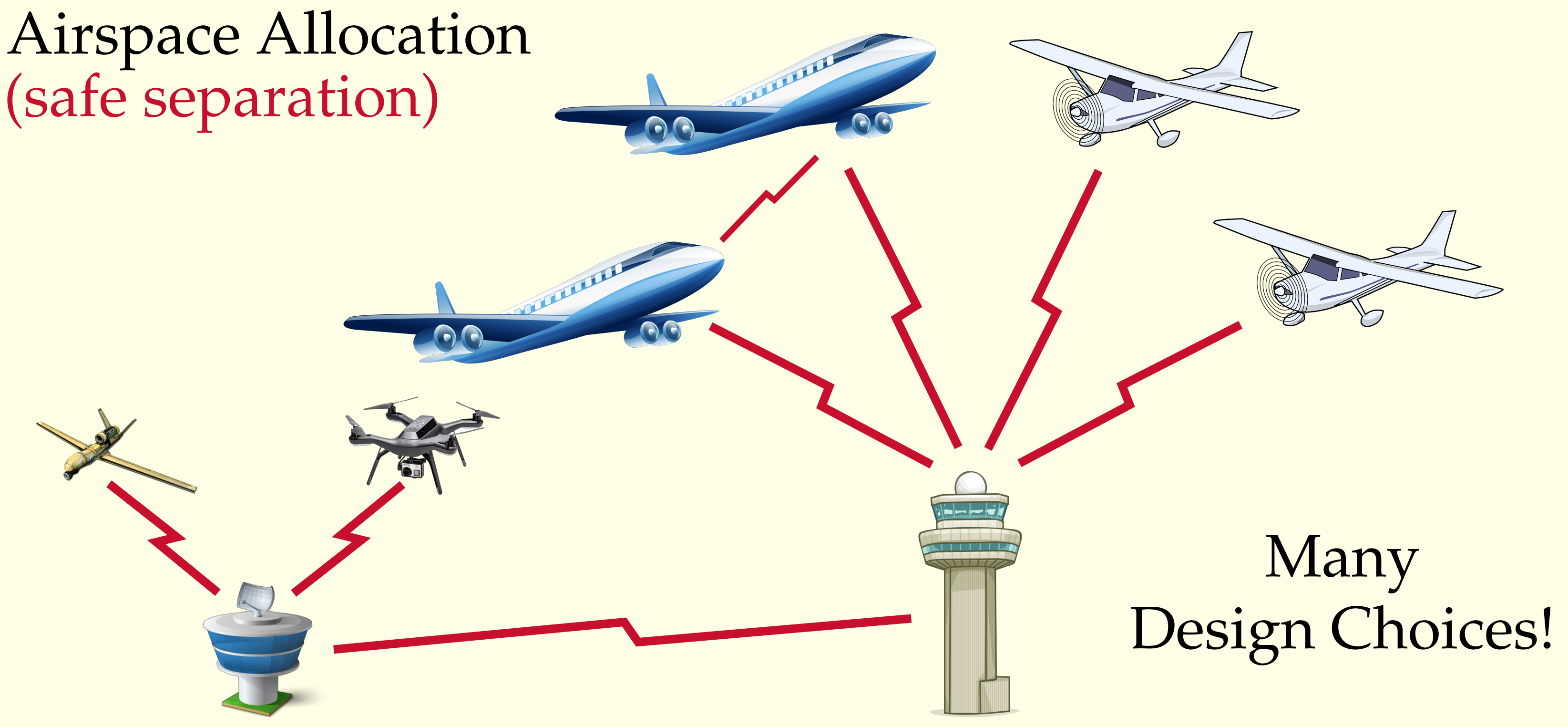
IOWA STATE UNIVERSITY

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

Rohit Dureja and Kristin Yvonne Rozier

Laboratory for Temporal Logic

1. Motivation



Airspace Allocation
(safe separation)

Many
Design Choices!

