On Solving Temporal Logic Constraints in Constrained Transition Systems

François Fages
Joint work with Thierry Martinez, Aurélien Rizk, Sylvain Soliman, Grégory Batt, Calin Belta, Neda Saeedloei

EPI Contraintes
INRIA Paris-Rocquencourt, France

CP meets CAV
28 June 2012

Outline

Motivation: Analysis/Synthesis of Gene/Protein Networks

States and Transitions as Constraints over $\mathbb{R}_{\text{lin}}$

Temporal Logic Constraints over $\mathbb{R}_{\text{lin}}$

Implementation of FOCTL($\mathbb{R}_{\text{lin}}$)

Conclusion

Outline

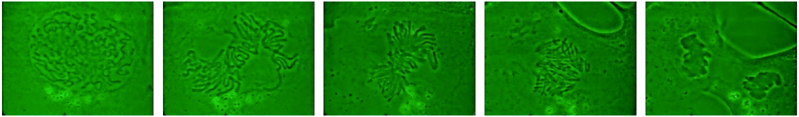
Systems Biology

System-level understanding of high-level cell functions from their biochemical basis at the molecular level [Kitano 2000]

Systems Biology

System-level understanding of high-level cell functions from their biochemical basis at the molecular level [Kitano 2000]

Example: explain the cell cycle control at the molecular level of



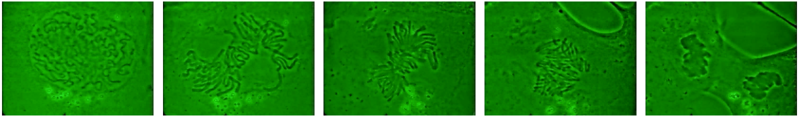
gene transcription, protein degradation and enzymatic reactions

$$\mathsf{S} + \mathsf{E} \underset{k_1}{\overset{k_2}{\leftrightarrows}} \mathsf{ES} \xrightarrow{k_3} \mathsf{E} + \mathsf{P}$$

Systems Biology

System-level understanding of high-level cell functions from their biochemical basis at the molecular level [Kitano 2000]

Example: explain the cell cycle control at the molecular level of



gene transcription, protein degradation and enzymatic reactions

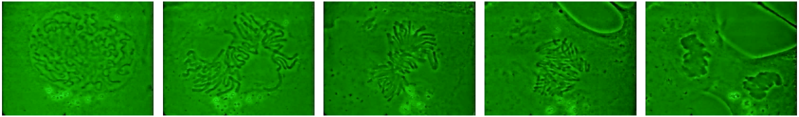
$$S + E \overset{k_2}{\underset{k_1}{\rightleftarrows}} ES \longrightarrow^{k_3} E + P$$

Petri nets ! with reaction rates

Systems Biology

System-level understanding of high-level cell functions from their biochemical basis at the molecular level [Kitano 2000]

Example: explain the cell cycle control at the molecular level of



gene transcription, protein degradation and enzymatic reactions

$$\mathrm{S} + \mathrm{E} \underset{k_1}{\overset{k_2}{\leftrightarrows}} \mathrm{ES} \xrightarrow{k_3} \mathrm{E} + \mathrm{P}$$

Petri nets ! with reaction rates

Ordinary Differential Equations

$$\begin{aligned}
\dot{S} &= -k_1 * S * E + k_2 * ES \\
\dot{P} &= k_3 * ES \\
\dot{E} &= -k_1 * S * E + (k_2 + k_3) * ES \\
\dot{ES} &= k_1 * S * E - (k_2 + k_3) * ES
\end{aligned}$$

Continuous-Time Markov Chain

A Logical Paradigm for Systems and Synthetic Biology

Biological Model = *(Quantitative) State Transition System* K

A Logical Paradigm for Systems and Synthetic Biology

Biological Model = (Quantitative) State Transition System K
Biological Properties = Temporal Logic Formulae ϕ

A Logical Paradigm for Systems and Synthetic Biology

Biological Model = (Quantitative) State Transition System K
Biological Properties = Temporal Logic Formulae ϕ
Automatic Verification = Model-checking $K \models \phi$

A Logical Paradigm for Systems and Synthetic Biology

Biological Model = *(Quantitative) State Transition System* K
Biological Properties = *Temporal Logic Formulae* ϕ
Automatic Verification = *Model-checking* $K \models \phi$
Model Inference = *Constraint Solving* $K' \models \phi'$

A Logical Paradigm for Systems and Synthetic Biology

Biological Model = (Quantitative) State Transition System K
Biological Properties = Temporal Logic Formulae ϕ
Automatic Verification = Model-checking $K \models \phi$
Model Inference = Constraint Solving $K' \models \phi'$

- Expression of high-level specifications
- Model-checking can be efficient on large complex systems
- Temporal logic with numerical constraints can deal with continuous time models (ODE or CTMC, hybrid systems)

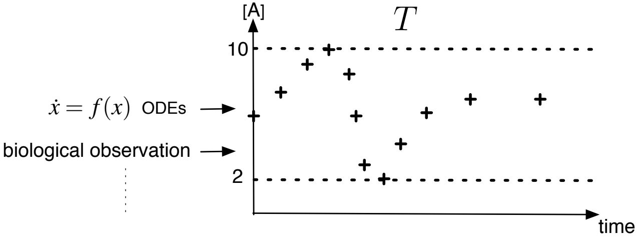
A Logical Paradigm for Systems and Synthetic Biology

Biological Model = (Quantitative) State Transition System K
Biological Properties = Temporal Logic Formulae ϕ
Automatic Verification = Model-checking $K \models \phi$
Model Inference = Constraint Solving $K' \models \phi'$

