Parameter Synthesis for Probabilistic Hyperproperties

Erika Ábrahám, RWTH Aachen, Germany
Ezio Bartocci, TU Wien, Austria
Borzoo Bonakdarpour and Oyendrila Dobe, Michigan State University

LPAR23 2020
13 January 2021

Trace properties and hyperproperties for reasoning about systems

[Clarkson, Schneider, 2010], [Clarkson, Finkbeiner, Koleini, Micinski, Rabe, Sánchez, 2014]

- A trace $t = s_0, s_1, \ldots$ is an infinite sequence of states $s_i \in S$.
- A trace property is a set of traces.

Trace properties and hyperproperties for reasoning about systems

[Clarkson, Schneider, 2010], [Clarkson, Finkbeiner, Koleini, Micinski, Rabe, Sánchez, 2014]

- A trace $t = s_0, s_1, \ldots$ is an infinite sequence of states $s_i \in S$.
- A trace property is a set of traces.

Example (LTL property)

I never start working before having a coffee.

$$(\mathcal{G}\ \neg work) \vee ((\neg work)\ \mathcal{U}\ coffee)$$

Trace properties and hyperproperties for reasoning about systems

[Clarkson, Schneider, 2010], [Clarkson, Finkbeiner, Koleini, Micinski, Rabe, Sánchez, 2014]

- A trace $t = s_0, s_1, \ldots$ is an infinite sequence of states $s_i \in S$.
- A trace property is a set of traces.

Example (LTL property)

I never start working before having a coffee.

$$(\mathcal{G}\ \neg work) \vee ((\neg work)\ \mathcal{U}\ coffee)$$

Classical trace properties cannot express relations between traces.

Trace properties and hyperproperties for reasoning about systems

[Clarkson, Schneider, 2010], [Clarkson, Finkbeiner, Koleini, Micinski, Rabe, Sánchez, 2014]

- A trace $t = s_0, s_1, \ldots$ is an infinite sequence of states $s_i \in S$.
- A trace property is a set of traces.

Example (LTL property)

I never start working before having a coffee.

$$(\mathcal{G}\ \neg work) \vee ((\neg work)\ \mathcal{U}\ coffee)$$

Classical trace properties cannot express relations between traces.

- A hyperproperty is a set of sets of traces.

Trace properties and hyperproperties for reasoning about systems

[Clarkson, Schneider, 2010], [Clarkson, Finkbeiner, Koleini, Micinski, Rabe, Sánchez, 2014]

- A trace $t = s_0, s_1, \ldots$ is an infinite sequence of states $s_i \in S$.
- A trace property is a set of traces.

Example (LTL property)

I never start working before having a coffee.

$$(\mathcal{G}\ \neg work) \vee ((\neg work)\ \mathcal{U}\ coffee)$$

Classical trace properties cannot express relations between traces.

- A hyperproperty is a set of sets of traces.

Example (HyperLTL property)

I drink coffee every day at the same time.

$$\forall \pi.\ \forall \pi'.\quad (\mathcal{G}\ (coffee_{\pi} \Leftrightarrow coffee_{\pi'}))$$

Probabilistic hyperproperties

Consider a parallel program (h: high input / l: low output).

$$\boxed{\text{while } h > 0 \text{ do } \{h \leftarrow h - 1\};\ l \leftarrow 2 \qquad || \qquad l \leftarrow 1}$$

Probabilistic hyperproperties

Consider a parallel program (h: high input / l: low output).

$$\boxed{\text{while } h > 0 \text{ do } \{h \leftarrow h - 1\};\ l \leftarrow 2 \quad || \quad l \leftarrow 1}$$

Assuming a uniform probabilistic scheduler:

Probabilistic hyperproperties

Consider a parallel program (h: high input / l: low output).

$$\boxed{\text{while } h > 0 \text{ do } \{h \leftarrow h - 1\};\ l \leftarrow 2 \quad || \quad l \leftarrow 1}$$

Assuming a uniform probabilistic scheduler:

- $h = 0 \rightarrow \mathbb{P}(l{=}1) = 1/4$ at termination.
- $h = 5 \rightarrow \mathbb{P}(l{=}1) = 1/4096$ at termination.

Probabilistic hyperproperties

Consider a parallel program (h: high input / l: low output).

$$\boxed{\text{while } h > 0 \text{ do } \{h \leftarrow h - 1\};\ l \leftarrow 2 \quad || \quad l \leftarrow 1}$$

Assuming a uniform probabilistic scheduler:

- $h = 0 \rightarrow \mathbb{P}(l{=}1) = 1/4$ at termination.
- $h = 5 \rightarrow \mathbb{P}(l{=}1) = 1/4096$ at termination.

Probabilistic noninterference stipulates that the probability distribution on the final values on publicly observable channels (low outputs) is independent of the initial values of secrets (high inputs).

