Hybrid Discrete-Continuous Modeling

Alexander Bockmayr
FU Berlin & Matheon

Ecole Jeunes Chercheurs
Ile de Porquerolles
10 June 2010
Outline

• Hybrid systems
• Formalization
• Reachability
• Linear hybrid automata
• Bisimulation
• Timed automata
1. Hybrid systems
1. Hybrid dynamical systems …

• **Dynamical system**: Describe the evolution of a state over time.
  – Continuous: \( \dot{x} = f(x, t) \)
  – Discrete: \( x_{k+1} = f(x_k, t_k) \)
  – Linear vs. non-linear

• **Hybrid dynamical system**: Interacting
  – continuous-time dynamics (modeled, e.g., by differential equations), and
  – discrete-event dynamics (modeled, e.g., by automata).
in biology

• Continuous dynamics to describe temporal/spatial variations of molecule concentrations inside a cell.
• Discrete switches, encoding molecule concentrations reaching certain thresholds, to trigger activation or deactivation of these continuous dynamics.
• Examples:
  – Caulobacter cell division
  – Delta-Notch signaling
  …
Hybrid automaton

- Discrete states $q \in Q$, $Q$ finite
- Continuous states $x \in \mathbb{R}^n$
- Initial states $\text{Init} \subseteq Q \times \mathbb{R}^n$
- Continuous dynamics $\dot{x} = f(q, x)$
- Discrete dynamics $R: Q \times \mathbb{R}^n \rightarrow 2^Q \times \mathbb{R}^n$
- Invariant $\text{Inv} \subseteq Q \times \mathbb{R}^n$ (defines states for which continuous evolution is allowed).
Example 1: Bouncing ball

- $x_1$: vertical position
- $x_2$: velocity
- $g$: acceleration due to gravity
- $c \in [0,1]$: coefficient of restitution
- continuous changes between bounces
- discrete changes at bounce times
Hybrid automaton

\[ x_1 = 0 \land x_2 \leq 0 \]

\[ \dot{x}_1 = x_2 \]
\[ \dot{x}_2 = -g \]
\[ x_1 \geq 0 \]

\[ x_2 := -cx_2 \]
Some properties

- non blocking: from any initial condition there exists at least one trajectory.
- $x_1 \geq 0$ is an invariant.
- for $c < 0$, the hybrid automaton model is Zeno (infinite number of discrete transitions in finite time).
Example 2: Thermostat

• Temperature in a room, \( x \), is controlled by switching a heater on and off.

• Thermostat regulates \( x \) around 75°
  – turn the radiator on when the temperature is between 68° and 70°,
  – turn the radiator off, when the temperature is between 80° and 82°
Hybrid automaton

- Cannot go from Off to On unless $x \leq 70$.
- Must go from Off to On if $x \leq 68$.

non-deterministic
Multiple executions
2. Formalization
Hybrid automaton

An autonomous hybrid automaton is a collection $H = (Q, X, \text{Init}, f, \text{Dom}, R)$, with

- $Q = \{q_1, q_2, ..., q_s\}$ a set of discrete states
- $X \subseteq \mathbb{R}^n$ a set of continuous states
- $\text{Init} \subseteq Q \times X$ a set of initial states
- $f(\cdot,\cdot): Q \times X \rightarrow \mathbb{R}^n$ a vector field
- $\text{Dom}(\cdot): Q \rightarrow 2^X$ a domain
- $R(\cdot,\cdot): Q \times X \rightarrow 2^{Q \times X}$ a reset relation
Intuitive behavior

• Start from an initial state \((q_0, x_0) \in \text{Init.}\).
• Continuous state flows according to differential equation \(\dot{x} = f(q_0, x), x(0) = x_0\), discrete state remains at \(q_0\).
• Continuous evolution can go as long as \(x(t) \in \text{Dom}(q_0)\).
• If at some point, \(R(q_0, x) \neq \emptyset\), a discrete transition can take place.
• During the discrete transition, the continuous and the discrete state may be reset to some value in \(R(q_0, x)\).
Hybrid time sets

Hybrid time set: Finite or infinite sequence of intervals

\[ \tau = \{I_i\}_{i=0}^{N}, \quad N \in \mathbb{N} \cup \{\infty\} \]

such that

- \( I_i = [\tau_i, \tau_i'] \), for all \( i < N \)
- if \( N < \infty \) then, either \( I_N = [\tau_N, \tau_N'] \) or \( I_N = [\tau_N, \tau_N'] \)
- \( \tau_i \leq \tau_i' = \tau_{i+1} \), for all \( i < N \).

Denote by \( \mathcal{T} \) the set of all hybrid time sets.
Example
Length of hybrid time sets

• Discrete extent: $<.>: \mathcal{T} \rightarrow \mathbb{N} \cup \{\infty\}$ with

$$\langle \tau \rangle = \begin{cases} 
\mathbb{N}, & \text{if } \tau \text{ is finite,} \\
\infty, & \text{if } \tau \text{ is infinite.}
\end{cases}$$

• Continuous extent: $\|\cdot\|: \mathcal{T} \rightarrow \mathbb{R}_+$ such that

for $\tau = \{I_i\}_{i=0}^{N} \in \mathcal{T}$

$$\|\tau\| = \sum_{i=0}^{N} (\tau'_i - \tau_i)$$
Classification of hybrid time sets

