Model checking of CTL

Definition: procedure semantics($\varphi$)

\[
\begin{align*}
\text{case } \varphi &= \neg \varphi_1 \\
\text{semantics}(\varphi_1) &\quad \text{O(|$S$|)} \\
[\varphi] &:= S \setminus [\varphi_1] \\
\text{case } \varphi &= \varphi_1 \lor \varphi_2 \\
\text{semantics}(\varphi_1); \text{semantics}(\varphi_2) &\quad \text{O(|$S$|)} \\
[\varphi] &:= [\varphi_1] \cup [\varphi_2] \\
\text{case } \varphi &= EX\varphi_1 \\
\text{semantics}(\varphi_1) &\quad \text{O(|$S$|)} \\
[\varphi] &:= 0 \\
\text{for all } (s, t) \in T \text{ do if } t \in [\varphi_1] \text{ then } [\varphi] := [\varphi] \cup \{s\} &\quad \text{O(|$T$|)} \\
\text{case } \varphi &= AX\varphi_1 \\
\text{semantics}(\varphi_1) &\quad \text{O(|$S$|)} \\
[\varphi] &:= S \\
\text{for all } (s, t) \in T \text{ do if } t \notin [\varphi_1] \text{ then } [\varphi] := [\varphi] \setminus \{s\} &\quad \text{O(|$T$|)}
\end{align*}
\]

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: $L[\varphi][s] = T$ if $s \in [\varphi]$.
The labelling $\ell$ is encoded in $L$: for $p \in AP$ we have $L[p][s] = T$ if $p \in \ell(s)$.
Model checking of CTL

Definition: procedure semantics($\varphi$)

\[
\text{case } \varphi = \varphi_1 \cup \varphi_2 \\
\text{semantics}(\varphi_1), \text{semantics}(\varphi_2) \\
L := \llbracket \varphi_2 \rrbracket \text{ // the "todo" set } L \text{ is implemented with a list} \\
Z := \llbracket \varphi_2 \rrbracket \text{ // the "result" is computed in the array } Z \\
\text{for all } s \in S \text{ do } c[s] := |T(s)| \\
\text{while } L \neq \emptyset \text{ do} \\
\text{Invariant: } L \subseteq Z \text{ and } \\
\forall s \in S, c[s] = |T(s) \setminus (Z \setminus L)| \text{ and} \\
\llbracket \varphi_2 \rrbracket \cup (\llbracket \varphi_1 \rrbracket \cap \{s \in S \mid c[s] = 0\}) \subseteq Z \subseteq [A \varphi_1 \cup \varphi_2] \\
\text{take } t \in L; L := L \setminus \{t\} \\
\text{for all } s \in T^{-1}(t) \text{ do} \\
c[s] := c[s] - 1 \\
\text{if } c[s] = 0 \land s \in \llbracket \varphi_1 \rrbracket \setminus Z \text{ then } L := L \cup \{s\}; Z := Z \cup \{s\} \\
\text{od} \\
\llbracket \varphi \rrbracket := Z
\]

Z is only used to make the invariant clear. It can be replaced by $\llbracket \varphi \rrbracket$.

Complexity of CTL

Definition: SAT(CTL)

Input: A formula $\varphi \in$ 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: $\bigwedge_i GF run_i$
- No process stays ultimately in the critical section: $\bigwedge_i \neg F GC S_i = \bigwedge_i GF \neg CS_i$

Definition: Fair Kripke structure

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

An infinite run $\sigma$ is fair if it visits infinitely often each $F_i$.

fair-CTL

Definition: Syntax of fair-CTL

$\varphi ::= \bot \mid p \in AP \mid \neg \varphi \mid \varphi \lor \varphi \mid E_f X \varphi \mid A_f X \varphi \mid E_f \varphi U \varphi \mid A_f \varphi U \varphi$

Definition: Semantics as a fragment of CTL$

\varphi := \bot \mid p \in AP \mid \neg \varphi \mid \varphi \lor \varphi \mid E_f X \varphi \mid A_f X \varphi \mid E_f \varphi U \varphi \mid A_f \varphi U \varphi$

