Theory of Computation

Lab Session 9

March 24, 2016

INNOPOLIS UNIVERSITY

Agenda

- History
- Non-determinism:
  - FSA;
  - TM;
  - PDA.

A bit of history.

Gottfried Wilhelm Leibniz

- "It is unworthy of excellent men to lose hours like slaves in the labor of calculation which could safely be regulated to anyone else if machines were used."
- 1646 – 1716

The "decision problem" (1928)

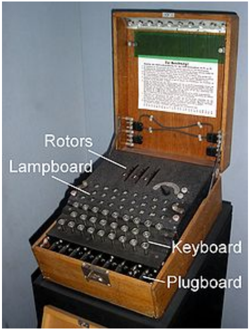
- The problem asks for an algorithm that takes as input a statement of a first-order logic and answers "Yes" or "No" according to whether the statement is provable from the axioms using the rules of logic.
- David Hilbert, 1928

Alan Turing (1)

- Independently negatively answered the decision problem.
- As we have seen he defined the nowadays Turing Machine – a machine foundation for computing.

Alan Turing (2)

- Led to Von Neumann computers and family of imperative programming languages.



(a)

(b)

(c)

Alonzo Church (1)

- Church's Theorem (1936)
- Independently negatively answered the decision problem.
- 1903 – 1995

Alonzo Church (2)

- Defined the Lambda (λ) Calculus - a language foundation for computing.
- Led to family of functional programming languages.
- Today the Lambda Calculus serves as a mathematical foundation for the study of functional programming languages.

Non-deterministic FSA.

Non-deterministic Finite State Automata (NDFSA)

Definition: NDFSA

A NDFSA is a tuple $\langle Q, I, \delta, q_0, F \rangle$, where Q, I, q_0, F are defined as in (D)FSA and the transition function is defined as

$$\delta : Q \times I \to \mathbb{P}(Q)$$

$\mathbb{P}$ is the powerset function (i.e. set of all possible subsets)

Non-deterministic Finite State Automata (NDFSA)

Definition: NDFSA

A NDFSA is a tuple $\langle Q, I, \delta, q_0, F \rangle$, where Q, I, q_0, F are defined as in (D)FSA and the transition function is defined as

$$\delta : Q \times I \rightarrow \mathbb{P}(Q)$$

$\mathbb{P}$ is the powerset function (i.e. set of all possible subsets)

A NDFSA modifies the definition of a FSA to permit transitions at each stage to either zero, one, or more than one states.

The extended transition δ^* for NDFSA

the extended transition δ^* for NDFSA

Let $M = \langle Q, I, \delta, q_0, F \rangle$ be a NDFSA. We define the extended transition function as follows:

1. For every $q \in Q$, $\delta^*(q, \epsilon) = \{q\}$
2. For every $q \in Q$, every $y \in I^*$, and every $i \in I$,

$$\delta^*(q, yi) = \bigcup_{q' \in \delta^*(q,y)} \delta(q', i)$$

Acceptance by a NDFSA

Acceptance by a NDFSA

Let $M = \langle Q, I, \delta, q_0, F \rangle$ be a NDFSA, and let $x \in I^*$. The string x is accepted by M iff

$$\delta^*(q_0, x) \cap F \neq \varnothing$$

and it is rejected by M otherwise.

Notion: Among the various possible runs (with the same input) of the NDFSA, it is sufficient that one of them succeeds to accept the input string.

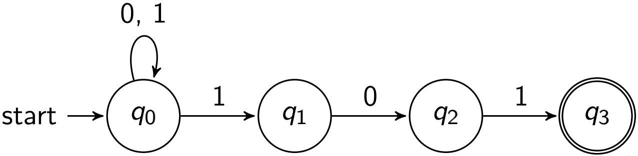
Exercises on NDFSA

Build NDFSAs that recognise the following languages:

- $L_1 = \{x \in \{0,1\}^* \mid \; x \text{ ends with } 101\}$;
- $L_2 = \{xy \mid x \in \{a\}^* \wedge y \in \{a,b\}^* \wedge\, y$ does not start with 'b' $\wedge$ every 'a' in y is followed by exactly one 'b'$\}$;
- $L_3 = \{x \in \{a,b,c\}^* \mid x$ ends with either ab, bc or $ca\}$;

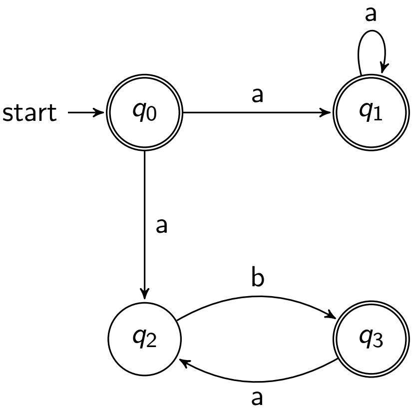
Solution (1)

NDFSA that recognises the language:
$L_1 = \{x \in \{0,1\}^* \mid \ x \text{ ends with } 101\}$



