Data Flow Analysis of Asynchronous Systems using Infinite Abstract Domains

Komondoor V Raghavan, *Indian Institute of Science, Bengaluru*

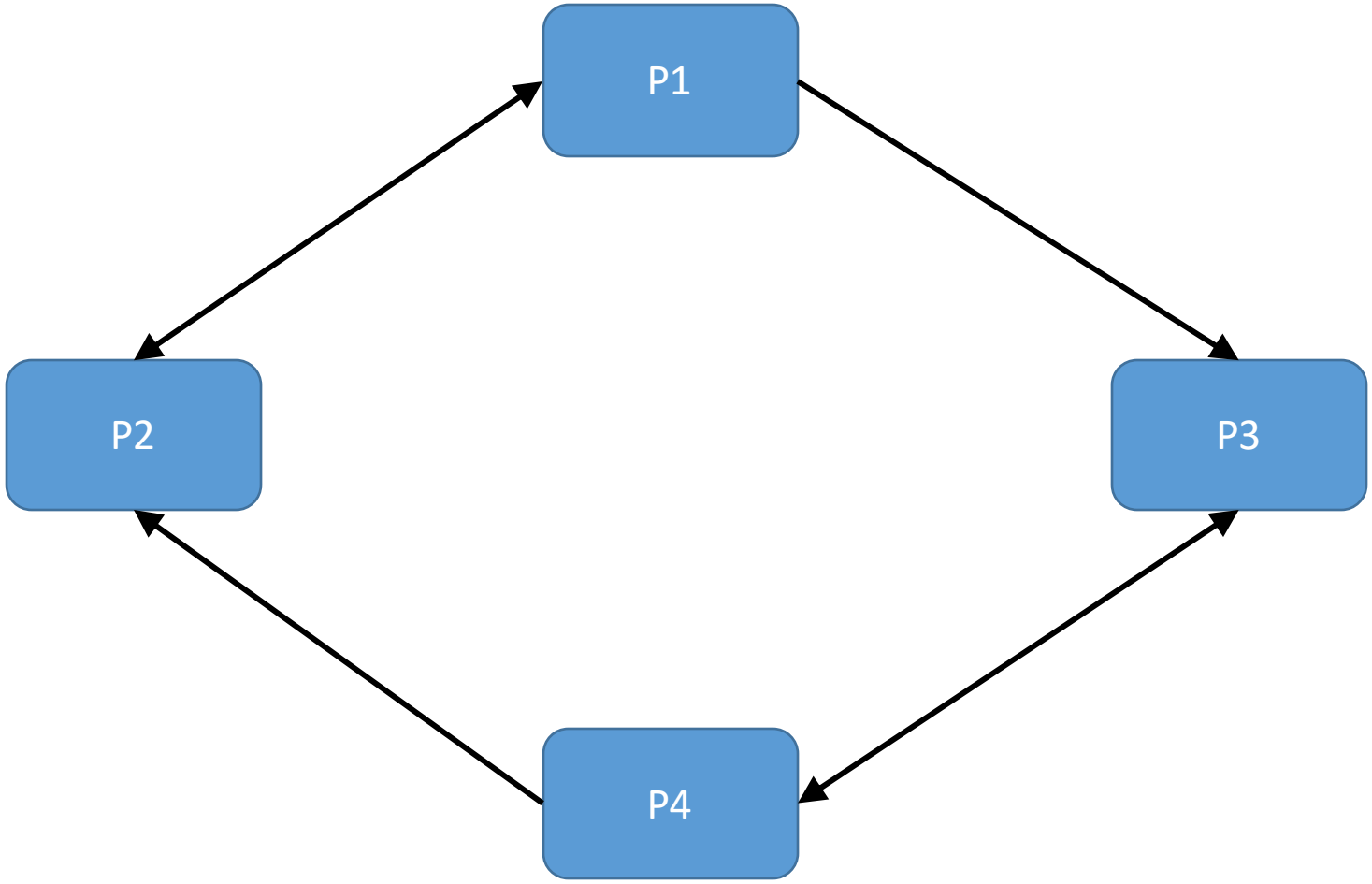
with

Snigdha Athaiya*, *Siemens Technology, India*

K Narayan Kumar, *Chennai Mathematical Institute, Chennai*

* Work partly done while with Indian Institute of Science

Distributed systems



Multiple Processes

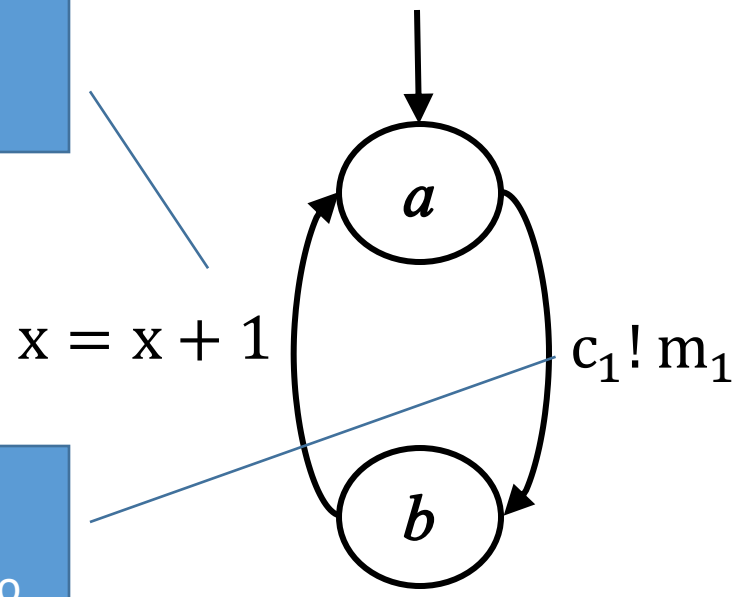