- \( \tau \) finite: \(<\tau>\) is finite and the last interval in \( \tau \) is closed.
- \( \tau \) finite-open: \(<\tau>\) is finite and the last interval in \( \tau \) is bounded and right open.
- \( \tau \) infinite: \(<\tau> = \infty \) or \( ||\tau|| = \infty \).
- \( \tau \) Zeno: \(<\tau>\) is infinite, but \( ||\tau|| < \infty \).
Example

\[ \tau_A \text{ finite, } \tau_B \text{ finite-open, } \]

\[ \tau_C, \tau_D \text{ infinite, } \tau_E, \tau_F \text{ Zeno} \]
Hybrid trajectory

A hybrid trajectory \((\tau, q, x)\) consists of

- a hybrid time set \(\tau = \{I_i\}_0^N \in \mathcal{T}\)
- a sequence of functions \(q = \{q_i(.)\}_0^N\), with \(q_i(.) : I_i \to Q\)
- a sequence of functions \(x = \{x_i(.)\}_0^N\), with \(x_i(.) : I_i \to \mathbb{R}^n\)
Execution

An execution of a hybrid automaton H is a hybrid trajectory \((\tau, q, x)\) which satisfies:

• initial condition: \((q_0(\tau_0), x_0(\tau_0)) \in \text{Init}\)
• discrete evolution:
  \[(q_{i+1}(\tau_{i+1}), x_{i+1}(\tau_{i+1})) \in R(q_i(\tau'_i), x_i(\tau'_i))\]
• continuous evolution: for all \(i\)
  1. \(q_i(\cdot): I_i \rightarrow Q\) is constant over \(t \in I_i\)
  2. \(x_i(\cdot): I_i \rightarrow X\) is solution to \(\dot{x}_i = f(q_i(t), x_i(t))\)
  over \(I_i\), starting at \(x_i(\tau_i)\)
  3. for all \(t \in [\tau_i, \tau'_i[\), \(x_i(t) \in \text{Dom}(q_i(t))\)
Remarks

- The reset relation $R$ is enabling discrete transitions: the execution may take a transition from a state $(q,x)$ as long as $R(q,x) \neq \emptyset$.
- The domain $\text{Dom}$ is forcing transitions: the execution must take a transition if the state is about to leave the domain.
- A hybrid automaton may accept multiple executions for a given initial state ($\rightarrow$ model uncertainty).
- Note that we do not require $x_i(\tau_i') \in \text{Dom}(q_i(t))$, i.e., executions may leave the domain for an instant.
Graphical representation

Associate with the automaton

\[ H = (Q, X, \text{Init}, f, \text{Dom}, R) \]

a directed graph \( G = (Q, E) \) with

\[ E = \{(q,q') \in Q \times Q \mid (q',x') \in R(q,x), \text{ for some } x, x' \in X \} \]

For each \( q \in Q \):

- \( \text{Init}_q = \{x \in X \mid (q,x) \in \text{Init}\} \)
- \( f_q : X \rightarrow \mathbb{R}^n, f_q(x) = f(q,x) \)
- \( \text{Dom}_q = \{x \in X \mid (q,x) \in \text{Dom}\} \)
For each \((q,q') \in E:\)

- \(\text{Guard}_{(q,q')} = \{ x \in X \mid (q',x') \in R(q,x), \text{ for some } x' \in X \}\)
- \(\text{Reset}_{(q,q')} (x) = \{ x' \in X \mid (q',x') \in R(q,x) \}\)
Example: Bouncing ball

\begin{align}
x_1 &= 0 \land \\
x_2 &\leq 0
\end{align}

\dot{x}_1 &= x_2 \\
\dot{x}_2 &= -g \\
x_1 &\geq 0 \\
q_0
Hybrid automaton

\[ Q = \{q_0\} \]
\[ x = (x_1, x_2) \in \mathbb{R}^2 \]
\[ \text{Init} = \{q_0\} \times \{x \in \mathbb{R}^2 \mid x_1 \geq 0\} \]
\[ \dot{x} = f(q_0, x) = (x_2, -g) \]
\[ \text{Inv} = \{q_0\} \times \{x \in \mathbb{R}^2 \mid x_1 \geq 0\} \]
\[ R(q_0, \{x \mid x_1 = 0 \land x_2 \leq 0\}) = \{(q_0, (x_1, -c x_2))\}\]
Example: Thermostat

\[ x \in X_T \]

- **on**
  \[ \dot{x} = -x + 100 \]
  \[ x \leq 82 \]
  \[ x := x \]

- **off**
  \[ \dot{x} = -x \]
  \[ x \geq 68 \]
  \[ x \leq 70 \]
  \[ x := x \]
Hybrid automaton

- \( Q = \{\text{on, off}\} \), \( X = \mathbb{R} \)
- \( \text{Init} = X \)
- \( f(\text{on}, x) = -x + 100, \ f(\text{off}, x) = -x \)
- \( \text{Dom}(\text{on}) = \{x \in X \mid x \leq 82\} \), \( \text{Dom}(\text{off}) = \{x \in X \mid x \geq 68\} \)
- \( R(\text{on}, x) = \{(\text{off}, x)\}, \text{ if } x \geq 80, \)
  \( R(\text{off}, x) = \{(\text{on}, x)\}, \text{ if } x \leq 70, \)
  \( R(q, x) = \emptyset, \text{ otherwise.} \)
3. Reachability
Transition systems

A transition system is a collection

\[ T = (S, \Sigma, \rightarrow, S_0, S_F), \]

where

- \( S \) is a set of states
- \( \Sigma \) is an alphabet of events
- \( \rightarrow: S \times \Sigma \rightarrow 2^S \) is a transition relation
- \( S_0 \subseteq S \) is a set of initial states
- \( S_F \subseteq S \) is a set of final states
Example

Autonomous hybrid automaton

\[ H = (Q, X, \text{Init}, f, \text{Dom}, R) \]

with \( S = Q \times X \), \( \Sigma = E \cup \{C\} \)

(\( E \) is set of edges in the graphical representation)

\( \rightarrow = \) discrete transitions \( \cup \) continuous evolution,

\( S_0 = \text{Init}, \ S_F = U \) (set of unsafe states).
Reachability problem

Given a transition system $T$, is there a state $s_f \in S_F$ that is reachable from a state $s_0 \in S_0$ by a finite sequence of transitions?