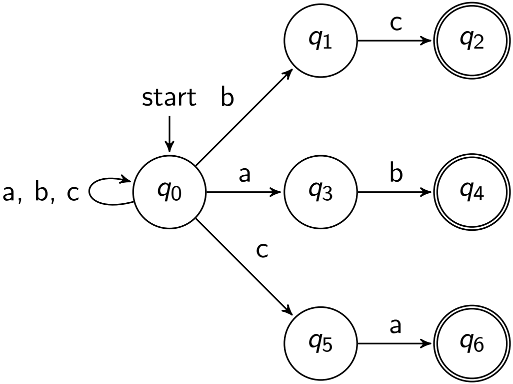
Solution (2)

NDFSA that recognises the language:
$L_2 = \{xy \mid x \in \{a\}^* \wedge y \in \{a, b\}^* \wedge$ y does not start with 'b' $\wedge$ every 'a' in y is followed by exactly one 'b'$\}$



Solution (3)

NDFSA that recognises the language: $L_3 = \{x \in \{a, b, c\}^* \mid x$ ends with either ab, bc or $ca\}$



Homework on NDFSA

NDFSAs are no more powerful than FSAs. A NFSA can be turned into an NDFSA that accepts the same language. Provide equivalent FSAs for previous exercises using the algorithm seen during the lecture.

Non-deterministic TM.

Non-deterministic Turing Machine (NDTM)

To define a NDTM, we need to change the transition function (all the other elements remain as in a (D)TM):

Definition: NDTM

A NDTM is a tuple $\langle Q, I, \Gamma, \delta, q_0, Z_0, F\rangle$, where $Q, I, \Gamma, q_0, Z_0, F$ are defined as in (D)TM and the transition function is defined as

$$\delta : (Q - F)\times(I\cup\{\_\})\times(\Gamma\cup\{\_\})^k \to \mathbb{P}\left(Q \times (\Gamma \cup \{\_\})^k \times \{R, L, S\}^{k+1}\right)$$

Acceptance: Among the various possible runs (with the same input) of the NDTM, it is sufficient that one of them succeeds to accept the input string.

Homework

Provide a proof for the following theorem

Theorem

For every NDTM $T = \langle Q, I, \Gamma, \delta, q_0, Z_0, F \rangle$, there is an (deterministic) TM $T_1 = \langle Q_1, I, \Gamma_1, \delta_1, q_1, Z_0, F_1 \rangle$ with $L(T_1) = L(T)$

Non-deterministic PDA.

Non-deterministic Pushdown Automaton (NDPDA)

Definition: NDPDA

A NDPDA is a tuple $\langle Q, I, \Gamma, \delta, q_0, Z_0, F \rangle$, where $Q, I, \Gamma, q_0, Z_0, F$ are defined as in (D)PDA and the transition function is defined as

$$\delta : Q \times (I \cup \{\epsilon\}) \times \Gamma \to \mathbb{P}_{\mathtt{F}}(Q \times \Gamma^*)$$

where $\mathbb{P}_{\mathtt{F}}$ indicates finite subsets.

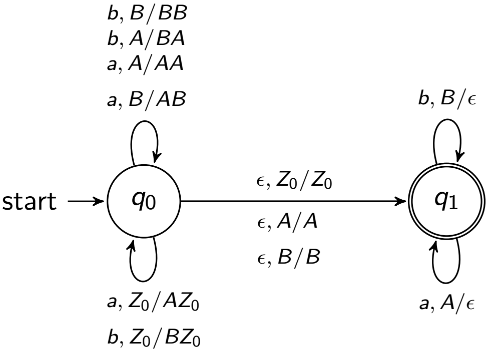
Exercises

Build NDPDAs that recognise the following languages:

1. $L_1 = \{ww^R \mid w \in \{a, b\}^*\}$ where w^R is the reversed string w.
2. $L_2 = \{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}$.
3. The language of well-parenthesised strings. E.g. a string in the language: (()())(), a string that does not belong to the language: (()()() – the alphabet is $I = \{`(', `)'\}$.
4. $L_4 = \{w \in \{a, b\}^* \mid \phi(w, a) = \phi(w, b)\}$ where $\phi(s, c)$ is the number of occurrences of the character c in the string s.

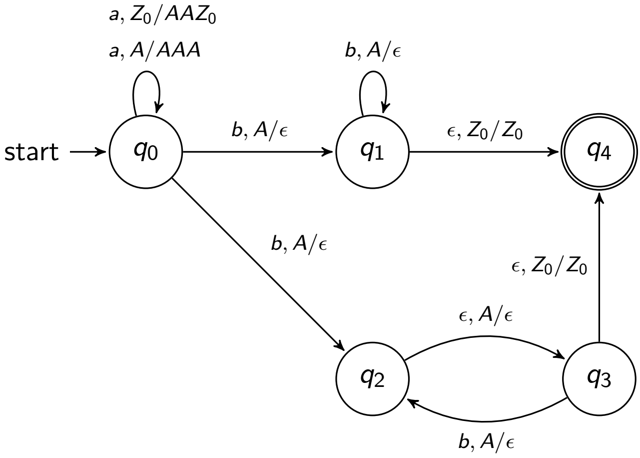
Solution (1)

NDPDA accepting $L_1 = \{ww^R \mid w \in \{a, b\}^*\}$ where w^R is the reversed string w.



