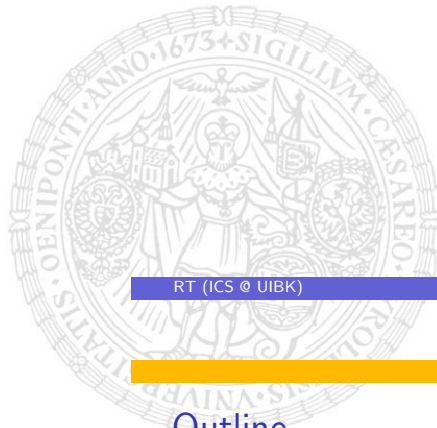


## Introduction to Model Checking

René Thiemann

Institute of Computer Science  
University of Innsbruck

WS 2007/2008



### Outline

- Last Lecture
- “Model Checking” Lecture
- Repetition
  - NanoPromela to Transition Systems
  - LTL to NBA

## CTL\*-Model Checking Algorithm [Emerson, Lei]

1. Eliminate existential path quantifiers

$$E\varphi \equiv \neg A\neg\varphi$$

2. Use bottom-up CTL-model checking procedure

- $Sat(\text{true}) = S$
- $Sat(a) = \{s \mid a \in L(s)\}$
- $Sat(\neg\Phi) = S \setminus Sat(\Phi)$
- $Sat(\Phi \wedge \Psi) = Sat(\Phi) \cap Sat(\Psi)$

3. Integrate LTL-model checker for universal path formulas

## Use LTL-Model Checker for Universal Formulas

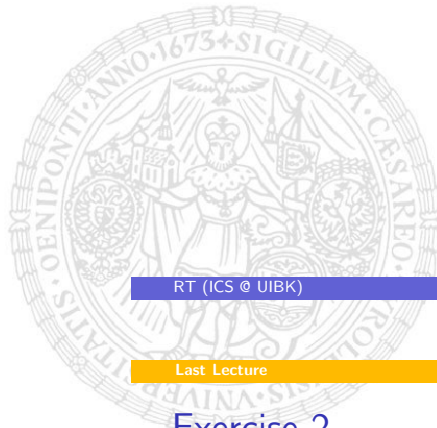
Idea: to compute  $s \in Sat(A\varphi)$  perform LTL model checking of  $\varphi$

- Replace every maximal state-formula  $\Psi$  in  $\varphi$  which is not an atomic proposition by a new atomic proposition  $a_\Psi$ , result: LTL-formula  $\varphi'$
- Extend labeling of states: Whenever  $s \in Sat(\Psi)$  then add  $a_\Psi$  to the set of labels of  $s$ .
- Apply LTL-model checker to determine  $Sat(A\varphi)$ :

$$s \in Sat(A\varphi) \text{ iff } s \models \varphi'$$

## Exercise 1

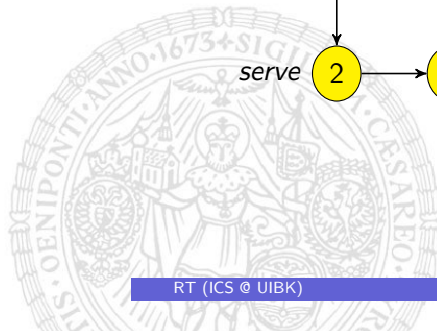
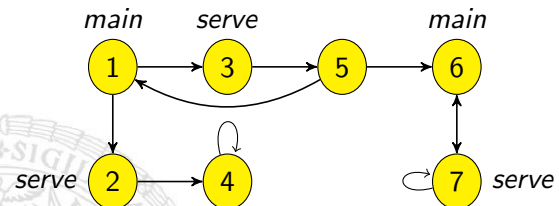
Prove that there is no LTL-formula which is equivalent to  
 $\Phi = \text{AF}(b \wedge \text{AX} b)$



## Exercise 2

Consider the formula  $\Phi = \text{AG}((\neg \text{EF } \textit{serve}) \vee (\text{EGF } \textit{main}))$

- Try to formulate the meaning in words
- Apply the CTL\*-model checking algorithm on the following example



