

Model Checking

René Thiemann

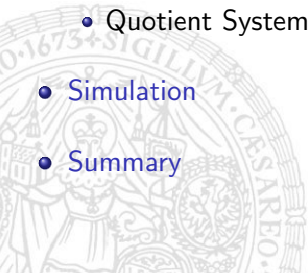
Institute of Computer Science
University of Innsbruck

SS 2009



Outline

- Motivation
- Abstraction
- Bisimulation
 - Bisimulation of Transition Systems
 - Bisimulation of States
 - Bisimulation and Temporal Logics
 - Quotient Systems
- Simulation
- Summary

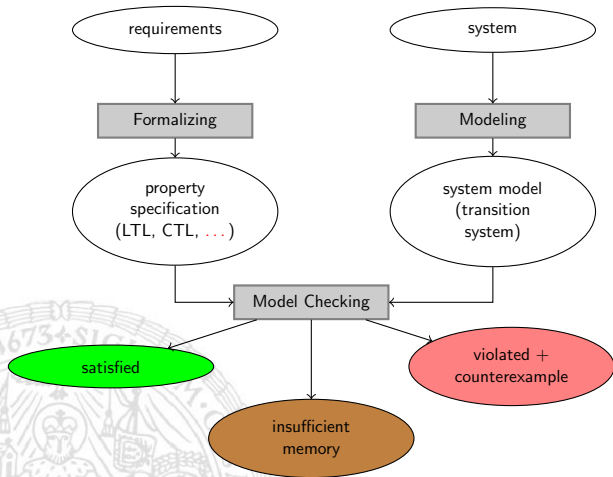


Outline

- Motivation
- Abstraction
- Bisimulation
 - Bisimulation of Transition Systems
 - Bisimulation of States
 - Bisimulation and Temporal Logics
 - Quotient Systems
- Simulation
- Summary



Model Checking Overview



Ways to Solve the State Space Explosion Problem

- Let $TS = (S, \rightarrow, I, AP, L)$ be transition system
- Abstraction: $f : S \rightarrow \hat{S}$ such that $|\hat{S}| \ll |S|$, obtain \hat{TS}
- Then perform model checking on abstract system: $\hat{TS} \models \varphi$?
- Questions:
 - If $\hat{TS} \models \varphi$, what about $TS \models \varphi$?
 - If $\hat{TS} \not\models \varphi$, what about $TS \not\models \varphi$?
 - How to obtain f ?
- Some answers:
 - If \hat{TS} is a **bisimulation** of TS then $\hat{TS} \models \varphi$ iff $TS \models \varphi$ (CTL*)
 - If \hat{TS} is a **simulation** of TS then $\hat{TS} \models \varphi$ implies $TS \models \varphi$ (ACTL*)
 - If TS is a **simulation** of \hat{TS} then $\hat{TS} \models \varphi$ implies $TS \models \varphi$ (ECTL*)
 - Computation of f such that \hat{TS} is smallest bisimilar system to TS

Outline

- Motivation
- **Abstraction**
- Bisimulation
 - Bisimulation of Transition Systems
 - Bisimulation of States
 - Bisimulation and Temporal Logics
 - Quotient Systems
- Simulation
- Summary



Abstraction

Let $TS = (S, \rightarrow, I, AP, L)$ and \widehat{S} be a set of (abstract) states

Definition (Abstraction Function)

A function $f : S \rightarrow \widehat{S}$ is an **abstraction function** iff

$$f(s) = f(s') \text{ implies } L(s) = L(s')$$

Definition (Abstracted Transition System)

For every abstraction function f , define the **over-approximation**

$TS^f = (\widehat{S}, \rightarrow^f, I^f, AP, L^f)$ where $L^f(f(s)) = L(s)$, $I^f = \{f(s) \mid s \in I\}$, and \rightarrow^f is smallest relation such that

- $s \rightarrow s'$ implies $f(s) \rightarrow^f f(s')$

The **under-approximation** is $TS_f = (\widehat{S}, \rightarrow_f, I_f, AP, L_f)$ where $L_f = L^f$, $I_f = I^f$, and \rightarrow_f is largest relation such that

