
COMP20121 The Implementation and Power of Computer Languages

'Power' Part

<http://www.cs.man.ac.uk/~petera/2121/index.html> .

Peter Aczel

room: CS2.52, tel: 56155

email: `petera@cs.man.ac.uk`

Department of Computer Science, University of Manchester

COMP20121, 'power' part: section 2

lecture 4: Context-free Grammars

LECTURE FIVE

Pushdown Automata

Pushdown automata

Idea: Add memory to automaton (stack).

Pushdown automata

Idea: Add memory to automaton (stack).

Then: transition function depends on

Pushdown automata

Idea: Add memory to automaton (stack).

Then: transition function depends on

- current state;

Pushdown automata

Idea: Add memory to automaton (stack).

Then: transition function depends on

- current state;
- current input symbol;

Pushdown automata

Idea: Add memory to automaton (stack).

Then: transition function depends on

- current state;
- current input symbol;
- current top of stack.

Pushdown automata–II

Idea: Automaton will need to

- push symbols from $\Sigma \cup N$ onto stack;

Pushdown automata–II

Idea: Automaton will need to

- push symbols from $\Sigma \cup N$ onto stack;
- pop symbols off the stack.

Pushdown automata–II

Idea: Automaton will need to

- push symbols from $\Sigma \cup N$ onto stack;
- pop symbols off the stack.

We want to allow it to just perform stack operations *without advancing on the input string*. (This will typically happen when there is a non-terminal symbol on top of the stack.)

Pushdown automata–II

So we get a third type of action:

- Advance on the input string.

Pushdown automata–II

So we get a third type of action:

- Advance on the input string.

So the possible actions are:

- pop
- push(X) (for $X \in \Sigma \cup N$)
- advance

Pushdown automata–II

So we get a third type of action:

- Advance on the input string.

So the possible actions are:

- pop
- push(X) (for $X \in \Sigma \cup N$)
- advance

Let A = set of all actions

Pushdown automata–III

For the transition function, need to have new symbols, not in $\Sigma \cup N$, for

Pushdown automata–III

For the transition function, need to have new symbols, not in $\Sigma \cup N$, for

- ‘the stack is empty’: EOS;

Pushdown automata–III

For the transition function, need to have new symbols, not in $\Sigma \cup N$, for

- ‘the stack is empty’: EOS;
- ‘the end of the input string has been reached’: EOF.

Pushdown automata–III

So the transition function will be from:

$$Q \times (\Sigma \cup \{\text{EOF}\}) \times (\Sigma \cup N \cup \{\text{EOS}\})$$

to $A^* \times Q$,

Pushdown automata–III

So the transition function will be from:

$$Q \times (\Sigma \cup \{\text{EOF}\}) \times (\Sigma \cup N \cup \{\text{EOS}\})$$

to $A^* \times Q$, sending

(state , input symbol , top of stack)

to

(list of actions , new state)

Transitions

So a transition, sending (q, x, X) to
(list of actions, q') can be written:

$$q \xrightarrow{(x, X) \mapsto \langle \text{list of actions} \rangle} q'.$$

where

Transitions

So a transition, sending (q, x, X) to $(\text{list of actions}, q')$ can be written:

$$q \xrightarrow{(x, X) \mapsto \langle \text{list of actions} \rangle} q'.$$

where q, q' are states, x is the input symbol, X is the symbol at the top of the stack and **list of actions** is in A^* .

Transitions

Use

$$q \quad (, X) \mapsto \langle \text{list of actions} \rangle \quad q'$$

\longrightarrow

to indicate that the transition occurs no matter what the current input symbol is.

Transitions

Use

$$q \quad (x,) \mapsto \langle \text{list of actions} \rangle \quad q'$$

\longrightarrow

to indicate that the transition occurs no matter what the current top of the stack is.

Transitions

Use

$$q \quad (,) \mapsto \langle \text{list of actions} \rangle \quad q'$$

\longrightarrow

to indicate that the transition occurs no matter what the current input symbol or the current top of the stack is.

Pushdown automata: Definition

Definition 12 A deterministic pushdown automaton or *DPDA* over an alphabet $\Sigma \cup N$ is given by the following:

$\Sigma \cup N$ is given by the following:

- a finite set Q of states;

Pushdown automata: Definition

Definition 12 A deterministic pushdown automaton or *DPDA* over an alphabet $\Sigma \cup N$ is given by the following:

$\Sigma \cup N$ is given by the following:

- a finite set Q of states;
- a start state $q_{\bullet} \in Q$;

Pushdown automata: Definition

Definition 12 A deterministic pushdown automaton or *DPDA* over an alphabet $\Sigma \cup N$ is given by the following:

$\Sigma \cup N$ is given by the following:

- a finite set Q of states;
- a start state $q_{\bullet} \in Q$;
- a set $F \subseteq Q$ of accepting states;

Pushdown automata: Definition

- a transition function δ from $Q \times (\Sigma \cup \{\text{EOF}\}) \times (\Sigma \cup N \cup \{\text{EOS}\})$ to $A^* \times Q$ where

Pushdown automata: Definition

- a transition function δ from $Q \times (\Sigma \cup \{\text{EOF}\}) \times (\Sigma \cup N \cup \{\text{EOS}\})$ to $A^* \times Q$ where
 - ▶ EOS and EOF are special symbols which are not in $\Sigma \cup N$ and

Pushdown automata: Definition

- a transition function δ from $Q \times (\Sigma \cup \{\text{EOF}\}) \times (\Sigma \cup N \cup \{\text{EOS}\})$ to $A^* \times Q$ where
 - ▶ EOS and EOF are special symbols which are not in $\Sigma \cup N$ and
 - ▶ A is the set of all actions pop, push(X) (for $X \in \Sigma \cup N$) and advance.

Non-deterministic choice

Definition 12 *(continued)* A **non-deterministic pushdown automaton**, *NPDA* or *PDA* differs in this only by the fact that δ is a relation rather than a function.

Accepting a word

Definition 13 *A string is accepted by a pushdown automaton if*

- *starting with an empty stack,*

Accepting a word

Definition 13 *A string is accepted by a pushdown automaton if*

- *starting with an empty stack,*
- *there is a path through the automaton*

Accepting a word

Definition 13 *A string is accepted by a pushdown automaton if*

- *starting with an empty stack,*
- *there is a path through the automaton*
- *ending as soon as EOF has been reached when in an accepting state.*

Accepting a word

Definition 13 *A string is accepted by a pushdown automaton if*

- *starting with an empty stack,*
- *there is a path through the automaton*
- *ending as soon as EOF has been reached when in an accepting state.*

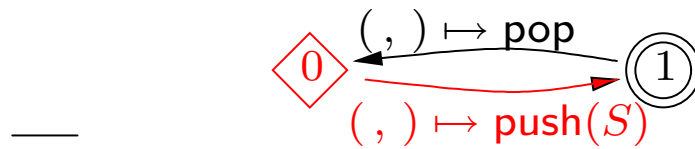
The content of the stack at that time is irrelevant.

PDA: Example

A (D)PDA may run forever.

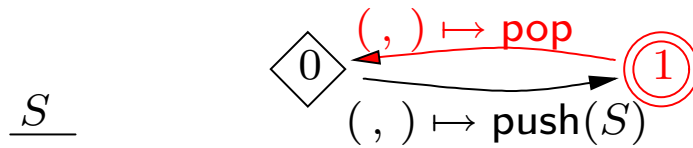
PDA: Example

A (D)PDA may run forever.



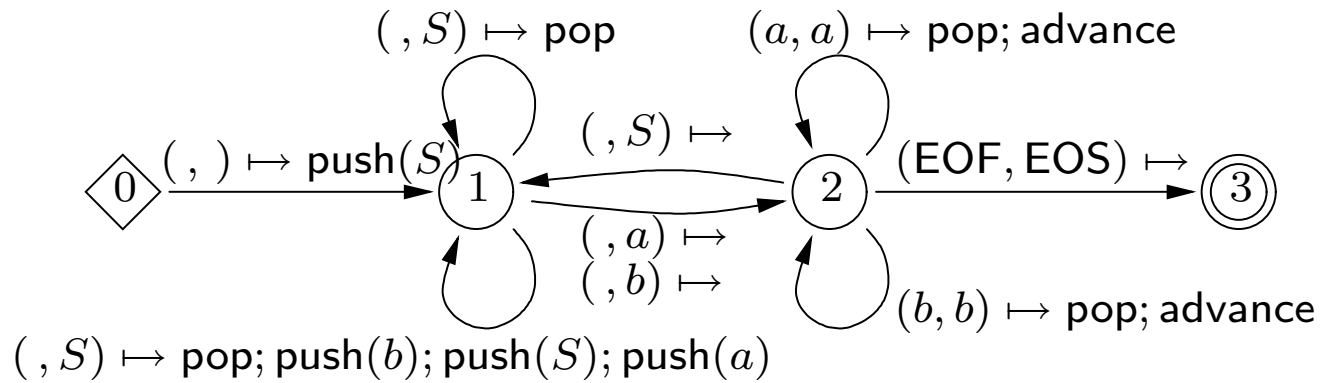
PDA: Example

A (D)PDA may run forever.



