ASIA SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE

SISCO: Selective Invariant Sharing, Clustering and Ordering for Effective Multi-Property Formal Verification

Sourav Das[1], Aritra Hazra[1]
Pallab Dasgupta[2], Himanshu Jain[2], Sudipta Kundu[2]

[1]Indian Institute of Technology Kharagpur

SYNOPSYS® [2]Synopsys Inc. USA

Outline

01 **Introduction**

02 **Current State of the Art**

03 **PDR**

Property Directed Reachability Overview

04 **Motivating Examples**

05 **SISCO**

Framework and Algorithm

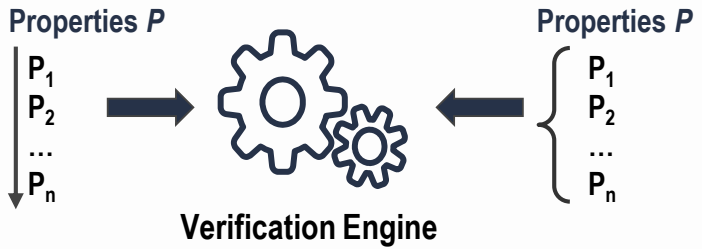
06 **Experimental Results**

07 **Conclusion and Future Work**

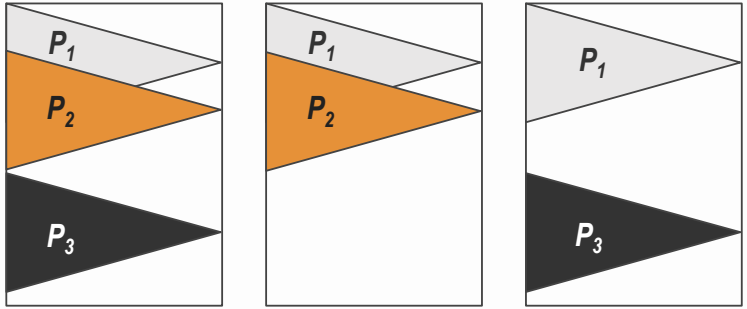
Introduction

- Model Checking tools are not designed for multi-property verification
- Static Cone-of-Influence (COI) methods exists but they do not scale for practical or industrial designs
- Key observation from recent experience:
    1. Inductive Invariants discovered while proving one property can help another property
    2. This motivates ordering properties in a way that earlier ones helps the later ones
    3. Unfortunately there is no way to know a priori which one will help which
- Question that lead to this work:
    1. Are all invariants required for solving undecided goals?
    2. Can we improve effectiveness of falsified goals?

