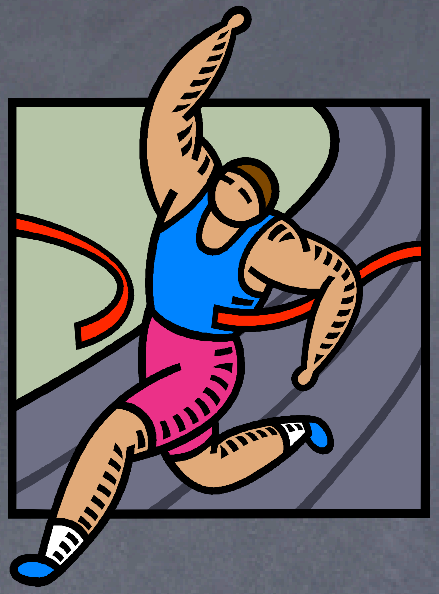
Looking good, behaving well

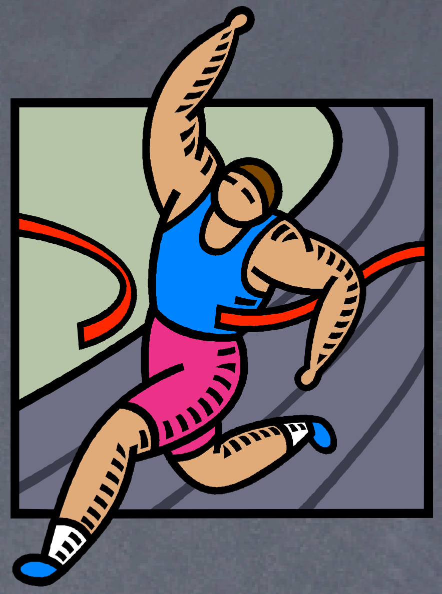
Michael Westergaard
Department of Computer Science
University of Aarhus

Example (1/4)



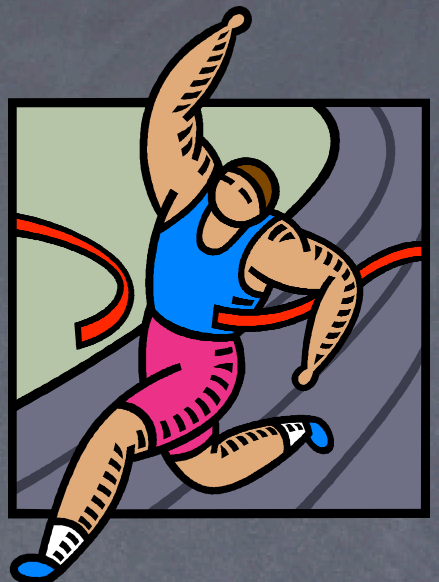
- 2 runners in a race, halfway through the race is a stand with water
- Either
  - run: a runner runs to the drink stand,
  - win: a runner wins the race, or
  - lose: a runner loses the race

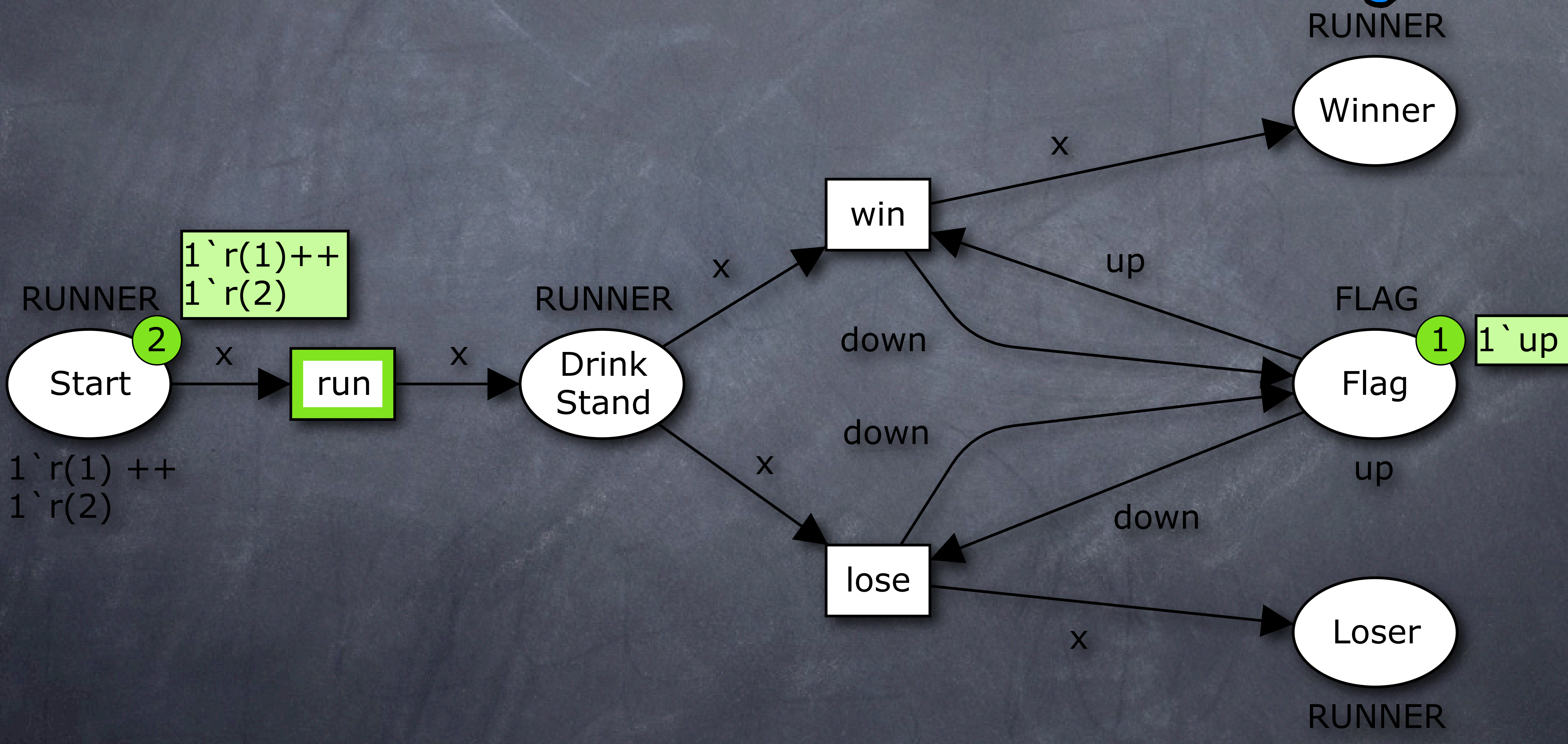
Example (2/4)



- Only one runner can win the race
- In the beginning neither of the runners have finished any laps
- We can model this using Timed Automata or Coloured Petri nets

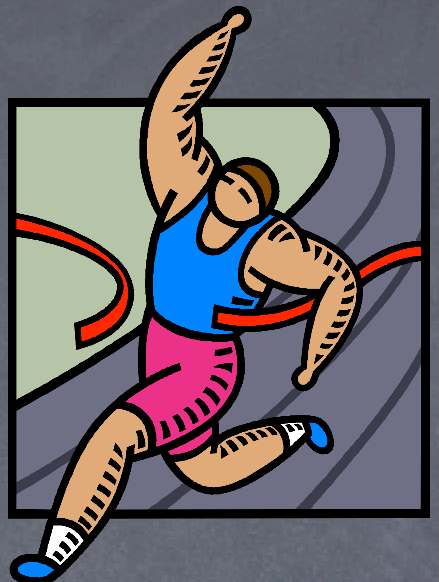
Example (3/4)

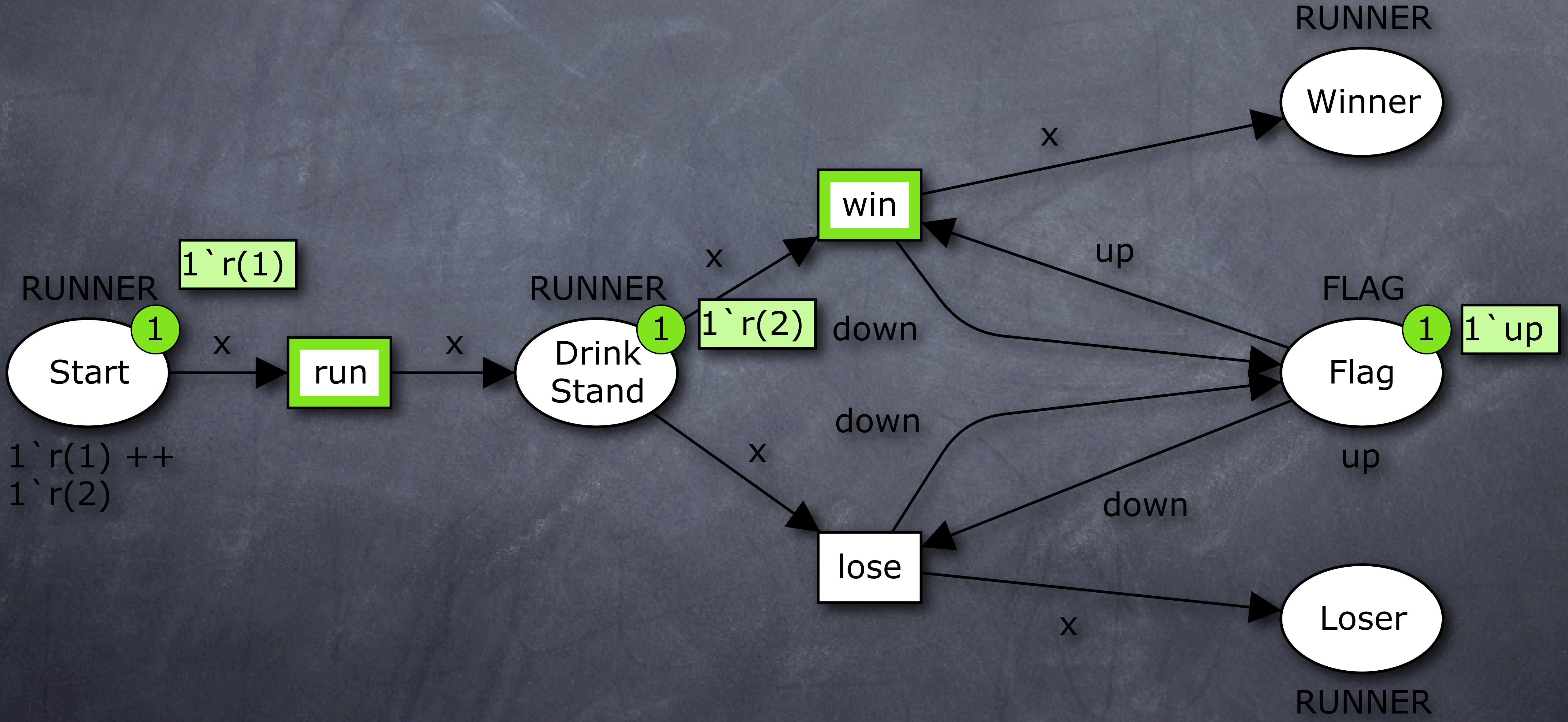




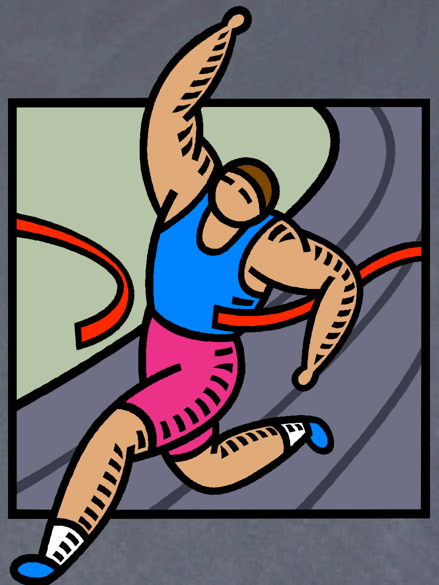
RUNNER
Start
2
1`r(1) ++
1`r(2)
1`r(1)++
1`r(2)
x
run
x
RUNNER
Drink
Stand
x
win
x
lose
x
RUNNER
Winner
x
RUNNER
Loser
up
down
down
up
down
FLAG
Flag
1
1`up

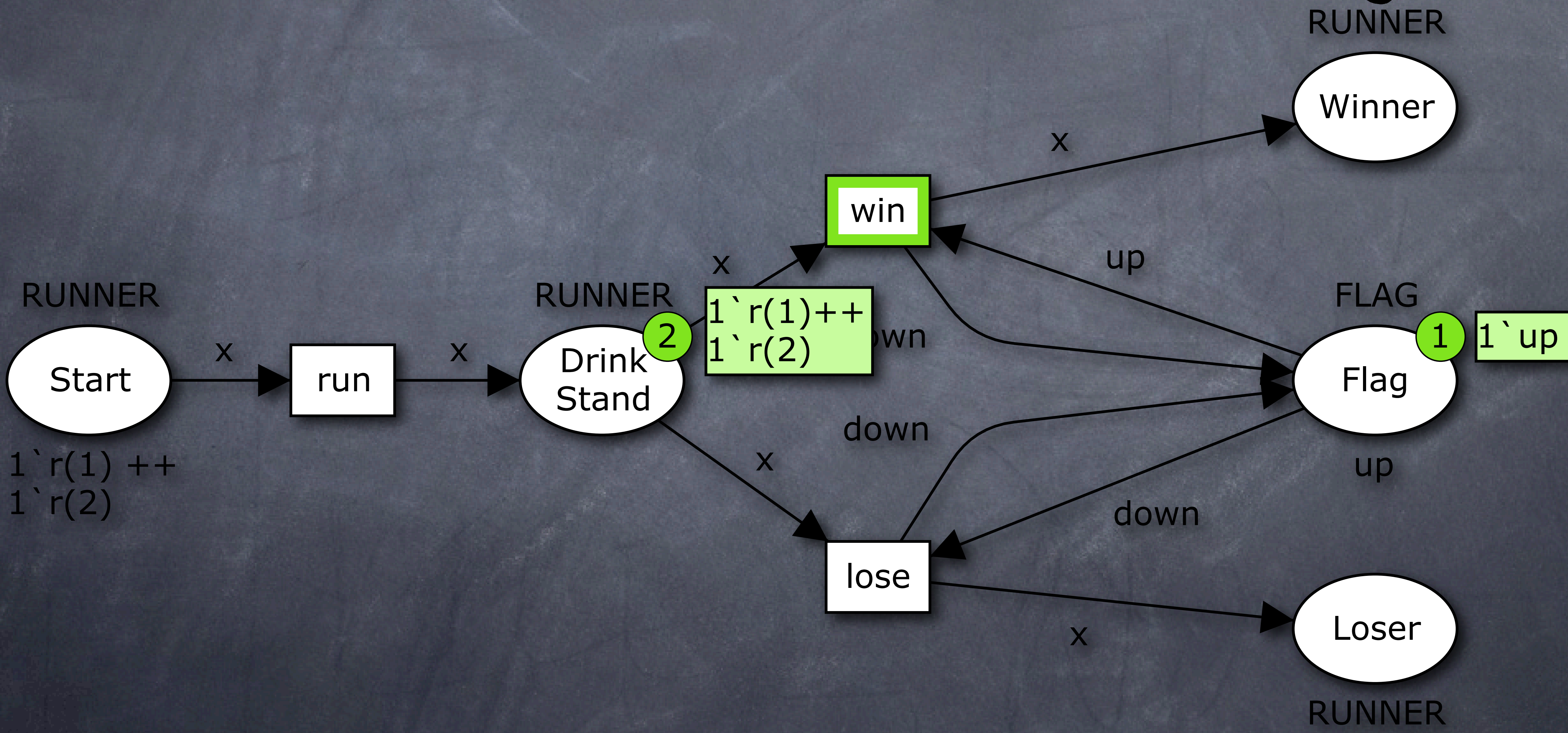
Example (3/4)



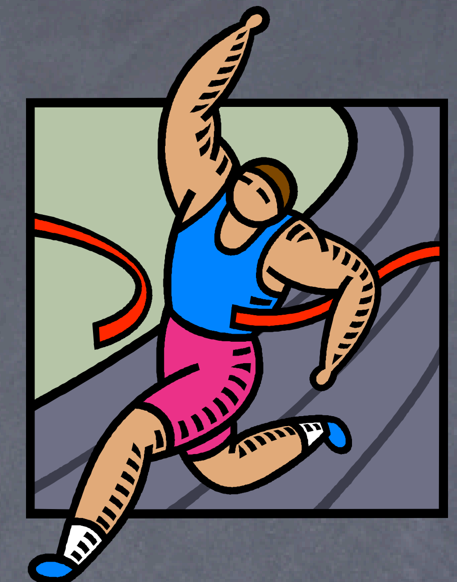

RUNNER
Winner
x
win
up
x
1`r(1)
RUNNER
RUNNER
1`r(2)
down
FLAG
1
1
1
1`up
Start
x
run
x
Drink
Stand
Flag
1`r(1) ++
1`r(2)
down
x
up
down
lose
x
Loser
RUNNER

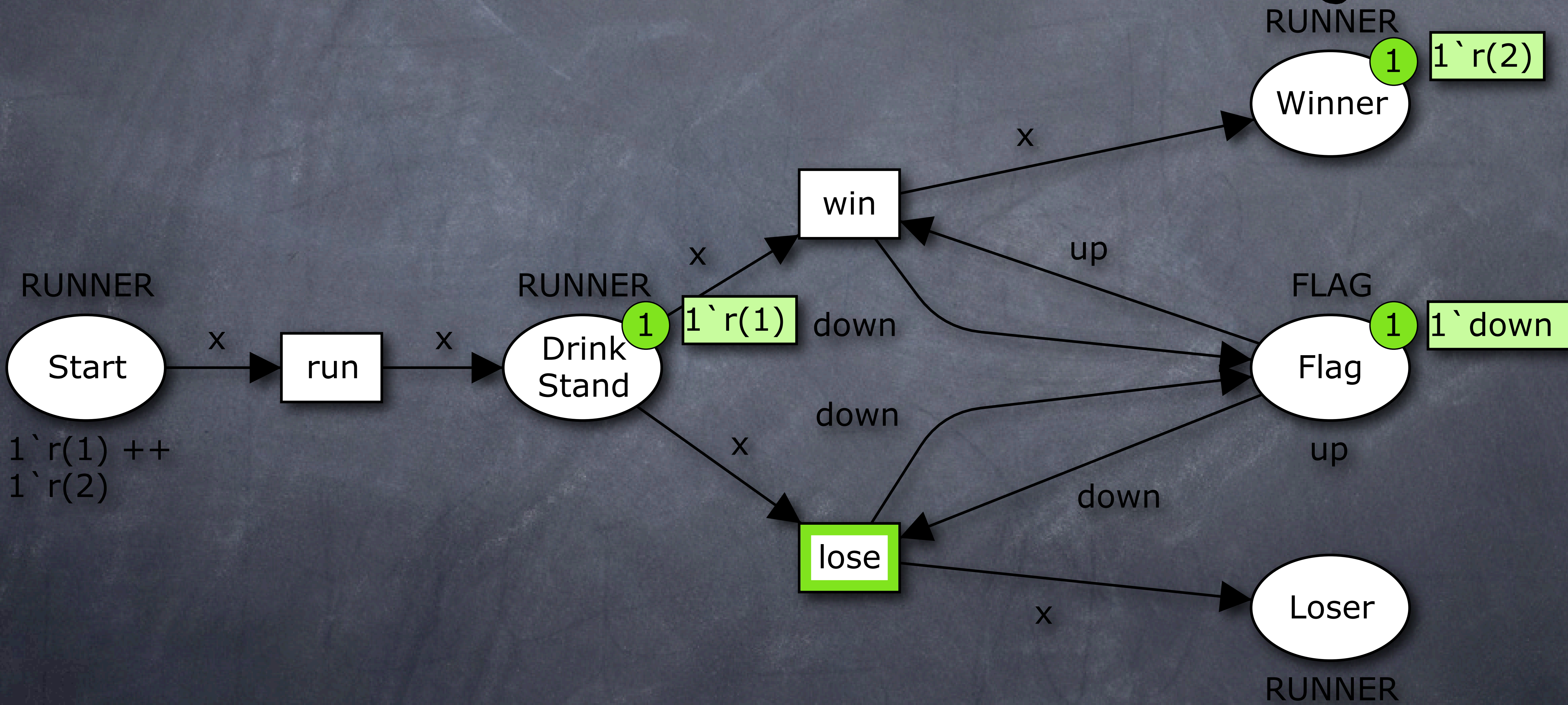
Example (3/4)




RUNNER
Winner
x
win
x
up
RUNNER
Start
x
run
x
RUNNER
Drink Stand
2
1`r(1)++
1`r(2)
FLAG
Flag
1
1`up
down
x
up
down
lose
x
Loser
RUNNER
1`r(1) ++
1`r(2)

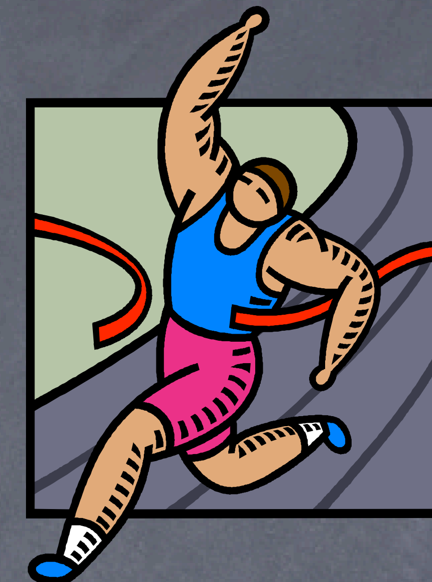
Example (3/4)

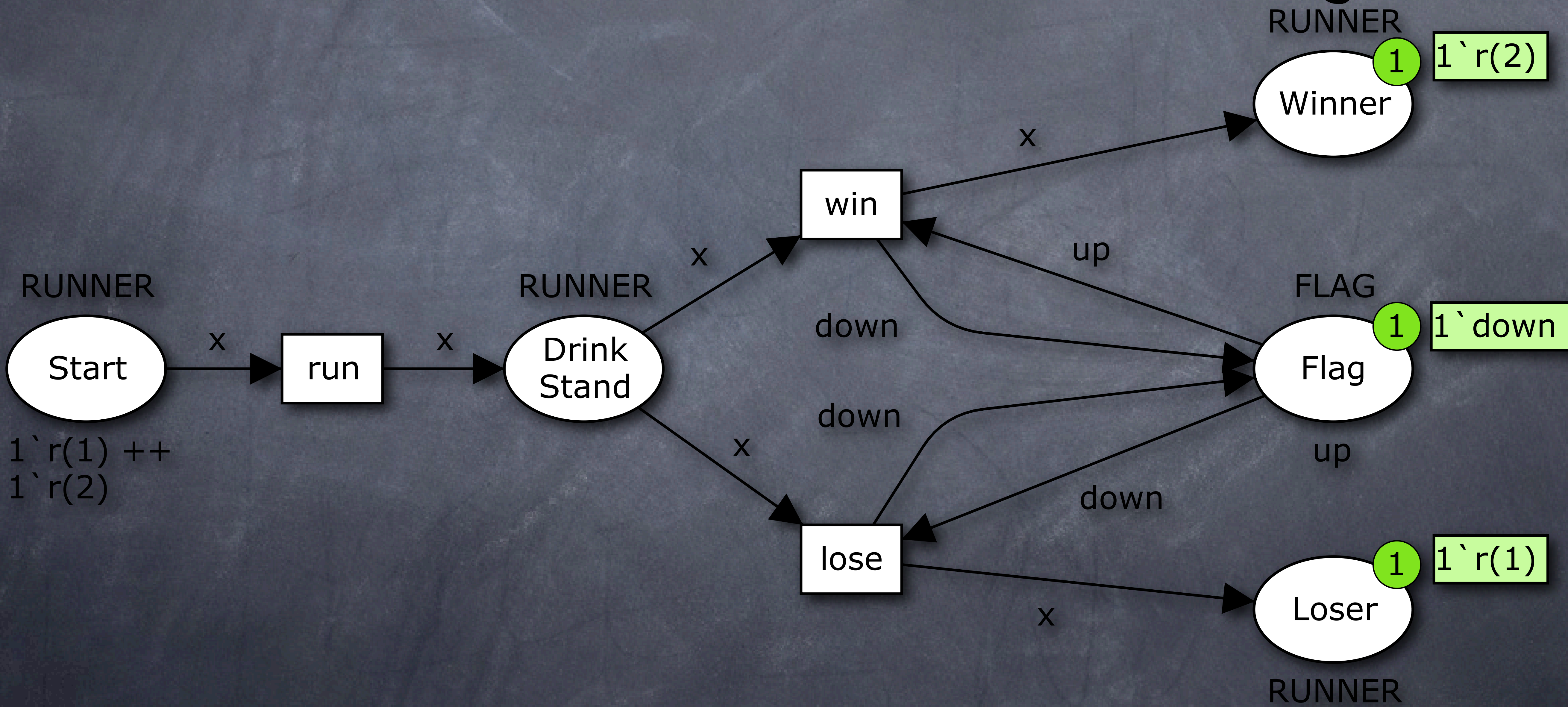




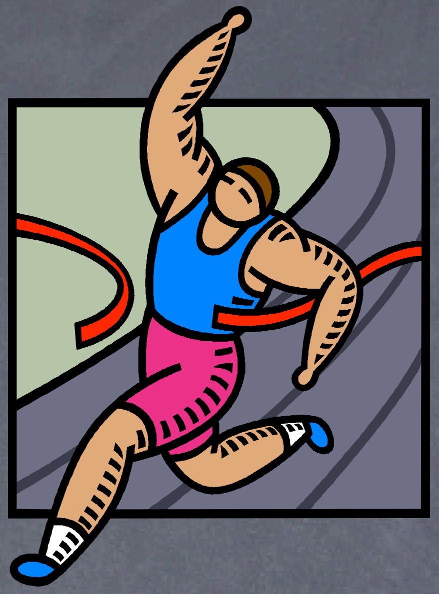
RUNNER
Winner
1
1`r(2)
x
win
up
x
RUNNER
FLAG
RUNNER
1
1`r(1)
down
1
1`down
Start
x
run
x
Drink Stand
Flag
1`r(1) ++
1`r(2)
down
x
up
down
lose
Loser
x
RUNNER

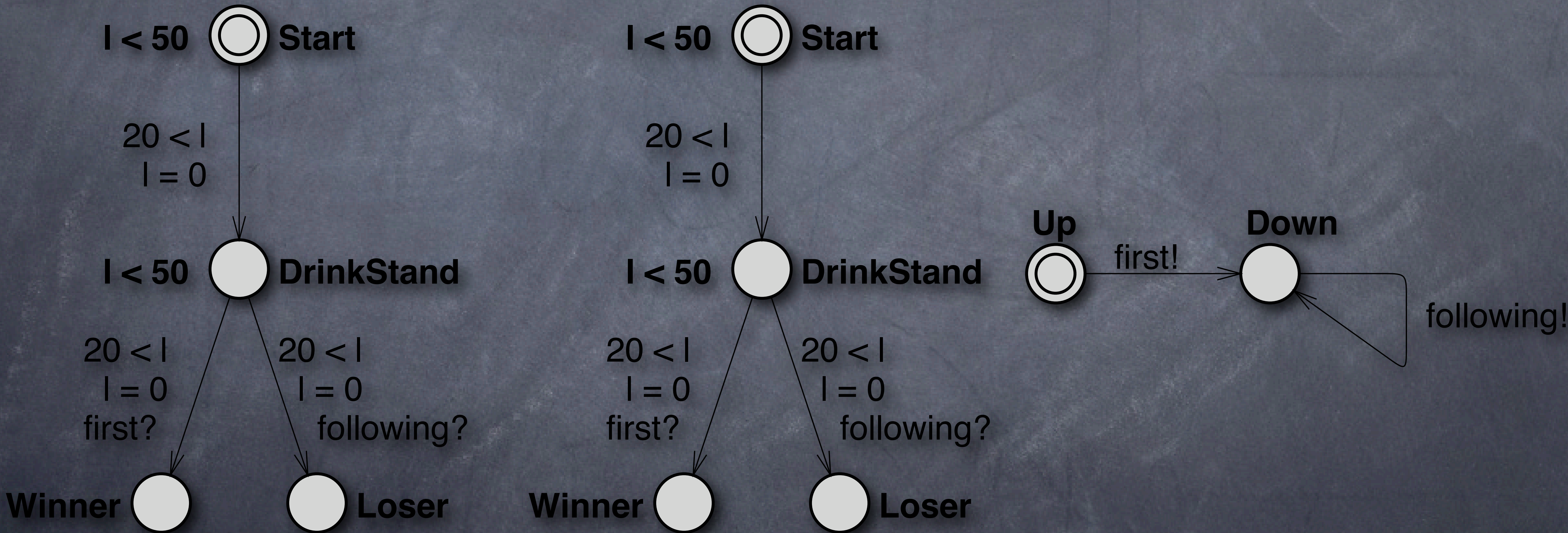
Example (3/4)




RUNNER
Start
1`r(1) ++
1`r(2)
x
run
x
RUNNER
Drink
Stand
x
x
win
lose
down
down
up
down
up
x
x
RUNNER
Winner
1
1`r(2)
FLAG
Flag
1
1`down
Loser
1
1`r(1)
RUNNER

Example (4/4)





l < 50
Start
20 < l
l = 0
l < 50
DrinkStand
20 < l
l = 0
first?
20 < l
l = 0
following?
Winner
Loser
l < 50
Start
20 < l
l = 0
l < 50
DrinkStand
20 < l
l = 0
first?
20 < l
l = 0
following?
Winner
Loser
Up
first!
Down
following!

Motivation

We want the model to

- look good (even to people not familiar with the modeling formalism)
- behave well (e.g. ensure only one runner can win the race)

Outline

- The BRITNeY animation tool
- A state space tool
  - Memory-efficient state storage using the sweep-line method
  - Memory-efficient state storage using hash compaction and backtracking

Looking Good

The BRITNeY animation tool

Motivation

Motivation

Motivation

Domain expert

Figure 2 shows the approach taken to use CPN models to develop a prototype of the interoperability protocol. A CPN model (lower left of Fig. 2) has been developed by modelling the natural language protocol specification [22] (lower right) of the interoperability protocol. The modelling activity transforms the natural language specification into a formal executable specification represented by the CPN model. The CPN model captures the network architecture depicted in Fig. 1 and the protocol mechanisms of the interoperability protocol, e.g., the periodic transmission of advertisements, the dynamic updates of the DNS database, and traffic flows between hosts in the core network and nodes in the ad-hoc

Specification

Motivation

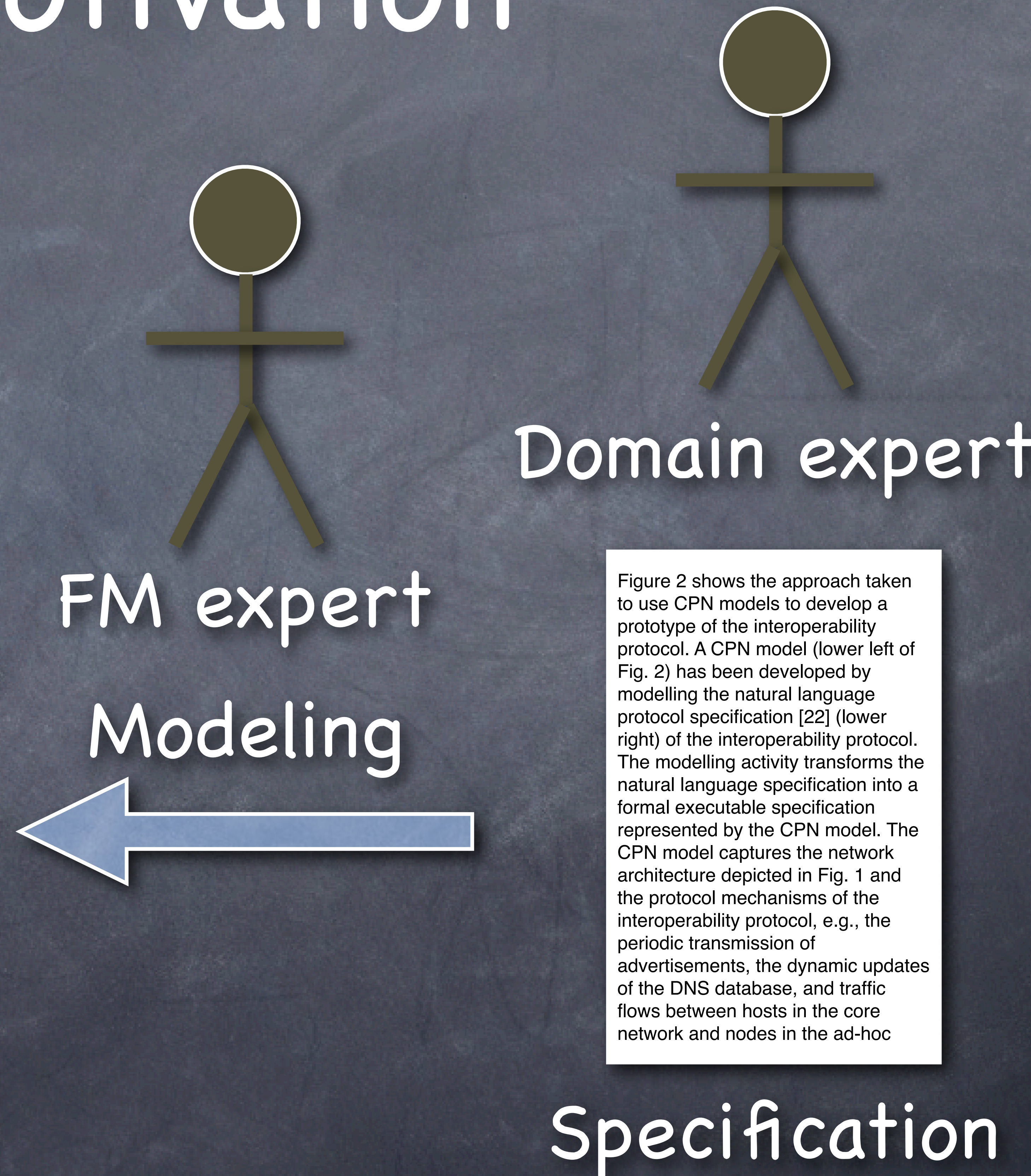
Domain expert

FM expert

Figure 2 shows the approach taken to use CPN models to develop a prototype of the interoperability protocol. A CPN model (lower left of Fig. 2) has been developed by modelling the natural language protocol specification [22] (lower right) of the interoperability protocol. The modelling activity transforms the natural language specification into a formal executable specification represented by the CPN model. The CPN model captures the network architecture depicted in Fig. 1 and the protocol mechanisms of the interoperability protocol, e.g., the periodic transmission of advertisements, the dynamic updates of the DNS database, and traffic flows between hosts in the core network and nodes in the ad-hoc

Specification

Motivation



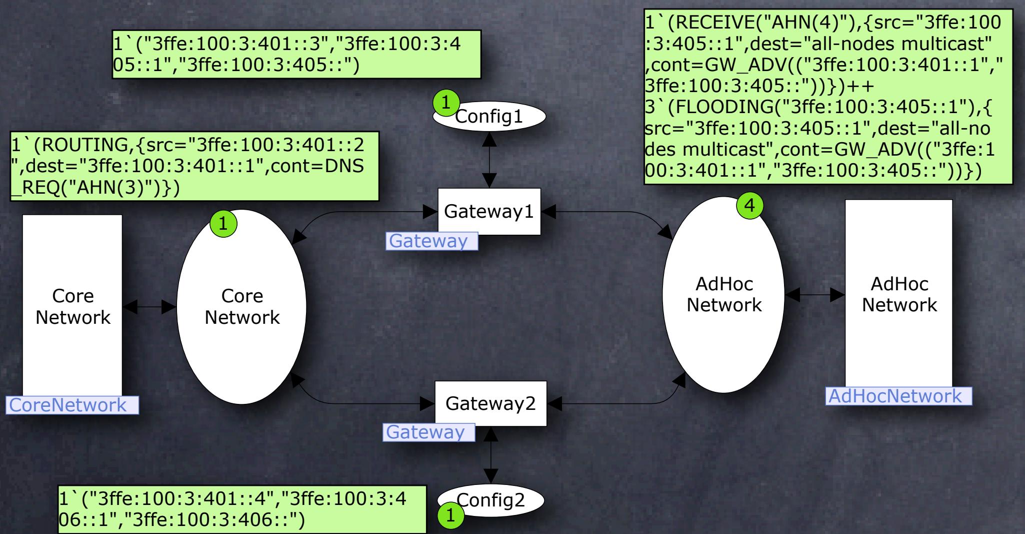
Domain expert
FM expert
Modeling
Figure 2 shows the approach taken to use CPN models to develop a prototype of the interoperability protocol. A CPN model (lower left of Fig. 2) has been developed by modelling the natural language protocol specification [22] (lower right) of the interoperability protocol. The modelling activity transforms the natural language specification into a formal executable specification represented by the CPN model. The CPN model captures the network architecture depicted in Fig. 1 and the protocol mechanisms of the interoperability protocol, e.g., the periodic transmission of advertisements, the dynamic updates of the DNS database, and traffic flows between hosts in the core network and nodes in the ad-hoc
Specification

Motivation

Domain expert

FM expert

Modeling



Formal model

Figure 2 shows the approach taken to use CPN models to develop a prototype of the interoperability protocol. A CPN model (lower left of Fig. 2) has been developed by modelling the natural language protocol specification [22] (lower right) of the interoperability protocol. The modelling activity transforms the natural language specification into a formal executable specification represented by the CPN model. The CPN model captures the network architecture depicted in Fig. 1 and the protocol mechanisms of the interoperability protocol, e.g., the periodic transmission of advertisements, the dynamic updates of the DNS database, and traffic flows between hosts in the core network and nodes in the ad-hoc

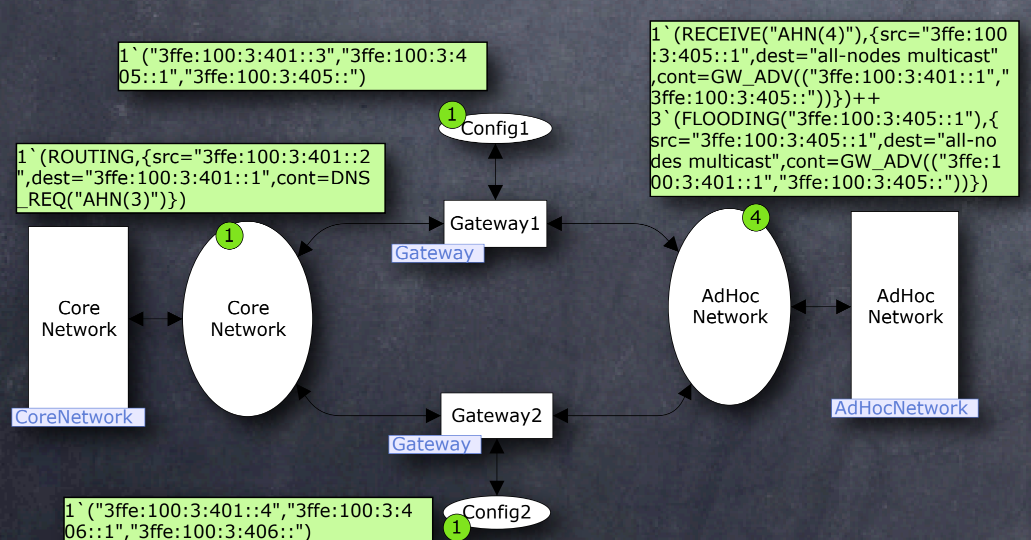
Specification

Motivation

Domain expert

Figure 2 shows the approach taken to use CPN models to develop a prototype of the interoperability protocol. A CPN model (lower left of Fig. 2) has been developed by modelling the natural language protocol specification [22] (lower right) of the interoperability protocol. The modelling activity transforms the natural language specification into a formal executable specification represented by the CPN model. The CPN model captures the network architecture depicted in Fig. 1 and the protocol mechanisms of the interoperability protocol, e.g., the periodic transmission of advertisements, the dynamic updates of the DNS database, and traffic flows between hosts in the core network and nodes in the ad-hoc

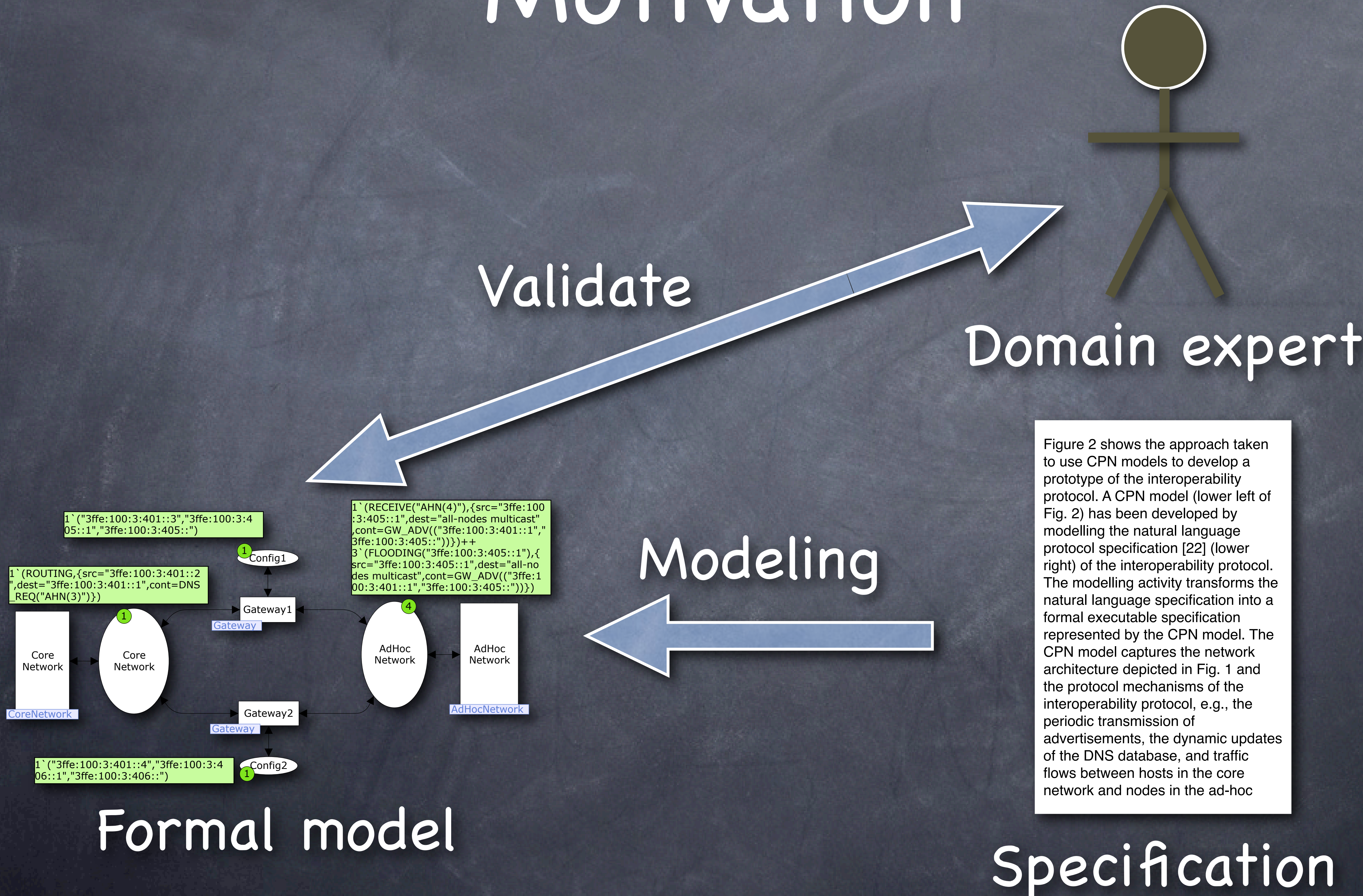
Modeling



Formal model

Specification

Motivation

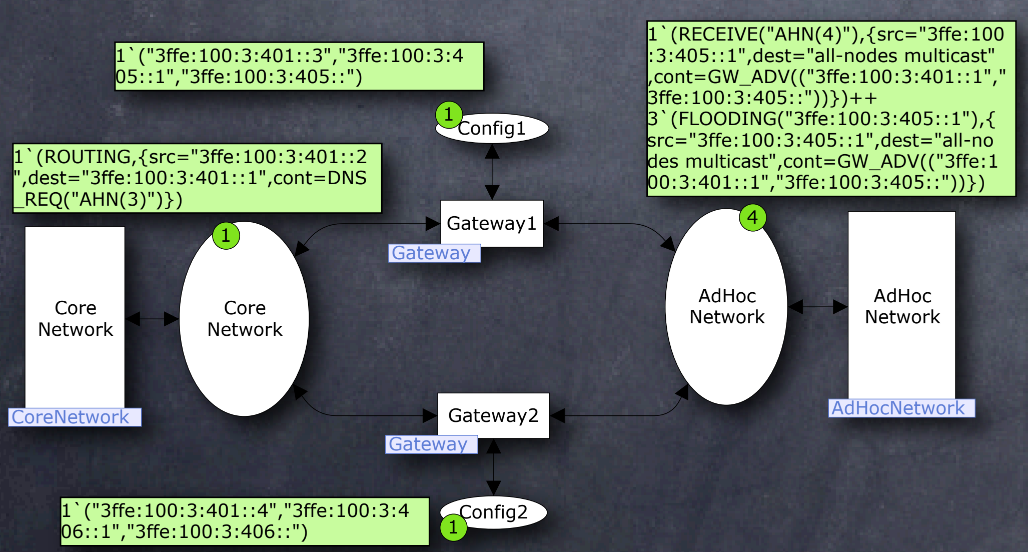


Validate
Domain expert
Modeling
1`("3ffe:100:3:401::3","3ffe:100:3:405::1","3ffe:100:3:405::")
Config1
1`(ROUTING,{src="3ffe:100:3:401::2",dest="3ffe:100:3:401::1",cont=DNS_REQ("AHN(3)")})
1`(RECEIVE("AHN(4)"),{src="3ffe:100:3:405::1",dest="all-nodes multicast",cont=GW_ADV(("3ffe:100:3:401::1","3ffe:100:3:405::"))})++
3`(FLOODING("3ffe:100:3:405::1"),{src="3ffe:100:3:405::1",dest="all-nodes multicast",cont=GW_ADV(("3ffe:100:3:401::1","3ffe:100:3:405::"))})
Gateway1
Gateway
Core Network
Core Network
CoreNetwork
AdHoc Network
AdHoc Network
AdHocNetwork
Gateway2
Gateway
Config2
1`("3ffe:100:3:401::4","3ffe:100:3:406::1","3ffe:100:3:406::")
Formal model
Figure 2 shows the approach taken to use CPN models to develop a prototype of the interoperability protocol. A CPN model (lower left of Fig. 2) has been developed by modelling the natural language protocol specification [22] (lower right) of the interoperability protocol. The modelling activity transforms the natural language specification into a formal executable specification represented by the CPN model. The CPN model captures the network architecture depicted in Fig. 1 and the protocol mechanisms of the interoperability protocol, e.g., the periodic transmission of advertisements, the dynamic updates of the DNS database, and traffic flows between hosts in the core network and nodes in the ad-hoc
Specification

Motivation

Domain expert

Modeling



Formal model

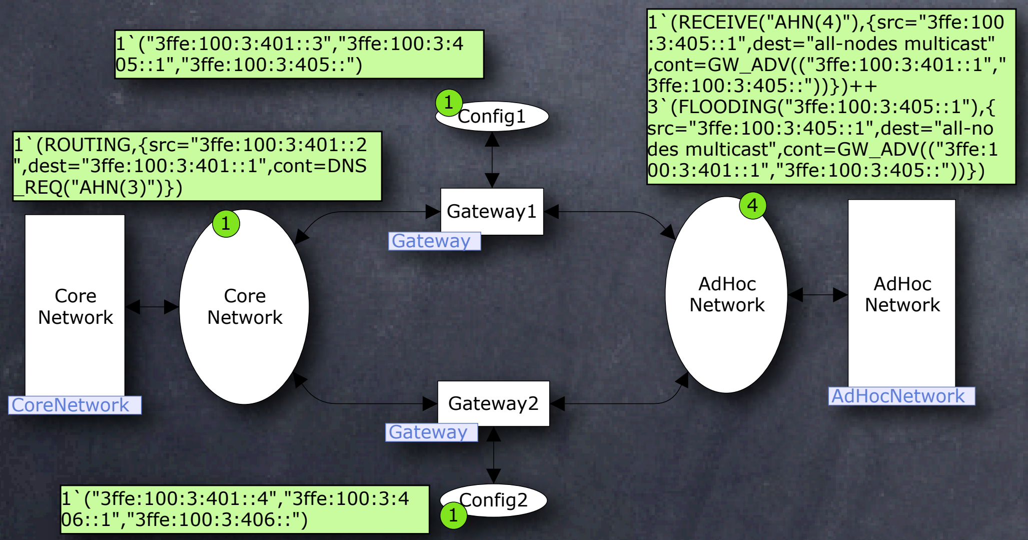
Figure 2 shows the approach taken to use CPN models to develop a prototype of the interoperability protocol. A CPN model (lower left of Fig. 2) has been developed by modelling the natural language protocol specification [22] (lower right) of the interoperability protocol. The modelling activity transforms the natural language specification into a formal executable specification represented by the CPN model. The CPN model captures the network architecture depicted in Fig. 1 and the protocol mechanisms of the interoperability protocol, e.g., the periodic transmission of advertisements, the dynamic updates of the DNS database, and traffic flows between hosts in the core network and nodes in the ad-hoc

Specification

Methodology

Domain expert

Modeling



Formal model

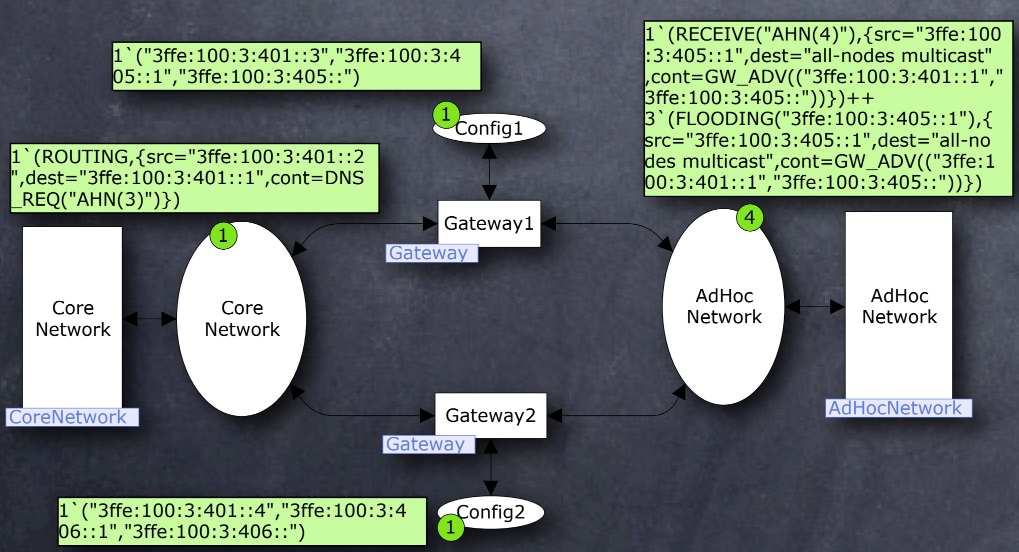
Figure 2 shows the approach taken to use CPN models to develop a prototype of the interoperability protocol. A CPN model (lower left of Fig. 2) has been developed by modelling the natural language protocol specification [22] (lower right) of the interoperability protocol. The modelling activity transforms the natural language specification into a formal executable specification represented by the CPN model. The CPN model captures the network architecture depicted in Fig. 1 and the protocol mechanisms of the interoperability protocol, e.g., the periodic transmission of advertisements, the dynamic updates of the DNS database, and traffic flows between hosts in the core network and nodes in the ad-hoc

Specification

Methodology

FM expert

Domain expert



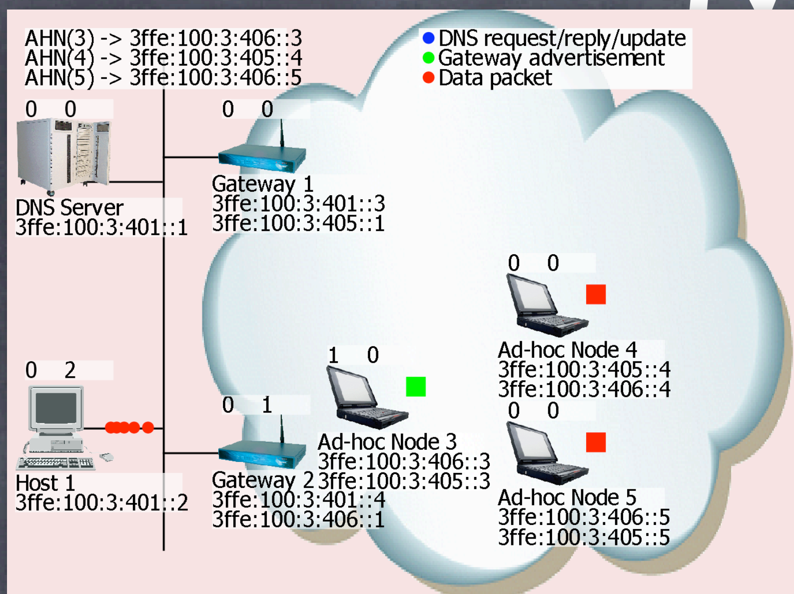
Formal model

Modeling

Figure 2 shows the approach taken to use CPN models to develop a prototype of the interoperability protocol. A CPN model (lower left of Fig. 2) has been developed by modelling the natural language protocol specification [22] (lower right) of the interoperability protocol. The modelling activity transforms the natural language specification into a formal executable specification represented by the CPN model. The CPN model captures the network architecture depicted in Fig. 1 and the protocol mechanisms of the interoperability protocol, e.g., the periodic transmission of advertisements, the dynamic updates of the DNS database, and traffic flows between hosts in the core network and nodes in the ad-hoc

Specification

Methodology



Animation

FM expert

Domain expert

Modeling

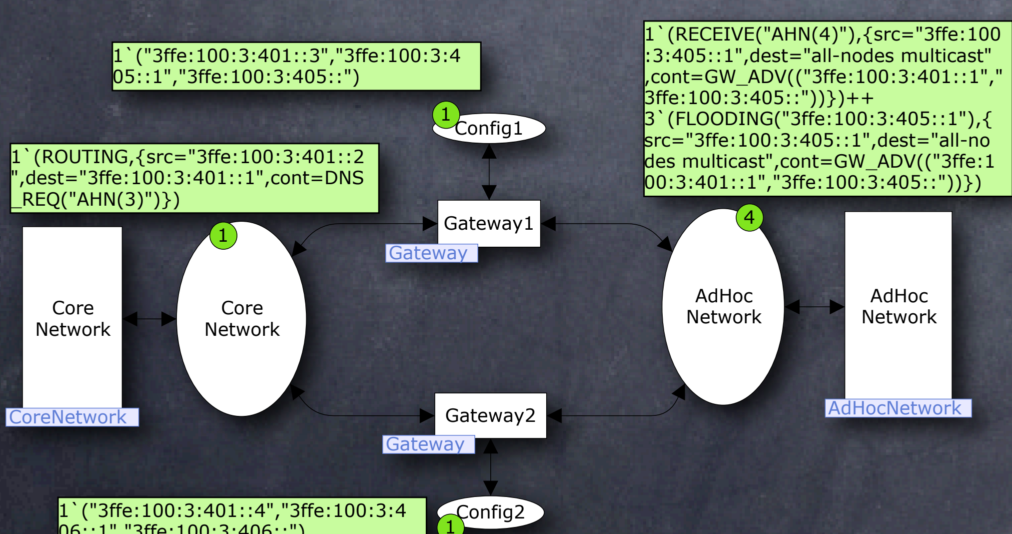
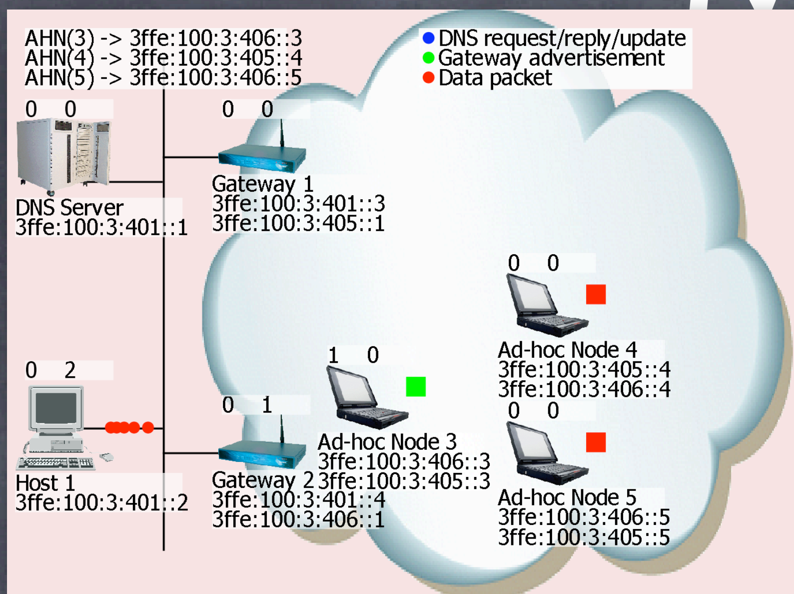


Figure 2 shows the approach taken to use CPN models to develop a prototype of the interoperability protocol. A CPN model (lower left of Fig. 2) has been developed by modelling the natural language protocol specification [22] (lower right) of the interoperability protocol. The modelling activity transforms the natural language specification into a formal executable specification represented by the CPN model. The CPN model captures the network architecture depicted in Fig. 1 and the protocol mechanisms of the interoperability protocol, e.g., the periodic transmission of advertisements, the dynamic updates of the DNS database, and traffic flows between hosts in the core network and nodes in the ad-hoc

