3.2 High-Level Specifications

Basic Concepts of Languages

- To define a language, we need words or strings, which are made up of symbols that consitute an alphabet
- Let W be an alphabet consisting of a set of symbols
    - Example: $W = \{a, b, c\}$
    - Then strings/words over W are "ε", "aab", "$baabc$", "$cccccca$", ...
    - "ε" is the empty string
    - W^* denotes the set of all finite strings over W and W^ω denotes the set of all infinite string over W
    - If x is a string over W, then $|x|$ denotes the length of x
    - Example, $|abc| = 3$, $|\varepsilon| = 0$

3.2 High-Level Specifications

Basic Concepts of Languages

- To define a language, we need words or strings, which are made up of symbols that consitute an alphabet
- Let W be an alphabet consisting of a set of symbols
  - Example: $W = \{a, b, c\}$
  - Then strings/words over W are "ε", "aab", "$baabc$", "$cccccca$", ...
  - "ε" is the empty string
  - W^* denotes the set of all finite strings over W and W^ω denotes the set of all infinite string over W
  - If x is a string over W, then $|x|$ denotes the length of x
  - Example, $|abc| = 3$, $|\varepsilon| = 0$
- A language is a set of (in)finite strings
- Examples
  - $\mathcal{L}_1 := \{\varepsilon, a, b, aa, ab\}$
  - $\mathcal{L}_2 := \{x \in \{a, b\}^* \mid |x| \leq 8\}$
- Since languages are sets of strings, new languages may be constructed by using operations on sets such as union, intersection, etc.
- Exmaples
  - $\mathcal{L}_1\mathcal{L}_2 = \{xy \mid x \in \mathcal{L}_1 \text{ and } y \in \mathcal{L}_2\}$

Regular/ω-Regular Expressions

- The four basic operations for constructing new languages from the existing ones are:
  - Union
  - Concatenation
  - Kleene star (finite repetition)
  - ω operator (infinite repetition)
- Let us start with the simplest possible languages: those consisting of a single string that is either the null string or a string of length one
- The languages that are obtained by repeatedly applying the four basic operations on these simple languages are called ω-regular languages
- ω-Regular expressions are representations of ω-regular languages

Notation: Language Concepts

- Let W be a non-empty set
  - Finite sequences $W^* = \bigcup_{T \in \mathbb{Z}_{\geq 0}} W^{[0;T[}$ where $\varepsilon = W^{[0;0[}$ is the empty word
  - Infinite sequences $W^\omega = W^{[0;\infty[}$
  - Finite and infinite sequences $W^\infty = W^\omega \cup W^*$
- Let $w_1 \in W^{[0;T_1[}$, $T_1 \in \mathbb{Z}_{\geq 0}$ and $w_2 \in W^{[0;T_2[}$, $T_2 \in \mathbb{Z}_{\geq 0} \cup \{\infty\}$
  - The concatenation of w_1 with w_2 is denoted by $w_1 w_2$ and defined by

$$w_1 w_2(t) := \begin{cases} w_1(t) & \text{for } t \in [0;\, T_1[\\ w_2(t - T_1) & \text{for } t \in [T_1;\, T_1 + T_2[\end{cases}$$

  - w_1 is a prefix of w_2, denoted by

$$w_1 \preceq w_2$$

  iff there exists $w_3 \in W^{[0;T_3[}$, $T_3 \in \mathbb{N} \cup \{\infty\}$ such that $w_1 w_3 = w_2$. In symbols

$$(w_1 \preceq w_2) \iff (\exists_{w_3 \in W^\infty}\, w_1 w_3 = w_2)$$

- Let $W_1 \subseteq W^*$ and $W_2 \subseteq W^\infty$, then we extend the concatenation to sets by

$$W_1 W_2 = \{w_1 w_2 \mid w_1 \in W_1 \wedge w_2 \in W_2\}$$

- We use ε to denote the identity element of the concatenation operator, i.e., for any $w_1 \in W^*$ and $w_2 \in W^\infty$ we have

$$\varepsilon w_1 = w_1 = w_1 \varepsilon \quad \text{and} \quad \varepsilon w_2 = w_2$$

Example: Prefixes

Let $W = \{a, b, c\}$ and $w_1 \in W^*$ and $w_2 \in W^\omega$

$$w_1 = abbbbccc$$
$$w_2 = abaabbaaabbbaaaabbbb\ldots$$

- $w_1' = ab$ is a prefix of w_1 and of w_2
- $w_1' = abbbbccc$ is a prefix of w_1
- $w_1' = abbc$ is not a prefix of w_1
- $w_2' = w_2$ is not a prefix of w_2

Regular/ω-Regular Expression

- Syntax: Let W be a finite set (often referred to as alphabet). ω-regular expressions over W are build from symbols

$$\varnothing \quad | \quad \varepsilon \quad | \quad w \quad | \quad . \quad | \quad + \quad | \quad {}^* \quad | \quad {}^\omega$$

  in an inductive manner:
  - $\varnothing$ and ε are ω-regular expressions.
  - If $w \in W$, then w is an ω-regular expressions.
  - If α and β are ω-regular expressions, then

$$\alpha.\beta, \quad \alpha + \beta, \quad \alpha^* \quad \text{and} \quad \alpha^\omega$$

    are ω-regular expressions.
  - A ω-regular expression over W that does not contain ω, is a regular expression over W.
- Semantics: Every ω-regular expression α over W induces a set

$$\mathcal{L}(\alpha) \subseteq W^\infty$$

  defined by
  - $\mathcal{L}(\varnothing) = \varnothing$
  - $\mathcal{L}(\varepsilon) = \{\varepsilon\}$
  - $\mathcal{L}(w) = \{w\}$
  - $\mathcal{L}(\alpha.\beta) = \mathcal{L}(\alpha)\mathcal{L}(\beta)$
  - $\mathcal{L}(\alpha + \beta) = \mathcal{L}(\alpha) \cup \mathcal{L}(\beta)$
  - $\mathcal{L}(\alpha^*) = \mathcal{L}(\alpha)^*$
  - $\mathcal{L}(\alpha^\omega) = \mathcal{L}(\alpha)^\omega$

Regular/ω-Regular Expression

Language	ω-Regular expression
ε	ε
$\{0, 1\}$	$0 + 1$
$\{0, 10\}$	$0 + 10$
$\{110\}^*\{0, 1\}$	$(110)^*(0 + 1)$
$\{1\}^*\{10\}$	1^*10
$\{10, 111, 11000\}^*$	$(10 + 111 + 11000)^*$

Regular/ω-Regular Expression

Language	ω-Regular expression
ε	ε
$\{0, 1\}$	$0 + 1$
$\{0, 10\}$	$0 + 10$
$\{110\}^*\{0, 1\}$	$(110)^*(0 + 1)$
$\{1\}^*\{10\}$	1^*10
$\{10, 111, 11000\}^*$	$(10 + 111 + 11000)^*$

- For ω-regular expressions r and s over W, corresponding to the languages $\mathcal{L}_r$ and $\mathcal{L}_s$, respectively, each of the following are ω-regular expressions over W, corresponding to the languages next to it
  - (rs) corresponding to $\mathcal{L}_r\mathcal{L}_s$
  - $(r + s)$ corresponding to $\mathcal{L}_r \cup \mathcal{L}_s$
  - (r^*) corresponding to $\mathcal{L}_r{}^*$
  - (r^ω) corresponding to $\mathcal{L}_r{}^\omega$
- Simplifying ω-regular expressions
  - $1^*(1 + \varepsilon) = 1^*$
  - $1^*1^* = 1^*$
  - $1^*1^\omega = 1^\omega$
  - $0^* + 1^* = 1^* + 0^*$

