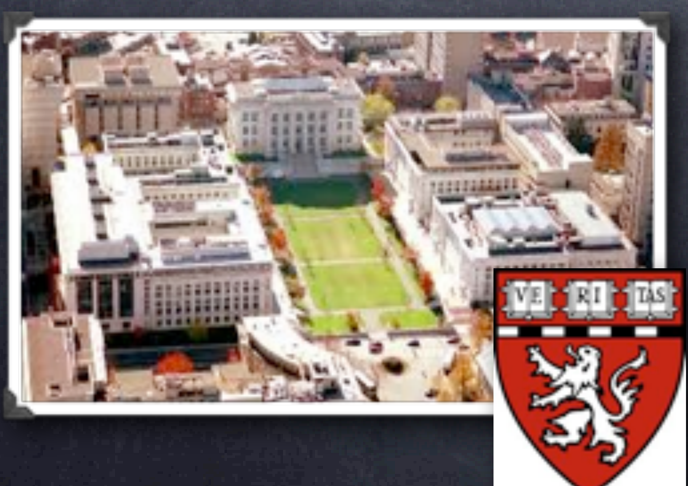
Non local causality analysis in rule-based models

Jean Krivine
CNRS & Univ. Paris Diderot

CMACS workshop NYC 2012

“Programs as models” project

- Import tools from theoretical computer science in order to describe and analyze biological systems
- Develop new technique to cope with the complexity of the cell



Theoretical computer science

- Abstract interpretation
- Concurrency theory
- Context free languages

Intrinsic Information Carriers in Combinatorial Dynamical Systems

Russ Harmer,[1, 2] Vincent Danos,[3] Jérôme Feret,[4] Jean Krivine,[2] and Walter Fontana[1]

Internal coarse-graining of molecular systems

Jérôme Feret *, Vincent Danos †, Jean Krivine *, Russ Harmer ‡, and Walter Fontana *

*Harvard Medical School, Boston, USA,†University of Edinburgh, Edinburgh, United Kingdom, and ‡CNRS & Paris Diderot, Paris, France

Submitted to Proceedings of the National Academy of Sciences of the United States of America

Rule-based modelling of cellular signalling

Vincent Danos[1,3,4], Jérôme Feret[2], Walter Fontana[3], Russell Harmer[3,4], and Jean Krivine[5]

Scalable simulation of cellular signaling networks

Vincent Danos[1,4⋆], Jérôme Feret[3], Walter Fontana[1,2], and Jean Krivine[5]

KaSim 3.0

Theoretical computer science

- Abstract interpretation
- Concurrency theory
- Context free languages

Intrinsic Information Carriers in Combinatorial Dynamical Systems

Russ Harmer,[1,2] Vincent Danos,[3] Jérôme Feret,[4] Jean Krivine,[2] and Walter Fontana[1]

Internal coarse-graining of molecular systems

Jérôme Feret *, Vincent Danos †, Jean Krivine * , Russ Harmer ‡, and Walter Fontana *

*Harvard Medical School, Boston, USA,†University of Edinburgh, Edinburgh, United Kingdom, and ‡CNRS & Paris Diderot, Paris, France

Submitted to Proceedings of the National Academy of Sciences of the United States of America

Today's talk!

Rule-based modelling of cellular signalling

Vincent Danos[1,3,4], Jérôme Feret[2], Walter Fontana[3], Russell Harmer[3,4], and Jean Krivine[5]

Scalable simulation of cellular signaling networks

Vincent Danos[1,4⋆], Jérôme Feret[3], Walter Fontana[1,2], and Jean Krivine[5]

KaSim 3.0

Causality analysis

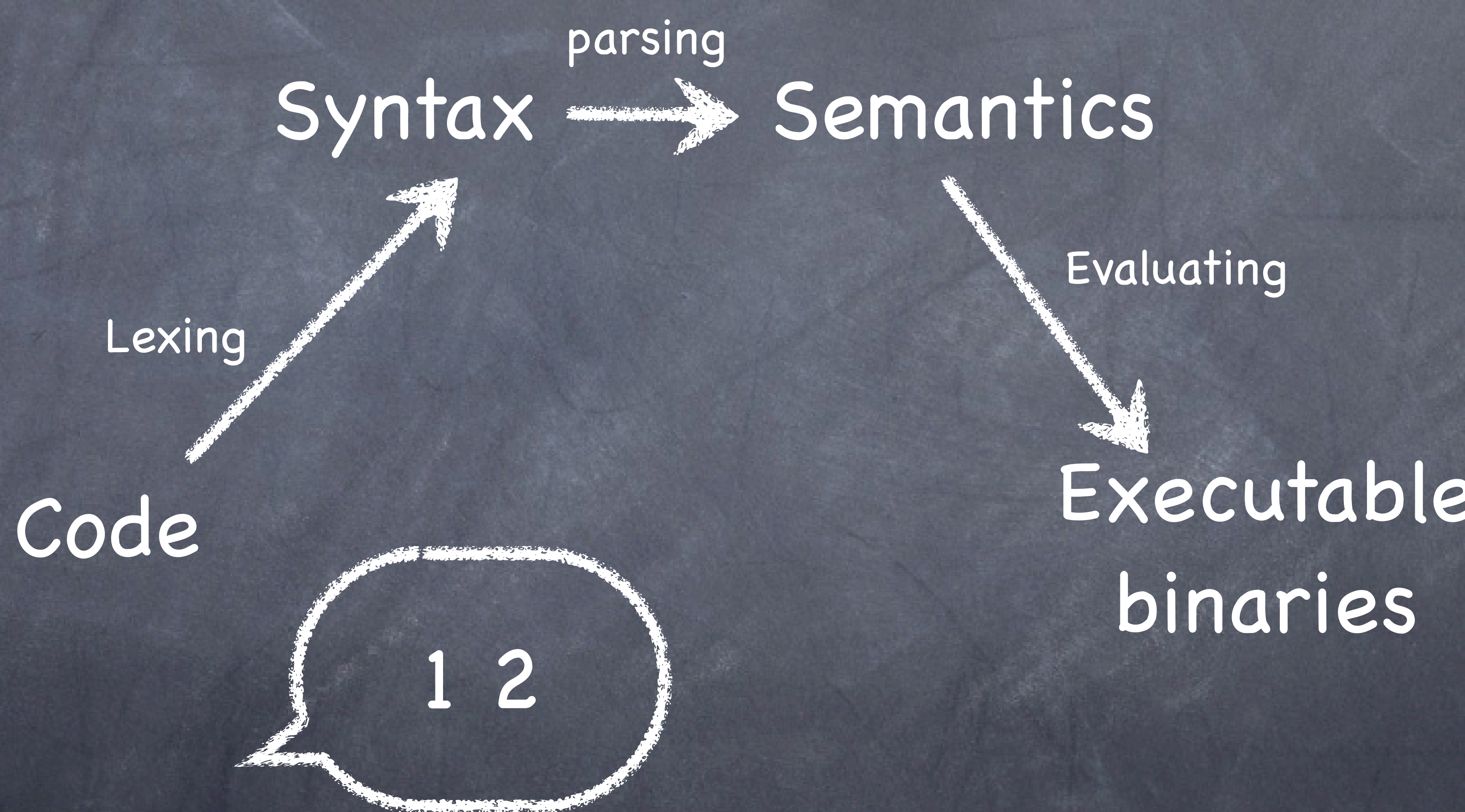
The easy life of a computer scientist

```
a := 0 ;
fork:
  if child then {a:=a+1 ; print(a)}
  else {a:=a+1 ; print(a)}
```

The easy life of a computer scientist

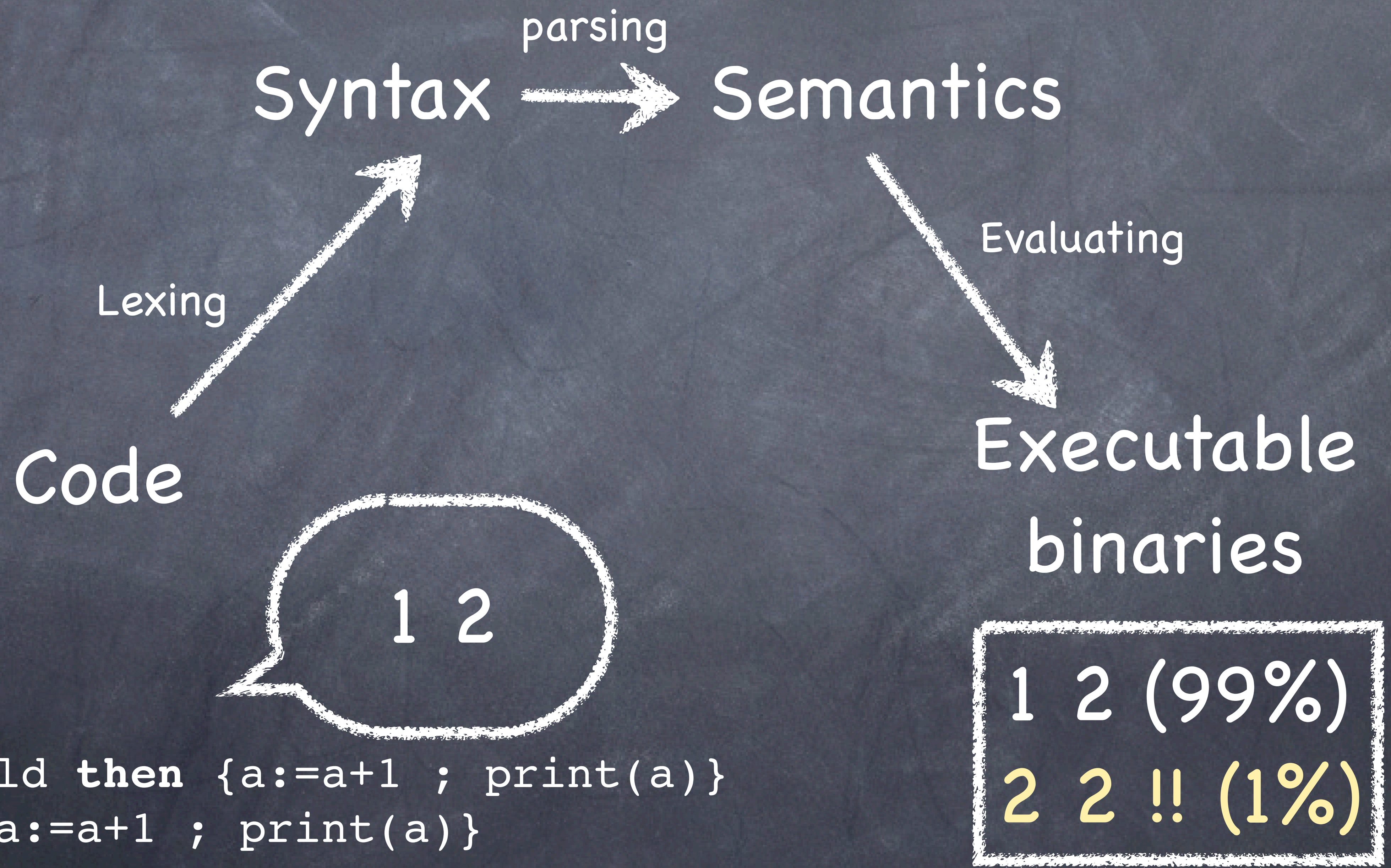

```
a := 0 ;
fork:
  if child then {a:=a+1 ; print(a)}
  else {a:=a+1 ; print(a)}
```

The easy life of a computer scientist




```
a := 0 ;
fork:
  if child then {a:=a+1 ; print(a)}
  else {a:=a+1 ; print(a)}
```

The easy life of a computer scientist




```
a := 0 ;
fork:
  if child then {a:=a+1 ; print(a)}
  else {a:=a+1 ; print(a)}
```

Debugging

```
T0: set a:=0

T0: spawn T1

T0: set a:=a+1

T1: set a:=a+1

T1: print (a)

T1: terminate

T0: print (a)

observation: 2 2
```

Interleaving semantics

Debugging

```
T0: set a:=0

T0: spawn T1

T0: set a:=a+1

T1: set a:=a+1

T1: print (a)

T1: terminate

T0: print (a)

observation: 2 2
```

Interleaving semantics   Non-Interleaving semantics

Debugging

T0: set a:=0

T0: spawn T1

T0: set a:=a+1

T1: set a:=a+1

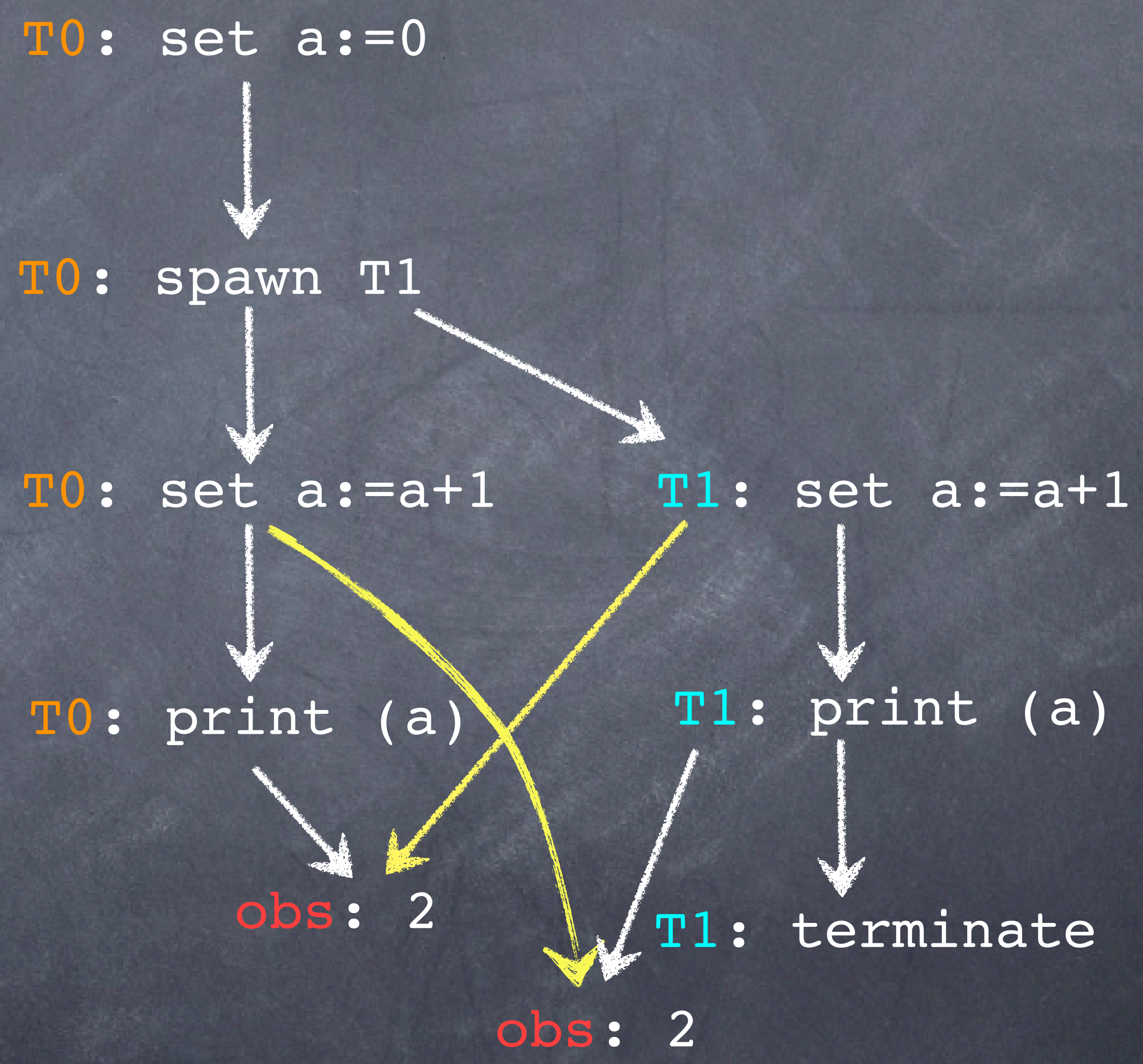
T1: print (a)

T1: terminate

T0: print (a)

