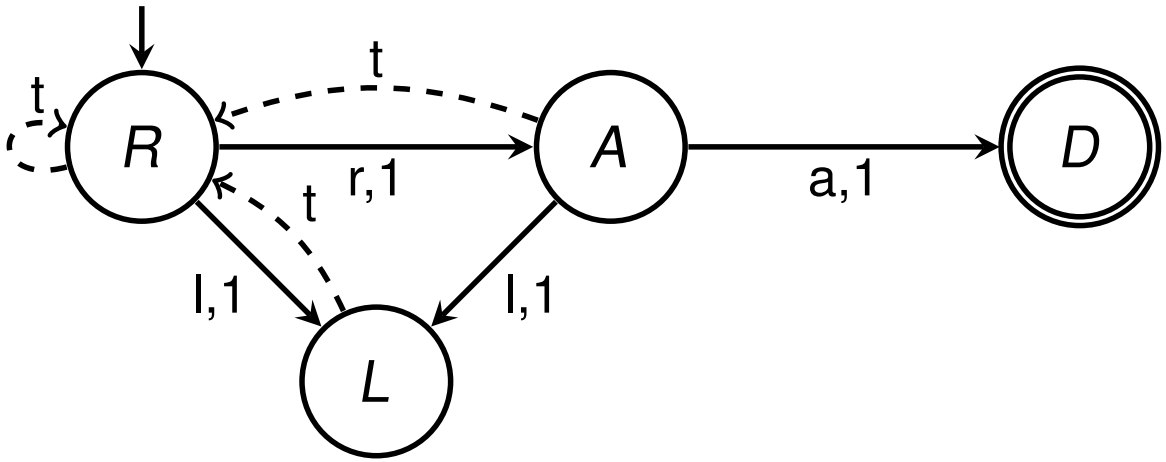
Efficient Timeout Synthesis in Fixed-Delay CTMC Using Policy Iteration

Ľuboš Korenčiak, Antonín Kučera, Vojtěch Řehák

Faculty of Informatics, Masaryk University, Brno, Czech Republic
{korenciak, kucera, rehak}@fi.muni.cz

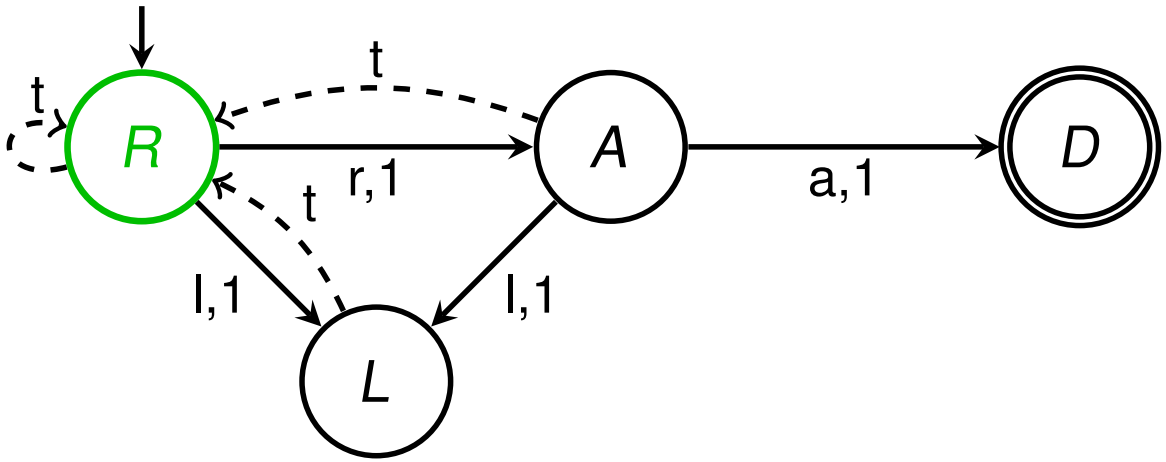
8th September 2016

Fixed-Delay Continuous-Time Markov Chain (fdCTMC)



equivalent to deterministic and stochastic Petri nets (DSPN)
Restriction: at most one concurrently active FD event
$\mathbf{d}(R) = 3$