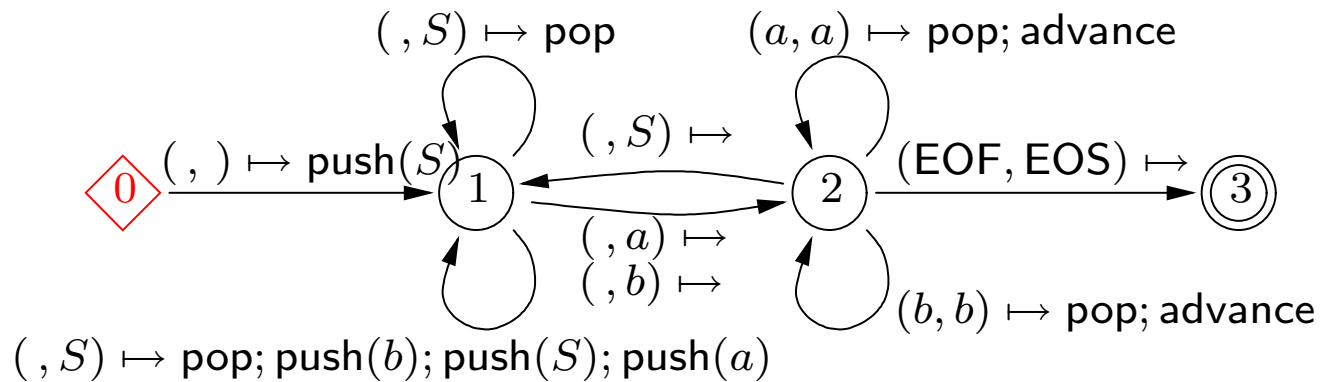
PDA: Example

A PDA.



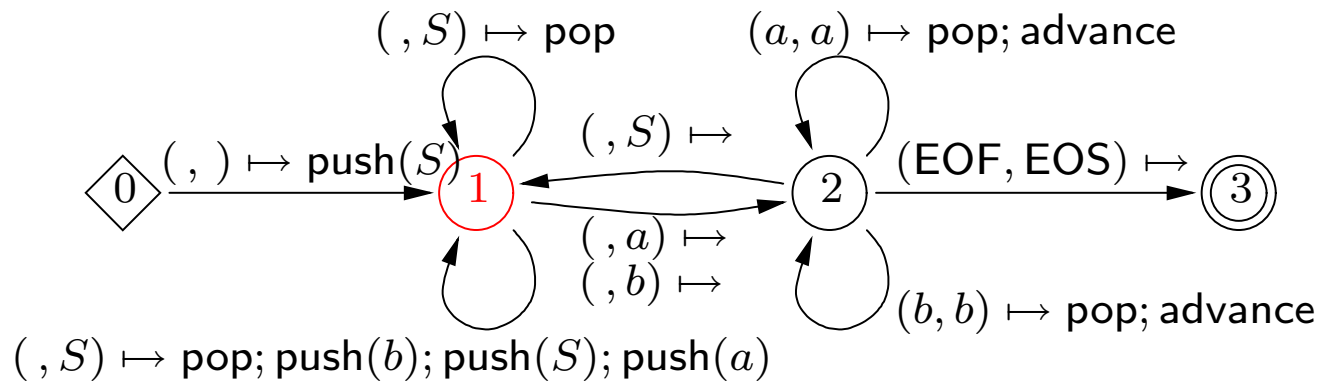
PDA: Example

A PDA.



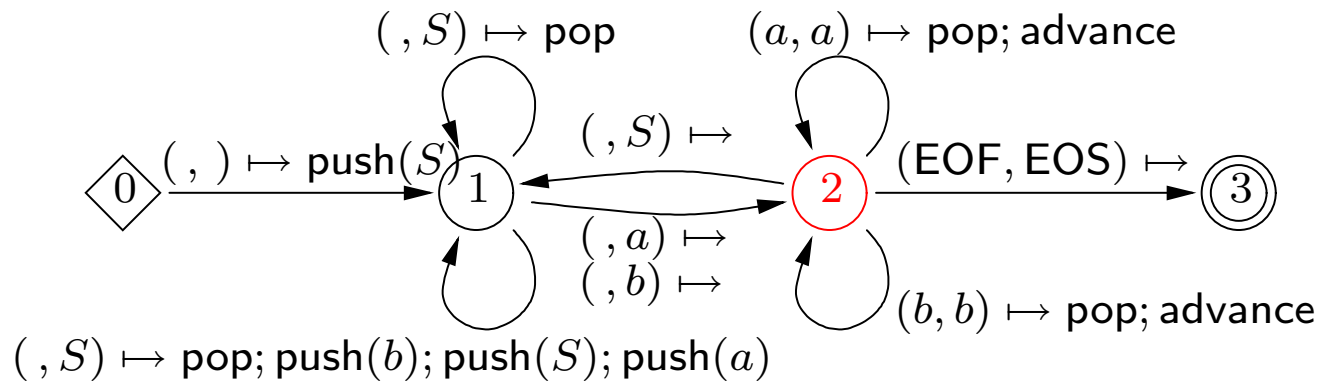
PDA: Example

A PDA.



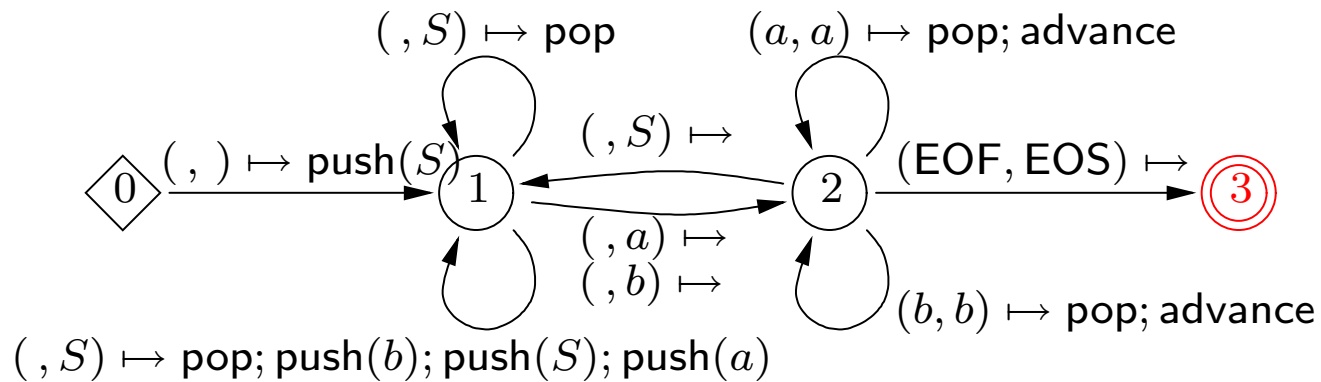
PDA: Example

A PDA.



PDA: Example

A PDA.



