INFUSE: Towards Efficient Context Consistency by Incremental-Concurrent Check Fusion

Lingyu Zhang, Huiyan Wang, Chang Xu, and Ping Yu

Context-aware Computing

- Context: a model of applications' runtime environments[9-12]
  - E.g., GPS data, speed, temperature, picture, etc.

- Usage: applications' smart adaptions based on contexts facilitate people's lives

SmartHome application

Context Problem

- Quality problems: inaccurate, incomplete, or conflicting with each other due to uncontrollable sensor instability[9-12]

- Unexpected consequence: leading to applications' misbehaviors or crashes

Misbehavior: improper temperature

Common Practice

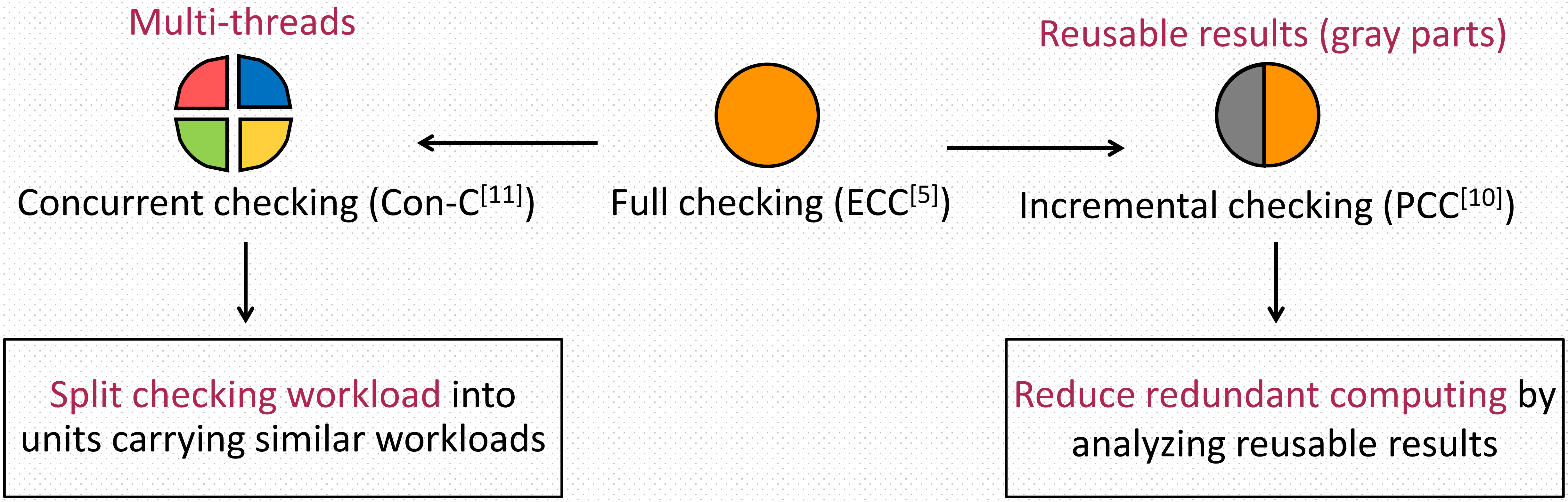
- Constraint checking: checking contexts against *consistency constraints* to see whether any violation (named *context inconsistency*) occurs

- Constraint example: “no robot can be in two rooms at the same time”

$$S_{loc}: \forall\ v_x \in R_x\ (\text{not}\ (\exists\ v_y \in R_y\ (\text{Same}(v_x, v_y))))$$

Existing Constraint Checking Techniques

- Two typical research lines



Low-efficiency Problem of Existing Techniques

- Features of context in nowadays dynamic environment: large volume and changing frequently

- Bringing unacceptable overhead to existing techniques

Technique	Time cost
ECC	19.1 ~ 137.7 h
Con-C	11.2 ~ 68.0 h
PCC	3.3 ~ 5.9 h

Cannot validate in time

Time cost of existing techniques for handling one-hour context data in SmartCity application (1.7 million data lines and 48 constraints)

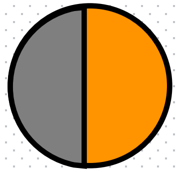
Call for more efficient checking techniques

Our Natural Idea: Check Fusion

- Two orthogonal dimensions: PCC (2006) and Con-C (2013)

Concurrent checking (Con-C)

Fuse together



Incremental checking (PCC)

The more, the better small workload not suitable to split	Fusion gap	The less, the better large workload hard to analyze
Con-C’s underlying assumption		PCC’s underlying assumption

No substantial work after one decade since their initial proposals

Two Brute-Force Solutions Do not Work

- Respect "the less, the better": splitting small workload into concurrent units

Concurrent checking (Con-C)
The more, the better

Incremental checking (PCC)
The less, the better

INFUSE_0

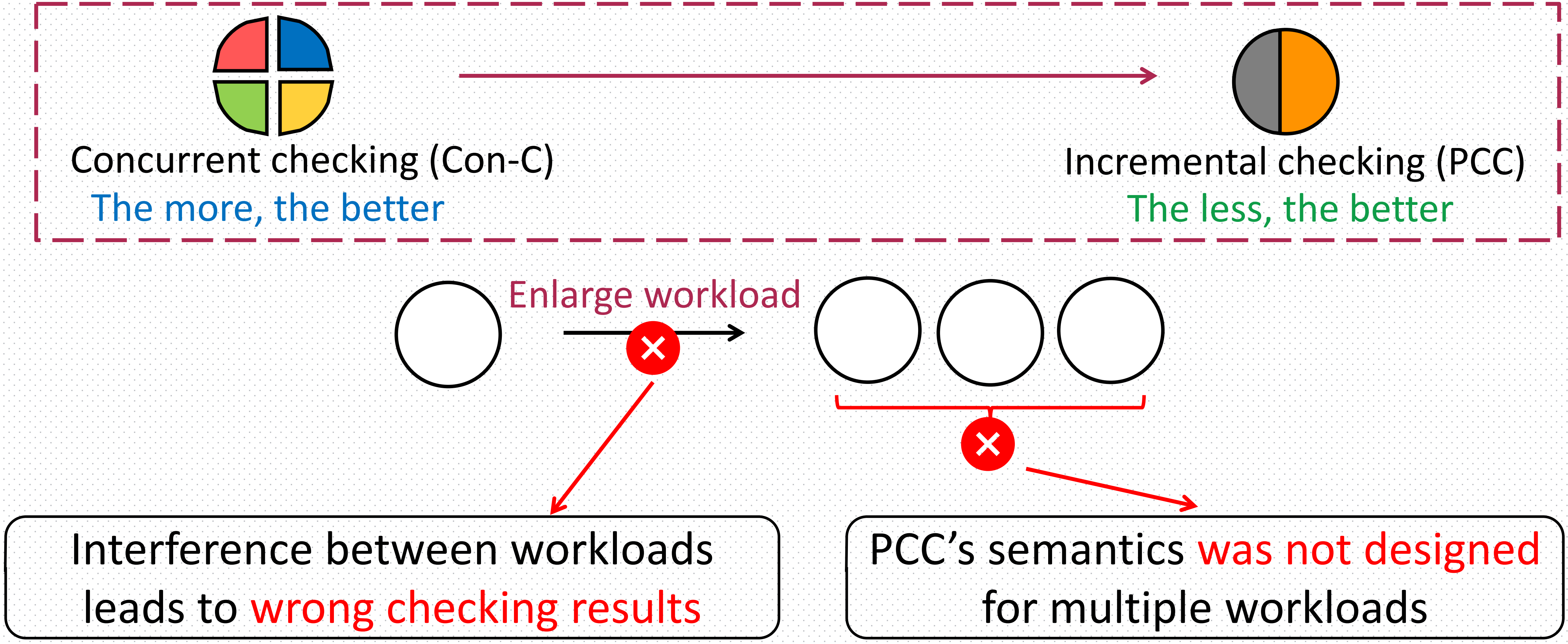
Technique	Checking time
INFUSE_0	$\geq$ 46.7 min
PCC	26.4 ~ 44.2 min

