

アンダースタンディング・コンピューテーション 第4章  
プッシュダウン・オートマトン

岡本 吉央

okamotoy@uec.ac.jp

電気通信大学

2019年7月19日

最終更新：2019年7月19日 15:25

- |   |                |      |
|---|----------------|------|
| 2 | プログラム意味論       | (5月) |
| 3 | 有限オートマトン       | (6月) |
| 4 | プッシュダウン・オートマトン | (7月) |
| 5 | チューリング機械       |      |
| 6 | ラムダ計算          |      |
| 7 | 万能性            |      |
| 8 | 決定可能性          |      |
| 9 | 抽象解釈／静的意味論     |      |

### 今からやること

「有限オートマトン」に簡単な「記憶 (メモリ)」の機能を追加して、有限オートマトンの能力を高める

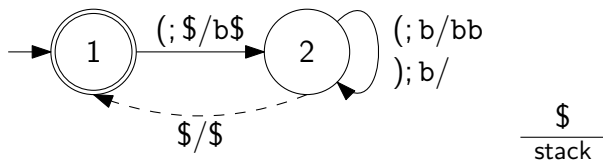
- ▶ ⇨ プッシュダウン・オートマトン

- ① 決定性プッシュダウン・オートマトン
- ② 非決定性プッシュダウン・オートマトン
- ③ プッシュダウン・オートマトンによるパース
- ④ 個人プロジェクト案の例

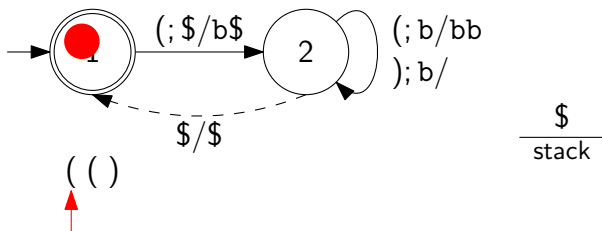
## 標語的にいえば

プッシュダウン・オートマトン = 有限オートマトン + スタック

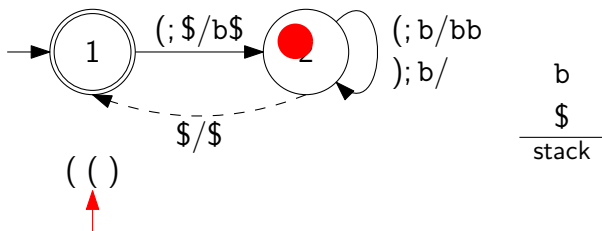
スタック = 後入れ先出し (last-in first-out) のデータ構造