- Expression of high-level specifications
- Model-checking can be efficient on large complex systems
- Temporal logic with numerical constraints can deal with continuous time models (ODE or CTMC, hybrid systems)

Query language for large reaction networks [Eker et al. PSB 02, Chabrier Fages CMSB 03, Batt et al. Bi 05]
Analysis of experimental data time series [Fages Rizk CMSB 07]
Kinetic parameter search [Bernot et al. JTB 04] [Calzone et al. TCSB 06] [Rizk et al. 08 CMSB]
Robustness analysis [Batt et al. 07] [Rizk et al. 09 ISMB]

Temporal Logic with constraints over $\mathbb{R}$



- **F**([A]>10) : the concentration of A eventually gets above 10.
- **FG**([A]>10) : the concentration of A eventually reaches and remains above value 10.
- **F**(Time=t1∧[A]=v1 ∧ **F**(.... ∧ **F**(Time=tN∧[A]=vN)...))
  Numerical data time series (e.g. experimental curves)
- Local maxima, oscillations, period constraints, etc.

True/False valuation of temporal logic formulae

The **True/False** valuation of temporal logic formulae is **not well suited** to several problems :

- parameter search, optimization and control of continuous models
- quantitative estimation of robustness
- sensitivity analyses

True/False valuation of temporal logic formulae

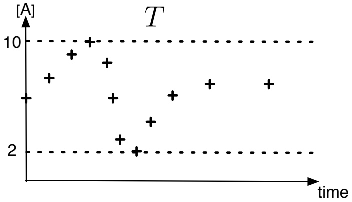
The **True/False** valuation of temporal logic formulae is **not well suited** to several problems :

- parameter search, optimization and control of continuous models
- quantitative estimation of robustness
- sensitivity analyses

$\rightarrow$ need for a continuous degree of satisfaction of temporal logic formulae

How far is the system from verifying the specification ?

From Model-Checking to TL Constraint Solving



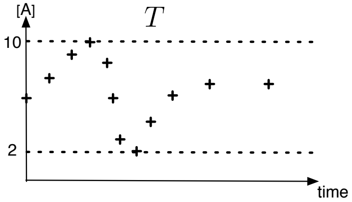
$LTL(\mathbb{R})$

$\Phi = F([A] \geq 7 \wedge F([A] \leq 0))$

Model-checking

the formula is false

From Model-Checking to TL Constraint Solving



$LTL(\mathbb{R})$

Φ=F([A]≥7
∧F([A]≤0))

Model-checking

the formula is false

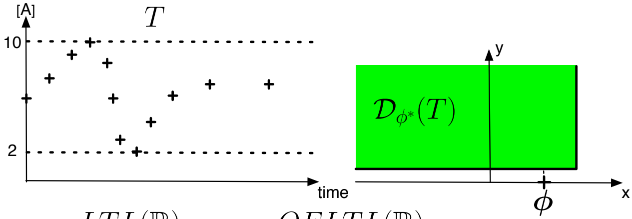
$QFLTL(\mathbb{R})$

Φ*=F([A]≥x
∧F([A]≤y))

Constraint solving

the formula is true for any
x≤10 ∧ y≥2

From Model-Checking to TL Constraint Solving



$LTL(\mathbb{R})$

Φ=F([A]≥7
∧F([A]≤0))

Model-checking

the formula is false

$QFLTL(\mathbb{R})$

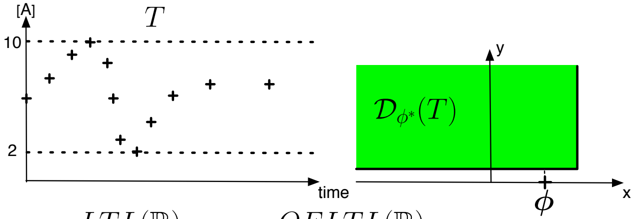
Φ*=F([A]≥x
∧F([A]≤y))

Constraint solving

the formula is true for any
x≤10 ∧ y≥2

Validity domain for the free variables [Fages Rizk CMSB'07, TCS 11]

From Model-Checking to TL Constraint Solving



$LTL(\mathbb{R})$	$QFLTL(\mathbb{R})$
Φ=F([A]≥7 ∧F([A]≤0))	Φ*=F([A]≥x ∧F([A]≤y))
Model-checking	**Constraint solving**
the formula is false vd=2 **sd=1/3**	the formula is true for any x≤10 ∧ y≥2

Validity domain for the free variables [Fages Rizk CMSB'07, TCS 11]

Violation degree $vd(T,\phi) = \text{distance}(val(\phi), D_{\phi^*}(T))$

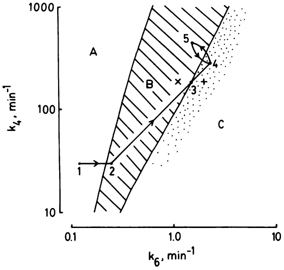
Satisfaction degree $sd(T,\phi) = \frac{1}{1+vd(T,\phi)} \in [0,1]$

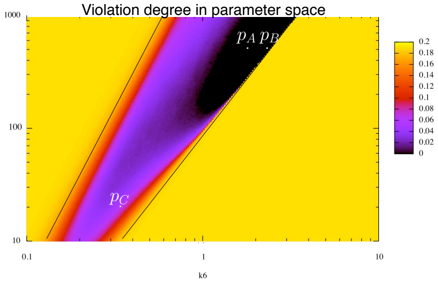
Bifurcation Diagrams and LTL($\mathbb{R}$) Satisfaction Diagrams