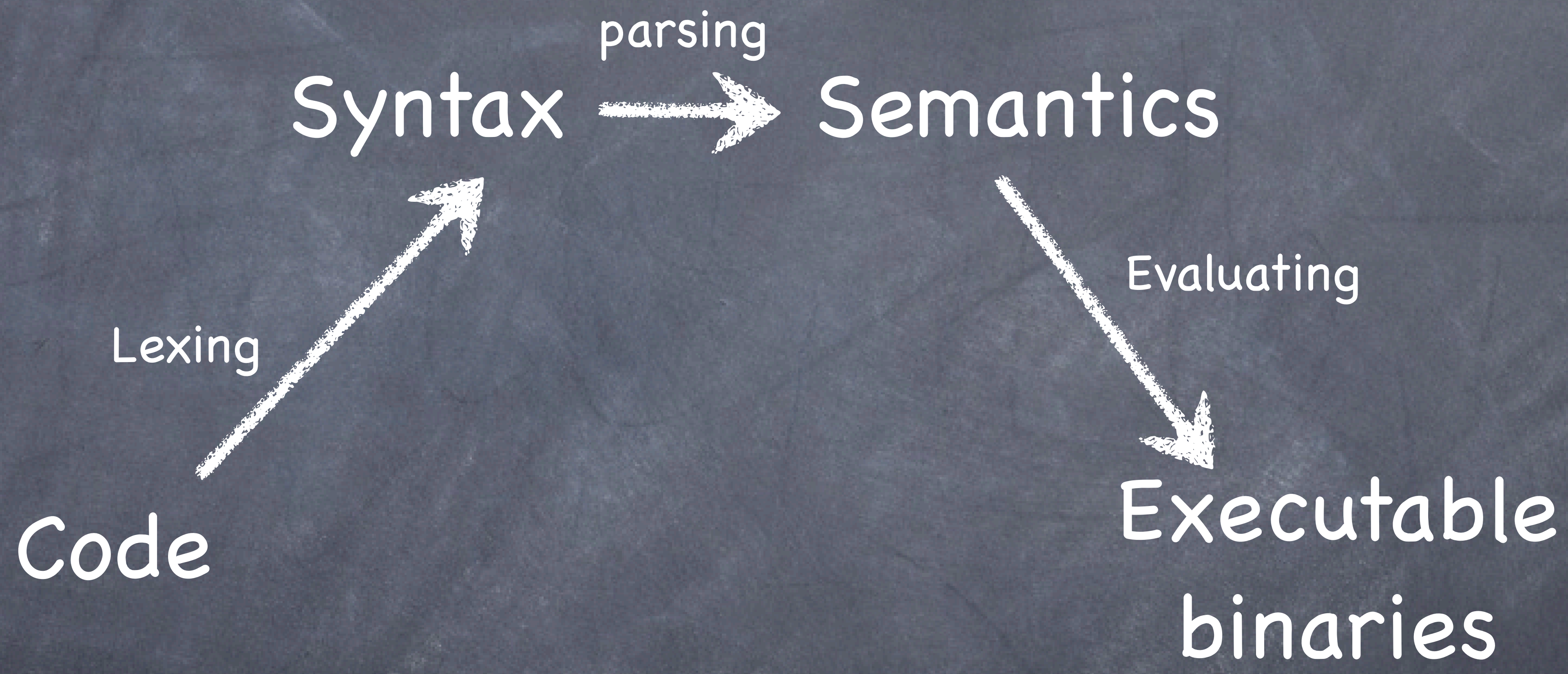
observation: 2 2

Interleaving semantics

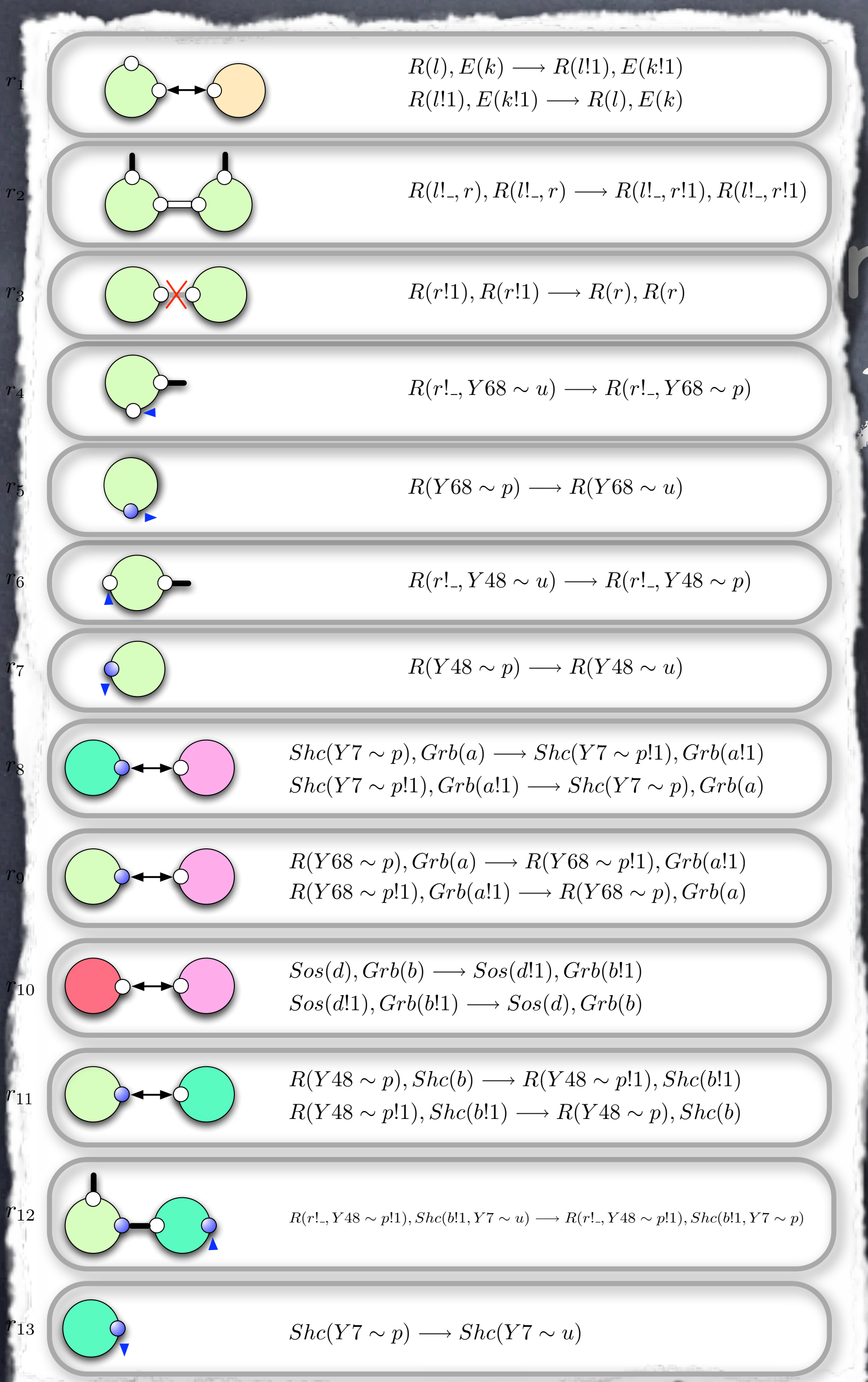


Non-Interleaving semantics

The difficult life of a modeler



The difficult life of a modeler

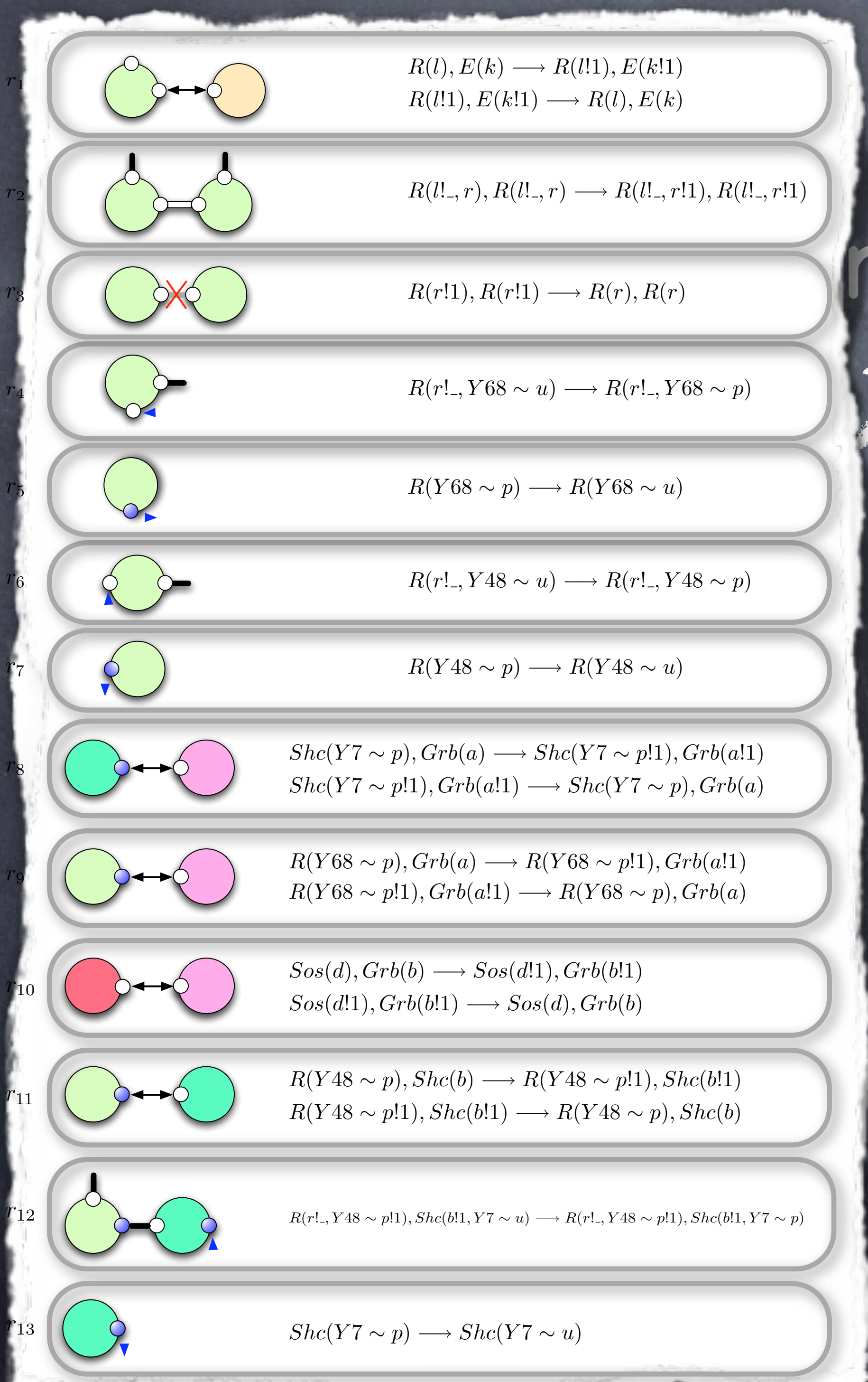


ntax —parsing→ Semantics

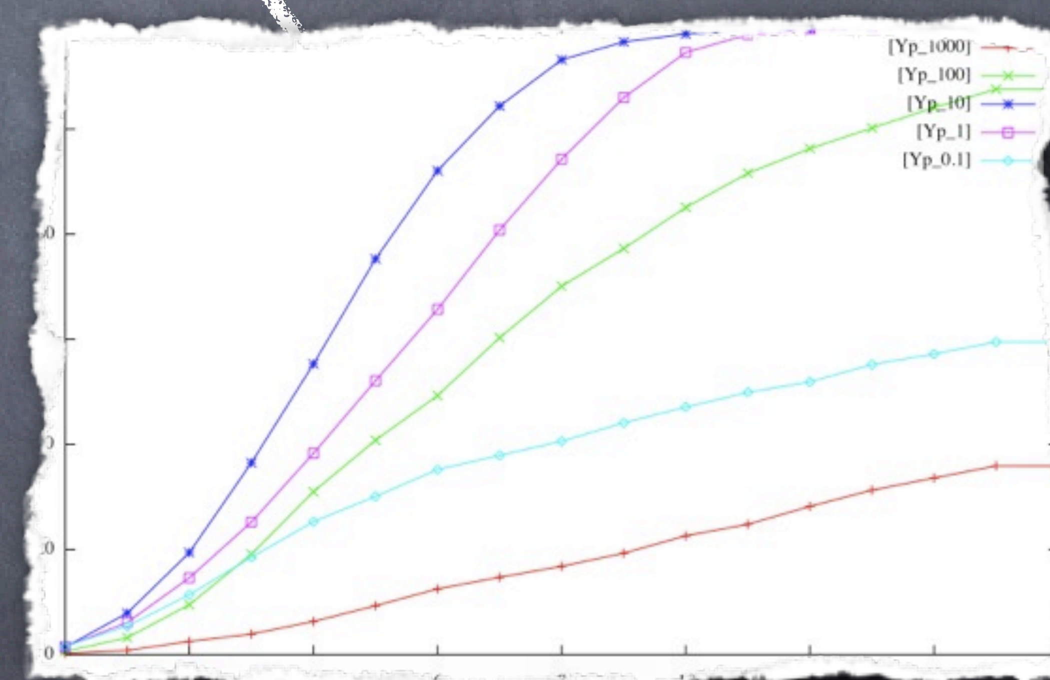
Evaluating

Executable binaries

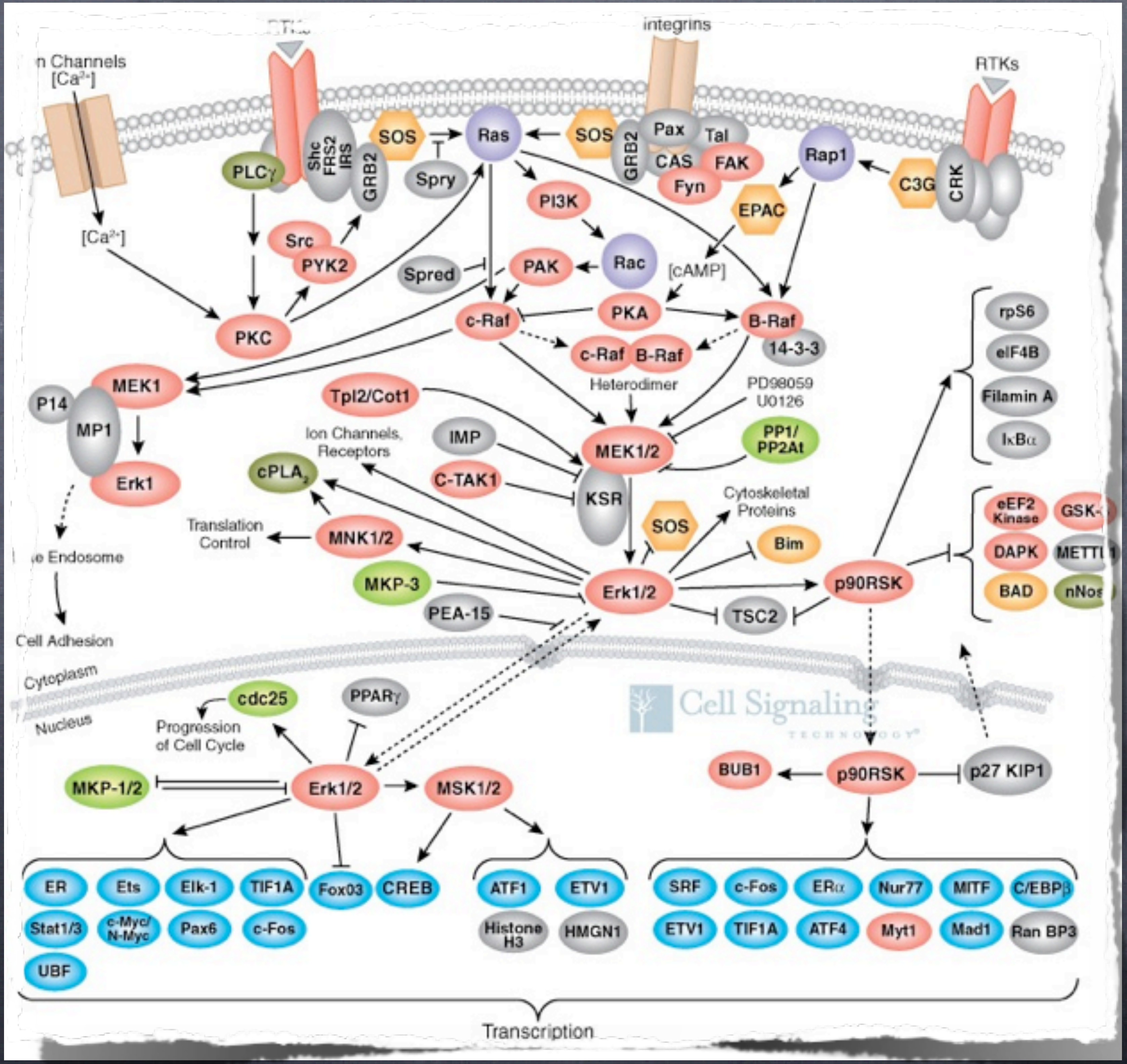
The difficult life of a modeler



ntax —parsing→ Semantics



… and causality

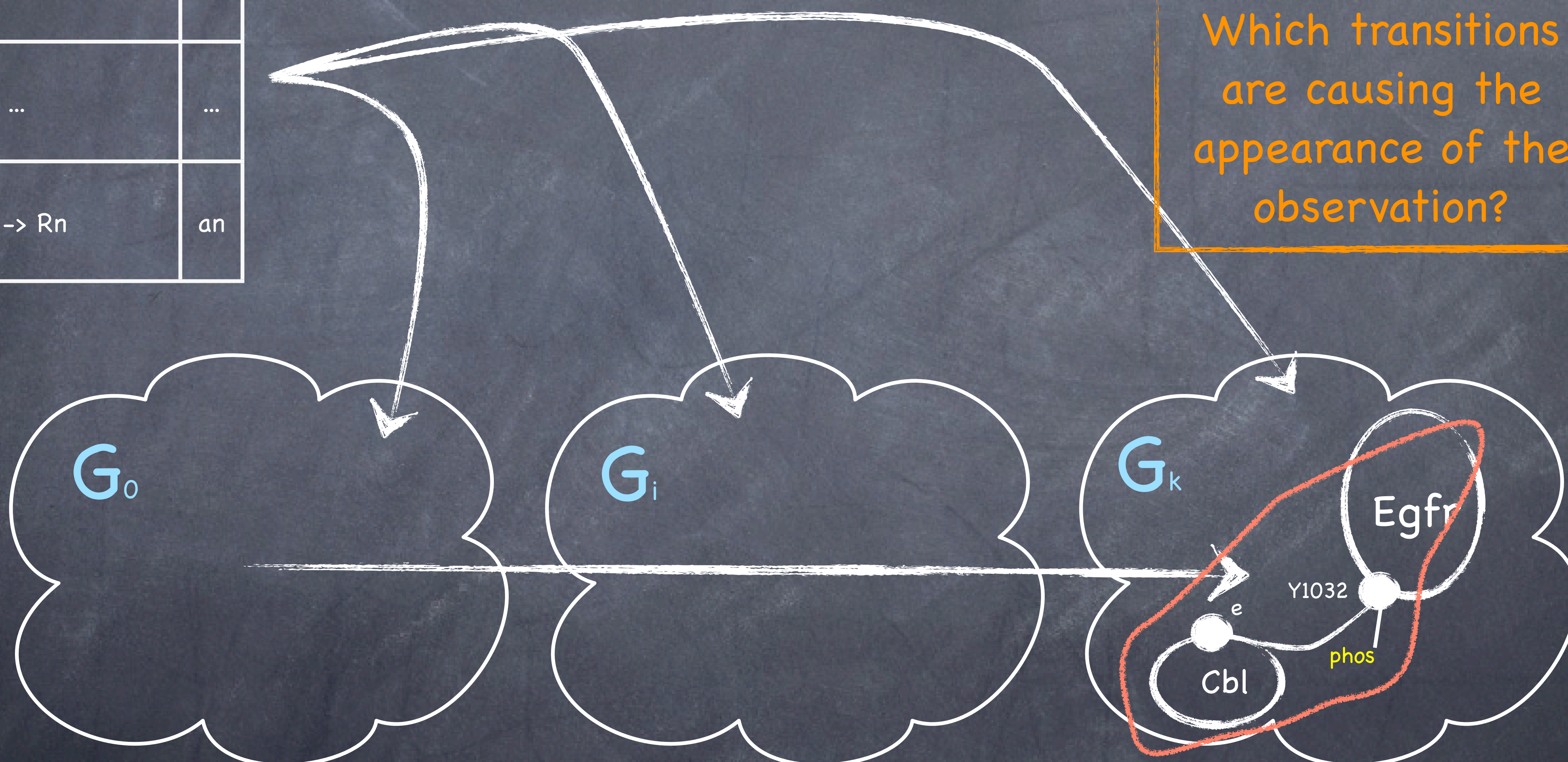


The question

The question

L1 -> R1	a1
...	...
Ln -> Rn	an

Which transitions are causing the appearance of the observation?



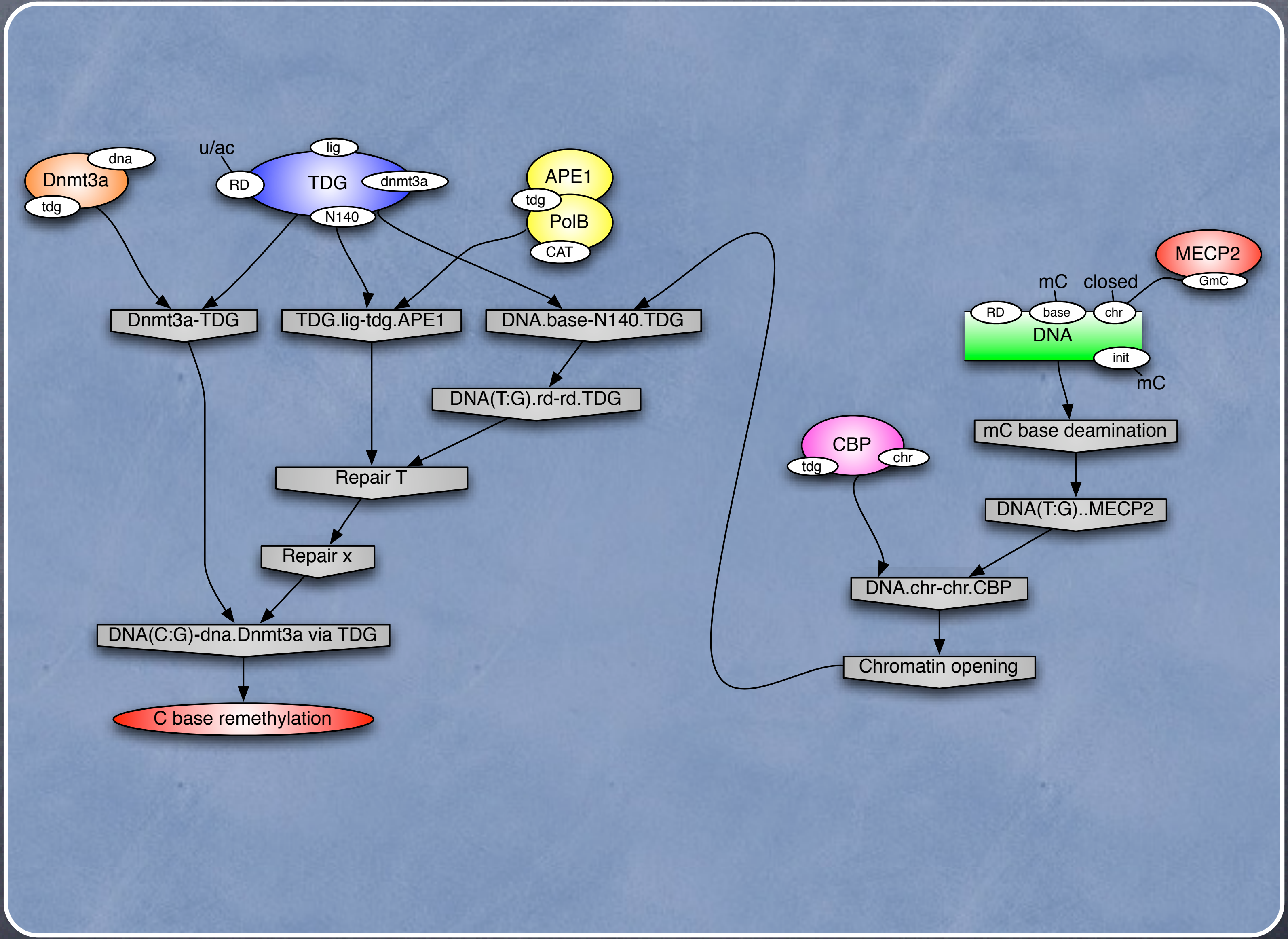
initial graph

observation

Deduce a pathway!

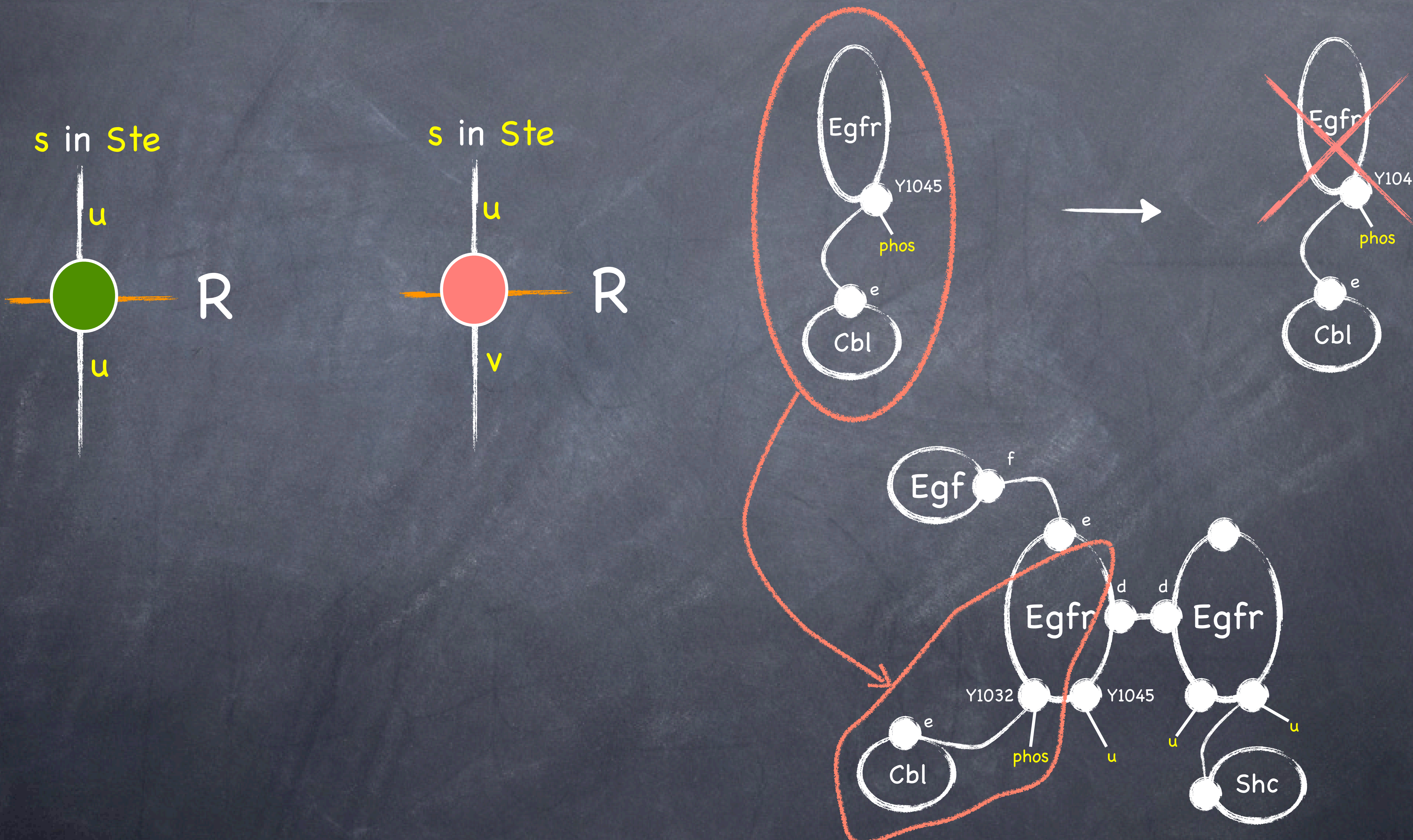
Remaining of the talk

- Classical causality analysis
- Notion of "knock-out" property
- Causal compression

