Algorithms for Model Checking (2IMF35)

Lecture 1

The temporal logics CTL*, CTL and LTL: syntax and semantics

Tim Willemse
(timw@win.tue.nl)
http://www.win.tue.nl/∼timw
MF 6.073

Outline

Motivation

Model checking is an automated verification method. It can be used to check that a requirement holds for a model of a system.

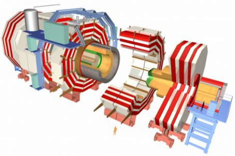
- A (software or hardware) system is usually modelled in a particular specification language
- The requirements are specified as properties in some temporal logic
- As an intermediate step, a state space is generated from the specification. This is a graph, representing all possible behaviours
- A model checking algorithm decides whether the property holds for the model: the property can be verified or refuted. Sometimes, witnesses or counter examples can be provided

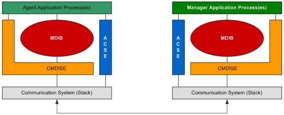
In practice, model checking proves to be an effective method to detect many *bugs* in early design phases

Motivation

Example

- What: control system for the Compact Muon Sollenoid detector at the LHC (CERN)
- Bugs: various kinds of livelocks





- What: Medical/health device communication standard IEEE 11073
- Bugs: devices can interpret data in different units of measurements

---

- What: Implantable Pulse Generators (pacemaker)
- Bugs: deadlock

Motivation

Complexity of model checking arises from:

- State space explosion: the state space is usually much larger than the specification
- Expressive logics have complex model checking algorithms

Ways to deal with the state space explosion:

- equivalence reduction: remove states with identical potentials from a state space
- on-the-fly: integrate the generation and verification phases, to prune the state space
- symbolic model checking: represent sets of states by clever data structures
- partial-order reduction: ignore some executions, because they are covered by others
- abstraction: remove details by working on approximations

Outline

The behaviour of a system is modelled by a graph consisting of:

- nodes, representing states of the system (e.g. the value of a program counter, variables, registers, stack/heap contents, etc.)
- edges, representing state transitions of the system (e.g. events, input/output actions, internal computations)

Information can be put in states or on transitions (or both).

- Kripke Structures (KS): information on states, called atomic propositions
- Labelled Transition Systems (LTS): information on edges, called action labels

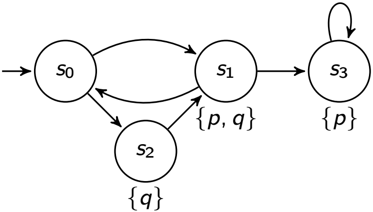
Today: only Kripke Structures

Let AP be a set of atomic propositions. A Kripke Structure over AP is a structure $M = \langle S, S_0, R, L \rangle$, where

- S is a finite set of states
- $S_0 \subseteq S$ is a non-empty set of initial states
- $R \subseteq S \times S$ is a total binary relation on S, representing the set of transitions. totality: for all $s \in S$, there exists $t \in S$, such that $(s, t) \in R$.
- $L\colon S \to 2^{AP}$, labels each state with the set of atomic propositions that hold in that state

Conventions:

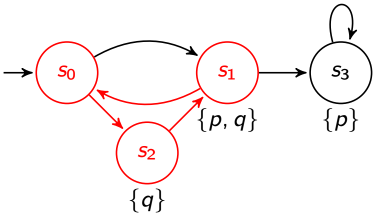
- Sometimes S_0 is irrelevant and dropped; sometimes it is a single state, in which case it is written as s_0
- Instead of $(s, t) \in R$, we write sRt



This is a Kripke Structure over AP, $M = \langle S, S_0, R, L\rangle$ as follows:

- $AP = \{p, q\}$
- $S = \{s_0, s_1, s_2, s_3\}$
- $S_0 = \{s_0\}$
- $R = \{(s_0, s_1), (s_1, s_0), (s_1, s_3), (s_3, s_3), (s_0, s_2), (s_2, s_1)\}$
- $L(s_0) = \emptyset, \quad L(s_1) = \{p, q\}$
  $L(s_2) = \{q\}, \quad L(s_3) = \{p\}$

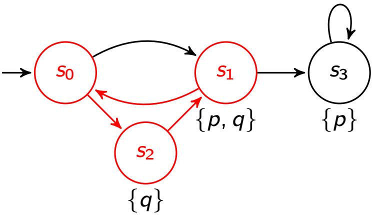
Note: without the self-loop (s_3, s_3), R would not be total and we would not have a Kripke structure



Terminology

Given a fixed Kripke Structure $M = \langle S, R, L \rangle$.

