TEMPORAL LOGIC

1. Introduction to temporal logic
2. Temporal logic and distributed and concurrent systems
3. CTL (Computation Tree Logic)
4. LTL (Linear Temporal Logic)
5. CTL*
6. Validity and satisfiability in CTL and LTL

E. M. Clarke et al.: *Model Checking*, Chapter 3 (pp. 27–33).

E. A. Emerson: *Automated Temporal Reasoning about Reactive Systems*, Section 2 (pp. 3–15).

1. Introduction to Temporal Logic

- Temporal logics are among the most widely applied logics in computer science
- Time interpretation of the possible world semantics:
  possible worlds are seen as possible time points
- Computational interpretation of the possible world semantics:
  possible worlds are seen as possible (global) states of the computation
- Formal model: $\langle S, R, v \rangle$
  where S is the set of possible points/states and sRt is interpreted:
  t is (some) possible future point from s and
  s is (some) possible past point from t.
  Typical requirements for R: transitive, linear/branching, discrete/continuous, ...

Temporal Operators

- $\mathbf{F}Q$ (future)
  $\mathcal{M}, s \Vdash \mathbf{F}Q$ iff $\mathcal{M}, t \Vdash Q$ for some $t \in S$ such that sRt.
- $\mathbf{G}Q = \neg\mathbf{F}\neg Q$ (globally)
- $\mathbf{P}Q$ (past)
  $\mathcal{M}, s \Vdash \mathbf{P}Q$ iff $\mathcal{M}, t \Vdash Q$ for some $t \in S$ such that tRs.
- $\mathbf{H}Q = \neg\mathbf{P}\neg Q$ (historically)
- **Always** $Q = \mathbf{G}Q \wedge Q \wedge \mathbf{H}Q$ (always)
- $\mathbf{X}$ (next):
  In every/some next world?
  Note that the accessibility relations $R_{\mathbf{X}}$ vs. $R_{\mathbf{F}}$ are related.

Binary Temporal Operators

- $\mathbf{U}$ (until):
  $\mathcal{M}, s \Vdash A\mathbf{U}B$ iff
  for some t such that sRt, $\mathcal{M}, t \Vdash B$ and for all $u \in S$,
  if sRu and uRt, then $\mathcal{M}, u \Vdash A$.
  $\Longrightarrow$ $\mathbf{F}B \leftrightarrow (\top\mathbf{U}B)$
- $\mathbf{S}$ (since):
  $\mathcal{M}, s \Vdash A\mathbf{S}B$ iff
  for some t such that tRs, $\mathcal{M}, t \Vdash B$ and for every $u \in S$,
  if uRs and tRu, then $\mathcal{M}, u \Vdash A$.
  $\Longrightarrow$ $\mathbf{P}B \leftrightarrow (\top\mathbf{S}B)$

Dynamic Logic

- For each action a, a modal operator $[a]P$ (P is true after executing action a).
- Composite actions are also allowed:

$a;b$	serial composition
$a \cup b$	nondeterministic choice
a^*	repetition
$P?$	test (if P is true, continue otherwise not)

Example. The following are formulas in dynamic logic:

$[(P?;a) \cup (\neg P?;b)]Q$ ([if P then a else b] Q)

$[(P?;a)^*;\neg P?]Q$ ([while P do a] Q)

2. Temporal Logic and Distributed and Concurrent Systems

Distributed and Concurrent Systems:

- Several distributed and concurrent processes
- Shared resources, coordination, communication
- Continuous operation
- Reactivity and nondeterminism
- Examples: operating systems, communication protocols, device drivers, instrumentation and control systems, ...

Reactive Systems

- Designing reactive systems is challenging:
  - It is hard to reproduce an erroneous execution
  - An execution of the system is an infinite sequence of states.
- Novel design methods are needed:
  - (i) Errors in design should be detected as early as possible in the design cycle.
  - (ii) When reasoning about the correctness of a design, infinite executions need to be handled.

Temporal Logic

- Provides a formal model for the executions of the system
- and a language for specifying requirements for the system.

