Structure and Hierarchy in Real-Time Systems

Modeling and Analysis

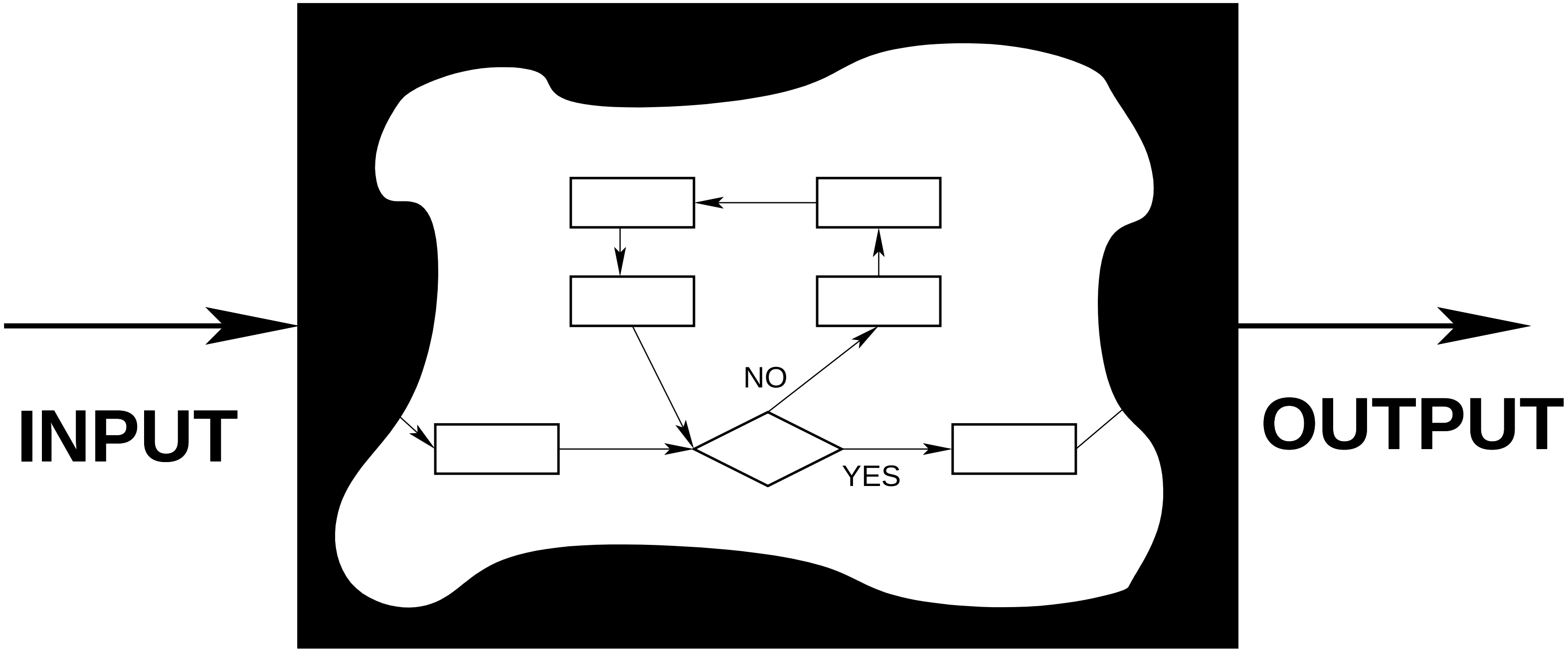
M. Oliver Möller

BRICS PhD School, Århus

Traditional Input/Output Programs

Correctness := relation over **Input** / **Output**

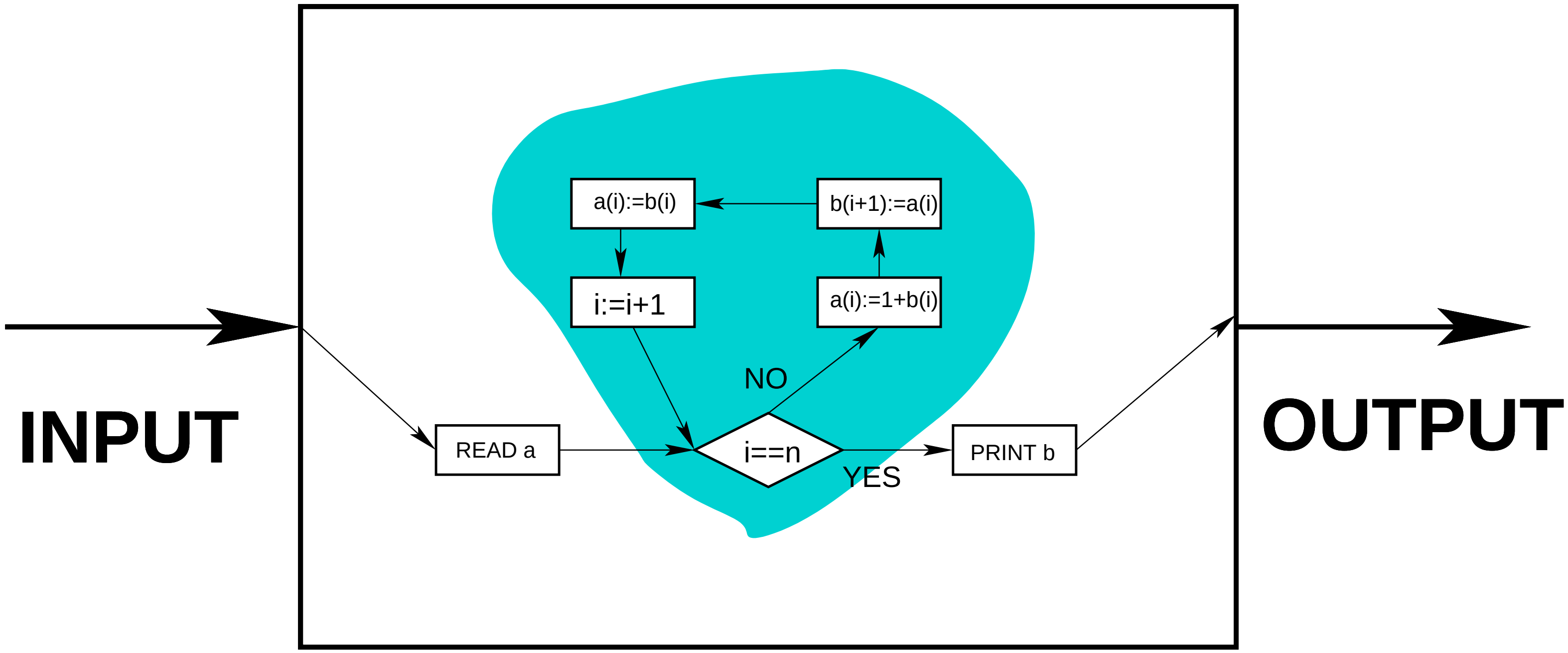
Traditional Input/Output Programs



Correctness := relation over **Input** / **Output**

Testing := try some **typical** and some **borderline** cases

Traditional Input/Output Programs



Correctness := relation over **Input** / **Output**

Testing := try some **typical** and some **borderline** cases

Analysis := proof something for **ALL** inputs

can use **assertions** on sub-structures

Application: Reactive Systems (?)

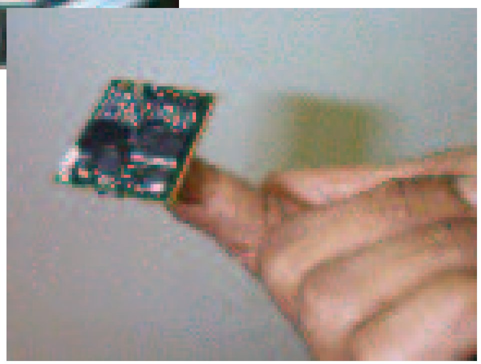
Embedded:

mixture of hard- and software;

severe resource limitations;

interaction with environment

Application: Reactive Systems (?)



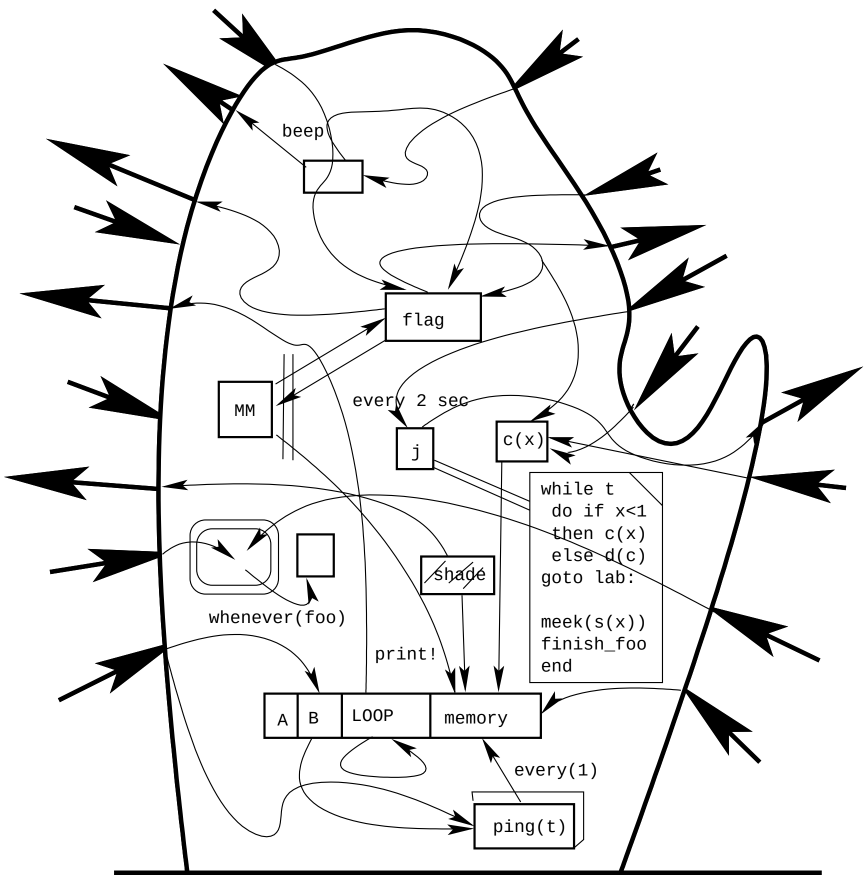
Embedded:
mixture of hard- and software;
severe resource limitations;
interaction with environment

Real-Time:
Correctness not only dependent on the logical order of events, but also on their timing