- A *path* π is an infinite sequence of states $s_0\ s_1 \ldots$ such that for all $i \in \mathbb{N}$: $s_i \in S$ and $s_i R s_{i+1}$
- Given a path $\pi = s_0\ s_1\ s_2\ \ldots$
  - $\pi(i)$ denotes the i-th state (counting from 0): s_i
  - π^i denotes the suffix of π starting at i: $s_i\ s_{i+1}\ \ldots$
- path(s) denotes the set of paths starting at s: $\{\pi \mid \pi(0) = s\}$



Terminology

Given a fixed Kripke Structure $M = \langle S, R, L \rangle$.

- A *path* π is an infinite sequence of states $s_0\ s_1 \ldots$ such that for all $i \in \mathbb{N}$: $s_i \in S$ and $s_i R s_{i+1}$
- Given a path $\pi = s_0\ s_1\ s_2\ \ldots$
  - $\pi(i)$ denotes the i-th state (counting from 0): s_i
  - π^i denotes the suffix of π starting at i: $s_i\ s_{i+1}\ \ldots$
- path(s) denotes the set of paths starting at s: $\{\pi \mid \pi(0) = s\}$

In the Kripke Structure above:

$(s_0\ s_2\ s_1)^\omega \in \text{path}(s_0)$, $\quad ((s_0\ s_2\ s_1)^\omega)(3) = s_0$, $\quad ((s_0\ s_2\ s_1)^\omega)^3 = (s_0\ s_2\ s_1)^\omega$

Technische Universiteit Eindhoven University of Technology

Outline

TU/e Technische Universiteit Eindhoven University of Technology

CTL* is the Full Computation Tree Logic

- CTL* formulae express properties over states or paths
- CTL* has the following temporal operators, which are used to express properties of paths: neXt, Future, Globally, Until, Releases
  The operators have the following intuitive meaning:
  - X f: f holds in the next state in this path
  - F f: f holds somewhere in this path
  - G f: f holds everywhere on this path
  - [f U g]: g holds somewhere on this path, and f holds in all preceding states
  - [f R g]: g holds as long as f did not hold before

Example

F G p versus G F p: *almost always* versus *infinitely often*

Temporal Logics: CTL*

CTL* consists of:

- Atomic propositions (AP)
- Boolean connectives: $\neg$ (not), $\vee$ (or), $\wedge$ (and)
- Temporal operators (on paths, see previous slide)
- Path quantifiers (on states, see below)

Path quantifiers are capable of expressing properties on a system's branching structure:

for All paths versus there Exists a path

Path quantifiers have the following intuitive meaning:

- A f: f holds for all paths from this state
- E f: f holds for at least one path from this state

CTL* state formulae ($\mathcal{S}$) and path formulae ($\mathcal{P}$) are defined simultaneously by induction:

$$\begin{array}{lcl}\mathcal{S} & ::= & \mathsf{true} \mid \mathsf{false} \mid AP \mid \neg\mathcal{S} \mid \mathcal{S}\wedge\mathcal{S} \mid \mathcal{S}\vee\mathcal{S} \mid \mathsf{E}\ \mathcal{P} \mid \mathsf{A}\ \mathcal{P} \\ \mathcal{P} & ::= & \mathcal{S} \mid \neg\mathcal{P} \mid \mathcal{P}\wedge\mathcal{P} \mid \mathcal{P}\vee\mathcal{P} \mid \mathsf{X}\ \mathcal{P} \mid \mathsf{F}\ \mathcal{P} \mid \mathsf{G}\ \mathcal{P} \mid [\mathcal{P}\ \mathsf{U}\ \mathcal{P}] \mid [\mathcal{P}\ \mathsf{R}\ \mathcal{P}]\end{array}$$

