INFORMEL
Indo-French Formal Methods Lab

UPPSALA UNIVERSITET

COMMUNICATING RECURSIVE PROGRAMS: CONTROL AND SPLIT-WIDTH

C. Aiswarya
Uppsala University, Sweden

Joint work with

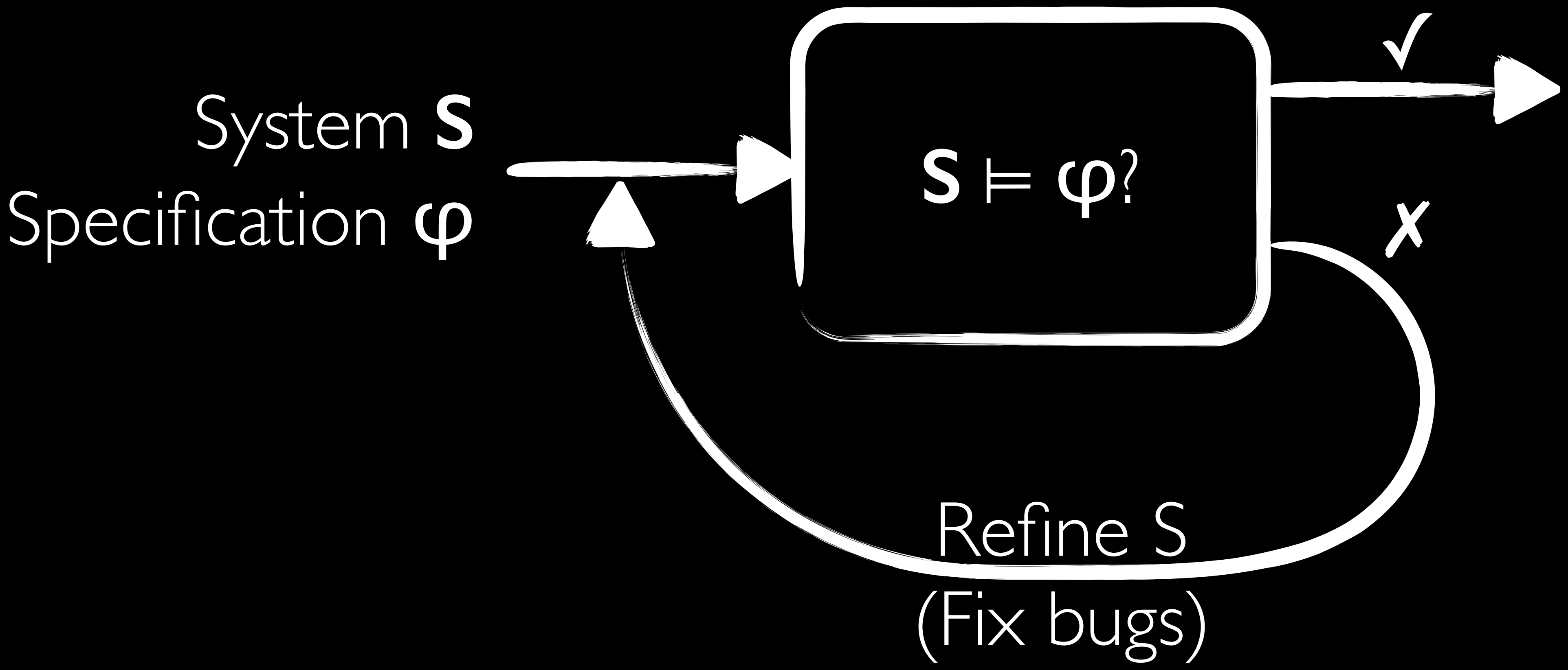
Paul Gastin
LSV, ENS Cachan, France

K. Narayan Kumar
Chennai Mathematical Institute, India

ACTS
09/02/2015, Chennai

VERIFICATION

Model Checking



System **S**

Specification **φ**

S ⊨ φ?

✓

✗

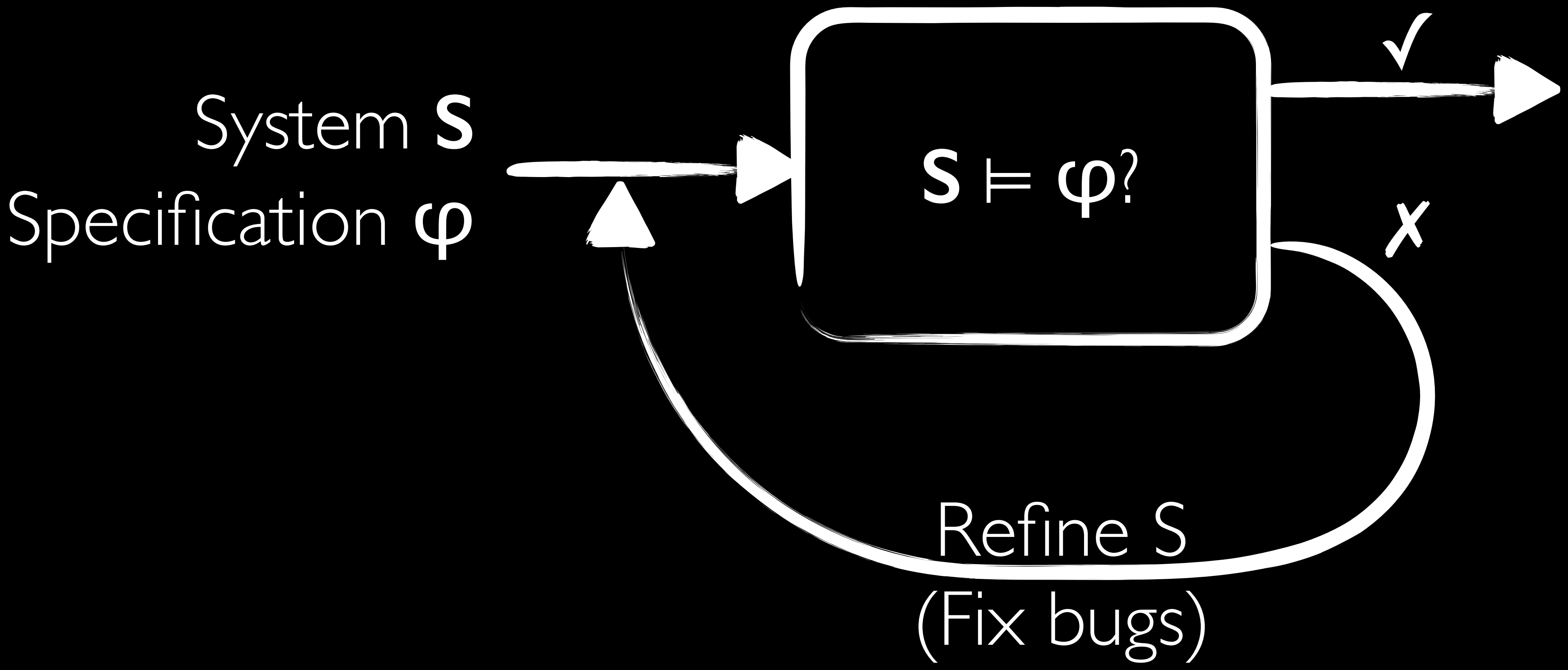
Refine S

(Fix bugs)

VERIFICATION

Model Checking

- **Undecidable in many cases**



System **S**

Specification **φ**

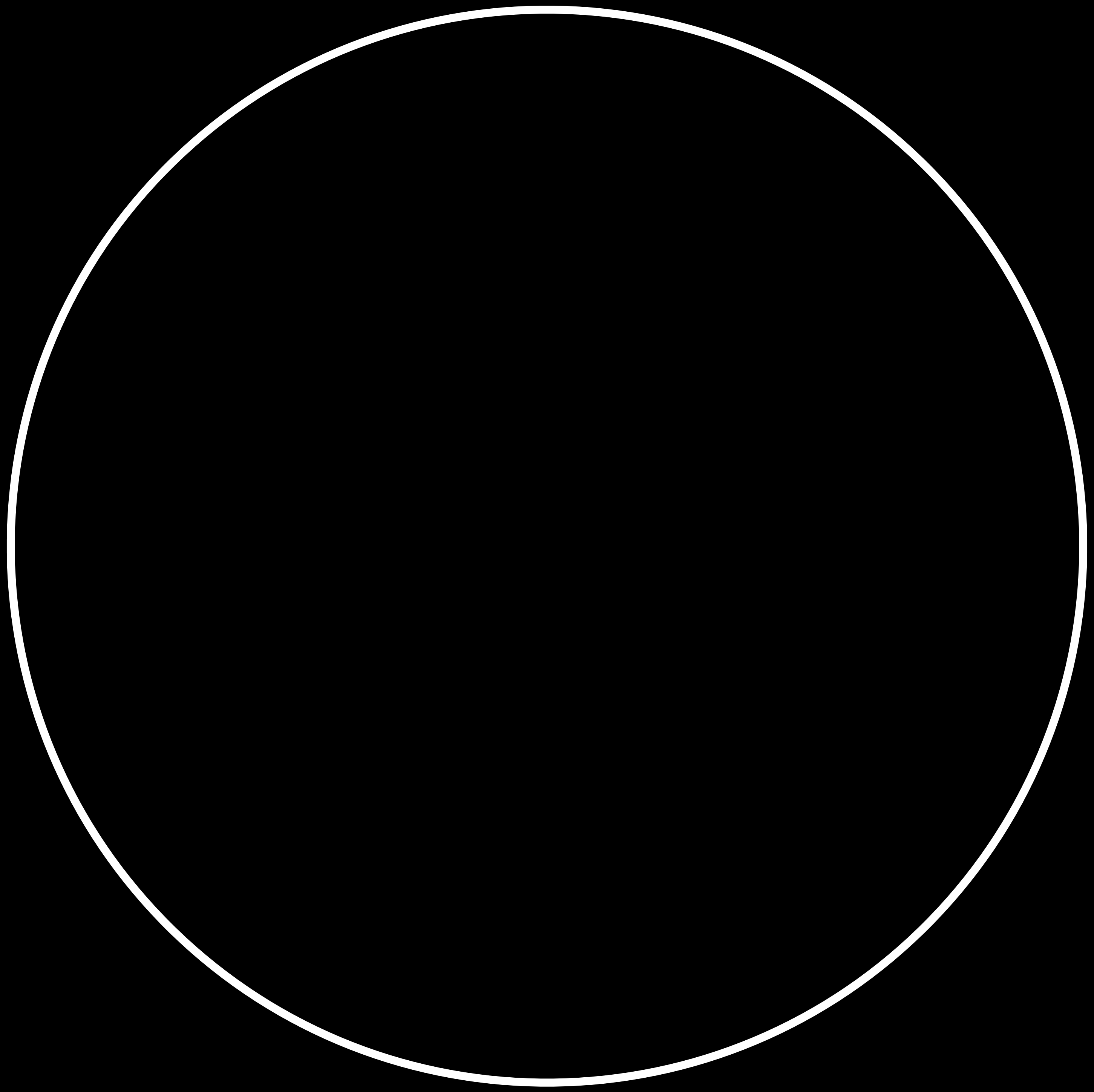
S ⊨ φ?

✓

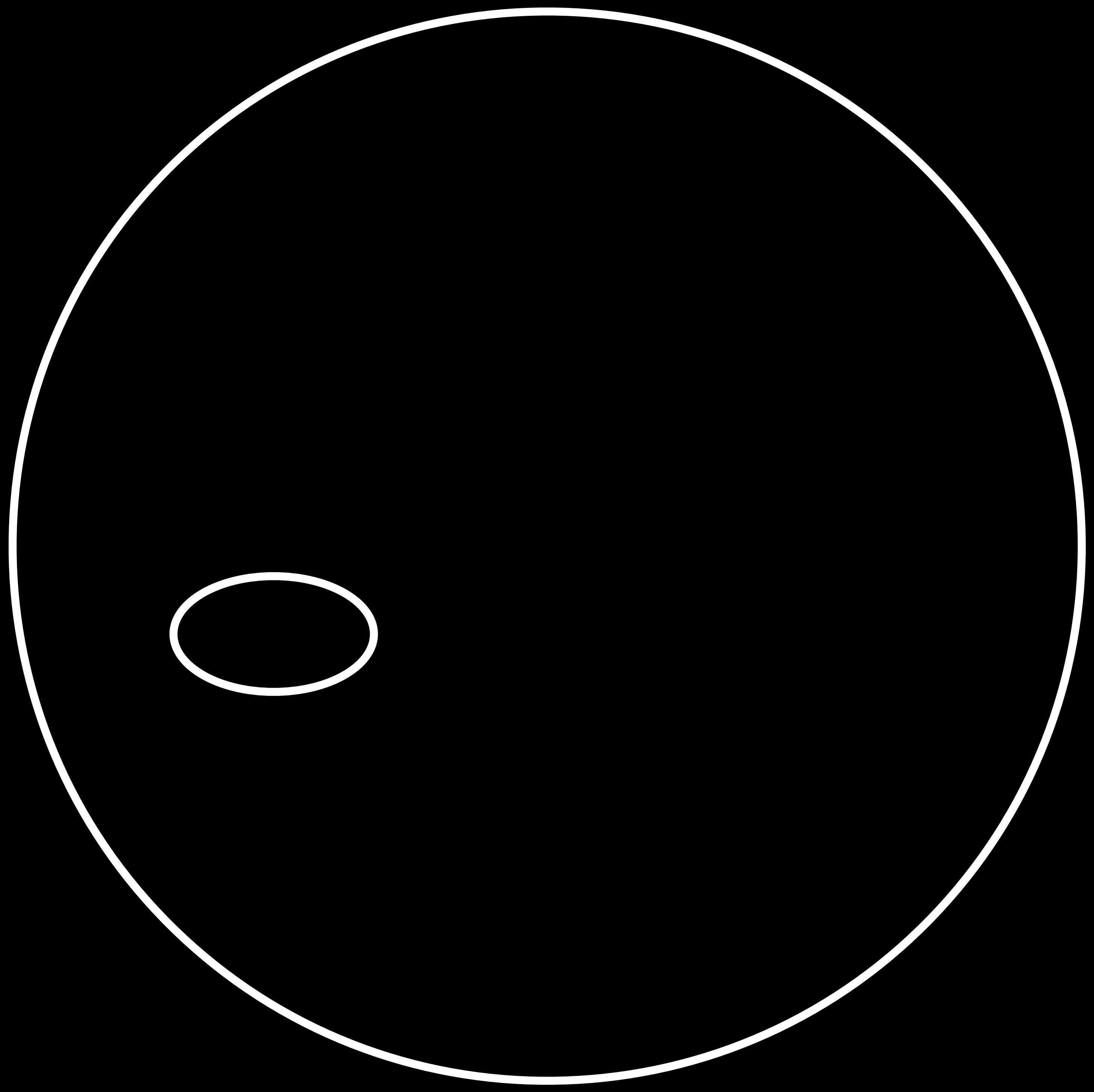
✗

Refine S
(Fix bugs)

