Automata-Based Verification of LTL

Property Verification

- A property is a set of infinite words (language) $P \subseteq \left(2^{AP}\right)^{\omega}$
- For an LTL formula ϕ, we have $Word(\phi) = \{\sigma \in \left(2^{AP}\right)^{\omega} : \sigma \vDash \phi\}$
- To check whether or not $T \vDash \phi$, it suffices to check whether or not
  - $Trace(T) \subseteq Word(\phi)$; or
  - $Trace(T) \cap Word(\neg\phi) = \emptyset$
- How to efficiently represent $Word(\phi)$?
  - Need finite structure not to enumerate all strings (not possible)
  - Approach: using Automata to generate language

Finite Words & Regular Language

- **Alphabet** (event set) Σ, e.g., $\Sigma = \{a, b, c\}$
- **Finite word** (string): $w = w_1 w_2 \dots w_n$ where $w_i \in \Sigma$, e.g., $w = aabbc$
- **Kleene-closure**: Σ^* is the set of all finite strings over all including ϵ
- **Language**: a set of strings $L \subseteq \Sigma^*$, e.g., $L = \{\epsilon, a, ab, aa, aabc\}$

Regular Expression

- $\emptyset, \{\epsilon\}$ and $\{a\}, a \in \Sigma$ are regular languages
- If L_1 and L_2 are regular languages, the $L_1 \cup L_2$, $L_1 L_2$ and L_1^* are also
  - **Catenation**: $L_1 L_2 = \{w_1 w_2 : w_1 \in L_1, w_2 \in L_2\}$

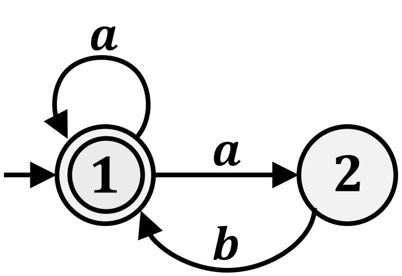
- $L = \{\epsilon, a\}\{a, b\}^* = \{\epsilon, a\}\,\{\epsilon, a, b, aa, ab, ba, bb, \dots\} = \{\epsilon, a, b, aa, ab, aaa, \dots\}$
- Remark: Σ can be 2^{AP} is previous examples!

Finite-State Automata

A **Non-deterministic Finite-State Automata (NFA)** is a tuple

$$A = (Q, Q_0, \delta, \Sigma, F)$$

- Q is a finite set of states
- $Q_0 \subseteq Q$ is the set of initial states
- Σ is the alphabet
- $\delta: Q \times \Sigma \to 2^Q$ is a partial transition function
- $F \subseteq Q$ is the set of accepting (final/marked) states.

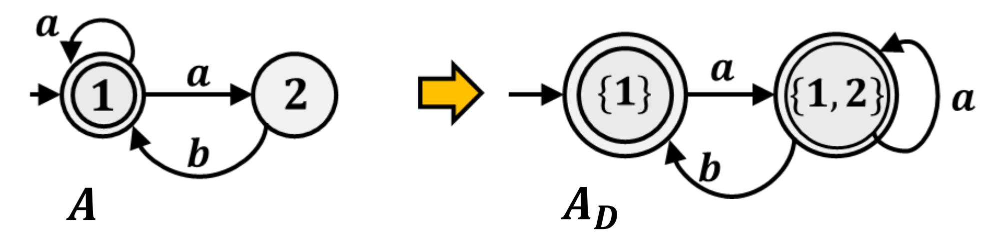


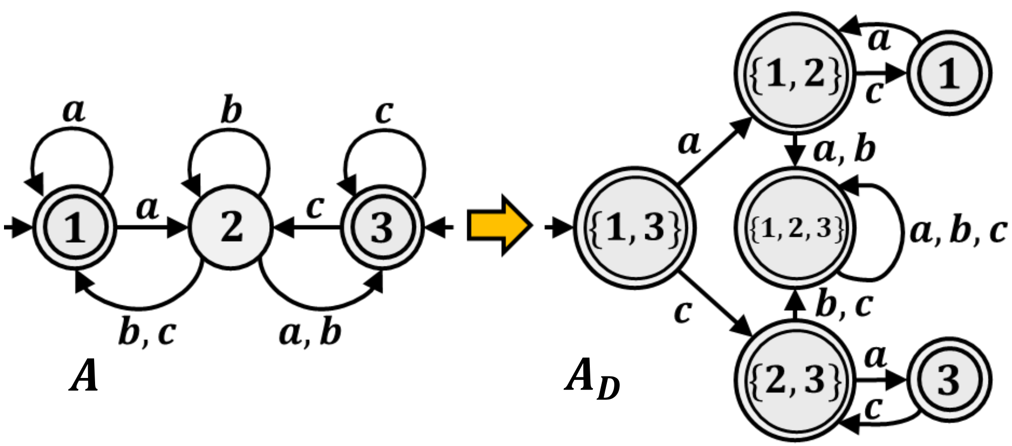
- $Q = \{1, 2\}, Q_0 = \{1\}, F = \{2\}, \Sigma = \{a, b\}, \delta(1, a) = \{1, 2\}$
- δ can be extended to $\delta: Q \times \Sigma^* \to 2^Q$, $\delta(1, aab) = \{1\}$
- **Accepted Language**: $\mathcal{L}(A) = \{s \in \Sigma^* : \exists q_0 \in Q_0, \delta(q_0, s) \cap F \neq \emptyset\}$
- $\mathcal{L}(A) = \{\epsilon, a, aa, ab, aab, aaba \ldots\}$

Theorem: A language is regular iff it can be accepted by a NFA.

