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).

Temporal Operators

- FQ (future)
  \[ M, s \models FQ \text{ iff } M, t \models Q \text{ for some } t \in S \text{ such that } sRt. \]
- GQ = \neg F\neg Q (globally)
- PQ (past)
  \[ M, s \models PQ \text{ iff } M, t \models Q \text{ for some } t \in S \text{ such that } tRs. \]
- HQ = \neg P\neg Q (historically)
- Always \( Q = GQ \land Q \land HQ \) (always)
- X (next):
  In every/some next world?
  Note that the accessibility relations \( R_X \) vs. \( R_F \) are related.

Binary Temporal Operators

- U (until):
  \[ M, s \models AUB \text{ iff } \]
  for some \( t \) such that \( sRt, M, t \models B \) and for all \( u \in S, \)
  if \( sRu \) and \( uRt, \) then \( M, u \models A. \)
  \[ \Rightarrow FB \leftrightarrow (TUB) \]
- S (since):
  \[ M, s \models ASB \text{ iff } \]
  for some \( t \) such that \( tRs, M, t \models B \) and for every \( u \in S, \)
  if \( uRs \) and \( tRu, \) then \( M, u \models A. \)
  \[ \Rightarrow PB \leftrightarrow (TUB) \]
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
  $P?\pi$ 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 $a$] $Q$

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: $G\neg(at_i(m) \land at_j(m'))$.
- 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 $m_f$:
  $at(m_0) \land P \rightarrow G(at(m_f) \rightarrow Q)$.
- Total correctness: requires in addition that the program always halts: $at(m_0) \land P \rightarrow Fat(m_f)$.
- No unnecessary replies: a reply $r_i$ is given only to a received request $p_i$: $Fv_i \rightarrow (\neg v_i)\cup p_i$.  

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, ...
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 (sketch of a) program satisfying the requirements (also executable temporal specifications are possible).
  - Easier to automate but still computationally challenging.

© 2008 TKK, Department of Information and Computer Science

3. CTL (Computation Tree Logic)

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

Example. CTL-formulas

\[(P \land Q) \land \neg Q\]
\[AX(P \land \neg Q)\]
\[E((AXP)UQ)\]

CTL Syntax—cont’d

- Note that nesting of temporal operators X/U and their Boolean combinations are limited in CTL.
  For example, AXAXP is a CTL-formula but AXXP and A¬XP are not.
- Other operator pairs (EX,AG,EG,AF,EF) can be defined as shorthands using the basic operators (AX, A(.U.), E(.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. AXP (For all paths in the next world P)
E(PUQ) (There exists a path where P until Q).

© 2008 TKK, Department of Information and Computer Science
**Possible World Semantics**

- CTL models are possible world models \((S, R, v)\) 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 \(X\).

- 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.

![Diagram](https://via.placeholder.com/150)

Examples of full paths:

\[
\begin{align*}
\text{s1, s5, s4, s5, s4,} & \ldots \\
\text{s2, s3, s4, s5, s4,} & \ldots
\end{align*}
\]

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

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

**More Temporal Operators**

- We can define further operators as shorthands:
  \[
  \begin{align*}
  \text{EXP} & \equiv \text{def} \neg \text{AX} \neg P \\
  \text{AGP} & \equiv \text{def} \neg \text{EF} \neg P \\
  \text{AFP} & \equiv \text{def} \text{A} (\top \text{UP}) \\
  \text{EGP} & \equiv \text{def} \neg \text{AF} \top P \\
  \text{EFP} & \equiv \text{def} \text{E} (\top \text{UP})
  \end{align*}
  \]

- Notice the reflexivity and transitivity of the operator \(U\):

  **Example.** If \(M, s_0 \models P\), then \(M, s_0 \models \text{A}(P\text{UQ})\) and \(M, s_0 \models \text{E}(P\text{UQ})\) (and, for example, \(M, s_0 \models \text{AFP}\)).