Regular Expressions

$$\alpha \quad ::= \quad \varnothing \quad | \quad \varepsilon \quad | \quad A \quad | \; \alpha_1 + \alpha_2 \; | \quad \alpha_1 . \alpha_2 \quad | \quad \alpha^*$$

symbol for the
empty language

Regular Expressions

$$\alpha \quad ::= \quad \varnothing \quad | \quad \varepsilon \quad | \quad A \quad | \quad \alpha_1 + \alpha_2 \quad | \quad \alpha_1 . \alpha_2 \quad | \quad \alpha^*$$

symbol for the singleton consisting of the empty word

Regular Expressions

$$\alpha \quad ::= \quad \varnothing \quad | \quad \varepsilon \quad | \quad A \quad | \quad \alpha_1 + \alpha_2 \quad | \quad \alpha_1.\alpha_2 \quad | \quad \alpha^*$$

symbol for the
singleton consisting
of the word A
where $A \in W$

Regular Expressions

$$\alpha \quad ::= \quad \varnothing \quad | \quad \varepsilon \quad | \quad A \quad | \quad \underset{\substack{\uparrow \\ \text{union}}}{\alpha_1 + \alpha_2} \quad | \quad \alpha_1 . \alpha_2 \quad | \quad \alpha^*$$

Regular Expressions

$$\alpha \quad ::= \quad \varnothing \quad | \quad \varepsilon \quad | \quad A \quad | \quad \alpha_1 + \alpha_2 \quad | \quad \underbrace{\alpha_1 . \alpha_2}_{\text{concatenation}} \quad | \quad \alpha^*$$

Regular Expressions

$$\alpha \quad ::= \quad \varnothing \quad | \quad \varepsilon \quad | \quad A \quad | \quad \alpha_1 + \alpha_2 \quad | \quad \alpha_1.\alpha_2 \quad | \quad \alpha^*$$

Kleene star

Regular Expressions

$$\alpha \quad ::= \quad \varnothing \quad | \quad \varepsilon \quad | \quad A \quad | \quad \alpha_1 + \alpha_2 \quad | \quad \alpha_1.\alpha_2 \quad | \quad \alpha^*$$

$\alpha \mapsto \mathcal{L}(\alpha) \subseteq W^*$ language of finite words

$\mathcal{L}(\varnothing) = \varnothing$	$\mathcal{L}(\varepsilon) = \{\varepsilon\}$	$\mathcal{L}(A) = \{A\}$
$\mathcal{L}(\alpha_1 + \alpha_2)$	$= \mathcal{L}(\alpha_1) \cup \mathcal{L}(\alpha_2)$	union
$\mathcal{L}(\alpha_1.\alpha_2)$	$= \mathcal{L}(\alpha_1)\mathcal{L}(\alpha_2)$	concatenation
$\mathcal{L}(\alpha^*)$	$= \mathcal{L}(\alpha)^*$	Kleene closure

Regular Expressions – Examples

- $\mathcal{L}$ is the language of all finite strings of 0s and 1s that have even length. What is the regular expression corresponding to $\mathcal{L}$?
- Answer:
  - $(00 + 01 + 10 + 11)^*$

- $\mathcal{L}$ is the language of all finite strings of 0s and 1s that have even length. What is the regular expression corresponding to $\mathcal{L}$?
- Answer:
  - $(00 + 01 + 10 + 11)^*$

- $\mathcal{L}$ is the language of all finite strings of 0s and 1s that have odd length. What is the regular expression corresponding to $\mathcal{L}$?
- Answer:
  - 0 or 1 followed by even length string
  - $(0 + 1)(00 + 01 + 10 + 11)^*$

Regular Expressions – Examples

- $\mathcal{L}$ is the language of all finite strings of 0s and 1s that have at least one 1. What is the regular expression corresponding to $\mathcal{L}$?

Regular Expressions – Examples

- $\mathcal{L}$ is the language of all finite strings of 0s and 1s that have at least one 1. What is the regular expression corresponding to $\mathcal{L}$?
- Possible answers:
  - $0^*1(0+1)^*$
  - $(0+1)^*1(0+1)^*$
  - $(0+1)^*10^*$

Regular Expressions – Examples

- $\mathcal{L}$ is the language of all finite strings of 0s and 1s that have at least one 1. What is the regular expression corresponding to $\mathcal{L}$?
- Possible answers:
  - $0^*1(0+1)^*$
  - $(0+1)^*1(0+1)^*$
  - $(0+1)^*10^*$

- $\mathcal{L}$ is the language of all finite strings of 0s and 1s that have a length of less than or equal 6. What is the regular expression corresponding to $\mathcal{L}$?

- $\mathcal{L}$ is the language of all finite strings of 0s and 1s that have at least one 1. What is the regular expression corresponding to $\mathcal{L}$?
- Possible answers:
  - $0^*1(0+1)^*$
  - $(0+1)^*1(0+1)^*$
  - $(0+1)^*10^*$

- $\mathcal{L}$ is the language of all finite strings of 0s and 1s that have a length of less than or equal 6. What is the regular expression corresponding to $\mathcal{L}$?
- Answer:
  - $(0+1+\varepsilon)^6$

- $\mathcal{L}$ is the language of all finite strings of 0s and 1s that have at least one 1. What is the regular expression corresponding to $\mathcal{L}$?
- Possible answers:
  - $0^*1(0+1)^*$
  - $(0+1)^*1(0+1)^*$
  - $(0+1)^*10^*$

- $\mathcal{L}$ is the language of all finite strings of 0s and 1s that have a length of less than or equal 6. What is the regular expression corresponding to $\mathcal{L}$?
- Answer:
  - $(0+1+\varepsilon)^6$

- $\mathcal{L}$ is the language of all strings of 0s and 1s that ends with 1 and does not contain the substring 00. What is the regular expression corresponding to $\mathcal{L}$?

Regular Expressions – Examples

- $\mathcal{L}$ is the language of all finite strings of 0s and 1s that have at least one 1. What is the regular expression corresponding to $\mathcal{L}$?
- Possible answers:
  - $0^*1(0+1)^*$
  - $(0+1)^*1(0+1)^*$
  - $(0+1)^*10^*$

- $\mathcal{L}$ is the language of all finite strings of 0s and 1s that have a length of less than or equal 6. What is the regular expression corresponding to $\mathcal{L}$?
- Answer:
  - $(0+1+\varepsilon)^6$

- $\mathcal{L}$ is the language of all strings of 0s and 1s that ends with 1 and does not contain the substring 00. What is the regular expression corresponding to $\mathcal{L}$?
- Answer:
  - $(1+01)^*(1+01)$

ω-Regular Expressions

$$\alpha \quad ::= \quad \varnothing \quad | \quad \varepsilon \quad | \quad A \quad | \quad \alpha_1 + \alpha_2 \quad | \quad \alpha_1 . \alpha_2 \quad | \quad \alpha^*$$

- ω-regular expressions
  - $=$ regular expressions $+$ ω-operator (represented as α^ω)
- Kleene star: finite reptition
- ω-operator: infinite repetition

for $\mathcal{L} \subseteq W^*$:

$$\mathcal{L}^\omega \stackrel{\text{def}}{=} \{ w_1 w_2 w_3 \ldots \mid w_i \in \mathcal{L} \text{ for all } i \geq 1 \}$$

note: $\mathcal{L}^\omega \subseteq W^\omega$ if $\varepsilon \notin \mathcal{L}$

ω-Regular Expressions

syntax of ω-regular expressions over alphabet W:

$$\gamma = \alpha_1.\beta_1^\omega + \ldots + \alpha_1.\beta_n^\omega$$

where α_i, β_i are regular expressions over W such that $\varepsilon \notin \mathcal{L}(\beta_i)$

syntax of ω-regular expressions over alphabet W:

$$\gamma = \alpha_1.\beta_1^\omega + \ldots + \alpha_1.\beta_n^\omega$$

where α_i, β_i are regular expressions over W such that $\varepsilon \notin \mathcal{L}(\beta_i)$

- The language generated by γ is given by:

$$\mathcal{L}_\omega(\gamma) \stackrel{\text{def}}{=} \bigcup_{1 \le i \le n} \mathcal{L}(\alpha_i)\mathcal{L}(\beta_i)^\omega \subseteq W^\omega$$

syntax of ω-regular expressions over alphabet W:

$$\gamma = \alpha_1.\beta_1^\omega + \ldots + \alpha_1.\beta_n^\omega$$

where α_i, β_i are regular expressions over W such that $\varepsilon \notin \mathcal{L}(\beta_i)$

- The language generated by γ is given by:

$$\mathcal{L}_\omega(\gamma) \stackrel{\text{def}}{=} \bigcup_{1 \le i \le n} \mathcal{L}(\alpha_i)\mathcal{L}(\beta_i)^\omega \subseteq W^\omega$$

- A language $\mathcal{L}$ is called ω-regular iff there exists an ω-regular expression γ such that $\mathcal{L} = \mathcal{L}_\omega(\gamma)$

- What is the language represented by $(A^*B)^\omega$?
  - Set of all infinite words over $W = \{A, B\}$ containing infinitely many Bs (Why not infinitely many As?)