Current State of the Art

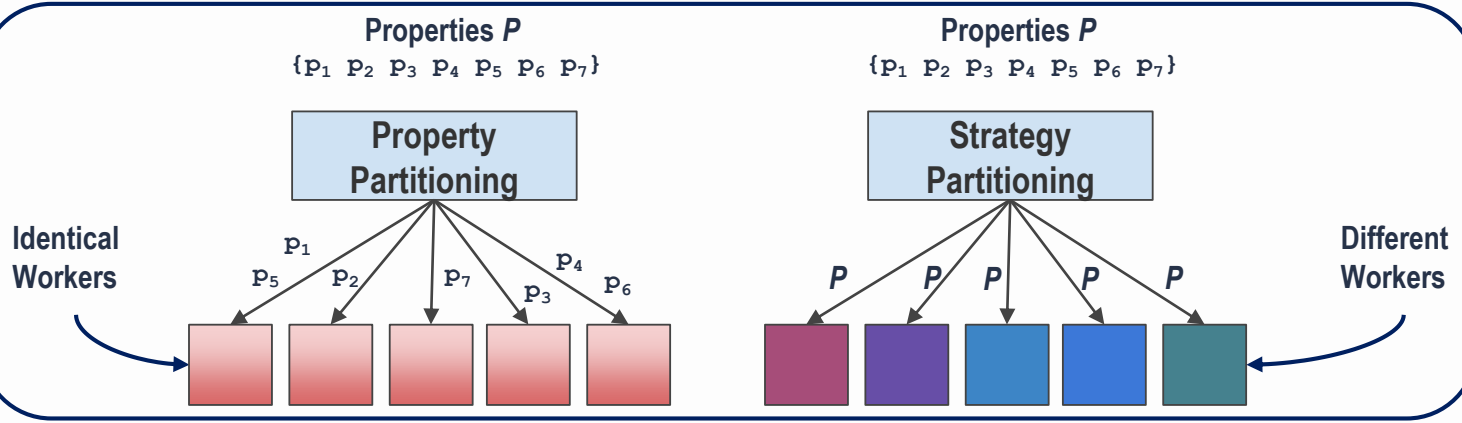


Sequential Verification of Multi-property Designs

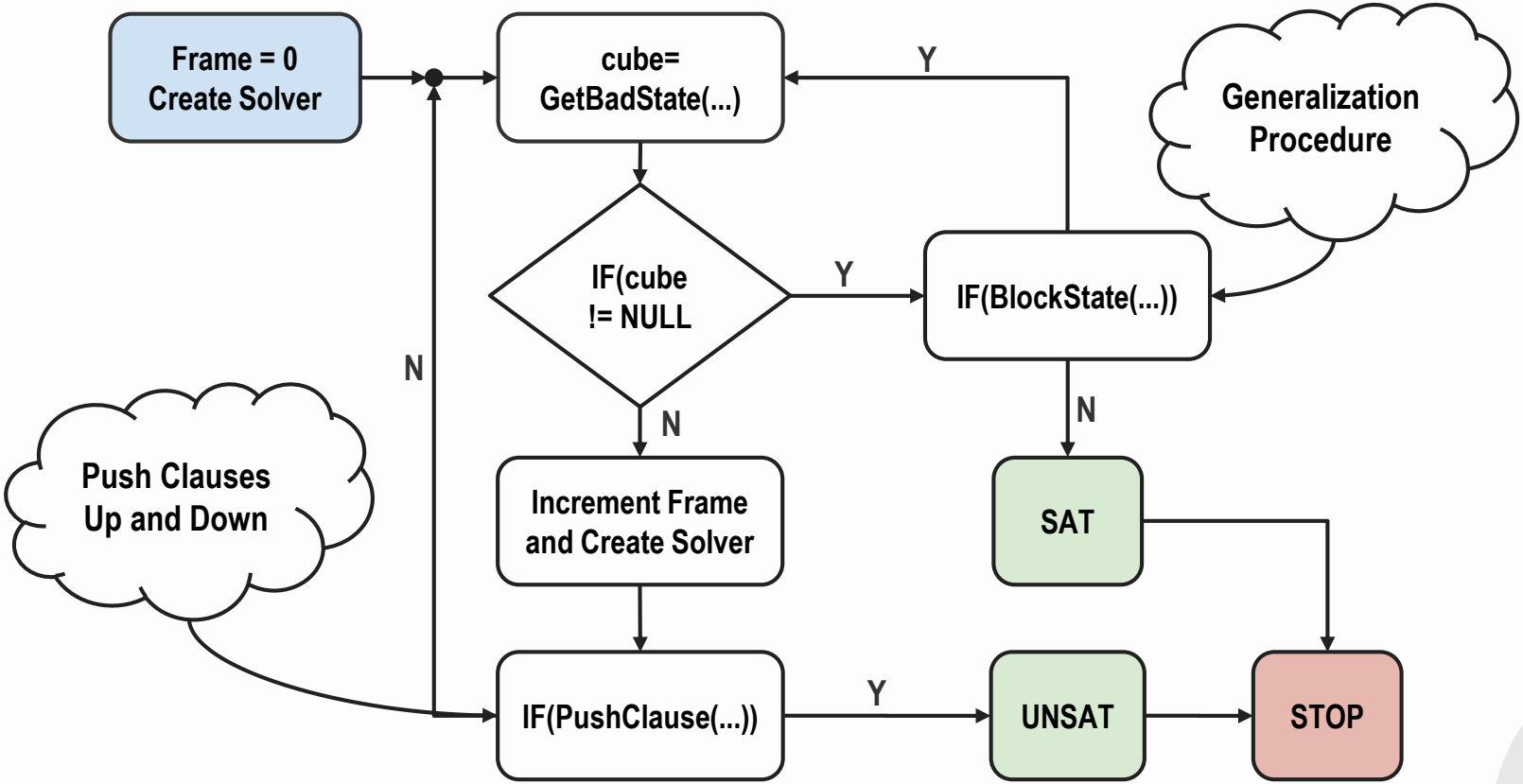


Property Clustering of designs based on COI

Parallel Verification of Multi-Property Designs



Property Directed Reachability (PDR)