UNDER-APPROXIMATE VERIFICATION

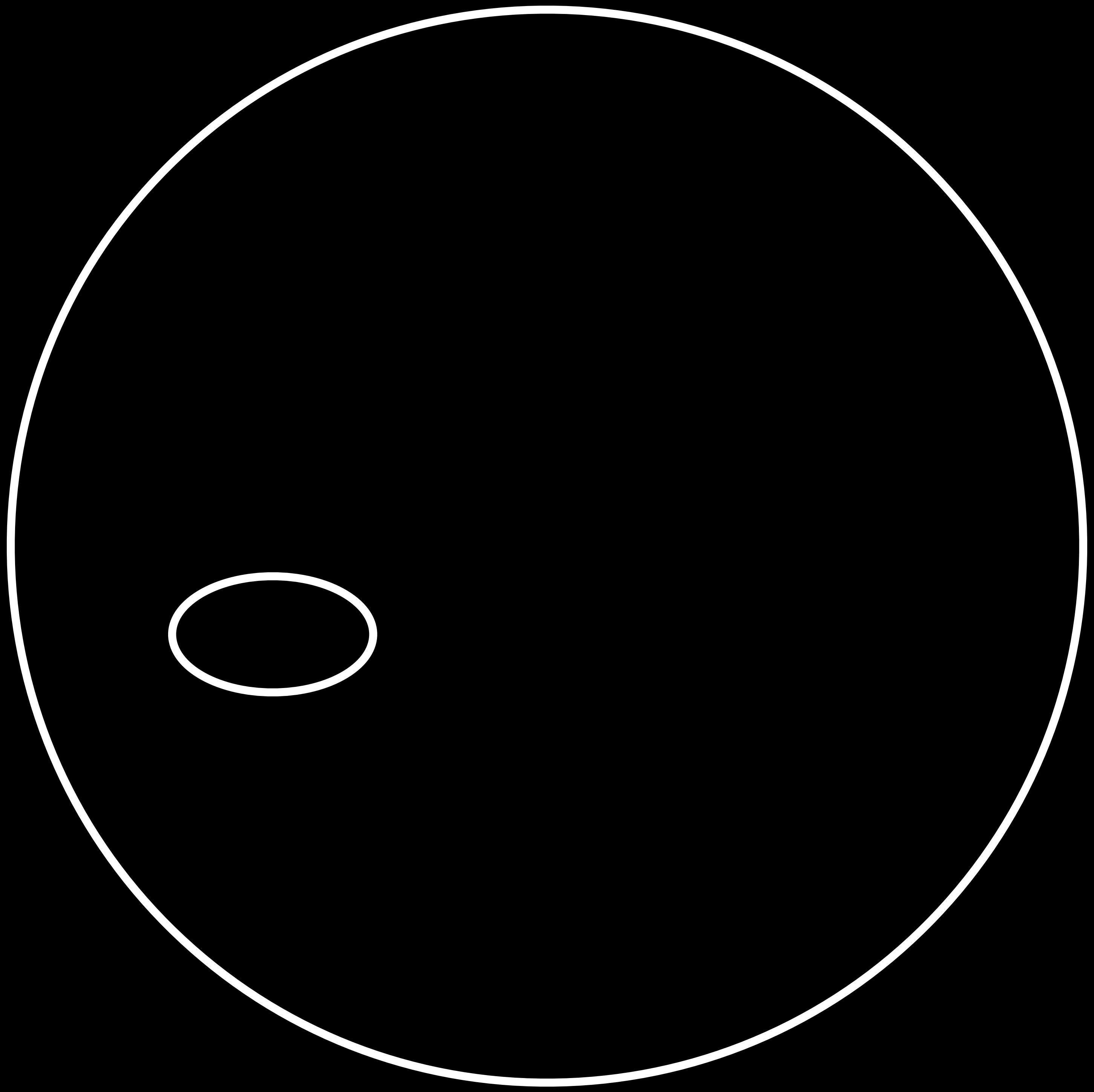


UNDER-APPROXIMATE VERIFICATION



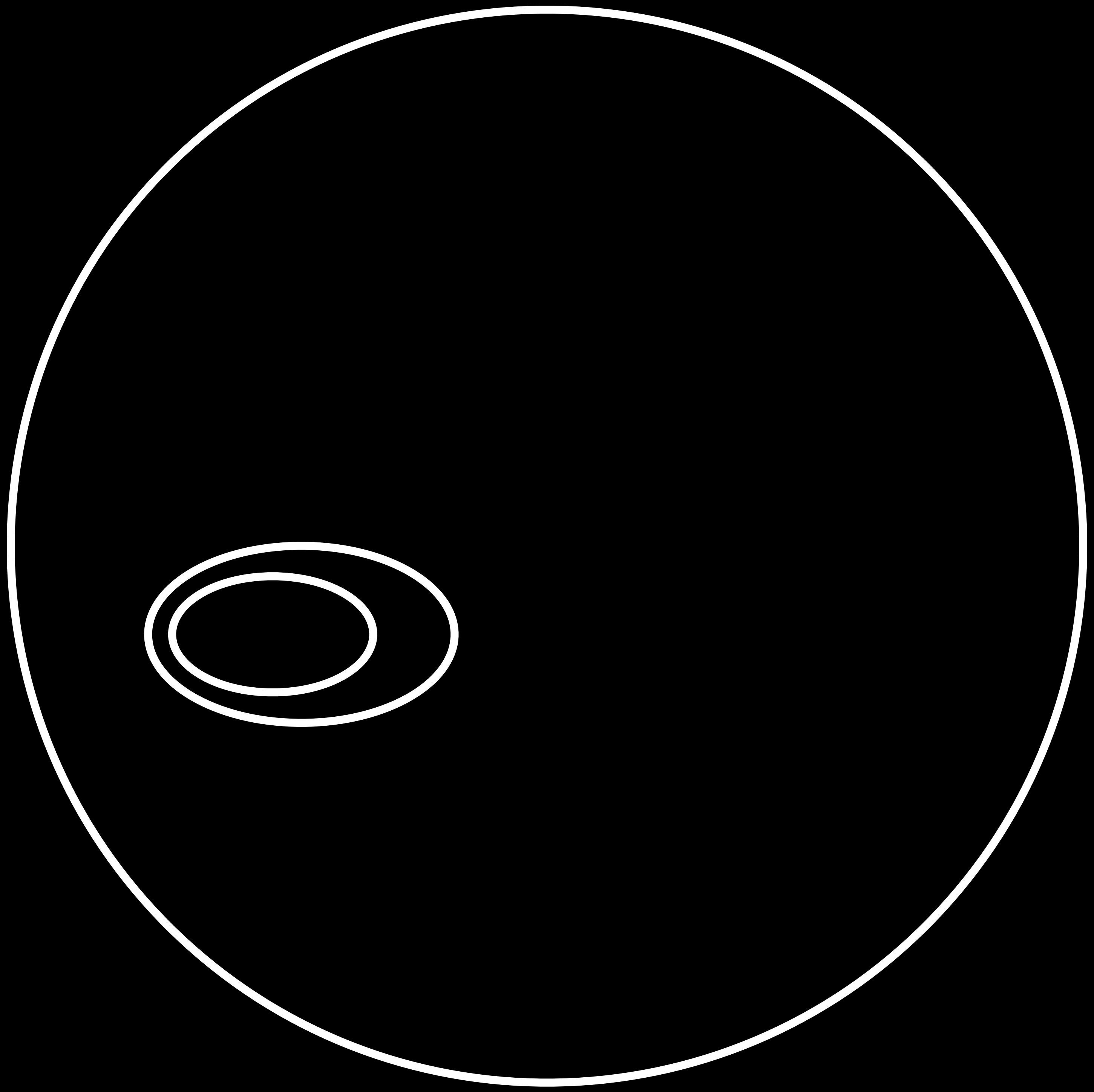
UNDER-APPROXIMATE VERIFICATION

- Parametrised



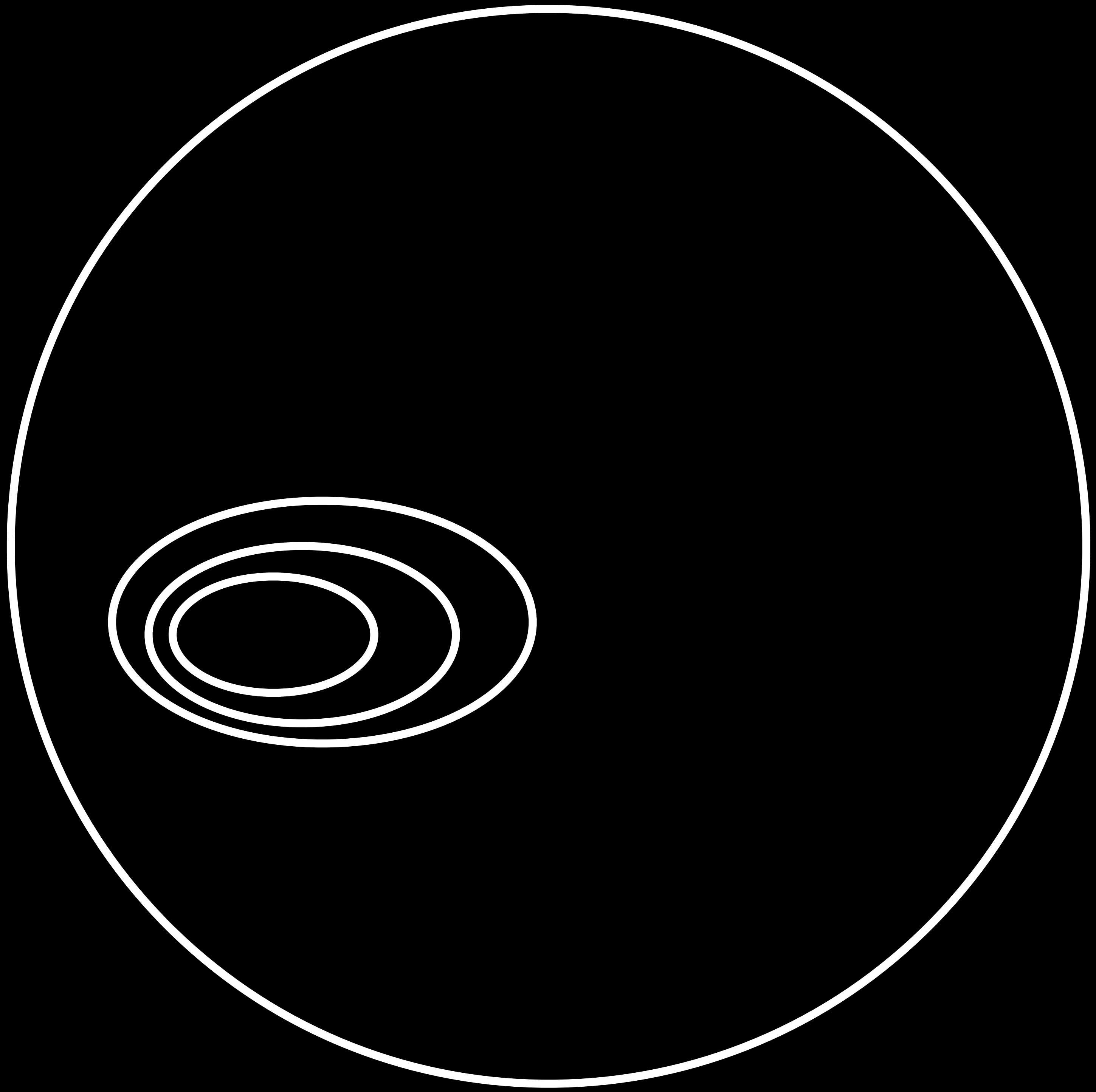
UNDER-APPROXIMATE VERIFICATION

- Parametrised



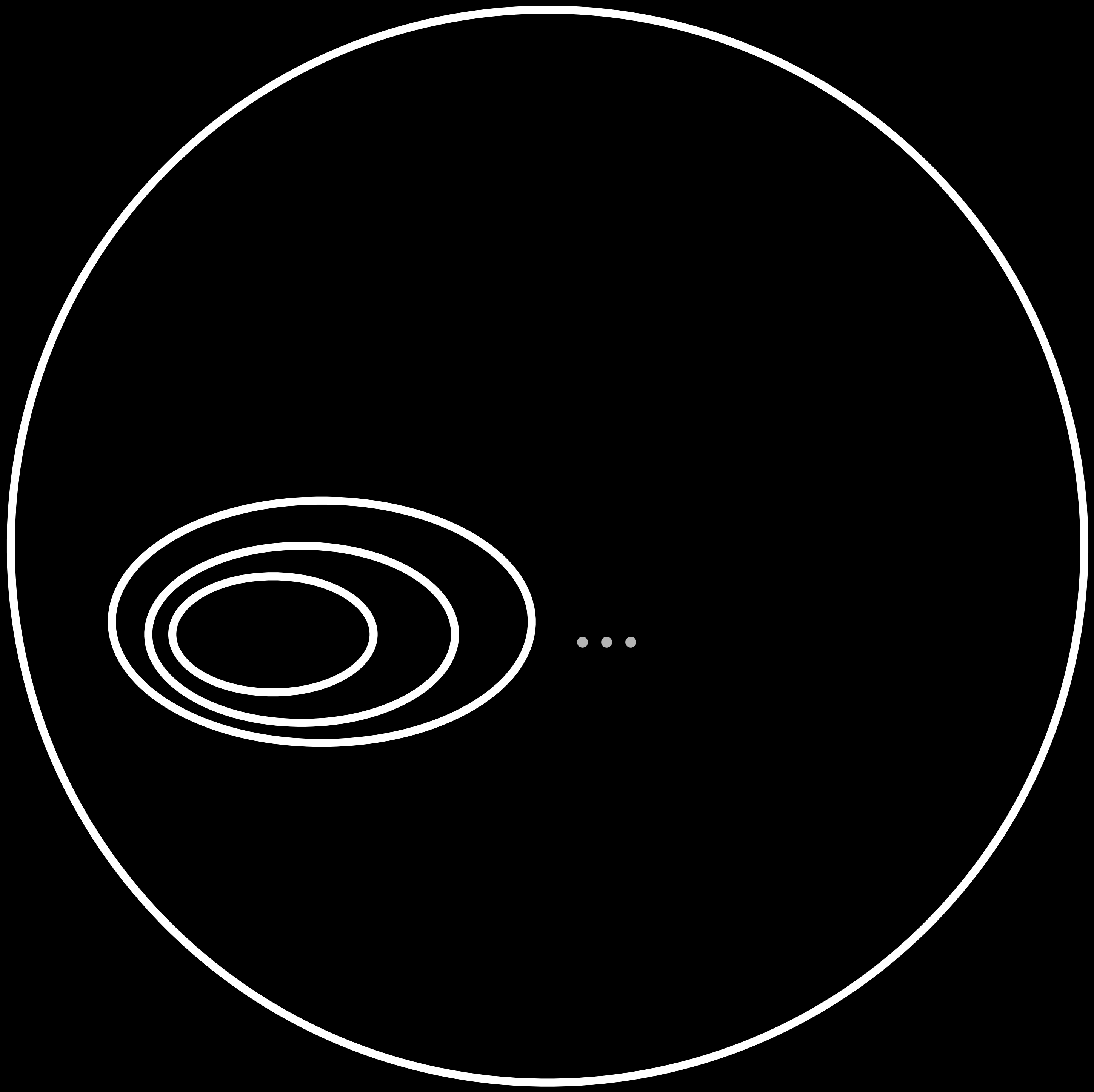
UNDER-APPROXIMATE VERIFICATION

- Parametrised



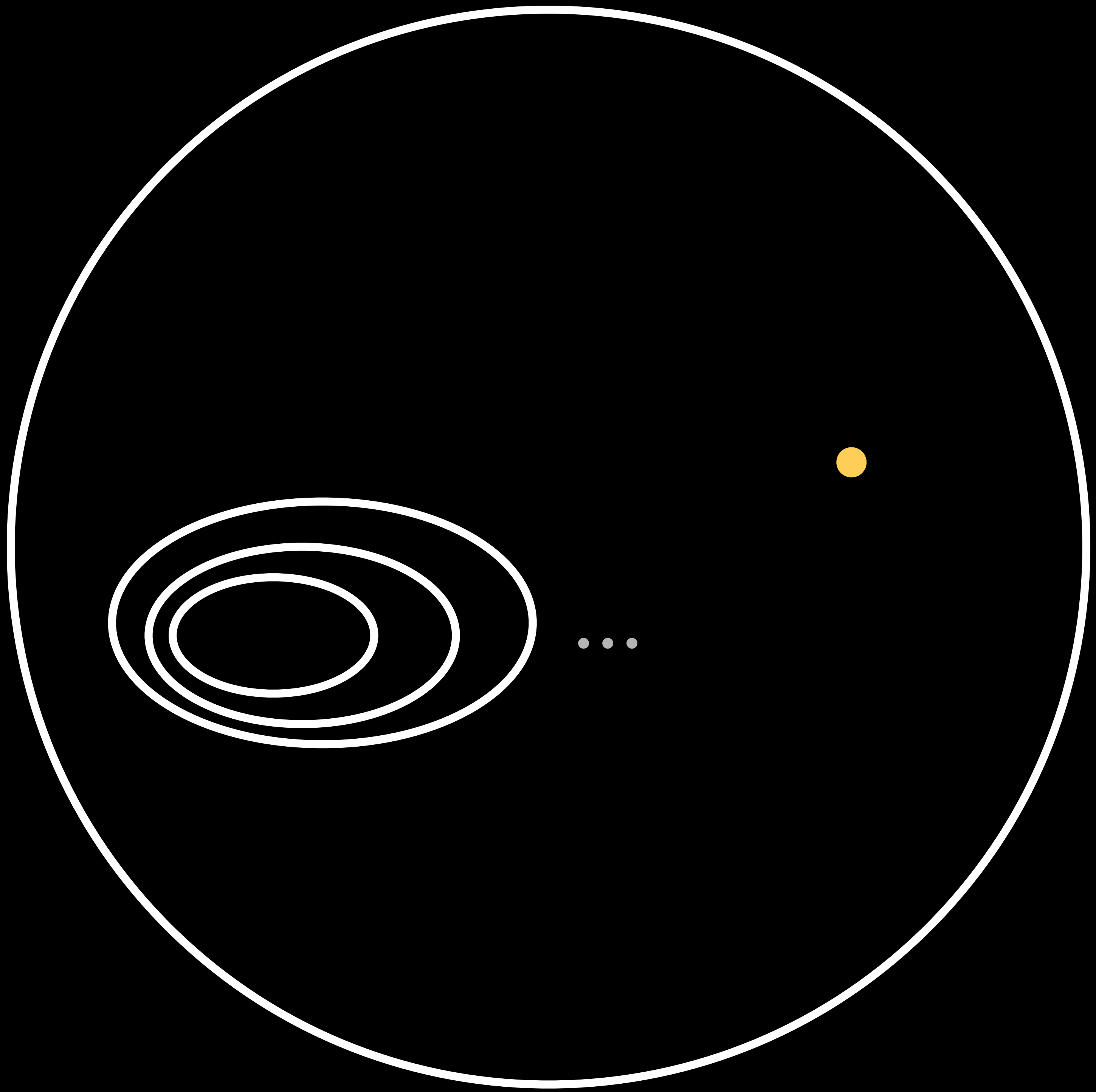