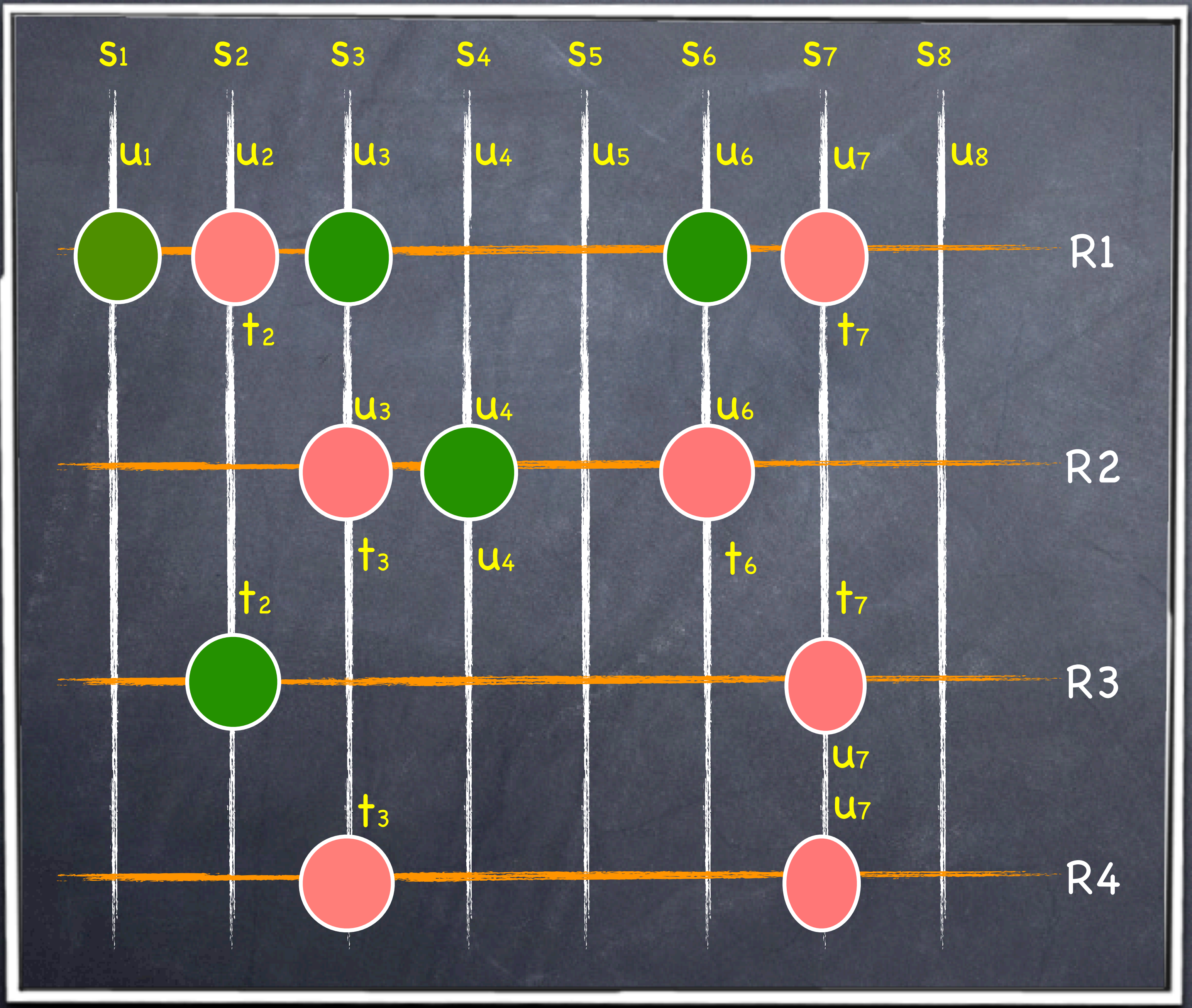


Classical causality analysis

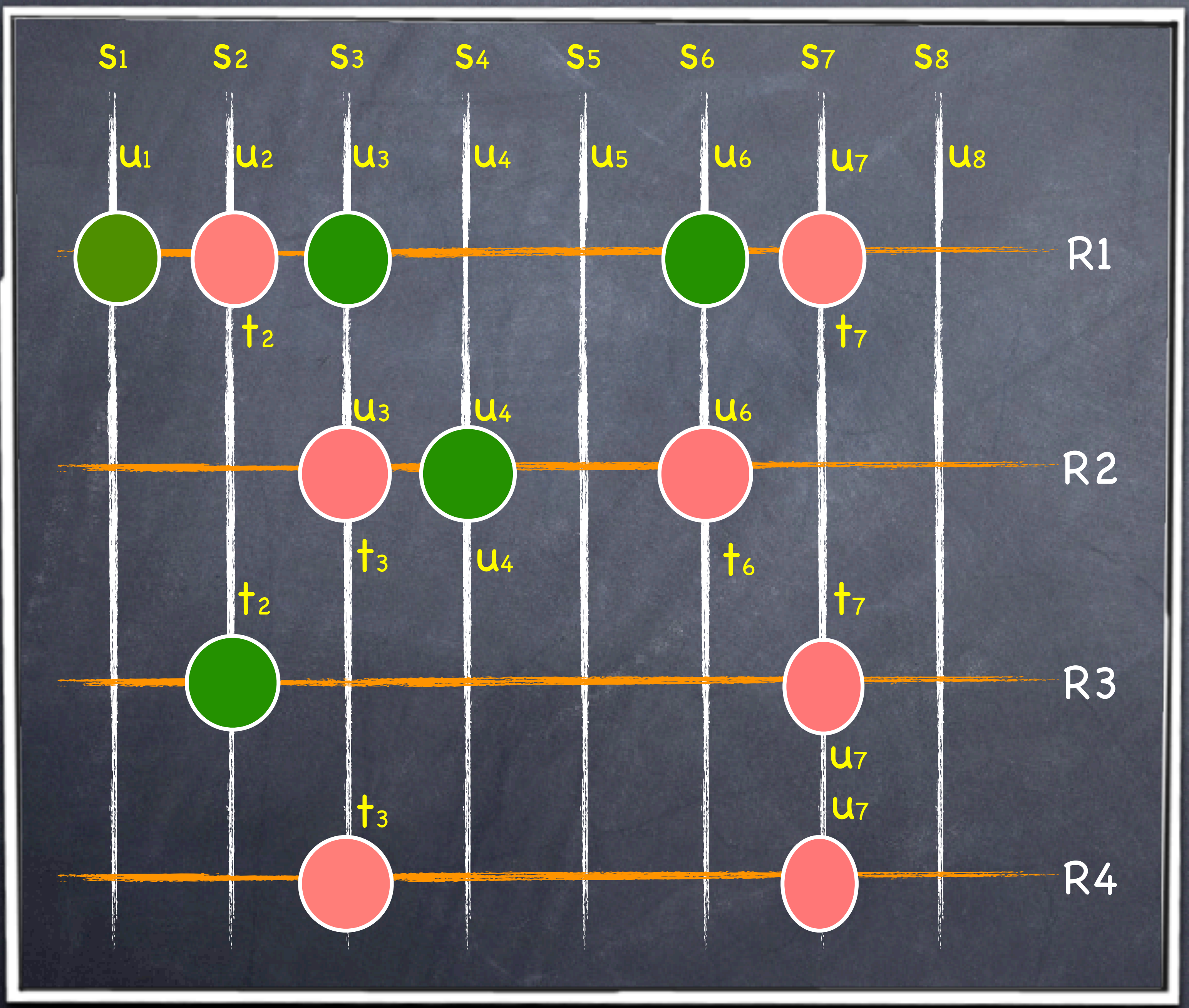
Tracking modifications

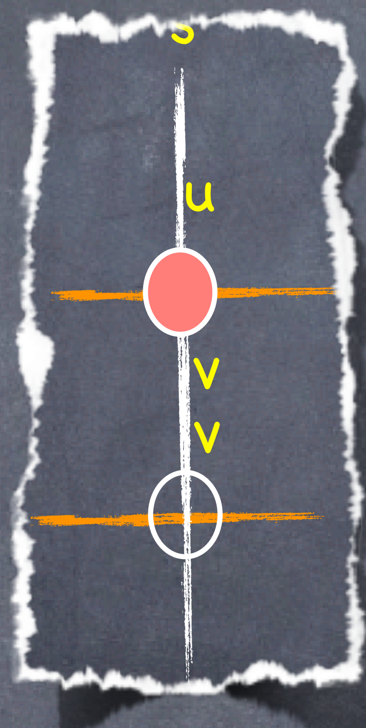


Dependencies



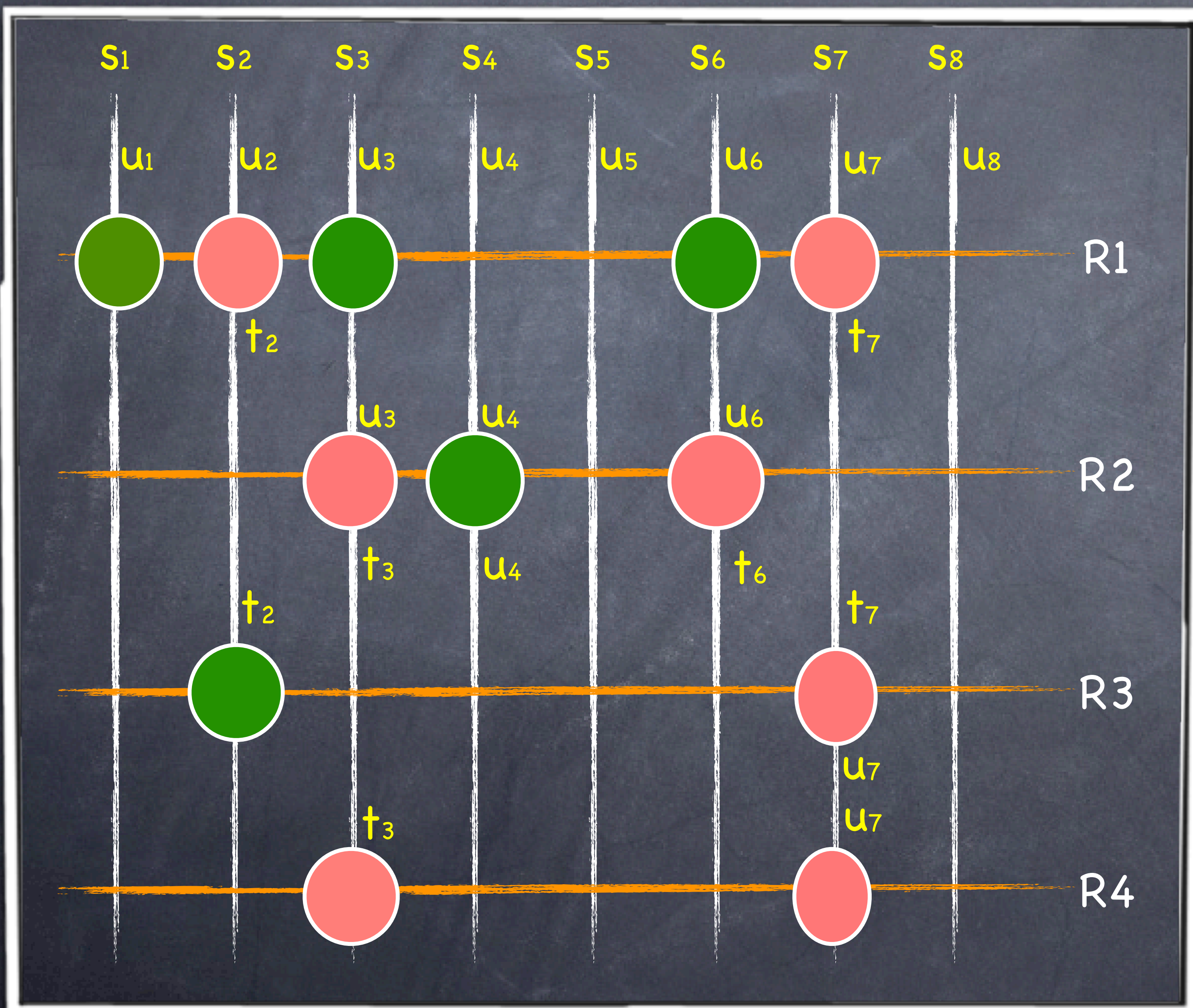
Dependencies

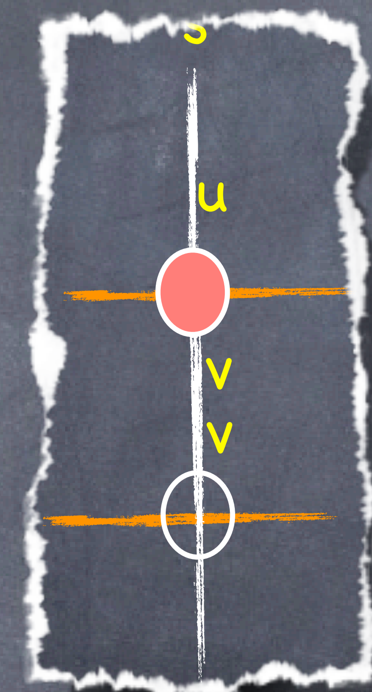




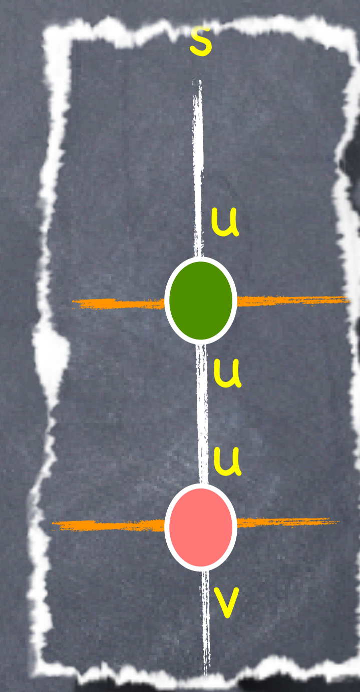
Causality

Dependencies



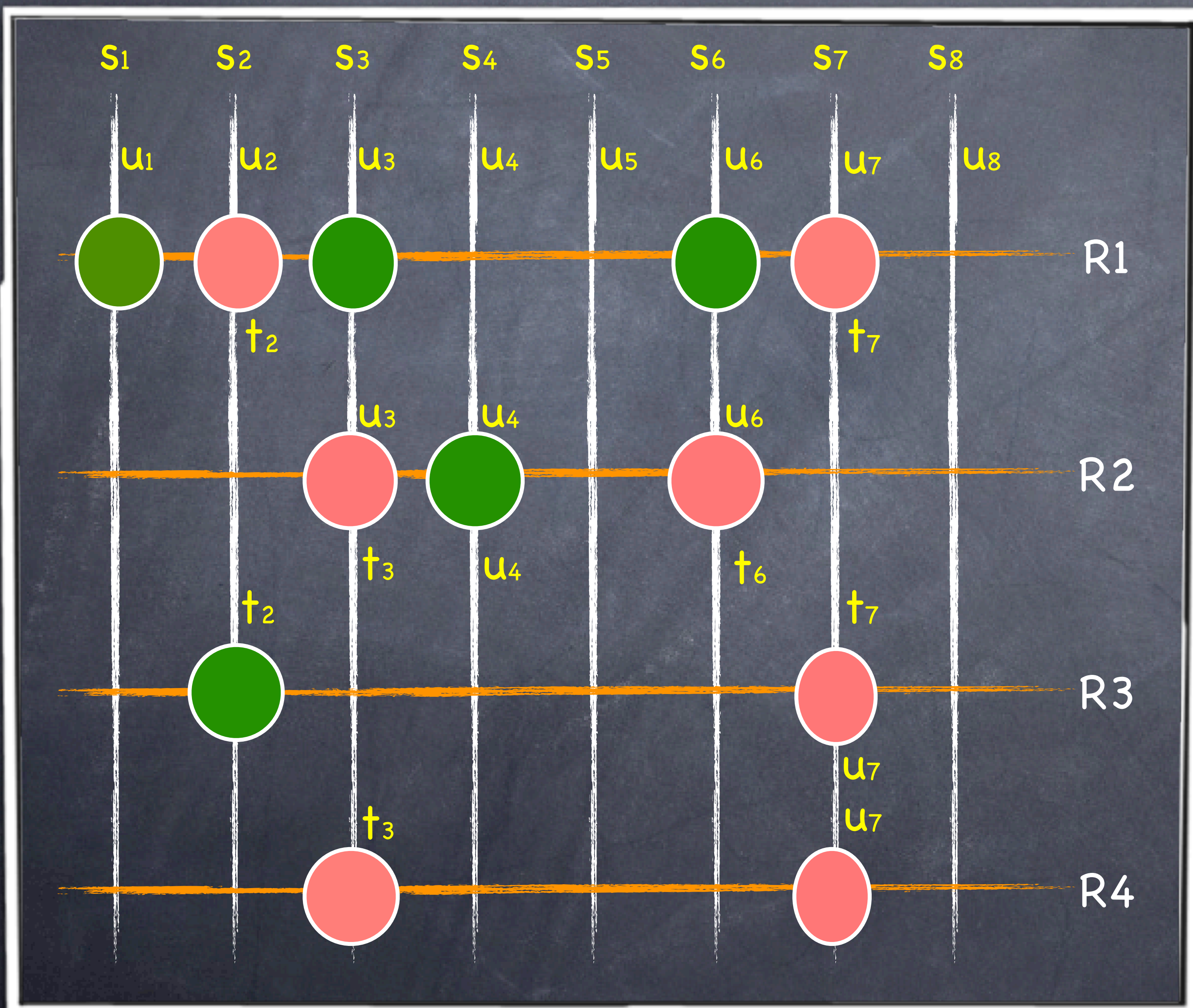


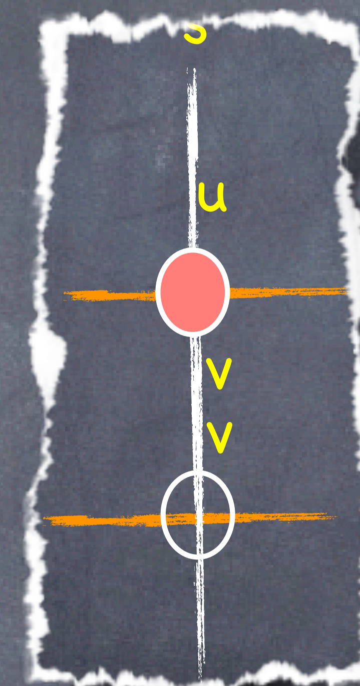
Causality



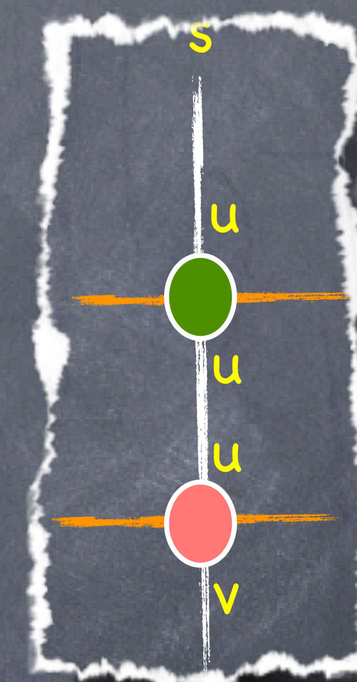
Precedence

Dependencies

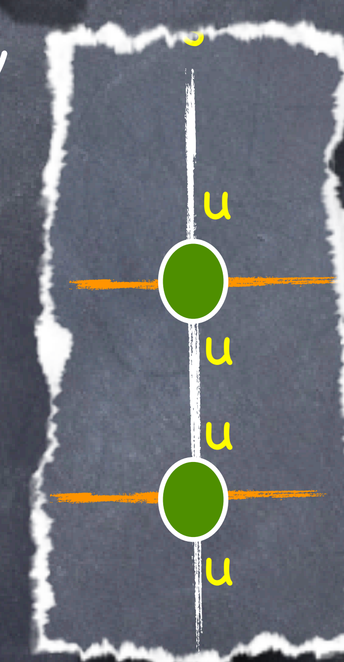




Causality

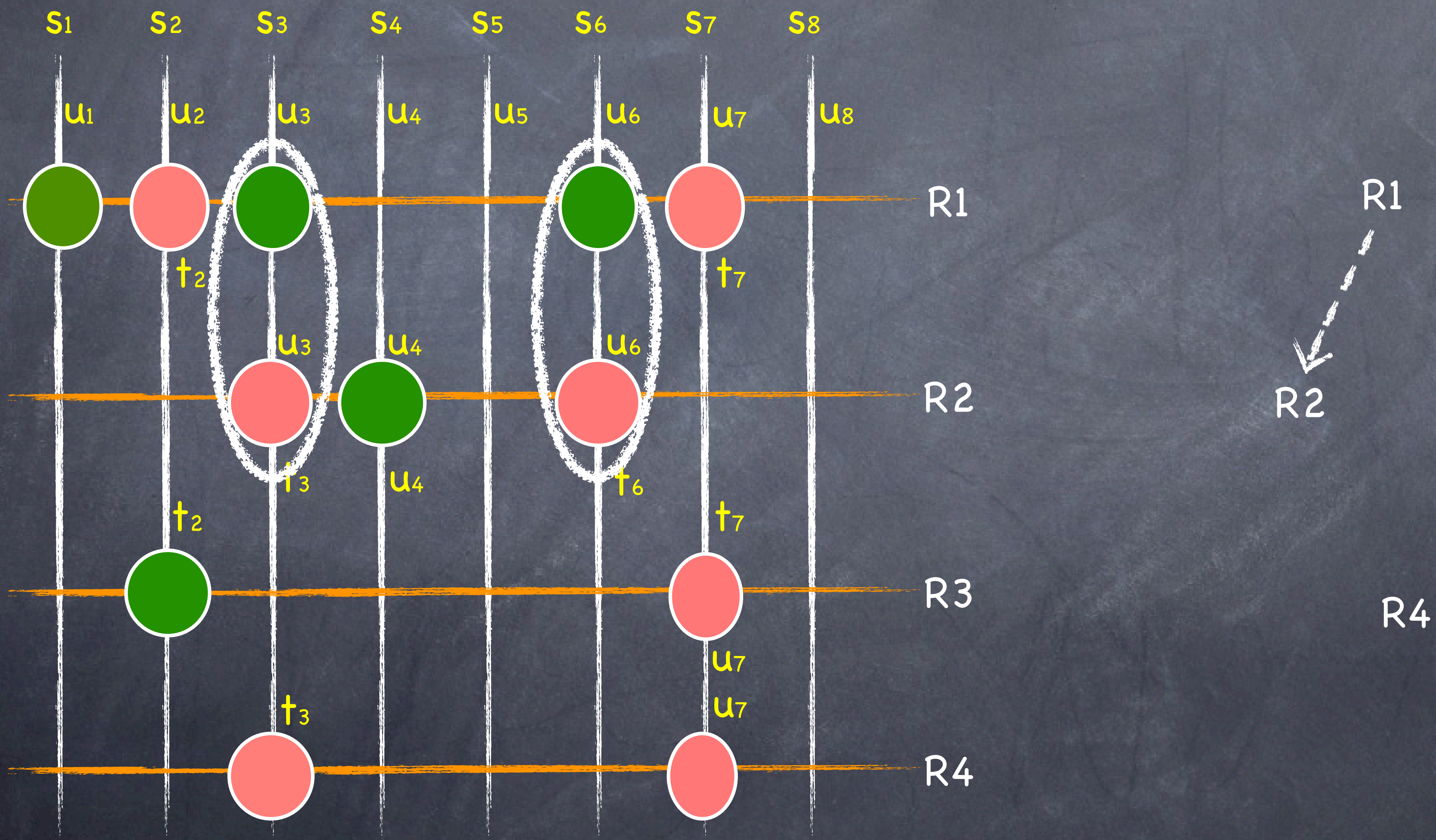


Precedence

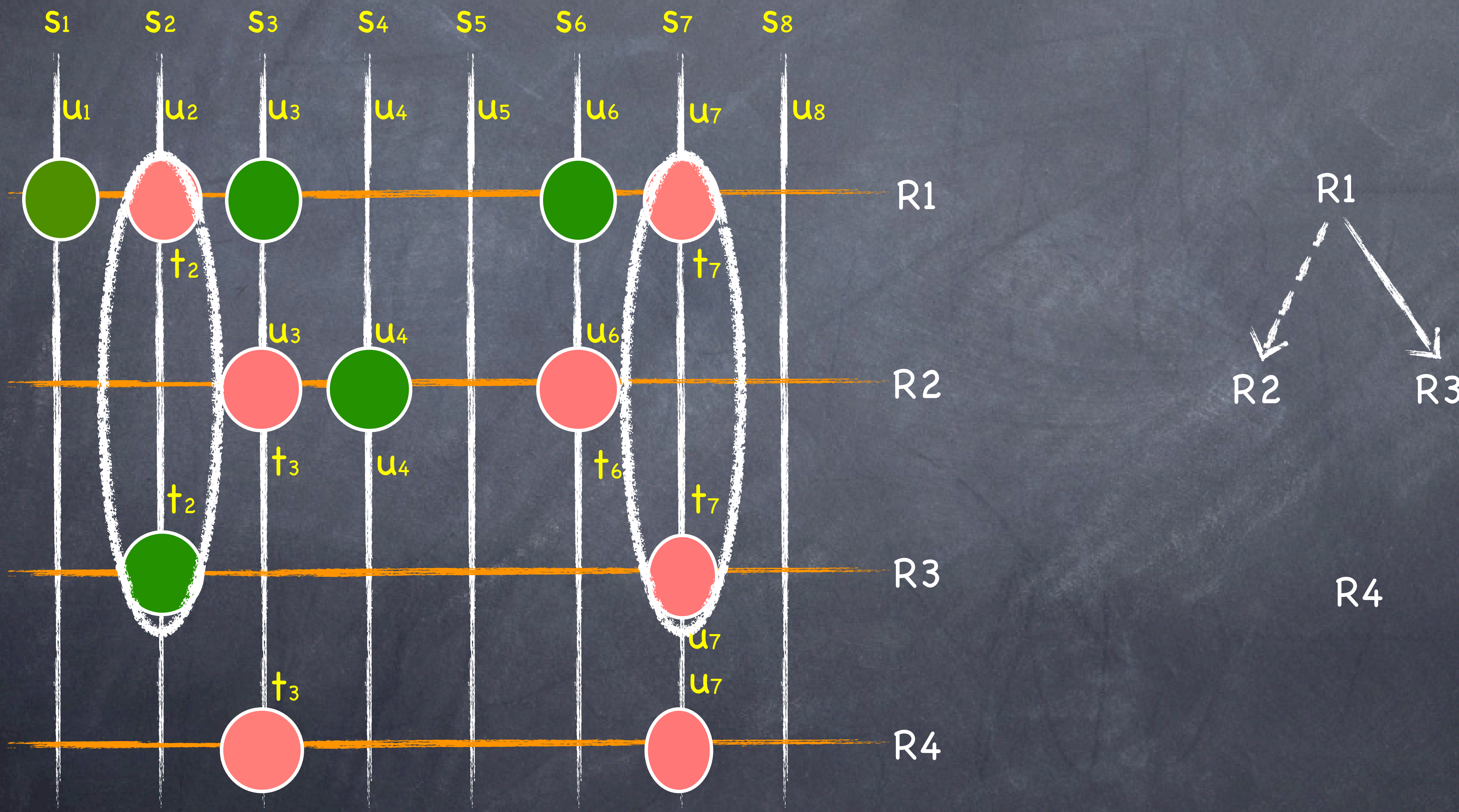


Concurrency

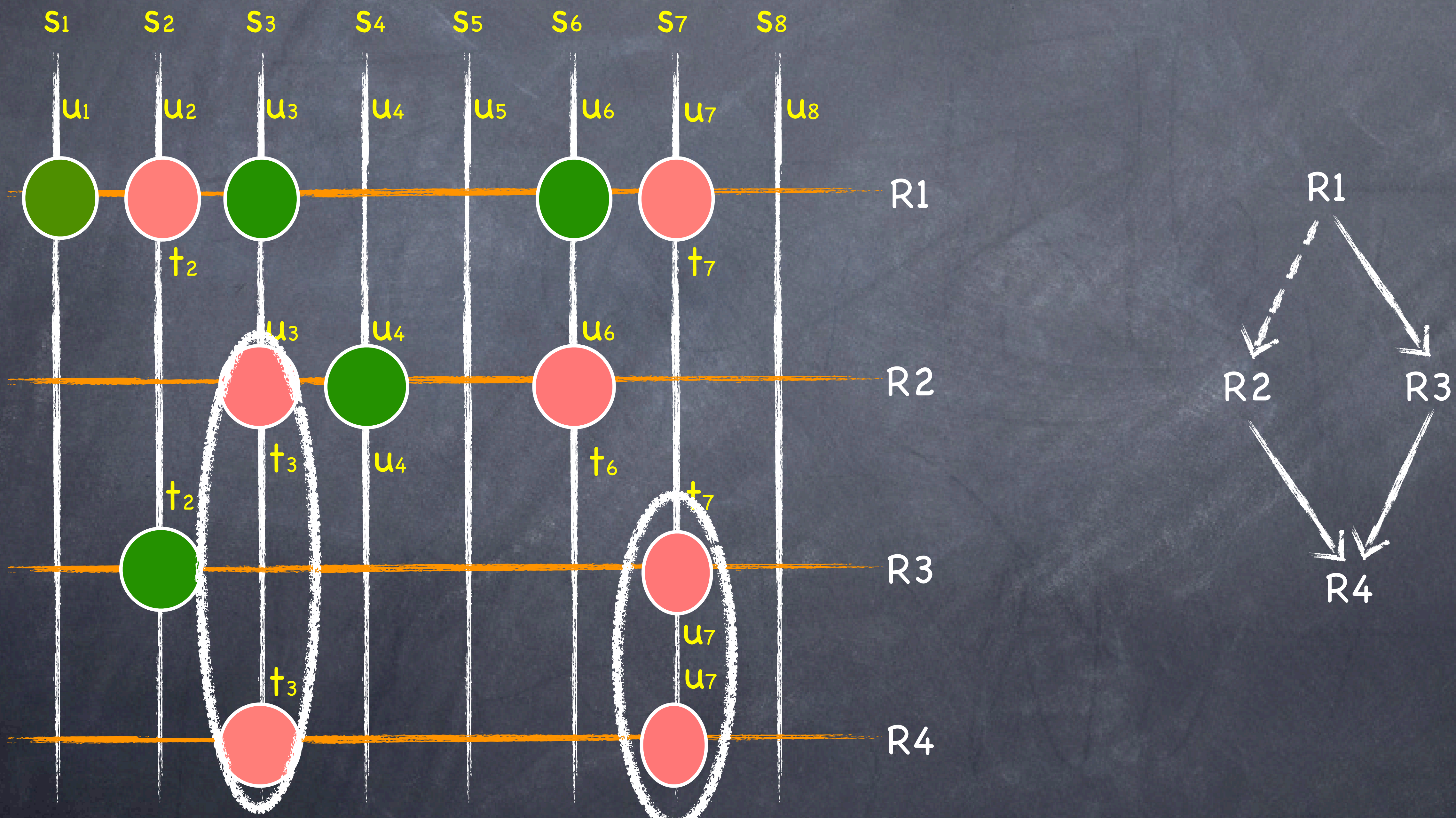
Simple causality analysis



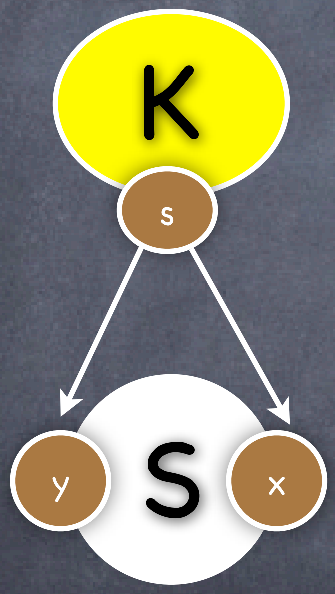
Simple causality analysis



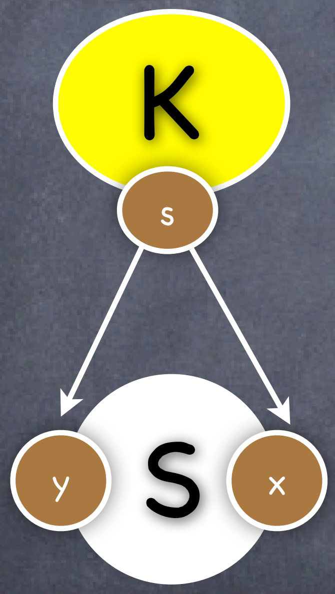
Simple causality analysis

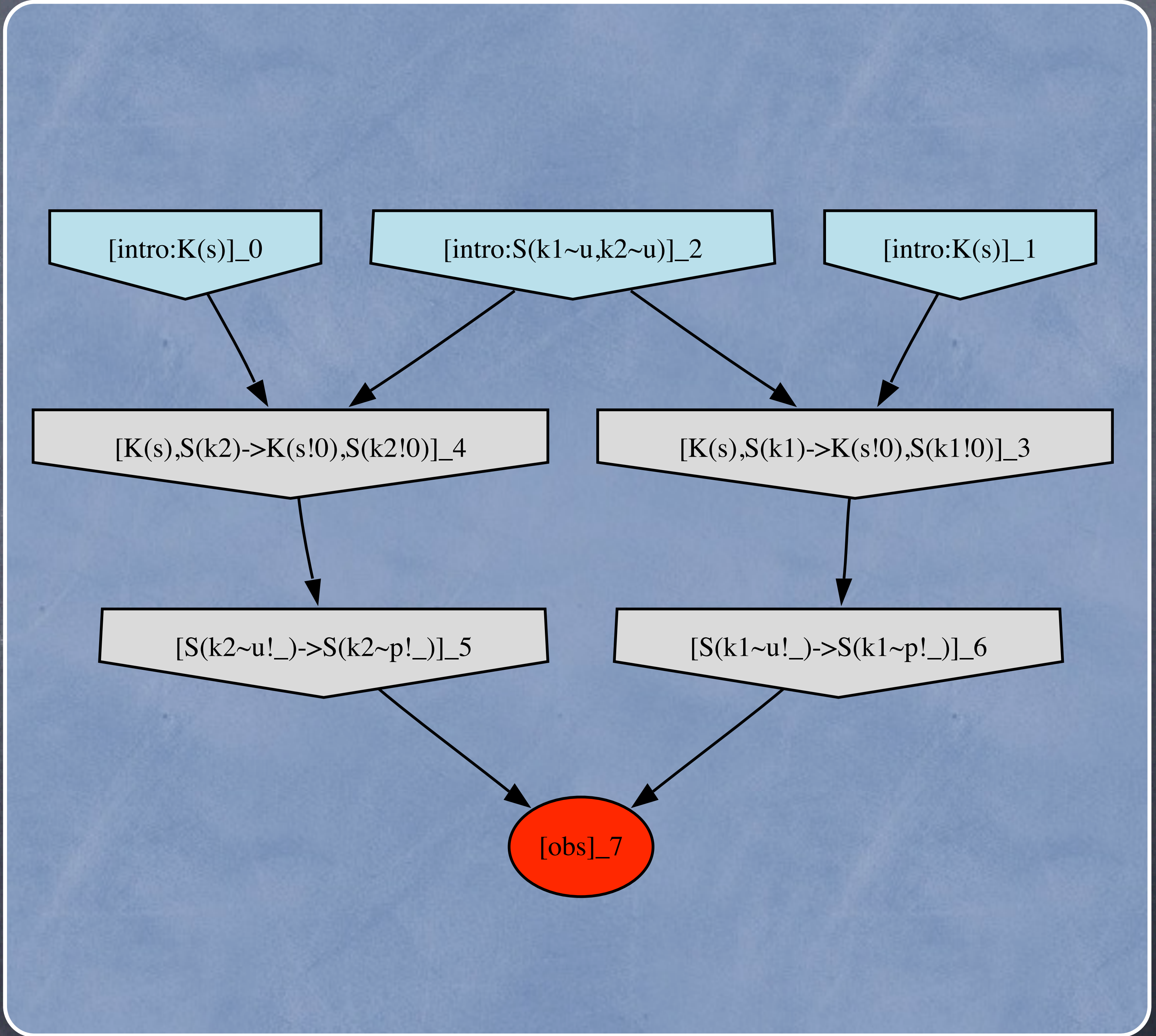


But this is not satisfactory...

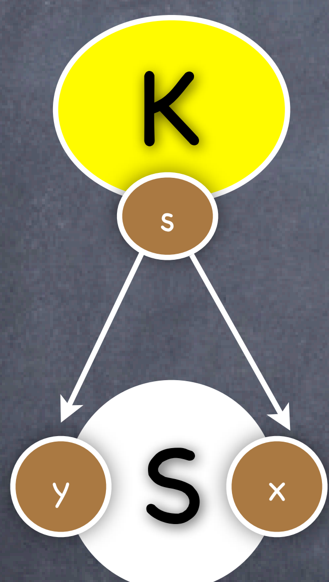


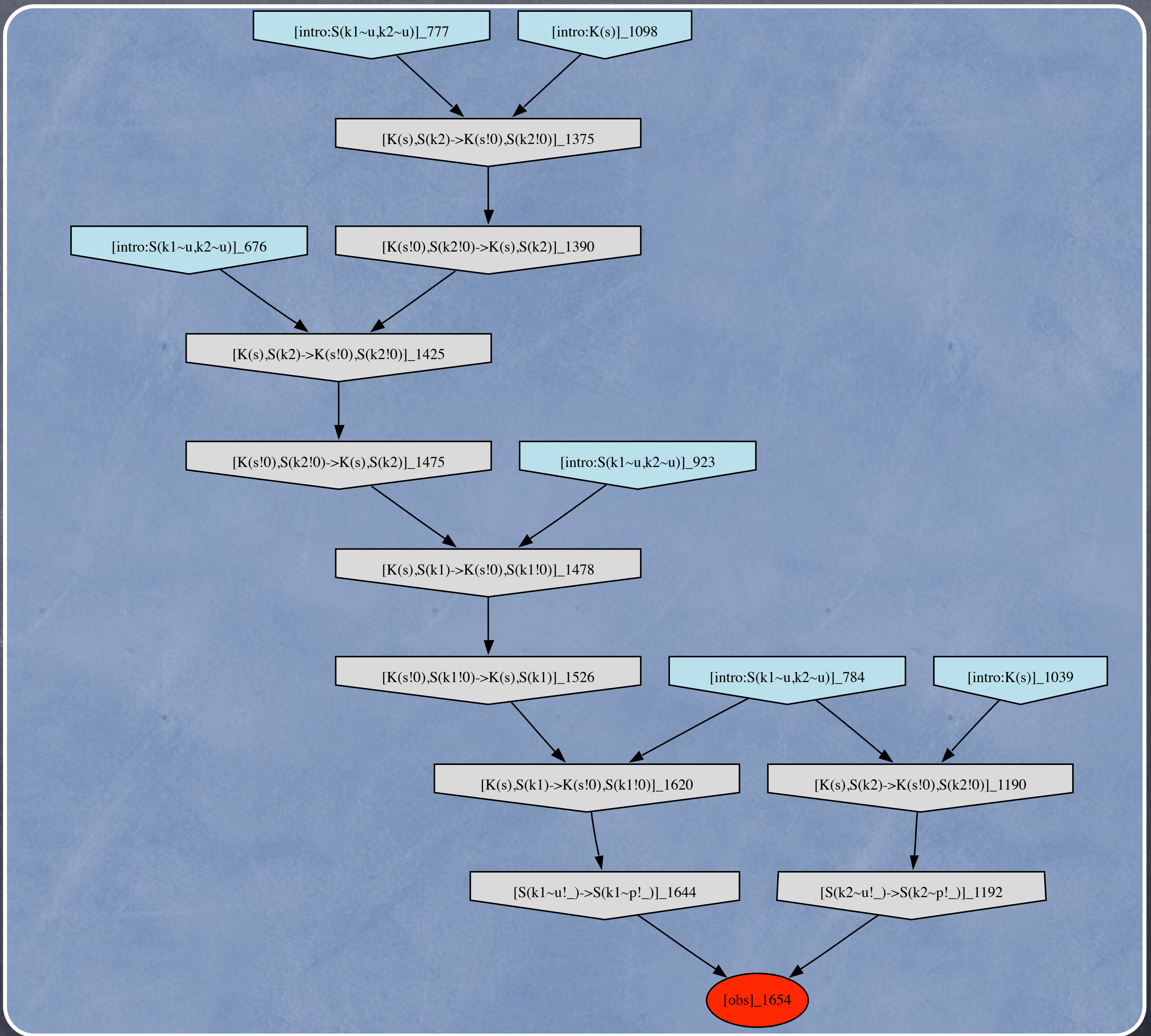
But this is not satisfactory...



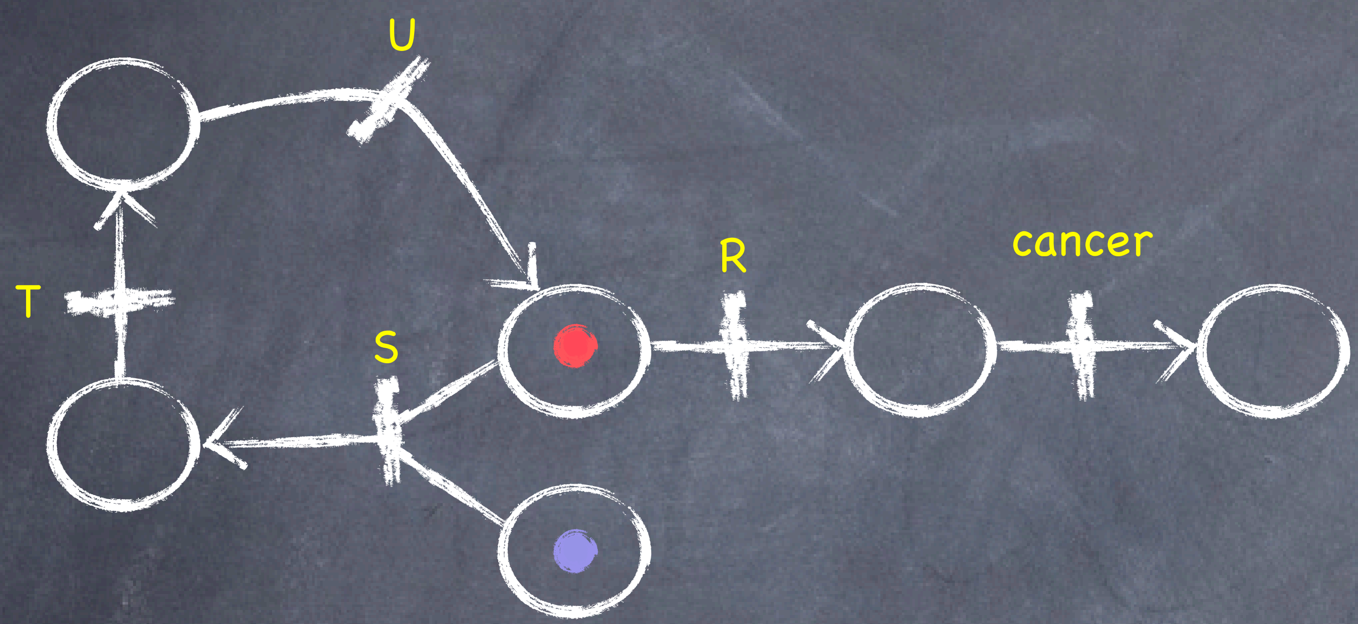


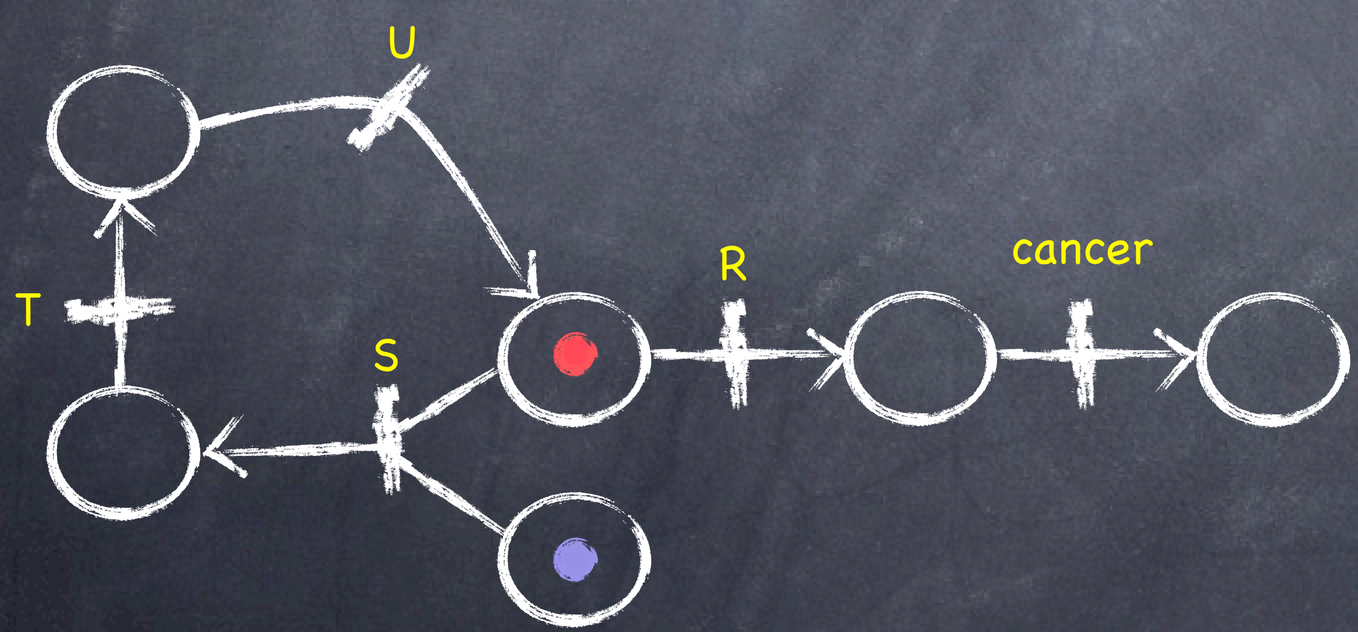
But this is not satisfactory...