PDA: Example

An alternative presentation of the same automaton:

configuration		

PDA: Example

An alternative presentation of the same automaton:

configuration			actions	new state
state	input symbol	top of stack		

PDA: Example

An alternative presentation of the same automaton:

configuration			actions	new state
state	input symbol	top of stack		
0			push(S)	1

PDA: Example

An alternative presentation of the same automaton:

configuration			actions	new state
input state	symbol	top of stack		
0			push(S)	1
1		a		2

PDA: Example

An alternative presentation of the same automaton:

configuration			actions	new state
state	input symbol	top of stack		
0			push(S)	1
1		a		2
1		b		2

PDA: Example

An alternative presentation of the same automaton:

configuration			actions	new state
state	input symbol	top of stack		
0			push(S)	1
1		a		2
1		b		2
1		S	$\left\{ \begin{array}{l} \text{pop} \\ \text{pop}; \text{push}(b); \text{push}(S); \text{push}(a) \end{array} \right.$	1

PDA: Example

An alternative presentation of the same automaton:

configuration			actions	new state
state	input symbol	top of stack		
0			push(S)	1
1		a		2
1		b		2
1		S	$\left\{ \begin{array}{l} \text{pop} \\ \text{pop; push}(b); \text{push}(S); \text{push}(a) \end{array} \right.$	1
2		S		1

PDA: Example

An alternative presentation of the same automaton:

configuration			actions	new state
state	input symbol	top of stack		
0			push(S)	1
1		a		2
1		b		2
1		S	$\left\{ \begin{array}{l} \text{pop} \\ \text{pop; push}(b); \text{push}(S); \text{push}(a) \end{array} \right.$	1
2		S		1
2	a	a	pop; advance	2

PDA: Example

An alternative presentation of the same automaton:

configuration			actions	new state
state	input symbol	top of stack		
0			push(S)	1
1		a		2
1		b		2
1		S	$\left\{ \begin{array}{l} \text{pop} \\ \text{pop; push}(b); \text{push}(S); \text{push}(a) \end{array} \right.$	1
2		S		1
2	a	a	pop; advance	2
2	b	b	pop; advance	2

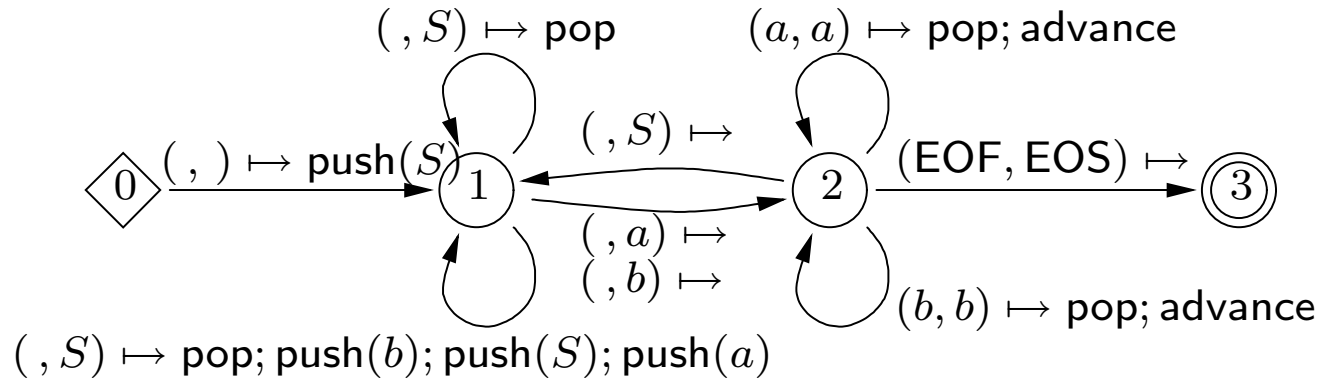
PDA: Example

An alternative presentation of the same automaton:

configuration			actions	new state
state	input symbol	top of stack		
0			push(S)	1
1		a		2
1		b		2
1		S	$\left\{ \begin{array}{l} \text{pop} \\ \text{pop; push}(b); \text{push}(S); \text{push}(a) \end{array} \right.$	1
2		S		1
2	a	a	pop; advance	2
2	b	b	pop; advance	2
2	EOF	EOS		3

