Computation Tree Logic

Lecture #17 of Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling & Verification

E-mail: `katoen@cs.rwth-aachen.de`

June 06, 2007

Overview Lecture #17

⇒ Summary of LTL model checking

- Branching temporal logic
- Syntax and semantics of CTL

Summary of LTL model checking (1)

- LTL is a logic for formalizing path-based properties
- Expansion law allows for rewriting until into local conditions and next
- LTL-formula φ can be transformed algorithmically into NBA $\mathcal{A}_\varphi$
  - this may cause an exponential blow up
  - algorithm: first construct a GNBA for φ; then transform it into an equivalent NBA
- LTL-formulae describe ω-regular LT-properties
  - but do not have the same expressivity as ω-regular languages

Summary of LTL model checking (2)

- $TS \models \varphi$ can be solved by a nested depth-first search in $TS \otimes \mathcal{A}_{\neg\varphi}$
  - time complexity of the LTL model-checking algorithm is linear in TS and exponential in $|\varphi|$
- Fairness assumptions can be described by LTL-formulae

  the model-checking problem for LTL with fairness is reducible
  to the standard LTL model-checking problem

- The LTL-model checking problem is PSPACE-complete
- Satisfiability and validity of LTL amounts to NBA emptiness-check
- The satisfiability and valditiy problem for LTL are PSPACE-complete

Overview Lecture #17

- Summary of LTL model checking

$\Rightarrow$ Branching temporal logic

- Syntax and semantics of CTL

Linear and branching temporal logic

- *Linear* temporal logic:

  "statements about (all) paths starting in a state"

  - $s \models \Box(x \leqslant 20)$ iff for all possible paths starting in s always $x \leqslant 20$

- *Branching* temporal logic:

  "statements about all or some paths starting in a state"

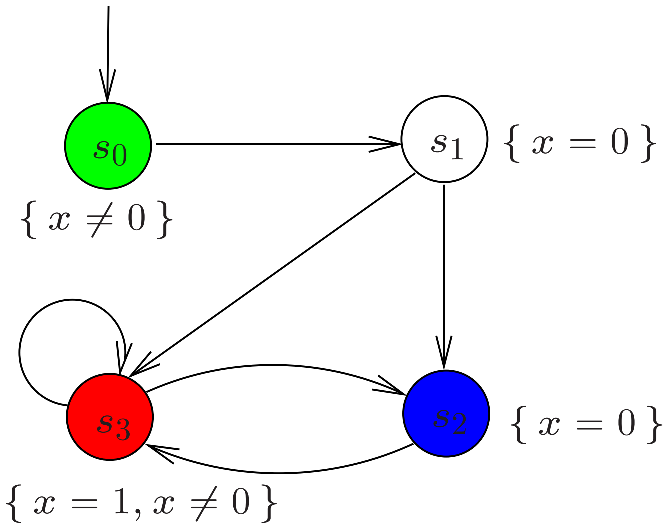
  - $s \models \forall\Box(x \leqslant 20)$ iff for **all** paths starting in s always $x \leqslant 20$
  - $s \models \exists\Box(x \leqslant 20)$ iff for **some** path starting in s always $x \leqslant 20$
  - nesting of path quantifiers is allowed

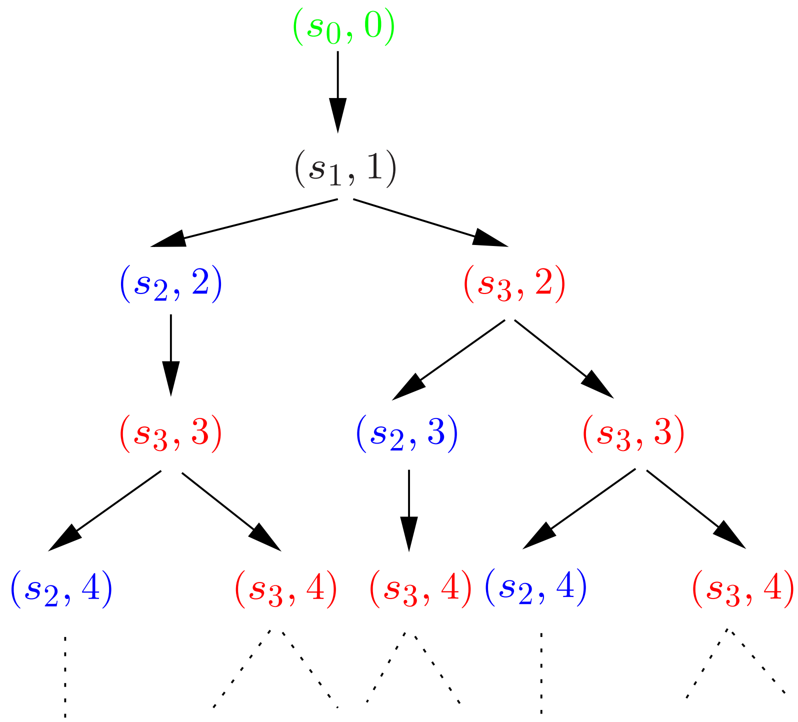
- Checking $\exists\varphi$ in LTL can be done using $\forall\neg\varphi$
  - . . . but this does not work for nested formulas such as $\forall\Box\exists\Diamond a$

