Model Checking

Lecture #12+#13: Branching Time Versus Linear Time

[Baier & Katoen, Chapter 6.3, 7.1+7.2]

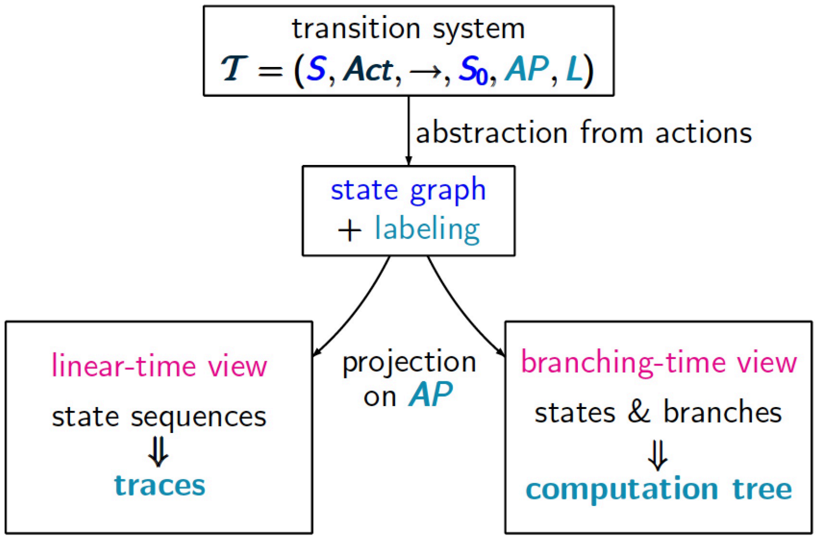
Joost-Pieter Katoen

Software Modeling and Verification Group

Model Checking Course, RWTH Aachen, WiSe 2019/2020

Overview

Topic



Linear Versus Branching Time

	linear time	branching time
behavior	path based traces	state based computation tree
temporal logic	**LTL** path formulas	**CTL** state formulas
model checking	PSPACE-complete $\mathcal{O}(size(\mathcal{T}) \cdot \exp(\lvert\varphi\rvert))$	PTIME $\mathcal{O}(size(\mathcal{T}) \cdot \lvert\Phi\rvert)$
impl. relation	trace inclusion trace equivalence PSPACE-complete	simulation bisimulation PTIME

Overview

LTL and CTL are Incomparable

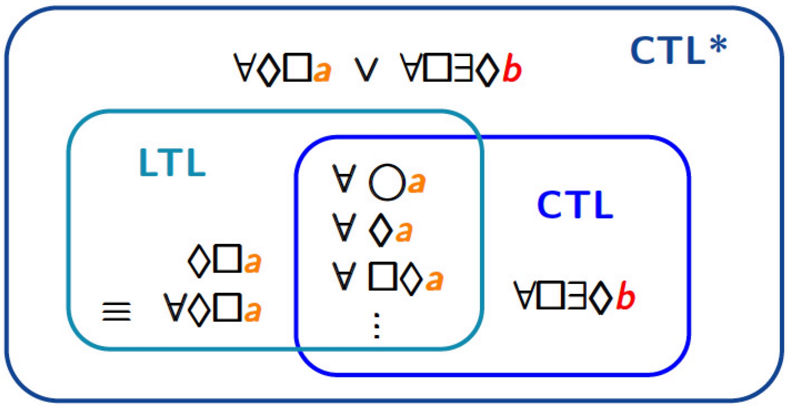
- Some LTL-formulas cannot be expressed in CTL, e.g.,
  - $\Diamond \Box\, a$
  - $\Diamond\, (a \wedge \bigcirc\, a)$

  There does not exist an equivalent CTL formula

- Some CTL-formulas cannot be expressed in LTL, e.g.,
  - $\forall \Diamond\, \forall \Box\, a$
  - $\forall \Diamond\, (a \wedge \forall \bigcirc\, a)$, and
  - $\forall \Box\, \exists \Diamond\, a$

  There does not exist an equivalent LTL formula

Relating LTL, CTL, and CTL*



Overview

CTL vs. LTL Model Checking

LTL model checking is PSPACE-complete
CTL model checking is PTIME-complete.