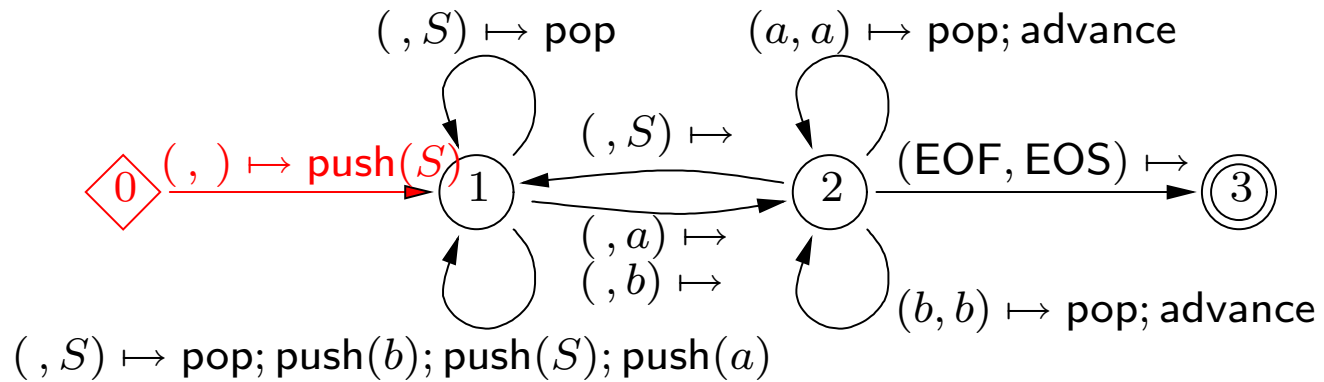
PDA processing an input string

aaabbb



PDA processing an input string

aaabbb

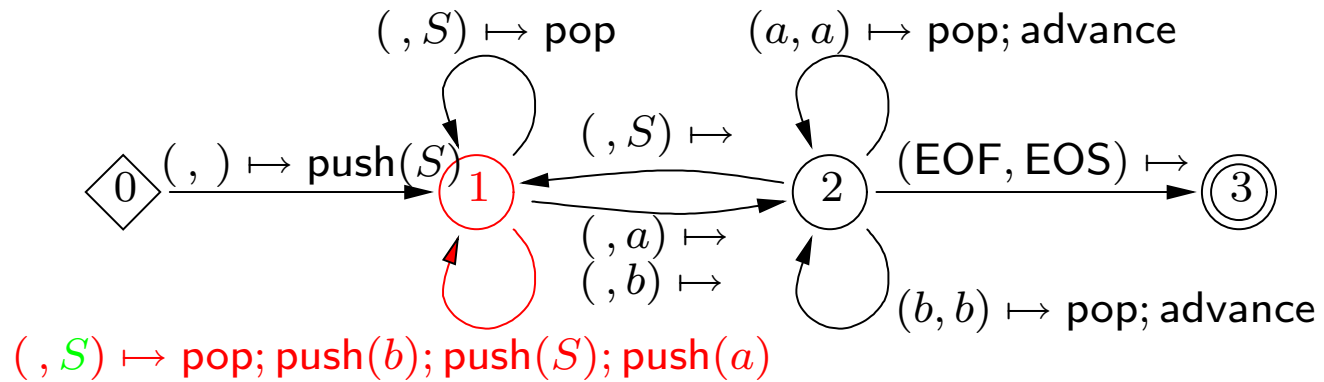


S

PDA processing an input string

aaabbb

S

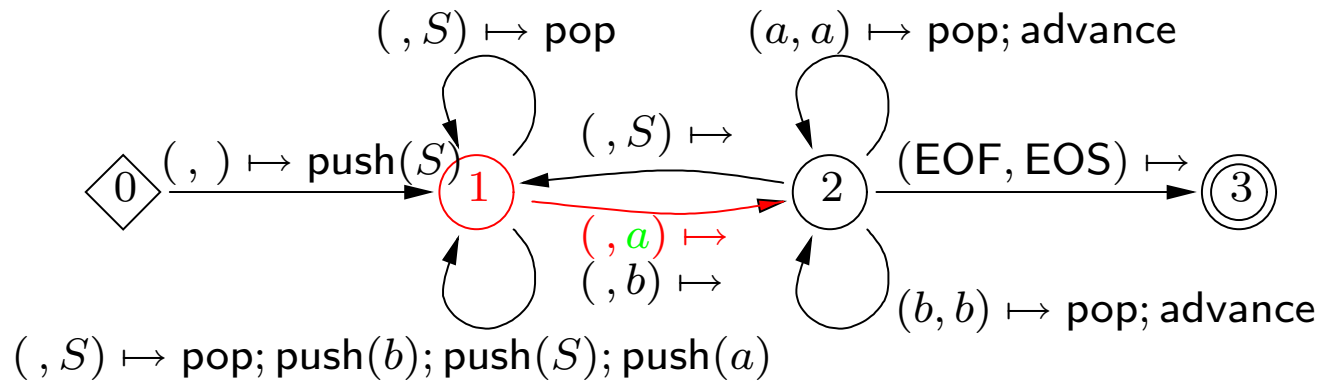


$$S \Rightarrow aSb$$

PDA processing an input string

aaabbb

a
S
b
—

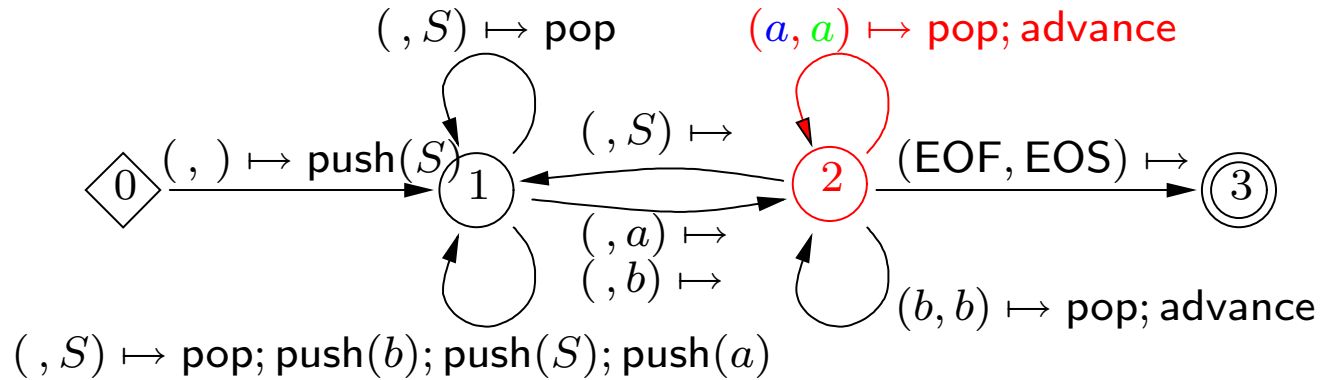


$$S \Rightarrow aSb$$

PDA processing an input string

aaabbb

a
S
b
—

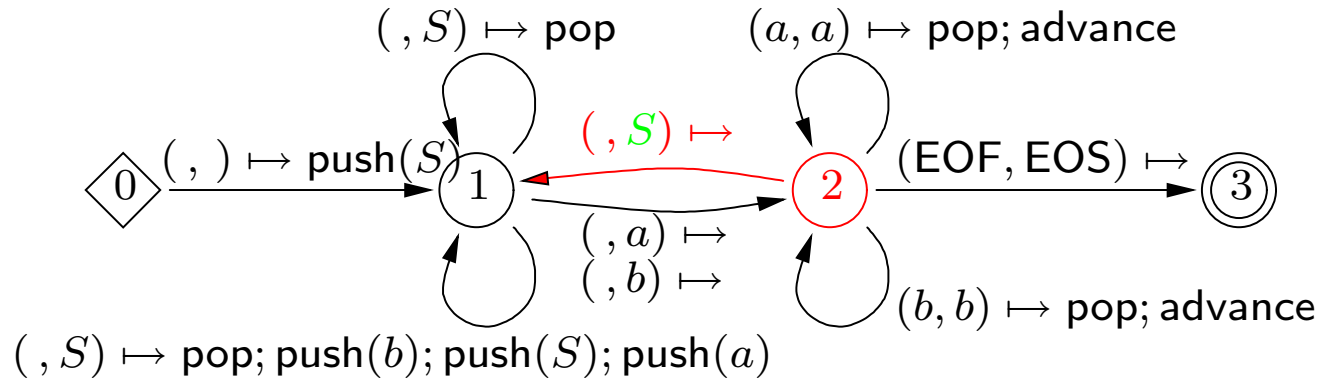


$$S \Rightarrow aSb$$

PDA processing an input string

aabbb

S
b

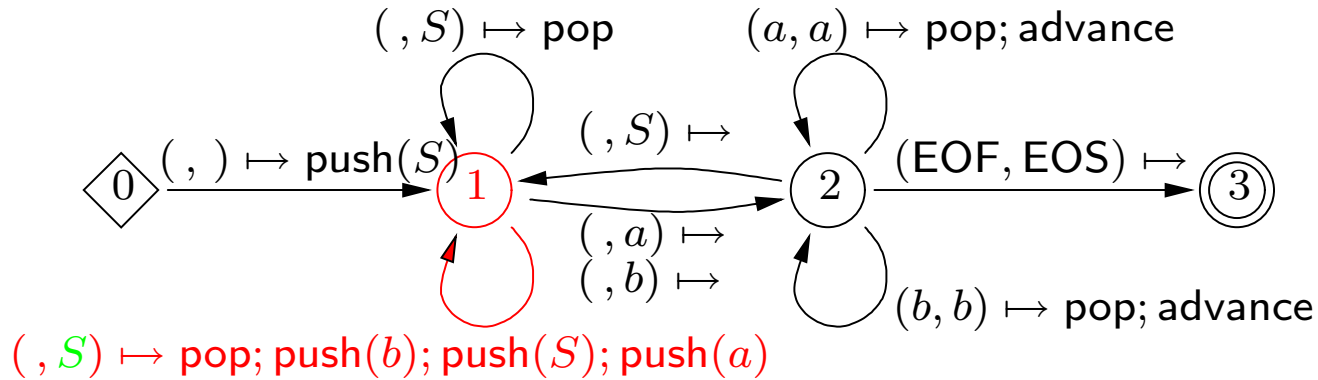


$$S \Rightarrow aSb$$

PDA processing an input string

aabbb

S
b

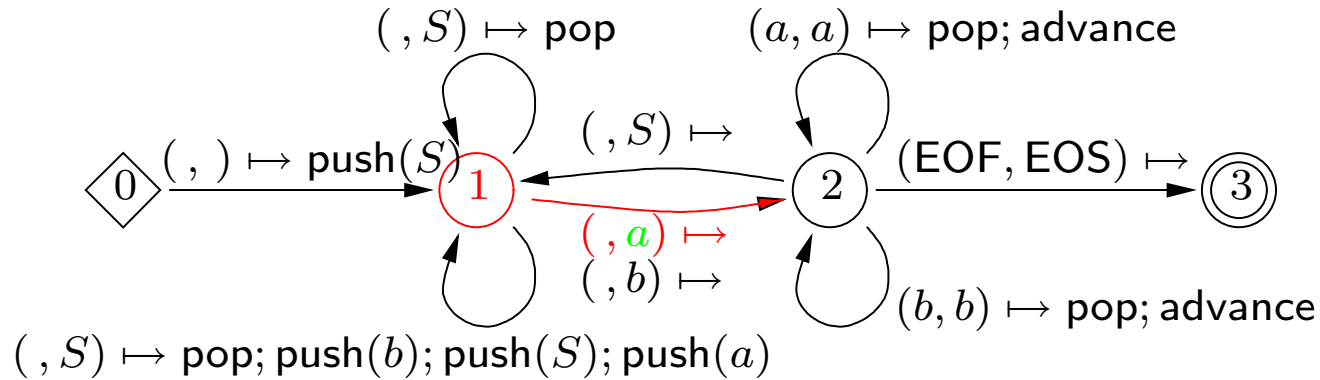


$$S \Rightarrow aSb \Rightarrow aaSbb$$

PDA processing an input string