NFA to DFA

- **Deterministic Finite-State Automata (DFA):** $|Q_0| = 1$ and $|\delta(q, \sigma)| = 1$
- **Accepted language can be simplified as** $\mathcal{L}(A) = \{s \in \Sigma^*: \delta(q_0, s) \in F\}$
- **Is NFA more powerful than DFA? No, they have the same power!**
- **Subset construction converts a NFA to a DFA with the same language**





Subset Construction

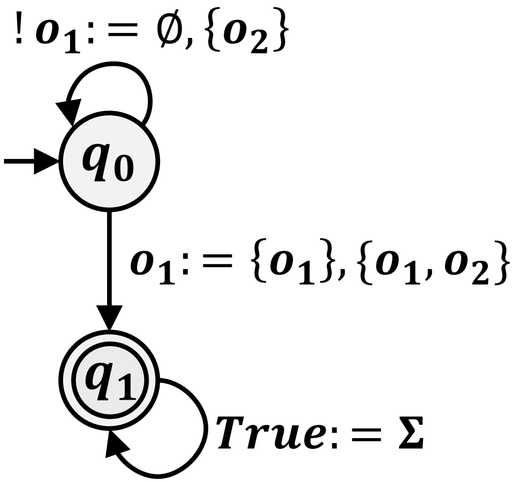
- start with Q_0
- for any $X \subseteq Q$ and $\sigma \in \Sigma$, compute

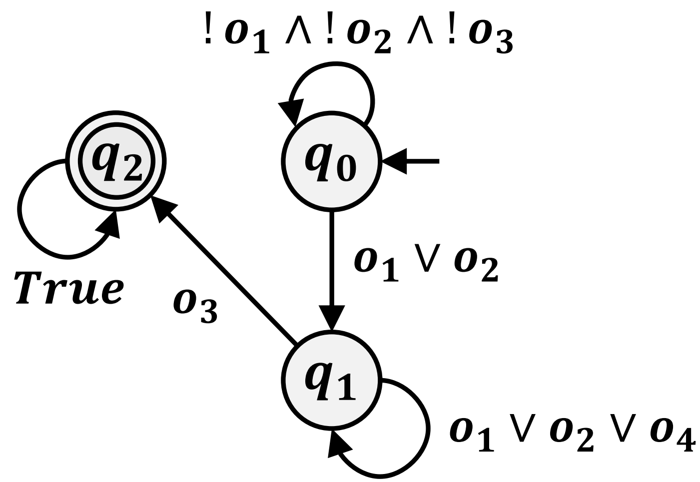
$$\delta_D(X, \sigma) = \cup_{q \in X} \delta(q, \sigma)$$

- mark X if it contains a state in F
- we have $\mathcal{L}(A_D) = \mathcal{L}(A)$
- A_D contains at most $2^{|Q|}$ states

From scLTL to DFA

For any scLTL formula ϕ over AP, there exists a DFA A_ϕ with alphabet $\Sigma = 2^{AP}$ that accepts all and only good prefixes of, i.e., $\mathcal{L}(A_\phi) = \mathcal{L}_{pref,\phi}$





- $AP = \{o_1, o_2\}$
- $\phi = \Diamond o_1$
- $\Sigma = \{\emptyset, \{o_1\}, \{o_2\}, \{o_1, o_2\}\}$

- $AP = \{o_1, o_2, o_3, o_4\}$
- $\phi = (\neg o_3 \; U \; (o_1 \vee o_2)) \wedge \Diamond o_3$

- A_ϕ contains at most $2^{|\phi|}$ states
- Software tools: scheck2 https://github.com/jsjolen/scheck2

Infinite Words & ω-Regular Language

- A regular language is a set of finite words
- For an alphabet Σ, Σ^ω is the set of all infinite words over Σ
- For a regular language $L \subseteq \Sigma^*$, we define $L^\omega = \{w_1 w_2 \cdots : w_i \in L\}$
- Example: for $L = \{ab, c\}$, we have $L^\omega = \{ababab \cdots, ccc \ldots, abcabcabc \ldots\}$
- **ω-Regular Language**: $L_1\left(L_{1,inf}\right)^\omega \cup L_2\left(L_{2,inf}\right)^\omega \cup \cdots \cup L_n\left(L_{n,inf}\right)^\omega$, where L_i and $L_{i,inf}$ are regular languages
- Safety: $\left(2^{AP}\right)^\omega \setminus P_{safe} = BadPref(P_{safe})\left(2^{AP}\right)^\omega$
- In fact, for any LTL formula ϕ, $Word(\phi)$ is ω-regular
- Question: how to generate a ω-regular language?

Non-deterministic Büchi Automata

A Non-deterministic Büchi Automata (NBA) is a tuple

$$A = (Q, Q_0, \delta, \Sigma, F)$$

- Q is a finite set of states
- $Q_0 \subseteq Q$ is the set of initial states
- Σ is the alphabet
- $\delta: Q \times \Sigma \to 2^Q$ is a partial transition function
- $F \subseteq Q$ is the set of accepting (final/marked) states.

- The structures of NBA and NFA are exactly the same
- The difference is how to interpret the accepting condition
- NBA is used to accept infinite words
- An infinite word is accepted if it visits accepting states infinitely many times

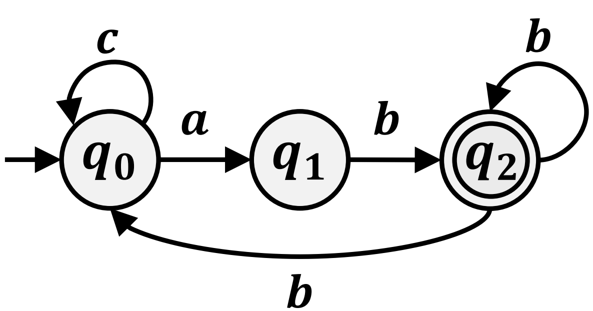
Non-deterministic Büchi Automata

- Given an infinite word $w = w_0 w_1 w_2 w_3 \cdots \in \Sigma^\omega$
- A run for w is an infinite sequence of states $q_0 q_1 q_2 \ldots$ such that

$$q_0 \in Q_0 \text{ and } \forall i \geq i: q_{i+1} \in \delta(q_i, w_i)$$

- A run $\rho = q_0 q_1 q_2 \ldots$ is said to be accepting if states in F occurs infinitely many times, i.e., $Inf(\rho) \cap F \neq \emptyset$
- Accepted language of NBA A is