- decidable for finite automata
- undecidable in general
Predecessors and successors

• Pre: $2^S \rightarrow 2^S$
  \[
  \text{Pre}(A) = \{ s \in S \mid \exists s' \in A, \exists \sigma \in \Sigma \text{ such that } (s, \sigma) \rightarrow s' \} 
  \]

• Post: $2^S \rightarrow 2^S$
  \[
  \text{Post}(A) = \{ s' \in S \mid \exists s \in A, \exists \sigma \in \Sigma \text{ such that } (s, \sigma) \rightarrow s' \} 
  \]
Forward vs. backward analysis
Practical realization

- Manipulate sets of states $A$
- Compute $\text{Pre}(A)$, $\text{Post}(A)$
- Take union and intersections of sets of states
- Check whether $A$ is empty
- Check whether two sets of states are equal

Not obvious for infinite sets of states!
4. Linear hybrid automata
A linear inequality is a formula
\[ a_1 x_1 + \ldots + a_n x_n \leq b, \]
with rational numbers \( a_1, \ldots, a_n, b \) and real variables \( x_1, \ldots, x_n \).

A convex linear predicate is a finite conjunction of linear inequalities.

A linear predicate is a finite disjunction of convex linear predicates.
A linear hybrid automaton is an hybrid automaton such that

- For every discrete state $q \in Q$, the sets $\text{Init}(q)$ and $\text{Dom}(q)$ are defined by convex linear predicates over the variables $X$.

- For every $q \in Q$, the continuous dynamics is defined by a convex linear predicate $\text{flow}(q)$ over the variables $\dot{X}$ specifying a differential inclusion, e.g. $\dot{x}_1 \leq 3 \land x_1 \leq \dot{x}_2$. 
Linear hybrid automata (contd)

- For every edge $e \in E$, the reset condition $R(e)$ is given by a linear predicate over the variables $X \cup X'$, which relates the values of the variables before a discrete transition to the possible values after the switch.

**Example:** $x_1 = x_2 \rightarrow x_1 := 2x_2$ corresponds to the linear predicate

$$x_1 = x_2 \land x'_1 = 2x_2 \land x'_2 = x_2.$$
State assertions

• A state assertion $\phi$ of the hybrid automaton $H$ is a function that assigns to each $q \in Q$ a predicate $\Phi(q)$ over the variables in $X$.

• For a state assertion $\phi$, let $\text{Post}(\Phi)$ be a state assertion that is true precisely for those states $s'$ that are reachable from a state $s$ satisfying $\Phi$ by one continuous or one discrete transition.

• A state assertion is linear if the predicate $\Phi(q)$ is linear, for all $q \in Q$. 
Reachability

• Theorem
If $H$ is a linear hybrid automaton and $\Phi$ is a linear state assertion, then $\text{Post}(\Phi)$ is computable, and itself a linear state assertion.

• Corollary
Given a linear state assertion $\psi$, the reachability problem

\textit{Is there a reachable state $s$ satisfying $\psi$?}

is semidecidable.
Reachability computation

• Basic structure
  (Discrete state \( q \), Polyhedron \( P \))
• Set of visited states: list of \((q, P)\) pairs
• Key steps:
  – Compute “discrete” successors of \((q, P)\).
  – Compute “continuous” successors of \((q, P)\).
  – Check if \( P \) intersects with the region given by \( \psi \).
  – Check if newly found \( P \) is covered by already visited polyhedra \( P_1, \ldots, P_k \) (expensive).
Computing discrete successors

Discrete successor of \((q,P)\)
• Intersect \(P\) with \(g\) (result \(R\) is a polyhedron)
• Apply linear transformation \(A\) to \(R\) (result is a polyhedron \(R'\)).
• Successor is \((q',R')\)
Computing continuous successors

Theorem
If $\phi$ and $\text{flow}(q)$ are convex linear predicates, the set of continuous successors of $\phi$ can be defined by a convex linear predicate.
Hytech Tool

• Symbolic model checker for linear hybrid automata

• Developed by T.A. Henzinger, P.-H. Ho, and H. Wong-Toi, UC Berkeley, 1995 - 2003

• Web site: http://embedded.eecs.berkeley.edu/research/hytech/
5. Bisimulation
Motivation

• Turn infinite state systems to finite state systems by grouping together states that have “similar” behavior.
• Partition the state space into finitely many equivalence classes such that equivalent states exhibit similar behaviors.
Quotient transition system

• $T = (S, \Sigma, \rightarrow, S_0, S_F)$ transition system
• $\sim$ equivalence relation on $S$

Quotient transition system

$T/\sim = (S/\sim, \Sigma, \rightarrow_{\sim}, S_0/\sim, S_F/\sim)$

where for $S_1, S_2 \in S/\sim$, $(S_1, \sigma) \rightarrow_{\sim} S_2$ if and only if there exists $s_1 \in S_1$ and $s_2 \in S_2$ such that $(s_1, \sigma) \rightarrow s_2$. 
Bisimulation

- $T = (S, \Sigma, \rightarrow, S_0, S_F)$ transition system
- $\sim$ equivalence relation on $S$