- What is the language represented by $(A^*B)^\omega$?
  - Set of all infinite words over $W = \{A, B\}$ containing infinitely many Bs (Why not infinitely many As?)

- What is the language represented by $(A^*B)^\omega + (B^*A)^\omega$?
  - Set of all infinite words over $W = \{A, B\}$ containing infinitely many As and infinitely many Bs
  - This is equivalent to W^ω

ω-Regular Expressions – Examples

- Let $W = \{A, B\}$
- What ist the ω-regular expression for the set of all infinite words over W containing only finitely many As?
- Answer: $(A + B)^* B^\omega$

- Let $W = \{A, B\}$
- What ist the ω-regular expression for the set of all infinite words over W containing only finitely many As?
- Answer: $(A + B)^* B^\omega$

- Let $W = \{A, B\}$
- What ist the ω-regular expression for the set of all infinite words over W where A is immediately followed by B?
- Answer: $(B^* AB)^* B^\omega + (B^* AB)^\omega$

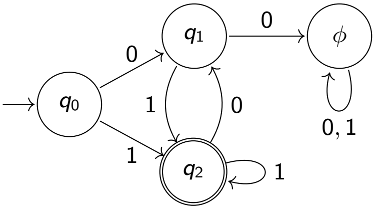
- Let $W = \{A, B\}$
- What ist the ω-regular expression for the set of all infinite words over W containing only finitely many As?
- Answer: $(A + B)^* B^\omega$

- Let $W = \{A, B\}$
- What ist the ω-regular expression for the set of all infinite words over W where A is immediately followed by B?
- Answer: $(B^* AB)^* B^\omega + (B^* AB)^\omega$

- Let $W = \{A, B\}$
- What is the ω-regular expression for the set of all infinite words over W where each A is followed eventually by B?
- Answer: $(B^* A^+ B)^* B^\omega + (B^* A^+ B)^\omega$
- This is equivalent to $(A^* B)^\omega$
- Remember that $\alpha^+ = \alpha\alpha^*$

Recognizing Regular Languages

- What kind of machine is needed to recognize a regular language?
  - How much memory is required?
- $\mathcal{L}$ is the language of all strings of 0s and 1s that ends with 1 and does not contain the substring 00. What is the regular expression corresponding to $\mathcal{L}$?
- $(1 + 01)^*(1 + 01)$



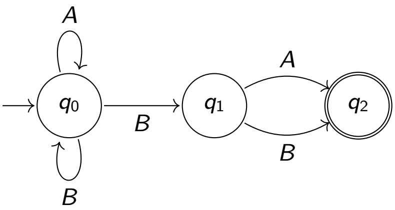
accepting state

Definition: Nondeterministic Finite Automaton

A nondeterministic finite automaton (NFA) $\mathcal{A}$ is a tuple $(Q, \Sigma, \delta, Q_0, F)$ where

- Q is a finite set of states
- Σ is an alphabet
- $\delta \colon Q \times \Sigma \to 2^Q$ ($\delta \colon Q \times \Sigma \rightrightarrows Q$) is a transition function
- $Q_0 \subseteq Q$ is a set of initial states
- $F \subseteq Q$ is a set of accepting states

Example: NFA



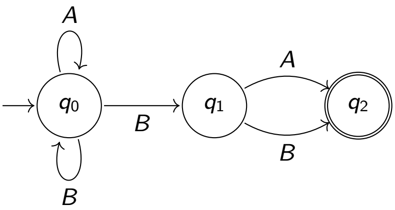
- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{A, B\}$
- $\delta(q_0, A) = \{q_0\}$, $\delta(q_0, B) = \{q_0, q_1\}$, $\delta(q_1, A) = \{q_2\}$, $\delta(q_1, B) = \{q_2\}$
- $Q_0 = \{q_0\}$
- $F = \{q_2\}$

Definition: Accepted Language of an NFA

- Let $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ be an NFA and $w = A_1, A_2, \ldots, A_n \in \Sigma^*$ a finite word. A run for w in $\mathcal{A}$ is a finite sequence of states $q_0, q_1, \ldots, q_n$ such that
  - $q_0 \in Q_0$
  - $q_{i+1} \in \delta(q_i, A_{i+1})$ for all $0 \leq i < n$
- Run $q_0, q_1, \ldots, q_n$ is called accepting if $q_n \in F$. A finite word $w \in \Sigma^*$ is called accepted by $\mathcal{A}$ if there exists an accepting run for w. The accepted language of $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, is the set of finite words accepted by $\mathcal{A}$, i.e.

$$\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid \text{there exists an accepting run for } w \text{ in } \mathcal{A}\}$$

Example: Accepted Language

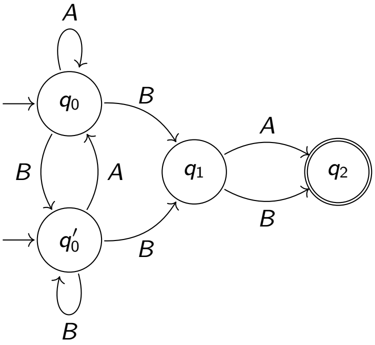


- $\mathcal{L}(\mathcal{A})$ is defined by the regular expression $(A + B)^* B(A + B)$
- Word over $\{A, B\}$ where the last but one symbol is B

Definition: Equivalence of NFAs

- Let $\mathcal{A}$ and $\mathcal{A}'$ be NFAs with the same alphabet. $\mathcal{A}$ and $\mathcal{A}'$ are called equivalent if $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.

Example



Definition: Synchronous Product of NFAs

- For NFA $\mathcal{A}_i = (Q_i, \Sigma, \delta_i, Q_{0,i}, F_i)$ with $i = 1, 2$, the product automaton $\mathcal{A}_1 \otimes \mathcal{A}_2 = (Q_1 \times Q_2, \Sigma, \delta, Q_{0,1} \times Q_{0,2}, F_1 \times F_2)$, where δ is defined by

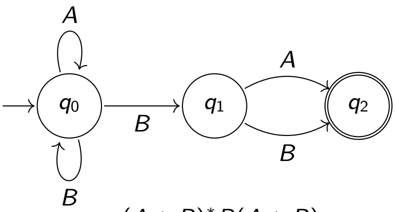
$$\frac{q_1 \xrightarrow{A}_1 q_1' \wedge q_2 \xrightarrow{A}_2 q_2'}{\langle q_1, q_2 \rangle \xrightarrow{A} \langle q_1', q_2' \rangle}$$

Fact

$$\mathcal{L}(\mathcal{A}_1 \otimes \mathcal{A}_2) = \mathcal{L}(A_1) \cap \mathcal{L}(\mathcal{A}_2)$$

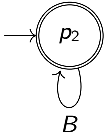
Example: Synchronous Product of NFAs

$\mathcal{A}_1$:



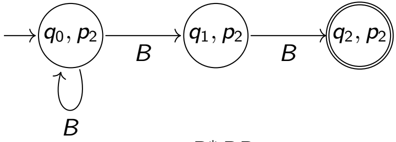
$(A + B)^* B(A + B)$

$\mathcal{A}_2$:



B^*

$\mathcal{A}_1 \otimes \mathcal{A}_2$:



$B^* BB$

Recognizing Regular Languages

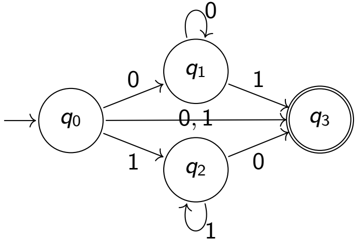
- Klenee's Theorem: A language $\mathcal{L}$ is regular if and only if there is a nondeterministic finite automaton recognizing it
- If $\mathcal{L}_1$ and $\mathcal{L}_2$ are regular languages in Σ^* then $\mathcal{L}_1 \cup \mathcal{L}_2, \mathcal{L}_1 \cap \mathcal{L}_2, \mathcal{L}_1 - \mathcal{L}_2, \Sigma^* \setminus \mathcal{L}_1$ are all regular languages

Definition: Deterministic Finite Automaton (DFA)

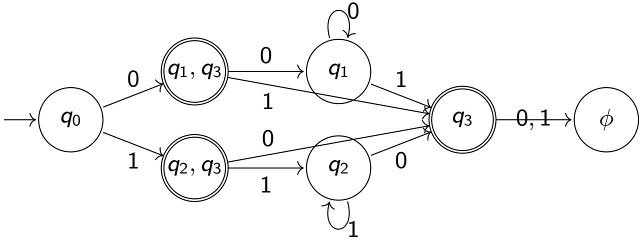
- Let $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ be an NFA. $\mathcal{A}$ is called deterministic if
  - $|Q_0| \leq 1$
  - $|\delta(q, A)| \leq 1$ for all states $q \in Q$ and all symbols $A \in \Sigma$
- Determination of a DFA from an NFA by powerset construction
- DFA $\mathcal{A}$ is called total if $|Q_0| = 1$ and $|\delta(q, A)| = 1$ for all states $q \in Q$ and all symbols $A \in \Sigma$

From NFA to DFA

- If a language $\mathcal{L}$ is recognized by an NFA, then there exists a DFA recognizing the same language



q	$\delta_1(q,0)$	$\delta_1(q,1)$
q_0	$\{q_1, q_3\}$	$\{q_2, q_3\}$
q_1	$\{q_1\}$	$\{q_3\}$
q_2	$\{q_3\}$	$\{q_2\}$
q_3	ϕ	ϕ



Exercise 5: NFA

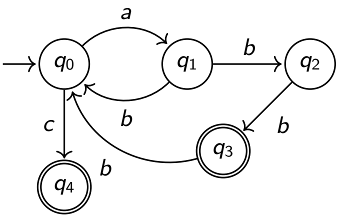
Exercise 1

Give the NFAs of the following languages.

1. $\mathcal{L}(((a+b)^* c)^* + a)$
2. $\mathcal{L}((a^+ b^* c)^+)$

Exercise 2

Consider the following NFA:



Give the regular expression that generates the language of the NFA.

Useful Notations

NFA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ over the alphabet $\Sigma = 2^{AP}$, where AP is a set of atomic propostions!

Notation: symbolic notation for the labels of transition

If Φ is a propositional formula over AP then $q \xrightarrow{\Phi} p$ stands for the set of transitions $q \xrightarrow{A} p$ where $A \subseteq 2^{AP}$ such that $A \models \Phi$.

Example

If $AP = \{a, b, c\}$, then

$$q \xrightarrow{a \wedge \neg b} p \mathrel{\hat{=}} \{q \xrightarrow{A} p \mid A = \{a, c\} \text{ or } A = \{a\}\}$$

$$q \xrightarrow{\text{true}} p \mathrel{\hat{=}} \{q \xrightarrow{A} p \mid A \subseteq 2^{AP}\}$$

Propositional formulae over set AP can be inductively rewritten as

$$\Phi \quad ::= \quad \text{true} \quad \mid \quad a \quad \mid \quad \Phi_1 \wedge \Phi_2 \quad \mid \quad \neg\Phi$$

where $a \in AP$.

ω-Regular Properties

Definition: ω-regular property

E is called an ω-regular property iff there exists an ω-regular expression γ over $\Sigma = 2^{AP}$ such that $E = \mathcal{L}_\omega(\gamma)$

Example

Examples for $AP = \{a, b\}$

- "always $a \vee \neg b$" $\quad \left(\varnothing + \{a\} + \{a, b\}\right)^\omega$
- Recall that the alphabet is $2^{AP} = \{\varnothing, \{a\}, \{b\}, \{a, b\}\}$

Another example

Examples for $AP = \{a, b\}$

- "infinitely often a" $\quad \left(\left(\varnothing + \{b\}\right)^*.(\{a\} + \{a, b\})\right)^\omega$
- "eventually a" $\quad \left(2^{AP}\right)^*.\left(\{a\} + \{a, b\}\right).\left(2^{AP}\right)^\omega$
- "from some moment on a" $\quad \left(2^{AP}\right)^*.\left(\{a\} + \{a, b\}\right)^\omega$

  where $2^{AP} \hat{=} \varnothing + \{a\} + \{b\} + \{a, b\}$

ω-Regular Properties: Using Symbolic Notation

Again, $AP = \{a, b\}$

- "always $a \vee \neg b$"

$$\big(a \vee \neg b\big)^{\omega} \quad \hat{=} \quad \big(\varnothing + \{a\} + \{a, b\}\big)^{\omega}$$

- "infinitely often a"

$$\big(\big(\neg a\big)^{*}.a\big)^{\omega} \quad \hat{=} \quad \big(\big(\varnothing + \{b\}\big)^{*}.\big(\{a\} + \{a, b\}\big)\big)^{\omega}$$

- "from some moment on a"

$$\text{true}^{*}.a^{\omega}$$

- "whenever a then b will hold somewhen later"

$$\big(\big(\neg a\big)^{*}.a.\text{true}^{*}.b\big)^{*}.\big(\neg a\big)^{\omega} + \big(\big(\neg a\big)^{*}.a.\text{true}^{*}.b\big)^{\omega}$$

ω-Automata

- Recall that regular languages were recognized by nondeterministic finite automata (NFA) (which was shown to be equivalent to DFA)
- How do we recognize ω-regular languages (i.e. languages described by ω-regular expressions)?
  - Use an ω-automata (which are acceptors for infinite words)
  - These are called Nondeterministic Büchi Automata (NBA)

Nondeterministic Büchi Automata (NBA):

- syntax as for NFA (non-deterministic finite automata)
- semantics: language of infinite words

Nondeterministic Büchi Automata (NBA)

Definition: NBA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$

- Q finite set of states
- Σ alphabet
- $\delta\colon Q \times \Sigma \to 2^Q$ transition relation
- $Q_0 \subseteq Q$ set of initial states
- $F \subseteq Q$ set of final states, also called accepting states

  run for a word $A_0 A_1 A_2 \ldots \in \Sigma^\omega$:

  $$\text{state sequence } \pi = q_0 q_1 q_2 \ldots$$
  $$\text{where } q_0 \in Q_0 \text{ and } q_{i+1} \in \delta(q_i, A_i) \text{ for } i \geq 0$$

  run π is accepting if there exists infinitely many $i \in \mathbb{N} : q_i \in F$

- A word is accepted if an accepting state is visited infinitely often

Nondeterministic Büchi Automata (NBA)

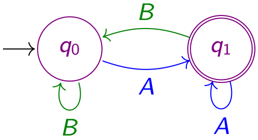
Definition: NBA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$

- Q finite set of states
- Σ alphabet
- $\delta\colon Q \times \Sigma \to 2^Q$ transition relation
- $Q_0 \subseteq Q$ set of initial states
- $F \subseteq Q$ set of final states, also called accepting states
- A word is accepted if an accept state is visited infinitely often

accepted language $\mathcal{L}_\omega(\mathcal{A}) \subseteq \Sigma^\omega$ is given by:

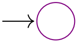
$$\mathcal{L}_\omega(\mathcal{A}) \stackrel{\text{def}}{=} \text{set of infinite words over } \Sigma \text{ that have an accepting run in } \mathcal{A}$$