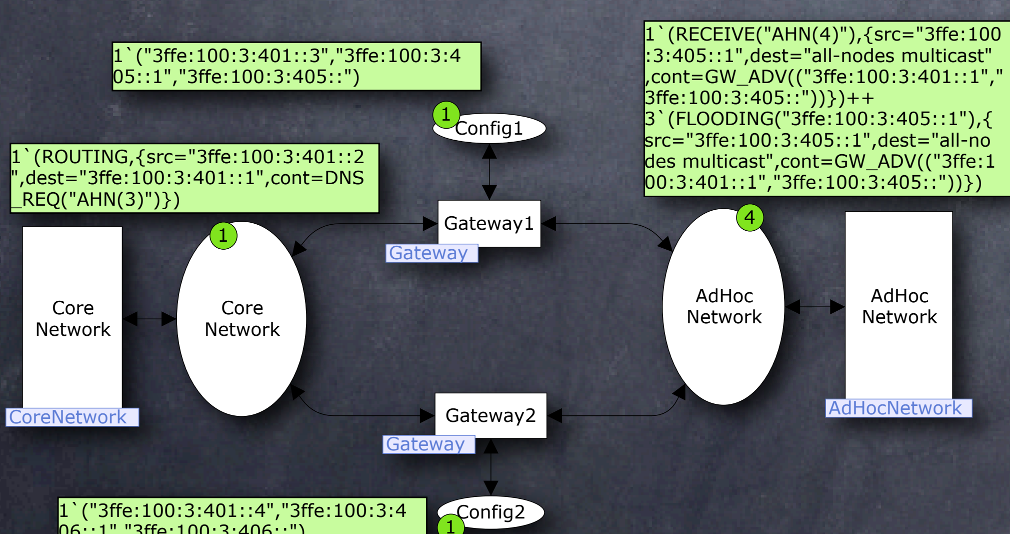
Formal model

Specification

Methodology



Animation



Formal model

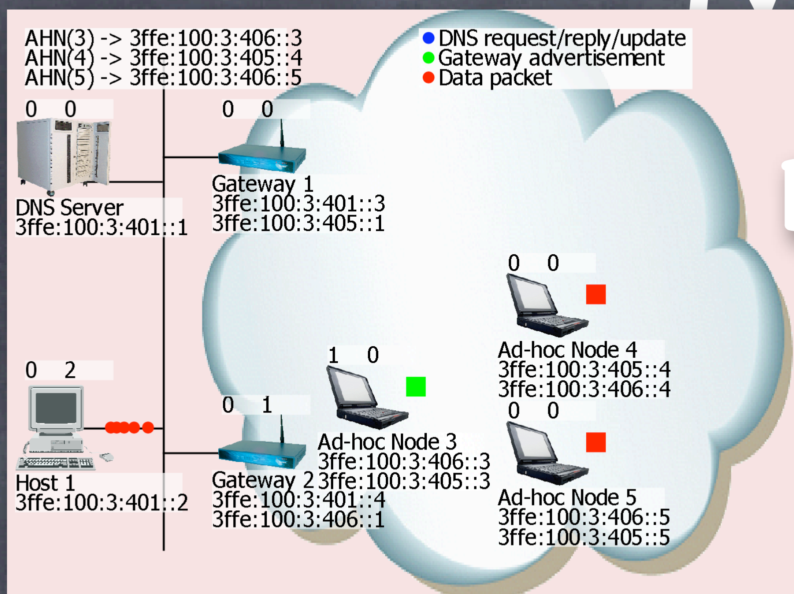
Modeling

Domain expert

Figure 2 shows the approach taken to use CPN models to develop a prototype of the interoperability protocol. A CPN model (lower left of Fig. 2) has been developed by modelling the natural language protocol specification [22] (lower right) of the interoperability protocol. The modelling activity transforms the natural language specification into a formal executable specification represented by the CPN model. The CPN model captures the network architecture depicted in Fig. 1 and the protocol mechanisms of the interoperability protocol, e.g., the periodic transmission of advertisements, the dynamic updates of the DNS database, and traffic flows between hosts in the core network and nodes in the ad-hoc

Specification

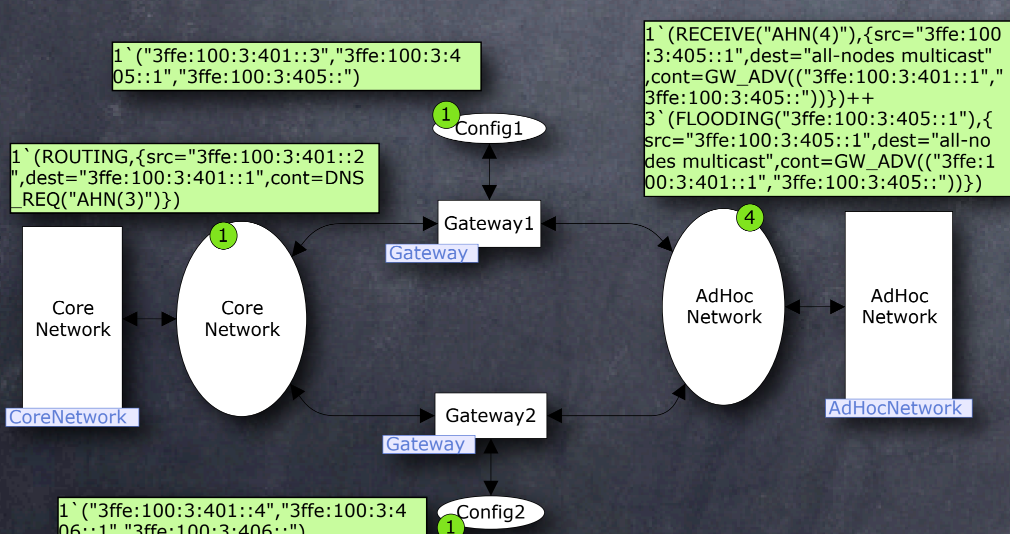
Methodology



Explore and interact

Domain expert

Animation



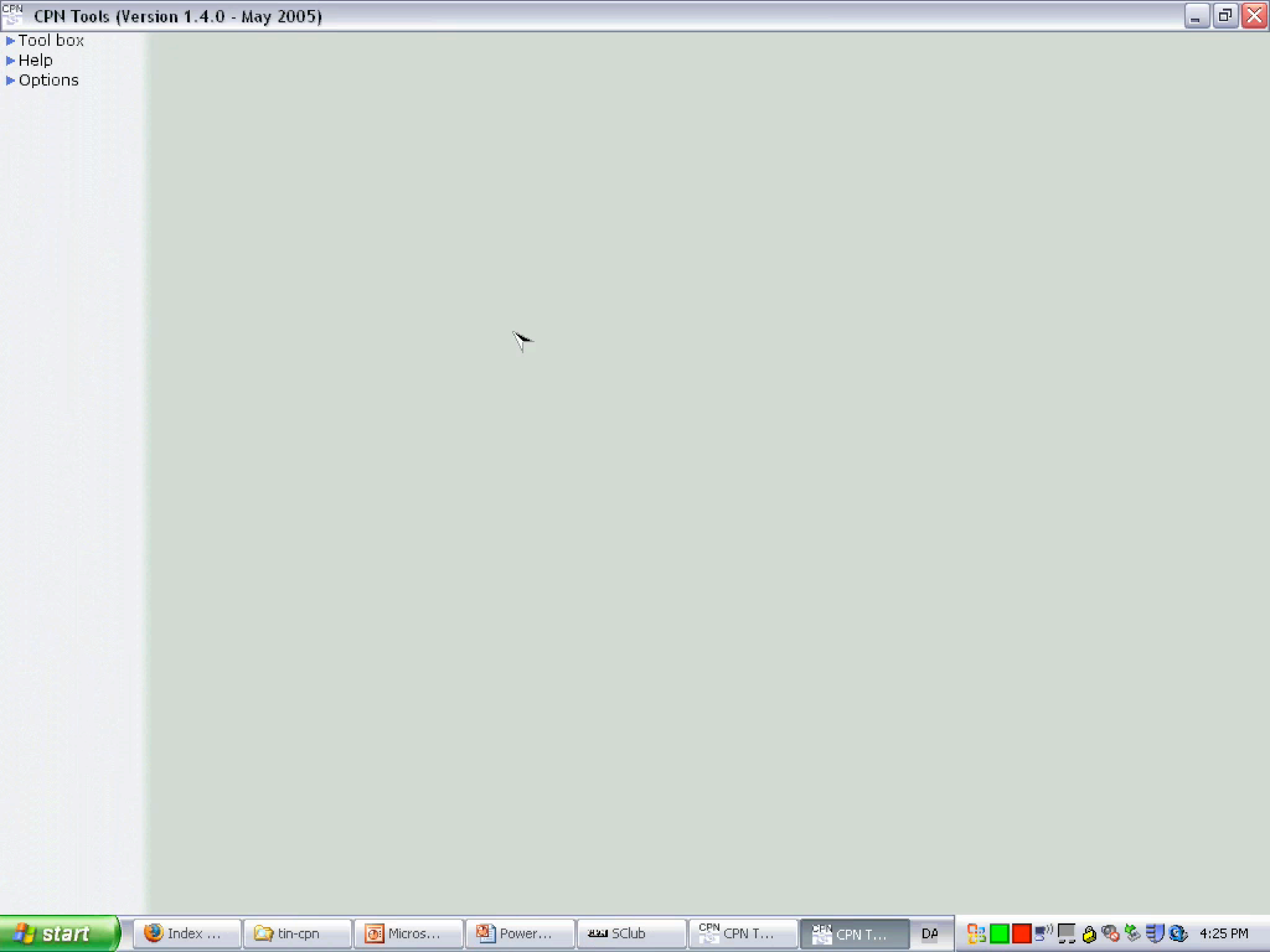
Modeling

Formal model

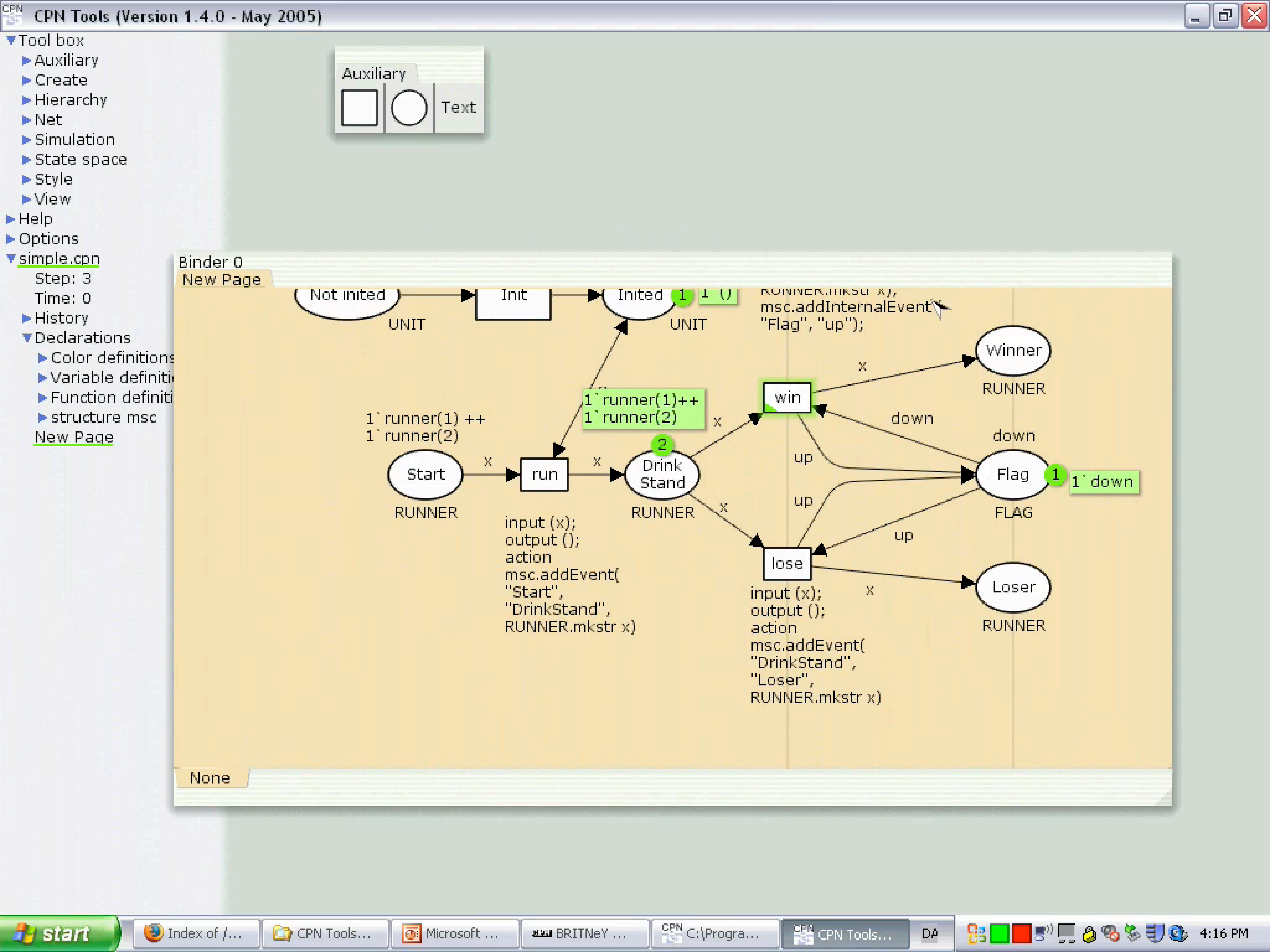
Figure 2 shows the approach taken to use CPN models to develop a prototype of the interoperability protocol. A CPN model (lower left of Fig. 2) has been developed by modelling the natural language protocol specification [22] (lower right) of the interoperability protocol. The modelling activity transforms the natural language specification into a formal executable specification represented by the CPN model. The CPN model captures the network architecture depicted in Fig. 1 and the protocol mechanisms of the interoperability protocol, e.g., the periodic transmission of advertisements, the dynamic updates of the DNS database, and traffic flows between hosts in the core network and nodes in the ad-hoc

Specification

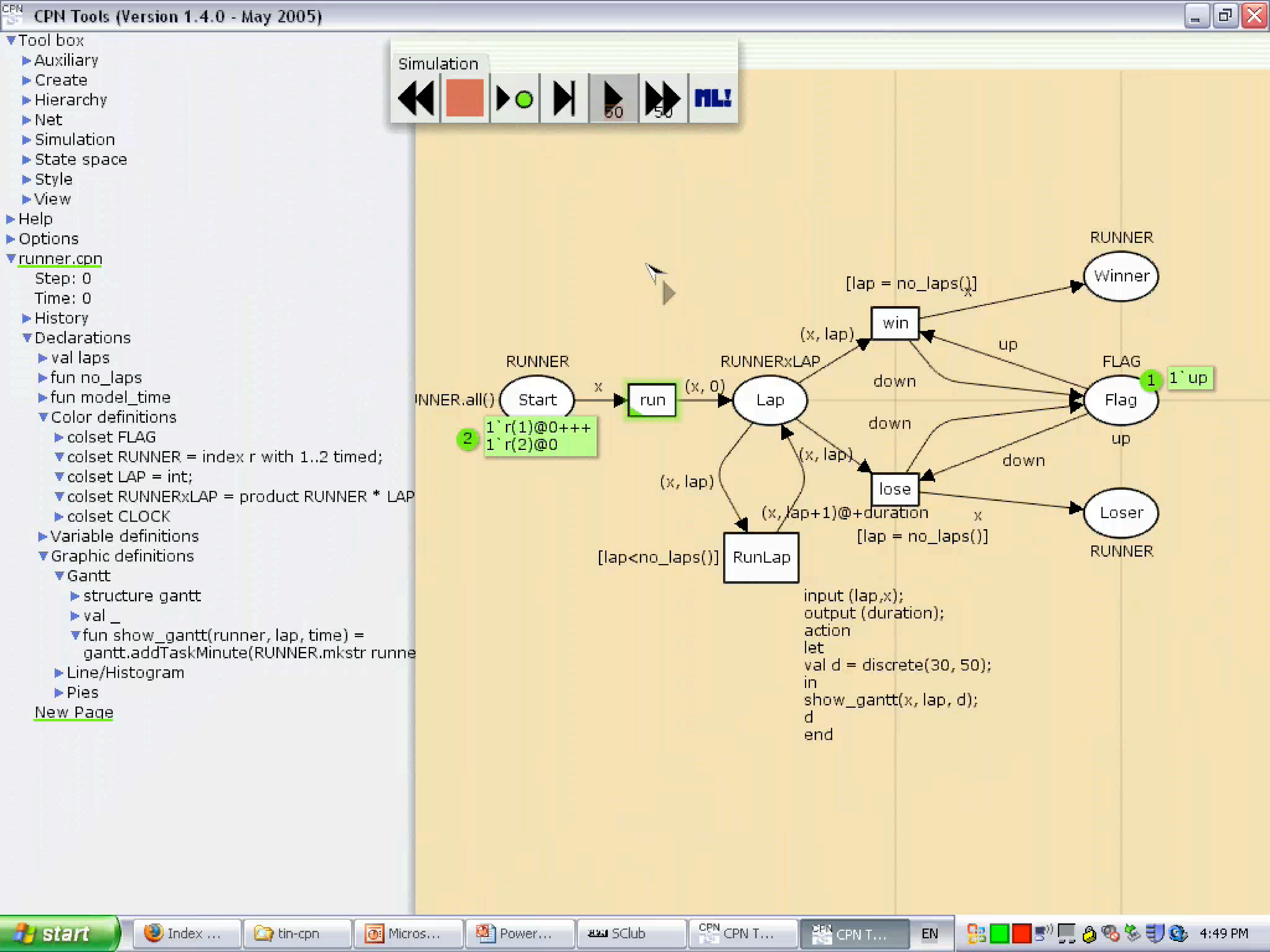
Good-looking Runners
Executive Edition


CPN Tools (Version 1.4.0 - May 2005)
Tool box
Help
Options
start
Index ...
tin-cpn
Micros...
Power...
SClub
CPN T...
CPN T...
4:25 PM

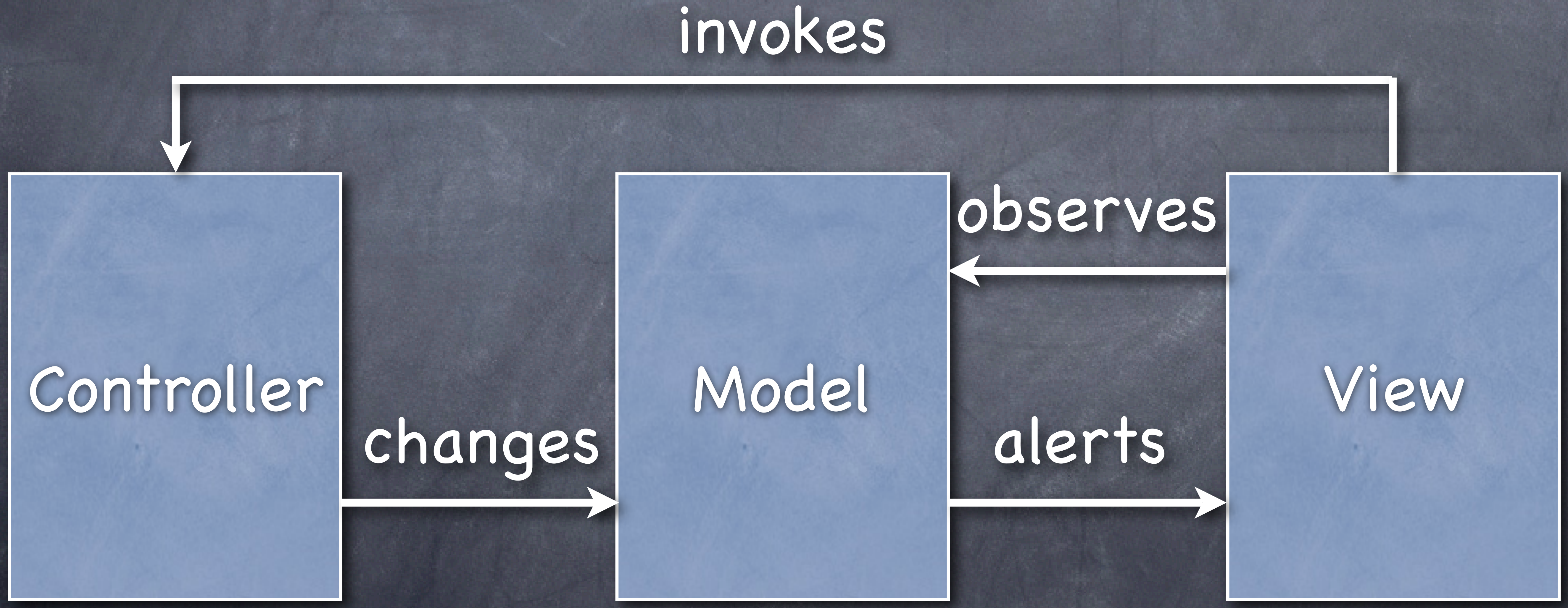
Good-looking Runners
Engineer Edition


CPN Tools (Version 1.4.0 - May 2005)
Tool box
Auxiliary
Create
Hierarchy
Net
Simulation
State space
Style
View
Help
Options
simple.cpn
Step: 3
Time: 0
History
Declarations
Color definitions
Variable definiti
Function definiti
structure msc
New Page
Auxiliary
Text
Binder 0
New Page
Not inited
UNIT
Init
Inited
1
1`()
UNIT
msc.addInternalEvent(
"Flag", "up");
Winner
RUNNER
x
win
1`runner(1)++
1`runner(2)
1`runner(1) ++
1`runner(2)
down
down
Start
RUNNER
x
run
x
2
Drink Stand
RUNNER
up
up
Flag
1
1`down
FLAG
up
input (x);
output ();
action
msc.addEvent(
"Start",
"DrinkStand",
RUNNER.mkstr x)
lose
Loser
RUNNER
input (x);
output ();
action
msc.addEvent(
"DrinkStand",
"Loser",
RUNNER.mkstr x)
None
start
Index of /...
CPN Tools...
Microsoft ...
BRITNeY ...
C:\Progra...
CPN Tools...
DA
4:16 PM

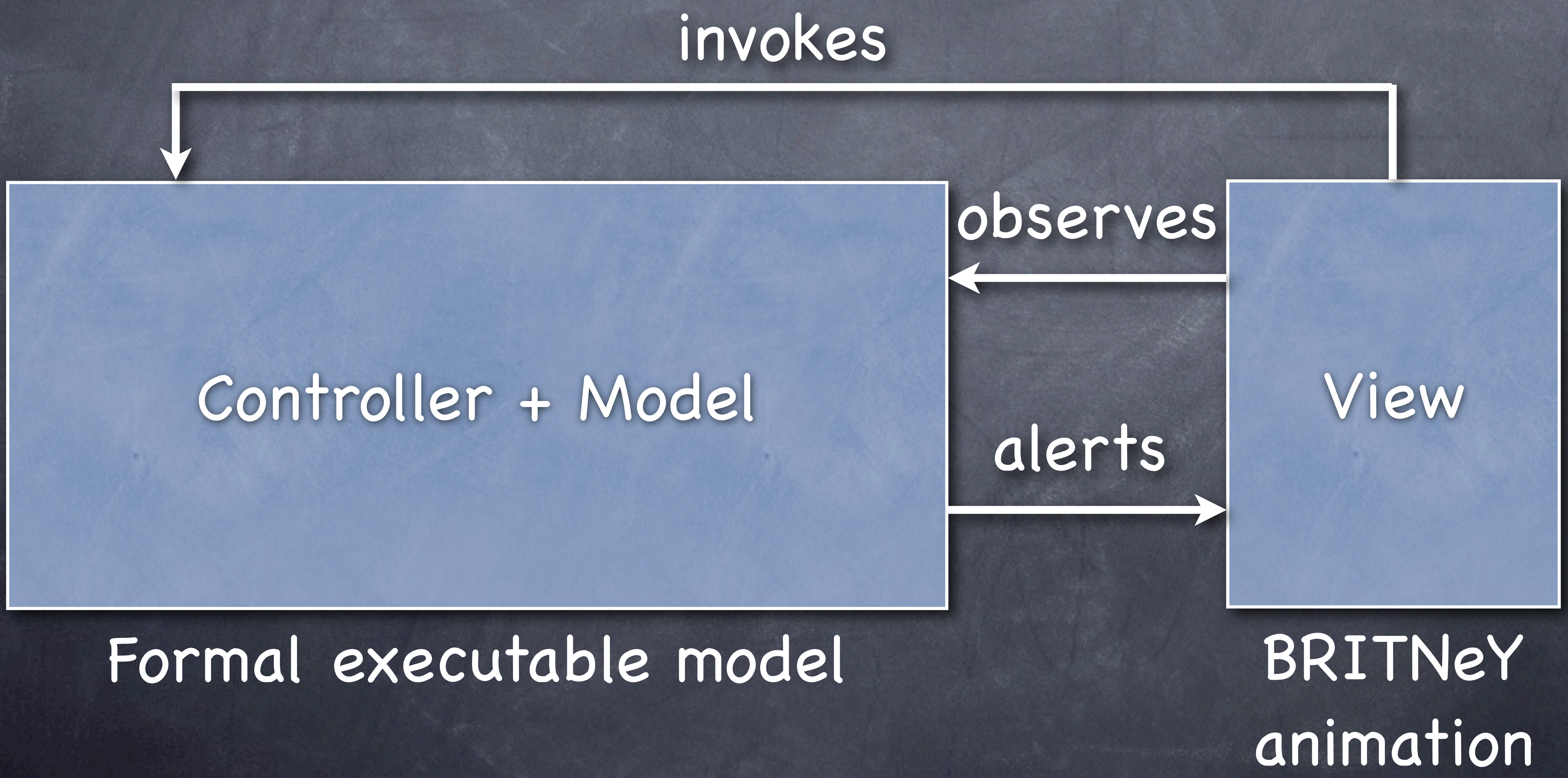
Good-looking Runners Performance Analyst Edition


CPN Tools (Version 1.4.0 - May 2005)
Tool box
Auxiliary
Create
Hierarchy
Net
Simulation
State space
Style
View
Help
Options
runner.cpn
Step: 0
Time: 0
History
Declarations
val laps
fun no_laps
fun model_time
Color definitions
colset FLAG
colset RUNNER = index r with 1..2 timed;
colset LAP = int;
colset RUNNERxLAP = product RUNNER * LAP
colset CLOCK
Variable definitions
Graphic definitions
Gantt
structure gantt
val _
fun show_gantt(runner, lap, time) =
gantt.addTaskMinute(RUNNER.mkstr runne
Line/Histogram
Pies
New Page
Simulation
RUNNER
Winner
[lap = no_laps()]
win
(x, lap)
up
RUNNER
RUNNERxLAP
FLAG
down
1`up
NNER.all()
Start
x
run
(x, 0)
Lap
Flag
1`r(1)@0+++
1`r(2)@0
down
up
(x, lap)
down
(x, lap)
lose
(x, lap+1)@+duration
Loser
[lap = no_laps()]
RUNNER
[lap<no_laps()]
RunLap
input (lap,x);
output (duration);
action
let
val d = discrete(30, 50);
in
show_gantt(x, lap, d);
d
end
start
Index ...
tin-cpn
Micros...
Power...
SClub
CPN T...
CPN T...
EN
4:49 PM

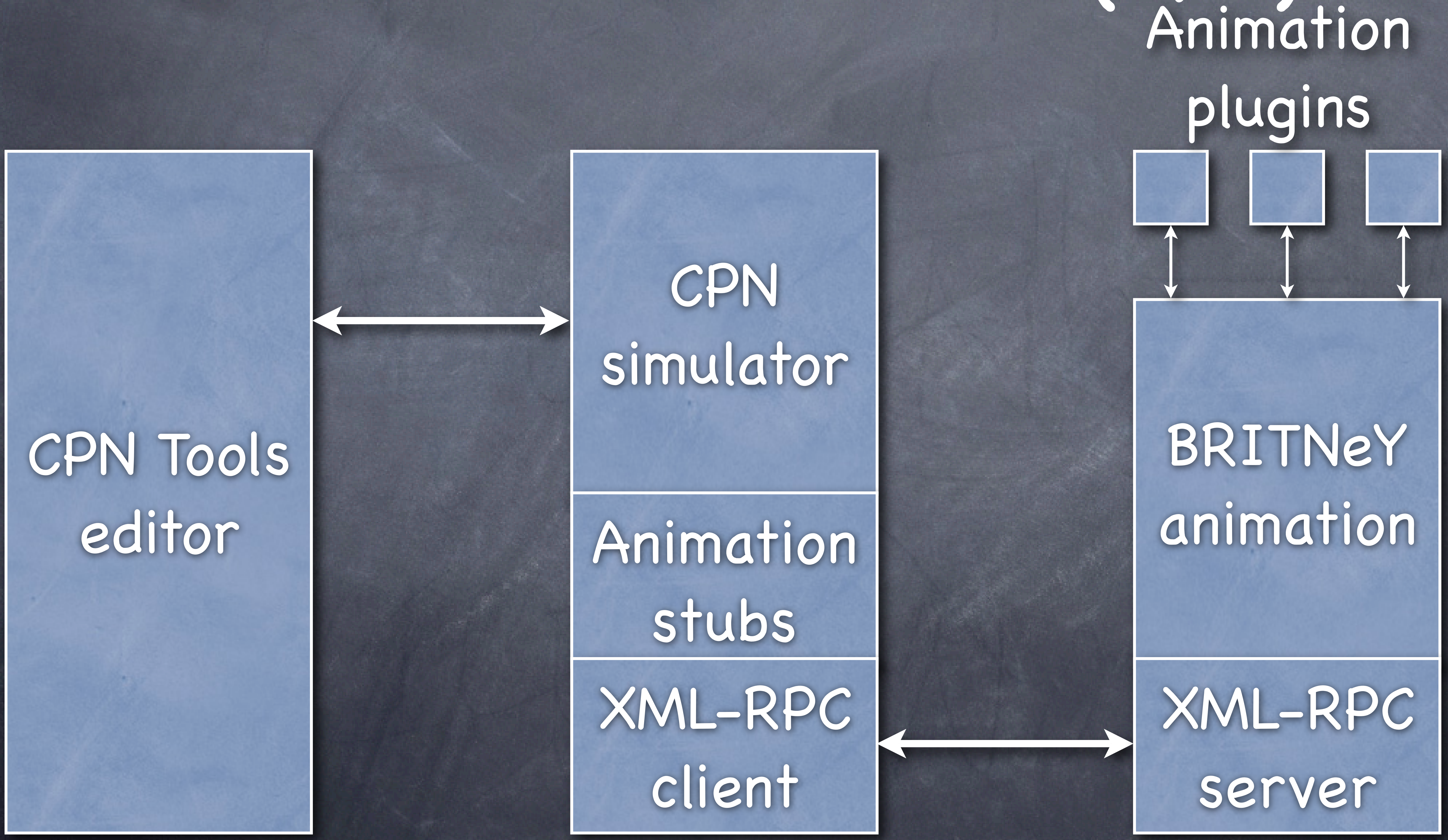
BRITNeY animation (1/3)



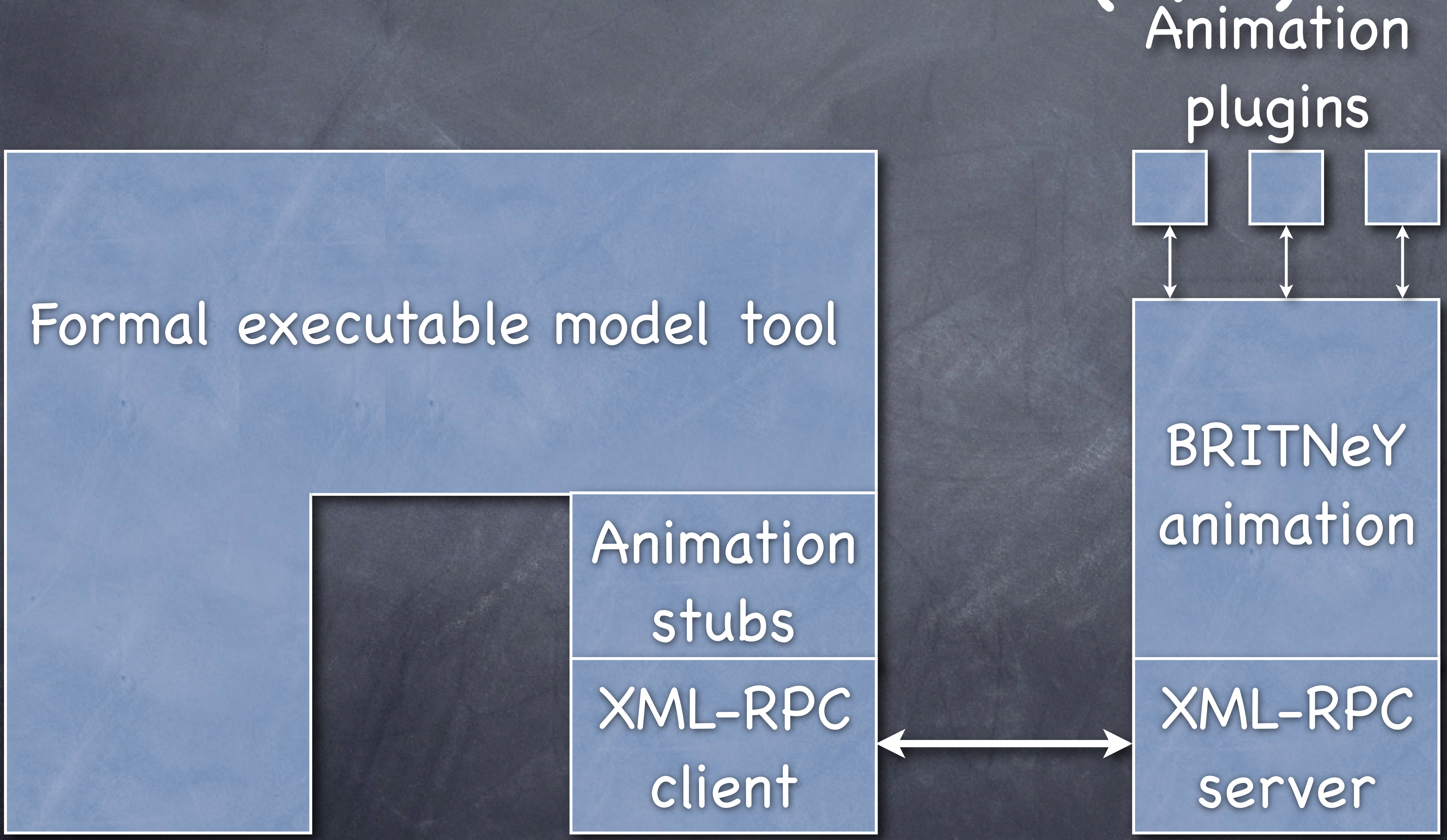
BRITNeY animation (2/3)



BRITNeY animation (3/3)



BRITNeY animation (3/3)



More Information about BRITNeY animation

- Tool web-page: http://wiki.daimi.au.dk/tincpn
- Screen-cast from CPN Workshop 2005 tutorial: http://www.daimi.au.dk/~mw/local/demo/BRITNeY animation/
- Case study: L.M. Kristensen, M. Westergaard, and P.C. Nørgaard: Model-based Prototyping of an Interoperability Protocol forMobile Ad-hoc Networks, IFM 2005

Behaving Well

A state space tool

State Space Analysis

State Space Analysis

- Reachability: Does any possible state of the system satisfy a given property?

State Space Analysis

- Reachability: Does any possible state of the system satisfy a given property?
  - E.g., can more than one runner win?

State Space Analysis

- Reachability: Does any possible state of the system satisfy a given property?
  - E.g., can more than one runner win?
- Analysis of CP-nets is impossible

State Space Analysis

- Reachability: Does any possible state of the system satisfy a given property?
  - E.g., can more than one runner win?
- Analysis of CP-nets is impossible
- How do we do it anyway? Try all possible states

State Space Analysis

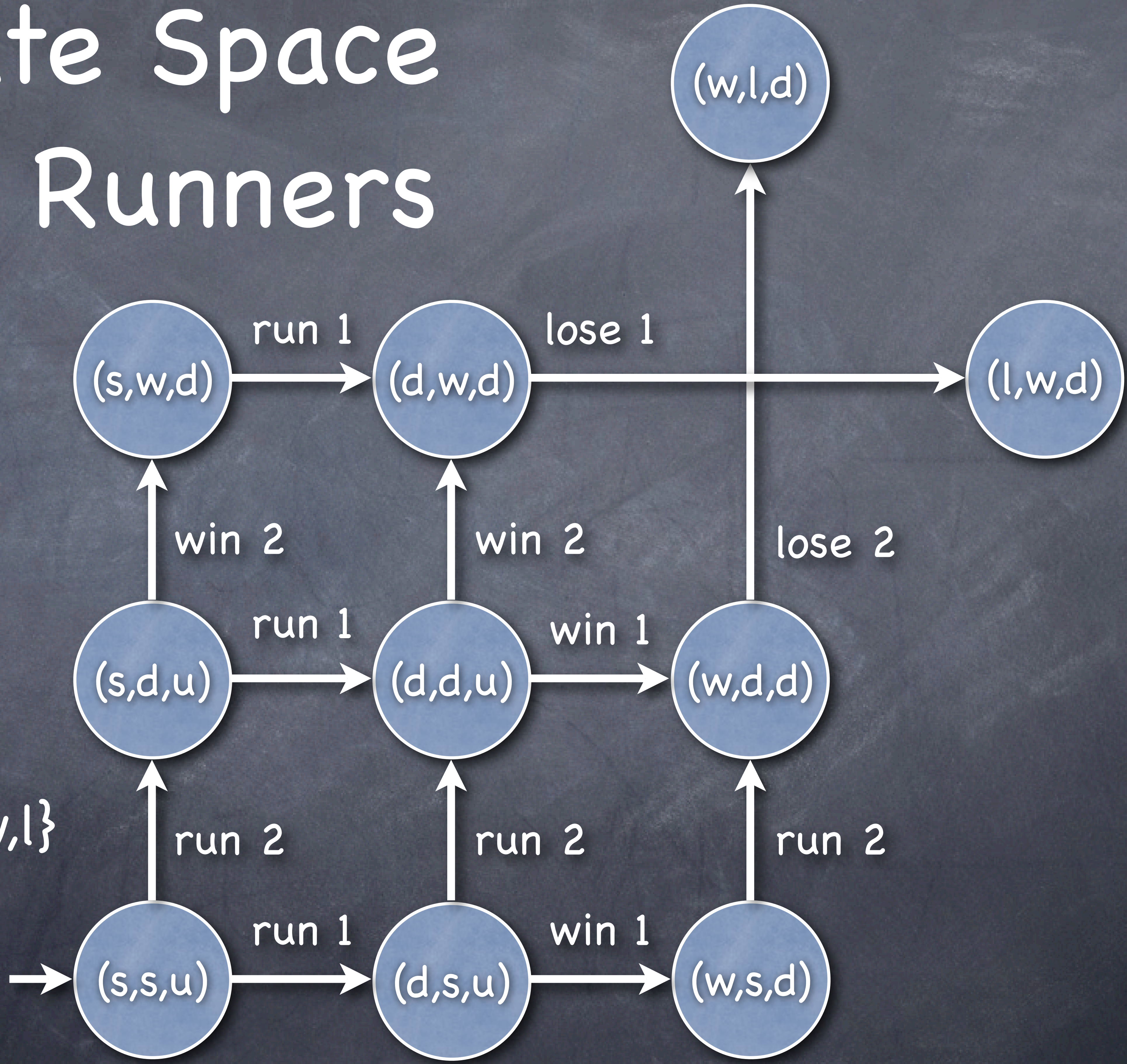
- Reachability: Does any possible state of the system satisfy a given property?
  - E.g., can more than one runner win?
- Analysis of CP-nets is impossible
- How do we do it anyway? Try all possible states
- Loops? Build reachability graph

State Space for Runners

(r1,r2,flag)

r1,r2 ∈ {s,d,w,l}

flag ∈ {u,d}



Simple Algorithm for State Space Analysis

```
Queue.add(Simulator.get_initial state())
while !Queue.is_empty() do
    s := Queue.remove_first()
    Storage.add(s)
    process(s)
    forall s' in Simulator.get_next(s) do
        if !Storage.contains(s') then
            Queue.add(s')
        endif
    endfor
endwhile
```

Parametrizing the Algorithm

- The algorithm relies on 3 data-structures:
  - Simulator (get_initial_state, get_next)
  - Queue (add, is_empty, remove_first)
  - Storage (add, contains)
- By providing different implementations, we can control which formalism to use (Simulator), how to traverse the state space (Queue - waiting/unprocessed), and how to store data efficiently (Storage - passed/processed)

Problems with State Space Analysis

Problems with State Space Analysis

- Problem: The reachability graph is large, often even infinite

Problems with State Space Analysis

- Problem: The reachability graph is large, often even infinite
  - Solution 1: Store only some of the graph

Problems with State Space Analysis

- Problem: The reachability graph is large, often even infinite
  - Solution 1: Store only some of the graph
  - Solution 2: Store each node more efficiently

Problems with State Space Analysis

- Problem: The reachability graph is large, often even infinite
  - Solution 1: Store only some of the graph
  - Solution 2: Store each node more efficiently
- A lot of so-called reduction methods exist and new reduction methods are found out every day

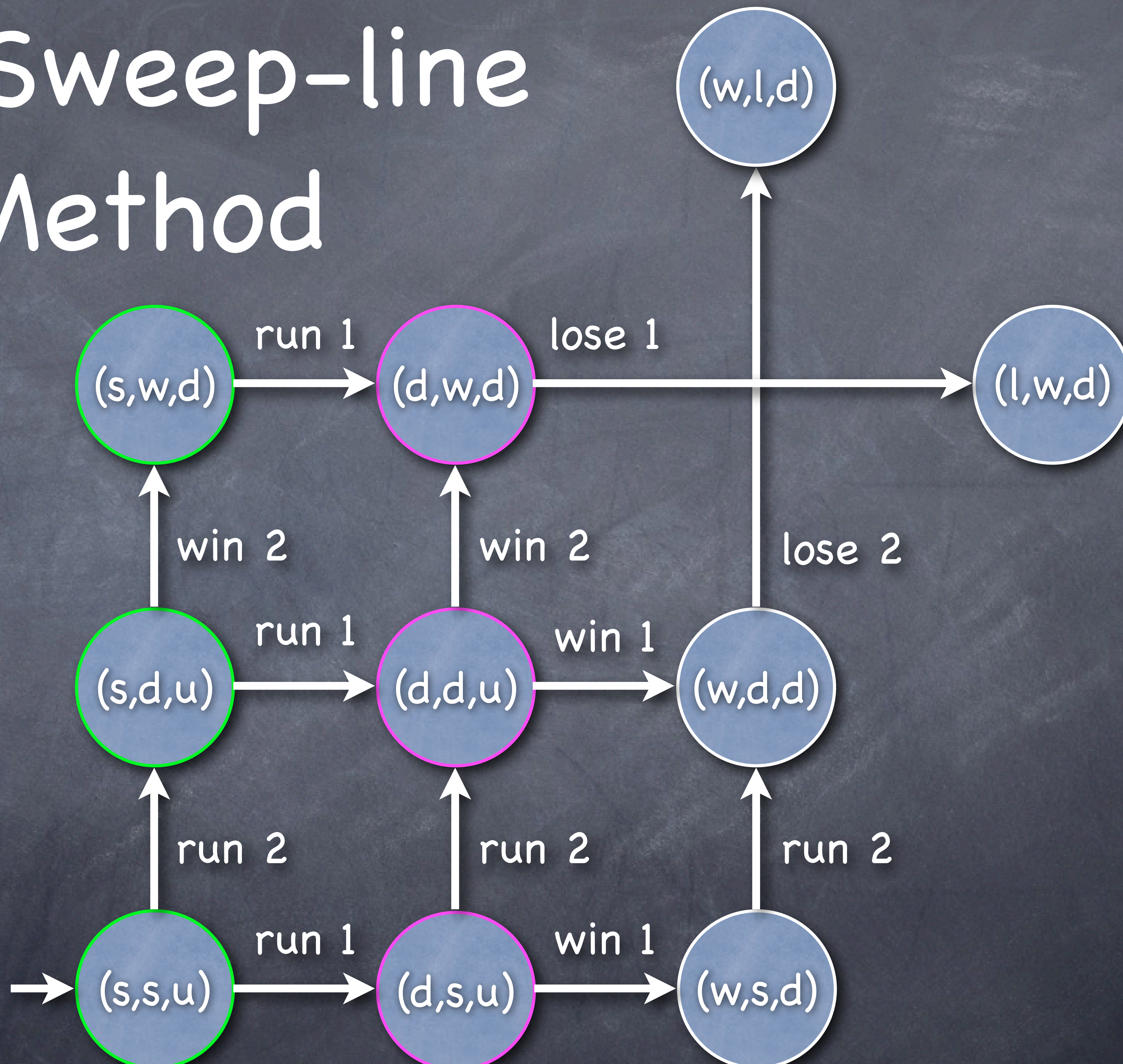
A Condensed Representation (1/2)

- We want to represent the entire state space
- A state of the system is (r1,r2,flag) with r1,r2$\in${s,d,w,l} and flag$\in${u,d}
- Only some (10) of the syntactically possible states ($4 \cdot 4 \cdot 2=32$) are reachable
- At least ceil(log(32))=5 bits are used to store each state, although ceil(log(10))=4 bits would suffice

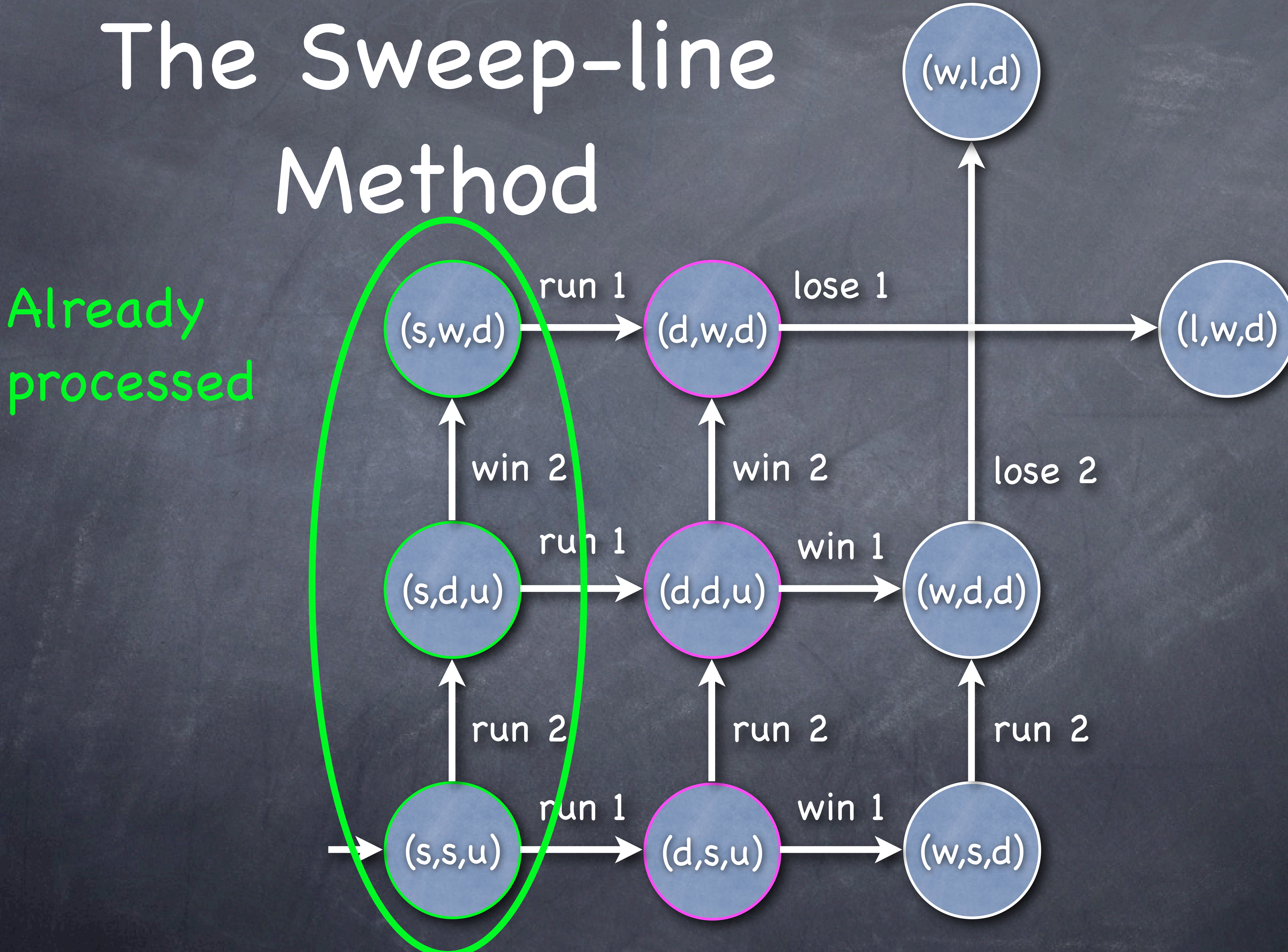
A Condensed Representation (2/2)

- In realistic examples, the number of syntactically possible states is much larger than the number of reachable states, so distinguishing only between reachable states yields a good reduction
- Alas, we first know the number of reachable states, when we have constructed the reachability graph

