

# IA169 System Verification and Assurance

## CTL Model Checking

Jiří Barnat

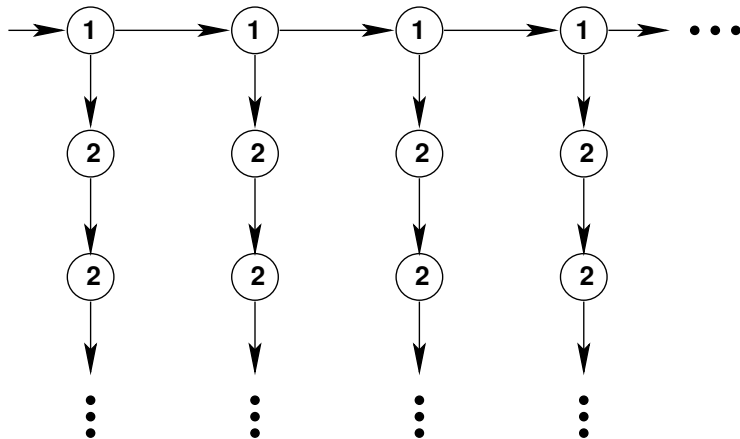
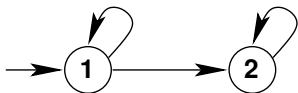
## **Pnueli, 1977**

- 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)

## 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.

Let  $AP$  be a set of atomic propositions.

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

Alternatively (Backus-Naur Form)

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

## 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

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

## 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  $M = (S, T, I, s_0)$ .
- $P_M(s) = \{\pi \mid \pi \text{ is a run initiated at state } s\}$



## 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  $\varphi, \psi$  denote syntactically correct CTL formulae.

## Semantics

$$s \models p \quad \text{iff} \quad p \in I(s)$$

$$s \models \neg\varphi \quad \text{iff} \quad \neg(s \models \varphi)$$

$$s \models \varphi \vee \psi \quad \text{iff} \quad s \models \varphi \text{ or } s \models \psi$$

$$s \models EX \varphi \quad \text{iff} \quad \exists \pi \in P_M(s). \pi(1) \models \varphi$$

$$s \models E[\varphi U \psi] \quad \text{iff} \quad \exists \pi \in P_M(s). (\exists k \geq 0. (\pi(k) \models \psi \text{ and} \\ \forall 0 \leq i < k. \pi(i) \models \varphi))$$

$$s \models A[\varphi U \psi] \quad \text{iff} \quad \forall \pi \in P_M(s). (\exists k \geq 0. (\pi(k) \models \psi \text{ and} \\ \forall 0 \leq i < k. \pi(i) \models \varphi))$$

## Atomic Propositions

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

## Express with CTL Formulae

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

## Model Checking CTL

## Model Checking CTL

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

## Alternatively

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

## Above mentioned approaches are also referred to as to

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

## Observation

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

## CTL Model Checking – Sketch

- Let  $M = (S, T, I)$  be a Kripke structure and  $\varphi$  a CTL Formula.
- A labelling function  $label : S \rightarrow 2^{2^\varphi}$  is computed such that it gives validity of all sub-formulae of  $\varphi$  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  $\varphi$ , starting with the simplest sub-formula and proceeding towards more complex sub-formulae, ending with  $\varphi$  itself.

## Sub-formulae of formula $\varphi$

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

## Inductive Definition of $2^\varphi$

- 1)  $\varphi \in 2^\varphi$  ( $\varphi$  is a sub-formula of  $\varphi$ )
- 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.

## Observation

- 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  $\varphi$  for the above mentioned equivalent syntax.

# Algorithm for CTL Model-Checking

INPUT: Kripke structure  $M = (S, T, I, s_0)$ , CTL formula  $\varphi$ .

OUTPUT: *True*, if  $s_0 \models \varphi$ ; *False* otherwise.

```
proc CTLMC( $\varphi, M$ )
  label := I
  Solved := AP  $\cap$   $2^\varphi$ 
  while  $\varphi \notin$  Solved do
    foreach (  $\eta \in \{\neg\psi_1, \psi_1 \vee \psi_2, EX \psi_1, E[\psi_1 U \psi_2], EG \psi_1 \mid \psi_1, \psi_2 \in$  Solved $\}$ ) do
      if ( $\eta \in 2^\varphi$  and  $\eta \notin$  Solved)
        then label := updateLabel( $\eta, label, M$ )
           Solved := Solved  $\cup$   $\{\eta\}$ 
        fi
      od
    od
  return ( $\varphi \in label(s_0)$ )
end
```



```
proc updateLabel( $\eta$ , label, M)
  if ( $\eta \equiv E[\psi_1 U \psi_2]$ )
    then return checkEU( $\psi_1, \psi_2, label, M$ )
  fi
  if ( $\eta \equiv EG \psi$ )
    then return checkEG( $\psi, label, M$ )
  fi
  foreach (  $s \in S$ )do
    if ( $\eta \equiv \neg\psi$  and  $\psi \notin label(s)$ ) or
      ( $\eta \equiv \psi_1 \vee \psi_2$  and ( $\psi_1 \in label(s) \vee \psi_2 \in label(s)$ )) or
      ( $\eta \equiv EX \psi$  and ( $\exists t \in \{t \mid (s, t) \in T\}$  such that  $\psi \in label(t)$ ))
      then  $label(s) := label(s) \cup \{\eta\}$ 
    fi
  od
  return label
end
```