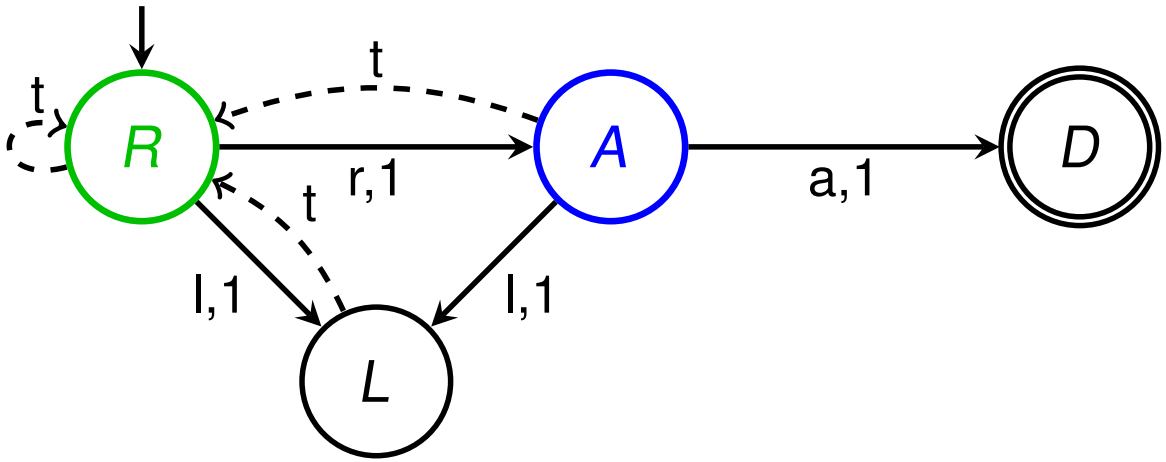
Fixed-Delay Continuous-Time Markov Chain (fdCTMC)



equivalent to deterministic and stochastic Petri nets (DSPN)
Restriction: at most one concurrently active FD event
$\mathbf{d}(R) = 3$

$\omega =$ (r=2, l=8, t=3),

Fixed-Delay Continuous-Time Markov Chain (fdCTMC)



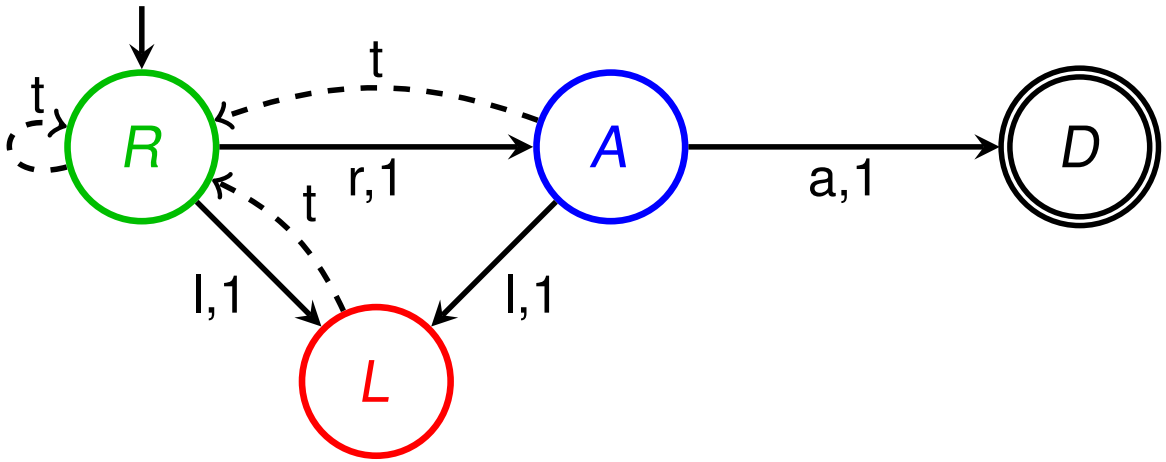
equivalent to deterministic and stochastic Petri nets (DSPN)

