Outline

Model checking of CTL

Theorem

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure and $\varphi \in \mathrm{CTL}$ a formula.
The model checking problem $M \models_\exists \varphi$ is decidable in time $\mathcal{O}(|M| \cdot |\varphi|)$

Proof:

Compute $[\![\varphi]\!] = \{s \in S \mid M, s \models \varphi\}$ by induction on the formula.

The set $[\![\varphi]\!]$ is represented by a boolean array: $L[s][\varphi] = \top$ if $s \in [\![\varphi]\!]$.

The labelling ℓ is encoded in L: for $p \in \mathrm{AP}$ we have $L[s][p] = \top$ if $p \in \ell(s)$.

For each $t \in S$, the set $T^{-1}(t)$ is represented as a *list*.

`for all` $t \in S$ `do for all` $s \in T^{-1}(t)$ `do ... od` takes time $\mathcal{O}(|T|)$.

Model checking of CTL

Definition: procedure semantics(φ)

case $\varphi = \neg\varphi_1$
- semantics(φ_1)
- $[\![\varphi]\!] := S \setminus [\![\varphi_1]\!]$ — $\mathcal{O}(|S|)$

case $\varphi = \varphi_1 \vee \varphi_2$
- semantics(φ_1); semantics(φ_2)
- $[\![\varphi]\!] := [\![\varphi_1]\!] \cup [\![\varphi_2]\!]$ — $\mathcal{O}(|S|)$

case $\varphi = EX\varphi_1$
- semantics(φ_1)
- $[\![\varphi]\!] := \emptyset$ — $\mathcal{O}(|S|)$
- for all $t \in [\![\varphi_1]\!]$ do for all $s \in T^{-1}(t)$ do $[\![\varphi]\!] := [\![\varphi]\!] \cup \{s\}$ — $\mathcal{O}(|T|)$

case $\varphi = AX\varphi_1$
- semantics(φ_1)
- $[\![\varphi]\!] := S$ — $\mathcal{O}(|S|)$
- for all $t \notin [\![\varphi_1]\!]$ do for all $s \in T^{-1}(t)$ do $[\![\varphi]\!] := [\![\varphi]\!] \setminus \{s\}$ — $\mathcal{O}(|T|)$

Model checking of CTL

Definition: procedure semantics(φ)

case $\varphi = \mathsf{E}\,\varphi_1 \,\mathsf{U}\, \varphi_2$ — $\mathcal{O}(|S| + |T|)$
- semantics(φ_1); semantics(φ_2)
- $\mathrm{Todo} := [\![\varphi_2]\!]$ // the “todo” set Todo is imlemented with a list — $\mathcal{O}(|S|)$
- $\mathrm{Good} := [\![\varphi_2]\!]$ // the “result” is computed in the array Good — $\mathcal{O}(|S|)$
- while $\mathrm{Todo} \neq \emptyset$ do — $|S|$ times
  - Invariant 1: $[\![\varphi_2]\!] \cup \mathrm{Todo} \subseteq \mathrm{Good} \subseteq [\![\mathsf{E}\,\varphi_1 \,\mathsf{U}\, \varphi_2]\!]$ and
  - Invariant 2: $[\![\varphi_1]\!] \cap T^{-1}(\mathrm{Good} \setminus \mathrm{Todo}) \subseteq \mathrm{Good}$
  - take $t \in \mathrm{Todo}$; $\mathrm{Todo} := \mathrm{Todo} \setminus \{t\}$ — $\mathcal{O}(1)$
  - for all $s \in T^{-1}(t)$ do — $|T|$ times
    - if $s \in [\![\varphi_1]\!] \setminus \mathrm{Good}$ then
      - $\mathrm{Todo} := \mathrm{Todo} \cup \{s\}$; $\mathrm{Good} := \mathrm{Good} \cup \{s\}$ — $\mathcal{O}(1)$
- od
- $[\![\varphi]\!] := \mathrm{Good}$ — $\mathcal{O}(|S|)$

Good is only used to make the invariant clear. It can be replaced by $[\![\varphi]\!]$.