Notations



NBA with state space $\{q_0, q_1\}$

q_0 initial state
q_1 accept state
alphabet $\Sigma = \{A, B\}$

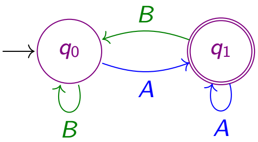
 initial state

 nonfinal state

 final state

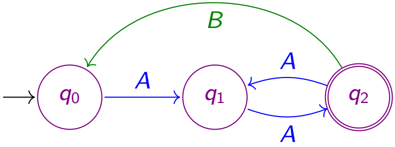
 final state
(alternative notation)

NBA – Examples



accepted language:
set of all infinite words that contain infinitely many A's

$$(B^*.A)^\omega$$



accepted language:
" every B is preceded by a positive even number of A's"

$$((A.A)^+.B)^\omega + ((A.A)^+.B)^*.A^\omega$$

$$\left.\begin{array}{l} AABAABAAB\ldots \\ AAAAAAAAA\ldots \end{array}\right\} \text{accepted words}$$

NBA for LT properties

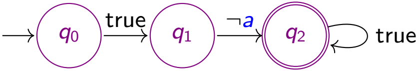
Recall: NBA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$

- Q finite set of states
- Σ alphabet $\rightarrow$ $\boxed{\text{here: } \Sigma = 2^{AP}}$
- $\delta\colon Q \times \Sigma \rightarrow 2^Q$ transition relation
- $Q_0 \subseteq Q$ set of initial states
- $F \subseteq Q$ set of final states, also called accepting states

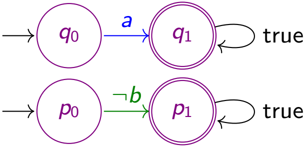
accepted language $\mathcal{L}_\omega(\mathcal{A}) \subseteq \Sigma^\omega$ is given by:

$$\mathcal{L}_\omega(\mathcal{A}) \stackrel{\text{def}}{=} \begin{array}{l}\text{set of infinite words over } \Sigma \\ \text{that have an accepting run in } \mathcal{A}\end{array}$$

NBA for LT properties



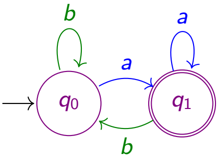
$\mathcal{L}_\omega(\mathcal{A}) \hat{=}$
$\text{true}.\neg a.\text{true}^\omega$



$(a \vee \neg b).\text{true}^\omega$

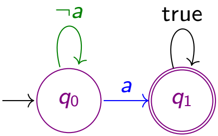
set of atomic propositions $AP = \{a, b\}$

NBA for LT properties



“infinitely often a and always $a \vee b$”

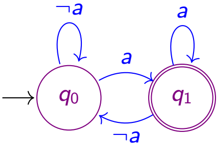
$$\hat{=} \quad ((a \vee b)^*.a)^\omega$$
$$\equiv \quad ((\neg a \wedge b)^*.a)^\omega$$
$$\equiv \quad (b^*.a)^\omega$$



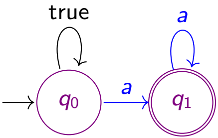
“eventually a”

$$\hat{=} \quad (\neg a)^* a (\text{true})^\omega$$

NBA for LT properties

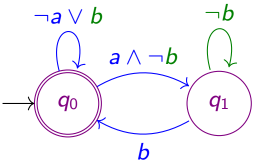


"infinitely often a"
$\left((\neg a)^*.a\right)^\omega$



"eventually always a"
$(\text{true})^*.(a)^\omega$

NBA for LT properties



"everytime a is true then b has to be true eventually"

From NBA to ω-Regular Expressions

Claim

For each NBA $\mathcal{A}$ there is an ω-regular expression γ with $\mathcal{L}_\omega(\mathcal{A}) = \mathcal{L}_\omega(\gamma)$.

Proof

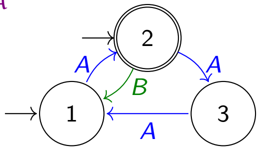
Let $\mathcal{A}$ be an NBA $(Q, \Sigma, \delta, Q_0, F)$ and $q, p \in Q$. Let $\mathcal{A}_{q,p}$ be the NFA $(Q, \Sigma, \delta, q, \{p\})$. Then

$$\mathcal{L}_\omega(\mathcal{A}) = \bigcup_{q \in Q_0} \bigcup_{p \in F} \mathcal{L}(\mathcal{A}_{q,p})(\mathcal{L}(\mathcal{A}_{p,p}) \setminus \{\varepsilon\})^\omega$$

is ω-regular as $\mathcal{L}(\mathcal{A}_{q,p})$ and $\mathcal{L}(\mathcal{A}_{p,p}) \setminus \{\varepsilon\}$ are regular.

From NBA to ω-Regular Expressions

NBA $\mathcal{A}$

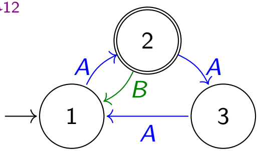


$$\mathcal{L}_\omega(\mathcal{A}) = L_{12}(L'_{22})^\omega \cup L_{22}(L'_{22})^\omega$$
$$L_{12} = \mathcal{L}(\mathcal{A}_{12})$$
$$L_{22} = \mathcal{L}(\mathcal{A}_{22})$$
$$L'_{22} = \mathcal{L}(\mathcal{A}_{22}) \setminus \{\varepsilon\}$$

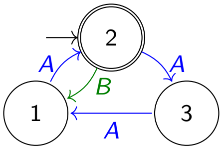
$$L_{12} \mathrel{\hat{=}} A.(B.A + A.A.A)^*$$

$$L_{22} \mathrel{\hat{=}} (B.A + A.A.A)^*$$

NFA $\mathcal{A}_{12}$

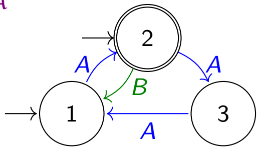


NFA $\mathcal{A}_{22}$



From NBA to ω-Regular Expressions

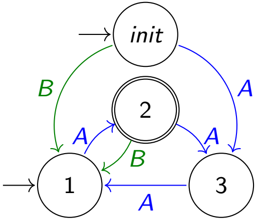
NBA $\mathcal{A}$



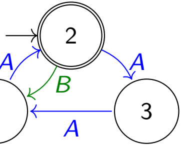
$$\mathcal{L}_\omega(\mathcal{A}) = L_{12}(L'_{22})^\omega \cup L_{22}(L'_{22})^\omega$$
$$L_{12} = \mathcal{L}(\mathcal{A}_{12})$$
$$L_{22} = \mathcal{L}(\mathcal{A}_{22})$$
$$L'_{22} = \mathcal{L}(\mathcal{A}_{12}) \setminus \{\varepsilon\}$$

$$L'_{22} \mathrel{\hat{=}} (B.A + A.A.A)^+$$

$$L_{22} \mathrel{\hat{=}} (B.A + A.A.A)^*$$

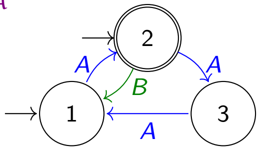


NFA $\mathcal{A}_{22}$



From NBA to ω-Regular Expressions

NBA $\mathcal{A}$



$$\mathcal{L}_\omega(\mathcal{A}) = L_{12}(L'_{22})^\omega \cup L_{22}(L'_{22})^\omega$$
$$L_{12} = \mathcal{L}(\mathcal{A}_{12})$$
$$L_{22} = \mathcal{L}(\mathcal{A}_{22})$$
$$L'_{22} = \mathcal{L}(\mathcal{A}_{12}) \setminus \{\varepsilon\}$$

language of $\mathcal{A}$:

$$\begin{aligned} & A.(B.A + A.A.A)^\omega \\ & +(B.A + A.A.A)^\omega \\ = {} & (A + \varepsilon).(B.A + A.A.A)^\omega \end{aligned}$$

From ω-Regular Expressions to NBA

Claim

For each ω-regular expression

$$\gamma = \alpha_1.\beta_1^\omega + \ldots + \alpha_n.\beta_n^\omega$$

there exists an NBA $\mathcal{A}$ with $\mathcal{L}_\omega(\mathcal{A}) = \mathcal{L}_\omega(\gamma)$.