Motivating Examples

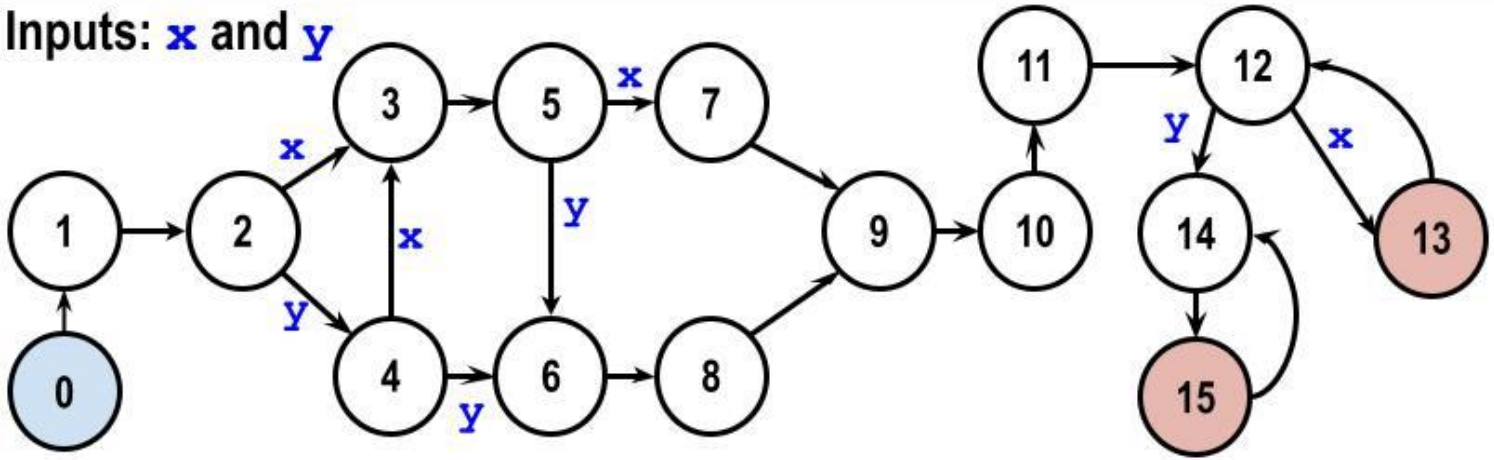
Claims and Objectives

01 **Storing Counter Example (CEX) traces helps**

02 **Invariant Sharing is required**

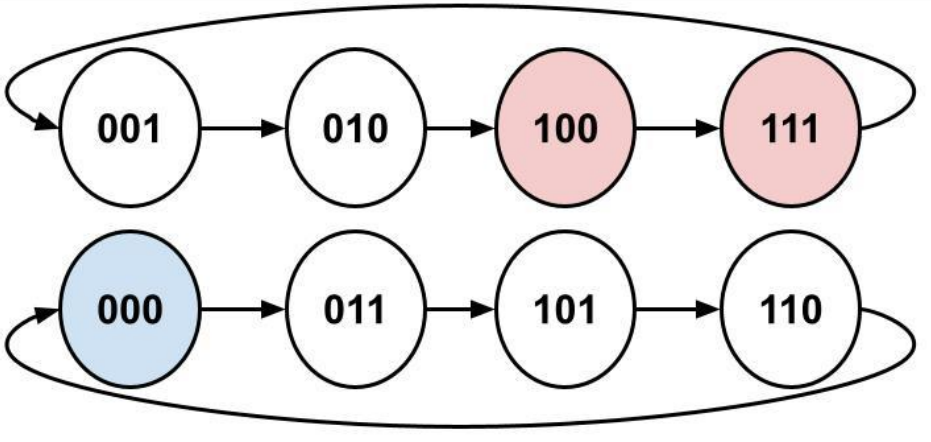
03 **All the Invariants are not useful**

Motivating Example (Falsified)



Without CEX Traces					With CEX Traces				
#Block	#PO	#Calls	#Clause	#Time	#Block	#PO	#Calls	#Clause	#Time
8	42	76	12	<1	8	29	63	9	<1

Motivating Example (Proven)



Without CEX traces and invariants					With CEX traces and all invariants					With CEX traces and selective invariants				
#Block	#PO	#Calls	#Clause	#Time	#Block	#PO	#Calls	#Clause	#Time	#Block	#PO	#Calls	#Clause	#Time
7	31	135	23	<1	3	13	60	14	<1	4	14	65	15	<1

Motivating Example

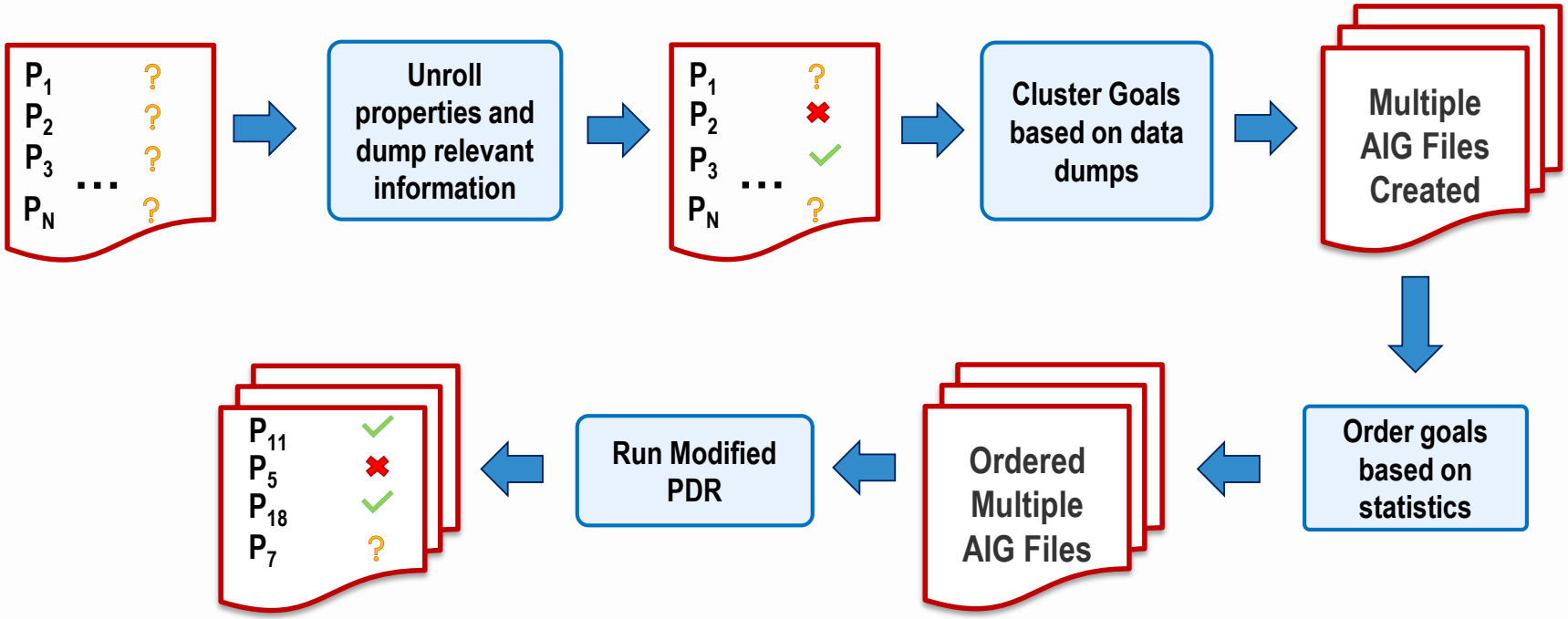
Design Stats of HWMCC Benchmark 6s421	
Flops	951
Inputs	153
Properties	150
Gates	6294