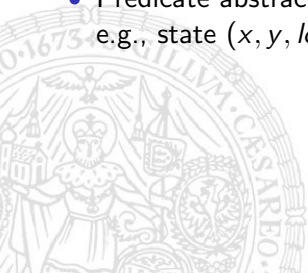
- $f(s) \rightarrow_f \widehat{s}$ implies $s \rightarrow s'$ for some s' such that $f(s') = \widehat{s}$

Example



Different Kinds of Abstractions

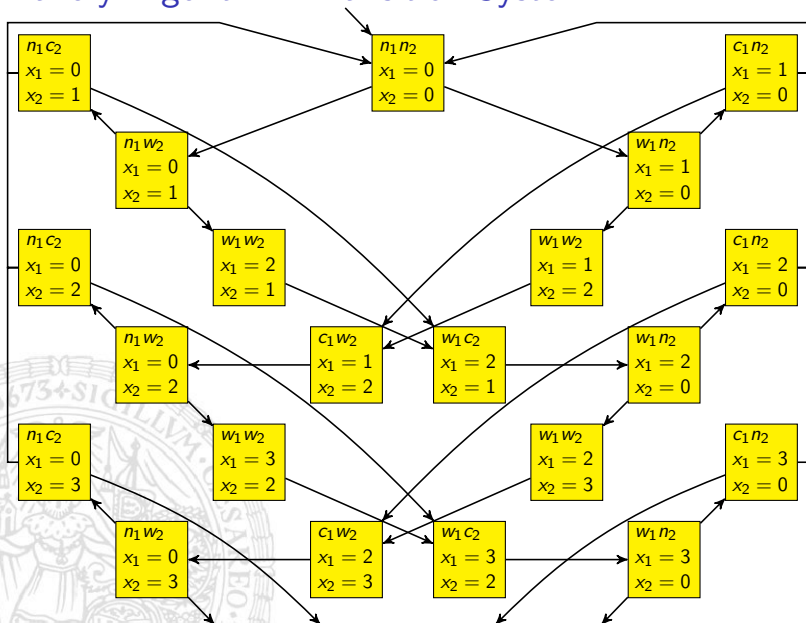
- Variable abstraction: only store subset of all variables
e.g., $\text{state}(x, y, loc) \rightsquigarrow \text{state}(x, loc)$
- Data abstraction: concrete domain \rightsquigarrow abstract (smaller) domain
e.g., $\mathbb{IN} \rightsquigarrow \{\text{even}, \text{odd}\}$ or $\mathbb{IN} \rightsquigarrow \{\text{pos}, 0, \text{neg}\}$
- Predicate abstraction: state \rightsquigarrow valuation of the predicates
e.g., $\text{state}(x, y, loc) \rightsquigarrow \text{state}(x > 0, x > y, loc = \text{crit})$



Example: Bakery algorithm



Bakery Algorithm: Transition System



Bakery Algorithm: Abstraction

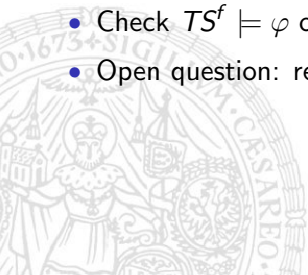


Abstraction Summary

- Abstraction function $f : S \rightarrow \widehat{S}$ for AP such that

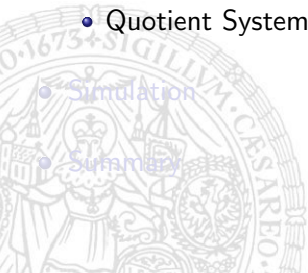
$$f(s) = f(s') \text{ implies } L(s) = L(s')$$

- From large (possibly infinite) system TS obtain small (possibly finite) abstract system TS^f or TS_f
- Check $TS^f \models \varphi$ or $TS_f \models \varphi$ instead of $TS \models \varphi$
- Open question: relation between $TS^f \models \varphi$, $TS_f \models \varphi$, and $TS \models \varphi$



Outline

- Motivation
- Abstraction
- **Bisimulation**
 - Bisimulation of Transition Systems
 - Bisimulation of States
 - Bisimulation and Temporal Logics
 - Quotient Systems
- Simulation
- Summary



Bisimulation Between Two Transition Systems

Let $TS_i = (S_i, \rightarrow_i, I_i, AP, L_i)$ be two transition systems.