Example with :

- yeast cell cycle model [Tyson PNAS 91]
- oscillation of at least 0.3

$$\phi^*: \mathbf{F}(\ [A]>x \wedge \mathbf{F}([A]<y)\ );\ \text{amplitude } x\text{-}y \geq 0.3$$

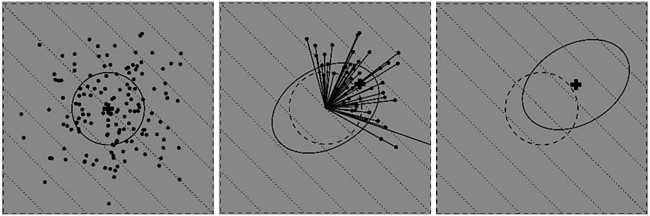




Bifurcation diagram LTL($\mathbb{R}$) satisfaction diagram

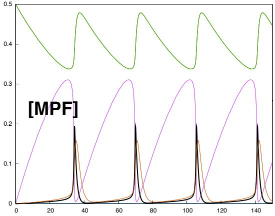
Parameter Inference by Local Search

- LTL($\mathbb{R}$) satisfaction degree as black-box fitness function (computed by TL constraint solving)
- Other constraints on parameter range, inequalities,... added to the fitness function
- Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [Hansen Osermeier 01, Hansen 08]: probabilistic neighborhood updated in covariance matrix at each move



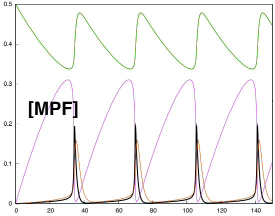
Kinetic Parameter Inference from LTL($\mathbb{R}$) Spec.

- yeast cell cycle control model [Tyson PNAS 91]
- 6 variables, 8 kinetic parameters



Kinetic Parameter Inference from LTL($\mathbb{R}$) Spec.

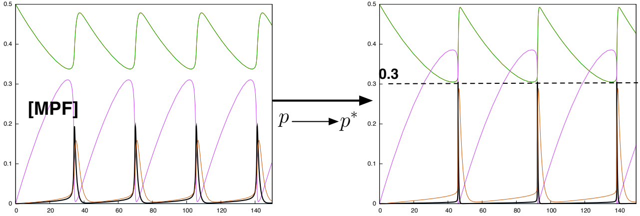
- yeast cell cycle control model [Tyson PNAS 91]
- 6 variables, 8 kinetic parameters



- Pb : find values of 8 parameters such that amplitude is ≥ 0.3
  ϕ^*: **F**([A]>x $\wedge$ **F**([A]<y))
  amplitude z=x-y
  goal : z = 0.3

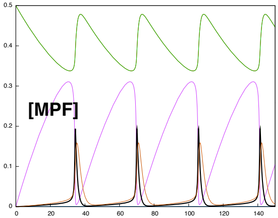
Kinetic Parameter Inference from LTL($\mathbb{R}$) Spec.

- yeast cell cycle control model [Tyson PNAS 91]
- 6 variables, 8 kinetic parameters



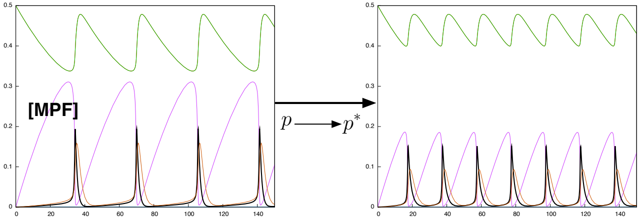
- Pb : find values of 8 parameters such that amplitude is ≥ 0.3
  ϕ^*: **F**([A]>x $\wedge$ **F**([A]<y))
  amplitude z=x-y
  goal : z $=$ 0.3
- $\rightarrow$ solution found after 30s (100 calls to the fitness function)

Kinetic Parameter Search from Period Constraints



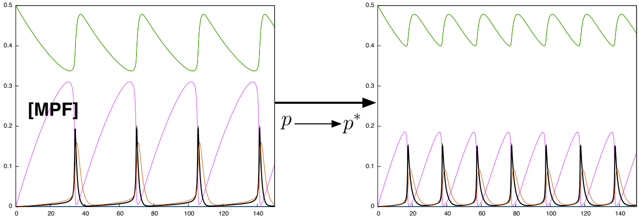
- Pb : find values of 8 parameters such that period is 20

Kinetic Parameter Search from Period Constraints



- Pb : find values of 8 parameters such that period is 20
- $\rightarrow$ Solution found after 60s (200 calls to the fitness function)

Kinetic Parameter Search from Period Constraints



- Pb : find values of 8 parameters such that period is 20
- $\rightarrow$ Solution found after 60s (200 calls to the fitness function)
- Scales up to 50-100 parameters.
- Linear speed-up on a cluster of 10000 cores. Parallelization of parameter sets and multiconditions for mutants.

