More Scalable LTL Model Checking via Discovering Design-Space Dependencies (D^3)

Rohit Dureja and Kristin Yvonne Rozier

IOWA STATE UNIVERSITY + Laboratory for Temporal Logic

April 17, 2018

Thanks to NSF CAREER Award CNS-1552934 for supporting this work.

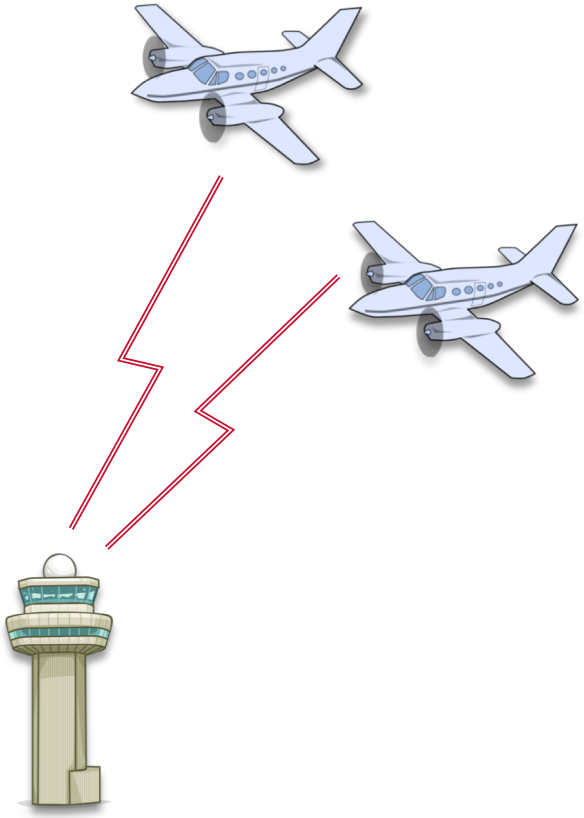
What is a Design Space?

Airspace Allocation

Problem Statement: Design of allocation policies to ensure all airborne agents are safely separated.

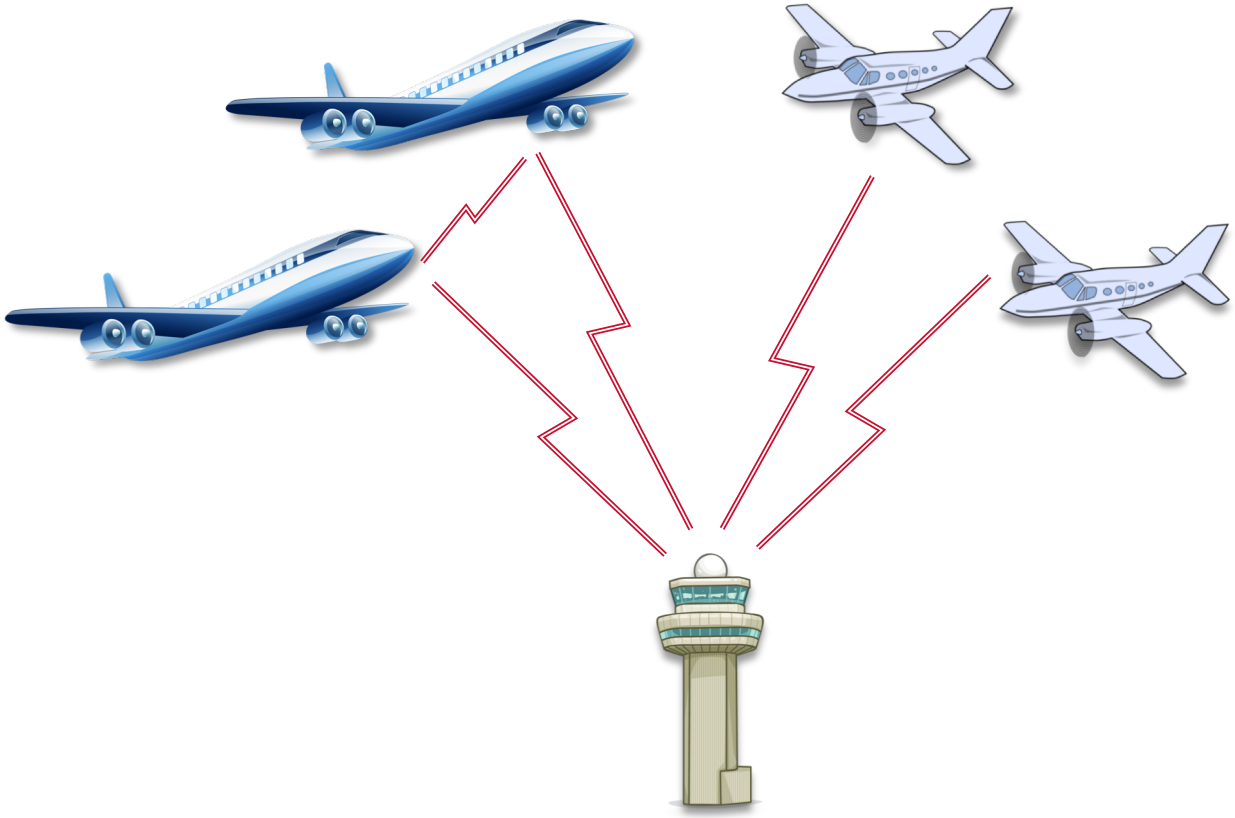
Airspace Allocation

Problem Statement: Design of allocation policies to ensure all airborne agents are safely separated.



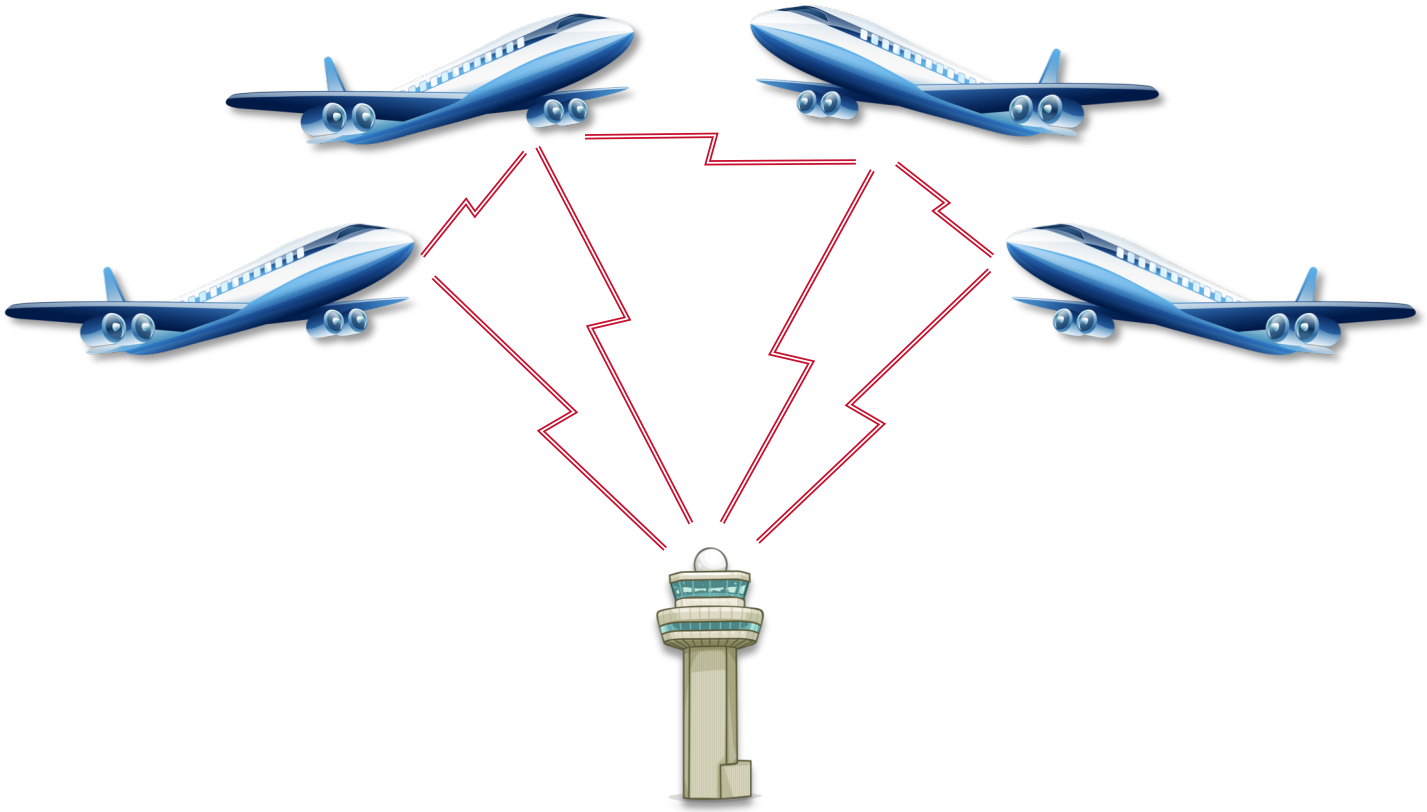
Airspace Allocation

Problem Statement: Design of allocation policies to ensure all airborne agents are safely separated.



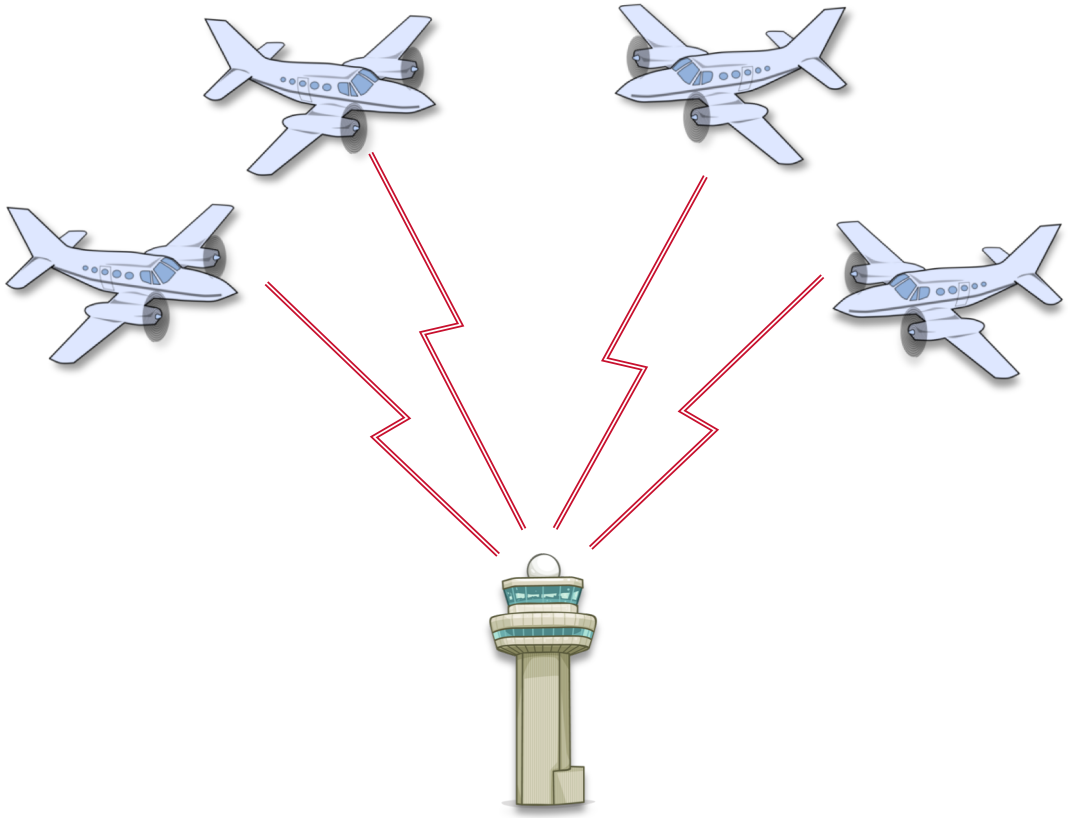
Airspace Allocation

Problem Statement: Design of allocation policies to ensure all airborne agents are safely separated.



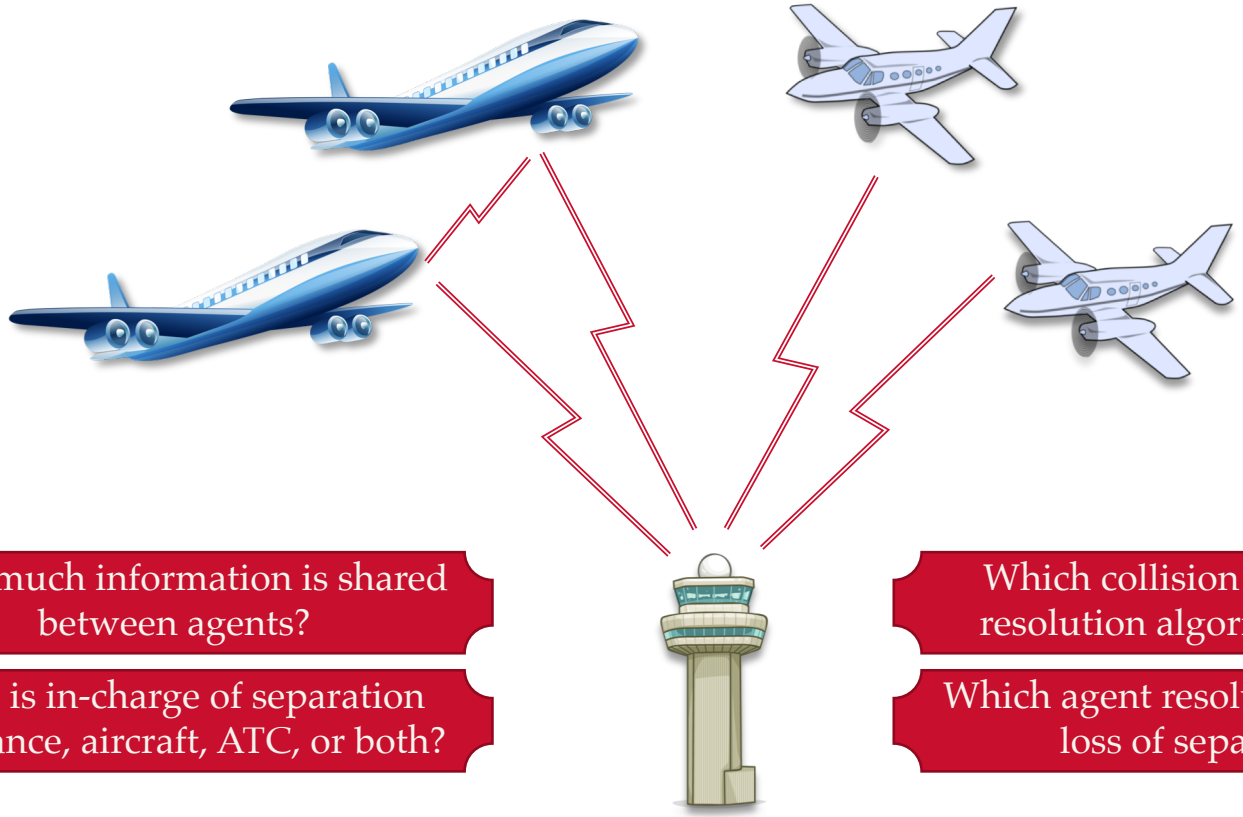
Airspace Allocation

Problem Statement: Design of allocation policies to ensure all airborne agents are safely separated.



Airspace Allocation

Problem Statement: Design of allocation policies to ensure all airborne agents are safely separated.



How much information is shared between agents?

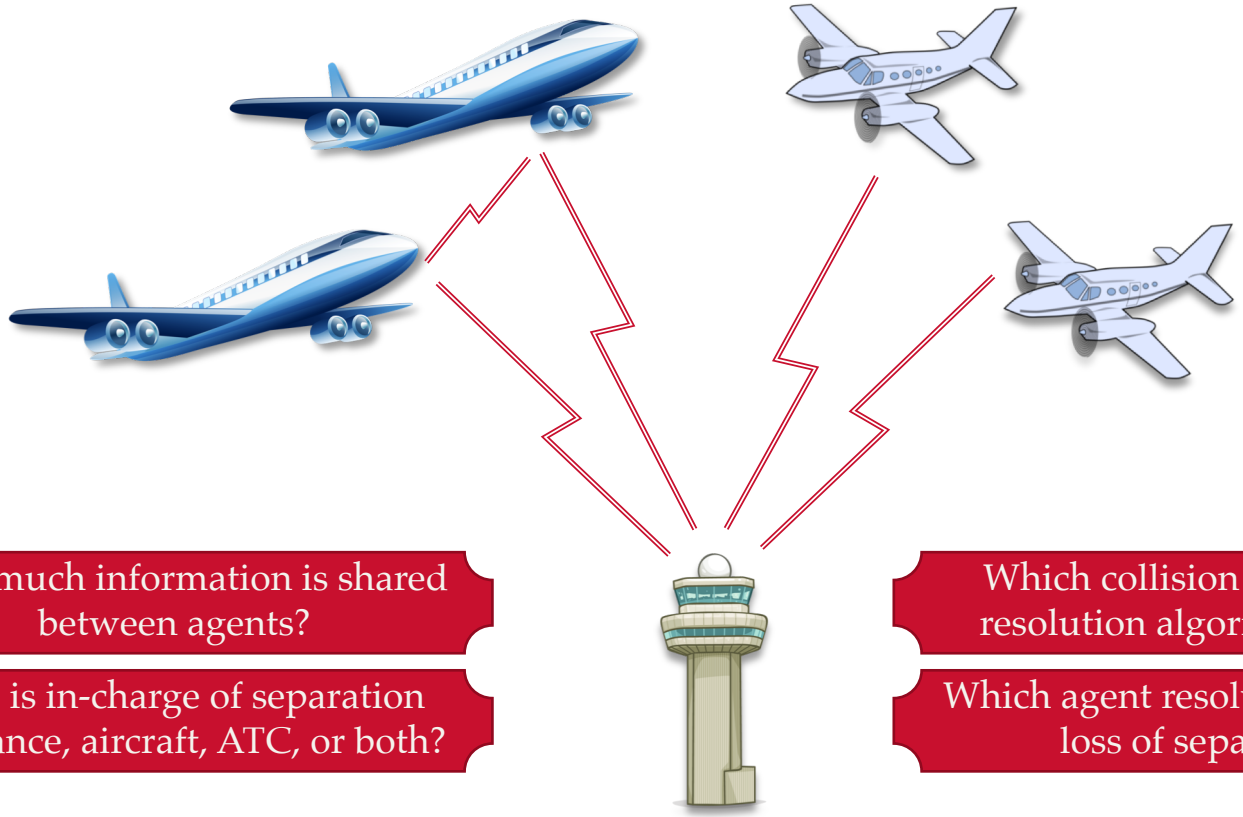
Who is in-charge of separation assurance, aircraft, ATC, or both?

Which collision detection & resolution algorithm to use?

Which agent resolves a potential loss of separation?

Airspace Allocation

Problem Statement: Design of allocation policies to ensure all airborne agents are safely separated.



How much information is shared between agents?

Who is in-charge of separation assurance, aircraft, ATC, or both?

Which collision detection & resolution algorithm to use?

Which agent resolves a potential loss of separation?

Lots of Design Choices!

What is a Design Space?

What is a Design Space?

Set of Design Choices for a System.

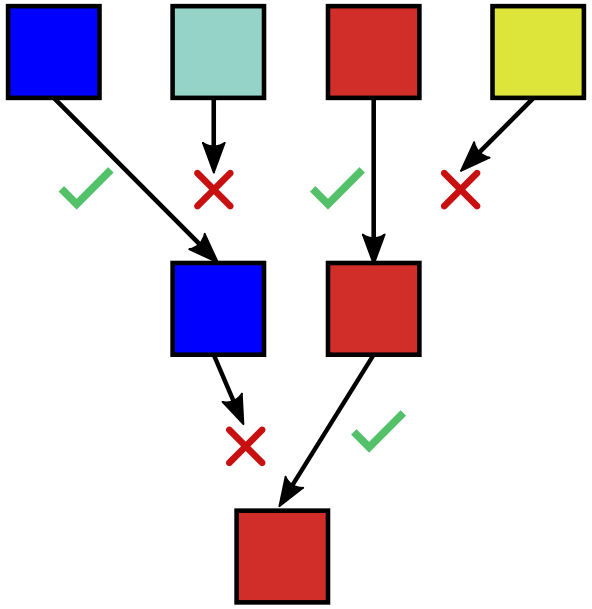
Design Space Analysis

Design Space Analysis

Complex systems are often modeled as design spaces.

Design Space Analysis

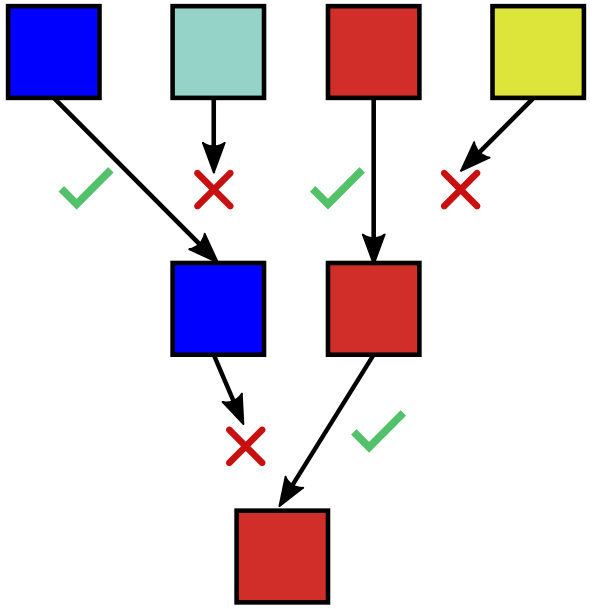
Complex systems are often modeled as design spaces.



Alternative comparison via design space exploration

Design Space Analysis

Complex systems are often modeled as design spaces.



Alternative comparison via design space exploration

Model Checking!

