LTL Model Checking

Barbara Jobstmann
Recall: Linear-Time View

Behavior of a system is a set of paths = language
LTL Syntax

• LTL Syntax (p...atomic proposition)
  \( \phi := p \mid \phi \land \phi \mid \neg \phi \mid X \phi \mid \phi U \phi \mid \phi R \phi \)
  
  – Boolean operators
  – Temporal operators
     \( X\phi \) (next)
     \( \phi U \psi \) (until)
     \( \phi R \psi \) (release)
  
  – Abbreviations:
     \( \text{false} = p \land \neg p \)
     \( \text{true} = \neg \text{false} \)
     \( \phi \lor \psi = \neg (\neg \phi \land \neg \psi) \)
     \( G(\phi) = \text{false} R \phi, F(\phi) = \text{true} U \phi \)
LTL Semantics

• Given an infinite path $\pi = \pi_0\pi_1\ldots \in (2^A)^\omega$
  - $\pi \models \varphi U \psi$ iff there exists $i \geq 0$ s.t. $\pi^i \models \psi$ and for all $0 \leq j < i$, $\pi^j \models \varphi$
  - $\pi \models \varphi R \psi$ iff for all $i \geq 0$ $\pi^i \models \psi$ or there exists $j \geq 0$ s.t. $\pi^j \models \varphi$ and for all $0 \leq i \leq j$, $\pi^i \models \psi$

• Applying this semantics to F and G gives:
  - $\pi \models F \varphi$ iff there exists $i \geq 0$ s.t. $\pi^i \models \varphi$
  - $\pi \models G \varphi$ iff for all $i \geq 0$. $\pi^i \models \varphi$

• Kripke structure $K \models \varphi$ iff all paths $\pi$ of $K$, $\pi \models \varphi$
Negation Normal Form

• Negation only on atomic propositions:
  \( \neg (\phi \cup \psi) = \neg \phi \mathbin{R} \neg \psi \)
  \( \neg (\phi \mathbin{R} \psi) = \neg \phi \cup \neg \psi \)
  \( \neg G(\phi) = F(\neg \phi) \)
  \( \neg F(\phi) = G(\neg \phi) \)

Example

\( \neg G(p) \lor F(\neg(a \cup b \land G c)) = F(\neg p) \lor F(\neg a \mathbin{R} \neg b \lor F \neg c) \)
Different Names/Symbols

- $X \varphi$: ○ (circle), next
- $\varphi U \psi$: until, until!,
- $\varphi R \psi$: release, weakuntil (W), before
- $F \varphi$: ◊ (diamond), eventually, eventually!,
- $G \varphi$: □ (box), always, never,
LTL Model Checking

• We want to check

\[ K \models \varphi \]

i.e., all paths/computations of K satisfy \( \varphi \)

\[ L(K) \subseteq L(\varphi) \]

\( L(K) \)…language of K (set of all paths of K)

\( L(\varphi) \)…language of \( \varphi \) (set of all paths sat. \( \varphi \))
LTL Model Checking

$L(K) \subseteq L(\varphi)$: language inclusion problem

$L(K) \subseteq L(\varphi)$ is true iff no path (execution) of $K$ violates $\varphi$

$L(K) \cap \overline{L(\varphi)} = \emptyset$ (emptiness problem)

We need

- language complement?
- language intersection?
- emptiness check?
LTL Model Checking

$\overline{L(\phi)}$ ?
$L(\phi)$ ... all infinite paths satisfying $\phi$

$\overline{L(\phi)}$ ... all infinite paths that violate $\phi$
$L(\neg \phi)$.. all infinite path at satisfying $\neg \phi$

$L(\phi) = L(\neg \phi)$

How can we represent $L(\neg \phi)$?
Recall: Finite Languages

- Regular expressions
- Finite state automata
- Allow you to express set of finite words (a language of finite words)
Recall: Finite Languages

$L = \{ w \in A^* \mid \text{at least two } a's \} \text{ with } A=\{a,b,c\}$

$(b|c)^*a(b|c)^*a(a|b|c)^*$

Intersection of two finite language?