Reactive Systems are Different

black cactus

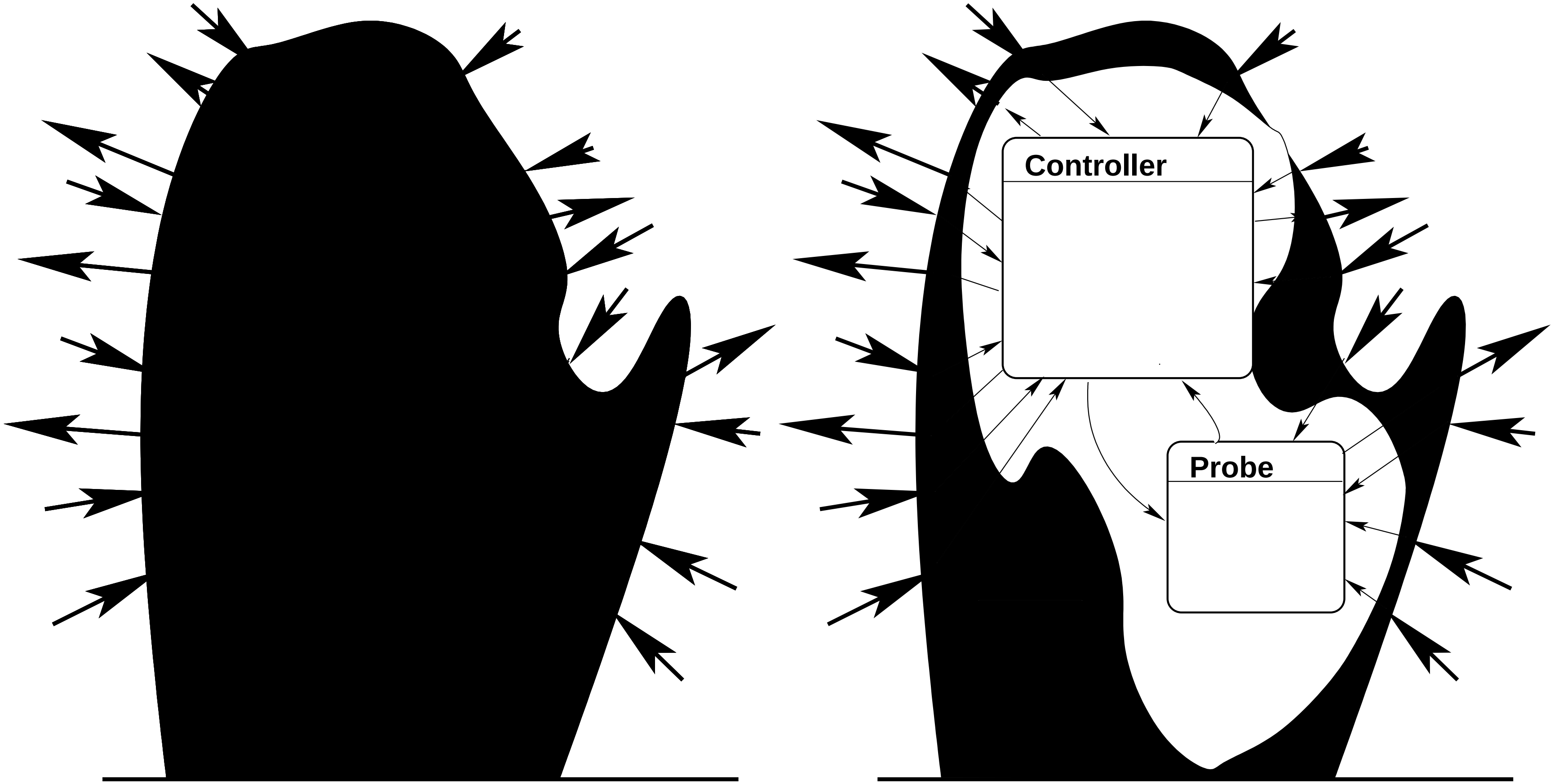
Reactive Systems are Different



black cactus

messy implementation

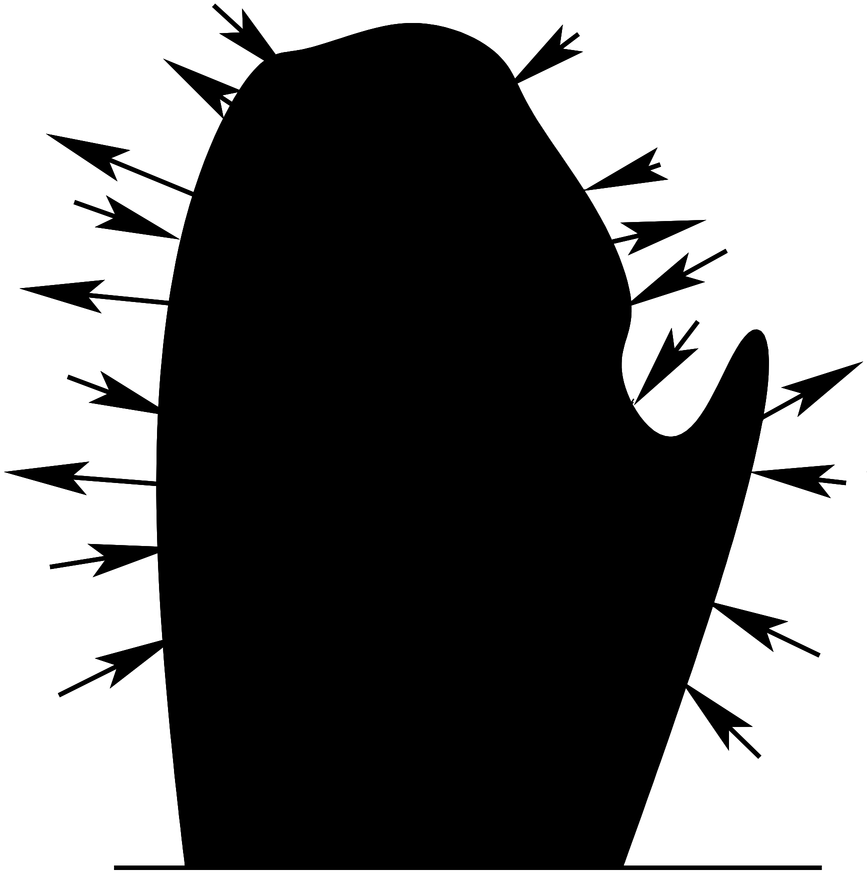
Reactive Systems are Different



black cactus | **structured**

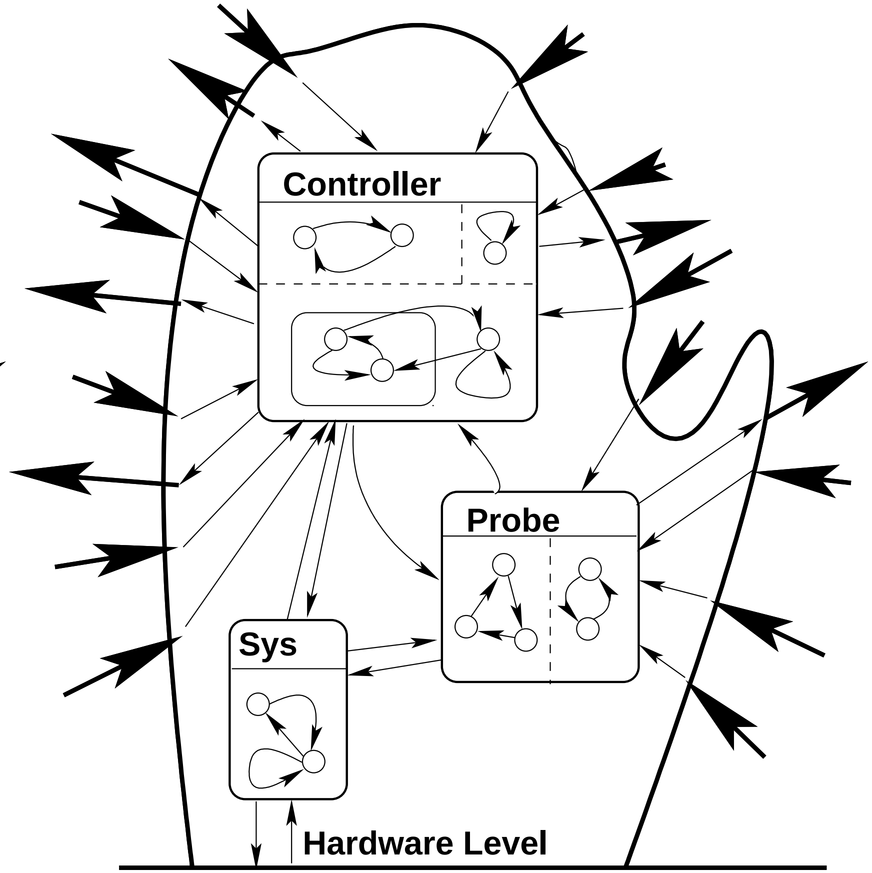
messy implementation

Reactive Systems are Different



black cactus

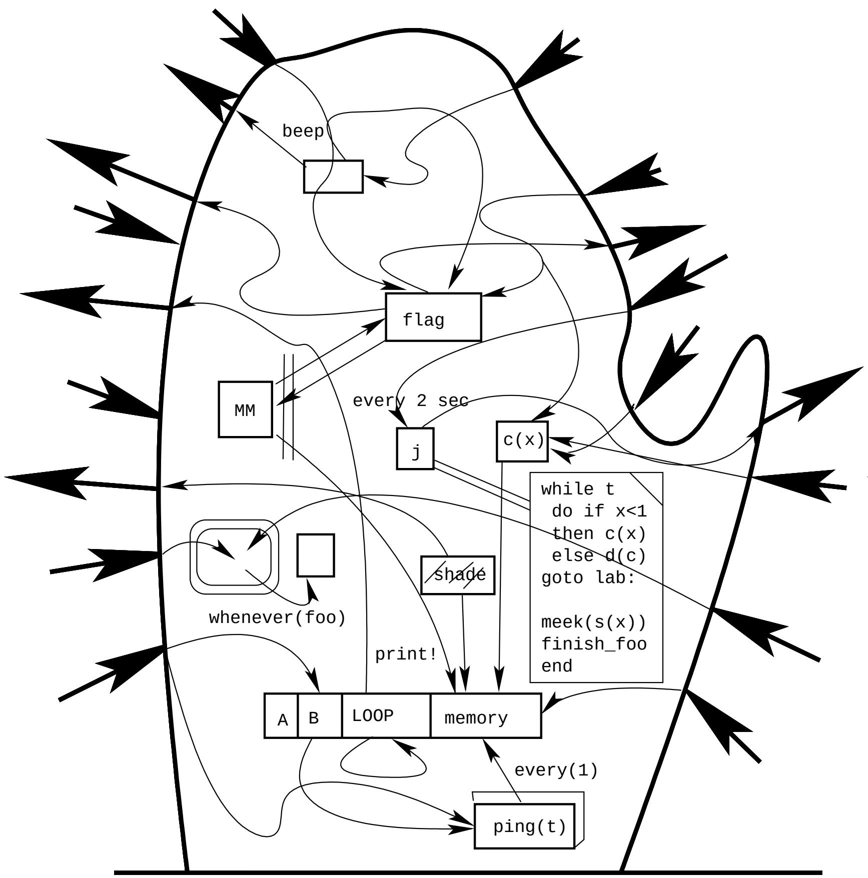
messy implementation



structured

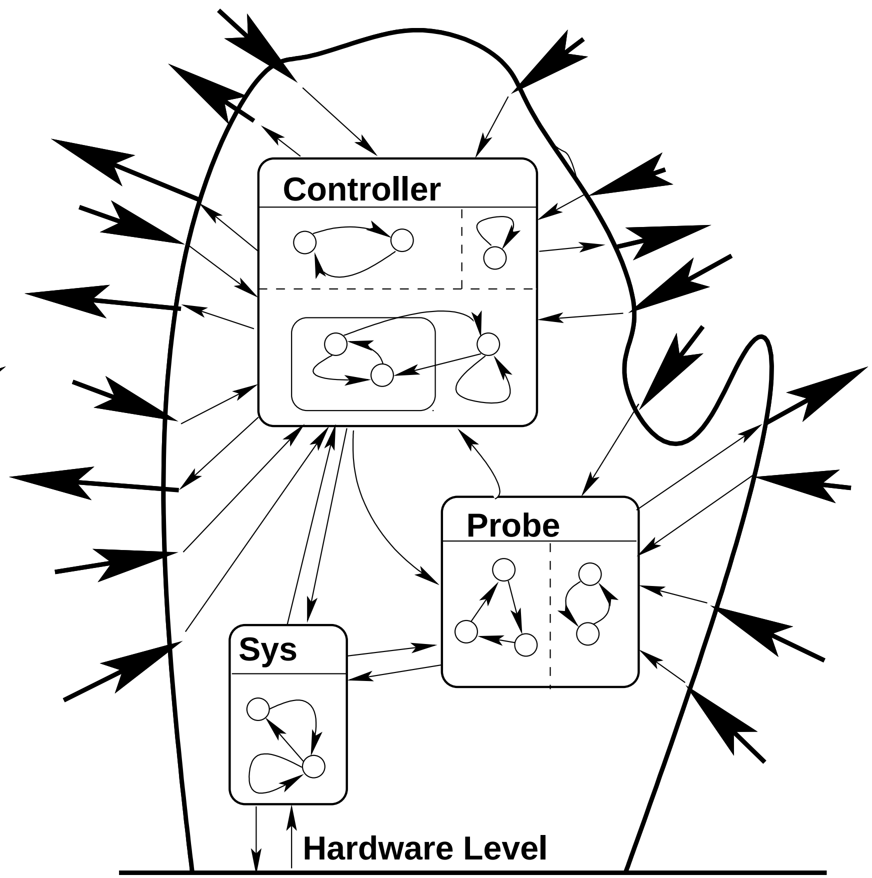
different levels

Reactive Systems are Different



black cactus

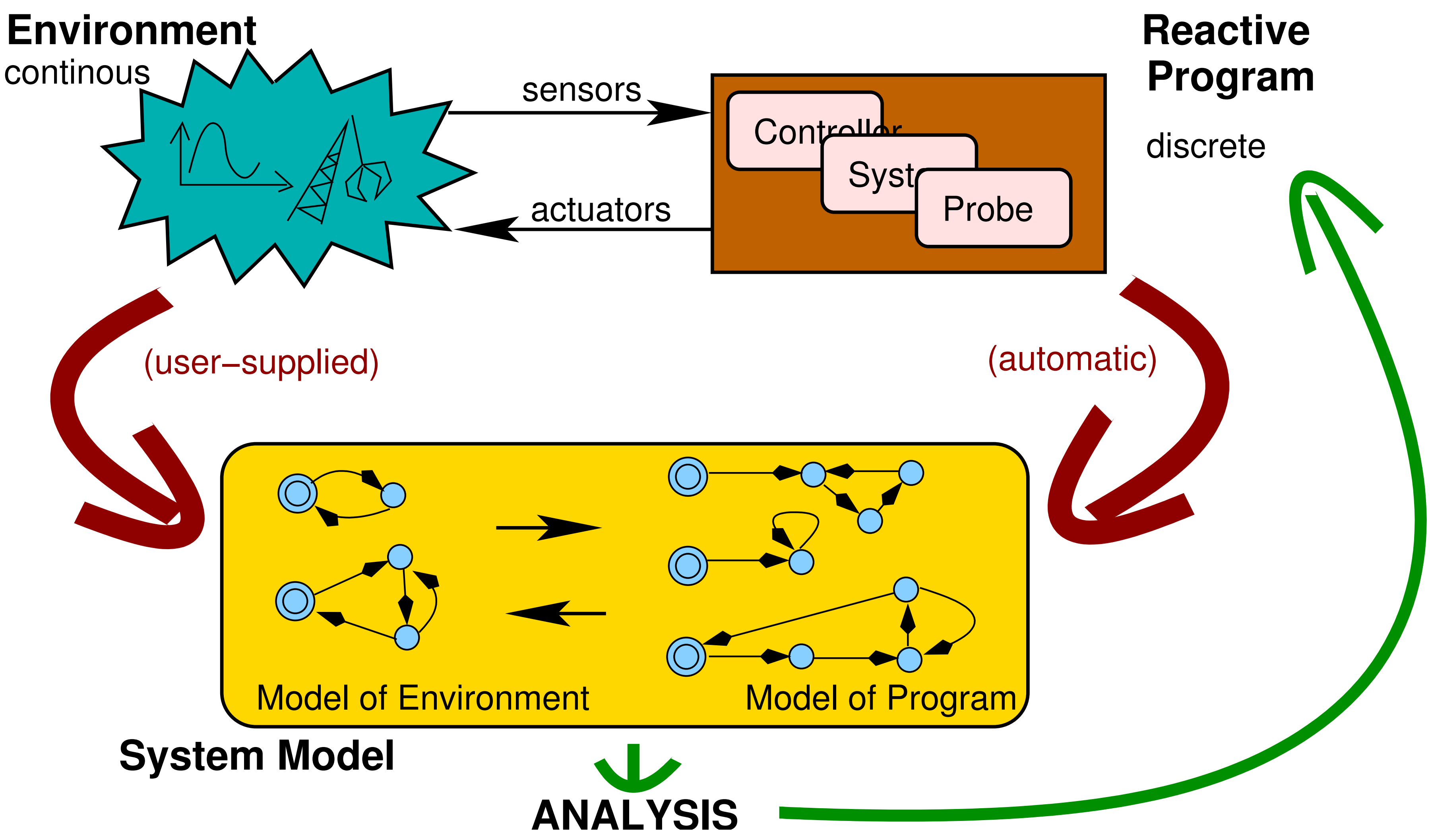
messy implementation



structured

different levels

Composing the Embedded System Model



The Big Questions

- What are appropriate **languages** to model a reactive system **?**
- How do we perform **analysis** on partially completed systems **?**

Outline of the Thesis

Part I: Modeling of Real-Time Systems

1. The unified modeling language (UML) and statecharts (overview)
2. The language of UPPAAL (trace-based semantics)
3. Hierarchical timed automata [NWPT'01,FASE'02,journal submission]

Part II: Algorithmic Verification of Real-Time Systems

4. Real-time model checking: forward analysis (correctness formalization)
5. Optimization techniques for real-time systems (benchmarks)
6. Model augmentation to speed-up model checking [TPTS'01]
7. Predicate abstraction for dense real-time [TPTS'01]

Part III: Making Use of Hierarchical Structure

8. Construction of good hierarchies from parallel components [CHARME'01]
9. Flattening hierarchical timed automata for model checking [NWPT'01,FASE'02,journal submission]

