[第2部] 自己適応システムと モデル検査の応用

2021.02.18　第64回OACIS技術座談会

大阪大学 大学院情報科学研究科　中川 博之

自己紹介: 中川 博之

- 大阪大学 大学院情報科学研究科
  ディペンダビリティ工学講座 (土屋研究室) 准教授

- 専門：ソフトウェア工学，知的システム
  - 要求工学, ゴールモデル
  - 自己適応システム, エージェント技術
  - ソフトウェア進化
- 書籍：
  - 大須賀昭彦, 田原康之, 中川博之, 川村隆浩，マルチエージェントによる自律ソフトウェア設計・開発，コロナ社，2017.
- 学会活動：
  - 電子情報通信学会　知能ソフトウェア工学研究会 (KBSE)：委員長
  - 情報処理学会　ソフトウェア工学研究会(SIG-SE) ：運営委員
  - ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS2021)：Local Arrangements Chairs

増え続けるシステム障害

- ANA 国内線旅客システム (2016年3月22日)
  - 全国49空港で搭乗手続きができない
    719便, 72,000人以上に影響, 予約もできない
    - 原因: スイッチ(ネットワーク機器)の不具合

- みずほ銀行 (2011年3月15日～24日)
  - 全ATM休止, 振込み未入金, 二重振込みも見つかる
    - 原因: 義援金口座の上限値をオーバー, 異常終了

- 東証システム (2020年10月1日, 2012年2月2日, 2005年11月1日)
  - 2020年10月1日: 終日株式売買が停止
    - 原因: NASの製品マニュアルに不備, バックアップ機への自動切り替えが5年間オフになっていた
  - 2012年2月2日: 241銘柄(全体の一割)の取引を停止
    - 原因: サーバ障害時の切り替えミス (人的要因)
  - 2005年11月1日: 午前中の取引が全面停止
    - 原因: システム増強時の移行体制の不備 (人的要因)

システム障害を無くすには？

- 原因を総括すると …
  - システム構成要素 (SW, HW共に) の増加, 複雑化
    - SWのバグ, HW機器の故障件数の増加
    - システム全体を詳細に把握することが困難に
  - 人的要因
    - 例え訓練されていてもミスを完全になくすことは困難

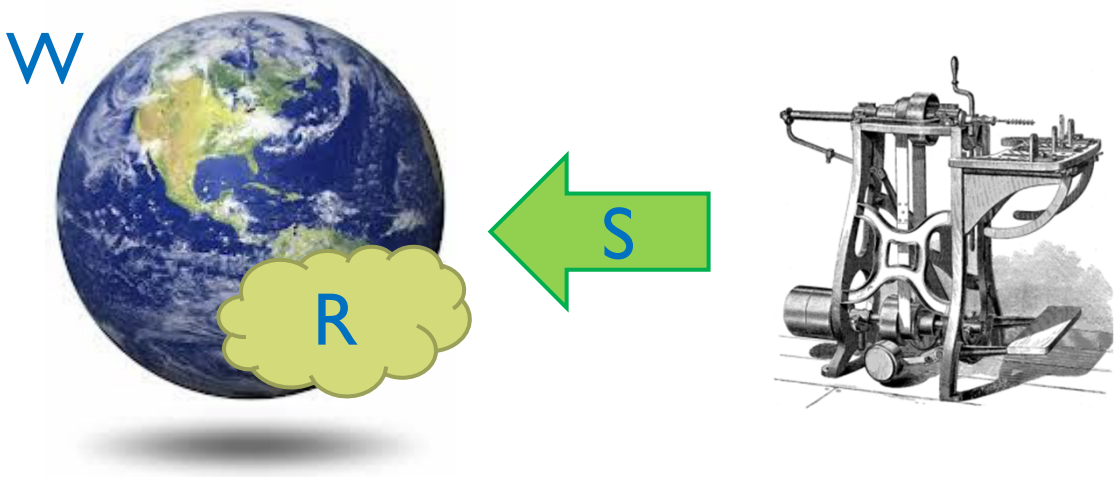
- では, どうすればよいか？？

→ システム自身が, 振る舞いや構成を切替えられるとよい

要求とソフトウェア仕様との関係

- Requirements and specifications [Zave97]
  - R = requirements, S = specifications, W = world

$$W \wedge S \vdash R$$



5 [Zave97] Pamela Zave and Michael Jackson. 1997, Four dark corners of requirements engineering. ACM Transactions on Software Engineering and Methodology, Vol. 6, No. 1, 1997.

ソフトウェア進化と適応

- Requirements and specifications [Zave97]
  - R = requirements, S = specifications, W = world

$$W \wedge S \vdash R$$

- ソフトウェア進化 (software evolution):

$$W \wedge S \vdash R \xrightarrow{R \to R'} W \wedge S \not\vdash R' \xrightarrow{S \to S'} W \wedge S' \vdash R'$$

- 適応 (adaptation):

$$W \wedge S \vdash R \xrightarrow{W \to W'} W' \wedge S \not\vdash R \xrightarrow{S \to S'} W' \wedge S' \vdash R$$

適応の例

- R:　フロアが綺麗な状態である
- W: ほこりが落ちている
- S:　吸引による清掃
- W′: 牛乳がこぼれている
- S′:　拭き掃除による清掃

$$W \wedge S \vdash R$$

$$W' \wedge S \not\vdash R$$

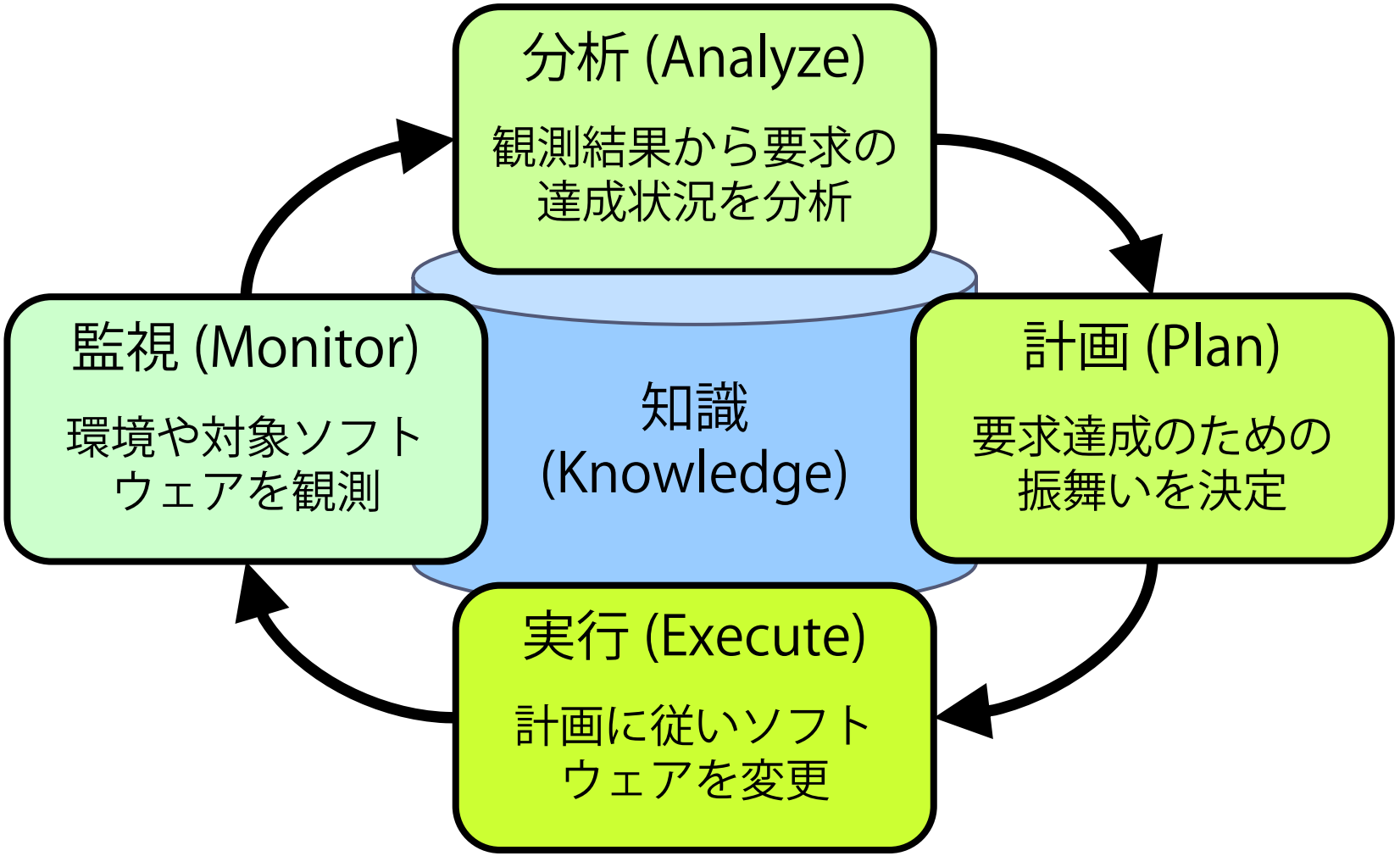
$$W' \wedge S' \vdash R$$

https://www.irobot-jp.com/

自己適応性

- 自己適応性 (self-adaptivity)：
  - $W \wedge S \vdash R \Rightarrow W' \wedge S' \vdash R$
    を自らが実現できる性質
- 以下の機能を有する必要がある
  - 環境を観測することができる (Monitor)
  - 現在の状況を分析することができる (Analyze)
  - 望ましい振る舞いを決定することができる (Plan)
  - 振舞いを切り替えることができる (Execute)