Linear versus branching temporal logic

- Semantics is based on a branching notion of time
  - an infinite tree of states obtained by unfolding transition system
  - one “time instant” may have several possible successor “time instants”
- Incomparable expressiveness
  - there are properties that can be expressed in LTL, but not in CTL
  - there are properties that can be expressed in most branching, but not in LTL
- Distinct model-checking algorithms, and their time complexities
- Distinct treatment of fairness assumptions
- Distinct equivalences (pre-orders) on transition systems
  - that correspond to logical equivalence in LTL and branching temporal logics

Transition systems and trees





"behavior" in a state s	path-based: *trace*(s)	state-based: computation tree of s
temporal logic	LTL: path formulas φ $s \models \varphi$ iff $\forall \pi \in \mathit{Paths}(s).\, \pi \models \varphi$	CTL: state formulas existential path quantification $\exists\varphi$ universal path quantification: $\forall\varphi$
complexity of the model checking problems	PSPACE–complete $\mathcal{O}\left(\lvert TS\rvert \cdot 2^{\lvert\varphi\rvert}\right)$	PTIME $\mathcal{O}\left(\lvert TS\rvert \cdot \lvert\Phi\rvert\right)$
implementation-relation	trace inclusion and the like (proof is PSPACE-complete)	simulation and bisimulation (proof in polynomial time)
fairness	no special techniques	special techniques needed

Branching temporal logics

There are various branching temporal logics:

- Hennessy-Milner logic
- Computation Tree Logic (CTL)
- Extended Computation Tree Logic (CTL*)
  - combines LTL and CTL into a single framework
- Alternation-free modal μ-calculus
- Modal μ-calculus
- Propositional dynamic logic

Overview Lecture #17

- Summary of LTL model checking
- Branching temporal logic

$\Rightarrow$ Syntax and semantics of CTL

Computation tree logic

modal logic over infinite trees [Clarke & Emerson 1981]

- Statements over states
  - $a \in AP$ — atomic proposition
  - $\neg \Phi$ and $\Phi \wedge \Psi$ — negation and conjunction
  - $\exists \varphi$ — there *exists* a path fulfilling φ
  - $\forall \varphi$ — *all* paths fulfill φ
- Statements over paths
  - $\bigcirc \Phi$ — the next state fulfills Φ
  - $\Phi \, \mathsf{U} \, \Psi$ — Φ holds until a Ψ-state is reached

⇒ note that $\bigcirc$ and U *alternate* with $\forall$ and $\exists$

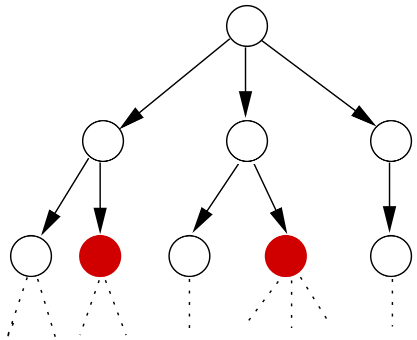
- $\forall \bigcirc \bigcirc \Phi$ and $\forall \exists \bigcirc \Phi \notin$ CTL, but $\forall \bigcirc \forall \bigcirc \Phi$ and $\forall \bigcirc \exists \bigcirc \Phi \in$ CTL

Derived operators

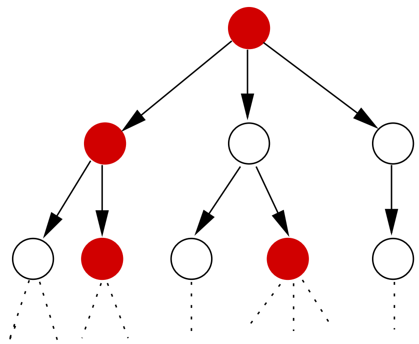
potentially Φ:	$\exists\Diamond\Phi$	$=$	$\exists(\mathsf{true}\,\mathsf{U}\,\Phi)$
inevitably Φ:	$\forall\Diamond\Phi$	$=$	$\forall(\mathsf{true}\,\mathsf{U}\,\Phi)$
potentially always Φ:	$\exists\Box\Phi$	$:=$	$\neg\forall\Diamond\neg\Phi$
invariantly Φ:	$\forall\Box\Phi$	$=$	$\neg\exists\Diamond\neg\Phi$
weak until:	$\exists(\Phi\,\mathsf{W}\,\Psi)$	$=$	$\neg\forall\big((\Phi\wedge\neg\Psi)\,\mathsf{U}\,(\neg\Phi\wedge\neg\Psi)\big)$
	$\forall(\Phi\,\mathsf{W}\,\Psi)$	$=$	$\neg\exists\big((\Phi\wedge\neg\Psi)\,\mathsf{U}\,(\neg\Phi\wedge\neg\Psi)\big)$

