Techniques for Precise and Scalable Data Flow Analysis

by

Komal Pathade

under the guidance of Prof. Uday Khedker

Thanks

Outline

1. Data Flow Analysis and Its Existing Solutions

2. Improving Precision of Data Flow Analysis

3. Improving Scalability of Data Flow Analysis

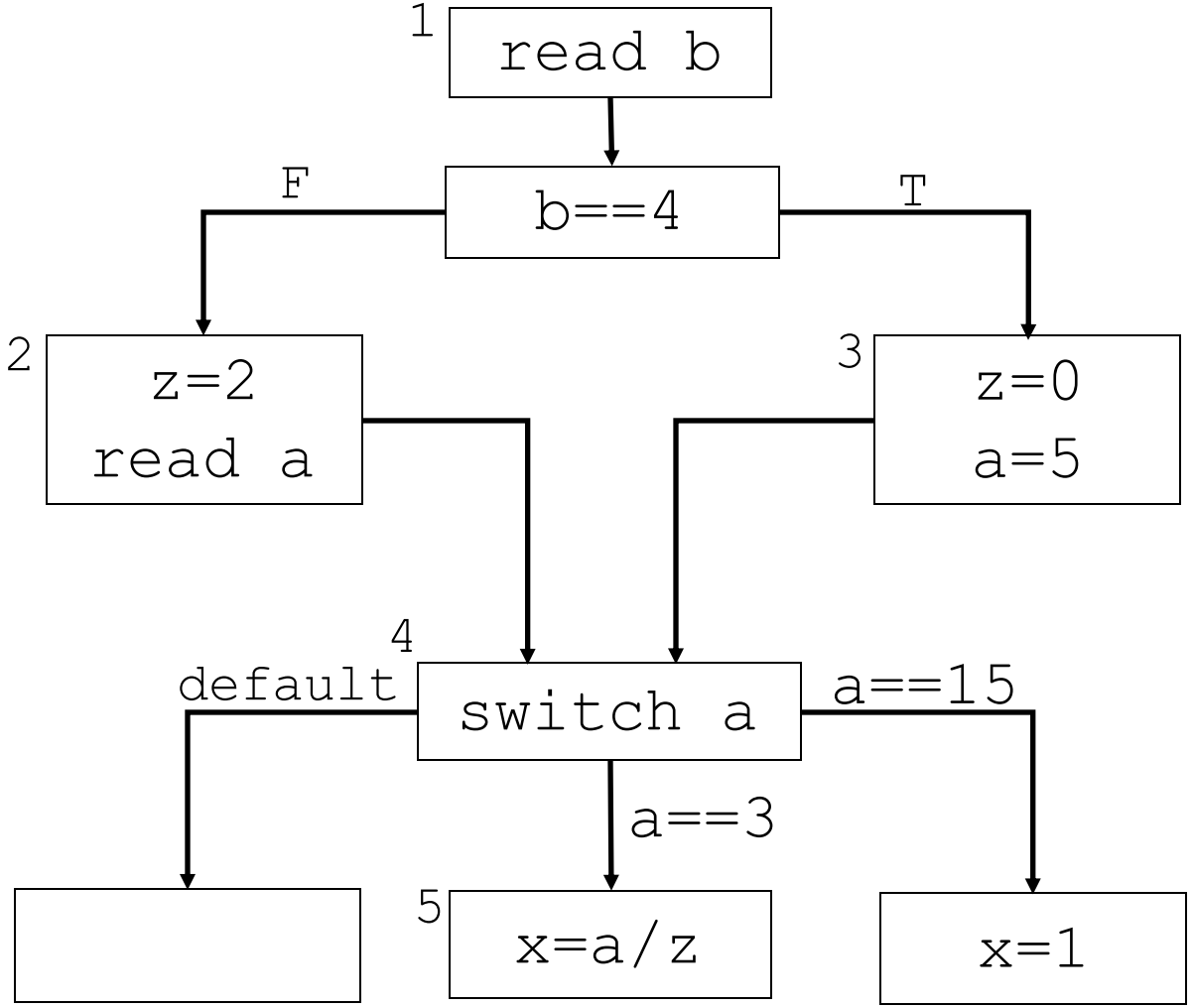
Outline

1. Data Flow Analysis and Its Existing Solutions

2. Improving Precision of Data Flow Analysis

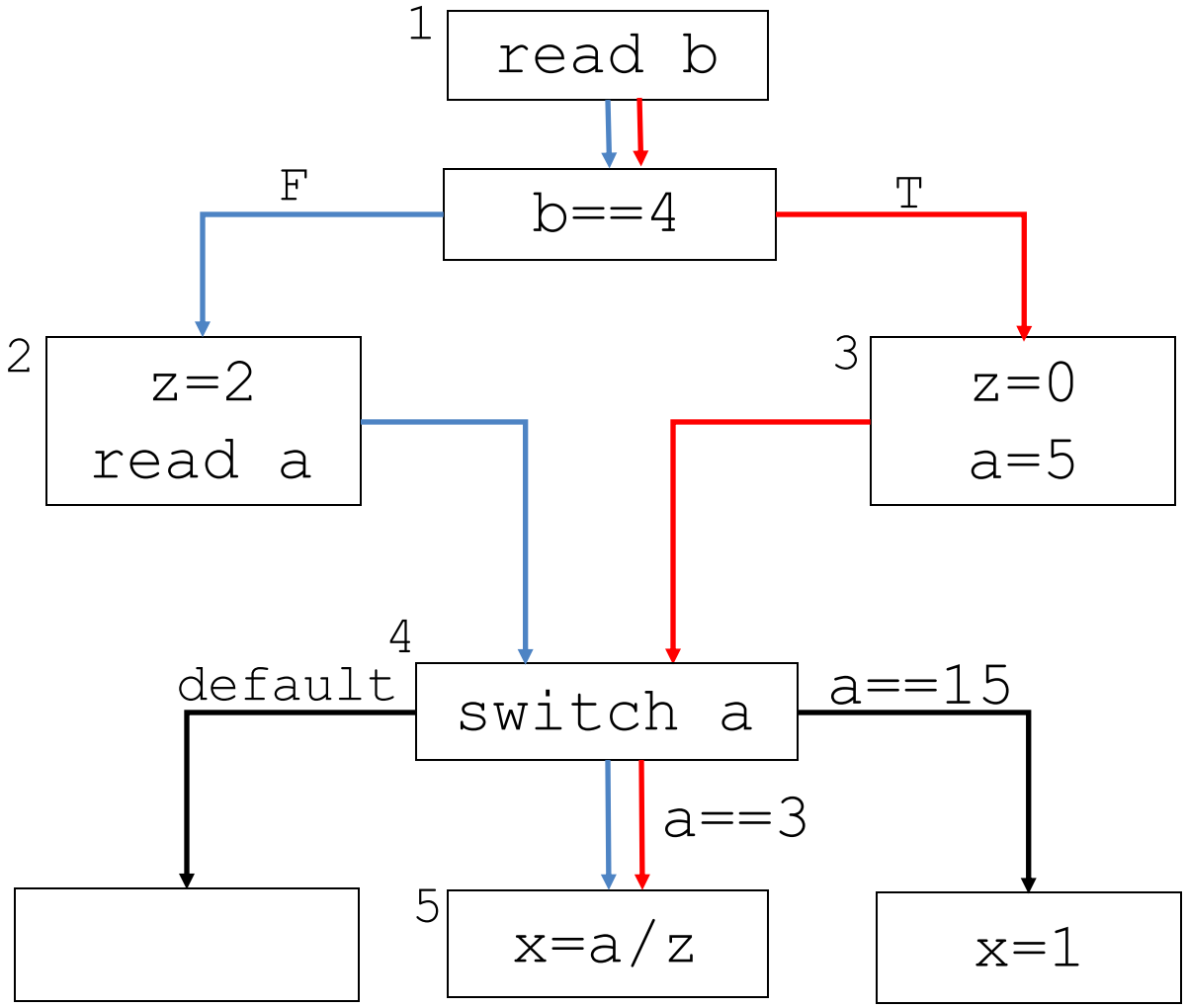
3. Improving Scalability of Data Flow Analysis

Data Flow Analysis- A simple example



Values of *z at node 5 ?*

Meet Over Paths (MOP) Solutions

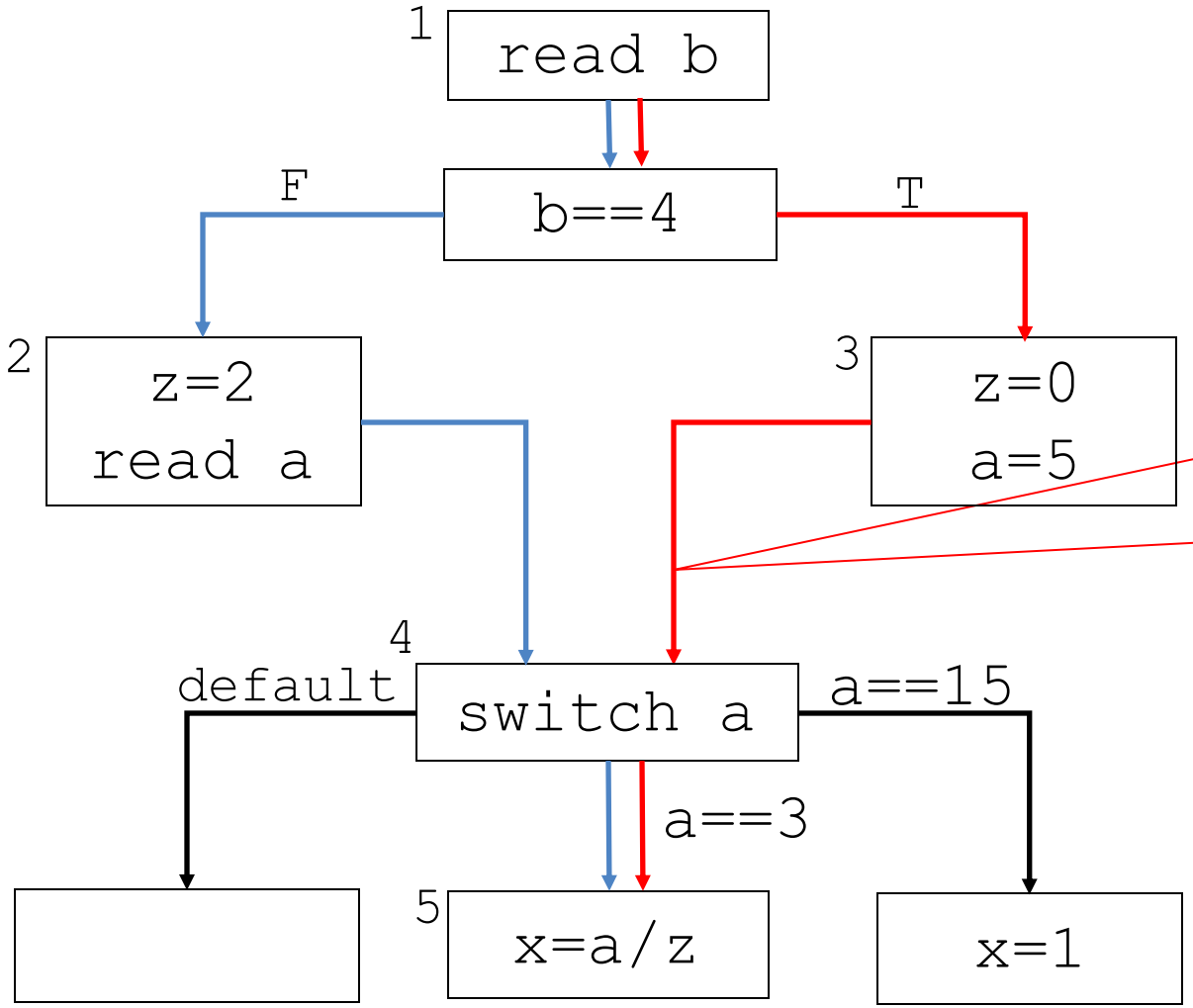


Values of *z at node 5 ?*

- Computes value along each path separately

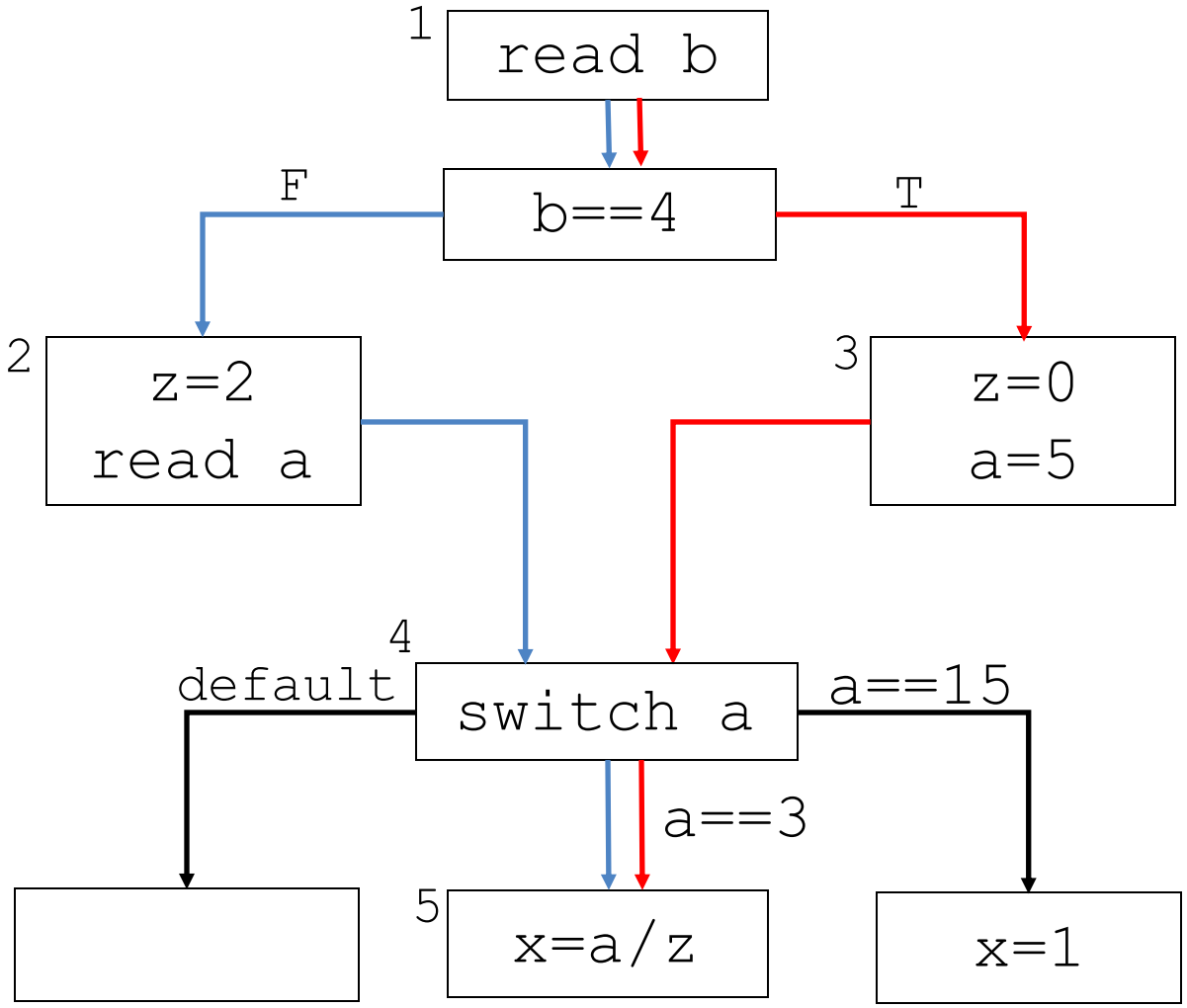
Node	Value of z from paths	
	Blue	Red
5	z=2	z=0

Meet Over Paths (MOP) Solutions



- An Infeasible Path
- Approaches exist to detect such paths (e.g., work by R. Bodik. et. al,1997)

Meet Over Paths (MOP) Solutions

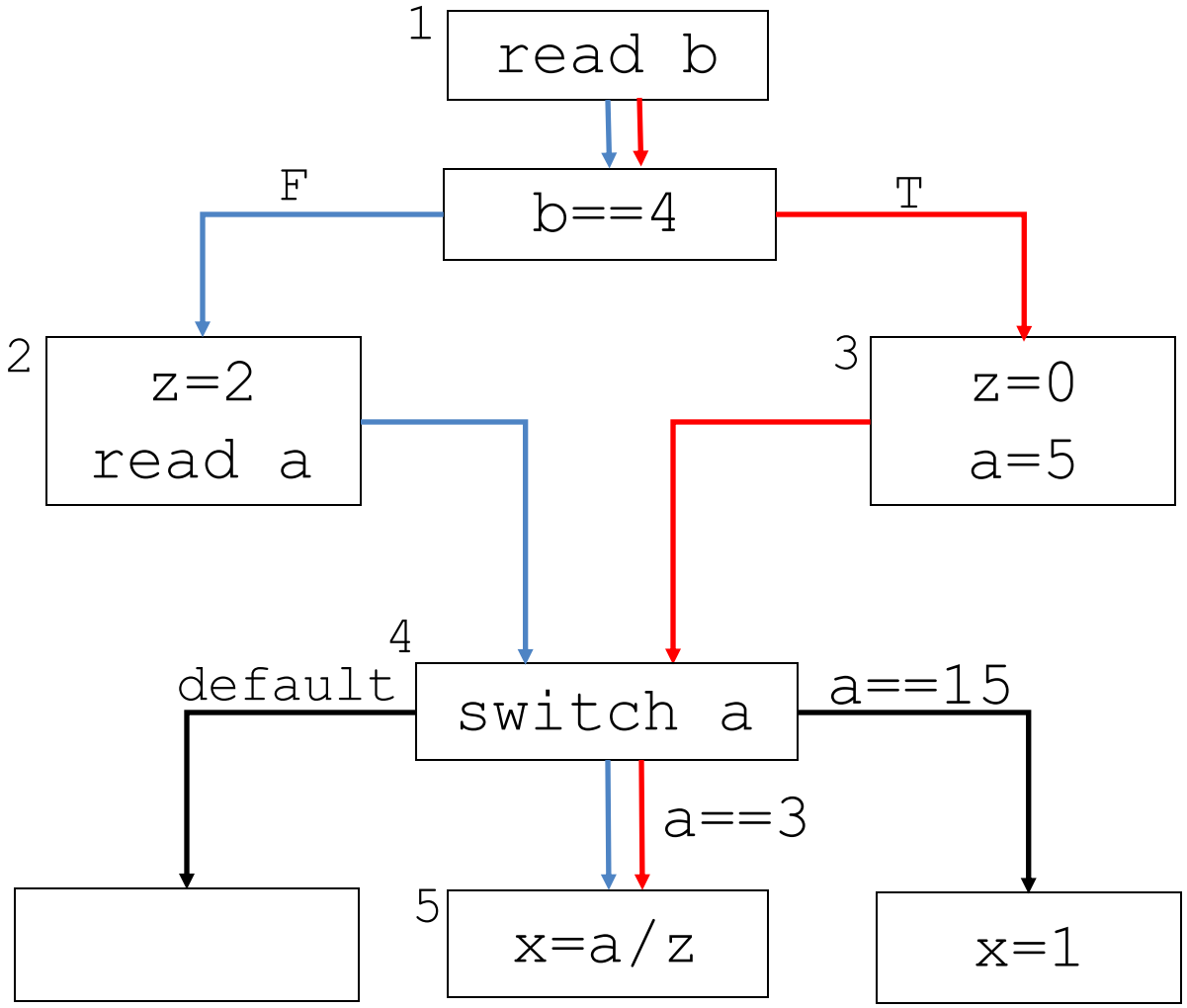


Values of *z at node 5 ?*

- Computes value along each path separately

Node	Value of z from paths	
	Blue	Red
5	z=2	z=0

Meet Over Paths (MOP) Solutions



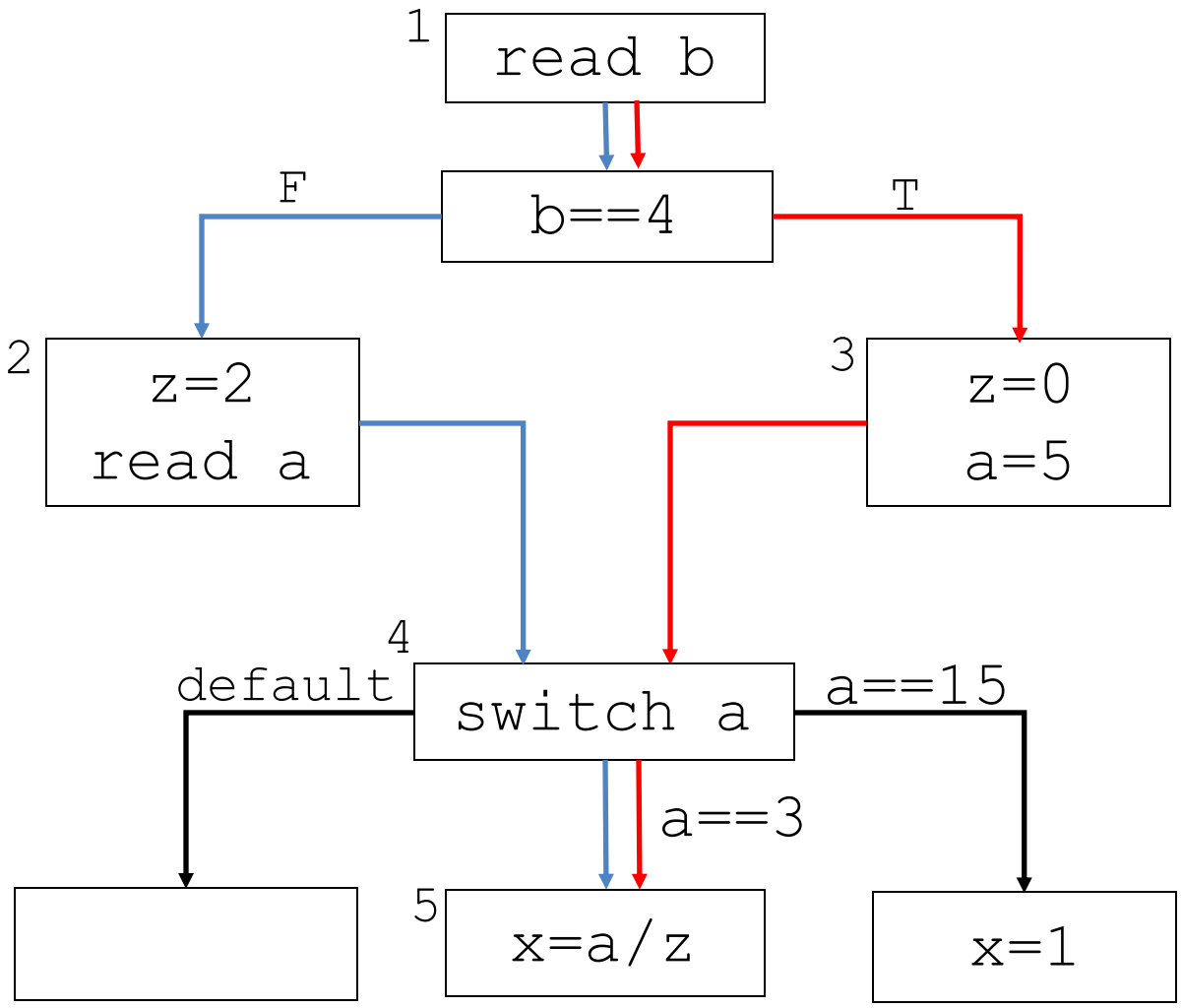
Values of *z at node 5 ?*

- Computes value along each path separately

Node	Value of z from paths	
	Blue	Red
5	z=2	z=0

- Red path is infeasible so z=0 is discarded,
- Thus, the remaining value z=2 is precise
- Not scalable, because the number of paths can be very large

Maximum Fix Point (MFP) Solutions



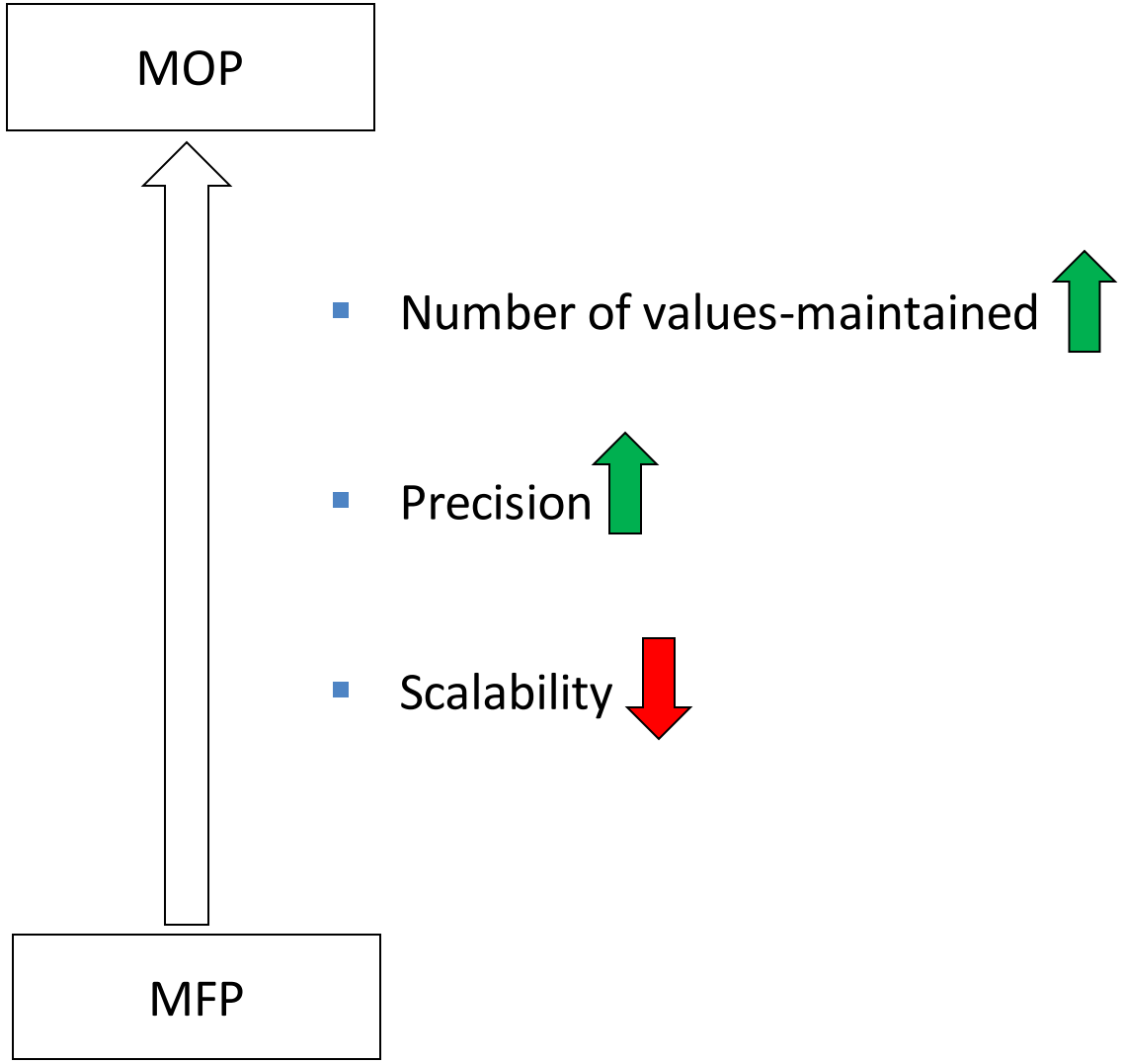
Values of *z at node 5 ?*

- Compute range of values reaching along all paths

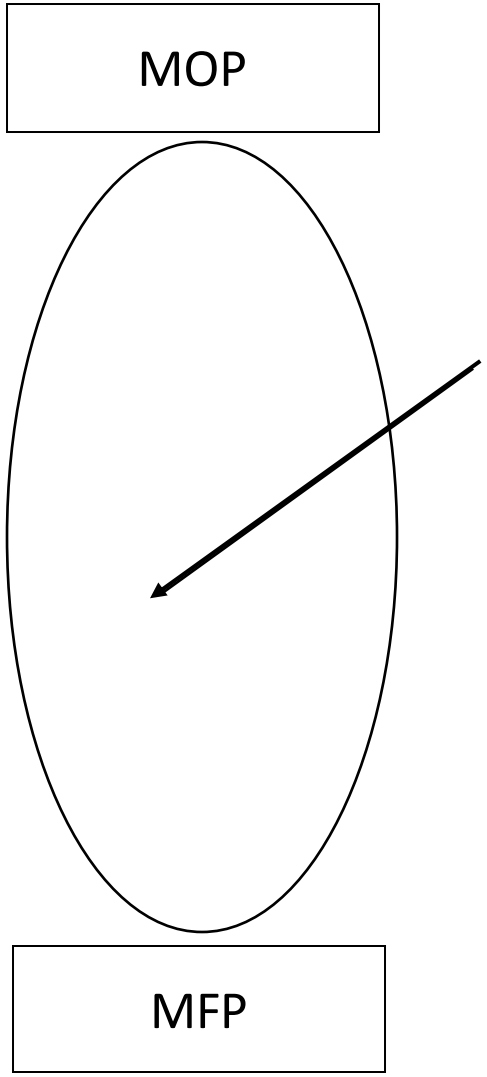
Node	Value Range of z from all paths
5	z=[0,2]

- Scalable: computes only one value per program point
- Imprecise: z=0 is included which reaches along infeasible path (marked in red)
- No obvious way of eliminating 0 at node 5

Trade-off


MOP
MFP
Number of values-maintained
Precision
Scalability

Research Gap



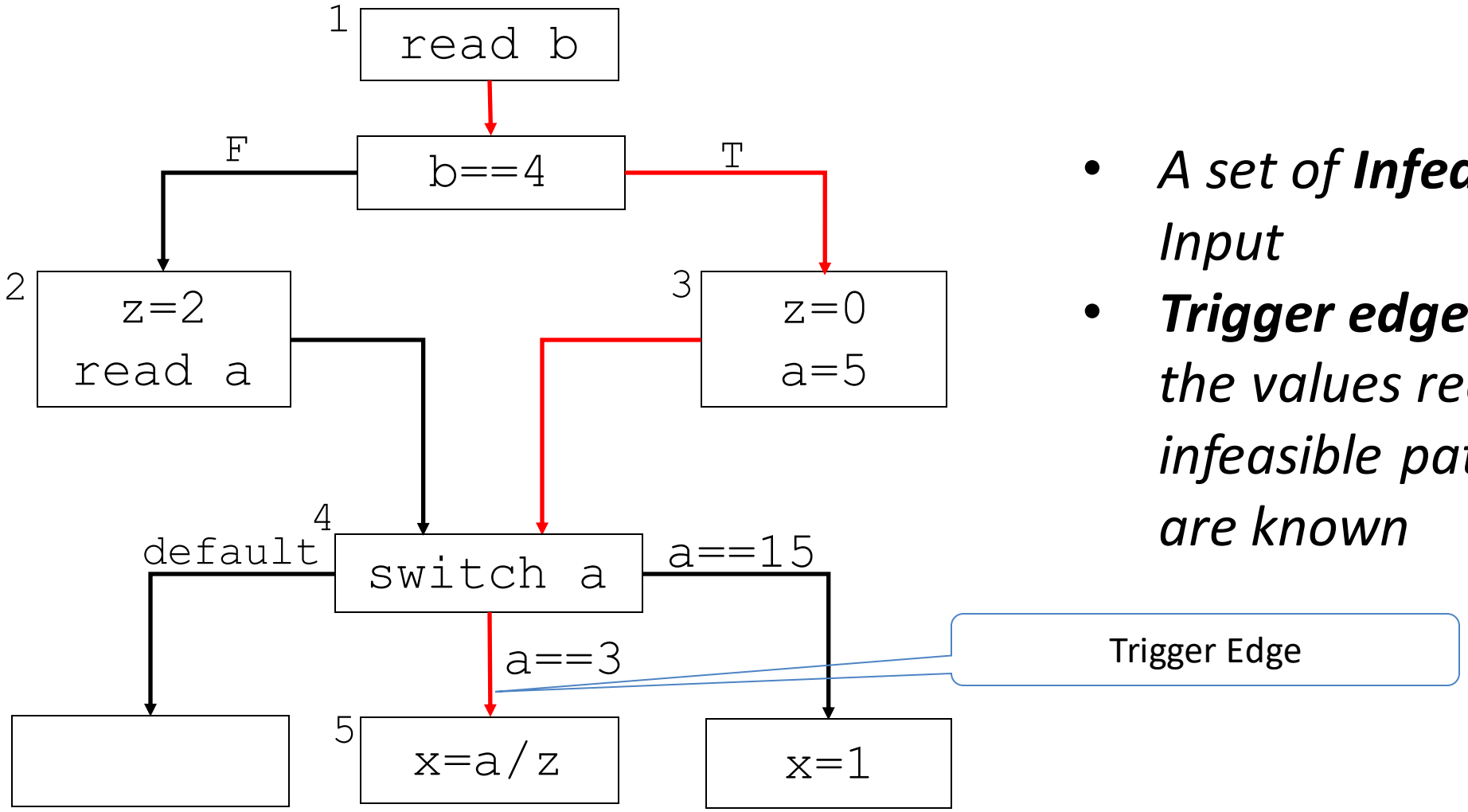
No generic and efficient solution for eliminating infeasible path effect in this area