Remark: $A_f \varphi = \neg E_f \neg \varphi$

Lemma: fair-CTL cannot be expressed in CTL
fair-CTL

Proof: fair-CTL cannot be expressed in CTL
Consider the Kripke structure $M_k$ defined by:

$\begin{array}{c}
2k \\
p
\end{array} \quad \begin{array}{c}
2k - 1 \\
p
\end{array} \quad \begin{array}{c}
2k - 2 \\
p
\end{array} \quad \begin{array}{c}
2k - 3 \\
p
\end{array} \quad \ddots \quad \begin{array}{c}
4 \\
p
\end{array} \\
p \quad \begin{array}{c}
3 \\
p
\end{array} \quad \begin{array}{c}
2 \\
p
\end{array} \quad \begin{array}{c}
1 \\
p
\end{array}
\end{array}$

- $M_k, 2k \models EGFp$ but $M_k, 2k - 2 \not\models EGFp$
- If $\varphi \in \text{CTL}$ and $|\varphi| \leq m \leq k$ then $M_k, 2k \models \varphi$ iff $M_k, 2m \models \varphi$

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

Model checking of fair-CTL

Theorem
The model checking problem for fair-CTL is decidable in time $O(|M| \cdot |\varphi|)$

Proof: Computation of FAIR $= \{ s \in S \mid M, s \models E_f \top \}$
Compute the SCC of $M$ with Tarjan’s algorithm (in time $O(|M|)$).
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 fair-CTL

Proof: Reductions
$E_f \varphi = EX(FAIR \land \varphi)$ and $E_f \varphi U \psi = E \varphi U (FAIR \land \psi)$

It remains to deal with $A_f \varphi U \psi$.
We have $A_f \varphi U \psi = \neg E_f G \neg \psi \land \neg E_f (\neg \psi U (\neg \varphi \land \neg \psi))$
Hence, we only need to compute the semantics of $E_f G \varphi$.

Proof: Computation of $E_f G \varphi$
Let $M_\varphi$ be the restriction of $M$ to $[\varphi]_f$.
Compute the SCC of $M_\varphi$ with Tarjan’s algorithm (in linear time).
Let $S'$ be the union of the (non trivial) SCCs of $M_\varphi$ which intersect each $F_i$.
Then, $M, s \models E_f G \varphi$ iff $M, s \models E \varphi U S'$ iff $M_\varphi, s \models EF S'$.
This is again a reachability problem which can be solved in linear time.

Büchi automata

Definition:
A Büchi automaton (BA) is a tuple $A = (Q, \Sigma, I, T, F)$ where
- $Q$: finite set of states
- $\Sigma$: 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 accepting (repeated, final) 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 $i \geq 0$.
$\rho$ is accepting if $q_0 \in I$ and $q_i \in F$ for infinitely many $i$’s.

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

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

Examples:
- Infinitely many $a$'s:
  - $\exists \varphi \in \text{FO}_2(<)$ such that $L(\varphi) = \{ w \in \Sigma^\omega \mid w \models \varphi \}$
- Finitely many $a$'s:
  - $L$ is $\omega$-regular.
- Whenever $a$ then later $b$:

**Properties**
- Büchi automata are closed under union, intersection, complement.
  - Union: trivial
  - Intersection: easy (exercise)
  - Complement: difficult

Let $L = \{ w \in \Sigma^\omega \mid w \models \varphi \}$ for $\varphi \in \text{MSO}_2(<)$.

**Exercises:**
1. Construct a BA for $L(\varphi)$ where $\varphi$ is the $\text{FO}_2(<)$ sentence
   
   $\forall x, (P_a(x) \rightarrow \exists y > x, P_b(y))) \rightarrow (\forall x, (P_b(x) \rightarrow \exists y > x, P_c(y)))$

2. Given BA for $L_1 \subseteq \Sigma^\omega$ and $L_2 \subseteq \Sigma^\omega$, construct BA for
   $\text{next}(L_1) = \Sigma \cdot L_1$
   $\text{SUntil}(L_1, L_2) = \{ uv \in \Sigma^\omega \mid u \in \Sigma^+ \land v \in L_2 \land u''v \in L_1 \text{ for all } u', u'' \in \Sigma^+ \text{ with } u = u'u'' \}$

---

**Generalized Büchi automata**

**Definition:** acceptance on states or on transitions

$A = (Q, \Sigma, I, T, F_1, \ldots, F_n)$ with $F_i \subseteq Q$.

