightarrow \mathrm{Num}$

$N_1 \rightarrow +$

$N_2 \rightarrow *$

T,E,S	-					
	N_1	E_1				
		T,E,S	-			
			N_2	T_1		
				T,E,S		
					N_2	
						T,E,S

入力 1 + 2 * 3 * 4

文法

$S \rightarrow E\ E_1$

$E \rightarrow E\ E_1$

$S \rightarrow T\ T_1$

$E \rightarrow T\ T_1$

$T \rightarrow T\ T_1$

$E_1 \rightarrow N_1\ E$

$T_1 \rightarrow N_2\ T$

$S \rightarrow \mathrm{Num}$

$E \rightarrow \mathrm{Num}$

$T \rightarrow \mathrm{Num}$

$N_1 \rightarrow +$

$N_2 \rightarrow *$

T,E,S	-					
	N_1	E_1				
		T,E,S	-			
			N_2	T_1		
				T,E,S	-	
					N_2	
						T,E,S

入力 1 + 2 * 3 * 4

文法

$S \to E\ E_1$

$E \to E\ E_1$

$S \to T\ T_1$

$E \to T\ T_1$

$T \to T\ T_1$

$E_1 \to N_1\ E$

$T_1 \to N_2\ T$

$S \to \mathrm{Num}$

$E \to \mathrm{Num}$

$T \to \mathrm{Num}$

$N_1 \to +$

$N_2 \to *$

T,E,S	-					
	N_1	E_1				
		T,E,S	-			
			N_2	T_1		
				T,E,S	-	
					N_2	T_1
						T,E,S

入力 1 + 2 * 3 * 4

文法

$S \rightarrow E\ E_1$

$E \rightarrow E\ E_1$

$S \rightarrow T\ T_1$

$E \rightarrow T\ T_1$

$T \rightarrow T\ T_1$

$E_1 \rightarrow N_1\ E$

$T_1 \rightarrow N_2\ T$

$S \rightarrow \mathrm{Num}$

$E \rightarrow \mathrm{Num}$

$T \rightarrow \mathrm{Num}$

$N_1 \rightarrow +$

$N_2 \rightarrow *$

T,E,S	-	E,S				
	N_1	E_1				
		T,E,S	-			
			N_2	T_1		
				T,E,S	-	
					N_2	T_1
						T,E,S

入力	1	+	2	*	3	*	4

文法

$S \rightarrow E\ E_1$

$E \rightarrow E\ E_1$

$S \rightarrow T\ T_1$

$E \rightarrow T\ T_1$

$T \rightarrow T\ T_1$

$E_1 \rightarrow N_1\ E$

$T_1 \rightarrow N_2\ T$

$S \rightarrow \mathrm{Num}$

$E \rightarrow \mathrm{Num}$

$T \rightarrow \mathrm{Num}$

$N_1 \rightarrow +$

$N_2 \rightarrow *$

T,E,S	-	E,S				
	N_1	E_1	-			
		T,E,S	-			
			N_2	T_1		
				T,E,S	-	
					N_2	T_1
						T,E,S

入力						
1	+	2	*	3	*	4

文法

$S \rightarrow E\ E_1$

$E \rightarrow E\ E_1$

$S \rightarrow T\ T_1$

$E \rightarrow T\ T_1$

$T \rightarrow T\ T_1$

$E_1 \rightarrow N_1\ E$

$T_1 \rightarrow N_2\ T$

$S \rightarrow \mathrm{Num}$

$E \rightarrow \mathrm{Num}$

$T \rightarrow \mathrm{Num}$

$N_1 \rightarrow +$

$N_2 \rightarrow *$

T,E,S	-	E,S				
	N_1	E_1	-			
		T,E,S	-	T,E,S		
			N_2	T_1		
				T,E,S	-	
					N_2	T_1
						T,E,S

入力 1 + 2 * 3 * 4

文法

$S \rightarrow E\ E_1$

$E \rightarrow E\ E_1$

$S \rightarrow T\ T_1$

$E \rightarrow T\ T_1$

$T \rightarrow T\ T_1$

$E_1 \rightarrow N_1\ E$

$T_1 \rightarrow N_2\ T$

$S \rightarrow \mathrm{Num}$

$E \rightarrow \mathrm{Num}$

$T \rightarrow \mathrm{Num}$

$N_1 \rightarrow +$

$N_2 \rightarrow *$

T,E,S	-	E,S				
	N_1	E_1	-			
		T,E,S	-	T,E,S		
			N_2	T_1	-	
				T,E,S	-	
					N_2	T_1
						T,E,S

入力	1	+	2	*	3	*	4

文法

$S \rightarrow E\ E_1$

$E \rightarrow E\ E_1$

$S \rightarrow T\ T_1$

$E \rightarrow T\ T_1$

$T \rightarrow T\ T_1$

$E_1 \rightarrow N_1\ E$

$T_1 \rightarrow N_2\ T$

$S \rightarrow \mathrm{Num}$

$E \rightarrow \mathrm{Num}$

$T \rightarrow \mathrm{Num}$

$N_1 \rightarrow +$

$N_2 \rightarrow *$

T,E,S	-	E,S				
	N_1	E_1	-			
		T,E,S	-	T,E,S		
			N_2	T_1	-	
				T,E,S	-	T,E,S
					N_2	T_1
						T,E,S

入力 1 + 2 * 3 * 4

文法

$S \rightarrow E\ E_1$

$E \rightarrow E\ E_1$

$S \rightarrow T\ T_1$

$E \rightarrow T\ T_1$

$T \rightarrow T\ T_1$

$E_1 \rightarrow N_1\ E$

$T_1 \rightarrow N_2\ T$

$S \rightarrow \mathrm{Num}$

$E \rightarrow \mathrm{Num}$

$T \rightarrow \mathrm{Num}$

$N_1 \rightarrow +$

$N_2 \rightarrow *$

	T,E,S	-	E,S	-			
		N_1	E_1	-			
			T,E,S	-	T,E,S		
				N_2	T_1	-	
					T,E,S	-	T,E,S
						N_2	T_1
							T,E,S
入力	1	+	2	*	3	*	4

文法

$S \to E\ E_1$

$E \to E\ E_1$

$S \to T\ T_1$

$E \to T\ T_1$

$T \to T\ T_1$

$E_1 \to N_1\ E$

$T_1 \to N_2\ T$

$S \to \mathrm{Num}$

$E \to \mathrm{Num}$

$T \to \mathrm{Num}$

$N_1 \to +$

$N_2 \to *$

T,E,S	-	E,S	-			
	N_1	E_1	-	E_1		
		T,E,S	-	T,E,S		
			N_2	T_1	-	
				T,E,S	-	T,E,S
					N_2	T_1
						T,E,S

入力 1 + 2 * 3 * 4

文法

$S \rightarrow E\ E_1$

$E \rightarrow E\ E_1$

$S \rightarrow T\ T_1$

$E \rightarrow T\ T_1$

$T \rightarrow T\ T_1$

$E_1 \rightarrow N_1\ E$

$T_1 \rightarrow N_2\ T$

$S \rightarrow \mathrm{Num}$

$E \rightarrow \mathrm{Num}$

$T \rightarrow \mathrm{Num}$

$N_1 \rightarrow +$

$N_2 \rightarrow *$

T,E,S	-	E,S	-			
	N_1	E_1	-	E_1		
		T,E,S	-	T,E,S	-	
			N_2	T_1	-	
				T,E,S	-	T,E,S
					N_2	T_1
						T,E,S

入力 1 + 2 * 3 * 4

文法

$S \rightarrow E\ E_1$

$E \rightarrow E\ E_1$

$S \rightarrow T\ T_1$

$E \rightarrow T\ T_1$

$T \rightarrow T\ T_1$

$E_1 \rightarrow N_1\ E$

$T_1 \rightarrow N_2\ T$

$S \rightarrow \mathrm{Num}$

$E \rightarrow \mathrm{Num}$

$T \rightarrow \mathrm{Num}$

$N_1 \rightarrow +$

$N_2 \rightarrow *$

T,E,S	-	E,S	-			
	N_1	E_1	-	E_1		
		T,E,S	-	T,E,S	-	
			N_2	T_1	-	T_1
				T,E,S	-	T,E,S
					N_2	T_1
						T,E,S

入力 1 + 2 * 3 * 4

文法

$S \to E\ E_1$

$E \to E\ E_1$

$S \to T\ T_1$

$E \to T\ T_1$

$T \to T\ T_1$

$E_1 \to N_1\ E$

$T_1 \to N_2\ T$

$S \to \mathrm{Num}$

$E \to \mathrm{Num}$

$T \to \mathrm{Num}$

$N_1 \to +$

$N_2 \to *$

T,E,S	-	E,S	-	E,S		
	N_1	E_1	-	E_1		
		T,E,S	-	T,E,S	-	
			N_2	T_1	-	T_1
				T,E,S	-	T,E,S
					N_2	T_1
						T,E,S

入力 1 + 2 * 3 * 4

文法

- $S \rightarrow E\ E_1$
- $E \rightarrow E\ E_1$
- $S \rightarrow T\ T_1$
- $E \rightarrow T\ T_1$
- $T \rightarrow T\ T_1$
- $E_1 \rightarrow N_1\ E$
- $T_1 \rightarrow N_2\ T$

- $S \rightarrow \mathrm{Num}$
- $E \rightarrow \mathrm{Num}$
- $T \rightarrow \mathrm{Num}$
- $N_1 \rightarrow +$
- $N_2 \rightarrow *$

T,E,S	-	E,S	-	E,S		
	N_1	E_1	-	E_1	-	
		T,E,S	-	T,E,S	-	
			N_2	T_1	-	T_1
				T,E,S	-	T,E,S
					N_2	T_1
						T,E,S

入力	1	+	2	*	3	*	4

文法

$S \rightarrow E\ E_1$

$E \rightarrow E\ E_1$

$S \rightarrow T\ T_1$

$E \rightarrow T\ T_1$

$T \rightarrow T\ T_1$

$E_1 \rightarrow N_1\ E$

$T_1 \rightarrow N_2\ T$

$S \rightarrow \mathrm{Num}$

$E \rightarrow \mathrm{Num}$

$T \rightarrow \mathrm{Num}$

$N_1 \rightarrow +$

$N_2 \rightarrow *$

T,E,S	-	E,S	-	E,S		
	N_1	E_1	-	E_1	-	
		T,E,S	-	T,E,S	-	T,E,S
			N_2	T_1	-	T_1
				T,E,S	-	T,E,S
					N_2	T_1
						T,E,S

入力	1	+	2	*	3	*	4

文法

$S \rightarrow E\ E_1$

$E \rightarrow E\ E_1$

$S \rightarrow T\ T_1$

$E \rightarrow T\ T_1$

$T \rightarrow T\ T_1$

$E_1 \rightarrow N_1\ E$

$T_1 \rightarrow N_2\ T$

$S \rightarrow \mathrm{Num}$

$E \rightarrow \mathrm{Num}$

$T \rightarrow \mathrm{Num}$

$N_1 \rightarrow +$

$N_2 \rightarrow *$

T,E,S	-	E,S	-	E,S	-	
	N_1	E_1	-	E_1	-	
		T,E,S	-	T,E,S	-	T,E,S
			N_2	T_1	-	T_1
				T,E,S	-	T,E,S
					N_2	T_1
						T,E,S

入力 1 + 2 * 3 * 4

文法

$S \to E\ E_1$
$E \to E\ E_1$
$S \to T\ T_1$
$E \to T\ T_1$
$T \to T\ T_1$
$E_1 \to N_1\ E$
$T_1 \to N_2\ T$

$S \to \mathrm{Num}$
$E \to \mathrm{Num}$
$T \to \mathrm{Num}$
$N_1 \to +$
$N_2 \to *$

T,E,S	-	E,S	-	E,S	-	
	N_1	E_1	-	E_1	-	E_1
		T,E,S	-	T,E,S	-	T,E,S
			N_2	T_1	-	T_1
				T,E,S	-	T,E,S
					N_2	T_1
						T,E,S

入力 1 + 2 * 3 * 4

文法

$S \rightarrow E\ E_1$

$E \rightarrow E\ E_1$

$S \rightarrow T\ T_1$

$E \rightarrow T\ T_1$

$T \rightarrow T\ T_1$

$E_1 \rightarrow N_1\ E$

$T_1 \rightarrow N_2\ T$

$S \rightarrow \mathrm{Num}$

$E \rightarrow \mathrm{Num}$

$T \rightarrow \mathrm{Num}$

$N_1 \rightarrow +$

$N_2 \rightarrow *$

T,E,S	-	E,S	-	E,S	-	E,S
	N_1	E_1	-	E_1	-	E_1
		T,E,S	-	T,E,S	-	T,E,S
			N_2	T_1	-	T_1
				T,E,S	-	T,E,S
					N_2	T_1
						T,E,S
1	+	2	*	3	*	4

入力

文法

$S \rightarrow E\ E_1$

$E \rightarrow E\ E_1$

$S \rightarrow T\ T_1$

$E \rightarrow T\ T_1$

$T \rightarrow T\ T_1$

$E_1 \rightarrow N_1\ E$

$T_1 \rightarrow N_2\ T$

$S \rightarrow \mathrm{Num}$

$E \rightarrow \mathrm{Num}$

$T \rightarrow \mathrm{Num}$

$N_1 \rightarrow +$

$N_2 \rightarrow *$

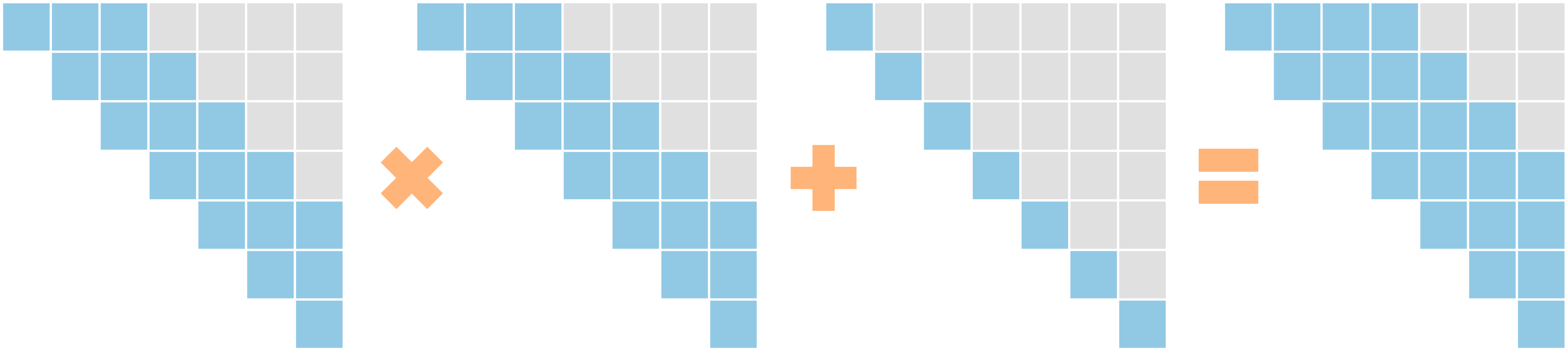
CYK algorithm [Cocke 1969, Younger 1967, Kasami 1965]

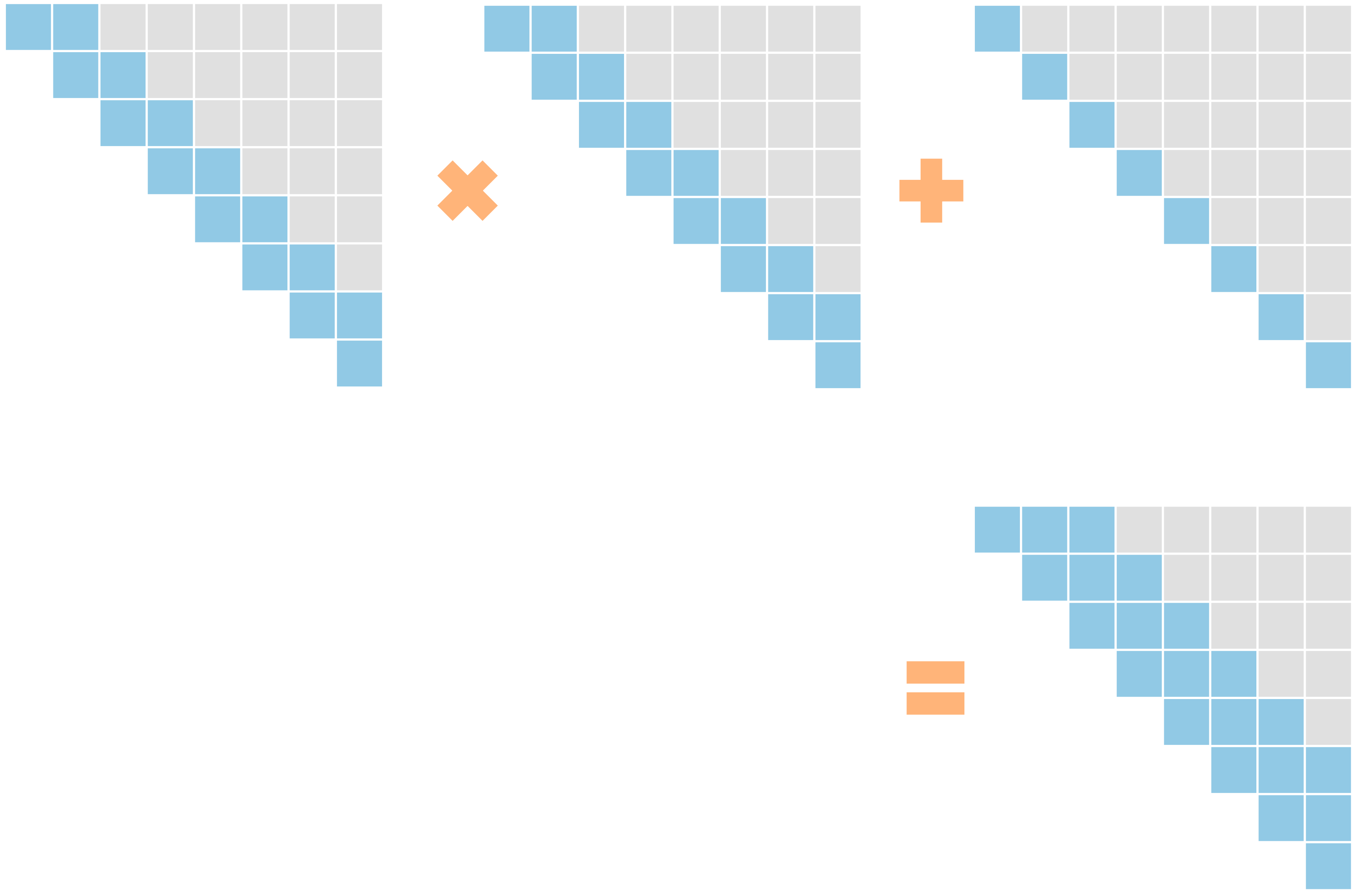
- 任意のCFGはChomsky標準形に自動変換可能
- 計算順序は変更可能
  - 左から順番に計算すると Earley parsing とほぼ同じ
- 能力
  - 計算量 : $O(n^3)$
  - 言語 : Context-free language
  - 文法 : Context-free grammar (Chomsky標準形)

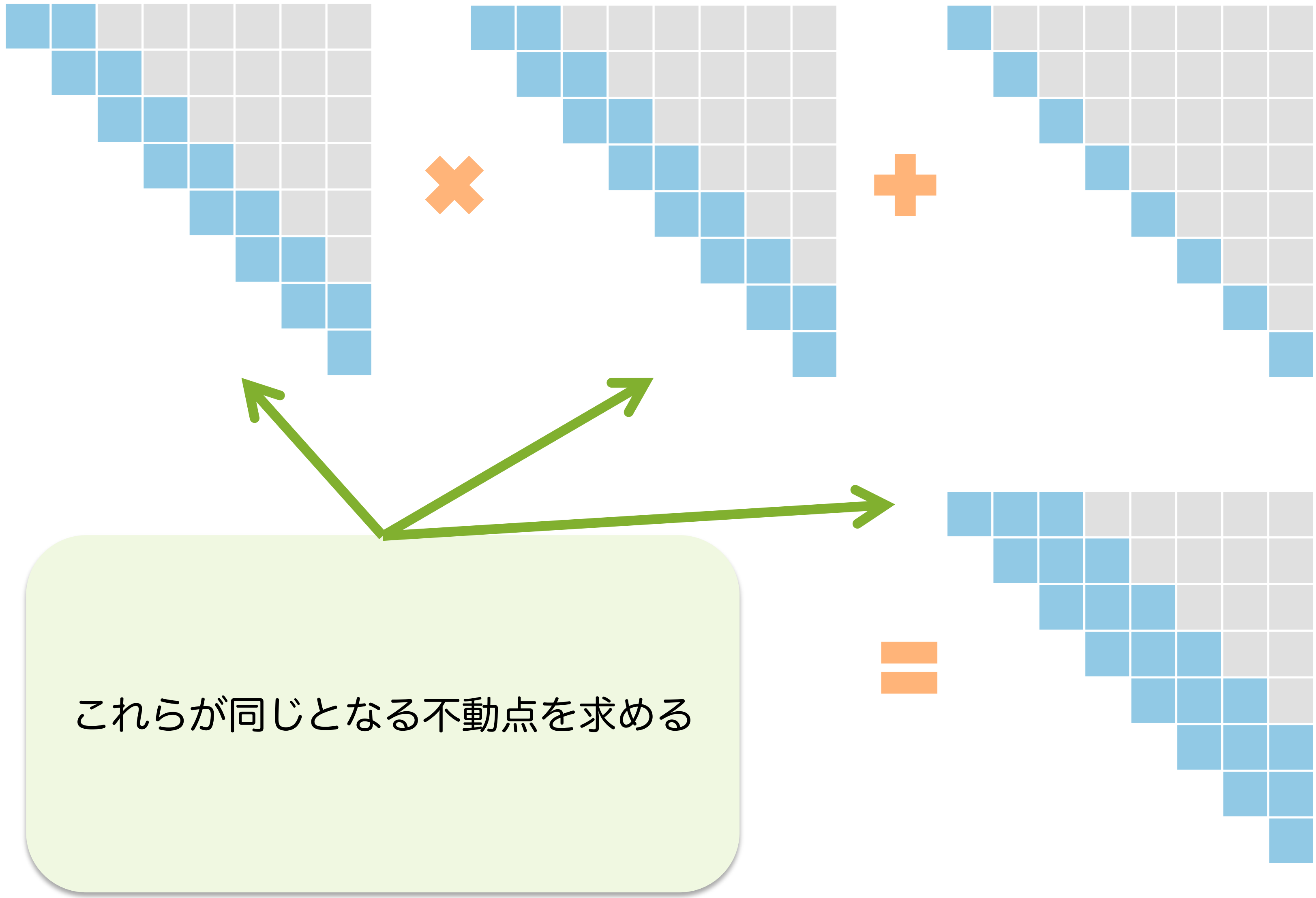
Valiant parsing [Valiant 1975]

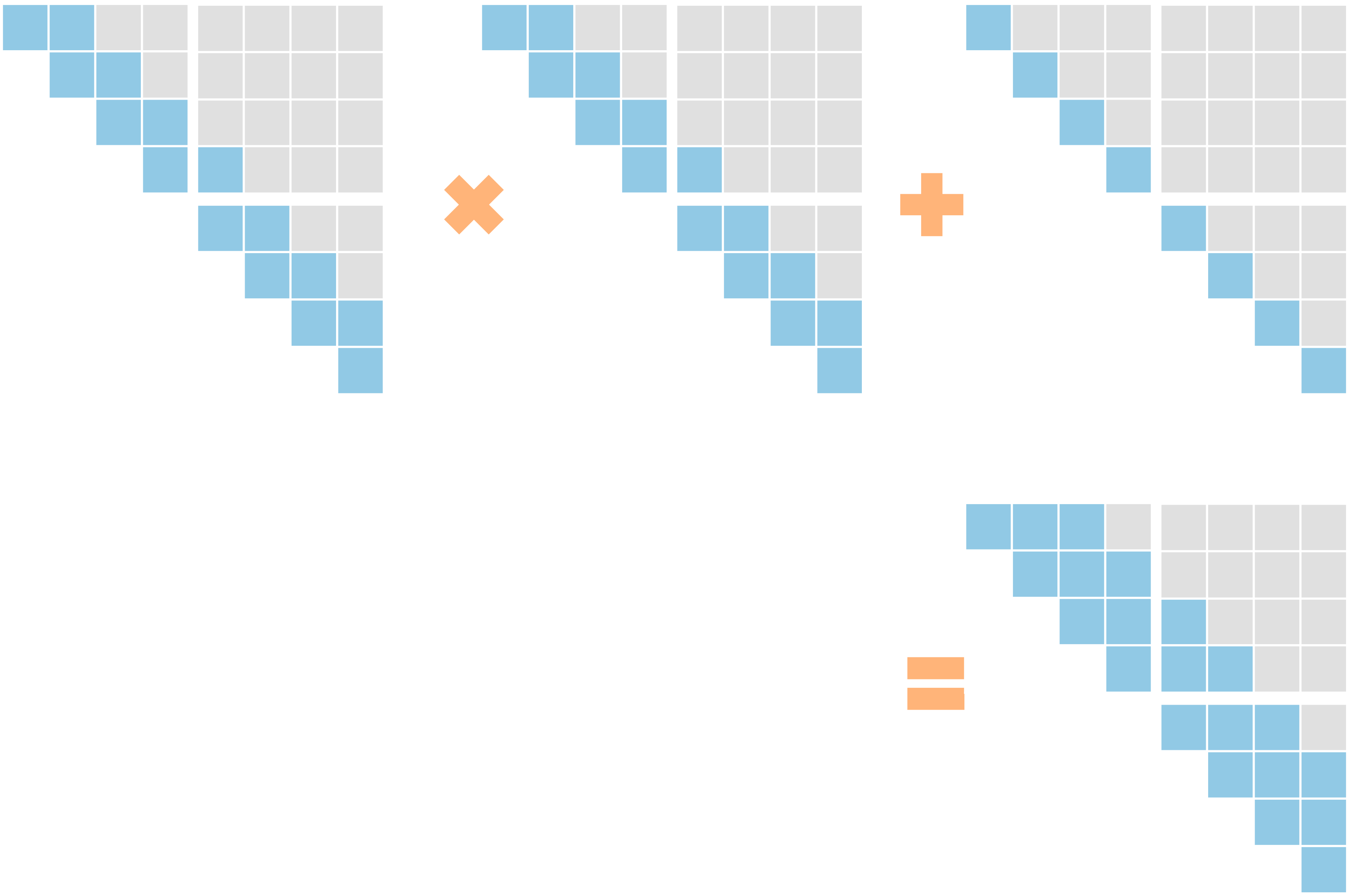
アイデア

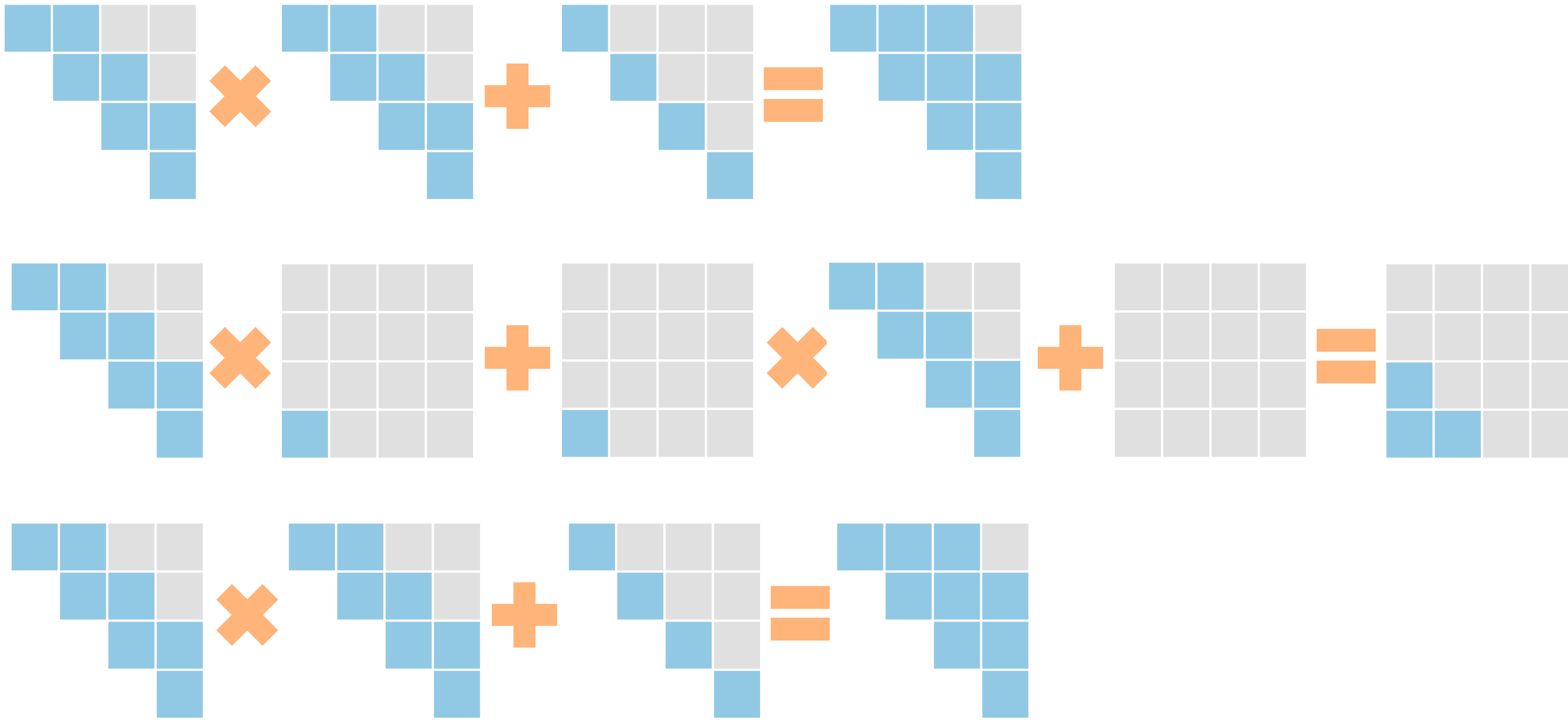
- CYKの表を行列とみなすと不動点問題に
- 行列積を使いながら帰納法的に解ける

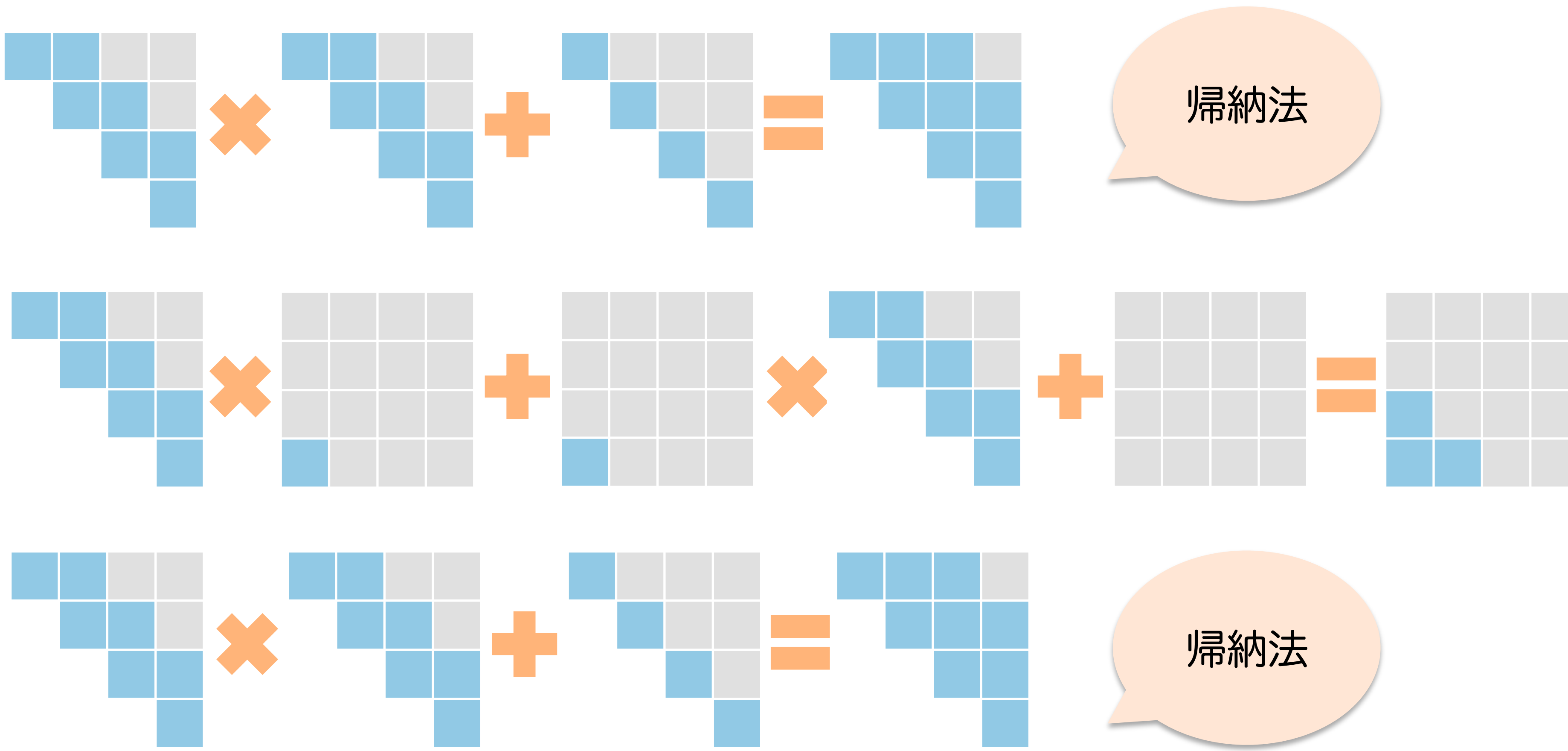


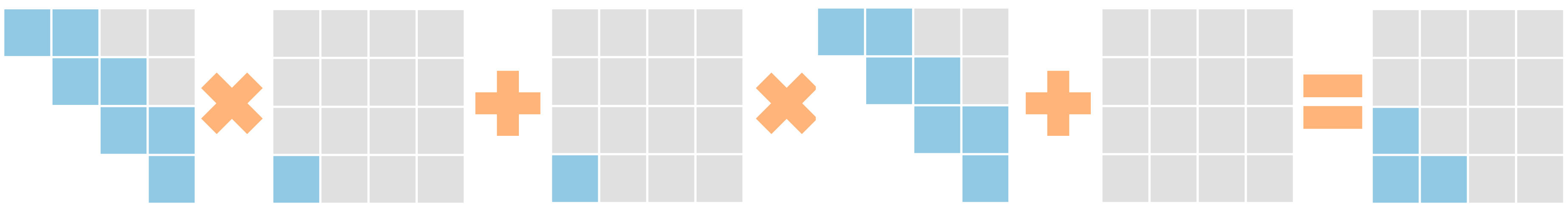


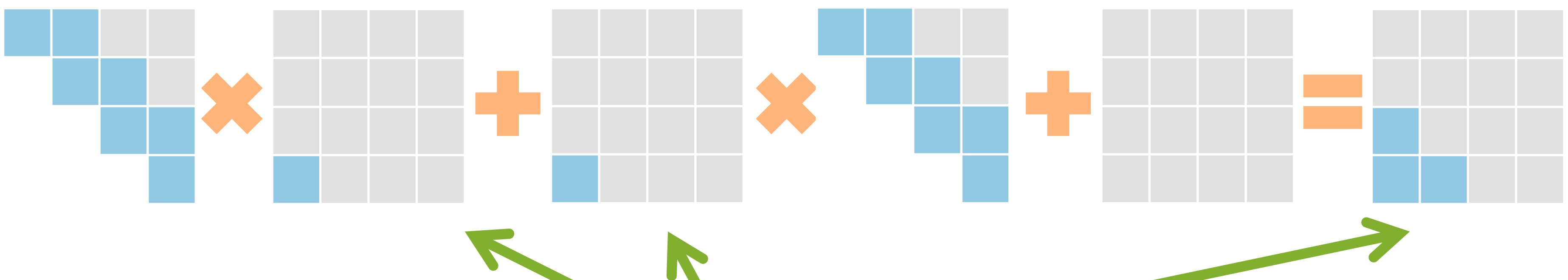

これらが同じとなる不動点を求める



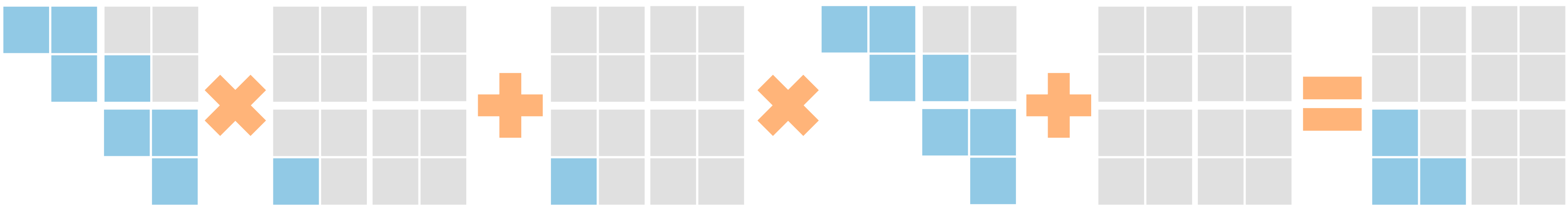



帰納法
帰納法





これらが同じなら良い



$$\begin{bmatrix} A_1 & A_2 \\ & A_3 \end{bmatrix} \times \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} + \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} \times \begin{bmatrix} B_1 & B_2 \\ & B_3 \end{bmatrix} + \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix}$$

$$A \times X + X \times B + C = X$$

$$\begin{bmatrix} A_1 & A_2 \\ & A_3 \end{bmatrix} \times \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} + \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} \times \begin{bmatrix} B_1 & B_2 \\ & B_3 \end{bmatrix} + \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix}$$

$$A \times X + X \times B + C = X$$

$$A_3 \times X_3 + X_3 \times B_1 + C_3 = X_3$$

$$\begin{bmatrix} A_1 & A_2 \\ & A_3 \end{bmatrix} \times \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} + \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} \times \begin{bmatrix} B_1 & B_2 \\ & B_3 \end{bmatrix} + \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix}$$

$$A \times X + X \times B + C = X$$

$$A_3 \times X_3 + X_3 \times B_1 + C_3 = X_3$$

帰納法

$$\begin{bmatrix} A_1 & A_2 \\ & A_3 \end{bmatrix} \times \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} + \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} \times \begin{bmatrix} B_1 & B_2 \\ & B_3 \end{bmatrix} + \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix}$$

$$A \times X + X \times B + C = X$$

$$A_1 \times X_1 + A_2 \times X_3 + X_1 \times B_1 + C_1 = X_1$$

$$\begin{bmatrix} A_1 & A_2 \\ & A_3 \end{bmatrix} \times \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} + \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} \times \begin{bmatrix} B_1 & B_2 \\ & B_3 \end{bmatrix} + \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix}$$

$$A \times X + X \times B + C = X$$

$$A_1 \times X_1 + X_1 \times B_1 + C_1' = X_1$$

$$C_1' = A_2 \times X_3 + C_1$$

$$\begin{bmatrix} A_1 & A_2 \\ & A_3 \end{bmatrix} \times \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} + \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} \times \begin{bmatrix} B_1 & B_2 \\ & B_3 \end{bmatrix} + \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix}$$

$$A \times X + X \times B + C = X$$

$$A_1 \times X_1 + X_1 \times B_1 + C_1' = X_1$$

帰納法

$$C_1' = A_2 \times X_3 + C_1$$

$$\begin{bmatrix} A_1 & A_2 \\ & A_3 \end{bmatrix} \times \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} + \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} \times \begin{bmatrix} B_1 & B_2 \\ & B_3 \end{bmatrix} + \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix}$$

$$A \times X + X \times B + C = X$$

$$A_3 \times X_4 + X_3 \times B_2 + X_4 \times B_3 + C_4 = X_4$$

$$\begin{bmatrix} A_1 & A_2 \\ & A_3 \end{bmatrix} \times \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} + \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} \times \begin{bmatrix} B_1 & B_2 \\ & B_3 \end{bmatrix} + \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix}$$

$$A \times X + X \times B + C = X$$

$$A_3 \times X_4 + X_4 \times B_3 + C_4' = X_4$$

$$C_4' = X_3 \times B_2 + C_4$$

$$\begin{pmatrix} A_1 & A_2 \\ & A_3 \end{pmatrix} \times \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix} + \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix} \times \begin{pmatrix} B_1 & B_2 \\ & B_3 \end{pmatrix} + \begin{pmatrix} C_1 & C_2 \\ C_3 & C_4 \end{pmatrix} = \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix}$$

$$A \times X + X \times B + C = X$$

$$A_3 \times X_4 + X_4 \times B_3 + C_4' = X_4$$

帰納法

$$C_4' = X_3 \times B_2 + C_4$$

$$\begin{bmatrix} A_1 & A_2 \\ & A_3 \end{bmatrix} \times \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} + \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} \times \begin{bmatrix} B_1 & B_2 \\ & B_3 \end{bmatrix} + \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix}$$

$$A \times X + X \times B + C = X$$

$$A_1 \times X_2 + A_2 \times X_4 + X_1 \times B_2 + X_2 \times B_3 + C_2 = X_2$$

$$\begin{bmatrix} A_1 & A_2 \\ & A_3 \end{bmatrix} \times \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} + \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix} \times \begin{bmatrix} B_1 & B_2 \\ & B_3 \end{bmatrix} + \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix}$$

$$A \times X + X \times B + C = X$$

$$A_1 \times X_2 + X_2 \times B_3 + C_2' = X_2$$

$$C_2' = A_2 \times X_4 + X_1 \times B_2 + C_2$$

$$\begin{pmatrix} A_1 & A_2 \\ & A_3 \end{pmatrix} \times \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix} + \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix} \times \begin{pmatrix} B_1 & B_2 \\ & B_3 \end{pmatrix} + \begin{pmatrix} C_1 & C_2 \\ C_3 & C_4 \end{pmatrix} = \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix}$$

$$A \times X + X \times B + C = X$$

$$A_1 \times X_2 + X_2 \times B_3 + C_2' = X_2$$

帰納法

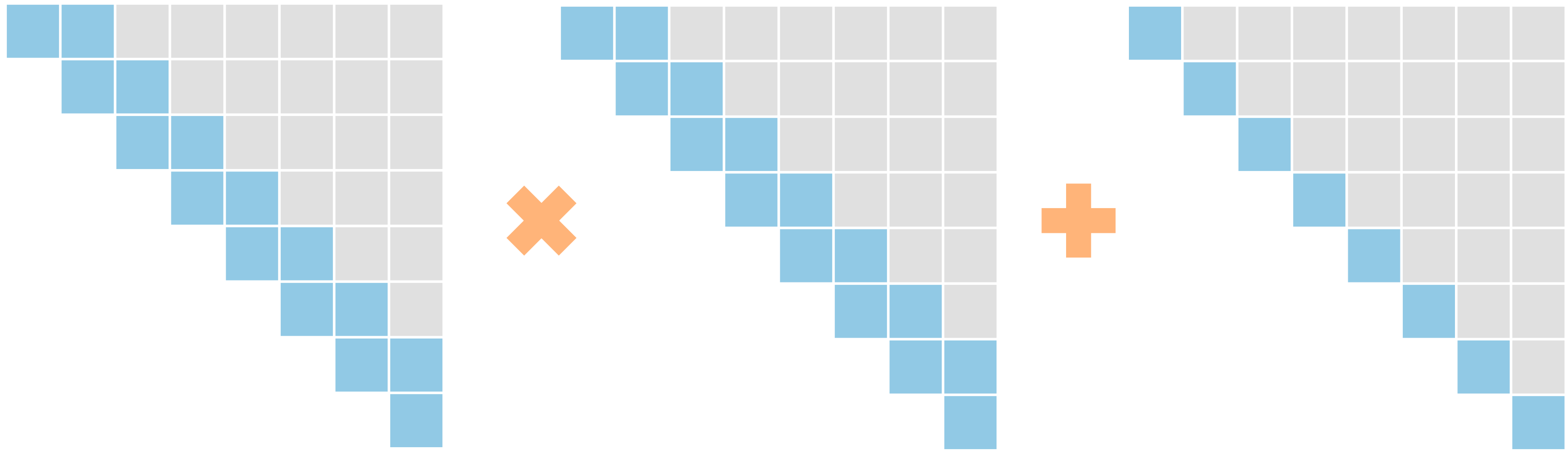
$$C_2' = A_2 \times X_4 + X_1 \times B_2 + C_2$$

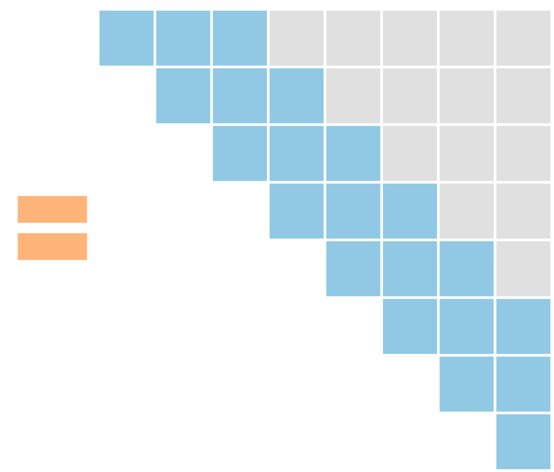
$$\begin{pmatrix} A_1 & A_2 \\ & A_3 \end{pmatrix} \times \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix} + \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix} \times \begin{pmatrix} B_1 & B_2 \\ & B_3 \end{pmatrix} + \begin{pmatrix} C_1 & C_2 \\ C_3 & C_4 \end{pmatrix} = \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix}$$

$$A \times X + X \times B + C = X$$

サイズが半分の問題4つと行列積で解ける！

→ 計算量は行列積の計算量に依存





サイズが半分の問題2つと
行列積の計算量で解ける問題に分割可能

→ 計算量は行列積の計算量に依存

Valiant parsing [Valiant 1975]

- 更に変形すると boolean 行列積に落ちる
- 計算量の下限を示すものであり実用的ではない
- 能力
  - 計算量 : $O(n^p)$ ($2 < p < 3$, boolean 行列積の計算量)
  - 言語 : Context-free language
  - 文法 : Context-free grammar (Chomsky標準形)

Top-down parsing

下向き構文解析

LL(k) parsing [Sippu 1979]

- 手書きではよく使われるが、表現力は少し弱い
- k > 1 の場合は自動生成に向かない

Packrat parsing [Ford 2002]

- 十分な表現力と解析速度を持つ
- 曖昧性を持たないプログラミング言語の構文解析に適する

Packrat parsing with left recursion [Warth 2008]

- 左再帰型の文法を扱えるように拡張したもの

GLL parsing [Scott 2010]

- 下向き構文解析で CFG を $O(n^3)$ で解析する手法

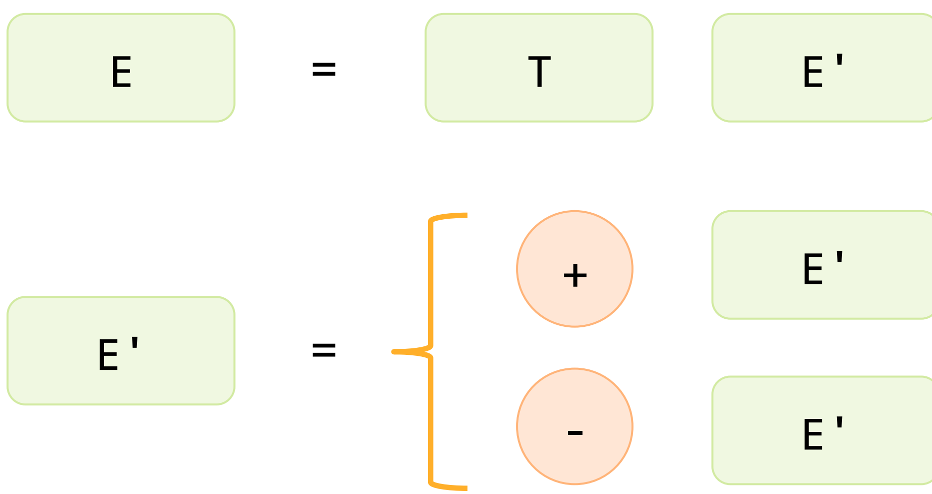
Adaptive LL(*) parsing [Parr 2014]

- LL(k) のアイデアを任意の CFG に適用

LL(k) parsing [Sippu 1979]

アイデア

- 分岐があるときは k 文字まで先読みして決める
- 本当は怖い LL(k) parsing



入力： abba

文法

S → a a A a a

S → a b A b a

A → b

A → ε

入力： abba

文法

S → a a A a a

S → a b A b a

A → b

A → ε

入力: abba

actions

```
[S] | aa → [a][aa][aaA][aaAa][aaAaa] | aa
[S] | ab → [a][ab][abA][abAb][abAba] | ab
[aaA] | aa → | aa
[aaA] | ba → [aab] | ba
[abA] | ba → | ba
[abA] | bb → [abb] | bb
```

入力: abba

逆順で書く
[xy] = z・y x

スタック

先読み

actions

```
[S] | aa → [a][aa][aaA][aaAa][aaAaa] | aa
[S] | ab → [a][ab][abA][abAb][abAba] | ab
[aaA] | aa → | aa
[aaA] | ba → [aab] | ba
[abA] | ba → | ba
[abA] | bb → [abb] | bb
```

```
[S] | abba
```

actions

```
[S] | aa → [a][aa][aaA][aaAa][aaAaa] | aa
[S] | ab → [a][ab][abA][abAb][abAba] | ab
[aaA] | aa → | aa
[aaA] | ba → [aab] | ba
[abA] | ba → | ba
[abA] | bb → [abb] | bb
```

[S] | abba

一致

actions

```
[S] | aa → [a][aa][aaA][aaAa][aaAaa] | aa
[S] | ab → [a][ab][abA][abAb][abAba] | ab
[aaA] | aa → | aa
[aaA] | ba → [aab] | ba
[abA] | ba → | ba
[abA] | bb → [abb] | bb
```

```
[S] | abba
  → [a][ab][abA][abAb][abAba] | abba
```

actions

```
[S] | aa → [a][aa][aaA][aaAa][aaAaa] | aa
[S] | ab → [a][ab][abA][abAb][abAba] | ab
[aaA] | aa → | aa
[aaA] | ba → [aab] | ba
[abA] | ba → | ba
[abA] | bb → [abb] | bb
```

```
[S] | abba
  → [a][ab][abA][abAb][abAba] | abba
```

一致

actions

```
[S] | aa → [a][aa][aaA][aaAa][aaAaa] | aa
[S] | ab → [a][ab][abA][abAb][abAba] | ab
[aaA] | aa → | aa
[aaA] | ba → [aab] | ba
[abA] | ba → | ba
[abA] | bb → [abb] | bb
```

```
[S] | abba
  → [a][ab][abA][abAb][abAba] | abba
  → [a][ab][abA][abAb] | bba
```

actions

```
[S] | aa → [a][aa][aaA][aaAa][aaAaa] | aa
[S] | ab → [a][ab][abA][abAb][abAba] | ab
[aaA] | aa → | aa
[aaA] | ba → [aab] | ba
[abA] | ba → | ba
[abA] | bb → [abb] | bb
```

```
[S] | abba
  → [a][ab][abA][abAb][abAba] | abba
  → [a][ab][abA][abAb] | bba
```

一致

actions

```
[S] | aa → [a][aa][aaA][aaAa][aaAaa] | aa
[S] | ab → [a][ab][abA][abAb][abAba] | ab
[aaA] | aa → | aa
[aaA] | ba → [aab] | ba
[abA] | ba → | ba
[abA] | bb → [abb] | bb
```

```
[S] | abba
  → [a][ab][abA][abAb][abAba] | abba
  → [a][ab][abA][abAb] | bba
  → [a][ab][abA] | ba
```

actions

```
[S] | aa → [a][aa][aaA][aaAa][aaAaa] | aa
[S] | ab → [a][ab][abA][abAb][abAba] | ab
[aaA] | aa → | aa
[aaA] | ba → [aab] | ba
[abA] | ba → | ba
[abA] | bb → [abb] | bb
```

```
[S] | abba
  → [a][ab][abA][abAb][abAba] | abba
  → [a][ab][abA][abAb] | bba
  → [a][ab][abA] | ba
```

一致

actions

```
[S] | aa → [a][aa][aaA][aaAa][aaAaa] | aa
[S] | ab → [a][ab][abA][abAb][abAba] | ab
[aaA] | aa → | aa
[aaA] | ba → [aab] | ba
[abA] | ba → | ba
[abA] | bb → [abb] | bb
```

```
[S] | abba
  → [a][ab][abA][abAb][abAba] | abba
  → [a][ab][abA][abAb] | bba
  → [a][ab][abA] | ba
  → [a][ab] | ba
```

actions

```
[S] | aa → [a][aa][aaA][aaAa][aaAaa] | aa
[S] | ab → [a][ab][abA][abAb][abAba] | ab
[aaA] | aa → | aa
[aaA] | ba → [aab] | ba
[abA] | ba → | ba
[abA] | bb → [abb] | bb
```

```
[S] | abba
  → [a][ab][abA][abAb][abAba] | abba
  → [a][ab][abA][abAb] | bba
  → [a][ab][abA] | ba
  → [a][ab] | ba
```

一致

actions

```
[S] | aa → [a][aa][aaA][aaAa][aaAaa] | aa
[S] | ab → [a][ab][abA][abAb][abAba] | ab
[aaA] | aa → | aa
[aaA] | ba → [aab] | ba
[abA] | ba → | ba
[abA] | bb → [abb] | bb
```

```
[S] | abba
  → [a][ab][abA][abAb][abAba] | abba
  → [a][ab][abA][abAb] | bba
  → [a][ab][abA] | ba
  → [a][ab] | ba
  → [a] | a
```

actions

```
[S] | aa → [a][aa][aaA][aaAa][aaAaa] | aa
[S] | ab → [a][ab][abA][abAb][abAba] | ab
[aaA] | aa → | aa
[aaA] | ba → [aab] | ba
[abA] | ba → | ba
[abA] | bb → [abb] | bb
```

```
[S] | abba
  → [a][ab][abA][abAb][abAba] | abba
  → [a][ab][abA][abAb] | bba
  → [a][ab][abA] | ba
  → [a][ab] | ba
  → [a] | a
```

一致

actions

```
[S] | aa → [a][aa][aaA][aaAa][aaAaa] | aa
[S] | ab → [a][ab][abA][abAb][abAba] | ab
[aaA] | aa → | aa
[aaA] | ba → [aab] | ba
[abA] | ba → | ba
[abA] | bb → [abb] | bb
```

```
[S] | abba
  → [a][ab][abA][abAb][abAba] | abba
  → [a][ab][abA][abAb] | bba
  → [a][ab][abA] | ba
  → [a][ab] | ba
  → [a] | a
  → |
```

actions

```
[S] | aa → [a][aa][aaA][aaAa][aaAaa] | aa
[S] | ab → [a][ab][abA][abAb][abAba] | ab
[aaA] | aa → | aa
[aaA] | ba → [aab] | ba
[abA] | ba → | ba
[abA] | bb → [abb] | bb
```

LL(k) parsing [Sippu 1979]

- LR と同様に SLL(k), LALL(k), LL(k) が存在
  - SLL(k) [Rosenkrantz 1969] は先読みに前文脈を考慮しない
- 手書きの構文解析器は大抵 SLL(k)
  - SLL(1) = LALL(1) = LL(1)
  - 任意の LL(k) 文法は SLL(k) 文法に変形可能
- 能力
  - 計算量 : O(n)
  - 言語 : LL(k) language
  - 文法 : SLL(k), LALL(k), LL(k)

Packrat parsing [Ford 2002]