1. Data Flow Analysis and Its Existing Solutions

2. Improving Precision of Data Flow Analysis

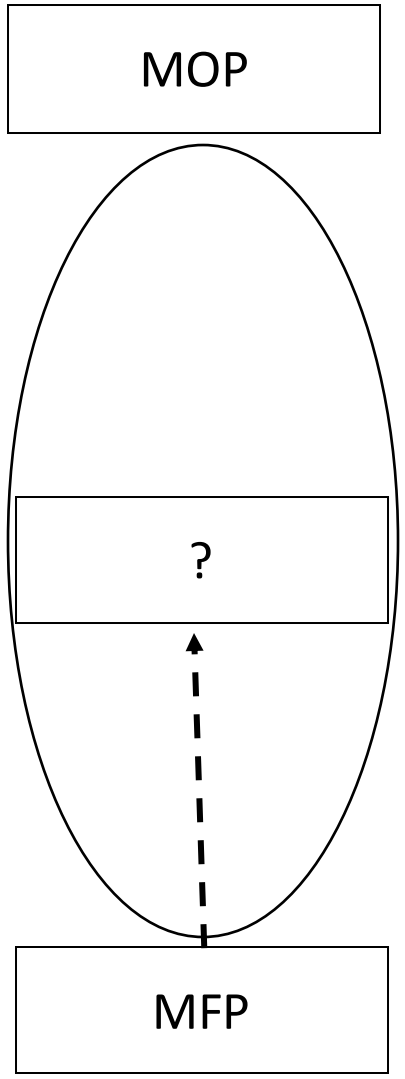
3. Improving Scalability of Data Flow Analysis

Assumption



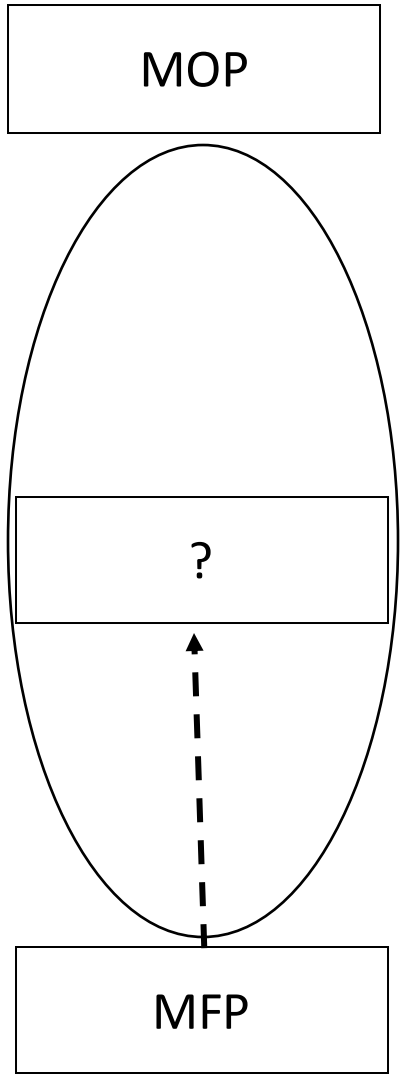
- *A set of **Infeasible paths** is given as Input*
- ***Trigger edges** (i.e., edges at which the values reaching along infeasible path should be blocked) are known*

Problem Statement



Can we improve the precision of MFP solutions, ***if we know which paths are infeasible*** *?*

Problem Statement



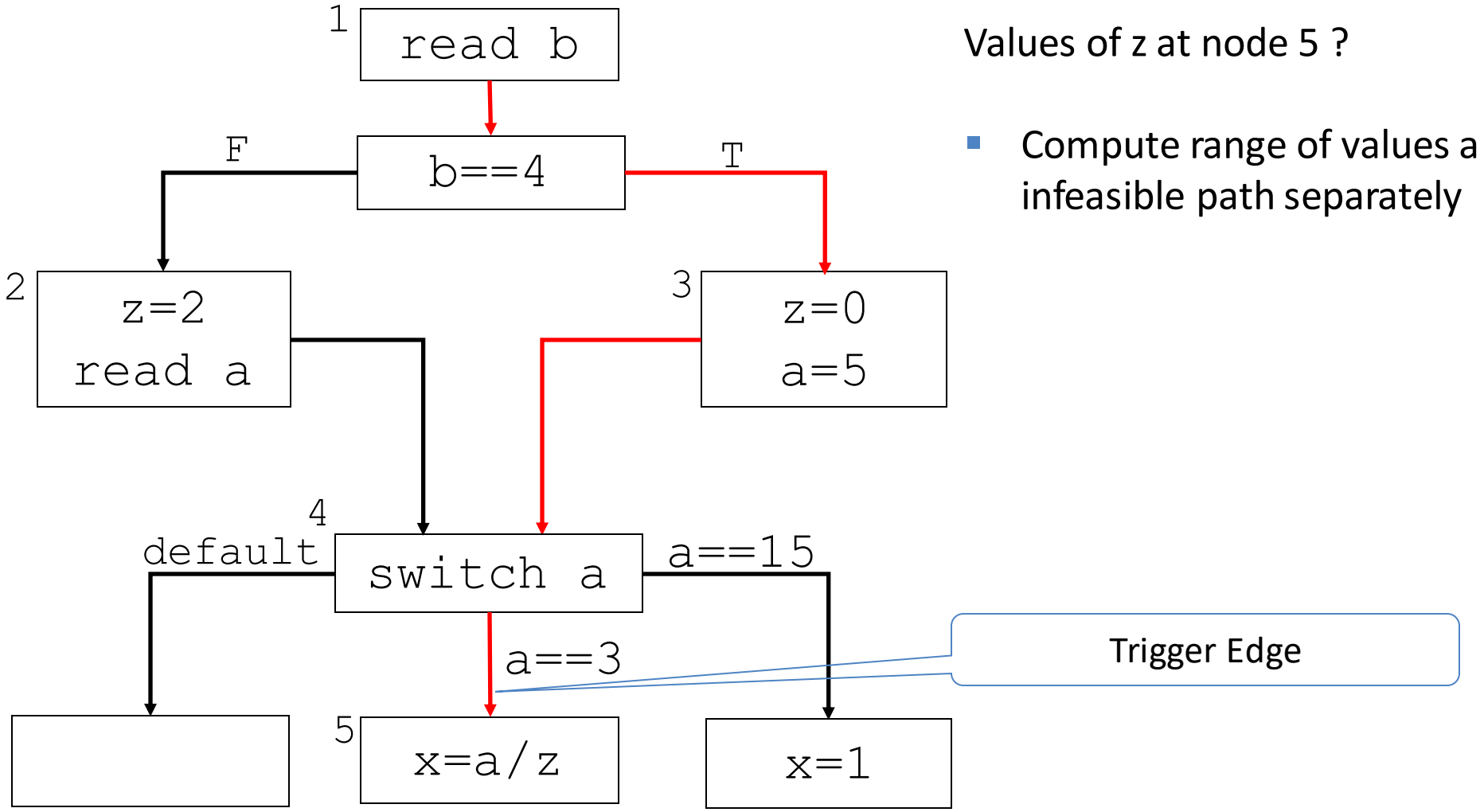
Can we improve the precision of MFP solutions, ***if we know which paths are infeasible*** *?*

Desirable:
A Generic and Efficient Solution

Our Contribution

We introduce **Feasible Path MFP Solutions**

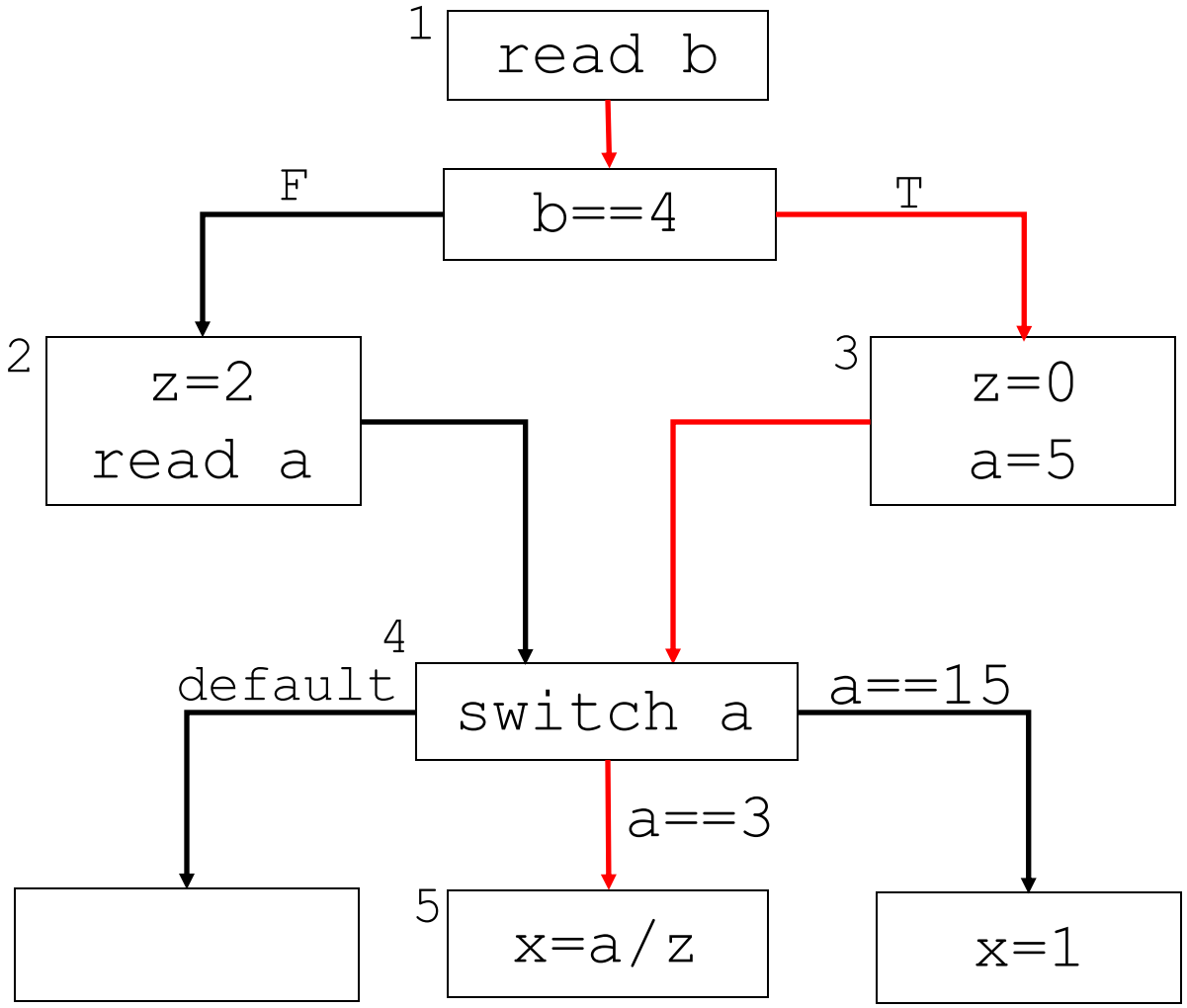
Overview of Feasible Path MFP Solutions



Values of z at node 5 ?

- Compute range of values along feasible and infeasible path separately

Overview of Feasible Path MFP Solutions

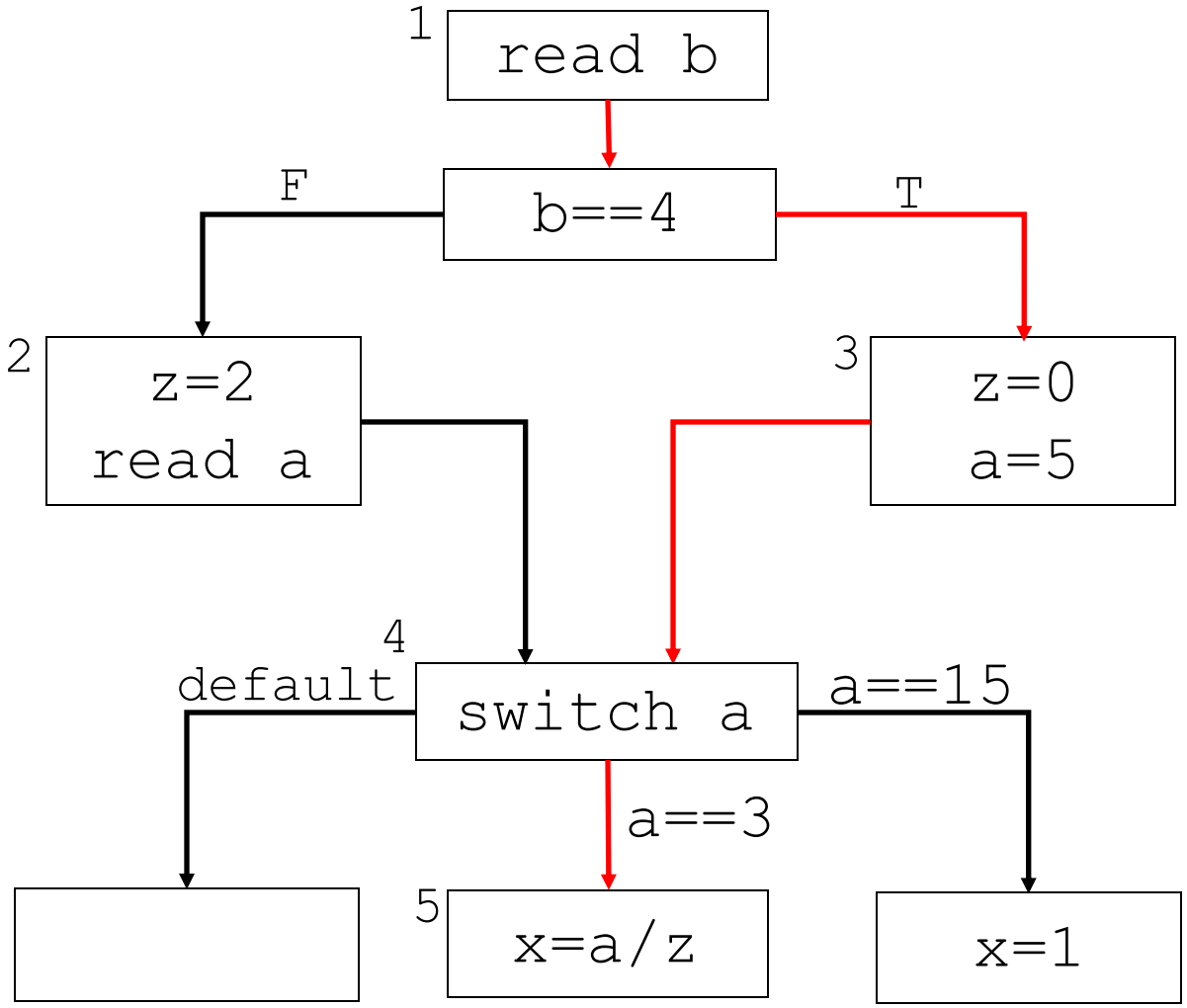


Values of z at node 5 ?

- Compute range of values along feasible and infeasible path separately

Node	Value range of z from	
	Feasible Paths	Infeasible Paths
		Red
4	z=[2,2]	z=[0,0]
5	z=[2,2]	z=[0,0]

Overview of Feasible Path MFP Solutions

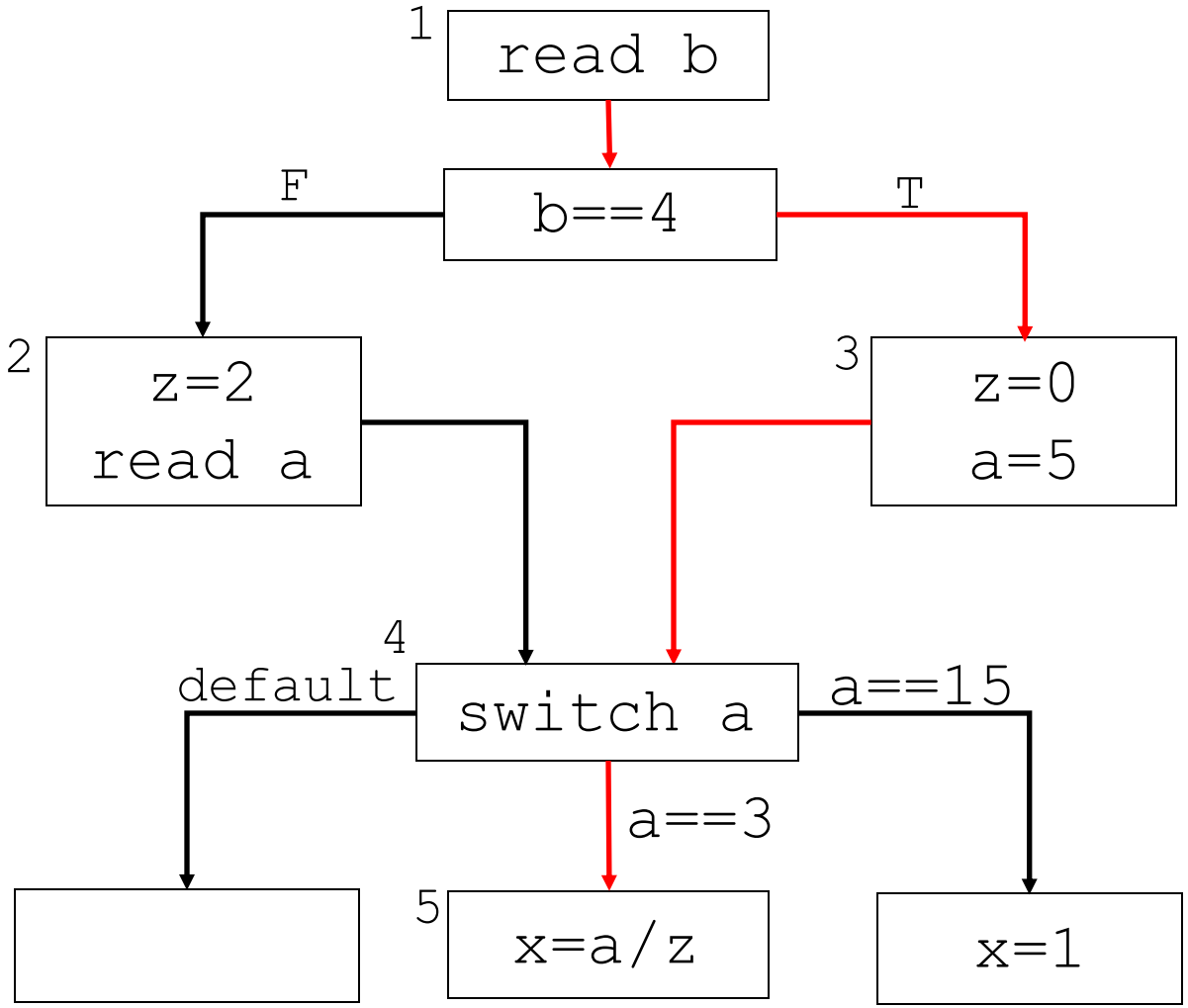


Values of z at node 5 ?

- Compute range of values along feasible and infeasible path separately

Node	Value range of z from	
	Feasible Paths	Infeasible Paths
		Red
4	z=[2,2]	z=[0,0]
5	z=[2,2]	z=[,0]

Overview of Feasible Path MFP Solutions



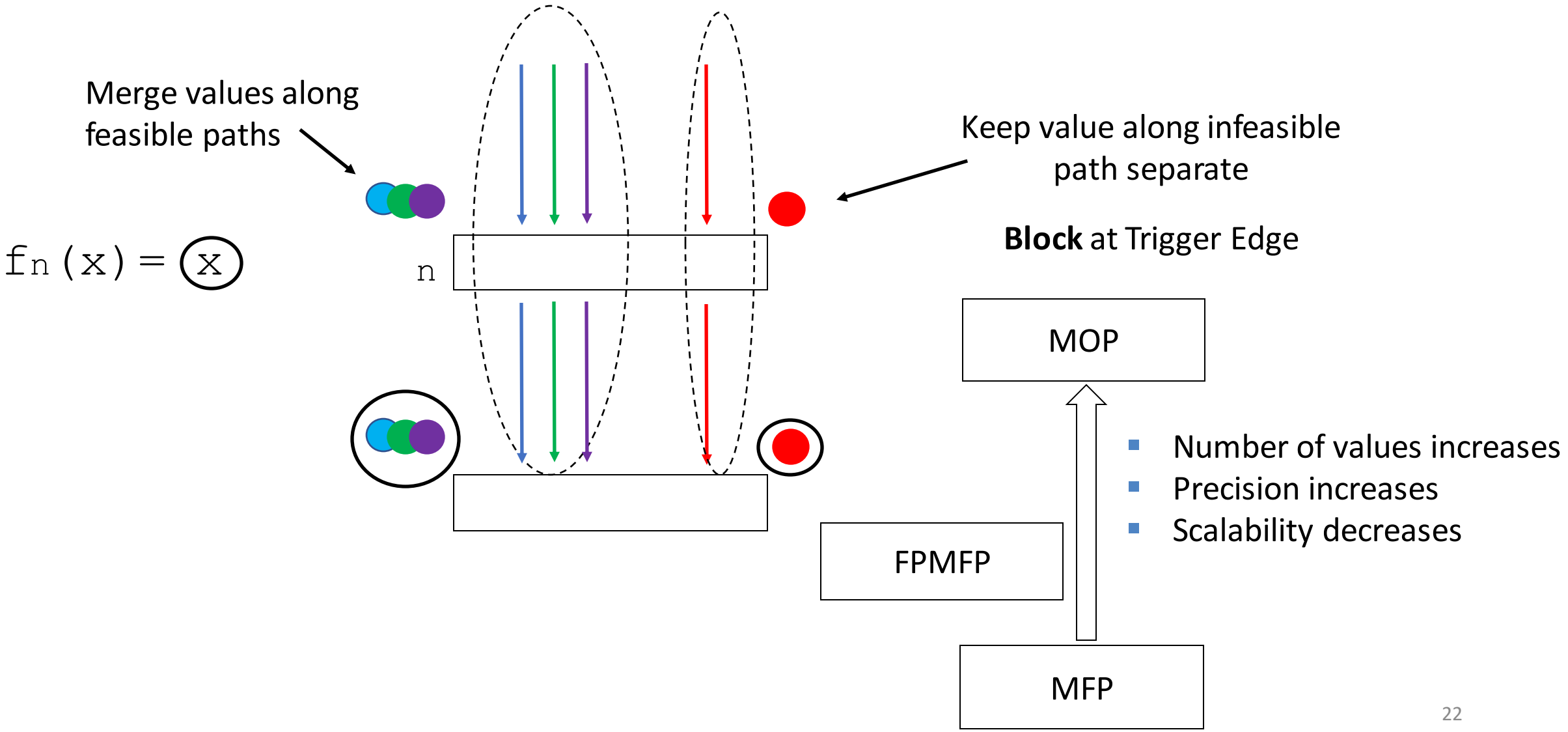
Values of z at node 5 ?

- Compute range of values along feasible and infeasible path separately

Node	Value range of z from	
	Feasible Paths	Infeasible Paths
		Red
4	z=[2,2]	z=[0,0]
5	z=[2,2]	z=[,0]

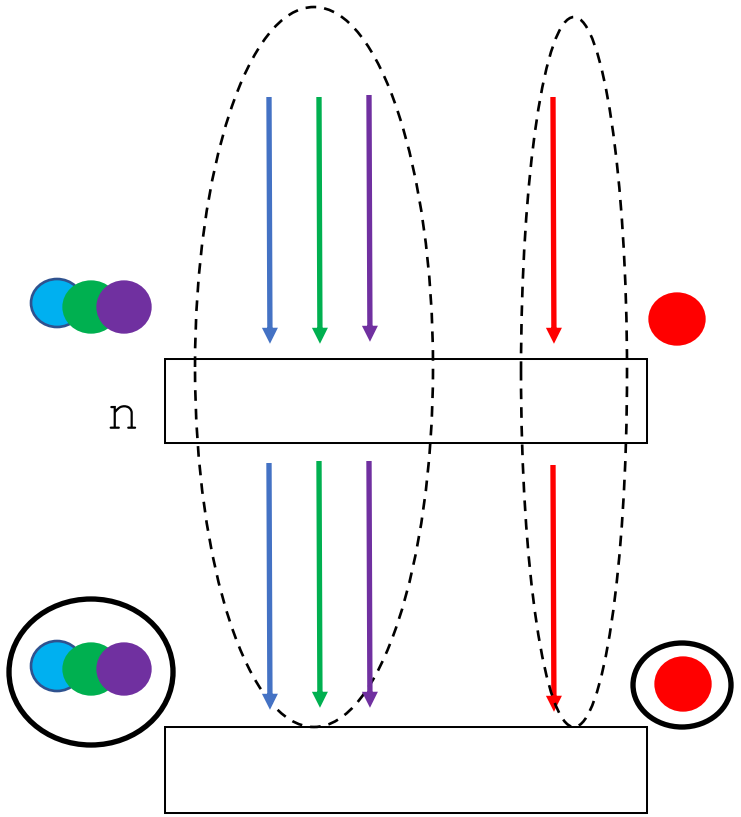
- The final range is meet of ranges from all buckets (except discarded ranges)

Overview of Feasible Path MFP Solutions



Challenges in Computing Precise FPMFP Solutions

$f_n(x) = x$



Is one Bucket sufficient for all infeasible paths?

Answer: Depends on the **structure** of infeasible paths

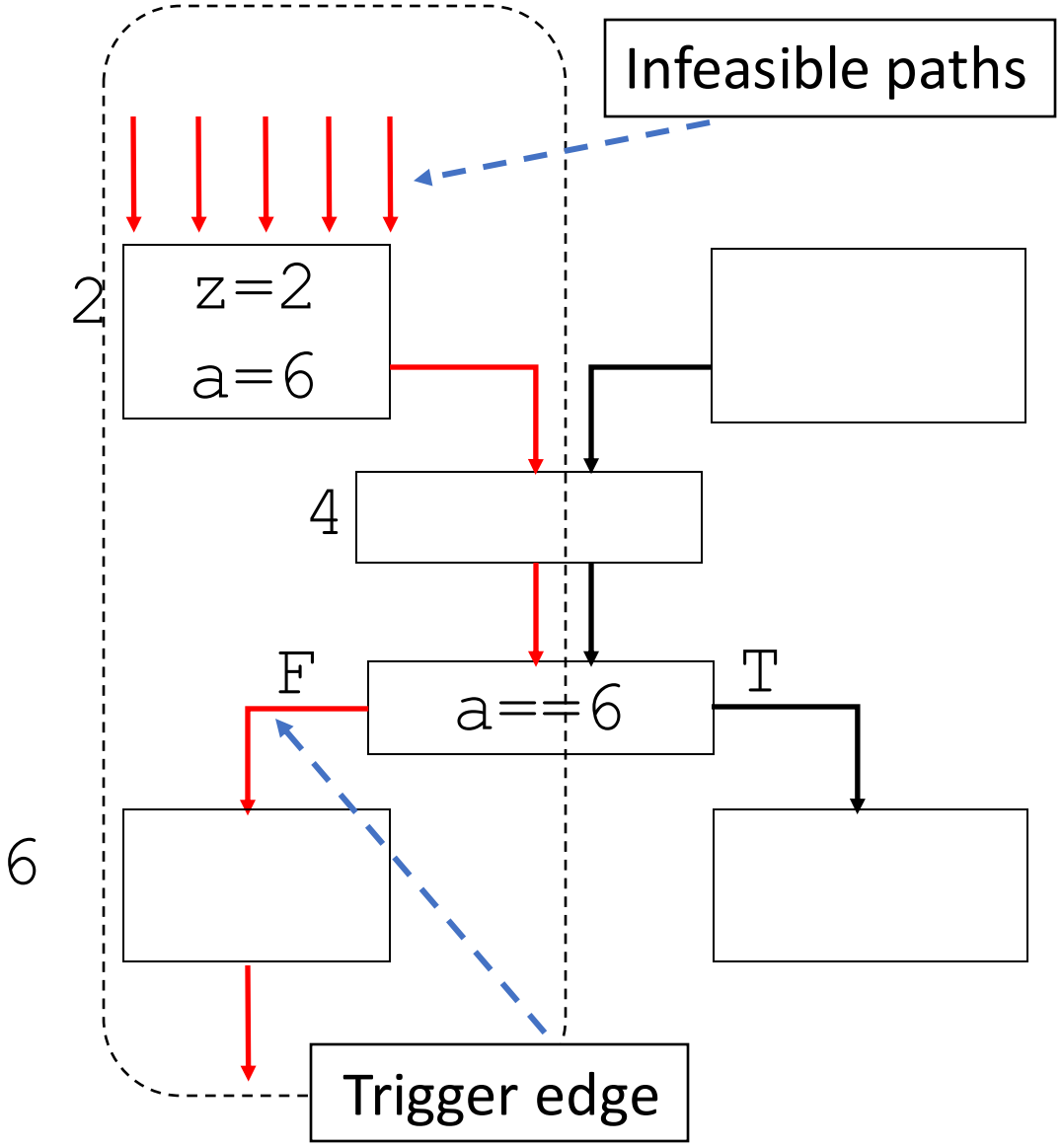
Challenges in Computing Precise FPMFP Solutions

- Dealing with Multiple Infeasible Paths
- Dealing with Overlapping Infeasible Paths
- Dealing with Infeasible Paths across Procedures

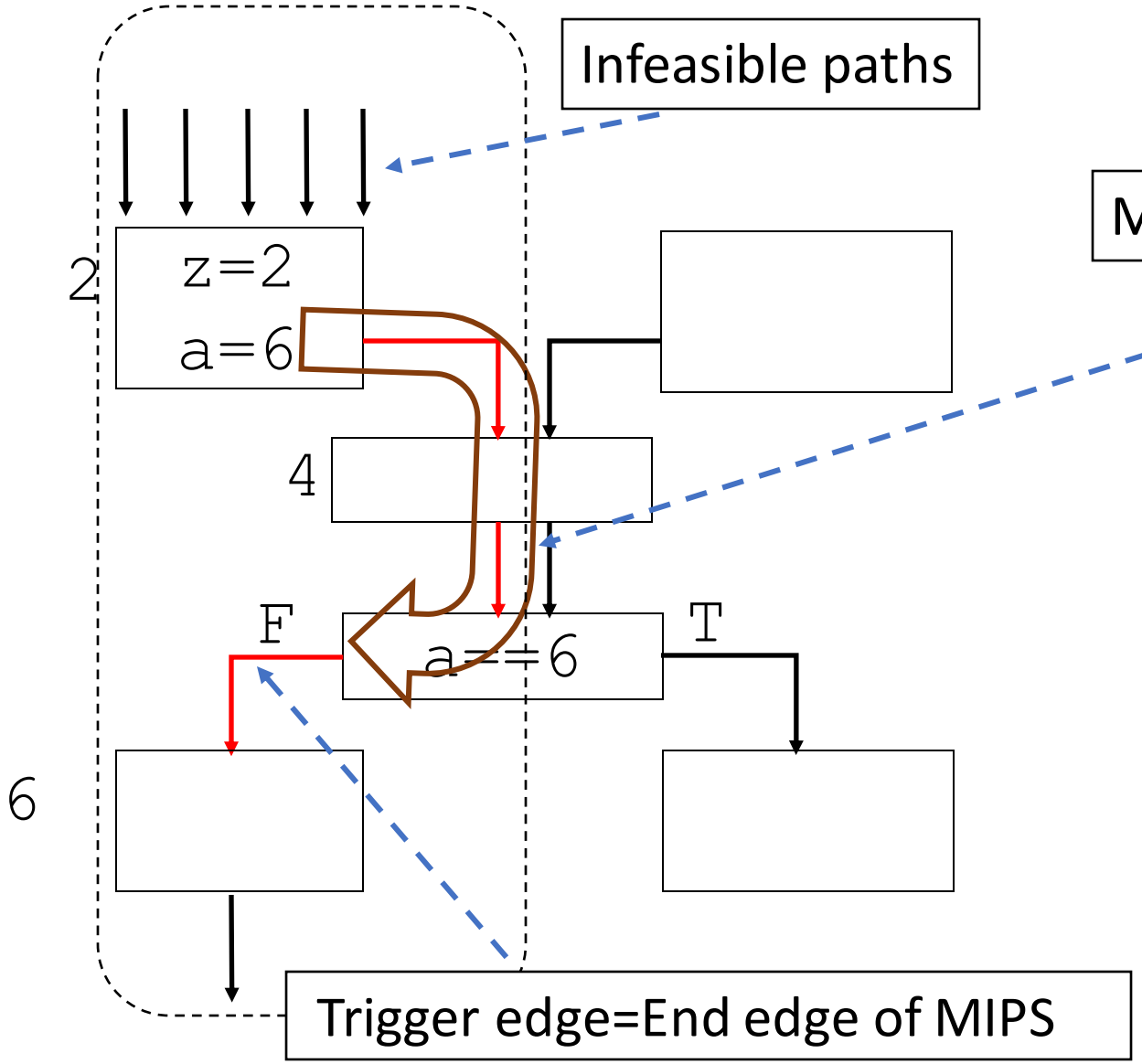
Challenges in Computing Precise FPMFP Solutions

- **Dealing with Multiple Infeasible Paths**
- Dealing with Overlapping Infeasible Paths
- Dealing with Infeasible Paths across Procedures