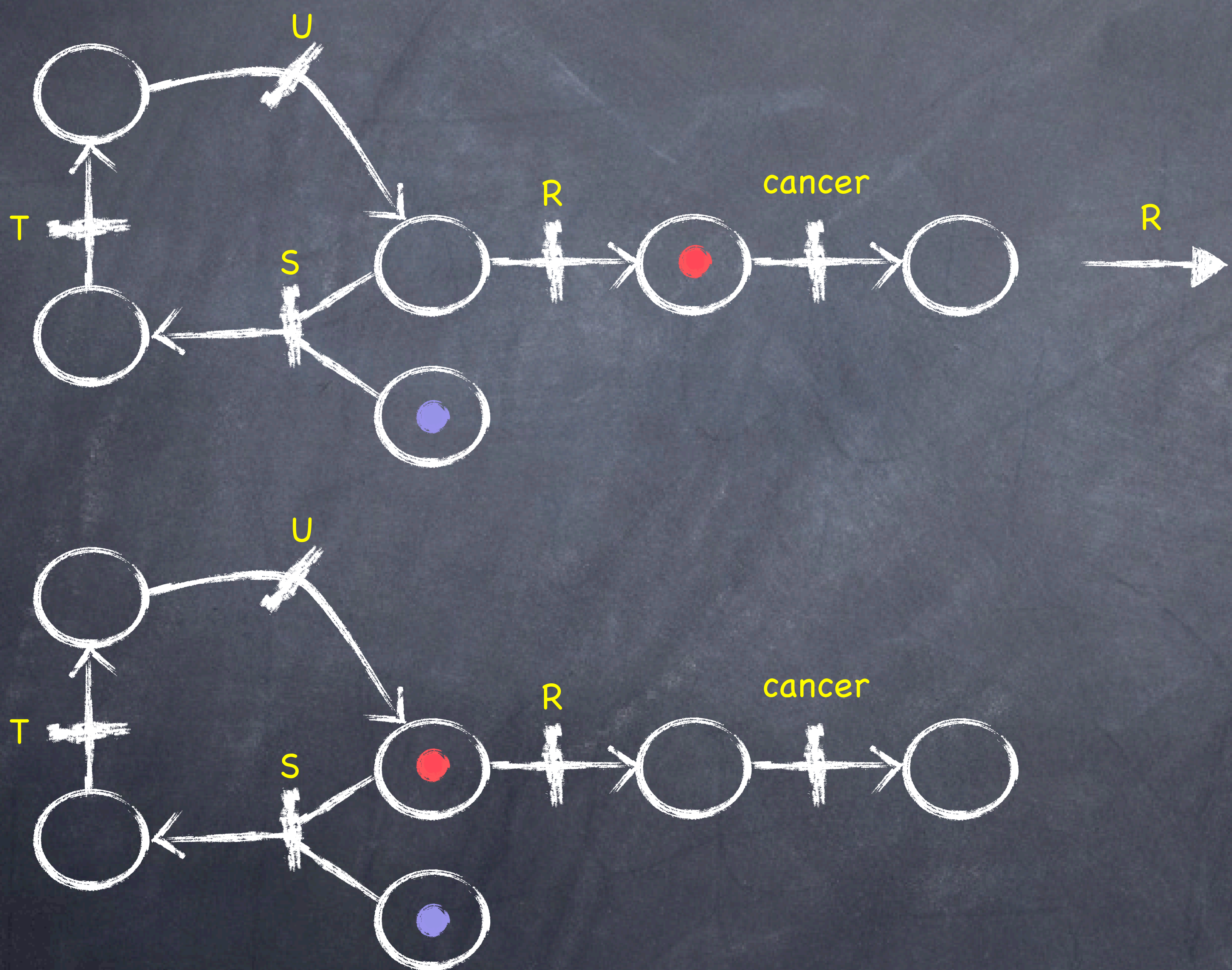


If life was a PT net

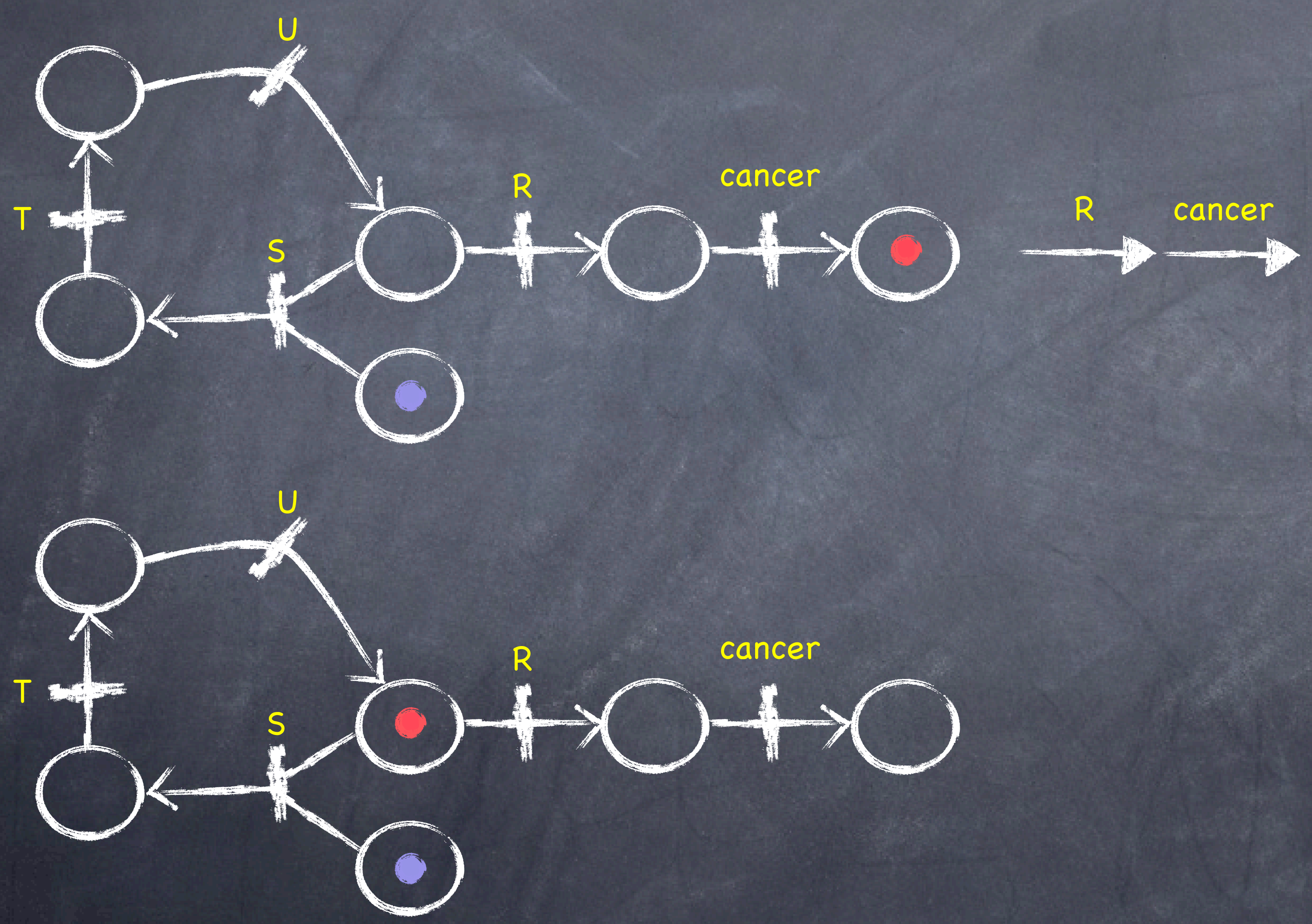




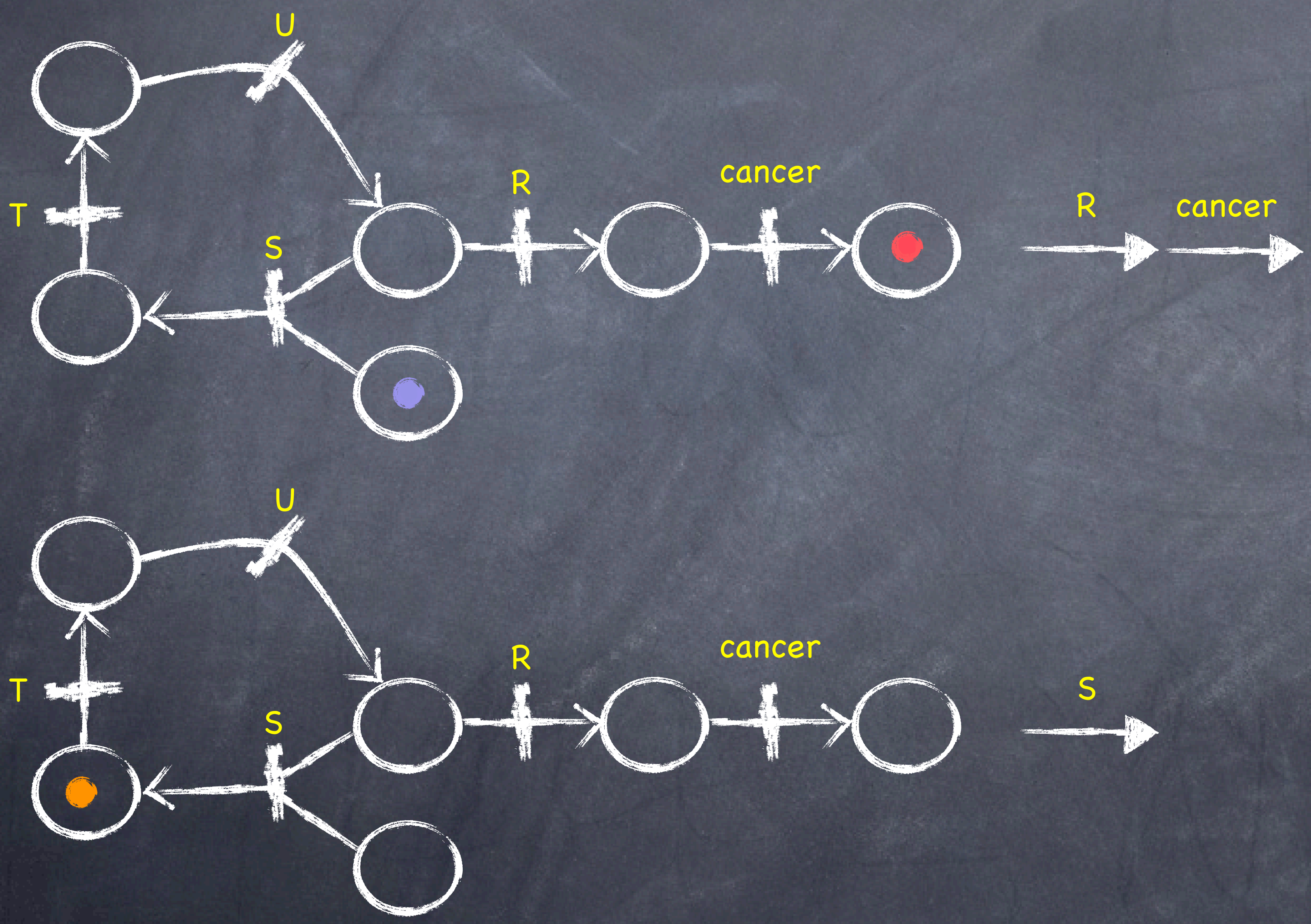
If life was a PT net



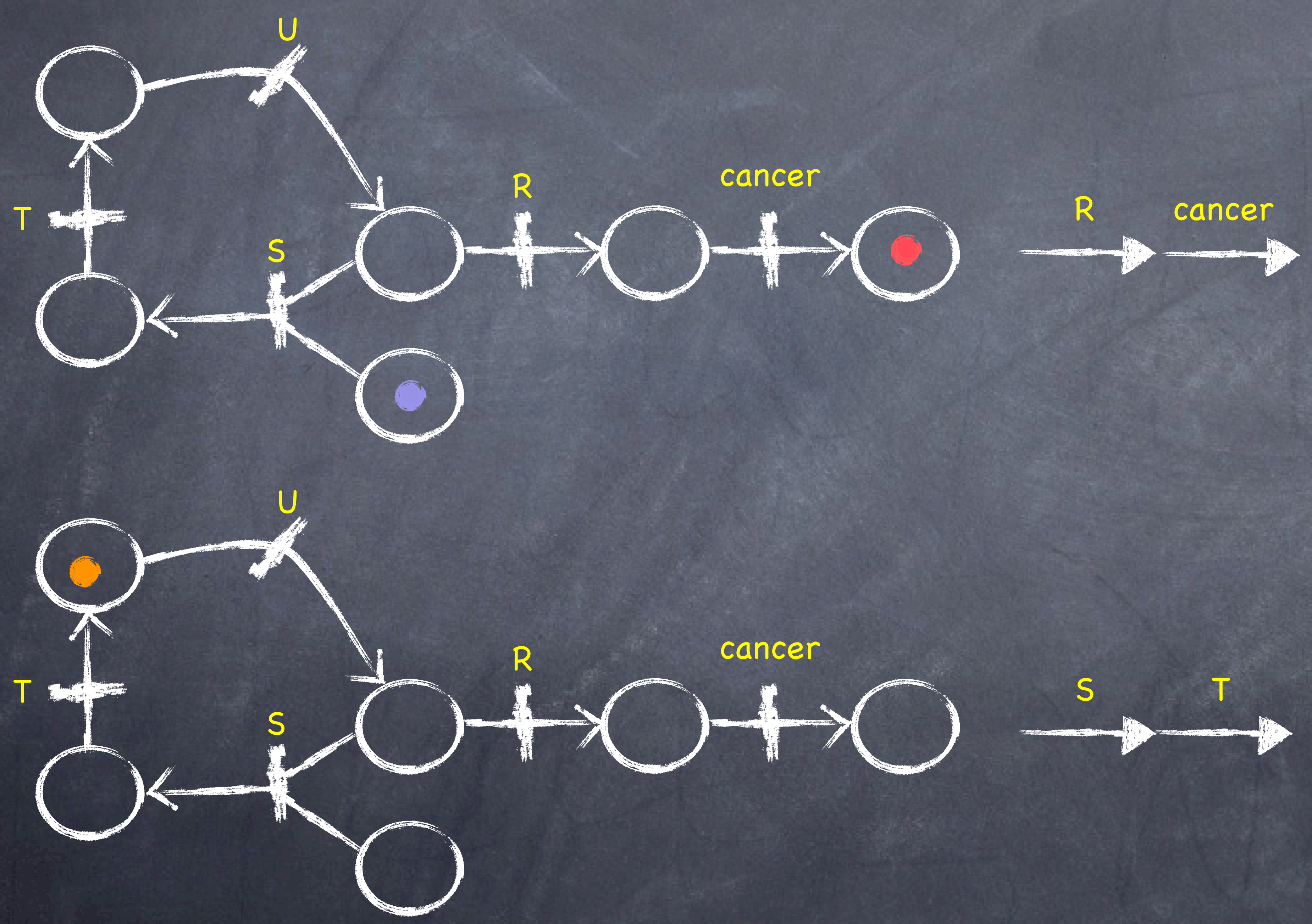
If life was a PT net



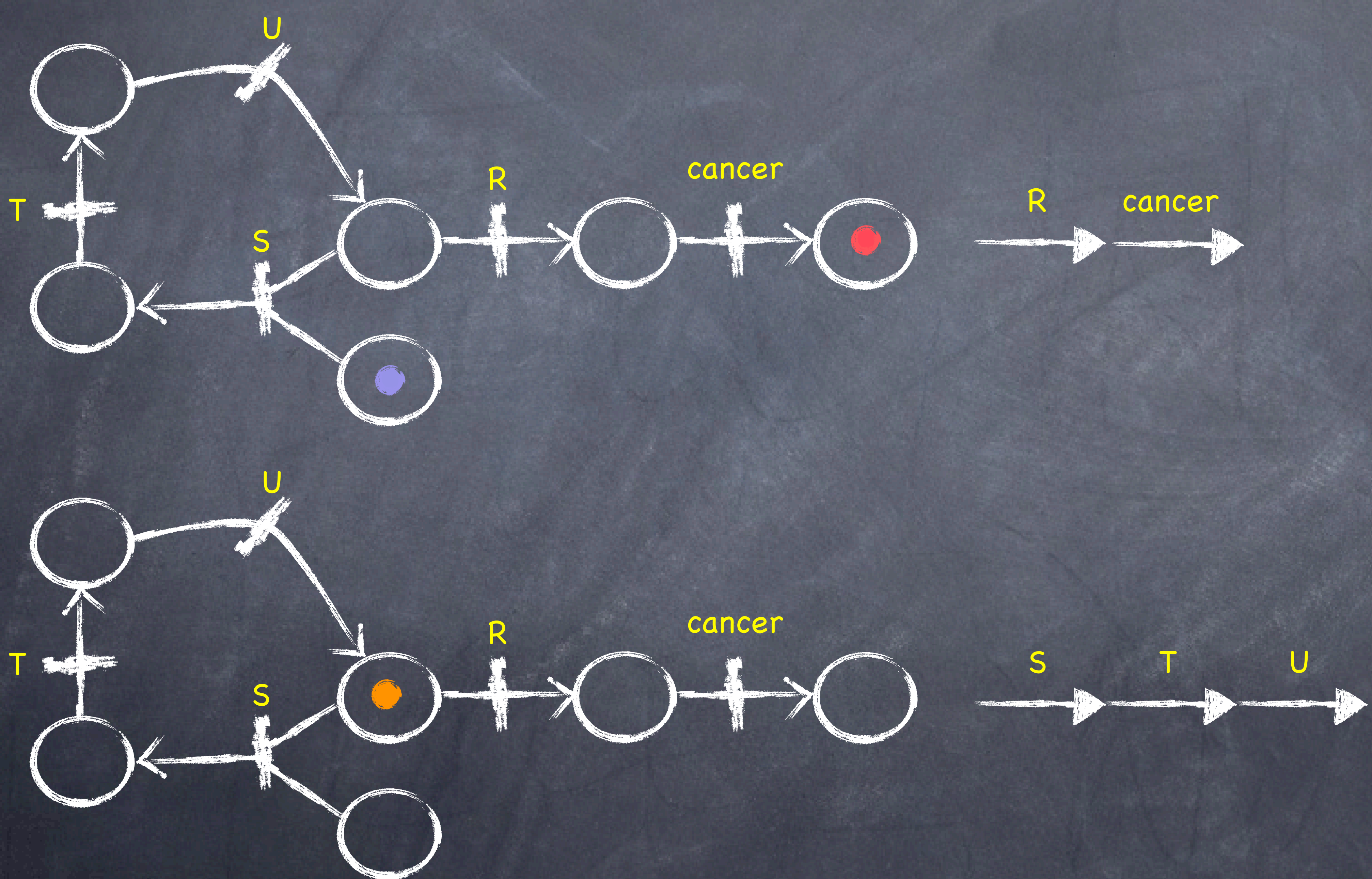
If life was a PT net



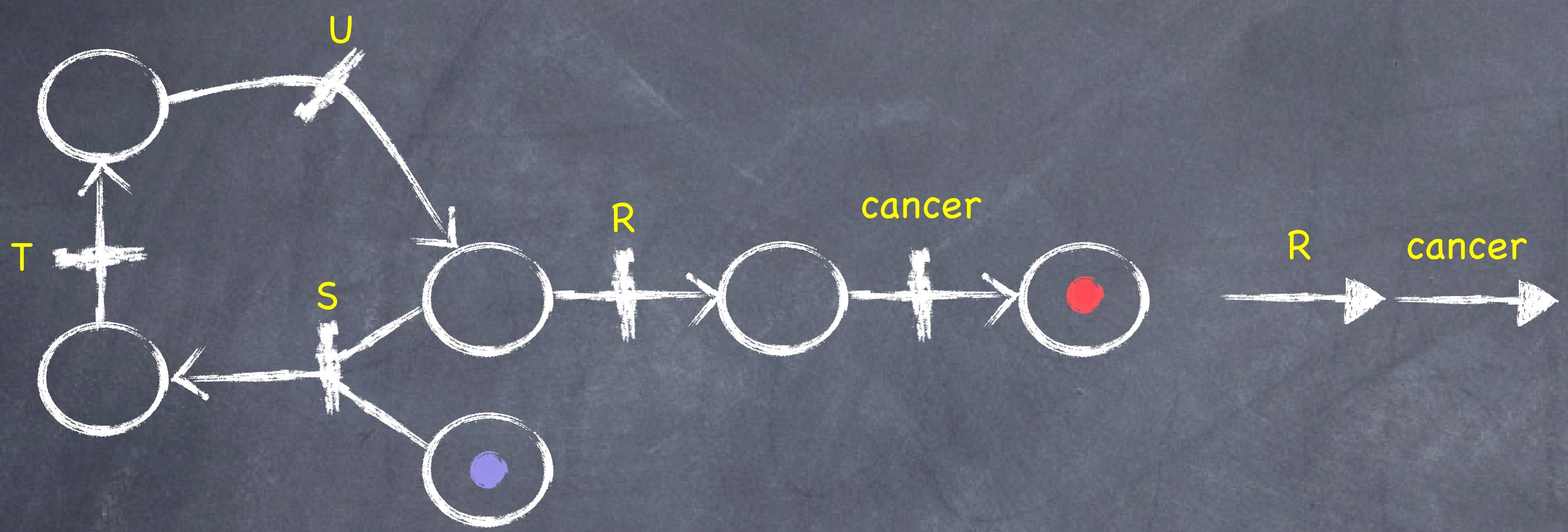
If life was a PT net

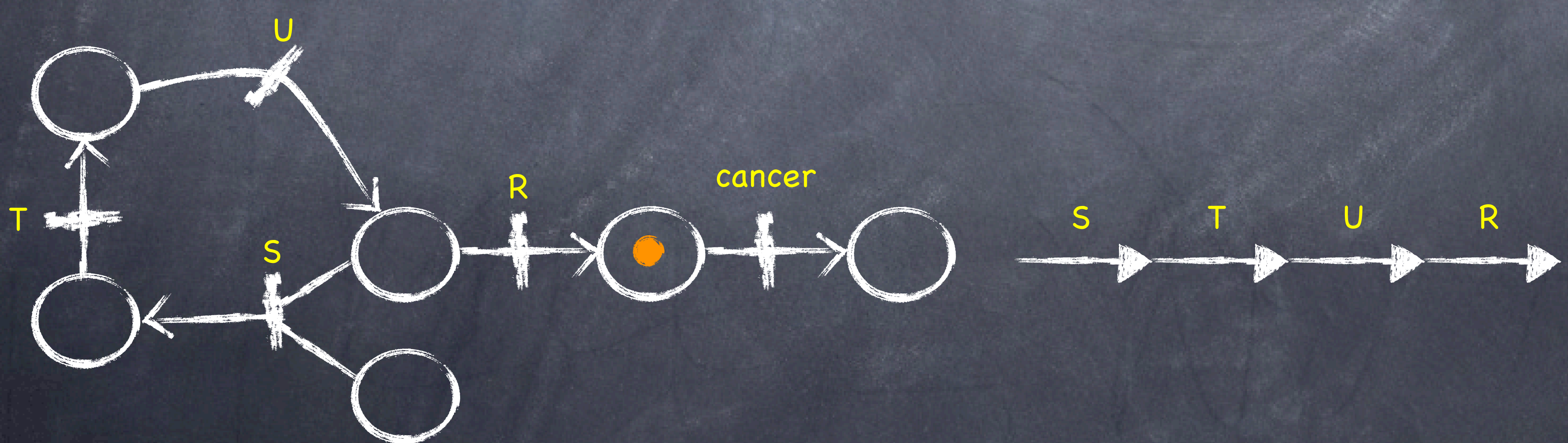


If life was a PT net

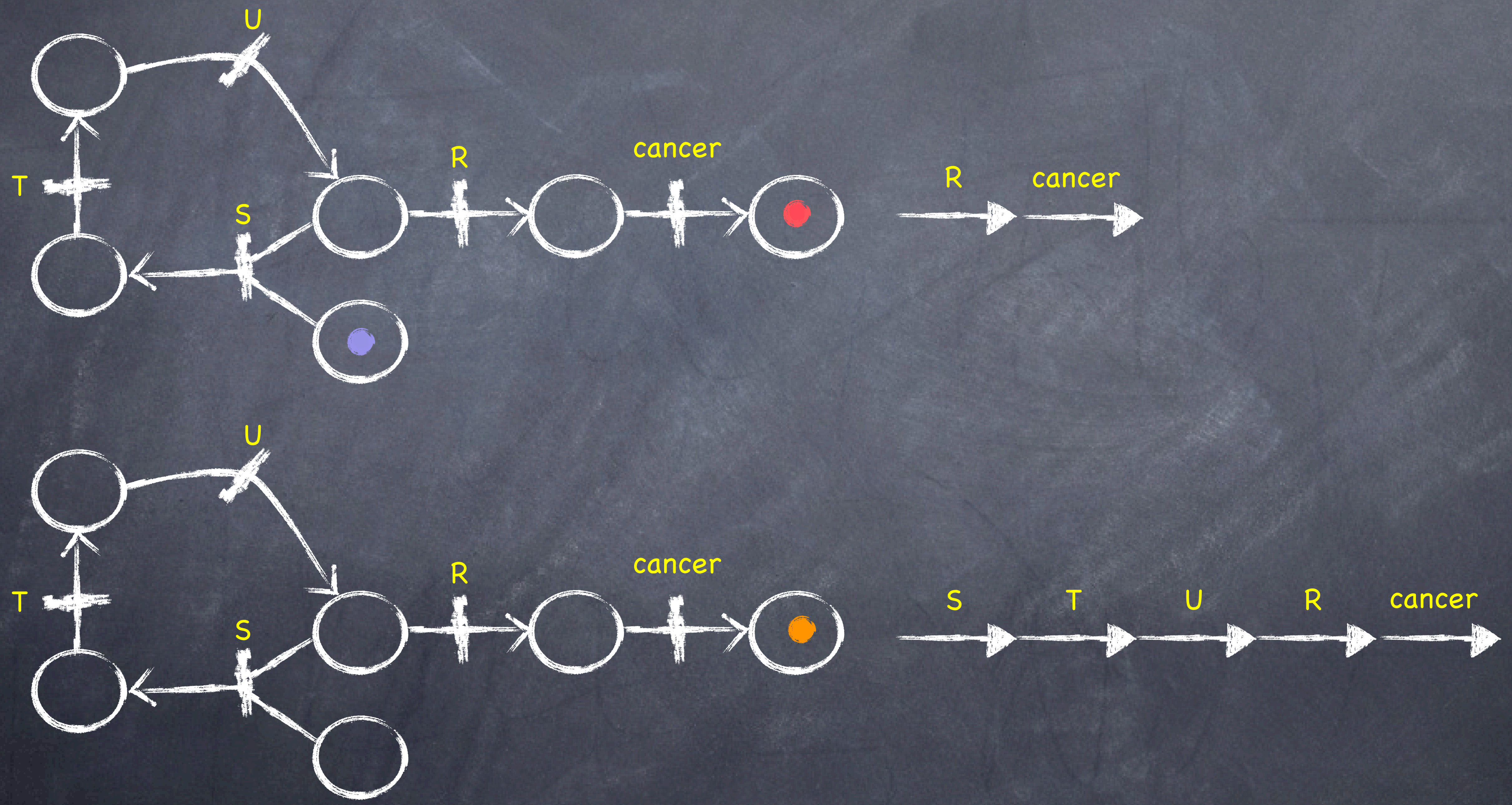


If life was a PT net



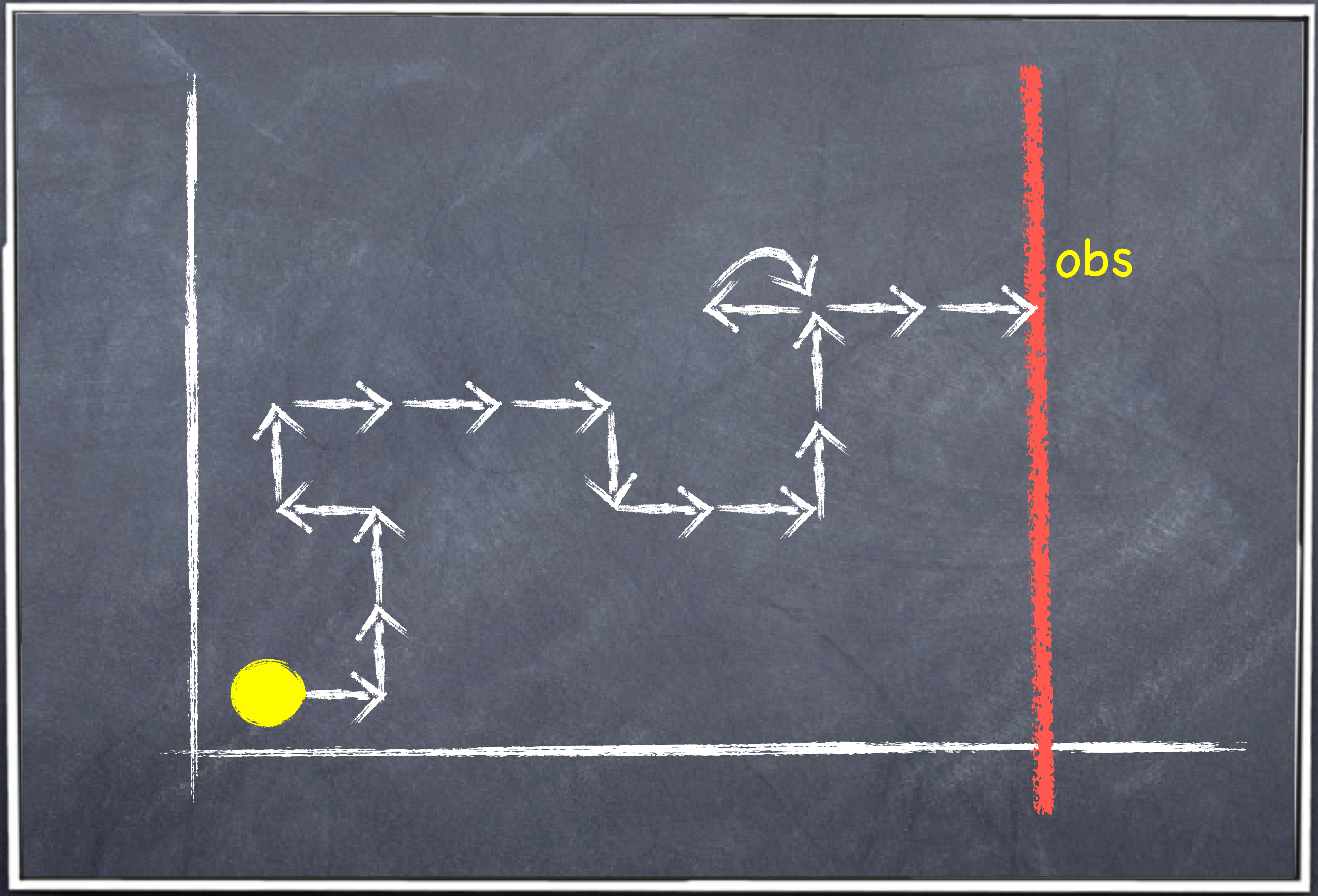


If life was a PT net



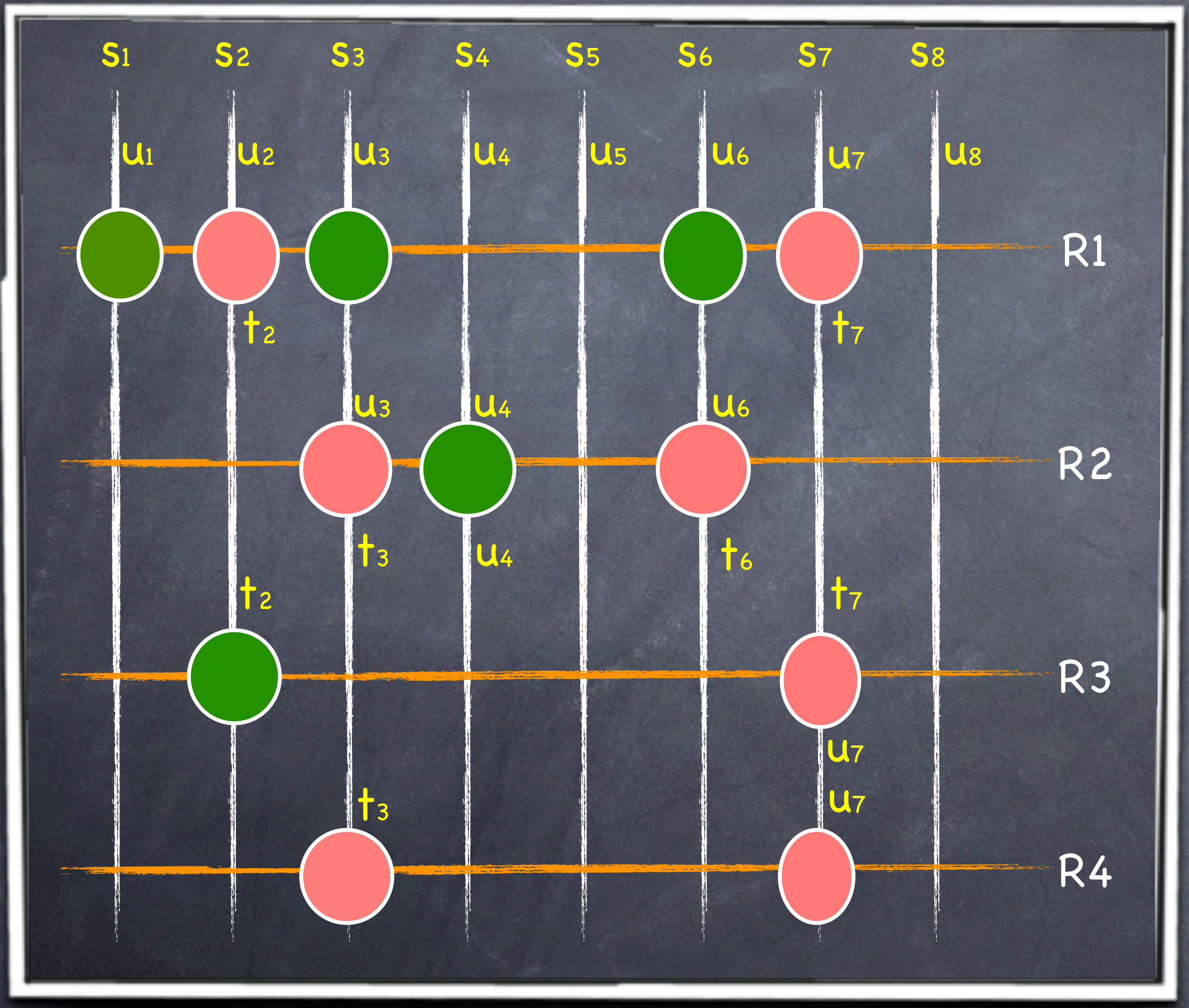
Final states don't match but the difference is not observed

Restricting to what can be observed…



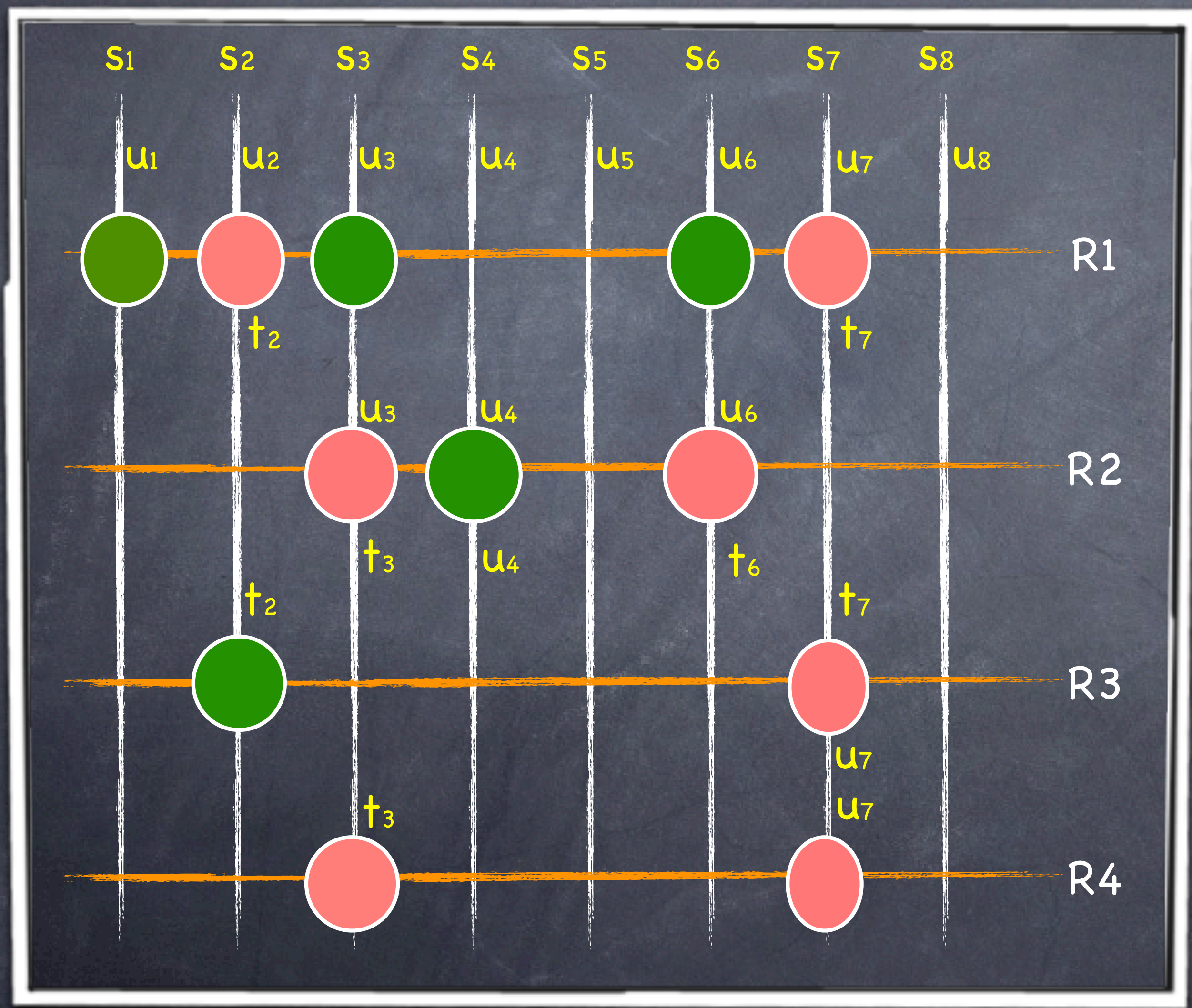
The “knock-out” property