Model checking of CTL

Definition: procedure semantics(φ)

case $\varphi = \mathsf{A}\,\varphi_1\,\mathsf{U}\,\varphi_2$ — $\mathcal{O}(|S|+|T|)$
  semantics(φ_1); semantics(φ_2)
  $\text{Todo} := [\![\varphi_2]\!]$ // the "todo" set Todo is imlemented with a list — $\mathcal{O}(|S|)$
  $\text{Good} := [\![\varphi_2]\!]$ // the "result" is computed in the array Good — $\mathcal{O}(|S|)$
  for all $s \in S$ do $c[s] := |T(s)|$ — $\mathcal{O}(|S|)$
  while $\text{Todo} \neq \emptyset$ do — $|S|$ times
  Invariant 1: $[\![\varphi_2]\!] \cup \text{Todo} \subseteq \text{Good} \subseteq [\![\mathsf{A}\,\varphi_1\,\mathsf{U}\,\varphi_2]\!]$ and
  Invariant 2: $\forall s \in S,\ c[s] = |T(s) \setminus (\text{Good} \setminus \text{Todo})|$ and
  Invariant 3: $[\![\varphi_1]\!] \cap \{s \in S \mid c[s] = 0\} \subseteq \text{Good}$
    take $t \in \text{Todo}$; $\text{Todo} := \text{Todo} \setminus \{t\}$ — $\mathcal{O}(1)$
    for all $s \in T^{-1}(t)$ do — $|T|$ times
      $c[s] := c[s] - 1$ — $\mathcal{O}(1)$
      if $c[s] = 0 \wedge s \in [\![\varphi_1]\!] \setminus \text{Good}$ then
        $\text{Todo} := \text{Todo} \cup \{s\}$; $\text{Good} := \text{Good} \cup \{s\}$ — $\mathcal{O}(1)$
  od
  $[\![\varphi]\!] := \text{Good}$ — $\mathcal{O}(|S|)$

Good is only used to make the invariant clear. It can be replaced by $[\![\varphi]\!]$.

Complexity of CTL

Definition: SAT(CTL)

Input: A formula $\varphi \in \mathrm{CTL}$

Question: Existence of a model M and a state s such that $M, s \models \varphi$?

Theorem: Complexity

- The model checking problem for CTL is PTIME-complete.
- The satisfiability problem for CTL is EXPTIME-complete.

fairness

Example: Fairness

Only fair runs are of interest

- Each process is enabled infinitely often: $\bigwedge_i \mathsf{G}\,\mathsf{F}\,\mathrm{run}_i$
- No process stays ultimately in the critical section: $\bigwedge_i \neg\,\mathsf{F}\,\mathsf{G}\,\mathrm{CS}_i = \bigwedge_i \mathsf{G}\,\mathsf{F}\,\neg\mathrm{CS}_i$

Definition: Fair Kripke structure

$M = (S, T, I, \mathrm{AP}, \ell, F_1, \ldots, F_n)$ with $F_i \subseteq S$.

An infinite run σ is fair if it visits infinitely often each F_i

fair CTL

Definition: Syntax of fair-CTL

$$\varphi ::= \bot \mid p\ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{E}_f\,\mathsf{X}\,\varphi \mid \mathsf{A}_f\,\mathsf{X}\,\varphi \mid \mathsf{E}_f\,\varphi\,\mathsf{U}\,\varphi \mid \mathsf{A}_f\,\varphi\,\mathsf{U}\,\varphi$$

Definition: Semantics as a fragment of CTL*

Let $M = (S, T, I, \mathrm{AP}, \ell, F_1, \ldots, F_n)$ be a fair Kripke structure.