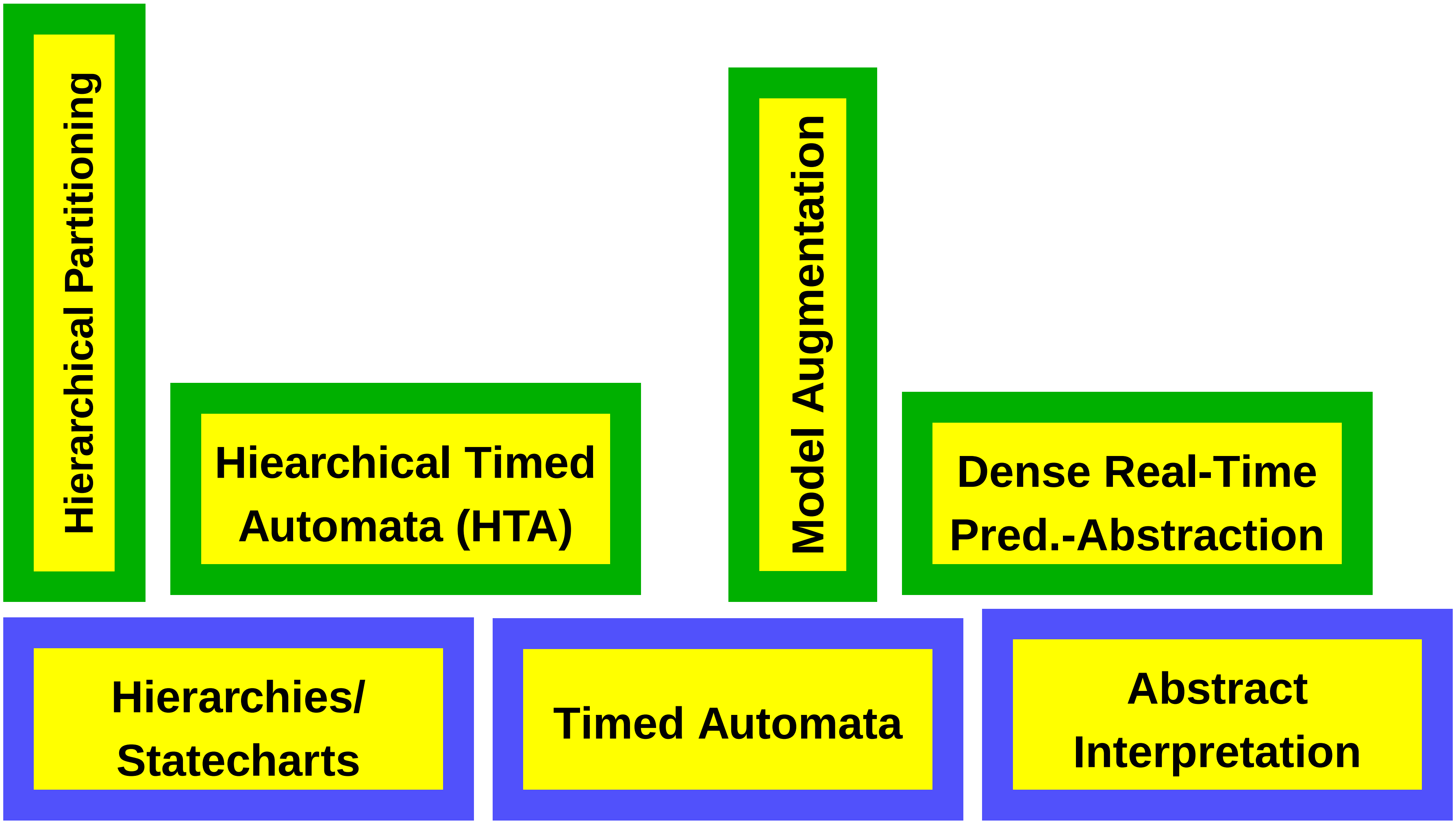
What was Known, What is New?

Hierarchies/
Statecharts

Timed Automata

Abstract
Interpretation

What was Known, What is New?



Outline of the Thesis – and this Talk

Part I: Modeling of Real-Time Systems

1. The unified modeling language (UML) and statecharts (overview)
2. The language of UPPAAL (trace-based semantics)
3. Hierarchical timed automata [NWPT'01,FASE'02,journal submission]

Part II: Algorithmic Verification of Real-Time Systems

4. Real-time model checking: forward analysis (correctness formalization)
5. Optimization techniques for real-time systems (benchmarks)
6. Model augmentation to speed-up model checking [TPTS'01]
7. Predicate abstraction for dense real-time [TPTS'01]

Part III: Making Use of Hierarchical Structure

8. Construction of good hierarchies from parallel components [CHARME'01]
9. Flattening hierarchical timed automata for model checking [NWPT'01,FASE'02,journal submission]

In the Following

Chapter 3: Hierarchical Timed Automata

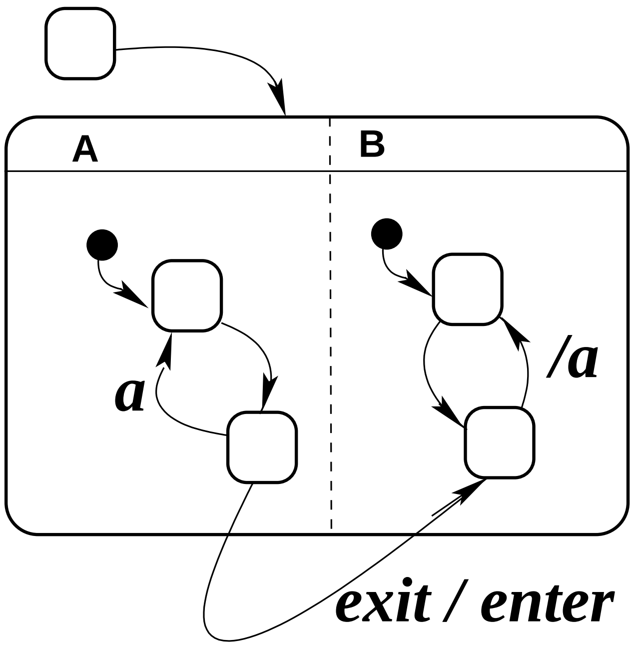
1. Restricted Statecharts with Real-Time
2. A Trace-Based Semantics
3. Flattening and Correspondence
4. Case Study: Cardiac Pacemaker

Chapter 7: Predicate Abstraction for Dense Real-Time

5. Galois Connection to an Untimed Model
6. Progress Assumption by Restricting Delays
7. Successive Refinement

Summary

The Statechart Formalism



Features

- hierarchical state machines
- parallelism (on any level)
- history
- event communication
- powerful synchronization mechanisms
- inter-level transitions
- actions that are dependent on states
- actions on entry/exit
- ...

