IA169 System Verification and Assurance

CTL Model Checking

Jiří Barnat

Liner vs. Branching Time

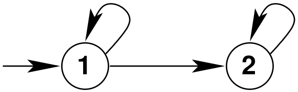
Pnueli, 1977

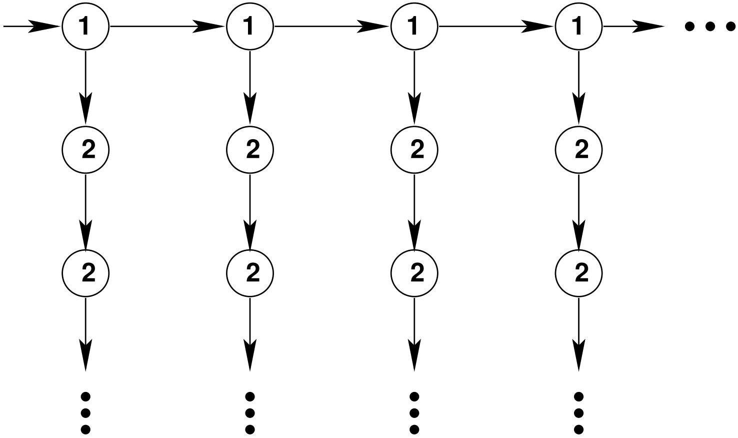
- System is viewed as a set of state sequences — **Runs**.
- System properties are given as properties of runs,
- … and can be described with a linear-time logic.

Clarke & Emerson, 1980

- System is viewed as a branching structure of possible executions from individual system states — **Computation Tree**.
- System properties are given as properties of the tree,
- … and can be described with a branching-time logic.

System and Computation Tree





Computation Tree Logic (CTL)

CTL Informally

Possible Future Computations

- For a given node of a computation tree, the sub-tree rooted in the given node describes all possible runs the system can still take.
- Every such a run is possible future computation.

CTL Formulae Allow For

- Specification of state qualities with atomic propositions.
- Quantify over possible future computations.
- Restrict the set of possible future computations with (quantified) LTL operators.

Example

- $\varphi \equiv EF(a)$
- It is possible to take a future computation such that *a* will hold true in the computation eventually.

Syntax of CTL

Let AP by a set of atomic propositions.

- If $p \in AP$, then p is a CTL formula.
- If φ is a CTL formula, then $\neg\varphi$ is a CTL formula.
- If φ and ψ are CTL formulae, then $\varphi \vee \psi$ is a CTL formula.
- If φ is a CTL formula, then $EX\,\varphi$ is a CTL formula.
- If φ and ψ are CTL formulae, then $E[\varphi\,U\,\psi]$ is a CTL formula.
- If φ and ψ are CTL formulae, then $A[\varphi\,U\,\psi]$ is a CTL formula.

Alternatively (Backus-Naur Form)

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid EX\,\varphi \mid E[\varphi\,U\,\varphi] \mid A[\varphi\,U\,\varphi]$$

Syntactic Shortcuts

Already Known

- The standard shortcuts from the propositional logic.
- Syntactic shortcuts from LTL
  - $F\,\varphi \equiv true\,U\,\varphi$
  - $G\,\varphi \equiv \neg F\,\neg\varphi$

Deduced CTL Operators

- $EF\,\varphi \equiv E[true\,U\,\varphi]$
- $AF\,\varphi \equiv A[true\,U\,\varphi]$
- $EG\,\varphi \equiv \neg AF\,\neg\varphi$
- $AG\,\varphi \equiv \neg EF\,\neg\varphi$
- $AX\,\varphi \equiv \neg EX\,\neg\varphi$

Models of CTL formulae

Model of a CTL formula

- Let AP be a set of atomic propositions.
- Model of a CTL formula is a state $s \in S$ of Kripke structure $M = (S, T, I, s_0)$.

Reminder

- Run of a Kripke structure is maximal path starting at the initial state of the structure.
- Finite maximal paths are viewed as infinite runs due to infinite repetition of the last state on the path.

Notation

- Let $s \in S$ be a state of Kripke structure $M = (S, T, I, s_0)$.
- $P_M(s) = \{\pi \mid \pi \text{ is a run initiated at state } s\}$

Semantics of CTL

Assumptions

- Let AP be a set of atomic propositions.
- Let $p \in AP$ be an atomic proposition.
- Let $s \in S$ be a state of Kripke structure $M = (S, T, I, s_0)$.
- Let φ, ψ denote syntactically correct CTL formulae.