UNDER-APPROXIMATE VERIFICATION

- Parametrised



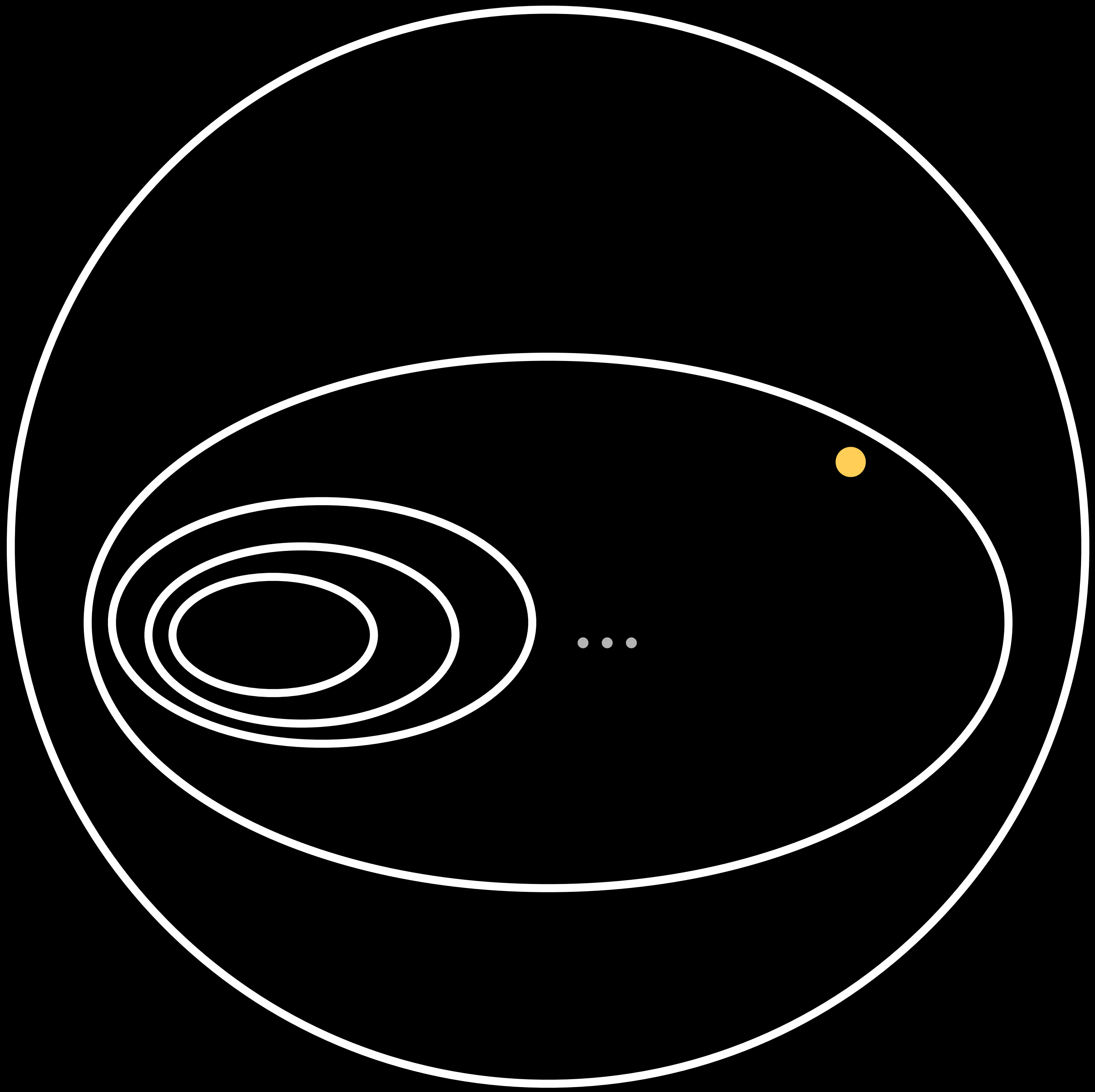
UNDER-APPROXIMATE VERIFICATION

- Parametrised
- Exhaustive



UNDER-APPROXIMATE VERIFICATION

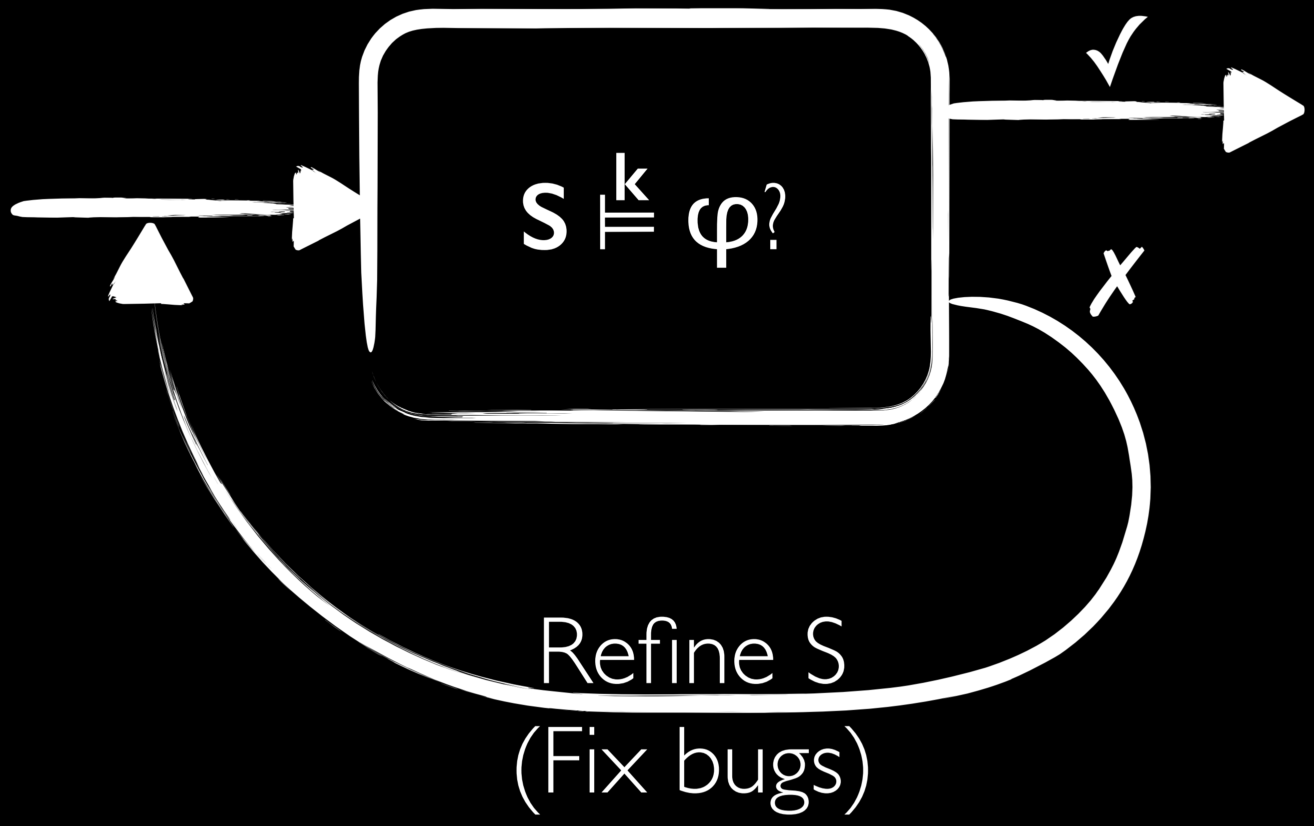
- Parametrised
- Exhaustive



UNDER-APPROXIMATE VERIFICATION

Model Checking

> **Decidable**



System **S**

Specification **φ**

$\mathbf{S} \overset{\mathbf{k}}{\models} \varphi$?