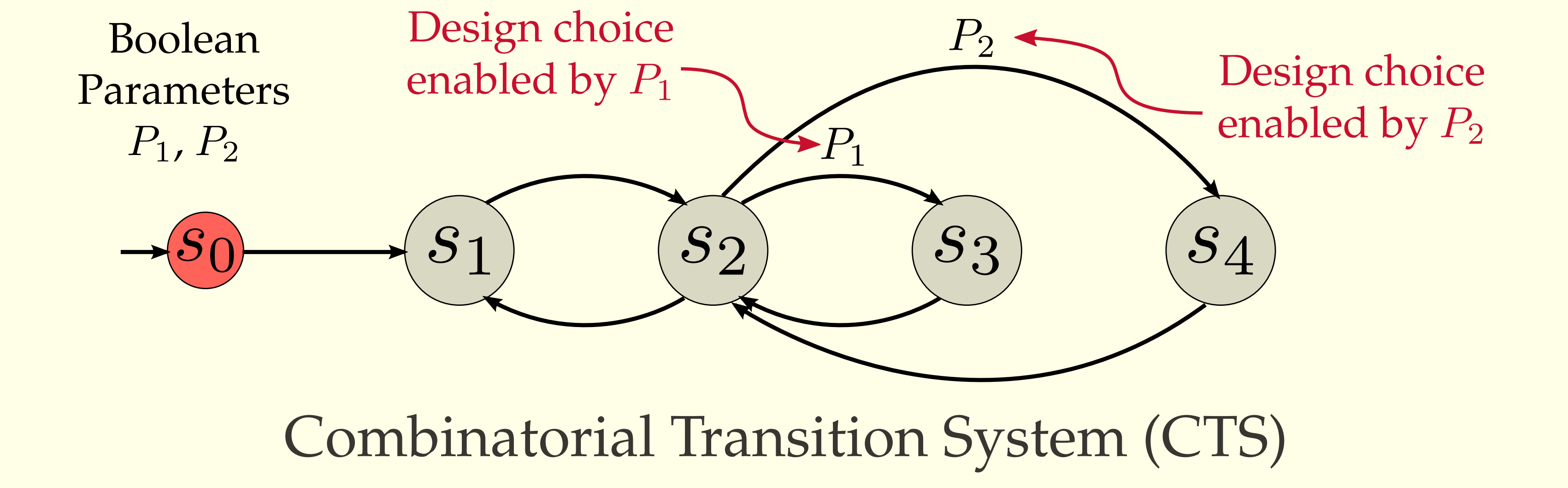
- The design of complex systems often requires analyzing several variants of the system under development for:
  - narrowing in on the final system design, and
  - check capabilities of system with varying features.
- The **design choices** constitute the system's **design space**.

Model checking aids system development via a thorough comparison of all design choices

2. Modeling Design Spaces

Classical Method – Every design choice is a model.
very hard to cross-validate as design-space grows

Scalable Method – Every design choice is a parameter.
efficient, easier to maintain as design evolves



