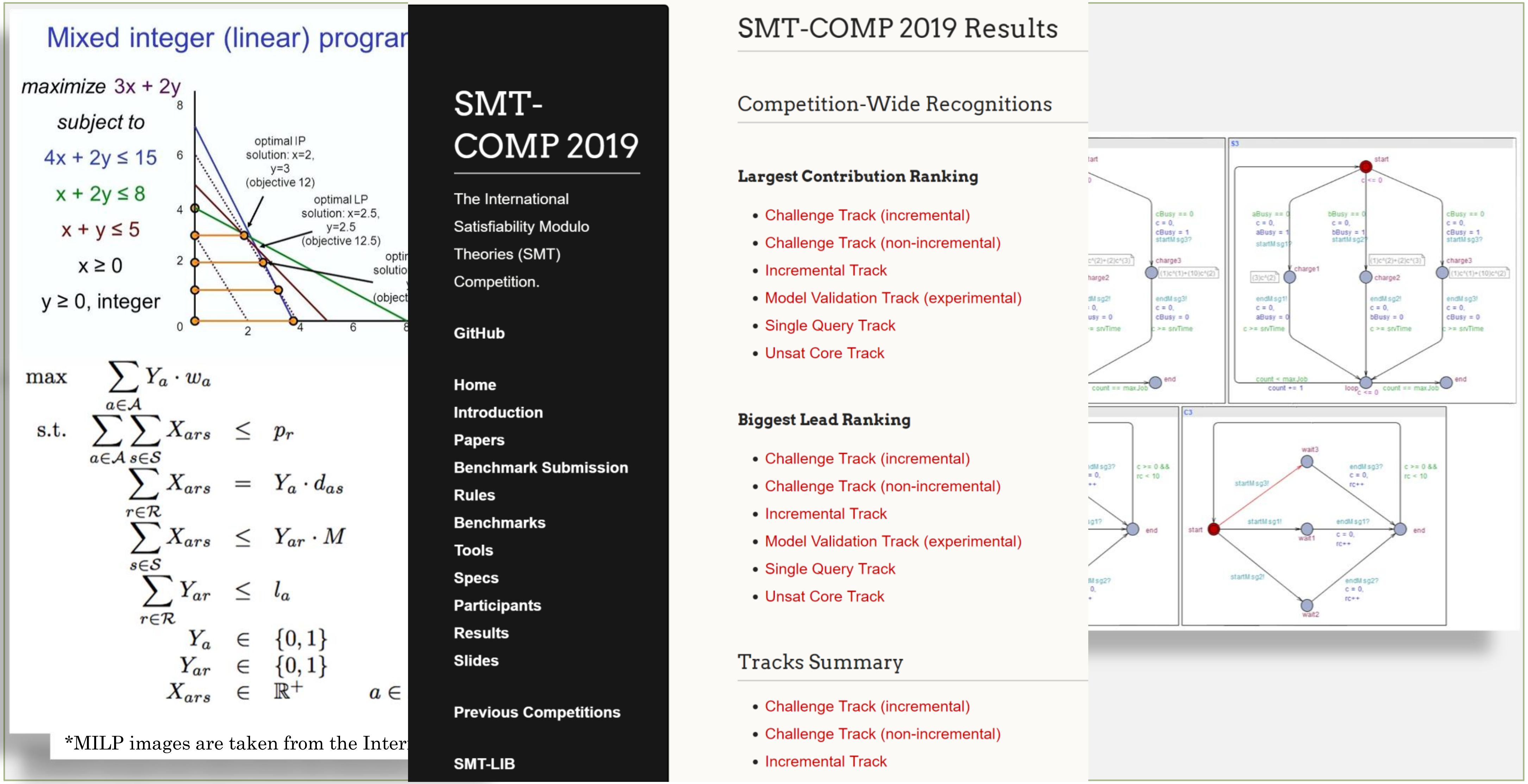
Task Scheduling with Nonlinear Costs using SMT Solvers

Mohammad Hekmatnejad, Giulia Pedrielli, and Georgios Fainekos

CPS Lab @ Arizona State University

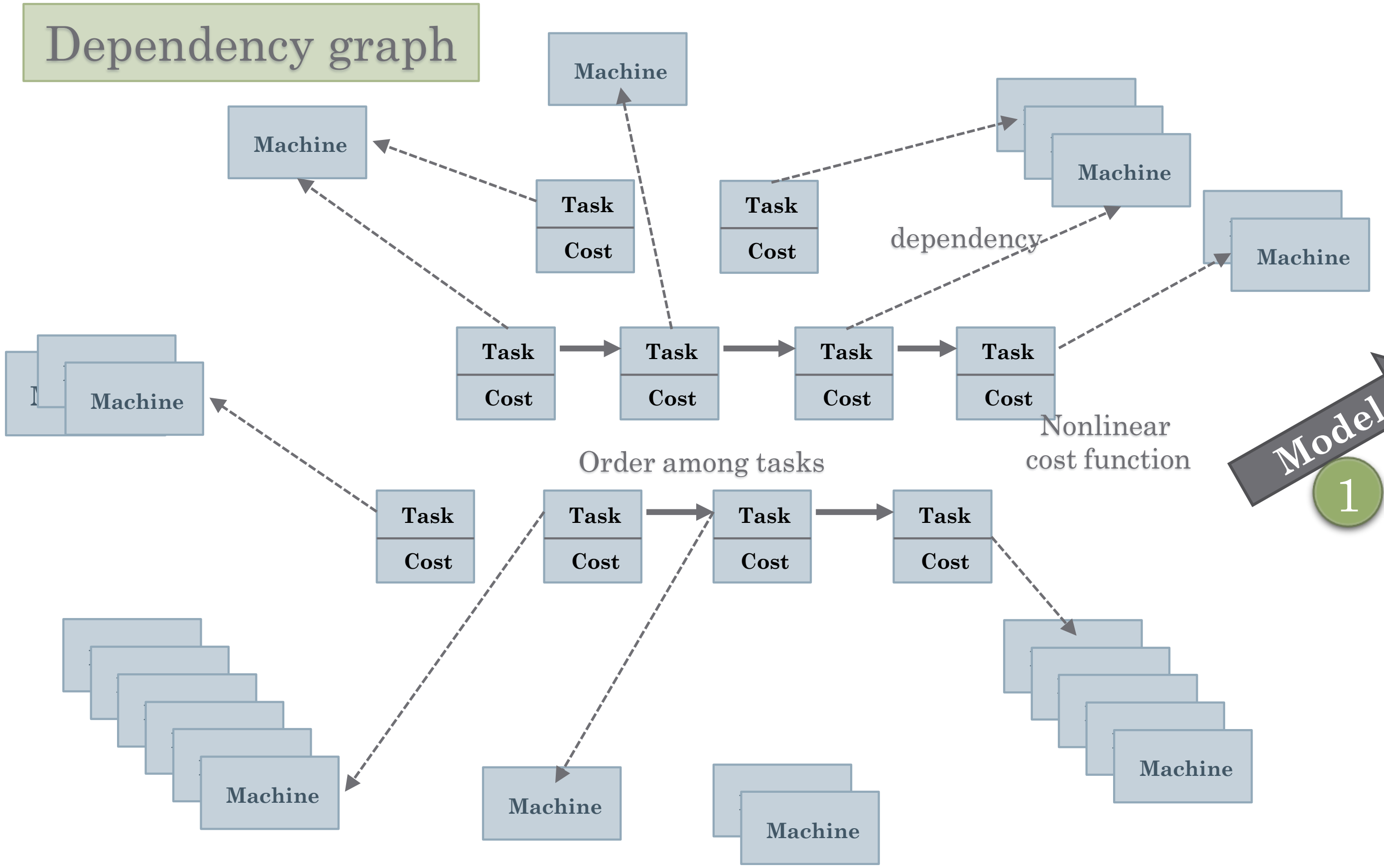
mhekmatn at asu edu
https://www.linkedin.com/in/mohammad-hekmatnejad-54535232