Take a property that can be expressed in both LTL and CTL

Is CTL model checking more efficient? No!

LTL-formulae can be exponentially shorter than their CTL-equivalent

CTL Versus LTL

If Φ is equivalent to some LTL-formula φ then:

$\Phi \equiv \varphi$ where φ is obtained by removing all path quantifiers from Φ.
In particular, $|\varphi| \leq |\Phi|$.

If P ≠ NP, then there is a sequence φ_n, $n \geq 0$ of LTL formulas such that:

- $|\varphi_n|$ is polynomial in n
- φ_n has an equivalent CTL formula
- no CTL formula of polynomial length is equivalent to φ_n

Proof.

φ_n = the absence of a Hamiltonian path in a digraph on n vertices □

LTL Encoding the Hamiltonian Path Problem

CTL Encoding the Hamiltonian Path Problem

All $n!$ possibilities need to be explicitly enumerated

Suppose there is a CTL-formula of polynomial length equivalent to φ_n.
Then: as CTL model-checking is $\in P$,
the Hamiltonian path problem $\in P$, and $P = NP$.

Satisfiability Problem

The LTL satisfiability problem is PSPACE-complete.

The LTL satisfiability problem is equally hard as
the LTL model checking problem.

- The CTL satisfiability problem is EXPTIME-complete.
- The CTL* satisfiability problem is 2EXPTIME-complete.

The CTL satisfiability problem is harder than
the CTL model checking problem.
This also applies to CTL* (and many more logics)

Overview

Trace Equivalence

Definition: trace equivalence

Transition systems TS and TS' (both over AP) are trace equivalent iff they exhibit the same traces:

$$TS \equiv_{trace} TS' \quad \text{if and only if} \quad Traces(TS) = Traces(TS').$$

Examples

Trace Equivalence and LT Properties

$TS \equiv_{trace} TS'$ if and only if TS and TS' satisfy the same LT properties:

$$TS \equiv_{trace} TS' \quad \text{if and only if} \quad \left(\forall E \subseteq \left(2^{AP}\right)^{\omega}.\ TS \models E \text{ iff } TS' \models E\right).$$

Logical Equivalence

Definition: logical equivalence

For transition systems TS and TS' (both over AP):

- $TS \equiv_{LTL} TS'$ iff $\big(\forall \varphi \in \text{LTL}.\ TS \models \varphi \text{ iff } TS' \models \varphi\big)$
- $TS \equiv_{CTL} TS'$ iff $\big(\forall \Phi \in \text{CTL}.\ TS \models \Phi \text{ iff } TS' \models \Phi\big)$

In a similar way, $\equiv_L$ can be defined for logic L (such as CTL* etc.).

$TS \equiv_{trace} TS'$ if and only if $TS \equiv_{LTL} TS'$

Bisimulation

Definition: bisimulation relation

Let $TS_i = (S_i, Act_i, \rightarrow_i, I_i, AP, L_i)$, i=1, 2, be transition systems. The symmetric relation $\mathfrak{R} \subseteq S_1 \times S_2$ is a bisimulation for (TS_1, TS_2) whenever:

1. for all initial states $s_1 \in I_1.\ (s_1, s_2) \in \mathfrak{R}$ for some $s_2 \in I_2$
2. for all states $(s_1, s_2) \in \mathfrak{R}$ it holds:

   2.1 $L_1(s_1) = L_2(s_2)$, and

   2.2 $s_1' \in Post(s_1)$ implies $(s_1', s_2') \in \mathfrak{R}$ for some $s_2' \in Post(s_2)$.