$L' = \{ w \in A^* \mid \text{at most one } b \} \text{ with } A=\{a,b,c\}$

$(a|c)^* \mid (a|c)^*b(a|c)^*$
Product Automaton
Language Emptiness

Does there exist a word $w$ in $\{a,b,c\}^*$ s.t. run of $A$ on $w$ ends in an accepting state?
Automata for $\omega$-Languages

- Büchi automaton $A= (\Sigma, Q, q_0, \delta, B)$
  - $\Sigma$...finite alphabet
  - $Q$...finite set of states
  - $q_0$...initial state
  - $\delta$...transition relation
  - $B$...accepting (Büchi) states
Run and Acceptance

Given
- a Büchi automaton $A=(\Sigma, Q, q_0, \delta, B)$ and
- an infinite word $w \in \Sigma^\omega$,

a run $\rho$ of $A$ on $w=w_0w_1w_2...$ is
- an infinite sequence $q_0q_1q_2...$ of states, s.t.,
  for all $i \geq 0$, $(q_i, w_i, q_{i+1}) \in \delta$ holds

$\rho$ is accepting if for all $i \geq 0$ exists $j > i$ s.t. $q_j \in B$

A accepts $w$ if there is an accepting run on $w$
Examples

$L_1 = \{ w \mid \text{at least two a's} \}$
$L_2 = \{ w \mid \text{at most one b} \}$
$L_3 = \{ w \mid \text{infinitely many a's} \}$
$L_4 = \{ w \mid \text{only finitely b's} \}$
Examples

$L_1 = \{ \text{w} \mid \text{at least two a's} \}$

$L_2 = \{ \text{w} \mid \text{at most one b} \}$

$L_3 = \{ \text{w} \mid \text{infinitely many a's} \}$

$L_4 = \{ \text{w} \mid \text{only finitely b's} \}$
Example

• Note that we can view a Kripke structure as a simple Büchi automaton.

• $K = (S, T, s_0, A, L)$ corresponds to $A = (\Sigma, Q, q_0, \delta, B)$ with
  - $\Sigma = 2^A$
  - $Q = S$
  - $q_0 = s_0$
  - $(s, a, s')$ in $\delta$ iff $(s, s')$ in $T$ and $a = L(s)$
  - $B = S$
Note that our alphabet is $2^{\{p,q,c\}}$, i.e., it is the set
$\Sigma=\{\emptyset, \{p\}, \{q\}, \{c\}, \{p,c\}, \{p,q\}, \{q,c\}, \{p,q,c\}\}$ which correspond to
$\Sigma=\{a,b,c,d,e,d,f,g\}$
Generalized Büchi Automata

- Generalized Büchi automaton $A = (\Sigma, Q, q_0, \delta, F)$
  - $\Sigma$...finite alphabet
  - $Q$...finite set of states
  - $q_0$...initial state
  - $\delta$...transition relation
  - $F = \{B_1, \ldots, B_d\}$...set of sets of Büchi states

- Run $q_0 q_1 q_2 \ldots$ is accepting if for all $1 \leq k \leq d$, for all $i \geq 0$ exists $j > i$ s.t. $q_j \in B_k$
GBA to BA

• Idea: pick some order and satisfy Büchi conditions according to this order

• Counting construction (using layers)
  – make $d+1$ copies $(C_1,\ldots,C_{d+1})$ of $A$
  – move from $i$ to $i+1$ copy if condition $i$ is satisfied
  – from copy $C_{d+1}$ move always back to $C_1$
  – all states in Copy $d+1$ are Büchi states
Example

$B_1 = \{1\}, \ B_2 = \{2\}$

Level 1

Level 2

Level 3
Closure Properties

• Are BA closed under conjunction?
• Are BA closed under disjunction?
• Are BA closed under complementation?
Closure Properties

• Are BA closed under conjunction
  – product automaton (GBA)
  – translate GBA to BA

• Are BA closed under disjunction
  – use non-determinism

• Are BA closed under complementation
  – true but difficult to prove
Emptiness

Given a Büchi automaton $A$, does there exist an infinite word $w$ that is accepted by $A$?

= Does there exist an accepting run of $A$?

= Does there exist an run that visits the Büchi states infinitely often?

= Does there exist an SCC that is reachable from $q_0$ and intersects the Büchi states?

We already know how to compute SCCs!
LTL Model Checking

1) Negate $\varphi$ and put $\neg \varphi$ into NNF (\textit{negation normal form}: negation only on atomic propositions)

2) Build an automaton $A_{\neg \varphi}$ that recognizes exactly those paths (words) for which $\neg \varphi$ holds.