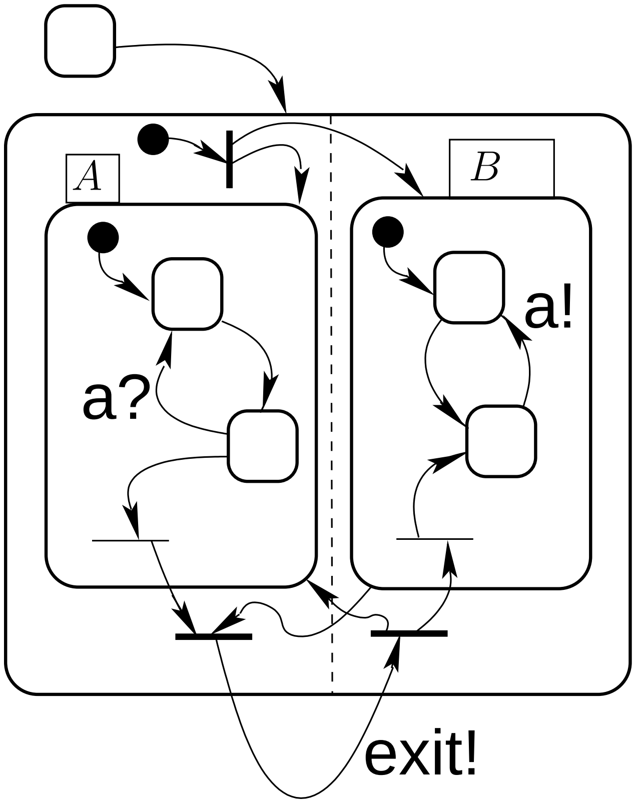
Claim:

The statechart formalism is *appropriate* for the development of reactive systems

Fact:

Basic statechart properties are undecidable $\Rightarrow$ automated analysis *impossible* in general

Restricted Statechart Formalism



Concentration on key features

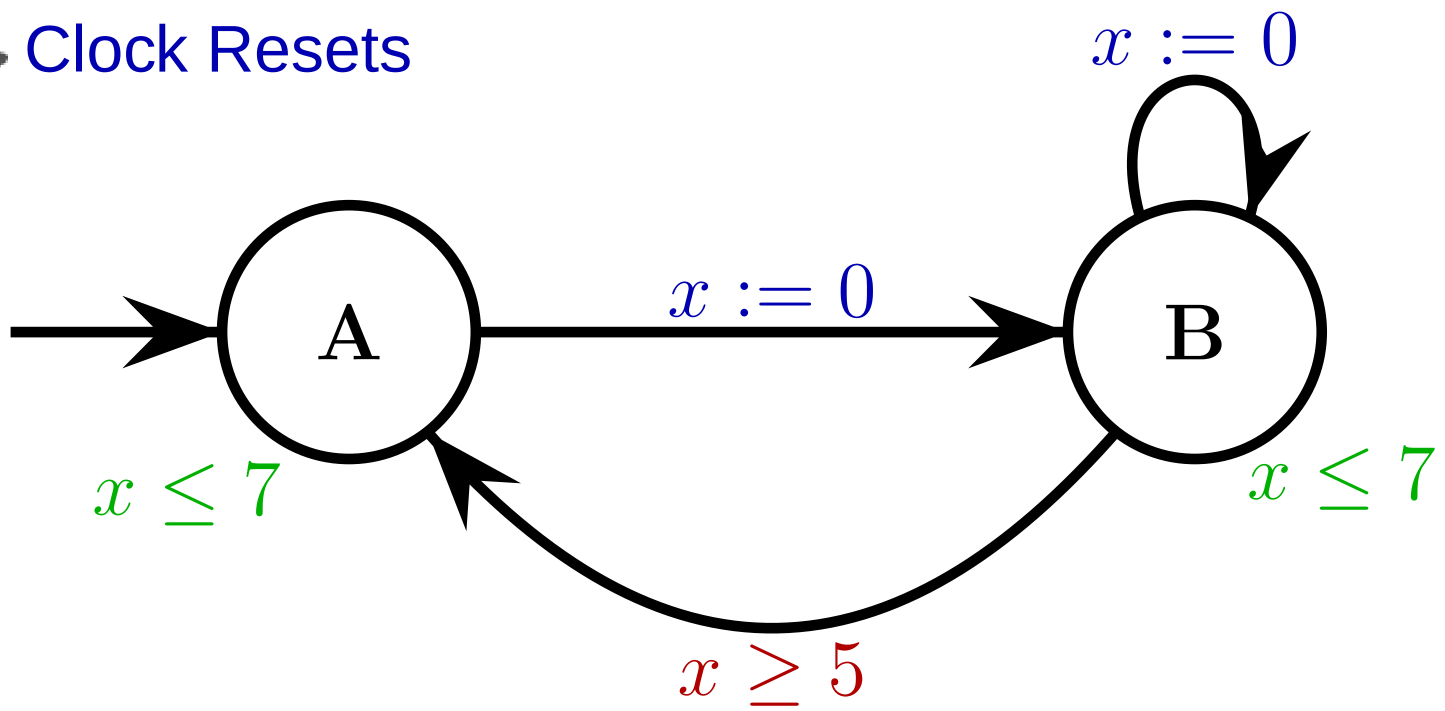
- hierarchical state machines ✔
- parallelism (on any level) ✔
- history ✔
- **no** event communication
- **no** sync states
- **no** inter-level transitions
- **no** actions that are dependent on states
- **no** actions on entry/exit

instead:

- hand-shake style synchronization
- shared variables

Real-Time Extensions

- Clocks
- (timed) Guards
- Invariants
- Clock Resets



Hierarchical Timed Automata (HTAs)

The HTA formalizm has a formal semantics

- operational style
- interprets time as ”dense real-time”
- trace-based

Benefits

- unambiguous
- mechanizable
- you can proof something about it

Semantic Rules (Example)

configuration: $\langle \rho,\ \mu,\ \nu,\ \theta \rangle$ with

ρ	:	control locations
μ	:	valuation of integer variables
ν	:	valuation of clocks
θ	:	history

operation:

$t : l \xrightarrow{g,s,r,u} l', \rho, \mu, \nu$ a transition

$$\frac{g(\mu,\nu) \quad \textit{JoinEnabled}(\rho,\mu,\nu,l) \quad \textit{Inv}(\rho^{\mathcal{T}_t},\nu^{\mathcal{T}_t}) \quad \neg\textit{EXIT}(l')}{(\rho,\mu,\nu,\theta) \xrightarrow{t} \mathcal{T}_t(\rho,\mu,\nu,\theta)}\ \textit{action}$$

Ingredients for the Semantic Rules

$$\textit{JoinEnabled}(\rho,\mu,\nu,S) := \textit{BASIC}(S) \vee \exists E \in \textit{PreExitSets}(S).\ \forall b \in \textit{Leaves}(\rho,S).\ \exists b' \in E.\ b \xrightarrow{g} b' \ \wedge\ g(\mu,\nu)$$

$$\textit{PreExitSets}(l) := \begin{cases} \left.\begin{array}{l} \displaystyle\bigcup_{n_1,\ldots,n_k} \boxtimes_{1\le i\le k} \textit{PreExitSets}(n_i), \text{ where} \\ \quad k = |\tilde{}\delta(\delta^{-1}(l))|,\ \{n_1,\ldots,n_k\} \subseteq \delta^{\times}(\delta^{-1}(l)), \\ \quad \forall i.\textit{EXIT}(n_i) \ \wedge\ n_i \rightarrow l \in T \\ \quad \{\delta^{-1}(n_1),\ldots,\delta^{-1}(n_k)\} = \tilde{}\delta(l) \end{array}\right\} & \text{if } \begin{array}{l}\textit{EXIT}(l)\wedge \\ \textit{AND}(\delta^{-1}(l))\end{array} \\ \left.\begin{array}{l} \displaystyle\bigcup_{m\in\delta(\delta^{-1}(l))} \textit{PreExitSets}(m), \text{ where } m \xrightarrow{g,r} l \in T \\ \quad \cup \{\{l\}\} \end{array}\right\} & \text{if } \begin{array}{l}\textit{EXIT}(l)\wedge \\ \textit{XOR}(\delta^{-1}(l))\end{array} \\ \{\} & \text{if } \textit{BASIC}(l) \end{cases}$$

This Formalizm is

1) hierarchical
2) timed
3) decidable

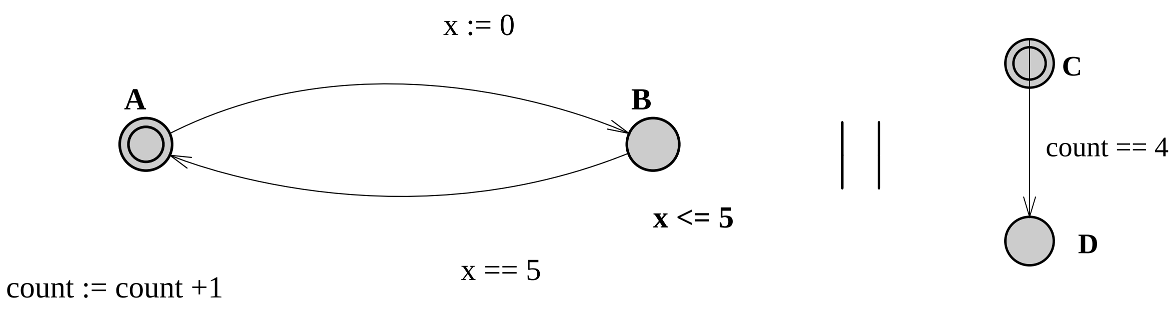
Model Checking

$$\boxed{M \overset{?}{\models} \varphi}$$

M	:	description of the system
φ	:	desired (correctness) property

- easier than proving a general theorem
- completely automatic ('yes' or counterexample)
- *efficient* algorithms tailored for classes of problems

Real-Time Model Checking with UPPAAL



network of (flat) timed automata

```
clock x; int count
```

Only a subset of timed computation tree logic (TCTL) supported:

E<> φ	reachability
A[] φ	safety (invariantly φ)
E[] φ	possibly always φ
A<> φ	inevitably φ
A[] φ $\Rightarrow$ A<> ψ	unbounded response