✓

✗

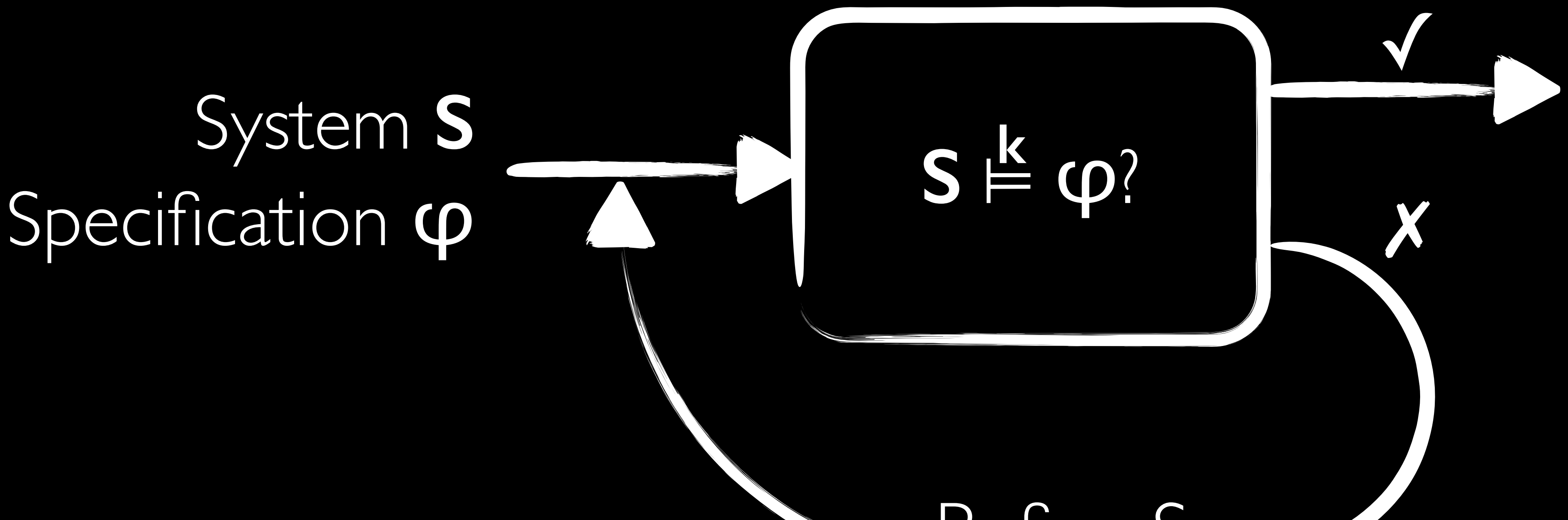
Refine S

(Fix bugs)

UNDER-APPROXIMATE VERIFICATION

Model Checking

> **Decidable**



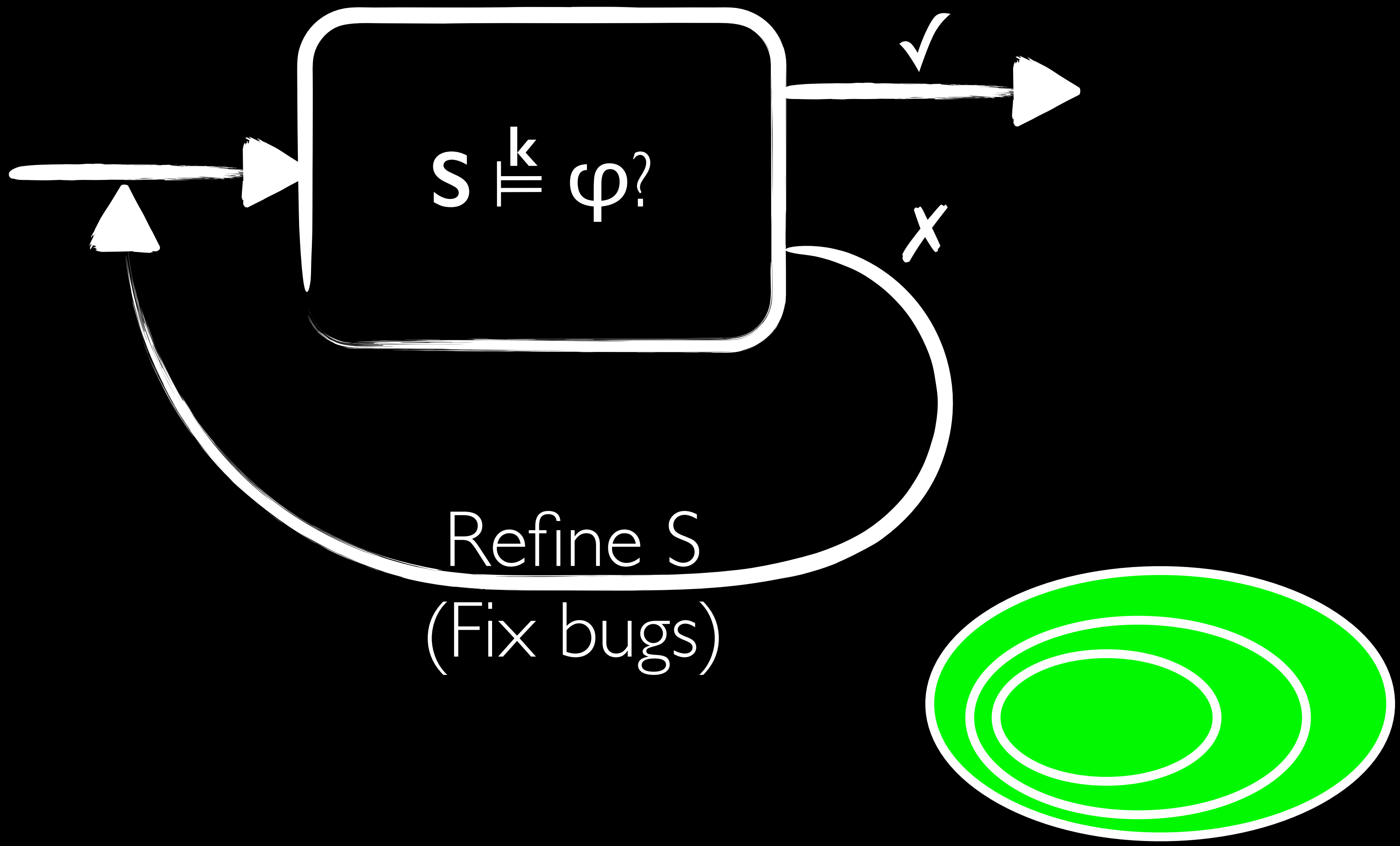
UNDER-APPROXIMATE VERIFICATION

Model Checking

› **Decidable**

System **S**

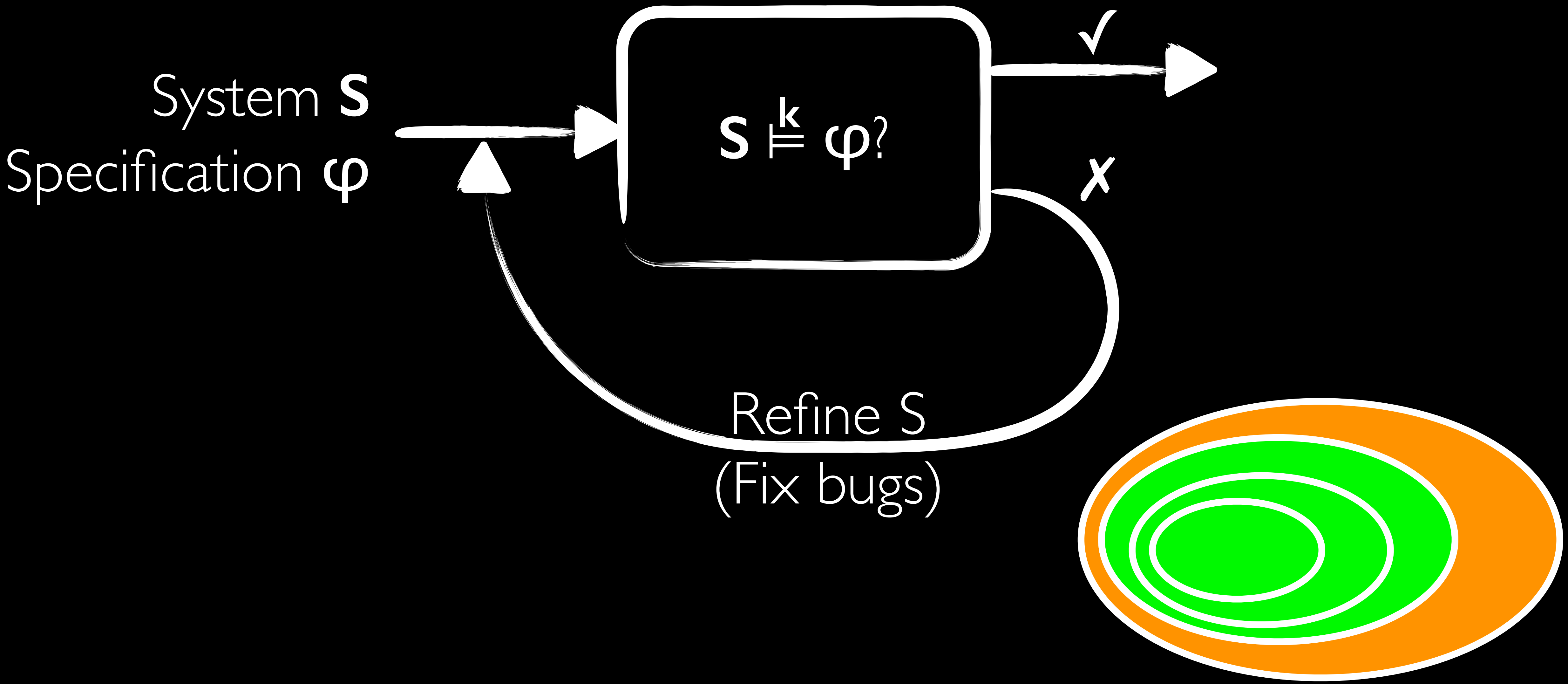
Specification **φ**



UNDER-APPROXIMATE VERIFICATION

Model Checking

> **Decidable**



System **S**

Specification **φ**

$\mathbf{S} \overset{\mathbf{k}}{\models} \varphi$?