アイデア

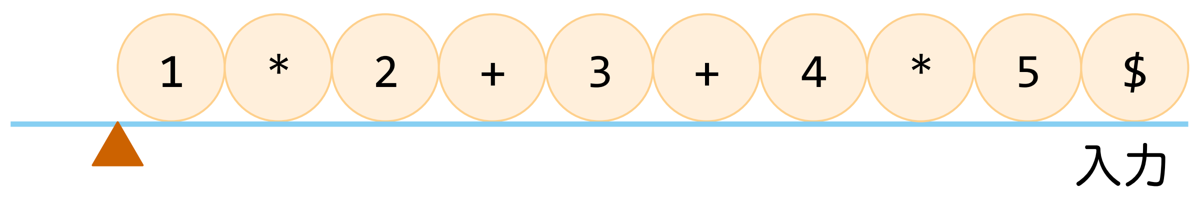
- 同じ開始位置に対して構文解析結果を一つに決める
- メモ化再帰

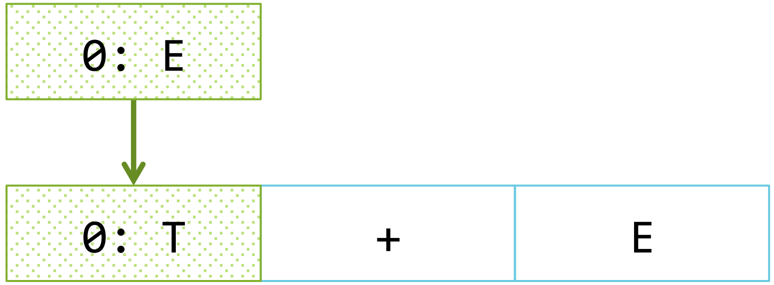
* 画像引用: en.wikipedia.org/wiki/Pack_rat

0: E

文法

E → T + E
E → T
T → Num * T
T → Num





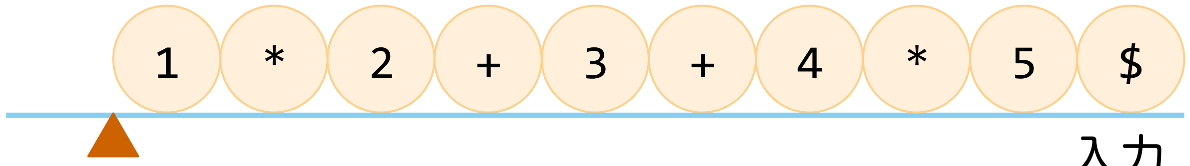
文法

E → T + E

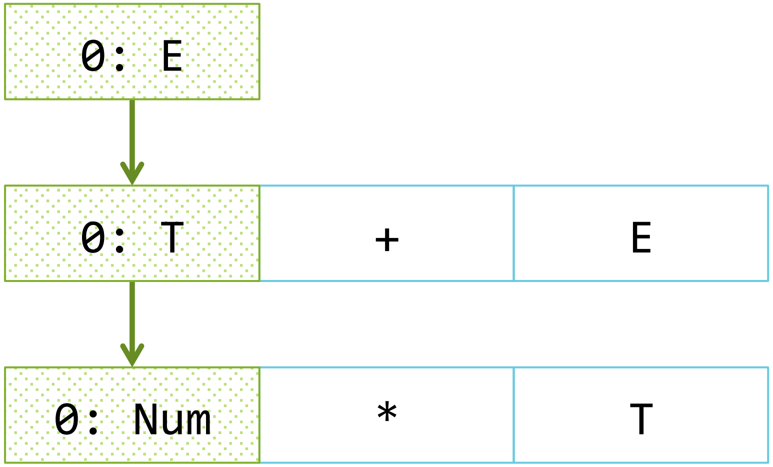
E → T

T → Num * T

T → Num

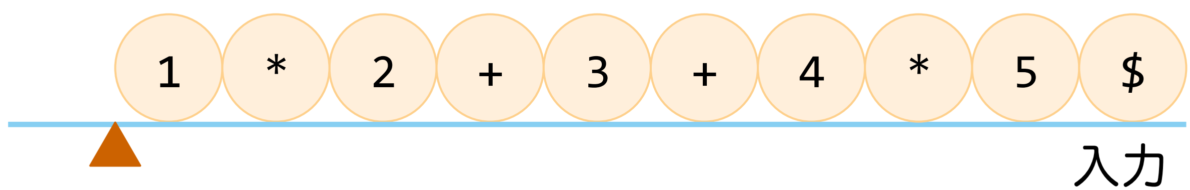


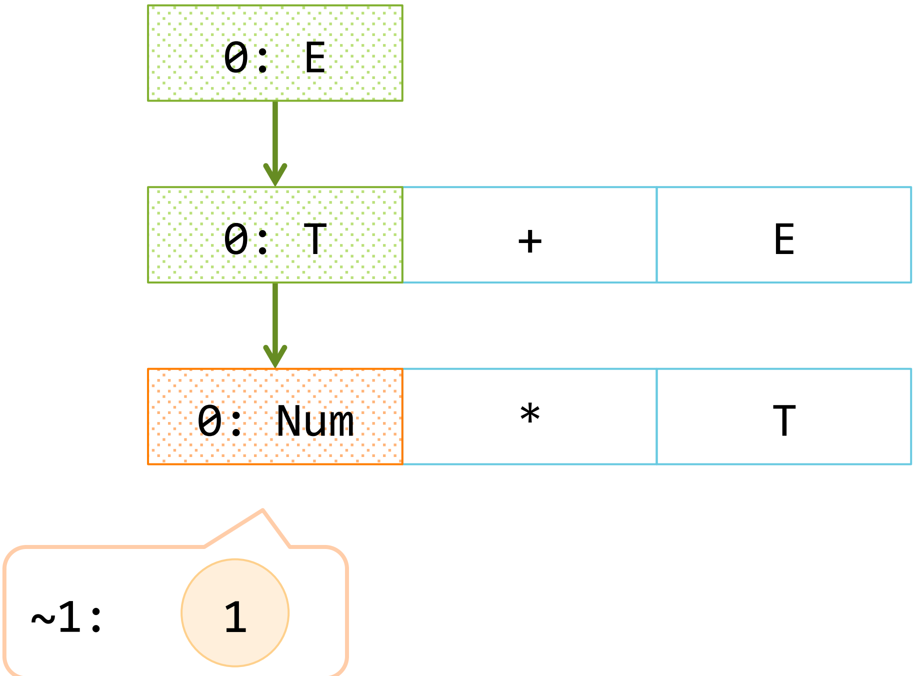
入力



文法

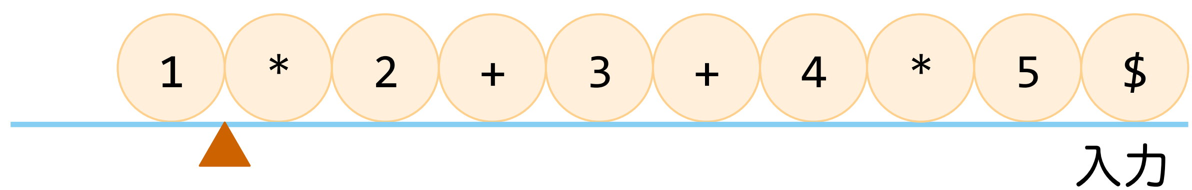
E → T + E
E → T
T → Num * T
T → Num

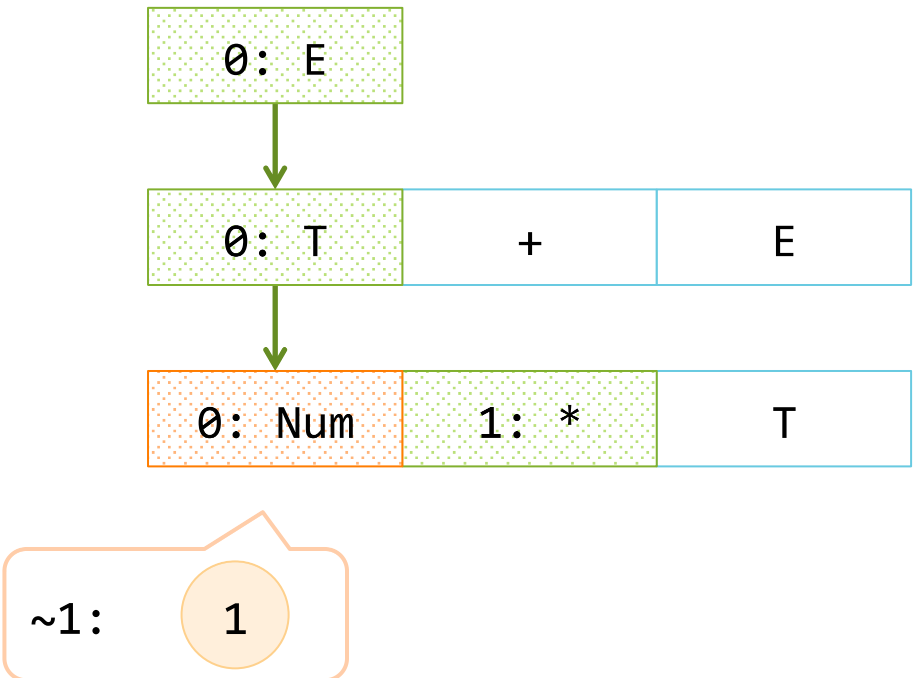




文法

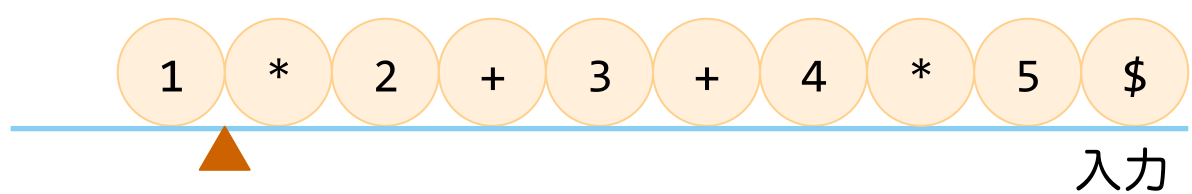
E → T + E
E → T
T → Num * T
T → Num

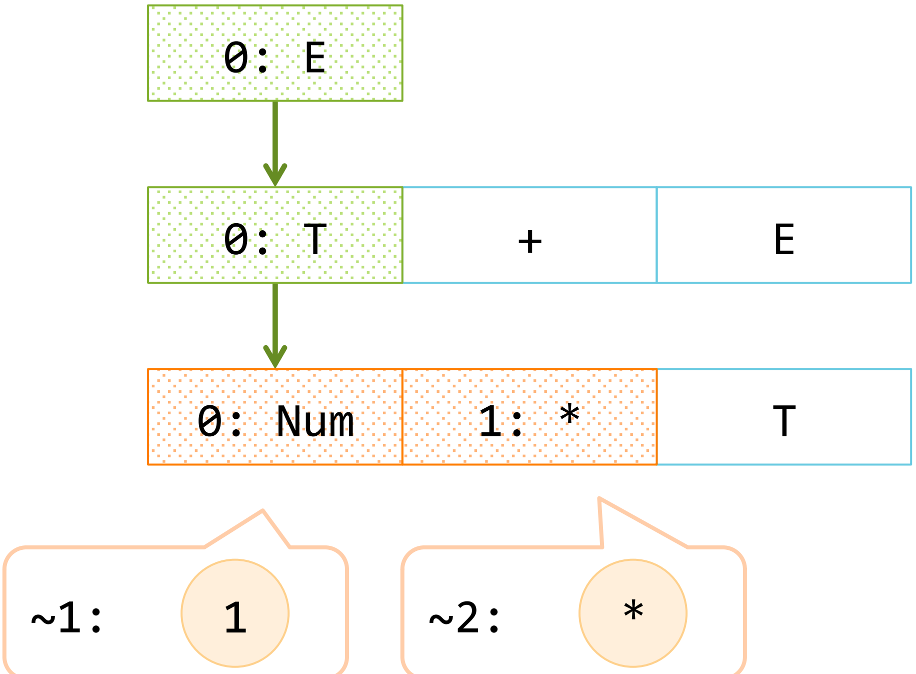




文法

E → T + E
E → T
T → Num * T
T → Num





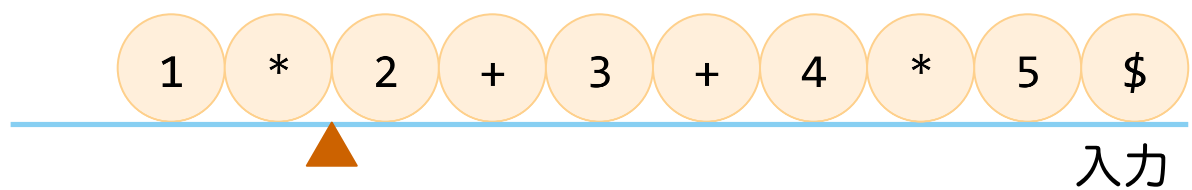
文法

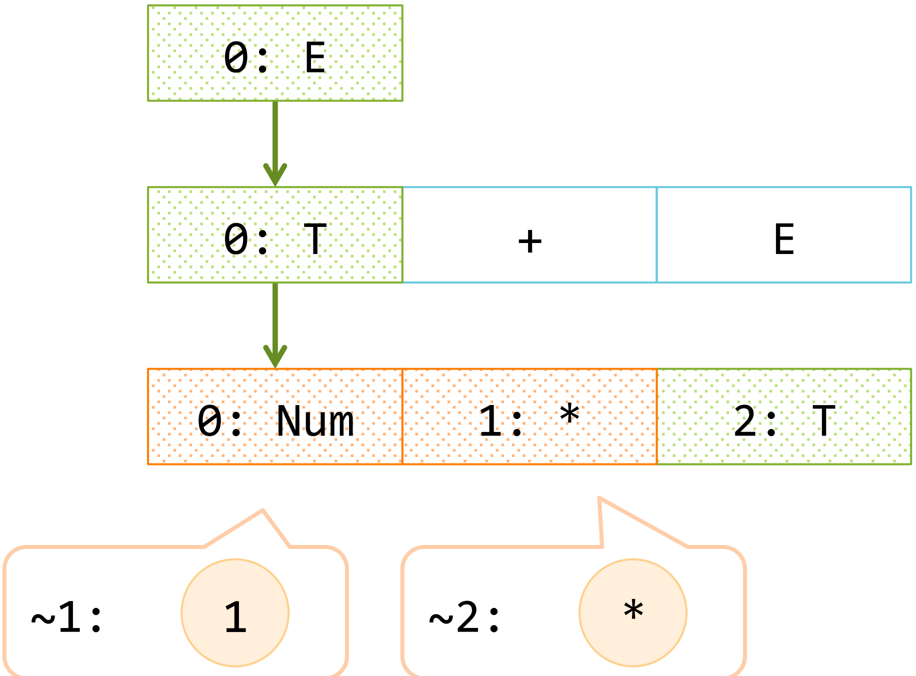
E → T + E

E → T

T → Num * T

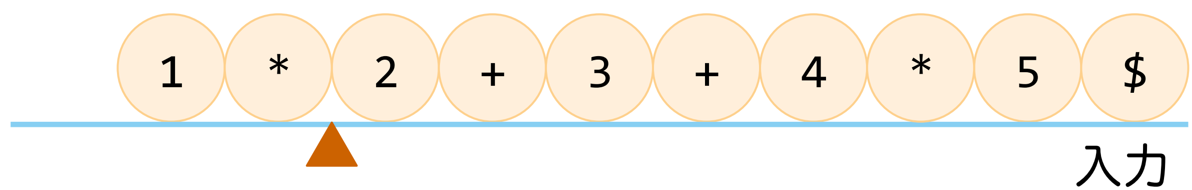
T → Num

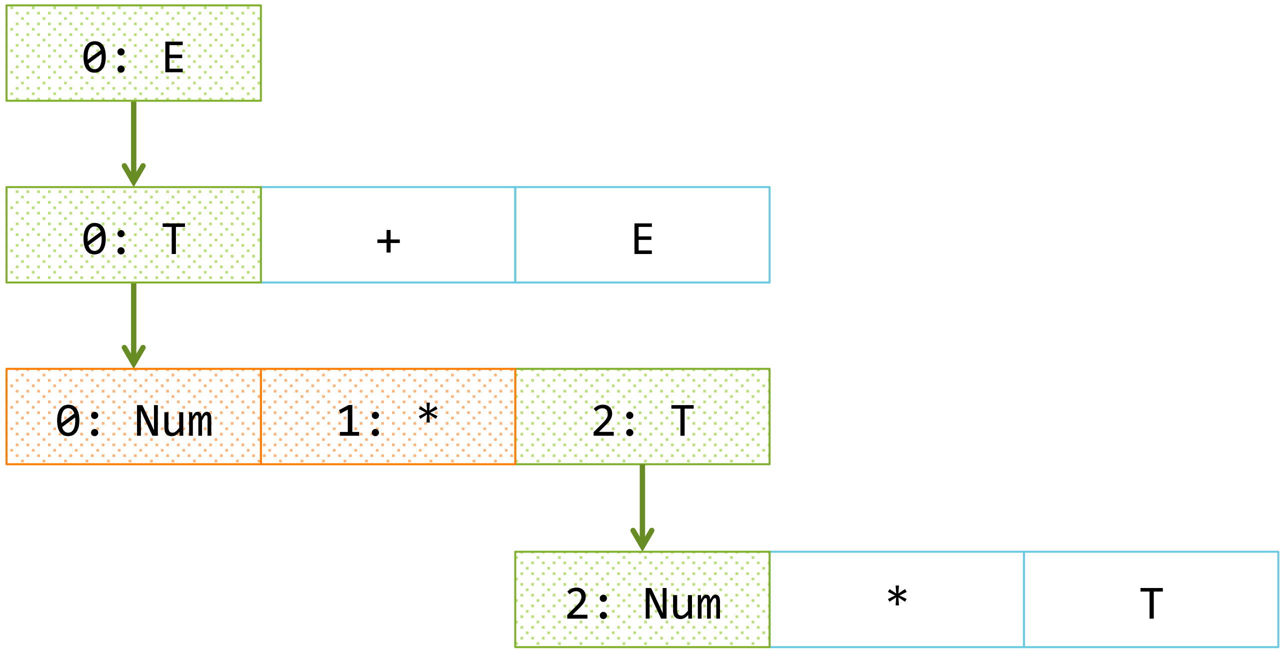




文法

E → T + E
E → T
T → Num * T
T → Num





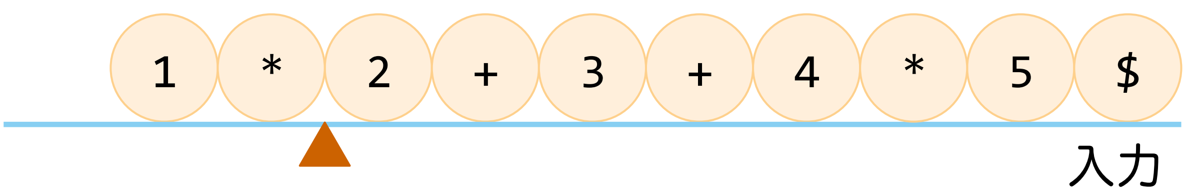
文法

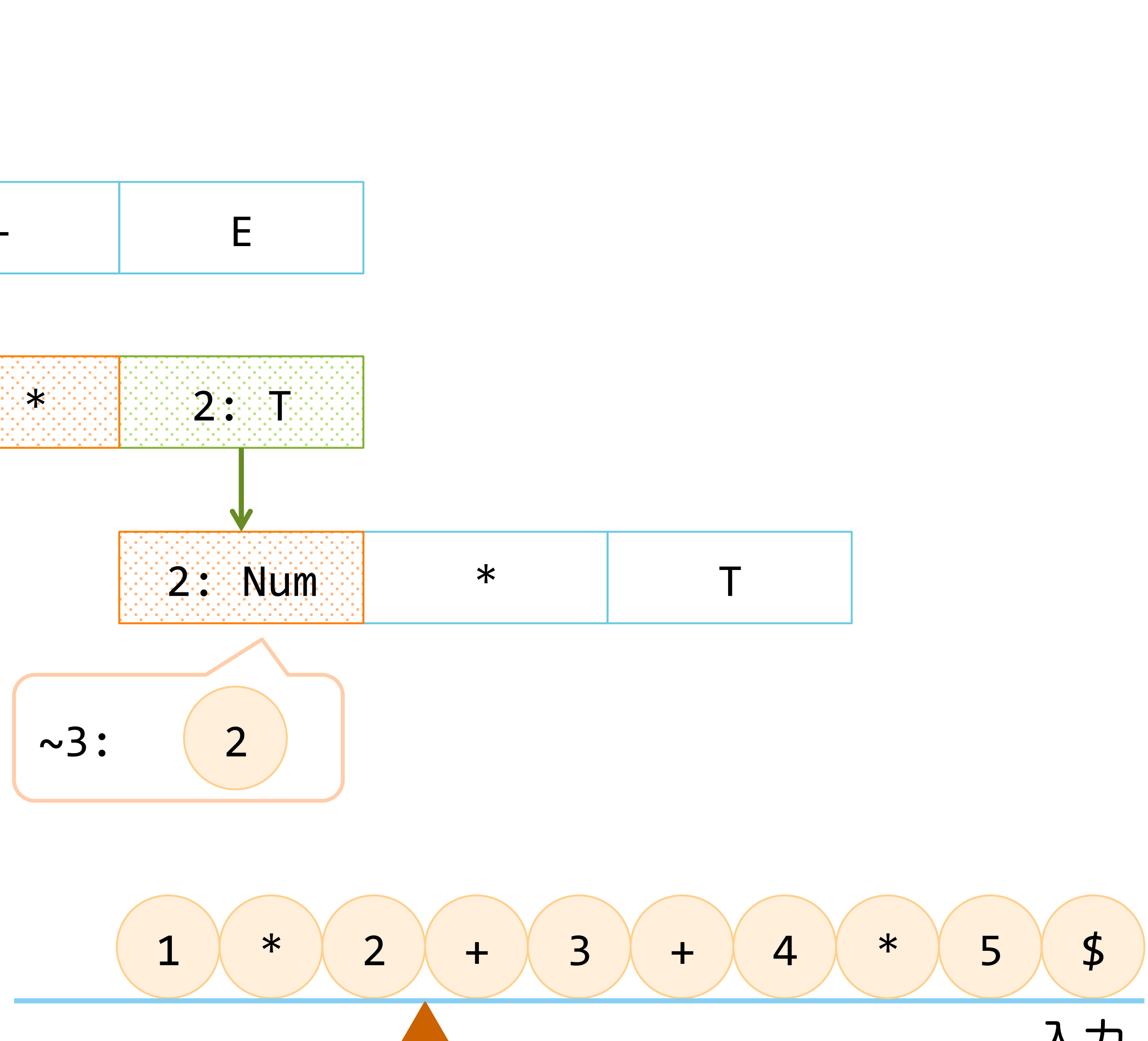
E → T + E

E → T

T → Num * T

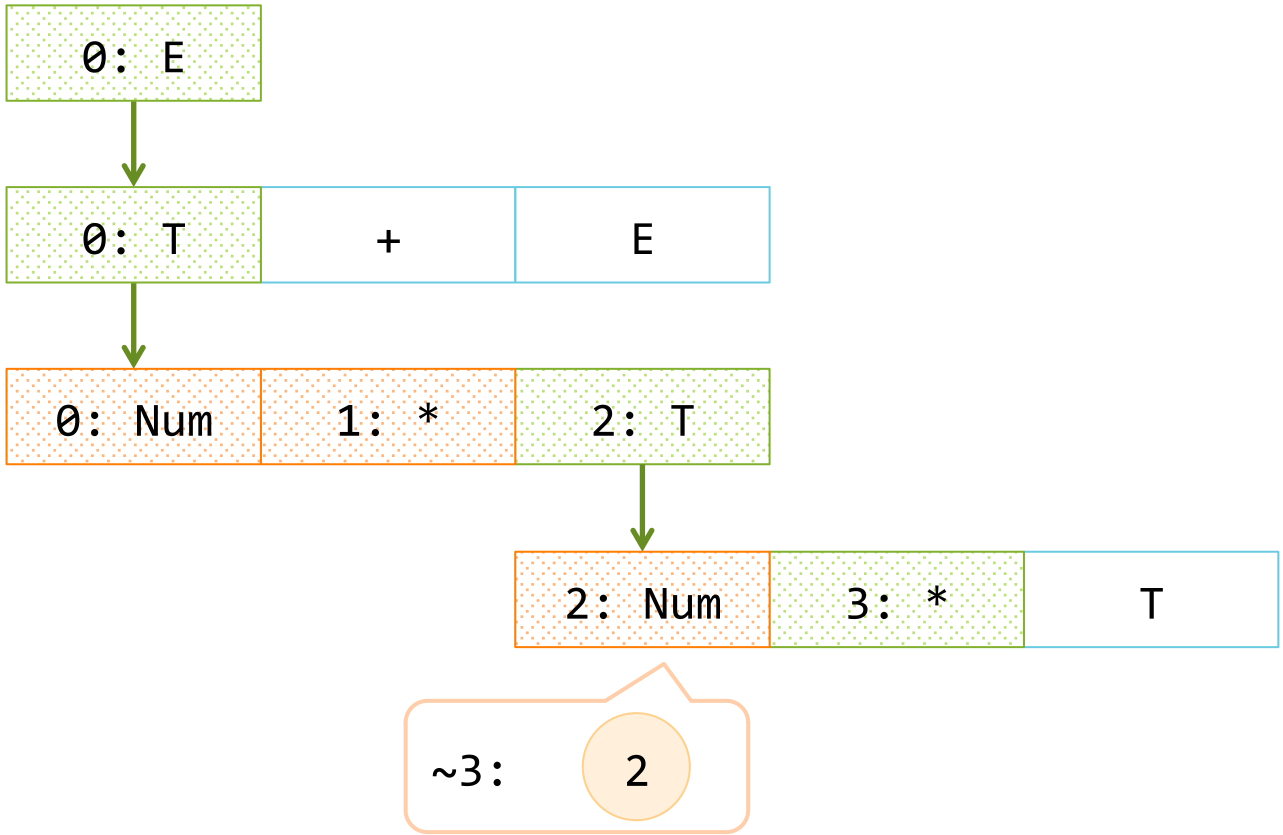
T → Num





文法

E → T + E
E → T
T → Num * T
T → Num



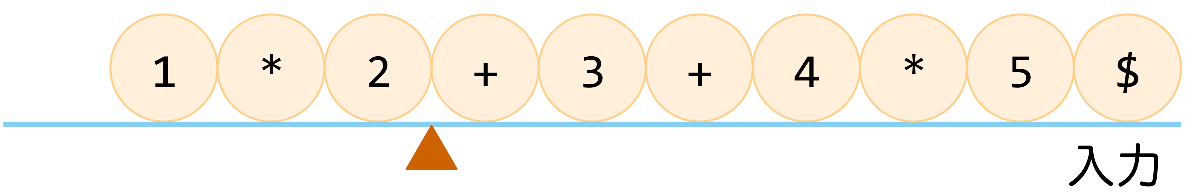
文法

E → T + E

E → T

T → Num * T

T → Num



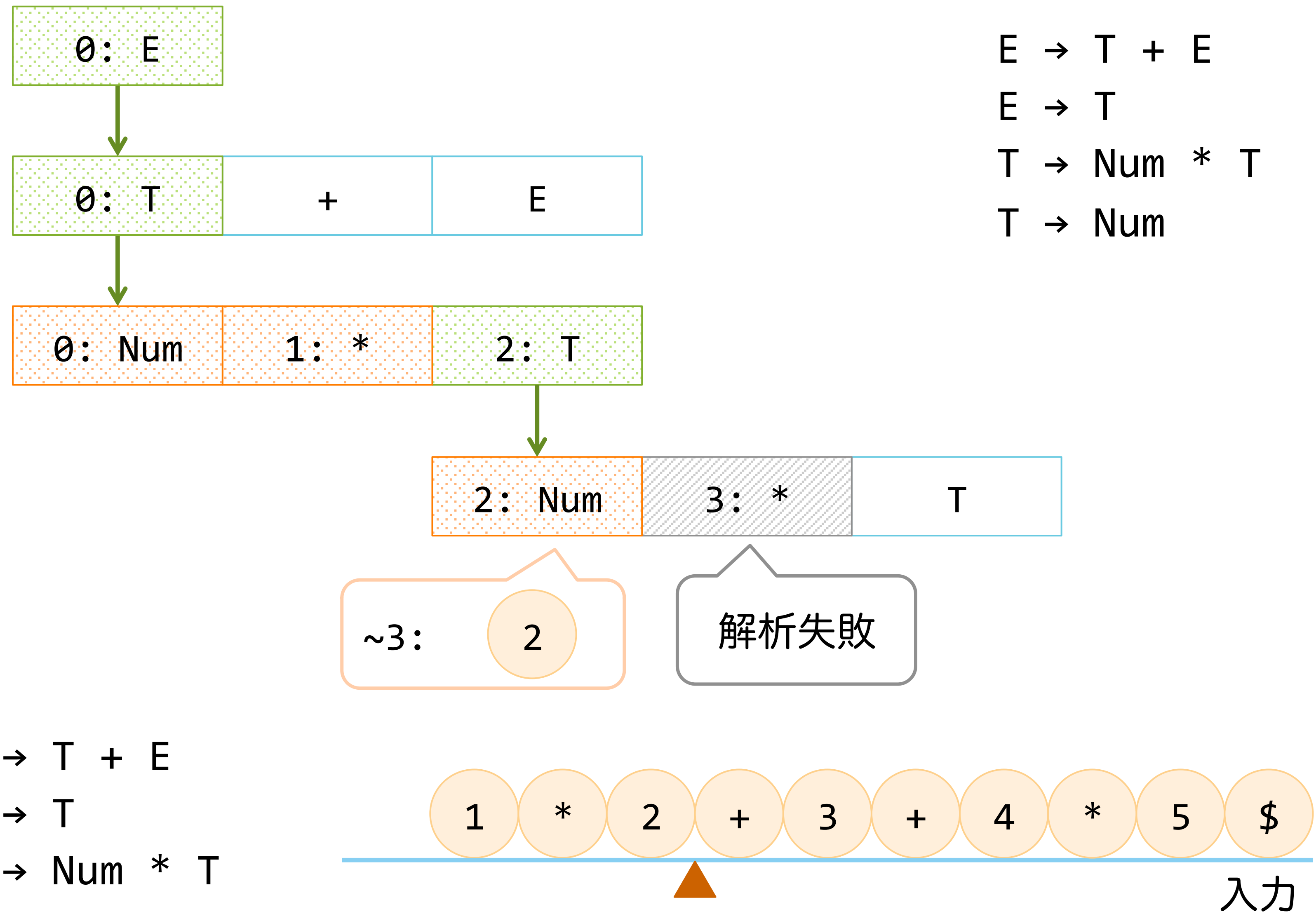
文法

E → T + E

E → T

T → Num * T

T → Num



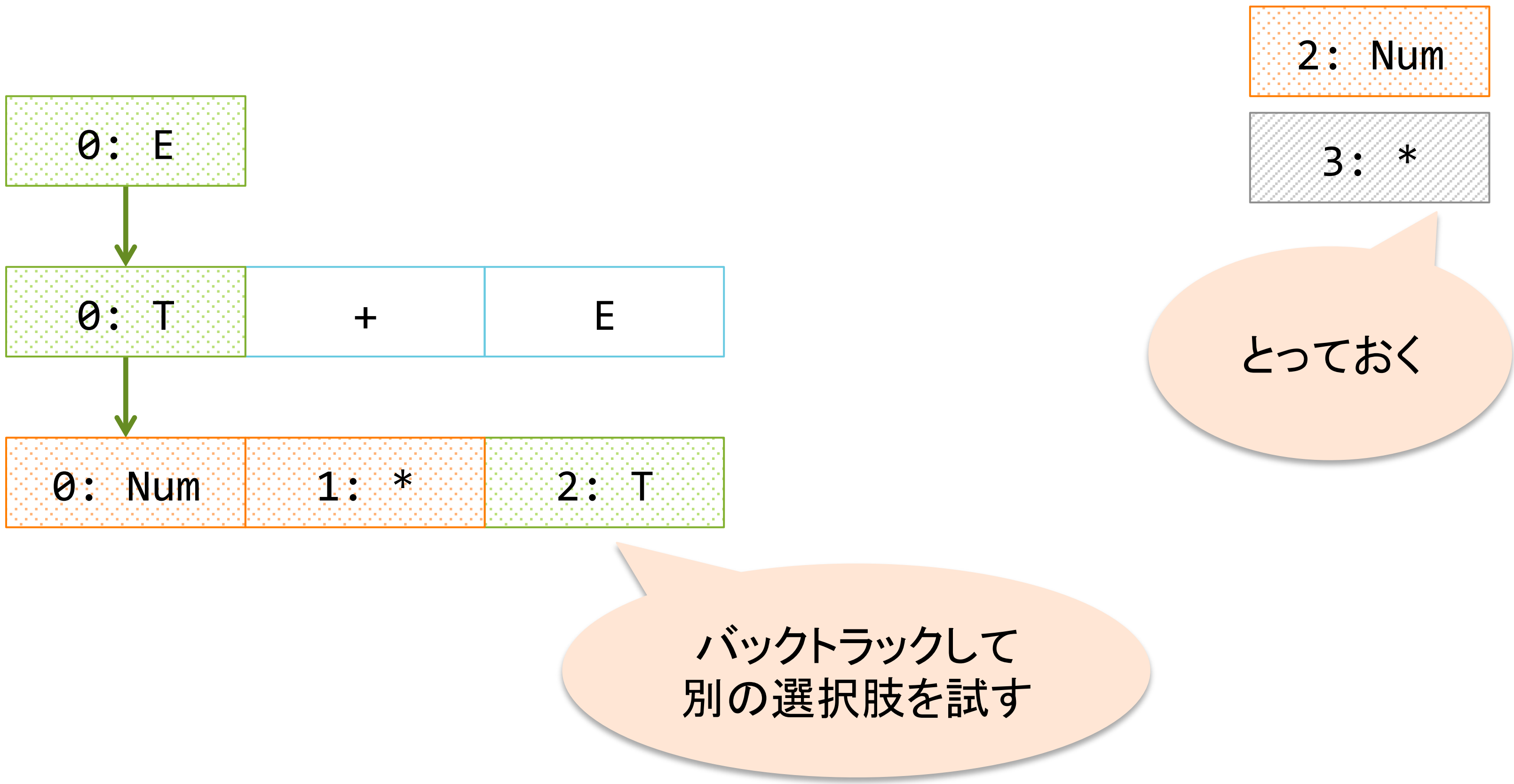
文法

E → T + E

E → T

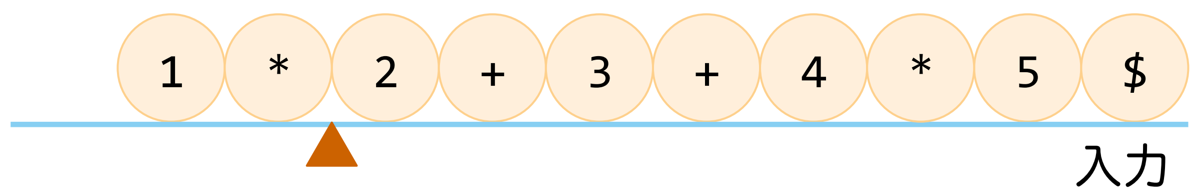
T → Num * T

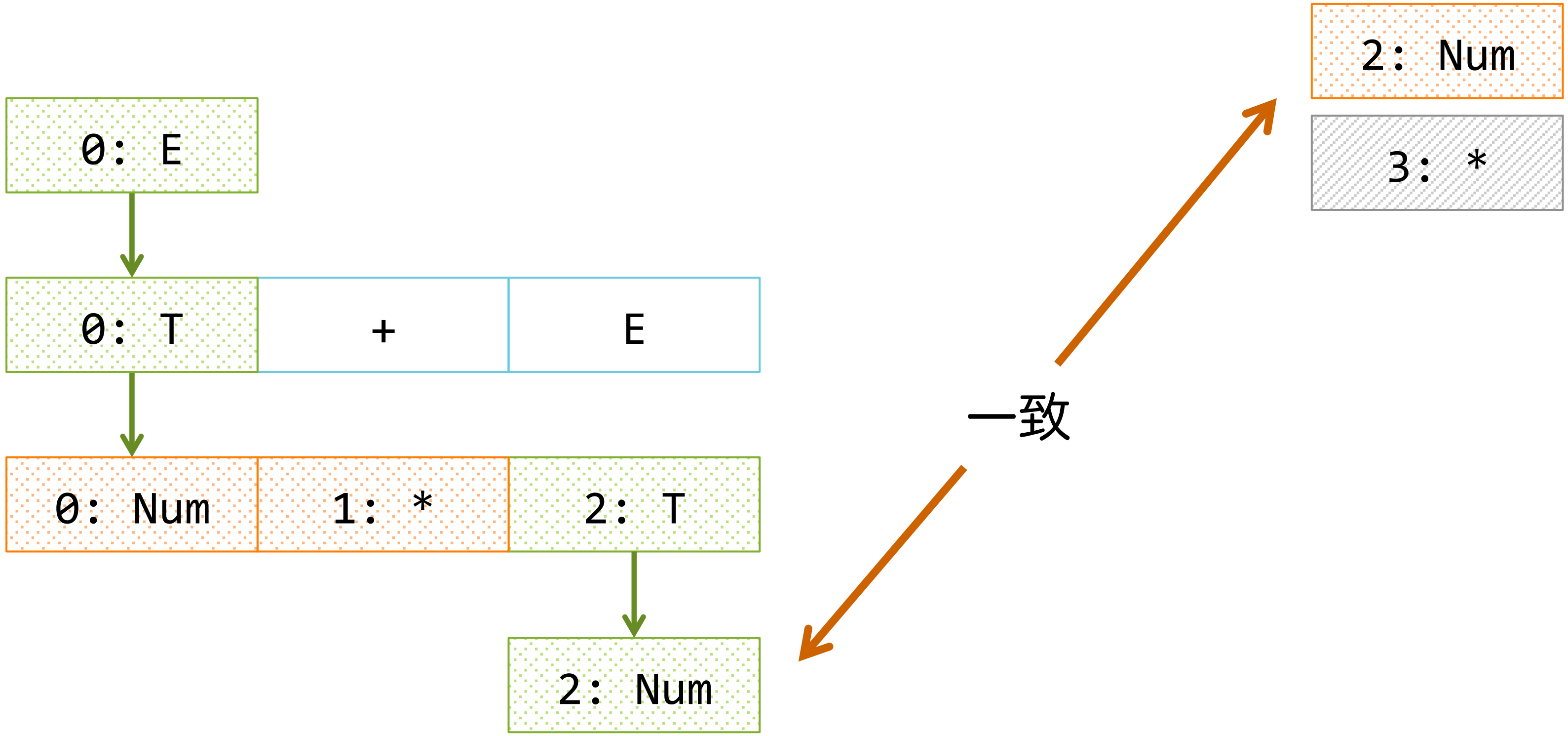
T → Num



文法

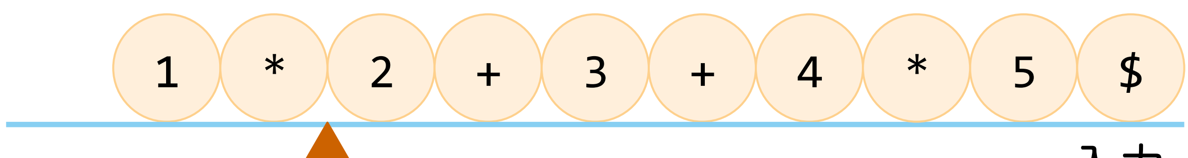
E → T + E
E → T
T → Num * T
T → Num



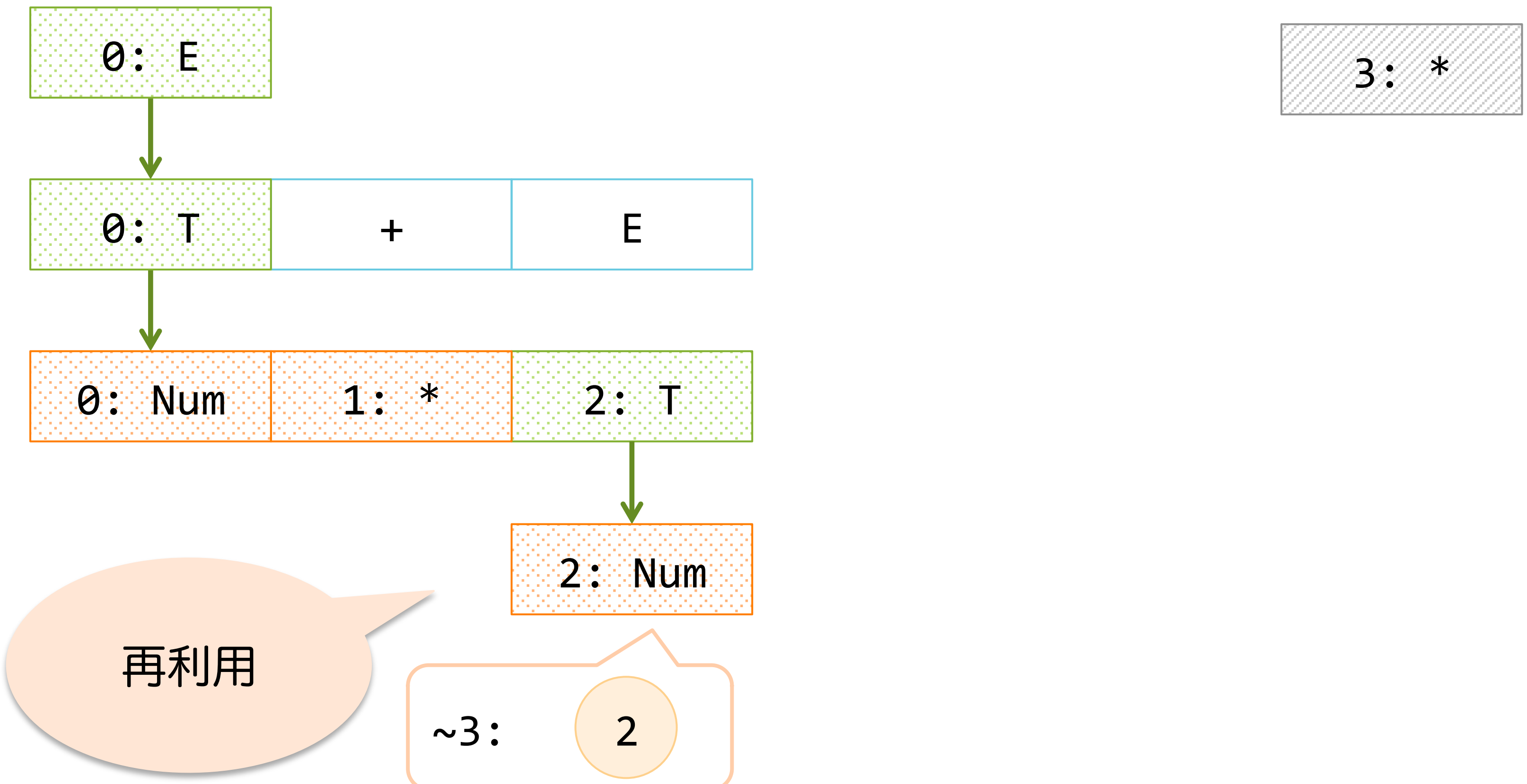


文法

E → T + E
E → T
T → Num * T
T → Num

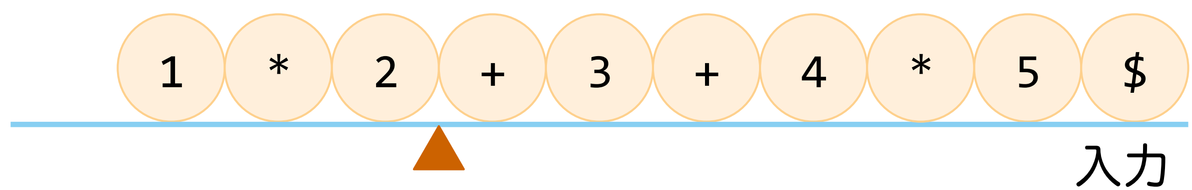


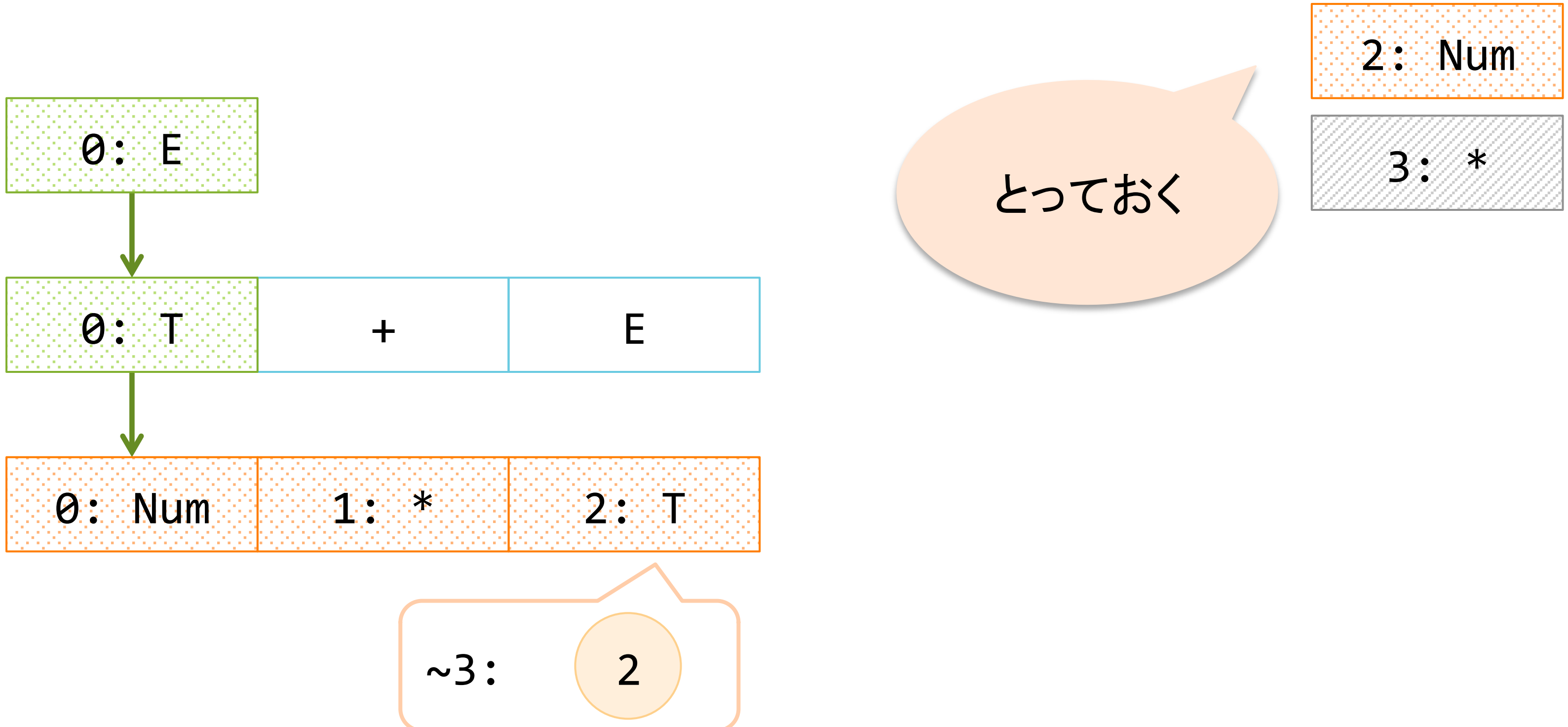
入力



文法

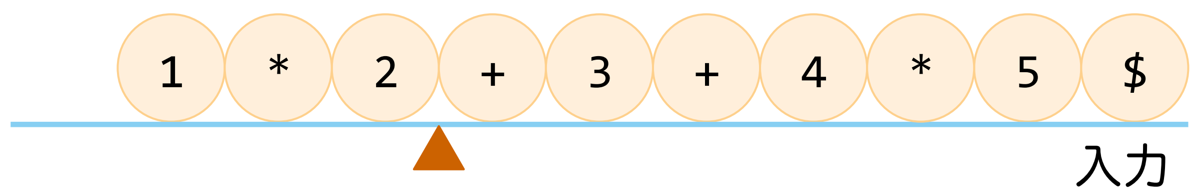
E → T + E
E → T
T → Num * T
T → Num

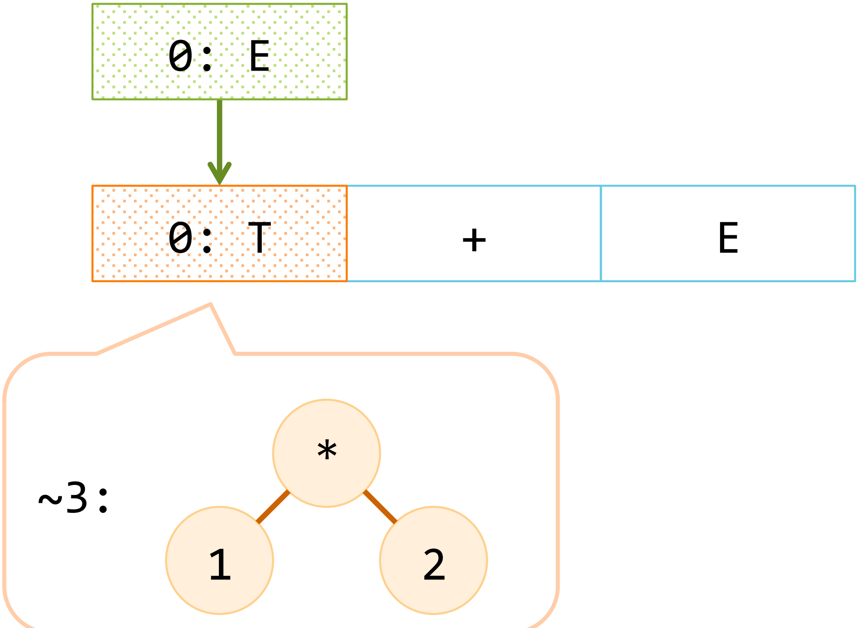


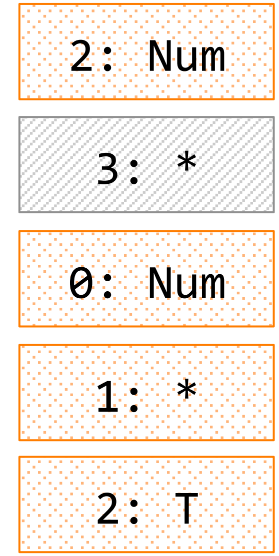


文法

E → T + E
E → T
T → Num * T
T → Num

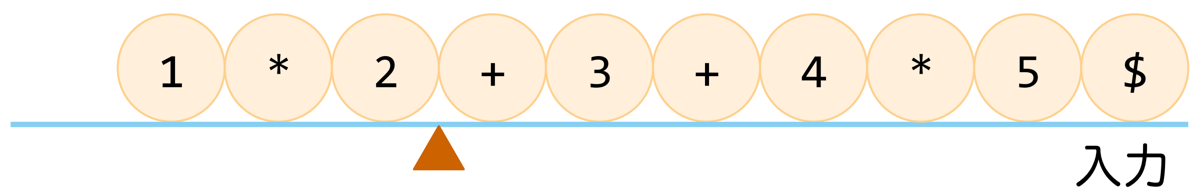


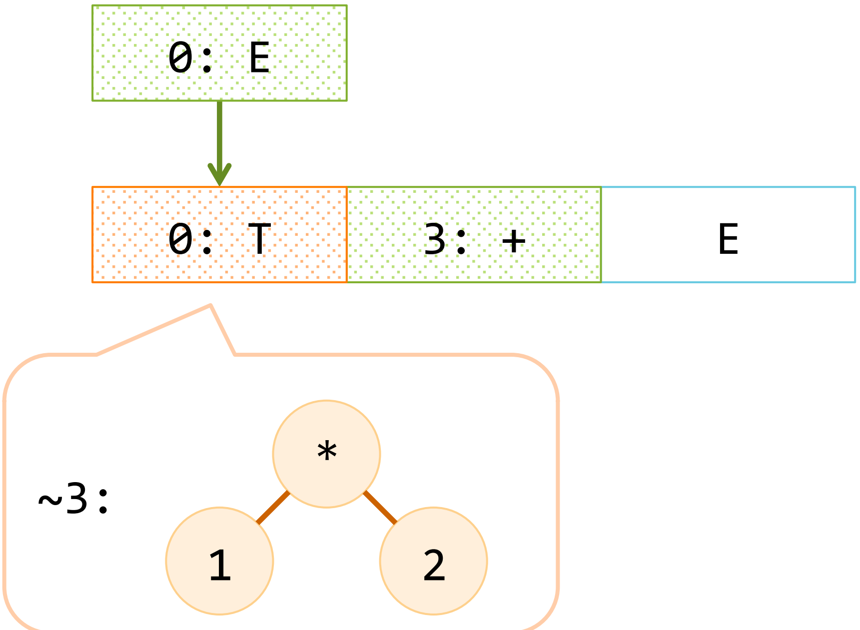


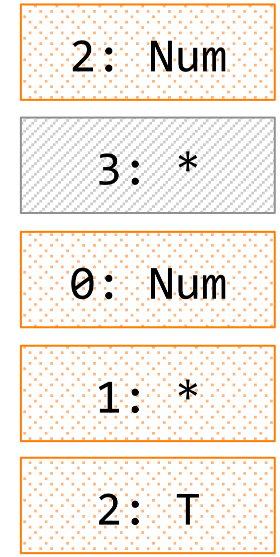


文法

E → T + E
E → T
T → Num * T
T → Num

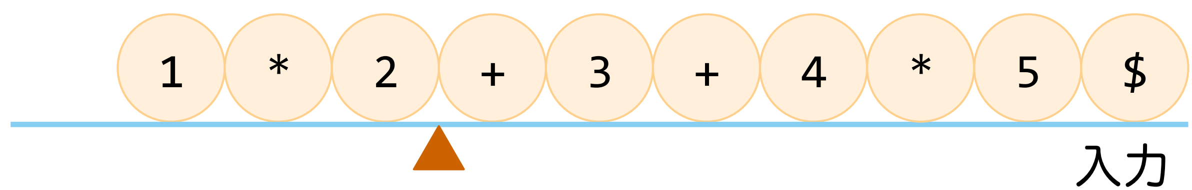


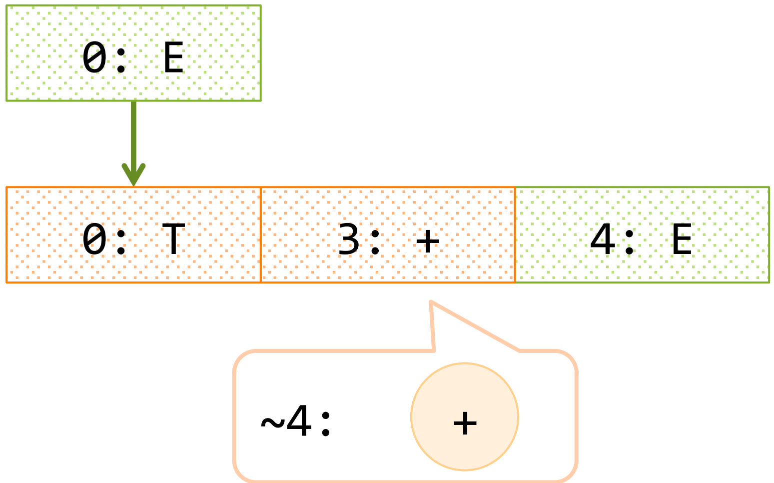


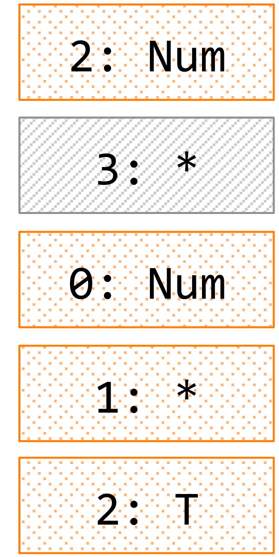


文法

E → T + E
E → T
T → Num * T
T → Num







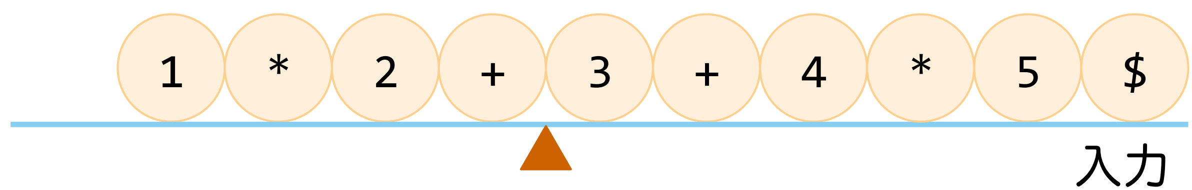
文法

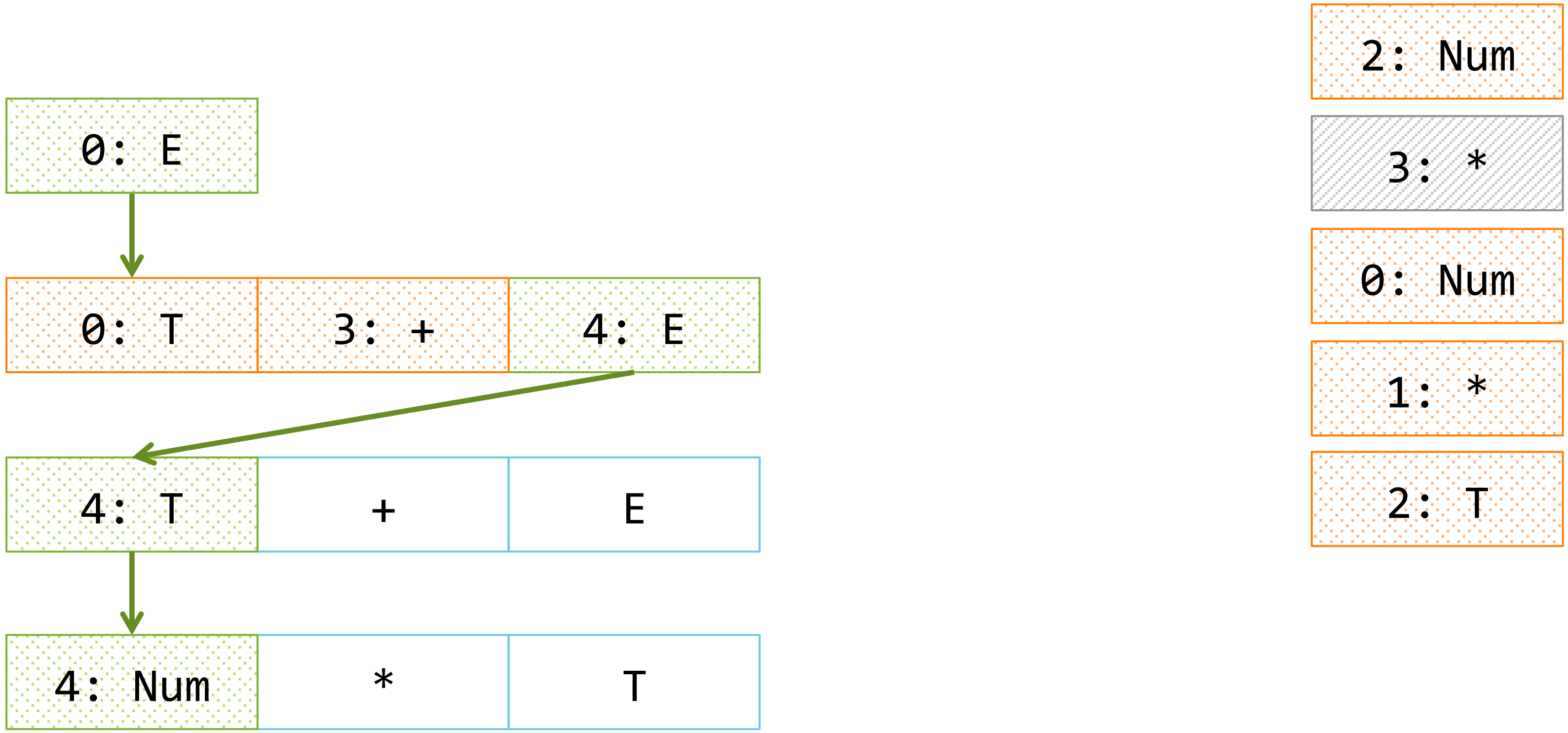
E → T + E

E → T

T → Num * T

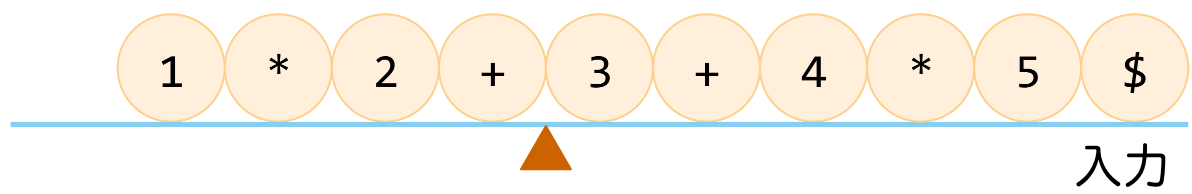
T → Num

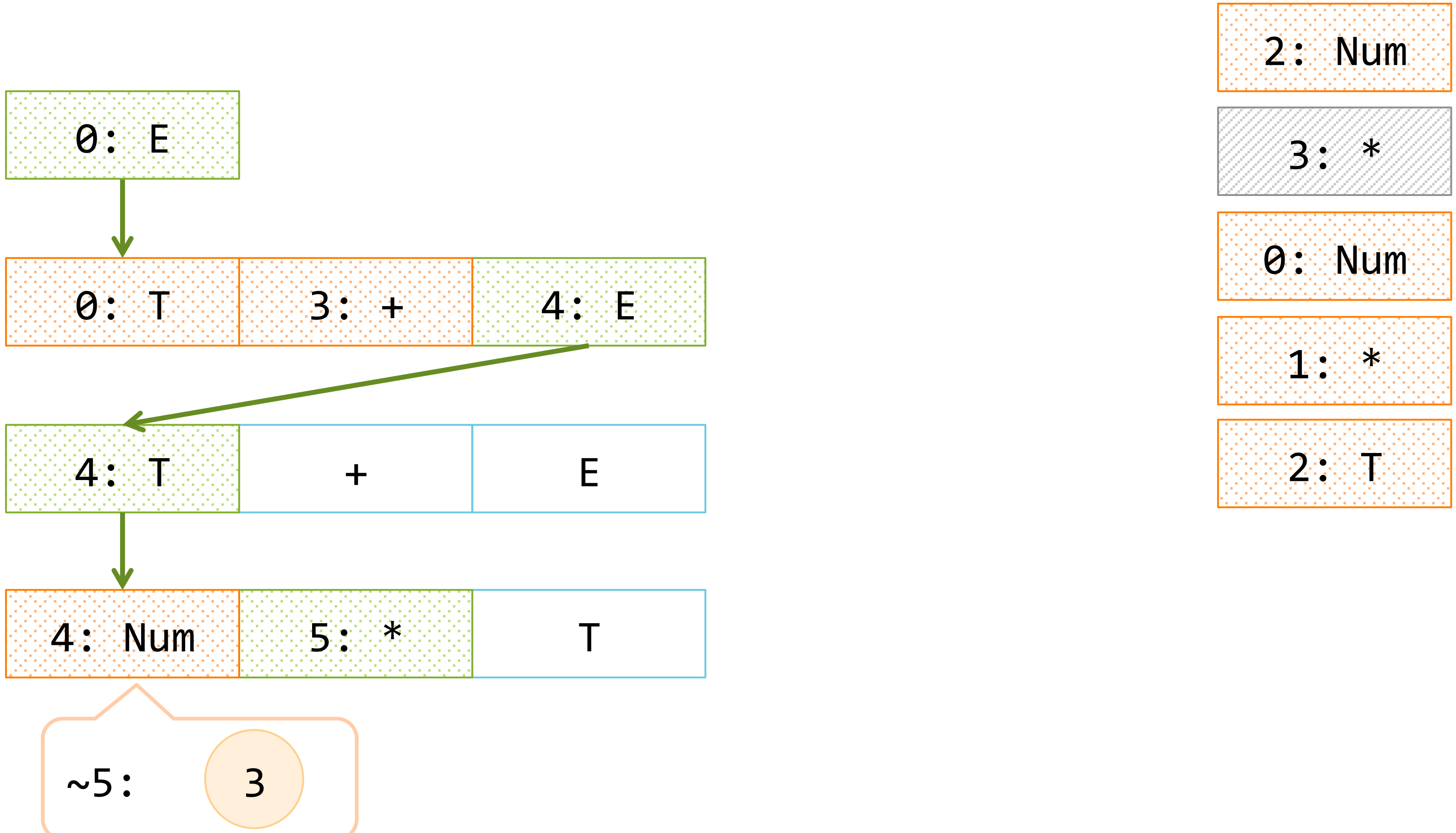




文法

E → T + E
E → T
T → Num * T
T → Num





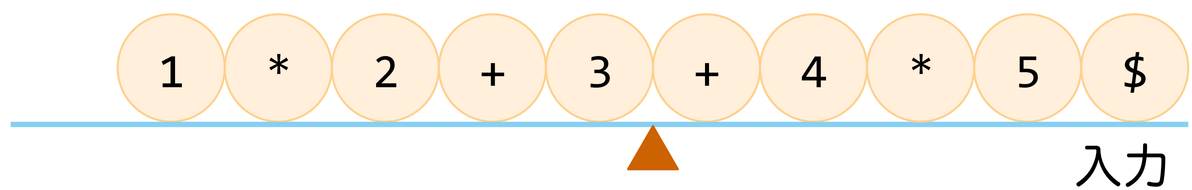
文法

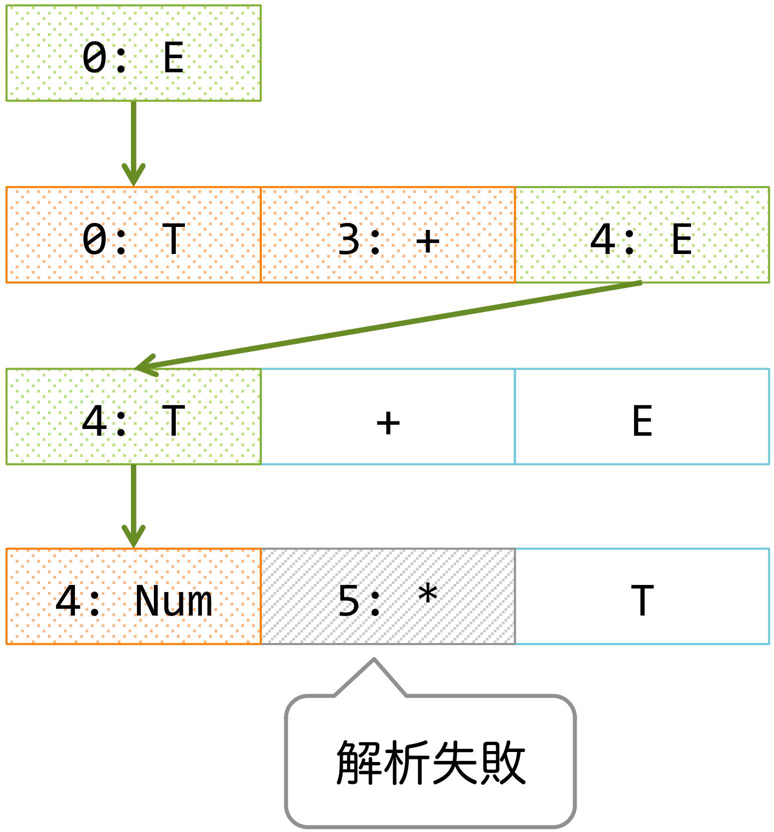
E → T + E

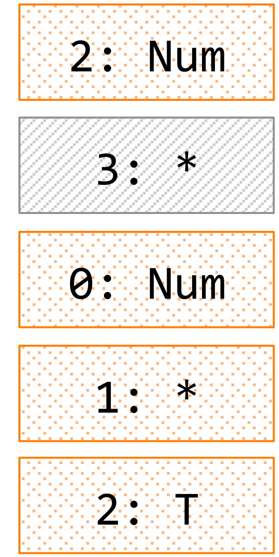
E → T

T → Num * T

T → Num

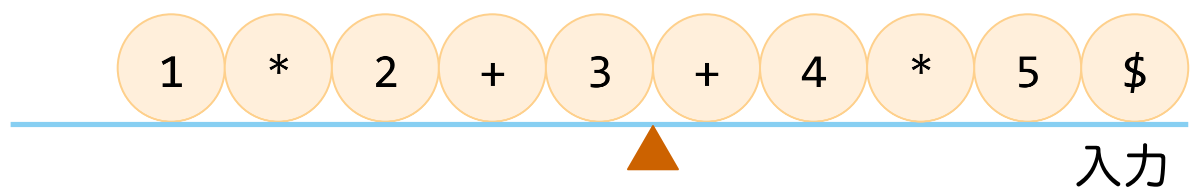


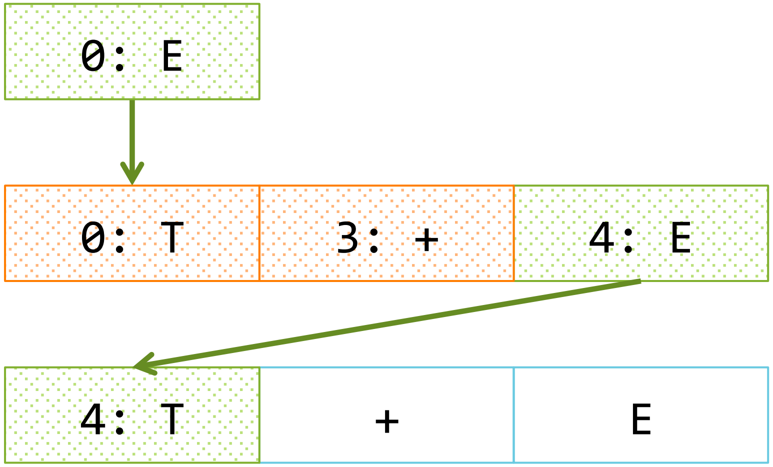


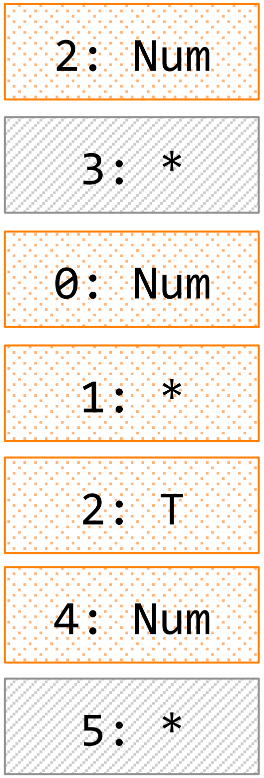


文法

E → T + E
E → T
T → Num * T
T → Num

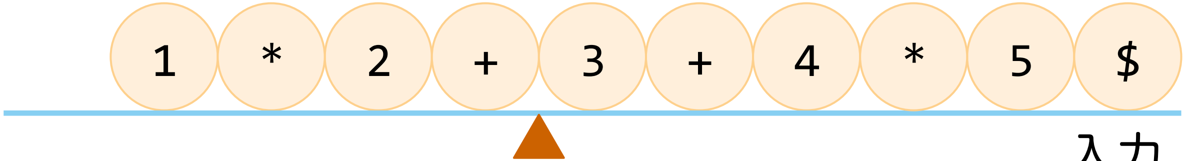




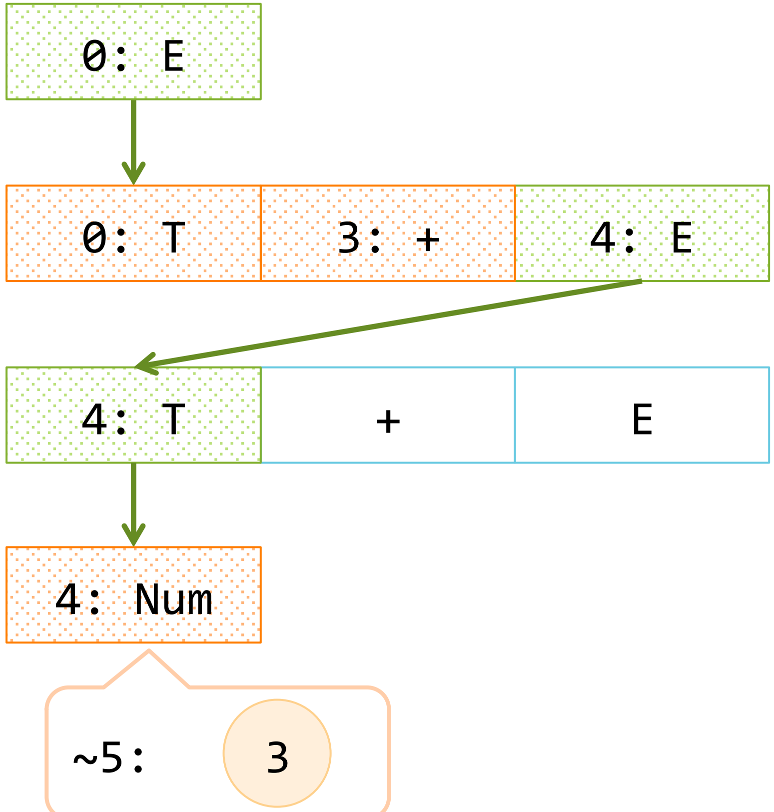


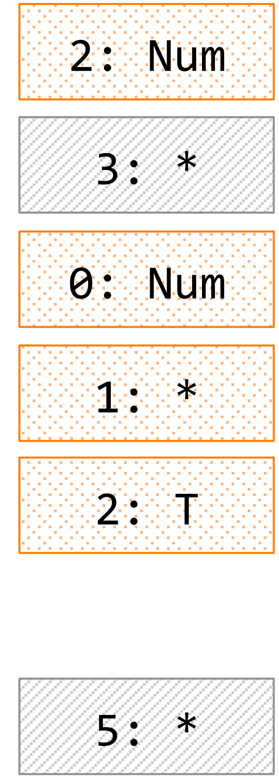
文法

E → T + E
E → T
T → Num * T
T → Num



入力





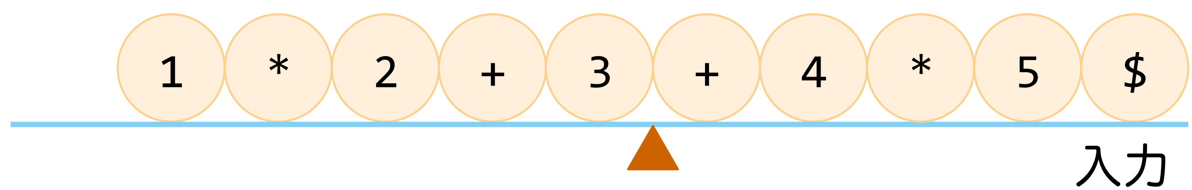
文法

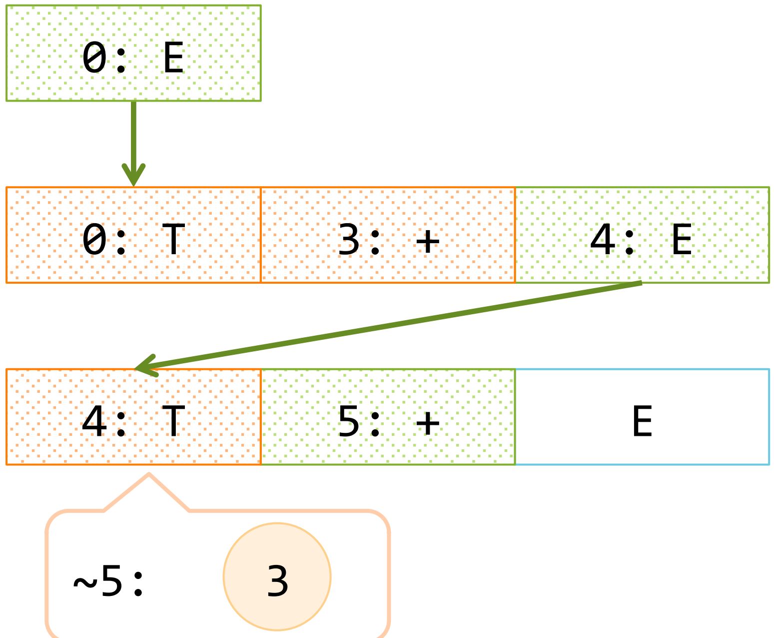
E → T + E

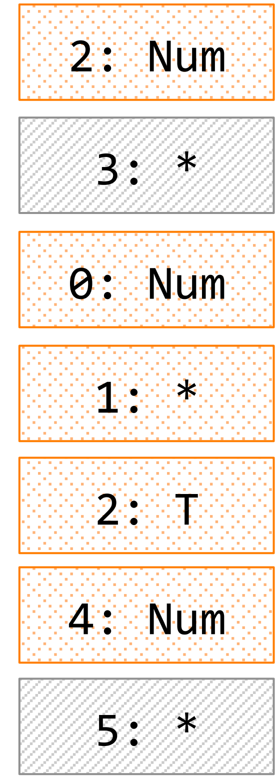
E → T

T → Num * T

T → Num

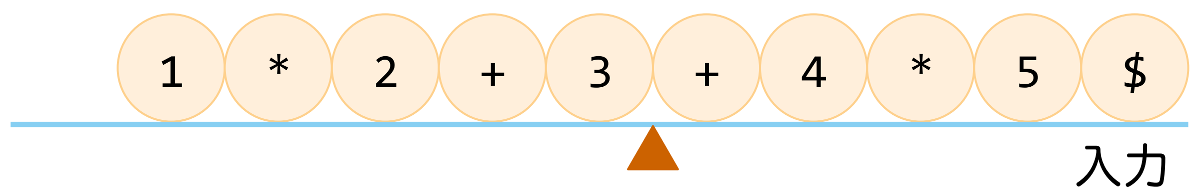


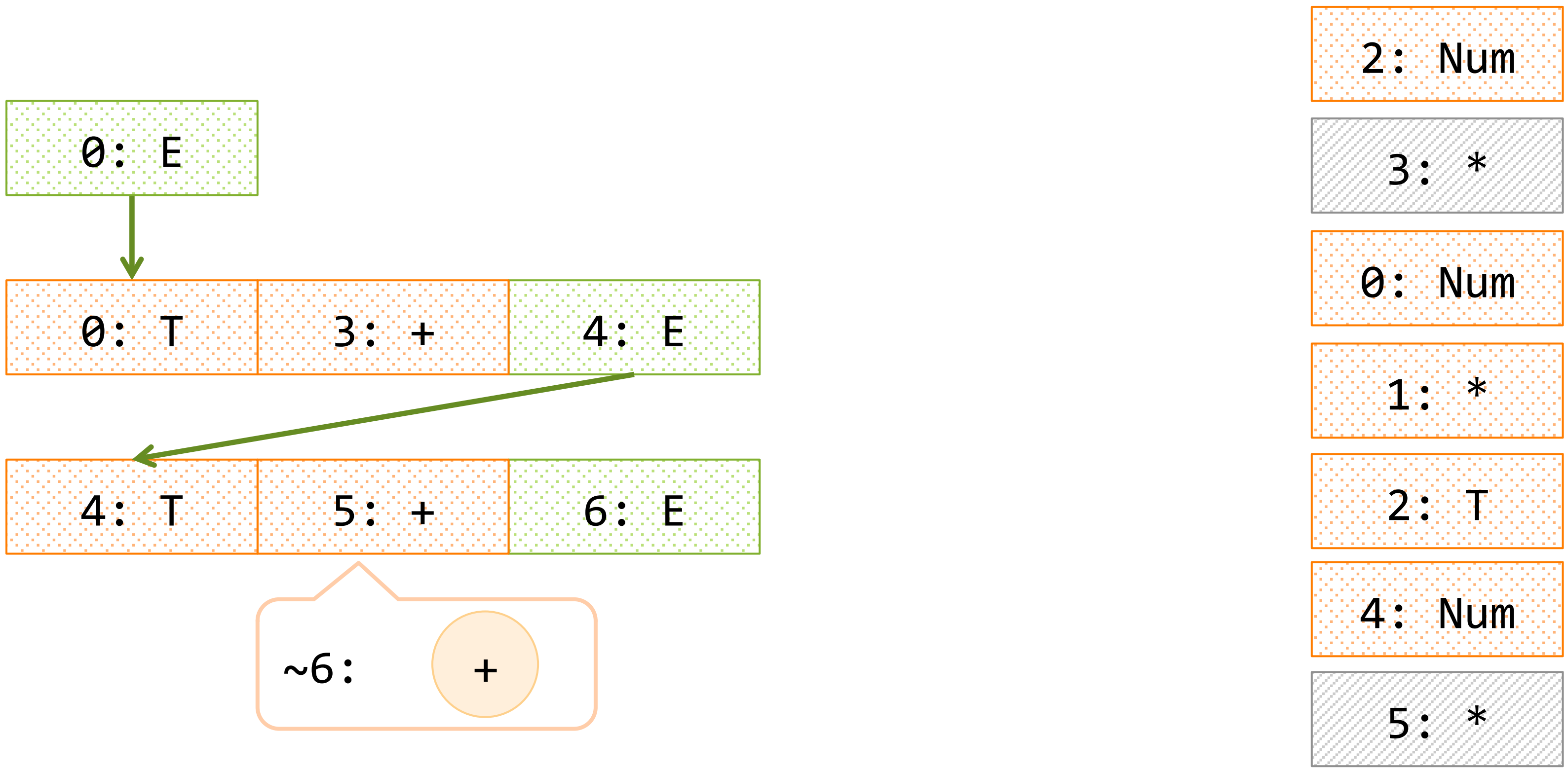




文法

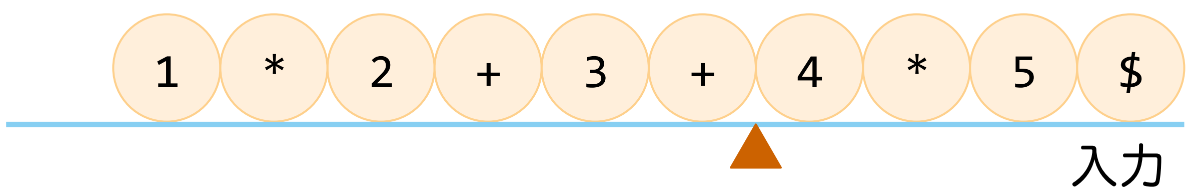
E → T + E
E → T
T → Num * T
T → Num

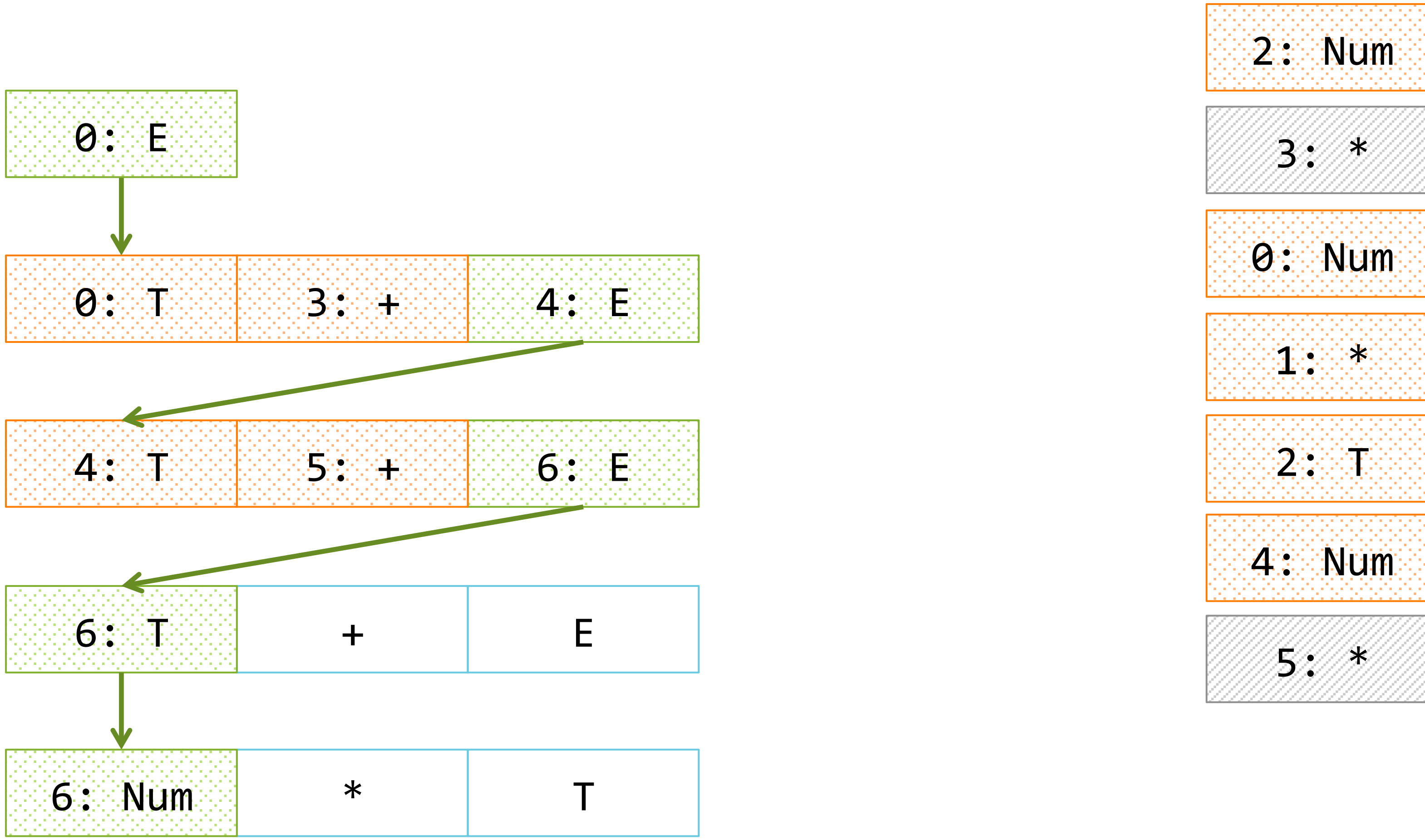




文法

E → T + E
E → T
T → Num * T
T → Num





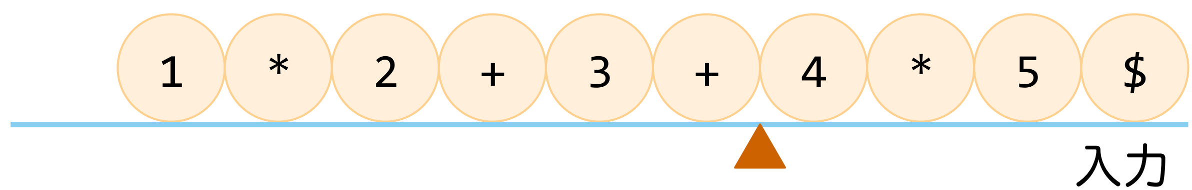
文法

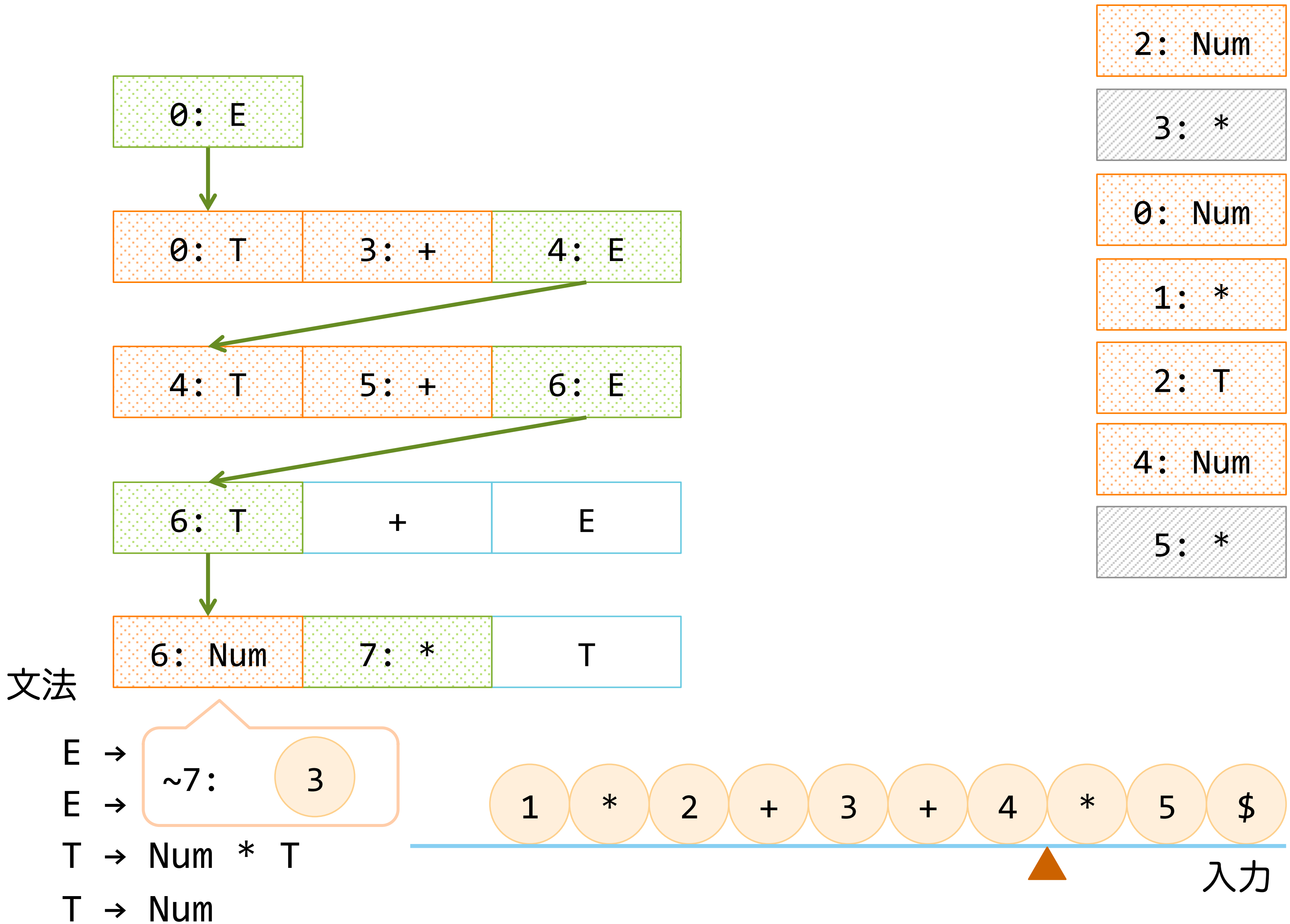
E → T + E

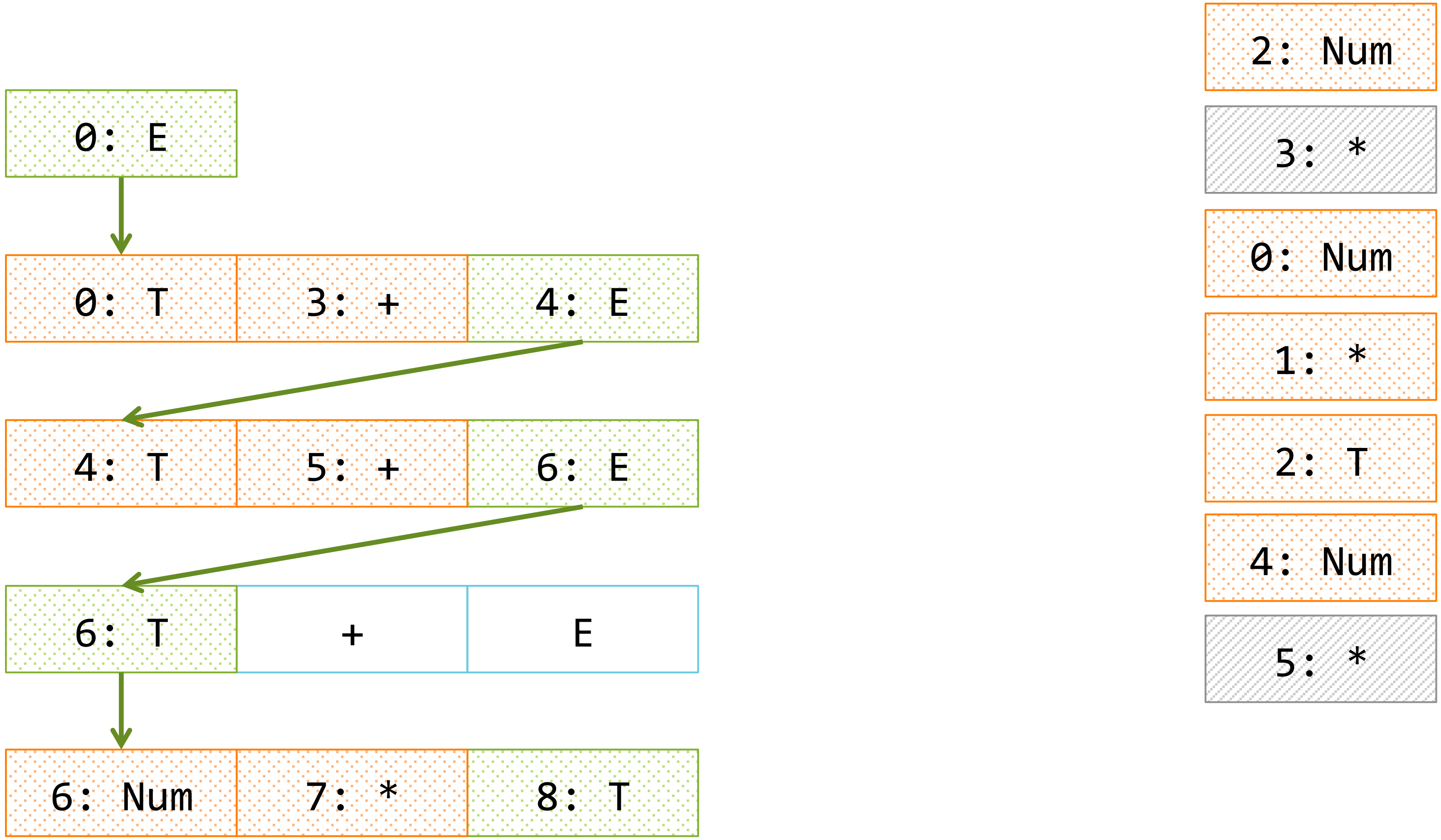
E → T

T → Num * T

T → Num

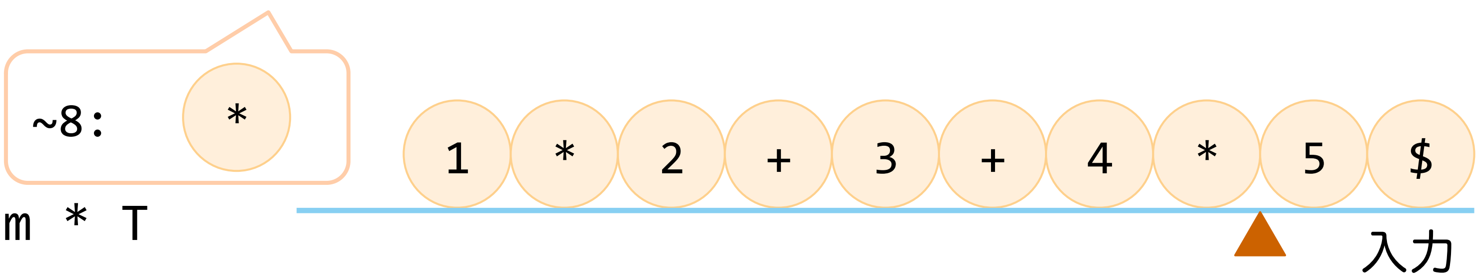


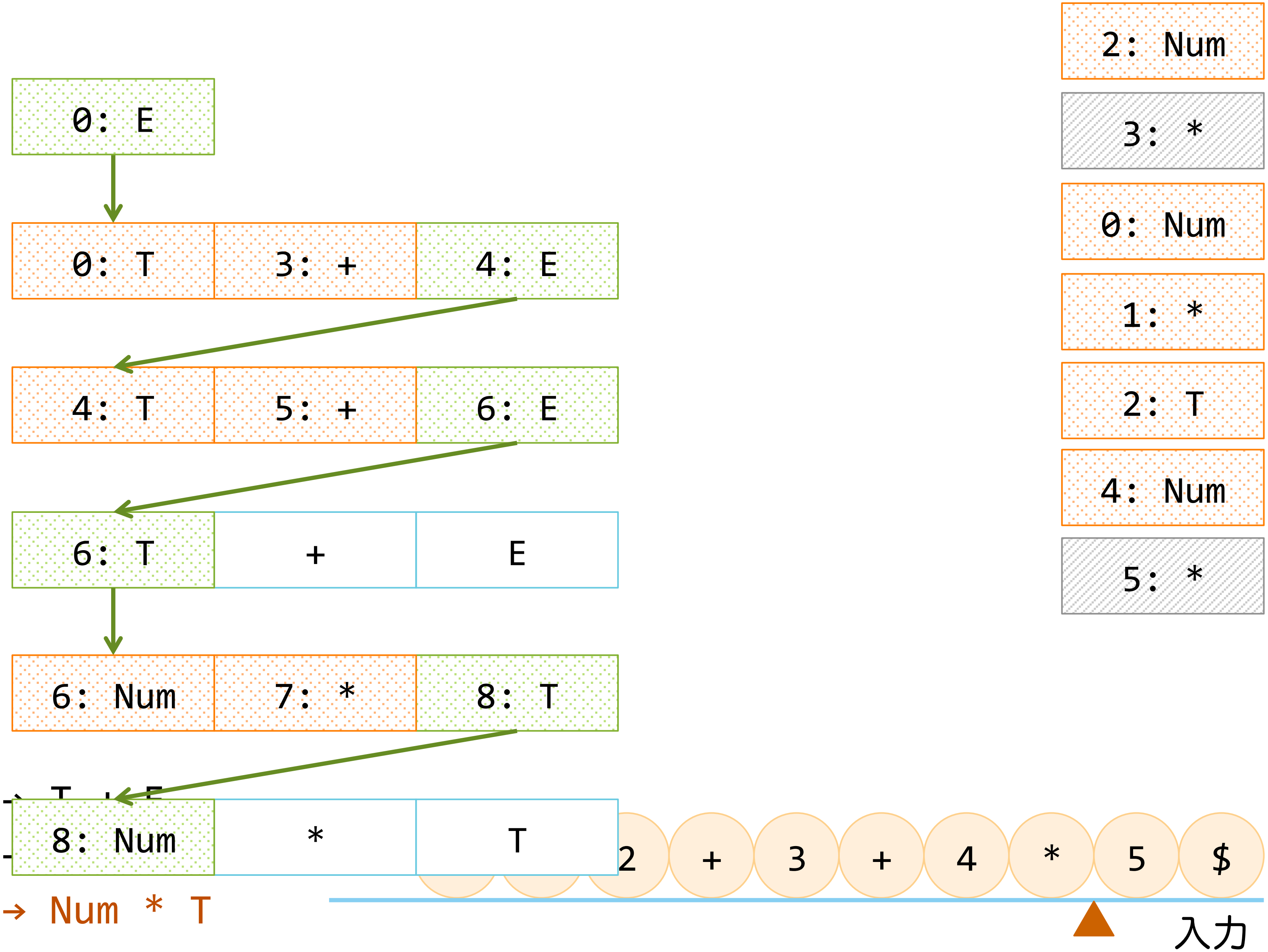

0: E
0: T
3: +
4: E
4: T
5: +
6: E
6: T
+
E
6: Num
7: *
T
2: Num
3: *
0: Num
1: *
2: T
4: Num
5: *
文法
E →
E →
T → Num * T
T → Num
~7:
3
1
*
2
+
3
+
4
*
5
$
入力



文法

E → T
E → T
T → Num * T
T → Num





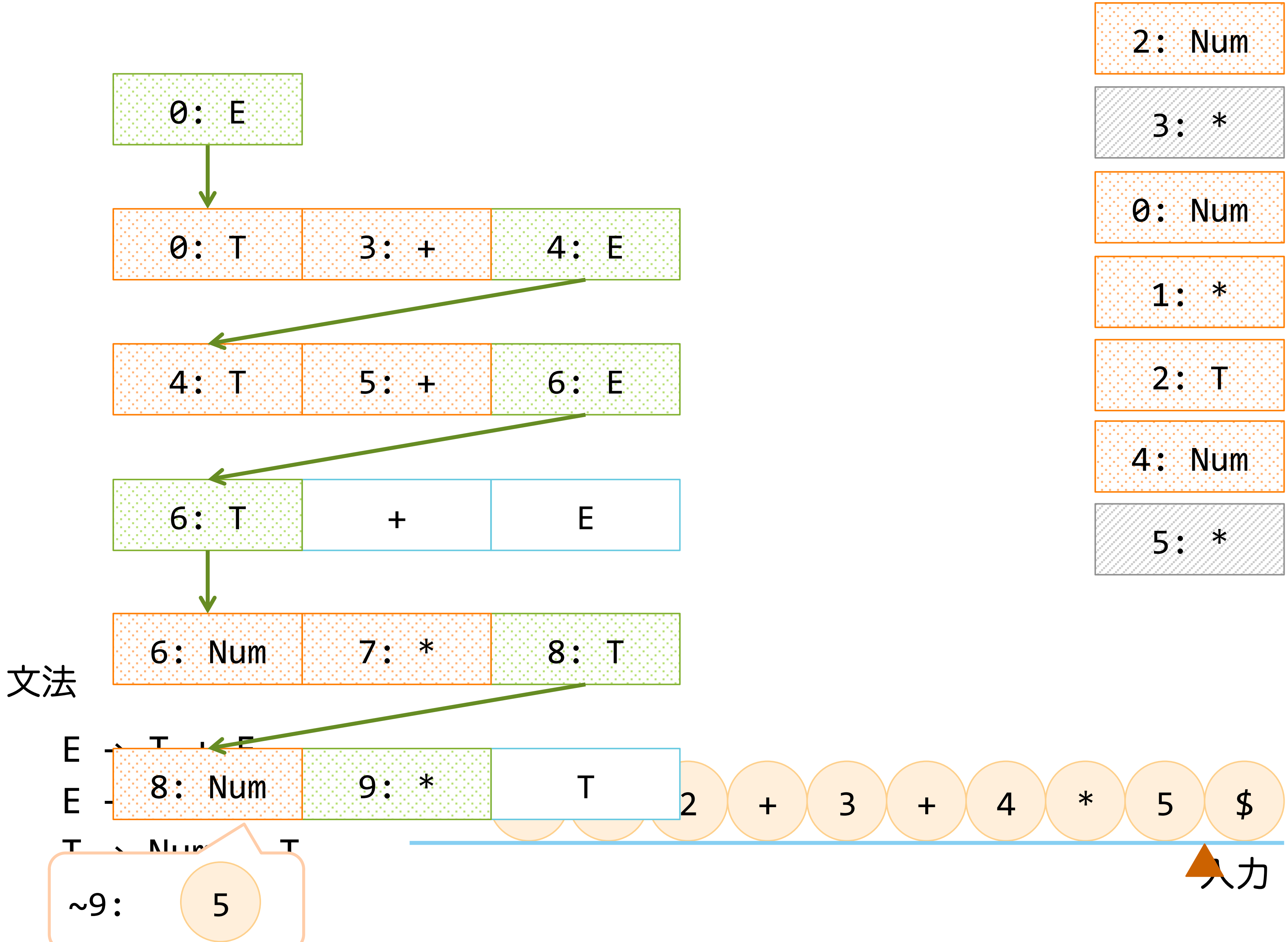
文法

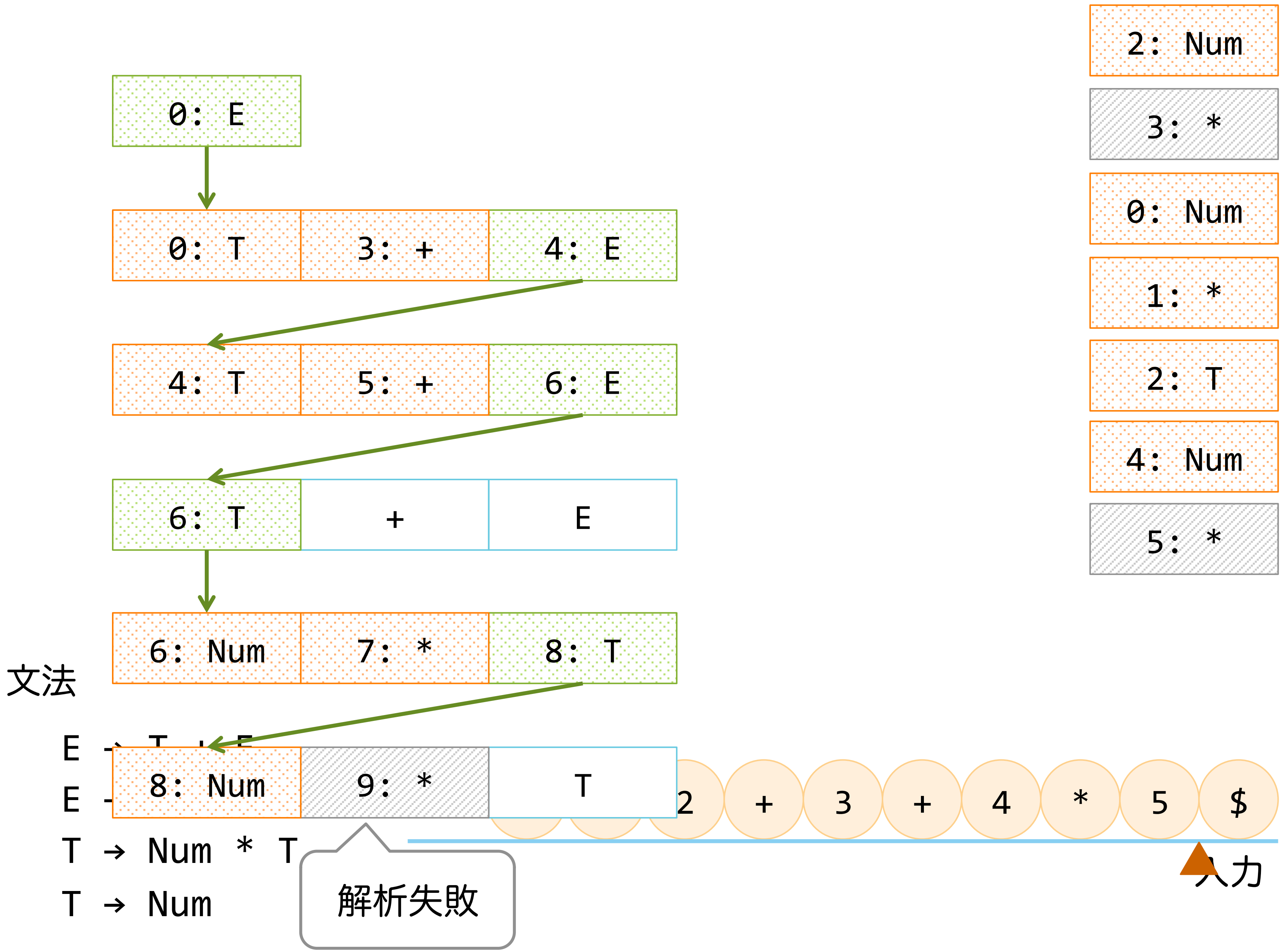
E → T + E

E → T

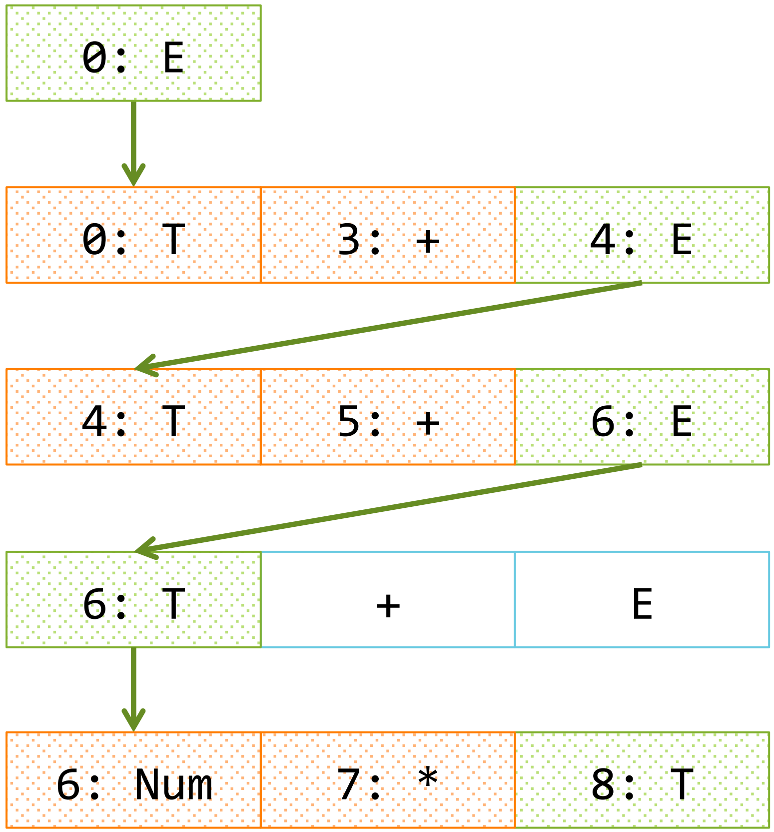
T → Num * T

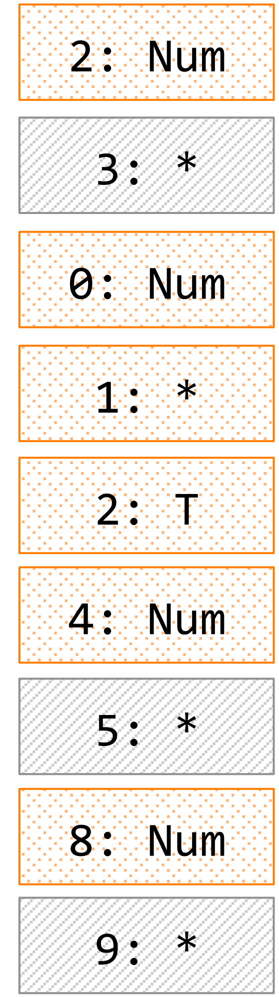
T → Num


0: E
0: T
3: +
4: E
4: T
5: +
6: E
6: T
+
E
6: Num
7: *
8: T
文法
8: Num
9: *
T
~9:
5
2
+
3
+
4
*
5
$
入力
2: Num
3: *
0: Num
1: *
2: T
4: Num
5: *



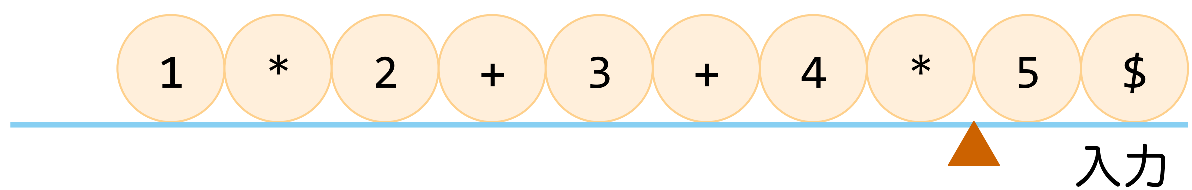
0: E
0: T
3: +
4: E
4: T
5: +
6: E
6: T
+
E
6: Num
7: *
8: T
8: Num
9: *
T
解析失敗
2: Num
3: *
0: Num
1: *
2: T
4: Num
5: *
文法
T → Num * T
T → Num
2
+
3
+
4
*
5
$
入力

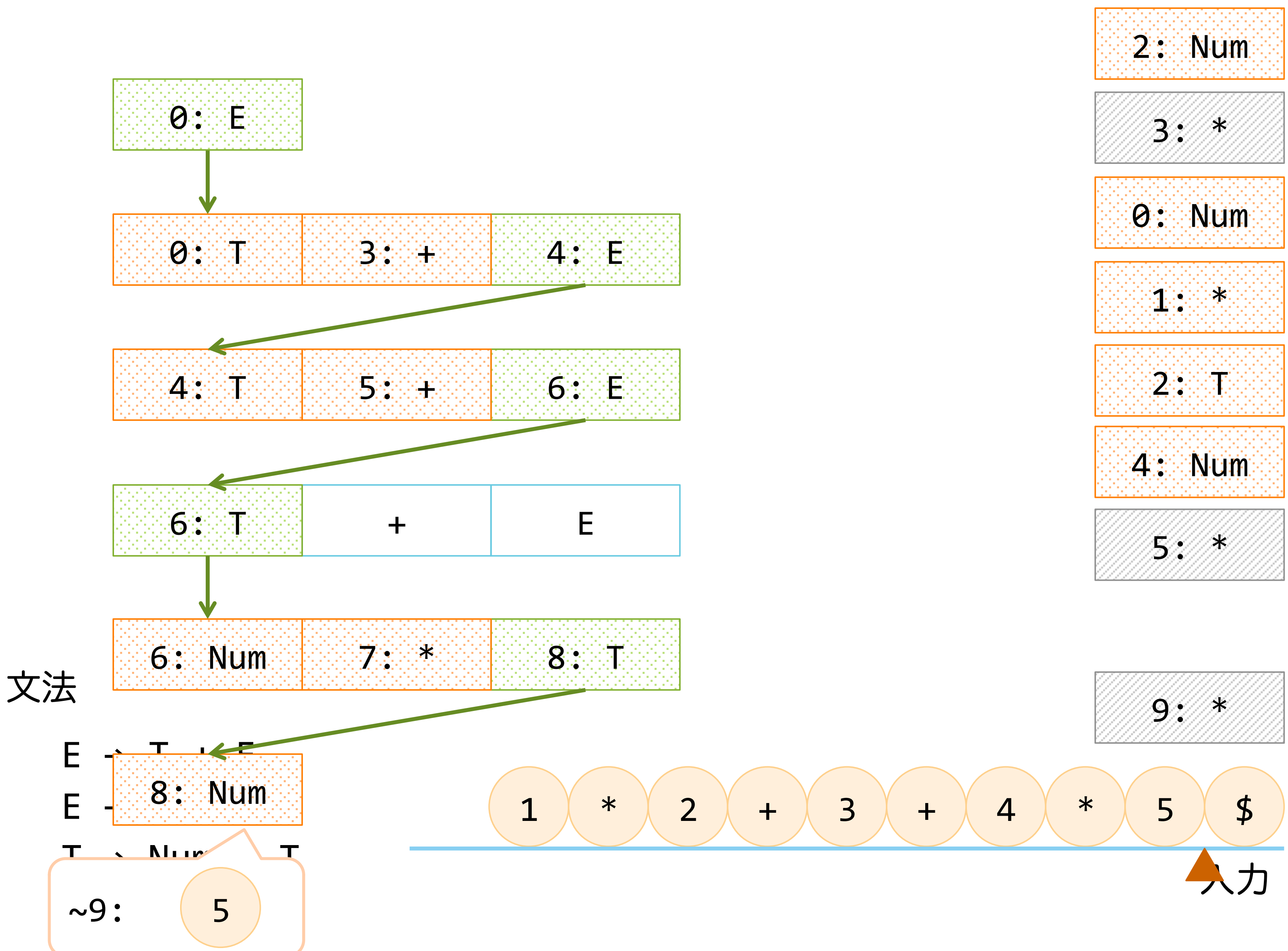




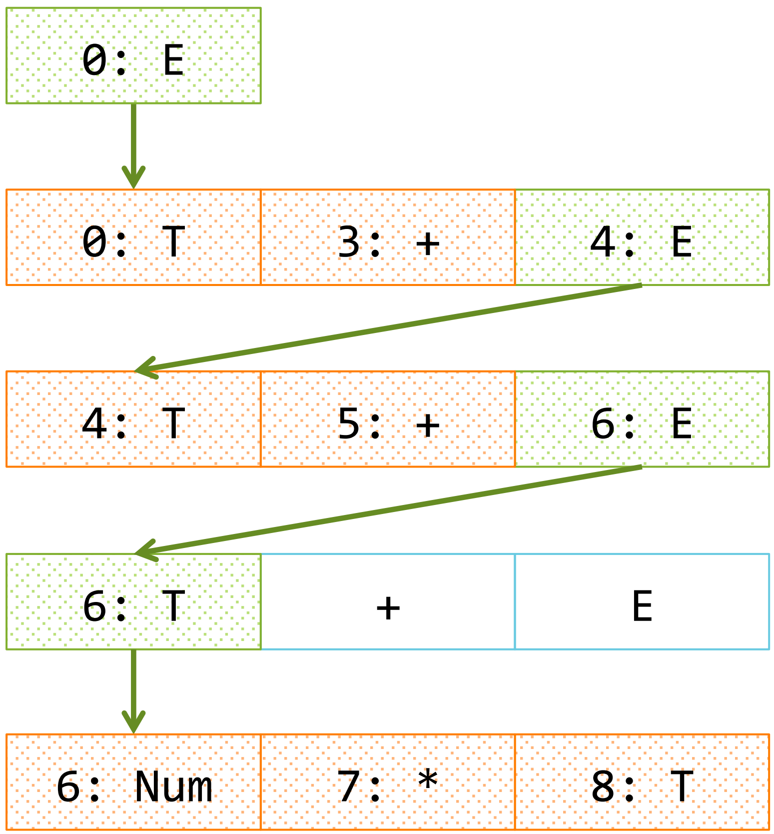
文法

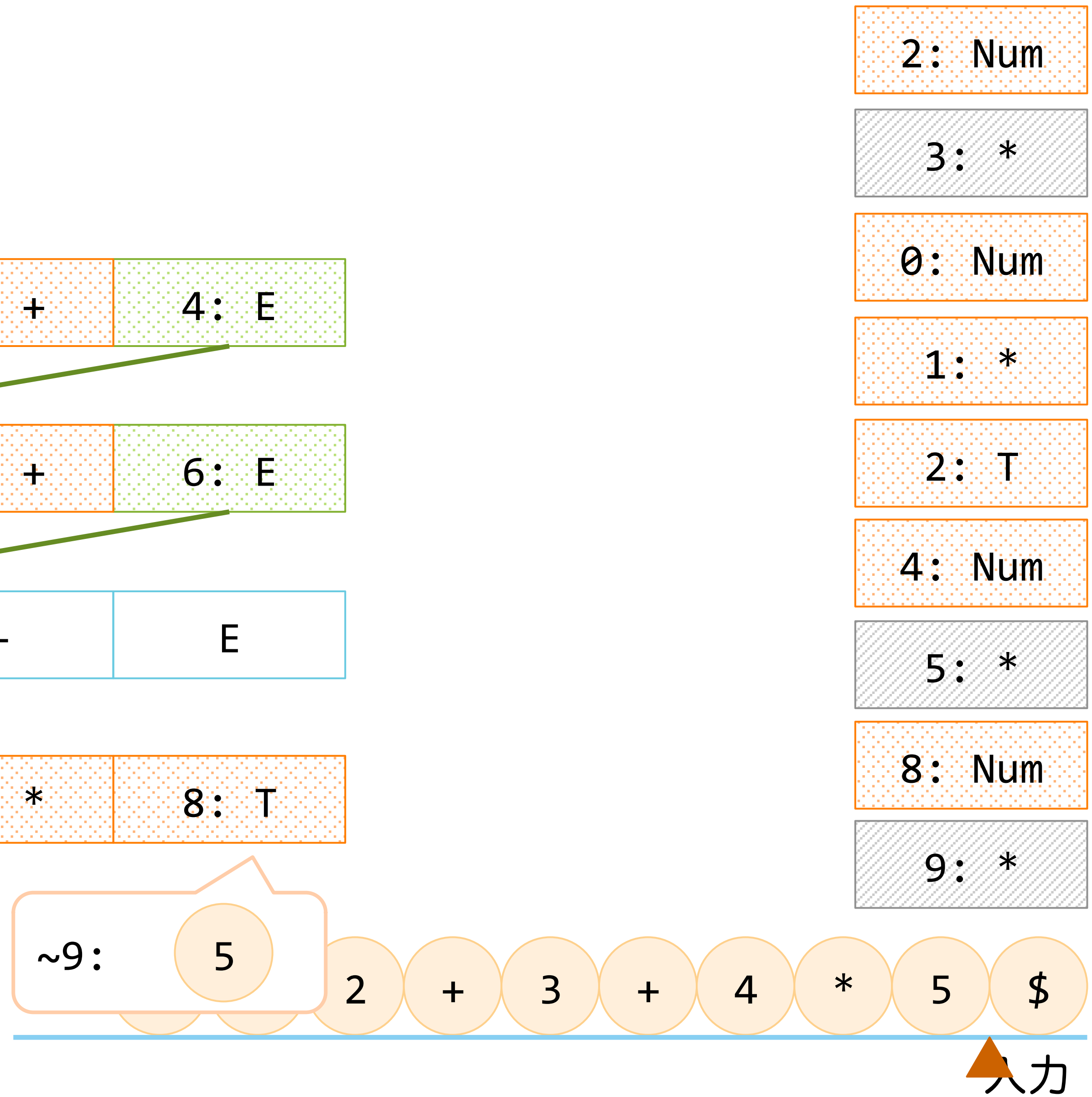
E → T + E
E → T
T → Num * T
T → Num





0: E
0: T
3: +
4: E
4: T
5: +
6: E
6: T
+
E
6: Num
7: *
8: T
8: Num
文法
~9:
5
2: Num
3: *
0: Num
1: *
2: T
4: Num
5: *
9: *
1
*
2
+
3
+
4
*
5
$
入力