Dealing with Multiple Infeasible Paths

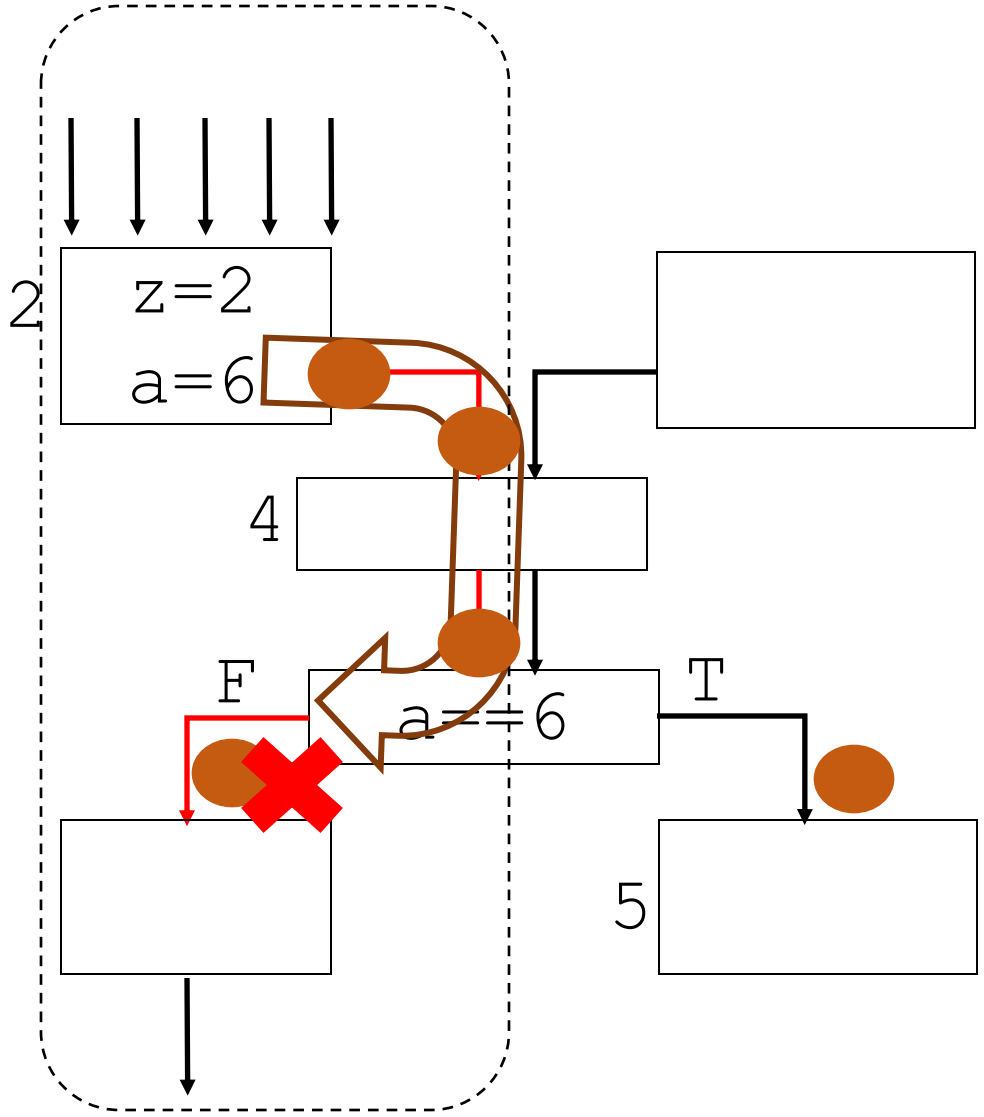


Dealing with Multiple Infeasible Paths



"Any path that contains a MIPS is infeasible" (Bodik [1])

Dealing with Multiple Infeasible Paths

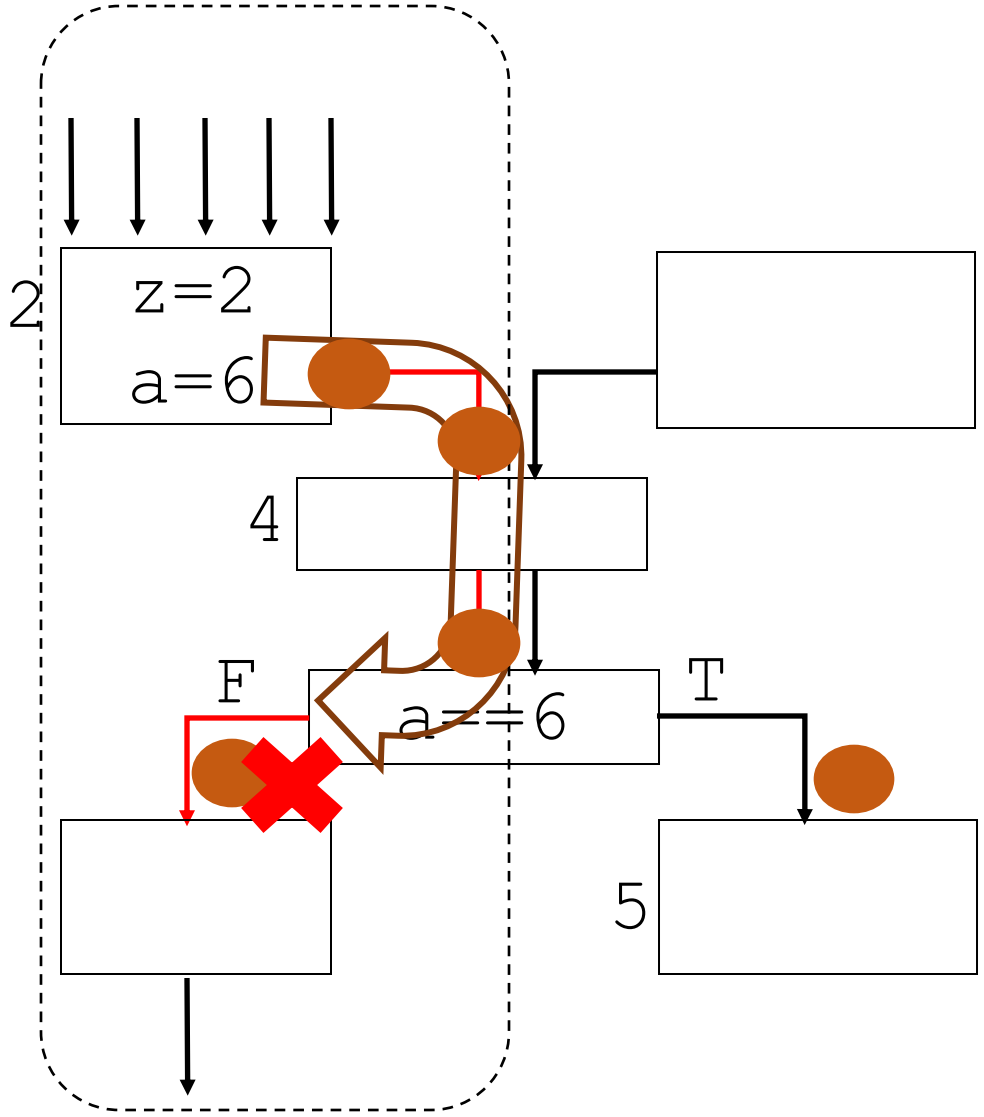


Observation:

We can eliminate the effect of all infeasible paths that **contain the same MIPS** using a single bucket

Node	Value range of z from	
	Feasible Paths	Infeasible segment start red
4		z=[2,2]
5		z=[2,2]
6		z=[2,2]

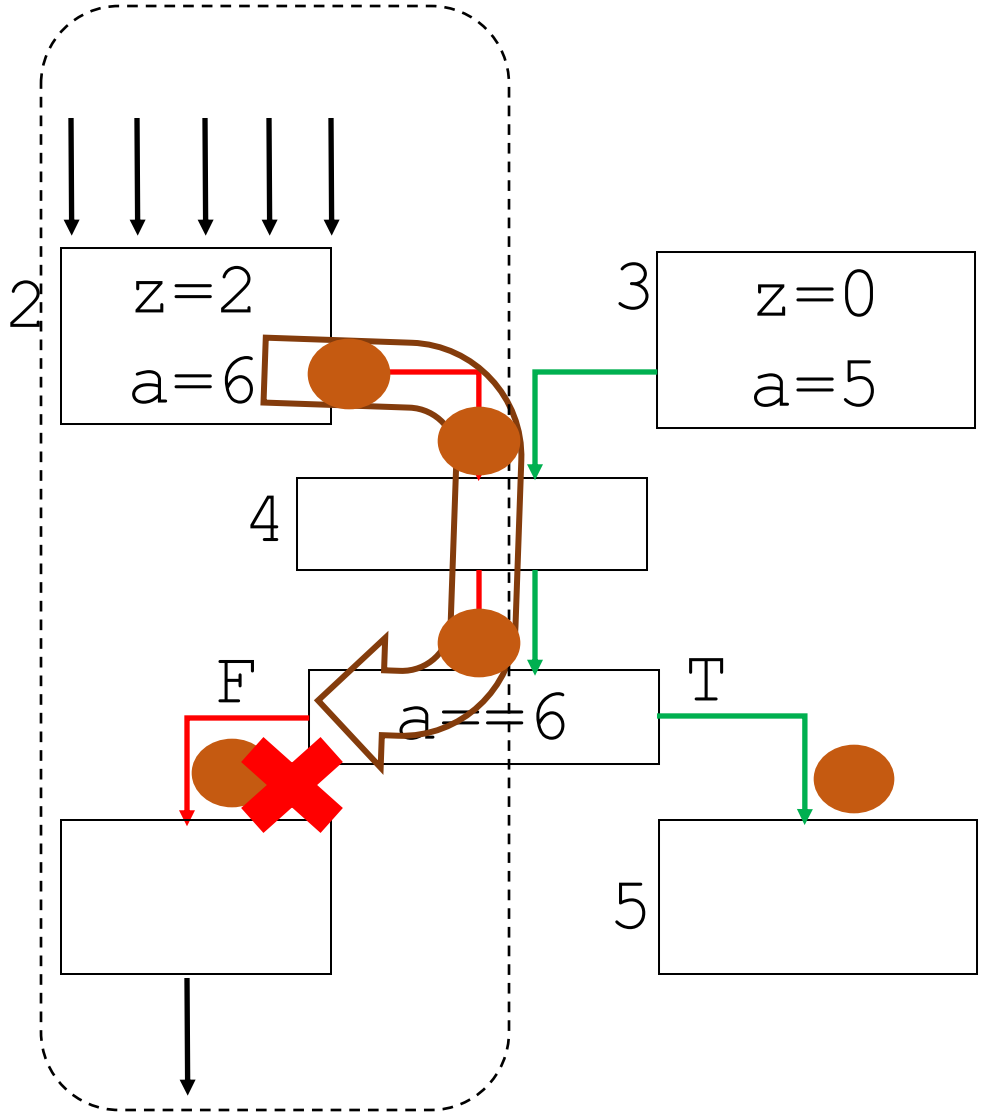
Dealing with Multiple Infeasible Paths



Is one bucket sufficient for **Multiple MIPS**?

Node	Value range of z from	
	Feasible Paths	Infeasible segment start red
4		z=[2,2]
5		z=[2,2]
6		z=[2,2]

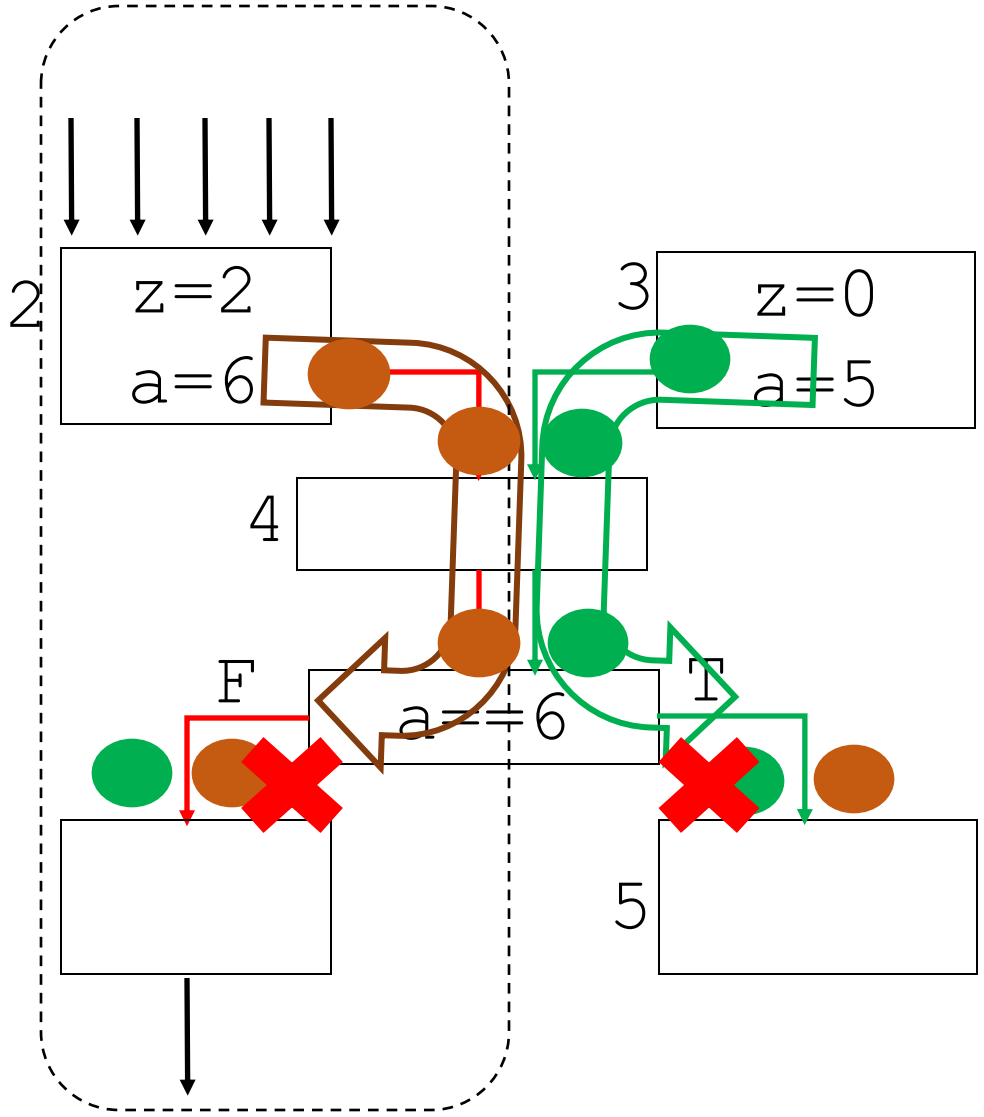
Dealing with Multiple Infeasible Paths



- Green path segment is also a MIPS
- Here one bucket is not sufficient for two MIPS
- Need one bucket per MIPS

Node	Value range of z from	
	Feasible Paths	Infeasible segment start red
4		z=[2,2]
5		z=[2,2]
6		z=[2,2]

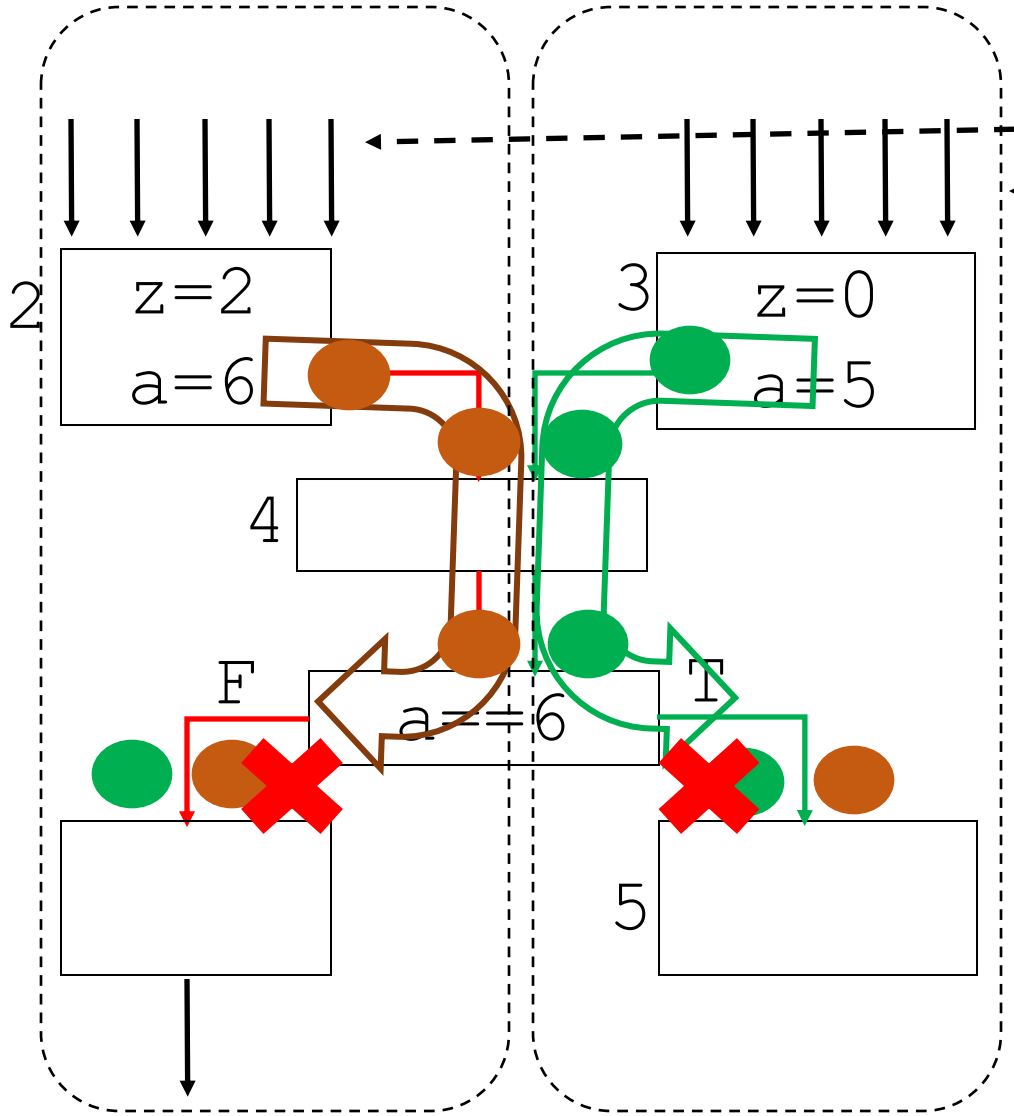
Dealing with Multiple Infeasible Paths



- Green path segment is also a MIPS
- Here one bucket is not sufficient for two MIPS
- Need one bucket per MIPS

Node	Feasible Paths	Value range of z from Infeasible segment start red	Green
4		z=[2,2]	z=[0,0]
5		z=[2,2]	z=[,0]
6		z=[,2]	z=[0,0]

Dealing with Multiple Infeasible Paths



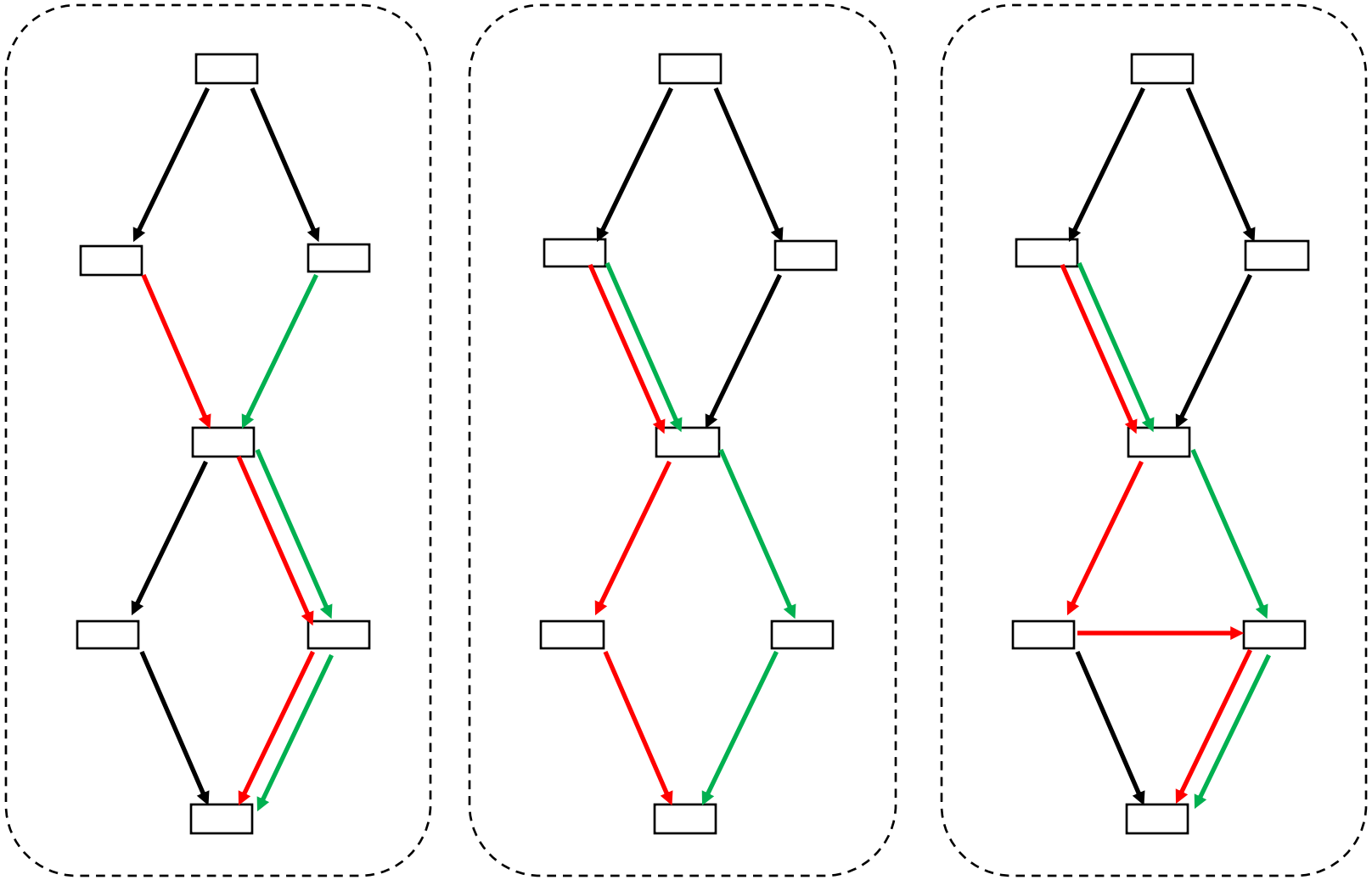
2 buckets eliminated the effect of all these infeasible paths

Node	Feasible Paths	Value range of z from Infeasible segment start red	Green
4		z=[2,2]	z=[0,0]
5		z=[2,2]	z=[,0]
6		z=[,2]	z=[0,0]

Challenges in Computing Precise FPMFP Solutions

- Dealing with Multiple Infeasible Paths
- **Dealing with Overlapping MIPS**
- Dealing with Infeasible Paths across Procedures

Dealing with overlapping MIPS

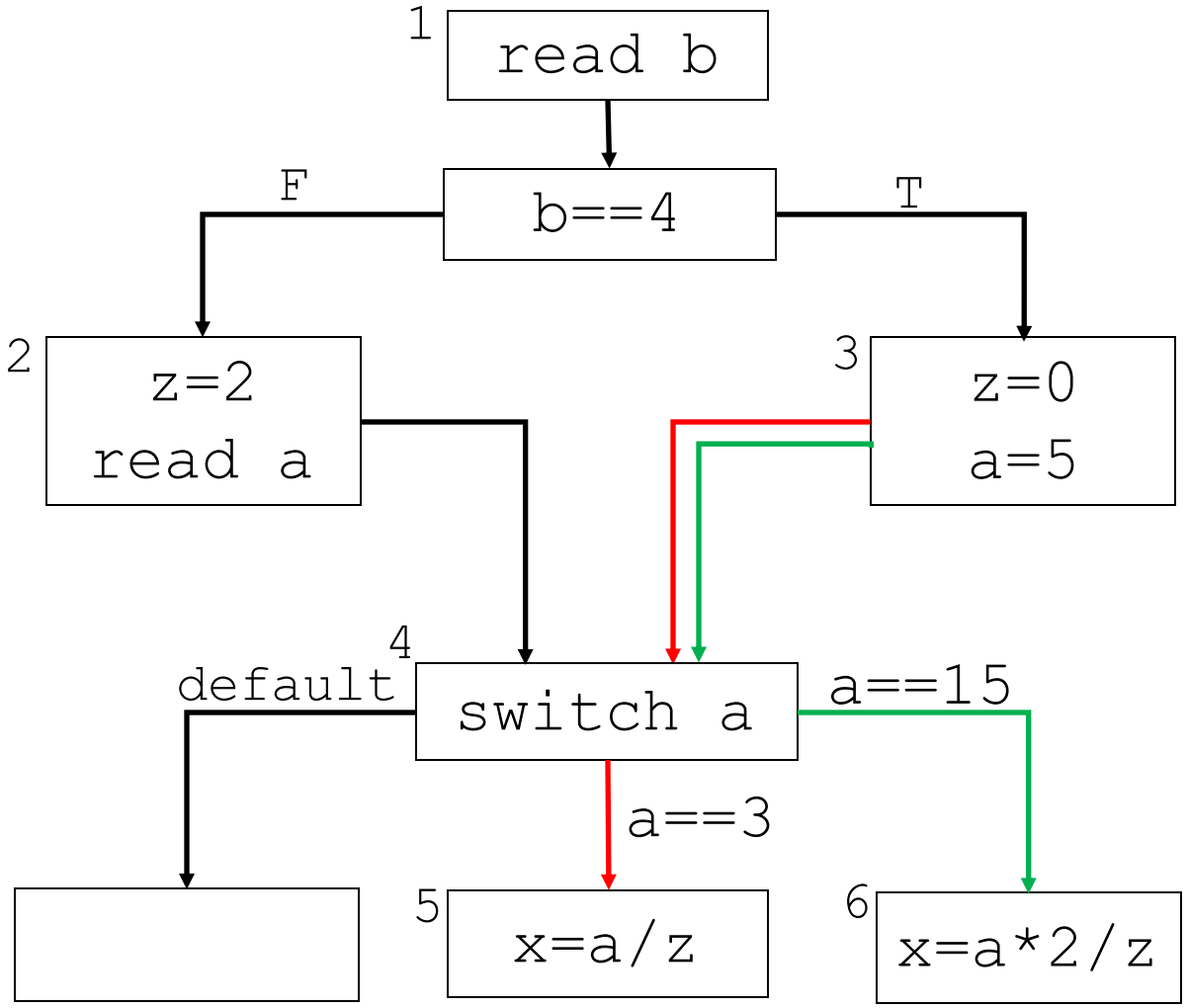


Possible Overlaps between 2 MIPS

Is one bucket per MIPS sufficient always?

...

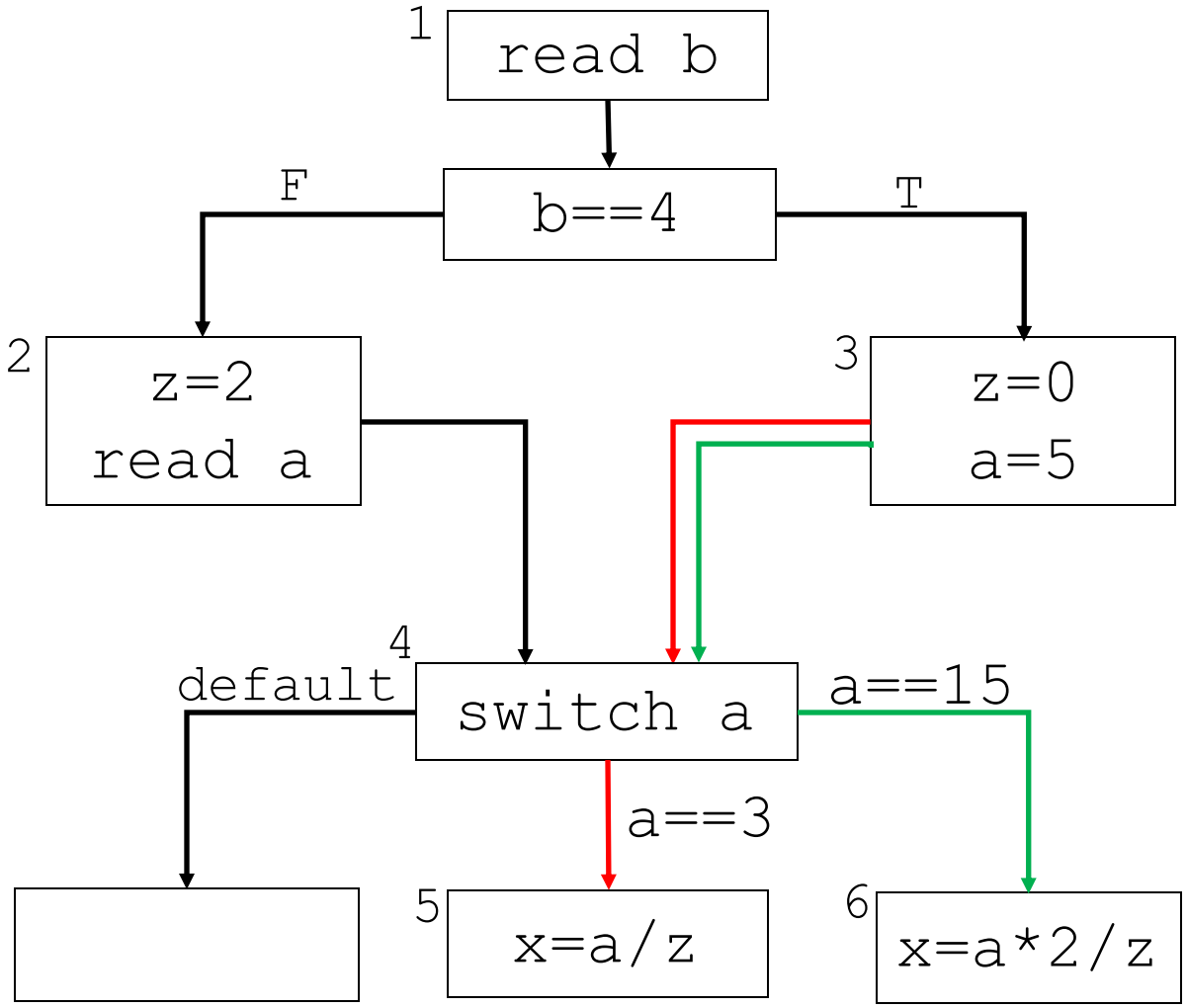
One bucket for each MIPS



Values of z at node 5 ?

Node	Value range of z from		
	Feasible Paths	Infeasible segment start	
		Red	Green
4	z=[2,2]	z=[0,0]	z=[0,0]
5	z=[2,2]	z=[0,0]	z=[0,0]
6	z=[2,2]	z=[0,0]	z=[0,0]

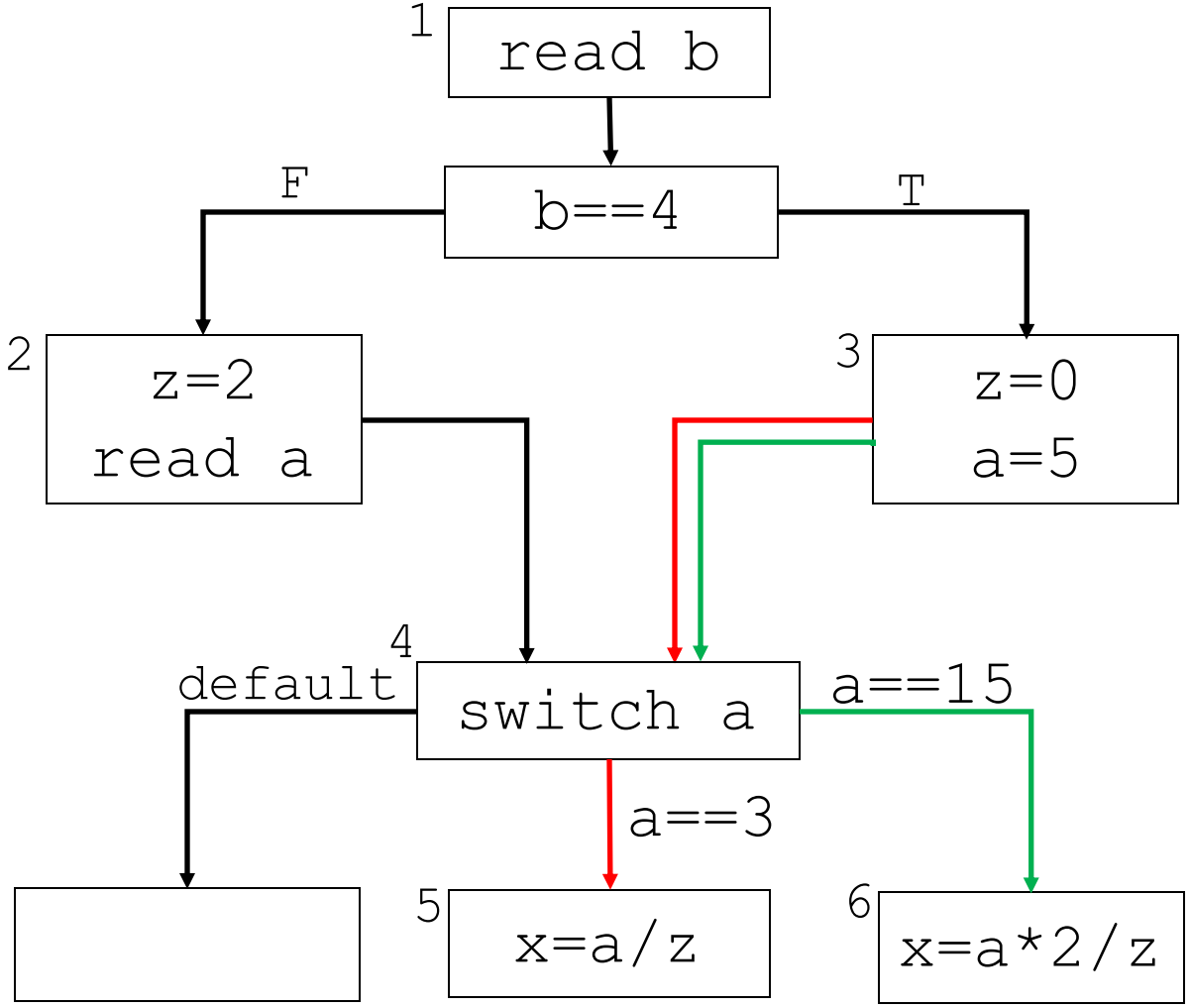
One bucket for each MIPS



Values of z at node 5 ?