Modeling Design Spaces

Modeling Design Spaces

- Classical Method
  - *every design choice leads to a separate model*

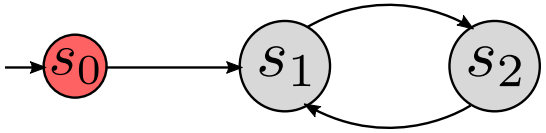
Modeling Design Spaces

- Classical Method
  - *every design choice leads to a separate model*
- Scalable Method
  - *every design choice is **parameter** to a meta-model*

Modeling Design Spaces

- Classical Method
  - *every design choice leads to a separate model*
- Scalable Method
  - *every design choice is **parameter** to a meta-model*

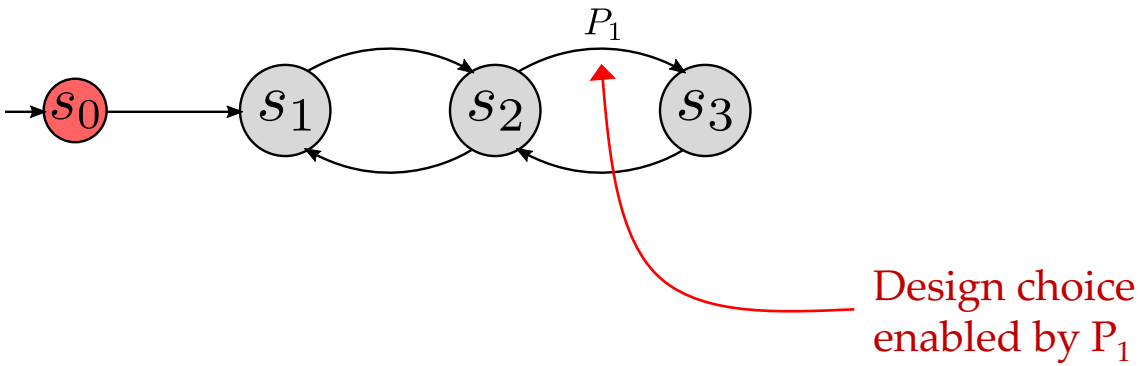
Simple Example



Modeling Design Spaces

- Classical Method
  - *every design choice leads to a separate model*
- Scalable Method
  - *every design choice is **parameter** to a meta-model*

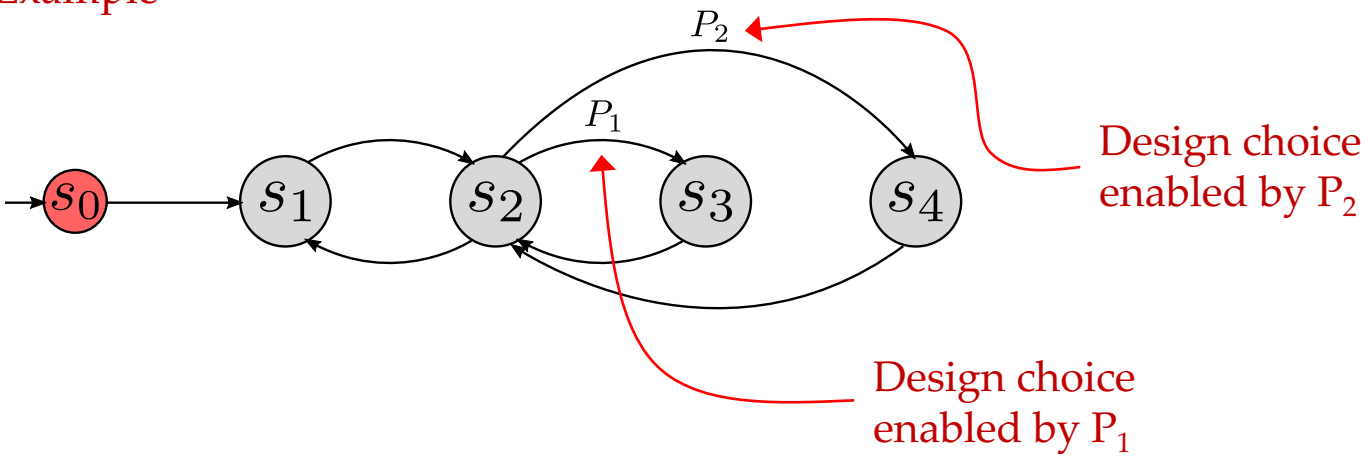
Simple Example



Modeling Design Spaces

- Classical Method
  - *every design choice leads to a separate model*
- Scalable Method
  - *every design choice is **parameter** to a meta-model*

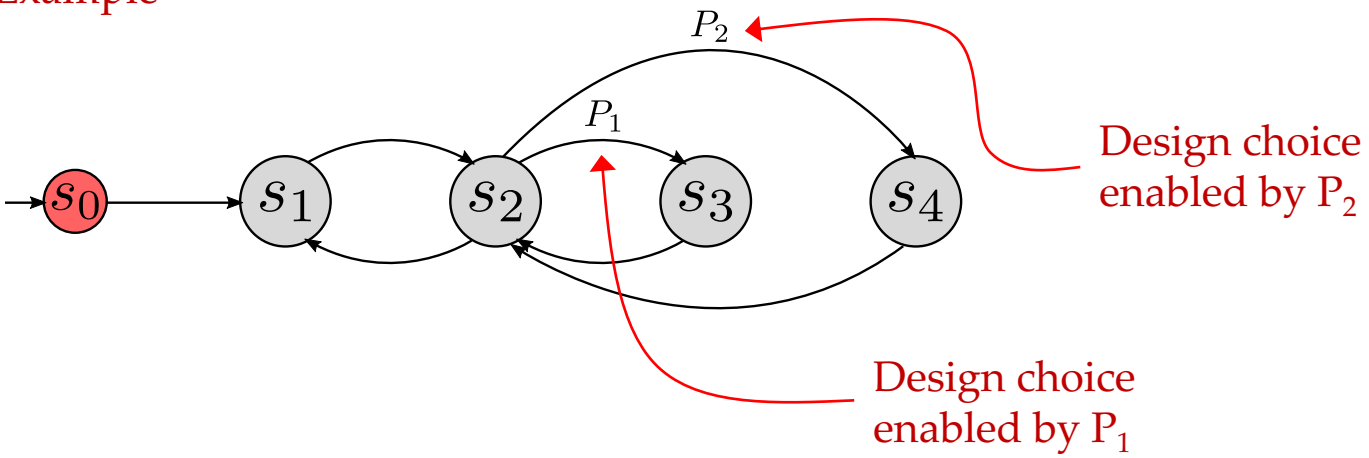
Simple Example



Modeling Design Spaces

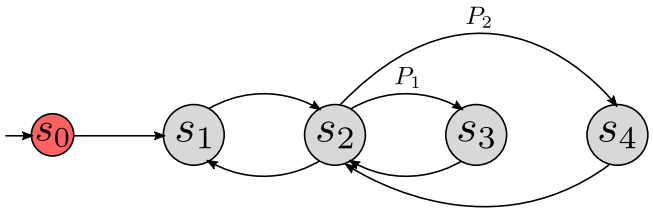
- Classical Method
  - *every design choice leads to a separate model*
- Scalable Method
  - *every design choice is **parameter** to a meta-model*

Simple Example

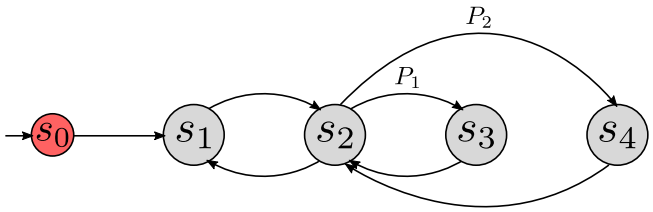


Combinatorial Transition System (CTS)

Model Checking Combinatorial Systems

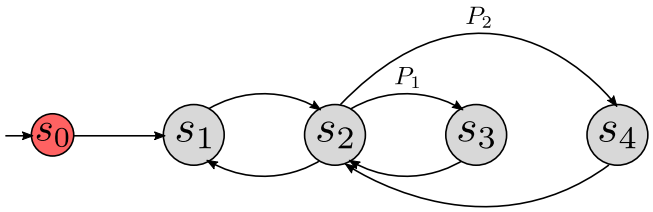


Model Checking Combinatorial Systems



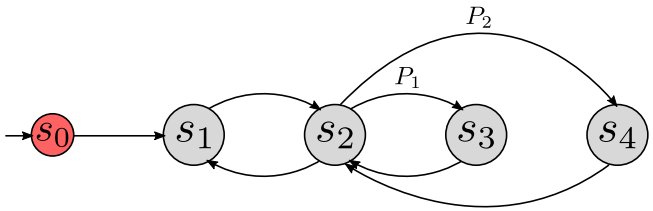
- Parameters as variables
  - *state-space explosion, counterexample belongs to single design choice*

Model Checking Combinatorial Systems

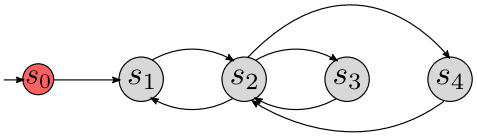


- Parameters as variables
  - *state-space explosion, counterexample belongs to single design choice*
- Parameter configurations
  - *every parameter configuration leads to an individual model*

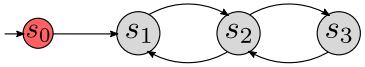
Model Checking Combinatorial Systems



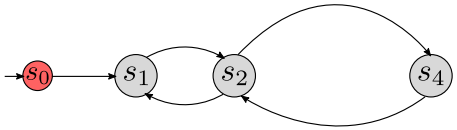
- Parameters as variables
  - *state-space explosion, counterexample belongs to single design choice*
- Parameter configurations
  - *every parameter configuration leads to an individual model*



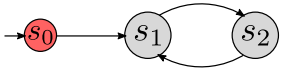
$(P_1 = \top, P_2 = \top)$



$(P_1 = \top, P_2 = \bot)$



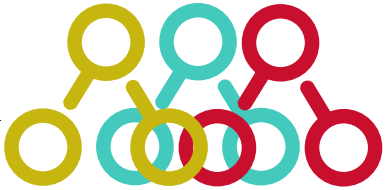
$(P_1 = \bot, P_2 = \top)$



$(P_1 = \bot, P_2 = \bot)$

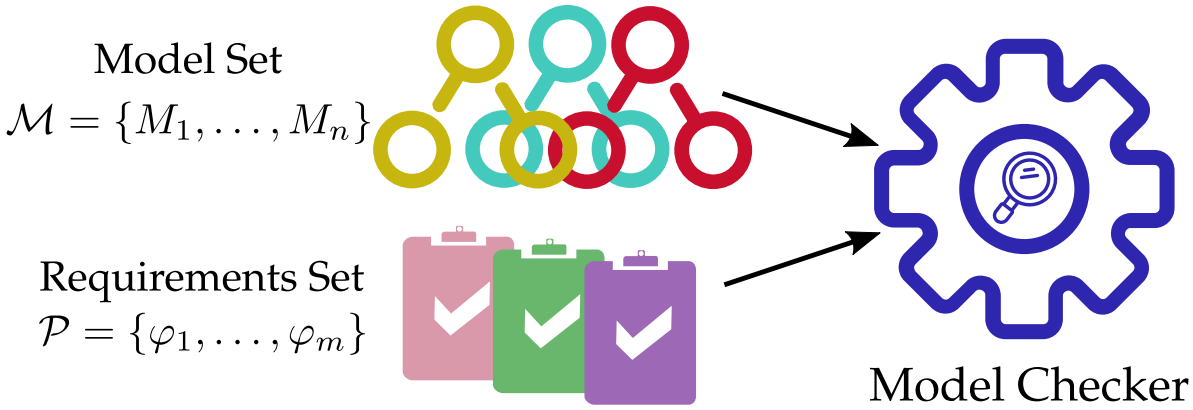
Model Checking Design Spaces

Model Set
$\mathcal{M} = \{M_1, \ldots, M_n\}$

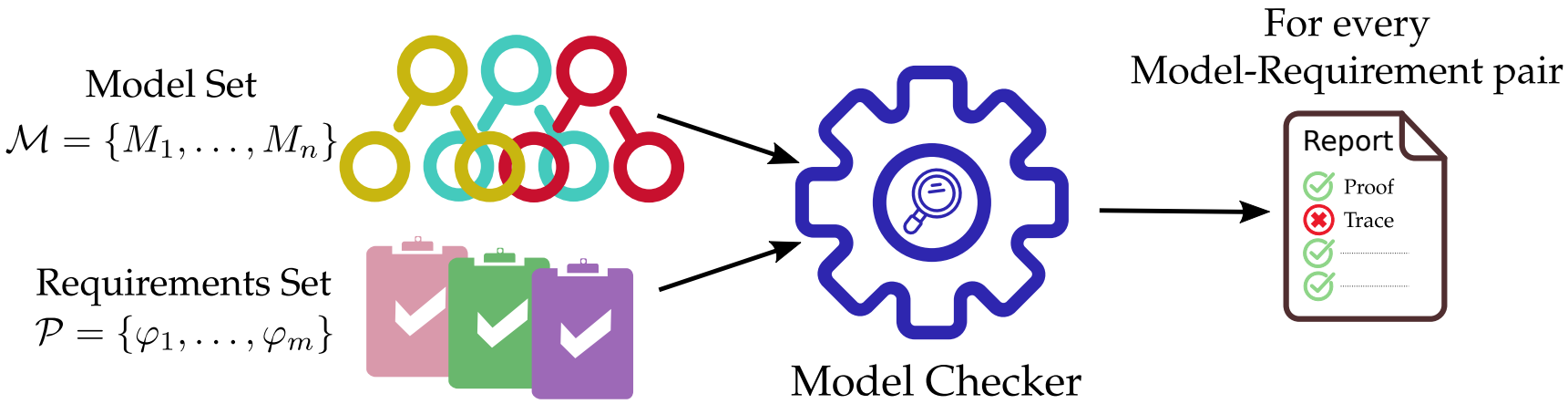


Requirements Set
$\mathcal{P} = \{\varphi_1, \ldots, \varphi_m\}$

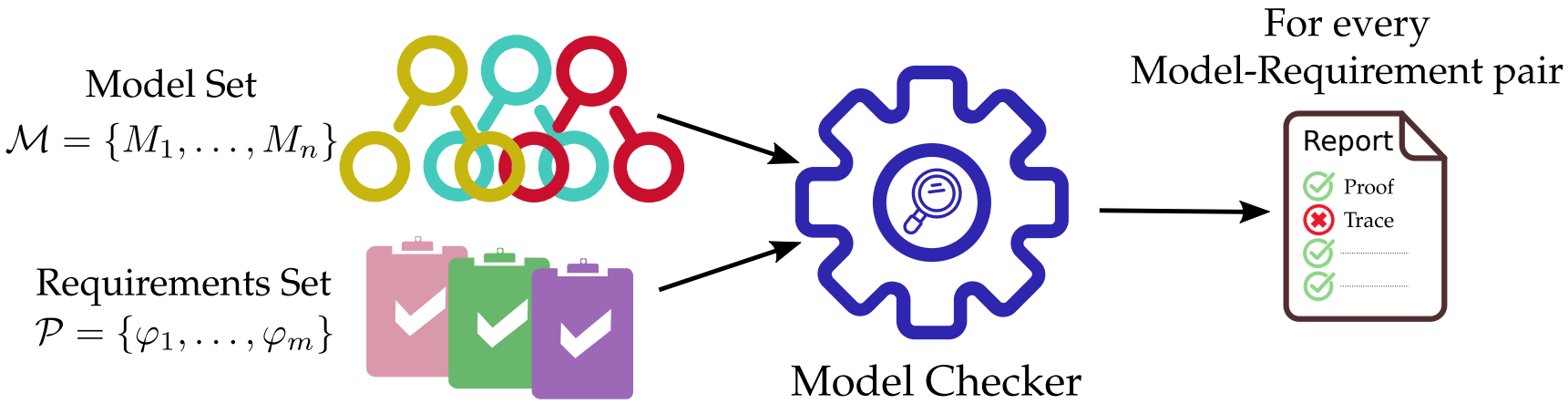
Model Checking Design Spaces



Model Checking Design Spaces



Model Checking Design Spaces



Our Goal

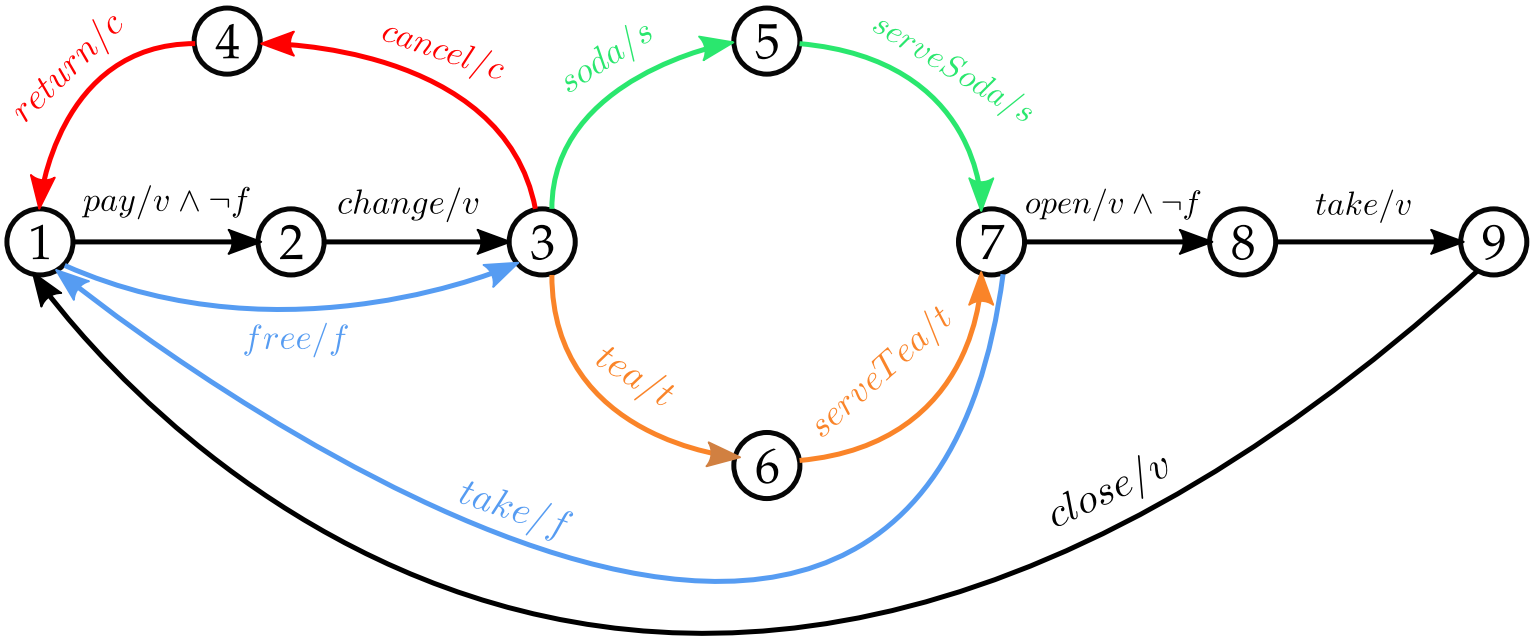
Make model-checking for design-spaces more scalable

Related Work

Related Work

1. Software Product Line verification
   (Resenmüller et al., 2010; Schirmeier et al., 2009; Classen et al., 2010, 2011, 2013; Dimovski et al., 2015)

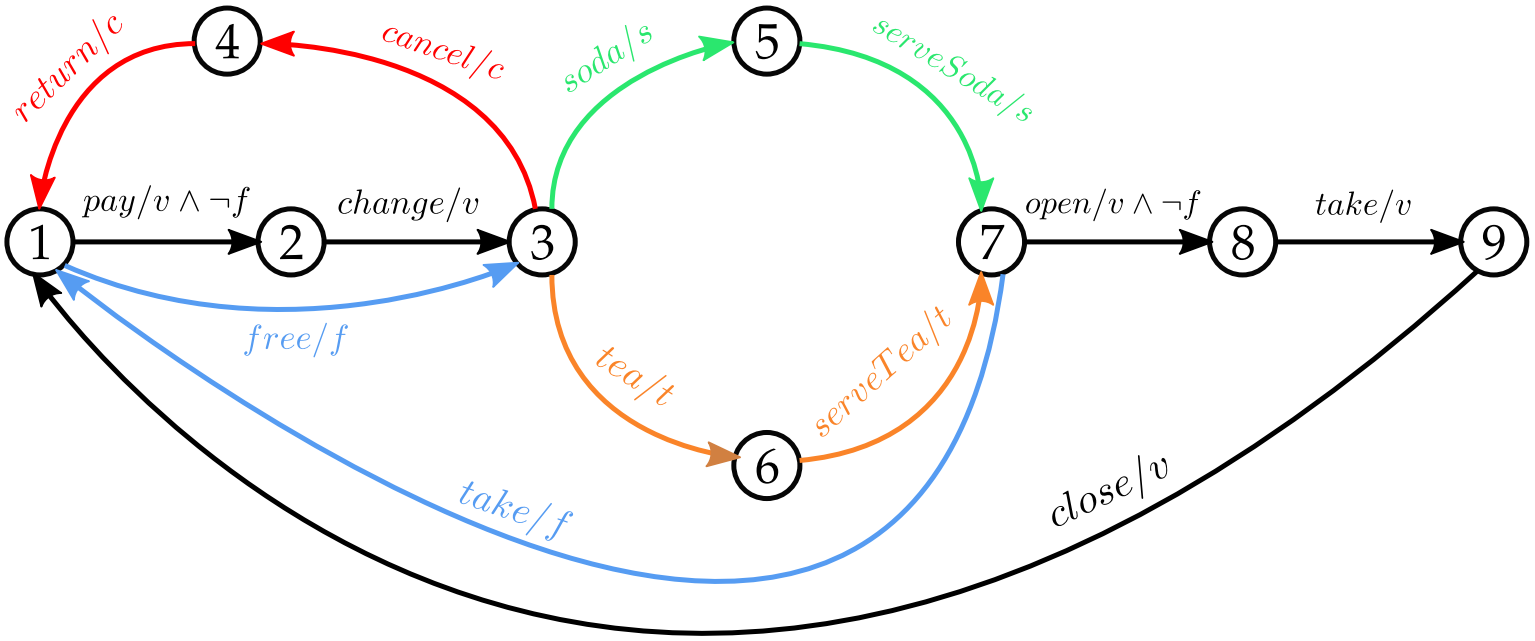
Many Models
Many Requirements



Related Work

1. Software Product Line verification
   (Resenmüller et al., 2010; Schirmeier et al., 2009; Classen et al., 2010, 2011, 2013; Dimovski et al., 2015)

