Verification

Lecture 17

Bernd Finkbeiner

UNIVERSITÄT
DES
SAARLANDES

Plan for today

- CTL*
- Bisimulation

REVIEW: LTL and CTL are incomparable

- Some LTL-formulas cannot be expressed in CTL, e.g.,
  - $\mathrm{F\,G}\, a$
  - $\mathrm{F}\,(a \wedge \mathrm{X}\, a)$
- Some CTL-formulas cannot be expressed in LTL, e.g.,
  - $\mathrm{AF\,AG}\, a$
  - $\mathrm{AF}\,(a \wedge \mathrm{AX}\, a)$
  - $\mathrm{AG\,EF}\, a$

⇒ Cannot be expressed = there does not exist an equivalent formula

Syntax of CTL*

CTL* state-formulas are formed according to:

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

where $a \in AP$ and φ is a path-formula

CTL* path-formulas are formed according to the grammar:

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

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

in CTL*: $\mathsf{A}\,\varphi = \neg\mathsf{E}\,\neg\varphi$.

CTL* semantics

$s \vDash a$	iff	$a \in L(s)$
$s \vDash \neg \Phi$	iff	not $s \vDash \Phi$
$s \vDash \Phi \wedge \Psi$	iff	$(s \vDash \Phi)$ and $(s \vDash \Psi)$
$s \vDash \mathrm{E}\,\varphi$	iff	$\pi \vDash \varphi$ for some $\pi \in Paths(s)$

$\pi \vDash \Phi$	iff	$\pi[0] \vDash \Phi$
$\pi \vDash \varphi_1 \wedge \varphi_2$	iff	$\pi \vDash \varphi_1$ and $\pi \vDash \varphi_2$
$\pi \vDash \neg\varphi$	iff	not $\pi \vDash \varphi$
$\pi \vDash \mathrm{X}\,\Phi$	iff	$\pi[1..] \vDash \Phi$
$\pi \vDash \Phi \,\mathrm{U}\, \Psi$	iff	$\exists j \geq 0.\ (\pi[j..] \vDash \Psi \ \wedge\ (\forall\, 0 \leq k < j.\ \pi[k..] \vDash \Phi))$

Transition system semantics

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

$$Sat(\Phi) = \{ q \in S \mid q \models \Phi \}$$

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

$$TS \models \Phi \quad \text{if and only if} \quad \forall q \in I.\, q_0 \models \Phi$$

this is exactly as for CTL

Embedding of LTL in CTL*

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

$$\underbrace{q \models \varphi}_{\text{LTL semantics}} \quad \text{if and only if} \quad \underbrace{q \models \mathsf{A}\,\varphi}_{\text{CTL}^* \text{ semantics}}$$

In particular:

$$TS \models_{LTL} \varphi \quad \text{if and only if} \quad TS \models_{CTL*} \mathsf{A}\,\varphi$$

CTL* is more expressive than LTL and CTL

For the CTL*-formula over $AP = \{ a, b \}$:

$$\Phi = (\mathrm{AF\,G}\, a) \vee (\mathrm{AG\,EF}\, b)$$

there does not exist any equivalent LTL or CTL formula

CTL$^+$

CTL$^+$ state-formulas are formed according to:

$$\Phi ::= \text{true} \;\big|\; a \;\big|\; \Phi_1 \wedge \Phi_2 \;\big|\; \neg\Phi \;\big|\; \mathsf{E}\,\varphi \;\big|\; \mathsf{A}\,\varphi$$

where $a \in AP$ and φ is a path-formula

CTL$^+$ path-formulas are formed according to the grammar:

$$\varphi ::= \varphi_1 \wedge \varphi_2 \;\big|\; \neg\varphi \;\big|\; \mathsf{X}\,\Phi \;\big|\; \Phi_1\,\mathsf{U}\,\Phi_2$$

where Φ, Φ_1, Φ_2 are state-formulas, and φ, φ_1 and φ_2 are path-formulas

CTL^+ is as expressive as CTL

For example:

$$\underbrace{E(Fa \wedge Fb)}_{CTL^+ \text{ formula}} \equiv \underbrace{EF(a \wedge EFb) \vee EF(b \wedge EFa)}_{CTL \text{ formula}}$$

