Improved Reachability Analysis in DTMC via Divide and Conquer

Songzheng Song[1] Lin Gui[1] Jun Sun[2] Yang Liu[3] Jin Song Dong[1]

[1]National University of Singapore

[2]Singapore University of Technology and Design

[3]Nanyang Technological University

iFM 2013

June 13, 2013

Outline

Outline

Motivation

Stochastic systems exist in many domains.

Motivation

Stochastic systems exist in many domains.

- Network: IPv4 Zeroconf
- Communication protocol: IEEE 802.3 CSMA/CD
- Biology: Cell cycles

Motivation

Stochastic systems exist in many domains.

- Network: IPv4 Zeroconf
- Communication protocol: IEEE 802.3 CSMA/CD
- Biology: Cell cycles

Motivation

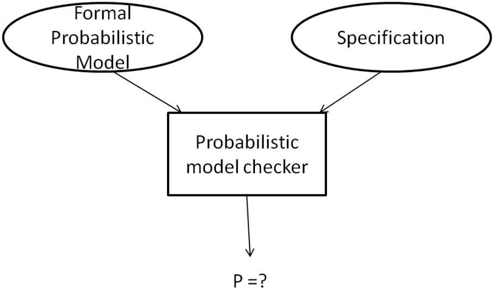
Stochastic systems exist in many domains.

- Network: IPv4 Zeroconf
- Communication protocol: IEEE 802.3 CSMA/CD
- Biology: Cell cycles

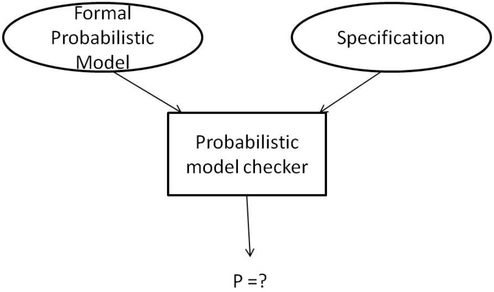
Formal analysis of stochastic systems is critical.

We focus on probabilistic model checking.

We focus on probabilistic model checking.



We focus on probabilistic model checking.



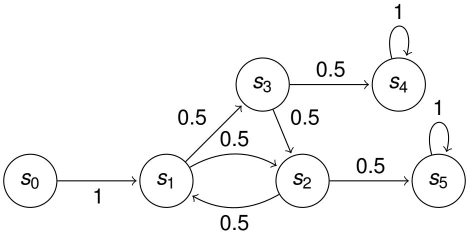
Discrete Time Markov Chain (DTMC) is a widely used formalism in probabilistic model checking.

Definition

A DTMC is a tuple (S, s_{init}, Pr, AP, L) where S is a countable set of states; $s_{init} \in S$ is the initial state; Pr is a function: $S \times S \rightarrow [0, 1]$ representing the transition relation, which satisfies $\forall\, s \in S$, $\Sigma_{s' \in S} Pr(s, s') = 1$; AP is a set of atomic propositions and L: $S \rightarrow 2^{AP}$ is a labeling function.

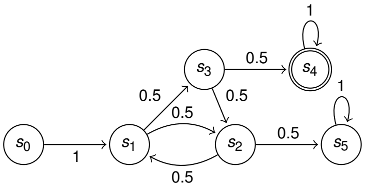
Definition

A DTMC is a tuple (S, s_{init}, Pr, AP, L) where S is a countable set of states; $s_{init} \in S$ is the initial state; Pr is a function: $S \times S \rightarrow [0, 1]$ representing the transition relation, which satisfies $\forall\, s \in S$, $\Sigma_{s' \in S} Pr(s, s') = 1$; AP is a set of atomic propositions and L: $S \rightarrow 2^{AP}$ is a labeling function.

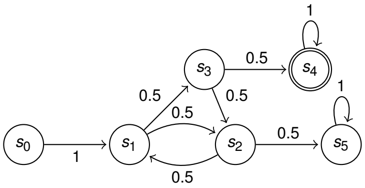


Reachability analysis plays a key role in DTMC verification, e.g., it is used to decide the probability of reaching certain disastrous state.