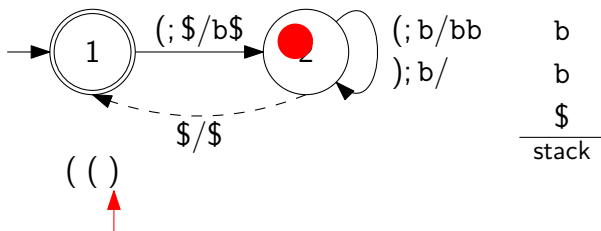
棄却する例



棄却する例

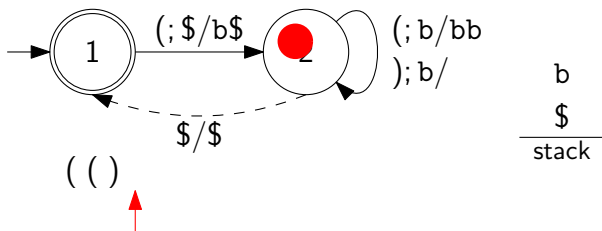


棄却する例

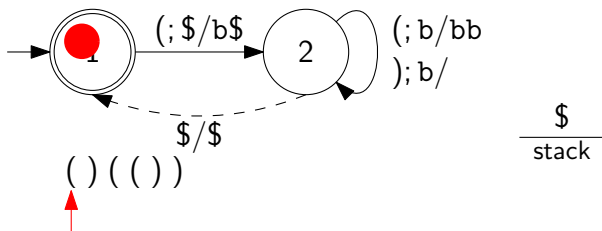




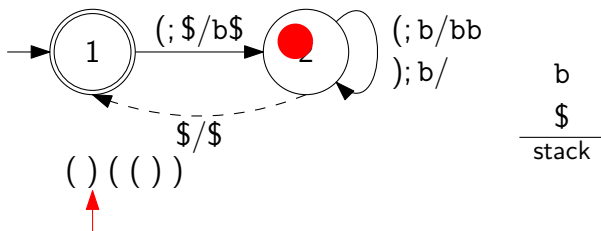
棄却する例



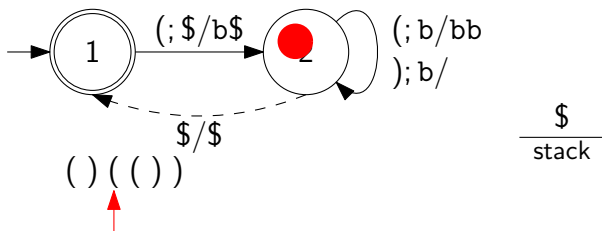
受理する例



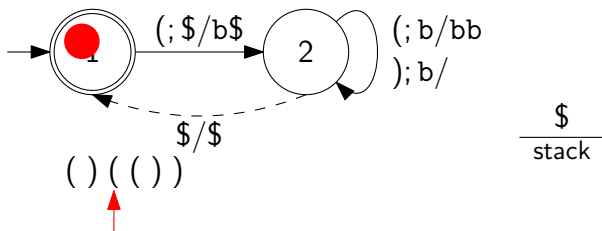
受理する例



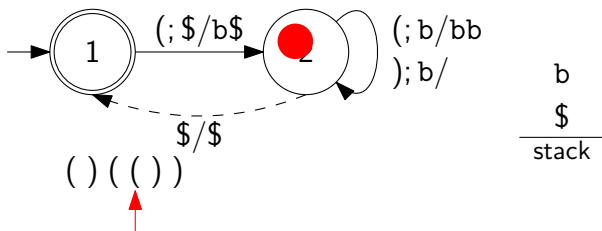
受理する例



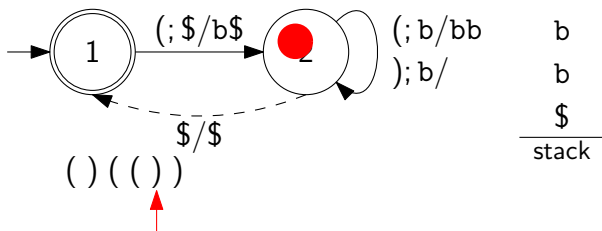
## 受理する例



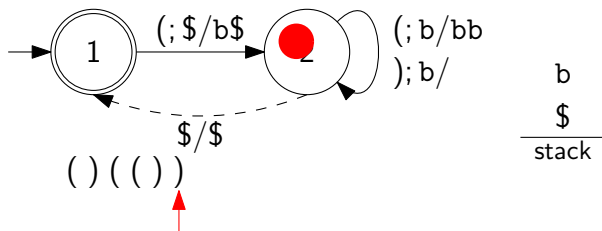