Probabilistic hyperproperties

Consider a parallel program (h: high input / l: low output).

$$\boxed{\text{while } h > 0 \text{ do } \{h \leftarrow h - 1\};\ l \leftarrow 2 \quad || \quad l \leftarrow 1}$$

Assuming a uniform probabilistic scheduler:

- $h = 0 \rightarrow \mathbb{P}(l{=}1) = 1/4$ at termination.
- $h = 5 \rightarrow \mathbb{P}(l{=}1) = 1/4096$ at termination.

Probabilistic noninterference stipulates that the probability distribution on the final values on publicly observable channels (low outputs) is independent of the initial values of secrets (high inputs).

We need probabilistic hyperproperties to express probabilistic relations between independent executions of a system.

HyperPCTL for DTMCs [Ábrahám, Bonakdarpour, 2018]

HyperPCTL: PCTL extended with quantification over initial states

Example (Probabilistic noninterference)

$$\forall s.\forall s'.\Big(init_s \wedge init_{s'} \wedge h_s \neq h_{s'}\Big) \Rightarrow \Big(\mathbb{P}\big(\mathcal{F}(fin_s \wedge (l{=}1)_s)\big) = \mathbb{P}\big(\mathcal{F}(fin_{s'} \wedge (l{=}1)_{s'})\big)\Big)$$

HyperPCTL for DTMCs [Ábrahám, Bonakdarpour, 2018]

HyperPCTL: PCTL extended with quantification over initial states

Example (Probabilistic noninterference)

$$\forall s.\forall s'.\Big(init_s \wedge init_{s'} \wedge h_s \neq h_{s'}\Big) \Rightarrow \Big(\mathbb{P}\big(\mathcal{F}(fin_s \wedge (l{=}1)_s)\big) = \mathbb{P}\big(\mathcal{F}(fin_{s'} \wedge (l{=}1)_{s'})\big)\Big)$$

HyperPCTL for DTMCs [Ábrahám, Bonakdarpour, 2018]

HyperPCTL: PCTL extended with quantification over initial states

Example (Probabilistic noninterference)

$$\forall s.\forall s'.\Big(init_s \wedge init_{s'} \wedge h_s \neq h_{s'}\Big) \Rightarrow \Big(\mathbb{P}\big(\mathcal{F}(fin_s \wedge (l{=}1)_s)\big) = \mathbb{P}\big(\mathcal{F}(fin_{s'} \wedge (l{=}1)_{s'})\big)\Big)$$

Our contribution:

To synthesize parametric configurations such that the specified probabilistic hyperproperty is satisfied in a given parametric deterministic probabilistic system !

ReachHyperPCTL syntax

ReachHyperPCTL syntax

HyperPCTL is similar to PCTL BUT

- they quantify ($Q_i \in \{\exists, \forall\}$) over schedulers and initial states:

$$Q_{\sigma_1} s_1 \dots Q_{\sigma_n} s_n. \quad \psi$$

- they index atomic propositions:

$$\psi ::= a_\sigma \;\Big|\; \psi \wedge \psi \;\Big|\; \neg\psi \;\Big|\; p \text{ op } p \qquad \textit{where } \mathrm{op} = \{>, \geq, =, <, \leq\}$$

- they support arithmetic computations with probability expressions:

$$p \quad ::= \quad \mathbb{P}(\mathcal{X}\psi) \;\Big|\; \mathbb{P}(\psi\,\mathcal{U}\,\psi) \;\Big|\; c \;\Big|\; p + p \;\Big|\; p - p \;\Big|\; p \cdot p$$

ReachHyperPCTL syntax

HyperPCTL is similar to PCTL BUT

- they quantify ($Q_i \in \{\exists, \forall\}$) over schedulers and initial states:

$$Q_{\sigma_1} s_1 \ldots Q_{\sigma_n} s_n. \quad \psi$$

- they index atomic propositions:

$$\psi ::= a_\sigma \;\Big|\; \psi \wedge \psi \;\Big|\; \neg\psi \;\Big|\; p \text{ op } p \qquad \textit{where } \mathrm{op} = \{>, \geq, =, <, \leq\}$$

- they support arithmetic computations with probability expressions:

$$p \quad ::= \quad \mathbb{P}(\mathcal{X}\psi) \;\Big|\; \mathbb{P}(\psi\,\mathcal{U}\,\psi) \;\Big|\; c \;\Big|\; p + p \;\Big|\; p - p \;\Big|\; p \cdot p$$

In this work, we use ReachHyperPCTL which is a restrictive fragment of HyperPCTL that does not allow nested probability operators.

Parametric DTMC (PDTMC)

Example

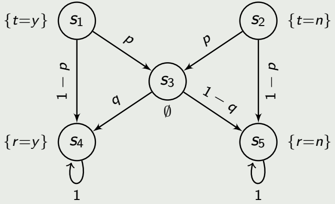


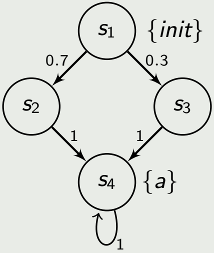
Figure: Parametric DTMC for randomized response protocol.