Reachability analysis plays a key role in DTMC verification, e.g., it is used to decide the probability of reaching certain disastrous state.

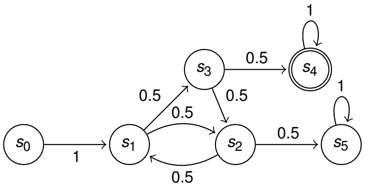


Reachability analysis plays a key role in DTMC verification, e.g., it is used to decide the probability of reaching certain disastrous state.



$$\begin{cases} p_0 = p_1 \\ p_1 = 0.5 \times p_2 + 0.5 \times p_3 \\ p_2 = 0.5 \times p_1 + 0.5 \times p_5 \\ p_3 = 0.5 \times p_2 + 0.5 \times p_4 \\ p_4 = 1 \\ p_5 = 0 \end{cases}$$

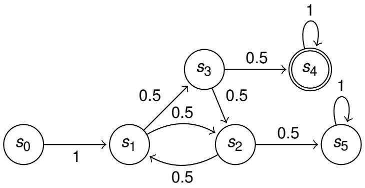
Reachability analysis plays a key role in DTMC verification, e.g., it is used to decide the probability of reaching certain disastrous state.



$$
\begin{cases}
p_0 = p_1 \\
p_1 = 0.5 \times p_2 + 0.5 \times p_3 \\
p_2 = 0.5 \times p_1 + 0.5 \times p_5 \\
p_3 = 0.5 \times p_2 + 0.5 \times p_4 \\
p_4 = 1 \\
p_5 = 0
\end{cases}
$$

1. Solving linear equations
2. Value iteration.

Reachability analysis plays a key role in DTMC verification, e.g., it is used to decide the probability of reaching certain disastrous state.



$$\begin{cases} p_0 = p_1 \\ p_1 = 0.5 \times p_2 + 0.5 \times p_3 \\ p_2 = 0.5 \times p_1 + 0.5 \times p_5 \\ p_3 = 0.5 \times p_2 + 0.5 \times p_4 \\ p_4 = 1 \\ p_5 = 0 \end{cases}$$

1. Solving linear equations
2. Value iteration.

We combine their advantages together!

Outline

1. Motivation
2. Background
3. Our Approach
4. Evaluations
5. Conclusion and Future Work

Strongly Connected Component

SCC

1. Those maximal sets of states mutually connected.
2. An SCC is called *trivial* if it just has one state without a self-loop.
3. An SCC is *nontrivial* iff it is not trivial.
4. A DTMC is *acyclic* iff it only has *trivial* SCCs.

Strongly Connected Component

SCC

1. Those maximal sets of states mutually connected.
2. An SCC is called *trivial* if it just has one state without a self-loop.
3. An SCC is *nontrivial* iff it is not trivial.
4. A DTMC is *acyclic* iff it only has *trivial* SCCs.

$$
\begin{tikzcd}
 & & & s_3 \arrow[r, "0.5"] \arrow[d, "0.5"] & s_4 \arrow[loop above, "1"] & \\
s_0 \arrow[rr, "1"'] & & s_1 \arrow[ur, "0.5"] \arrow[r, bend left, "0.5"] & s_2 \arrow[l, bend left, "0.5"] \arrow[rr, "0.5"] & & s_5 \arrow[loop above, "1"]
\end{tikzcd}
$$

Nontrival SCCs : $\{\{s_1, s_2, s_3\}, \{s_4\}, \{s_5\}\}$

Input and Output

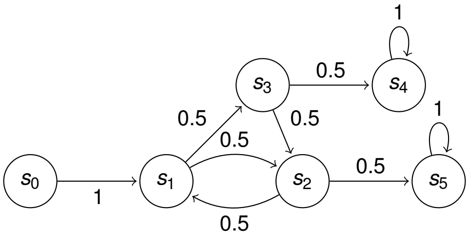
in a DTMC $\mathcal{M} = (S, s_{init}, Pr, AP, L)$, given a group of states $\mathcal{D} \subseteq S$, the input states of $\mathcal{D}$ are defined as the states in $\mathcal{D}$ having incoming transitions from states outside $\mathcal{D}$; the output states of $\mathcal{D}$ are defined as states outside $\mathcal{D}$ which have incoming transitions from states in $\mathcal{D}$.

