CS 228 : Logic in Computer Science

Krishna. S

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 $\rho(0)\rho(1)\rho(2)\dots$ 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

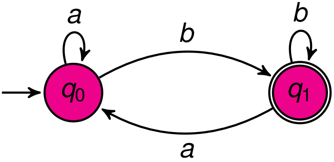
A run ρ of $\mathcal{A}$ on an ω-word $\alpha = a_1 a_2 \cdots \in \Sigma^\omega$ is an infinite state sequence $\rho(0)\rho(1)\rho(2)\ldots$ such that

- $\rho(0) = q_0$,
- $\rho(i) = \delta(\rho(i-1), a_i)$ 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 $Inf(\rho) \cap G \neq \emptyset$.

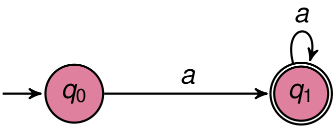
ω-Automata with Büchi Acceptance

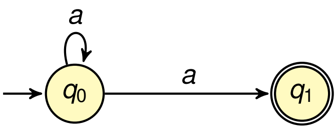


$L(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \alpha$ has a run ρ such that $\mathit{Inf}(\rho) \cap G \neq \emptyset$

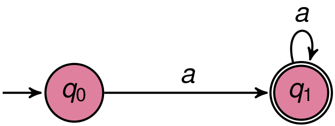
Language accepted=Infinitely many b's.

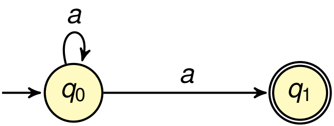
Comparing NFA and NBA

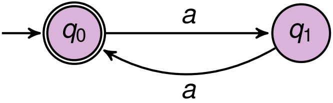


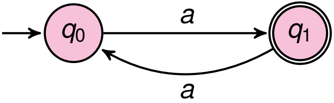


Comparing NFA and NBA









Some Facts

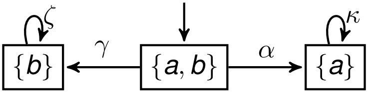
- Each LTL formula φ can be translated to an NBA A_φ such that $L(\varphi) = L(A_\varphi)$
- To check if the language accepted by an NBA is empty, it is enough to find if there exists a cycle, reachable from an initial state, containing a good state. This check can be done in time polynomial in the size of the NBA. This gives an algorithm for satisfiability checking of LTL.
- The class of languages accepted by an NBA are called ω-regular languages
- NBA are strictly more expressive than DBA
- ω-regular languages are closed under union, intersection and complementation

LTL ModelChecking

- Given transition system TS, and LTL formula φ, does $TS \models \varphi$?
- $Tr(TS) \subseteq L(\varphi)$ iff $Tr(TS) \cap \overline{L(\varphi)} = \emptyset$
- First construct NBA $A_{\neg\varphi}$ for $\neg\varphi$.
- Construct product of TS and $A_{\neg\varphi}$, obtaining a new TS, say TS'.
- Check some very simple property on TS', to check $TS \models \varphi$.

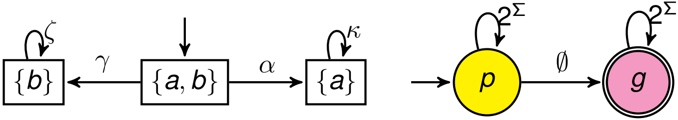
An Example $TS \models \varphi$

- Let $\varphi = \Box(a \vee b), \neg\varphi = \Diamond(\neg a \wedge \neg b)$
- Take TS and $A_{\neg\varphi}$, and check the intersection.



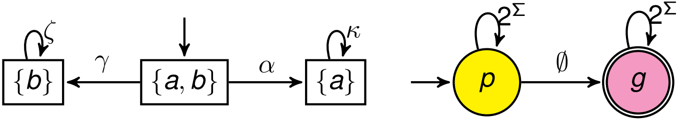
An Example $TS \models \varphi$

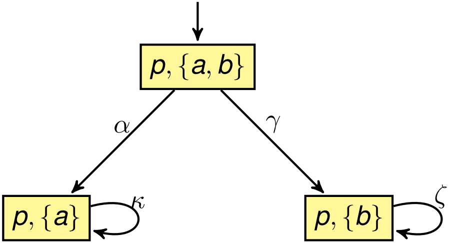
- Let $\varphi = \Box(a \vee b), \neg\varphi = \Diamond(\neg a \wedge \neg b)$
- Take TS and $A_{\neg\varphi}$, and check the intersection.



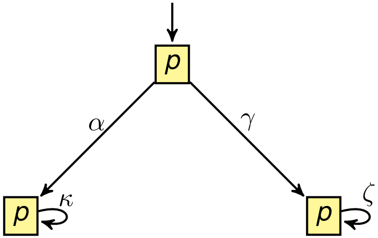
An Example $TS \models \varphi$

- Let $\varphi = \Box(a \vee b), \neg\varphi = \Diamond(\neg a \wedge \neg b)$
- Take TS and $A_{\neg\varphi}$, and check the intersection.





