FM Update 2023

ViEqui

Optimal Stateless Model Checking based on *View-equivalence*

Sanjana Singh and Subodh Sharma

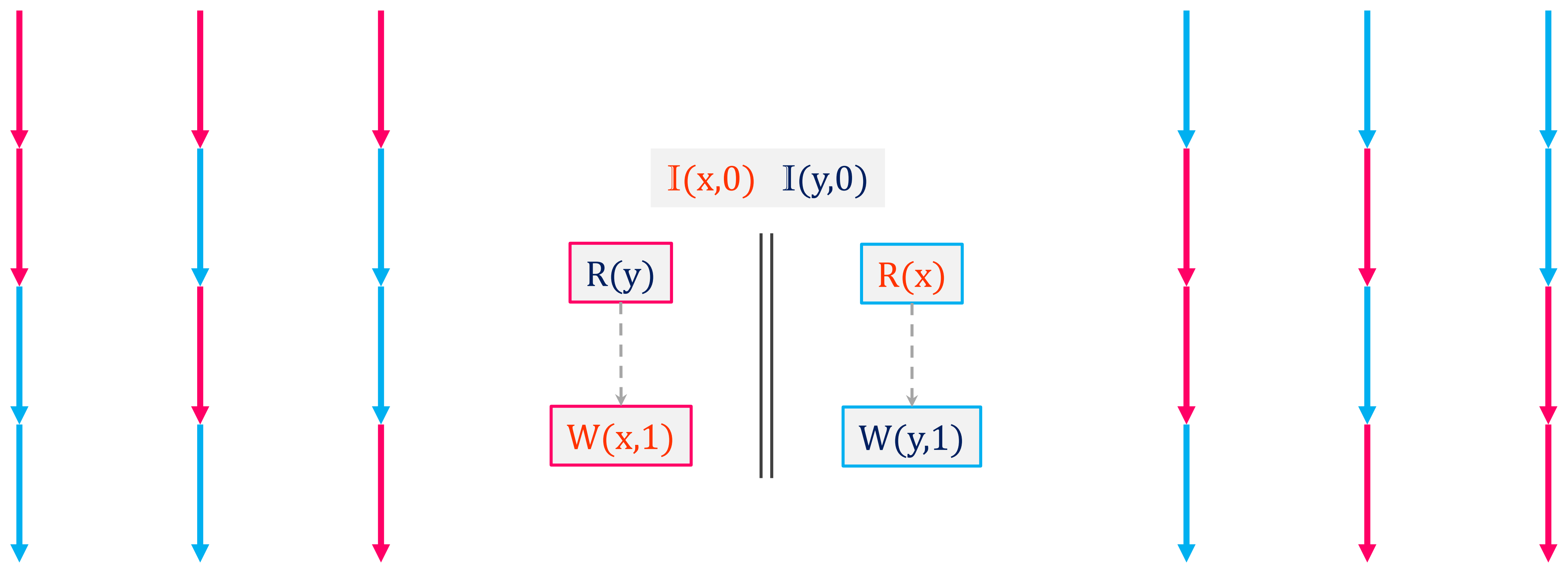
Indian Institute of Technology Delhi

Interleaving model of concurrency

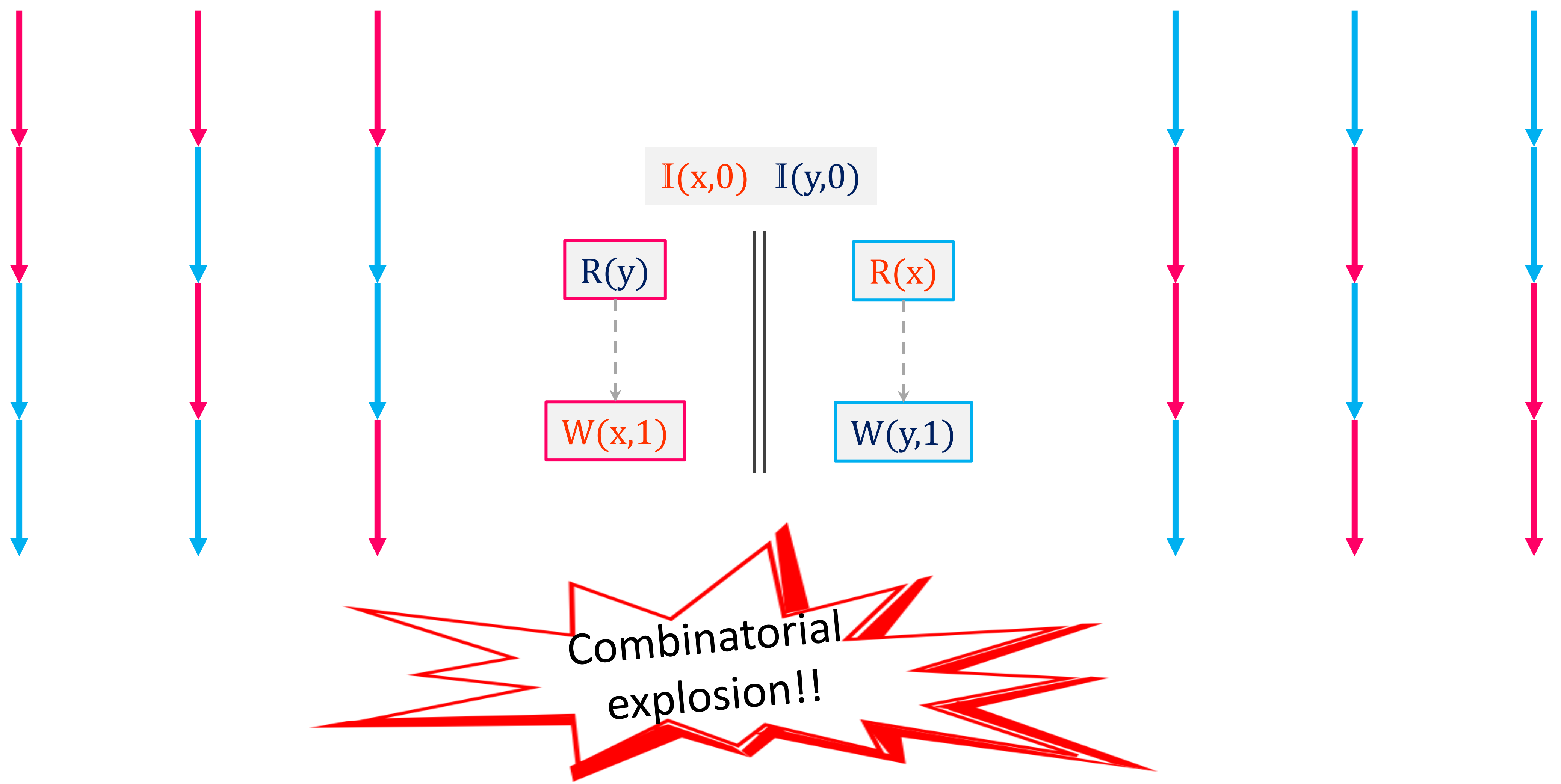
I(x,0) I(y,0)

R(y)	‖	R(x)
W(x,1)	‖	W(y,1)

Interleaving model of concurrency



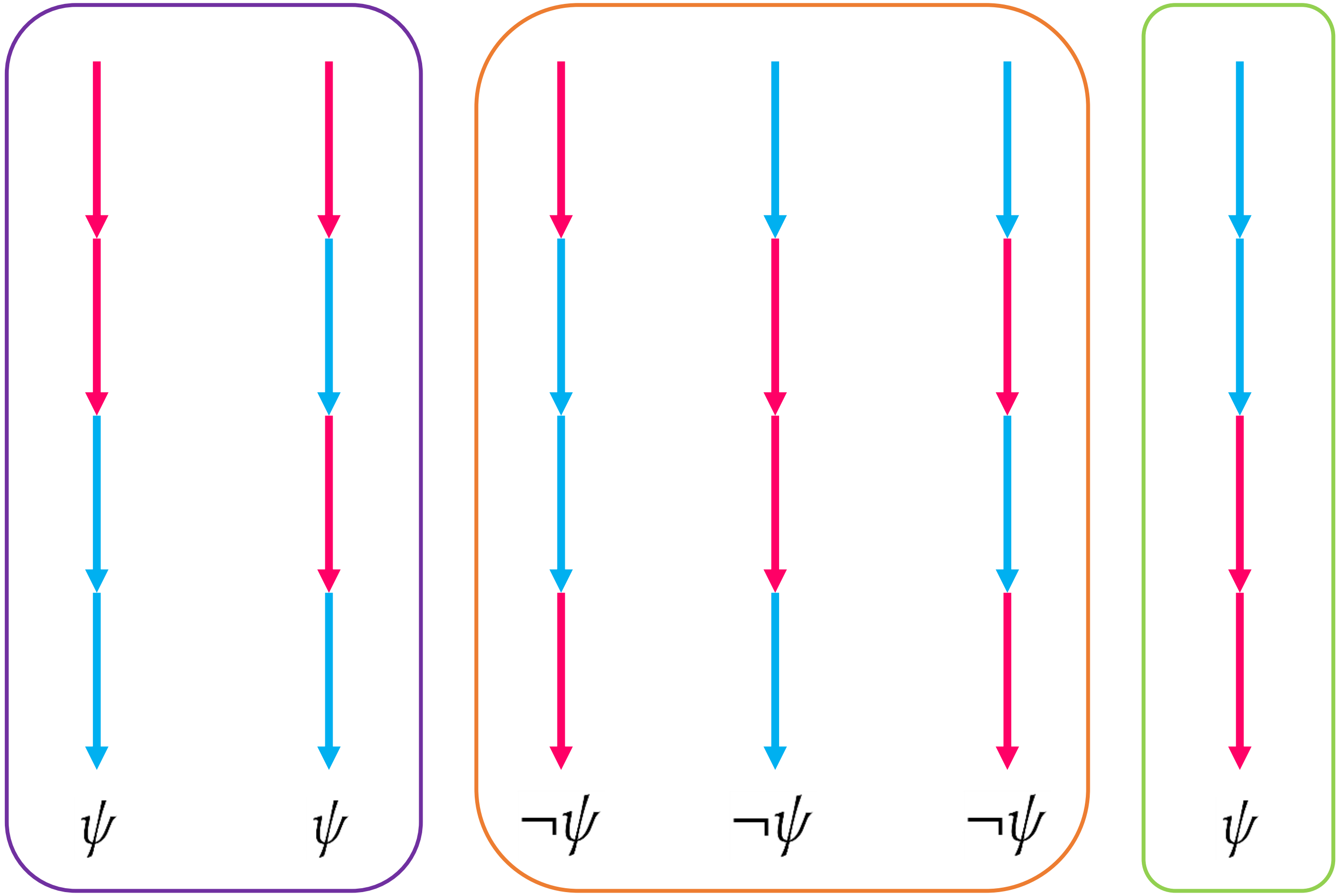
Interleaving model of concurrency



Equivalence classes

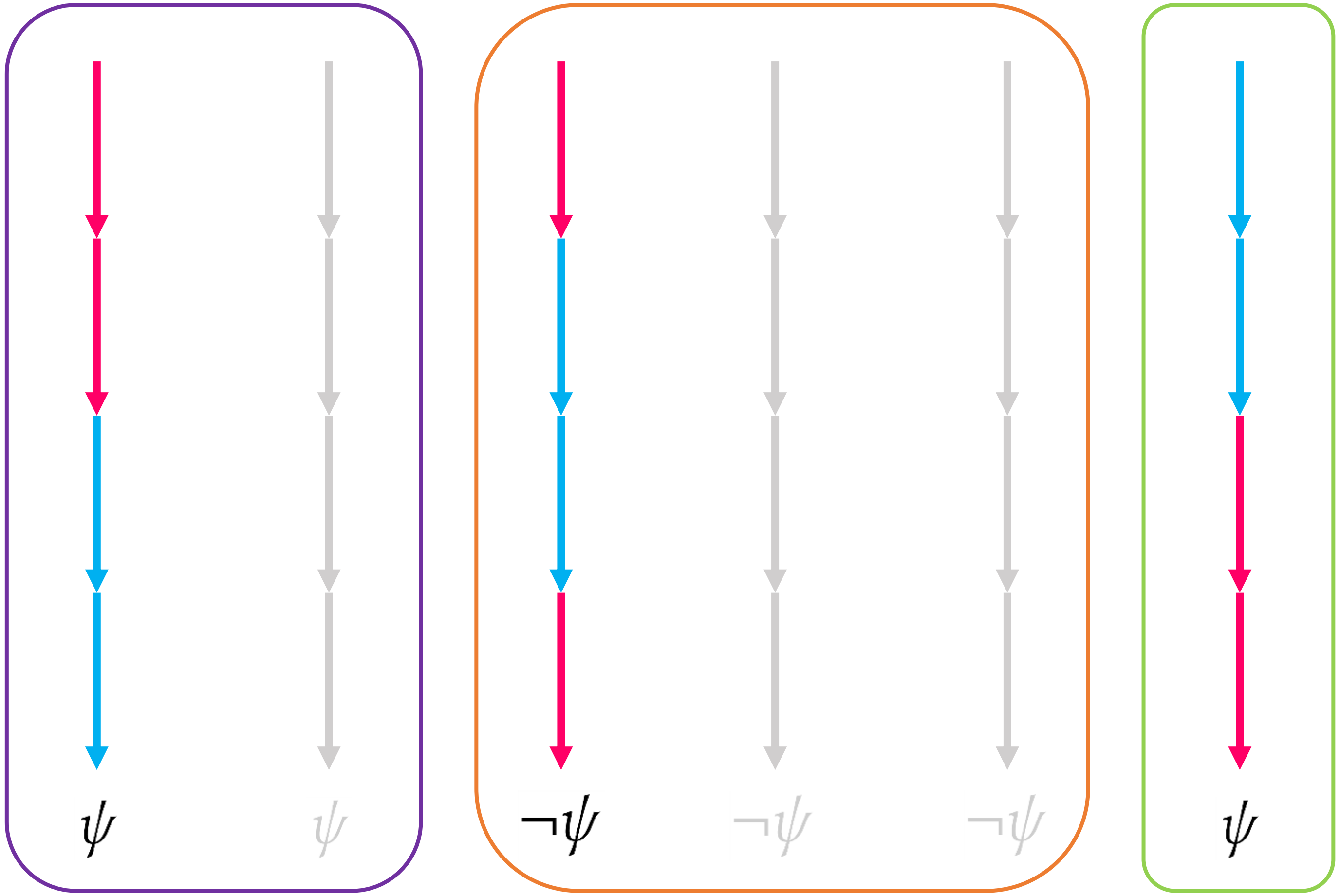
Stateless model checkers partition executions into

equivalence classes based on *equivalence relations*

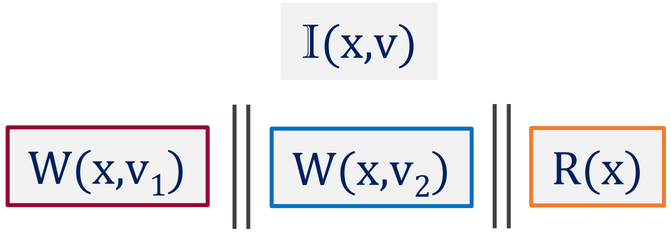


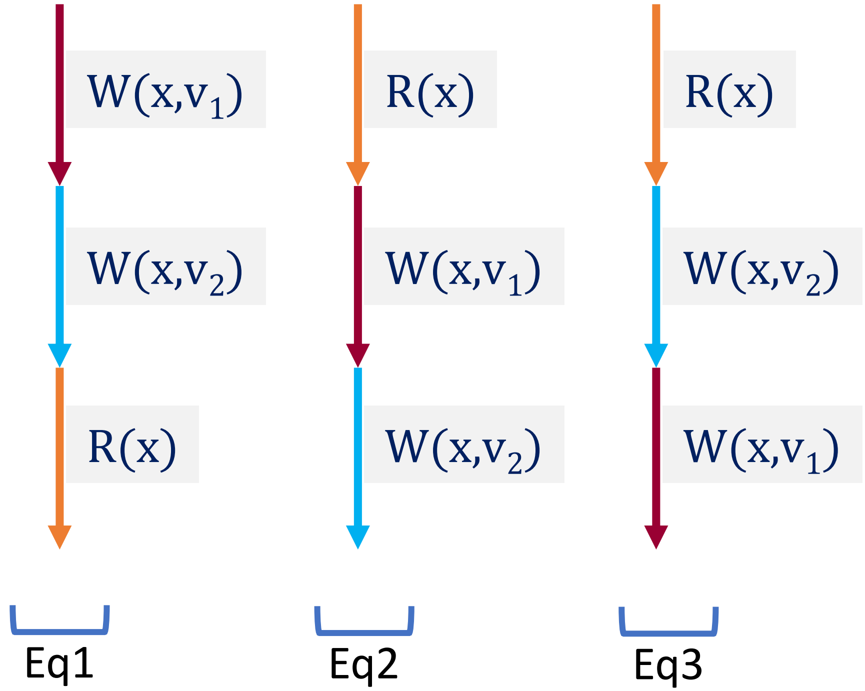
Stateless model checkers explore a reduced state graph

Stateless model checkers partition executions into *equivalence classes* based on *equivalence relations*



Existing equivalence relations





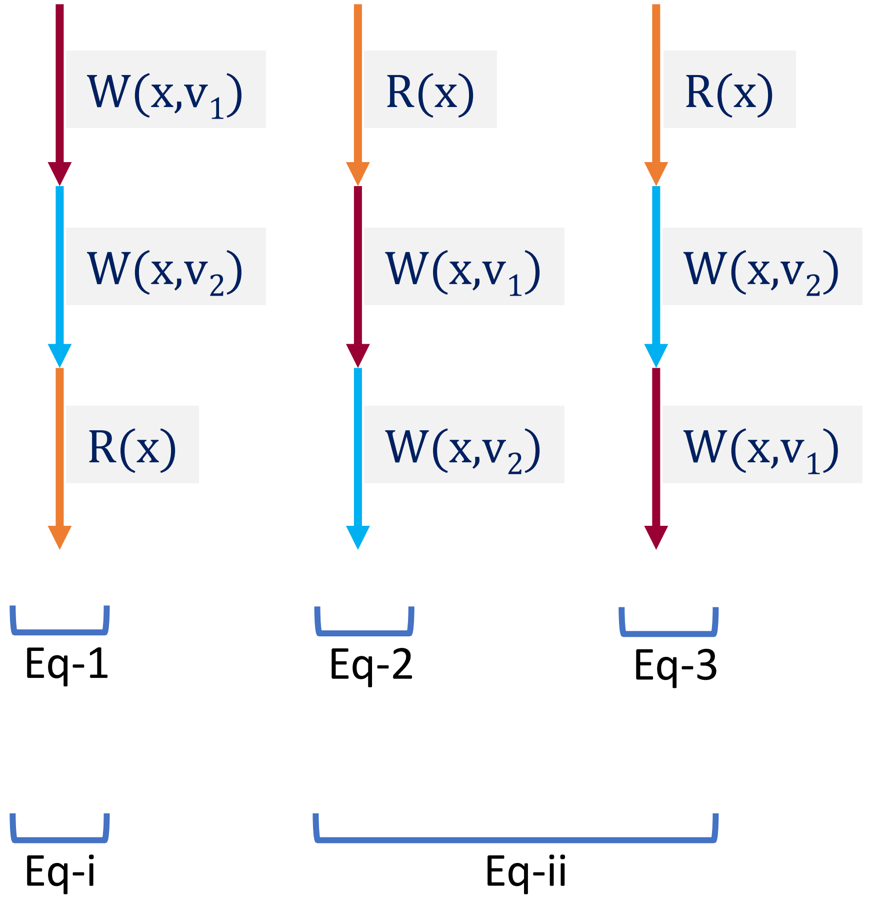
- Classical (*Mazurkiewicz*)

[Flanagan & Godefroid, POPL'05], [Abdulla et al., POPL'14] [Nguyen et al., CAV'18] [Zhang et al., PLDI'15] [Abdulla et al., TACAS'15]