$$Inp(\mathcal{D}) = \{s' \in \mathcal{D} \mid \exists\, s \in S \backslash \mathcal{D}.Pr(s, s') > 0\}$$
$$Out(\mathcal{D}) = \{s' \in S \backslash \mathcal{D} \mid \exists\, s \in \mathcal{D}.Pr(s, s') > 0\}$$

Input and Output

in a DTMC $\mathcal{M} = (S, s_{init}, Pr, AP, L)$, given a group of states $\mathcal{D} \subseteq S$, the input states of $\mathcal{D}$ are defined as the states in $\mathcal{D}$ having incoming transitions from states outside $\mathcal{D}$; the output states of $\mathcal{D}$ are defined as states outside $\mathcal{D}$ which have incoming transitions from states in $\mathcal{D}$.

$$Inp(\mathcal{D}) = \{s' \in \mathcal{D} \mid \exists s \in S \backslash \mathcal{D}. Pr(s, s') > 0\}$$
$$Out(\mathcal{D}) = \{s' \in S \backslash \mathcal{D} \mid \exists s \in \mathcal{D}. Pr(s, s') > 0\}$$



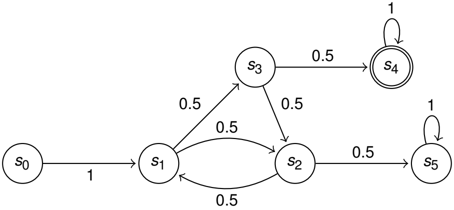
If $\mathcal{D} = \{s_1, s_2, s_3\}$, then $Inp(\mathcal{D}) = \{s_1\}$, $Out(\mathcal{D}) = \{s_4, s_5\}$.

Abstraction

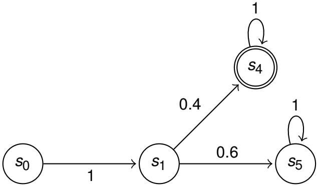
$\mathcal{D}$ can be abstracted by calculating the transition probability from *Inp*$(\mathcal{D})$ to *Out*$(\mathcal{D})$.

Abstraction

$\mathcal{D}$ can be abstracted by calculating the transition probability from *Inp*$(\mathcal{D})$ to *Out*$(\mathcal{D})$.



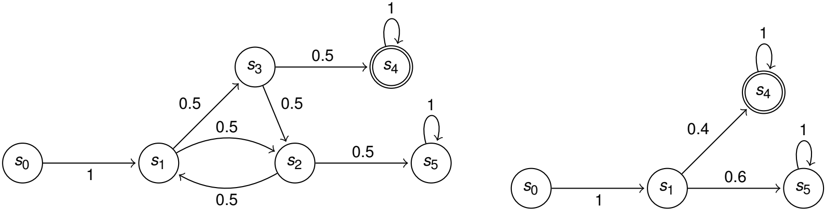
(a) Before Abstraction



(b) After Abstraction

Abstraction

$\mathcal{D}$ can be abstracted by calculating the transition probability from *Inp*$(\mathcal{D})$ to *Out*$(\mathcal{D})$.



(a) Before Abstraction (b) After Abstraction

$$[A \mid B] = \left[\begin{array}{ccc|cc} 1 & -0.5 & -0,5 & 0 & 0 \\ 0 & 1 & -0.5 & 0 & -0.5 \\ 0 & -0.5 & 1 & -0.5 & 0 \end{array}\right]$$

$$[I \mid C] = \left[\begin{array}{ccc|cc} 1 & 0 & 0 & -0.4 & -0.6 \\ 0 & 1 & 0 & -0.2 & -0.8 \\ 0 & 0 & 1 & -0.6 & -0.4 \end{array}\right]$$

$\mathcal{D}$ is an SCC and $|Out(\mathcal{D})| \leq 1$

$\mathcal{D}$ is an SCC and $|Out(\mathcal{D})| \leq 1$

- If $|Out(\mathcal{D})| = 0$, $\mathcal{D}$ has no outgoing transitions, then no matter whether $\mathcal{D}$ has target states or not, we do not need to solve $\mathcal{D}$. If $\mathcal{D} \cap G = \phi$, it is obvious that all states in $\mathcal{D}$ has probability 0 to reach G; otherwise, it is trivial to show that all states in $\mathcal{D}$ has probability 1 to reach G.