MAPE loop /MAPE-K loop



自己適応システム

- 以下の分野での実用化が期待されている
- 大規模サーバ管理
  - クラウドサーバ
  - IBMのAutonomic Computing

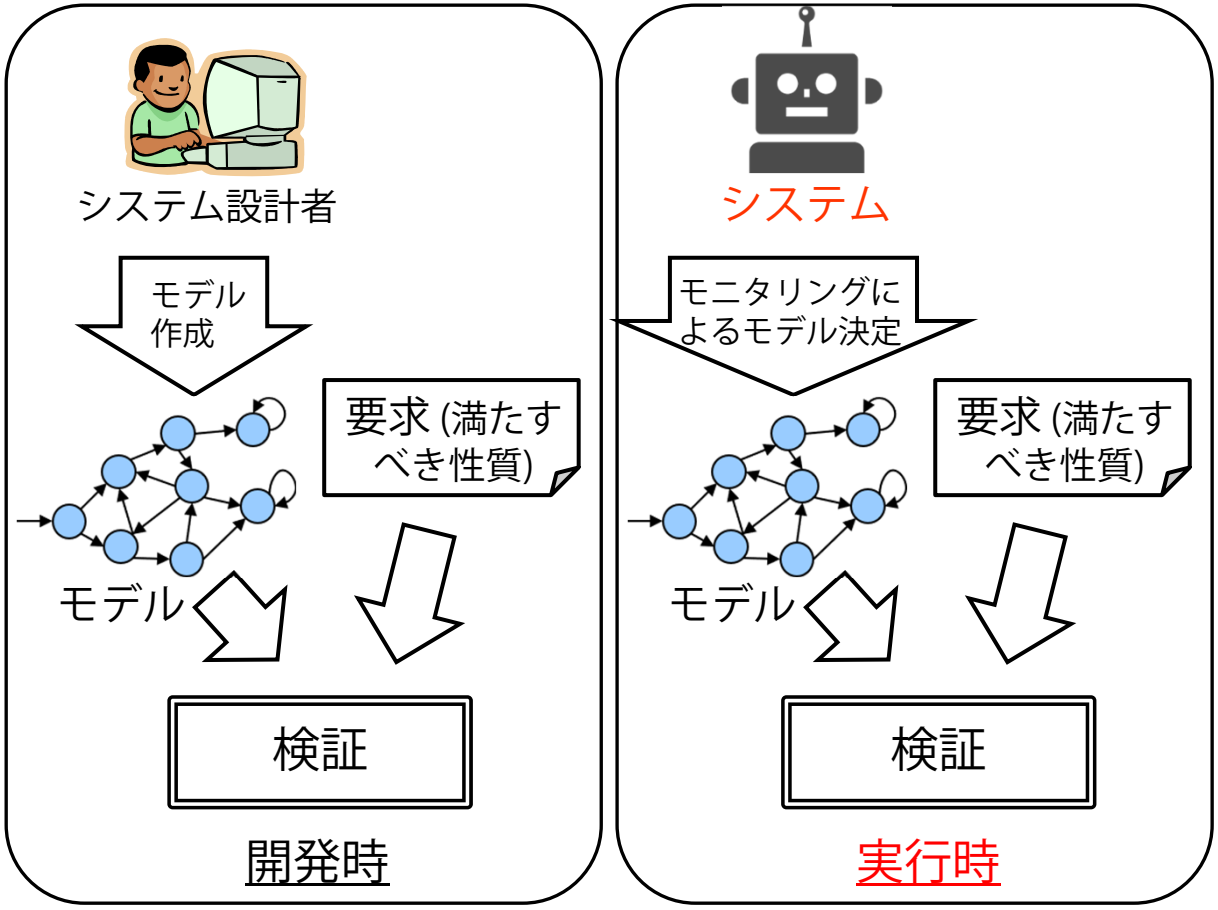
- CPS, IoTシステム
  - ワイヤレスセンサネットワーク
  - DeltaIoT: ネットワークのメンテナンス作業を自動化
  - ドローン制御

ソフトウェア工学各分野からのアプローチ

- 既存の各要素技術を拡張させる形で発展

- 要求工学からのアプローチ
  - 要求の監視，ゴールモデルの活用，不確かさ
- ソフトウェア・アーキテクチャからのアプローチ
  - コンポーネント，MAPEループの分散配置
- 検証技術からのアプローチ
  - 動的検証
- 実行環境としてのアプローチ
  - プログラミングフレームワーク

検証技術



従来の検証　　動的検証

不確かさ(Uncertainty)

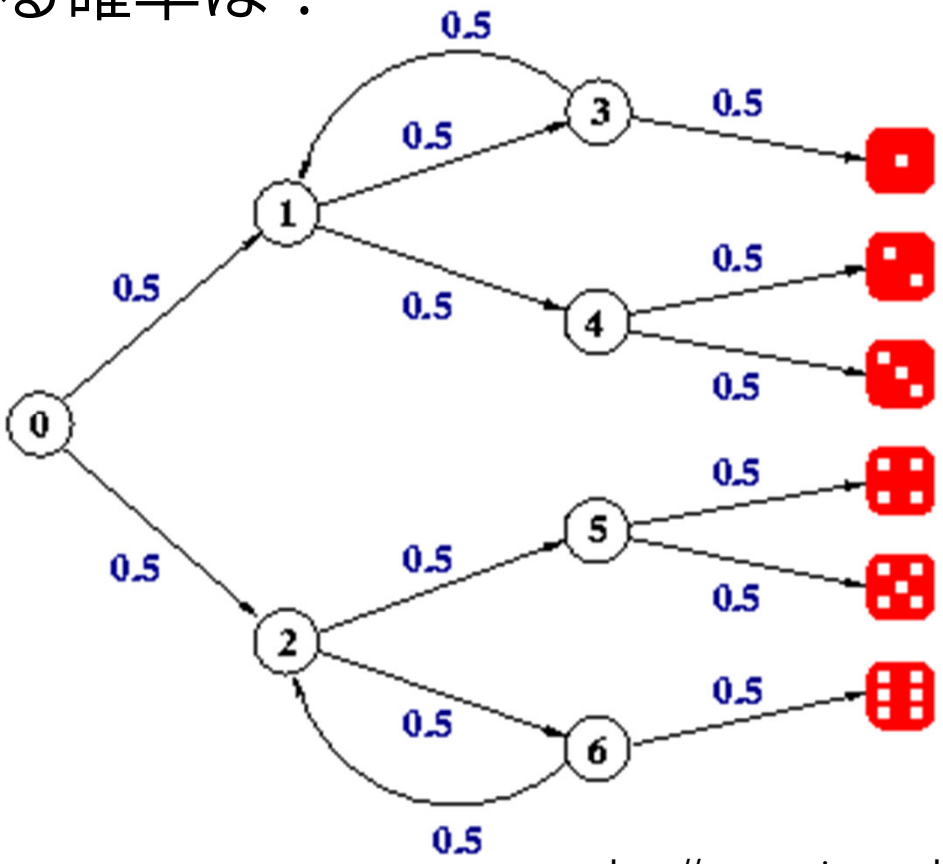
- 自己適応システムでは，環境の不確かさを扱う必要がある
  - 実行時の環境に依存する
    - 開発時には未知
  - 必ずこうなるとは断定できない
    - システムの振舞いが成功するかもわからない
  - …