aabbb

a
S
b
b
—

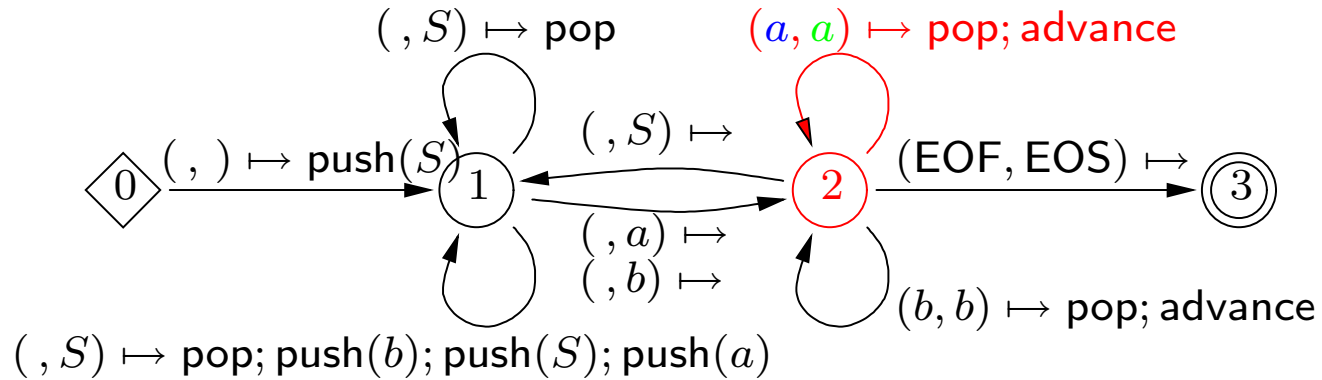


$$S \Rightarrow aSb \Rightarrow aaSbb$$

PDA processing an input string

aabbb

a
S
b
b
—

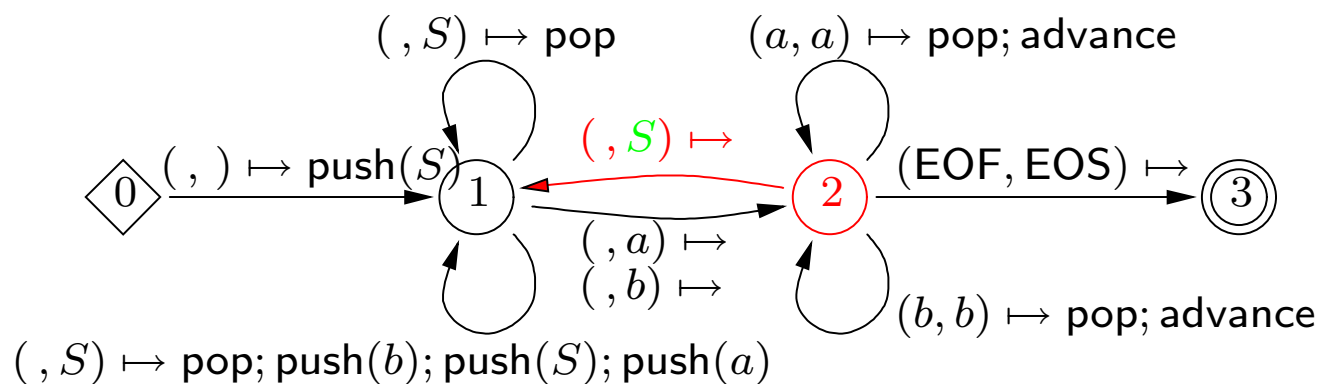


$$S \Rightarrow aSb \Rightarrow aaSbb$$

PDA processing an input string

abbb

S
b
b

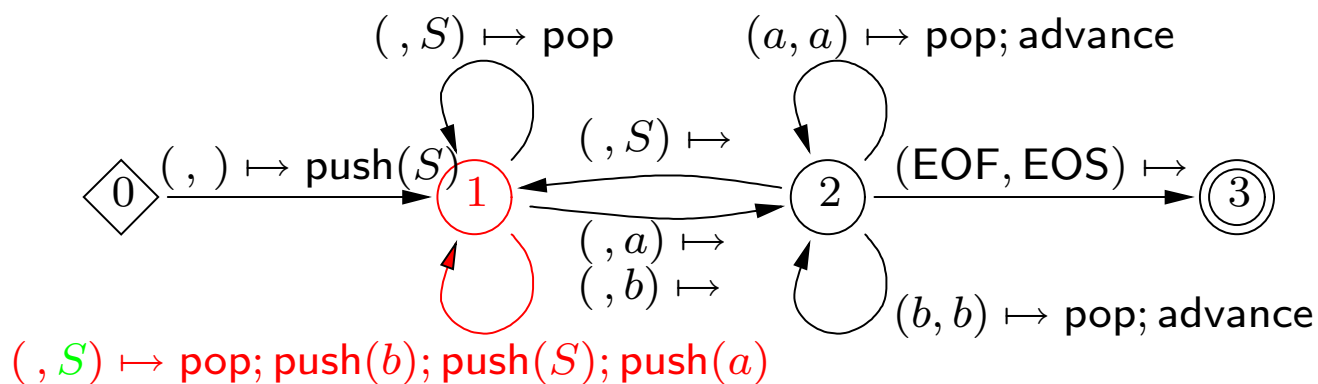


$$S \Rightarrow aSb \Rightarrow aaSbb$$

PDA processing an input string

abbb

S
b
b

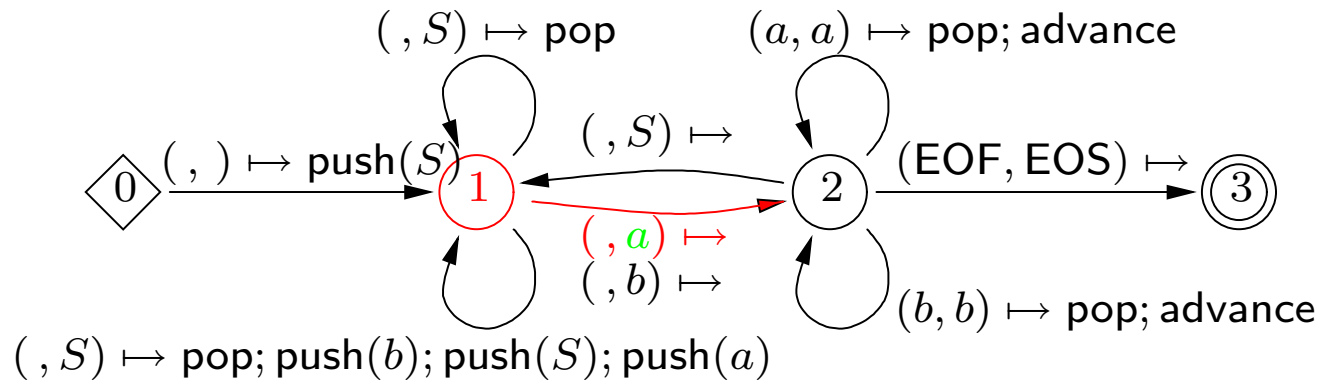


$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow \mathbf{aaaSbbb}$

PDA processing an input string

abbb

a
S
b
b
b

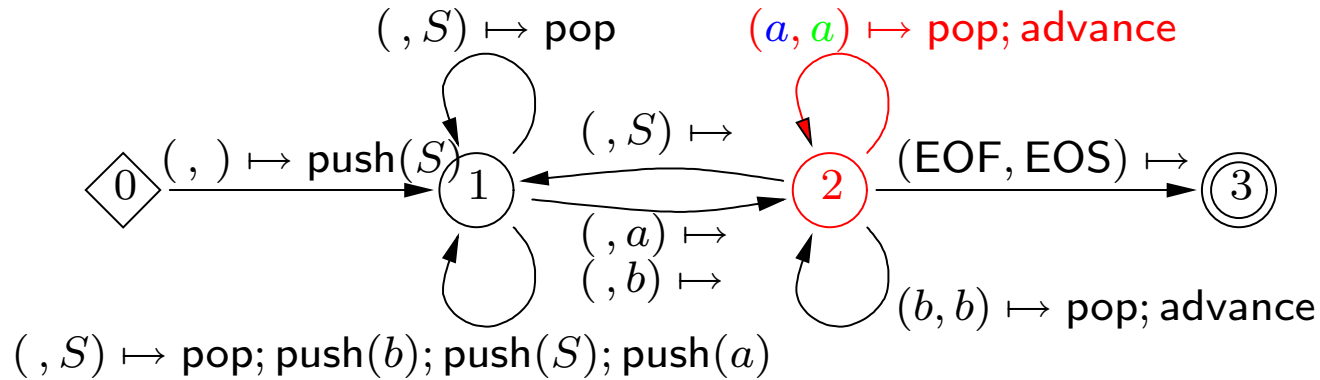


$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow \mathbf{aaaSbbb}$

PDA processing an input string

abbb

a
S
b
b
b

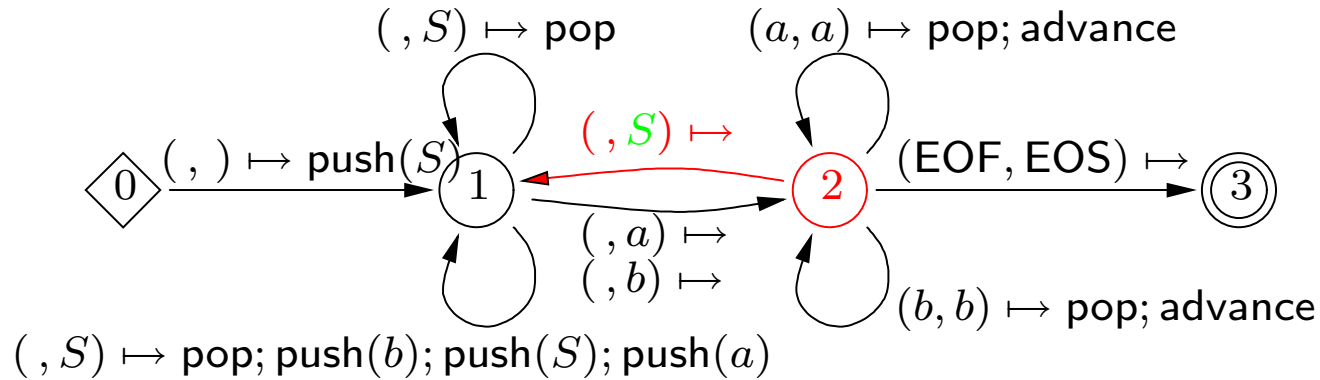