A configuration u for V, a finite set of parameters (here, p, q) is valid for the PDTMC, if,

$$\sum_{s' \in S} \mathbb{P}_u(s, s') = 1$$

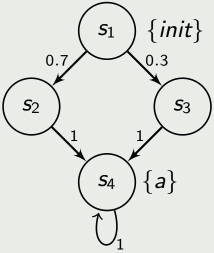
DTMCs and their self-composition

Example



DTMCs and their self-composition

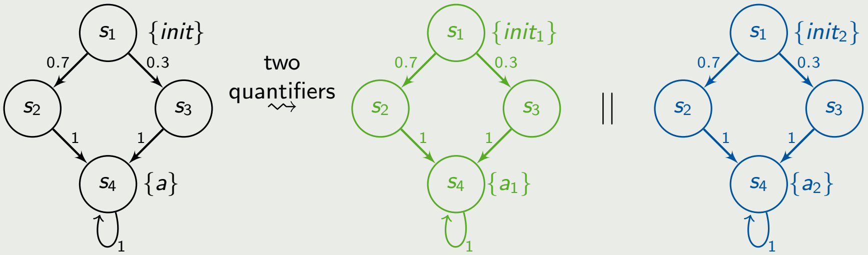
Example



two quantifiers $\rightsquigarrow$

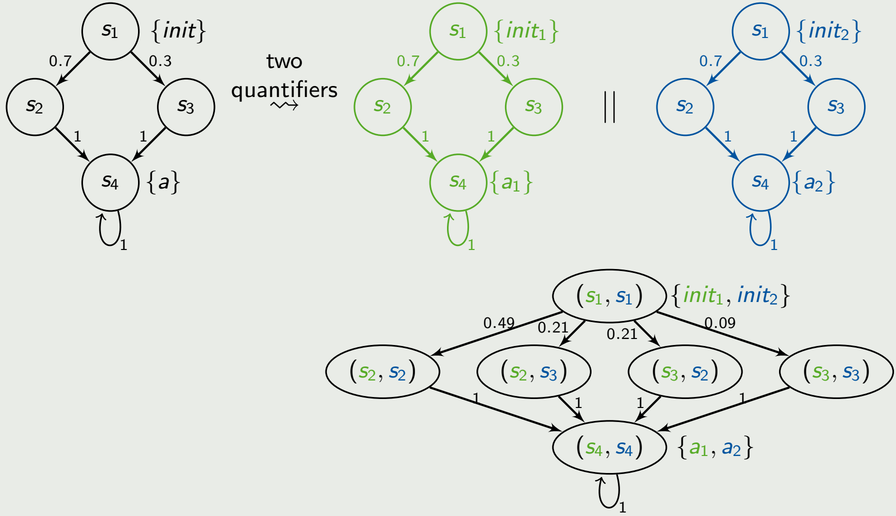
DTMCs and their self-composition

Example



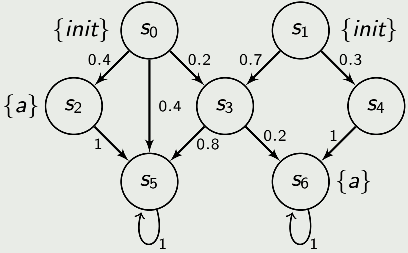
DTMCs and their self-composition

Example



HyperPCTL semantics

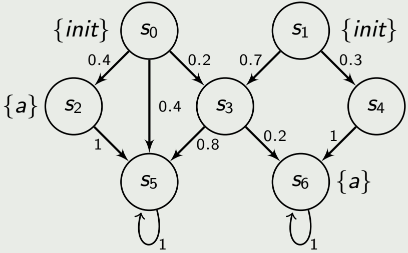
Example



$$\psi = \forall s.\forall s'.(init_s \wedge init_{s'}) \Rightarrow \Big(\mathbb{P}(\mathcal{F} a_s) = \mathbb{P}(\mathcal{F} a_{s'})\Big)$$

HyperPCTL semantics

Example

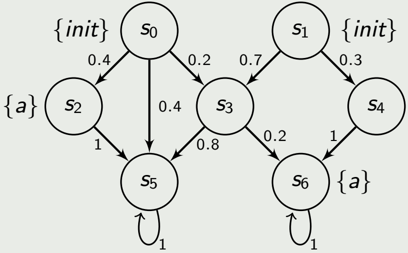


$$\psi = \forall s.\forall s'.(init_s \wedge init_{s'}) \Rightarrow \Big(\mathbb{P}(\mathcal{F} a_s) = \mathbb{P}(\mathcal{F} a_{s'})\Big)$$

The probability of reaching a from s_0 is $0.4 + (0.2 \times 0.2) = 0.44$.