→ 確率の導入

- 確率に基づいた状態遷移モデル
- 不確かさを表現, 不確かさに基づいた定量的な検証が可能

サイコロ[KY76]

- コイントスによりサイコロの目を決定
- 各目の出る確率は？

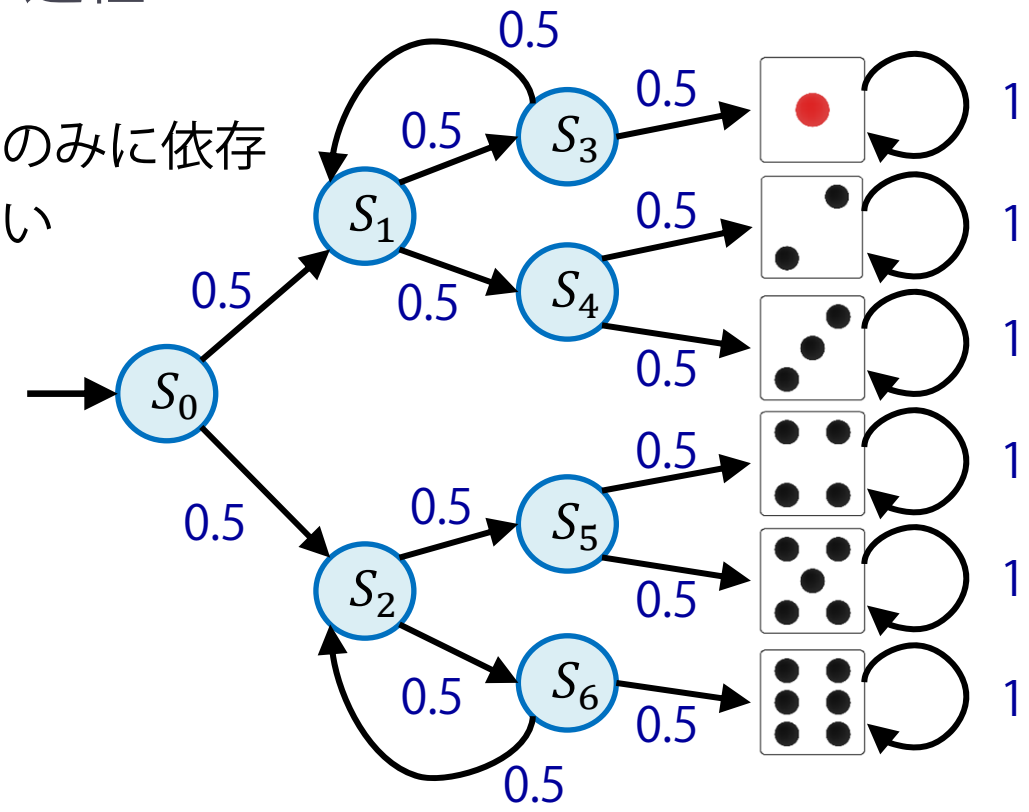


http://www.prismmodelchecker.org/tutorial/die.php

[KY76] D. Knuth and A. Yao, The complexity of nonuniform random number generation. In Algorithms and Complexity: New Directions and Recent Results, Academic Press. 1976.

サイコロのモデル化

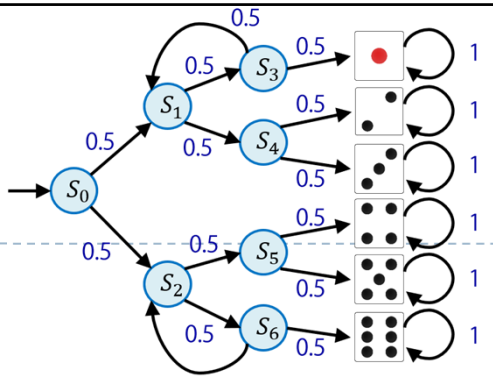
- DTMC (Discrete Time Markov Chains)
  離散時間マルコフ連鎖
  - 時間を離散とした確率過程
  - マルコフ連鎖
    - 将来の状態が現在状態のみに依存
    - 過去の状態に依存しない



https://www.prismmodelchecker.org/lectures/pmc/

PRISM

https://www.prismmodelchecker.org/



- 確率的モデル検査ツール

PRISM 4.6

File Edit Model Properties Simulator Log Options

PRISM Model File: <Untitled>*

Model: <Untitled>
Type: DTMC
Modules

```
1  dtmc
2
3  module die
4
5          // local state
6          s : [0..7] init 0;
7          // value of the die
8          d : [0..6] init 0;
9
10         [] s=0 -> 0.5 : (s'=1) + 0.5 : (s'=2);
11         [] s=1 -> 0.5 : (s'=3) + 0.5 : (s'=4);
12         [] s=2 -> 0.5 : (s'=5) + 0.5 : (s'=6);
13         [] s=3 -> 0.5 : (s'=1) + 0.5 : (s'=7) & (d'=1);
14         [] s=4 -> 0.5 : (s'=7) & (d'=2) + 0.5 : (s'=7) & (d'=3);
15         [] s=5 -> 0.5 : (s'=7) & (d'=4) + 0.5 : (s'=7) & (d'=5);
16         [] s=6 -> 0.5 : (s'=2) + 0.5 : (s'=7) & (d'=6);
17         [] s=7 -> (s'=7);
18
19 endmodule
20
```

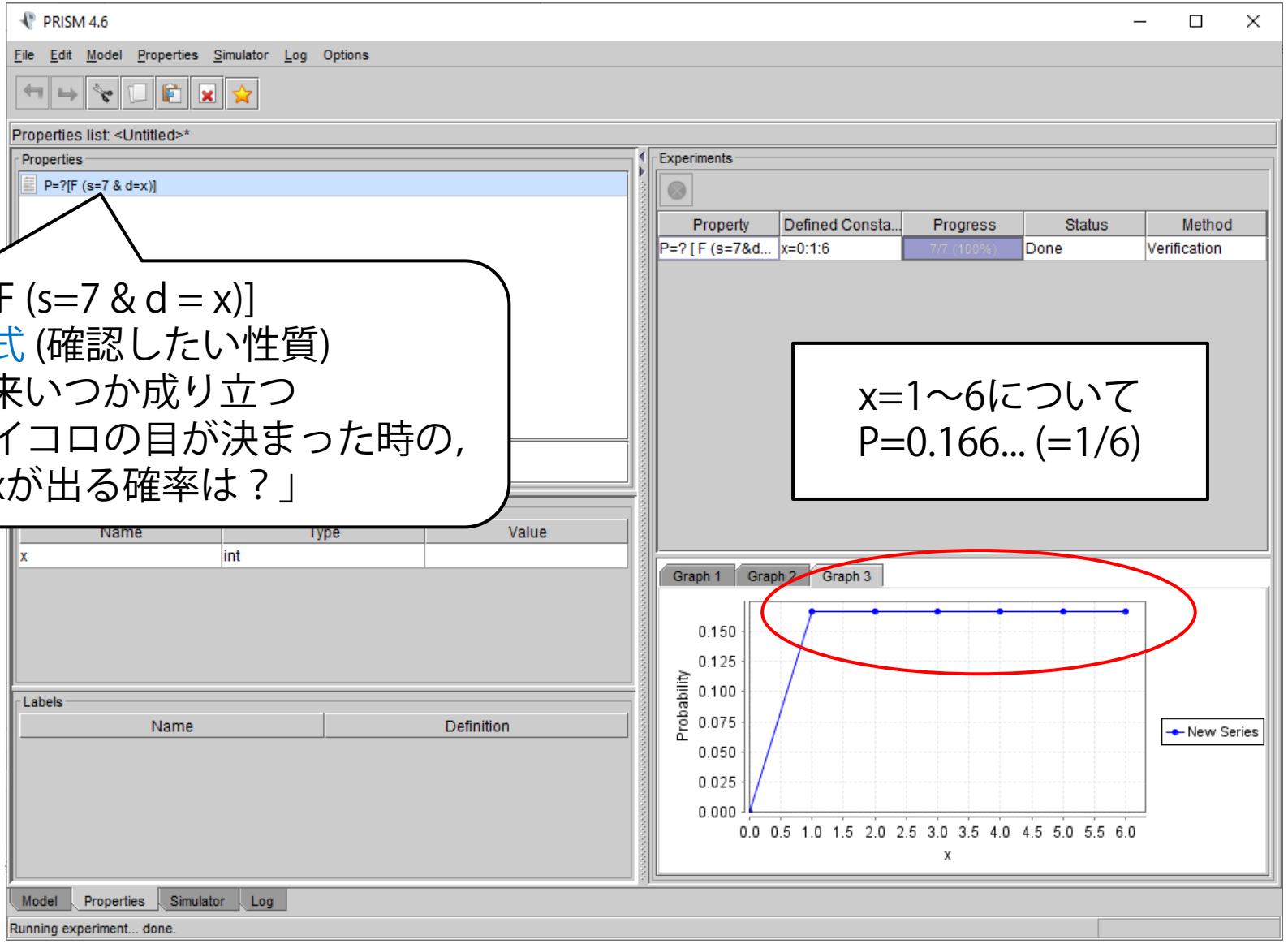
Built Model

States: 13
Initial states: 1
Transitions: 20

Model Properties Simulator Log

Running experiment... done.

