Pushdown automata (PAs)

Implementation of Computational Models (TC2037)

Edgar Covantes Osuna, PhD
edgar.covantes@tec.mx

Pushdown Automata?

Informal description of a PA

A Pushdown Automata (PA) is an Automata with a *stack*.

The automata has a stack, it is used as extra **memory** to make more complicated operations.

The Pushdown Automata are **equivalent** to the Context-free Grammars—they serve to represent context-free languages.

Pushdown Automata?

Informal description of a PA

A Pushdown Automata (PA) is an Automata with a *stack*.

The automata has a stack, it is used as extra **memory** to make more complicated operations.

The Pushdown Automata are **equivalent** to the Context-free Grammars—they serve to represent context-free languages.

Pushdown Automata?

Informal description of a PA

A Pushdown Automata (PA) is an Automata with a *stack*.

The automata has a stack, it is used as extra **memory** to make more complicated operations.

The Pushdown Automata are **equivalent** to the Context-free Grammars—they serve to represent context-free languages.

What is a Pushdown Automata?

Informal description of a PA

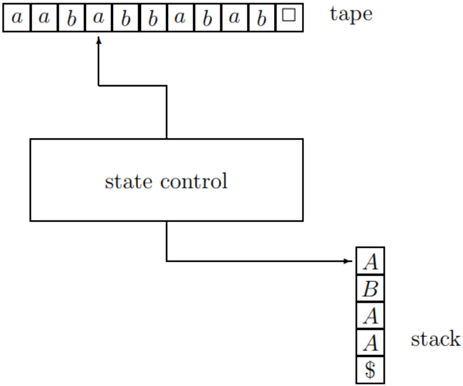


Figure 3.1: *A pushdown automaton.*

Image from Maheshwari and Smid's *Theory of Computation*, 2017

What is a Pushdown Automata?

Informal description of a PA

A PA has several elements:

1) A tape divided in **cells**

- Each cell contains a **symbol** of the input word that belongs to some **alphabet** Σ (also called **tape alphabet**).
- At the end of the tape, we have a **new symbol** to represent the end of the word: $\Box$. This new symbol does not belong to the alphabet of the word.

2) A tape head that can read the value of each cell according to its position, and it can perform two actions $\sigma = \{N, R\}$: N if it does not move and R if it moves to the right.

What is a Pushdown Automata?

Informal description of a PA

A PA has several elements:

1) A tape divided in **cells**

- Each cell contains a **symbol** of the input word that belongs to some **alphabet** Σ (also called **tape alphabet**).
- At the end of the tape, we have a **new symbol** to represent the end of the word: $\square$. This new symbol does not belong to the alphabet of the word.

2) A tape head that can read the value of each cell according to its position, and it can perform two actions $\sigma = \{N, R\}$: N if it does not move and R if it moves to the right.

What is a Pushdown Automata?

Informal description of a PA

A PA has several elements:

1) A tape divided in **cells**

- Each cell contains a **symbol** of the input word that belongs to some **alphabet** Σ (also called **tape alphabet**).
- At the end of the tape, we have a **new symbol** to represent the end of the word: $\square$. This new symbol does not belong to the alphabet of the word.

2) A tape head that can read the value of each cell according to its position, and it can perform two actions $\sigma = \{N, R\}$: N if it does not move and R if it moves to the right.

What is a Pushdown Automata?

Informal description of a PA

A PA has several elements:

1) A tape divided in **cells**

- Each cell contains a **symbol** of the input word that belongs to some **alphabet** Σ (also called **tape alphabet**).
- At the end of the tape, we have a **new symbol** to represent the end of the word: $\square$. This new symbol does not belong to the alphabet of the word.

2) A tape head that can read the value of each cell according to its position, and it can perform two actions $\sigma = \{N, R\}$: N if it does not move and R if it moves to the right.

What is a Pushdown Automata?

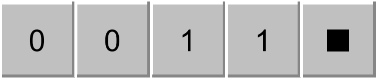
Informal description of a PA

A PA has several elements:

1) A tape divided in **cells**

- Each cell contains a **symbol** of the input word that belongs to some **alphabet** Σ (also called **tape alphabet**).
- At the end of the tape, we have a **new symbol** to represent the end of the word: $\Box$. This new symbol does not belong to the alphabet of the word.

2) A tape head that can read the value of each cell according to its position, and it can perform two actions $\sigma = \{N, R\}$: N if it does not move and R if it moves to the right.



What is a Pushdown Automata?

Informal description of a PA

3) A stack that can store symbols.

- The symbols that can be stored in the stack belongs to **another alphabet**: Γ.
- One of these symbols is \$, which is **inside of the tape alphabet**.

4) A stack head that reads **the last symbol** of the same.

- The stack head can *stack* (`push`) more symbols, or it can *remove* the symbol from the top (`pop`).