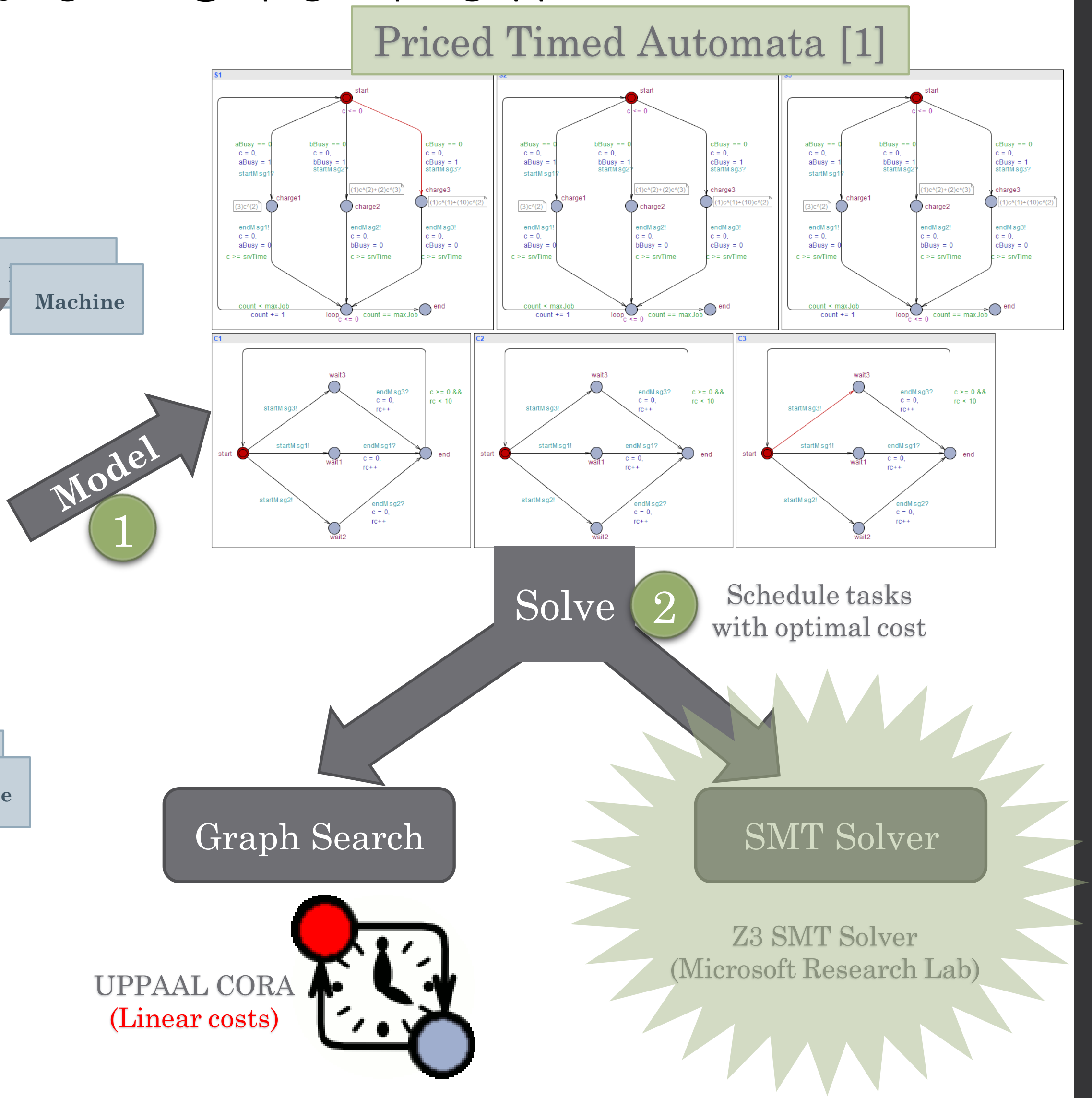
Motivation



*G. Audemard, A. Cimatti, A. Kornilowicz, and R. Sebastiani, “Bounded model checking for timed systems,” in FORTE. Springer, 2002
*K. G. Larsen, “Priced timed automata: Theory and tools,” in FSTTCS. Schloss Dagstuhl-Leibniz-Zentrum f¨ur Informatik, 2009.

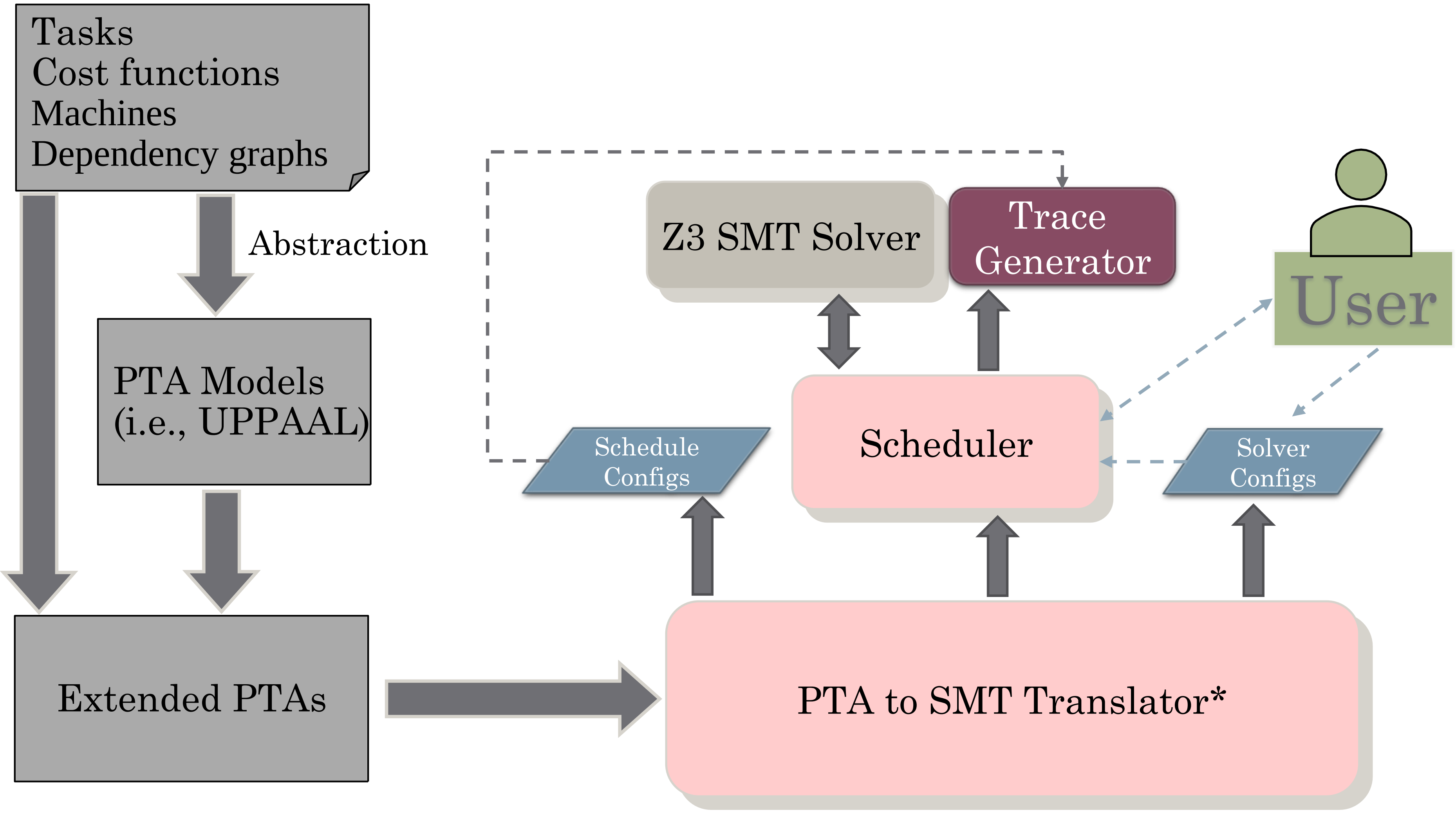
Problem Definition & Solution Overview





[1] G. Behrmann, K. G. Larsen, and J. I. Rasmussen, “**Priced timed automata: Algorithms and applications**,” in Formal Methods for Components and Objects (FMCO), ser. LNCS, 2005, pp. 162–182.