Existing equivalence relations

$\mathbb{I}(x,v)$

$W(x,v_1) \parallel W(x,v_2) \parallel R(x)$



- Classical (*Mazurkiewicz*)
- reads-from

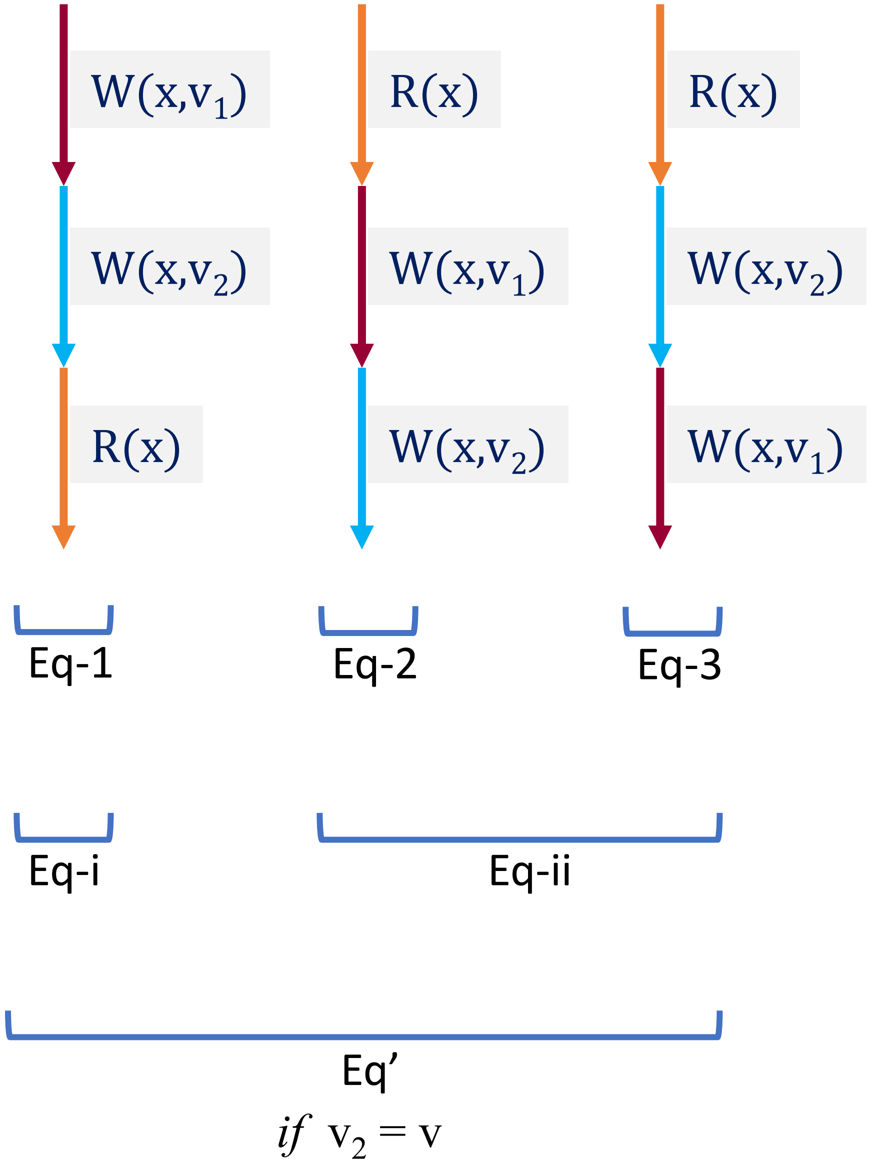
[Albert et al., CAV'17] [Chalupa et al., POPL'18] [Abdulla et al., OOPSLA'18]

Existing equivalence relations

$\mathbb{I}(x,v)$

$\boxed{W(x,v_1)} \parallel \boxed{W(x,v_2)} \parallel \boxed{R(x)}$

- Classical (*Mazurkiewicz*)
- reads-from
- reads-value-from



[Agarwal et al., CAV '21]

Existing equivalence relations

equivalence relation

finer ⟷ coarser

higher ⟷ lower

#equivalence classes

View-equivalence

equivalence relation

finer

coarser

higher

lower

#equivalence classes

Classical equivalence

reads-from equivalence

View-equivalence

Classical modulo observers equivalence

value equivalence

reads-value-from equivalence

coarsest existing

lowest #equivalence classes

View-equivalence

$$\textit{Exn-1} \quad \sim \quad \textit{Exn-2}$$

$$\{\ r_1, r_2, r_3\ \} \qquad \{\ r_1, r_2, r_3\ \}$$

I. same set of *read events*

View-equivalence

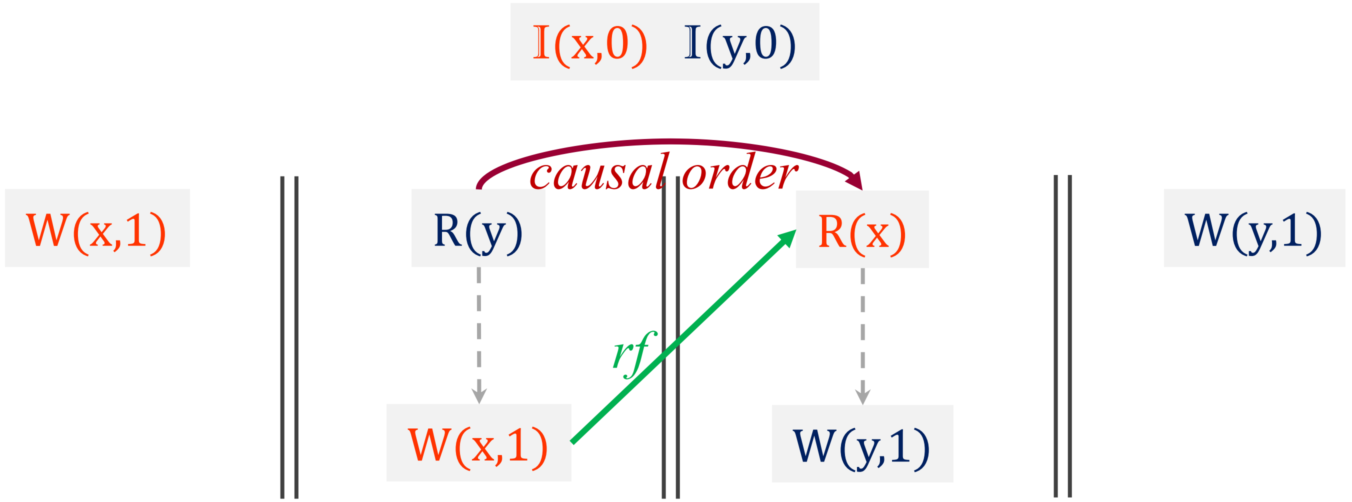
$$Exn\text{-}1 \sim Exn\text{-}2$$

$$\{\ r_1 \rightarrow v_1 \quad r_2 \rightarrow v_2 \quad r_3 \rightarrow v_3\ \} \qquad \{\ r_1 \rightarrow v_1 \quad r_2 \rightarrow v_2 \quad r_3 \rightarrow v_3\ \}$$

I. same set of *read events*

II. Each *read event* reads the same value

reads-value-from *vs* view-equivalence



6

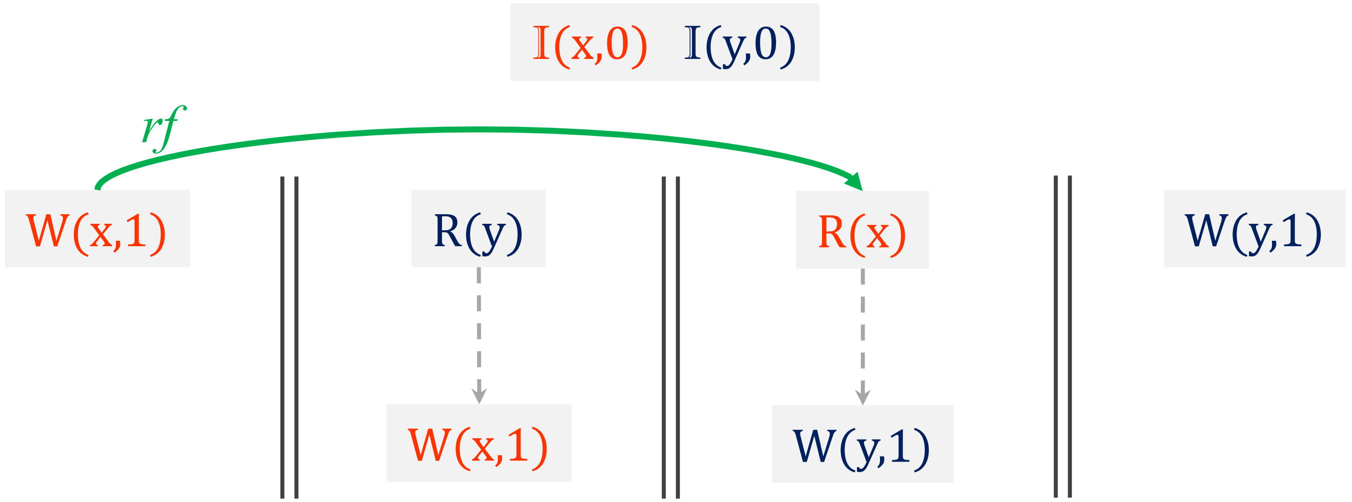
reads-value-from

equivalence classes

4

view

equivalence class



Tradeoff of causality

equivalence relation

finer — coarser

higher — lower

ordering on events

Tradeoff of causality

- SMCs discover equivalence classes *on-the-fly*

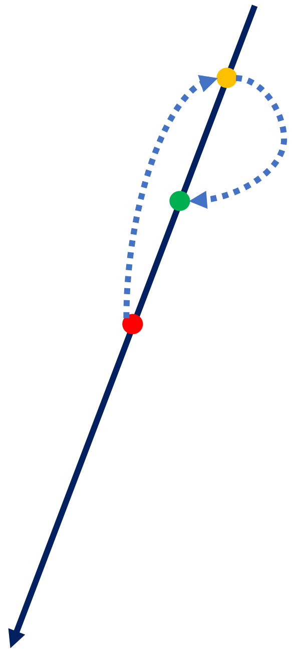
  - explore and execution (eq-1)

Eq-1

Tradeoff of causality

- SMCs discover equivalence classes *on-the-fly*

  - explore and execution (eq-1)
  - manipulate order of events

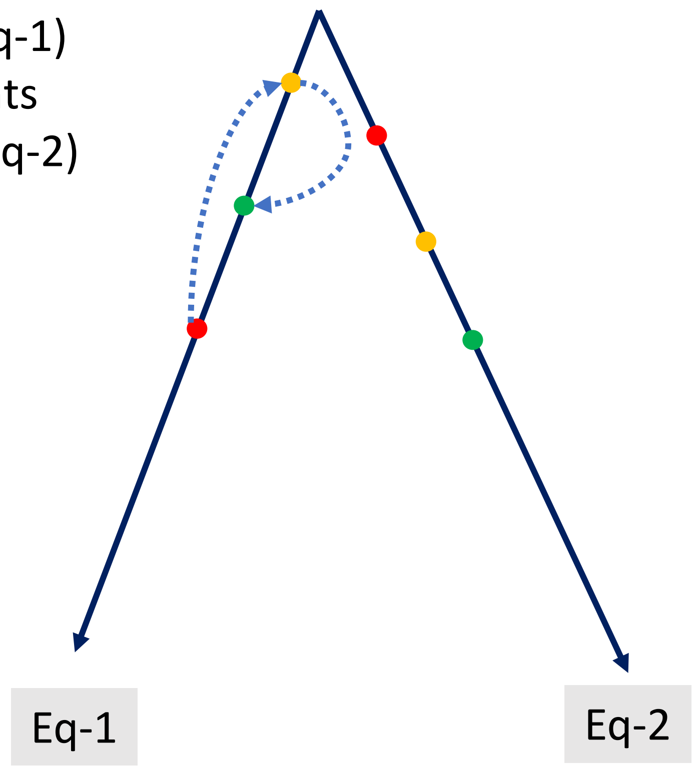


Eq-1

Tradeoff of causality

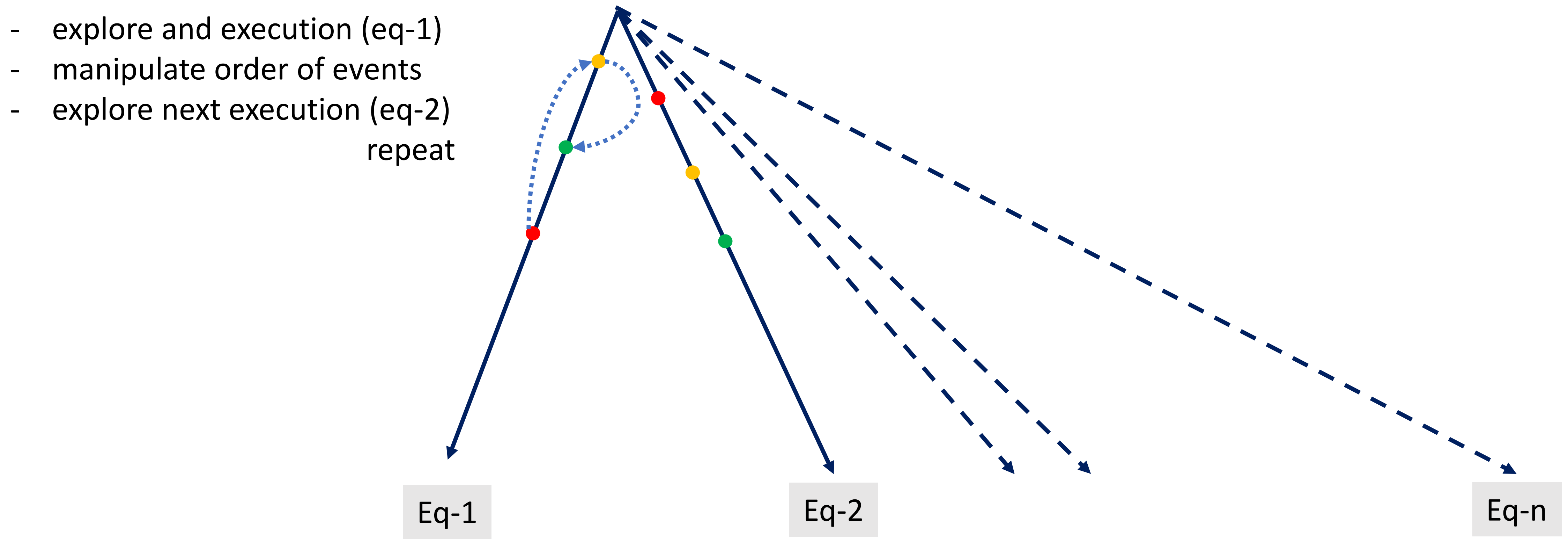
- SMCs discover equivalence classes *on-the-fly*

- explore and execution (eq-1)
- manipulate order of events
- explore next execution (eq-2)