Some rules for transforming CTL^+ formulas into equivalent CTL formulas:

$$\begin{aligned}
E\big(\neg(\Phi_1 \, U \, \Phi_2)\big) &\equiv E\big((\Phi_1 \wedge \neg\Phi_2) \, U \, (\neg\Phi_1 \wedge \neg\Phi_2) \big) \vee EG\neg\Phi_2 \\
E\big(X\Phi_1 \wedge X\Phi_2\big) &\equiv EX(\Phi_1 \wedge \Phi_2) \\
E\big(X\Phi \wedge (\Phi_1 \, U \, \Phi_2)\big) &\equiv \big(\Phi_2 \wedge EX\Phi\big) \vee \big(\Phi_1 \wedge EX(\Phi \wedge E(\Phi_1 \, U \, \Phi_2))\big) \\
E\big((\Phi_1 \, U \, \Phi_2) \wedge (\Psi_1 \, U \, \Psi_2)\big) &\equiv E\big((\Phi_1 \wedge \Psi_1) \, U \, (\Phi_2 \wedge E(\Psi_1 \, U \, \Psi_2))\big) \vee \\
& E\big((\Phi_1 \wedge \Psi_1) \, U \, (\Psi_2 \wedge E(\Phi_1 \, U \, \Phi_2))\big) \\
&\vdots
\end{aligned}$$

adding boolean combinations of path formulas to CTL does not change its expressiveness

but CTL^+ formulas can be much shorter than shortest equivalent CTL formulas

CTL* model checking

- Adopt the same bottom-up procedure as for (fair) CTL
- Replace each maximal proper state subformula Ψ by new proposition a_Ψ
  - $a_\Psi \in L(s)$ if and only if $s \in Sat(\Psi)$
- Most interesting case: formulas of the form $\mathsf{E}\,\varphi$
  - by replacing all maximal state sub-formulas in φ, an LTL-formula results!
- $q \models \mathsf{E}\,\varphi$ iff $\underbrace{q \not\models \mathsf{A}\,\neg\varphi}_{\text{CTL}^* \text{ semantics}}$ iff $\underbrace{q \not\models \neg\varphi}_{\text{LTL semantics}}$
  - $Sat_{CTL*}(\mathsf{E}\,\varphi) = S \setminus Sat_{LTL}(\neg\varphi)$

CTL* model-checking algorithm

for all $i \le |\Phi|$ **do**
  for all $\Psi \in Sub(\Phi)$ with $|\Psi| = i$ **do**
    switch(Ψ):

true	:	$Sat(\Psi) := S$;
a	:	$Sat(\Psi) := \{ q \in S \mid a \in L(q) \}$;
$a_1 \wedge a_2$	:	$Sat(\Psi) := Sat(a_1) \cap Sat(a_2)$;
$\neg a$	:	$Sat(\Psi) := S \setminus Sat(a)$;
E φ	:	determine $Sat_{LTL}(\neg\varphi)$ by means of an LTL model checker;
	:	$Sat(\Psi) := S \setminus Sat_{LTL}(\neg\varphi)$

    end switch
    $AP := AP \cup \{ a_\Psi \}$; {introduce fresh atomic proposition}
    replace Ψ with a_Ψ
    forall $q \in Sat(\Psi)$ **do** $L(q) := L(q) \cup \{ a_\Psi \}$; **od**
  end for
end for
return $I \subseteq Sat(\Phi)$

Time complexity

For transition system TS with N states and M transitions,
CTL* formula Φ, the CTL* model-checking problem $TS \models \Phi$
can be determined in time $\mathcal{O}((N+M) \cdot 2^{|\Phi|})$.

the CTL* model-checking problem is PSPACE-complete

Bisimulation

Implementation relations

- A binary relation on transition systems
  - when does a transition systems correctly implement another?
- Important for system synthesis
  - stepwise refinement of a system specification TS into an "implementation" TS'
- Important for system analysis
  - use the implementation relation as a means for abstraction
  - replace $TS \models \varphi$ by $TS' \models \varphi$ where $|TS'| \ll |TS|$ such that:

$$TS \models \varphi \text{ iff } TS' \models \varphi \quad \text{or} \quad TS' \models \varphi \;\Rightarrow\; TS \models \varphi$$

$\Rightarrow$ Focus on state-based bisimulation and simulation

- logical characterization: which logical formulas are preserved by bisimulation?