Restriction: at most one concurrently active FD event

$\mathbf{d}(R) = 3$

$\omega =$ (r=2, l=8, t=3), (a=6, l=1.5, t=1),

Fixed-Delay Continuous-Time Markov Chain (fdCTMC)



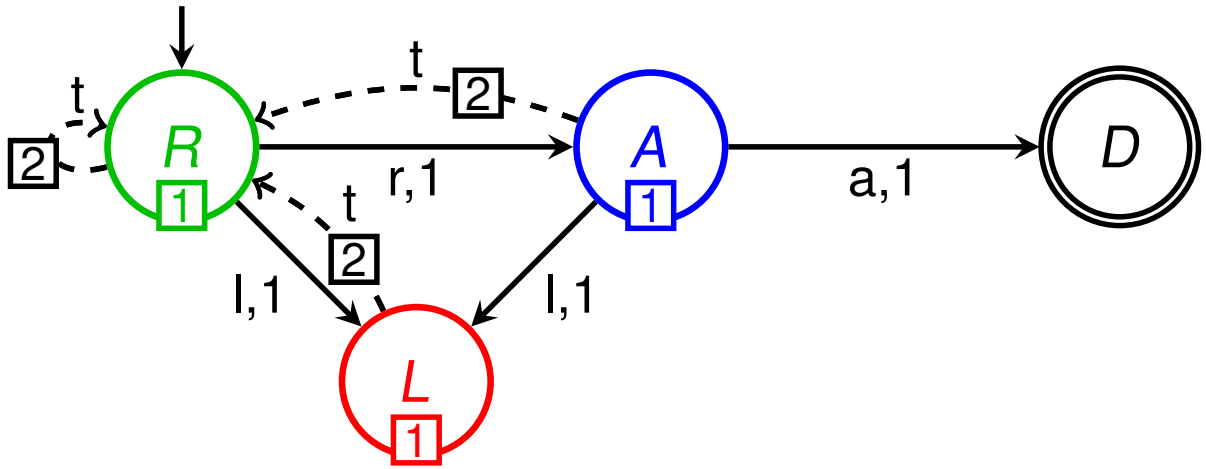
equivalent to deterministic and stochastic Petri nets (DSPN)

Restriction: at most one concurrently active FD event

$\mathbf{d}(R) = 3$

$\omega =$ (r=2, l=8, t=3), (a=6, l=1.5, t=1), (. . . , t=3), . . .

Fixed-Delay Continuous-Time Markov Chain (fdCTMC)



equivalent to deterministic and stochastic Petri nets (DSPN)