Proof

Consider NFA $\mathcal{A}_i$ for α_i and $\mathcal{B}_i$ for β_i.

- construct NBA $\mathcal{B}_i^\omega$ for β_i^ω
- construct NBA $\mathcal{C}_i$ for $\alpha_i.\beta_i^\omega$
- construct NBA for $\bigcup_{1 \leq i \leq n} \mathcal{L}_\omega(\mathcal{C}_i)$

Equivalence of ω-Regular Expressions and NBA

Summary: equivalence of ω-regular expressions and NBA

- For each NBA $\mathcal{A}$ there exists an ω-regular expression γ with $\mathcal{L}_\omega(\mathcal{A}) = \mathcal{L}_\omega(\gamma)$
- For each ω-regular expression γ there exists an $\mathcal{A}$ with $\mathcal{L}_\omega(\mathcal{A}) = \mathcal{L}_\omega(\gamma)$

Exercise

Depict an NBA for the language described by the ω-regular expression

$$(ab + c)^*((aa + b)c)^\omega + (a^* c)^\omega.$$

Simplification of Regular Expressions

Here are a few laws that can be used to simplify regular expressions: for regular expressions α, β, and γ

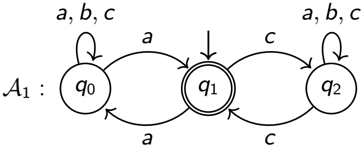
$$\begin{aligned}
\alpha + (\beta + \gamma) &= (\alpha + \beta) + \gamma \\
\alpha + \beta &= \beta + \alpha \\
\alpha + \varnothing &= \alpha \\
\alpha + \alpha &= \alpha \\
\alpha(\beta\gamma) &= (\alpha\beta)\gamma \\
\varepsilon\alpha = \alpha\varepsilon &= \alpha \\
\alpha(\beta + \gamma) &= \alpha\beta + \alpha\gamma \\
(\alpha + \beta)\gamma &= \alpha\gamma + \beta\gamma \\
\varnothing\alpha = \alpha\varnothing &= \varnothing \\
\varepsilon + \alpha\alpha^* &= \alpha^* \\
\varepsilon + \alpha^*\alpha &= \alpha^*
\end{aligned}$$

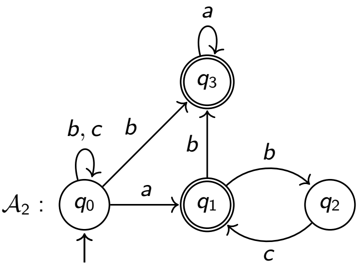
$$\begin{aligned}
(\alpha\beta)^*\alpha &= \alpha(\beta\alpha)^* \\
(\alpha^*\beta)^*\alpha^* &= (\alpha + \beta)^* \\
\alpha^*(\beta\alpha^*)^* &= (\alpha + \beta)^* \\
(\varepsilon + \alpha)^* &= \alpha^* \\
\alpha\alpha^* &= \alpha^*\alpha \\
(\alpha^*\beta)^* &= \varepsilon + (\alpha + \beta)^*\beta
\end{aligned}$$

Exercise 6: NBA

Question 1

Consider the following NBA $\mathcal{A}_1$ and $\mathcal{A}_2$ over the alphabet $\Sigma = \{a, b, c\}$:

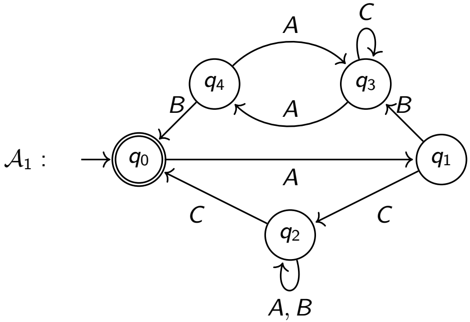


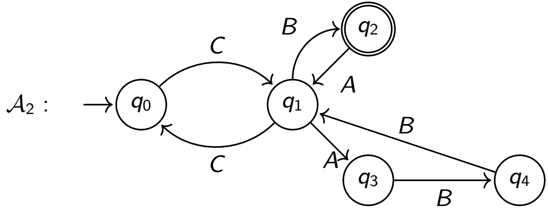


Find ω-regular expressions for the languages accepted by $\mathcal{A}_1$ and $\mathcal{A}_2$.

Question 2

Consider the NFA $\mathcal{A}_1$ and $\mathcal{A}_2$:





Construct an NBA for the language $\mathcal{L}(\mathcal{A}_1)\mathcal{L}(\mathcal{A}_2)^\omega$.

3.3 Linear Temporal Logic

Recap: Propositional Formulas

- Syntax: Let AP be a finite set of atomic propositions. We use $\neg$, $\vee$ construct propositional formulas. The set of propositional formulas over AP is defined inductively by the rules:
  - If $p \in AP$, then p is a propositional formula;
  - If φ and ψ are propositional formulas, then

    $$\neg\varphi \quad \text{and} \quad \varphi \vee \psi$$

    are propositional formulas.
- Semantics: A set $P \subseteq AP$ satisfies a propositional formula ψ, if (P, ψ) is an element of the satisfaction relation $\models$, which is defined as follows. Let $p \in AP$ and φ, ψ be two propositional formulas
  - $P \models p$ iff $p \in P$;
  - $P \models \neg\varphi$ iff $P \not\models \varphi$;
  - $P \models \varphi \vee \psi$ iff $P \models \varphi$ or $P \models \psi$ holds. ■

Abbreviations

- true $:= \neg p \vee p$ (for some $p \in AP$)
- $\varphi \rightarrow \psi := \neg\varphi \vee \psi$
- $\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$

Examples

Let $AP = \{a, b, c\}$, then

- $\{a, b, c\} \models a$
- $\{a, b\} \models a$
- $\varnothing \models \neg c$
- $\{a\} \models a \wedge \neg b$

Definition: Linear Temporal Logic (Syntax)

Let AP be a finite set of atomic propositions, i.e., a finite set of boolean variables. We use $\neg$, $\vee$, $\bigcirc$ and U to denote the logic and modal operators. The set of linear temporal logic (LTL) formulas on AP is defined inductively by the rules

- If $p \in AP$, then p is an LTL formula;
- If φ and ψ are LTL formulas, then

$$\neg\varphi, \quad \bigcirc\varphi, \quad \varphi \vee \psi \quad \text{and} \quad \varphi \mathsf{U} \psi$$

  are LTL formulas. ■

Abbreviations

- $\mathsf{true} := \neg p \vee p$ (for some $p \in AP$)
- $\varphi \rightarrow \psi := \neg\varphi \vee \psi$
- $\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$
- $\Diamond\varphi := \mathsf{true}\mathsf{U}\varphi$
- $\Box\varphi := \neg\Diamond\neg\varphi$

Naming

- $\bigcirc$ is the next operator
- U is the until operator
- $\Diamond$ is the eventually/finally operator
- $\Box$ is the always/globally operator

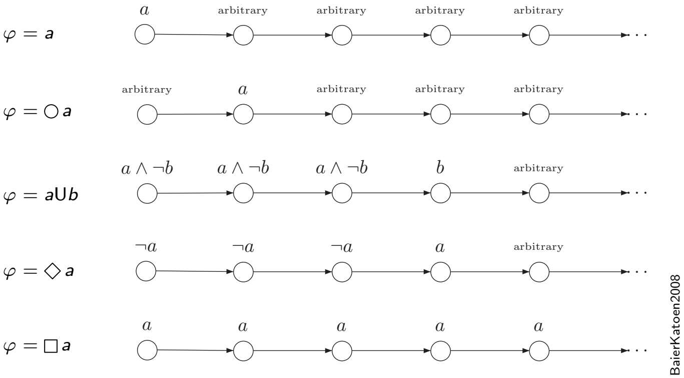
Examples

Let $AP = \{a, b\}$, then

- $(a \vee b)\mathsf{U}(\neg a)$ is an LTL formula
- $\mathsf{U}(\neg a)$ is not an LTL formula
- $a \bigcirc b$ is not an LTL formula
- $a \Box b$ is not an LTL formula

LTL formulas stand for properties of sequences

Let φ be an LTL formula over $AP = \{a, b\}$ and $w : [0; \infty] \rightrightarrows AP$. The figure shows the intuitive idea behind "w satisfies φ".



