Model Checking, Performance Evaluation, Synthesis and Optimization of Cyber–Physical Systems

Kim G. Larsen
Aalborg University, DENMARK
Model Checking, Performance Evaluation, Synthesis and Optimization of Cyber–Physical Systems

Kim G. Larsen
Aalborg University, DENMARK
Overview

- **Timed Automata / UPPAAL**
  - Verification

- **Stochastic Priced Timed Automata / UPPAAL SMC**
  - Performance Evaluation
  - SMC in a Nutshell
  - Stochastic Hybrid Automata

- **Timed Games / UPPAAL TIGA**
  - Controller Syntesis

- **Stochastic Priced Timed Games / UPPAAL STRATEGI**
  - Optimal & Safe Syntheses

- **Conclusion**
Timed Games
Timed Automata & Model Checking

State (L1, x=0.81)
Transitions
(L1, x=0.81)  
- 2.1 →  
(L1, x=2.91)  
- →  
(goal, x=2.91)

E⟨⟩ goal ?  
A⟨⟩ goal ?  
A[ ] ¬ L4 ?
Timed Games & Synthesis

Uncontrollable

Controllable

Question
Does there exist a strategy that guarantees $A \leftrightarrow \text{Goal}$?

Strategy:
$\sigma : (l, u) \mapsto \{\lambda, \text{c}_{\text{act}}\}$

EMSIG Autumn School 2015
Kim Larsen [6]
Decidability of Timed Games

Theorem [AMPS98,HK99]
Reachability and safety timed games are decidable and EXPTIME-complete. Furthermore memoryless and “region-based” strategies are sufficient.

\[ \sim \text{classical regions are sufficient for solving such problems} \]

Theorem [AM99,BHPR07,JT07]
Optimal-time reachability timed games are decidable and EXPTIME-complete.

Computing Winning States
Reachability Games

Definitions

- $cPred(X) = \{ q \in Q \mid \exists q' \in X. q \xrightarrow{c} q' \}$
- $uPred(X) = \{ q \in Q \mid \exists q' \in X. q \xrightarrow{u} q' \}$
- $Pred_t(X,Y) = \{ q \in Q \mid \exists t. q^t \in X \text{ and } \forall s \leq t. q^s \in Y^C \}$

- $\pi(X) = Pred_t[ X \cup cPred(X) , uPred(X^C) ]$

Theorem:
The set of winning states is obtained as the least fixpoint of the function: $X \mapsto \pi(X) \cup \text{Goal}$
Symbolic On-the-fly Algorithms for Timed Games

- $S, S'$ are symbolic states, i.e. sets of concrete states;
- $G$ is the set of (concrete) goal states;
- $E = \{S \xrightarrow{c} S', S \xrightarrow{u} S'\}$ is the (finite) set of symbolic transitions (controller moves);
- $\text{Waiting} \subseteq E$ is the list of symbolic transitions waiting to be processed;
- $\text{Passed}$ is the list of the passed symbolic states;
- $\text{Win}(S) \subseteq S$ is the subset of $S$ currently known to be winning;
- $\text{Depend}(S) \subseteq E$ indicates the edges (predecessors) of $S$ which must be updated.

Initialization:

$\text{Passed} \leftarrow \{S_0\}$ where $S_0 = \{(l_0, \bar{0})\}$;
$\text{Waiting} \leftarrow \{(S_0, \alpha, S') : S' = \text{Post}_\alpha(S_0)\}$;
\[ \text{Win}(S_0) \leftarrow S_0 \cap \{\text{Goal}\} \times \mathbb{R}_{\geq 0} \];
\[ \text{Depend}(S_0) \leftarrow \emptyset ; \]

Main:

while $((\text{Waiting} \neq \emptyset) \land (s_0 \notin \text{Win}(S_0)))$ do
$c = (S, \alpha, S') \leftarrow \text{pop}(\text{Waiting})$;

if $S' \notin \text{Passed}$ then
$\text{Passed} \leftarrow \text{Passed} \cup \{S'\}$;
$\text{Depend}(S') \leftarrow \{(S, \alpha, S')\}$;
$\text{Win}(S') \leftarrow S' \cap \{\text{Goal}\} \times \mathbb{R}_{\geq 0}$;
$\text{Waiting} \leftarrow \text{Waiting} \cup \{(S', \alpha, S'') : S'' = \text{Post}_\alpha(S')\}$;
if $\text{Win}(S') \neq \emptyset$ then $\text{Waiting} \leftarrow \text{Waiting} \cup \{c\}$;

