CS 228 : Logic in Computer Science

Krishna. S

Transition System Semantics $TS \models \varphi$

Let $TS = (S, S_0, \rightarrow, AP, L)$ be a transition system, and φ an LTL formula over AP

- For an infinite path fragment π of TS,

$$\pi \models \varphi \text{ iff } \mathit{trace}(\pi) \models \varphi$$

Transition System Semantics $TS \models \varphi$

Let $TS = (S, S_0, \rightarrow, AP, L)$ be a transition system, and φ an LTL formula over AP

- For an infinite path fragment π of TS,

$$\pi \models \varphi \text{ iff } \mathit{trace}(\pi) \models \varphi$$

- For $s \in S$,

$$s \models \varphi \text{ iff } \forall \pi \in \mathit{Paths}(s), \pi \models \varphi$$

Transition System Semantics $TS \models \varphi$

Let $TS = (S, S_0, \rightarrow, AP, L)$ be a transition system, and φ an LTL formula over AP

- For an infinite path fragment π of TS,

$$\pi \models \varphi \text{ iff } \mathit{trace}(\pi) \models \varphi$$

- For $s \in S$,

$$s \models \varphi \text{ iff } \forall \pi \in \mathit{Paths}(s), \pi \models \varphi$$

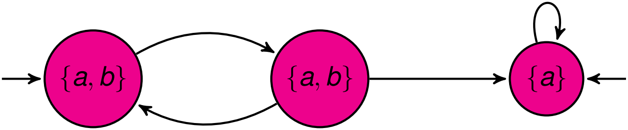
- $TS \models \varphi$ iff $\mathit{Traces}(TS) \subseteq L(\varphi)$

Transition System Semantics $TS \models \varphi$

Assume all states in TS are reachable from S_0.

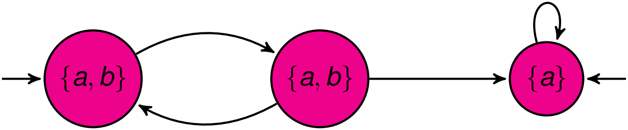
- $TS \models \varphi$ iff $TS \models L(\varphi)$ iff $Traces(TS) \subseteq L(\varphi)$
- $TS \models L(\varphi)$ iff $\pi \models \varphi \; \forall \pi \in Paths(TS)$
- $\pi \models \varphi \; \forall \pi \in Paths(TS)$ iff $s_0 \models \varphi \; \forall s_0 \in S_0$

Example



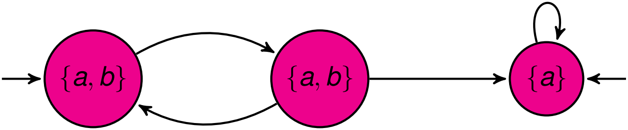
- $TS \models \Box a$,

Example



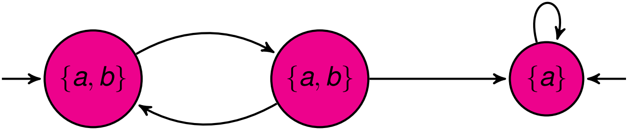
- $TS \models \Box a$,
- $TS \not\models \bigcirc(a \wedge b)$

Example



- $TS \models \square a$,
- $TS \not\models \bigcirc(a \wedge b)$
- $TS \not\models (b \,\mathsf{U}(a \wedge \neg b))$

Example



- $TS \models \Box a$,
- $TS \not\models \bigcirc(a \wedge b)$
- $TS \not\models (b \,\mathsf{U}(a \wedge \neg b))$
- $TS \models \Box(\neg b \rightarrow \Box(a \wedge \neg b))$