## 受理する例



受理する例

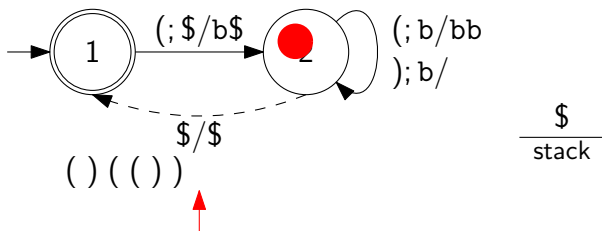


受理する例

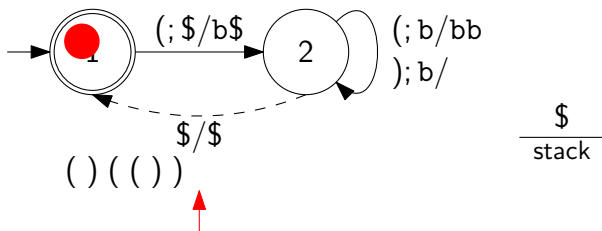




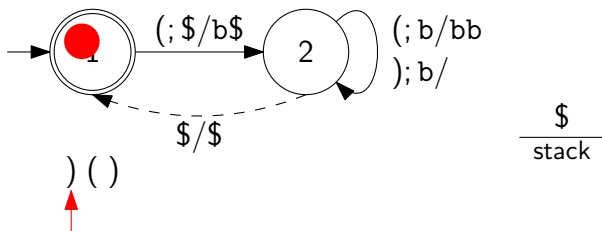
受理する例



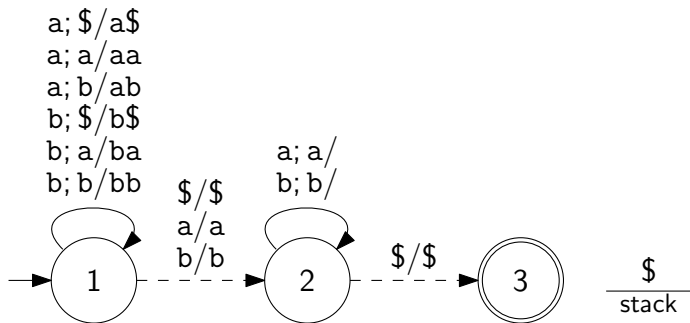
受理する例



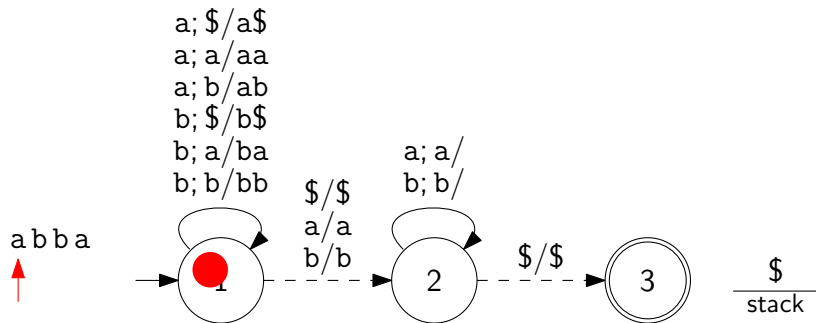
棄却する例



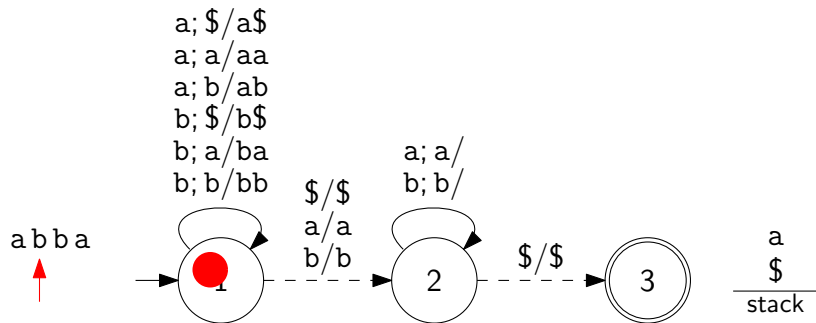
- ① 決定性プッシュダウン・オートマトン
- ② 非決定性プッシュダウン・オートマトン
- ③ プッシュダウン・オートマトンによるパース
- ④ 個人プロジェクト案の例