Many Models
Many Requirements

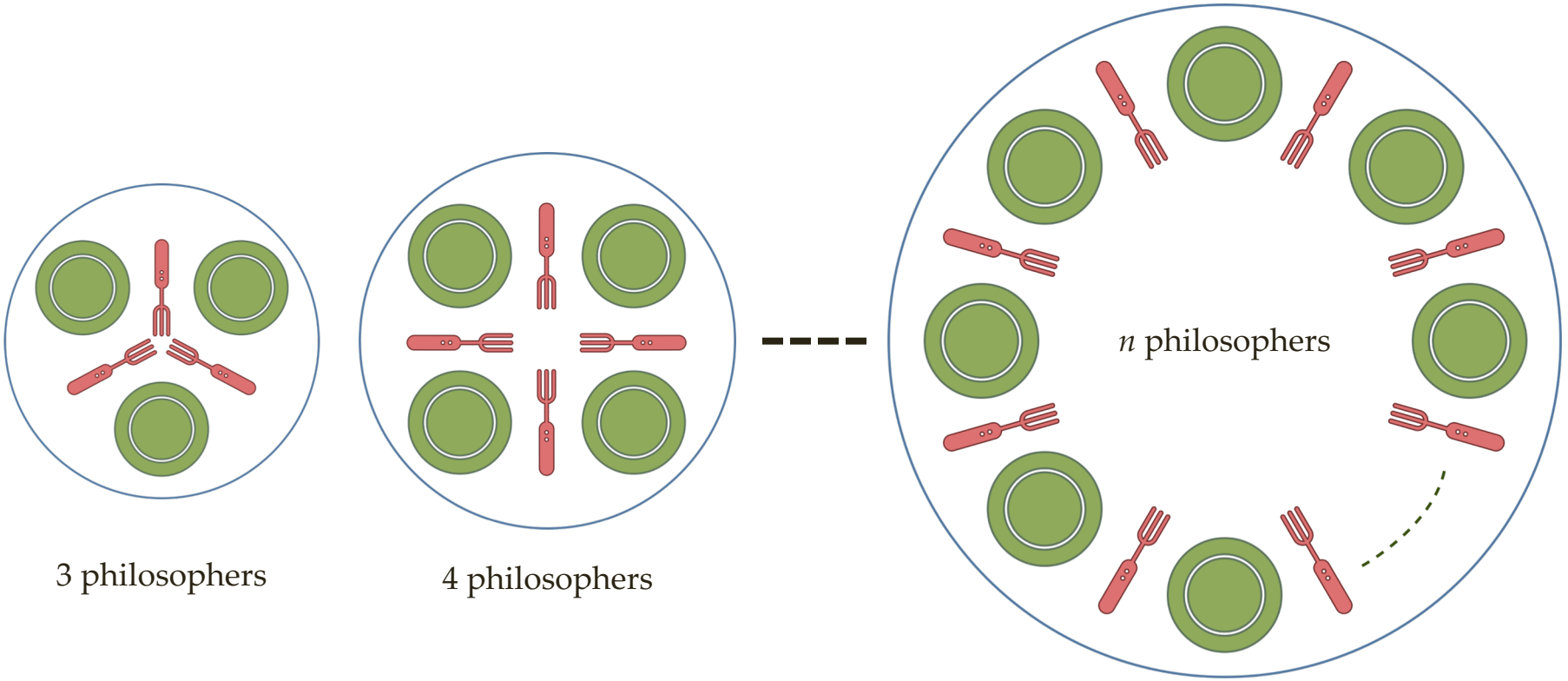


We focus on a more general approach that works with off-the-shelf model checkers

Related Work

2. Parameterized model checking (Emerson and Kahlon, 2000)

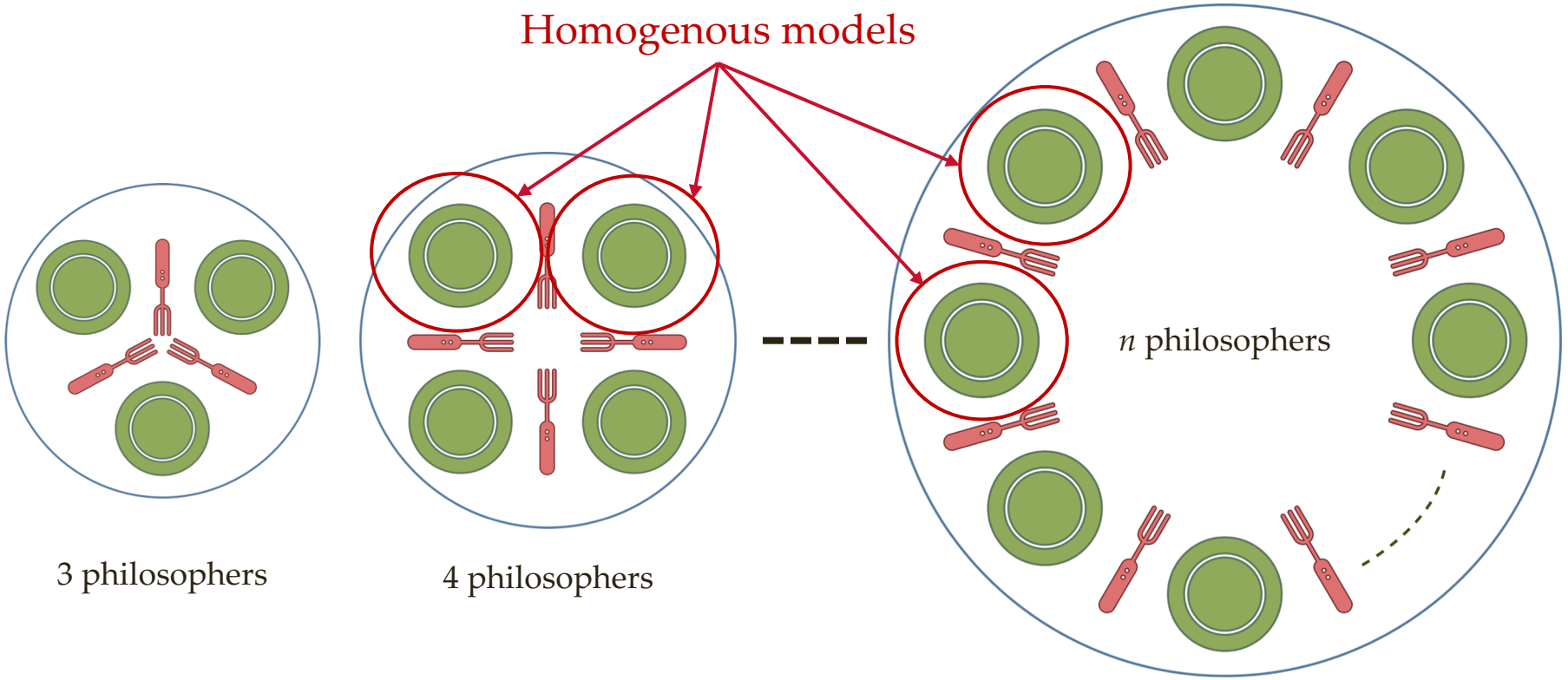
Many Models
One Requirement



Related Work

2. Parameterized model checking (Emerson and Kahlon, 2000)

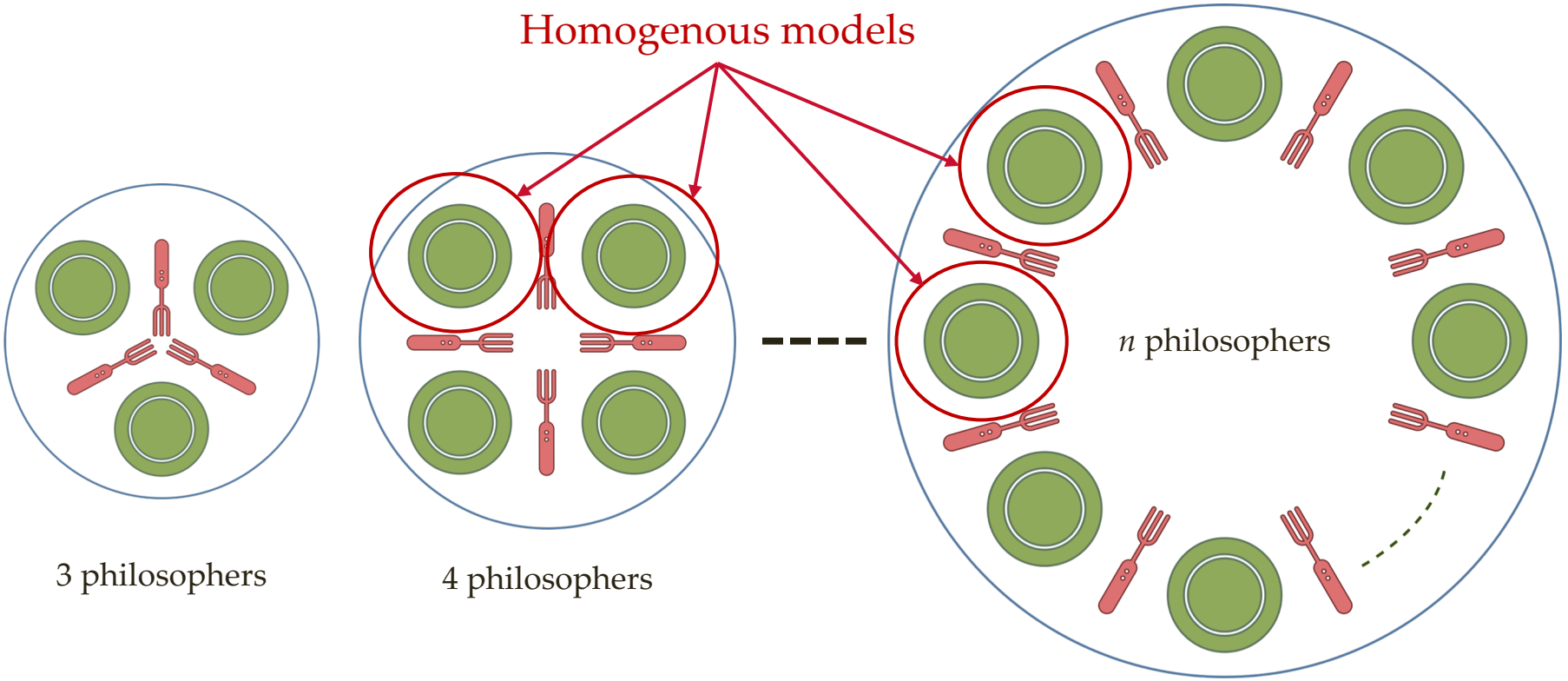
Many Models
One Requirement



Related Work

2. Parameterized model checking (Emerson and Kahlon, 2000)

Many Models
One Requirement



We focus on heterogenous and finite number of models

Related Work

3. Multi-property model checking

(Cabodi et al., 2010, 2011, 2017; Goldberg et al., 2018; Khasidashvili et al., 2006; Khasidashvili and Nadel, 2012; Camurati et al., 2014; Dureja and Rozier, 2017)

One Model
Many Requirements

Related Work

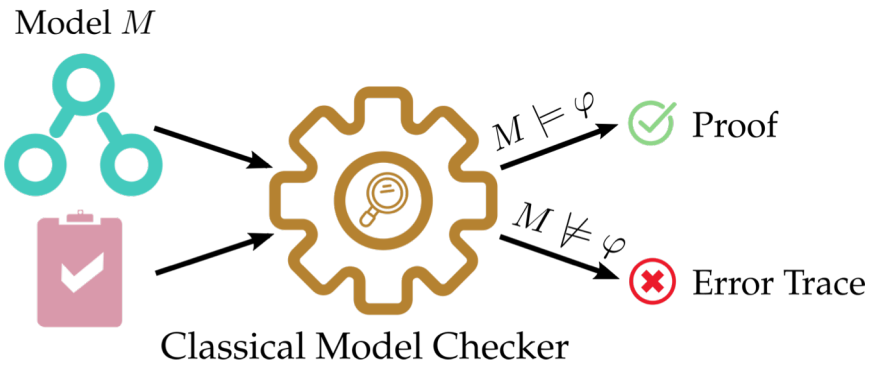
3. Multi-property model checking

(Cabodi et al., 2010, 2011, 2017; Goldberg et al., 2018; Khasidashvili et al., 2006; Khasidashvili and Nadel, 2012; Camurati et al., 2014; Dureja and Rozier, 2017)

One Model
Many Requirements

Requirements

φ_1
φ_2
φ_3
φ_4
φ_5
φ_6
φ_7
φ_8

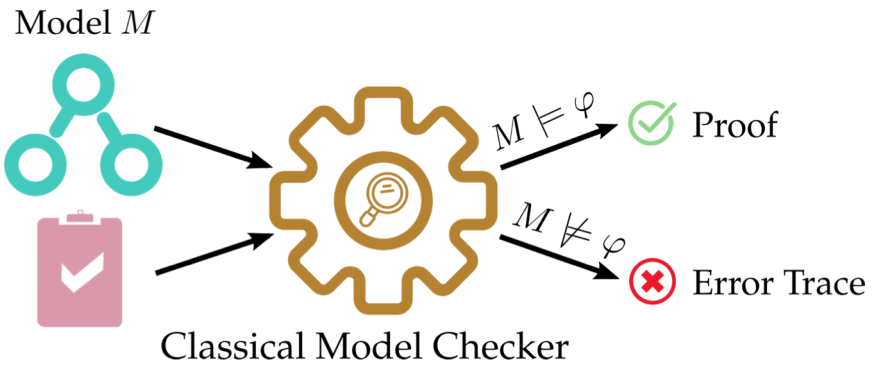


Related Work

3. Multi-property model checking

(Cabodi et al., 2010, 2011, 2017; Goldberg et al., 2018; Khasidashvili et al., 2006; Khasidashvili and Nadel, 2012; Camurati et al., 2014; Dureja and Rozier, 2017)

One Model
Many Requirements

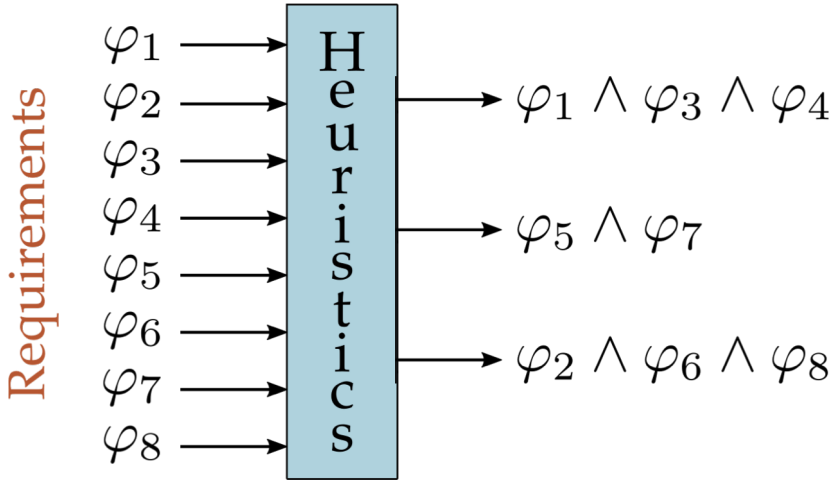


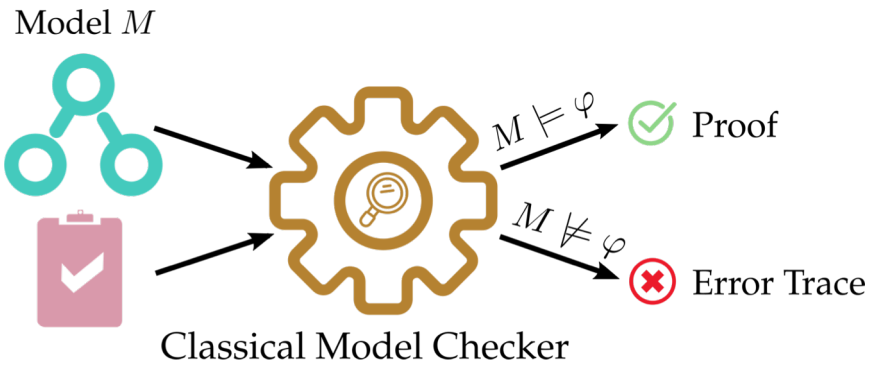
Related Work

3. Multi-property model checking

(Cabodi et al., 2010, 2011, 2017; Goldberg et al., 2018; Khasidashvili et al., 2006; Khasidashvili and Nadel, 2012; Camurati et al., 2014; Dureja and Rozier, 2017)

One Model
Many Requirements



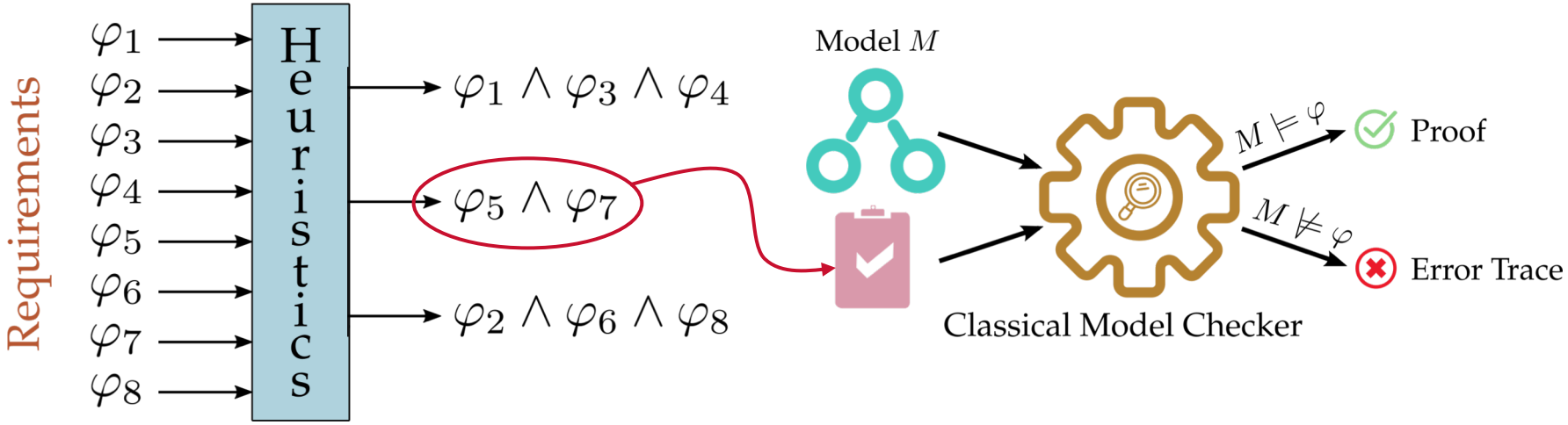


Related Work

3. Multi-property model checking

(Cabodi et al., 2010, 2011, 2017; Goldberg et al., 2018; Khasidashvili et al., 2006; Khasidashvili and Nadel, 2012; Camurati et al., 2014; Dureja and Rozier, 2017)

One Model
Many Requirements

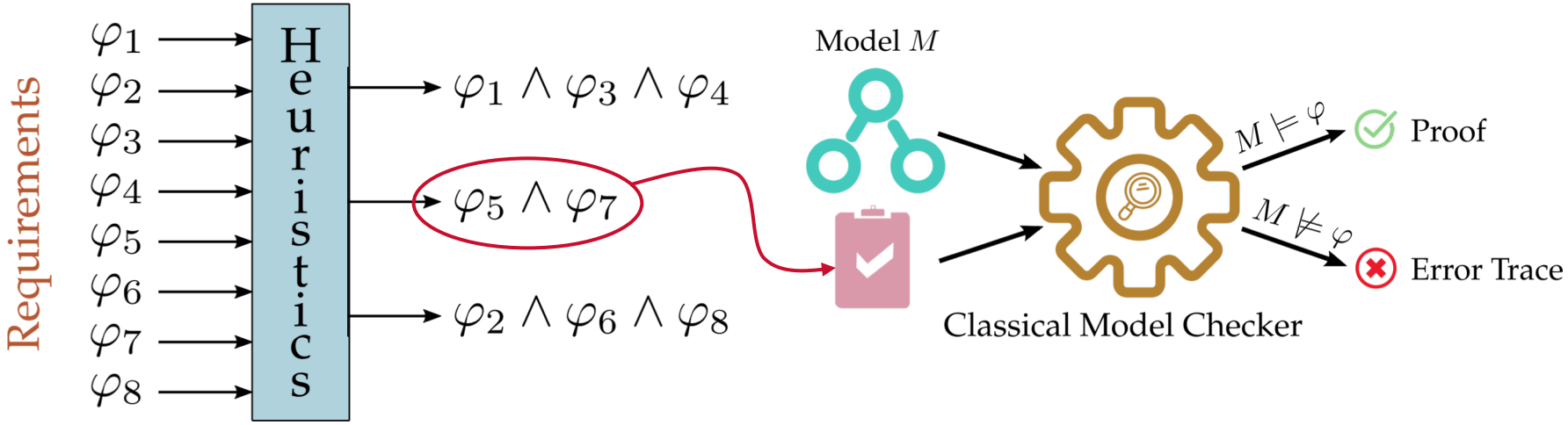


Related Work

3. Multi-property model checking

(Cabodi et al., 2010, 2011, 2017; Goldberg et al., 2018; Khasidashvili et al., 2006; Khasidashvili and Nadel, 2012; Camurati et al., 2014; Dureja and Rozier, 2017)

One Model
Many Requirements



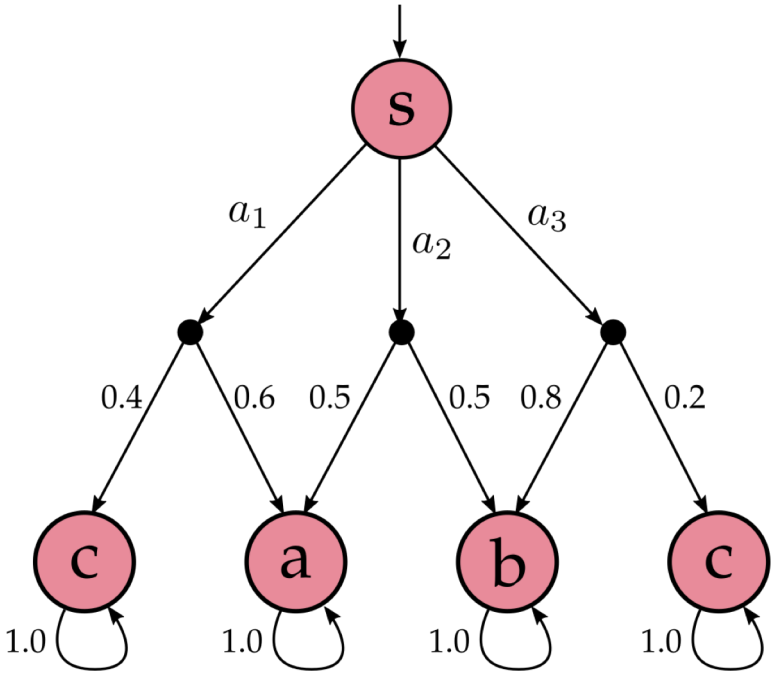
We focus on reducing the number of total model checking runs

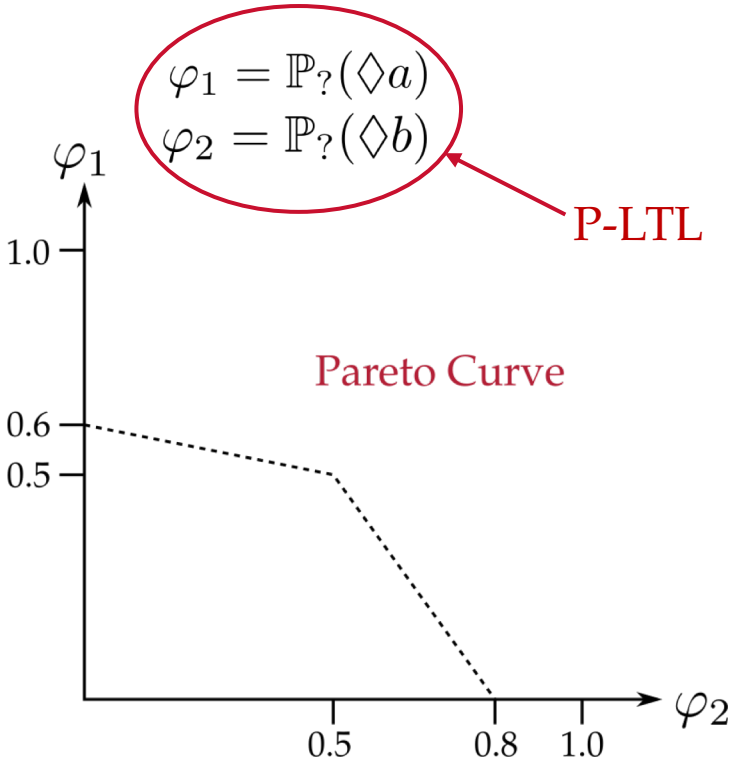
Related Work

4. Multi-objective probabilistic model checking

(Denhert et al., 2015, 2016, Baier et al., 2014, Etessami et al., 2007, Forejt et al., 2011, Kwiatkowska et al., 2013, Quatmann et al., 2016)

One Model
Many Requirements



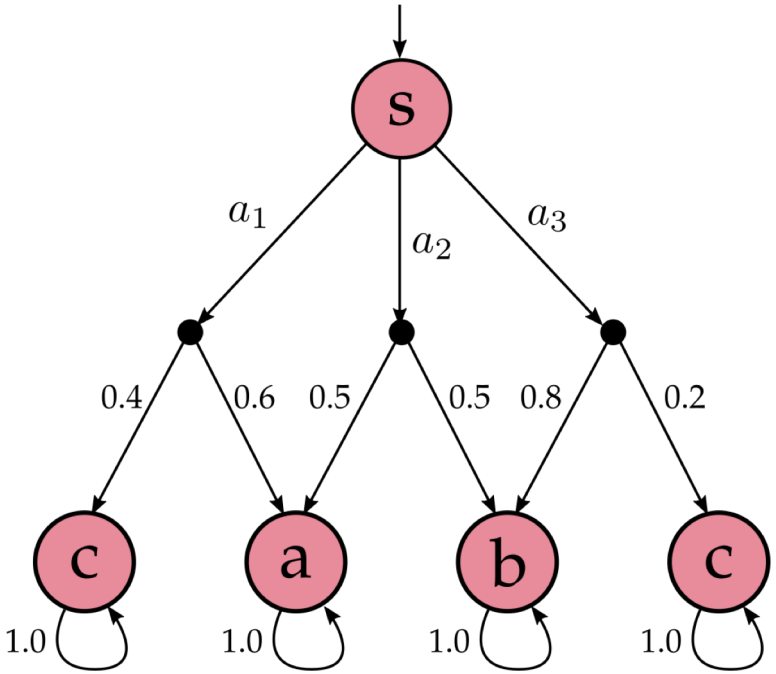


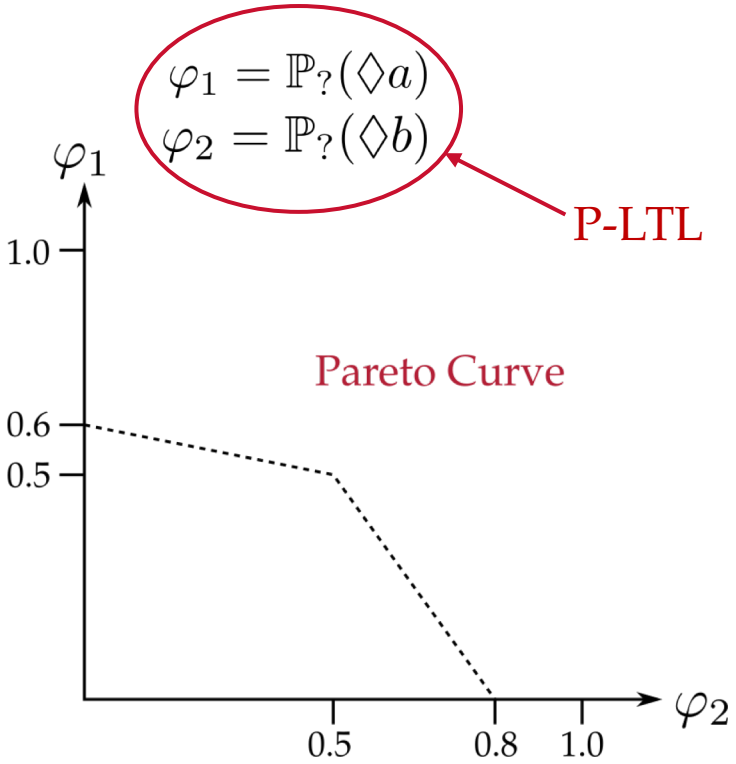
Related Work

4. Multi-objective probabilistic model checking

(Denhert et al., 2015, 2016, Baier et al., 2014, Etessami et al., 2007, Forejt et al., 2011, Kwiatkowska et al., 2013, Quatmann et al., 2016)

One Model
Many Requirements

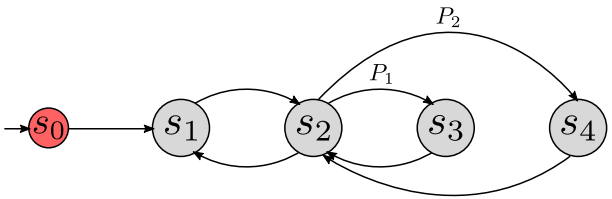




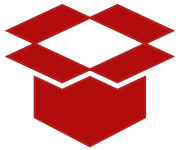
We focus on LTL/Invariant model checking with Boolean result.