検証結果



P=? [F (s=7 & d = x)]
検証式 (確認したい性質)
F: 将来いつか成り立つ
「サイコロの目が決まった時の，
目xが出る確率は？」

x=1～6について
P=0.166... (=1/6)

Expression caching for runtime verification based on parameterized probabilistic models

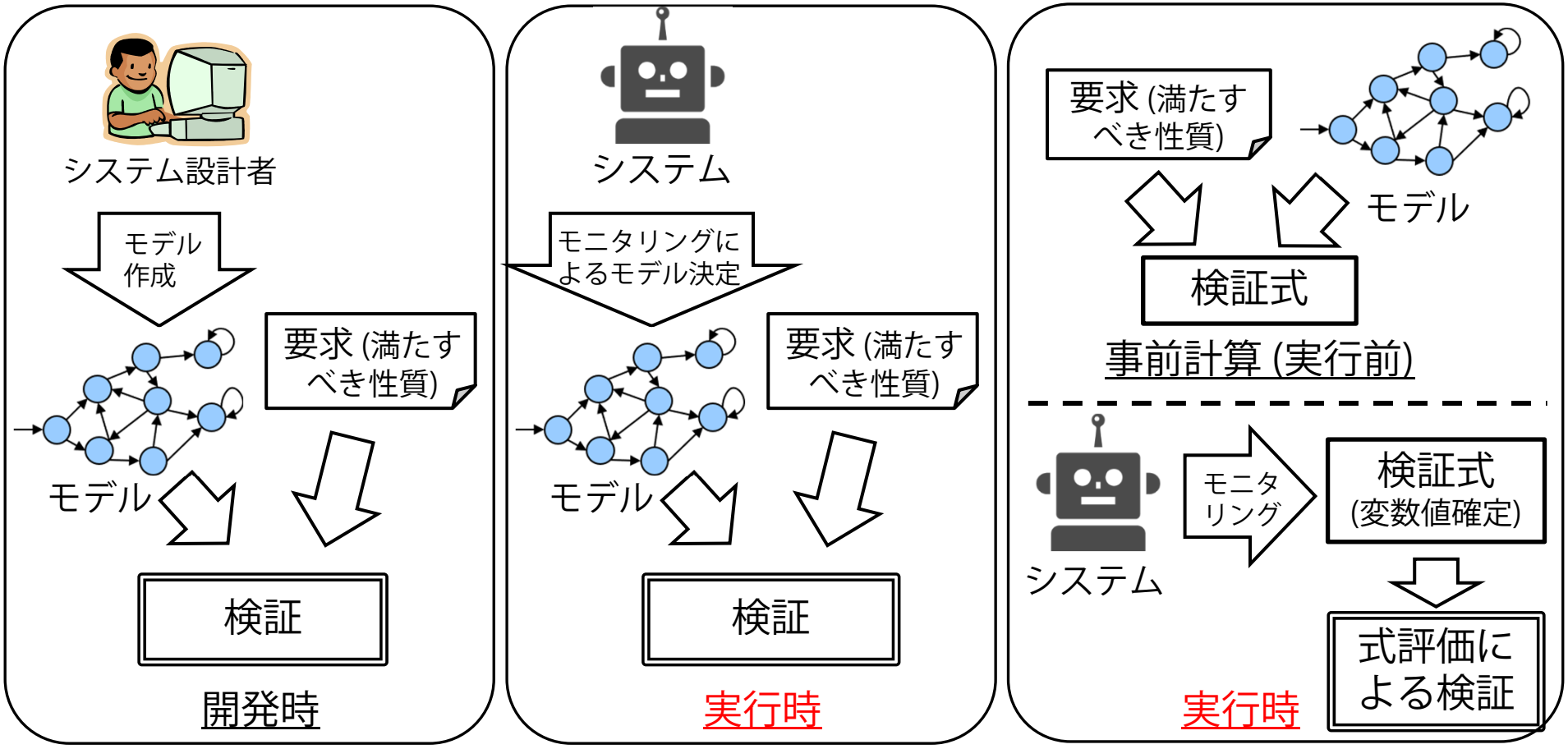
効率的な動的検証のためのパラメータ化確率モデルに基づいた検証式キャッシュ

Hiroyuki Nakagawa[1], Hiromu Toyama[2], Tatsuhiro Tsuchiya[1]

[1]Osaka University, [2]The University of Tokyo

The Journal of Systems and Software 156 (2019) 300–311, Elsevier

検証技術



[Filieri16] A.Filieri, G.Tamburrelli, C. Ghezzi, Supporting self-adaptation via quantitative verification and sensitivity analysis at run time, IEEE Transactions on Software Engineering, 2016.

確率モデルを用いた振舞いの検査

開発時

システムモデル

変数：値は実行時に判明

$$\begin{bmatrix} 0 & 1-x_0 & x_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & x_1 & 1-x_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-x_2 & 0 & 0 & x_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0.8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.9(1-x_3) & 0 & 0.1(1-x_3) & x_3 \\ 0 & 0 & 0 & 0 & 1-x_4 & 0 & 0 & x_4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

行列化

要求

・成功(S_8に到達する)確率は95%以上

検証式の構築
(計算時間大)

$f(x)$

実行時

モニタリング

検証

x=0.1, x2=0.5,…

$f(0.1,0.5,\ldots)$

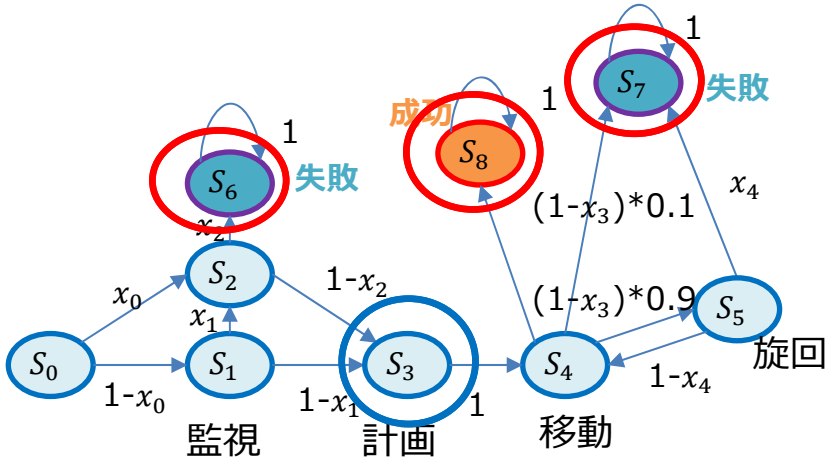
要求

or

検証式中の変数への
値代入による検証

[Filieri16] A. Filieri, G. Tamburrelli, and C. Ghezzi, “Supporting self-adaptation via quantitative verification and sensitivity analysis at run time,” IEEE Transactions on Software Engineering (TSE), vol. 42, no. 1, pp. 75–99, Jan 2016.