φ, ψ : propositional formula over locations and (existing) clocks

Outline of the Flattening

Basically:

one superstate $\longrightarrow$ one (parallel) automaton

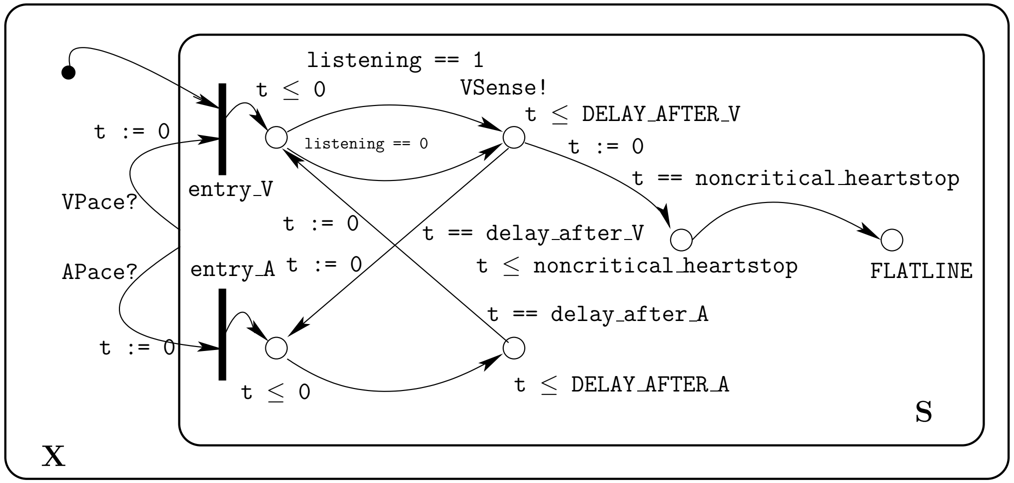
$+$ some housekeeping

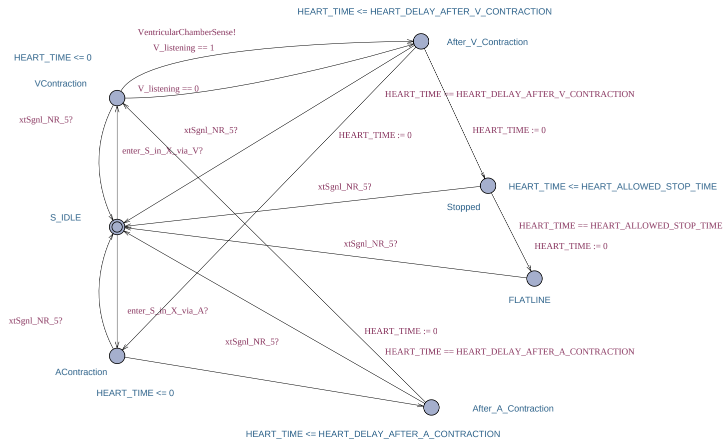
Problems:

- template mechanism
- scope of channels
- pre-compuation of all possible global joins

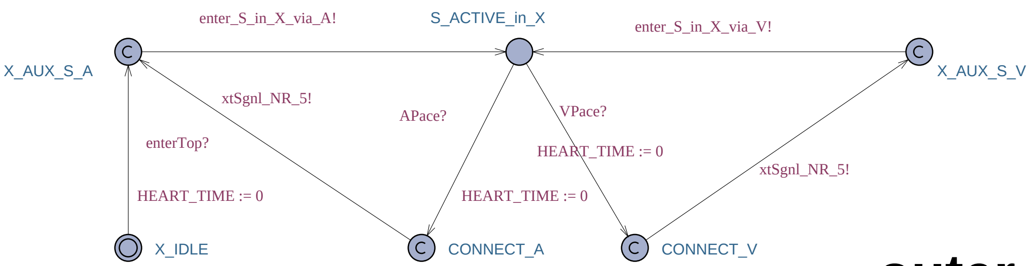
$\approx$ **10˙000 lines of documented Java code**

Example: Flattening the Model of a Human Heart





inner superstate



outer superstate

Soundness & Correctness

Translations introduce slack. Thus

$$\boxed{M_H} \models \varphi \quad \not\Leftrightarrow \quad \boxed{\textit{flatten}(M_H)} \models \textit{flatten}(\varphi)$$

but

$$M_H \models \varphi \quad \Leftrightarrow \quad \textit{flatten}(M_H) \models_{project(M)} \textit{flatten}(\varphi)$$

timed transition system		timed *flatten*(M) traces
$\downarrow$ give rise to		$\downarrow$ project to M_H
timed M_H traces	*match*	timed M_H traces

From (timed) Statecharts to UPPAAL

Rhapsody timed Statechart	$\longrightarrow$	*HTA model* $\boxed{M_H}$	$\longrightarrow$	*TA model* $\boxed{\mathit{flatten}(M_H)}$
hierarchical model		TA-close hierarchy		MODEL-CHECK
informal description		formal semantics		formal semantics

From (timed) Statecharts to UPPAAL

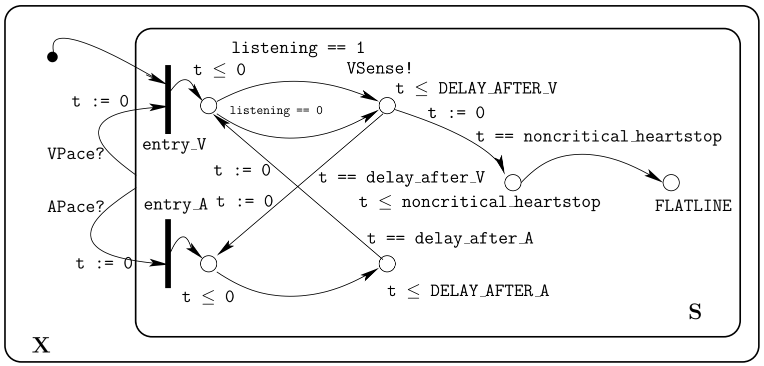
Rhapsody timed Statechart	$\longrightarrow$	*HTA model* M_H	$\longrightarrow$	*TA model* *flatten*(M_H)
hierarchical model		TA-close hierarchy		MODEL-CHECK
informal description		formal semantics		formal semantics

NORMALIZATION
simplification of data
(safe) omission of **C**$^{++}$ code

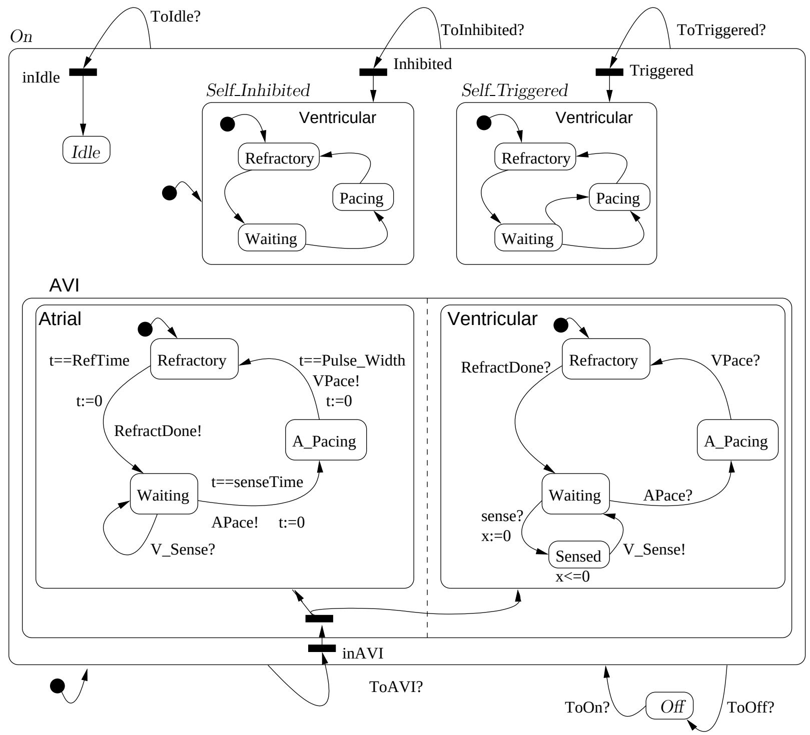
FLATTENING
auxiliary locations
auxiliary variables

Guiding Principle: *Make it easy to adjust to small changes!*

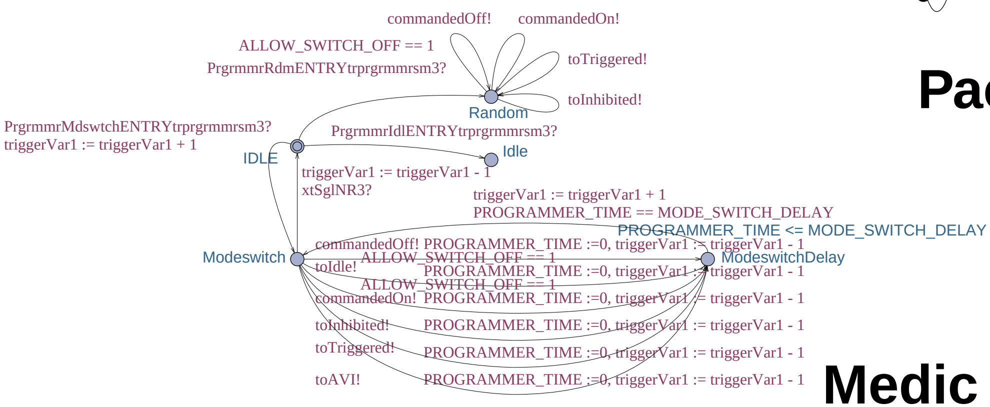
Model Checking a Pacemaker



Human Heart



Pacemaker



Medic

Model-Checking the Pacemaker

- DEADLOCK:
  possible (if heart stops)
- SAFETY:
  `A[]` $\neg$`heart stops`
  only true for 'good' medic
- LIVENESS:
  ```
  A[] ( Vcontract
        => A<> Acontract )
  ```

Model-Checking the Pacemaker

- DEADLOCK:
  possible (if heart stops)
- SAFETY:
  `A[] ¬heart stops`
  only true for 'good' medic
- LIVENESS:
  ```
  A[] ( Vcontract
          => A<> Acontract )
  ```

Parameters:

```
REFRACTORY_TIME   = 50
SENSE_TIMEOUT     = 15

DELAY_AFTER_V = 50
DELAY_AFTER_A =  5

HEART_ALLOWED_STOP_TIME = 135

MODE_SWITCH_DELAY  = 66
```

E.g. for `MODE_SWITCH_DELAY  = 65`, `A[] ¬heart stops` is violated

Summary on Hierarchical Timed Automata

The Major Gains:

- for **modeler**:
  more flexible and compact modeling (than with flat automata)
- for **development process**:
  intermediate format for automated analysis of design models
  (requires typically an abstraction step)

Future Work:

- put to use in AIT-WOODDES project
  RHAPSODY UML statecharts to be model-checked via UPPAAL
- to be integrated in the UPPAAL tool
  - → UPPAAL timed automata are a special case of HTAs
  - → editor for XML grammar is work in progress
  - → model checking engine for HTAs planned

In the Following (II)

Chapter 3: Hierarchical Timed Automata

1. Restricted Statecharts with Real-Time
2. A Trace-Based Semantics
3. Flattening and Correspondence
4. Case Study: Cardiac Pacemaker