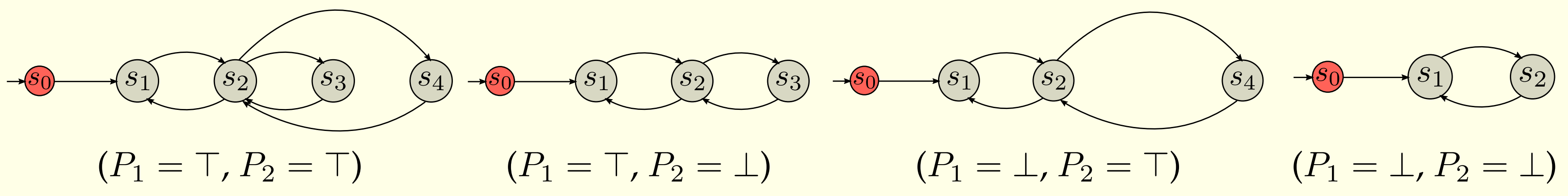
Boolean
Parameters
P_1, P_2

Design choice
enabled by P_1

Design choice
enabled by P_2

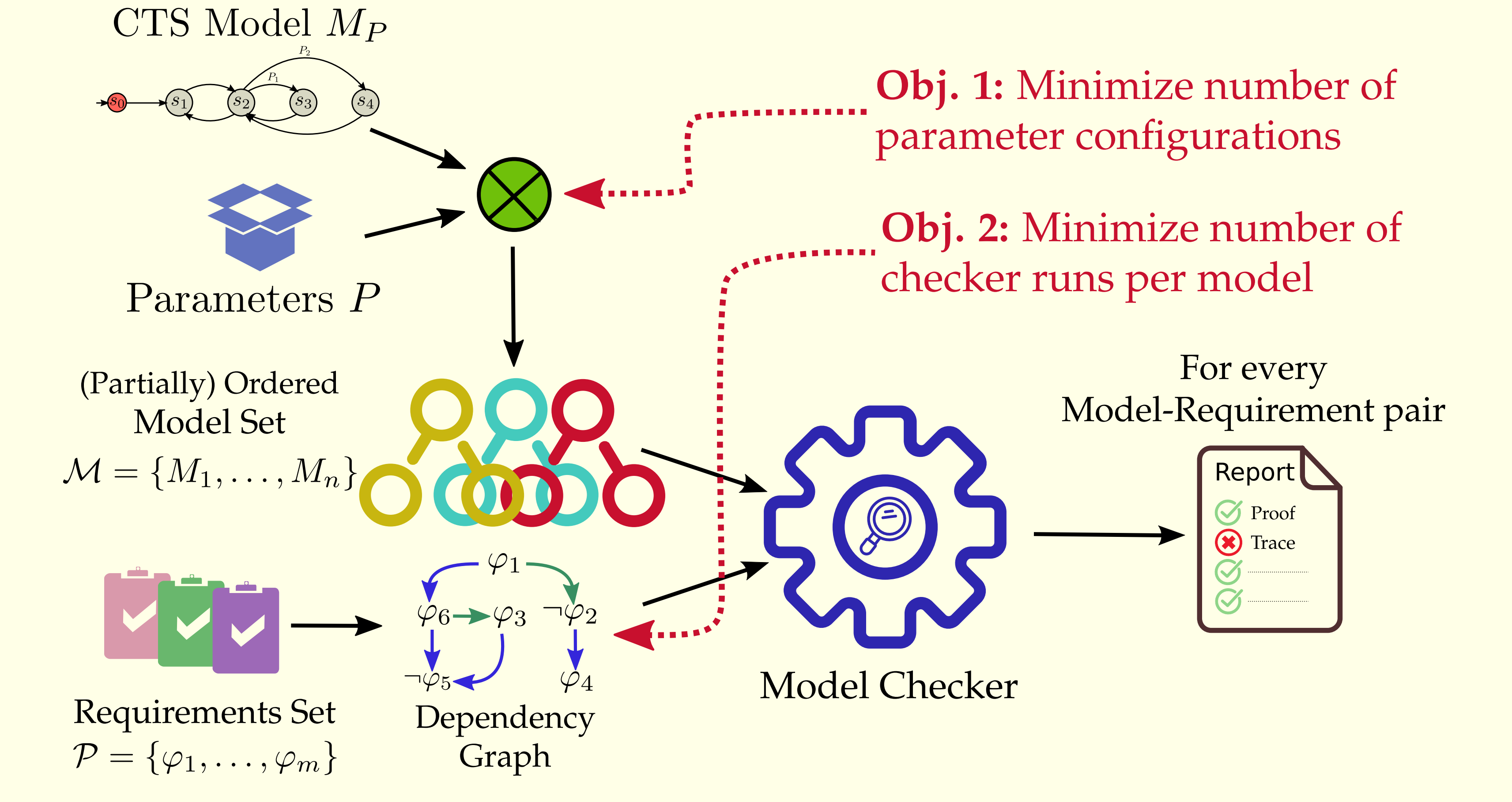
Combinatorial Transition System (CTS)