Parallel and independent execution

Processes can communicate with each other

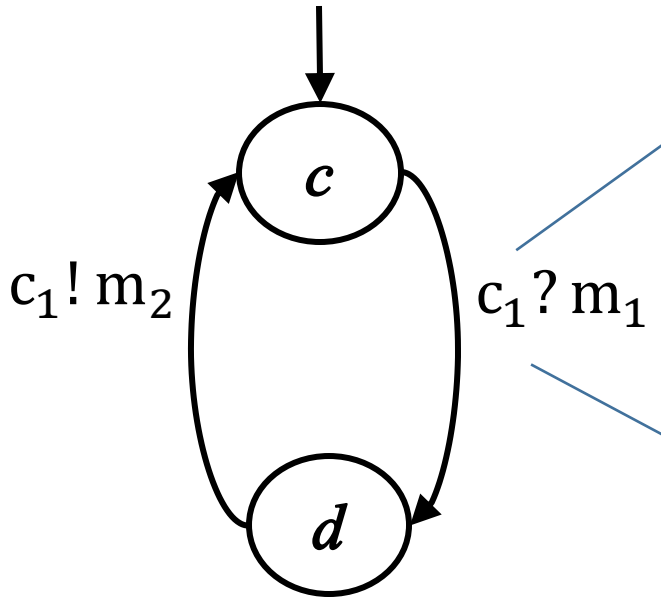
Asynchronous System in Detail

Process 1 accesses the variable x

Sending of message m_1 to channel c_1



Process 1

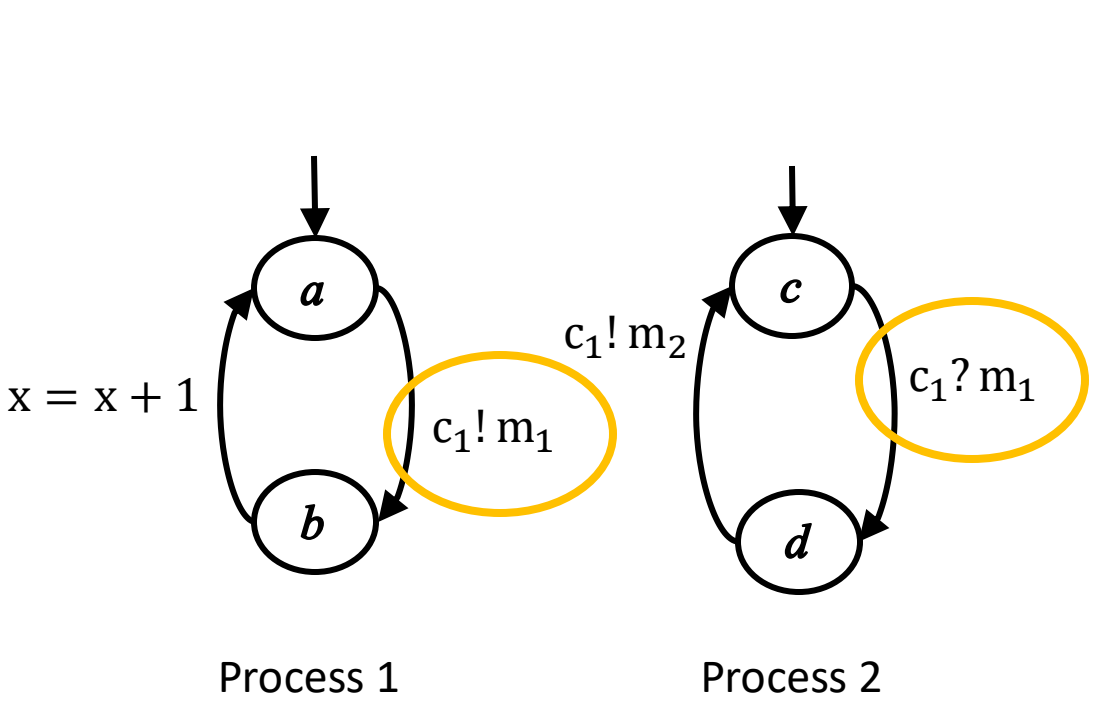


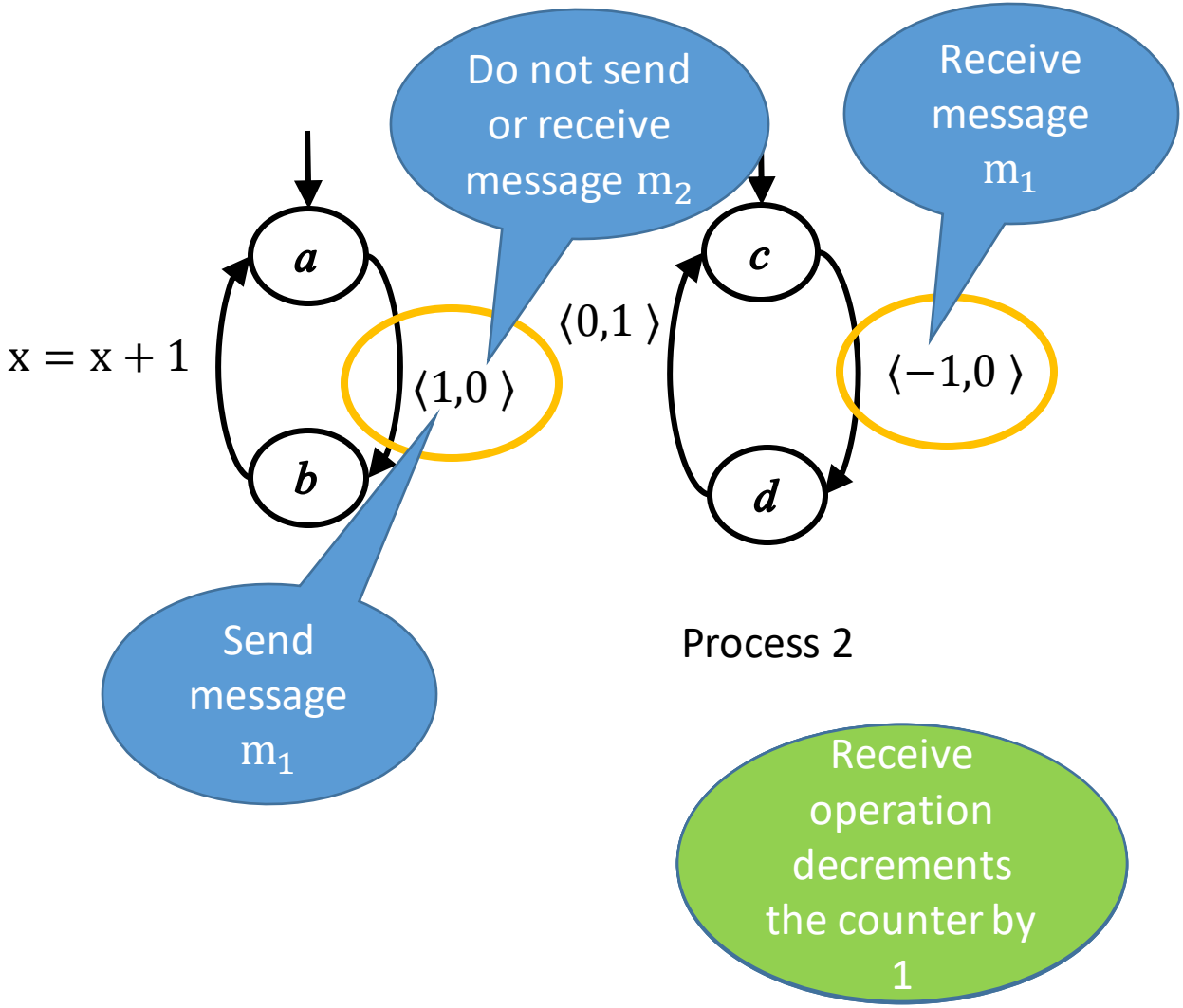
Process 2

- c_1 is a channel
- m_1, m_2 are messages

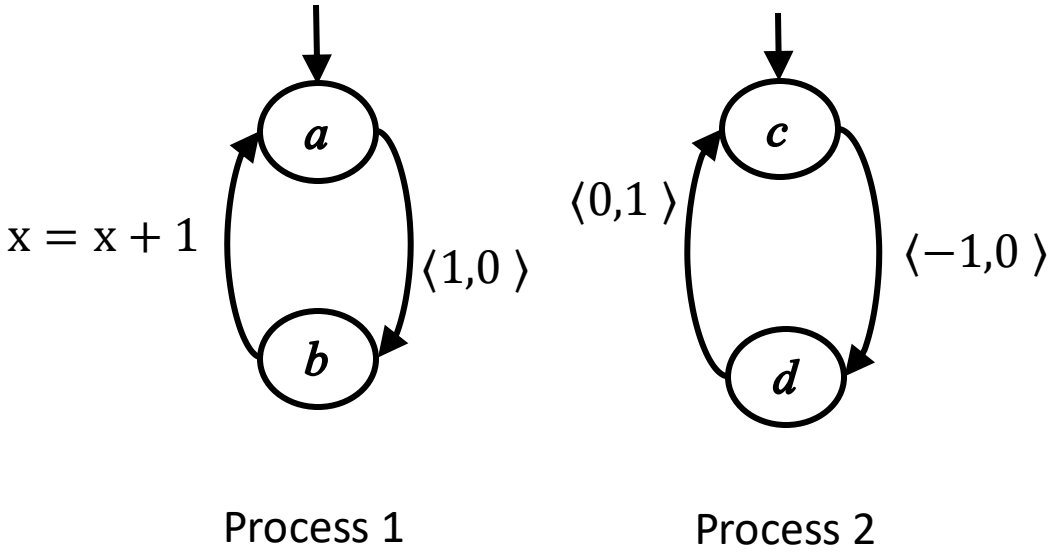
Receive of message m_1 from the head of channel c_1