Solution Architecture



*Based on work by D. Bhave, S. N. Krishna, and A. Trivedi, "**On nonlinear prices in timed automata**," in V2CPS 2016

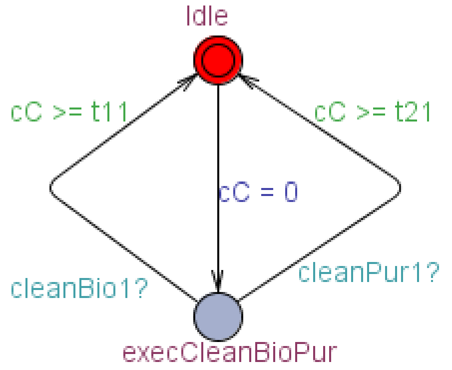
Summary of Our Contribution

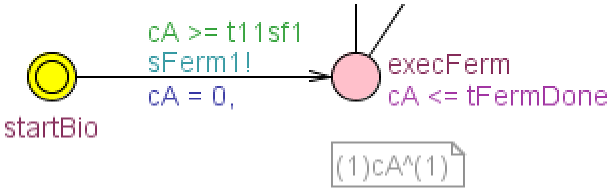
- We model tasks and machines as PTAs annotated with nonlinear cost functions on the clock variables.
- By modifying the framework proposed in [2], we translate the PTA reachability problem to an SMT formula whose models correspond to feasible schedules that satisfy a given cost constraint.
- For nonlinear cost functions, a bisection method can be used to compute optimal schedules.
- We demonstrate that the resulting framework based on SMT solvers can outperform UPPAAL CORA when the costs are linear functions of the clocks.
- Finally, we have released a publicly available tool called CEPTA2SMT available at: https://cpslab.assembla.com/spaces/bio-manufacturing/ .

[2] D. Bhave, S. N. Krishna, and A. Trivedi, "**On nonlinear prices in timed automata**," in V2CPS, ser. EPTCS, vol. 232, 2016, pp. 65–78.

Preliminaries

Sample PTA with 2 locations, 3 transitions, one local clock (two guards, one reset), and two actions.





- Priced Timed Automaton
  - $\mathcal{A} = \langle L, l_0, \mathcal{C}, A, E, I, P_L, P_E \rangle$
  - $E \subseteq L \times \mathcal{X}(\mathcal{C}) \times A \times 2^{\mathcal{C}} \times L$ i.e., $\langle l, \phi, a, \gamma, l' \rangle \in E$
    - $\mathcal{X}(\mathcal{C})$ is the set of conjunctive formulas with atomic clock constraints $c \bowtie r$ and $c_1 - c_2 \bowtie r$ for $c, c_1, c_2 \in \mathcal{C}$, $r \in \mathbb{R}^+$, and $\bowtie \in \{>, <, =, \geq, \leq\}$.
  - $I : L \rightarrow \mathcal{X}(\mathcal{C})$ assigns invariants to locations.
  - $P_E : E \rightarrow \mathbb{R}^+$ assigns constant prices to transitions
  - $P_L : L \rightarrow \Psi(\mathcal{C})$ assigns a nonlinear function of clocks to locations.
  - we consider actions as blocking send/receive signals over channels for inter-communication purposes.

*Preliminaries (cont')

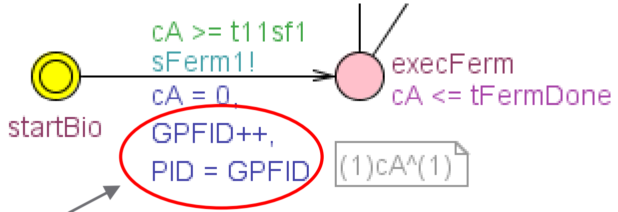
- A trace over PTA $\mathcal{A}$
  - is a sequence of locations and transitions i.e.,

$$p = l_0 \xrightarrow{a_0,\gamma_0,p_0,t_0} l_1 \xrightarrow{a_1,\gamma_1,p_1,t_1} l_2 \xrightarrow{a_2,\gamma_2,p_2,t_2} \dots l_n$$

  - $T_i = \langle l_i, \phi_i, a_i, \gamma_i, l_{i+1} \rangle \in E$
  - $\mathcal{C}_0 = 0$
  - $\mathcal{C}_i = (\mathcal{C}_{i-1}+t_{i-1})[\gamma_{i-1} = 0]$
  - $|p| = n$
  - At each location l_i, every clock valuation $(\mathcal{C}_i+t)$ satisfies $I(l_i)$ for $t < t_i$
  - The clock valuation $(\mathcal{C}_i+t_i)$ satisfies ϕ_i
  - $p_i = P_E(T_i) + P_L(l_i)$ is the price calculated by taking transition T_i from location l_i

Preliminaries (cont’)

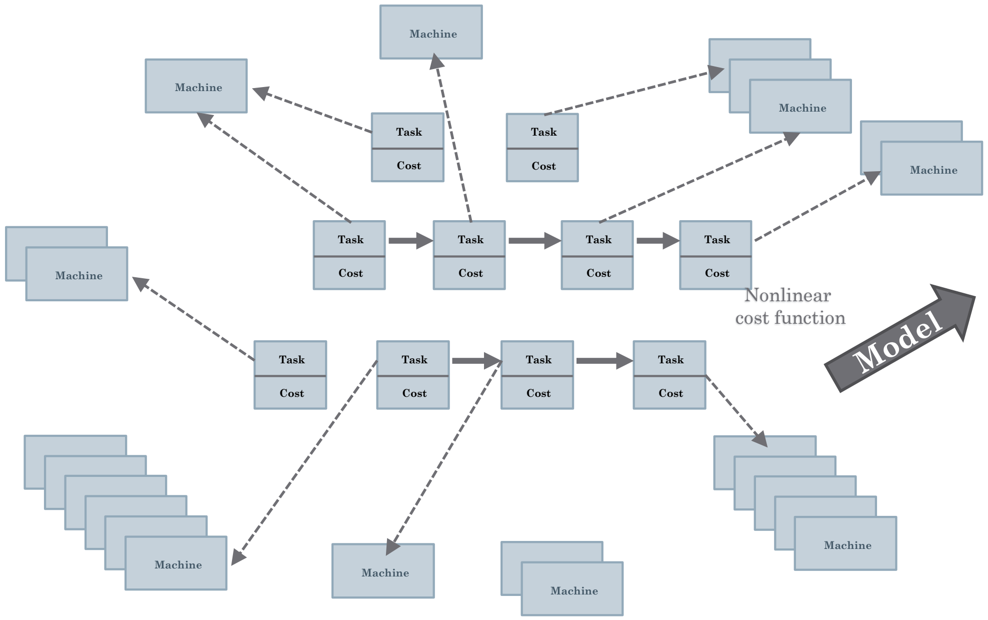
- PTA = better visualization, scalability, and more understandable, maintainable? (counters for loops, semaphores for mutual exclusion)
- **Extended PTA** is an extension of a standard PTA by adding a set of integer update variables (updates) U.
  - These variables are updated and evaluated in locations and transitions like clocks but more expressive.
  - $\mathcal{A} = \langle L, l_0, \mathcal{C}, A, E, I, P_L, P_E, U, U^0 \rangle$
  - $\boldsymbol{\mathcal{X}^u(U)}$ represents the same type of formulas as for $\mathcal{X}(\mathcal{C})$ using update variables rather than clocks.
  - $\boldsymbol{\mathcal{U}(U)}$ is the set of conjunctive formulas in which atomic update assignments are of the form $u \odot, u \bowtie n$ and $u_1 \bowtie u_2$ for $u, u_1, u_2 \in U, n \in \mathbb{Z}, \odot \in \{++, --\}, and \bowtie \in \{=, +=, -=\}$.
  - E is a set of transitions $s.t. E \subseteq L \times \mathcal{X}(\mathcal{C}) \cup \mathcal{X}^u(U) \cup \mathcal{U}(U) \times A \times 2^{\mathcal{C}} \times 2^U \times L$, and for $T = \langle l, \phi, a, \gamma, \lambda, l' \rangle$, λ is a set of update variables that needs to be updated after taking the transition.
  - $I : L \rightarrow \mathcal{X}(\mathcal{C}) \cup \mathcal{X}^u(U)$ assigns invariants to locations.

