DECISION METHODS FOR CTL AND LTL

1. Tableau method for CTL
2. Deciding satisfiability and validity in CTL and LTL

E. A. Emerson: *Automated Temporal Reasoning about Reactive Systems*, Section 4 (pp. 18–23).

Background

- Temporal logics CTL and LTL are decidable because they satisfy the finite model property (and there is an upper bound on the size of a counter-model).
- More efficient decision methods can be developed using tableau techniques and automata theory.

Key Decision Techniques

- Tableau method (for example CTL):
  (i) Given a CTL formula, a tableau graph is built representing (essentially) all possible models of the formula.
  (ii) The tableau is pruned and the checked whether it represents any model of the formula.
- Automata theory methods (for example LTL):
  (i) Given an LTL formula, a finite state (Büchi-) automaton is constructed accepting infinite words (paths) such that the automaton accepts (essentially) all possible full paths satisfying the formula.
  (ii) Then it is checked whether the language accepted by the automaton is empty.

1. Tableau Method for CTL

- A CTL tableau is a bipartite graph where nodes are sets of formulas and of two types: OR-nodes and AND-nodes.
- For a CTL formula P a tableau is constructed by first transforming P to the *positive normal form* (where negations can appear only in front of atomic propositions) and then proceeding in two stages:
  (i) building an initial tableau T_0 and
  (ii) reducing T_0 to the final tableau T_1 using pruning rules.
- In the positive normal form propositional connectives $\wedge$, $\vee$ are used and negation $\neg$ appears only in front of atomic propositions.
- We denoted by $\sim P$ the formula $\neg P$ in the positive normal form.

Transformation Rules

A CTL formula P can be transformed to the positive normal form using the following rules:

$P \rightarrow Q$	$\mapsto$	$\neg P \vee Q$
$\neg(P \vee Q)$	$\mapsto$	$\neg P \wedge \neg Q$
$\neg(P \wedge Q)$	$\mapsto$	$\neg P \vee \neg Q$
$\neg\neg P$	$\mapsto$	P
$\neg\mathbf{AG}P$	$\mapsto$	$\mathbf{EF}\neg P$
$\neg\mathbf{EF}P$	$\mapsto$	$\mathbf{AG}\neg P$
$\neg\mathbf{EG}P$	$\mapsto$	$\mathbf{AF}\neg P$
$\neg\mathbf{AF}P$	$\mapsto$	$\mathbf{EG}\neg P$
$\neg\mathbf{AX}P$	$\mapsto$	$\mathbf{EX}\neg P$
$\neg\mathbf{EX}P$	$\mapsto$	$\mathbf{AX}\neg P$

$\neg\mathbf{A}(P\mathbf{U}Q)$	$\mapsto$	$\mathbf{E}(\neg P\mathbf{B}Q)$
$\neg\mathbf{E}(P\mathbf{U}Q)$	$\mapsto$	$\mathbf{A}(\neg P\mathbf{B}Q)$
$\neg\mathbf{A}(P\mathbf{B}Q)$	$\mapsto$	$\mathbf{E}(\neg P\mathbf{U}Q)$
$\neg\mathbf{E}(P\mathbf{B}Q)$	$\mapsto$	$\mathbf{A}(\neg P\mathbf{U}Q)$

Note the shorthands:

$\mathbf{A}(P\mathbf{B}Q)$: $\neg\mathbf{E}((\neg P)\mathbf{U}Q)$

$\mathbf{E}(P\mathbf{B}Q)$: $\neg\mathbf{A}((\neg P)\mathbf{U}Q)$

Example

Constructing the positive normal form $\sim\mathbf{AG}(R \rightarrow (\neg Q \wedge \mathbf{A}(P\mathbf{U}Q)))$:

$$
\begin{aligned}
&\neg\mathbf{AG}(R \rightarrow (\neg Q \wedge \mathbf{A}(P\mathbf{U}Q)))\\
\mapsto\ &\mathbf{EF}\neg(\neg R \vee (\neg Q \wedge \mathbf{A}(P\mathbf{U}Q)))\\
\mapsto\ &\mathbf{EF}(R \wedge \neg(\neg Q \wedge \mathbf{A}(P\mathbf{U}Q)))\\
\mapsto\ &\mathbf{EF}(R \wedge (Q \vee \neg\mathbf{A}(P\mathbf{U}Q)))\\
\mapsto\ &\mathbf{EF}(R \wedge (Q \vee \mathbf{E}(\neg P\mathbf{B}Q)))
\end{aligned}
$$

Building the Initial Tableau T_0

- Given a CTL formula P we start with the OR-node $D_0 = \{P\}$.
- The successors of an OR-node D are AND-nodes obtained by applying α/β-*rules* to the node D.
- The successors of an AND-node C are OR-nodes obtained by applying the *successor rule* to C.