Definition

A relation $R \subseteq S_1 \times S_2$ is a **bisimulation relation** iff

1. for all $s \in I_1$ exists $t \in I_2 : sRt$ and
for all $t \in I_2$ exists $s \in I_1 : sRt$ and
2. for all sRt it holds:
 - $L_1(s) = L_2(t)$
 - if $s \rightarrow_1 s'$ then $t \rightarrow_2 t'$ where $s'Rt'$
 - if $t \rightarrow_2 t'$ then $s \rightarrow_1 s'$ where $s'Rt'$

TS_1 and TS_2 are **bisimilar** ($TS_1 \sim TS_2$) iff there is a bisimulation relation R for TS_1 and TS_2

Example



Properties of Bisimulations

Lemma

\sim is an **equivalence relation** (\sim is reflexive, symmetric, transitive)

Lemma (Path Bisimulation)

Let R be a bisimulation of TS_1 and TS_2 , let $s_0 R t_0$.

Then for each path

$s_0 s_1 s_2 s_3 \dots$ of TS_1

there is a **bisimilar path**, i.e., a path

$t_0 t_1 t_2 t_3 \dots$ of TS_2

such that for all i : $s_i R t_i$

Corollary (LTL-Equivalence of Bisimilar Systems)

If $TS_1 \sim TS_2$ then $TS_1 \models \varphi$ iff $TS_2 \models \varphi$ for all LTL-formulas φ

Bisimulation of States

- Up to now: Bisimulation between two transition systems
 - Upcoming: Bisimulation between states of same system
- ⇒ Minimize number of states

Definition (Bisimilar States)

Let $TS = (S, \rightarrow, I, AP, L)$ be a transition system.

$R \subseteq S \times S$ is a **bisimulation** for TS such that for all sRt :

- $L(s) = L(t)$
- if $s \rightarrow s'$ then $t \rightarrow t'$ where $s'Rt'$
- if $t \rightarrow t'$ then $s \rightarrow s'$ where $s'Rt'$

States s and t are **bisimilar** for TS ($s \sim_{TS} t$) iff there exists bisimulation R for TS with sRt .

Properties of \sim_{TS}

Let $TS = (S, \rightarrow, I, AP, L)$ be a transition system.

Lemma

- \sim_{TS} is an equivalence relation on S
- \sim_{TS} is a bisimulation for TS
- \sim_{TS} is the largest bisimulation for TS
- $s_1 \sim_{TS} s_2$ iff $(S, \rightarrow, \{s_1\}, AP, L) \sim (S, \rightarrow, \{s_2\}, AP, L)$

Consequence: Deciding $TS_0 \sim TS_1$ via \sim_{TS}

Corollary (Check of bisimilarity of transition systems)

Let $TS_i = (S_i, \rightarrow_i, I_i, AP, L_i)$ with $S_0 \cap S_1 = \emptyset$. Then $TS_0 \sim TS_1$ iff

for all $s_i \in I_i$ there is $s_{1-i} \in I_{1-i}$ such that $s_i \sim_{TS} s_{1-i}$

where $TS = (S_0 \cup S_1, \rightarrow_0 \cup \rightarrow_1, \emptyset, AP, L_0 \cup L_1)$

Proof of Lemma



Short Reminder: CTL*

A **state-formula** Φ holds in state s (written $s \models \Phi$) iff

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

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

$$s \models \Phi \wedge \Psi \quad \text{iff} \quad s \models \Phi \text{ and } s \models \Psi$$

$$s \models E\varphi \quad \text{iff} \quad \pi \models \varphi \text{ for some path } \pi \text{ that starts in } s$$

A **path-formula** φ holds for path π (written $\pi \models \varphi$) iff

$$\pi \models X\varphi \quad \text{iff} \quad \pi[1..] \models \varphi$$

$$\pi \models \varphi U \psi \quad \text{iff} \quad (\exists n \geq 0. \pi[n..] \models \psi \text{ and } (\forall 0 \leq i < n. \pi[i..] \models \varphi))$$

$$\pi \models \varphi \wedge \psi \quad \text{iff} \quad \pi \models \varphi \text{ and } \pi \models \psi$$

$$\pi \models \neg \varphi \quad \text{iff} \quad \pi \not\models \varphi$$

$$\pi \models \Phi \quad \text{iff} \quad \pi[0] \models \Phi$$

Derived operators: **A**, **F**, **G**, \forall , \dots

Bisimulation and CTL*

Let $TS = (S, \rightarrow, I, AP, L)$. Define $\equiv_{CTL^*} \subseteq S \times S$ as