Discover Design-Space Dependencies (D^3)

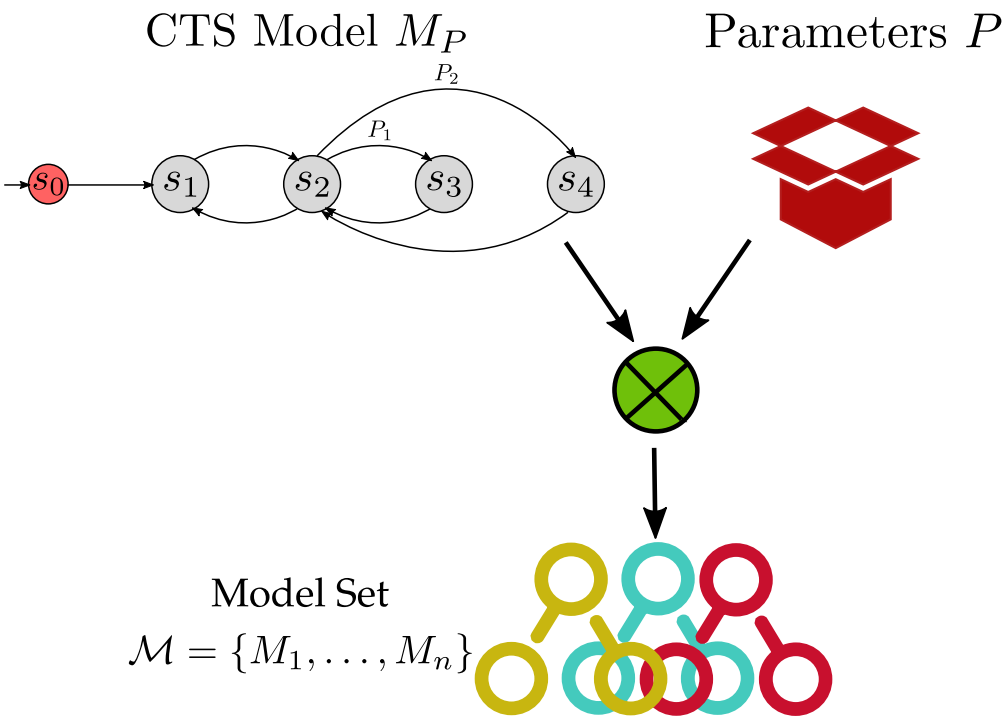
CTS Model M_P



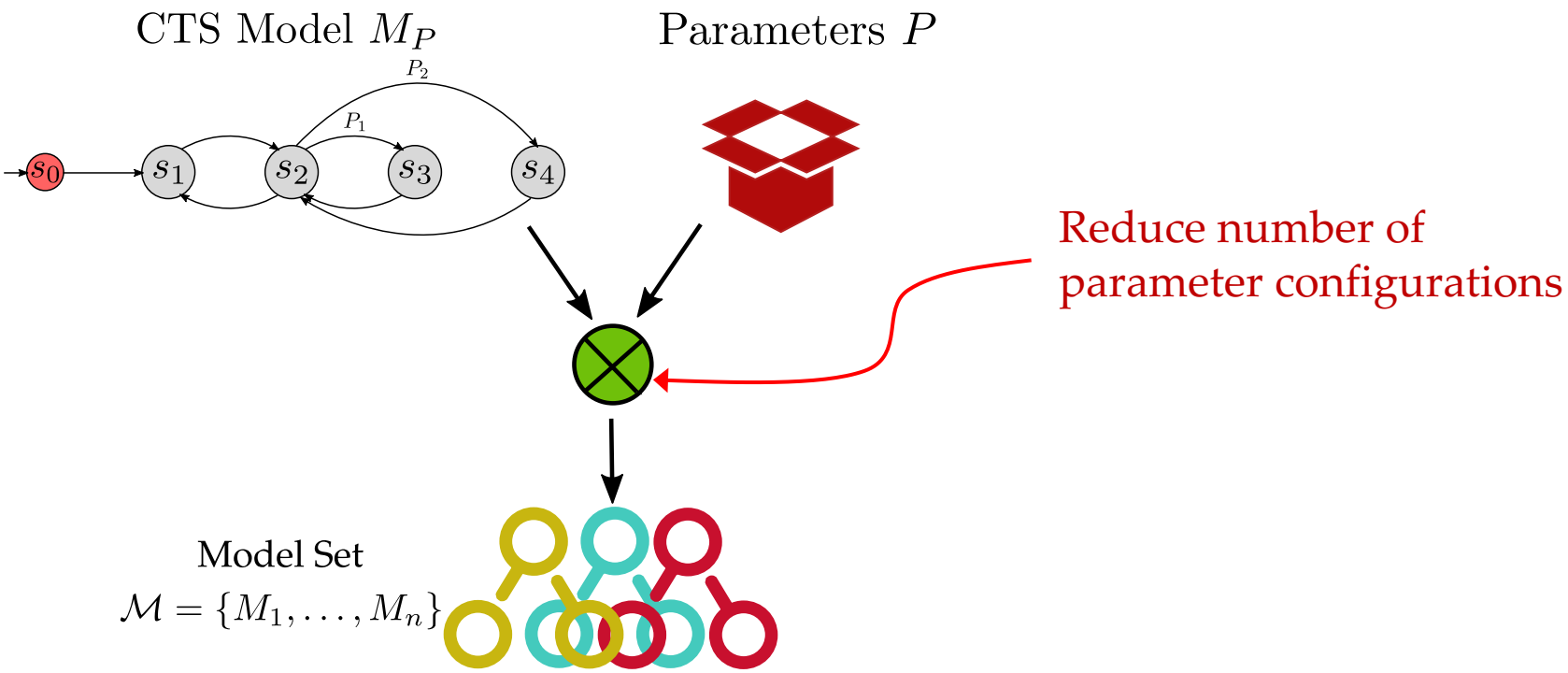
Parameters P



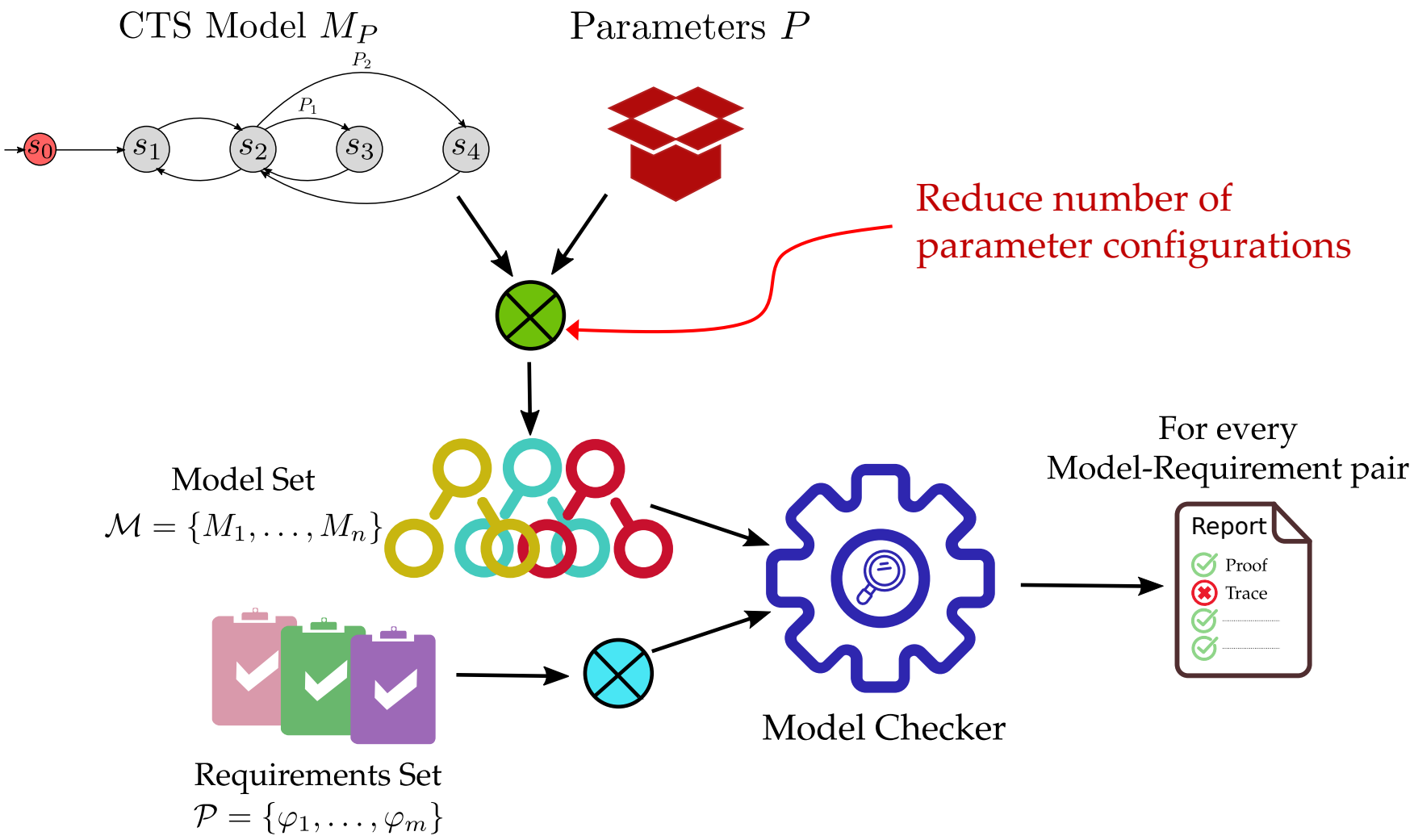
Discover Design-Space Dependencies (D^3)



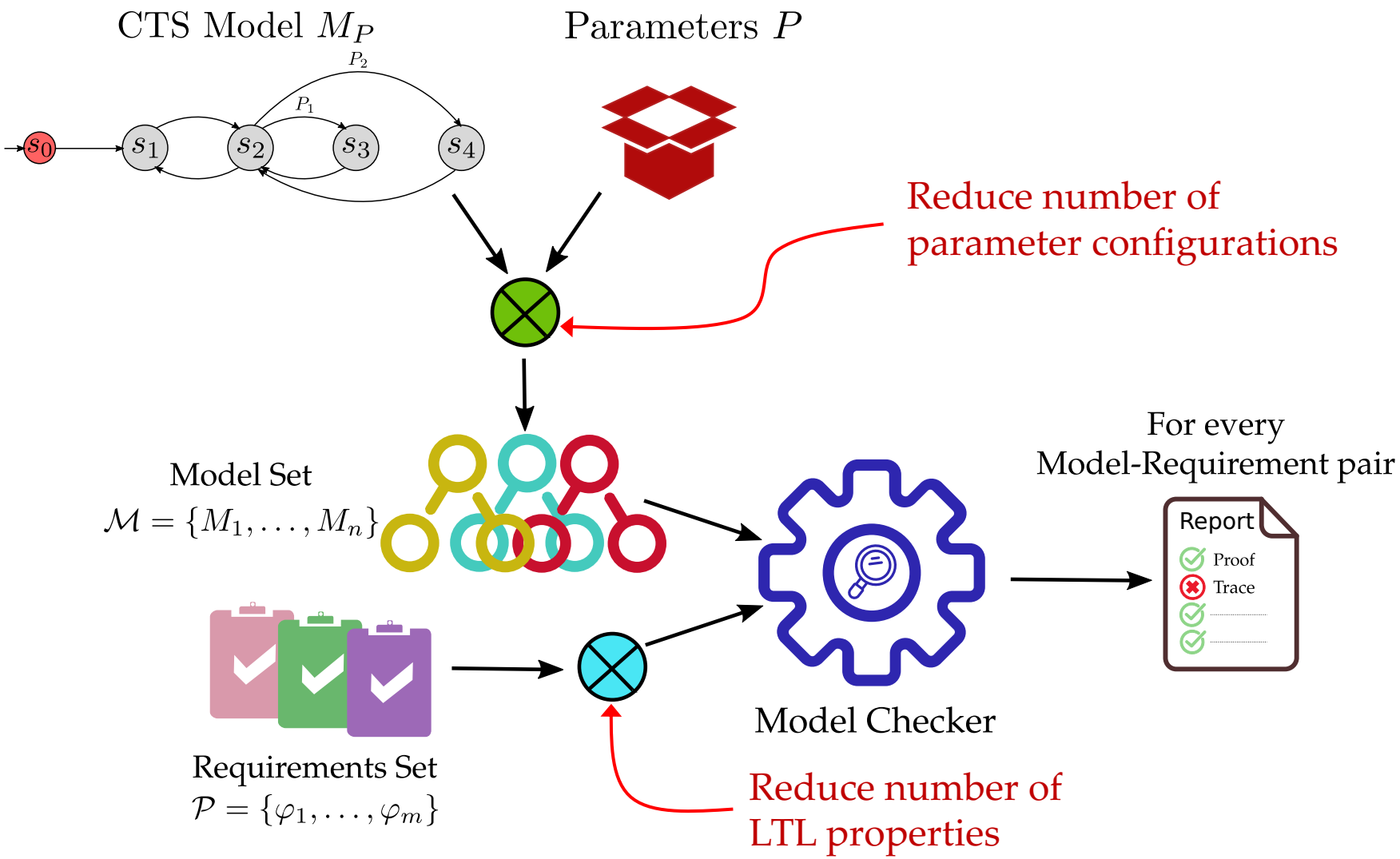
Discover Design-Space Dependencies (D^3)



Discover Design-Space Dependencies (D^3)



Discover Design-Space Dependencies (D^3)

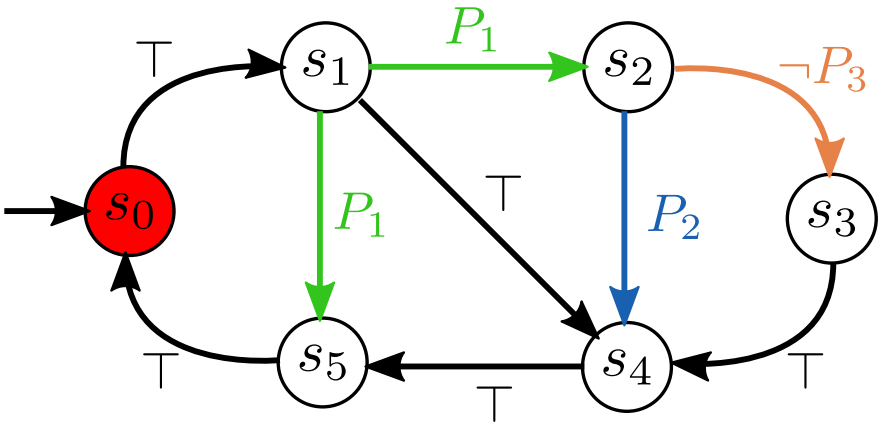


Parameter Configurations

Parameter Configurations

Motivating Example

M_P

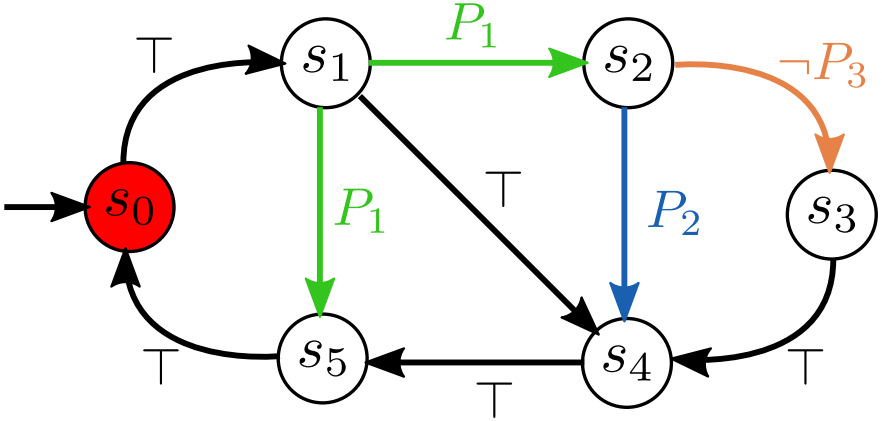


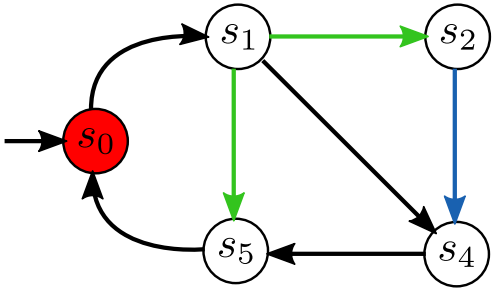
Boolean parameters P_1, P_2, P_3

Parameter Configurations

Motivating Example

M_P





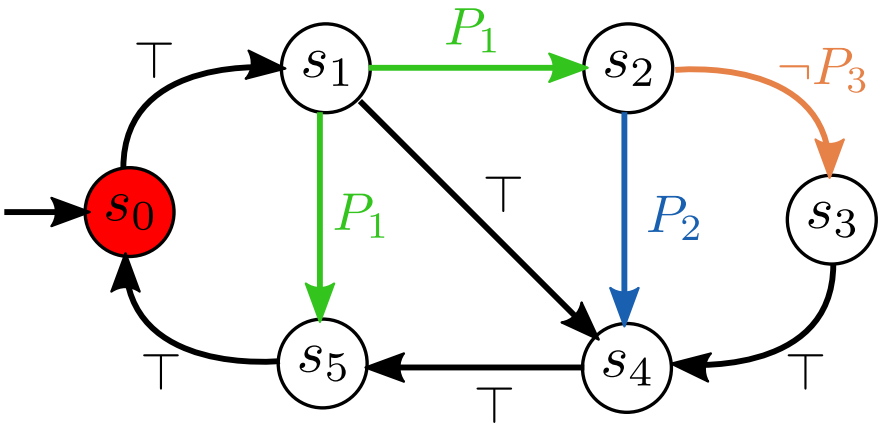
$(P_1 = \top, P_2 = \top, P_3 = \top)$

Boolean parameters P_1, P_2, P_3

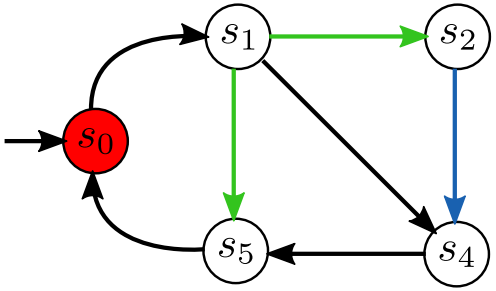
Parameter Configurations

Motivating Example

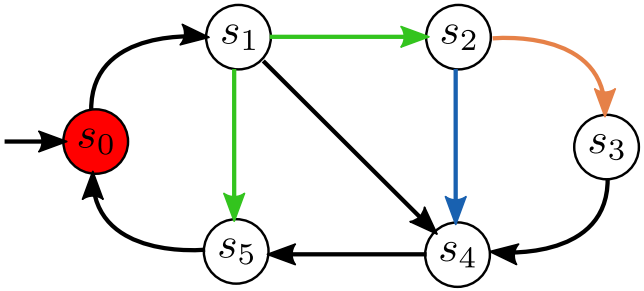
M_P



Boolean parameters P_1, P_2, P_3



$(P_1 = \top, P_2 = \top, P_3 = \top)$

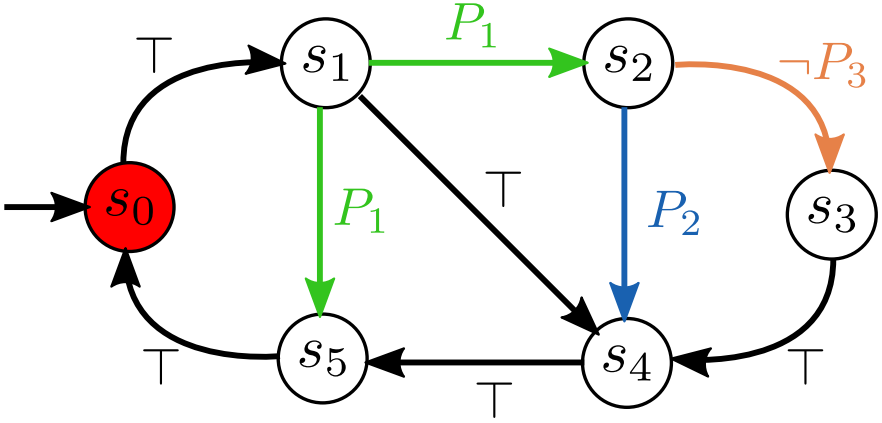


$(P_1 = \top, P_2 = \top, P_3 = \bot)$

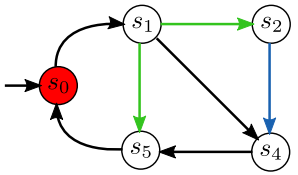
Parameter Configurations

Motivating Example

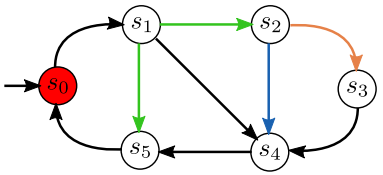
M_P



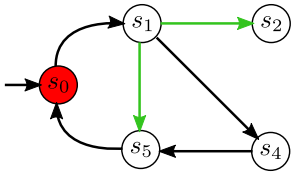
Boolean parameters P_1, P_2, P_3



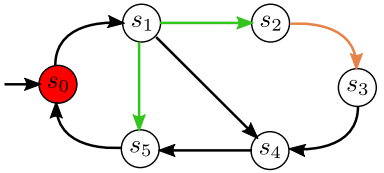
$(P_1 = \top, P_2 = \top, P_3 = \top)$



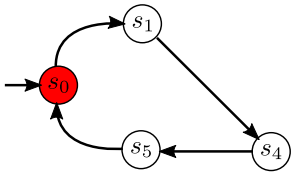
$(P_1 = \top, P_2 = \top, P_3 = \bot)$



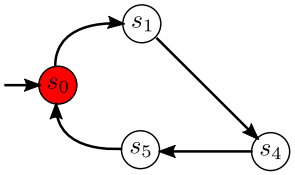
$(P_1 = \top, P_2 = \bot, P_3 = \top)$



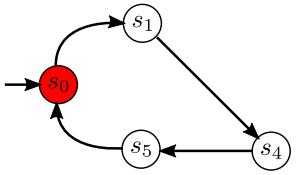
$(P_1 = \top, P_2 = \bot, P_3 = \bot)$



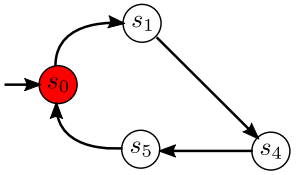
$(P_1 = \bot, P_2 = \top, P_3 = \top)$



$(P_1 = \bot, P_2 = \top, P_3 = \bot)$



$(P_1 = \bot, P_2 = \bot, P_3 = \top)$

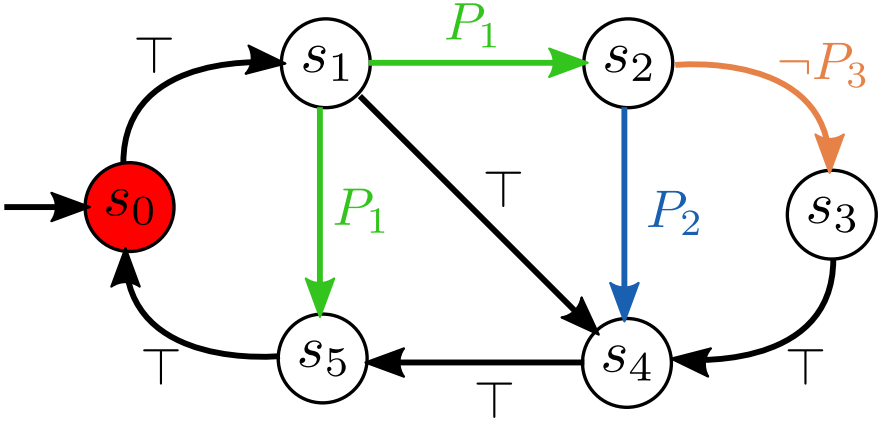


$(P_1 = \bot, P_2 = \bot, P_3 = \bot)$

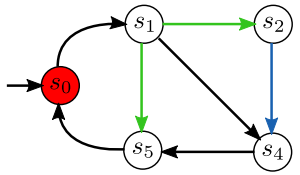
Parameter Configurations

Motivating Example

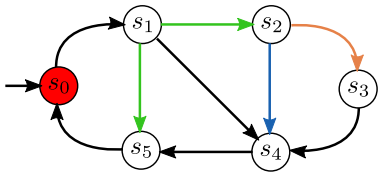
M_P



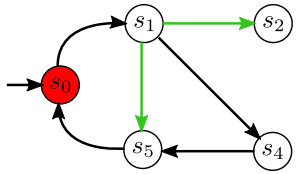
Boolean parameters P_1, P_2, P_3



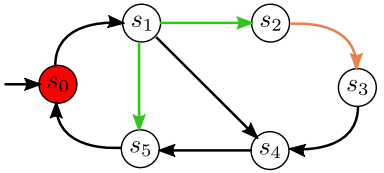
$(P_1 = \top, P_2 = \top, P_3 = \top)$



$(P_1 = \top, P_2 = \top, P_3 = \bot)$

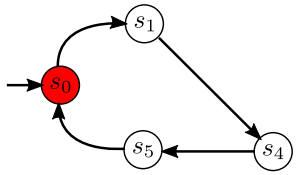


$(P_1 = \top, P_2 = \bot, P_3 = \top)$

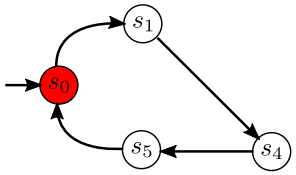


$(P_1 = \top, P_2 = \bot, P_3 = \bot)$

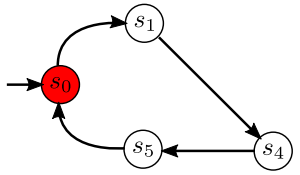
Same Models



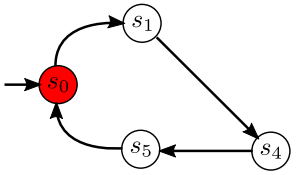
$(P_1 = \bot, P_2 = \top, P_3 = \top)$



$(P_1 = \bot, P_2 = \top, P_3 = \bot)$



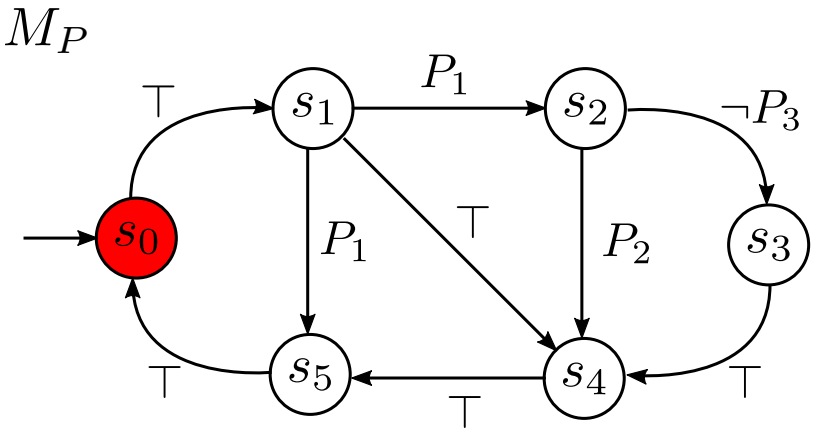
$(P_1 = \bot, P_2 = \bot, P_3 = \top)$



$(P_1 = \bot, P_2 = \bot, P_3 = \bot)$

Parameter Configurations

Goal: Reduce the number of parameter configurations to check

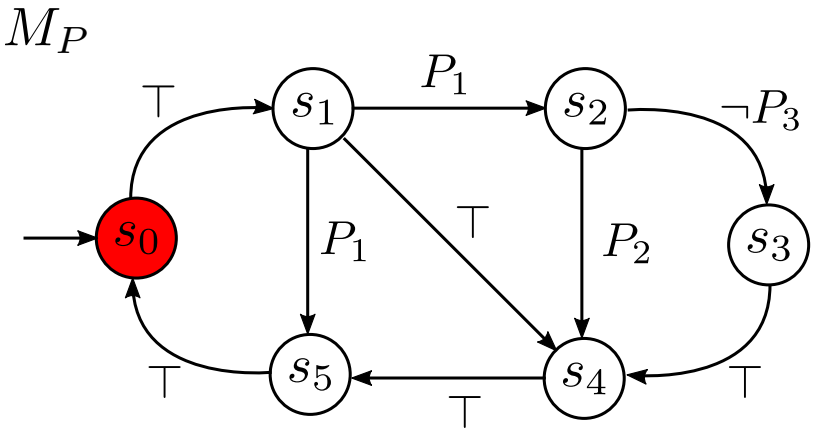


Parameter Configurations

?

