

Algorithms for Model Checking

2IW50 – 2006

Lecture 1:

The temporal logics CTL*, CTL, LTL
– syntax, semantics and fairness –
Chapter 2.1, Chapter 3

Jaco van de Pol

vdpol@cwi.nl

<http://www.cwi.nl/~vdpol/amc.html>

Technical University Eindhoven
Centrum voor Wiskunde en Informatica

Logistics

- Teacher: Jaco van de Pol, vdpol@cwi.nl
Monday-Thursday: CWI, Amsterdam;
Friday: TU/e, room HG 6.88, tel 2963
- Website for this course:
<http://www.cwi.nl/~vdpol/amc.html>
- Mandatory Book: Model Checking,
by: *E.M. Clarke, O. Grumberg, D.A. Peled*.
MIT Press, 1999, ISBN 0-262-03270-8.
- 1 trimester lecture, 2 blocks (A+B)
- 2 possibilities for examination:
end of blocks B+C
- The examinations will be open book

Model Checking: Motivation

Model checking is an automated verification method, 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, containing all possible behaviour
- 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.

Complexity of Model Checking

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 solve the state space explosion:

- 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 certain execution orders, because they are covered by others
- abstraction: remove details by working on conservative over-approximations

OVERVIEW

1. Model Checking: Motivation
2. Kripke Structures
3. Syntax of CTL*
4. Semantics
5. Dualities
6. Restrictions CTL and LTL
7. Fairness

State Spaces

The behaviour of a system is modeled by a graph, consisting of:

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

Information can be put in states or on transitions. So there are two kind of models:

- Kripke structures: information on states, called “atomic propositions”
- Labeled Transition Systems: information on transitions, called “action labels”

Mixes are possible as well

Kripke Structures

Let AP be a set of atomic propositions

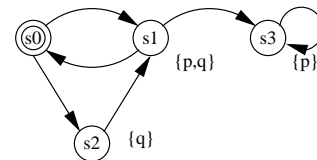
A Kripke structure over AP is a structure $M = (S, S_0, R, L)$, where

- S is a finite set of states
- $S_0 \subseteq S$ is the set of initial states
- $R \subseteq S \times S$ is a total binary relation on S , so for all $s \in S$, there exists $t \in S$, such that sRt . R represents the set of transitions
- $L : S \rightarrow \mathcal{P}(AP)$, assigns the set of atomic propositions that hold in a state

Sometimes S_0 is irrelevant and dropped; sometimes it is a single state.

A path π in M is an infinite sequence s_0, s_1, s_2, \dots such that for all i , $s_i \in S$ and $s_i R s_{i+1}$.

Kripke Structure: Example



This is a Kripke Structure over AP , $M = (S, S_0, R, L)$ as follows:

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

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

CTL* operators (1)

CTL* = (Full) Computation Tree Logic

CTL* formulas hold for states and/or infinite paths

CTL* has the following temporal operators, which are used to express properties of infinite paths:

neXt, **Future**, **Globally**, **Until**, **Releases**

They 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

Examples: **F G** p versus **G F** p .
(almost always, infinitely often)

CTL* operators (2)

CTL* consists of:

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

The path quantifiers of CTL* are needed to express properties on the branching structure of a system:

for **All** paths, there **Exists** a path

They have the following intuitive meaning:

- **A** f : f holds for all paths from this state
- **E** f : f holds for some path from this state

CTL* syntax (formal)

CTL* state formulas (\mathcal{S}) and path formulas (\mathcal{P}) are defined simultaneously by induction as follows:

$$\begin{aligned}\mathcal{S} &::= AP \mid \neg \mathcal{S} \mid \mathcal{S} \wedge \mathcal{S} \mid \mathcal{S} \vee \mathcal{S} \mid \mathbf{E} \mathcal{P} \mid \mathbf{A} \mathcal{P} \\ \mathcal{P} &::= \mathcal{S} \mid \neg \mathcal{P} \mid \mathcal{P} \wedge \mathcal{P} \mid \mathcal{P} \vee \mathcal{P} \mid \\ &\quad \mathbf{X} \mathcal{P} \mid \mathbf{F} \mathcal{P} \mid \mathbf{G} \mathcal{P} \mid \mathcal{P} \mathbf{U} \mathcal{P} \mid \mathcal{P} \mathbf{R} \mathcal{P}\end{aligned}$$

Summarizing:

- State formulas (\mathcal{S}) are:
 - atomic propositions (basis)
 - Boolean combinations of state formulas
 - quantified path formulas
- Path formulas (\mathcal{P}) are:
 - state formulas (basis)
 - Boolean combinations of path formulas
 - temporal combinations of path formulas