HyperPCTL semantics

Example



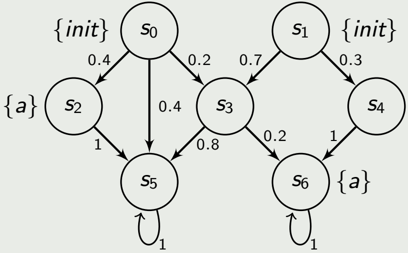
$$\psi = \forall s.\forall s'.(init_s \wedge init_{s'}) \Rightarrow \Big(\mathbb{P}(\mathcal{F} a_s) = \mathbb{P}(\mathcal{F} a_{s'})\Big)$$

The probability of reaching a from s_0 is $0.4 + (0.2 \times 0.2) = 0.44$.

The probability of reaching a from s_1 is $0.3 + (0.7 \times 0.2) = 0.44$.

HyperPCTL semantics

Example



$$\psi = \forall s.\forall s'.(init_s \wedge init_{s'}) \Rightarrow \Big(\mathbb{P}(\mathcal{F} a_s) = \mathbb{P}(\mathcal{F} a_{s'})\Big)$$

The probability of reaching a from s_0 is $0.4 + (0.2 \times 0.2) = 0.44$.

The probability of reaching a from s_1 is $0.3 + (0.7 \times 0.2) = 0.44$.

Hence, $\mathcal{M}, (s_0,\ s_1) \models \psi$

Problem statement

Given a PDTMC, D, and a ReachHyperPCTL formula, $\psi = \quad Q_{\sigma_1} s_1 \ldots Q_{\sigma_n} s_n . \psi'$, we aim to provide algorithms to,

- Synthesize valid parameter configurations for D such that ψ is satisfied.

Problem statement

Given a PDTMC, D, and a ReachHyperPCTL formula, $\psi = \quad Q_{\sigma_1} s_1 \ldots Q_{\sigma_n} s_n.\psi'$, we aim to provide algorithms to,

- Synthesize valid parameter configurations for D such that ψ is satisfied.
- For a box R of the valid parameter configurations of D, point to regions where,
  - all configurations
  - some configurations
  - none of the configurations

  satisfy ψ for D.

Problem statement

Given a PDTMC, D, and a ReachHyperPCTL formula, $\psi = \quad Q_{\sigma_1} s_1 \ldots Q_{\sigma_n} s_n . \psi'$, we aim to provide algorithms to,

- Synthesize valid parameter configurations for D such that ψ is satisfied.
- For a box R of the valid parameter configurations of D, point to regions where,
  - all configurations
  - some configurations
  - none of the configurations

  satisfy ψ for D.

Theorem

The problem of parameter synthesis and checking of parameter space are both decidable for deterministic systems.

Parameter Synthesis Algorithm for ReachHyperPCTL

We propose an symbolic encoding technique for solving the model checking problem, such that

$$\boxed{\begin{array}{c} \mathcal{D} \;=\; (S, V, \mathrm{P}, AP, L) \\ \text{satisfies } \psi \text{ in the region } R \end{array}}$$

iff

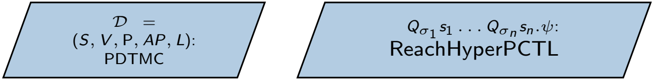
$$\boxed{\mathit{Symb}(\psi, s) \wedge R \text{ is satisfied}}$$

Given a parameter space, we recursively divide it to find regions where the equation is always satisfied, always violated, or satisfied for some configurations.

Overall Algorithm

$\mathcal{D} = (S, V, \mathrm{P}, AP, L)$: PDTMC

Overall Algorithm



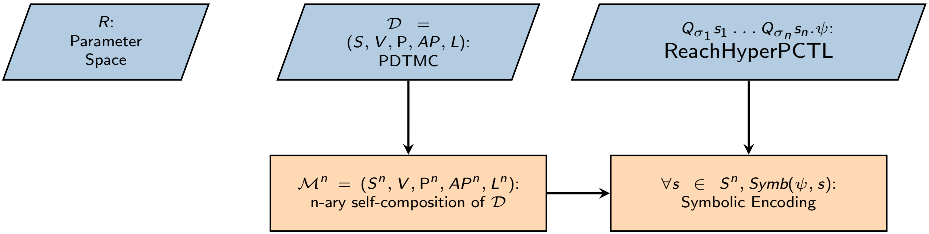
Overall Algorithm

R: Parameter Space

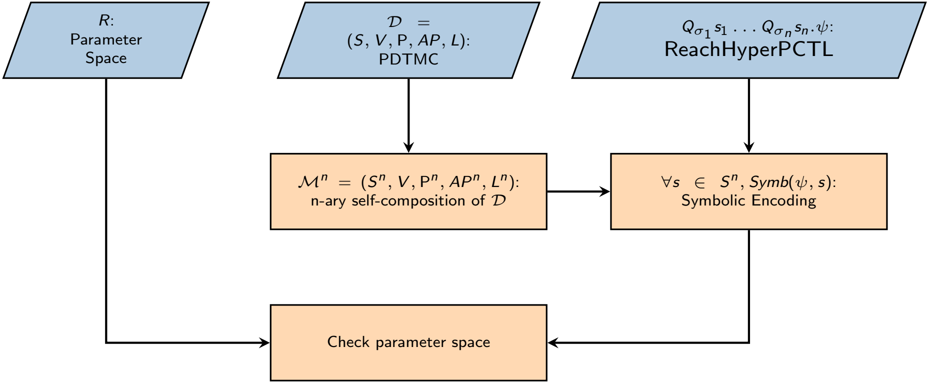
$\mathcal{D} = (S, V, \mathrm{P}, AP, L)$: PDTMC