Checking time comparison from our evaluation

Performance compromise: even less efficient than pure incremental checking

Two Brute-Force Solutions Do not Work

- Respect "the more, the better": enlarging workload for fusion checking



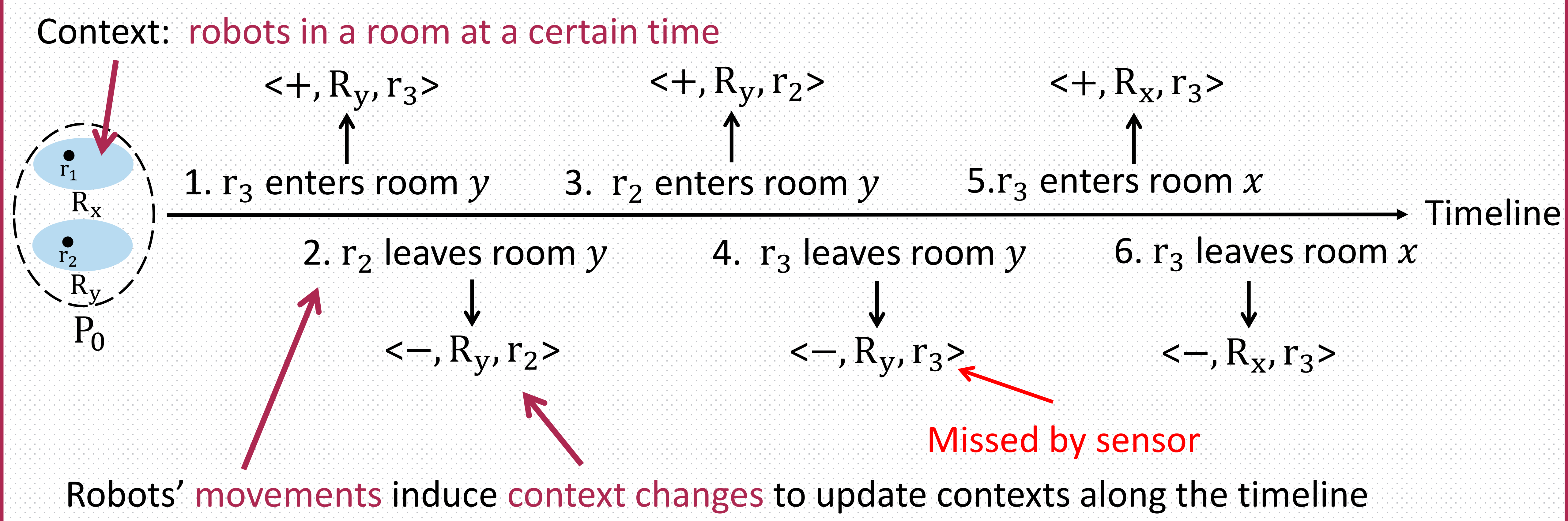
Two Faced Problems

- Summary of two brute-force solutions
  - Respect “the less, the better”: correct but inefficient
  - Respect “the more, the better”: efficient but incorrect

- Two problems for achieving both correctness and efficiency
  - What-To-Check: Which workloads can be checked together?
  - How-To-Check: How to correctly conduct fusion checking for multiple workloads?

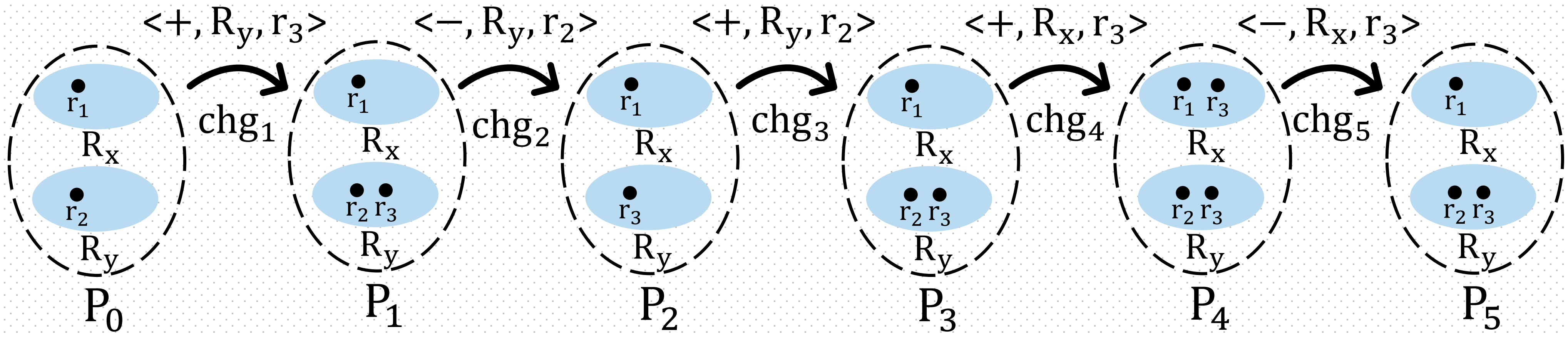
What-To-Check: Example

- Robot localization application
  - Three robots (r_1, r_2, and r_3) move between two rooms (x and y)



What-To-Check: Example

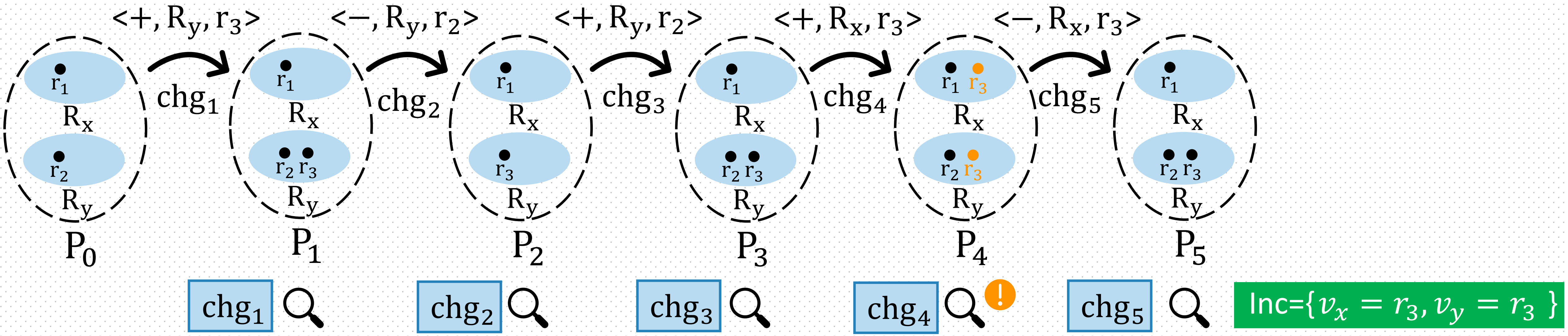
$$S_{loc}: \forall\, v_x \in R_x\ (\text{not}\ (\exists\, v_y \in R_y\ (\text{Same}(v_x, v_y))))$$



What-To-Check: Example

Correct but time consuming

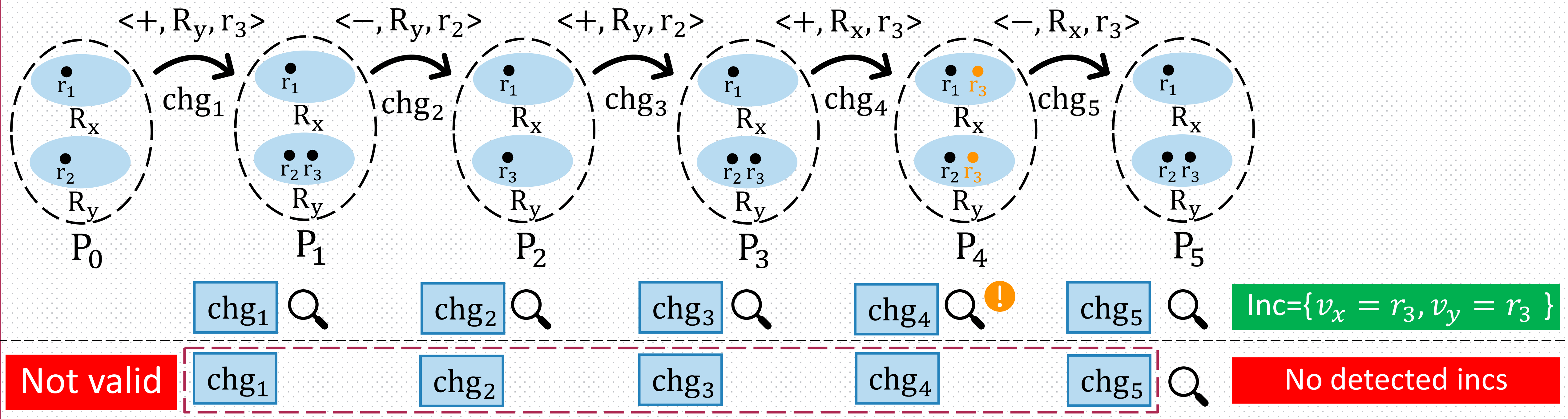
$S_{loc}: \forall\ v_x \in R_x\ (\text{not}\ (\exists\ v_y \in R_y\ (\text{Same}(v_x, v_y))))$



What-To-Check: Example

Cannot compromise validity for efficiency

$$S_{loc}: \forall\, v_x \in R_x\ (\text{not}\ (\exists\, v_y \in R_y\ (\text{Same}(v_x, v_y))))$$



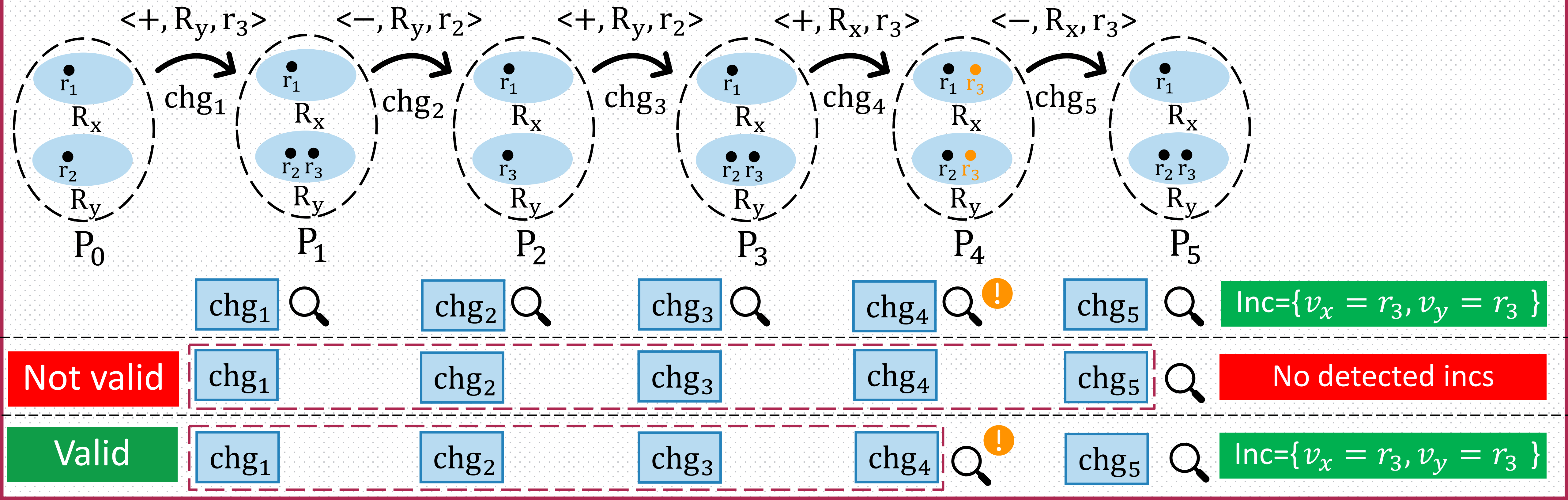
Pursuing Efficiency with Validity Guarantee

- Goal: composing a group with context changes as many as possible while guaranteeing the correctness of checking results

How to Pursue Efficiency with Validity Guaranteed?