Chapter 7: Predicate Abstraction for Dense Real-Time

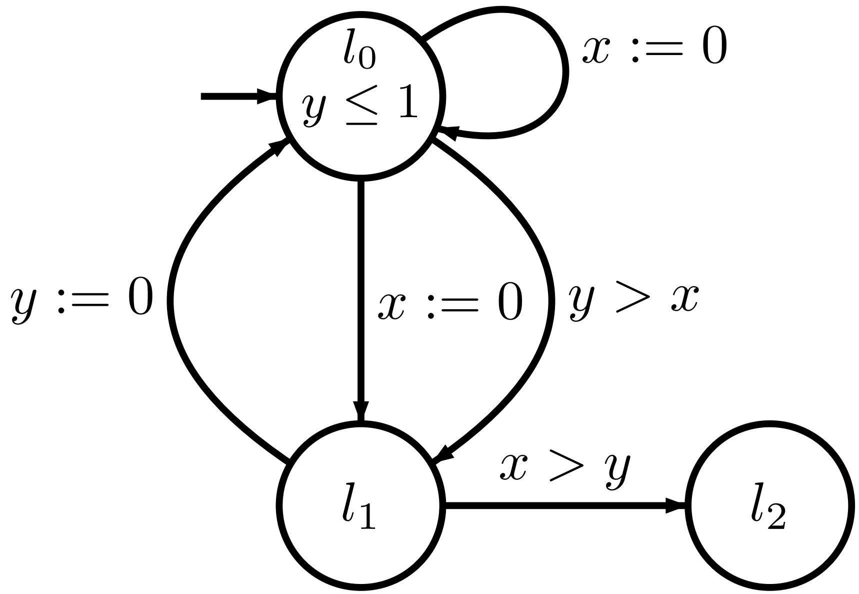
5. Galois Connection to an Untimed Model
6. Progress Assumption by Restricting Delays
7. Successive Refinement

Summary

Timed Systems

Timing constraints Γ, propositional Symbols A

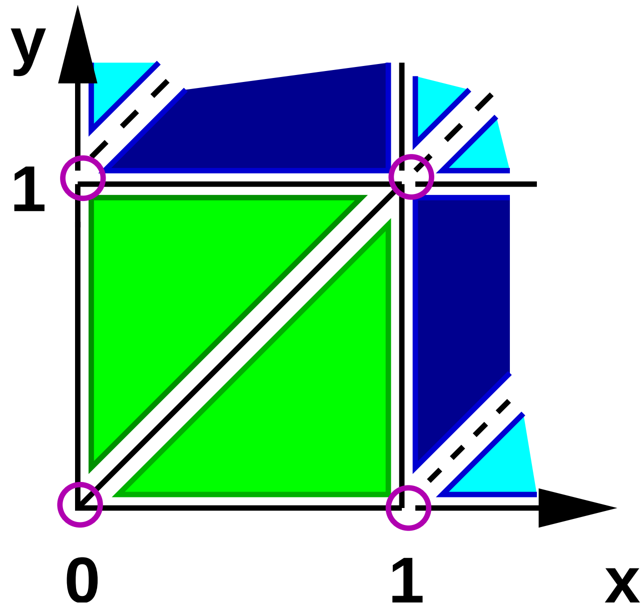
Timed System $\mathcal{S} = \langle L, P, C, \rightarrow, l_0, I \rangle$



Semantics as transition system $\mathcal{M} = \langle L \times \mathcal{V}_C, P, \Rightarrow, (l_0, \nu_0) \rangle$ with *non-zenoness assumption:*

if trace infinite, sum over all delays is ∞

Clock Regions



- Given: $\mathcal{S}$, C, $\tilde{c}$
- Finite partition of the infinite state space
- Clock region: $\mathcal{X}C \subseteq \mathcal{V}_C$ s.t. for all $\chi \in \mathit{Constr}(c)$ and for any two $\nu, \nu' \in \mathcal{X}C$ it is the case that $\nu \models\!\approx \chi$ if and only if $\nu' \models\!\approx \chi$
- $\nu_1 \equiv_{\mathcal{S}} \nu_2$

Propositional Next-Free μ-Calculus

Syntax:

$$\varphi := tt \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists\,(\varphi_1 U \varphi_2) \mid \forall\,(\varphi_1 U \varphi_2) \mid Z \mid \mu Z.\varphi$$

Semantics: $[\![\varphi]\!]^{\mathcal{M}}_{\vartheta}$ . . . set of states for which φ holds

Intuitively, an existential (strong) until formula $\exists\,(\varphi_1 U \varphi_2)$ holds in some states $\mathbf{s}$ iff φ_1 holds on some path from $\mathbf{s}$ until φ_2 holds.

$$[\![\exists\,(\varphi_1 U \varphi_2)]\!]^{\mathcal{M}}_{\vartheta} \stackrel{\text{def}}{=}$$

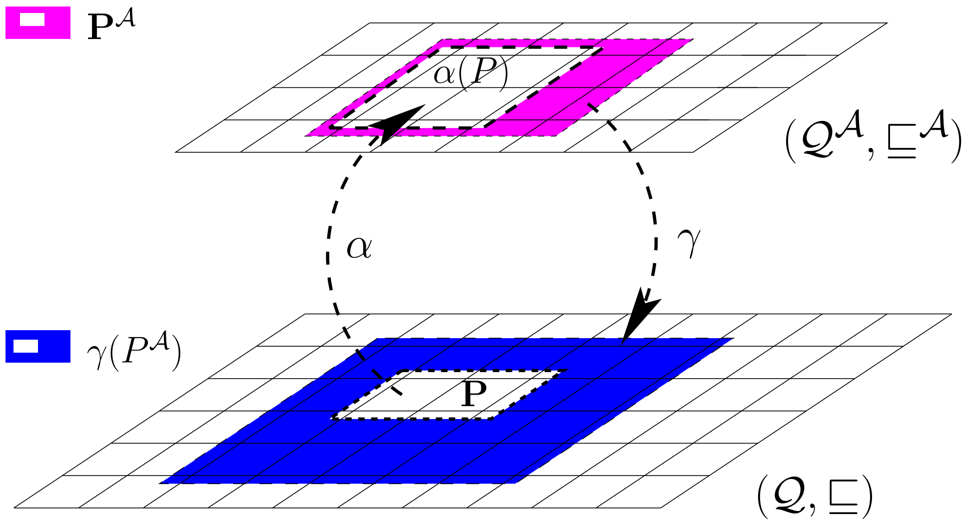
$$\{s_0 \in S \mid \text{there exists a path } \tau = (s_0 \Rightarrow s_1 \Rightarrow \ldots),\ \text{s.t. } s_i \in [\![\varphi_2]\!]^{\mathcal{M}}_{\vartheta}$$

$$\text{for some } i \geq 0,\ \text{and for all } 0 \leq j < i,\ s_j \in [\![\varphi_1]\!]^{\mathcal{M}}_{\vartheta}\}$$

State-Based Model Checking

- Semantics of formula φ $:=$ the set of configurations satisfying φ
- Model checking problem: $l_0 \stackrel{?}{\in} [\![\varphi]\!]^{\mathcal{M}} \rightarrow$ **Yes/No**
- Finite quotient for timed systems: *region construction*
- Our approach: successive refinements of finite *approximations*

Abstract Interpretation: Galois Connections



$(\mathcal{Q}^{\mathcal{A}}, \sqsubseteq^{\mathcal{A}})$ abstract system

$(\mathcal{Q}, \sqsubseteq)$ concrete system

$\alpha : \mathcal{Q} \rightarrow \mathcal{Q}^{\mathcal{A}}$ abstraction

$\gamma : \mathcal{Q}^{\mathcal{A}} \rightarrow \mathcal{Q}$ concretization

$$\boxed{\alpha(P) \sqsubseteq^{\mathcal{A}} P^{\mathcal{A}} \quad \Leftrightarrow \quad P \sqsubseteq \gamma(P^{\mathcal{A}})}$$

Essence: connection of 2 lattice structures

Problems: stability and self-loops

Predicate Abstraction of Timed Systems

Abstraction Predicates

- formula over clocks in C
  E.g.: $x - y \leq 3$, $x^2 - y^2 = 3.1415$,
- partition the (uncountable) state space with respect to their truth value
- set of abstractions predicates $\Psi = \{\psi_0, \ldots, \psi_{n-1}\}$

Abstraction function

- $\alpha : \mathcal{V}_C \rightarrow B_n$
- $\alpha(\nu)(i) := \psi_i \nu$

Concretization function

- $\gamma : B_n \rightarrow \wp(\mathcal{V}_C)$
- $\gamma(b) := \{\nu \in \mathcal{V}_C \mid \bigwedge_{i=0}^{n-1} \psi_i \nu \equiv b(i)\}$

Predicate Abstraction of Timed Systems

Abstraction Predicates

- formula over clocks in C
  E.g.: $x - y \leq 3$, $x^2 - y^2 = 3.1415$,
- partition the (uncountable) state space with respect to their truth value
- set of abstractions predicates $\Psi = \{\psi_0, \ldots, \psi_{n-1}\}$

Abstraction function

- $\alpha : L \times \mathcal{V}_C \rightarrow L \times B_n$
- $\alpha(l,\nu)(i) := (l,\psi_i \nu)$

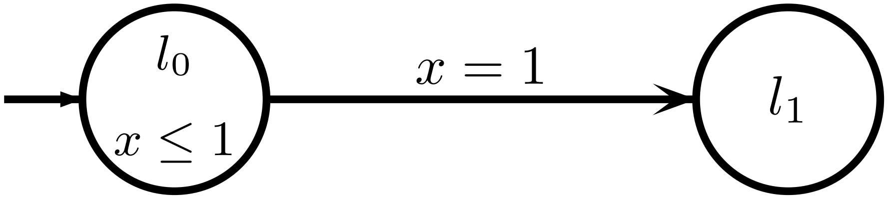
Concretization function

- $\gamma : L \times B_n \rightarrow L \times \wp(\mathcal{V}_C)$
- $\gamma(l,b) := \{\nu \in \mathcal{V}_C \mid \bigwedge_{i=0}^{n-1} \psi_i \nu \equiv b(i) \wedge I(l)\}$

Predicate Abstracted Semantics

$$
\begin{array}{rcl}
[\![tt]\!]_{\vartheta}^{\mathcal{M}_{\Psi}^{\sigma}} & := & S^{A} \\
[\![p]\!]_{\vartheta}^{\mathcal{M}_{\Psi}^{\sigma}} & := & \{(l,b) \in S^{A} \mid p \in P(l)\} \\
[\![\varphi_1 \wedge \varphi_2]\!]_{\vartheta}^{\mathcal{M}_{\Psi}^{\sigma}} & := & [\![\varphi_1]\!]_{\vartheta}^{\mathcal{M}_{\Psi}^{\sigma}} \;\cap\; [\![\varphi_2]\!]_{\vartheta}^{\mathcal{M}_{\Psi}^{\sigma}} \\
[\![\neg\varphi]\!]_{\vartheta}^{\mathcal{M}_{\Psi}^{\sigma}} & := & S^{A} \setminus [\![\varphi]\!]_{\vartheta}^{\mathcal{M}_{\Psi}^{\bar{\sigma}}} \\
[\![\exists\,(\varphi_1 U \varphi_2)]\!]_{\vartheta}^{\mathcal{M}_{\Psi}^{\sigma}} & := & \{s_0 \in S^{A} \mid \text{there exists a path } \tau = (s_0 \Rightarrow^{\sigma} s_1 \Rightarrow^{\sigma} s_1 \ldots), \\
 & & \text{s.t. } s_i \in [\![\varphi_2]\!]_{\vartheta}^{\mathcal{M}_{\Psi}^{\sigma}} \text{ for some } i \geq 0, \text{ and} \\
 & & \text{for all } 0 \leq j < i,\, s_j \in [\![\varphi_1]\!]_{\vartheta}^{\mathcal{M}_{\Psi}^{\sigma}} \\
[\![\forall\,(\varphi_1 U \varphi_2)]\!]_{\vartheta}^{\mathcal{M}_{\Psi}^{\sigma}} & := & \{s_0 \in S^{A} \mid \text{for every path } \tau = (s_0 \Rightarrow^{\bar{\sigma}} s_1 \Rightarrow^{\bar{\sigma}} \ldots), \\
 & & \text{there exists } i \geq 0 \text{ s.t. } s_i \in [\![\varphi_2]\!]_{\vartheta}^{\mathcal{M}_{\Psi}^{\sigma}}, \text{ and} \\
 & & \text{for all } 0 \leq j < i,\, s_j \in [\![\varphi_1]\!]_{\vartheta}^{\mathcal{M}_{\Psi}^{\sigma}}\} \\
[\![Z]\!]_{\vartheta}^{\mathcal{M}_{\Psi}^{\sigma}} & := & \vartheta(Z) \\
[\![\mu Z.\varphi]\!]_{\vartheta}^{\mathcal{M}_{\Psi}^{\sigma}} & := & \cap\{S' \in S^{A} \mid [\![\varphi]\!]_{\vartheta[Z:=S']}^{\mathcal{M}_{\Psi}^{\sigma}} \subseteq S'\}
\end{array}
$$