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



FINDUP

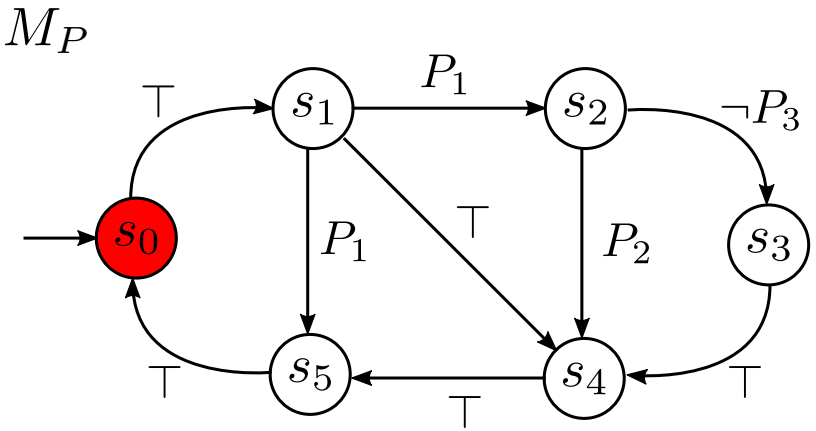
(finds reachable unset parameters)

Parameter Configurations

?

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



FINDUP

(finds reachable unset parameters)

Configuration

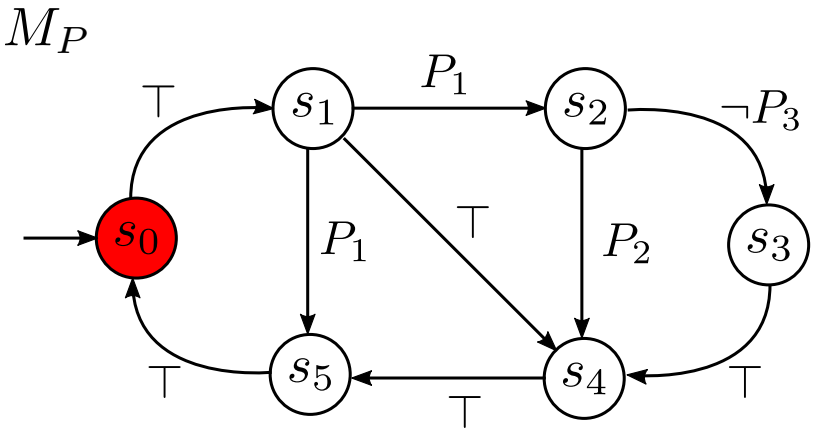
$\hat{c} =$ initially empty

Parameter Configurations

?

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



Parameter Configurations

?

FindUP

(finds reachable unset parameters)

Configuration

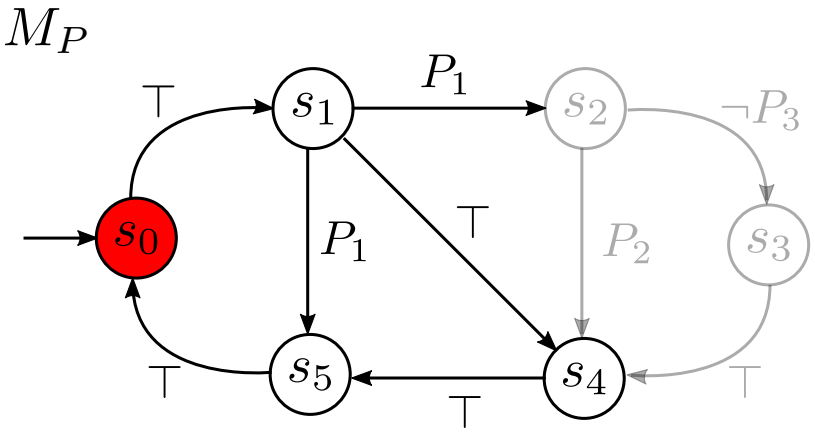
$\hat{c} =$ initially empty

Unset Parameters

$P_u = \text{FindUP}(M_P, \hat{c})$

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



Parameter Configurations

?

FindUP

(finds reachable unset parameters)

Configuration

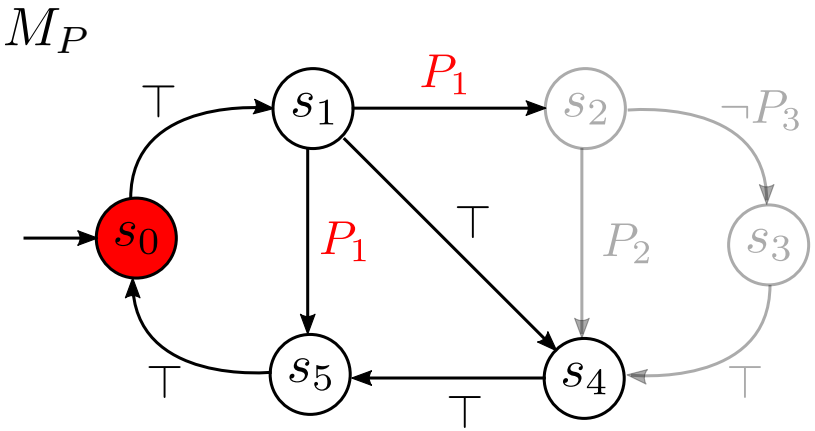
$\hat{c} =$ initially empty

Unset Parameters

$P_u = \text{FindUP}(M_P, \hat{c})$

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



Parameter Configurations

?

FindUP

(finds reachable unset parameters)

Configuration

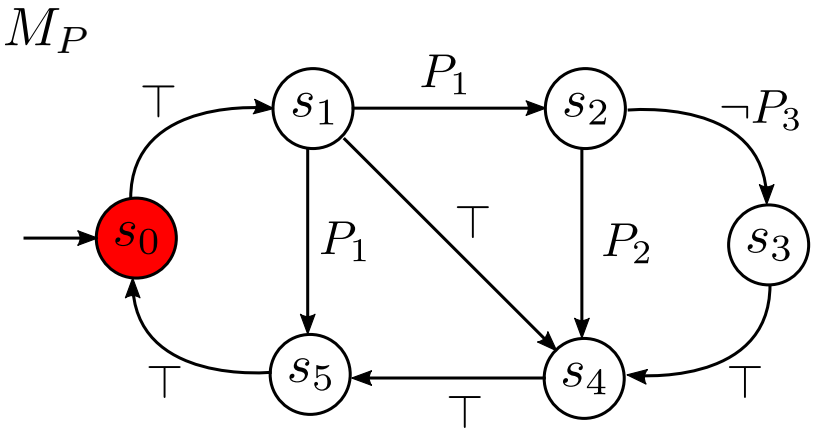
$\hat{c} =$ initially empty

Unset Parameters

$$P_u = \text{FindUP}(M_P, \hat{c})$$
$$= (P_1)$$

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



$P_u = (P_1)$

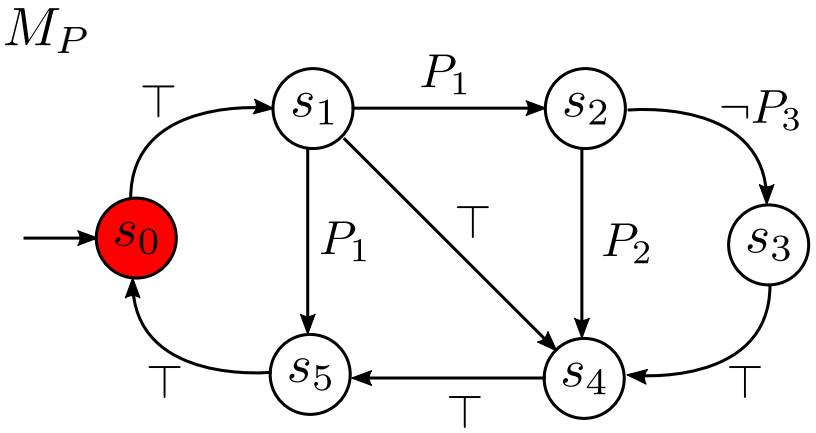
$\hat{c} =$ initially empty

Parameter Configurations

?

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



$P_u = (P_1)$

$\hat{c} =$ initially empty

GENPC

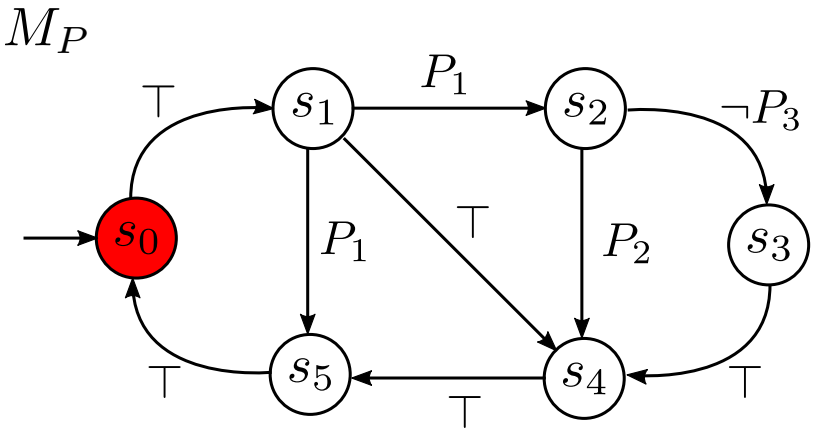
(recursively generate parameter configurations)

Parameter Configurations

?

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



$P_u = (P_1)$

$\hat{c} =$ initially empty

GenPC

(recursively generate parameter configurations)

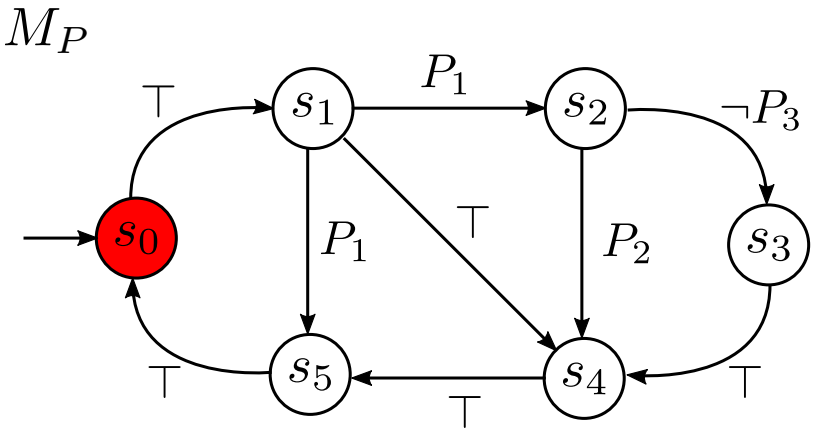
Parameter Configurations

?

Call GenPC$(M_P, P_u, \hat{c})$

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



Parameter Configurations

?

$P_u = (P_1)$

$\hat{c} =$ initially empty

GENPC

(recursively generate parameter configurations)

Call GENPC$(M_P, P_u, \hat{c})$

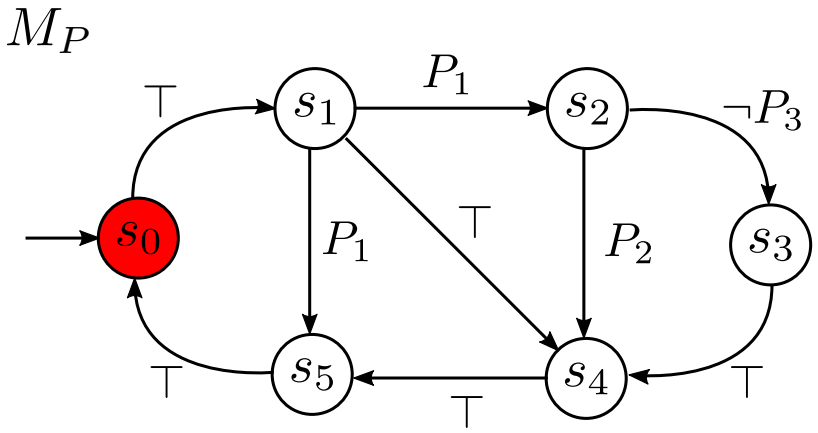
$p =$ dequeue from P_u

$= P_1$

$P_u =$ empty

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



$P_u = (P_1)$

$\hat{c} =$ initially empty

GENPC

(recursively generate parameter configurations)

Parameter Configurations

?

Call GENPC$(M_P, P_u, \hat{c})$

$p =$ dequeue from P_u

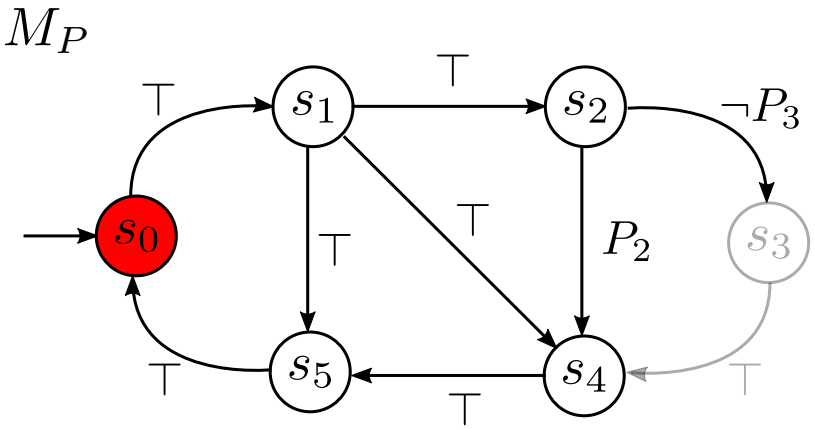
$= P_1$

$P_u =$ empty

$\hat{c} = (P_1 = \top)$ and $\hat{c} = (P_1 = \bot)$

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



$P_u = \text{empty}$

$\hat{c} = (P_1 = \top)$

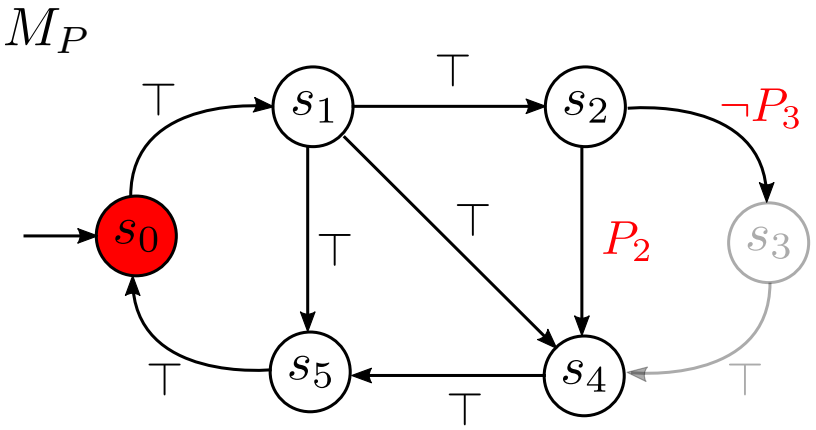
$P_u = \textsc{FindUP}(M_P, \hat{c})$

Parameter Configurations

?

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



$P_u = \text{empty}$

$\hat{c} = (P_1 = \top)$

$P_u = \textsc{FindUP}(M_P, \hat{c})$

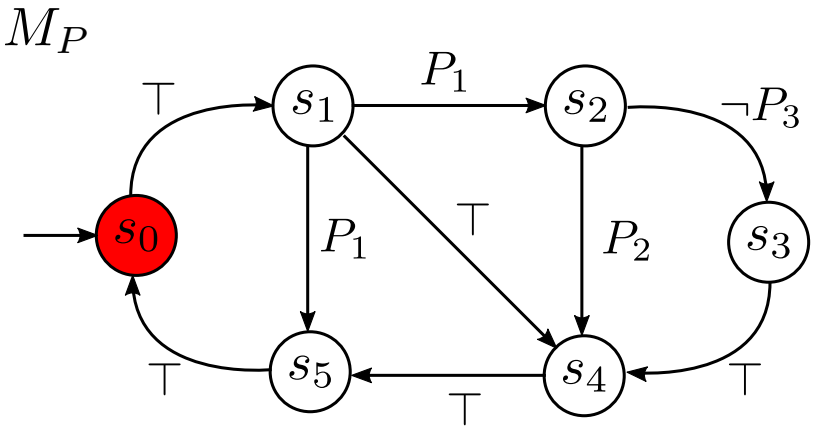
$= (P_2, P_3)$

Parameter Configurations

?

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



$P_u = (P_2, P_3)$

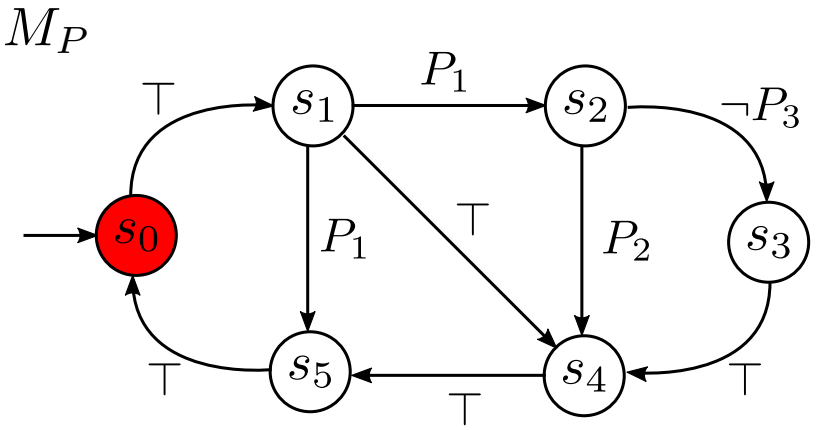
$\hat{c} = (P_1 = \top)$

Parameter Configurations

?

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



$P_u = (P_2, P_3)$

$\hat{c} = (P_1 = \top)$

Call GenPC$(M_P, P_u, \hat{c})$

$p =$ dequeue from P_u

$= P_2$

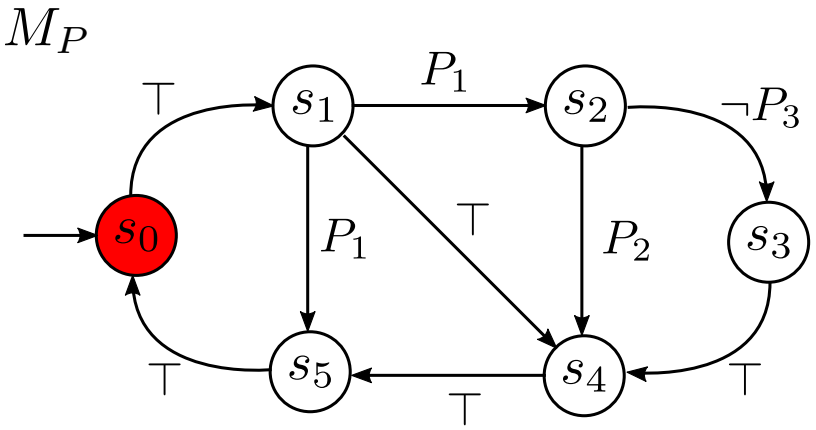
$P_u = (P_3)$

Parameter Configurations

?

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



Parameter Configurations

?

$P_u = (P_2, P_3)$

$\hat{c} = (P_1 = \top)$

Call GenPC$(M_P, P_u, \hat{c})$

p = dequeue from P_u

$= P_2$

$P_u = (P_3)$

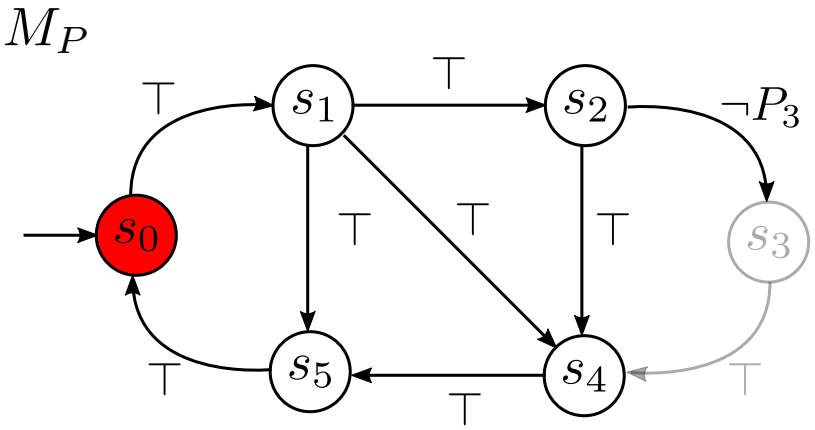
$\hat{c} = (P_1 = \top, P_2 = \top)$

and

$\hat{c} = (P_1 = \top, P_2 = \bot)$

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



$$P_u = (P_3)$$
$$\hat{c} = (P_1 = \top, P_2 = \top)$$

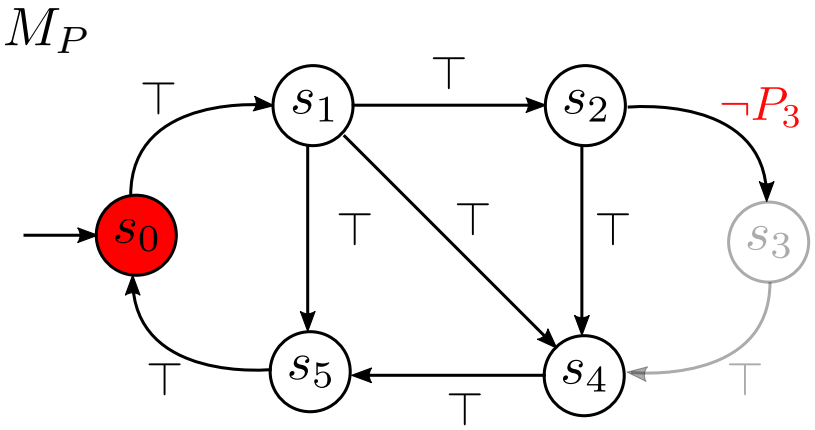
$$P_u = \textsc{FindUP}(M_P, \hat{c})$$

Parameter Configurations

?

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



$$P_u = (P_3)$$
$$\hat{c} = (P_1 = \top, P_2 = \top)$$

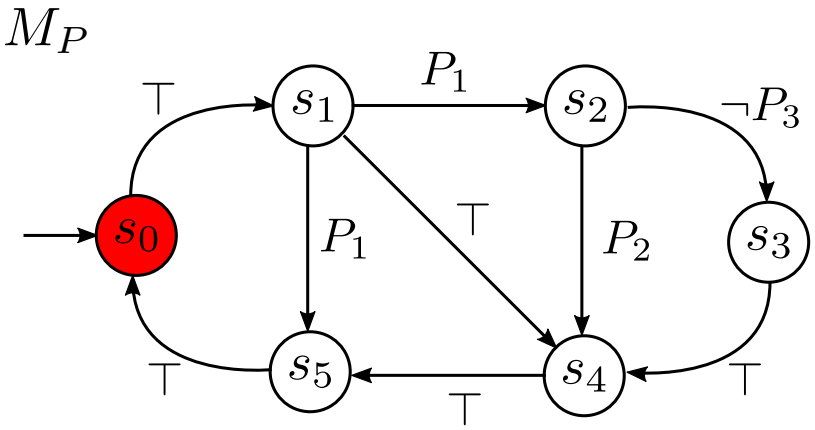
$$P_u = \textsc{FindUP}(M_P, \hat{c})$$
$$= (P_3)$$

Parameter Configurations

?