$s \equiv_{CTL^*} t$ iff $(s \models \Phi \text{ iff } t \models \Phi)$ for all CTL*-state-formulas Φ

Similar definition for \equiv_{CTL}

Theorem

$$\equiv_{CTL} = \equiv_{CTL^*} = \sim_{TS}$$

\Rightarrow Bisimilar systems satisfy the same CTL*-formulas

\Rightarrow Non-bisimilar systems can be distinguished by a CTL-formula

Proof



Proof Continued



Quotient System

Since \sim_{TS} is equivalence relation, we can write $[s]_{\sim_{TS}}$ as the equivalence class to which s belongs ($[s]_{\sim_{TS}} = \{t \mid s \sim_{TS} t\}$).

Definition (Quotient of a Transition System)

Let $TS = (S, \rightarrow, I, AP, L)$. The **quotient system** TS/\sim_{TS} (or TS/\sim for short) is defined as $(S', \rightarrow', I', AP, L')$:

- $S' = S/\sim_{TS} = \{[s]_{\sim_{TS}} \mid s \in S\}$
- whenever $s \rightarrow t$ then $[s]_{\sim_{TS}} \rightarrow' [t]_{\sim_{TS}}$
- $I' = I/\sim_{TS} = \{[s]_{\sim_{TS}} \mid s \in I\}$
- $L'([s]_{\sim_{TS}}) = L(s)$

Theorem

$$TS \sim (TS/\sim)$$

Examples

- Bakery-Algorithm: $TS^f = TS/\sim$
(However, often TS^f is not a bisimulation)
- Vending machines: $TS_2/\sim = TS_1$, $s_3 = [t_2]_{\sim_{TS_2}} = [t_3]_{\sim_{TS_2}} = \{t_2, t_3\}$



Obtaining Quotients

If one can compute \sim_{TS} then one can easily

- minimize TS to quotient system TS/\sim
- check whether $TS_0 \sim TS_1$

Problem: How to obtain \sim_{TS} ?

- Naive algorithm:

$$\sim_{TS} := \emptyset$$

for all $R \subseteq S \times S$ do

if R is bisimulation for TS then $\sim_{TS} := \sim_{TS} \cup R$

Naive algorithm is exponential in $|S| \Rightarrow$ not applicable

- Partition-Refinement-Algorithm, complexity: $\mathcal{O}(|S| \cdot (|AP| + |\rightarrow|))$
- (Improved PR-Algorithm, complexity: $\mathcal{O}(|S| \cdot |AP| + \log|S| \cdot |\rightarrow|)$)

Idea of a Partition Refinement Algorithm

- Work with partitions $\Pi = \{B_1, \dots, B_n\}$ of S
($\cup B_i = S$, $B_i \cap B_j = \emptyset$ for $i \neq j$, $B_i \neq \emptyset$)
- Partition Π contains candidates for equivalence classes
- If Π is too coarse since some B contains obviously non-equivalent states s and t then refine Π and split B into smaller parts B_1 and B_2 such that $s \in B_1$ and $t \in B_2$

\Rightarrow Refine initial Π until no further splitting is required

- Final value of $\Pi = \{C_1, \dots, C_k\}$ contains real equivalence classes C_i of \sim_{TS}

$\Rightarrow s \sim_{TS} t$ iff s, t are contained in same C_i

Partition Refinement Algorithm

$\Pi := \Pi_{AP}$ // partitioning of S due to labeling with AP

repeat

$\Pi_{old} := \Pi$

for all $C \in \Pi_{old}$ **do**

$\Pi := \text{refine}(\Pi, C)$

until $\Pi = \Pi_{old}$

return Π // result: S/\sim_{TS}

function $\text{refine}(\Pi, C)$ // divide partitions due to transitions to C

return $\bigcup_{B \in \Pi} \text{refine}(B, C)$

function $\text{refine}(B, C)$

return $\{\{s \in B \mid s \rightarrow t, t \in C\}, \{s \in B \mid \text{no } s \rightarrow t \text{ with } t \in C\}\} \setminus \emptyset$