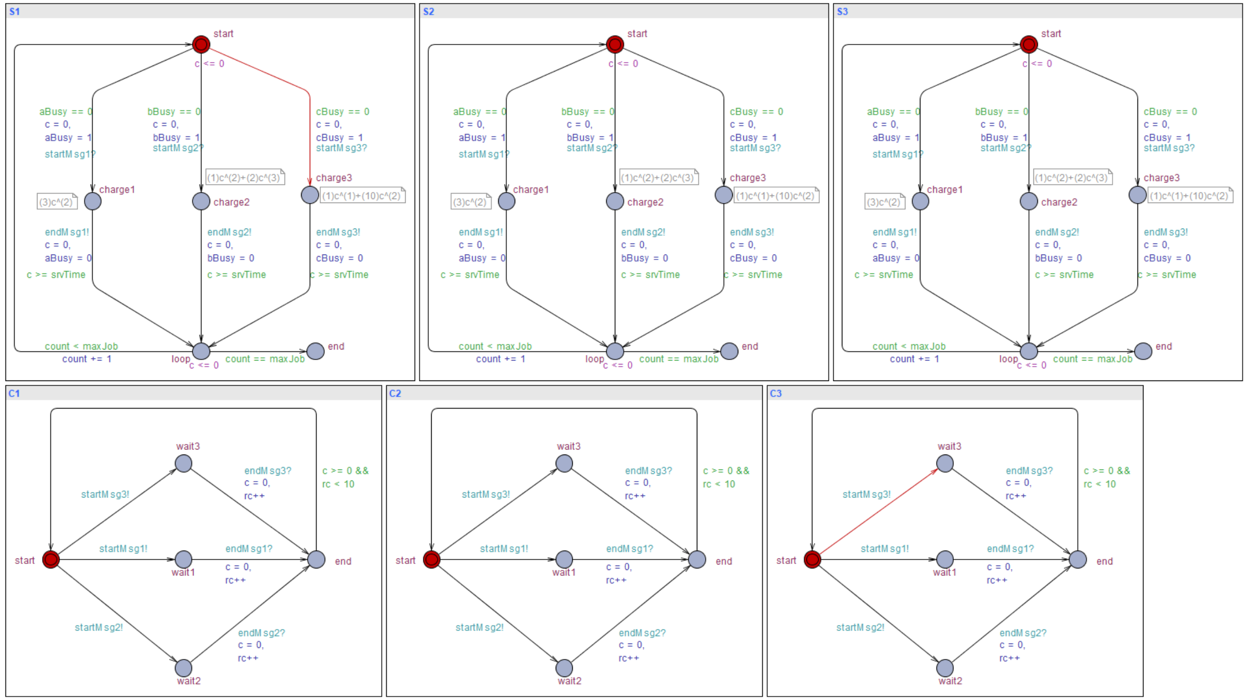


Preliminaries (cont')

- **Composite PTA:**
  - The parallel composition of m PTAs $\mathcal{A}_1, \ldots, \mathcal{A}_m$ denoted by
  - $\mathcal{A}_{||}^{m} = \mathcal{A}_1 || \mathcal{A}_2 \ldots || \mathcal{A}_m$ is the synchronized product of all the automata.
  - By $\mathcal{R} = s_0 \xrightarrow{p_0, t_0} s_1 \xrightarrow{p_1, t_1} \ldots s_{n-1} \xrightarrow{p_{n-1}, t_{n-1}} s_n$,

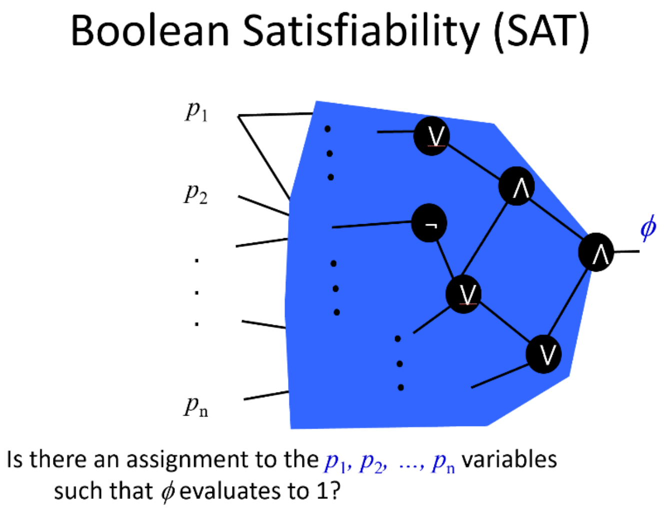
    we represent a trace over a composite PTA.
  - We let a set of global update variables $\mathcal{V}^g$ to be used among PTAs.

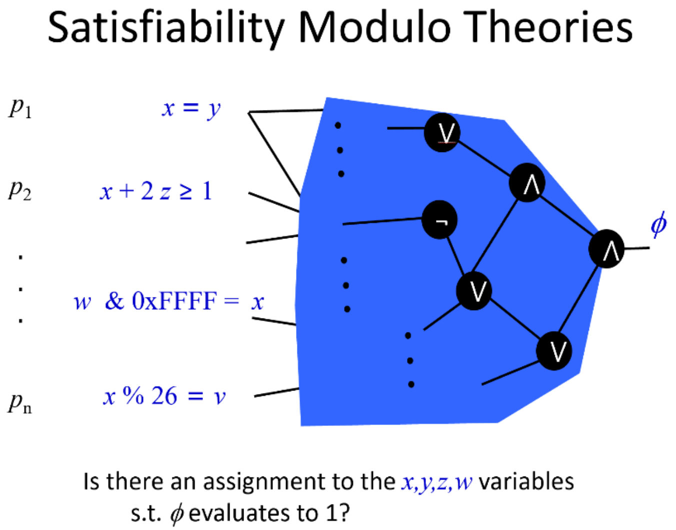




Preliminaries (cont')

- **Satisfiability Modulo Theory (SMT)**





- The SMT problem is checking if a given closed logical formula $\boldsymbol{\phi}$ is satisfiable with respect to some background theory $\boldsymbol{\mathcal{T}}$ which restricts the range of used symbols in $\boldsymbol{\phi}$.
- The SMT problem for ϕ and $\mathcal{T}$ is about the existence of a model of $\mathcal{T}$ that satisfies the formula ϕ.
- An *SMT solver* is a software that implements a procedure for satisfiability modulo for some given theory.