$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow \text{aaaSbbb}$

PDA processing an input string

bbb

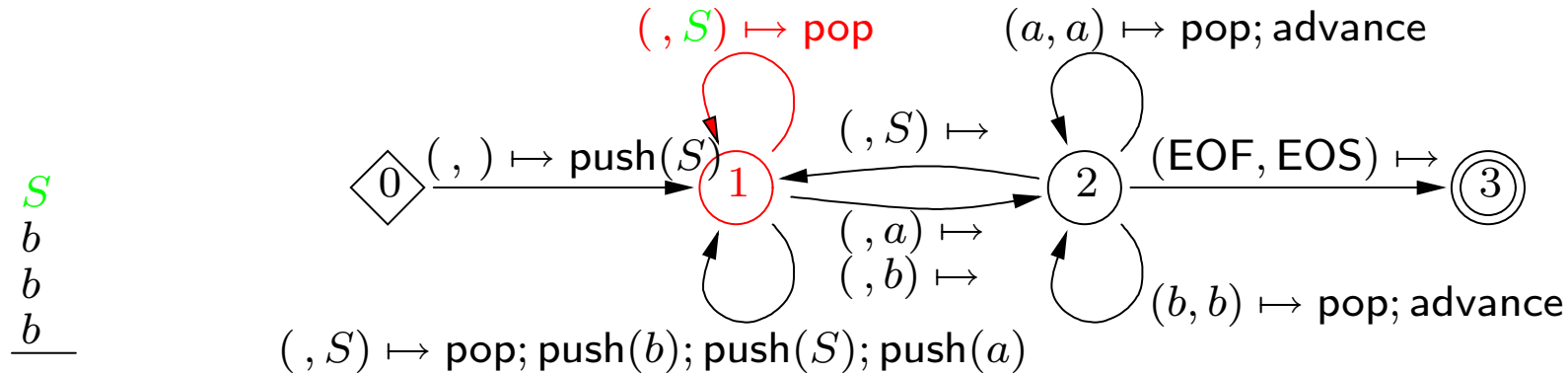
S
b
b
b



$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow \text{aaaSbbb}$

PDA processing an input string

bbb



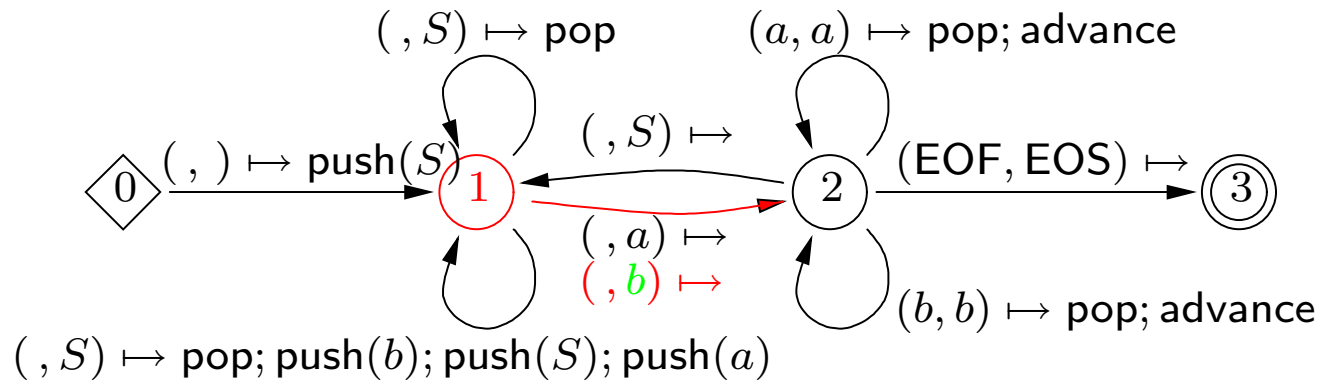
S
 b
 b
 b

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb$
 $\Rightarrow aabbbb$

PDA processing an input string

bbb

b
b
b

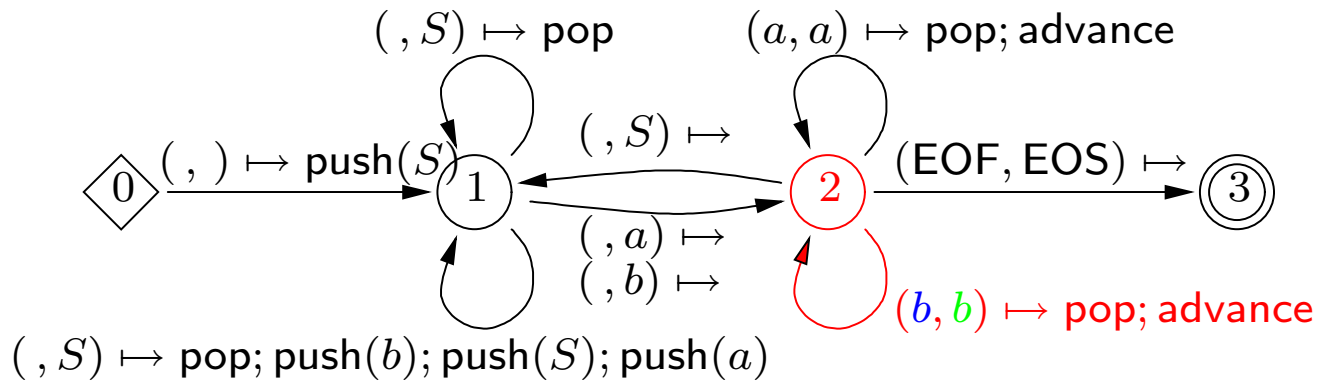


$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb$
 $\Rightarrow \text{aaa}bbb$

PDA processing an input string

bbb

b
b
b

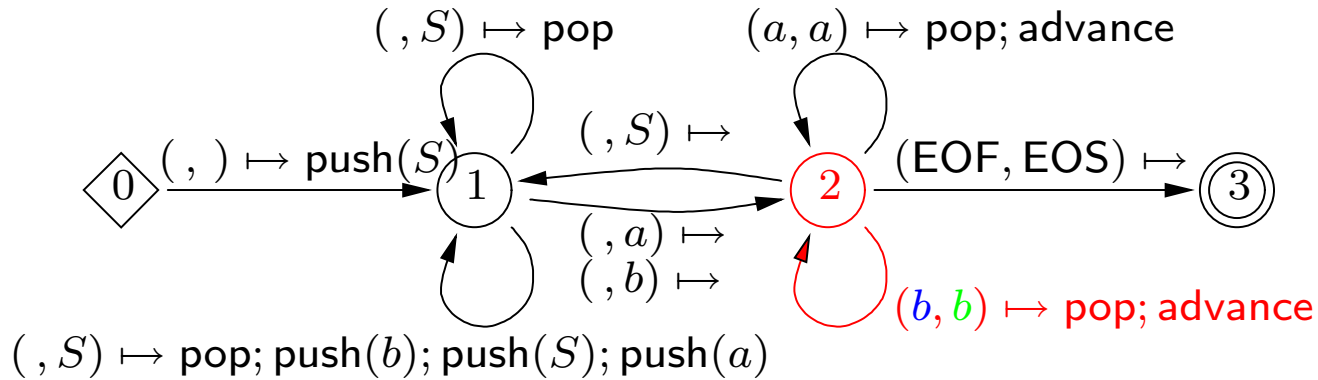


$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb$
 $\Rightarrow \text{aaabbb}$