Remark. If a successor C of an OR-node D already appears in the tableau, another copy of C is not introduced in the tableau but the successor of D is set to be the already existing node (and similarly for the successors of AND-nodes).

$\Longrightarrow$ The tableau remains finite.

Successors of an OR-Node

Each successor of an OR-node D is a smallest set of formulas C such that $D \subseteq C$ and C is downward closed under the α and β rules below such that

1. if $\alpha \in C$, then $\alpha_1 \in C$ and $\alpha_2 \in C$;
2. if $\beta \in C$, then $\beta_1 \in C$ or $\beta_2 \in C$.

α rules:

$$\frac{P \wedge Q}{\begin{array}{c}P\\Q\end{array}} \qquad \frac{\mathbf{AG}P}{\begin{array}{c}P\\\mathbf{AXAG}P\end{array}} \qquad \frac{\mathbf{EG}P}{\begin{array}{c}P\\\mathbf{EXEG}P\end{array}}$$

$$\frac{\mathbf{A}(P\mathbf{B}Q)}{\begin{array}{c}\sim Q\\P \vee \mathbf{AXA}(P\mathbf{B}Q)\end{array}} \qquad \frac{\mathbf{E}(P\mathbf{B}Q)}{\begin{array}{c}\sim Q\\P \vee \mathbf{EXE}(P\mathbf{B}Q)\end{array}}$$

Successors of an OR-Node–cont'd

β rules:

$$\frac{P \vee Q}{P \mid Q} \qquad \frac{\mathbf{AF}P}{P \mid \mathbf{AXAF}P} \qquad \frac{\mathbf{EF}P}{P \mid \mathbf{EXEF}P}$$

$$\frac{\mathbf{A}(P\mathbf{U}Q)}{Q \mid P \wedge \mathbf{AXA}(P\mathbf{U}Q)} \qquad \frac{\mathbf{E}(P\mathbf{U}Q)}{Q \mid P \wedge \mathbf{EXE}(P\mathbf{U}Q)}$$

Remark. For literals and for formulas of the form $\mathbf{AX}P$ and $\mathbf{EX}P$ there are no applicable rules (and the expansion of a successor C ends in such formulas).

Example

From the set $D = \{\mathbf{AFEF}((P \wedge Q) \vee R)\}$ we can construct the following downward closed sets of formulas:

$C_1 = \{\mathbf{AFEF}((P \wedge Q) \vee R), \mathbf{EF}((P \wedge Q) \vee R), (P \wedge Q) \vee R, P \wedge Q, P, Q\}$

$C_2 = \{\mathbf{AFEF}((P \wedge Q) \vee R), \mathbf{EF}((P \wedge Q) \vee R), (P \wedge Q) \vee R, R\}$

$C_3 = \{\mathbf{AFEF}((P \wedge Q) \vee R), \mathbf{EF}((P \wedge Q) \vee R), \mathbf{EXEF}((P \wedge Q) \vee R)\}$

$C_4 = \{\mathbf{AFEF}((P \wedge Q) \vee R), \mathbf{AXAFEF}((P \wedge Q) \vee R)\}$

Remark. Downward closed sets of formulas for a node D can be built by constructing a tableau where the formulas in D are put to the root of the tableau and then α and β rules are applied as in the tableau method for propositional logic. Each branch of the resulting tableau corresponds to a downward closed set.

Successors of AND-Nodes

The successors of an AND-node C are obtained with the *successor rule*:

- Let the set of formulas C contain the following $\mathbf{AX}P_i/\mathbf{EX}Q_j$ formulas:

  $$\mathbf{AX}P_1, \ldots, \mathbf{AX}P_l \text{ and } \mathbf{EX}Q_1, \ldots, \mathbf{EX}Q_k.$$

  Then the successors of C are

  $$D_1 = \{P_1, \ldots, P_l, Q_1\}, \ldots, D_k = \{P_1, \ldots, P_l, Q_k\}.$$

- If the set C has no formulas of the form $\mathbf{EX}Q_i$, then C has a unique successor $\{P_1, \ldots, P_l\}$.

  Note that there is always at least one successor (which is the empty set if there are no formulas of the form $\mathbf{AX}P_i$ either).