More Semantics

- For paths π, $\pi \models \varphi$ iff $\pi \not\models \neg\varphi$

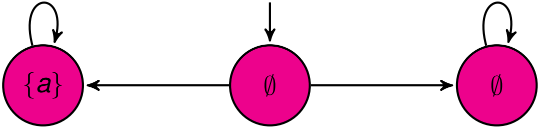
More Semantics

- For paths π, $\pi \models \varphi$ iff $\pi \not\models \neg\varphi$
  $trace(\pi) \in L(\varphi)$ iff $trace(\pi) \notin L(\neg\varphi) = \overline{L(\varphi)}$
- $TS \not\models \varphi$ iff $TS \models \neg\varphi$?
  - $TS \models \neg\varphi \rightarrow \forall$ paths π of TS, $\pi \models \neg\varphi$
  - Thus, $\forall\pi$, $\pi \not\models \varphi$. Hence, $TS \not\models \varphi$

More Semantics

- For paths π, $\pi \models \varphi$ iff $\pi \not\models \neg\varphi$
  $trace(\pi) \in L(\varphi)$ iff $trace(\pi) \notin L(\neg\varphi) = \overline{L(\varphi)}$
- $TS \not\models \varphi$ iff $TS \models \neg\varphi$?
  - $TS \models \neg\varphi \rightarrow \forall$ paths π of TS, $\pi \models \neg\varphi$
  - Thus, $\forall \pi$, $\pi \not\models \varphi$. Hence, $TS \not\models \varphi$
  - Now assume $TS \not\models \varphi$
  - Then $\exists$ some path π in TS such that $\pi \models \neg\varphi$
  - However, there could be another path π' such that $\pi' \models \varphi$
  - Then $TS \not\models \neg\varphi$ as well
- Thus, $TS \not\models \varphi \not\equiv TS \models \neg\varphi$.

An Example



$TS \not\models \Diamond a$ and $TS \not\models \Box \neg a$

Equivalence of LTL Formulae

Equivalence

φ and ψ are equivalent ($\varphi \equiv \psi$) iff $L(\varphi) = L(\psi)$.

Expansion Laws

- $\varphi \,\mathsf{U} \psi \equiv \psi \vee (\varphi \wedge \bigcirc(\varphi \,\mathsf{U} \psi))$
- $\Diamond\varphi \equiv \varphi \vee \bigcirc\Diamond\varphi$
- $\Box\varphi \equiv \varphi \wedge \bigcirc\Box\varphi$

Equivalence of LTL Formulae

φ and ψ are equivalent iff $L(\varphi) = L(\psi)$.

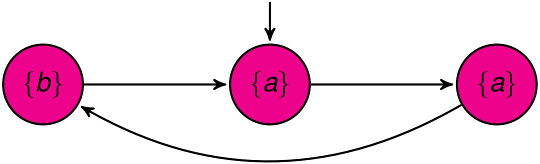
Equivalence of LTL Formulae

φ and ψ are equivalent iff $L(\varphi) = L(\psi)$.

Distribution

$\bigcirc(\varphi \vee \psi) \equiv \bigcirc\varphi \vee \bigcirc\psi$,
$\bigcirc(\varphi \wedge \psi) \equiv \bigcirc\varphi \wedge \bigcirc\psi$,
$\bigcirc(\varphi \,\mathsf{U} \psi) \equiv (\bigcirc\varphi) \,\mathsf{U} (\bigcirc\psi)$,
$\Diamond(\varphi \vee \psi) \equiv \Diamond\varphi \vee \Diamond\psi$,
$\Box(\varphi \wedge \psi) \equiv \Box\varphi \wedge \Box\psi$

Equivalence of LTL Formulae



$TS \models \Diamond a \wedge \Diamond b$, $TS \not\models \Diamond(a \wedge b)$

$TS \models \Box(a \vee b)$, $TS \not\models \Box a \vee \Box b$

Satisfiability, Model Checking of LTL

Two Questions

Given transition system TS, and an LTL formula φ. Does $TS \models \varphi$?
Given an LTL formula φ, is $L(\varphi) = \emptyset$?

How we go about this:

- Translate φ into an automaton A_φ that accepts infinite words such that $L(A_\varphi) = L(\varphi)$.
- Check for emptiness of A_φ to check satisfiability of φ.
- Check if $TS \cap \overline{A_\varphi}$ is empty, to answer the model-checking problem.