Example.

- Mutual exclusion: $\mathbf{G}\neg(at_i(m) \wedge at_j(m'))$.
- Partial correctness: If a property P holds in the initial state m_0 of the program, then a property Q holds in the final state m_e: $at(m_0) \wedge P \rightarrow \mathbf{G}(at(m_e) \rightarrow Q)$.
- Total correctness: requires in addition that the program always halts: $at(m_0) \wedge P \rightarrow \mathbf{F}at(m_e)$.
- No unnecessary replies: a reply v_i is given only to a received request p_i: $\mathbf{F}v_i \rightarrow (\neg v_i)\mathbf{U}p_i$.

Applying Temporal Logic

- Proving correctness:
  - The system and requirements are described in temporal logic.
  - Correctness is established by proving (compositionally in temporal logic) that the requirements are logical consequences of the premises describing the executions of the system.
  - This is error prone and hard to automate.
- Program synthesis:
  - Detailed requirements of the system specified in temporal logic.
  - A model for the requirements gives a (skeleton of a) program satisfying the requirements
    (also executable temporal specifications are possible).
  - Easier to automate but still computationally challenging.

Applying Temporal Logic–cont'd

- Model checking:
  - Given a model of the system, check whether the model satisfies a given requirement.
  - Requirements specified using temporal logic
  - Efficient model checkers available.
- CTL and LTL are among the most widely applied temporal logics.

3. CTL (Computation Tree Logic)

- In CTL temporal operators are pairs of
  (i) path quantifiers ($\mathbf{A}/\mathbf{E}$) and
  (ii) temporal operators ($\mathbf{X}/\mathbf{U}/\mathbf{G}/\mathbf{F}$).
- CTL syntax
  - Every atomic proposition is a CTL-formula
  - If P, Q are CTL-formulas, then $P \wedge Q$, $\neg P$, $\mathbf{AX}P$, $\mathbf{A}(P\mathbf{U}Q)$, $\mathbf{E}(P\mathbf{U}Q)$ are CTL-formulas.

Example. CTL-formulas

$$(P \wedge Q) \wedge \neg Q$$
$$\mathbf{AX}(P \wedge \neg Q)$$
$$\mathbf{E}((\mathbf{AX}P)\mathbf{U}Q)$$

CTL Syntax—cont'd

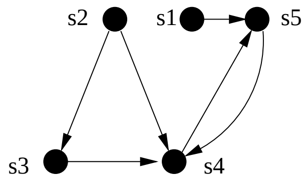
- Note that nesting of temporal operators $\mathbf{X}/\mathbf{U}$ and their Boolean combinations are limited in CTL.
  For example, $\mathbf{AXAX}P$ is a CTL-formula but $\mathbf{AXX}P$ and $\mathbf{A}\neg\mathbf{X}P$ are not.
- Other operator pairs ($\mathbf{EX}, \mathbf{AG}, \mathbf{EG}, \mathbf{AF}, \mathbf{EF}$) can be defined as shorthands using the basic operators ($\mathbf{AX}$, $\mathbf{A}(.\mathbf{U}.)$, $\mathbf{E}(.\mathbf{U}.)$).
- In CTL, executions of the system are seen as a computation tree (more details given in the next lecture) and given a state path quantifiers specify whether the property in question holds for some or all paths (branches) starting from the state.

Example. $\mathbf{AX}P$ (For all paths in the next world P)
$\mathbf{E}(P\mathbf{U}Q)$ (There exists a path where P until Q).

Possible World Semantics

- CTL models are possible world models $\langle S, R, v \rangle$ where the accessibility relation is serial. (Typically in CTL models possible worlds in S are called states of the model).
- Note that the relation R here is that for the operator $\mathbf{X}$.
- A full path is an infinite sequence $s_0, s_1, \ldots$ of states such that for all i: $s_i R s_{i+1}$. (A full path is one of the possible executions from the state s_0).

Example. Consider the model M in the figure.