Example. The node

$$C = \{\mathbf{A}(P\mathbf{U}Q), \mathbf{AXA}(P\mathbf{U}Q), \mathbf{EG}P, P, \mathbf{EXEG}P, \mathbf{EF}Q, \mathbf{EXEF}Q\}.$$

has successors $D_1 = \{\mathbf{A}(P\mathbf{U}Q), \mathbf{EG}P\}$ and $D_2 = \{\mathbf{A}(P\mathbf{U}Q), \mathbf{EF}Q\}$.

Pruning Rules

The final tableau is obtained by pruning the initial tableau T_0 using the following rules until none of them is applicable.

- Remove an AND-node containing a formula and its negation.
- Remove an AND-node if one of its original successors have been removed.
- Remove an OR-node if all its original successors have been removed.
- Remove an AND-node if it contains a **eventuality formula** not satisfiable in the current tableau.

Eventuality formulas are of the form:

$$\mathbf{E}(P\mathbf{U}Q),\ \mathbf{EF}Q,\ \mathbf{A}(P\mathbf{U}Q) \text{ and } \mathbf{AF}Q.$$

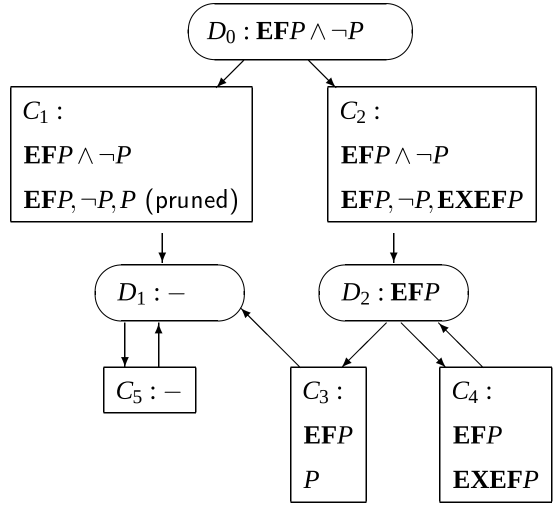
Satisfiability of Eventuality Formulas

- An eventuality formula $\mathbf{EF}Q$ ($\mathbf{E}(P\mathbf{U}Q)$) is satisfiable in an AND-node C iff the tableau includes a path from C to an AND-node C' containing the formula Q (and all other AND-nodes in the path contain the formula P).
- An eventuality formula $\mathbf{AF}Q$ ($\mathbf{A}(P\mathbf{U}Q)$) is satisfiable in an AND-node C iff there is a finite acyclic subgraph in the tableau such that
  (i) The root of the subgraph is the node C.
  (ii) For each interior OR-node in the subgraph exactly one of its successor AND-nodes in the current tableau is in the subgraph.
  (iii) For each interior AND-node all of its successor OR-nodes in the current tableau are in the subgraph.
  (iv) Every leaf node of the subgraph is an AND-node containing the formula Q (and all other AND-nodes contain P).

Example

- The node $D_0 = \{\mathbf{EF}P \wedge \neg P\}$ has AND-successors $C_1 = \{\mathbf{EF}P \wedge \neg P, \mathbf{EF}P, \neg P, P\}$ and $C_2 = \{\mathbf{EF}P \wedge \neg P, \mathbf{EF}P, \neg P, \mathbf{EXEF}P\}$.
- The OR-successors of C_1: $D_1 = \{\}$.
  The OR-successors of C_2: $D_2 = \{\mathbf{EF}P\}$.
- The AND-successors of D_2: $C_3 = \{\mathbf{EF}P, P\}$ and $C_4 = \{\mathbf{EF}P, \mathbf{EXEF}P\}$.
  The AND-successors of D_1: $C_5 = \{\}$.
- The OR-successors of C_3 and C_5: $\{\} = D_1$.
  The OR-successors of C_4: $\{\mathbf{EF}P\} = D_2$.
- Initial tableau T_0 is now finished.
- Pruning: Remove C_1 (contains a formula and its negations). Final tableau T_1 now ready (an eventuality formula $\mathbf{EF}P$ satisfiable in AND-nodes C_2, C_3, C_4).

The Tableau as a Graph



2. Deciding Satisfiability and Validity in CTL and LTL

- The final tableau T_1 for a CTL formula P provides a model for P in case P is satisfiable.
- The validity of a CTL formula P can be determined by checking whether the formula $\sim P$ is unsatisfiable.
- The satisfiability/validity of an LTL formula P can be reduced to the satisfiability/validity of a CTL formula.

Deciding CTL Satisfiability using Tableaux

Theorem. Let P be a CTL formula in the positive normal form. Then P is satisfiable iff the final tableau T_1 for P has an AND-node containing P.

