SPECIFYING PROPERTIES USING TEMPORAL LOGIC

1. CTL vs. LTL
2. Examples of temporal properties
3. Requirement specifications
4. Fairness properties and CTL

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

1. CTL vs. LTL

- LTL is a linear time logic and the truth of an LTL formula is evaluated on a full path given by the model.
- CTL is a branching time logic and the truth of a CTL formula is evaluated (in effect) on a computation tree given by the model.
- This means that when determining the truth value of a CTL formula in state s_0 in a model $\mathcal{M}$ the evaluation unfolds the model $\mathcal{M}$ as a computation tree starting from s_0 and temporal operators are evaluated in this tree.

CTL Computation Tree

- For a model $\mathcal{M} = \langle S, R, v\rangle$ and a state $s_0 \in S$, the computation tree $\hat{\mathcal{M}} = \langle \hat{S}, \hat{R}, \hat{v}\rangle$, starting from s_0 is constructed as follows:
  - (i) start with the node $\langle s_0, 0\rangle$.
  - (ii) unfold the model using the rule:
    if $\langle s,n\rangle \in \hat{S}$ and sRt, then $\langle t,m\rangle \in \hat{S}$ and $\langle\langle s,n\rangle,\langle t,m\rangle\rangle \in \hat{R}$
    where m is a new number not used before.
  - (iii) The valuation $\hat{v}$ is given by $\hat{v}(\langle s,n\rangle, P) = v(s,P)$ for all $\langle s,n\rangle$.
- Now temporal formulas are evaluated on $\hat{\mathcal{M}}$ as follows:
  - $\hat{\mathcal{M}}, \hat{s_0} \models \mathbf{A}(P\mathbf{U}Q)$ iff for all branches of the computation tree $\hat{\mathcal{M}}$ $(\hat{s_0}, \hat{s_1}, \ldots)$ there is some $i \geq 0$ such that $\hat{\mathcal{M}}, \hat{s_i} \models Q$ and $\hat{\mathcal{M}}, \hat{s_j} \models P$ for all $0 \leq j < i$.
  - $\hat{\mathcal{M}}, \hat{s_0} \models \mathbf{E}(P\mathbf{U}Q)$ iff there is some branch of $\hat{\mathcal{M}}$ $(\hat{s_0}, \hat{s_1}, \ldots)$ and some $i \geq 0$ such that $\hat{\mathcal{M}}, \hat{s_i} \models Q$ and $\hat{\mathcal{M}}, \hat{s_j} \models P$ for all $0 \leq j < i$.

Comparing CTL and LTL formulas

- For model checking consider the correspondence

CTL:	LTL:
$\mathcal{M}, s_0 \models P$	$\mathcal{M}, x \models P$ for all full paths $x = (s_0, \ldots)$.

- CTL and LTL operators are similar but differ in some respects.
- For instance, "temporal possibilities" can be expressed in CTL but not in LTL.

Example. For a CTL formula $\mathbf{AGEF}P$, there is no corresponding LTL formula.

Consider LTL formula $\mathbf{GF}P$ and the model $\mathcal{M} = \langle S, R, v\rangle$ where $S = \{s_0, s_1\}$, $R = \{\langle s_0, s_0\rangle, \langle s_0, s_1\rangle, \langle s_1, s_0\rangle\}$, $v(s_0, P) = \text{false}$ and $v(s_1, P) = \text{true}$.

The formula $\mathbf{GF}P$ is not valid in $\mathcal{M}$ because it is false in a full path $(s_0, s_0, s_0, \ldots)$ although CTL formula $\mathbf{AGEF}P$ is valid in $\mathcal{M}$.

Differences between CTL and LTL—cont'd

- Thus, a CTL formula of the type "there is a path ..." is not expressible as an LTL formula.

  Example. For a CTL formula $\mathbf{EF}P$ there is no corresponding LTL formula.

  For instance, the LTL formula $\mathbf{F}P$ is not valid in the previous model $\mathcal{M}$ because it is false in a full path $(s_0, s_0, s_0, \ldots)$ but the CTL formula $\mathbf{EF}P$ is valid.

- Fairness properties are not expressible in CTL.

  Example. For an LTL formula $\mathbf{FG}Q$ there is no corresponding CTL formula.

  Consider the previous model $\mathcal{M}$ where we set $v(s_0, Q) = \text{true}$ and $v(s_1, Q) = \text{false}$.

  Now $\mathbf{FG}Q$ is true in a full path $(s_0, s_0, \ldots)$ but the CTL formula $\mathbf{AFAG}Q$ is not satisfiable in $\mathcal{M}$ and neither is $\mathbf{EFAG}Q$.

2. Examples of Temporal Properties

- $\mathbf{EF}(\mathit{started} \wedge \neg \mathit{ready})$:

  It is possible to reach a state where *started* is true but *ready* is not.

- $\mathbf{AG}(\mathit{req} \rightarrow \mathbf{AF}\mathit{ack})$:

  If a request is received then it will be acknowledged.