Node	Value range of z from		
	Feasible Paths	Infeasible segment start	
		Red	Green
4	z=[2,2]	z=[0,0]	z=[0,0]
5	z=[2,2]	z=[0,0]	z=[0,0]
6	z=[2,2]	z=[0,0]	z=[0,0]

One bucket for each MIPS

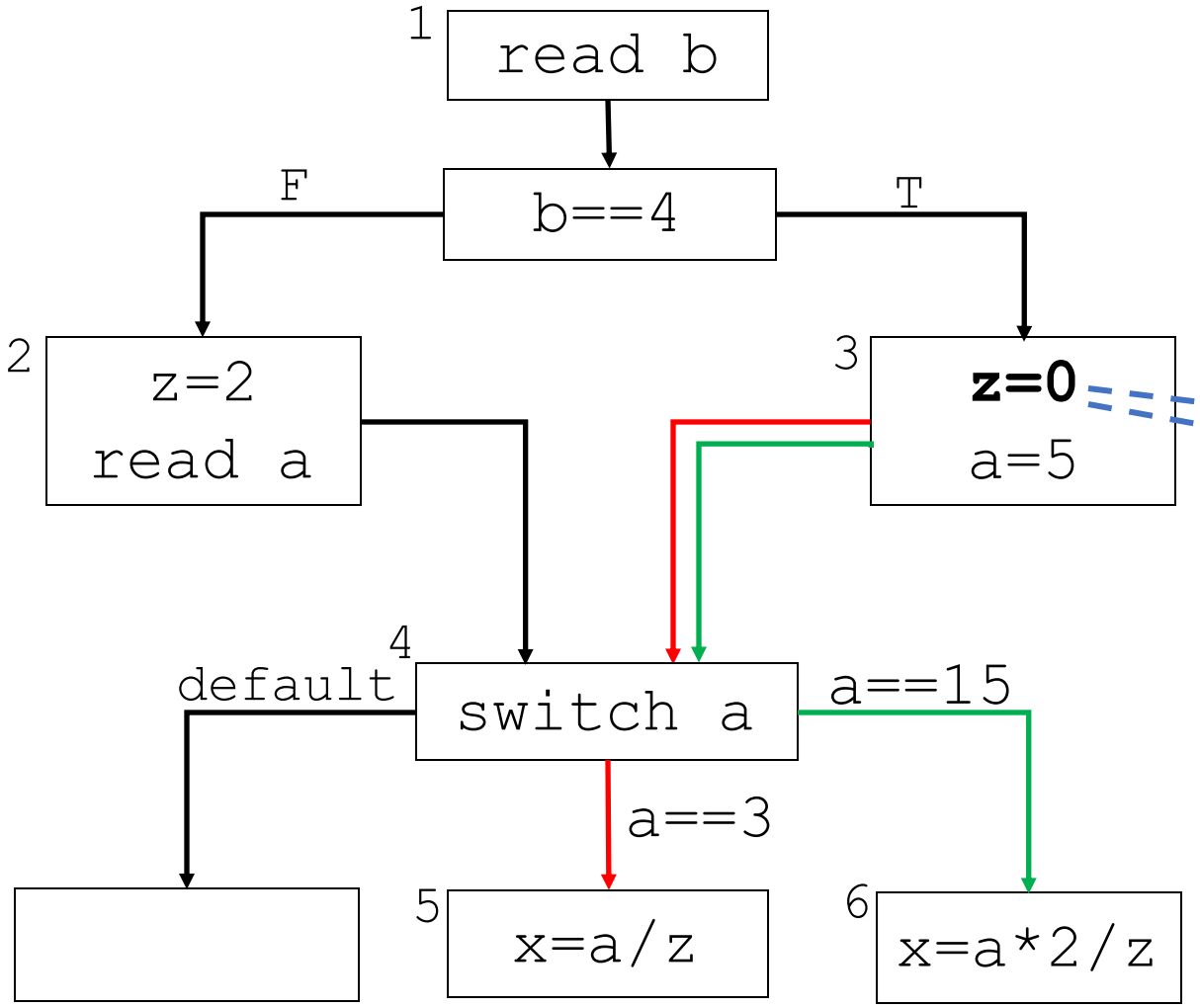


Values of z at node 5 ?

Node	Value range of z from		
	Feasible Paths	Infeasible segment start	
		Red	Green
4	z=[2,2]	z=[0,0]	z=[0,0]
5	z=[2,2]	z=[0,0]	z=[0,0]
6	z=[2,2]	z=[0,0]	z=[0,0]

Imprecise

What went wrong?

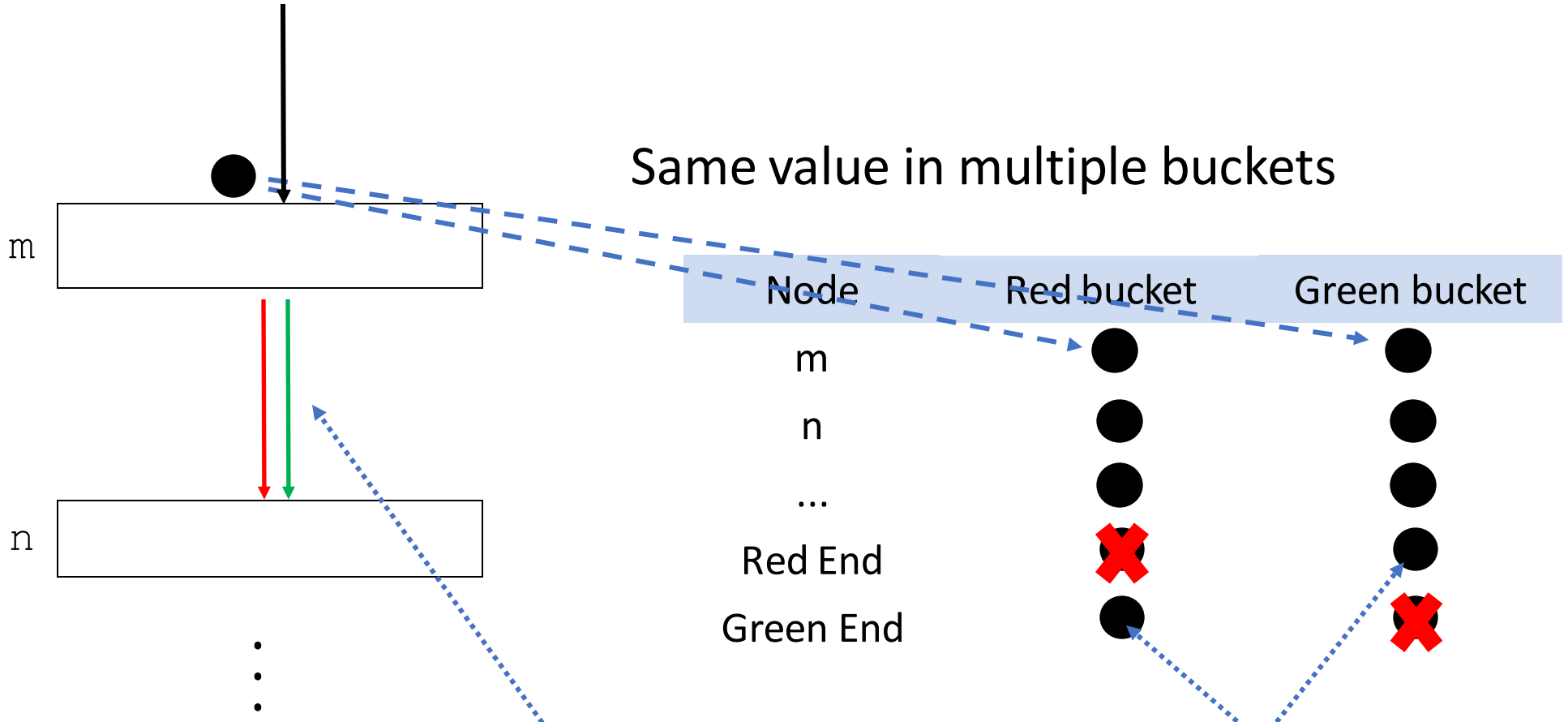


- The same value appears in multiple buckets

Node	Value range of z from		
	Feasible Paths	Infeasible segment start	
		Red	Green
4	z=[2,2]	z=[0,0]	z=[0,0]
5	z=[2,2]	z=[✗,0]	z=[0,0]
6	z=[2,2]	z=[0,0]	z=[✗,0]

- Blocking from one bucket is not sufficient

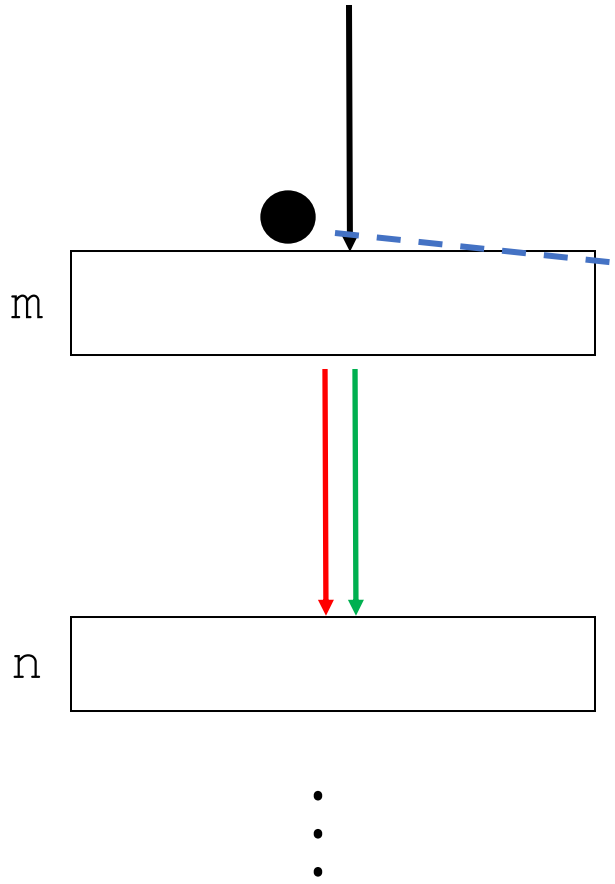
Intuitively, we got a covering, instead of a partition



So the value does not get completely blocked at trigger edge

Start edges of red and green MIPS

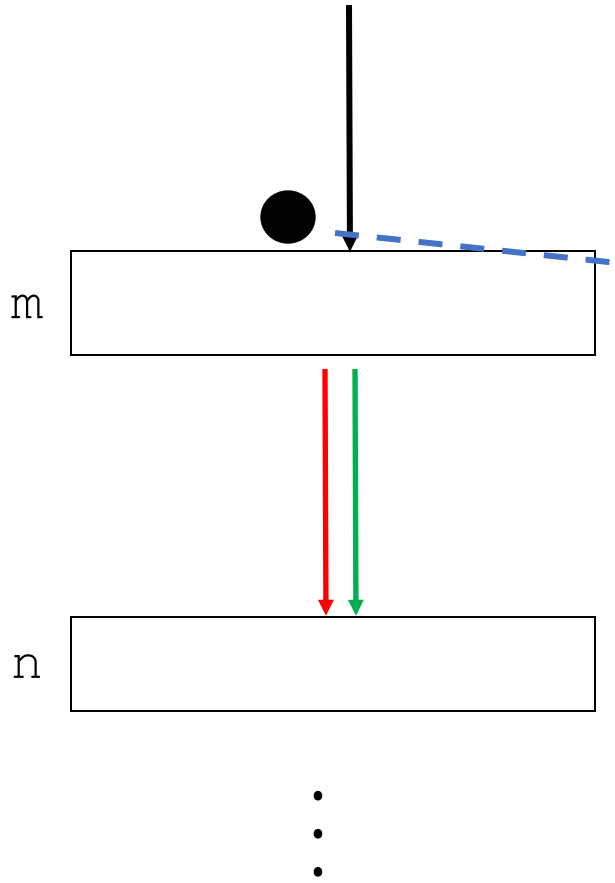
Create bucket representing combination of MIPS



Assign value to a single bucket

Node	Red bucket	Green bucket	Red • Green bucket
m			
n			
...			
Red End			
Green End			

Create bucket representing combination of MIPS



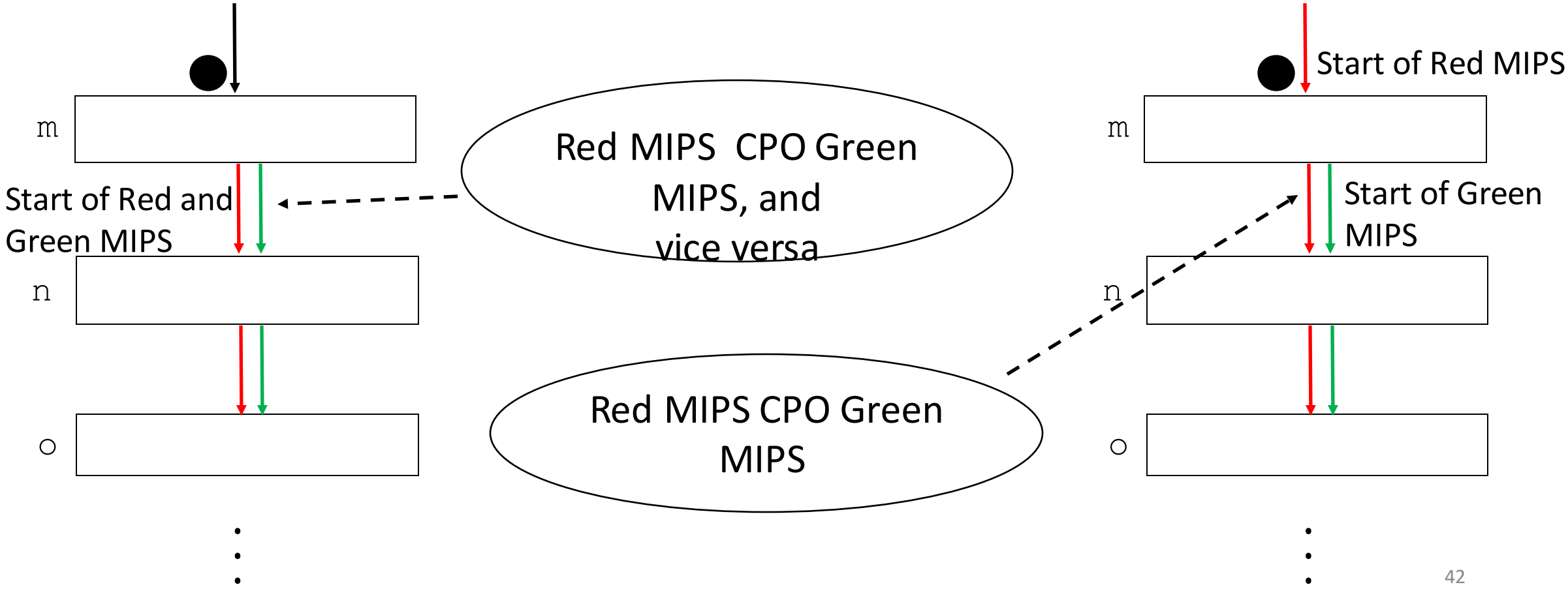
Assign value to a single bucket

Node	Red bucket	Green bucket	Red • Green bucket
m			
n			
...			
Red End			
Green End			

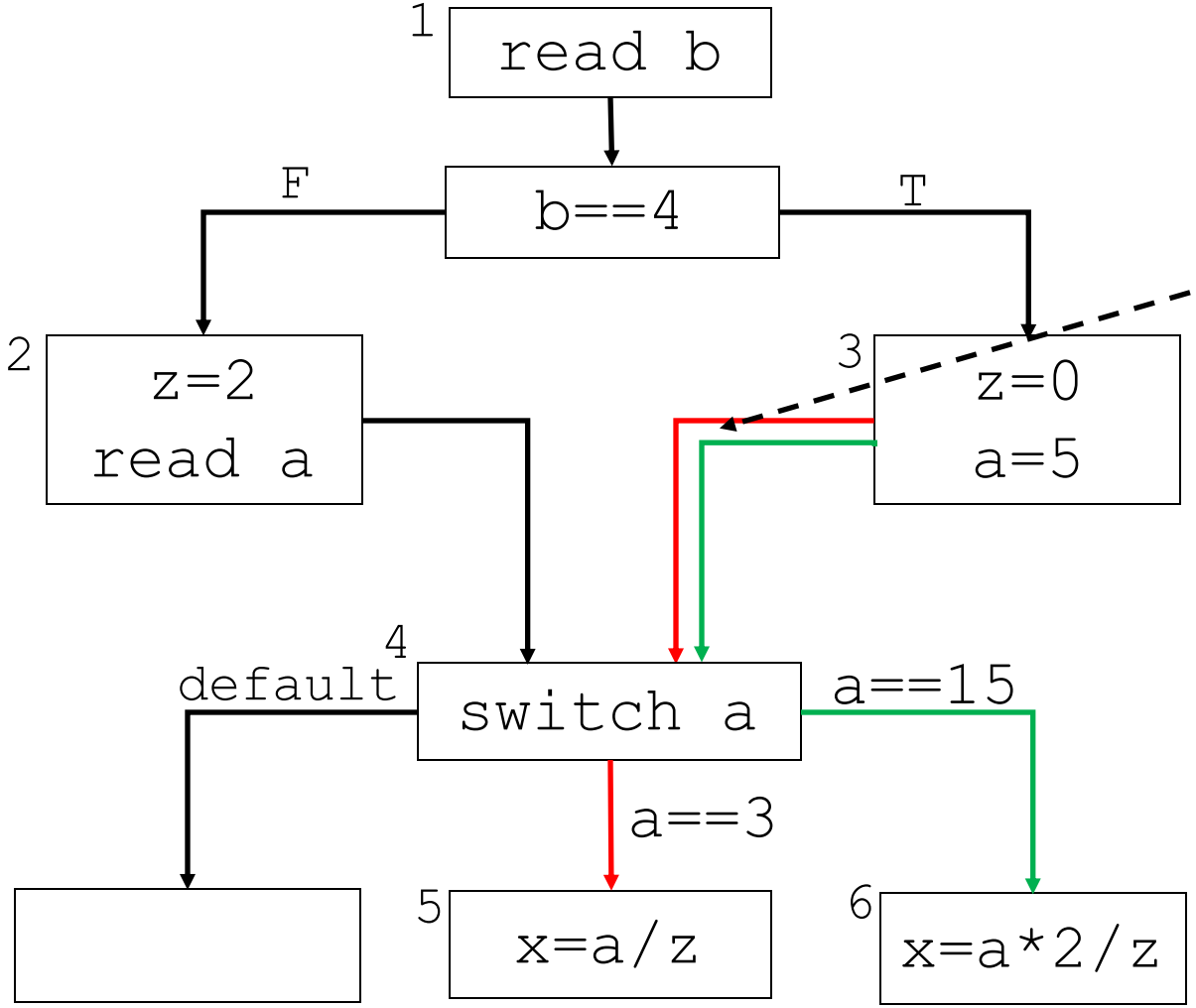
Do we need to create composite buckets for **all combinations of MIPS**?

Observation:

We **only** need to consider cases when **A MIPS contains-prefix-of (CPO) other MIPS** because in such cases they contain the same values in corresponding buckets.



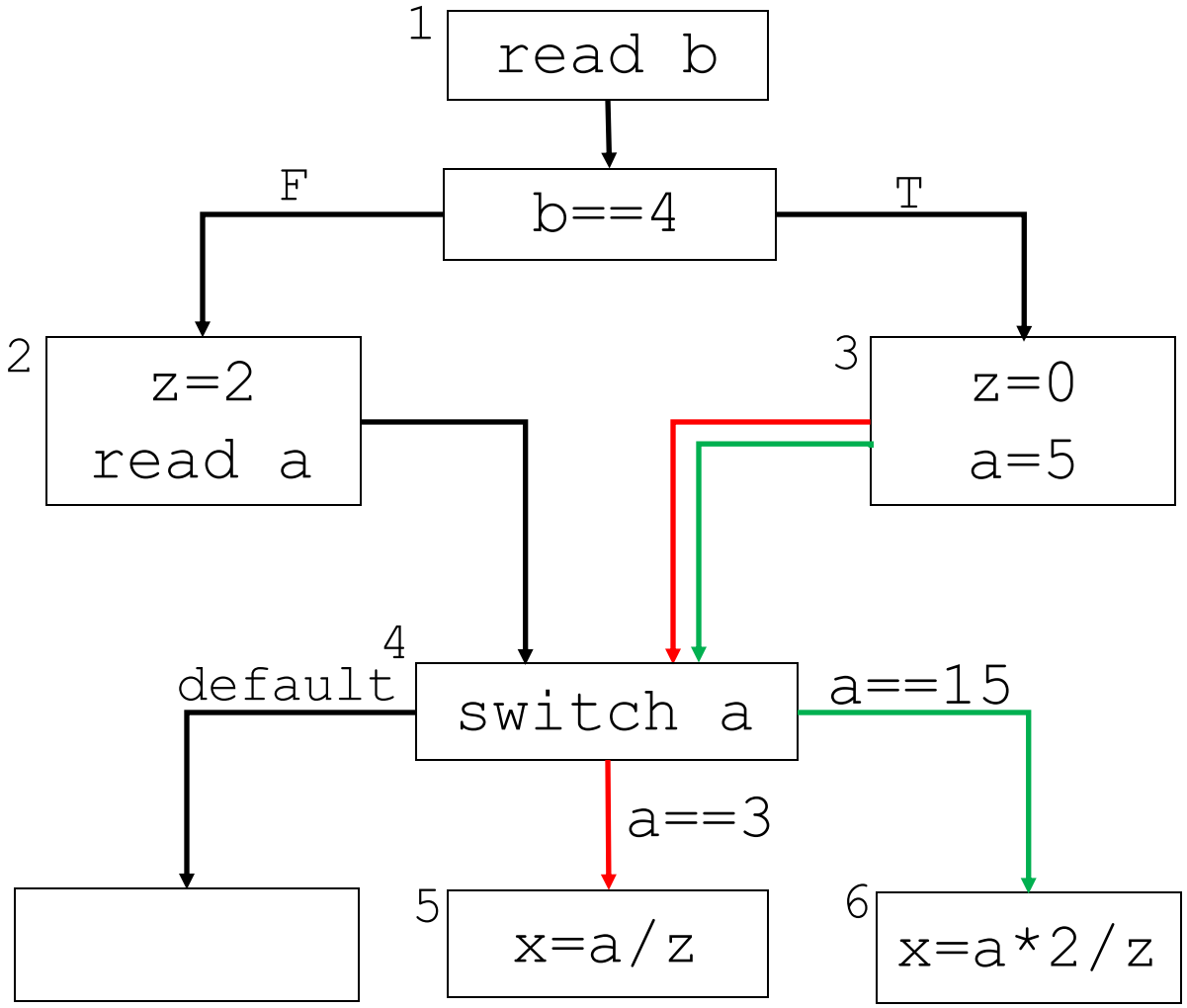
Our running example



Red contains-prefix-of green
So create a composite bucket

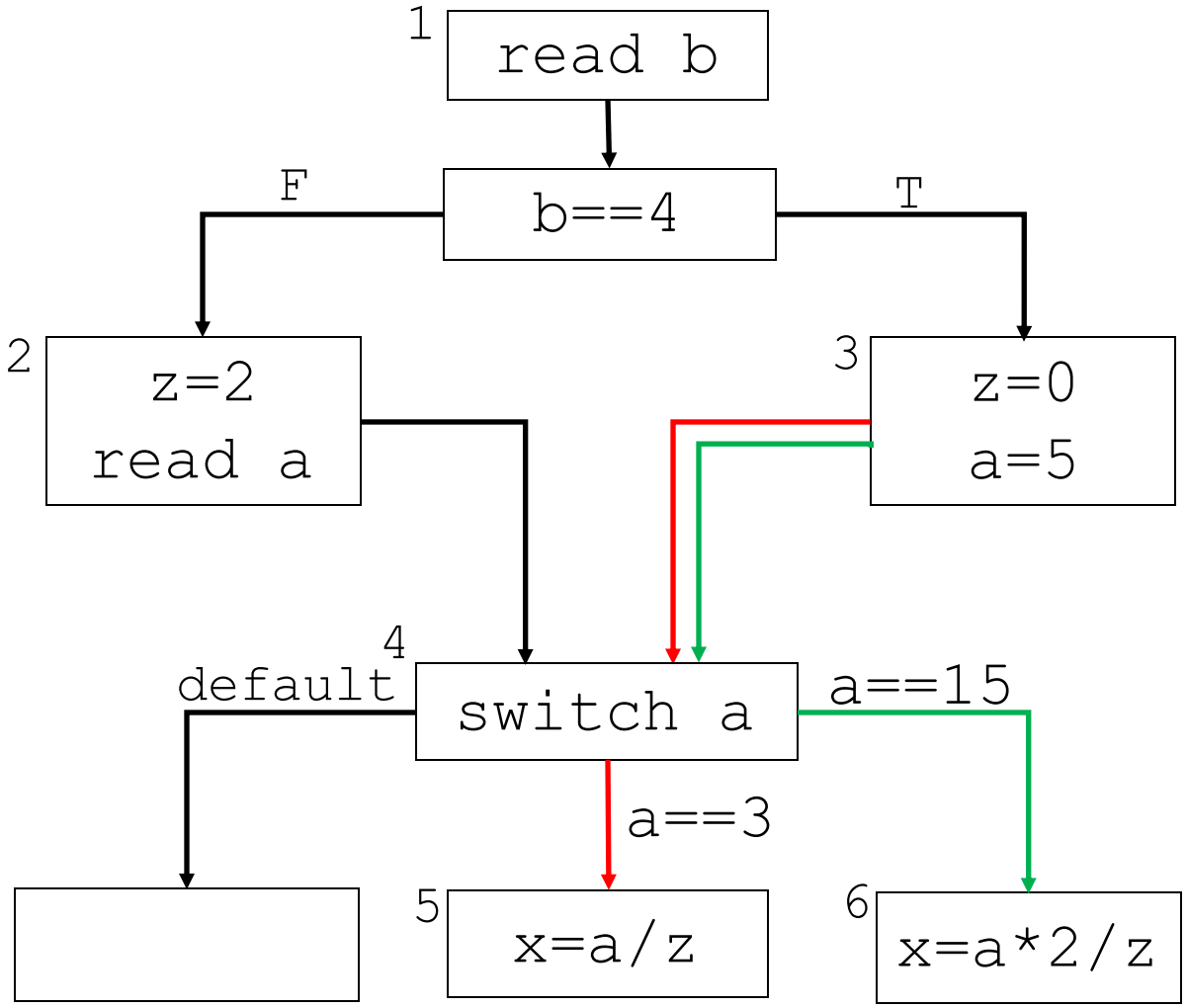
Node	Feasible Paths	Value range of z from Infeasible segment start		
		Red	Green	Red•Green

Assign the value to composite bucket



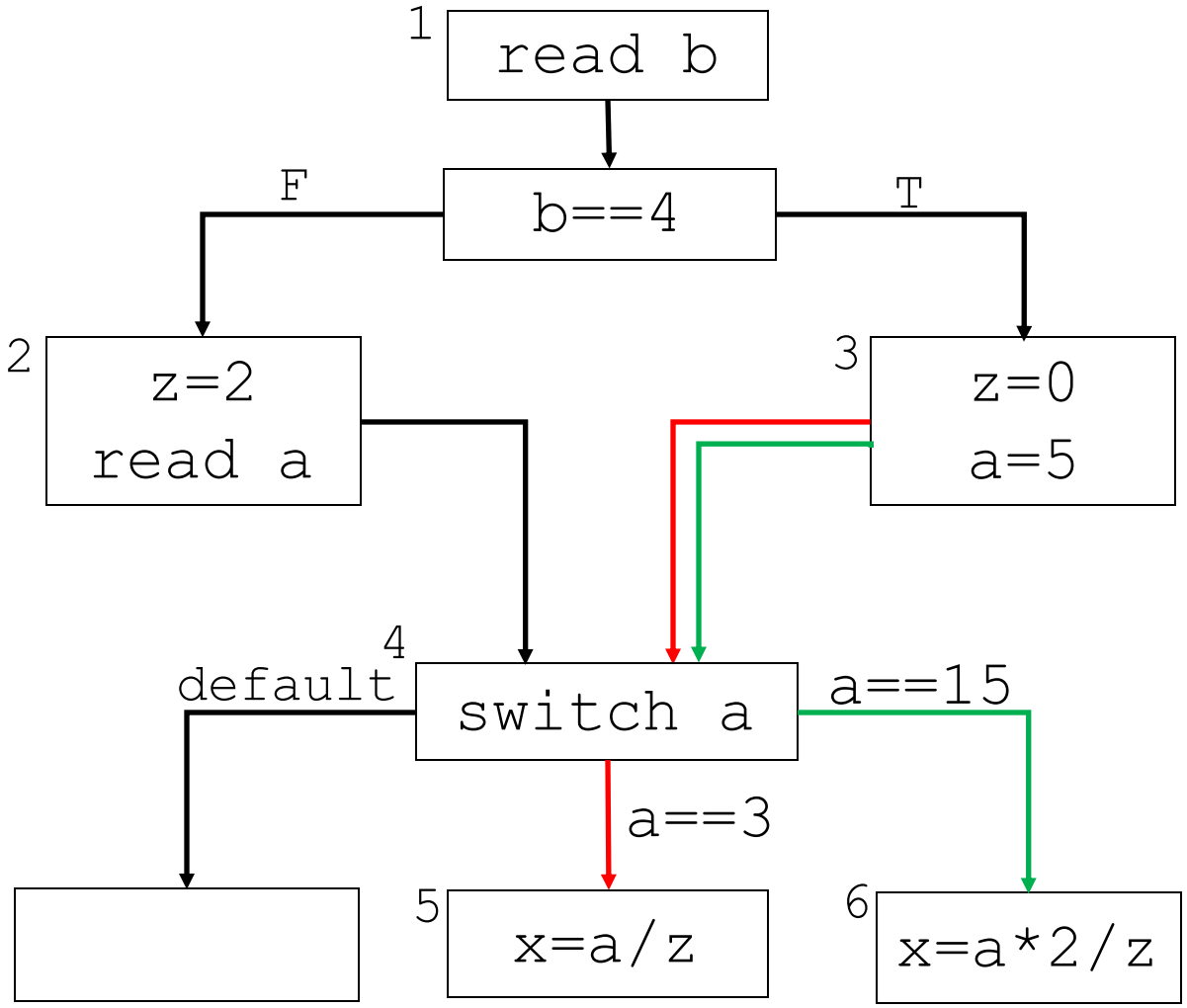
Node	Feasible Paths	Value range of z from Infeasible segment start		
		Red	Green	Red•Green
3				z=[0,0]

Move the value when overlap changes



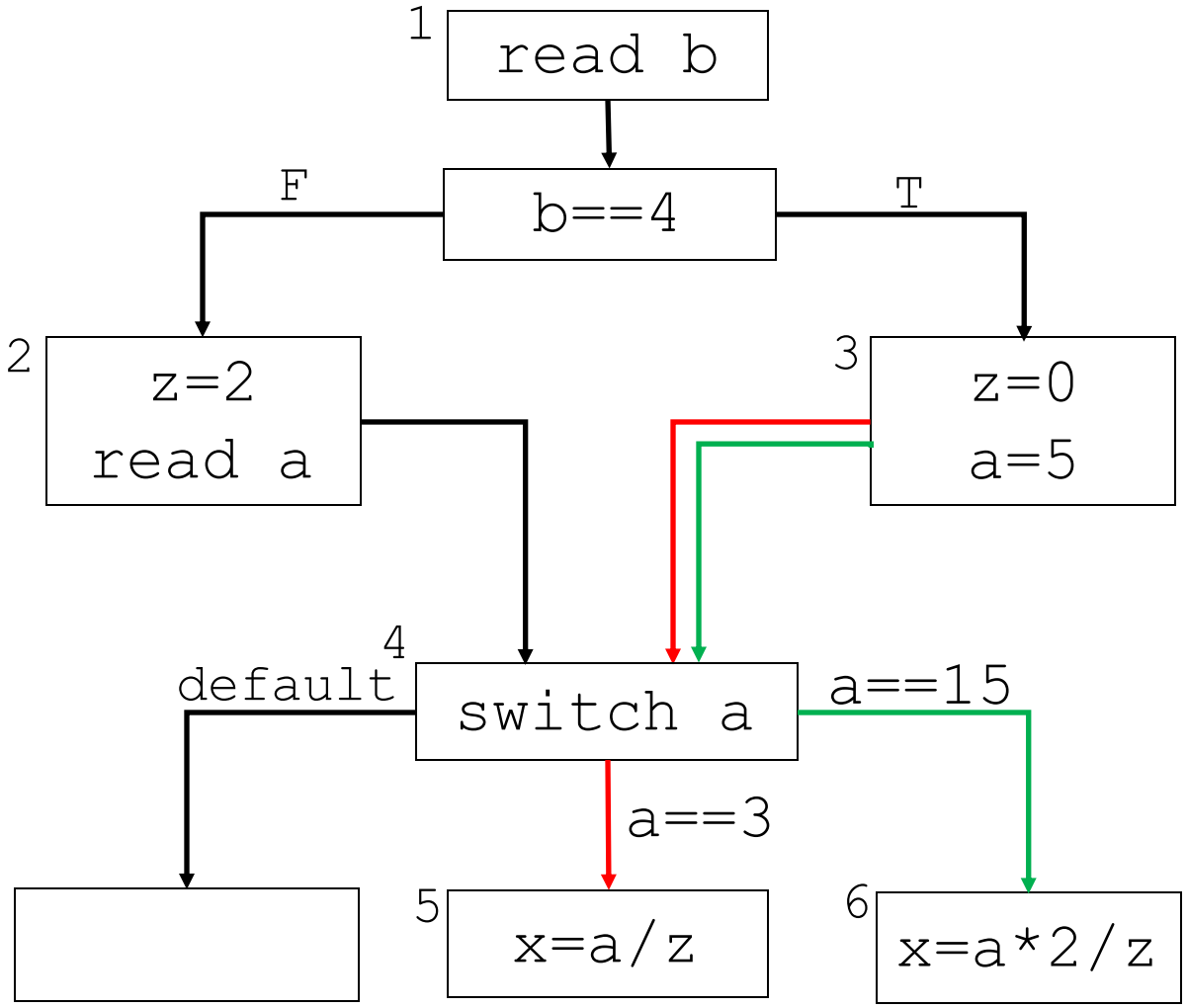
Node	Feasible Paths	Value range of z from Infeasible segment start		
		Red	Green	Red•Green
3				z=[0,0]
5	z=[2,2]	z=[0,0]		
6	z=[2,2]		z=[0,0]	

Move the value when overlap changes



Node	Feasible Paths	Value range of z from Infeasible segment start		
		Red	Green	Red•Green
3				z=[0,0]
5	z=[2,2]	z=[,0]		
6	z=[2,2]		z=[,0]	