DTMC (Discrete Time Markov Chains)



- 吸収状態 s_i:
  - 一度到達すると他状態へ遷移できない状態
  - 遷移確率 p_{ii} =1
- 遷移状態:
  - 吸収状態ではない状態

$$P = \left(\begin{array}{cccccc|ccc} 0 & 1-x_0 & x_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & x_1 & 1-x_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-x_2 & 0 & 0 & x_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.9(1-x_3) & 0 & 0.1(1-x_3) & x_3 \\ 0 & 0 & 0 & 0 & 1-x_4 & 0 & 0 & x_4 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right)$$

Q 遷移状態 → 遷移状態

R 遷移状態 → 吸収状態

O 吸収状態 → 遷移状態 (zero matrix)

I 吸収状態 → 吸収状態 (identity matrix)

f(x)の生成法

- B: 遷移状態を複数回 (0回以上) 遷移した後に吸収状態へ遷移する確率を表す行列
- $B = N \times R$
  - $N = I + Q + Q^2 + \ldots = \sum_{i=0}^{\infty} Q^i = (I - Q)^{-1}$
    - Q: 遷移状態 → 遷移状態, R: 遷移状態 → 吸収状態

$f(x)$

検証式

$$= b_{ik} = \frac{1}{\det(I - Q)} \sum_{x \in 0 \cdots t-1} \alpha_{xi} (I - Q) \cdot r_{xj}$$

- b_{ik}: s_i から s_k への遷移確率
- $\det(M)$: Mの行列式 (determinant)
- α_{xi}: 余因子(cofactor)

Laplace展開/LU分解を用いて算出

確率モデルを用いた振舞いの検査

開発時

システムモデル

$$\begin{bmatrix} 0 & 1-x_0 & x_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & x_1 & 1-x_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-x_2 & 0 & 0 & x_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0.8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.9(1-x_3) & 0 & 0.1(1-x_3) & x_3 \\ 0 & 0 & 0 & 0 & 1-x_4 & 0 & 0 & x_4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

行列化

検証式の構築
(計算時間大)

$f(x)$

・成功(S_8に到達する)確率は95%以上

変数: 値は実行時に判明

実行時

モニタリング

検証

x=0.1, x2=0.5,…

$f(0.1,0.5,\ldots)$

要求

or

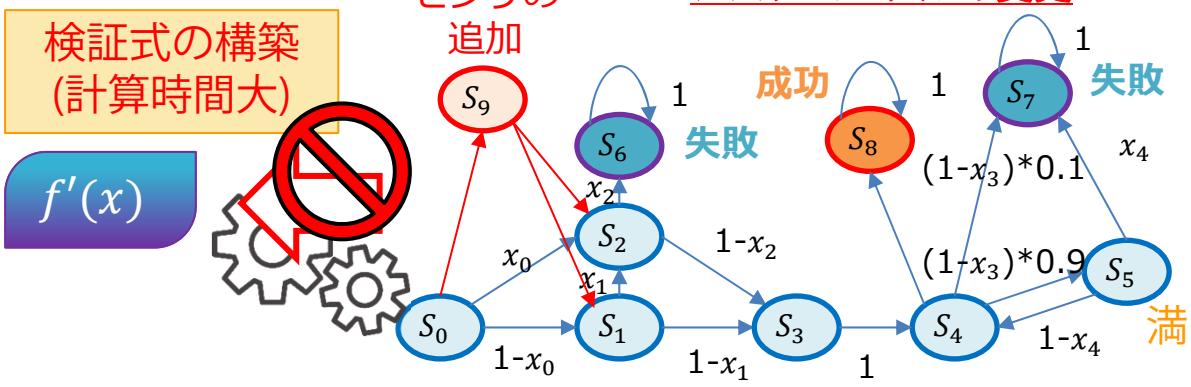
検証式中の変数への
値代入による検証

振舞い変更に伴う
システムモデルの変更

センサの
追加

検証式の構築
(計算時間大)

$f'(x)$



要求が
満たされない

23 [Filieri16] A. Filieri, G. Tamburrelli, and C. Ghezzi, "Supporting self-adaptation via quantitative verification and sensitivity analysis at run time," IEEE Transactions on Software Engineering (TSE), vol. 42, no. 1, pp. 75–99, Jan 2016.

解決策：キャッシュの利用 [Nakagawa16]

- 設計時

$f(x)$ の計算過程

(サブ)行列と検証式のペアを
キャッシュに保存

$$f_1'(x) = \begin{vmatrix} 0 & 0.2 & 0.1 & 0.5 \\ 0.01 & x_1 & 0 & 0.9 \\ 0 & x_2 & 0.5 & 0 \\ 0.2 & 0.7 & 0.01 & 0 \end{vmatrix}$$

Laplace展開

$$= 0.01 \begin{vmatrix} 0.2 & 0.1 & 0.5 \\ x_2 & 0.5 & 0 \\ 0.7 & 0.01 & 0.02 \end{vmatrix} + x_1 \begin{vmatrix} 0 & 0.1 & 0.5 \\ 0 & 0.5 & 0 \\ 0.2 & 0.01 & 0.02 \end{vmatrix} + 0.9 \begin{vmatrix} 0 & 0.2 & 0.1 \\ 0 & x_2 & 0.5 \\ 0.2 & 0.7 & 0.01 \end{vmatrix}$$

$f_2'(x)$ $f_3'(x)$ $f_4'(x)$

- 実行時 (振舞いの変化)

$f'(x)$ の計算過程

状態が
追加された

$$\begin{vmatrix} 0 & 0.2 & 0.1 & 0.5 & x_4 \\ 0.01 & x_1 & 0 & 0.9 & 0 \\ 0 & x_2 & 0.5 & 0 & 0 \\ 0.2 & 0.7 & 0.01 & 0.02 & 0 \\ 0 & 1-x_3 & 0 & 0 & x_3 \end{vmatrix} = (1-x_3) \begin{vmatrix} 0 & 0.1 & 0.5 & x_4 \\ 0.01 & 0 & 0.9 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0.2 & 0.01 & 0.02 & 0 \end{vmatrix} + x_3 \begin{vmatrix} 0 & 0.2 & 0.1 & 0.5 \\ 0.01 & x_1 & 0 & 0.9 \\ 0 & x_2 & 0.5 & 0 \\ 0.2 & 0.7 & 0.01 & 0.02 \end{vmatrix}$$