Tradeoff of causality

- SMCs discover equivalence classes *on-the-fly*



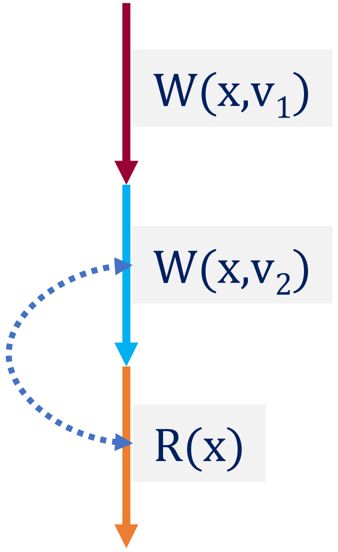
Tradeoff of causality

equivalence relation

finer coarser

higher lower

ordering on events



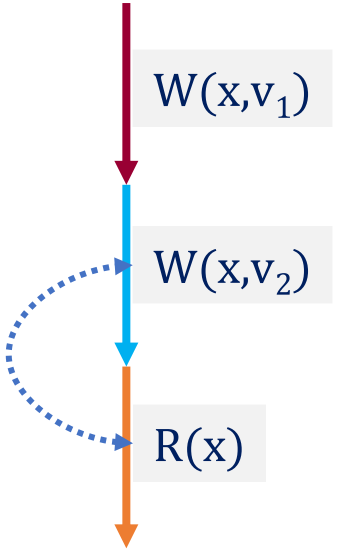
Tradeoff of causality

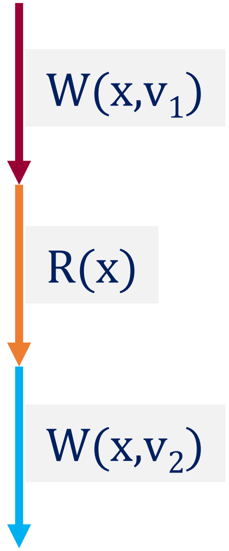
equivalence relation

finer ⟷ coarser

higher ⟷ lower

ordering on events





Tradeoff of causality

equivalence relation

finer

coarser

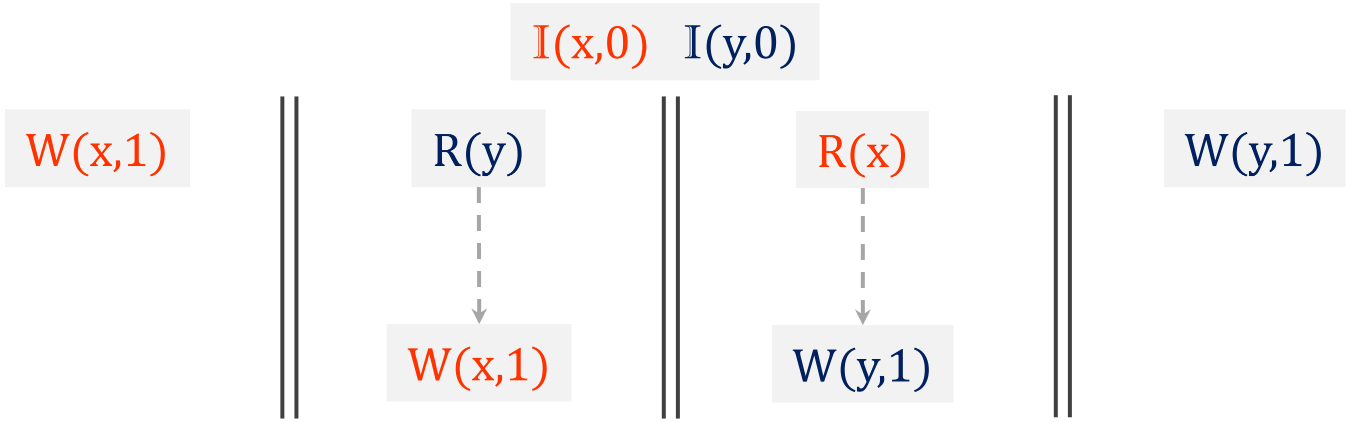
higher

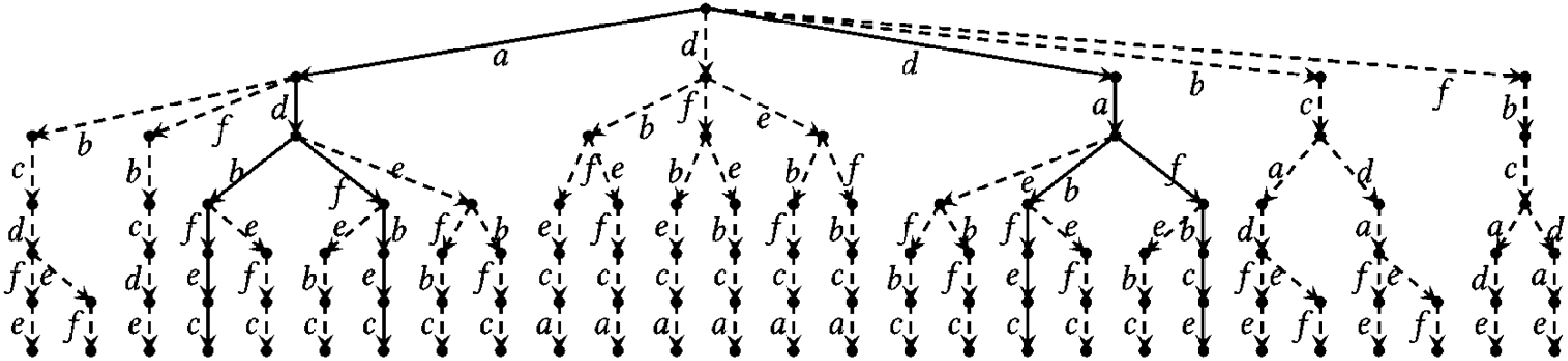
lower

ordering on events

Has necessary information for constructing a coherent ordering

Tradeoff of causality

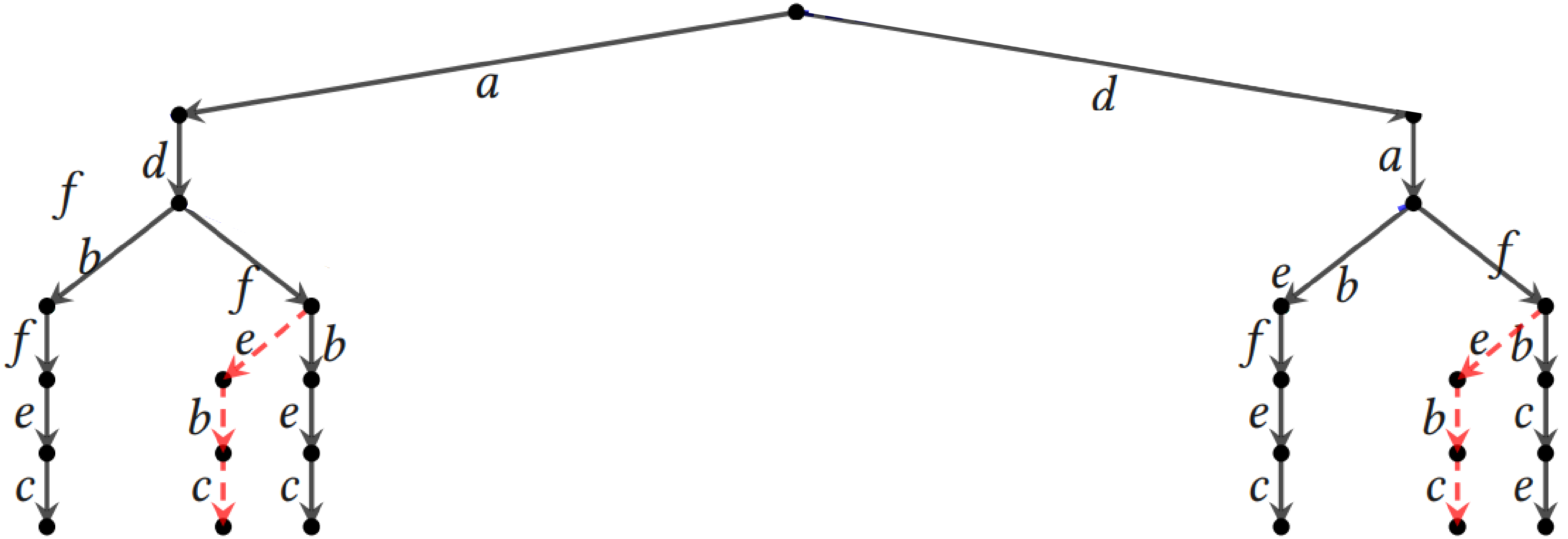




Tradeoff of causality

I(x,0) I(y,0)

W(x,1) || R(y) W(x,1) || R(x) W(y,1) || W(y,1)



Tradeoff of causality

I(x,0) I(y,0)

W(x,1) || R(y) → W(x,1) || R(x) → W(y,1) || W(y,1)

#loops	ODPOR (classical)		RVF-SMC		ViEqui	
	#Seq	Time	#Seq	Time	#Seq	Time
10	-	Timeout	3703196	705.69	40	0.06
20	-	Timeout	-	Timeout	80	0.28

Tradeoff of causality

equivalence relation

finer ⟷ coarser

higher ⟷ lower

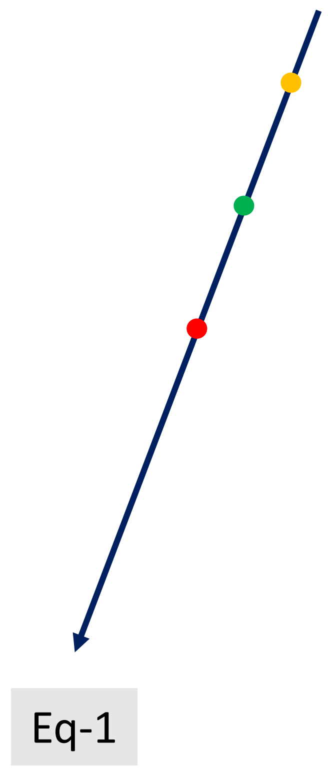
ordering on events

+ Has necessary information for constructing a coherent ordering

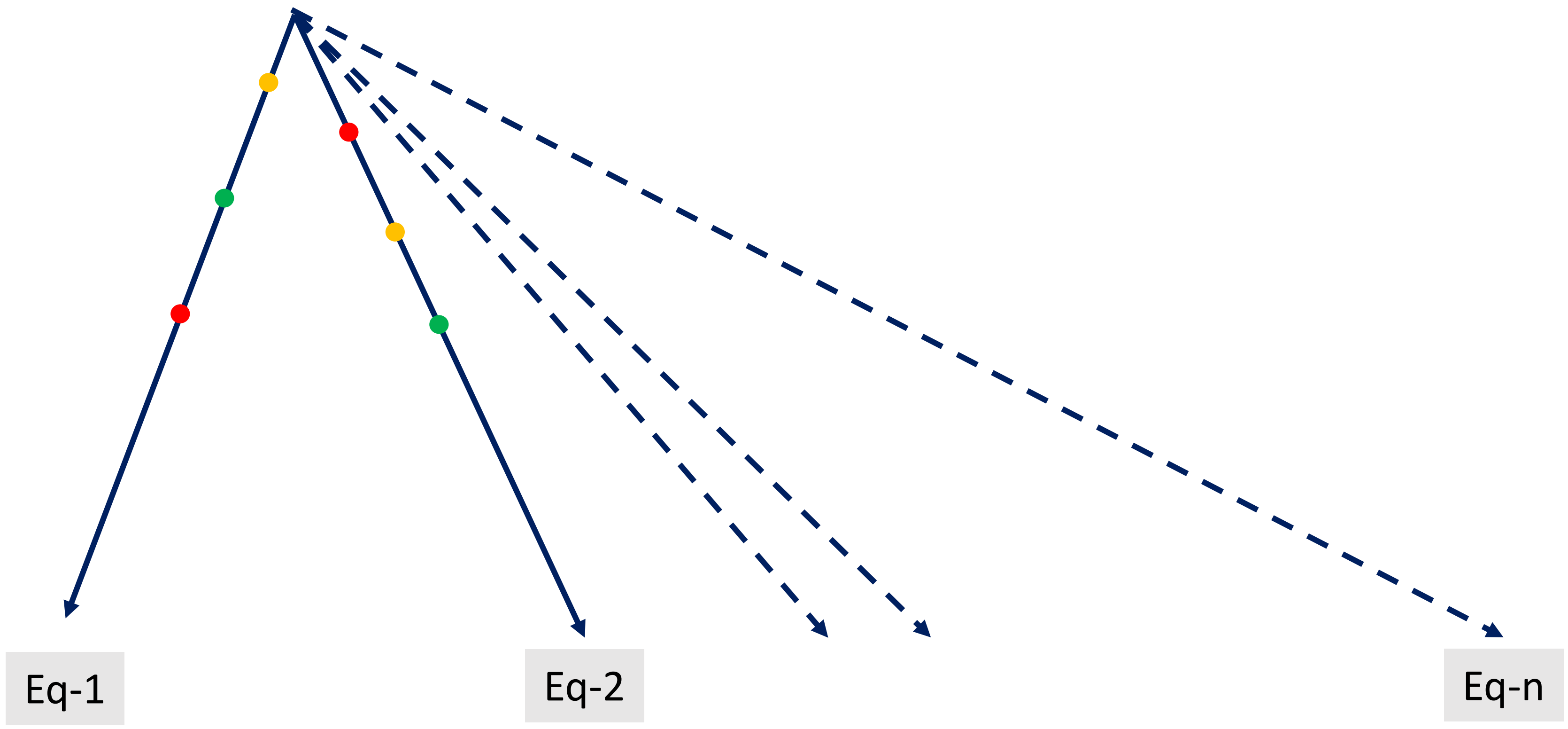
+ can reduce the number of explorations and achieve higher performance
- compute coherence operationally

ViEqui. SMC under view-equivalence

ViEqui. SMC under view-equivalence



ViEqui. SMC under view-equivalence



ViEqui. SMC under view-equivalence

ViEqui Scheduler

ViEqui. SMC under view-equivalence

ViEqui Scheduler

defines

Execution order of events

representative of an equivalence class

ViEqui. SMC under view-equivalence

ViEqui Scheduler

defines

Execution order of events

representative of an equivalence class

uses

Scheduling directives

what has been examined previously?
what needs to be explored next ?

ViEqui. SMC under view-equivalence

ViEqui Scheduler

defines → Execution order of events → *representative of an equivalence class*

uses ↓

Scheduling directives

what has been examined previously?

what needs to be explored next ?

φ: a boolean formula

next: an sequence of events

ViEqui. SMC under view-equivalence

ViEqui Scheduler

defines

Execution order of events

representative of an equivalence class

uses

Scheduling directives

what has been examined previously?
what needs to be explored next ?

φ: a boolean formula

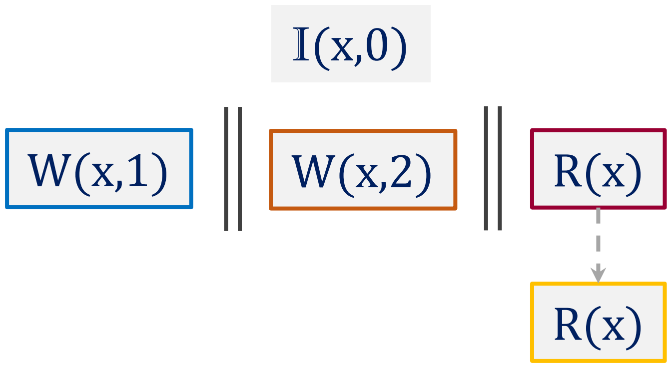
next: an sequence of events

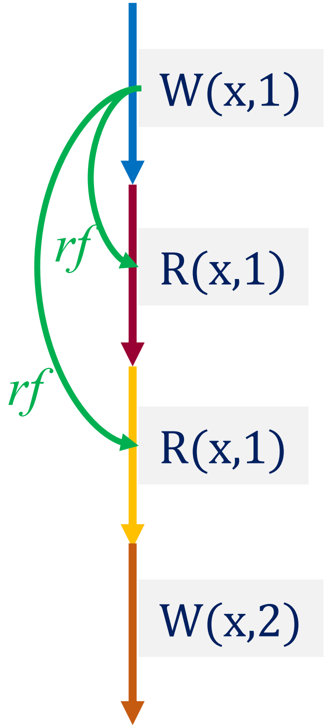
Eq-i

Scheduling directives

what has been examined previously?

what needs to be explored next ?

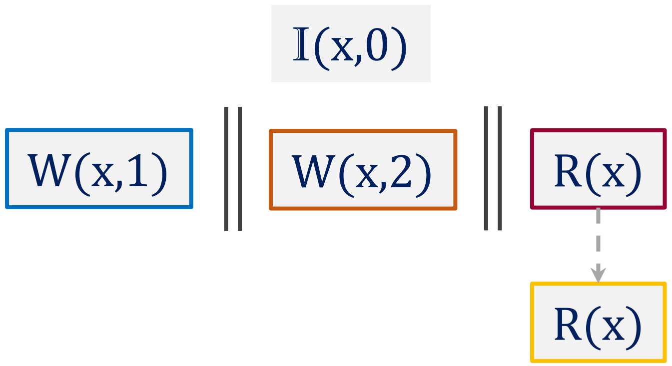


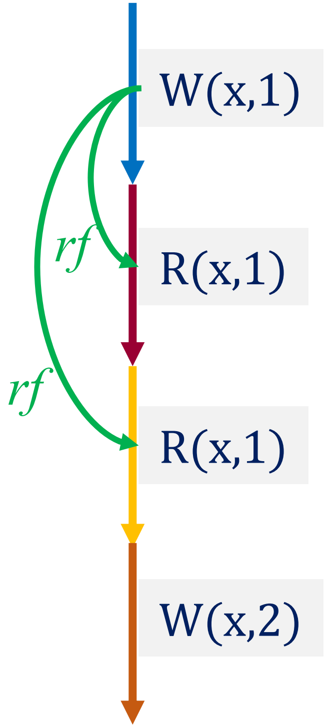


Scheduling directives

what has been examined previously? R(x) → 1 R(x) → 1

what needs to be explored next ?





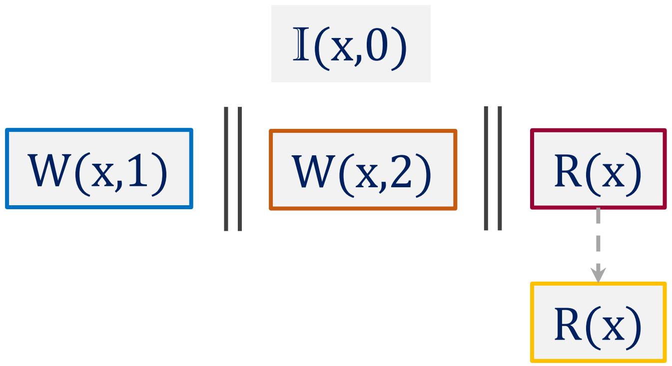
Scheduling directives

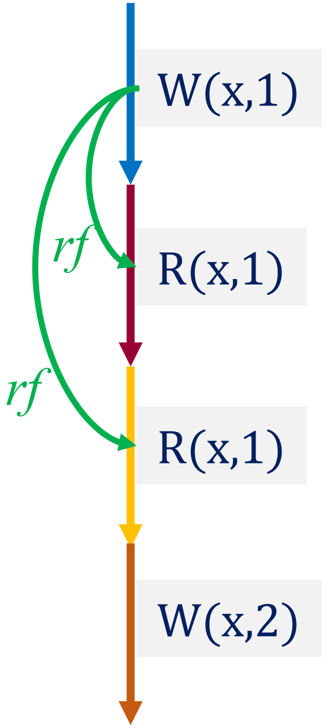
what has been examined previously?

R(x) → 1	R(x) → 1

what needs to be explored next ?

R(x) → 1	R(x) → 2
R(x) → 0	R(x) → ?
R(x) → 2	R(x) → ?





Scheduling directives

(φ*:* a boolean formula) *what has been examined previously?*

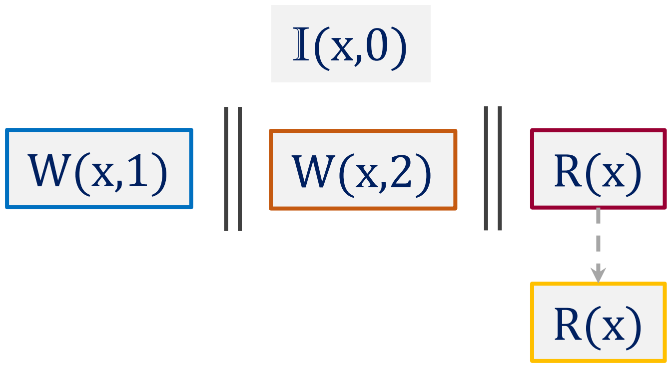
R(x) → 1 R(x) → 1

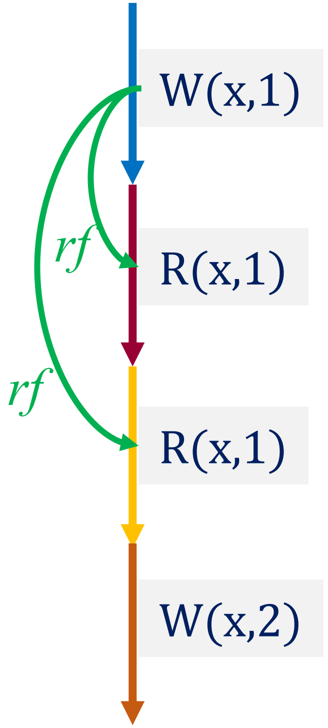
(*next:* an sequence of events) *what needs to be explored next ?*

R(x) → 1 R(x) → 2

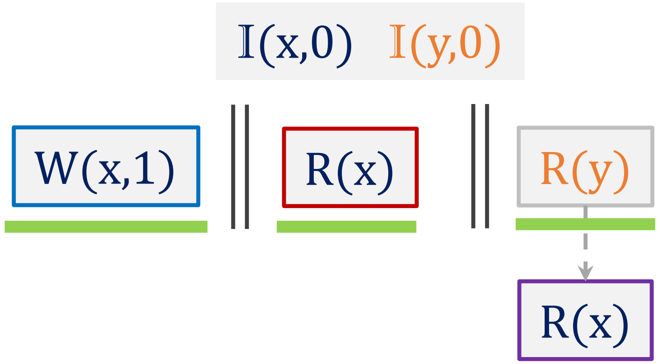
R(x) → 0 R(x) → ?

R(x) → 2 R(x) → ?