Move the value when overlap changes



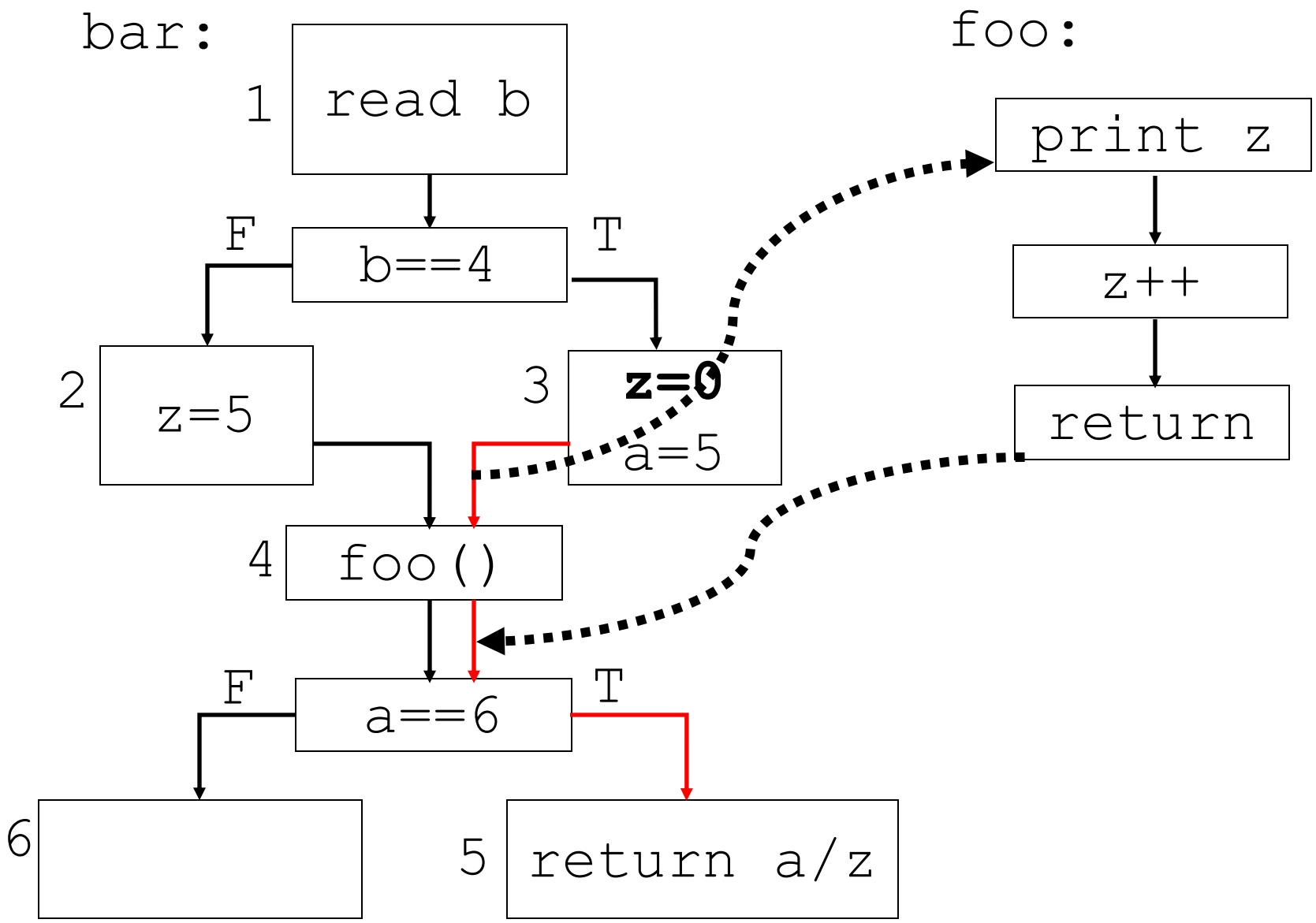
Precise value

Node	Feasible Paths	Value range of z from Infeasible segment start		
		Red	Green	Red•Green
3				z=[0,0]
5	z=[2,2]	z=[,0]		
6	z=[2,2]		z=[,0]	

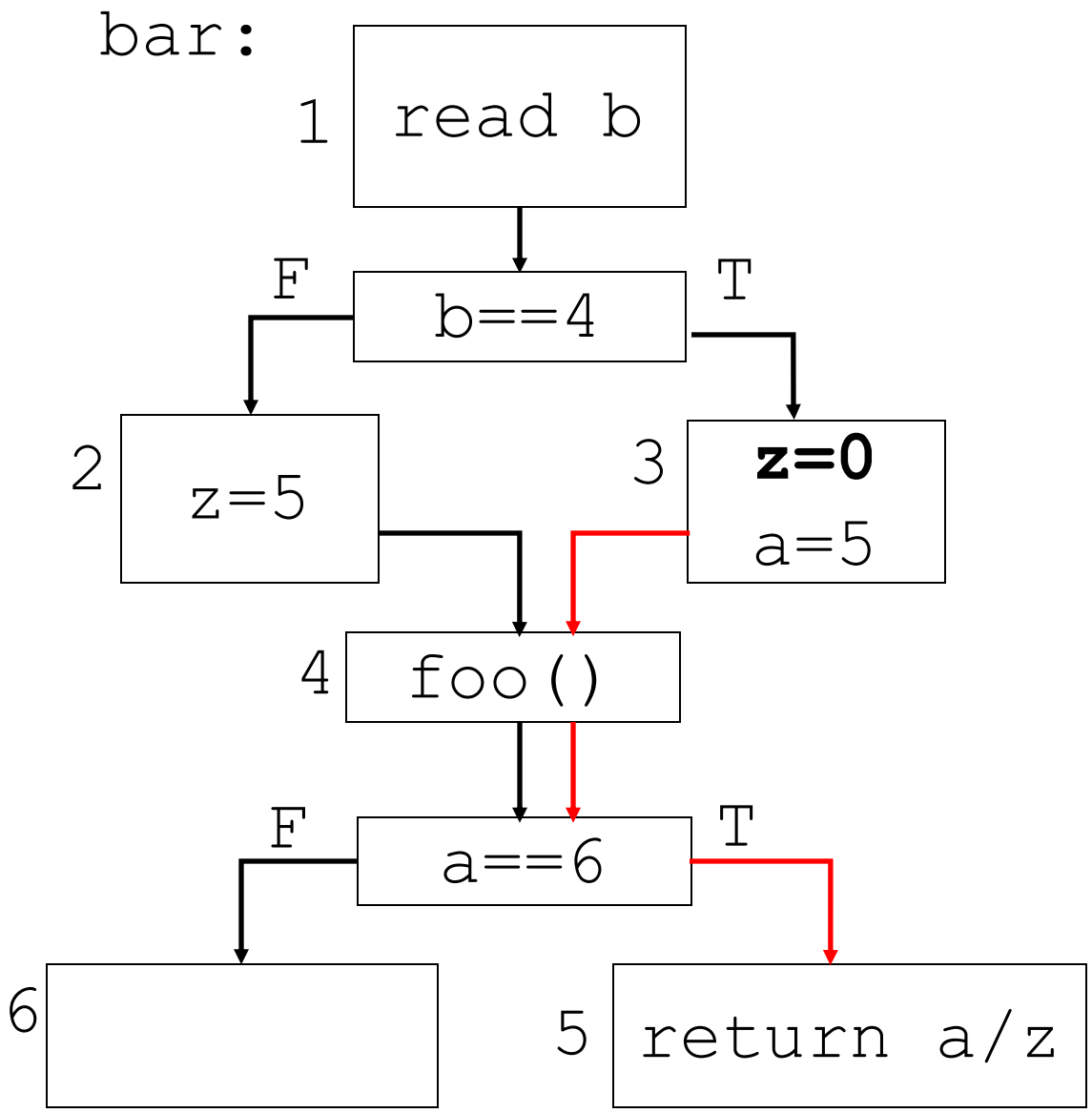
Challenges in Computing Precise FPMFP Solutions

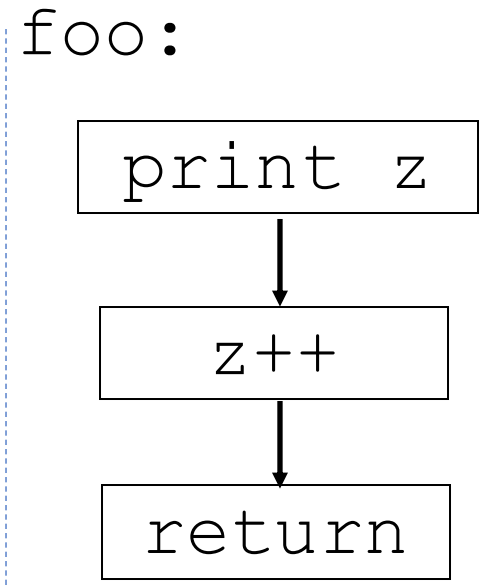
- Dealing with Multiple Infeasible Paths
- Dealing with Overlapping MIPS
- **Dealing with Infeasible Paths across Procedures**

Functional Approach for FPMFP Computation



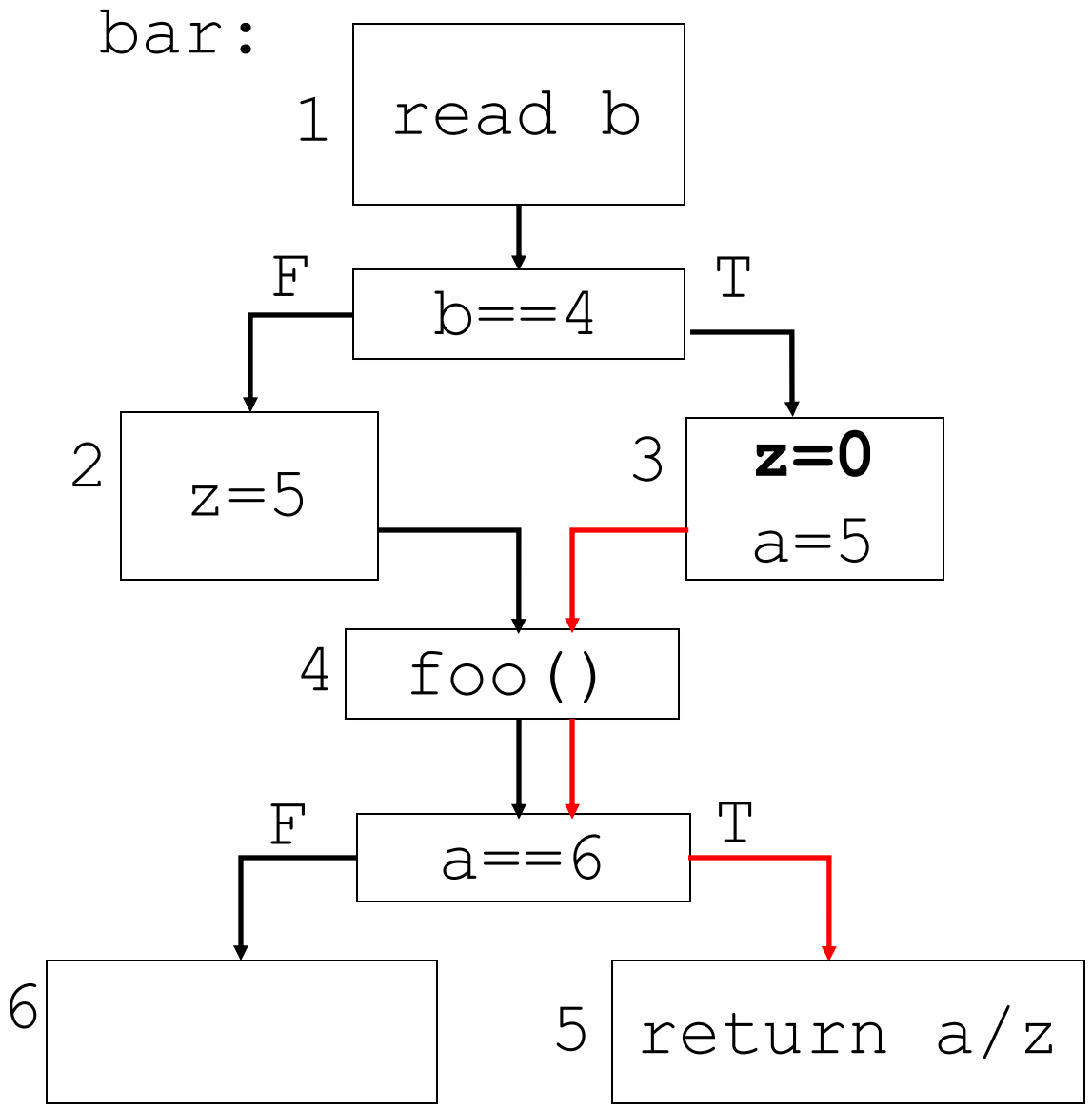
Compute GEN and KILL Summary for Functions

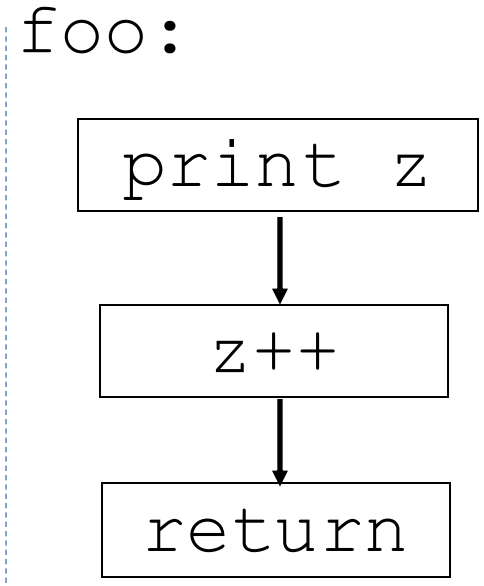




- Uses constant boundary value
- Traverses call graph in bottom-up order

Compute GEN and KILL Summary for Functions

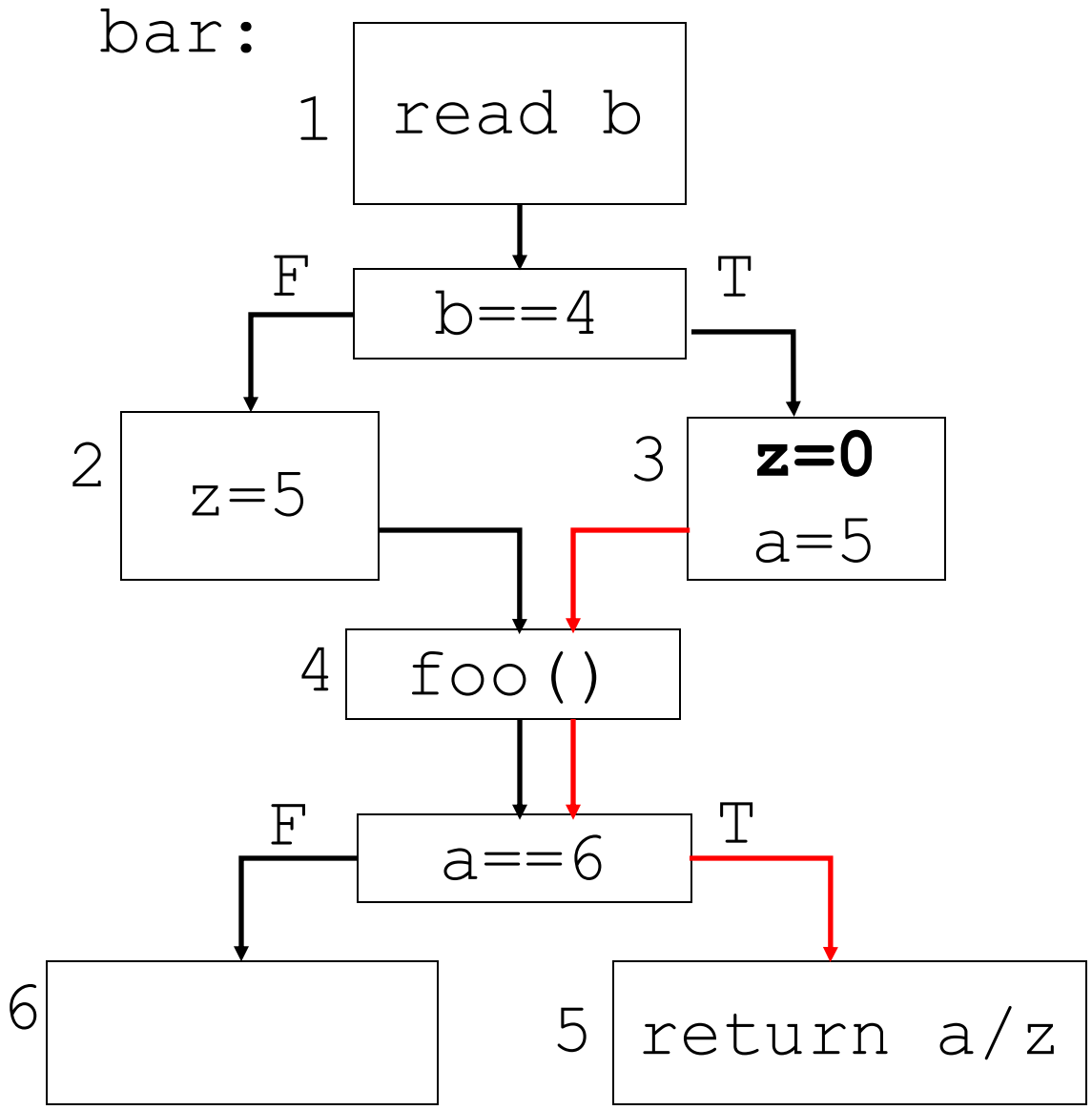




foo

- GEN Summary : z++
- KILL Summary : {z}
  (contains modified variables)

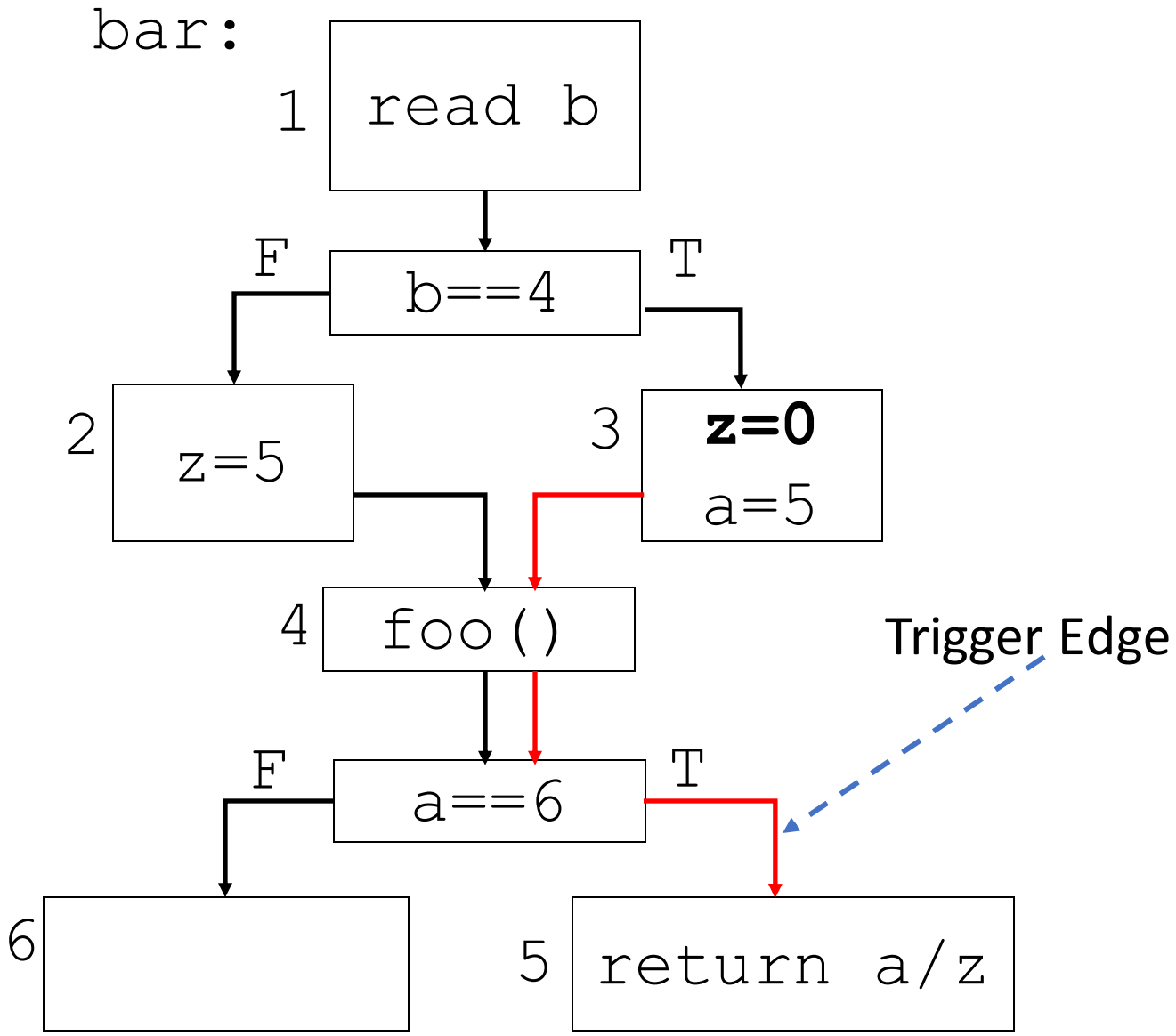
Substitute Summary in FPMFP Computation



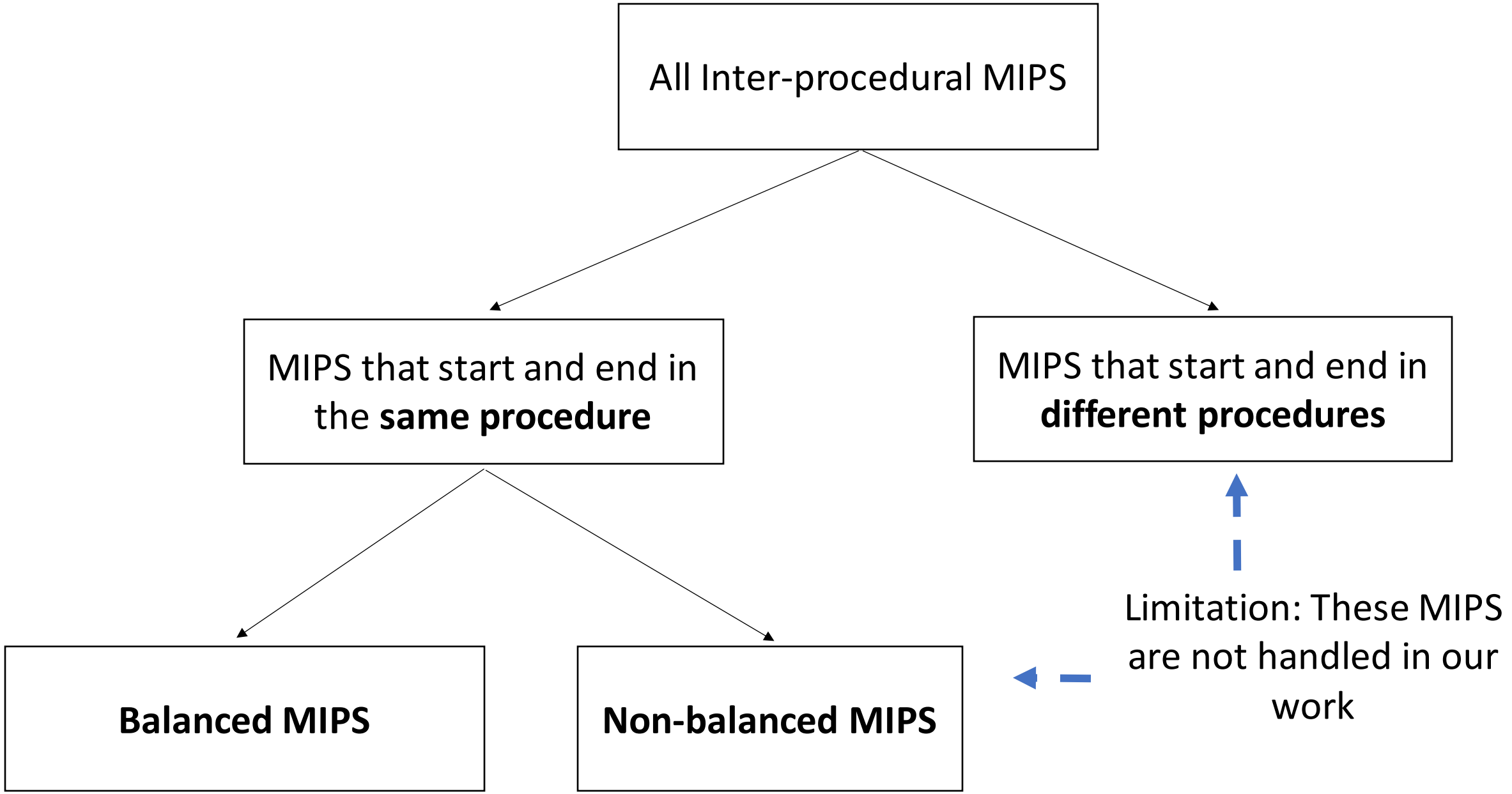
- OUT4 is computed using summary of foo and IN4.

Node	Value range of z from	
	Feasible Paths	Red
IN4	z=[5,5]	z=[0,0]
OUT4	z=[6,6]	z=[1,1]
5	z=[6,6]	z=[1,1]
6	z=[6,6]	z=[1,1]

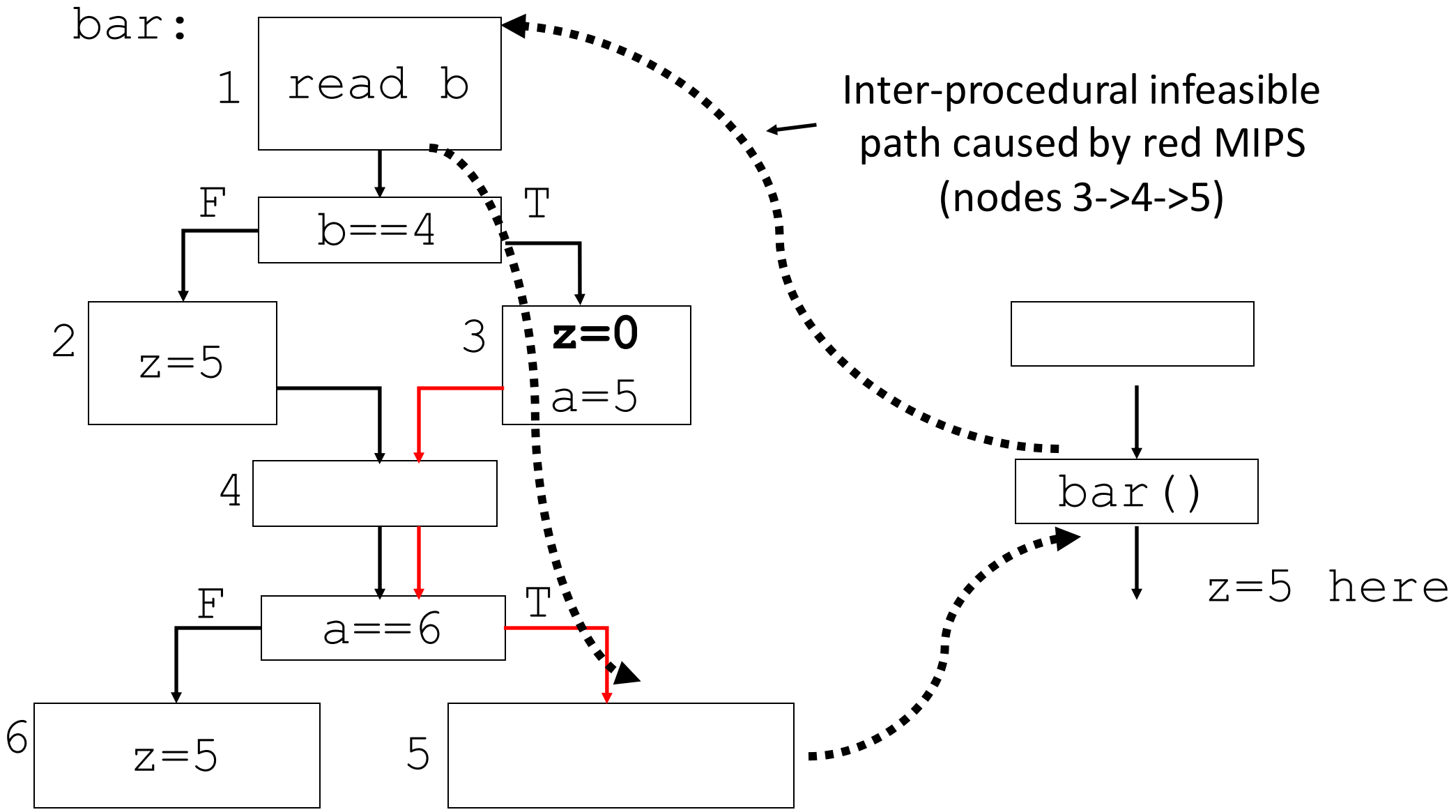
Eliminates effect of balanced MIPS



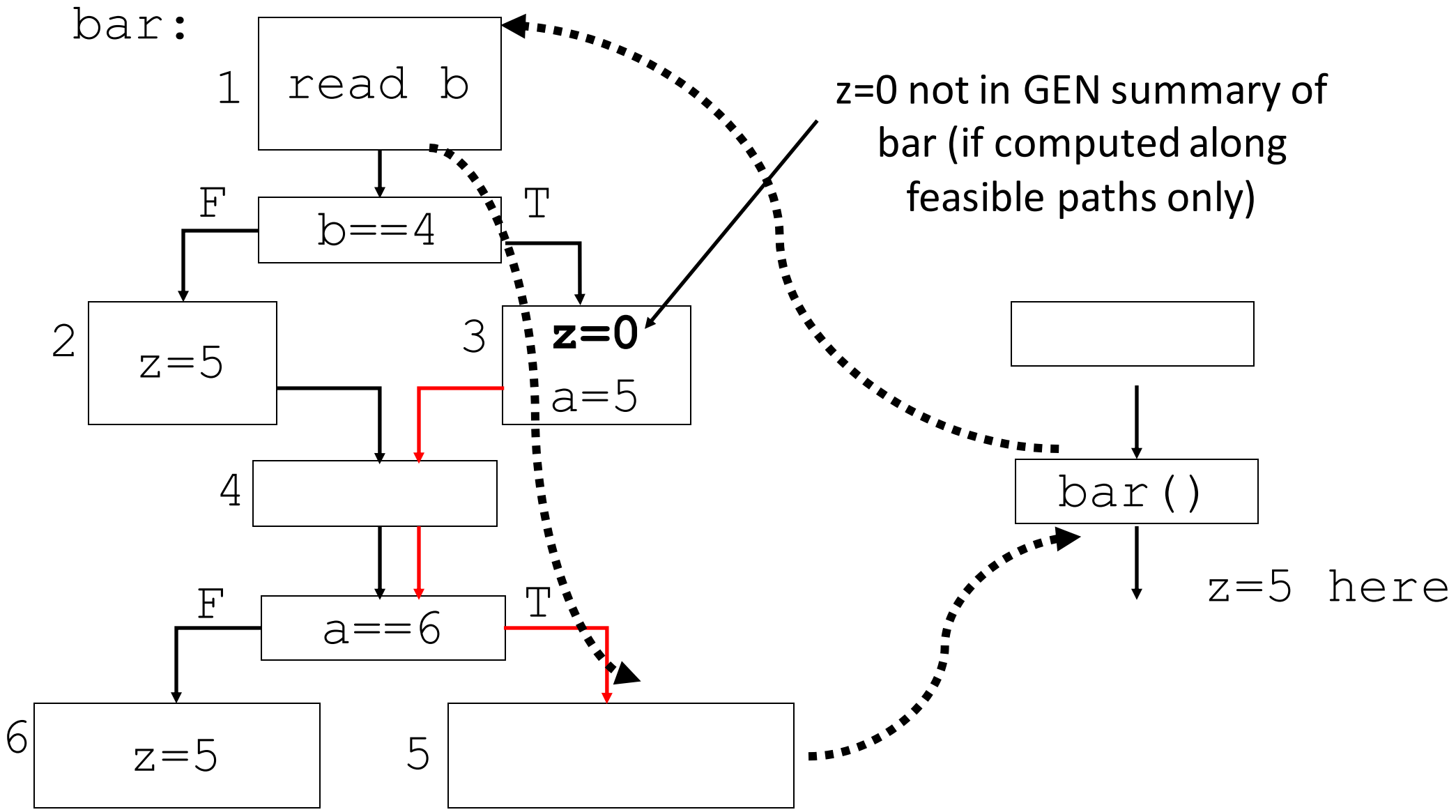
- **The variables** in the condition on the Trigger edge are not modified at intermediate nodes of a balanced MIPS
- Condition on trigger edge (a==6)
- "a" is not modified along Red MIPS


All Inter-procedural MIPS
MIPS that start and end in the **same procedure**
MIPS that start and end in **different procedures**
Balanced MIPS
Non-balanced MIPS
Limitation: These MIPS are not handled in our work

Benefit of summaries along feasible paths



Benefit of summaries along feasible paths

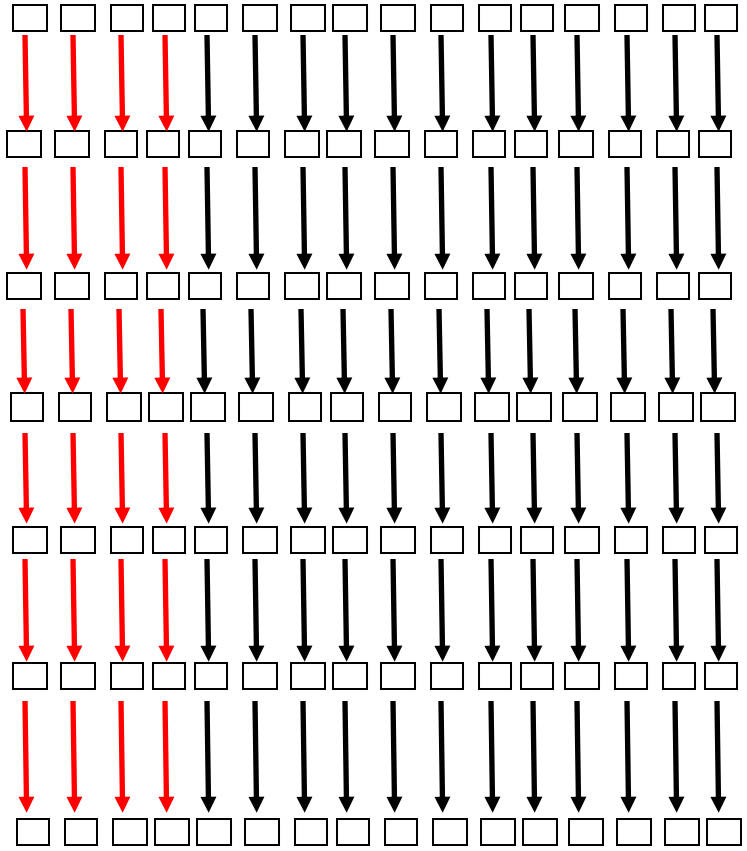


1. Data Flow Analysis and Its Existing Solutions

2. Improving Precision of Data Flow Analysis

3. Improving Scalability of Data Flow Analysis

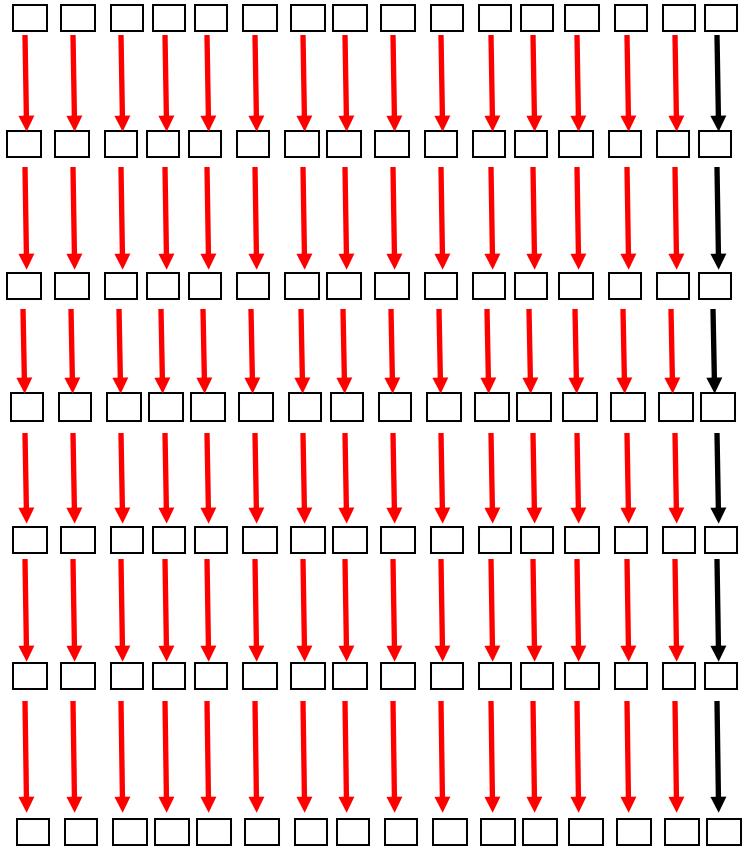
The Good Thing is...