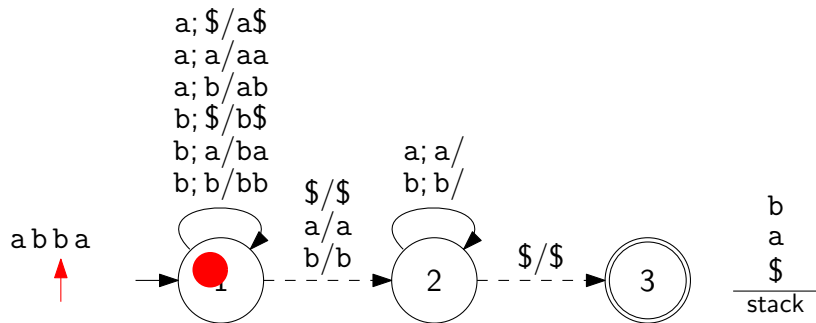
## 受理する例



## 受理する例

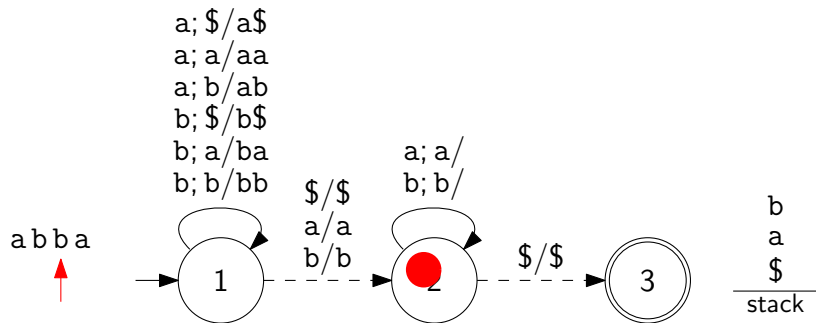


## 受理する例

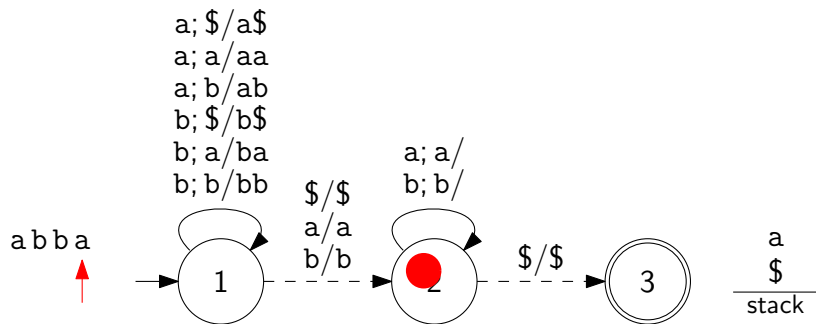




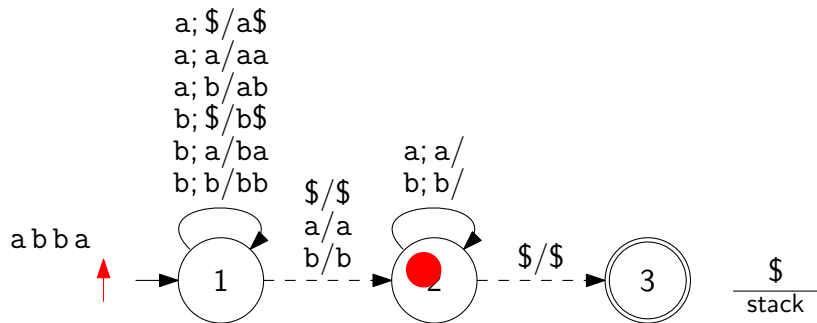
## 受理する例



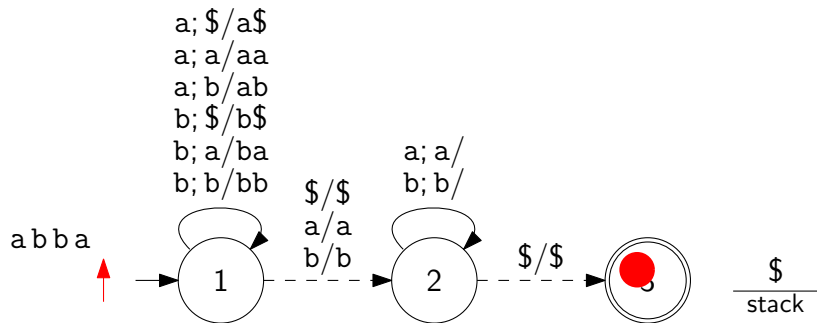
## 受理する例



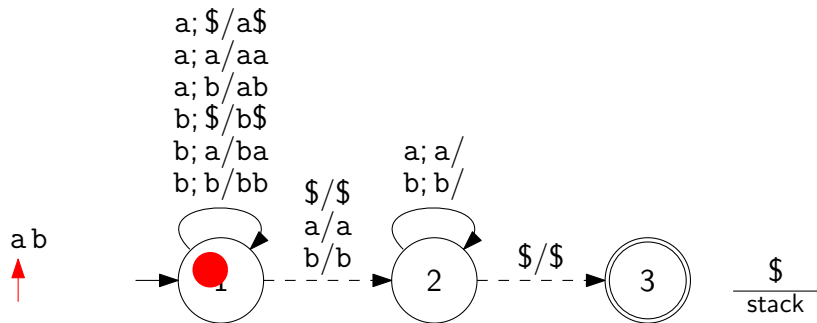
## 受理する例



## 受理する例



## 棄却する例



$A \subseteq B$  「AでできることはBでもできる」

$A \neq B$  「AでできることとBでできることは異なる」

### 計算能力の比較

$$\text{DFA} = \text{NFA} \subsetneq \text{DPDA} \subsetneq \text{NPDA}$$

- ▶ DPDA \ NFA の例：バランスのとれた括弧列
- ▶ NPDA \ DPDA の例：回文

### 注意

単に「プッシュダウン・オートマトン」と言ったら  
「非決定性プッシュダウン・オートマトン」を指す

- ① 決定性プッシュダウン・オートマトン
- ② 非決定性プッシュダウン・オートマトン
- ③ プッシュダウン・オートマトンによるパース
- ④ 個人プロジェクト案の例

2つの部分に分かれる

## 字句解析 (lexical analysis)

- ▶ 「文字列」を「トークン列」に変換

正規表現を使って (つまり, 有限オートマトンを使って) 実装

## 構文解析 (syntactic analysis)

- ▶ 「トークン列」が有効なプログラムを表現しているか判断