Summarising:

- State formulae ($\mathcal{S}$) are:
  - constants true and false and atomic propositions (basis)
  - Boolean combinations of state formulae
  - quantified path formulae
- Path formulae ($\mathcal{P}$) are:
  - state formulae (basis)
  - Boolean combinations of path formulae
  - temporal combinations of path formulae

The semantics of CTL* state formulae and path formulae is defined relative to a fixed Kripke Structure $M = \langle S, S_0, R, L \rangle$ over AP:

For state formulae:

$$
\begin{array}{lll}
s \models \text{true} & & \\
s \not\models \text{false} & & \\
s \models p & \text{iff} & p \in L(s) \\
s \models \neg f & \text{iff} & s \not\models f \\
s \models f \wedge g & \text{iff} & s \models f \text{ and } s \models g \\
s \models f \vee g & \text{iff} & s \models f \text{ or } s \models g \\
s \models \mathrm{E}\ f & \text{iff} & \text{for some } \pi \in \text{path}(s), \pi \models f \\
s \models \mathrm{A}\ f & \text{iff} & \text{for all } \pi \in \text{path}(s), \pi \models f
\end{array}
$$

TU/e Technische Universiteit Eindhoven University of Technology

The semantics of CTL* state formulae and path formulae is defined relative to a fixed Kripke Structure $M = \langle S, S_0, R, L\rangle$ over AP:

For path formulae:

$$
\begin{array}{lll}
\pi \models f & \text{iff} & \pi(0) \models f \qquad \text{(if } f \text{ is a state formula)} \\
\pi \models \neg f & \text{iff} & \pi \not\models f \\
\pi \models f \wedge g & \text{iff} & \pi \models f \text{ and } \pi \models g \\
\pi \models f \vee g & \text{iff} & \pi \models f \text{ or } \pi \models g \\
\pi \models \mathrm{X}\ f & \text{iff} & \pi^1 \models f \\
\pi \models \mathrm{F}\ f & \text{iff} & \text{for some } i \geq 0, \pi^i \models f \\
\pi \models \mathrm{G}\ f & \text{iff} & \text{for all } i \geq 0, \pi^i \models f \\
\pi \models [f\ \mathrm{U}\ g] & \text{iff} & \exists i \geq 0.\ \pi^i \models g \wedge \forall j < i.\ \pi^j \models f \\
\pi \models [f\ \mathrm{R}\ g] & \text{iff} & \forall j \geq 0.\ ((\forall i < j.\ \pi^i \not\models f) \Rightarrow \pi^j \models g)
\end{array}
$$

A property f is satisfied by a Kripke Structure $M = \langle S, S_0, R, L\rangle$, denoted $M \models f$, iff $\forall s \in S_0.\ M, s \models f$.

Equivalence between two CTL* properties is defined as follows:

$$f \equiv g \ \text{ iff } \ \forall M\ \forall s\ .(M, s \models f \quad \Leftrightarrow \quad M, s \models g)$$

Likewise for paths

According to the semantics, we can derive several dualities:

- $\neg \mathrm{G}\ f \equiv \mathrm{F}\ (\neg f)$
- $\neg\neg f \equiv f$
- $\neg(f \wedge g) \equiv \neg f \vee \neg g$
- $\neg \mathrm{A}\ f \equiv \mathrm{E}\ (\neg f)$
- $\neg[f\ \mathrm{R}\ g] \equiv [(\neg f)\ \mathrm{U}\ (\neg g)]$
- $\neg \mathrm{X}\ f \equiv \mathrm{X}\ (\neg f)$
- $\mathrm{F}\ f \equiv [\mathrm{true}\ \mathrm{U}\ f]$

So all CTL* properties can be expressed using only: $\neg, \mathrm{true}, \vee, \mathrm{X}\ , [\ \mathrm{U}\ ], \mathrm{E}$

Two simpler sublogics of CTL* are defined:

- LTL: linear time logic
  - checks temporal operators along single paths
  - pro: -counter examples are easy: "lasso"
    -nice automata-theoretic algorithm
  - typical tool: SPIN
- CTL: computation tree logic
  - branching time logic
  - temporal operators should be preceded by path quantifiers
  - pro: -efficient model checking algorithm
    -amenable to symbolic techniques
  - typical tool: nuSMV

The expressive power of LTL and CTL is incomparable.