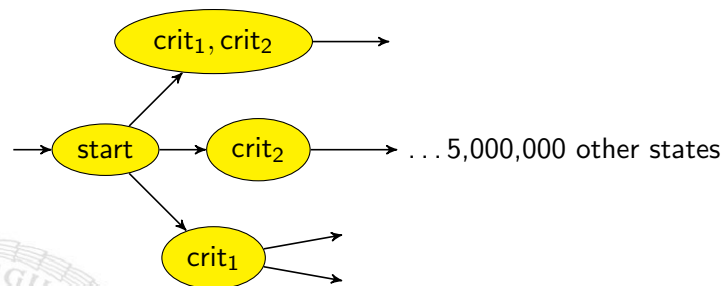
## Selection of Topics

- Model checking on the fly
- $\mu$ -calculus
- SIS
- Model checking of real-time systems
- Controlling the state-space explosion problem
- ...

## Model Checking on the Fly

Consider  $\varphi = G \neg(\text{crit}_1 \wedge \text{crit}_2)$

Transition system:



Full construction of  $TS \otimes \mathcal{A}_{\neg\varphi}$  to check emptiness is too expensive  
 $\Rightarrow$  only construct part of  $TS$  which suffices to check  $TS \models \varphi$

### Model Checking on the Fly

## $\mu$ -Calculus

In CTL: semantics based on least and greatest fixpoint

In  $\mu$ -calculus:

- explicit least- and greatest fixpoint operators
  - easy to implement
  - many logics can be translated into  $\mu$ -calculus
  - parallel model checking algorithms available
- ⇒  $\mu$ -calculus as efficient basis for model-checking for several logics

## S1S

Consider the following property:

Between every green and red phase there is at least one orange phase.

Formulating these kinds of properties in LTL is doable, but not intuitive

$$G(\text{red} \Rightarrow X(G \neg \text{green}) \vee (\neg \text{green} \wedge (X \neg \text{green} U \text{orange}))))$$

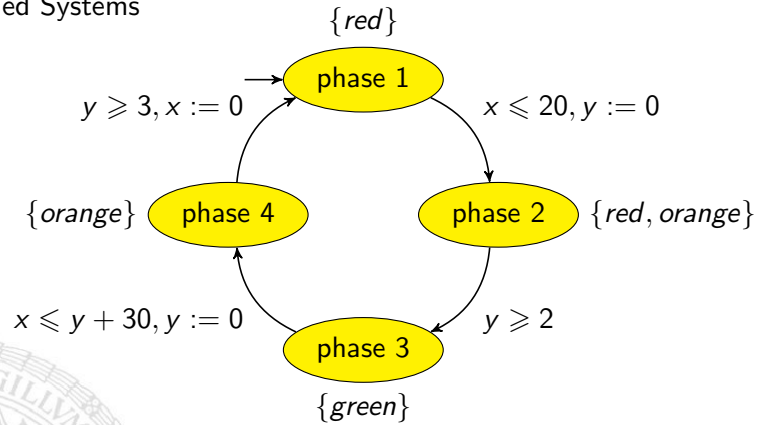
Use S1S instead:

$$\forall t_1, t_2 : (t_1 < t_2 \wedge \text{green}(t_1) \wedge \text{red}(t_2)) \Rightarrow \exists t_3 : t_1 < t_3 < t_2 \wedge \text{orange}(t_3)$$

- Allows readable and succinct specifications
- One can perform model checking by NBAs

## Model Checking of Real-Time Systems

- Timed Systems



- Timed Specifications

One does not have to wait more than 50 seconds for green:

$$\Phi = GF \leq^{50} \text{green}$$

## Controlling the State-Space Explosion Problem

Reduce search space in various ways

- Abstraction:
  - instead of 16-bit integer, only distinguish between even and odd, or between positive, 0, negative, or between ...
- Partial order reduction:
  - if process 1 and process 2 perform operations on **local** variables, then schedule process 1 always before process 2
  - ⇒ less interleaving, smaller transition system