文脈自由文法を使って (つまり, PDA を使って) 実装

自然言語との対応

プログラミング言語	自然言語
プログラム	文章
文字列	文
トークン	品詞
有効なプログラム	文法に従った文章



## SIMPLE の文法 (の一部)

```
<statement> ::= <while> | <assign>
<while>      ::= 'w' '(' <expression> ')' '{' <statement> '}'
<assign>     ::= 'v' '=' <expression>
<expression> ::= <less-than>
<less-than>  ::= <multiply> '<' <less-than> | <multiply>
<multiply>   ::= <term> '*' <multiply> | <term>
<term>       ::= 'n' | 'v'
```

```
while (x < 3) x = x * 4
```

↓字句解析↓

```
w ( v < n ) v = v * n
```

↓構文解析↓

```
w ( v < n ) { v = v * n }
```

```
while (x < 3) x = x * 4
```

↓ 字句解析 ↓

```
w ( v < n ) v = v * n
```

↓ 構文解析 ↓

$$w \left( \underbrace{v}_{\langle \text{term} \rangle} < \underbrace{n}_{\langle \text{term} \rangle} \right) \left\{ v = \underbrace{v}_{\langle \text{term} \rangle} * \underbrace{n}_{\langle \text{term} \rangle} \right\}$$

```
while (x < 3) x = x * 4
```