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



NBA A

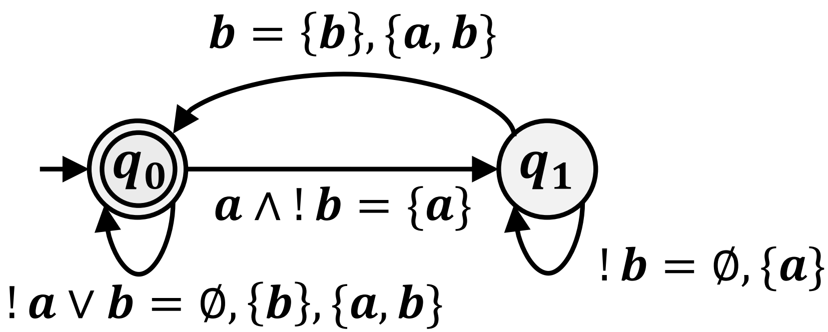
- word c^ω only has one run q_0^ω
- word ab^ω has accepting run $q_0 q_1 q_2^\omega$
- Word $(cabb)^\omega$ has accepting run $(q_1 q_1 q_2 q_3)^\omega$
- This NBA actually accepts ω-regular language

$$\{c\}^*\{ab\}(\{b\}^+ \cup \{b\}\{c\}^*\{ab\})^\omega$$

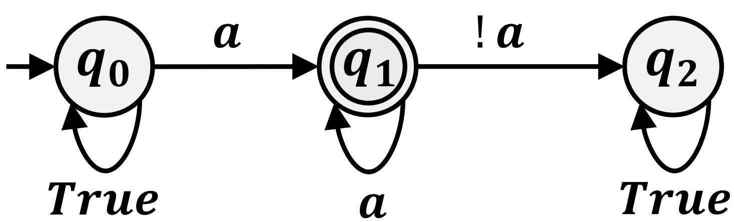
$$\text{where } \{b\}^+ = \{b\}^* \setminus \{\epsilon\} = \{b, bb, bbb, \ldots\}$$

From LTL to NBA

- A language is ω-regular iff it can be accepted by a NBA
- For any LTL formula ϕ over AP, there exists an NBA A_ϕ with alphabet $\Sigma = 2^{AP}$ such that $\mathcal{L}^\omega(A_\phi) = Word(\phi)$



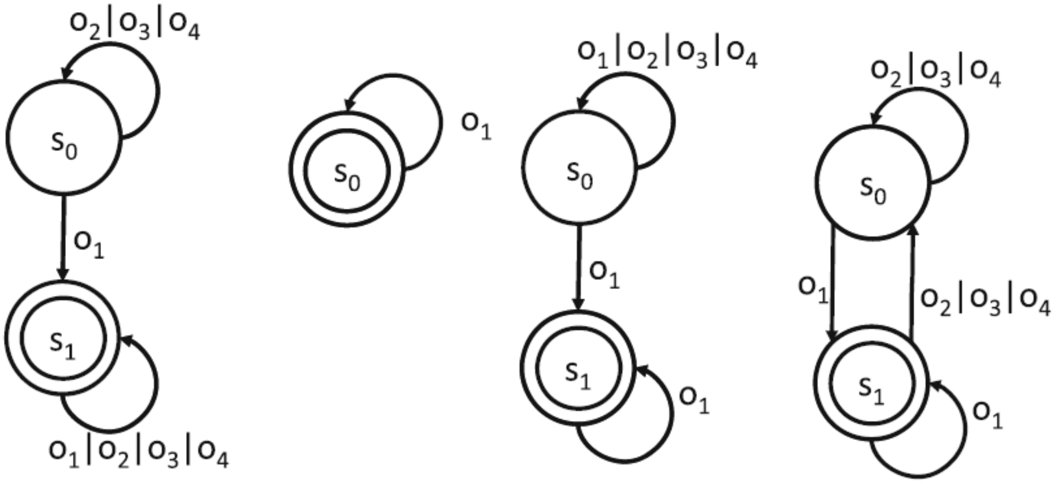
NBA for $\square(a \rightarrow \lozenge b)$