Robustness Measure

Robustness defined as mean functionality with respect to :

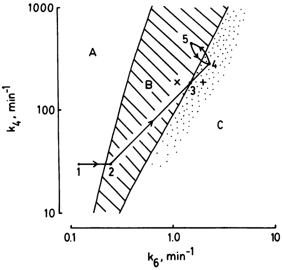
- a biological system
- a functionality property ϕ
- a set P of perturbations

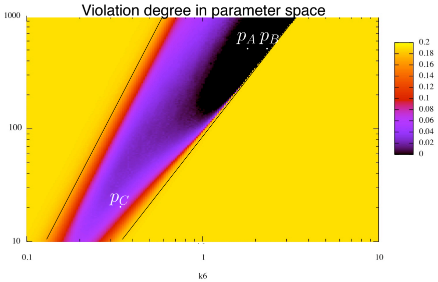
Measure of robustness w.r.t. LTL($\mathbb{R}$) spec:

$$\mathcal{R}_{\phi,P} = \int_{p \in P} sd(\phi, T(p)) \; prob(p) \; dp$$

where $T(p)$ is the trace obtained by numerical integration of the ODE for perturbation p of the parameters
$\longrightarrow$ estimated by sampling

Example on Cell Cycle Control

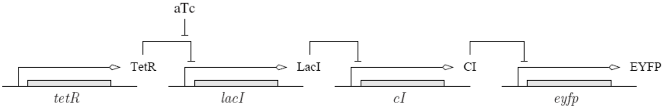


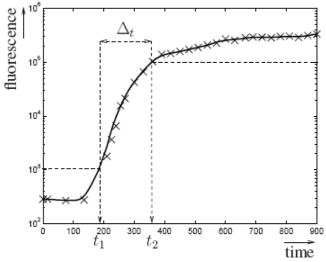


$$\mathcal{R}_{\phi,p_A} = 0.83,\ \mathcal{R}_{\phi,p_B} = 0.43,\ \mathcal{R}_{\phi,p_C} = 0.49$$

Example on a Synthetic Toggle Switch for *E. Coli*

Cascade of transcriptional inhibitions added to *E.coli* [Weiss et al PNAS 05] **input** small molecule aTc **output** protein EYFP





Specification: EYFP has to remain below 10^3 for at least 150 min., then exceeds 10^5 after at most 450 min., and switches from low to high levels in less than 150 min.

Specification in FOLTL($\mathbb{R}$)

The timing specifications can be formalized in temporal logic as follows:

$$\begin{aligned}\phi(t_1, t_2) = & \quad \mathbf{G}(\mathit{time} < t_1 \rightarrow [\mathrm{EYFP}] < 10^3) \\ \wedge & \quad \mathbf{G}(\mathit{time} > t_2 \rightarrow [\mathrm{EYFP}] > 10^5) \\ \wedge & \quad t_1 > 150 \wedge t_2 < 450 \wedge t_2 - t_1 < 150\end{aligned}$$

which is abstracted into

$$\begin{aligned}\phi(t_1, t_2, b_1, b_2, b_3) = & \quad \mathbf{G}(\mathit{time} < t_1 \rightarrow [\mathrm{EYFP}] < 10^3) \\ \wedge & \quad \mathbf{G}(\mathit{time} > t_2 \rightarrow [\mathrm{EYFP}] > 10^5) \\ \wedge & \quad t_1 > b1 \wedge t_2 < b_2 \wedge t_2 - t_1 < b_3\end{aligned}$$

for computing validity domains for b_1, b_2, b_3

with the objective $b_1 = 150, b_2 = 450, b_3 = 150$
for computing the satisfaction degree in a given trace.

Improving robustness

Variance-based global sensitivity indices $S_i = \frac{Var(E(R|P_i))}{Var(R)} \in [0,1]$

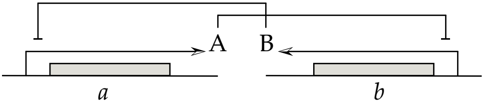
S_γ	20.2 %	$S_{\kappa_{eyfp},\gamma}$	8.7 %
$S_{\kappa_{eyfp}}$	7.4 %	$S_{\kappa_{cl},\gamma}$	6.2 %
$S_{\kappa_{cl}}$	6.1 %	$S_{\kappa^0_{cl},\gamma}$	5.0 %
$S_{\kappa^0_{lacl}}$	3.3 %	$S_{\kappa^0_{cl},\kappa_{eyfp}}$	2.8 %
$S_{\kappa^0_{cl}}$	2.0 %	$S_{\kappa_{cl},\kappa_{eyfp}}$	1.8 %
$S_{\kappa_{lacl}}$	1.5 %	$S_{\kappa^0_{eyfp},\gamma}$	1.5 %
$S_{\kappa^0_{eyfp}}$	0.9 %	$S_{\kappa^0_{cl},\kappa_{cl}}$	1.1 %
$S_{u_{aTc}}$	0.4 %	$S_{\kappa^0_{cl},\kappa_{lacl}}$	0.5 %
total first order	40.7 %	total second order	31.2 %

degradation factor γ has the strongest impact on the cascade.

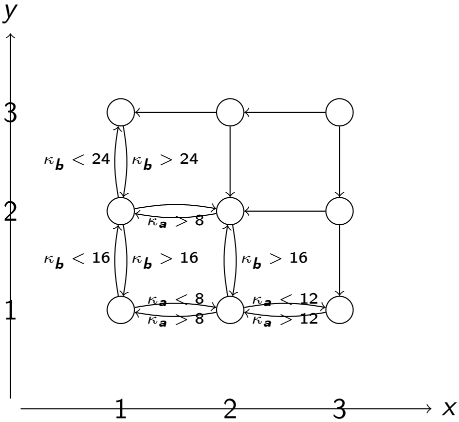
aTc variations have a very low impact

different importance of the basal κ^0_{eyfp} and regulated κ_{eyfp} EYFP production rates

Constraints on Transitions in Piecewise Multi-affine Models



$$\dot{x}_a = \kappa_a r_a(x_b, \theta_b, \theta'_b) - \gamma_a x_a$$
$$\dot{x}_b = \kappa_b r_b(x_a, \theta_a, \theta'_a) - \gamma_b x_b$$



