Computation Tree Logic (CTL) & Basic Model Checking Algorithms

Martin Fränzle

Carl von Ossietzky Universität
Dpt. of Computing Science
Res. Grp. Hybride Systeme
Oldenburg, Germany
What you’ll learn

1. Rationale behind declarative specifications:
   - Why operational style is insufficient

2. Computation Tree Logic CTL:
   - Syntax
   - Semantics: Kripke models

3. Explicit-state model checking of CTL:
   - Recursive coloring
Operational models

Nowadays, a lot of ES design is based on executable behavioral models of the system under design, e.g. using

- Statecharts (a syntactically sugared variant of Moore automata)
- VHDL.

Such operational models are good at

- supporting system analysis
  - simulation / virtual prototyping
- supporting incremental design
  - executable models
- supporting system deployment
  - executable model as “golden device”
  - code generation for rapid prototyping or final product
  - hardware synthesis
Operational models

...are bad at

- supporting non-operational descriptions:
  - *What* instead of *how*.
  - E.g.: Every request is eventually answered.

- supporting negative requirements:
  - “Thou shalt not...”
  - E.g.: The train ought not move, unless it is manned.

- providing a structural match for requirement *lists*:
  - System has to satisfy $R_1$ and $R_2$ and ...
  - If system fails to satisfy $R_1$ then $R_2$ should be satisfied.
Multiple viewpoints

Aspects
"What?"

Algorithmics
"How?"

Tests & proofs
"Consistent?"

Requirements analysis

Programming

Validation / verification
Model checking

Device Description:

architecture behaviour of processor is

process fetch
  if halt=0 then
    if mem_wait=0 then
      nextins <= dport

...
Exhaustive state-space search

Automatic verification/falsification of invariants
Safety requirement: Gate has to be closed whenever a train is in “In”.
The gate model

Opening

Open

~enter?

enter?

Closing

Empty

Appr.

In

leave!

enter!

~leave?

Track model

— safe abstraction —
Gate reaction: Open, Closing, Closed, Opening, Open.
Computation Tree Logic
Syntax of CTL

We start from a countable set $AP$ of atomic propositions. The CTL formulae are then defined inductively:

- Any proposition $p \in AP$ is a CTL formula.
- The symbols $\perp$ and $\top$ are CTL formulae.
- If $\phi$ and $\psi$ are CTL formulae, so are
  - $\neg \phi$, $\phi \land \psi$, $\phi \lor \psi$, $\phi \rightarrow \psi$
  - $EX \phi$, $AX \phi$
  - $EF \phi$, $AF \phi$
  - $EG \phi$, $AG \phi$
  - $\phi EU \psi$, $\phi AU \psi$
Semantics (informal)

- **E** and **A** are path quantifiers:
  - **A**: for all paths in the computation tree...
  - **E**: for some path in the computation tree...

- **X**, **F**, **G** und **U** are temporal operators which refer to the path under investigation, as known from LTL:
  - **Xφ** (Next): evaluate φ in the next state on the path
  - **Fφ** (Finally): φ holds for some state on the path
  - **Gφ** (Globally): φ holds for all states on the path
  - **φUψ** (Until): φ holds on the path at least until ψ holds

**N.B.** Path quantifiers and temporal operators are compound in CTL: there never is an isolated path quantifier or an isolated temporal operator. There is a lot of things you can’t express in CTL because of this...
CTL formulae are interpreted over Kripke structures.

A Kripke structure $K$ is a quadruple $K = (V, E, L, I)$ with

- $V$ a set of vertices (interpreted as system states),
- $E \subseteq V \times V$ a set of edges (interpreted as possible transitions),
- $L \in V \rightarrow \mathcal{P}(AP)$ labels the vertices with atomic propositions that apply in the individual vertices,
- $I \subseteq V$ is a set of initial states.
A path $\pi$ in a Kripke structure $K = (V, E, L, I)$ is an edge-consistent infinite sequence of vertices:

- $\pi \in V^\omega$,
- $(\pi_i, \pi_{i+1}) \in E$ for each $i \in \mathbb{N}$.

Note that a path need not start in an initial state!

The labelling $L$ assigns to each path $\pi$ a propositional trace

$$tr_\pi = L(\pi) \overset{\text{def}}{=} \langle L(\pi_0), L(\pi_1), L(\pi_2), \ldots \rangle$$

that *path formulae* ($X\phi, F\phi, G\phi, \phi U \psi$) can be interpreted on.
Let $K = (V, E, L, I)$ be a Kripke structure and $v \in V$ a vertex of $K$.