else (* reevaluate *)\(^a\)
$\text{Win}^* \leftarrow \text{Pred}^t(Win[S] \cup \bigcup_{S \xrightarrow{c} T} \text{Pred}^c(Win[T]),$
\[ \bigcup_{S \xrightarrow{u} T} \text{Pred}^u(T \setminus \text{Win}[T]) \right) \cap S ; \]
if $\text{Win}(S) \subseteq \text{Win}^*$ then
$\text{Waiting} \leftarrow \text{Waiting} \cup \text{Depend}(S)$; $\text{Win}(S) \leftarrow \text{Win}^*$;
$\text{Depend}(S') \leftarrow \text{Depend}(S') \cup \{c\}$;
endif
endwhile

\(^a\) Cassez, David, Fleury, Larsen, Lime. Efficient on-the-fly algorithms for the analysis of timed games (CONCUR'05).

![symbolic version of on-the-fly MC algorithm for modal mu-calculus Liu & Smolka 98](image-url)
Model Checking (ex Train Gate)

\[ \phi: \text{Never two trains at the crossing at the same time} \]
Synthesis (ex Train Gate)

$$\phi$$: Never two trains at the crossing at the same time
Timed Games

Find strategy for controllable actions st behaviour satisfies $\phi$

$\phi$: Never two trains at the crossing at the same time

EMSIG Autumn School 2015

Kim Larsen [14]
A Buggy Brick Sorting Program

Exercise: Design Controller so that only yellow boxes are being pushed out.

Boxes
Piston
Black
Yellow
9 18 81 90
99
Blck
Yel
remove
eject
Controller
Ken Tindell
MAIN PUSH
Brick Sorting

start

on1
pos>=8

pos<=9

ok?

turn==1D
pos=0, turn++

pos<=18

sensor

pos>=17

x>=1

x=0

ok!

x=1

blck?

eject!

red?

pos>=79

on2

pos<=81

piston

pos<=90

remove?

y=0

y>=1

remove!

s2

s1

end

Generic Plate

Controller

Piston

EMSIG Autumn School 2015
Brick Sorting

Controller

Generic Plate

EMSIG Autumn School 2015
Production Cell Overview

- Realistic case-study described in several formalisms (1994 and later).
- **Objective:** stamp metal plates in press.
- feed belt, two-armed robot, press, and deposit belt.
Production Cell in UPPAAL Tiga
## Experimental Results

<table>
<thead>
<tr>
<th>Plates</th>
<th>Basic</th>
<th>Basic + inc</th>
<th>Basic + inc + pruning</th>
<th>Basic + inc + inc + pruning</th>
<th>Basic + inc + inc + topt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>mem</td>
<td>time</td>
<td>mem</td>
<td>time</td>
</tr>
<tr>
<td>2</td>
<td>win</td>
<td>0.0s</td>
<td>0.0s</td>
<td>0.0s</td>
<td>0.04s</td>
</tr>
<tr>
<td></td>
<td>lose</td>
<td>0.0s</td>
<td>0.0s</td>
<td>0.0s</td>
<td>n/a</td>
</tr>
<tr>
<td>3</td>
<td>win</td>
<td>0.5s</td>
<td>0.0s</td>
<td>0.0s</td>
<td>0.27s</td>
</tr>
<tr>
<td></td>
<td>lose</td>
<td>1.1s</td>
<td>0.1s</td>
<td>0.0s</td>
<td>4M</td>
</tr>
<tr>
<td>4</td>
<td>win</td>
<td>33.9s</td>
<td>0.2s</td>
<td>0.1s</td>
<td>1.88s</td>
</tr>
<tr>
<td></td>
<td>lose</td>
<td>n/a</td>
<td>0.5s</td>
<td>0.4s</td>
<td>n/a</td>
</tr>
<tr>
<td>5</td>
<td>win</td>
<td>n/a</td>
<td>3.0s</td>
<td>1.5s</td>
<td>13.35s</td>
</tr>
<tr>
<td></td>
<td>lose</td>
<td>n/a</td>
<td>11.1s</td>
<td>5.9s</td>
<td>59M</td>
</tr>
<tr>
<td>6</td>
<td>win</td>
<td>n/a</td>
<td>89.1s</td>
<td>38.9s</td>
<td>220.3s</td>
</tr>
<tr>
<td></td>
<td>lose</td>
<td>n/a</td>
<td>699s</td>
<td>317s</td>
<td>369M</td>
</tr>
<tr>
<td>7</td>
<td>win</td>
<td>n/a</td>
<td>3256s</td>
<td>1181s</td>
<td>6188s</td>
</tr>
<tr>
<td></td>
<td>lose</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>c3</th>
<th>c6</th>
<th>c12</th>
<th>u3</th>
<th>u6</th>
<th>u12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old</td>
<td>0.1s</td>
<td>1M</td>
<td>12s</td>
<td>63M</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>New</td>
<td>0.05s</td>
<td>3.5M</td>
<td>0.05s</td>
<td>3.5M</td>
<td>0.14s</td>
<td>55M</td>
</tr>
</tbody>
</table>