Examples of full paths:

$s1, s5, s4, s5, s4, \ldots$

$s2, s4, s5, s4, \ldots$

$s2, s3, s4, s5, s4, \ldots$

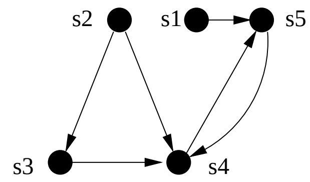
Truth in a Model

Next we defined when a CTL-formula is true in a state s ($\mathcal{M}, s \models P$):

- $\mathcal{M}, s \models P$ iff $v(s, P) = \text{true}$, when P an atomic proposition.
- $\mathcal{M}, s \models \neg P$ iff $\mathcal{M}, s \not\models P$.
- $\mathcal{M}, s \models P \wedge Q$ iff $\mathcal{M}, s \models P$ and $\mathcal{M}, s \models Q$.
- $\mathcal{M}, s \models \mathbf{AX}P$ iff $\mathcal{M}, t \models P$ for all t such that sRt.
- $\mathcal{M}, s \models \mathbf{A}(P\mathbf{U}Q)$ iff for all full paths $(s_0, s_1, \ldots)$ where $s = s_0$, there is some $i \geq 0$ such that $\mathcal{M}, s_i \models Q$ and $\mathcal{M}, s_j \models P$ for all $0 \leq j < i$.
- $\mathcal{M}, s \models \mathbf{E}(P\mathbf{U}Q)$ iff there is a full path $(s_0, s_1, \ldots)$ with $s = s_0$ and with some $i \geq 0$ such that $\mathcal{M}, s_i \models Q$ and $\mathcal{M}, s_j \models P$ for all $0 \leq j < i$.

Example

Consider again the model M:



with a valuation:
$v(P, s_4) = \text{true}$ and otherwise $v(P, s) = \text{false}$;
$v(Q, s_2) = \text{true}$ and otherwise $v(Q, s) = \text{false}$.

1. $\mathcal{M}, s_2 \not\models \mathbf{AX}P$ but $\mathcal{M}, s_3 \models \mathbf{AX}P$.
2. $\mathcal{M}, s_2 \not\models \mathbf{A}(Q\mathbf{U}P)$ but $\mathcal{M}, s_2 \models \mathbf{E}(Q\mathbf{U}P)$.
3. $\mathcal{M}, s_3 \not\models \mathbf{E}(Q\mathbf{U}P)$ but $\mathcal{M}, s_4 \models \mathbf{A}(Q\mathbf{U}P)$.

More Temporal Operators

- We can define further operators as shorthands:

$$\mathbf{EX}P \equiv_{\text{def}} \neg\mathbf{AX}\neg P \qquad \mathbf{AG}P \equiv_{\text{def}} \neg\mathbf{EF}\neg P$$
$$\mathbf{AF}P \equiv_{\text{def}} \mathbf{A}(\top\mathbf{U}P) \qquad \mathbf{EG}P \equiv_{\text{def}} \neg\mathbf{AF}\neg P$$
$$\mathbf{EF}P \equiv_{\text{def}} \mathbf{E}(\top\mathbf{U}P)$$

- Notice the reflexivity and transitivity of the operator $\mathbf{U}$:

  Example. If $\mathcal{M}, s_0 \models P$, then $\mathcal{M}, s_0 \models \mathbf{A}(Q\mathbf{U}P)$ and $\mathcal{M}, s_0 \models \mathbf{E}(Q\mathbf{U}P)$ (and, for example, $\mathcal{M}, s_0 \models \mathbf{AF}P$).
  If $s_0 R s_1$, $s_1 R s_2$ and $\mathcal{M}, s_2 \models P$, then $\mathcal{M}, s_0 \models \mathbf{E}(\top\mathbf{U}P)$ $(= \mathbf{EF}P)$.