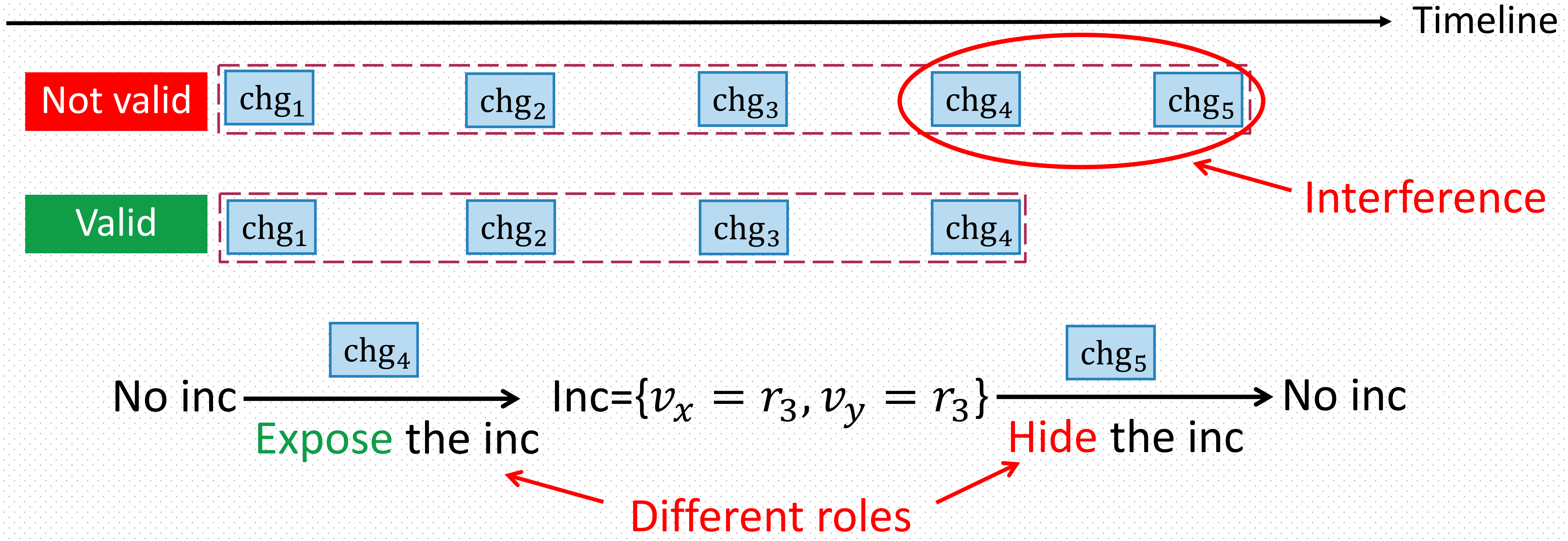
What makes the two groups different in validity?

$$S_{loc}: \forall\ v_x \in R_x\ (\text{not}\ (\exists\ v_y \in R_y\ (\text{Same}(v_x, v_y))))$$



Interference Between Changes Breaks Validity

How many roles can context changes play concerning inconsistency occurrence?



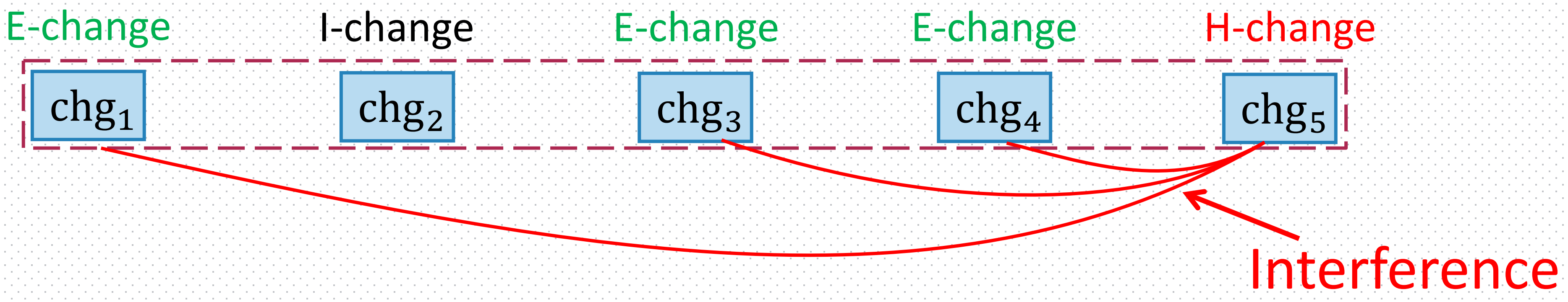
Validity Criterion

How to know the role of a context change?

- Three roles of context changes concerning inconsistency occurrence

E-change	H-change	I-change
Possibly expose new inconsistencies	Possibly hide existing inconsistencies	Irrelevant to any inconsistency

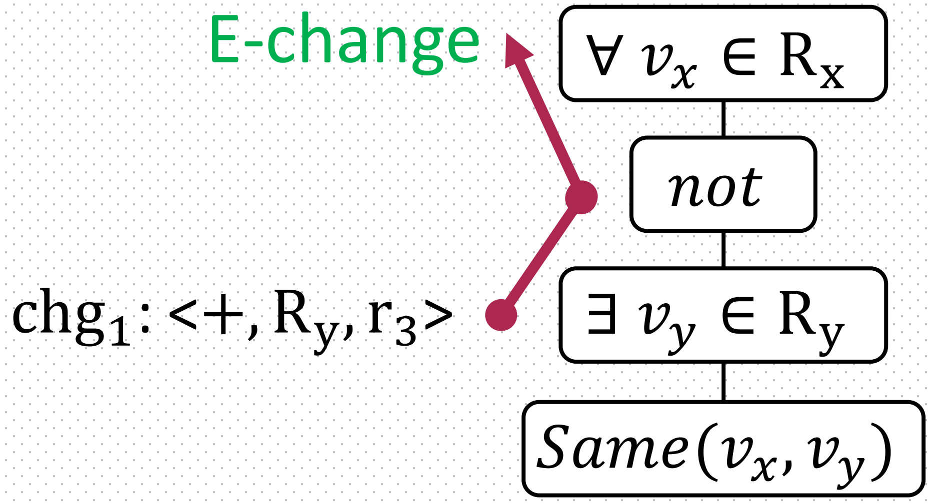
- Interference: E-change followed by H-change (may not be contiguous)



- Validity criterion: avoiding any interference in a group

Knowing the Role by Bottom-up Derivation

$S_{loc}: \forall\, v_x \in R_x$ (not ($\exists\, v_y \in R_y$ (Same(v_x, v_y))))