$Q_{\sigma_1} s_1 \ldots Q_{\sigma_n} s_n . \psi$: ReachHyperPCTL

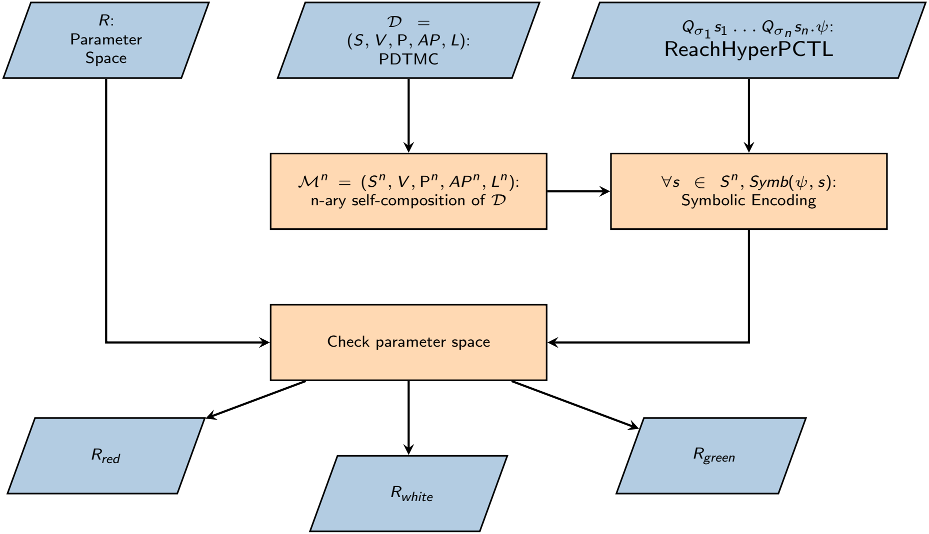
Overall Algorithm



Overall Algorithm



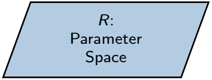
Overall Algorithm



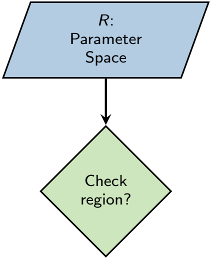
We utilize the given $\mathcal{D}$ and ReachHyperPCTL,

- to generate an encoding which is essentially a real-arithmetic formula $Symb(\psi, s)$ for each n-ary composed state, s.
- recursively parse the formula to calculate *Symb*.
- *Symb* represents equations representing reachability probability in case of temporal operators.
- Solving these equations along with the parameter space allowed would give us parametric configurations that we aim to find.