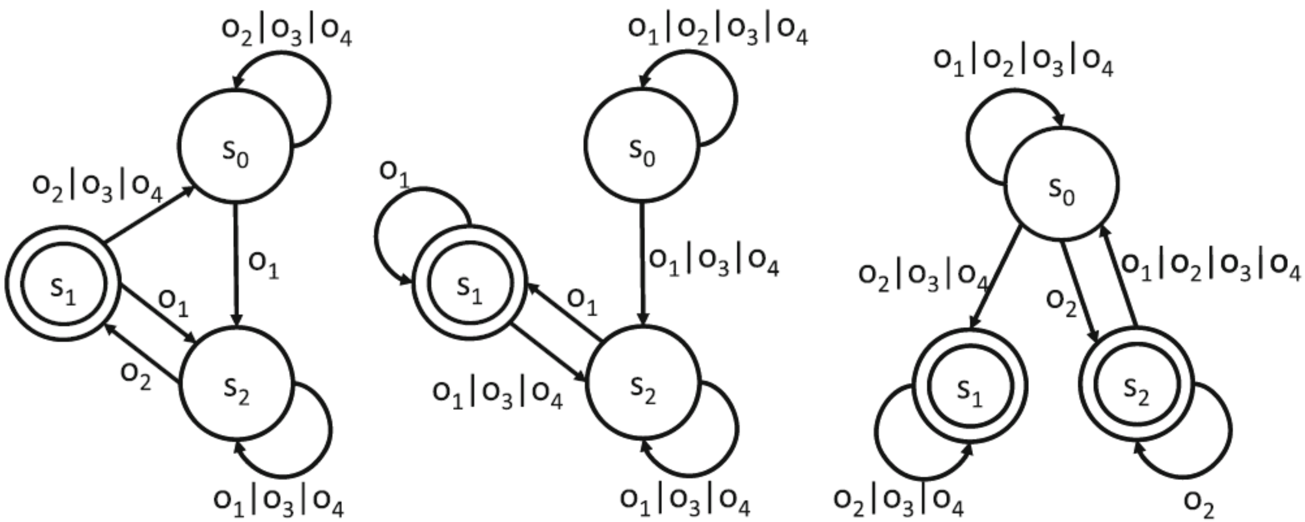
NBA for $\lozenge\square a$

- A_ϕ contains at most $|\phi|2^{|\phi|}$ states
- Software tools: ltl2ba http://www.lsv.fr/~gastin/ltl2ba/

More Examples for LTL to NBA



(a) $\phi_1 = \Diamond o_1$ (b) $\phi_2 = \Box o_1$ (c) $\phi_3 = \Diamond\Box o_1$ (d) $\phi_4 = \Box\Diamond o_1$



(e) $\phi_5 = \Box(\Diamond o_1 \wedge \Diamond o_2)$ (f) $\phi_6 = \Box\Diamond o_1 \wedge \neg\Box\Diamond o_2$ (g) $\phi_7 = \Box\Diamond o_1 \Rightarrow \Box\Diamond o_2$

Pictures from the book of Belta

Model Checking for Regular Safety

- P_{safe} is a safety property if each $\sigma \notin P_{safe}$ has a finite bad prefix
- P_{safe} is regular safety is its bad prefixes is a regular language
- Suppose that NFA A_{bad} accepts the bad prefixes, then

	$T \nvDash P$
if and only if	$Trace(T) \nsubseteq P$
if and only if	$Trace(T) \cap \left(\left(2^{AP}\right)^{\omega} \setminus P\right) \neq \emptyset$
if and only if	$L(x_1)L(x_2)\ldots L(x_n) \in Trace^f(T) \cap L_{bad}$
if and only if	$Trace^f(T) \cap \mathcal{L}(A_{bad}) \neq \emptyset$

➢ The last condition can be checked by "synchronizing" T and A_{bad}!

Model Checking for Regular Safety

Let $T = (X, U, \rightarrow, X_0, AP, L)$ be a LTS and $A = (Q, Q_0, \delta, 2^{\mathrm{AP}}, F)$ be an NFA.
Then the **product of T and A** is a new tuple

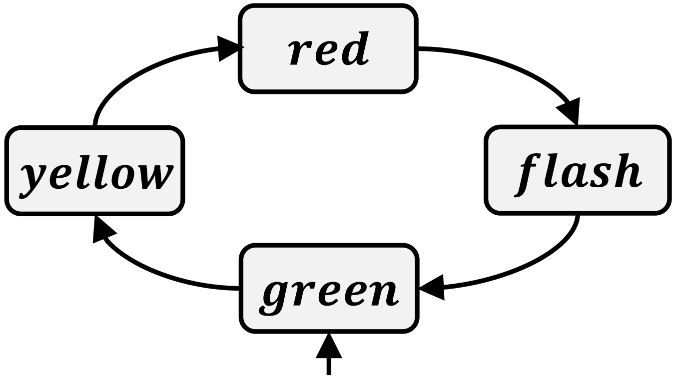
$$T \otimes A = (Q_\otimes, Q_{0\otimes}, \delta_\otimes, U, F_\otimes)$$

- $Q_\otimes = X \times Q$
- $Q_{0\otimes} = \{(x_0, q) : x_0 \in X_0 \wedge \exists q_0 \in Q_0 . q_0 \xrightarrow{L(x_0)} q\}$
- $F_\otimes = X \times F$
- $\delta_\otimes : Q_\otimes \times U \rightarrow 2^{Q_\otimes}$ is defined by:
  - $\delta_\otimes\big((x, q), u\big) = \{(x', q') \in Q_\otimes : x \xrightarrow{u} x'$ and $q \xrightarrow{L(x')} q'\}$

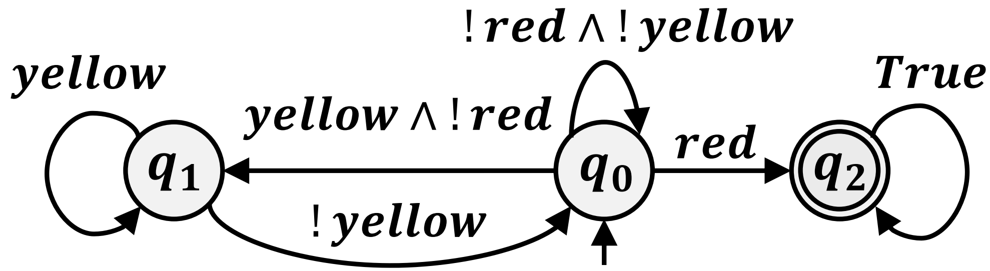
Key Observation:

- $T \otimes A$ accepts finite traces both generated by T and accepted by A
- $(x_0, q_1)(x_1, q_2) \cdots (x_n, q_{n+1}) \Rightarrow q_0 \xrightarrow{L(x_0)} q_1 \xrightarrow{L(x_1)} q_2 \xrightarrow{L(x_2)} \cdots \xrightarrow{L(x_n)} q_{n+1}$