Bisimulation equivalence

Let $TS_i = (S_i, Act_i, \rightarrow_i, I_i, AP, L_i)$, i=1, 2, be transition systems
A bisimulation for (TS_1, TS_2) is a binary relation $\mathcal{R} \subseteq S_1 \times S_2$ such that:

1. $\forall s_1 \in I_1 \; \exists s_2 \in I_2. \; (s_1, s_2) \in \mathcal{R}$ and $\forall s_2 \in I_2 \; \exists s_1 \in I_1. \; (s_1, s_2) \in \mathcal{R}$
2. for all states $s_1 \in S_1$, $s_2 \in S_2$ with $(s_1, s_2) \in \mathcal{R}$ it holds:

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

   2.2 if $s_1' \in Post(s_1)$ then there exists $s_2' \in Post(s_2)$ with $(s_1', s_2') \in \mathcal{R}$

   2.3 if $s_2' \in Post(s_2)$ then there exists $s_1' \in Post(s_1)$ with $(s_1', s_2') \in \mathcal{R}$

TS_1 and TS_2 are bisimilar, denoted $TS_1 \sim TS_2$, if there exists a bisimulation for (TS_1, TS_2)

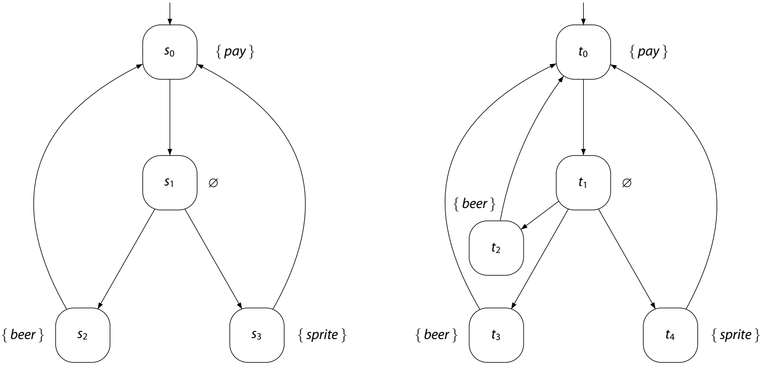
Bisimulation equivalence

$$\begin{array}{ccc} q_1 & \to & q_1' \\ \mathcal{R} & & \\ q_2 & & \end{array} \quad \text{can be completed to} \quad \begin{array}{ccc} q_1 & \to & q_1' \\ \mathcal{R} & & \mathcal{R} \\ q_2 & \to & q_2' \end{array}$$

and

$$\begin{array}{ccc} q_1 & & \\ \mathcal{R} & & \\ q_2 & \to & q_2' \end{array} \quad \text{can be completed to} \quad \begin{array}{ccc} q_1 & \to & q_1' \\ \mathcal{R} & & \mathcal{R} \\ q_2 & \to & q_2' \end{array}$$

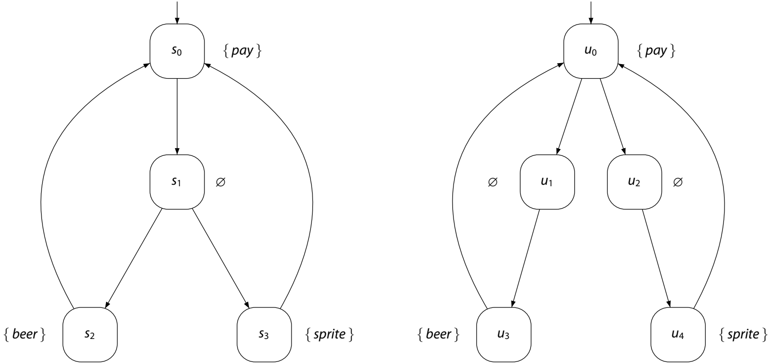
Example (1)



$$\mathcal{R} = \{(s_0, t_0), (s_1, t_1), (s_2, t_2), (s_2, t_3), (s_3, t_4)\}$$

is a bisimulation for (TS_1, TS_2) where $AP = \{ pay, beer, sprite \}$

Example (2)



$TS_1 \not\sim TS_3$ for $AP = \{ pay, beer, sprite \}$

But: $\{ (s_0, u_0), (s_1, u_1), (s_1, u_2), (s_2, u_3), (s_2, u_4), (s_3, u_3), (s_3, u_4) \}$

is a bisimulation for (TS_1, TS_3) for $AP = \{ pay, drink \}$

$\sim$ is an equivalence

For any transition systems TS, TS_1, TS_2 and TS_3 over AP:

$TS \sim TS$ (reflexivity)

$TS_1 \sim TS_2$ implies $TS_2 \sim TS_1$ (symmetry)

$TS_1 \sim TS_2$ and $TS_2 \sim TS_3$ implies $TS_1 \sim TS_3$ (transitivity)

Bisimulation on paths

Whenever we have:

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

this can be completed to

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

proof: by induction on index i of state s_i

Bisimulation vs. trace equivalence

$$TS_1 \sim TS_2 \quad \text{implies} \quad Traces(TS_1) = Traces(TS_2)$$

bisimilar transition systems thus satisfy the same LT properties!

