**Outline**

Introduction

Models

Temporal Specifications

- General Definitions
- (Linear) Temporal Specifications
- Branching Temporal Specifications
- CTL
- $\mathcal{CTL}^*$
- CTL

Satisfiability and Model Checking

More on Temporal Specifications

---

**Static and dynamic properties**

Example: Static properties

- Mutual exclusion
- Safety properties are often static.
  They can be reduced to reachability.

Example: Dynamic properties

- Every elevator request should be eventually granted.

- The elevator should not cross a level for which a call is pending without stopping.

---

**Temporal Structures**

Definition: Flows of time

A *flow of time* is a strict order $(T, <)$ where $T$ is the nonempty set of *time points* and $<$ is an irreflexive transitive relation on $T$.

Example: Flows of time

- $\{(0, \ldots, n), <\}$: Finite runs of sequential systems.
- $(\mathbb{N}, <)$: Infinite runs of sequential systems.
- $(\mathbb{R}, <)$: runs of real-time sequential systems.
- *Trees*: Finite or infinite run-trees of sequential systems.
- *Mazurkiewicz traces*: runs of distributed systems (partial orders).
- and also $(\mathbb{Z}, <)$ or $(\mathbb{Q}, <)$ or $(\mathbb{R}^2, <)$, ...

Definition: Temporal Structures

Let $AP$ be a set of atoms (atomic propositions).

A *temporal structure* over a class $C$ of time flows and $AP$ is a triple $(\mathbb{T}, <, h)$ where $(\mathbb{T}, <)$ is a time flow in $C$ and $h : AP \rightarrow 2^T$ is an assignment.

If $p \in AP$ then $h(p) \subseteq \mathbb{T}$ gives the time points where $p$ holds.

---

**Linear behaviors and specifications**

Let $M = (S, T, I, AP, \ell)$ be a Kripke structure.

Definition: Runs as temporal structures

An infinite run $\sigma = s_0s_1s_2 \cdots$ of $M$ with $(s_i, s_{i+1}) \in T$ for all $i \geq 0$ defines a *linear* temporal structure $\ell(\sigma) = (\mathbb{N}, <, h)$ where $h(p) = \{i \in \mathbb{N} | p \in \ell(s_i)\}$.

Such a temporal structure can be seen as an infinite word over $\Sigma = 2^{AP}$:

$\ell(\sigma) = \ell(s_0)\ell(s_1)\ell(s_2)\cdots = (\mathbb{N}, <, w)$ with $w(i) = \ell(s_i) \in \Sigma$.

Linear specifications only depend on runs.

Example: The printer manager is fair.

- On each run, whenever some process requests the printer, it eventually gets it.

Remark:

- Two Kripke structures having the same linear temporal structures satisfy the same linear specifications.
Branching behaviors and specifications

The system has an infinite active run, but it may always reach an inactive state.

Definition: Computation-tree or run-tree: unfolding of the TS

Let \( M = (S, T, I, AP, \ell) \) be a Kripke structure. Wlog. \( I = \{s_0\} \) is a singleton. Let \( D \) be a finite set with \( |D| \) the outdegree of the transition relation \( T \).

The computation-tree of \( M \) is an unordered tree \( t : D^* \rightarrow S \) (partial map) s.t.:
- \( t(\varepsilon) = s_0 \),
- For every node \( u \in \text{dom}(t) \) labelled \( s = t(u) \), if \( T(s) = \{s_1, \ldots, s_k\} \) then \( u \) has exactly \( k \) children which are labelled \( s_1, \ldots, s_k \).

Associated temporal structure \( \ell(t) = (\text{dom}(t), <, h) \) where:
- \( < \) is the strict prefix relation over \( D^* \),
- \( h(p) = \{u \in \text{dom}(t) | p \in \ell(t(u))\} \).

(Linear) runs of \( M \) are branches of the computation-tree \( t \).

First-order vs Temporal

First-order logic
- \( \text{FO}(<) \) has a good expressive power
  ... but \( \text{FO}(<) \)-formulae are not easy to write and to understand.
- \( \text{FO}(<) \) is decidable
  ... but satisfiability and model checking are non elementary.

Temporal logics
- no variables: time is implicit.
- quantifications and variables are replaced by modalities.
- Usual specifications are easy to write and read.
- Good complexity for satisfiability and model checking problems.
- Good expressive power.

Linear Temporal Logic (LTL) over \((\mathbb{N},<)\) introduced by Pnueli (1977) as a convenient specification language for verification of systems.