Restriction: at most one concurrently active FD event

$\mathbf{d}(R) = 3$

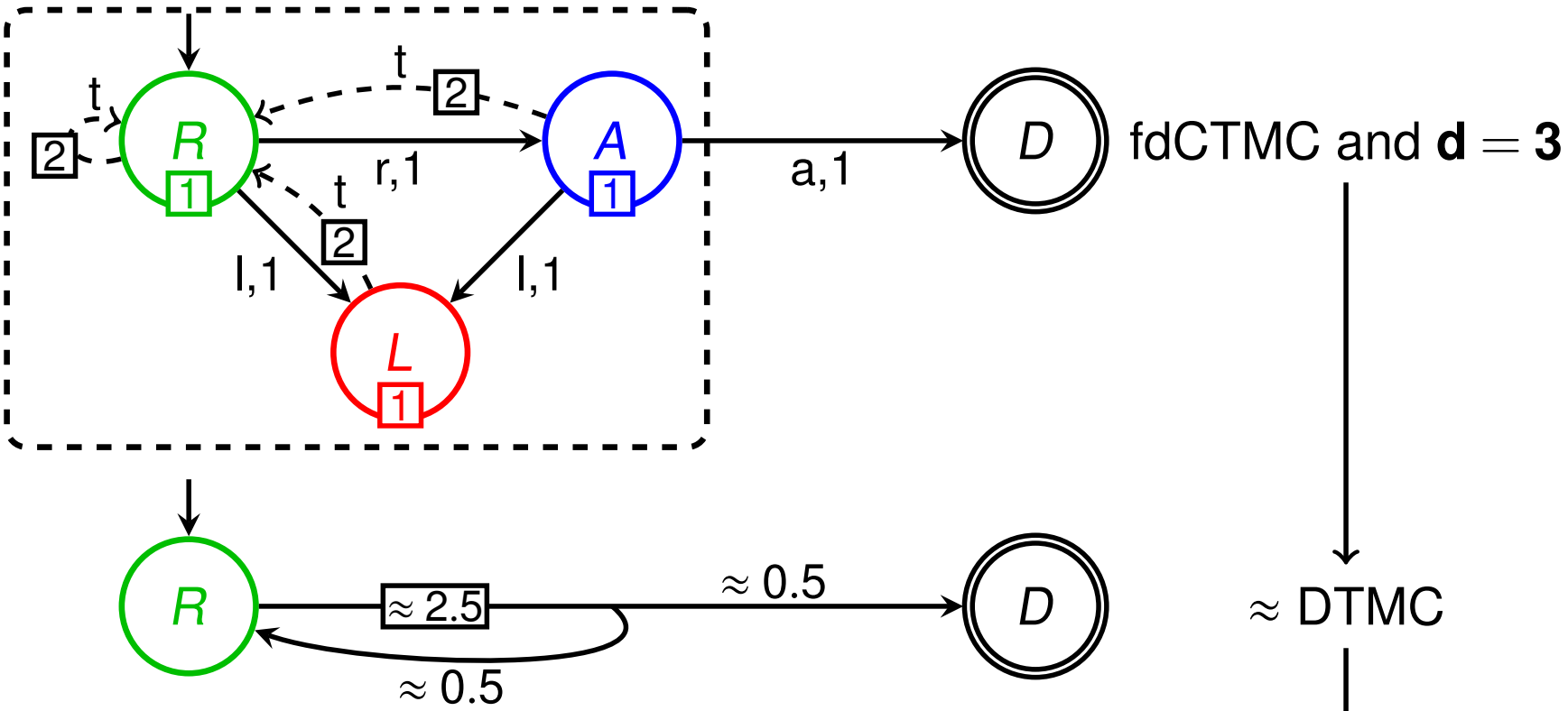
$\omega =$ (r=2, l=8, t=3), (a=6, l=1.5, t=1), (..., t=3), ...

$Cost(\omega) =$ 2 $+1+2$ $+3$ ...

Objective: $E_{\mathbf{d}}(Cost)$

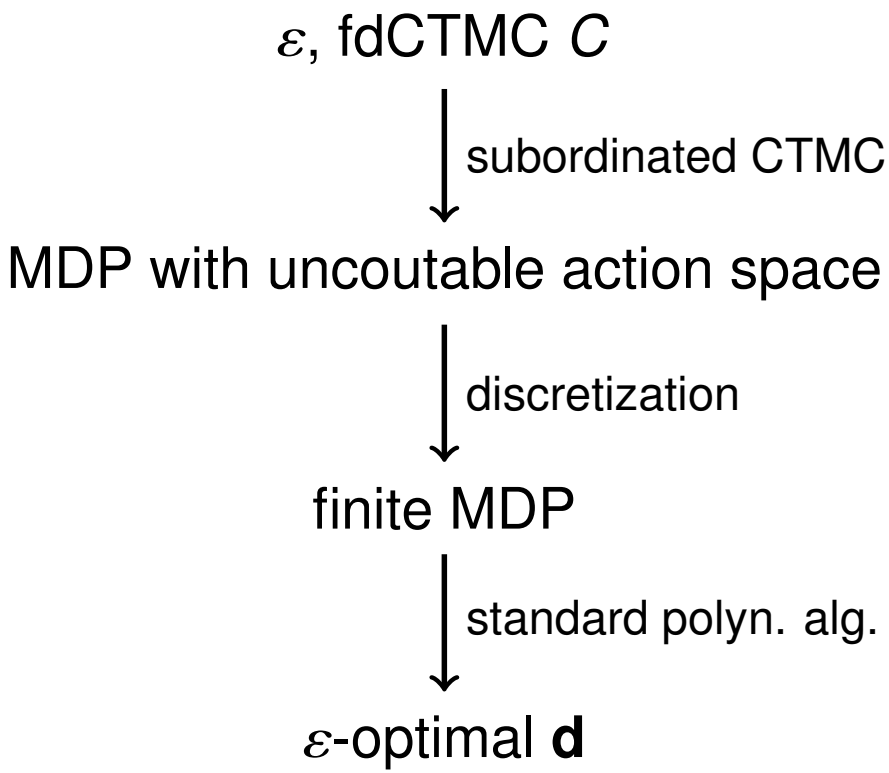
Algorithm for Computation of $E_{\mathbf{d}}(Cost)$