Notations for Infinite Words

- Σ is a finite alphabet
- Σ^* set of finite words over Σ
- An infinite word is written as $\alpha = \alpha(0)\alpha(1)\alpha(2)\ldots$, where $\alpha(i) \in \Sigma$
- Such words are called ω-words
- $Inf(\alpha) = \{a \in \Sigma \mid \alpha(i) = a \text{ for infinitely many } i\}$. $Inf(\alpha)$ gives the set of symbols occurring infinitely often in α.

ω-automata

An ω-automaton is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, Acc)$ where

- Q is a finite set of states
- Σ is a finite alphabet
- $\delta : Q \times \Sigma \to 2^Q$ is a state transition function (if non-deterministic, otherwise, $\delta : Q \times \Sigma \to Q$)
- $q_0 \in Q$ is an initial state and Acc is an acceptance condition

ω-automata

An ω-automaton is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, Acc)$ where

- Q is a finite set of states
- Σ is a finite alphabet
- $\delta : Q \times \Sigma \to 2^Q$ is a state transition function (if non-deterministic, otherwise, $\delta : Q \times \Sigma \to Q$)
- $q_0 \in Q$ is an initial state and Acc is an acceptance condition

Run

A run ρ of $\mathcal{A}$ on an ω-word $\alpha = a_1 a_2 \cdots \in \Sigma^\omega$ is an infinite state sequence $\rho(0)\rho(1)\rho(2)\ldots$ such that

- $\rho(0) = q_0$,
- $\rho(i) = \delta(\rho(i-1), a_i)$ if $\mathcal{A}$ is deterministic,
- $\rho(i) \in \delta(\rho(i-1), a_i)$ if $\mathcal{A}$ is non-deterministic,

ω-automata

An ω-automaton is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, Acc)$ where

- Q is a finite set of states
- Σ is a finite alphabet
- $\delta : Q \times \Sigma \rightarrow 2^Q$ is a state transition function (if non-deterministic, otherwise, $\delta : Q \times \Sigma \rightarrow Q$)
- $q_0 \in Q$ is an initial state and Acc is an acceptance condition

Run

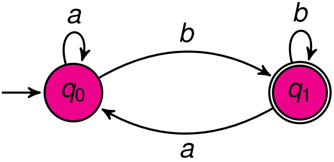
A run ρ of $\mathcal{A}$ on an ω-word $\alpha = a_1 a_2 \cdots \in \Sigma^\omega$ is an infinite state sequence $\rho(0)\rho(1)\rho(2)\ldots$ such that

- $\rho(0) = q_0$,
- $\rho(i) = \delta(\rho(i-1), a_i)$ if $\mathcal{A}$ is deterministic,
- $\rho(i) \in \delta(\rho(i-1), a_i)$ if $\mathcal{A}$ is non-deterministic,

Büchi Acceptance

For Büchi Acceptance, Acc is specified as a set of states, $G \subseteq Q$. The ω-word α is accepted if there is a run ρ of α such that $Inf(\rho) \cap G \neq \emptyset$.

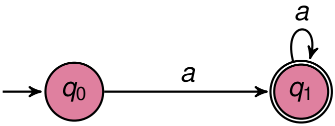
ω-Automata with Büchi Acceptance

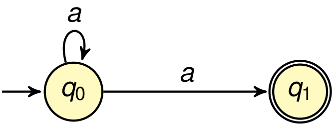


$L(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \alpha$ has a run ρ such that $Inf(\rho) \cap G \neq \emptyset$

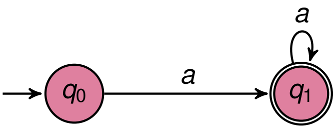
Language accepted=Infinitely many *b*'s.