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



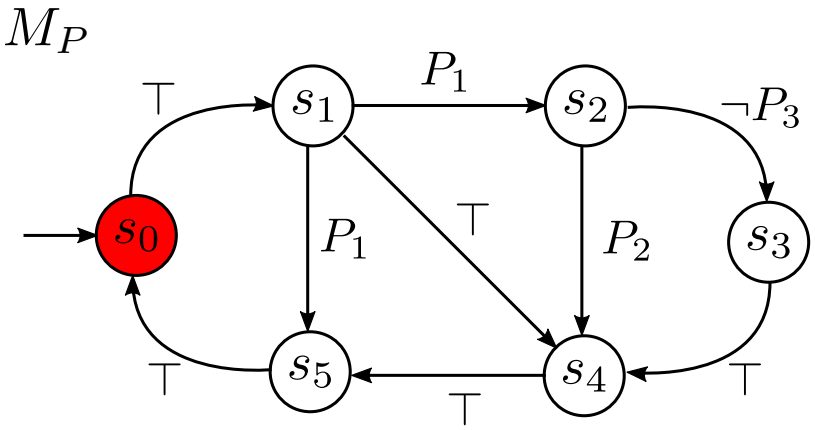
$$P_u = (P_3)$$
$$\hat{c} = (P_1 = \top, P_2 = \top)$$

Parameter Configurations

?

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



$P_u = (P_3)$

$\hat{c} = (P_1 = \top, P_2 = \top)$

Call GenPC$(M_P, P_u, \hat{c})$

p = dequeue from P_u
$= P_3$

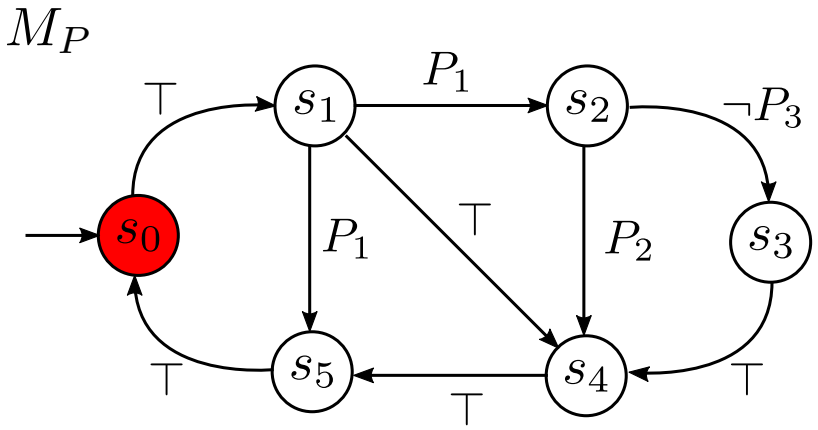
P_u = empty

Parameter Configurations

?

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



Parameter Configurations

?

$P_u = (P_3)$

$\hat{c} = (P_1 = \top, P_2 = \top)$

Call GenPC$(M_P, P_u, \hat{c})$

p = dequeue from P_u
$= P_3$

P_u = empty

$\hat{c} = (P_1 = \top, P_2 = \top, P_3 = \top)$

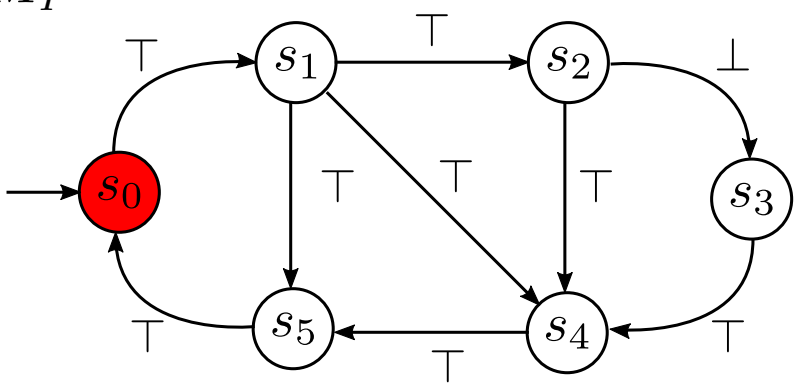
and

$\hat{c} = (P_1 = \top, P_2 = \top, P_3 = \bot)$

Parameter Configurations

Goal: Reduce the number of parameter configurations to check

M_P



$P_u = \text{empty}$

$\hat{c} = (P_1 = \top, P_2 = \top, P_3 = \top)$

$P_u = \textsc{FindUP}(M_P, \hat{c})$

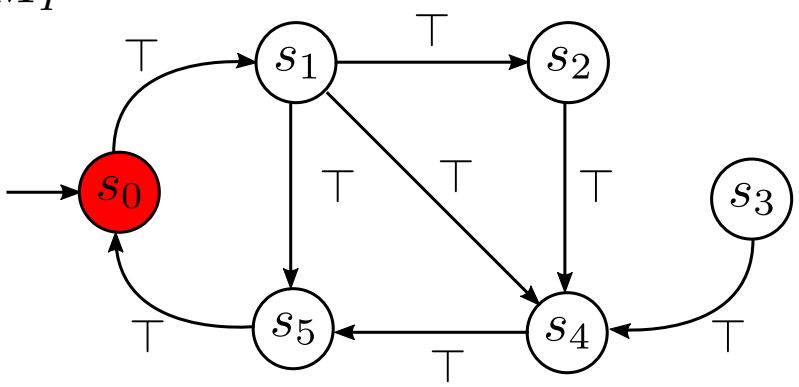
Parameter Configurations

?

Parameter Configurations

Goal: Reduce the number of parameter configurations to check

M_P



$P_u = \text{empty}$

$\hat{c} = (P_1 = \top, P_2 = \top, P_3 = \top)$

$P_u = \textsc{FindUP}(M_P, \hat{c})$

$= \text{empty}$

Parameter Configurations

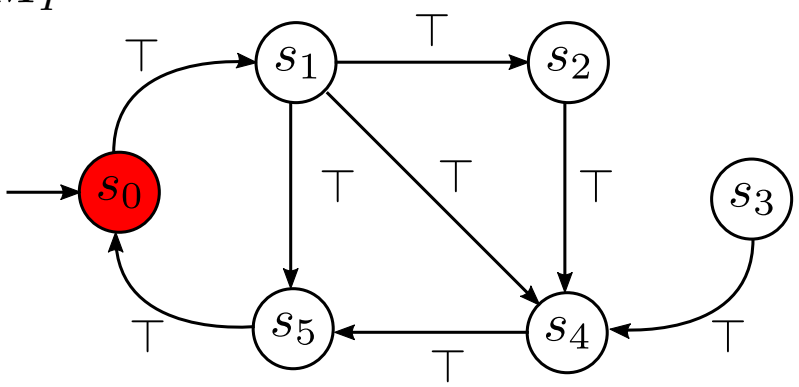
$(P_1 = \top, P_2 = \top, P_3 = \top)$

?

Parameter Configurations

Goal: Reduce the number of parameter configurations to check

M_P



$P_u = \text{empty}$

$\hat{c} = (P_1 = \top, P_2 = \top, P_3 = \top)$

$P_u = \textsc{FindUP}(M_P, \hat{c})$

$= \text{empty}$

Parameter Configurations

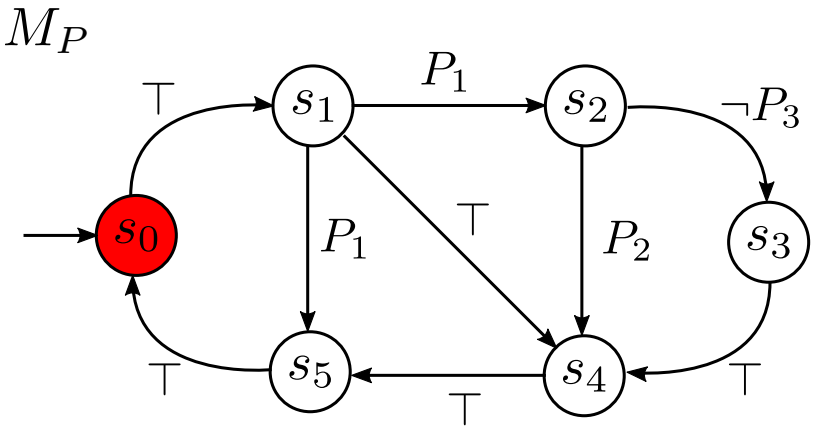
$(P_1 = \top, P_2 = \top, P_3 = \top)$

?

Configuration Found!

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



$P_u = \text{empty}$

$\hat{c} = (P_1 = \top, P_2 = \top, P_3 = \bot)$

$P_u = \textsc{FindUP}(M_P, \hat{c})$

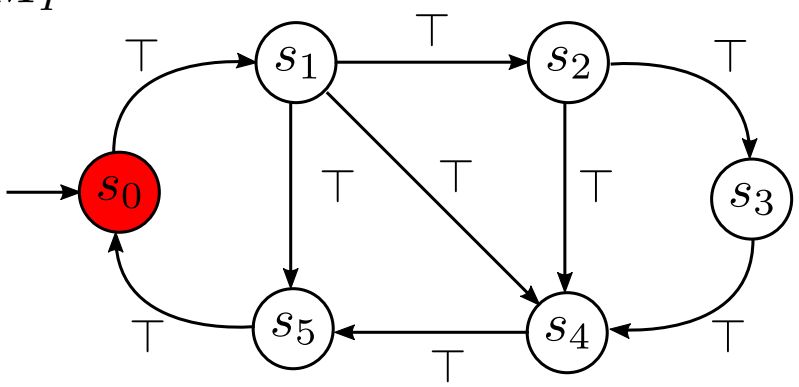
Parameter Configurations

$(P_1 = \top, P_2 = \top, P_3 = \top)$

Parameter Configurations

Goal: Reduce the number of parameter configurations to check

M_P



$P_u = \text{empty}$

$\hat{c} = (P_1 = \top, P_2 = \top, P_3 = \bot)$

$P_u = \textsc{FindUP}(M_P, \hat{c})$

$= \text{empty}$

Configuration Found!

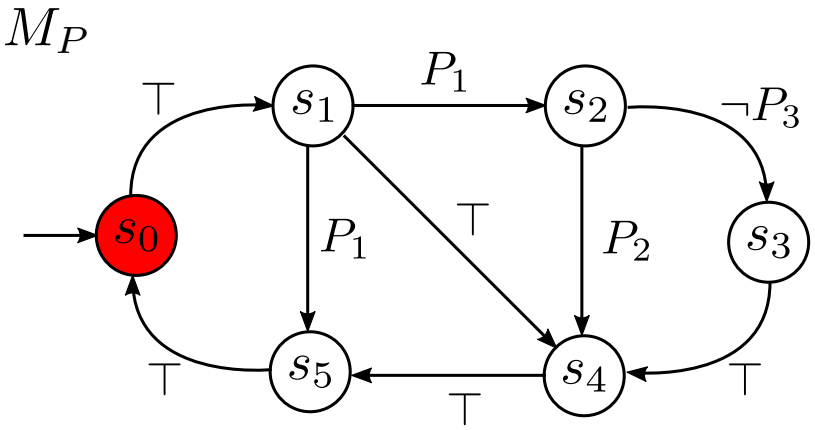
Parameter Configurations

$(P_1 = \top, P_2 = \top, P_3 = \top)$

$(P_1 = \top, P_2 = \top, P_3 = \bot)$

Parameter Configurations

Goal: Reduce the number of parameter configurations to check



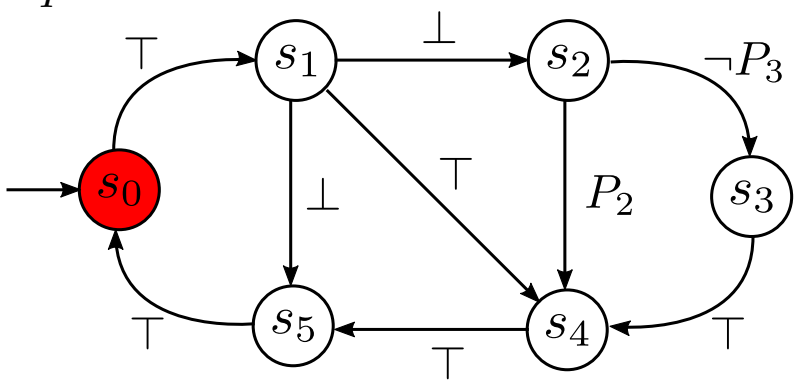
Parameter Configurations

$(P_1 = \top, P_2 = \top, P_3 = \top)$
$(P_1 = \top, P_2 = \top, P_3 = \bot)$
$(P_1 = \top, P_2 = \bot, P_3 = \top)$
$(P_1 = \top, P_2 = \bot, P_3 = \bot)$

Parameter Configurations

Goal: Reduce the number of parameter configurations to check

M_P



$P_u = \text{empty}$

$\hat{c} = (P_1 = \bot)$

$P_u = \textsc{FindUP}(M_P, \hat{c})$

Parameter Configurations

$(P_1 = \top, P_2 = \top, P_3 = \top)$

$(P_1 = \top, P_2 = \top, P_3 = \bot)$

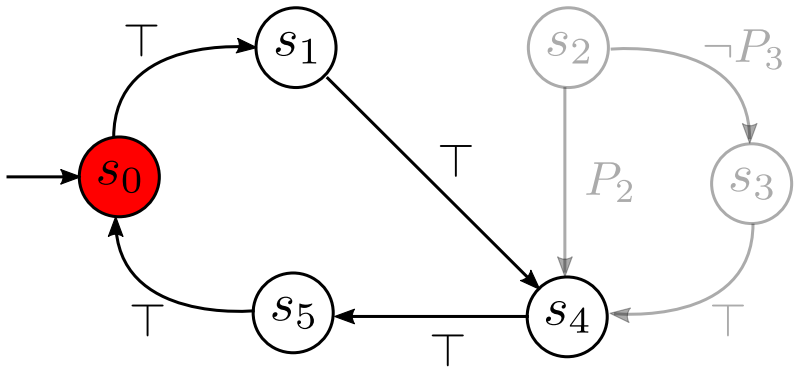
$(P_1 = \top, P_2 = \bot, P_3 = \top)$

$(P_1 = \top, P_2 = \bot, P_3 = \bot)$

Parameter Configurations

Goal: Reduce the number of parameter configurations to check

M_P



$P_u = \text{empty}$

$\hat{c} = (P_1 = \bot)$

$P_u = \textsc{FindUP}(M_P, \hat{c})$

$= \text{empty}$

Parameter Configurations

$(P_1 = \top, P_2 = \top, P_3 = \top)$

$(P_1 = \top, P_2 = \top, P_3 = \bot)$

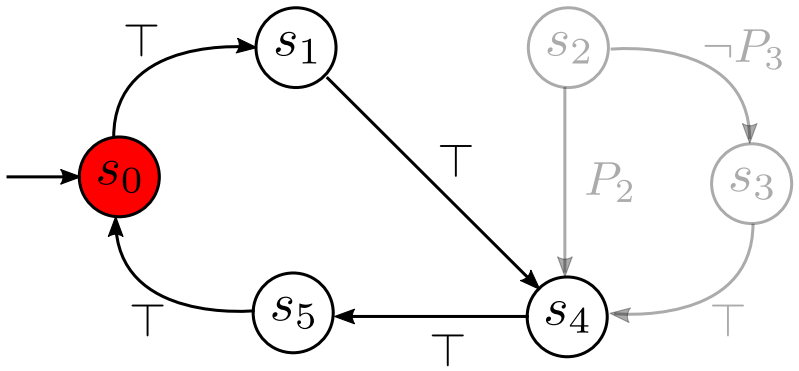
$(P_1 = \top, P_2 = \bot, P_3 = \top)$

$(P_1 = \top, P_2 = \bot, P_3 = \bot)$

Parameter Configurations

Goal: Reduce the number of parameter configurations to check

M_P



$P_u = \text{empty}$

$\hat{c} = (P_1 = \bot)$

$P_u = \textsc{FindUP}(M_P, \hat{c})$

$= \text{empty}$

Parameter Configurations

$(P_1 = \top, P_2 = \top, P_3 = \top)$

$(P_1 = \top, P_2 = \top, P_3 = \bot)$

$(P_1 = \top, P_2 = \bot, P_3 = \top)$

$(P_1 = \top, P_2 = \bot, P_3 = \bot)$

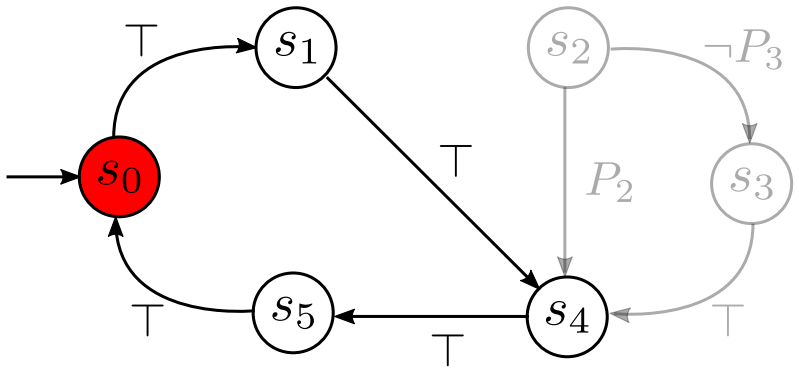
$(P_1 = \bot)$

Configuration Found!

Parameter Configurations

Goal: Reduce the number of parameter configurations to check

M_P



$P_u = \text{empty}$

$\hat{c} = (P_1 = \bot)$

$P_u = \textsc{FindUP}(M_P, \hat{c})$

$= \text{empty}$

Parameter Configurations

$(P_1 = \top, P_2 = \top, P_3 = \top)$

$(P_1 = \top, P_2 = \top, P_3 = \bot)$

$(P_1 = \top, P_2 = \bot, P_3 = \top)$

$(P_1 = \top, P_2 = \bot, P_3 = \bot)$

$(P_1 = \bot)$

Configuration Found!

Minimal Guarantee

LTL Properties

LTL Properties

Motivating Example

Given: Model M over variables p and q

$$\varphi_1 = \Box q$$
$$\varphi_2 = \Box(p \wedge q)$$

LTL Properties

Motivating Example

Given: Model M over variables p and q

$$\varphi_1 = \Box q$$
$$\varphi_2 = \Box(p \wedge q)$$

$$M \models \varphi_2 \text{ then } M \models \varphi_1$$

LTL Properties

Motivating Example

Given: Model M over variables p and q

$$\varphi_1 = \Box q$$
$$\varphi_2 = \Box(p \wedge q)$$

$$M \models \varphi_2 \text{ then } M \models \varphi_1 \; (M \not\models \varphi_1 \text{ then } M \not\models \varphi_2)$$

LTL Properties

Motivating Example

Given: Model M over variables p and q

$$\varphi_1 = \Box q$$
$$\varphi_2 = \Box(p \wedge q)$$

$M \models \varphi_2$ then $M \models \varphi_1$ ($M \not\models \varphi_1$ then $M \not\models \varphi_2$)

φ_1 and φ_2 are dependent

LTL Properties

Motivating Example

Given: Model M over variables p and q

$$\varphi_1 = \Box q$$
$$\varphi_2 = \Box(p \wedge q)$$

$M \models \varphi_2$ then $M \models \varphi_1$ ($M \not\models \varphi_1$ then $M \not\models \varphi_2$)

φ_1 and φ_2 are dependent

$$\varphi_3 = \Box(p \vee q)$$

LTL Properties

Motivating Example

Given: Model M over variables p and q

$\varphi_1 = \Box q$

$\varphi_2 = \Box(p \wedge q)$

$M \models \varphi_2$ then $M \models \varphi_1$ ($M \not\models \varphi_1$ then $M \not\models \varphi_2$)

φ_1 and φ_2 are dependent

$\varphi_3 = \Box(p \vee q)$

$M \models \varphi_2$ then $M \models \varphi_3$

LTL Properties

Motivating Example

Given: Model M over variables p and q

$\varphi_1 = \Box q$

$\varphi_2 = \Box(p \wedge q)$

$M \models \varphi_2$ then $M \models \varphi_1$ ($M \not\models \varphi_1$ then $M \not\models \varphi_2$)

φ_1 and φ_2 are dependent

$\varphi_3 = \Box(p \vee q)$

$M \models \varphi_2$ then $M \models \varphi_3$

φ_2 and φ_3 are dependent

LTL Properties

Motivating Example

Given: Model M over variables p and q

$\varphi_1 = \Box q$

$\varphi_2 = \Box(p \wedge q)$

$M \models \varphi_2$ then $M \models \varphi_1$ ($M \not\models \varphi_1$ then $M \not\models \varphi_2$)

φ_1 and φ_2 are dependent

$\varphi_3 = \Box(p \vee q)$

$M \models \varphi_2$ then $M \models \varphi_3$

φ_2 and φ_3 are dependent

Knowing dependencies *at design-time* is hard!

LTL Properties

Motivating Example

Given: Model M over variables p and q

$\varphi_1 = \Box q$

$\varphi_2 = \Box(p \wedge q)$

$M \models \varphi_2$ then $M \models \varphi_1$ ($M \not\models \varphi_1$ then $M \not\models \varphi_2$)

φ_1 and φ_2 are dependent

$\varphi_3 = \Box(p \vee q)$

$M \models \varphi_2$ then $M \models \varphi_3$

φ_2 and φ_3 are dependent

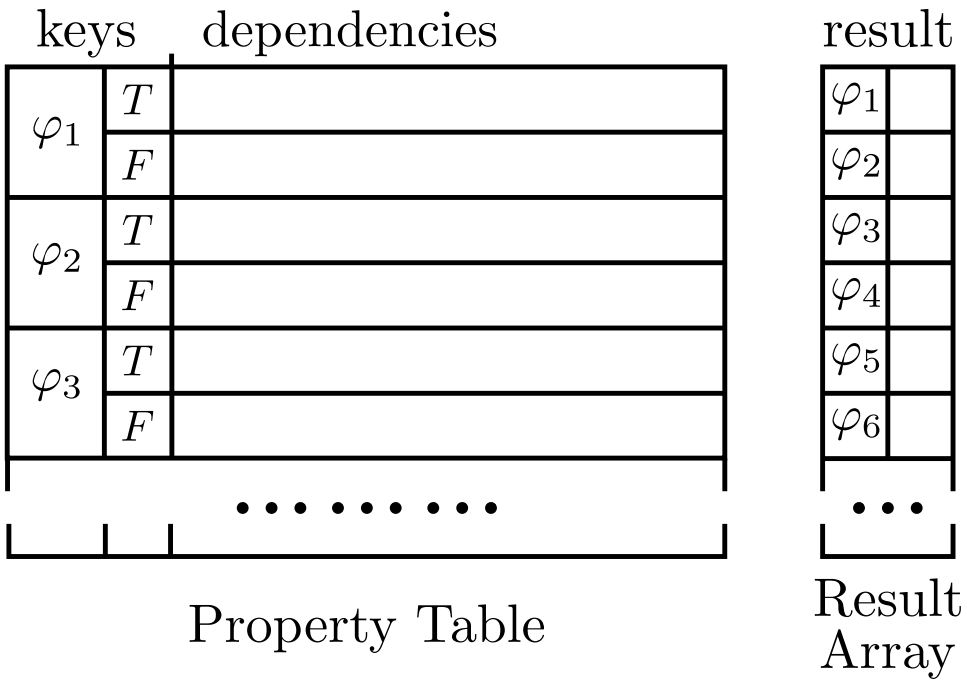
Knowing dependencies *at design-time* is hard!