Definition: LTL Semantics over Infinite Sequences

Let φ be an LTL formula over AP and $w : [0;\infty[\rightrightarrows AP$. We say that w satisfies φ at time $t \in \mathbb{N}$, denoted by

$$w, t \models \varphi$$

if (w, t, φ) is an element of the satisfaction relation $\models$. Let $p \in AP$ and φ, ψ be LTL formulas over AP, then satisfaction relation is defined inductively by

- $w, t \models p$ iff $p \in w(t)$;
- $w, t \models \neg\varphi$ iff $w, t \not\models \varphi$;
- $w, t \models \varphi \vee \psi$ iff $w, t \models \varphi$ or $w, t \models \psi$ holds;
- $w, t \models \bigcirc\varphi$ iff $w, t+1 \models \varphi$;
- $w, t \models \varphi \mathsf{U} \psi$ iff $\exists_{t' \in [t;\infty[}(w, t' \models \psi)$ and $\forall_{t'' \in [t;t'[}(w, t'' \models \varphi)$.

We say that w satisfies φ if w satisfies φ at time $t = 0$ and

$$w \models \varphi \quad \text{is used for} \quad w, 0 \models \varphi.$$

The set of all satisfying sequences is denoted by

$$P(\varphi) = \{w : [0;\infty[\rightrightarrows AP \mid w \models \varphi\}.$$

Example: LTL Semantics

Let $AP = \{a, b, c\}$ and

$$w = \{a, b\}\{a, c\}\{b\}\{c\}\{a\}^{\omega}$$

Are the following statements true?

- $w \models a$
- $w, 2 \models a$
- $w \not\models c$
- $w \not\models \bigcirc c$
- $w \models b \mathrm{U} c$
- $w \models c \mathrm{U} b$
- $w, 2 \models c \mathrm{U} a$

Exercise 7: Derived Symbol Semantics

Consider the following LTL formulas φ over AP, with $p \in AP$

1. $\varphi = \Diamond p$;
2. $\varphi = \Box p$;
3. $\varphi = \Diamond\Box p$;
4. $\varphi = \Box\Diamond p$.

Provide the conditions on a sequence w such that $w \in P(\varphi)$.

Example: Printer Specifications

Atomic propositions

- j_i job $i \in \{1, 2\}$ submitted;
- p_i job $i \in \{1, 2\}$ printed.

Specification

- Two jobs are not printed at the same time

$$\Box \neg (p_1 \wedge p_2)$$

- Every job is eventually printed

$$\Box \left((j_1 \rightarrow \Diamond p_1) \wedge (j_2 \rightarrow \Diamond p_2) \right)$$

Example: Robot Task Planning

Description

A robot moves in a factory environment. The robot should move parts from the stockroom to the two stations, while avoiding obstacles.

$$\Box\Diamond\, stockroom \land \Box\Diamond\, station\_1 \land \Box\Diamond\, station\_2$$

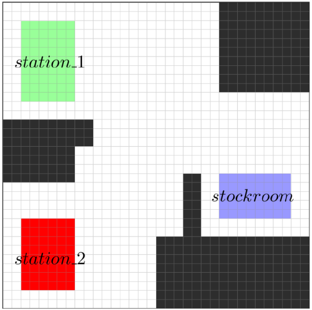
$$\Box\Diamond\, \neg s1\_occupied \land \Box\Diamond\, \neg s2\_occupied$$

$$\Box(s1\_occupied \implies \bigcirc\neg station\_1)$$

$$\Box(s2\_occupied \implies \bigcirc\neg station\_2)$$

$$\Box(station\_1 \implies \neg s1\_occupied)$$

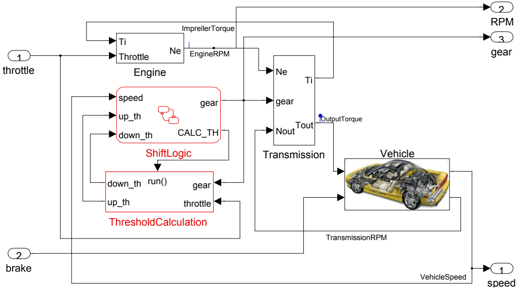
$$\Box(station\_2 \implies \neg s2\_occupied)$$



Example: Automatic Transmission Controller

A Simulink model of an automatic transmission system.

- Inputs: Throttle, brake
- Outputs: RPM, gear, speed



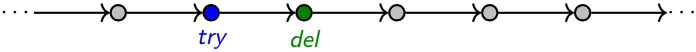
X. Jin et al., HSCC, 2013.

Some specifications

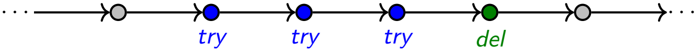
- The velocity and RPM are always below some thresholds: $\square(\text{speed}_{\leq 200} \wedge \text{RPM}_{\leq 4500})$
- Whenever the system shifts to gear 2, it dwells in gear 2 for at least one time step: $\square\left((\neg \text{g2} \wedge \bigcirc \text{g2}) \rightarrow \bigcirc\bigcirc \text{g2}\right)$

Examples

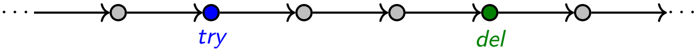
$\Box(\mathit{try\_to\_send} \rightarrow \bigcirc\, \mathit{delivered})$



$\Box(\mathit{try\_to\_send} \rightarrow \mathit{try\_to\_send}\, \mathsf{U}\, \mathit{delivered})$

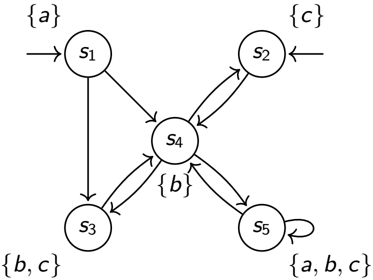


$\Box(\mathit{try\_to\_send} \rightarrow \Diamond\, \mathit{delivered})$



Exercise

Consider the system S over the set of atomic propositions $AP = \{a, b, c\}$:



Decide for each of the LTL formulae φ_i below, whether $S \models \varphi_i$ holds. Justify your answers! If $S \not\models \varphi_i$, provide a state path (a.k.a. state run) $\pi = x_0 x_1 x_2 \ldots$ such that $\text{trace}(\pi) \not\models \varphi_i$, where $\text{trace}(\pi) := y(x_0)y(x_1)\ldots$:

1. $\varphi_a = \Diamond \Box c$
2. $\varphi_b = \Box \Diamond c$
3. $\varphi_c = \bigcirc \neg c \rightarrow \bigcirc \bigcirc c$
4. $\varphi_d = \Box a$
5. $\varphi_e = a \mathsf{U} \Box (b \vee c)$
6. $\varphi_f = (\bigcirc \bigcirc b) \mathsf{U} (b \vee c)$

Question1

Prove the following equivalence or provide a counterexample that illustrates that the formula on the left and the formula on the right are not equivalent.

- $\Diamond\Box\varphi_1 \wedge \Diamond\Box\varphi_2 = \Diamond(\Box\varphi_1 \wedge \Box\varphi_2)$

Question2 Provide an NBA for each of the following LTL formula where $\Sigma = \{a, b\}$:

- $\Box(a \vee \neg\bigcirc b)$
- $\bigcirc\bigcirc(a \vee \Diamond\Box b)$
- $\Diamond\Box a$

Question3

Consider an elevator that services 3 floors numbered 1 through 3. There is an elevator door at each floor with a call-button and an indicator light that signals whether or not the elevator has been called. We use the following propositions to reason about the system.