The Sweep-line Method



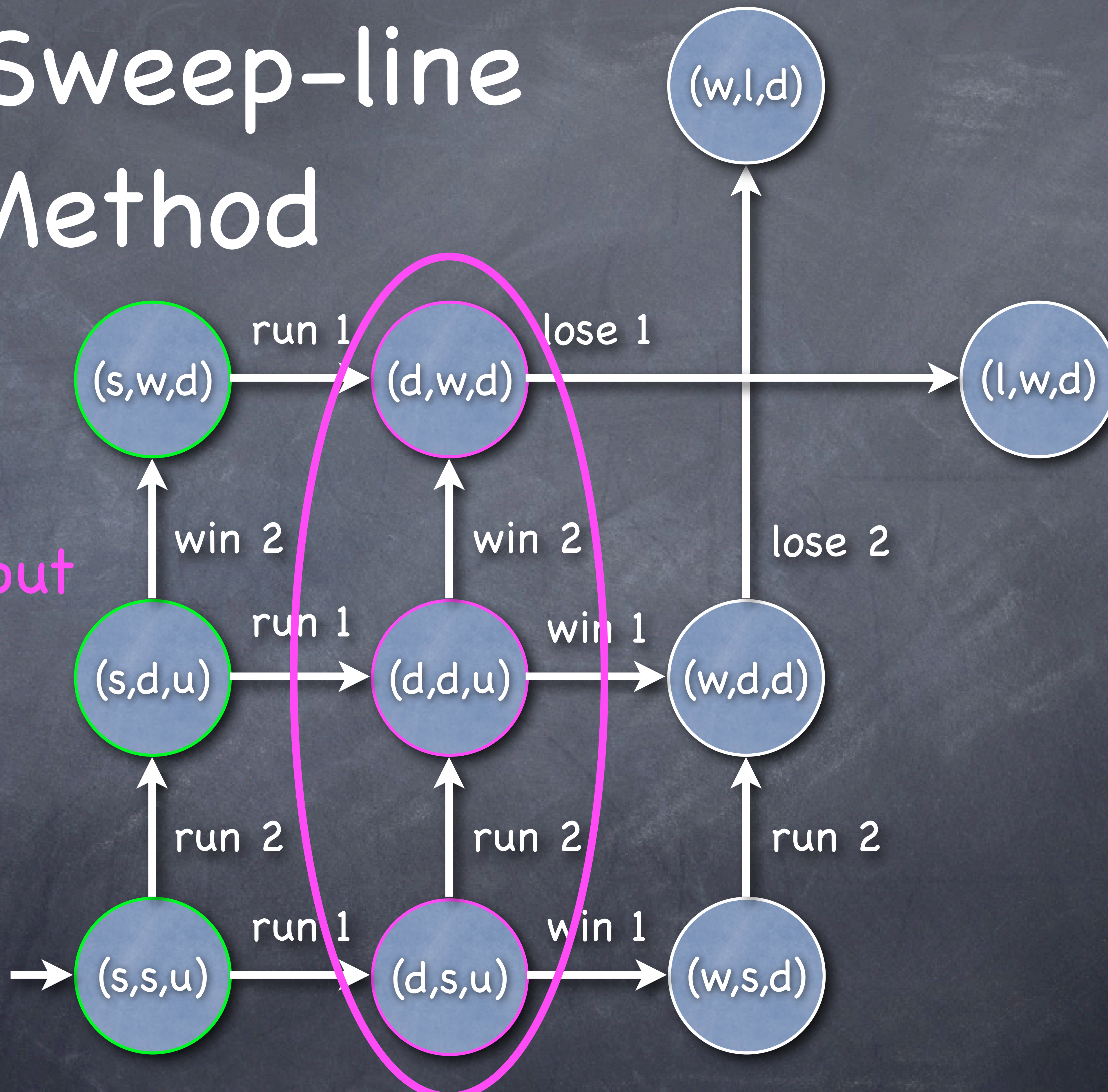
The Sweep-line Method



The Sweep-line Method

Already processed

Discovered but not yet processed

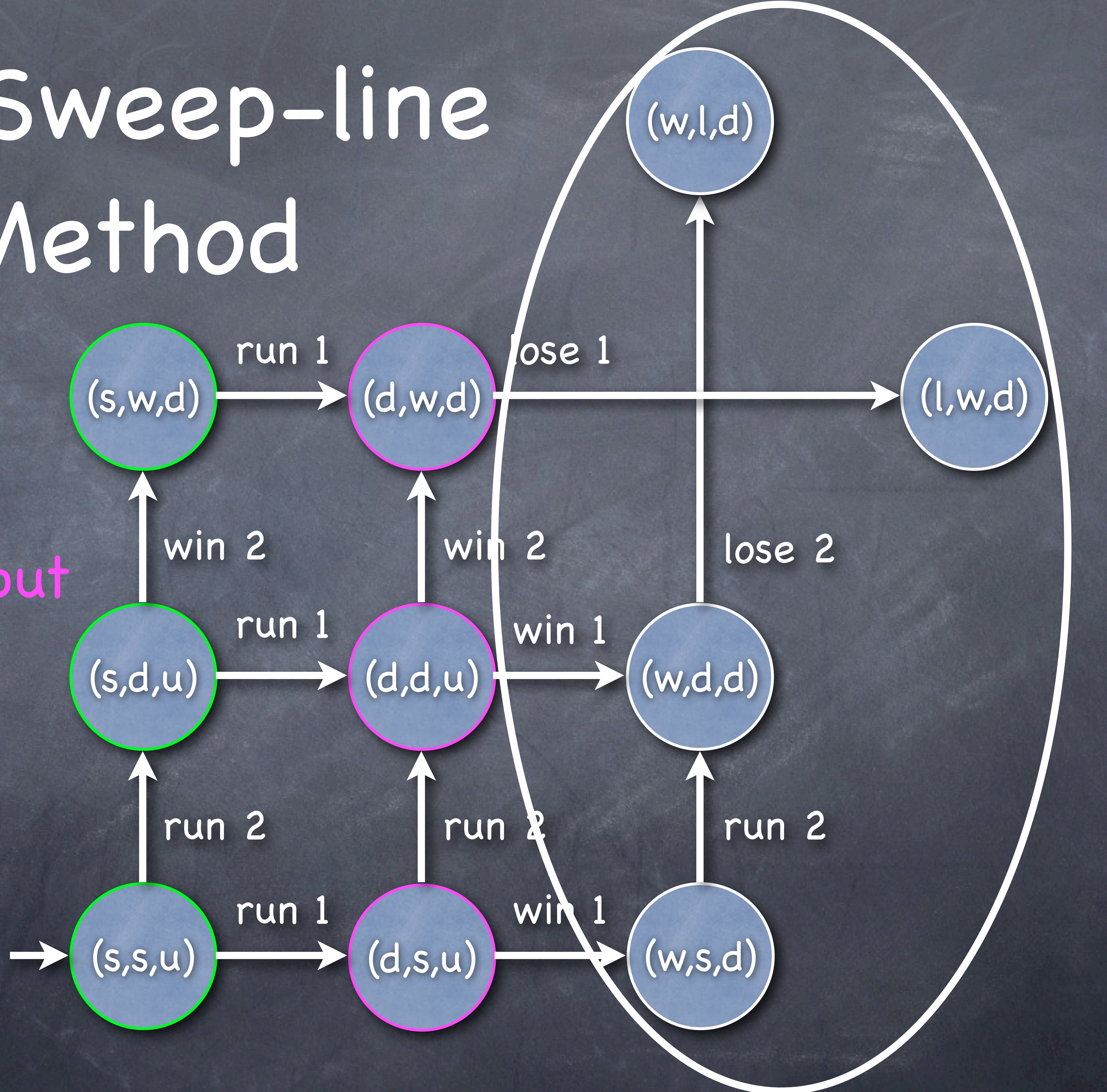


The Sweep-line Method

Already processed

Discovered but not yet processed

Not yet discovered

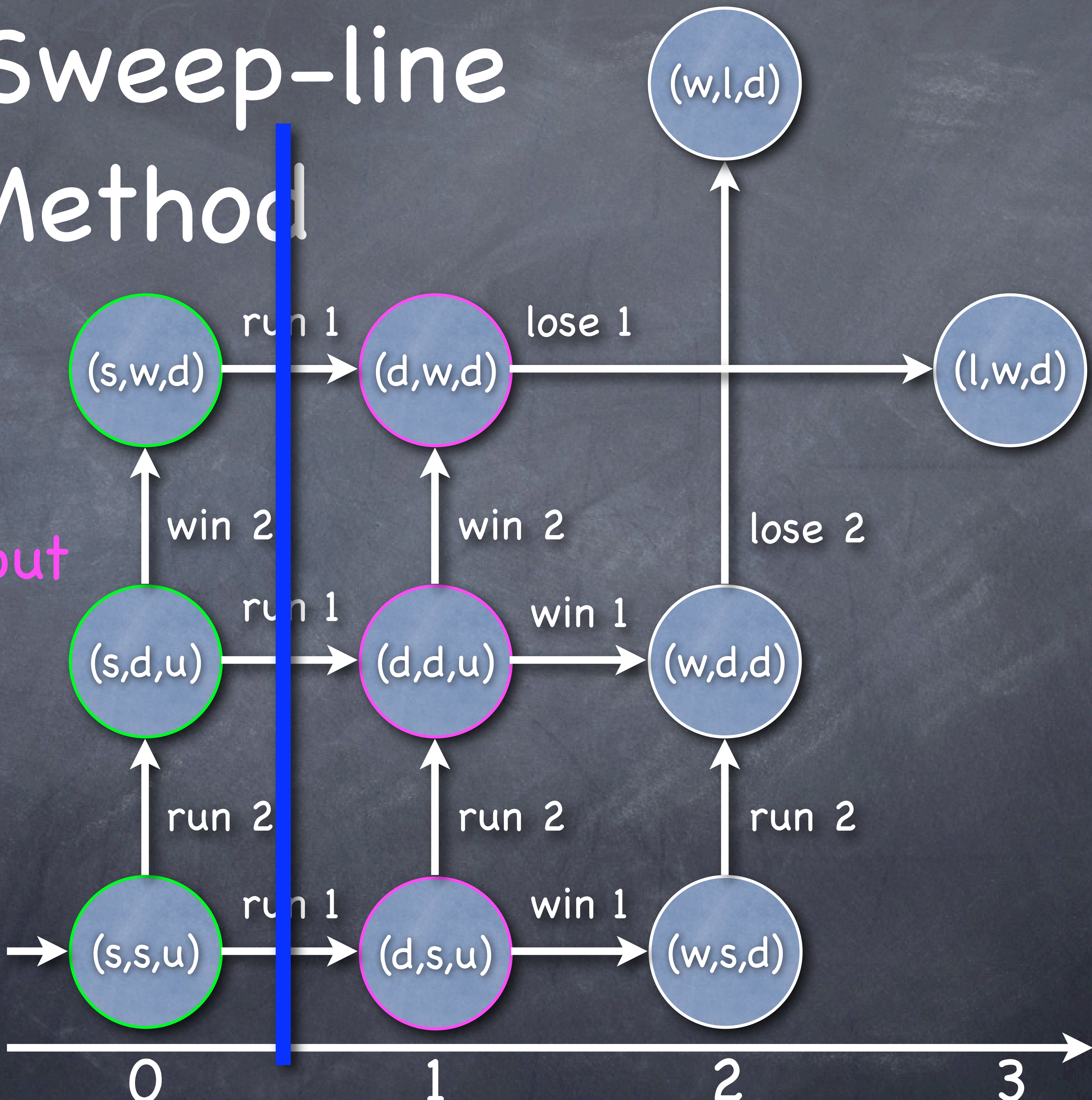


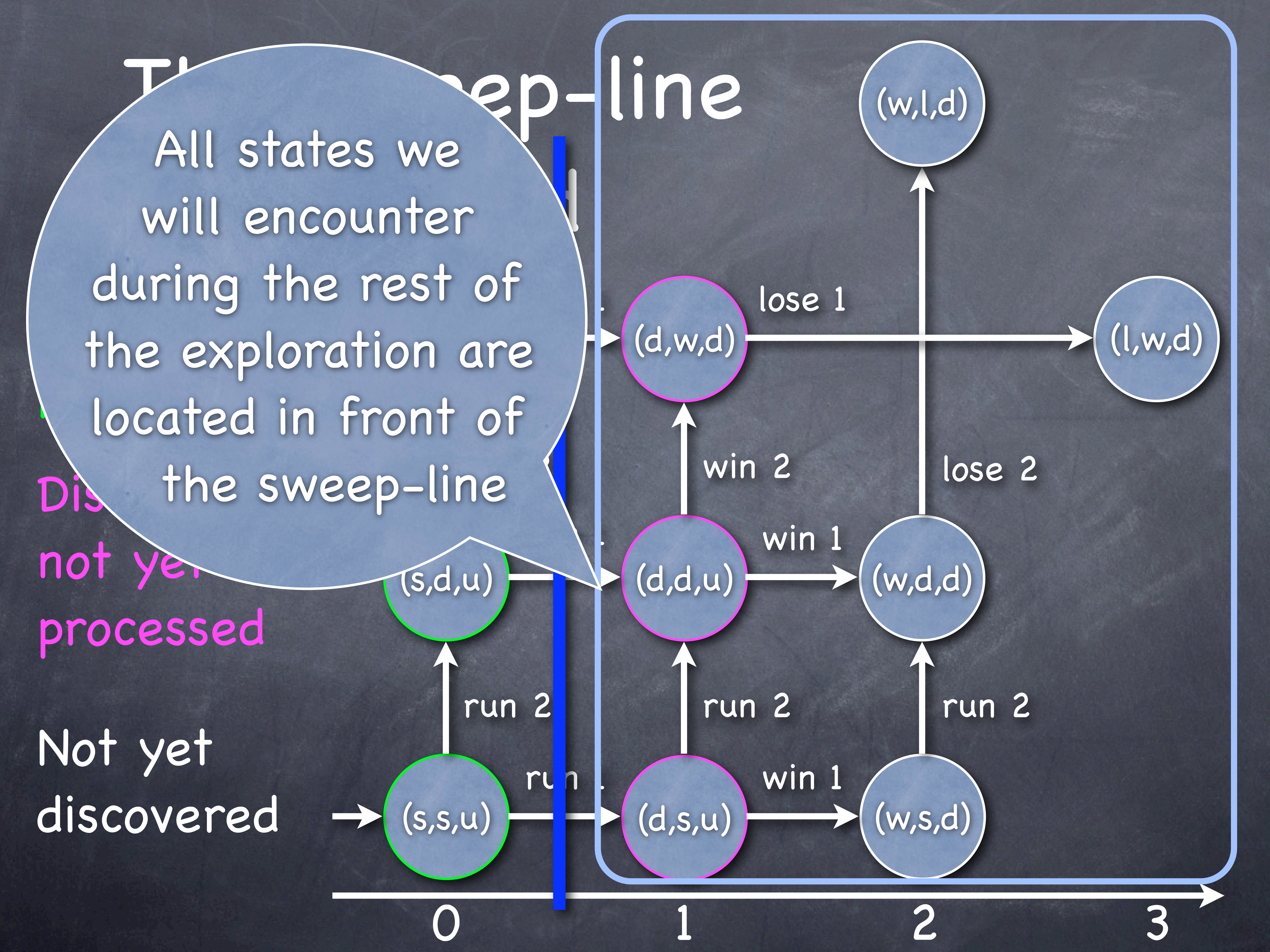
The Sweep-line Method

Already processed

Discovered but not yet processed

Not yet discovered





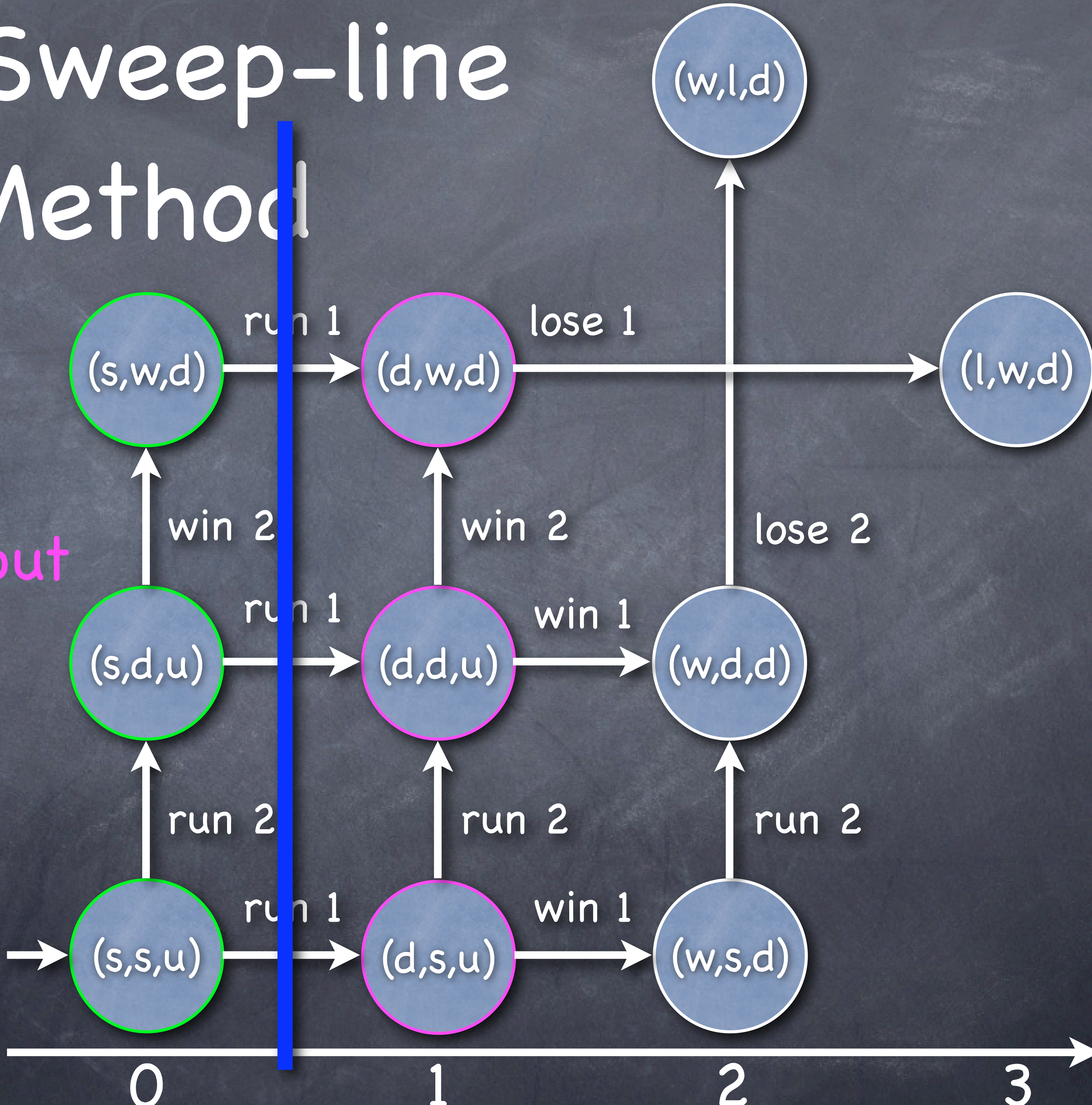
All states we will encounter during the rest of the exploration are located in front of the sweep-line
Dis
not yet
processed
Not yet
discovered
(w,l,d)
lose 1
(d,w,d)
(l,w,d)
win 2
lose 2
(s,d,u)
(d,d,u)
win 1
(w,d,d)
run 2
run 2
run 2
(s,s,u)
(d,s,u)
win 1
(w,s,d)
0
1
2
3

The Sweep-line Method

Already processed

Discovered but not yet processed

Not yet discovered

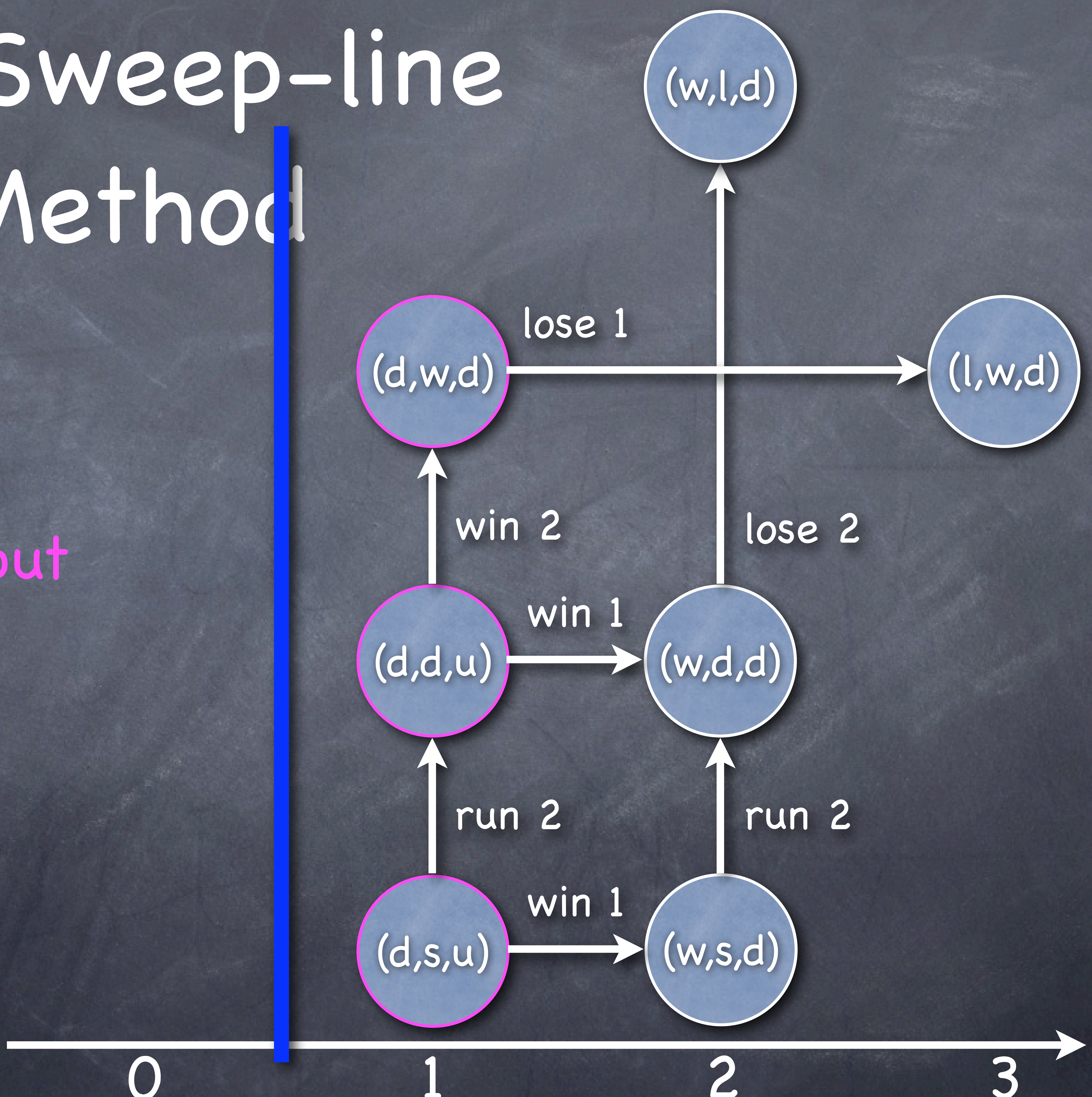


The Sweep-line Method

Already processed

Discovered but not yet processed

Not yet discovered

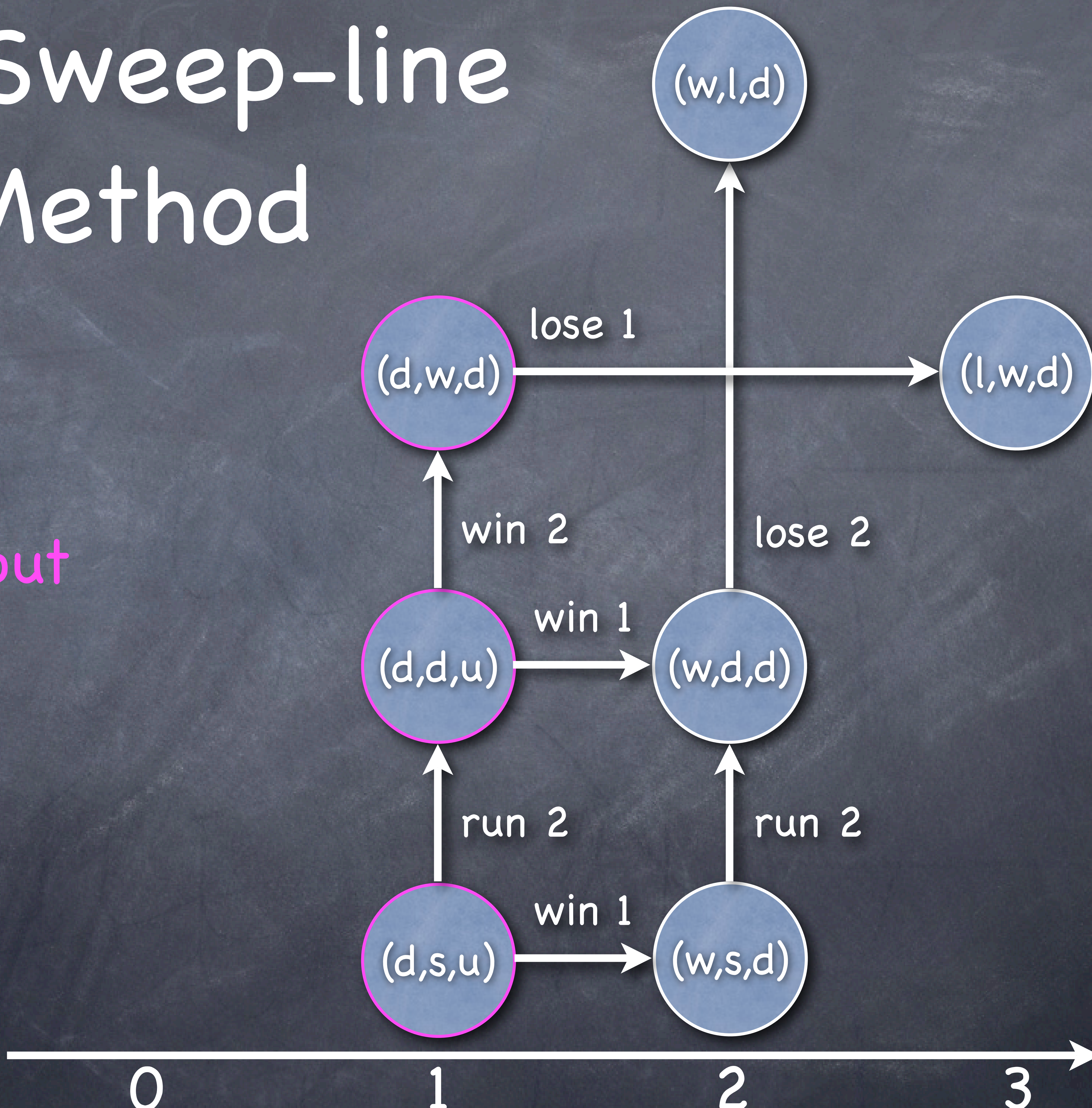


The Sweep-line Method

Already processed

Discovered but not yet processed

Not yet discovered



A Neighbor List Representation

Assume we can enumerate all transitions:

0: run 1

1: run 2

2: win 1

3: win 2

4: lose 1

5: lose 2

nbor List entation

bor List entation

Assume we can enumerate all transitions:

0: run 1

1: run 2

2: win 1

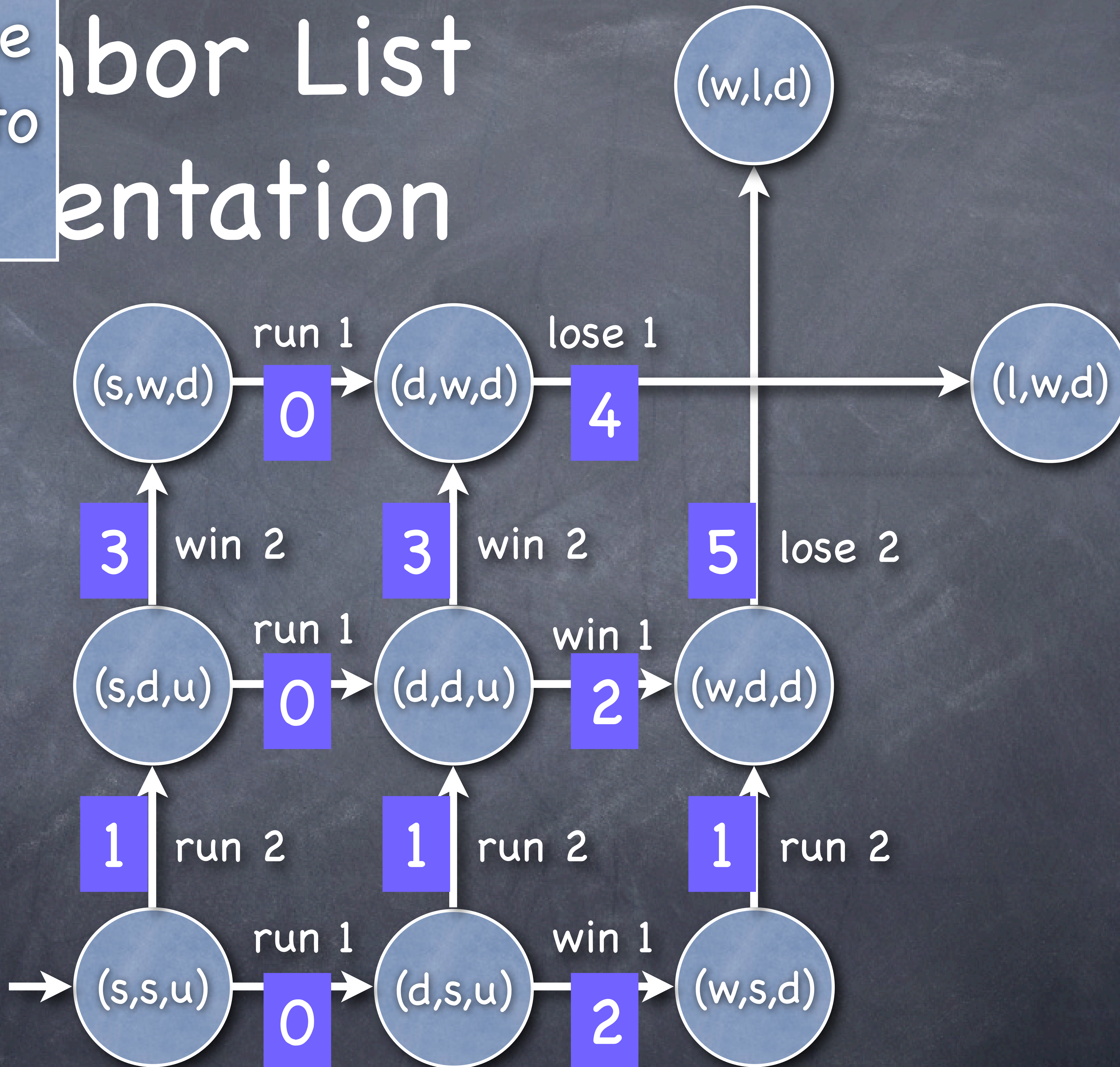
3: win 2

4: lose 1

5: lose 2

Assign a unique number, 0...9, to each state

bor List entation



Assign a unique number, 0...9, to each state

...bor List ...entation

A Neighbor List Representation

A Neighbor List Representation

0	2	(0,1)	(1,2)
1	2	(1,3)	(2,6)
2	2	(0,3)	(3,4)
3	2	(2,7)	(3,5)
4	1	(0,5)	
5	1	(4,8)	
6	1	(1,7)	
7	1	(5,9)	
8	0		
9	0		

A Neighbor List

entation

State number

0	2	(0,1) (1,2)
1	2	(1,3) (2,6)
2	2	(0,3) (3,4)
3	2	(2,7) (3,5)
4	1	(0,5)
5	1	(4,8)
6	1	(1,7)
7	1	(5,9)
8	0	
9	0	

A Neighbor List Representation

0	2	(0,1) (1,2)
1	2	[illegible]
2	2	[illegible]
3	2	(2,7) [illegible]
4	1	(0,5)
5	1	(4,8)
6	1	(1,7)
7	1	(5,9)
8	0	
9	0	

successors

A Neighbor List Representation

Transition number

successors

0	2	(0,1) (1,2)
1	2	[illegible]
2	2	[illegible]
3	2	[illegible]
4	1	(0,5)
5	1	(4,8)
6	1	(1,7)
7	1	(5,9)
8	0	
9	0	

A Neighbor List

Transition number

Successor state number

successors

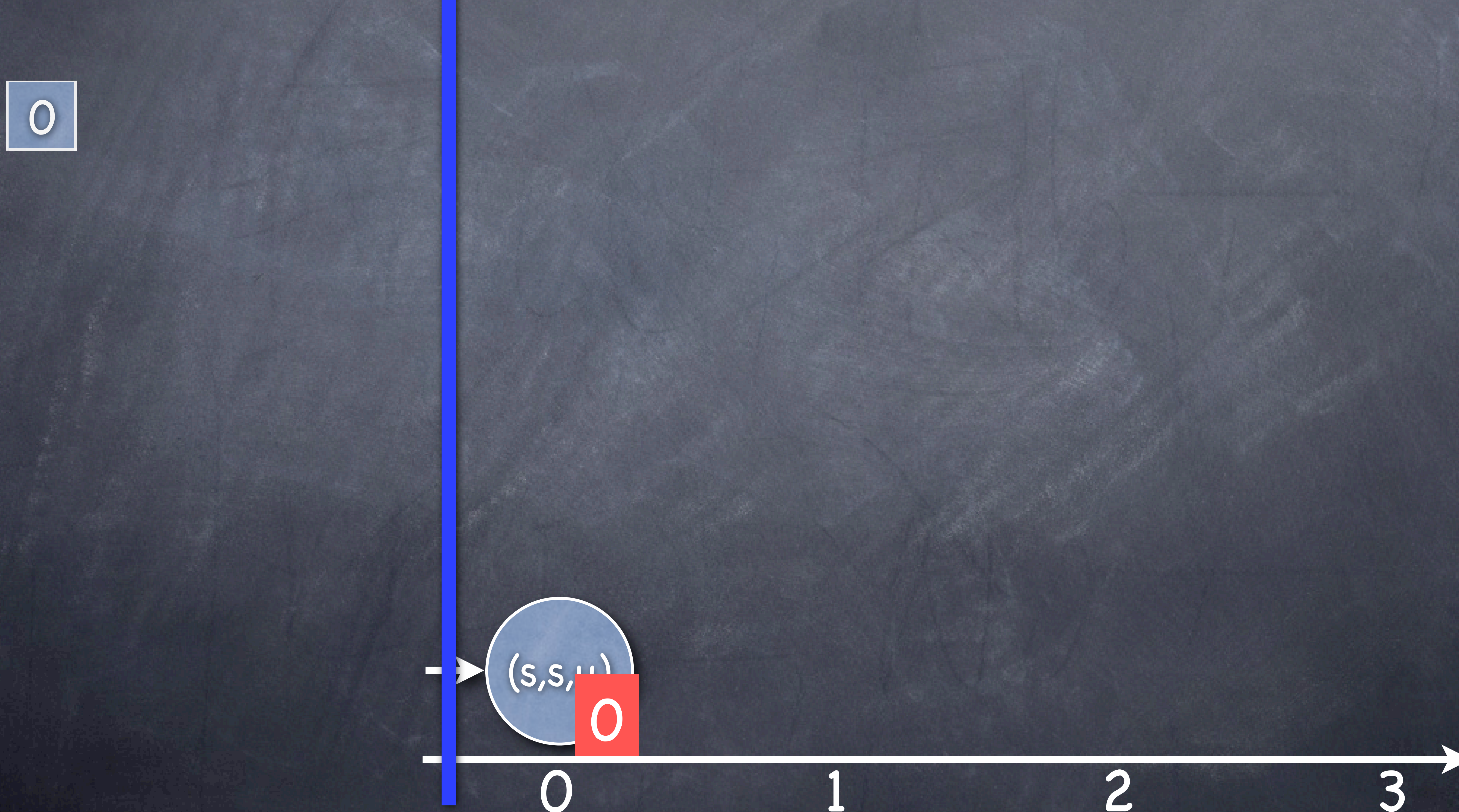
0	2	(0,1) (1,2)
1	2	
2	2	
3	2	(2,7)
4	1	(0,5)
5	1	(4,8)
6	1	(1,7)
7	1	(5,9)
8	0	
9	0	

Building the Condensed Representation

Building the Condensed Representation

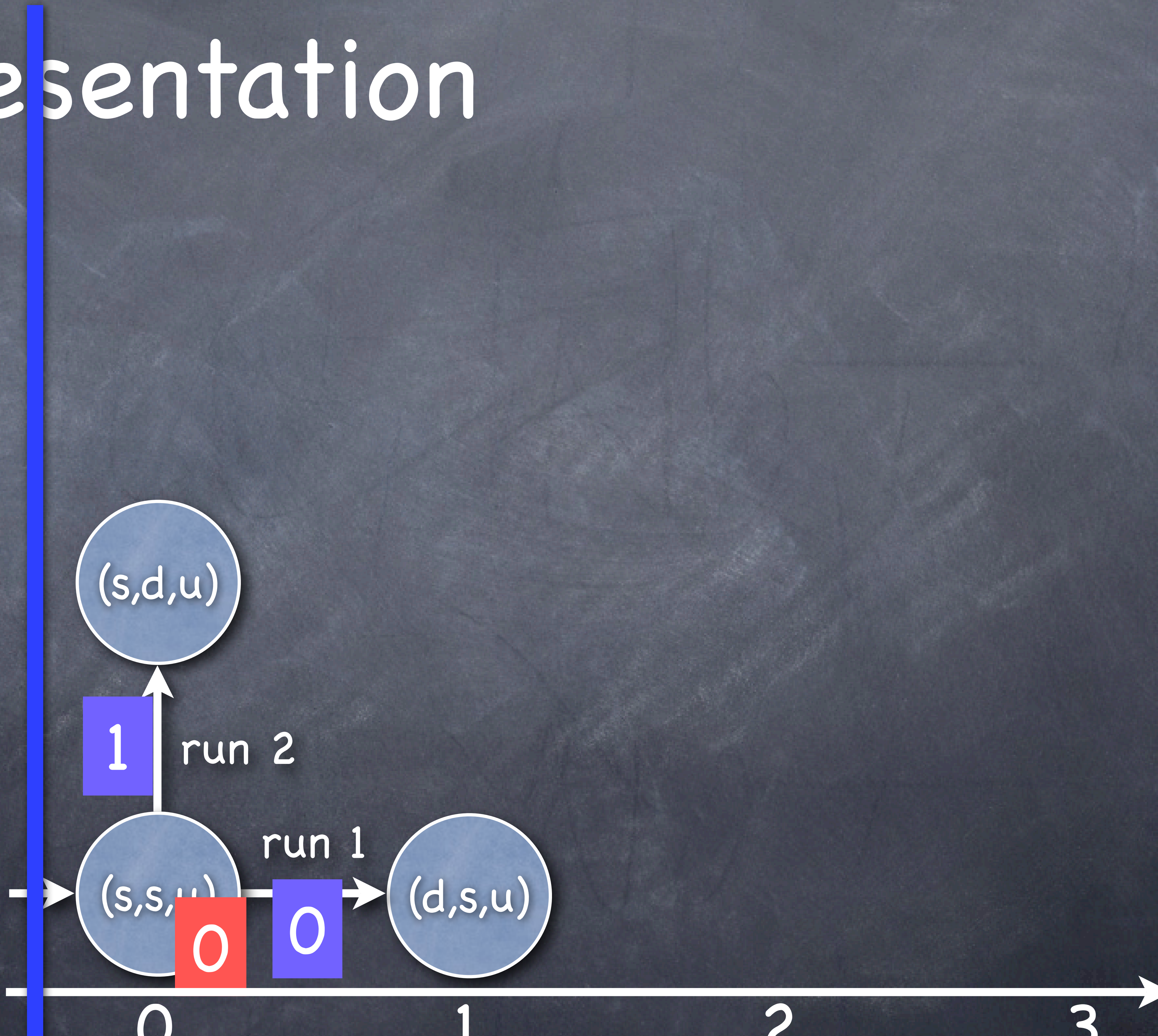
Building the Condensed Representation

Building the Condensed Representation

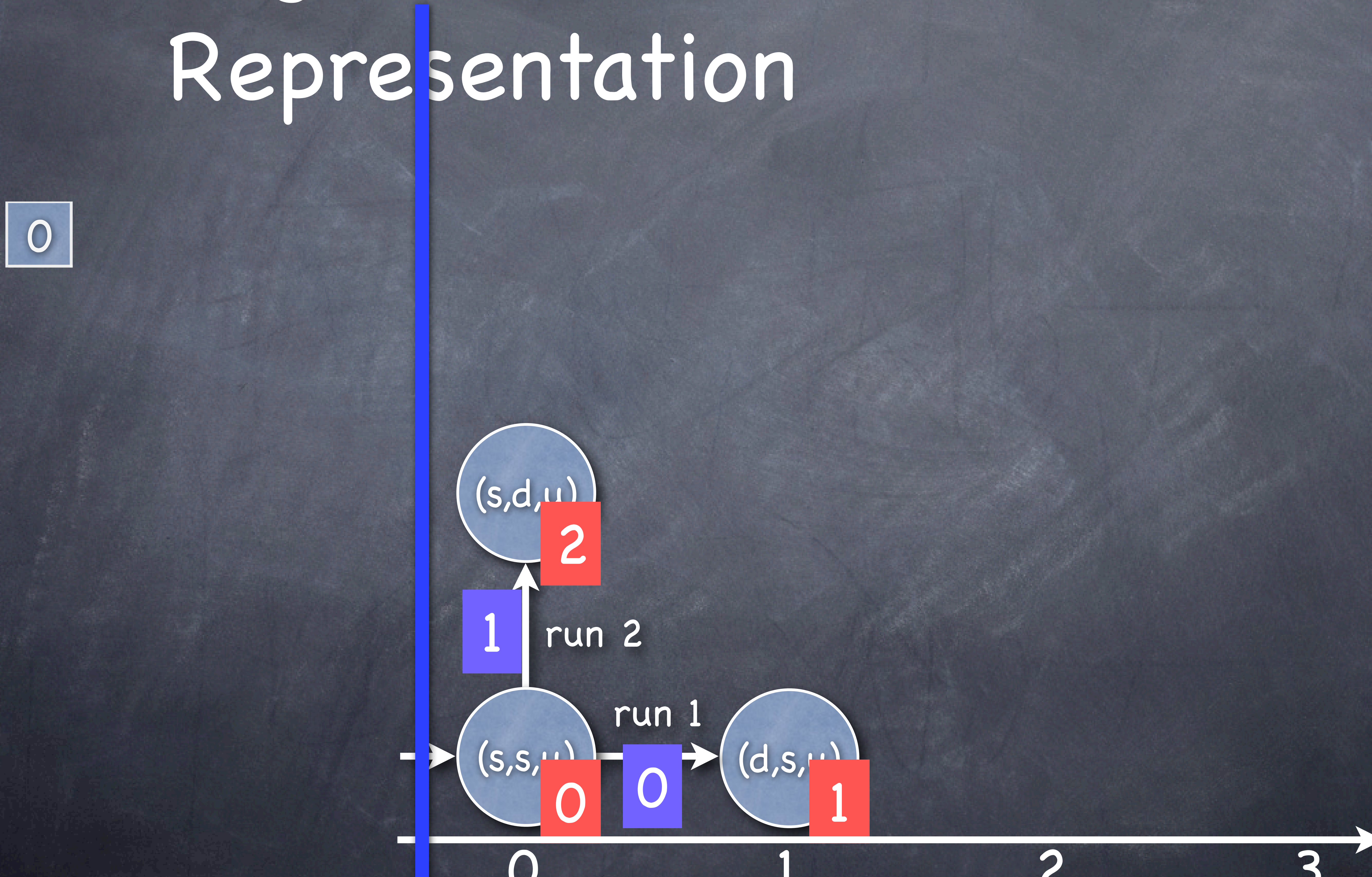


Building the Condensed Representation

0

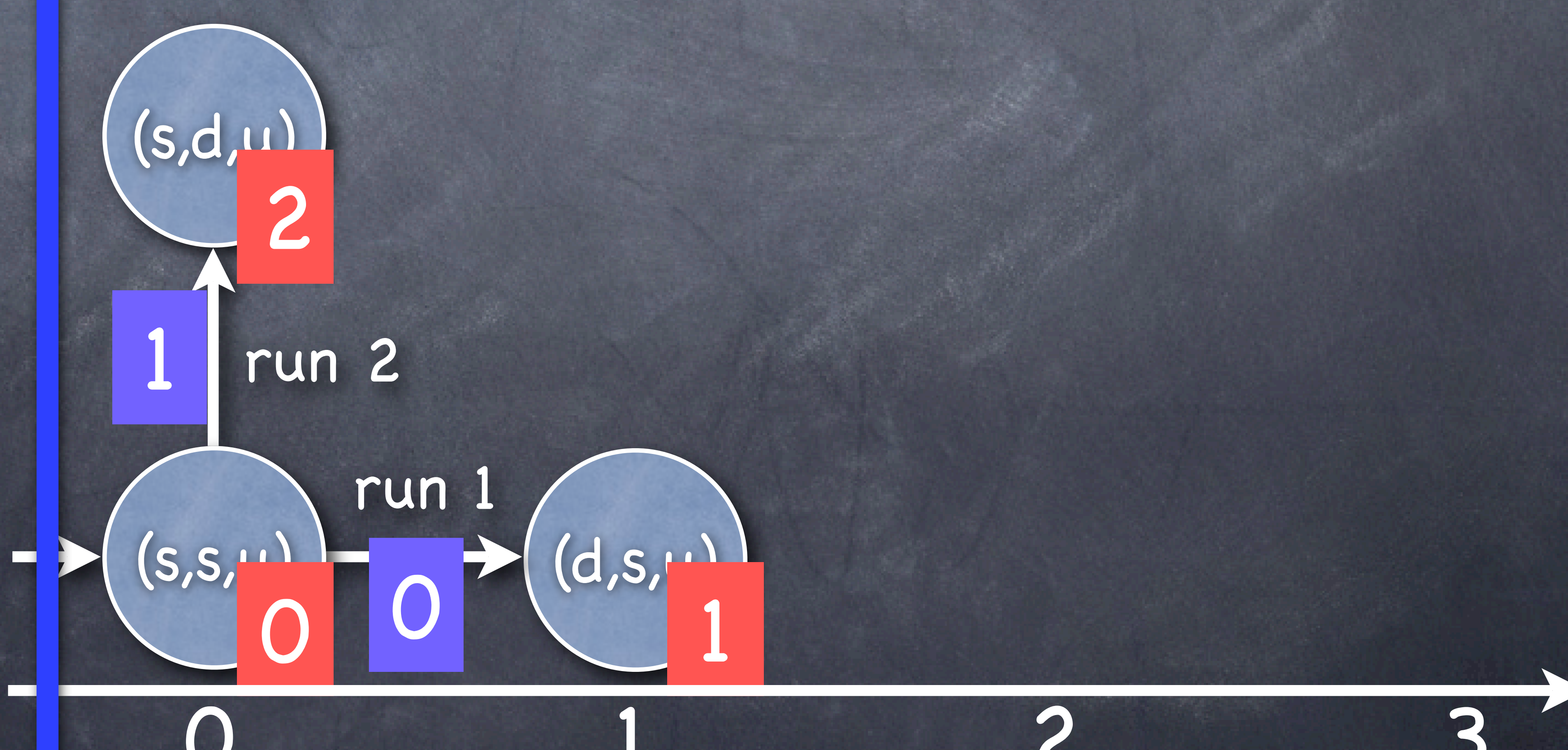


Building the Condensed Representation



Building the Condensed Representation

0	2	2	(0,1) (1,2)
1			
2			



Buildin… d
R…

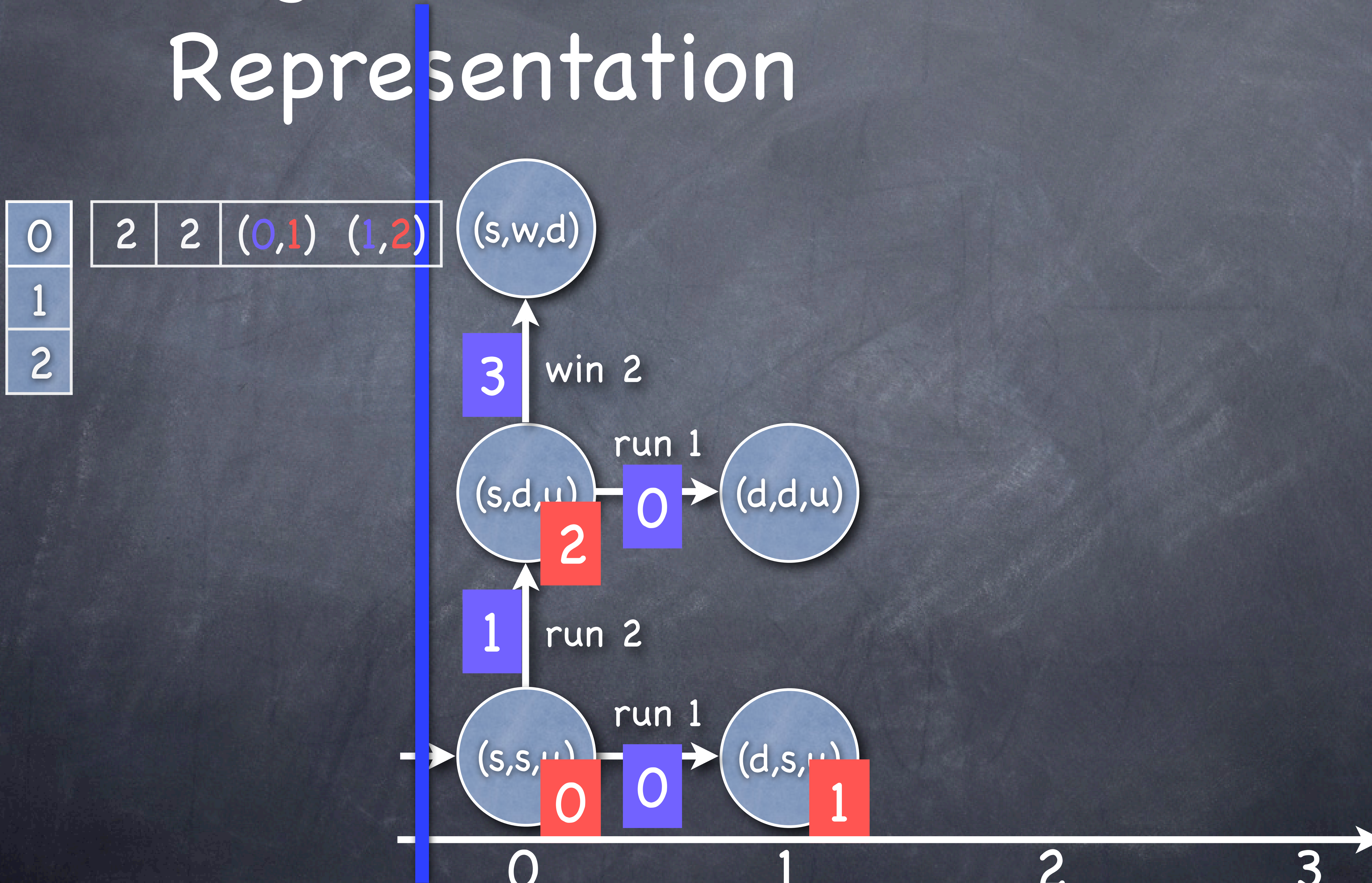
New header:
bits used to store each successor