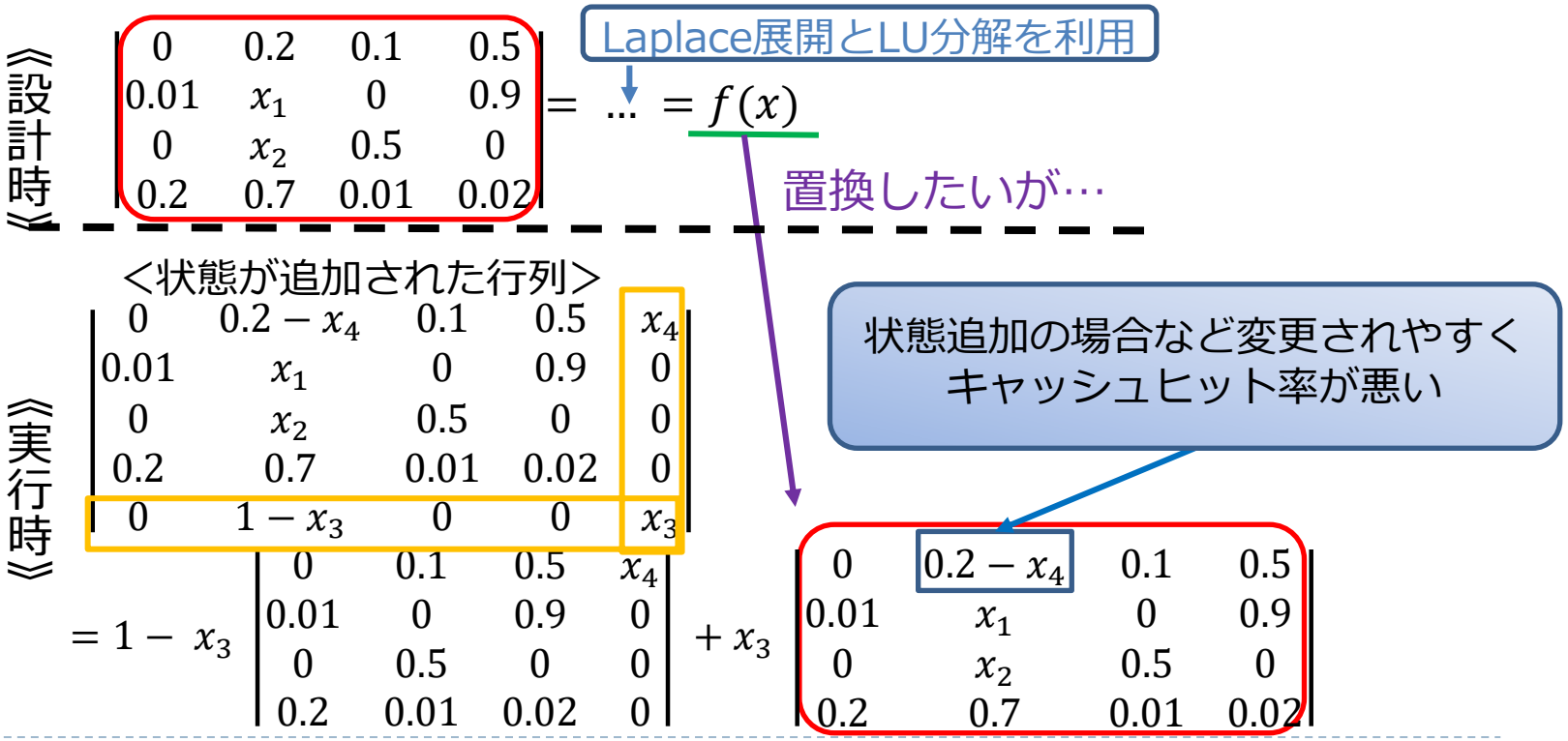
$f_1'(x)$

キャッシュから行列を
探し，検証式に置き換える

[Nakagawa16] H. Nakagawa, K. Ogawa, T. Tsuchiya, “Caching strategies for run-time probabilistic model checking," in Models@run.time 2016 (MRT2016).

キャッシュ利用時の問題点

- 遷移確率が少しでも変化した場合に対応できない
- DTMCモデルが完全に一致する可能性が低く，キャッシュヒットしにくい



[アイデア] 遷移確率のパラメータ化

遷移確率が変化した場合

先行研究のキャッシュ内モデル

$$\begin{pmatrix} 0.3-x_0 & 0.7+x_0 & 0 & 0 & 0 & 0 & 0 \\ 0.2 & 0.2 & 0.1-x_2 & 0 & x_2 & 0.3 & 0.2 \\ 0 & 0 & 0.4-x_1 & 0.6+x_1 & 0 & 0 & 0 \\ 0.3 & 0 & 0.2 & 0.3 & 0 & 0.1 & 0.1 \\ 0.4 & 0.3 & 0 & 0 & 0.1 & 0.1 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

振る舞い変更後モデル

$$\begin{pmatrix} 0.2-x_0 & 0.8+x_0 & 0 & 0 & 0 & 0 & 0 \\ 0.1 & 0.3 & 0.1-x_2 & 0 & x_2 & 0.3 & 0.2 \\ 0 & 0 & 0.4-x_1 & 0.6+x_1 & 0 & 0 & 0 \\ 0.3 & 0 & 0.2 & 0.3 & 0 & 0.1 & 0.1 \\ 0.6-x_3 & 0.4+x_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

再利用できない

更に変数化したモデル

$$\begin{pmatrix} \alpha_2 & \alpha_3 & 0 & 0 & \alpha_4 & 0 & 0 \\ \alpha_5 & \alpha_6 & \alpha_7 & 0 & \alpha_8 & \alpha_9 & \alpha_{10} \\ 0 & 0 & 0.4-x_1 & 0.6+x_1 & 0 & 0 & 0 \\ 0.3 & 0 & 0.2 & 0.3 & 0 & 0.1 & 0.1 \\ \alpha_{11} & \alpha_{12} & 0 & 0 & \alpha_{13} & \alpha_{14} & \alpha_{15} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

再利用可能

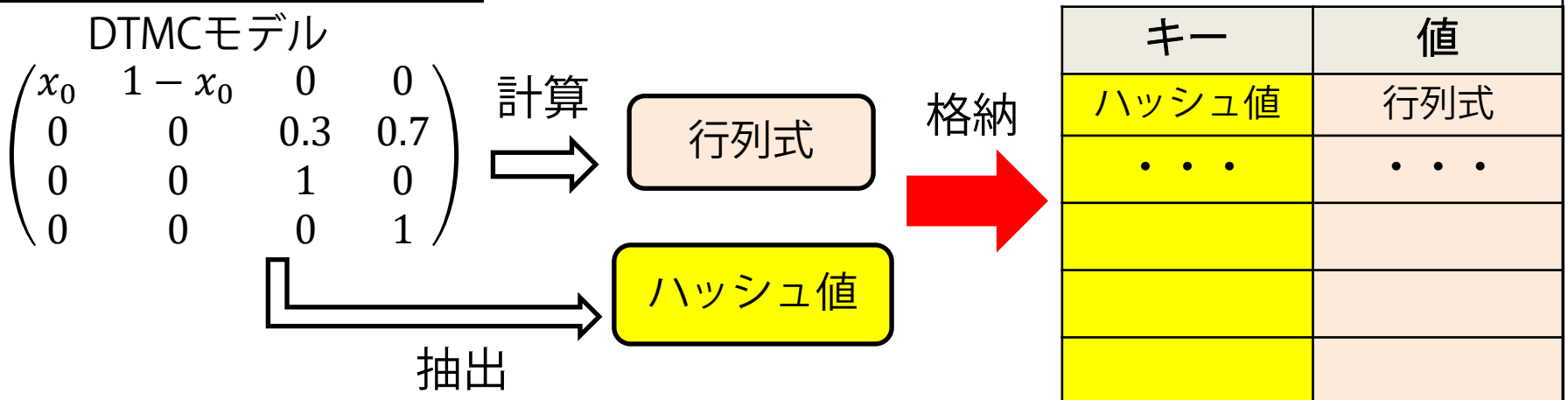
$\alpha_2 \leftarrow 0.2 - x_0$, $\alpha_3 \leftarrow 0.7 + x_0$