FOCTL queries:

Reachability

? $EF(X = 3)$

Enumeration of stable states:

? $X = V \wedge AX(X = V)$

$X = V = 1 \wedge \kappa_a < 8 \wedge \kappa_b < 16$

$X = V = 2 \wedge 8 < \kappa_a < 12 \wedge \kappa_b < 16$

$X = V = 3 \wedge 12 < \kappa_a$

$X = V = 4 \wedge \kappa_a < 8 \wedge 16 < \kappa_b < 24$

$X = V = 7 \wedge 24 < \kappa_b$

Outline

States as Constraints

- Computation domain $\mathcal{D}$
- Constraint language closed by conjunction, negation, quantification (e.g. $\mathbb{R}_{\mathsf{lin}}$: finite unions of polyhedra)

States as Constraints

- Computation domain $\mathcal{D}$
- Constraint language closed by conjunction, negation, quantification (e.g. $\mathbb{R}_{\text{lin}}$: finite unions of polyhedra)

- Finite set X of state variables: $\vec{x}$
- State constraint: any constraint over $\vec{x}$, noted $s(\vec{x})$

States as Constraints

- Computation domain $\mathcal{D}$
- Constraint language closed by conjunction, negation, quantification (e.g. $\mathbb{R}_{\text{lin}}$: finite unions of polyhedra)

- Finite set X of state variables: $\vec{x}$
- State constraint: any constraint over $\vec{x}$, noted $s(\vec{x})$

- Set of ground states: $|s|_{\mathcal{D}} = \{\rho(\vec{x}) \mid \rho : X \longrightarrow \mathcal{D},\ \mathcal{D} \models \rho(s)\}$

Transitions as Constraints

- Finite set of primed state variables: $\vec{x}'$
- Transition constraint: any constraint over $\vec{x} \cup \vec{x}'$, noted $r(\vec{x}, \vec{x}')$

- Set of ground transitions: from state $\rho(\vec{x})$ to state $\rho(\vec{x}')$ for all valuations ρ s.t. $\mathcal{D} \models \rho(r)$.

Transitions as Constraints

- Finite set of primed state variables: $\vec{x}'$
- Transition constraint: any constraint over $\vec{x} \cup \vec{x}'$, noted $r(\vec{x}, \vec{x}')$

- Set of ground transitions: from state $\rho(\vec{x})$ to state $\rho(\vec{x}')$ for all valuations ρ s.t. $\mathcal{D} \models \rho(r)$.

- Predecessor state constraint: $\mathit{pred}(r) = \exists \vec{x}' \ r$

Transitions as Constraints

- Finite set of primed state variables: $\vec{x}'$
- Transition constraint: any constraint over $\vec{x} \cup \vec{x}'$, noted $r(\vec{x}, \vec{x}')$
- Set of ground transitions: from state $\rho(\vec{x})$ to state $\rho(\vec{x}')$ for all valuations ρ s.t. $\mathcal{D} \models \rho(r)$.
- Predecessor state constraint: $pred(r) = \exists \vec{x}' \ r$
- A Constrained Transition System (CTS) is a transition constraint r s.t. $|succ(r)|_{\mathcal{D}} \subseteq |pred(r)|_{\mathcal{D}}$, i.e. $\mathcal{D} \models succ(r) \Rightarrow pred(r)$.

Transitions as Constraints

- Finite set of primed state variables: $\vec{x}'$
- Transition constraint: any constraint over $\vec{x} \cup \vec{x}'$, noted $r(\vec{x}, \vec{x}')$
- Set of ground transitions: from state $\rho(\vec{x})$ to state $\rho(\vec{x}')$ for all valuations ρ s.t. $\mathcal{D} \models \rho(r)$.
- Predecessor state constraint: $pred(r) = \exists \vec{x}'\ r$
- A Constrained Transition System (CTS) is a transition constraint r s.t. $|succ(r)|_{\mathcal{D}} \subseteq |pred(r)|_{\mathcal{D}}$, i.e. $\mathcal{D} \models succ(r) \Rightarrow pred(r)$.
- A CTS r is finitary (resp. bounded) if for any s the set of predecessors $\{\exists \vec{x}'(r^i \wedge s[\vec{x}'/\vec{x}])\}_{i \geq 1}$ is finite (bounded card.).

$\mathbb{R}_{\mathsf{lin}}$ Linear Arithmetic over the Reals

- Atomic constraints in $\mathbb{R}_{\mathsf{lin}}$ are inequalities over linear combination of variables

$$2 * x + 4 * y \leq 5$$

$\mathbb{R}_{\text{lin}}$ Linear Arithmetic over the Reals

- Atomic constraints in $\mathbb{R}_{\text{lin}}$ are inequalities over linear combination of variables

$$2 * x + 4 * y \leq 5$$

- Such constraints admit for solutions sets of valuation which are (open) polyhedra in the n-dimensional space, where n is the number of variables

$\mathbb{R}_{\text{lin}}$ Linear Arithmetic over the Reals

- Atomic constraints in $\mathbb{R}_{\text{lin}}$ are inequalities over linear combination of variables

$$2 * x + 4 * y \leq 5$$

- Such constraints admit for solutions sets of valuation which are (open) polyhedra in the n-dimensional space, where n is the number of variables

- The conjunction of constraints $a \wedge b$ admits for solutions the intersection of the polyhedra solutions of a and b which is a polyhedron.

