EECS 219C: Computer-Aided Verification

Properties as Automata and Explicit-State Model Checking

Sanjit A. Seshia
EECS, UC Berkeley

Mental Picture

System \[\xrightarrow{\text{trace}}\] Monitor Automaton
“checking that trace is correct”
Recap: Automata over Finite Traces

• (Regular) Finite automaton with accepting states
  – All finite traces (words) that take the automaton into the accepting state are “in its language”
• But behaviors (and traces) are infinite length
  – So we need a new notion of acceptance

Automata over Infinite Traces

• What does “Accept” mean?
  – Certain states of the automaton are called “accepting states”
  – The trace must visit an (any) accepting state infinitely often
• Such automata are called Büchi automata
  – Also Omega-automata (written \(\omega\)-automata)
Example from Class

Language of the automaton = all finite-length binary strings with an odd number of 1s

Reg. expr.: $0*1 (0 + 10*1)^*$

If you interpret it as a Büchi automaton over infinite words: all infinite-length binary strings with an odd parity of 1s or infinitely many 1s

w-regular expr: $0*1 (0 + 10*1)^w$

From Temporal Logic to Monitors

- A monitor for a temporal logic formula
  - is a finite automaton
  - Accepts exactly those behaviors that satisfy the temporal logic formula
    - “Accepts” means that an accepting state is visited infinitely often

- Properties are often specified as automata
Mental Picture

System → trace → Monitor Automaton
“checking that trace is correct”

Summary

- A (Büchi) automaton corresponding to a temporal logic formula $\phi$ accepts exactly those traces that satisfy $\phi$
Automaton for G p, p a Boolean formula

Automaton for F p
From LTL to Automata (1)

- Any LTL formula can be translated to a corresponding automaton
- There are many translation algorithms
  - We discuss the classical tableaux-based one
From LTL to Automata (2)

- The tableaux-based algorithm has three steps:
  - Translating the LTL formula into negation normal form (NNF)
  - Translating NNF to a generalized Büchi automaton
  - Degeneralizing the gen. Büchi automaton

LTL Negation normal form

- Idea: all negations can be pushed inwards

\[
\begin{align*}
\neg(\psi \lor \phi) & \equiv (\neg \psi) \land (\neg \phi) \\
\neg(\psi \land \phi) & \equiv (\neg \psi) \lor (\neg \phi) \\
\neg \mathsf{G} \psi & \equiv \mathsf{F} \neg \psi \\
\neg \mathsf{F} \psi & \equiv \mathsf{G} \neg \psi \\
\neg \mathsf{X} \psi & \equiv \mathsf{X} \neg \psi \\
\neg(\psi \mathsf{U} \phi) & \equiv (\neg \psi) \mathsf{R} (\neg \phi) \\
\neg(\psi \mathsf{R} \phi) & \equiv (\neg \psi) \mathsf{U} (\neg \phi)
\end{align*}
\]

Consider the LTL formula \(\neg((\mathsf{G} p) \mathsf{U} q) \lor \neg((\mathsf{F} \neg(p \mathsf{U} q)))\) for the atomic proposition set \(\{p, q\}\). We can rewrite this formula into negation normal form as follows:

\[
\begin{align*}
\neg((\mathsf{G} p) \mathsf{U} q) \lor \neg((\mathsf{F} \neg(p \mathsf{U} q))) & \\
\equiv (\neg \mathsf{G} p) \mathsf{R} \neg q \lor \mathsf{G} (\neg (p \mathsf{U} q)) & \\
\equiv ((\mathsf{F} \neg p) \mathsf{R} q) \lor \mathsf{G} (p \mathsf{U} q)
\end{align*}
\]
Generalized Büchi acceptance

• Idea: We have multiple sets of accepting states, and all sets have to be visited infinitely often for the automata to accept a word.

Tableaux construction

• Main Idea:

\[
\psi U \phi \equiv \phi \lor (\psi \land X(\psi U \phi)) \\
\psi R \phi \equiv (\psi \land \phi) \lor (\phi \land X(\psi R \phi))
\]

• Complexity?
From Automata to LTL

• Any LTL formula can be translated to a corresponding automaton
• How about the other way around?
  – Can an arbitrary Büchi automaton be translated into an LTL formula?

Automaton without LTL counterpart

Automata are more expressive than LTL
What traces does the automaton below accept?

Claim: This cannot be expressed in LTL.
(How about \( a \land G (a \Rightarrow X X a) \) ?)