An infinite run $\sigma$ is successful if it visits infinitely often each $F_i$.

$A = (Q, \Sigma, I, T, T_1, \ldots, T_n)$ with $T_i \subseteq T$.

An infinite run $\sigma$ is successful 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.
**Büchi automata with output**

**Definition:** SBT: Synchronous (letter to letter) Büchi transducer

Let $A$ and $B$ be two alphabets.

A synchronous Büchi transducer from $A$ to $B$ is a tuple $A = (Q, A, I, T, F, \mu)$ where $(Q, A, I, T, F)$ is a Büchi automaton (input) and $\mu : T \rightarrow B$ is the output function.

It computes the relation $[A] = \{(u, v) \in A^* \times B^* \mid \exists \rho = q_0, a_0, q_1, a_1, q_2, a_2, q_3, \ldots \text{ accepting run with } u = a_0a_1a_2 \ldots \text{ and } v = \mu(\tau_0)\mu(\tau_1)\mu(\tau_2) \ldots \text{ where } \tau_i = (q_i, a_i, q_{i+1})\}$

If $(Q, A, I, T, F)$ is unambiguous then $[A] : A^* \rightarrow B^*$ is a (partial) function.

We will also use SGBT: synchronous transducers with generalized Büchi acceptance.

**Example:** Left shift with $A = B = \{a, b\}$

```
\begin{tikzpicture}
    \node (1) at (0,0) {$1$};
    \node (2) at (1,0) {$2$};
    \path[->] (1) edge[above] node {$a/a$} (2)
          edge[below] node {$b/a$} (2)
    (2) edge[above] node {$a/b$} (2)
          edge[below] node {$b/b$} (2);
\end{tikzpicture}
```

**Product of Büchi transducers**

**Definition:** Product

Let $A$, $B$, $C$ be alphabets.

Let $A = (Q, A, I, T, (F_i), \mu)$ be a SGBT from $A$ to $B$.

Let $A' = (Q', A', I', T', (F'_i'), \mu')$ be an SGBT from $A$ to $C$.

Then $A \times A' = (Q \times Q', A \times A', I \times I', T \times T'$, $(F_i \times F'_i), (Q \times F'_i), \mu' \circ \mu)$ is an SGBT from $A \times A'$ to $B \times C$.

**Proposition:** Product

We identify $(B \times C)^* \times (B^* \times C^*)$.

1. We have $[A \times A'] = \{(u, v, v') \mid (u, v) \in [A] \text{ and } (u, v') \in [A']\}$.

2. If $(Q, A, I, T, (F_i), \mu)$ and $(Q', A', I', T', (F'_i'), \mu')$ are unambiguous then $(Q \times Q', A \times A', I \times I', T \times T', (F_i \times F'_i), (Q \times F'_i), \mu' \circ \mu)$ is also unambiguous.

Then, $\forall u \in A^*$ we have $[A \times A'](u) = ([A](u), [A'](u))$.

**Composition of Büchi transducers**

**Definition:** Composition

Let $A$, $B$, $C$ be alphabets.

Let $A = (Q, A, I, T, (F_i), \mu)$ be an SGBT from $A$ to $B$.

Let $A' = (Q', B, I', T', (F'_i'), \mu')$ be an SGBT from $B$ to $C$.

Then $A \cdot A' = (Q \times Q', A, I \times I', T \times T', (F_i \times F'_i), (Q \times F'_i), \mu' \circ \mu)$ is defined by:

$\tau'' = (p, p') \overset{a}{\rightarrow} (q, q') \in T'' \text{ and } \mu''(\tau'') = (b, c)$

iff

$\tau = p \overset{a}{\rightarrow} q \in T \text{ and } b = \mu(\tau) \text{ and } \tau' = p' \overset{a}{\rightarrow} q' \in T' \text{ and } c = \mu'(\tau')$.