↓ 字句解析 ↓

```
w ( v < n ) v = v * n
```

↓ 構文解析 ↓

```
w ( v < n ) { v = v * n }
```

<term>
<term>
<term>
<term>
<multiply>

```
while (x < 3) x = x * 4
```

↓ 字句解析 ↓

```
w ( v < n ) v = v * n
```

↓ 構文解析 ↓

$$w \left( \underbrace{v}_{\langle \text{term} \rangle} < \underbrace{n}_{\langle \text{term} \rangle} \right) \left\{ v = \underbrace{v}_{\langle \text{term} \rangle} * \underbrace{n}_{\langle \text{term} \rangle} \right\}$$

$$\underbrace{\hspace{15em}}_{\langle \text{multiply} \rangle}$$

$$\underbrace{\hspace{15em}}_{\langle \text{multiply} \rangle}$$

while (x < 3) x = x \* 4

↓ 字句解析 ↓

w ( v < n ) v = v \* n

↓ 構文解析 ↓

$$w \left( \underbrace{v}_{\langle \text{term} \rangle} < \underbrace{n}_{\langle \text{term} \rangle} \right) \left\{ v = \underbrace{v}_{\langle \text{term} \rangle} * \underbrace{n}_{\langle \text{term} \rangle} \right\}$$

$$\underbrace{\hspace{15em}}_{\langle \text{multiply} \rangle}$$

$$\underbrace{\hspace{15em}}_{\langle \text{less-than} \rangle}$$

```
while (x < 3) x = x * 4
```

↓ 字句解析 ↓

```
w ( v < n ) v = v * n
```

↓ 構文解析 ↓

```
w ( v < n ) { v = v * n }
```

$\frac{\frac{\frac{\text{<multiply>}}{\text{<less-than>}}}{\text{<expression>}}}{\text{<term>}}$

while (x < 3) x = x \* 4

↓ 字句解析 ↓

w ( v < n ) v = v \* n

↓ 構文解析 ↓

$$\begin{array}{c}
 w \quad ( \quad \underline{v} \quad < \quad \underline{n} \quad ) \quad \{ \quad v \quad = \quad \underline{v} \quad * \quad \underline{n} \quad \} \\
 \quad \quad \quad \underline{\langle \text{term} \rangle} \quad \quad \underline{\langle \text{term} \rangle} \quad \quad \quad \quad \quad \quad \underline{\langle \text{term} \rangle} \quad \quad \underline{\langle \text{term} \rangle} \\
 \quad \underline{\langle \text{multiply} \rangle} \\
 \quad \underline{\langle \text{multiply} \rangle} \\
 \quad \underline{\langle \text{less-than} \rangle} \\
 \quad \underline{\langle \text{expression} \rangle} \\
 \quad \underline{\langle \text{assign} \rangle}
 \end{array}$$



```
while (x < 3) x = x * 4
```

↓ 字句解析 ↓

```
w ( v < n ) v = v * n
```

↓ 構文解析 ↓

w	(	<u>v</u>	<	<u>n</u>	)	{	v	=	<u>v</u>	*	<u>n</u>	}
		<u>&lt;term&gt;</u>		<u>&lt;term&gt;</u>					<u>&lt;term&gt;</u>		<u>&lt;term&gt;</u>	
											<u>&lt;multiply&gt;</u>	
											<u>&lt;multiply&gt;</u>	
											<u>&lt;less-than&gt;</u>	
											<u>&lt;expression&gt;</u>	
											<u>&lt;assign&gt;</u>	
											<u>&lt;statement&gt;</u>	

```
while (x < 3) x = x * 4
```