5) A set of states, connected by transitions that depend of the symbols read by **both heads**.

What is a Pushdown Automata?

Informal description of a PA

3) A stack that can store symbols.

- The symbols that can be stored in the stack belongs to **another alphabet**: Γ.
- One of these symbols is \$, which is **inside of the tape alphabet**.

4) A stack head that reads **the last symbol** of the same.

- The stack head can *stack* (`push`) more symbols, or it can *remove* the symbol from the top (`pop`).

5) A set of states, connected by transitions that depend of the symbols read by **both heads**.

What is a Pushdown Automata?

Informal description of a PA

3) A stack that can store symbols.

- The symbols that can be stored in the stack belongs to **another alphabet**: Γ.
- One of these symbols is $\$$, which is **inside of the tape alphabet**.

4) A stack head that reads **the last symbol** of the same.

- The stack head can *stack* (`push`) more symbols, or it can *remove* the symbol from the top (`pop`).

5) A set of states, connected by transitions that depend of the symbols read by **both heads**.

What is a Pushdown Automata?

Informal description of a PA

3) A stack that can store symbols.

- The symbols that can be stored in the stack belongs to **another alphabet**: Γ.
- One of these symbols is $\$$, which is **inside of the tape alphabet**.

4) A stack head that reads **the last symbol** of the same.

- The stack head can *stack* (`push`) more symbols, or it can *remove* the symbol from the top (`pop`).

5) A set of states, connected by transitions that depend of the symbols read by **both heads**.

What is a Pushdown Automata?

Informal description of a PA

3) A stack that can store symbols.

- The symbols that can be stored in the stack belongs to **another alphabet**: Γ.
- One of these symbols is $\$$, which is **inside of the tape alphabet**.

4) A stack head that reads **the last symbol** of the same.

- The stack head can *stack* (`push`) more symbols, or it can *remove* the symbol from the top (`pop`).

5) A set of states, connected by transitions that depend of the symbols read by **both heads**.

What is a Pushdown Automata?

Informal description of a PA

3) A stack that can store symbols.

- The symbols that can be stored in the stack belongs to **another alphabet**: Γ.
- One of these symbols is $\$$, which is **inside of the tape alphabet**.

4) A stack head that reads **the last symbol** of the same.

- The stack head can *stack* (`push`) more symbols, or it can *remove* the symbol from the top (`pop`).

5) A set of states, connected by transitions that depend of the symbols read by **both heads**.

A stack as a data structure

Informal description of a PA

A stack (or in Spanish *pila*) works by the principle LIFO, that means *Last In, First Out*—the last element to enter is the first to be removed.

Example

Let A be a stack with the values $A = \langle 1, 2, 3 \rangle$ and $F = \{\mathrm{pop}, \mathrm{push}\}$ the set of functions *in-place* applicable to stacks.

If we apply the function pop on A, then we obtain the value 3, and the stack will change to $A = \langle 1, 2 \rangle$.

If after that we introduce one more value—let us say 4—to the stack (using the function push), then the stack will turn to $A = \langle 1, 2, 4 \rangle$.

What would happen if we apply pop again?

A stack as a data structure

Informal description of a PA

A stack (or in Spanish *pila*) works by the principle LIFO, that means *Last In, First Out*—the last element to enter is the first to be removed.

Example

Let A be a stack with the values $A = \langle 1, 2, 3 \rangle$ and $F = \{\mathrm{pop}, \mathrm{push}\}$ the set of functions *in-place* applicable to stacks.

If we apply the function pop on A, then we obtain the value 3, and the stack will change to $A = \langle 1, 2 \rangle$.

If after that we introduce one more value—let us say 4—to the stack (using the function push), then the stack will turn to $A = \langle 1, 2, 4 \rangle$.

What would happen if we apply pop again?

A stack as a data structure

Informal description of a PA

A stack (or in Spanish *pila*) works by the principle LIFO, that means *Last In, First Out*—the last element to enter is the first to be removed.

Example

Let A be a stack with the values $A = \langle 1, 2, 3 \rangle$ and $F = \{\texttt{pop}, \texttt{push}\}$ the set of functions *in-place* applicable to stacks.

If we apply the function pop on A, then we obtain the value 3, and the stack will change to $A = \langle 1, 2 \rangle$.

If after that we introduce one more value—let us say 4—to the stack (using the function $\texttt{push}$), then the stack will turn to $A = \langle 1, 2, 4 \rangle$.

What would happen if we apply pop again?

A stack as a data structure

Informal description of a PA

A stack (or in Spanish *pila*) works by the principle LIFO, that means *Last In, First Out*—the last element to enter is the first to be removed.

Example

Let A be a stack with the values $A = \langle 1, 2, 3 \rangle$ and $F = \{\texttt{pop}, \texttt{push}\}$ the set of functions *in-place* applicable to stacks.