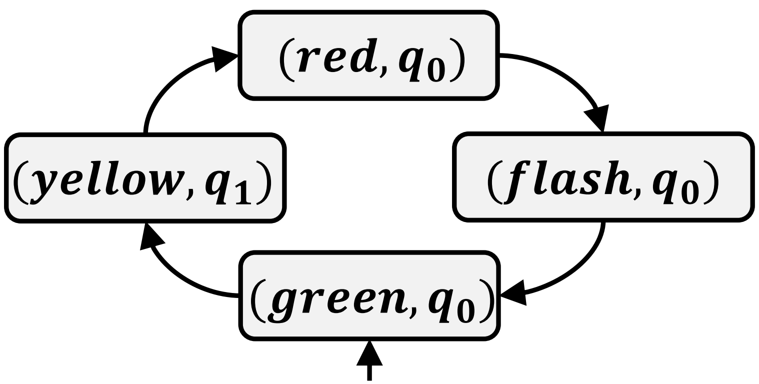
Example: LTS-NFA Product



LTS T for traffic light



DFA A_{bad} for “each red is preceded by yellow”



$T \otimes A$ as the product

Model Checking for Regular Safety

Given: T and NFA A_{bad}

- Build the product $T \otimes A$
- If $\mathcal{L}(T \otimes A) = \emptyset$, then return “safe”
- If $\mathcal{L}(T \otimes A) \neq \emptyset$, i.e., there is a reachable accepting state in $T \otimes A$, then return “not safe”

Model Checking for LTL

- Suppose we have an ω-regular property $P \subseteq \left(2^{AP}\right)^{\omega}$
- Now we have an LTS $T = (X, U, \to, X_0, AP, L)$ and we want to check whether or not $T \vDash P$
- Based the previous discussions, we have

$$
\begin{array}{ll}
 & T \nvDash P \\
\text{if and only if} & Trace(T) \nsubseteq P \\
\text{if and only if} & Trace(T) \cap \left(\left(2^{AP}\right)^{\omega} \setminus P\right) \neq \emptyset \\
\text{if and only if} & Trace(T) \cap P^c \neq \emptyset \\
\text{if and only if} & Trace(T) \cap \mathcal{L}^{\omega}(A^c) \neq \emptyset
\end{array}
$$

- $\left(2^{AP}\right)^{\omega} \setminus P$ is also ω-regular
- without loss of generality, we can assume $\mathcal{L}^{\omega}(A^c) = P^c$
- If $P = Word(\phi)$, then $P^c = Word(\neg\phi)$and we can build $A_{\neg\phi}$!

Product between LTS and NBA

Let $T = (X, U, \rightarrow, X_0, AP, L)$ be a LTS and $A = (Q, Q_0, \delta, 2^{AP}, F)$ be an NBA. Then the **product of T and A** is a new tuple

$$T \otimes A = (Q_\otimes, Q_{0\otimes}, \delta_\otimes, U, F_\otimes)$$

- $Q_\otimes = X \times Q$
- $Q_{0\otimes} = \{(x_0, q) : x_0 \in X_0 \wedge \exists q_0 \in Q_0 . q_0 \xrightarrow{L(x_0)} q\}$
- $F_\otimes = X \times F$
- $\delta_\otimes : Q_\otimes \times U \rightarrow 2^{Q_\otimes}$ is defined by:
  - $\delta_\otimes\big((x, q), u\big) = \{(x', q') \in Q_\otimes : x \xrightarrow{u} x'$ and $q \xrightarrow{L(x')} q'\}$

Exactly the same as the case of NFA!

LTL Model Checking Algorithm

- Suppose that we have an LTS T and an LTL formula ϕ
- We have $T \nvDash \phi \Leftrightarrow Trace(T) \nsubseteq Word(\phi) \Leftrightarrow Trace(T) \cap Word(\neg\phi) \neq \emptyset$
- Let $A_{\neg\phi}$ be an NBA such that $\mathcal{L}^{\omega}(A_{\neg\phi}) = Word(\neg\phi)$. Then

$$Trace(T) \cap Word(\neg\phi) \neq \emptyset \Leftrightarrow \mathcal{L}^{\omega}(T \otimes A_{\neg\phi}) \neq \emptyset$$

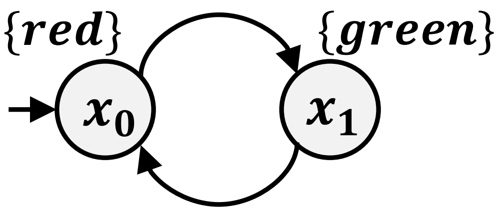
- The above is equivalent to the existences an accepting state in $T \otimes A_{\neg\phi}$ that can be reached infinitely often, i.e., in a cycle!

Model Checking for LTL

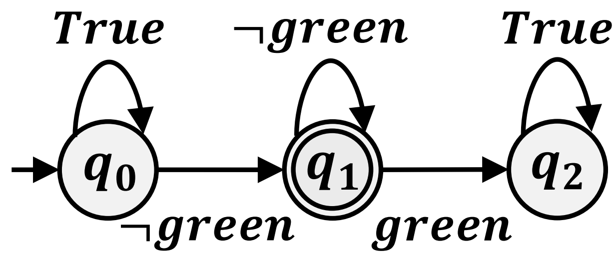
Given: T and LTL Formula ϕ

- Build the NBA $A_{\neg\phi}$ that accepts $Word(\neg\phi)$
- Build the product $T \otimes A_{\neg\phi}$
- Find all strongly connected components (SCC) of $T \otimes A_{\neg\phi}$
- Check if there exists a SCC that contains a state in $F_{\otimes}$ and at least a transition
- If so, return "$T \nvDash \phi$"; otherwise, return "$T \vDash \phi$"