LTL state formulae ($\mathcal{S}$) and path formulae ($\mathcal{P}$):

$$\begin{aligned}
\mathcal{S} \quad &::= \mathsf{A}\ \mathcal{P} \\
\mathcal{P} \quad &::= \mathsf{true} \mid \mathsf{false} \mid AP \mid \neg\mathcal{P} \mid \mathcal{P} \wedge \mathcal{P} \mid \mathcal{P} \vee \mathcal{P} \\
&\qquad \mid \mathsf{X}\ \mathcal{P} \mid \mathsf{F}\ \mathcal{P} \mid \mathsf{G}\ \mathcal{P} \mid [\mathcal{P}\ \mathsf{U}\ \mathcal{P}] \mid [\mathcal{P}\ \mathsf{R}\ \mathcal{P}]
\end{aligned}$$

Summarising:

- The only state formulae are:
  - all-quantified path formulae (hence, the A is sometimes omitted)
- Path formulae are:
  - constants true and false and atomic propositions
  - Boolean combinations of path formulae
  - temporal combinations of path formulae

Example

LTL expressions: $\mathsf{A\ F\ G}\ p$, $\mathsf{A}\ (\neg(\mathsf{G\ F}\ p) \vee \mathsf{F}\ q)$;
syntactically not in LTL: $\mathsf{A\ F\ A\ G}\ p$, $\mathsf{A\ G\ E\ F}\ p$
Question: $\mathsf{A\ F\ G}\ p \stackrel{?}{\equiv} \mathsf{A\ F\ A\ G}\ p$

TU/e Technische Universiteit Eindhoven University of Technology

CTL state formulae ($\mathcal{S}$) and path formulae ($\mathcal{P}$):

$$\begin{array}{ll} \mathcal{S} & ::= \mathsf{true} \mid \mathsf{false} \mid AP \mid \neg\mathcal{S} \mid \mathcal{S} \vee \mathcal{S} \mid \mathsf{E}\ \mathcal{P} \mid \mathsf{A}\ \mathcal{P} \\ \mathcal{P} & ::= \mathsf{X}\ \mathcal{S} \mid \mathsf{F}\ \mathcal{S} \mid \mathsf{G}\ \mathcal{S} \mid [\mathcal{S}\ \mathsf{U}\ \mathcal{S}] \mid [\mathcal{S}\ \mathsf{R}\ \mathcal{S}] \end{array}$$

Summarising:

- State formulae are:
  - constants true and false and atomic propositions
  - Boolean combinations of state formulae
  - quantified path formulae
- The only path formulae are:
  - temporal combinations of state formulae

Example

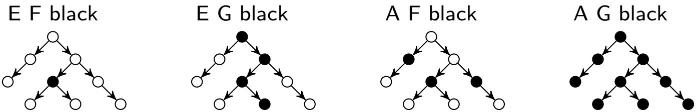
CTL expressions: $\mathsf{A}\ \mathsf{G}\ \mathsf{E}\ \mathsf{F}\ p$, $\mathsf{E}\ [p\ \mathsf{U}\ (\mathsf{E}\ \mathsf{X}\ q)]$;
not in CTL: $\mathsf{A}\ \mathsf{F}\ \mathsf{G}\ p$, $\mathsf{A}\ \mathsf{X}\ \mathsf{X}\ p$, $\mathsf{E}\ [p\ \mathsf{U}\ (\mathsf{X}\ q)]$

Question: $\mathsf{A}\ \mathsf{X}\ \mathsf{X}\ p \stackrel{?}{\equiv} \mathsf{A}\ \mathsf{X}\ \mathsf{A}\ \mathsf{X}\ p$

Temporal Logics: CTL and LTL

Alternative view: CTL has only state formulae, with the following ten temporal combinators:

- A X and E X : for all/some next state
- A F and E F : inevitably and potentially
- A G and E G : invariantly and potentially always
- A [U] and E [U]: for all/some paths, until
- A [R] and E [R]: for all/some paths, releases