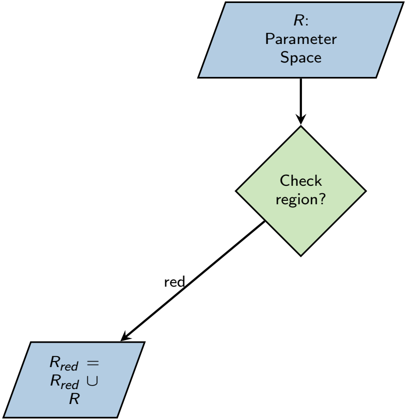
Algorithm for checking parameter space



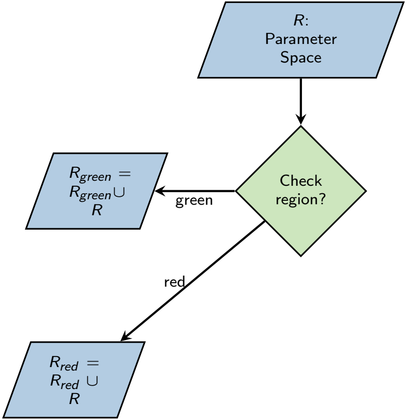
Algorithm for checking parameter space



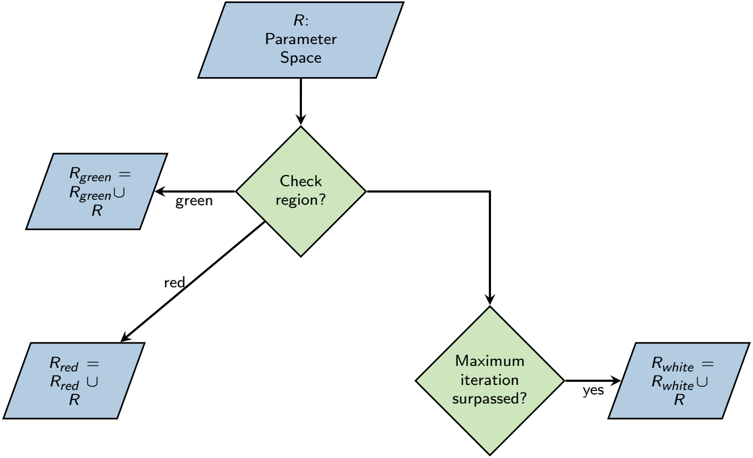
Algorithm for checking parameter space



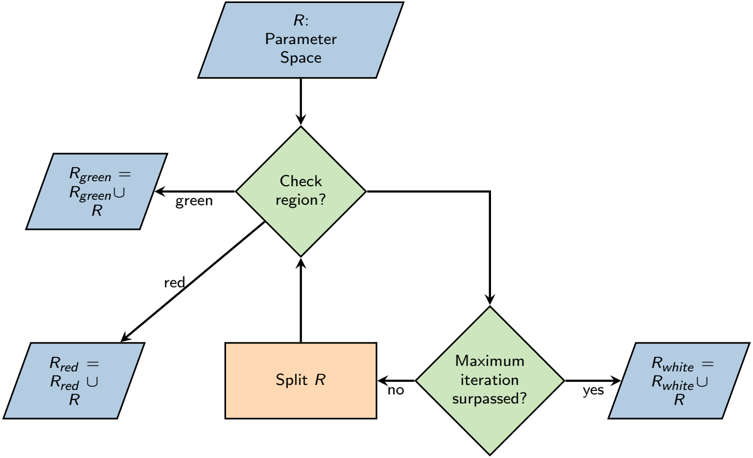
Algorithm for checking parameter space



Algorithm for checking parameter space

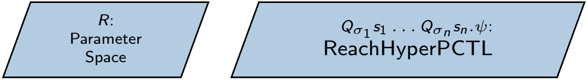


Algorithm for checking parameter space



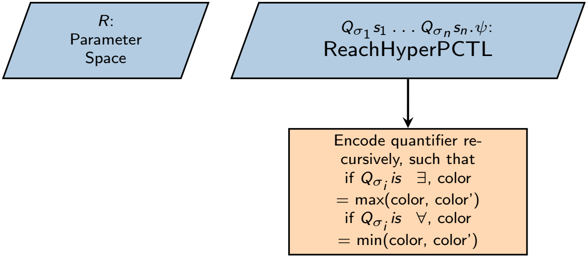
Algorithm for checking region

Here, we assume, green = 1, white = 0, red = -1.



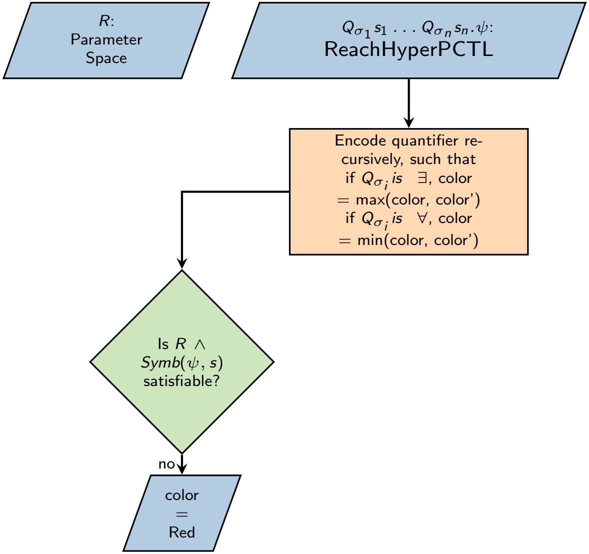
Algorithm for checking region

Here, we assume, green = 1, white = 0, red = -1.



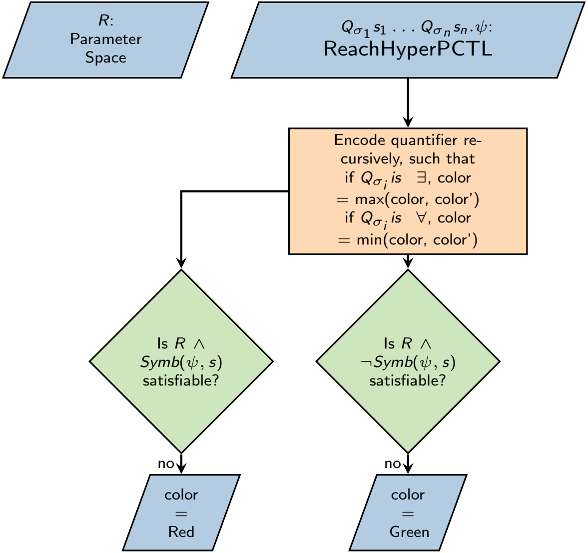
Algorithm for checking region

Here, we assume, green = 1, white = 0, red = -1.