$\sim$ is a bisimulation if

1. $(s_1 \sim s_2) \land (s_1 \in S_0) \Rightarrow (s_2 \in S_0)$
2. $(s_1 \sim s_2) \land (s_1 \in S_F) \Rightarrow (s_2 \in S_F)$
3. $(s_1 \sim s_2) \land ((s_1, \sigma) \rightarrow s'_1) \Rightarrow$
   
   $\exists s'_2 [(s'_1 \sim s'_2) \land ((s_2, \sigma) \rightarrow s'_2)]$
Bisimulation and reachability

Theorem

If $\sim$ is a bisimulation of a transition system $T$ and $T/\sim$ is the quotient system, then $S_F$ is reachable by $T$ if and only if $S_F/\sim$ is reachable by $T/\sim$.

reason on $T/\sim$ instead of $T$; of particular interest if $T$ is infinite, but admits a finite bisimulation $T/\sim$. 
6. Timed automata
Clock constraints

• $C = \{x_1, \ldots, x_n\}$ finite set of real variables, called clocks.

• The set $\Phi(C)$ of clock constraints for $C$ is the set of logical formulas defined by the grammar

$$\delta ::= (x_i \leq c) \mid (x_i \geq c) \mid \neg \delta \mid \delta \land \delta,$$

where $x_i \in C$ and $c \geq 0$ is a rational number.

• For $\delta \in \Phi(C)$, let $\hat{\delta} = \{v \in \mathbb{R}^n \mid \delta(v) \equiv \text{true}\}$
Examples

• $C = \{x_1, x_2\}$
• $(x_1 \leq 1) \in \Phi(C)$
• $(0 \leq x_1 \leq 1) \in \Phi(C)$, since
  $(0 \leq x_1 \leq 1) \iff (x_1 \geq 0) \land (x_1 \leq 1)$
• $(x_1 = 1) \in \Phi(C)$, since
  $(x_1 = 1) \iff (x_1 \geq 1) \land (x_1 \leq 1)$
• $(x_1 < 1) \in \Phi(C)$, since
  $(x_1 < 1) \iff (x_1 \leq 1) \land \neg (x_1 \geq 1)$
• $(x_1 \leq x_2) \notin \Phi(C)$
Timed automaton

A **timed automaton** is a hybrid automaton $H = (Q, X, \text{Init}, f, \text{Dom}, R)$ and a set of clocks $C$, where

- $Q$ is a set of discrete variables, $Q = \{q_1, \ldots, q_m\}$
- $C = \{x_1, \ldots, x_n\}$, $X = \mathbb{R}^n$
- $\text{Init}(.): Q \to \Phi(C)$, i.e., $\text{Init} = \{\{q_i\} \times \delta_i\}_{i=1}^m$, for some $\delta_i \in \Phi(C)$
- $f(q, x) = (1, \ldots, 1)$, for all $q \in Q$
- $\text{Dom}: Q \to \Phi(C)$
- $R: Q \times \Phi(C) \to Q \times X$, where $R(\cdot, \cdot)$ either leaves $x_i$ unaffected or resets it to 0.
Example

\[ x_1 := 0 \]
\[ x_2 := 0 \]

\[ q_1 \]
\[ \dot{x}_1 = 1 \]
\[ \dot{x}_2 = 1 \]
\[ x \geq 0 \]

\[ q_2 \]
\[ \dot{x}_1 = 1 \]
\[ \dot{x}_2 = 1 \]
\[ x \geq 0 \]

\[ x_1 \leq 3, x_2 \leq 2 \]

\[ x_1 := 0, x_2 := x_2 \]

\[ x_1 \leq 1 \]

\[ x_1 := x_1, x_2 := x_2 \]
Example

- $Q = \{q_1, q_2\}$
- $C = \{x_1, x_2\}$, $X = \mathbb{R}^2$
- Init = $\{(q_1,0,0)\}$
- $f(q,x) = (1,1)$, for all $q \in Q$
- $\text{Dom}(q) = \mathbb{R}^2_{\geq 2}$, for all $q \in Q$
- $E = \{(q_1,q_2), (q_2,q_1)\}$
- $G(q_1,q_2) = \{x \in \mathbb{R}^2 \mid (x_1 \leq 3) \land (x_2 \geq 2)\}$,
  $G(q_2,q_1) = \{x \in \mathbb{R}^2 \mid (x_1 \leq 1) \}$
- $R(q_1,x_1,x_2) = (q_2,0,x_2)$, $R(q_2,x_1,x_2) = (q_1,x_1,x_2)$
Transition system

• Discrete transition
\[(q,v) \rightarrow^e (q´,v´) \text{ iff } e = (q_1,q_2) \in E \text{ and } (q´,v´) \in R(q,v)\]

• Time-abSTRACTED continuous transition
\[(q,v) \rightarrow^\tau (q´,v´) \text{ iff } q = q´ \text{ and there exists } t \geq 0 \text{ such that } v´ = v + t(1,\ldots,1).\]
Notation

• Assume all constants are integer
• Denote by $\lfloor v_i \rfloor$ the integer part of $v_i$, and by $\langle v_i \rangle$ the fractional part
• Let $c_i$ be the largest constant with which $x_i$ is compared (in an invariant or guard).
Region equivalence

- Identify two states with the same discrete part if they agree on the integral part for all clocks and on the ordering of the fractional part for all clocks.
- Integral part needed to determine whether or not a particular clock constraint is met.
- Fractional part needed to decide which clock will change its integral part first.
- Integral parts of clocks can get arbitrarily large, but if a clock $x$ is never compared with a constant greater than $c$, then its value, once it exceeds $c$, is of no consequence in deciding allowed transitions.
Formal definition

\((q,v) \sim (q',v')\) if

1. \(q = q'\)