Computing scheduling directives: *forward analysis*



$\bullet s_{[\langle\rangle]}$

Computing scheduling directives: *forward analysis*

I(x,0) I(y,0)

W(x,1) || R(x) || R(y)

R(x)

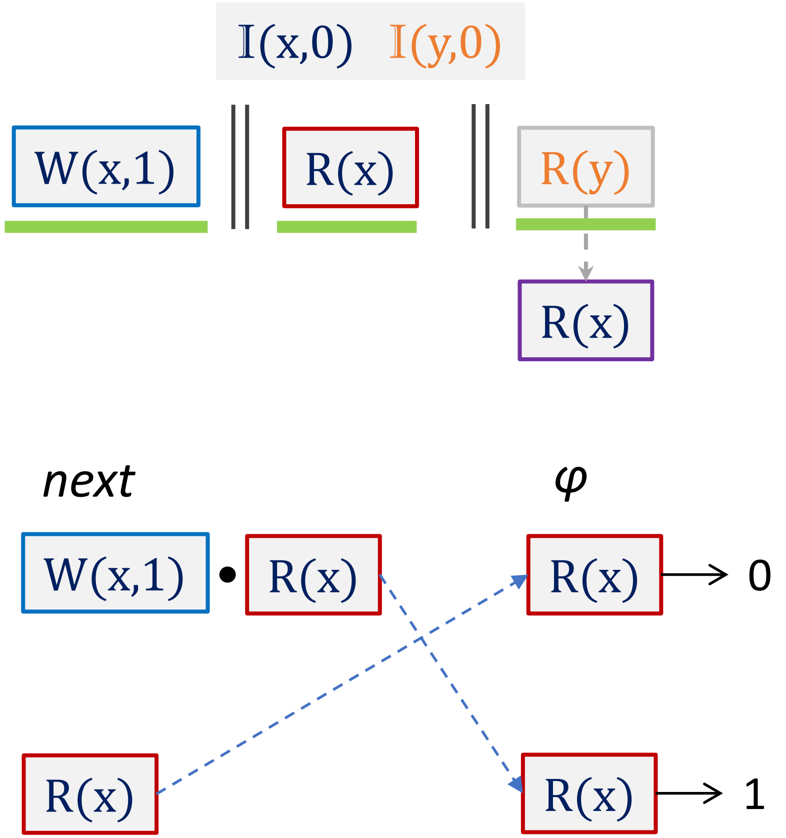
$\bullet s_{[\langle\rangle]}$

next

W(x,1) • R(x)

R(x)

Computing scheduling directives: *forward analysis*



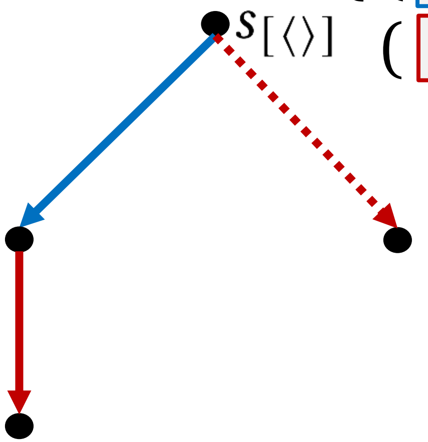
- $s_{[\langle\rangle]}$

Computing scheduling directives: *forward analysis*

I(x,0) I(y,0)

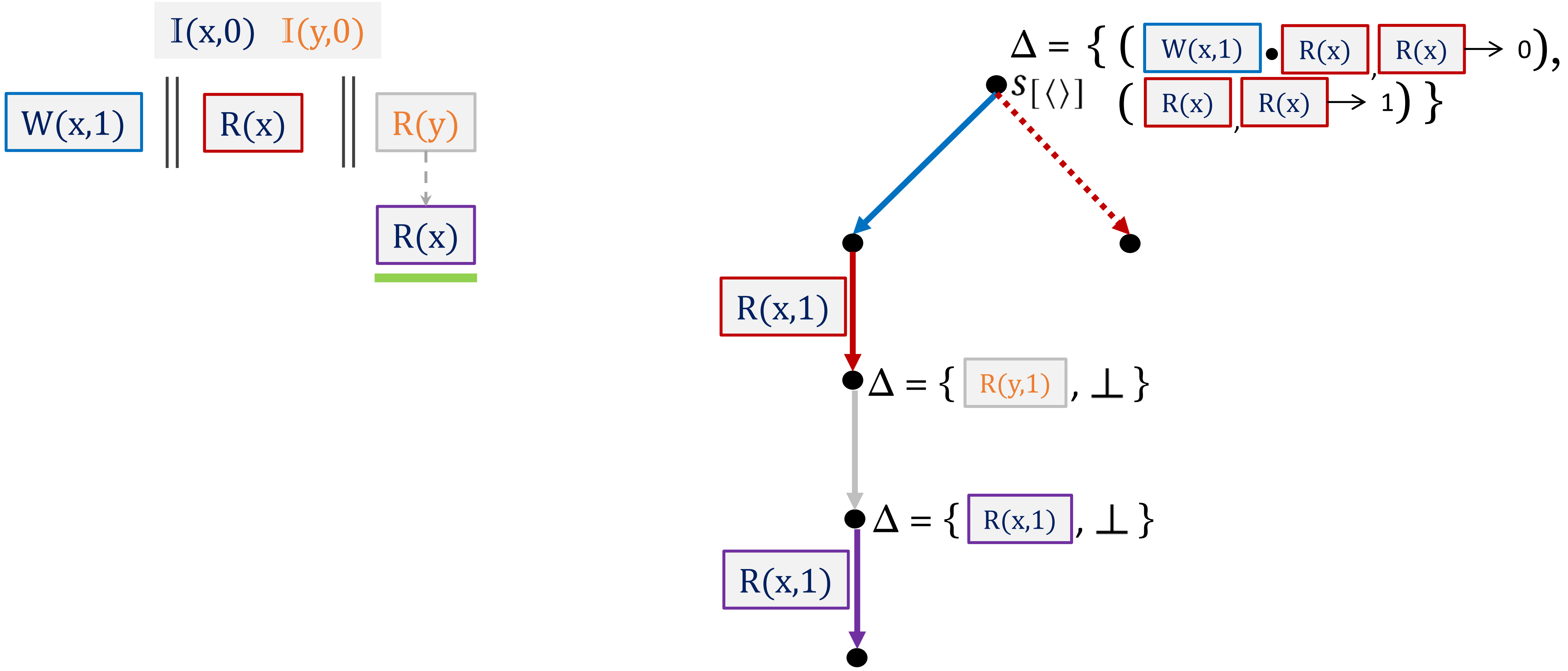
$\boxed{W(x,1)} \parallel \boxed{R(x)} \parallel \boxed{R(y)}$

$\boxed{R(x)}$

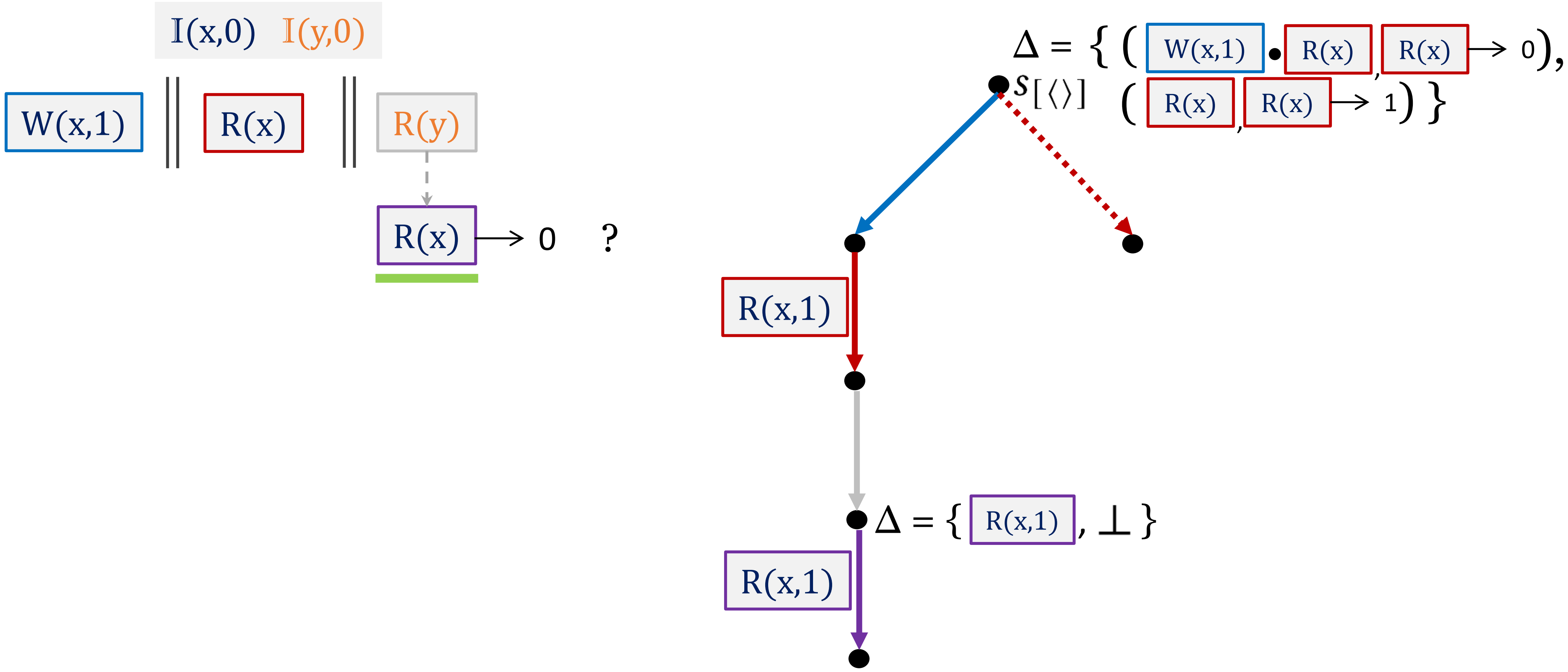


$$\Delta = \{ (\boxed{W(x,1)} \bullet \boxed{R(x)}, \boxed{R(x)} \rightarrow 0), (\boxed{R(x)}, \boxed{R(x)} \rightarrow 1) \}$$

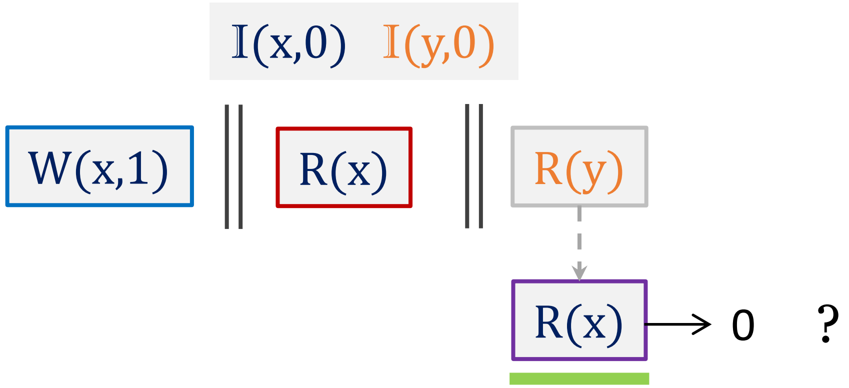
Computing scheduling directives: *forward analysis*



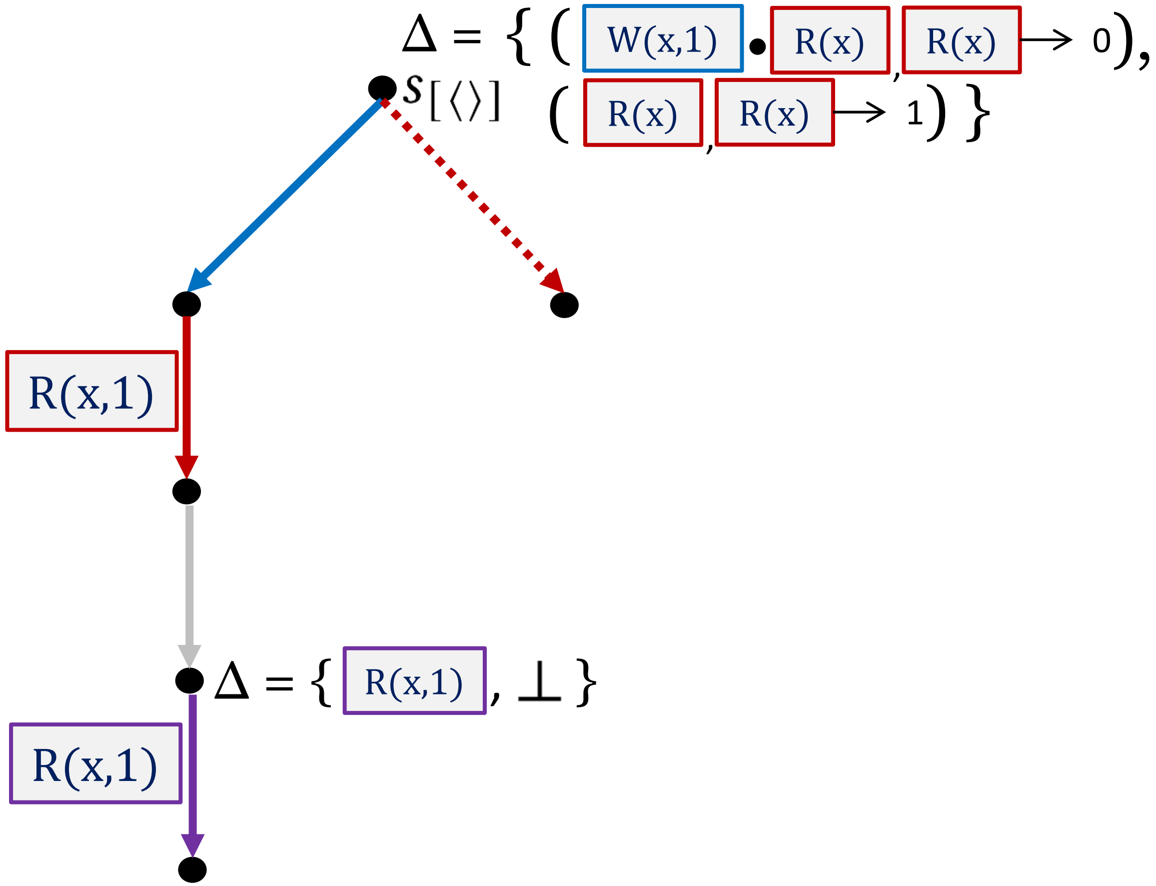
Computing scheduling directives: *forward analysis*



Computing scheduling directives: *backward analysis*



A. choose a state $s[\langle\rangle]$



Computing scheduling directives: *backward analysis*

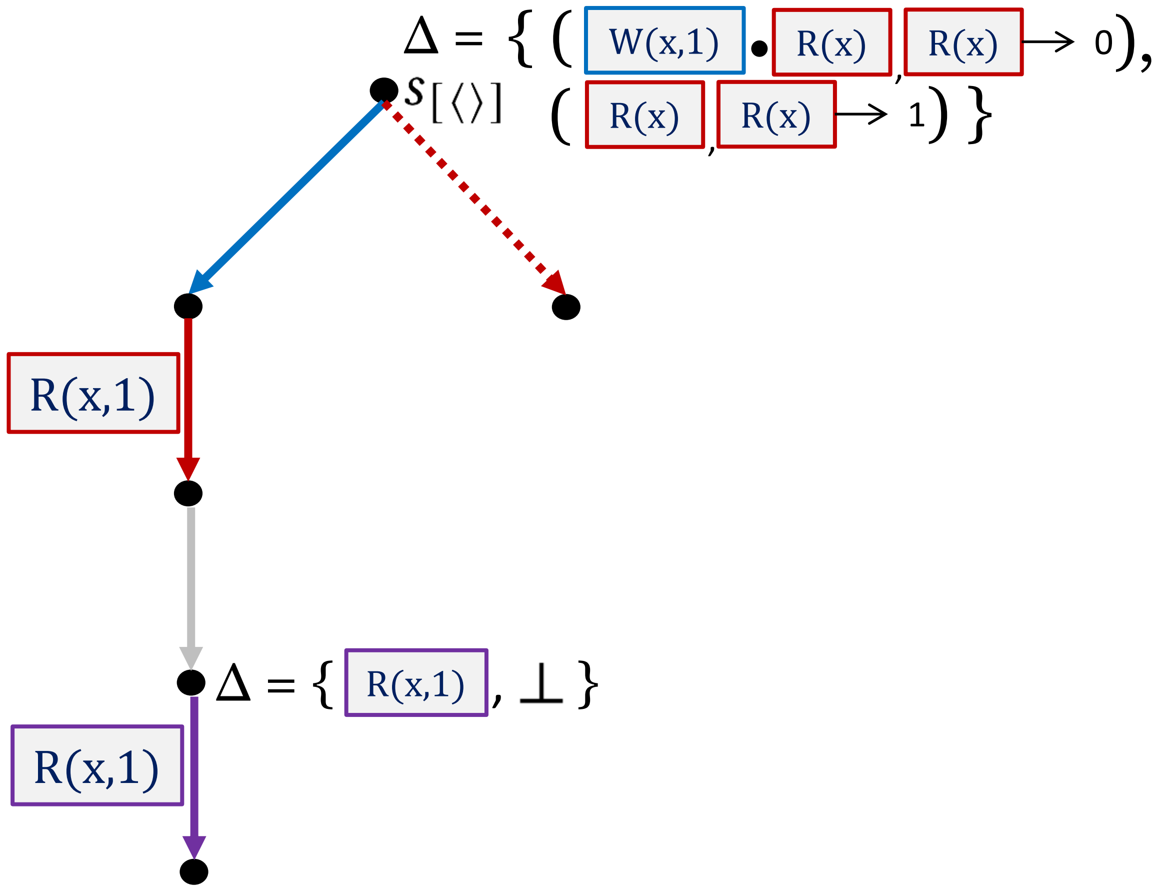
I(x,0) I(y,0)

W(x,1) || R(x) || R(y)

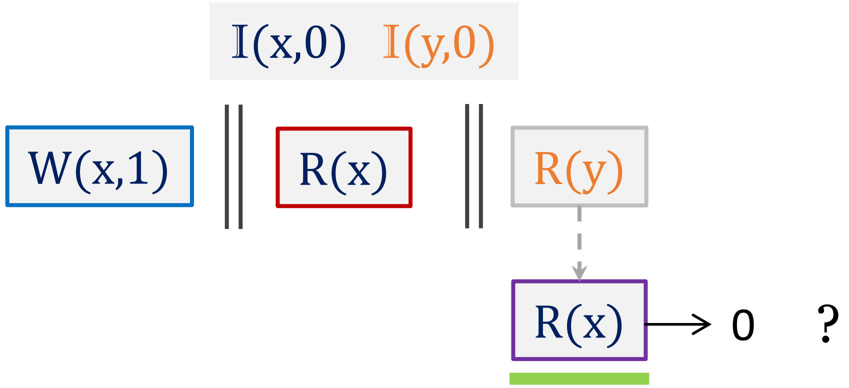
R(x) ⟶ 0 ?

A. choose a state $s[\langle\rangle]$

B. compute *next*



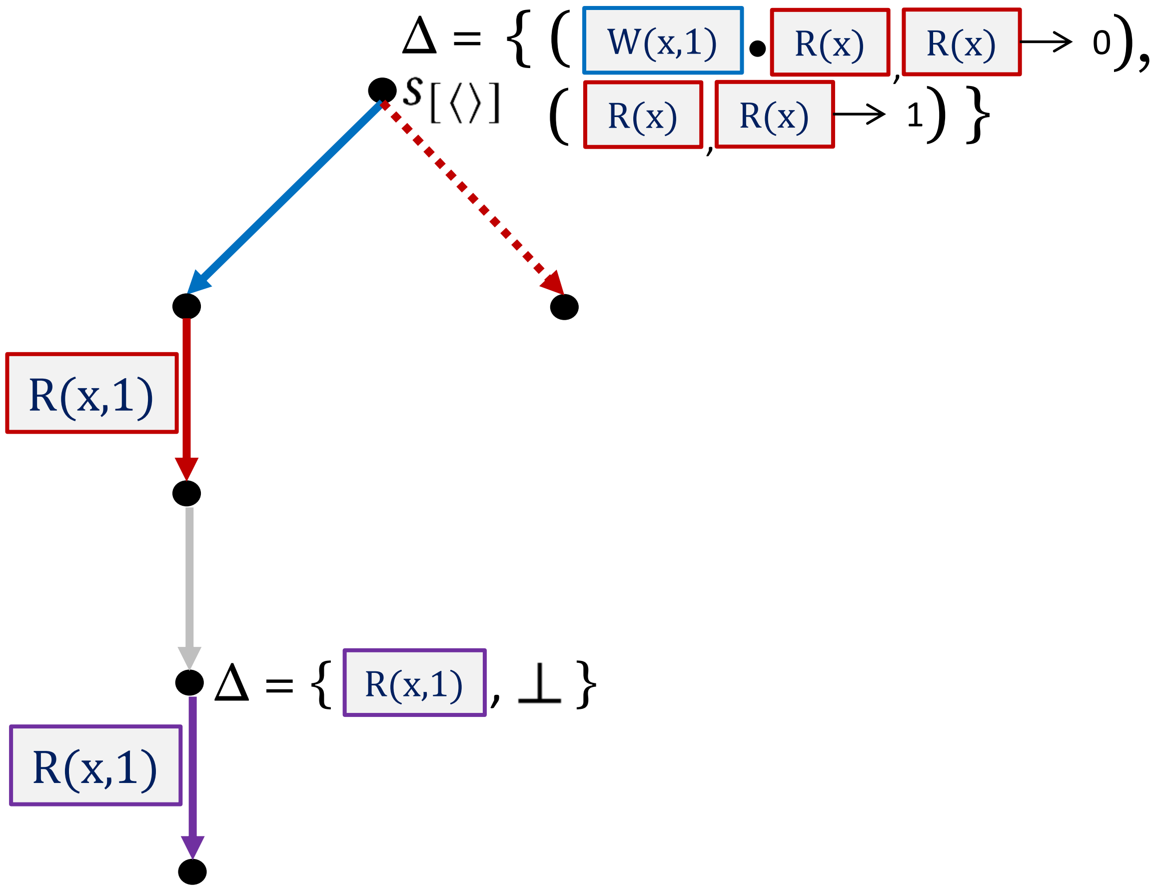
Computing scheduling directives: *backward analysis*