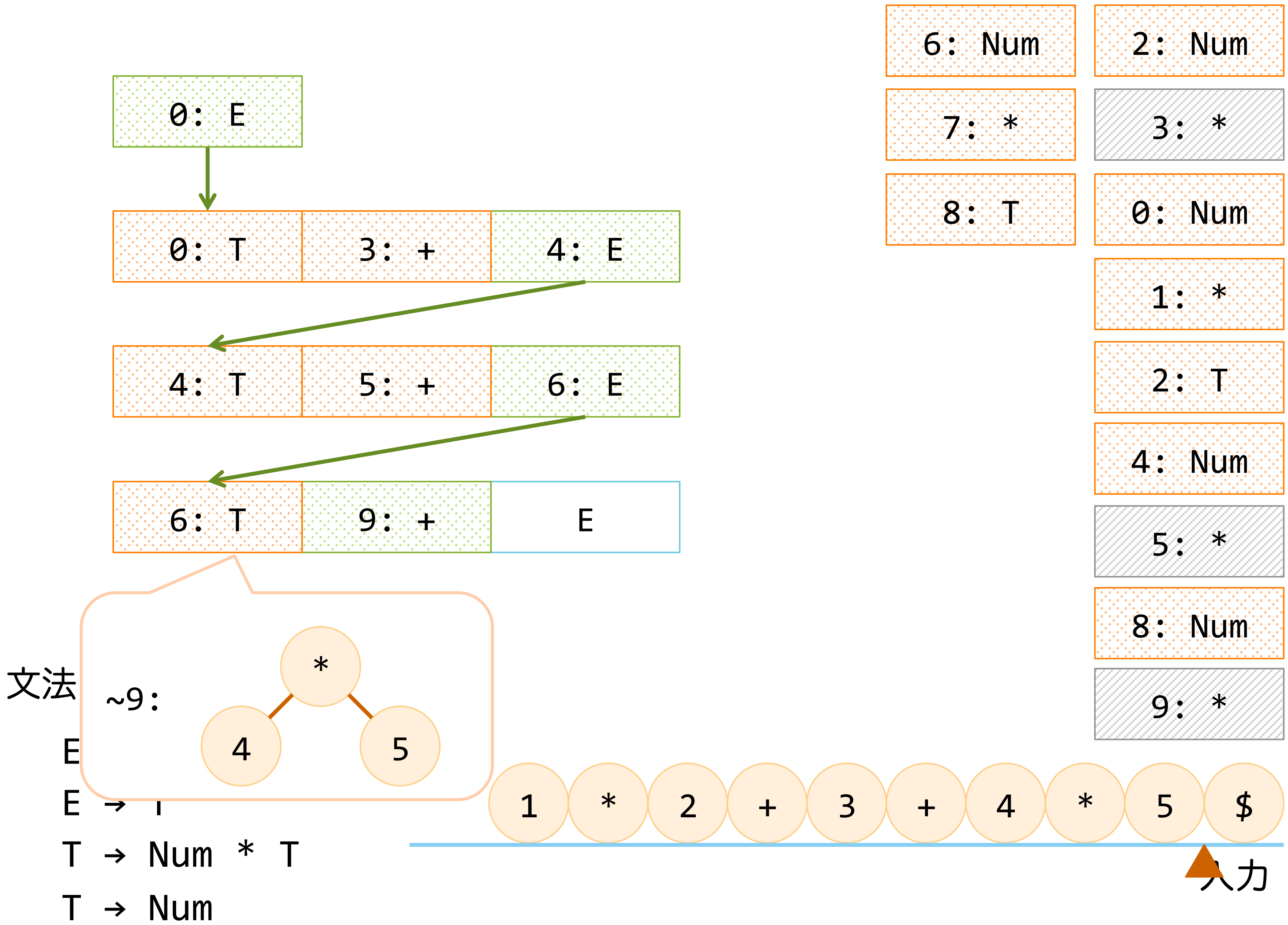
文法

E → T + E

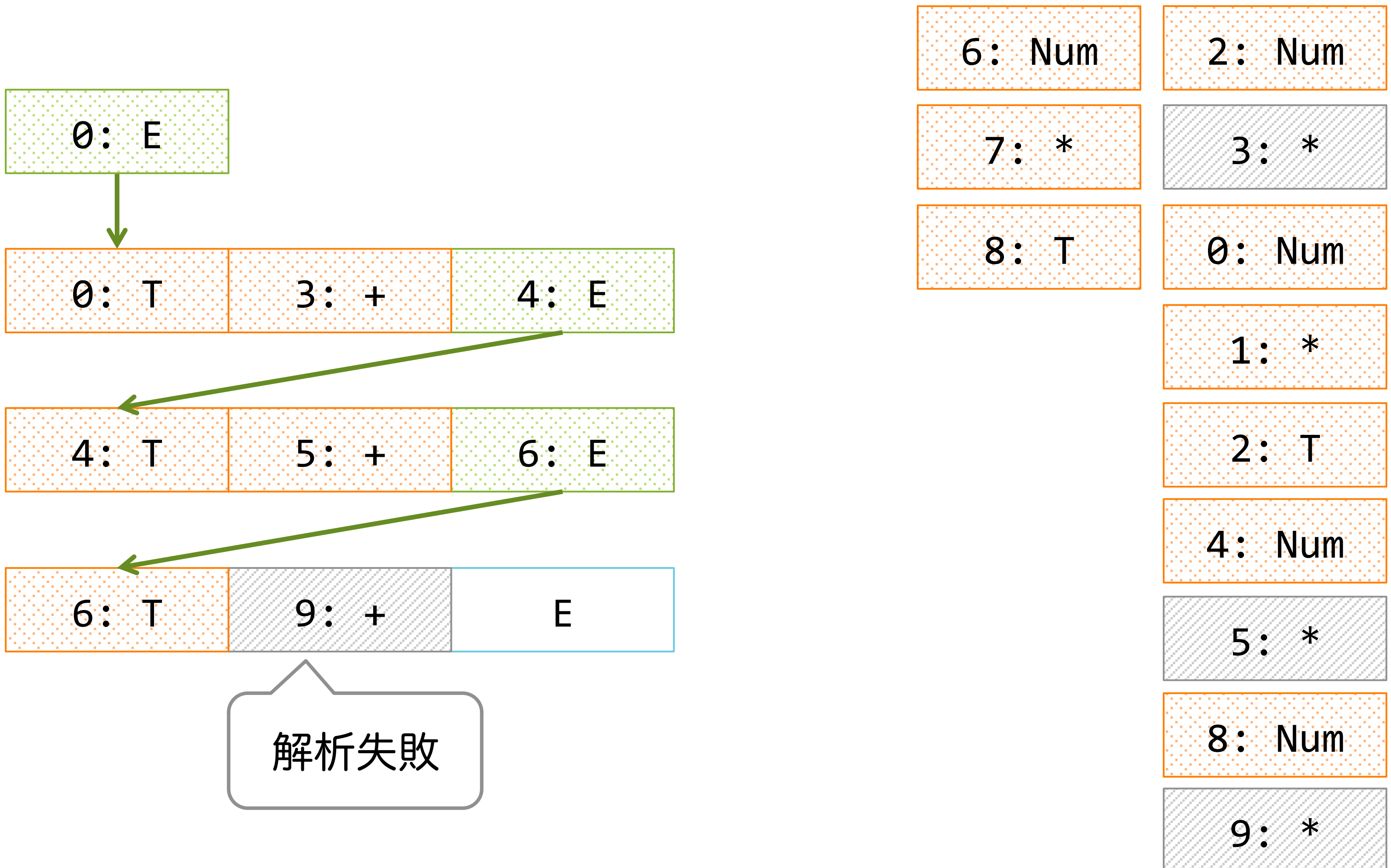
E → T

T → Num * T

T → Num



0: E
0: T
3: +
4: E
4: T
5: +
6: E
6: T
9: +
E
~9:
*
4
5
6: Num
7: *
8: T
2: Num
3: *
0: Num
1: *
2: T
4: Num
5: *
8: Num
9: *
1
*
2
+
3
+
4
*
5
$
入力
文法
E
E →
T → Num * T
T → Num



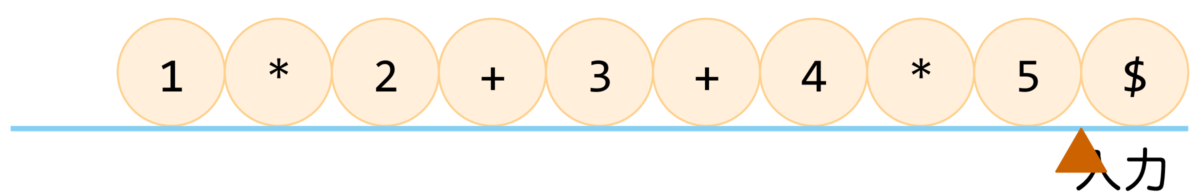
文法

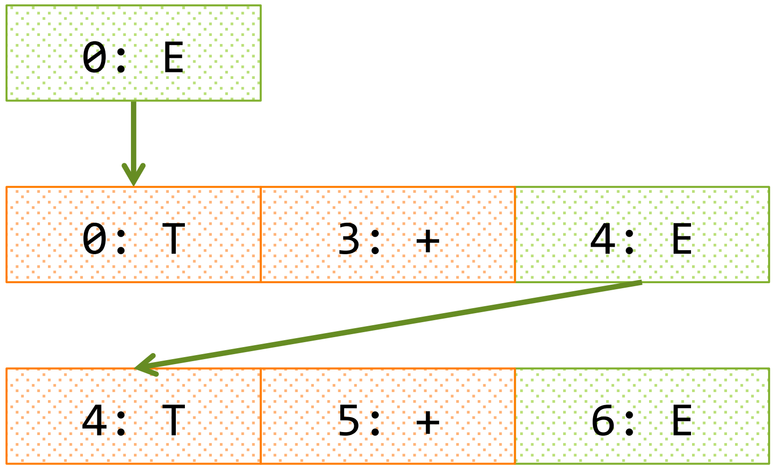
E → T + E

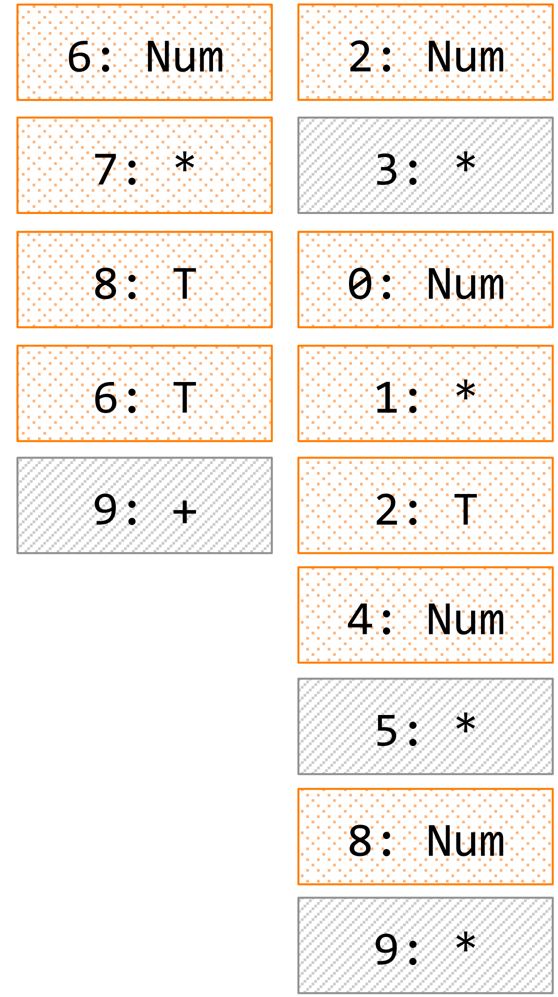
E → T

T → Num * T

T → Num







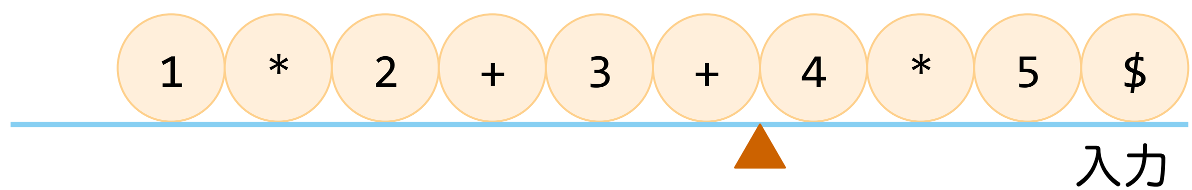
文法

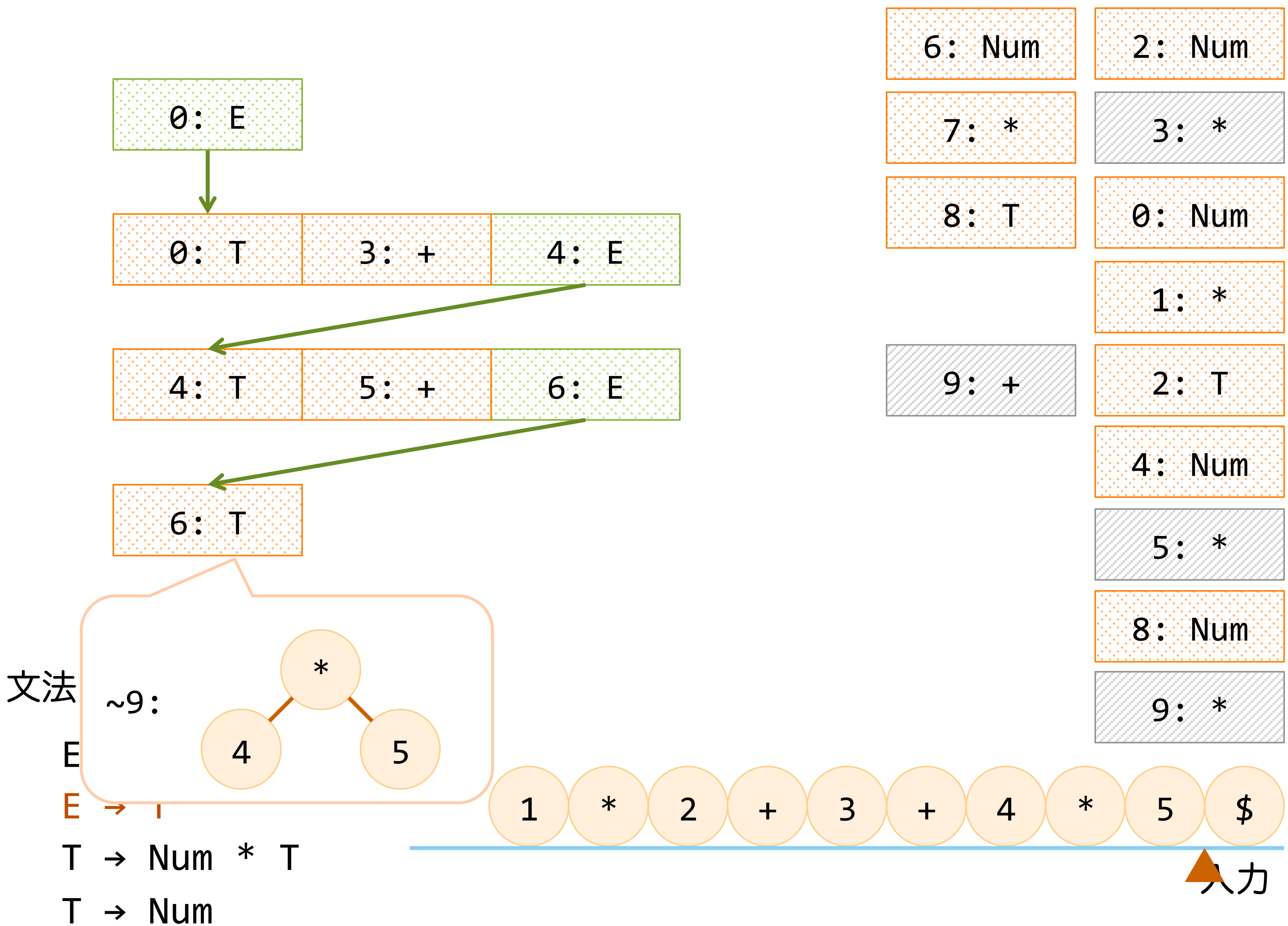
E → T + E

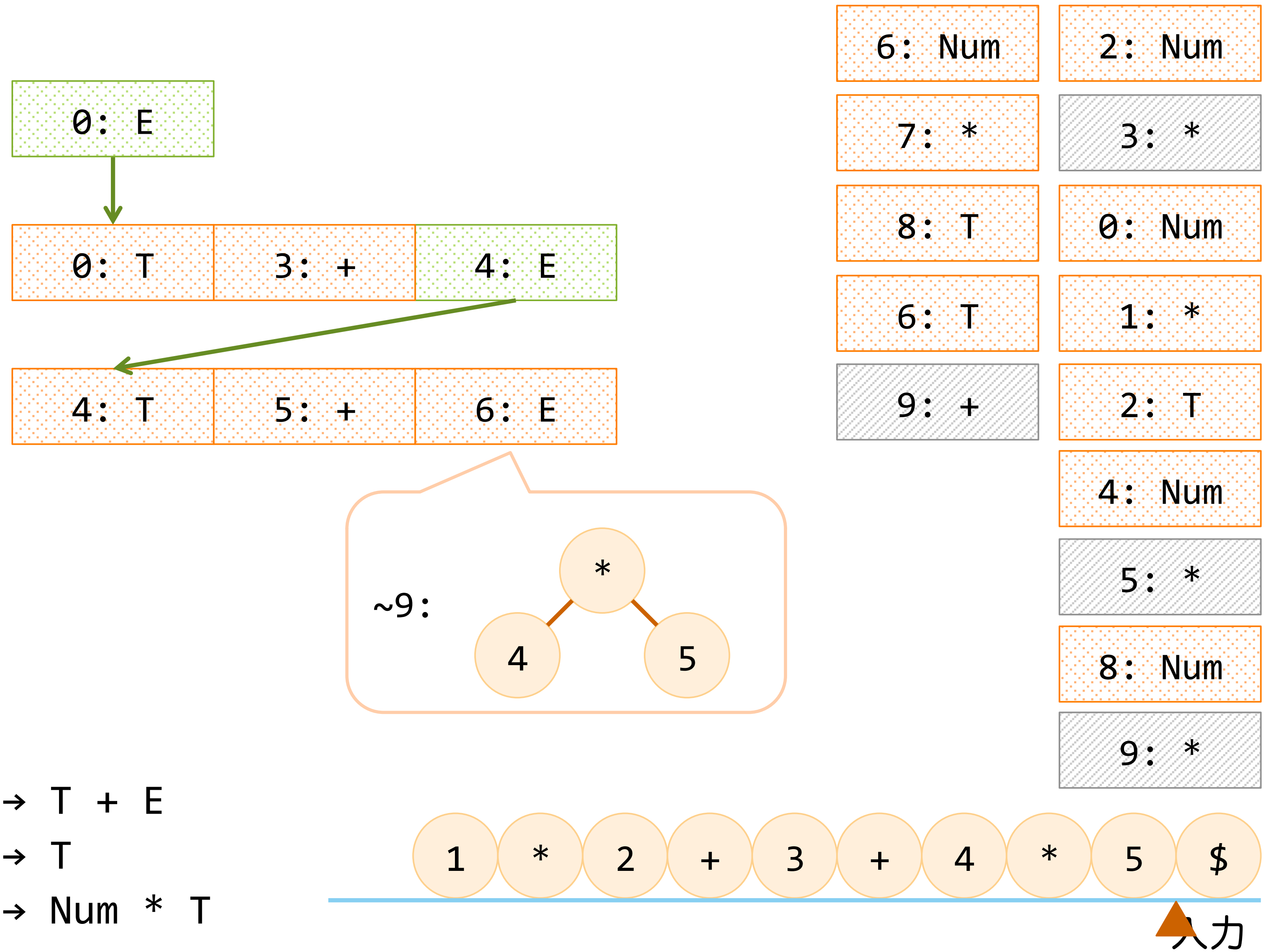
E → T

T → Num * T

T → Num

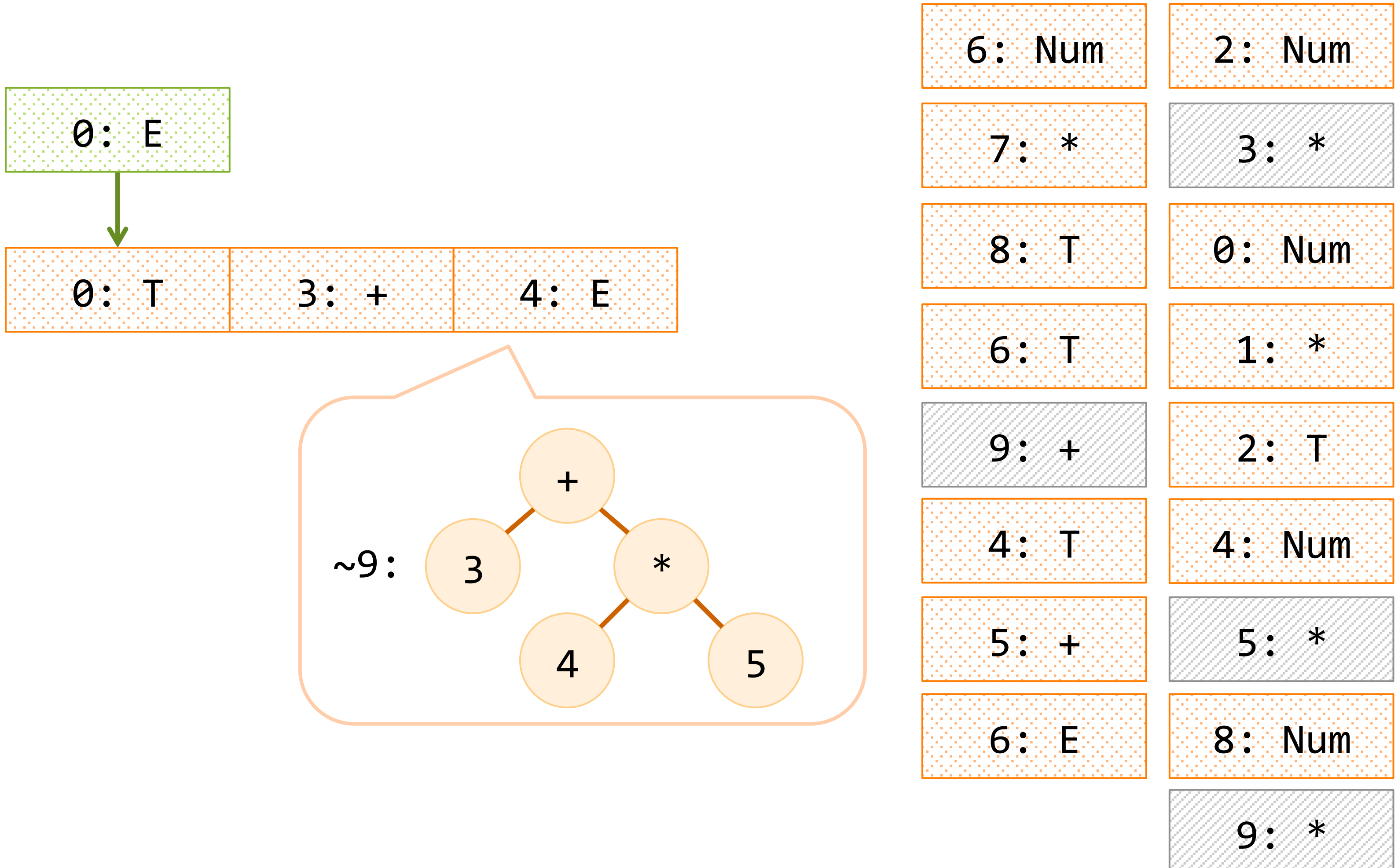


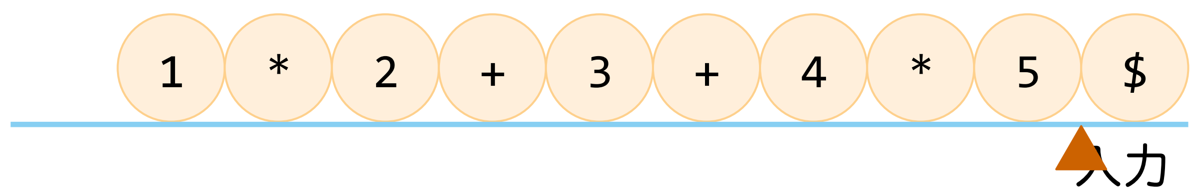

0: E
0: T
3: +
4: E
4: T
5: +
6: E
6: T
~9:
*
4
5
6: Num
7: *
8: T
9: +
2: Num
3: *
0: Num
1: *
2: T
4: Num
5: *
8: Num
9: *
文法
E
E → T
T → Num * T
T → Num
1
*
2
+
3
+
4
*
5
$
入力



文法

E → T + E
E → T
T → Num * T
T → Num





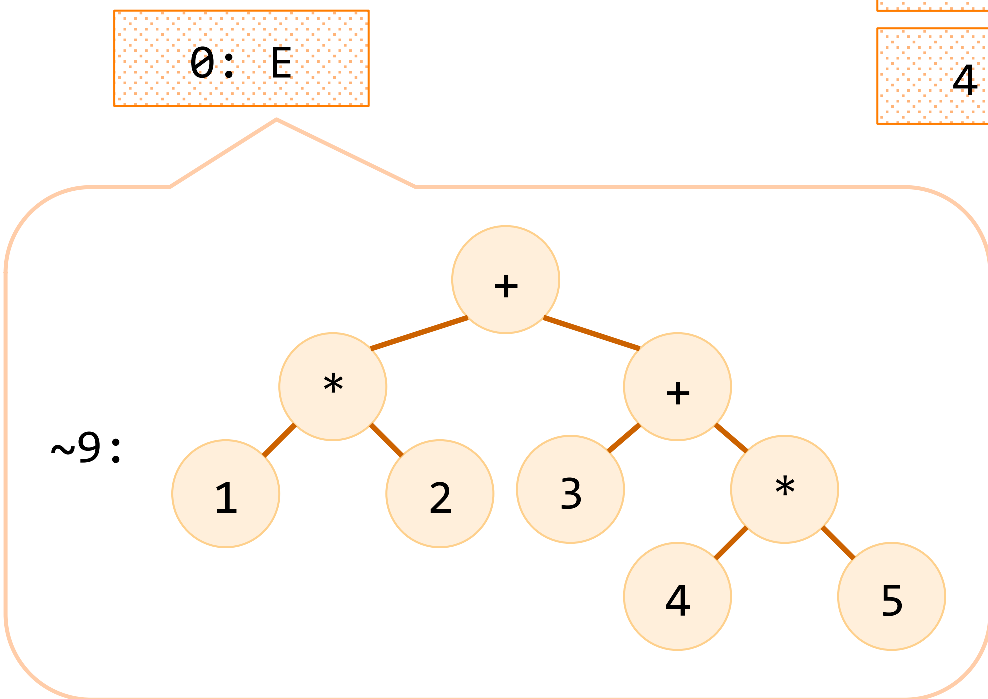
文法

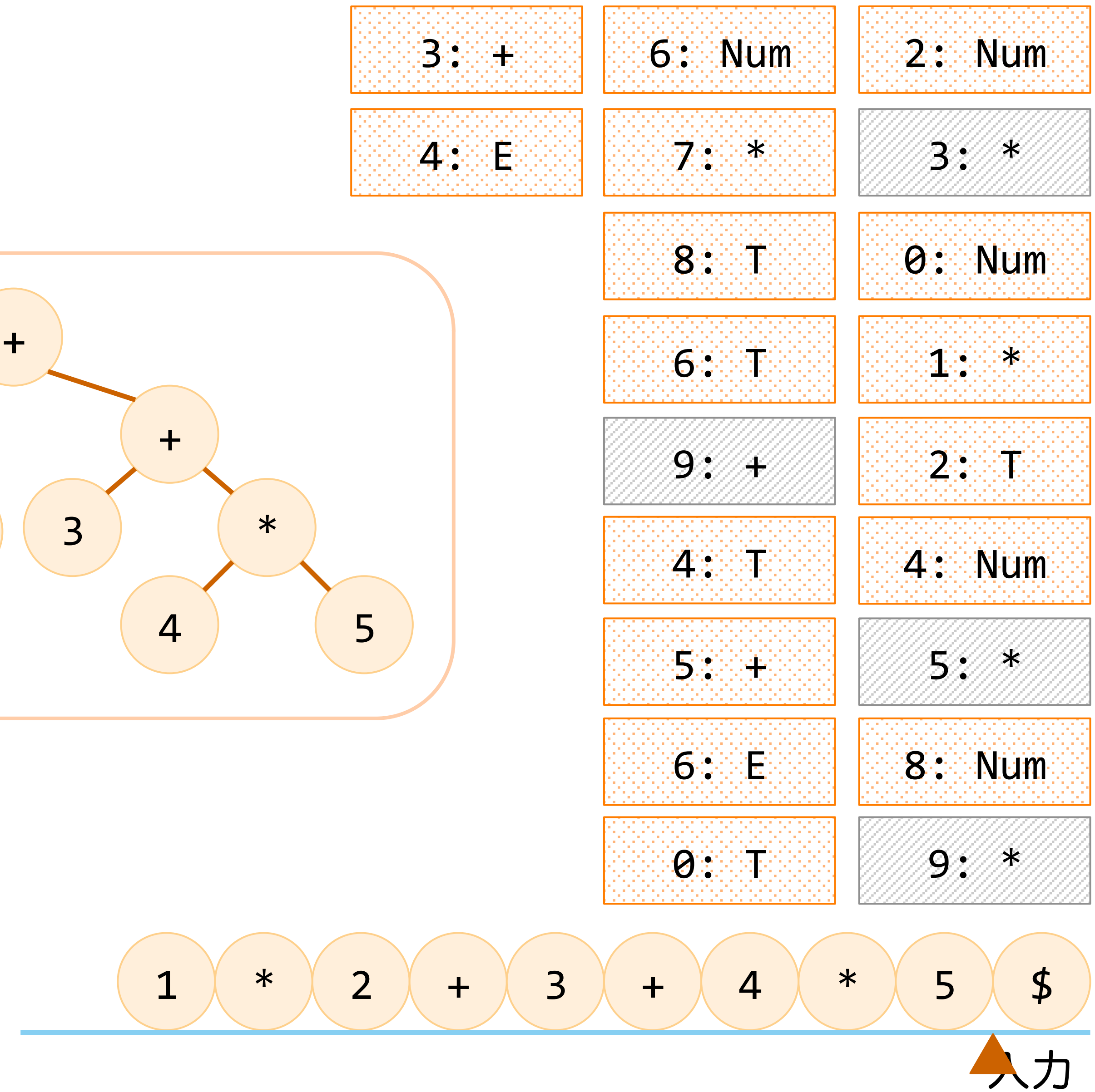
E → T + E

E → T

T → Num * T

T → Num





文法

E → T + E

E → T

T → Num * T

T → Num

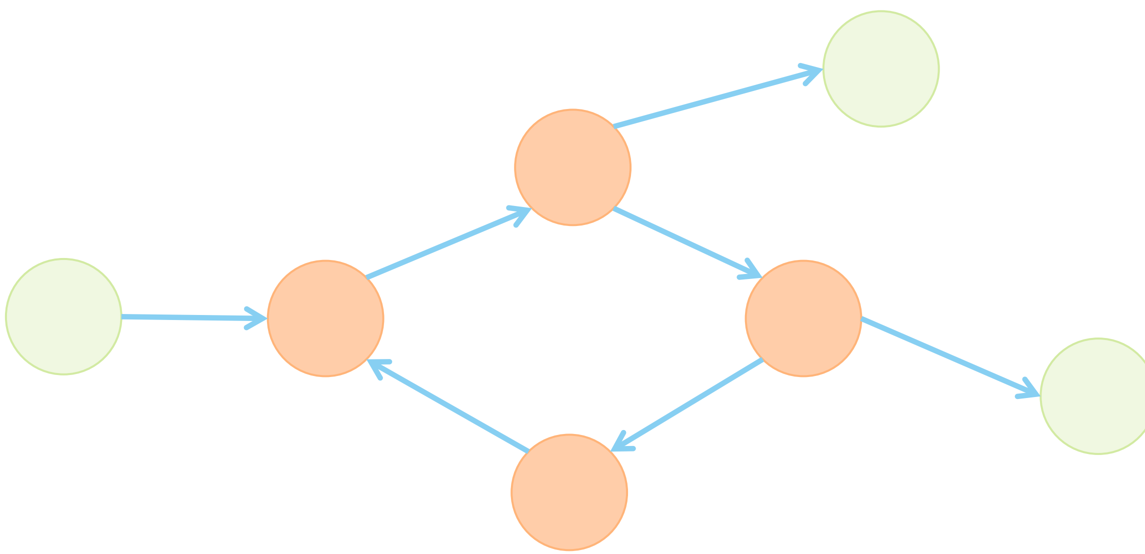
Packrat parsing [Ford 2002]

- SLL(k) で k → ∞ したものとみなせる
  - 文脈を無視して無限先読みして分岐を一つに決定する
  - 何らかの方法で一つに決まりさえすれば良い
  - LL(*) [Parr 2011] は可能なら SLL(k) で分岐するもの
- 動的計画法でも同様の解析は可能
  - 入力の後ろから順にたどって表を作る
- 能力
  - 計算量 : O(n)
  - 言語 : Parsing expression language
  - 文法 : Parsing expression grammar

Packrat parsing with left recursion [Warth 2008]

アイデア

- 構文解析中に左再帰検出
- 左再帰部は成長しなくなるまで繰り返し適用



文法

```
E → E + T
E → T
T → T * Num
T → Num
```

動的計画法ベースで説明する。
下向き構文解析と言えないかも？

入力

1 * 2 + 3 + 4 * 5 $

文法

```
E → E + T
E → T
T → T * Num
T → Num
```

入力

1 * 2 + 3 + 4 * 5 $

終端記号

```
+   :
*   :
Num:
```

非終端記号

```
T   :
E   :
```

文法

E → E + T

E → T

T → T * Num

T → Num

入力を読む

入力

1 * 2 + 3 + 4 * 5 $

終端記号

+ : なし

* : なし

Num: なし

非終端記号

T :

E :

文法

E → E + T
E → T
T → T * Num
T → Num

文法を見る

入力

1 * 2 + 3 + 4 * 5 $

終端記号
+ : なし
* : なし
Num: なし

非終端記号
T : なし
E :

文法

文法を見る

E → E + T
E → T
T → T * Num
T → Num

入力

1 * 2 + 3 + 4 * 5 $

終端記号
+ : なし
* : なし
Num: なし

非終端記号
T : なし
E : なし

文法

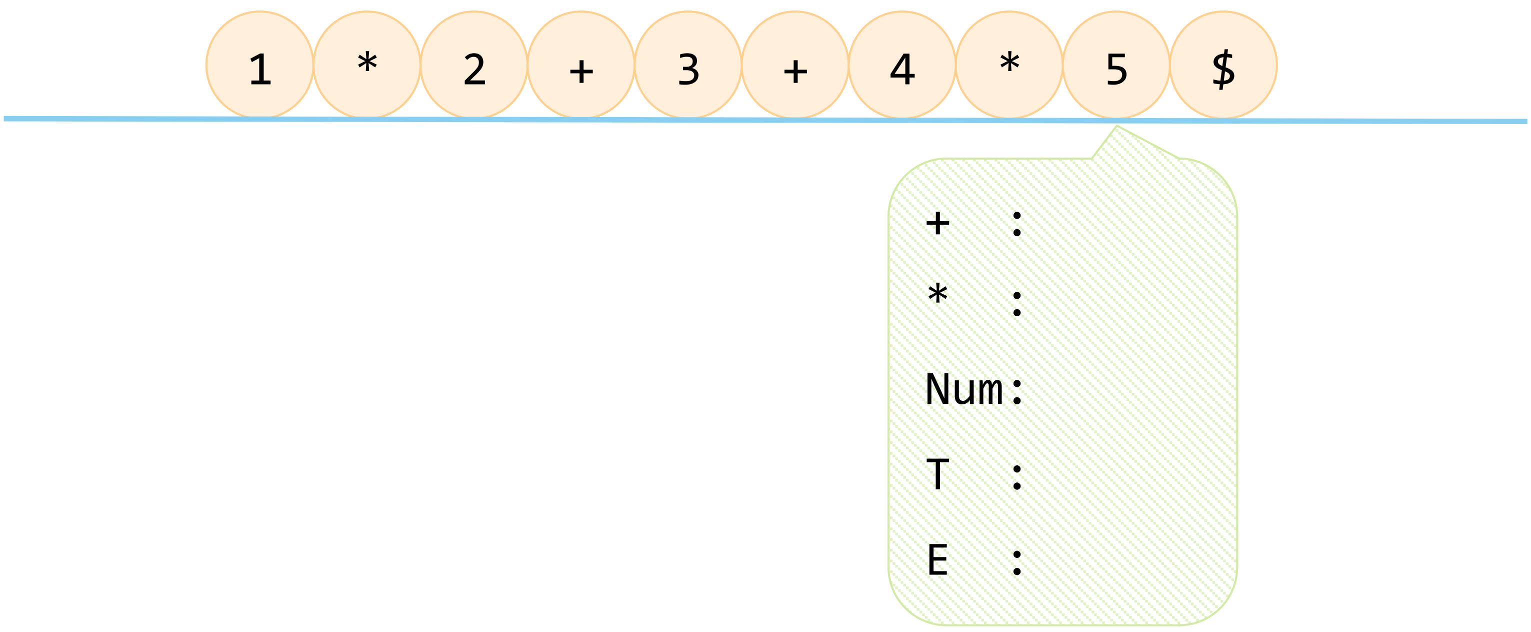
E → E + T

E → T

T → T * Num

T → Num

入力



文法

E → E + T

E → T

T → T * Num

T → Num

入力を読む

入力

1 * 2 + 3 + 4 * 5 $

+ : なし

* : なし

Num: 5

T :

E :

文法

```
E → E + T
E → T
T → T * Num
T → Num
```

文法を見る

入力

1 * 2 + 3 + 4 * 5 $

```
+  : なし
*  : なし
Num: 5
T  : 5
E  :
```

文法

文法を見る

E → E + T

E → T

T → T * Num

T → Num

入力

1	*	2	+	3	+	4	*	5	$

+　: なし

*　: なし

Num: 5

T　: 5

E　: 5

文法

E → E + T
E → T
T → T * Num
T → Num

もう一度文法を見る

入力

1 * 2 + 3 + 4 * 5 $

+ : なし
* : なし
Num: 5
T : 5
E : 5

更新不可

文法

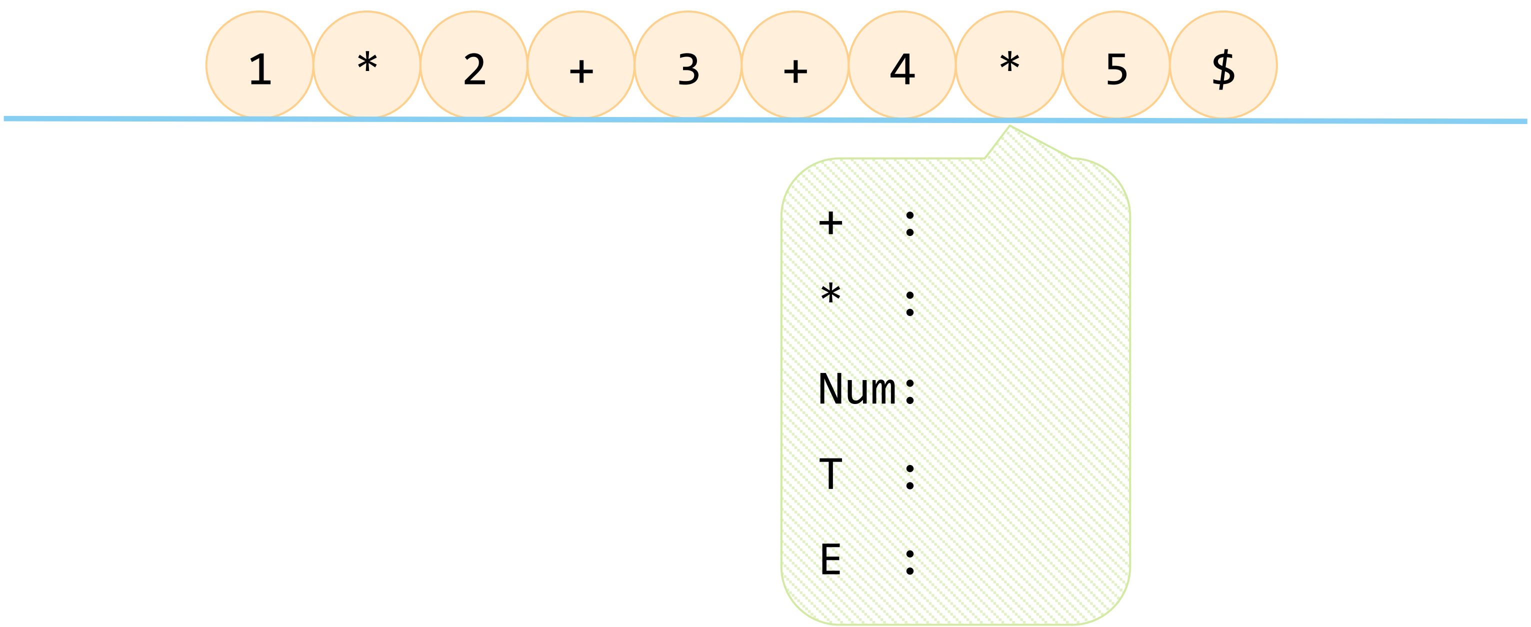
E → E + T

E → T

T → T * Num

T → Num

入力



文法

E → E + T

E → T

T → T * Num

T → Num

入力を読む

入力

1 * 2 + 3 + 4 * 5 $

+ : なし

* : *

Num: なし

T :

E :

文法

E → E + T

E → T

T → T * Num

T → Num

文法を見る

入力

1 * 2 + 3 + 4 * 5 $

+ : なし

* : *

Num: なし

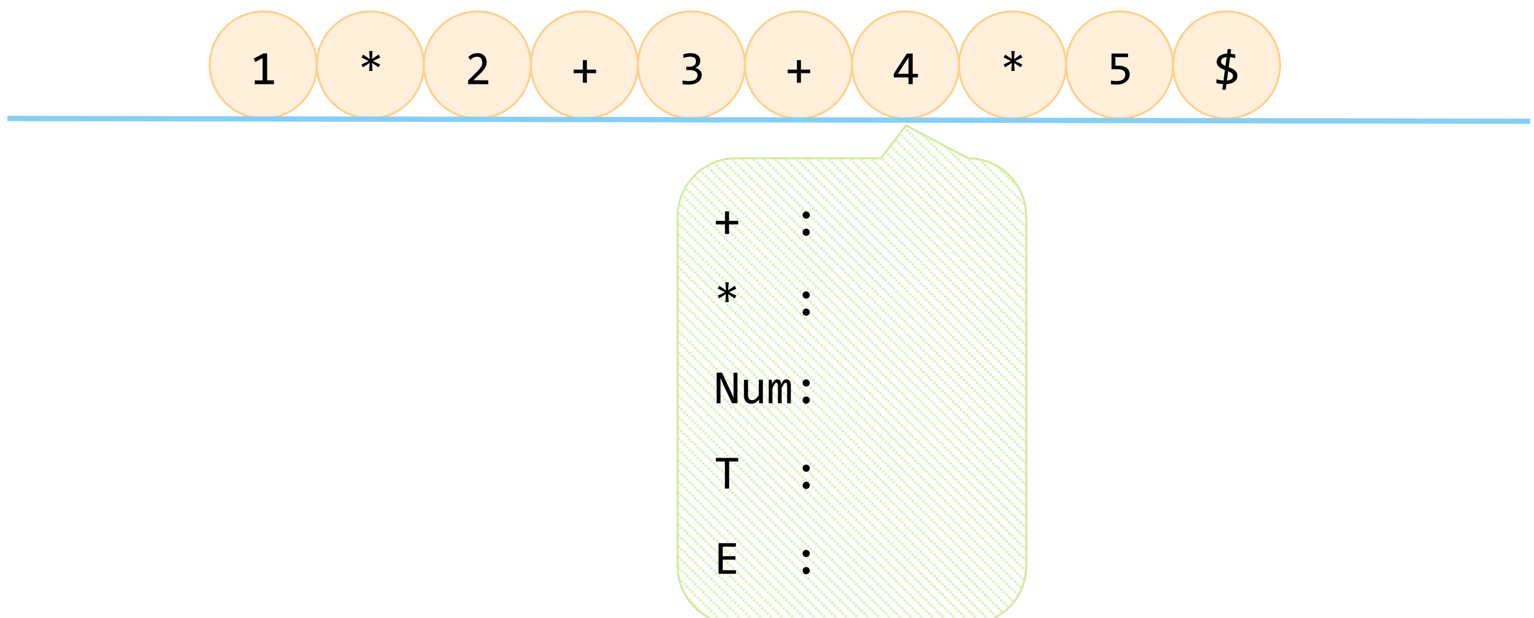
T : なし

E : なし

文法

```
E → E + T
E → T
T → T * Num
T → Num
```

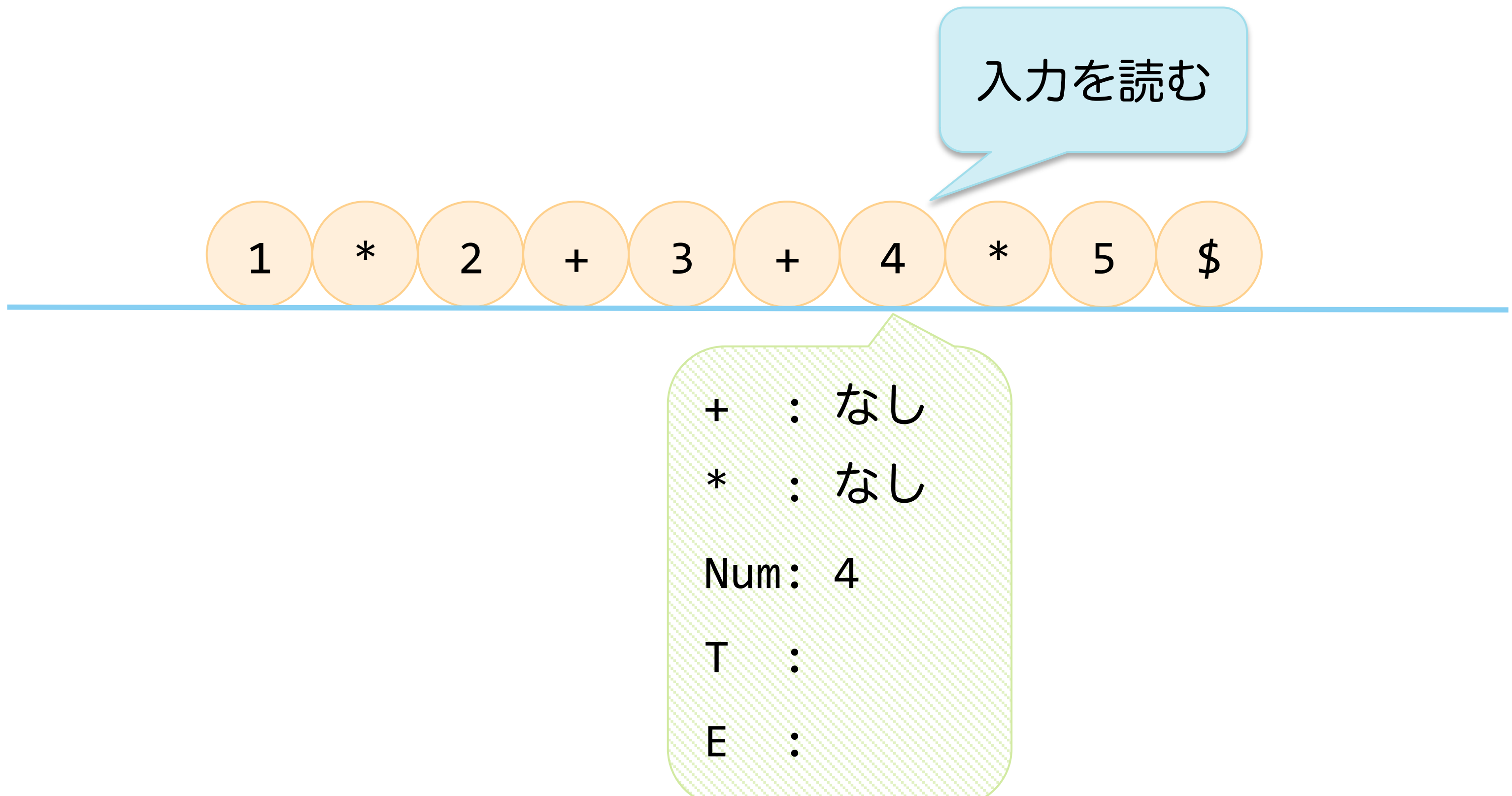
入力



文法

E → E + T
E → T
T → T * Num
T → Num

入力



文法

E → E + T

E → T

T → T * Num

T → Num

文法を見る

入力

1	*	2	+	3	+	4	*	5	$

+ : なし

* : なし

Num: 4

T : 4

E :

文法

E → E + T

E → T

T → T * Num

T → Num

文法を見る

入力

1 * 2 + 3 + 4 * 5 $

+ : なし

* : なし

Num: 4

T : 4

E : 4

文法

E → E + T

E → T

T → T * Num

T → Num

もう一度文法を見る

*: *

Num: 5

入力

1 * 2 + 3 + 4 * 5 $

+ : なし

* : なし

Num: 4

T : 4 * 5

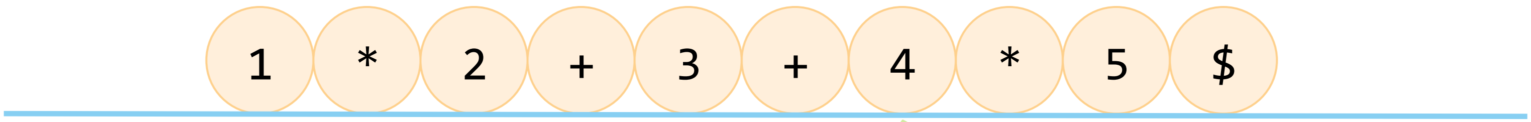
E : 4

文法

E → E + T
E → T
T → T * Num
T → Num

もう一度文法を見る

入力



\+ : なし
* : なし
Num: 4
T : 4 * 5
E : 4 * 5

文法

E → E + T
E → T
T → T * Num
T → Num

更にもう一度文法を見る

入力

1 * 2 + 3 + 4 * 5 $

\+ : なし
* : なし
Num: 4
T : 4 * 5
E : 4 * 5

更新不可

文法

E → E + T

E → T

T → T * Num

T → Num

以降同様に進める

入力

1	*	2	+	3	+	4	*	5	$

+ : +

* : なし

Num: なし

T :

E :

文法

```
E → E + T
E → T
T → T * Num
T → Num
```

入力

1 * 2 + 3 + 4 * 5 $

```
+  : +
*  : なし
Num: なし
T  : なし
E  :
```

文法

```
E → E + T
E → T
T → T * Num
T → Num
```

入力

1 * 2 + 3 + 4 * 5 $

```
+  : +
*  : なし
Num: なし
T  : なし
E  : なし
```

文法

```
E → E + T
E → T
T → T * Num
T → Num
```

入力

1 * 2 + 3 + 4 * 5 $

```
+  : なし
*  : なし
Num: 3
T  :
E  :
```

文法

```
E → E + T
E → T
T → T * Num
T → Num
```

入力

1 * 2 + 3 + 4 * 5 $

```
+   : なし
*   : なし
Num: 3
T   : 3
E   :
```

文法

```
E → E + T
E → T
T → T * Num
T → Num
```

入力

1 * 2 + 3 + 4 * 5 $

```
+   : なし
*   : なし
Num: 3
T   : 3
E   : 3
```

文法

E → E + T

E → T

T → T * Num

T → Num

もう一度

+: +

T: 4 * 5

入力

1 * 2 + 3 + 4 * 5 $

+ : なし

* : なし

Num: 3

T : 3

E : 3 + (4 * 5)

文法

E → E + T

E → T

T → T * Num

T → Num

更にもう一度

入力

1	*	2	+	3	+	4	*	5	$

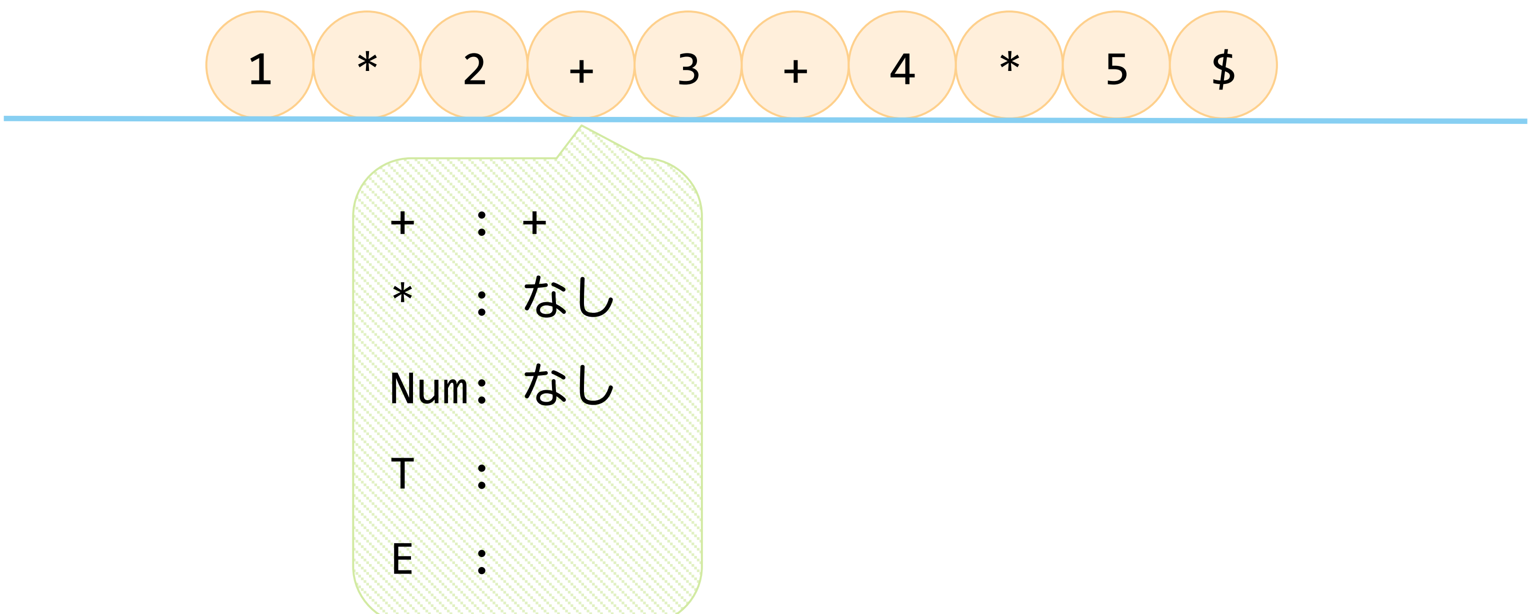
```
+   : なし
*   : なし
Num: 3
T   : 3
E   : 3 + (4 * 5)
```

更新不可

文法

E → E + T
E → T
T → T * Num
T → Num

入力



文法

```
E → E + T
E → T
T → T * Num
T → Num
```

入力

1 * 2 + 3 + 4 * 5 $

```
+  : +
*  : なし
Num: なし
T  : なし
E  :
```

文法

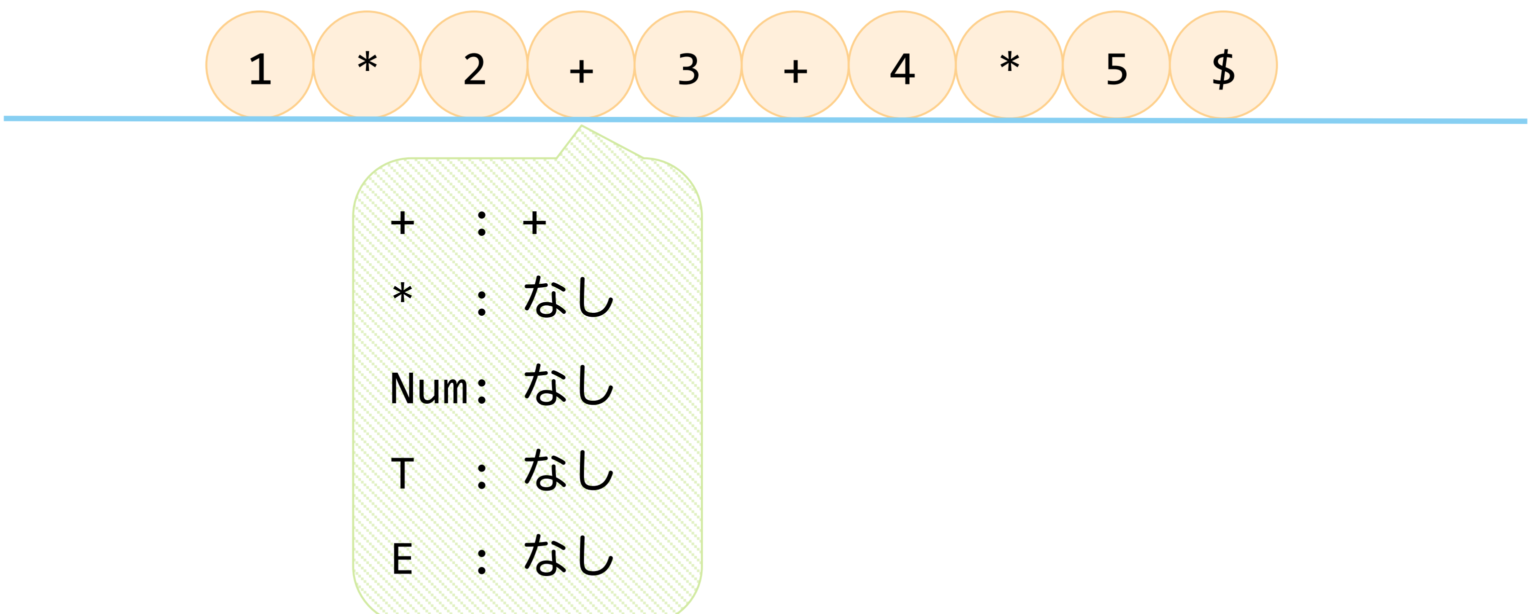
E → E + T

E → T

T → T * Num

T → Num

入力



文法

E → E + T

E → T

T → T * Num

T → Num

入力

1 * 2 + 3 + 4 * 5 $

+ : なし

* : なし

Num: 2

T :

E :

文法

E → E + T

E → T

T → T * Num

T → Num

入力

1 * 2 + 3 + 4 * 5 $

+ : なし

* : なし

Num: 2

T : 2

E :

文法

E → E + T
E → T
T → T * Num
T → Num

入力

1 * 2 + 3 + 4 * 5 $

\+ : なし
* : なし
Num: 2
T : 2
E : 2

文法

E → E + T

E → T

T → T * Num

T → Num

もう一度

入力

+: +

T: 3

1 * 2 + 3 + 4 * 5 $

+ : なし

* : なし

Num: 2

T : 2

E : 2 + 3

文法

E → E + T
E → T
T → T * Num
T → Num

更にもう一度

入力

+: +

T: 4 * 5

1 * 2 + 3 + 4 * 5 $

+ : なし
* : なし
Num: 2
T : 2
E : (2 + 3) + (4 * 5)

文法

E → E + T
E → T
T → T * Num
T → Num

更に更にもう一度

入力

1 * 2 + 3 + 4 * 5 $

+ : なし
* : なし
Num: 2
T : 2
E : (2 + 3) + (4 * 5)

更新不可

文法

E → E + T

E → T

T → T * Num

T → Num

入力

1 * 2 + 3 + 4 * 5 $

+ : なし

* : *

Num: なし

T :

E :

文法

E → E + T

E → T

T → T * Num

T → Num

入力

1 * 2 + 3 + 4 * 5 $

+ : なし

* : *

Num: なし

T : なし

E :

文法

E → E + T

E → T

T → T * Num

T → Num

入力

1 * 2 + 3 + 4 * 5 $

+	: なし
*	: *
Num	: なし
T	: なし
E	: なし

文法

E → E + T

E → T

T → T * Num

T → Num

入力

1 * 2 + 3 + 4 * 5 $

+ : なし

* : なし

Num: 1

T :

E :

文法

E → E + T
E → T
T → T * Num
T → Num

入力

1 * 2 + 3 + 4 * 5 $

+ : なし
* : なし
Num: 1
T : 1
E :

文法

```
E → E + T
E → T
T → T * Num
T → Num
```

入力

1 * 2 + 3 + 4 * 5 $

```
+   : なし
*   : なし
Num: 1
T   : 1
E   : 1
```

文法

E → E + T

E → T

T → T * Num

T → Num

もう一度

入力

*: *

Num: 2

1 * 2 + 3 + 4 * 5 $

+ : なし

* : なし

Num: 1

T : 1 * 2

E : 1

文法

E → E + T

E → T

T → T * Num

T → Num

もう一度

入力

1	*	2	+	3	+	4	*	5	$

\+ : なし

* : なし

Num: 1

T : 1 * 2

E : 1 * 2

文法

E → E + T

E → T

T → T * Num

T → Num

更にもう一度

入力

+: +

T: 3

1	*	2	+	3	+	4	*	5	$

\+ : なし

* : なし

Num: 1

T : 1 * 2

E : (1 * 2) + 3

文法

E → E + T
E → T
T → T * Num
T → Num

更に更にもう一度

入力

+: +

T: 4 * 5

1 * 2 + 3 + 4 * 5 $

+ : なし
* : なし
Num: 1
T : 1 * 2
E : ((1 * 2) + 3) + (4 * 5)

文法

E → E + T
E → T
T → T * Num
T → Num

更に更に更にもう一度

入力

1 * 2 + 3 + 4 * 5 $

+ : なし
* : なし
Num: 1
T : 1 * 2
E : ((1 * 2) + 3) + (4 * 5)

更新不可

文法

```
E → E + T
E → T
T → T * Num
T → Num
```

入力

1 * 2 + 3 + 4 * 5 $

```
+   : なし
*   : なし
Num: 1
T   : 1 * 2
E   : ((1 * 2) + 3) + (4 * 5)
```

終了

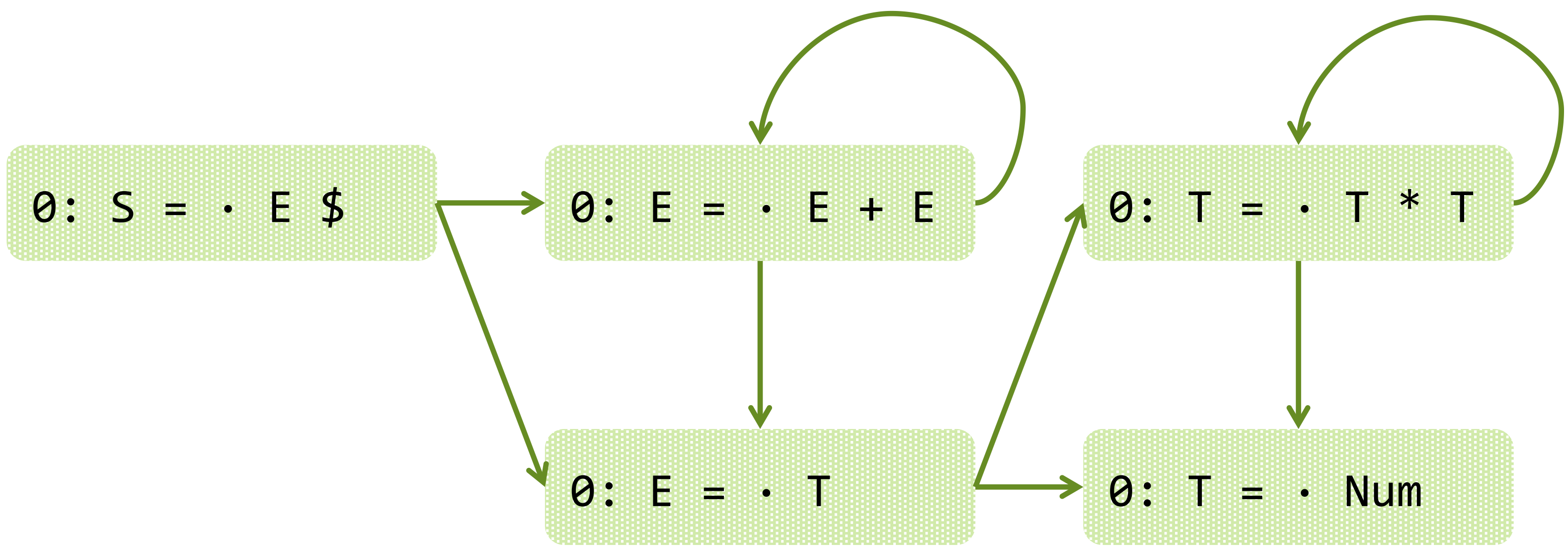
Packrat parsing with left recursion [Warth 2008]

- メモ化再帰ベースの方がやや効率が良い
  - 実装は難しい。論文のやつバグってない？
- 大抵のケースで計算量は O(n)
  - メモ化が無駄になるケースは計算量が悪化
  - 現実的な文法には最悪ケースはほぼ出現しないらしい
- 能力
  - 計算量 : $O(n^2)$ (論文にははっきりとは書いてないがたぶん)
  - 言語 : Parsing expression language
  - 文法 : 左再帰を含む Parsing expression grammar

GLL parsing [Scott 2010]

アイデア

- グラフ構造の継続を利用
- 幅優先探索



0: · E $

文法

S → E
E → E + E
E → T
T → T * T
T → Num

1 * 2 + 3 + 4 $

入力

0: · E $

0: · E + E

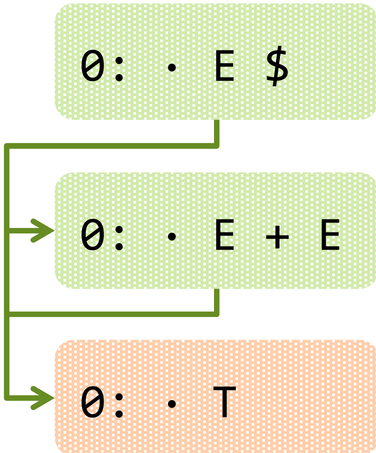
0: · T

文法

S → E
E → E + E
E → T
T → T * T
T → Num

1	*	2	+	3	+	4	$

入力



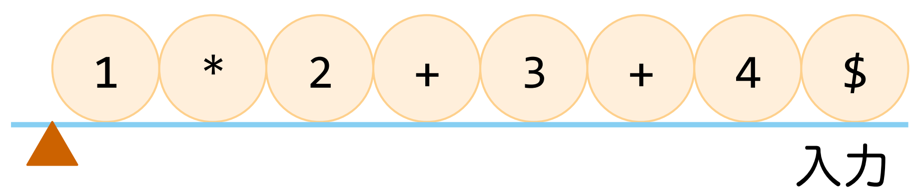
文法 S → E

E → E + E

E → T

T → T * T

T → Num



0: · E $

0: · E + E

0: · T

0: · T * T

0: · Num

文法

S → E

E → E + E

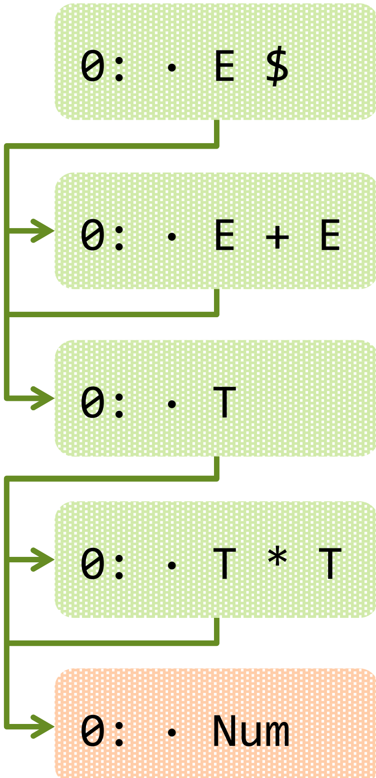
E → T

T → T * T

T → Num

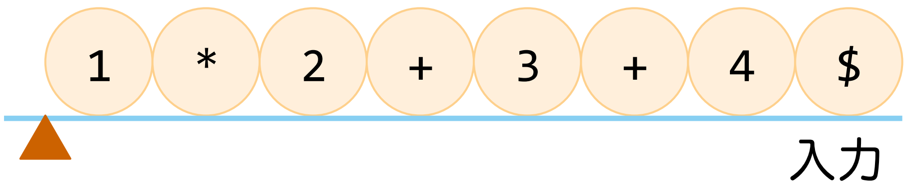
1	*	2	+	3	+	4	$

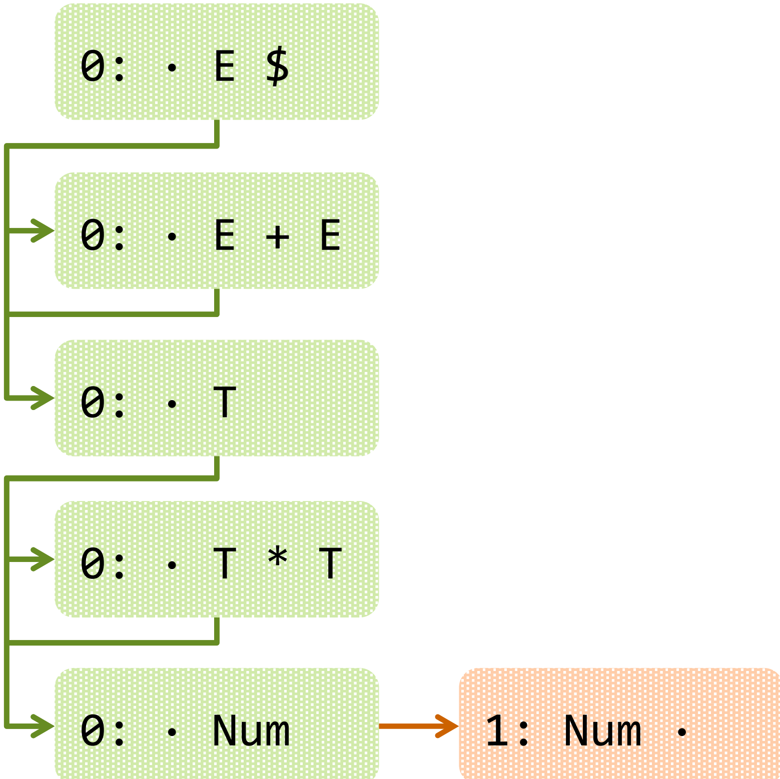
入力



文法

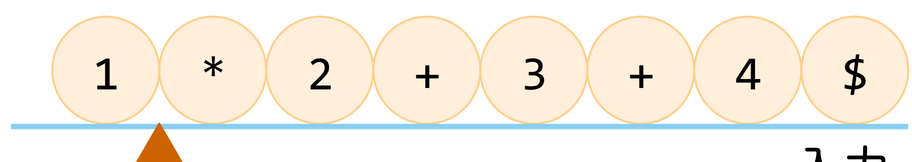
S → E
E → E + E
E → T
T → T * T
T → Num

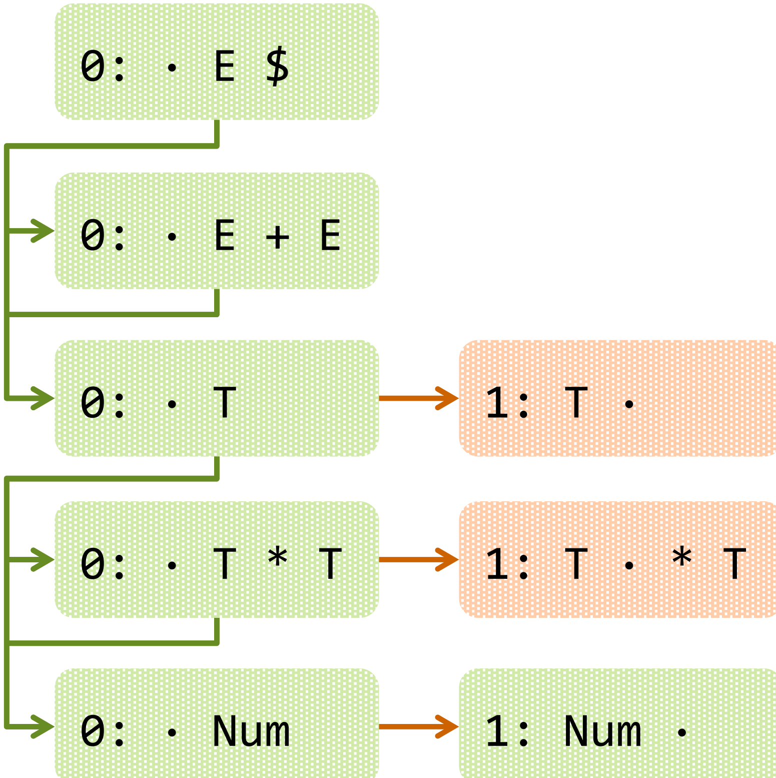




文法

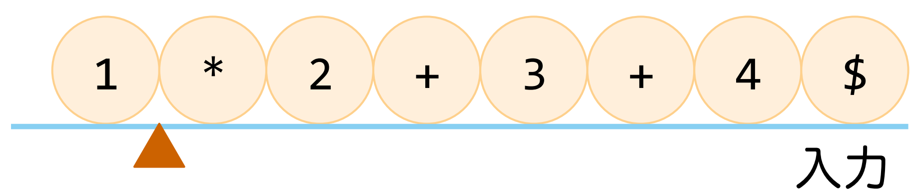
S → E
E → E + E
E → T
T → T * T
T → Num

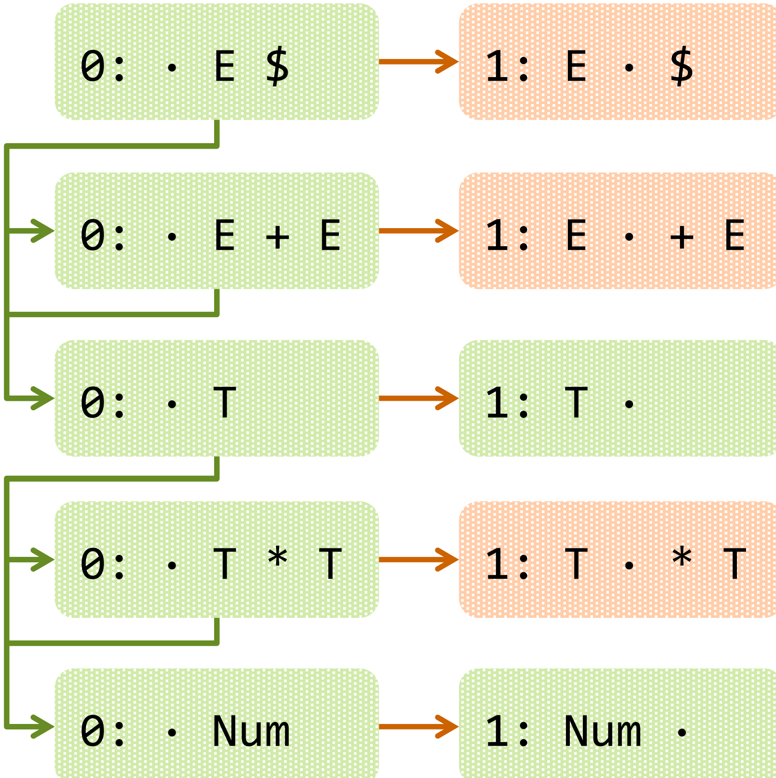




文法

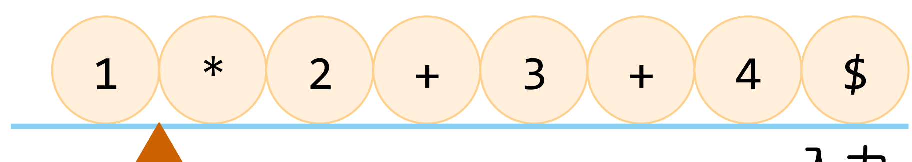
S → E
E → E + E
E → T
T → T * T
T → Num



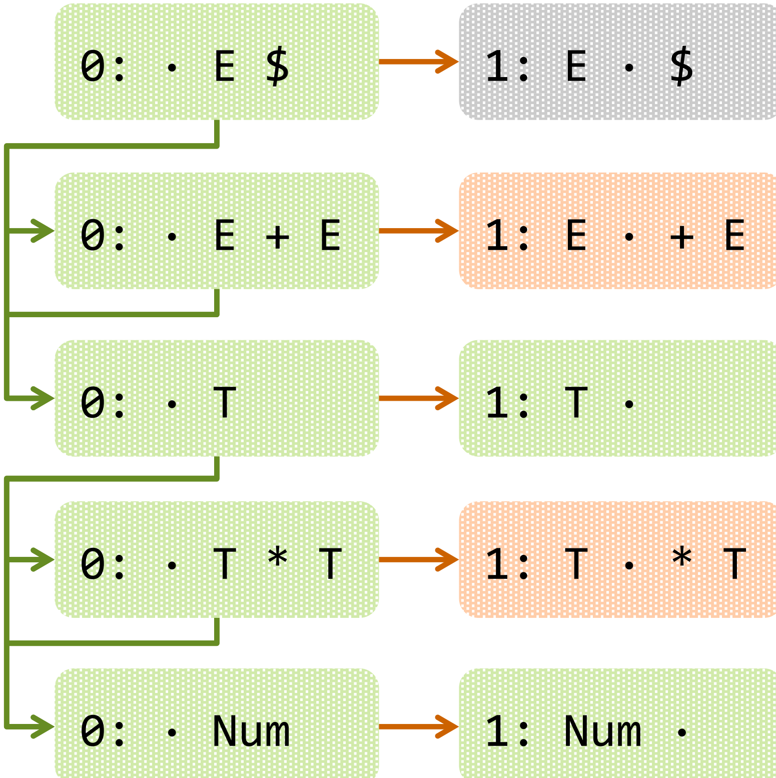


文法

$$
\begin{aligned}
S &\rightarrow E \\
E &\rightarrow E + E \\
E &\rightarrow T \\
T &\rightarrow T * T \\
T &\rightarrow \mathrm{Num}
\end{aligned}
$$