Refine S

(Fix bugs)

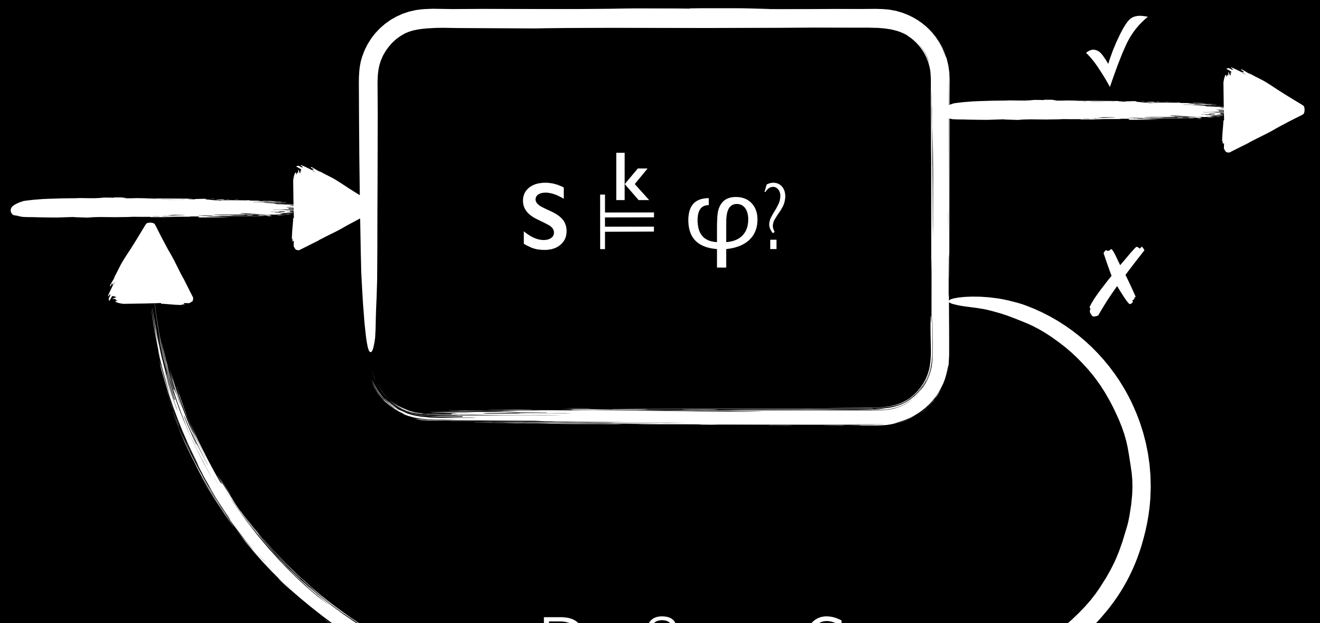
UNDER-APPROXIMATE VERIFICATION

Model Checking

> **Decidable**

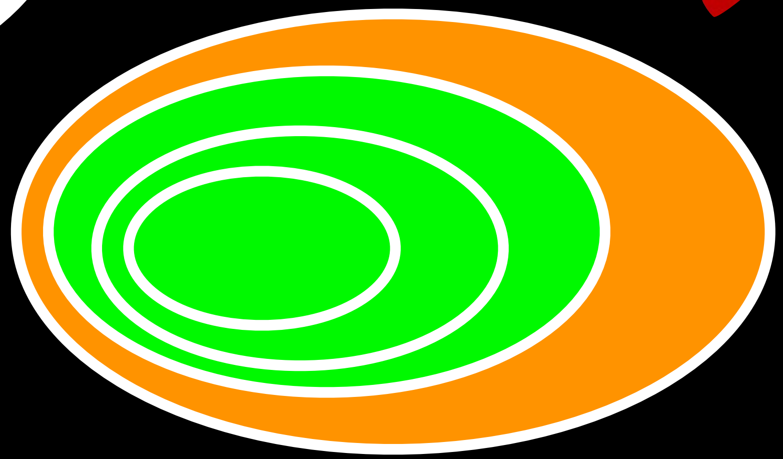
System **S**

Specification **φ**



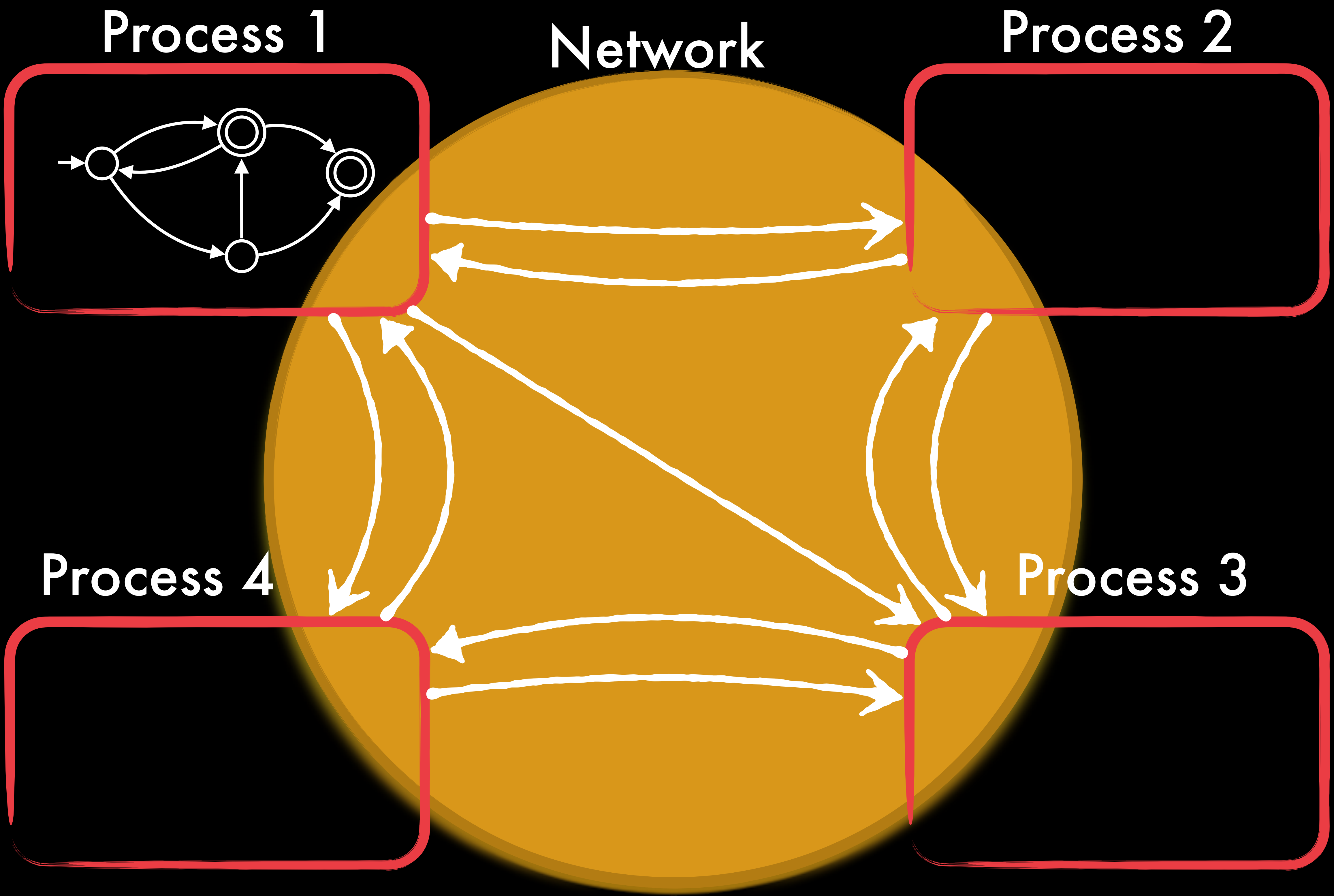
Refine S

(Fix bugs)

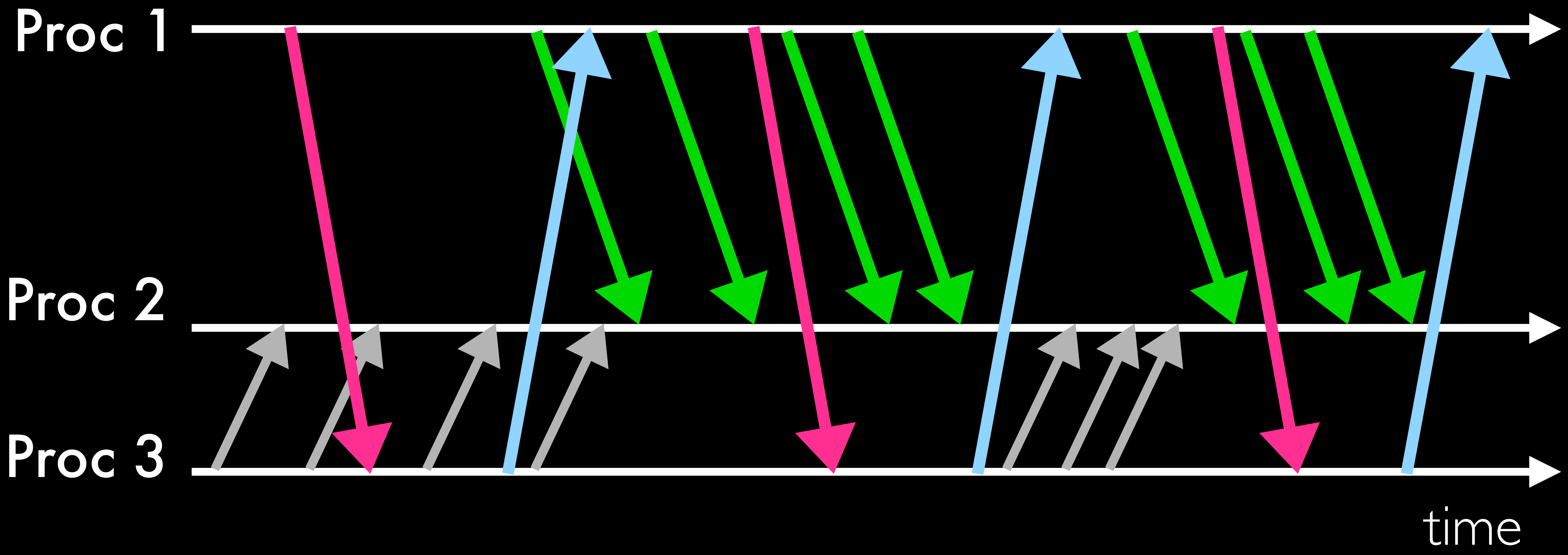


COMMUNICATING RECURSIVE PROGRAMS: CONTROL AND SPLIT-WIDTH

COMMUNICATING DISTRIBUTED SYSTEMS



BEHAVIOURS : MESSAGE SEQUENCE CHARTS



VERIFICATION PROBLEMS

- Emptiness or Reachability
- Inclusion or Universality
- Satisfiability ϕ
- Model Checking: $S \models \phi$
  - Temporal logics
  - Propositional dynamic logics
  - Monadic second order logic

COMMUNICATING RECURSIVE PROGRAMS:

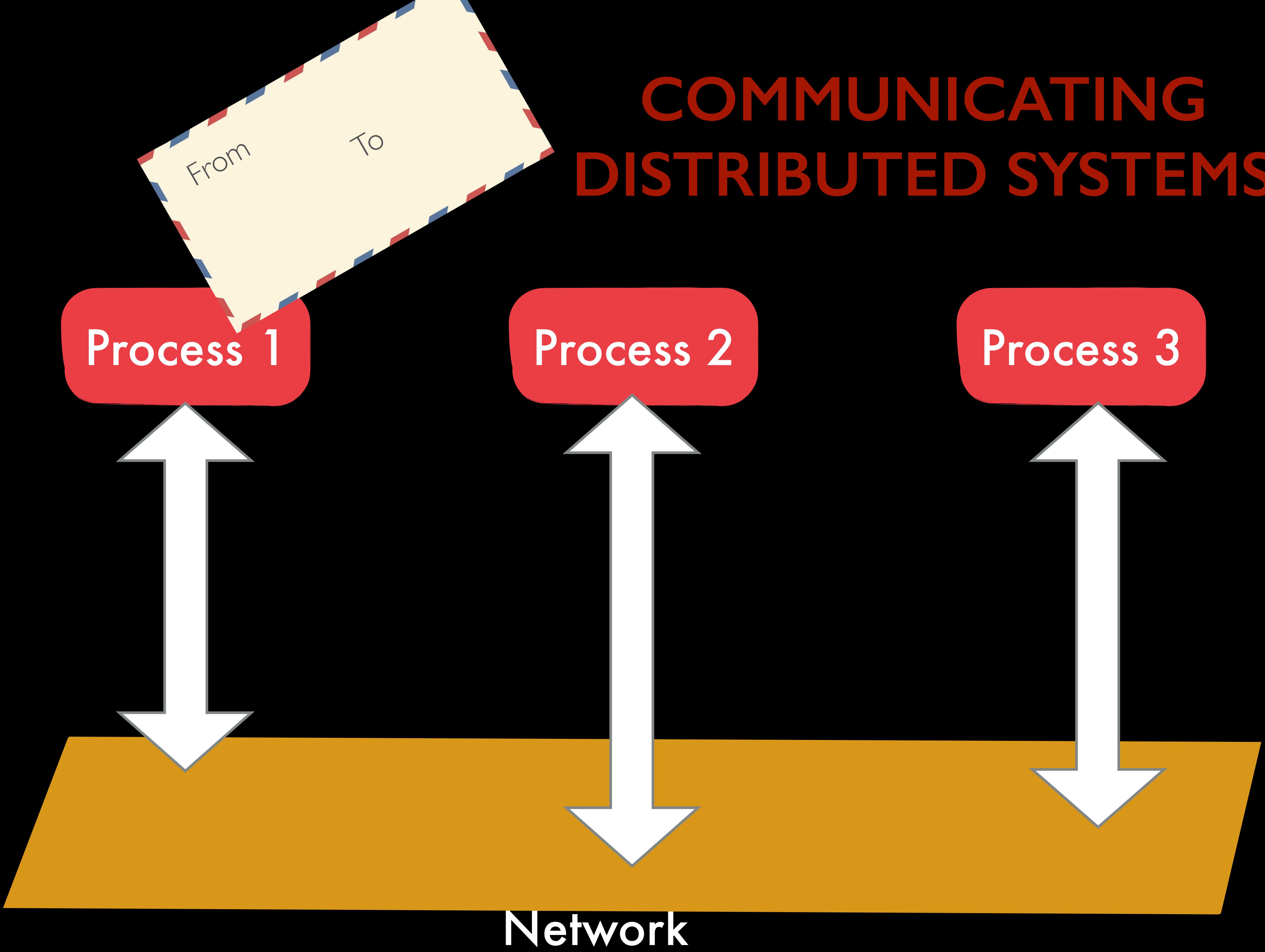
- Turing powerful: verification undecidable
- Under-upproximations
    - Decidable
    - Controllable