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

```

proc checkEU( $\psi_1, \psi_2, label, M$ )
  Q := {s |  $\psi_2 \in label(s)$ }
  foreach ( s  $\in$  Q)do
     $label(s) := label(s) \cup \{E[\psi_1 U \psi_2]\}$ 
  od
  while (Q  $\neq$   $\emptyset$ ) do
    choose s  $\in$  Q
    Q := Q  $\setminus$  {s}
    foreach ( t  $\in$  {t | T(t, s)} ) do          /* all immediate predecessors */
      if ( $E[\psi_1 U \psi_2] \notin label(t) \wedge \psi_1 \in label(t)$ )
        then  $label(t) := label(t) \cup \{E[\psi_1 U \psi_2]\}$ 
           Q := Q  $\cup$  {t}
        fi
      od
    od
  return label
end

```

## 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  $E' \neq \emptyset$ .

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

```

proc checkEG( $\psi$ , label, M)
  S' := {s |  $\psi \in label(s)$ }
  SCC := {C | C is non-trivial SCC  $G' = (S', T \cap S' \times S')$ }
  Q :=  $\bigcup_{C \in SCC} \{s \mid s \in C\}$ 
  foreach ( s  $\in$  Q)do
    label(s) := label(s)  $\cup$  {EG  $\psi$ }
  od
  while Q  $\neq$   $\emptyset$  do
    choose s  $\in$  Q
    Q := Q  $\setminus$  {s}
    foreach ( t  $\in$  ( $S' \cap \{t \mid T(t, s)\}$ ))do /* all immediate predecessors in S' */
      if EG  $\psi \notin label(t)$ 
        then label(t) := label(t)  $\cup$  {EG  $\psi$ }
           Q := Q  $\cup$  {t}
        fi
      od
    od
  od
end

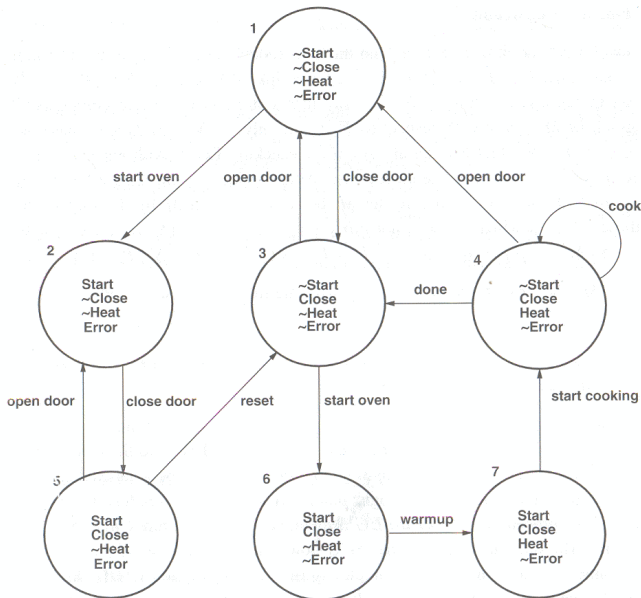
```

## Observation

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

## Overall complexity

- Algorithm *CTLMC* exhibits  $\mathcal{O}(|\varphi| |S|)$  space and  $\mathcal{O}(|\varphi| (|S| + |T|))$  time complexity.



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\*



## 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

$$\psi ::= \varphi \mid \neg\psi \mid \psi \vee \psi \mid X\psi \mid \psi U\psi$$

## Assumption

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

## Semantics

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

## Comparison of Expressive Power of LTL, CTL and CTL\*

## 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  $\psi$  be CTL\* path formula, then

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

## Goal

- 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  $\varphi$  and  $\psi$  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

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

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

## Less Expressiveness

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

## 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  $FG(q)$ .



## 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  $p U q$ .

## Practicals

- System modelling and property specification for NuSMV (NuXmv) model checker.
- Follow the NuSMV 2.5 tutorial

## Homework

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