PDA processing an input string

bb

$\frac{b}{b}$

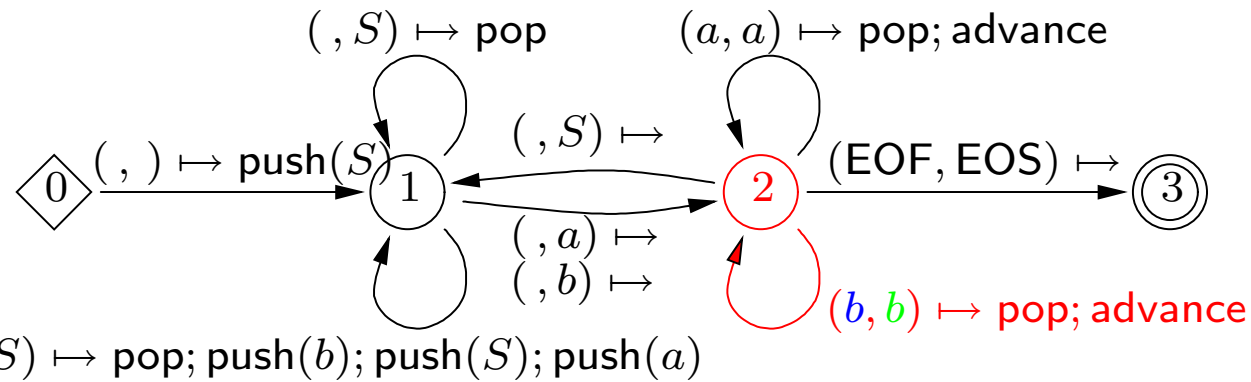


$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb$$

$$\Rightarrow aaabbb$$

PDA processing an input string

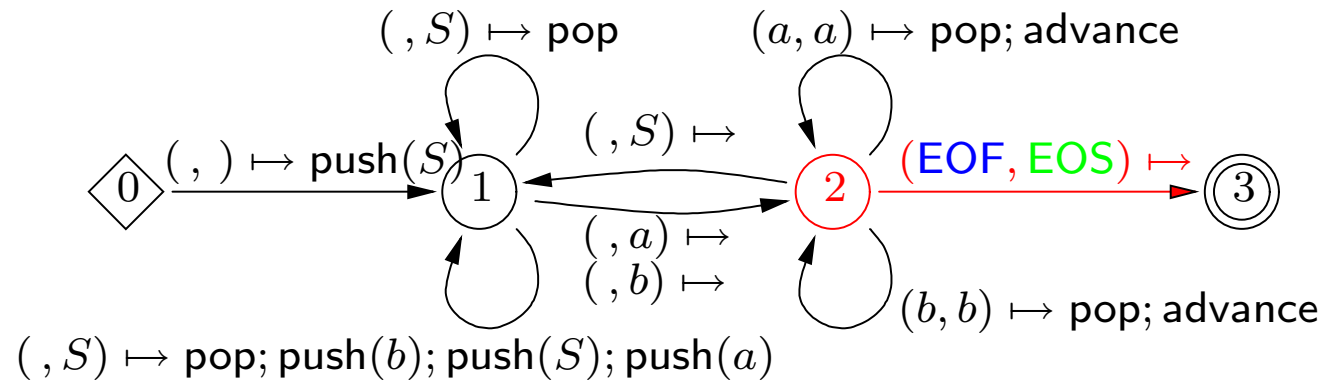
b



b

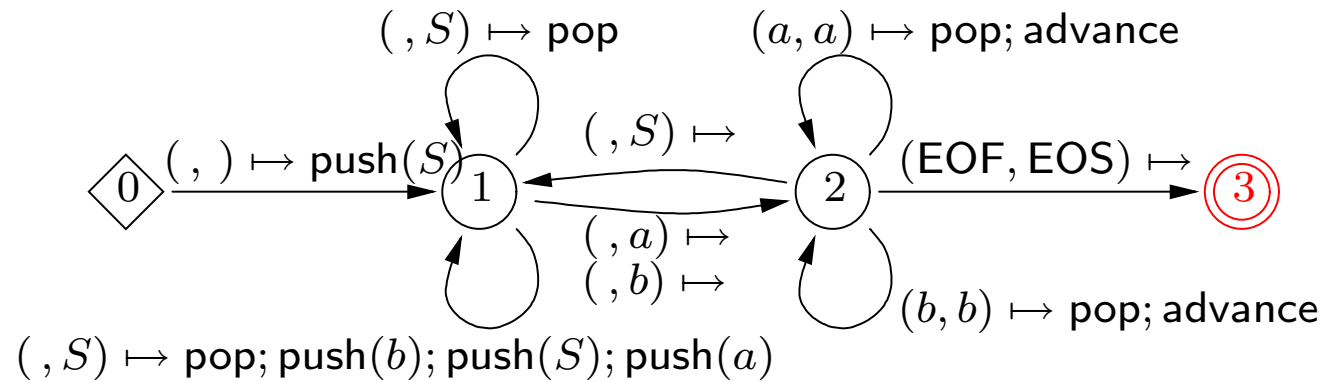
$$\begin{aligned}
 S &\Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \\
 &\Rightarrow \text{aaabbb}
 \end{aligned}$$

PDA processing an input string



$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb$
 $\Rightarrow aaabbb$

PDA processing an input string



$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb$
 $\Rightarrow aaabbb$

PDA processing an input string-II

old state	remaining input	stack contents	actions	new state

PDA processing an input string-II

old state	remaining input	stack contents	actions	new state
0	<i>aaabbb</i> EOF	EOS	push(<i>S</i>)	1

PDA processing an input string-II

old state	remaining input	stack contents	actions	new state
0	<i>aaabbb</i> EOF	EOS	push(<i>S</i>)	1
1	<i>aaabbb</i> EOF	<i>S</i> EOS	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1

PDA processing an input string-II

old state	remaining input	stack contents	actions	new state
0	<i>aaabbb</i> EOF	EOS	push(<i>S</i>)	1
1	<i>aaabbb</i> EOF	<i>S</i> EOS	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>aaabbb</i> EOF	<i>aSb</i> EOS		2

PDA processing an input string-II

old state	remaining input	stack contents	actions	new state
0	<i>aaabbb</i> EOF	EOS	push(<i>S</i>)	1
1	<i>aaabbb</i> EOF	<i>S</i> EOS	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>aaabbb</i> EOF	<i>aSb</i> EOS		2
2	<i>aaabbb</i> EOF	<i>aSb</i> EOS	pop; advance	2

PDA processing an input string-II

old state	remaining input	stack contents	actions	new state
0	<i>aaabbb</i> EOF	EOS	push(<i>S</i>)	1
1	<i>aaabbb</i> EOF	<i>S</i> EOS	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>aaabbb</i> EOF	<i>aSb</i> EOS		2
2	<i>aaabbb</i> EOF	<i>aSb</i> EOS	pop; advance	2
2	<i>aabbb</i> EOF	<i>Sb</i> EOS		1

PDA processing an input string-II

old state	remaining input	stack contents	actions	new state
0	<i>aaabbb</i> EOF	EOS	push(<i>S</i>)	1
1	<i>aaabbb</i> EOF	<i>S</i> EOS	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>aaabbb</i> EOF	<i>aSb</i> EOS		2
2	<i>aaabbb</i> EOF	<i>aSb</i> EOS	pop; advance	2
2	<i>aabbb</i> EOF	<i>Sb</i> EOS		1
1	<i>aabbb</i> EOF	<i>Sb</i> EOS	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1

PDA processing an input string-II

old state	remaining input	stack contents	actions	new state
0	<i>aaabbb</i> EOF	EOS	push(<i>S</i>)	1
1	<i>aaabbb</i> EOF	<i>S</i> EOS	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>aaabbb</i> EOF	<i>aSb</i> EOS		2
2	<i>aaabbb</i> EOF	<i>aSb</i> EOS	pop; advance	2
2	<i>aabbb</i> EOF	<i>Sb</i> EOS		1
1	<i>aabbb</i> EOF	<i>Sb</i> EOS	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>aabbb</i> EOF	<i>aSbb</i> EOS		2

PDA processing an input string-II

old state	remaining input	stack contents	actions	new state
0	<i>aaabbb</i> EOF	EOS	push(<i>S</i>)	1
1	<i>aaabbb</i> EOF	<i>S</i> EOS	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>aaabbb</i> EOF	<i>aSb</i> EOS		2
2	<i>aaabbb</i> EOF	<i>aSb</i> EOS	pop; advance	2
2	<i>aabbb</i> EOF	<i>Sb</i> EOS		1
1	<i>aabbb</i> EOF	<i>Sb</i> EOS	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>aabbb</i> EOF	<i>aSbb</i> EOS		2
2	<i>aabbb</i> EOF	<i>aSbb</i> EOS	pop; advance	2