- $v, K \models \top$
- $v, K \not\models \bot$
- $v, K \models p$ for $p \in AP$ iff $p \in L(v)$
- $v, K \models \neg \phi$ iff $v, K \not\models \phi$,
- $v, K \models \phi \land \psi$ iff $v, K \models \phi$ and $v, K \models \psi$,
- $v, K \models \phi \lor \psi$ iff $v, K \models \phi$ or $v, K \models \psi$,
- $v, K \models \phi \Rightarrow \psi$ iff $v, K \not\models \phi$ or $v, K \models \psi$. 
Semantics (contd.)

• \( v, K \models \mathsf{EX} \phi \) iff there is a path \( \pi \) in \( K \) s.t. \( v = \pi_1 \) and \( \pi_2, K \models \phi \),

• \( v, K \models \mathsf{AX} \phi \) iff all paths \( \pi \) in \( K \) with \( v = \pi_1 \) satisfy \( \pi_2, K \models \phi \),

• \( v, K \models \mathsf{EF} \phi \) iff there is a path \( \pi \) in \( K \) s.t. \( v = \pi_1 \) and \( \pi_i, K \models \phi \) for some \( i \),

• \( v, K \models \mathsf{AF} \phi \) iff all paths \( \pi \) in \( K \) with \( v = \pi_1 \) satisfy \( \pi_i, K \models \phi \) for some \( i \) (that may depend on the path),

• \( v, K \models \mathsf{EG} \phi \) iff there is a path \( \pi \) in \( K \) s.t. \( v = \pi_1 \) and \( \pi_i, K \models \phi \) for all \( i \),

• \( v, K \models \mathsf{AG} \phi \) iff all paths \( \pi \) in \( K \) with \( v = \pi_1 \) satisfy \( \pi_i, K \models \phi \) for all \( i \),

• \( v, K \models \phi \mathsf{EU} \psi \), iff there is a path \( \pi \) in \( K \) s.t. \( v = \pi_1 \) and some \( k \in \mathbb{N} \) s.t. \( \pi_i, K \models \phi \) for each \( i < k \) and \( \pi_k, K \models \psi \),

• \( v, K \models \phi \mathsf{AU} \psi \), iff all paths \( \pi \) in \( K \) with \( v = \pi_1 \) have some \( k \in \mathbb{N} \) s.t. \( \pi_i, K \models \phi \) for each \( i < k \) and \( \pi_k, K \models \psi \).

A Kripke structure \( K = (V, E, L, I) \) satisfies \( \phi \) iff all its initial states satisfy \( \phi \), i.e. iff \( is, K \models \phi \) for all \( is \in I \).
CTL Model Checking

Explicit-state algorithm
Rationale

We will extend the idea of verification/falsification by exhaustive state-space exploration to full CTL.

- Main technique will again be breadth-first search, i.e. graph coloring.
- Need to extend this to other modalities than $\text{AG}$.
- Need to deal with nested modalities.
Model-checking CTL: General layout

Given: a Kripke structure $K = (V, E, L, I)$ and a CTL formula $\phi$

Core algorithm: find the set $V_\phi \subseteq V$ of vertices in $K$ satisfying $\phi$ by
1. for each atomic subformula $p$ of $\phi$, mark the set $V_p \subseteq V$ of vertices in $K$ satisfying $\phi$
2. for increasingly larger subformulae $\psi$ of $\phi$, synthesize the marking $V_\psi \subseteq V$ of nodes satisfying $\psi$ from the markings for $\psi$'s immediate subformulae
until all subformulae of $\phi$ have been processed (including $\phi$ itself)

Result: report “$K \models \phi$” iff $V_\phi \supseteq I$
The tautologies

$$\phi \lor \psi \iff \neg(\neg \phi \land \neg \psi)$$
$$AX \phi \iff \neg EX \neg \phi$$
$$AG \phi \iff \neg EF \neg \phi$$
$$EF \phi \iff \top \land EU \phi$$
$$EG \phi \iff \neg AF \neg \phi$$
$$\phi AU \psi \iff \neg((\neg \psi) EU \neg(\phi \lor \psi)) \land AF \psi$$

indicate that we can rewrite each formula to one only containing atomic propositions, $\neg$, $\land$, $EX$, $EU$, $AF$.