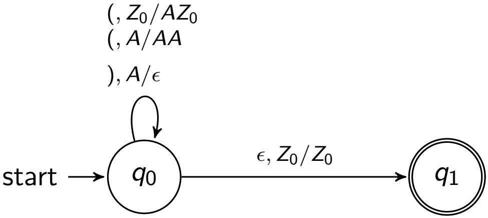
Solution (2)

NDPDA accepting $L_2 = \{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}$.



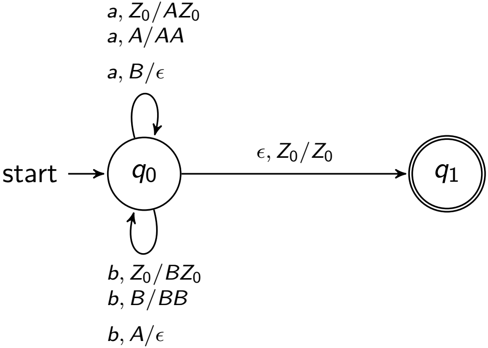
Solution (3)

NDPDA accepting The language of well-parenthesised strings. E.g. a string in the language: (()())(), a string that does not belong to the language: (()()() – the alphabet is $I = \{`(', `)'\}$.



Solution (4)

NDPDA accepting the language
$L_4 = \{w \in \{a, b\}^* \mid \phi(w, a) = \phi(w, b)\}$ where $\phi(s, c)$ is the number of occurrences of the character c in the string s.



Exercises

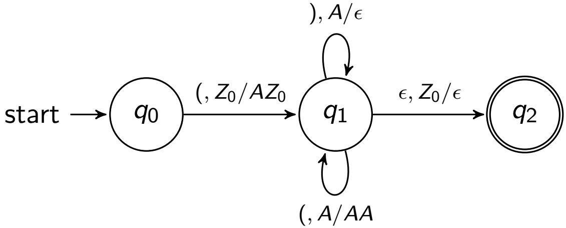
NDPDAs are more powerful than (D)PDA. Let's try to build (D)PDAs that recognise the languages previously defined. (D)PDA accepting the language of well-parenthesised strings. E.g. a string in the language: (()())(), a string that does not belong to the language: (()()() – the alphabet is $I = \{`(', `)'\}$.

Exercises

NDPDAs are more powerful than (D)PDA. Let's try to build (D)PDAs that recognise the languages previously defined. (D)PDA accepting the language of well-parenthesised strings. E.g. a string in the language: (()())(), a string that does not belong to the language: (()()() – the alphabet is $I = \{'(', ')'\}$. We did it in a previous lab. session:

Exercises

NDPDAs are more powerful than (D)PDA. Let's try to build (D)PDAs that recognise the languages previously defined. (D)PDA accepting the language of well-parenthesised strings. E.g. a string in the language: (()())(), a string that does not belong to the language: (()()() – the alphabet is $I = \{‘(’, ‘)’\}$. We did it in a previous lab. session:



Exercises

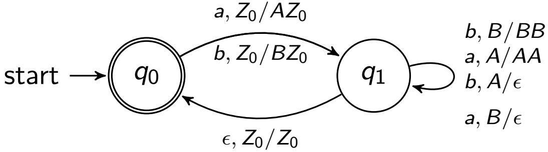
(D)PDA accepting the language
$L_4 = \{w \in \{a, b\}^* \mid \phi(w, a) = \phi(w, b)\}$ where $\phi(s, c)$ is the number of occurrences of the character c in the string s.

Exercises

(D)PDA accepting the language $L_4 = \{w \in \{a, b\}^* \mid \phi(w, a) = \phi(w, b)\}$ where $\phi(s, c)$ is the number of occurrences of the character c in the string s. We did it in a previous lab. session:

Exercises

(D)PDA accepting the language
$L_4 = \{w \in \{a, b\}^* \mid \phi(w, a) = \phi(w, b)\}$ where $\phi(s, c)$ is the number of occurrences of the character c in the string s. We did it in a previous lab. session:



Exercises

What about (D)PDAs accepting the languages

1. $L_1 = \{ww^R \mid w \in \{a, b\}^*\}$ where w^R is the reversed string w.
2. $L_2 = \{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}$.

Exercises

What about (D)PDAs accepting the languages

1. $L_1 = \{ww^R \mid w \in \{a,b\}^*\}$ where w^R is the reversed string w.
2. $L_2 = \{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}$.

These languages cannot be accepted by any (D)PDA.

Exercises

What about (D)PDAs accepting the languages

1. $L_1 = \{ww^R \mid w \in \{a, b\}^*\}$ where w^R is the reversed string w.
2. $L_2 = \{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}$.

These languages cannot be accepted by any (D)PDA. **Homework:** Show that language L_1 cannot be accepted by a (D)PDA.