Paths in Kripke Structures

Some definitions, given a fixed Kripke structure $M = (S, R, L)$: (set of initial states is dropped)

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

Next the semantics of a CTL* formula will be defined by simultaneous recursion on formulas:

- $s \models f$: state formula f holds in state s
- $\pi \models g$: path formula g holds along path π

CTL* semantics

$$s \models p \Leftrightarrow p \in L(s)$$

$$s \models \neg f \Leftrightarrow s \not\models f$$

$$s \models f \vee g \Leftrightarrow s \models f \text{ or } s \models g$$

$$s \models f \wedge g \Leftrightarrow s \models f \text{ and } s \models g$$

$$s \models \mathbf{E}f \Leftrightarrow \text{for some } \pi \in \text{path}(s), \pi \models f$$

$$s \models \mathbf{A}f \Leftrightarrow \text{for all } \pi \in \text{path}(s), \pi \models f$$

$$\pi \models f \Leftrightarrow \pi(0) \models f \quad \text{if } f \text{ is a state formula}$$

$$\pi \models \neg f \Leftrightarrow \pi \not\models f$$

$$\pi \models f \vee g \Leftrightarrow \pi \models f \text{ or } \pi \models g$$

$$\pi \models f \wedge g \Leftrightarrow \pi \models f \text{ and } \pi \models g$$

$$\pi \models \mathbf{X}f \Leftrightarrow \pi^1 \models f$$

$$\pi \models \mathbf{F}f \Leftrightarrow \text{for some } i \geq 0, \pi^i \models f$$

$$\pi \models \mathbf{G}f \Leftrightarrow \text{for all } i \geq 0, \pi^i \models f$$

$$\pi \models f \mathbf{U} g \Leftrightarrow \exists i \geq 0. \pi^i \models g \wedge \forall j < i. \pi^j \models f$$

$$\pi \models f \mathbf{R} g \Leftrightarrow \forall j \geq 0. ((\forall i < j. \pi^i \not\models f) \Rightarrow \pi^j \models g)$$

CTL* Dualities and other Identities

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

According to the semantics, we can derive several dualities:

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

Furthermore, we have

- $\mathbf{F}f \equiv \text{True} \mathbf{U} f$

So (besides $\text{True} \in AP$) we only need the operators

$$\neg, \vee, \mathbf{X}, \mathbf{U}, \mathbf{E}$$

to express all CTL* properties.

CTL and LTL

Two simpler sublogics of CTL* are defined:

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

The expressive power of LTL and CTL is incomparable.

LTL: single universal path quantifier

$$\mathcal{S} ::= \mathbf{A}\mathcal{P}$$

$$\mathcal{P} ::= AP \mid \neg \mathcal{P} \mid \mathcal{P} \wedge \mathcal{P} \mid \mathcal{P} \vee \mathcal{P} \mid$$

$$\mathbf{X}\mathcal{P} \mid \mathbf{F}\mathcal{P} \mid \mathbf{G}\mathcal{P} \mid \mathcal{P} \mathbf{U} \mathcal{P} \mid \mathcal{P} \mathbf{R} \mathcal{P}$$

Summarizing:

- The only state formulas (\mathcal{S}) are:
 - all-quantified path formulas
- Path formulas (\mathcal{P}) are:
 - atomic propositions
 - Boolean combinations of path formulas
 - temporal combinations of path formulas

Examples:

In LTL: $\mathbf{A}(\mathbf{F}\mathbf{G}p)$, $\mathbf{A}(\neg(\mathbf{G}\mathbf{F}p) \vee \mathbf{F}(q))$

Not in LTL: $\mathbf{A}\mathbf{F}\mathbf{A}\mathbf{G}p$, $\mathbf{A}\mathbf{G}(\mathbf{E}\mathbf{F}p)$

$$\mathbf{A}(\mathbf{F}\mathbf{G}p) \stackrel{?}{\equiv} \mathbf{A}\mathbf{F}\mathbf{A}\mathbf{G}p$$

CTL: formal syntax

Temporal operators are preceded by path quantifiers

$$\mathcal{S} ::= AP \mid \neg \mathcal{S} \mid \mathcal{S} \wedge \mathcal{S} \mid \mathcal{S} \vee \mathcal{S} \mid \mathbf{E} \mathcal{P} \mid \mathbf{A} \mathcal{P}$$

$$\mathcal{P} ::= \mathbf{X} \mathcal{S} \mid \mathbf{F} \mathcal{S} \mid \mathbf{G} \mathcal{S} \mid \mathcal{S} \mathbf{U} \mathcal{S} \mid \mathcal{S} \mathbf{R} \mathcal{S}$$