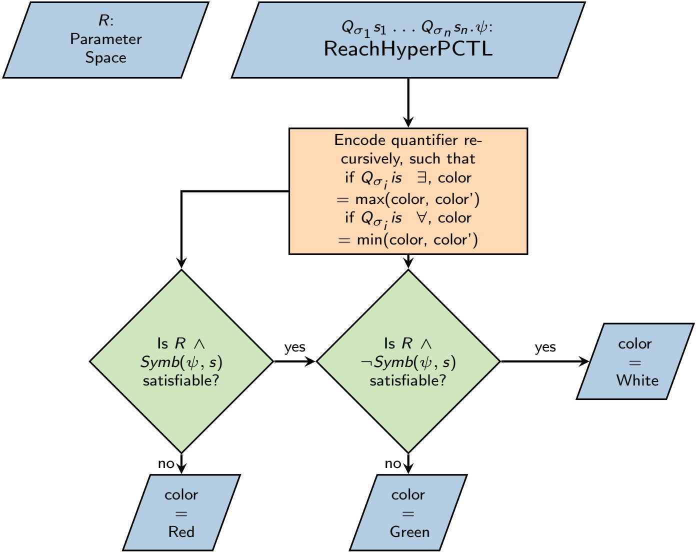
Algorithm for checking region

Here, we assume, green = 1, white = 0, red = -1.



Algorithm for checking region

Here, we assume, green = 1, white = 0, red = -1.



Randomized Response

- Guarantees differential privacy by creating noise and providing plausible deniability.
- In this example, p is the parameter whose value we try to calculate depending on the ϵ we want to achieve in the privacy protocol.
- This algorithm should satisfy the following property:

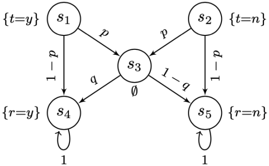


Figure: Differential privacy

$$\varphi_{\mathrm{dp}} = \forall\sigma.\forall\sigma'.\Big[\Big(\big(t{=}n\big)_\sigma \wedge \big(t{=}y\big)_{\sigma'}\Big) \Rightarrow \Big(\mathbb{P}\big(\mathcal{F}(r{=}n)_\sigma\big) \le e^{\epsilon}\cdot\mathbb{P}\big(\mathcal{F}(r{=}n)_{\sigma'}\big)\Big)\Big] \wedge$$
$$\Big[\Big(\big(t{=}y\big)_\sigma \wedge \big(t{=}n\big)_{\sigma'}\Big) \Rightarrow \Big(\mathbb{P}\big(\mathcal{F}(r{=}y)_\sigma\big) \le e^{\epsilon}\cdot\mathbb{P}\big(\mathcal{F}(r{=}y)_{\sigma'}\big)\Big)\Big]$$

Randomized Response - Results

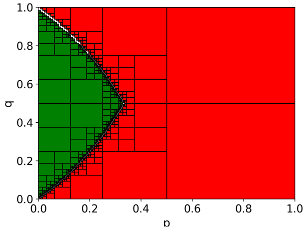


Figure: Parameter Space for ($\ln 2$)-differential privacy.

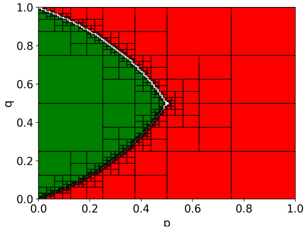


Figure: Parameter Space for ($\ln 3$)-differential privacy.

We terminated the algorithm after 1500 rounds and the synthesis took 0.1 seconds.

Probabilistic Conformance

- Check if implementation conforms with given specification.
- Synthesize a protocol such that the reachability probability is the same for the 5 final states when the die is rolled as well as when the die is simulated by fair coin tosses.

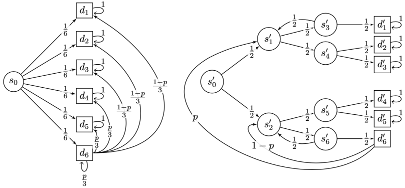


Figure: **Left**: 5-sided die
Right: parametric adaption of 5-sided die obtained by fair coin tosses

$$\varphi_{\mathsf{pc}} = \forall \sigma . \exists \sigma' . \left(s_{0_\sigma} \wedge s'_{0_{\sigma'}} \right) \Rightarrow \bigwedge_{i=1}^{5} \left(\mathbb{P}(\mathcal{F} d_{i_\sigma}) = \mathbb{P}(\mathcal{F} d'_{i_{\sigma'}}) \right)$$

We synthesized the parameter $p = 0.5$ in 28 seconds. It is the only valid solution.