If we apply the function pop on A, then we obtain the value 3, and the stack will change to $A = \langle 1, 2 \rangle$.

If after that we introduce one more value—let us say 4—to the stack (using the function push), then the stack will turn to $A = \langle 1, 2, 4 \rangle$.

What would happen if we apply pop again?

A stack as a data structure

Informal description of a PA

A stack (or in Spanish *pila*) works by the principle LIFO, that means *Last In, First Out*—the last element to enter is the first to be removed.

Example

Let A be a stack with the values $A = \langle 1, 2, 3 \rangle$ and $F = \{\texttt{pop}, \texttt{push}\}$ the set of functions *in-place* applicable to stacks.

If we apply the function pop on A, then we obtain the value 3, and the stack will change to $A = \langle 1, 2 \rangle$.

If after that we introduce one more value—let us say 4—to the stack (using the function push), then the stack will turn to $A = \langle 1, 2, 4 \rangle$.

What would happen if we apply pop again?

Formal definitions

Formal definition and design of PAs

There are **many** ways to express PAs and its elements:

- Brena (2003) uses transitions expressed in terms of the input and `pop` and `push` of the stack, belonging to a *relation of transition*, with final states and new symbols.
- Maheshwari and Smid (2017) use a **full** transition function in terms of the state, the input, tape movement ($\sigma = \{N, R\}$) and the `replace` function of the stack, without final states.
- Tinelli (2016) uses transitions expressed in terms of the input and the `replace` function of the stack, with final states.

In our case we will use **a combination of the three**: using the notations of Maheshwari and Smid for the design, we improve the notations using Brena with the symbols of Tinelli because we're cool like that.

Formal definitions

Formal definition and design of PAs

There are **many** ways to express PAs and its elements:

- Brena (2003) uses transitions expressed in terms of the input and `pop` and `push` of the stack, belonging to a *relation of transition*, with final states and new symbols.
- Maheshwari and Smid (2017) use a **full** transition function in terms of the state, the input, tape movement ($\sigma = \{N, R\}$) and the `replace` function of the stack, without final states.
- Tinelli (2016) uses transitions expressed in terms of the input and the `replace` function of the stack, with final states.

In our case we will use **a combination of the three**: using the notations of Maheshwari and Smid for the design, we improve the notations using Brena with the symbols of Tinelli because we're cool like that.

Formal definitions

Formal definition and design of PAs

There are **many** ways to express PAs and its elements:

- Brena (2003) uses transitions expressed in terms of the input and `pop` and `push` of the stack, belonging to a *relation of transition*, with final states and new symbols.

- Maheshwari and Smid (2017) use a **full** transition function in terms of the state, the input, tape movement ($\sigma = \{N, R\}$) and the `replace` function of the stack, without final states.

- Tinelli (2016) uses transitions expressed in terms of the input and the `replace` function of the stack, with final states.

In our case we will use **a combination of the three**: using the notations of Maheshwari and Smid for the design, we improve the notations using Brena with the symbols of Tinelli because we're cool like that.

Formal definitions

Formal definition and design of PAs

There are **many** ways to express PAs and its elements:

- Brena (2003) uses transitions expressed in terms of the input and `pop` and `push` of the stack, belonging to a *relation of transition*, with final states and new symbols.
- Maheshwari and Smid (2017) use a **full** transition function in terms of the state, the input, tape movement ($\sigma = \{N, R\}$) and the `replace` function of the stack, without final states.
- Tinelli (2016) uses transitions expressed in terms of the input and the `replace` function of the stack, with final states.

In our case we will use **a combination of the three**: using the notations of Maheshwari and Smid for the design, we improve the notations using Brena with the symbols of Tinelli because we're cool like that.

Formal definitions

Formal definition and design of PAs

There are **many** ways to express PAs and its elements:

- Brena (2003) uses transitions expressed in terms of the input and `pop` and `push` of the stack, belonging to a *relation of transition*, with final states and new symbols.
- Maheshwari and Smid (2017) use a **full** transition function in terms of the state, the input, tape movement ($\sigma = \{N, R\}$) and the `replace` function of the stack, without final states.
- Tinelli (2016) uses transitions expressed in terms of the input and the `replace` function of the stack, with final states.

In our case we will use **a combination of the three**: using the notations of Maheshwari and Smid for the design, we improve the notations using Brena with the symbols of Tinelli because we're cool like that.

Formal definitions

Formal definition and design of PAs

There are **many** ways to express PAs and its elements:

- Brena (2003) uses transitions expressed in terms of the input and `pop` and `push` of the stack, belonging to a *relation of transition*, with final states and new symbols.
- Maheshwari and Smid (2017) use a **full** transition function in terms of the state, the input, tape movement ($\sigma = \{N, R\}$) and the `replace` function of the stack, without final states.
- Tinelli (2016) uses transitions expressed in terms of the input and the `replace` function of the stack, with final states.