Semantics

$$
\begin{array}{rcl}
s \models p & \text{iff} & p \in I(s) \\
s \models \neg\varphi & \text{iff} & \neg(s \models \varphi) \\
s \models \varphi \vee \psi & \text{iff} & s \models \varphi \text{ or } s \models \psi \\
s \models EX\,\varphi & \text{iff} & \exists \pi \in P_M(s).\pi(1) \models \varphi \\
s \models E[\varphi\, U\, \psi] & \text{iff} & \exists \pi \in P_M(s).(\exists k \geq 0.(\pi(k) \models \psi \text{ and} \\
 & & \forall 0 \leq i < k.\pi(i) \models \varphi)) \\
s \models A[\varphi\, U\, \psi] & \text{iff} & \forall \pi \in P_M(s).(\exists k \geq 0.(\pi(k) \models \psi \text{ and} \\
 & & \forall 0 \leq i < k.\pi(i) \models \varphi))
\end{array}
$$

Task

Atomic Propositions

- AP$=\{a, b, Req, Ack, Restart\}$

Express with CTL Formulae

- A state where a is true, but b is not, is reachable.
- Whenever system receives a request Req, it generates acknowledgement Ack eventually.
- In every run there are infinitely many b's.
- There is always an option to reset the system (reach state $Restart$).

Model Checking CTL

Problem Statements

Model Checking CTL

- Let $M = (S, T, I, s_0)$ be a Kripke structure.
- Let φ be a CTL formula.
- Does initial state of M satisfies φ?

Alternatively

- Let $M = (S, T, I, s_0)$ be a Kripke structure.
- Let φ be a CTL formula.
- Compute a set of states of M satisfying φ.

Above mentioned approaches are also referred to as to

- Local model checking problem — $M, s_0 \models \varphi$.
- Global model checking problem — $\{s \mid M, s \models \varphi\}$.

Algorithm for CTL Model Checking — Idea

Observation

- If the validity of formulae φ and ψ is known for all states, it is easy to deduce validity of formulae $\neg\varphi$, $\varphi \vee \psi$, $EX\,\varphi$, ...

CTL Model Checking – Sketch

- Let $M = (S, T, I)$ be a Kripke structure and φ a CTL Formula.
- A labelling function $label : S \to 2^{2^{\varphi}}$ is computed such that it gives validity of all sub-formulae of φ for all states of Kripke structure M.
- Obviously, $s_0 \models \varphi \iff \varphi \in label(s_0)$.
- Function *label* is computed gradually for individual sub-formulae of φ, starting with the simplest sub-formula and proceeding towards more complex sub-formulae, ending with φ itself.

Sub-formulae of a CTL Formula

Sub-formulae of formula φ

- Let φ be a CTL formula.
- The set of all sub-formulae of formula φ is denoted by 2^φ.
- 2^φ is defined inductively according to the structure of φ.

Inductive Definition of 2^φ

1) $\varphi \in 2^\varphi$ (φ is a sub-formula of φ)
2) If $\eta \in 2^\varphi$ and
   - $\eta \equiv \neg\psi$, then $\psi \in 2^\varphi$
   - $\eta \equiv \psi_1 \vee \psi_2$, then $\psi_1, \psi_2 \in 2^\varphi$
   - $\eta \equiv EX\,\psi$, then $\psi \in 2^\varphi$
   - $\eta \equiv E[\psi_1\, U\, \psi_2]$, then $\psi_1, \psi_2 \in 2^\varphi$
   - $\eta \equiv A[\psi_1\, U\, \psi_2]$, then $\psi_1, \psi_2 \in 2^\varphi$
3) Nothing else.

Equivalent Existential Form of CTL

Observation

- It is easier to prove validity of existential quantified modal operators than validity of universally quantified ones.
- For the purpose of verification of CTL-specified properties, it is possible to express the CTL formula in an equivalently expressive existential form of CTL.

Equivalent CTL Syntax

- $\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid EX\,\varphi \mid E[\varphi\, U\, \varphi] \mid EG\,\varphi$

Task

- Express formula $EG\,\varphi$ in the original syntax of CTL.
- Give accordingly modified definition of the set of sub-formulae of φ for the above mentioned equivalent syntax.