After preprocessing, our algorithm need only tackle these!
Given: A finite Kripke structure with vertices $V$ and edges $E$ and a labelling function $L$ assigning atomic propositions to vertices. Furthermore an atomic proposition $p$ to be checked.

Algorithm: Mark all vertices that have $p$ as a label.

Complexity: $O(|V|)$
Model-checking CTL: $\neg \phi$

Given: A set $V_\phi$ of vertices satisfying formula $\phi$.

Algorithm: Mark all vertices not belonging to $V_\phi$.

Complexity: $O(|V|)$
Model-checking CTL: $\phi \land \psi$

**Given:** Sets $V_\phi$ and $V_\psi$ of vertices satisfying formulae $\phi$ or $\psi$, resp.

**Algorithm:** Mark all vertices belonging to $V_\phi \cap V_\psi$.

**Complexity:** $O(|V|)$
Given: Set $V_\phi$ of vertices satisfying formulae $\phi$.

Algorithm: Mark all vertices that have a successor state in $V_\phi$.

Complexity: $O(|V| + |E|)$
Model-checking CTL: $\phi E U \psi$

Given: Sets $V_\phi$ and $V_\psi$ of vertices satisfying formulae $\phi$ or $\psi$, resp.

Algorithm: Incremental marking by

1. Mark all vertices belonging to $V_\psi$.
2. Repeat
   if there is a state in $V_\phi$ that has some successor state marked then mark it also
   until no new state is found.

Termination: Guaranteed due to finiteness of $V_\phi \subset V$.

Complexity: $O(|V| + |E|)$ if breadth-first search is used.
Given: Set $V_\phi$ of vertices satisfying formula $\phi$.

Algorithm: Incremental marking by
1. Mark all vertices belonging to $V_\phi$.
2. Repeat
   - if there is a state in $V$ that has all successor states marked
     then mark it also
   until no new state is found.

Termination: Guaranteed due to finiteness of $V$.

Complexity: $O(|V| \cdot (|V| + |E|))$. 
Model-checking CTL: \( \text{EG} \phi \), for efficiency

**Given:** Set \( V_\phi \) of vertices satisfying formula \( \phi \).

**Algorithm:** Incremental marking by

1. Strip Kripke structure to \( V_\phi \)-states:
   \[
   (V, E) \rightsquigarrow (V_\phi, E \cap (V_\phi \times V_\phi)).
   \]
   \( \rightsquigarrow \) **Complexity:** \( O(|V| + |E|) \)

2. Mark all states belonging to loops in the reduced graph.
   \( \rightsquigarrow \) **Complexity:** \( O(|V_\phi| + |E_\phi|) \) by identifying *strongly connected components*.

3. Repeat
   - if there is a state in \( V_\phi \) that has *some* successor states marked then mark it also
   - until no new state is found.
   \( \rightsquigarrow \) **Complexity:** \( O(|V_\phi| + |E_\phi|) \)

**Complexity:** \( O(|V| + |E|) \).
**Theorem:** It is decidable whether a finite Kripke structure \((V, E, L, I)\) satisfies a CTL formula \(\phi\).

The complexity of the decision procedure is \(O(|\phi| \cdot (|V| + |E|))\), i.e.

- linear in the size of the formula, given a fixed Kripke structure,
- linear in the size of the Kripke structure, given a fixed formula.

However, size of Kripke structure is exponential in number of parallel components in the system model.
Appendix

Fair Kripke Structures &
Fair CTL Model Checking
A fair Kripke structure is a pair \((K, \mathcal{F})\), where

- \(K = (V, E, L, I)\) is a Kripke structure
- \(\mathcal{F} \subseteq \mathcal{P}(V)\) is a set of vertex sets, called a fairness condition.

A fair path \(\pi\) in a fair Kripke structure \(((V, E, L, I), \mathcal{F})\) is an edge-consistent infinite sequence of vertices which visits each set \(F \in \mathcal{F}\) infinitely often:

- \(\pi \in V^\omega\),
- \((\pi_i, \pi_{i+1}) \in E\) for each \(i \in \mathbb{N}\),
- \(\forall F \in \mathcal{F}. \exists \infty i \in \mathbb{N}. \pi_i \in F\).