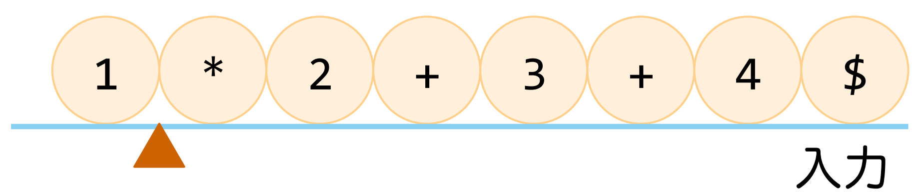


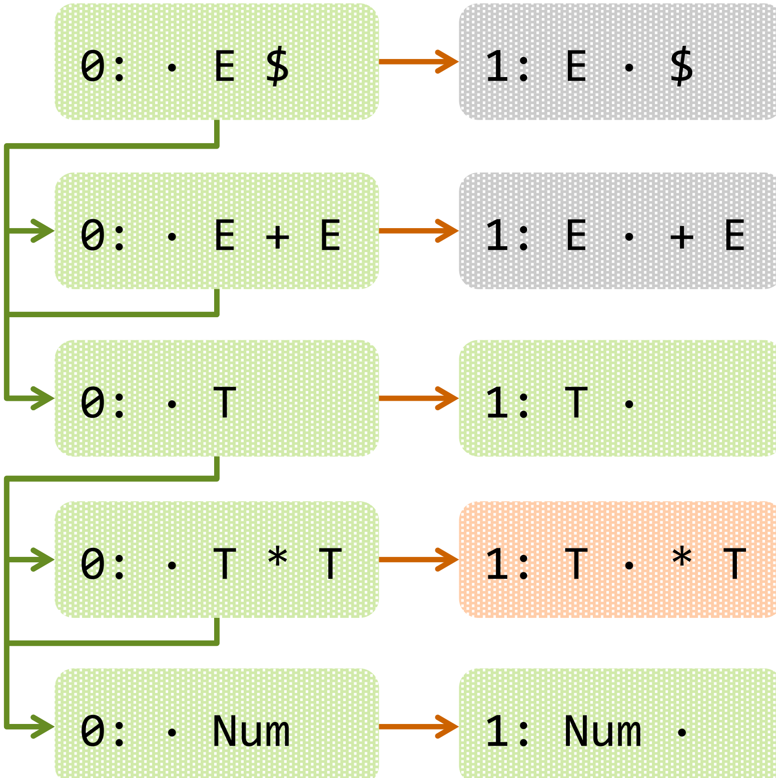
入力



文法

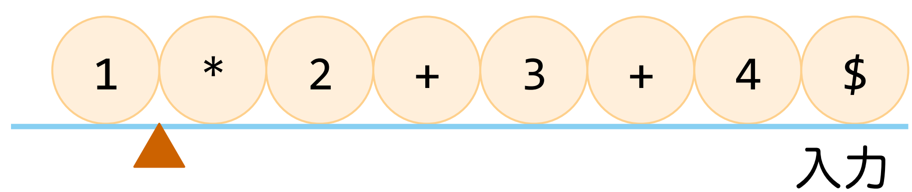
S → E
E → E + E
E → T
T → T * T
T → Num

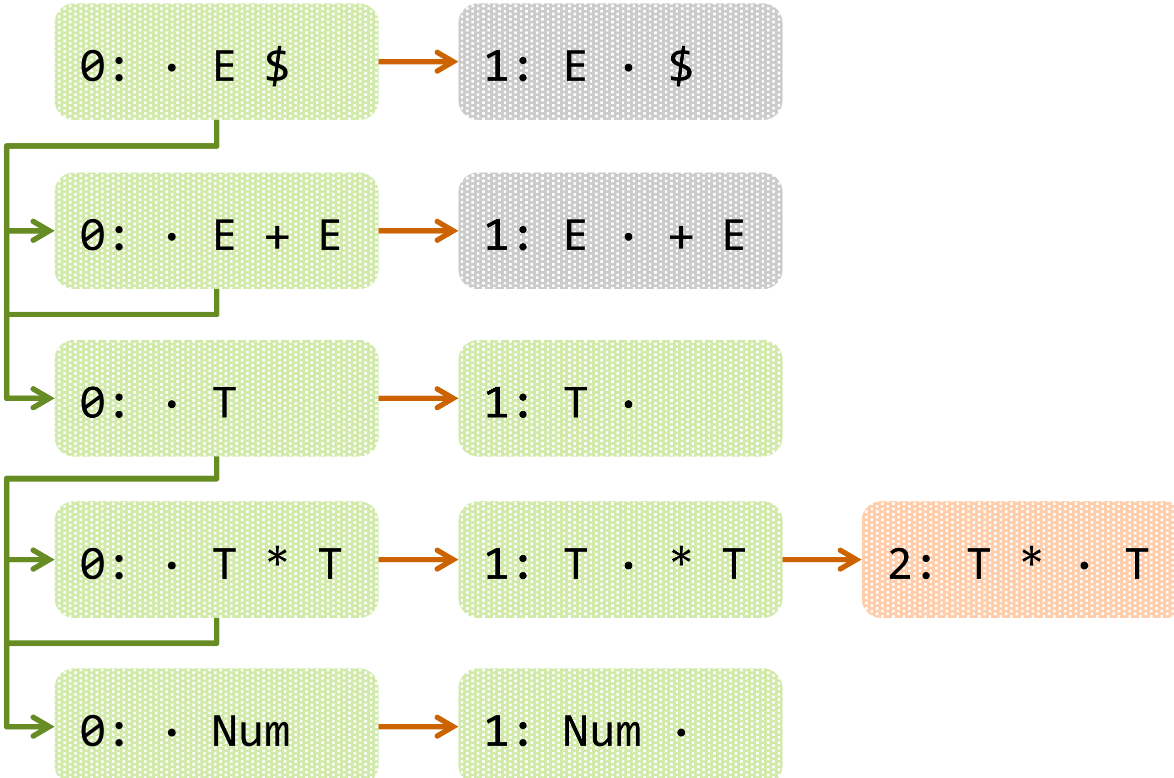




文法

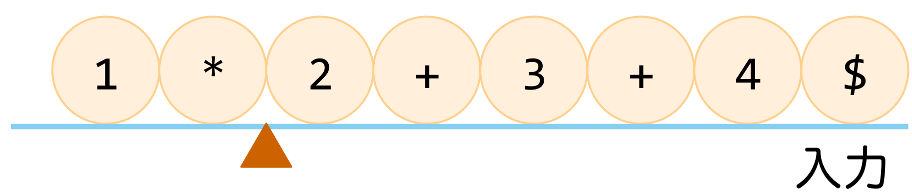
S → E
E → E + E
E → T
T → T * T
T → Num

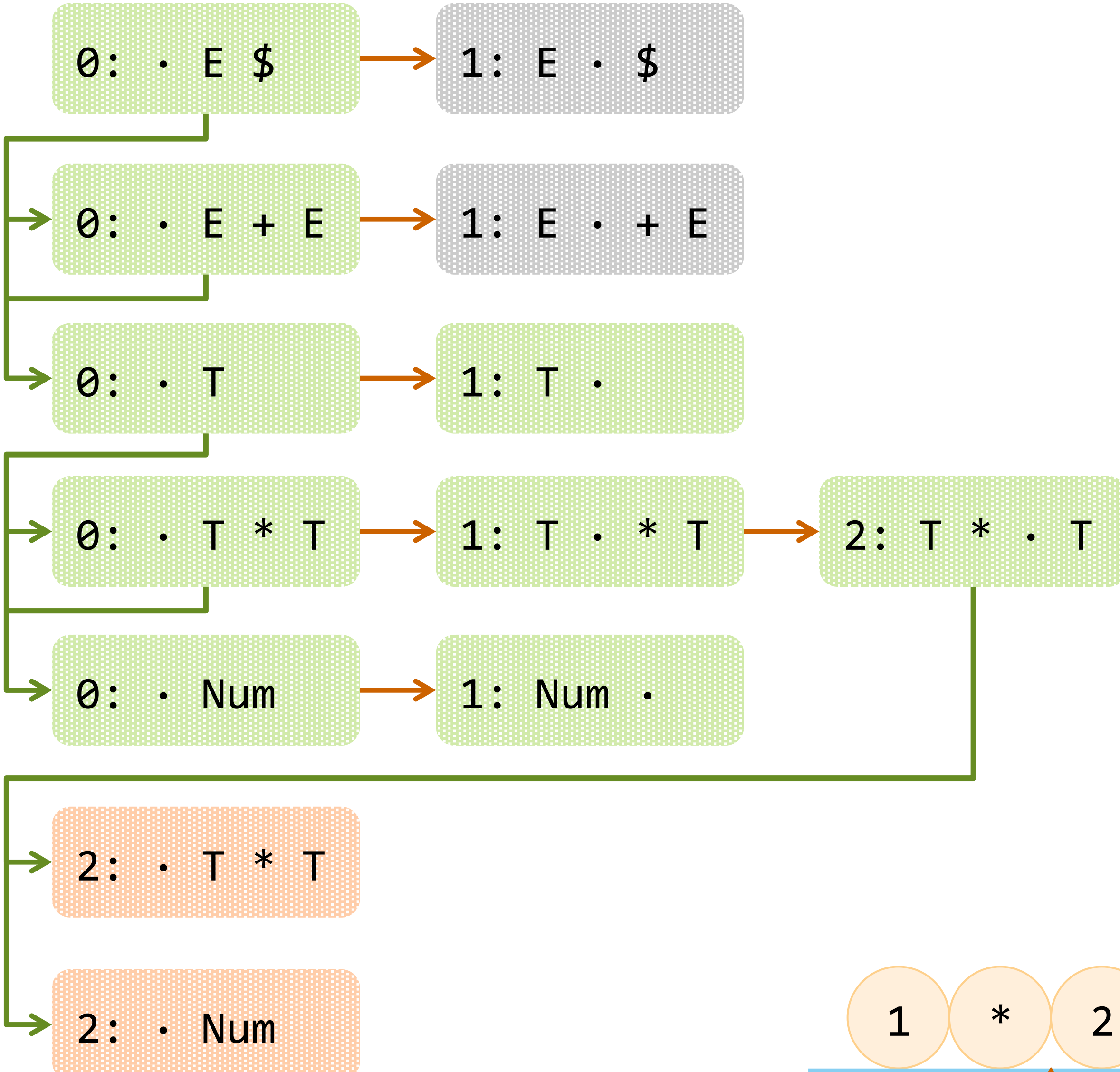




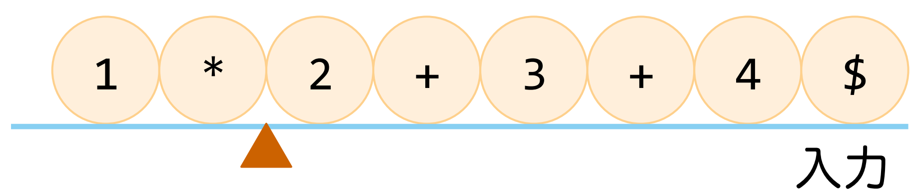
文法

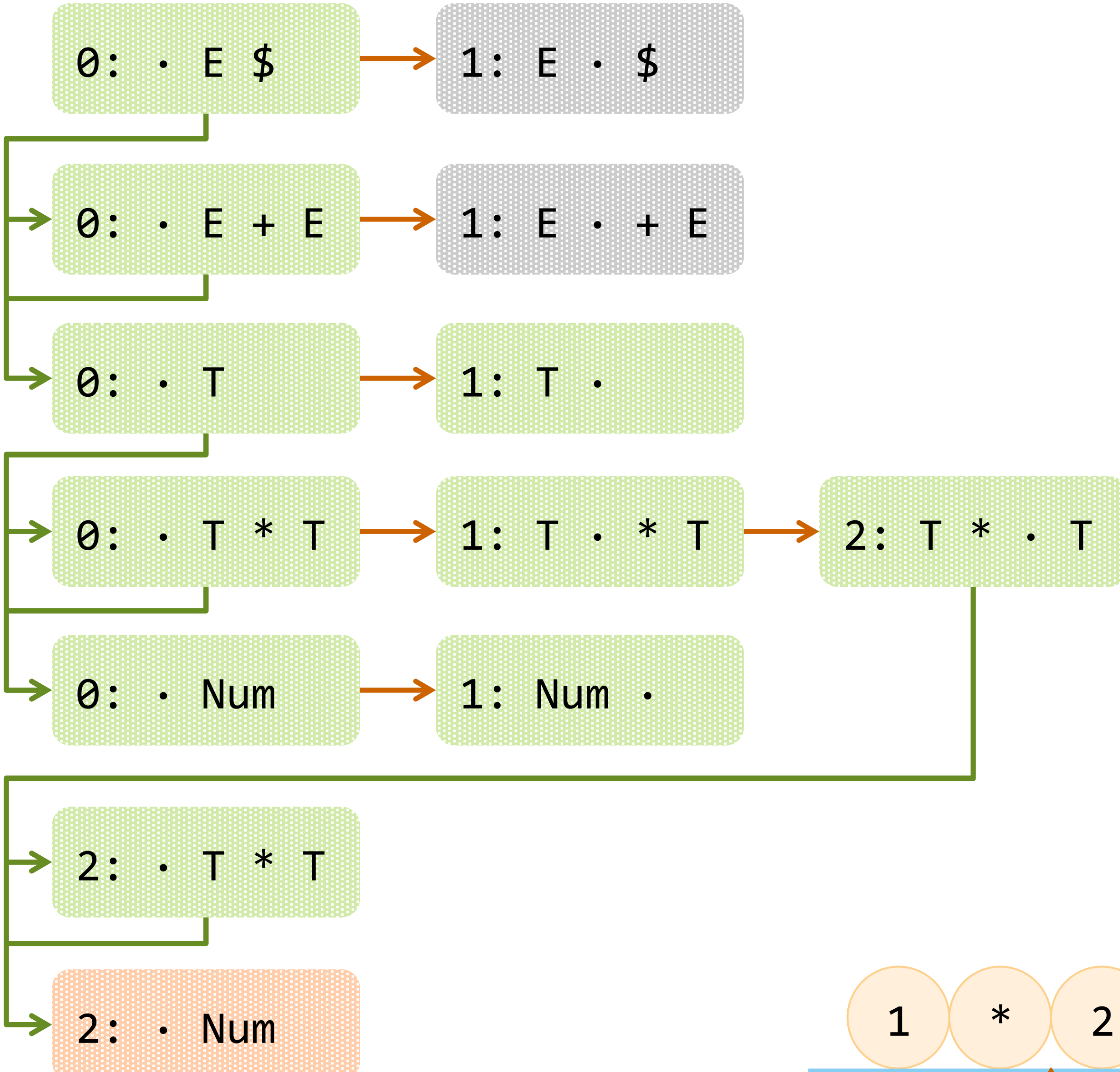
S → E
E → E + E
E → T
T → T * T
T → Num





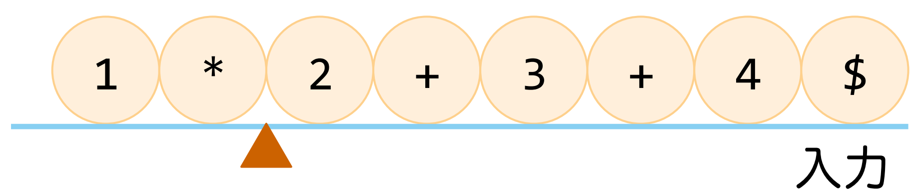
文法
S → E
E → E + E
E → T
T → T * T
T → Num

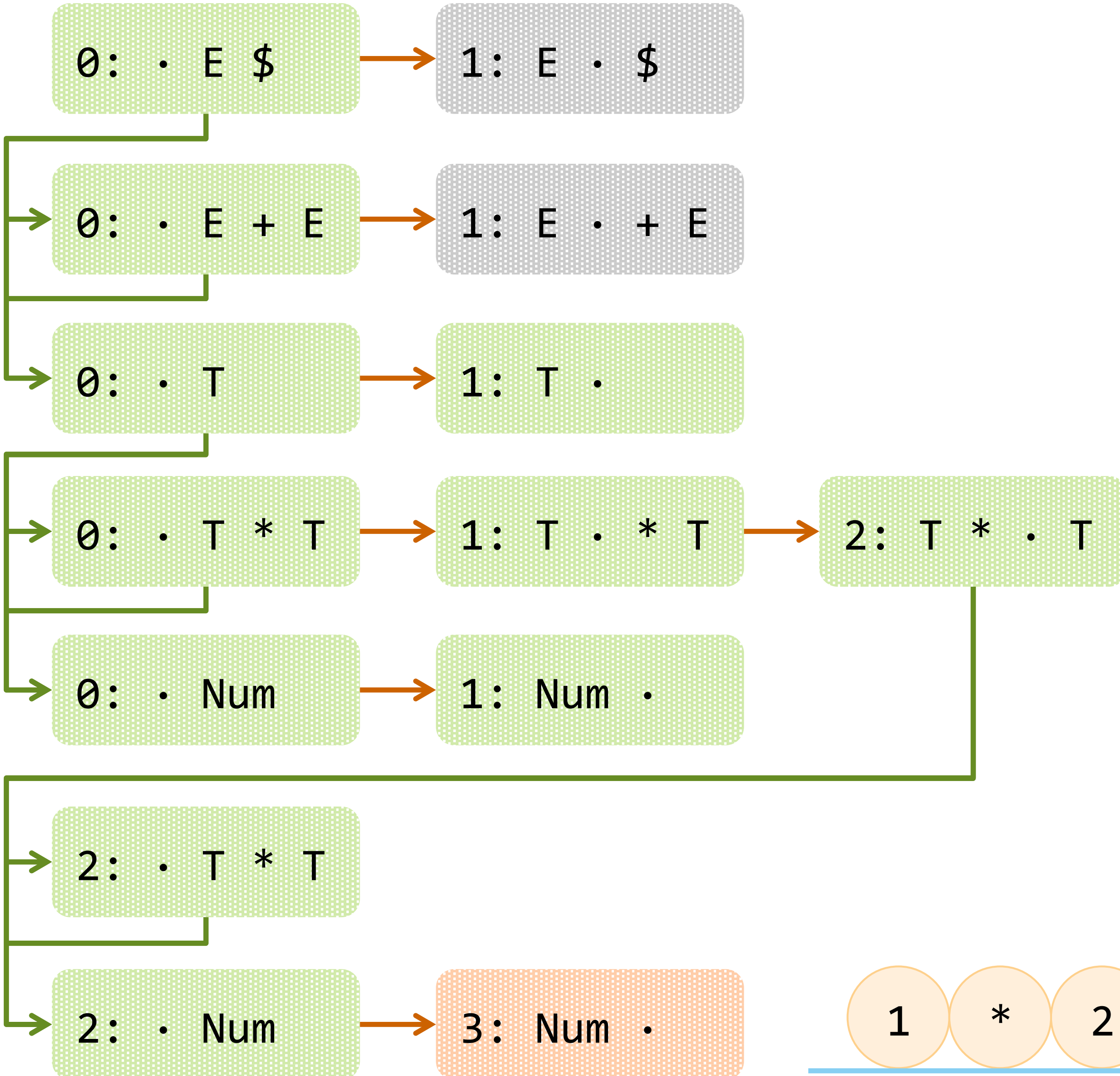




文法

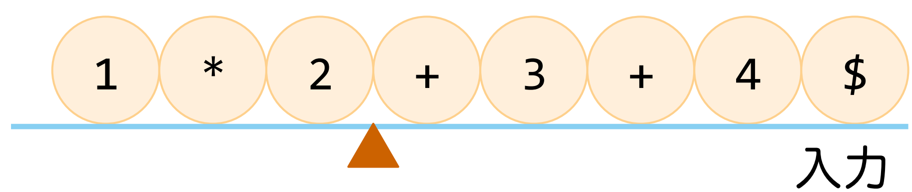
S → E
E → E + E
E → T
T → T * T
T → Num

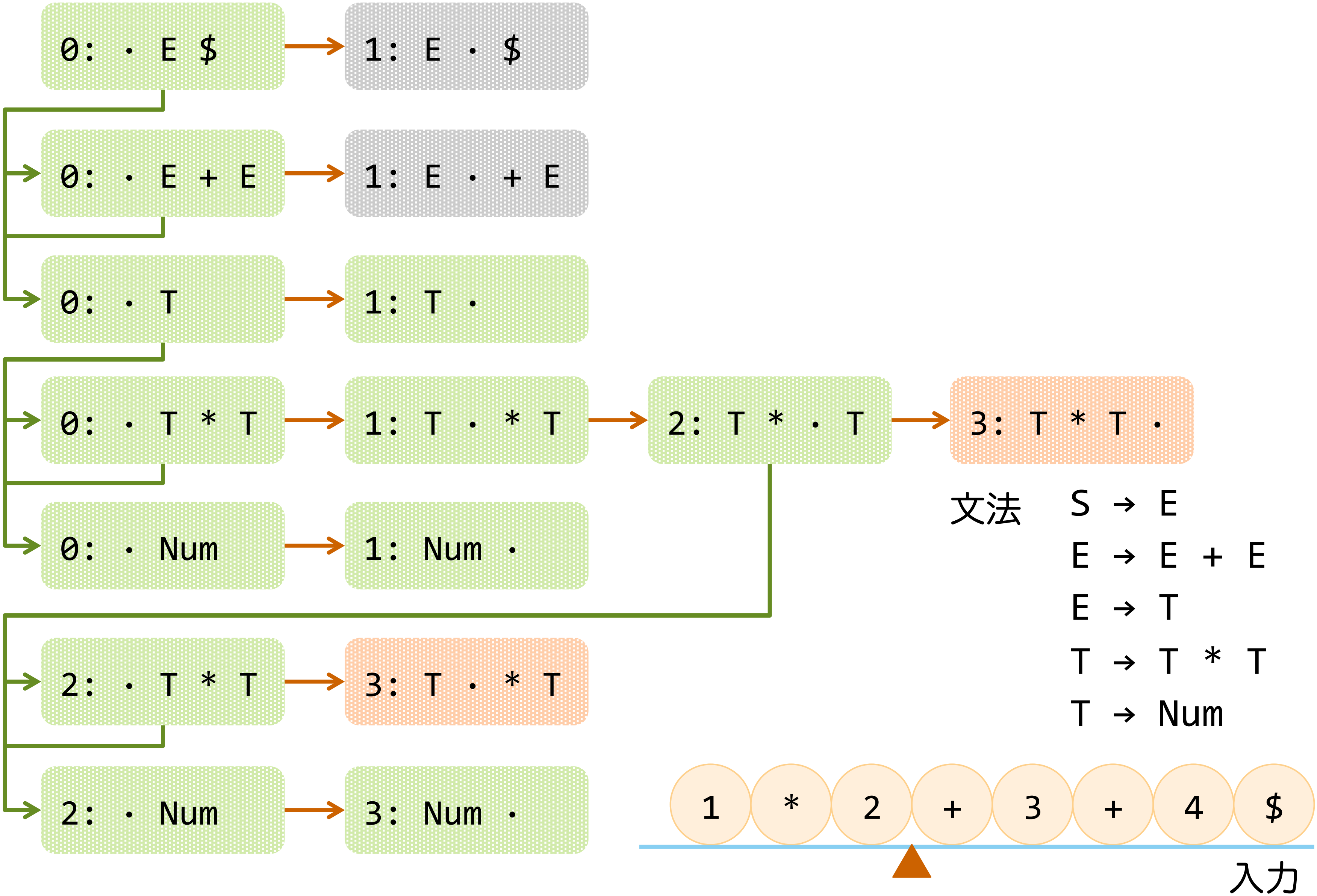


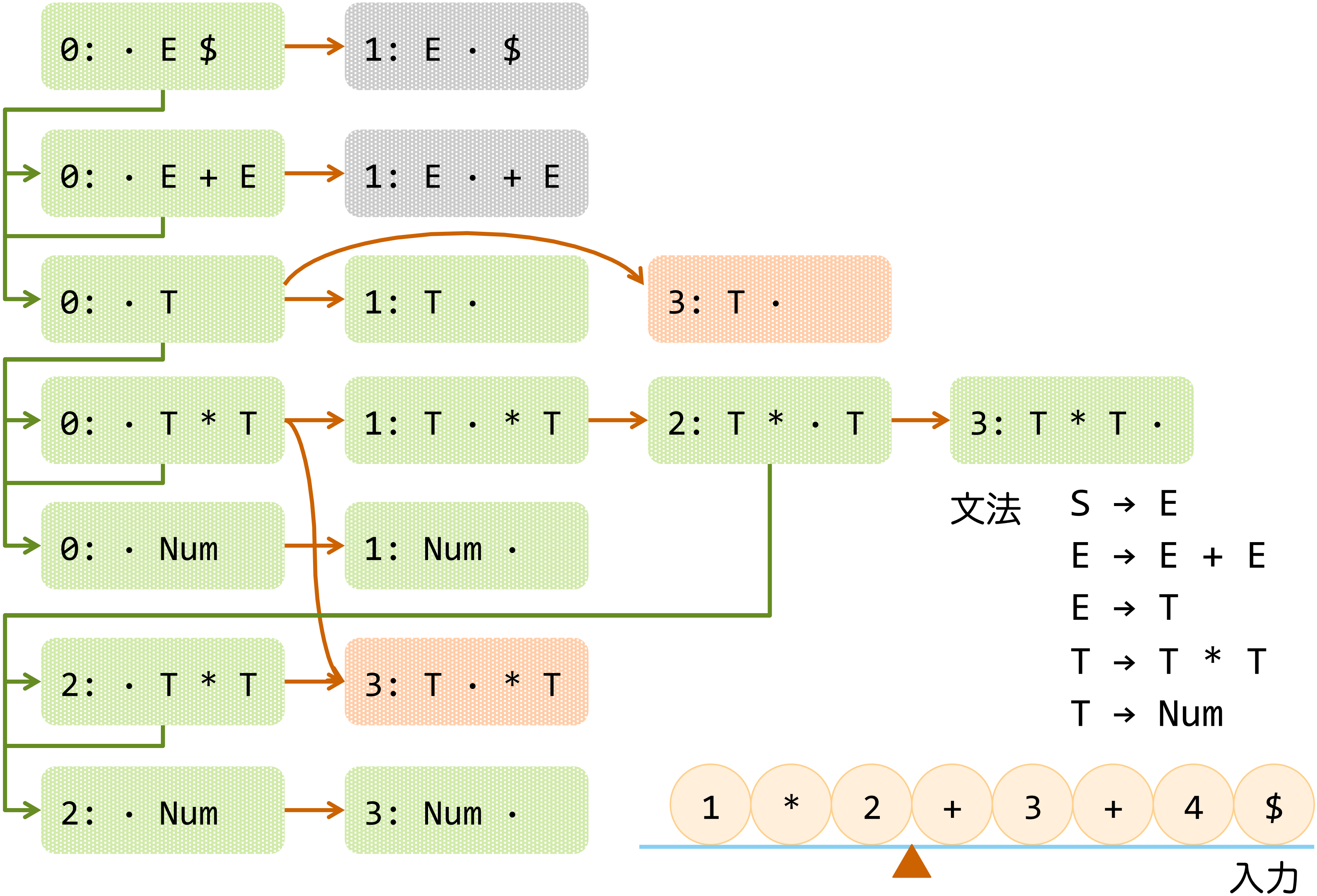


文法

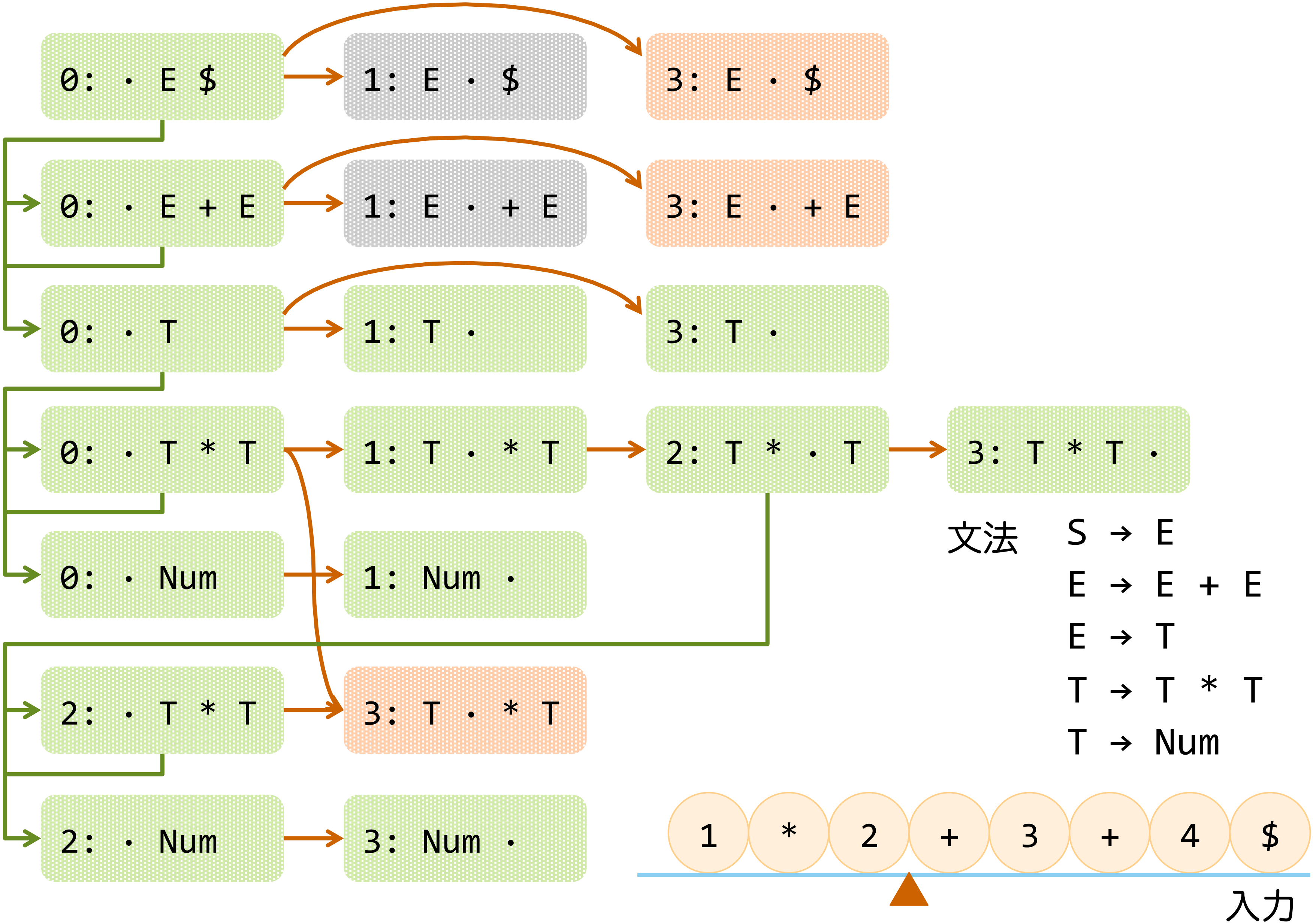
S → E
E → E + E
E → T
T → T * T
T → Num

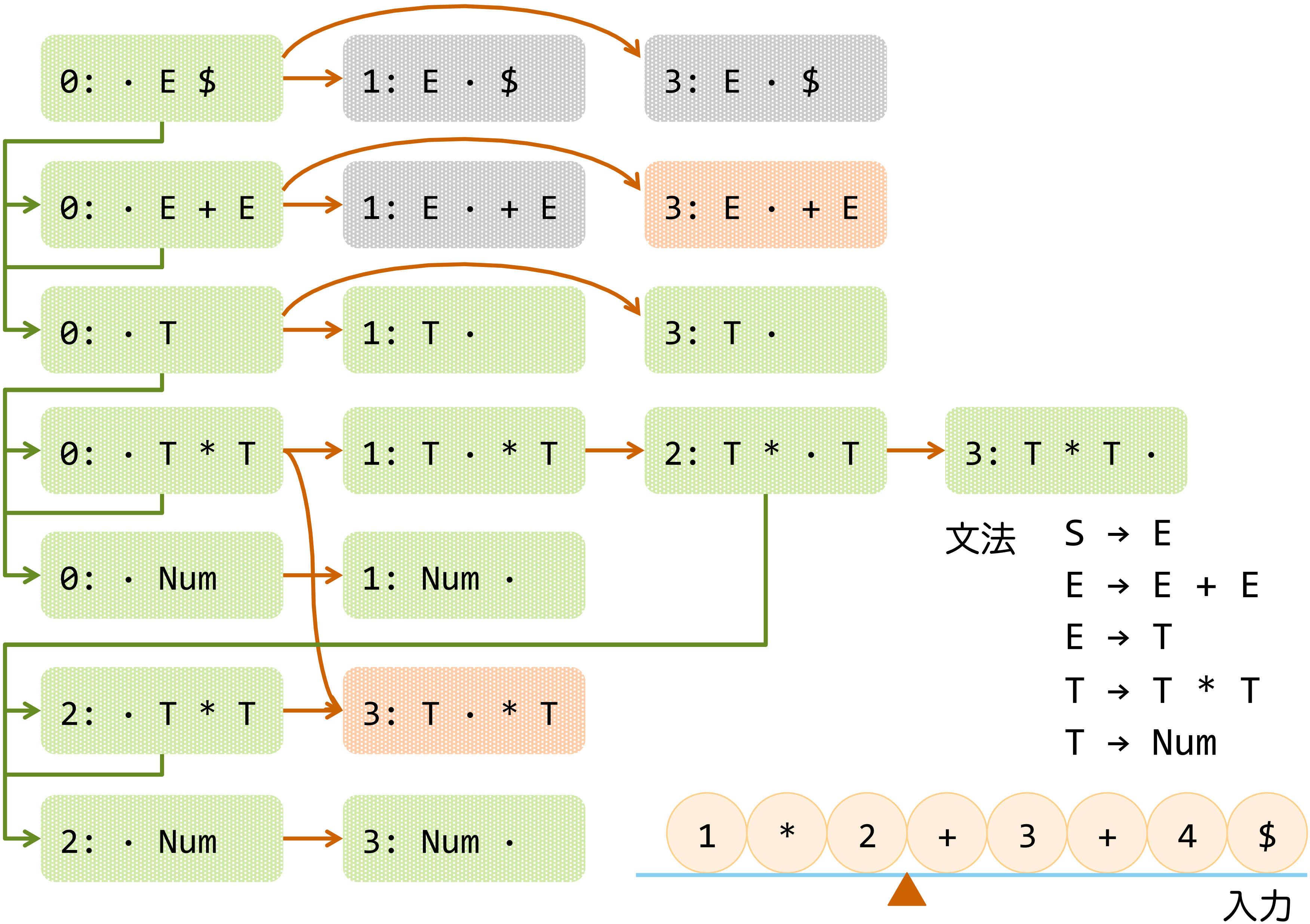


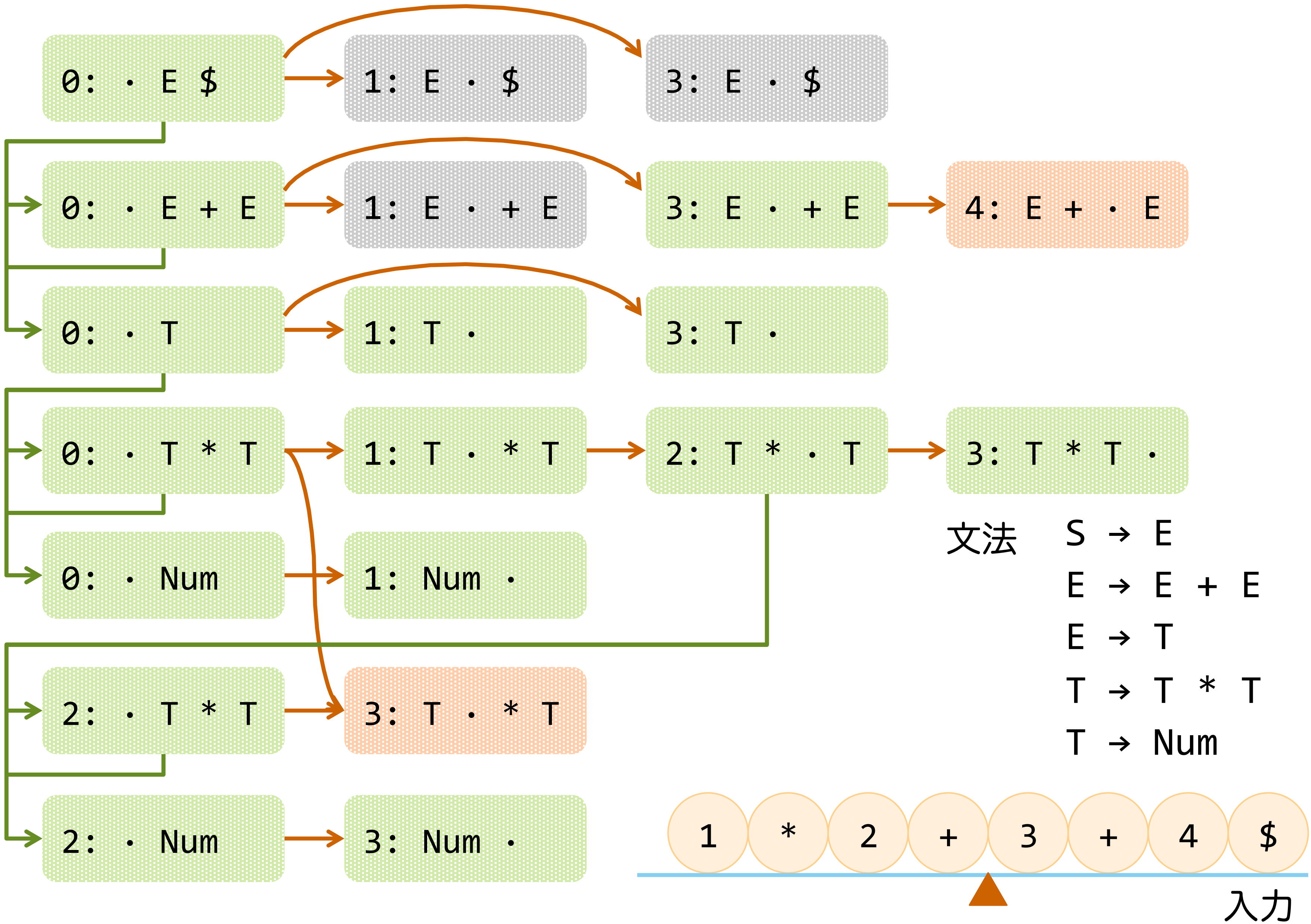

0: · E $
1: E · $
0: · E + E
1: E · + E
0: · T
1: T ·
0: · T * T
1: T · * T
2: T * · T
3: T * T ·
0: · Num
1: Num ·
2: · T * T
3: T · * T
2: · Num
3: Num ·
文法
S → E
E → E + E
E → T
T → T * T
T → Num
1
*
2
+
3
+
4
$
入力



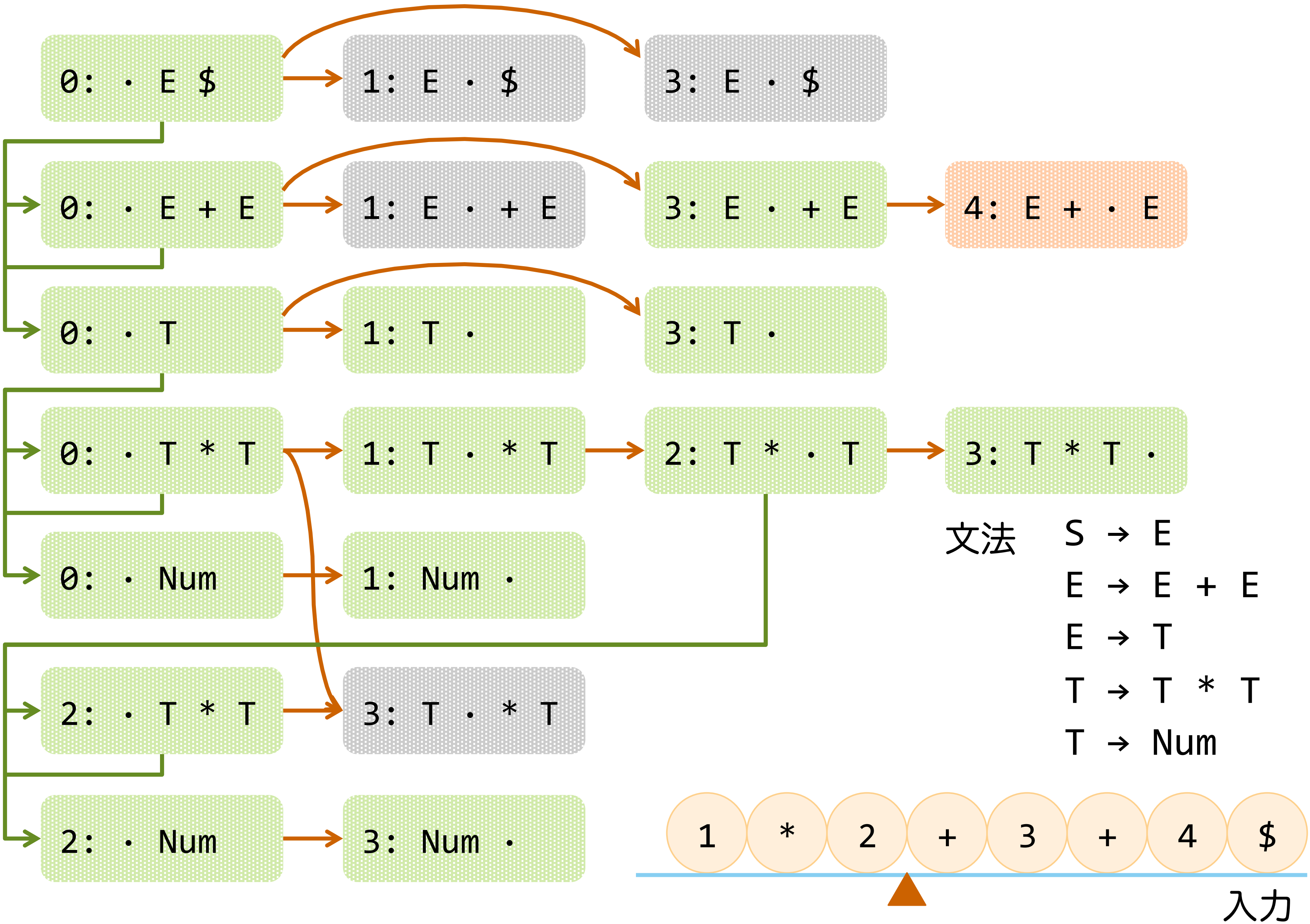
文法 S → E
E → E + E
E → T
T → T * T
T → Num


0: · E $
1: E · $
3: E · $
0: · E + E
1: E · + E
3: E · + E
0: · T
1: T ·
3: T ·
0: · T * T
1: T · * T
2: T * · T
3: T * T ·
0: · Num
1: Num ·
2: · T * T
3: T · * T
2: · Num
3: Num ·
文法
S → E
E → E + E
E → T
T → T * T
T → Num
1 * 2 + 3 + 4 $
入力

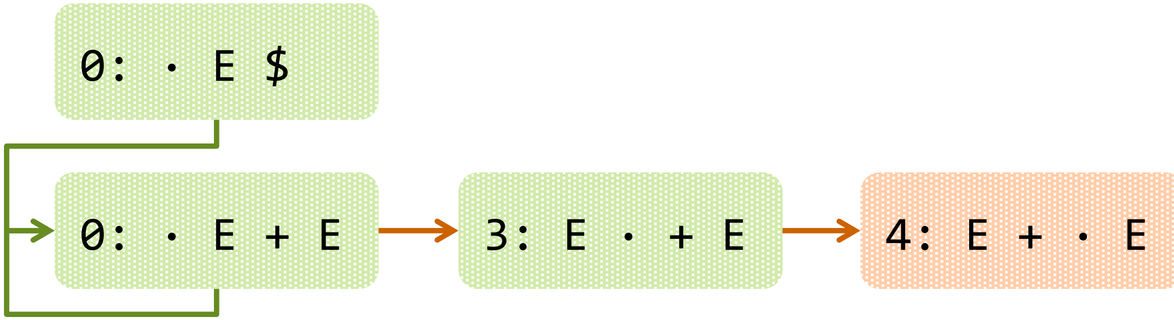

0: · E $
1: E · $
3: E · $
0: · E + E
1: E · + E
3: E · + E
0: · T
1: T ·
3: T ·
0: · T * T
1: T · * T
2: T * · T
3: T * T ·
0: · Num
1: Num ·
2: · T * T
3: T · * T
2: · Num
3: Num ·
文法
S → E
E → E + E
E → T
T → T * T
T → Num
1 * 2 + 3 + 4 $
入力



0: · E $
1: E · $
3: E · $
0: · E + E
1: E · + E
3: E · + E
4: E + · E
0: · T
1: T ·
3: T ·
0: · T * T
1: T · * T
2: T * · T
3: T * T ·
0: · Num
1: Num ·
2: · T * T
3: T · * T
2: · Num
3: Num ·
文法
S → E
E → E + E
E → T
T → T * T
T → Num
1
*
2
+
3
+
4
$
入力

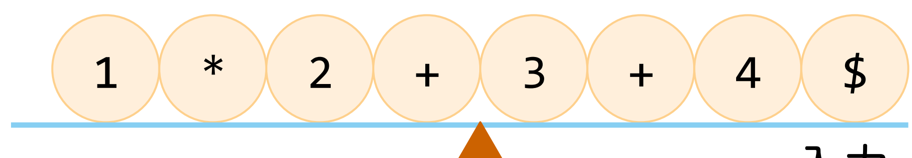


0: · E $
1: E · $
3: E · $
0: · E + E
1: E · + E
3: E · + E
4: E + · E
0: · T
1: T ·
3: T ·
0: · T * T
1: T · * T
2: T * · T
3: T * T ·
0: · Num
1: Num ·
2: · T * T
3: T · * T
2: · Num
3: Num ·
文法
S → E
E → E + E
E → T
T → T * T
T → Num
1 * 2 + 3 + 4 $
入力

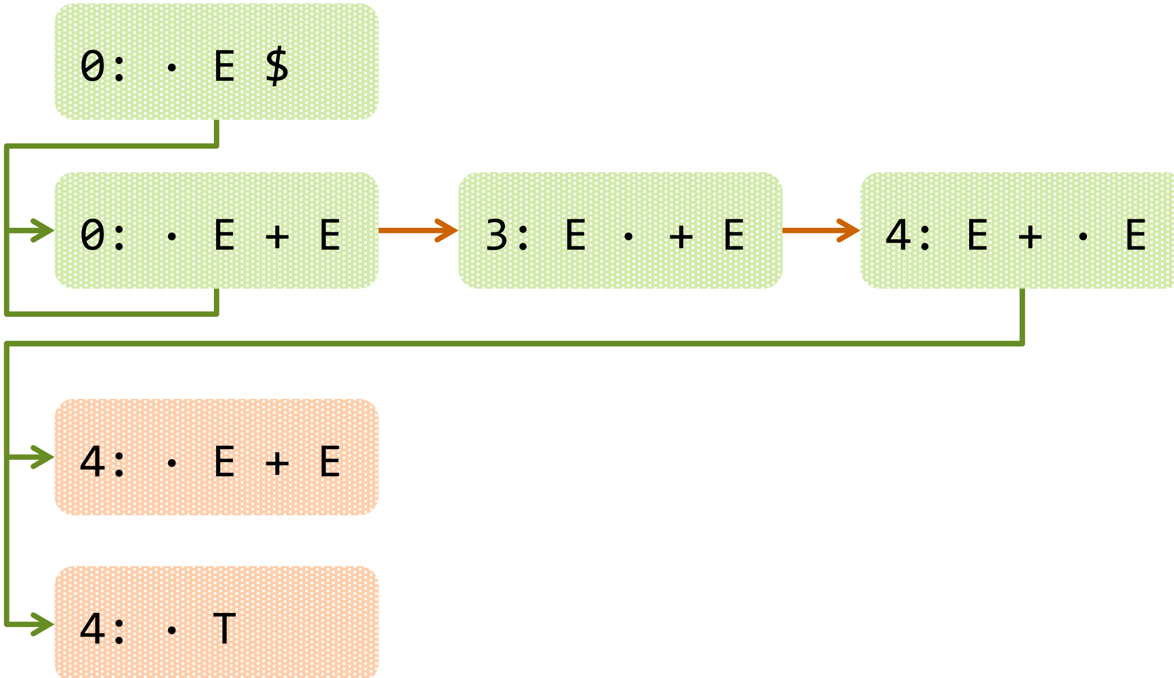


文法

S → E
E → E + E
E → T
T → T * T
T → Num



入力



文法

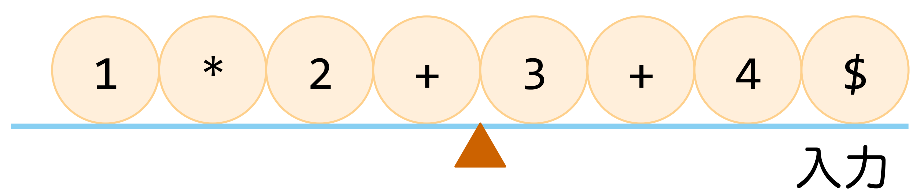
$S \to E$

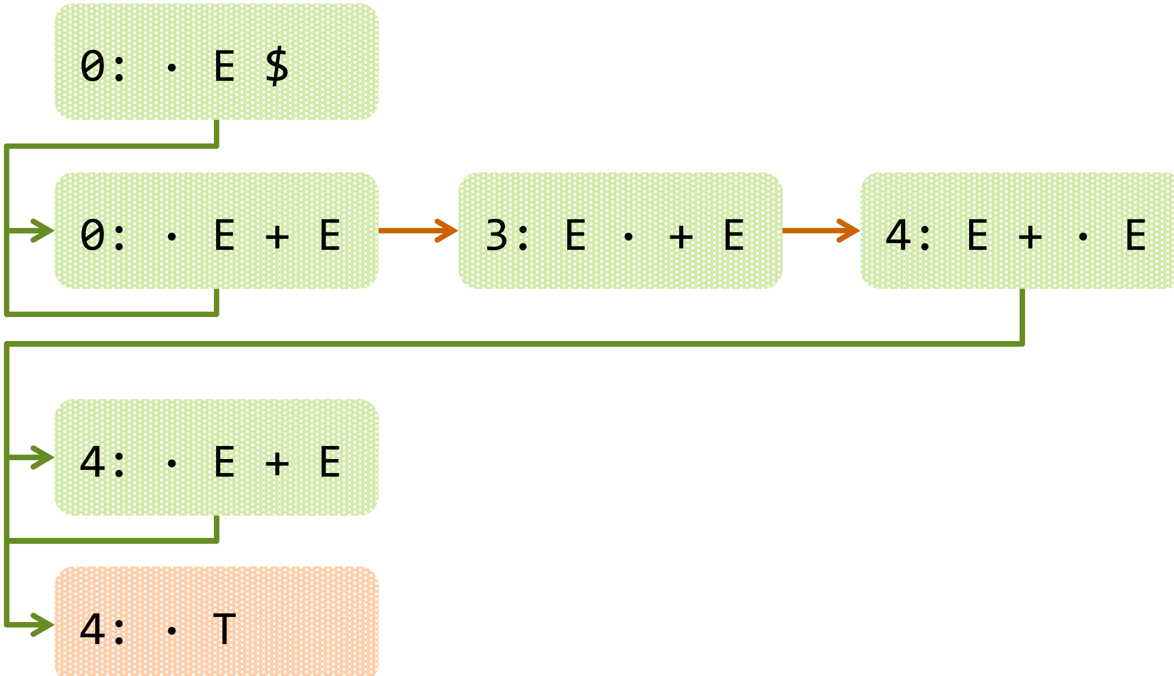
$E \to E + E$

$E \to T$

$T \to T * T$

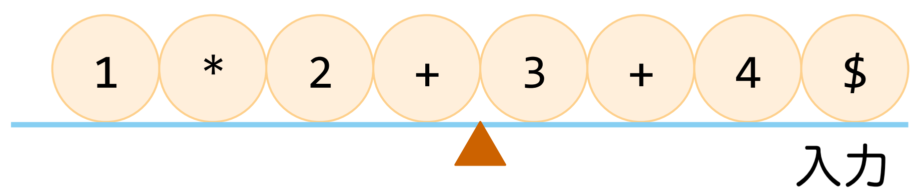
$T \to Num$

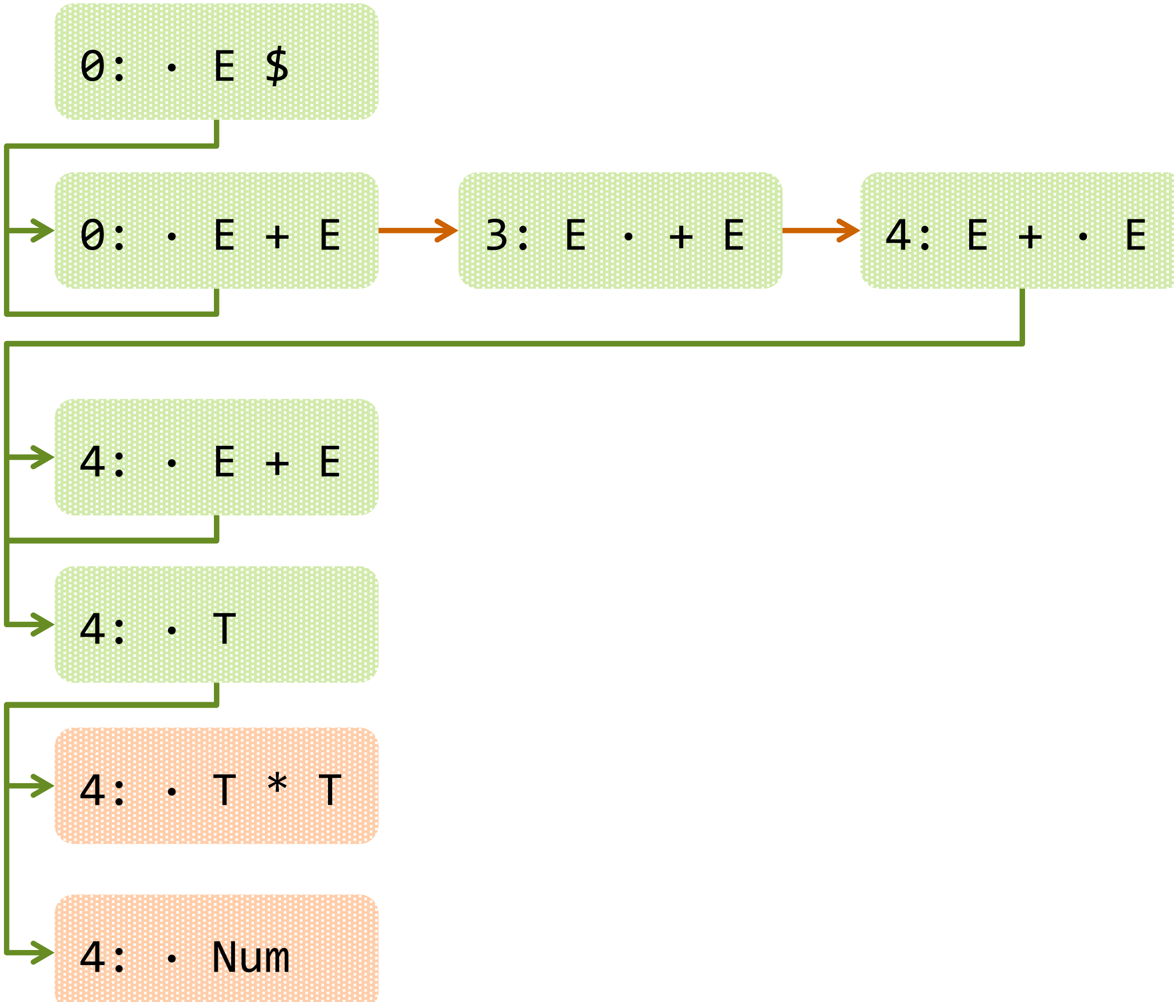




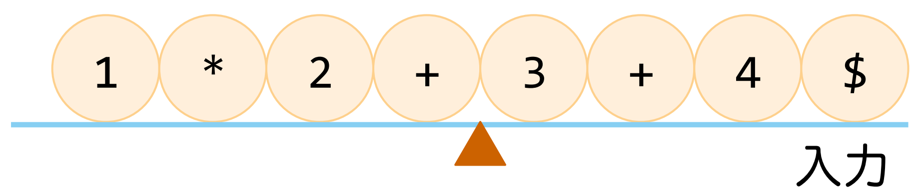
文法

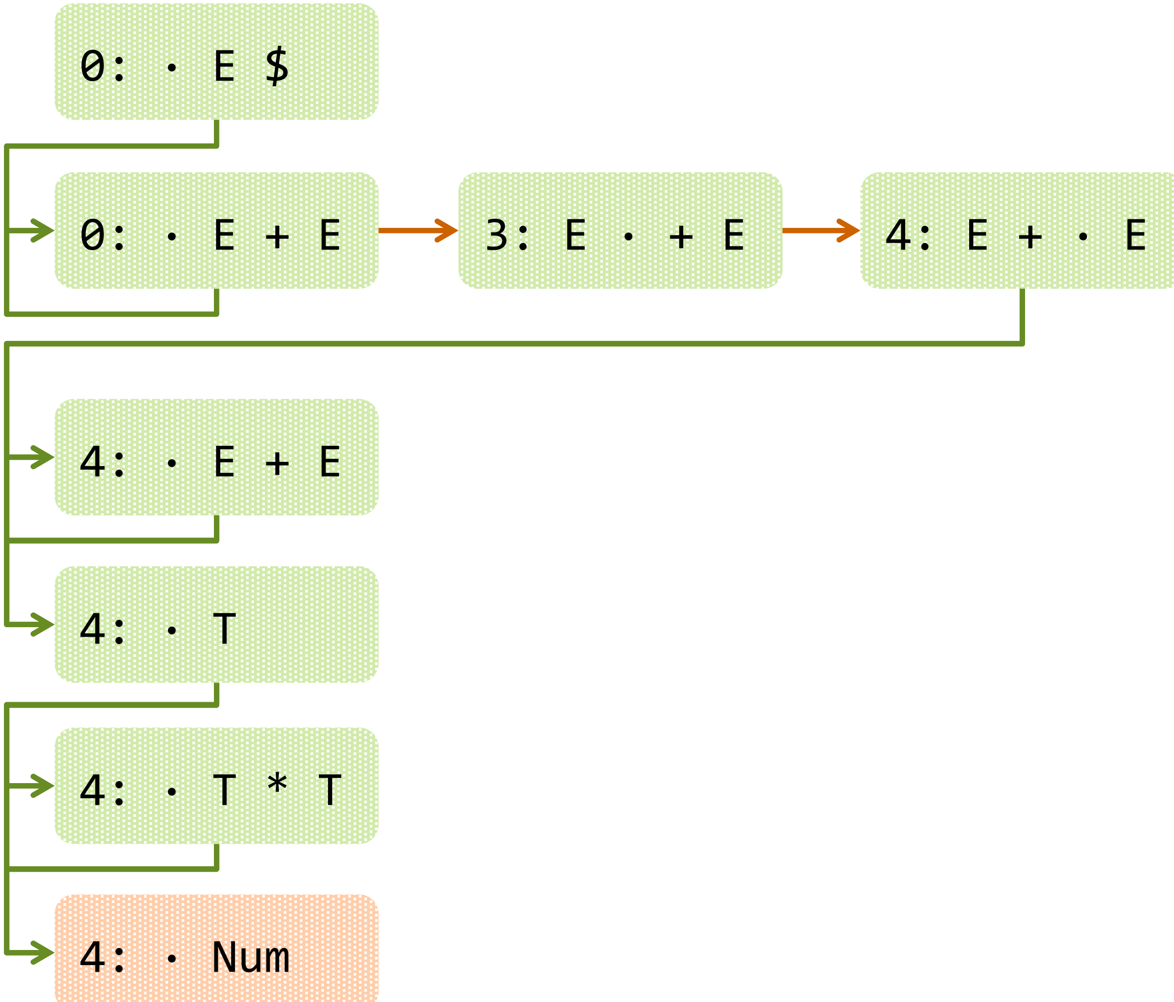
S → E
E → E + E
E → T
T → T * T
T → Num





文法 S → E
E → E + E
E → T
T → T * T
T → Num





文法

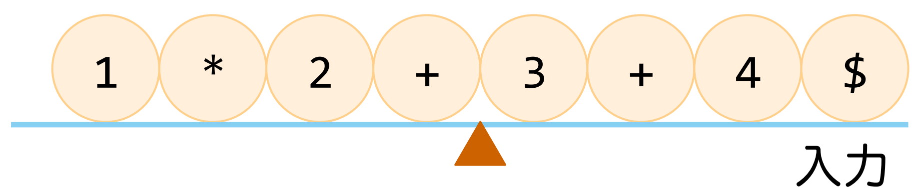
$S \to E$

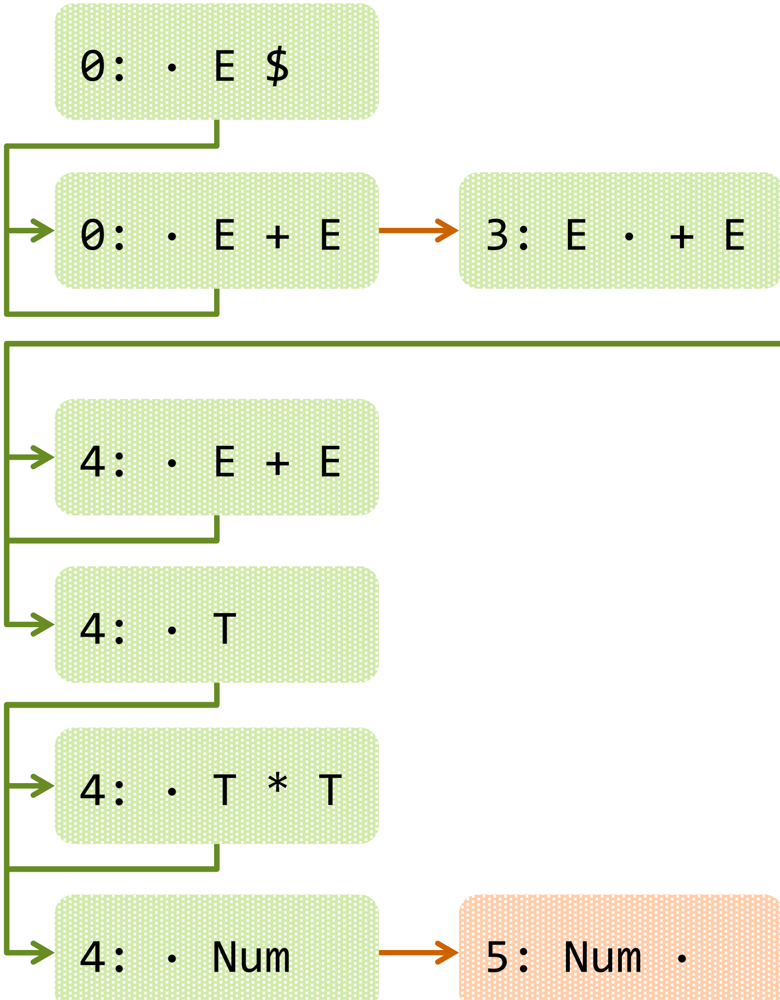
$E \to E + E$

$E \to T$

$T \to T * T$

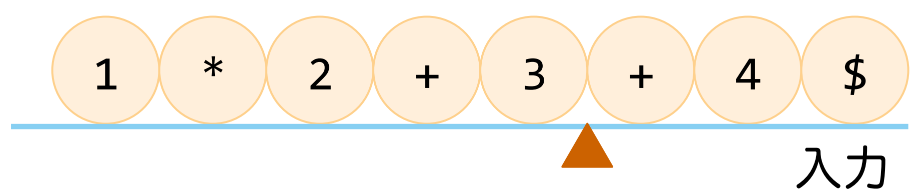
$T \to Num$

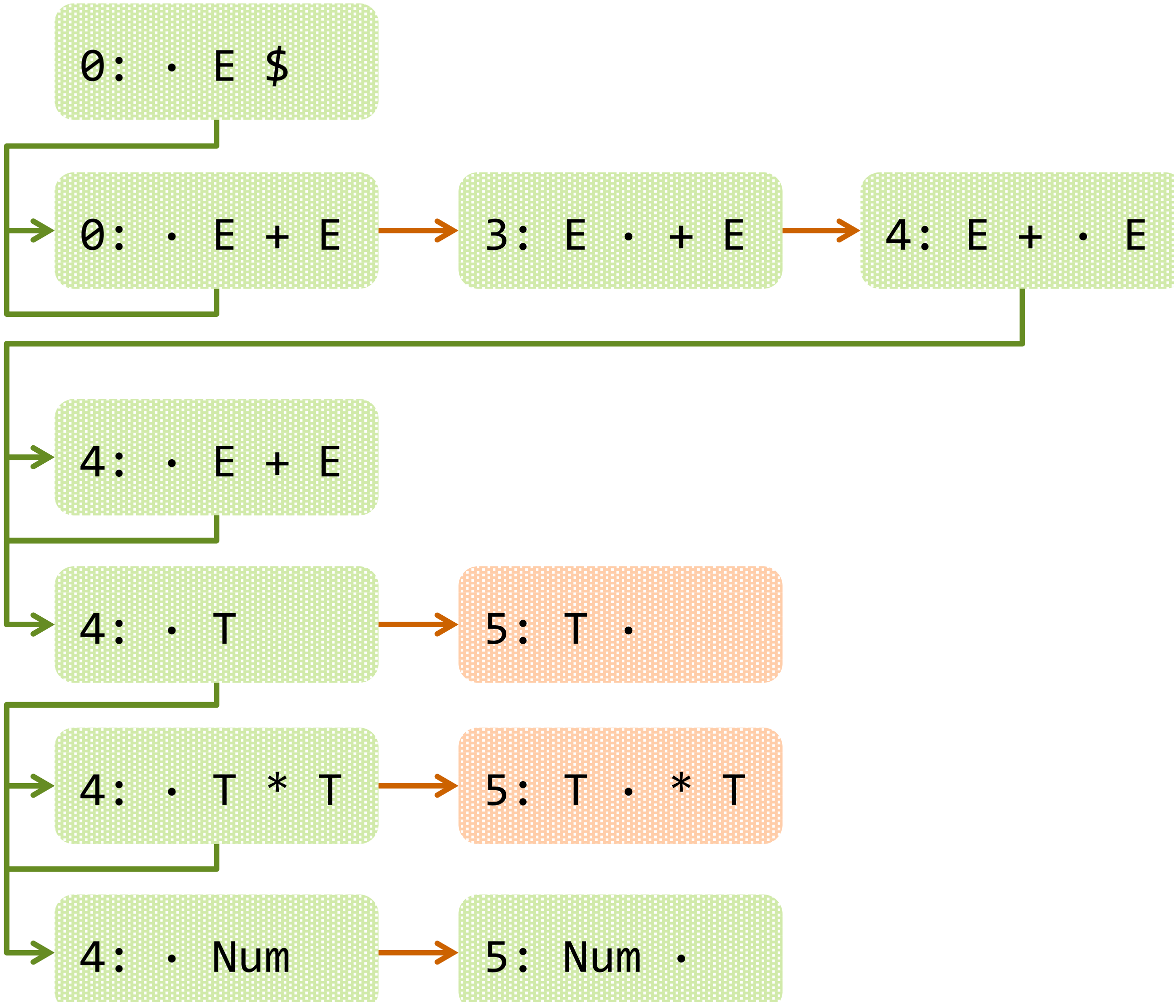




文法

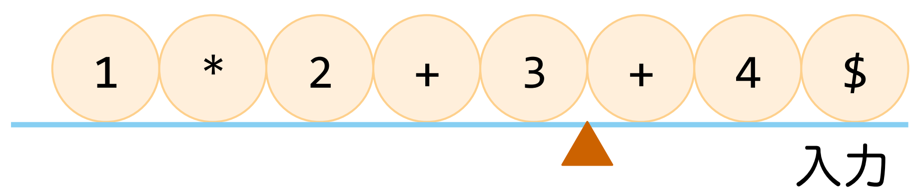
S → E
E → E + E
E → T
T → T * T
T → Num

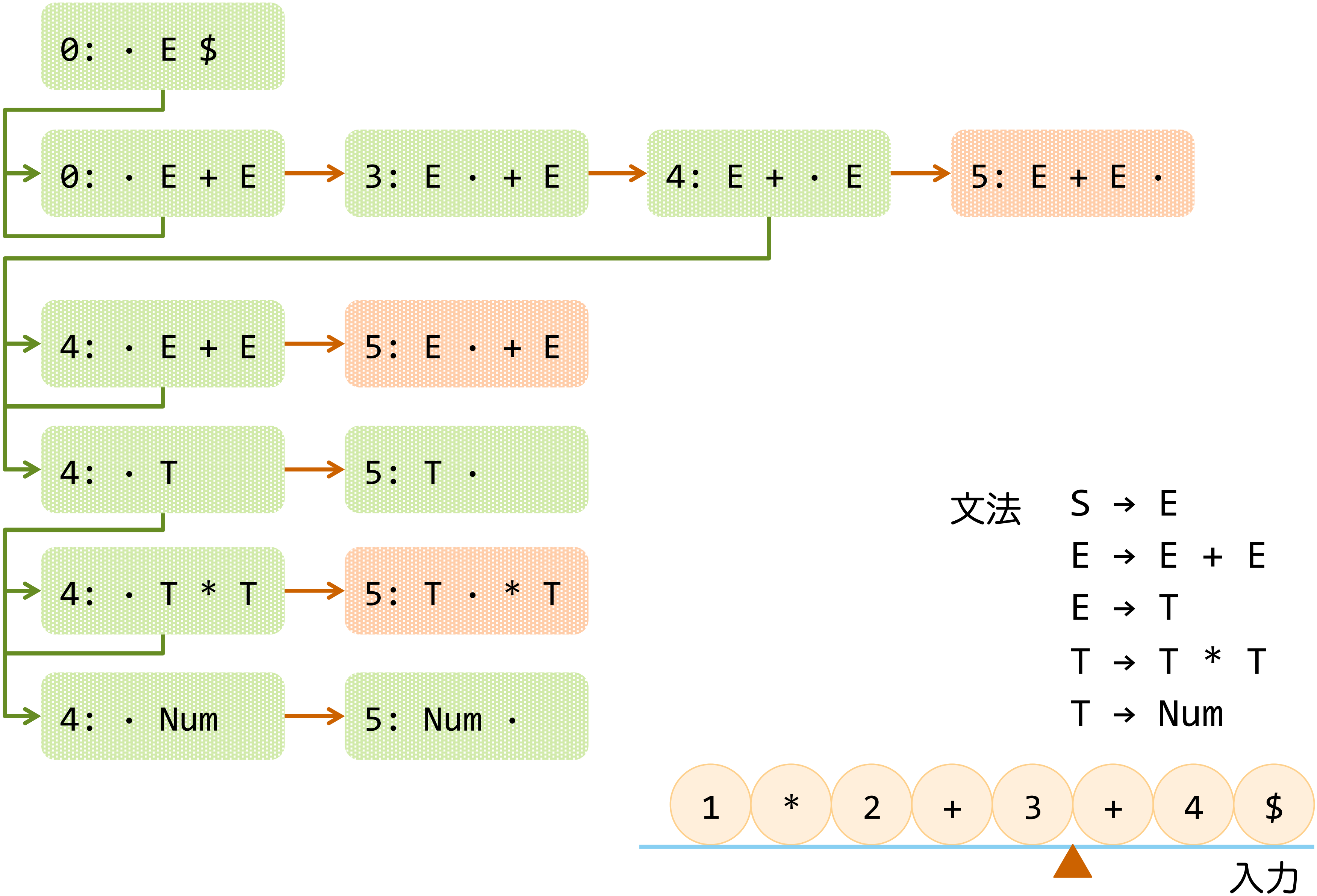




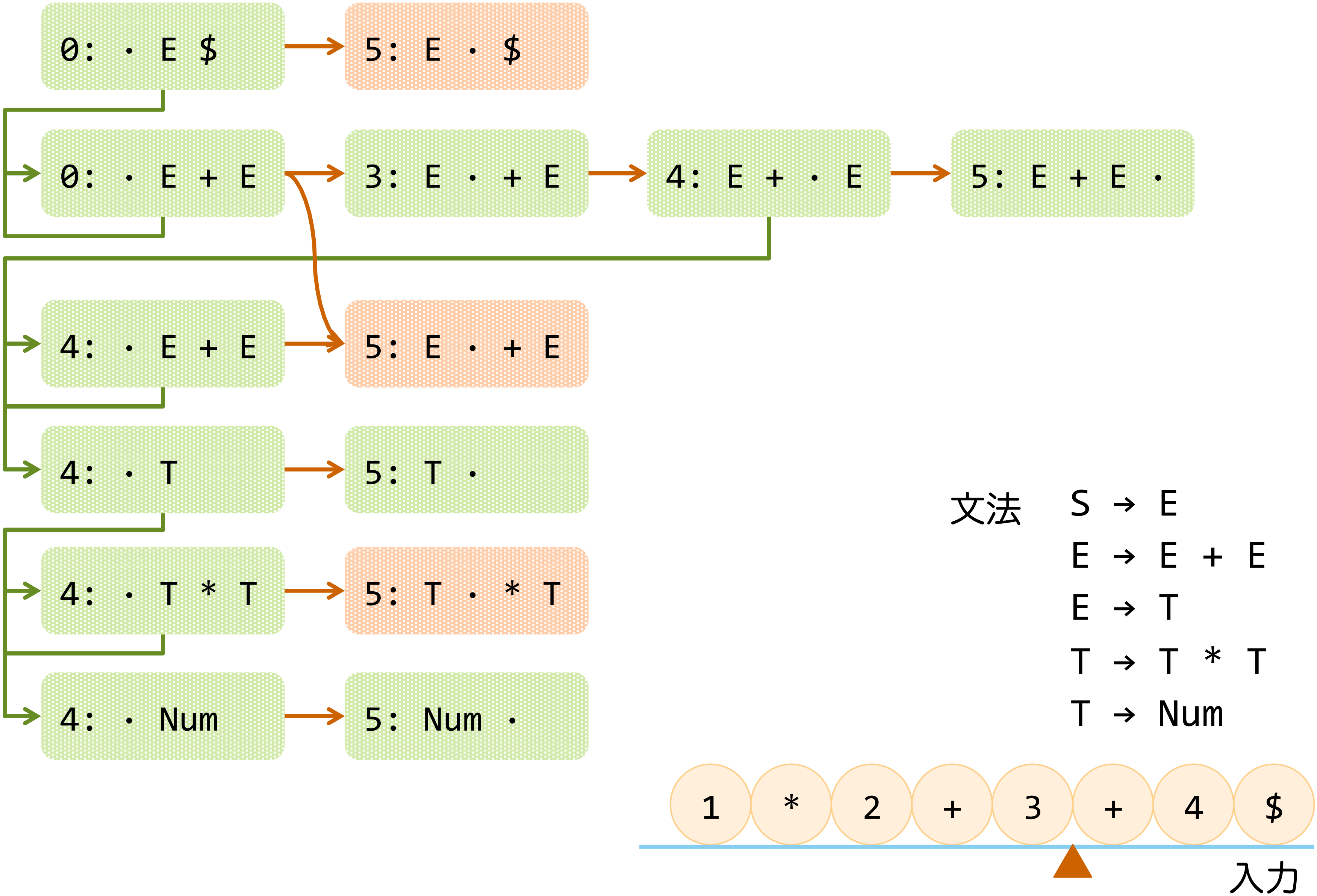
文法

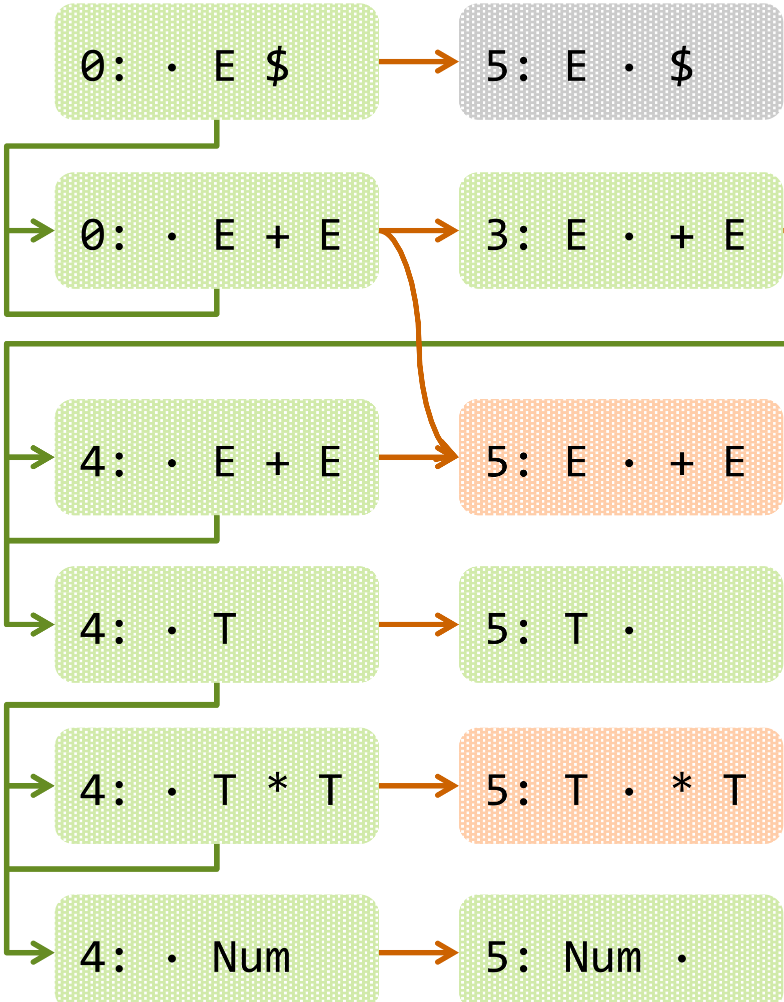
$$
\begin{aligned}
S &\rightarrow E \\
E &\rightarrow E + E \\
E &\rightarrow T \\
T &\rightarrow T * T \\
T &\rightarrow \mathrm{Num}
\end{aligned}
$$



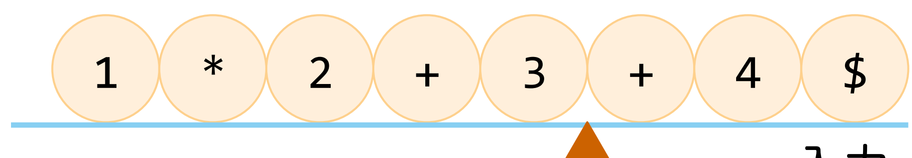


0: · E $
0: · E + E
3: E · + E
4: E + · E
5: E + E ·
4: · E + E
5: E · + E
4: · T
5: T ·
4: · T * T
5: T · * T
4: · Num
5: Num ·
文法
S → E
E → E + E
E → T
T → T * T
T → Num
1
*
2
+
3
+
4
$
入力

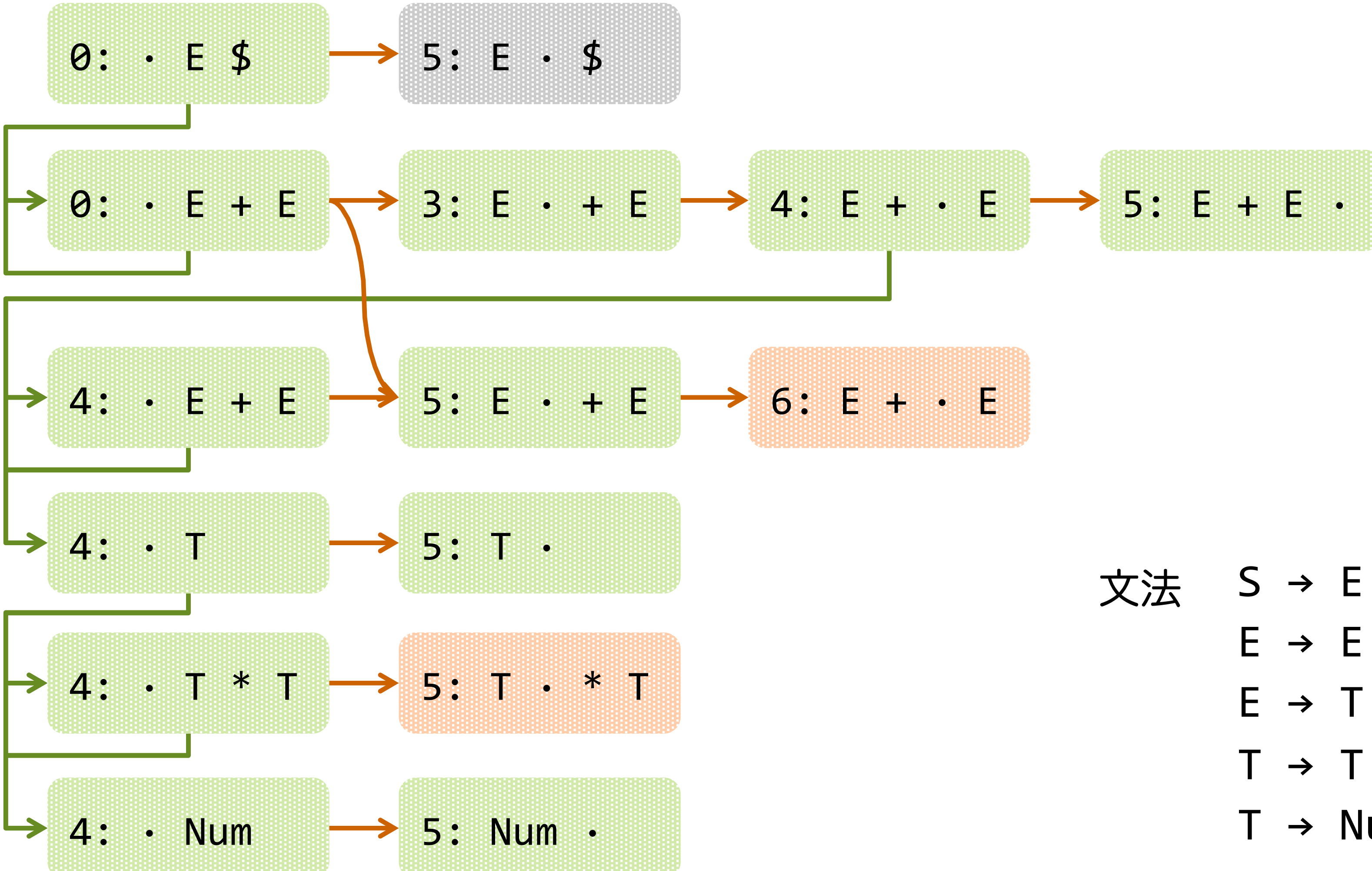

0: · E $
5: E · $
0: · E + E
3: E · + E
4: E + · E
5: E + E ·
4: · E + E
5: E · + E
4: · T
5: T ·
4: · T * T
5: T · * T
4: · Num
5: Num ·
文法
S → E
E → E + E
E → T
T → T * T
T → Num
1 * 2 + 3 + 4 $
入力