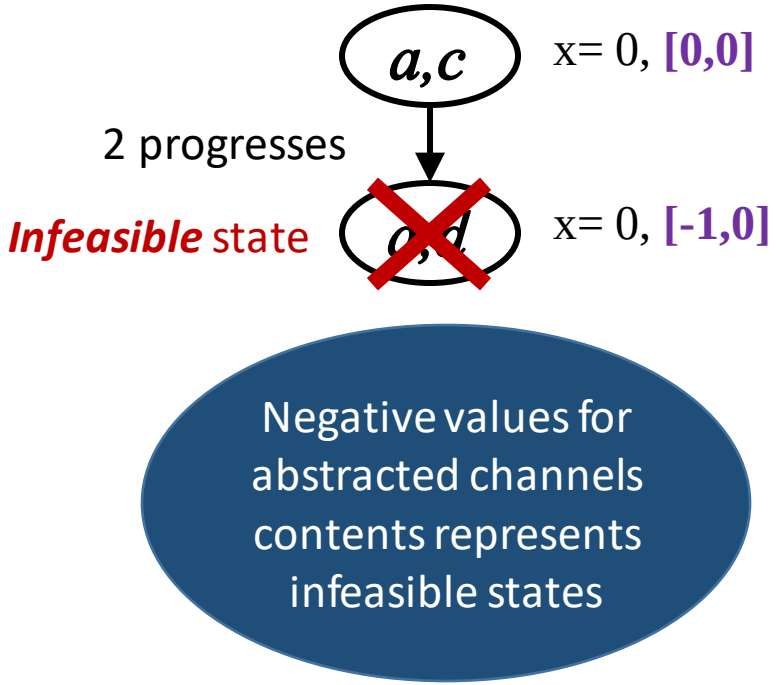
Unordered Channel Abstraction



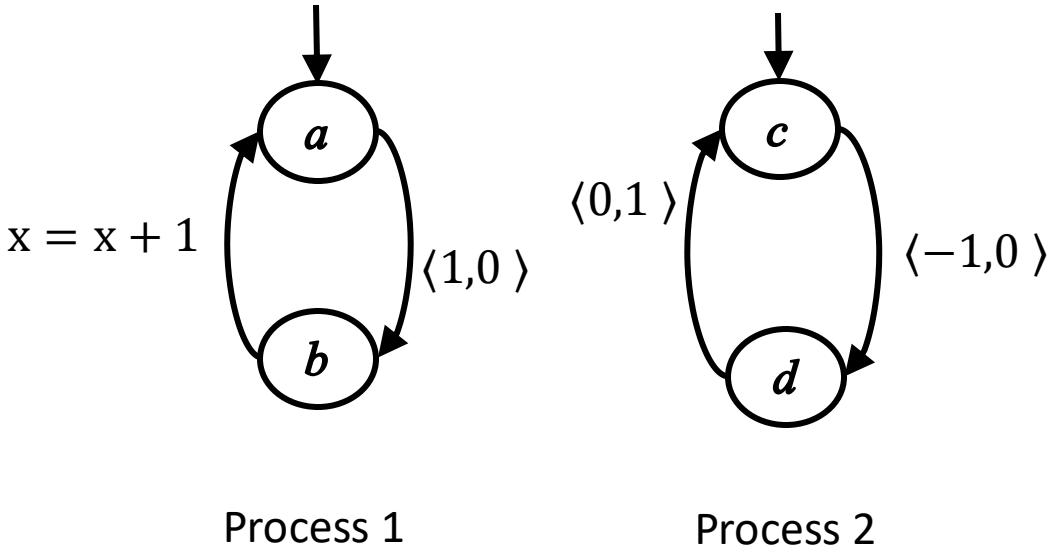


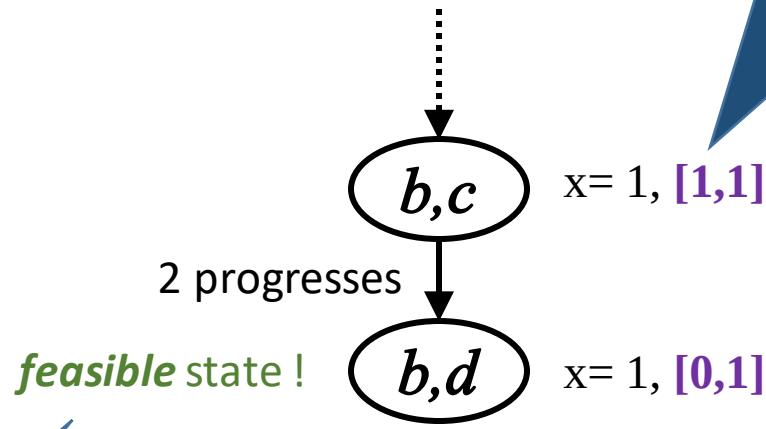
Semantics


x = x + 1
a
b
⟨1,0 ⟩
⟨0,1 ⟩
c
d
⟨−1,0 ⟩
Process 1
Process 2


a,c
x= 0, [0,0]
2 progresses
Infeasible state
c,d
x= 0, [-1,0]
Negative values for abstracted channels contents represents infeasible states

Semantics



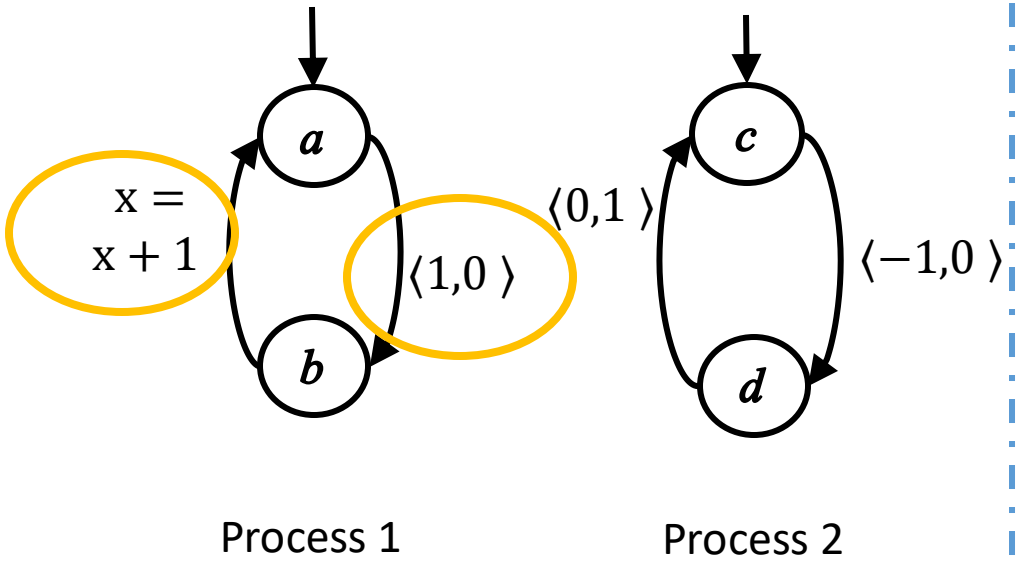


abstraction of concrete config $\langle x= 1, c_1: [m_2, m_1]\rangle$

feasible state !