- $open_i$: the door on floor i is unlocked, $i \in \{1, 2, 3\}$
- $floor_i$: the cabin is located on floor i (not moving), $i \in \{1, 2, 3\}$
- req_i: the ith floor is requested, $i \in \{1, 2, 3\}$

State the LTL formulae for the following informal properties.

- The doors are "safe", i.e., a floor door is never open if the cabin is not present at a given floor
- A requested floor will be served sometime.
- Again and again the lift returns to floor 1.
- When the top floor is requested, the lift serves it immediately and does not stop on the way there.

Definition: Satisfaction, Realizability of LTL Formulas

- Let $S = (X, X_0, U, Y, F, H)$ be a system.
- Let φ be an LTL formula over AP.
- Let $L : Y \rightrightarrows AP$ be a strict map (the labeling function).

Every element of the behavior $y \in \mathcal{B}(S)$ induces a sequence $w \in (2^{AP})^\infty$

$$w(t) = L(y(t)).$$

$P(\varphi)$ is a property over 2^{AP}. We use L to define a property $P_L(\varphi)$ over Y:

- If $P(\varphi) \subseteq (2^{AP})^\omega$, then

$$P_L(\varphi) = \{y \in (Y)^\omega \mid L(y) \in P(\varphi)\}$$

We say that y satisfies φ if $y \in P_L(\varphi)$.

We say that

- S satisfies φ (under L), if S satisfies $P_L(\varphi)$;
- φ is realizable on S (under L), if $P_L(\varphi)$ is realizable on S.

Examples of labeling functions

- Robot task planning
  - Consider a "unicycle" robot. The state alphabet $X = \mathbb{R}^3$ is given by the position (x_1, x_2) and orientation $x_3 = \theta$ of the mobile robot.
  - Some atomic propositions are given by $AP = \{\mathrm{stockroom}, \mathrm{station}_1, \mathrm{station}_2, \dots\}$. We define

$$\mathrm{stockroom} = \begin{cases} 1 & \text{if the position } (x_1, x_2) \text{ is at the stockroom coordinates} \\ 0 & \text{otherwise.} \end{cases}$$

  - We use the labeling function L to map from $x = (x_1, x_2, x_3)$ to the atomic propositions $\mathrm{stockroom}$, $\mathrm{station}_1$, $\mathrm{station}_2$. The labeling function contains $\mathrm{stockroom} \in L(x)$ if the robot is at the stockroom coordinates.
- Automatic transimission controller
  - Outputs $X = \{\mathrm{gear}, \mathrm{velocity}, \mathrm{RPM}, \dots\}$
  - Atomic propositions $AP = \{\mathrm{speed}_{\leq 200}, \mathrm{RPM}_{\leq 4500}, g1, g2, \dots\}$
  - Labeling function $\left(\mathrm{speed}_{\leq 200} \in L(x)\right) \iff \mathrm{velocity} \leq 200$

Signal Temporal Logic (STL)

From LTL to STL:
Extension of LTL with real-time and real-valued constraints

LTL (Linear Temporal Logic)
$\Box(a \implies \Diamond b)$
Boolean predicates (atomic propositions), discrete-time

MITL (Metric Interval Temporal Logic)
$\Box(a \implies \Diamond_{[0,0.5s]} b)$
Boolean predicates, real-time

STL (Signal Temporal logic)
$\Box(x(t) > 0 \implies \Diamond_{[0,0.5s]} y(t) > 0)$
Predicates over real values, real-time

STL Semantics

The satisfaction of a formula φ by a signal $\mathbf{x} = (x_1, \ldots, x_n)$ at time t is

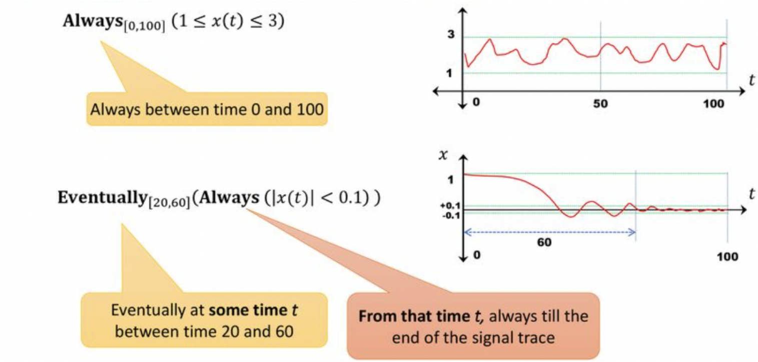
$$\begin{array}{lcl}
(\mathbf{x}, t) \models \mu & \Leftrightarrow & f(x_1[t], \ldots, x_n[t]) > 0 \\
(\mathbf{x}, t) \models \varphi \wedge \psi & \Leftrightarrow & (x, t) \models \varphi \wedge (x, t) \models \psi \\
(\mathbf{x}, t) \models \neg\varphi & \Leftrightarrow & \neg((x, t) \models \varphi) \\
(\mathbf{x}, t) \models \varphi \, \mathcal{U}_{[a,b]} \, \psi & \Leftrightarrow & \exists t' \in [t + a, t + b] \text{ such that } (x, t') \models \psi \wedge \\
 & & \qquad\qquad \forall t'' \in [t, t'], \; (x, t'') \models \varphi\}
\end{array}$$

Similar to LTL, the eventually and always operator can be derived from the above four operators.

μ is predicate obtained after evaluation of function $f : \mathbb{R}^n \longleftarrow \mathbb{R}$ defined as:

$$\begin{aligned}
\mu &= \text{True} \quad \text{if } f(x) \geq 0 \\
\mu &= \text{False} \quad \text{if } f(x) < 0
\end{aligned}$$

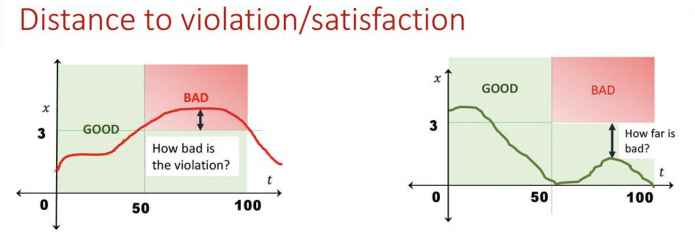
STL Example



$\textbf{Always}_{[0,100]}\ (1 \leq x(t) \leq 3)$
Always between time 0 and 100
3
1
t
0
50
100
$\textbf{Eventually}_{[20,60]}(\textbf{Always}\ (|x(t)| < 0.1)\ )$
Eventually at **some time** ***t*** between time 20 and 60
From that time ***t*****,** always till the end of the signal trace
x
1
+0.1
-0.1
t
60
0
100

Robust STL symantics

$$\rho^{\mu}(\boldsymbol{x}, t) := h(\boldsymbol{x}(t))$$
$$\rho^{\neg\phi}(\boldsymbol{x}, t) := -\rho^{\phi}(\boldsymbol{x}, t)$$
$$\rho^{\phi_1 \wedge \phi_2}(\boldsymbol{x}, t) := \min\left(\rho^{\phi_1}(\boldsymbol{x}, t), \rho^{\phi_2}(\boldsymbol{x}, t)\right)$$
$$\rho^{F_{[a,b]}\phi}(\boldsymbol{x}, t) := \max_{t_1 \in [t+a, t+b]} \rho^{\phi}(\boldsymbol{x}, t_1)$$
$$\rho^{G_{[a,b]}\phi}(\boldsymbol{x}, t) := \min_{t_1 \in [t+a, t+b]} \rho^{\phi}(\boldsymbol{x}, t_1).$$



$$\mathbf{G}_{[50,100]}(x(t) < 3)$$

4 Verification for Autonomous Systems

4.1 Checking Regular Safety Properties (Finite Systems)

4.2 Checking ω-Regular Properties (Finite Systems)

4.3 Barrier Certificates (Infinite Systems)