In our case we will use **a combination of the three**: using the notations of Maheshwari and Smid for the design, we improve the notations using Brena with the symbols of Tinelli because we're cool like that.

Formal Definition

Formal definition and design of PAs

Definition of a Pushdown Automata

A pushdown automata M is a 6-tuple with the form $M = (Q, \Sigma, \Gamma, \delta, q, F)$ where:

- Q is a finite set of **states**,
- Σ is the **tape alphabet** (with no $\Box$),
- Γ is the **stack alphabet** (including $\$$),
- $q \in Q$ is the **initial state**,
- $F \subseteq Q$ is a finite set of **final states** and
- δ is the transition function.

Formal Definition

Definition of a Pushdown Automata

A pushdown automata M is a 6-tuple with the form $M = (Q, \Sigma, \Gamma, \delta, q, F)$ where:

- Q is a finite set of **states**,
- Σ is the **tape alphabet** (with no $\Box$),
- Γ is the **stack alphabet** (including $\$$),
- $q \in Q$ is the **initial state**,
- $F \subseteq Q$ is a finite set of **final states** and
- δ is the transition function.

Formal Definition

Definition of a Pushdown Automata

A pushdown automata M is a 6-tuple with the form $M = (Q, \Sigma, \Gamma, \delta, q, F)$ where:

- Q is a finite set of **states**,
- Σ is the **tape alphabet** (with no $\square$),
- Γ is the **stack alphabet** (including $\$$),
- $q \in Q$ is the **initial state**,
- $F \subseteq Q$ is a finite set of **final states** and
- δ is the transition function.

Formal Definition

Formal definition and design of PAs

Definition of a Pushdown Automata

A pushdown automata M is a 6-tuple with the form $M = (Q, \Sigma, \Gamma, \delta, q, F)$ where:

- Q is a finite set of **states**,
- Σ is the **tape alphabet** (with no $\Box$),
- Γ is the **stack alphabet** (including $\$$),
- $q \in Q$ is the **initial state**,
- $F \subseteq Q$ is a finite set of **final states** and
- δ is the transition function.

Formal Definition

Formal definition and design of PAs

Definition of a Pushdown Automata

A pushdown automata M is a 6-tuple with the form $M = (Q, \Sigma, \Gamma, \delta, q, F)$ where:

- Q is a finite set of **states**,
- Σ is the **tape alphabet** (with no $\Box$),
- Γ is the **stack alphabet** (including $\$$),
- $q \in Q$ is the **initial state**,
- $F \subseteq Q$ is a finite set of **final states** and
- δ is the transition function.

Formal Definition

Formal definition and design of PAs

Definition of a Pushdown Automata

A pushdown automata M is a 6-tuple with the form $M = (Q, \Sigma, \Gamma, \delta, q, F)$ where:

- Q is a finite set of **states**,
- Σ is the **tape alphabet** (with no $\Box$),
- Γ is the **stack alphabet** (including $\$$),
- $q \in Q$ is the **initial state**,
- $F \subseteq Q$ is a finite set of **final states** and
- δ is the transition function.

Formal Definition

Formal definition and design of PAs

Definition of a Pushdown Automata

A pushdown automata M is a 6-tuple with the form $M = (Q, \Sigma, \Gamma, \delta, q, F)$ where:

- Q is a finite set of **states**,
- Σ is the **tape alphabet** (with no $\square$),
- Γ is the **stack alphabet** (including $\$$),
- $q \in Q$ is the **initial state**,
- $F \subseteq Q$ is a finite set of **final states** and
- δ is the transition function.

Formal Definition

Formal definition and design of PAs

The transition function δ is a function with the form:

$$\delta : Q \times (\Sigma \cup \{\Box\}) \times \Gamma \to Q \times \{N, R\} \times \Gamma^*$$

Example

We can write $q_0 1 S \to q_1 RSS$ that means:

- Being in the state q_0,
- upon receiving a 1,
- and if the top of the stack has S

then the PA

- change to the state q_1,
- moves the tape to the right (Right) and
- replace the top of the stack with SS.

Formal Definition

Formal definition and design of PAs

The transition function δ is a function with the form:

$$\delta : Q \times (\Sigma \cup \{\square\}) \times \Gamma \rightarrow Q \times \{N, R\} \times \Gamma^*$$

Example

We can write $q_0 1 S \rightarrow q_1 R S S$ that means:

- Being in the state q_0,
- upon receiving a 1,
- and if the top of the stack has S

then the PA

- change to the state q_1,
- moves the tape to the right (Right) and
- replace the top of the stack with SS.