First-order Specifications

Definition: Syntax of \( \text{FO}(<) \)
Let \( P, Q, \ldots \) be unary predicates twinned with atoms \( p, q, \ldots \) in \( \text{AP} \).
Let \( \text{Var} = \{x, y, \ldots\} \) be first-order variables.
\[
\varphi ::= \bot \mid P(x) \mid x = y \mid x < y \mid \neg \varphi \mid \varphi \lor \varphi \mid \exists x \varphi
\]

Definition: Semantics of \( \text{FO}(<) \)
Let \( w = (\mathbb{T}, <, h) \) be a temporal structure.
Precidates \( P, Q, \ldots \) twinned with \( p, q, \ldots \) are interpreted as \( h(p), h(q), \ldots \).
Let \( \nu : \text{Var} \rightarrow \mathbb{T} \) be an assignment of first-order variables to time points.
\[
w, \nu \models P(x) \text{ if } \nu(x) \in h(p)
w, \nu \models x = y \text{ if } \nu(x) = \nu(y)
w, \nu \models x < y \text{ if } \nu(x) < \nu(y)
w, \nu \models \exists x \varphi \text{ if } w, \nu[x \mapsto t] \models \varphi \text{ for some } t \in \mathbb{T}
\]
where \( \nu[x \mapsto t] \) maps \( x \) to \( t \) and \( y \neq x \) to \( \nu(y) \).

Previous specifications can be written in \( \text{FO}(<) \) (except the branching one).

Temporal Specifications

Definition: Syntax of \( \text{TL}(\text{AP}, \text{SU}, \text{SS}) \)
\[
\varphi ::= \bot \mid (p \in \text{AP}) \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \text{ SU } \varphi \mid \varphi \text{ SS } \varphi
\]

Definition: Semantics: \( w = (\mathbb{T}, <, h) \) temporal structure and \( i \in \mathbb{T} \)
\[
w, i \models p \text{ if } i \in h(p)
w, i \models \neg \varphi \text{ if } w, \nu \not\models \varphi
nw, i \models \varphi \text{ SU } \psi \text{ if } w, i \models \varphi \text{ or } w, i \models \psi
nw, i \models \varphi \text{ SS } \psi \text{ if } \exists k \text{ } i < k \text{ and } w, k \models \psi \text{ and } \forall j \text{ } (i < j < k \rightarrow w, j \models \varphi)
w, i \models \varphi \text{ SS } \psi \text{ if } \exists k \text{ } i > k \text{ and } w, k \models \psi \text{ and } \forall j \text{ } (i > j > k \rightarrow w, j \models \varphi)
\]

Previous specifications can be written in \( \text{TL}(\text{AP}, \text{SU}, \text{SS}) \) (except the branching one).
Temporal Specifications

Definition: non-strict versions of until and since

\[ \varphi U \psi \overset{def}{=} \psi \lor (\varphi \land \varphi SU \psi) \quad \varphi S \psi \overset{def}{=} \psi \lor (\varphi \land \varphi SS \psi) \]

\( w, i \models \varphi U \psi \) if \( \exists k \; i \leq k \; \land \; w, k \models \psi \) and \( \forall j \; (i \leq j < k \rightarrow w, j \models \varphi) \)

\( w, i \models \varphi S \psi \) if \( \exists k \; i \geq k \; \land \; w, k \models \psi \) and \( \forall j \; (i \geq j > k \rightarrow w, j \models \varphi) \)

Definition: Derived modalities

\[ X \varphi \overset{def}{=} \bot \land \varphi \quad \text{Next} \quad Y \varphi \overset{def}{=} \bot \land SS \varphi \quad \text{Yesterday} \]

\( w, i \models X \varphi \) if \( \exists k \; i < k \; \land \; w, k \models \varphi \) and \( \forall j \; (i < j < k) \)

\( w, i \models Y \varphi \) if \( \exists k \; i > k \; \land \; w, k \models \varphi \) and \( \forall j \; (i > j > k) \)

\[ F \varphi \overset{def}{=} T U \varphi \quad P \varphi \overset{def}{=} T S \varphi \]

\[ G \varphi \overset{def}{=} \neg F \neg \varphi \quad H \varphi \overset{def}{=} \neg P \neg \varphi \]

\[ \varphi W \psi \overset{def}{=} (G \varphi) \lor (\varphi U \psi) \quad \text{Weak Until} \]

\[ \varphi R \psi \overset{def}{=} (G \psi) \lor (\psi U (\varphi \land \psi)) \quad \text{Release} \]

Discrete linear time flows

Definition: discrete linear time flows \((T, <)\)

A linear time flow is discrete if \( SF \; T \rightarrow X \; T \) and \( SP \; T \rightarrow Y \; T \) are valid formulae.