3) Taken the product of $A_{\neg \varphi}$ and $K$

4) Check if language of product is empty
LTL to BA

Expansion Rules

\[ \varphi \ U \ \psi = \psi \lor (\varphi \land X(\varphi \ U \ \psi)) \]
\[ \varphi \ R \ \psi = \psi \land (\varphi \lor X(\varphi \ R \ \psi)) \]
\[ G \ \psi = \psi \land X(G \ \psi) \]
\[ F \ \psi = \psi \lor X(F \ \psi) \]

Last two are special cases of first two!
Construction

1) Make initial state $s$ label with $\text{true} \land X\varphi$

2) Expand $\varphi$ until you reach a formula consisting of only formulas beginning with $X$, APs, and Boolean connectors

3) Rewrite in DNF (disjunctive normal form): every disjunct has two parts:
   A) APs that have to be true/false now
   B) Formulas have to be true in next state

4) Introduce a state $s'$ for every disjunct
Construction

5) Add edges from s to every s' for every letter l in $2^{\text{AP}}$ that models A)
6) Do Step 2) to 5) for all s'
7) Add acceptance conditions $B_i$ for every Until-subformula $\varphi_i$
   State s in $B_i$ iff s is not labeled with $X(\varphi_i)$

$\varphi$ is U-formula if U is outermost operator in $\varphi$. Note every F-formula is also an U-formula!
Example

\[ \varphi = G(a \rightarrow Fb) \]

\[ L = \{ w \mid \text{every } a \text{ is eventually followed by } b \} \]

\[ \varphi = G(a \rightarrow Fb) = (a \rightarrow Fb) \land X\varphi \]

\[ = (\neg a \lor Fb) \land X\varphi \]

\[ = (\neg a \lor b \lor XFb) \land X\varphi \]

\[ = (\neg a \land X\varphi) \lor (b \land X\varphi) \lor (XFb \land X\varphi) \]
Example

\[ \varphi = (\neg a \land X\varphi) \lor (b \land X\varphi) \lor (\text{True} \land X(Fb \land \varphi)) \]
Example

$$\varphi = (\neg a \land X\varphi) \lor (b \land X\varphi) \lor (\text{True} \land X(Fb \land \varphi))$$
Example

\[ Fb \land \varphi = Fb \land G(a \rightarrow Fb) \]
\[ = Fb \land (a \rightarrow Fb) \land X \varphi = Fb \land X \varphi \]
\[ = (b \lor XFb) \land X \varphi = (b \land X \varphi) \lor (XFb \land X \varphi) \]
Example

Acceptance condition
Minimization Using Simulation

• Given $A=(\Sigma, Q, q_0, \delta, F)$, a relation $R \subseteq Q \times Q$ is called a direct simulation relation if $(q,r) \in R$ implies
  (1) if $q \in F$ then $r \in F$
  (2) if $(q,a,q') \in \delta$ then
      $\exists r' \in Q (r,a,r') \in \delta$, s.t. $(q',r') \in R$

• If $(q,r)$ and $(r,q)$ in $R$, then $q$ and $r$ are simulation equivalent
Minimization Using Simulation

• Idea: if two q' simulates q, then L(q) ⊆ L(q') (all words accepted when starting from q are also accepted when starting in q')

• One way to minimize:
  – Merge simulation equivalent states (= partition states wrt to R)
  – Remove edges, e.g., (q,a,q') and (q,a,q'') and q' simulates q'', we can remove (q,a,q''), since all words accepted from q'' are also acc. from q'
Example

¬a, Xφ

¬a

T, Xφ

b

T

a, Xφ

b

b

b

T

b

X(Fb ∧ φ)

¬a

T

T

T

T
Example

1, 2, 3 are direct simulation equivalent

1 | Y | Y | Y | N
2 | Y | Y | Y | N
3 | Y | Y | Y | N
4 | Y | Y | Y | Y
Example

$$\varphi = G(a \rightarrow Fb)$$
LTL Model Checking

1) Negate $\varphi$ and put $\neg \varphi$ into NNF
(*negation normal form*: negation only on atomic propositions)

2) Build an automaton $A_{\neg \varphi}$ that recognizes exactly those paths (words) for which $\neg \varphi$ holds.

3) Take the product of $A_{\neg \varphi}$ and $K$

4) Check if language of product is empty
Emptiness Check

- Explicit: SCC
- Symbolic: CTL with fairness
  - $E_c G \text{ true with } c=B$
  - $E_c G\varphi = \forall Z. \ EX (E \ Z \ U (Z \land c))$
  - $E_C G \text{ true with } C=\{B_1,\ldots,B_d\}$

$$E_C G\varphi = \forall Z. \ EX \land_{c\in C} (E \ Z \ U (Z \land c))$$
Exercises

• GOAL [http://goal.im.ntu.edu.tw](http://goal.im.ntu.edu.tw) to
  - translate LTL to Automaton
  - check equivalence between automata
  - minimize automata using simulation relation