Many Infeasible Paths Exist (9-40% [1]), so scope for precision improvement

[1] Rastislav Bodik, Rajiv Gupta, and Mary Lou Soa. 1997. Refining data flow information using infeasible paths. In Software Engineering ESEC/FSE'97. Springer, 361–377.

The Not So Good Thing is…



Too Many Infeasible Paths May Exist
(> 99%, theoretically)

Therefore, the challenge is …

1. Reducing infeasible path detection cost

2. Reducing the number of Buckets

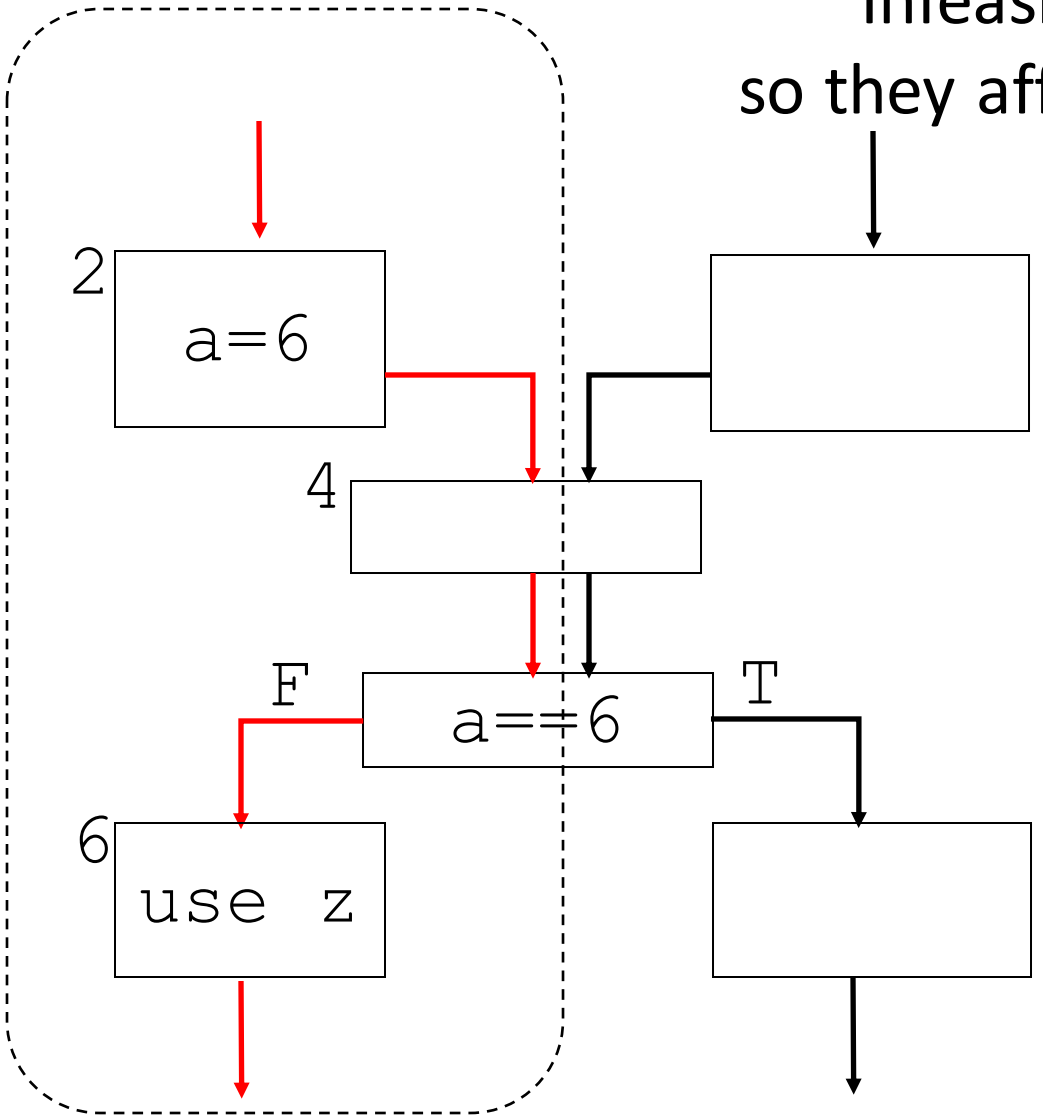
Node	**Bucket1**	**Bucket2**	**…**	**Bucket∞**

1. Reducing infeasible path detection cost

Observation:

Infeasible paths is a property of programs

so they affect all flow sensitive data flow analysis



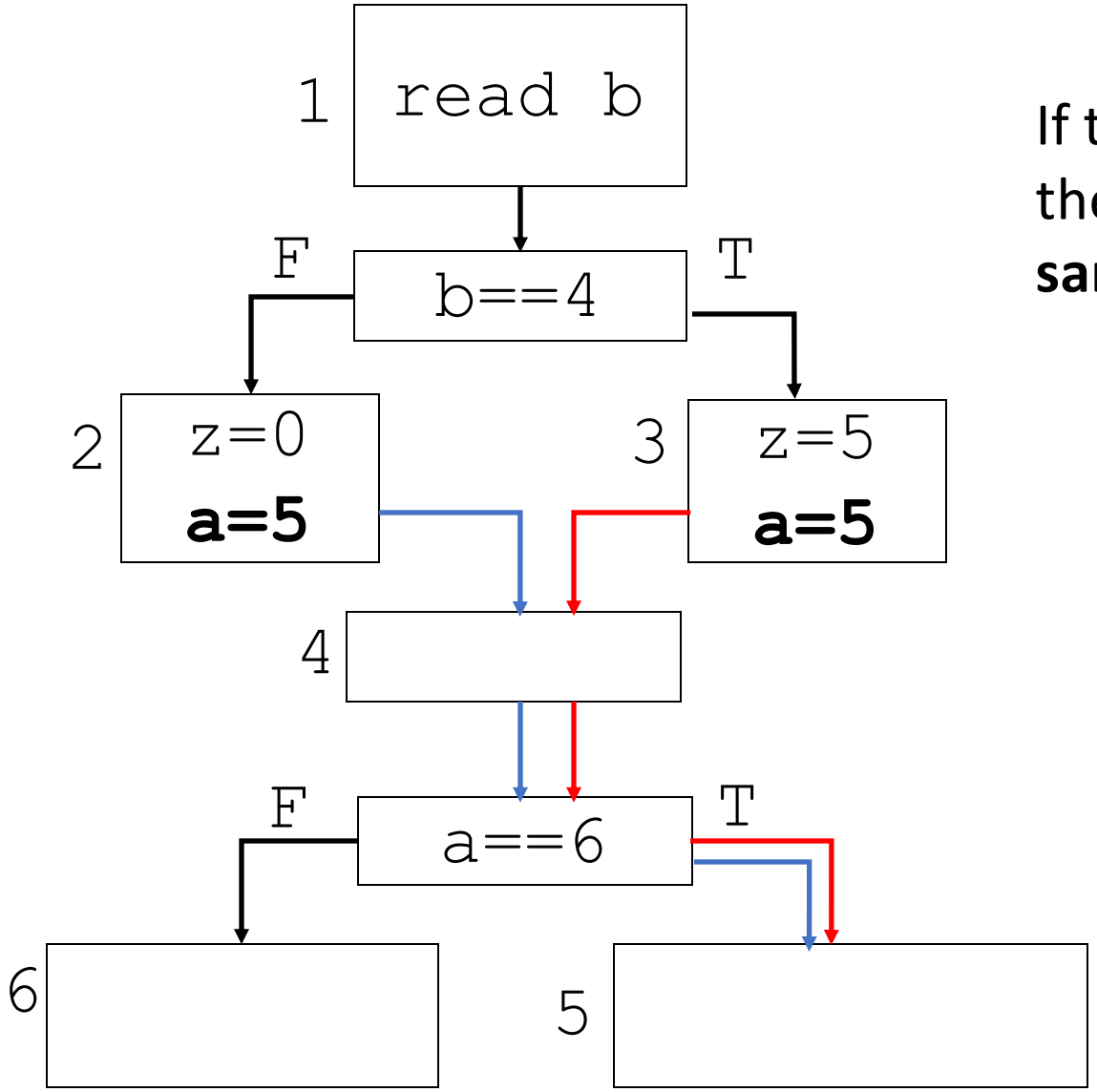
- *For example on LHS*
  -Reaching definition
  (a=6 doesn't reach node 3)
  -Liveness (z is not live at node 1)
  - ….

- *So detect infeasible paths only once for a program (instead of for each data flow analysis)*

2. Reducing the number of buckets

- **Reduce Distinct MIPS using MIPS Equivalence**
- Reduce Number of Composite Buckets
- Use Equality of Data Flow Values to Merge Buckets

Use of MIPS Equivalence

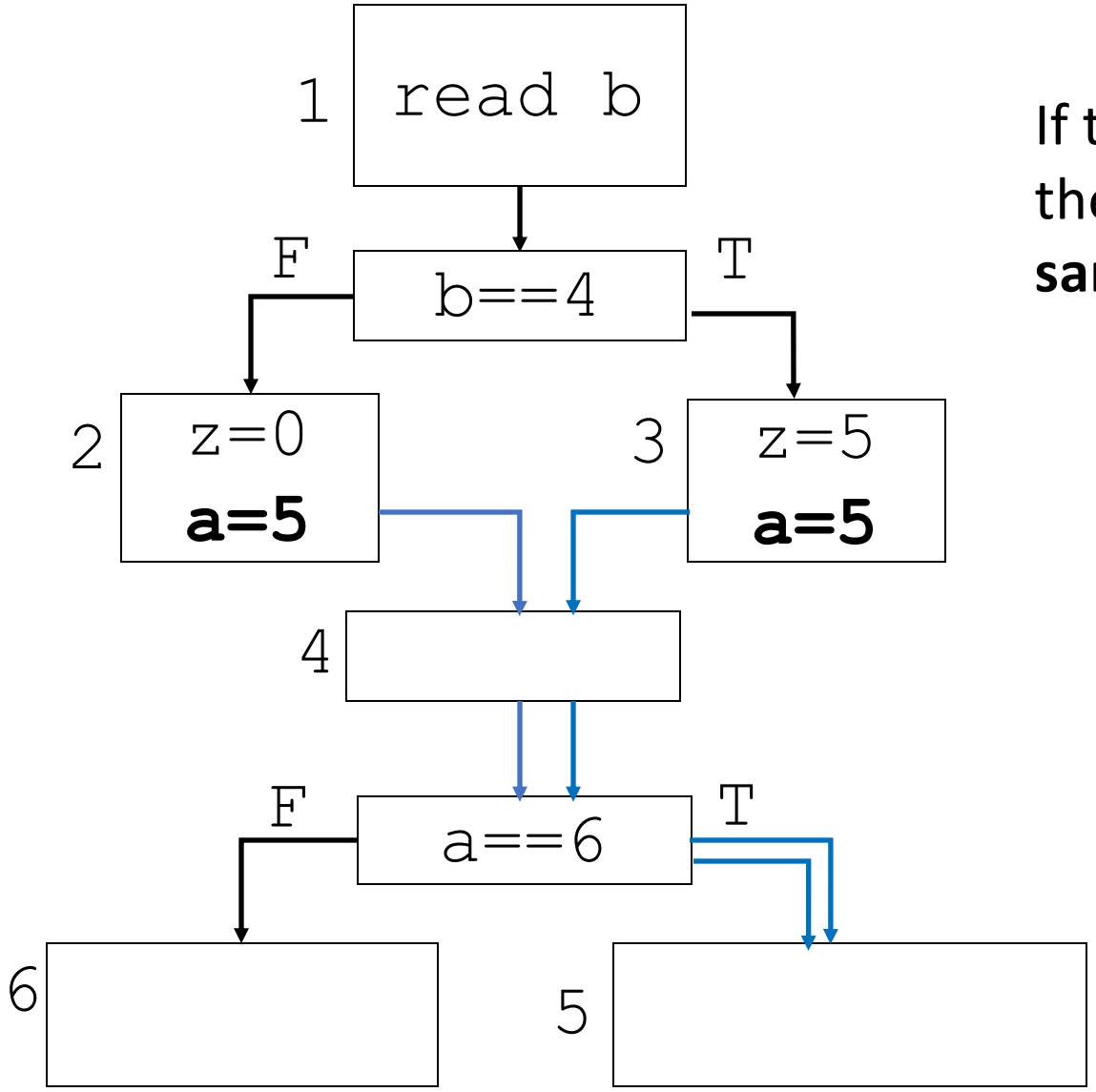


Observation:

If two balanced MIPS have the **same trigger edge** then the corresponding data flow values can be kept in **same bucket**.

Node	Value range of z from		
	Feasible Paths	Infeasible segment start	
		Blue	Red
4		z=[0,0]	z=[5,5]
5		z=[1,0]	z=[6,5]
6	z=[0,5]		

Use of MIPS Equivalence



Observation:

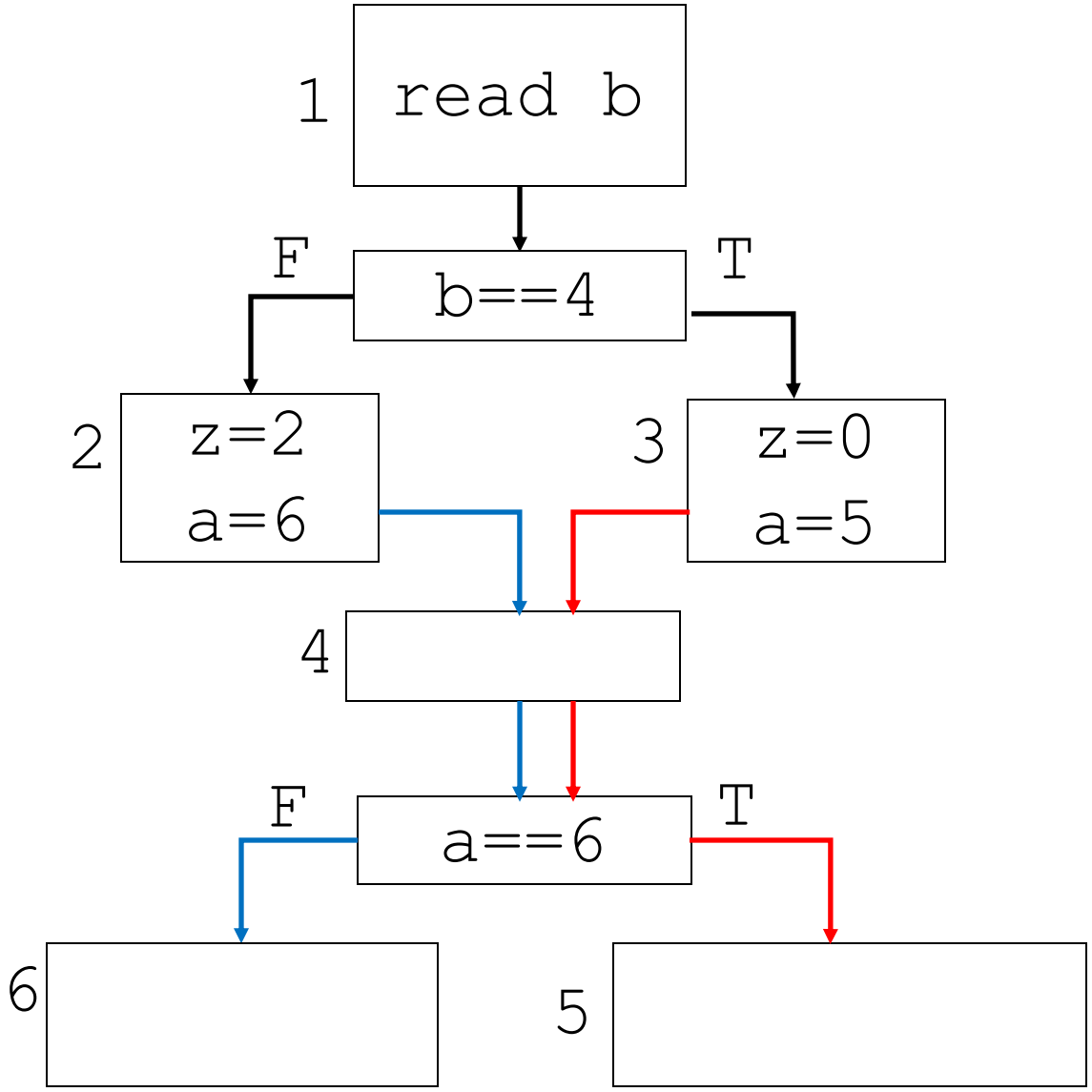
If two balanced MIPS have the **same trigger edge** then the corresponding data flow values can be kept in **same bucket**.

Node	Value range of z from	
	Feasible Paths	Infeasible segment start Blue
4		z=[0,5]
5		z=[0,5]
6	z=[0,5]	

2. Reducing the number of buckets

- Reduce Distinct MIPS using MIPS Equivalence
- **Reduce Number of Composite Buckets**
- Use Equality of Data Flow Values to Merge Buckets

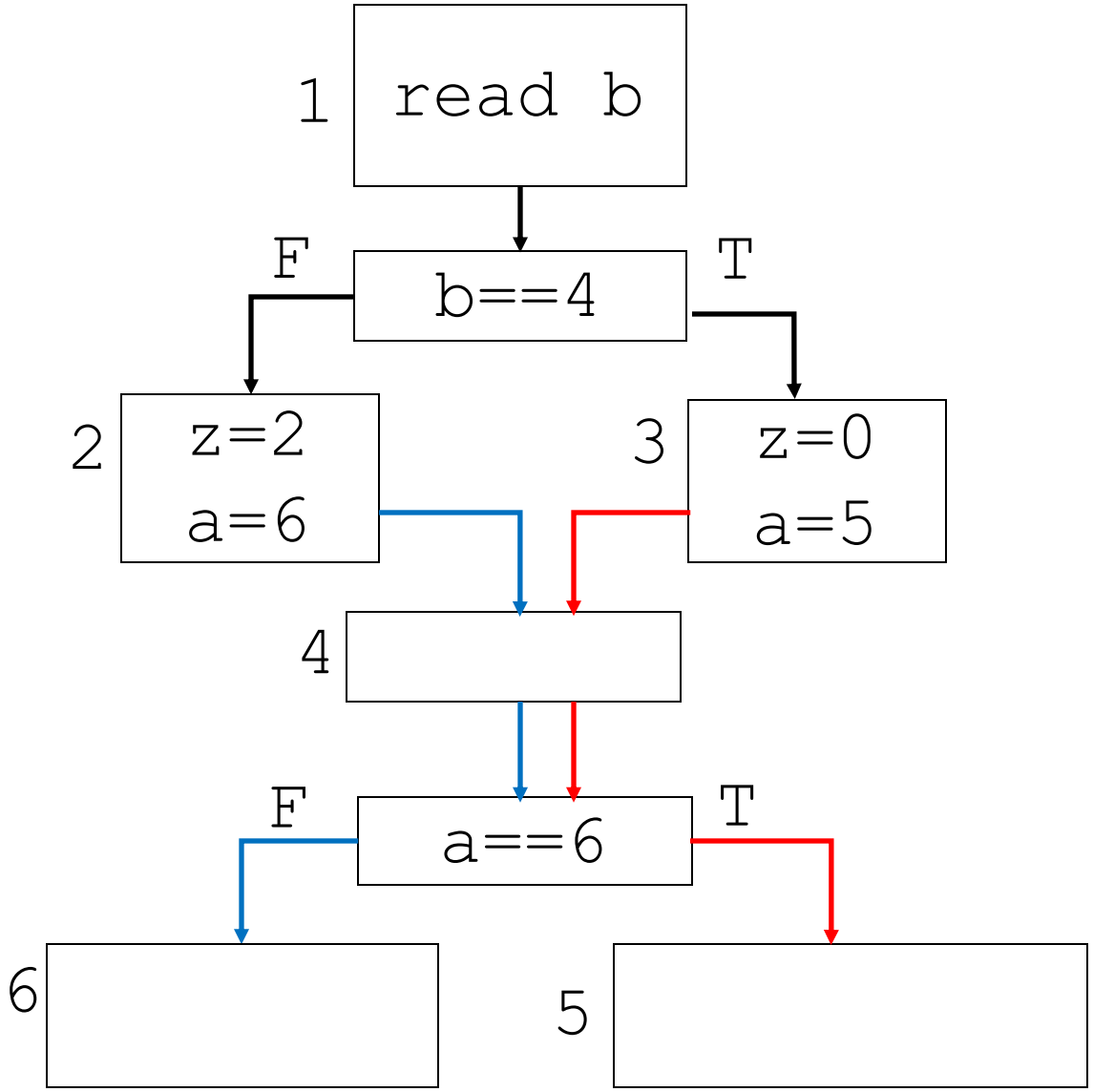
When not to create composite buckets



- When MIPS do not have contains-prefix-of relation

Node	Value range of z from		
	Feasible Paths	Infeasible segment start	
		Blue	Red
4		z=[2,2]	z=[0,0]
5		z=[2,2]	z=[0,0]
6		z=[2,2]	z=[0,0]

When not to create composite buckets



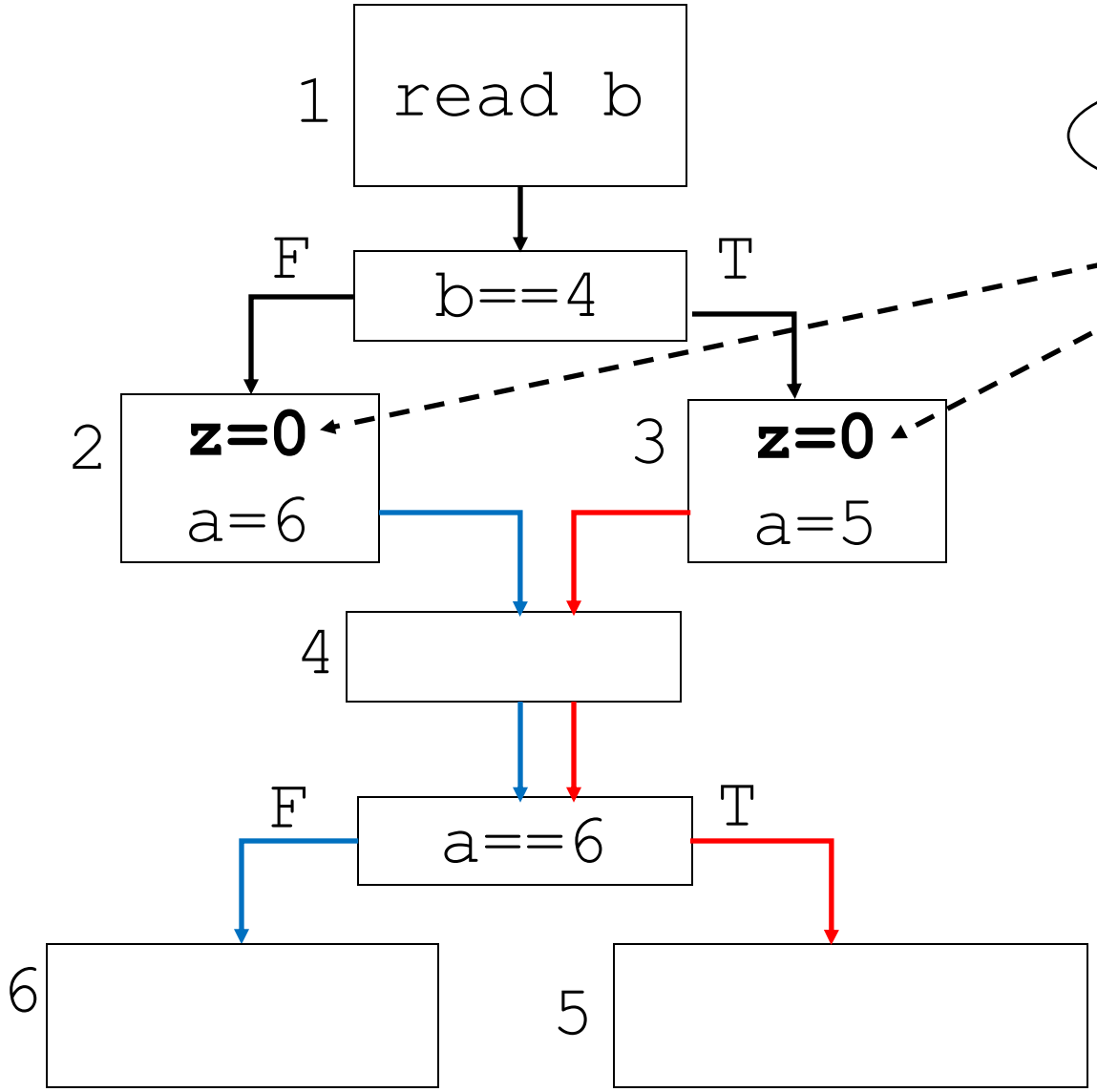
- When MIPS do not have contains-prefix-of relation

Node	Feasible Paths	Value range of z from Infeasible segment start	
		Blue	Red
4		z=[2,2]	z=[0,0]
5		z=[2,2]	z=[~~2~~,0]
6		z=[~~0~~,2]	z=[0,0]

2. Reducing the number of buckets

- Reduce Distinct MIPS using MIPS Equivalence
- Reduce Number of Composite Buckets
- **Use Equality of Data Flow Values to Merge Buckets**

Same data flow values in buckets

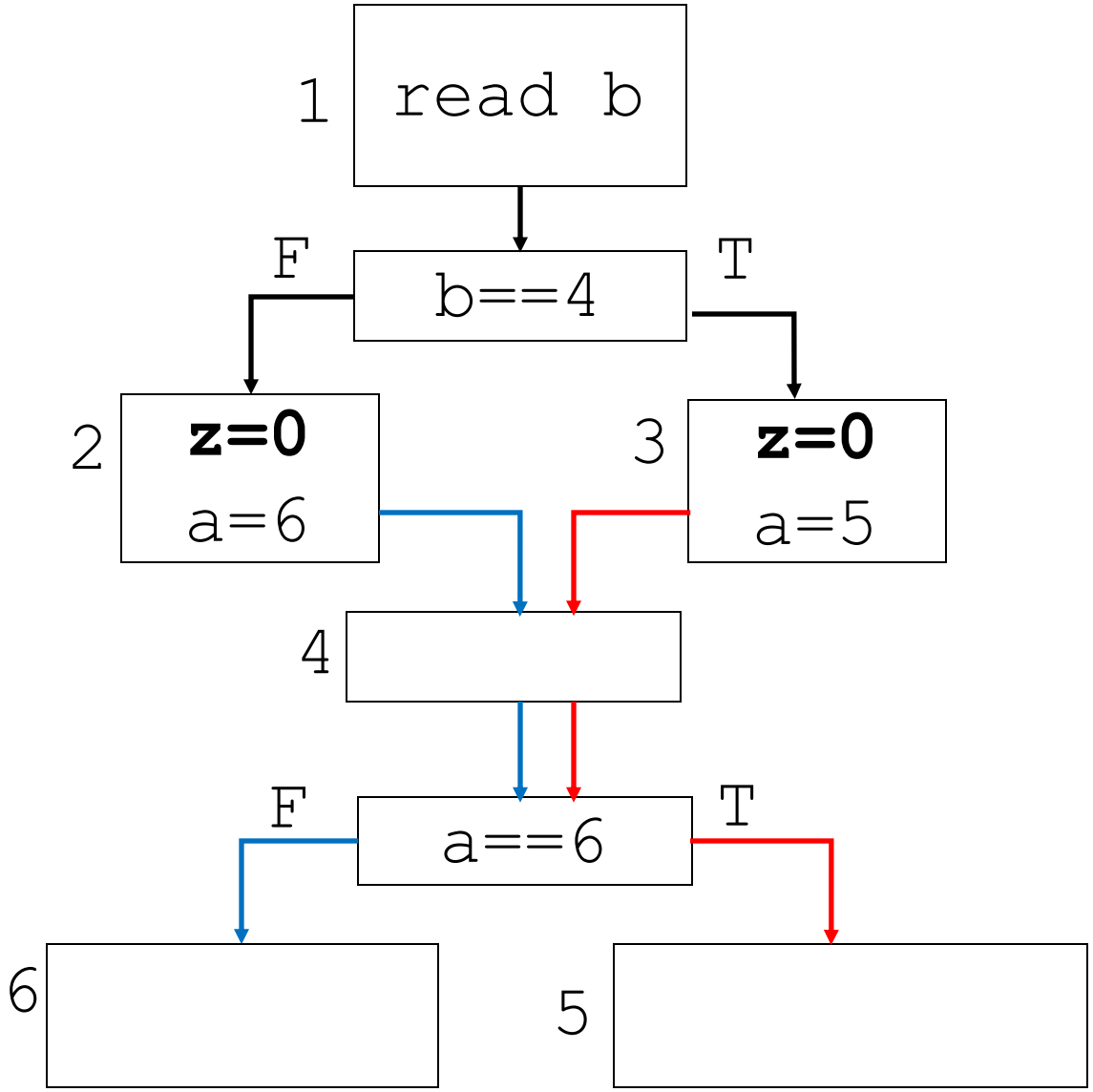


Same value

Node	Value range of z from Feasible Paths	Infeasible segment start Blue	Infeasible segment start Red
4		z=[0,0]	z=[0,0]
5		z=[0,0]	z=[,0]
6		z=[,0]	z=[0,0]

Reaches along feasible paths

Same data flow values in buckets



- Bucketing is not useful in this case

Node	Value range of z from		
	Feasible Paths	Infeasible segment start	
4	z=[0,0]		
5	z=[0,0]		
6	z=[0,0]		

Experiments

Implementation and Benchmarks

- **Reaching definitions analysis** and **potentially uninitialized variable analysis** in TCS ECA (Embedded Code Analyzer)
- 30 benchmarks that include Open Source(18), Industry(5), and SPEC CPU 2006(7) benchmarks

Results:

1. Improvement
   - depends on the presence of infeasible paths and analysis type

Analysis	Precision Improvement			Benchmarks with Non-zero improvement
	Maximum	Average	Geometric mean	
Un-initialized Variables	100%	18.5%	3	14/27
Def-Use Pairs	13%	2.8%	1.75	28/30

2. Analysis Time: Average 3x more than MFP.

Benchmarks		Uninitialized Variable Alarms		
		MFP	FPMFP	reduction(%)
Open Source	1.acpid	1	1	0(0)
	2.polymorph	4	4	0(0)
	3.nlkain	4	0	4(100)
	4.spell	3	3	0(0)
	5.ncompress	7	7	0(0)
	6.gzip	0	0	-
	7.stripcc	27	1	26(96.30)
	8.barcode-nc	0	0	-
	9.barcode	2	0	2(100)
	10.archimedes	61	56	5(8.20)
	11.combine	63	63	0(0)
	12.httpd	117	117	0(0)
	13.sphinxbase	46	43	3(6.52)
	14.chess	16	16	0(0)
	15.antiword	18	18	0(0)
	16.sendmail	103	102	1(0.97)
	17.sudo	62	58	4(6.45)
	18.ffmpeg	124	112	12(9.68)

Uninitialized Alarms:

- Improvement is observed when all paths along which variable is undefined are infeasible.
- 100% reduction is achievable in some cases
- Reasons for less improvement
  - initialization in Library calls
  - function call in path condition