0	2	2	(0,1) (1,2)
1			
2			

Building the Condensed Representation

0	2	2	(0,1) (1,2)
1			
2			

Building the Condensed Representation



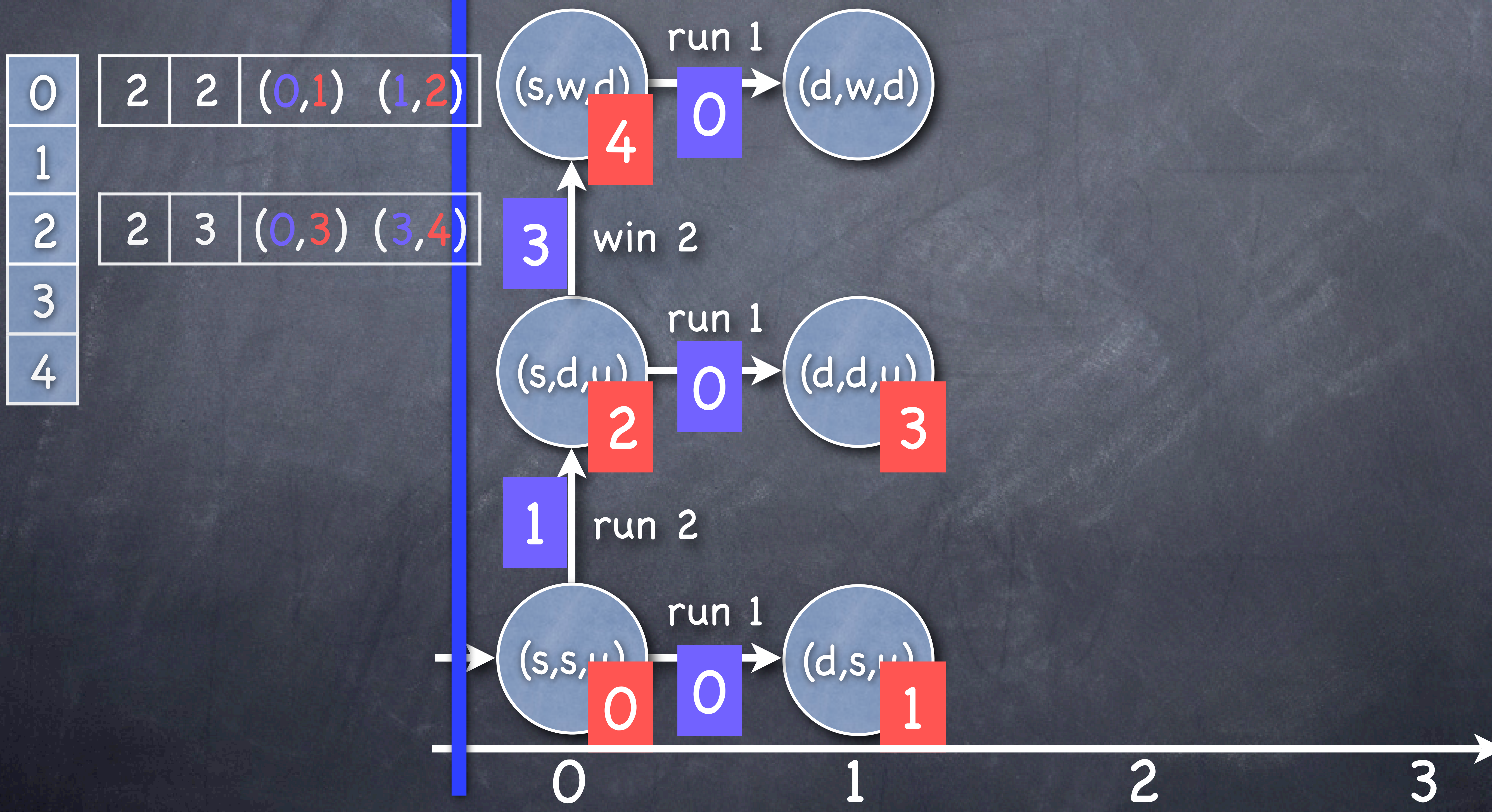
Building the Condensed Representation

0	2	2	(0,1) (1,2)
1			
2			

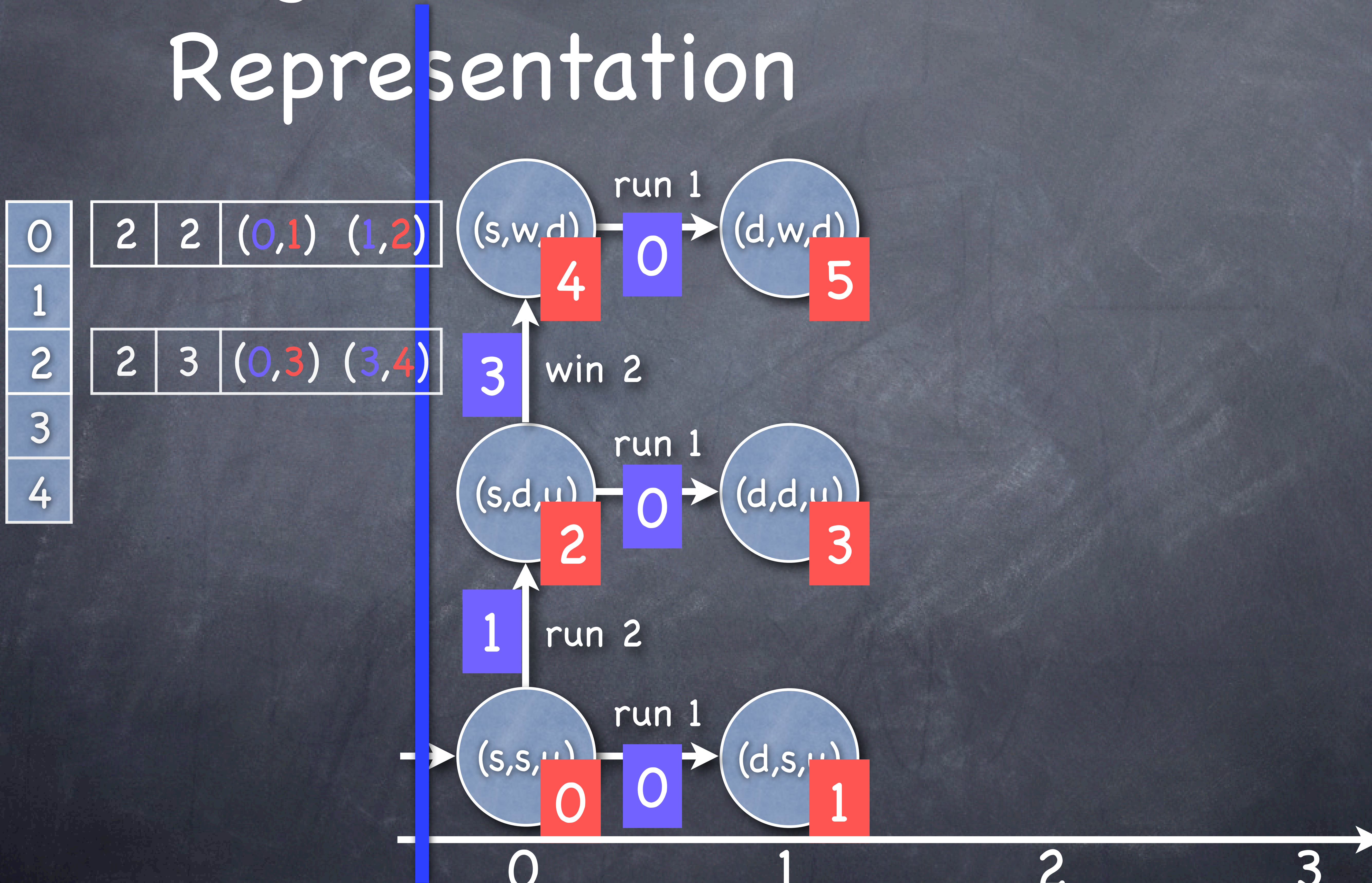
Building the Condensed Representation

0	2	2	(0,1) (1,2)
1			
2	2	3	(0,3) (3,4)
3			
4			

Building the Condensed Representation



Building the Condensed Representation

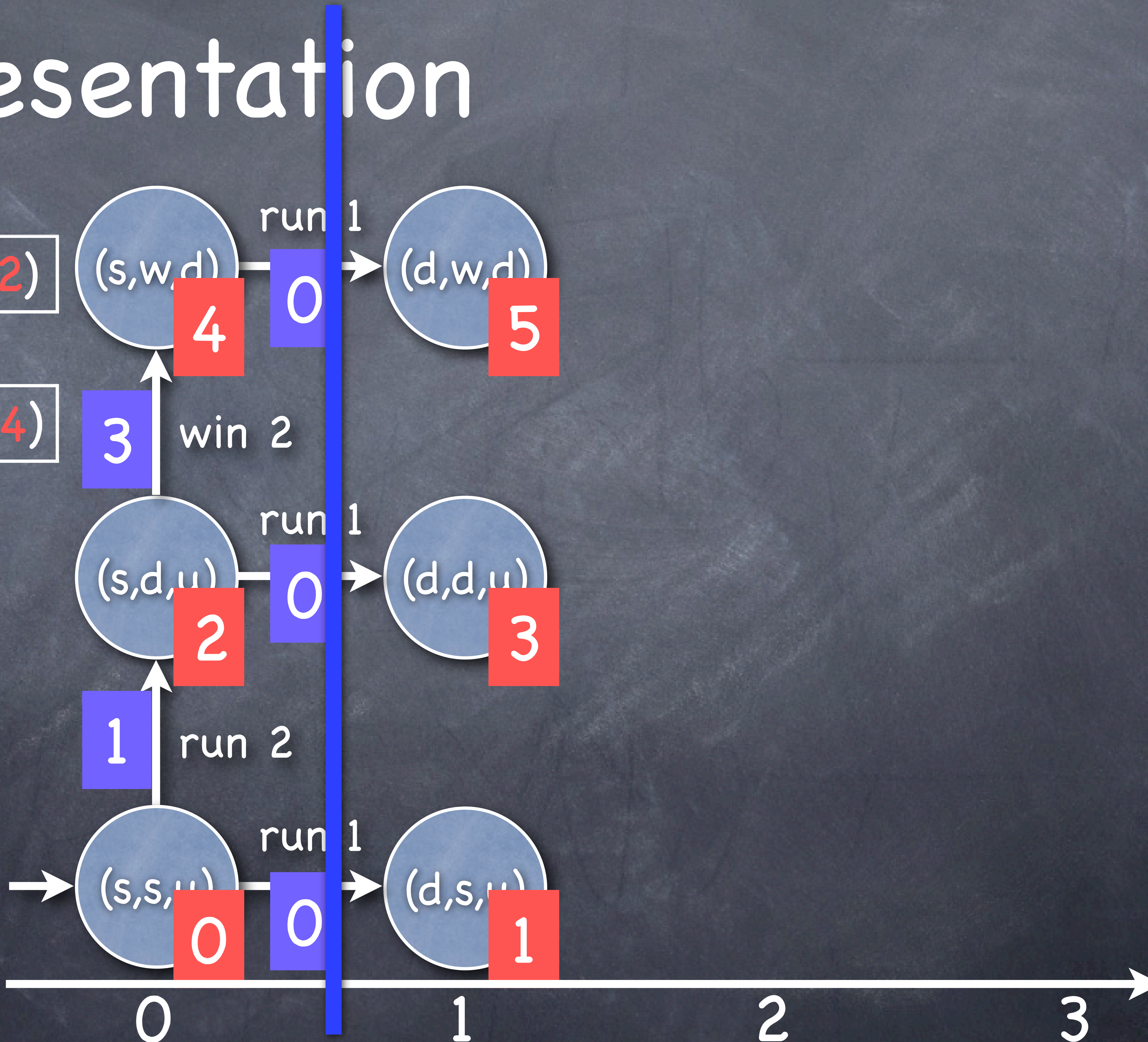


Building the Condensed Representation

0	2	2	(0,1) (1,2)
1			
2	2	3	(0,3) (3,4)
3			
4	1	3	(0,5)
5			

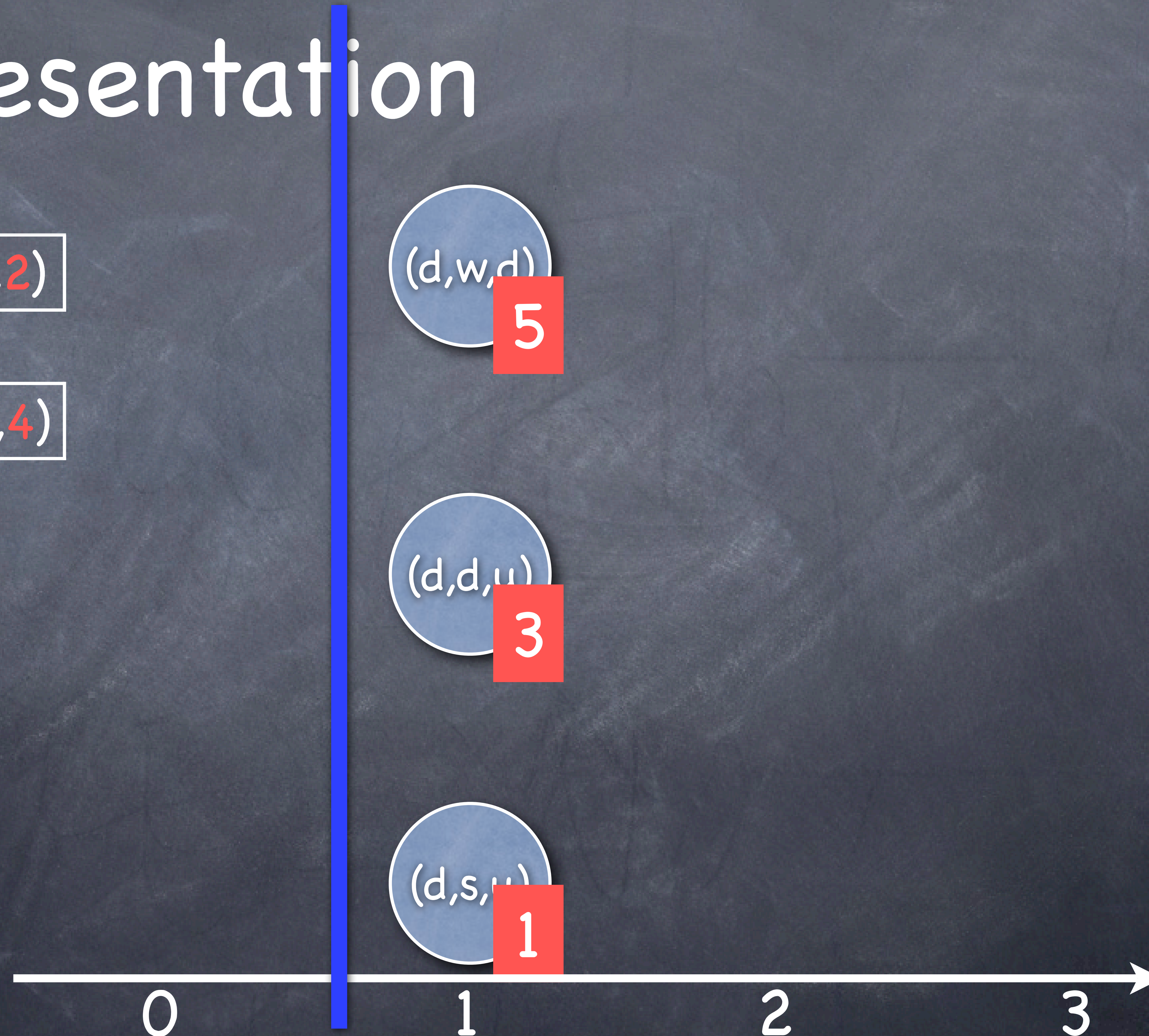
Building the Condensed Representation

0	2	2	(0,1) (1,2)
1			
2	2	3	(0,3) (3,4)
3			
4	1	3	(0,5)
5			



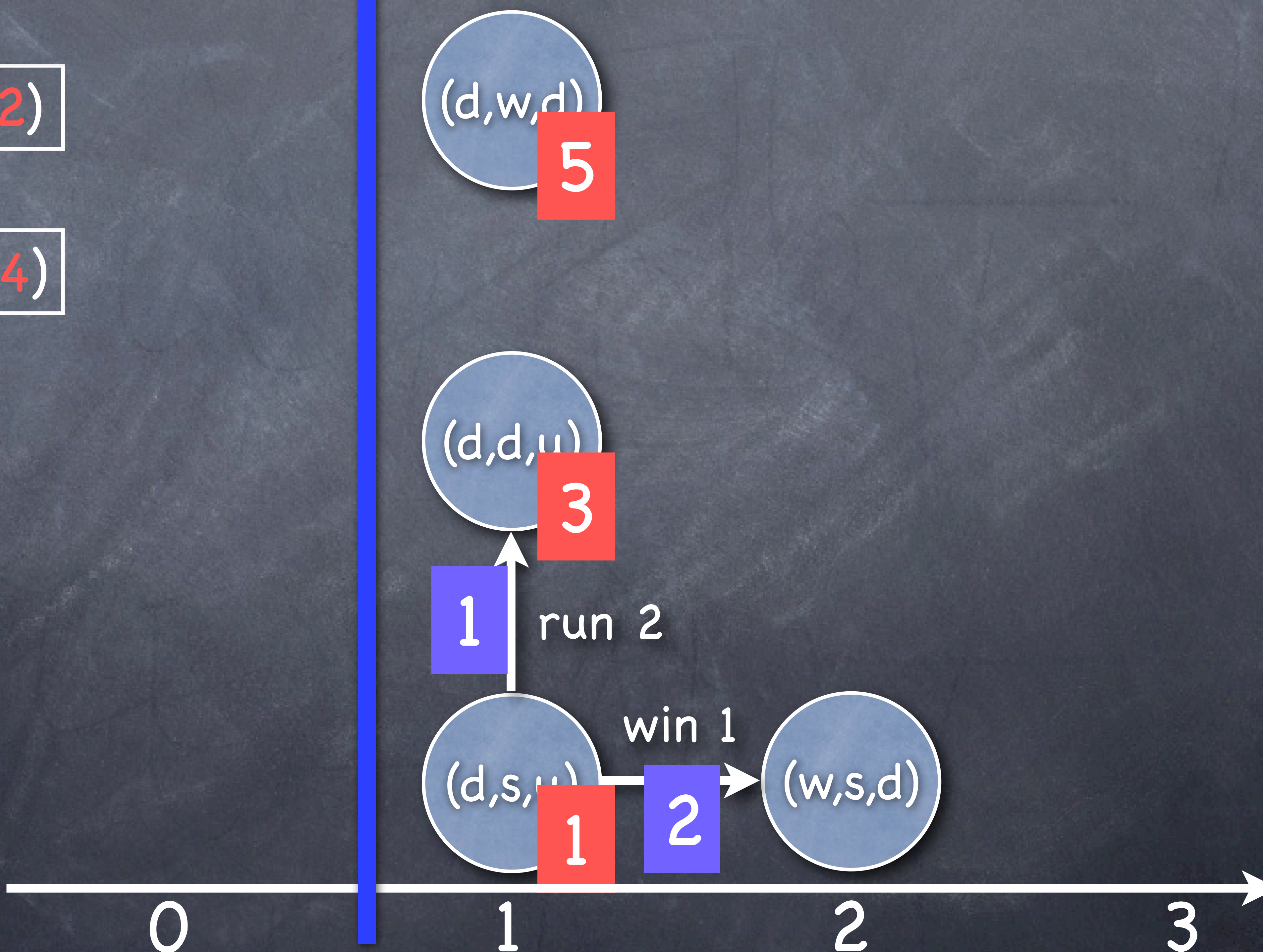
Building the Condensed Representation

0	2	2	(0,1) (1,2)
1			
2	2	3	(0,3) (3,4)
3			
4	1	3	(0,5)
5			



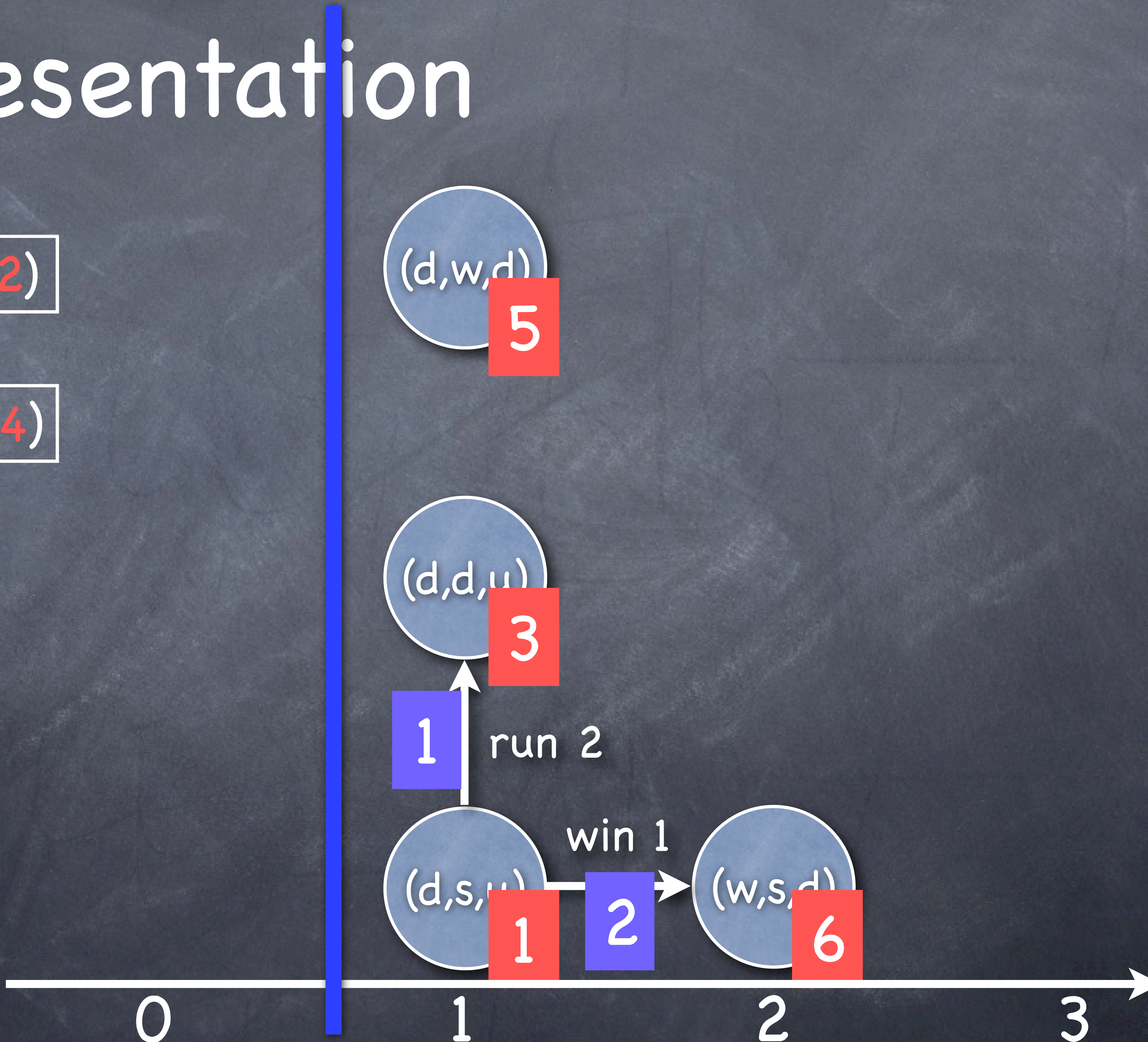
Building the Condensed Representation

0	2	2	(0,1) (1,2)
1			
2	2	3	(0,3) (3,4)
3			
4	1	3	(0,5)
5			



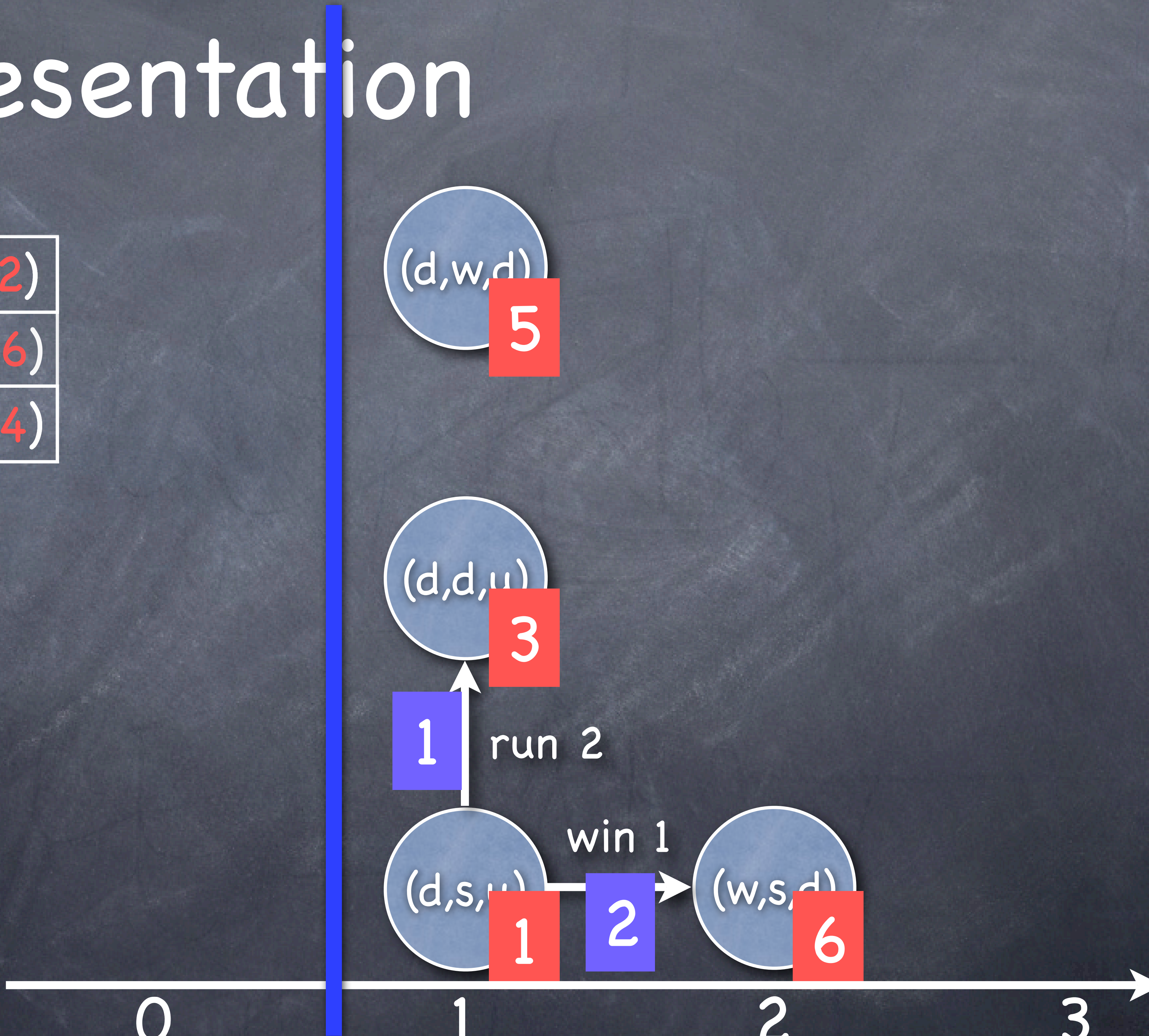
Building the Condensed Representation

0	2	2	(0,1) (1,2)
1			
2	2	3	(0,3) (3,4)
3			
4	1	3	(0,5)
5			



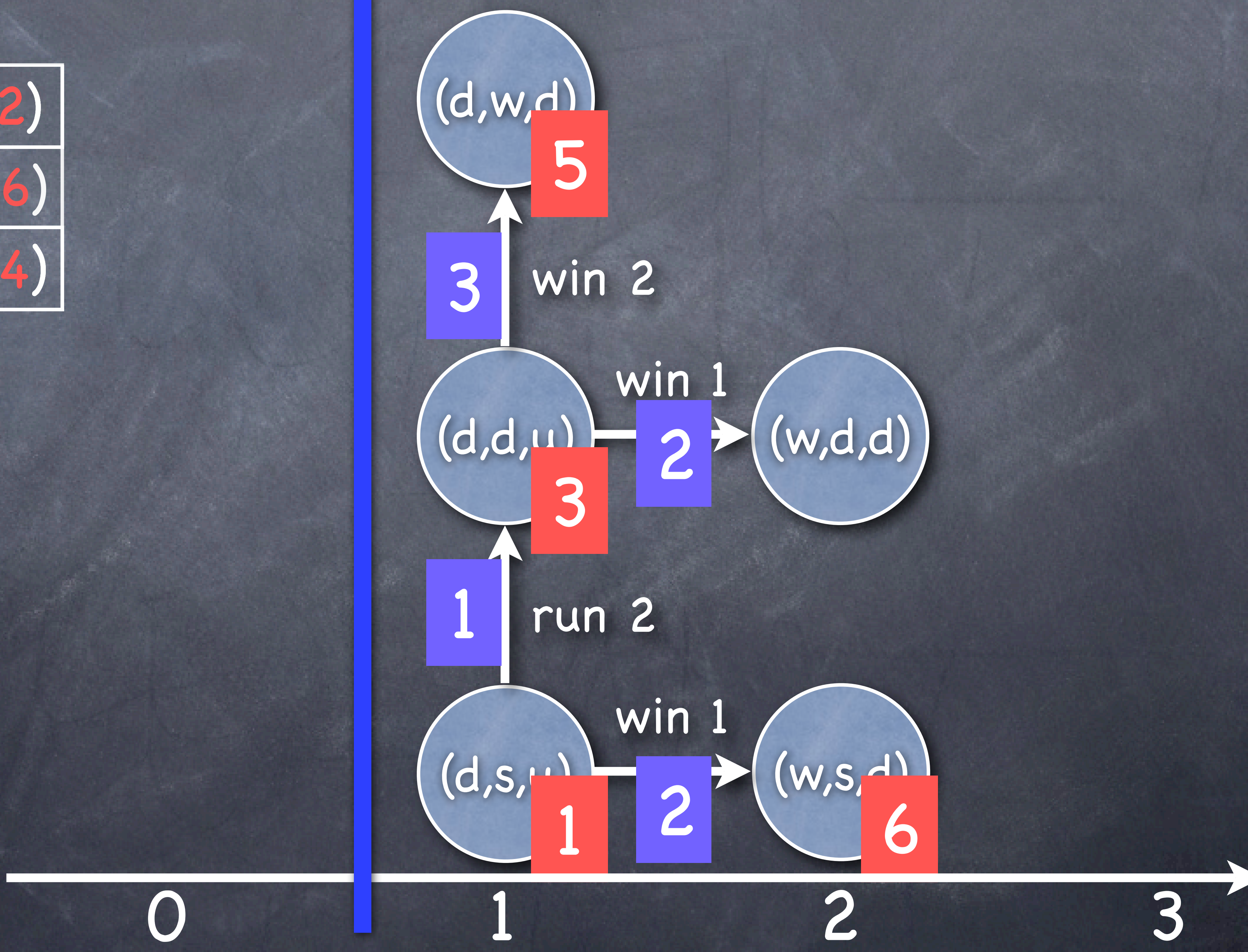
Building the Condensed Representation

0	2	2	(0,1) (1,2)
1	2	3	(1,3) (2,6)
2	2	3	(0,3) (3,4)
3			
4	1	3	(0,5)
5			
6			



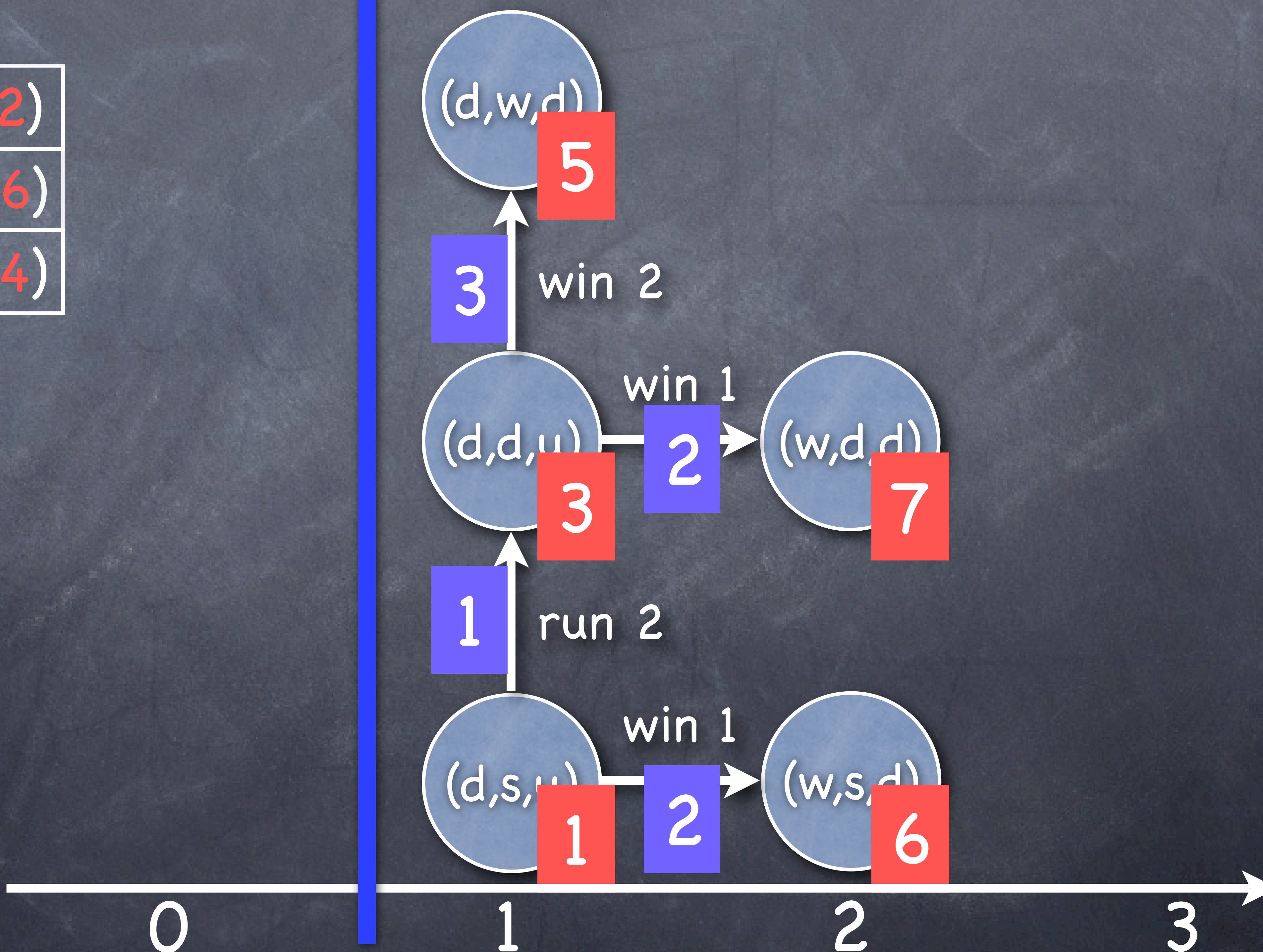
Building the Condensed Representation

0	2	2	(0,1) (1,2)
1	2	3	(1,3) (2,6)
2	2	3	(0,3) (3,4)
3			
4	1	3	(0,5)
5			
6			



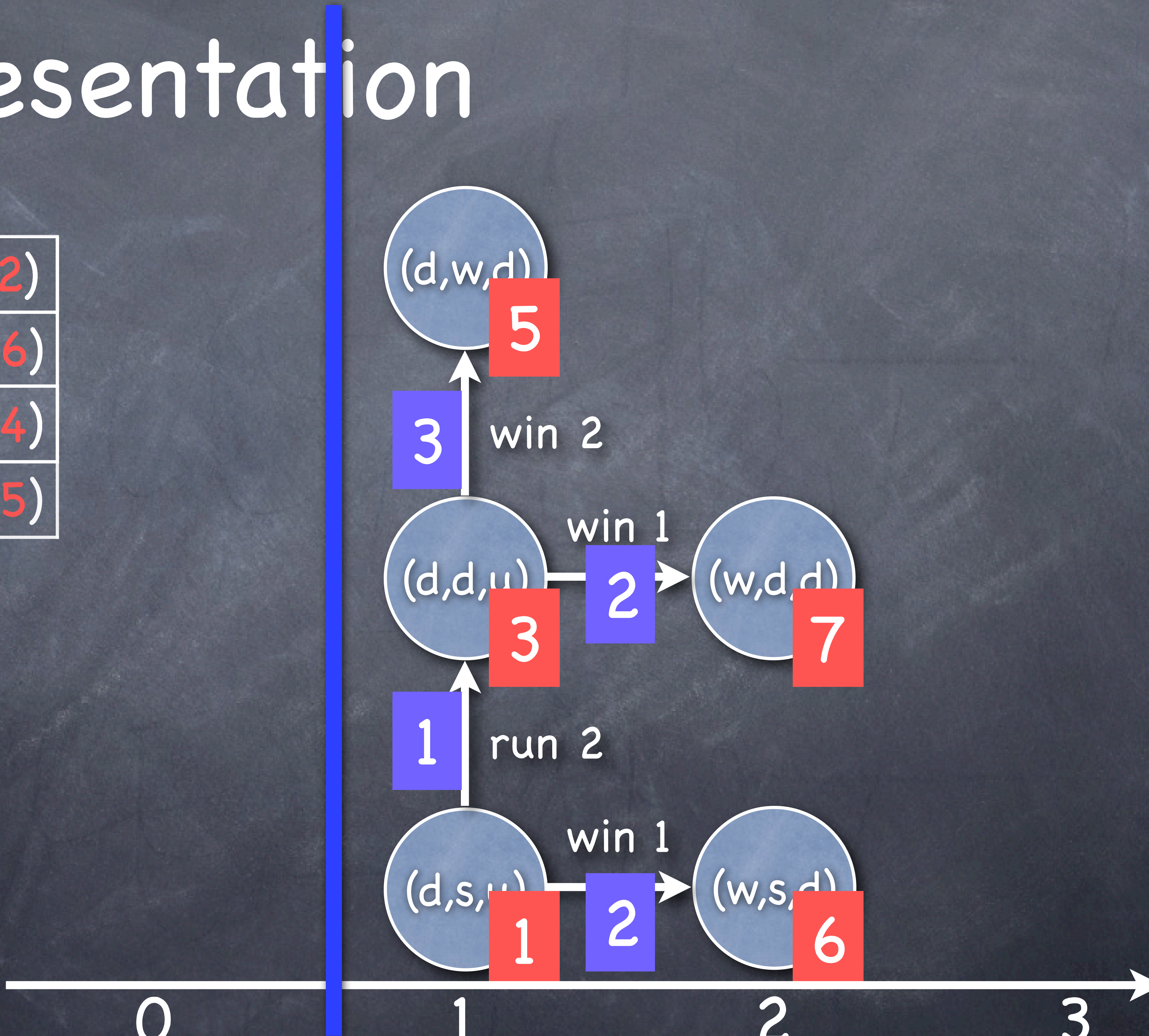
Building the Condensed Representation

0	2	2	(0,1) (1,2)
1	2	3	(1,3) (2,6)
2	2	3	(0,3) (3,4)
3			
4	1	3	(0,5)
5			
6			



Building the Condensed Representation

0	2	2	(0,1) (1,2)
1	2	3	(1,3) (2,6)
2	2	3	(0,3) (3,4)
3	2	3	(2,7) (3,5)
4	1	3	(0,5)
5			
6			
7			

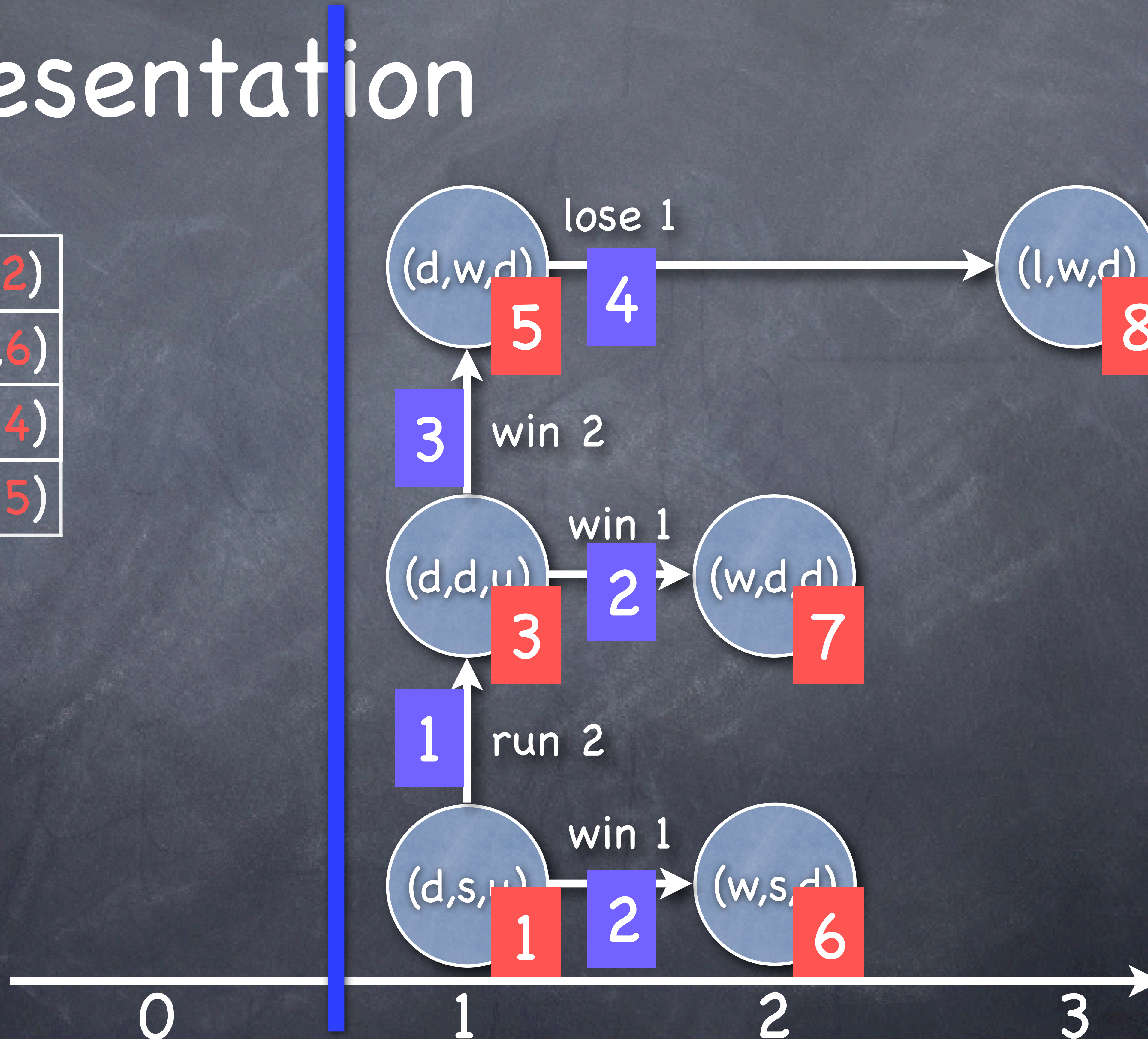


Building the Condensed Representation

0	2	2	(0,1)	(1,2)
1	2	3	(1,3)	(2,6)
2	2	3	(0,3)	(3,4)
3	2	3	(2,7)	(3,5)
4	1	3	(0,5)	
5				
6				
7				

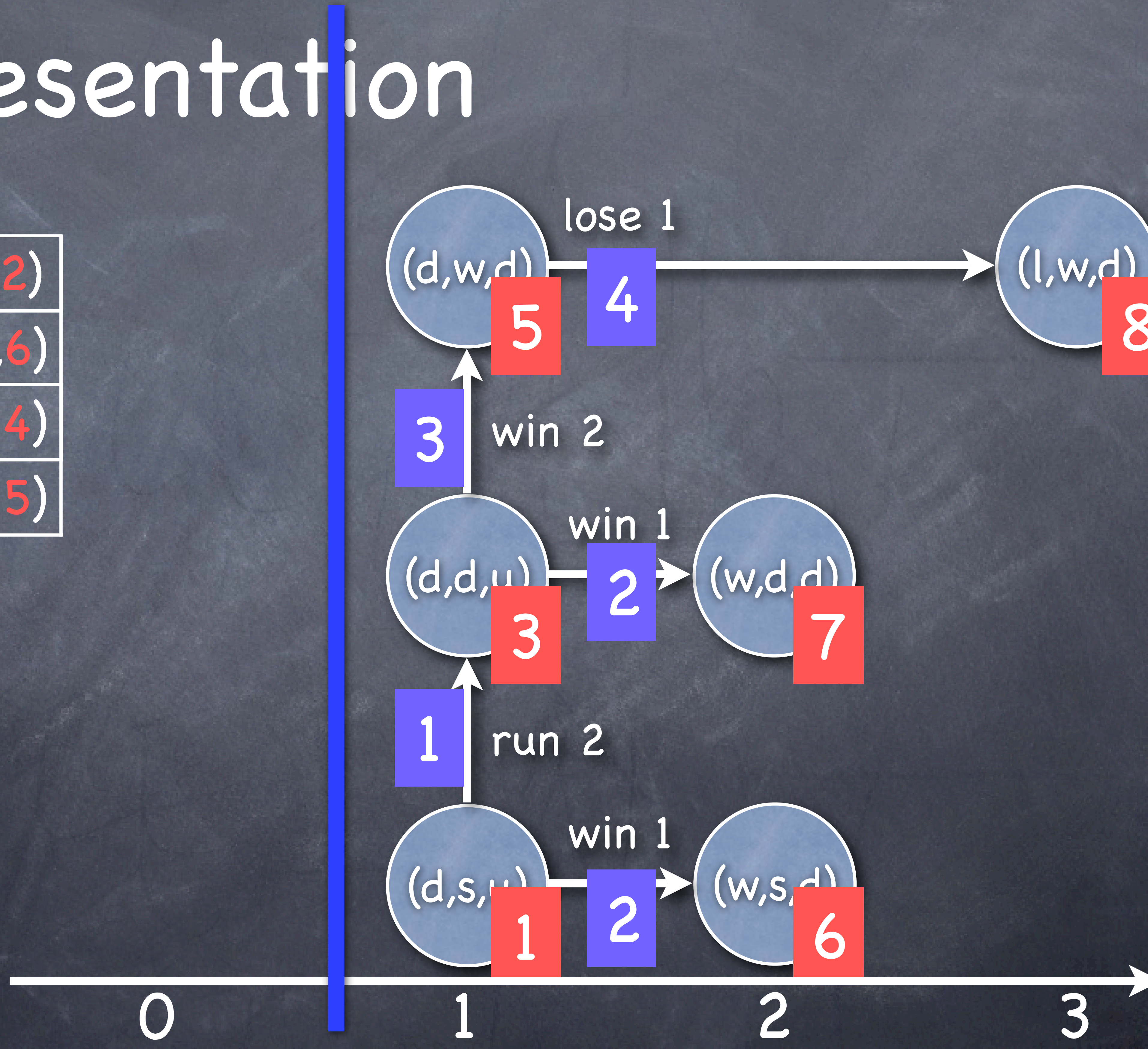
Building the Condensed Representation

0	2	2	(0,1)	(1,2)
1	2	3	(1,3)	(2,6)
2	2	3	(0,3)	(3,4)
3	2	3	(2,7)	(3,5)
4	1	3	(0,5)	
5				
6				
7				



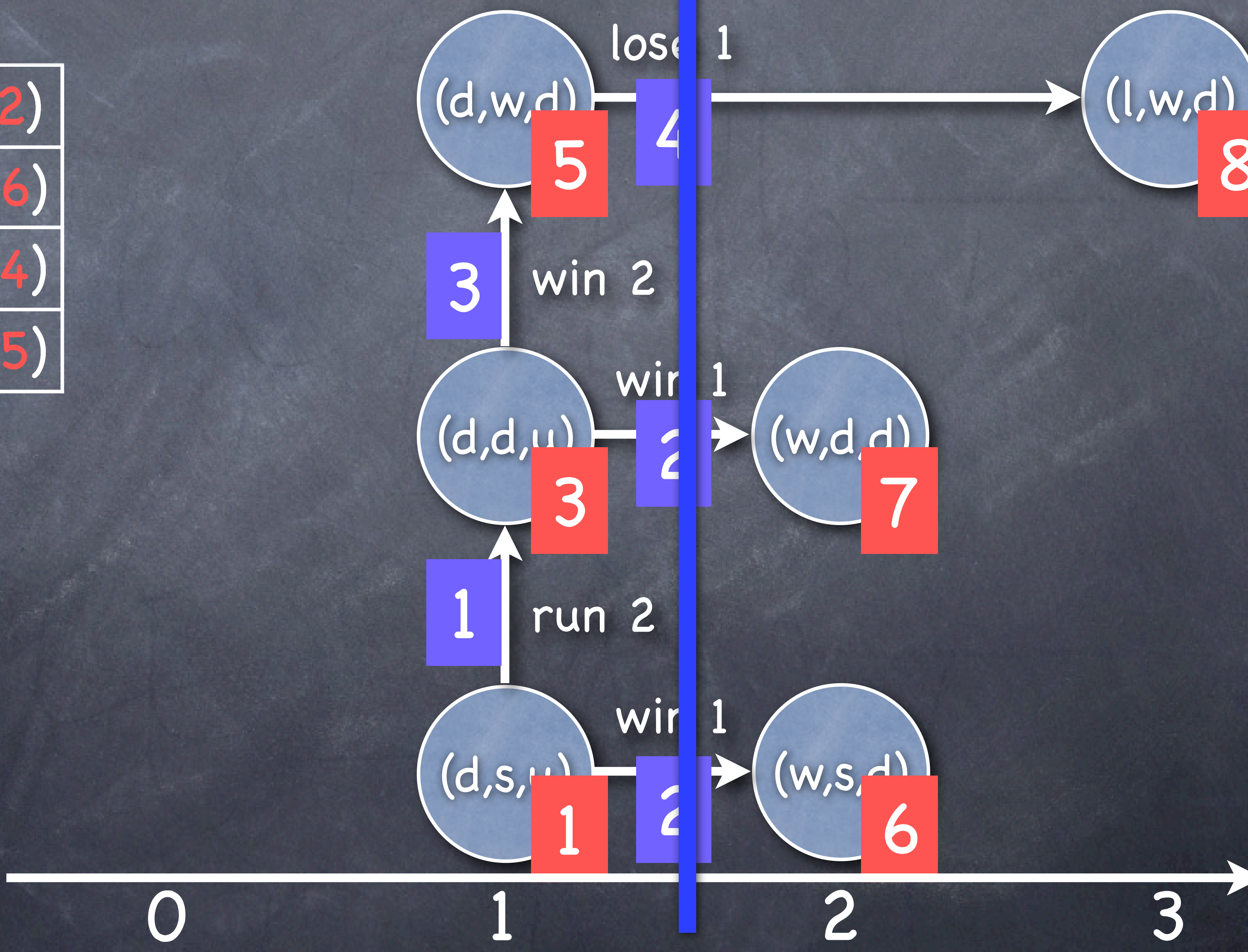
Building the Condensed Representation

0	2	2	(0,1) (1,2)
1	2	3	(1,3) (2,6)
2	2	3	(0,3) (3,4)
3	2	3	(2,7) (3,5)
4	1	3	(0,5)
5	1	4	(4,8)
6			
7			
8			



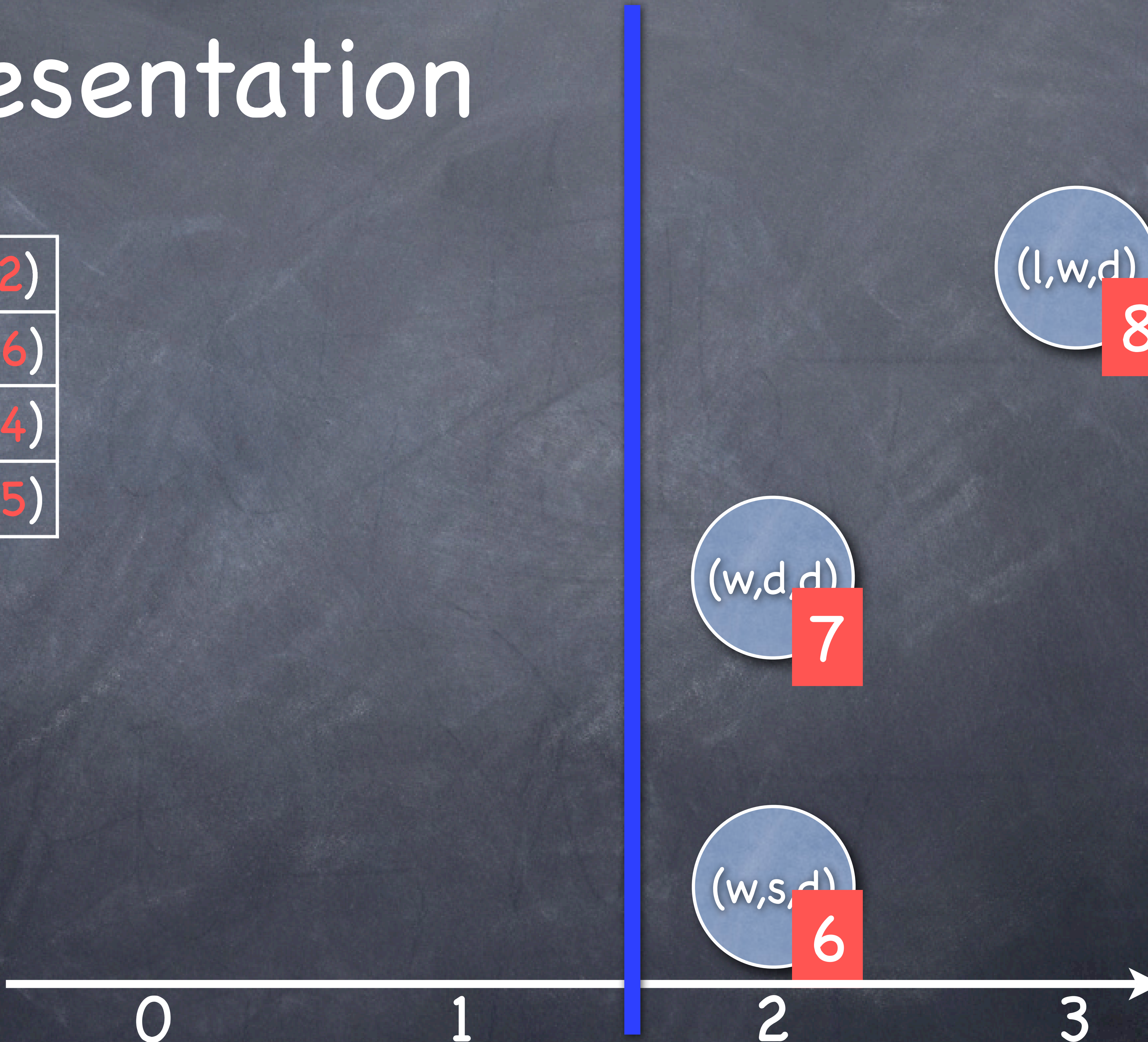
Building the Condensed Representation

0	2	2	(0,1) (1,2)
1	2	3	(1,3) (2,6)
2	2	3	(0,3) (3,4)
3	2	3	(2,7) (3,5)
4	1	3	(0,5)
5	1	4	(4,8)
6			
7			
8			



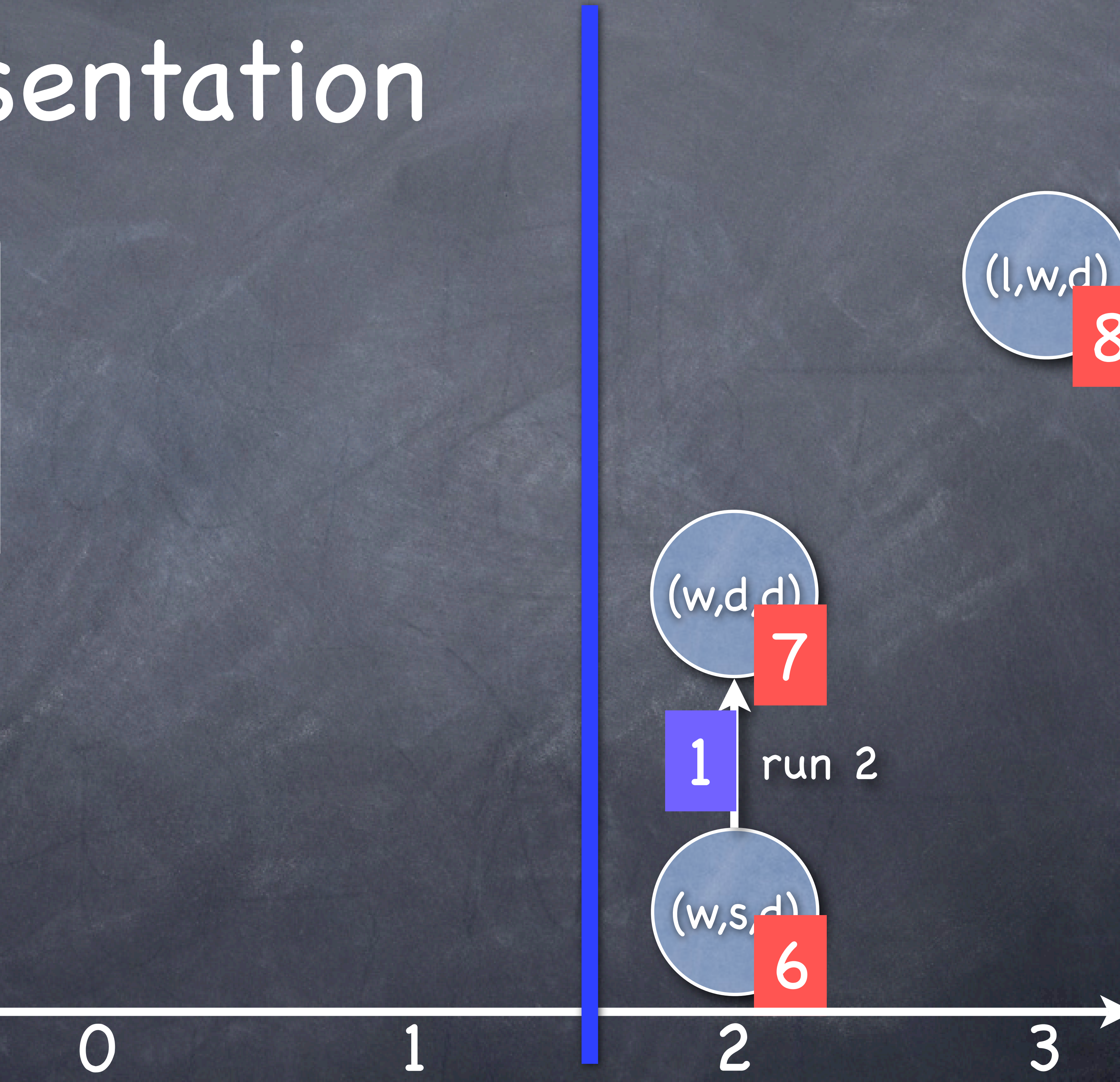
Building the Condensed Representation

0	2	2	(0,1)	(1,2)
1	2	3	(1,3)	(2,6)
2	2	3	(0,3)	(3,4)
3	2	3	(2,7)	(3,5)
4	1	3	(0,5)	
5	1	4	(4,8)	
6				
7				
8				



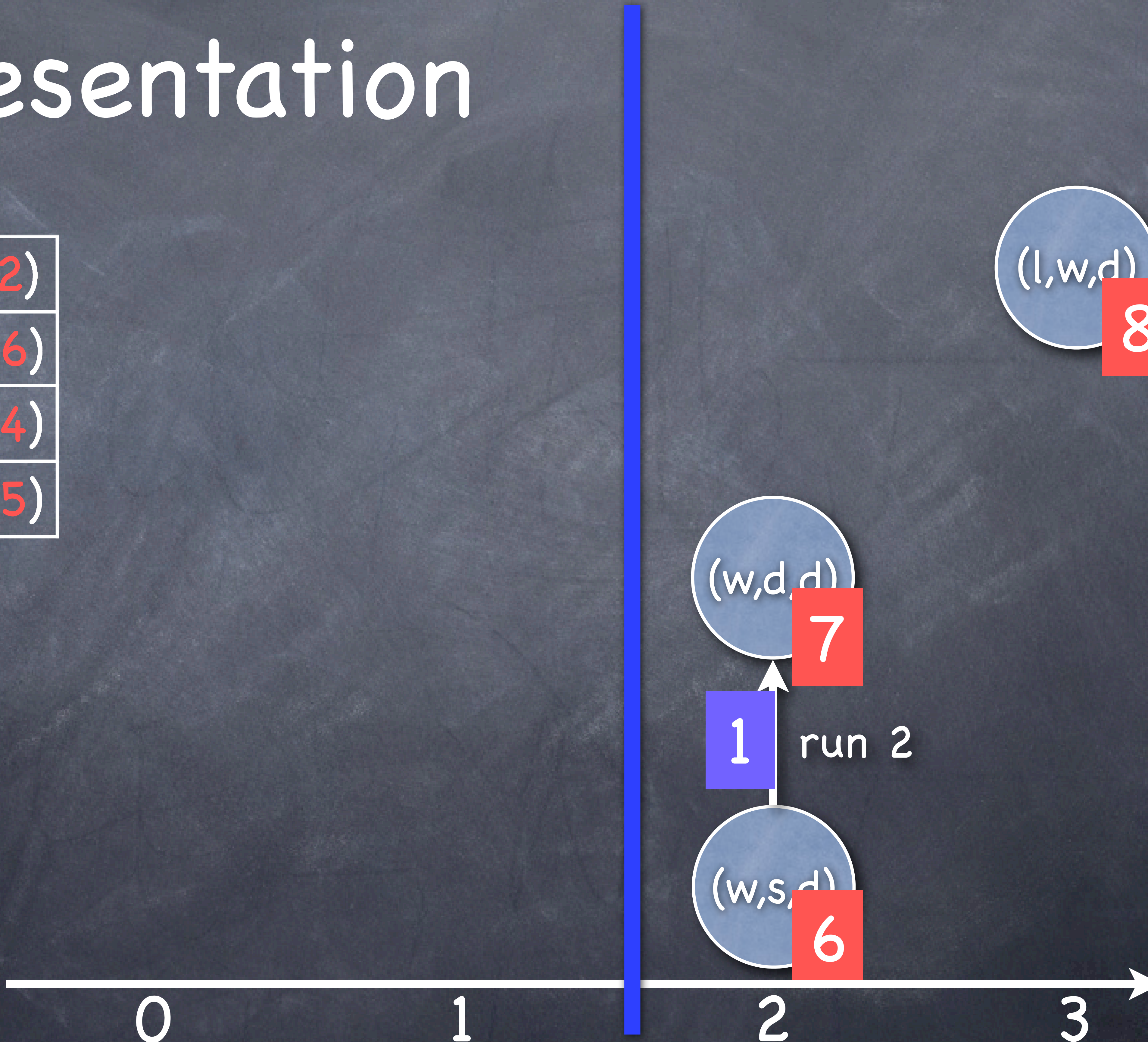
Building the Condensed Representation

0	2	2	(0,1)	(1,2)
1	2	3	(1,3)	(2,6)
2	2	3	(0,3)	(3,4)
3	2	3	(2,7)	(3,5)
4	1	3	(0,5)	
5	1	4	(4,8)	
6				
7				
8				



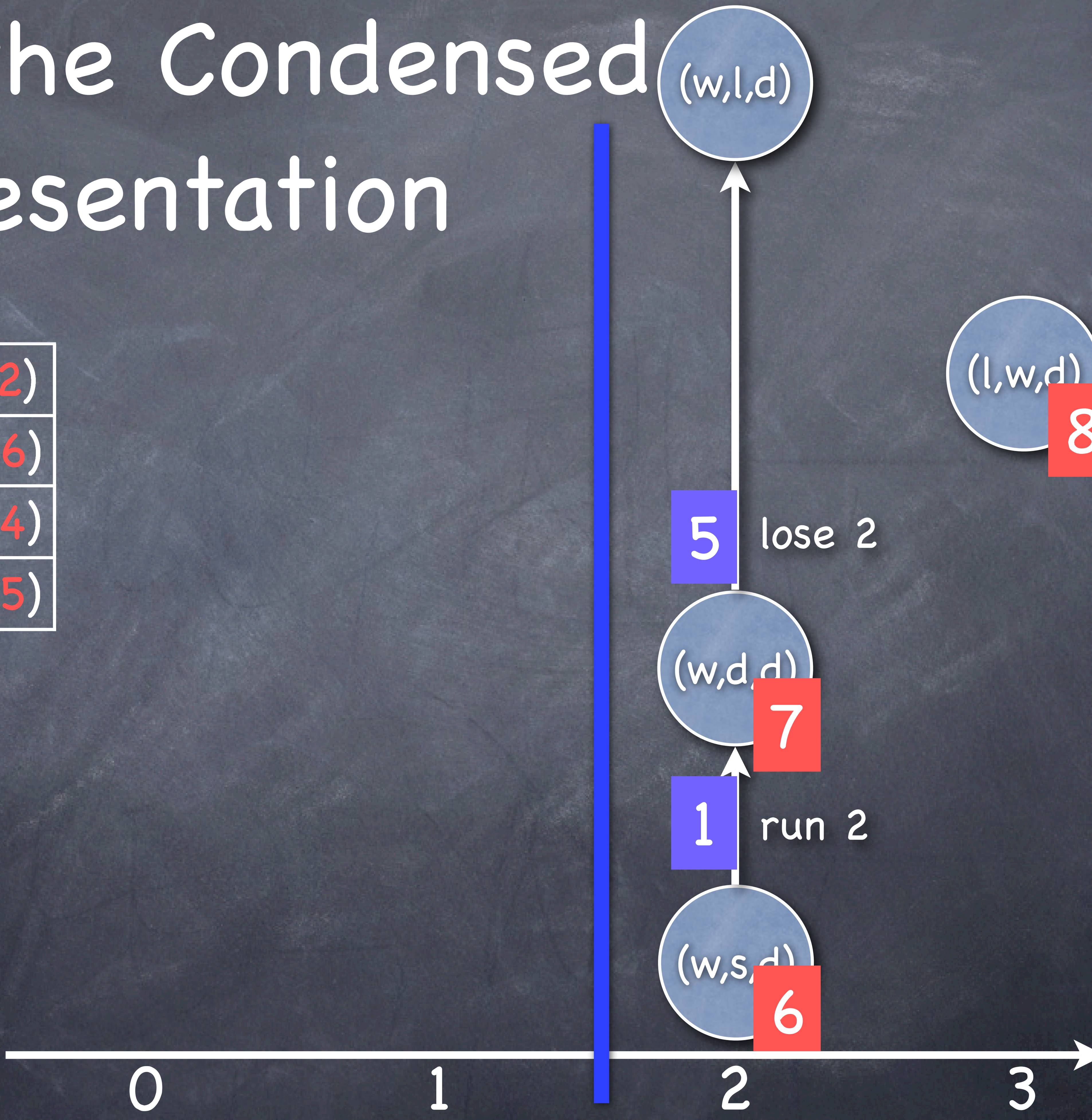
Building the Condensed Representation

0	2	2	(0,1)	(1,2)
1	2	3	(1,3)	(2,6)
2	2	3	(0,3)	(3,4)
3	2	3	(2,7)	(3,5)
4	1	3	(0,5)	
5	1	4	(4,8)	
6	1	3	(1,7)	
7				
8				



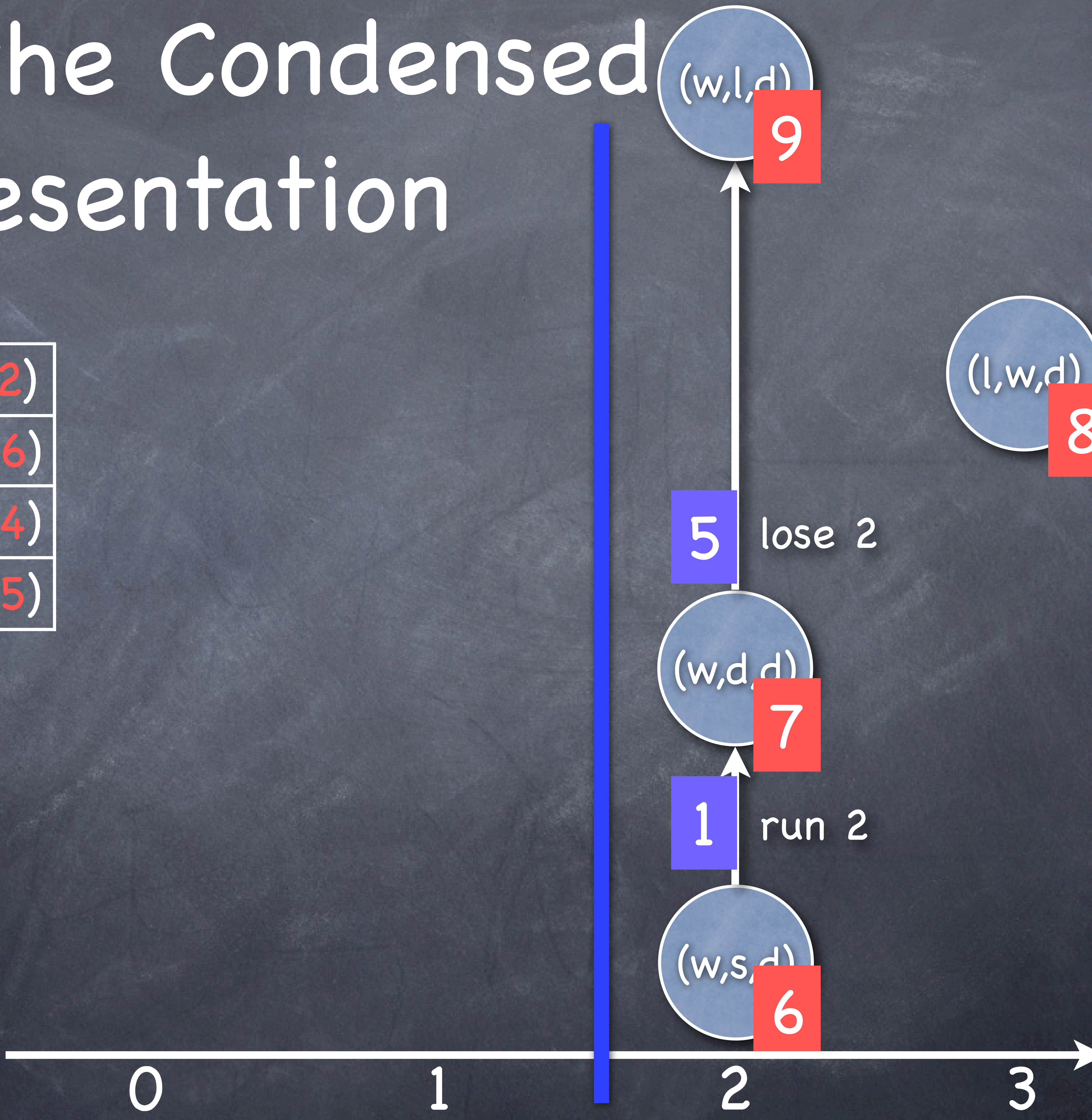
Building the Condensed Representation

0	2	2	(0,1)	(1,2)
1	2	3	(1,3)	(2,6)
2	2	3	(0,3)	(3,4)
3	2	3	(2,7)	(3,5)
4	1	3	(0,5)	
5	1	4	(4,8)	
6	1	3	(1,7)	
7				
8				