Without CEX traces and invariants					With CEX traces and all invariants					With CEX traces and selective invariants				
#Block	#PO	#Calls	#Clause	#Time	#Block	#PO	#Calls	#Clause	#Time	#Block	#PO	#Calls	#Clause	#Time
882	44681	272187	23399	87.91	884	32053	4336287	1066727	816.16	883	76745	286433	33625	82.92

SISCO

Selective Invariant Sharing, Clustering and Ordering for Effective Multi-Property Formal Verification

SISCO: Framework



SISCO: Contributions

01

Clustering and Ordering

- Clustering of goals based on clause and variable overlap
- Order goals within the cluster based on statistics dumped

02

Selective Invariant Sharing

- Selectively share invariants of already proven goals within a cluster
- Avoid sharing of invariants across different clusters

03

Storing CEX traces for falsified goals

- CEX traces are stored within a cluster to avoid repetitive task of the PDR runtime engine

Clustering and Ordering

Clustering of goals based on clause and variable overlap

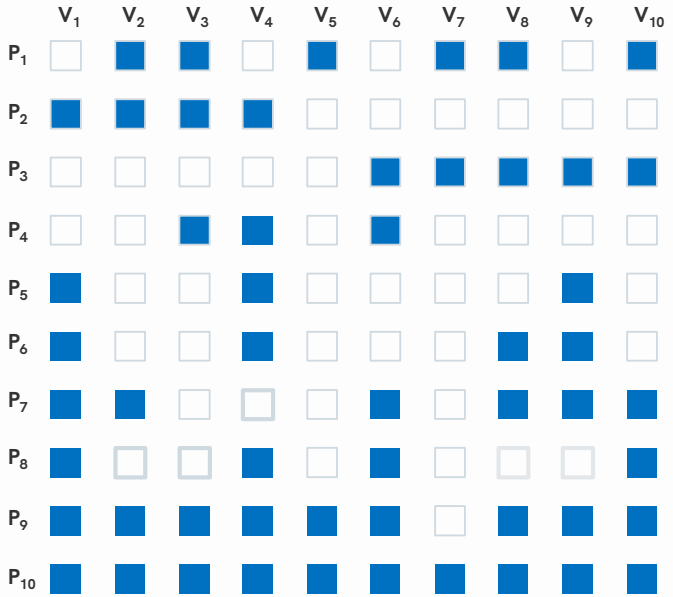
- Jaccard Index (A,B) = $|A \cap B| / |A \cup B|$
- **JI (clauses(P_1), clauses (P_2)) > CL_Upper_Threshold**
- **JI (clauses(P_1), clauses (P_2)) > CL_Lower_Threshold**

  Λ

  JI (variables(P_1), variables(P_2)) > Literals_Threshold

Order properties within the cluster based on statistics

- (Total calls to SAT solver / Frames Explored)*

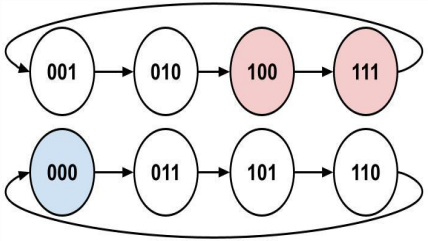


* PURSE: Property Ordering Using Runtime Statistics for Efficient Multi-Property Verification

Selective Invariant Sharing

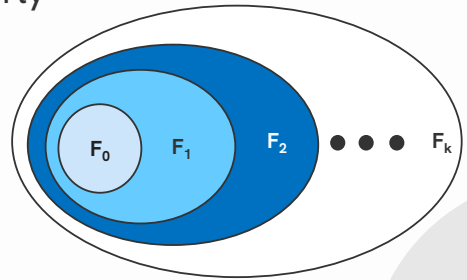
When a property gets proven store invariants along with flop count.

- Property P_0: 100 and Property P_1: 111
- Invariant of Property P_0: {001, 010, 100, 111}
- Flop count: {x_0: 4, x_1: 4, x_2: 4}
- Maintain a map that has the count of each flops and sort it in descending order of flops count.