We use LTL Satisfiability Checking

LTL Properties

Goal: Reduce the number of LTL properties to check

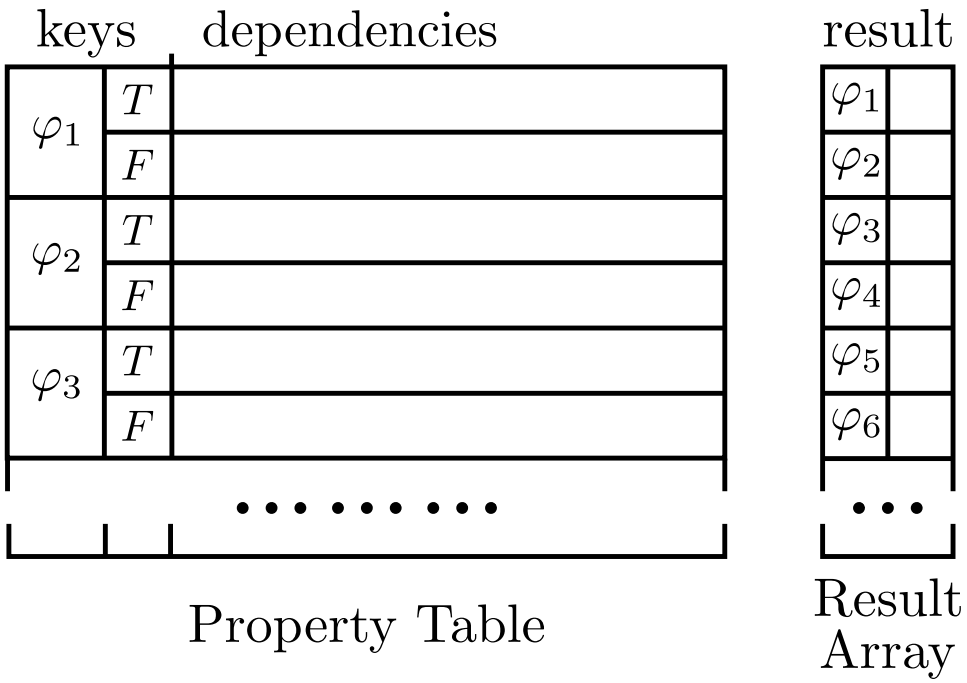
$\mathcal{P} = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6\}$



LTL Properties

Goal: Reduce the number of LTL properties to check

$\mathcal{P} = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6\}$



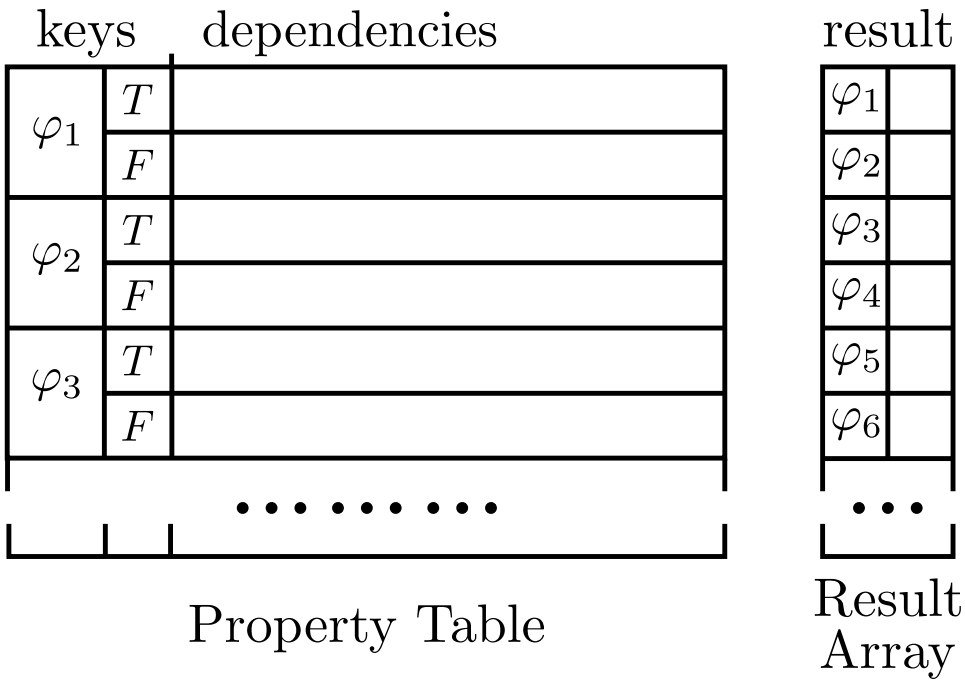
CheckRP

(checks reduced number or properties)

LTL Properties

Goal: Reduce the number of LTL properties to check

$\mathcal{P} = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6\}$



CheckRP

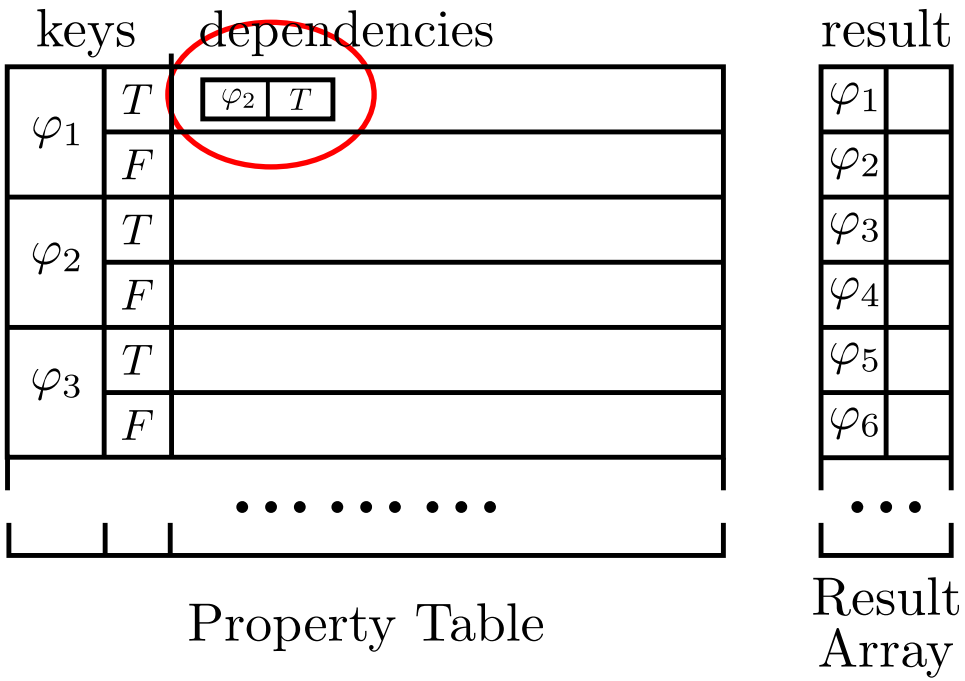
(checks reduced number or properties)

Is $(\varphi_1 \rightarrow \varphi_2)$ satisfiable?

LTL Properties

Goal: Reduce the number of LTL properties to check

$\mathcal{P} = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6\}$



CheckRP

(checks reduced number or properties)

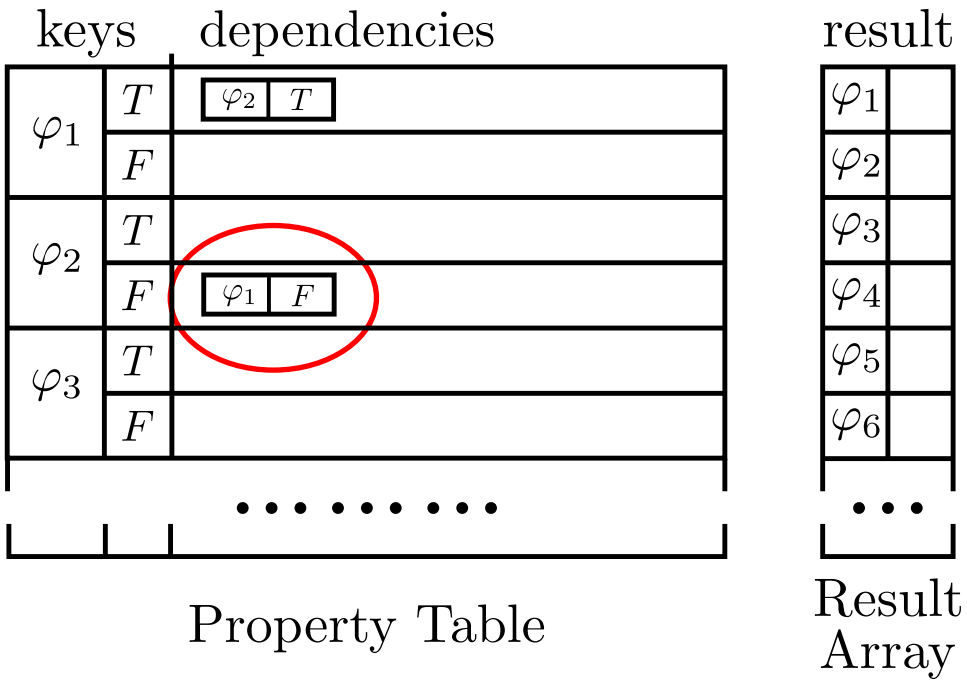
Is $(\varphi_1 \rightarrow \varphi_2)$ satisfiable?

Yes!

LTL Properties

Goal: Reduce the number of LTL properties to check

$\mathcal{P} = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6\}$



CheckRP

(checks reduced number or properties)

Is $(\varphi_1 \rightarrow \varphi_2)$ satisfiable?

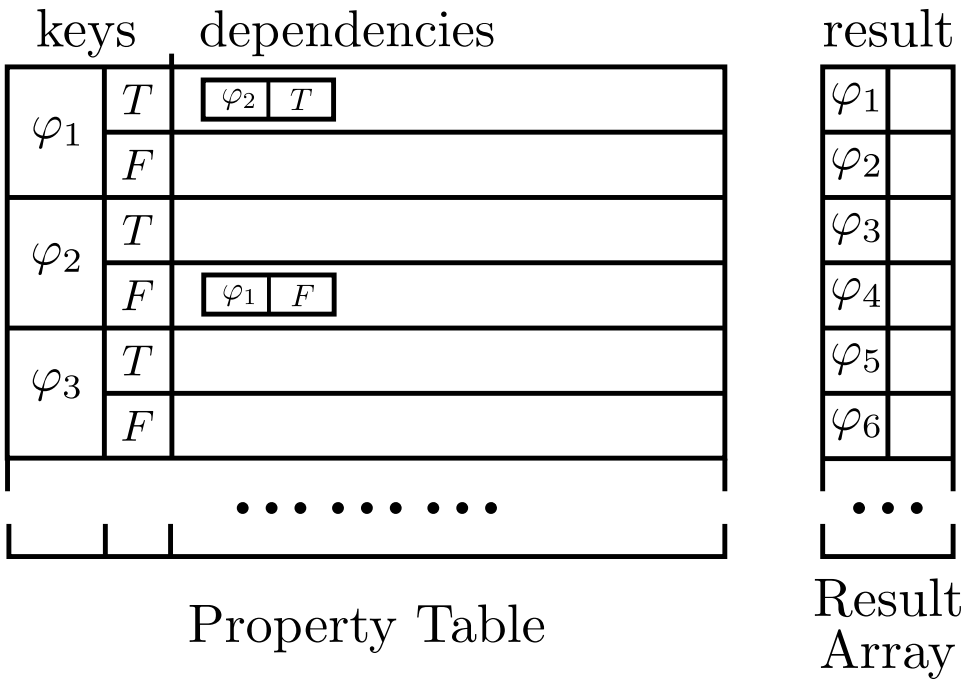
Yes!

$(\neg\varphi_2 \rightarrow \neg\varphi_1)$ also satisfiable.

LTL Properties

Goal: Reduce the number of LTL properties to check

$\mathcal{P} = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6\}$



CheckRP

(checks reduced number or properties)

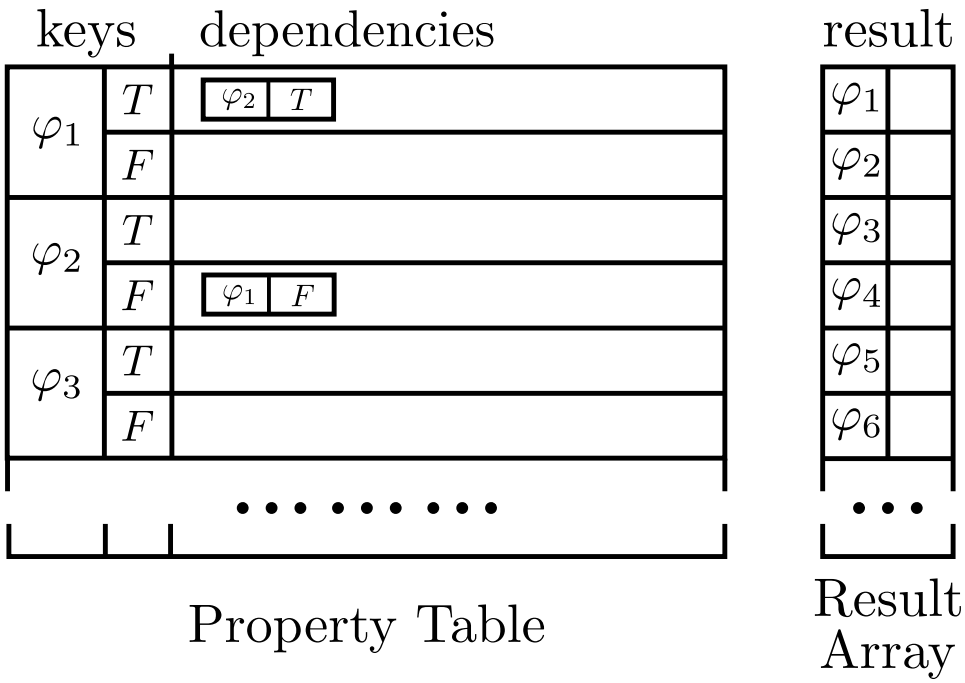
$\varphi_1 \rightarrow \varphi_2$
$\neg\varphi_1 \rightarrow \varphi_2$
$\neg\varphi_1 \rightarrow \neg\varphi_2$
$\varphi_1 \rightarrow \neg\varphi_2$

\+ contra-positives

LTL Properties

Goal: Reduce the number of LTL properties to check

$\mathcal{P} = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6\}$



CheckRP

(checks reduced number or properties)

$\varphi_1 \rightarrow \varphi_2$

$\neg\varphi_1 \rightarrow \varphi_2$

$\neg\varphi_1 \rightarrow \neg\varphi_2$

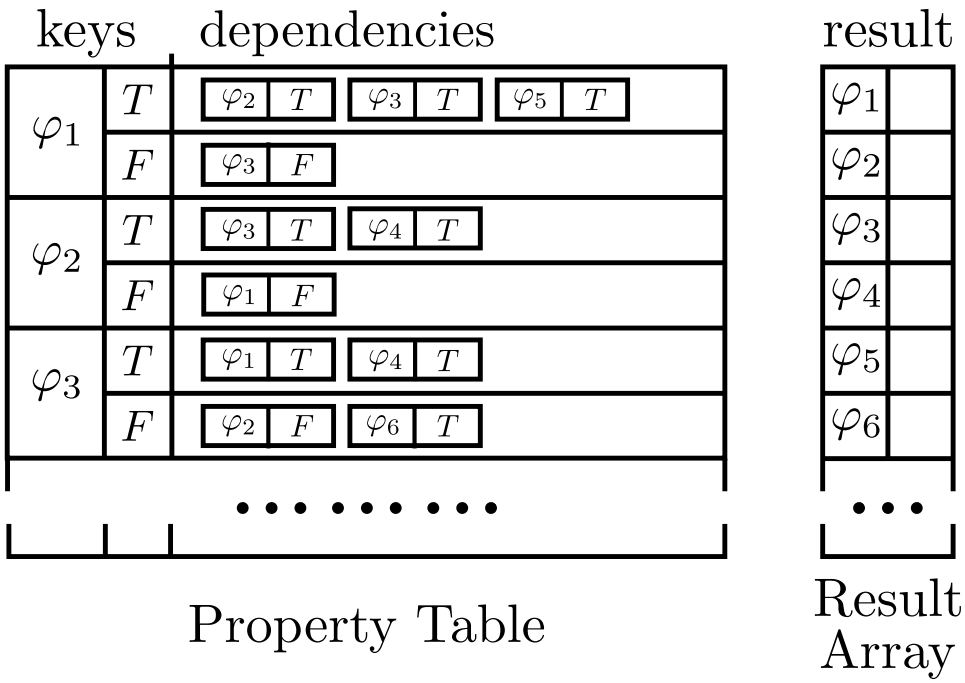
Interesting

$\varphi_1 \rightarrow \neg\varphi_2$

LTL Properties

Goal: Reduce the number of LTL properties to check

$\mathcal{P} = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6\}$



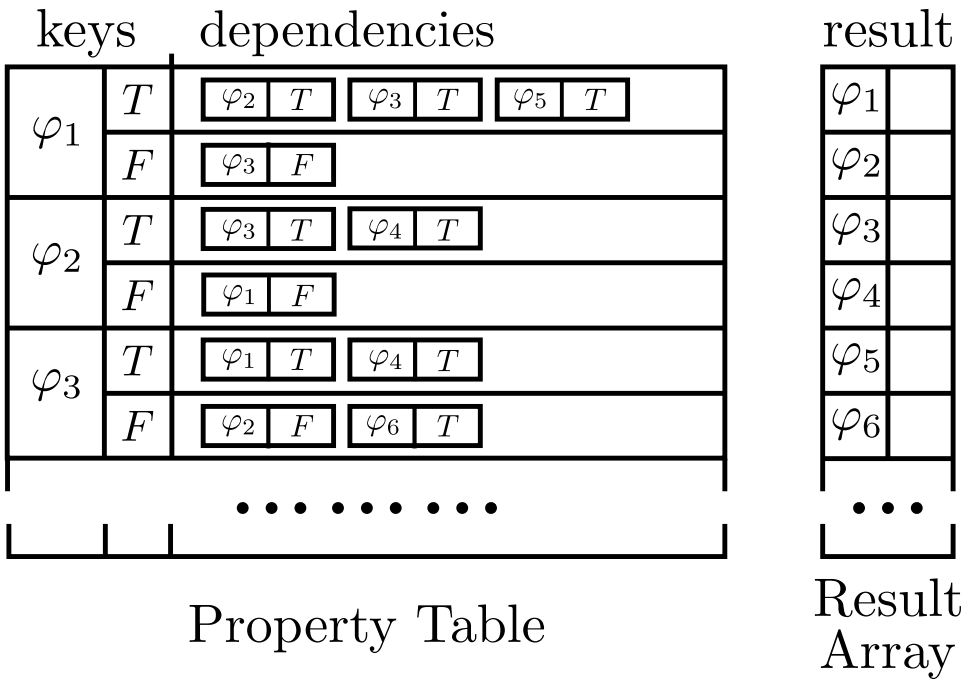
CheckRP

(checks reduced number or properties)

LTL Properties

Goal: Reduce the number of LTL properties to check

$\mathcal{P} = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6\}$



CheckRP

(checks reduced number or properties)

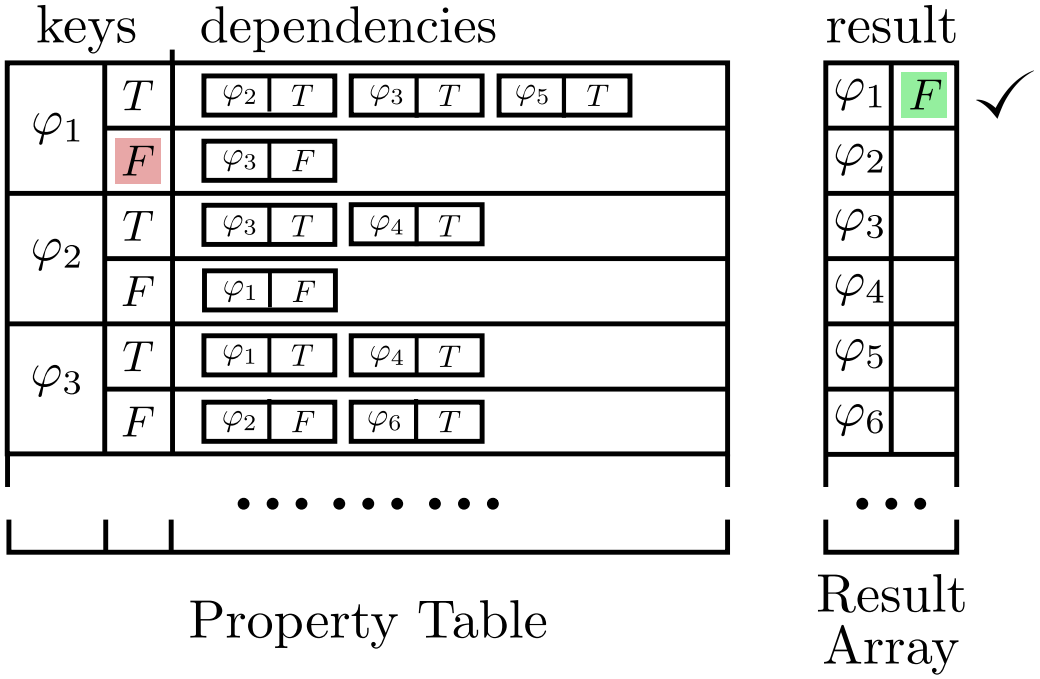
Check if $\forall \varphi \in \mathcal{P}$

$M \models ? \varphi$

LTL Properties

Goal: Reduce the number of LTL properties to check

$\mathcal{P} = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6\}$



CheckRP

(checks reduced number or properties)

Check if $\forall \varphi \in \mathcal{P}$

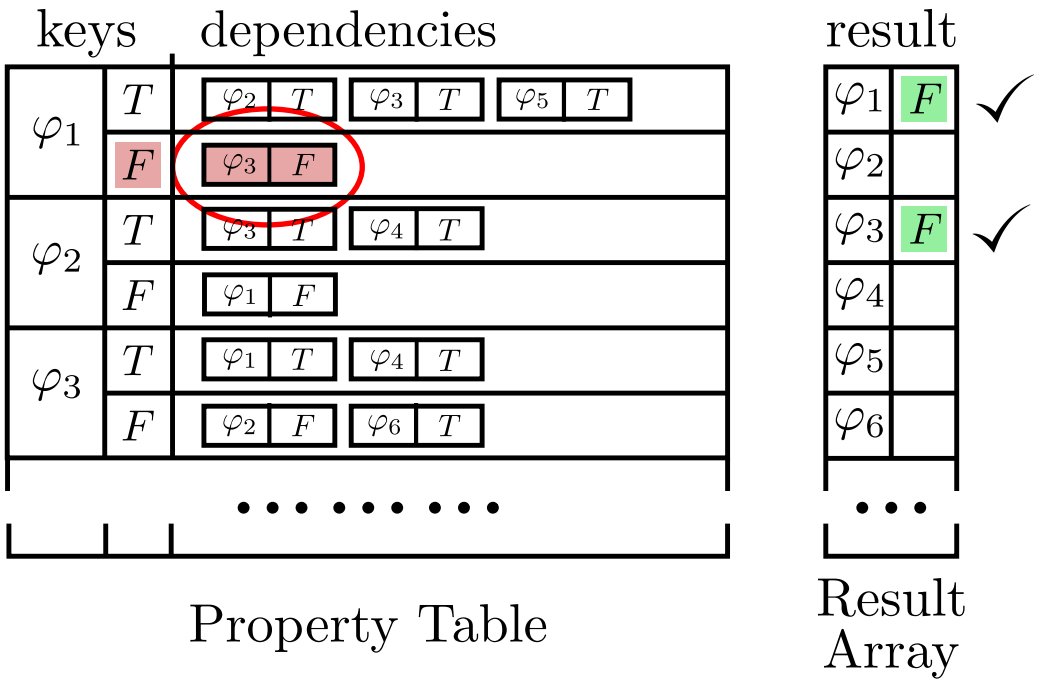
$M \models ? \varphi$

$M \not\models \varphi_1$

LTL Properties

Goal: Reduce the number of LTL properties to check

$\mathcal{P} = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6\}$



CheckRP
(checks reduced number or properties)

Check if $\forall \varphi \in \mathcal{P}$
$M \models ? \varphi$

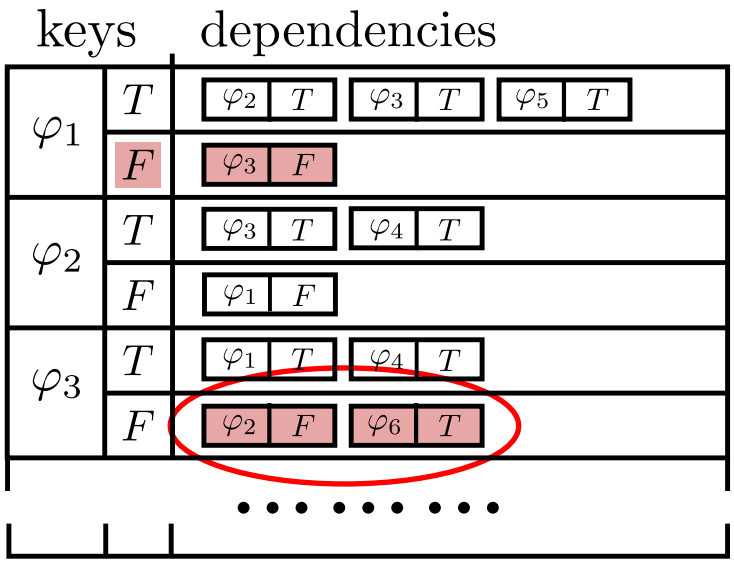
$M \not\models \varphi_1$

$M \not\models \varphi_3$

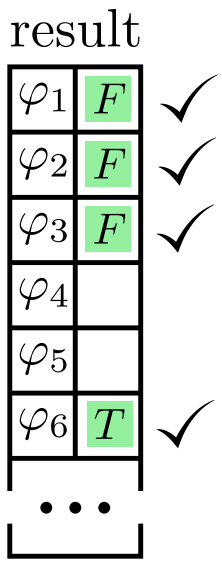
LTL Properties

Goal: Reduce the number of LTL properties to check

$\mathcal{P} = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6\}$



Property Table



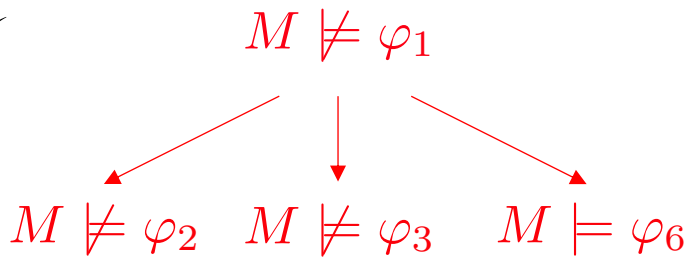
Result Array

CheckRP

(checks reduced number or properties)