$\mathcal{D}$ is an SCC and $|Out(\mathcal{D})| \leq 1$

- If $|Out(\mathcal{D})| = 0$, $\mathcal{D}$ has no outgoing transitions, then no matter whether $\mathcal{D}$ has target states or not, we do not need to solve $\mathcal{D}$. If $\mathcal{D} \cap G = \phi$, it is obvious that all states in $\mathcal{D}$ has probability 0 to reach G; otherwise, it is trivial to show that all states in $\mathcal{D}$ has probability 1 to reach G.
- If $|Out(\mathcal{D})| = 1$, assume s_{out} is the output state. All paths entering $\mathcal{D}$ will leave it eventually. Therefore, for every $s_i \in Inp(\mathcal{D})$, the probability of paths entering $\mathcal{D}$ via s_i, staying in $\mathcal{D}$ and exiting $\mathcal{D}$ to s_{out} should be 1. So $\mathcal{D}$ can be abstracted directly.

Outline

Instead of directly calculating the probability from initial states to targets, we divide the whole state space into **several partitions**, and solve them individually to eliminate loops.

Instead of directly calculating the probability from initial states to targets, we divide the whole state space into **several partitions**, and solve them individually to eliminate loops.

1. Find SCCs;
2. Abstract SCCs.

Instead of directly calculating the probability from initial states to targets, we divide the whole state space into **several partitions**, and solve them individually to eliminate loops.

1. Find SCCs;
2. Abstract SCCs.

If the SCCs are huge, what should we do?

Instead of directly calculating the probability from initial states to targets, we divide the whole state space into **several partitions**, and solve them individually to eliminate loops.

1. Find SCCs;
2. Abstract SCCs.

If the SCCs are huge, what should we do?

- Divide each SCC having a large number of states to several smaller partitions.
- Abstract each partition.

Instead of directly calculating the probability from initial states to targets, we divide the whole state space into **several partitions**, and solve them individually to eliminate loops.

1. Find SCCs;
2. Abstract SCCs.

If the SCCs are huge, what should we do?

- Divide each SCC having a large number of states to several smaller partitions.
- Abstract each partition.

These steps could be repeated.

Our algorithm

input : A DTMC $\mathcal{M} = (S, s_{init}, Pr, AP, L)$, target states $G \subseteq S$ and a Bound B
output: $\mathcal{P}(s_{init} \models \Diamond G)$

Let $\mathcal{C}$ be the set of all nontrivial SCCs in $\mathcal{M}$;
while $|\mathcal{C}| > 0$ **do**
    Let $\mathcal{D} \in \mathcal{C}$;
    if $|\mathcal{D}| \leq B \vee |Out(\mathcal{D})| \leq 1$ **then**
        $Abs(\mathcal{D})$ and $\mathcal{C} \leftarrow \mathcal{C} \backslash \mathcal{D}$
    else
        Divide $\mathcal{D}$ into a set of partitions denoted as $\mathcal{A}$;
        for *each* $\mathcal{E} \in \mathcal{A}$ **do** $Abs(\mathcal{E})$;
        Let $\mathcal{D}'$ be the set of remaining states in $\mathcal{D}$;
        if $|\mathcal{D}'| \leq B \vee |\mathcal{D}'| = |\mathcal{D}|$ **then**
            $Abs(\mathcal{D}')$ and $\mathcal{C} \leftarrow \mathcal{C} \backslash \mathcal{D}$
        else
            Let $\mathcal{C}_{\mathcal{D}'}$ be the set of all nontrivial SCCs in $\mathcal{D}'$; $\mathcal{C} \leftarrow (\mathcal{C} \backslash \mathcal{D}) \cup \mathcal{C}_{\mathcal{D}'}$;
return $VI(\mathcal{M}, G)$;

Theorems

1. This algorithm always terminates;
   - This can be proved via the total number of states in $\mathcal{C}$ is decreasing.