Plastic Injection Molding Machine

- Robust and optimal control
- Tool Chain
  - Synthesis: UPPAAL, TIGA
  - Verification: PHAVer
  - Performance: SIMULINK
- 40% improvement of existing solutions.
Oil Pump Control Problem

- **R1**: stay within safe interval $[4.9, 25.1]$  

- **R2**: minimize average/overall oil volume

\[
\int_{t=0}^{t=T} v(t) \, dt / T
\]
The Machine (consumption)

- Infinite cyclic demand to be satisfied by our control strategy.
- **P**: latency 2 s between state change of pump
- **F**: noise 0.1 l/s
Hybrid Game Model

(a) The Machine

(b) The Accumulator

(c) The Pump

Undecidability

Discretization
Abstract Game Model

- UPPAAL Tiga offers games of perfect information

Abstract game model such that states only contain information about:
- Volume of oil at the beginning of cycle
- The ideal volume as predicted by the consumption cycle
- Current time within the cycle
- State of the Pump (on/off)
- Discrete model

Kim Larsen [27]
Machine (uncontrollable)

Checks whether \( V \) under noise gets outside \([V_{\text{min}} + 0.1, V_{\text{max}} - 0.1]\)

```c
bool Noise(int s){
    // s is the duration of consumption (in t.u.)
    return (V-s<(V_{\text{min}}+1)\times D \mid V+s>(V_{\text{max}}-1)\times D);}
```
Pump (controllable)

Every 1 (one) seconds

```c
void update_val()
{
    int V_pred = V;
    time++;
    V += V_rate;
    V_acc += V + V_pred;
}
```

```
z >= 2*D && i < N
V_rate += 22, z = 0,
start[i] = time
```

```
z >= 2*D
update_pump?
```

```
V_rate -= 22, z = 0,
stop[i] = time, i++
```

OFF

ON
Global Approach

- Find some interval $I_1 = [V_1, V_2]$ such that $I_1$ is $m$-stable, i.e., from any $V_0$ in $I_1$ there is a strategy $st$ that keeps the fluctuation volume always within $[5, 25]$ and at the end within $I_2 = [V_1 + m, V_1 - m]$.

- $I_1$ is optimal among all $m$-stable intervals.

Queries (min. $K$)

control: $A \Diamond (\text{time}=20 \land \neg \text{BAD} \land V \in I_2 \land V_{\text{acc}} \leq K)$
Tool Chain

Strategy Synthesis TIGA

Performance Evaluation SIMULINK

Guaranteed
Correctness
Robustness
with
40% Improvement

Verification PHAVER

EMSIG Autumn School 2015

Kim Larsen [31]
Stochastic Priced Timed Games
Can I get to Sidney? (1–player)

Will I always come to Sidney? (1–player)

What is the optimal WC strategy? (2–player)

Is there a strategy guaranteeing WC <= 60? (2–player)

What is the optimal strategy? (1½–player)

What is the optimal strategy Guaranteeing WC <= 60? (1½–player)
Strategy:

\[ \sigma : \text{Exec}_G \rightarrow \mathcal{P} (\Sigma_c \cup \{\lambda\}) \setminus \{\emptyset\} \]

- Memoryless, deterministic, most permissive.