the boolean connectives are derived as usual

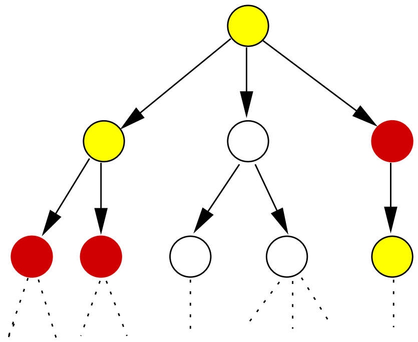
Visualization of semantics



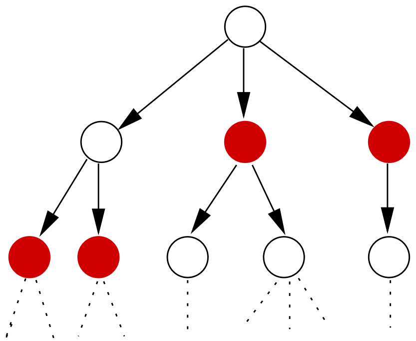
$\exists\Diamond \mathit{red}$



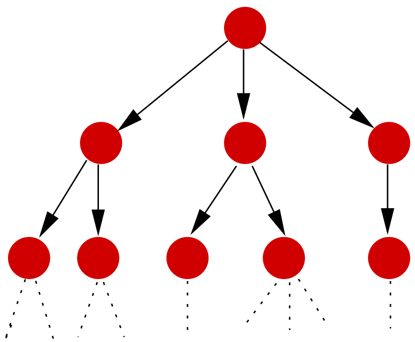
$\exists\Box \mathit{red}$



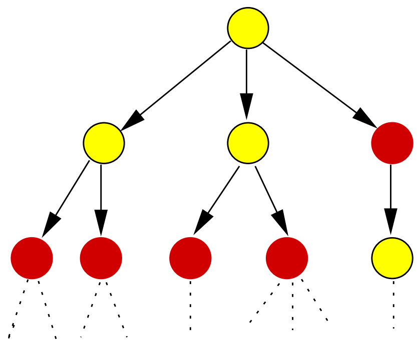
$\exists(\mathit{yellow}\ \mathsf{U}\ \mathit{red})$



$\forall\Diamond \mathit{red}$



$\forall\Box \mathit{red}$



$\forall(\mathit{yellow}\ \mathsf{U}\ \mathit{red})$

Example properties in CTL

Semantics of CTL state-formulas

Defined by a relation $\models$ such that

$s \models \Phi$ if and only if formula Φ holds in state s

$$
\begin{array}{lll}
s \models a & \text{iff} & a \in L(s) \\
s \models \neg\Phi & \text{iff} & \neg(s \models \Phi) \\
s \models \Phi \wedge \Psi & \text{iff} & (s \models \Phi) \wedge (s \models \Psi) \\
s \models \exists\varphi & \text{iff} & \pi \models \varphi \text{ for } \textit{some} \text{ path } \pi \text{ that starts in } s \\
s \models \forall\varphi & \text{iff} & \pi \models \varphi \text{ for } \textit{all} \text{ paths } \pi \text{ that start in } s
\end{array}
$$

Semantics of CTL path-formulas

Define a relation $\models$ such that

$\pi \models \varphi$ if and only if path π satisfies φ

$$\pi \models \bigcirc\Phi \quad \text{iff } \pi[1] \models \Phi$$

$$\pi \models \Phi \,\mathsf{U}\, \Psi \quad \text{iff } (\exists\, j \geqslant 0.\, \pi[j] \models \Psi \ \wedge \ (\forall\, 0 \leqslant k < j.\, \pi[k] \models \Phi))$$

where $\pi[i]$ denotes the state s_i in the path π

Transition system semantics

- For CTL-state-formula Φ, the *satisfaction set* $\mathit{Sat}(\Phi)$ is defined by:

$$\mathit{Sat}(\Phi) \;=\; \{\, s \in S \mid s \models \Phi \,\}$$

- TS satisfies CTL-formula Φ iff Φ holds in all its initial states:

$$\mathit{TS} \models \Phi \quad \text{if and only if} \quad \forall s_0 \in I \,.\, s_0 \models \Phi$$

  - this is equivalent to $I \subseteq \mathit{Sat}(\Phi)$

- Point of attention: $\mathit{TS} \not\models \Phi$ and $\mathit{TS} \not\models \neg\Phi$ is possible!
  - because of several initial states, e.g. $s_0 \models \exists\Box\Phi$ and $s_0' \not\models \exists\Box\Phi$

A triple modular redundant system

Infinitely often

$s \models \forall\Box\forall\Diamond a$ if and only if $\forall\pi \in$ *Paths*(s) an a-state is visited infinitely often