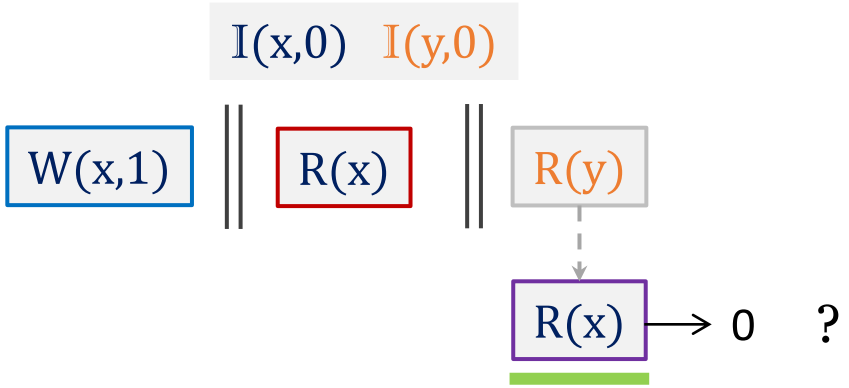
A. choose a state $s[\langle\rangle]$

B. compute *next*

    1. enable write $\langle\rangle$



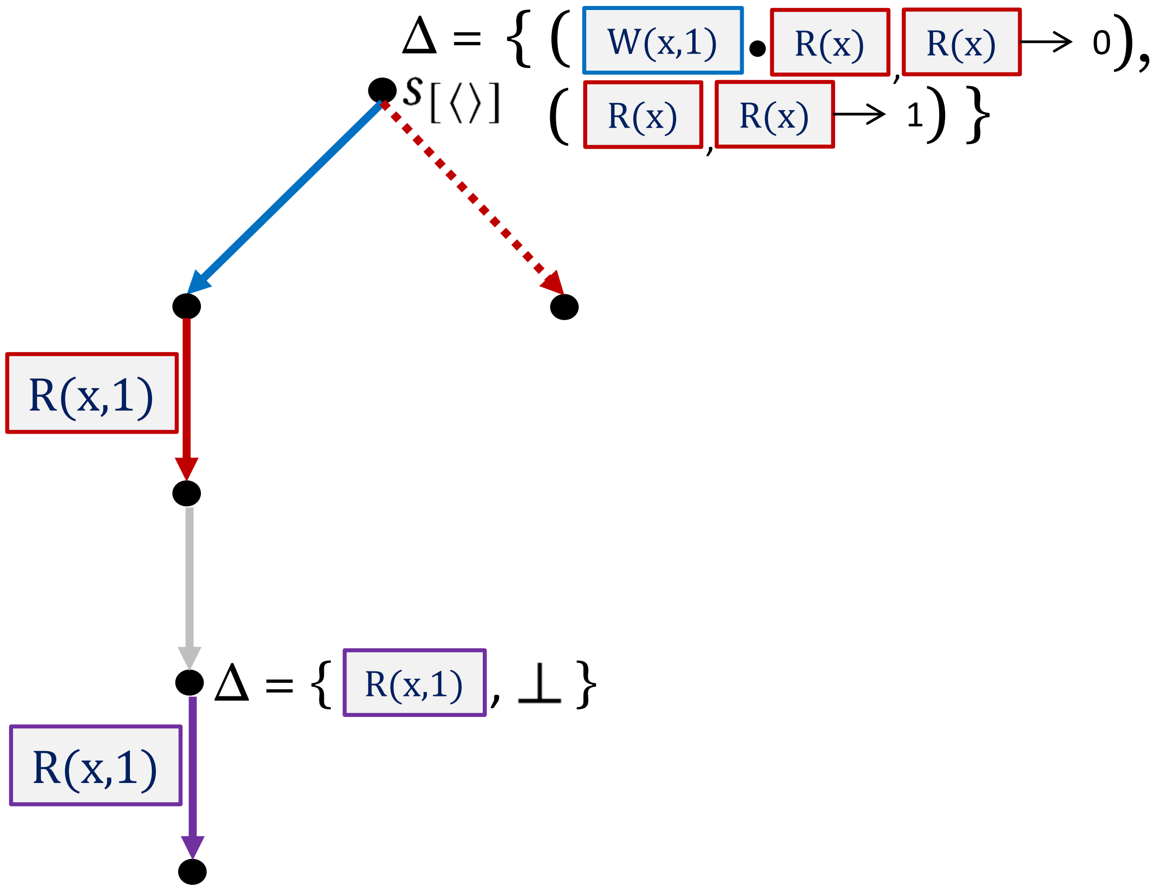
Computing scheduling directives: *backward analysis*



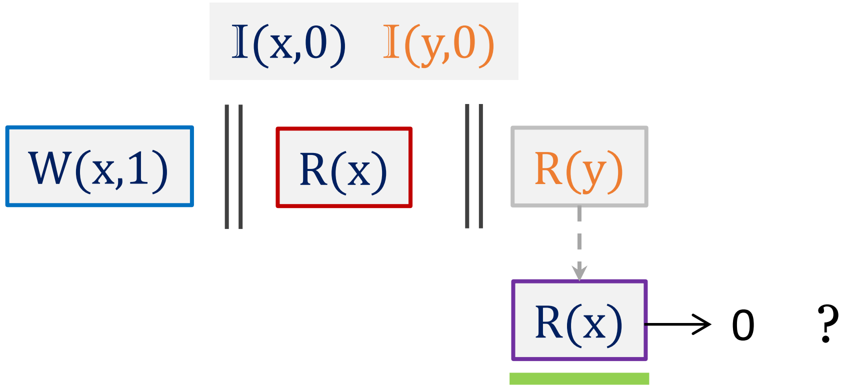
A. choose a state $s[\langle\rangle]$

B. compute *next*

1. enable write $\langle\rangle$
2. enable read R(y) • R(x)



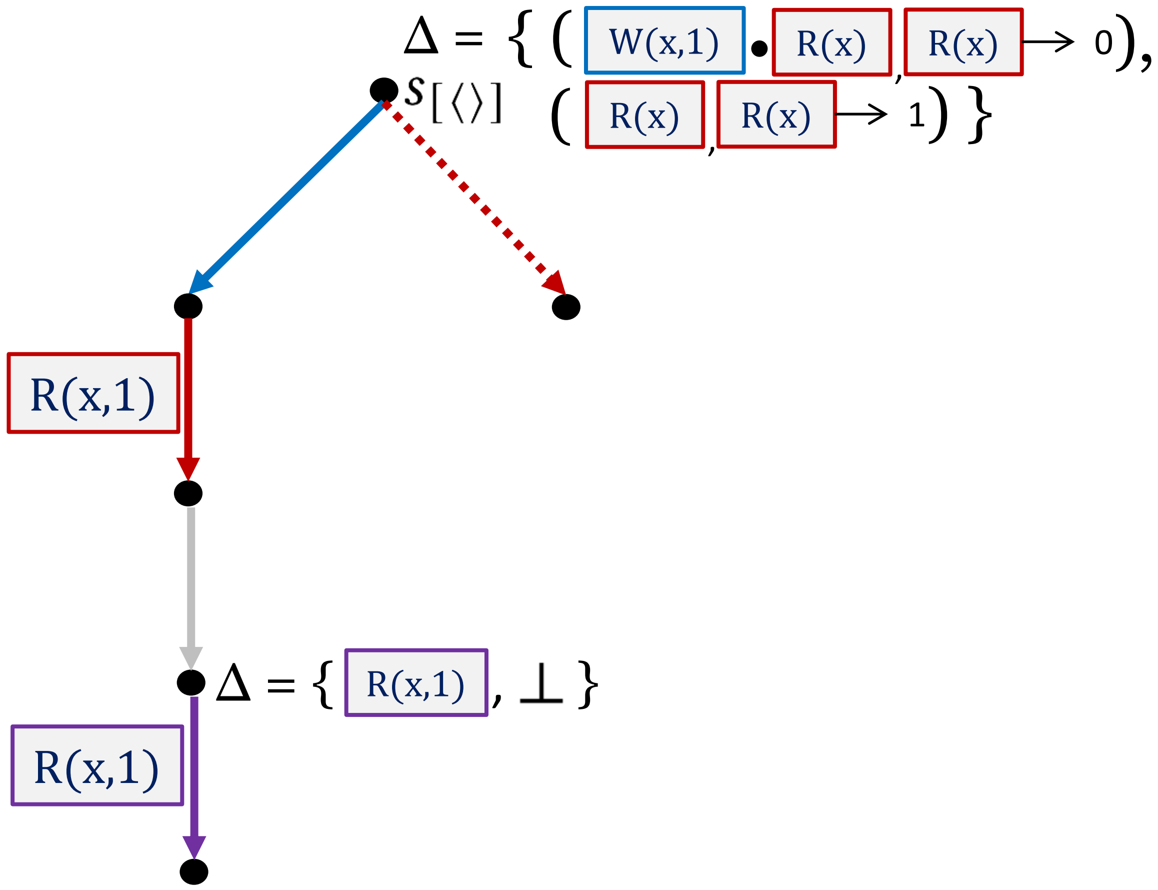
Computing scheduling directives: *backward analysis*



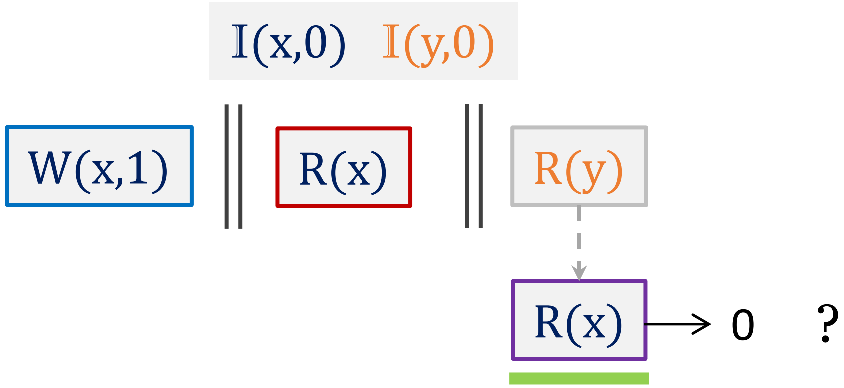
A. choose a state $s[\langle\rangle]$

B. compute *next*

1. enable write $\langle\rangle$
2. enable read R(y) • R(x)
3. necessary *rf* $\langle\rangle$ • R(x)



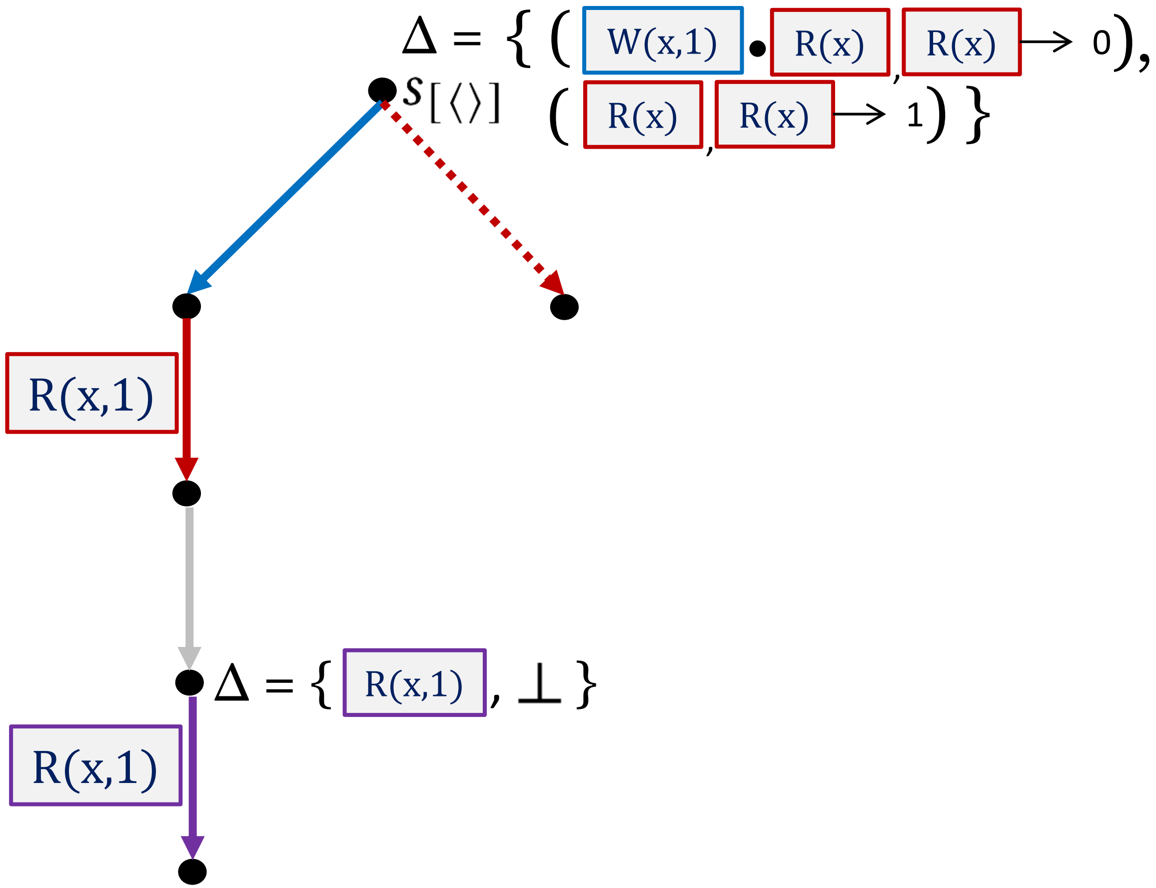
Computing scheduling directives: *backward analysis*



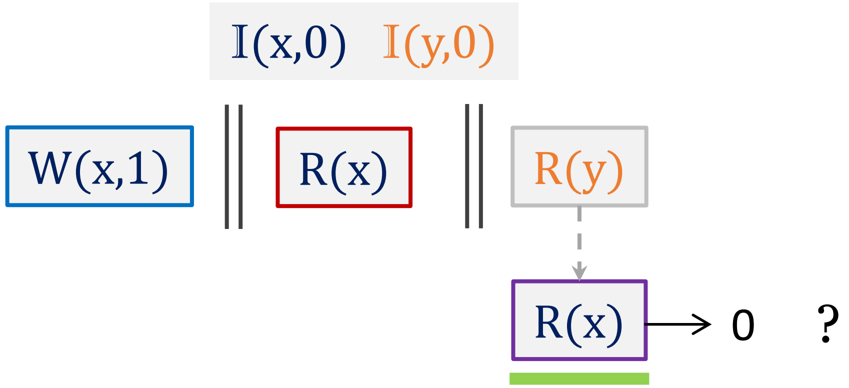
A. choose a state $s[\langle\rangle]$

B. compute *next*

1. enable write $\langle\rangle$
2. enable read R(y) • R(x)
3. necessary *rf* $\langle\rangle$ • R(x)
4. other values W(x,1) • R(x)



Computing scheduling directives: *backward analysis*

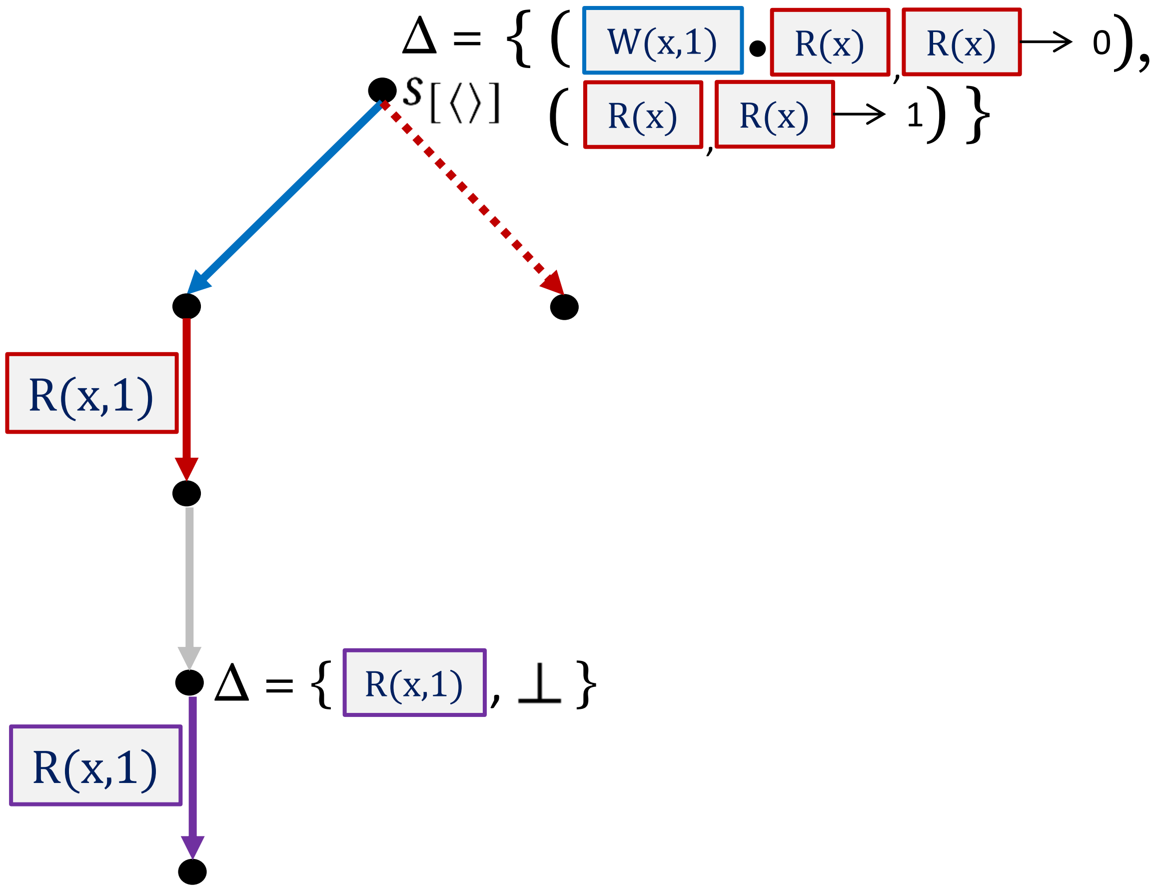


A. choose a state $s[\langle\rangle]$

B. compute *next*

1. enable write $\langle\rangle$
2. enable read R(y) • R(x)
3. necessary *rf* $\langle\rangle$ • R(x)
4. other values W(x,1) • R(x)

1 ⊕ 2 ⊕ 3 ⊕ 4 R(y) • R(x) • W(x,1) • R(x)

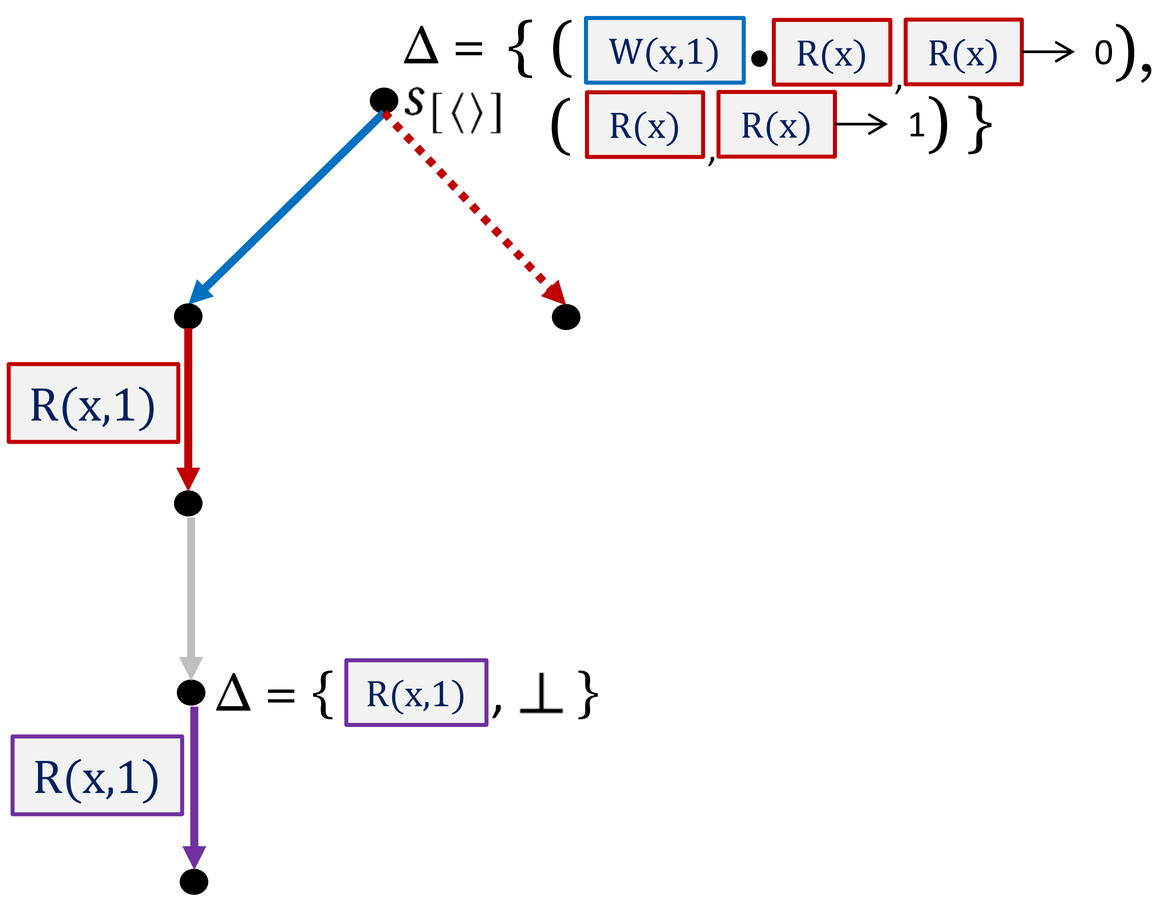


Computing scheduling directives: *backward analysis*

I(x,0) I(y,0)

W(x,1) || R(x) || R(y)

R(x) ⟶ 0 ?