Benchmarks		Reaching Definition Analysis		
Type	Name	#def-use pairs		reduction(%)
		MFP	FPMFP	
Open Source	1.acpid	156	156	0(0.00)
	2.polymorph	228	228	0(0.00)
	3.nlkain	1042	965	77(7.39)
	4.spell	516	515	1(0.19)
	5.ncompress	1201	1175	26(2.16)
	6.gzip	3423	3401	22(0.64)
	7.stripcc	2703	2645	58(2.15)
	8.barcode-nc	3051	3007	44(1.44)
	9.barcode	3709	3653	56(1.51)
	10.archimedes	44337	44216	121(0.27)
	11.combine	16618	15859	759(4.57)
	12.httpd	10475	10072	403(3.85)
	13.sphinxbase	9641	8482	1159(12.02)
	14.chess	31386	31303	83(0.26)
	15.antiword	68889	60144	8745(12.69)
	16.sendmail	102812	101470	1342(1.31)
	17.sudo	14391	14211	180(1.25)
	18.ffmpeg	89148	86607	2541(2.85)

Def-Use Query:

- Improvement is observed only when all paths between definition and use are infeasible
- 100% reduction may not be achievable by any technique

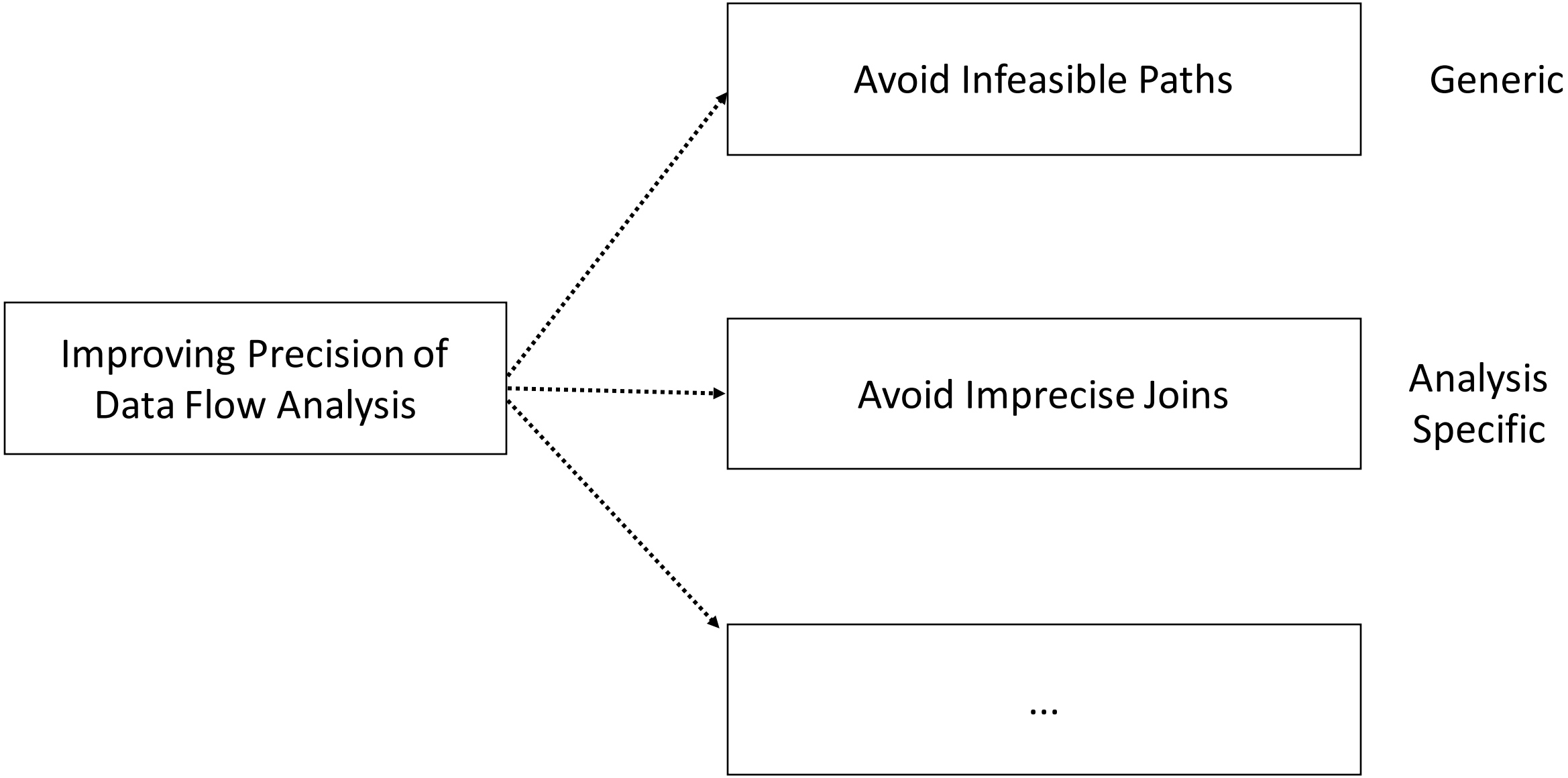
	Benchmarks	Analysis Time (Sec)				Memory Consumption (MB)			
		Prep	MFP	FPMFP	Increase(x)	Prep	MFP	FPMFP	Increase(x)
Open Source	1.acpid	1.1	0.8	0.7	-0.1	[illegible]	0	0	0
	2.polymorph 0.96	1.1	0.8	0.7	-0.1	1	0	0	0
	3.nlkain	1.2	0.9	1.2	0.3	1	1	1	0
	4.spell	1.1	0.8	1	0.2	1	1	0	0
	5.ncompress	1	0	1	1(1)	2	1	4	2(2)
	6.gzip	1	1	5	4(4)	5	10	24	14(1.4)
	7.stripcc	3	1	6	5(5)	8	5	21	15(3)
	8.barcode-nc	2	1	2	1(1)	6	8	6	-1(-0.1)
	9.barcode	2	2	2	0	7	10	8	-1(-0.1)
	10.archimedes	6	3	32	29(10)	24	44	156	112(2.5)
	11.combine	5	3	5	2(0.6)	16	23	19	-3(-0.1)
	12.httpd	40	14	19	5(0.3)	28	39	21	-18(-0.46)
	13.sphinxbase	7	3	3	0	13	14	7	-9(-0.56)
	14.chess	13	7	30	23(3)	31	93	191	98(1)
	15.antiword	13	16	82	66(4)	47	101	218	117(1)
	16.sendmail	110	142	2060	1918(13)	81	340	1548	1208(3.5)
	17.sudo	15	9	17	8(1)	35	71	19	-52(-0.7)
	18.ffmpeg	234	51	80	29 (0.5)	158	266	90	-175(-0.6)

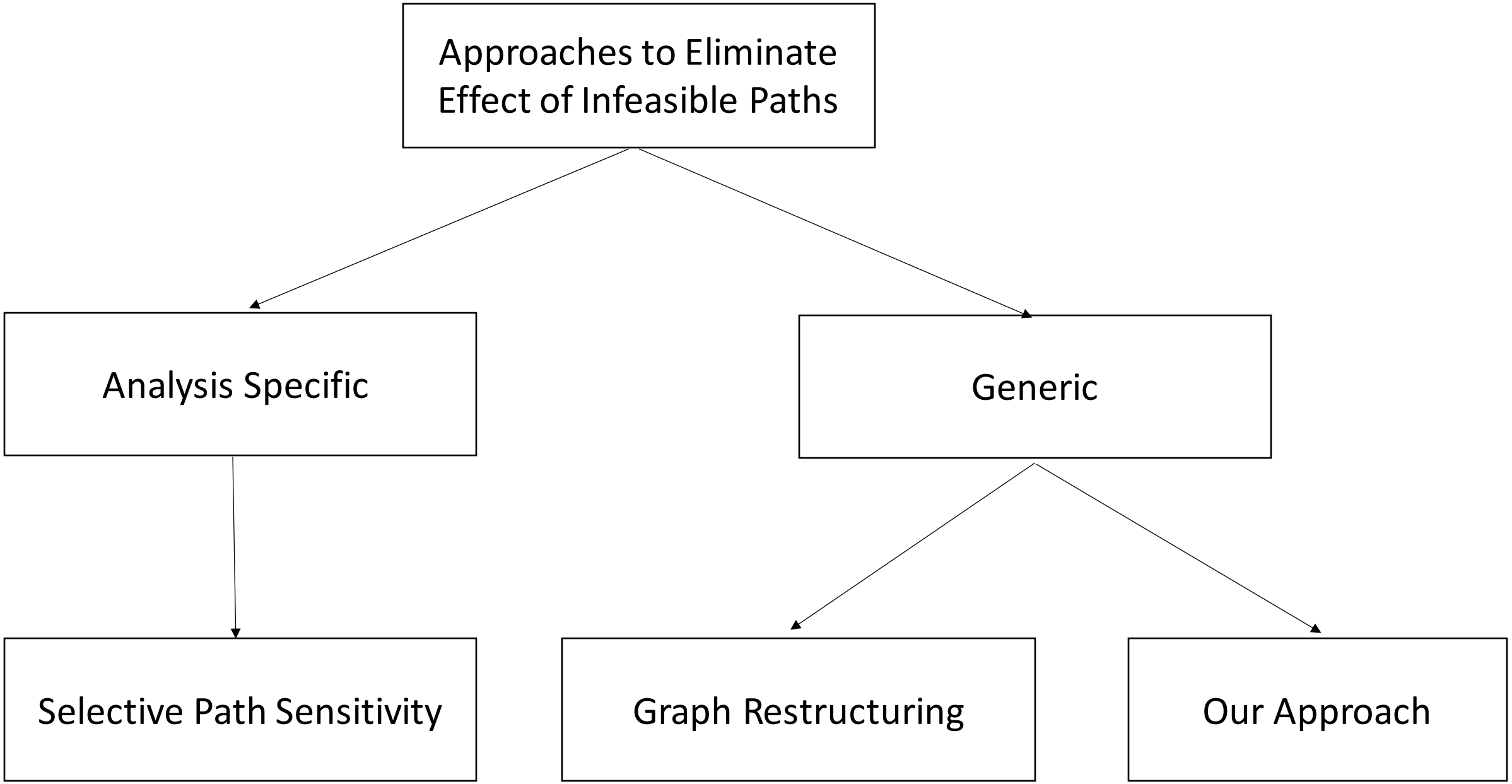
Prep. Represents Infeasible path detection cost

High analysis time because number of MIPS is large (i.e, 669)

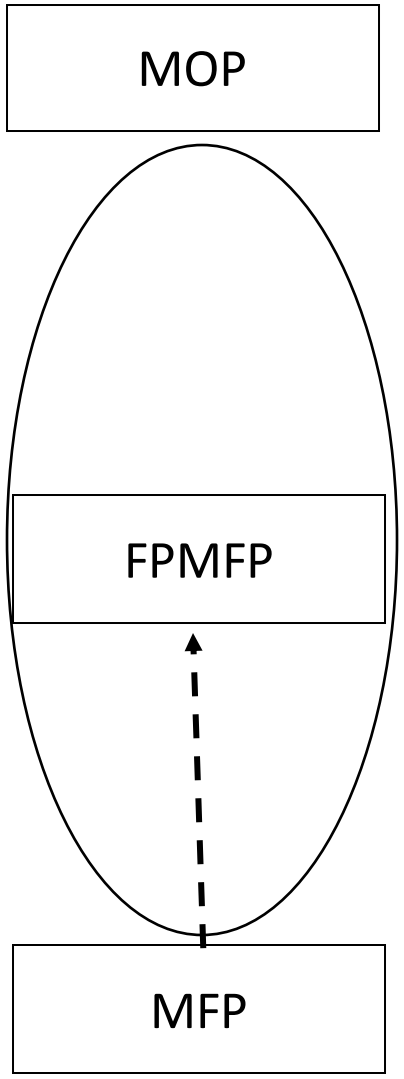
On 50% benchmarks memory consumption was reduced

Related Work


Improving Precision of Data Flow Analysis
Avoid Infeasible Paths
Avoid Imprecise Joins
...
Generic
Analysis Specific


Approaches to Eliminate Effect of Infeasible Paths
Analysis Specific
Generic
Selective Path Sensitivity
Graph Restructuring
Our Approach

Main Contributions



1. A new generic solution for data flow analysis (FPMFP) (more precise than MFP and more scalable than MOP)

2. Inter-procedural Extension of FPMFP

3. Novel Optimizations to make FPMFP practically effective

4. Implemented the solution in industry strength static analysis tool. Performed experiments on 30 benchmarks

Publications based on this work

1. "Computing partially path-sensitive MFP solutions in data flow analyses"
   Komal Pathade, Uday Khedker
   Published in International Conference on Compiler Construction (CC) 2018.

2. "Path-sensitive MFP solutions in presence of intersecting infeasible path segments in data flow analyses"
   Komal Pathade, Uday Khedker
   Published in International Conference on Compiler Construction (CC) 2019.

Directions for Future Work

1. Inter-procedural infeasible paths that start and end in different procedures

2. A pre-analysis to estimate the impact of FPMFP.

Thank You

Questions?

References

1. Rastislav Bodik, Rajiv Gupta, and Mary Lou Soa. 1997. Refining data flow information using infeasible paths. In Software Engineering ESEC/FSE'97. Springer, 361–377.

2. Paulo Marcos Siqueira Bueno and Mario Jino. 2000. Identication of potentially infeasible program paths by monitoring the search for test data. In Automated Software Engineering, 2000. Proceedings ASE 2000. The Fifteenth IEEE International Conference on. IEEE, 209–218.

References

3. Rastislav Bodik, Rajiv Gupta, and Mary Lou Soa. 1997. Interprocedural conditional branch elimination. In ACM SIGPLAN Notices, Vol. 32. ACM, 146–158.

4. Aditya Thakur and R Govindarajan. 2008. Comprehensive path sensitive dataflow analysis. In Proceedings of the 6th annual IEEE/ACM international symposium on Code generation and optimization. ACM, 55–63.

References

5. Nurit Dor, Stephen Adams, Manuvir Das, and Zhe Yang. 2004. Software validation via scalable path-sensitive value flow analysis. In ACM SIGSOFT Software Engineering Notes, Vol. 29. ACM, 12–22.

6. Yichen Xie, Andy Chou, and Dawson Engler. 2003. Archer: using symbolic, path-sensitive analysis to detect memory access errors. ACM SIGSOFT Software Engineering Notes 28, 5 (2003), 327–336.

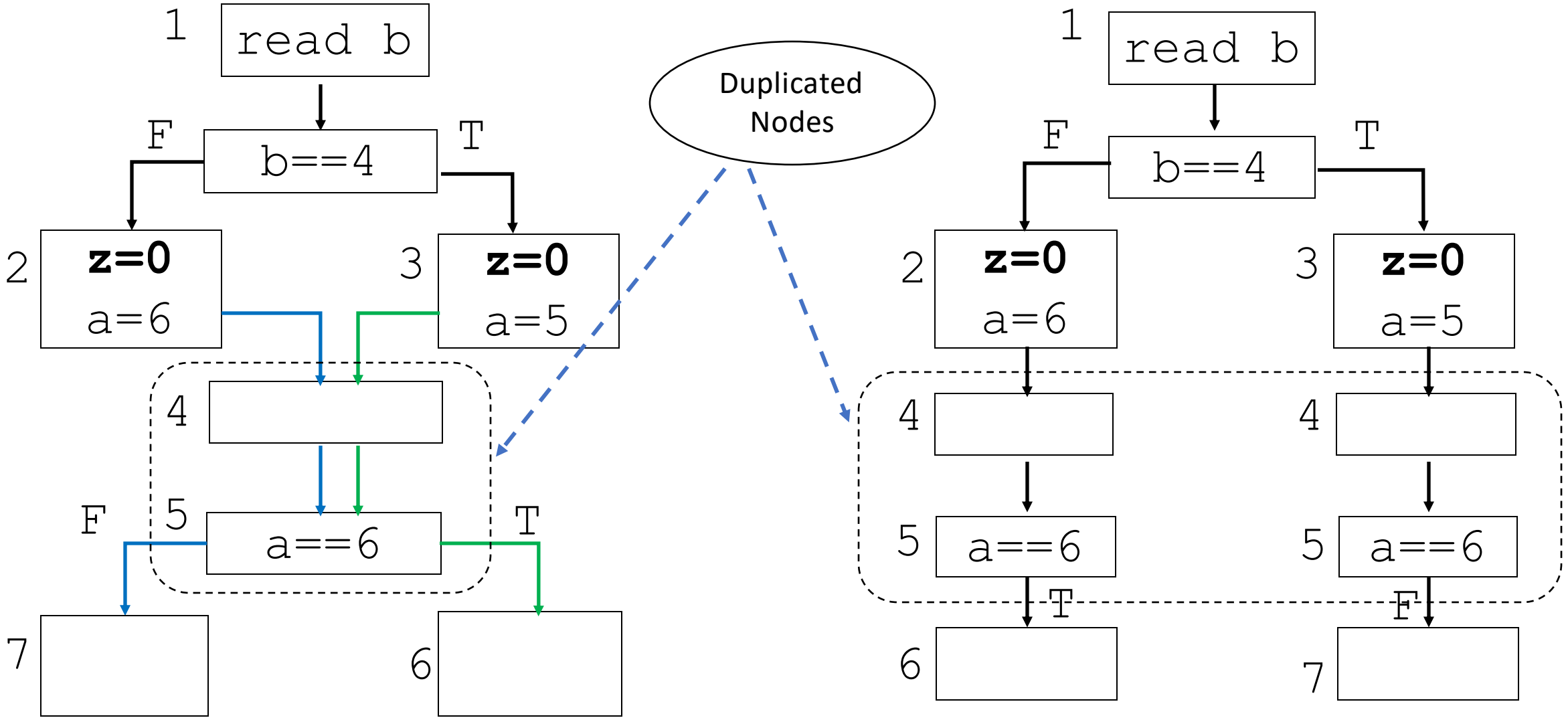
References

7. Komal Pathade, Uday Khedker,
"Computing partially path-sensitive MFP solutions in data flow analyses"
Published in International Conference on Compiler Construction (CC) 2018.

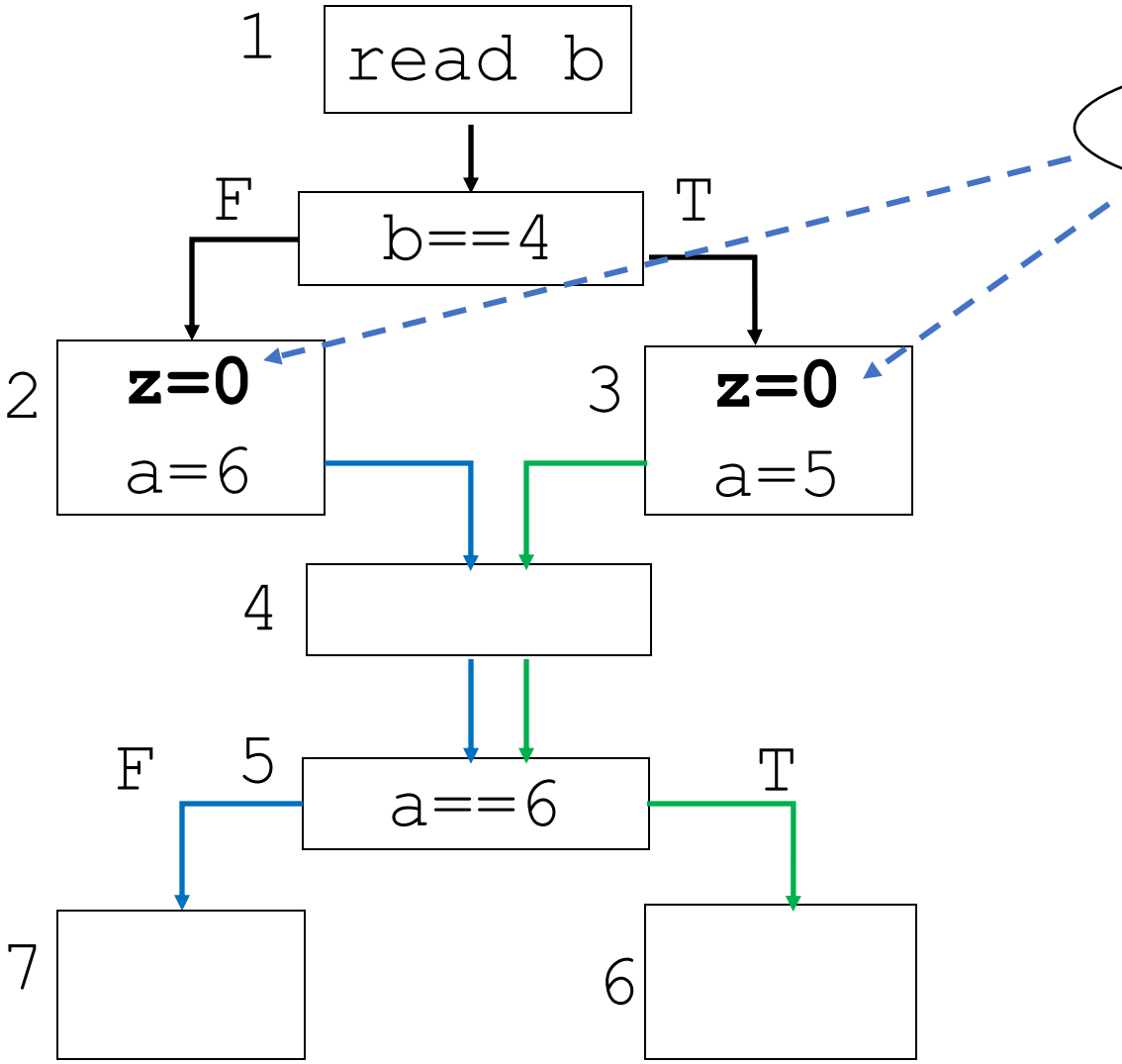
8. Komal Pathade, Uday Khedker
"Path-sensitive MFP solutions in presence of intersecting infeasible path segments in data flow analyses"
Published in International Conference on Compiler Construction (CC) 2019.

Additional Slides

CFG Restructuring



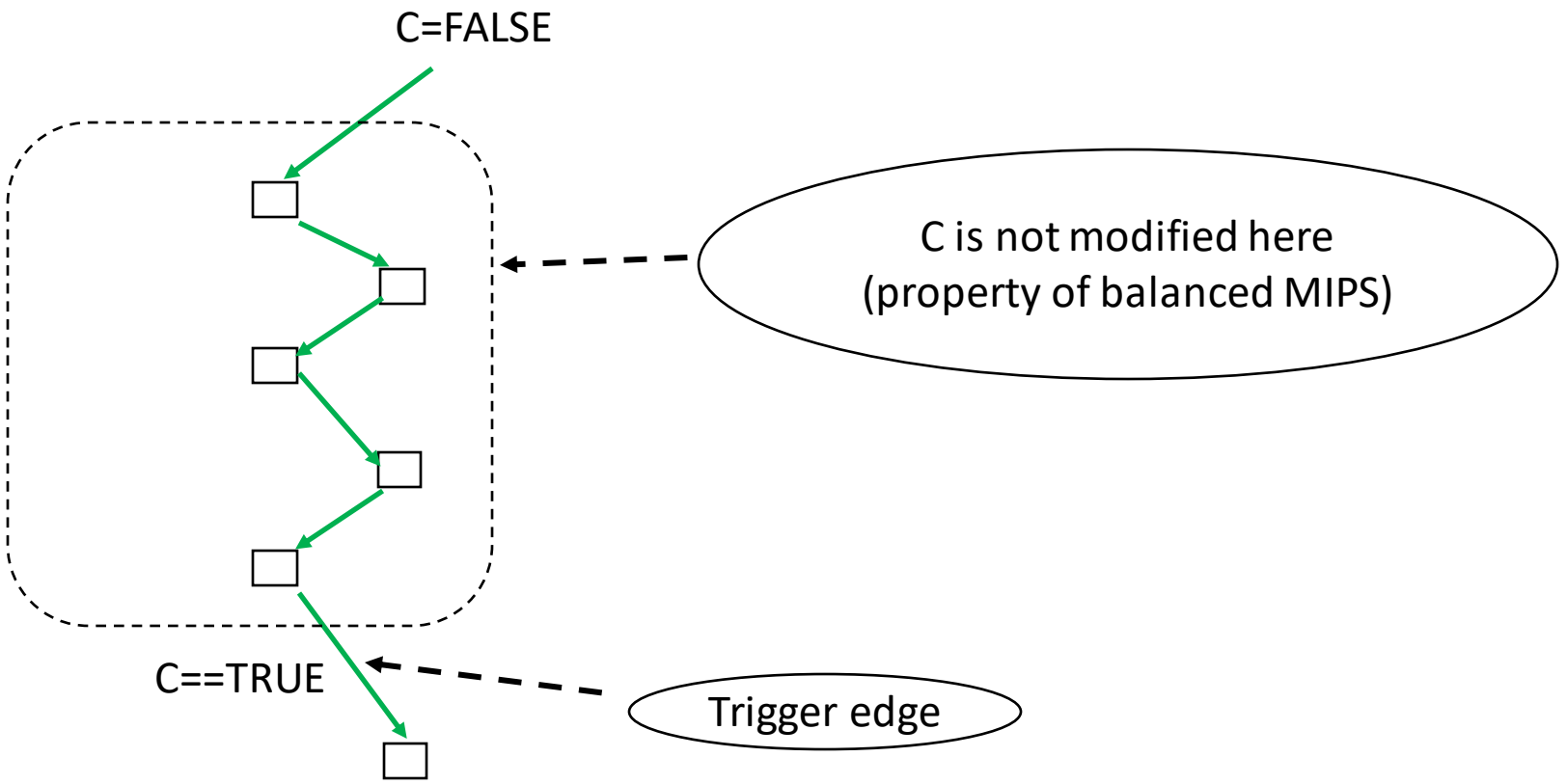
Feasible Path MFP



Same value so bucketing not needed

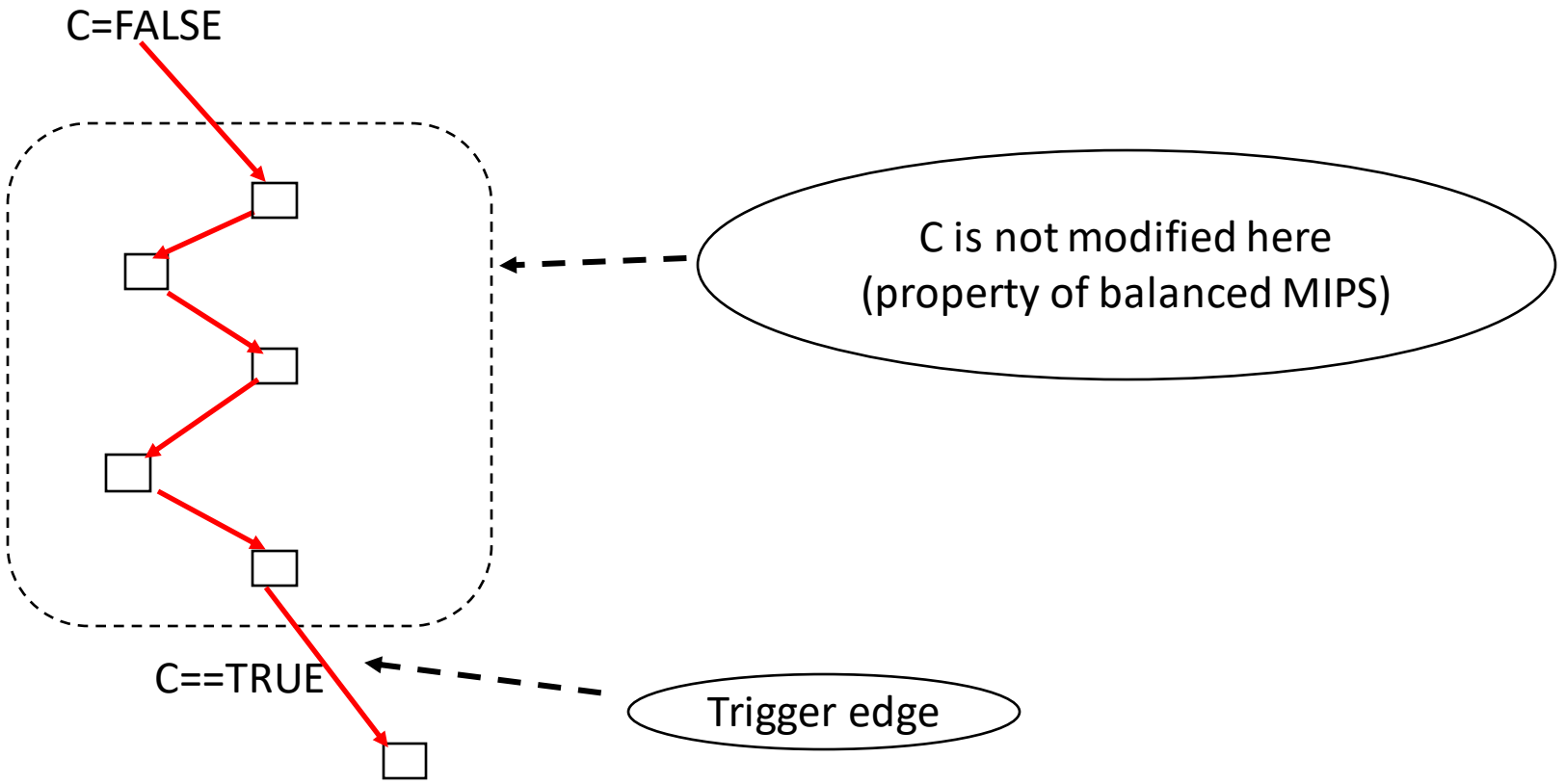
Node	Value range of z from		
	Feasible Paths	Infeasible segment start	
		Blue	Green
4	z=[0,0]		
5	z=[0,0]		
6	z=[0,0]		