Bisimulation on states

$\mathcal{R} \subseteq S \times S$ is a bisimulation on TS if for any $(q_1, q_2) \in \mathcal{R}$:

- $L(q_1) = L(q_2)$
- if $q_1' \in Post(q_1)$ then there exists an $q_2' \in Post(q_2)$ with $(q_1', q_2') \in \mathcal{R}$
- if $q_2' \in Post(q_2)$ then there exists an $q_1' \in Post(q_1)$ with $(q_1', q_2') \in \mathcal{R}$

q_1 and q_2 are bisimilar, $q_1 \sim_{TS} q_2$, if $(q_1, q_2) \in \mathcal{R}$ for some bisimulation $\mathcal{R}$ for TS

$q_1 \sim_{TS} q_2$ if and only if $TS_{q_1} \sim TS_{q_2}$

Coarsest bisimulation

$\sim_{TS}$ is an equivalence and the coarsest bisimulation for *TS*

Quotient transition system

For $TS = (S, Act, \to, I, AP, L)$ and bisimulation $\sim_{TS} \subseteq S \times S$ on TS let

$TS/\sim_{TS} = (S', \{\tau\}, \to', I', AP, L')$, the quotient of TS under $\sim_{TS}$

where

- $S' = S/\sim_{TS} = \{ [s]_\sim \mid s \in S \}$ with $[s]_\sim = \{ s' \in S \mid s \sim_{TS} s' \}$
- $\to'$ is defined by: $$\frac{s \xrightarrow{\alpha} s'}{[s]_\sim \xrightarrow{\tau}{}' [s']_\sim}$$
- $I' = \{ [s]_\sim \mid s \in I \}$
- $L'([s]_\sim) = L(s)$

The Bakery algorithm

$$P_1 :: \left[\begin{array}{l} \textbf{loop forever do} \\ \left[\begin{array}{ll} & \textbf{noncritical} \\ n_1: & y_1 := y_2 + 1 \\ w_1: & \textbf{await}\ (y_2 = 0 \lor y_1 < y_2) \\ c_1: & \textbf{critical} \\ & y_1 := 0 \end{array}\right] \end{array}\right] \quad \| \quad P_2 :: \left[\begin{array}{l} \textbf{loop forever do} \\ \left[\begin{array}{ll} & \textbf{noncritical} \\ n_1: & y_2 := y_1 + 1 \\ w_1: & \textbf{await}\ (y_1 = 0 \lor y_2 < y_1) \\ c_1: & \textbf{critical} \\ & y_2 := 0 \end{array}\right] \end{array}\right]$$

Example path fragment

process P_1	process P_2	y_1	y_2	effect
n_1	n_2	0	0	P_1 requests access to critical section
w_1	n_2	1	0	P_2 requests access to critical section
w_1	w_2	1	2	P_1 enters the critical section
c_1	w_2	1	2	P_1 leaves the critical section
n_1	w_2	0	2	P_1 requests access to critical section
w_1	w_2	3	2	P_2 enters the critical section
w_1	c_2	3	2	P_2 leaves the critical section
w_1	n_2	3	0	P_2 requests access to critical section
w_1	w_2	3	4	P_1 enters the critical section
...	...	..	..	...