Visually

$$\begin{array}{ccc} s_1 & \rightarrow & s_1' \\ \mathfrak{R} & & \\ s_2 & & \end{array} \quad \text{can be completed to} \quad \begin{array}{ccc} s_1 & \rightarrow & s_1' \\ \mathfrak{R} & & \color{red}{\mathfrak{R}} \\ s_2 & \color{red}{\rightarrow} & \color{red}{s_2'} \end{array}$$

and by symmetry

$$\begin{array}{ccc} s_1 & & \\ \mathfrak{R} & & \\ s_2 & \rightarrow & s_2' \end{array} \quad \text{can be completed to} \quad \begin{array}{ccc} s_1 & \color{red}{\rightarrow} & \color{red}{s_1'} \\ \mathfrak{R} & & \color{red}{\mathfrak{R}} \\ s_2 & \rightarrow & s_2' \end{array}$$

Example

Bisimulation on Paths

Whenever we have:

$$\begin{array}{ccccccccc} s_0 & \rightarrow & s_1 & \rightarrow & s_2 & \rightarrow & s_3 & \rightarrow & s_4 \ldots\ldots \\ \mathfrak{R} & & & & & & & & \\ t_0 & & & & & & & & \end{array}$$

this can be completed to

$$\begin{array}{ccccccccc} s_0 & \rightarrow & s_1 & \rightarrow & s_2 & \rightarrow & s_3 & \rightarrow & s_4 \ldots\ldots \\ \mathfrak{R} & & \mathfrak{R} & & \mathfrak{R} & & \mathfrak{R} & & \mathfrak{R} \\ t_0 & \rightarrow & t_1 & \rightarrow & t_2 & \rightarrow & t_3 & \rightarrow & t_4 \ldots\ldots \end{array}$$

Proof.

By induction on index i of state s_i. □

Bisimulation Equivalence

Definition: bisimulation equivalence

TS_1 and TS_2 are bisimulation equivalent (short: bisimilar), denoted $TS_1 \sim TS_2$, if there exists a bisimulation for (TS_1, TS_2). That is:

$$\sim \;=\; \bigcup \left\{ \mathfrak{R} \mid \mathfrak{R} \text{ is a bisimulation on } (TS_1, TS_2) \right\}.$$

Bisimilarity ($\sim$) is an equivalence relation.

Proof.

- (Reflexivity). The identity relation is a bisimulation for (TS, TS).
- (Symmetry). If $\mathfrak{R}$ is a bisimulation for (TS, TS'), then $\mathfrak{R}^{-1}$ is a bisimulation for (TS', TS).
- (Transitivity). If $\mathfrak{R}_{1,2}$ is a bisimulation for (TS_1, TS_2) and $\mathfrak{R}_{2,3}$ a bisimulation for (TS_2, TS_3), then $\mathfrak{R}_{2,3} \circ \mathfrak{R}_{1,2}$ is a bisimulation for (TS_1, TS_3). □

Bisimilarity And Trace Equivalence

1. $TS_1 \sim TS_2$ implies $TS_1 \equiv_{trace} TS_2$.
2. For AP-deterministic[1] TS_1, TS_2: $TS_1 \sim TS_2$ iff $TS_1 \equiv_{trace} TS_2$.

Proof.

1. Follows from the fact that bisimulation carries over to infinite paths.
2. Left as exercise. □

$TS_1 \sim TS_2$ implies $TS_1 \equiv_{LTL} TS_2$. The converse also holds for AP-deterministic transition systems

[1] Transition system TS is AP-deterministic whenever it has at most one initial state and $|Post(s) \cap \{ s' \in S \mid L(s') = A \}| \leq 1$.

Distinguishing Bisimilarity And Trace Equivalence

Bisimulation on States

Definition: bisimulation/bisimilarity on states

Symmetric relation $\mathfrak{R} \subseteq S \times S$ is a bisimulation on TS (with state space S) if for any $(s_1, s_2) \in \mathfrak{R}$:

1. $L(s_1) = L(s_2)$
2. $s_1' \in Post(s_1)$ then $(s_1', s_2') \in \mathfrak{R}$ for some $s_2' \in Post(s_2)$.

The states s_1 and s_2 are bisimilar, denoted $s_1 \sim_{TS} s_2$, if $(s_1, s_2) \in \mathfrak{R}$ for some bisimulation $\mathfrak{R}$ for TS.

$s_1 \sim_{TS} s_2$ if and only if $TS_{s_1} \sim TS_{s_2}$ where TS_{s_i} denotes the transition system TS in which s_i is the only initial state.

Bisimilarity And CTL

Theorem: Bisimilarity, CTL and CTL*

Let TS be a finitely branching[2] transition system and s, s' states in TS. The following statements are equivalent:

1. $s \sim_{TS} s'$
2. s and s' are CTL-equivalent, i.e., $s \equiv_{CTL} s'$
3. s and s' are CTL*-equivalent, i.e., $s \equiv_{CTL^*} s'$.