Check if $\forall \varphi \in \mathcal{P}$

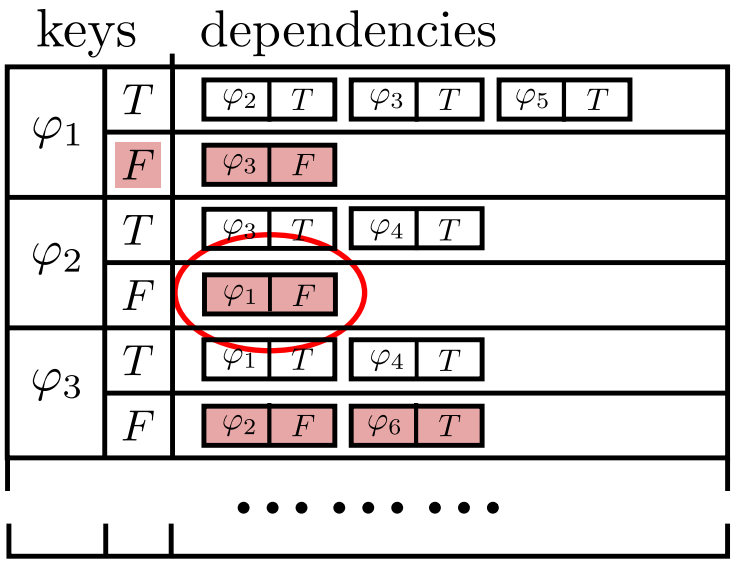
$M \models_? \varphi$



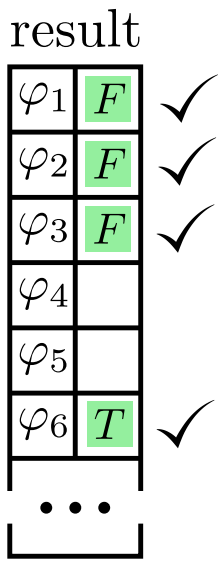
LTL Properties

Goal: Reduce the number of LTL properties to check

$\mathcal{P} = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6\}$



Property Table

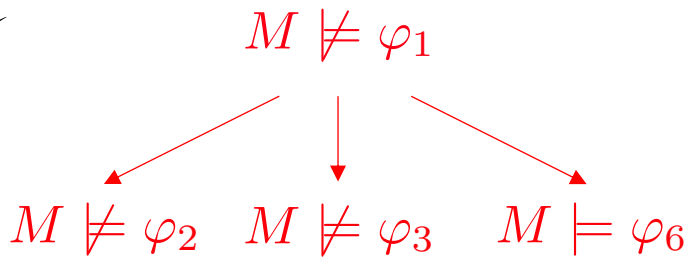


Result Array

CHECKRP

(checks reduced number or properties)

Check if $\forall \varphi \in \mathcal{P}$ $M \models_? \varphi$

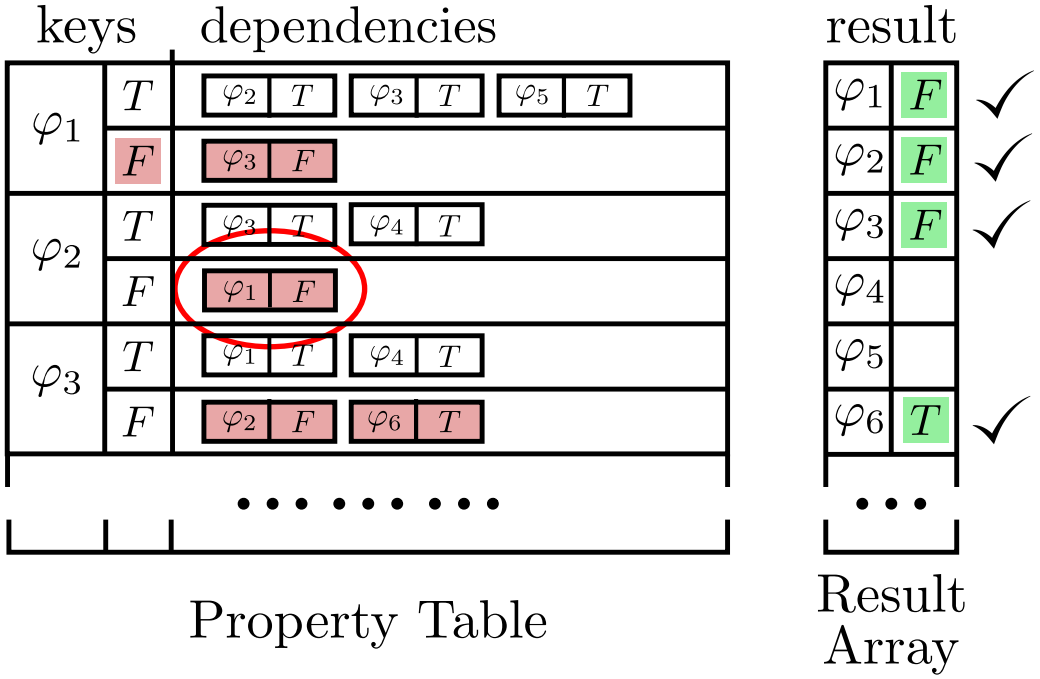


Self Validating!

LTL Properties

Goal: Reduce the number of LTL properties to check

$\mathcal{P} = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6\}$



CHECKRP
(checks reduced number or properties)

Heuristics

H1: Maximum Dependence

H2: Property Grouping

Experiment Setup

Experiment Setup

- D^3 is implemented as a preprocessing script in ~2,000 lines of Python.

Source code available at

`http://temporallogic.org/research/TACAS18`

Experiment Setup

- D^3 is implemented as a preprocessing script in ~2,000 lines of Python.
- We use NUXMV as the interfaced state-of-the-art model checker

Source code available at

`http://temporallogic.org/research/TACAS18`

Experiment Setup

- D^3 is implemented as a preprocessing script in ~2,000 lines of Python.
- We use NUXMV as the interfaced state-of-the-art model checker
- Benchmarks Evaluated
  - NASA NextGen Air Traffic Control (ATC) System (Gario et al., 2016)
  - BOEING AIR 6110 Wheel Braking (WBS) System (Bozzano et al., 2015)

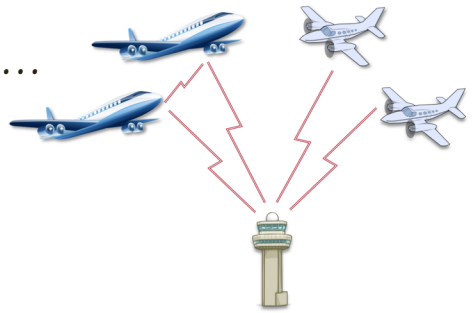
Source code available at

`http://temporallogic.org/research/TACAS18`

NASA ATC Benchmark

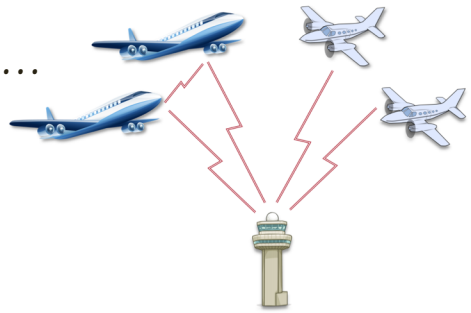
NASA ATC Benchmark

- Large design-space with 1,620 design choices
  different types of aircraft, varying levels of information sharing, …



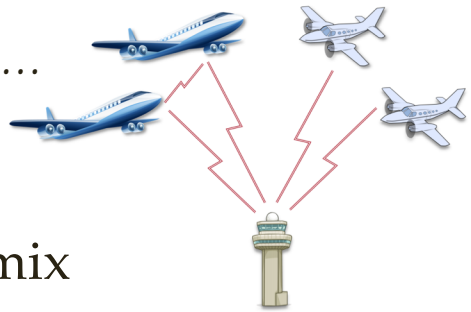
NASA ATC Benchmark

- Large design-space with 1,620 design choices
  different types of aircraft, varying levels of information sharing, …
- 191 LTL properties (liveness + safety) per model



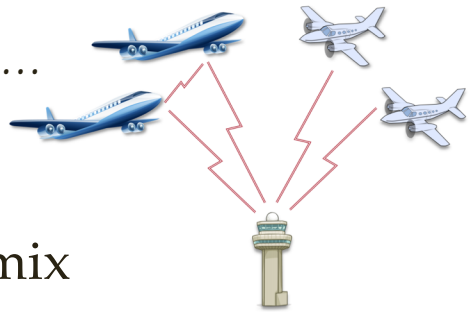
NASA ATC Benchmark

- Large design-space with 1,620 design choices
  different types of aircraft, varying levels of information sharing, …
- 191 LTL properties (liveness + safety) per model
- Verification split into five phases based on property mix
  mix of SAT and UNSAT properties



NASA ATC Benchmark

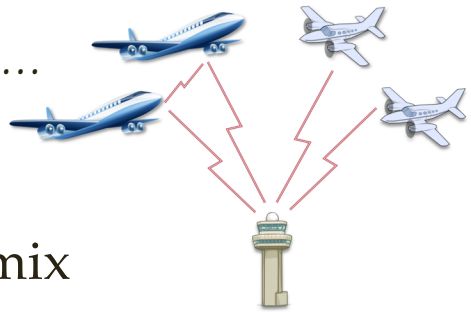
- Large design-space with 1,620 design choices
  different types of aircraft, varying levels of information sharing, …
- 191 LTL properties (liveness + safety) per model
- Verification split into five phases based on property mix
  mix of SAT and UNSAT properties



Individual model checking: >200 hours

NASA ATC Benchmark

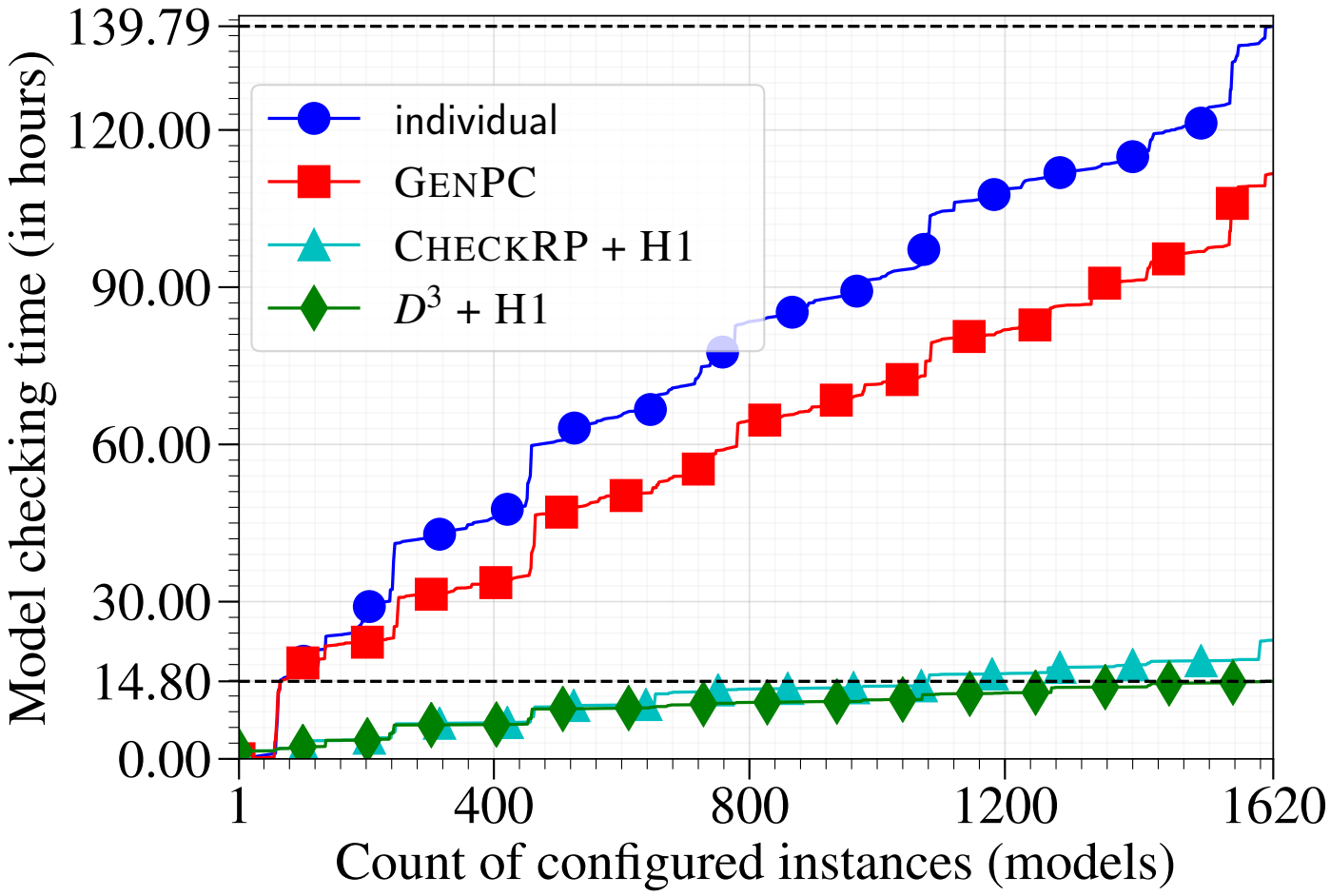
- Large design-space with 1,620 design choices
  different types of aircraft, varying levels of information sharing, …
- 191 LTL properties (liveness + safety) per model
- Verification split into five phases based on property mix
  mix of SAT and UNSAT properties



Individual model checking: >200 hours

Model checking with D³: 50 hours

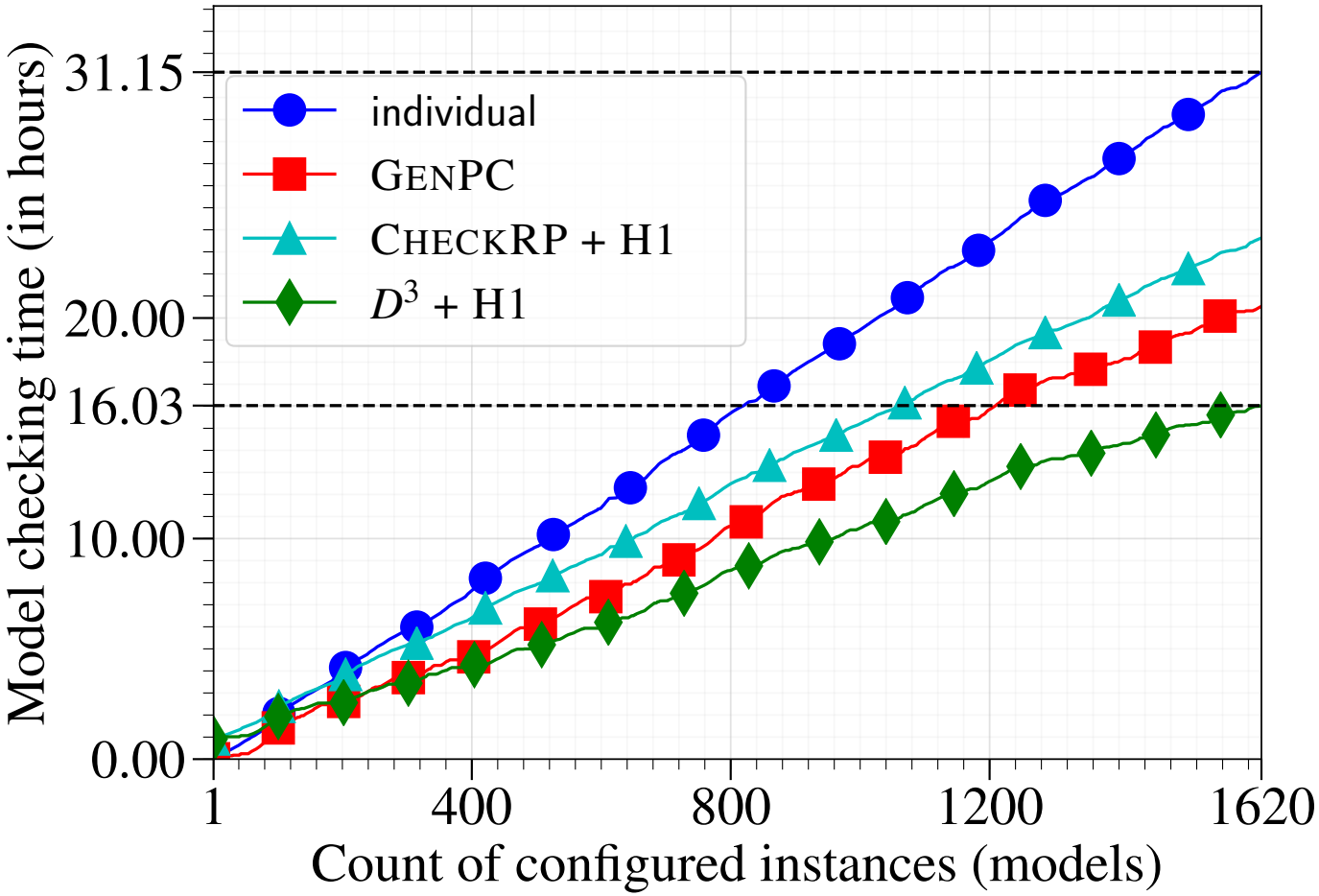
NASA ATC Benchmark



UNSAT properties

9.0x speedup

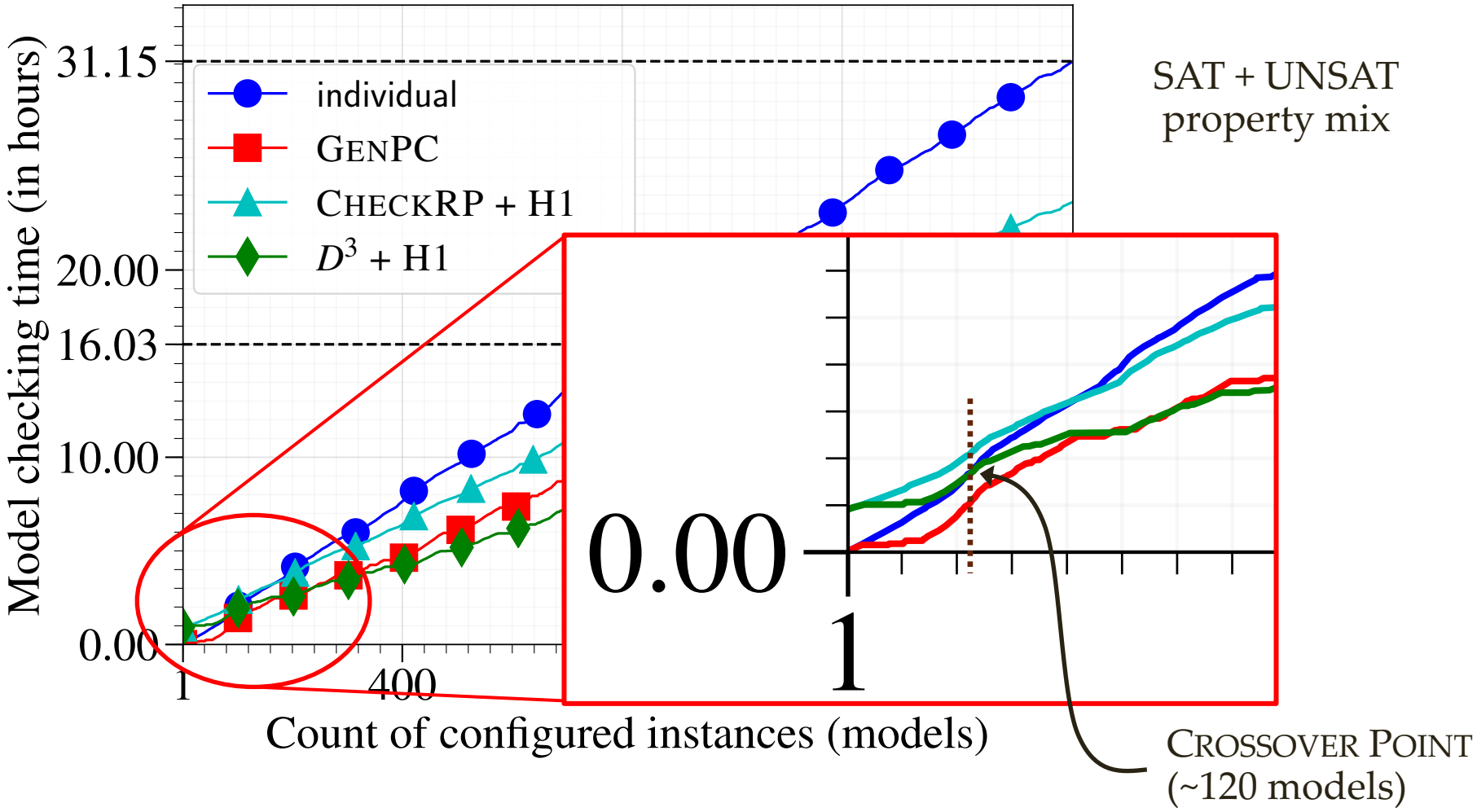
NASA ATC Benchmark



SAT + UNSAT property mix

1.9x speedup

NASA ATC Benchmark



BOEING WBS Benchmark

BOEING WBS Benchmark

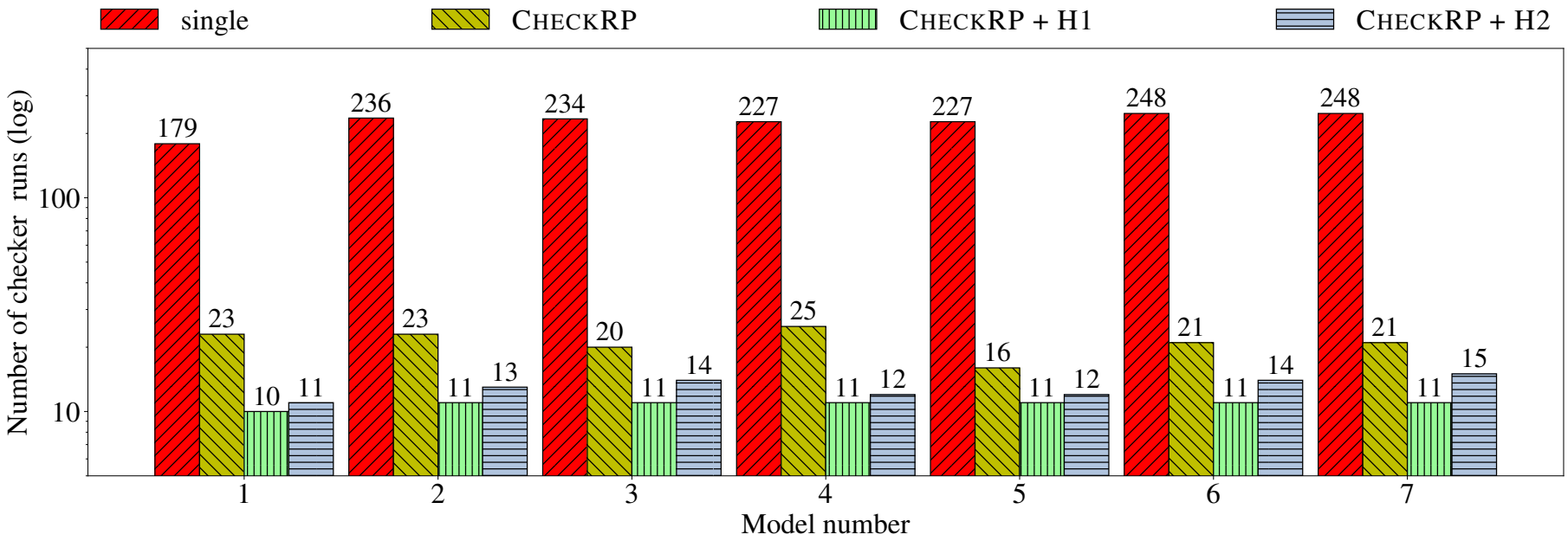
- Set of 7 models; ~200 LTL properties per model

BOEING WBS Benchmark

- Set of 7 models; ~200 LTL properties per model
- Evaluate performance in multi-property verification workflows

BOEING WBS Benchmark

- Set of 7 models; ~200 LTL properties per model
- Evaluate performance in multi-property verification workflows



Discussion

Discussion

- D^3 is an efficient *preprocessing algorithm*; design-space exploration
  - Design-space parameters – reduction to a reachability problem
  - Finds **minimal** configurations, **all** property dependencies

Discussion

- D^3 is an efficient *preprocessing algorithm*; design-space exploration
  - Design-space parameters – reduction to a reachability problem
  - Finds **minimal** configurations, **all** property dependencies
- Works with *off-the-shelf model checkers*; easy integration
  - SMV models extended with parameters as preprocessor directives

Discussion

- D^3 is an efficient *preprocessing algorithm*; design-space exploration
  - Design-space parameters – reduction to a reachability problem
  - Finds **minimal** configurations, **all** property dependencies
- Works with *off-the-shelf model checkers*; easy integration
  - SMV models extended with parameters as preprocessor directives
- *Crossover Point* is a crucial measure when using D^3

Discussion

- D^3 is an efficient *preprocessing algorithm*; design-space exploration
  - Design-space parameters – reduction to a reachability problem
  - Finds **minimal** configurations, **all** property dependencies
- Works with *off-the-shelf model checkers*; easy integration
  - SMV models extended with parameters as preprocessor directives
- *Crossover Point* is a crucial measure when using D^3
- Future work and research questions
  - Can we re-use intermediate results across several models?
  - Extend to other logics and transitions systems, like Markov processes
  - Heuristics for predicting crossover point for model sets

Discussion

- D^3 is an efficient *preprocessing algorithm*; design-space exploration
  - Design-space parameters – reduction to a reachability problem
  - Finds **minimal** configurations, **all** property dependencies
- Works with *off-the-shelf model checkers*; easy integration
  - SMV models extended with parameters as preprocessor directives
- *Crossover Point* is a crucial measure when using D^3
- Future work and research questions
  - Can we re-use intermediate results across several models?
  - Extend to other logics and transitions systems, like Markov processes
  - Heuristics for predicting crossover point for model sets

Thank You!

`http://temporallogic.org/research/TACAS18`