文法　S → E
E → E + E
E → T
T → T * T
T → Num

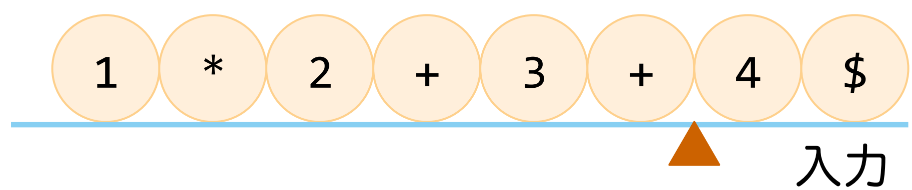


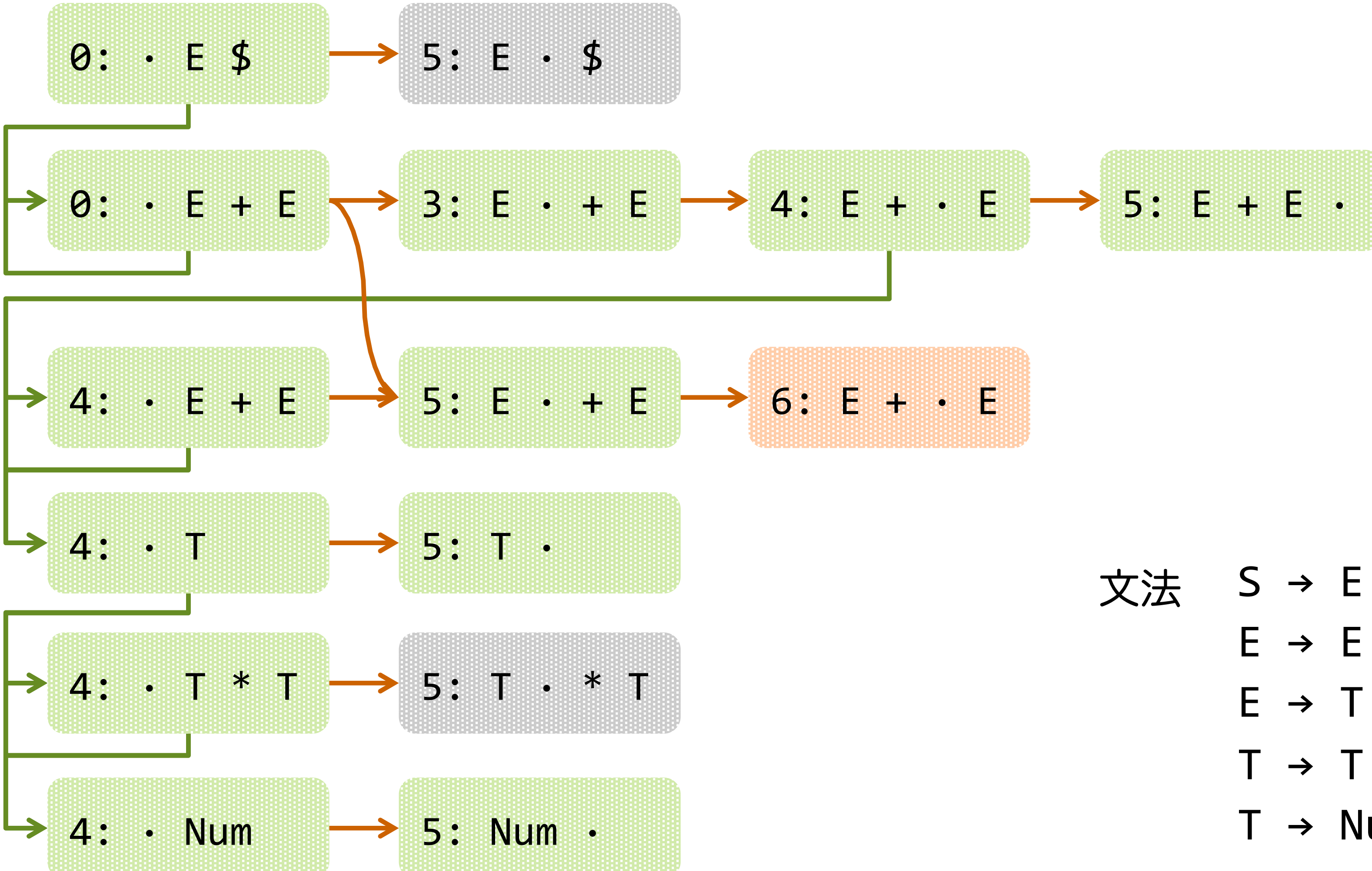
入力



文法

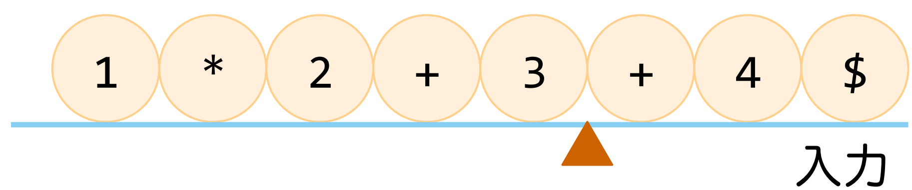
S → E
E → E + E
E → T
T → T * T
T → Num

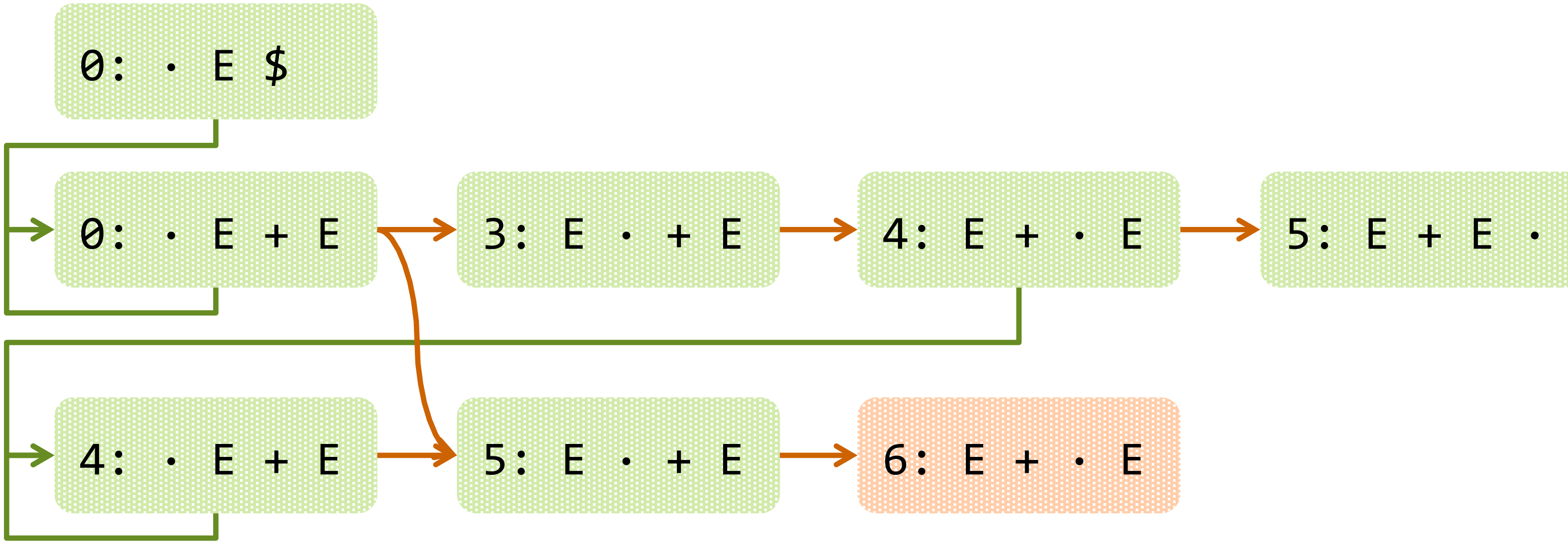




文法

S → E
E → E + E
E → T
T → T * T
T → Num





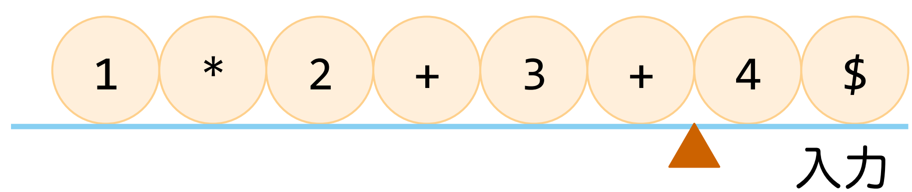
文法 S → E

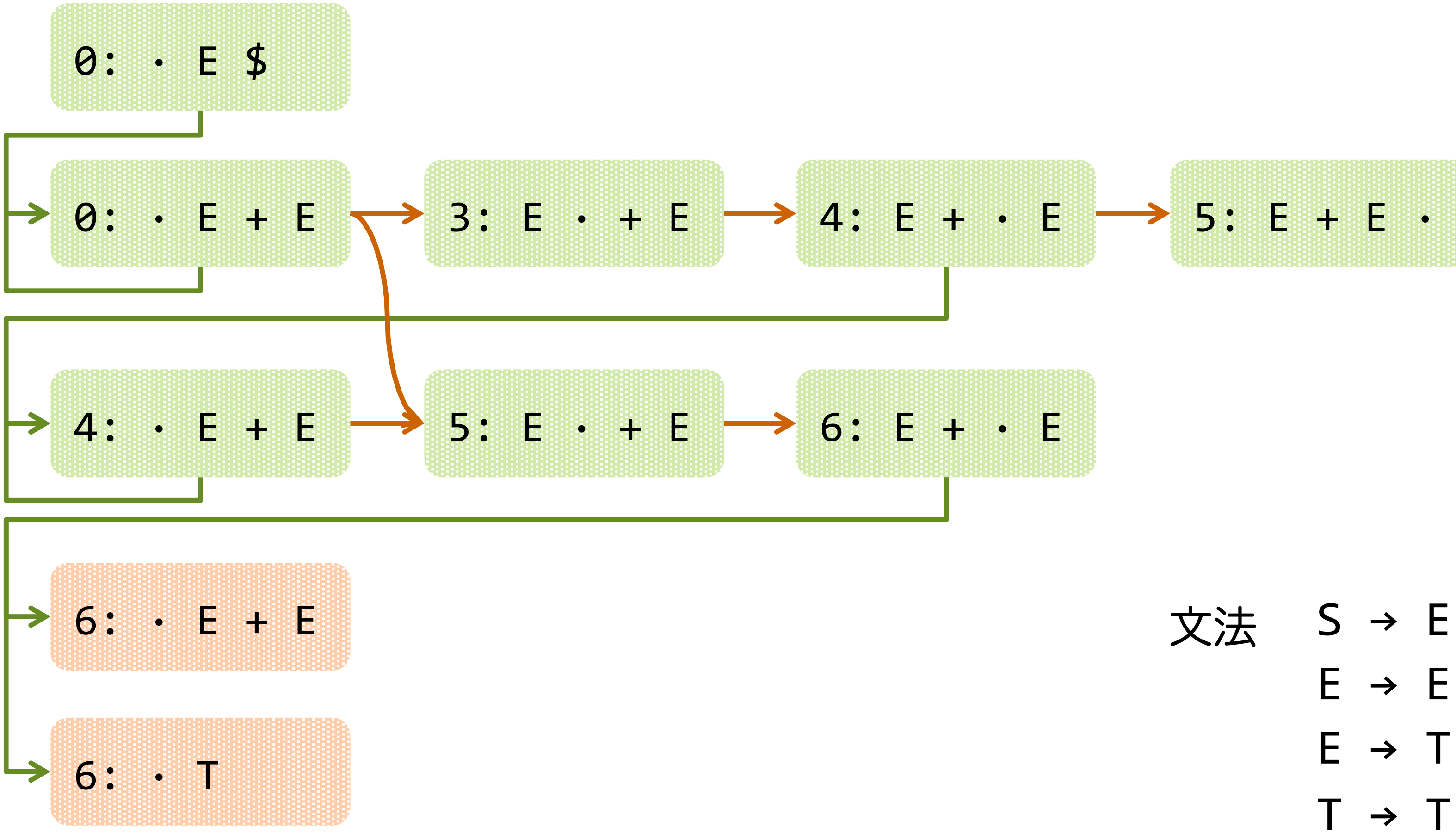
E → E + E

E → T

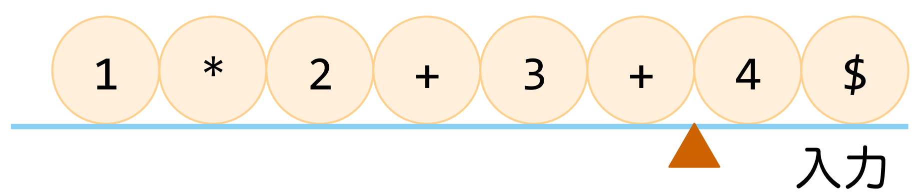
T → T * T

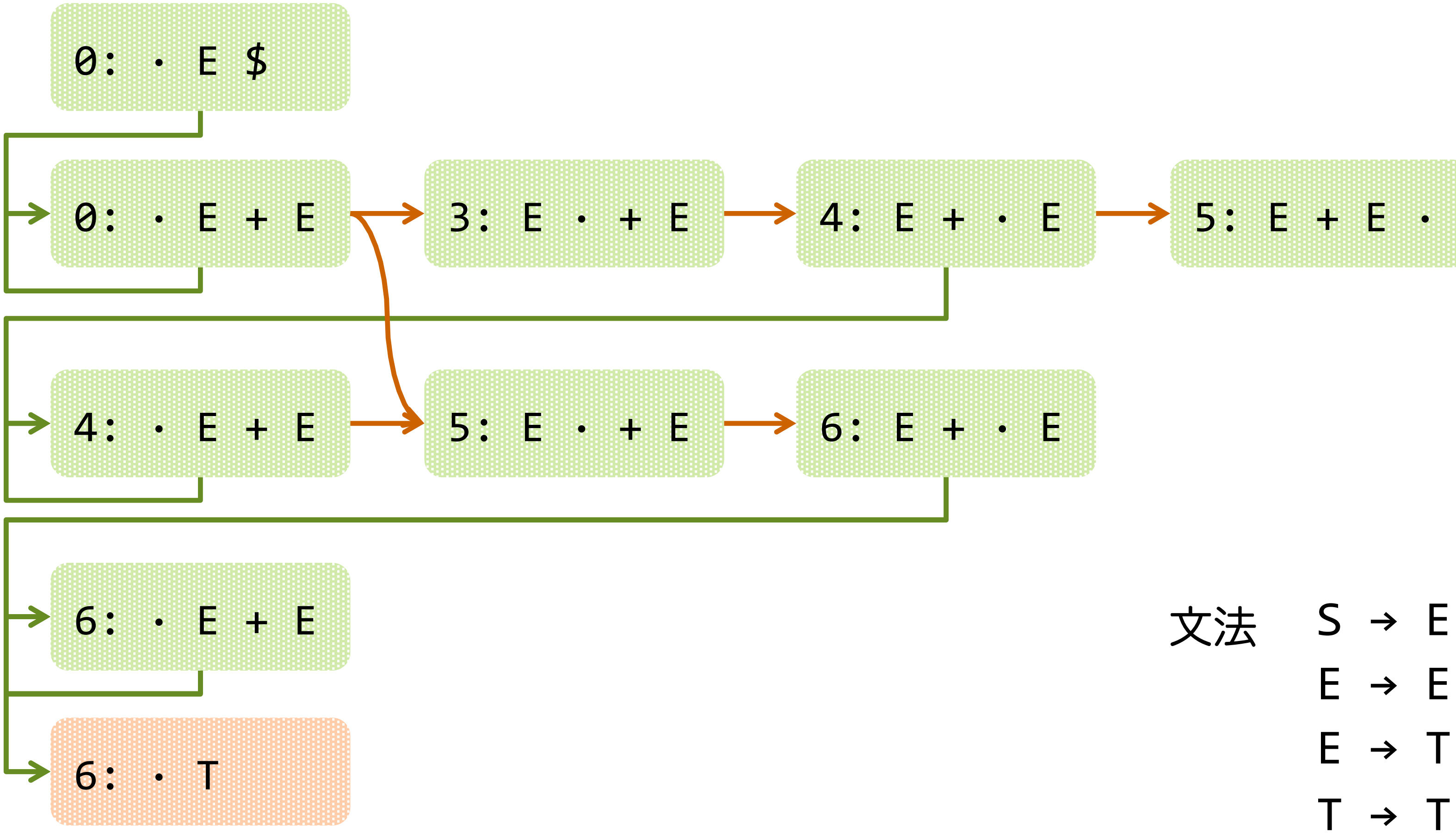
T → Num



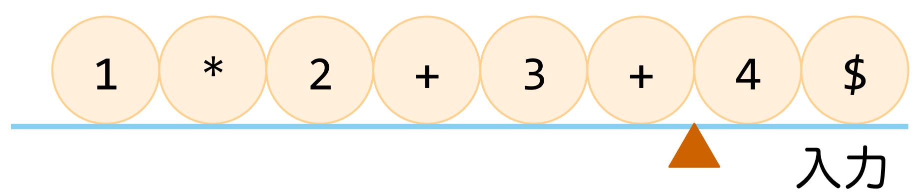


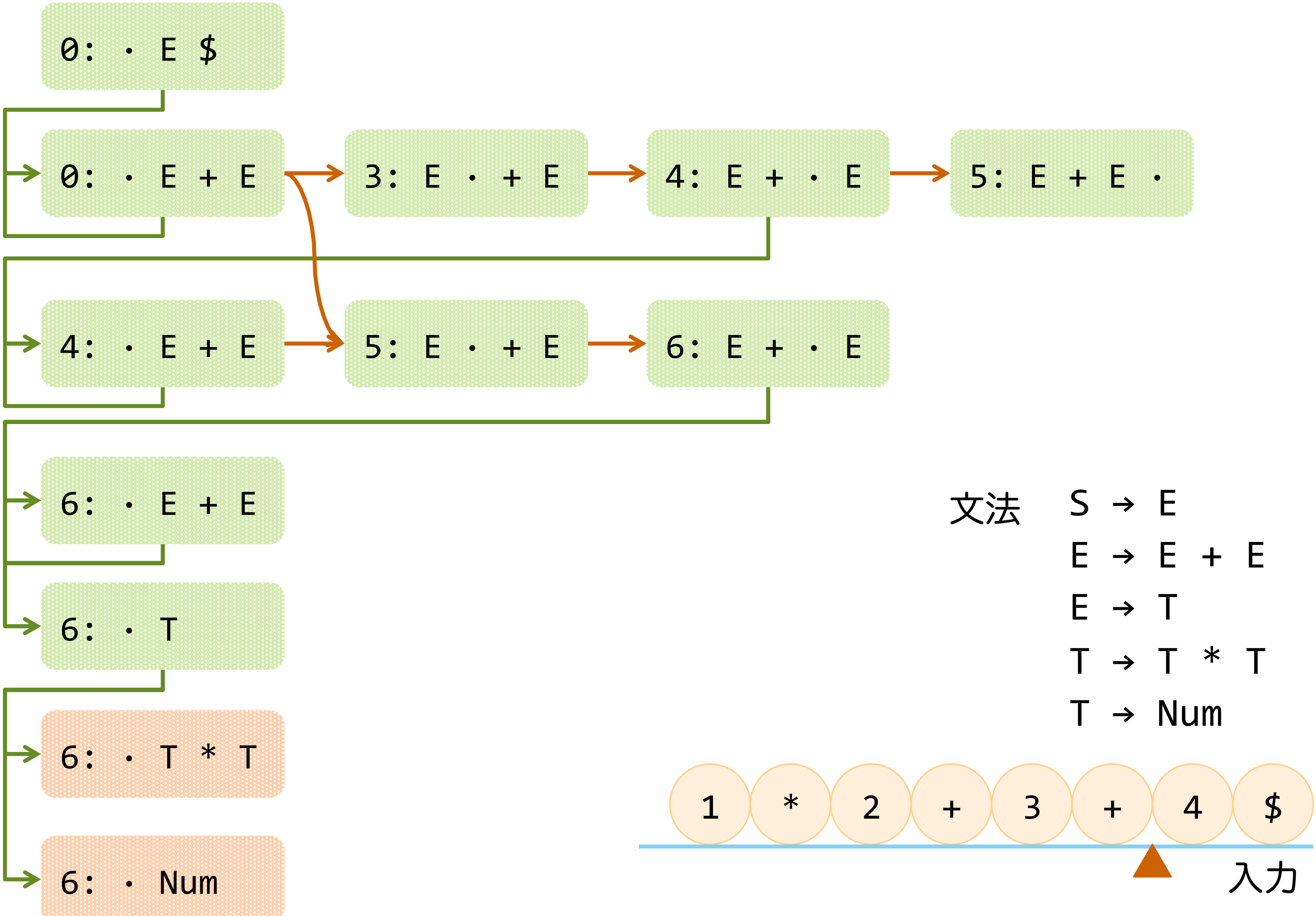
文法 S → E
E → E + E
E → T
T → T * T
T → Num

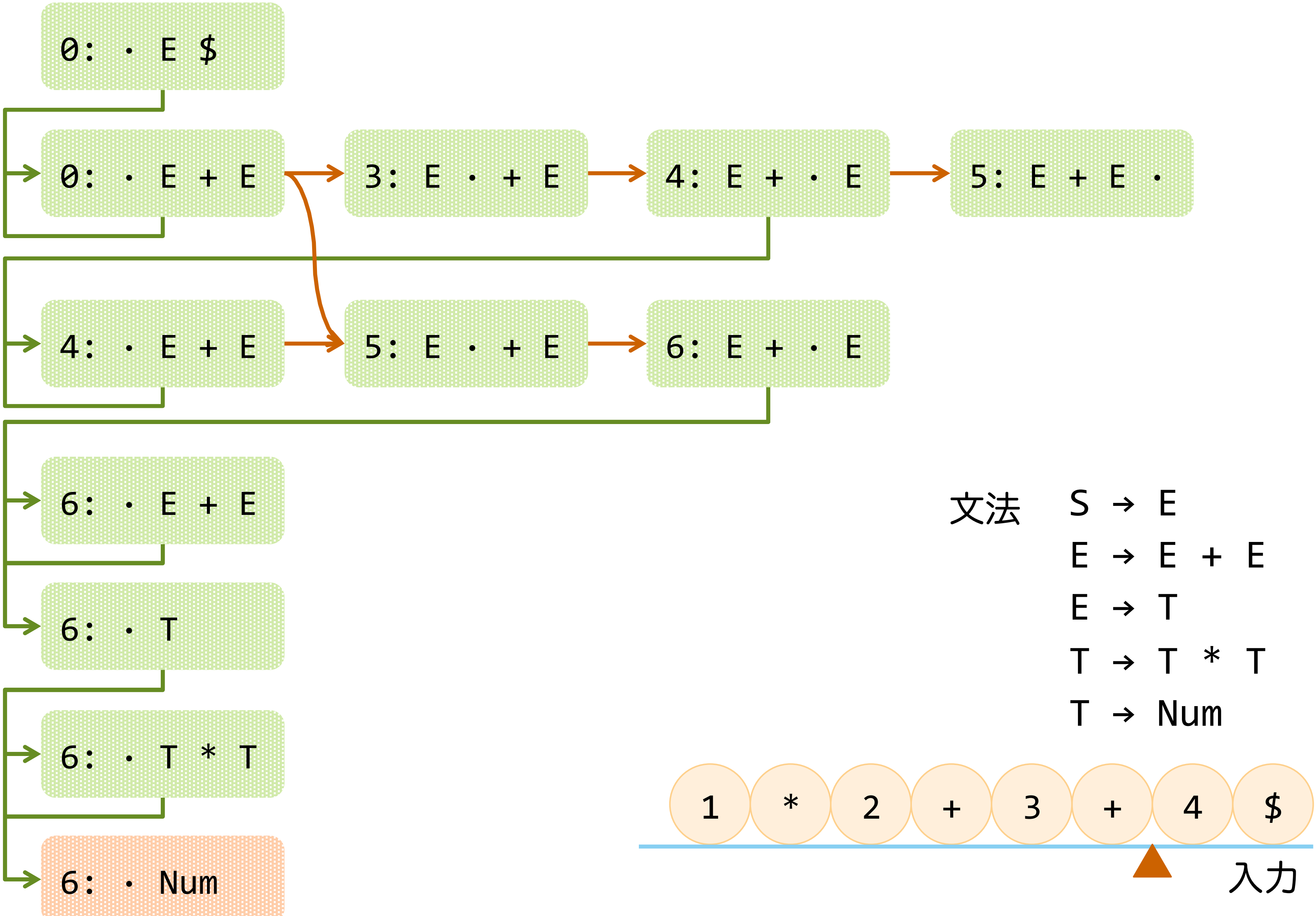




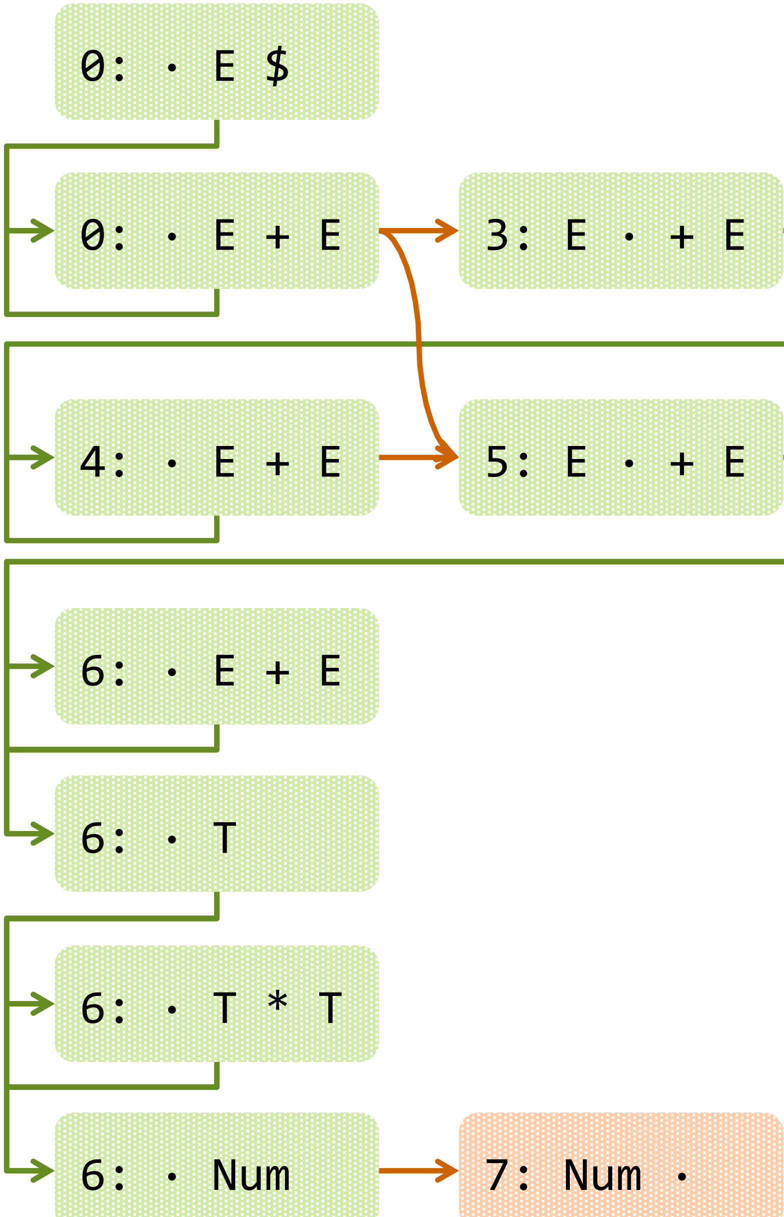
文法
S → E
E → E + E
E → T
T → T * T
T → Num




0: · E $
0: · E + E
3: E · + E
4: E + · E
5: E + E ·
4: · E + E
5: E · + E
6: E + · E
6: · E + E
6: · T
6: · T * T
6: · Num
文法
S → E
E → E + E
E → T
T → T * T
T → Num
1 * 2 + 3 + 4 $
入力



0: · E $
0: · E + E
3: E · + E
4: E + · E
5: E + E ·
4: · E + E
5: E · + E
6: E + · E
6: · E + E
6: · T
6: · T * T
6: · Num
文法
S → E
E → E + E
E → T
T → T * T
T → Num
1
*
2
+
3
+
4
$
入力

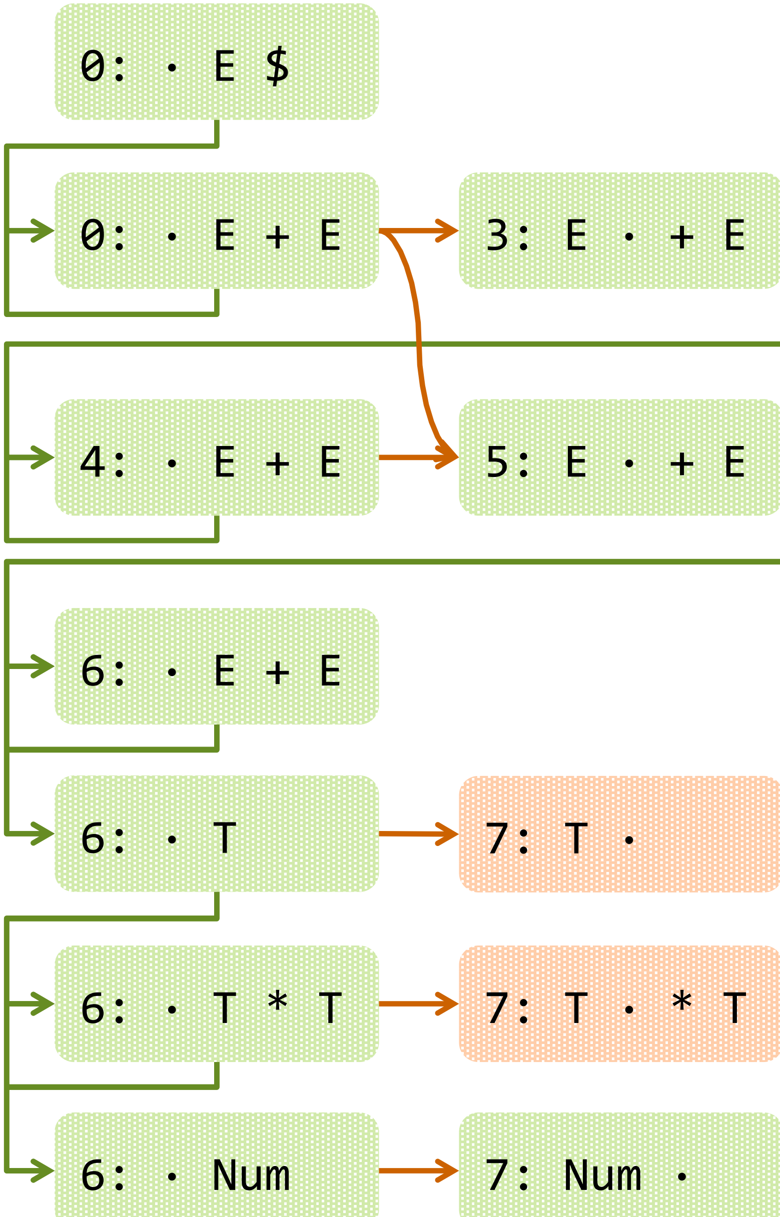


文法

$$
\begin{aligned}
S &\to E \\
E &\to E + E \\
E &\to T \\
T &\to T * T \\
T &\to Num
\end{aligned}
$$

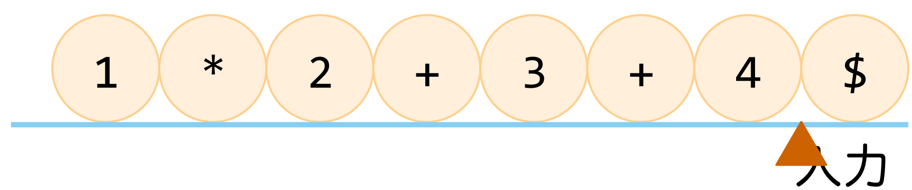
1	*	2	+	3	+	4	$

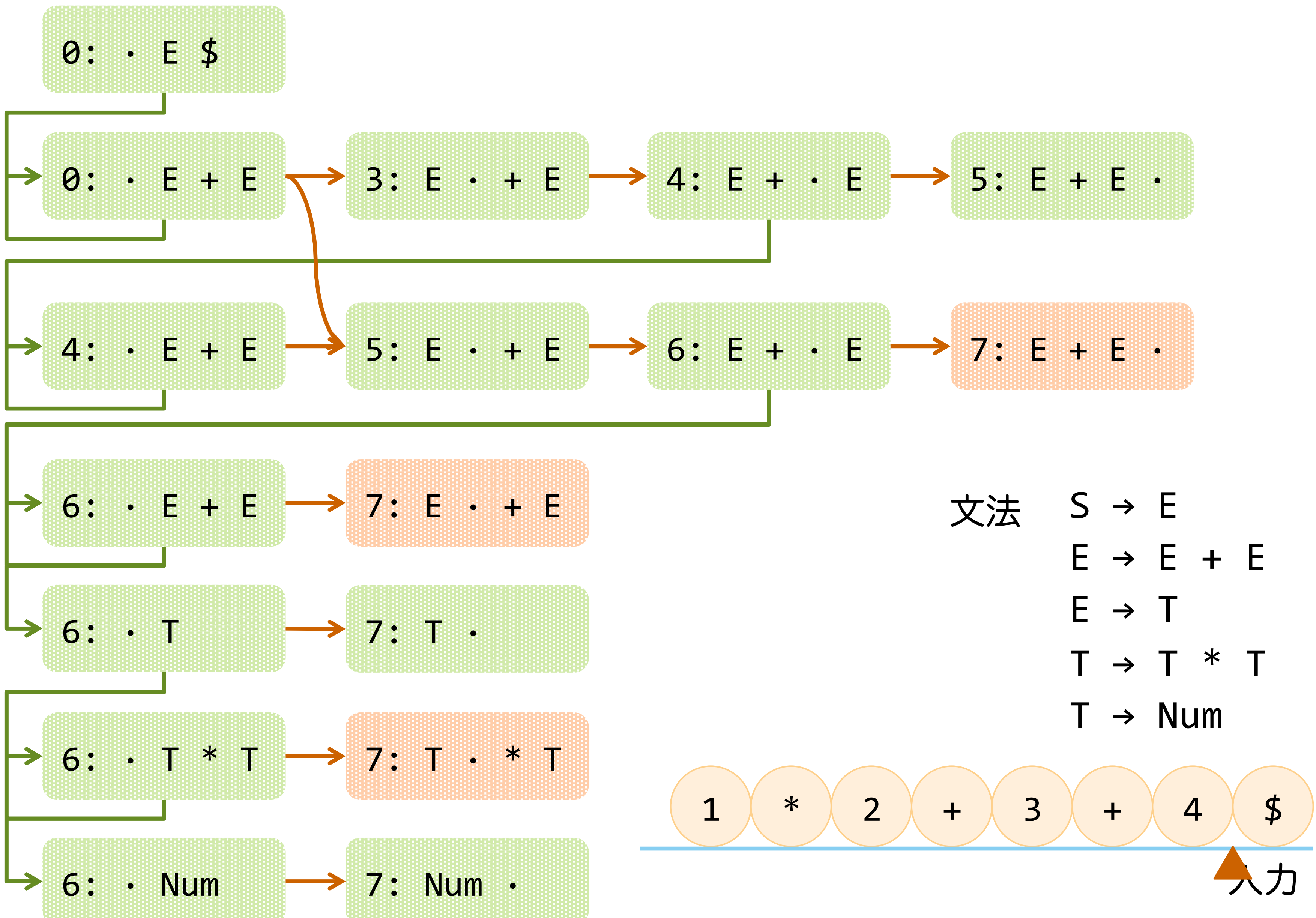
入力



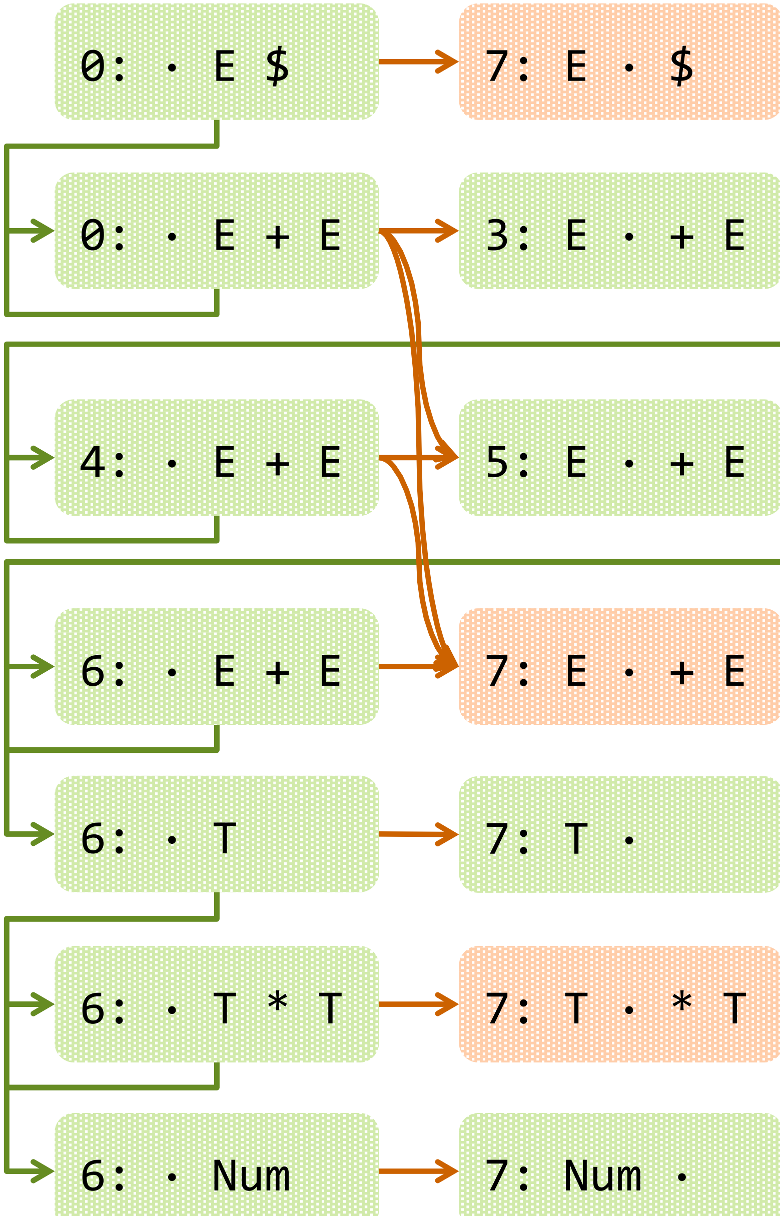
文法

S → E
E → E + E
E → T
T → T * T
T → Num

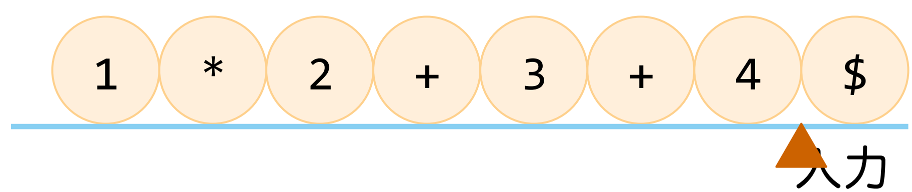


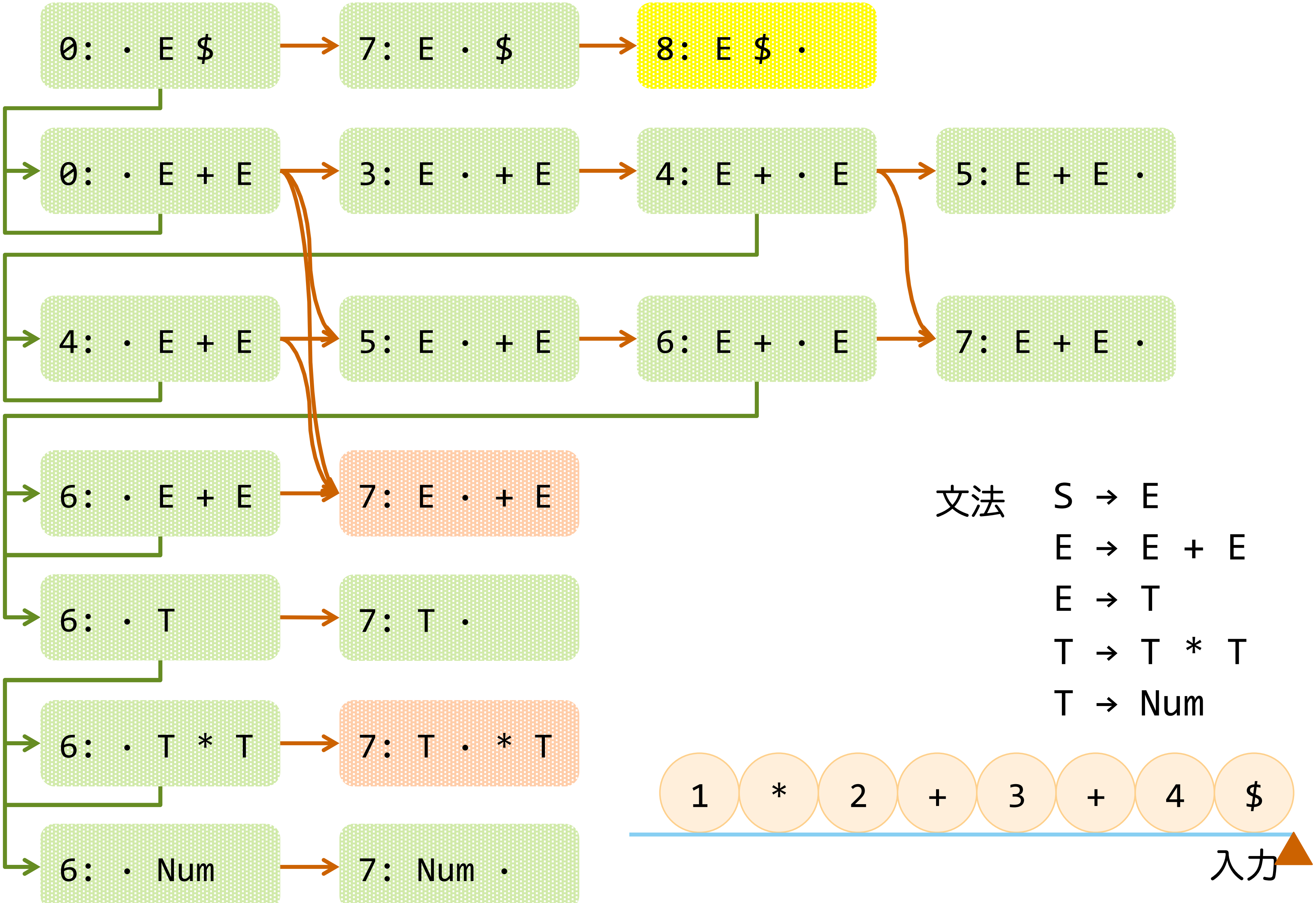


文法
$$
\begin{aligned}
S &\rightarrow E \\
E &\rightarrow E + E \\
E &\rightarrow T \\
T &\rightarrow T * T \\
T &\rightarrow \mathrm{Num}
\end{aligned}
$$

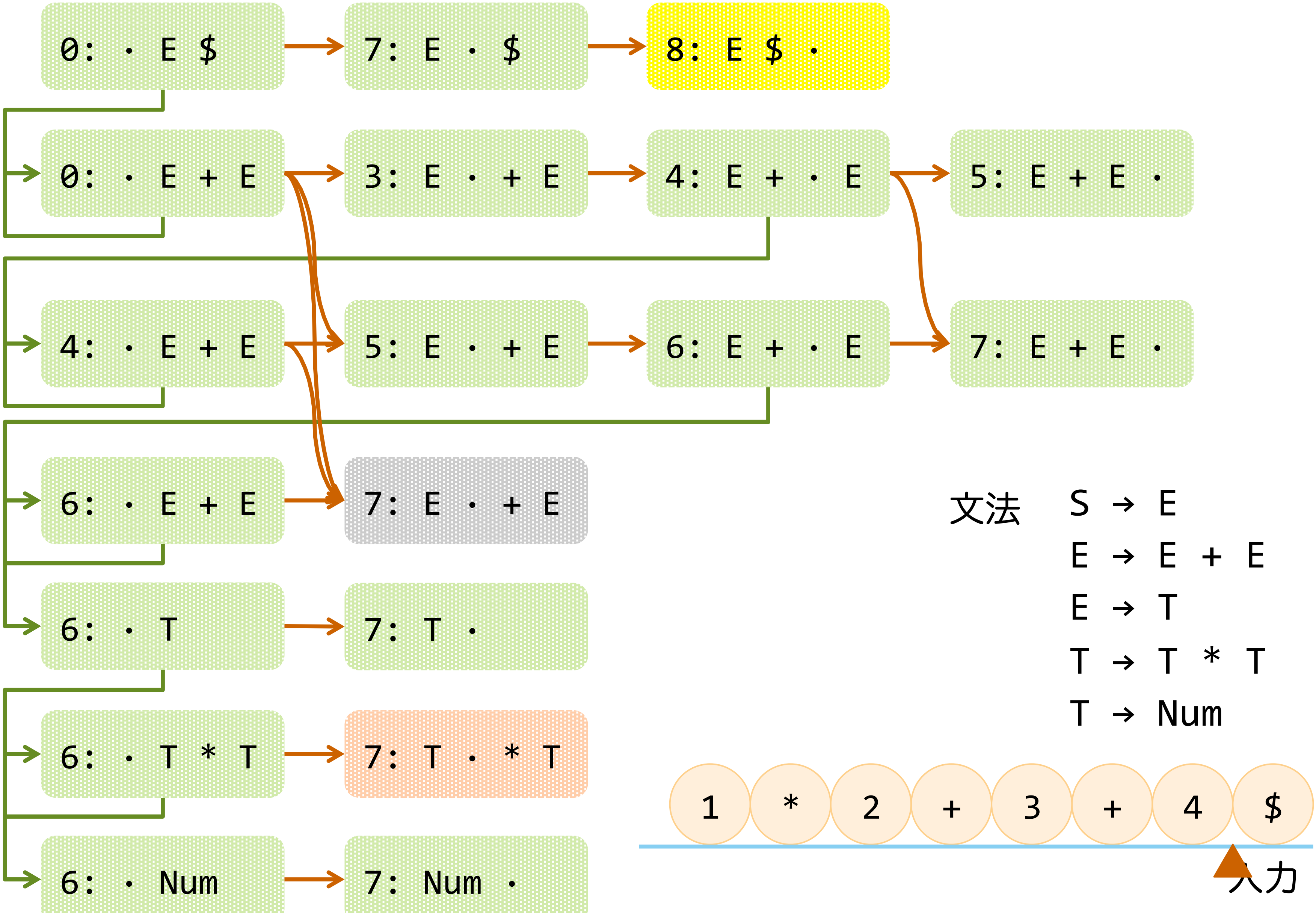


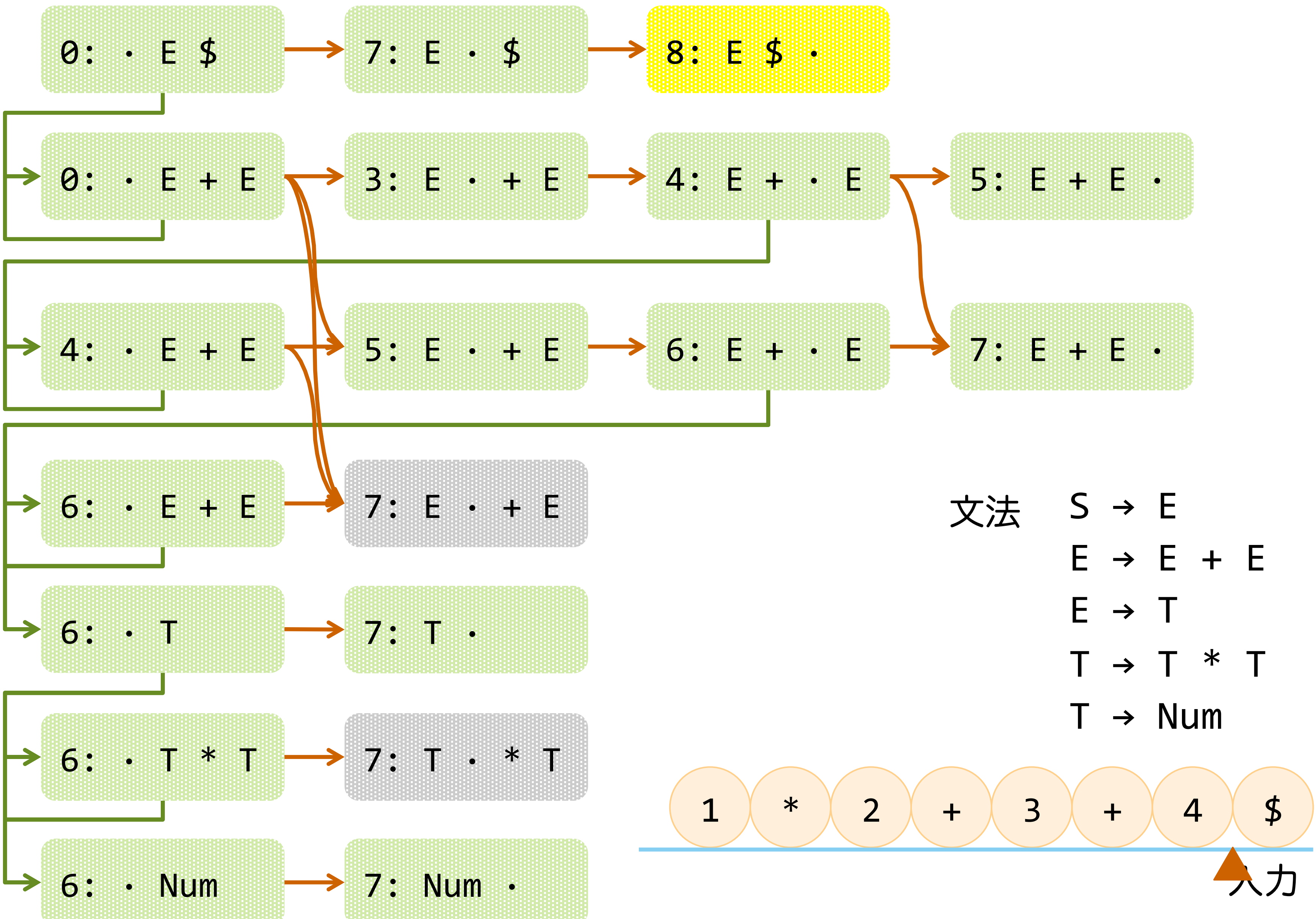
文法 S → E
E → E + E
E → T
T → T * T
T → Num





0: · E $
7: E · $
8: E $ ·
0: · E + E
3: E · + E
4: E + · E
5: E + E ·
4: · E + E
5: E · + E
6: E + · E
7: E + E ·
6: · E + E
7: E · + E
6: · T
7: T ·
6: · T * T
7: T · * T
6: · Num
7: Num ·
文法
S → E
E → E + E
E → T
T → T * T
T → Num
1 * 2 + 3 + 4 $
入力


0: · E $
7: E · $
8: E $ ·
0: · E + E
3: E · + E
4: E + · E
5: E + E ·
4: · E + E
5: E · + E
6: E + · E
7: E + E ·
6: · E + E
7: E · + E
6: · T
7: T ·
6: · T * T
7: T · * T
6: · Num
7: Num ·
文法
S → E
E → E + E
E → T
T → T * T
T → Num
1 * 2 + 3 + 4 $
入力



文法 S → E
E → E + E
E → T
T → T * T
T → Num

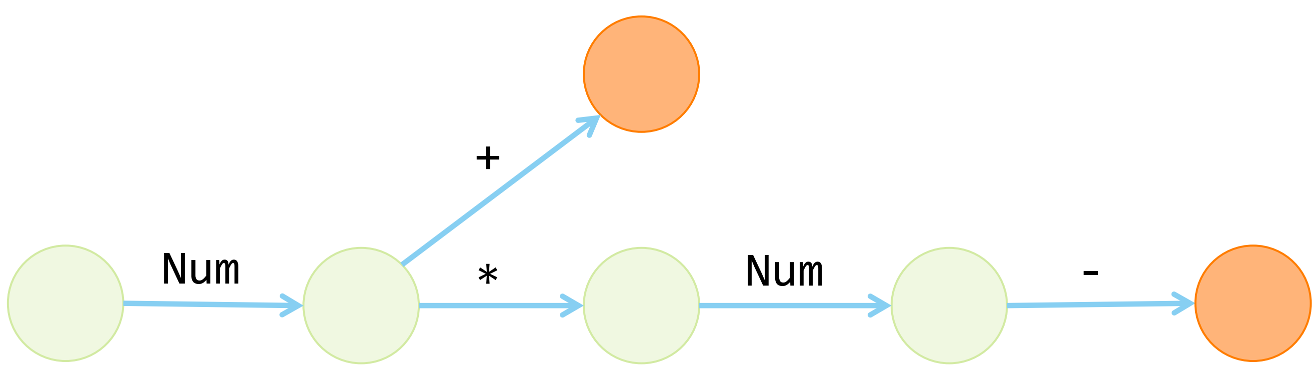
GLL parsing [Scott 2010]

- パーサコンビネータとして実装可能
- LL(k) grammar に対しては O(n) で動作する
- 能力
  - 計算量 : $O(n^3)$
  - 言語 : Context-free language
  - 文法 : Context-free grammar

Adaptive LL(*) parsing [Parr 2014]

アイデア

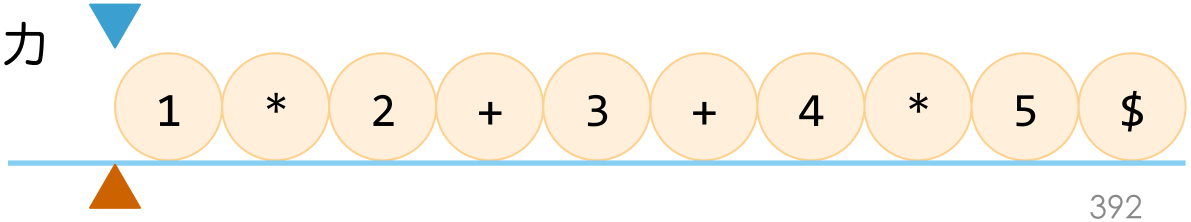
- 先読みオートマトンにより分岐を選択する
  - ➢ バックトラックを避けることで高速化を図る
  - ➢ ここまでは LL(*) のアイデア
- 動的に先読みオートマトンを作る
  - ➢ 入力言語は有限集合なので、先読み言語は常に正規



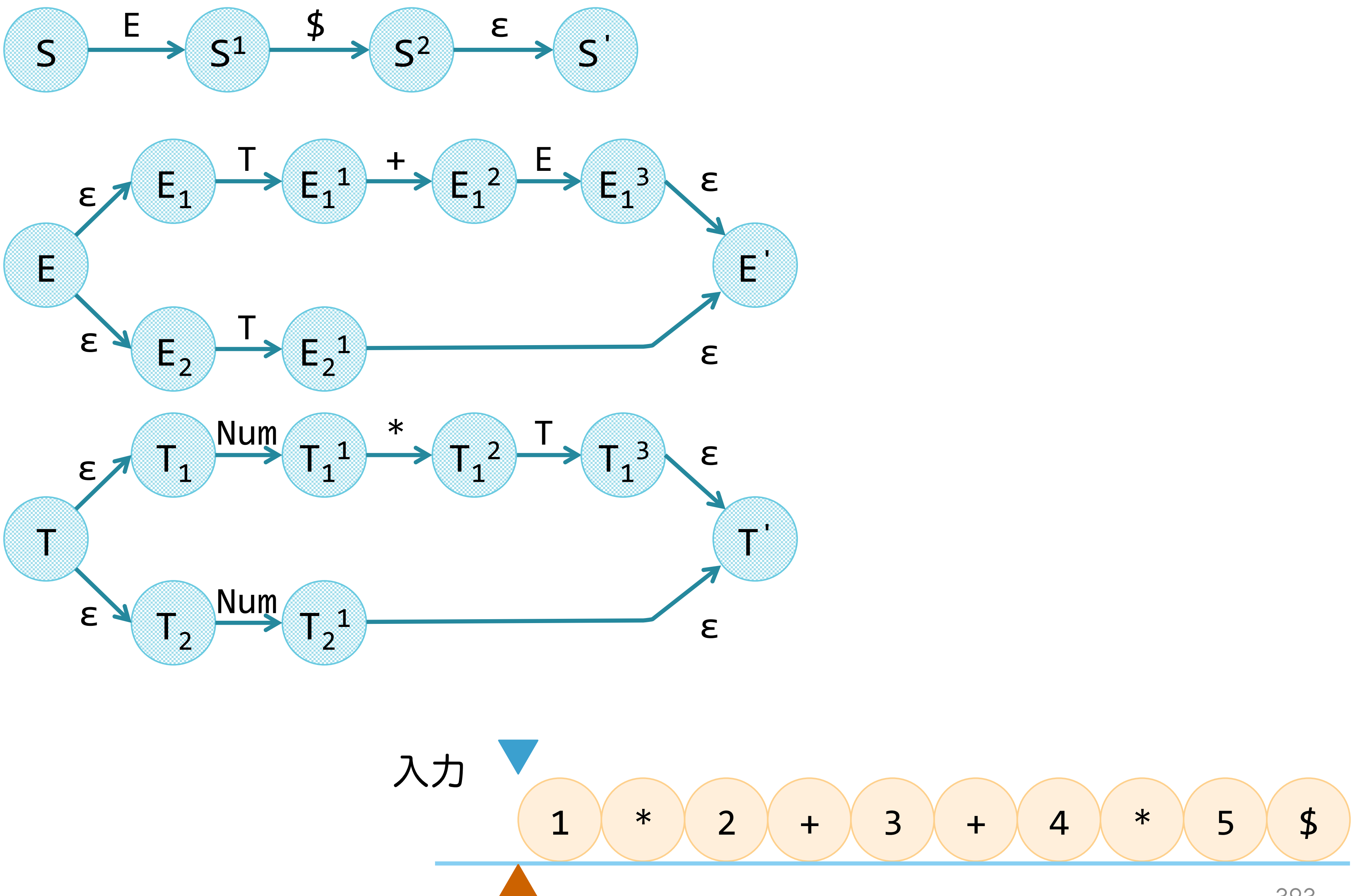
文法

```
E → T + E
E → T
T → Num * T
T → Num
```

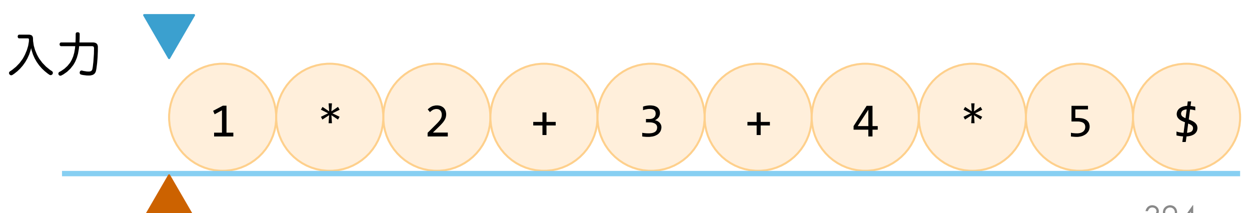
入力



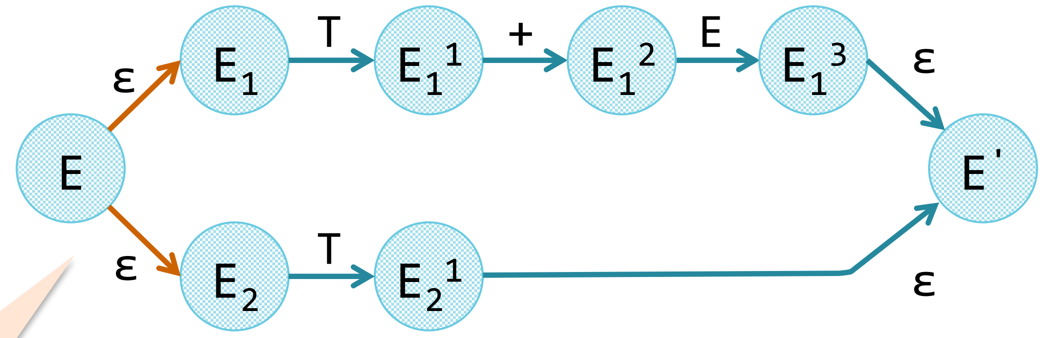
Augmented transition networks (ATNs)



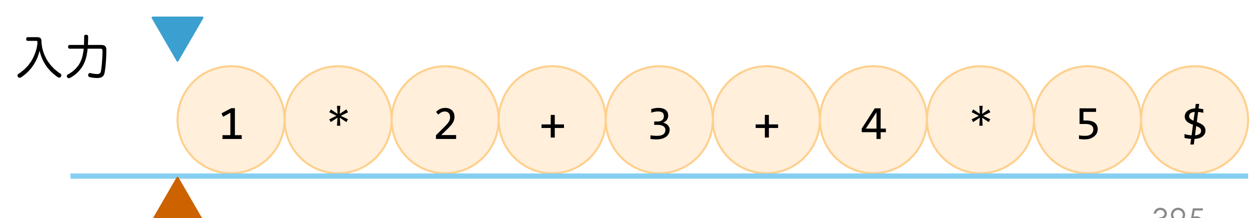
```
0: E
```

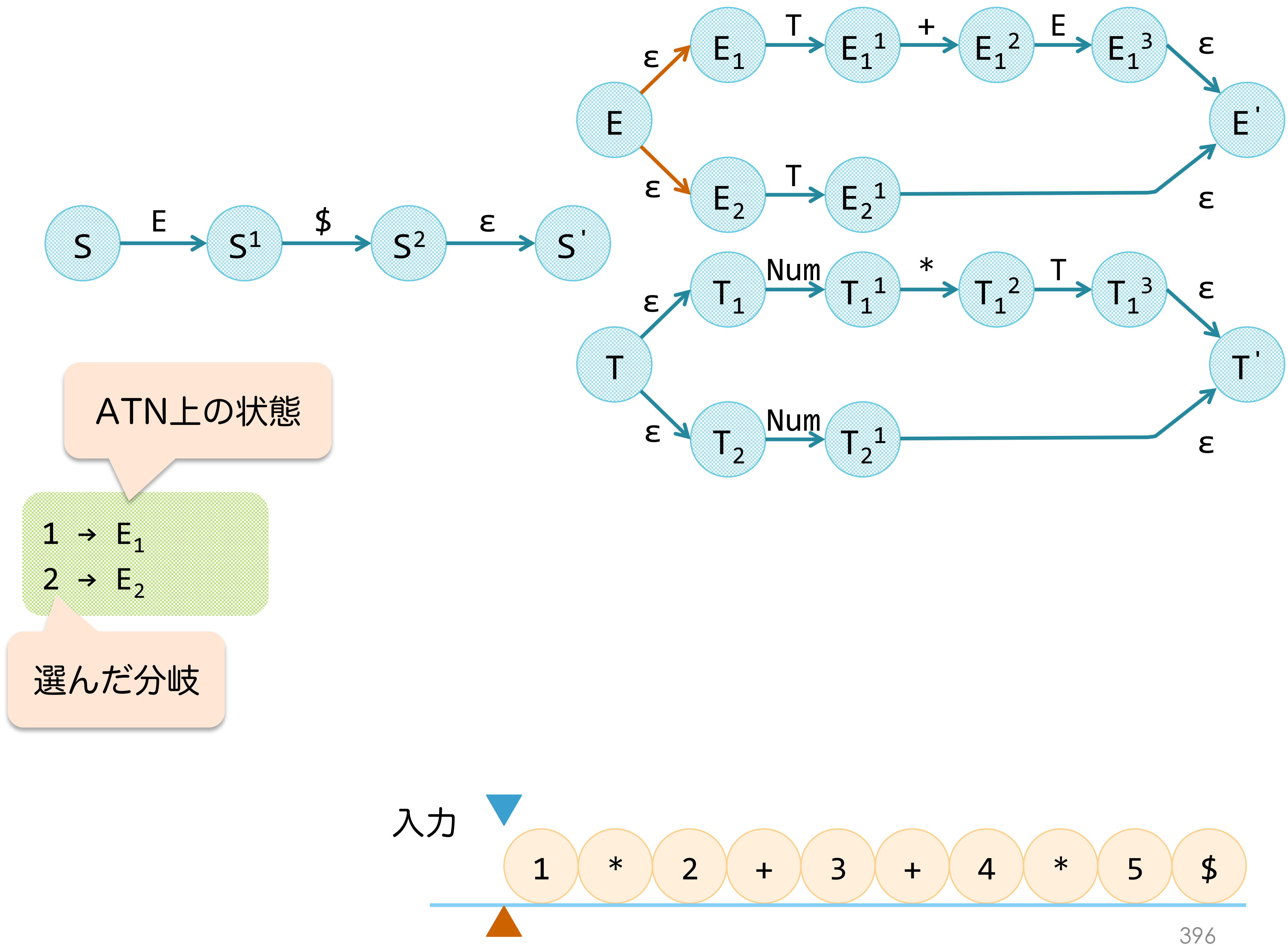


`0: E`

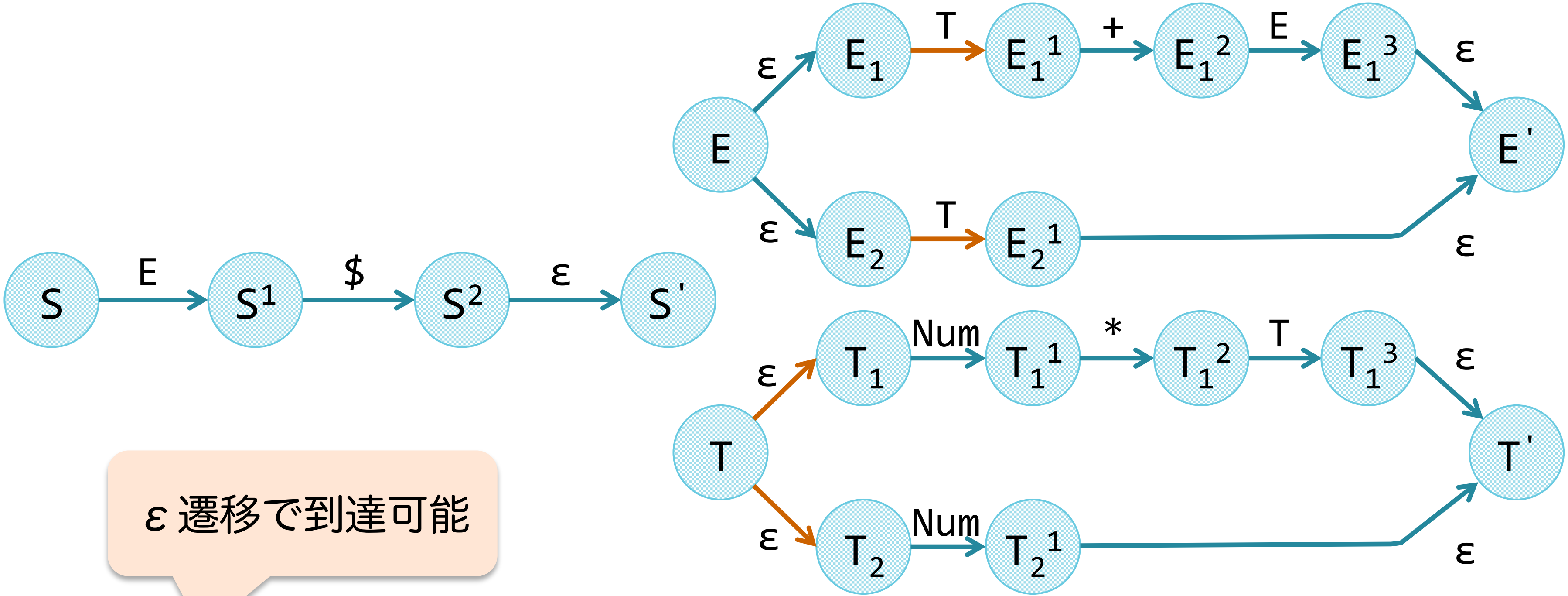


どちらの分岐か
先読みして選ぶ





S
E
S^1
$
S^2
ε
S'
E
ε
E_1
T
E_1^1
+
E_1^2
E
E_1^3
ε
E'
ε
E_2
T
E_2^1
ε
T
ε
T_1
Num
T_1^1
*
T_1^2
T
T_1^3
ε
T'
ε
T_2
Num
T_2^1
ε
ATN上の状態
1 → E_1
2 → E_2
選んだ分岐
入力
1 * 2 + 3 + 4 * 5 $



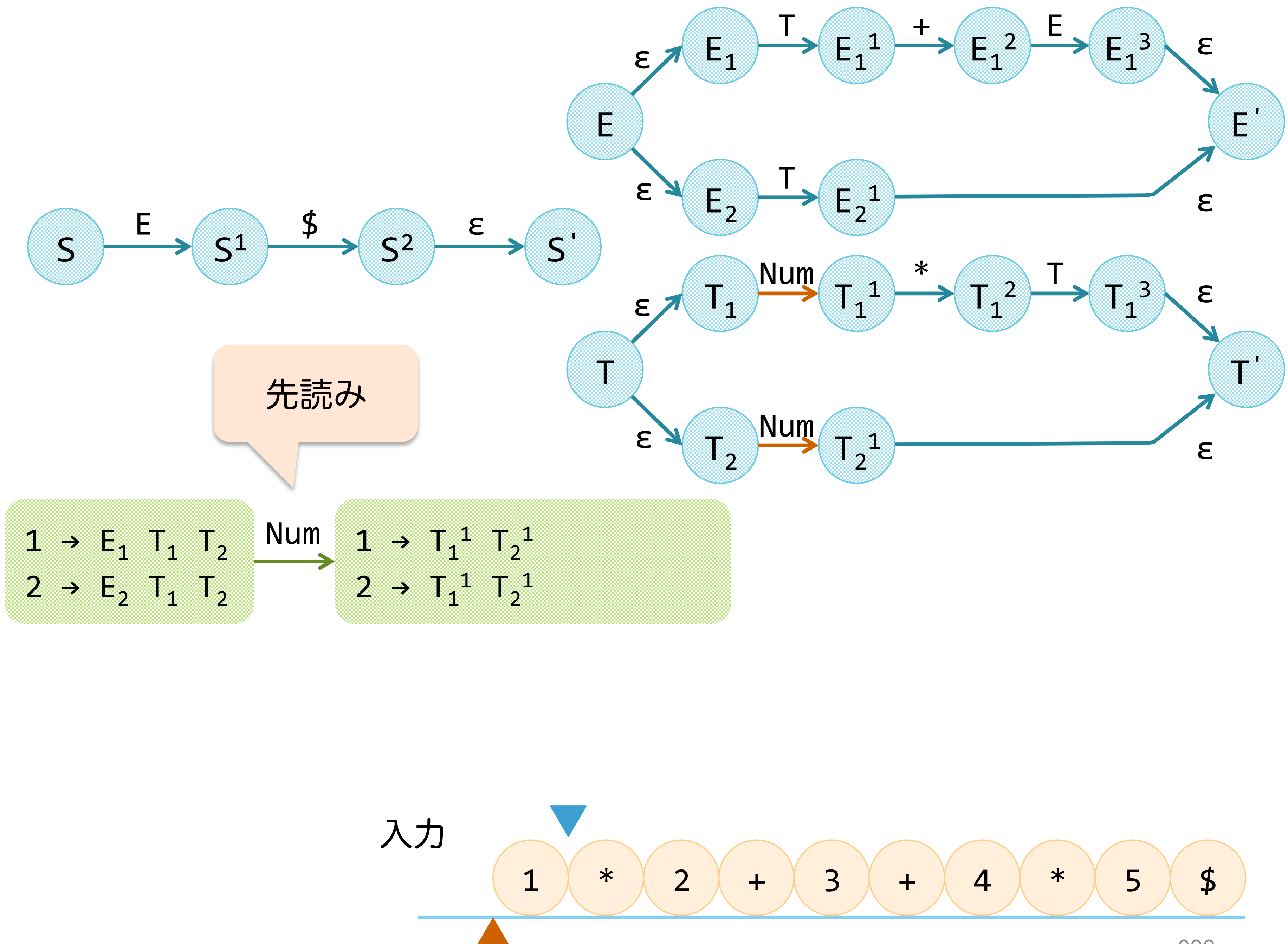
ε遷移で到達可能

1 → E_1 T_1 T_2

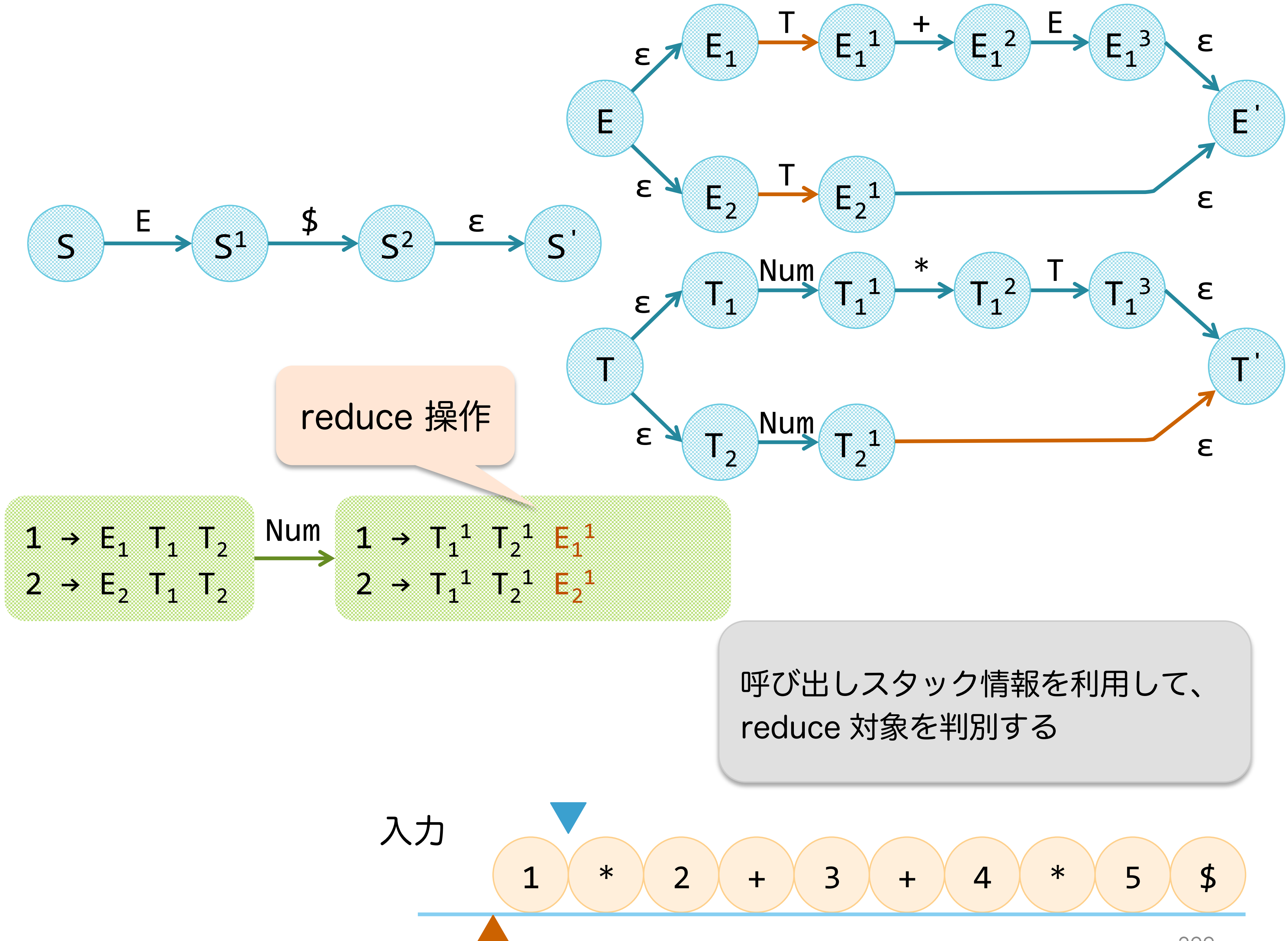
2 → E_2 T_1 T_2

入力

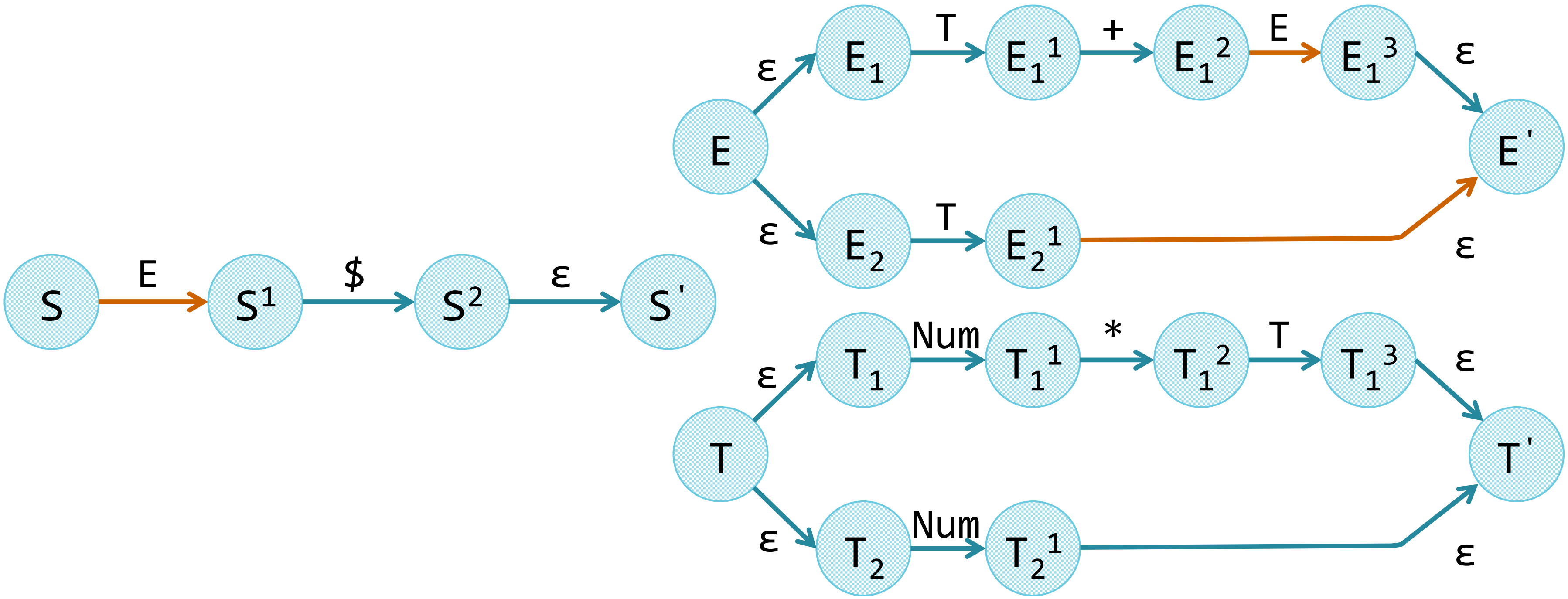
1	*	2	+	3	+	4	*	5	$

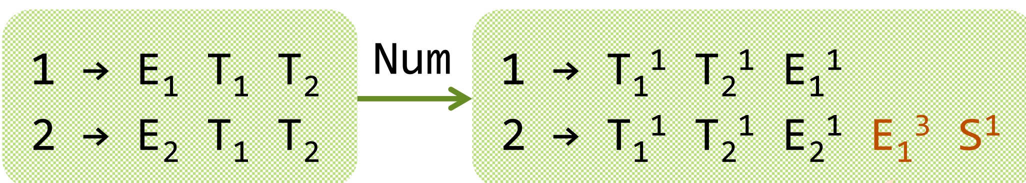


S
E
S^1
$
S^2
ε
S'
E
ε
E_1
T
$E_1{}^1$
+
$E_1{}^2$
E
$E_1{}^3$
ε
E'
ε
E_2
T
$E_2{}^1$
ε
T
ε
T_1
Num
$T_1{}^1$
*
$T_1{}^2$
T
$T_1{}^3$
ε
T'
ε
T_2
Num
$T_2{}^1$
ε
先読み
1 → E_1 T_1 T_2
2 → E_2 T_1 T_2
Num
1 → $T_1{}^1$ $T_2{}^1$
2 → $T_1{}^1$ $T_2{}^1$
入力
1 * 2 + 3 + 4 * 5 $



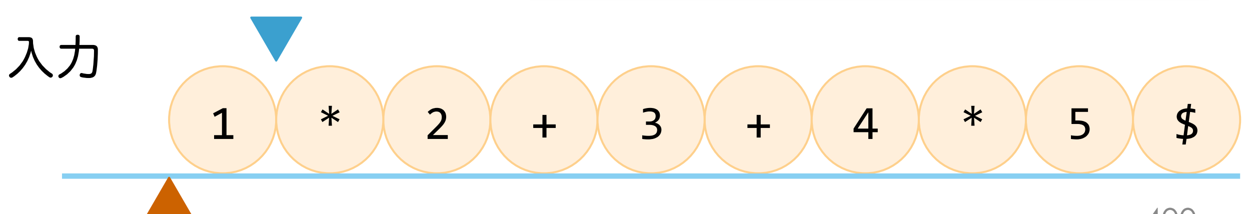
呼び出しスタック情報を利用して、
reduce 対象を判別する

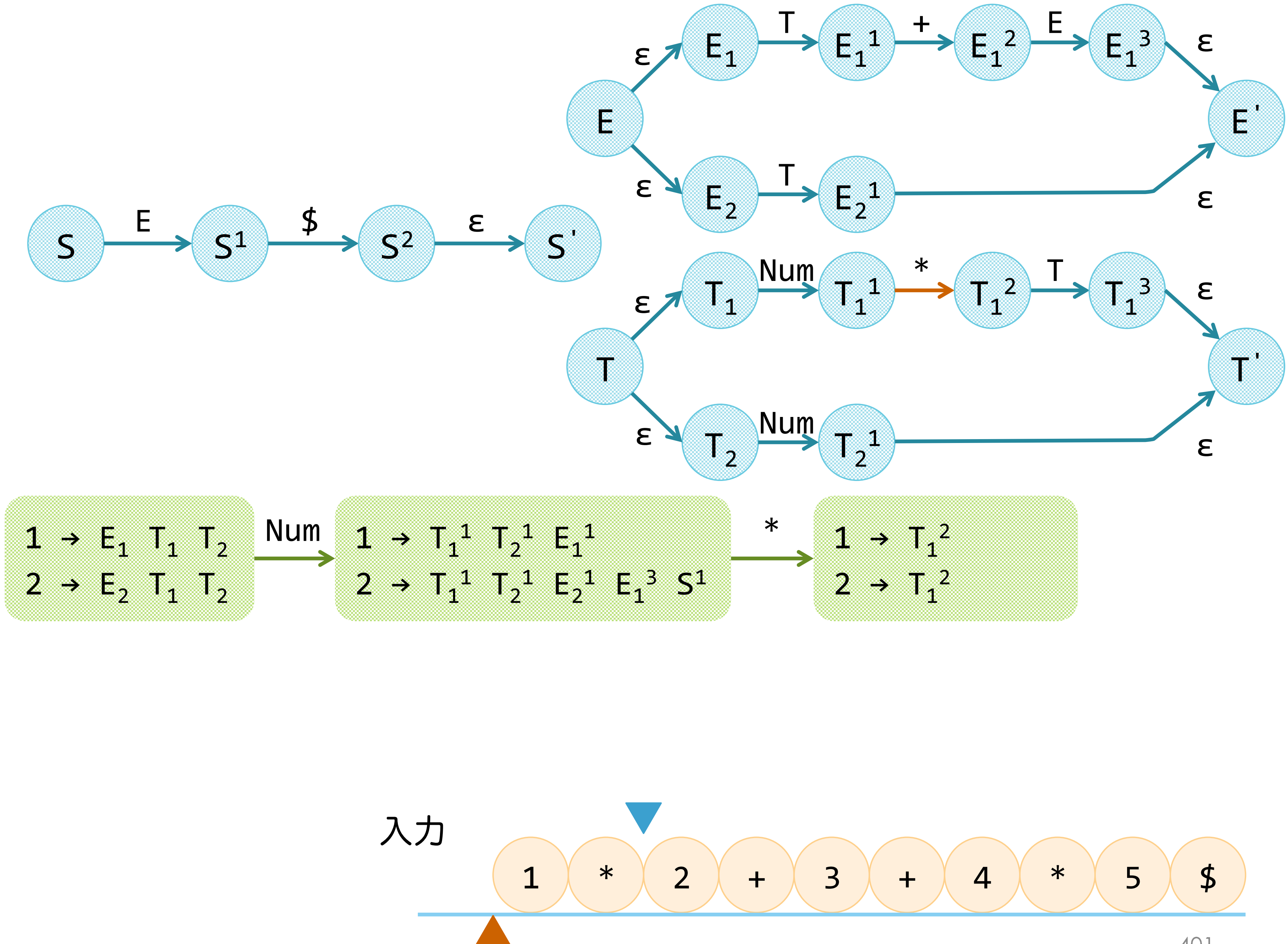




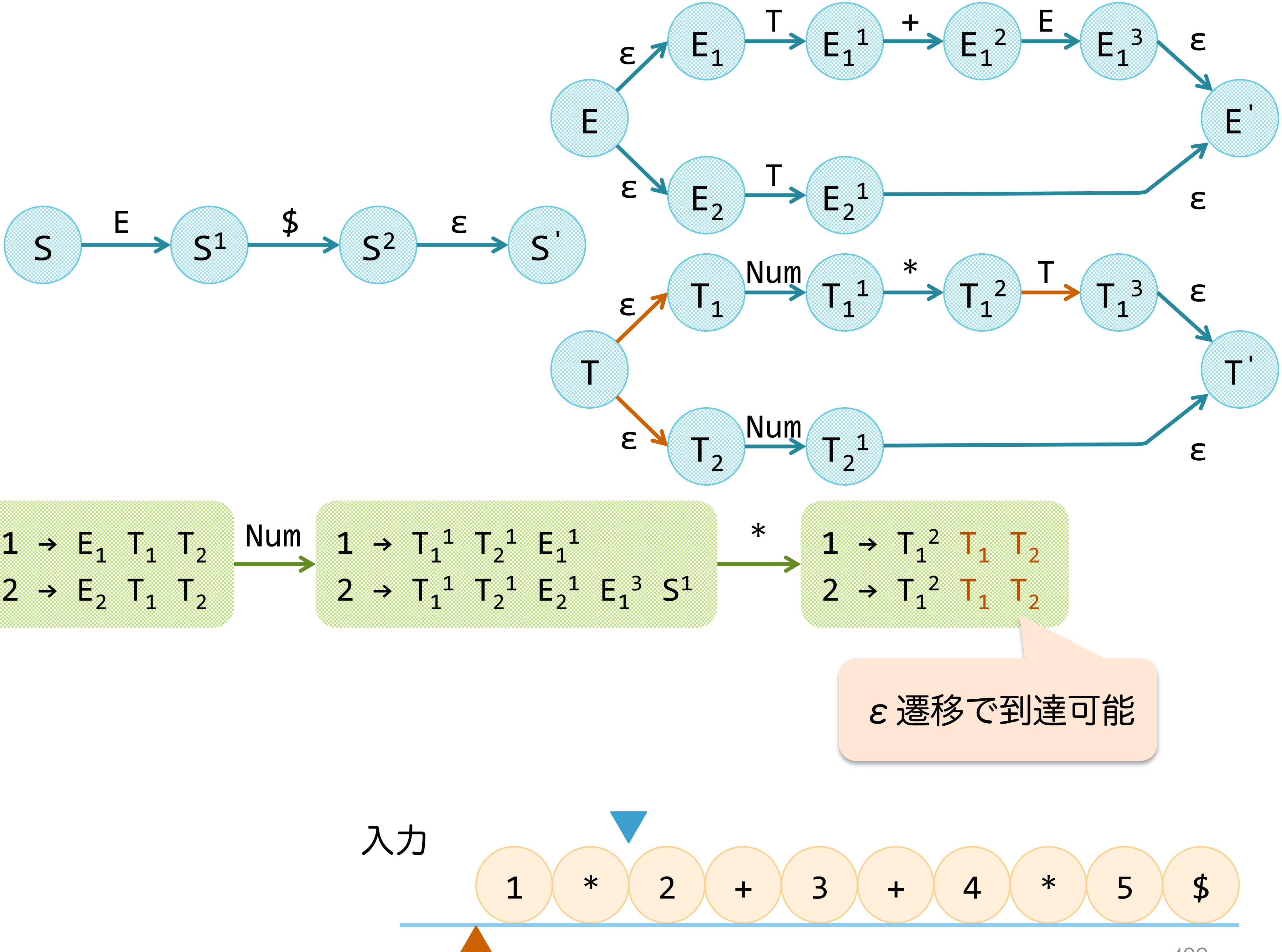
reduce 操作

外側の文脈は不明なので、
E による遷移の右辺を追加する

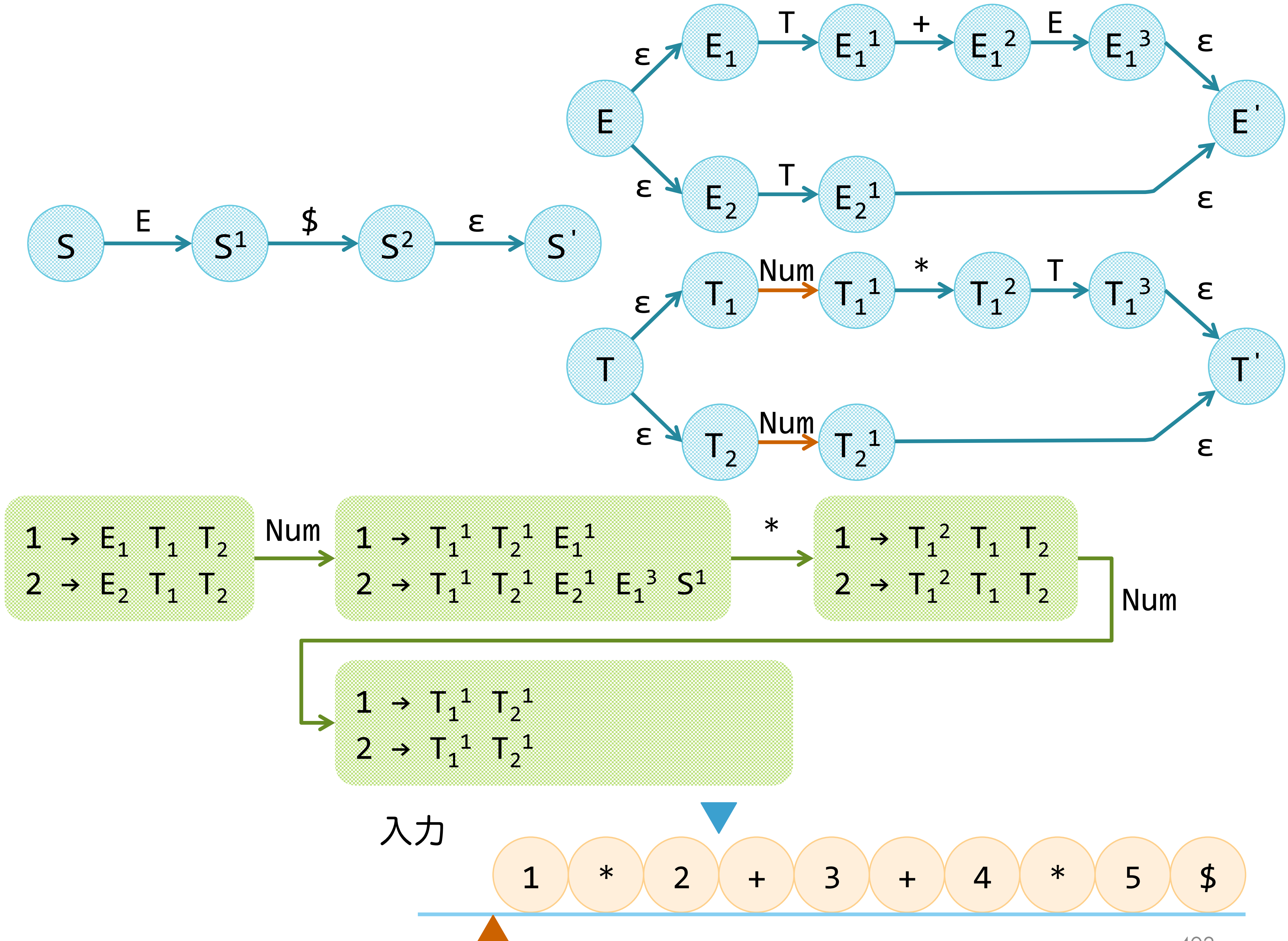




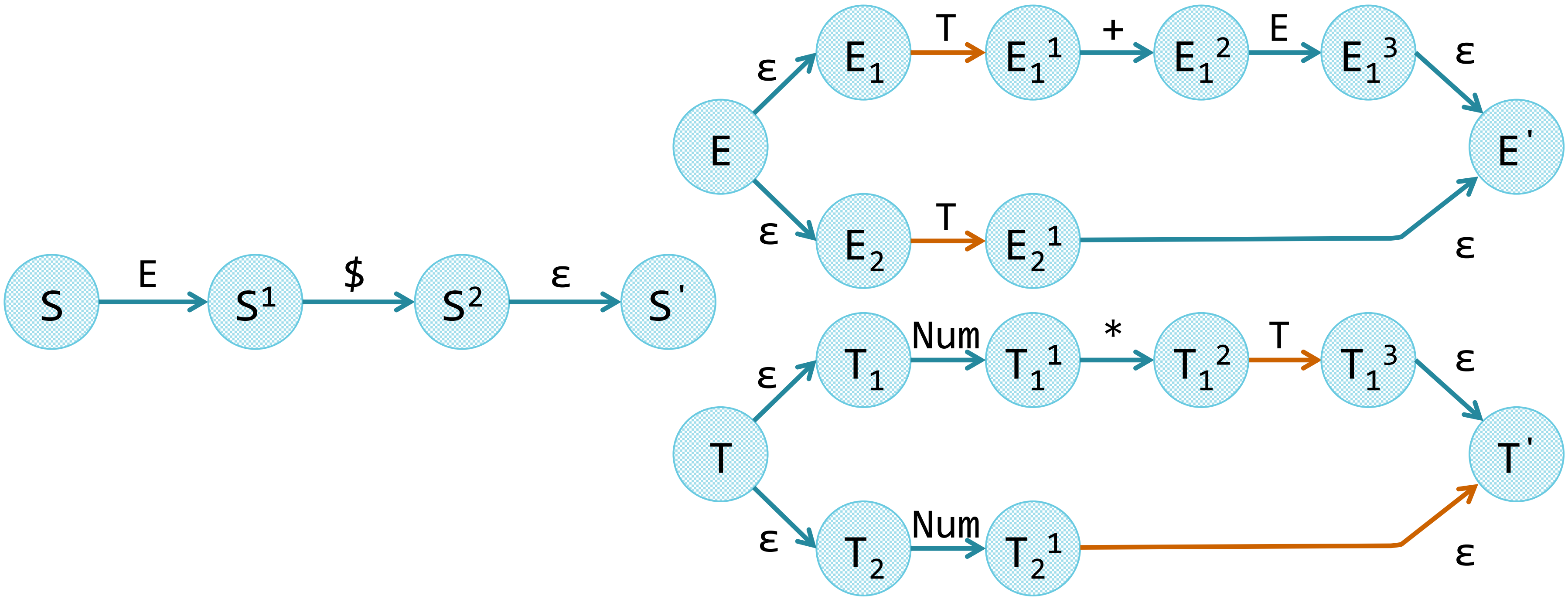
S
E
S1
$
S2
ε
S'
E
ε
E1
T
E1 1
+
E1 2
E
E1 3
ε
E'
ε
E2
T
E2 1
ε
T
ε
T1
Num
T1 1
*
T1 2
T
T1 3
ε
T'
ε
T2
Num
T2 1
ε
1 → E1 T1 T2
2 → E2 T1 T2
Num
1 → T1 1 T2 1 E1 1
2 → T1 1 T2 1 E2 1 E1 3 S1
*
1 → T1 2
2 → T1 2
入力
1 * 2 + 3 + 4 * 5 $