FO($\mathbb{R}_{\text{lin}}$) through Finite Unions of Polyhedra

Disjunction $\vee$ A disjunction of conjunctions of linear arithmetic constraints is a finite union of polyhedra and $(\bigcup_i A_i) \cap \left(\bigcup_j B_j\right) = \bigcup_{i,j}(A_i \cap B_j)$

Negation $\neg$ The complement of a polyhedron is a union of polyhedra
The complement of a union of polyhedra is the intersection of the complements of each polyhedron

Existential $\exists$ Projection to the subspace of the other dimensions

Universal $\forall$ $\neg\exists x(\neg c)$

Equality $=$ double inclusion $U \cap \overline{V} = \overline{U} \cap V = \emptyset$

Outline

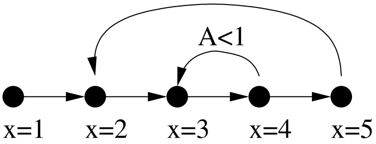
First-Order Computation Tree Logic FOCTL($\mathcal{D}$)

$$
\begin{aligned}
CTL ::= \;& \mid c \\
& \mid \phi \wedge \psi \mid \phi \vee \psi \mid \neg\phi \mid \exists x \phi \mid \forall x \phi \\
& \mid \mathsf{E}X(\phi) \mid \mathsf{E}F(\phi) \mid \mathsf{E}G(\phi) \\
& \mid \mathsf{A}X(\phi) \mid \mathsf{A}F(\phi) \mid \mathsf{A}G(\phi)
\end{aligned}
$$

First-Order Computation Tree Logic FOCTL($\mathcal{D}$)

$$
\begin{aligned}
CTL ::= \;& \mid c \\
& \mid \phi \wedge \psi \mid \phi \vee \psi \mid \neg\phi \mid \exists x \phi \mid \forall x \phi \\
& \mid \mathrm{E}X(\phi) \mid \mathrm{E}F(\phi) \mid \mathrm{E}G(\phi) \\
& \mid \mathrm{A}X(\phi) \mid \mathrm{A}F(\phi) \mid \mathrm{A}G(\phi)
\end{aligned}
$$

Example:

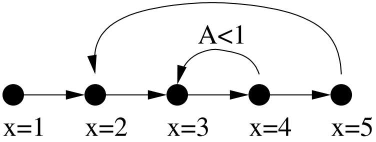


$\mathrm{E}G(x \leq V)$?

First-Order Computation Tree Logic FOCTL($\mathcal{D}$)

$$
\begin{aligned}
CTL ::= \; & \mid c \\
& \mid \phi \wedge \psi \mid \phi \vee \psi \mid \neg\phi \mid \exists x \phi \mid \forall x \phi \\
& \mid \mathsf{E}X(\phi) \mid \mathsf{E}F(\phi) \mid \mathsf{E}G(\phi) \\
& \mid \mathsf{A}X(\phi) \mid \mathsf{A}F(\phi) \mid \mathsf{A}G(\phi)
\end{aligned}
$$

Example:



$$\mathsf{E}G(x \leq V) \;?$$
$$(V \geq 5) \vee (1 \leq x \leq 4 \wedge V \geq 4 \wedge A < 1)$$

FOCTL Constraint Solving Fixpoint Algorithm

- $ex(s) = \exists \vec{x}'(r \wedge s[\vec{x}'/\vec{x}])$
- $ax(s) = \forall \vec{x}'(r \Rightarrow s[\vec{x}'/\vec{x}])$

FOCTL Constraint Solving Fixpoint Algorithm

- $ex(s) = \exists \vec{x}'(r \wedge s[\vec{x}'/\vec{x}])$
- $ax(s) = \forall \vec{x}'(r \Rightarrow s[\vec{x}'/\vec{x}])$

- $$\begin{aligned} \neg(ex(s)) &= \forall \vec{x}'(\neg r \vee \neg s[\vec{x}'/\vec{x}]) \\ &= ax(\neg s) \end{aligned}$$

FOCTL Constraint Solving Fixpoint Algorithm

- $ex(s) = \exists \vec{x}'(r \wedge s[\vec{x}'/\vec{x}])$
- $ax(s) = \forall \vec{x}'(r \Rightarrow s[\vec{x}'/\vec{x}])$

- $$\begin{aligned} \neg(ex(s)) &= \forall \vec{x}'(\neg r \vee \neg s[\vec{x}'/\vec{x}]) \\ &= ax(\neg s) \end{aligned}$$

- $ef(s) = \mu s'.s \vee ex(s')$, $af(s) = \mu s'.s \vee ax(s')$
  $\simeq (s \vee ex(s \vee ex(s \vee \ldots)))$

FOCTL Constraint Solving Fixpoint Algorithm

- $ex(s) = \exists \vec{x}'(r \wedge s[\vec{x}'/\vec{x}])$
- $ax(s) = \forall \vec{x}'(r \Rightarrow s[\vec{x}'/\vec{x}])$

- $$\begin{aligned} \neg(ex(s)) &= \forall \vec{x}'(\neg r \vee \neg s[\vec{x}'/\vec{x}]) \\ &= ax(\neg s) \end{aligned}$$

- $ef(s) = \mu s'.s \vee ex(s'),\ af(s) = \mu s'.s \vee ax(s')$
  $\simeq (s \vee ex(s \vee ex(s \vee \ldots)))$

- $eg(s) = \nu s'.s \wedge ex(s'),\ ag(s) = \nu s'.s \wedge ax(s')$
  $\simeq (s \wedge ex(s \wedge ex(s \wedge \ldots)))$