Then, $\mathsf{E}_f\,\varphi = \mathsf{E}(\mathrm{fair} \wedge \varphi)$ and $\mathsf{A}_f\,\varphi = \mathsf{A}(\mathrm{fair} \rightarrow \varphi)$

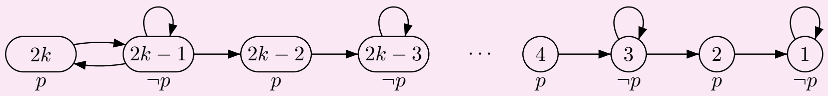
where $\mathrm{fair} = \bigwedge_i \mathsf{G}\,\mathsf{F}\,F_i$

Lemma: CTL_f cannot be expressed in CTL

fair CTL

Proof: CTL_f cannot be expressed in CTL

Consider the Kripke structure M_k defined by:



- $M_k, 2k \models \mathsf{E\,G\,F}\,p$ but $M_k, 2k-2 \not\models \mathsf{E\,G\,F}\,p$
- If $\varphi \in \mathrm{CTL}$ and $|\varphi| \leq m \leq k$ then

$$M_k, 2k \models \varphi \text{ iff } M_k, 2m \models \varphi$$

$$M_k, 2k-1 \models \varphi \text{ iff } M_k, 2m-1 \models \varphi$$

If the fairness condition is $\ell^{-1}(p)$ then $\mathsf{E}_f\,\top$ cannot be expressed in CTL.

Model checking of CTL_f

Theorem

The model checking problem for CTL_f is decidable in time $\mathcal{O}(|M| \cdot |\varphi|)$

Proof: Computation of $\mathrm{Fair} = \{s \in S \mid M, s \models \mathsf{E}_f\,\top\}$

Compute the SCC of M with Tarjan's algorithm (in time $\mathcal{O}(|M|)$).

Let S' be the union of the (non trivial) SCCs which intersect each F_i.

Then, Fair is the set of states that can reach S'.

Note that reachability can be computed in linear time.

Model checking of CTL_f

Proof: Reductions

$\mathsf{E}_f\,\mathsf{X}\,\varphi = \mathsf{E\,X}(\mathrm{Fair} \wedge \varphi)$ and $\mathsf{E}_f\,\varphi\,\mathsf{U}\,\psi = \mathsf{E}\,\varphi\,\mathsf{U}\,(\mathrm{Fair} \wedge \psi)$

It remains to deal with $\mathsf{A}_f\,\varphi\,\mathsf{U}\,\psi$.

We have $\mathsf{A}_f\,\varphi\,\mathsf{U}\,\psi = \neg\,\mathsf{E}_f\,\mathsf{G}\,\neg\psi \wedge \neg\,\mathsf{E}_f(\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi))$

Hence, we only need to compute the semantics of $\mathsf{E}_f\,\mathsf{G}\,\varphi$.

Proof: Computation of $\mathsf{E}_f\,\mathsf{G}\,\varphi$

Let M_φ be the restriction of M to $[\![\varphi]\!]_f$.

Compute the SCC of M_φ with Tarjan's algorithm (in linear time).

Let S' be the union of the (non trivial) SCCs of M_φ which intersect each F_i.

Then, $M, s \models \mathsf{E}_f\,\mathsf{G}\,\varphi$ iff $M, s \models \mathsf{E}\,\varphi\,\mathsf{U}\,S'$ iff $M_\varphi, s \models \mathsf{EF}\,S'$.

This is again a reachability problem which can be solved in linear time.

Büchi automata

Definition:

A Büchi automaton (BA) is a tuple $\mathcal{A} = (Q, \Sigma, I, T, F)$ where

- Q: finite set of states
- Σ: finite set of labels
- $I \subseteq Q$: set of initial states
- $T \subseteq Q \times \Sigma \times Q$: set of transitions (non-deterministic)
- $F \subseteq Q$: set of final (repeated) states

Run: $\rho = q_0, a_0, q_1, a_1, q_2, a_2, q_3, \ldots$ with $(q_i, a_i, q_{i+1}) \in T$ for all $i \geq 0$.

ρ is initial if $q_0 \in I$.

ρ is final (successful) if $q_i \in F$ for infinitely many i's.

ρ is accepting if it is both initial and final.

$$\mathcal{L}(\mathcal{A}) = \{a_0 a_1 a_2 \cdots \in \Sigma^\omega \mid \exists\, \rho = q_0, a_0, q_1, a_1, q_2, a_2, q_3, \ldots \text{ accepting run}\}$$

A language $L \subseteq \Sigma^\omega$ is ω-regular if it can be accepted by some Büchi automaton.