A tableau provides a satisfying model for a CTL formula P:

- The states of the model are given by the AND-nodes and the valuation is given by the atomic propositions in the AND-nodes.
- The model must contain at least one AND-node containing P.
- The successors need to be chosen such that the model is serial and for all AND-nodes the eventuality formulas in the AND-nodes are satisfiable.

Remark. The tableau method can be used for program synthesis (to construct program control skeletons):
(i) The specification of the program is provided as a CTL formula.
(ii) The method is used to construct a model satisfying the specification (providing the control skeleton).

Example

For a CTL formula $\mathbf{EF}P \wedge \neg P$ we can construct a model $\langle S, R, v\rangle$ from the tableaux T_1 as follows:

- Let $S = \{C_2, C_3, C_5\}$, $R = \{\langle C_2, C_3\rangle, \langle C_3, C_5\rangle, \langle C_5, C_5\rangle\}$ and $v(P, s) = \text{true}$ if $s = C_3$ and otherwise $v(P, s) = \text{false}$.
- Another possibility:
  $S = \{C_2, C_3, C_4, C_5\}$, $R = \{\langle C_2, C_4\rangle, \langle C_4, C_3\rangle, \langle C_3, C_5\rangle, \langle C_5, C_5\rangle\}$ and $v(P, s) = \text{true}$ if $s = C_3$ and otherwise $v(P, s) = \text{false}$.

Remark. Consider a model

$S = \{C_2, C_4\}$, $R = \{\langle C_2, C_4\rangle, \langle C_4, C_4\rangle\}$ and $v(P, C_2) = v(P, C_4) = \text{false}$.

This is not a satisfying model because the eventuality formula $\mathbf{EF}P$ in C_2 and C_4 is not satisfiable.

Deciding CTL Validity using Tableaux

- A formula φ is valid iff its negation $\neg\varphi$ is not satisfiable.
- Satisfiability can be determined using the tableau method:
  (i) transform the formula $\neg\varphi$ to the positive normal form $\sim\varphi$;
  (ii) construct a tableau for the formula $\sim\varphi$.
- Hence, a CTL formula φ is valid iff $\sim\varphi$ is not satisfiable iff in the final tableau for the formula $\sim\varphi$ there is no AND-node containing the formula $\sim\varphi$.

Example

Is a CTL formula

$$\varphi = \mathbf{EX}(P \vee Q) \rightarrow (\mathbf{EX}P \vee \mathbf{EX}Q)$$

valid?

Transform $\neg\varphi$ to the positive normal form $\sim\varphi$:

$\neg(\mathbf{EX}(P \vee Q) \rightarrow (\mathbf{EX}P \vee \mathbf{EX}Q)$
$\mapsto \quad \mathbf{EX}(P \vee Q) \wedge \neg(\mathbf{EX}P \vee \mathbf{EX}Q)$
$\mapsto \quad \mathbf{EX}(P \vee Q) \wedge (\neg\mathbf{EX}P \wedge \neg\mathbf{EX}Q)$
$\mapsto \quad \mathbf{EX}(P \vee Q) \wedge \mathbf{AX}\neg P \wedge \mathbf{AX}\neg Q$

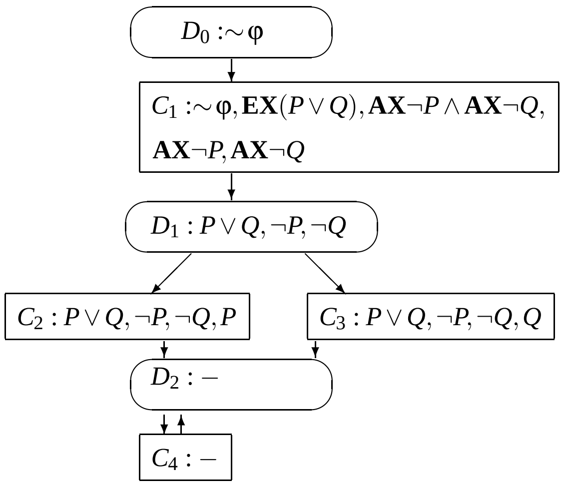
Hence, $\sim\varphi = \mathbf{EX}(P \vee Q) \wedge \mathbf{AX}\neg P \wedge \mathbf{AX}\neg Q$.

