Alcino Cunha

SPECIFICATION AND MODELING

LTL MODEL CHECKING

Universidade do Minho & INESC TEC

2019/20

LINEAR TEMPORAL LOGIC

SYNTAX

$$
\begin{array}{rcl}
\phi & ::= & p \\
& \mid & \top \\
& \mid & \bot \\
& \mid & \neg\phi \\
& \mid & \phi_1 \wedge \phi_2 \\
& \mid & \phi_1 \vee \phi_2 \\
& \mid & \phi_1 \longrightarrow \phi_2 \\
& \mid & \mathsf{G}\,\phi \\
& \mid & \mathsf{F}\,\phi \\
& \mid & \mathsf{X}\,\phi \\
& \mid & \phi \;\mathsf{U}\; \psi \\
& \mid & \phi \;\mathsf{R}\; \psi
\end{array}
$$

SEMANTICS

- Defined over a model (transition system) M
- For non first-order LTL, M is just a *Kripke structure* (S, I, R, L)
  - S is a finite set of states
  - $I \subseteq S$ is the set of initial states
  - $R \subseteq S \times S$ is a total transition relation
  - $L : S \longrightarrow 2^A$ is a function that labels each state with the set of atomic propositions valid in that state (draw from domain A)
- A formula is valid iff it holds in all paths of M

$$M \models \phi \quad \text{iff} \quad \forall \pi \in M \cdot \pi \models \phi$$

MODEL CHECKING

A LANGUAGE THEORETIC APPROACH TO LTL MODEL CHECKING

- Considering the set of states S as an *alphabet* Σ, the language of M, denoted $\mathcal{L}(M)$, is the set of all paths of M

$$\mathcal{L}(M) = \{\pi \mid \pi \in M\}$$

- The language of a LTL formula ϕ, denoted $\mathcal{L}(\phi)$, is the set of all paths that satisfy ϕ

$$\mathcal{L}(\phi) = \{\pi \mid \pi \models \phi\}$$

- A formula is valid in a model iff the language of the model is contained in the language of the formula

$$M \models \phi \quad \text{iff} \quad \mathcal{L}(M) \subseteq \mathcal{L}(\phi)$$

- Alternatively we have

$$M \models \phi \quad \text{iff} \quad \mathcal{L}(M) \cap \mathcal{L}(\neg\phi) = \varnothing$$

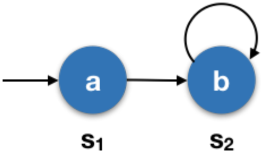
BÜCHI AUTOMATA

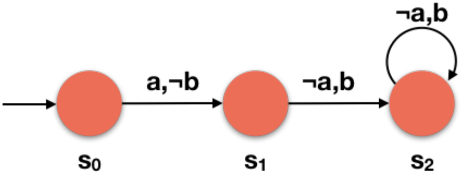
- In general, the language $\mathcal{L}(\phi)$ cannot be captured by a transition system
  - We need the related concept of *Büchi automaton*
- A *Non-deterministic Büchi automaton* (NBA) is a tuple (S, Σ, R, I, F) where
  - S is a set of states
  - Σ is a alphabet
  - $R \subseteq S \times \Sigma \times S$ is a transition relation
  - $I \subseteq S$ is a set of initial states
  - $F \subseteq S$ is a set of accepting (or final) states
- A valid path in a NBA must visit an accepting state infinitely often
- The language of an NBA is the set of all valid paths

FROM KRIPKE STRUCTURES TO BÜCHI AUTOMATA

- Given a Kripke structure M it is possible to construct a NBA $\mathcal{A}_M$ such that $\mathcal{L}(\mathcal{A}_M) = \mathcal{L}(M)$
  - Using as alphabet conjunctions of atomic propositions $\Sigma = 2^A$
  - Adding a new separate initial state
  - A transition is possible iff transition label matches the next state label
  - All states are accepting