Summarizing:

- State formulas (\mathcal{S}):
 - atomic propositions are state formulas
 - Boolean combinations of state formulas
 - quantified path formulas are state formulas
- Path formulas (\mathcal{P}):
 - temporal combinations of state formulas

Examples in CTL: $\mathbf{AG}(\mathbf{EF} p)$, $\mathbf{E}(p \mathbf{U}(\mathbf{EX} q))$

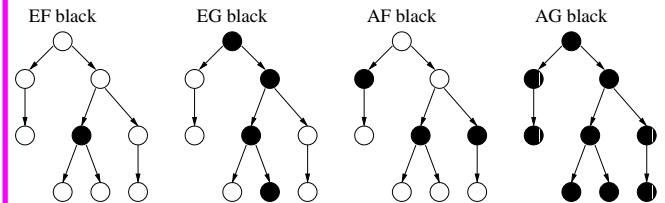
Not in CTL: $\mathbf{A}(\mathbf{F} \mathbf{G} p)$, $\mathbf{A}(\mathbf{X} \mathbf{X}(p))$, $\mathbf{E}(p \mathbf{U}(\mathbf{X} q))$

The first one is not expressible in CTL, the second is expressible in CTL: $\mathbf{AX}(\mathbf{AX} p)$

CTL: combinations of operators

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

- \mathbf{AX} and \mathbf{EX} : for all/some next state
- \mathbf{AF} : inevitably
- \mathbf{EF} : potentially
- \mathbf{AG} : invariantly
- \mathbf{EG} : potentially always
- \mathbf{AU} and \mathbf{EU} : for all/some paths, until
- \mathbf{AR} and \mathbf{ER} : for all/some paths, releases



CTL: minimal set of operators

Only the following operators are needed:

- Boolean connectives: $\neg, \vee, True \in AP$
- Temporal combinations: $\mathbf{EX}, \mathbf{EG}, \mathbf{EU}$

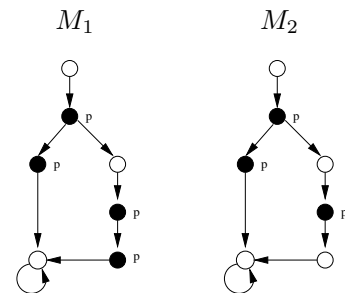
Standard transformations (in CTL*):

- $\mathbf{EF} f \equiv \mathbf{E}[True \mathbf{U} f]$
- $\mathbf{AX} f \equiv \neg \mathbf{EX}(\neg f)$
- $\mathbf{AG} f \equiv \neg \mathbf{EF}(\neg f)$
- $\mathbf{AF} f \equiv \neg \mathbf{EG}(\neg f)$
- $\mathbf{A}[f \mathbf{R} g] \equiv \neg \mathbf{E}[\neg f \mathbf{U} \neg g]$
- $\mathbf{E}[f \mathbf{R} g] \equiv \neg \mathbf{A}[\neg f \mathbf{U} \neg g]$

To remove \mathbf{AU} note that;

- $f \mathbf{R} g \equiv (g \mathbf{U} (f \wedge g)) \vee \mathbf{G} g$, and hence:
- $\mathbf{A}[f \mathbf{U} g] \equiv \neg \mathbf{E}[\neg g \mathbf{U} (\neg f \wedge \neg g)] \wedge \neg \mathbf{EG}(\neg g)$

Examples: CTL vs LTL

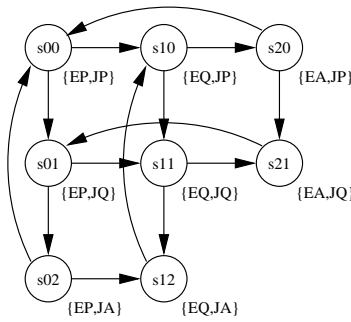


- $M_1 \models \mathbf{AF}(p \wedge \mathbf{X} p)$ but $M_1 \not\models \mathbf{AF}(p \wedge \mathbf{AX} p)$
- $M_2 \not\models \mathbf{AF}(p \wedge \mathbf{X} p)$ but $M_2 \models \mathbf{AF}(p \wedge \mathbf{EX} p)$

This shows that the LTL-formula $\mathbf{AF}(p \wedge \mathbf{X} p)$ is not equivalent to one of the CTL formulas $\mathbf{AF}(p \wedge \mathbf{AX} p)$ or $\mathbf{AF}(p \wedge \mathbf{EX} p)$.