$\alpha_5 \leftarrow 0.2$, $\alpha_6 \leftarrow 0.2$

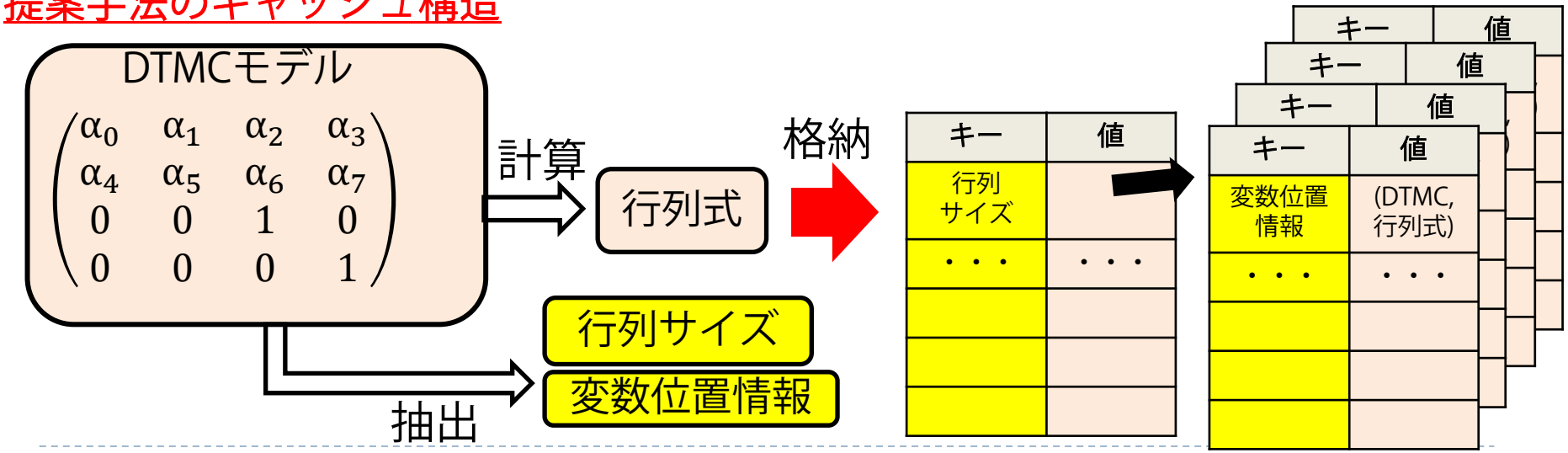
…

[細かい話] キャッシュの再構築

先行研究でのキャッシュ構造



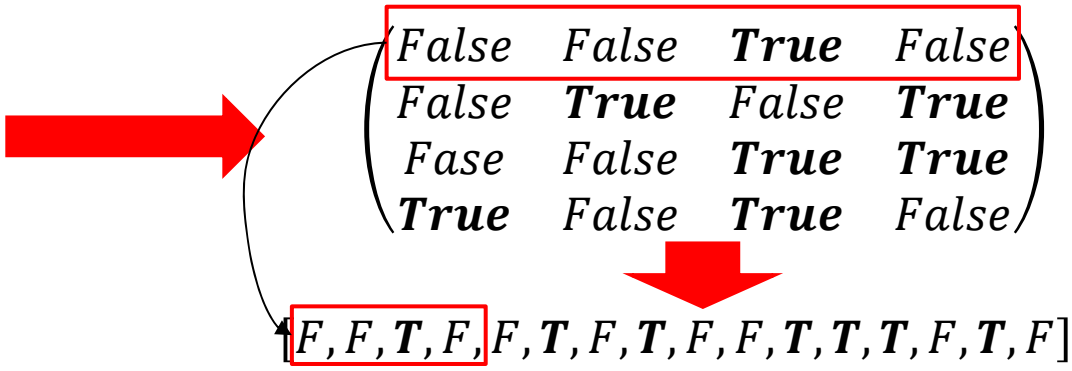
提案手法のキャッシュ構造



[細かい話] 変数位置によるフィルタリング

行列

$$\begin{pmatrix} 1 & 1 & \alpha_0 & 0 \\ 1 & \alpha_1 & 1 & \alpha_2 \\ 0 & 0 & \alpha_3 & \alpha_4 \\ \alpha_5 & 0 & \alpha_6 & 0 \end{pmatrix}$$



変数位置情報配列

入力a	入力b	出力c
$True$	$True$	$True$
$True$	$False$	$False$
$False$	$True$	$True$
$False$	$False$	$True$

入力時：
$True$:変数を含む
$False$:変数を含まない

出力時：
$True$:キャッシュヒットする
$False$:キャッシュヒットしない

a :検索している行列の要素
b :キャッシュに格納されている行列の要素
c :出力

$$c = \bar{a} + b$$

[細かい話] 変数位置によるフィルタリング

検索している行列

$$\begin{pmatrix} 1 & 1 & x_0 \\ 1 & x_1 & 1 \\ 0 & 0 & x_2 \end{pmatrix}$$

$a[F,F,T,F,T,F,F,F,T]$

キャッシュに保存されている行列

$$\begin{pmatrix} \alpha_i & 1 & \alpha_0 \\ \alpha_j & \alpha_1 & 1 \\ 0 & 0 & \alpha_2 \end{pmatrix}$$

$b[T,F,T,T,T,F,F,F,T]$

$c = \bar{a} + b$

$c[T,T,T,T,T,T,T,T,T]$

全て$True$なのでキャッシュヒット

検索している行列

$$\begin{pmatrix} 1 & 1 & x_0 \\ 1 & x_1 & 1 \\ 0 & 0 & \boldsymbol{x_2} \end{pmatrix}$$

$a[F,F,T,F,T,F,F,F,\boldsymbol{T}]$

キャッシュに保存されている行列

$$\begin{pmatrix} 1 & 1 & \alpha_0 \\ 1 & \alpha_1 & 1 \\ 0 & 0 & \boldsymbol{1} \end{pmatrix}$$