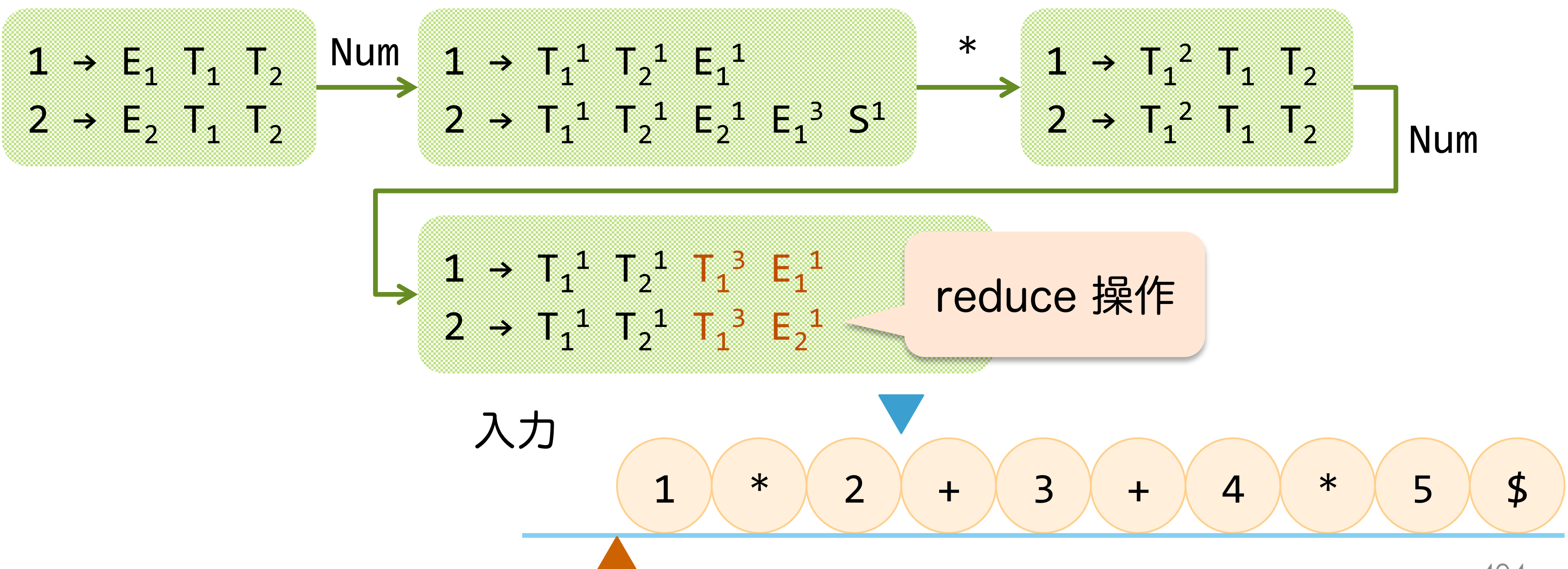


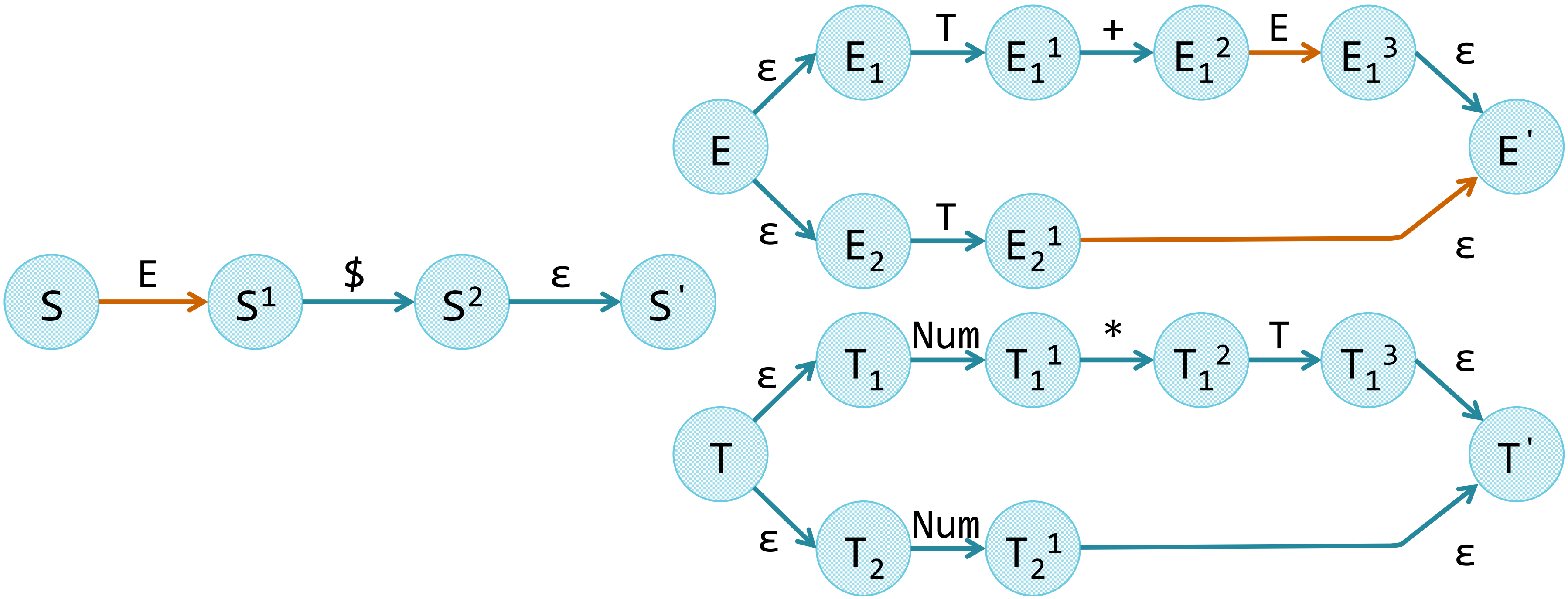
1 → E₁ T₁ T₂
2 → E₂ T₁ T₂
Num
1 → T₁¹ T₂¹ E₁¹
2 → T₁¹ T₂¹ E₂¹ E₁³ S¹
*
1 → T₁² T₁ T₂
2 → T₁² T₁ T₂
ε 遷移で到達可能
入力
1 * 2 + 3 + 4 * 5 $

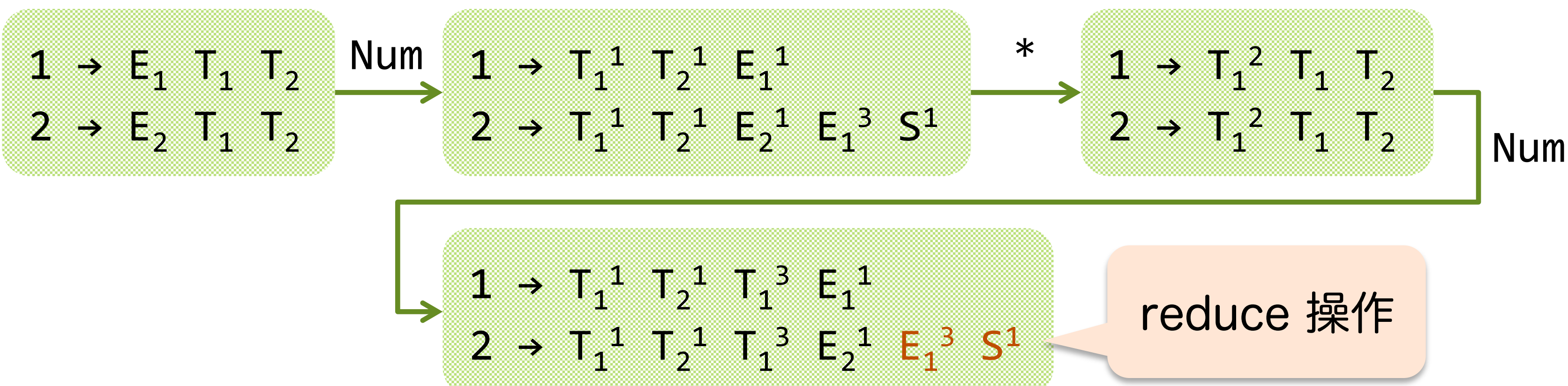


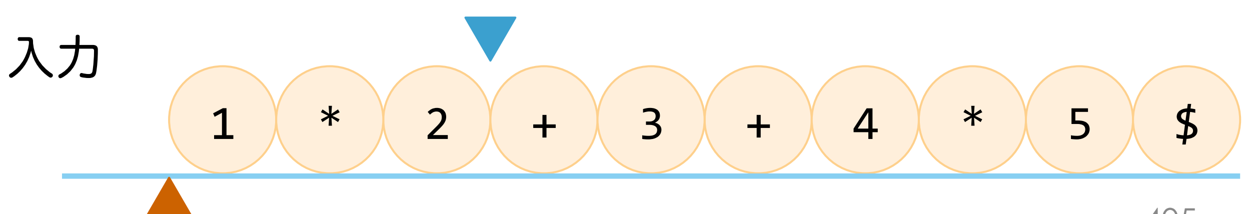
S
E
S1
$
S2
ε
S'
E
E1
T
E1 1
+
E1 2
E1 3
E'
E2
E2 1
T
Num
T1
T1 1
*
T1 2
T1 3
T'
T2
T2 1
1 → E1 T1 T2
2 → E2 T1 T2
1 → T1 1 T2 1 E1 1
2 → T1 1 T2 1 E2 1 E1 3 S1
1 → T1 2 T1 T2
2 → T1 2 T1 T2
1 → T1 1 T2 1
2 → T1 1 T2 1
入力
1 * 2 + 3 + 4 * 5 $

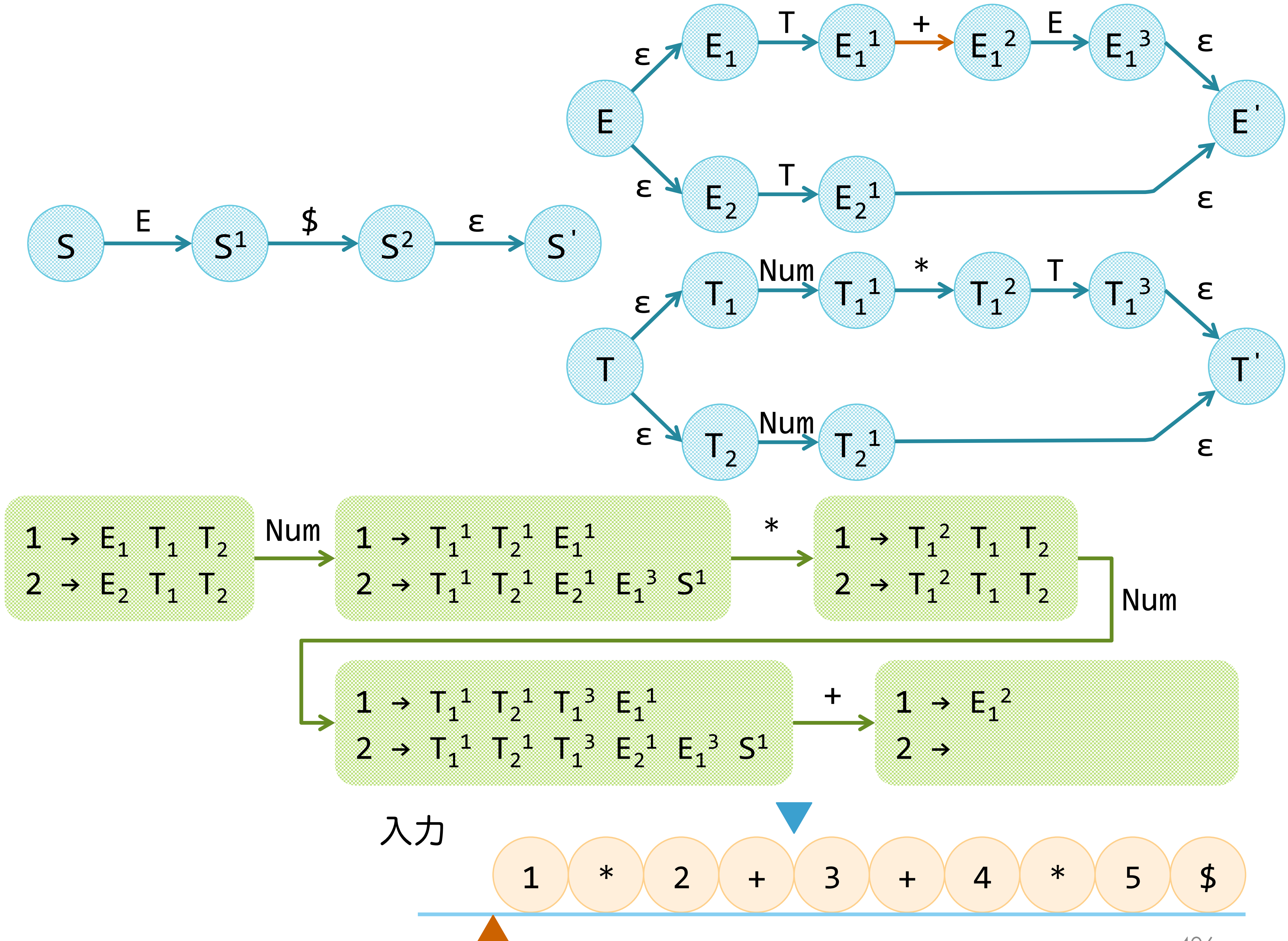

S
E
S1
$
S2
ε
S'
E
ε
E1
T
E1 1
+
E1 2
E
E1 3
ε
E'
ε
E2
T
E2 1
ε
T
ε
T1
Num
T1 1
*
T1 2
T
T1 3
ε
T'
ε
T2
Num
T2 1
ε

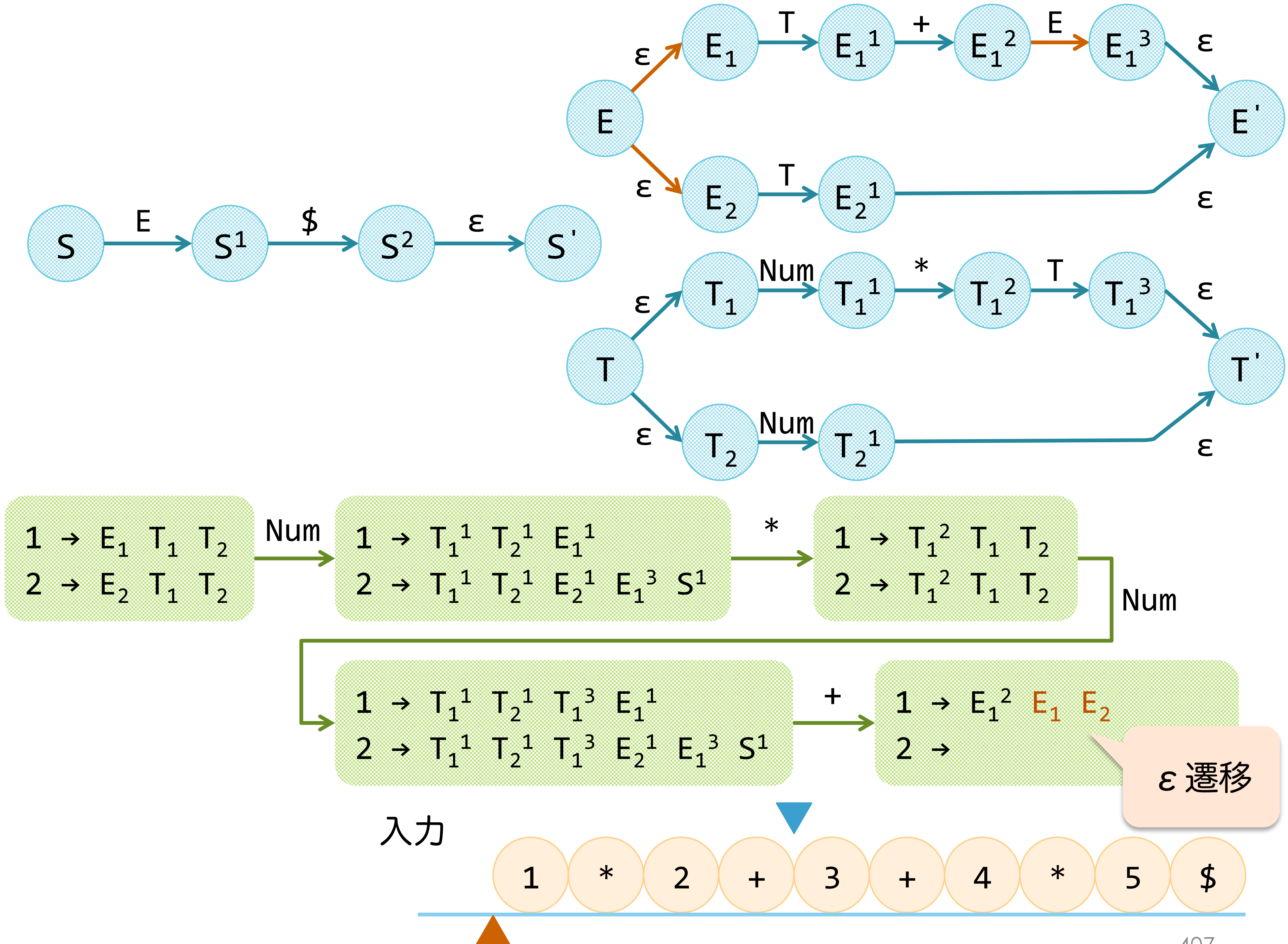

1 → E1 T1 T2
2 → E2 T1 T2
Num
1 → T1 1 T2 1 E1 1
2 → T1 1 T2 1 E2 1 E1 3 S1
*
1 → T1 2 T1 T2
2 → T1 2 T1 T2
Num
1 → T1 1 T2 1 T1 3 E1 1
2 → T1 1 T2 1 T1 3 E2 1
reduce 操作
入力
1
*
2
+
3
+
4
*
5
$

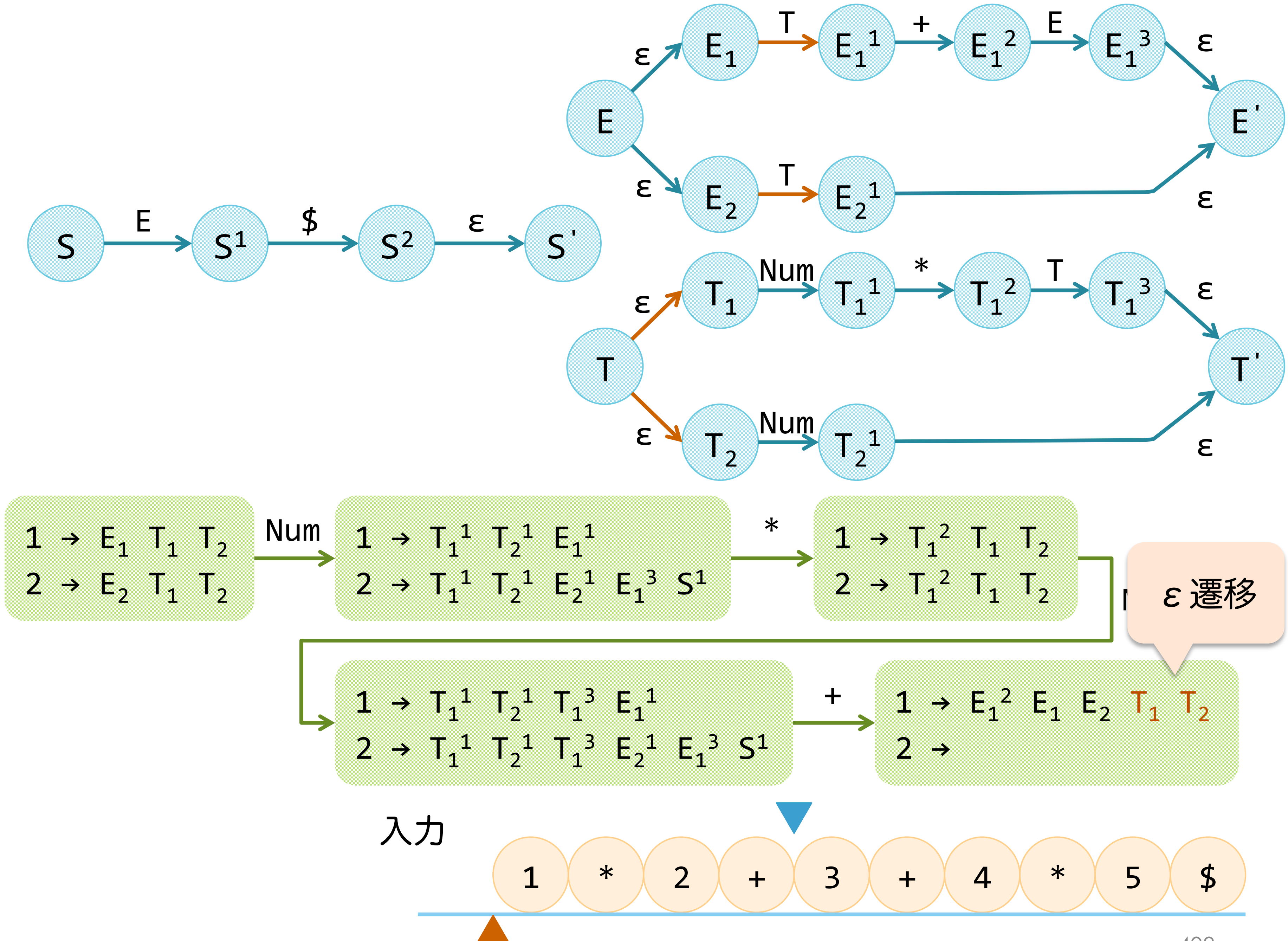

S
E
S1
$
S2
ε
S'
E
ε
E1
T
E1 1
+
E1 2
E
E1 3
ε
E'
ε
E2
T
E2 1
ε
T
ε
T1
Num
T1 1
*
T1 2
T
T1 3
ε
T'
ε
T2
Num
T2 1
ε


1 → E1 T1 T2
2 → E2 T1 T2
Num
1 → T1 1 T2 1 E1 1
2 → T1 1 T2 1 E2 1 E1 3 S1
*
1 → T1 2 T1 T2
2 → T1 2 T1 T2
Num
1 → T1 1 T2 1 T1 3 E1 1
2 → T1 1 T2 1 T1 3 E2 1 E1 3 S1
reduce 操作

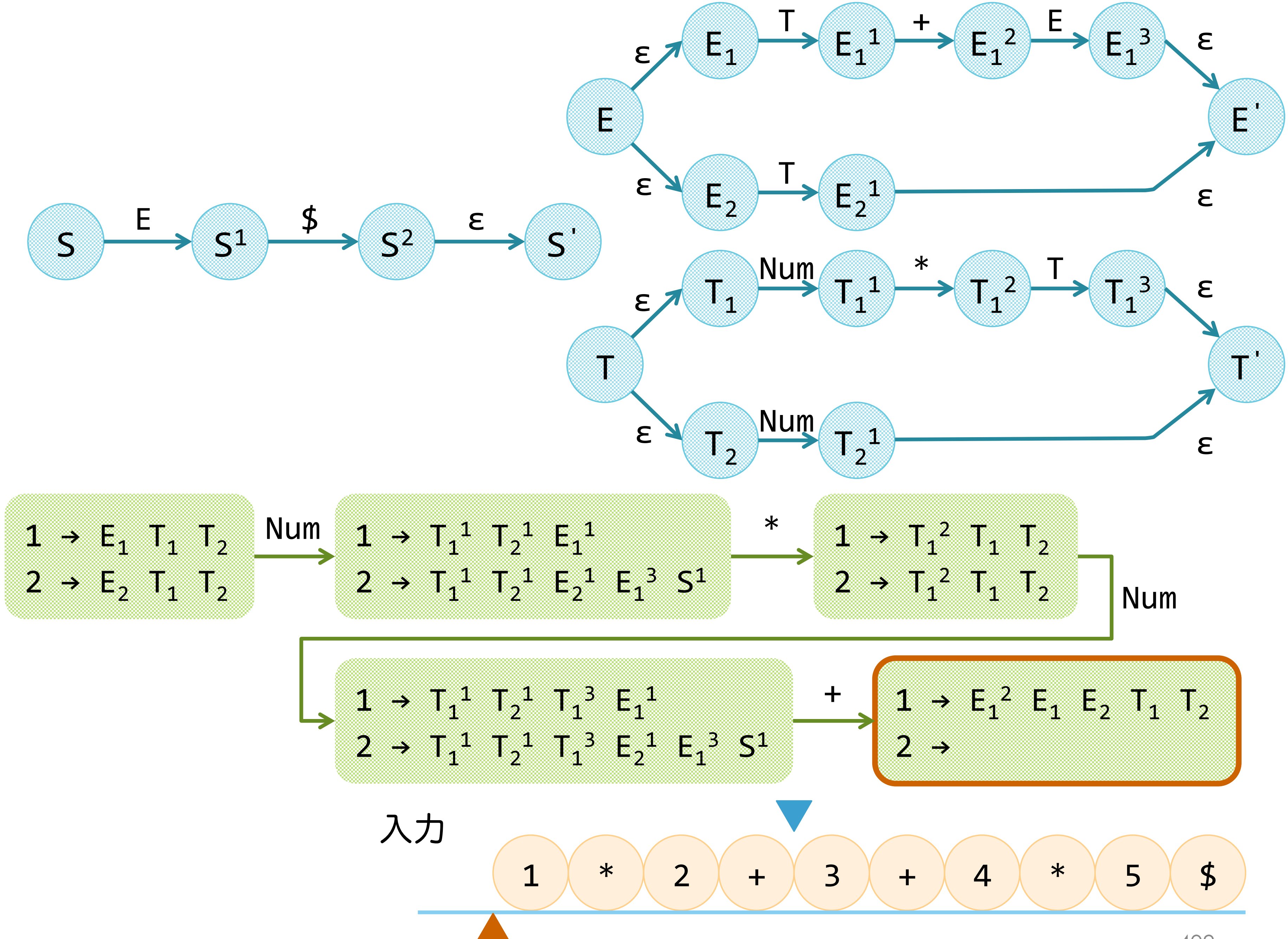

入力
1
*
2
+
3
+
4
*
5
$


S —E→ S¹ —$→ S² —ε→ S'
E —ε→ E1 —T→ E1¹ —+→ E1² —E→ E1³ —ε→ E'
E —ε→ E2 —T→ E2¹ —ε→ E'
T —ε→ T1 —Num→ T1¹ —*→ T1² —T→ T1³ —ε→ T'
T —ε→ T2 —Num→ T2¹ —ε→ T'
1 → E1 T1 T2
2 → E2 T1 T2
Num
1 → T1¹ T2¹ E1¹
2 → T1¹ T2¹ E2¹ E1³ S¹
*
1 → T1² T1 T2
2 → T1² T1 T2
Num
1 → T1¹ T2¹ T1³ E1¹
2 → T1¹ T2¹ T1³ E2¹ E1³ S¹
+
1 → E1²
2 →
入力
1 * 2 + 3 + 4 * 5 $


S
E
S¹
$
S²
ε
S'
E
ε
E1
T
E1 1
+
E1 2
E
E1 3
ε
E'
E2
T
E2 1
ε
T
T1
Num
T1 1
*
T1 2
T
T1 3
T'
T2
Num
T2 1
1 → E1 T1 T2
2 → E2 T1 T2
Num
1 → T1 1 T2 1 E1 1
2 → T1 1 T2 1 E2 1 E1 3 S1
*
1 → T1 2 T1 T2
2 → T1 2 T1 T2
Num
1 → T1 1 T2 1 T1 3 E1 1
2 → T1 1 T2 1 T1 3 E2 1 E1 3 S1
+
1 → E1 2 E1 E2
2 →
ε 遷移
入力
1 * 2 + 3 + 4 * 5 $

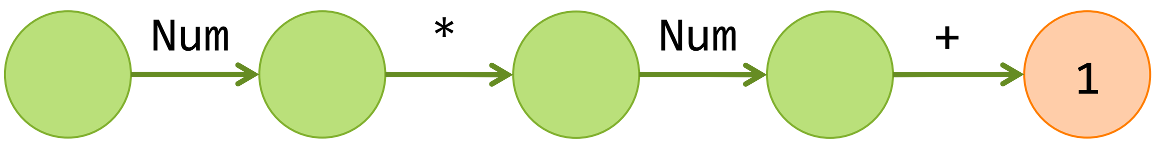


S → E → S¹ → $ → S² → ε → S'
E → ε → E₁ → T → E₁¹ → + → E₁² → E → E₁³ → ε → E'
E → ε → E₂ → T → E₂¹ → ε → E'
T → ε → T₁ → Num → T₁¹ → * → T₁² → T → T₁³ → ε → T'
T → ε → T₂ → Num → T₂¹ → ε → T'
1 → E₁ T₁ T₂
2 → E₂ T₁ T₂
Num
1 → T₁¹ T₂¹ E₁¹
2 → T₁¹ T₂¹ E₂¹ E₁³ S¹
*
1 → T₁² T₁ T₂
2 → T₁² T₁ T₂
ε 遷移
1 → T₁¹ T₂¹ T₁³ E₁¹
2 → T₁¹ T₂¹ T₁³ E₂¹ E₁³ S¹
+
1 → E₁² E₁ E₂ T₁ T₂
2 →
入力
1 * 2 + 3 + 4 * 5 $

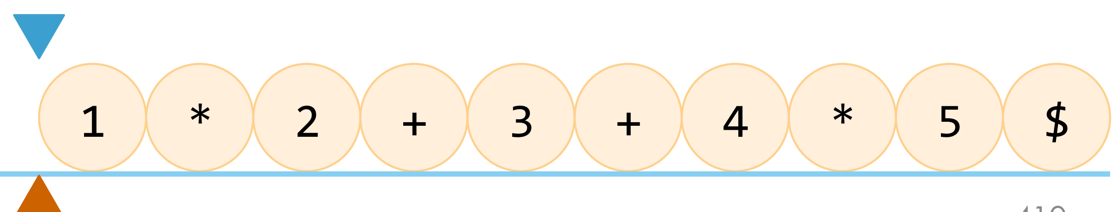

S
E
S1
$
S2
ε
S'
E
E1
T
E1 1
+
E1 2
E1 3
E'
E2
E2 1
T1
Num
T1 1
*
T1 2
T1 3
T'
T2
T2 1
1 → E1 T1 T2
2 → E2 T1 T2
1 → T1 1 T2 1 E1 1
2 → T1 1 T2 1 E2 1 E1 3 S1
1 → T1 2 T1 T2
2 → T1 2 T1 T2
1 → T1 1 T2 1 T1 3 E1 1
2 → T1 1 T2 1 T1 3 E2 1 E1 3 S1
1 → E1 2 E1 E2 T1 T2
2 →
入力
1
*
2
+
3
+
4
*
5
$

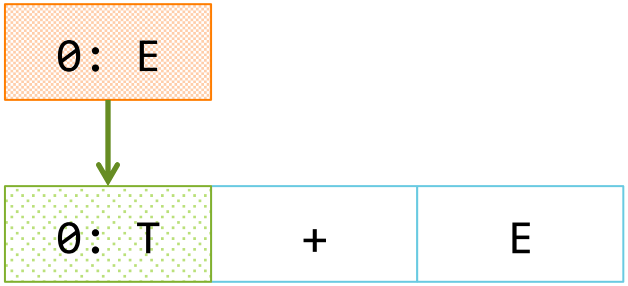
```
0: E
```

E の先読みオートマトン

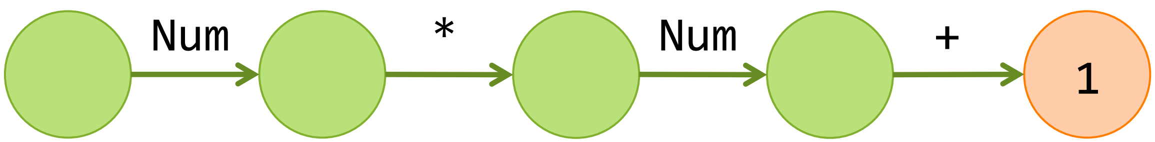


入力

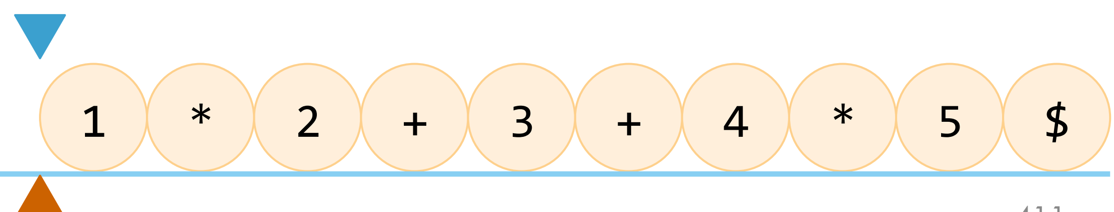


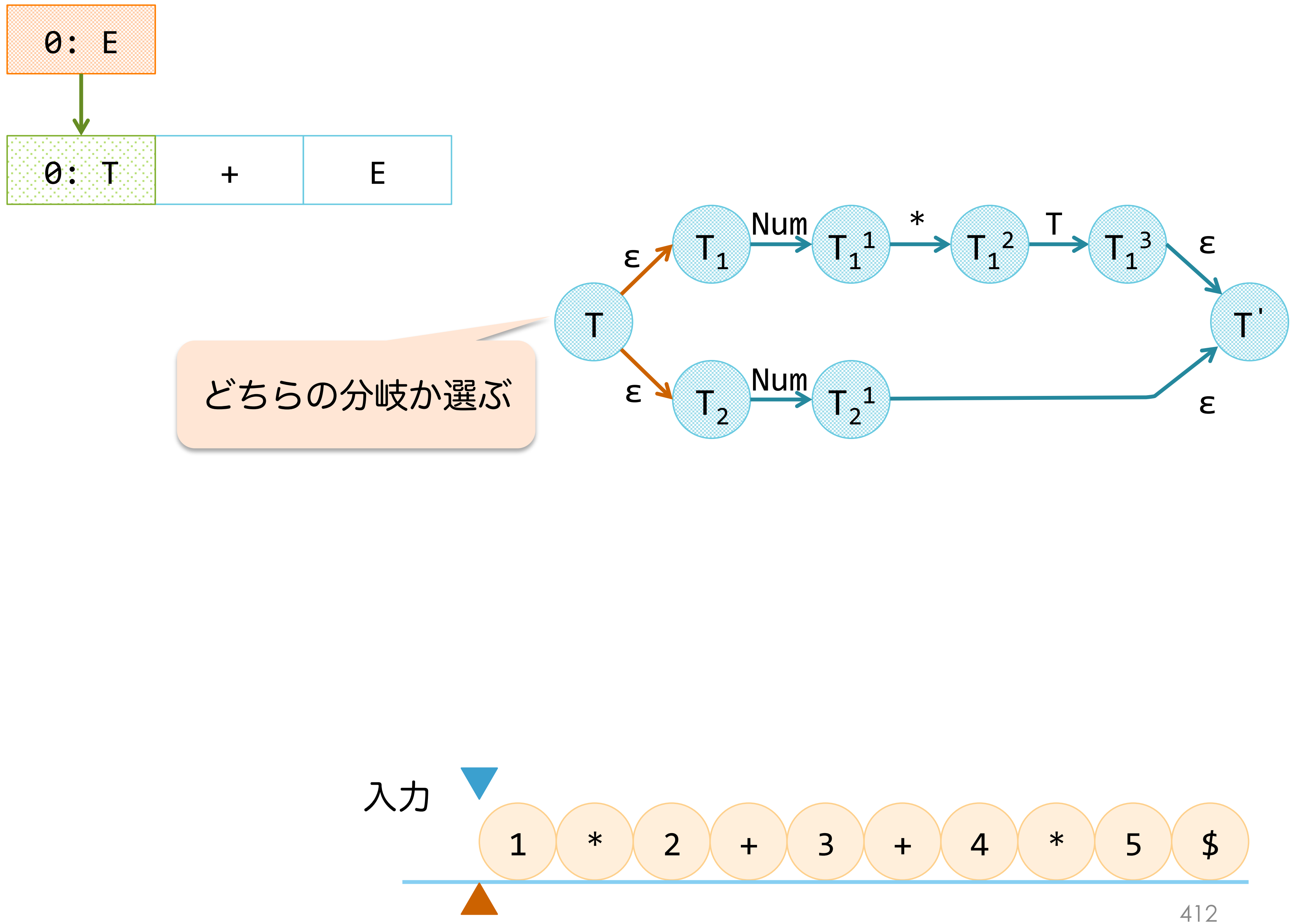


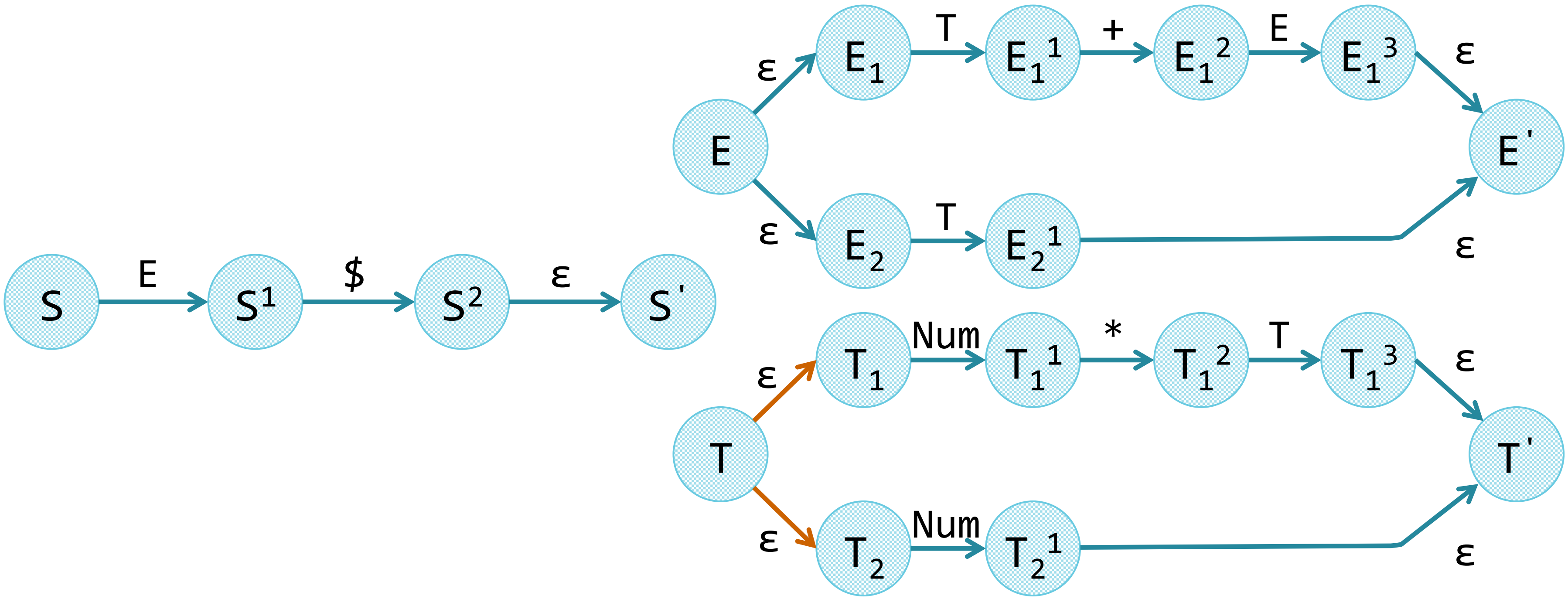
E の先読みオートマトン



入力

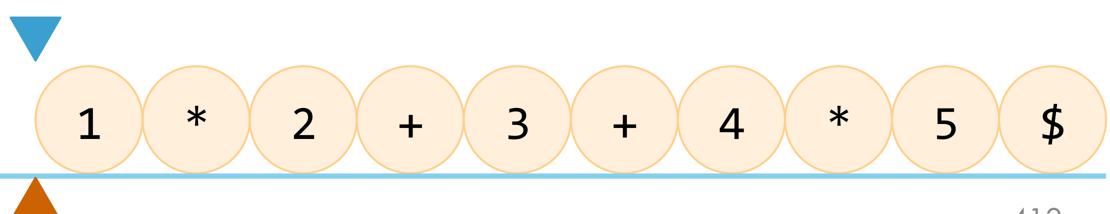


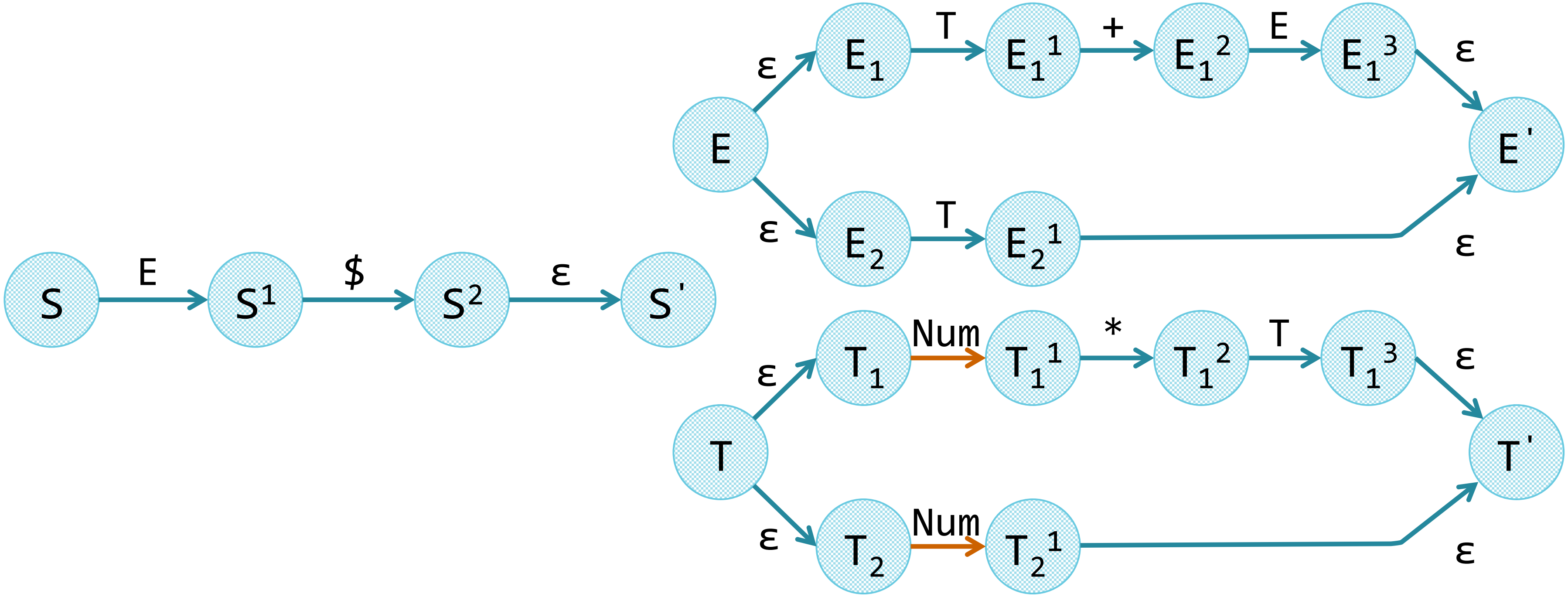

0: E
0: T
+
E
T
ε
ε
Num
*
T
ε
ε
T'
どちらの分岐か選ぶ
入力
1
*
2
+
3
+
4
*
5
$

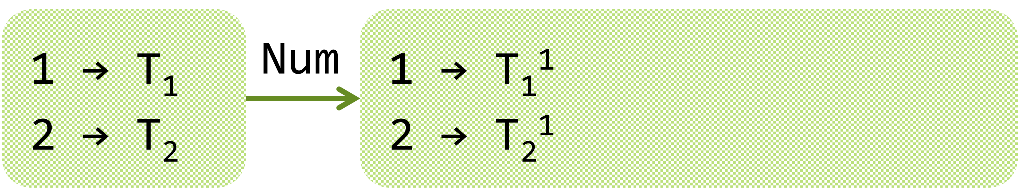

S
E
S¹
$
S²
ε
S'
E
ε
E₁
T
E₁¹
+
E₁²
E
E₁³
ε
E'
ε
E₂
T
E₂¹
ε
T
ε
T₁
Num
T₁¹
*
T₁²
T
T₁³
ε
T'
ε
T₂
Num
T₂¹
ε

1 → T_1
2 → T_2

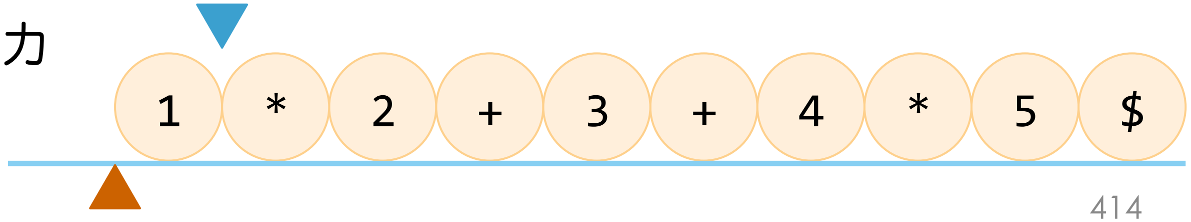
入力

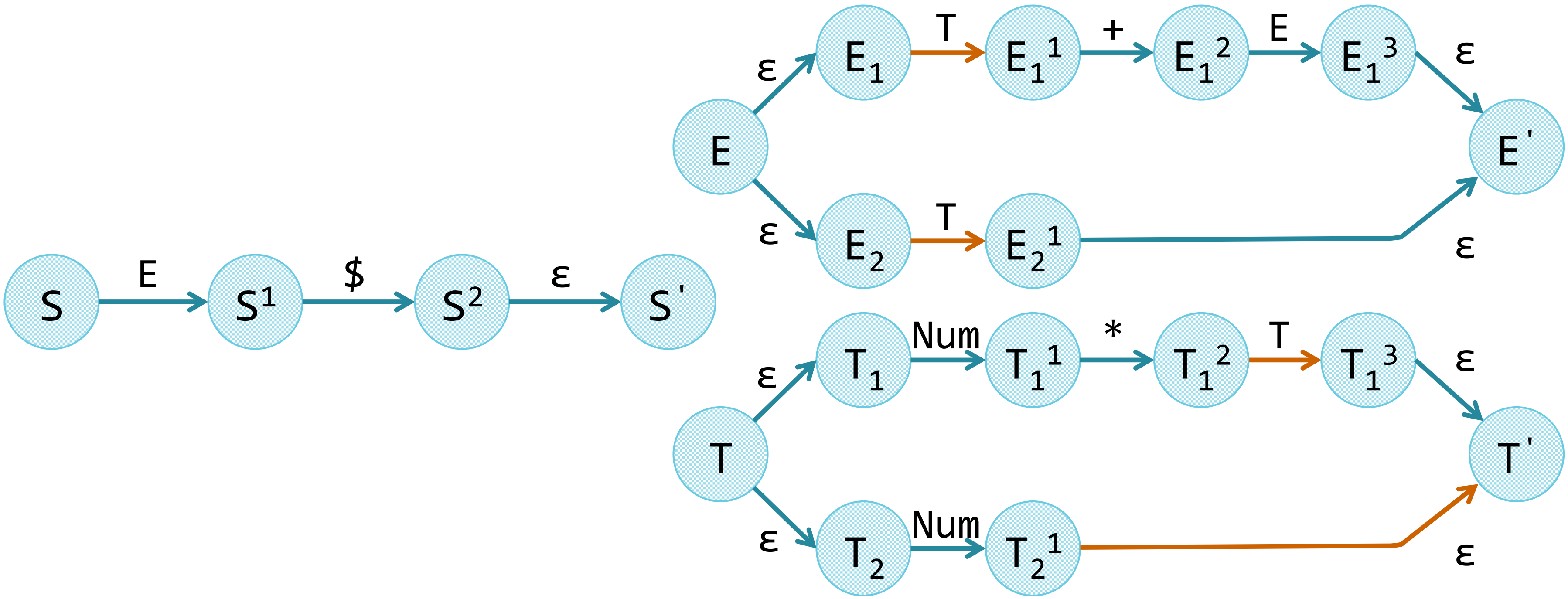

1
*
2
+
3
+
4
*
5
$

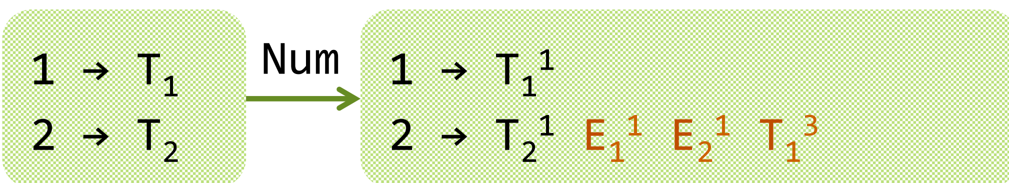

S
E
S1
$
S2
ε
S'
E
ε
E1
T
E1 1
+
E1 2
E
E1 3
ε
E'
ε
E2
T
E2 1
ε
T
ε
T1
Num
T1 1
*
T1 2
T
T1 3
ε
T'
ε
T2
Num
T2 1
ε

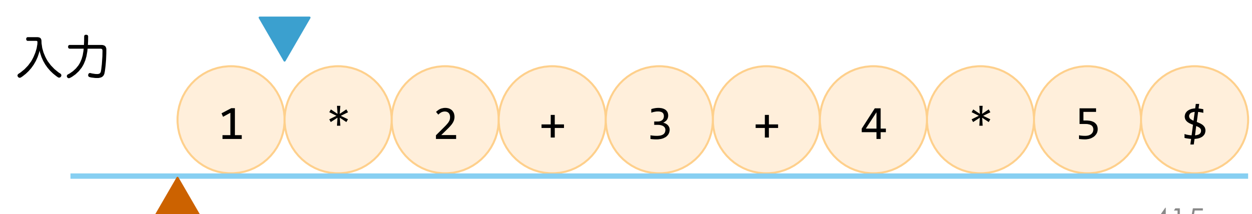

1 → T1
2 → T2
Num
1 → T1 1
2 → T2 1

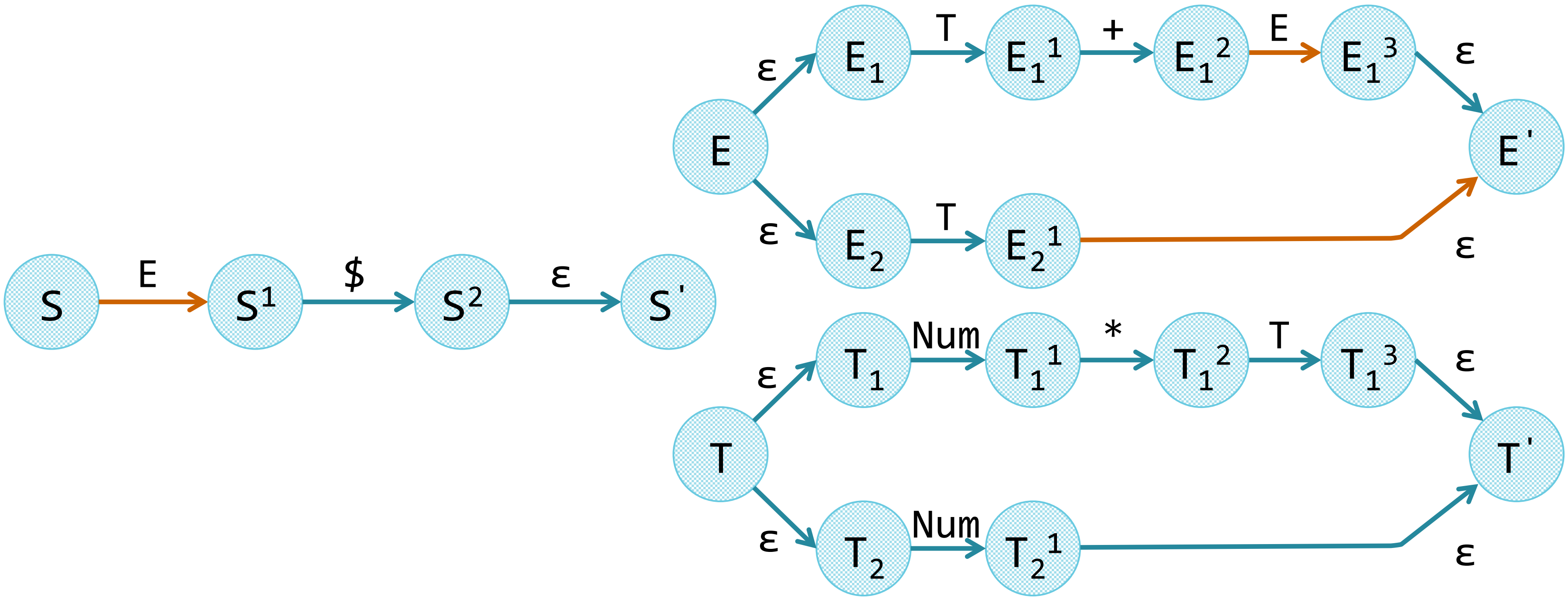
入力


1
*
2
+
3
+
4
*
5
$

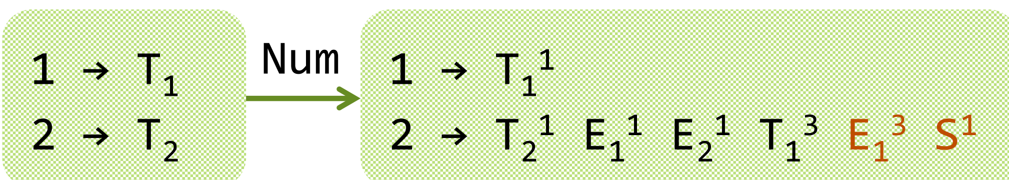

S
E
S¹
$
S²
ε
S'
E
ε
E₁
T
E₁¹
+
E₁²
E
E₁³
ε
E'
ε
E₂
T
E₂¹
ε
T
ε
T₁
Num
T₁¹
*
T₁²
T
T₁³
ε
T'
ε
T₂
Num
T₂¹
ε


1 → T_1
2 → T_2
Num
1 → T_1^1
2 → T_2^1 E_1^1 E_2^1 T_1^3


入力
1 * 2 + 3 + 4 * 5 $

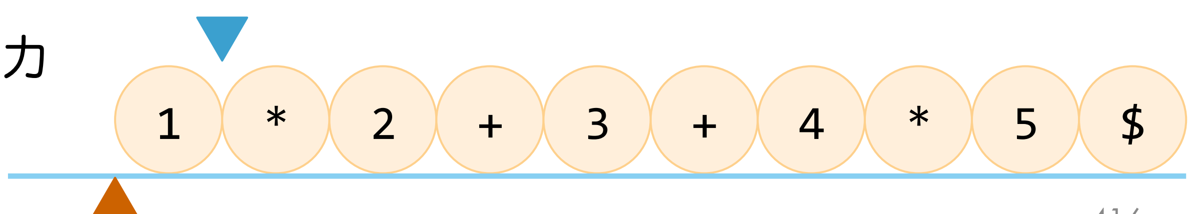


S $\xrightarrow{E}$ S^1 $\xrightarrow{\$}$ S^2 $\xrightarrow{\varepsilon}$ S'
E $\xrightarrow{\varepsilon}$ E_1 $\xrightarrow{T}$ E_1^1 $\xrightarrow{+}$ E_1^2 $\xrightarrow{E}$ E_1^3 $\xrightarrow{\varepsilon}$ E'
E $\xrightarrow{\varepsilon}$ E_2 $\xrightarrow{T}$ E_2^1 $\xrightarrow{\varepsilon}$ E'
T $\xrightarrow{\varepsilon}$ T_1 $\xrightarrow{Num}$ T_1^1 $\xrightarrow{*}$ T_1^2 $\xrightarrow{T}$ T_1^3 $\xrightarrow{\varepsilon}$ T'
T $\xrightarrow{\varepsilon}$ T_2 $\xrightarrow{Num}$ T_2^1 $\xrightarrow{\varepsilon}$ T'

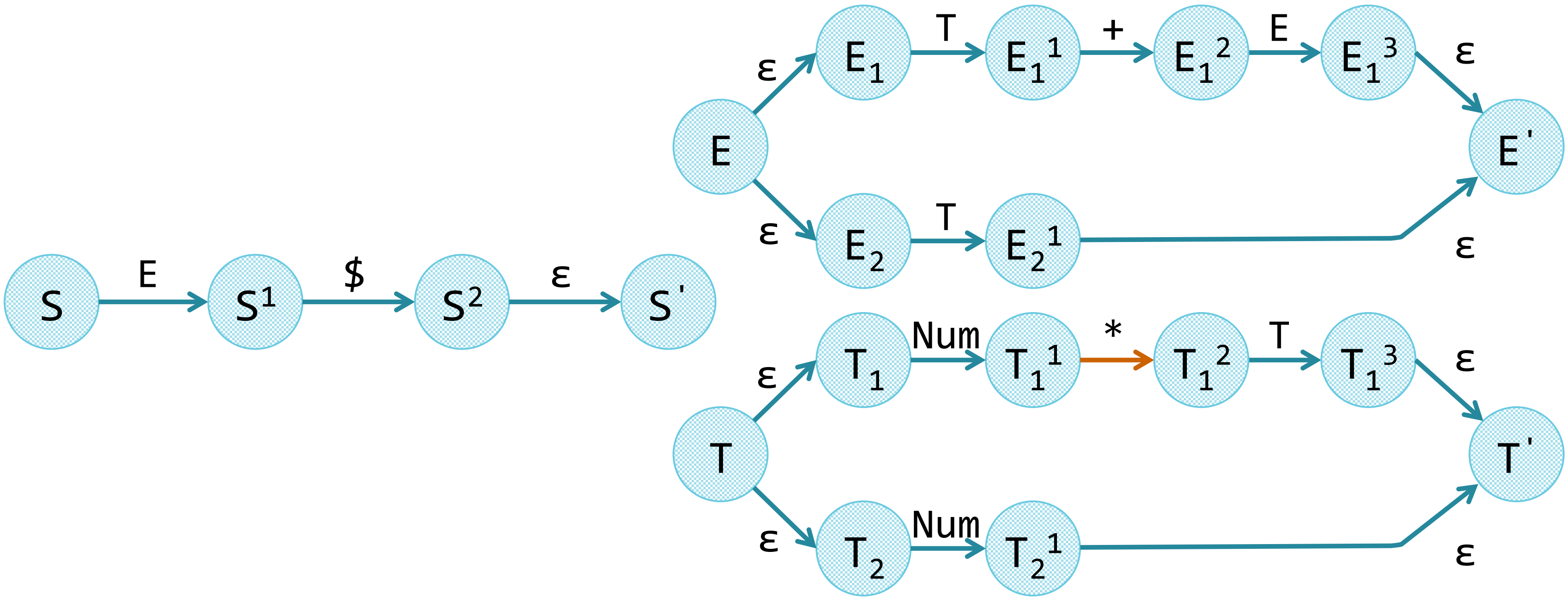


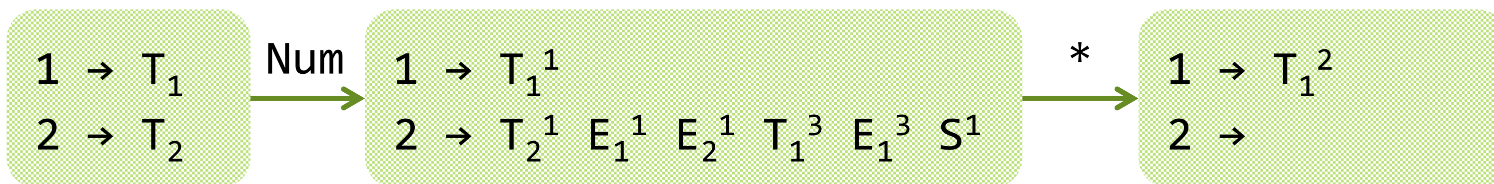
1 → T_1
2 → T_2
Num
1 → T_1^1
2 → T_2^1 E_1^1 E_2^1 T_1^3 E_1^3 S^1

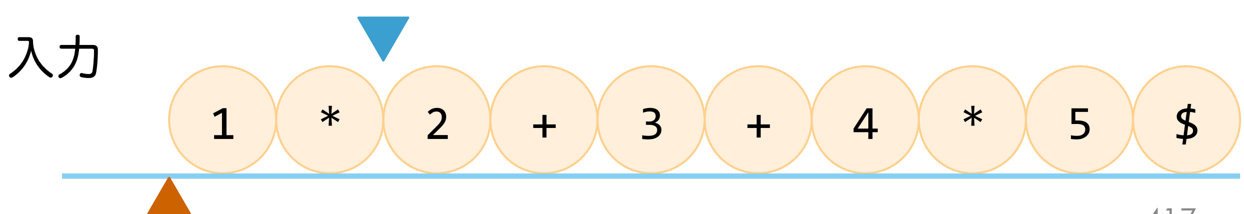
入力

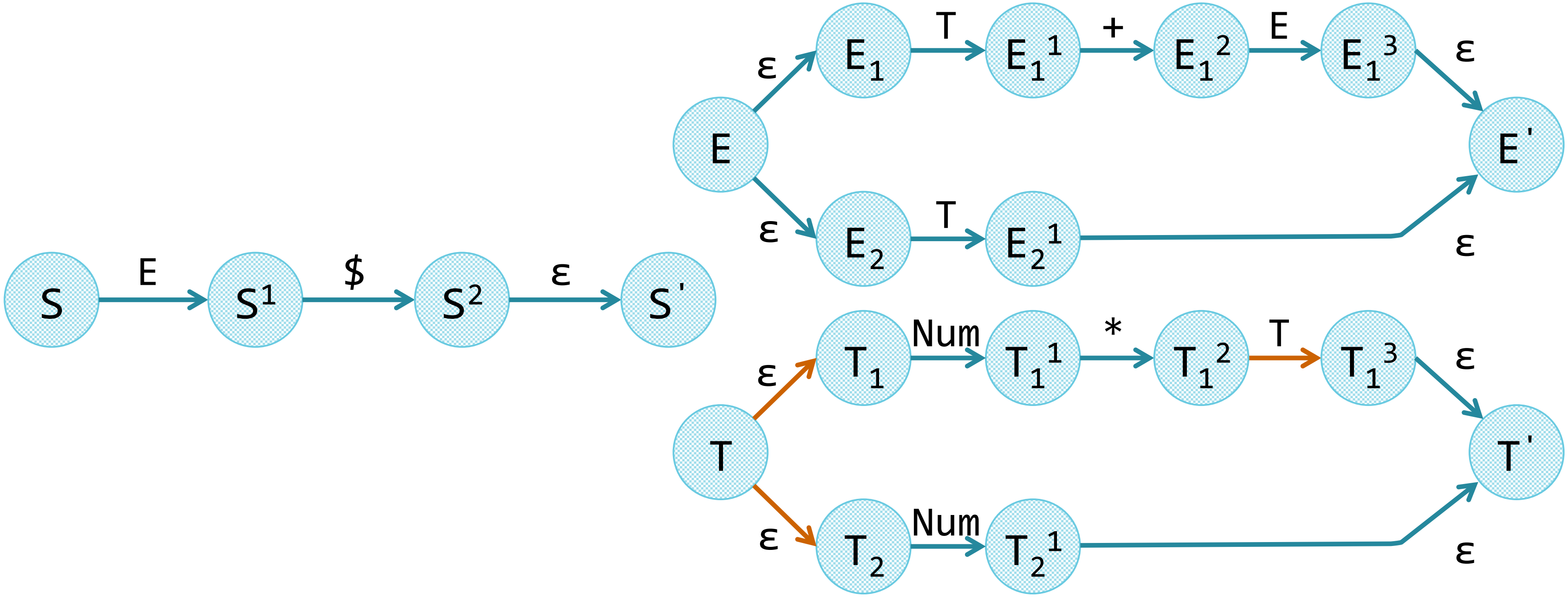


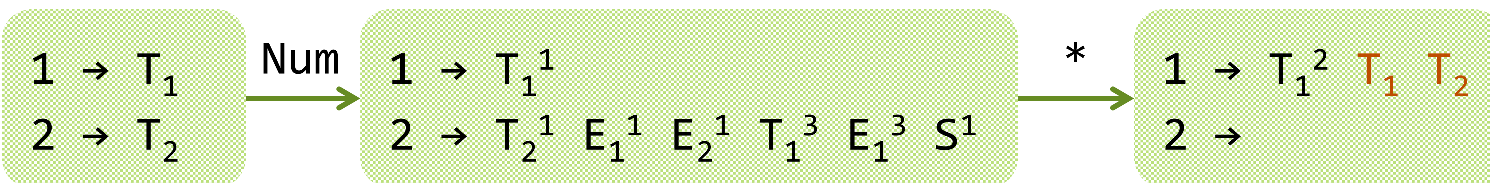
1 * 2 + 3 + 4 * 5 $

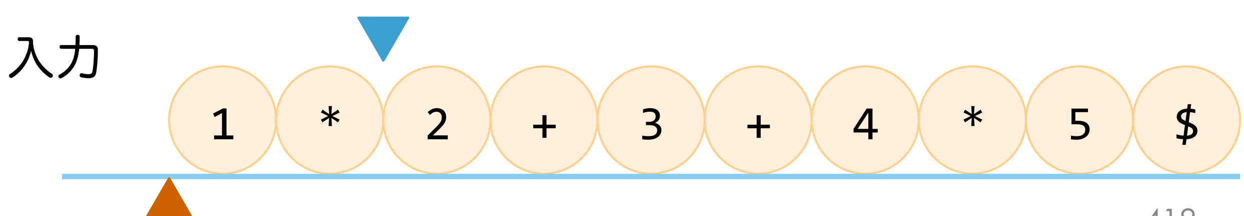

S
E
S¹
$
S²
ε
S'
E
ε
E₁
T
E₁¹
+
E₁²
E
E₁³
ε
E'
ε
E₂
T
E₂¹
ε
T
ε
T₁
Num
T₁¹
*
T₁²
T
T₁³
ε
T'
ε
T₂
Num
T₂¹
ε

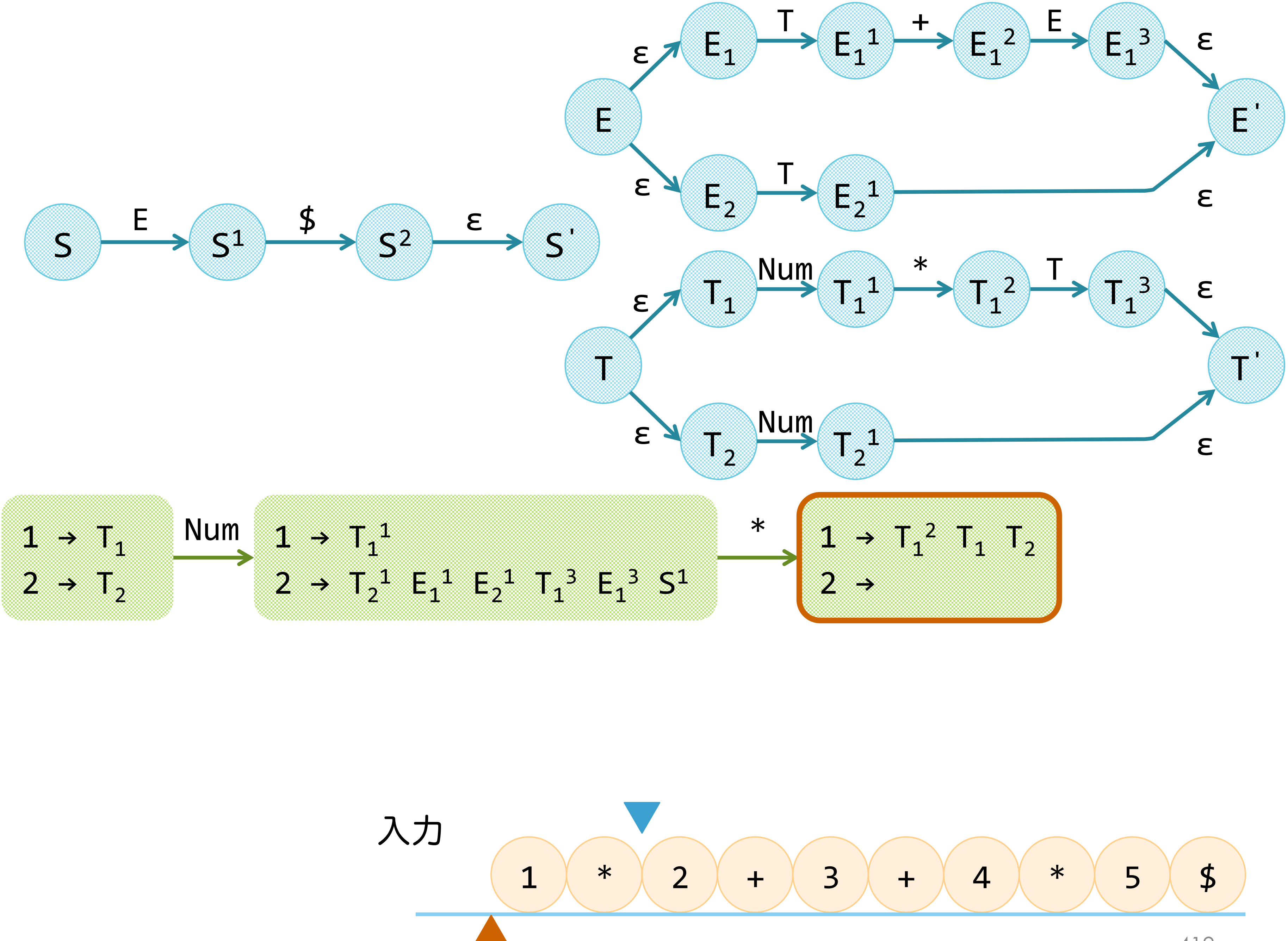

1 → T₁
2 → T₂
Num
1 → T₁¹
2 → T₂¹ E₁¹ E₂¹ T₁³ E₁³ S¹
*
1 → T₁²
2 →


入力
1
*
2
+
3
+
4
*
5
$

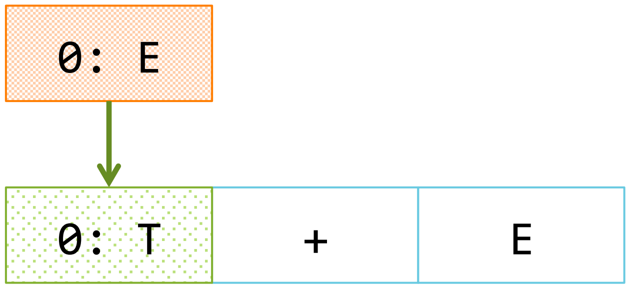

S
E
S1
$
S2
ε
S'
E
ε
E1
T
E1 1
+
E1 2
E
E1 3
ε
E'
ε
E2
T
E2 1
ε
T
ε
T1
Num
T1 1
*
T1 2
T
T1 3
ε
T'
ε
T2
Num
T2 1
ε


1 → T1
2 → T2
Num
1 → T1 1
2 → T2 1 E1 1 E2 1 T1 3 E1 3 S1
*
1 → T1 2 T1 T2
2 →

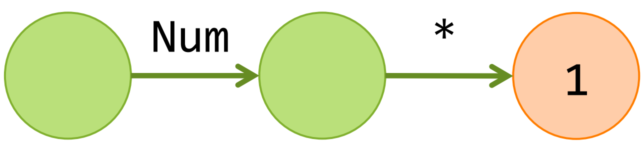

入力
1
*
2
+
3
+
4
*
5
$

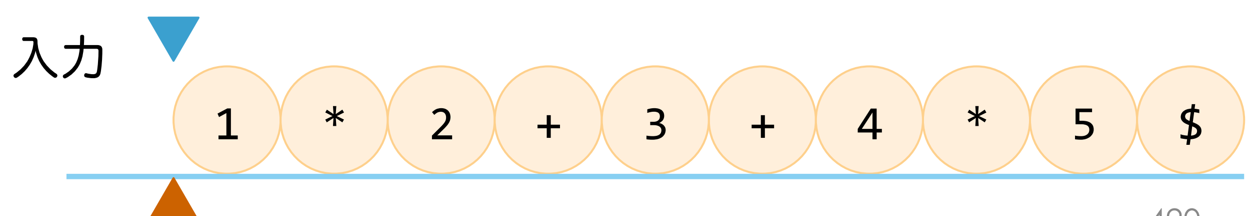


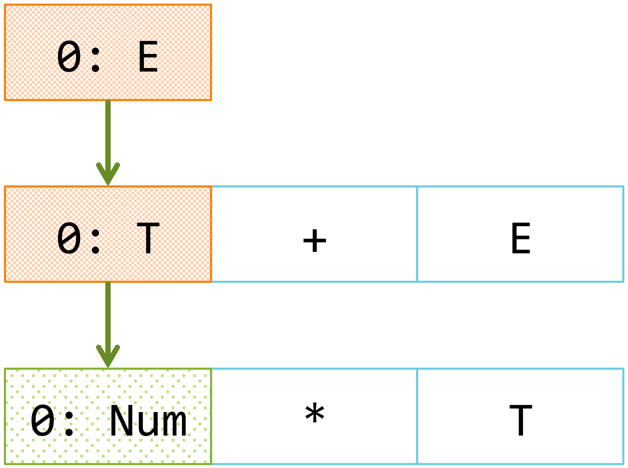
S
E
S1
$
S2
ε
S'
E
ε
E1
T
E11
+
E12
E
E13
ε
E'
ε
E2
T
E21
ε
T
ε
T1
Num
T11
*
T12
T
T13
ε
T'
ε
T2
Num
T21
ε
1 → T1
2 → T2
Num
1 → T11
2 → T21 E11 E21 T13 E13 S1
*
1 → T12 T1 T2
2 →
入力
1
*
2
+
3
+
4
*
5
$