$A \cdot A'$ is an SGBT from $A$ to $C$.

When the transducers define functions, we also denote the composition by $A' \circ A$.

**Proposition:** Composition

1. We have $[A \cdot A'] = [A] \cdot [A']$.

2. If $(Q, A, I, T, (F_i), \mu)$ and $(Q', B, I', T', (F'_i), \mu')$ are unambiguous then $(Q \times Q', A, I \times I', T \times T', (F_i \times F'_i), (Q \times F'_i), \mu' \circ \mu)$ is also unambiguous.

Then, $\forall u \in A^*$ we have $[A' \circ A](u) = [A']([A](u))$.

**Subalphabets of $\Sigma = 2^{AP}$**

**Definition:**

For a propositional formula $\xi$ over $AP$, we let $\Sigma_\xi = \{a \in \Sigma \mid a \models \xi\}$.

For instance, for $p, q \in AP$,

\[\Sigma_p = \{a \in \Sigma \mid p \in a\}\] and $\Sigma_{\neg p} = \Sigma \setminus \Sigma_p$,

\[\Sigma_{p \lor q} = \Sigma_p \cup \Sigma_q\] and $\Sigma_{p \land q} = \Sigma_p \cap \Sigma_q$,

\[\Sigma_{p \land \neg q} = \Sigma_p \setminus \Sigma_q\] ....

**Notation:**

In automata, $p \overset{\Sigma_i}{\rightarrow} q$ stands for the set of transitions $\{p\} \times \Sigma_i \times \{q\}$.

To simplify the pictures, we use $p \overset{\xi}{\rightarrow} q$ instead of $p \overset{\Sigma_i}{\rightarrow} q$.

**Example:**

```
\begin{tikzpicture}
    \node (1) at (0,0) {$1$};
    \node (2) at (1,0) {$2$};
    \path[->] (1) edge[above] node {$\neg p \lor q$} (2)
          edge[above] node {$p \land \neg q$} (2)
    (2) edge[above] node {$q$} (2);
\end{tikzpicture}
```
Semantics of LTL with sequential functions

**Definition:** Semantics of \( \varphi \in \text{LTL}(\text{AP, SU, SS}) \)

Let \( \mathcal{A} = 2^{\text{AP}} \) and \( B = \{0, 1\} \).

Define \( \llbracket \varphi \rrbracket : \mathcal{A}^\omega \rightarrow B^\omega \) by \( \llbracket \varphi \rrbracket (u) = b_0 b_1 b_2 \cdots \) with \( b_i = \begin{cases} 1 & \text{if } u, i \models \varphi \smallskip \\ 0 & \text{otherwise}. \end{cases} \)

**Example:**

\[
\llbracket p \text{ SU } q \rrbracket (u) ; \{q\} ; \{p\} ; \{q\} ; \{p, q\} ; \{p, q\} ; \{q\} \smallskip \\
\llbracket Xp \rrbracket (u) ; \{q\} ; \{p\} ; \{q\} ; \{p\} ; \{p, q\} \smallskip \\
\llbracket Fp \rrbracket (u) ; \{q\} ; \{p\} ; \{q\} ; \{p\} ; \{p, q\} \smallskip \\
\]

The aim is to compute \( \llbracket \varphi \rrbracket \) with Büchi transducers.

Synchronous Büchi transducer for \( p \text{ SU } q \)

**Example:** An SBT for \( \llbracket p \text{ SU } q \rrbracket \)

![Diagram of SBT for \( p \text{ SU } q \)]

**Lemma:** The input BA is prophetic

For all \( u = a_0 a_1 a_2 \cdots \in \Sigma^\omega \),

there is a unique accepting run \( \rho = q_0, a_0, q_1, a_1, q_2, a_2, q_3, \ldots \) of \( A \) on \( u \).