A. choose a state $s[\langle\rangle]$

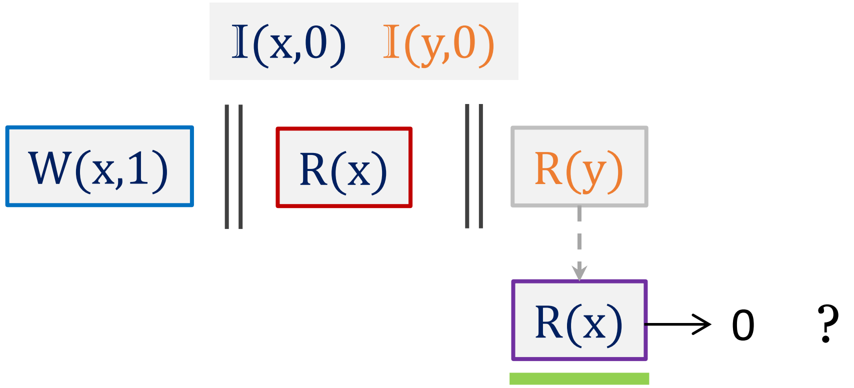
B. compute *next*

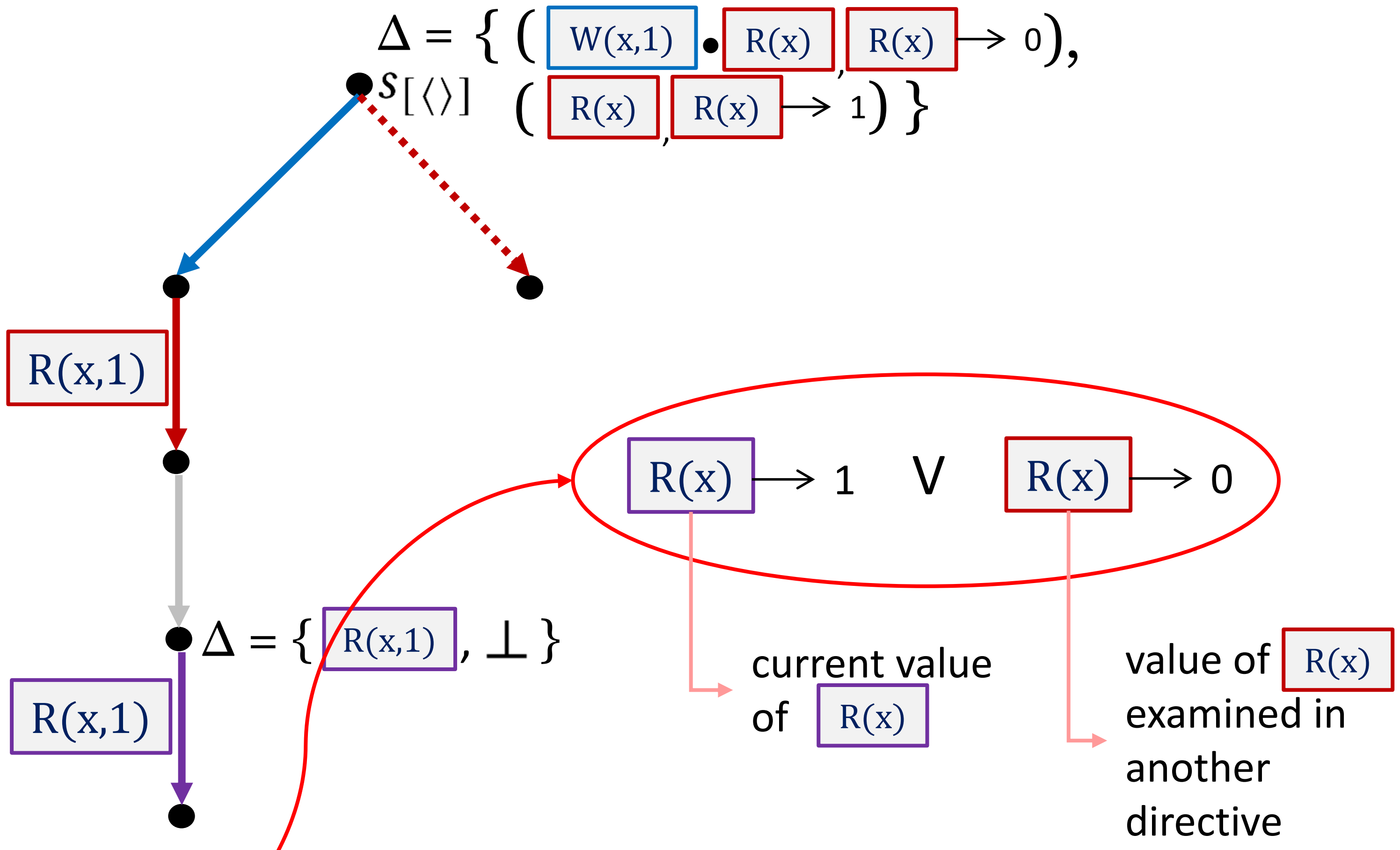
1. enable write $\langle\rangle$
2. enable read R(y) • R(x)
3. necessary *rf* $\langle\rangle$ • R(x)
4. other values W(x,1) • R(x)

1 ⊕ 2 ⊕ 3 ⊕ 4 R(y) • R(x) • W(x,1) • R(x)

C. compute *φ* R(x) ⟶ 1 V R(x) ⟶ 0

Computing scheduling directives: *backward analysis*





A. choose a state $s[\langle\rangle]$

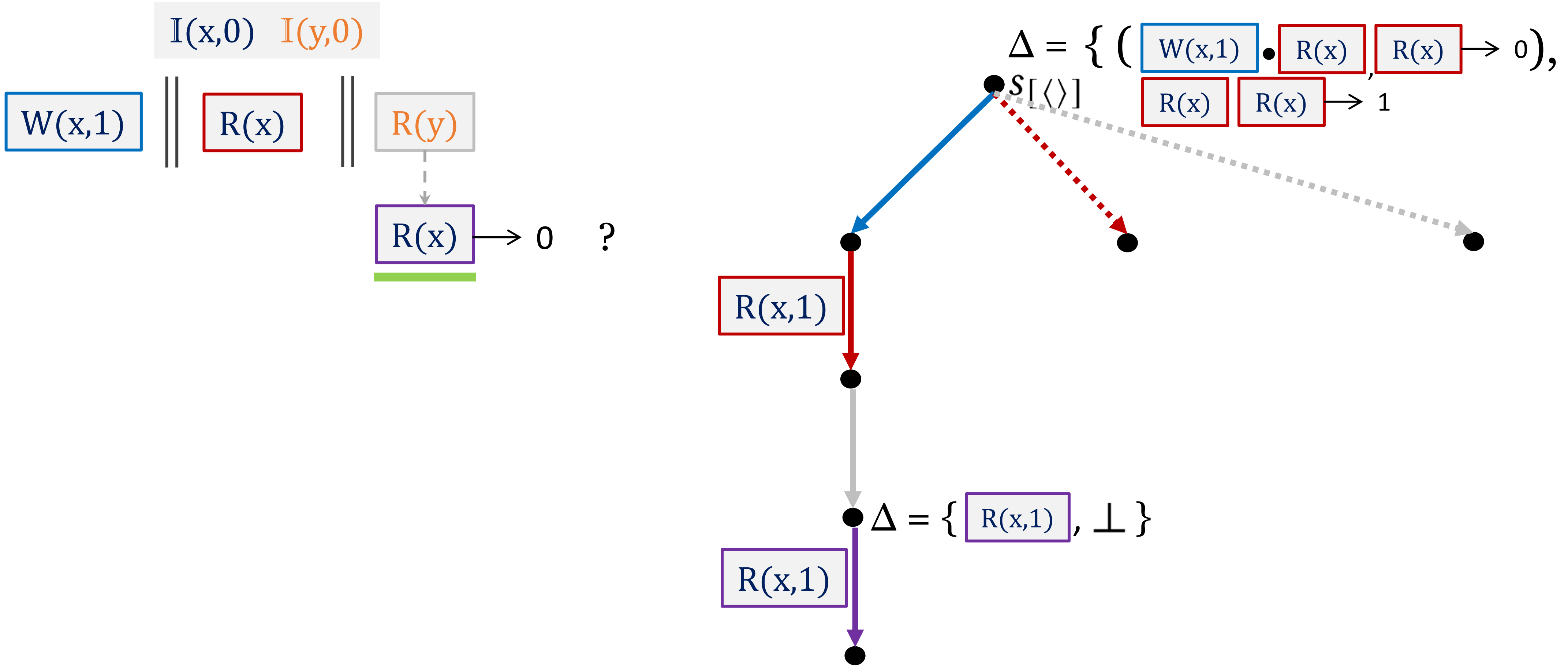
B. compute *next*

1. enable write $\langle\rangle$
2. enable read R(y) • R(x)
3. necessary *rf* $\langle\rangle$ • R(x)
4. other values W(x,1) • R(x)

1 ⊕ 2 ⊕ 3 ⊕ 4 R(y) • R(x) • W(x,1) • R(x)

C. compute *φ* R(x) ⟶ 1 V R(x) ⟶ 0

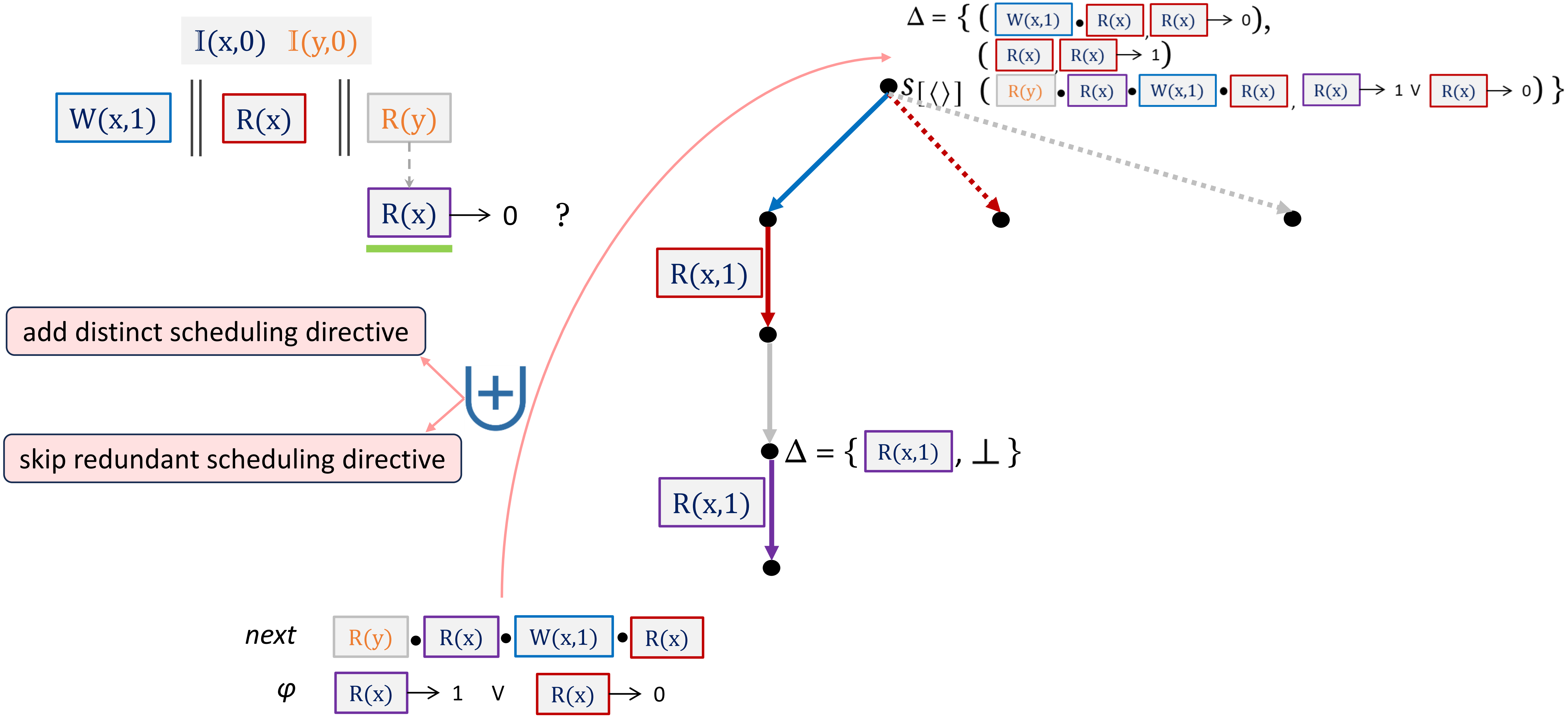
Computing scheduling directives: *backward analysis*



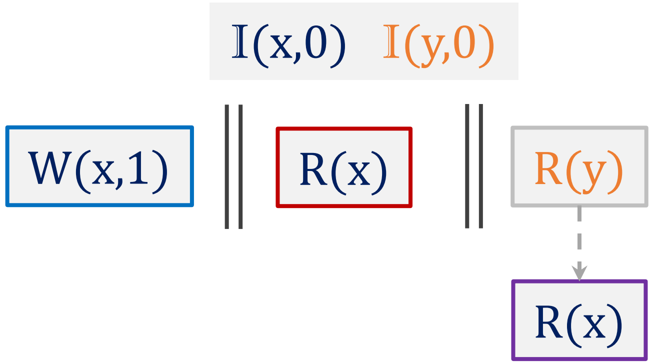
next R(y) • R(x) • W(x,1) • R(x)

φ R(x) → 1 V R(x) → 0

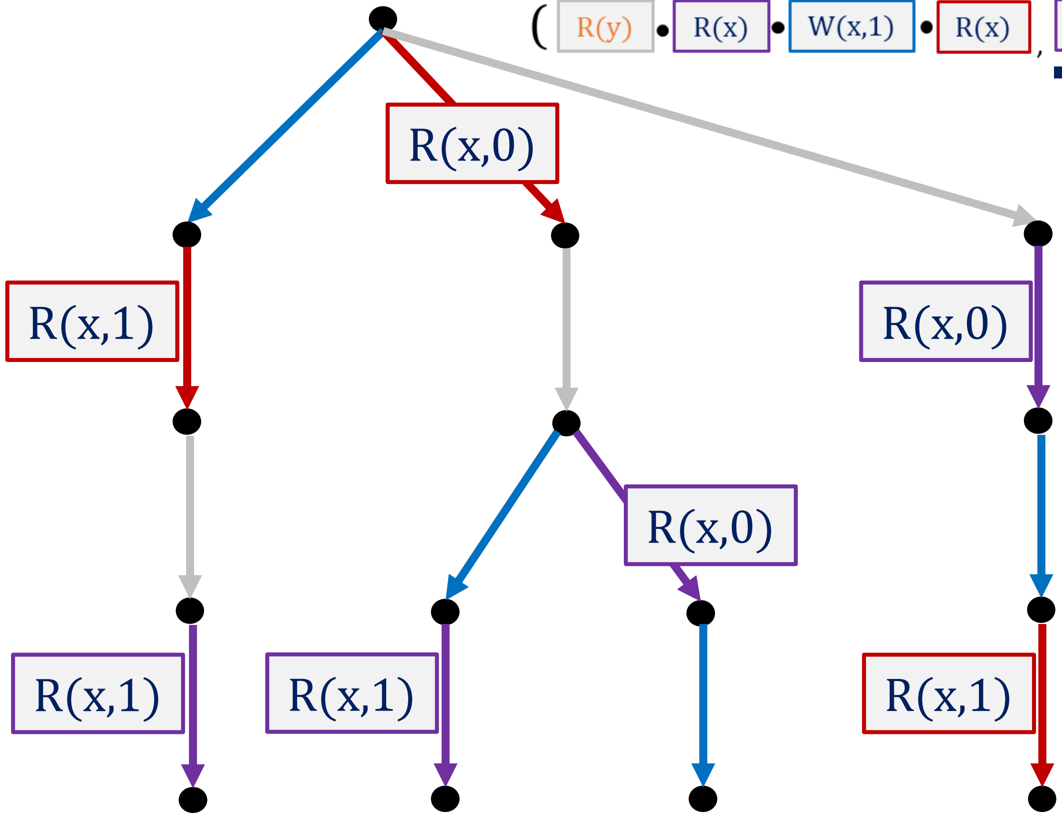
Computing scheduling directives: *backward analysis*



Computing scheduling directives



$$\Delta = \{ (\boxed{W(x,1)} \bullet \boxed{R(x)}, \boxed{R(x)} \mapsto 0), (\boxed{R(x)}, \boxed{R(x)} \mapsto 1), (\boxed{R(y)} \bullet \boxed{R(x)} \bullet \boxed{W(x,1)} \bullet \boxed{R(x)}, \underline{\boxed{R(x)} \mapsto 1 \vee \boxed{R(x)} \mapsto 0}) \}$$



$\boxed{R(x)} \mapsto 1 \Vdash \boxed{R(x)} \mapsto 1 \vee \boxed{R(x)} \mapsto 0$

$\boxed{R(x)} \mapsto 0 \Vdash \boxed{R(x)} \mapsto 1 \vee \boxed{R(x)} \mapsto 0$

ViEqui *is*

$\mathbb{E}$ explorations by ViEqui
Π view-equivalence classes
$[\![\tau]\!]$ view-equivalence class of $\tau \in \mathbb{E}$

- Complete: *each maximal sequence represents an equivalence class*

$$\forall \tau \in \mathbb{E}, \exists \pi \in \Pi \text{ s.t. } [\![\tau]\!] = \pi$$

ensured by:

runtime analysis and $\oplus$

ViEqui *is*

$\mathbb{E}$ explorations by ViEqui
Π view-equivalence classes
$[\![\tau]\!]$ view-equivalence class of $\tau \in \mathbb{E}$

- Complete: *each maximal sequence represents an equivalence class*

$$\forall \tau \in \mathbb{E}, \exists \pi \in \Pi \ \text{ s.t. } [\![\tau]\!] = \pi$$

- Sound: *each equivalence class is explored*

$$\forall \pi \in \Pi, \exists \tau \in \mathbb{E} \ \text{ s.t. } [\![\tau]\!] = \pi$$

ensured by:

analyzing all *write events* and $\uplus$

ViEqui *is*

$\mathbb{E}$ explorations by ViEqui
Π view-equivalence classes
$[\![\tau]\!]$ view-equivalence class of $\tau \in \mathbb{E}$

- Complete: *each maximal sequence represents an equivalence class*

$$\forall \tau \in \mathbb{E}, \exists \pi \in \Pi \ \text{ s.t. } [\![\tau]\!] = \pi$$

- Sound: *each equivalence class is explored*

$$\forall \pi \in \Pi, \exists \tau \in \mathbb{E} \ \text{ s.t. } [\![\tau]\!] = \pi$$

- Optimal: *each equivalence class is explored exactly once*

$$\nexists \tau_1, \tau_2 \in E, \text{where } \tau_1 \neq \tau_2, \ \text{ s.t. } [\![\tau_1]\!] = [\![\tau_2]\!]$$

ensured by:

non-redundant scheduling directives using $\uplus$

Time complexity

$$\mathcal{O}(|\mathcal{V}|^{|\mathcal{E}^{\mathbb{R}}|}.|\mathcal{E}^{\mathbb{R}}|.(\mathscr{F} + \mathscr{B}))$$

backward-analysis

forward-analysis

for each read event

for each explored sequence

where,

$$\mathscr{F} = \mathcal{O}(|\mathcal{T}|^2 + \log(|\mathcal{V}|^{|\mathcal{E}^{\mathbb{R}}|}))$$

$$\mathscr{B} = \mathcal{O}(|\mathcal{E}^{\mathbb{W}}|.(|\tau|^3 + |\mathcal{V}|^{|\mathcal{E}^{\mathbb{R}}|}.|\tau|))$$

Space complexity

$$\mathcal{O}(|\mathcal{E}|^2.|\mathcal{V}|^{|\mathcal{E}^{\mathbb{R}}|})$$

view-equivalence classes

events

existing DPOR algorithms are either exploration-optimal but may use exponential in the size of the program[1], or maintain polynomial memory consumption but potentially explore exponentially many redundant interleavings[2]." [Kokologiannakis et al. POPL '22]

[1] [Abdulla et al. 2014; Albert et al. 2019; Aronis et al. 2018; Kokologiannakis et al. 2017]

[2] [Abdulla et al. 2017; Godefroid 2005]."