- $\mathbf{AGAF}\mathit{enabled}$:

  enabled is true infinitely often on every computation path.

- $\mathbf{AGEF}\mathit{restart}$:

  From every state is it possible to reach a state where *restart* is true.

3. Requirement Specifications

- Temporal logic can be used to state requirement specifications for reactive systems.
- Typical requirement specifications can be divided into the following classes:
  1. Reachability properties
  2. Safety properties
  3. Liveness properties
  4. Fairness properties

Reachability Properties

- This is a simple class of properties stating that some state (where a given condition P is true) can be reached (from the initial state of the system).
- Can be expressed using temporal formulas of the form $\mathbf{EF}P$.
- Conditional reachability can be expressed using temporal formulas of the form $\mathbf{E}(Q\mathbf{U}P)$ (there is an execution where Q is true reaching a state where P is true).

Example. Typical reachability properties:

1. $\mathbf{EF}(\mathit{started} \wedge \neg \mathit{ready})$.
2. $\mathbf{EF}(\mathit{restart})$.
3. $\mathbf{E}(\neg \mathit{restart}\mathbf{U}\mathit{ready})$.

Safety Properties

- Safety properties state that nothing "bad" happens during an execution of the system.
- A safety property is a requirement which has a finite counter-execution:

  if the system does not satisfy a safety property P, then it has a finite execution where the property P does not hold.

Example. Examples of typical safety properties:

1. Mutual exclusion: $\mathbf{AG}\neg(atCS_1 \wedge atCS_2)$.
2. Partial correctness: $atl_0 \wedge P \rightarrow \mathbf{AG}(atl_h \rightarrow Q)$.

Liveness Properties

- Liveness properties express that something "good" happens.
- Liveness properties do not have finite counter-executions:

  if a system does not satisfy a liveness property P, then this can be demonstrated only using an infinite counter-execution.

Example. Typical examples of liveness properties:

1. (nested) reachability : $\mathbf{AGEF}restart$.
2. temporal implication: $\mathbf{AG}(P \rightarrow \mathbf{AF}Q)$.
3. Starvation freeness: $\mathbf{AG}(atTry_i \rightarrow \mathbf{AF}atCS_i)$.
4. Total correctness: $atl_0 \wedge P \rightarrow \mathbf{AF}(atl_h \wedge Q)$.

Fairness Properties

- Fairness properties are liveness properties which require that states where a given condition is true occur infinitely often.
- Fairness properties are not directly expressible in CTL but they are in LTL.

Example. Consider two atomic propositions for a process:
en (the process is enabled) and
ex (the process is executed).

1. Unconditional fairness: $\mathbf{GF}ex$.
2. Strong fairness: $\mathbf{GF}en \rightarrow \mathbf{GF}ex$.
3. Weak fairness: $\mathbf{FG}en \rightarrow \mathbf{GF}ex$.

4. Fairness Properties and CTL

- When using CTL fairness properties are handled by modifying the semantics of the path quantifiers ($\mathbf{A}/\mathbf{E}$).
- Quantification is considered over all fair paths (and not over all paths as in the basic case).
- Fairness conditions are given as a set of formulas F and when evaluating the truth of a formula only F-fair paths are considered.

Definition. A full path x is F-fair iff every $P \in F$ is true infinitely often on the path x.

Modified Semantics

Relation $\models_F$ is defined as $\models$ except that path quantification is over F-fair paths.

- $\mathcal{M}, s \models_F P$ iff there is a F-fair full path starting from the state s and $v(s,P) = \text{true}$ when P is an atomic proposition.
- $\mathcal{M}, s \models_F \mathbf{A}(P\mathbf{U}Q)$ iff for all F-fair full path $(s_0, s_1, \ldots)$ where $s = s_0$, there is some $i \geq 0$ such that $\mathcal{M}, s_i \models_F Q$ and $\mathcal{M}, s_j \models_F P$ for all $0 \leq j < i$.
- $\mathcal{M}, s \models_F \mathbf{E}(P\mathbf{U}Q)$ iff there is some F-fair full path $(s_0, s_1, \ldots)$ with $s = s_0$ and there is some $i \geq 0$ such that $\mathcal{M}, s_i \models_F Q$ and $\mathcal{M}, s_j \models_F P$ for all $0 \leq j < i$.

Example. *Unconditional fairness* can be express using the set $F = \{ex\}$ and a *fair channel* using a set $F = \{send \rightarrow rec\}$.

Summary

- Although CTL and LTL are based on similar temporal operators, they are different because LTL is a linear time logic where formulas are evaluated on paths whereas CTL is a branching time logic where formulas are evaluated on computation trees.
- Hence, there are CTL formulas (for instance of the form "there is a path ...") which cannot be expressed in LTL and LTL formulas (for example fairness formulas) which cannot be expressed in CTL.
- Temporal logics are suitable for requirement specification of reactive systems.
- Typical requirement specifications include reachability properties, safety properties, liveness properties, and fairness properties.