Pictures are taken from C. Barrett & S. A. Seshia, ICCAD 2009 Tutorial

Formal Problem Statement

- **Given** a set of jobs (tasks or operations) each with its nonlinear cost function, a set of machines, and a dependency graph among the jobs and machines, compute a schedule for the jobs which minimizes the total additive cost over all jobs.
- **Solution Overview:**

  - Formally, given a set of m concurrent PTAs $\mathcal{A}_i$, a finite horizon n, and a set of target locations $\mathcal{L} \subseteq \prod_{i=1}^{m} L_i$, we are interested in an execution path $\mathcal{R} = s_0 \xrightarrow{p_0, t_0} s_1 \xrightarrow{p_1, t_1} \dots s_{n-1} \xrightarrow{p_{n-1}, t_{n-1}} s_n$ for which $s_n \in \mathcal{L}$, and for any other execution path $\mathcal{R}' = s_0 \xrightarrow{p'_0, t'_0} s'_1 \xrightarrow{p'_1, t'_1} \dots s'_{l-1} \xrightarrow{p'_{l-1}, t'_{l-1}} s'_l$ of length $l \leq n$ and $s'_l \in \mathcal{L}$, we have $\sum_{i=0}^{n-1} p_i \leq \sum_{i=0}^{l-1} p'_i$.

Extended PTA to SMT Translation

$$s_{l_0} \Rightarrow U^0, \qquad \bigwedge_{l \in L} s_l \Rightarrow I(l)$$

At each time-step, for each PTA, for all $l \in L$ only-and-only one s_l is true.

$$\bigwedge_{T=\langle l,\phi,a,\gamma,\lambda,l'\rangle} T \Rightarrow (s_l \wedge \phi \wedge s'_{l'} \wedge (a \neq \epsilon \Rightarrow a) \wedge \bigwedge_{c \in \gamma} c' = z' \wedge \bigwedge_{c \notin \gamma} c' = c \wedge z' = z)$$

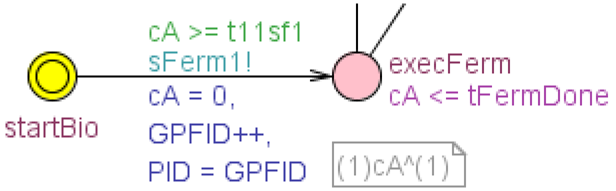
Regular transition

$T, T_{null}, T_\delta : E \rightarrow \{true, false\}$

z: $\mathbb{R}$ as a global clock

$$T_\delta \Rightarrow (s_l = s'_l \wedge (z' - z < 0) \wedge \bigwedge_{c \in \mathcal{C}} c' = c \wedge \bigwedge_{a \in A \setminus \{\epsilon\}} \neg a)$$

Delay transition



$$T_{null} \Rightarrow (s_l = s'_l \wedge z' = z \wedge \bigwedge_{c \in \mathcal{C}} c' = c \wedge \bigwedge_{a \in A \setminus \{\epsilon\}} \neg a)$$

Null transition

$$\bigwedge_{T=\langle l,\phi,a,\gamma,\lambda,l'\rangle} \neg T \Rightarrow \bigwedge_{u \in \lambda \setminus \mathcal{V}^g} u' = u \ , \quad \bigwedge_{a,b \in A \setminus \{\epsilon\}, a \neq b} (\neg a \vee \neg b)$$

At each time-step, for each PTA, at most one signal can be activated.

At each time-step, for each PTA, for all $T \in E \cup T_{null} \cup T_\delta$ only-and-only one T is true.

$$\left(\bigvee_{T=\langle l,\phi,\epsilon,\gamma,\lambda,l'\rangle} T \right) \Rightarrow \bigwedge_{a \in A \setminus \{\epsilon\}} \neg a \,, \qquad T_{null} \vee T_\delta \vee \bigvee_{T \in E} T$$

PTA to SMT Translation (cont')

$P \in \mathbb{R} \quad P_0 = 0, \quad \bigwedge_{T \in E} T \implies (P' = P + P_E(T))$ Cost rules

$$\bigwedge_{l \in L} T_\delta \wedge s_l \implies P' = P + P_L(l), \qquad T_{null} \implies P' = P$$

$(1 \leq i, j, k \leq m)$: Composite constraint rules

$$\mathcal{A}_i.A.a \implies \exists j \neq i \wedge \mathcal{A}_j.A.a \wedge \bigwedge_{k \neq i,j} \neg \mathcal{A}_k.A.a$$

At each time-step, for all PTAs, only one pair of the same signals can be activated.

Reachability rules

$$\forall u \in \mathcal{V}^g, \left(\bigwedge_{T_i = \langle l, \phi, a, \gamma, \lambda, l' \rangle, u \in \lambda} \neg T_i \right) \implies u' = u \qquad \bigwedge_{l \in \mathcal{L}} s_l$$

Cost optimization constraint rules

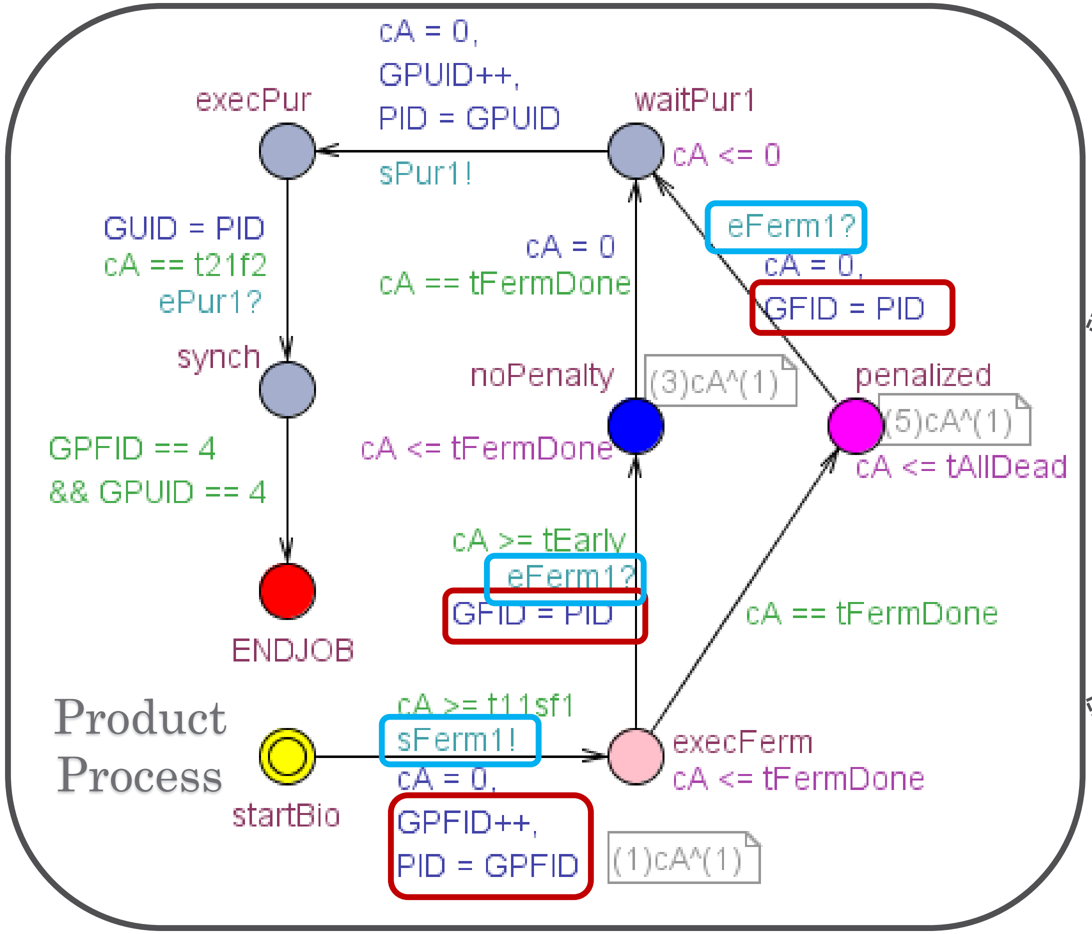
Z3 internal optimizer

`Optimal` $(\sum_{i=1}^{n} P_i)$ OR $Y_{min} \leq \sum_{i=1}^{n} P_i \leq Y_{max}$