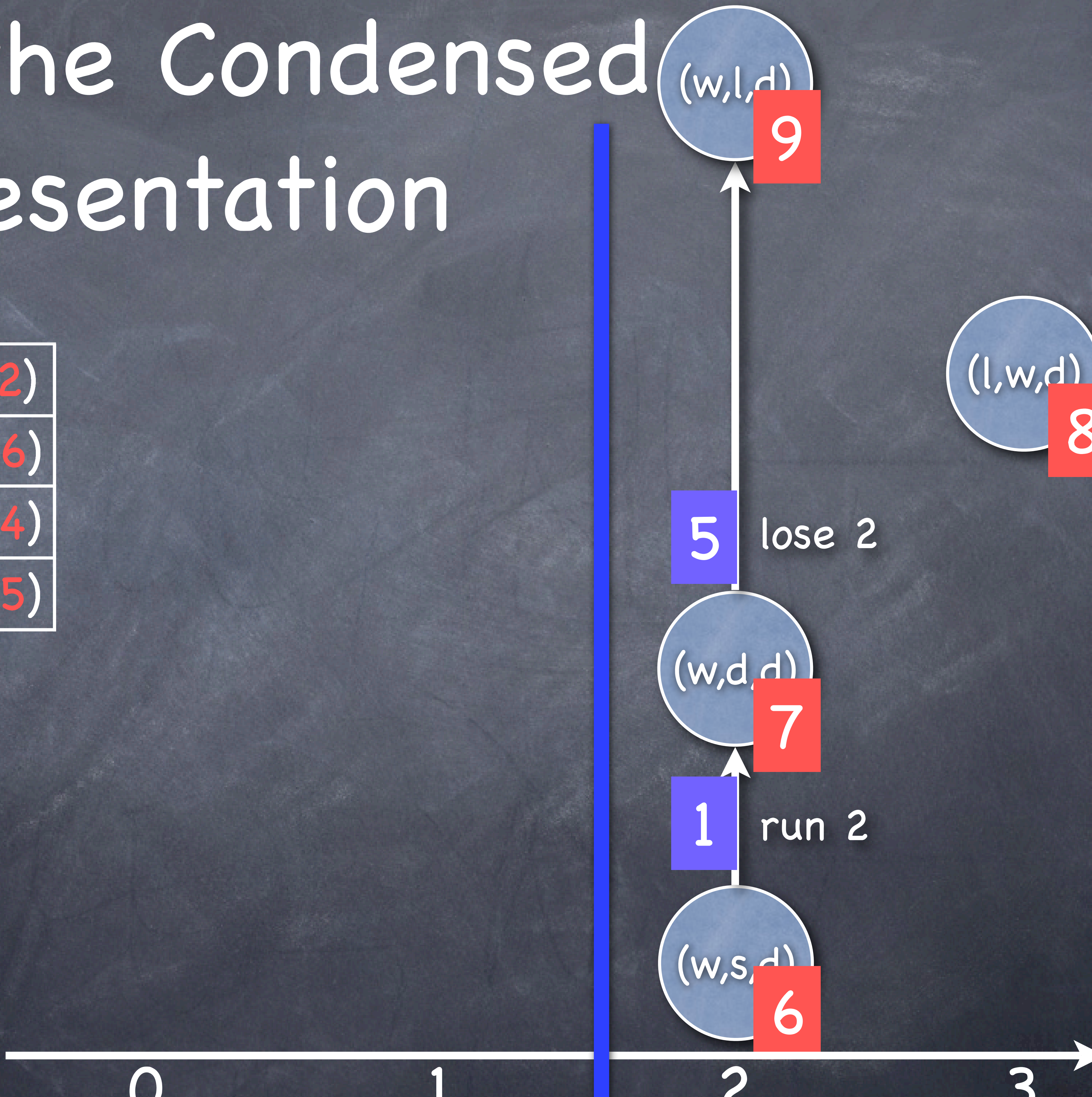
Building the Condensed Representation

0	2	2	(0,1) (1,2)
1	2	3	(1,3) (2,6)
2	2	3	(0,3) (3,4)
3	2	3	(2,7) (3,5)
4	1	3	(0,5)
5	1	4	(4,8)
6	1	3	(1,7)
7			
8			



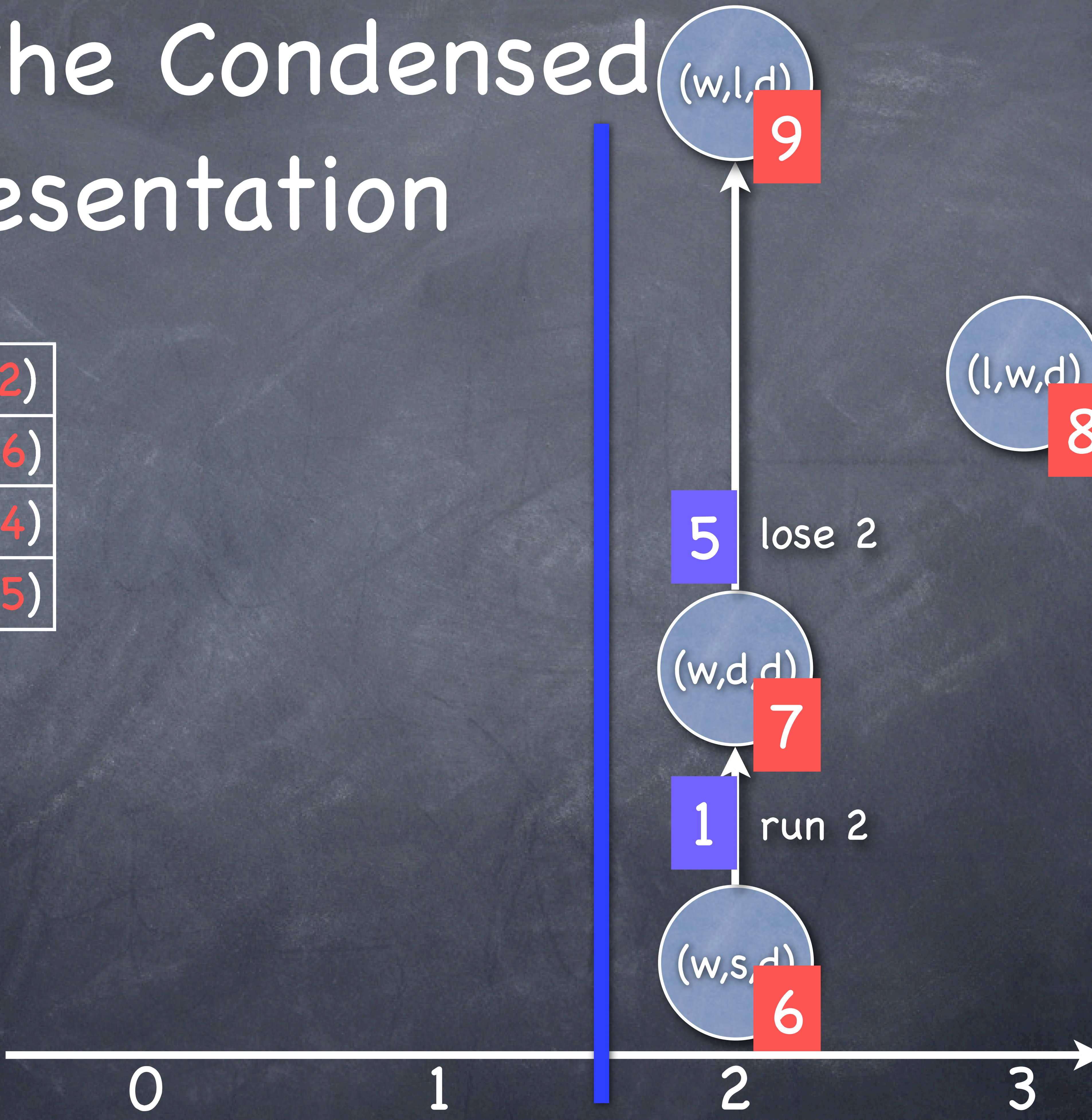
Building the Condensed Representation

0	2	2	(0,1)	(1,2)
1	2	3	(1,3)	(2,6)
2	2	3	(0,3)	(3,4)
3	2	3	(2,7)	(3,5)
4	1	3	(0,5)	
5	1	4	(4,8)	
6	1	3	(1,7)	
7	1	4	(5,9)	
8				
9				



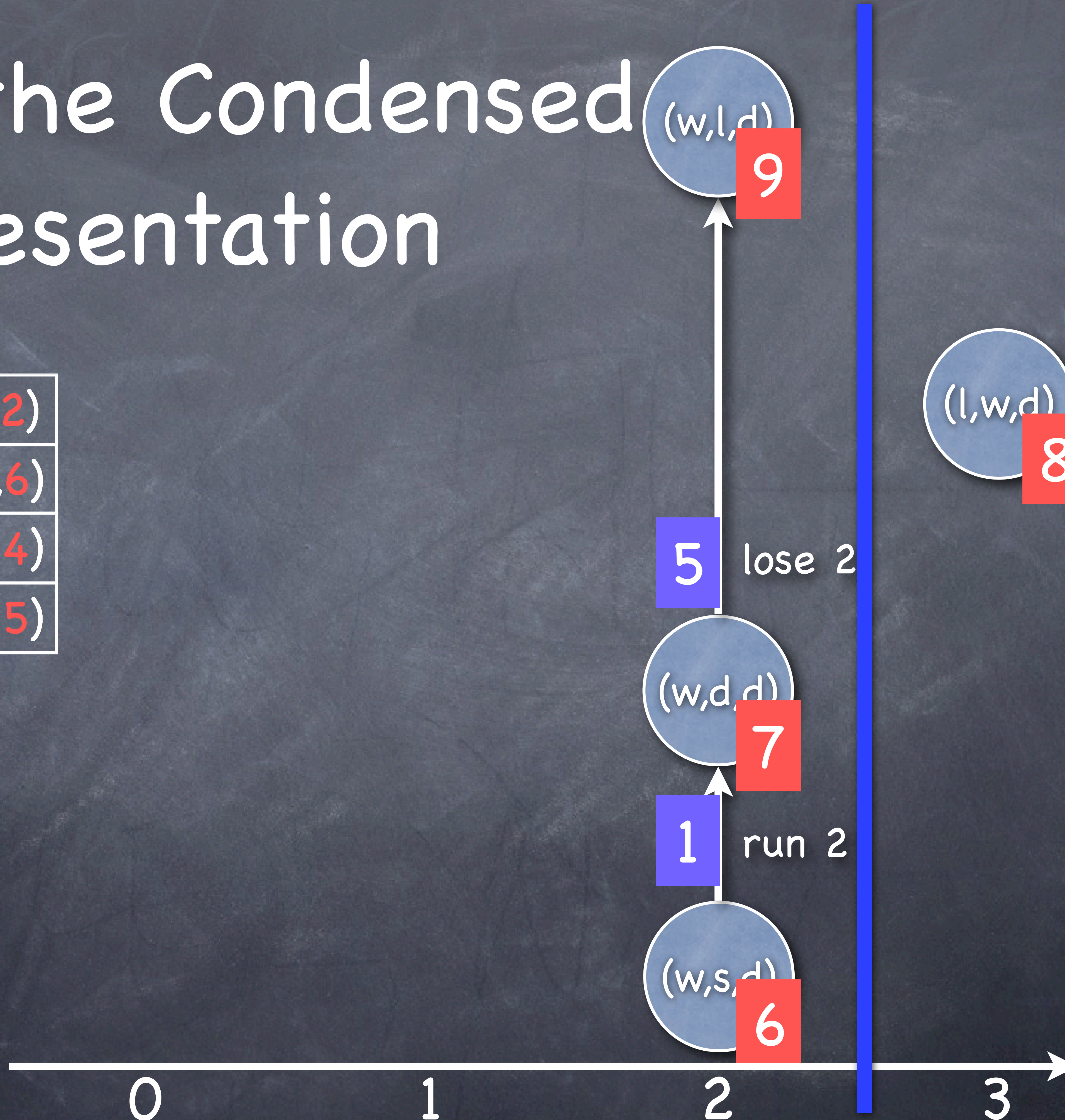
Building the Condensed Representation

0	2	2	(0,1)	(1,2)
1	2	3	(1,3)	(2,6)
2	2	3	(0,3)	(3,4)
3	2	3	(2,7)	(3,5)
4	1	3	(0,5)	
5	1	4	(4,8)	
6	1	3	(1,7)	
7	1	4	(5,9)	
8				
9	0			



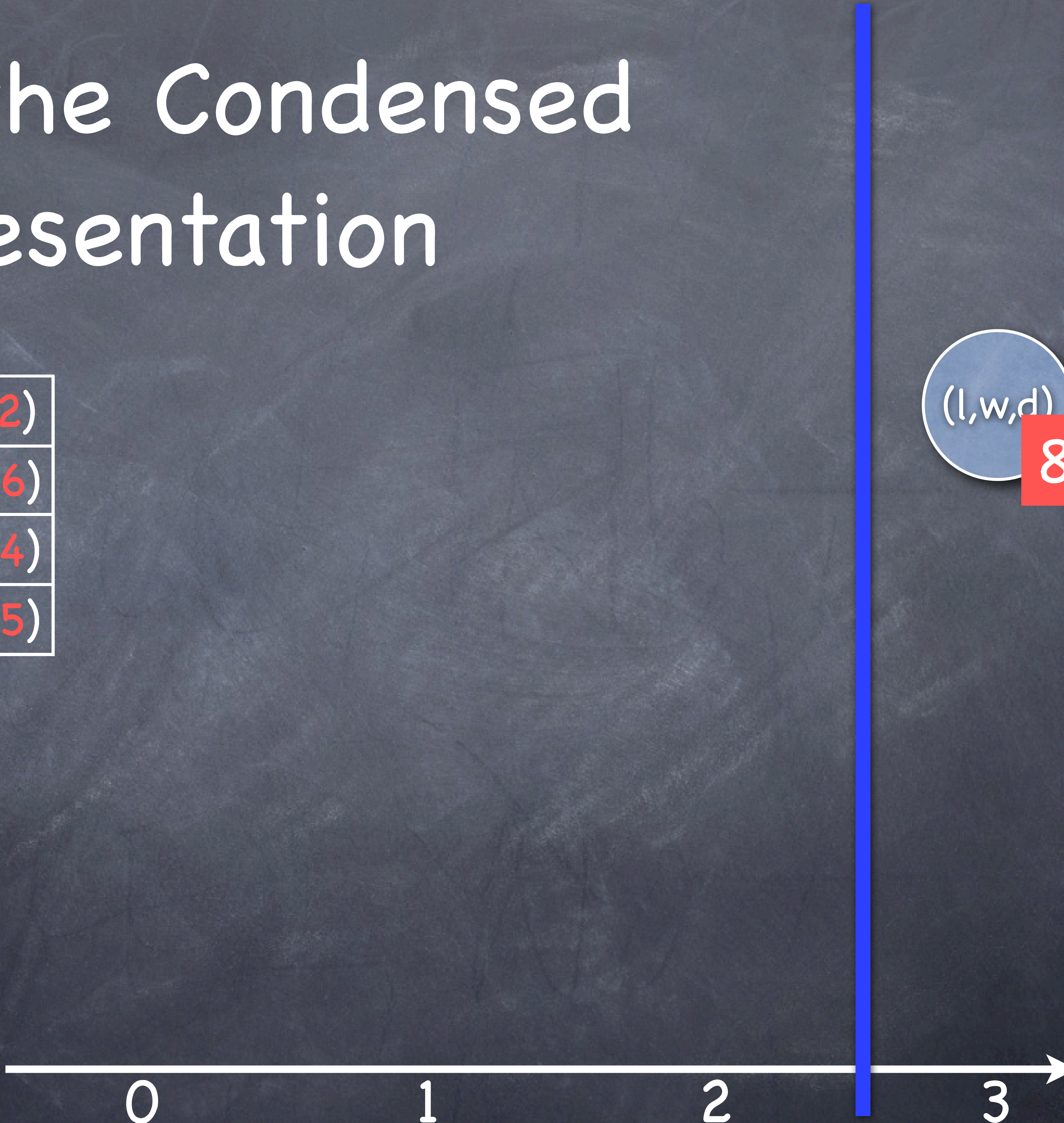
Building the Condensed Representation

0	2	2	(0,1)	(1,2)
1	2	3	(1,3)	(2,6)
2	2	3	(0,3)	(3,4)
3	2	3	(2,7)	(3,5)
4	1	3	(0,5)	
5	1	4	(4,8)	
6	1	3	(1,7)	
7	1	4	(5,9)	
8				
9	0			



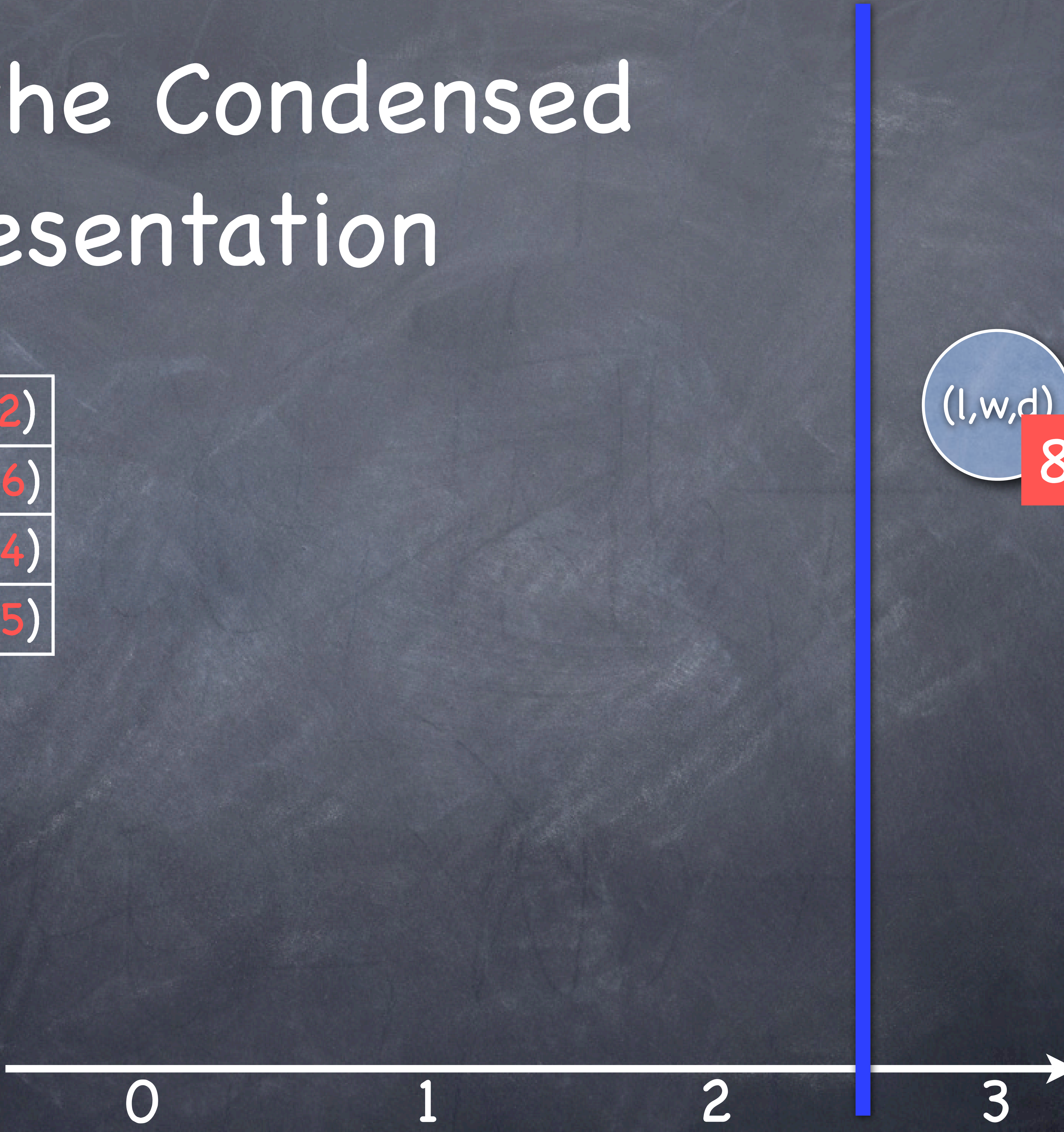
Building the Condensed Representation

0	2	2	(0,1)	(1,2)
1	2	3	(1,3)	(2,6)
2	2	3	(0,3)	(3,4)
3	2	3	(2,7)	(3,5)
4	1	3	(0,5)	
5	1	4	(4,8)	
6	1	3	(1,7)	
7	1	4	(5,9)	
8				
9	0			



Building the Condensed Representation

0	2	2	(0,1)	(1,2)
1	2	3	(1,3)	(2,6)
2	2	3	(0,3)	(3,4)
3	2	3	(2,7)	(3,5)
4	1	3	(0,5)	
5	1	4	(4,8)	
6	1	3	(1,7)	
7	1	4	(5,9)	
8	0			
9	0			



Building the Condensed Representation

0	2	2	(0,1)	(1,2)
1	2	3	(1,3)	(2,6)
2	2	3	(0,3)	(3,4)
3	2	3	(2,7)	(3,5)
4	1	3	(0,5)	
5	1	4	(4,8)	
6	1	3	(1,7)	
7	1	4	(5,9)	
8	0			
9	0			

Building the Condensed Representation

0	2	2	(0,1)	(1,2)
1	2	3	(1,3)	(2,6)
2	2	3	(0,3)	(3,4)
3	2	3	(2,7)	(3,5)
4	1	3	(0,5)	
5	1	4	(4,8)	
6	1	3	(1,7)	
7	1	4	(5,9)	
8	0			
9	0			

0 1 2 3

Evaluation of the Algorithm

- We only store a few (6) actual states
- The condensed representation uses $4 \cdot |R| \cdot w + |E| \cdot (\log|T| + \log|R|) + |F| \cdot \log|S|$ bits
  - R: all reachable states
  - w: size of machine word
  - E: all reachable arcs
  - T: all transitions
  - S: all syntactically possible states
  - F: states on the front
- Efficient standard representation $|R| \cdot (3 \cdot w + \log|S|) + |E| \cdot \log|S|$

More Information about the Algorithm

- T. Mailund, M. Westergaard: Obtaining Memory-Efficient Reachability Graph Representations Using the Sweep-Line Method, TACAS 2004

Backtrack
the State Space

Backtrack the State Space

- We need to support the operations add(s) and contains(s) but not any get operation

Backtrack the State Space

- We need to support the operations add(s) and contains(s) but not any get operation
- Idea of hash compaction: Store hash value for each state only

Backtrack the State Space

- We need to support the operations add(s) and contains(s) but not any get operation
- Idea of hash compaction: Store hash value for each state only
- Hash collision?

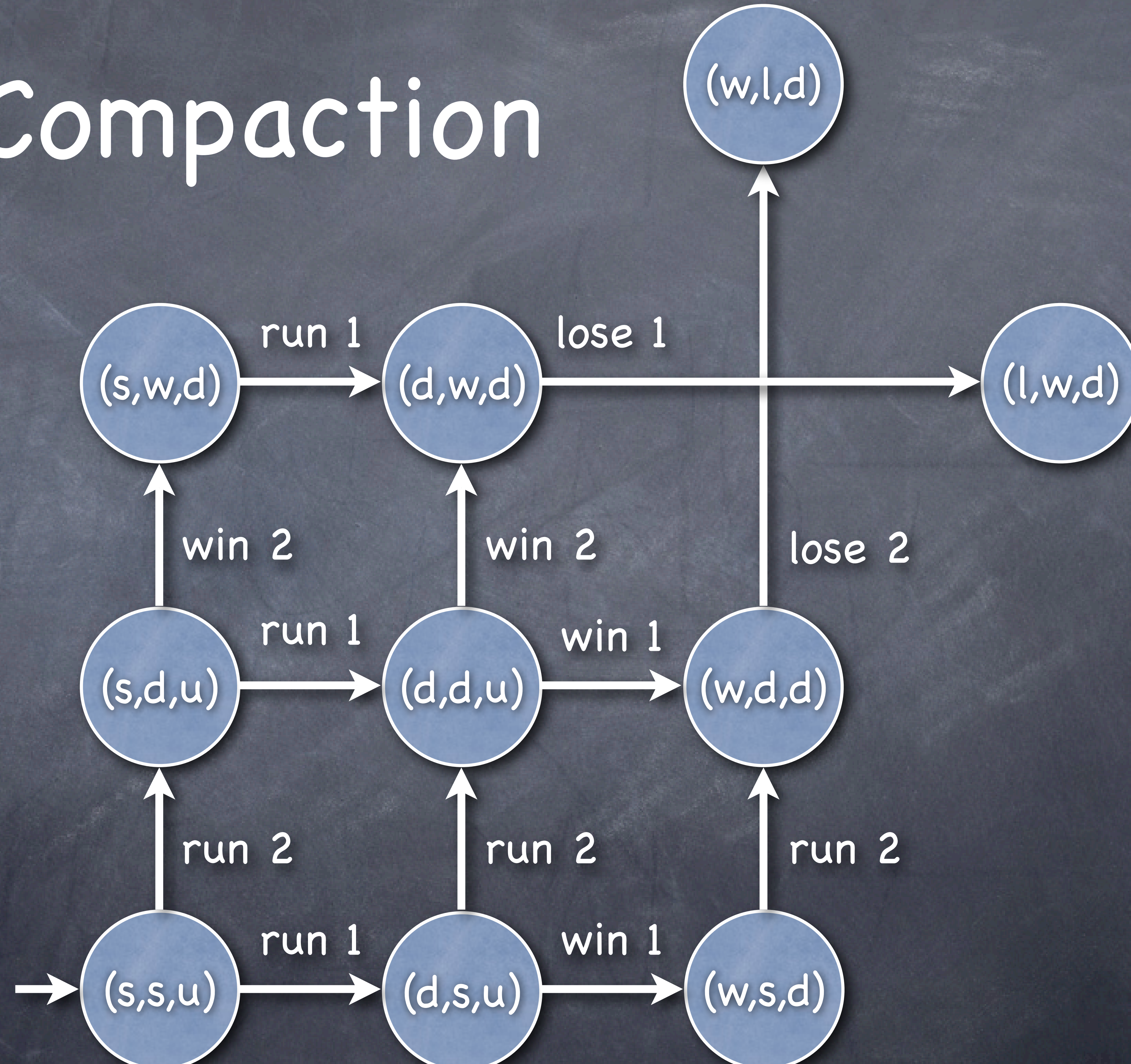
Backtrack the State Space

- We need to support the operations add(s) and contains(s) but not any get operation
- Idea of hash compaction: Store hash value for each state only
- Hash collision?
  - Hash compaction fails

Backtrack the State Space

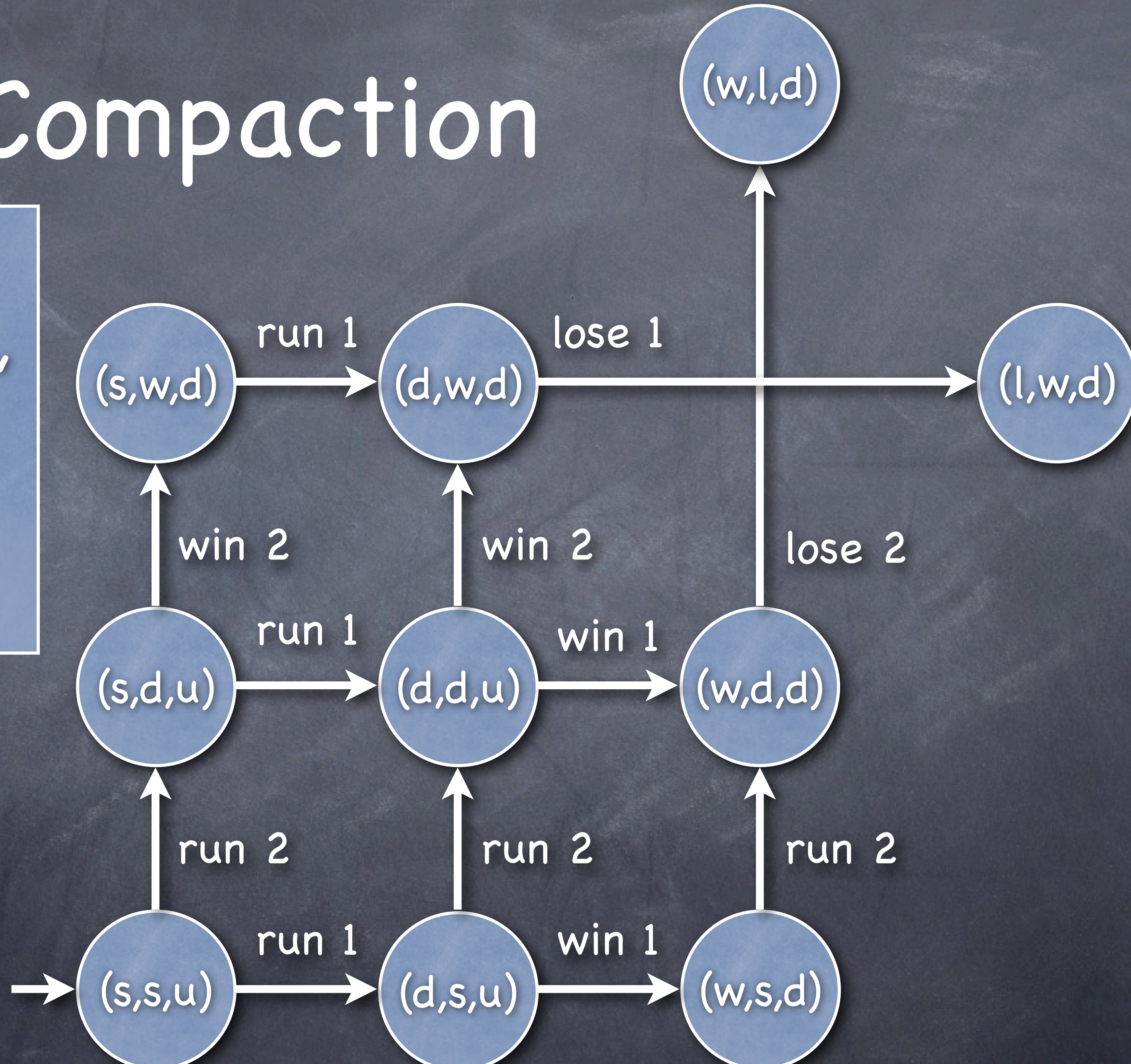
- We need to support the operations add(s) and contains(s) but not any get operation
- Idea of hash compaction: Store hash value for each state only
- Hash collision?
  - Hash compaction fails
  - We will store for each state a predecessor and trace from the initial state

Hash Compaction



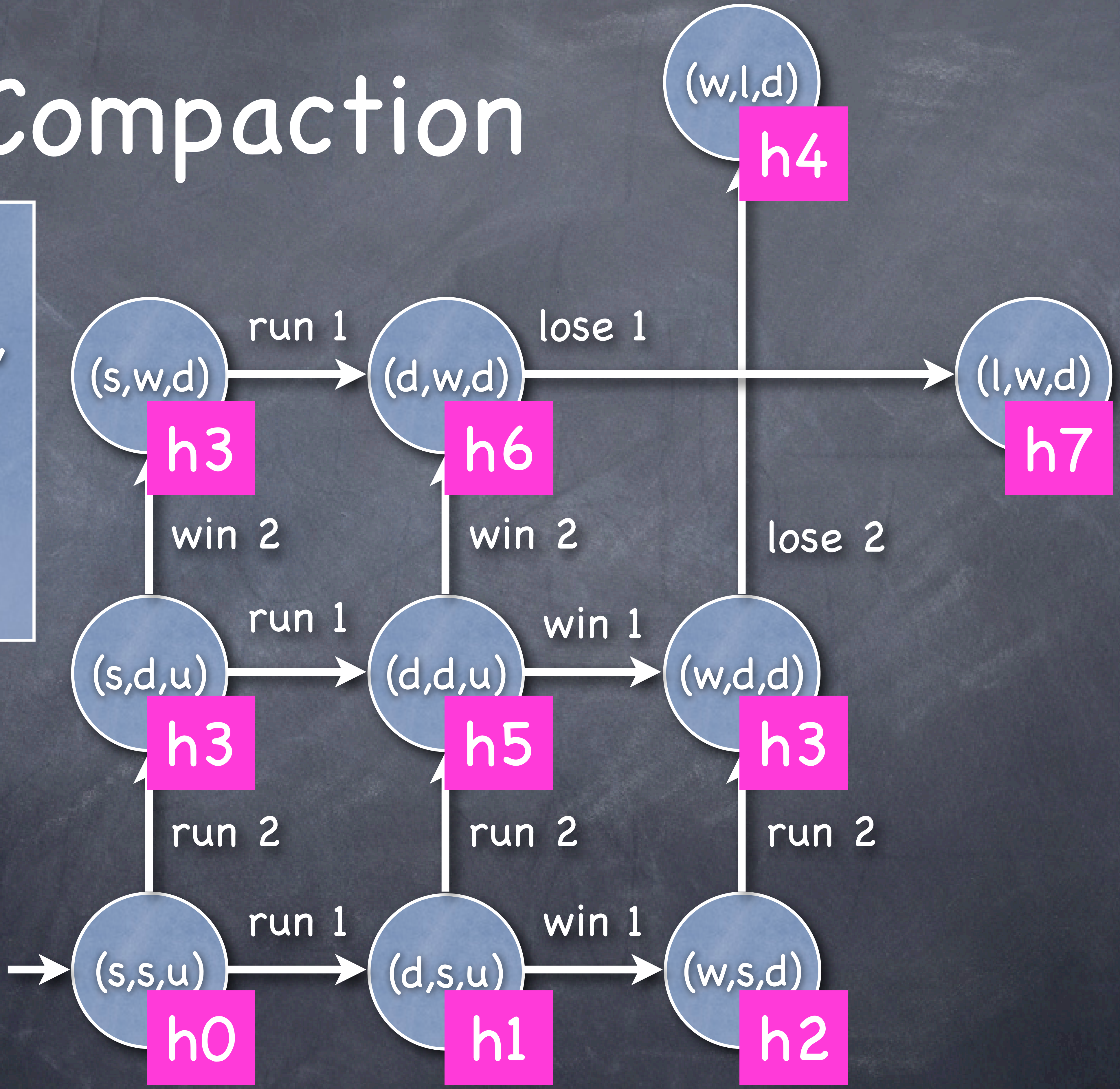
Hash Compaction

We assume a hash function, assigning to each state a hash-value



Hash Compaction

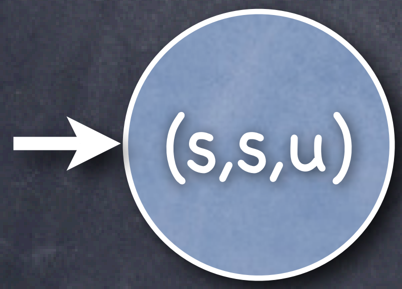
We assume a hash function, assigning to each state a hash-value