Complexity

For a bounded CTS r with maximum cardinality K for the set of predecessor state constraints for any state constraint

the time complexity for deciding the $\mathcal{D}$-satisfiability of ϕ in r with an oracle constraint solver for $\mathcal{D}$ is

$$O(|\phi| * K^2)$$

(what is implemented in FOCTL($\mathbb{R}_{\text{lin}}$))

$$O(|\phi| * K)$$

for traces of length K without branching
(what is implemented in Biocham)

Outline

Motivation: Analysis/Synthesis of Gene/Protein Networks

States and Transitions as Constraints over $\mathbb{R}_{lin}$

Temporal Logic Constraints over $\mathbb{R}_{lin}$

Implementation of FOCTL($\mathbb{R}_{lin}$)

Conclusion

Implementation and Optimizations

- In SWI-Prolog and Parma Polyhedra Library PPL

Implementation and Optimizations

- In SWI-Prolog and Parma Polyhedra Library PPL
- Results are projected to $\mathrm{pred}(r) = \exists \vec{x}'(r)$: all intermediate results (in particular for AX) can be intersected with $\mathrm{pred}(r)$.
- $ax(s) = \forall \vec{x}'(r \Rightarrow s[\vec{x}'/\vec{x}])$

Implementation and Optimizations

- In SWI-Prolog and Parma Polyhedra Library PPL
- Results are projected to $\mathrm{pred}(r) = \exists \vec{x}'(r)$: all intermediate results (in particular for AX) can be intersected with $\mathrm{pred}(r)$.
- $ax(s) = \forall \vec{x}'(r \Rightarrow s[\vec{x}'/\vec{x}])$
  $= \neg \exists \vec{x}' \neg(\neg r \vee s[\vec{x}'/\vec{x}])$

Implementation and Optimizations

- In SWI-Prolog and Parma Polyhedra Library PPL
- Results are projected to $\mathrm{pred}(r) = \exists \vec{x}'(r)$: all intermediate results (in particular for AX) can be intersected with $\mathrm{pred}(r)$.
- $$\begin{aligned} ax(s) &= \forall \vec{x}'(r \Rightarrow s[\vec{x}'/\vec{x}]) \\ &= \neg\exists \vec{x}' \neg(\neg r \vee s[\vec{x}'/\vec{x}]) \\ &= \neg\exists \vec{x}'(r \wedge \neg s[\vec{x}'/\vec{x}]) \end{aligned}$$

Implementation and Optimizations

- In SWI-Prolog and Parma Polyhedra Library PPL
- Results are projected to $\mathrm{pred}(r) = \exists \vec{x}'(r)$: all intermediate results (in particular for AX) can be intersected with $\mathrm{pred}(r)$.
- $$\begin{aligned} ax(s) &= \forall \vec{x}'(r \Rightarrow s[\vec{x}'/\vec{x}]) \\ &= \neg\exists\vec{x}'\neg(\neg r \vee s[\vec{x}'/\vec{x}]) \\ &= \neg\exists\vec{x}'(r \wedge \neg s[\vec{x}'/\vec{x}]) \\ &= \neg\exists\vec{x}'(r \wedge ex(s) \wedge \neg s[\vec{x}'/\vec{x}]) \end{aligned}$$

Implementation and Optimizations

- In SWI-Prolog and Parma Polyhedra Library PPL
- Results are projected to $\mathrm{pred}(r) = \exists \vec{x}'(r)$: all intermediate results (in particular for AX) can be intersected with $\mathrm{pred}(r)$.
- $$\begin{aligned} ax(s) &= \forall \vec{x}'(r \Rightarrow s[\vec{x}'/\vec{x}]) \\ &= \neg\exists \vec{x}' \neg(\neg r \vee s[\vec{x}'/\vec{x}]) \\ &= \neg\exists \vec{x}'(r \wedge \neg s[\vec{x}'/\vec{x}]) \\ &= \neg\exists \vec{x}'(r \wedge ex(s) \wedge \neg s[\vec{x}'/\vec{x}]) \end{aligned}$$
- $eg(s) = \nu s'.s \wedge ex(s')$

Implementation and Optimizations

- In SWI-Prolog and Parma Polyhedra Library PPL
- Results are projected to $\mathrm{pred}(r) = \exists\vec{x}'(r)$: all intermediate results (in particular for AX) can be intersected with $\mathrm{pred}(r)$.
- $$\begin{aligned} ax(s) &= \forall\vec{x}'(r \Rightarrow s[\vec{x}'/\vec{x}]) \\ &= \neg\exists\vec{x}'\neg(\neg r \vee s[\vec{x}'/\vec{x}]) \\ &= \neg\exists\vec{x}'(r \wedge \neg s[\vec{x}'/\vec{x}]) \\ &= \neg\exists\vec{x}'(r \wedge ex(s) \wedge \neg s[\vec{x}'/\vec{x}]) \end{aligned}$$
- $$\begin{aligned} eg(s) &= \nu s'.s \wedge ex(s') \\ &= \nu s'.s \wedge \exists\vec{x}'(r \wedge s'[\vec{x}'/\vec{x}]) \end{aligned}$$