2. For all \(i\):
   \(|v_i| = |v'_i|\) or \(|v_i| > c_i\) and \(|v'_i| > c_i\)

3. For all \(i, j\) with \(v_i \leq c_i\) and \(v_j \leq c_j\)
   
   \[\langle v_i \rangle \leq \langle v_j \rangle \iff \langle v_i' \rangle \leq \langle v_j' \rangle\]

4. For all \(i\) with \(v_i \leq c_i\)
   
   \[\langle v_i \rangle = 0 \iff \langle v_i' \rangle = 0\]

Proposition

\(\sim\) is an equivalence relation.
Example

c_1 = 3, c_2 = 2

Equivalence classes

points
open lines
open sets

Number of equivalence classes is finite!
Reachability

• Reachability questions for timed automata can be answered on a finite state system, defined as the quotient space of \( \sim \)

\[ \text{region automaton} \]

• Theorem (Alur/Dill 94)
  The reachability problem is decidable for timed automata.
Complexity

- Up to $m(n!) \prod_{i=1}^{n} (2c_i + 2)$ states in the region automaton.
- Need not necessarily construct full automaton.
- Reachability problem can be shown to be PSPACE complete.
Further result

Theorem

CTL and LTL model checking is decidable for timed automata (provided every proposition occurring in temporal formulas is either a discrete state or a rectangular set.)
References


Temporal constraints in the logical analysis of regulatory networks

Alexander Bockmayr

Joint work with Heike Siebert

DFG-Research Center MATHEON, Freie Universität Berlin

Ile de Porquerolles, 10 June 2010
Outline

1. Logical analysis
2. Temporal constraints on time delays
3. Biological applications
Outline

1. Logical analysis
2. Temporal constraints on time delays
3. Biological applications
Logical analysis of regulatory networks

Thomas’73, Snoussi’89

**Structure:** Interaction Graph
Logical analysis of regulatory networks
Thomas’73, Snoussi’89

**Structure:** Interaction Graph

- Genes $\alpha_1, \ldots, \alpha_n$.

\[ \alpha_1 \quad \alpha_2 \]
Logical analysis of regulatory networks
Thomas’73, Snoussi’89

**Structure:** Interaction Graph

- Genes $\alpha_1, \ldots, \alpha_n$.
- Activating or inhibiting interactions.
Logical analysis of regulatory networks

Thomas’73, Snoussi’89

**Structure:** Interaction Graph

- Genes $\alpha_1, \ldots \alpha_n$.
- Activating or inhibiting interactions.
- Expression levels necessary for edge activity.
Logical analysis of regulatory networks
Thomas’73, Snoussi’89

**Structure:** Interaction Graph

- Genes $\alpha_1, \ldots, \alpha_n$.
- Activating or inhibiting interactions.
- Expression levels necessary for edge activity.
- Expression levels $0, \ldots, p_j$ associated with each $\alpha_j$.

Expression levels of $\alpha_1$: $0, 1$
Expression levels of $\alpha_2$: $0, 1, 2$
Logical analysis of regulatory networks
Thomas’73, Snoussi’89

**Dynamics:** State Space and Parameters
Logical analysis of regulatory networks

Thomas’73, Snoussi’89

**Dynamics:** State Space and Parameters

- State space
  \[ S^n := \{0, \ldots, p_1\} \times \cdots \times \{0, \ldots, p_n\} \]

\[ S^2 = \{0, 1\} \times \{0, 1, 2\} \]
Logical analysis of regulatory networks

Thomas’73, Snoussi’89

Dynamics: State Space and Parameters

- State space
  \[ S^n := \{0, \ldots, p_1\} \times \cdots \times \{0, \ldots, p_n\} \]
- Parameter values \( K_{\alpha_i, \omega} \in \{0, \ldots, p_i\} \)

\[ S^2 = \{0, 1\} \times \{0, 1, 2\} \]

\[
\begin{align*}
K_{\alpha_1, \emptyset} &= 0 \\
K_{\alpha_1, \{\alpha_1\}} &= 0 \\
K_{\alpha_1, \{\alpha_2\}} &= 1 \\
K_{\alpha_1, \{\alpha_1, \alpha_2\}} &= 1 \\
K_{\alpha_2, \emptyset} &= 0 \\
K_{\alpha_2, \{\alpha_1\}} &= 1 \\
K_{\alpha_2, \{\alpha_2\}} &= 0 \\
K_{\alpha_2, \{\alpha_1, \alpha_2\}} &= 2
\end{align*}
\]
Logical analysis of regulatory networks

Thomas’73, Snoussi’89

**Dynamics:** State Space and Parameters

- State space
  \[ S^n := \{0, \ldots, p_1\} \times \cdots \times \{0, \ldots, p_n\} \]
- Parameter values \( K_{\alpha_i, \omega} \in \{0, \ldots, p_i\} \)
  - Depend on set of predecessors \( \omega \) contributing to an activating input of \( \alpha_i \)
  - Specify the value to which \( \alpha_i \) tends to evolve.
Logical analysis of regulatory networks

Thomas’73, Snoussi’89

Dynamics: State Space and Parameters

- **State space**
  \[ S^n := \{0,\ldots,p_1\} \times \cdots \times \{0,\ldots,p_n\} \]

- **Parameter values** \( K_{\alpha_i,\omega} \in \{0,\ldots,p_i\} \)
  - Depend on set of predecessors \( \omega \) contributing to an activating input of \( \alpha_i \)
  - Specify the value to which \( \alpha_i \) tends to evolve.