Hash Compaction

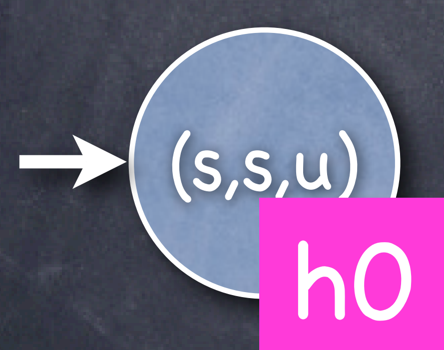
→

Hash Compaction



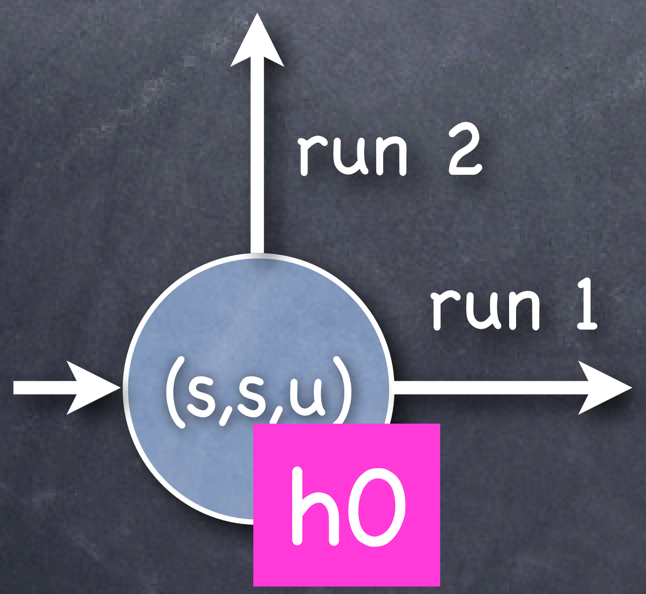
Hash Compaction

h0



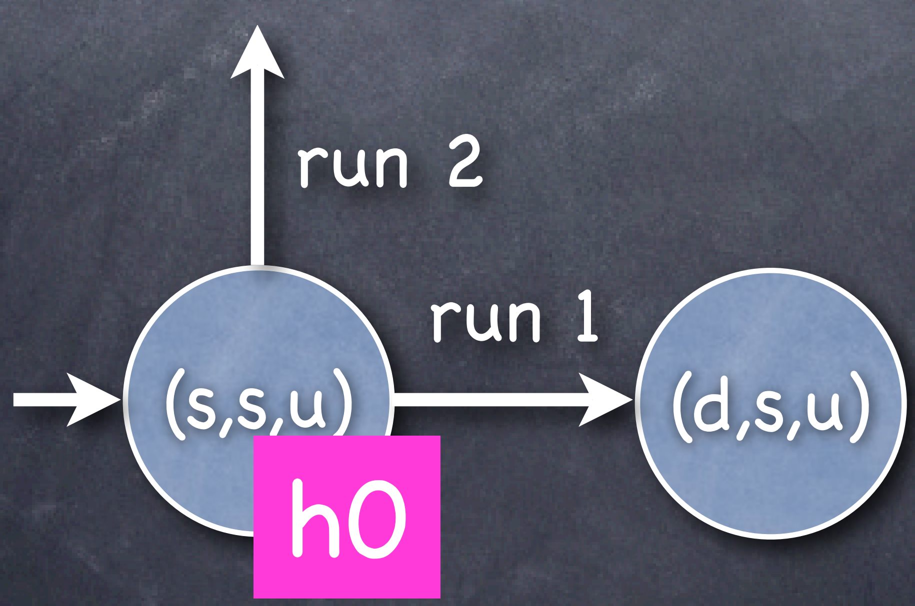
Hash Compaction

$\boxed{h0}$



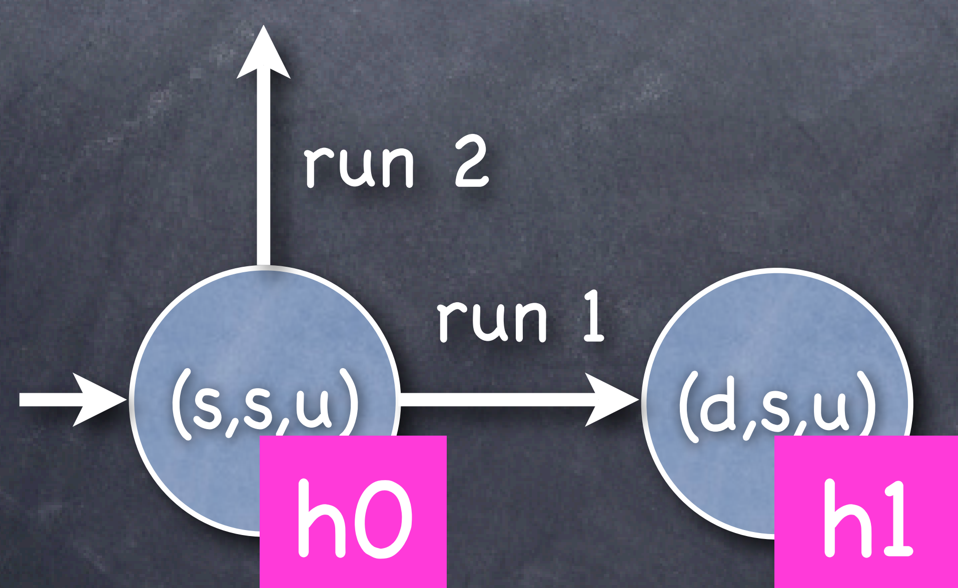
Hash Compaction

h0



Hash Compaction

h0
h1



Hash Compaction

h0
h1
h2
h3
h4

Hash Compaction

h0
h1
h2
h3
h4

Hash Compaction

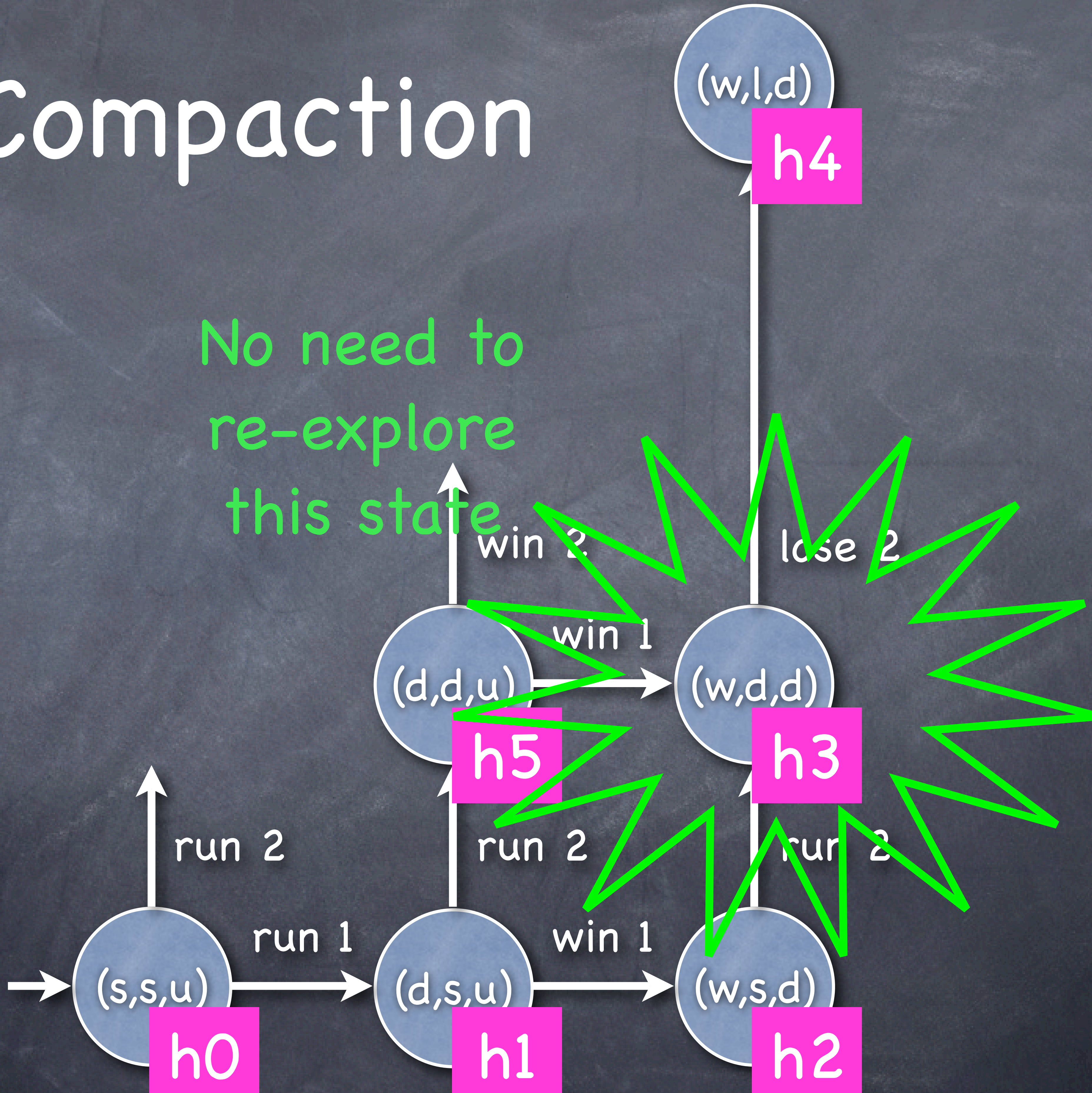
h0
h1
h2
h3
h4
h5

Hash Compaction

h0
h1
h2
h3
h4
h5

Hash Compaction

h0
h1
h2
h3
h4
h5

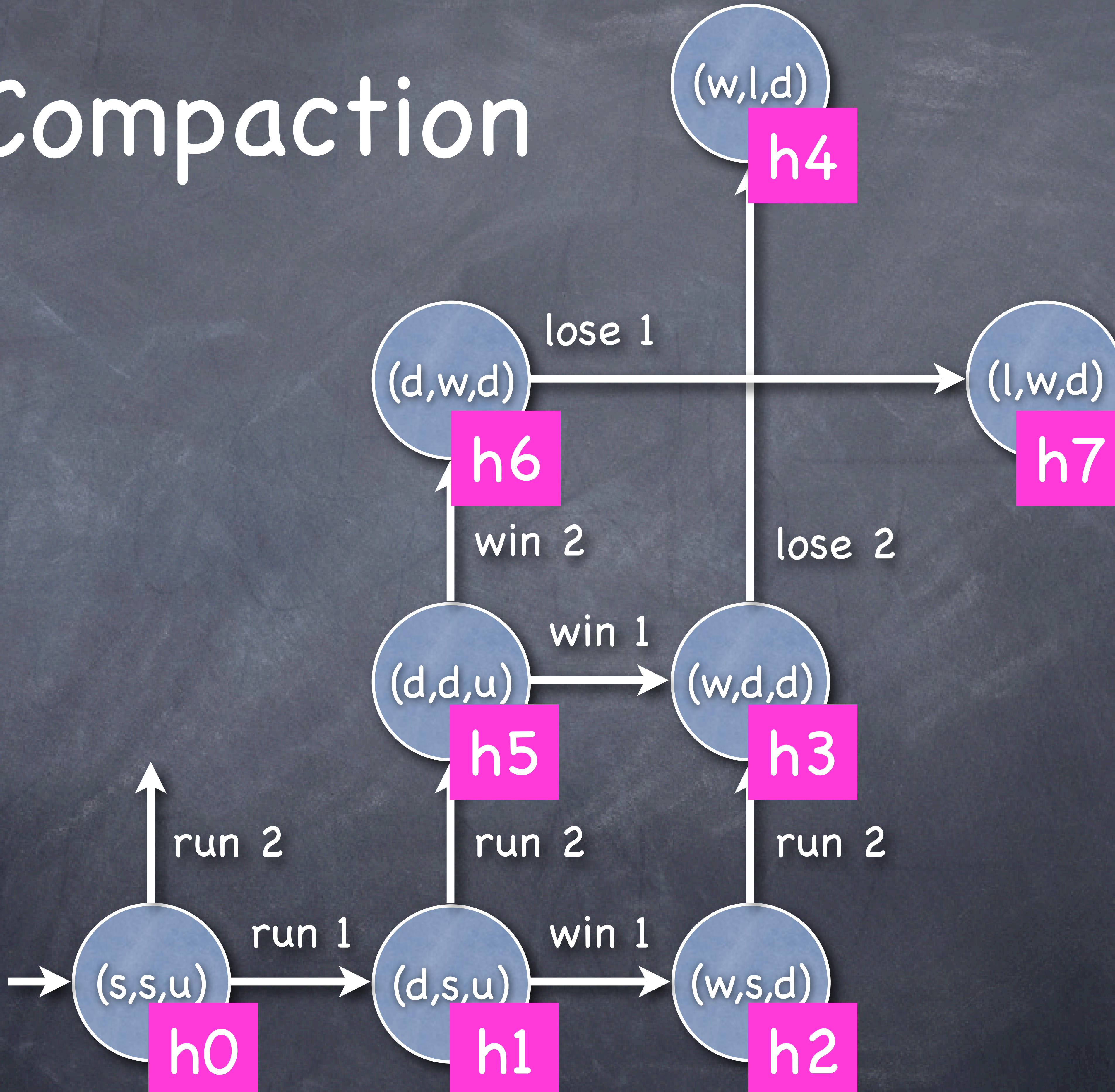


Hash Compaction

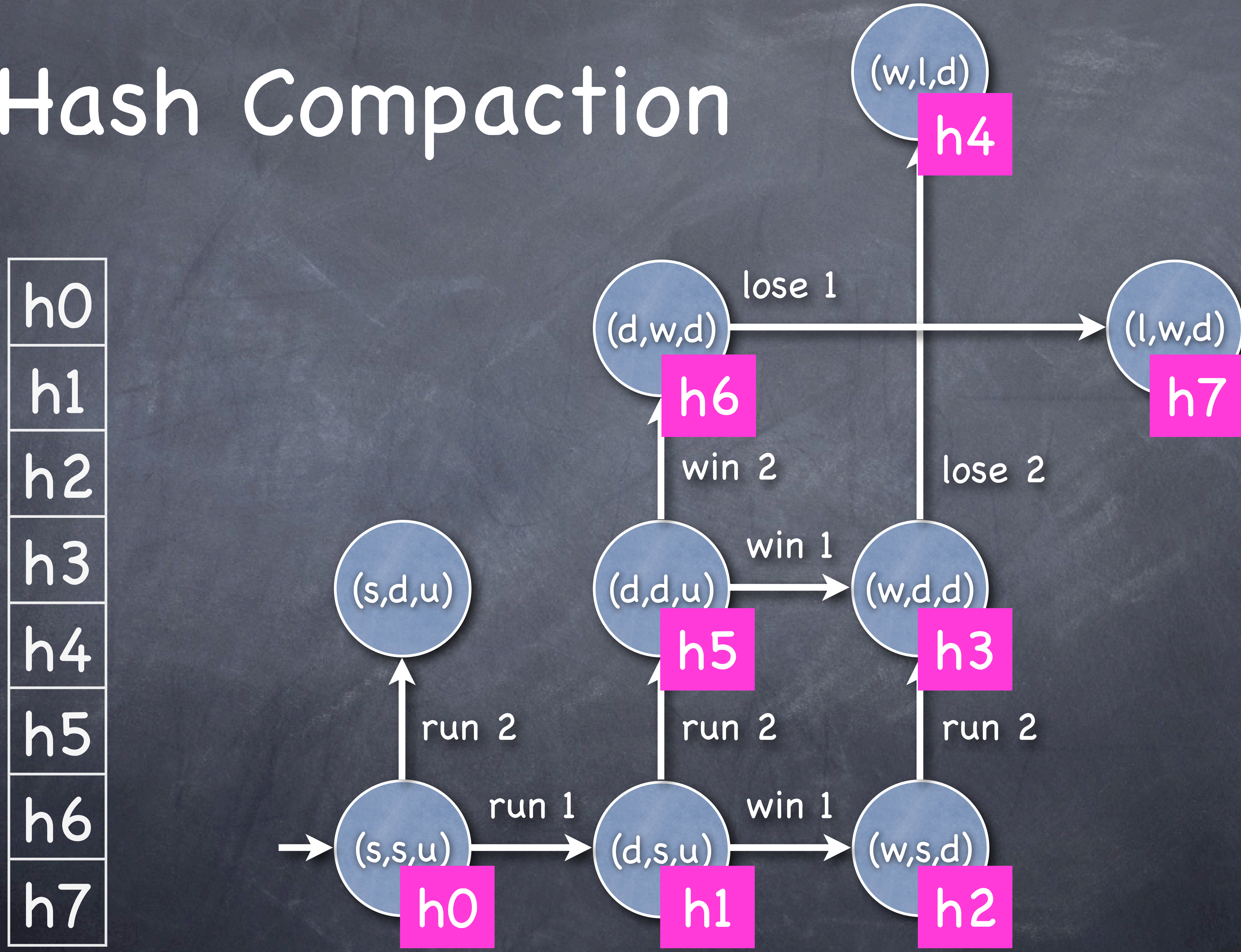
h0
h1
h2
h3
h4
h5

Hash Compaction

h0
h1
h2
h3
h4
h5
h6
h7

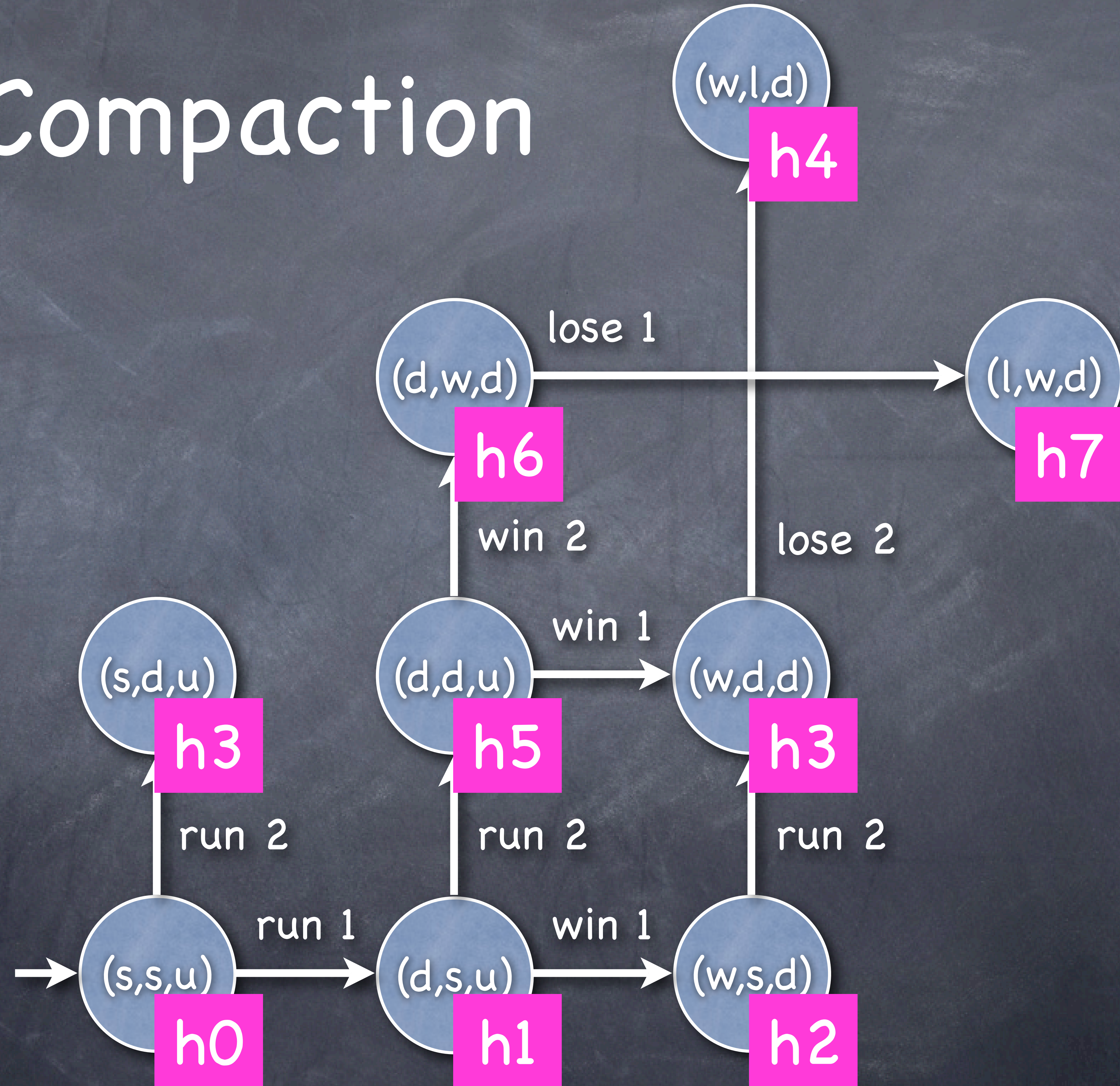


Hash Compaction

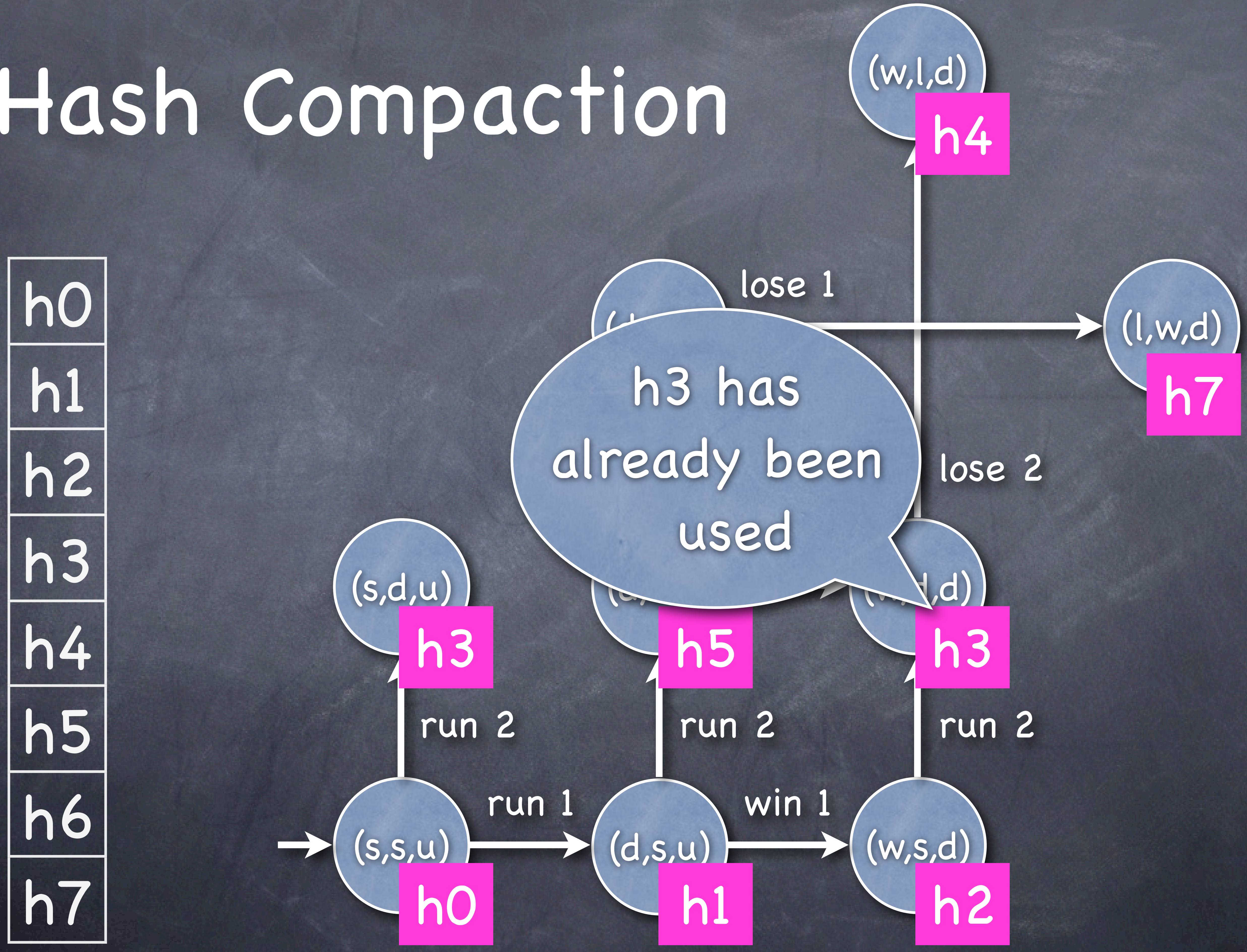


Hash Compaction

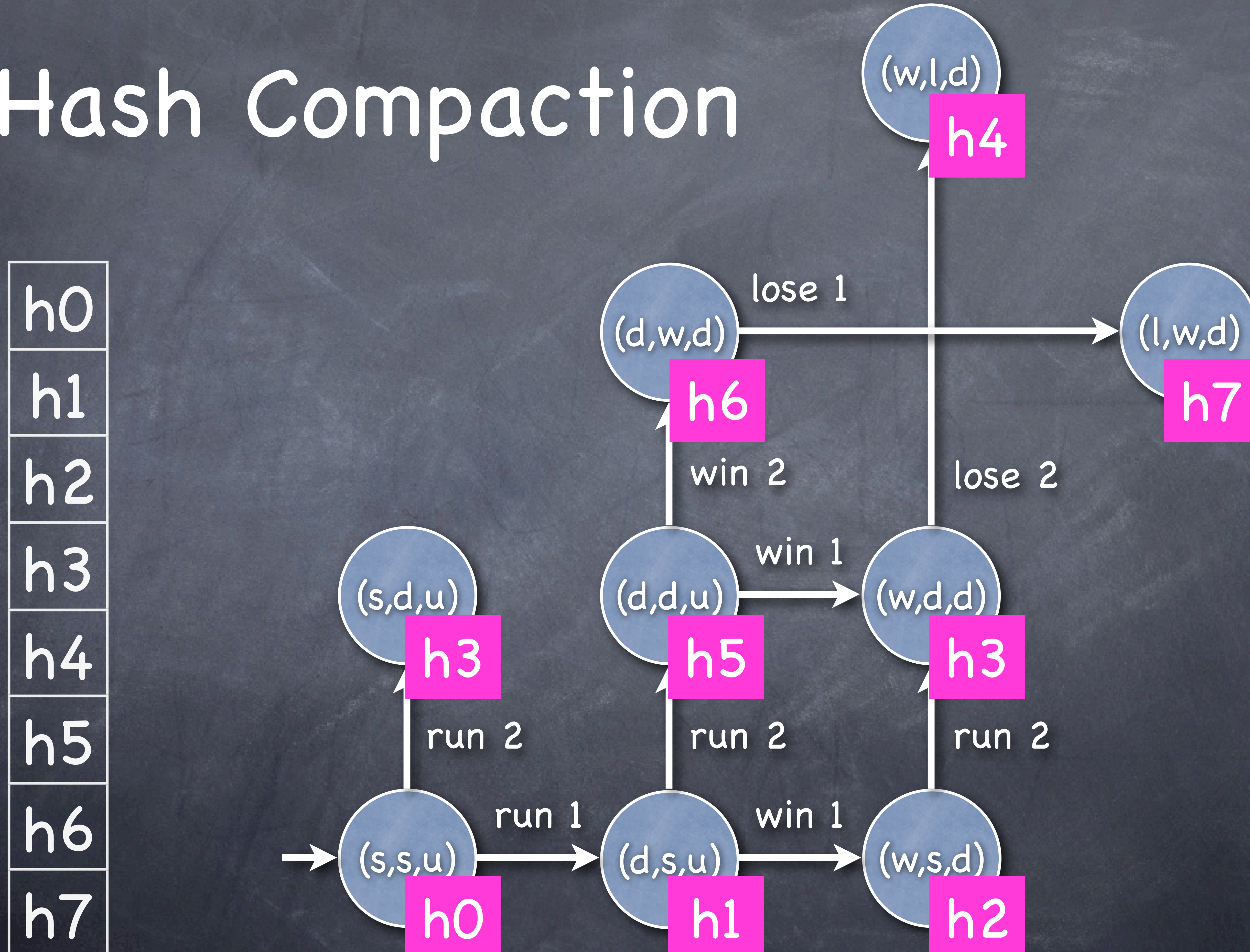
h0
h1
h2
h3
h4
h5
h6
h7



Hash Compaction

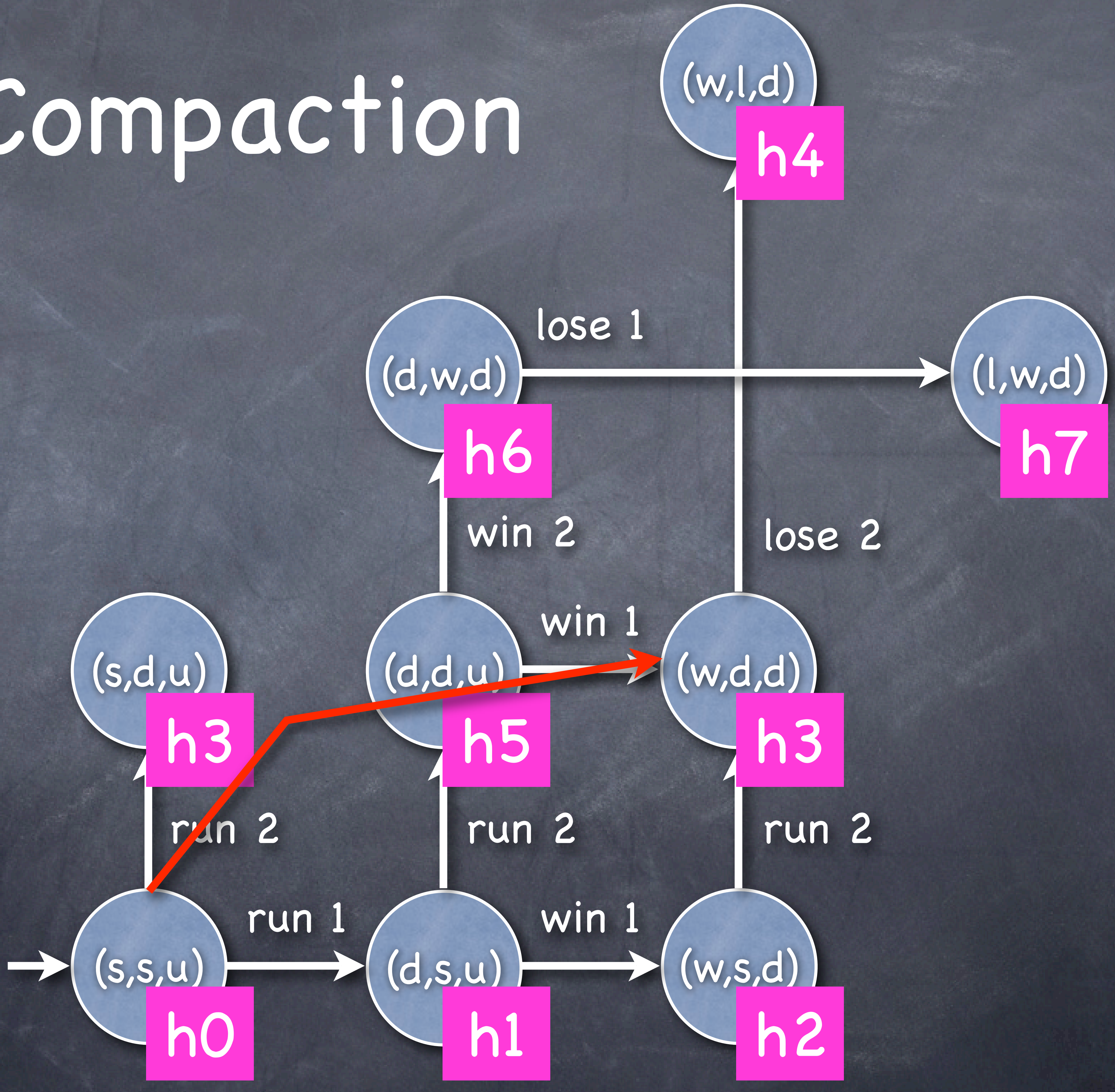


Hash Compaction



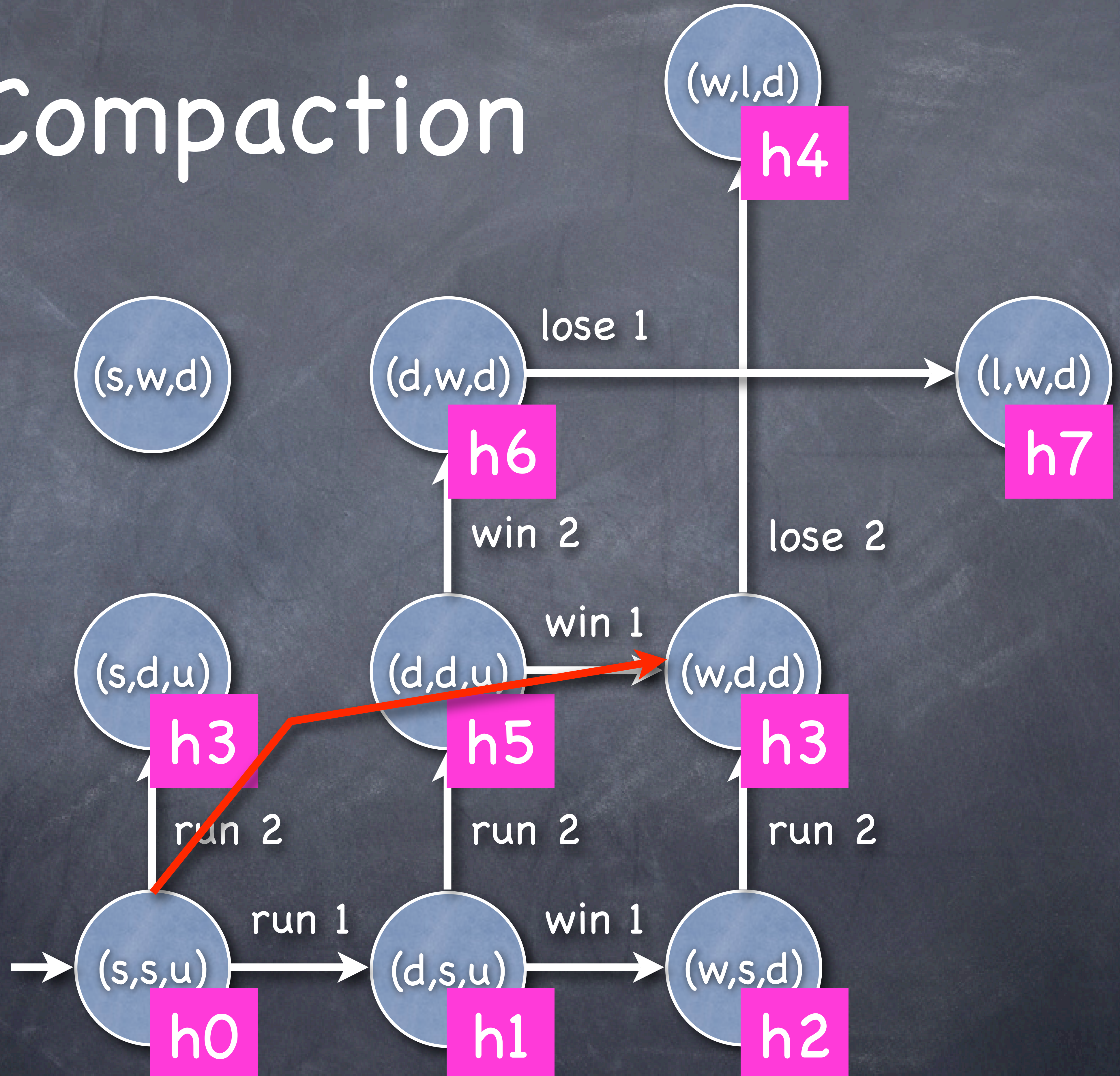
Hash Compaction

h0
h1
h2
h3
h4
h5
h6
h7



Hash Compaction

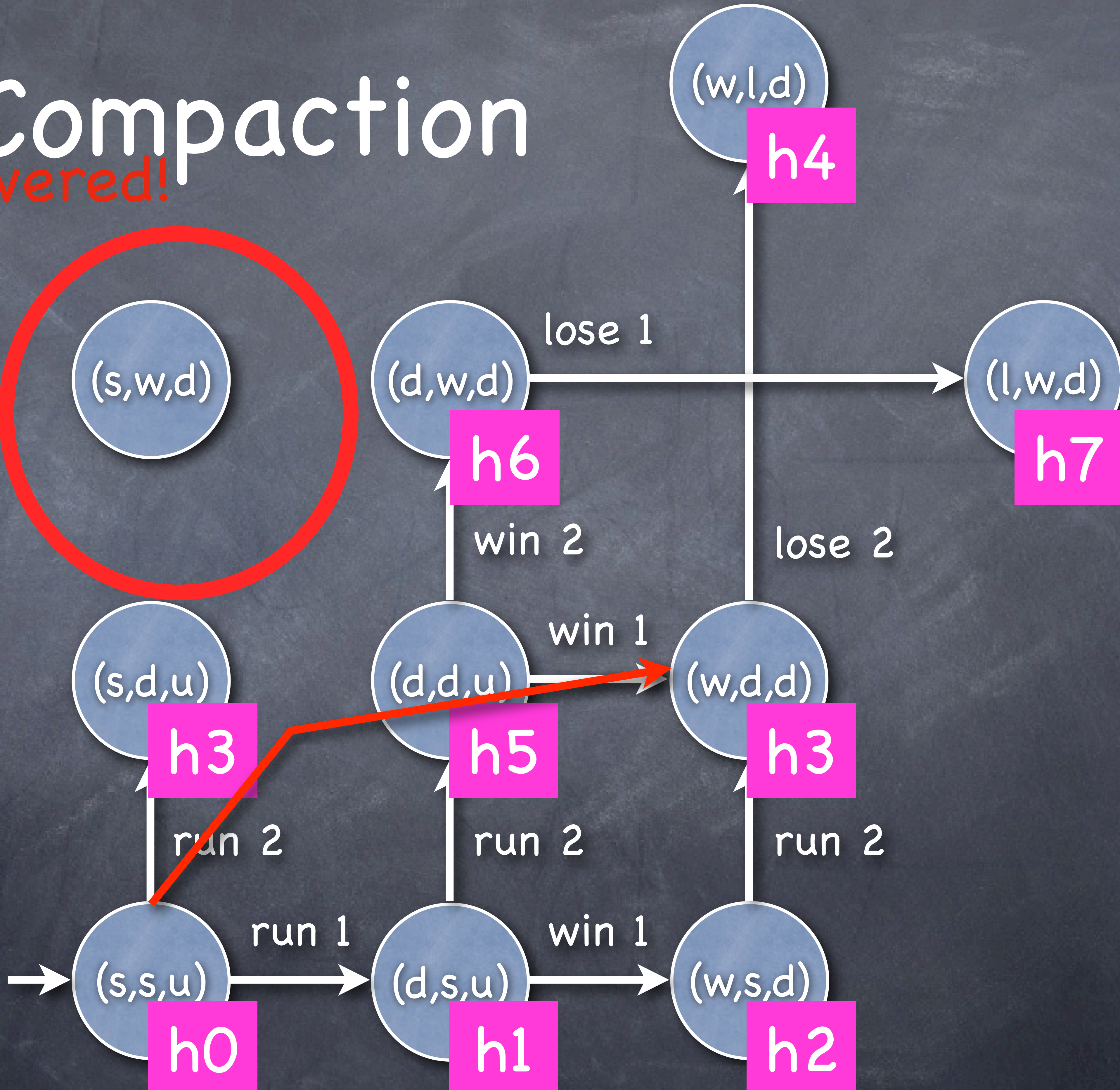
h0
h1
h2
h3
h4
h5
h6
h7



Hash Compaction

Never discovered!

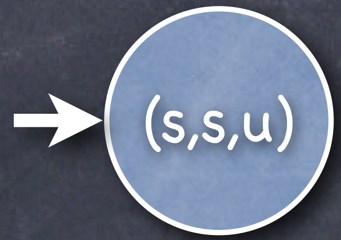
h0
h1
h2
h3
h4
h5
h6
h7



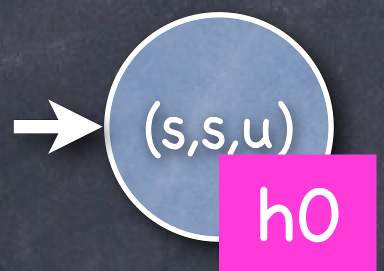
The ComBack Algorithm

→

The ComBack Algorithm

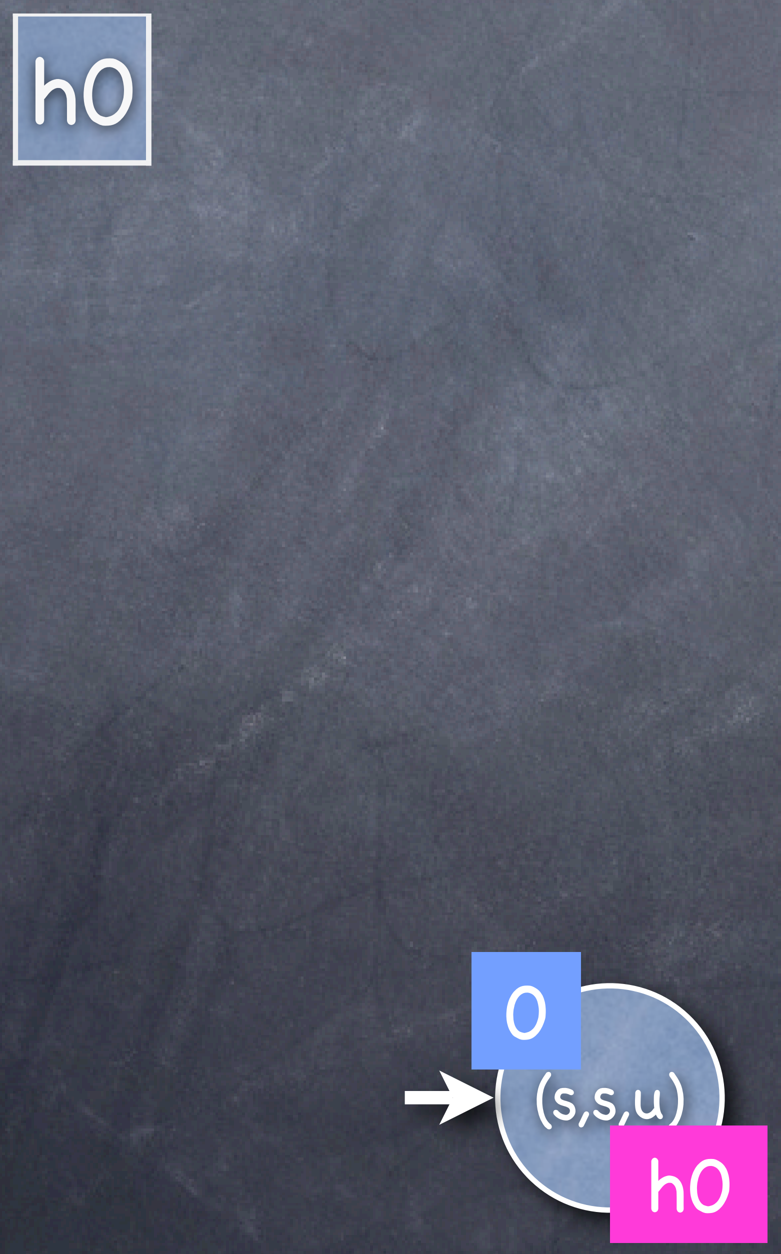


The ComBack Algorithm

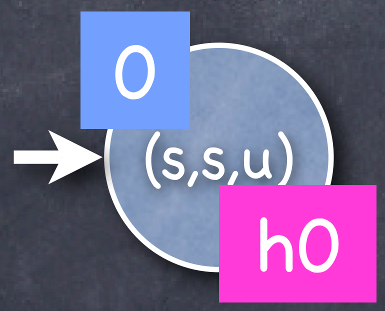


The ComBack Algorithm

The ComBack Algorithm



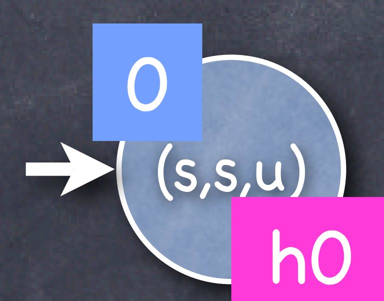
The ComBack Algorithm



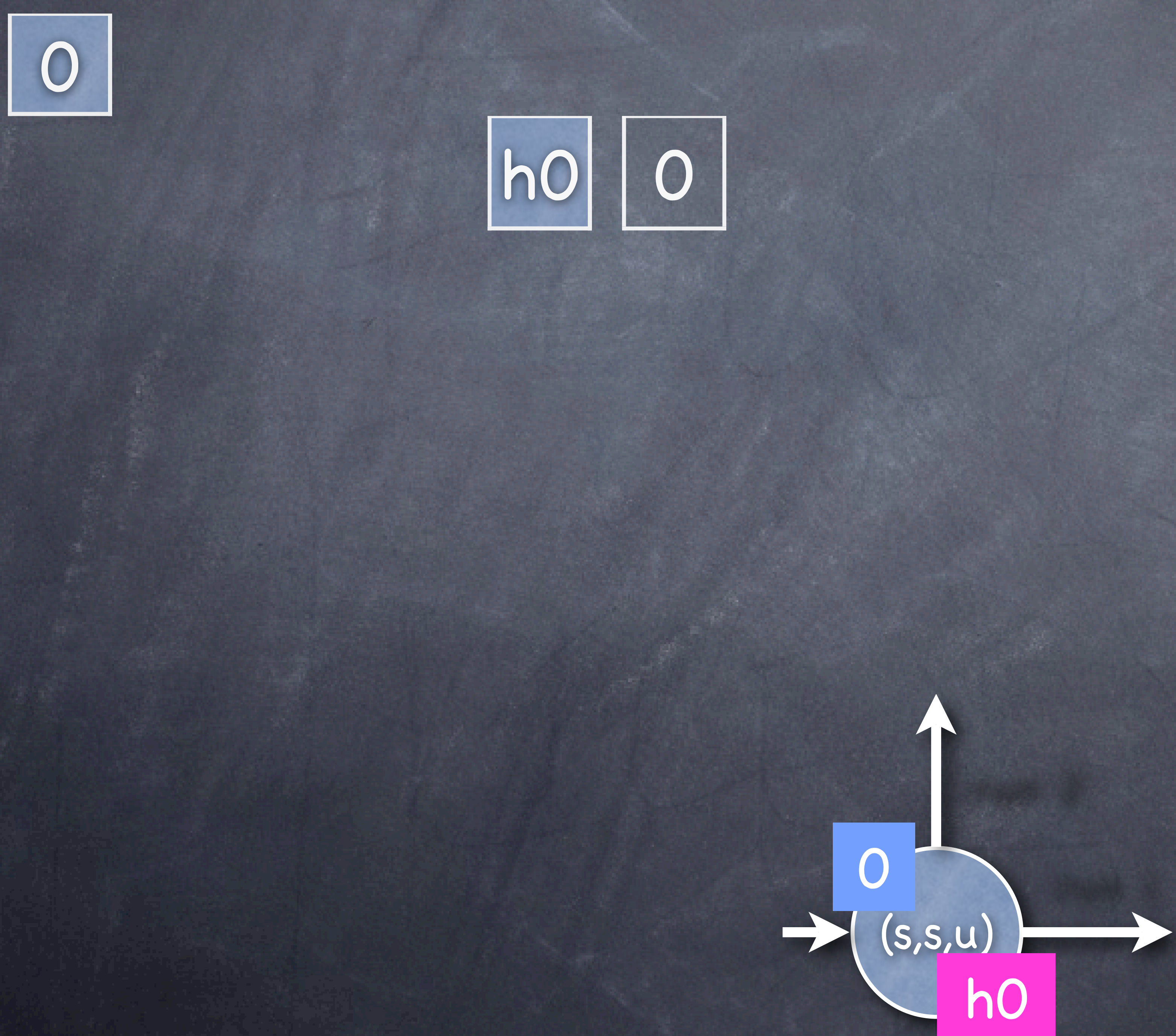
The ComBack Algorithm

0

h0 0



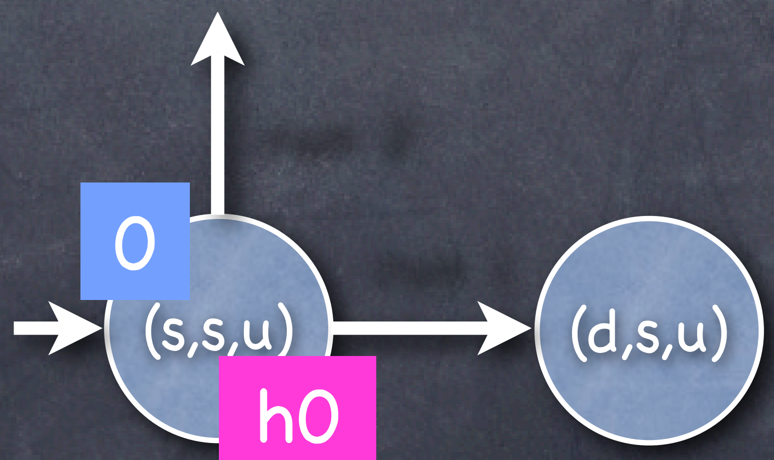
The ComBack Algorithm



The ComBack Algorithm

0

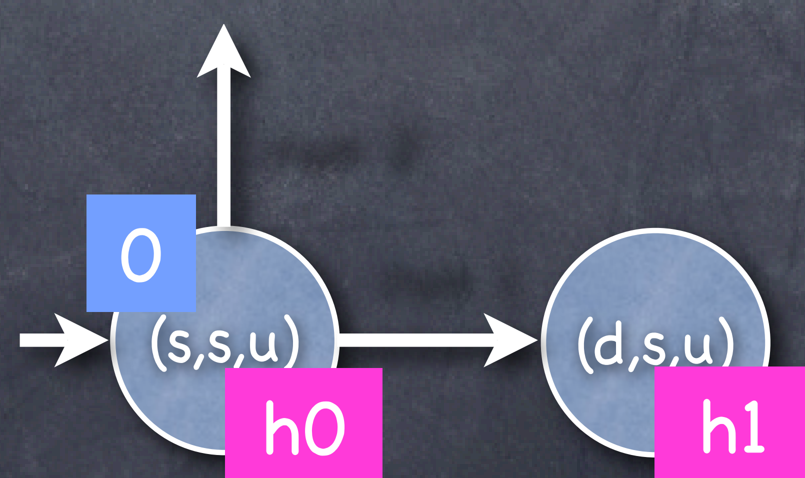
h0 0



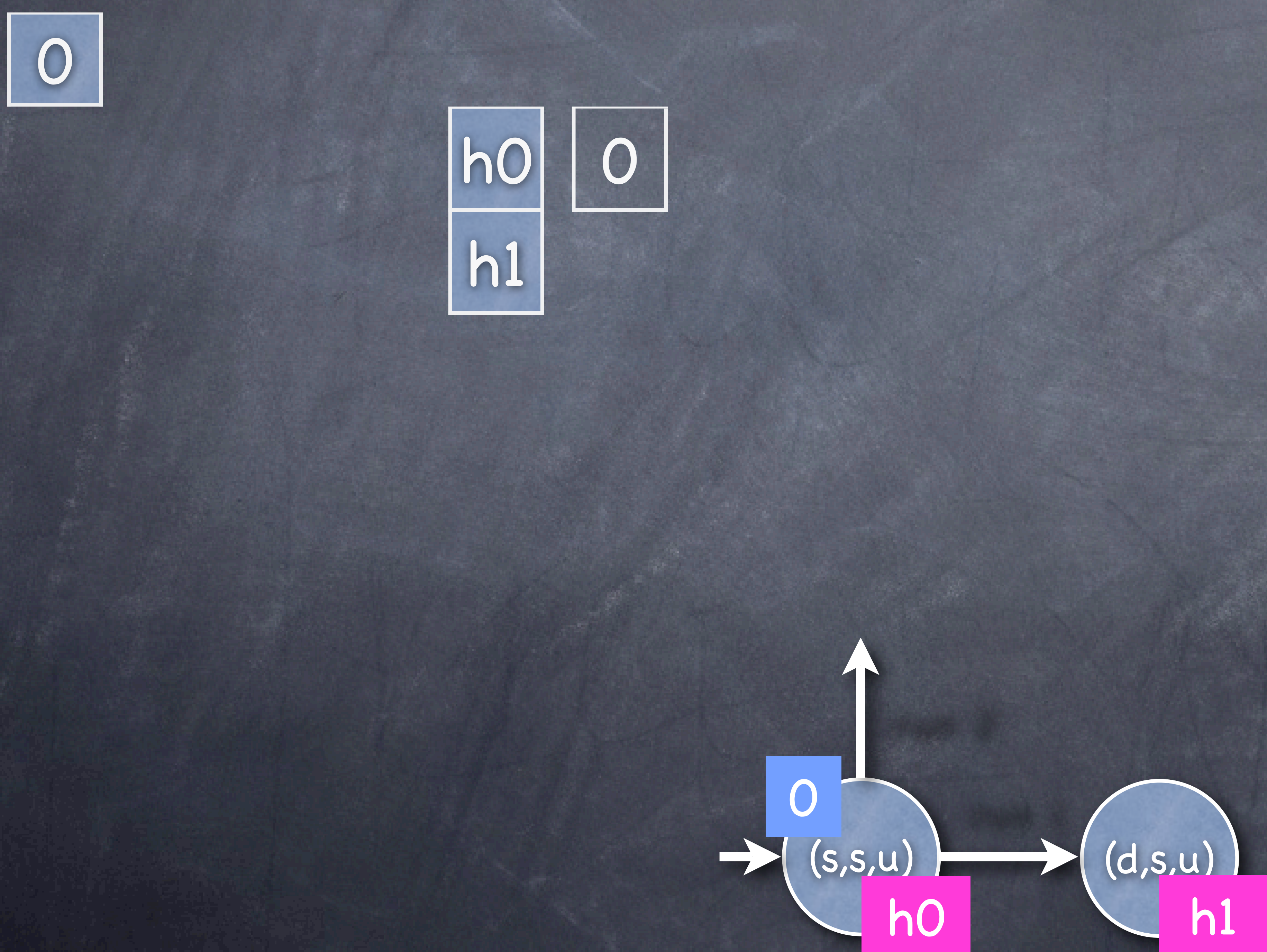
The ComBack Algorithm

0

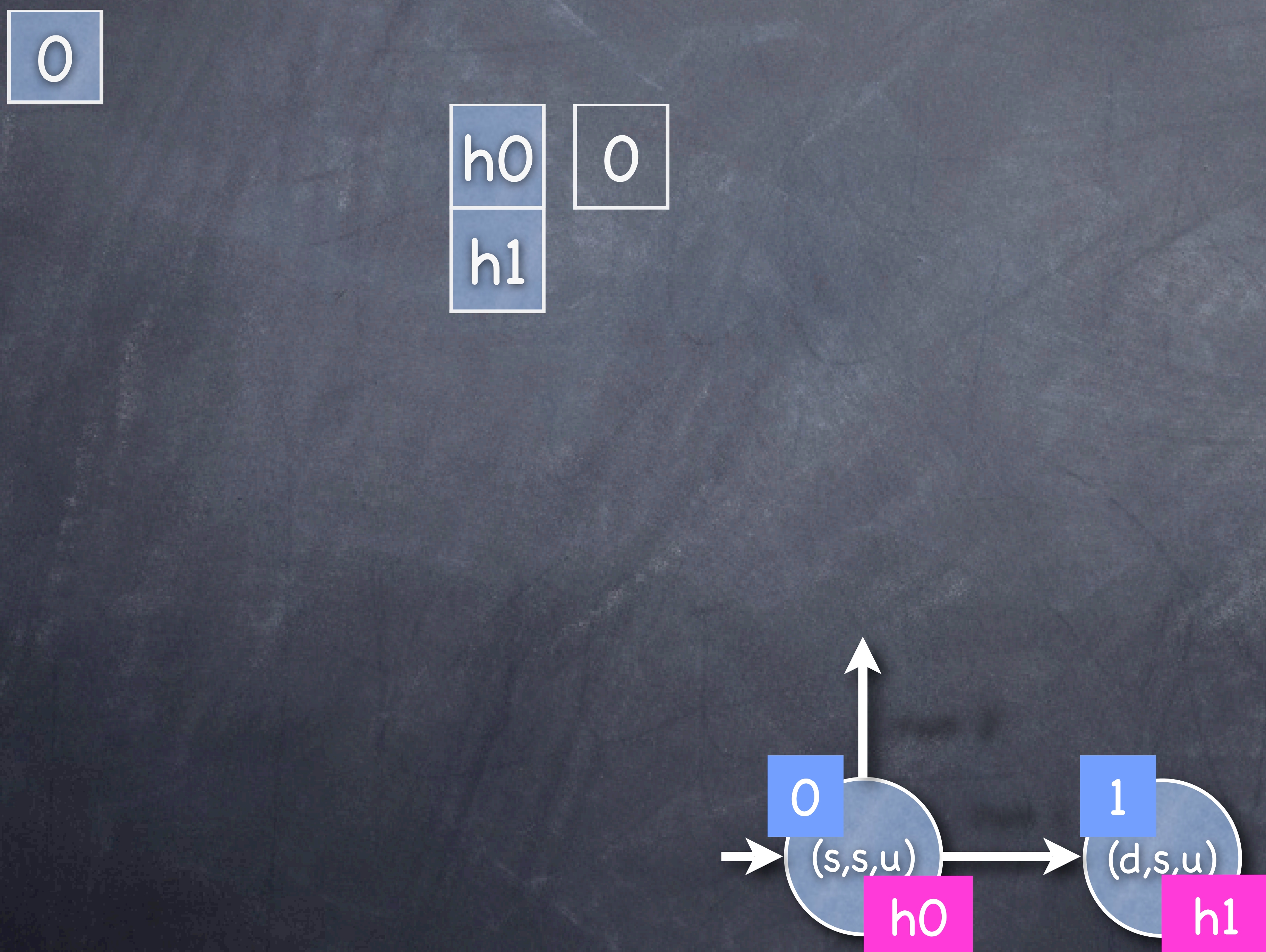
h0	0



The ComBack Algorithm

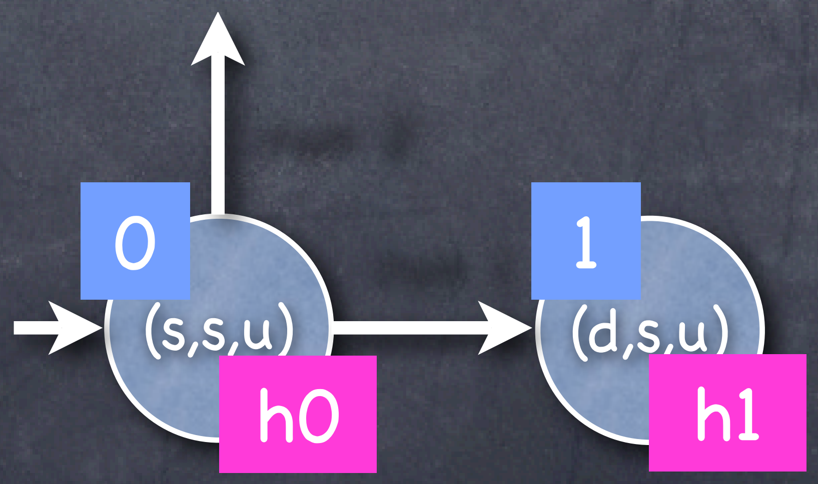


The ComBack Algorithm

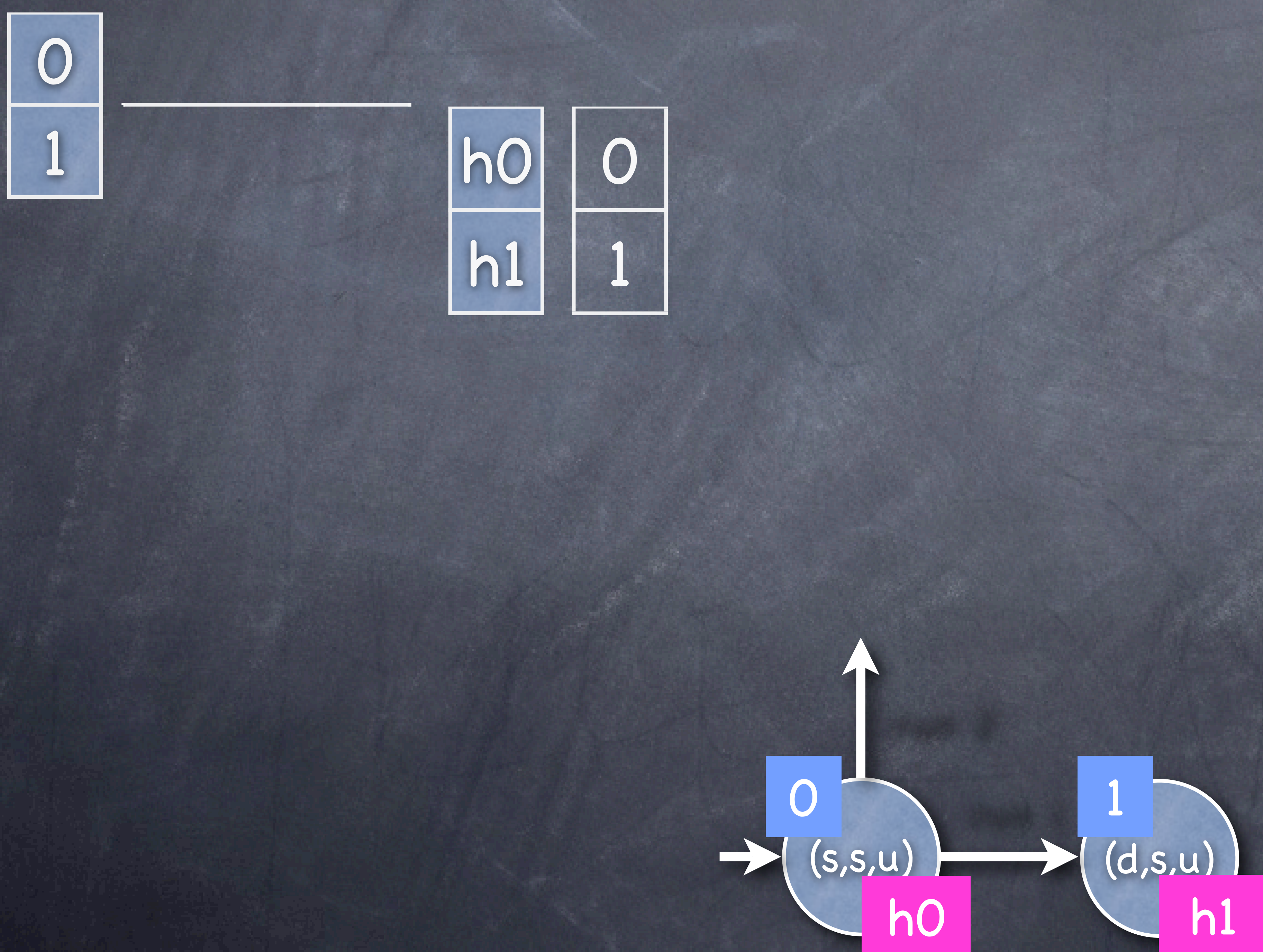


The ComBack Algorithm

0	
h0	0
h1	1



The ComBack Algorithm

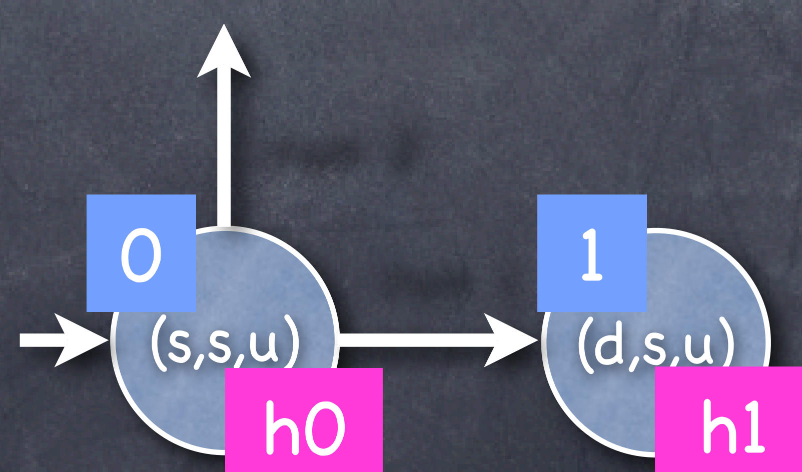


The ComBack Algorithm

0
1

run 1	0

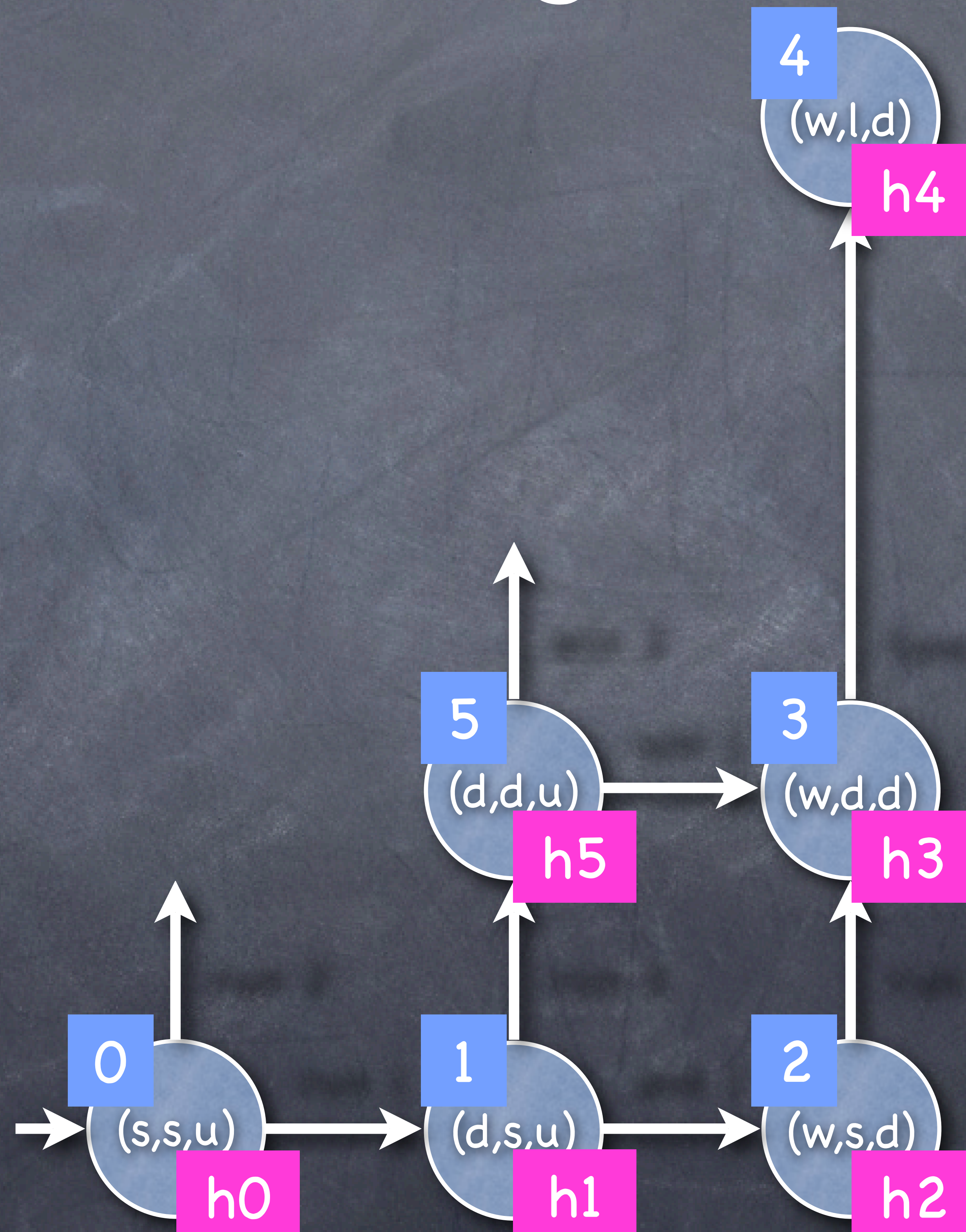
h0	0
h1	1



The ComBack Algorithm

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1

h0	0
h1	1
h2	2
h3	3
h4	4
h5	5



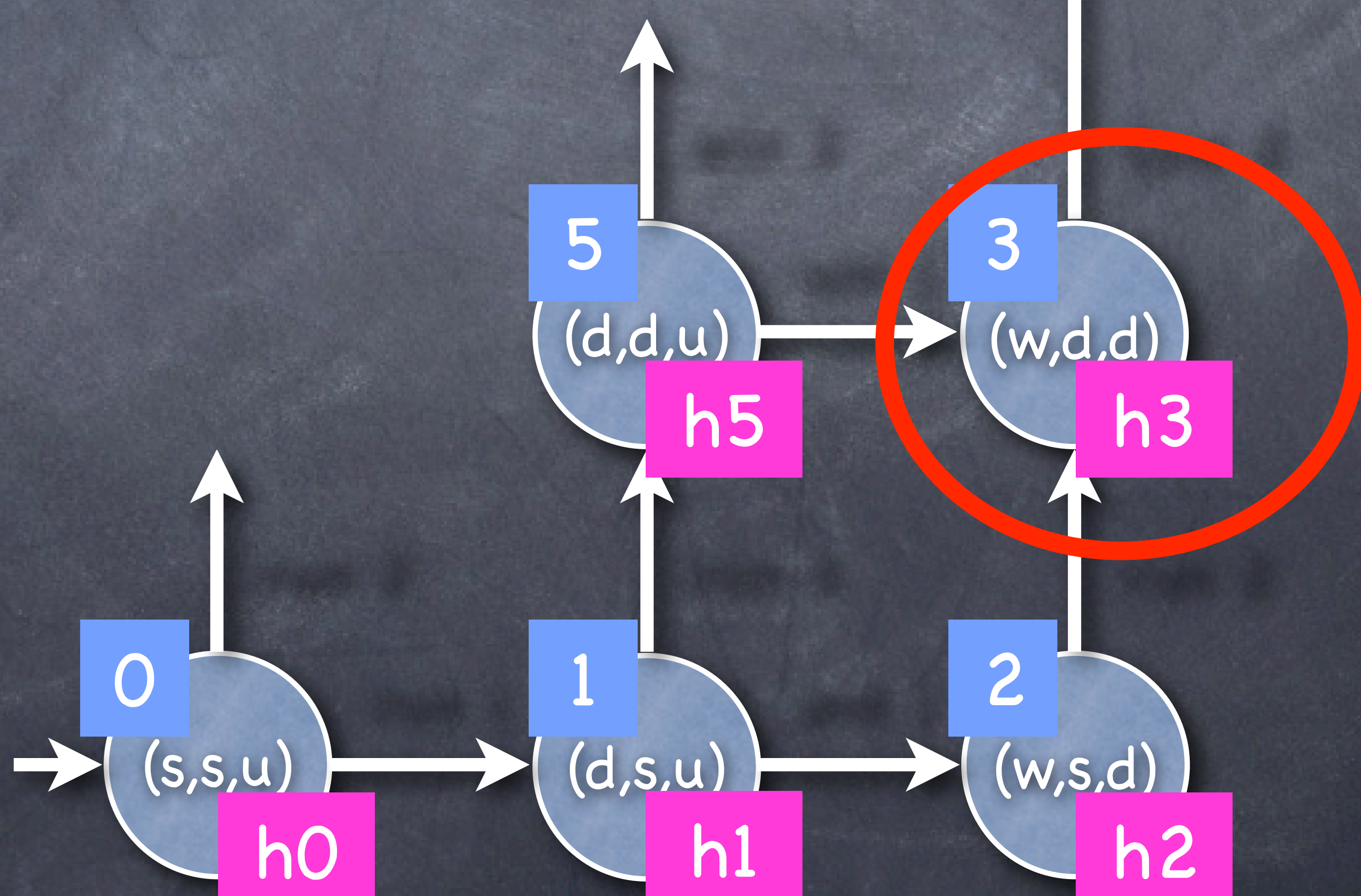
The Com

We rediscover the state (w,d,d) and compute the hash-value, h3...
How do we know, we have seen the state before (i.e. it is not a hash -collision)?

0
1
2
3
4
5

run 1	0
win 1	1
run 2	2
lose 2	3
run 2	1

h0	0
h1	1
h2	2
h3	3
h4	4
h5	5



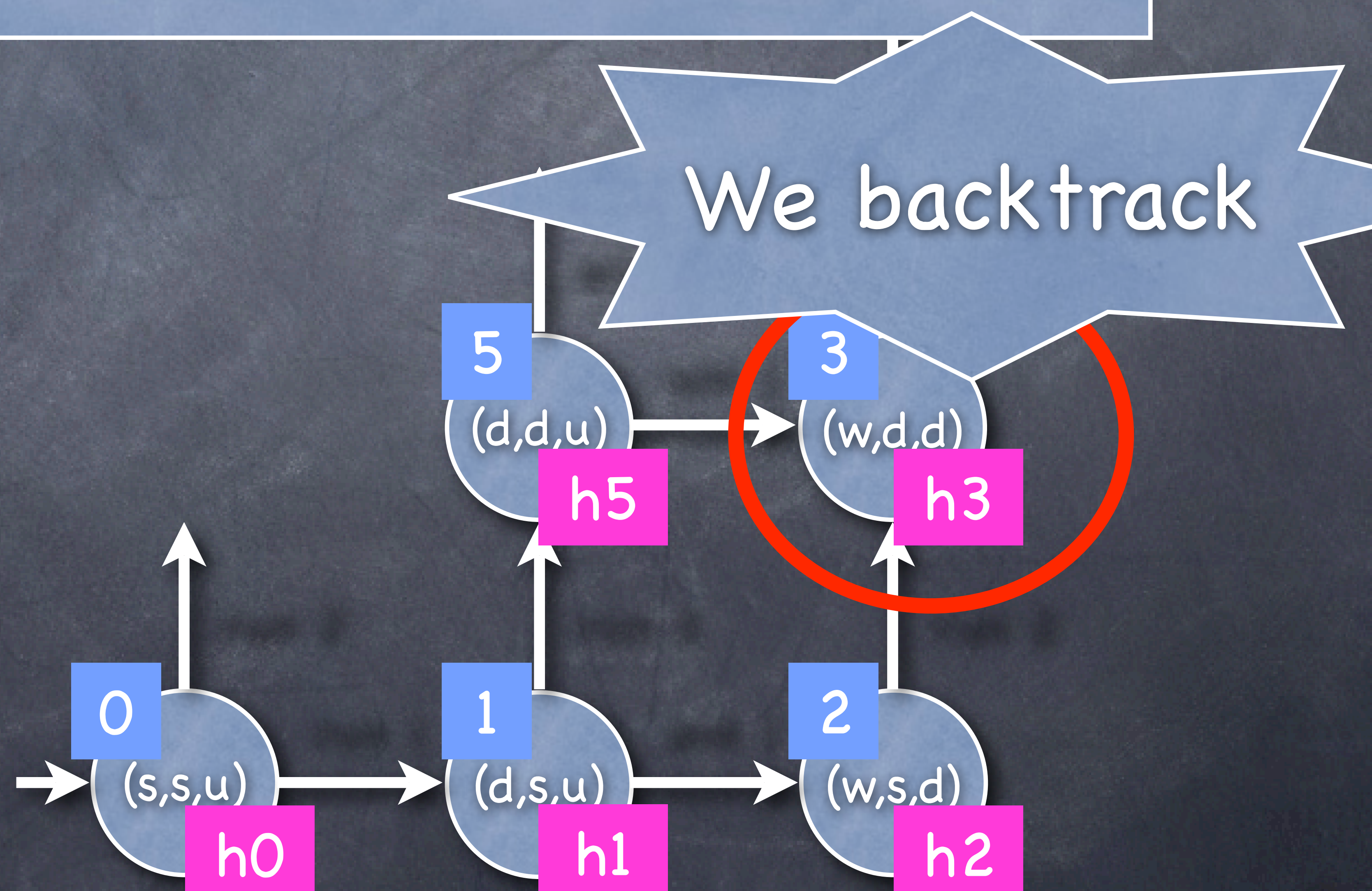
The Com

We rediscover the state (w,d,d) and compute the hash-value, h3...
How do we know, we have seen the state before (i.e. it is not a hash -collision)?

0
1
2
3
4
5

run 1	0
win 1	1
run 2	2
lose 2	3
run 2	1

h0	0
h1	1
h2	2
h3	3
h4	4
h5	5



The Com

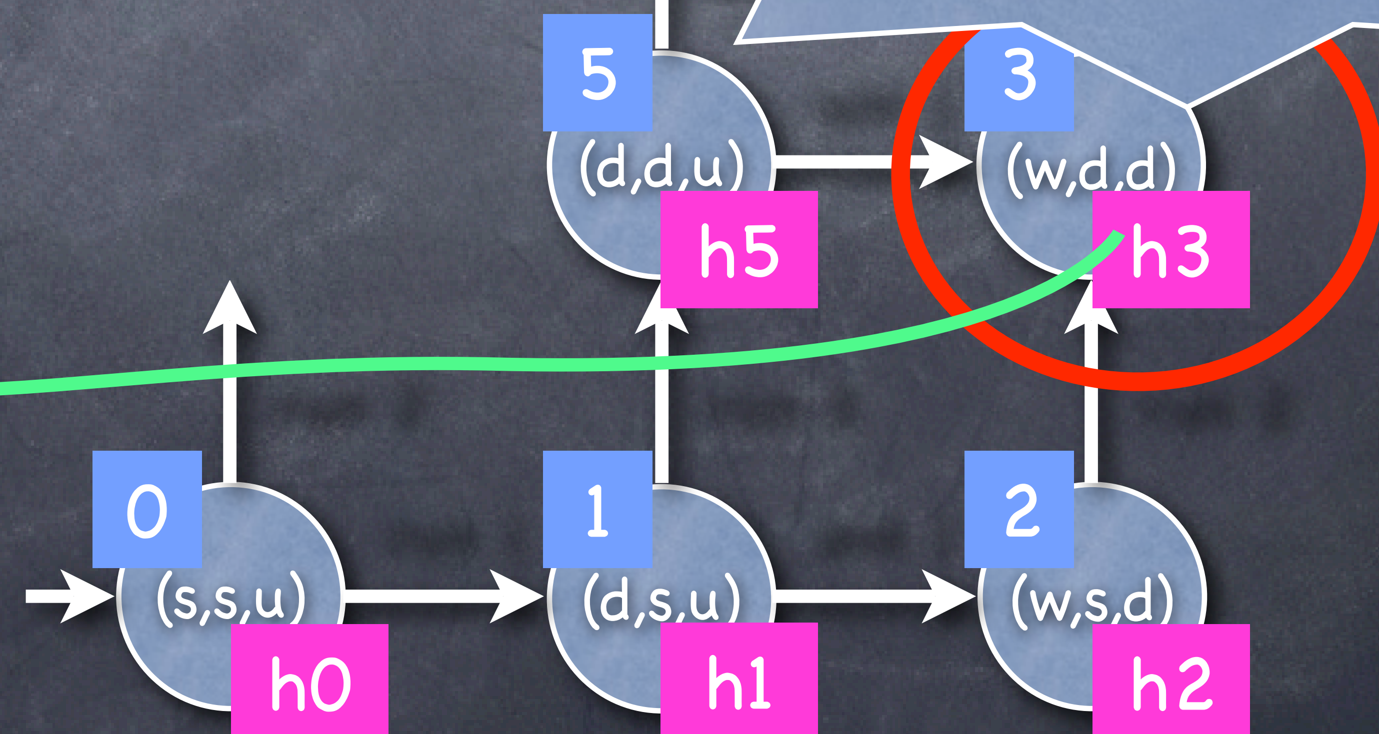
We rediscover the state (w,d,d) and compute the hash-value, h3...
How do we know, we have seen the state before (i.e. it is not a hash -collision)?

0
1
2
3
4
5

run 1	0
win 1	1
run 2	2
lose 2	3
run 2	1

h0	0
h1	1
h2	2
h3	3
h4	4
h5	5

We backtrack



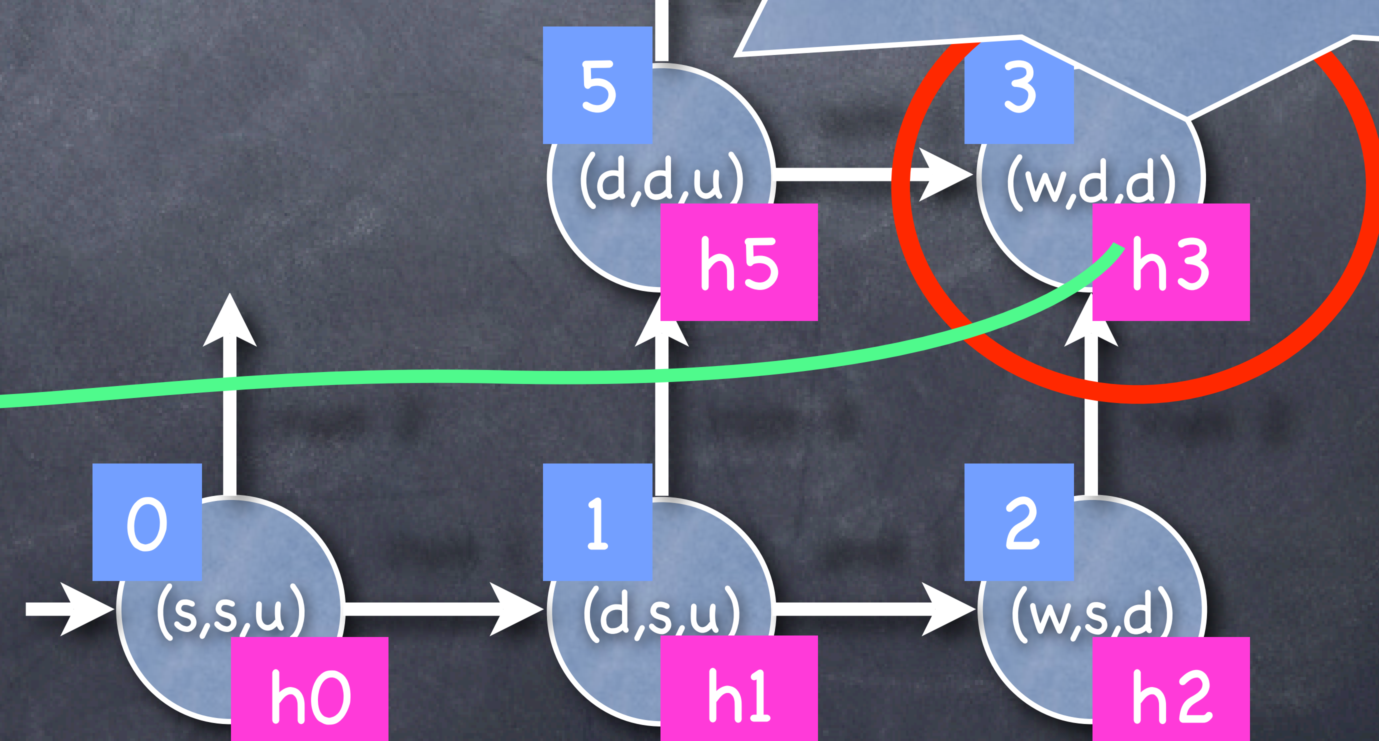
The Com

We rediscover the state (w,d,d) and compute the hash-value, h3...
How do we know, we have seen the state before (i.e. it is not a hash -collision)?