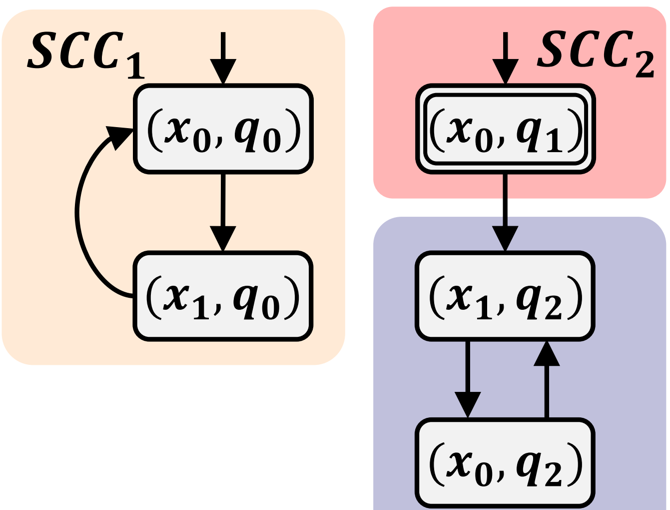
Example: LTL Model Checking



LTS T for traffic light



NBA A for $\neg(\Box\Diamond green) = \Diamond\Box\neg green$

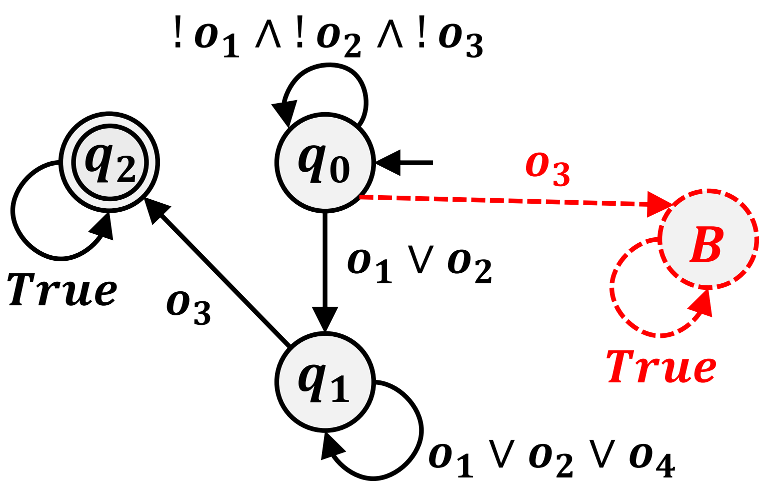


$T \otimes A$

- **There are three SCCs in $T \otimes A$**
- **SCC_2 contains $F_\otimes$ but does not have transition**
- **SCC_1, SCC_3 have transitions but have no $F_\otimes$**
- **No infinite accepting word can be generated**
- **Therefore, $T \vDash \Box\Diamond green$**

Case of scLTL

- For any scLTL formula ϕ, there is a DFA $A = (Q, q_0, \delta, \Sigma, F)$ that accepts all good prefixes, i.e., $\mathcal{L}(A) = \mathcal{L}_{pref,\phi}$
- Non-satisfaction means
  - Never reach an accepting state, i.e., loop in non-accepting; or
  - Outside of the transitions of A
- Build A_{com} to "complete" the transition and compute $T \otimes A_{com}$
- If $T \otimes A_{com}$ contains a cycle in which there is no accepting state, then $T \not\models \phi$



- $AP = \{o_1, o_2, o_3, o_4\}$
- $\phi = (\neg o_3 \; U \; (o_1 \vee o_2)) \wedge \Diamond o_3$

Build $A_{com} = (Q^c, q_0, \delta^c, \Sigma, F)$

- $Q^c = Q \cup \{Bad\}$
- If $\delta(q, w)!$, then $\delta^c(q, w) = \delta(q, w)$
- If $\delta(q, w)\neg!$, then $\delta^c(q, w) = Bad$

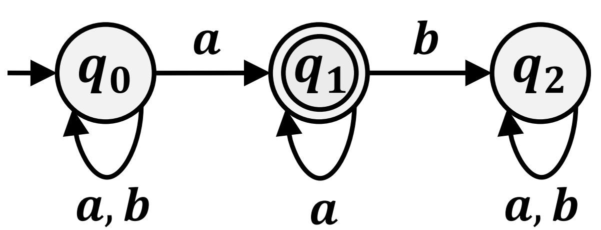
Discussions

- NFA accepts finite words, i.e., generates regular language
- NFA can DFA are equivalent according to the subset construction
- NBA accepts infinite words, i.e., generates ω-regular language
- Regular safety is essentially non-reachability
- Co-Safe LTL is essentially safety + finite reachability
- Safety can be converted to non-reachability by adding state Bad
- General LTL is essentially persistence
- Are NBA and DBA equivalent? No!

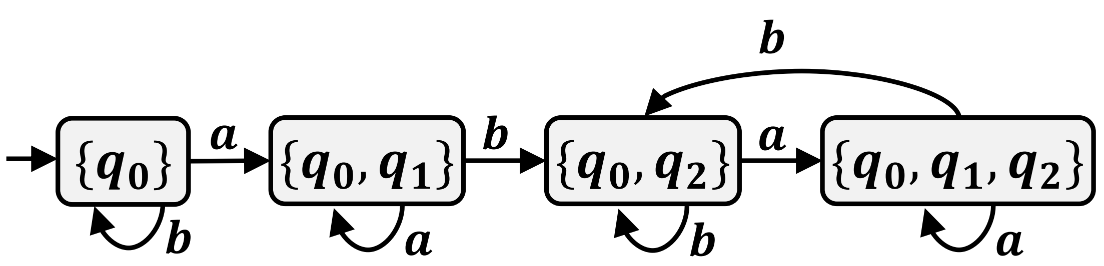
NBA v.s. DBA

NBA is strictly more powerful than DBA, i.e., there exists ω-regular language that cannot be accepted by a DBA.

- Deterministic Büchi Automata (DBA): $|Q_0| = 1$ and $|\delta(q, \sigma)| = 1$
- "eventually for every" cannot be captured by DBA
- Consider ω-regular language $L = \{a, b\}^*\{a\}^\omega$
- We need to nondeterministically decide from which instant the proposition a is continuously true



NBA for $L = \{a, b\}^*\{a\}^\omega$



- ➢ Automata obtained by the subset constriction
- ➢ Defining accepting states is problematic!