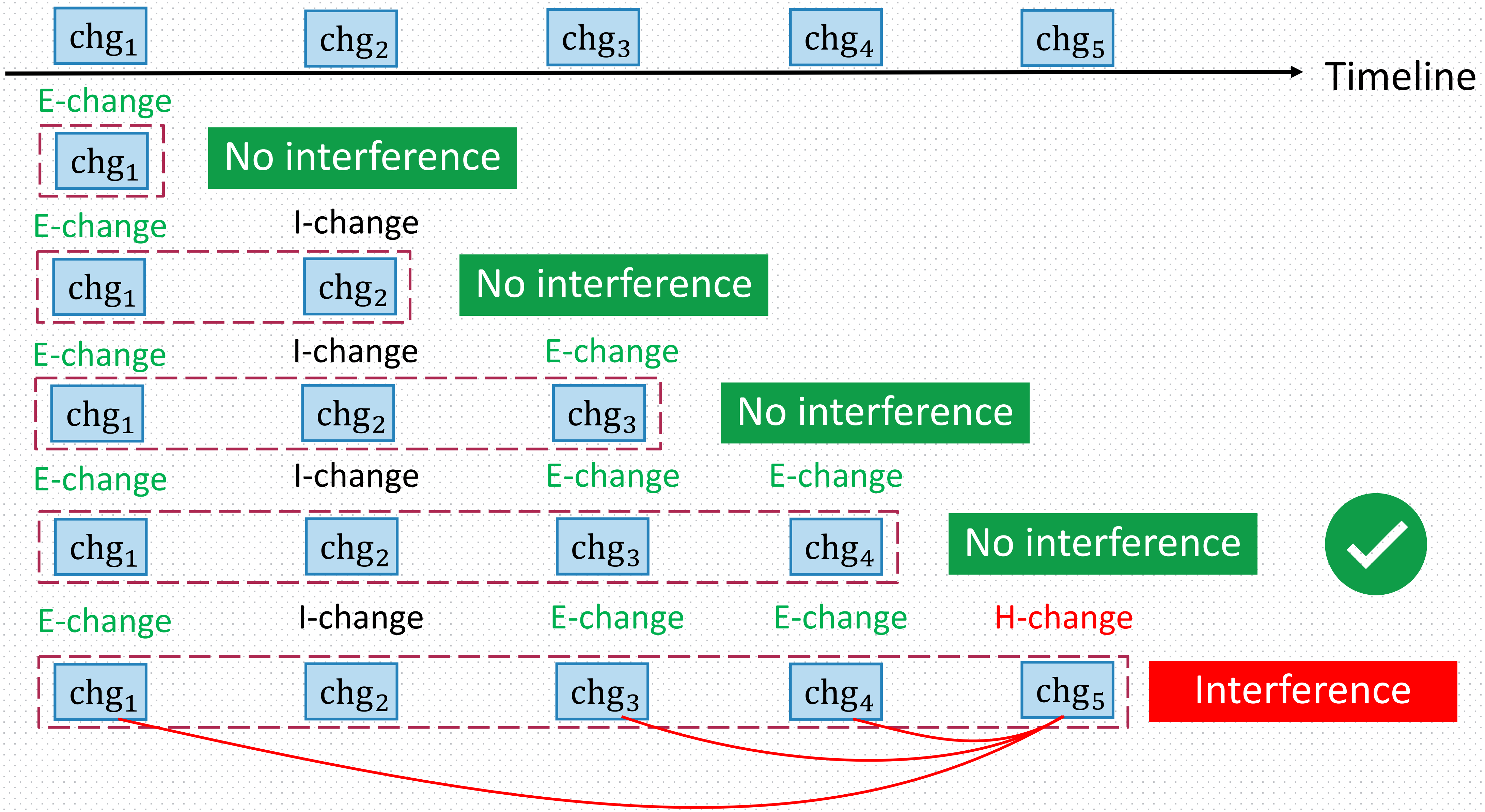


- impact(chg, $\forall v \in C(f)$) =
  (1) base_impact(chg, $\forall$), when chg affects C,
  (2) impact(chg, f) $\cup$ {m_{FF}}, when chg affects f;
- impact(chg, $\exists v \in C(f)$) =
  (1) base_impact(chg, $\exists$), when chg affects C,
  (2) impact(chg, f) $\cup$ {m_{TT}}, when chg affects f;
- impact(chg, not (f)) = flipSet(impact(chg, f));
- impact(chg, (f_1) and (f_2)) =
  (1) impact(chg, f_1) $\cup$ {m_{FF}}, when chg affects f_1,
  (2) impact(chg, f_2) $\cup$ {m_{FF}}, when chg affects f_2;
- impact(chg, (f_1) or (f_2)) =
  (1) impact(chg, f_1) $\cup$ {m_{TT}}, when chg affects f_1,
  (2) impact(chg, f_2) $\cup$ {m_{TT}}, when chg affects f_2;
- impact(chg, (f_1) implies (f_2)) =
  (1) flipSet(impact(chg, f_1)) $\cup${m_{TT}}, when chg affects f_1,
  (2) impact(chg, f_2) $\cup$ {m_{TT}}, when chg affects f_2.

Derivation rules

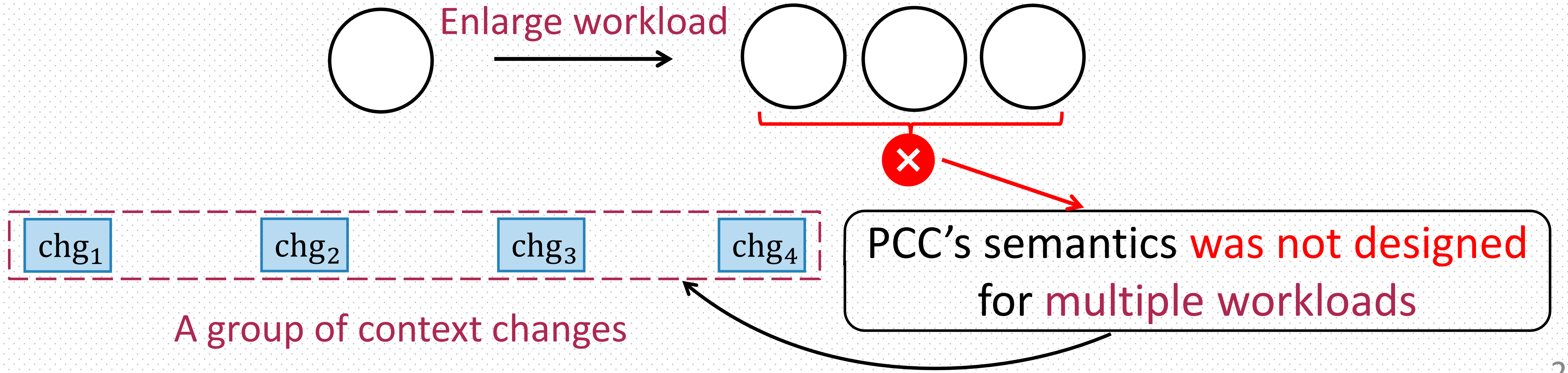
- Efficient: only related to static structure of constraints and previous checking results

Composing Groups by Validity Criterion



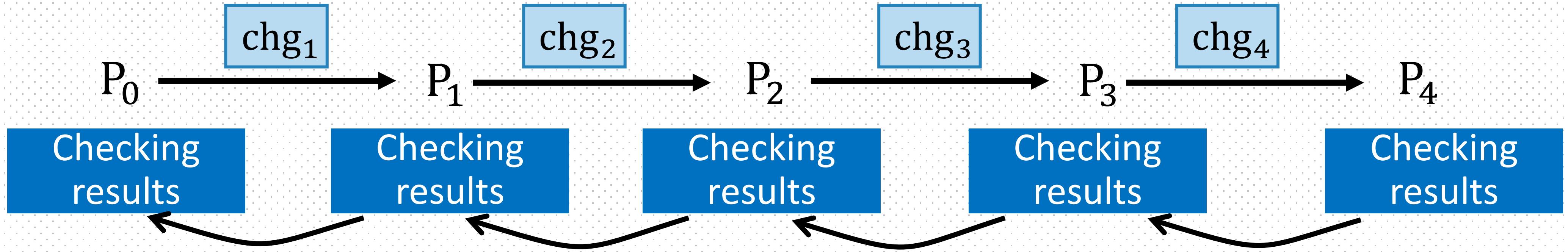
Two Faced Problems

- What-To-Check
  - Which workloads can be checked together for enlarging workload? ✓
- How-To-Check
  - How to correctly conduct fusion checking after enlarging workload?



PCC Originally Designed for Single Context Change

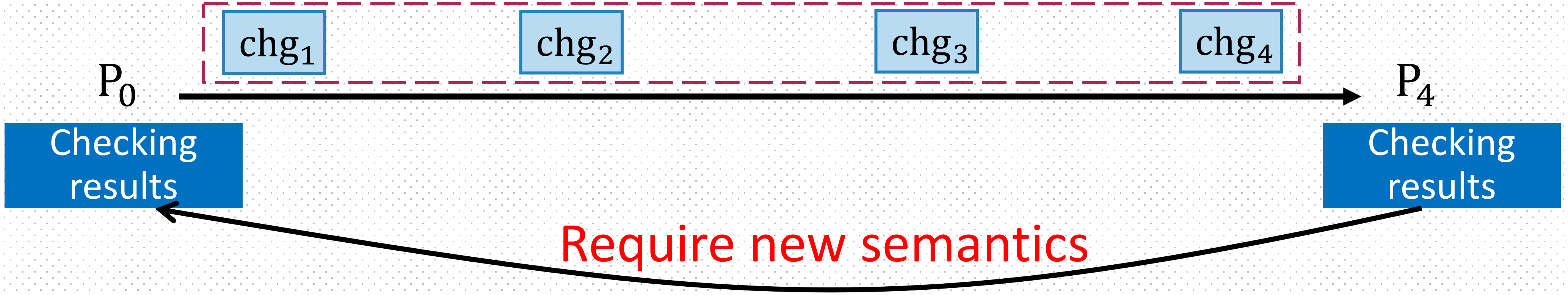
- Incremental checking in PCC



- Incremental checking we need

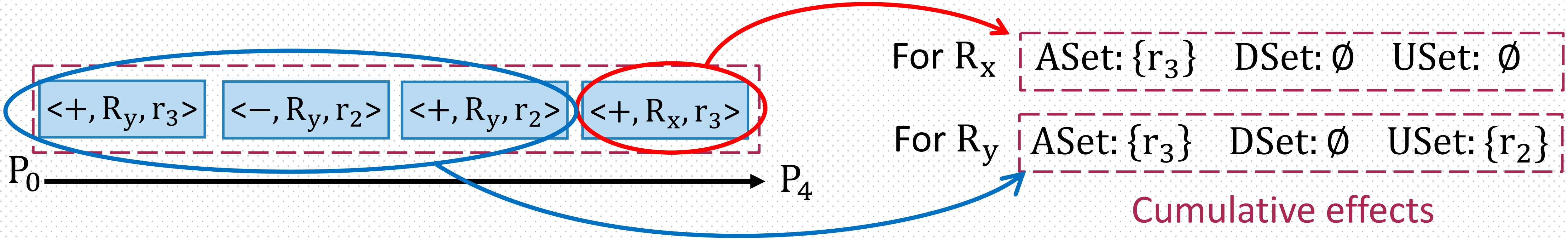
Core of incremental checking: difference of contexts before and after changing

How to know the difference of contexts between P_0 and P_4?