Actually: $\mathbf{AF}(p \wedge \mathbf{X} p)$ is not expressible in CTL (but this is not a proof).

Motivation for Fairness



- Atomic Propositions: EP, EQ, EA, JP, JQ, JA .
- Intended meaning: Jan or Ella is either Playing, posing Questions, getting Answers.
- To exclude that one child gets all attention, we want that both $\neg EQ$ as well as $\neg JQ$ hold infinitely often.
- this leads to the following so-called fairness constraints:

$$F = \{\{s_{00}, s_{01}, s_{02}, s_{20}, s_{21}\}, \{s_{00}, s_{10}, s_{20}, s_{02}, s_{12}\}\}$$

Fairness constraints

Sometimes properties are violated by “unrealistic” paths only, for instance due to a scheduler. In this case, one may restrict to fair paths.

A Kripke structure over AP with fairness constraints is a structure $M = (S, R, L, F)$, where:

- as before, S is a set of states, $R \subseteq S \times S$ is a total transition relation, $L : S \rightarrow \mathcal{P}(AP)$ labels states with atomic propositions
- $F \subseteq \mathcal{P}(S)$ is a set of fairness constraints

Now a path is fair, if it hits each fairness constraint infinitely often:

$$fair(\pi) \Leftrightarrow \forall C \in F. \{i \mid \pi(i) \in C\} \text{ is infinite}$$

In CTL* with fairness semantics (\models_F), only fair paths will be considered

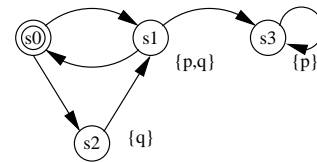
Fairness Semantics for CTL*

In CTL* with fairness semantics (\models_F), only fair paths will be considered. Given a fixed Kripke Structure with Fairness Constraints $M = (S, R, L, F)$, $s \models_F f$ means: formula f holds in state s in the fair CTL* semantics.

The definition coincides with \models except for the following three clauses:

- $s \models_F p \Leftrightarrow p \in L(s)$ and there is some fair path starting in s
- $s \models_F \mathbf{A}f \Leftrightarrow$ for all fair paths π starting in s , we have $\pi \models_F f$
- $s \models_F \mathbf{E}f \Leftrightarrow$ for some fair path π starting in s , we have $\pi \models_F f$

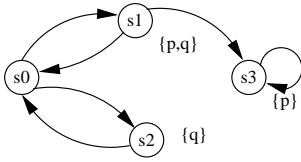
Fairness: Example



Note that $s_0 \models \mathbf{E}(\mathbf{F} \mathbf{G} p)$, but $s_0 \not\models \mathbf{A}(\mathbf{F} \mathbf{G} p)$.

- First, consider as Fairness constraint: $F = \{\{s_3\}\}$.
 - then all fair paths contain s_3 infinitely often.
 - we have: $s_0 \models_F \mathbf{A}(\mathbf{F} \mathbf{G} p)$.
- Next, consider as Fairness constraint: $F = \{\{s_2\}\}$.
 - then all fair paths contain s_2 infinitely often.
 - in particular, fair paths cannot contain s_3 .
 - So $s_0 \not\models_F \mathbf{E}(\mathbf{F} \mathbf{G} p)$.

Excercise



p , $\mathbf{E}[q \mathbf{R} p]$, $\mathbf{E}(\mathbf{F} \mathbf{G} p)$, $\mathbf{A}(\mathbf{G} \mathbf{F} p)$, $\mathbf{A} \mathbf{G} (\mathbf{E} \mathbf{F} p)$,
 $\mathbf{A} \mathbf{G} \mathbf{F} (p \wedge \mathbf{X} q)$, $\mathbf{A} \mathbf{G} (\neg q \vee \mathbf{F} p)$, $\mathbf{A}((\mathbf{G} p) \vee (\mathbf{F} q))$

1. Are these formulas in LTL and/or CTL?
2. Determine in which states of the Kripke Structure above the formulas above hold
3. Extend the Kripke structure with the fairness constraints $F = \{\{s_1\}, \{s_2\}\}$. In which states do the formulas above *fairly* hold?
4. Similar, for the fairness constraint $F = \{\{s_3\}\}$.
- 5*. Draw a Kripke structure where $\mathbf{A}(\mathbf{F} \mathbf{G} p)$ holds, but $\mathbf{A} \mathbf{F} \mathbf{A} \mathbf{G} p$ doesn't.