Is this a causal trace ?



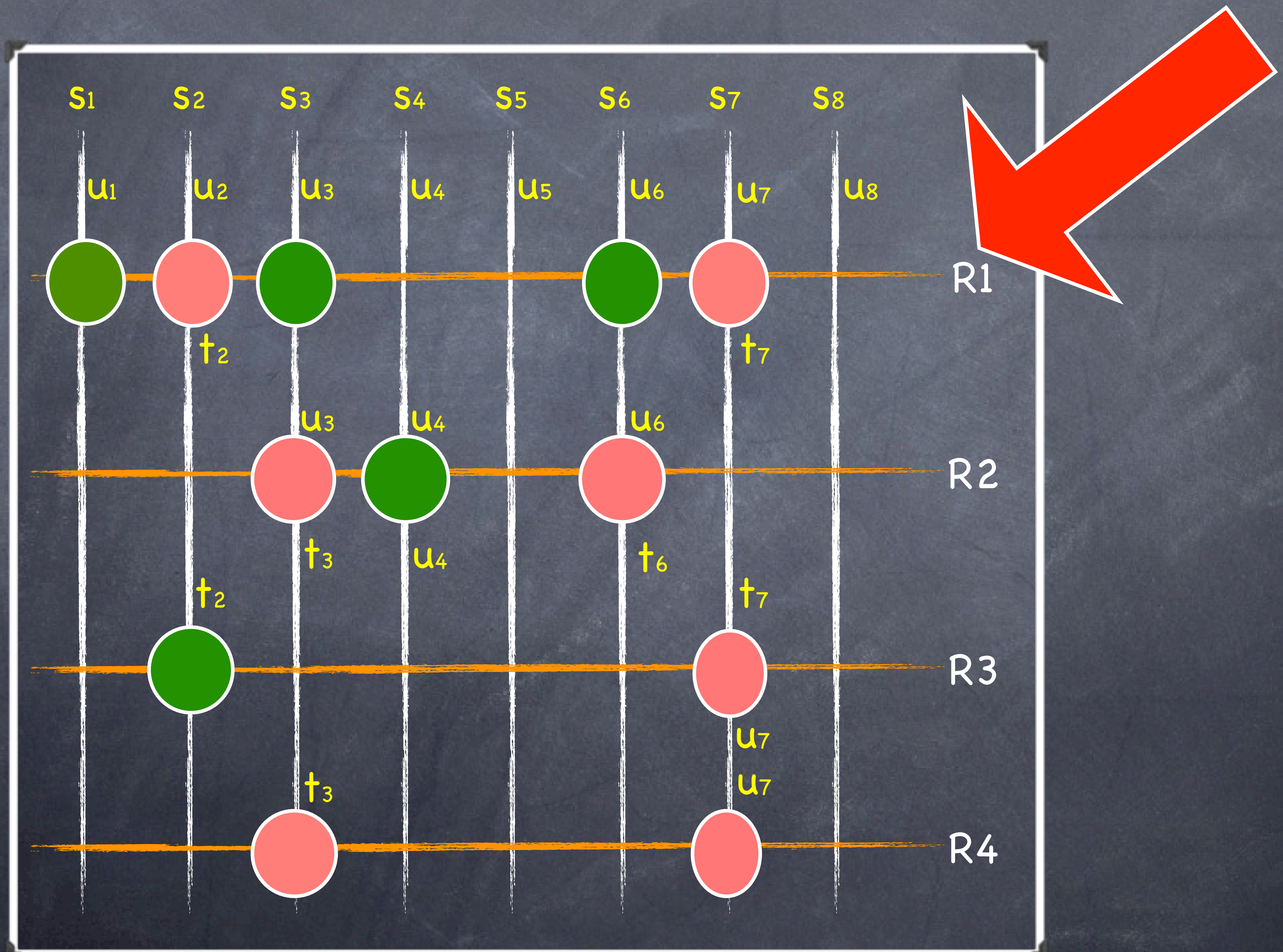
"Yes, because each event is a local cause of the next one"

Is this a causal trace ?

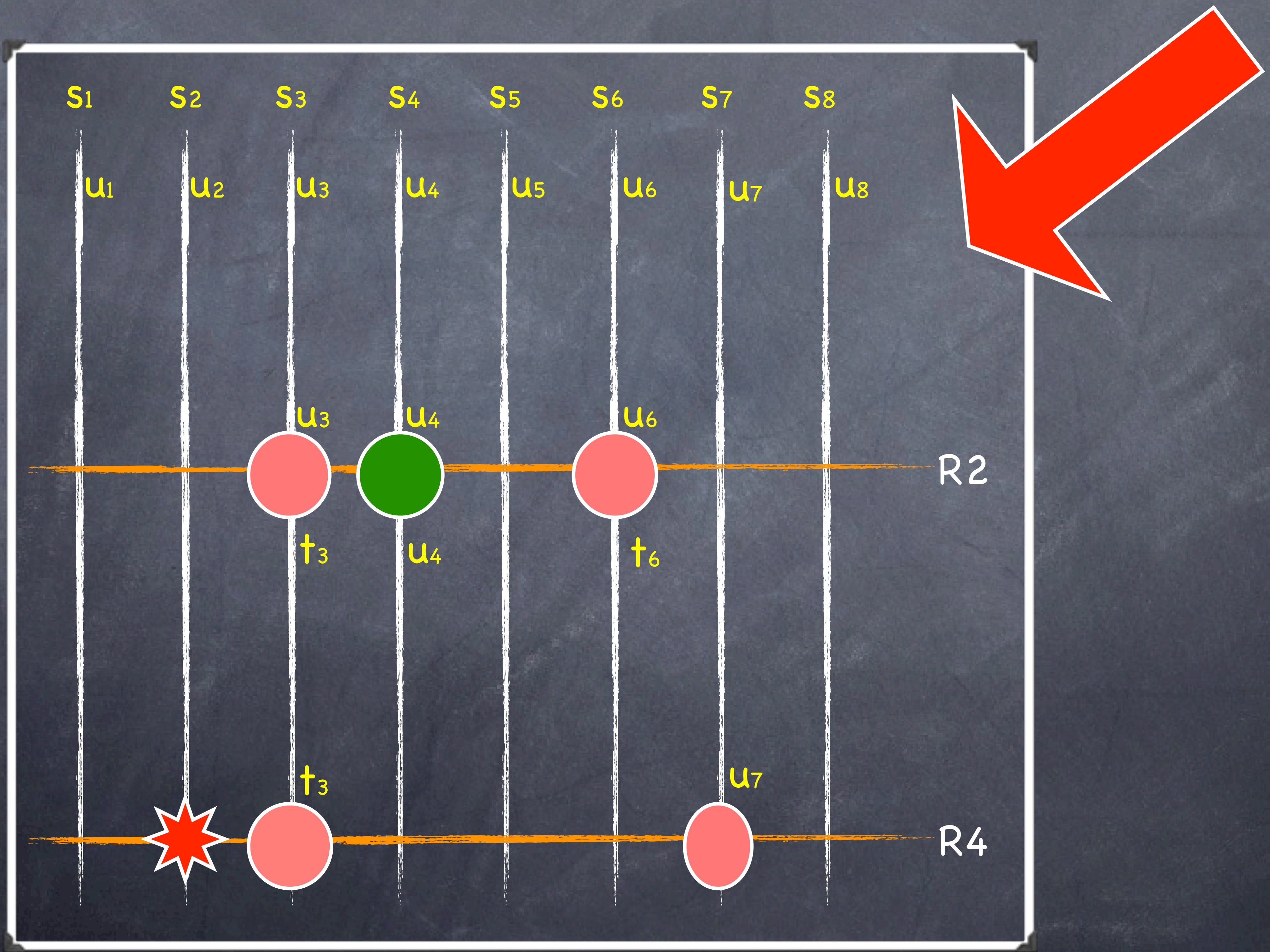


"No, because I can knock-out R1 and still observe R4!"

Knock-out and stabilization



Knock-out and stabilization



Causal Compression

Rule application

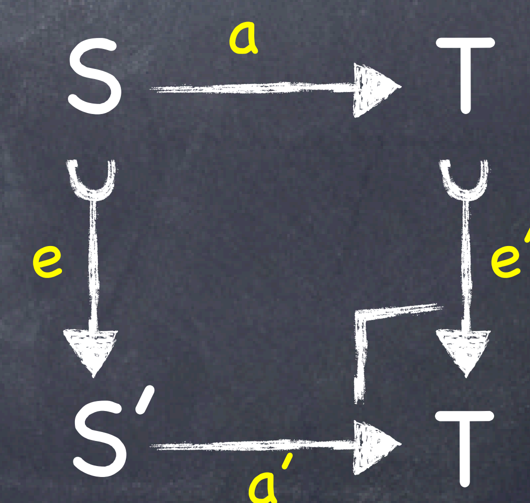
A rule denote an action:

$$S \xrightarrow{a} T$$

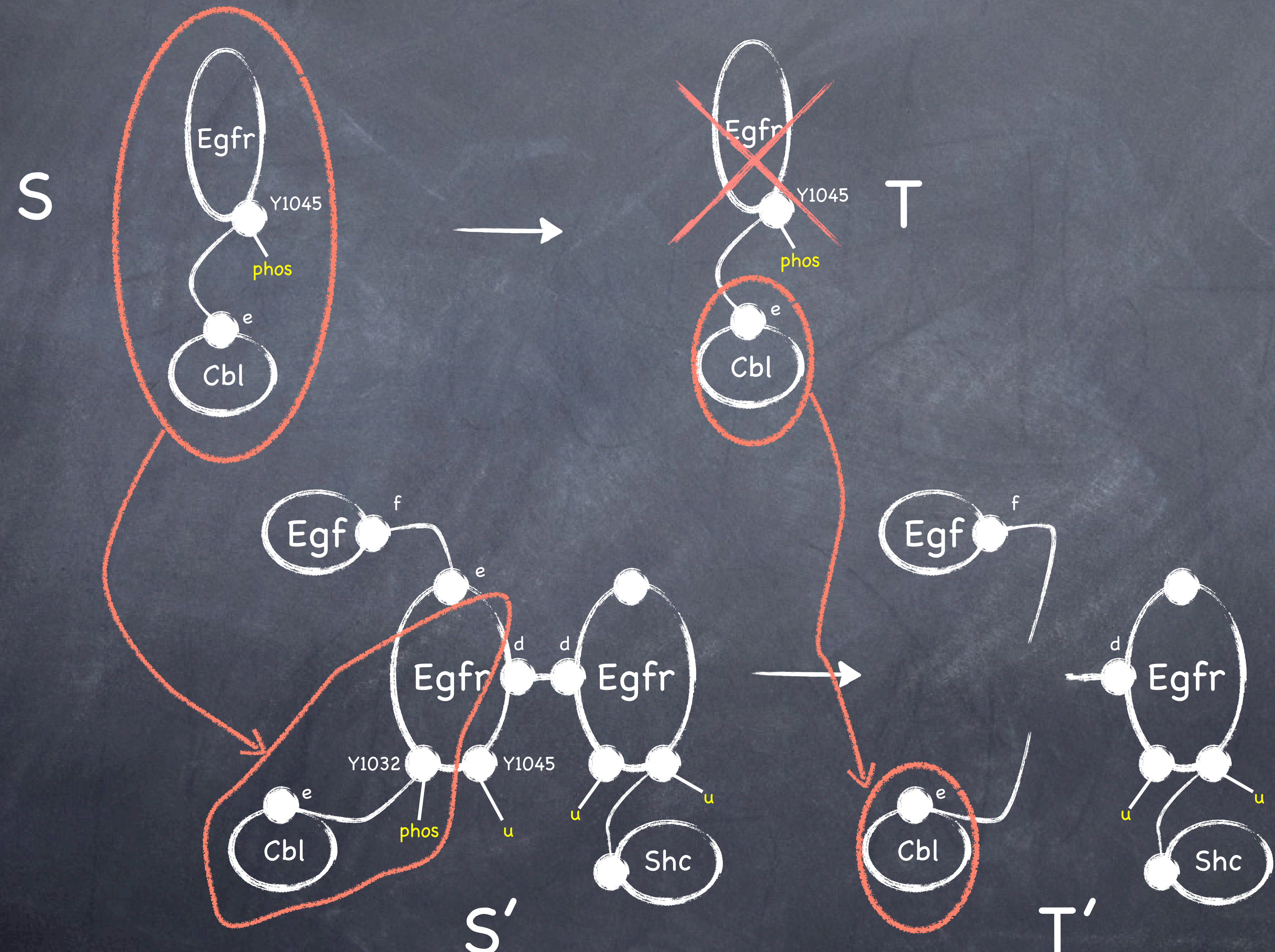
An application of such rule is determined by an embedding:

$$S \overset{e}{\rightarrowtail} S'$$

Its effect is given by the pushout:

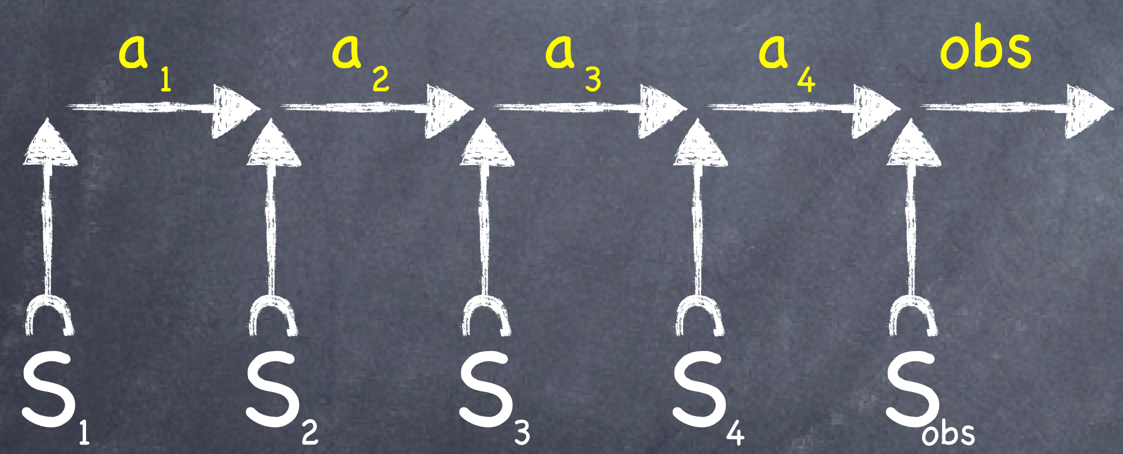


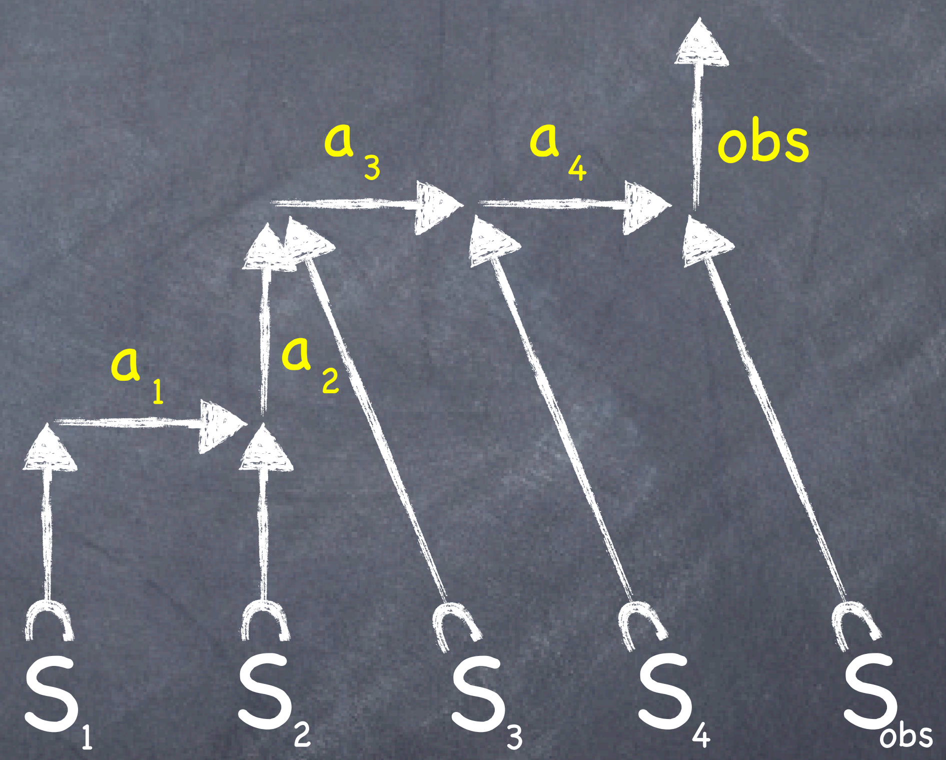
For instance



Simple compression

A trace (obtained by simulation)