- Parameters are added as preprocessor directives.
  - works with off-the-shelf checkers, like NUXMV
  - every **parameter configuration** is a valid model



s_0
s_1
s_2
s_3
s_4
$(P_1 = \top, P_2 = \top)$
$(P_1 = \top, P_2 = \bot)$
$(P_1 = \bot, P_2 = \top)$
$(P_1 = \bot, P_2 = \bot)$

3. Problem Statement



CTS Model M_P

Parameters P

(Partially) Ordered
Model Set
$\mathcal{M} = \{M_1, \ldots, M_n\}$

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

Dependency
Graph

Model Checker

For every
Model-Requirement pair

Obj. 1: Minimize number of parameter configurations

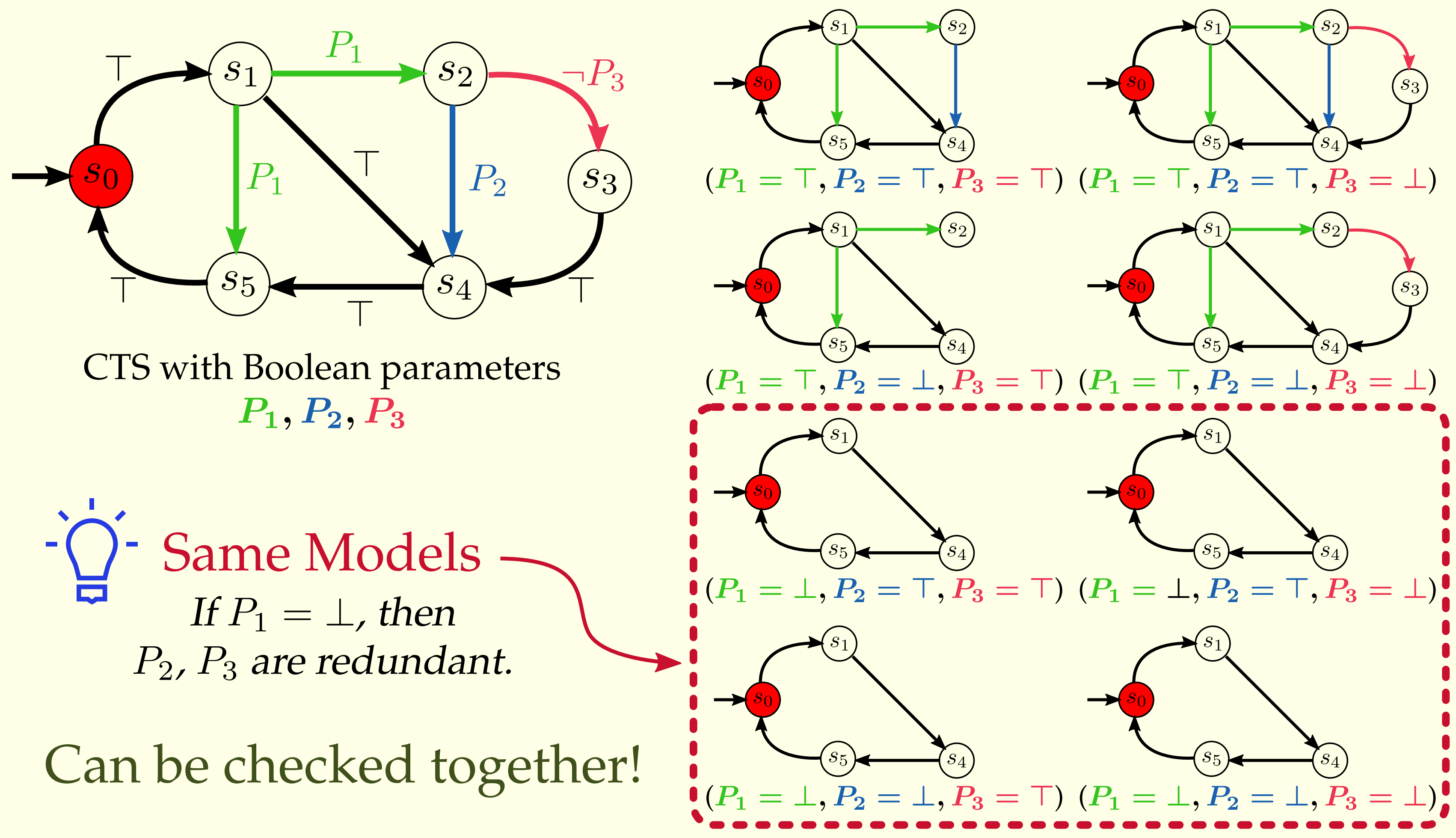
Obj. 2: Minimize number of checker runs per model