Run

\[ \pi = (\text{INIT}, x = 0) \xrightarrow{50.1 \ r} (\text{CHOICE}, x = 0) \xrightarrow{2.4 \ a} (A, x = 0) \xrightarrow{20.3 \ d} (\text{END}, x = 20.3) \]

Total time = 50.1 + 2.4 + 20.3 = 72.8
Objective: $A() (\text{END} \land \text{time} \leq 210)$

Deterministic, memoryless strategy:

Most permissive, memoryless strategy
Priced Timed Games

- Cost optimal strategy
  - take b immediately
  - WC = 280

Priced Run

$$\pi = (\text{Init}, x = 0) \xrightarrow{0} (\text{CHOICE}, x = 0) \xrightarrow{9.6} (A, x = 0) \xrightarrow{20.3} (\text{END}, x = 20.3)$$

Total cost = 0 + 9.6 + 60.9 = 75.5
Priced Timed MDP

- Cost optimal strategy
  - take b immediately
    - $WC = 280$

- Priced Timed MDP
- Optimal expected cost str
  - take b immediately
    - expectation $= 160$
Priced Timed MDP

- Cost optimal strategy
  - take b immediately overall = 280

- Priced Timed MDP
- Optimal expected cost str
  - take b immediately expectation = 160

- Minimal Expected Cost while guaranteeing END is reached within time 210:

  Strat.:
  \[\begin{align*}
  t > 90 & \rightarrow (100, w) \\
  t > 70 & \rightarrow (0, a) \\
  t < 70 & \rightarrow (0, b) \\
  \end{align*}\]
  \[= 204\]
Stochastic Strategies for Learning!

Objective: \( A() (\text{END} \land \text{time} \leq 210) \)

Most permissive, memoryless strategy:

Stochastic Strategies

\[ \mu^c_{(\ell,v)} : (\Sigma_c \cup \{\lambda\}) \rightarrow [0, 1]. \]

Cost optimal deterministic sub-strategy!
**Reinforcement Learning**

\[ \mu^c = U(\sigma^p) \]

**Time Bounded Reachability**

\[(G, T)\]

- **Simulation**
  - maxRuns
  - maxGood
  - maxBest

- **Filtering**
  - \(\Pi\)

- **Learning**
  - \(\Pi'\)
  - \(\mu^{c'}\)
  - TIGA

- **Determinization**
  - evalRuns
  - maxNoBetter
  - maxIterations
  - maxResets

- **Evaluation**
  - det(\(\mu^{c'}\))
  - SMC

- **Zonification**
  - \(\sigma_d\)
  - \(\sigma_z\)

**ATVA 2014, November 4, 2014**

Kim Larsen [40]
Strategies – Representation

Nondeterministic Strategies

\[ \sigma^n_{(\ell,v)} \subseteq (\Sigma_c \cup \{\lambda\}) \]

\[ R_\ell = \{(Z_1, a_1), \ldots, (Z_k, a_k)\}, \text{ where } a_i \in \Sigma_c \cup \{\lambda\}. \]

Stochastic Strategies

\[ \mu^s_{(\ell,v)} : (\Sigma_c \cup \{\lambda\}) \rightarrow [0,1] \]

Covariance Matrices

\[ R_\ell \]

Splitting

\[ \{(\bullet, 3), (\triangle, 1), (\blacksquare, 3)\} \]

Logistic Regression

\[ \{(\bullet, 1), (\triangle, 6), (\blacksquare, 1)\} \]
Learned Strategies

Covariance Matrices

More plots of runs according to strategies learned.

\[ \Sigma_{ab} \]
Learned Strategies

Covariance Matrices

More plots of runs according to strategies learned.

\[ \begin{array}{cc}
\lambda \\
\alpha \\
b
\end{array} \]
Learned Strategies

More plots of runs according to strategies learned.

Covariance Matrices
Learned Strategies

Covariance Matrices

More plots of runs according to strategies learned.
Learned Strategies

Covariance Matrices

More plots of runs according to strategies learned.
Learned Strategies

Covariance Matrices

More plots of runs according to strategies learned.

\[ \begin{pmatrix} \lambda & \alpha \\ \alpha & \beta \end{pmatrix} \]
**Uppaal TIGA**
strategy NS = control: A<> goal
strategy NS = control: A[] safe

**G**
Timed Game

**Φ**
synthesis

**P**
Stochastic
Priced
Timed Game

**G | σ**
Timed Automata

**σ**
Strategy

**σ°**
optimized Strategy

**P | σ**
Stochastic Priced
Timed Game

**P | σ°**
Stochastic Priced
Timed Automata

**Statistical Learning**
strategy DS = minE (cost) [<=10]: <> done under NS
strategy DS = maxE (gain) [<=10]: <> done under NS