Changing the labeling to keep only states of NBA



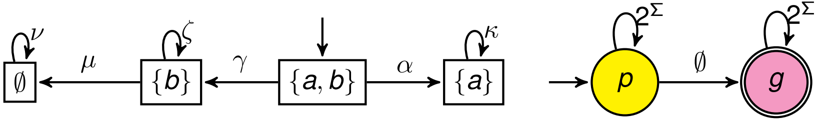
An Example : $TS \not\models \varphi$

- Let $\varphi = \Box(a \vee b), \neg\varphi = \Diamond(\neg a \wedge \neg b)$
- Take TS and $A_{\neg\varphi}$, and check the intersection.

$$\begin{tikzcd} \boxed{\emptyset} \arrow[loop above, "\nu"] & \boxed{\{b\}} \arrow[l, "\mu"'] \arrow[loop above, "\zeta"] & \boxed{\{a, b\}} \arrow[l, "\gamma"'] \arrow[r, "\alpha"] & \boxed{\{a\}} \arrow[loop above, "\kappa"] \end{tikzcd}$$

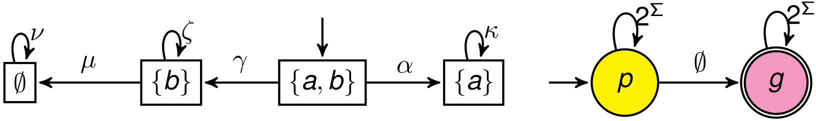
An Example : $TS \not\models \varphi$

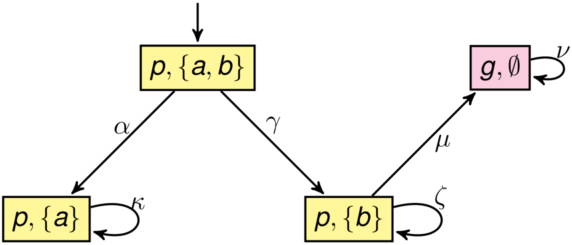
- Let $\varphi = \Box(a \vee b), \neg\varphi = \Diamond(\neg a \wedge \neg b)$
- Take TS and $A_{\neg\varphi}$, and check the intersection.



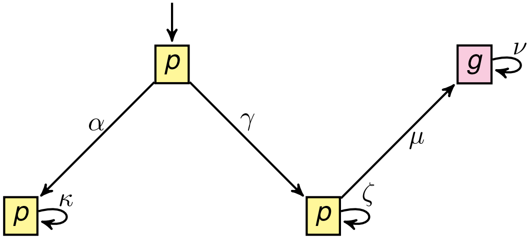
An Example : $TS \not\models \varphi$

- Let $\varphi = \Box(a \vee b), \neg\varphi = \Diamond(\neg a \wedge \neg b)$
- Take TS and $A_{\neg\varphi}$, and check the intersection.





Changing the labeling to keep only states of NBA



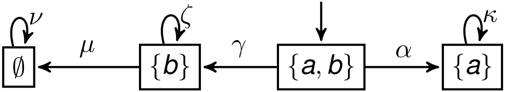
Product of TS and NBA

Given $TS = (S, Act, I, AP, L)$ and $\mathcal{A} = (Q, 2^{AP}, \delta, Q_0, G)$ NBA.
Define $TS \otimes \mathcal{A} = (S \times Q, Act, I', AP', L')$ such that

- $I' = \{(s_0, q) \mid s_0 \in I \text{ and } \exists q_0 \in Q_0, q_0 \stackrel{L(s_0)}{\rightarrow} q\}$
- $AP' = Q$, $L' : S \times Q \rightarrow 2^Q$ such that $L'((s, q)) = \{q\}$
- If $s \stackrel{\alpha}{\rightarrow} t$ and $q \stackrel{L(t)}{\rightarrow} p$, then $(s, q) \stackrel{\alpha}{\rightarrow} (t, p)$

Persistence Properties

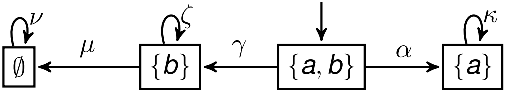
Let η be a propositional logic formula over AP. A persistence property P_{pers} has the form $\Diamond\Box\eta$. How will you check a persistence property on a TS?



- For example, $TS \not\models \Diamond\Box(a \vee b)$
- For example, $TS \models \Diamond\Box(a \vee (a \rightarrow b))$

Persistence Properties

Let η be a propositional logic formula over AP. A persistence property P_{pers} has the form $\Diamond\Box\eta$. How will you check a persistence property on a TS?



- For example, $TS \not\models \Diamond\Box(a \vee b)$
- For example, $TS \models \Diamond\Box(a \vee (a \rightarrow b))$
- $TS \not\models P_{pers}$ iff there is a reachable cycle in the TS containing a state with a label which satisfies $\neg\eta$.

LTL ModelChecking

- Given TS and LTL formula φ. Does $TS \models \varphi$?
- Construct $A_{\neg\varphi}$, and let $g_1, \ldots, g_n$ be the good states in $A_{\neg\varphi}$.
- Build $TS' = TS \otimes A_{\neg\varphi}$.
- The labels of TS' are the state names of $A_{\neg\varphi}$.
- Check if $TS' \models \Diamond\Box(\neg g_1 \wedge \ldots \neg g_n)$.

LTL ModelChecking

- Given TS and LTL formula φ. Does $TS \models \varphi$?
- Construct $A_{\neg\varphi}$, and let $g_1, \ldots, g_n$ be the good states in $A_{\neg\varphi}$.
- Build $TS' = TS \otimes A_{\neg\varphi}$.
- The labels of TS' are the state names of $A_{\neg\varphi}$.
- Check if $TS' \models \Diamond\Box(\neg g_1 \wedge \ldots \neg g_n)$.

ModelChecking LTL in TS = Check Persistence in TS'

The following are equivalent.

- $TS \models \varphi$
- $Tr(TS) \cap L(A_{\neg\varphi}) = \emptyset$
- $TS' \models \Diamond\Box(\neg g_1 \wedge \ldots \neg g_n)$.