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1

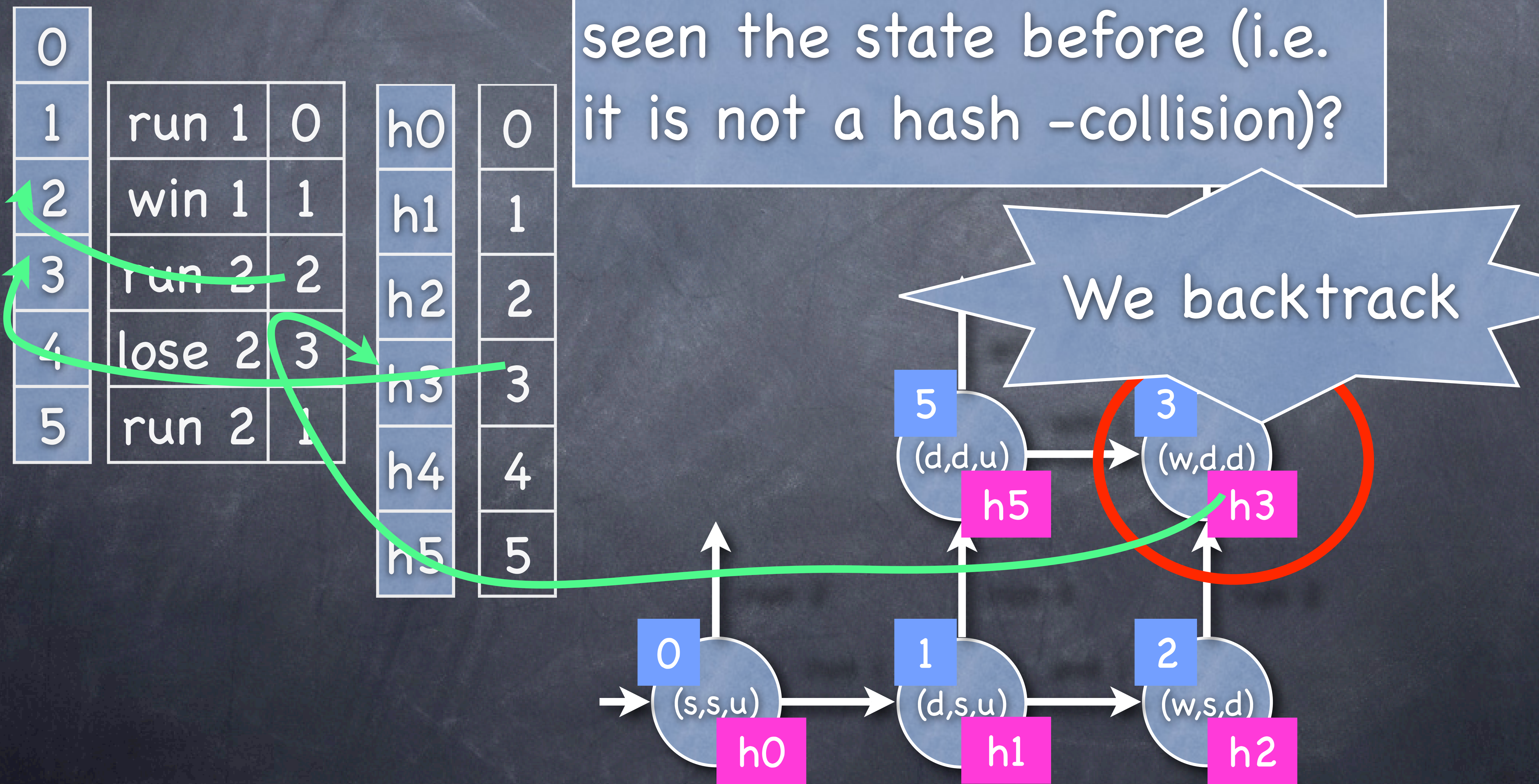
h0	0
h1	1
h2	2
h3	3
h4	4
h5	5

We backtrack



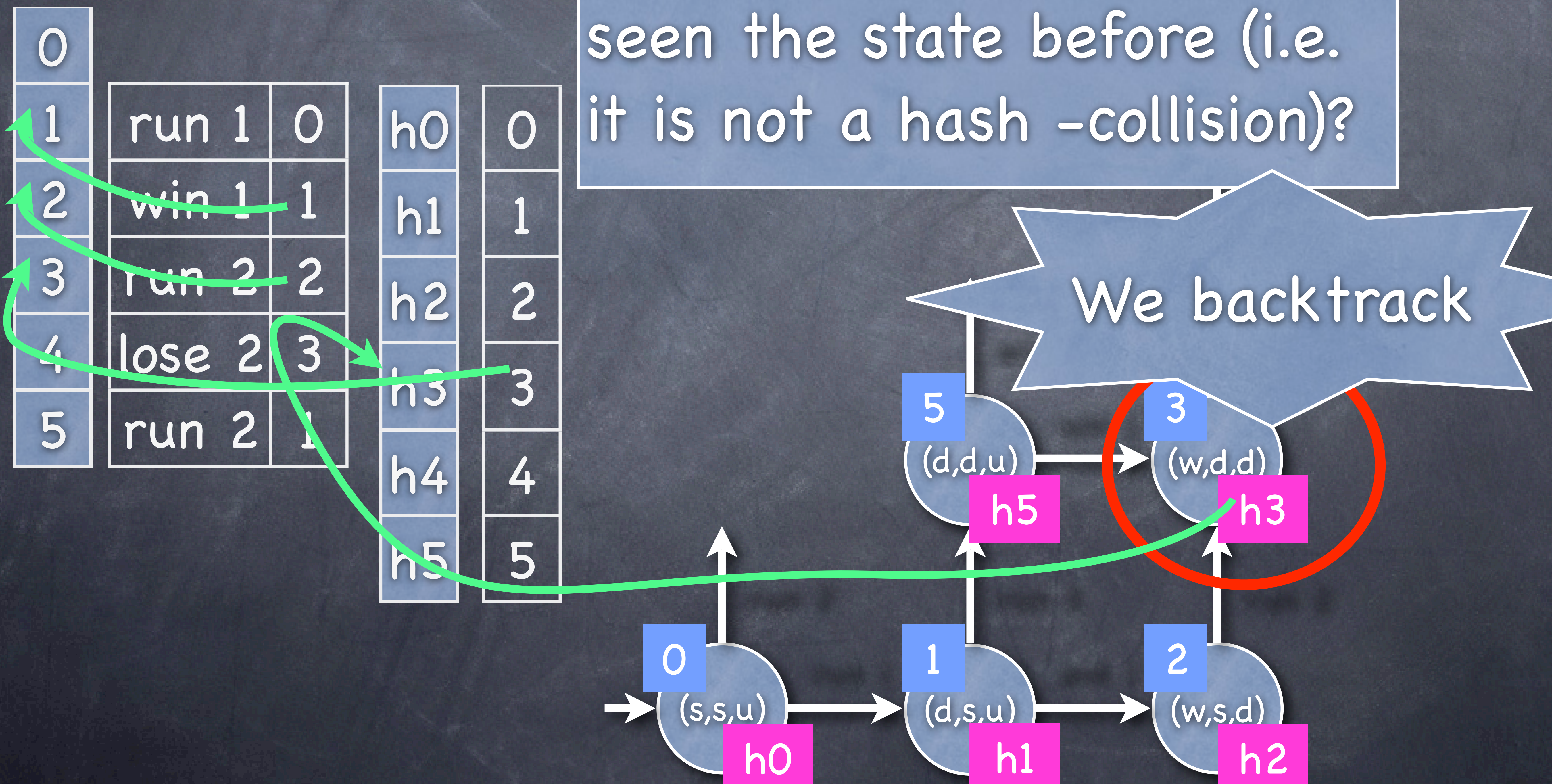
The Com

We rediscover the state (w,d,d) and compute the hash-value, h3...
How do we know, we have seen the state before (i.e. it is not a hash -collision)?



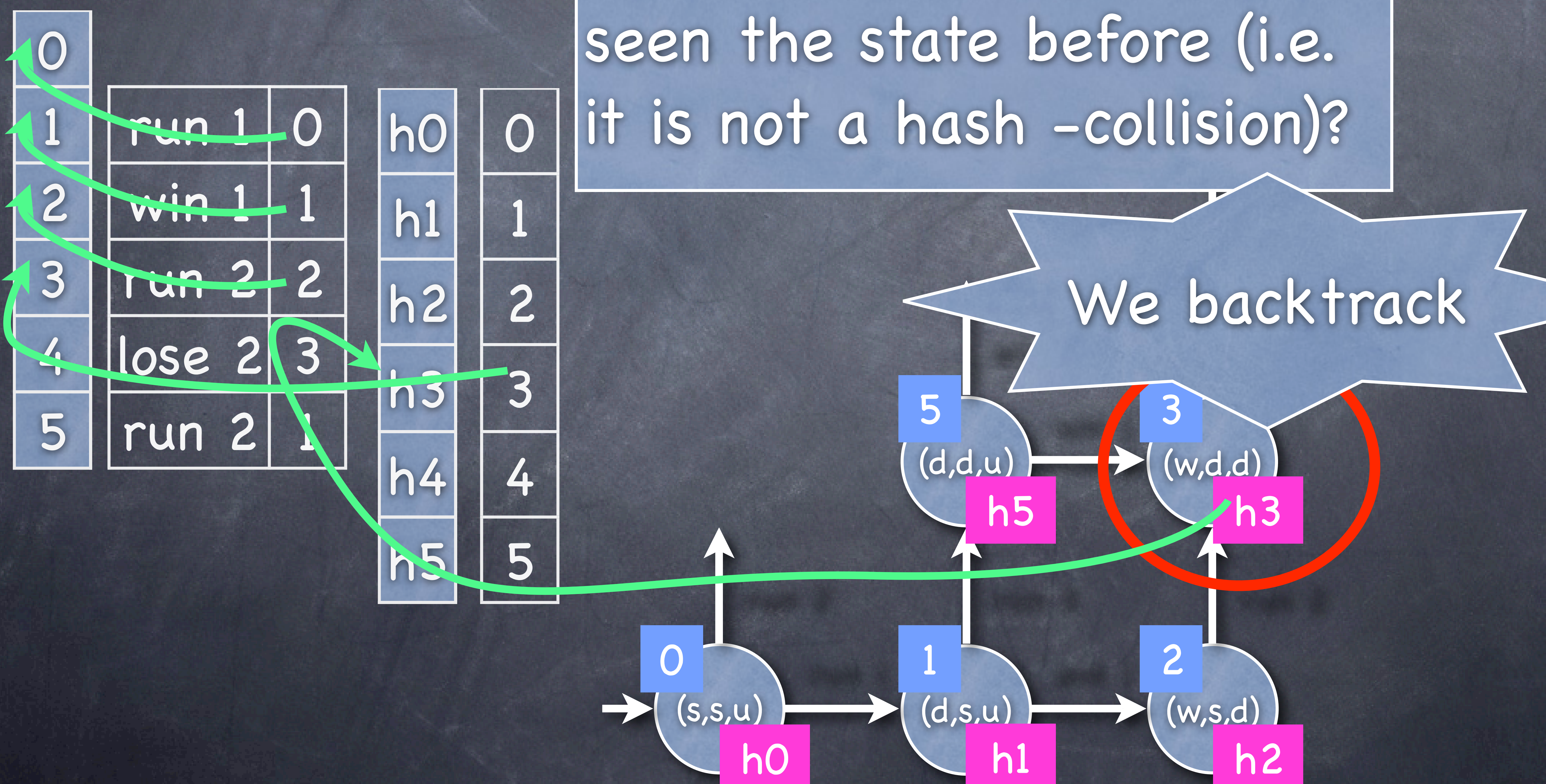
The Com

We rediscover the state (w,d,d) and compute the hash-value, h3...
How do we know, we have seen the state before (i.e. it is not a hash -collision)?



The Com

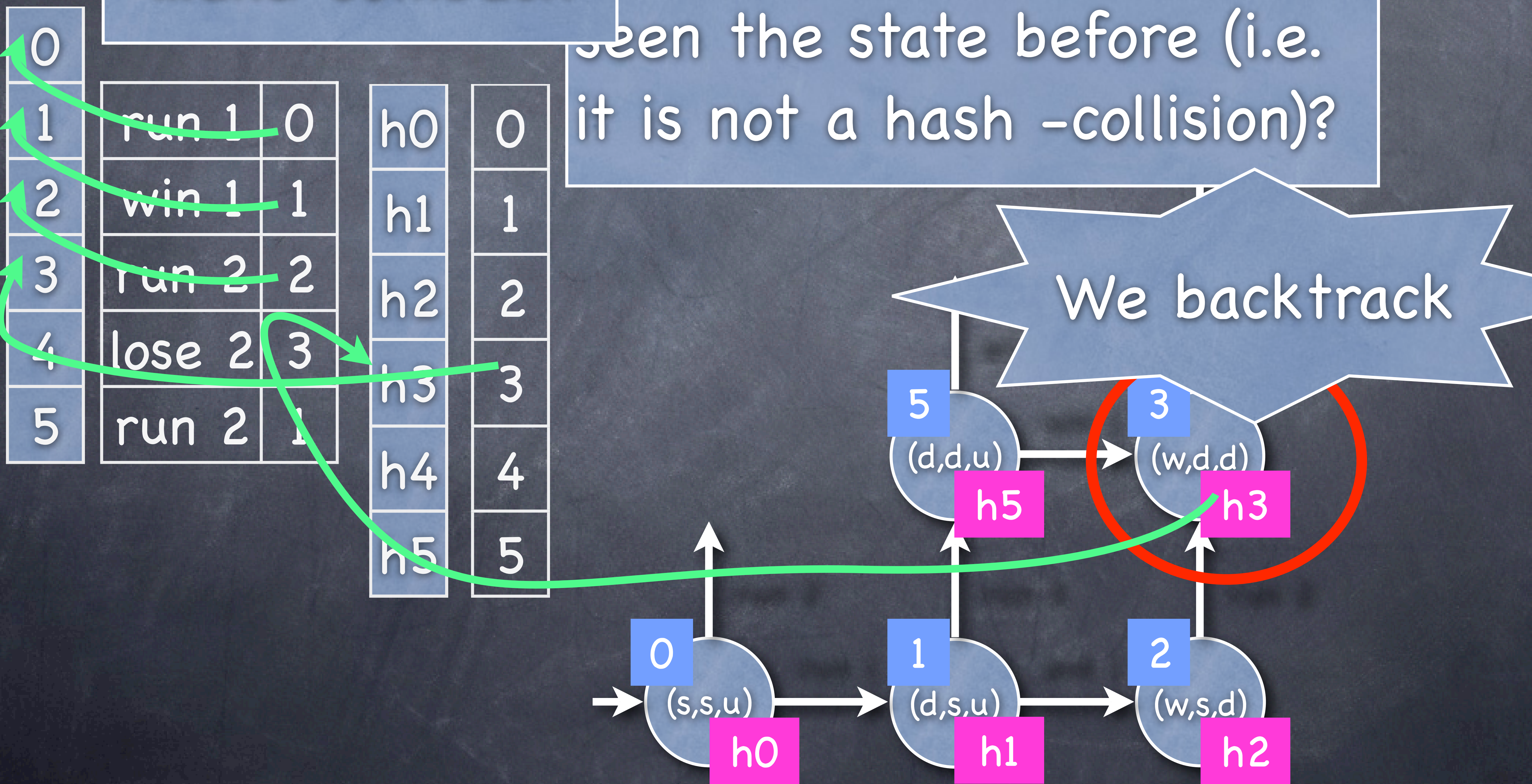
We rediscover the state (w,d,d) and compute the hash-value, h3...
How do we know, we have seen the state before (i.e. it is not a hash -collision)?



The Com

We rediscover the state (w,d,d) and compute the hash-value, h3...
How do we know, we have seen the state before (i.e. it is not a hash -collision)?

...and ComBack



The Com

We rediscover the state (w,d,d) and compute the hash-value, h3...
ow do we know, we have seen the state before (i.e. it is not a hash -collision)?

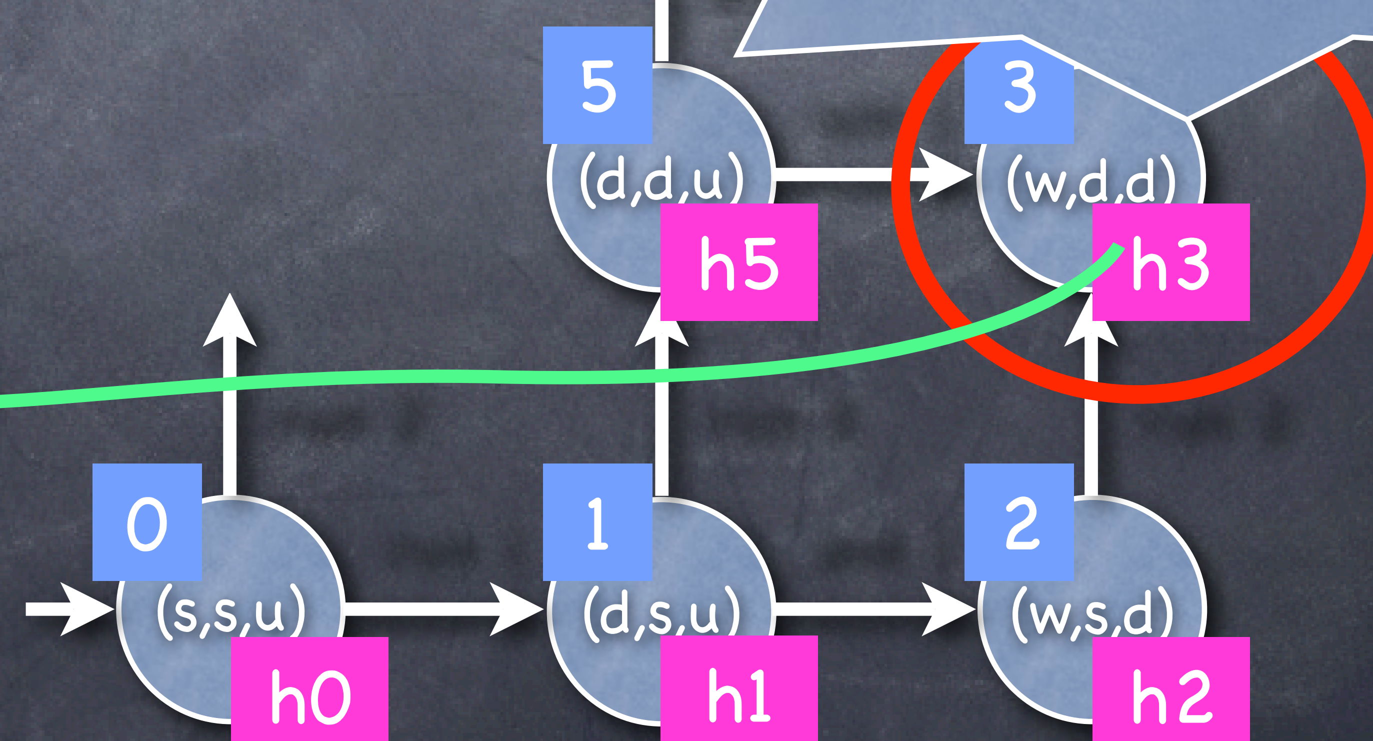
...and ComBack

0
1
2
3
4
5

run 1	0
win 1	1
run 2	2
lose 2	3
run 2	1

h0	0
h1	1
h2	2
h3	3
h4	4
h5	5

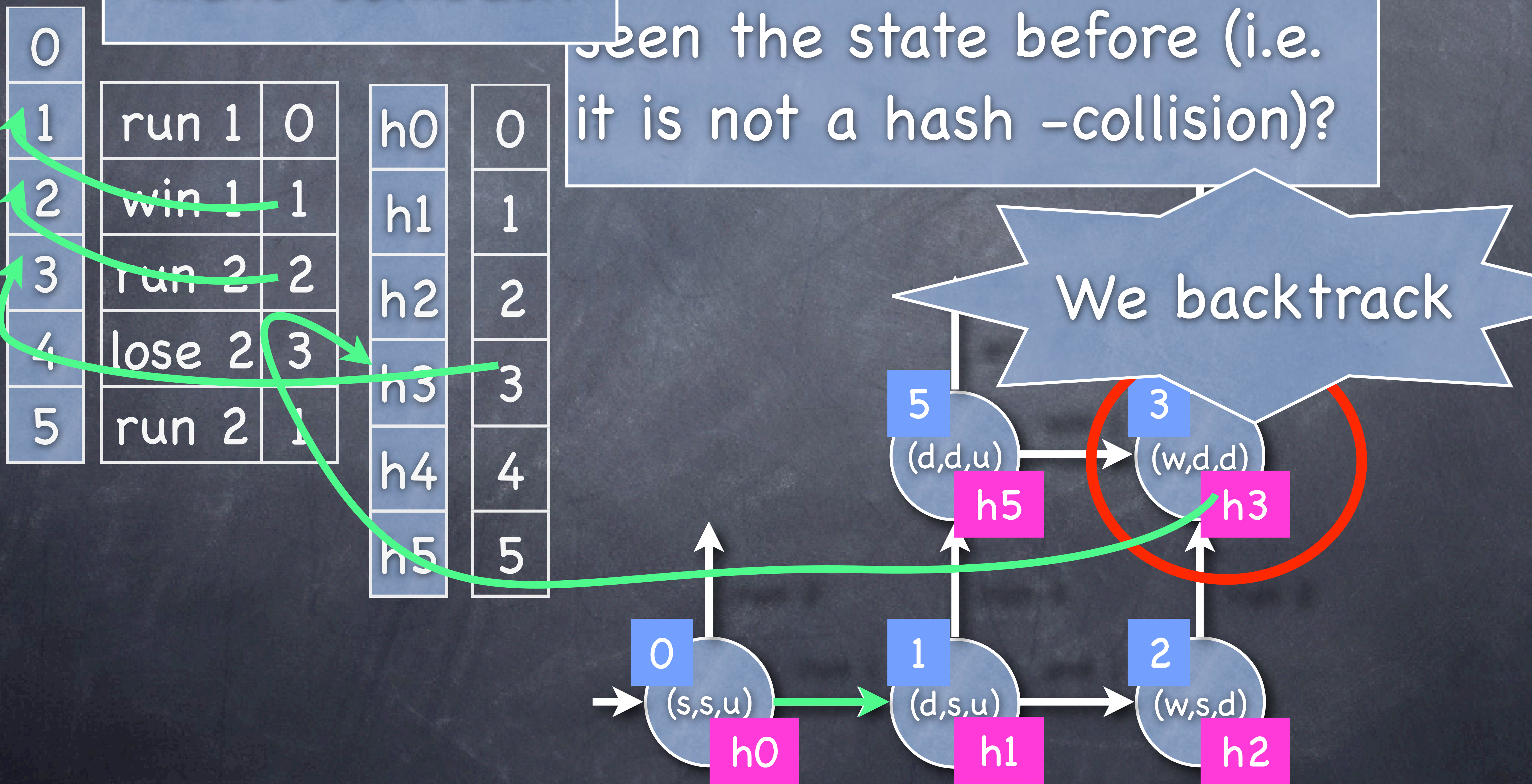
We backtrack



The Com

We rediscover the state (w,d,d) and compute the hash-value, h3...
ow do we know, we have
een the state before (i.e. it is not a hash -collision)?

...and ComBack



The Com

We rediscover the state (w,d,d) and compute the hash-value, h3...
ow do we know, we have seen the state before (i.e. it is not a hash -collision)?

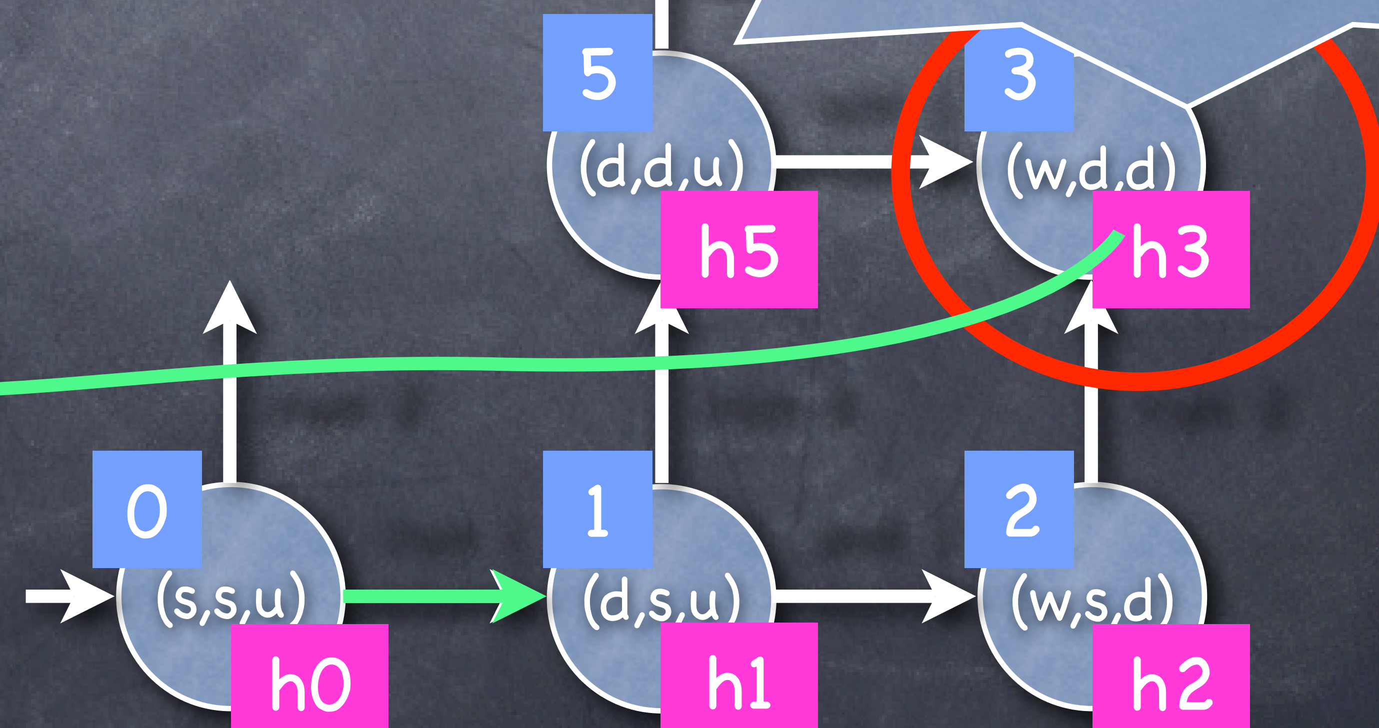
...and ComBack

0
1
2
3
4
5

run 1	0
win 1	1
run 2	2
lose 2	3
run 2	1

h0	0
h1	1
h2	2
h3	3
h4	4
h5	5

We backtrack



The Com

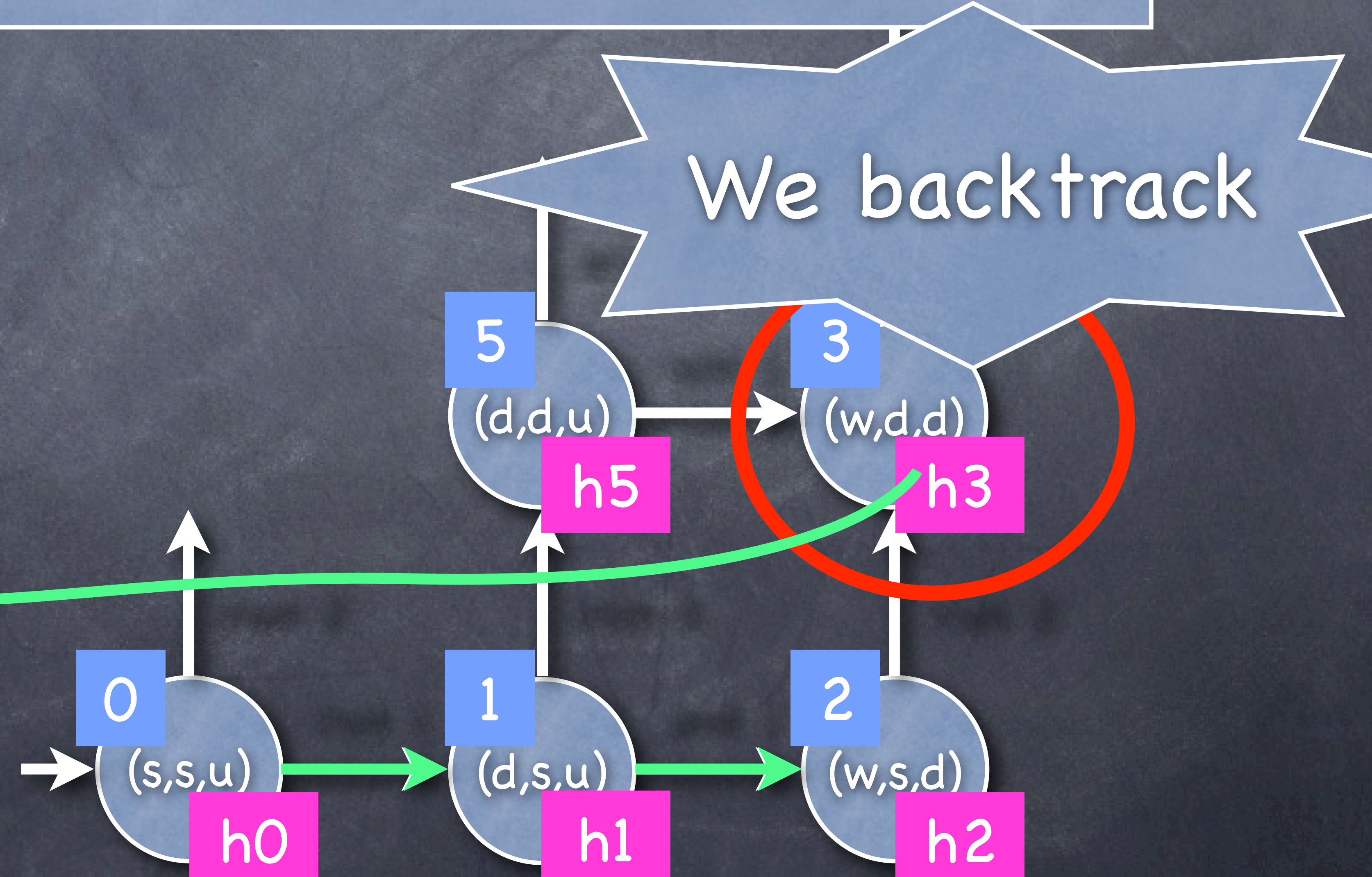
...and ComBack

We rediscover the state (w,d,d) and compute the hash-value, h3...
ow do we know, we have seen the state before (i.e. it is not a hash -collision)?

0
1
2
3
4
5

run 1	0
win 1	1
run 2	2
lose 2	3
run 2	1

h0	0
h1	1
h2	2
h3	3
h4	4
h5	5



The Com

We rediscover the state (w,d,d) and compute the hash-value, h3...
How do we know, we have seen the state before (i.e. it is not a hash -collision)?

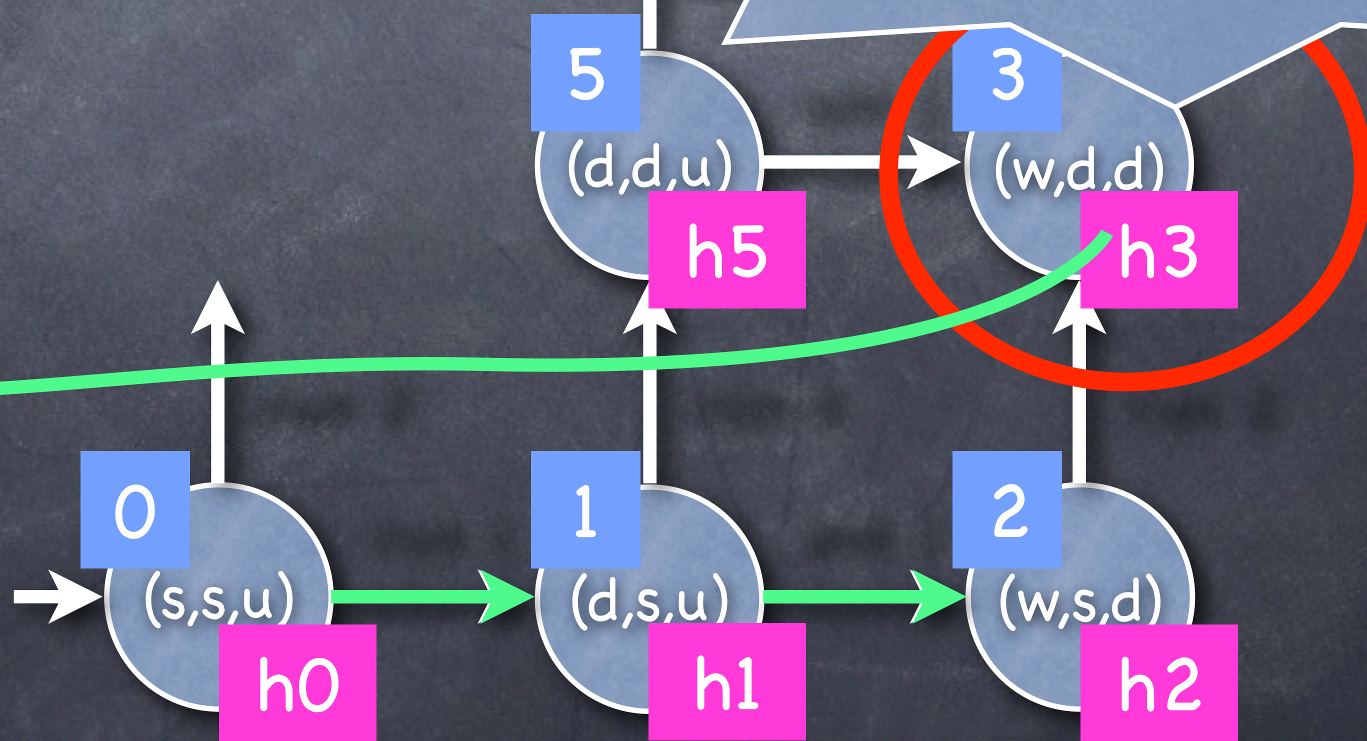
...and ComBack

0
1
2
3
4
5

run 1	0
win 1	1
run 2	2
lose 2	3
run 2	1

h0	0
h1	1
h2	2
h3	3
h4	4
h5	5

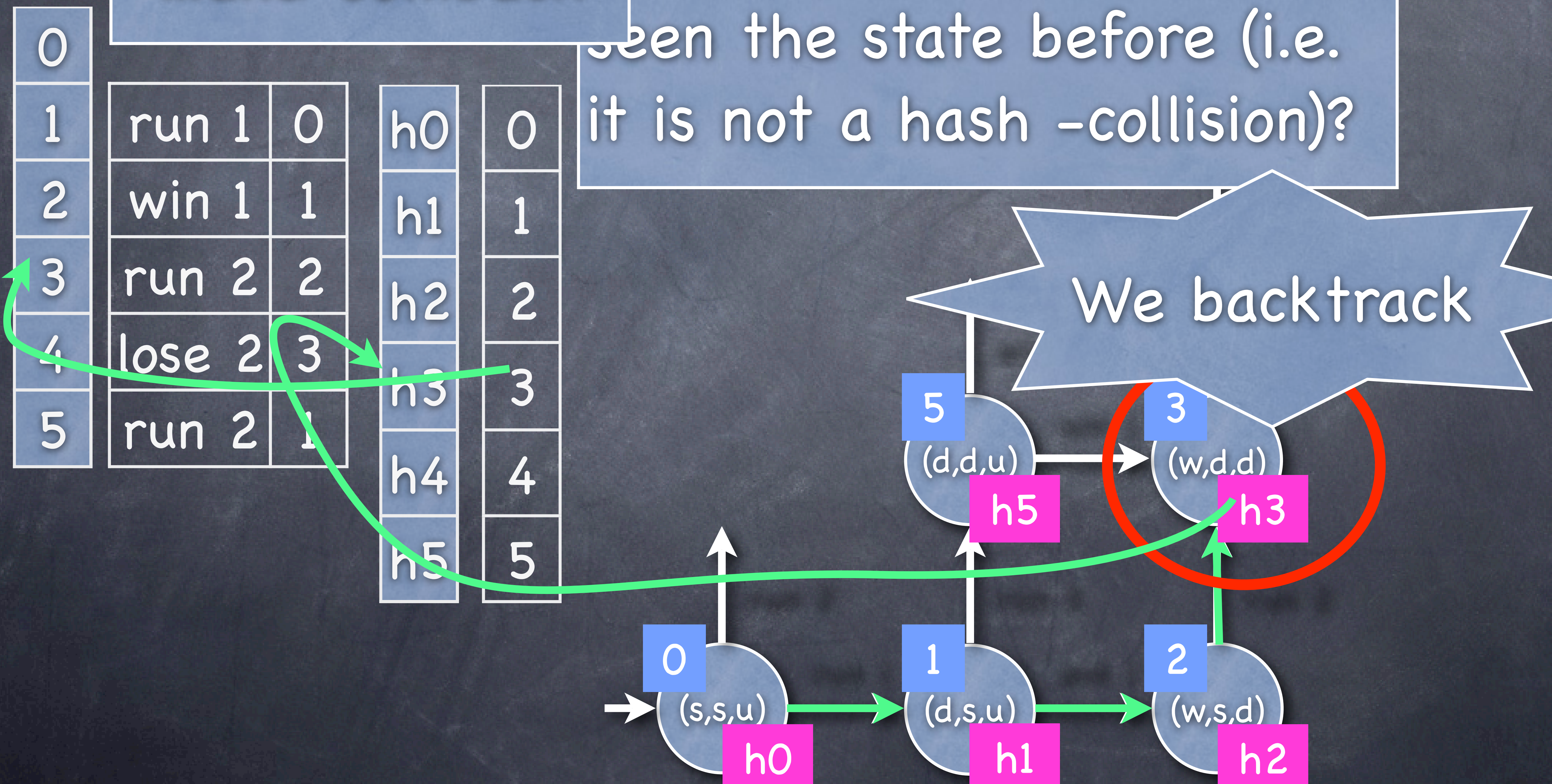
We backtrack



The Com

...and ComBack

We rediscover the state (w,d,d) and compute the hash-value, h3...
ow do we know, we have seen the state before (i.e. it is not a hash -collision)?



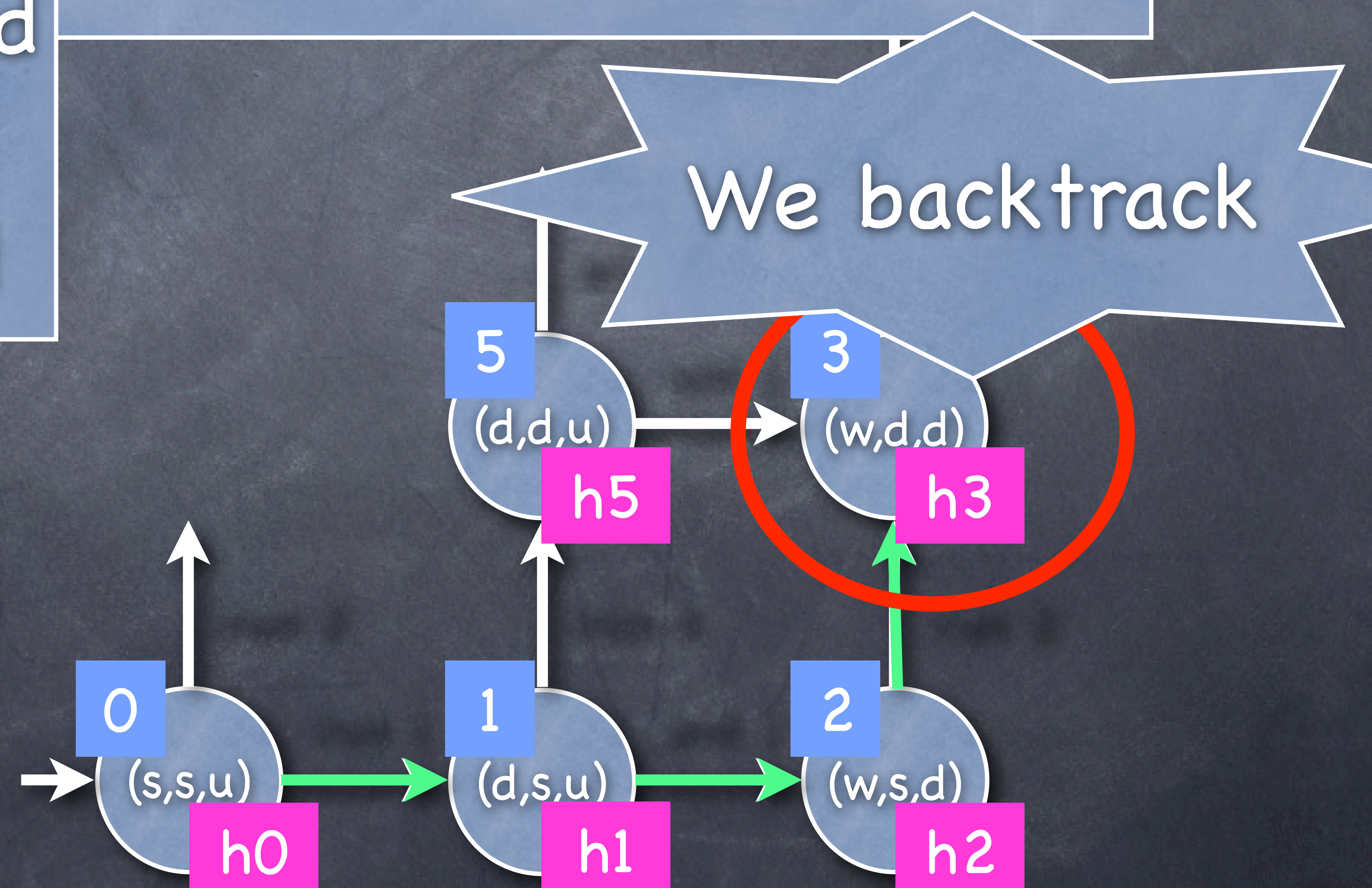
The Com

We rediscover the state (w,d,d) and compute the hash-value, h3...
ow do we know, we have een the state before (i.e. is not a hash -collision)?

...and ComBack

...and notice we have (re-)arrived at (w,d,d). No need to proceed

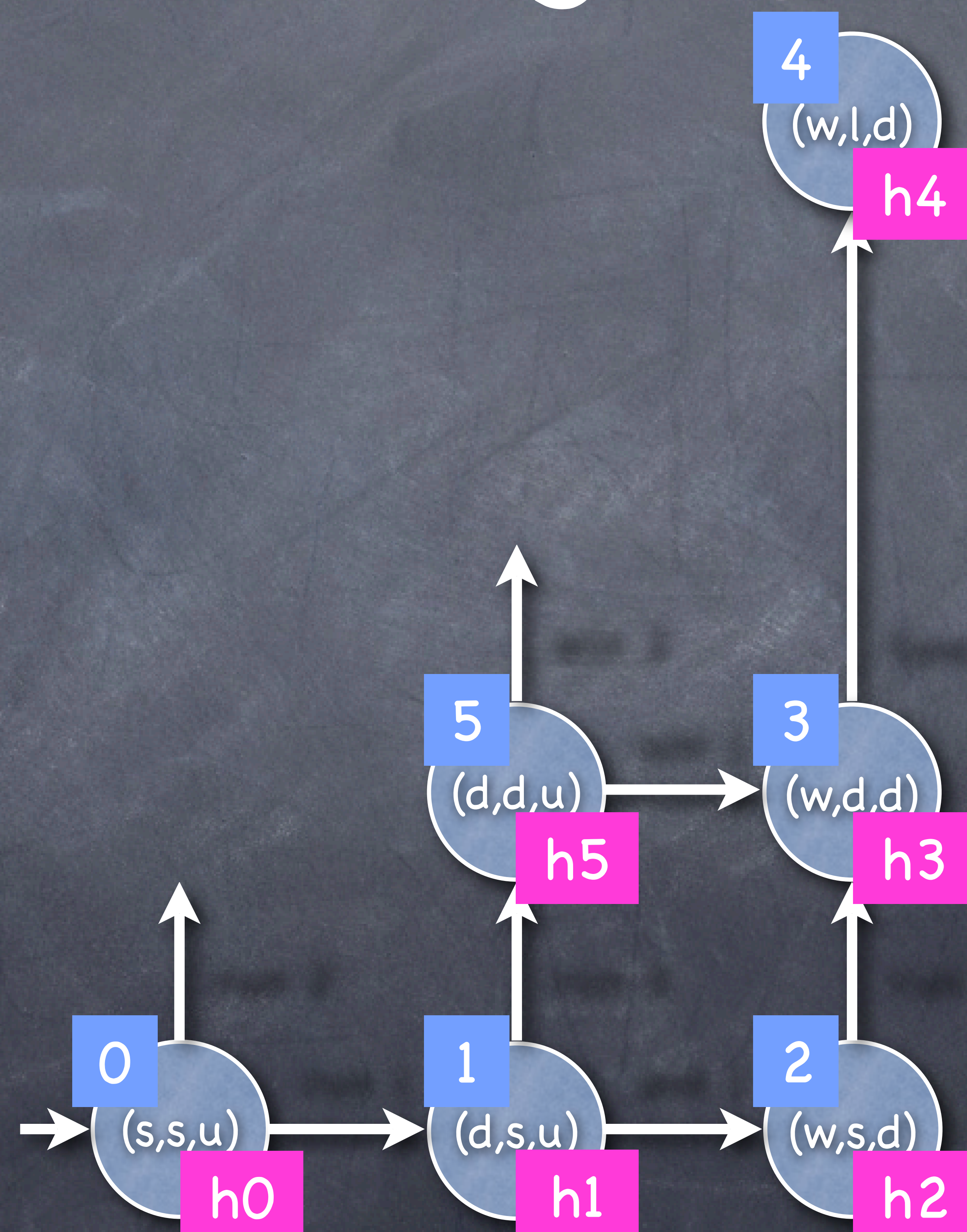
0					
1					
2					
3					
4				h4	4
5	run 2	1		h5	5



The ComBack Algorithm

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1

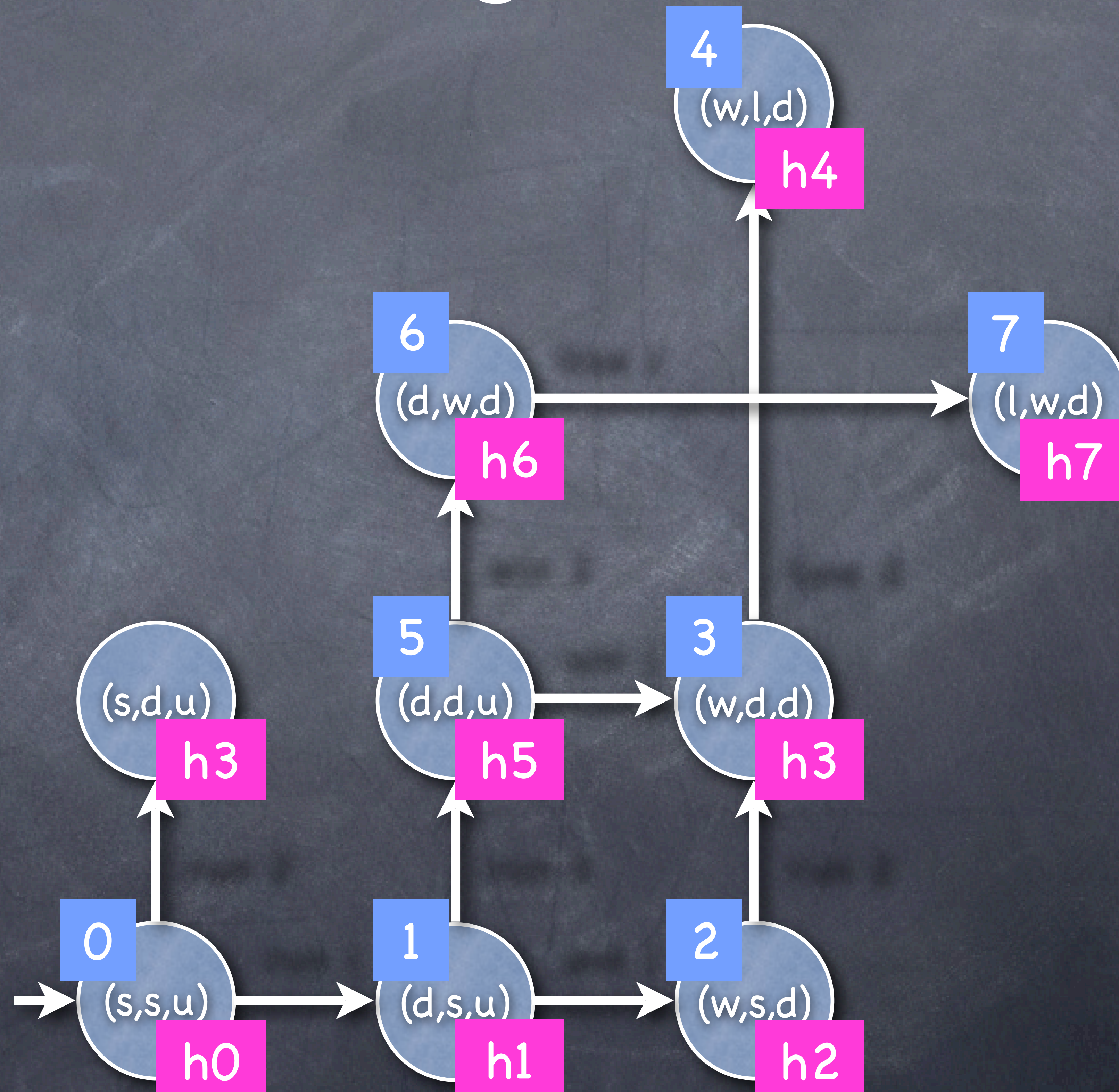
h0	0
h1	1
h2	2
h3	3
h4	4
h5	5



The ComBack Algorithm

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1
6	win 2	5
7	lose 1	6

h0	0
h1	1
h2	2
h3	3
h4	4
h5	5
h6	6
h7	7

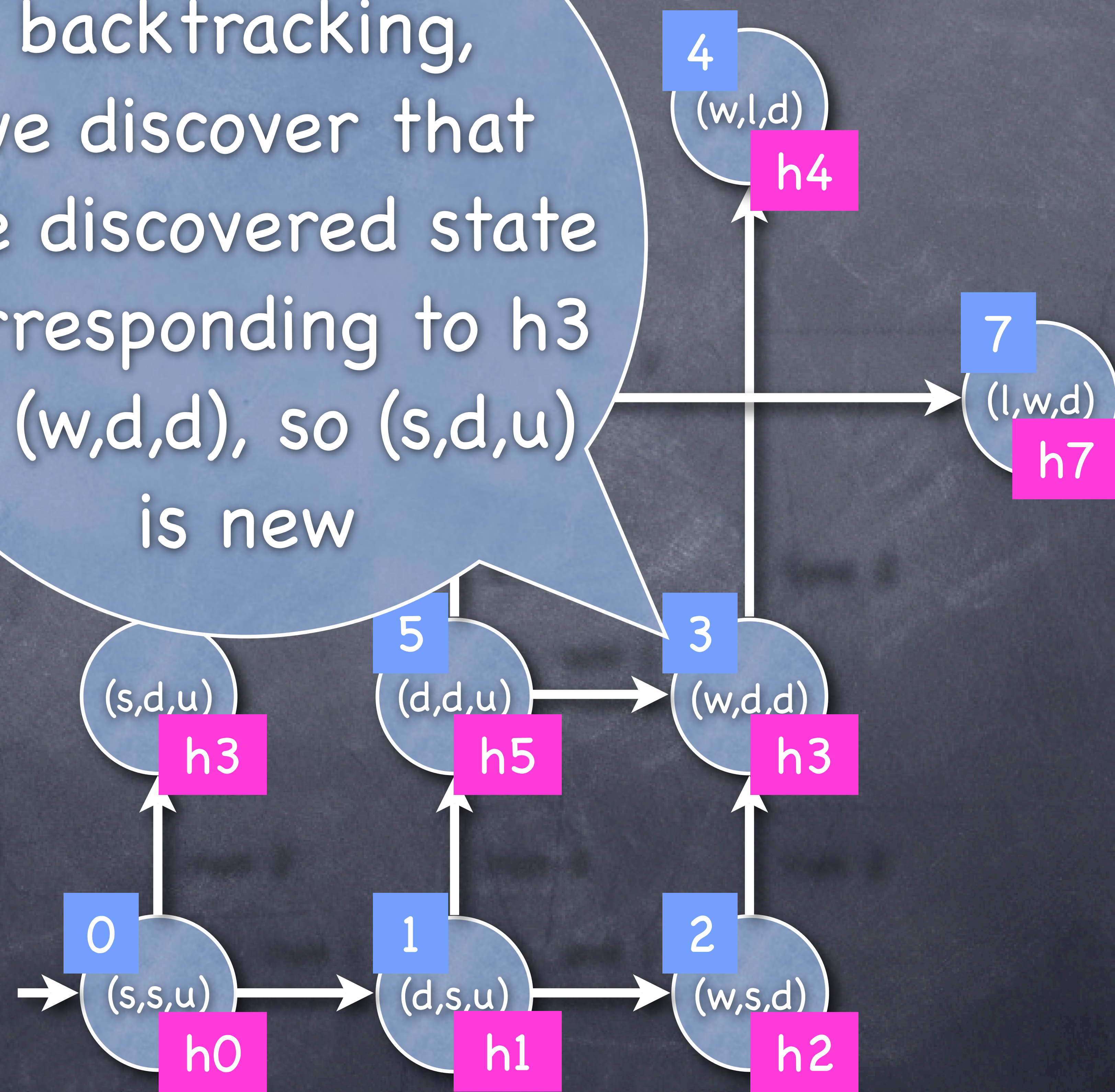


The Com gorithm

By backtracking, we discover that the discovered state corresponding to h3 is (w,d,d), so (s,d,u) is new