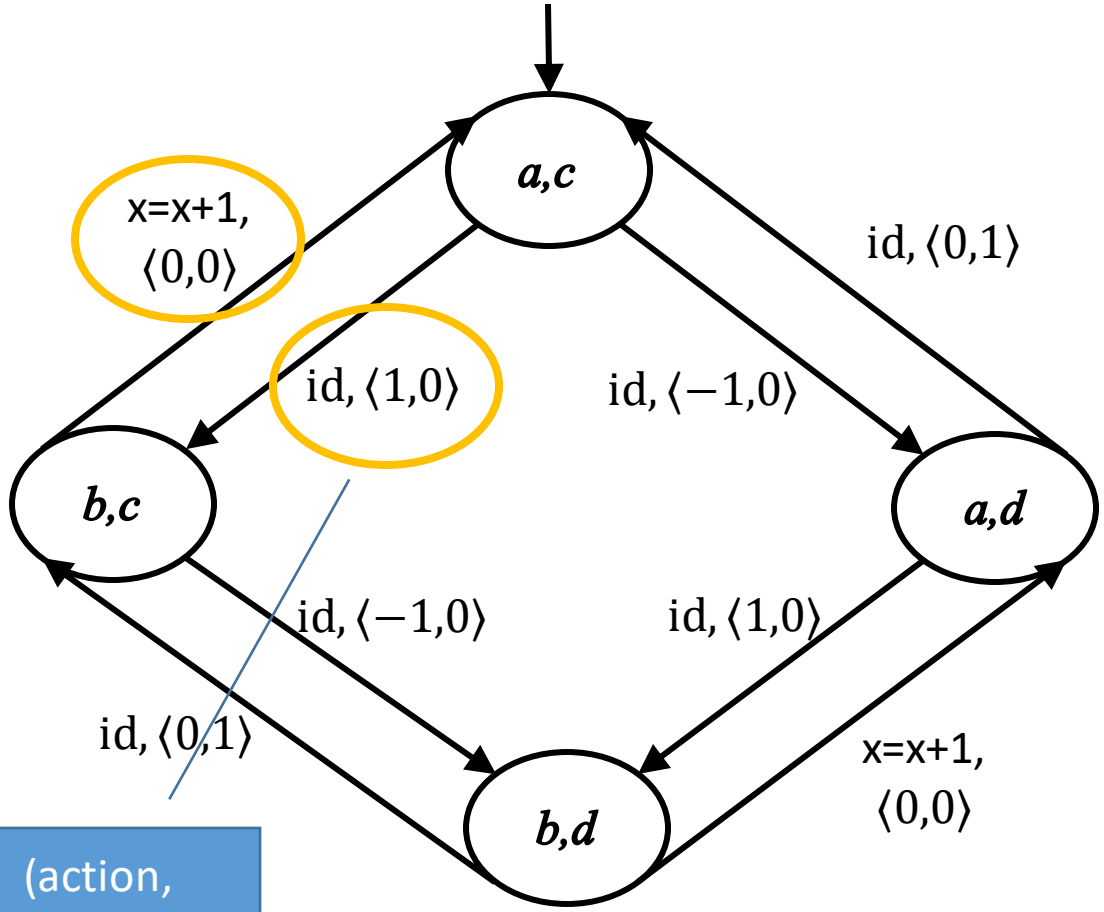
But this is acceptable in literature

VASS-Control Flow Graph(VCFG) Construction



Process 1 Process 2

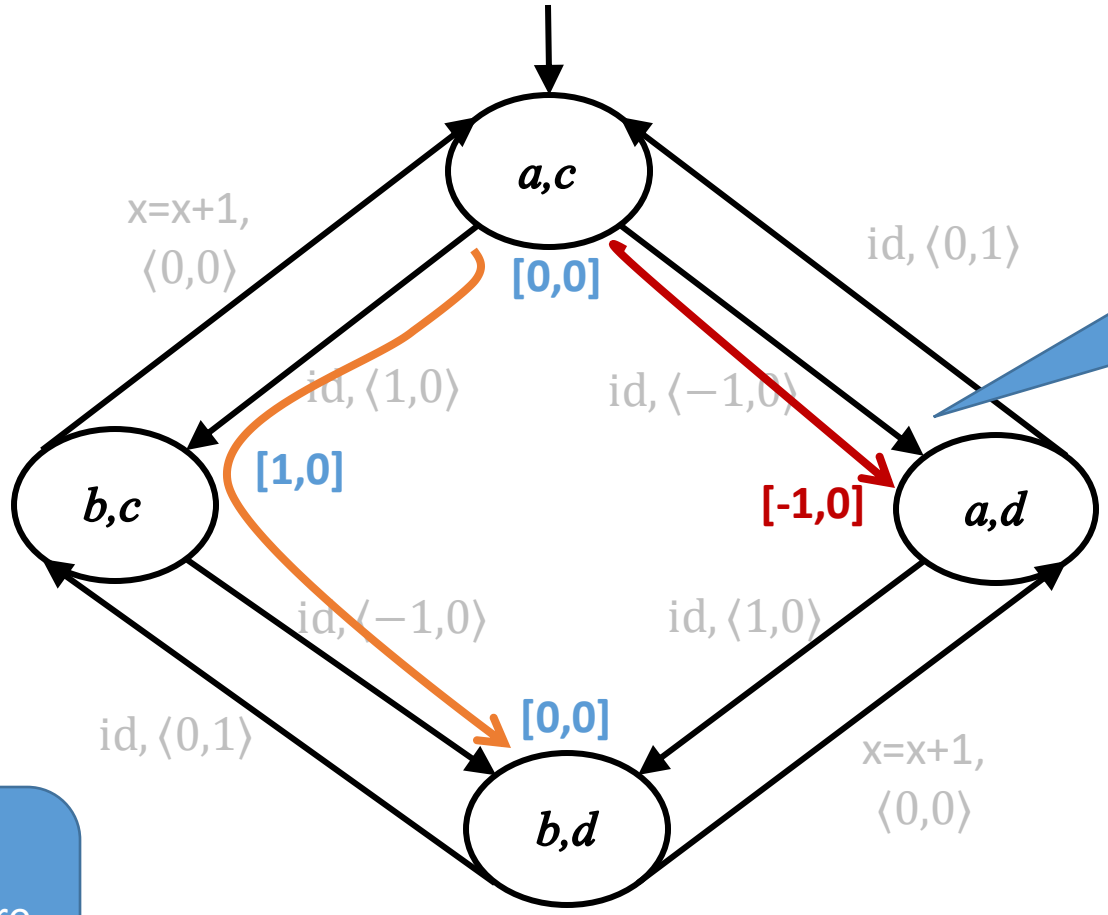
Asynchronous System with unordered channels