↓ 字句解析 ↓

```
w ( v < n ) v = v * n
```

↓ 構文解析 ↓

w	(	<u>v</u>	<	<u>n</u>	)	{	v	=	<u>v</u>	*	<u>n</u>	}
		<u>&lt;term&gt;</u>		<u>&lt;term&gt;</u>					<u>&lt;term&gt;</u>		<u>&lt;term&gt;</u>	
		<u>&lt;multiply&gt;</u>							<u>&lt;multiply&gt;</u>			
									<u>&lt;multiply&gt;</u>			
									<u>&lt;less-than&gt;</u>			
									<u>&lt;expression&gt;</u>			
									<u>&lt;assign&gt;</u>			
									<u>&lt;statement&gt;</u>			

while (x < 3) x = x \* 4

↓ 字句解析 ↓

w ( v < n ) v = v \* n

↓ 構文解析 ↓

w	(	<u>v</u>	<	<u>n</u>	)	{	v	=	<u>v</u>	*	<u>n</u>	}
		<u>&lt;term&gt;</u>		<u>&lt;term&gt;</u>					<u>&lt;term&gt;</u>		<u>&lt;term&gt;</u>	
		<u>&lt;multiply&gt;</u>		<u>&lt;multiply&gt;</u>					<u>&lt;multiply&gt;</u>			
		<u>&lt;less-than&gt;</u>							<u>&lt;multiply&gt;</u>			
									<u>&lt;less-than&gt;</u>			
									<u>&lt;expression&gt;</u>			
									<u>&lt;assign&gt;</u>			
									<u>&lt;statement&gt;</u>			

while (x < 3) x = x \* 4

↓ 字句解析 ↓

w ( v < n ) v = v \* n

↓ 構文解析 ↓

w	(	<u>v</u>	<	<u>n</u>	)	{	v	=	<u>v</u>	*	<u>n</u>	}
		<u>&lt;term&gt;</u>		<u>&lt;term&gt;</u>					<u>&lt;term&gt;</u>		<u>&lt;term&gt;</u>	
		<u>&lt;multiply&gt;*multiply&gt;</u>							<u>&lt;multiply&gt;</u>			
		<u>&lt;less-than&gt;</u>							<u>&lt;multiply&gt;</u>			
		<u>&lt;less-than&gt;</u>							<u>&lt;less-than&gt;</u>			
									<u>&lt;expression&gt;</u>			
									<u>&lt;assign&gt;</u>			
									<u>&lt;statement&gt;</u>			

while (x < 3) x = x \* 4

↓ 字句解析 ↓

w ( v < n ) v = v \* n

↓ 構文解析 ↓

w	(	<u>v</u>	<	<u>n</u>	)	{	v	=	<u>v</u>	*	<u>n</u>	}
		<u>&lt;term&gt;</u>		<u>&lt;term&gt;</u>					<u>&lt;term&gt;</u>		<u>&lt;term&gt;</u>	
		<u>&lt;multiply&gt;*multiply&gt;</u>							<u>&lt;multiply&gt;</u>			
			<u>&lt;less-than&gt;</u>						<u>&lt;multiply&gt;</u>			
		<u>&lt;less-than&gt;</u>							<u>&lt;less-than&gt;</u>			
		<u>&lt;expression&gt;</u>							<u>&lt;expression&gt;</u>			
									<u>&lt;assign&gt;</u>			
									<u>&lt;statement&gt;</u>			

while (x < 3) x = x \* 4

↓ 字句解析 ↓

w ( v < n ) v = v \* n

↓ 構文解析 ↓

w	(	<u>v</u>	<	<u>n</u>	)	{	v	=	<u>v</u>	*	<u>n</u>	}
		<u>&lt;term&gt;</u>		<u>&lt;term&gt;</u>					<u>&lt;term&gt;</u>		<u>&lt;term&gt;</u>	
		<u>&lt;multiply&gt;</u>	*	<u>&lt;multiply&gt;</u>					<u>&lt;multiply&gt;</u>		<u>&lt;multiply&gt;</u>	
			<u>&lt;less-than&gt;</u>						<u>&lt;multiply&gt;</u>		<u>&lt;multiply&gt;</u>	
		<u>&lt;less-than&gt;</u>							<u>&lt;less-than&gt;</u>		<u>&lt;less-than&gt;</u>	
		<u>&lt;expression&gt;</u>							<u>&lt;expression&gt;</u>		<u>&lt;expression&gt;</u>	
									<u>&lt;assign&gt;</u>		<u>&lt;assign&gt;</u>	
									<u>&lt;statement&gt;</u>		<u>&lt;statement&gt;</u>	
<u>&lt;while&gt;</u>												