Rabin Automata

- In many problems we do need deterministic mechanism, but the expressiveness of DFA is limited
- Using different accepting condition: Rabin acceptance

A Deterministic Rabin Automata (DRA) is a tuple

$$A = (Q, q_0, \delta, \Sigma, Acc)$$

- Q is a finite set of states, $q_0 \in Q$ is the initial state, Σ is the alphabet
- $\delta: Q \times \Sigma \to Q$ is a partial deterministic transition function
- $Acc = \{(L_1, K_1), \dots, (L_n, K_n)\} \subseteq 2^Q \times 2^Q$ is the acceptance condition.

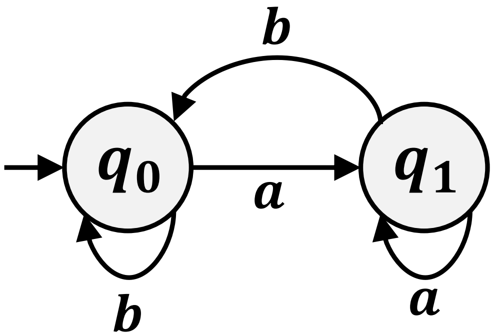
- A run $\rho = q_0 q_1 q_2 \dots$ is accepting if there exists a pair $(L, K) \in Acc$ s.t.

$$[Inf(\rho) \cap L = \emptyset] \wedge [Inf(\rho) \cap K \neq \emptyset]$$

- Accepted language of DBA A is

$$\mathcal{L}^\omega(A) = \{w \in \Sigma^\omega : \text{the run induced by } w \text{ is accepting in } A\}$$

Rabin Automata

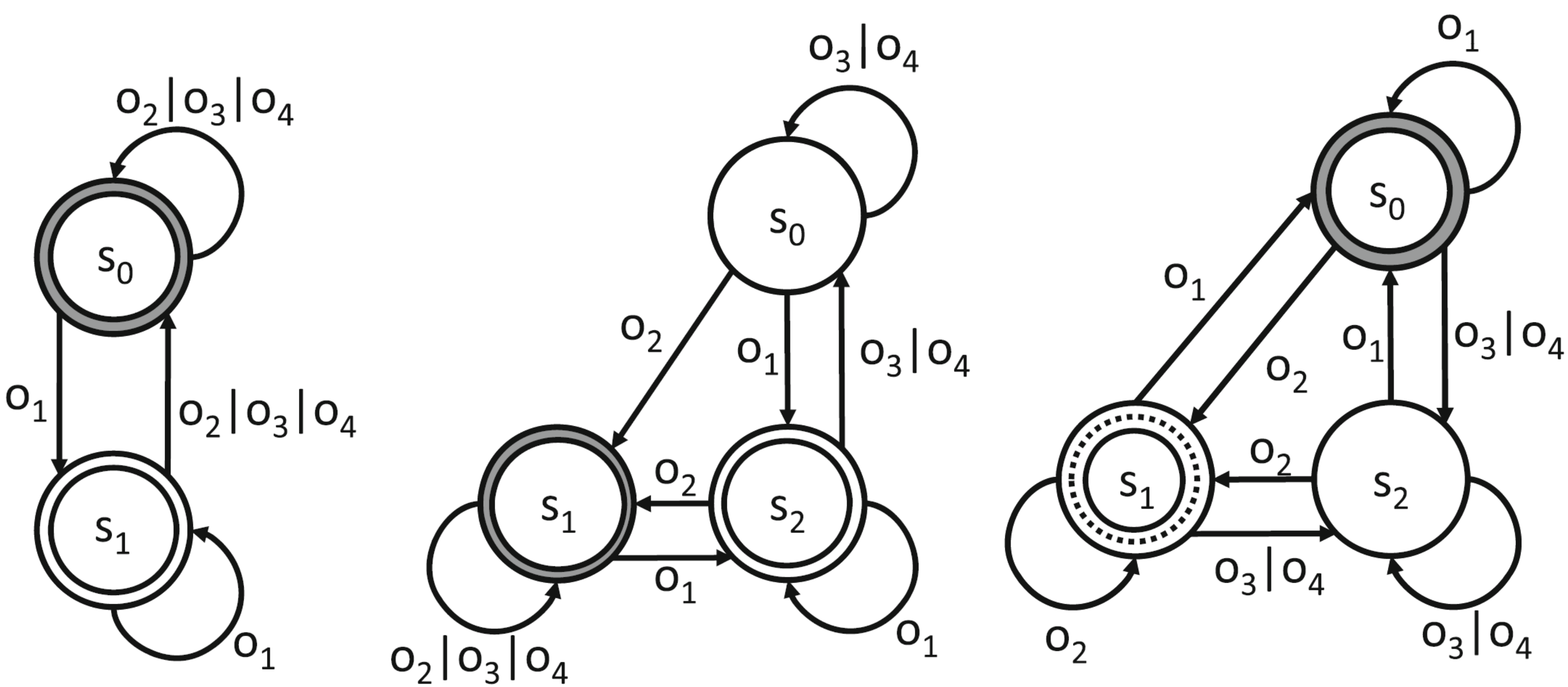


- DRA for "$\Diamond\Box a$"
- $Acc = \{(\{q_0\}, \{q_1\})\}$
- Looping between q_0 and q_1 is rejected

- **The class of languages accepted by DRA is the same as that of NBA**
- **For any LTL formula ϕ over AP, there exists a DRA A_ϕ with alphabet $\Sigma = 2^{AP}$ such that $\mathcal{L}^\omega(A_\phi) = Word(\phi)$**
- **Cost: we may need $2^{2^{|\phi|\cdot\log|\phi|}}$ states and $2^{|\phi|}$ pairs**
- **Tools: ltl2dstar https://www.ltl2dstar.de/**

Rabin Automata: More Examples

$Acc = \{(\{s_0\}, \{s_1\})\}$ $Acc = \{(\{s_1\}, \{s_2\})\}$ $Acc = \{(\{s_0\}, \{s_1, s_2\}), (\emptyset, \{s_1\})\}$