\[ S^2 = \{0,1\} \times \{0,1,2\} \]

\[
\begin{align*}
K_{\alpha_1,\emptyset} &= 0 \\
K_{\alpha_1,\{\alpha_1\}} &= 0 \\
K_{\alpha_1,\{\alpha_2\}} &= 1 \\
K_{\alpha_1,\{\alpha_1,\alpha_2\}} &= 1 \\
K_{\alpha_2,\emptyset} &= 0 \\
K_{\alpha_2,\{\alpha_1\}} &= 1 \\
K_{\alpha_2,\{\alpha_2\}} &= 0 \\
K_{\alpha_2,\{\alpha_1,\alpha_2\}} &= 2
\end{align*}
\]

Example: \((1,1) \mapsto (K_{\alpha_1,\{\alpha_1\}}, K_{\alpha_2,\{\alpha_2\}}) = (0,0)\)
Logical analysis of regulatory networks
Thomas’73, Snoussi’89

**Dynamics:** State Transition Graph
Logical analysis of regulatory networks
Thomas’73, Snoussi’89

**Dynamics**: State Transition Graph

- **Vertex set** $S^n$

  - $(0, 2)$
  - $(1, 2)$
  - $(0, 1)$
  - $(1, 1)$
  - $(0, 0)$
  - $(1, 0)$
Logical analysis of regulatory networks
Thomas’73, Snoussi’89

**Dynamics:** State Transition Graph

- Vertex set $S^n$
- Edges derived from parameter values.
Logical analysis of regulatory networks
Thomas’73, Snoussi’89

**Dynamics:** State Transition Graph

- Vertex set $S^n$
- Edges derived from parameter values.
  - Corresponding components differ at most by 1.
Logical analysis of regulatory networks
Thomas’73, Snoussi’89

**Dynamics: State Transition Graph**

- Vertex set $S^n$
- Edges derived from parameter values.
  - Corresponding components differ at most by 1.
  - States differ in only one component.
Logical analysis of regulatory networks
Thomas’73, Snoussi’89

**Dynamics:** State Transition Graph

- Vertex set $S^n$
- Edges derived from parameter values.
  - Corresponding components differ at most by 1.
  - States differ in only one component.
- Asynchronous update
- Non-determinism

![State Transition Graph]

- $(0, 2) \rightarrow (1, 2)$
- $(0, 1) \rightarrow (1, 1)$
- $(0, 0) \rightarrow (1, 0)$
- $(1, 0) \rightarrow (0, 0)$
- $(1, 1) \rightarrow (0, 1)$
- $(1, 2) \rightarrow (0, 2)$
Logical analysis of regulatory networks
Thomas’73, Snoussi’89

**Dynamics:** State Transition Graph

- Vertex set $S^n$
- Edges derived from parameter values.
  - Corresponding components differ at most by 1.
  - States differ in only one component.
  - Asynchronous update
  - Non-determinism

- Verification of dynamic properties by model checking
  (Bernot/Comet/Richard/Guespin’04)
Outline

1. Logical analysis

2. Temporal constraints on time delays

3. Biological applications
Time delays

Thomas’78

Update “command” for more than one component $\leadsto$ non-determinism

Allow for the possibility of time delay equality.

Temporal constraints
Time delays
Thomas’78

Update “command” for more than one component $\leadsto$ non-determinism

- Compare time delays associated with different processes.
  - Distinguish between genes.

\[
\begin{align*}
(0, 2) & \xrightarrow{\tau_1 < \tau_2} (1, 2) \\
(0, 1) & \xrightarrow{\tau_1 < \tau_2} (1, 1) \\
(0, 0) & \xrightarrow{\tau_1 < \tau_2} (1, 0) \\
(0, 2) & \xleftarrow{\tau_2 < \tau_1} (1, 2) \\
(0, 1) & \xleftarrow{\tau_2 < \tau_1} (1, 1) \\
(0, 0) & \xleftarrow{\tau_2 < \tau_1} (1, 0)
\end{align*}
\]
Time delays
Thomas’78

Update “command” for more than one component $\leadsto$ non-determinism

- Compare time delays associated with different processes.
  - Distinguish between genes.
  - Distinguish between production and decay processes.

![Diagram]

- $\tau_2^- < \tau_1^-$
- $\tau_1^- < \tau_2^-$
- $\tau_2^+ < \tau_1^+$
- $\tau_1^+ < \tau_2^+$
Time delays
Thomas’78

Update “command” for more than one component \(\leadsto\) non-determinism

- Compare time delays associated with different processes.
  - Distinguish between genes.
  - Distinguish between production and decay processes.
  - Take expression levels into account.

\[
\begin{align*}
(0, 2) & \xrightarrow{\tau_1^- < \tau_2^-} (1, 2) \\
(0, 1) & \xrightarrow{\tau_1^- < \tau_2^-} (1, 1) \\
(0, 0) & \xrightarrow{\tau_1^0 < \tau_2^0} (1, 0) \\
(0, 0) & \xrightarrow{\tau_2^0 < \tau_1^0} (1, 0) \\
(0, 1) & \xrightarrow{\tau_1^0 < \tau_2^0} (1, 1) \\
(0, 2) & \xrightarrow{\tau_1^- < \tau_2^-} (1, 2)
\end{align*}
\]
Time delays
Thomas’78

Update “command” for more than one component $\Rightarrow$ non-determinism

- Compare time delays associated with different processes.
  - Distinguish between genes.
  - Distinguish between production and decay processes.
  - Take expression levels into account.
- Allow for the possibility of time delay equality.

\[
\begin{align*}
(0, 2) & \xrightarrow{\tau_1^- < \tau_2^-} (1, 2) \\
(0, 1) & \xrightarrow{\tau_1^- = \tau_2^-} (1, 1) \\
(0, 0) & \xrightarrow{\tau_1^0+ < \tau_2^0+} (1, 0) \\
\end{align*}
\]
Time delays
Thomas’78

