

# Algorithms for Model Checking (2IW55)

## Lecture 1

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

Tim Willemse

(timw@win.tue.nl)

<http://www.win.tue.nl/~timw>

HG 6.76

Motivation

Kripke Structures

Temporal Logics

CTL\*

CTL and LTL

Exercise

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**

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 conservative over-approximation

Motivation

Kripke Structures

Temporal Logics

CTL\*

CTL and LTL

Exercise

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). There are two prevailing models, which will be used interchangeably in these lectures:

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

Today: only Kripke Structures

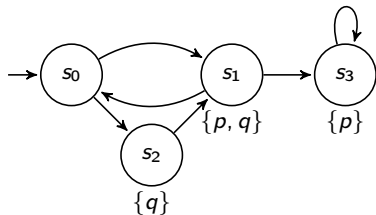
Let  $AP$  be a set of atomic propositions. A **Kripke Structure** over  $AP$  is a structure  $M = \langle S, S_0, R, L \rangle$ , where

- ▶  $S$  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  $(s, t) \in R$ .
- ▶  $L: S \rightarrow 2^{AP}$ , 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  $sRt$

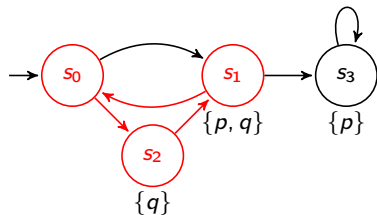
This is a Kripke Structure over  $AP$ ,  $M = \langle S, S_0, R, L \rangle$  as follows:



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

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

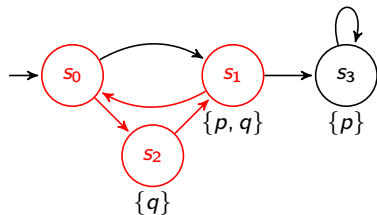




### Terminology

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

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

In the Kripke Structure above:

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

Motivation

Kripke Structures

Temporal Logics

CTL\*

CTL and LTL

Exercise

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*

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:

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

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:

$$s \models \text{true}$$

$$s \not\models \text{false}$$

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

$$s \models \neg f \quad \text{iff} \quad s \not\models f$$

$$s \models f \wedge g \quad \text{iff} \quad s \models f \text{ and } s \models g$$

$$s \models f \vee g \quad \text{iff} \quad s \models f \text{ or } s \models g$$

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

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

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:

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



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:

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

According to the semantics, we can derive several dualities:

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

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

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 ( $\mathcal{P}$ ):

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

$$\mathcal{P} ::= \text{true} \mid \text{false} \mid A\mathcal{P} \mid \neg\mathcal{P} \mid \mathcal{P} \wedge \mathcal{P} \mid \mathcal{P} \vee \mathcal{P} \\ \mid X\mathcal{P} \mid F\mathcal{P} \mid G\mathcal{P} \mid [\mathcal{P} U \mathcal{P}] \mid [\mathcal{P} R \mathcal{P}]$$

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:  $A F G p$ ,  $A (\neg(G F p) \vee F q)$ ; not in LTL:  $A F A G p$ ,  $A G E F p$

Question:  $A F G p \stackrel{?}{\equiv} A F A G p$

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

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

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:**  $A G E F p$ ,  $E [p U (E X q)]$ ;

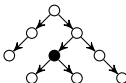
**not in CTL:**  $A F G p$ ,  $A X X p$ ,  $E [p U (X q)]$

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

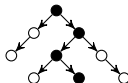
**Alternative view:** CTL has only state formulae, with the following ten temporal combinators:

- ▶  $A X$  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

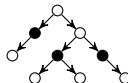
E F black



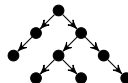
E G black



A F black



A G black



For CTL, only the following operators are needed:

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

Standard transformations (derived from CTL\*):

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

To remove  $A [ U ]$ , note that:

1.  $[f R g] \equiv [g U (f \wedge g)] \vee G g$
2.  $A [f U g] \equiv \neg E [(\neg f) R (\neg g)]$
3.  $E (f \vee g) \equiv E f \vee E g$

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

## Example (CTL versus LTL)



- ▶  $M_1 \models \text{A F } (p \wedge \text{X } p)$  but  $M_1 \not\models \text{A F } (p \wedge \text{A X } p)$
- ▶  $M_2 \not\models \text{A F } (p \wedge \text{X } p)$  but  $M_2 \models \text{A F } (p \wedge \text{E X } p)$

This shows that the LTL-formula  $\text{A F } (p \wedge \text{X } p)$  is **not equivalent** to one of the CTL formulae  $\text{A F } (p \wedge \text{A X } p)$  or  $\text{A F } (p \wedge \text{E X } p)$ .

Actually:  $\text{A F } (p \wedge \text{X } p)$  is **not expressible** in CTL (does **not** follow from these observations)

Motivation

Kripke Structures

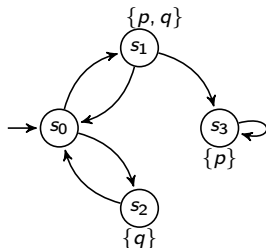
Temporal Logics

CTL\*

CTL and LTL

Exercise





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

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