Theorems

1. This algorithm always terminates;
   - This can be proved via the total number of states in $\mathcal{C}$ is decreasing.
2. The abstract has no affect of final results.
   - This has been proved in previous work.

E. Ábrahám, N. Jansen, R. Wimmer, J.-P. Katoen, B. Becker
DTMC Model Checking by SCC Reduction.
In *QEST*, pages 37-46, 2010.

Divide Strategy

The divide-and-conquer approach's efficiency is highly dependent on how an SCC is divided. Assume $\mathcal{E}$ is a partition.

1. $\mathcal{E}$ should not have too many states.
2. $\mathcal{E}$ should not have too few states.
3. The smaller $|Out(\mathcal{E})|$ is, the better reduction is achieved.

In practice, the structure of $\mathcal{D}$ could be arbitrary. This increases the difficulty of finding a general strategy for all cases.

Simple division method

The simplest division method is to try to set each partition to have the same number of states.

Simple division method

The simplest division method is to try to set each partition to have the same number of states.

- Pros
  - The number of states in each partition is easily controlled.
  - It can be very efficient in cases where the states in $\mathcal{D}$ has few transitions.

Simple division method

The simplest division method is to try to set each partition to have the same number of states.

- Pros
  - The number of states in each partition is easily controlled.
  - It can be very efficient in cases where the states in $\mathcal{D}$ has few transitions.
- Cons
  - Cannot control the number of output states of each partition.
  - A predefined B may not be suitable for $\mathcal{D}$'s structure.

Improved division method

Try to automatically decide the number of states in each partition. We set a lower bound B_L and an upper bound B_U for each partition.

1. B_L states will be grouped into $\mathcal{E}$, and $|Out(\mathcal{E})|$ is recorded.
2. Some states in $Out(\mathcal{E})$ are added into $\mathcal{E}$, and $|Out(\mathcal{E})|$ is updated (this step can be repeated).
3. The number of states in $\mathcal{E}$ should be always below B_U.

Improved division method

Try to automatically decide the number of states in each partition. We set a lower bound B_L and an upper bound B_U for each partition.

1. B_L states will be grouped into $\mathcal{E}$, and $|Out(\mathcal{E})|$ is recorded.
2. Some states in $Out(\mathcal{E})$ are added into $\mathcal{E}$, and $|Out(\mathcal{E})|$ is updated (this step can be repeated).
3. The number of states in $\mathcal{E}$ should be always below B_U.

- Pros
  - The number of states in $\mathcal{E}$ is under control.
  - The outputs of $\mathcal{E}$ are also manageable.

- Our approach is independent from the topological order of SCCs.
- Even each partition in one SCC is also independent from others.

Parallel computation is suitable.

Outline

PAT

We have implemented the algorithm into our model checking framework PAT, which supports explicit probabilistic model checking and can be freely downloaded at

http://www.patroot.com.

- 10+ modules
- 2600+ downloads from 60+ countries
- various OS supported

Parameters

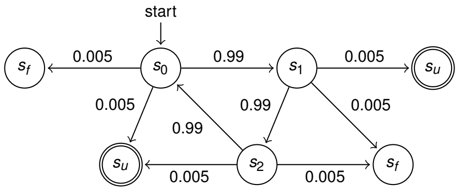
In these experiments, we use the improved dividing strategy. Parameters are set according to our experience.

- $B = 300$: an SCC with more than 300 states should be divided.
- $B_L = 100$, $B_U = 150$: each group has states between 100 and 150.

We show the improvement via comparing to PAT itself, which was based on **value iteration** method previously.

A simple case

$N+2$ states $\{s_0, s_1, ..., s_{N-1}, s_u, s_f\}$ exist. Each state $s_i, i \in [0..N-1]$, has probability 0.99 to reach $s_{(i+1)\%n}$, and also has probability 0.005 to reach s_u and s_f separately.