Update “command” for more than one component $\leadsto$ non-determinism

- Compare time delays associated with different processes.
  - Distinguish between genes.
  - Distinguish between production and decay processes.
  - Take expression levels into account.
- Allow for the possibility of time delay equality.
- Time constraints get more complicated when following a path.
Constraint-based modeling

- Represent knowledge about system in the form of constraints ("pieces of partial information")
- Make deductions on what is possible and not \( \leadsto \) set of possible behaviors
Constraint-based modeling

- Represent knowledge about system in the form of constraints ("pieces of partial information")
- Make deductions on what is possible and not \( \implies \) set of possible behaviors

Here

1. Model each gene \textit{locally} incorporating information on
   - expression levels,
   - interactions,
   - parameter values,
   - time delays.
Constraint-based modeling

- Represent knowledge about system in the form of constraints ("pieces of partial information")
- Make deductions on what is possible and not \(\leadsto\) set of possible behaviors

Here

1. Model each gene \textit{locally} incorporating information on
   - expression levels,
   - interactions,
   - parameter values,
   - time delays.

2. Combine the gene models to a \textit{global} network model supplying information on
   - the state space,
   - possible state transitions,
   - temporal constraints along a pathway.
Timed automata

Alur/Dill'94

- **Clocks** measure time, progress linear and synchronously, i.e., $\dot{c} = 1$. 

Clock constraints formulated in the grammar

$$\varphi ::= c \leq q \mid c \geq q \mid c < q \mid c > q \mid \varphi_1 \land \varphi_2.$$ 

Timed automata may be represented by directed graphs, where

- vertices (locations) represent states,
- edges represent (discrete) state transitions,
- temporal constraints may be imposed on states and transitions,
- clocks may be reset.
Clocks measure time, progress linear and synchronously, i.e., $\dot{c} = 1$.

Clock constraints formulated in the grammar

$$\varphi ::= c \leq q \mid c \geq q \mid c < q \mid c > q \mid \varphi_1 \land \varphi_2.$$
Clocks measure time, progress linear and synchronously, i.e., $\dot{c} = 1$.

Clock constraints formulated in the grammar

$$\varphi ::= c \leq q \mid c \geq q \mid c < q \mid c > q \mid \varphi_1 \land \varphi_2.$$  

Timed automata may be represented by directed graphs, where

- vertices (locations) represent states,
- edges represent (discrete) state transitions,
Timed automata
Alur/Dill'94

- **Clocks** measure time, progress linear and synchronously, i.e., $\dot{c} = 1$.
- **Clock constraints** formulated in the grammar
  \[
  \varphi ::= c \leq q \mid c \geq q \mid c < q \mid c > q \mid \varphi_1 \land \varphi_2.
  \]
- **Timed automata** may be represented by directed graphs, where
  - vertices (locations) represent **states**,
  - edges represent (discrete) state **transitions**,
  - temporal constraints may be imposed on **states and transitions**,
  - clocks may be reset.

\[\begin{array}{c}
A \\
c_1 \leq q_1 \\
c_1 \geq q_3 \\
c_2 \geq q_2
\end{array} \quad \begin{array}{c}
B \\
c_2 \geq q_1 \\
c_1 := 0
\end{array} \quad \begin{array}{c}
C \\
c_1 \leq q_1 \\
c_1 := 0 \quad c_2 := 0
\end{array}\]
Timed automata
Alur/Dill'94

- **Clocks** measure time, progress linear and synchronously, i.e., $\dot{c} = 1$.
- **Clock constraints** formulated in the grammar
  \[ \varphi ::= c \leq q \mid c \geq q \mid c < q \mid c > q \mid \varphi_1 \land \varphi_2. \]

- **Timed automata** may be represented by directed graphs, where
  - vertices (locations) represent **states**,
  - edges represent (discrete) state **transitions**,
  - **temporal constraints** may be imposed on states and transitions,
  - clocks may be reset.

Special class of **hybrid automata**
Modeling individual genes

\[ \alpha_1 \alpha_2 -, 1 \]

\[ K_{\alpha_1,\{\alpha_2}\}} = 1 \]

\[ K_{\alpha_1,\{\alpha_1,\alpha_2\}} = 1 \]
Modeling individual genes

- Expression levels:

\[ \alpha_1^{0}, \alpha_1^{1}, \alpha_2 \]

\[ K_{\alpha_1,\{\alpha_2\}} = 1 \]

\[ K_{\alpha_1,\{\alpha_1,\alpha_2\}} = 1 \]
Modeling individual genes

- Expression levels: regular vs. intermediate states.

\[ \alpha_1^{0+} \quad \alpha_1^{0} \quad \alpha_1^{-1} \quad \alpha_1^{1} \]

\[ K_{\alpha_1,\{\alpha_2\}} = 1 \]

\[ K_{\alpha_1,\{\alpha_1,\alpha_2\}} = 1 \]
Modeling individual genes

- Expression levels: regular vs. intermediate states.
- Maximal and minimal time delays for expression level change.
- Location changes due to elapse of time.

\[ K_{\alpha_1, \{\alpha_2\}} = 1 \]
\[ K_{\alpha_1, \{\alpha_1, \alpha_2\}} = 1 \]
Modeling individual genes

- Expression levels: regular vs. intermediate states.
- Maximal and minimal time delays for expression level change.
- Location changes due to elapse of time.
- Network interactions (“switch conditions”) can only be evaluated in the network context.