(action, vector) for each edge

VCFG

Feasible/Infeasible Paths in VCFG

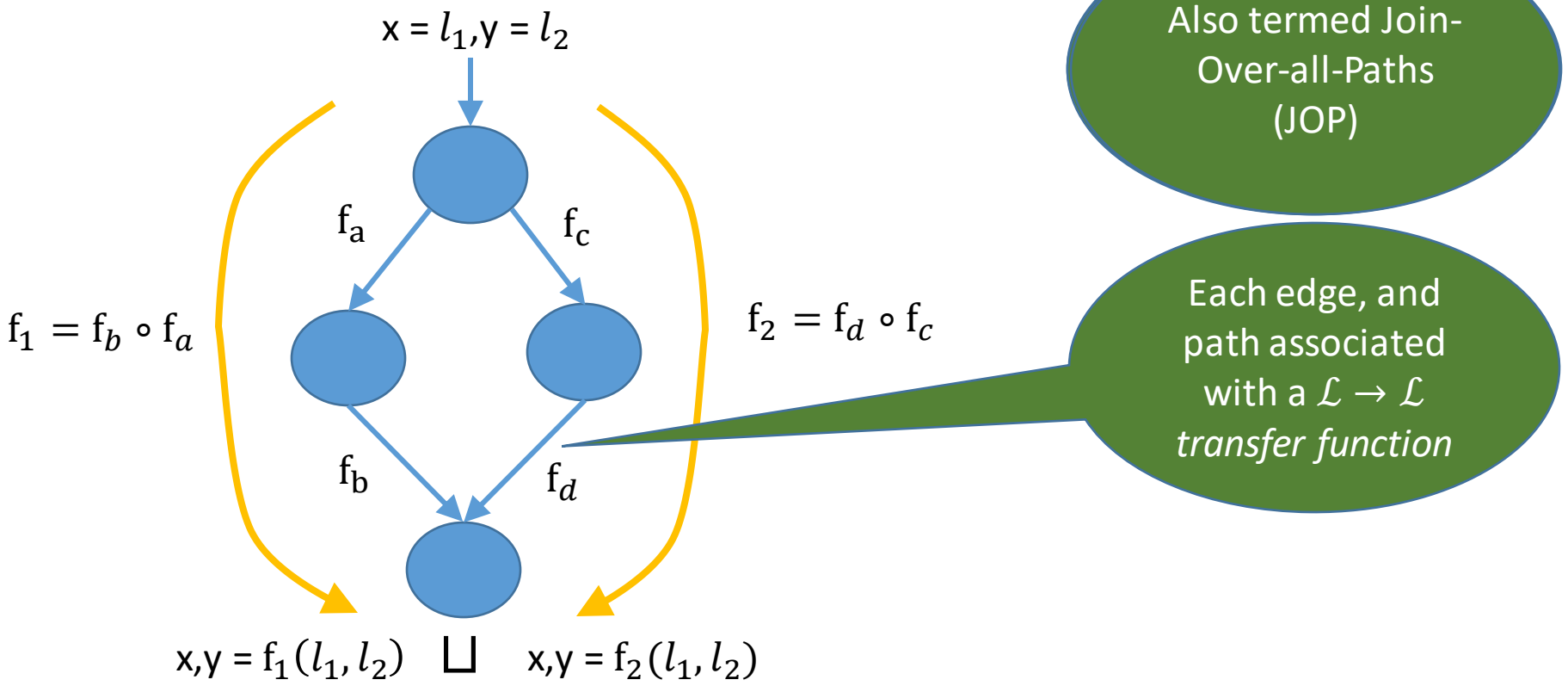


Paths with negative abstract channel contents for any prefix are infeasible

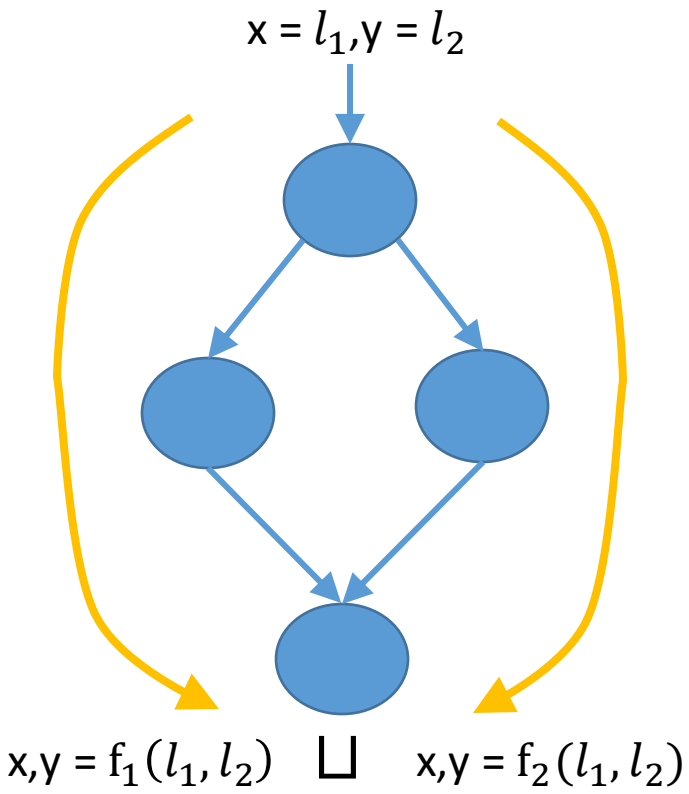
All abstract channel contents are non-negative for all prefixes

VCFG

Data Flow Analysis



Data Flow Analysis of VCFG



Need to compute Join-Over-all-Feasible-Paths (JOFP)

Problem Statement

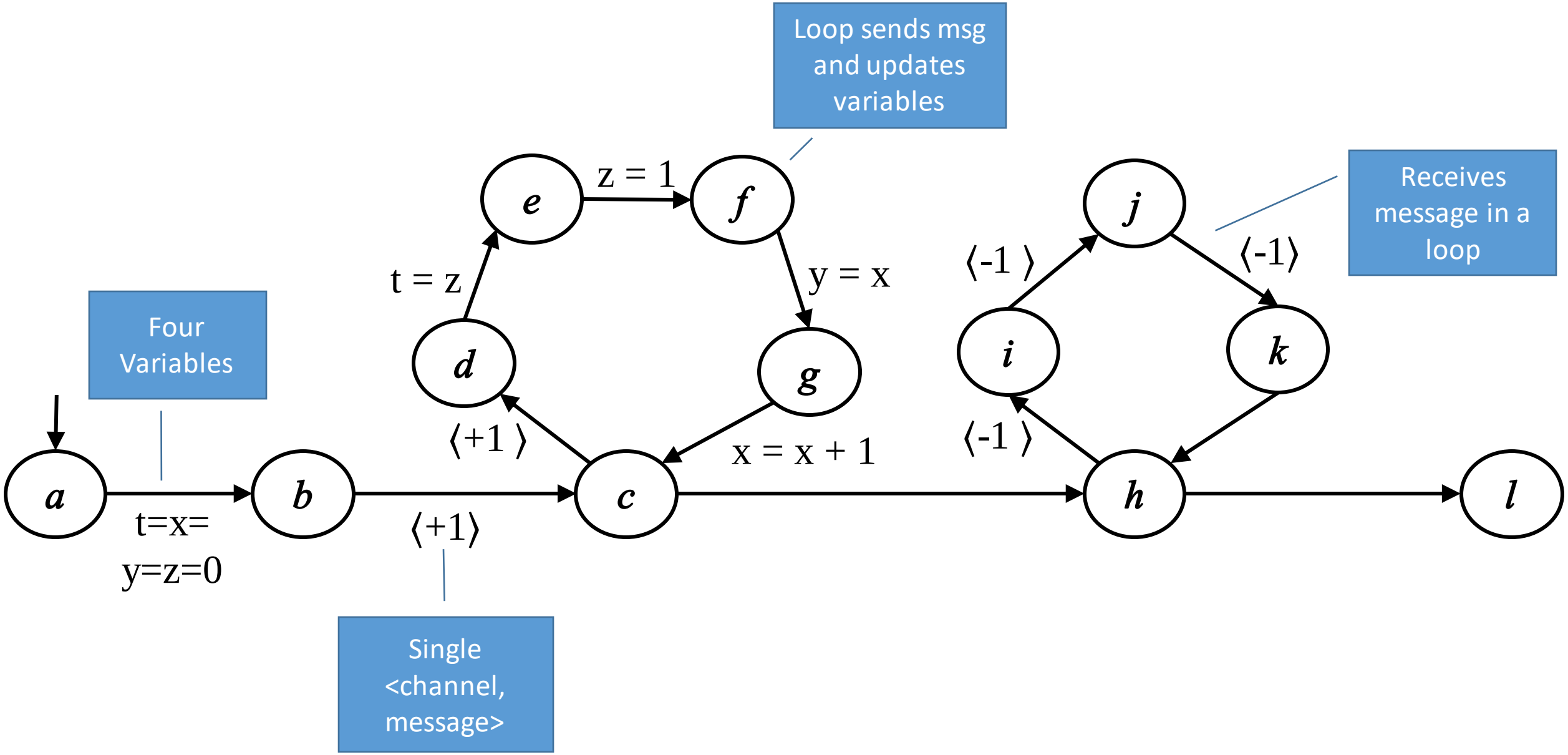
Given an asynchronous system VCFG, and an arbitrary given finite or infinite abstract domain $\mathcal{L}$, an initial value $l_0 \in \mathcal{L}$, initial node v_0, and a target node v, compute the precise join-over-all-feasible-paths(JOFP) for v, where JOFP is defined as:

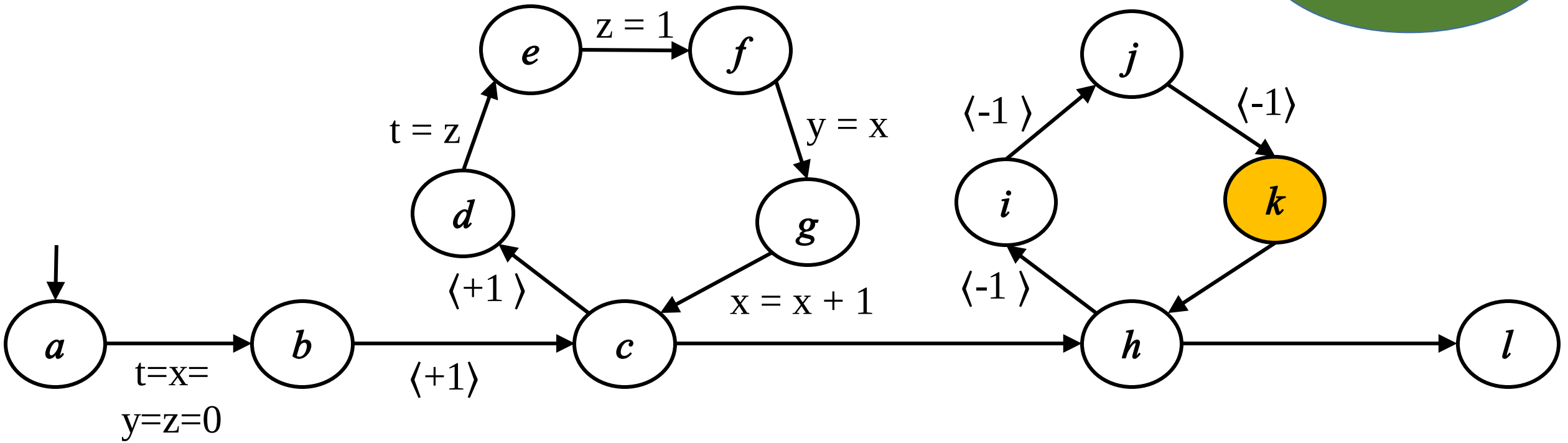
$$\text{JOFP}(v) = \bigsqcup_{\substack{p = v_0 \to^* v, \\ p \text{ is feasible}}} (ptf(p)(l_0))$$