Formal Definition

Formal definition and design of PAs

The transition function δ is a function with the form:

$$\delta : Q \times (\Sigma \cup \{\square\}) \times \Gamma \to Q \times \{N, R\} \times \Gamma^*$$

Example

We can write $q_0 1 S \to q_1 RSS$ that means:

- Being in the state q_0,
- upon receiving a 1,
- and if the top of the stack has S

then the PA

- change to the state q_1,
- moves the tape to the right (Right) and
- replace the top of the stack with SS.

Formal Definition

Formal definition and design of PAs

The transition function δ is a function with the form:

$$\delta : Q \times (\Sigma \cup \{\Box\}) \times \Gamma \to Q \times \{N, R\} \times \Gamma^*$$

Example

We can write $q_0 1 S \to q_1 RSS$ that means:

- Being in the state q_0,
- upon receiving a 1,
- and if the top of the stack has S

then the PA

- change to the state q_1,
- moves the tape to the right (Right) and
- replace the top of the stack with SS.

Formal Definition

Formal definition and design of PAs

The transition function δ is a function with the form:

$$\delta : Q \times (\Sigma \cup \{\Box\}) \times \Gamma \to Q \times \{N, R\} \times \Gamma^*$$

Example

We can write $q_0 1 S \to q_1 RSS$ that means:

- Being in the state q_0,
- upon receiving a 1,
- and if the top of the stack has S

then the PA

- change to the state q_1,
- moves the tape to the right (Right) and
- replace the top of the stack with SS.

Formal Definition

Formal definition and design of PAs

The transition function δ is a function with the form:

$$\delta : Q \times (\Sigma \cup \{\square\}) \times \Gamma \rightarrow Q \times \{N, R\} \times \Gamma^*$$

Example

We can write $q_0 1 S \rightarrow q_1 RSS$ that means:

- Being in the state q_0,
- upon receiving a 1,
- and if the top of the stack has S

then the PA

- change to the state q_1,
- moves the tape to the right (Right) and
- replace the top of the stack with SS.

Formal Definition

Formal definition and design of PAs

The transition function δ is a function with the form:

$$\delta : Q \times (\Sigma \cup \{\Box\}) \times \Gamma \to Q \times \{N, R\} \times \Gamma^*$$

Example

We can write $q_0 1 S \to q_1 R S S$ that means:

- Being in the state q_0,
- upon receiving a 1,
- and if the top of the stack has S

then the PA

- change to the state q_1,
- moves the tape to the right (Right) and
- replace the top of the stack with SS.

Formal Definition

Formal definition and design of PAs

The transition function δ is a function with the form:

$$\delta : Q \times (\Sigma \cup \{\Box\}) \times \Gamma \to Q \times \{N, R\} \times \Gamma^*$$

Example

We can write $q_0 1 S \to q_1 R S S$ that means:

- Being in the state q_0,
- upon receiving a 1,
- and if the top of the stack has S

then the PA

- change to the state q_1,
- moves the tape to the right (Right) and
- replace the top of the stack with SS.

Additional considerations

Formal definition and design of PAs

Initial configuration

- The PA starts in the state q.
- The tape head starts in the initial symbol of word w.
- The stack starts with only one symbol, $\$$.

Computation and termination

The PA makes a series of computation steps and *finishes* at the moment the stack is empty. If the stack is not empty, then the program does not end (infinite *loop*).

Acceptance

The PA accept the words w if the following conditions are met:

1. the automaton ends, and
2. at the time of finishing, (i.e. the stack is empty) the tape head is with the symbol $\square$.

Additional considerations

Formal definition and design of PAs

Initial configuration

- The PA starts in the state q.
- The tape head starts in the initial symbol of word w.
- The stack starts with only one symbol, $\$$.

Computation and termination

The PA makes a series of computation steps and *finishes* at the moment the stack is empty. If the stack is not empty, then the program does not end (infinite *loop*).

Acceptance

The PA accept the words w if the following conditions are met:

1. the automaton ends, and
2. at the time of finishing, (i.e. the stack is empty) the tape head is with the symbol $\square$.

Additional considerations

Formal definition and design of PAs

Initial configuration

- The PA starts in the state q.
- The tape head starts in the initial symbol of word w.
- The stack starts with only one symbol, $\$$.

Computation and termination

The PA makes a series of computation steps and *finishes* at the moment the stack is empty. If the stack is not empty, then the program does not end (infinite *loop*).

Acceptance

The PA accept the words w if the following conditions are met:

1. the automaton ends, and
2. at the time of finishing, (i.e. the stack is empty) the tape head is with the symbol $\square$.

Additional considerations

Formal definition and design of PAs

Initial configuration

- The PA starts in the state q.
- The tape head starts in the initial symbol of word w.
- The stack starts with only one symbol, $\$$.

Computation and termination

The PA makes a series of computation steps and *finishes* at the moment the stack is empty. If the stack is not empty, then the program does not end (infinite *loop*).