When the next property starts

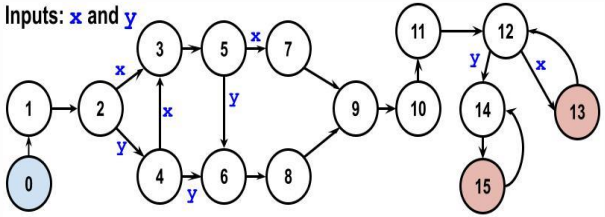
- Compute the flop count of the frame clauses and sort them similarly
- Pick the first ‘k’ variables from the invariant map and the current property frame clauses and check the overlap.
- If it crosses a certain threshold then add that invariant
- Check for other stored invariants as well
- Continue the process for each new frames added



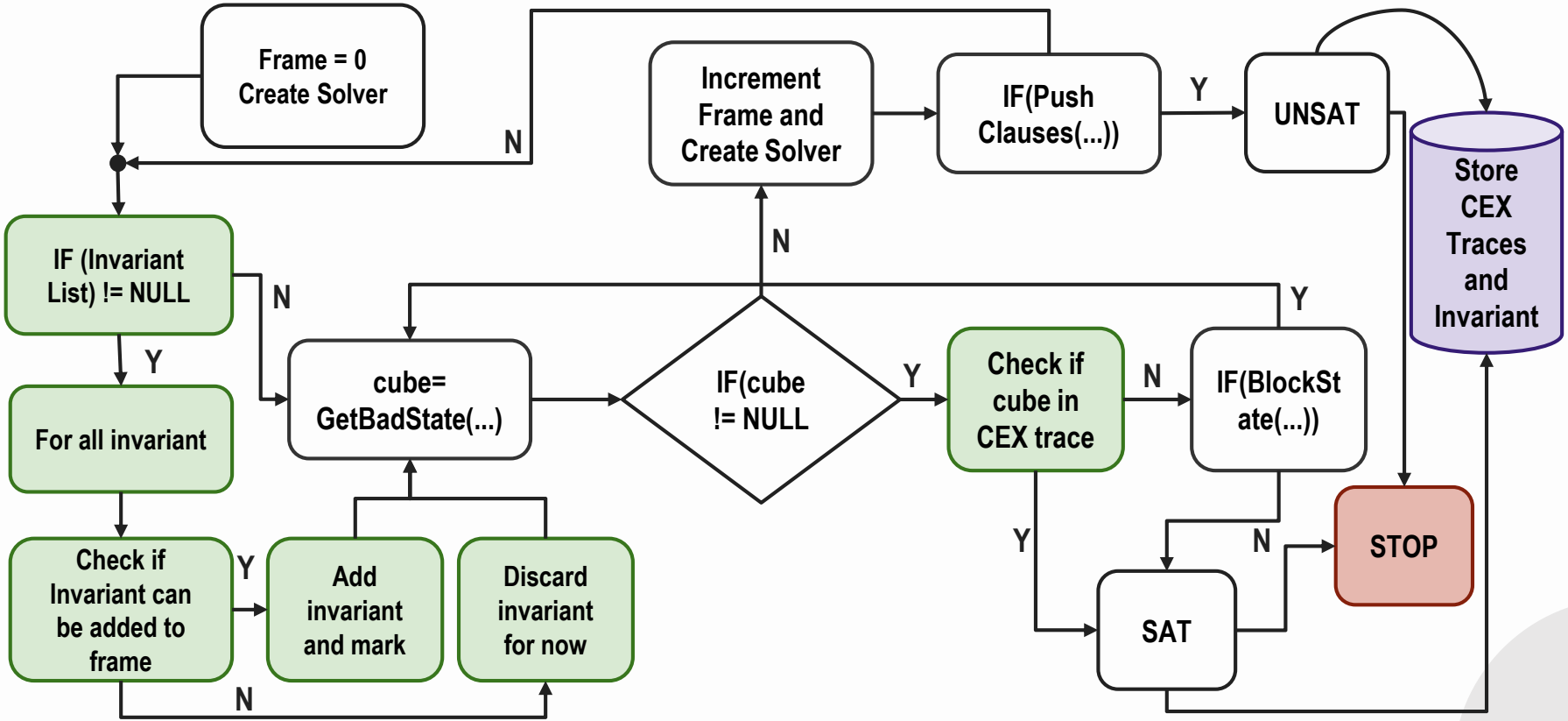
Storing CEX Traces

When a property gets falsified store CEX trace

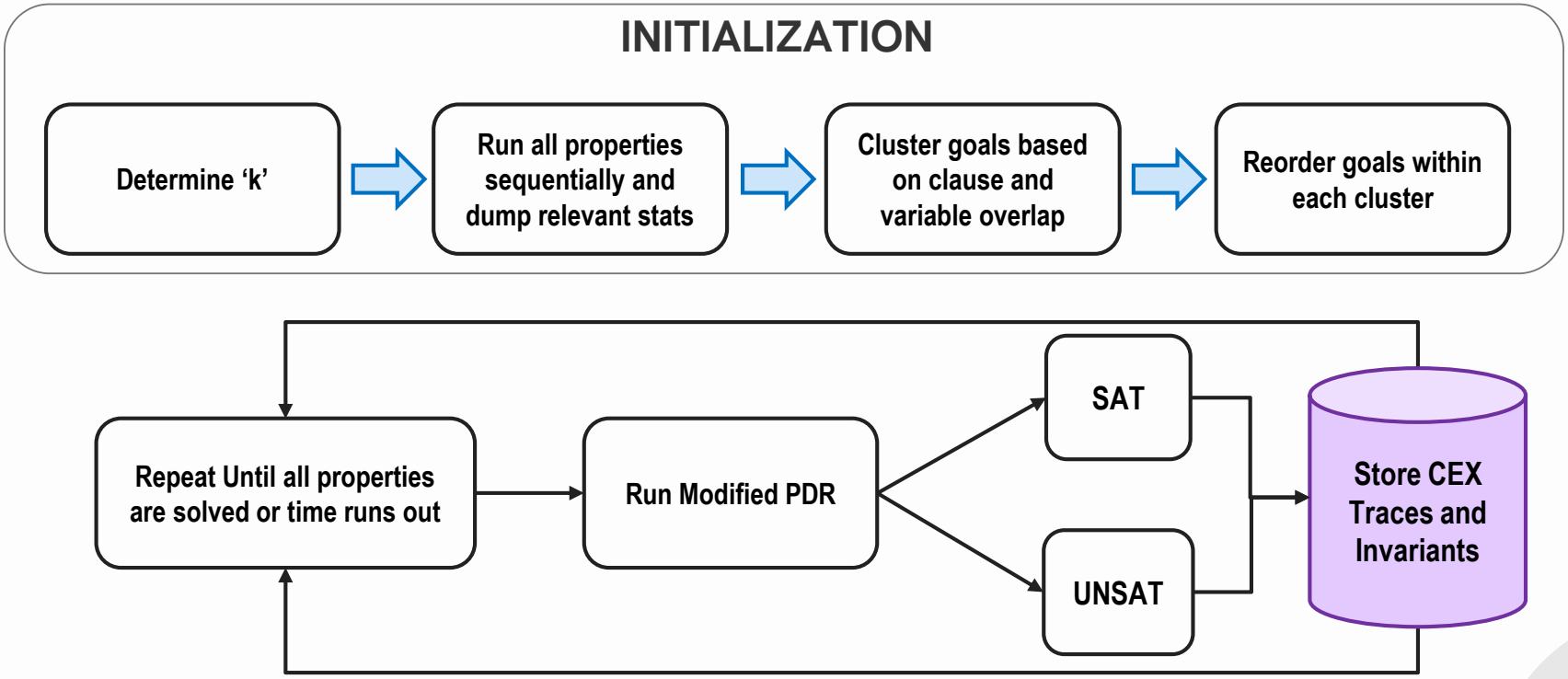
- The hash of each state in the CEX trace is stored also
- Initially when a bad state is encountered its hash is matched
- For example the CEX trace for property 13 is:
  - {13 → 11 → 9 → 6 → 2 → 1 → 0}
- When property 15 starts it needs to block the following:
  - { (15,1) (15,2) (15,3) (15,4) (15,5) (14,4) (12,3) (11,3) }
- Once a state that needs to be blocked exists in the CEX trace then stop and report SAT
- CEX trace is recomputed by extracting the states from the current queue followed by the states from the CEX trace of the already falsified property
- Here CEX trace for property 15 is:
  - {15 → 14 → 12 → 11 → 9 → 6 → 2 → 1 → 0}