\((\mathbb{N}, <)\) and \((\mathbb{Z}, <)\) are discrete.

\((\mathbb{Q}, <)\) and \((\mathbb{R}, <)\) are not discrete.

Exercise: For discrete linear time flows \((T, <)\)

\[ \varphi SU \psi \overset{\equiv}{=} X(\varphi U \psi) \]

\[ \varphi SS \psi \overset{\equiv}{=} Y(\varphi S \psi) \]

\[ \neg X \varphi \overset{\equiv}{=} \neg X T \lor X \neg \varphi \]

\[ \neg Y \varphi \overset{\equiv}{=} \neg Y T \lor Y \neg \varphi \]

\[ \neg (\varphi U \psi) \overset{\equiv}{=} (G \neg \psi) \lor (\neg \psi U (\neg \varphi \land \neg \psi)) \]

\[ \neg \psi W (\neg \varphi \land \neg \psi) \]

\[ \neg \varphi R \neg \psi \]

Temporal Specifications

Example: Specifications on the time flow \((\mathbb{N}, <)\)

- Safety: \( G \; \text{good} \)
- MutEx: \( \neg F(\text{crit}_1 \land \text{crit}_2) \)
- Liveness: \( GF \; \text{active} \)
- Response: \( G(\text{request} \rightarrow F \; \text{grant}) \)
- Response': \( G(\text{request} \rightarrow (\neg \text{request \; SU \; grant})) \)
- Release: \( \text{reset \; R \; alarm} \)
- Strong fairness: \( (GF \; \text{request}) \rightarrow (GF \; \text{grant}) \)
- Weak fairness: \( (FG \; \text{request}) \rightarrow (FG \; \text{grant}) \)

Model checking for linear behaviors

Definition: Model checking problem

Input: A Kripke structure \( M = (S, T, I, AP, \ell) \)
A formula \( \varphi \in \text{LTL}(AP, SU, SS) \)

Question: Does \( M \models \varphi \)?

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

\[ M \models \varphi \iff M \not\models \neg \varphi \]

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

The Model checking problem for LTL is PSPACE-complete.
Weaknesses of linear behaviors

Example:
\( \psi: \) Whenever \( p \) holds, it is possible to reach a state where \( q \) holds.
\( \psi \) cannot be checked on linear runs.
We need to consider the computation-trees.

Weaknesses of FO specifications

Example:
\( \psi: \) The system has an infinite active run, but it may always reach an inactive state.
\( \psi \) cannot be expressed in FO.

We need quantifications on runs:
\[ \psi = \text{EG} (\text{Active} \land \text{EF} \neg \text{Active}) \]

We need a temporal logic
- with no explicit variables,
- allowing quantifications over runs,
- usual specifications should be easy to write and read,
- with good complexity for satisfiability and model checking problems,
- with good expressive power.

Computation Tree Logic (CTL*) introduced by Emerson & Halpern (1986).

MSO Specifications

Definition: Syntax of MSO(<)
Let \( P, Q, \ldots \) be unary predicates twinned with atoms \( p, q, \ldots \) in \( \text{AP} \).

\[ \varphi ::= \bot \mid P(x) \mid x = y \mid x < y \mid x \in X \mid \neg \varphi \mid \varphi \lor \varphi \mid \exists x \varphi \mid \exists X \varphi \]

where \( x, y \) are first-order variables and \( X \) is a second-order variable.

Definition: Semantics of MSO(<)
Let \( w = (T, <, h) \) be a temporal structure.
An assignment \( \nu \) maps first-order variables to time points in \( T \)
and second-order variables to sets of time points.
The semantics of first-order constructs is unchanged.

\[ w, \nu \models x \in X \quad \text{if} \quad \nu(x) \in \nu(X) \]
\[ w, \nu \models \exists X \varphi \quad \text{if} \quad w, \nu[X \mapsto T] \models \varphi \text{ for some } T \subseteq T \]

where \( \nu[X \mapsto T] \) maps \( X \) to \( T \) and keeps unchanged the other assignments.

MSO vs Temporal

MSO logic
- MSO(<) has a good expressive power
  ... but MSO(<)-formulae are not easy to write and to understand.
- MSO(<) is decidable on computation trees
  ... but satisfiability and model checking are non elementary.