Acceptance

The PA accept the words w if the following conditions are met:

1. the automaton ends, and
2. at the time of finishing, (i.e. the stack is empty) the tape head is with the symbol $\square$.

Additional considerations

Formal definition and design of PAs

Initial configuration

- The PA starts in the state q.
- The tape head starts in the initial symbol of word w.
- The stack starts with only one symbol, $\$$.

Computation and termination

The PA makes a series of computation steps and *finishes* at the moment the stack is empty. If the stack is not empty, then the program does not end (infinite *loop*).

Acceptance

The PA accept the words w if the following conditions are met:

1. the automaton ends, and
2. at the time of finishing, (i.e. the stack is empty) the tape head is with the symbol $\square$.

Additional considerations

Formal definition and design of PAs

Initial configuration

- The PA starts in the state q.
- The tape head starts in the initial symbol of word w.
- The stack starts with only one symbol, $\$$.

Computation and termination

The PA makes a series of computation steps and *finishes* at the moment the stack is empty. If the stack is not empty, then the program does not end (infinite *loop*).

Acceptance

The PA accept the words w if the following conditions are met:

1. the automaton ends, and
2. at the time of finishing, (i.e. the stack is empty) the tape head is with the symbol $\Box$.

Example: *Matching Parentheses*

Formal definition and design of PAs

Examples of accepted words: $(), ((())), ()(), \dots$

Structure:

- Before finishing reading the whole word, the amount of "(" must be greater or equal than ")", and
- When you finished reading the whole word, the number of "(" must be equal to the number of ")".

Let us use a to represent "(" and b to represent ")" in the tape.

For each a read, we introduce as S in the stack. For each b read, we take out an S. How many states do we need? How many are finals?
Is a **complicated** process, so it is first suggested to define the PA formally (e.g. the complete transition function, and part by part), and then create the transition diagram and then pass it to a transition diagram with a smaller transition relation: $Q \times \Sigma \cup \{\Box\} \times \Gamma \to Q \times \Gamma^*$.
Why not consider whether or not the tape head moves?

Example: *Matching Parentheses*

Formal definition and design of PAs

Examples of accepted words: $(), ((())), ()(), \ldots$
Structure:

- Before finishing reading the whole word, the amount of "(" must be greater or equal than ")", and
- When you finished reading the whole word, the number of "(" must be equal to the number of ")".

Let us use a to represent "(" and b to represent ")" in the tape.

For each a read, we introduce as S in the stack. For each b read, we take out an S. How many states do we need? How many are finals?
Is a **complicated** process, so it is first suggested to define the PA formally (e.g. the complete transition function, and part by part), and then create the transition diagram and then pass it to a transition diagram with a smaller transition relation: $Q \times \Sigma \cup \{\Box\} \times \Gamma \to Q \times \Gamma^*$.
Why not consider whether or not the tape head moves?

Example: *Matching Parentheses*

Examples of accepted words: $(), ((())), ()(), \ldots$

Structure:

- Before finishing reading the whole word, the amount of "(" must be greater or equal than ")", and
- When you finished reading the whole word, the number of "(" must be equal to the number of ")".

Let us use a to represent "(" and b to represent ")" in the tape.

For each a read, we introduce as S in the stack. For each b read, we take out an S. How many states do we need? How many are finals?
Is a **complicated** process, so it is first suggested to define the PA formally (e.g. the complete transition function, and part by part), and then create the transition diagram and then pass it to a transition diagram with a smaller transition relation: $Q \times \Sigma \cup \{\Box\} \times \Gamma \to Q \times \Gamma^*$.
Why not consider whether or not the tape head moves?

Example: *Matching Parentheses*

Formal definition and design of PAs

Examples of accepted words: $(), ((())), ()(), \ldots$
Structure:

- Before finishing reading the whole word, the amount of "(" must be greater or equal than ")", and
- When you finished reading the whole word, the number of "(" must be equal to the number of ")".

Let us use a to represent "(" and b to represent ")" in the tape.

For each a read, we introduce as S in the stack. For each b read, we take out an S. How many states do we need? How many are finals?
Is a **complicated** process, so it is first suggested to define the PA formally (e.g. the complete transition function, and part by part), and then create the transition diagram and then pass it to a transition diagram with a smaller transition relation: $Q \times \Sigma \cup \{\Box\} \times \Gamma \to Q \times \Gamma^*$.
Why not consider whether or not the tape head moves?

Example: *Matching Parentheses*

Formal definition and design of PAs

Examples of accepted words: $(), ((())), ()(), \ldots$
Structure:

- Before finishing reading the whole word, the amount of "(" must be greater or equal than ")", and
- When you finished reading the whole word, the number of "(" must be equal to the number of ")".

Let us use a to represent "(" and b to represent ")" in the tape.