$\Pi_{AP} = \{\{s \mid L(s) = A\} \mid A \subseteq AP\} \setminus \emptyset$

Example



Properties of refine

Definition

Partition Π is finer than Π' (Π' is coarser than Π) iff

for all $B \in \Pi$ there exists $C \in \Pi'$ such that $B \subseteq C$

Key lemmas:

Lemma (Coarsest Partition)

S/\sim_{TS} is coarsest partition Π such that

- Π is finer than Π_{AP}
- $\text{refine}(\Pi, C) = \Pi$ for all $C \in \Pi$

Lemma (Properties of refine)

If Π, Π' are coarser than S/\sim_{TS} then

- $\text{refine}(\Pi, C)$ is finer than Π
- $\text{refine}(\Pi, C)$ is coarser than S/\sim_{TS} for all $C \in \Pi'$

Proof of Coarsest-Partition Lemma



Properties of the Algorithm

Theorem

- *The algorithm terminates*
- *The complexity is $\mathcal{O}(|S| \cdot (|AP| + |\rightarrow|))$*
- *The result is the set of equivalence classes of \sim_{TS} , i.e., S/\sim_{TS}*



Proof

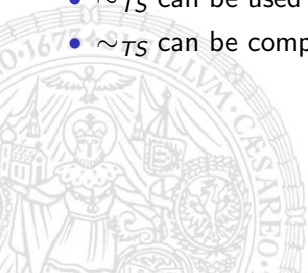


Bisimulation Summary

- $TS_1 \sim TS_2$ iff for all CTL*-formulas Φ : $TS_1 \models \Phi \Leftrightarrow TS_2 \models \Phi$

$$\sim = \equiv_{CTL^*}$$

- Smallest bisimilar system to TS : $TS/\sim_{TS} = TS/\sim$
- \sim_{TS} can be used to decide $TS_1 \sim TS_2$
- \sim_{TS} can be computed by partitioning algorithm



Outline

- Motivation
- Abstraction
- Bisimulation
 - Bisimulation of Transition Systems
 - Bisimulation of States
 - Bisimulation and Temporal Logics
 - Quotient Systems
- **Simulation**
- Summary



A Problem

Current approach:

- Given TS , compute TS/\sim_{TS} and then check formula
- Often, TS/\sim_{TS} is still too large
- Solution: Use abstraction function f such that $TS^f(TS_f) \ll TS/\sim_{TS}$
- Problem: for these f , $TS^f \not\sim TS$ and $TS_f \not\sim TS$

\Rightarrow There are CTL*-formulas Φ and Ψ such that

$$TS^f \models \Phi \not\sim TS \models \Phi \quad \text{and} \quad TS_f \models \Psi \not\sim TS \models \Psi$$

\Rightarrow Need for another connection between transition systems

Simulation Between Two Transition Systems

Let $TS_i = (S_i, \rightarrow_i, I_i, AP, L_i)$ be two transition systems.

Definition

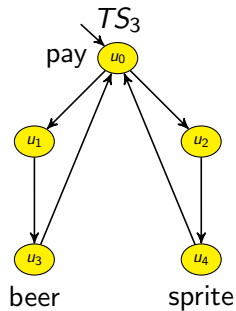
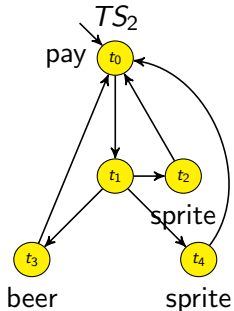
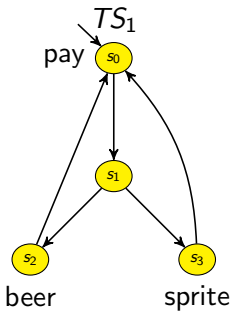
A relation $R \subseteq S_1 \times S_2$ is a **simulation relation** iff

1. for all $s \in I_1$ exists $t \in I_2 : sRt$ and
2. for all sRt it holds:
 - $L_1(s) = L_2(t)$
 - if $s \rightarrow_1 s'$ then $t \rightarrow_2 t'$ where $s'Rt'$

TS_1 is **simulated** by TS_2 ($TS_1 \preceq TS_2$) iff there is a **simulation** relation R for TS_1 and TS_2

Note that unlike \sim , \preceq is **no equivalence** relation

Example



Previous results: $TS_1 \sim TS_2 \not\sim TS_3$

Lemma (Path Simulation)

Let R be a *simulation* of TS_1 and TS_2 , let $s_0 R t_0$.