Algorithm for CTL Model-Checking

INPUT: Kripke structure $M = (S, T, I, s_0)$, CTL formula φ.
OUTPUT: *True*, if $s_0 \models \varphi$; *False* otherwise.

```
proc CTLMC(φ, M)
   label := I
   Solved := AP ∩ 2^φ
   while φ ∉ Solved do
         foreach ( η ∈ {¬ψ1, ψ1 ∨ ψ2, EX ψ1, E[ψ1 U ψ2], EG ψ1 | ψ1, ψ2 ∈ Solved})do
            if (η ∈ 2^φ and η ∉ Solved)
               then label := updateLabel(η, label, M)
                    Solved := Solved ∪ {η}
            fi
         od
   od
   return (φ ∈ label(s0))
end
```

```
proc updateLabel(η, label, M)
  if (η ≡ E[ψ1 U ψ2])
    then return checkEU(ψ1, ψ2, label, M)
  fi
  if (η ≡ EG ψ)
    then return checkEG(ψ, label, M)
  fi
  foreach ( s ∈ S)do
    if (η ≡ ¬ψ and ψ ∉ label(s)) or
      (η ≡ ψ1 ∨ ψ2 and (ψ1 ∈ label(s) ∨ ψ2 ∈ label(s))) or
      (η ≡ EX ψ and (∃t ∈ {t | (s, t) ∈ T} such that ψ ∈ label(t)))
      then label(s) := label(s) ∪ {η}
    fi
  od
  return label
end
```

Algorithm for CTL Model-Checking $E[\psi_1 \, U \, \psi_2]$

INPUT: Kripke structure $M = (S, T, I)$,
Labelling function $label : S \rightarrow 2^{\varphi}$, correct w.r.t validity of ψ_1 and ψ_2
OUTPUT: Labelling function $label : S \rightarrow 2^{\varphi}$, correct w.r.t $E[\psi_1 \, U \, \psi_2]$

```
proc checkEU(ψ1, ψ2, label, M)
  Q := {s | ψ2 ∈ label(s)}
  foreach ( s ∈ Q)do
    label(s) := label(s) ∪ {E[ψ1 U ψ2]}
  od
  while (Q ≠ ∅) do
        choose s ∈ Q
        Q := Q ∖ {s}
        foreach ( t ∈ {t | T(t, s)}) do        /* all immediate predecessors */
          if (E[ψ1 U ψ2] ∉ label(t) ∧ ψ1 ∈ label(t))
            then label(t) := label(t) ∪ {E[ψ1 U ψ2]}
                  Q := Q ∪ {t}
          fi
        od
  od
  return label
end
```

Strongly Connected Components

Sub-graph

- Let $G = (V, E)$ be a graph, ie. $E \subseteq V \times V$.
- Graph $G' = (V', E')$ is called sub-graph of G if it holds that $V' \subseteq V$ and $E' = E \cap V' \times V'$.

Sub-graph $C = (V', E')$ of $G = (V, E)$ is called

- Strongly Connected Component, if $\forall u, v \in V'$ it holds that $(u, v) \in E'^{*}$ and $(v, u) \in E'^{*}$.
- Maximal Strongly Connected Component (SCC), if C is strongly connected component and for every $v \in (V \setminus V')$ it is the case that $(V' \cup \{v\}, E \cap (V' \cup \{v\} \times V' \cup \{v\}))$ is not.
- Non-trivial SCC, if C is Strongly Connected Component and $E' \neq \emptyset$.

Algorithm for CTL Model-Checking $EG\,\psi$

INPUT: Kripke structure $M = (S, T, I, s_0)$,
Labelling function $label : S \to 2^{\varphi}$, correct w.r.t. ψ
OUTPUT: Labelling function $label : S \to 2^{\varphi}$, correct w.r.t. $EG\,\psi$

```
proc checkEG(ψ, label, M)
   S' := {s | ψ ∈ label(s)}
   SCC := {C | C is non-trivial SCC G' = (S', T ∩ S' × S')}
   Q := ⋃_{C∈SCC} {s | s ∈ C}
   foreach ( s ∈ Q)do
      label(s) := label(s) ∪ {EG ψ}
   od
   while Q ≠ ∅ do
         choose s ∈ Q
         Q := Q ∖ {s}
         foreach ( t ∈ (S' ∩ {t | T(t, s)}))do    /* all immediate predecessors in S' */
            if EG ψ ∉ label(t)
               then label(t) := label(t) ∪ {EG ψ}
                    Q := Q ∪ {t}
            fi
         od
   od
end
```