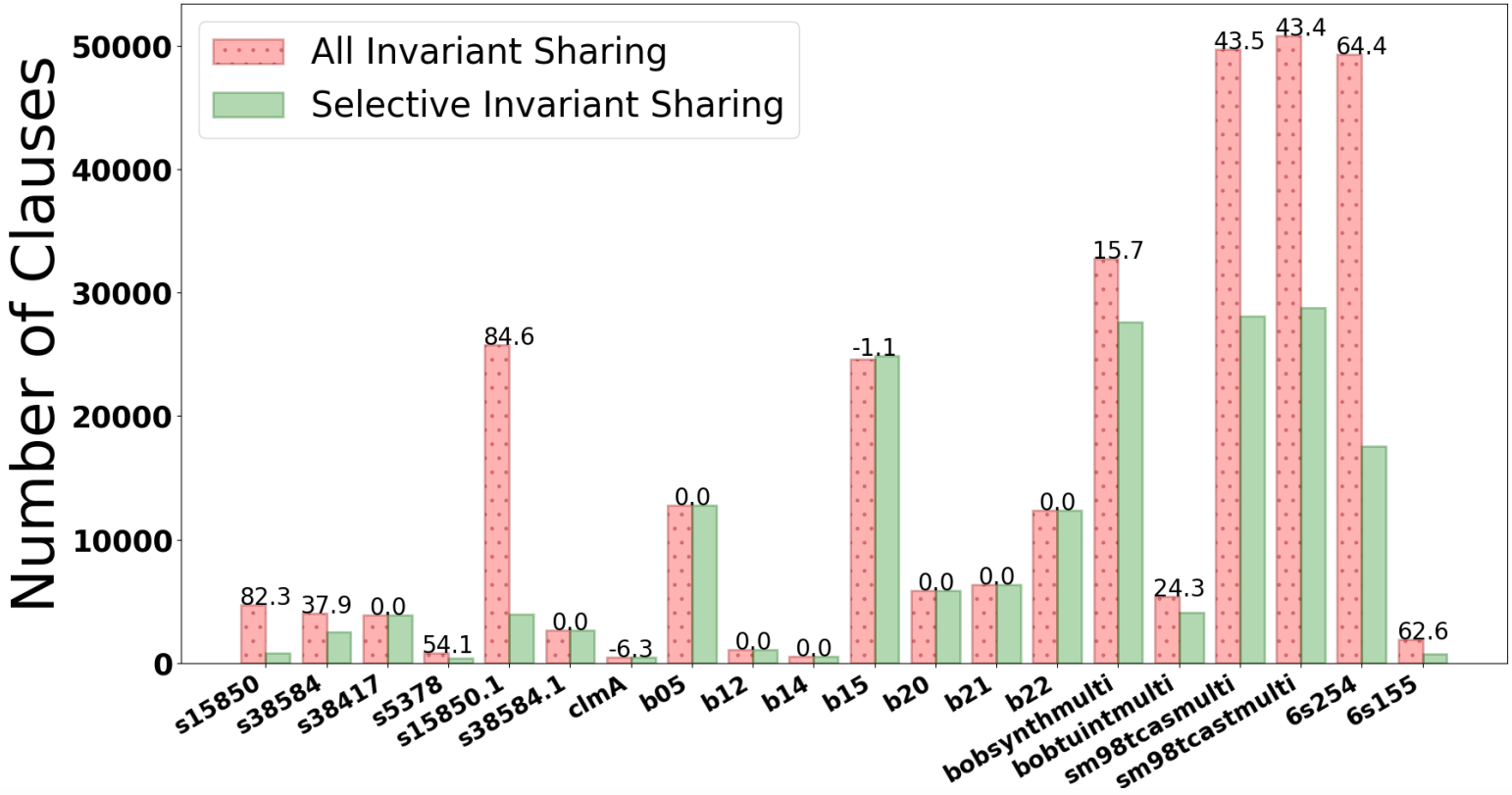
Modified PDR



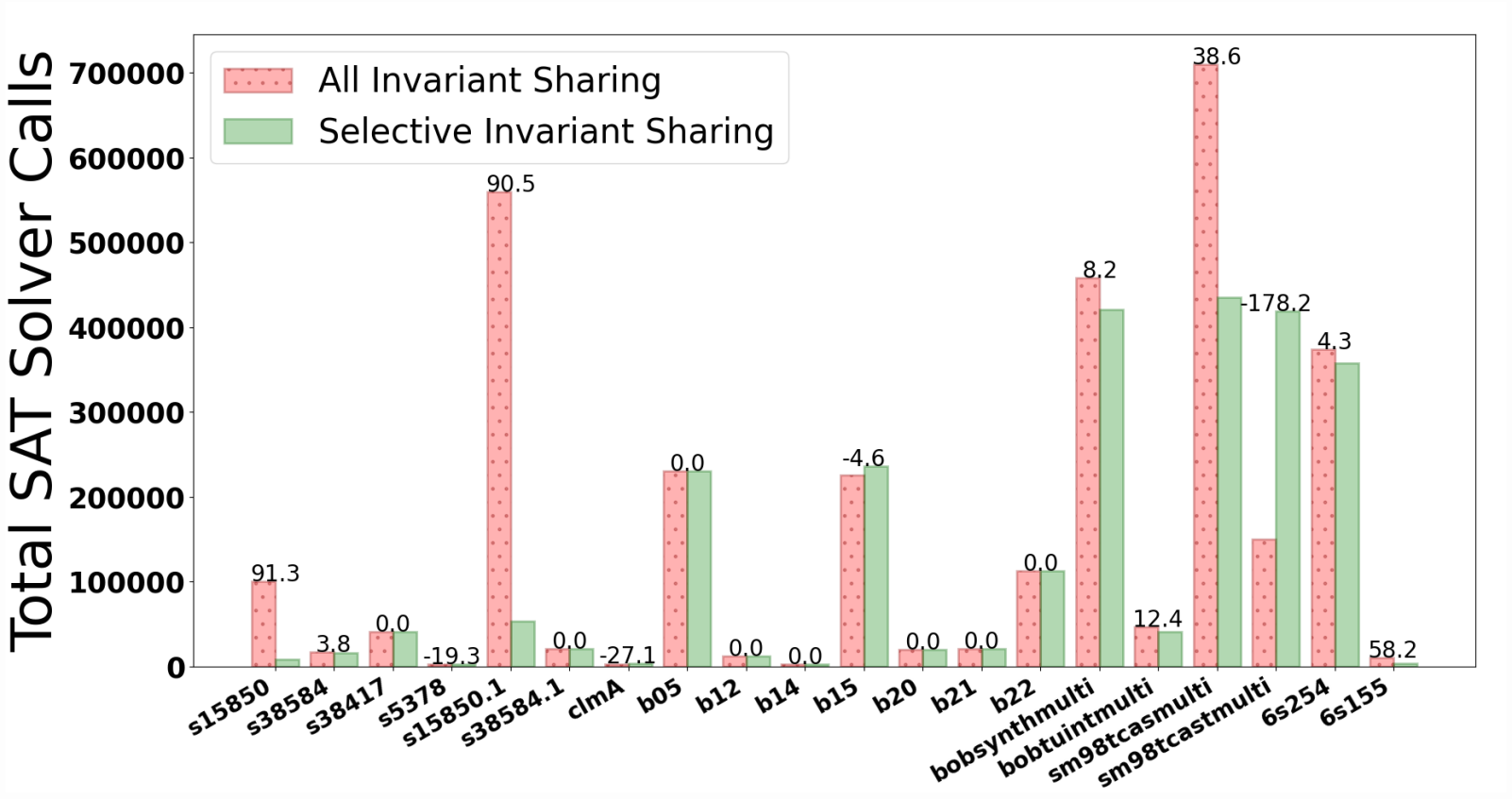
SISCO: Algorithm



Experimental Results



Experimental Results



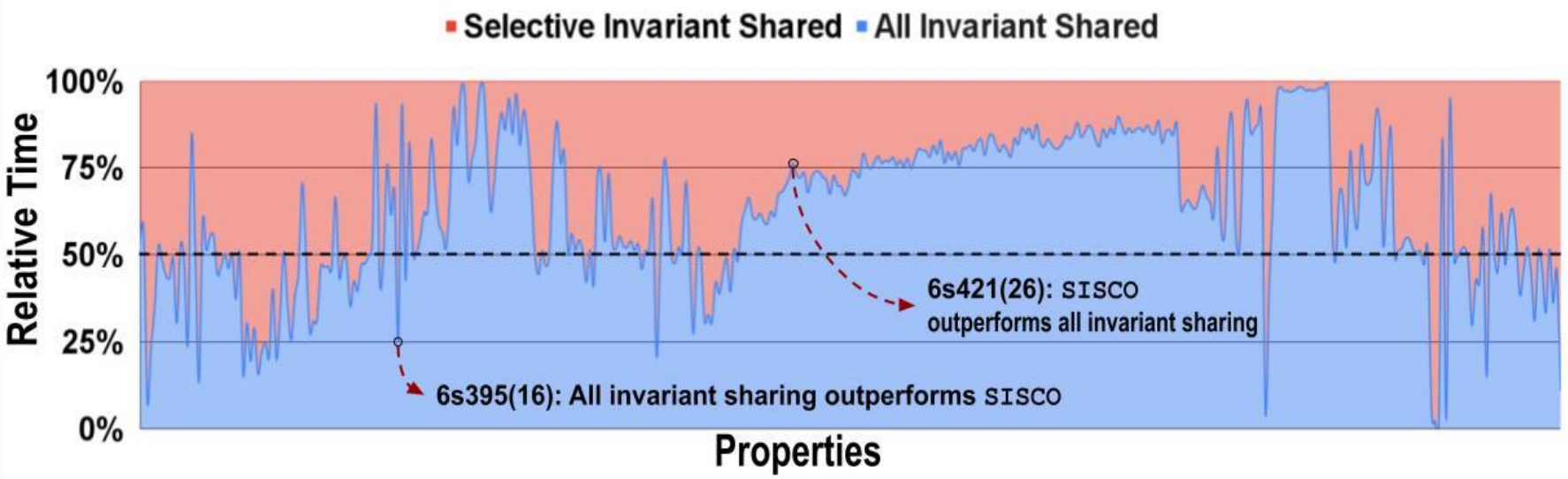
Experimental Results

Designs	Design Stats				Cluster Information		Undecided Goals		
	#Inputs	#Flops	#Gates	#Prop.	#Clusters	#Time	Baseline	State-of-Art	SISCO
6s168	34	98	2722	16	1	0.01	2	0 (7200)	**3**
pdtvsarmultip	17	130	2743	33	1	0.02	3	0 (772)	**0 (822.10)**
pdtvsar8multip	23	195	6956	33	2	0.03	4	2	**2**
bob12m18m	57	261	2140	163	1	0.14	158	6	**6**
bob12m09	44	285	30250	85	4	0.15	0 (4037.13)	0 (4127)	**0 (4631.15)**
sm98tcas16multi	279	310	5068	6	1	0.01	4	1	**1**
sm98tcas16tmulti	279	310	5168	6	1	0.01	4	5	**5**
6s395	21	463	3703	129	13	1.73	128	1	**1**
s13207.1	62	638	2721	152	1	0.56	59	6	**6**
s13207	31	669	2721	121	1	0.2	34	7	**7**

Experimental Results

Designs	Design Stats				Cluster Information		Undecided Goals		
	#Inputs	#Flops	#Gates	#Prop.	#Clusters	#Time	Baseline	State-of-Art	SISCO
6s421	153	951	6294	150	1	1.5	0 (87.91)	0 (3004.92)	**0 (116.95)**
b17	37	1415	27549	97	1	3.63	78	0 (2935.16)	**0 (3838.53)**