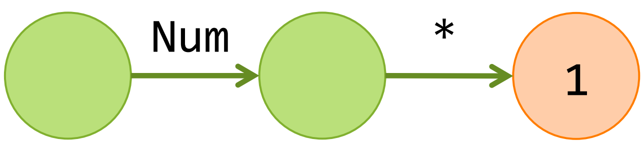
T の先読みオートマトン



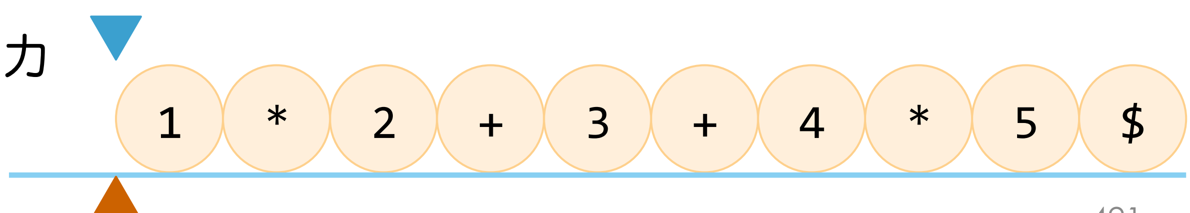


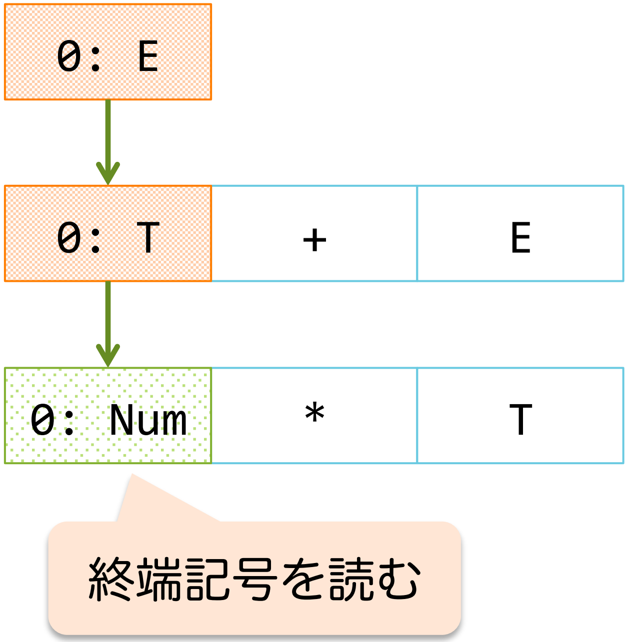


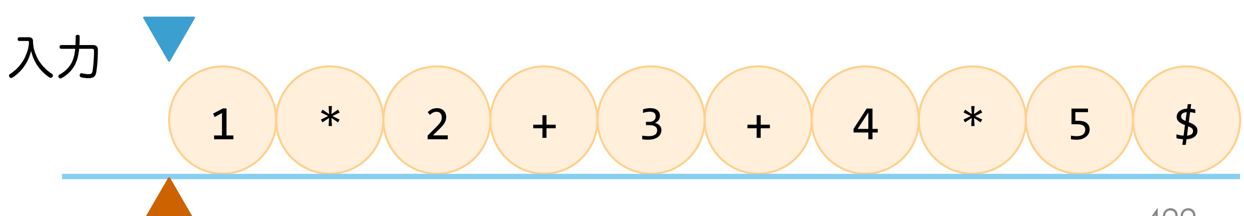
T の先読みオートマトン

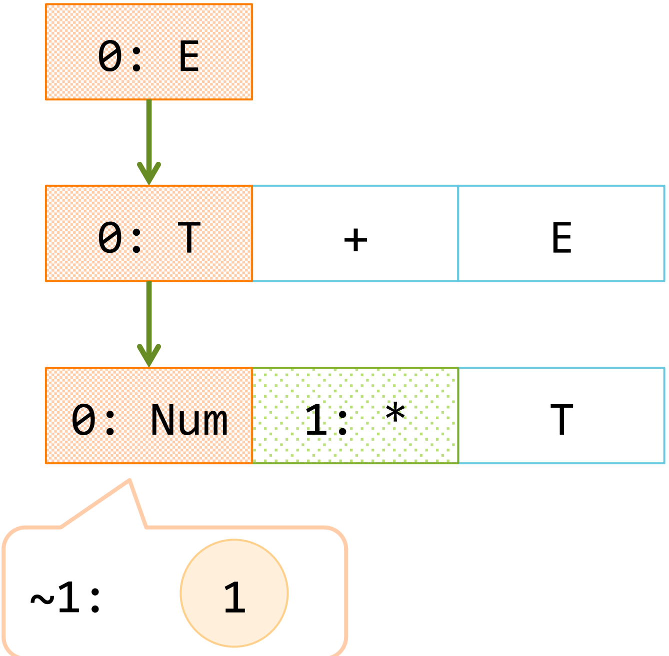


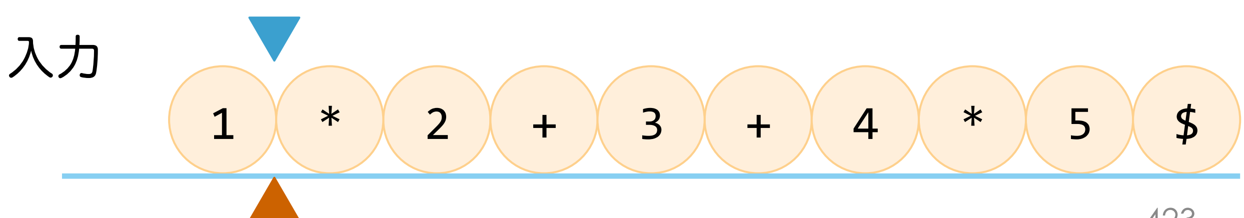
入力

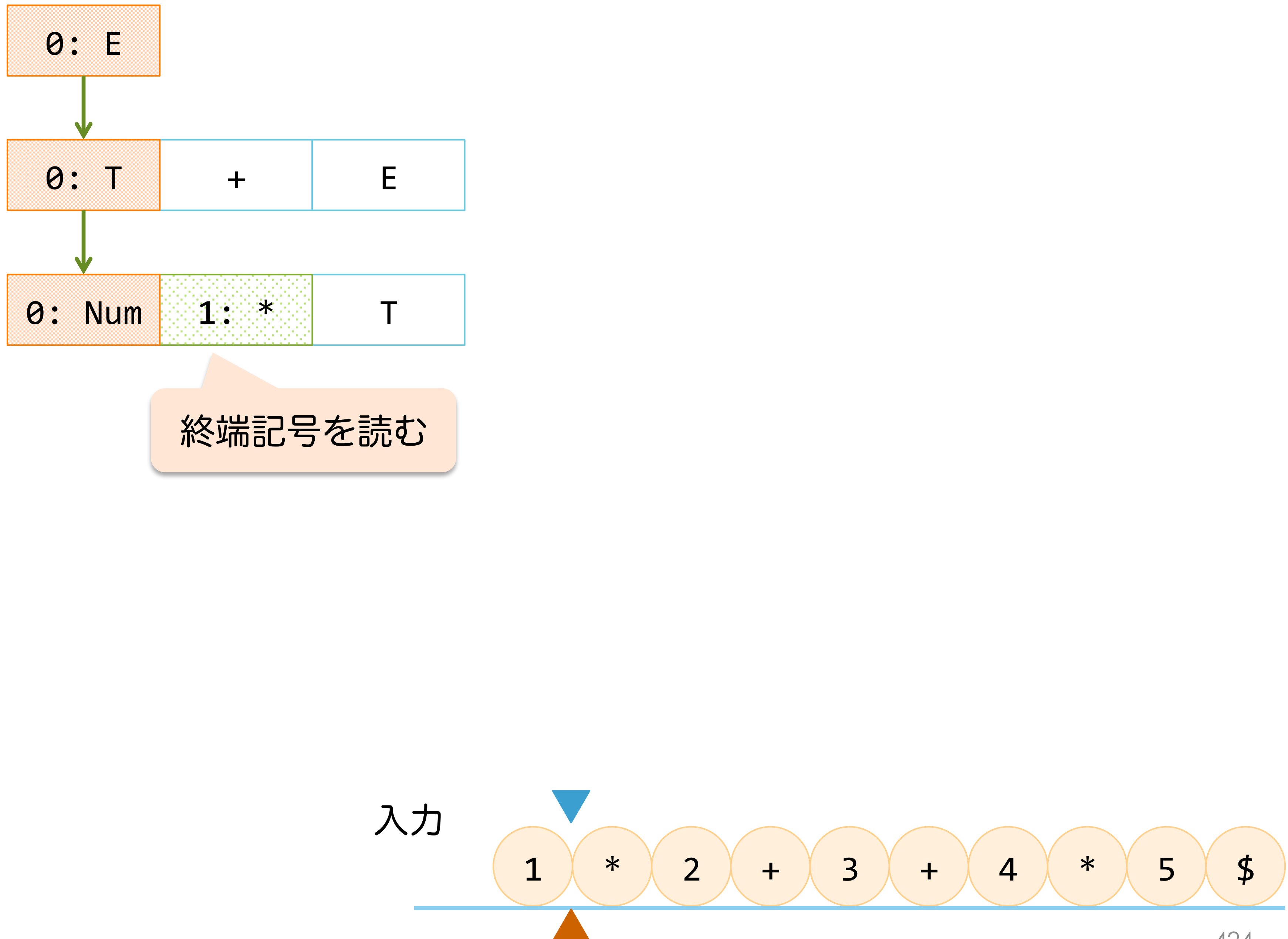


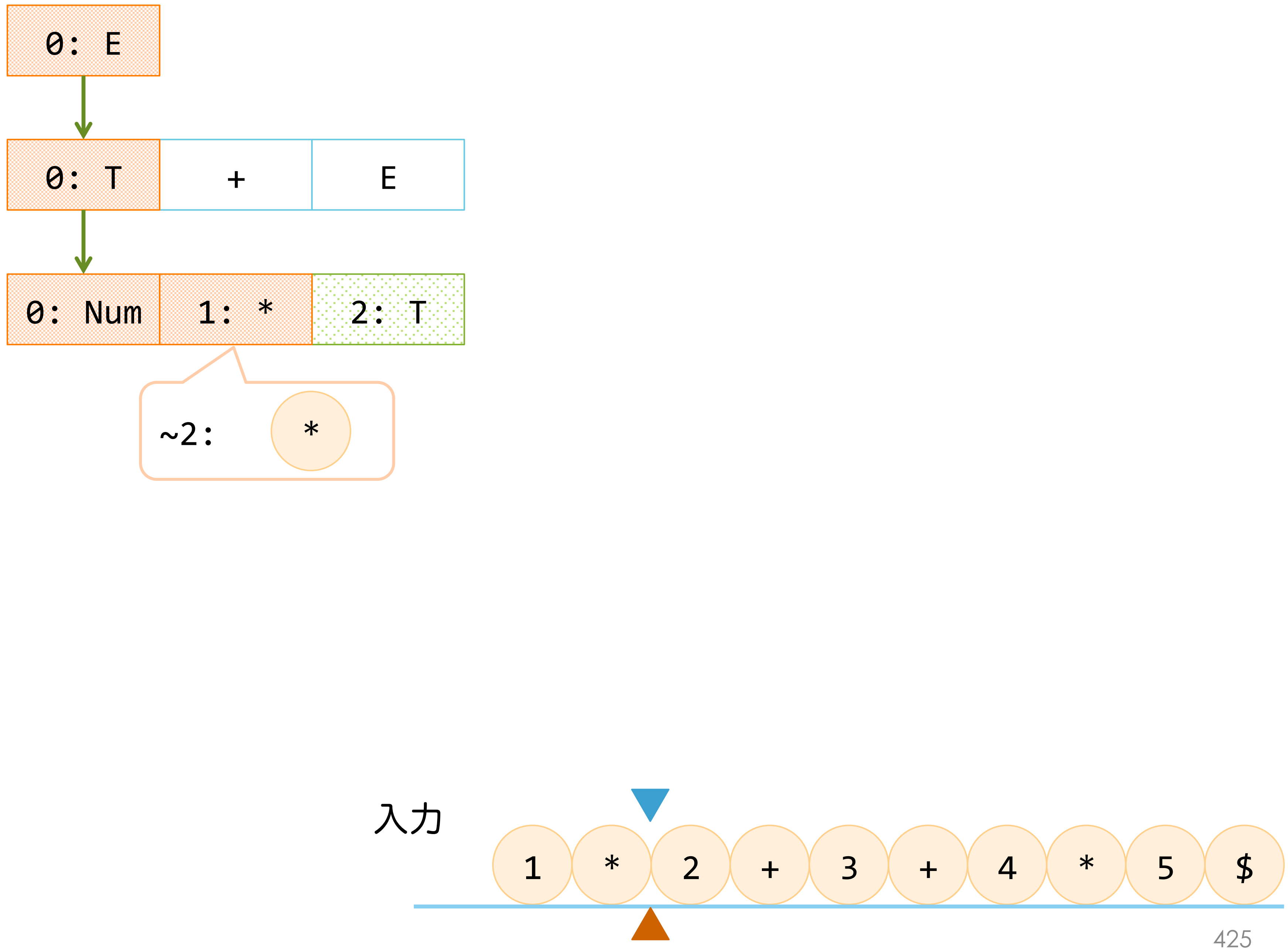

0: E
0: T
+
E
0: Num
*
T
終端記号を読む

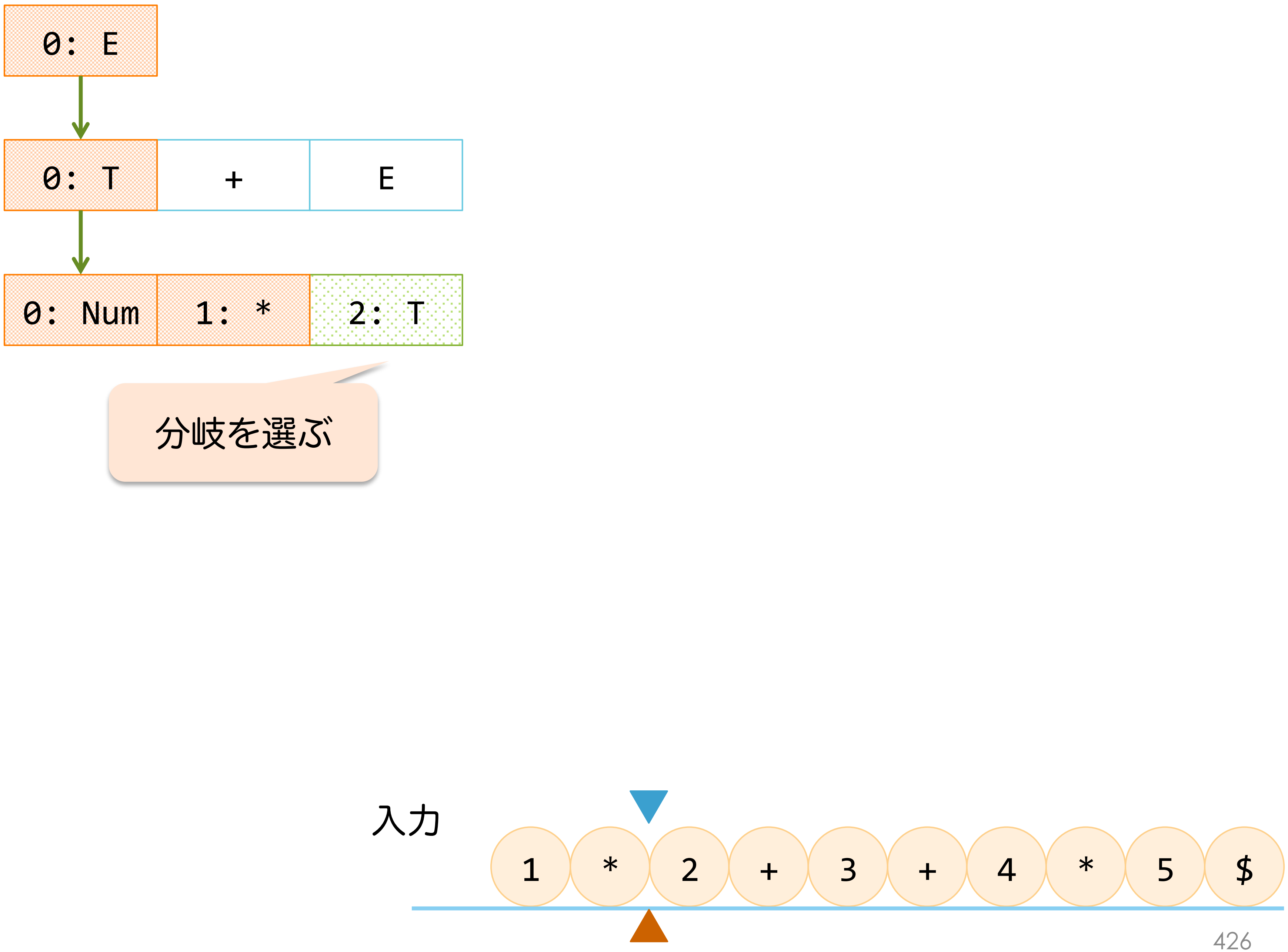

入力
1
*
2
+
3
+
4
*
5
$

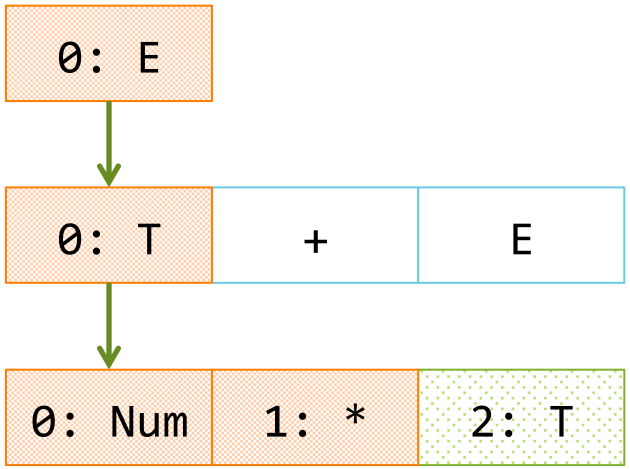

0: E
0: T
+
E
0: Num
1: *
T
~1:
1


入力
1
*
2
+
3
+
4
*
5
$

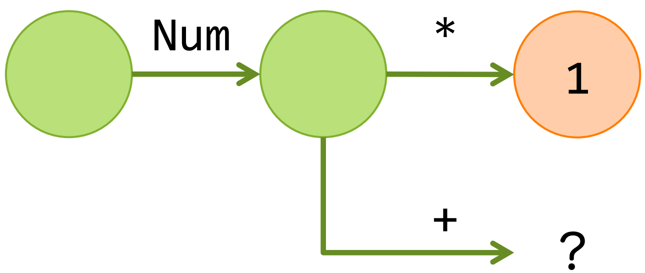

0: E
0: T
+
E
0: Num
1: *
T
終端記号を読む
入力
1
*
2
+
3
+
4
*
5
$


0: E
0: T
+
E
0: Num
1: *
2: T
~2:
*
入力
1
*
2
+
3
+
4
*
5
$

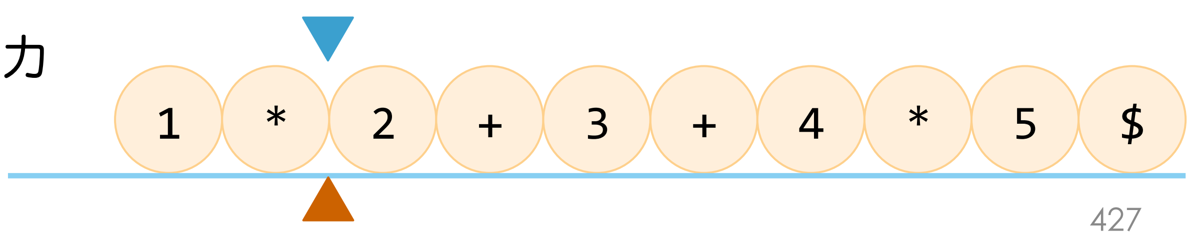

0: E
0: T
+
E
0: Num
1: *
2: T
分岐を選ぶ
入力
1
*
2
+
3
+
4
*
5
$

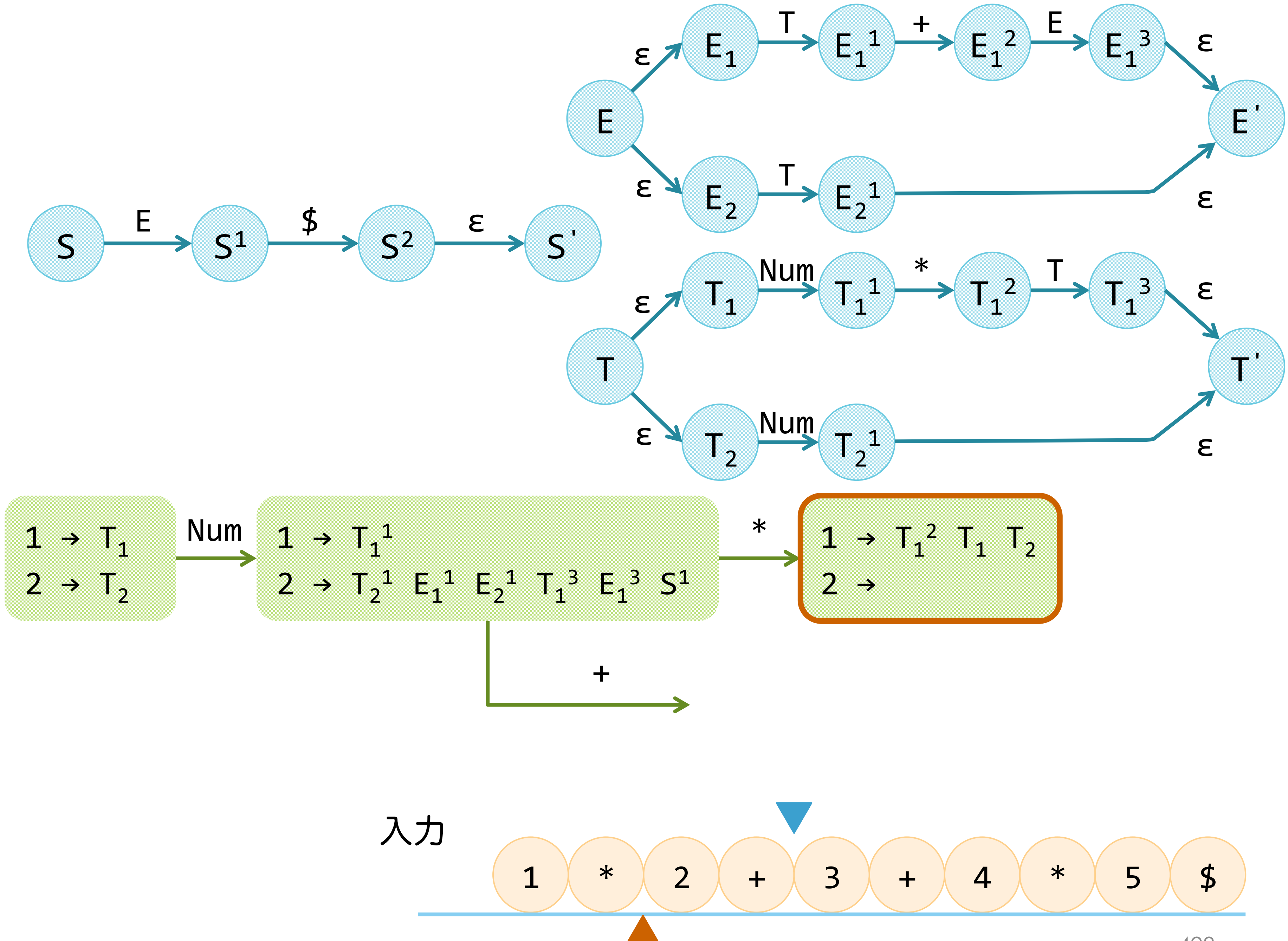


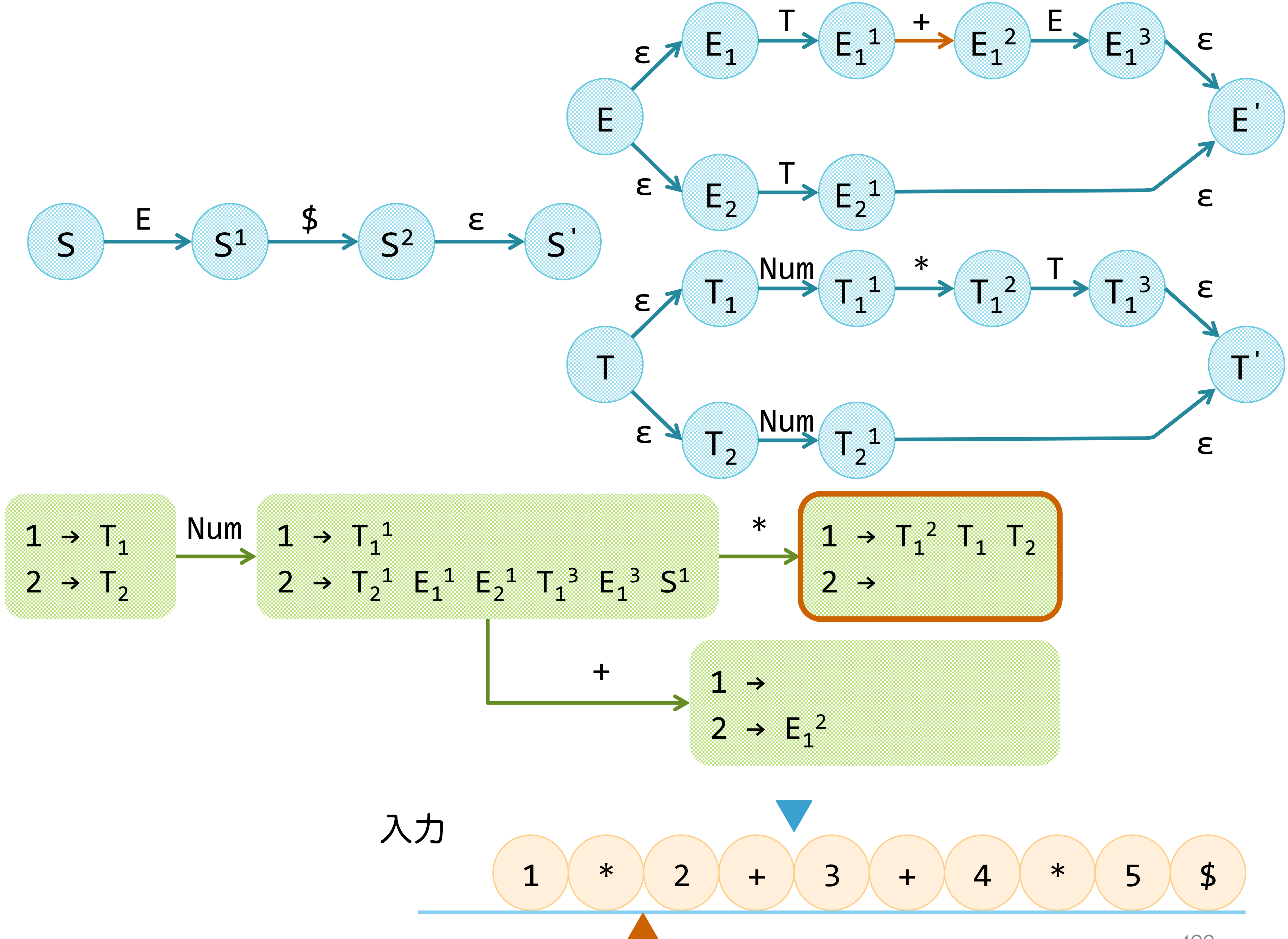
T の先読みオートマトン



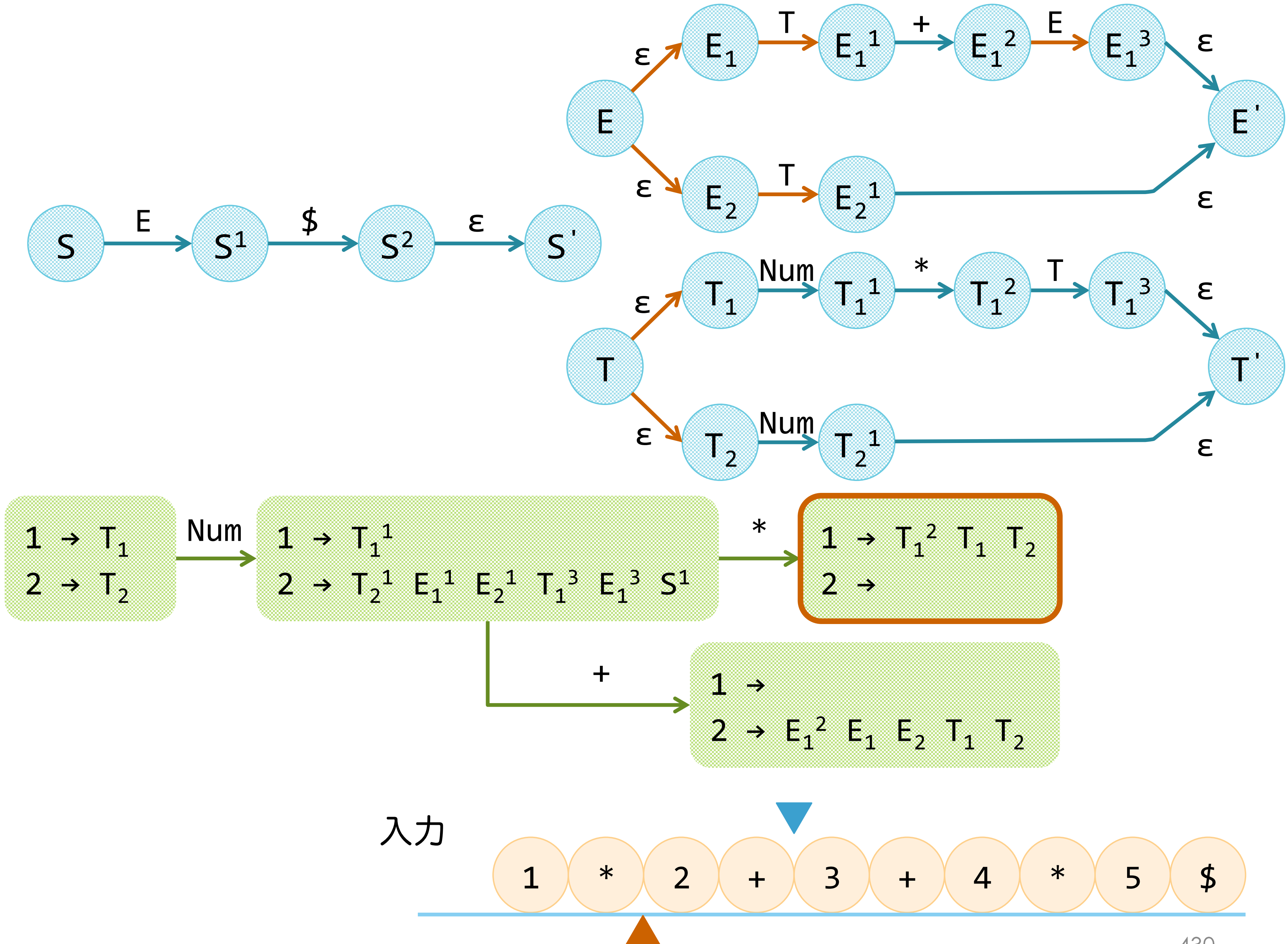
入力

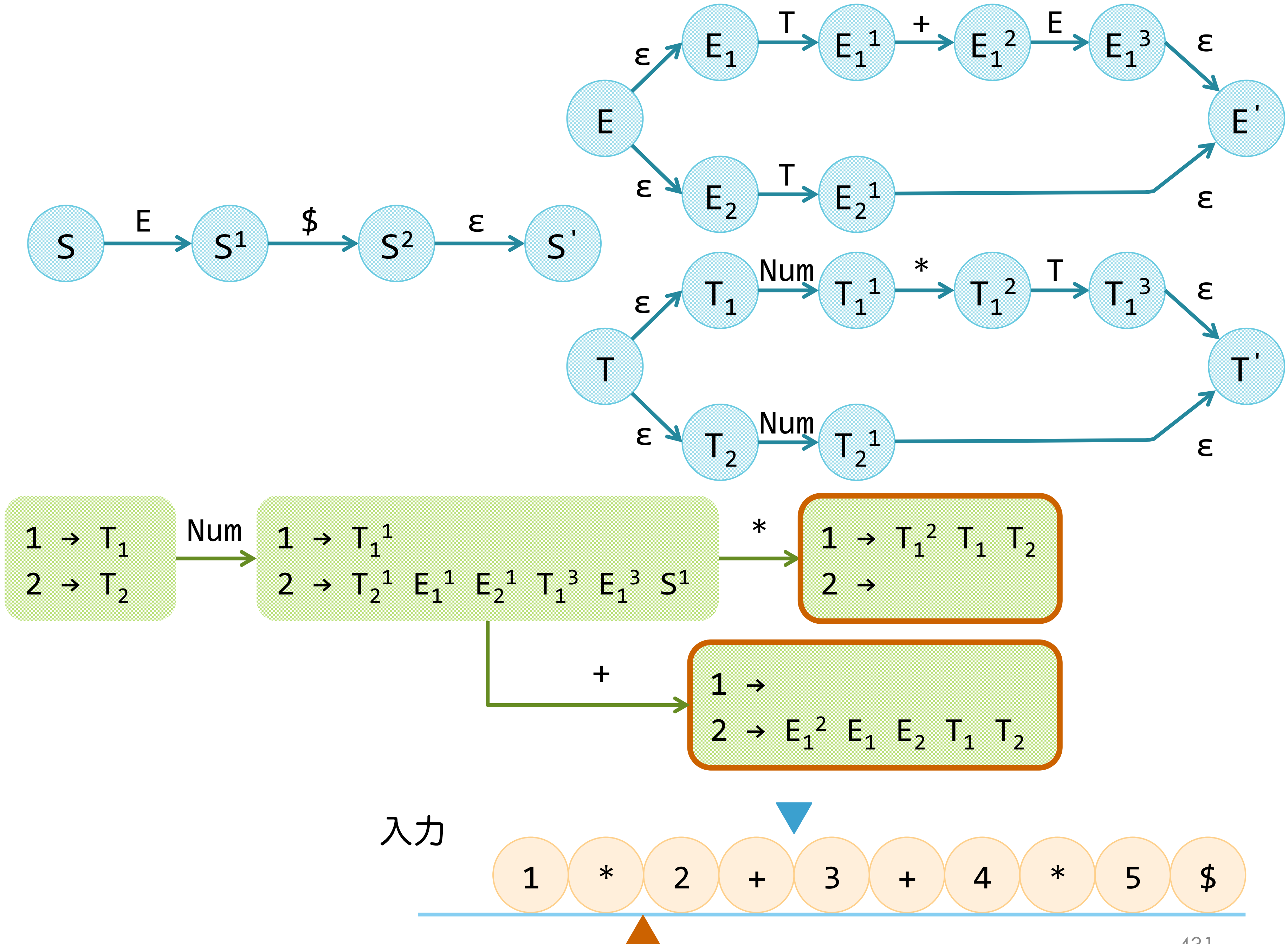


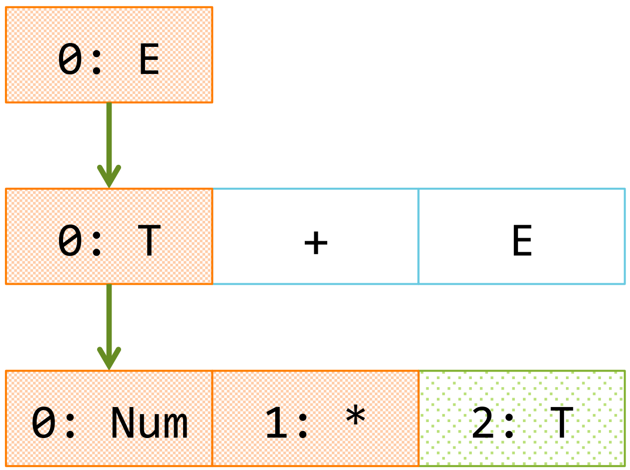

S
E
S1
$
S2
ε
S'
E
ε
E1
T
E1 1
+
E1 2
E
E1 3
ε
E'
ε
E2
T
E2 1
ε
T
ε
T1
Num
T1 1
*
T1 2
T
T1 3
ε
T'
ε
T2
Num
T2 1
ε
1 → T1
2 → T2
Num
1 → T1 1
2 → T2 1 E1 1 E2 1 T1 3 E1 3 S1
*
1 → T1 2 T1 T2
2 →
+
入力
1 * 2 + 3 + 4 * 5 $



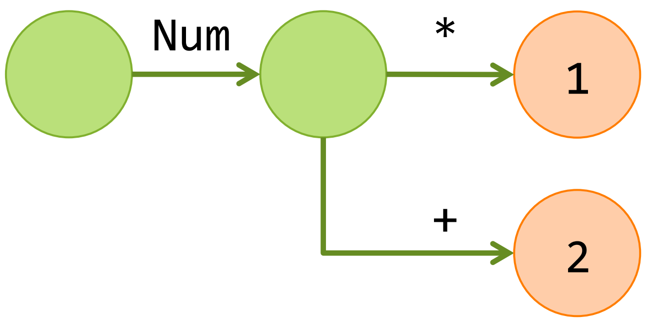
S
E
S1
$
S2
ε
S'
E
ε
E1
T
E1 1
+
E1 2
E
E1 3
ε
E'
ε
E2
T
E2 1
ε
T
ε
T1
Num
T1 1
*
T1 2
T
T1 3
ε
T'
ε
T2
Num
T2 1
ε
1 → T1
2 → T2
Num
1 → T1 1
2 → T2 1 E1 1 E2 1 T1 3 E1 3 S1
*
1 → T1 2 T1 T2
2 →
+
1 →
2 → E1 2
入力
1 * 2 + 3 + 4 * 5 $


S
E
S1
$
S2
ε
S'
E
ε
E1
T
E1 1
+
E1 2
E
E1 3
ε
E'
ε
E2
T
E2 1
ε
T
ε
T1
Num
T1 1
*
T1 2
T
T1 3
ε
T'
ε
T2
Num
T2 1
ε
1 → T1
2 → T2
Num
1 → T1 1
2 → T2 1 E1 1 E2 1 T1 3 E1 3 S1
*
1 → T1 2 T1 T2
2 →
+
1 →
2 → E1 2 E1 E2 T1 T2
入力
1
*
2
+
3
+
4
*
5
$

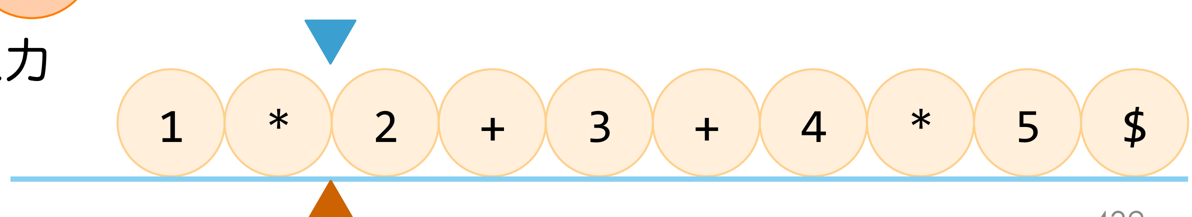

S
S¹
S²
S'
E
$
ε
E
E₁
E₁¹
E₁²
E₁³
E₂
E₂¹
E'
T
+
T
T₁
T₁¹
T₁²
T₁³
T₂
T₂¹
T'
Num
*
1 → T₁
2 → T₂
1 → T₁¹
2 → T₂¹ E₁¹ E₂¹ T₁³ E₁³ S¹
1 → T₁² T₁ T₂
2 →
1 →
2 → E₁² E₁ E₂ T₁ T₂
入力
1 * 2 + 3 + 4 * 5 $

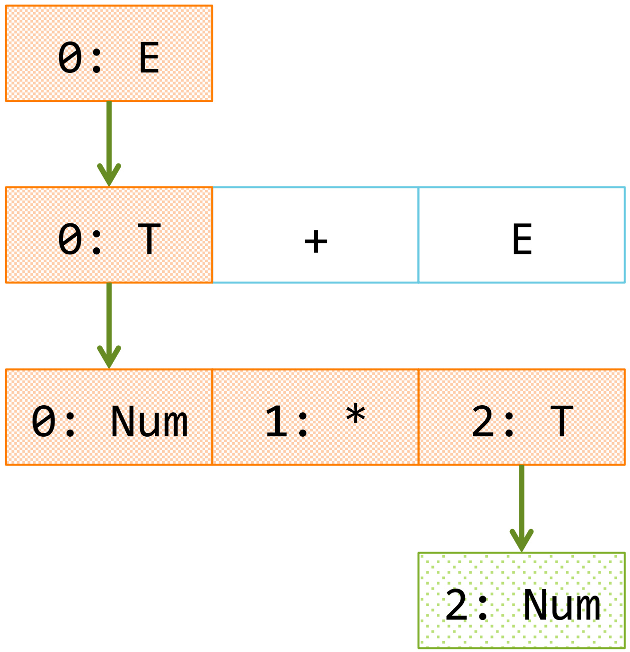


T の先読みオートマトン

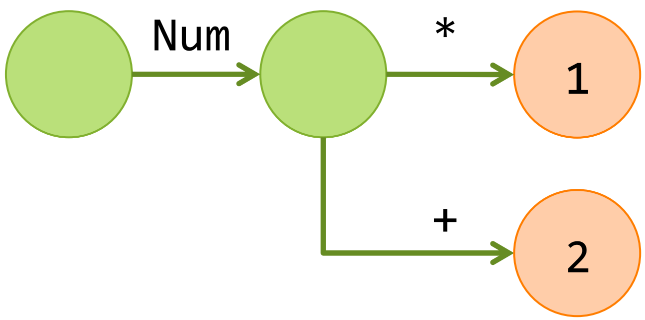


入力

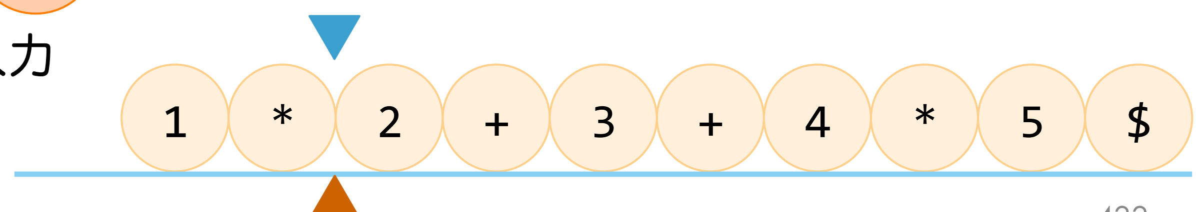


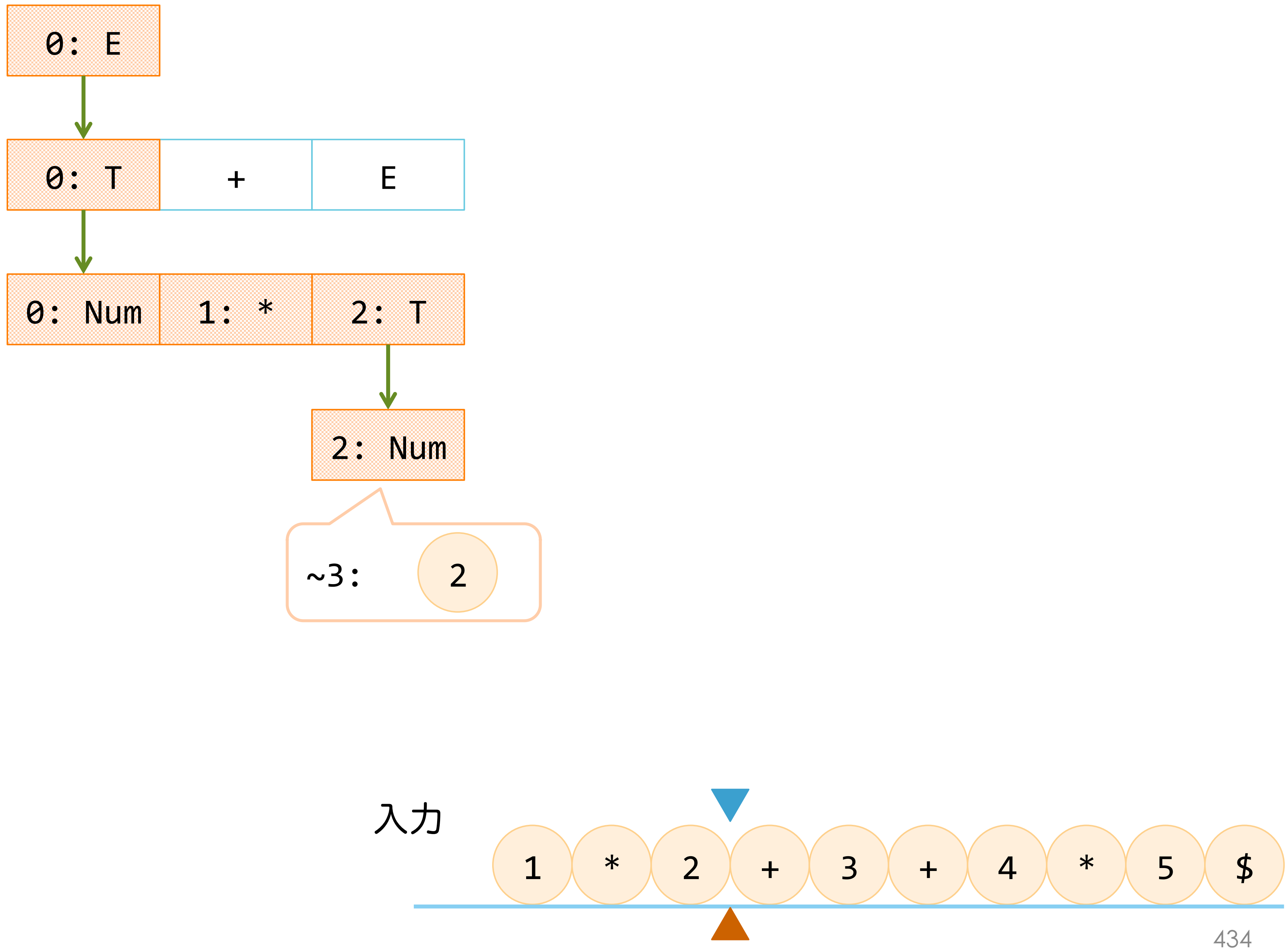


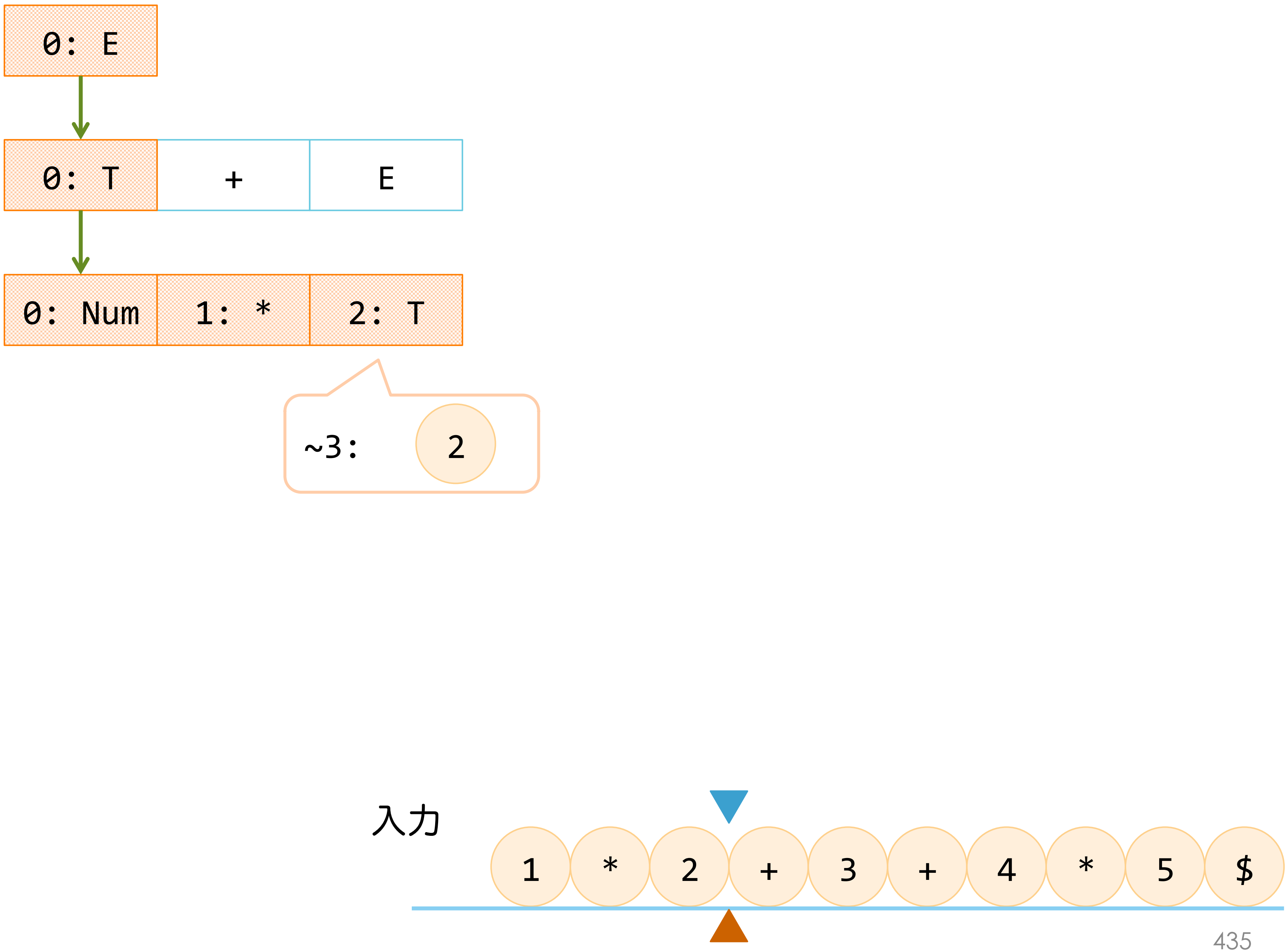
T の先読みオートマトン



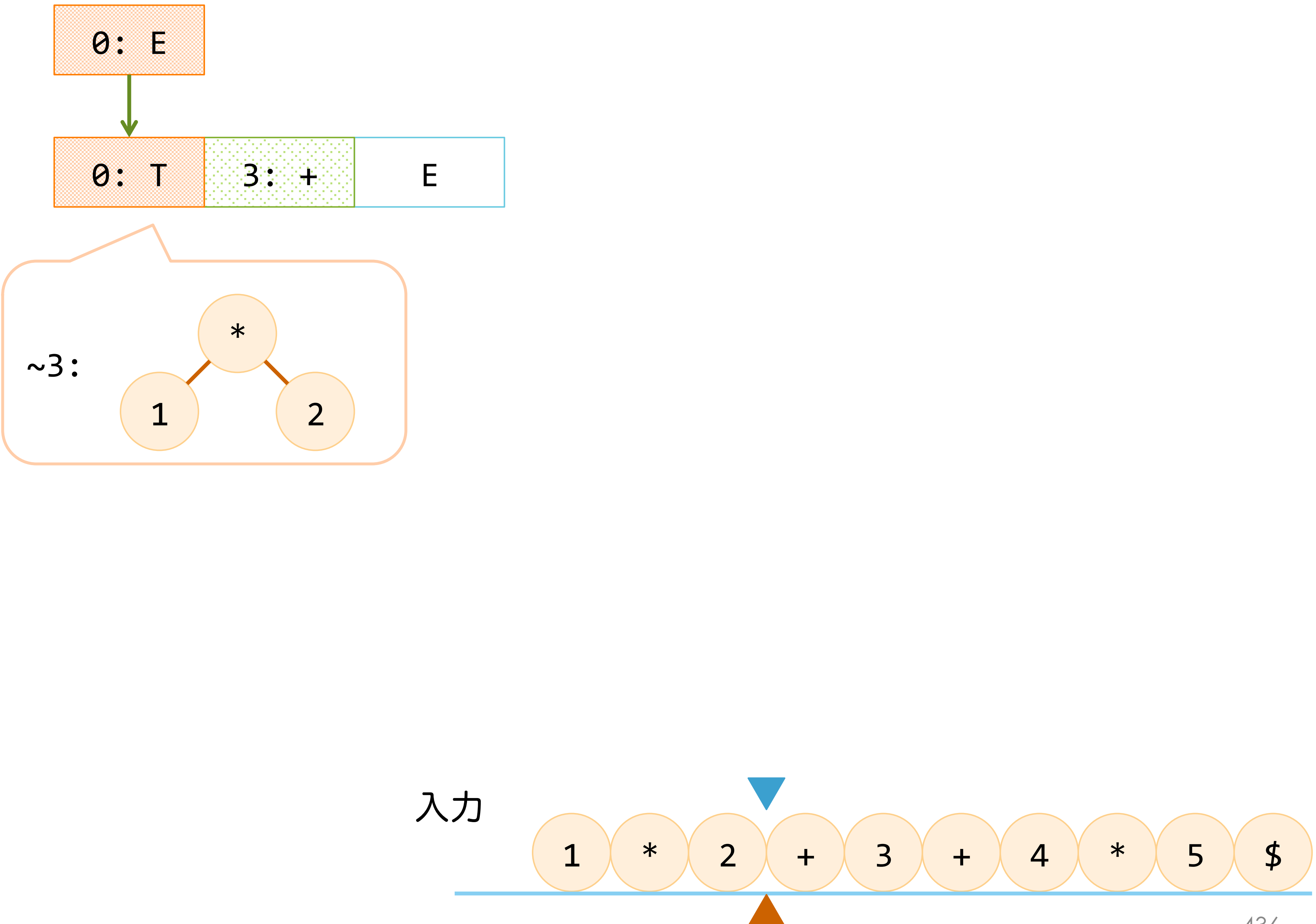
入力



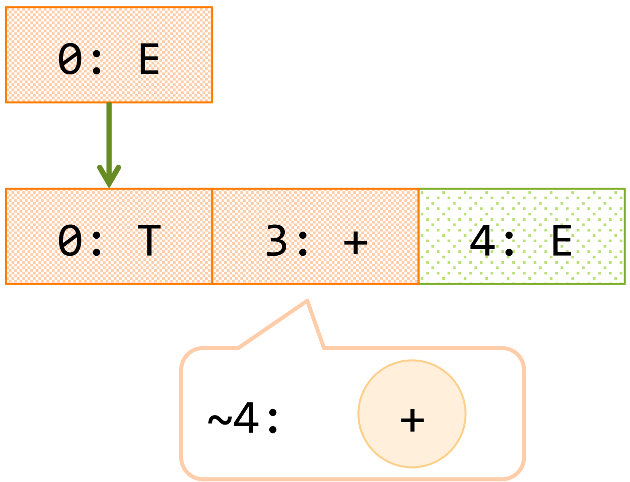

0: E
0: T
+
E
0: Num
1: *
2: T
2: Num
~3:
2
入力
1
*
2
+
3
+
4
*
5
$

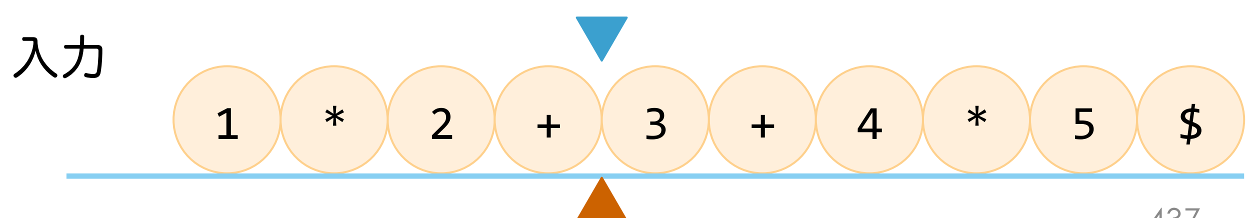


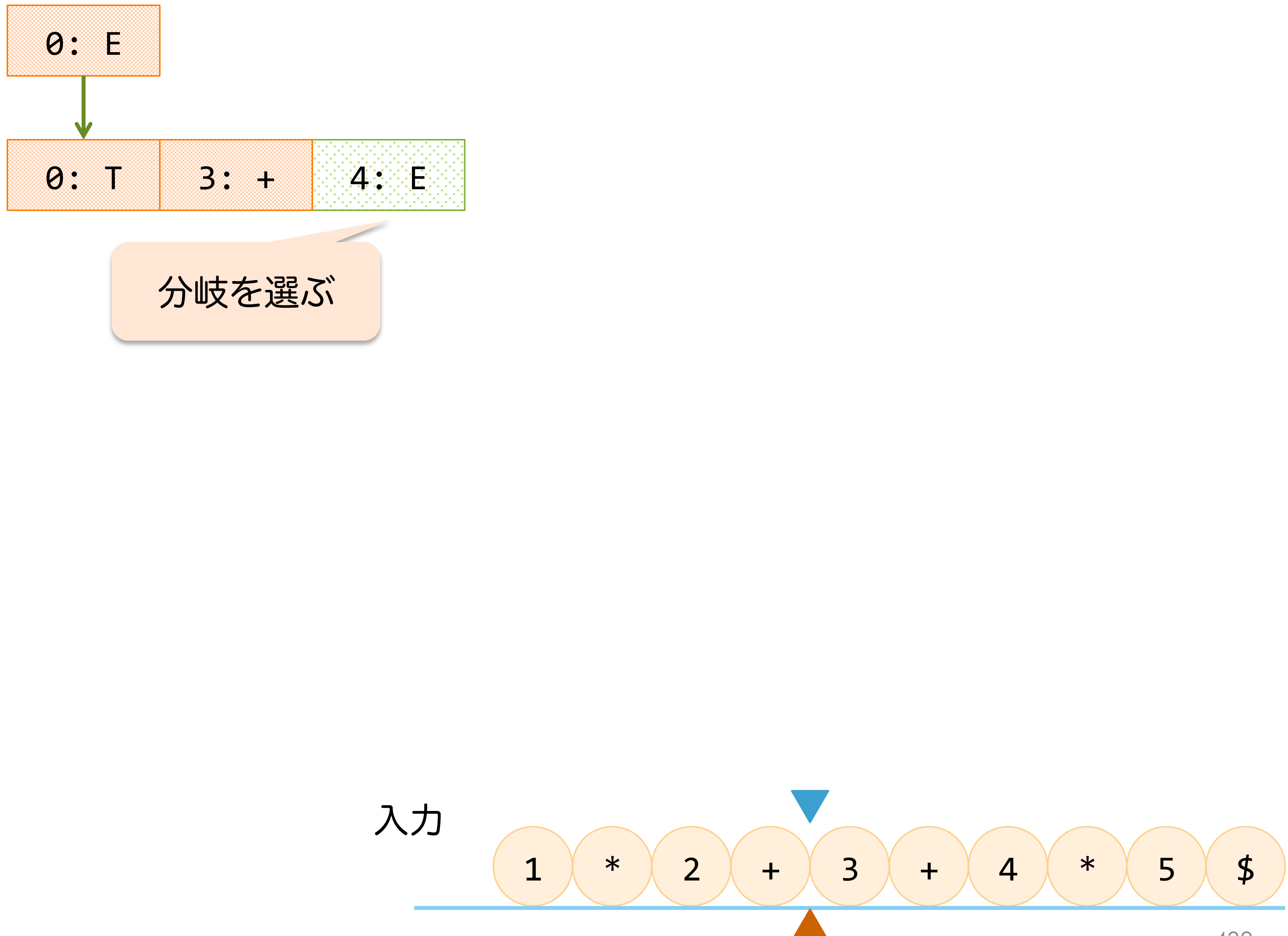
0: E
0: T
+
E
0: Num
1: *
2: T
~3:
2
入力
1
*
2
+
3
+
4
*
5
$

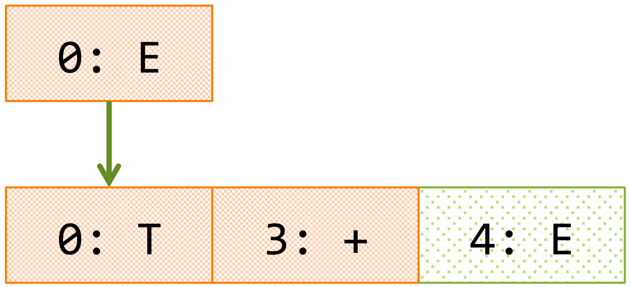


0: E
0: T
3: +
E
~3:
*
1
2
入力
1
*
2
+
3
+
4
*
5
$

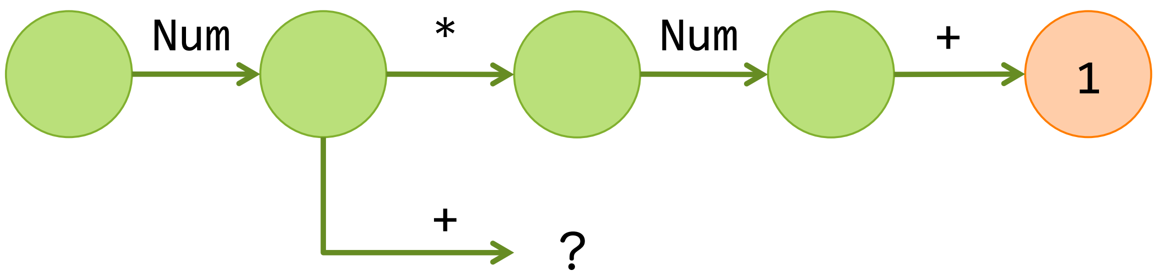

0: E
0: T
3: +
4: E
~4:
+


入力
1
*
2
+
3
+
4
*
5
$

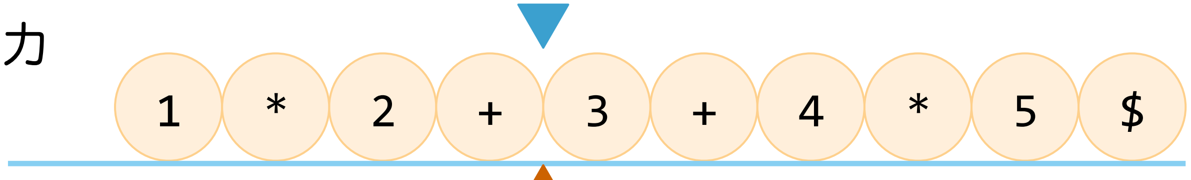

0: E
0: T
3: +
4: E
分岐を選ぶ
入力
1
*
2
+
3
+
4
*
5
$

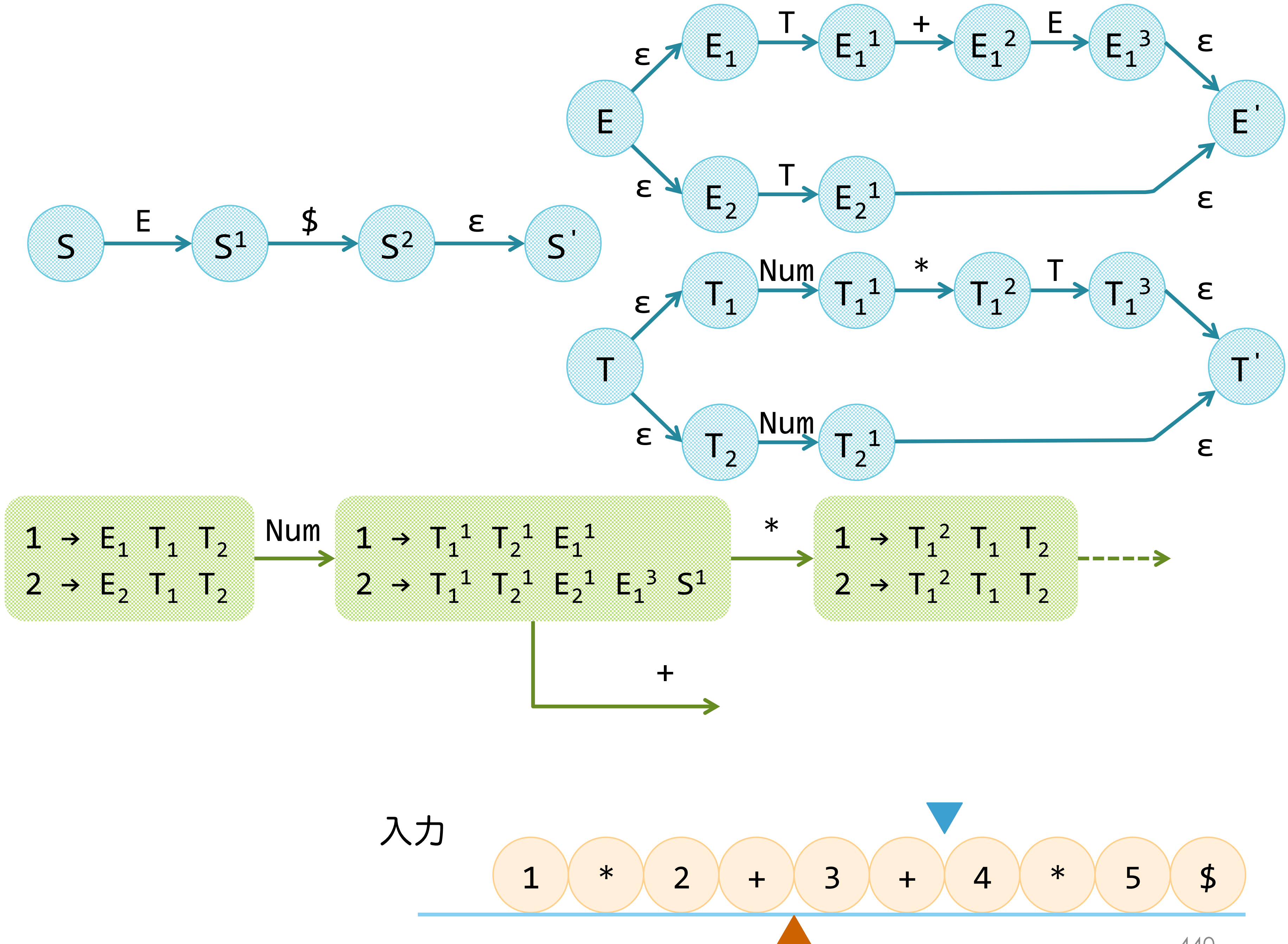


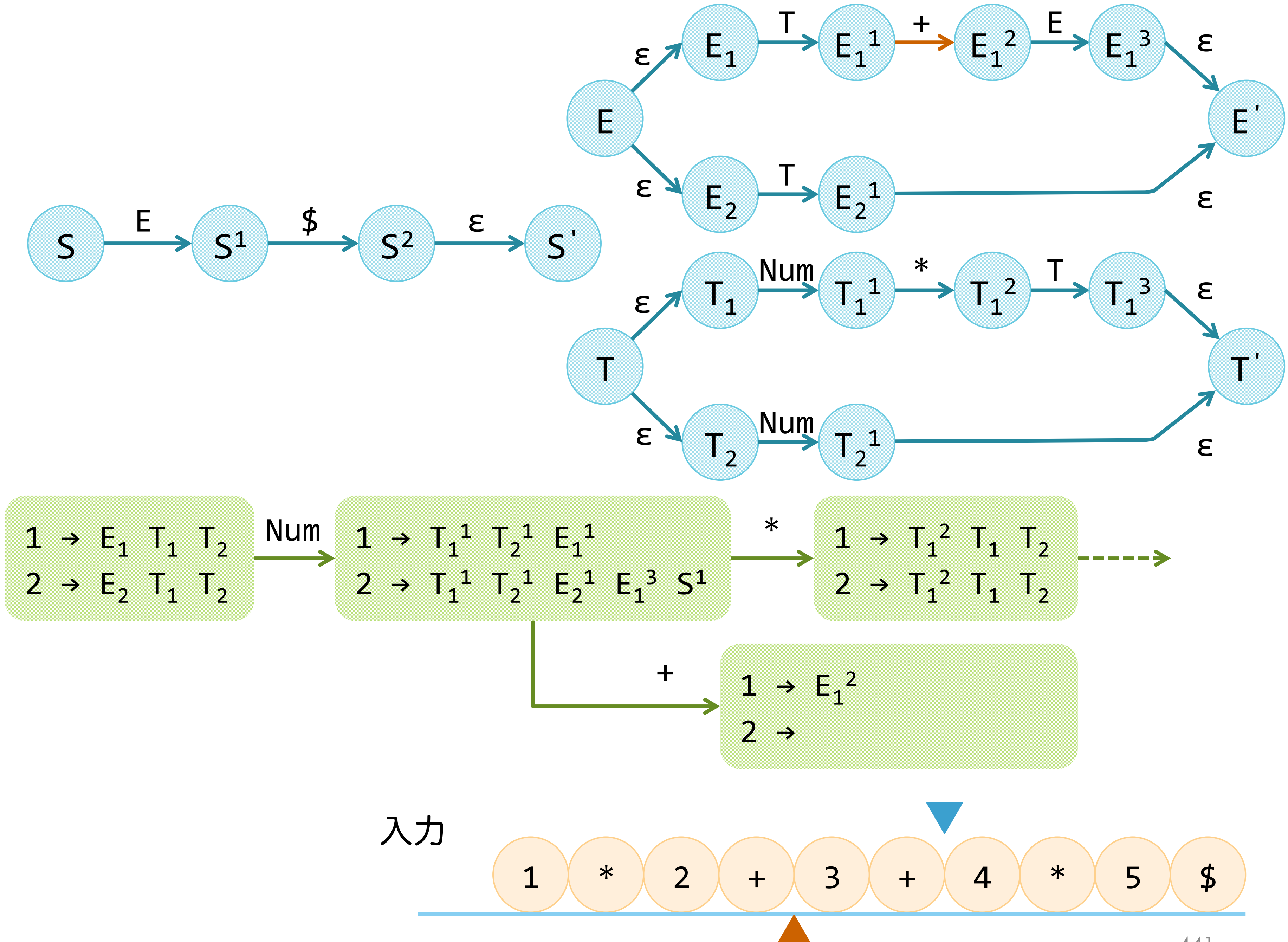
E の先読みオートマトン

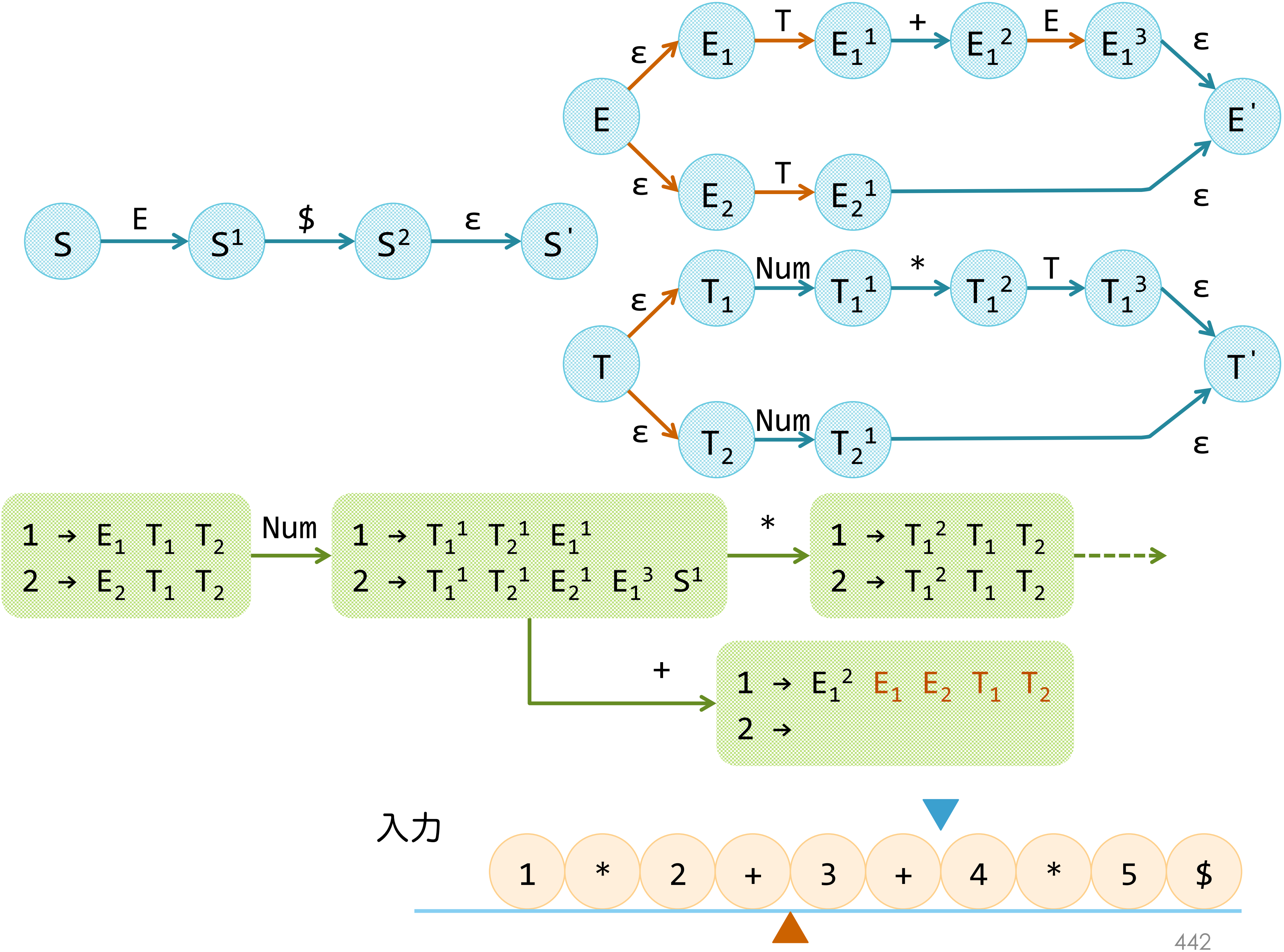


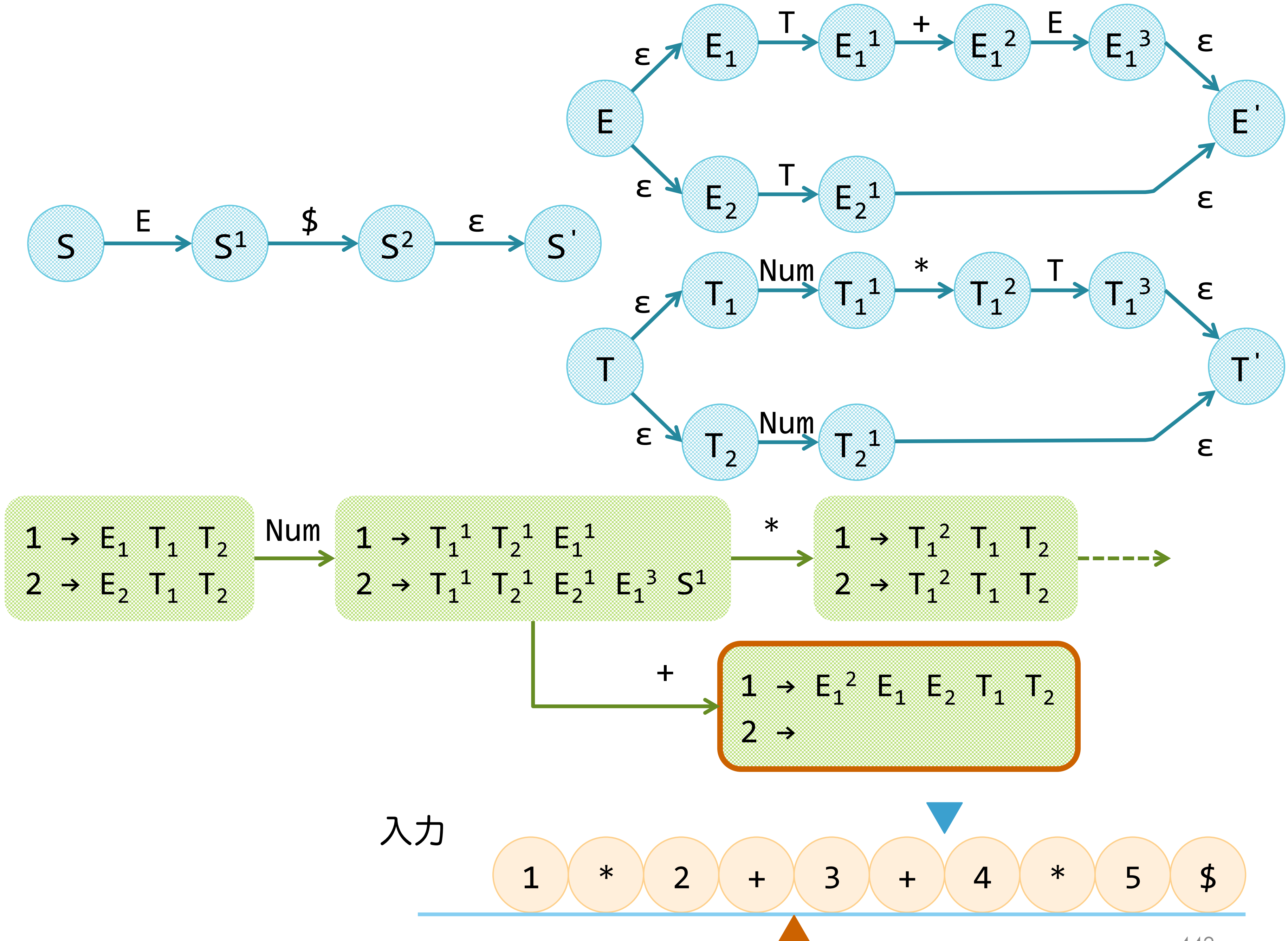
入力



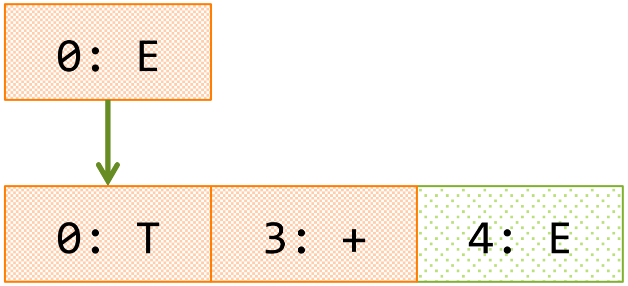

S
E
S1
$
S2
ε
S'
E
ε
E1
T
E1 1
+
E1 2
E
E1 3
ε
E'
ε
E2
T
E2 1
ε
T
ε
T1
Num
T1 1
*
T1 2
T
T1 3
ε
T'
ε
T2
Num
T2 1
ε
1 → E1 T1 T2
2 → E2 T1 T2
Num
1 → T1 1 T2 1 E1 1
2 → T1 1 T2 1 E2 1 E1 3 S1
*
1 → T1 2 T1 T2
2 → T1 2 T1 T2
+
入力
1
*
2
+
3
+
4
*
5
$


S
E
S^1
$
S^2
ε
S'
E
E_1
T
E_1^1
+
E_1^2
E
E_1^3
E'
E_2
E_2^1
T
T_1
Num
T_1^1
*
T_1^2
T_1^3
T'
T_2
T_2^1
1 → E_1 T_1 T_2
2 → E_2 T_1 T_2
Num
1 → T_1^1 T_2^1 E_1^1
2 → T_1^1 T_2^1 E_2^1 E_1^3 S^1
*
1 → T_1^2 T_1 T_2
2 → T_1^2 T_1 T_2
+
1 → E_1^2
2 →
入力
1 * 2 + 3 + 4 * 5 $

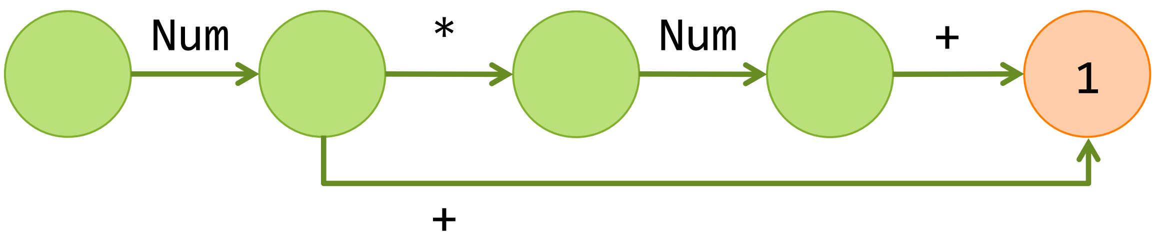

S
E
S1
$
S2
ε
S'
E
ε
E1
T
E1 1
+
E1 2
E
E1 3
ε
E'
ε
E2
T
E2 1
ε
T
ε
T1
Num
T1 1
*
T1 2
T
T1 3
ε
T'
ε
T2
Num
T2 1
ε
1 → E1 T1 T2
2 → E2 T1 T2
Num
1 → T1 1 T2 1 E1 1
2 → T1 1 T2 1 E2 1 E1 3 S1
*
1 → T1 2 T1 T2
2 → T1 2 T1 T2
+
1 → E1 2 E1 E2 T1 T2
2 →
入力
1 * 2 + 3 + 4 * 5 $



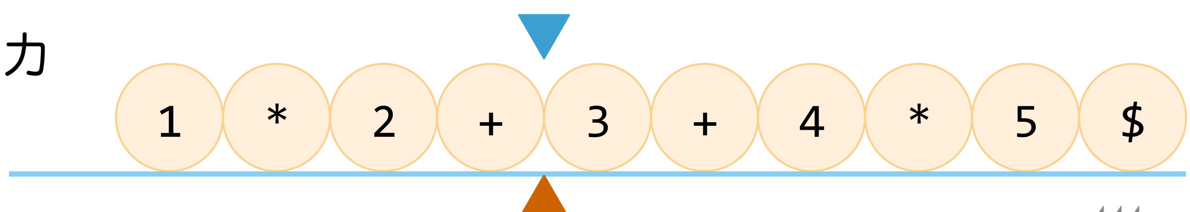
S
E
S¹
$
S²
ε
S'
E
ε
E1
T
E1 1
+
E1 2
E
E1 3
ε
E'
ε
E2
T
E2 1
ε
T
ε
T1
Num
T1 1
*
T1 2
T
T1 3
ε
T'
ε
T2
Num
T2 1
ε
1 → E1 T1 T2
2 → E2 T1 T2
Num
1 → T1 1 T2 1 E1 1
2 → T1 1 T2 1 E2 1 E1 3 S1
*
1 → T1 2 T1 T2
2 → T1 2 T1 T2
+
1 → E1 2 E1 E2 T1 T2
2 →
入力
1 * 2 + 3 + 4 * 5 $