ViEqui tool

Implemented over Nidhugg.

available at: https://github.com/nidhugg/nidhugg

Tested over 16,154 litmus tests of concurrent C programs borrowed from [Abdulla et al., OOPSLA '18]

Performance analysis

ODPOR → classical equivalence [Abdulla et al., POPL '14]
rfsc → reads-from equivalence [Abdulla et al., OOPSLA '19]
RVF-SMC → reads-value-from equivalence [Agarwal et al., CAV '21]

	ODPOR		rfsc		RVF-SMC		ViEqui	
benchmark	#Seq	Time	#Seq	Time	#Seq	Time	#Seq	Time
monabsex(5)	14400	2.56	1296	3.56	6	0.01	1	0.02
monabsex(100)	-	To	-	To	101	0.99	1	0.09
monabsex(500)	-	To	-	To	501	162.84	1	3.00
redundant-co(8)	1969110	338.84	217	0.12	11	0.02	7	0.02
redundant-co(10)	-	To	331	0.32	11	0.01	7	0.02
redundant-co(50)	-	To	7651	2.11	11	0.02	7	0.03
redundant-co(1000)	-	To	-	To	11	0.11	7	3.79
IBM-incdec(50)	-	To	-	To	-	To	3	9.03
IBM-incdec(100)	-	To	-	To	-	To	3	38.46

Performance analysis

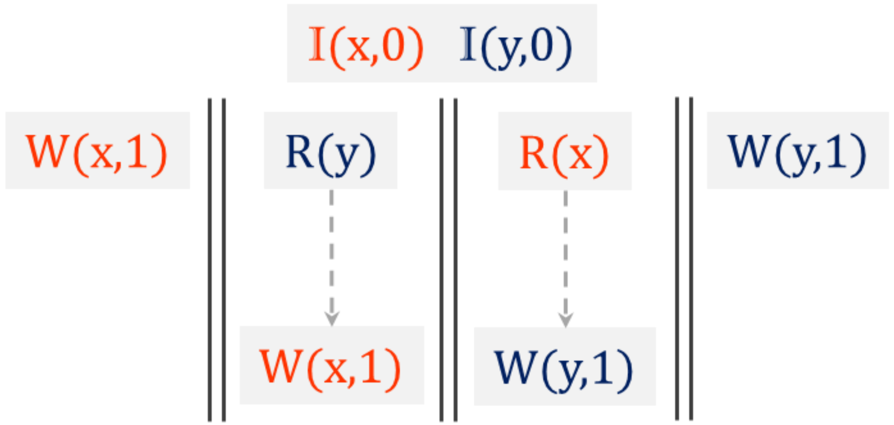
benchmark	ODPOR #Seq	ODPOR Time	rfsc #Seq	rfsc Time	RVF-SMC #Seq	RVF-SMC Time	ViEqui #Seq	ViEqui Time
monabsex(5)	14400	2.56	1296	3.56	6	0.01	1	0.02
monabsex(100)	-	To	-	To	101	0.99	1	0.09
monabsex(500)	-	To	-	To	501	162.84	1	3.00
redundant-co(8)	1969110	338.84	217	0.12	11	0.02	7	0.02
redundant-co(10)	-	To	331	0.32	11	0.01	7	0.02
redundant-co(50)	-	To	7651	2.11	11	0.02	7	0.03
redundant-co(1000)	-	To	-	To	11	0.11	7	3.79
IBM-incdec(50)	-	To	-	To	-	To	3	9.03
IBM-incdec(100)	-	To	-	To	-	To	3	38.46

Performance analysis

benchmark	ODPOR #Seq	ODPOR Time	rfsc #Seq	rfsc Time	RVF-SMC #Seq	RVF-SMC Time	ViEqui #Seq	ViEqui Time
burns(5)	2353602	1046.92	2353602	582.46	17382	5.14	36	0.05
burns(40)	-	To	-	To	-	To	1681	188.63
burns(60)	-	To	-	To	-	To	3721	1589.24
dekker-simple(10)	739021	420.96	739021	264.73	2713870	704.97	21	0.03
dekker-simple(100)	-	To	-	To	-	To	201	42.99
dekker-simple(150)	-	To	-	To	-	To	301	302.43
dekker-simple(200)	-	To	-	To	-	To	401	1331.79
peterson(5)	2782162	1432.44	2709013	769.10	-	To	31	0.04
peterson(50)	-	To	-	To	-	To	301	20.30
peterson(100)	-	To	-	To	-	To	601	495.38
peterson(120)	-	To	-	To	-	To	721	1230.92
szymanski(4)	396583	198.87	396583	108.56	1444246	319.78	5335	5.08
szymanski(5)	-	To	-	To	-	To	19349	26.86
szymanski(7)	-	To	-	To	-	To	264209	678.89

Performance analysis

benchmark	ODPOR #Seq	ODPOR Time	rfsc #Seq	rfsc Time	RVF-SMC #Seq	RVF-SMC Time	ViEqui #Seq	ViEqui Time
swsc-co1(20)	-	To	8040	4.48	8060	14.14	7240	4.91
swsc-co1(50)	-	To	125100	228.18	125150	1705.93	120100	327.25
swsc-co1(60)	-	To	216120	518.64	-	To	208920	771.69
swsc-co10(10)	-	To	10	0.12	11	0.02	10	0.02
swsc-co10(100)	-	To	100	0.60	101	7.46	100	0.62
swsc-co10(250)	-	To	250	7.03	251	278.73	250	7.24



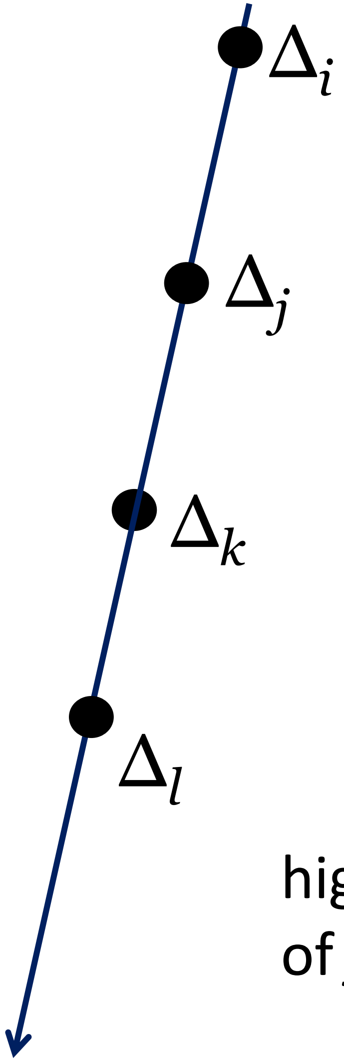
#loops	ODPOR #Seq	ODPOR Time	rfsc #Seq	rfsc Time	RVF-SMC #Seq	RVF-SMC Time	ViEqui #Seq	ViEqui Time
10	-	To	-	To	3703196	705.69	40	0.06
20	-	To	-	To	-	To	80	0.28

Performance analysis

	ODPOR		rfsc		RVF-SMC		ViEqui	
benchmark	#Seq	Time	#Seq	Time	#Seq	Time	#Seq	Time
pgsql(5,5)	781	0.72	781	0.45	19900	3.06	781	0.73
pgsql(6,7)	55987	68.57	55987	30.72	2292077	654.66	55987	192.92
pgsql(7,7)	137257	171.45	137257	76.59	5356580	1620.66	137257	943.17
unverif(5,5)	14400	2.74	14400	1.78	68890	11.70	14400	198.57
unverif(5,10)	14400	2.98	14400	1.91	70890	12.76	14400	200.92
unverif(6,5)	518400	110.60	518400	68.04	2625944	699.47	-	To

Performance analysis

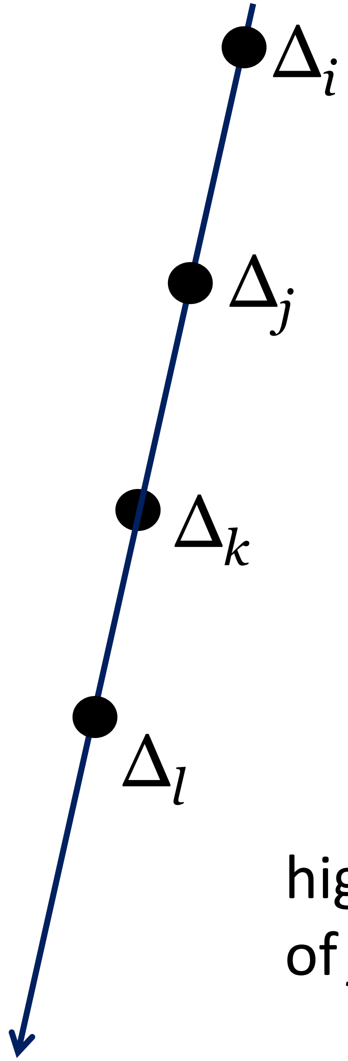
	ViEqui		Assert
benchmark	#Seq	Time	violation
tas(20,50)	3	49.11	Yes
tas(30,50)	3	108.29	Yes
tas(40,50)	3	197.33	Yes



highlights weakness of *forward-analysis*

Performance analysis

benchmark	ViEqui #Seq	ViEqui Time	Assert violation
tas(20,50)	3	49.11	Yes
tas(30,50)	3	108.29	Yes
tas(40,50)	3	197.33	Yes



benchmark	ODPOR #Seq	ODPOR Time	rfsc #Seq	rfsc Time	RVF-SMC #Seq	RVF-SMC Time
tas(20,50)	-	To	-	To	23	0.08
tas(30,50)	-	To	-	To	33	0.15
tas(40,50)	-	To	-	To	43	0.26

highlights weakness of *forward-analysis*

Performance analysis

	ViEqui		Assert
benchmark	#Seq	Time	violation
triangular-2(5)	1558	0.97	Yes
triangular-2(7)	31522	442.58	Yes
triangular-2(8)	-	To	Yes

large enabling sequences

no well-formed join

highlights weakness
of *backward-analysis*

Performance analysis

benchmark	ViEqui #Seq	ViEqui Time	Assert violation
triangular-2(5)	1558	0.97	Yes
triangular-2(7)	31522	442.58	Yes
triangular-2(8)	-	To	Yes

large enabling sequences

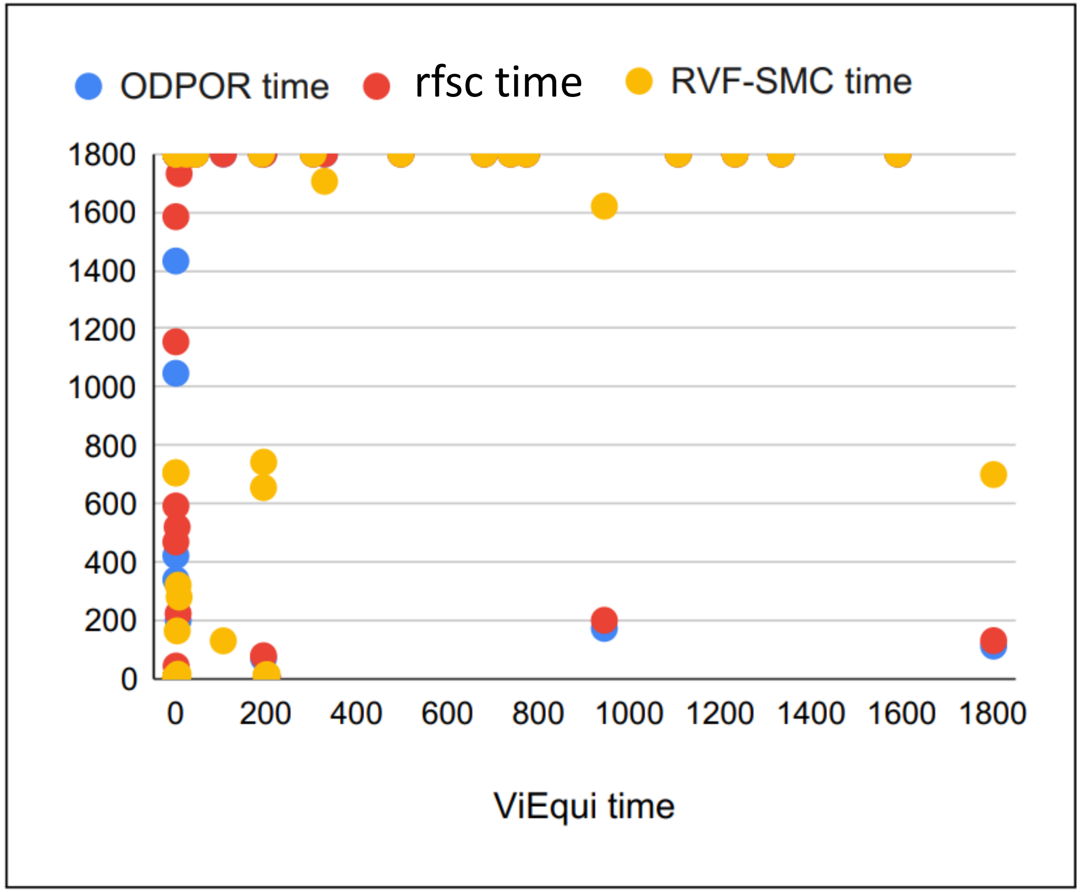
no well-formed join

benchmark	ODPOR #Seq	ODPOR Time	rfsc #Seq	rfsc Time	RVF-SMC #Seq	RVF-SMC Time
triangular-2(5)	20172	2.69	-	✗	26272	2.41
triangular-2(7)	1695856	266.81	-	✗	644193	70.10
triangular-2(8)	-	To	-	✗	3045756	369.60

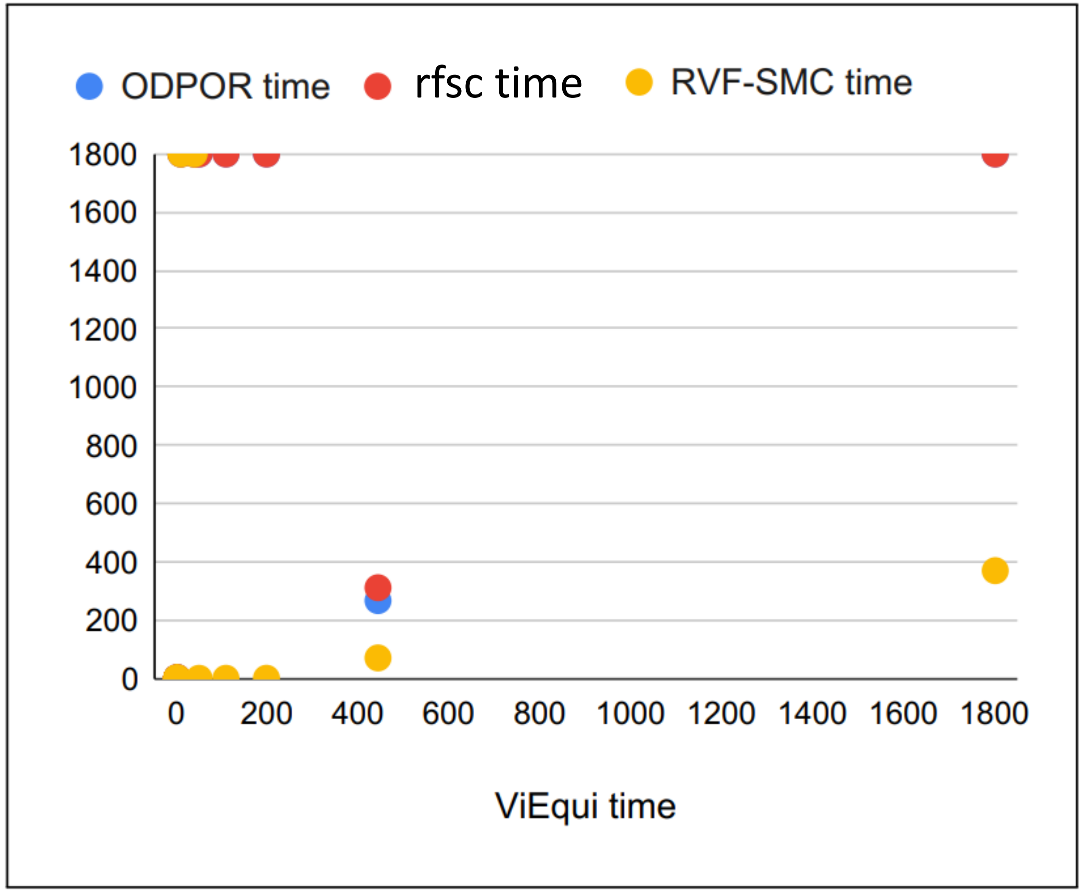
✗: Failed to run

highlights weakness
of *backward-analysis*

Performance analysis



benchmarks with no assert violation



benchmarks with assert violation

Future Scope

- view-equivalence based SMC for *weak memory models*
- *coarsening* by considering the assert condition in the equivalence realtion
- applicability for database *transactions*
- Richer constructs like *coarse grained synchronization*

Future Scope

- *scalability*

benchmark	ODPOR #Seq	ODPOR Time	rfsc #Seq	rfsc Time	RVF-SMC #Seq	RVF-SMC Time	ViEqui #Seq	ViEqui Time
FreeBSD-abd-kbd	1	0.03	1	0.12	1	0.02	1	0.02
FreeBSD-rdma-addr	1	0.02	1	0.12	1	0.01	1	0.02
NetBSD-sysmon-power	4	0.03	26	0.15	6	0.02	6	0.04
Solaris-space-map	2	0.03	2	0.12	1	0.02	1	0.01

Future Scope

- *scalability*

benchmark	ODPOR #Seq	ODPOR Time	rfsc #Seq	rfsc Time	RVF-SMC #Seq	RVF-SMC Time	ViEqui #Seq	ViEqui Time
FreeBSD-abd-kbd	1	0.03	1	0.12	1	0.02	1	0.02
FreeBSD-rdma-addr	1	0.02	1	0.12	1	0.01	1	0.02
NetBSD-sysmon-power	4	0.03	26	0.15	6	0.02	6	0.04
Solaris-space-map	2	0.03	2	0.12	1	0.02	1	0.01
Safestack		oom		oom		To		To

oom: out of memory

Thank You

Questions?