Then for each path

$s_0 s_1 s_2 s_3 \dots$ of TS_1

there is a *similar* path, i.e., a path

$t_0 t_1 t_2 t_3 \dots$ of TS_2

such that for all i : $s_i R t_i$

Corollary (LTL and Similar Systems)

If $TS_1 \preceq TS_2$ then $TS_1 \models \varphi$ *if* $TS_2 \models \varphi$ for all LTL-formulas φ
and $TS_1 \not\models \varphi$ implies $TS_2 \not\models \varphi$

Corollary (LTL and Similar Systems)

Define $\simeq = \preceq \cap \succeq$ (*simulation equivalence*). Then

$$\simeq \subseteq \equiv_{LTL}$$

Simulations and Abstractions

Theorem

Let TS be some transition system, and f be an abstraction function. Then

$$TS \preceq TS^f \quad \text{and} \quad TS_f \preceq TS.$$

Corollary (Model Checking using Abstractions)

Let φ be arbitrary LTL-formula.

- If $TS^f \models \varphi$ then $TS \models \varphi$
- If $TS_f \not\models \varphi$ then $TS \not\models \varphi$

Proof of Theorem



Properties of \preceq

Lemma

- \preceq is a pre-order (reflexive and transitive)
- \simeq is an equivalence relation
- $\sim \subseteq \simeq$

Note that both \sim and \simeq satisfy the path simulation lemma and are equivalence relations. Moreover,

$$\equiv_{CTL^*} = \sim \subseteq \simeq \subseteq \equiv_{LTL}$$

Questions:

- Is $\sim = \simeq$? Then $\simeq = \equiv_{CTL^*}$
- If not, then where is the difference?

Example



Strengthening the Logic

Knowledge:

- $TS_1 \preceq TS_2$ implies $TS_1 \models \varphi \Leftrightarrow TS_2 \models \varphi$ for LTL-formulas φ
- $TS_1 \succeq TS_2$ implies $TS_1 \not\models \varphi \Leftrightarrow TS_2 \not\models \varphi$ for LTL-formulas φ
- $TS_1 \simeq TS_2$ implies $TS_1 \models \varphi \Leftrightarrow TS_2 \models \varphi$ for LTL-formulas φ
- $TS_1 \simeq TS_2$ does not imply $TS_1 \models \Phi \Leftrightarrow TS_2 \models \Phi$ for CTL-formulas Φ
- $TS \preceq TS^f$ and $TS \succeq TS_f$

Want:

- Stronger logic than LTL which allows model-checking via TS^f :

$$TS \models \Phi \Leftrightarrow TS^f \models \Phi$$

- Logic which allows model-checking via TS_f :

$$TS \models \Phi \Leftrightarrow TS_f \models \Phi$$

ACTL* = CTL* with All-Quantifier Only

ACTL*-state-formulas:

$$\Phi ::= a \mid \neg a \mid \Phi \vee \Phi \mid \Phi \wedge \Phi \mid A\varphi$$

ACTL*-path-formulas:

$$\varphi ::= X\varphi \mid \varphi U\varphi \mid \varphi R\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \Phi$$

Semantics of release-operator R :

$$\pi \models \varphi R\psi \text{ iff } \forall n : \pi[n..] \models \psi \text{ or } (\exists i : \pi[i..] \models \varphi \text{ and } \forall j \leq i : \pi[j..] \models \psi)$$

Derived path-operators:

$$F\varphi \equiv \text{true } U\varphi \quad \text{and} \quad G\varphi \equiv \text{false } R\varphi$$

Equivalences:

$$\neg(\varphi U\psi) \equiv \neg\varphi R\neg\psi \quad \text{and} \quad \neg(\varphi R\psi) \equiv \neg\varphi U\neg\psi$$