6s419	279	1653	26811	9	2	0.13	9	3	**3**
6s258	286	1790	27501	80	9	9.6	80	77	**77**
6s391	433	2686	13716	387	3	3.21	375	56	**56**
mentorbm1	224	4376	31613	13	1	0.11	0 (553.90)	1	**0 (1821.35)**
6s116	4912	4922	25104	17	1	0.18	1	3	**3**
6s321	22	13126	66695	6	1	0.07	6	2	**2**
6s118	452	13706	410804	100	2	2.06	100	83	**83**
6s281	91099	177235	2179584	90	1	1.86	76	71	**72**

Experimental Results (Observations)



- SISCO outperforms all invariant sharing in 75.13 % of the cases and provides 2x improvements in 37.56 % cases
- SISCO provides an average improvement of 4.73x in the runtime of individual goals across all designs
- Frames explored by the last unsolved goal is more in SISCO than all invariant sharing

Conclusion

01

Clustering and Ordering

SISCO clusters and orders properties based on data dumps after the initial unrolling phase

02

Storing Invariants and CEX Traces

Invariants are shared selectively and CEX traces are stored along with their hashes for faster comparison

03

Flexibility in Framework

One can use a completely different ordering, clustering or modified PDR and tests its effectiveness

Thanks!

Do you have any questions?