実行時パラメータは定数にならない

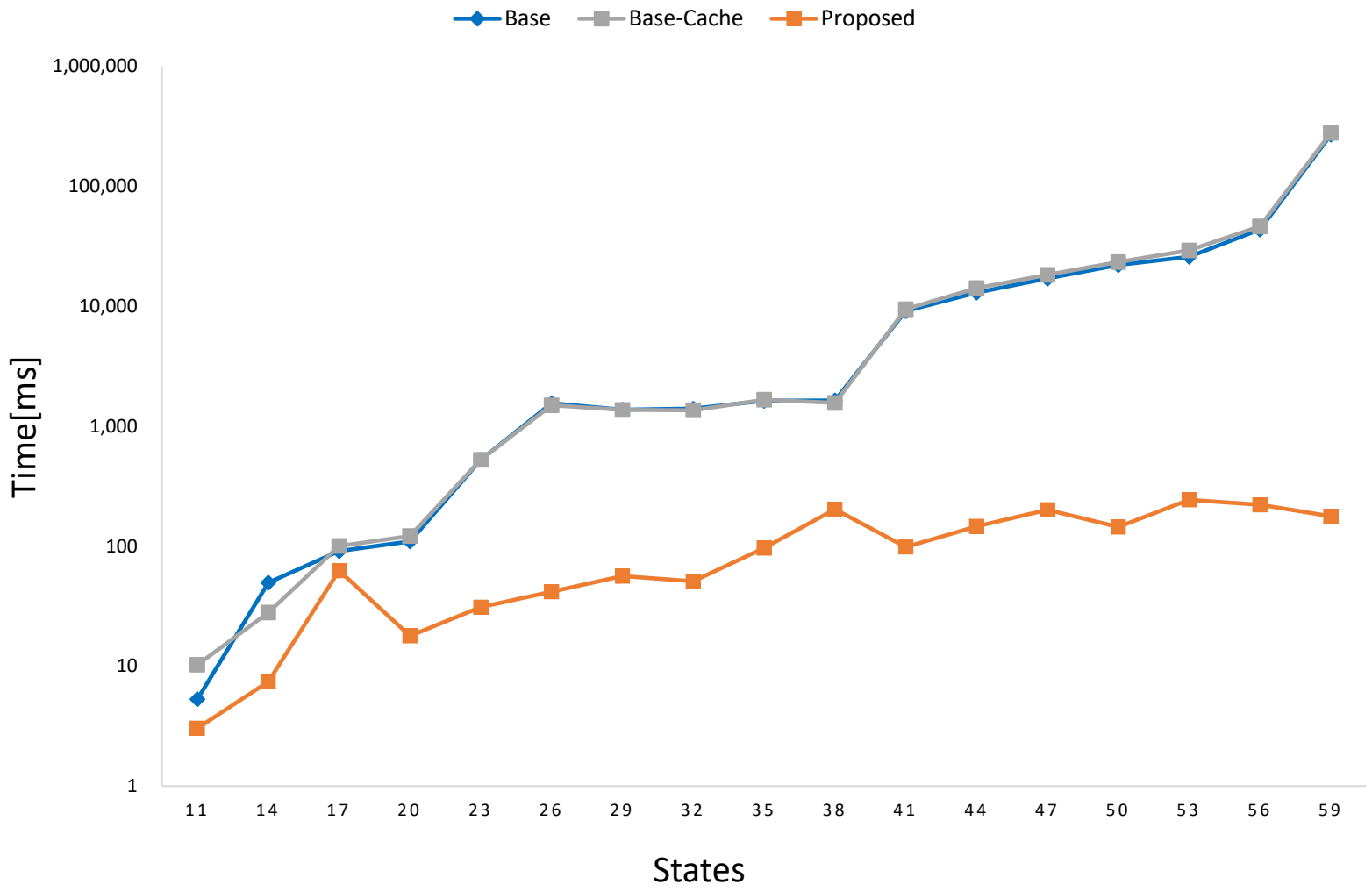
$c = \bar{a} + b$

$b[F,F,T,F,T,F,F,F,\boldsymbol{F}]$

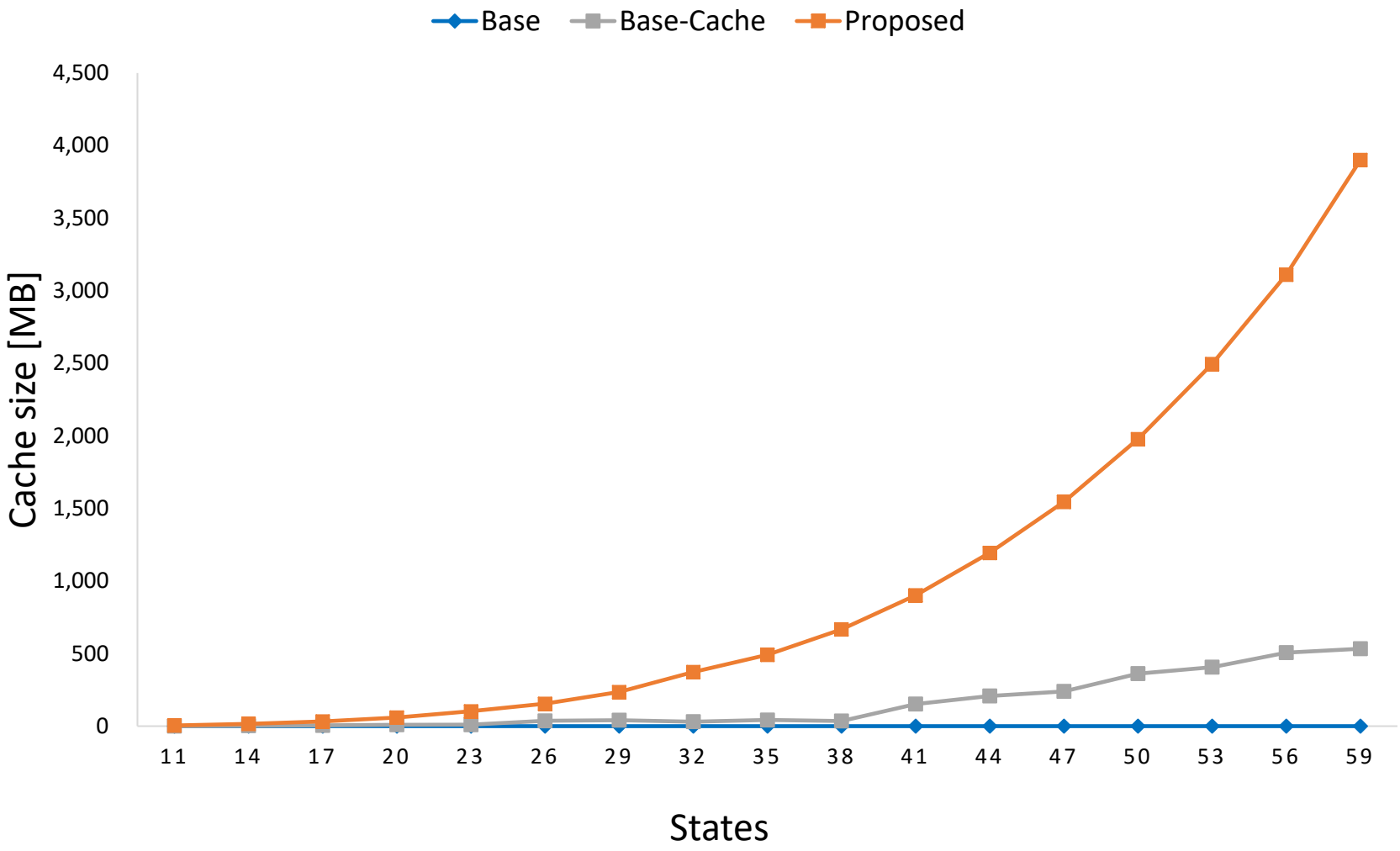
$c[T,T,T,T,T,T,T,T,\boldsymbol{F}]$

$False$があるのでキャッシュヒットしない

実験結果：検証式構築時間

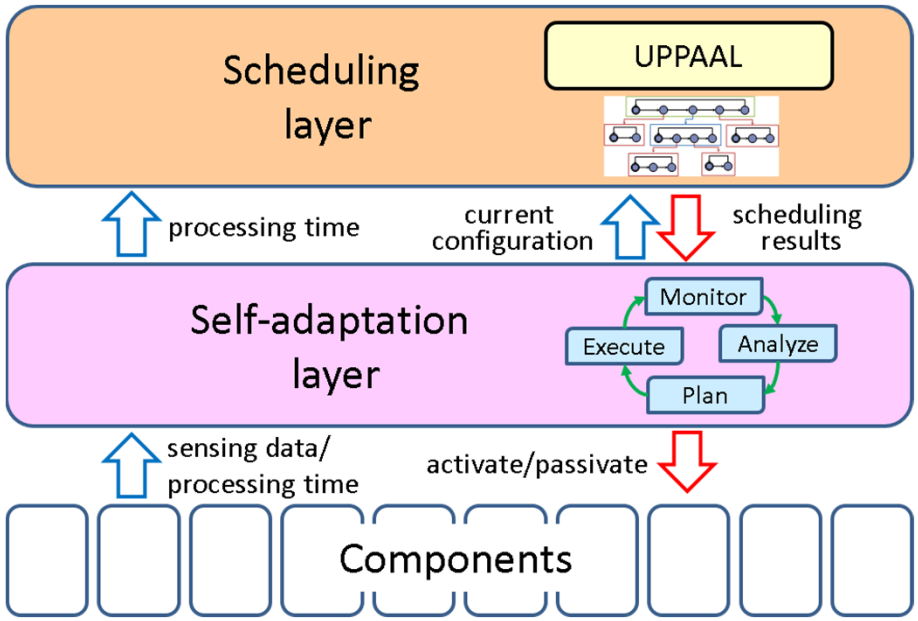


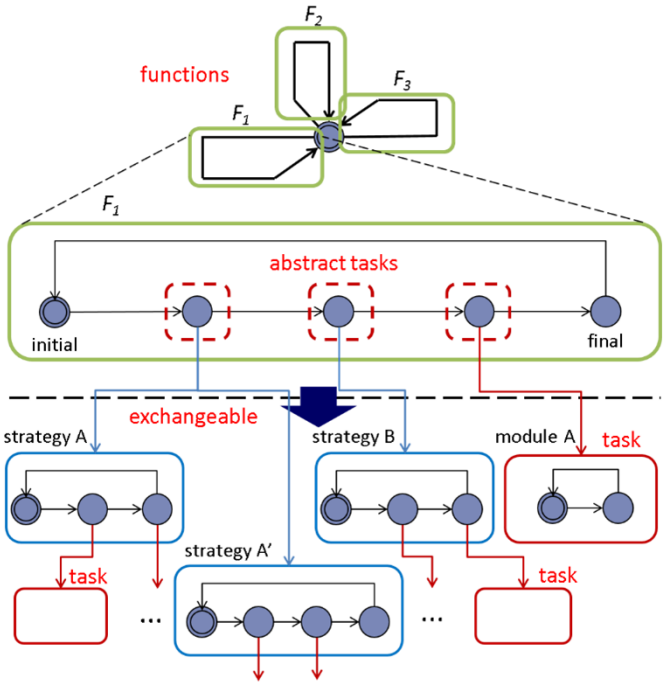
キャッシュサイズ

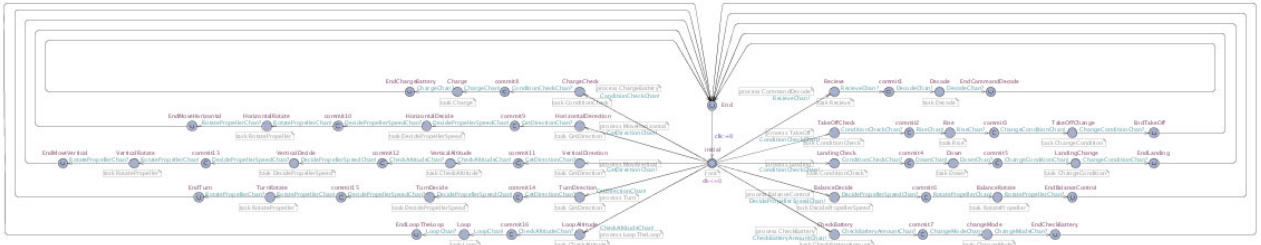

Base
Base-Cache
Proposed
Cache size [MB]
4,500
4,000
3,500
3,000
2,500
2,000
1,500
1,000
500
0
11
14
17
20
23
26
29
32
35
38
41
44
47
50
53
56
59
States

時間制約を遵守する自己適応システム [Nakagawa19]

- UPPAAL(モデル検査ツール)を動的に利用した時間管理







32 [Nakagawa19] H. Nakagawa, H. Tsuda, T. Tsuchiya, Towards Real-time Self-adaptation Using a Verification Mechanism , The 31st Euromicro Conference on Real-Time Systems (ECRTS 2019), 2019.

まとめ

[第2部] 自己適応システムとモデル検査の応用

- 自己適応システム
  - $W \wedge S \vdash R \xrightarrow{W \to W'} W' \wedge S \not\vdash R \xrightarrow{S \to S'} W' \wedge S' \vdash R$
    を自らが実現可能なシステム
  - MAPE-K loopメカニズム

- モデル検査技術
  - 自己適応システム：動的検証，不確かな環境
    → 確率的モデル検査
    - DTMCを用いたシステムモデル記述
    - 確率的モデル検査ツール: PRISM