Contributions

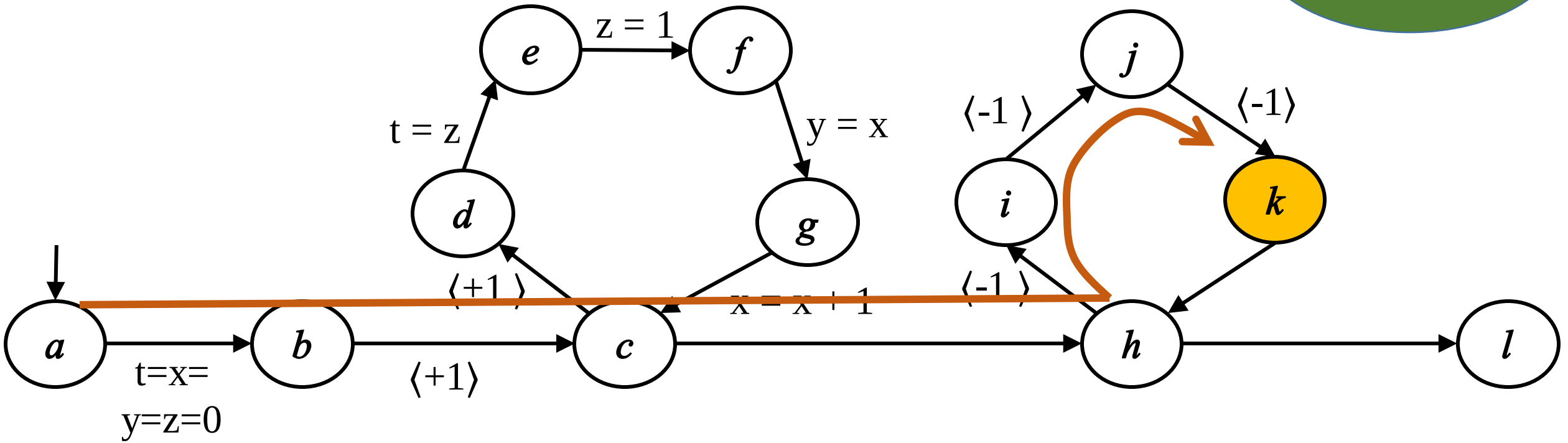
- A sound and precise *Backward DFAS Algorithm,* which admits a class of infinite abstract domains
  - Computes precise JOFP
- A sound *Forward DFAS Algorithm,* admitting a broader class of infinite abstract domains
  - In general, computes conservative over-approximation of JOFP
- Both algorithms use unordered channel abstraction
- In the presentation, we discuss Backward DFAS as it is conceptually more interesting.

Illustrative Example


Loop sends msg and updates variables
Receives message in a loop
Four Variables
Single <channel, message>
a
b
c
d
e
f
g
h
i
j
k
l
z = 1
t = z
y = x
x = x + 1
⟨+1⟩
⟨-1⟩
⟨-1⟩
⟨-1⟩
t=x=
y=z=0

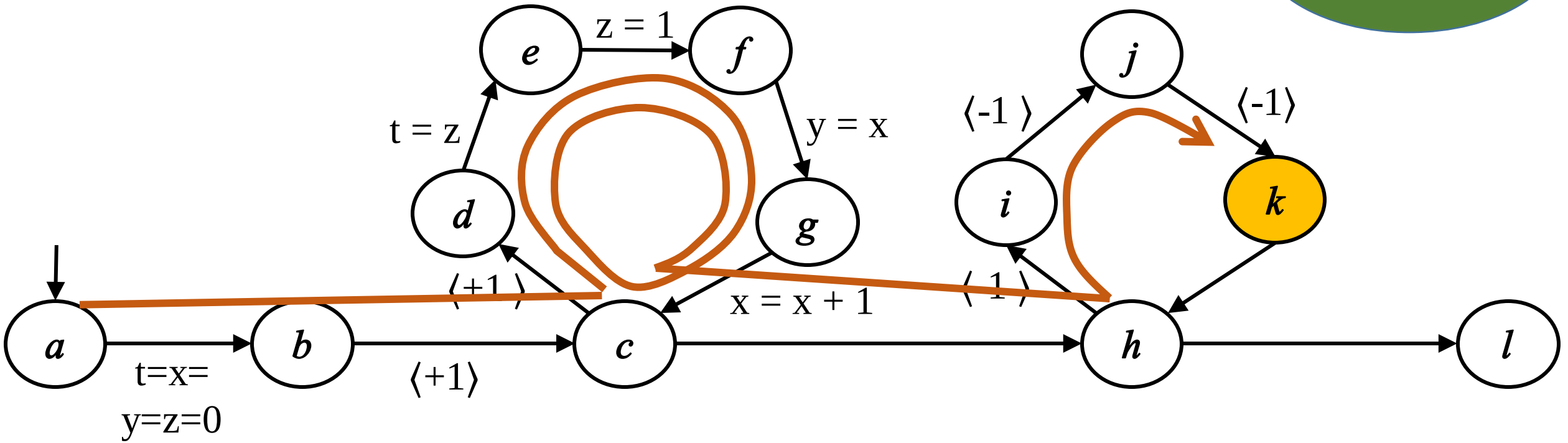

Which variables are constants at k ?
e
z = 1
f
t = z
y = x
d
g
⟨+1 ⟩
x = x + 1
j
⟨-1 ⟩
⟨-1⟩
i
k
⟨-1 ⟩
a
b
c
h
l
t=x=
y=z=0
⟨+1⟩

Which variables are constants at *k* ?



Path 1
Variable values : t = 0, x=0, y=0, z=0
channel contents :[−2]
Infeasible State

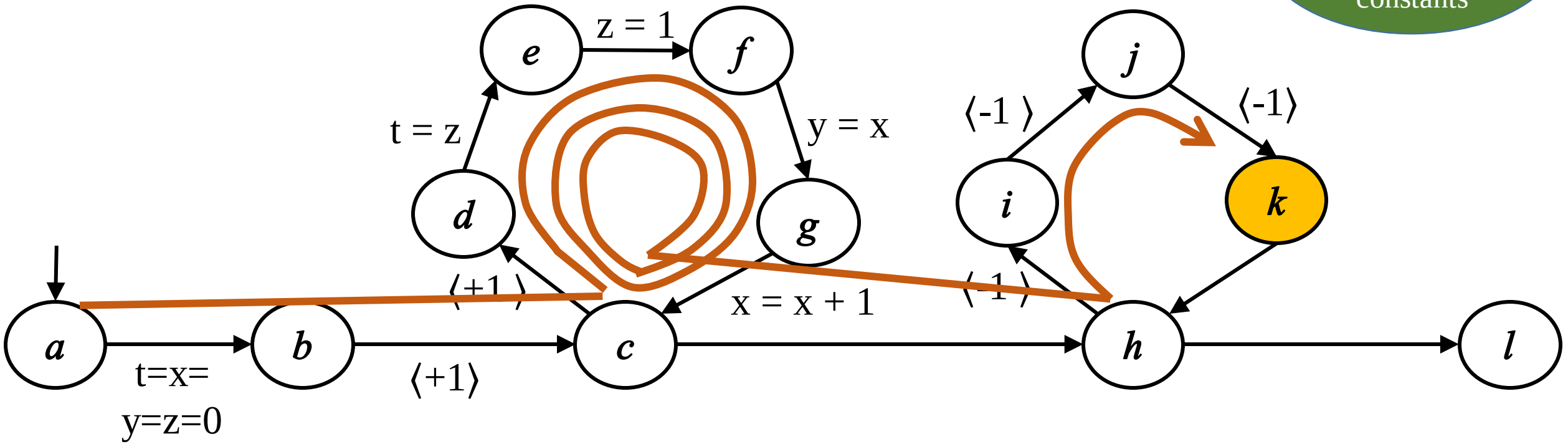
Which variables are constants at *k* ?