4. Our Solution

Discover **D**esign-Space **D**ependencies, or D^3

- Reduces design space by finding dependencies between:
  - **parameters** (number of models to check)
  - **properties** (number of model-checking runs)
- Is fully automatic, works with off-the-shelf checkers

i) **Minimize number of parameter configurations** (GENPC)



CTS with Boolean parameters
P_1, P_2, P_3

Same Models
If $P_1 = \bot$, then
P_2, P_3 are redundant.

Can be checked together!

s_0
s_1
s_2
s_3
s_4
s_5
P_1
P_2
$\neg P_3$
$\top$
$(P_1 = \top, P_2 = \top, P_3 = \top)$
$(P_1 = \top, P_2 = \top, P_3 = \bot)$
$(P_1 = \top, P_2 = \bot, P_3 = \top)$
$(P_1 = \top, P_2 = \bot, P_3 = \bot)$
$(P_1 = \bot, P_2 = \top, P_3 = \top)$
$(P_1 = \bot, P_2 = \top, P_3 = \bot)$
$(P_1 = \bot, P_2 = \bot, P_3 = \top)$
$(P_1 = \bot, P_2 = \bot, P_3 = \bot)$

- Finds dependencies between parameter settings via reduction to a reachability problem.

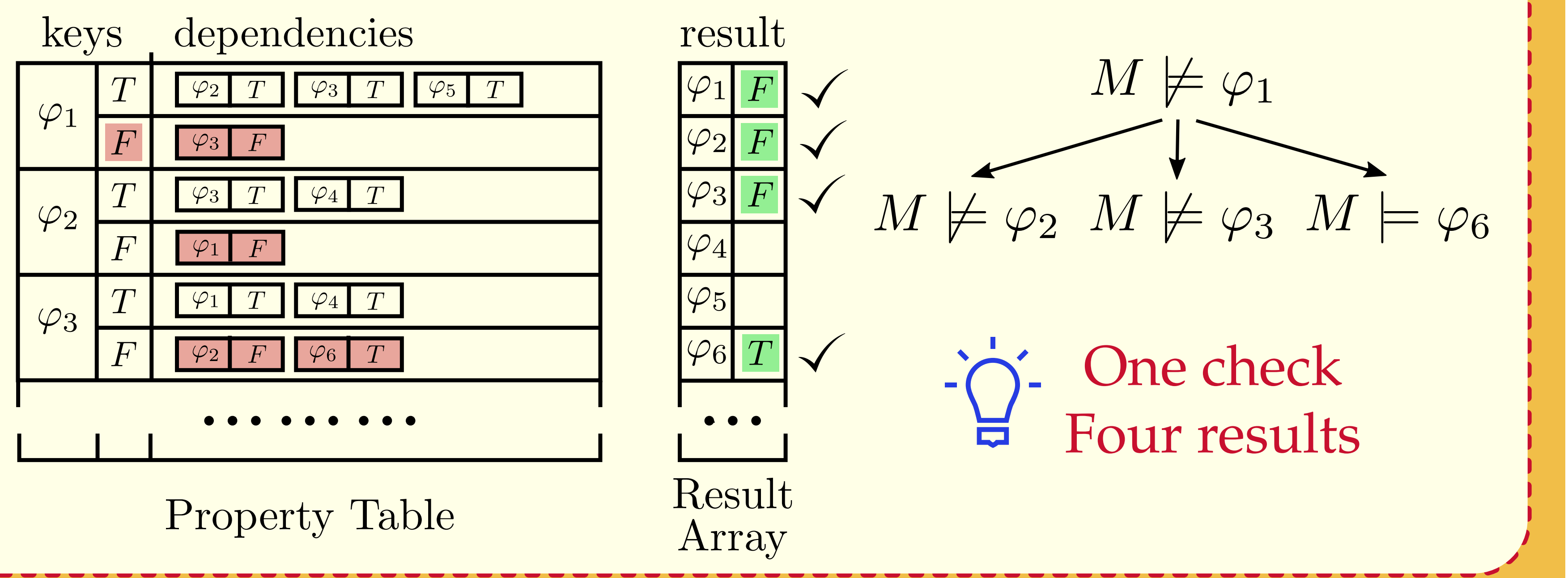
ii) **Minimize number of model-checking runs** (CHECKRP)