Cumulative Effects Show the Difference

- Accumulate context changes’ effects on contexts in their temporal orders



ASet	DSet	USet
Truly added elements	Truly deleted elements	Updated elements

Extending PCC's Capability to a Group of Changes

- Divide incremental checking into several mutually exclusive cases according to cumulative effects

How to fuse concurrent checking?

ASet	$\emptyset$	$\{a\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{a\}$	$\{a\}$	$\{a\}$
Dset	$\emptyset$	$\emptyset$	$\{d\}$	$\emptyset$	$\{d\}$	$\emptyset$	$\{d\}$	$\{d\}$
USet	$\emptyset$	$\emptyset$	$\emptyset$	$\{u\}$	$\{u\}$	$\{u\}$	$\emptyset$	$\{u\}$

When subformula not affected
(Affected(f) = F)

ASet	$\emptyset$	$\{a\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{a\}$	$\{a\}$	$\{a\}$
Dset	$\emptyset$	$\emptyset$	$\{d\}$	$\emptyset$	$\{d\}$	$\emptyset$	$\{d\}$	$\{d\}$
USet	$\emptyset$	$\emptyset$	$\emptyset$	$\{u\}$	$\{u\}$	$\{u\}$	$\emptyset$	$\{u\}$

When subformula affected
(Affected(f) = T)

$\tau_{\text{partial}}[\forall v \in C(f)]_\alpha =$

(1) $\tau_0[\forall v \in C(f)]_\alpha$, if $\text{Affected}(f) = \text{F}$ and $(ASet = \emptyset$ and $DSet = \emptyset$ and $USet = \emptyset)$.

(2) $\tau_0[\forall v \in C(f)]_\alpha \wedge t_1 \wedge \cdots \wedge t_a$, where $(t_1, \cdots, t_a) = \text{eval}_{\text{entire}}(\tau[f]_{\text{bind}((v,y_j),\alpha)} \mid y_j \in ASet)$,
if $\text{Affected}(f) = \text{F}$ and $(ASet \neq \emptyset$ and $DSet = \emptyset$ and $USet = \emptyset)$.

(3) $\text{T} \wedge \tau_0[f]_{\text{bind}((v,x_1),\alpha)} \wedge \cdots \wedge \tau_0[f]_{\text{bind}((v,x_{n-a-u}),\alpha)} \wedge t_1 \wedge \cdots \wedge t_{a+u} \mid x_i \in C - (ASet \cup USet))$,
where $(t_1, \cdots, t_{a+u}) = \text{eval}_{\text{entire}}(\tau[f]_{\text{bind}((v,y_j),\alpha)} \mid y_j \in ASet \cup USet)$,
if $\text{Affected}(f) = \text{F}$ and $(DSet \neq \emptyset$ or $USet \neq \emptyset)$.

(4) $\text{T} \wedge t_1 \wedge \cdots \wedge t_n$, where $(t_1, \cdots, t_n) = \text{eval}_{\text{partial}}(\tau[f]_{\text{bind}((v,x_i),\alpha)} \mid x_i \in C)$,
if $\text{Affected}(f) = \text{T}$ and $(ASet = \emptyset$ and $DSet = \emptyset$ and $USet = \emptyset)$.

(5) $\text{T} \wedge t_1 \wedge \cdots \wedge t_n$, where $(t_1, \cdots, t_{a+u}) = \text{eval}_{\text{entire}}(\tau[f]_{\text{bind}((v,y_j),\alpha)} \mid y_j \in ASet \cup USet)$
and $(t_{a+u+1}, \cdots, t_n) = \text{eval}_{\text{partial}}(\tau[f]_{\text{bind}((v,x_i),\alpha)} \mid x_i \in C - (ASet \cup USet))$,
if $\text{Affected}(f) = \text{T}$ and $(ASet \neq \emptyset$ or $DSet \neq \emptyset$ or $USet \neq \emptyset)$.

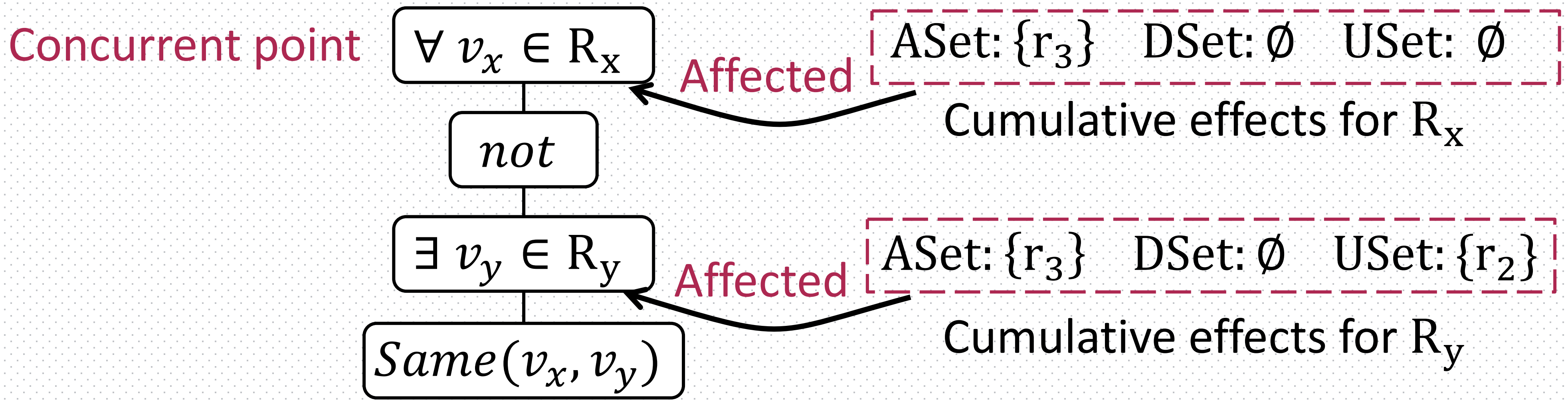
Incremental semantics for universal formula

Concurrent Point Selection

- Concurrent point: indicating where concurrent checking starts
- Selection criterion: the highest affected universal or existential formula

Sufficient workload to be split into units

Units contain similar workloads due to different variable assignments



Adaptive Switching among Different Semantics

When further checking required

$\tau_{\text{partial}}[\forall v \in C(f)]_\alpha =$

(1) $\tau_0[\forall v \in C(f)]_\alpha$, if $\text{Affected}(f) = \text{F}$ and $(ASet = \emptyset$ and $DSet = \emptyset$ and $USet = \emptyset)$.

(2) $\tau_0[\forall v \in C(f)]_\alpha \wedge t_1 \wedge \cdots \wedge t_a$, where $(t_1, \cdots, t_a) = \text{eval}_{\text{entire}}(\tau[f]_{\text{bind}((v,y_j),\alpha)} \mid y_j \in ASet)$, if $\text{Affected}(f) = \text{F}$ and $(ASet \neq \emptyset$ and $DSet = \emptyset$ and $USet = \emptyset)$.

(3) $\text{T} \wedge \tau_0[f]_{\text{bind}((v,x_1),\alpha)} \wedge \cdots \wedge \tau_0[f]_{\text{bind}((v,x_{n-a-u}),\alpha)} \wedge t_1 \wedge \cdots \wedge t_{a+u} \mid x_i \in C - (ASet \cup USet))$, where $(t_1, \cdots, t_{a+u}) = \text{eval}_{\text{entire}}(\tau[f]_{\text{bind}((v,y_j),\alpha)} \mid y_j \in ASet \cup USet)$, if $\text{Affected}(f) = \text{F}$ and $(DSet \neq \emptyset$ or $USet \neq \emptyset)$.

(4) $\text{T} \wedge t_1 \wedge \cdots \wedge t_n$, where $(t_1, \cdots, t_n) = \text{eval}_{\text{partial}}(\tau[f]_{\text{bind}((v,x_i),\alpha)} \mid x_i \in C)$, if $\text{Affected}(f) = \text{T}$ and $(ASet = \emptyset$ and $DSet = \emptyset$ and $USet = \emptyset)$.

(5) $\text{T} \wedge t_1 \wedge \cdots \wedge t_n$, where $(t_1, \cdots, t_{a+u}) = \text{eval}_{\text{entire}}(\tau[f]_{\text{bind}((v,y_j),\alpha)} \mid y_j \in ASet \cup USet)$ and $(t_{a+u+1}, \cdots, t_n) = \text{eval}_{\text{partial}}(\tau[f]_{\text{bind}((v,x_i),\alpha)} \mid x_i \in C - (ASet \cup USet))$, if $\text{Affected}(f) = \text{T}$ and $(ASet \neq \emptyset$ or $DSet \neq \emptyset$ or $USet \neq \emptyset)$.

Incremental semantics

Concurrent or not

$\text{eval}_{\text{entire}}(\tau[f]_{\text{bind}((v,x_i),\alpha)} \mid x_i \in Set) =$

(1) $\tau_{\text{entire}}[f]_{\text{bind}((v,x_1),\alpha)} \parallel \cdots \parallel \tau_{\text{entire}}[f]_{\text{bind}((v,x_s),\alpha)}$, if $\forall v \in C(f)$ is a concurrent point;

(2) $\tau_{\text{entire}}[f]_{\text{bind}((v,x_1),\alpha)}$; $\cdots$; $\tau_{\text{entire}}[f]_{\text{bind}((v,x_s),\alpha)}$, otherwise.