Proof.

This is proven in three steps: $\equiv_{CTL} \subseteq \sim \subseteq \equiv_{CTL^*} \subseteq \equiv_{CTL}$. The last step is trivial, since CTL* is more expressive than CTL. □

[2] This means that every state has only finitely many direct successors. This theorem does not hold for arbitrary infinite-state transition systems.

Proof (1)

Proof (2)

Example

Infinite-Branching

For any transition systems TS and TS' (over AP):

$$TS \sim TS' \quad \text{iff} \quad TS \equiv_{CTL} TS' \quad \text{iff} \quad TS \equiv_{CTL^*} TS'.$$

Definition: CTL$^-$

CTL$^-$ state-formulas with $a \in AP$ obey the grammar:

$$\Phi ::= \text{true} \;\Big|\; a \;\Big|\; \Phi_1 \wedge \Phi_2 \;\Big|\; \neg\Phi \;\Big|\; \exists\bigcirc\Phi \;\Big|\; \forall\bigcirc\Phi$$

No until-modalities, so no $\Box$ and no $\Diamond$

1. CTL$^-$ is strictly less expressive than CTL (and than CTL*).
2. CTL$^-$ equivalence coincides with CTL (and CTL*) equivalence.

Proof.

Follows from the fact that in the proof of equivalence of $\sim$, $\equiv_{CTL}$ and $\equiv_{CTL^*}$ only CTL$^-$-formulas are used. In particular, no until-modalities are used. $\square$

The Importance of These Results

- CTL$^-$-, CTL- and CTL*-equivalence coincide
  - despite the fact that these logics have different expressivity
- Bisimilar transition systems preserve the same CTL* formulas
  - and thus the same LTL formulas (and LT properties)
- Non-bisimilarity can be shown by a single CTL$^-$ formula Φ
  - $TS_1 \models \Phi$ and $TS_2 \not\models \Phi$ implies $TS_1 \not\sim TS_2$
- One does not even need to use an until-modality

On Complexity

The decision problem whether two finite transition systems are trace equivalent is PSPACE-complete.

Proof.

Reduction from language equivalence of finite-state automata. □

The decision problem whether two finite transition systems are bisimilar is PTIME-complete.

Proof.

A polynomial-time algorithm will be dealt with in an upcoming lecture. PTIME-hardness is outside the scope of this lecture. □

Overview

Syntax of CTL*

Definition: Syntax CTL*

- CTL* state-formulas with $a \in AP$ obey the grammar:

$$\Phi ::= \text{true} \;\Big|\; a \;\Big|\; \Phi_1 \wedge \Phi_2 \;\Big|\; \neg\Phi \;\Big|\; \exists\varphi$$

- and φ is a CTL* path-formula formed by the grammar:

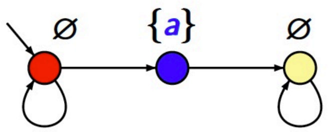
$$\varphi ::= \Phi \;\Big|\; \varphi_1 \wedge \varphi_2 \;\Big|\; \neg\varphi \;\Big|\; \bigcirc\varphi \;\Big|\; \varphi_1 \,\mathsf{U}\, \varphi_2$$

where Φ is a CTL* state-formula, and φ, φ_1 and φ_2 are path-formulas.

in CTL*: $\forall\varphi \;=\; \neg\exists\neg\varphi$. This does not hold in CTL.