while (x < 3) x = x \* 4

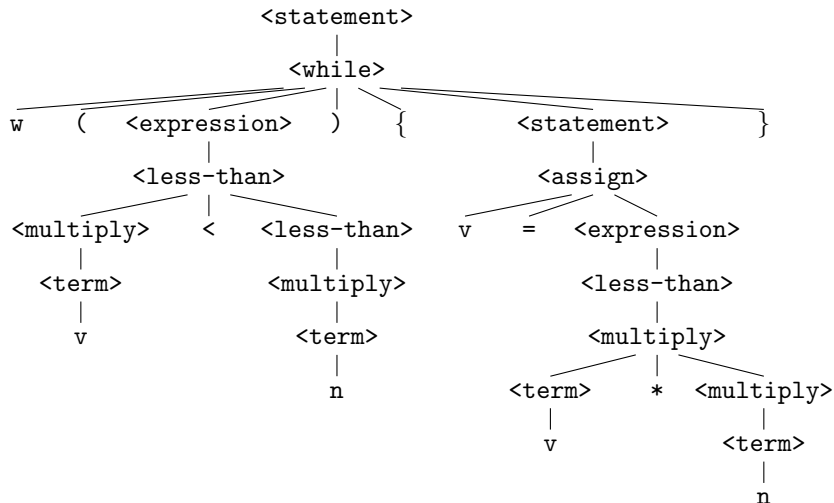
↓ 字句解析 ↓

w ( v < n ) v = v \* n

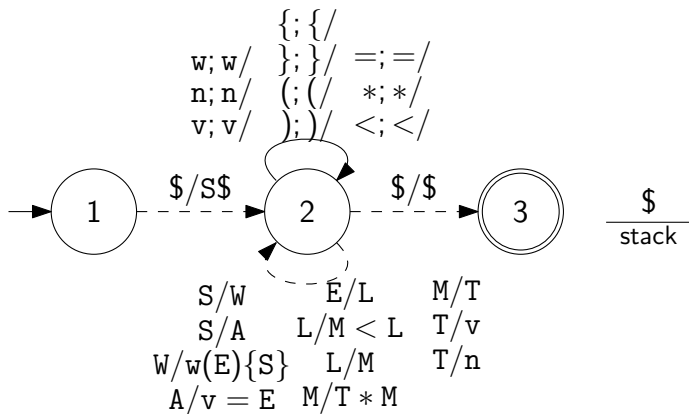
↓ 構文解析 ↓

w	(	<u>v</u>	<	<u>n</u>	)	{	v	=	<u>v</u>	*	<u>n</u>	}
		<term>		<term>					<term>		<term>	
		<multiply>	*	<multiply>					<multiply>		<multiply>	
			<less-than>						<multiply>			
		<less-than>							<less-than>			
		<expression>							<expression>			
									<assign>			
									<statement>			
<while>												
<statement>												

## 構文解析の結果として得られる木構造







「 $w ( v < n ) \quad v = v * n$ 」の構文解析に必要な部分だけ

- ① 決定性プッシュダウン・オートマトン
- ② 非決定性プッシュダウン・オートマトン
- ③ プッシュダウン・オートマトンによるパース
- ④ 個人プロジェクト案の例

- ▶ クラス DPDA, NPDA にメソッドを追加する
  - ▶ 文字列を入力して、状態とスタックの変化を一文字ずつ追い、到達した状態とスタックの内容を画面に逐一出力するメソッド `trace(string)` を実装する
- ▶ 4.3.1 節「字句解析」を第3章で実装した正規表現を使って構築し直す
- ▶ 4.3.2 節「構文解析」を完成させる
- ▶ LL 法以外の構文解析アルゴリズムを調べて、実装する
  - ▶ 例えば、LR 法
- ▶ 「プッシュダウン・オートマトンの類似概念」について調べて、実装してみる