Path 2
Variable values : t = 1, x=2, y=1, z=1
channel contents :[0]

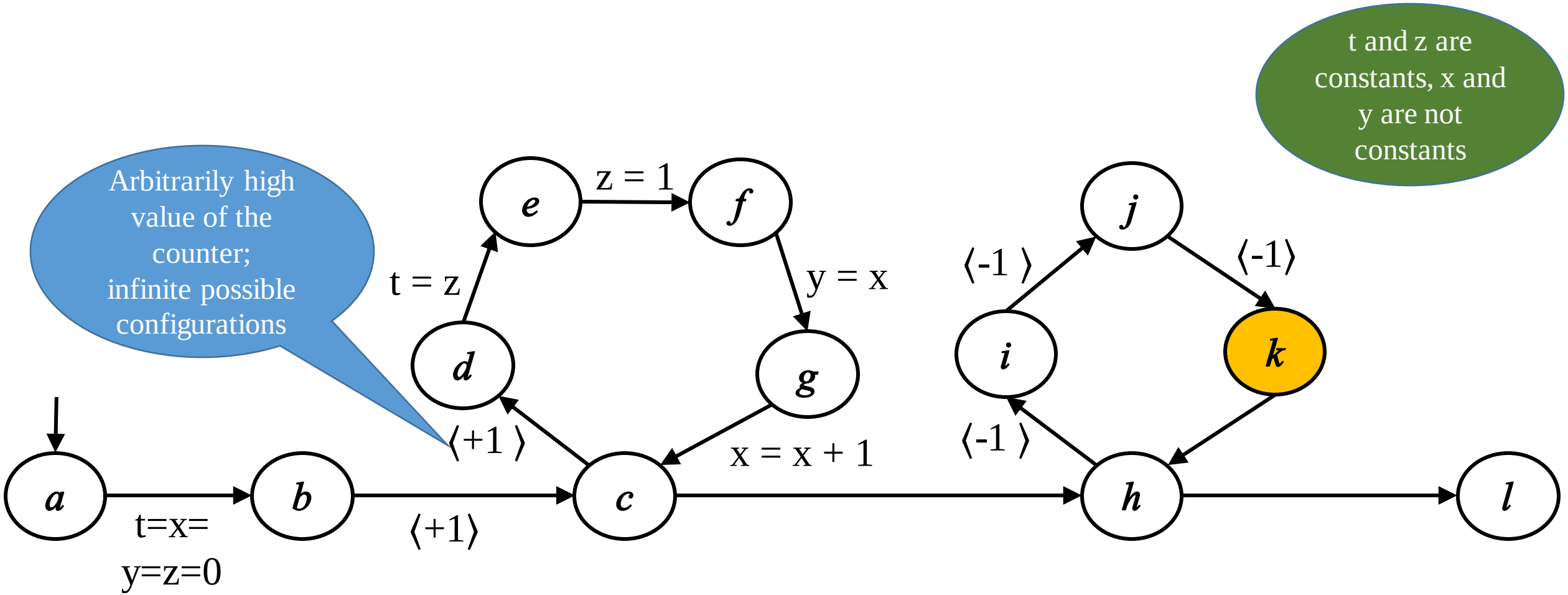
1. t = 1, x=2, y=1, z=1

t and z are constants, x and y are not constants



Path 3
Variable values : t = 1, x=3, y=2, z=1
channel contents :[1]

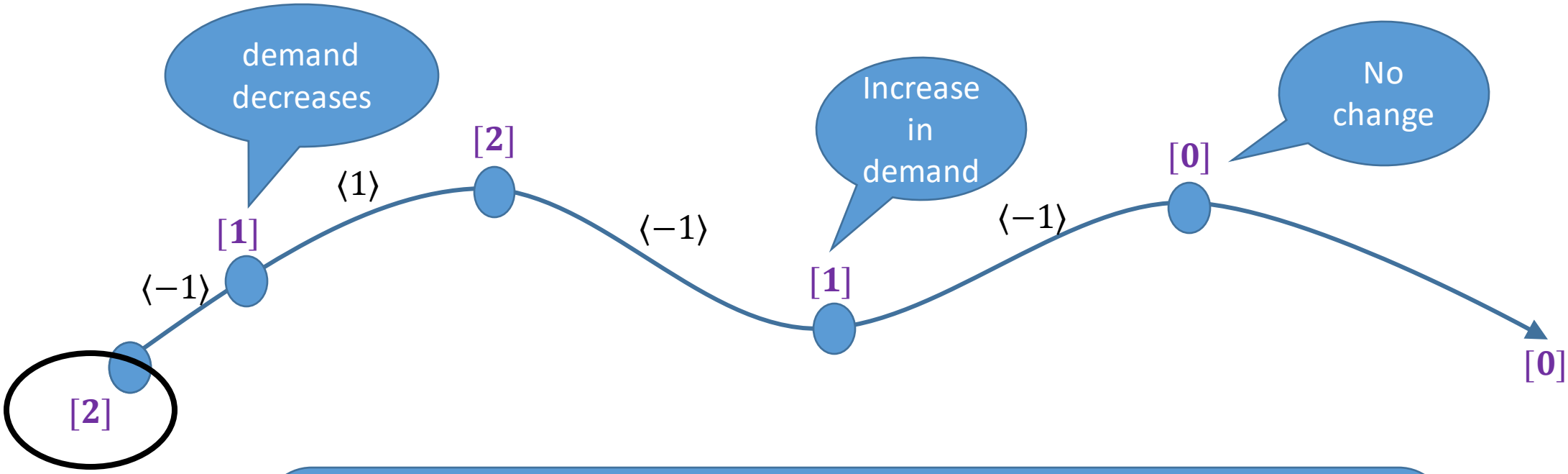
1. t = 1, x=2, y=1, z=1
2. t = 1, x=3, y=2, z=1



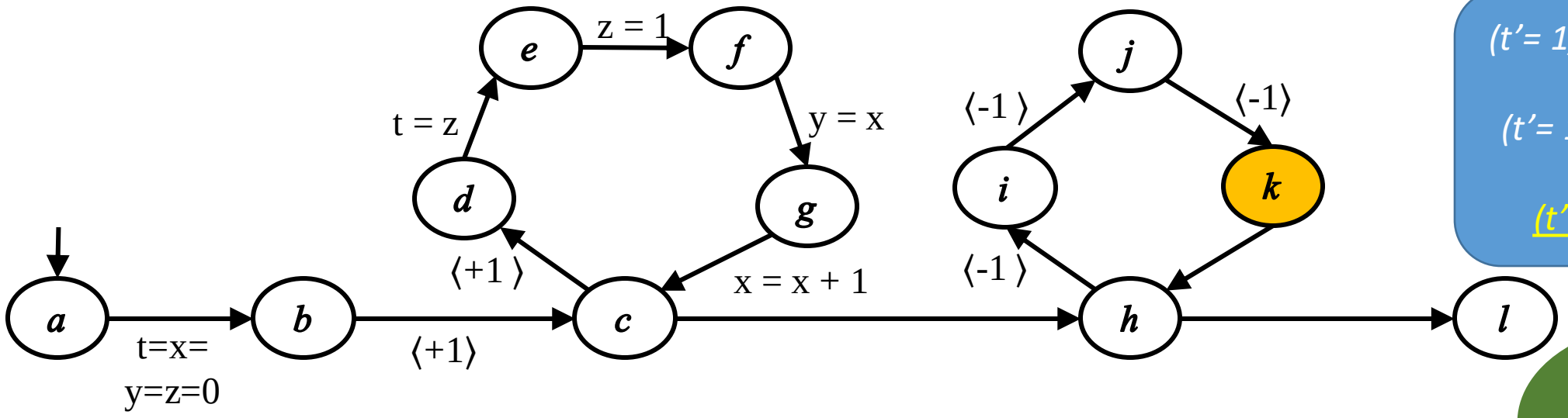
Exploring all paths or all counter configurations to detect constants is infeasible

Backward DFAS Algorithm

Demand of a path



Demand of a path is the minimum channel contents at the beginning of the path, such that, the ending value is >= 0



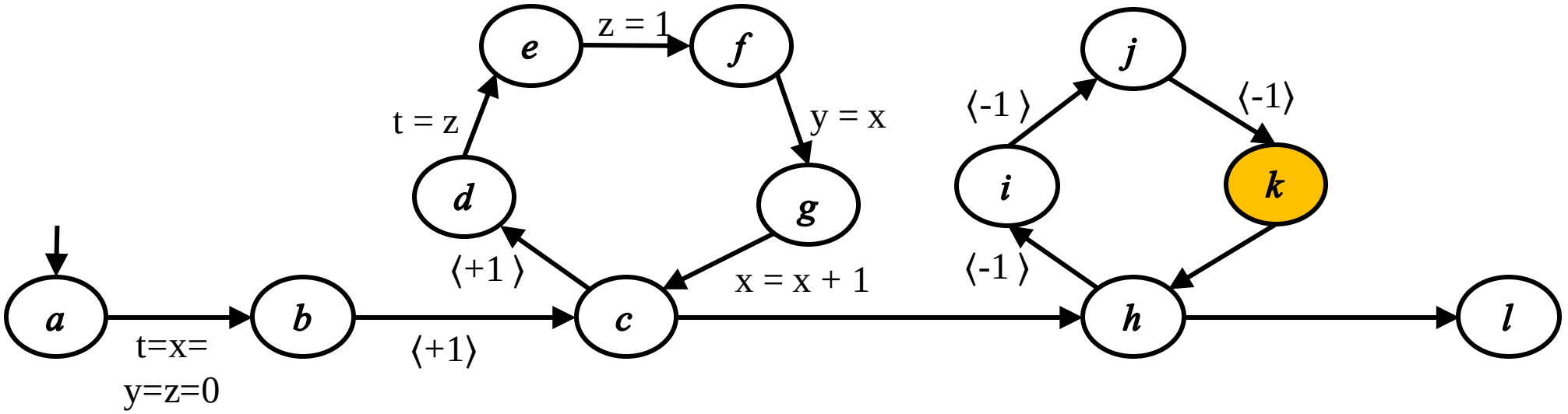
(t'= 1, x'= x+3, y'= x+2, z'= 1)
⊔
(t'= 1,x'= x+4,y'= x+3,z'= 1)
=
(t'= 1,x'=⊤,y'= ⊤,z'= 1)