Example for Abstraction



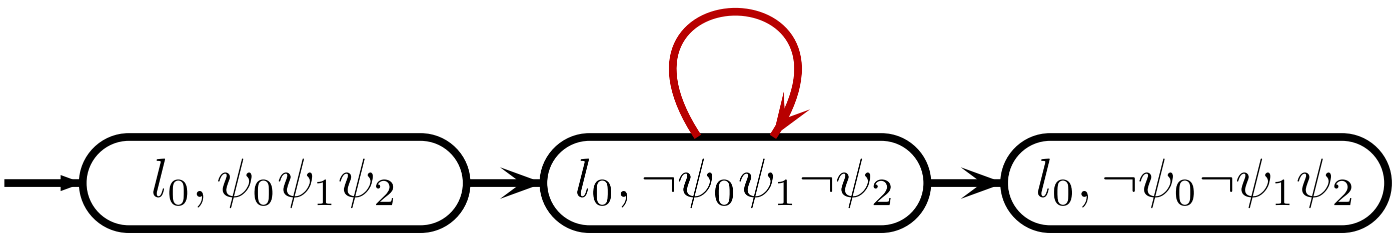
We want to verify: $\varphi = \forall\,(tt\; U\; at\_l_1)$

Abstraction predicates: $\{x = 0, x < 1, x = 1\}$

Assume the following sequence in the concrete trace:

$$(l_0, x = 0) \stackrel{1/2}{\Rightarrow} (l_0, x = 1/2) \stackrel{1/4}{\Rightarrow} (l_0, x = 3/4) \stackrel{1/4}{\Rightarrow} (l_0, x = 1) \stackrel{\mathbf{true}}{\Rightarrow} (l_1, x = 1)$$

Abstraction yields (only a fragment is illustrated):



Problem: spurious self-loop

Modified Semantics: Restricted Delay Step

Given: $\mathcal{S}$,C, $\tilde{c}$

A delay step $(l,\nu) \xrightarrow{\delta} (l,(\nu+\delta))$ is a restricted delay step iff it crosses the border of the current clock region:

$$\boxed{\exists x \in C.\, \exists k \in \{0,\ldots,c\}.\; \nu(x) = k \;\vee\; (\nu(x) < k \wedge \nu(x) + \delta \geq k)}$$

Restricted transition relation: $\Rightarrow_R \subseteq (L, \mathcal{V}_C) \times (L, \mathcal{V}_C)$

The second delay step in the previous trace is disallowed:

$$(l_0, x=0) \Rightarrow_R (l_0, x=1/2) \not\Rightarrow_R (l_0, x=3/4) \Rightarrow_R (l_0, x=1) \Rightarrow_R (l_1, x=1)$$

Theorem:

$$[\![\varphi]\!]_\vartheta^{\mathcal{M}} \;=\; [\![\varphi]\!]_\vartheta^{\mathcal{M}_\mathcal{R}}$$

Abstraction is Sound & Complete

Given: $\mathcal{M} = \langle S^C, P, \Rightarrow, s_0^C \rangle$ a transition system

Ψ a set of predicates

$\mathcal{M}_\Psi^+, \mathcal{M}_\Psi^-$ the over-/under-approximations

Theorem: $\gamma(\llbracket \varphi \rrbracket^{\mathcal{M}_\Psi^-}) \subseteq \llbracket \varphi \rrbracket^{\mathcal{M}} \subseteq \gamma(\llbracket \varphi \rrbracket^{\mathcal{M}_\Psi^+})$

Theorem:

If $(\forall \psi \in \Psi.\ \psi\nu_1 \Leftrightarrow \psi\nu_2) \Rightarrow \nu_1 \equiv_{\mathcal{S}} \nu_2$

Then $\llbracket \varphi \rrbracket_\vartheta^{\mathcal{M}_\Psi^-} = \llbracket \varphi \rrbracket_\vartheta^{\mathcal{M}_\Psi^+}$

Refinement of the Abstraction

- Basis $\widehat{\Psi}$: the "exact" abstract transition system can be computed
  Not practicable
- Successive approximation of the abstract transition relation:

Algorithm: *refine_approximation*

INPUT $\mathcal{M}$, $\widehat{\Psi}$, φ

CHOOSE $\Psi \subseteq \widehat{\Psi}$

WHILE $l_0 \notin [\![\varphi]\!]^{\mathcal{M}^-_\Psi}$ /⋆ YES ⋆/

$\wedge \quad l_0 \notin [\![\neg\varphi]\!]^{\mathcal{M}^-_\Psi}$ /⋆ NO ⋆/

CHOOSE $\psi \in \widehat{\Psi} \setminus \Psi$

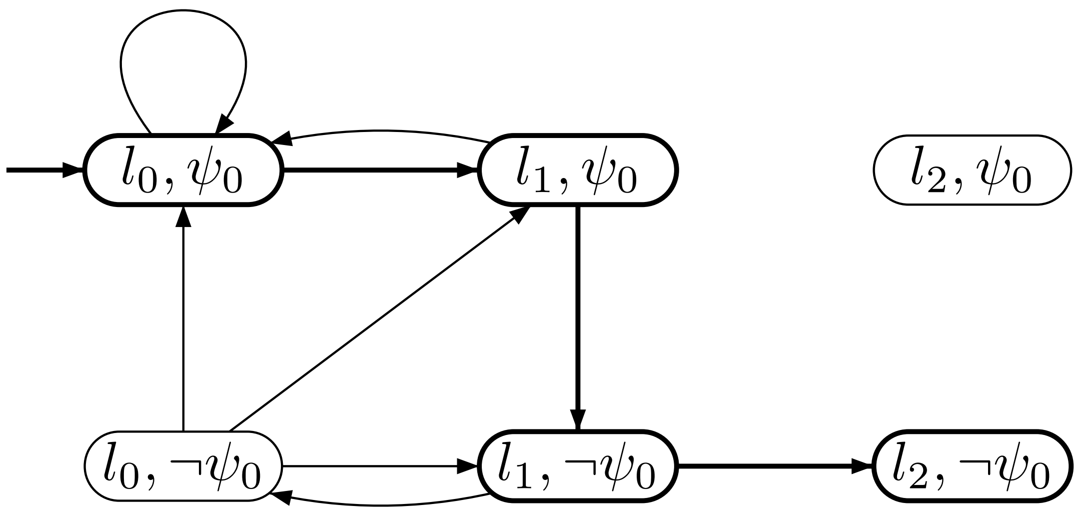
$\Psi := \Psi \cup \{\psi\}$

Example (Refinement)

$$\varphi := \neg\exists\,(tt\,U\,at\_l_2)$$

$$\Psi := \{x = 0, y = 0, x \leq 1, x \geq 1, y \leq 1, y \geq 1, x > y, x < y\}$$

I. $\psi_0 \equiv x = 0$



$\mathcal{M}^{+}_{\{x=0\}} \overset{?}{\models} \varphi$ NO

$$\tau = \big(\ (l_0, \psi_0) \Rightarrow^{+} (l_1, \psi_0) \Rightarrow^{+} (l_1, \neg\psi_0) \Rightarrow^{+} (l_2, \neg\psi_0)\ \big)$$

Example – Continuation I.

$$\tau = (\underbrace{(l_0, \psi_0)}_{s_0} \Rightarrow^+ \underbrace{(l_1, \psi_0)}_{s_1} \Rightarrow^+ \underbrace{(l_1, \neg\psi_0)}_{s_2} \Rightarrow^+ \underbrace{(l_2, \neg\psi_0)}_{s_3})$$

Is there a corresponding counterexample on the concrete system?

$\exists\, \tau^c = (y_0 \Rightarrow y_1 \Rightarrow y_2 \Rightarrow y_3)$ s.t.

$y_0 \in \gamma(s_0),\ y_1 \in \gamma(s_1),\ y_2 \in \gamma(s_2),\ y_3 \in \gamma(s_3),\ y_0 = s_0^c$

$$\begin{aligned} F := \quad & y_0 \in \gamma(s_0) \ \wedge\ y_1 \in \gamma(s_1) \ \wedge\ y_2 \in \gamma(s_2) \ \wedge\ y_3 \in \gamma(s_3) \ \wedge \\ & y_1 \Rightarrow y_2 \ \wedge\ y_2 \Rightarrow y_3 \ \wedge\ y_0 = s_0^c \end{aligned}$$

Is F satisfiable?

Example – Continuation II.

Here F is unsatisfiable!

$$\begin{array}{lcr}
y_0 \in & (l_0, x = y = 0) & \in \gamma(s_0) \\
& \Downarrow & \\
y_1 \in & (l_1, x = 0 \wedge 0 \le y \le 1) & \in \gamma(s_1) \\
& \Downarrow & \\
y_2 \in & (l_1, x > 0 \wedge y > x) & \in \gamma(s_2) \\
& \not\Downarrow & \\
y_3 \in & (l_1, x > 0 \wedge y \ge 0) & = \gamma(s_3)
\end{array}$$

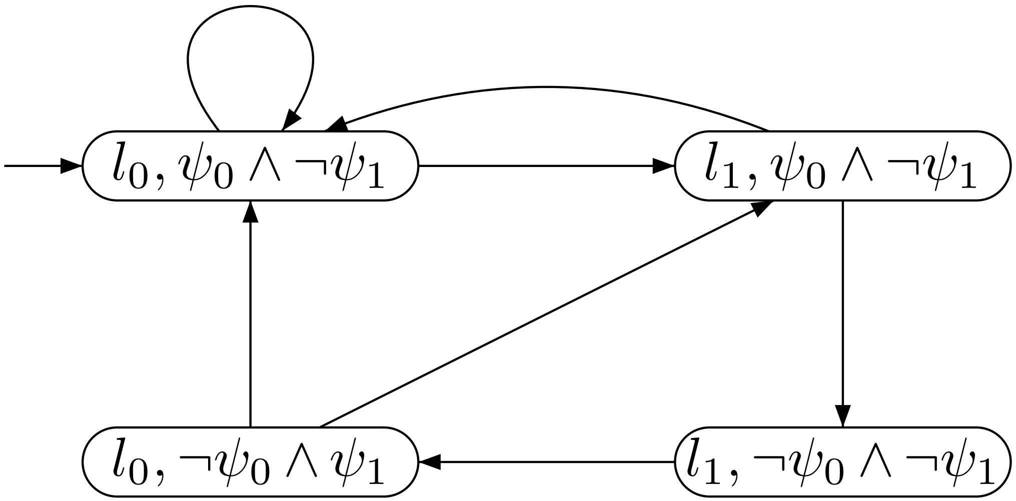
Choose $\psi_1 \in \Psi$ s.t. $\forall\, y \in \gamma(s_k),\ y' \in \gamma(s_{k+1}).\, y \not\Rightarrow y'$

Here: $k = 2$ $\psi_1 \equiv x > y$

Example – Continuation III.