System	PAT (w)			PAT (w/o)		
	Prob	Time (s)	Memory (MB)	Prob	Time (s)	Memory (MB)
N = 500	0.5	0.03	71	0.49987	0.5	24
N = 5000	0.5	0.3	83	0.49987	5.5	63
N = 50000	0.5	2.6	151	0.49987	125.2	111
N = 500000	0.5	29.7	885	0.49987	1612.8	838

Benchmark systems

- BSS: Basic simple strategy
- ESS: Extend simple strategy
- CS: Randomized consensus

System	States	Prob	PAT (w)			PAT (w/o)		
			Time (s)	BMR	Memory (MB)	Time (s)	BMR	Memory (MB)
BSS (4)	4196	1	1.3	92.3%	39	0.2	50%	35
BSS (5)	49572	1	3.5	94.3%	297	4.4	11.4%	142
BSS (6)	605890	1	41.4	72.7%	1297	105.3	6.7%	417
BSS (7)	7462639	1	1671	30.1%	6350	2073.1	4.1%	5039
ESS (6, 4)	32662	1	1.4	92.8%	16.3	2.7	14.8%	5.6
ESS (6, 5)	162945	1	6.7	91.1%	48.5	11.4	16.7%	13.9
ESS (7, 5)	463460	1	27.9	84.9%	310	75.8	7.1%	292
ESS (8, 5)	1114480	1	70.5	74.7%	619	278.5	6.1%	643
ESS (8, 6)	6476524	1	438.0	68.5%	4209	1168.1	7.5%	3904
CS (4, 3)	4966	0.023	0.8	87.5%	45	2.4	8.3%	35
CS (6, 3)	34529	0.023	15.7	81.5%	214	124.1	0.9%	108
CS (6, 4)	45281	0.015	24.8	86.7%	324	243.8	0.6%	81
CS (6, 5)	56033	0.012	38.6	91.2%	312	432.1	0.4%	104
CS (7, 4)	99265	0.014	102.3	87.6%	1062	983.1	0.4%	97
CS (7, 5)	122785	0.011	161.7	92.1%	1145	1384.8	0.3%	97
CS (7, 6)	146305	0.01	245.5	94.9%	1404	2409.5	0.2%	156
CS (8, 4)	200083	0.013	585.1	93.4%	1974	-	-	-

- BMR: The rate of model building (BM) time to total time.

Outline

- Conclusion
  1. We proposed a divide-and-conquer approach to speed up reachability analysis of DTMCs.
  2. We have implemented our approach in PAT.
- Future work
  1. Find more efficient division strategies.
  2. Extend our approach to Markov Decision Processes (MDP).

References I

E. Ábrahám, N. Jansen, R. Wimmer, J.-P. Katoen, B. Becker
DTMC Model Checking by SCC Reduction.
In *QEST*, pages 37-46, 2010.

M.E. Andrés, P. R. D' Argenio, and P. V. Rossum.
Significant Diagnostic Counterexamples in Probabilistic Model Checking.
HVC, pages 129-148, 2008.

F. Ciesinski, C. Baier, M. Größer and J. Klein.
Reduction Techniques for Model Checking Markov Decision Processes
QEST, pages 45-54, 2008.

References II

M. Kwiatkowska, D. Parker, and H. Qu.
Incremental Quantitative Verification for Markov Decision Processes.
DSN, pages 359-370, 2011.

J. P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen.
The Ins and Outs of the Probabilistic Model Checker MRMC.
QEST, pages 167-176, 2009.

C. Baier and J. Katoen.
Principles of Model checking.
The MIT Press, 2008.

References III

R. E. Tarjan.
Depth-First Search and Linear Graph Algorithms.
SIAM J. Comput., 1(2):146¨C 160, 1972.

J. Sun, Y. Liu, J. S. Dong, and J. Pang.
PAT: Towards Flexible Verification under Fairness.
CAV, pages 709¨C714, 2009.

J. Sun, S. Song and Y. Liu
Model Checking Hierarchical Probabilistic Systems.
ICFEM, pages 388-403, 2010.

THANK YOU!

www.patroot.com