The run \( \rho \) satisfies for all \( i \geq 0 \),

\[
q_i = \begin{cases} 1 & \text{if } u, i \models q \smallskip \\ 2 & \text{if } u, i \models \neg q \land (p \text{ U } q) \smallskip \\ 3 & \text{if } u, i \models \neg (p \text{ U } q) \end{cases} \]

Special cases of Until: Future and Next

**Example:** \( Fq = T \text{ U } q \) and \( Xq = \downarrow \text{ SU } q \)

![Diagram of SBTs for \( Fq \) and \( Xq \)]

**Exercise:** Give SBT’s for the following formulae:

\( p \text{ U } q, Gq, p \text{ R } q, p \text{ SR } q, p \text{ S } q, p \text{ SS } q, G(p \to Fq) \).

From LTL to Büchi automata

**Definition:** SBT for LTL modalities

- \( A_T \) from \( \Sigma \) to \( B = \{0, 1\} \):
  - \( \llbracket T \rrbracket : \Sigma^\omega \rightarrow B^\omega \) with \( b_i = \begin{cases} 1 & \text{if } u, i \models T \smallskip \\ 0 & \text{otherwise} \end{cases} \)

- \( A_p \) from \( \Sigma \) to \( B = \{0, 1\} \):
  - \( \llbracket p \rrbracket : \Sigma^\omega \rightarrow B^\omega \) with \( b_i = \begin{cases} 1 & \text{if } u, i \models p \smallskip \\ 0 & \text{otherwise} \end{cases} \)

- \( A_\neg \) from \( B \) to \( B \):
  - \( \llbracket \neg \rrbracket : B^\omega \rightarrow B^\omega \) with \( b_i = \begin{cases} 1 & \text{if } u, i \models \neg \smallskip \\ 0 & \text{otherwise} \end{cases} \)

- \( A_U \) from \( B^2 \) to \( B \):
  - \( \llbracket U \rrbracket : B^2 \rightarrow B^\omega \) with \( b_i = \begin{cases} 1 & \text{if } u, i \models U \smallskip \\ 0 & \text{otherwise} \end{cases} \)

- \( A_A \) from \( B^2 \) to \( B \):
  - \( \llbracket A \rrbracket : B^2 \rightarrow B^\omega \) with \( b_i = \begin{cases} 1 & \text{if } u, i \models A \smallskip \\ 0 & \text{otherwise} \end{cases} \)
From LTL to Büchi automata

Definition: SBT for LTL modalities (cont.)

\[ \mathcal{A}_{SU} \text{ from } \mathbb{B}^2 \text{ to } B: \]

- \( 0, 1/1 \)
- \( 1, 1/1 \)
- \( 0, 1/0 \)
- \( 1, 0/1 \)
- \( 0, 0/0 \)
- \( 1, 0/0 \)

\[ \mathcal{A}_{SS} \text{ from } \mathbb{B}^2 \text{ to } B: \]

- \( 0, 0/0 \)
- \( 1, 0/0 \)
- \( 0, 0/1 \)
- \( 1, 0/1 \)
- \( 0, 1/0 \)
- \( 1, 1/1 \)

Useful simplifications

Reducing the number of temporal subformulae

- \( (X \varphi) \land (X \psi) \equiv X(\varphi \land \psi) \)
- \( (X \varphi) \lor (X \psi) \equiv X(\varphi \lor \psi) \equiv \varphi SU \psi \)
- \( (G \varphi) \land (G \psi) \equiv G(\varphi \land \psi) \equiv G(\varphi \lor \psi) \)
- \( (\varphi_1 U \psi) \land (\varphi_2 U \psi) \equiv (\varphi_1 \land \varphi_2) U \psi \)
- \( (\varphi_1 U \psi_1) \lor (\varphi U \psi_2) \equiv (\varphi U \psi_1) \lor (\varphi U \psi_2) \equiv U(\varphi_1 \lor \varphi_2) \)

Merging equivalent states

Let \( \mathcal{A} = (Q, \Sigma, I, T, T_1, \ldots, T_n, \mu) \) be an SGBT and \( s_1, s_2 \in Q \).
We can merge \( s_1 \) and \( s_2 \) if they have the same outgoing transitions:
\[ \forall a \in \Sigma, \forall s \in Q, \]

\[ (s_1, a, s) \in T \iff (s_2, a, s) \in T \]

and \( (s_1, a, s) \in T_i \iff (s_2, a, s) \in T_i \) for all \( 1 \leq i \leq n \)

and \( \mu(s_1, a, s) = \mu(s_2, a, s) \)