PDA processing an input string-II

old state	remaining input	stack contents	actions	new state
0	<i>aaabbb</i> EOF	EOS	push(<i>S</i>)	1
1	<i>aaabbb</i> EOF	<i>SEOS</i>	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>aaabbb</i> EOF	<i>aSbEOS</i>		2
2	<i>aaabbb</i> EOF	<i>aSbEOS</i>	pop; advance	2
2	<i>aabbb</i> EOF	<i>SbEOS</i>		1
1	<i>aabbb</i> EOF	<i>SbEOS</i>	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>aabbb</i> EOF	<i>aSbbEOS</i>		2
2	<i>aabbb</i> EOF	<i>aSbbEOS</i>	pop; advance	2
2	<i>abbb</i> EOF	<i>SbbEOS</i>		1

PDA processing an input string-II

old state	remaining input	stack contents	actions	new state
0	<i>aaabbb</i> EOF	EOS	push(<i>S</i>)	1
1	<i>aaabbb</i> EOF	<i>SEOS</i>	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>aaabbb</i> EOF	<i>aSbEOS</i>		2
2	<i>aaabbb</i> EOF	<i>aSbEOS</i>	pop; advance	2
2	<i>aabbb</i> EOF	<i>SbEOS</i>		1
1	<i>aabbb</i> EOF	<i>SbEOS</i>	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>aabbb</i> EOF	<i>aSbbEOS</i>		2
2	<i>aabbb</i> EOF	<i>aSbbEOS</i>	pop; advance	2
2	<i>abbb</i> EOF	<i>SbbEOS</i>		1
1	<i>abbb</i> EOF	<i>SbbEOS</i>	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1

PDA processing an input string-II

old state	remaining input	stack contents	actions	new state
0	<i>aaabbb</i> EOF	EOS	push(<i>S</i>)	1
1	<i>aaabbb</i> EOF	<i>S</i> EOS	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>aaabbb</i> EOF	<i>aSb</i> EOS		2
2	<i>aaabbb</i> EOF	<i>aSb</i> EOS	pop; advance	2
2	<i>aabbb</i> EOF	<i>Sb</i> EOS		1
1	<i>aabbb</i> EOF	<i>Sb</i> EOS	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>aabbb</i> EOF	<i>aSbb</i> EOS		2
2	<i>aabbb</i> EOF	<i>aSbb</i> EOS	pop; advance	2
2	<i>abb</i> EOF	<i>Sbb</i> EOS		1
1	<i>abb</i> EOF	<i>Sbb</i> EOS	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>abb</i> EOF	<i>aSbbb</i> EOS		2

PDA processing an input string-II

old state	remaining input	stack contents	actions	new state
0	<i>aaabbb</i> EOF	EOS	push(<i>S</i>)	1
1	<i>aaabbb</i> EOF	<i>SEOS</i>	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>aaabbb</i> EOF	<i>aSbEOS</i>		2
2	<i>aaabbb</i> EOF	<i>aSbEOS</i>	pop; advance	2
2	<i>aabbb</i> EOF	<i>SbEOS</i>		1
1	<i>aabbb</i> EOF	<i>SbEOS</i>	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>aabbb</i> EOF	<i>aSbbEOS</i>		2
2	<i>aabbb</i> EOF	<i>aSbbEOS</i>	pop; advance	2
2	<i>abbb</i> EOF	<i>SbbEOS</i>		1
1	<i>abbb</i> EOF	<i>SbbEOS</i>	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>abbb</i> EOF	<i>aSbbbEOS</i>		2
2	<i>abbb</i> EOF	<i>aSbbbEOS</i>	pop; advance	2

PDA processing an input string-II

old state	remaining input	stack contents	actions	new state
0	<i>aaabbb</i> EOF	EOS	push(<i>S</i>)	1
1	<i>aaabbb</i> EOF	<i>SEOS</i>	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>aaabbb</i> EOF	<i>aSbEOS</i>		2
2	<i>aaabbb</i> EOF	<i>aSbEOS</i>	pop; advance	2
2	<i>aabbb</i> EOF	<i>SbEOS</i>		1
1	<i>aabbb</i> EOF	<i>SbEOS</i>	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>aabbb</i> EOF	<i>aSbbEOS</i>		2
2	<i>aabbb</i> EOF	<i>aSbbEOS</i>	pop; advance	2
2	<i>abb</i> EOF	<i>SbbEOS</i>		1
1	<i>abb</i> EOF	<i>SbbEOS</i>	pop; push(<i>b</i>); push(<i>S</i>); push(<i>a</i>)	1
1	<i>abb</i> EOF	<i>aSbbbEOS</i>		2
2	<i>abb</i> EOF	<i>aSbbbEOS</i>	pop; advance	2
2	<i>bbb</i> EOF	<i>SbbbEOS</i>		1

PDA processing an input string-III

old state	remaining input	stack contents	actions	new state
⋮	⋮	⋮	⋮	⋮
2	<i>abbb</i> EOF	<i>aSbbb</i> EOS	pop; advance	2
2	<i>bbb</i> EOF	<i>Sbbb</i> EOS		1

PDA processing an input string-III

old state	remaining input	stack contents	actions	new state
⋮	⋮	⋮	⋮	⋮
2	<i>abb</i> EOF	<i>aSbbb</i> EOS	pop; advance	2
2	<i>bb</i> EOF	<i>Sbbb</i> EOS		1
1	<i>bb</i> EOF	<i>Sbbb</i> EOS	pop	1

PDA processing an input string-III

old state	remaining input	stack contents	actions	new state
⋮	⋮	⋮	⋮	⋮
2	<i>abb</i> EOF	<i>aSbbb</i> EOS	pop; advance	2
2	<i>bb</i> EOF	<i>Sbbb</i> EOS		1
1	<i>bb</i> EOF	<i>Sbbb</i> EOS	pop	1
1	<i>bb</i> EOF	<i>bbb</i> EOS		2

PDA processing an input string-III

old state	remaining input	stack contents	actions	new state
⋮	⋮	⋮	⋮	⋮
2	<i>abbb</i> EOF	<i>aSbbb</i> EOS	pop; advance	2
2	<i>bbb</i> EOF	<i>Sbbb</i> EOS		1
1	<i>bbb</i> EOF	<i>Sbbb</i> EOS	pop	1
1	<i>bbb</i> EOF	<i>bbb</i> EOS		2
2	<i>bbb</i> EOF	<i>bbb</i> EOS	pop; advance	2

PDA processing an input string-III

old state	remaining input	stack contents	actions	new state
⋮	⋮	⋮	⋮	⋮
2	<i>abbb</i> EOF	<i>aSbbb</i> EOS	pop; advance	2
2	<i>bbb</i> EOF	<i>Sbbb</i> EOS		1
1	<i>bbb</i> EOF	<i>Sbbb</i> EOS	pop	1
1	<i>bbb</i> EOF	<i>bbb</i> EOS		2
2	<i>bbb</i> EOF	<i>bbb</i> EOS	pop; advance	2
2	<i>bb</i> EOF	<i>bb</i> EOS	pop; advance	2

PDA processing an input string-III

old state	remaining input	stack contents	actions	new state
⋮	⋮	⋮	⋮	⋮
2	<i>abbb</i> EOF	<i>aSbbb</i> EOS	pop; advance	2
2	<i>bbb</i> EOF	<i>Sbbb</i> EOS		1
1	<i>bbb</i> EOF	<i>Sbbb</i> EOS	pop	1
1	<i>bbb</i> EOF	<i>bbb</i> EOS		2
2	<i>bbb</i> EOF	<i>bbb</i> EOS	pop; advance	2
2	<i>bb</i> EOF	<i>bb</i> EOS	pop; advance	2
2	<i>b</i> EOF	<i>b</i> EOS	pop; advance	2

PDA processing an input string-III

old state	remaining input	stack contents	actions	new state
⋮	⋮	⋮	⋮	⋮
2	<i>abbb</i> EOF	<i>aSbbb</i> EOS	pop; advance	2
2	<i>bbb</i> EOF	<i>Sbbb</i> EOS		1
1	<i>bbb</i> EOF	<i>Sbbb</i> EOS	pop	1
1	<i>bbb</i> EOF	<i>bbb</i> EOS		2
2	<i>bbb</i> EOF	<i>bbb</i> EOS	pop; advance	2
2	<i>bb</i> EOF	<i>bb</i> EOS	pop; advance	2
2	<i>b</i> EOF	<i>b</i> EOS	pop; advance	2
2	EOF	EOS		3