COMMUNICATING RECURSIVE PROGRAMS: CONTROL AND SPLIT-WIDTH

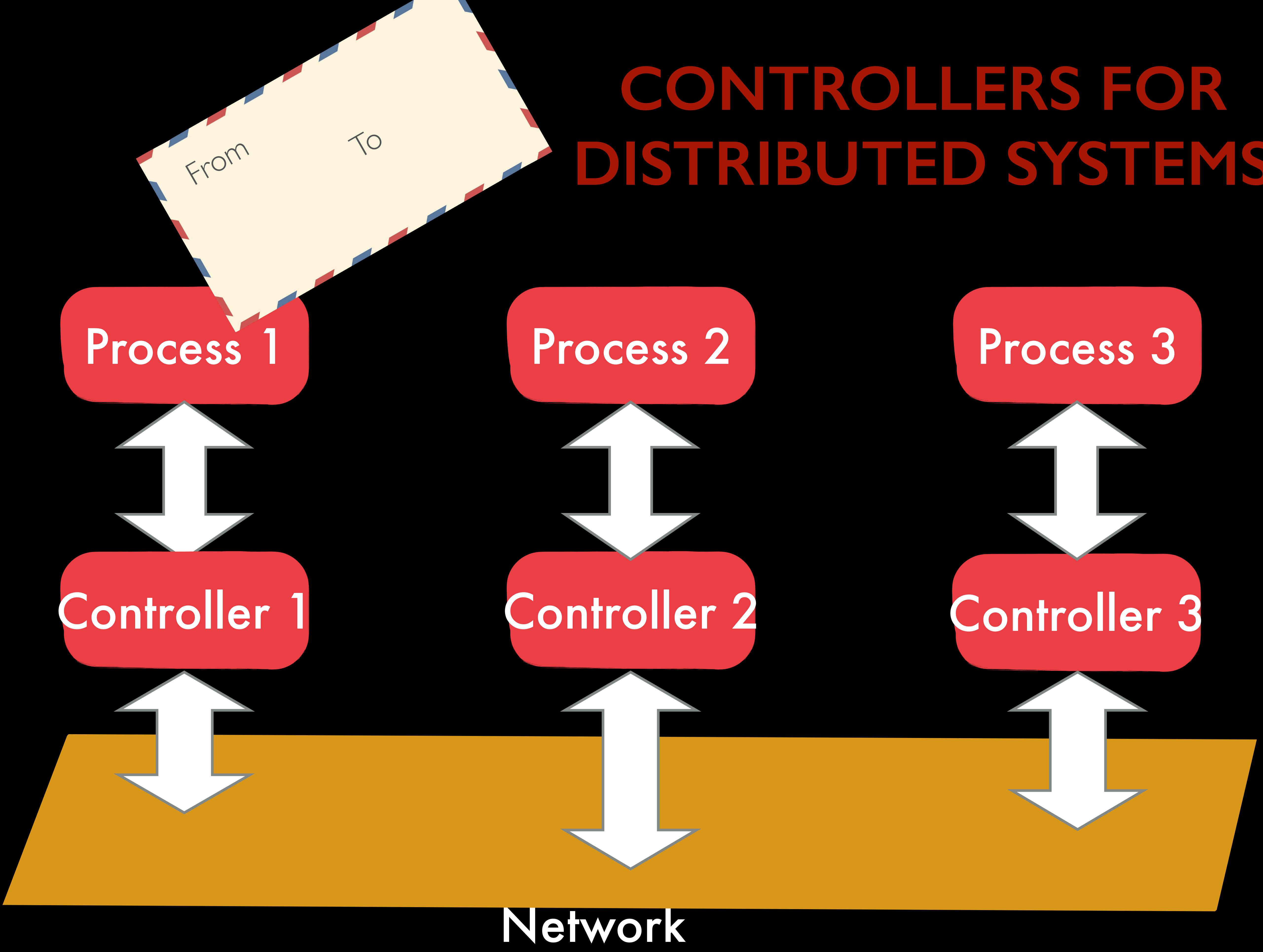
- Turing powerful: verification undecidable
- Under-upproximations
    - Decidable
    - Controllable