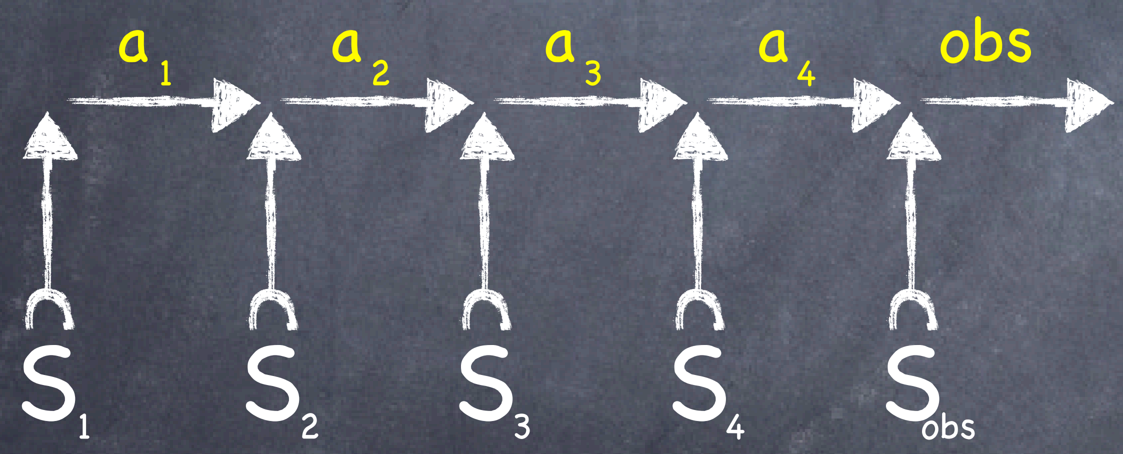


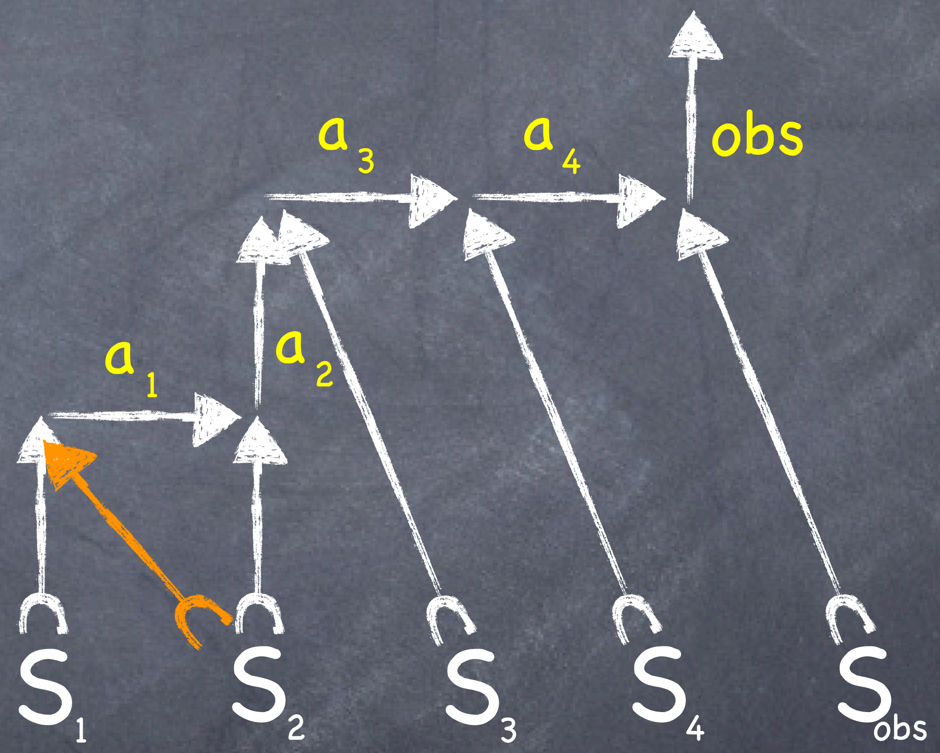


Assume the above partition of arrows is given by an oracle

Simple compression

A trace (obtained by simulation)

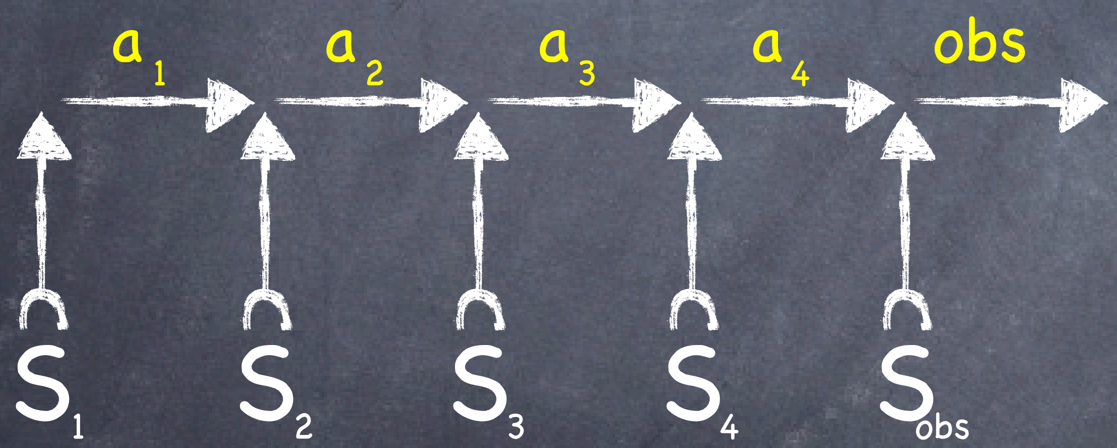


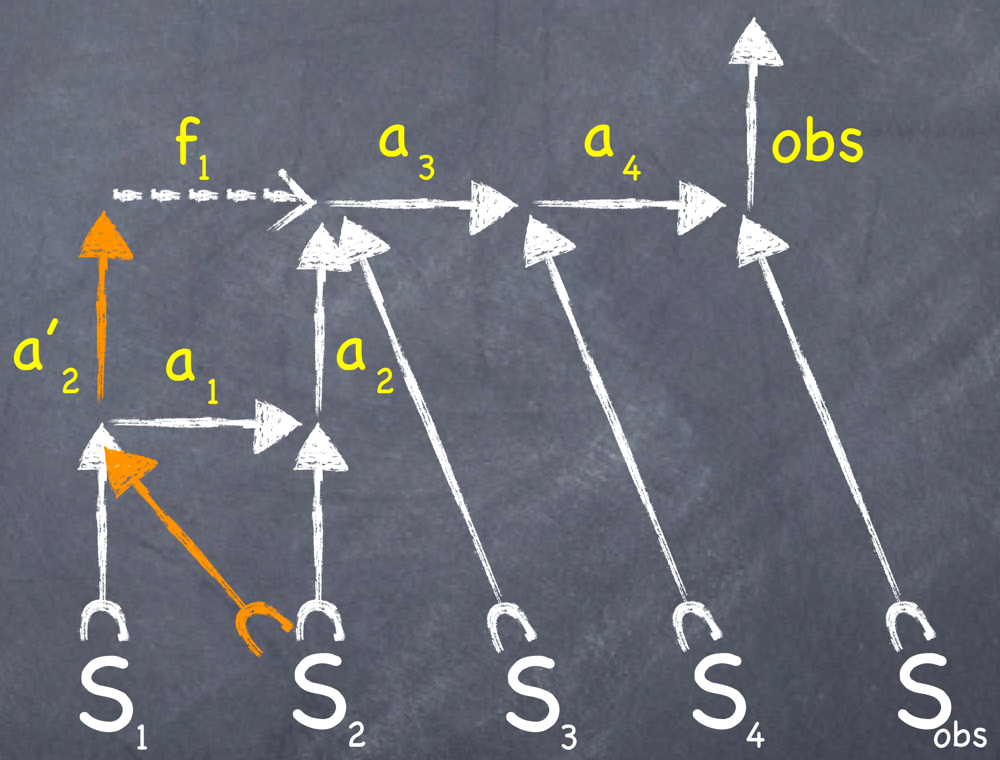


Assume the above partition of arrows is given by an oracle

Simple compression

A trace (obtained by simulation)

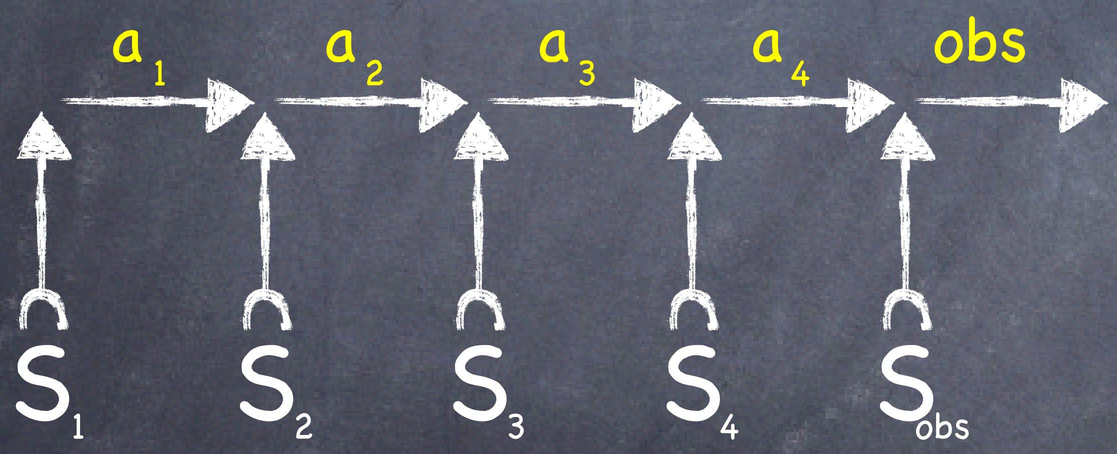


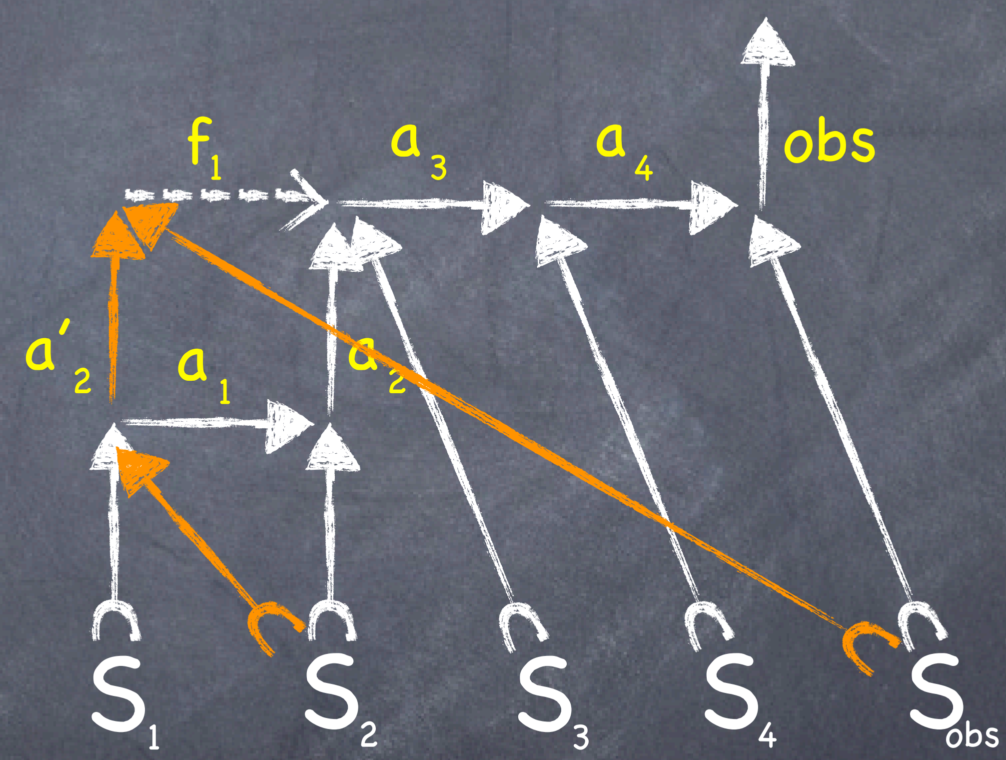


Assume the above partition of arrows is given by an oracle

Simple compression

A trace (obtained by simulation)

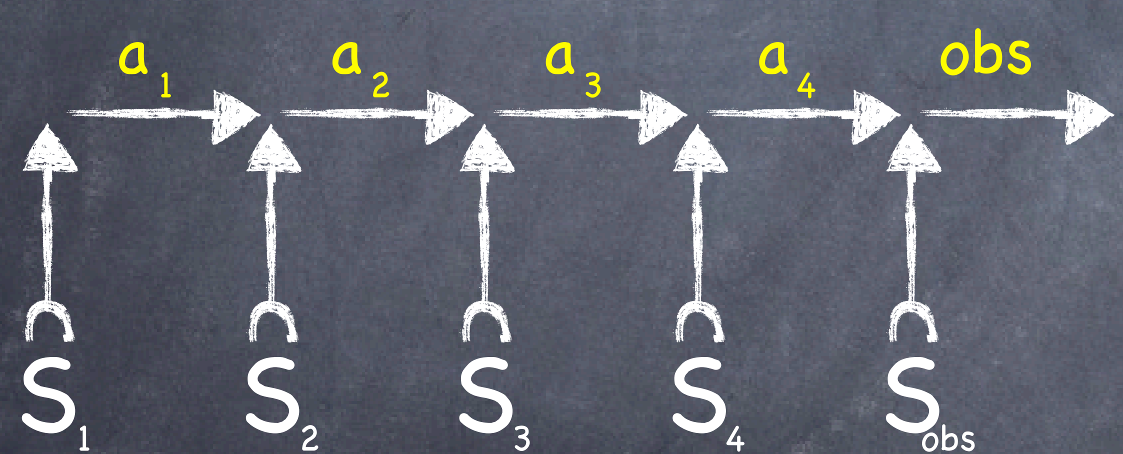


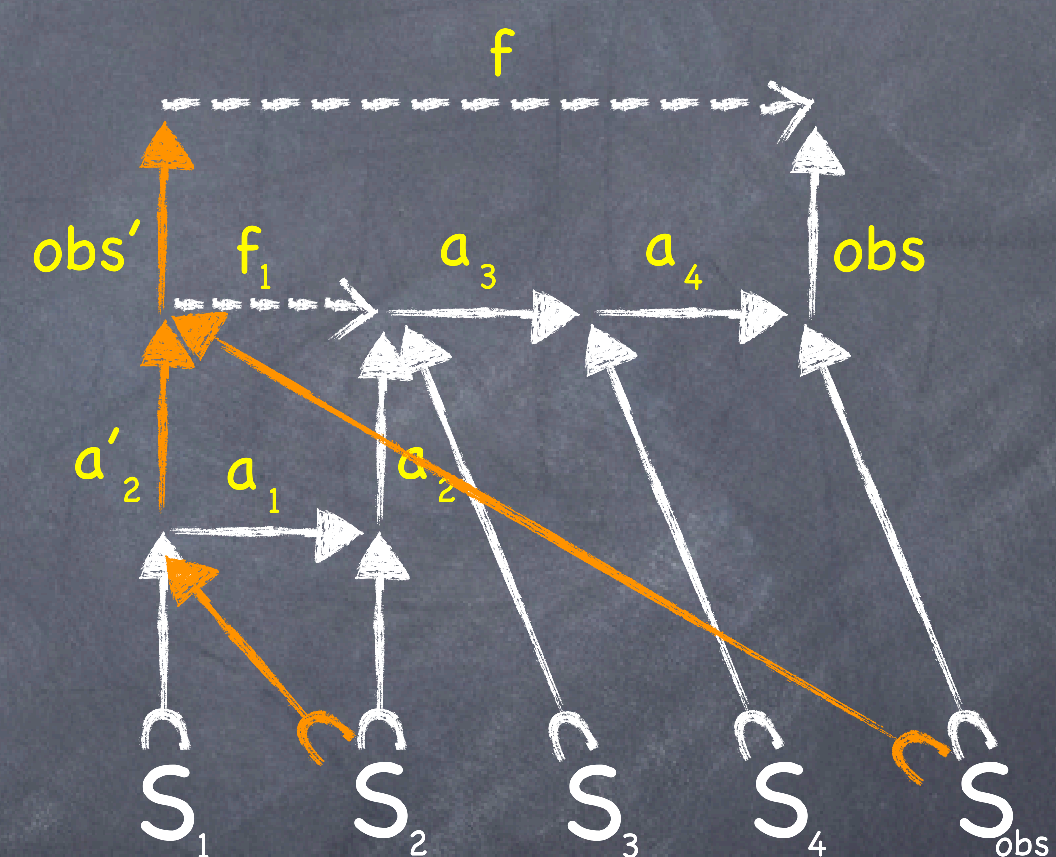


Assume the above partition of arrows is given by an oracle

Simple compression

A trace (obtained by simulation)

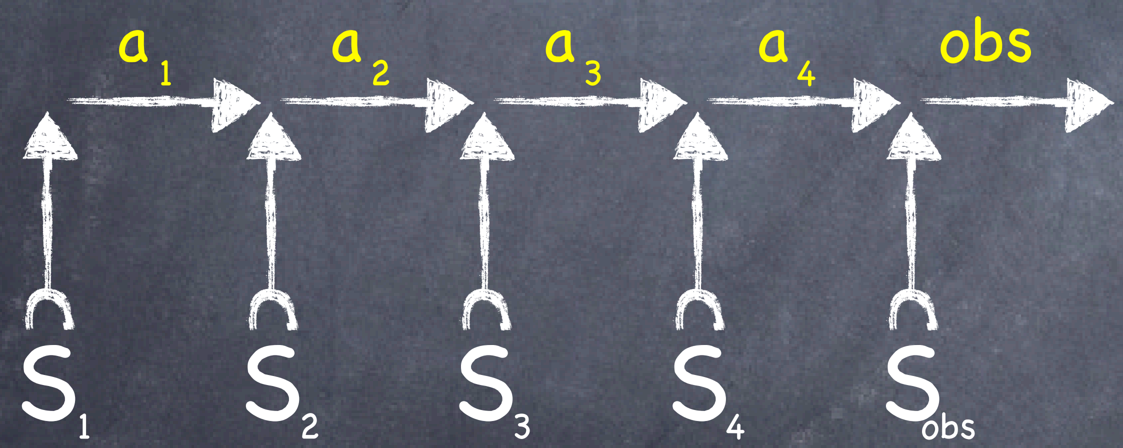


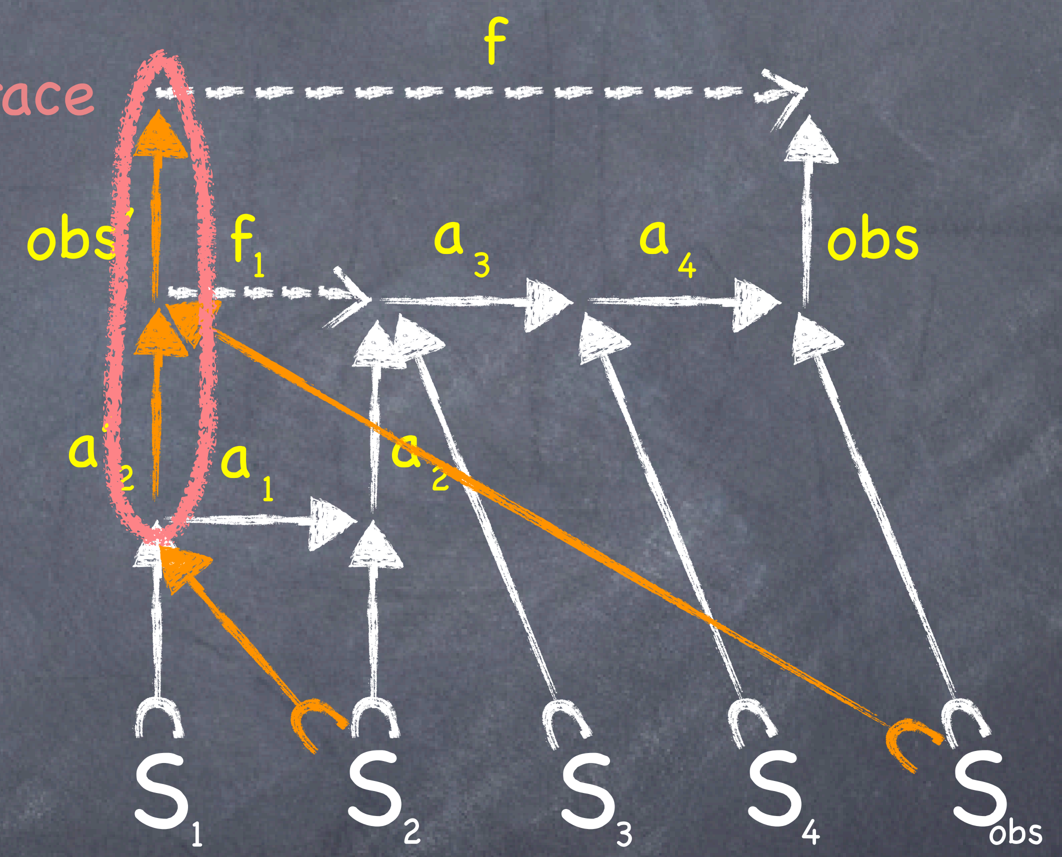


Assume the above partition of arrows is given by an oracle

Simple compression

A trace (obtained by simulation)

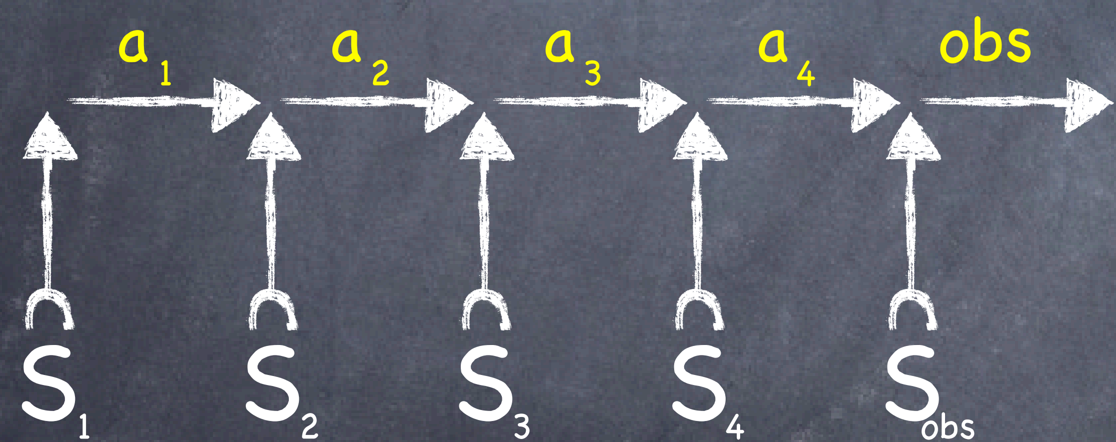


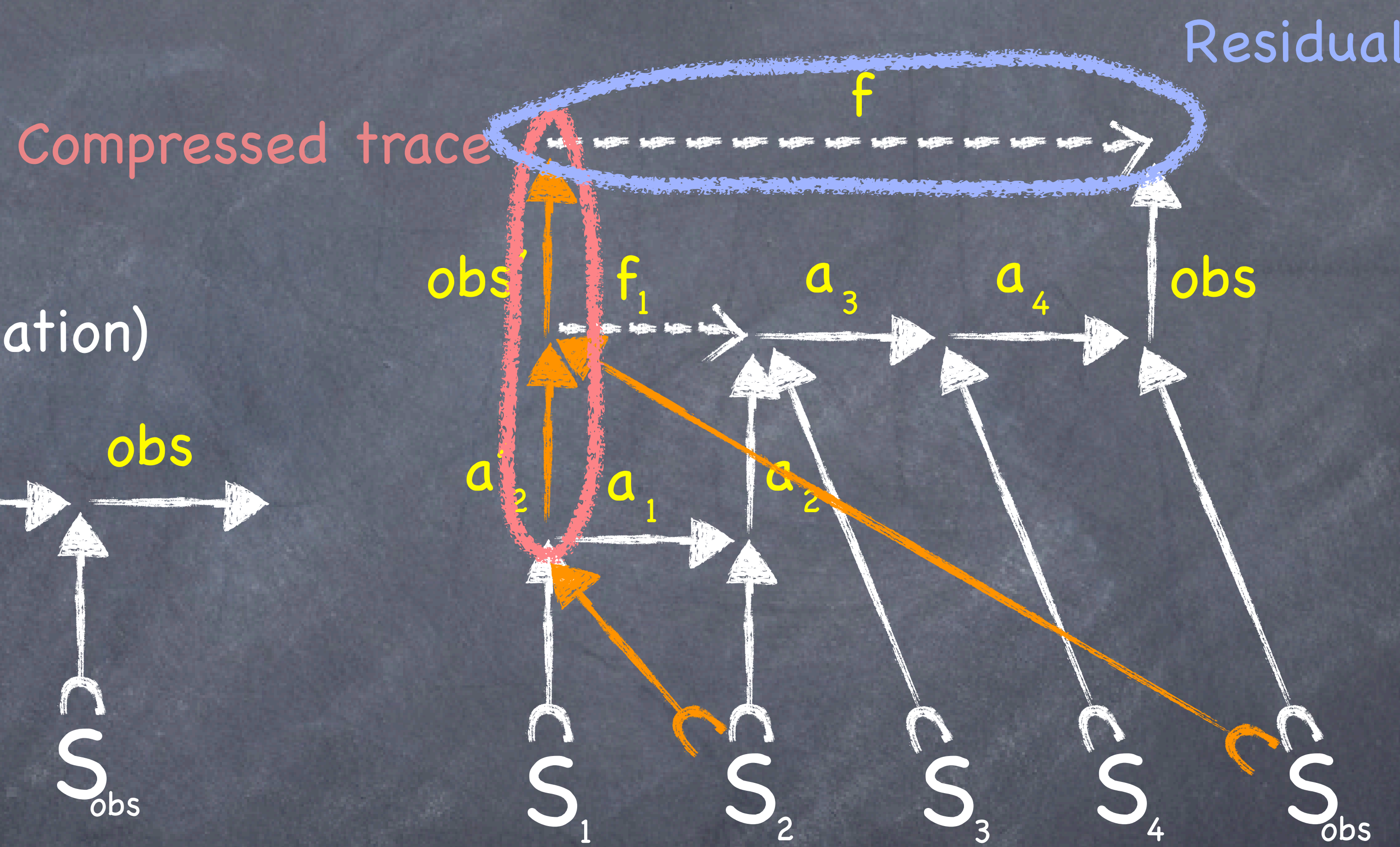


Assume the above partition of arrows is given by an oracle

Simple compression

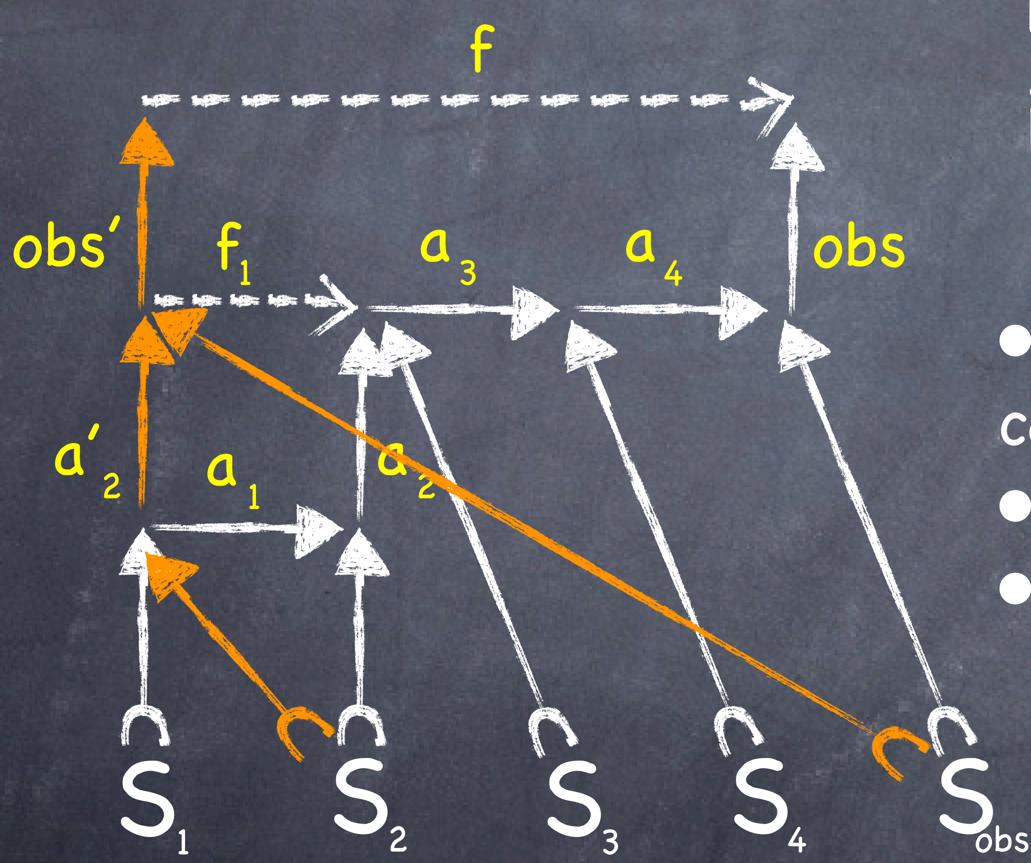
A trace (obtained by simulation)