EXAMPLE



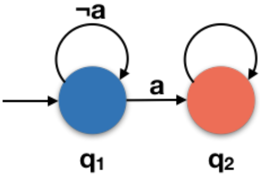


FROM LTL FORMULAS TO BÜCHI AUTOMATA

- Given a LTL formula in negation normal form it is possible to construct a NBA $\mathcal{A}_\phi$ such that $\mathcal{L}(\mathcal{A}_\phi) = \mathcal{L}(\phi)$
  - Using as alphabet conjunctions of atomic propositions $\Sigma = 2^A$

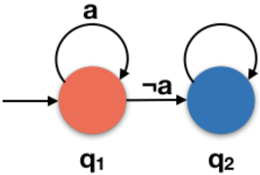
EXAMPLE

$$\mathrm{F}\, a$$



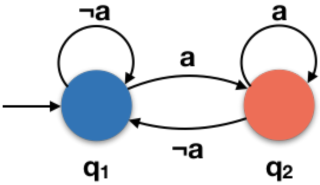
EXAMPLE

$$\mathrm{G}\, a$$



EXAMPLE

$$\mathrm{G}\,\mathrm{F}\,a$$



CHECKING EMPTINESS OF LANGUAGE INTERSECTION

- Checking the emptiness of language intersection can be reduced to checking the emptiness of the product automaton

$$M \models \phi \quad \text{iff} \quad \mathcal{L}(M) \cap \mathcal{L}(\neg\phi) = \emptyset \quad \text{iff} \quad \mathcal{L}(\mathcal{A}_M \otimes \mathcal{A}_{\neg\phi}) = \emptyset$$

- Since all states of $\mathcal{A}_M$ are accepting, the product of $\mathcal{A}_M = (S_M, \Sigma, R_M, I_M, S_M)$ and $\mathcal{A}_{\neg\phi} = (S_{\neg\phi}, \Sigma, R_{\neg\phi}, I_{\neg\phi}, F_{\neg\phi})$ can be computed as follows

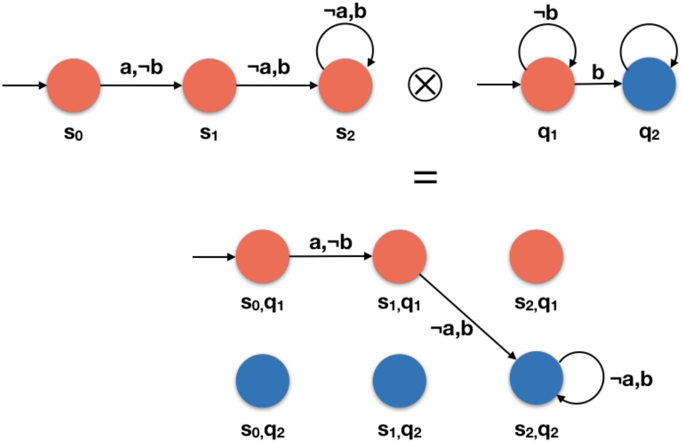
$$\mathcal{A}_M \otimes \mathcal{A}_{\neg\phi} = (S_M \times S_{\neg\phi}, \Sigma, R, I_M \times I_{\neg\phi}, S_M \times F_{\neg\phi})$$

where

$$((s, q), a, (s', q')) \in R \quad \text{iff} \quad (s, a, s') \in R_M \wedge (q, a, q') \in R_{\neg\phi}$$

EXAMPLE

$$M \models \mathsf{F}\, b \quad \text{iff} \quad \mathcal{L}(\mathcal{A}_M \otimes \mathcal{A}_{\mathsf{G}\,\neg b}) = \emptyset$$



CHECKING (NON) EMPTINESS OF AUTOMATON

1. Compute the Strongly Connected Components (SCCs) and check if a SCC containing an accepting state is reachable from the initial state
   - Requires storing the entire automaton in memory
2. Determine reachable states using Depth-First Search (DFS) and if an accepting state is reachable run a nested DFS to determine if it there is a cycle
   - Better for on-the-fly model checking
3. Use a (fair) CTL model checking procedure to check if EG $\top$ is valid assuming the system is fair to the accepting states
   - Enables symbolic model checking for LTL