For each a read, we introduce as S in the stack. For each b read, we take out an S. How many states do we need? How many are finals?
Is a **complicated** process, so it is first suggested to define the PA formally (e.g. the complete transition function, and part by part), and then create the transition diagram and then pass it to a transition diagram with a smaller transition relation: $Q \times \Sigma \cup \{\Box\} \times \Gamma \to Q \times \Gamma^*$.
Why not consider whether or not the tape head moves?

Example: *Matching Parentheses*

Formal definition and design of PAs

Examples of accepted words: $(), ((())), ()(), \ldots$

Structure:

- Before finishing reading the whole word, the amount of "(" must be greater or equal than ")", and
- When you finished reading the whole word, the number of "(" must be equal to the number of ")".

Let us use a to represent "(" and b to represent ")" in the tape.

For each a read, we introduce as S in the stack. For each b read, we take out an S. How many states do we need? How many are finals?

Is a **complicated** process, so it is first suggested to define the PA formally (e.g. the complete transition function, and part by part), and then create the transition diagram and then pass it to a transition diagram with a smaller transition relation: $Q \times \Sigma \cup \{\Box\} \times \Gamma \to Q \times \Gamma^*$.

Why not consider whether or not the tape head moves?

Example: *Matching Parentheses*

Formal definition and design of PAs

Examples of accepted words: $(), ((())), ()(), \ldots$

Structure:

- Before finishing reading the whole word, the amount of "(" must be greater or equal than ")", and
- When you finished reading the whole word, the number of "(" must be equal to the number of ")".

Let us use a to represent "(" and b to represent ")" in the tape.

For each a read, we introduce as S in the stack. For each b read, we take out an S. How many states do we need? How many are finals?

Is a **complicated** process, so it is first suggested to define the PA formally (e.g. the complete transition function, and part by part), and then create the transition diagram and then pass it to a transition diagram with a smaller transition relation: $Q \times \Sigma \cup \{\square\} \times \Gamma \to Q \times \Gamma^*$.

Why not consider whether or not the tape head moves?

Example: *Matching Parentheses*

Formal definition and design of PAs

Examples of accepted words: $(), ((())), ()(), \ldots$

Structure:

- Before finishing reading the whole word, the amount of "(" must be greater or equal than ")", and
- When you finished reading the whole word, the number of "(" must be equal to the number of ")".

Let us use a to represent "(" and b to represent ")" in the tape.

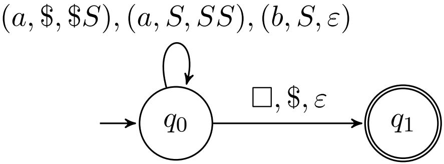
For each a read, we introduce as S in the stack. For each b read, we take out an S. How many states do we need? How many are finals?
Is a **complicated** process, so it is first suggested to define the PA formally (e.g. the complete transition function, and part by part), and then create the transition diagram and then pass it to a transition diagram with a smaller transition relation: $Q \times \Sigma \cup \{\Box\} \times \Gamma \to Q \times \Gamma^*$.
Why not consider whether or not the tape head moves?

Example: *Matching Parentheses*

Formal definition and design of PAs

$M = (Q, \Sigma, \Gamma, \delta, q, F)$:

- $Q = \{q_0, q_1\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{\$, S\}$
- $\delta =$
  - $((q_0, a, \$)(q_0, \$S))$
  - $((q_0, a, S)(q_0, SS))$
  - $((q_0, b, S)(q_0, \varepsilon))$
  - $((q_0, \Box, \$), (q_1, \varepsilon))$
- $q = q_0$
- $F = \{q_1\}$

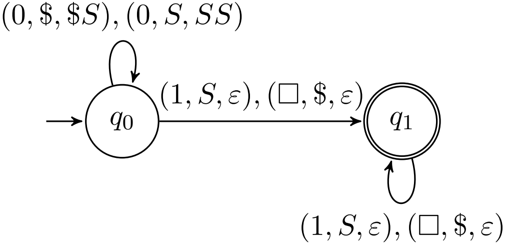


Example: $\{0^n 1^n\}$

Formal definition and design of PAs

$M = (Q, \Sigma, \Gamma, \delta, q, F)$:

- $Q = \{q_0, q_1\}$
- $\Sigma = \{0, 1\}$
- $\Gamma = \{\$, S\}$
- $\delta =$
  - $((q_0, 0, \$)(q_0, \$S))$
  - $((q_0, 0, S)(q_0, SS))$
  - $((q_0, 1, S)(q_1, \varepsilon))$
  - $((q_0, \Box, \$), (q_1, \varepsilon))$
  - $((q_1, 1, S)(q_1, \varepsilon))$
  - $((q_1, \Box, \$)(q_1, \varepsilon))$
- $q = q_0$
- $F = \{q_1\}$



Combination and concatenation

Combination of PAs

In a similar way we combine two NFAs, the idea is to make a **previous** initial state that unites the initial states of the PAs using empty transitions $(\varepsilon, \varepsilon, \varepsilon)$.