We need a temporal logic
- with no explicit variables,
- allowing quantifications over runs,
- usual specifications should be easy to write and read,
- with good complexity for satisfiability and model checking problems,
- with good expressive power.

Computation Tree Logic (CTL*) introduced by Emerson & Halpern (1986).
**CTL* (Emerson & Halpern 86)**

**Definition: Syntax of the Computation Tree Logic CTL***

\[ \varphi ::= \perp \mid p \ (p \in \text{AP}) \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid E \varphi \mid A \varphi \]

We may also add the past modality \( S^\bullet \).

**Definition: Semantics of CTL***

Let \( M = (S, T, I, \text{AP}, \ell) \) be a Kripke structure.

Let \( \sigma = s_0 s_1 s_2 \cdots \) be an infinite run of \( M \).

\[
M, \sigma, i \models p \quad \text{if} \quad p \in \ell(s_i) \\
M, \sigma, i \models \varphi \lor \psi \quad \text{if} \quad \exists k > i, \ M, \sigma, k \models \psi \quad \text{and} \quad \forall i < j < k, \ M, \sigma, j \models \varphi \\
M, \sigma, i \models \varphi \quad \text{if} \quad M, \sigma', i \models \varphi \quad \text{for all infinite runs} \ \sigma' \quad \text{such that} \ \sigma'[i] = \sigma[i] \\
M, \sigma, i \models \neg \varphi \quad \text{if} \quad M, \sigma', i \models \varphi \quad \text{for some infinite run} \ \sigma' \quad \text{such that} \ \sigma'[i] = \sigma[i]
\]

where \( \sigma[i] = s_0 \cdots s_i \).

**Remark:**
- \( A \varphi \equiv \neg E \neg \varphi \)
- \( \sigma'[i] = \sigma[i] \) means that future is branching but past is not.

---

**State formulae and path formulae**

**Definition: State formulae**

\( \varphi \in \text{CTL}^* \) is a state formula if \( \forall M, \sigma, \sigma', i, j \) such that \( \sigma(i) = \sigma'(j) \) we have

\[
M, \sigma, i \models \varphi \iff M, \sigma', j \models \varphi
\]

If \( \varphi \) is a state formula and \( M = (S, T, I, \text{AP}, \ell) \), define

\[
\{ \varphi \}^M = \{ s \in S \mid M, s \models \varphi \}
\]

**Example: State formulae**

Atomic propositions are state formulae: \( \{ p \} = \{ s \in S \mid p \in \ell(s) \} \)

State formulae are closed under boolean connectives.

\[
\neg \varphi = S \setminus \{ \varphi \} \\
\varphi_1 \lor \varphi_2 = \{ \varphi_1 \} \cup \{ \varphi_2 \}
\]

Formulae of the form \( E \varphi \) or \( A \varphi \) are state formulae, provided \( \varphi \) is future.

**Definition: Alternative syntax**

State formulae

\[ \varphi ::= \perp \mid p \ (p \in \text{AP}) \mid \neg \varphi \mid \varphi \lor \varphi \mid E \varphi \mid A \varphi \]

Path formulae

\[ \psi ::= \varphi \mid \neg \psi \mid \psi \lor \psi \mid \psi \land \psi \mid E \psi \mid A \psi \]

---

**Model checking of CTL***

**Definition: Existential and universal model checking**

Let \( M = (S, T, I, \text{AP}, \ell) \) be a Kripke structure and \( \varphi \in \text{CTL}^* \) a formula.

\[
M \models \exists \varphi \quad \text{if} \quad M, 0 \models \varphi \quad \text{for some initial infinite run} \ \sigma \quad \text{of} \ M. \\
M \models \forall \varphi \quad \text{if} \quad M, 0 \models \varphi \quad \text{for all initial infinite runs} \ \sigma \quad \text{of} \ M.
\]

**Remark:**
- \( M \models \exists \varphi \iff I \cap \{ \text{E} \varphi \} \neq \emptyset \)
- \( M \models \forall \varphi \iff I \subseteq \{ \text{A} \varphi \} \)
- \( M \models \forall \varphi \iff M \not\models \neg \varphi \)

**Definition: Model checking problems \( \text{MC}^\forall_{\text{CTL}^*} \) and \( \text{MC}^\exists_{\text{CTL}^*} \)**

**Input:** A Kripke structure \( M = (S, T, I, \text{AP}, \ell) \) and a formula \( \varphi \in \text{CTL}^* \)

**Question:** Does \( M \models \forall \varphi \) ?

\( \text{or} \)

Does \( M \models \exists \varphi \)?