New approximation $\mathcal{M}^{+}_{\{x=0,x>y\}}$

Satisfies formula $\varphi = \neg\exists\,(tt\,U\,at\_l_2)$



Algorithm terminates with **true**

$(l_0, x = y = 0) \;\in\; [\![\neg\exists\,(tt\,U\,at\_l_2)]\!]^{\mathcal{M}}$

Summary on Predicate Abstraction for Real-Time

What can be verified?

- Safety (known before)
- Liveness (!)

Observations:

- self-loops problem:
  solved by restricting the delay steps in *concrete* system
- logic is un-timed and *without next*
- a weaker assumption than non-zenoness suffices
  (only restrict infinite sequences of delay steps)

Summarizing...

Part I: Modeling of Real-Time Systems

1. The unified modeling language (UML) and statecharts (overview)
2. The language of UPPAAL (trace-based semantics)
3. Hierarchical timed automata [NWPT'01,FASE'02,journal submission]

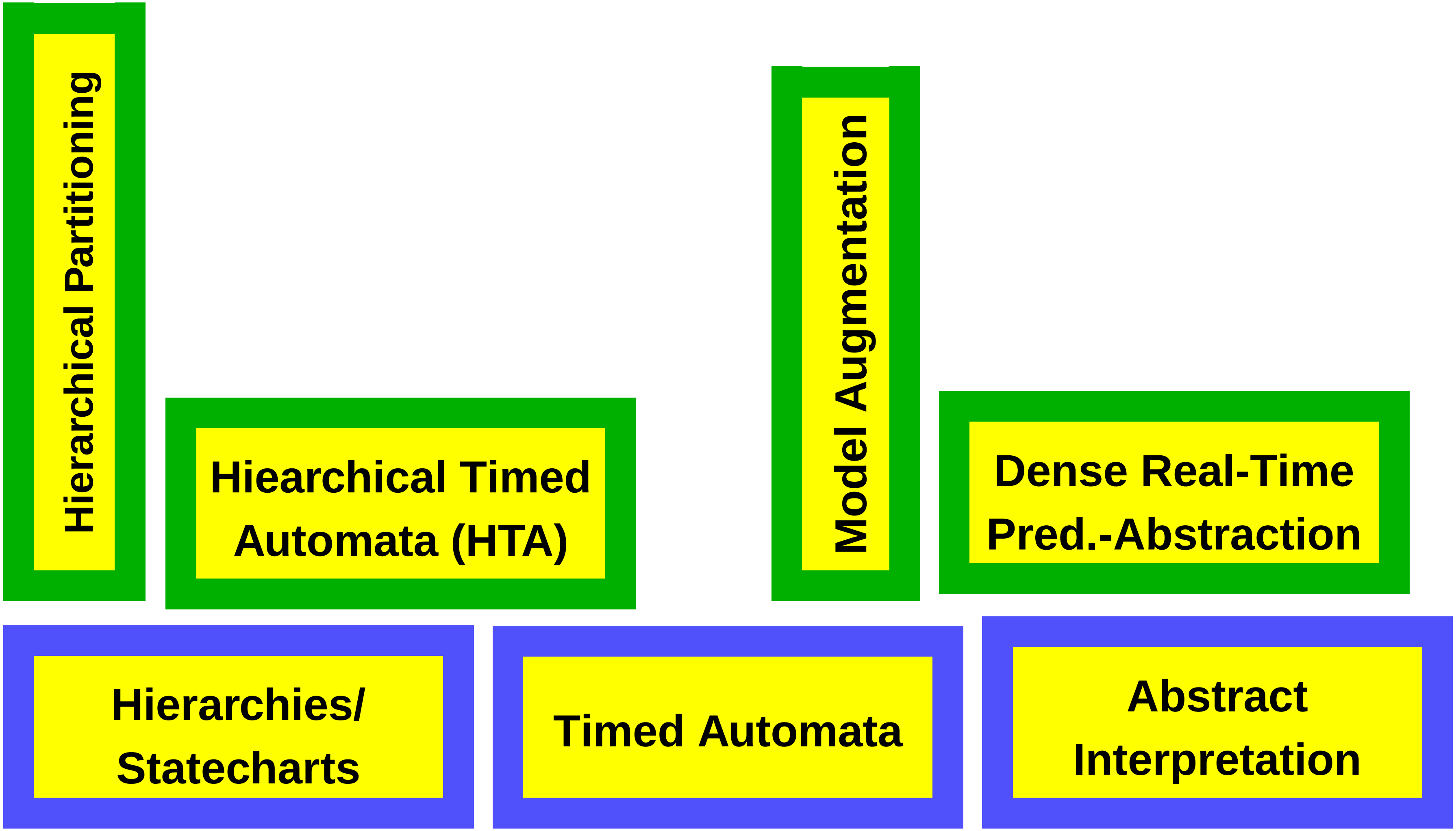
Part II: Algorithmic Verification of Real-Time Systems

4. Real-time model checking: forward analysis (correctness formalization)
5. Optimization techniques for real-time systems (benchmarks)
6. Model augmentation to speed-up model checking [TPTS'01]
7. Predicate abstraction for dense real-time [TPTS'01]

Part III: Making Use of Hierarchical Structure

8. Construction of good hierarchies from parallel components [CHARME'01]
9. Flattening hierarchical timed automata for model checking [NWPT'01,FASE'02,journal submission]

What was Known, What was New?



What was Known, What was New?

Hierarchical Partitioning

Event Encoding

Trace Semantics

UPPAAL Benchmarking

Model Augmentation

Hiearchical Timed Automata (HTA)

Dense Real-Time Pred.-Abstraction

Hierarchies/ Statecharts

Timed Automata

Abstract Interpretation

What is (still) to be Done

- Model checking **engine** for HTAs
  - → future work of Alexandre David, Uppsala University
- Exploiting HTAs via **re-use**
  - → similar to re-use in *modecharts* (Rajeev Alur et al.)
    similar to CBR technique in VISUALSTATE (Gerd Behrmann et al.)
    time gives rise to difficulties
- Sound abstraction step from **UML statecharts** to the HTA formalism
  - → Clearly requires *approximation* of data and events
- Implement the successive refinement idea for timed automata

Conclusion – on Real-Time Systems

- **Hierarchies** complicate—but do not hinder—the formal analysis; whether they also can be exploited remains to be seen

- Fully automated analysis is expensive but often **feasible** for reasonably sized models
  (which implies that formal methods should be applied *a priori*)

- Predominant efficiency gain is via *abstractions*;
  Techniques that **approximate** timed systems can go *beyond safety*

Outline of the Thesis

Part I: Modeling of Real-Time Systems

1. The unified modeling language (UML) and statecharts (overview)
2. The language of UPPAAL (trace-based semantics)
3. Hierarchical timed automata [NWPT'01,FASE'02,journal submission]

Part II: Algorithmic Verification of Real-Time Systems

4. Real-time model checking: forward analysis (correctness formalization)
5. Optimization techniques for real-time systems (benchmarks)
6. Model augmentation to speed-up model checking [TPTS'01]
7. Predicate abstraction for dense real-time [TPTS'01]

Part III: Making Use of Hierarchical Structure

8. Construction of good hierarchies from parallel components [CHARME'01]
9. Flattening hierarchical timed automata for model checking
   [NWPT'01,FASE'02,journal submission]

Bibliography

[ABB+01] Tobias Amnell, Gerd Behrmann, Johan Bengtsson, Pedro R. D'Argenio, Alexandre David, Ansgar Fehnker, Thomas Hune, Bertrand Jeannet, Kim G. Larsen, M. Oliver Möller, Paul Pettersson, Carsten Weise, and Wang Yi. UPPAAL - Now, Next, and Future. In F. Cassez, C. Jard, B. Rozoy, and M. Ryan, editors, *Modelling and Veri£cation of Parallel Processes*, number 2067 in Lecture Notes in Computer Science Tutorial, pages 100–125. Springer–Verlag, 2001.

[ADF+01] Tobias Amnell, Alexandre David, Elena Fersman, M. Oliver Möller, Paul Petterson, and Wang Yi. Tools for Real-Time UML: Formal Veri£cation and Code Synthesis, June 2001. in *Implementation and Validation of Object-oriented Embedded Systems (SIVOES'2001)*, Budapest, Hungary.

[BDL+01a] Gerd Behrman, Alexandre David, Kim G. Larsen, M. Oliver Möller, Paul Petterson, and Wang Yi. UPPAAL - Present and Future. In P. Pettersson and S. Yovine, editors, *Workshop on Real-Time Tools*, August 2001. Proceedings appeared as technical report 2001-014 Uppsala University, Sweden.

[BDL+01b] Gerd Behrman, Alexandre David, Kim G. Larsen, M. Oliver Möller, Paul Petterson, and Wang Yi. UPPAAL - Present and Future. In *Proc. of the 40th IEEE Conference on*

Decision and Control, pages 2281–2286, Orlando, Florida, December 2001. IEEE Service Center.

[DM01] Alexandre David and M. Oliver Möller. From HUPPAAL to UPPAAL: A Translation from Hierarchical Timed Automata to Flat Timed Automata. Research Series RS-01-11, BRICS, Department of Computer Science, University of Aarhus, March 2001.

[DMY01] Alexandre David, M. Oliver Möller, and Wang Yi. Formal Veri£cation of UML Statecharts with Real-Time Extensions. In M. R. Hansen, editor, *The 13th Nordic Workshop on Programming Theory (NWPT'01),* appeared as technical report of the Technical University of Denmark, *IMM-TR-2001-12*, October 2001.

[DMY02] Alexandre David, M. Oliver Möller, and Wang Yi. Formal Veri£cation of UML Statecharts with Real-Time Extensions. to appear in Fundamental Approaches to Software Engineering (FASE'2002), 2002.

[MA00] M. Oliver Möller and Rajeev Alur. Heuristics for Hierarchical Partitioning with Application to Model Checking. Research Series RS-00-21, BRICS, Department of Computer Science, University of Aarhus, August 2000. 30 pp, available online at `http://www.brics.dk/RS/00/21/`.

[MA01] M. Oliver Möller and Rajeev Alur. Heuristics for Hierarchical Partitioning with Application to Model Checking. In T. Margaria and T. Melham, editors, *Correct Hardware Design and Veri£cation Methods, 11th IFIP WG 10.5 Advanced Research Working Conference, CHARME 2001, Livingston, Scotland, UK*, volume 2144 of

Lecture Notes in Computer Science (LNCS), pages 71–85, New York, NY, USA, September 2001. Springer–Verlag.

[Möl02] M. Oliver Möller. Parking Can Get You There Faster - Model Augmentation to Speed up Real-Time Model-Checking. to appear in Theory and Practice of Timed Systems (TPTS'2002), 2002.

[MRS01] M. Oliver Möller, Harald Rueß, and Maria Sorea. Predicate Abstraction for Dense Real-Time Systems. Research Series RS-01-44, BRICS, Department of Computer Science, University of Aarhus, November 2001.

[MRS02] M. Oliver Möller, Harald Rueß, and Maria Sorea. Predicate Abstraction for Dense Real-Time Systems. to appear in Theory and Practice of Timed Systems (TPTS'2002), 2002.