## NanoPromela to Channel System: Locations = Sub-Expr.

$$\begin{array}{c}
 \frac{}{x := \text{expr} \xrightarrow{\text{true} : \text{assign}(x, \text{expr})} \text{exit}} \\
 \frac{\text{stmt}_1 \xrightarrow{g:\alpha} \text{stmt}'_1 \neq \text{exit}}{\text{stmt}_1; \text{stmt}_2 \xrightarrow{g:\alpha} \text{stmt}'_1; \text{stmt}_2} \\
 \frac{\text{stmt}_1 \xrightarrow{g:\alpha} \text{exit}}{\text{stmt}_1; \text{stmt}_2 \xrightarrow{g:\alpha} \text{stmt}_2} \\
 \frac{\text{stmt}_i \xrightarrow{h:\alpha} \text{stmt}'_i}{\text{if } \dots :: g_i \Rightarrow \text{stmt}_i \dots \text{fi} \xrightarrow{g_i \wedge h:\alpha} \text{stmt}'_i} \\
 \frac{\text{stmt}_i \xrightarrow{h:\alpha} \text{stmt}'_i \neq \text{exit}}{\text{do } \dots :: g_i \Rightarrow \text{stmt}_i \dots \text{od} \xrightarrow{g_i \wedge h:\alpha} \text{stmt}'_i; \text{do } \dots \text{od}} \\
 \frac{\text{stmt}_i \xrightarrow{h:\alpha} \text{exit}}{\text{do } \dots :: g_i \Rightarrow \text{stmt}_i \dots \text{od} \xrightarrow{g_i \wedge h:\alpha} \text{do } \dots \text{od}}
 \end{array}$$

## Example: Mutual Exclusion of 2 Processes

```

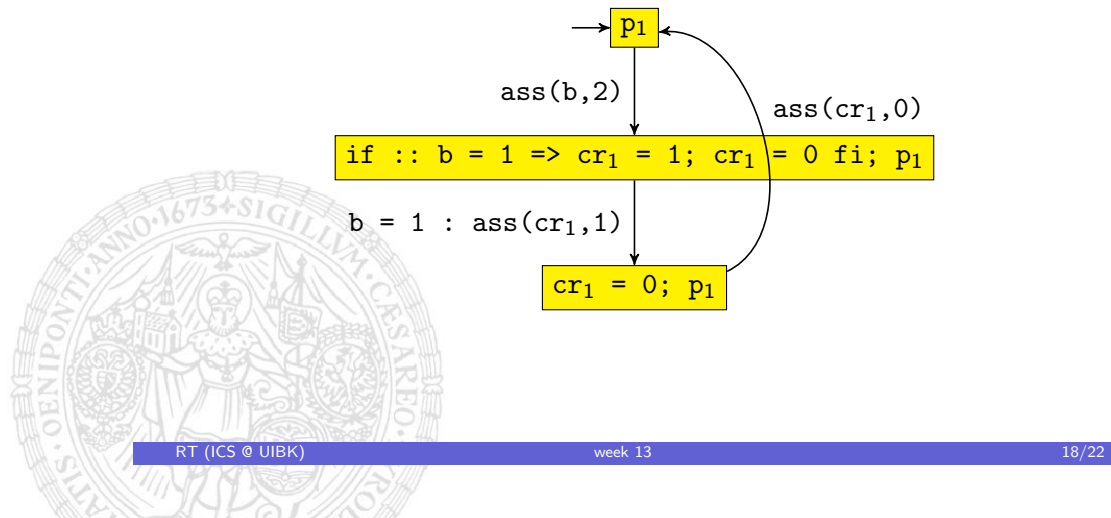
p1 = do :: true => b = 2; if :: b = 1 => cr1 = 1; cr1 = 0 fi od
p2 = do :: true => b = 1; if :: b = 2 => cr2 = 1; cr2 = 0 fi od