CONTROLLERS FOR VERIFICATION OF COMMUNICATING SYSTEMS

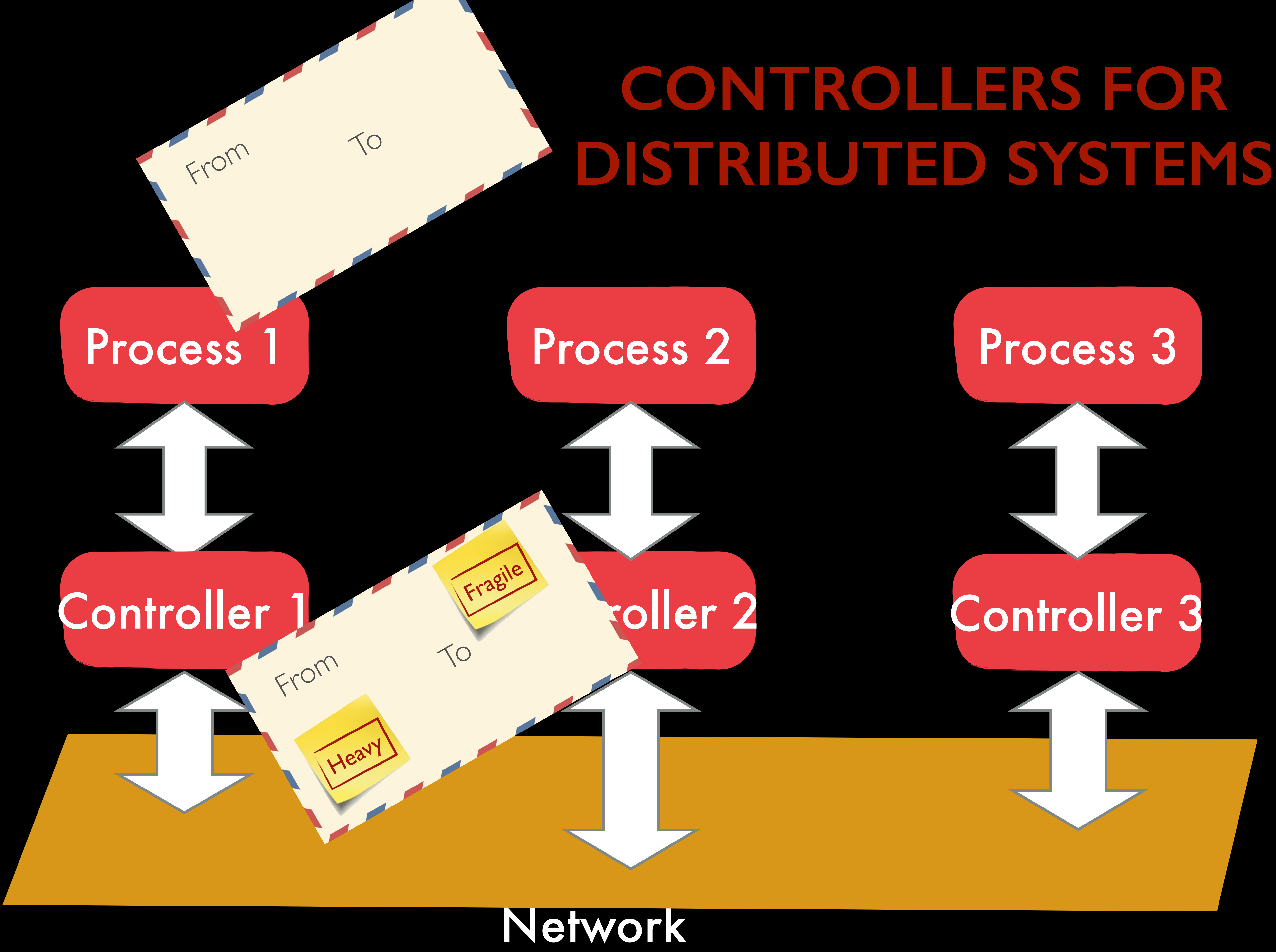
COMMUNICATING DISTRIBUTED SYSTEMS

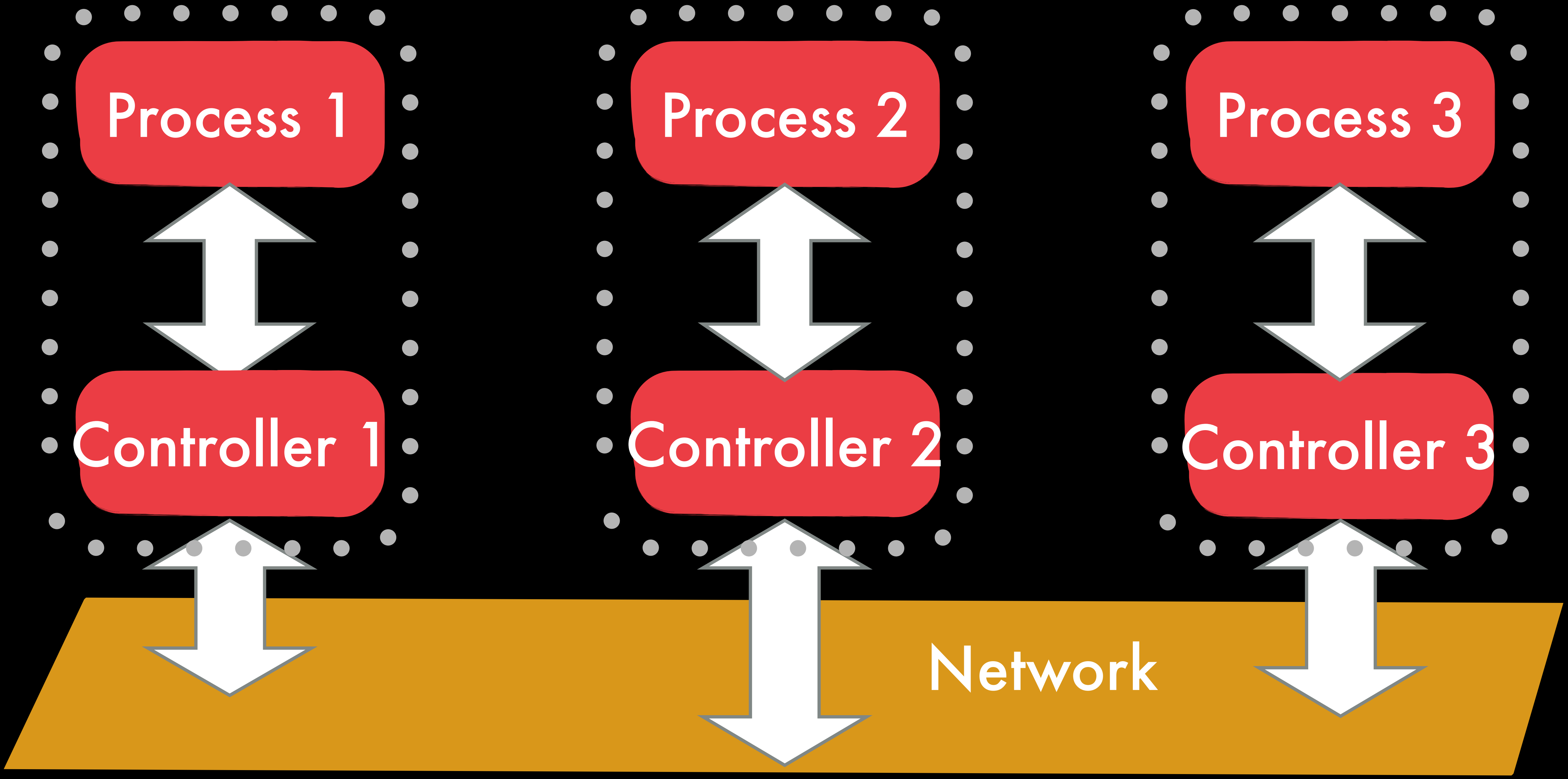


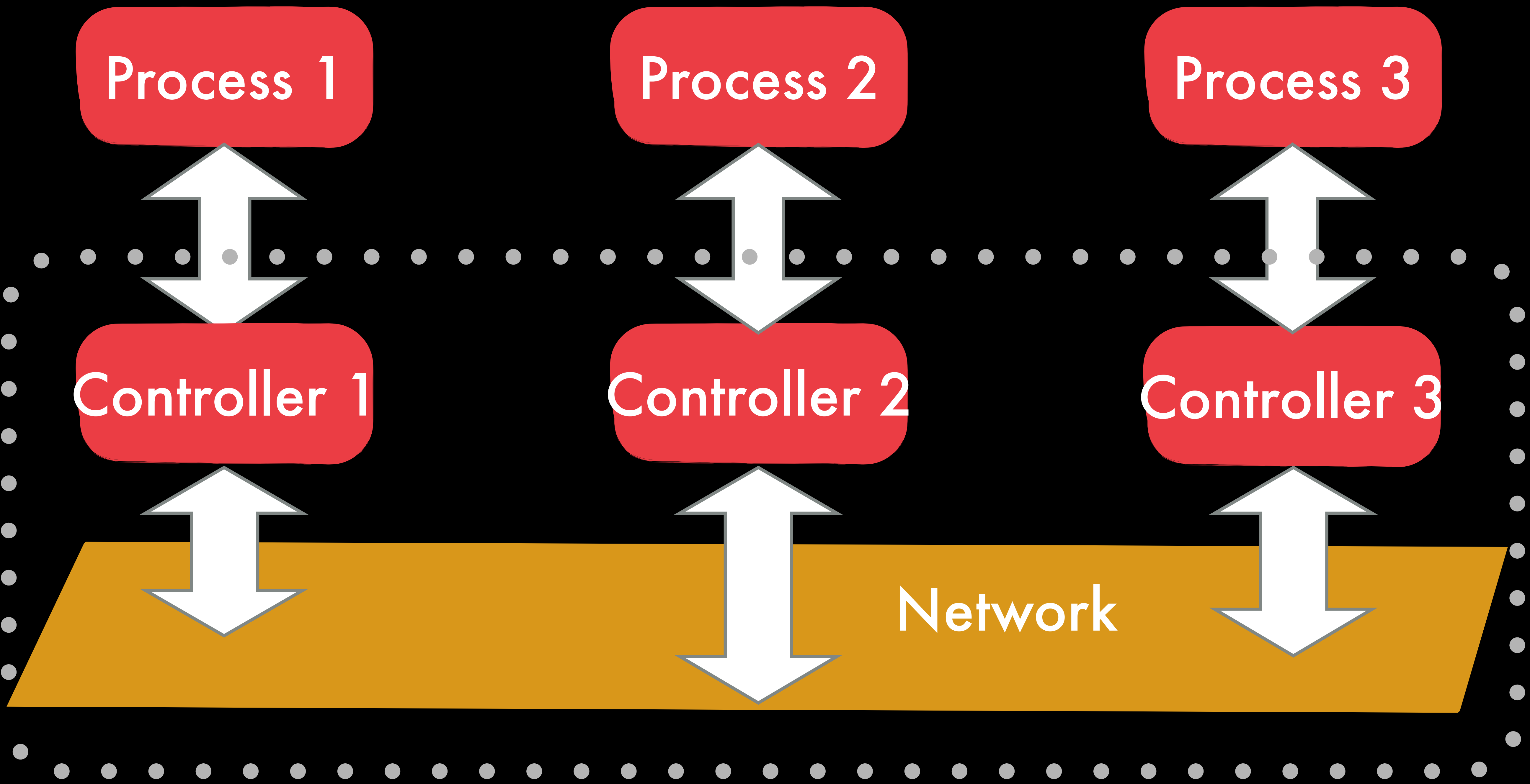
CONTROLLERS FOR DISTRIBUTED SYSTEMS



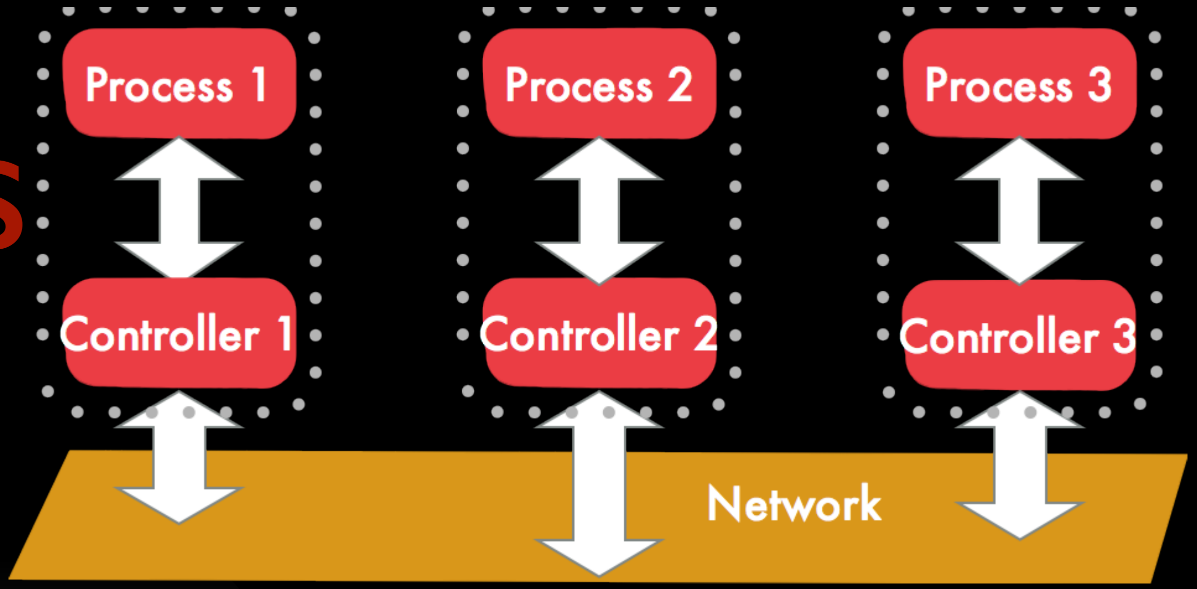
CONTROLLERS FOR DISTRIBUTED SYSTEMS




Process 1
Process 2
Process 3
Controller 1
Controller 2
Controller 3
Network

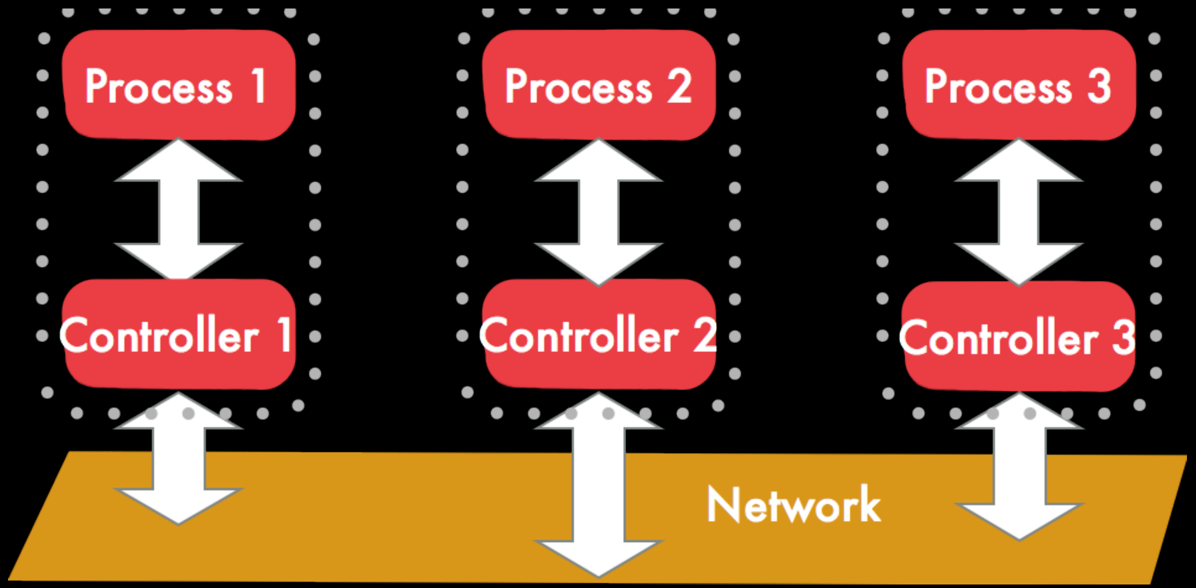

Process 1
Process 2
Process 3
Controller 1
Controller 2
Controller 3
Network

CONTROLLERS FOR DISTRIBUTED SYSTEMS



- Collection of local controllers
- Communication via piggy-backing
- Privacy: Do NOT read states/messages