$\text{eval}_{\text{partial}}(\tau[f]_{\text{bind}((v,x_i),\alpha)} \mid x_i \in Set) =$

(1) $\tau_{\text{partial}}[f]_{\text{bind}((v,x_1),\alpha)} \parallel \cdots \parallel \tau_{\text{partial}}[f]_{\text{bind}((v,x_s),\alpha)}$, if $\forall v \in C(f)$ is a concurrent point;

(2) $\tau_{\text{partial}}[f]_{\text{bind}((v,x_1),\alpha)}$; $\cdots$; $\tau_{\text{partial}}[f]_{\text{bind}((v,x_s),\alpha)}$, otherwise.

Concurrent semantics

$\tau_{\text{entire}}[\forall v \in C(f)]_\alpha =$
$\text{T} \wedge \tau_{\text{entire}}[f]_{\text{bind}((v,x_1),\alpha)} \wedge \cdots \wedge \tau_{\text{entire}}[f]_{\text{bind}((v,x_n),\alpha)} | x_i \in C$

Full semantics

How to further check

Two Faced Problems

- What-To-Check
  - Which workloads can be checked together for enlarging workload? ✓
- How-To-Check
  - How to correctly conduct fusion checking after enlarging workload? ✓

Theoretical Guarantee

- What-To-Check
  - Which workloads can be checked together for enlarging workload? ✓ ✓

WHAT-Correctness Theorem

Given any consistency constraint and associated context pool, INFUSE produces the same result for its arranged valid context changes, no matter it checks these changes as a whole or individually.

- How-To-Check
  - How to correctly conduct fusion checking after enlarging workload? ✓ ✓

HOW-Correctness Theorem

Given any consistency constraint and associated context pool, INFUSE produces the same result by its check fusion semantics, as existing constraint checking techniques do.

Evaluation

- Research Questions
  - RQ1 (Motivation): How do existing constraint checking techniques behave when handling large-volume dynamic contexts? (already shown earlier)
  - RQ2 (Effectiveness): How effective is INFUSE in constraint checking for detecting context inconsistencies, as compared with existing techniques?
  - RQ3 (Practical Usage): How effective is INFUSE in constraint checking under real-life settings?

Experimental Design and Set Up

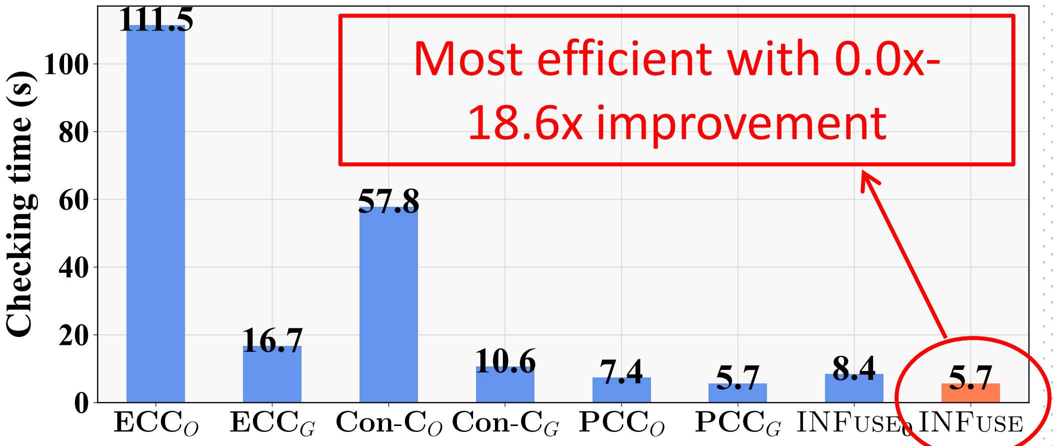
- Subjects
  - SmartCity application with 4.3 million vehicle data (e.g., GPS data, speed, direction) and 48 consistency constraints (also used in existing work[9-12] for evaluation)

- Workloads
  - Three distinct hour-based groups of data with light (311,240 changes), median (843,686 changes) and heavy (1,664,900 changes) workloads

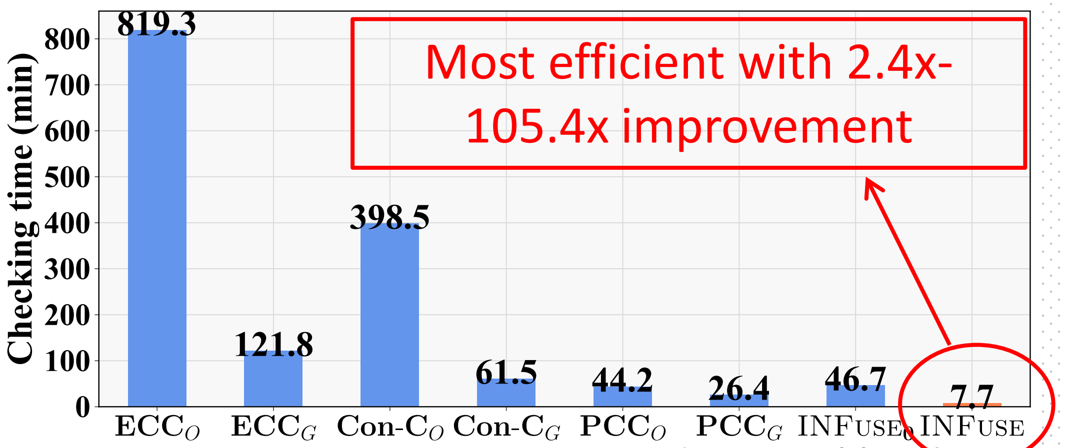
- Techniques for comparison
  - Two versions in our work: INFUSE (elite version), INFUSE_0 (brute-force version)
  - Existing techniques and their improved versions: ECC_O[5] , ECC_G, Con-C_O[11] , Con-C_G, PCC_O[10] , PCC_G

RQ2 (Effectiveness)

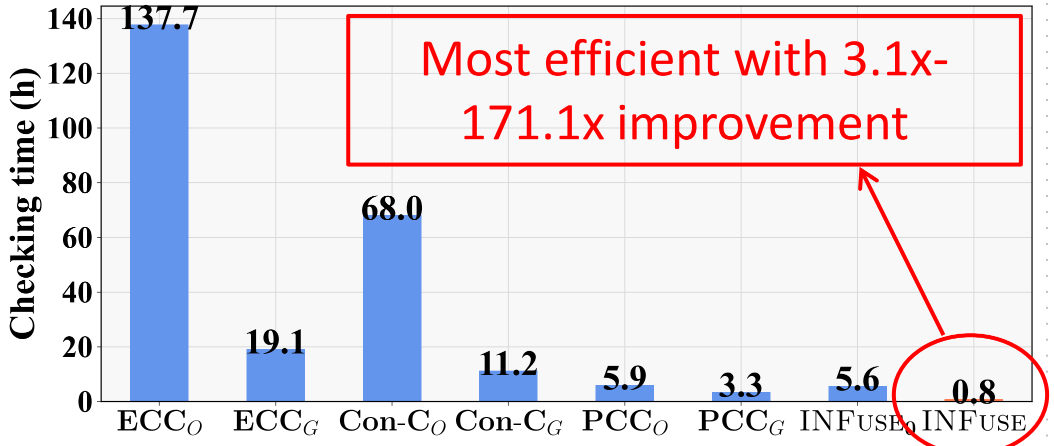
- Checking time comparison for all techniques on all workloads