Note the similarity to (generalized) Büchi acceptance!
Fair CTL: Semantics

- $v, K, \mathcal{F} \models_F \text{EX} \phi$ iff there is a fair path $\pi$ in $K$ s.t. $v = \pi_0$ and $\pi_1, K, \mathcal{F} \models_F \phi$,
- $v, K, \mathcal{F} \models_F \text{AX} \phi$ iff all fair paths $\pi$ in $K$ with $v = \pi_0$ satisfy $\pi_1, K, \mathcal{F} \models_F \phi$,
- $v, K, \mathcal{F} \models_F \text{EF} \phi$ iff there is a fair path $\pi$ in $K$ s.t. $v = \pi_0$ and $\pi_i, K, \mathcal{F} \models_F \phi$ for some $i$,
- $v, K, \mathcal{F} \models_F \text{AF} \phi$ iff all fair paths $\pi$ in $K$ with $v = \pi_0$ satisfy $\pi_i, K, \mathcal{F} \models_F \phi$ for some $i$ (that may depend on the fair path),
- $v, K, \mathcal{F} \models_F \text{EG} \phi$ iff there is a fair path $\pi$ in $K$ s.t. $v = \pi_0$ and $\pi_i, K, \mathcal{F} \models_F \phi$ for all $i$,
- $v, K, \mathcal{F} \models_F \text{AG} \phi$ iff all fair paths $\pi$ in $K$ with $v = \pi_0$ satisfy $\pi_i, K, \mathcal{F} \models_F \phi$ for all $i$,
- $v, K, \mathcal{F} \models_F \phi \text{ EU } \psi$, iff there is a fair path $\pi$ in $K$ s.t. $v = \pi_0$ and some $k \in \mathbb{N}$ s.t. $\pi_i, K, \mathcal{F} \models_F \phi$ for each $i < k$ and $\pi_k, K, \mathcal{F} \models_F \psi$,
- $v, K, \mathcal{F} \models_F \phi \text{ AU } \psi$, iff all fair paths $\pi$ in $K$ with $v = \pi_0$ have some $k \in \mathbb{N}$ s.t. $\pi_i, K, \mathcal{F} \models_F \phi$ for each $i < k$ and $\pi_k, K, \mathcal{F} \models_F \psi$.

A fair Kripke structure $((V, E, L, I), \mathcal{F})$ satisfies $\phi$, denoted $((V, E, L, I), \mathcal{F}) \models_F \phi$, iff all its initial states satisfy $\phi$, i.e. iff $i_s, K, \mathcal{F} \models_F \phi$ for all $i_s \in I$. 

02917: CTL & Model Checking – p.33/37
Lemma: Given a fair Kripke structure $(((V, E, L, I), F)$, the set $Fair \subseteq V$ of states from which a fair path originates can be determined algorithmically.

Alg.: This is a problem of finding adequate SCCs:
1. Find all SCCs in K.
2. Select those SCCs that do contain at least one state from each fairness set $F \in \mathcal{F}$.
3. Find all states from which at least one of the selected SCCs is reachable.
Model-checking fair CTL: $\text{E} \text{X} \phi$

Given: Set $V_\phi$ of vertices fairly satisfying formulae $\phi$.

Algorithm: Mark all vertices that have a successor state in $V_\phi \cap \text{Fair}$.

Note that the intersection with $\text{Fair}$ is necessary even though the states in $V_\phi$ fairly satisfy $\phi$:

- $\phi$ may be an atomic proposition, in which case fairness is irrelevant;
- $\phi$ may start with an $\text{A}$ path quantifier that is trivially satisfied by non-existence of a fair path.
Model-checking fair CTL: $\phi \textsf{EU} \psi$

**Given:** Sets $V_{\phi}$ and $V_{\psi}$ of vertices fairly satisfying formulae $\phi$ or $\psi$, resp.

**Algorithm:** Incremental marking by

1. Mark all vertices belonging to $V_{\psi} \cap \textit{Fair}$.
2. Repeat
   - if there is a state in $V_{\phi}$ that has some successor state marked then mark it also
   until no new state is found.
Model-checking fair CTL: $\text{EG} \phi$

**Given:** Set $V_\phi$ of vertices fairly satisfying formula $\phi$.

**Algorithm:** Incremental marking by

1. Strip Kripke structure to $V_\phi$-states:
   \[ (V, E) \sim \to (V_\phi, E \cap (V_\phi \times V_\phi)) \].

2. Mark all states that can reach a *fair* SCC in the *reduced* graph.
   (Same algorithm as for finding the set $Fair$, yet applied to the reduced graph.)