Example: AND- and OR-Successors

- The AND-successors of the node $D_0 = \{\sim\varphi\}$:
  $C_1 = \{\sim\varphi, \mathbf{EX}(P \vee Q), \mathbf{AX}\neg P \wedge \mathbf{AX}\neg Q, \mathbf{AX}\neg P, \mathbf{AX}\neg Q\}$.
- The OR-successor of C_1:
  $D_1 = \{P \vee Q, \neg P, \neg Q\}$.
- The AND-successors of D_1:
  $C_2 = D_1 \cup \{P\}$ and $C_3 = D_1 \cup \{Q\}$.
- The OR-successors of C_2: $D_3 = \{\}$.
  The OR-successors of C_3: $\{\} = D_3$.
  The AND-successors of D_3: $C_4 = \{\}$.
  The OR-successors of C_4: $\{\} = D_3$.

$\Longrightarrow$ The initial tableau T_0 is ready.

Example: The Initial Tableau



Example: Pruning of the Initial Tableau

1. Nodes C_2 and C_3 can be removed because they contain a formula and its negation.
2. OR-node D_1 can be removed (all successors removed).
3. AND-node C_1 can be removed (a successor removed).
4. OR-node D_0 can be removed (all successors removed).

$\Longrightarrow$ Final tableau T_1 is ready.

T_1 does not contain an AND-node which includes $\sim\varphi$.
Hence, $\sim\varphi$ is not satisfiable and φ is valid.

Deciding LTL Satisfiability using Tableaux

- CTL tableaux can be used for deciding LTL satisfiability.

Theorem. Let P be an LTL formula in the positive normal form and let the CTL formula P' be obtained from P by replacing operators $\mathbf{F}, \mathbf{G}, \mathbf{X}, \mathbf{U}, \mathbf{B}$ systematically by $\mathbf{AF}, \mathbf{AG}, \mathbf{AX}, \mathbf{AU}, \mathbf{AB}$, respectively.

Then P is satisfiable in LTL iff P' is satisfiable in CTL.

Example. An LTL formula $\mathbf{G}(\neg P\mathbf{U}Q)$ is satisfiable iff $\mathbf{AGA}(\neg P\mathbf{U}Q)$ is satisfiable (in CTL).

Computational Complexity

- CTL
  Model checking: **P**-complete, $O(|M| \cdot |P|)$
  Satisfiability: **EXPTIME**-complete
- LTL
  Model checking: **PSPACE**-complete, $O(|M| \cdot exp(|P|))$
  Satisfiability: **PSPACE**-complete
- CTL*
  Model checking: **PSPACE**-complete, $O(|M| \cdot exp(|P|))$
  Satisfiability: 2**EXPTIME**-complete

Restricted Subclasses

- Satisfiability can be decided in polynomial time, for instance, in subclasses which are interesting for program synthesis.
- For example SCTL (Simplified CTL):
  $P_1 \vee \cdots \vee P_n$, $\mathbf{AG}(P_1 \vee \cdots \vee P_n)$
  $\mathbf{AG}(P_0 \rightarrow \mathbf{AF}(P_1 \vee \cdots \vee P_n))$,
  $\mathbf{AG}(P_0 \rightarrow \mathbf{A}(P_1 \vee \cdots \vee P_n \mathbf{U} Q_1 \vee \cdots \vee Q_m))$
  $\mathbf{AG}(P_0 \rightarrow \mathbf{AX}(P_1 \vee \cdots \vee P_n) \wedge \mathbf{EX}(Q_1 \vee \cdots \vee Q_m) \wedge \cdots \wedge \mathbf{EX}(R_1 \vee \cdots \vee R_l))$
  where P_i, Q_i, R_i atomic propositions such that the ESC-assumption holds (eventualities are "history-free").
- For example, RLTL (Restricted LTL)
  $\mathbf{G}(P_1 \vee \cdots \vee P_n)$
  $\mathbf{G}(P_0 \rightarrow \mathbf{F}(P_1 \vee \cdots \vee P_n))$
  $\mathbf{G}(P_0 \rightarrow \mathbf{X}(P_1 \vee \cdots \vee P_n))$
  where each P_i is an atomic proposition.

Summary

- Methods for deciding satisfiability (and validity) in temporal logics are typically based on tableau techniques and automata theory.
- The tableau method for CTL can be seen as a systematic procedure to build a model for a formula (in positive normal form).
- In the tableau method first an initial tableau is built which is then reduced using pruning rules. If the reduced tableau satisfies a given condition, a model of the original formula can be obtained from the reduced tableau.
- The method can be used for synthesizing (control skeletons of) programs.
- Satisfiability of a LTL formula can be reduced to satisfiability of a CTL formula and, hence, the CTL tableau method can be used for deciding satisfiability (and validity) in LTL.