Equivalence of Two Balanced MIPS



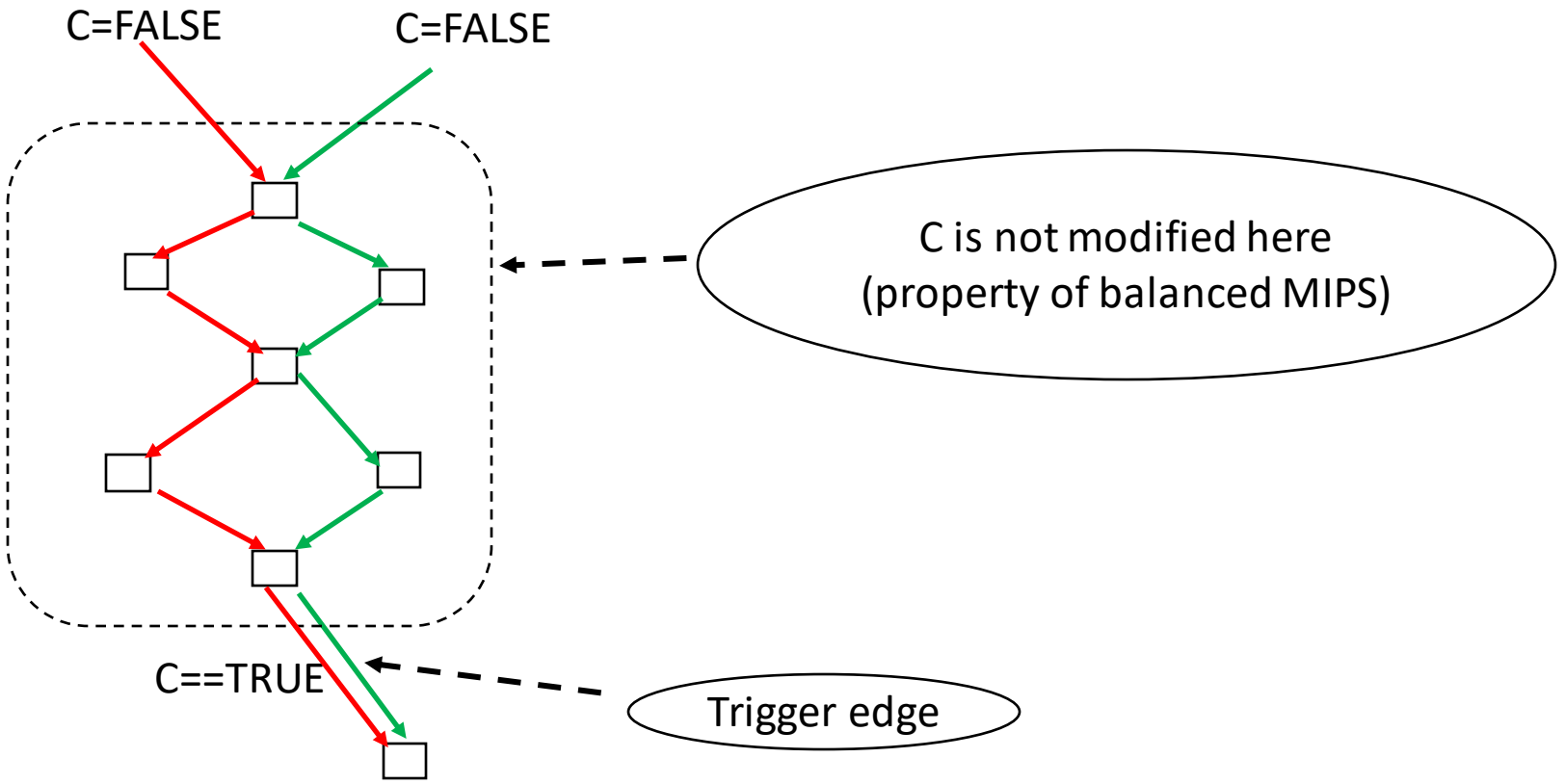
A balanced MIPS

Equivalence of Two Balanced MIPS



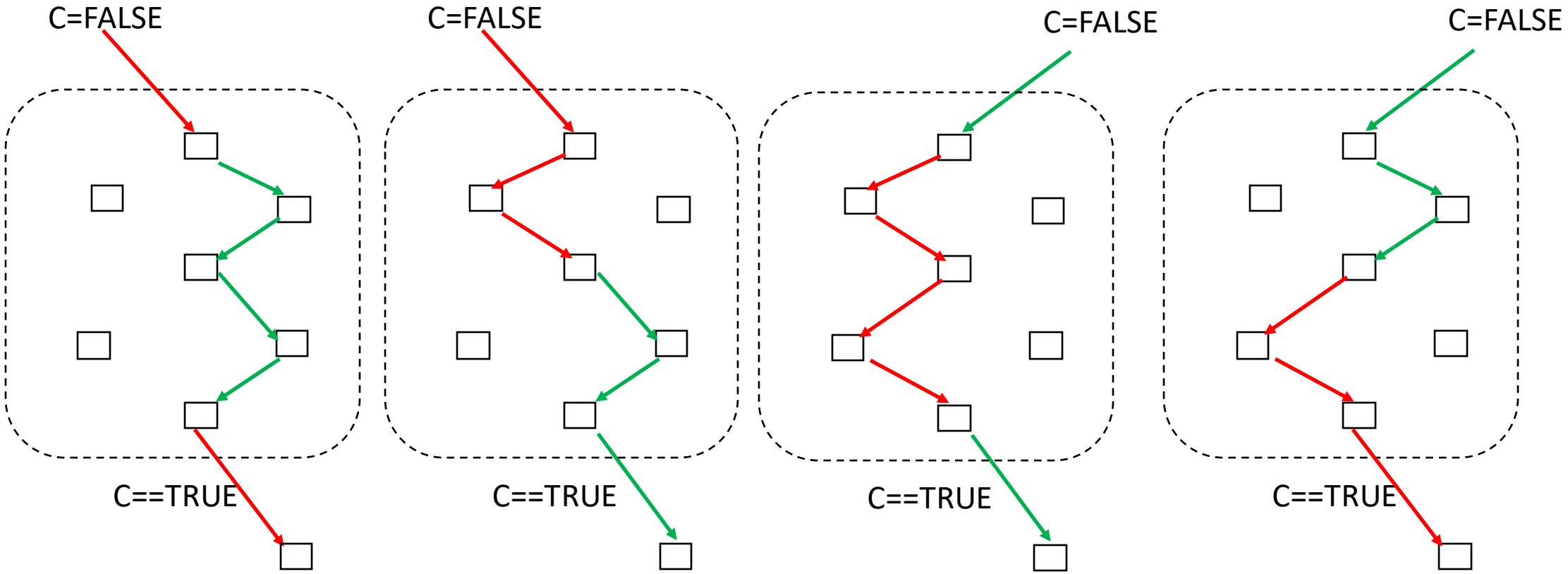
Second balanced MIPS

Equivalence of Two Balanced MIPS



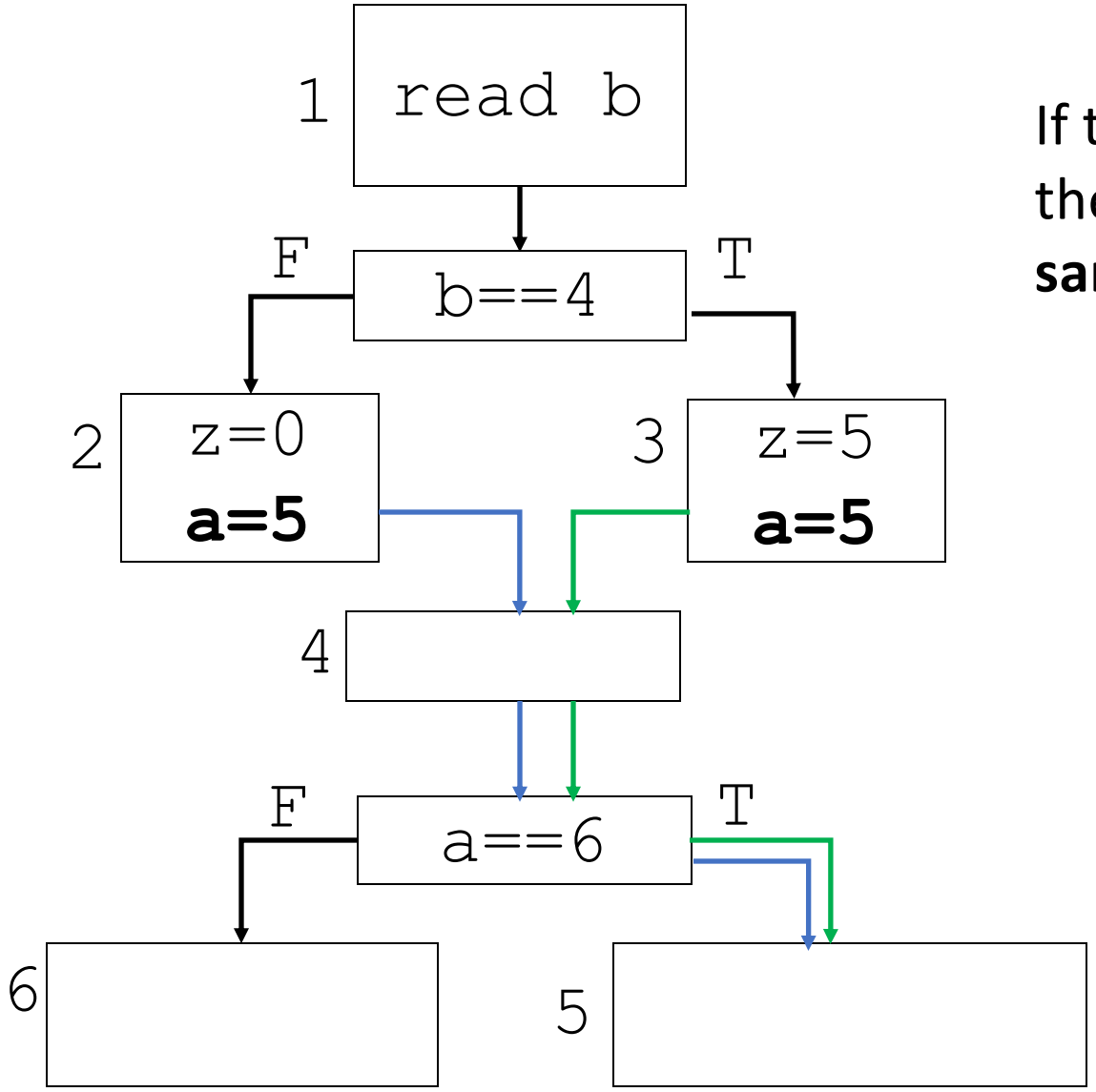
Interleaving of two balanced MIPS

Paths in the Interleaving are infeasible



MIPS Equivalence Theorem:
All paths in the interleaving of two balanced MIPS are infeasible, if the MIPS have the same trigger edge.

Use of MIPS Equivalence

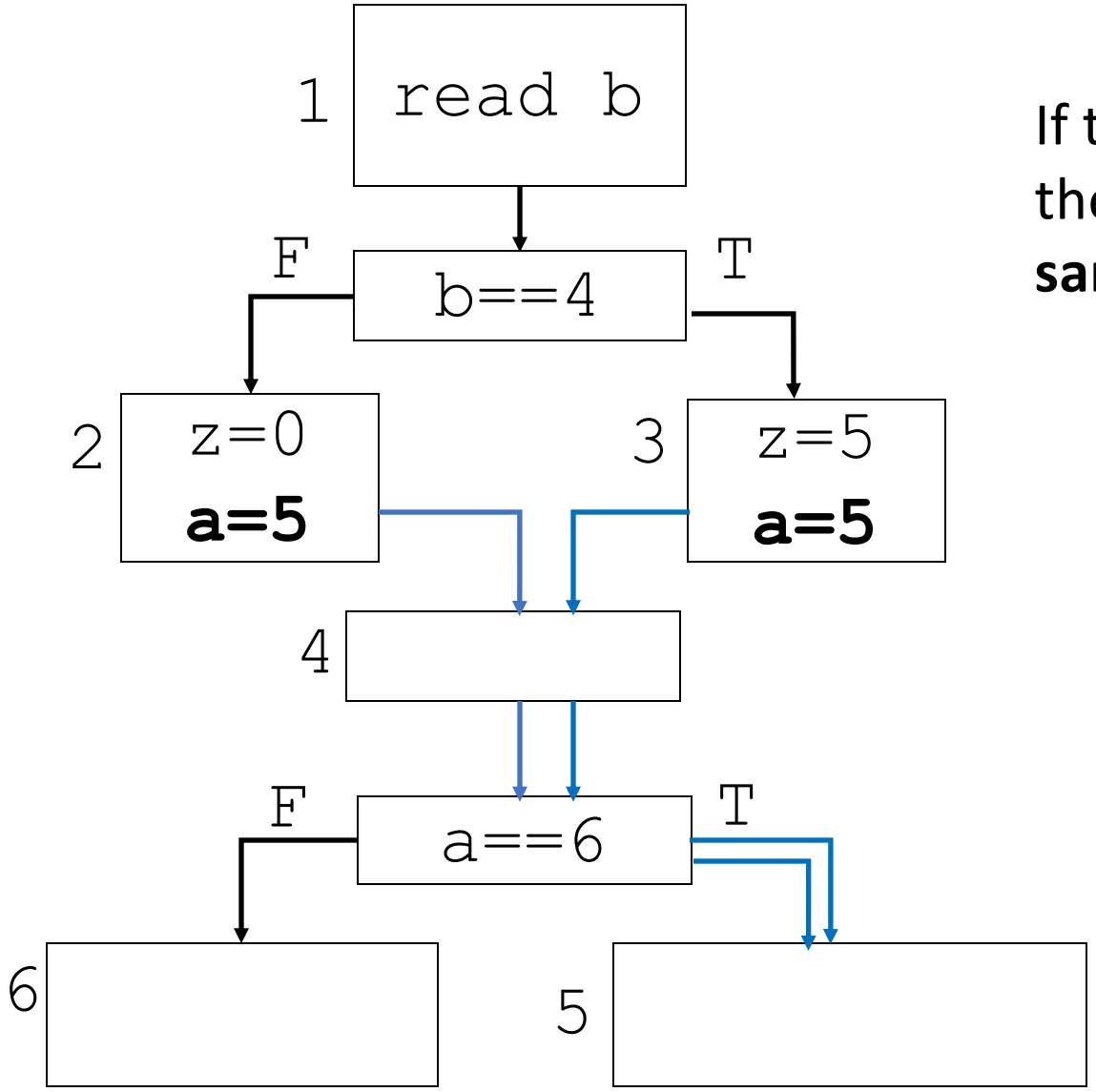


Observation:

If two balanced MIPS have the same trigger edge then the corresponding data flow values can be kept in **same bucket**.

Node	Value range of z from		
	Feasible Paths	Infeasible segment start	
		Blue	Green
4		z=[0,0]	z=[5,5]
5		z=[,0]	z=[,5]
6	z=[0,5]		

Use of MIPS Equivalence

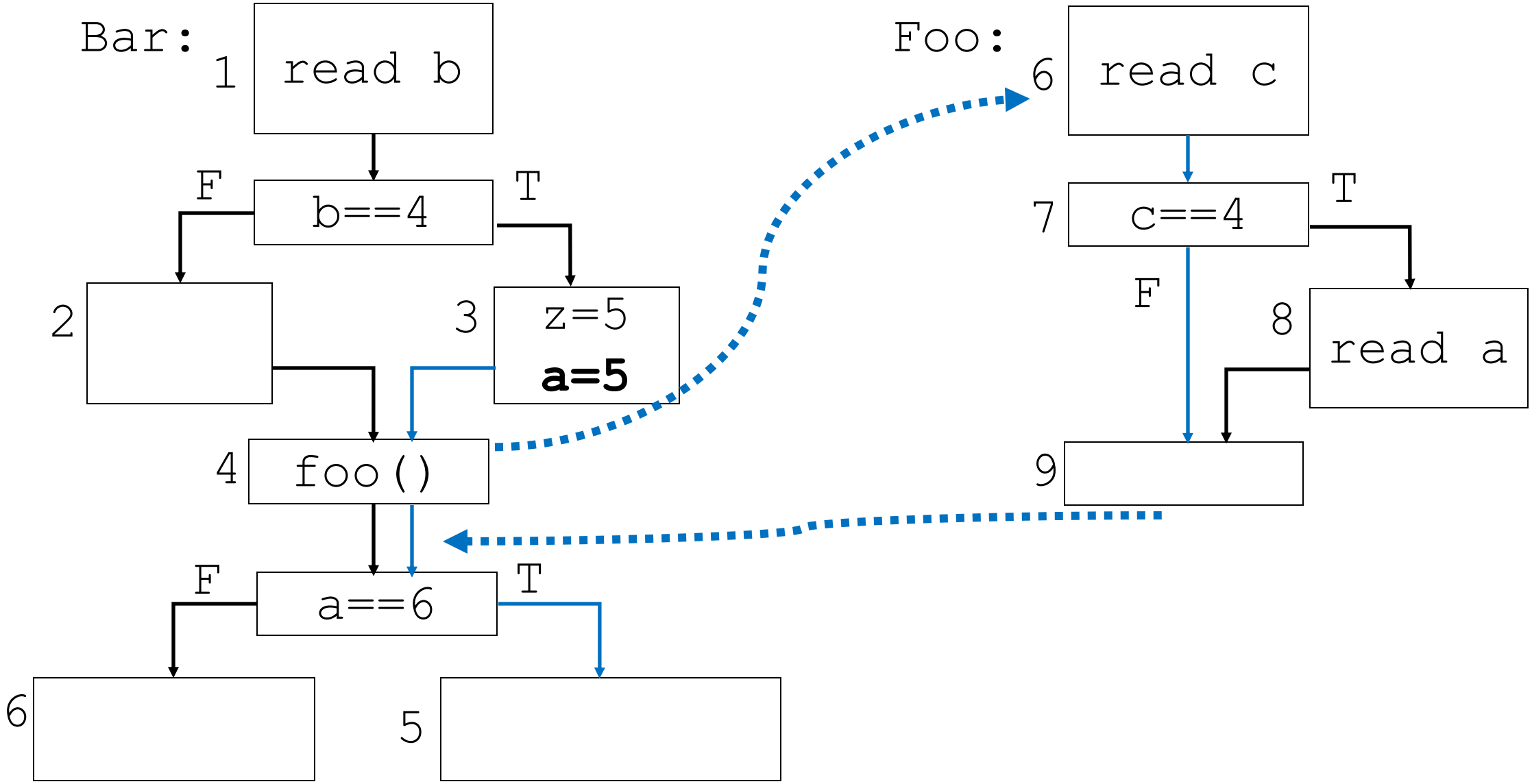


Observation:

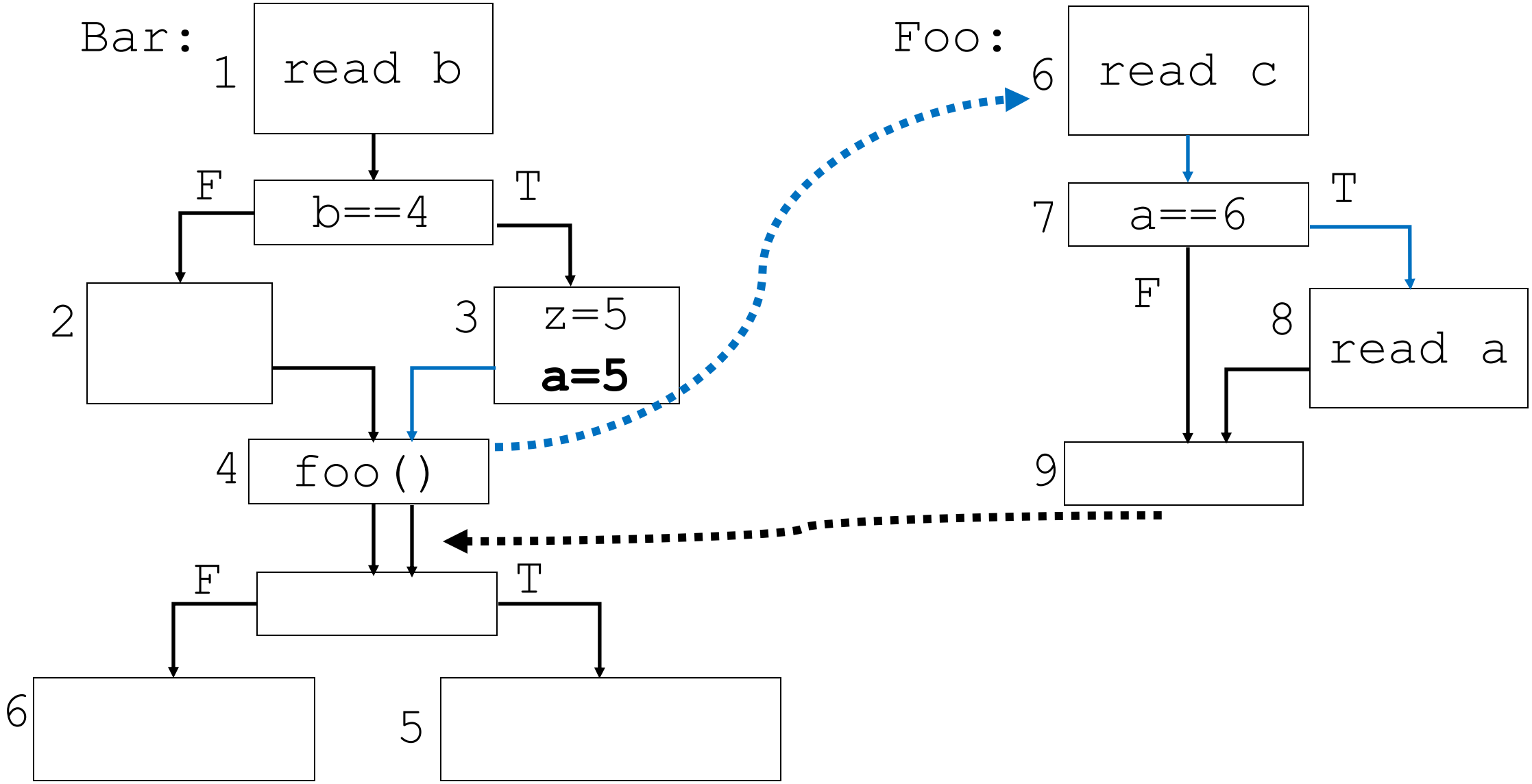
If two balanced MIPS have the same trigger edge then the corresponding data flow values can be kept in **same bucket**.

Node	Value range of z from	
	Feasible Paths	Infeasible segment start Blue
4		z=[0,5]
5		z=[0,5]
6	z=[0,5]	

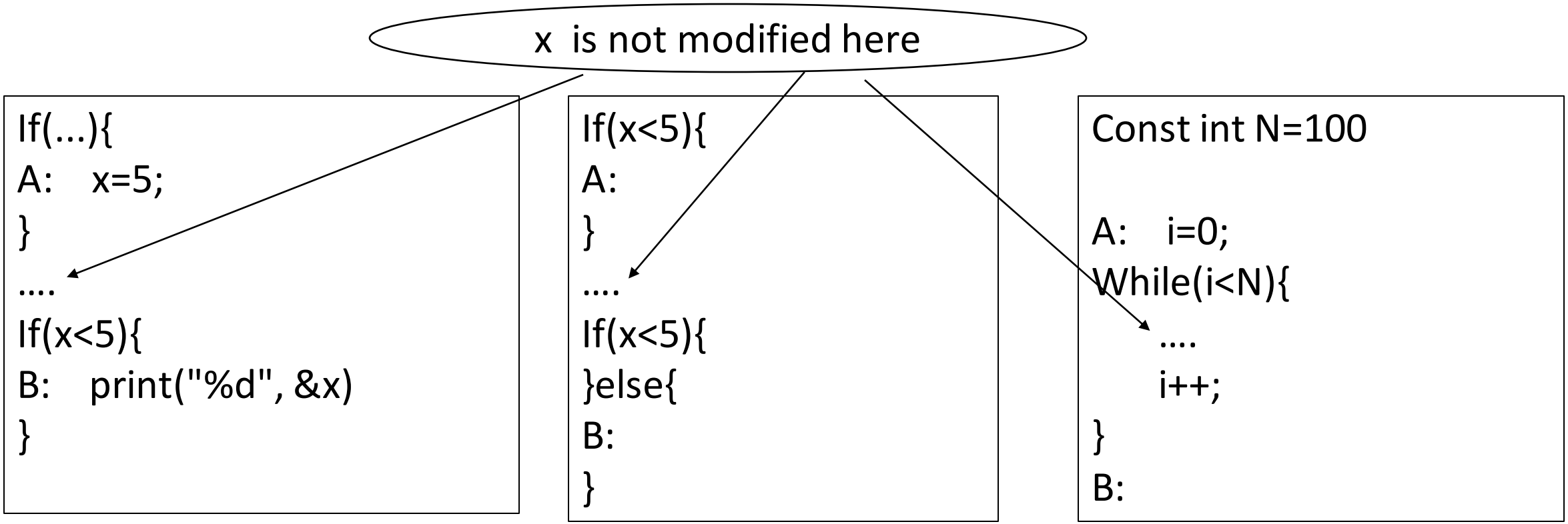
Non balanced MIPS: start and end in same proc



MIPS start and end in different procedure



Types of MIPS Observed



Conflict between assignment and branch condition

Conflict between two branch conditions

Loop with at least one iteration

Path segments connecting A and B are MIPS

	Benchmarks	Number of Distinct MIPS Before and After Optimization 1		
		Before	After	Reduction(%)
Open Source	1.acpid	83	6	77(92.77)
	2.polymorph 0.96	65	24	41(63.08)
	3.nlkain	35	33	2(5.71)
	4.spell	99	14	85(85.86)
	5.ncompress	14163	30	14133(99.79)
	6.gzip	23.5M	49	23.5M(99.99)
	7.stripcc	0.7M	107	0.7M(99.99)
	8.barcode-nc	208	60	148(71.15)
	9.barcode	3851	76	3775(98.03)
	10.archimedes	1M	118	1M(99.99)
	11.combine	27061	234	26827(99.14)
	12.httpd	76.5B	226	76.5B(99.99)
	13.sphinxbase	6463	213	6250(96.70)
	14.chess	19.7B	262	19.7B(99.99)
	15.antiword	1446.7T	669	1446.7T(99.99)
	16.sendmail	94T	726	94T(99.99)
	17.sudo	549565	250	549315(99.95)
	18.ffmpeg	5T	1363	5T(99.99)
SPEC 2006	19.mcf	44	19	25(56.82)
	20.bzip2	115M	501	115M(99.99)
	21.hmmer	8.6M	595	8.6M(99.99)
	22.sjeng	1508T	438	1508T(99.99)
	23.milc	2322	497	1825(78.60)
	24.h264ref	51B	2189	51B(99.99)
	25.gobmk	$> 2^{64}$	1834	$> 2^{64}$(99.99)

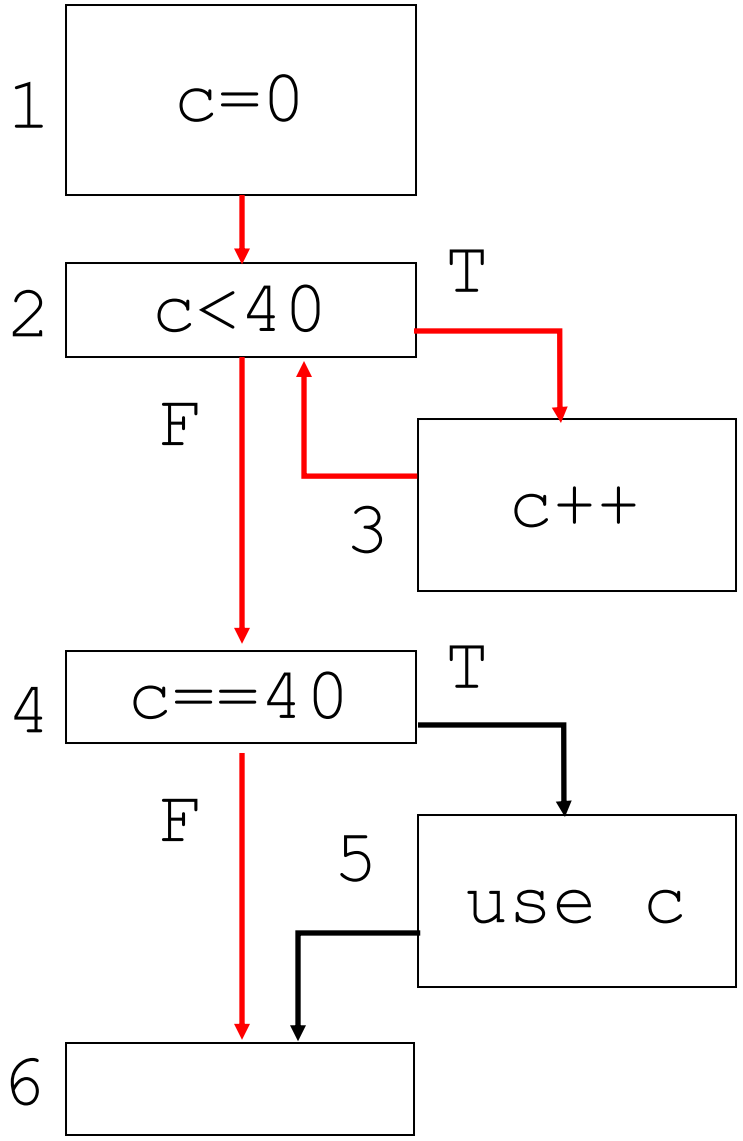
Reduction in number of distinct MIPS

- Upto: 99%,
- Average: 85%
- Geometric mean: 78%

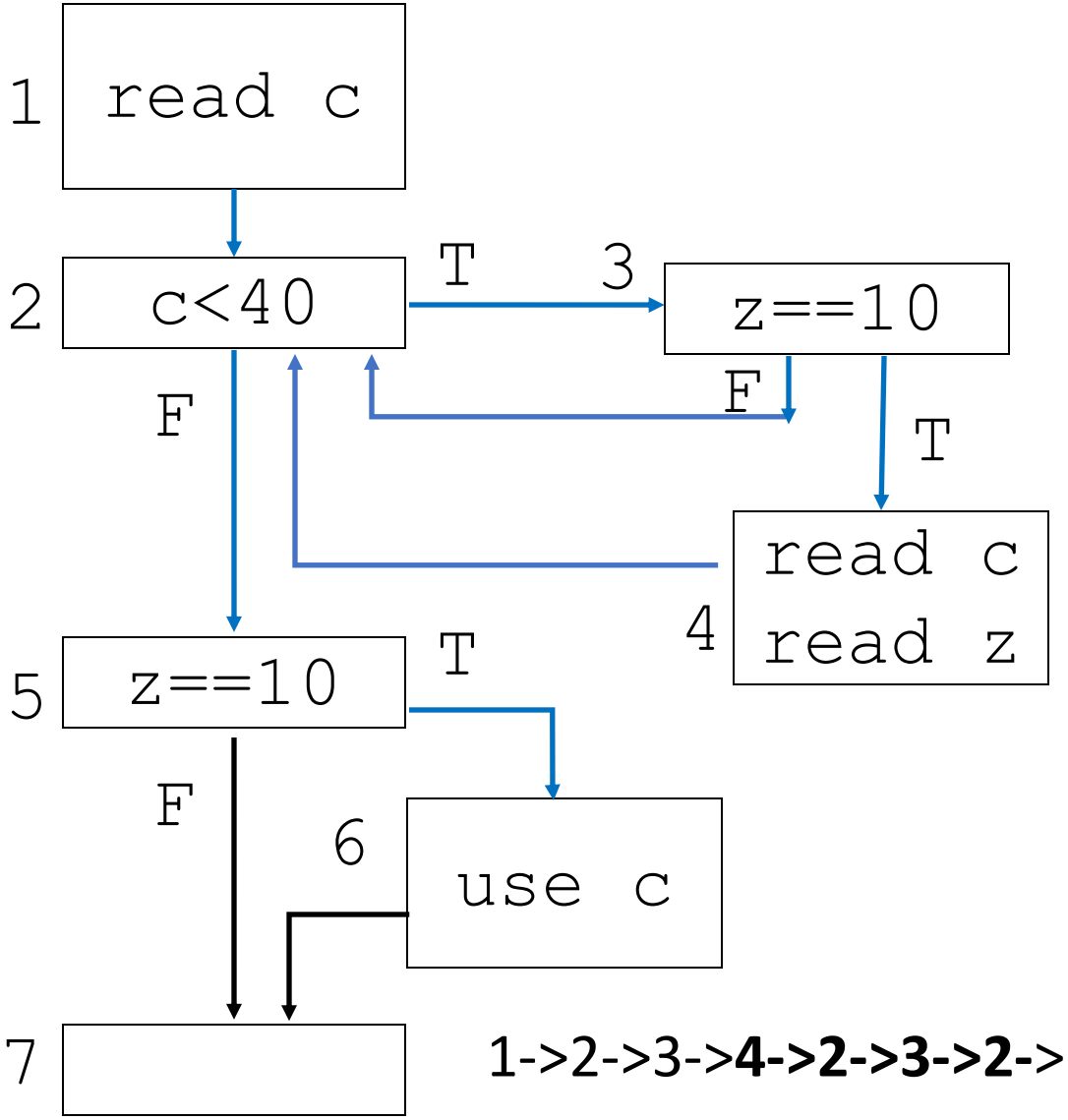
	Benchmarks	Analysis Time (in Seconds) Before and After Optimization 2		
		Before	After	Reduction(%)
Open Source	1.acpid	0.6	0.7	-0.1(-16)
	2.polymorph 0.96	0.5	0.7	-0.2(-40)
	3.nlkain	0.6	1.2	-0.6(-100)
	4.spell	0.7	1	-0.3(-42)
	5.ncompress	3	1	2(66)
	6.gzip	22	5	17(77)
	7.stripcc	55	6	49(89)
	8.barcode-nc	2	2	0
	9.barcode	2	2	0
	10.archimedes	53	32	21(39)
	11.combine	19	5	14(73)
	12.httpd	290	19	271 (93)
	13.sphinxbase	4	3	1(25)
	14.chess	76	30	46(60)
	15.antiword	705	82	623(88)
	16.sendmail	Timeout	2060	-
	17.sudo	64	17	47(73)
	18.ffmpeg	Timeout	80	-
SPEC 2006	19.mcf	1	1	0
	20.bzip2	931	69	862 (92)
	21.hmmer	143	23	120(83)
	22.sjeng	501	62	439(87)
	23.milc	11	7	4(36)
	24.h264ref	2977	451	2526(84)
	25.gobmk	Timeout	9818	-

Reduction in analysis time

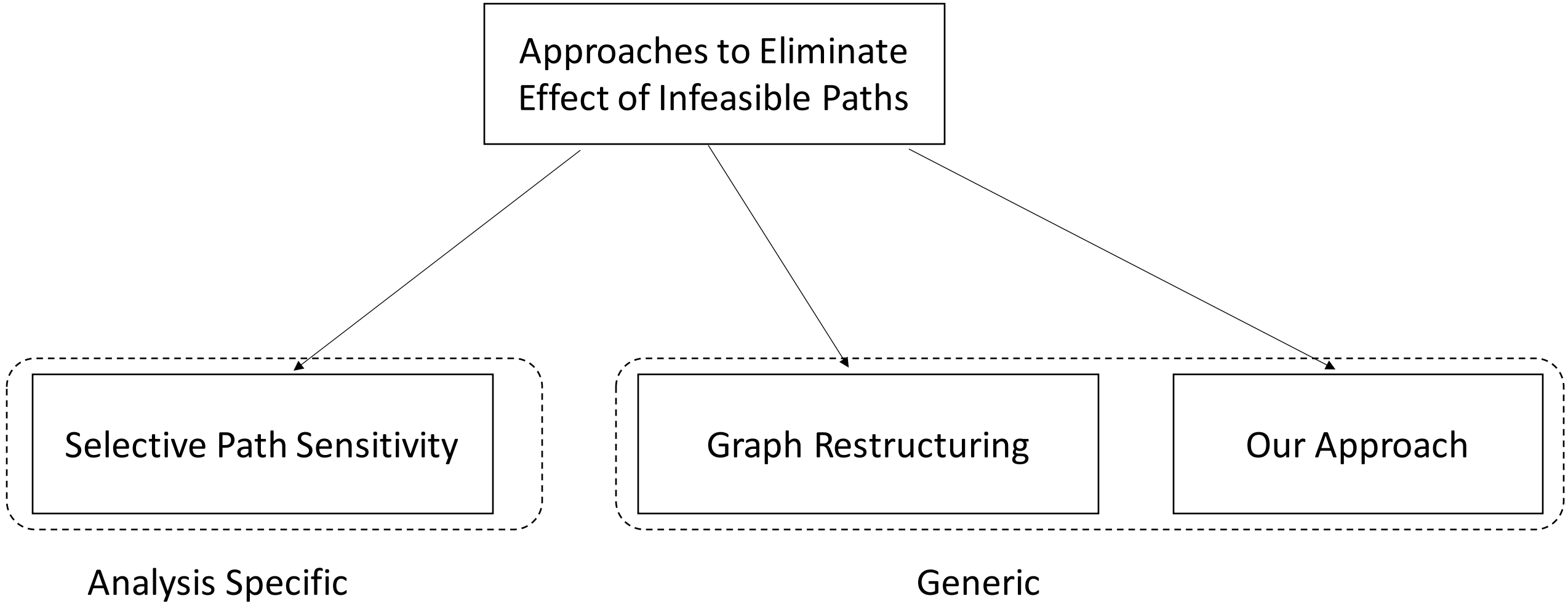
- Upto: 99%,
- Average: 43%
- Geometric mean: 19.8%



1->2->3->(2->3)^{39}->4->6



1->2->3->**4->2->3->2-**>5->6


Approaches to Eliminate Effect of Infeasible Paths
Selective Path Sensitivity
Graph Restructuring
Our Approach
Analysis Specific
Generic

Thank You ALL !

Guide
- Prof. Uday Khedker

RPC Members
- Prof. Amitabh Sanyal
- Prof. Supratim Biswas

TCS Mentors
- R Venkatesh
- Ulka Shrotri
- Shrawan Kumar
- Advaita Datar
- Tukaram Muske
- Ravindra Metta
- Kumar Madhukar

Family
- Nanda Pathade (Mom)
- Dadasaheb Pathade (Dad)
- Priyanka Pathade (Sister)

Friends
- Divyesh Unadkat
- Anshuman Dhuliya
- Punit Shah
- Rohit Saxena
- Arun
- Prashant
- Supriya
- Madhav
- Swati, Pritam, Nisha, Diptesh, Kevin