Assume the above partition of arrows is given by an oracle

Generalized commutation

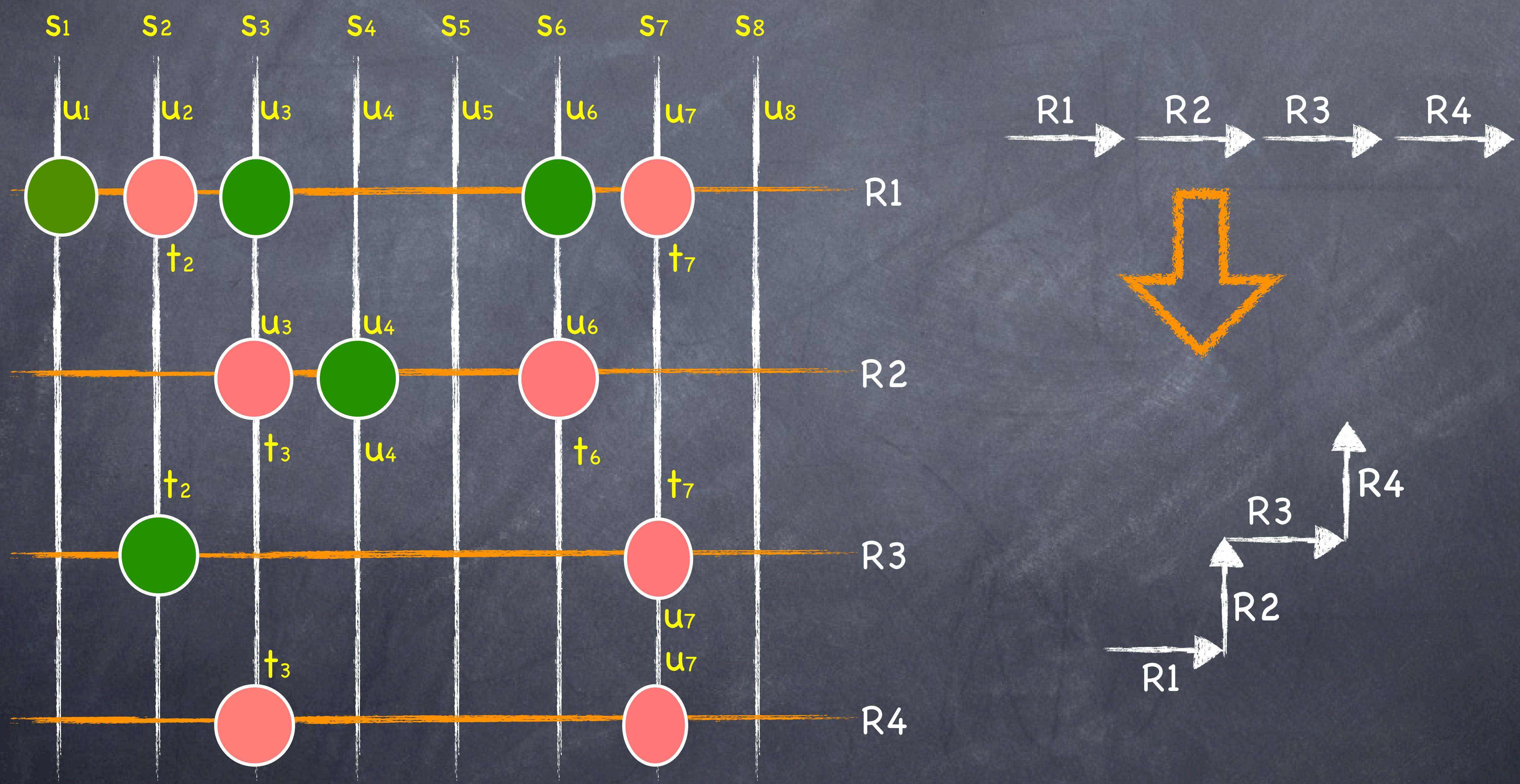


Looking at f gives us some information about compression:

- Concurrency (f is the sequence of all applied rules concurrent to the observable)
- Loop elimination (f is a subset of concurrent rules)
- Helices elimination (a general -action- map)

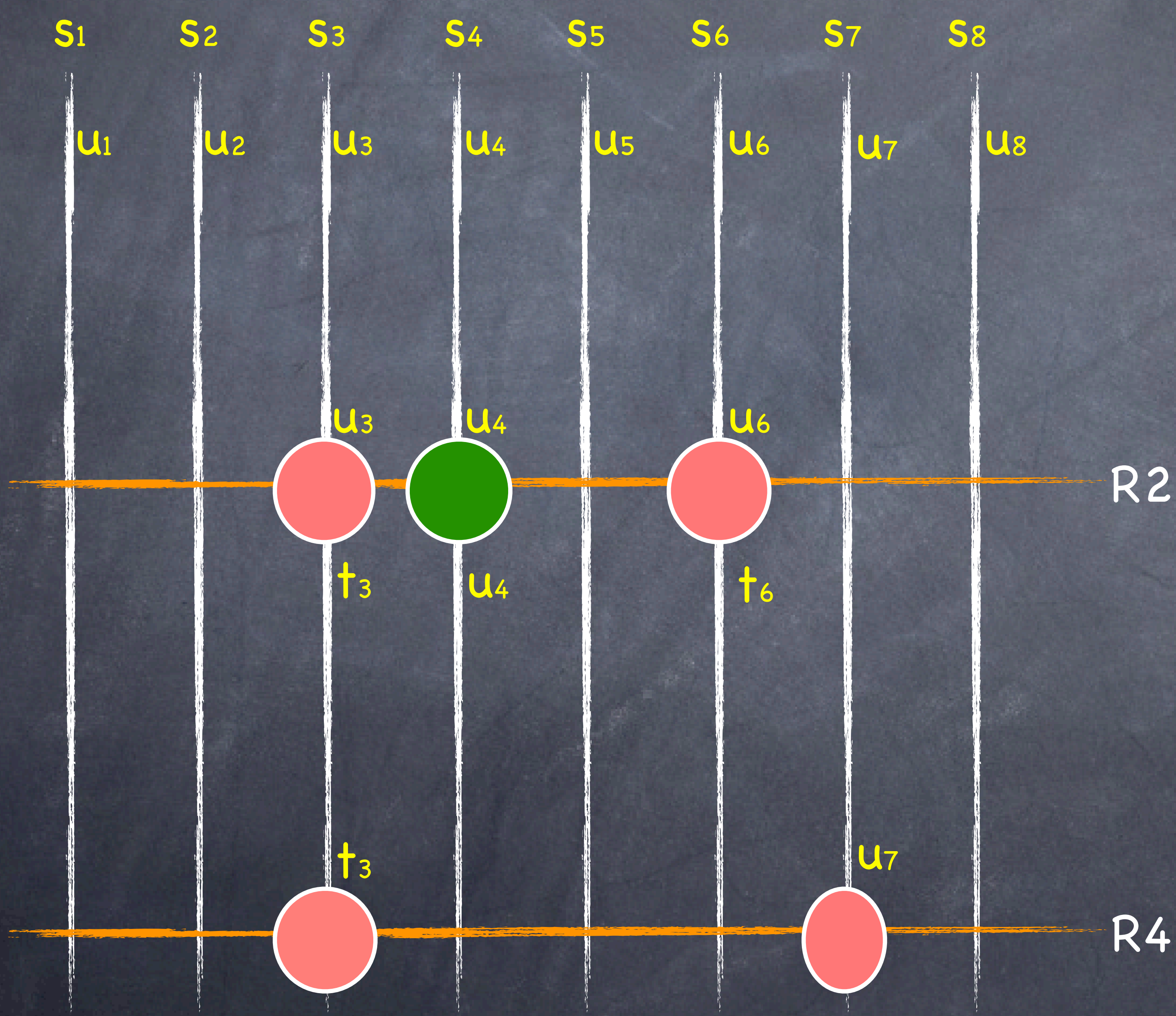
Say a trace is hyper causal if it cannot be compressed further

The oracle

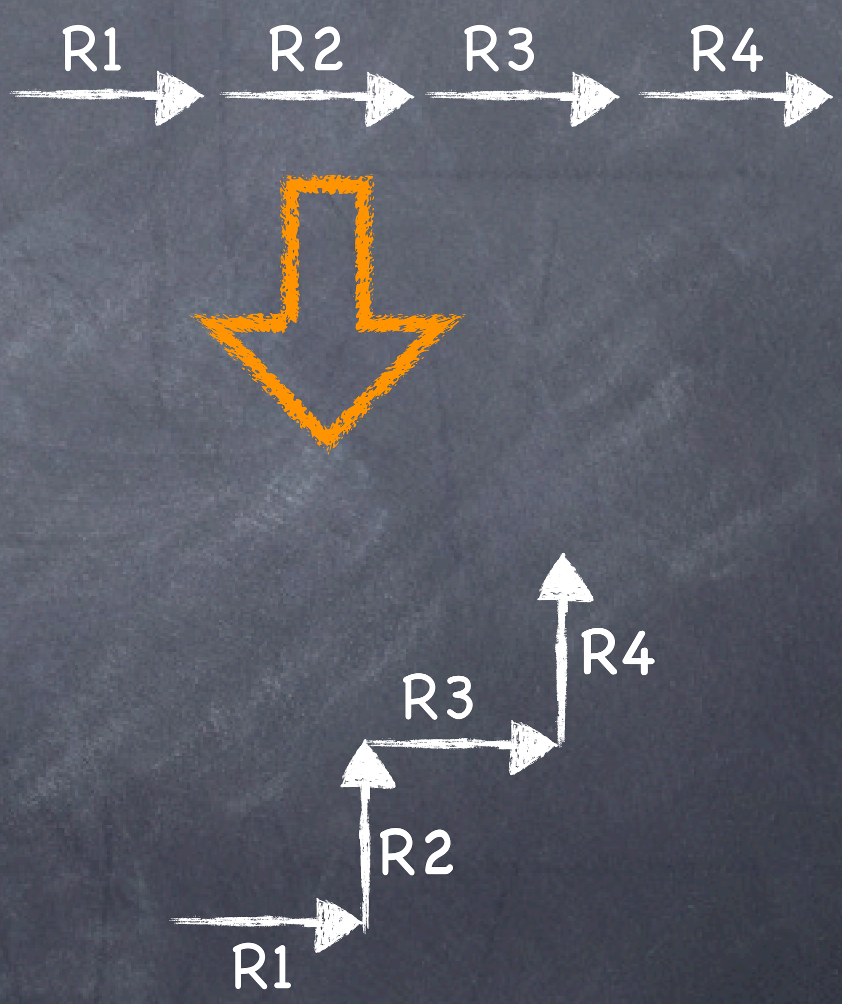


s1 s2 s3 s4 s5 s6 s7 s8
u1 u2 u3 u4 u5 u6 u7 u8
t2 t7
u3 u4 u6
t3 u4 t6
t2 t7
u7
t3 u7
R1
R2
R3
R4
R1 R2 R3 R4
R1 R2 R3 R4

The oracle

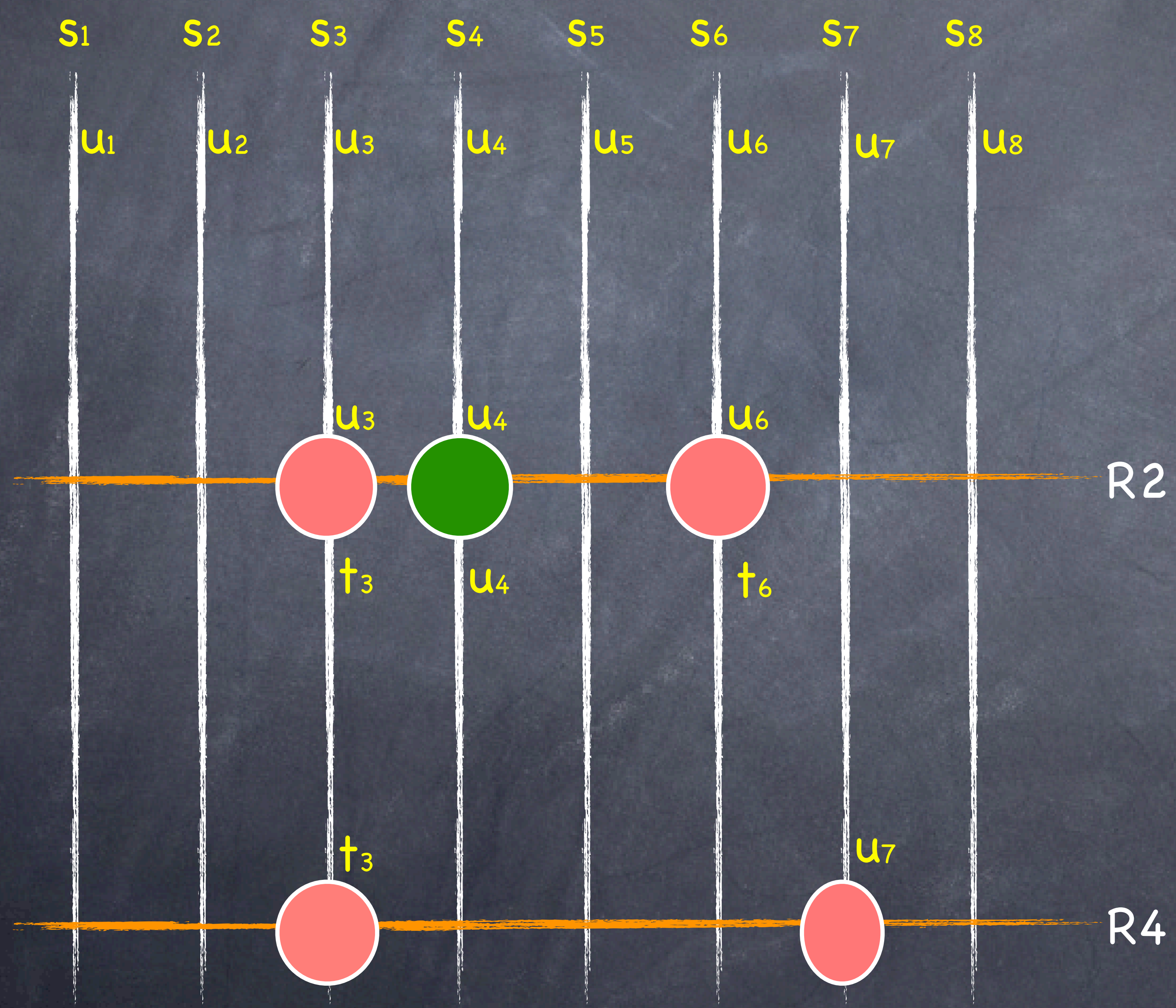


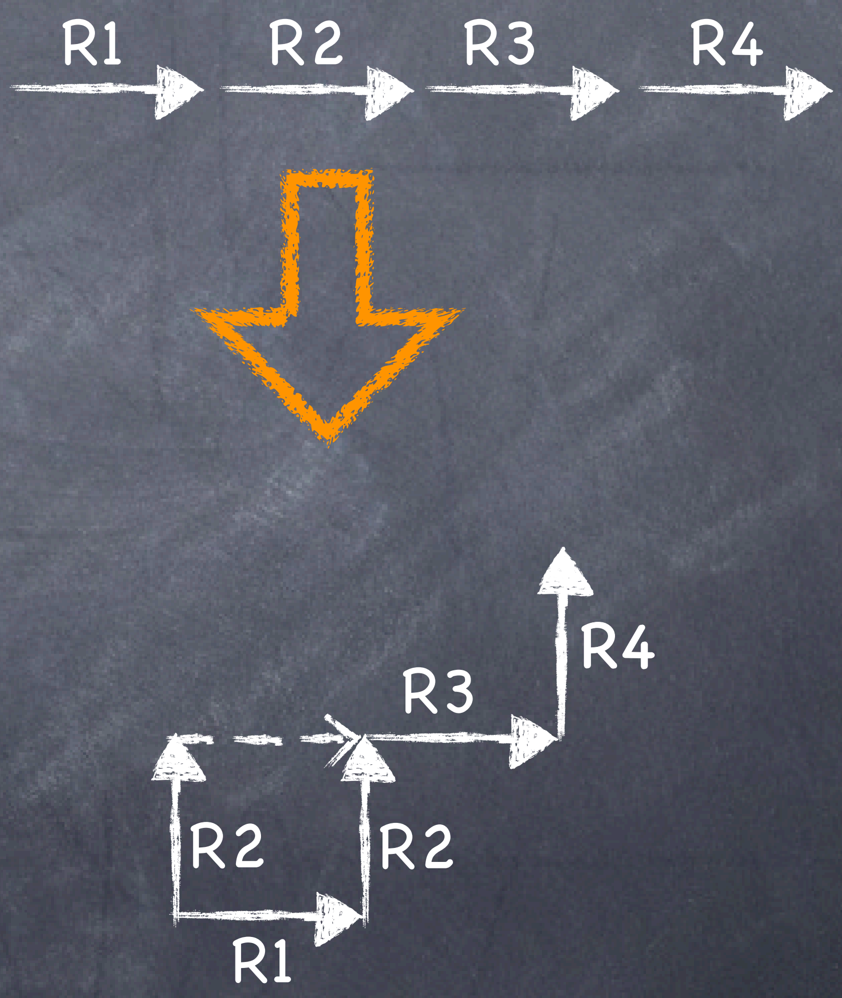
s1 s2 s3 s4 s5 s6 s7 s8
u1 u2 u3 u4 u5 u6 u7 u8
u3 u4 u6
R2
t3 u4 t6
t3 u7
R4



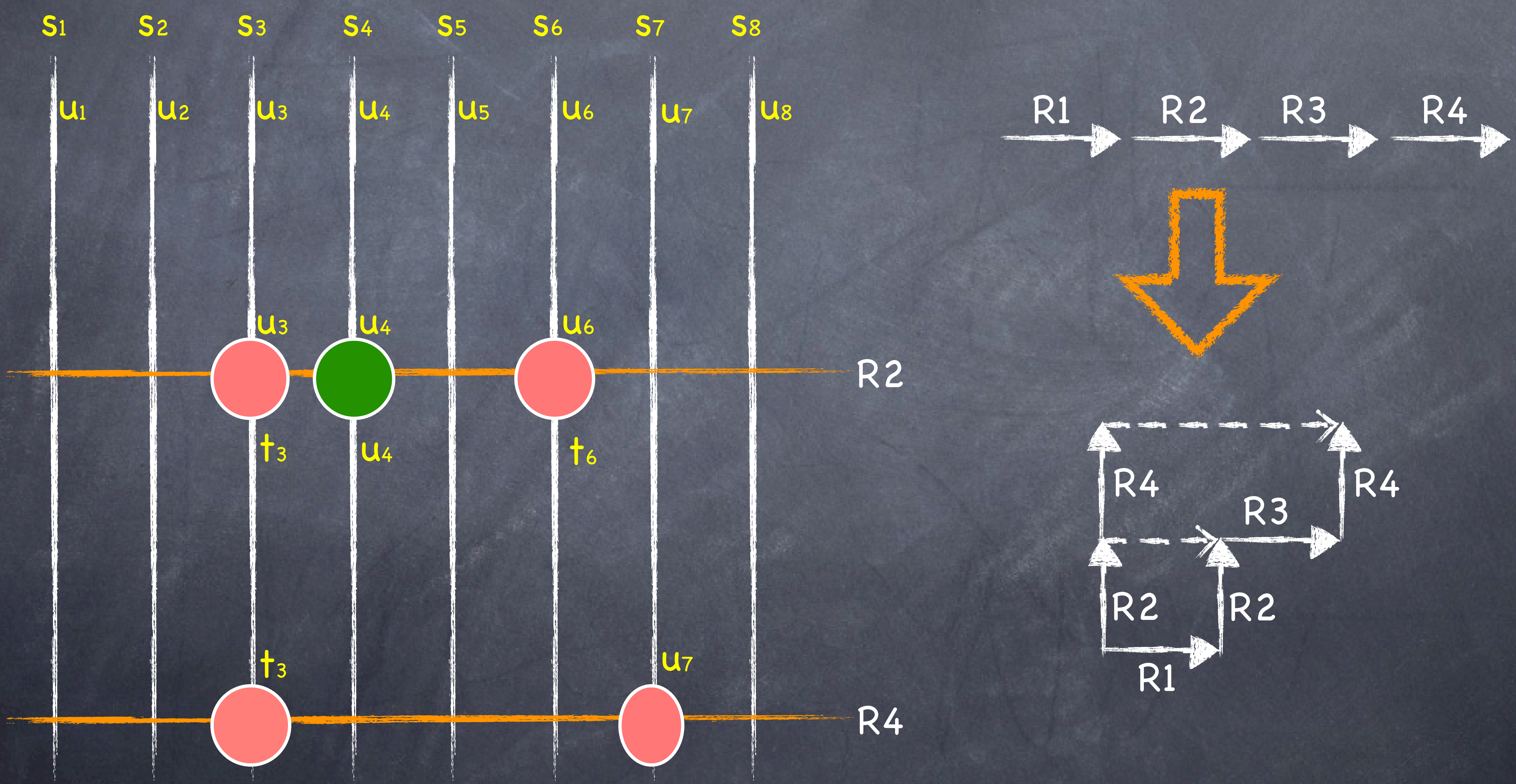
R1 R2 R3 R4
R4
R3
R2
R1

The oracle


s1 s2 s3 s4 s5 s6 s7 s8
u1 u2 u3 u4 u5 u6 u7 u8
u3 u4 u6
R2
t3 u4 t6
t3 u7
R4


R1 R2 R3 R4
R4
R3
R2 R2
R1

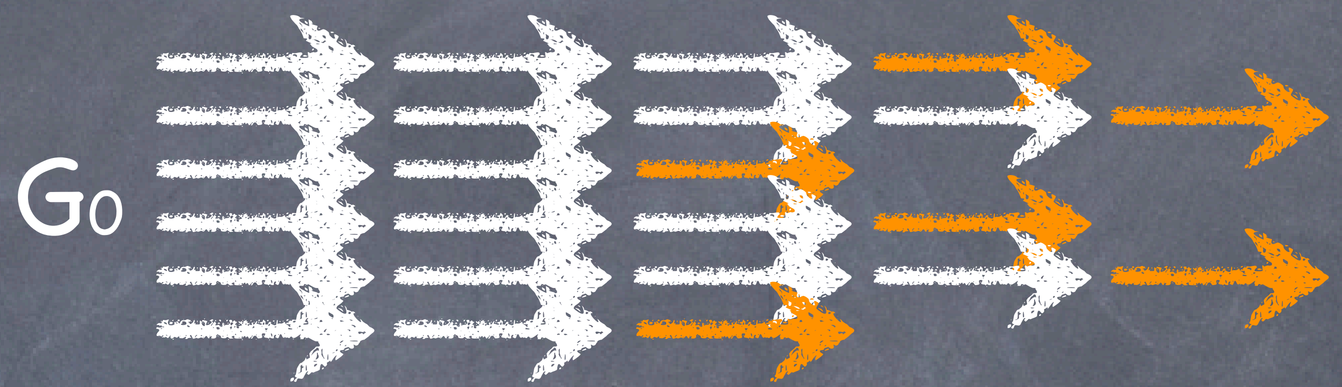
The oracle



s1
s2
s3
s4
s5
s6
s7
s8
u1
u2
u3
u4
u5
u6
u7
u8
u3
u4
u6
R2
t3
u4
t6
t3
u7
R4
R1
R2
R3
R4
R4
R4
R3
R2
R2
R1

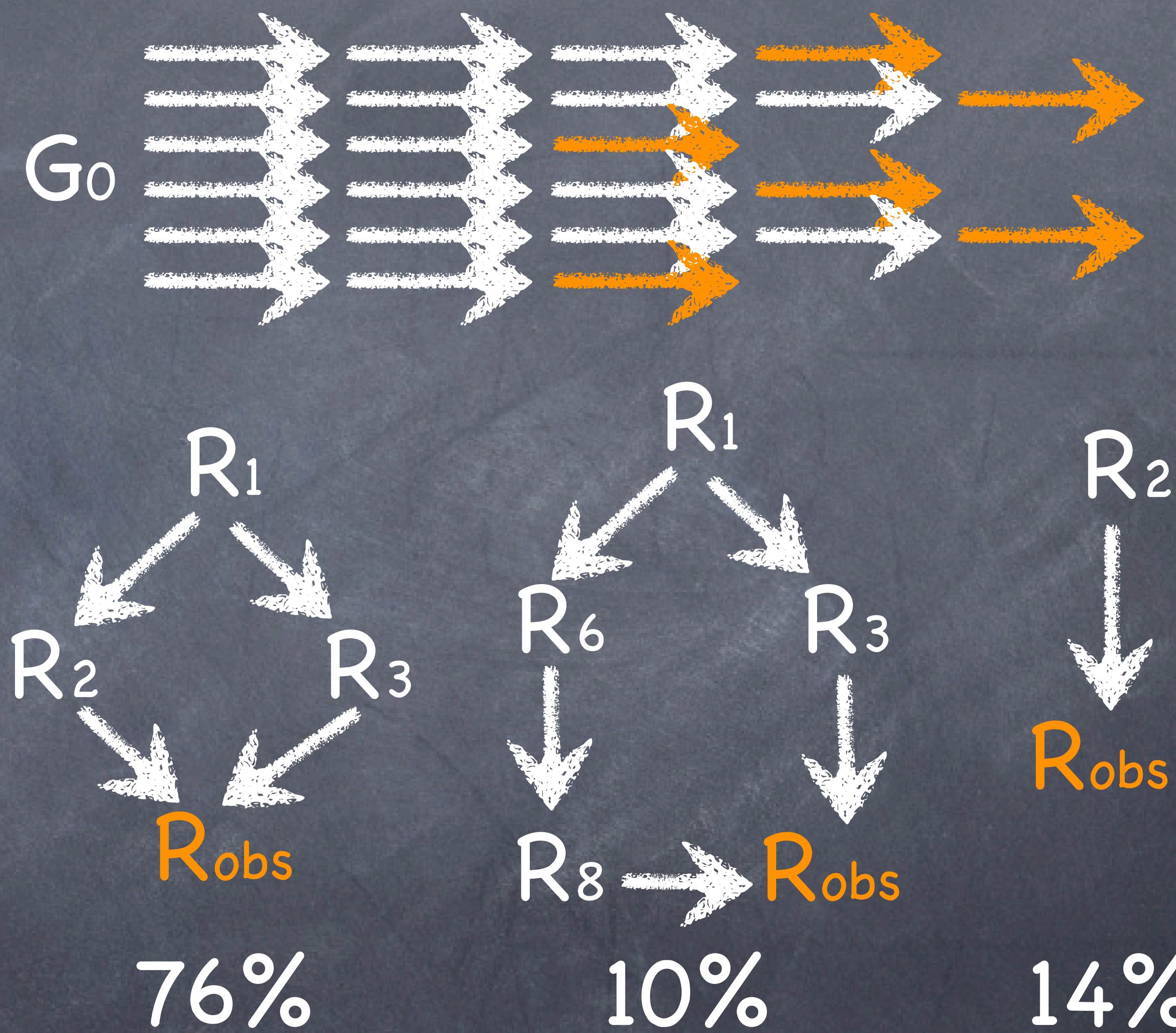
Workflow

- Run n simulations



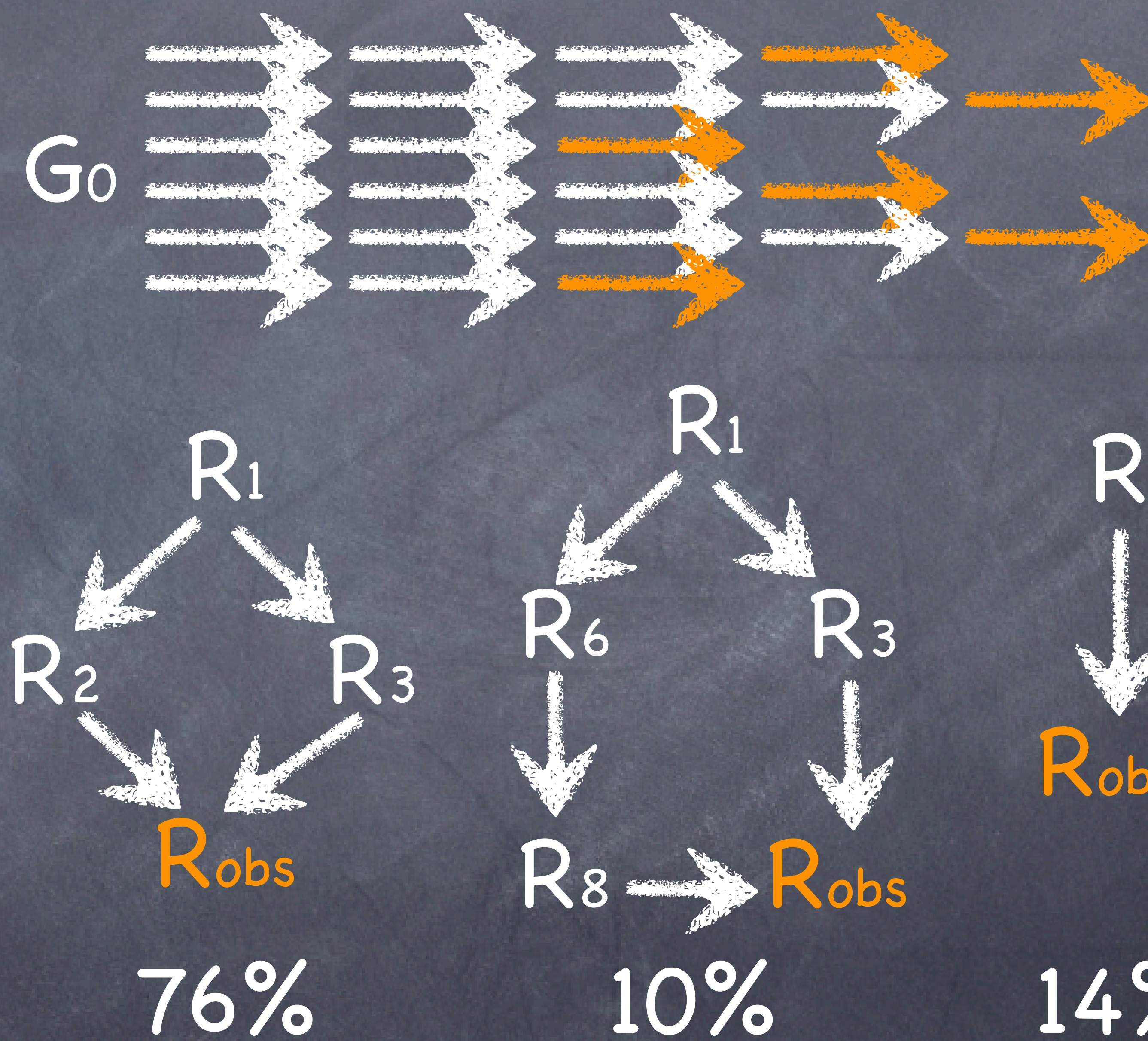
Workflow

- Run n simulations
- Simple causality analysis



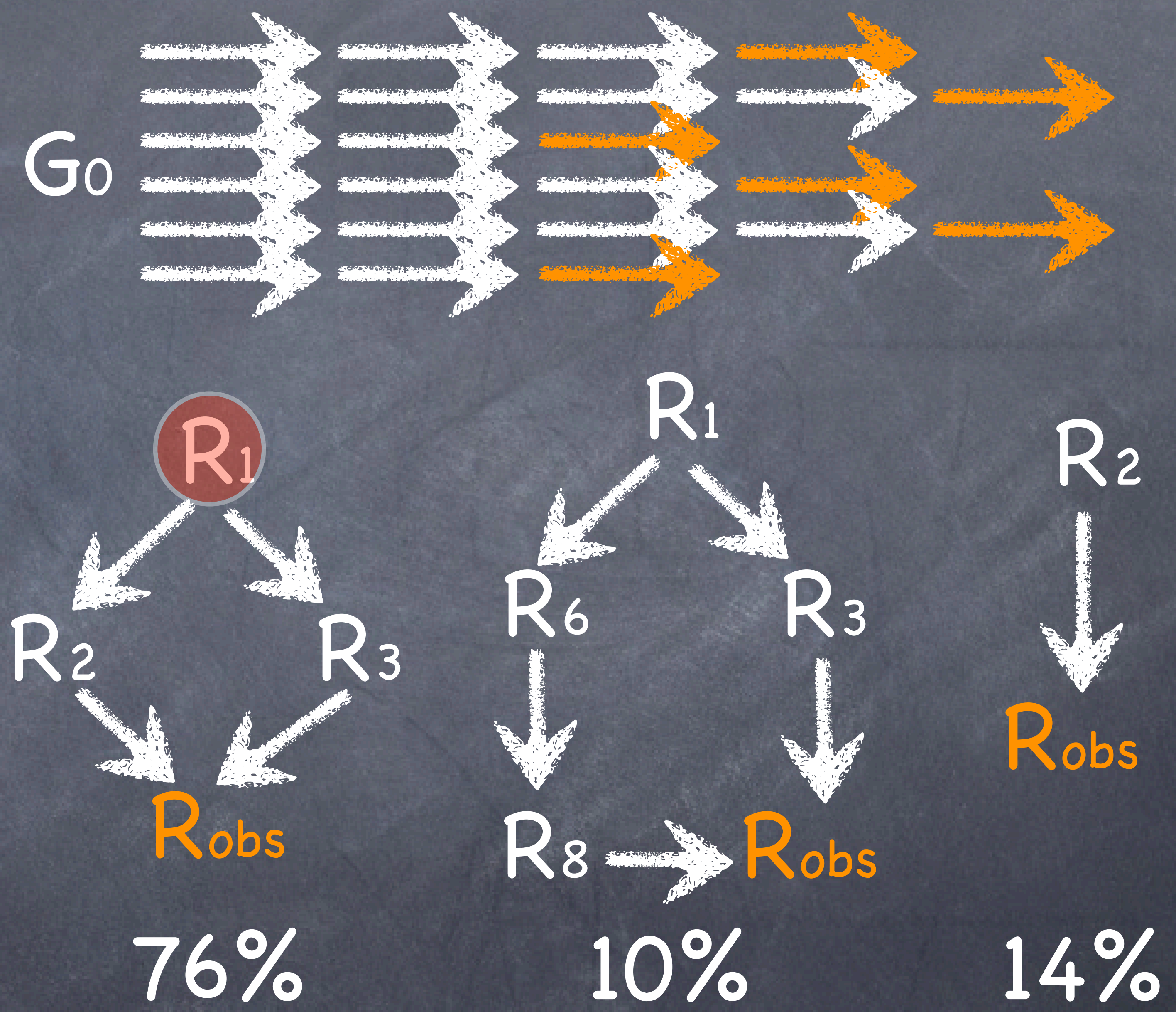
Workflow

- Run n simulations
- Simple causality analysis
- Knock-out events to reduce to minimal configurations