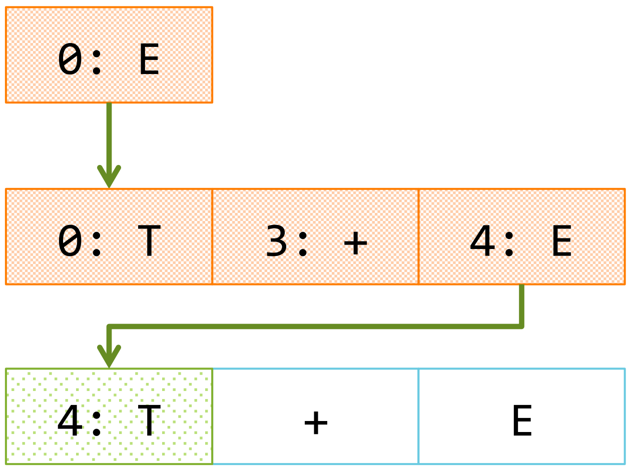


E の先読みオートマトン

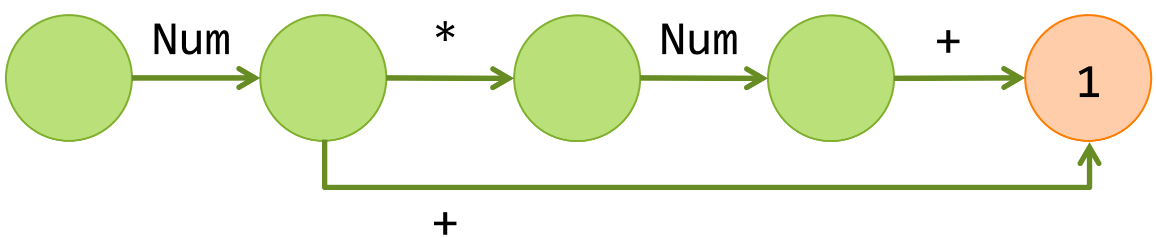


入力

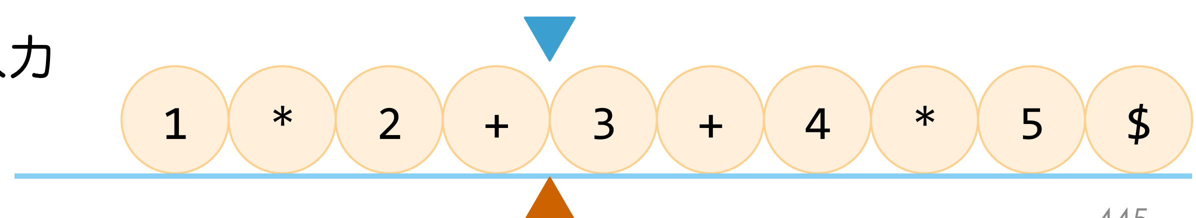


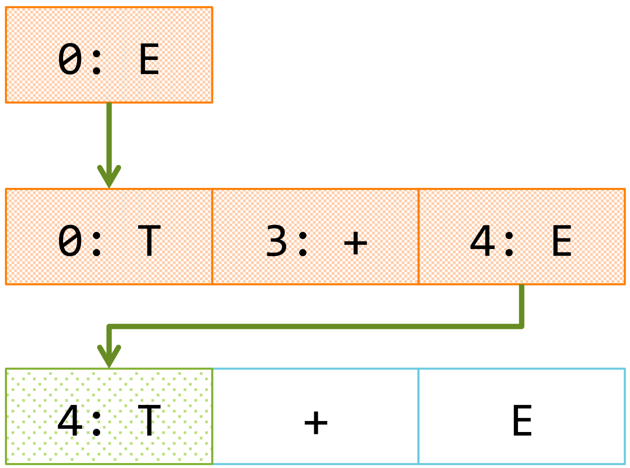


E の先読みオートマトン

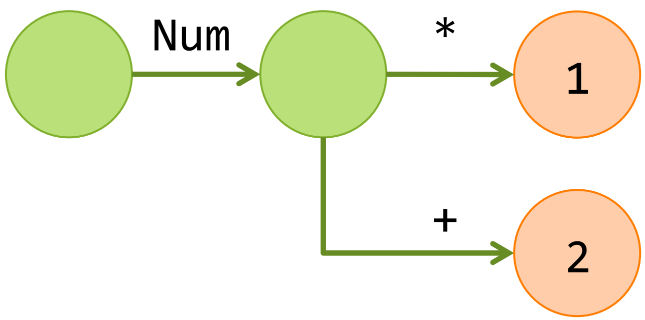


入力

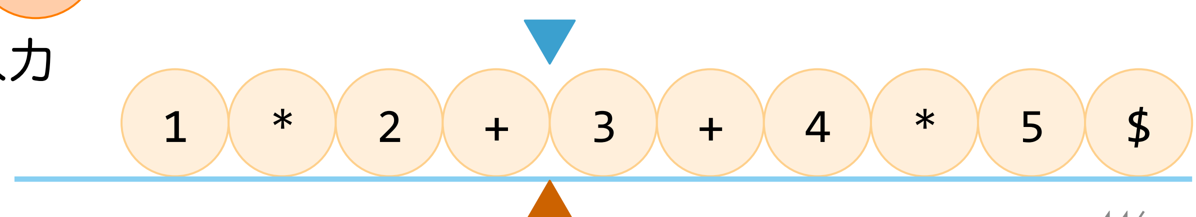


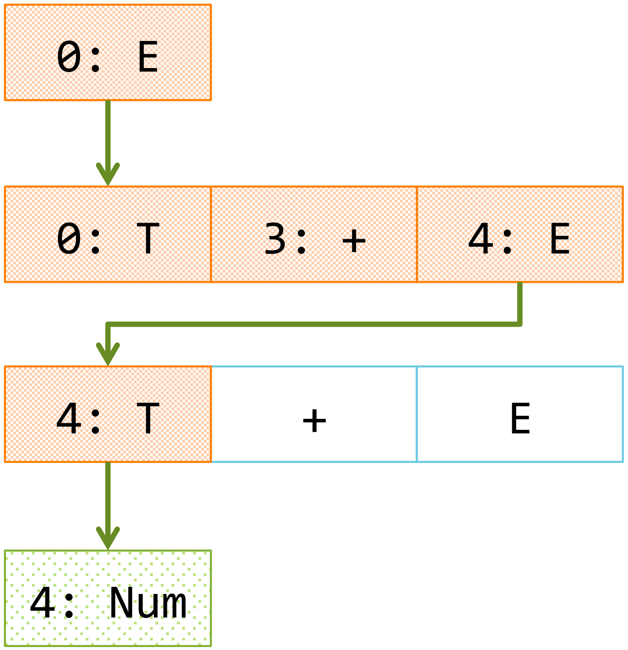


T の先読みオートマトン

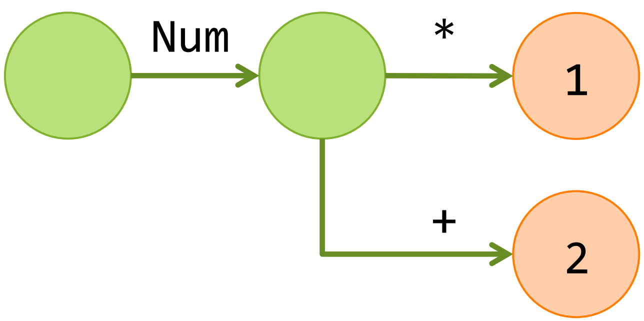


入力

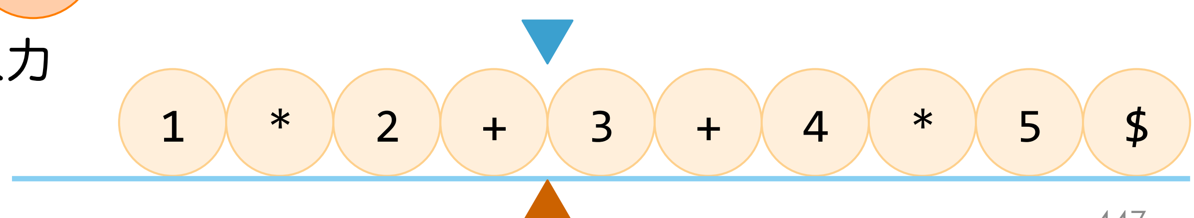


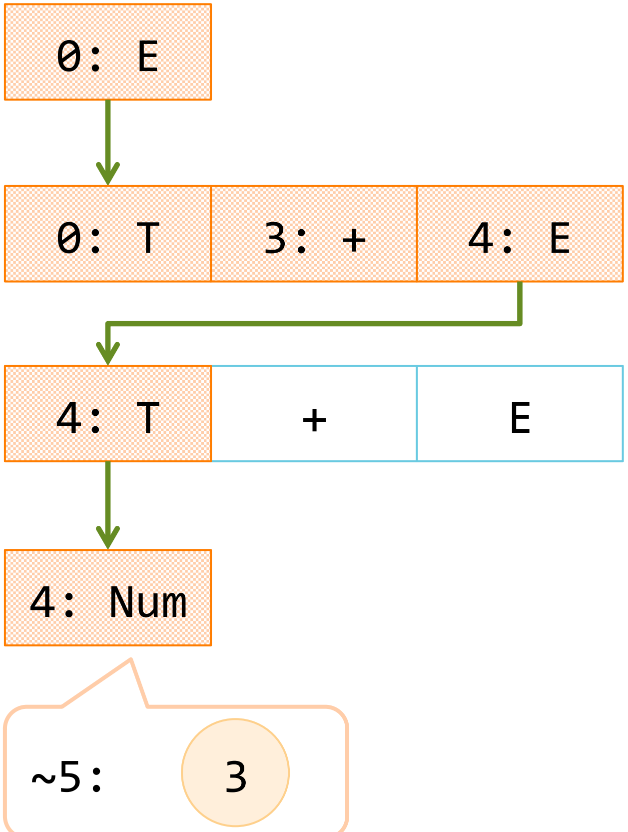


T の先読みオートマトン

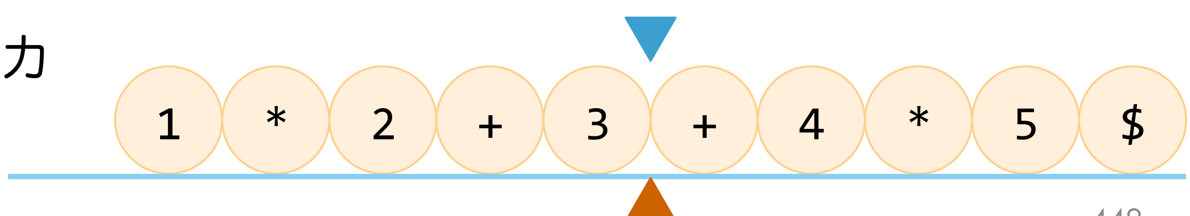


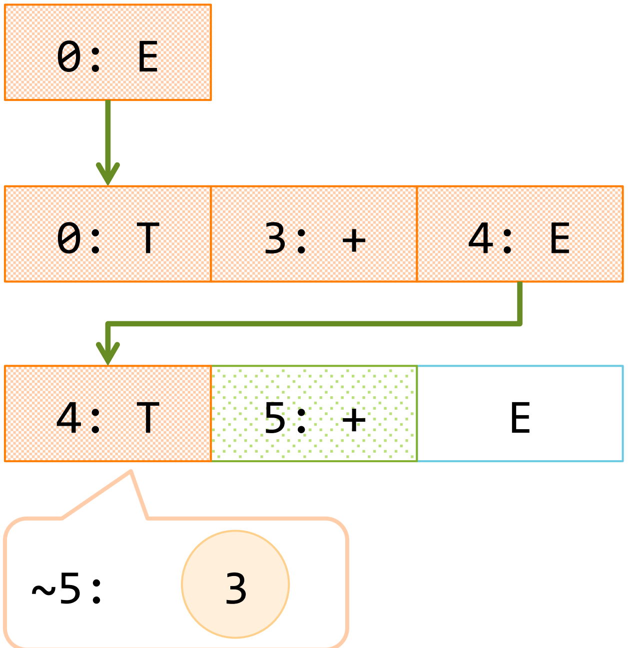
入力

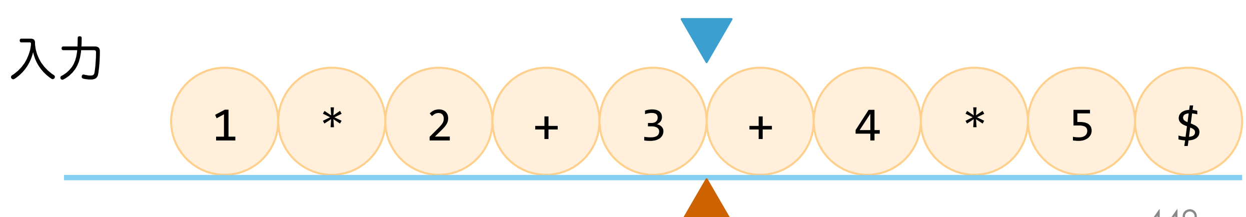


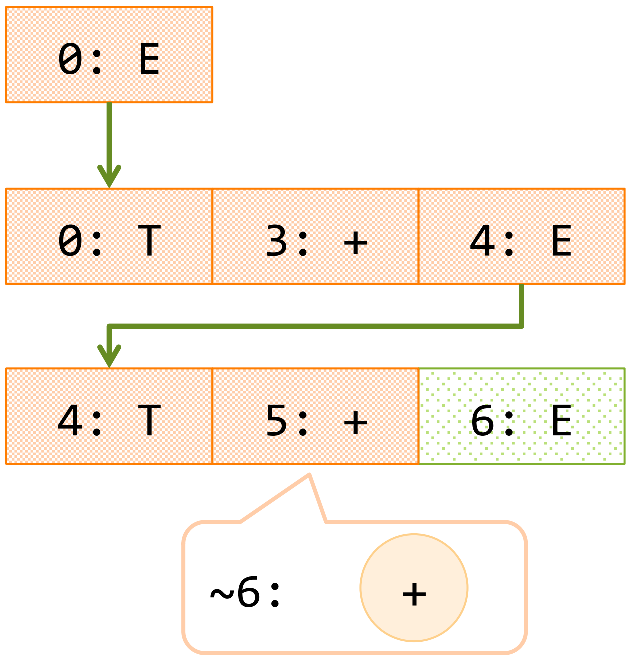


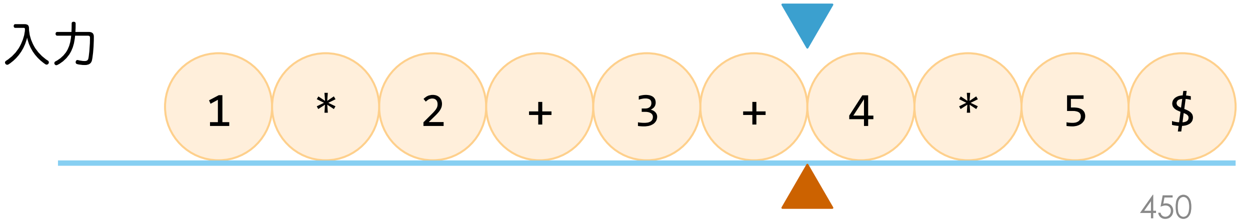
入力

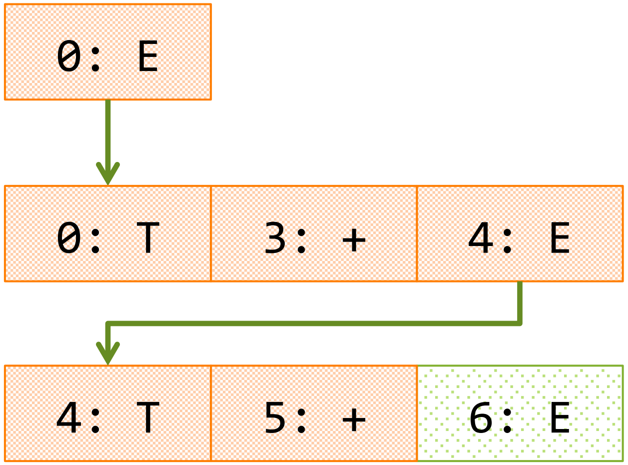



0: E
0: T
3: +
4: E
4: T
5: +
E
~5:
3

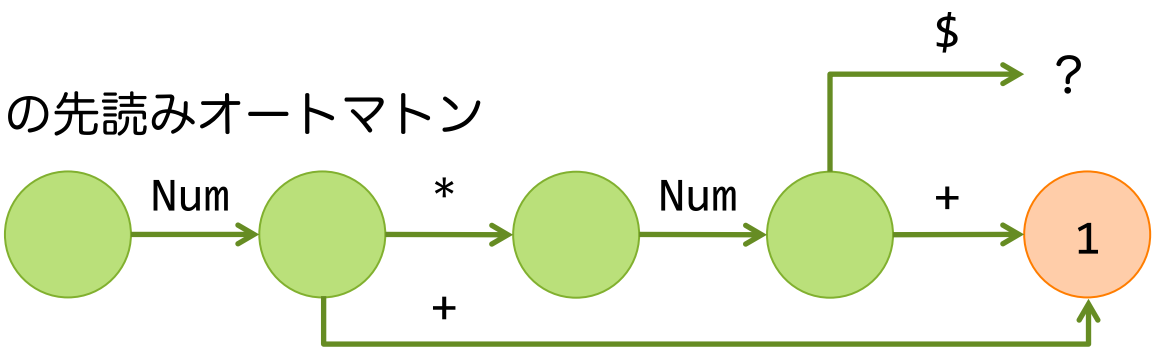

入力
1
*
2
+
3
+
4
*
5
$


0: E
0: T
3: +
4: E
4: T
5: +
6: E
~6:
+

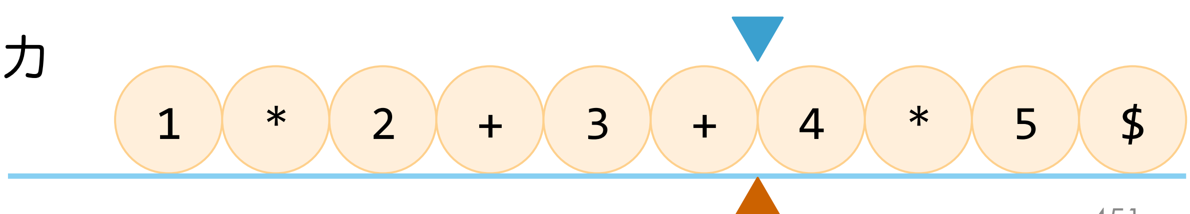

入力
1
*
2
+
3
+
4
*
5
$

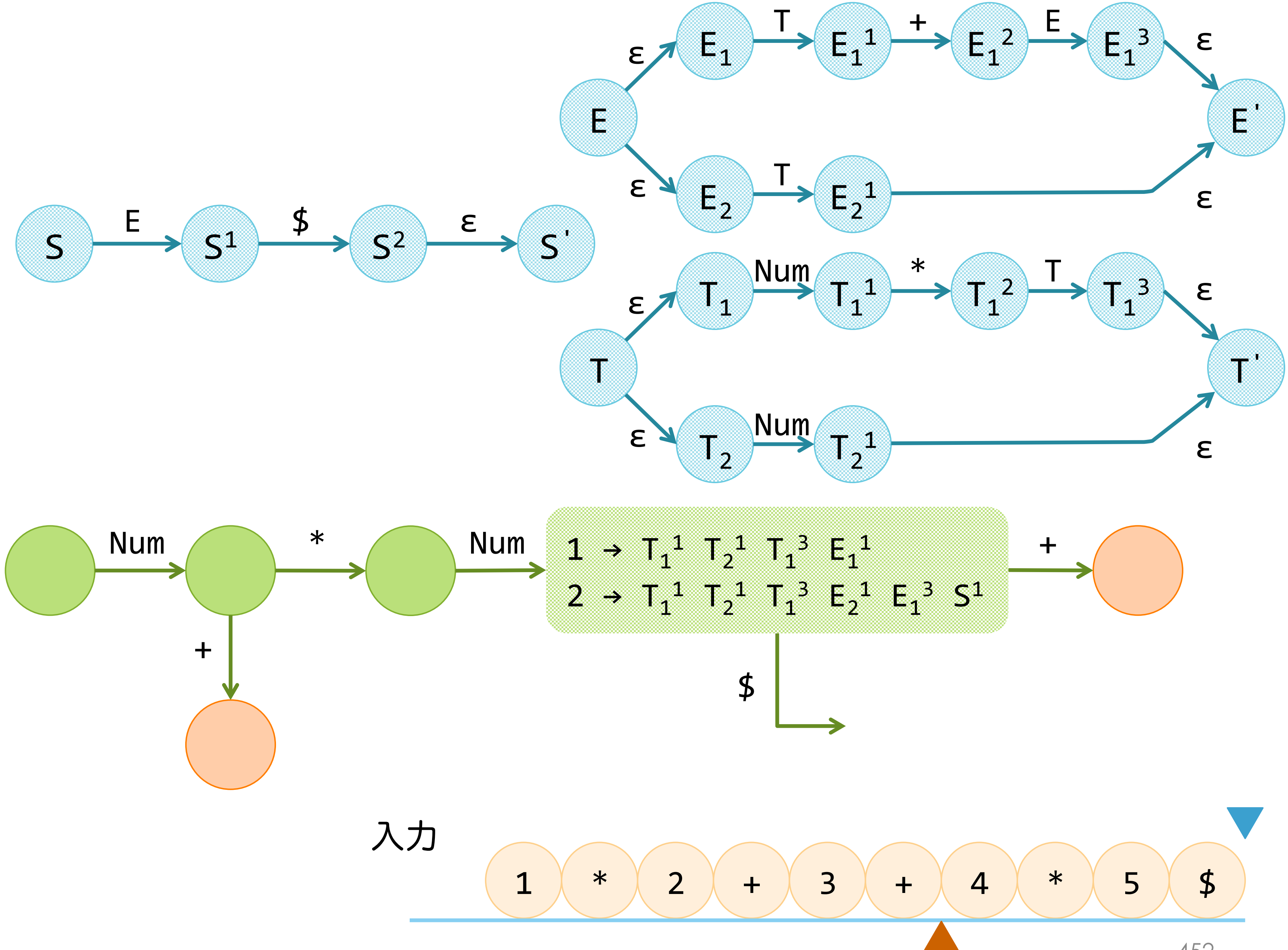


E の先読みオートマトン

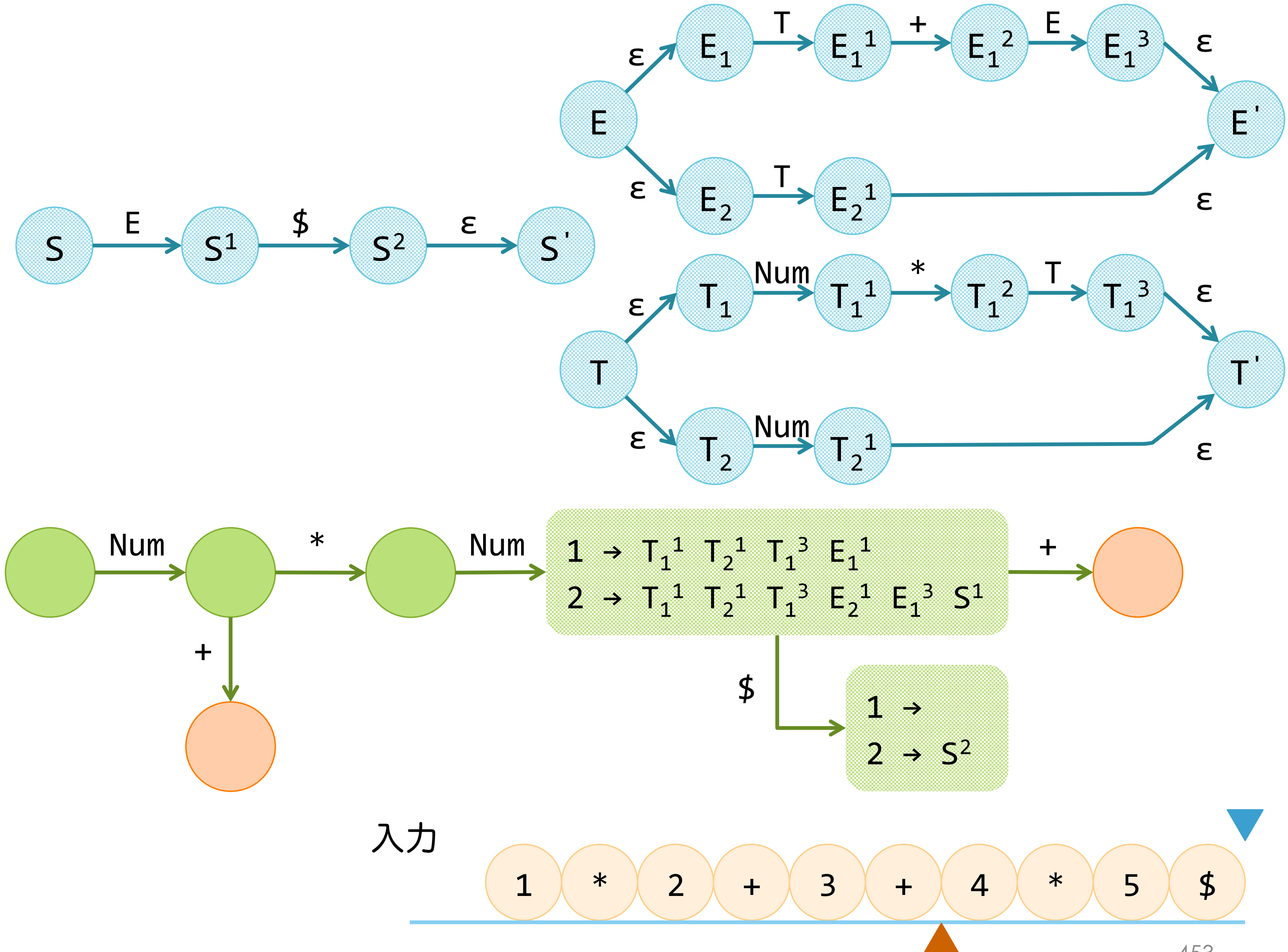


入力

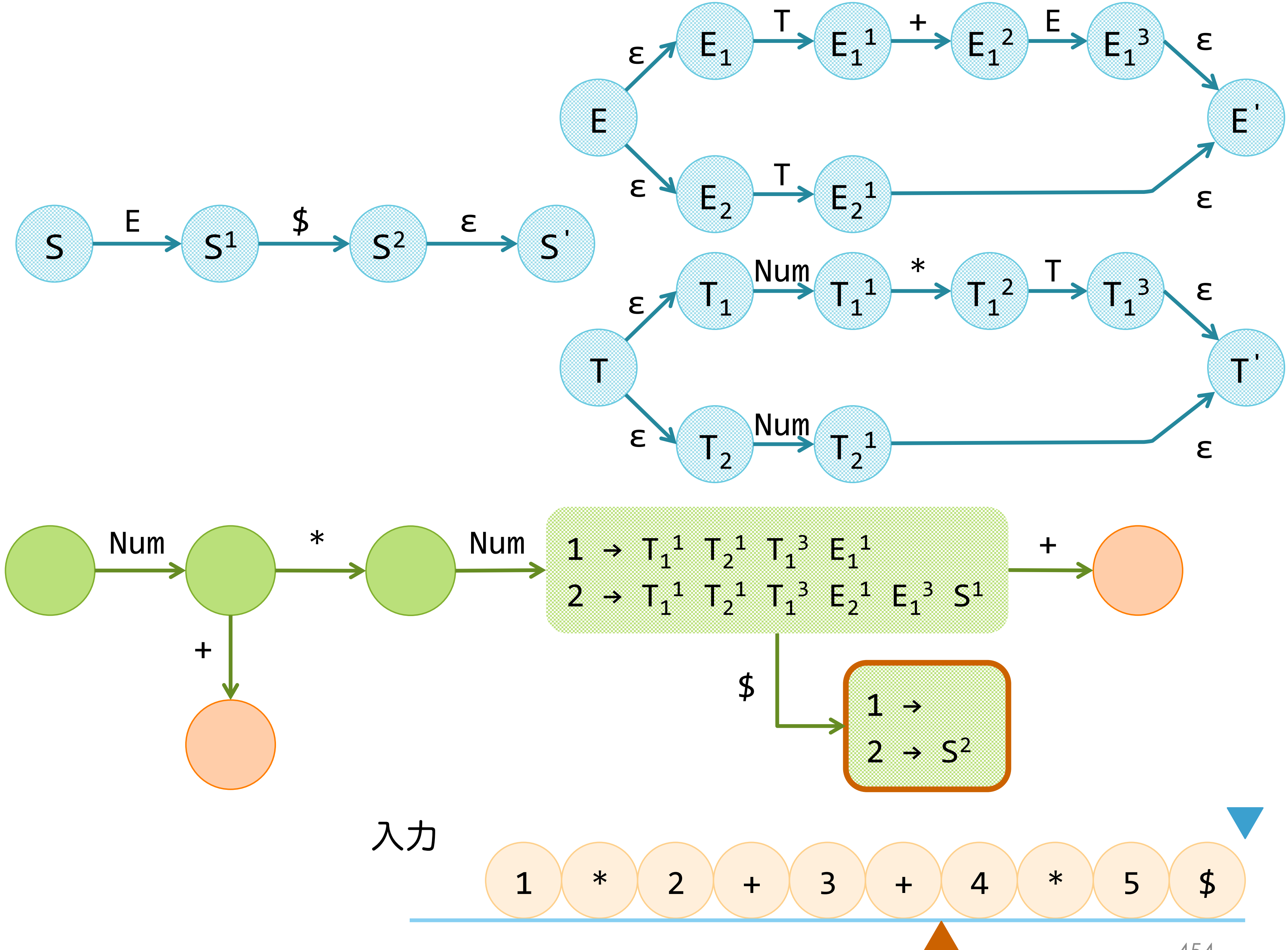




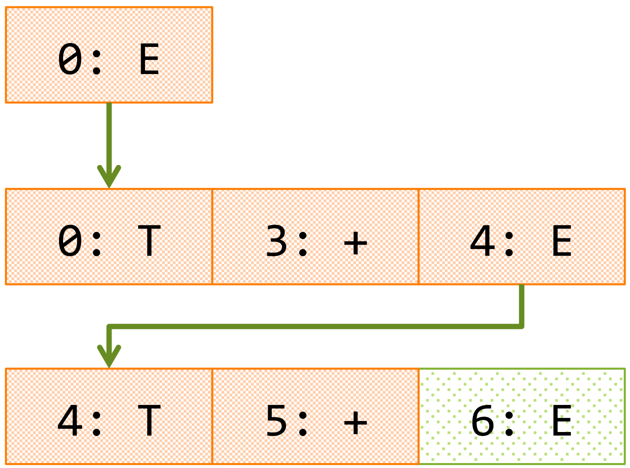
S
E
S1
$
S2
ε
S'
E
E1
T
E1 1
+
E1 2
E1 3
E'
E2
E2 1
T
T1
Num
T1 1
*
T1 2
T1 3
T'
T2
T2 1
1 → T1 1 T2 1 T1 3 E1 1
2 → T1 1 T2 1 T1 3 E2 1 E1 3 S1
入力
1 * 2 + 3 + 4 * 5 $



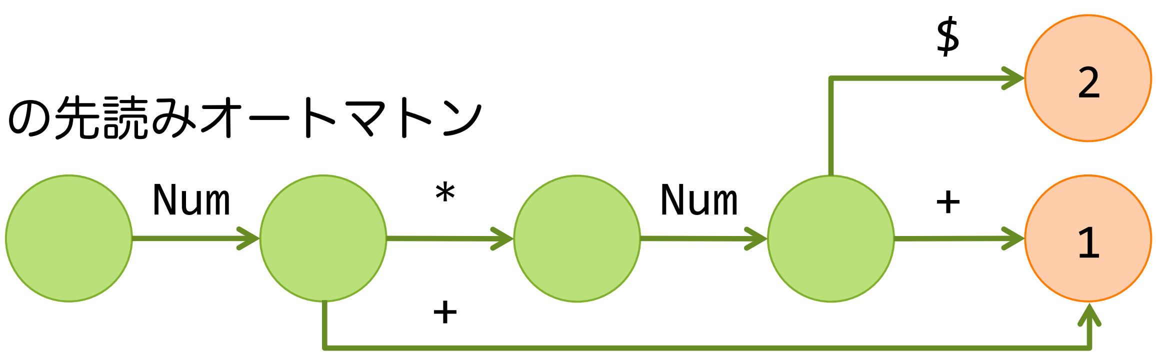
S
E
S1
$
S2
ε
S'
E
ε
E1
T
E11
+
E12
E
E13
ε
E'
ε
E2
T
E21
ε
T
ε
T1
Num
T11
*
T12
T
T13
ε
T'
ε
T2
Num
T21
ε
Num
*
Num
1 → T11 T21 T13 E11
2 → T11 T21 T13 E21 E13 S1
+
+
$
1 →
2 → S2
入力
1
*
2
+
3
+
4
*
5
$



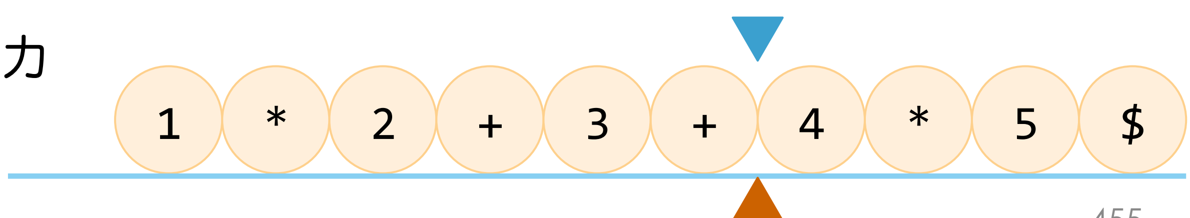
S
E
S1
$
S2
ε
S'
E
ε
E1
T
E1 1
+
E1 2
E
E1 3
ε
E'
ε
E2
T
E2 1
ε
T
ε
T1
Num
T1 1
*
T1 2
T
T1 3
ε
T'
ε
T2
Num
T2 1
ε
Num
*
Num
+
1 → T1 1 T2 1 T1 3 E1 1
2 → T1 1 T2 1 T1 3 E2 1 E1 3 S1
+
$
1 →
2 → S2
入力
1
*
2
+
3
+
4
*
5
$

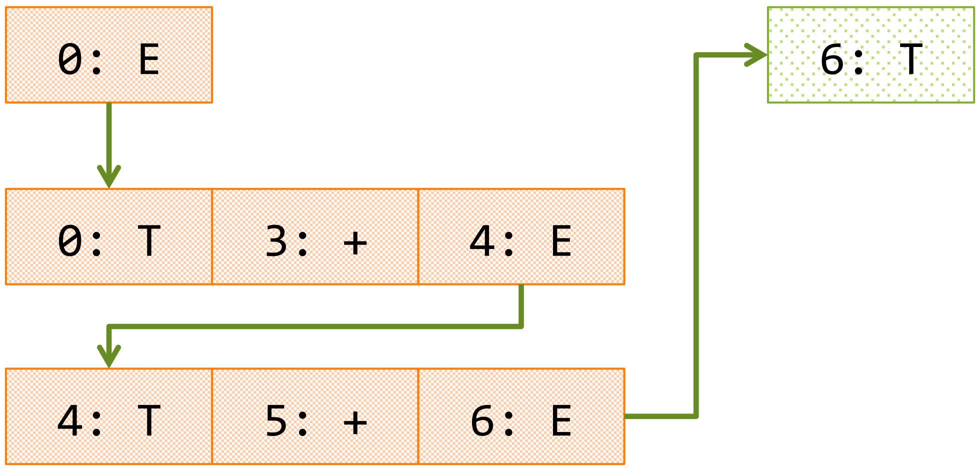


E の先読みオートマトン

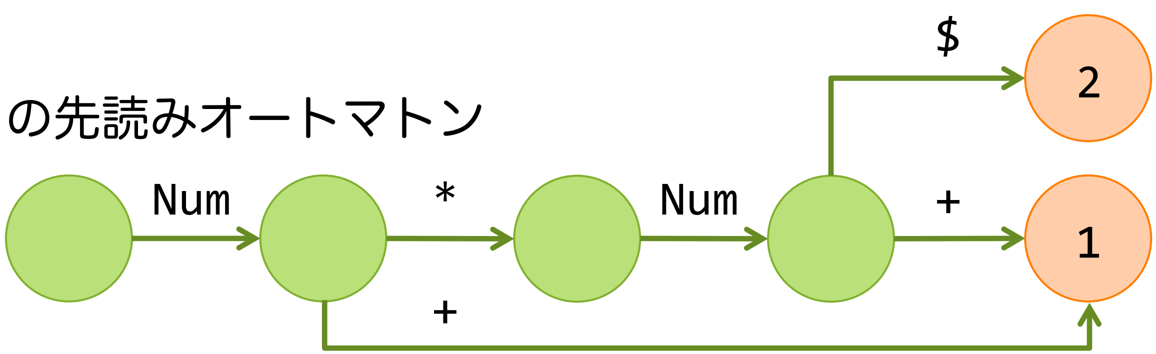


入力

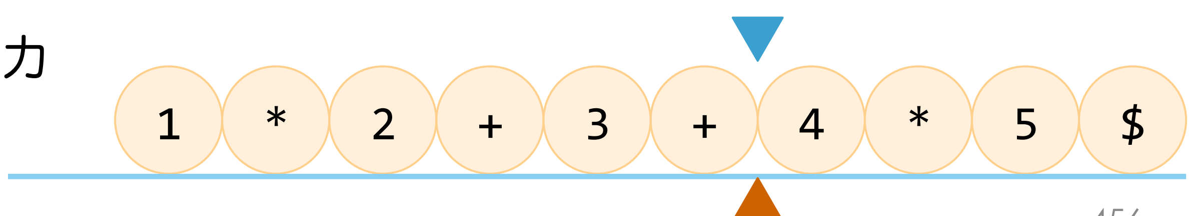


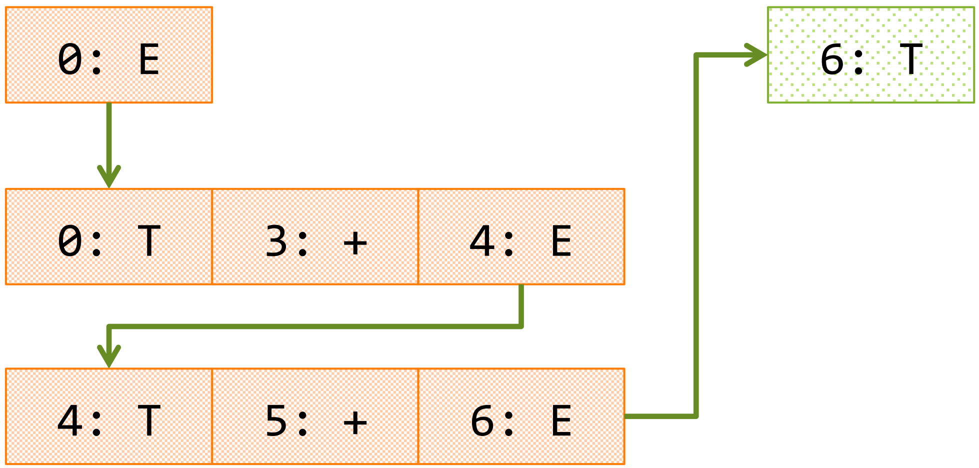


E の先読みオートマトン

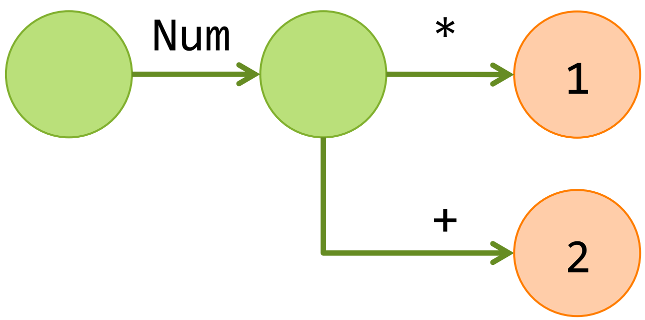


入力

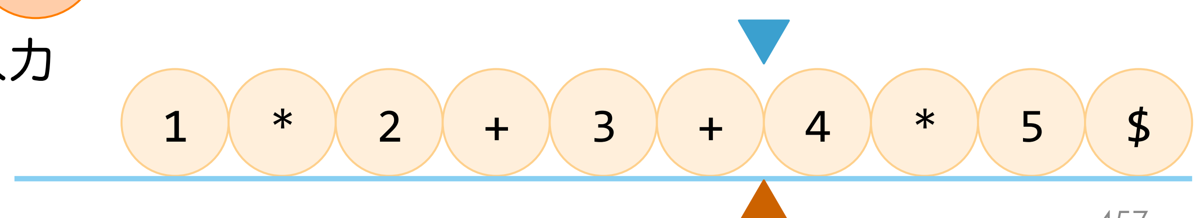


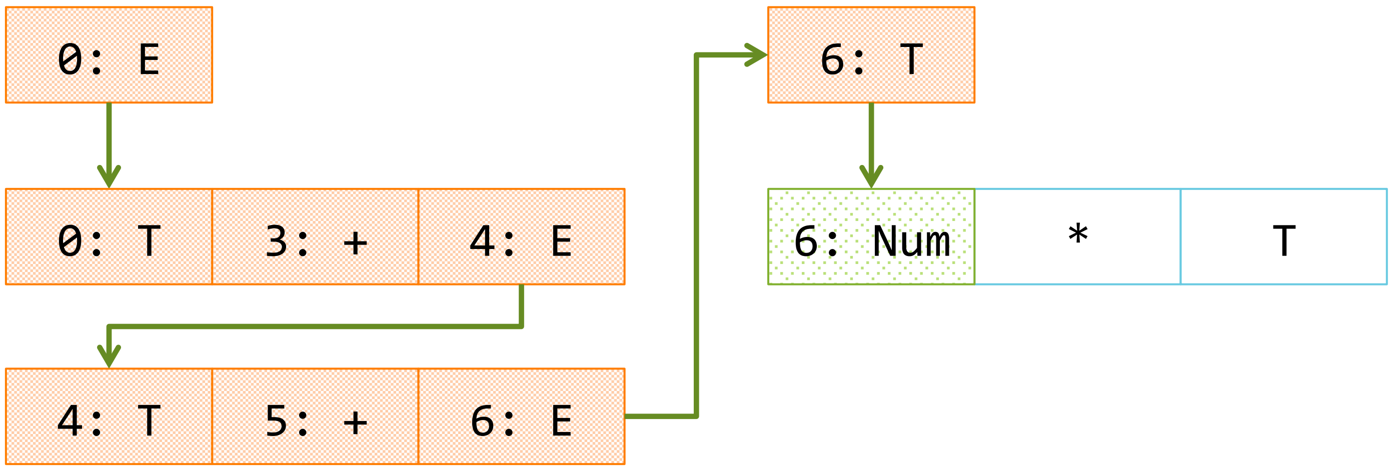


T の先読みオートマトン

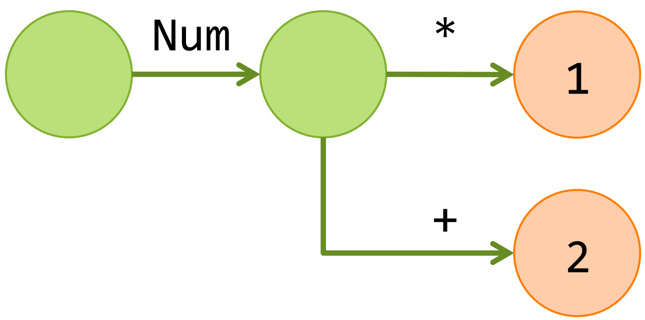


入力

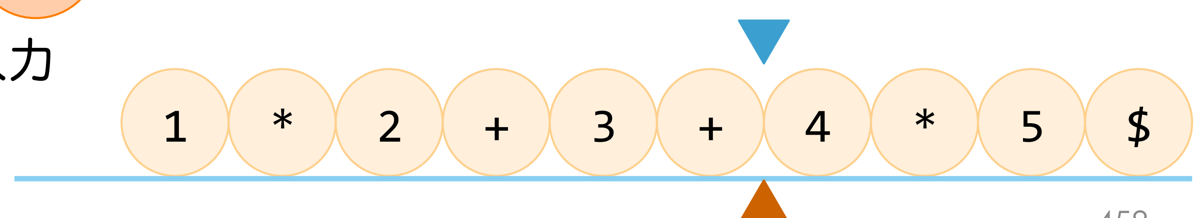


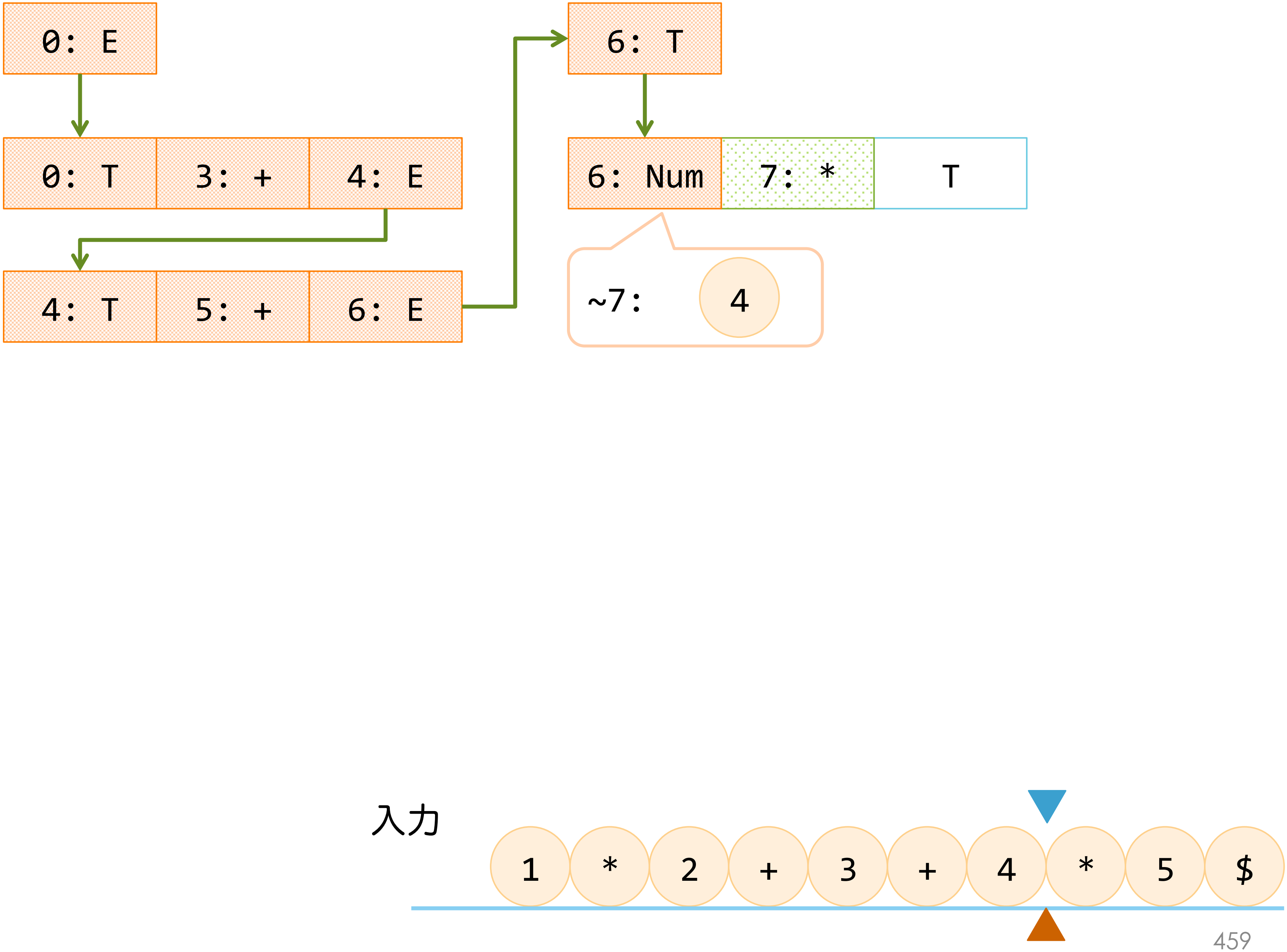


T の先読みオートマトン

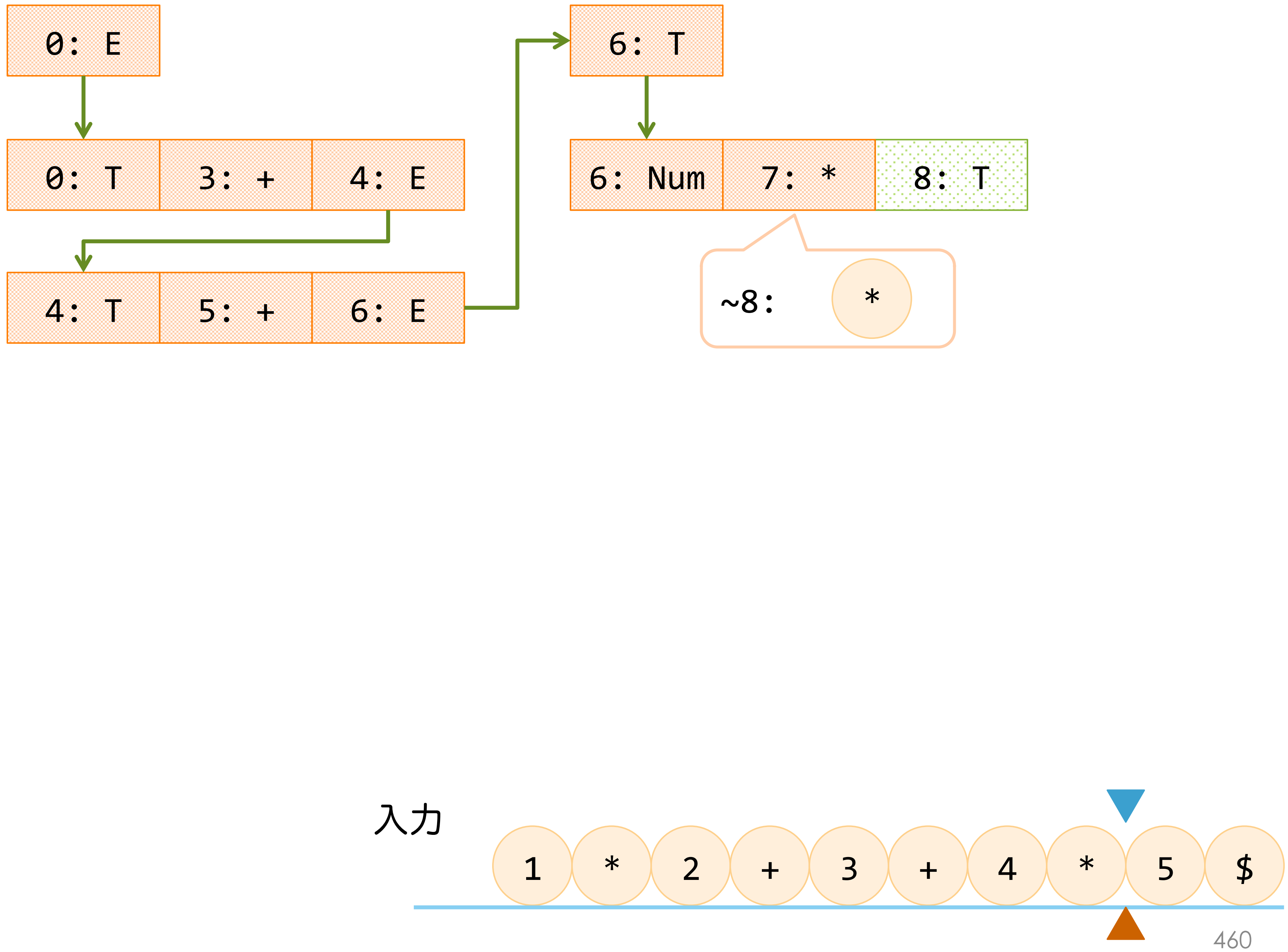


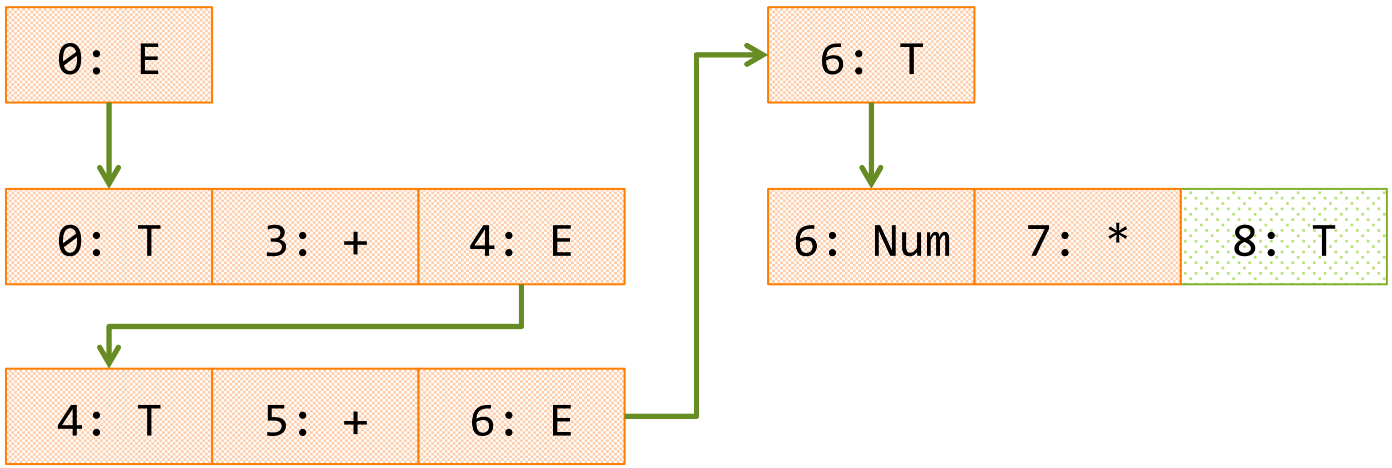
入力

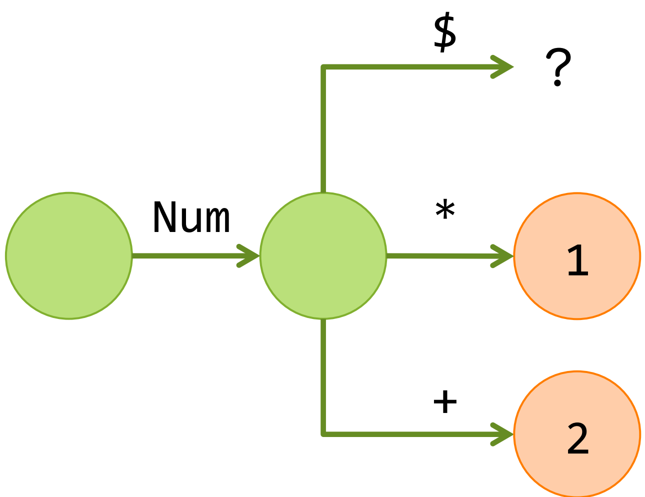




0: E
0: T
3: +
4: E
4: T
5: +
6: E
6: T
6: Num
7: *
T
~7:
4
入力
1
*
2
+
3
+
4
*
5
$

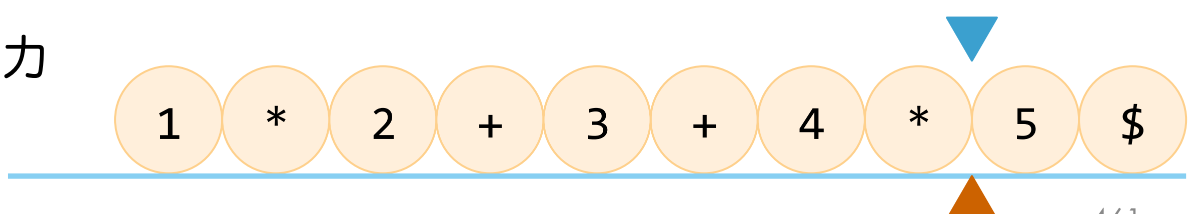

0: E
0: T
3: +
4: E
4: T
5: +
6: E
6: T
6: Num
7: *
8: T
~8:
*
入力
1
*
2
+
3
+
4
*
5
$

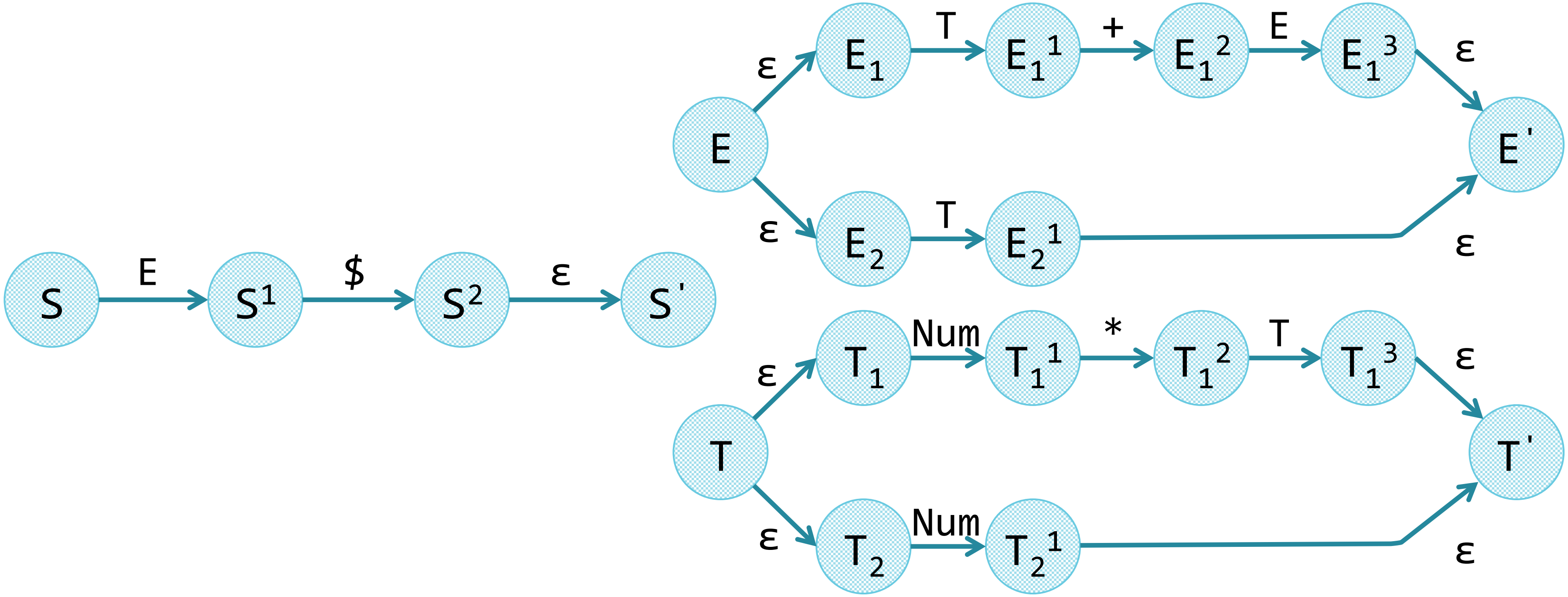


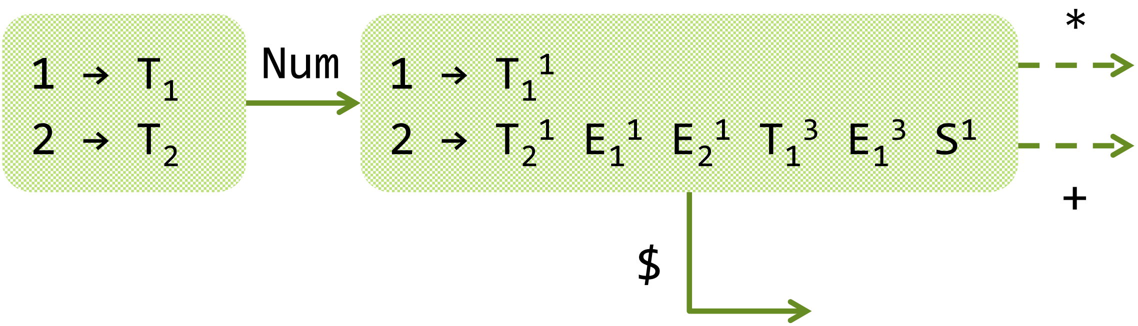


T の先読みオートマトン

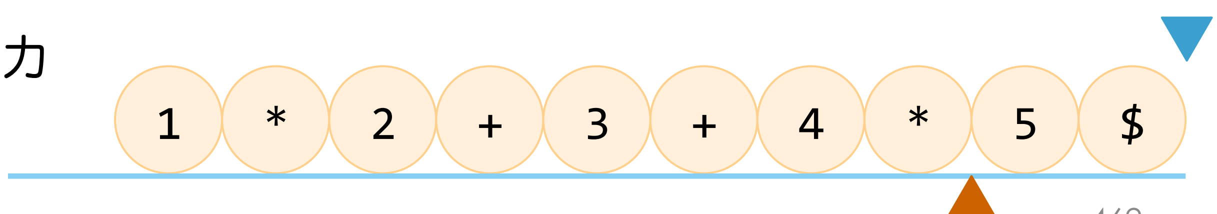
入力

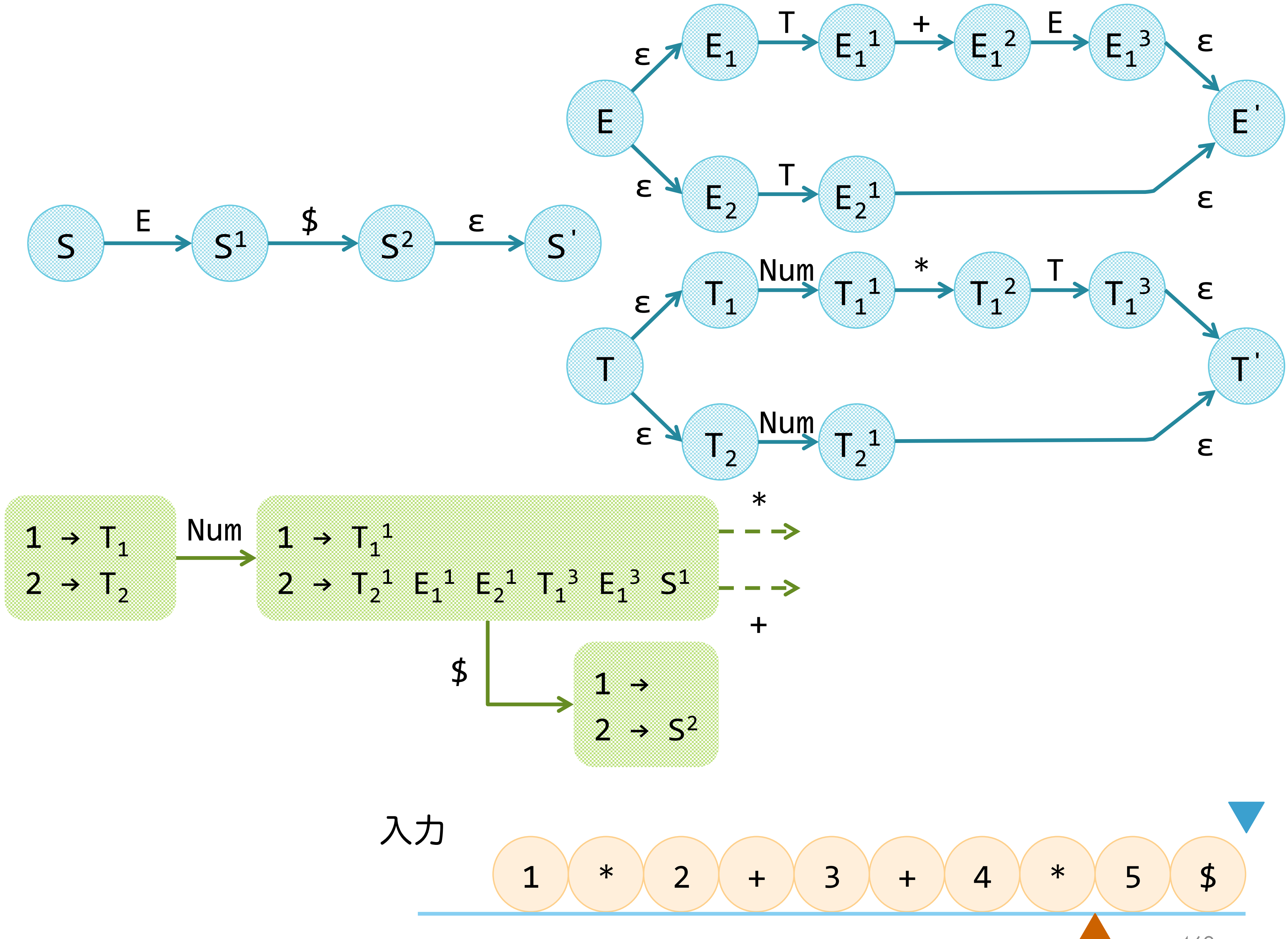



E
S¹
S²
S'
$
ε
E₁
E₁¹
E₁²
E₁³
E'
E₂
E₂¹
T
+
T₁
T₁¹
T₁²
T₁³
T'
T₂
T₂¹
Num
*

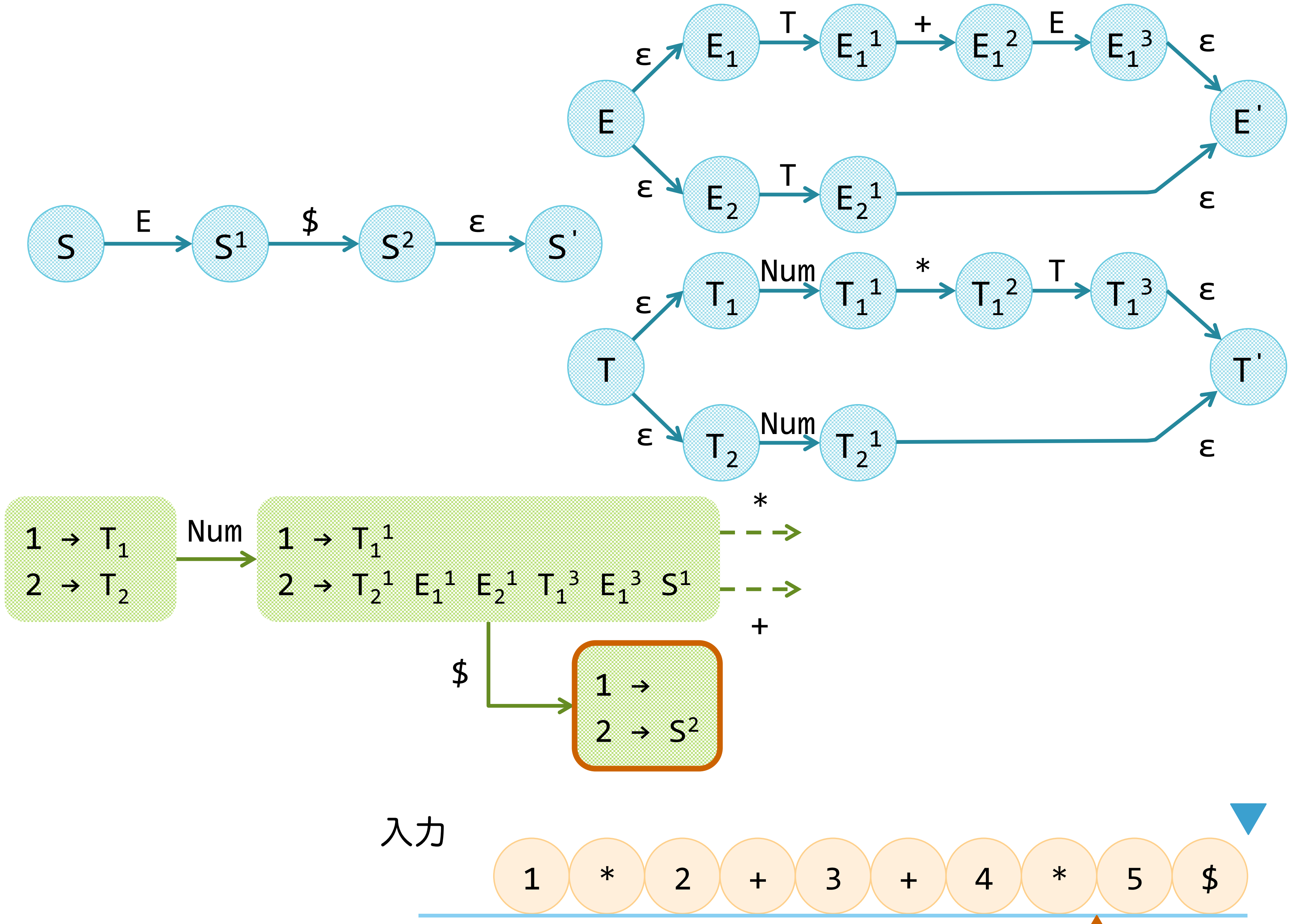

1 → T₁
2 → T₂
Num
1 → T₁¹
2 → T₂¹ E₁¹ E₂¹ T₁³ E₁³ S¹
*
+
$

入力

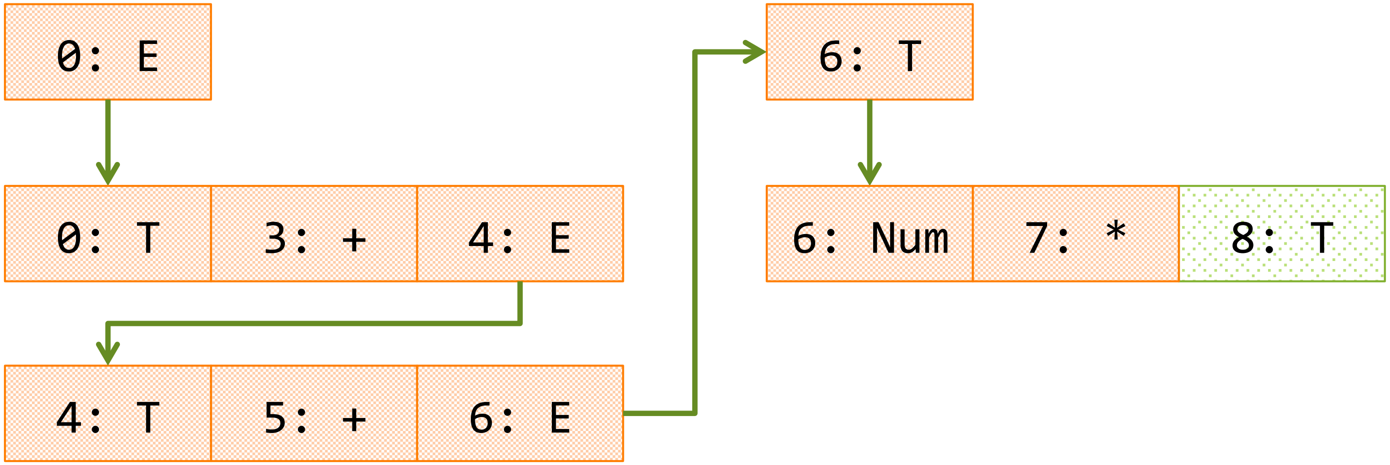

1
*
2
+
3
+
4
*
5
$



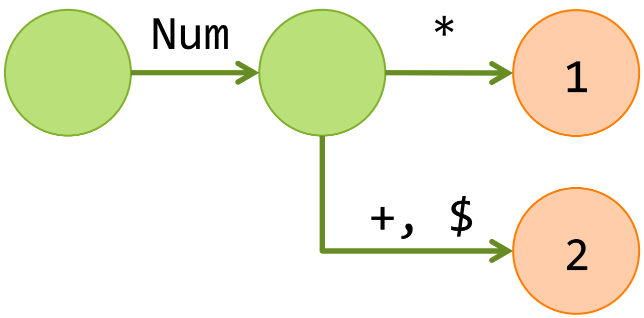
S
E
S1
$
S2
ε
S'
E
ε
E1
T
E11
+
E12
E
E13
ε
E'
ε
E2
T
E21
ε
T
ε
T1
Num
T11
*
T12
T
T13
ε
T'
ε
T2
Num
T21
ε
1 → T1
2 → T2
Num
1 → T11
2 → T21 E11 E21 T13 E13 S1
*
+
$
1 →
2 → S2
入力
1 * 2 + 3 + 4 * 5 $



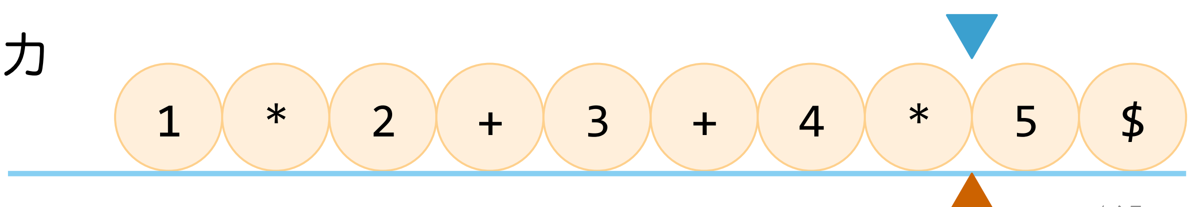
E
T
+
ε
Num
*
$
S
E'
T'
1 → T1
2 → T2
1 → T1 1
2 → T2 1 E1 1 E2 1 T1 3 E1 3 S1
1 →
2 → S2
入力
1 * 2 + 3 + 4 * 5 $

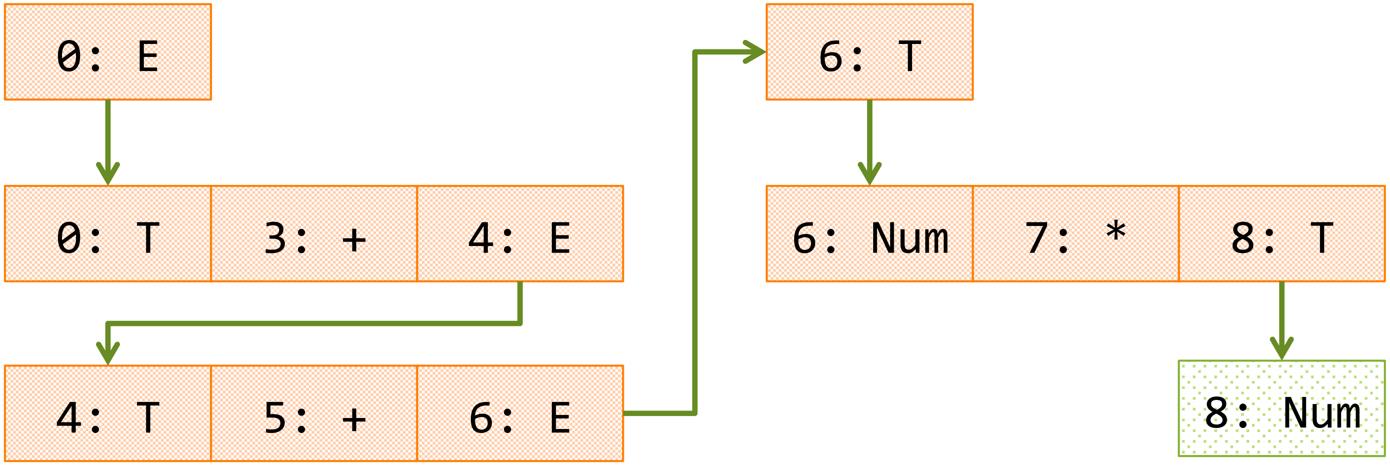


T の先読みオートマトン

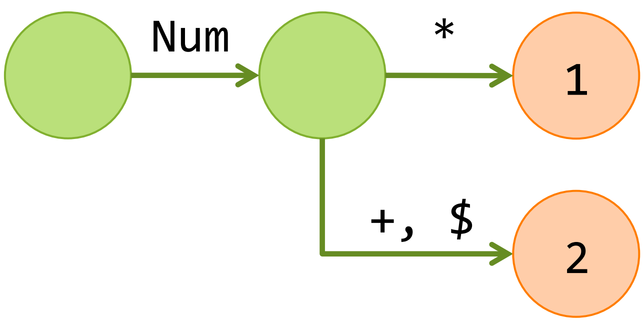


入力

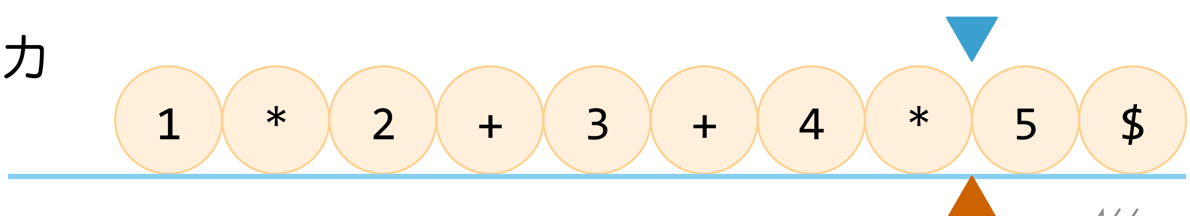


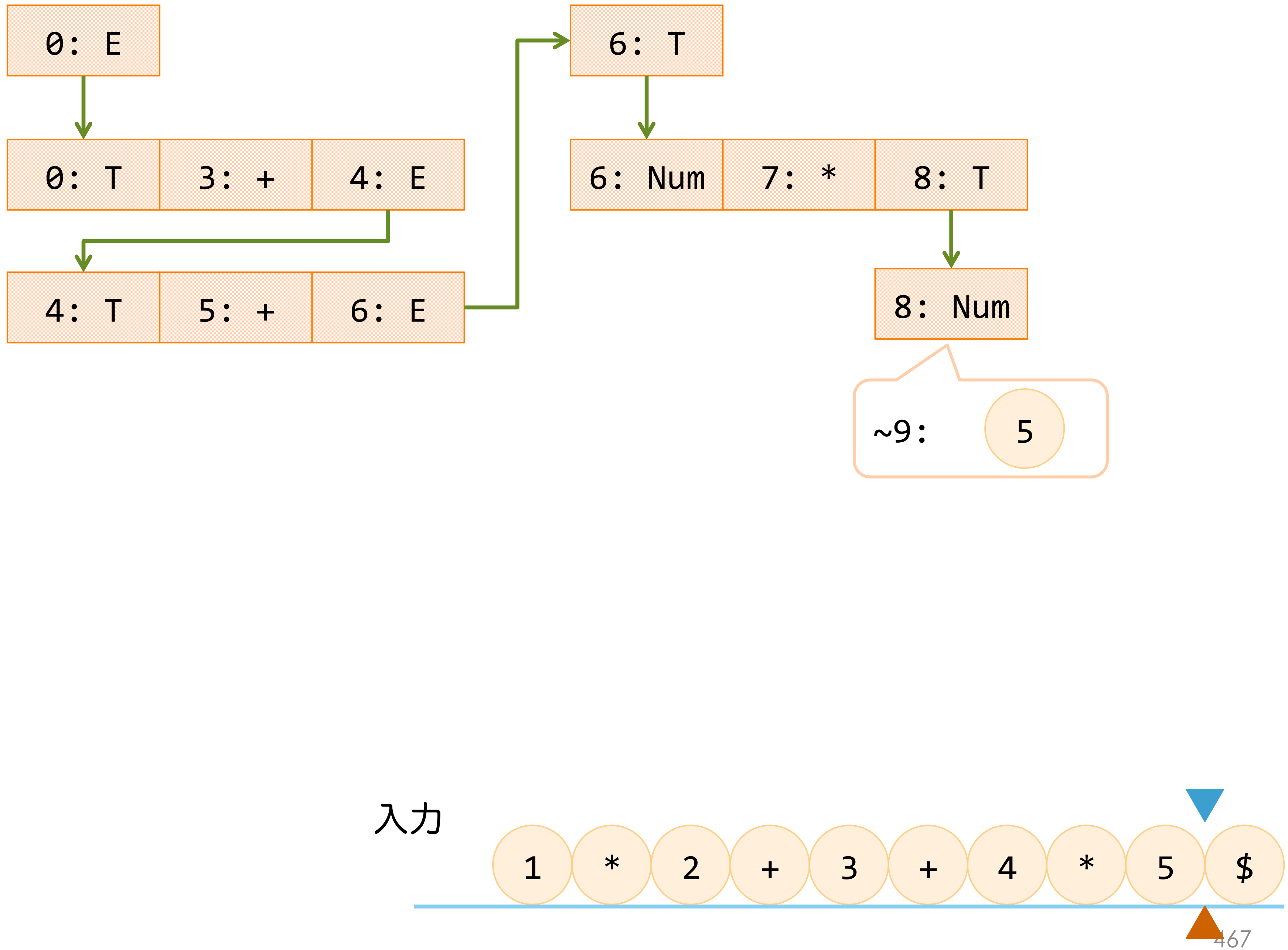


T の先読みオートマトン

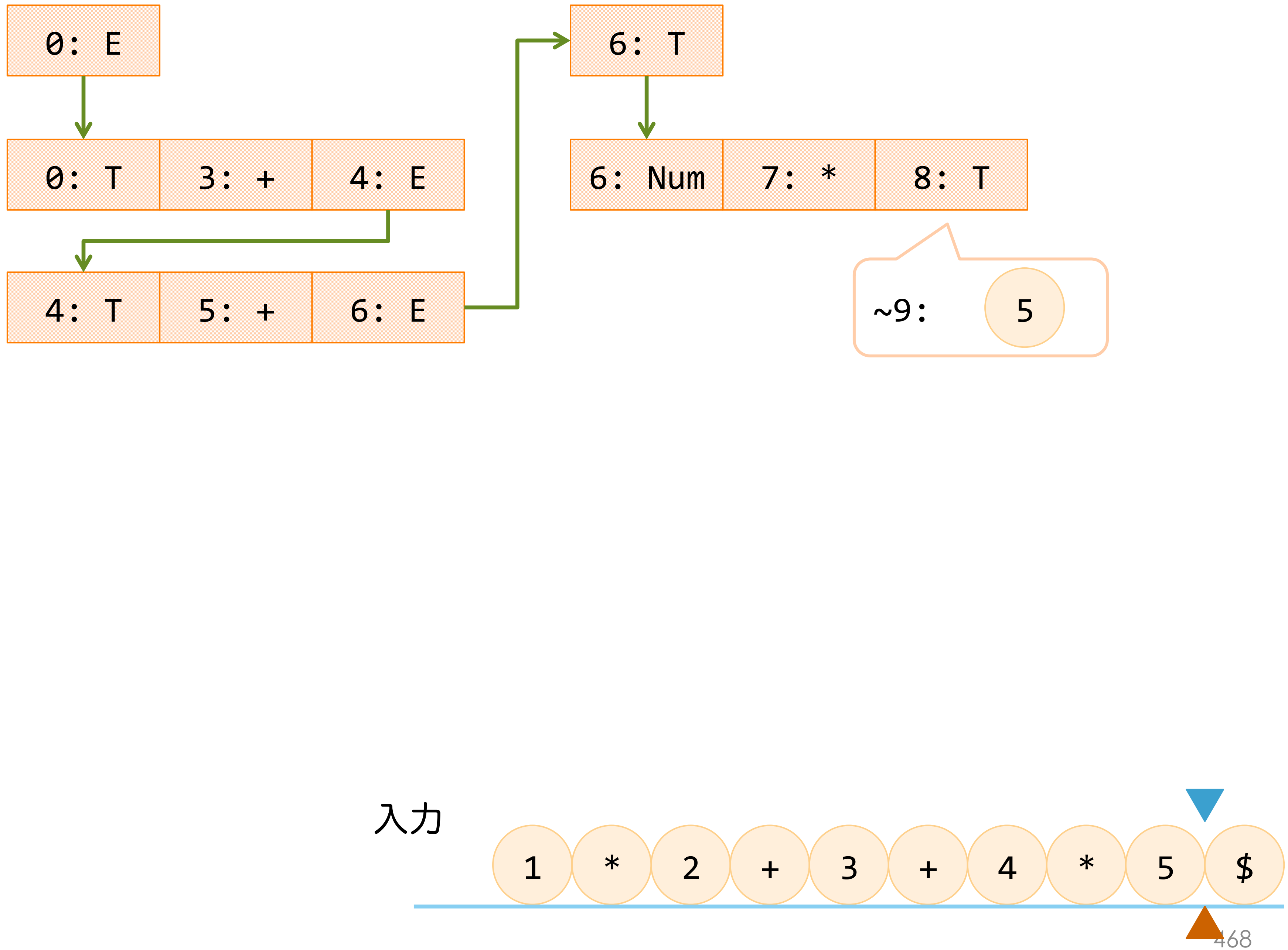


入力

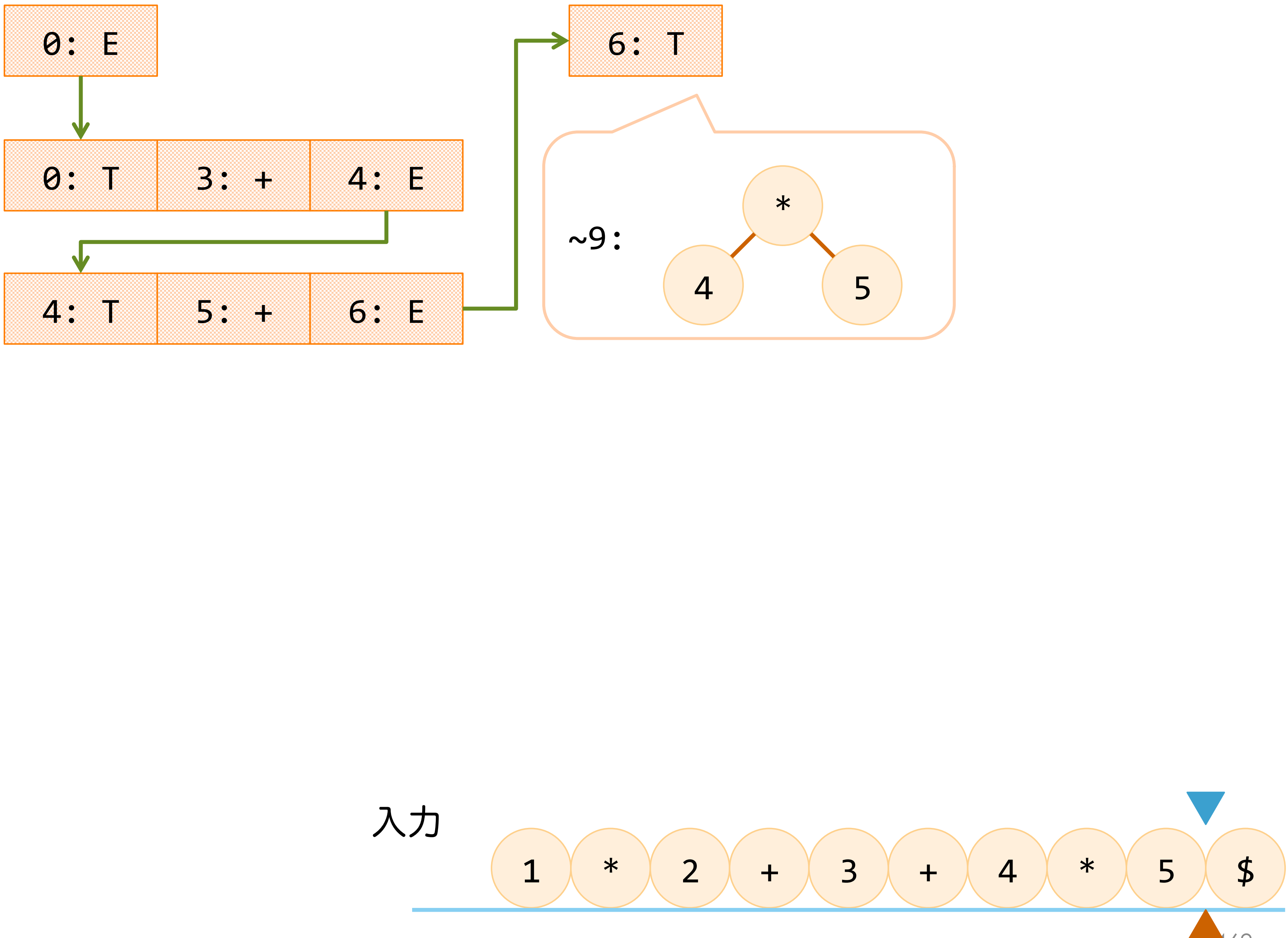


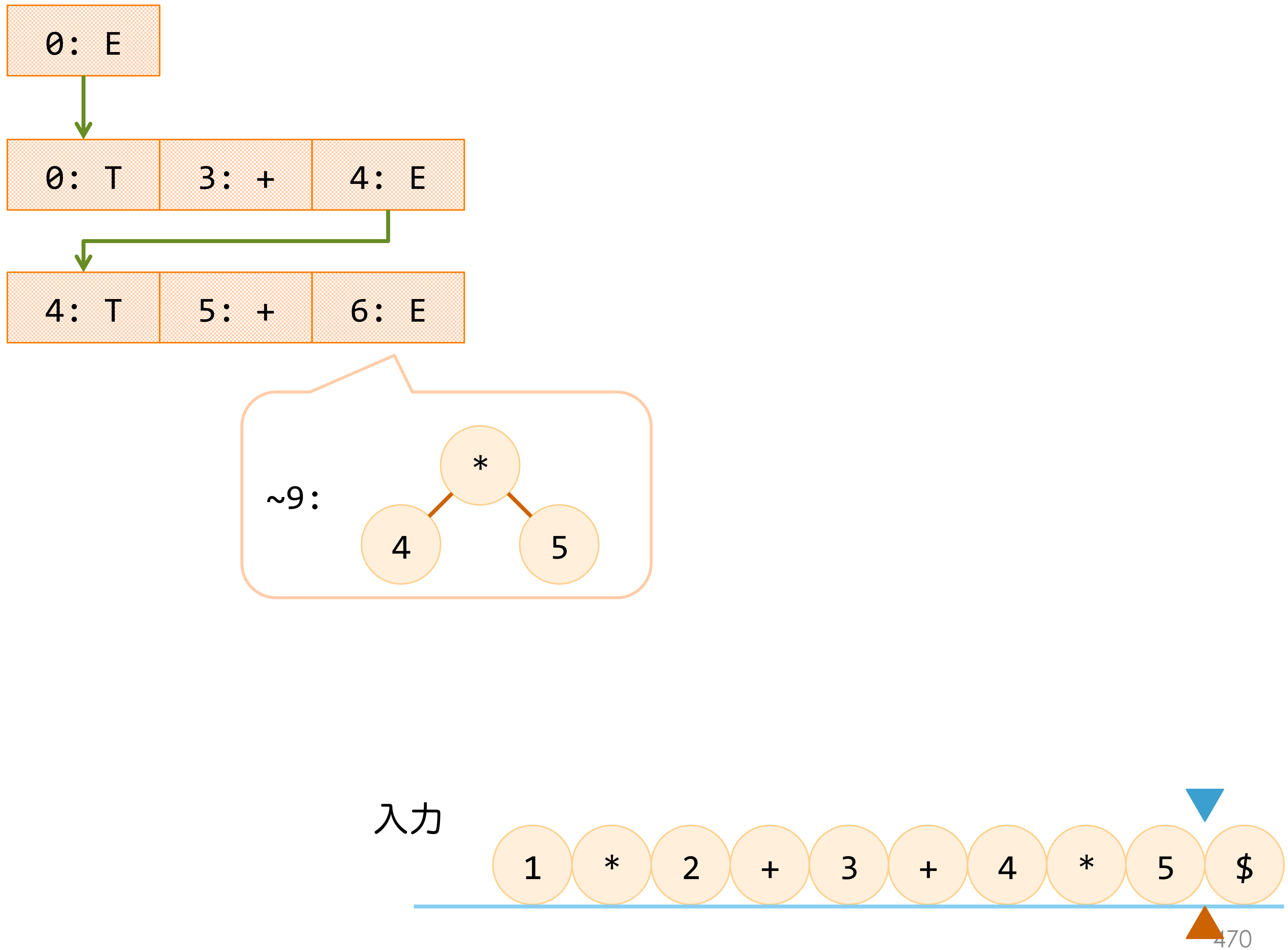


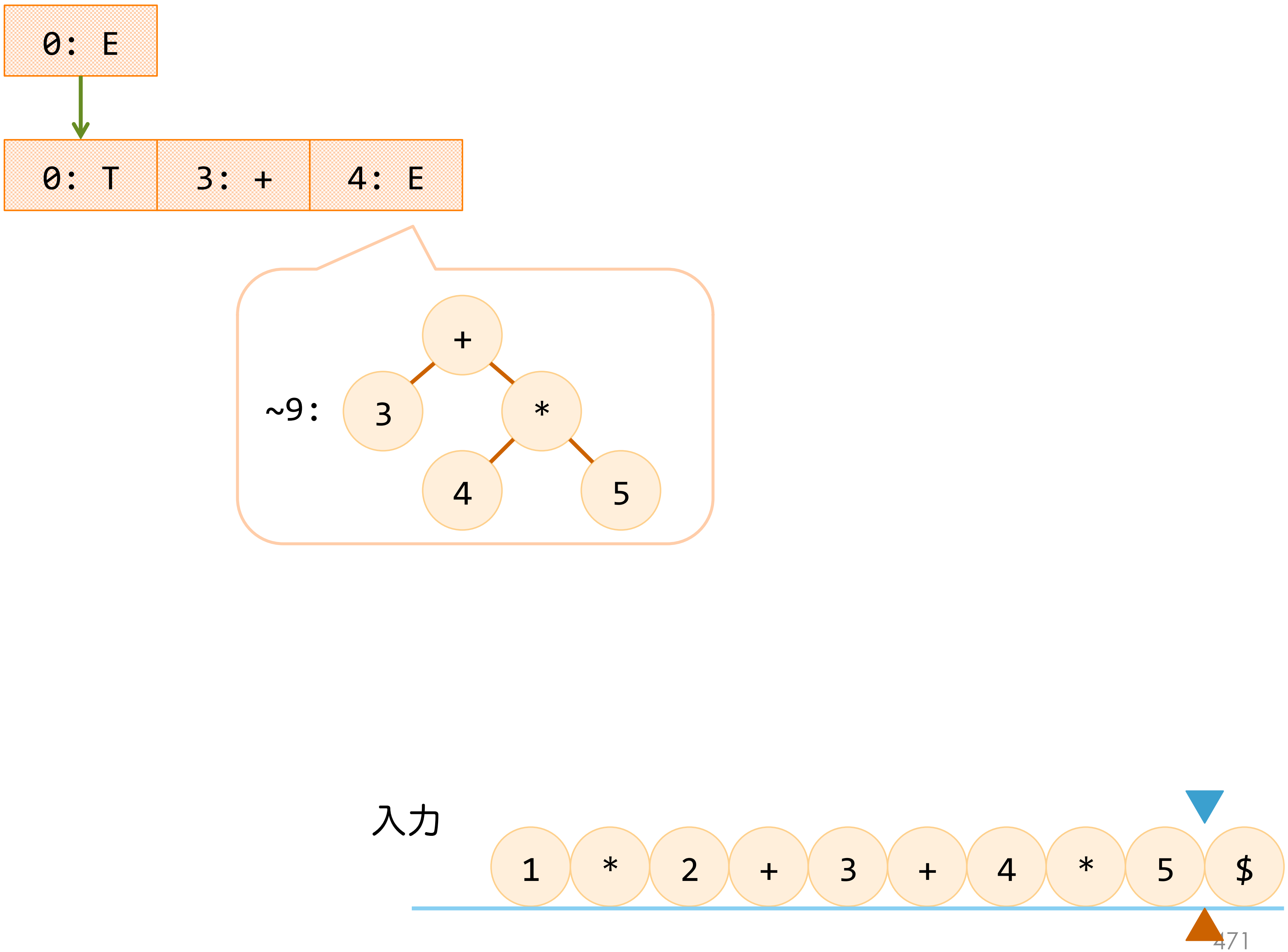
0: E
0: T
3: +
4: E
4: T
5: +
6: E
6: T
6: Num
7: *
8: T
8: Num
~9:
5
入力
1
*
2
+
3
+
4
*
5
$



0: E
0: T
3: +
4: E
4: T
5: +
6: E
6: T
6: Num
7: *
8: T
~9:
5
入力
1
*
2
+
3
+
4
*
5
$

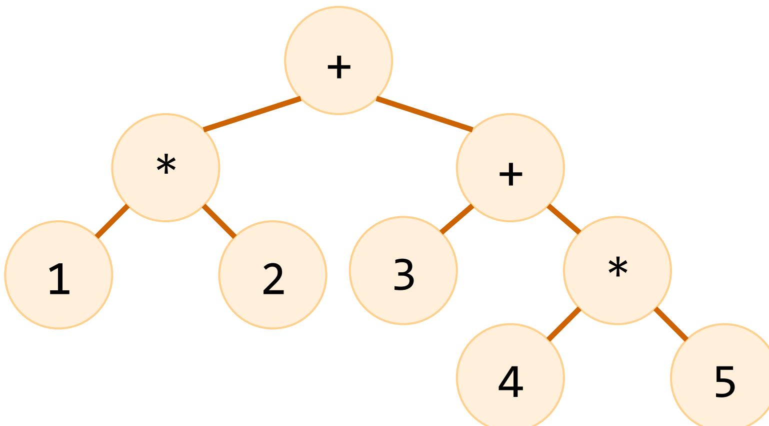

0: E
0: T
3: +
4: E
4: T
5: +
6: E
6: T
~9:
*
4
5
入力
1
*
2
+
3
+
4
*
5
$


0: E
0: T
3: +
4: E
4: T
5: +
6: E
~9:
*
4
5
入力
1
*
2
+
3
+
4
*
5
$

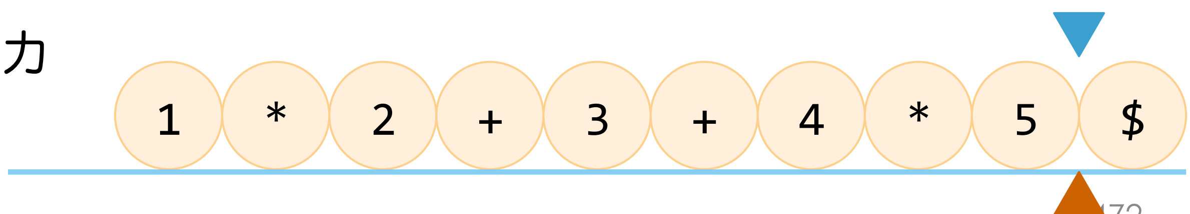

0: E
0: T
3: +
4: E
~9:
+
3
*
4
5
入力
1
*
2
+
3
+
4
*
5
$

0: E

~9:



入力



Adaptive LL(*) parsing [Parr 2014]

- ANTLR v4 のアルゴリズム
  - v2 は LL(k), v3 は LL(*)
- 大抵の場合に O(n) で動作
  - GLR や GLL よりも高速
- 能力
  - 計算量 : $O(n^4)$
  - 言語 : ALL(*) language
  - 文法 : Predicated grammar

年表

1961: Shunting-yard

1973: Pratt

1979: Precedence climbing

1965: LR

1965-1969: CYK

1968: Earley

1969: LALR(k)

1971: SLR(k)

1975: Valiant

1985: GLR

1969: SLL(k)

1979: LL(k)

1991: メモ化再帰 Earley

2002: Packrat

2008: Packrat+左再帰

2010: GLL

2011: LL(*)

2014: Adaptive LL(*)