```

### Example: Mutual Exclusion of 2 Processes

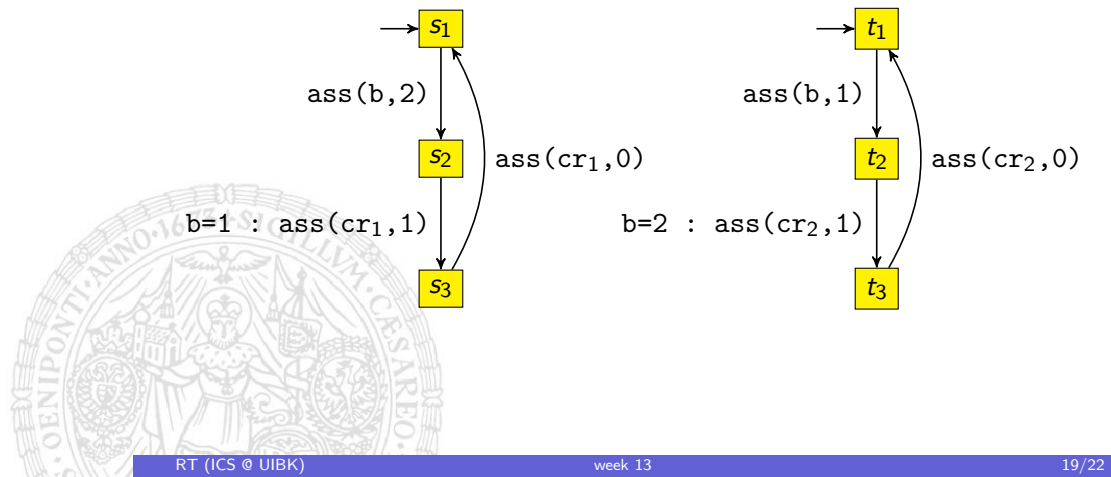
$p_1 = \text{do} :: \text{true} \Rightarrow b = 2; \text{if} :: b = 1 \Rightarrow \text{cr}_1 = 1; \text{cr}_1 = 0 \text{ fi} \text{ od}$   
 $p_2 = \text{do} :: \text{true} \Rightarrow b = 1; \text{if} :: b = 2 \Rightarrow \text{cr}_2 = 1; \text{cr}_2 = 0 \text{ fi} \text{ od}$

For  $p_1$  obtain:



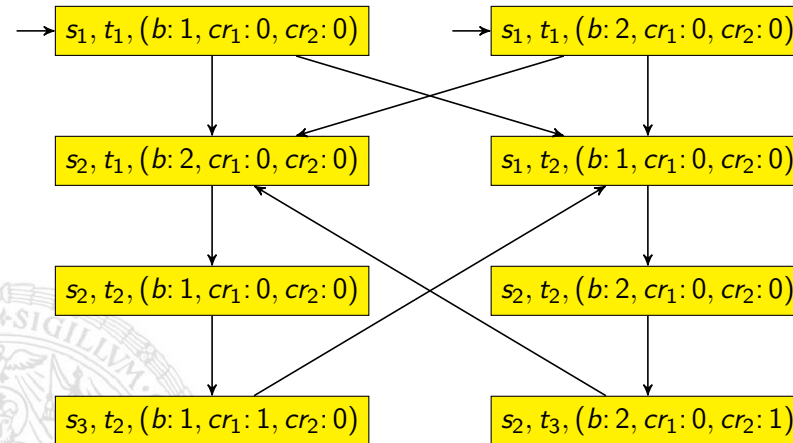
### Example: Resulting Channel System

Renaming states of  $p_1$  to  $s_1, s_2, s_3$  and those of  $p_2$  to  $t_1, t_2, t_3$  yields the following channel system





## Example: Resulting Transition System



## LTL to NBAs: $\varphi$ -Expansion and Consistency Checks

$$\begin{aligned}
 \varphi_j = \neg \varphi_{j_1} &\Rightarrow w[i..]^j = 1 \text{ iff } w[i..]^{j_1} = 0 \\
 \varphi_j = \varphi_{j_1} \wedge \varphi_{j_2} &\Rightarrow w[i..]^j = 1 \text{ iff } w[i..]^{j_1} = 1 \text{ and } w[i..]^{j_2} = 1 \\
 \varphi_j = \text{X} \varphi_{j_1} &\Rightarrow w[i..]^j = 1 \text{ iff } w[i+1..]^{j_1} = 1 \\
 \varphi_j = \varphi_{j_1} \cup \varphi_{j_2} &\Rightarrow w[i..]^j = 1 \text{ iff } w[i..]^{j_1} = 1 \text{ or } \\
 &\quad (w[i..]^{j_2} = 1 \text{ and } w[i+1]^{j_2} = 1)
 \end{aligned}$$

States of NBA: Vectors where components represent sub-formulas

Transitions  $s_1 \xrightarrow{in} s_2$  where

- $s_1$ : preceding values of  $\varphi$ -expansion or start state  $q_0$
- $in$ : current input
- $s_2$ : current values of  $\varphi$ -expansion

Transitions satisfy consistency checks and  $q_0$  leads to states with  $\varphi$ -component 1

Example  $\varphi = X(a \cup b)$