From LTL to Büchi automata

Definition: Translation from LTL to SGBT

For each \( \xi \in \text{LTL}(\text{AP}, \text{SU}, \text{SS}) \) we define inductively an SGBT \( \mathcal{A}_\xi \) as follows:

- \( \mathcal{A}_T \) and \( \mathcal{A}_p \) for \( p \in \text{AP} \) are already defined
- \( \mathcal{A}_{\neg \varphi} = \mathcal{A}_\varphi \circ \mathcal{A}_p \)
- \( \mathcal{A}_{\varphi \lor \psi} = \mathcal{A}_\varphi \circ (\mathcal{A}_\varphi \times \mathcal{A}_\psi) \)
- \( \mathcal{A}_{\varphi \land \psi} = \mathcal{A}_{\varphi \lor \psi} \circ (\mathcal{A}_p \times \mathcal{A}_p) \)
- \( \mathcal{A}_{\varphi SU} = \mathcal{A}_{\varphi SU} \circ (\mathcal{A}_p \times \mathcal{A}_p) \)

Theorem: Correctness of the translation

For each \( \xi \in \text{LTL}(\text{AP}, \text{SU}, \text{SS}) \), we have \([\mathcal{A}_\xi] = [\xi]\).

Moreover, the number of states of \( \mathcal{A}_\xi \) is at most \( 2^{|\xi|_{\text{SS}}} \cdot 3^{|\xi|_{\text{SU}}} \)

where \( |\xi|_{\text{SS}} \) (resp. \( |\xi|_{\text{SU}} \)) is the number of SS (resp. SU) occurring in \( \xi \).

Remark:

- If a subformula \( \varphi \) occurs several times in \( \xi \), we only need one copy of \( \mathcal{A}_\varphi \).
- We may also use automata for other modalities: \( \mathcal{A}_X, \mathcal{A}_U, \mathcal{A}_F, \ldots \)

Other constructions

- Tableau construction. See for instance [15, Wolper 85]
  + : Easy definition, easy proof of correctness
  + : Works both for future and past modalities
  – : Inefficient without strong optimizations
- Using Very Weak Alternating Automata [16, Gastin & Oddoux 01].
  + : Very efficient
  – : Only for future modalities
Online tool: http://www.lsv.ens-cachan.fr/~gastin/ltl2ba/
- Using reduction rules [6, Demri & Gastin 10].
  + : Efficient and produces small automata
  + : Can be used by hand on real examples
  – : Only for future modalities
- The domain is still very active.
**Satisfiability for LTL over $(\mathbb{N}, <)$**

Let $\text{AP}$ be the set of atomic propositions and $\Sigma = 2^\text{AP}$.

**Definition:** Satisfiability problem

**Input:** A formula $\varphi \in \text{LTL}(\text{AP}, \text{SU}, \text{SS})$

**Question:** Existence of $w \in \Sigma^\omega$ and $i \in \mathbb{N}$ such that $w, i \vDash \varphi$.

**Definition:** Initial Satisfiability problem

**Input:** A formula $\varphi \in \text{LTL}(\text{AP}, \text{SU}, \text{SS})$

**Question:** Existence of $w \in \Sigma^\omega$ such that $w, 0 \vDash \varphi$.

**Remark:** $\varphi$ is satisfiable iff $F \varphi$ is initially satisfiable.

**Definition:** (Initial) validity

$\varphi$ is valid iff $\neg \varphi$ is not satisfiable.

**Theorem [10, Sistla, Clarke 85], [9, Lichtenstein & Pnueli 85]**

The satisfiability problem for LTL is PSPACE-complete.