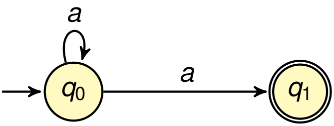
Comparing NFA and NBA

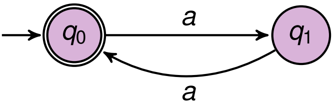


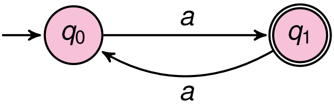


Comparing NFA and NBA

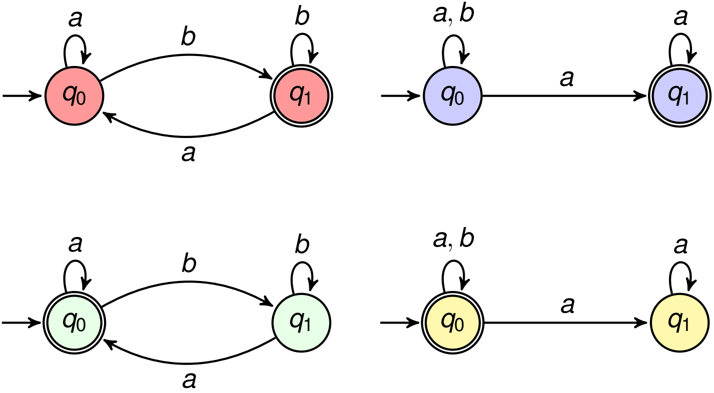








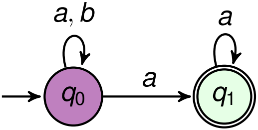
ω-Automata with Büchi Acceptance



- Left (T-B): Inf many *b*'s, Inf many *a*'s
- Right (T-B): Finitely many *b*'s, $(a + b)^\omega$

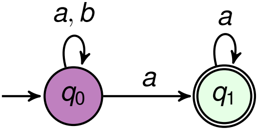
NBA and DBA

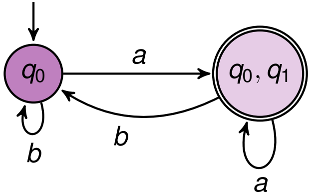
- Is every DBA as expressible as a NBA, like in the case of DFA and NFA?
- Can we do subset construction on NBA and obtain DBA?



NBA and DBA

- Is every DBA as expressible as a NBA, like in the case of DFA and NFA?
- Can we do subset construction on NBA and obtain DBA?



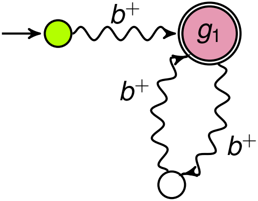


NBA and DBA

There does not exist a deterministic Büchi automata capturing the language finitely many a's.

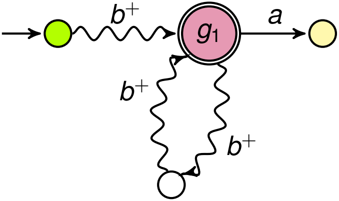
NBA and DBA

There does not exist a deterministic Büchi automata capturing the language finitely many *a*'s.



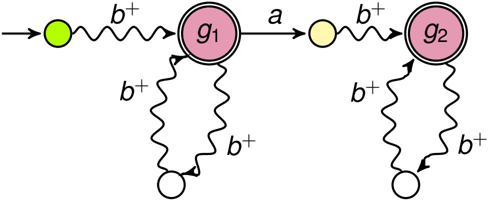
NBA and DBA

There does not exist a deterministic Büchi automata capturing the language finitely many a's.



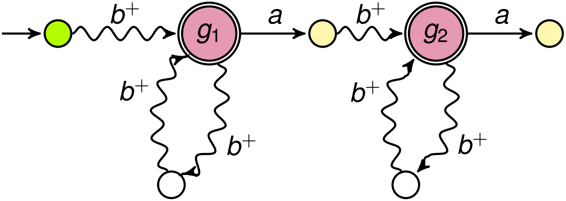
NBA and DBA

There does not exist a deterministic Büchi automata capturing the language finitely many *a*'s.



NBA and DBA

There does not exist a deterministic Büchi automata capturing the language finitely many a's.



NBA and DBA

There does not exist a deterministic Büchi automata capturing the language finitely many *a*'s.