Method of subordinated Markov chain:



- distribution on transitions from FD territory,
- expected cost accumulated in FD territory,
- creation and solution of embedded DTMC.

Algorithm for Synthesis of FD



Published in QEST 2015.

Old Results

Synthesis was implemented and optimized in PRISM (iFM 2016).

ε	0.005	0.0025	0.00125	0.00100
Model	CPU time [s]			
rejuv	5.87	12.09	21.60	23.84
dpm2	58.22	121.15	234.58	248.52
dpm4	156.02	354.35	2197.10	2652.05
dpm6	259.76	532.47	3026.77	5124.10
dpm8	616.47	3142.44	22507.55	27406.62

The worst computation required 148 GiB of memory.

Policy Iteration

- method for solving MDP
- it optimizes one action at a time for each state s

$$\arg\min_{d \in D} \quad \mathbf{r}_d(s) + \sum_{s' \in S} P_d(s, s') \cdot \mathbf{v}(s')$$

- $r_d(s)$ and $P_d(s, s')$ can be expressed by an expolynomial function of a special form with one free variable d

$$e^{-\lambda d} \cdot Q(d)$$

- after differentiation it suffices to isolate all real roots of a bounded univariate polynomial function
- thus we compute actions on demand instead of computing all actions

Current and Future Work

Current results accepted to MASCOTS 2016

ε	0.005	0.0025	0.00125	0.00100
Model	CPU time [s]			
rejuv	5.87	12.09	21.60	23.84
dpm2	58.22	121.15	234.58	248.52
dpm4	156.02	354.35	2197.10	2652.05
dpm6	259.76	532.47	3026.77	5124.10
dpm8	616.47	3142.44	22507.55	27406.62

All tests finished in 3 CPU seconds.

Current and Future Work

Current results accepted to MASCOTS 2016

Num. of components	Error	Num. of states	CPU time [s]	
			symbolic	explicit
1	10^{-2}	4	2.91	4.4
2	10^{-2}	32	3.65	33.00
3	10^{-2}	192	6.02	1765.71
4	10^{-2}	1024	10.71	N/A
5	10^{-2}	5120	26.36	N/A
6	10^{-2}	24576	221.76	N/A

Current and Future Work

Current results accepted to MASCOTS 2016

Num. of components	Error	Num. of states	CPU time [s]	
			symbolic	explicit
1	10^{-2}	4	2.91	4.4
2	10^{-2}	32	3.65	33.00
3	10^{-2}	192	6.02	1765.71
4	10^{-2}	1024	10.71	N/A
5	10^{-2}	5120	26.36	N/A
6	10^{-2}	24576	221.76	N/A

Future work

- reimplement using BigDecimal
- mean payoff

Current and Future Work

Current results accepted to MASCOTS 2016

Num. of components	Error	Num. of states	CPU time [s]	
			symbolic	explicit
1	10^{-2}	4	2.91	4.4
2	10^{-2}	32	3.65	33.00
3	10^{-2}	192	6.02	1765.71
4	10^{-2}	1024	10.71	N/A
5	10^{-2}	5120	26.36	N/A
6	10^{-2}	24576	221.76	N/A

Future work

- reimplement using BigDecimal
- mean payoff

Thank you for your attention!