---

**Model checking for LTL**

**Definition:** Model checking problem

**Input:** A Kripke structure $M = (S, T, I, \text{AP}, \ell)$

A formula $\varphi \in \text{LTL}(\text{AP}, \text{SU}, \text{SS})$

**Question:** Does $M \vDash \varphi$?

- **Universal MC:** $M \vDash \varphi$ if $\ell(\sigma), 0 \vDash \varphi$ for all initial infinite runs of $M$.
- **Existential MC:** $M \vDash \exists \varphi$ if $\ell(\sigma), 0 \vDash \varphi$ for some initial infinite run of $M$.

$M \vDash \varphi$ iff $M \not\vDash \neg \varphi$

**Theorem [10, Sistla, Clarke 85], [9, Lichtenstein & Pnueli 85]**

The Model checking problem for LTL is PSPACE-complete.

---

**MC$^3$(SU) $\leq_P$ SAT(SU) [10, Sistla & Clarke 85]**

Let $M = (S, T, I, \text{AP}, \ell)$ be a Kripke structure and $\varphi \in \text{LTL}(\text{AP}, \text{SU})$

Introduce new atomic propositions: $\text{AP}_S = \{a_t \mid s \in S\}$

Define $\text{AP}^\prime = \text{AP} \cup \text{AP}_S$, $\Sigma' = 2^{\text{AP}^\prime}$, $\pi : \Sigma^\omega \rightarrow \Sigma^\omega$ by $\pi(a) = a \cap \text{AP}$.

Let $w \in \Sigma^\omega$. We have $w \vDash \varphi$ iff $\pi(w) \vDash \varphi$.

Define $\psi_M \in \text{LTL}(\text{AP}^\prime, \text{X}, \text{F})$ of size $O(|M|^2)$ by

$$
\psi_M = \left( \bigvee_{s \in S} a_s \right) \land \left( \bigvee_{s \in S} \left( \bigwedge_{t \in S} \neg a_t \land \bigwedge_{p \in \ell(s)} p \land \bigwedge_{t \in F(s)} \text{X} a_t \right) \right)
$$

Let $w = a_0a_1a_2 \cdots \in \Sigma^\omega$. Then, $w \vDash \psi_M$ iff there exists an initial infinite run $\sigma = a_0a_1a_2 \cdots$ of $M$ such that $\ell(\sigma) = \pi(w)$ and $a_i \cap \text{AP}_S = \{a_t \mid s \}$ for all $i \geq 0$.

Therefore, $M \vDash \exists \varphi$ iff $\psi_M \land \varphi$ is satisfiable

$M \vDash \exists \varphi$ iff $\psi_M \land \neg \varphi$ is not satisfiable

**Remark:** we also have $\text{MC}^3(\text{X}, \text{F}) \leq_P \text{SAT}(\text{X}, \text{F})$.

---

**QBF Quantified Boolean Formulae**

**Definition:** QBF

**Input:** A formula $\gamma = Q_1x_1 \cdots Q_nx_n \gamma'$ with $\gamma' = \bigwedge_{1 \leq i \leq m} a_{i,j}$ $Q_i \in \{\forall, \exists\}$ and $a_{i,j} \in \{x_i, \neg x_i, \ldots, x_n, \neg x_n\}$.

**Question:** Is $\gamma$ valid?

**Definition:**

An assignment of the variables $\{x_1, \ldots, x_n\}$ is a word $v = v_1 \cdots v_n \in \{0, 1\}^n$. We write $v[i]$ for the prefix of length $i$.

Let $V \subseteq \{0, 1\}^n$ be a set of assignments.

- $V$ is valid (for $\gamma'$) if $v \vDash \gamma'$ for all $v \in V$.
- $V$ is closed (for $\gamma$) if $\forall V \subseteq \{0, 1\}^n$ s.t. $Q_i = \forall$, $\exists v' \in V$ s.t. $v'[i - 1] = v'[i - 1]$ and $\{v_i, v'_i\} = \{0, 1\}$.