Complexity of Algorithm for CTL Model Checking

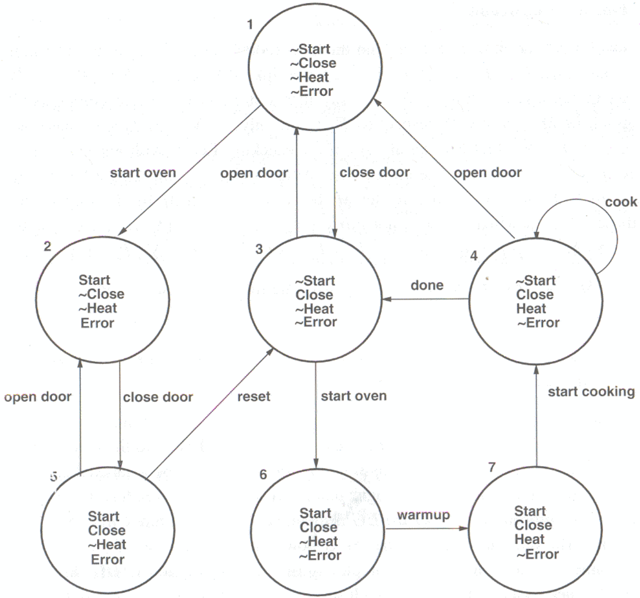
Observation

- Every CTL formula φ is made of at most $\mid \varphi \mid$ sub-formulae.
- Decomposition of every sub-graph of $G = (S, T)$ into SCCs can be done in time $\mathcal{O}(\mid S \mid + \mid T \mid)$.
- Every call to *updateLabel* terminates in time $\mathcal{O}(\mid S \mid + \mid T \mid)$.

Overall complexity

- Algorithm *CTLMC* exhibits $O(\mid \varphi \mid\mid S \mid)$ space and $O(\mid \varphi \mid (\mid S \mid + \mid T \mid))$ time complexity.

Example: Microwave oven $AG(Start \implies AF(Heat))$



Example: Microwave oven $AG(Start \implies AF(Heat))$

Transformation of formula $\varphi \equiv AG(Start \implies AF(Heat))$

- $AG(Start \implies AF(Heat))$
- $AG(\neg(Start \wedge \neg AF(Heat)))$
- $AG(\neg(Start \wedge EG(\neg Heat)))$
- $\neg EF(Start \wedge EG(\neg Heat))$
- $\neg E[true\, U\, (Start \wedge EG(\neg Heat))]$

Validity of sub-formulae $[S(\varphi) = \{s \mid s \models \varphi\}]$

- $S(Start) = \{2, 5, 6, 7\}$
- $S(Heat) = \{4, 7\}$
- $S(\neg Heat) = \{1, 2, 3, 5, 6\}$
- $S(EG(\neg Heat)) = \{1, 2, 3, 5\}$
- $S(Start \wedge EG(\neg Heat)) = \{2, 5\}$
- $S(E[true\, U\, (Start \wedge EG(\neg Heat))]) = \{1, 2, 3, 4, 5, 6, 7\}$
- $S(\neg E[true\, U\, (Start \wedge EG(\neg Heat))]) = \emptyset$

CTL*

CTL* as Extension of CTL

Observation

- Every use of temporal operator in a formula of CTL must be immediately preceded with a quantifier, i.e. use of a modal operator without quantification is not possible.

Logic CTL*

- Branching time logic.
- Similar to CTL.
- Unlike CTL, allows for standalone use of modal operators.

Example

- $A[p \wedge X(\neg p)]$ is CTL*, but is not CTL formula.

Types of CTL* formulae

- Quantifiers E and A are standalone operators in syntax construction rules. As a result there are two types of formulae in CTL: **path** and **state** formulae.
- Application of E and A operators on a path formula (formula of which model is a run of Kripke structure) results in a state formula (formula of which model is a state of Kripke structure)

Syntax of CTL*

state formula	$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid E\,\psi$
path formula	$\psi ::= \varphi \mid \neg\psi \mid \psi \vee \psi \mid X\,\psi \mid \psi\,U\,\psi$

Semantics of CTL*

Assumption

- Let AP be a set of atomic propositions, and $p \in AP$.
- Let $M = (S, T, I)$ be a Kripke structure.
- Let φ_i denote CTL* state formulae, and ψ_i denote CTL* state formulae.