A. Bockmayr (MATHEON/FU Berlin)
Modeling the network ("Product automaton")

- Product locations.
Modeling the network ("Product automaton")

- Product locations.
- Edges specified in gene automata.
Modeling the network ("Product automaton")

- Product locations.
- Edges specified in gene automata.
- Edges due to network interactions, parameters and current state of the system.
Description includes time component.

Consider behavior in agreement with temporal constraints.
Dynamics

- Description includes time component.
- Consider behavior in agreement with temporal constraints.

\[ ((\alpha_0^0, \alpha_0^0), (0, 0)) \]
\[ ((\alpha_0^+, \alpha_0^+), (0, 0)) \]
Dynamics

- Description includes time component.
- Consider behavior in agreement with temporal constraints.

\[ ((\alpha_1^0, \alpha_2^0), (0, 0)) \]

\[ ((\alpha_1^0+, \alpha_2^0+), (0, 0)) \]

\[ t \leq T_1^{0+}, T_2^{0+} \]

\[ ((\alpha_1^0+, \alpha_2^0+), (t, t)) \]
- Description includes time component.
- Consider behavior in agreement with temporal constraints.
- Description includes time component.
- Consider behavior in agreement with temporal constraints.

\[
((\alpha_0^0, \alpha_0^0), (0, 0))
\]
\[
((\alpha_0^+, \alpha_0^+), (0, 0))
\]
\[
\longrightarrow t \leq T_1^{0+}, T_2^{0+}
\]
\[
((\alpha_0^+, \alpha_0^+), (t, t))
\]
\[
\longrightarrow t \geq t_2^{0+}, t \leq T_1^{0+}
\]
\[
((\alpha_1^0, \alpha_1^0), (t, t))
\]
\[
((\alpha_1^+, \alpha_1^+), (0, t))
\]
Description includes time component.

Consider behavior in agreement with temporal constraints.
Analyzing the dynamics

- Dynamics represented by a transition system
  - Infinite due to time component
  - Non-deterministic

Consistency: Thomas' asynchronous semantics can be obtained as a special case.

Formal verification
- Analysis and verification by means of model checking techniques.
- CTL model checking is decidable for timed automata.
- Software for editing, simulating and verification of timed automata available.
- Original implementation in UPPAAL.
- Improved implementation (“harmonized automata”) in PHAVer (by T. Merle).
Analyzing the dynamics

- Dynamics represented by a transition system
  - Infinite due to time component
  - Non-deterministic
- Consistency: Thomas’ asynchronous semantics can be obtained as a special case.
Analyzing the dynamics

- Dynamics represented by a transition system
  - Infinite due to time component
  - Non-deterministic

- Consistency: Thomas’ asynchronous semantics can be obtained as a special case.

- Formal verification
  - Analysis and verification by means of model checking techniques.
  -CTL model checking is decidable for timed automata.
Analyzing the dynamics

- Dynamics represented by a transition system
  - Infinite due to time component
  - Non-deterministic
- Consistency: Thomas’ asynchronous semantics can be obtained as a special case.

- Formal verification
  - Analysis and verification by means of model checking techniques.
  - CTL model checking is decidable for timed automata.
  - Software for editing, simulating and verification of timed automata available.
  - Original implementation in UPPAAL.
  - Improved implementation (“harmonized automata”) in PHAVer (by T. Merle).
Information on time delays can lead to
   ▶ elimination of pathways violating clock constraints,
Refined analysis

- Information on time delays can lead to
  - elimination of pathways violating clock constraints,
Information on time delays can lead to

- elimination of pathways violating clock constraints,
- evaluation of feasibility and stability of behavior.
Refined analysis

- Information on time delays can lead to
  - elimination of pathways violating clock constraints,
  - evaluation of feasibility and stability of behavior.

- The model offers additional information on status of gene activity.

\[ (el(\alpha_1), el(\alpha_2)) = (0, 1) \]
\[ \leadsto (0^+, 1^+), (0^+, 1), (0^+, 1^-), \\
    (0, 1^+), (0, 1), (0, 1^-) \]
Refined analysis

- Information on time delays can lead to
  - elimination of pathways violating clock constraints,
  - evaluation of feasibility and stability of behavior.

- The model offers additional information on status of gene activity.

⇒ A much more detailed analysis of the network dynamics becomes possible.
Outline

1. Logical analysis

2. Temporal constraints on time delays

3. Biological applications
Bacteriophage $\lambda$

- Genes: $\alpha_1 \xrightarrow{cI}$, $\alpha_2 \xrightarrow{cro}$
- Lytic response: Logical cycle $01 \leftrightarrow 02$
- Lysogenic response: Stable state 10
Dynamic behavior

\[ T_i^{k_+} = T_i^{k_-} = 10 \text{ and } t_i^{k_+} = t_i^{k_-} = 8 \]

\[ T_2^{0_+} = 8 \text{ and } t_2^{0_+} = 6 \]
\[ T_2^{0_+} = 7 \text{ and } t_2^{0_+} = 5 \]
\[ T_1^{0_+} = 7 \text{ and } t_1^{0_+} = 5 \]
Four genes model of bacteriophage λ
Thieffry/Thomas’95

Lytic response: Logical cycle 0200 ↔ 0300
Lysogenic response: Stable state 2000
Dynamic behavior

(cro, $N$ expressed first)

stable and unstable cycle
Conclusion

- Constraint-based modeling: evaluating the consequences of constraints
- Logical analysis of regulatory networks
  - Interaction graph
  - State transition graph
  - Discrete non-deterministic dynamics
- Temporal constraints
  - Constraints on time delays
  - Timed automata
  - Formal verification of properties \( \models \) model checking
- Biological applications
  - Elimination of pathways
  - Feasibility and stability analysis
  - Inference of temporal constraints