Time comparison on light workload



Time comparison on median workload

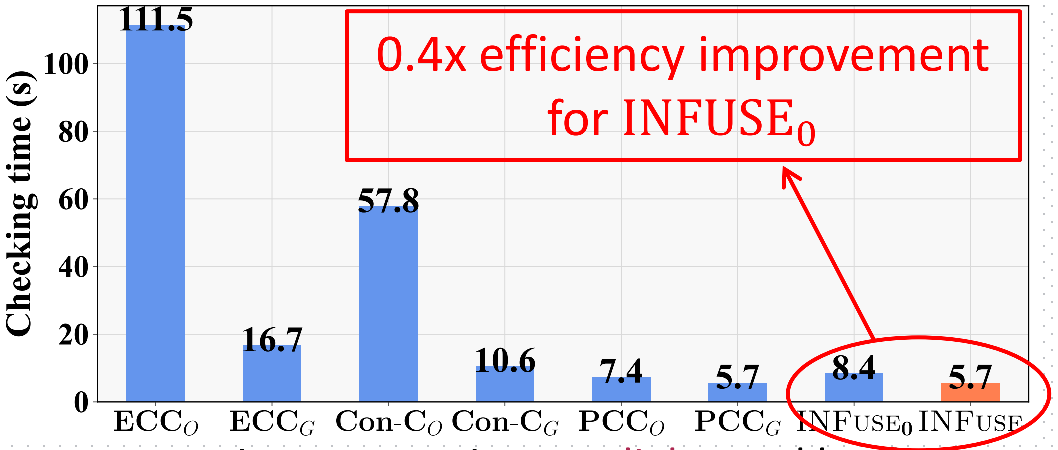


Time comparison on heavy workload

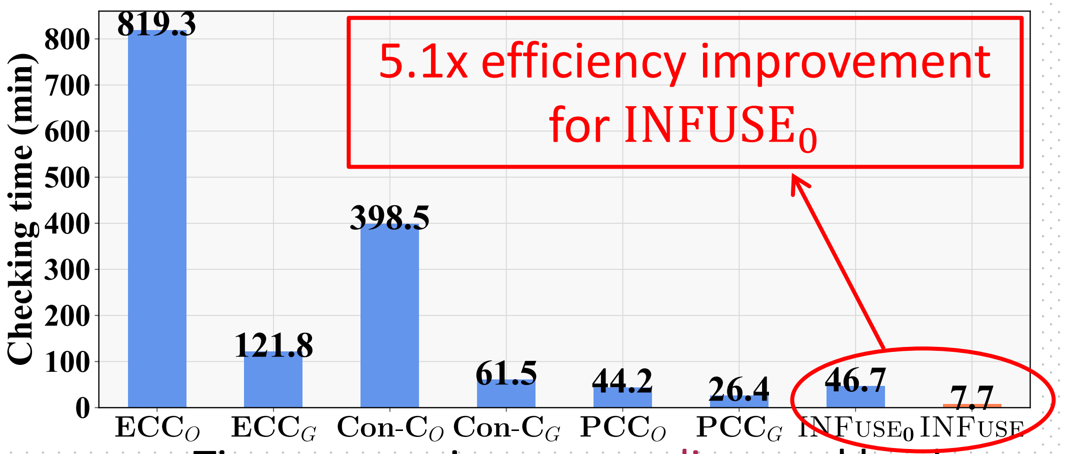
INFUSE was **the most efficient** technique on all workloads

RQ2 (Effectiveness)

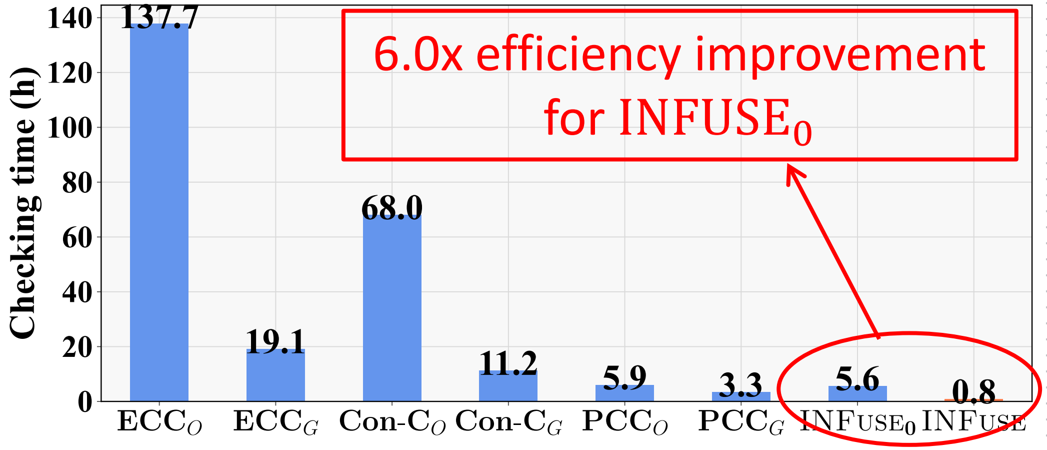
- Checking time comparison for all techniques on all workloads



Time comparison on light workload



Time comparison on median workload



Time comparison on heavy workload

Difference between INFUSE and INFUSE_0 **was large and kept increasing**

Valid Context Change Groups

Fusion Soundness

RQ3 (Practical usage)

- Simulate real-life settings according to real timestamps

False negative rate

False positive rate

Workload	Checking techniques	Oracle incs (#)	Reported incs/* (#)	T_{cost}(s)	R_{FN}(%)	R_{FP}(%)
Light	ECC_O	3,254	3,254	128.6	0.0%	0.0%
	$Con\text{-}C_O$		3,254	54.3	0.0%	0.0%
	PCC_O		3,254	12.8	0.0%	0.0%
	ECC_G		3,254	26.9	0.0%	0.0%
	$Con\text{-}C_G$		3,254	16.9	0.0%	0.0%
	PCC_G		3,254	13.1	0.0%	0.0%
	INFUSE_0		3,254	13.1	0.0%	0.0%
	INFUSE		3,254	10.8	0.0%	0.0%

All techniques reported correct checking results, but INFUSE took **the least time**

RQ3 (Practical usage)

- False negative/positive rates are more crucial since they reflects correctness

The less, the better

Workload	Checking techniques	Oracle incs (#)	Reported incs/* (#)	T_{cost}(s)	$R_{FN}(\%)$	$R_{FP}(\%)$
Median	ECC_O	21,436	8,647/694*	3,850.9	96.8%	92.0%
	$Con\text{-}C_O$		14,209/897*	3,593.9	95.8%	93.7%
	PCC_O		20,942/19,369*	1,513.7	9.6%	7.5%
	ECC_G		20,412/1,415*	3,588.4	93.4%	93.1%
	$Con\text{-}C_G$		20,779/19,293*	1,950.8	10.0%	7.2%
	PCC_G		21,377/19,414*	1,099.7	9.4%	9.2%
	INFUSE_0		20,922/19,371*	1,588.5	9.6%	7.4%
	INFUSE		21,436	456.6	0.0%	0.0%
Heavy	ECC_O	29,642	4,934/392*	4,032.1	98.7%	92.1%
	$Con\text{-}C_O$		6,611/463*	3,748.2	98.4%	93.0%
	PCC_O		22,574/1,028*	3,410.8	96.5%	95.5%
	ECC_G		14,617/801*	3,574.8	97.3%	94.5%
	$Con\text{-}C_G$		20,824/957*	3,375.5	96.8%	95.4%
	PCC_G		29,115/1,178*	3,594.4	96.0%	96.0%
	INFUSE_0		22,302/1,013*	3,463.2	96.6%	95.5%
	INFUSE		29,642	2954.6	0.0%	0.0%

INFUSE still took **the least time**

INFUSE still **reported correct checking results** while others suffered from varying degrees of quality problems

Effective under real-life settings

Conclusion and Future Work

- Work summary
    - Addressed what-to-check and how-to-check problems of fusion checking with theoretical guarantee
    - 18.6x–171.1x speed up to existing techniques with quality guarantees

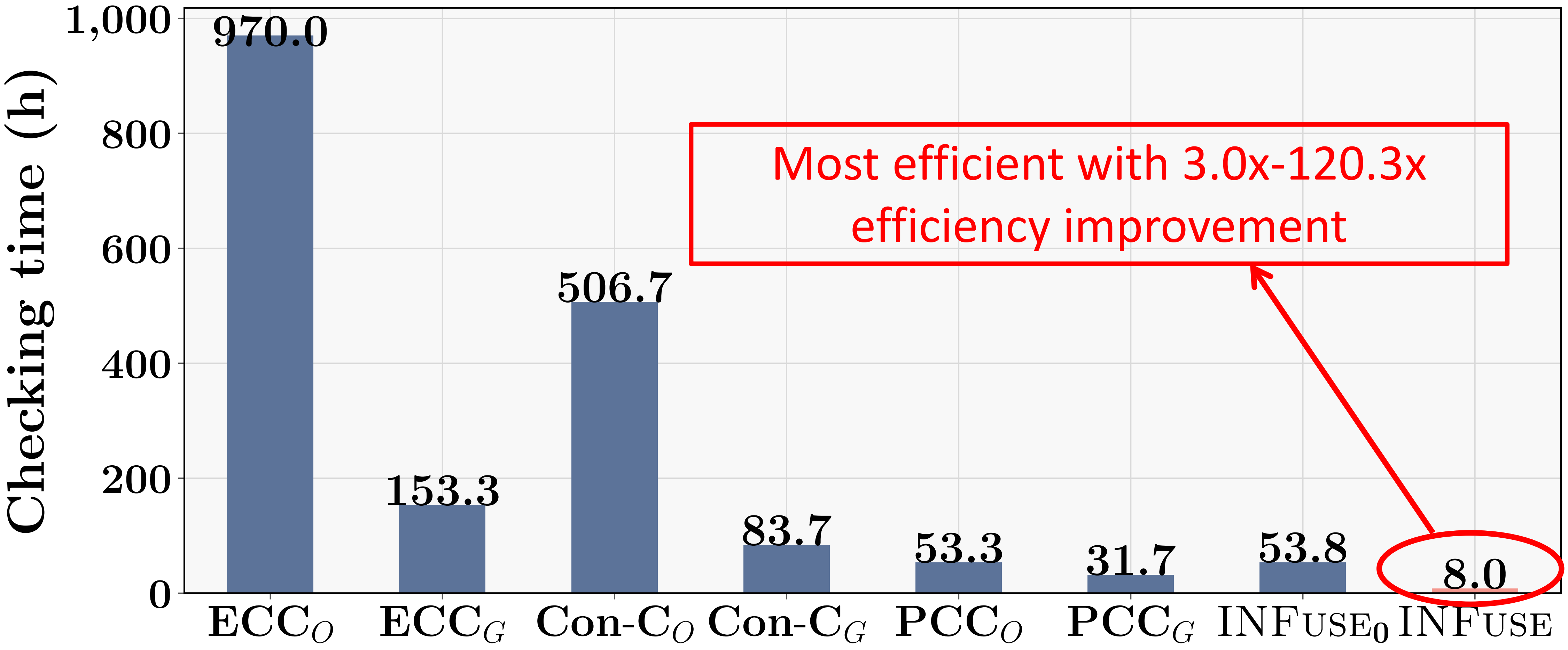
- Future work
    - Less conservative grouping strategy
    - Adaptive concurrency control

Thank you!