Büchi automata

Examples:

Infinitely many a's:

Finitely many a's:

Whenever a then later b:

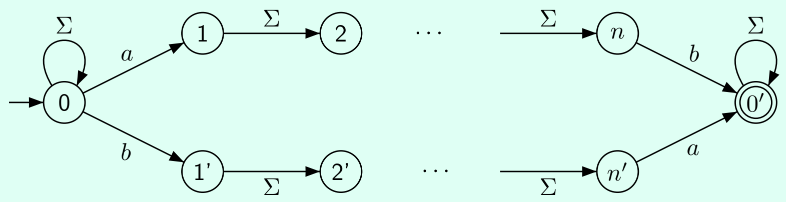
Büchi automata

Properties

Büchi automata are closed under union, intersection, complement.

- Union: trivial
- Intersection: easy (exercise)
- complement: difficult

  Let $L = \Sigma^*(a\Sigma^{n-1}b \cup b\Sigma^{n-1}a)\Sigma^\omega$



  Any non deterministic Büchi automaton for $\Sigma^\omega \setminus L$ has at least 2^n states.

Büchi automata

Theorem: Büchi

Let $L \subseteq \Sigma^\omega$ be a language. The following are equivalent:

- L is ω-regular
- L is ω-rational, i.e., L is a finite union of languages of the form $L_1 \cdot L_2^\omega$ where $L_1, L_2 \subseteq \Sigma^+$ are rational.
- L is MSO-definable, i.e., there is a sentence $\varphi \in \mathrm{MSO}_\Sigma(<)$ such that $L = \mathcal{L}(\varphi) = \{w \in \Sigma^\omega \mid w \models \varphi\}$.

Exercises:

1. Construct a BA for $\mathcal{L}(\varphi)$ where φ is the $\mathrm{FO}_\Sigma(<)$ sentence

$$(\forall x, (P_a(x) \to \exists y > x, P_a(y))) \to (\forall x, (P_b(x) \to \exists y > x, P_c(y)))$$

2. Given BA for $L_1 \subseteq \Sigma^\omega$ and $L_2 \subseteq \Sigma^\omega$, construct BA for

$$\begin{aligned} \mathrm{next}(L_1) &= \Sigma \cdot L_1 \\ \mathrm{until}(L_1, L_2) &= \{uv \in \Sigma^\omega \mid u \in \Sigma^+ \wedge v \in L_2 \wedge \\ &\qquad u''v \in L_1 \text{ for all } u', u'' \in \Sigma^+ \text{ with } u = u'u''\} \end{aligned}$$

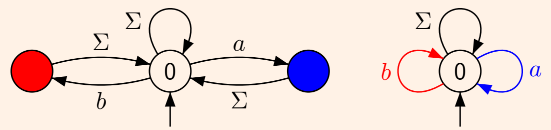
Generalized Büchi automata

Definition: final condition on states or on transitions

$\mathcal{A} = (Q, \Sigma, I, T, F_1, \ldots, F_n)$ with $F_i \subseteq Q$.
An infinite run σ is final (successful) if it visits infinitely often each F_i.

$\mathcal{A} = (Q, \Sigma, I, T, T_1, \ldots, T_n)$ with $T_i \subseteq T$.
An infinite run σ is final if it uses infinitely many transitions from each T_i.

Example: Infinitely many a's and infinitely many b's



Theorem:

1. GBA and BA have the same expressive power.
2. Checking whether a BA or GBA has an accepting run is NLOGSPACE-complete.

Unambiguous or prophetic Büchi automata

Definition: Unambiguous Büchi automata

A BA or GBA $\mathcal{A}$ is unambiguous if every word has at most one accepting run in $\mathcal{A}$.

Definition: Prophetic Büchi automata

A BA or GBA $\mathcal{A}$ is prophetic if every word has exactly one final run in $\mathcal{A}$.

Examples: UBA and PBA

- Finitely many a's.
- $\mathsf{G}(a \to \mathsf{F}\, b)$ with $\Sigma = \{a, b, c\}$.

Theorem: Prophetic Büchi automata (Carton-Michel 2003)

Every ω-regular language can be accepted by a prophetic BA.