Technische Universiteit Eindhoven University of Technology

Temporal Logics: CTL and LTL

For CTL, only the following operators are needed:

- Boolean connectives: $\neg$, $\vee$ and constants true and AP
- Temporal combinations: $\mathsf{E\,X}$, $\mathsf{E\,G}$, $\mathsf{E\,[\;U\;]}$

Standard transformations (derived from CTL*):

1. $\mathsf{E\,F}\,f \equiv \mathsf{E}\,[\text{true}\ \mathsf{U}\ f]$
2. $\mathsf{A\,X}\,f \equiv \neg\mathsf{E\,X}\,(\neg f)$
3. $\mathsf{A\,G}\,f \equiv \neg\mathsf{E\,F}\,(\neg f)$
4. $\mathsf{A\,F}\,f \equiv \neg\mathsf{E\,G}\,(\neg f)$
5. $\mathsf{A}\,[f\ \mathsf{R}\ g] \equiv \neg\mathsf{E}\,[(\neg f)\ \mathsf{U}\ (\neg g)]$
6. $\mathsf{E}\,[f\ \mathsf{R}\ g] \equiv \neg\mathsf{A}\,[(\neg f)\ \mathsf{U}\ (\neg g)]$

To remove $\mathsf{A\,[\;U\;]}$, note that:

- $[f\ \mathsf{R}\ g] \equiv [g\ \mathsf{U}\ (f \wedge g)] \vee \mathsf{G}\,g$
- $\mathsf{A}\,[f\ \mathsf{U}\ g] \equiv \neg\mathsf{E}\,[(\neg f)\ \mathsf{R}\ (\neg g)]$ (rule 6)
- $\mathsf{E}\,(f \vee g) \equiv \mathsf{E}\,f \vee \mathsf{E}\,g$

from this, we obtain $\mathsf{A}\,[f\ \mathsf{U}\ g] \equiv \neg\mathsf{E}\,[(\neg g)\ \mathsf{U}\ (\neg(f \vee g))] \wedge \neg\mathsf{E\,G}\,(\neg g)$

Technische Universiteit Eindhoven University of Technology

Example (CTL versus LTL)

Is there an equivalent CTL formula for the LTL formula A F $(p \wedge \mathrm{X}\ p)$?

- A F $(p \wedge \mathrm{X}\ p) \not\equiv$ A F $(p \wedge \mathrm{A}\ \mathrm{X}\ p)$: $M_1 \models$ A F $(p \wedge \mathrm{X}\ p)$ but $M_1 \not\models$ A F $(p \wedge \mathrm{A}\ \mathrm{X}\ p)$
- A F $(p \wedge \mathrm{X}\ p) \not\equiv$ A F $(p \wedge \mathrm{E}\ \mathrm{X}\ p)$: $M_2 \not\models$ A F $(p \wedge \mathrm{X}\ p)$ but $M_2 \models$ A F $(p \wedge \mathrm{E}\ \mathrm{X}\ p)$

- Actually: A F $(p \wedge \mathrm{X}\ p)$ is not expressible in CTL (does not follow from these observations)
- Open problem: which LTL formulae admit equivalent CTL formulae.
- The reverse problem (which CTL formulae are equivalent to an LTL formula) is solved [Clarke and Draghicescu]

Technische Universiteit Eindhoven University of Technology

Outline

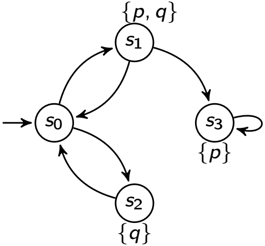
Motivation

Kripke Structures

Temporal Logics
- CTL*
- CTL and LTL

Exercise

Exercise



CTL* formulae: p, E $[q \text{ R } p]$, E F G p, A G F p, A G E F p, A G F $(p \wedge \text{X } q)$, A G $(\neg q \vee \text{F } p)$, A $((\text{G } p) \vee (\text{F } q))$

- For each formula, indicate whether it is (syntactically) in LTL and/or CTL
- Determine for each formula in which states of the above Kripke Structure it holds