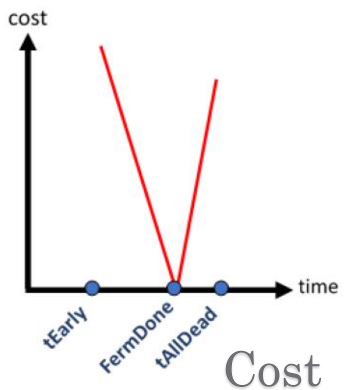
Bisection method

The cost of a satisfiable model can be used as a bound to start with as a strategy

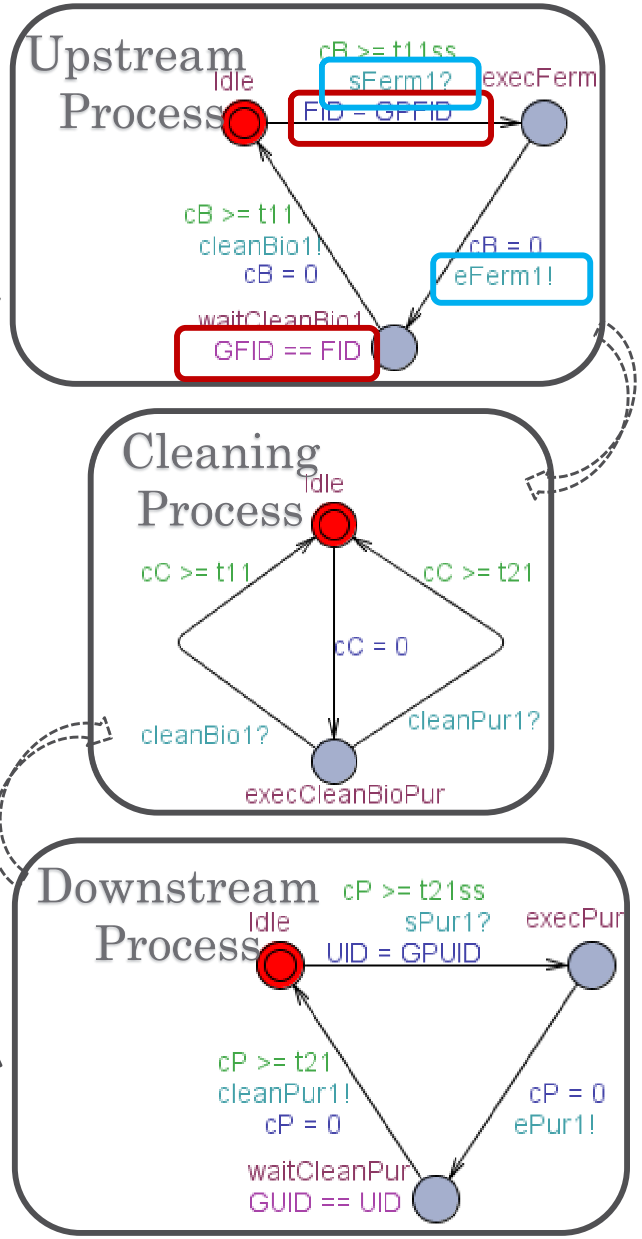
Bio-manufacturing example [3][4]



Dependency relation



Cost Function [5]



Eventually, all the products reach the ENDJOB location

[3] D. Petrides, D. Carmichael, C. Siletti, and A. Koulouris, "**Biopharmaceutical process optimization with simulation and scheduling tools**," Bioengineering, vol. 1, no. 4, pp. 154–187, 2014.
[4] M. Dorceus, S. S. Willard, A. Suttle, K. Han, P.-J. Chen, and M. Sha, "**Comparing culture methods in monoclonal antibody production: Batch, fed-batch, and perfusion**," BioProcess International, 3 2017.
[5] UPPAAL CORA http://people.cs.aau.dk/~adavid/cora/casestudies.html

Example (cont')

- System Configurations

Non-Deterministic

```
chan  sFerm1,  eFerm1;
chan  sPur1,  ePur1;
chan  cleanBio1,  cleanPur1;

ProdA1 = ProductA1(  sFerm1,  eFerm1, sPur1 ,  ePur1  );
ProdA2 = ProductA1(  sFerm1,  eFerm1, sPur1 ,  ePur1  );
ProdA3 = ProductA1(  sFerm1,  eFerm1, sPur1 ,  ePur1  );

Bio11 = Bioreactor1(  sFerm1,  eFerm1,  cleanBio1  );
Bio12 = Bioreactor1(  sFerm1,  eFerm1,  cleanBio1  );
Bio13 = Bioreactor1(  sFerm1,  eFerm1,  cleanBio1  );

Pur1 = Purifier1(  sPur1,  ePur1,  cleanPur1  );
Pur2 = Purifier1(  sPur1,  ePur1,  cleanPur1  );

Clean1 = Cleaner1(  cleanBio1, cleanPur1  );

system ProdA1, ProdA2, ProdA3, Bio11, Bio12, Bio13, Pur1, Pur2, Clean1 ;
```

Channels are signal actions in PTAs

Deterministic and Non-Deterministic

```
chan  sFerm1,  eFerm1;
chan  sFerm2,  eFerm2;
chan  sFerm3,  eFerm3;
chan  sPur1,  ePur1;
chan  sPur2,  ePur2;
chan  cleanBio1,  cleanPur1;

ProdA1 = ProductA1(  sFerm1,  eFerm1, sPur1 ,  ePur1  );
ProdA2 = ProductA1(  sFerm2,  eFerm2, sPur2 ,  ePur2  );
ProdA3 = ProductA1(  sFerm3,  eFerm3, sPur1 ,  ePur1  );

Bio11 = Bioreactor1(  sFerm1,  eFerm1,  cleanBio1  );
Bio12 = Bioreactor1(  sFerm2,  eFerm2,  cleanBio1  );
Bio13 = Bioreactor1(  sFerm3,  eFerm3,  cleanBio1  );

Pur1 = Purifier1(  sPur1,  ePur1,  cleanPur1  );
Pur2 = Purifier1(  sPur2,  ePur2,  cleanPur1  );

Clean1 = Cleaner1(  cleanBio1, cleanPur1  );

system ProdA1, ProdA2, ProdA3, Bio11, Bio12, Bio13, Pur1, Pur2, Clean1;
```

The results of solving bio-manufacturing task scheduling problems using SMT and CORA