Data abstraction

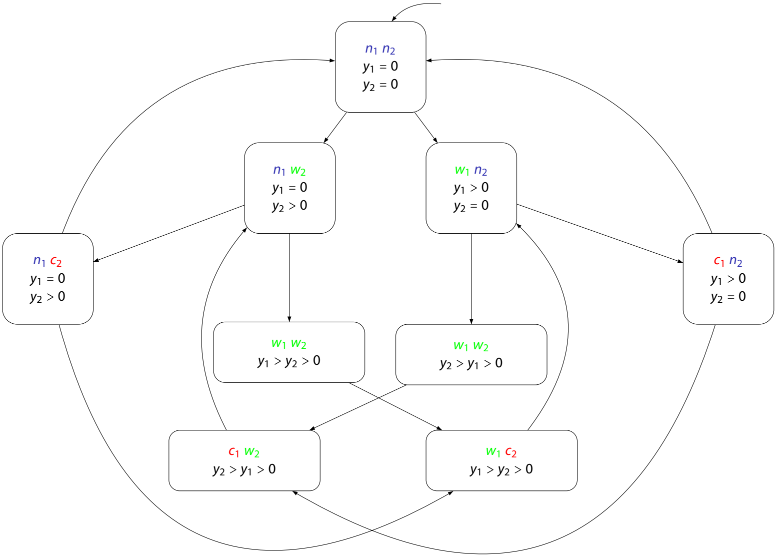
Function f maps a reachable state of TS_{Bak} onto an abstract one in TS_{Bak}^{abs}
Let $s = \langle \ell_1, \ell_2, y_1 = b_1, y_2 = b_2 \rangle$ be a state of TS_{Bak} with $\ell_i \in \{ n_i, w_i, c_i \}$ and $b_i \in \mathbb{N}$
Then:

$$f(s) = \begin{cases} \langle \ell_1, \ell_2, y_1 = 0, y_2 = 0 \rangle & \text{if } b_1 = b_2 = 0 \\ \langle \ell_1, \ell_2, y_1 = 0, y_2 > 0 \rangle & \text{if } b_1 = 0 \text{ and } b_2 > 0 \\ \langle \ell_1, \ell_2, y_1 > 0, y_2 = 0 \rangle & \text{if } b_1 > 0 \text{ and } b_2 = 0 \\ \langle \ell_1, \ell_2, y_1 > y_2 > 0 \rangle & \text{if } b_1 > b_2 > 0 \\ \langle \ell_1, \ell_2, y_2 > y_1 > 0 \rangle & \text{if } b_2 > b_1 > 0 \end{cases}$$

$\mathcal{R} = \{ (s, f(s)) \mid s \in S \}$ is a bisimulation for $(TS_{Bak}, TS_{Bak}^{abs})$

for any subset of $AP = \{ noncrit_i, wait_i, crit_i \mid i = 1, 2 \}$

Bisimulation quotient



$$TS_{Bak}^{abs} = TS_{Bak}/\sim \quad \text{for} \quad AP = \{\, crit_1, crit_2 \,\}$$

Remarks

- In this example, data abstraction yields a bisimulation relation
  - (typically, only a simulation relation is obtained, more later)
- $TS_{Bak}^{abs} \models \varphi$ with, e.g.,:
  - $\Box(\neg crit_1 \vee \neg crit_2)$ and
    $(\Box\Diamond wait_1 \Rightarrow \Box\Diamond crit_1) \wedge (\Box\Diamond wait_2 \Rightarrow \Box\Diamond crit_2)$
- Since $TS_{Bak}^{abs} \sim TS_{Bak}$, it follows $TS_{Bak} \models \varphi$
- Note: $Traces(TS_{Bak}^{abs}) = Traces(TS_{Bak})$

CTL* equivalence

States q_1 and q_2 in TS (over AP) are CTL*-equivalent:

$$q_1 \equiv_{CTL^*} q_2 \quad \text{if and only if} \quad (q_1 \models \Phi \text{ iff } q_2 \models \Phi)$$

for all CTL* state formulas over AP

$$TS_1 \equiv_{CTL^*} TS_2 \quad \text{if and only if} \quad (TS_1 \models \Phi \text{ iff } TS_2 \models \Phi)$$

for any sublogic of CTL*, logical equivalence is defined analogously

Bisimulation vs. CTL* and CTL equivalence

Let TS be a finite state graph 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., $s \equiv_{CTL^*} s'$

this is proven in three steps: $\equiv_{CTL} \subseteq \sim \subseteq \equiv_{CTL^*} \subseteq \equiv_{CTL}$

important: equivalence is also obtained for any sub-logic containing $\neg$, $\wedge$ and X

The importance of this result

- CTL and CTL* equivalence coincide
  - despite the fact that CTL* is more expressive than CTL
- 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 (or CTL*) formula
  - $TS_1 \models \Phi$ and $TS_2 \not\models \Phi$ implies $TS_1 \not\sim TS_2$
- You even do not need to use an until-operator!
- To check $TS \models \Phi$, it suffices to check $TS/\!\sim\, \models \Phi$