**Theorem:**

The model checking problem for CTL* is PSPACE-complete.

*Proof later*
**CTL (Clarke & Emerson 81)**

**Definition: Computation Tree Logic (CTL)**

**Syntax:**

\[ \varphi ::= \bot \mid p \mid \neg \varphi \mid \varphi \lor \psi \mid EX \varphi \mid AX \varphi \mid E \varphi U \psi \mid A \varphi U \psi \]

The semantics is inherited from CTL*.

**Remark:** All CTL formulæ are state formulæ

\[ [\varphi]^M = \{ s \in S \mid M, s \models \varphi \} \]

**Examples: Macros**

- \( EF \varphi = E \top U \varphi \) and \( AG \varphi = \neg E \neg \varphi \)
- \( AF \varphi = A \top U \varphi \) and \( EG \varphi = \neg A \neg \varphi \)
- \( AG(req \rightarrow EF \text{grant}) \)
- \( AG(req \rightarrow AF \text{grant}) \)

**Example:***

\[ [EX p] = [AX p] = [EF p] = [E q U r] = [A q U r] = \]

---

**Definition: Semantics**

All CTL-formulæ are state formulæ. Hence, we have a simpler semantics.

Let \( M = (S, \ell, I, \ell, \ell) \) be a Kripke structure without deadlocks and let \( s \in S \).

\[ s \models p \quad \text{if} \quad p \in \ell(s) \]

\[ s \models EX \varphi \quad \text{if} \quad \exists s' \text{ with } s' \models \varphi \]

\[ s \models AX \varphi \quad \text{if} \quad \forall s \rightarrow s' \text{ we have } s' \models \varphi \]

\[ s \models E \varphi U \psi \quad \text{if} \quad \exists s = s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_k \text{ finite path, with } s_k \models \psi \text{ and } s_j \models \varphi \text{ for all } 0 \leq j < k \]

\[ s \models A \varphi U \psi \quad \text{if} \quad \forall s = s_0 \rightarrow s_1 \rightarrow \cdots \text{ infinite path, } \exists k \geq 0 \text{ with } s_k \models \psi \text{ and } s_j \models \varphi \text{ for all } 0 \leq j < k \]

**Remark: Equivalent formulæ**

- \( AX \varphi = \neg EX \neg \varphi \)
- \( \neg (\varphi U \psi) = G \neg \psi V (\neg \psi U (\neg \varphi \land \neg \psi)) \)
- \( A \varphi U \psi = \neg EG \neg \psi \land \neg E(\neg \psi U (\neg \varphi \land \neg \psi)) \)
- \( AG(req \rightarrow F \text{grant}) = AG(req \rightarrow AF \text{grant}) \)
- \( A \varphi G \varphi = AG \varphi \)
- \( E \varphi G \varphi = EF \varphi \)
- \( E \varphi F \varphi \neq E \varphi G \varphi \)
- \( AF \varphi G \varphi \neq AF \varphi \)
- \( EG \varphi \neq EG \varphi \)
- \( E \varphi G \varphi \neq E \varphi X \varphi \)
Model checking of CTL

Definition: Existential and universal model checking

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

$M \models_\exists \varphi$ if $M, s \models \varphi$ for some $s \in I$.

$M \models_\forall \varphi$ if $M, s \models \varphi$ for all $s \in I$.

Remark:

$M \models_\exists \varphi$ iff $I \cap \llbracket \varphi \rrbracket \neq \emptyset$

$M \models_\forall \varphi$ iff $I \subseteq \llbracket \varphi \rrbracket$

$M \models_\forall \varphi$ iff $M \models_\exists \neg \varphi$

Definition: Model checking problems $\text{MC}_\exists^{\text{CTL}}$ and $\text{MC}_\forall^{\text{CTL}}$

Input: A Kripke structure $M = (S, T, I, AP, \ell)$ and a formula $\varphi \in \text{CTL}$

Question: Does $M \models_\forall \varphi$? or Does $M \models_\exists \varphi$?

Theorem:

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

The model checking problem $M \models_\exists \varphi$ is decidable in time $O(|M| \cdot |\varphi|)$

References

Principles of Model Checking.


Model Checking.


Temporal Verification of Reactive Systems: Safety.

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

Temporal logic: mathematical foundations and computational aspects.

On the temporal analysis of fairness.
In 7th Annual ACM Symposium PoPL’80, 163–173. ACM Press.

Checking that finite state concurrent programs satisfy their linear specification.
In ACM Symposium PoPL’85, 97–107.

The complexity of propositional linear temporal logic.