$\varphi_1 = \Box p$ $\qquad \varphi_2 = \Box(p \wedge q)$ $\qquad \varphi_3 = \Box(p \vee q)$

$M \models \varphi_2$ then $M \models \varphi_1$
φ_1 and φ_2 are dependent

$M \models \varphi_2$ then $M \models \varphi_3$
φ_2 and φ_3 are dependent

- Finds dependencies between properties via fast LTL satisfiability checking.



keys		dependencies
φ_1	T	φ_2 T, φ_3 T, φ_5 T
	F	φ_3 F
φ_2	T	φ_3 T, φ_4 T
	F	φ_1 F
φ_3	T	φ_1 T, φ_4 T
	F	φ_2 F, φ_6 T
		………

Property Table

result		
φ_1	F	✓
φ_2	F	✓
φ_3	F	✓
φ_4		
φ_5		
φ_6	T	✓
…		

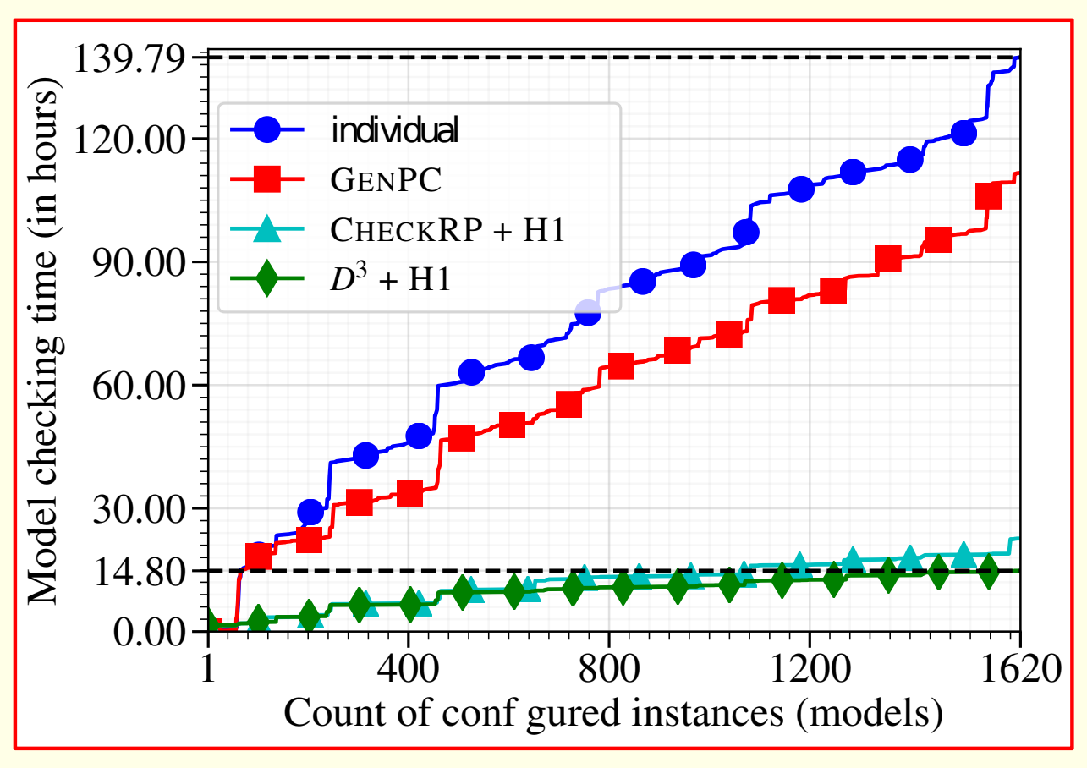
Result
Array

$$\begin{tikzcd} & M \not\models \varphi_1 \arrow[dl] \arrow[d] \arrow[dr] & \\ M \not\models \varphi_2 & M \not\models \varphi_3 & M \models \varphi_6 \end{tikzcd}$$