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1
6	win 2	5
7	lose 1	6

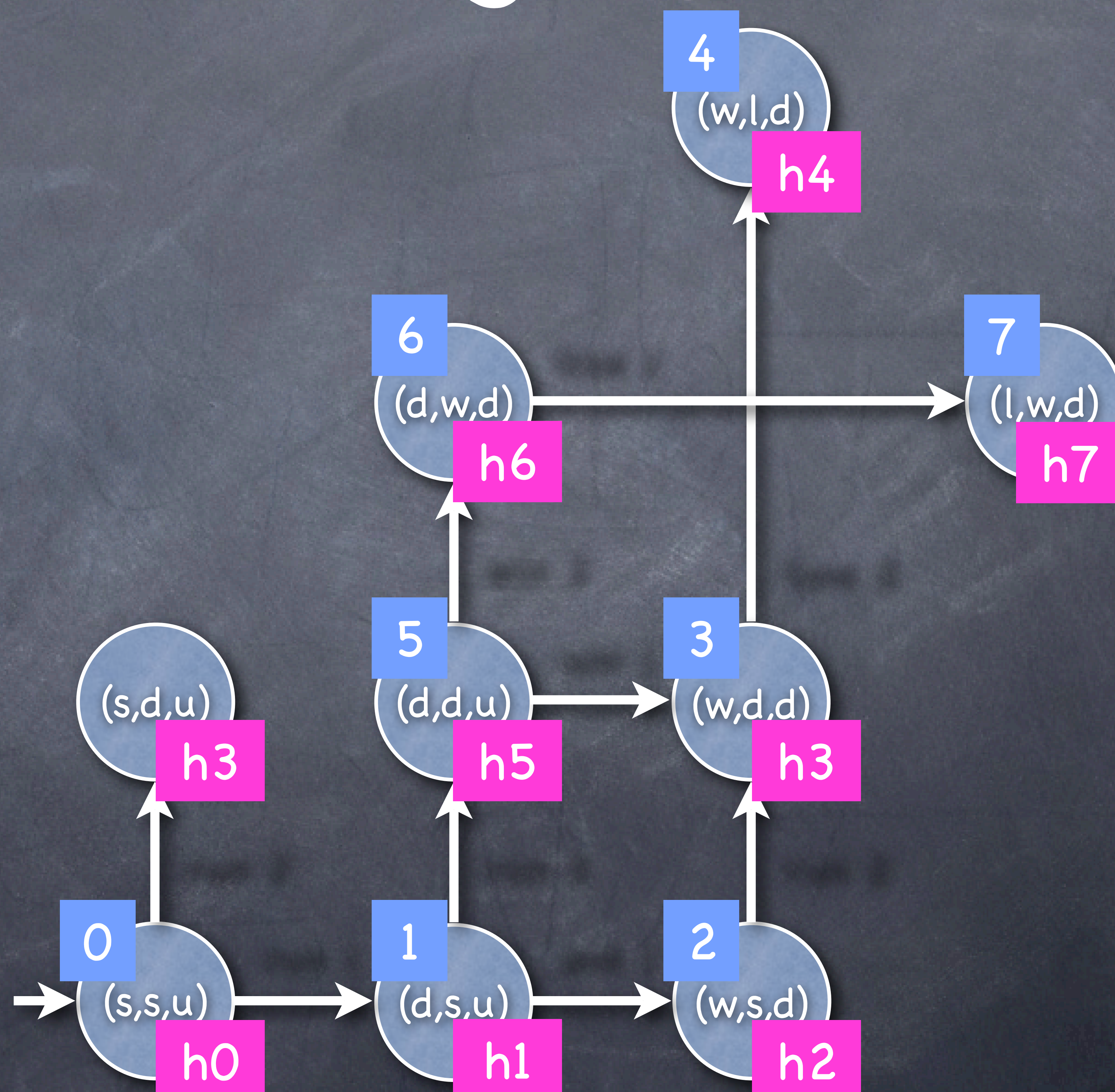
h0	
h1	
h2	2
h3	3
h4	4
h5	5
h6	6
h7	7



The ComBack Algorithm

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1
6	win 2	5
7	lose 1	6

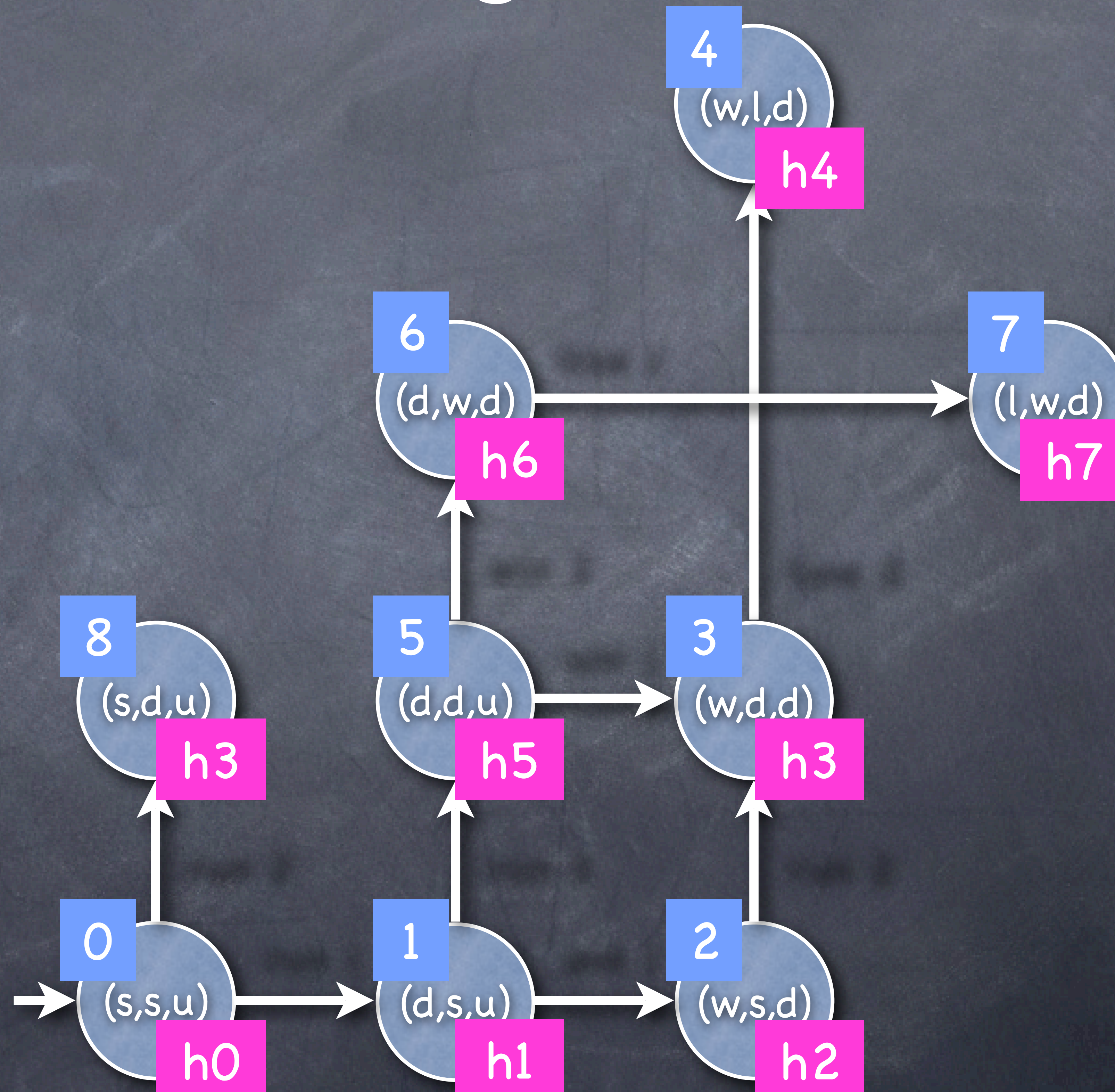
h0	0
h1	1
h2	2
h3	3
h4	4
h5	5
h6	6
h7	7



The ComBack Algorithm

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1
6	win 2	5
7	lose 1	6

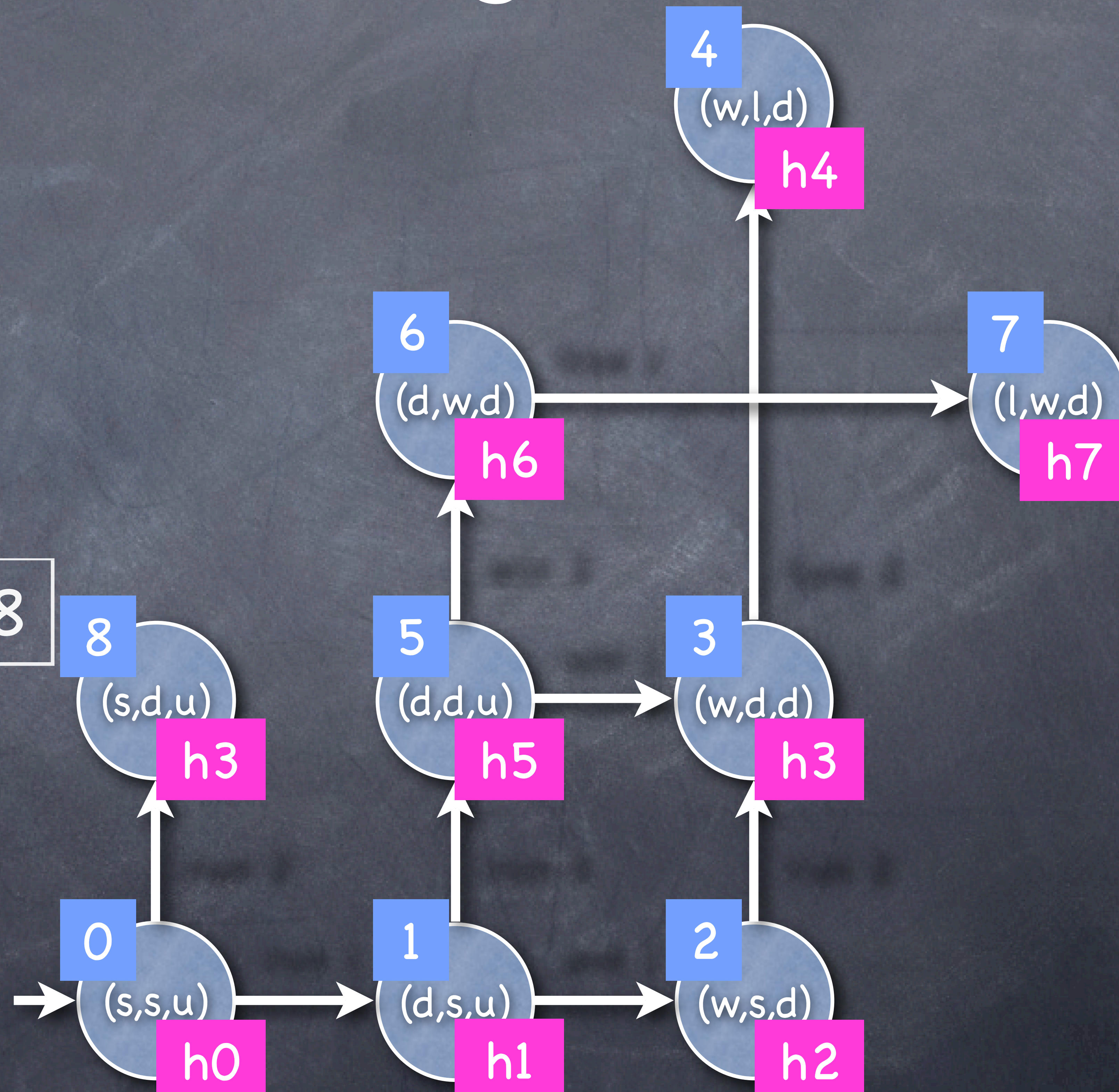
h0	0
h1	1
h2	2
h3	3
h4	4
h5	5
h6	6
h7	7



The ComBack Algorithm

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1
6	win 2	5
7	lose 1	6

h0	0	
h1	1	
h2	2	
h3	3	8
h4	4	
h5	5	
h6	6	
h7	7	

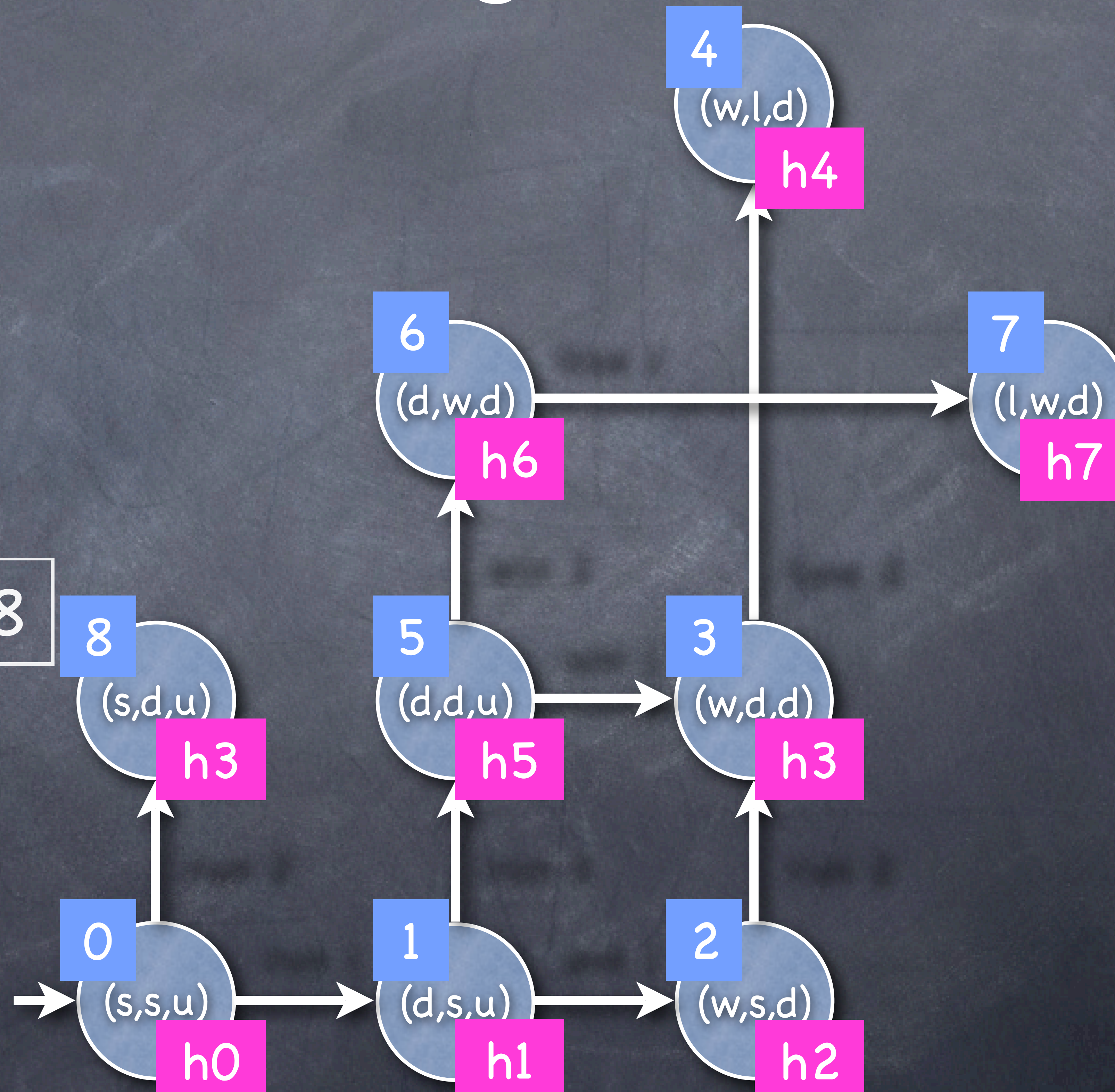


The ComBack Algorithm

0
1
2
3
4
5
6
7
8

run 1	0
win 1	1
run 2	2
lose 2	3
run 2	1
win 2	5
lose 1	6

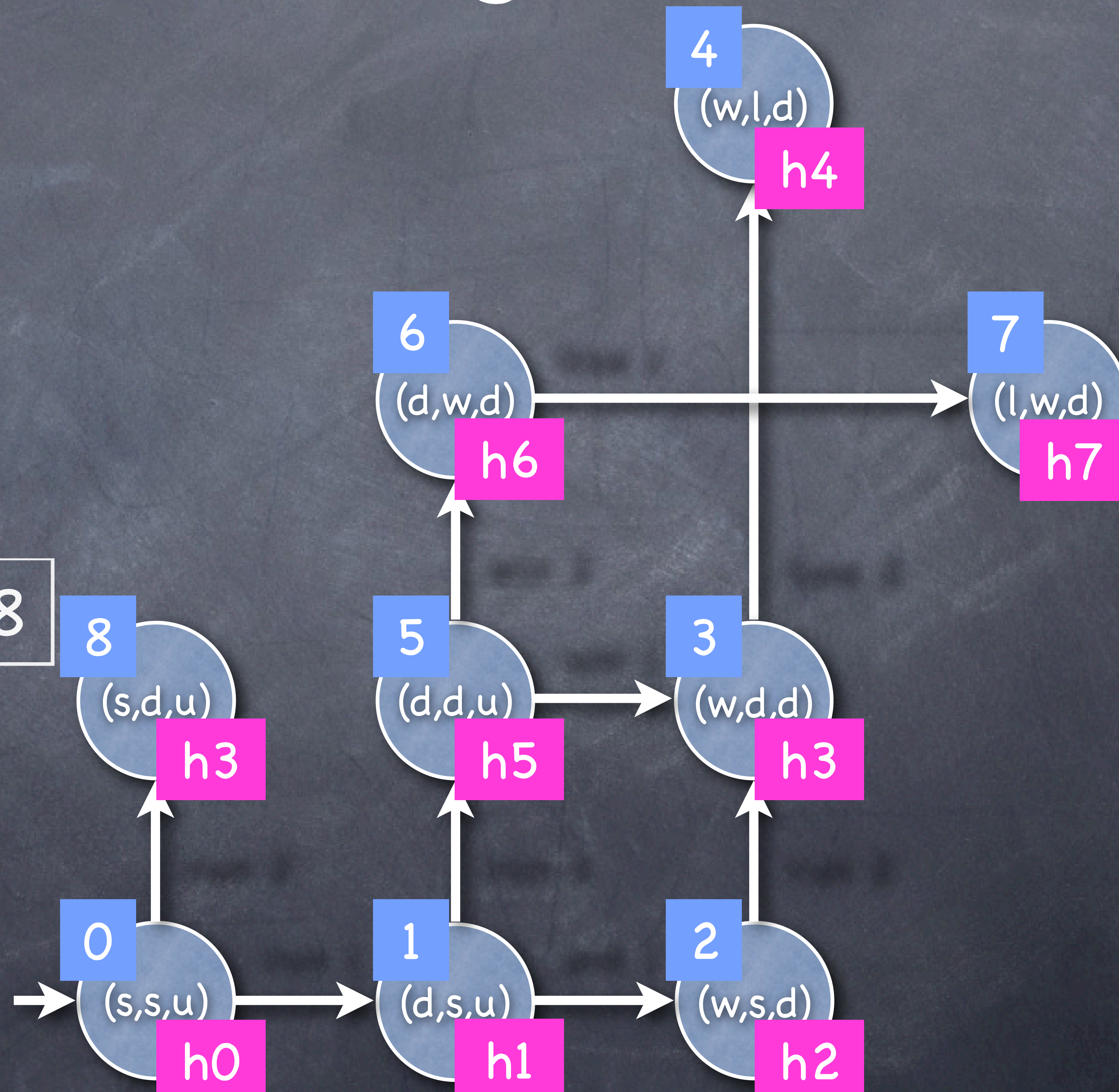
h0	0	
h1	1	
h2	2	
h3	3	8
h4	4	
h5	5	
h6	6	
h7	7	



The ComBack Algorithm

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1
6	win 2	5
7	lose 1	6
8	run 2	0

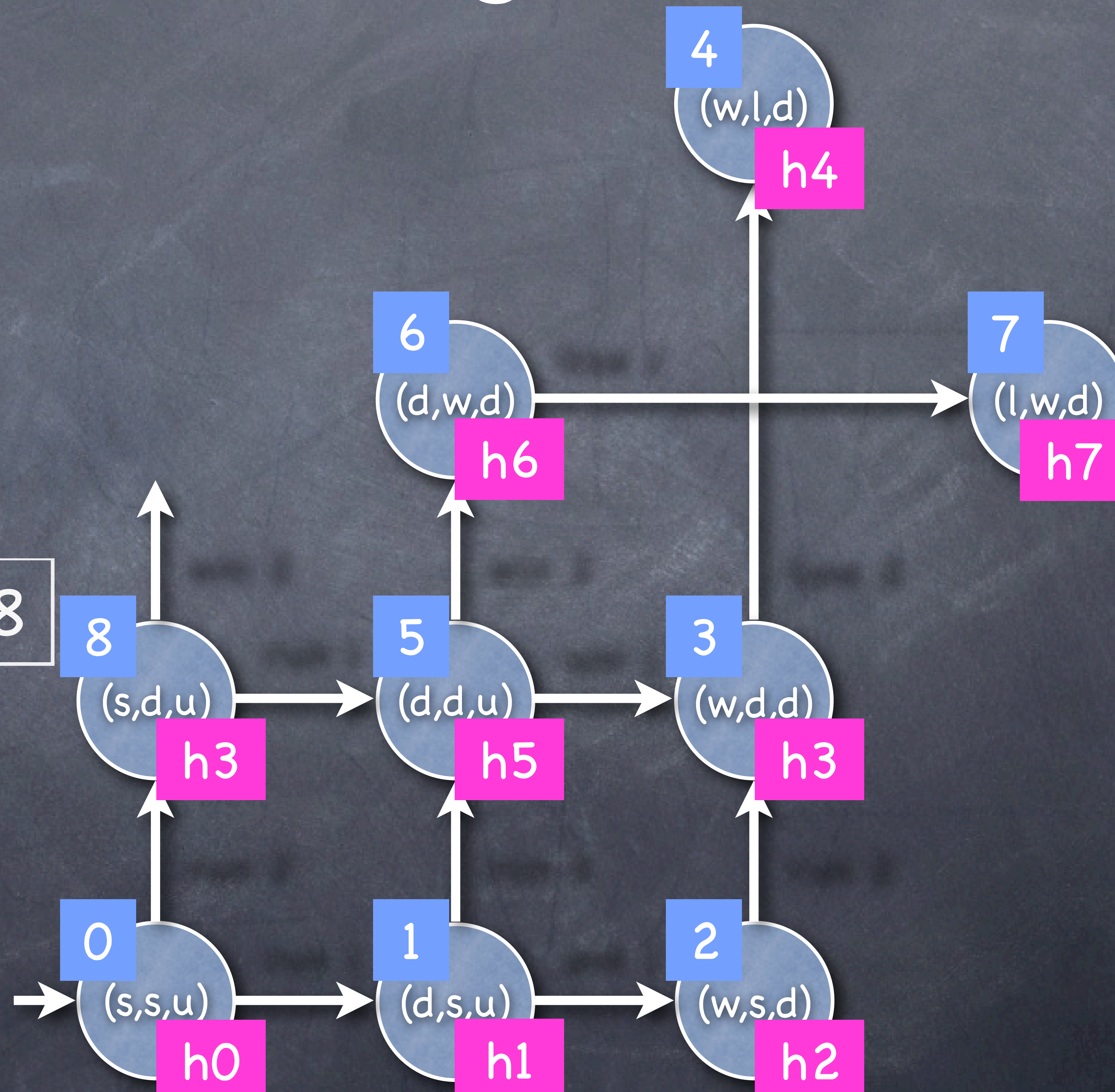
h0	0	
h1	1	
h2	2	
h3	3	8
h4	4	
h5	5	
h6	6	
h7	7	



The ComBack Algorithm

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1
6	win 2	5
7	lose 1	6
8	run 2	0

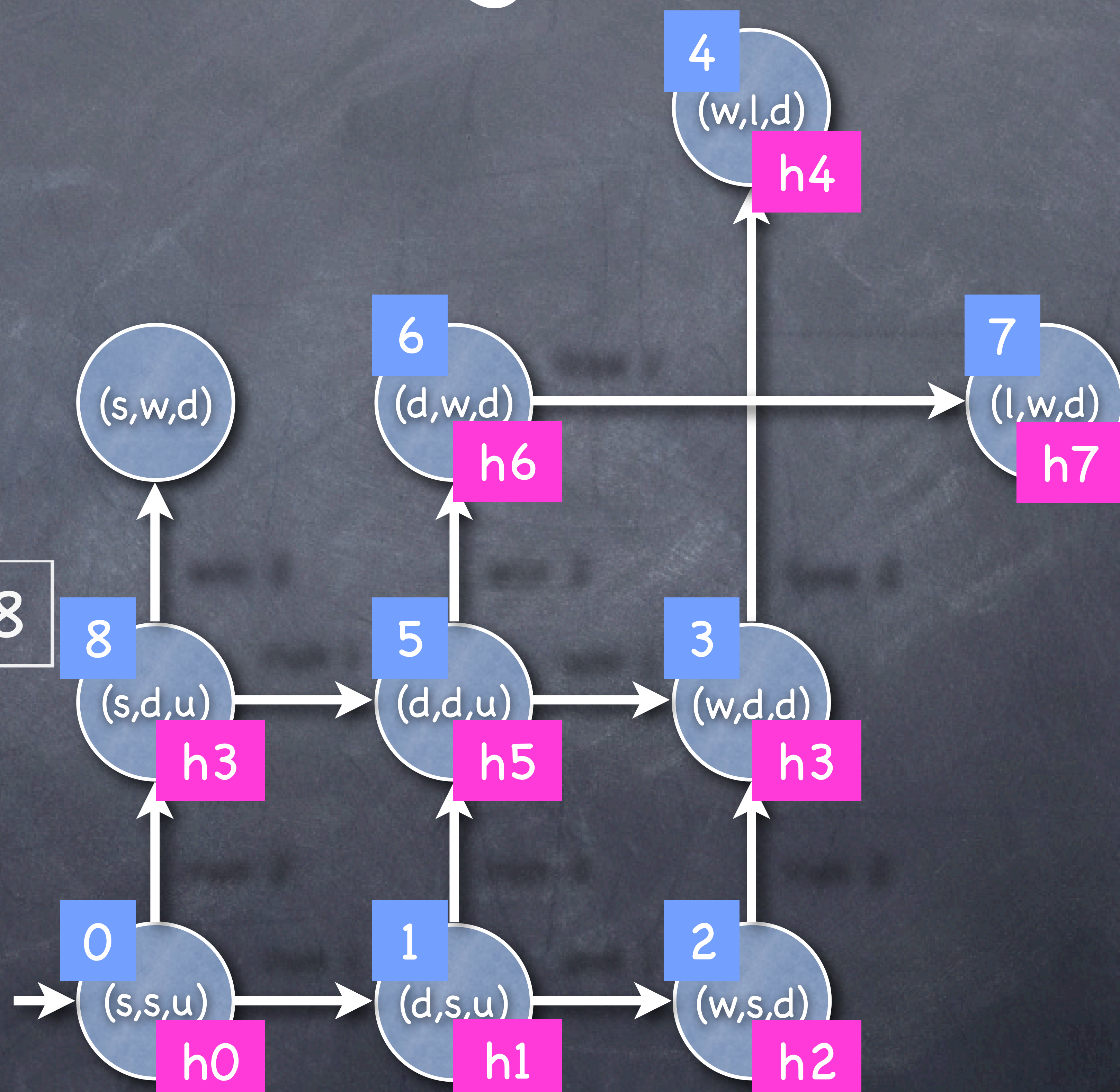
h0	0	
h1	1	
h2	2	
h3	3	8
h4	4	
h5	5	
h6	6	
h7	7	



The ComBack Algorithm

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1
6	win 2	5
7	lose 1	6
8	run 2	0

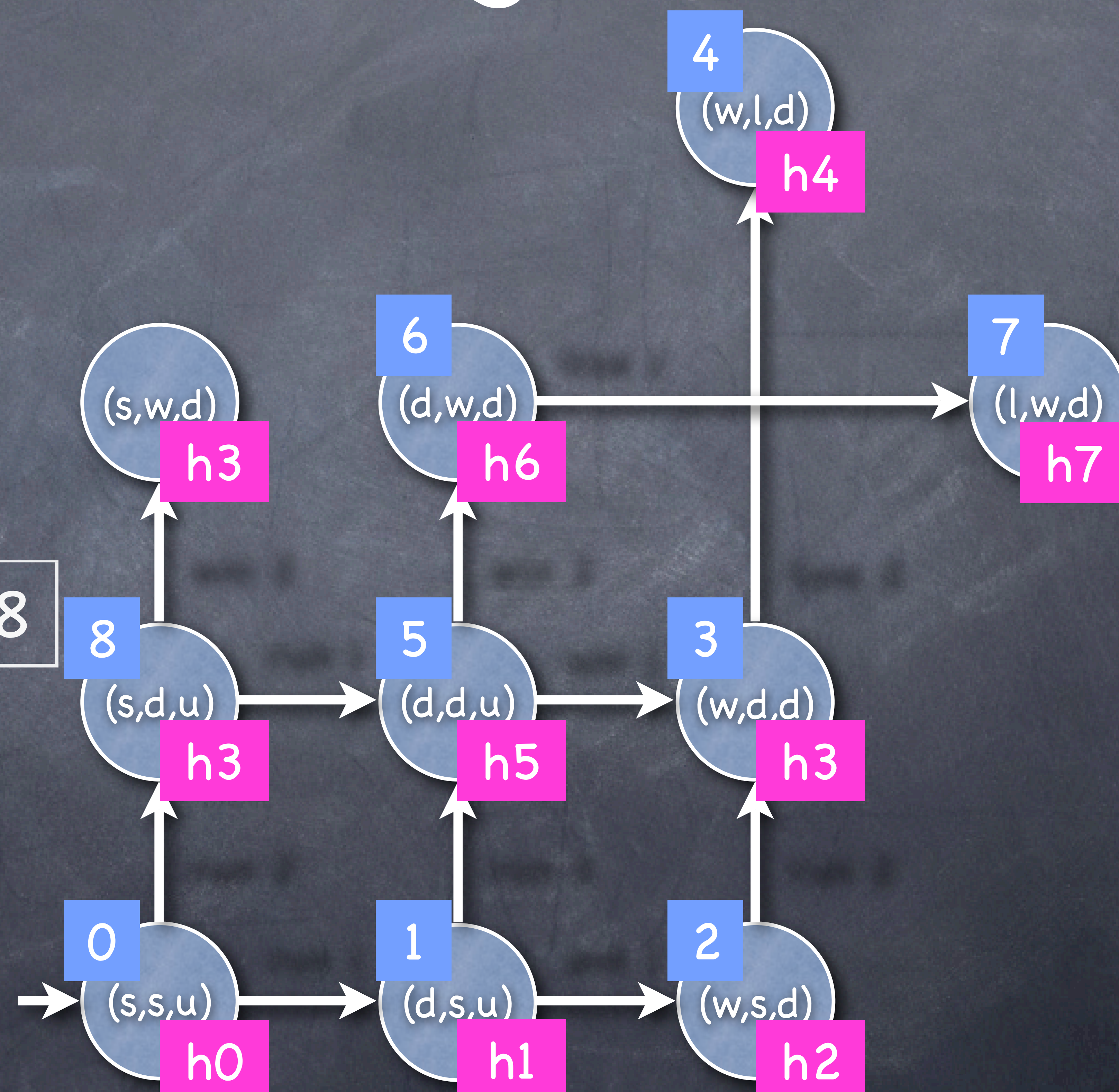
h0	0	
h1	1	
h2	2	
h3	3	8
h4	4	
h5	5	
h6	6	
h7	7	



The ComBack Algorithm

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1
6	win 2	5
7	lose 1	6
8	run 2	0

h0	0	
h1	1	
h2	2	
h3	3	8
h4	4	
h5	5	
h6	6	
h7	7	

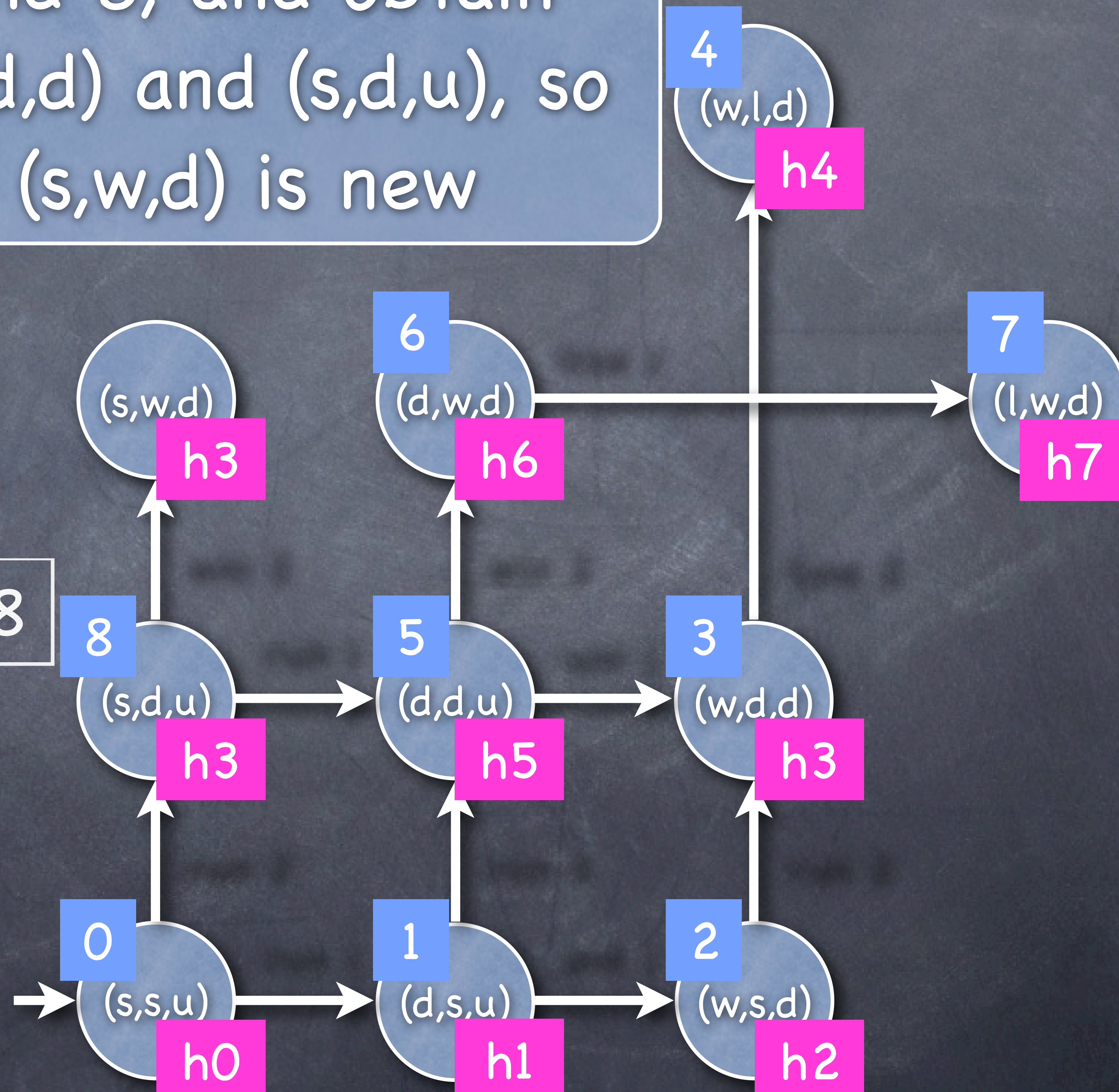


The C… ithm

We need to backtrack for state 3 and 8, and obtain (w,d,d) and (s,d,u), so (s,w,d) is new

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1
6	win 2	5
7	lose 1	6
8	run 2	0

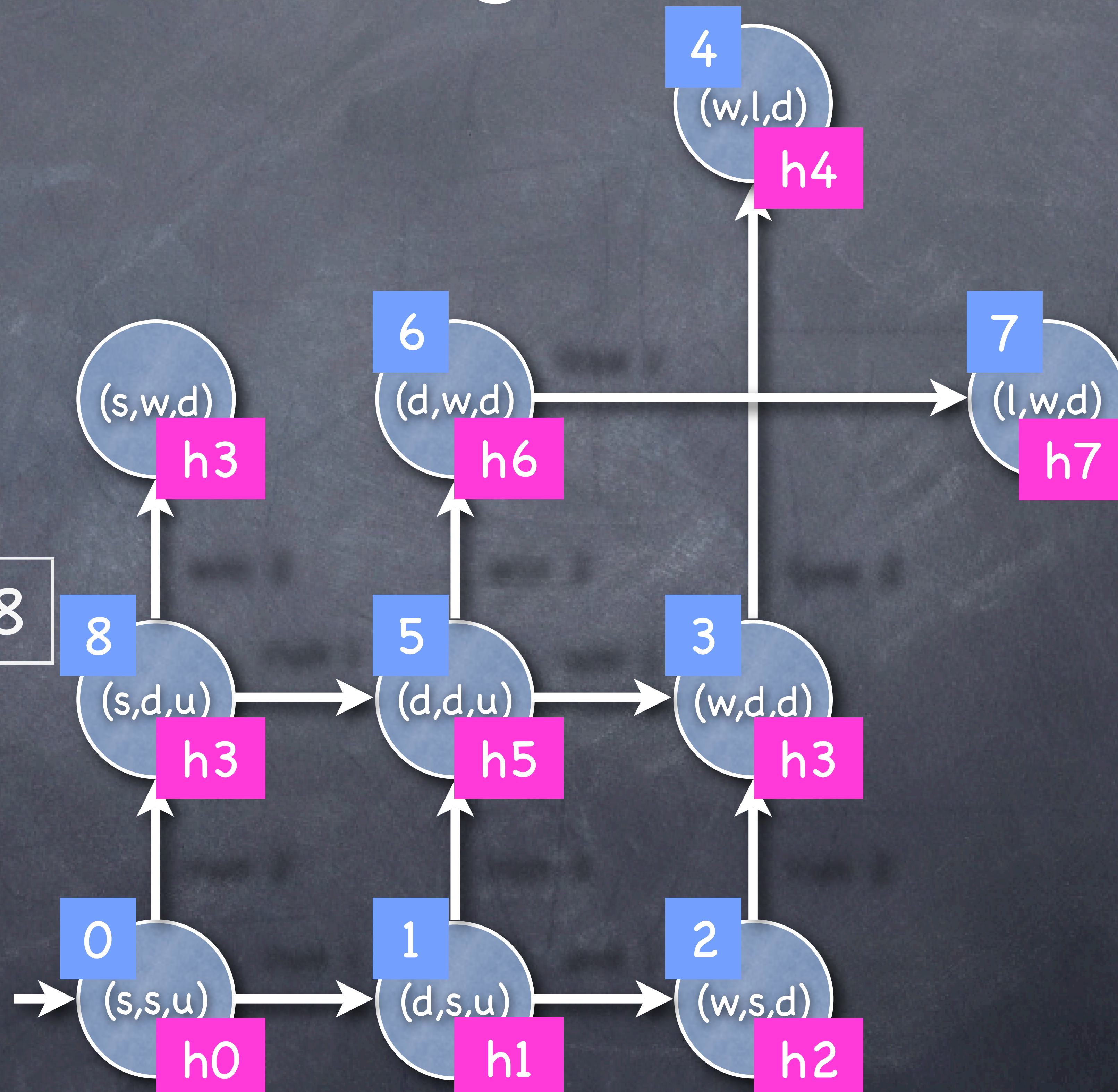
h0	0	
h1	1	
h2	2	
h3	3	8
h4	4	
h5	5	
h6	6	
h7	7	



The ComBack Algorithm

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1
6	win 2	5
7	lose 1	6
8	run 2	0

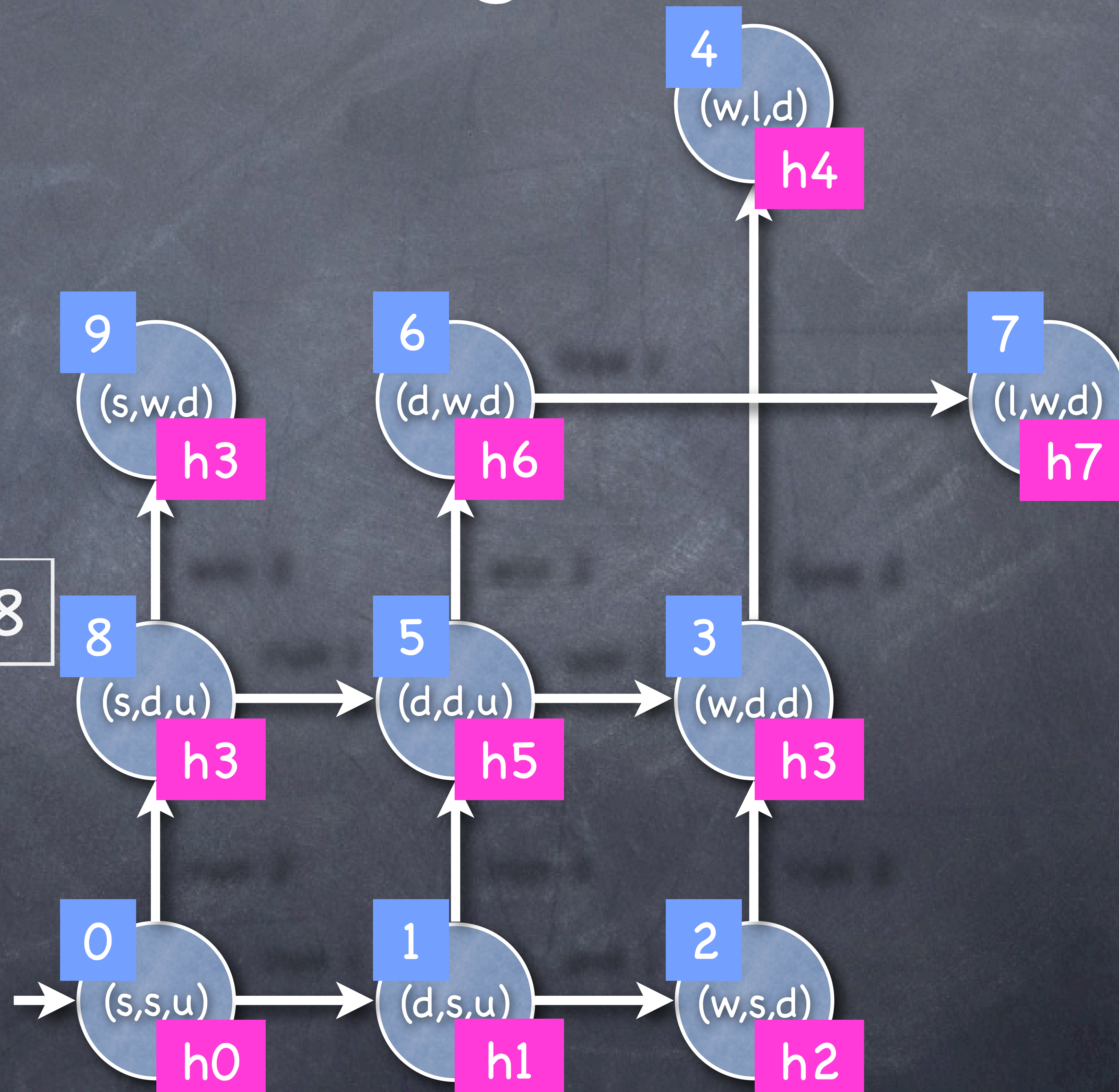
h0	0	
h1	1	
h2	2	
h3	3	8
h4	4	
h5	5	
h6	6	
h7	7	



The ComBack Algorithm

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1
6	win 2	5
7	lose 1	6
8	run 2	0

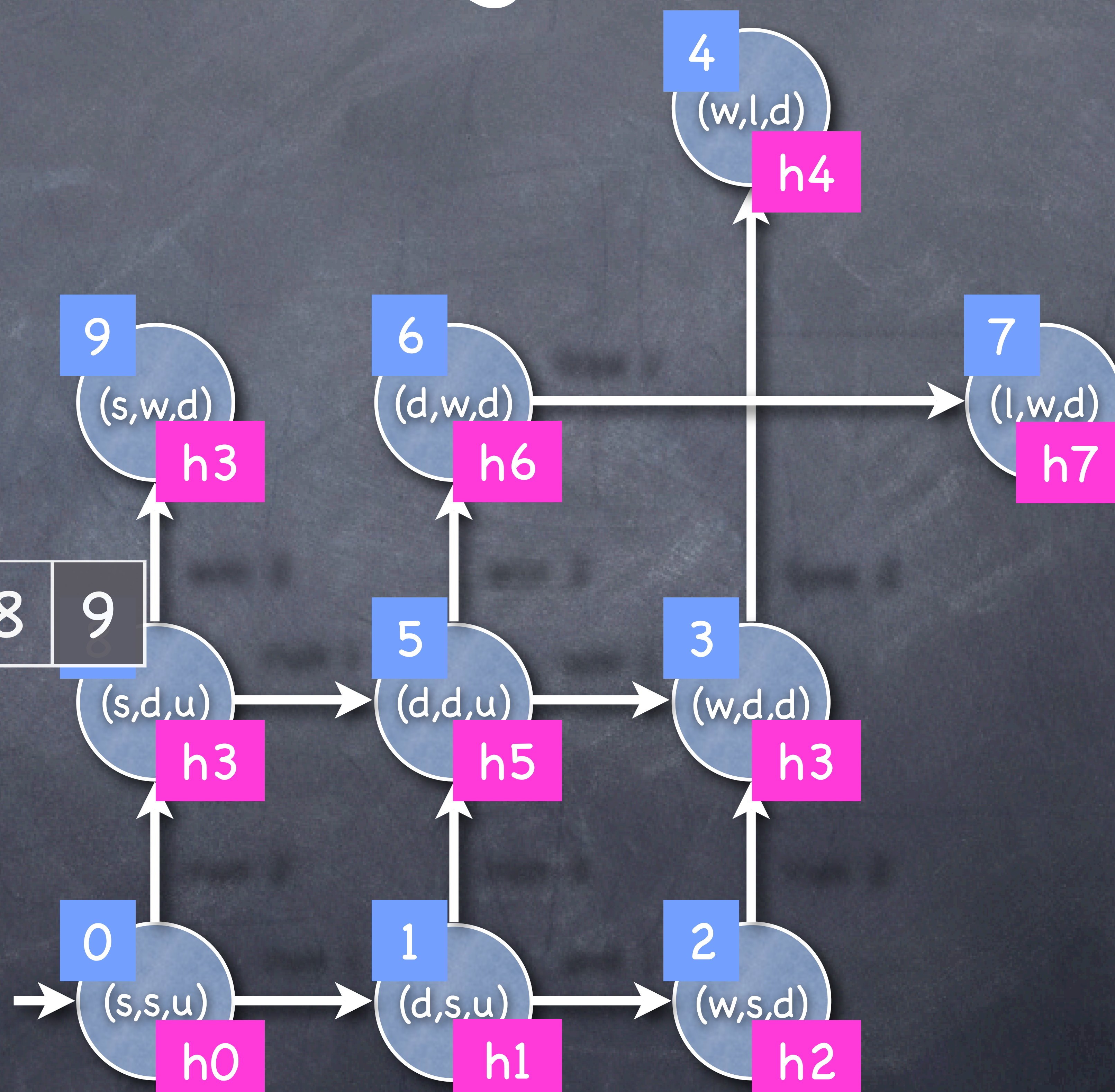
h0	0	
h1	1	
h2	2	
h3	3	8
h4	4	
h5	5	
h6	6	
h7	7	



The ComBack Algorithm

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1
6	win 2	5
7	lose 1	6
8	run 2	0

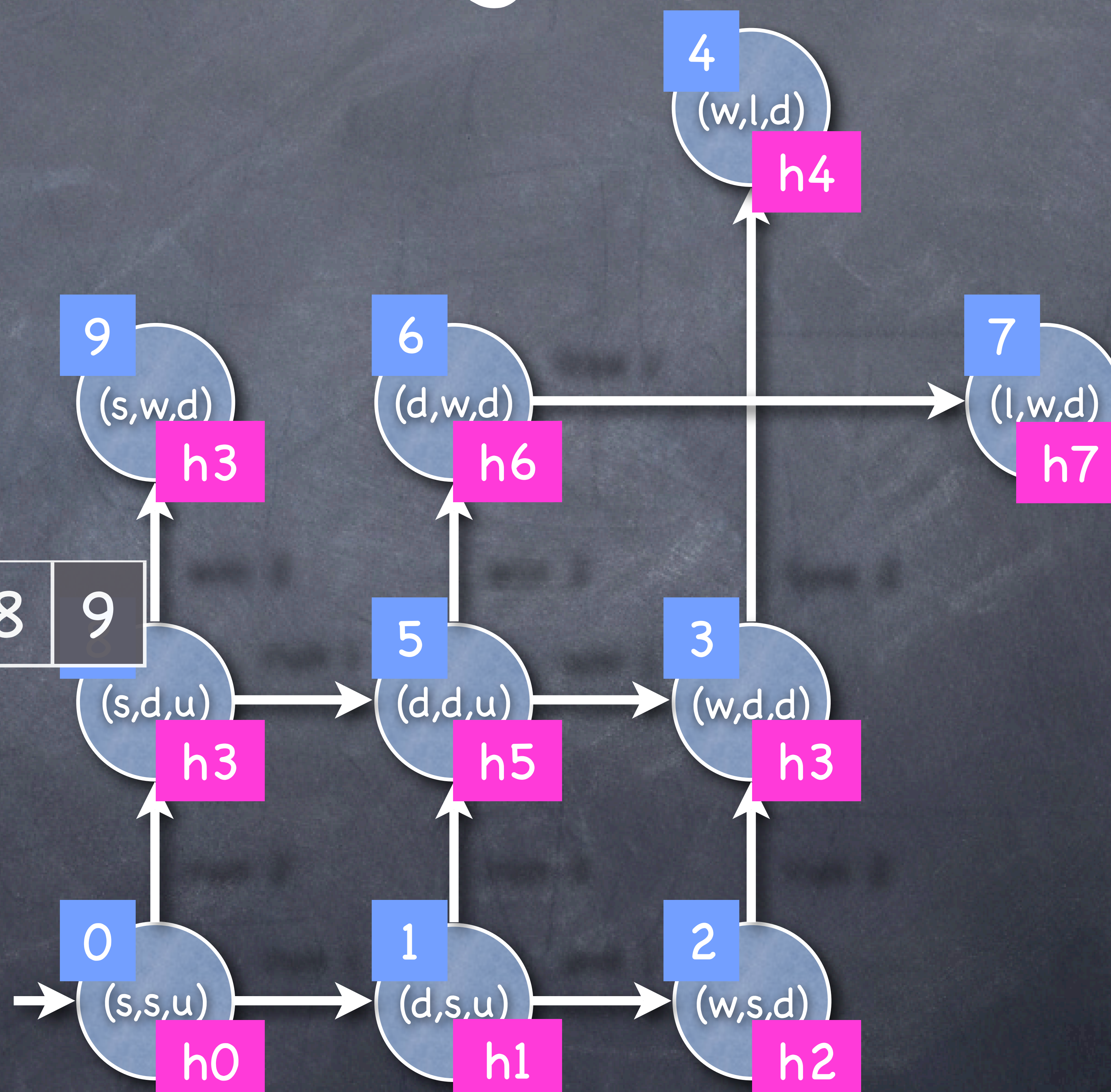
h0	0		
h1	1		
h2	2		
h3	3	8	9
h4	4		
h5	5		
h6	6		
h7	7		



The ComBack Algorithm

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1
6	win 2	5
7	lose 1	6
8	run 2	0
9		

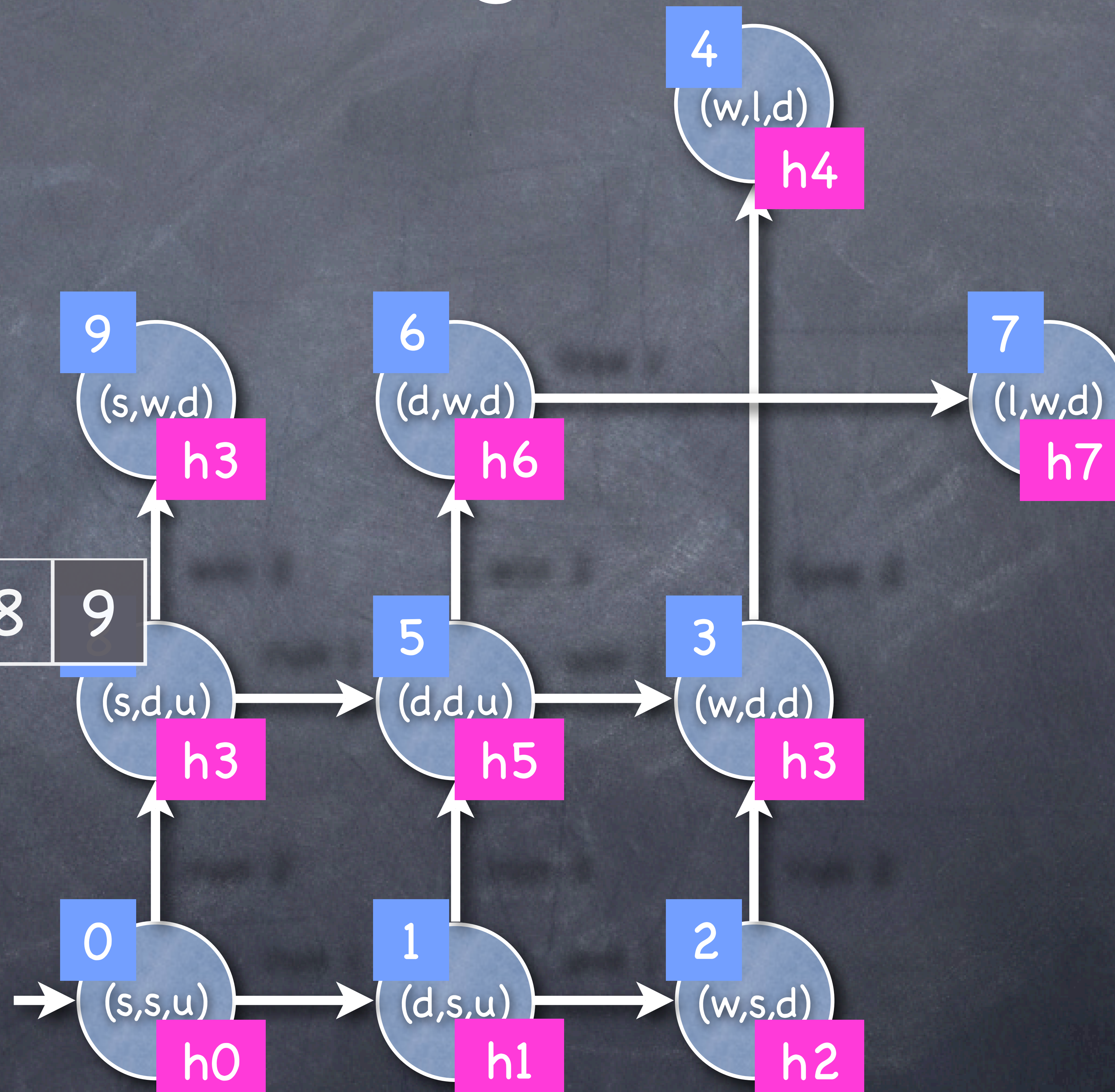
h0	0		
h1	1		
h2	2		
h3	3	8	9
h4	4		
h5	5		
h6	6		
h7	7		



The ComBack Algorithm

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1
6	win 2	5
7	lose 1	6
8	run 2	0
9	win 2	8

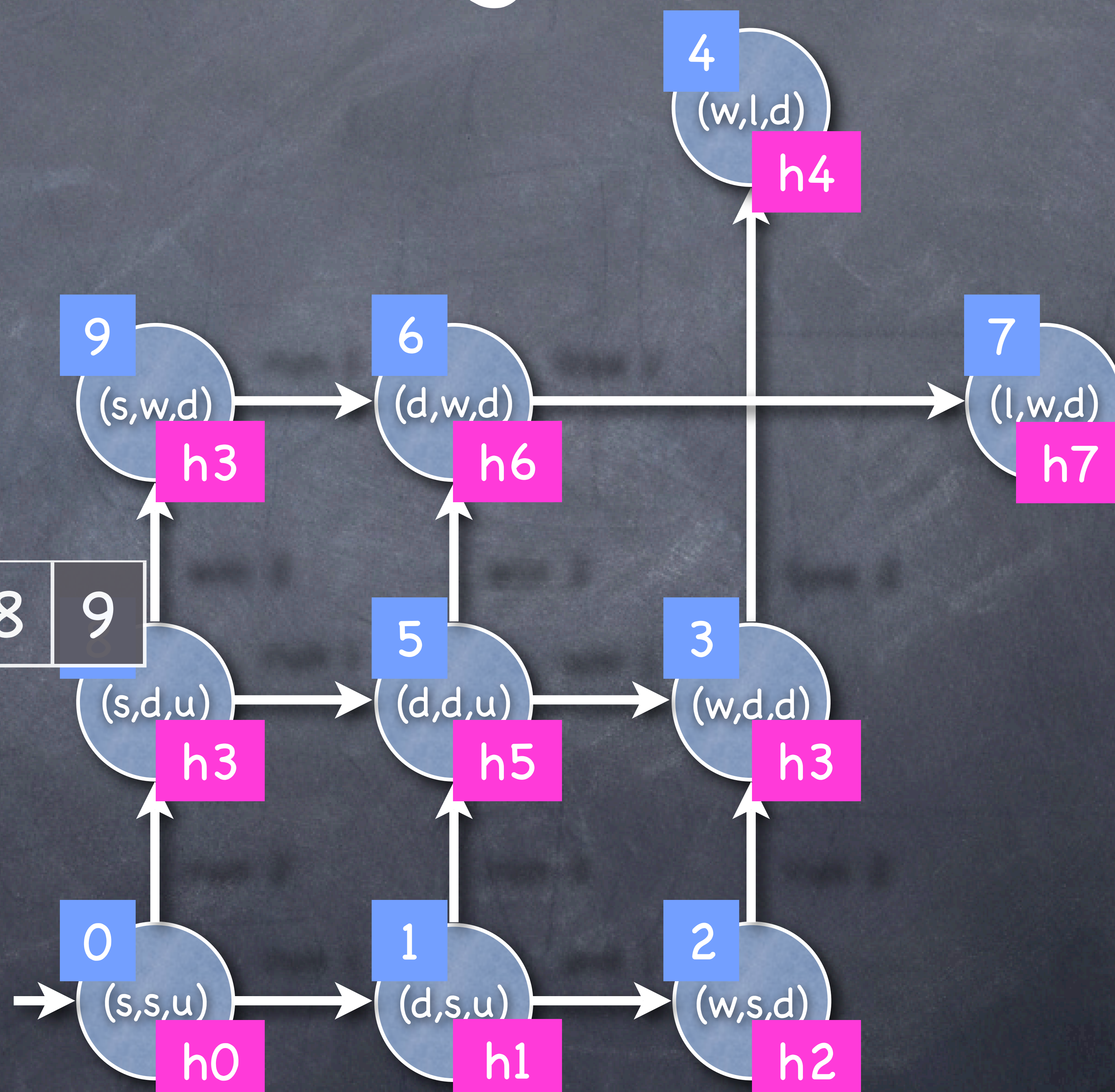
h0	0		
h1	1		
h2	2		
h3	3	8	9
h4	4		
h5	5		
h6	6		
h7	7		



The ComBack Algorithm

0		
1	run 1	0
2	win 1	1
3	run 2	2
4	lose 2	3
5	run 2	1
6	win 2	5
7	lose 1	6
8	run 2	0
9	win 2	8

h0	0		
h1	1		
h2	2		
h3	3	8	9
h4	4		
h5	5		
h6	6		
h7	7		



Evaluation of the Algorithm

- The algorithm uses $|R|\cdot(w + (2\cdot w + h)) + 2\cdot|R|\cdot(h + \log|T|) = |R|\cdot(3\cdot(w + h) + \log|T|) <= 7\cdot w\cdot|R|$
  - R: reachable states
  - w: size of machine word
  - h: size of hash-value
  - T: transitions
- Experiments show that $16\cdot w\cdot|R|$ is used
  - $\log|T| = 3\cdot w$
  - In SML-NJ everything is a reference

More Information about the Algorithm

- L. Arge, G.S. Brodal, S. Christensen, L.M. Kristensen, and M. Westergaard: The ComBack State Space Exploration Method: Combining Hash Compaction with Backtracking, not yet submitted

Conclusion

- BRITNeY animation: formalism-independent tool for making formal models look good
- State space tool: formalism-independent platform for experimenting with multiple reduction methods of state spaces