**Proposition:**

$\gamma$ is valid iff $\exists V \subseteq \{0, 1\}^n$ s.t. $V$ is nonempty valid and closed
Complexity of LTL

Theorem: Complexity of LTL

The following problems are PSPACE-complete:
- \( \text{SAT}(\text{LTL}(SU,SS)) \), \( \text{MC}^3(\text{LTL}(SU,SS)) \), \( \text{MC}^3(\text{LTL}(SU)) \)
- \( \text{SAT}(\text{LTL}(X,F)) \), \( \text{MC}^3(\text{LTL}(X,F)) \), \( \text{MC}^3(\text{LTL}(X)) \)
- \( \text{SAT}(\text{LTL}(U)) \), \( \text{MC}^3(\text{LTL}(U)) \), \( \text{MC}^3(\text{LTL}(U)) \)
- \( \text{MC}^3(\text{LTL}(U)) \)
- The restriction of the above problems to a unique propositional variable.

The following problems are NP-complete:
- \( \text{SAT}(\text{LTL}(F)) \), \( \text{MC}^3(\text{LTL}(F)) \)

\[ \text{Complexity of } \text{CTL}^* \]

Proof:
For \( \psi \in \text{LTL} \), let \( \text{MC}^3_{\text{LTL}}(M,t,\psi) \) be the function which computes in polynomial space whether \( M,t \models \exists ! \psi \), i.e., if \( M,t \models E \psi \).

Let \( M = (S,T,I,AP,t) \) be a Kripke structure, \( s \in S \) and \( \varphi \in \text{CTL}^* \).
Replacing \( A \psi \) by \( \neg E \neg \psi \) we assume \( \varphi \) only contains the existential path quantifier.

\[ \text{MC}^3_{\text{CTL}^*} \text{ in PSPACE} \]

Proof:
For \( \psi \in \text{LTL} \), let \( \text{MC}^3_{\text{LTL}}(M,t,\psi) \) be the function which computes in polynomial space whether \( M,t \models \exists ! \psi \), i.e., if \( M,t \models E \psi \).

Let \( M = (S,T,I,AP,t) \) be a Kripke structure, \( s \in S \) and \( \varphi \in \text{CTL}^* \).
Replacing \( A \psi \) by \( \neg E \neg \psi \) we assume \( \varphi \) only contains the existential path quantifier.

\[ \text{MC}^3_{\text{CTL}^*}(M,s,\varphi) \]

If \( E \) does not occur in \( \varphi \) then return \( \text{MC}^3_{\text{LTL}}(M,s,\varphi) \) fi
Let \( E \psi \) be a subformula of \( \varphi \) with \( \psi \in \text{LTL} \).
Let \( e_\psi \) be a new propositional variable
Define \( \ell' : S \to 2^{AP'} \) with \( AP' = AP \cup \{e_\psi\} \) by
\[ \ell'(t) \cap AP = \ell(t) \text{ and } e_\psi \in \ell'(t) \text{ iff } \text{MC}^3_{\text{LTL}}(M,t,\psi) \]
Let \( M' = (S,T,I,AP',\ell') \)
Let \( \varphi' = \varphi[e_\psi/E \psi] \) be obtained from \( \varphi \) by replacing each \( E \psi \) by \( e_\psi \).
Return \( \text{MC}^3_{\text{CTL}^*}(M',s,\varphi') \)
Satisfiability for $\text{CTL}^*$

**Definition:** Satisfiability problem for $\text{CTL}^*$

**Input:** A formula $\varphi \in \text{CTL}^*$

**Question:** Existence of a model $M$, a run $\sigma$, a position $i$ such that $M, \sigma, i \models \varphi$?

**Definition:** Initial Satisfiability problem for $\text{CTL}^*$

**Input:** A formula $\varphi \in \text{CTL}^*$

**Question:** Existence of a model $M$ and a run such that $M, 0 \models \varphi$?

**Theorem**

The (initial) satisfiability problem for $\text{CTL}^*$ is 2-EXPTIME-complete

---

**Some References**

Checking that finite state concurrent programs satisfy their linear specification.

The tableau method for temporal logic: An overview,

The complexity of propositional linear temporal logic.

Fast LTL to Büchi automata translation.

*Specification and Verification using Temporal Logics.*
In Modern applications of automata theory, IISc Research Monographs 2.

http://www.lsv.ens-cachan.fr/~gastin/mes-publis.php