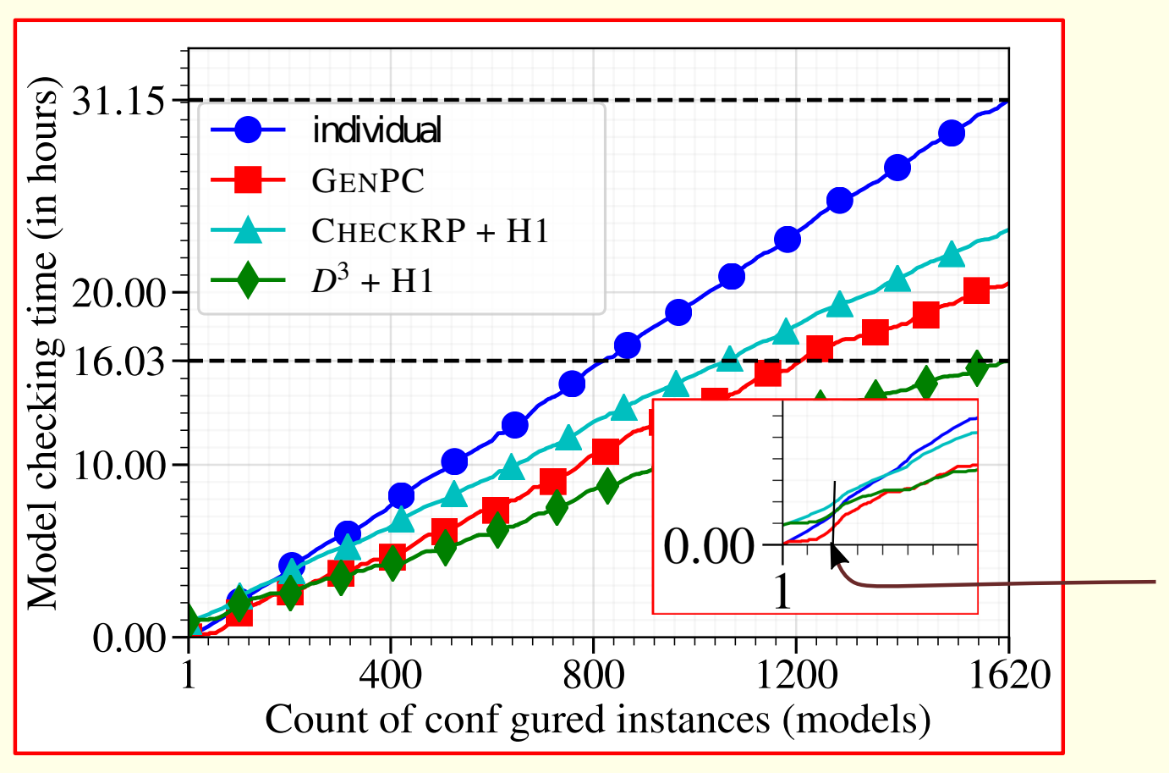
One check
Four results

5. Experimental Results

- NASA NextGen Air Traffic Control System



UNSAT only (9.0x)

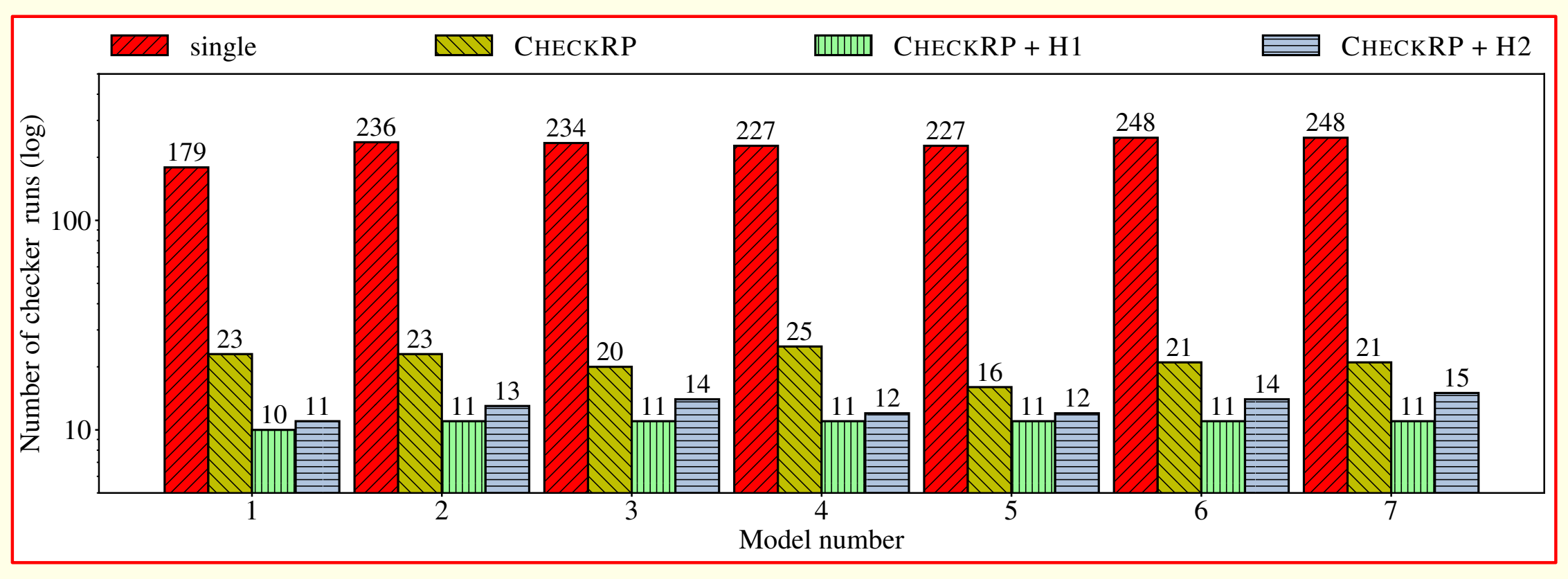


SAT + UNSAT (1.9x)

4.0x
speedup

Crossover point
(~ 120 models)

- BOEING Wheel Braking System



Heuristics
H1: Maximum Dependence
H2: Property Grouping

Fast
multi-property
verification

Supported by NSF CAREER Award CNS-1552934.

ETAPS 2018

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