Probabilistic Noninterference in Randomized Schedulers

Assume two threads:

$$th_1 : \text{ while } h > 0 \text{ do } \{h := h - 1\};\; l := 2$$

$$th_2 : \; l := 1$$

- Attacker should not be able to distinguish two computations from their low observable outputs, if their high inputs are different.
- It requires that the probability of every low-observable trace pattern is the same for every low-equivalent initial state.

$$\varphi_{\text{pni}} = \forall\sigma.\forall\sigma'.\Big(i_\sigma \wedge (h{=}0)_\sigma \wedge i_{\sigma'} \wedge (h{=}1)_{\sigma'}\Big) \Rightarrow \Big(\Big(\mathbb{P}(\mathcal{F}(f_\sigma \wedge (l{=}1)_\sigma)) = \mathbb{P}(\mathcal{F}(f_{\sigma'} \wedge (l{=}1)_{\sigma'}))\Big) \wedge \Big(\mathbb{P}(\mathcal{F}(f_\sigma \wedge (l{=}2)_\sigma)) = \mathbb{P}(\mathcal{F}(f_{\sigma'} \wedge (l{=}2)_{\sigma'}))\Big)\Big).$$

Probabilistic Noninterference in Randomized Schedulers - Results

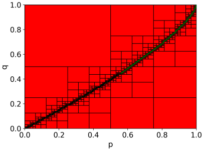


Figure: Parameter Space for $(h = 3)_{\sigma}, (h = 5)_{\sigma'}$.

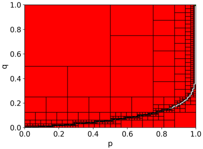


Figure: Parameter Space for $(h = 0)_{\sigma}, (h = 10)_{\sigma'}$.

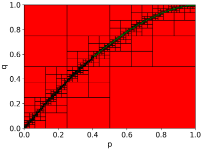


Figure: Parameter Space for $(h = 14)_{\sigma}, (h = 8)_{\sigma'}$.

Input h	**Running time (s)**			#white boxes	#red boxes	red area	#samples
	Alg: Symbolic encoding	**Alg: Check parameter space**	**Total**				
$(0, 1)$	2.90	100.03	102.93	378	748	0.79	477
$(0, 5)$	15.61	143.58	159.2	374	752	0.815	421
$(0, 10)$	55.73	259.3	315.06	374	752	0.8164	480
$(0, 15)$	113.58	459.60	573.18	377	749	0.711	413
$(1, 2)$	8.33	114.55	122.88	368	758	0.706	425
$(3, 5)$	31.95	204.42	236.38	411	715	0.831	496
$(4, 8)$	72.23	397.91	470.14	371	755	0.6622	481
$(8, 14)$	213.96	2924.61	3138.07	378	748	0.825	496

Table: Experimental results for thread scheduling.

Information Leakage in Dining Cryptographer's protocol

- Three cryptographers having dinner, want to find out who paid for it, one of them or their boss, which respecting each other's privacy.
- They each flip a coin and inform the person on the right about its outcome.
- If the cryptographer didn't pay, he announces whether the two coin results he knows are the same ('a') or different, else he states the opposite.

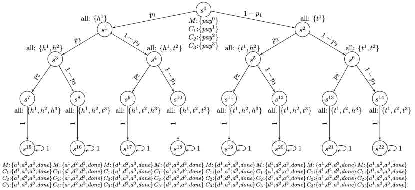


Figure: PDTMC for dining cryptographer's protocol

$$\varphi_{\mathrm{dc}} = \quad \forall\sigma.\ \forall\sigma'. \quad \Big(\big(\bigvee_{i \in 3} pay^{i}_{\sigma}\big) \wedge \big(\bigvee_{i \in 3} pay^{i}_{\sigma'}\big) \Big) \Rightarrow$$

$$\underbrace{\mathbb{P}\Big(\mathcal{F}\big(done_{\sigma} \wedge (a^{1}_{\sigma} \oplus a^{2}_{\sigma} \oplus a^{3}_{\sigma})\big)\Big)}_{F_1} = \underbrace{\mathbb{P}\Big(\mathcal{F}\big(done_{\sigma'} \wedge (a^{1}_{\sigma'} \oplus a^{2}_{\sigma'} \oplus a^{3}_{\sigma'})\big)\Big)}_{F_2}$$

Information Leakage in Dining Cryptographer's protocol: Result

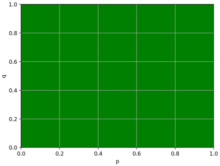


Figure: Parameter space for dining cryptographer's protocol

- The whole parameter domain of $[0, 1]^3$ is green as the property is satisfied for all parameters.
- It took us approximately 40 minutes to get the result.
- This was due to the large state-space and the lack of recognition of non-reachable states in our implementation.

Summary

Provided algorithm to synthesize parameters for a given PDTMC such that it satisfies a given probabilistic hyperproperty.

Our algorithm works in two steps.

- It computes symbolic conditions for satisfying the formula, involving rational functions on the set of parameters.
- Identify regions of satisfying, unsatisfying, or grey parameter configurations by decomposing the domain of parameter configurations.

Future work

- Optimize the current prototypical implementation to lead to better scalability.
- Improve by exploiting the existence of symmetries.
- Extend the current work to more expressive logics like HyperPCTL and HyperPCTL*