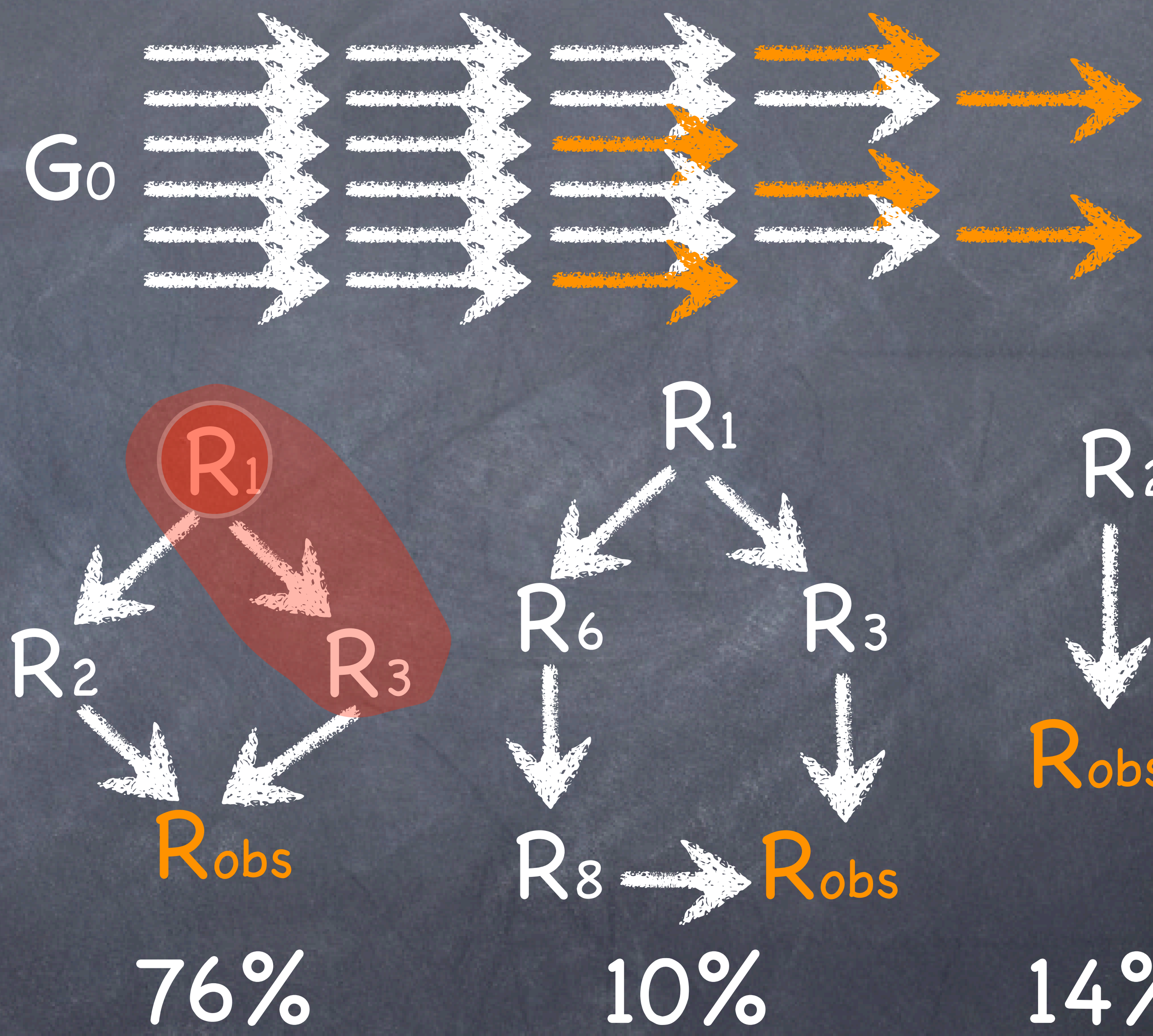
Workflow

- Run n simulations
- Simple causality analysis
- Knock-out events to reduce to minimal configurations



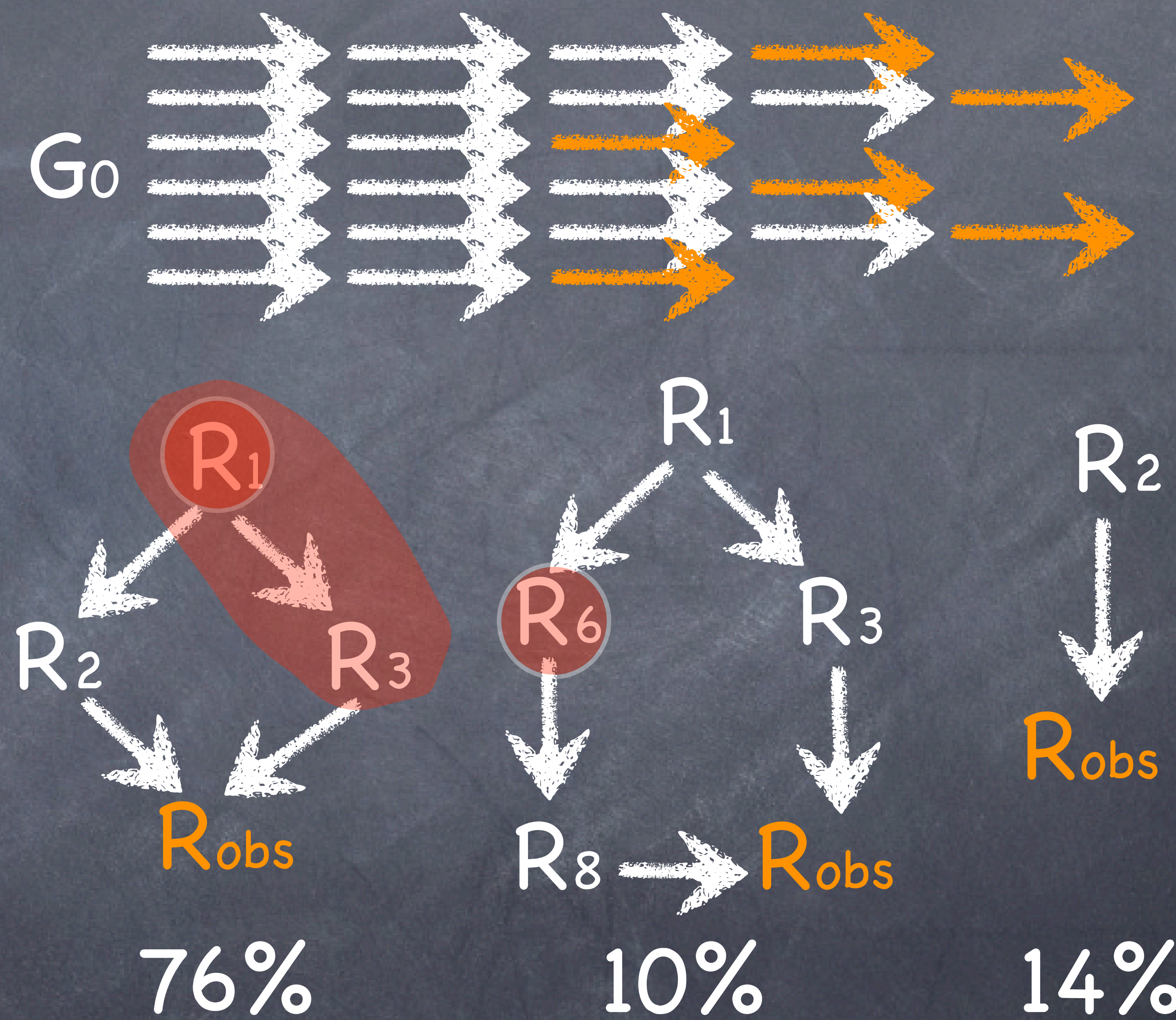
Workflow

- Run n simulations
- Simple causality analysis
- Knock-out events to reduce to minimal configurations



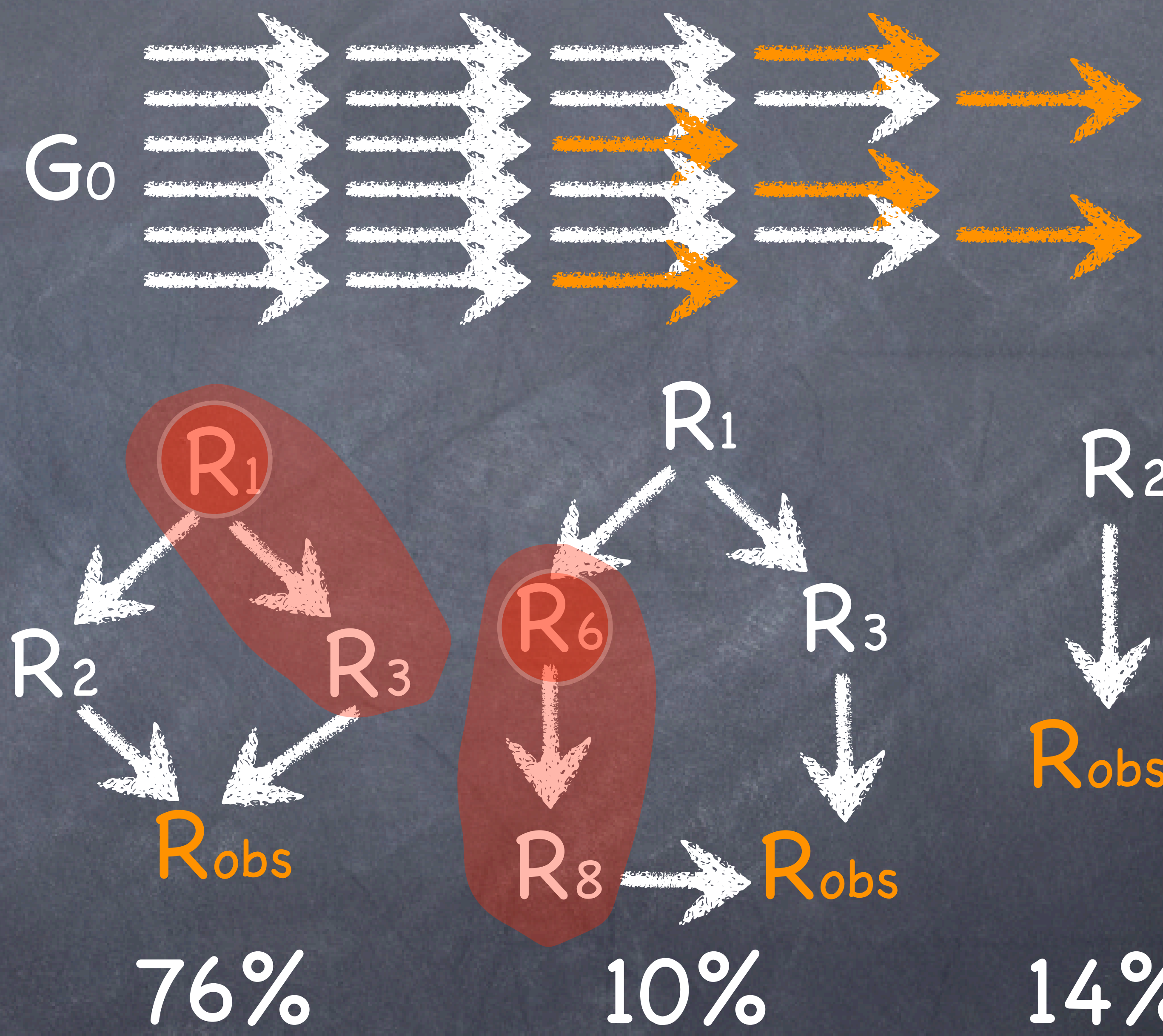
Workflow

- Run n simulations
- Simple causality analysis
- Knock-out events to reduce to minimal configurations



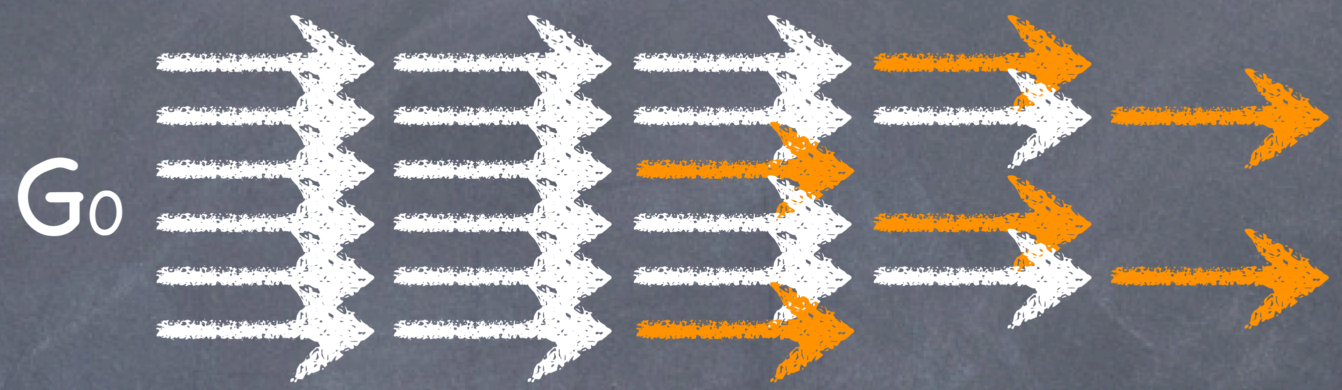
Workflow

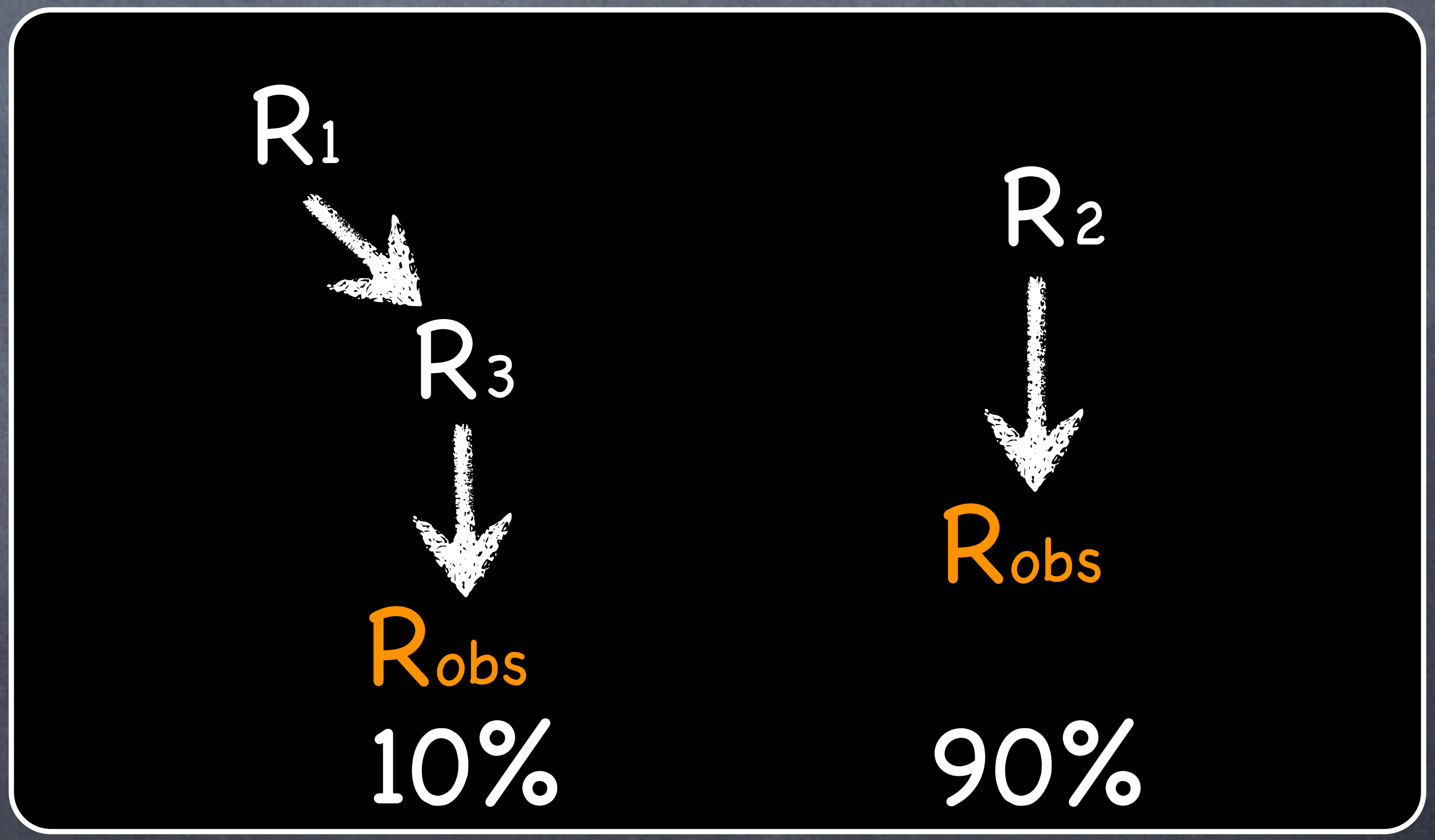
- Run n simulations
- Simple causality analysis
- Knock-out events to reduce to minimal configurations



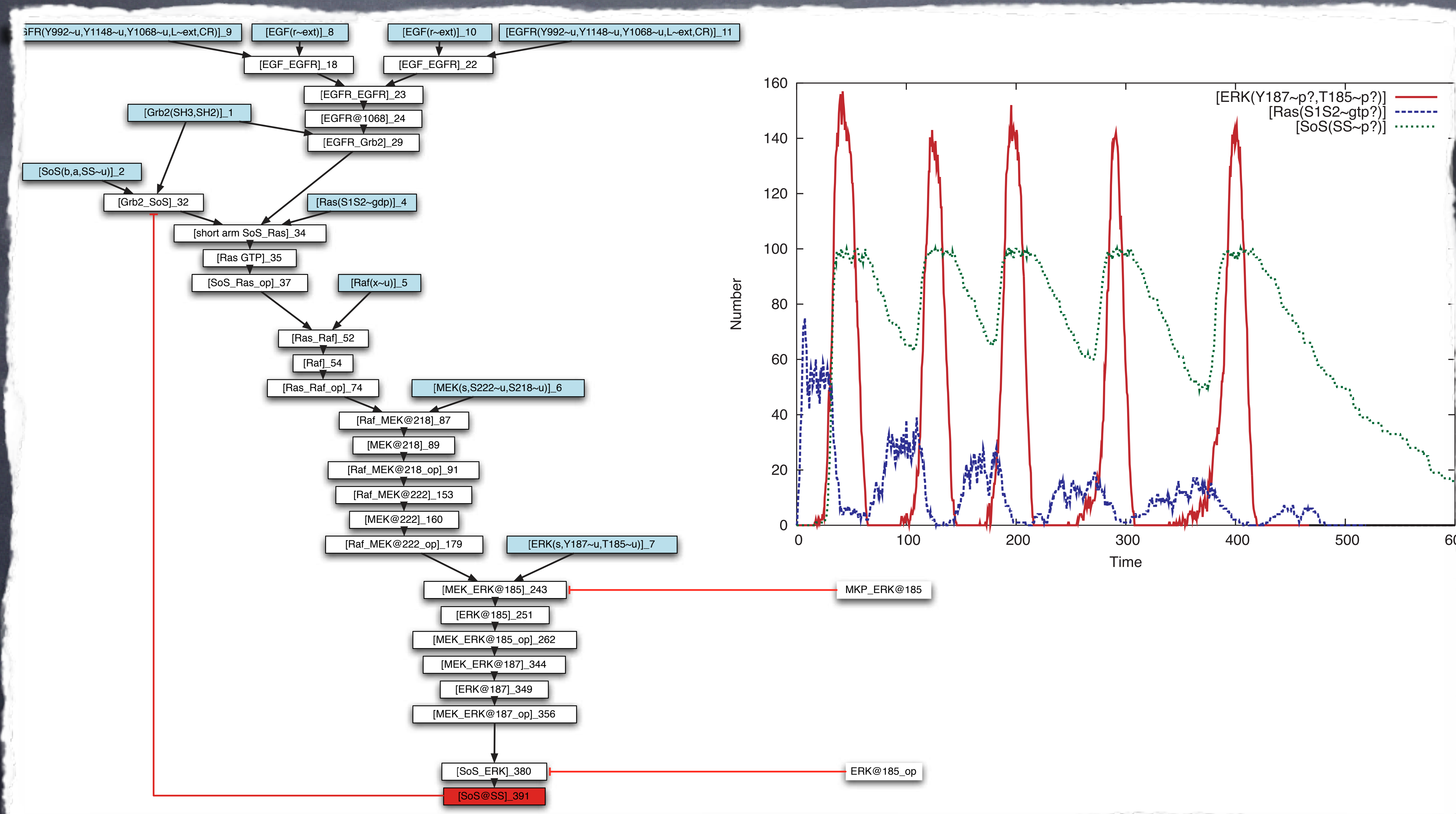
Workflow

- Run n simulations
- Simple causality analysis
- Knock-out events to reduce to minimal configurations





Overview



GFR(Y992~u,Y1148~u,Y1068~u,L~ext,CR)]_9
[EGF(r~ext)]_8
[EGF(r~ext)]_10
[EGFR(Y992~u,Y1148~u,Y1068~u,L~ext,CR)]_11
[EGF_EGFR]_18
[EGF_EGFR]_22
[EGFR_EGFR]_23
[Grb2(SH3,SH2)]_1
[EGFR@1068]_24
[EGFR_Grb2]_29
[SoS(b,a,SS~u)]_2
[Grb2_SoS]_32
[Ras(S1S2~gdp)]_4
[short arm SoS_Ras]_34
[Ras GTP]_35
[SoS_Ras_op]_37
[Raf(x~u)]_5
[Ras_Raf]_52
[Raf]_54
[Ras_Raf_op]_74
[MEK(s,S222~u,S218~u)]_6
[Raf_MEK@218]_87
[MEK@218]_89
[Raf_MEK@218_op]_91
[Raf_MEK@222]_153
[MEK@222]_160
[Raf_MEK@222_op]_179
[ERK(s,Y187~u,T185~u)]_7
[MEK_ERK@185]_243
MKP_ERK@185
[ERK@185]_251
[MEK_ERK@185_op]_262
[MEK_ERK@187]_344
[ERK@187]_349
[MEK_ERK@187_op]_356
[SoS_ERK]_380
ERK@185_op
[SoS@SS]_391
[ERK(Y187~p?,T185~p?)]
[Ras(S1S2~gtp?)]
[SoS(SS~p?)]
Number
Time
0
20
40
60
80
100
120
140
160
0
100
200
300
400
500
60

Temporary conclusions

- While exporting standard causality analysis, one discovers theory needs extension!
- Tools already useful for debugging purpose
- It remains to see whether one can use them for prediction (emergent behavior)

Knock-out in CS

T1	T2	T3	T4
i++	i++	i*i	sqrt i

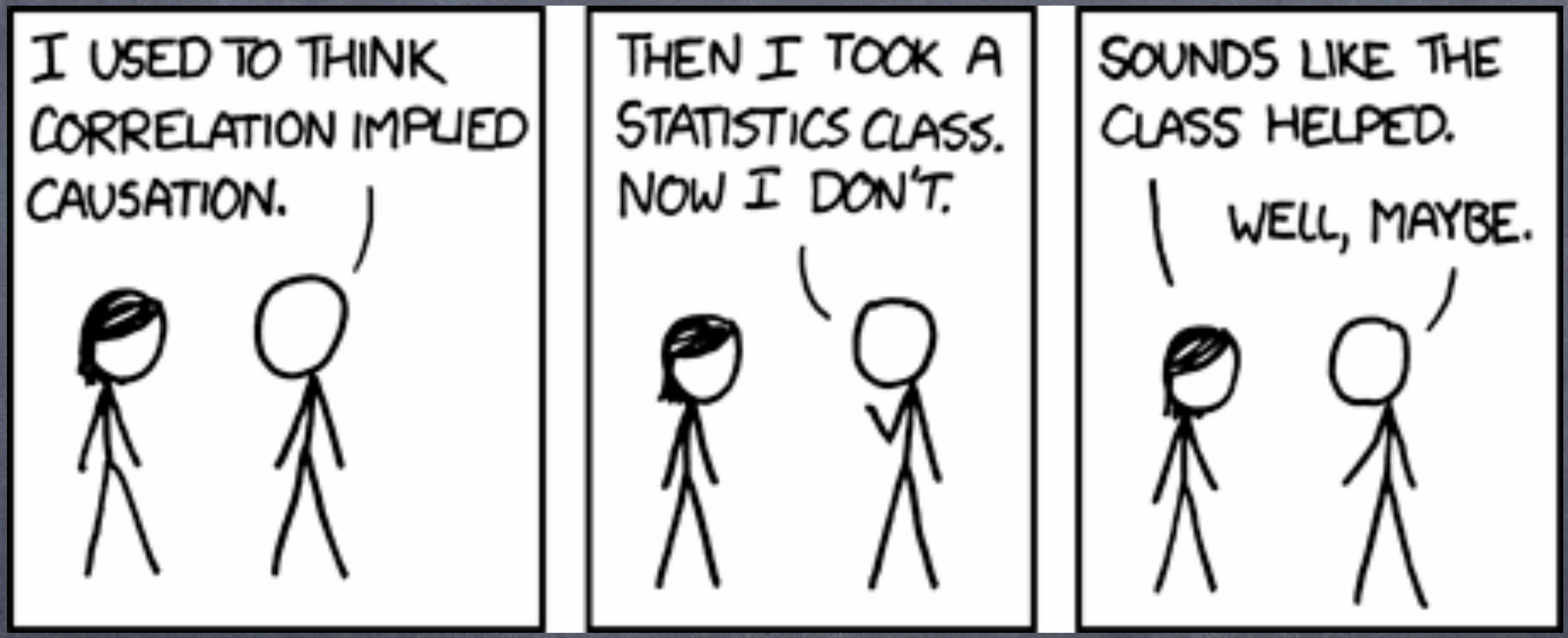
i:=0

...

i:=nan

Knock out an event and check whether obs:nan is preserved!

Thanks



http://xkcd.com/