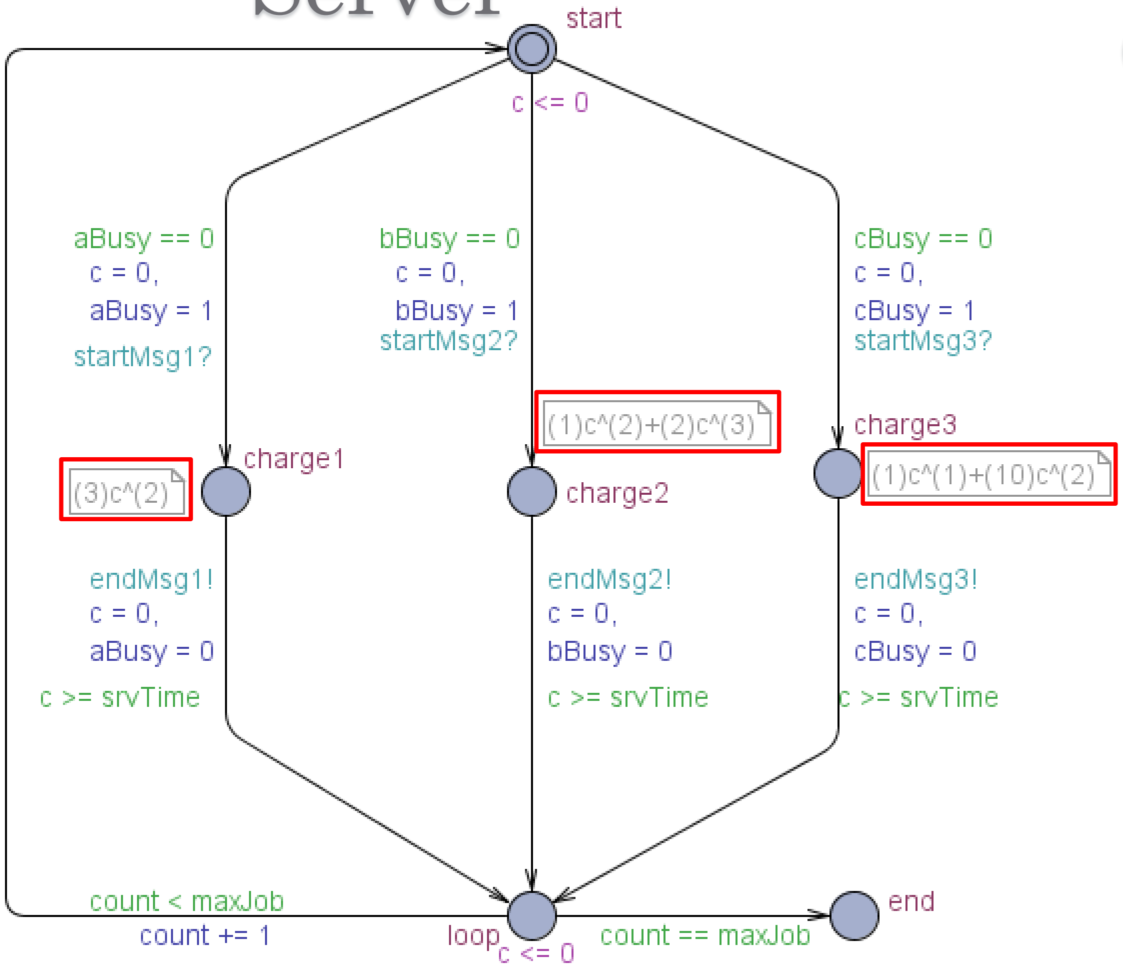
Test	1	2	3	4	5	6
# Prod	3	3	3	3	6	6
# Bio	3	3	3	3	3	6
# Pur	1	2	1	2	3	6
# Clean	1	1	2	2	3	6
EXCLUSIVELY NON-DETERMINISTIC OPTIONS						
SMT Max Step	40	40	40	40	50	50
Opt Cost	120	120	120	120	TO	TO
SMT Len	25	19	25	19	38	22
CORA Len	39	34	SO	SO	SO	SO
SMT Time	**56**	**33**	**73**	**30**	**363**	**685**
CORA Time	202	2074	SO	SO	SO	SO

The results of solving bio-manufacturing task scheduling problems using SMT and CORA

Test	1	2	3	4	5	6
# Prod	3	3	3	3	6	6
# Bio	3	3	3	3	3	6
# Pur	1	2	1	2	3	6
# Clean	1	1	2	2	3	6
DETERMINISTIC AND NON-DETERMINISTIC OPTIONS						
SMT Max Step	40	16	25	16	19	10
Opt Cost	220	120	220	120	240	240
SMT Len	25	16	25	16	19	10
CORA Len	34	28	35	29	SO	SO
SMT Time	**57**	**2**	**43**	**2**	**54**	**5**
CORA Time	43	41	8515	6860	SO	SO