Computing scheduling directives: *backward analysis*

I(x,0) I(y,0)

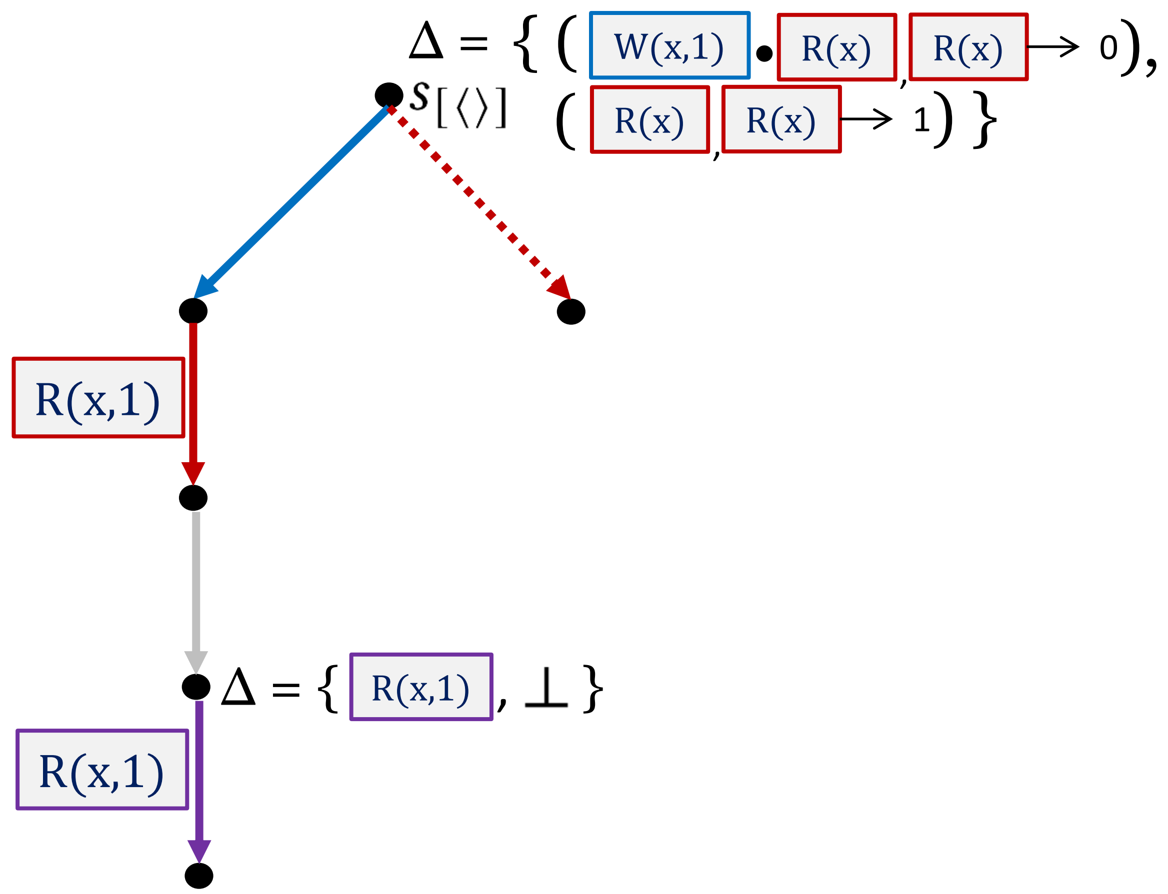
W(x,1) ‖ R(x) ‖ R(y)

R(x) → 0 ?

A. choose a state $s[\langle\rangle]$

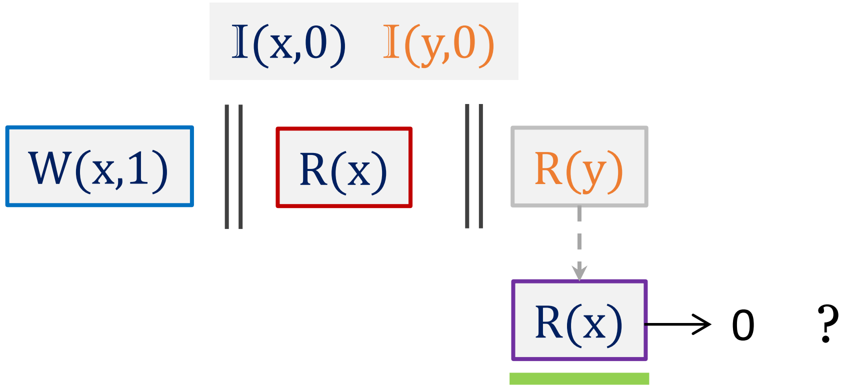
B. compute *next*

1. enable write $\langle\rangle$
2. enable read R(y) • R(x)
3. necessary *rf* $\langle\rangle$ • R(x)
4. other values W(x,1) • R(x)



back

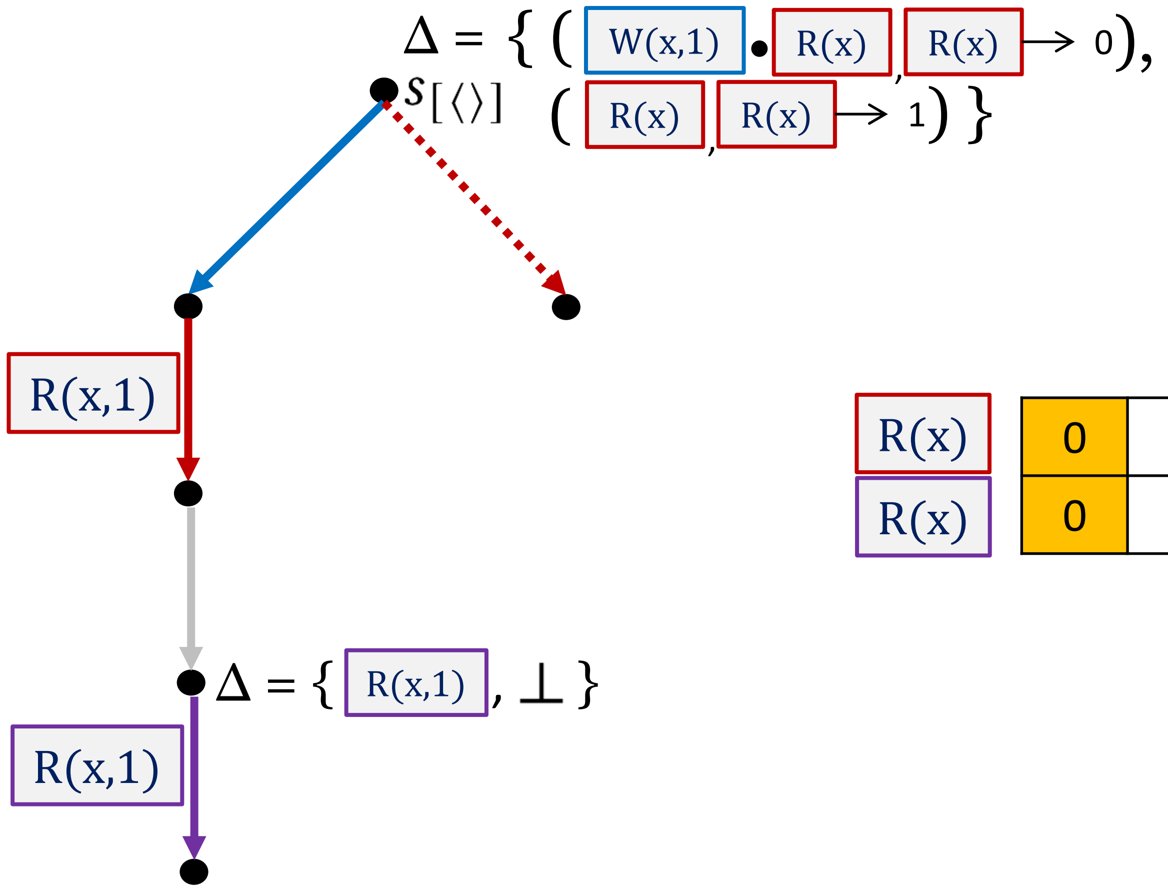
Computing scheduling directives: *backward analysis*



A. choose a state $s[\langle\rangle]$

B. compute *next*

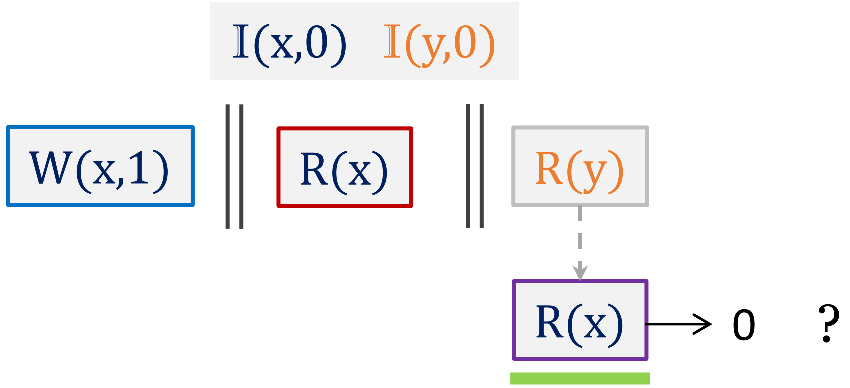
1. enable write $\langle\rangle$
2. enable read R(y) • R(x)
3. necessary *rf* $\langle\rangle$ • R(x)
4. other values W(x,1) • R(x)



R(x)	0	0	1	1
R(x)	0	1	0	1

back

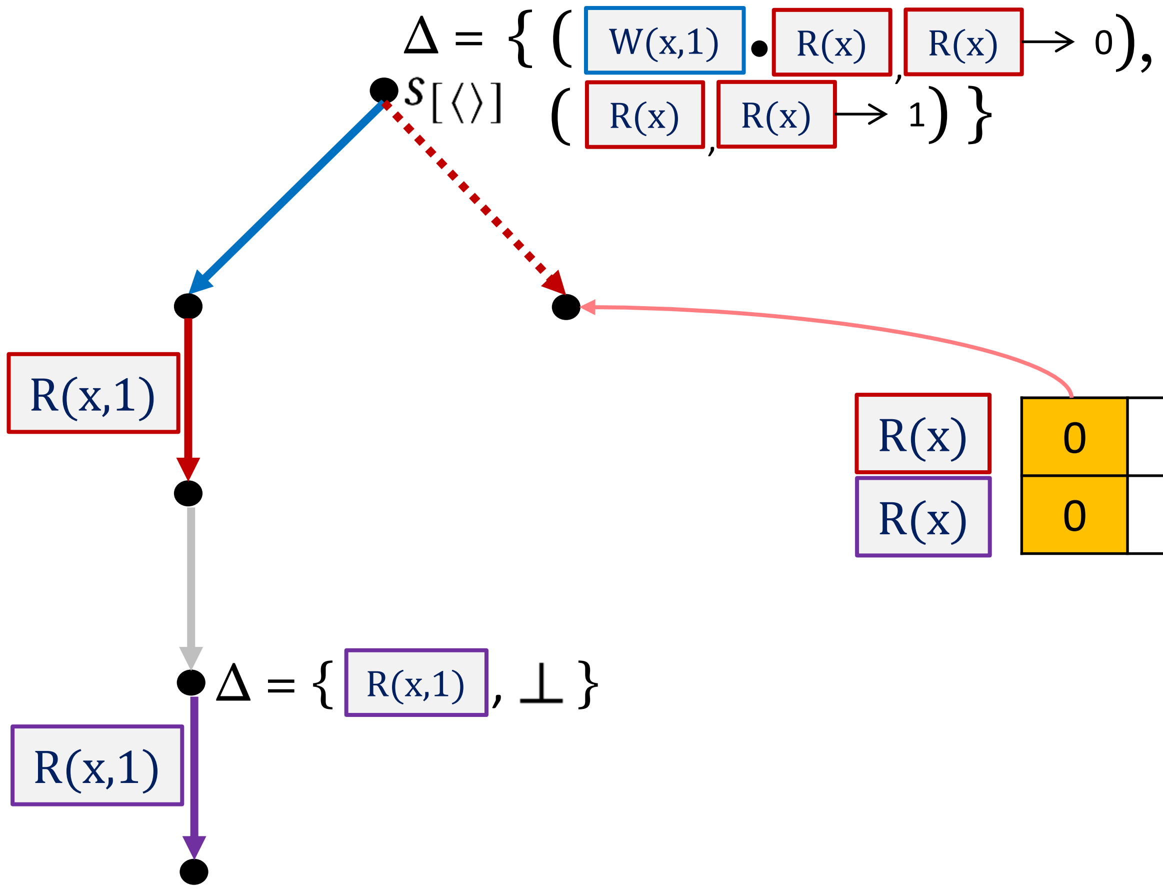
Computing scheduling directives: *backward analysis*



A. choose a state $s[\langle\rangle]$

B. compute *next*

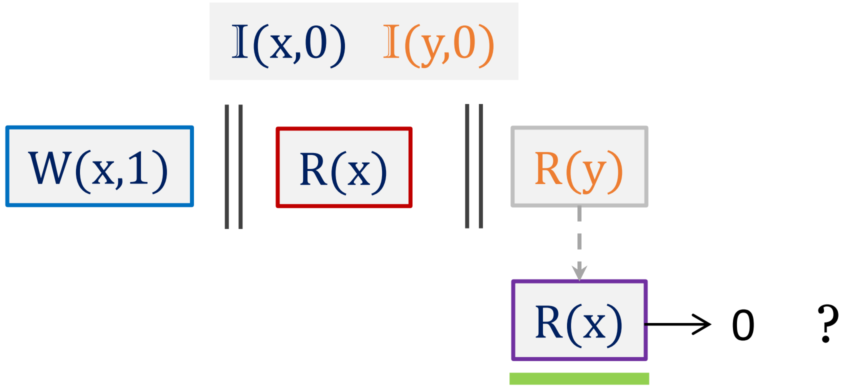
1. enable write $\langle\rangle$
2. enable read R(y) • R(x)
3. necessary *rf* $\langle\rangle$ • R(x)
4. other values W(x,1) • R(x)



R(x)	0	0	1	1
R(x)	0	1	0	1

back

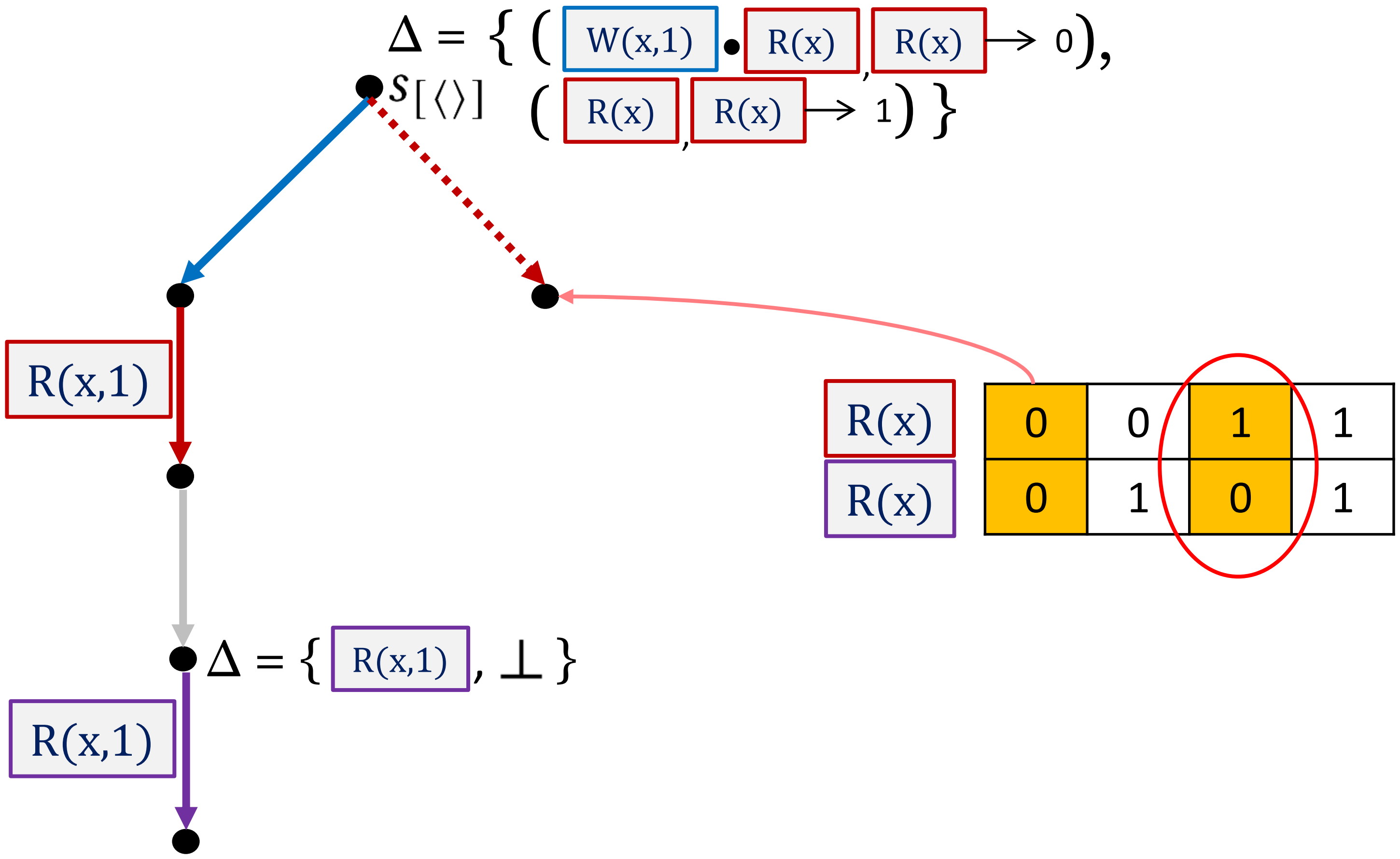
Computing scheduling directives: *backward analysis*



A. choose a state $s[\langle\rangle]$

B. compute *next*

1. enable write $\langle\rangle$
2. enable read R(y) • R(x)
3. necessary *rf* $\langle\rangle$ • R(x)
4. other values W(x,1) • R(x)



back

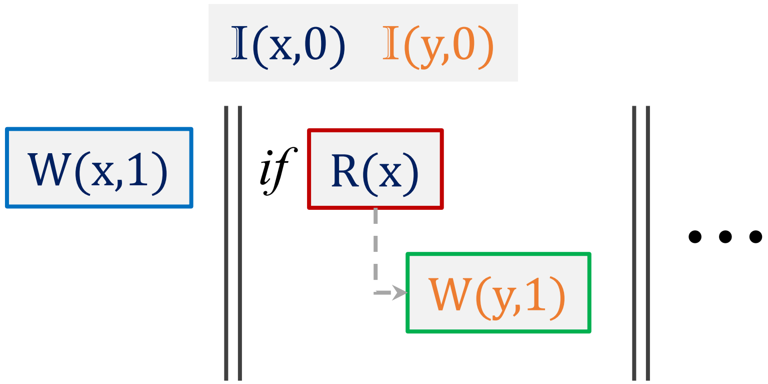
Handling causality

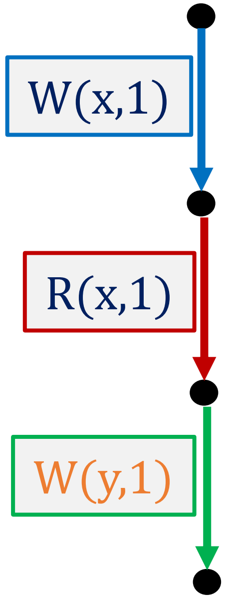
I(x,0) I(y,0)

W(x,1) || *if* R(x) W(y,1) || …

back

Handling causality





1. enabling sequence ($\boxed{W(y,1)}$) = $\boxed{W(x,1)} \bullet \boxed{R(x,1)} \bullet \boxed{W(y,1)}$

2. well-formed join ($\oplus$) preserves $(po \cup rf)^+$

back

View-equivalence of DBMS transactions

2 schedules $S1$, $S2$ of transactions are equivalent if in $S1$ and $S2$:

1. Each reading transaction reads the same value

2. Each reading transaction reads from the same transaction

3. Final write is performed by the same transaction

back