Semantics

$$
\begin{array}{rcl}
M, s \models p & \text{iff} & p \in I(s) \\
M, s \models \neg\varphi_1 & \text{iff} & \neg(M, s \models \varphi_1) \\
M, s \models \varphi_1 \vee \varphi_2 & \text{iff} & M, s \models \varphi_1 \text{ or } M, s \models \varphi_2 \\
M, s \models E\,\psi_1 & \text{iff} & \exists \pi \in P_M(s). \pi \models \psi_1 \\
M, \pi \models \varphi_1 & \text{iff} & M, \pi(0) \models \varphi_1 \\
M, \pi \models \neg\psi_1 & \text{iff} & \neg(M, \pi \models \psi_1) \\
M, \pi \models \psi_1 \vee \psi_2 & \text{iff} & M, \pi \models \psi_1 \text{ or } M, \pi \models \psi_2 \\
M, \pi \models X\,\psi_1 & \text{iff} & M, \pi^1 \models \psi_1 \\
M, \pi \models \psi_1\,U\,\psi_2 & \text{iff} & \exists k \geq 0. (M, \pi^k \models \psi_2 \text{ and} \\
 & & \forall 0 \leq i < k. M, \pi^i \models \psi_1)
\end{array}
$$

Comparison of Expressive Power of LTL, CTL and CTL*

Model Unification

Observation

- Every LTL formula is a CTL* path formula.
- Every CTL formula is a CTL* state formula.
- Model of a path formula is a run of Kripke structure.
- Model of a state formula is a state of Kripke structure.
- Not very suitable for comparison.

Model Unification

- For the purpose of comparison we define how a CTL* path formula is evaluated in a state of Kripke structure.
- Let ψ be CTL* path formula, then

$$M, s \models \psi \quad \text{iff} \quad M, s \models A\,\psi$$

Motivation

Goals

- We intend to find out whether there are properties (formulae) that can be expressed in one of the logic, but cannot be expressed in another one.
- We intend to find out in which logic more properties can be expressed.
- We intend to identify concrete properties, that cannot be expressed in some other logic, i.e. to find out a formula of logic $\mathcal{L}_1$, for which an equivalent formula of logic $\mathcal{L}_2$ does not exist.

Formula Equivalence

- Formulae φ and ψ are equivalent if and only if for any possible Kripke structure $M = (S, T, I, s_0)$ and any state $s \in S$ it is true that

$$M, s \models \varphi \quad \text{iff} \quad M, s \models \psi.$$

Equivalently Expressive

- Temporal logic $\mathcal{L}_1$ and $\mathcal{L}_2$ have the same expressive power, if for all Kripke structures $M = (S, T, I, s_0)$ and states $s \in S$ it holds that

$$\forall \varphi \in \mathcal{L}_1.(\exists \psi \in \mathcal{L}_2.(M, s \models \varphi \iff M, s \models \psi)) \qquad (1)$$
$$\wedge\ \forall \psi \in \mathcal{L}_2.(\exists \varphi \in \mathcal{L}_1.(M, s \models \varphi \iff M, s \models \psi)). \qquad (2)$$

Less Expressiveness

- If only statement (1) is valid, then logic $\mathcal{L}_1$ is less expressive than logic $\mathcal{L}_2$, and vice versa.

Comparison of LTL, CTL, and CTL*

Theorem

- LTL and CTL are incomparable in expressive power.
  1) $AG(EF(q))$ is a CTL formula that cannot be expressed in LTL.
  2) $FG(q)$ is an LTL formula that cannot be expressed in CTL.

Example – Proof Sketch for 1)

- Find two different Kripke structures and identify two states that can be differentiated with CTL formula $AG(EF(q))$, but cannot be differentiated with any LTL formula (they generate the same set of runs).

Example – Intuition behind 2) [proof is too complex]

- Show that CTL formula $AF(AG(q))$ is not equivalent to LTL formula $FG(q)$.

Comparison of LTL, CTL, and CTL*

Consequence

- CTL* is strictly more expressive than LTL.
  - Every LTL formula is a CTL* formula.
  - CTL* formula $AG(EFq)$ is not expressible in LTL.

Consequence 2

- CTL* is strictly more expressive than CTL.
  - Every CTL formula is a CTL* formula.
  - CTL* formula $FG(q)$ is not expressible in CTL.

Observation

- There are properties expressible on both LTL and CTL.
  - CTL formula $A[p\, U\, q]$ is equivalent to LTL formula $p\, U\, q$.

Homework

Homework

- Solve The wolf, goat and cabbage problem with NuSMV
- Moshe Vardi: *Branching vs. Linear Time: Final Showdown*