Implementation and Optimizations

- In SWI-Prolog and Parma Polyhedra Library PPL
- Results are projected to $\mathrm{pred}(r) = \exists \vec{x}'(r)$: all intermediate results (in particular for AX) can be intersected with $\mathrm{pred}(r)$.
- $$\begin{aligned} ax(s) &= \forall \vec{x}'(r \Rightarrow s[\vec{x}'/\vec{x}]) \\ &= \neg\exists \vec{x}'\neg(\neg r \vee s[\vec{x}'/\vec{x}]) \\ &= \neg\exists \vec{x}'(r \wedge \neg s[\vec{x}'/\vec{x}]) \\ &= \neg\exists \vec{x}'(r \wedge ex(s) \wedge \neg s[\vec{x}'/\vec{x}]) \end{aligned}$$
- $$\begin{aligned} eg(s) &= \nu s'.s \wedge ex(s') \\ &= \nu s'.s \wedge \exists \vec{x}'(r \wedge s'[\vec{x}'/\vec{x}]) \\ &= \nu s'.\exists \vec{x}'(r \wedge s \wedge s'[\vec{x}'/\vec{x}]) \end{aligned}$$

Implementation and Optimizations

- In SWI-Prolog and Parma Polyhedra Library PPL
- Results are projected to $\mathrm{pred}(r) = \exists \vec{x}'(r)$: all intermediate results (in particular for AX) can be intersected with $\mathrm{pred}(r)$.
- $$\begin{aligned} ax(s) &= \forall \vec{x}'(r \Rightarrow s[\vec{x}'/\vec{x}]) \\ &= \neg \exists \vec{x}' \neg(\neg r \vee s[\vec{x}'/\vec{x}]) \\ &= \neg \exists \vec{x}'(r \wedge \neg s[\vec{x}'/\vec{x}]) \\ &= \neg \exists \vec{x}'(r \wedge ex(s) \wedge \neg s[\vec{x}'/\vec{x}]) \end{aligned}$$
- $$\begin{aligned} eg(s) &= \nu s'.s \wedge ex(s') \\ &= \nu s'.s \wedge \exists \vec{x}'(r \wedge s'[\vec{x}'/\vec{x}]) \\ &= \nu s'.\exists \vec{x}'(r \wedge s \wedge s'[\vec{x}'/\vec{x}]) \end{aligned}$$
- idem for $ag(s)$

Optimizing representation

- Remove any subsumed P_i from a union of polyhedra $\bigcup_i P_i$,
  But subsumption test quadratic in the size of the union...

Optimizing representation

- Remove any subsumed P_i from a union of polyhedra $\bigcup_i P_i$,
  But subsumption test quadratic in the size of the union...
- Convex hull computation and maintenance for subsumption test between unions and for intersections.

Optimizing representation

- Remove any subsumed P_i from a union of polyhedra $\bigcup_i P_i$,
  But subsumption test quadratic in the size of the union...

- Convex hull computation and maintenance for subsumption test between unions and for intersections.

- partitionning of discrete dimensions where the variable x always appears in the form $x = \mathit{constant}$.

Computation Results

Comparison to Delzanno and Podelski's CLP($\mathbb{R}$) implementation on an Intel(R) Core(TM)2 CPU at 1.86GHz with 2GB of RAM.

	CLP($\mathbb{R}$)	FOCTL($\mathbb{R}_{\text{lin}}$)
bakery	0.69	2.29
bakery3	12.47	3.15
bakery4	47.73	4.21
ticket	0.64	2.76 (widening)
bbuffer1	4.09	3.70
bbuffer2	0.67	6.80
ubuffer	4.49	2.93 (widening)
insertion	0.02	4.43 (widening)
selection	0.02	10.21
mesi	1.03	5.56
matrix-mul 0	.02	16.07
csm	3.81	7.88

Outline

Motivation: Analysis/Synthesis of Gene/Protein Networks

States and Transitions as Constraints over $\mathbb{R}_{\text{lin}}$

Temporal Logic Constraints over $\mathbb{R}_{\text{lin}}$

Implementation of FOCTL($\mathbb{R}_{\text{lin}}$)

Conclusion

Conclusion

- FOCTL($\mathbb{R}_{\text{lin}}$) provides an expressive modeling and querying language for infinite-state transition systems:
  - computes validity domains for free variables in the formula
  - computes continuous satisfaction degree for the formula
  - computes validity domains for parameters in transition system

Conclusion

- FOCTL($\mathbb{R}_{\text{lin}}$) provides an expressive modeling and querying language for infinite-state transition systems:
  - computes validity domains for free variables in the formula
  - computes continuous satisfaction degree for the formula
  - computes validity domains for parameters in transition system
- Potential blow-up with large unions of polyhedra
  - from $\vee$, $\neg$, $\Rightarrow$
  - from fixpoint computation

  Implementated in Prolog + PPL

Conclusion

- FOCTL($\mathbb{R}_{\text{lin}}$) provides an expressive modeling and querying language for infinite-state transition systems:
  - computes validity domains for free variables in the formula
  - computes continuous satisfaction degree for the formula
  - computes validity domains for parameters in transition system
- Potential blow-up with large unions of polyhedra
  - from $\vee$, $\neg$, $\Rightarrow$
  - from fixpoint computation

  Implementated in Prolog + PPL
- More efficient implementation for boxes and for non-branching traces in Biocham: successful for parameter inference problems in high dimension

Conclusion

- FOCTL($\mathbb{R}_{lin}$) provides an expressive modeling and querying language for infinite-state transition systems:
  - computes validity domains for free variables in the formula
  - computes continuous satisfaction degree for the formula
  - computes validity domains for parameters in transition system
- Potential blow-up with large unions of polyhedra
  - from $\vee$, $\neg$, $\Rightarrow$
  - from fixpoint computation

  Implementated in Prolog + PPL
- More efficient implementation for boxes and for non-branching traces in Biocham: successful for parameter inference problems in high dimension
- Hybrid computation domains ? (FO)CTL($\mathbb{R}_{lin}$,f,Graph,FD) ?