  If \(s_0 R s_1, s_1 R s_2\) and \(M, s_2 \models P\), then \(M, s_0 \models \text{E}(\top \text{UP})\) (\(\equiv \text{EFP}\)).
4. LTL (Linear Temporal Logic)

- LTL is a linear time temporal logic with operators X, U, G, F.
- Syntax:
  - Every atomic proposition is a LTL-formula.
  - If \( P, Q \) are LTL-formulas, then \( P \land Q, \neg P, XP, P \lor Q \) are LTL-formulas.
- Examples: \( \neg X(P \land \neg Q) \) and \( X(XPU(Q \land P)) \land P \).
- For example, operators G and F can be defined as shorthands using the basic operators: \( FP \equiv \top U P \) and \( GP \equiv \neg F \neg P \).
- Note that nesting of operators X and U and their Boolean combinations are possible: for instance, \( (X(\neg XP))U(XXP) \) 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 \( x = (s_0, x_1, \ldots) \) is a full path, then \( x' = (s_i, x_{i+1}, \ldots) \) is the suffix of \( x \) starting at \( s_i \).

Definition. Let \( M \) be a LTL model and \( x = (s_0, x_1, \ldots) \) one of its full paths.

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

Example

Consider again the model \( M \):

\[
\begin{array}{c}
\text{s3} \\
\text{s4} \\
\text{s5} \\
\end{array}
\begin{array}{c}
\text{s2} \\
\text{s1} \\
\text{s3} \\
\end{array}
\]

the valuation:
\( v(P, s_1) = \text{true} \), otherwise \( v(P, s) = \text{false} \);
\( v(Q, s_2) = \text{true} \), otherwise \( v(Q, s) = \text{false} \).

For full paths \( x_1 = (s_2, s_3, s_4, s_5, s_4, \ldots) \) and \( x_2 = (s_2, s_4, s_5, s_4, \ldots) \):
1. \( M, x_1 \not\models XP \) but \( M, x_2 \models XP \).
2. \( M, x_1 \not\models QUP \) but \( M, x_2 \models QUP \).

More Temporal Operators

- We adopt the following shorthands:
  \( FP \equiv \top UP \)
  \( GP \equiv \neg F \neg P \)
  \( FP \equiv \top FGP \)
  \( GP \equiv \top FGP \)
  \( PBQ \equiv \neg ((\neg P)UQ) \) (before)
- Note the reflexivity and transitivity of the operator U:

Example. If \( M, x \models P \), then \( M, x \models (QUP) \).

If \( M, x \models XP \) for some \( i \geq 0 \), then \( M, x \models (TUP) \).

In fact, for all \( M, x \) holds, for example: \( M, x \models GP \rightarrow P \) and \( M, x \models GP \rightarrow GGP \).
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 \( P \land Q \) and \( \neg P \) are, too.
  - If \( P \) is a path formula, then \( EP \) and \( AP \) are state formulas.
  - Every state formula is a path formula.
  - If \( P, Q \) are path formulas, then \( P \land Q \) and \( \neg P \) are, too.
  - If \( P, Q \) path formulas, then \( XP \) and \( PUQ \) are, too.

**Example.** Note that, for instance, \( E \neg(PUQ) \) is a CTL*-formula but \( \neg(PUQ) \) is not (it is a path formula but not a state formula).

© 2008 TKK, Department of Information and Computer Science

---

6. **Validity and Satisfiability**

- **CTL/CTL**
  A (state) formula \( P \) is valid in a model \( M \) (\( M \models P \)) iff \( M, s \models P \) for all states \( s \) in the model \( M \).
  A formula \( P \) is **satisfiable** iff there is a model \( M \) and a state \( s \) such that \( M, s \models P \).
  
- **LTL**
  A (path) formula \( P \) is valid in a model \( M \) (\( M \models P \)) iff \( M, x \models P \) for all full paths \( x \) in the model \( M \).
  A formula \( P \) is **satisfiable** if there is a model \( M \) with a full path \( x \) such that \( M, x \models P \).
  
- A formula is **valid** if 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.