4. LTL (Linear Temporal Logic)

- LTL is a linear time temporal logic with operators $\mathbf{X}, \mathbf{U}, \mathbf{G}, \mathbf{F}$.
- Syntax:
  - Every atomic proposition is a LTL-formula
  - If P, Q are LTL-formulas, then $P \wedge Q$, $\neg P$, $\mathbf{X}P$, $P\mathbf{U}Q$ are LTL-formulas.
- Examples: $\neg\mathbf{X}(P \wedge \neg Q)$ and $\mathbf{X}(\mathbf{X}(\mathbf{X}P\mathbf{U}(Q \wedge P)) \wedge P)$.
- For example, operators $\mathbf{G}$ and $\mathbf{F}$ can be defined as shorthands using the basic operators: $\mathbf{F}P \equiv_{\text{def}} \top\mathbf{U}P$ and $\mathbf{G}P \equiv_{\text{def}} \neg\mathbf{F}\neg P$.
- Note that nesting of operators $\mathbf{X}$ and $\mathbf{U}$ and their Boolean combinations are possible: For instance, $(\mathbf{X}(\neg\mathbf{X}P))\mathbf{U}(\mathbf{X}(\mathbf{X}P))$ are LTL-formulas.

LTL Models

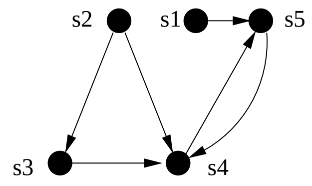
- An LTL model is a possible world model with a serial accessibility relation as for CTL but the formulas are interpreted on full paths (and not on states as in CTL).
- If $x = (s_0, s_1, \ldots)$ is a full path, then $x^i = (s_i, s_{i+1}, \ldots)$ is the suffix of x starting at s_i.

Definition. Let $\mathcal{M}$ be a LTL model and $x = (s_0, s_1, \ldots)$ one of its full paths.

- $\mathcal{M}, x \models P$ iff $v(s_0, P) = \text{true}$ where P is an atomic proposition.
- $\mathcal{M}, x \models \neg P$ iff $\mathcal{M}, x \not\models P$.
- $\mathcal{M}, x \models P \wedge Q$ iff $\mathcal{M}, x \models P$ and $\mathcal{M}, x \models Q$.
- $\mathcal{M}, x \models \mathbf{X}P$ iff $\mathcal{M}, x^1 \models P$.
- $\mathcal{M}, x \models P\mathbf{U}Q$ iff there is some $i \geq 0$ such that $\mathcal{M}, x^i \models Q$ and $\mathcal{M}, x^j \models P$ for all $0 \leq j < i$.

Example

Consider again the model M:



the valuation:
$v(P, s_4) = \text{true}$, otherwise $v(P, s) = \text{false}$;
$v(Q, s_2) = \text{true}$, otherwise $v(Q, s) = \text{false}$.

For full paths $x_1 = (s2, s3, s4, s5, s4, \ldots)$ and $x_2 = (s2, s4, s5, s4, \ldots)$:

1. $M, x_1 \not\models \mathbf{X}P$ but $M, x_2 \models \mathbf{X}P$.
2. $M, x_1 \not\models Q\mathbf{U}P$ but $M, x_2 \models Q\mathbf{U}P$.

More Temporal Operators

- We adopt the following shorthands:

$\mathbf{F}P \equiv_{\text{def}} \top\mathbf{U}P$	$\mathbf{G}P \equiv_{\text{def}} \neg\mathbf{F}\neg P$
$\overset{\infty}{\mathbf{F}}P \equiv_{\text{def}} \mathbf{G}\mathbf{F}P$	$\overset{\infty}{\mathbf{G}}P \equiv_{\text{def}} \mathbf{F}\mathbf{G}P$
$P\mathbf{B}Q \equiv_{\text{def}} \neg((\neg P)\mathbf{U}Q)$	(before)

- Note the reflexivity and transitivity of the operator $\mathbf{U}$:

  Example. If $\mathcal{M}, x \models P$, then $\mathcal{M}, x \models (Q\mathbf{U}P)$.

  If $\mathcal{M}, x \models \mathbf{X}^i P$ for some $i \geq 0$, then $\mathcal{M}, x \models (\top\mathbf{U}P)$.