Concatenation of PAs

The concatenation works in a similar way as the NFAs, however we must guarantee that the stack is found with *certain conditions* before moving to the next PA. The solution is to use a special symbol before starting with the first PA, and take it out before starting any operation with the sencond PA.

Combination and concatenation

Combination of PAs

In a similar way we combine two NFAs, the idea is to make a **previous** initial state that unites the initial states of the PAs using empty transitions $(\varepsilon, \varepsilon, \varepsilon)$.

Concatenation of PAs

The concatenation works in a similar way as the NFAs, however we must guarantee that the stack is found with *certain conditions* before moving to the next PA. The solution is to use a special symbol before starting with the first PA, and take it out before starting any operation with the sencond PA.

Exercises

Formal definition and design of PAs

Construct (deterministic or nondeterministic) PA that accept the following languages

1. $\{w \in \{0,1\}^* : w \text{ is a palindrome}\}$.
2. $\{w \in \{0,1\}^* : w \text{ contains: }\}$
    - more 1s than 0s.
    - less 1s than 0s.
    - equal number of 1s and 0s.
3. Combine all PAs from exercise 2.
4. $\{0^{2n}1^n : n \geq 1\}$.
5. $\{0^n 1^m 0^n : n \geq 1, m \geq 1\}$.

Exercises

Formal definition and design of PAs

Construct (deterministic or nondeterministic) PA that accept the following languages

1. $\{w \in \{0,1\}^* : w \text{ is a palindrome}\}$.
2. $\{w \in \{0,1\}^* : w \text{ contains: }\}$
   - more 1s than 0s.
   - less 1s than 0s.
   - equal number of 1s and 0s.
3. Combine all PAs from exercise 2.
4. $\{0^{2n}1^n : n \geq 1\}$.
5. $\{0^n1^m0^n : n \geq 1, m \geq 1\}$.

Exercises

Formal definition and design of PAs

Construct (deterministic or nondeterministic) PA that accept the following languages

1. $\{w \in \{0,1\}^* : w \text{ is a palindrome}\}$.
2. $\{w \in \{0,1\}^* : w \text{ contains: }\}$
    - more 1s than 0s.
    - less 1s than 0s.
    - equal number of 1s and 0s.
3. Combine all PAs from exercise 2.
4. $\{0^{2n}1^n : n \geq 1\}$.
5. $\{0^n1^m0^n : n \geq 1, m \geq 1\}$.

Exercises

Formal definition and design of PAs

Construct (deterministic or nondeterministic) PA that accept the following languages

1. $\{w \in \{0,1\}^* : w \text{ is a palindrome}\}$.
2. $\{w \in \{0,1\}^* : w \text{ contains: }\}$
   - more 1s than 0s.
   - less 1s than 0s.
   - equal number of 1s and 0s.
3. Combine all PAs from exercise 2.
4. $\{0^{2n}1^n : n \geq 1\}$.
5. $\{0^n 1^m 0^n : n \geq 1, m \geq 1\}$.

Exercises

Formal definition and design of PAs

Construct (deterministic or nondeterministic) PA that accept the following languages

1. $\{w \in \{0,1\}^* : w \text{ is a palindrome}\}$.
2. $\{w \in \{0,1\}^* : w \text{ contains: }\}$
   - more 1s than 0s.
   - less 1s than 0s.
   - equal number of 1s and 0s.
3. Combine all PAs from exercise 2.
4. $\{0^{2n}1^n : n \geq 1\}$.
5. $\{0^n 1^m 0^n : n \geq 1, m \geq 1\}$.

Exercises

Formal definition and design of PAs

Construct (deterministic or nondeterministic) PA that accept the following languages

1. $\{w \in \{0,1\}^* : w \text{ is a palindrome}\}$.
2. $\{w \in \{0,1\}^* : w \text{ contains: }\}$
   - more 1s than 0s.
   - less 1s than 0s.
   - equal number of 1s and 0s.
3. Combine all PAs from exercise 2.
4. $\{0^{2n}1^n : n \geq 1\}$.
5. $\{0^n 1^m 0^n : n \geq 1, m \geq 1\}$.

Exercises

Formal definition and design of PAs

Construct (deterministic or nondeterministic) PA that accept the following languages

1. $\{w \in \{0,1\}^* : w \text{ is a palindrome}\}$.
2. $\{w \in \{0,1\}^* : w \text{ contains: }\}$
   - more 1s than 0s.
   - less 1s than 0s.
   - equal number of 1s and 0s.
3. Combine all PAs from exercise 2.
4. $\{0^{2n}1^n : n \geq 1\}$.
5. $\{0^n1^m0^n : n \geq 1, m \geq 1\}$.