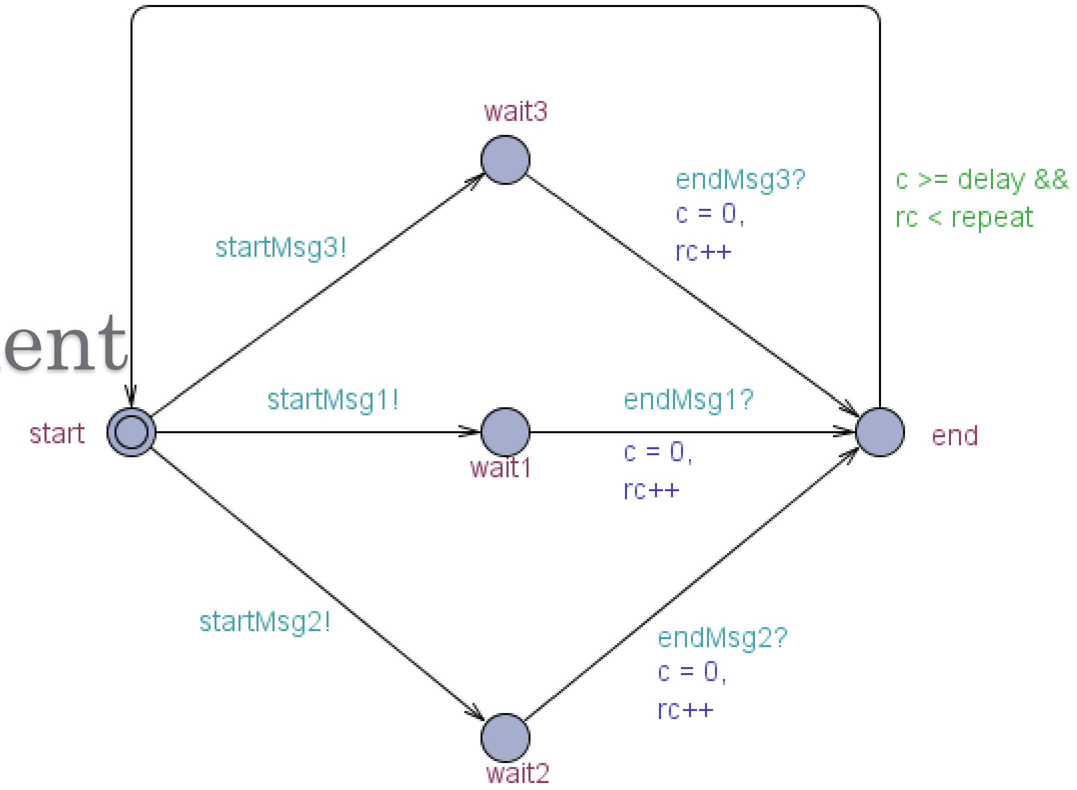
Nonlinear Cost Example

Server



Eventually, all the servers reach the end location

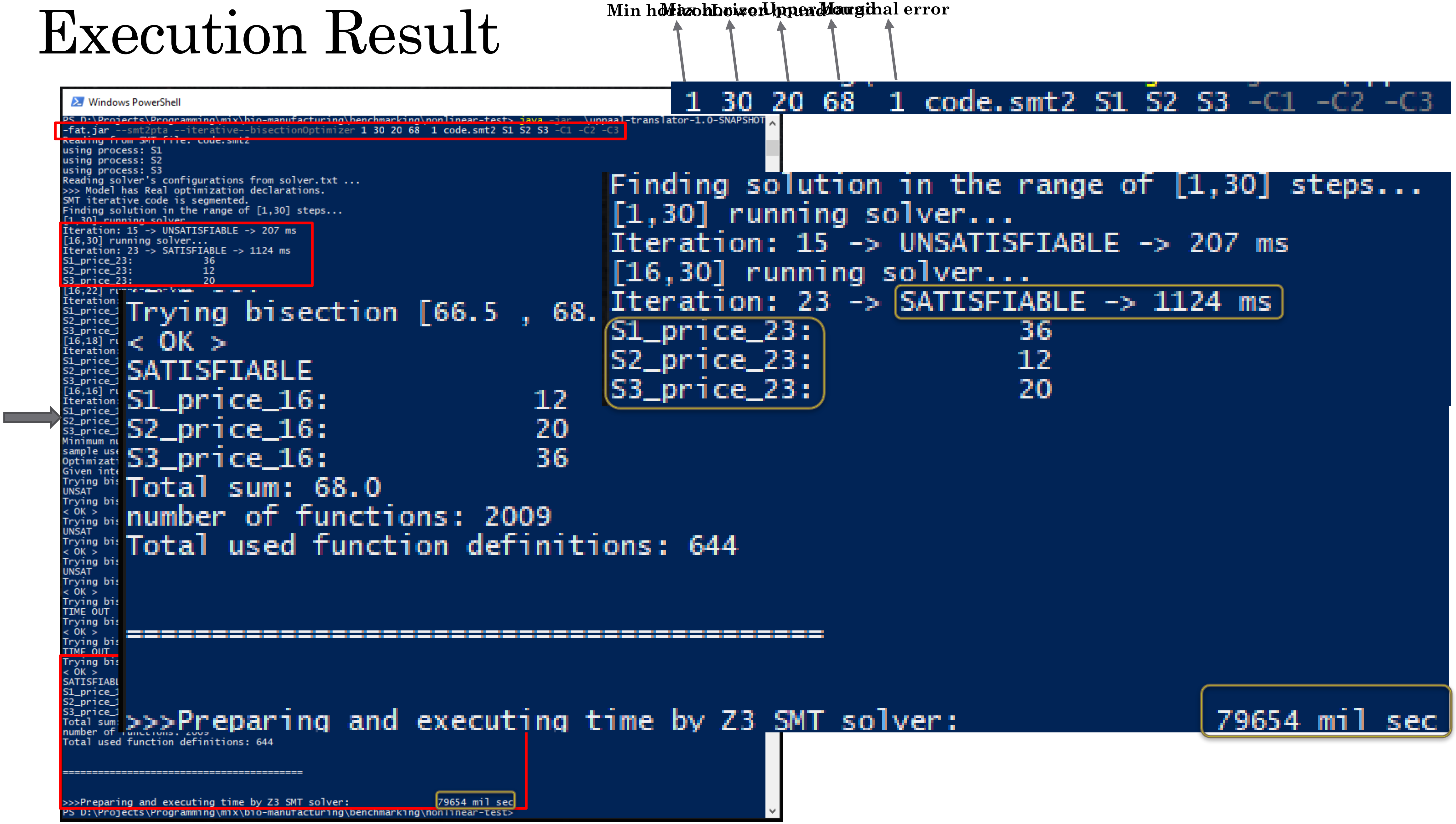
Client




```
chan startMsg1;
chan endMsg1;

chan startMsg2;
chan endMsg2;

chan startMsg3;
chan endMsg3;

S1 = Server( startMsg1, endMsg1, startMsg2, endMsg2, startMsg3, endMsg3 );
S2 = Server( startMsg1, endMsg1, startMsg2, endMsg2, startMsg3, endMsg3 );
S3 = Server( startMsg1, endMsg1, startMsg2, endMsg2, startMsg3, endMsg3 );

C1 = Client( startMsg1, endMsg1, startMsg2, endMsg2, startMsg3, endMsg3, 0, 10 );
C2 = Client( startMsg1, endMsg1, startMsg2, endMsg2, startMsg3, endMsg3, 0, 10 );
C3 = Client( startMsg1, endMsg1, startMsg2, endMsg2, startMsg3, endMsg3, 0, 10 );

system S1, S2, S3, C1, C2, C3;
```

srvTime = 1
maxJob = 3
count = 0
delay = 0
repeat = 10
rc = 0

Execution Result



Execution Result (cont')

Windows PowerShell

```
PS D:\Projects\Programming\mix\bio-manufacturing\benchmarking\nonlinear-test> java -jar .\uppaal-translator-1.0-SNAPSHOT
-fat.jar --smt2pta --iterative--bisectionOptimizer 16 16 51 68 1 code.smt2 S1 S2 S3 -C1 -C2 -C3
Reading from SMT file: code.smt2
using process: S1
using process: S2
using process: S3
Reading solver's configurations from solver.txt ...
>>> Model has Real optimization declarations.
SMT iterative code is segmented.
Finding solution in the range of [16,16] steps...
[16,16] running solver...
Iteration: 16 -> SATISFIABLE -> 650 ms
S1_price_16:            28
S2_price_16:            20
S3_price_16:            20
Minimum number of steps: 16
sample used SMT code saved as test.smt2
Optimization is activated.
Given interval = [51.0 , 68.0]
Trying bisection [51.0 , 59.5]
UNSAT
Trying bisection [59.5 , 68.0]
< OK >
Trying bisection [59.5 , 63.75]
UNSAT
Trying bisection [63.75 , 68.0]
< OK >
Trying bisection [63.75 , 65.875]
TIME OUT
Trying bisection [65.875 , 68.0]
< OK >
Trying bisection [65.875 , 66.9375]
TIME OUT
Trying bisection [66.9375 , 68.0]
< OK >
SATISFIABLE
S1_price_16:            12
S2_price_16:            20
S3_price_16:            36
Total sum: 68.0
number of functions: 2009
Total used function definitions: 644


========================================

>>>Preparing and executing time by Z3 SMT solver:            453003 mil sec
```


```
Trying bisection [66.9375 , 68.0]
< OK >
SATISFIABLE
S1_price_16:            12
S2_price_16:            20
S3_price_16:            36
Total sum: 68.0
```

Summary of Our Contribution

- We model tasks and machines as PTAs annotated with nonlinear cost functions on the clock variables.
- By modifying the framework proposed in [2], we translate the PTA reachability problem to an SMT formula whose models correspond to feasible schedules that satisfy a given cost constraint.
- For nonlinear cost functions, a bisection method can be used to compute optimal schedules.
- We demonstrate that the resulting framework based on SMT solvers can outperform UPPAAL CORA when the costs are linear functions of the clocks.
- Finally, we have released a publicly available tool called CEPTA2SMT available at: https://cpslab.assembla.com/spaces/bio-manufacturing/ .

[2] D. Bhave, S. N. Krishna, and A. Trivedi, "**On nonlinear prices in timed automata**," in V2CPS, ser. EPTCS, vol. 232, 2016, pp. 65–78.

Conclusion

- Used PTA to model bio-manufacturing scheduling problem.
  - UPPAAL CORA vs SMT based
  - **Lessons learned:**
    - modeling concurrent PTAs with many non-deterministic transitions in them significantly decreases the performance in both approaches.
    - the SMT approach scales better than the graph-based search algorithms.
    - the length of potential solutions (horizon) for a given problem is a critical performance factor.
- Our SMT based framework supports non-linear cost functions.
  - since the problem is undecidable in the general form, it is possible that the SMT solver may terminate without a solution.
- We incorporated a bisection method in our tool for dealing with non-linear cost functions when an upper bound and a lower bound exist.

Future Work

- Implementation:
  - Implement other design types for mutual exlusion and compare the computation results.
  - Use different translations for graph structures and compare the results.
  - Use other SMT based solvers that support Real theory.
  - Compare our results with classic methods such as MILP and Monte Carlo.
- Application:
  - Implement more realistic models in bio-pharmaceutical domain.

Thank You!

Acknowledgement:
This work was partially supported by NSF-CMMI 1829238.