Comparing LTL, ACTL*, and CTL*

Theorem

- *ACTL* strictly subsumes LTL*
- *CTL* strictly subsumes ACTL**



ACTL* strictly subsumes LTL

- First we show that each LTL-formula φ can be translated into positive normal form (PNF), where LTL-formula in PNF has following shape:

$$\varphi ::= X\varphi \mid \varphi U \varphi \mid \varphi R \varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid a \mid \neg a$$

$$\neg\neg\varphi \rightsquigarrow \varphi$$

$$\neg X\varphi \rightsquigarrow X\neg\varphi$$

$$\neg(\varphi U \psi) \rightsquigarrow \neg\varphi R \neg\psi$$

$$\neg(\varphi R \psi) \rightsquigarrow \neg\varphi U \neg\psi$$

$$\neg(\varphi \wedge \psi) \rightsquigarrow \neg\varphi \vee \neg\psi$$

$$\neg(\varphi \vee \psi) \rightsquigarrow \neg\varphi \wedge \neg\psi$$

Hence, for LTL-formula φ obtain equivalent ψ in PNF. Then φ is equivalent to the ACTL*-formula $A\psi$. Thus, ACTL* subsumes LTL.

CTL* strictly subsumes ACTL*

- Obviously, CTL* subsumes ACTL* as release can be expressed using negation and until:

$$\varphi R \psi \equiv \neg \neg (\varphi R \psi) \equiv \neg (\neg \varphi U \neg \psi)$$

- Similar to the previous results between \sim and CTL* one can show that for all ACTL* formulas Φ :

$$TS_1 \preceq TS_2 \text{ implies } TS_1 \models \Phi \text{ if } TS_2 \models \Phi$$

Hence,

$$\equiv_{CTL^*} = \sim \subset \simeq \subseteq \equiv_{ACTL^*}$$

shows that there must be CTL*-formulas which cannot be expressed in ACTL*, i.e., CTL* strictly subsumes ACTL*.

ECTL*

Results so far:

- ACTL*: Stronger logic than LTL, model-checking via TS^f :

$$TS \models \Phi \iff TS^f \models \Phi$$

- ECTL*: Logic, model-checking via TS_f :

$$TS \models \Phi \iff TS_f \models \Phi$$

ECTL*-state-formulas:

$$\Phi ::= a \mid \neg a \mid \Phi \vee \Phi \mid \Phi \wedge \Phi \mid E\varphi$$

ECTL*-path-formulas:

$$\varphi ::= X\varphi \mid \varphi U \varphi \mid \varphi R \varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \Phi$$

Simulation Summary

- Abstractions do not often lead to bisimulations, but always result in simulations:

$$TS \preceq TS^f \quad \text{and} \quad TS_f \succeq TS$$

- ACTL* is between LTL and CTL* and can be checked for model-checking using abstractions (over-approximations)

$$TS_1 \preceq TS_2 \text{ implies } TS_1 \models \Phi \text{ if } TS_2 \models \Phi$$

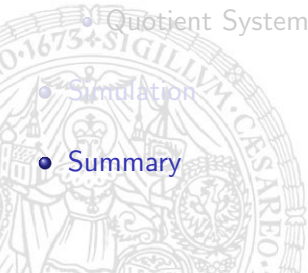
- ECTL* is sublogic of CTL* and can be checked for model-checking using abstractions (under-approximations)

$$TS_1 \succeq TS_2 \text{ implies } TS_1 \models \Phi \text{ if } TS_2 \models \Phi$$

- Reversing the directions yields methods to refute formulas
- Not shown:
 - Computing the quotient of \simeq in analogy to S/\sim
 - How to obtain initial abstractions, abstraction refinement

Outline

- Motivation
- Abstraction
- Bisimulation
 - Bisimulation of Transition Systems
 - Bisimulation of States
 - Bisimulation and Temporal Logics
 - Quotient Systems
- Simulation
- Summary



Summary

- Aim: Try to solve the state-space explosion problem
- Bisimilar systems satisfy the same CTL*-formulas
- Quotient S/\sim can efficiently be determined by partition-refinement
- If quotient is too large, one can further reduce the system-size by abstractions (over-approximation TS^f and under-approximation TS_f)
 \Rightarrow obtain simulation only
- For simulations LTL and (A/E)CTL* can be used, but neither CTL nor CTL*
- Challenge: Find good abstractions