Example

$\mathcal{T}$:



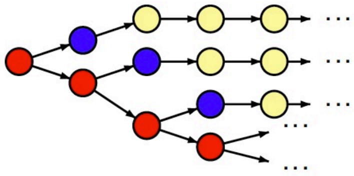
$\boxed{\exists\Box\exists\Diamond a \not\equiv \exists\Box\Diamond a}$

$\mathcal{T} \not\models \exists\Box\Diamond a$

$\mathcal{T} \models \exists\Box\exists\Diamond a$ note: $Sat(\exists\Diamond a) = \{ \bullet, \bullet \}$

hence: $\bullet\bullet\bullet\ldots \models \Box\exists\Diamond a$

computation tree:



Embedding LTL

For LTL formula φ and TS without terminal states (both over AP) and for each $s \in S$:

$$\underbrace{s \models \varphi}_{\text{LTL semantics}} \quad \text{if and only if} \quad \underbrace{s \models \forall\varphi}_{\text{CTL}^* \text{ semantics}}$$

In particular:

$$TS \models_{LTL} \varphi \quad \text{if and only if} \quad TS \models_{CTL*} \forall\varphi$$

CTL* Model Checking

[Emerson & Lei, 1985]

- Adopt a recursive descent over the parse tree of Φ (as for CTL)
- Replace maximal proper state sub-formula Ψ by new proposition a_Ψ
  - adjust labeling such that $a_\Psi \in L(s)$ if and only if $s \in Sat(\Psi)$
- In the end, this yields an LTL formula
- Most interesting case: formulas of the form $\exists\varphi$

$$s \models_{CTL^*} \exists\varphi \quad \text{iff} \quad \underbrace{s \not\models_{CTL^*} \forall\neg\varphi}_{\text{CTL}^* \text{ semantics}} \quad \text{iff} \quad \underbrace{s \not\models_{LTL} \neg\varphi}_{\text{LTL semantics}}$$

$$Sat_{CTL*}(\exists\varphi) = S \setminus Sat_{LTL}(\neg\varphi) = S \setminus \{ s \in S \mid s \models_{LTL} \neg\varphi \}$$

Example

Complexity

The CTL* model-checking algorithm for finite transition system *TS* and CTL*-formula Φ has a time complexity in $O(|TS| \cdot 2^{|\Phi|})$.

Proof.

The recursive descent is linear in $|\Phi|$. The most expensive procedure for a node, i.e., sub-formula $\Psi = \exists \varphi$ of Φ, in the parse tree is in $O(|TS| \cdot 2^{|\Psi|})$. □

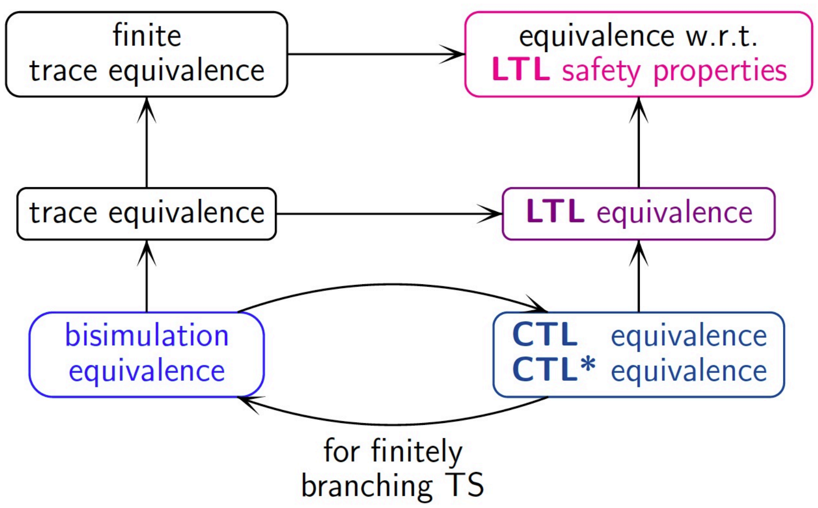
The CTL* model-checking problem is PSPACE-complete.

Proof.

Outside the scope of this lecture series. □

Overview

Summary: Equivalences



Complexity Overview

	CTL	LTL	CTL*
model checking	PTIME	PSPACE	PSPACE
algorithmic complexity	$\lvert TS\rvert \cdot \lvert\Phi\rvert$	$\lvert TS\rvert \cdot \exp(\lvert\varphi\rvert)$	$\lvert TS\rvert \cdot \exp(\lvert\Phi\rvert)$
satisfiability	EXPTIME	PSPACE	2EXPTIME
equivalence	bisimilarity	trace equivalence	bisimilarity
equivalence checking	PTIME	PSPACE	PTIME

All theoretical complexity indications are complete.

Next Lecture

Thursday December 5, 10:30