Comments are welcome!

Email: zly@smail.nju.edu.cn

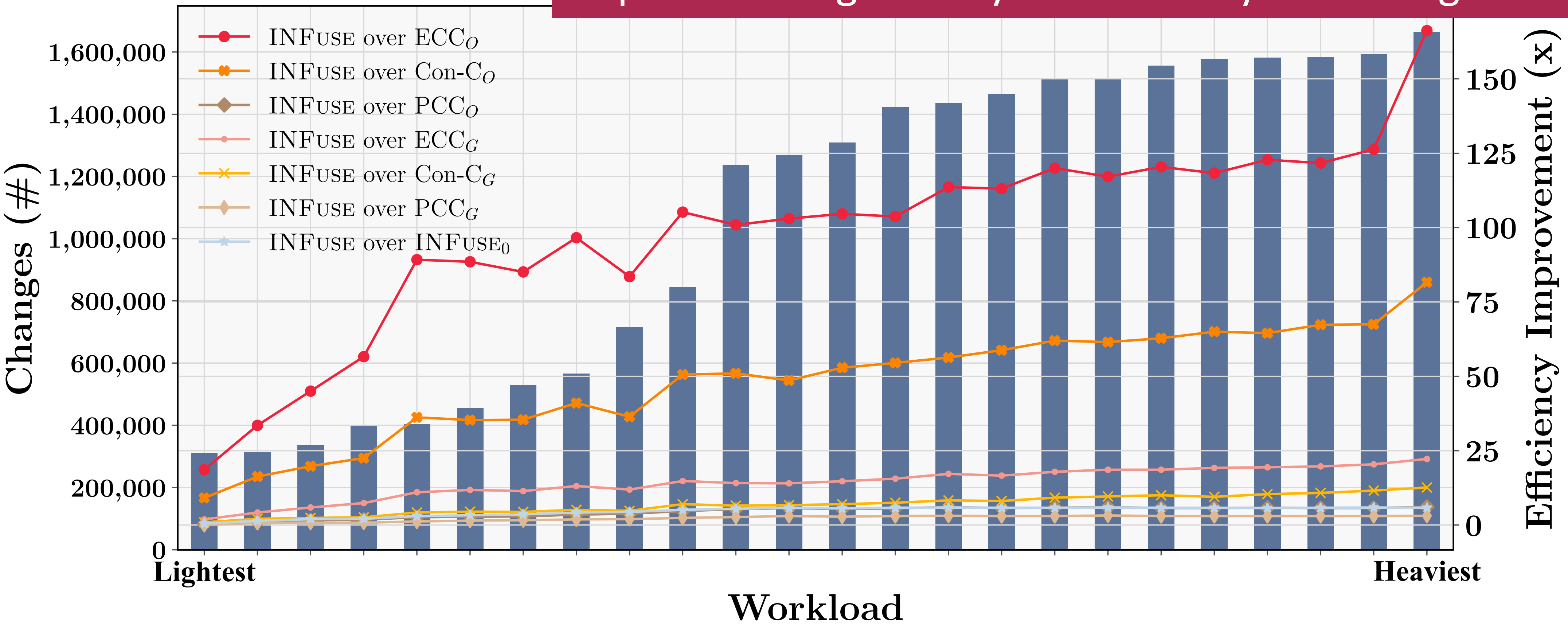
Experimental Results of all 24 hours data (1)



Time comparison for all 24-hour data

Experimental Results of all 24 hours data (2)

With the growth of workload, INFUSE's efficiency improvement generally hold a stably increasing trend.



INFUSE's efficiency improvement over existing checking techniques on 24 hour-based groups (sorted by increasing workloads)

Experimental Results of all 24 hours data (3)

The less red color the better

Checking techniques	Metrics	Grou																					
		0	1	2	3	4	5	6	7	8	9	10	11	1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	23
ECC_O	T_{cost}(min)	57.5	33.9	14.8	4.4	2.1	2.3	18.4	59.1	61.7	64.2	66.8	62.3	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
	R_{FN}(%)	95.9%	0.0%	0.0%	0.0%	0.0%	0.0%	84.8%	76.9%	96.6%	96.8%	98.5%	98.5%	98.1%	98.4%	98.6%	98.7%	98.4%	98.7%	98.6%	98.6%	98.4%	98.6%	98.5%	98.3%
	R_{FP}(%)	95.6%	0.0%	0.0%	0.0%	0.0%	0.0%	84.8%	73.4%	93.0%	92.0%	91.6%	92.7%	93.0%	93.7%	92.5%	92.3%	91.0%	92.1%	91.7%	92.0%	91.9%	91.9%	91.7%	93.8%
Con-C_O	T_{cost}(min)	29.9	13.0	5.8	1.8	0.9	1.0	7.1	42.7	60.2	59.9	61.0	63.1	60.7	59.8	62.5	60.7	64.3	62.5	63.5	63.3	61.5	63.3	60.8	60.3
	R_{FN}(%)	95.2%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	25.3%	94.7%	95.8%	97.9%	98.0%	97.9%	97.8%	98.3%	98.4%	98.0%	98.4%	98.2%	98.1%	98.2%	98.1%	98.1%	97.8%
	R_{FP}(%)	95.2%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	24.1%	93.0%	93.7%	93.0%	93.6%	94.3%	94.5%	93.7%	92.9%	92.7%	93.0%	92.8%	92.8%	93.4%	93.0%	93.3%	95.0%
PCC_O	T_{cost}(min)	3.4	1.9	0.8	0.4	0.2	0.2	0.8	5.8	19.4	25.2	56.6	57.0	58.5	58.7	56.9	56.7	57.0	56.8	56.9	57.2	56.4	56.4	56.6	58.4
	R_{FN}(%)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	9.6%	96.4%	96.5%	96.3%	94.4%	96.4%	96.6%	96.5%	96.5%	96.3%	96.4%	96.3%	96.2%	96.1%	96.3%
	R_{FP}(%)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	7.5%	95.9%	96.2%	96.2%	94.2%	95.7%	95.6%	95.8%	95.5%	95.4%	95.5%	95.6%	95.3%	95.6%	96.2%
ECC_G	T_{cost}(min)	11.0	5.3	2.2	0.9	0.4	0.5	2.7	16.4	55.1	59.8	57.7	57.5	58.3	58.6	57.2	58.4	58.2	59.6	58.8	57.8	57.2	56.9	57.6	58.0
	R_{FN}(%)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	64.7%	93.4%	96.8%	97.1%	96.4%	97.0%	97.2%	97.3%	96.8%	97.3%	97.0%	97.0%	96.9%	97.2%	96.9%	96.7%
	R_{FP}(%)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	64.5%	93.1%	95.3%	95.9%	95.9%	96.4%	95.3%	94.9%	94.6%	94.5%	94.6%	94.7%	95.2%	95.2%	95.3%	96.2%
Con-C_G	T_{cost}(min)	4.6	2.2	0.9	0.5	0.3	0.3	1.1	6.8	24.3	32.5	57.3	57.2	59.3	58.6	56.3	55.7	56	56.3	56.6	56.1	56.4	56.4	57.2	59.7
	R_{FN}(%)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	10.0%	96.5%	96.6%	96.5%	96.3%	96.8%	96.7%	96.5%	96.8%	96.2%	96.4%	96.6%	96.4%	96.2%	96.7%
	R_{FP}(%)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	7.2%	96.2%	96.4%	96.2%	96.2%	96.2%	95.6%	95.8%	95.4%	95.2%	95.6%	96.0%	95.6%	95.7%	96.7%
PCC_G	T_{cost}(min)	2.6	1.5	0.7	0.3	0.2	0.2	0.6	4.1	13.0	18.3	59.9	60.0	52.2	54.7	59.9	59.9	60.2	59.9	60.0	59.9	59.7	60.1	59.9	54.3
	R_{FN}(%)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	9.4%	96.1%	96.2%	95.4%	67.1%	95.6%	96.2%	95.9%	96.0%	95.7%	95.8%	95.9%	95.6%	95.7%	95.2%
	R_{FP}(%)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	9.2%	96.1%	96.2%	95.4%	67.0%	95.6%	96.0%	95.9%	96.0%	95.7%	95.6%	95.8%	95.4%	95.5%	95.1%
$INFUSE_O$	T_{cost}(min)	3.5	2.0	1.0	0.4	0.2	0.2	0.9	6.2	21.0	26.5	57.0	57.2	58.7	58.0	57.2	57.0	57.5	57.7	56.9	56.8	57.0	57.2	57.2	58.6
	R_{FN}(%)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	9.6%	96.5%	96.4%	96.2%	94.2%	96.4%	96.7%	96.4%	96.6%	96.3%	96.4%	96.5%	96.3%	96.2%	96.4%
	R_{FP}(%)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	7.4%	95.9%	96.1%	96.1%	93.8%	95.7%	95.6%	95.7%	95.5%	95.4%	95.6%	95.7%	95.5%	95.6%	96.3%
INFUSE	T_{cost}(min)	0.9	0.8	0.4	0.2	0.2	0.2	0.3	1.6	4.0	7.6	28.7	27.5	16.2	19.7	39.7	42.0	38.5	49.2	43.6	38.8	32.7	34.8	27.8	16.1
	R_{FN}(%)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	1.8%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
	R_{FP}(%)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	1.8%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

[dark blue], [light blue], [pink] and [red] represent the false negative rate or the false positive rate is 0.0%, (0.0%, 10.0%), [10.0%, 90.0%], and (90.0%, 100.0%] respectively.

INFUSE achieved zero false negative and positive for almost all groups