  In fact, for all $\mathcal{M}, x$ holds, for example: $\mathcal{M}, x \models \mathbf{G}P \rightarrow P$ and $\mathcal{M}, x \models \mathbf{G}P \rightarrow \mathbf{G}\mathbf{G}P$.

5. CTL*

- The idea: CTL* = CTL (state formulas) + LTL (path formulas).
- **CTL*-formulas** are state formulas obtained by the following rules:
  - Every atomic proposition is a state formula.
  - If P, Q are state formulas, then $P \wedge Q$ and $\neg P$ are, too.
  - If P is a path formula, then $\mathbf{E}P$ and $\mathbf{A}P$ are state formulas.
  - Every state formula is a path formula.
  - If P, Q are path formulas, then $P \wedge Q$ and $\neg P$ are, too.
  - If P, Q path formulas, then $\mathbf{X}P$ and $P\mathbf{U}Q$ are, too.

Example. Note that, for instance, $\mathbf{E}\neg(P\mathbf{U}Q)$ is a CTL*-formula but $\neg(P\mathbf{U}Q)$ is not (it is a path formula but not a state formula).

CTL* Models

A CTL* model is like a CTL model.

Definition.

- $\mathcal{M}, s \models P$ iff $v(s,P) = \text{true}$, where P is an atomic proposition.
- $\mathcal{M}, s \models \neg P$ iff $\mathcal{M}, s \not\models P$.
- $\mathcal{M}, s \models P \wedge Q$ iff $\mathcal{M}, s \models P$ and $\mathcal{M}, s \models Q$.
- $\mathcal{M}, s \models \mathbf{E}P$ iff there is a full path $x = (s_0, s_1, \ldots)$ where $s_0 = s$ and for which $\mathcal{M}, x \models P$ holds.
- $\mathcal{M}, s \models \mathbf{A}P$ iff for every full path $x = (s_0, s_1, \ldots)$ where $s_0 = s$ holds that $\mathcal{M}, x \models P$.

Relation $\mathcal{M}, x \models P$ is defined next.

Truth in a Model—cont'd

Definition.
Let $x = (s_0, s_1, \ldots)$ be a full path in a CTL* model $\mathcal{M}$.

- $\mathcal{M}, x \models P$ iff $\mathcal{M}, s_0 \models P$, where P is a state formula.
- $\mathcal{M}, x \models \neg P$ iff $\mathcal{M}, x \not\models P$.
- $\mathcal{M}, x \models P \wedge Q$ iff $\mathcal{M}, x \models P$ and $\mathcal{M}, x \models Q$.
- $\mathcal{M}, x \models \mathbf{X}P$ iff $\mathcal{M}, x^1 \models P$
- $\mathcal{M}, x \models P\mathbf{U}Q$ iff there is some $i \geq 0$ such that $\mathcal{M}, x^i \models Q$ and $\mathcal{M}, x^j \models P$ for all $0 \leq j < i$.

6. Validity and Satisfiability

- CTL/CTL*

  A (state) formula P is valid in a model $\mathcal{M}$ ($\mathcal{M} \models P$) iff $\mathcal{M}, s \models P$ for all states s in the model $\mathcal{M}$.

  A formula P is satisfiable iff there is a model $\mathcal{M}$ and a state s such that $\mathcal{M}, s \models P$.
- LTL

  A (path) formula P is valid in a model $\mathcal{M}$ ($\mathcal{M} \models P$) iff $\mathcal{M}, x \models P$ for all full paths x in the model $\mathcal{M}$.

  A formula P is satisfiable if there is a model $\mathcal{M}$ with a full path x such that $\mathcal{M}, x \models P$.
- A formula is valid iff it is valid in every model iff its negation is not satisfiable.

Summary

- Temporal logics are among the most widely applied logics in computer science
- Temporal logic is employed especially in the design methods for distributed and concurrently systems.
- It can be applied to prove correctness of designs, to synthesize automatically designs satisfying given requirements, and to model check designs.
- Model checking is already now applied industrially.
- CTL and LTL are among the most widely applied temporal logics. The rest of course is focusing on CTL and LTL.