**Uppaal SMC**
simulate 5 [<=10]{e1, e2} under SS
Pr[<=10](<> error) under SS
E[<=10;100](max: cost) under SS

**Uppaal**
E<> error under NS
A[] safe under NS
Can I get to Sidney? (1-player)

Will I always come to Sidney? (1-player)

What is the optimal WC strategy? (2-player)

Is there a strategy guaranteeing WC <= 60? (2-player)

What is the optimal strategy? (1½-player)

What is the optimal strategy guaranteeing WC <= 60? (1½-player)
DEMO
Safe and Optimal Train Gate

Find strategy that minimize
Expected time from Appr to Cross while being Safe
Q1: Find a safety strategy for Ego such no crash will ever occur no matter what Front is doing.
Q2: Find the most permissive strategy ensuring safety
Q3: Find the optimal sub-strategy that will allow Ego to go as far as possible (without overtaking).
Discretization

```c
void updateDiscrete()
{
    int oldVel, newVel;
    oldVel = velocityFront - velocityEgo;
    velocityEgo = velocityEgo + accelerationEgo;
    velocityFront = velocityFront + accelerationFront;
    newVel = velocityFront - velocityEgo;
}
```

Discrete

Continuous
No Strategy

\[ \Pr[\leq 100] \ (\neq \text{distance} \leq 5) \]

\[ A[] \ \text{distance} > 5 \]
Safety Strategy

strategy safe = control: A[] distance > 5

A[] distance > 5 under safe
\[
\inf\{\text{velocityFront} - \text{velocityEgo} = v\}: \text{distance under safe}
\]
strategy safeFast = minE (D) [<=100]: <> time >= 100 under safe
Other Case Studies

- **FIREWIRE**
- **BLUETOOTH**
- **10 node LMAC**
- Schedulability Analysis for Mix Cr Sys

- **Smart Grid Demand / Response**
- **Energy Aware Buildings**
- **Genetic Oscilator (HBS)**
- **Passenger Seating in Aircraft**
- **Battery Scheduling**
Floorheating

Bang-Bang Controller

STRATEGO-ON-CL Controller

Thursday Afternoon
Daniel Lux, Seluxit
Marco Muniz, AAU
2015-2021, 70MMDKK Innovation Fund DK

Center for Data-Intensive Cyber-Physical Systems

TU Dresden  INRIA  U College London
Learning, Analysis, Synthesis and Optimization of Cyber-Physical Systems
Applications
Case Studies: Controllers

- Memory Arbiter Synthesis and Verification for a Radar Memory Interface Card, 2005
- Analyzing a $\chi$ model of a turntable system using Spin, CADP and Uppaal, 2006
- Designing, Modelling and Verifying a Container Terminal System Using UPPAAL, 2008
- Model–based system analysis using Chi and Uppaal: An industrial case study, 2008
- Climate Controller for Pig Stables, 2008 (synth)
- Optimal and Robust Controller for Hydraulic Pump, 2009 (synth)
References

- Frits Vaandrager: A first introduction to UPPAAL
- Alexandre David, Kim G. Larsen: More features in UPPAAL

- For more see http://people.cs.aau.dk/~kgl/SSFT2015/
Case Studies: Protocols

- Analysis of a protocol for dynamic configuration of IPv4 link local addresses using Uppaal, 2006
- Formalizing SHIM6, a Proposed Internet Standard in UPPAAL, 2007
- Verifying the distributed real-time network protocol RTnet using Uppaal, 2007
- Analysis of the Zeroconf protocol using UPPAAL, 2009
- Model Checking the FlexRay Physical Layer Protocol, 2010
Using UPPAAL as Back-end

- Timed automata translator from Uppaal to PVS
- Component-Based Design and Analysis of Embedded Systems with UPPAAL PORT, 2008
UPPAAL is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata, extended with data types (bounded integers, arrays, etc.).

The tool is developed in collaboration between the Department of Information Technology at Uppsala University, Sweden and the Department of Computer Science at Aalborg University in Denmark.

License

The UPPAAL tool is free for non-profit applications. For information about commercial licenses, please email sales(at)uppaal(dot)com.

To find out more about UPPAAL, read this short introduction. Further information may be found at this web site in the pages About, Documentation, Download, and Examples.

Mailing Lists

UPPAAL has an open discussion forum group at YahooGroups intended for users of the tool. To join or post to the forum, please refer to the information at the discussion forum page. Bugs should be reported using the bug tracking system. To email the development team directly, please use uppaal(at)list(dot)it(dot)uu(dot)se.