LCP join operation

Path p	Demand of path p	Path Transfer Function of p
cdefgchijk	*2*	*t'= z, x'= x+1, y'= x, z'= 1*
$(cdefg)^2chijk$	*1*	*t'= 1, x'= x+2, y'= x+1, z'= 1*
$(cdefg)^3chijk$	*0*	*t'= 1, x'= x+3, y'= x+2, z'= 1*
$(cdefg)^4chijk$	*0*	*t'= 1, x'= x+4, y'= x+3, z'= 1*
~~$(cdefg)^5chijk$~~	~~*0*~~	~~*t'= 1, x'= x+5, y'= x+4, z'= 1*~~

The path $(cdefg)^5chijk$ *is covered*

<= demand paths



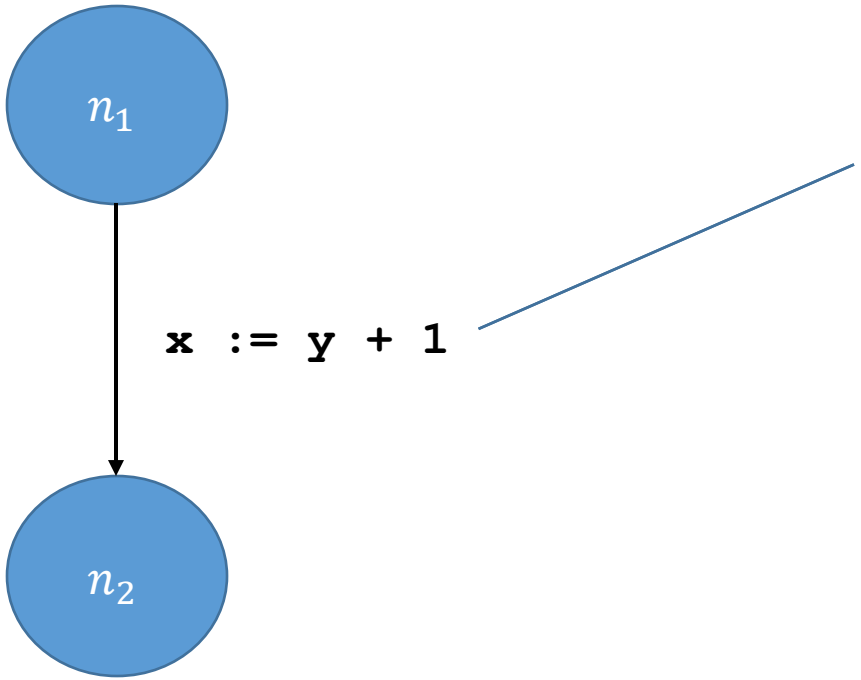
Path From Worklist	Path Extension	Demand of a path	Path Transfer Functions
chijk	***ab****chijk*	*2*	*t'= 0, x'= 0, y'= 0, z'= 0*
cdefgchijk	***ab****cdefgchijk*	*1*	*t'= 0, x'= 1, y'= 0, z'= 1*
$(cdefg)^2chijk$	$ab(cdefg)^2chijk$	*0*	*t'= 1, x'= 2, y'= 1, z'= 1*
$(cdefg)^3chijk$	$ab(cdefg)^3chijk$	*0*	*t'= 1, x'= 3, y'= 2, z'= 1*
~~$(cdefg)^4chijk$~~	~~$ab(cdefg)^4chijk$~~	~~*0*~~	~~*t'= 1, x'= 4, y'= 3, z'= 1*~~

Implementation and Experimental Evaluation

Implementation

- implemented both Backward and Forward DFAS algorithms.
- used 14 benchmarks; Spin, P, JPF-actor models, Go Programs
  - Implementation assumes an XML representation of the program
- highly parallel prototype, to handle the complexity of the algorithm

Experimental setup



Nodes part of a VCFG

Objective is to find whether the use of **y** is a constant at n_1

Constant Propagation (CP) is the popular abstract domain for detecting constants

Implementation

- CP is an infinite abstract domain; Forward DFAS instantiated with CP
  - analyzes all instructions with “best” precision
- Linear Constant Propagation(LCP) is a less precise variant of CP; Backward DFAS instantiated with the LCP analysis
  - LCP is an infinite abstract domain. However, finite-height transfer function lattice unlike CP
- Baselines:
  - Join-Over-all-Paths (JOP): ignores all queue operations, CP analysis
  - JOFP using Copy Constant Propagation (CCP):
    - CCP is a finite abstract domain, and is less precise than LCP
    - ***Closest related work that computes JOFP admits only abstract finite domains***

Results Summary

- Uses Identified as constants:

Total Number of Uses	189
Constants identified by Forward DFAS	63
Constants identified by Backward DFAS	49
Constants identified by CP JOP (Baseline)	22
Constants identified by CCP JOFP (Baseline)	39

- Both outperform the baselines
  - Forward DFAS finds 2.9x more constants than JOP, 62% more than CCP
  - Backward DFAS finds 2.2x more constants than JOP, 26% more than CCP
- In summary Forward DFAS appears more precise, but Backward DFAS infers more constants in 2 benchmarks. Thus, they are incomparable.

Results Summary

- Use the inferred constants to verify assertions present in the benchmark models
- Assertions verified:

Total Assertions	42
Assertions verified by Forward DFAS	22
Assertions verified by Backward DFAS	20
Assertions verified by CP JOP	4
Assertions verified by CCP JOFP	14

- Both Forward and Backward DFAS verify almost 5 times the assertions verified by JOP
- Forward DFAS verifies 57% more assertions than CCP, Backward DFAS verifies 43% more

Execution Time

- Execution Time slowdown with respect to the Naïve baseline, i.e., JOP:

Approach	Geometric Mean	Execution Time Range
Forward DFAS	0.64	1.2s – 18s (did not terminate on 2 benchmarks)
Backward DFAS	0.09	1s – 284s
CCP JOFP Baseline	0.11	1s – 226s

Backward DFAS seems slower; demand-driven nature results in cumulative reporting of time

JOP is always

CCP is comparable with Backward DFAS

Future Work

- To extend Backward DFAS algorithm to admit any infinite domain, and support widening
- Improving the scalability of Forward DFAS; explore approaches like Partial Order Reduction
- Handling parameters in function calls
- To run the algorithm on real world asynchronous programs.

Related Work

- *Jhala, Ranjit, and Rupak Majumdar. "Interprocedural analysis of asynchronous programs." ACM SIGPLAN Notices. Vol. 42. No. 1. ACM, 2007.*
- *Abdulla, Parosh Aziz, and Bengt Jonsson. "Verifying programs with unreliable channels." information and computation 127.2 (1996): 91-101.*
- *Bouajjani, Ahmed, and Michael Emmi. "Analysis of recursively parallel programs." ACM Sigplan Notices. Vol. 47. No. 1. ACM, 2012.*

All these approaches admit only finite abstract domains

Conclusion

- Proposed two approaches for interprocedural data flow analysis of asynchronous systems using infinite abstract domains,
  - Backward DFAS that computes the precise JOFP solution, admits a restricted class of infinite abstract domains
  - A more general Forward DFAS algorithm, that gives a sound solution. It over-approximates the JOFP in general.
- We have implemented the interprocedural algorithms as a tool
- The proposed algorithm performs significantly better than two different baseline approaches on 14 benchmarks