(a) $\phi_3 = \Diamond\Box o_1$ (b) $\phi_6 = \Box\Diamond o_1 \wedge \neg\Box\Diamond o_2$ (c) $\phi_7 = \Box\Diamond o_1 \Rightarrow \Box\Diamond o_2$

Pictures from the book of Belta

Another Definition of Product

Alternative Definition

Let $T = (X, U, \to, X_0, AP, L)$ be a LTS and $A = (Q, Q_0, \delta, 2^{\mathrm{AP}}, F)$ be an NBA. Then the product of T and A is a new tuple

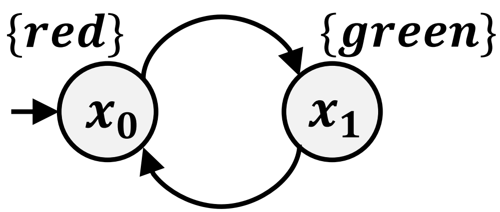
$$T \otimes A = (Q_\otimes, Q_{0\otimes}, \delta_\otimes, U, F_\otimes)$$

- $Q_\otimes = X \times Q$, $Q_{0\otimes} = X_0 \times Q_0, F_\otimes = X \times F$
- $\delta_\otimes : Q_\otimes \times U \to 2^{Q_\otimes}$ is defined by:
  - $\delta_\otimes\big((x, q), u\big) = \{(x', q') \in Q_\otimes : x \xrightarrow{u} x'$ and $q \xrightarrow{L(x)} q'\}$

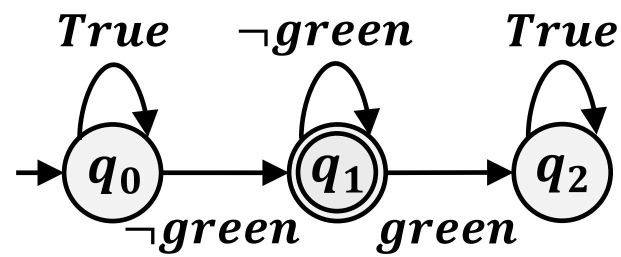
Previous Definition

- $Q_{0\otimes} = \{(x_0, q) : x_0 \in X_0 \wedge \exists q_0 \in Q_0 . q_0 \xrightarrow{L(x_0)} q\}$
- $\delta_\otimes : Q_\otimes \times U \to 2^{Q_\otimes}$ is defined by:
  - $\delta_\otimes\big((x, q), u\big) = \{(x', q') \in Q_\otimes : x \xrightarrow{u} x'$ and $q \xrightarrow{L(x')} q'\}$

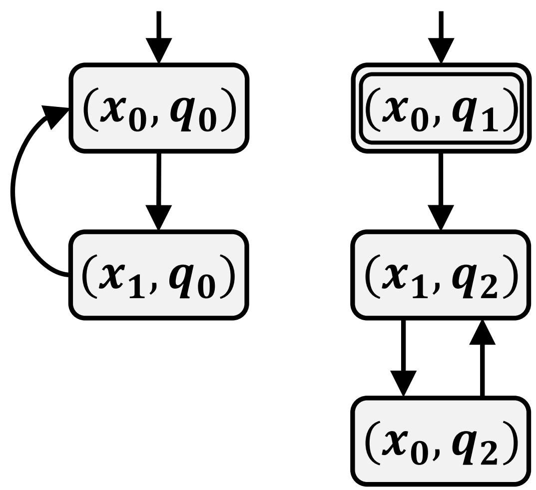
Example: LTS-NBA Product



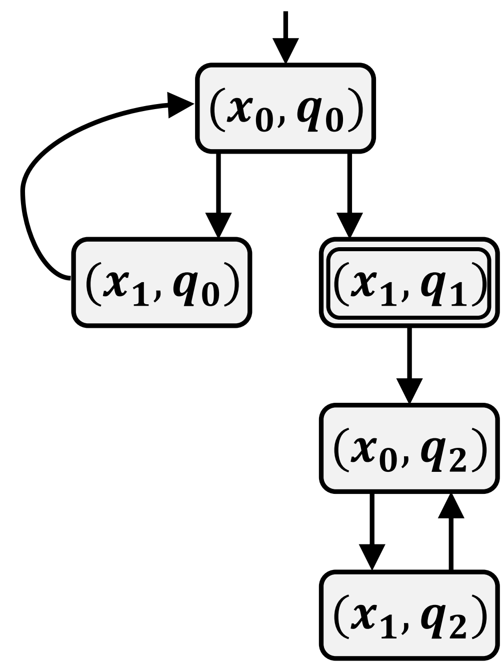
LTS T for traffic light



NBA A for $\neg(\Box\Diamond green) = \Diamond\Box\neg green$



$T \otimes A$ by Definition 1



$T \otimes A$ by Definition 2

Stage Summary

- Any regular language can be accepted by an NFA
- Any ω-regular language can be accepted by an NBA
- NFA and DFA are equivalent but NBA and DBA are not equivalent
- Any LTL formula can be translated to an NBA (DBA is not enough)
- Any LTL formula can be translated to DRA (if we need determinsm)
- Büchi is smaller but need to pay nondeterminism
- Rabin can resolve nondeterminism but need to pay larger state-space
- Good prefixes of scLTL can be accepted by DFA
- Model checking by synchronizing the LTS and the automaton for LTL

Review of Last Course

- We use automata to generate language of interest
- Good prefixes of scLTL can be accepted by Deterministic FA
- Any LTL formula can be translated to an Non-deterministic BA
- Any LTL formula can be translated to Deterministic Rabin Automaton
- Model checking by synchronizing the LTS and the automaton for LTL
    - Regular safety: non-reachability of bad states
    - scLTL: finite reachability of accepting states
    - LTL: persistency of accepting states