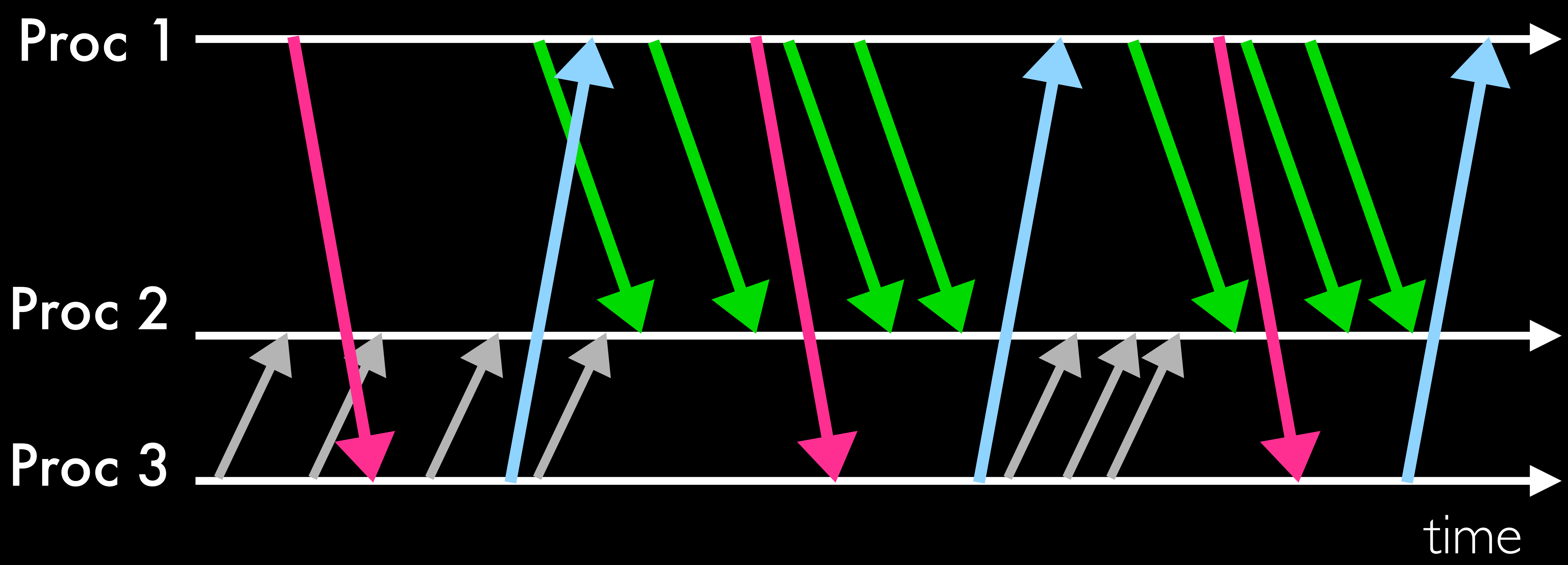
LET'S DESIGN A CONTROLLER



UNDER-APPROXIMATION: BOUNDED (*K*) PHASE

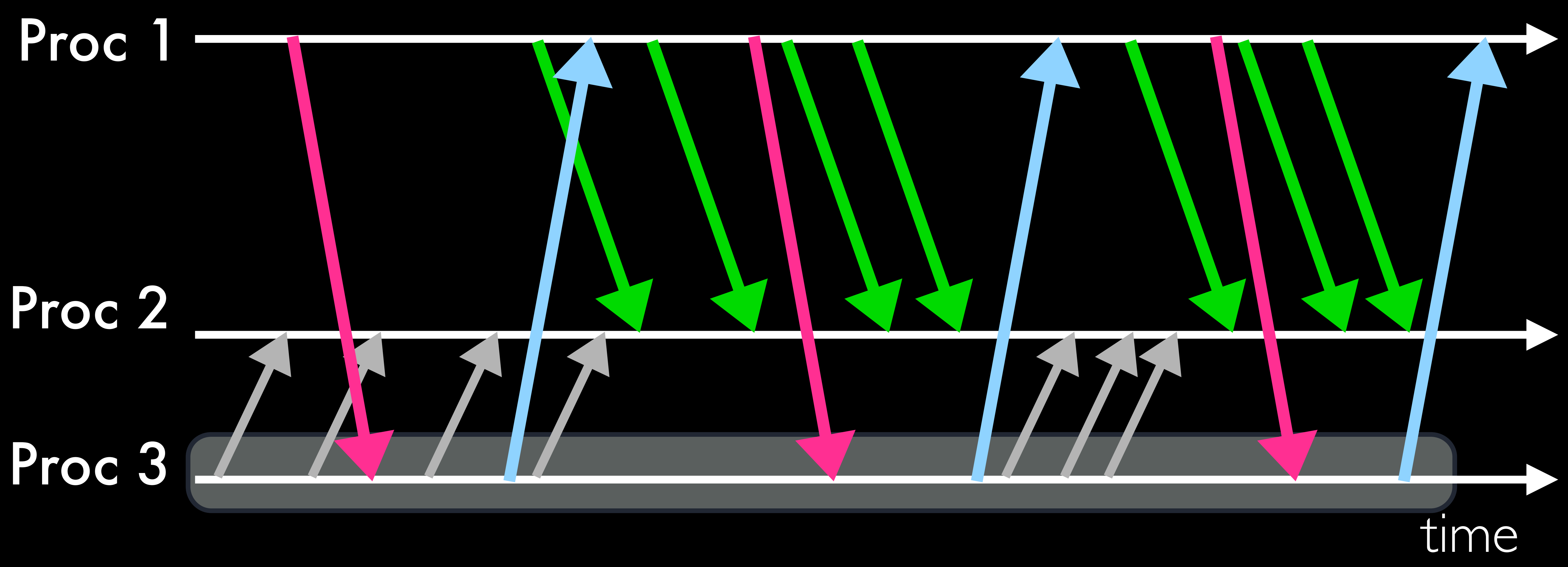
PHASE

Receive from one process, send to all processes



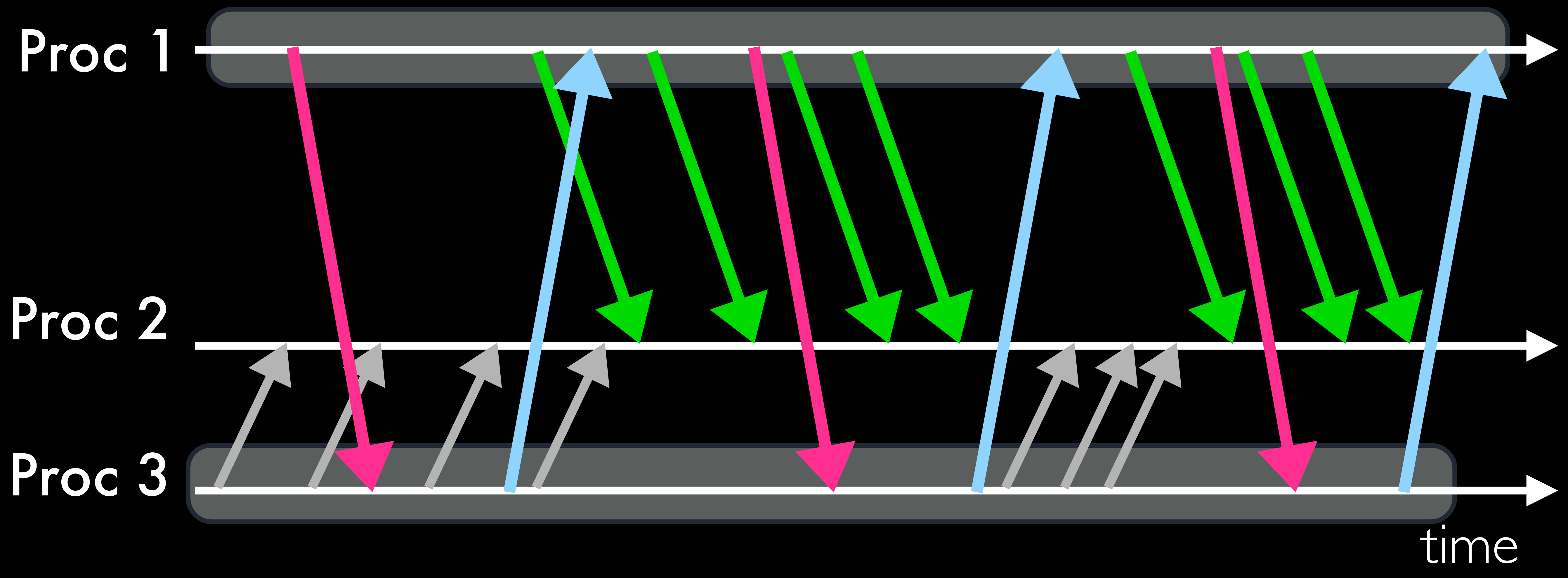
PHASE

Receive from one process, send to all processes



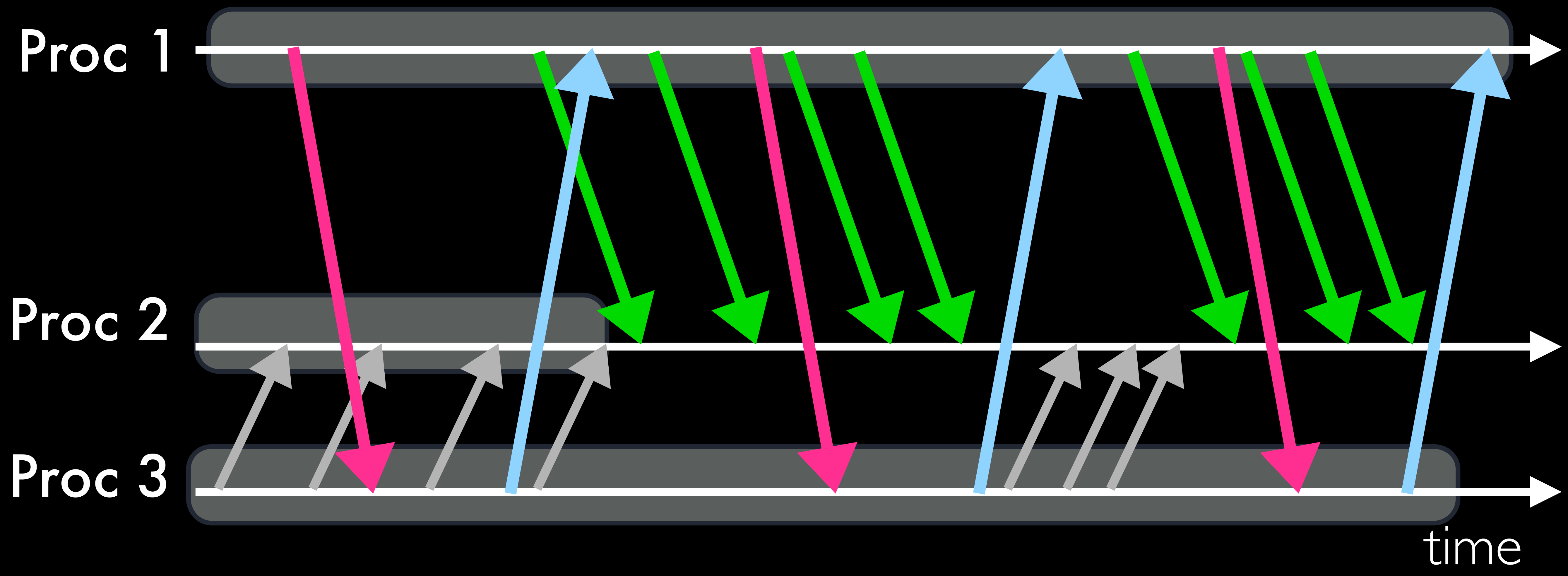
PHASE

Receive from one process, send to all processes



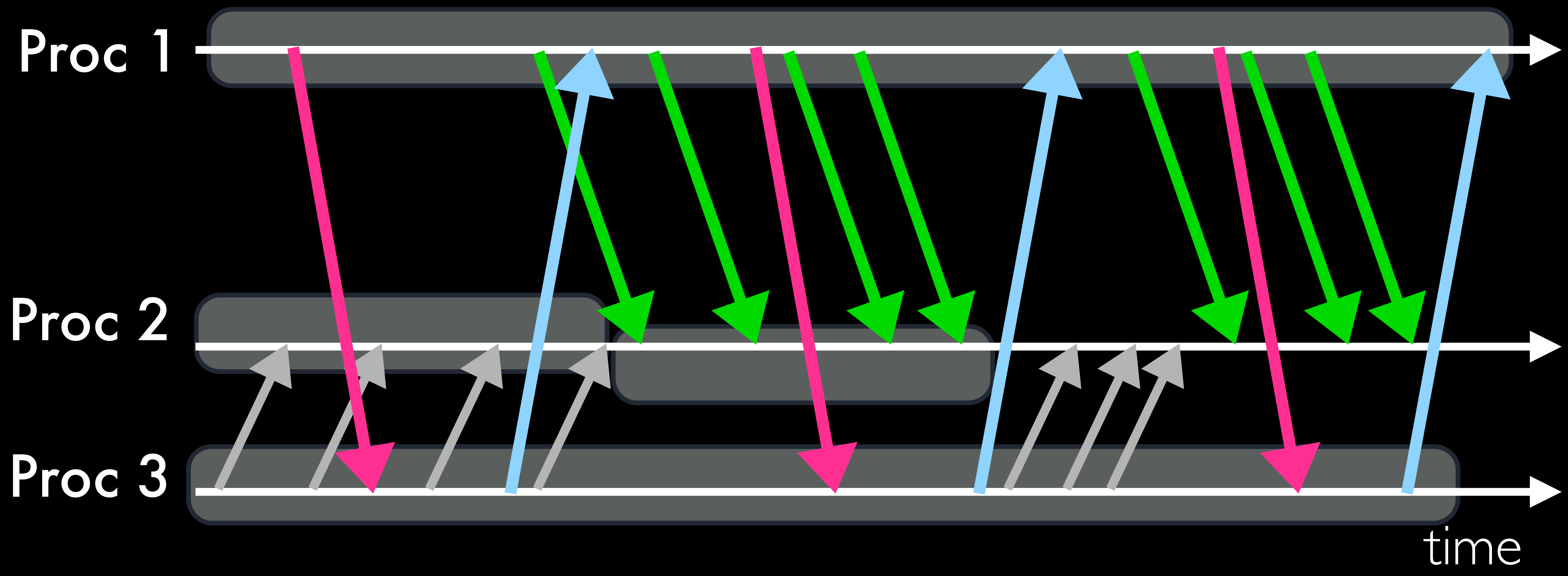
PHASE

Receive from one process, send to all processes



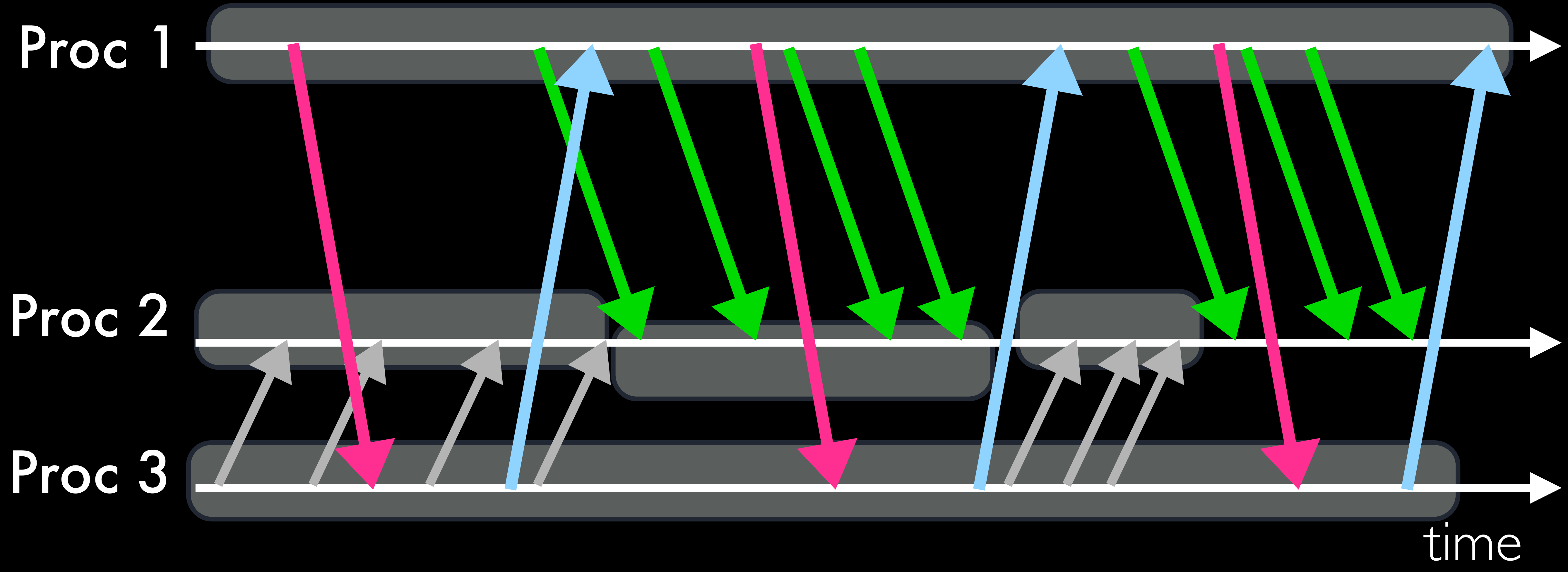
PHASE

Receive from one process, send to all processes



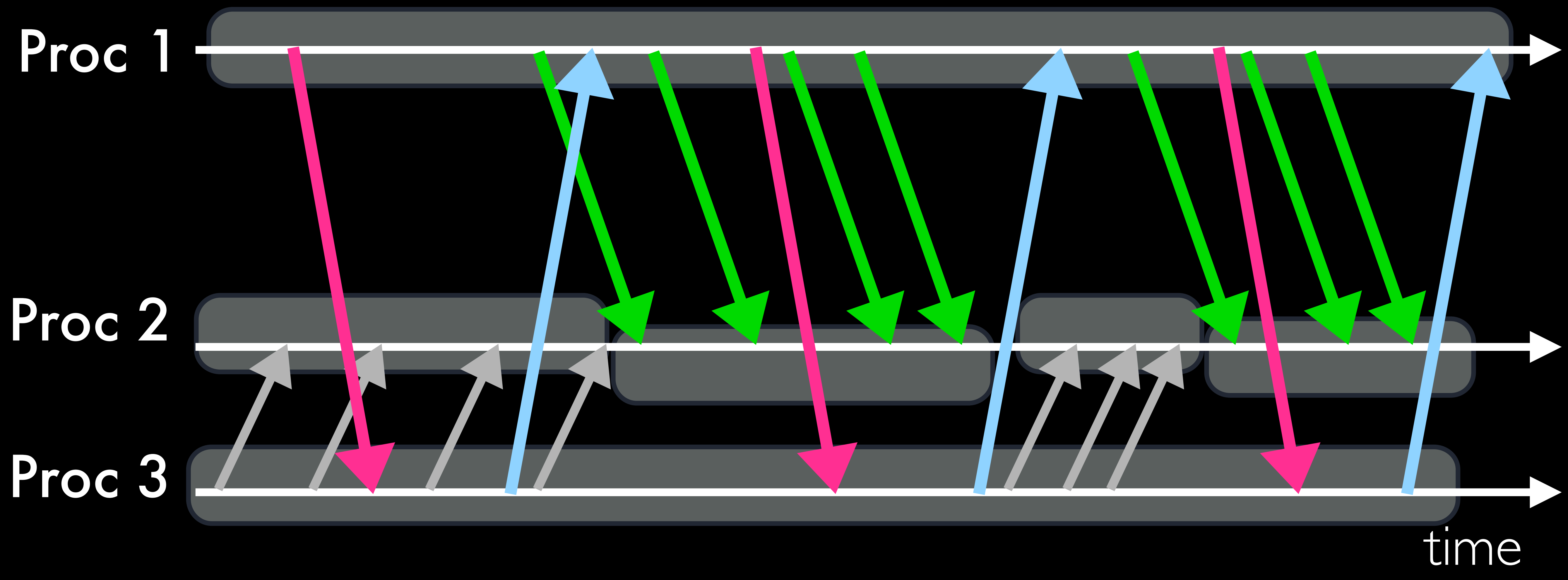
PHASE

Receive from one process, send to all processes



PHASE

Receive from one process, send to all processes

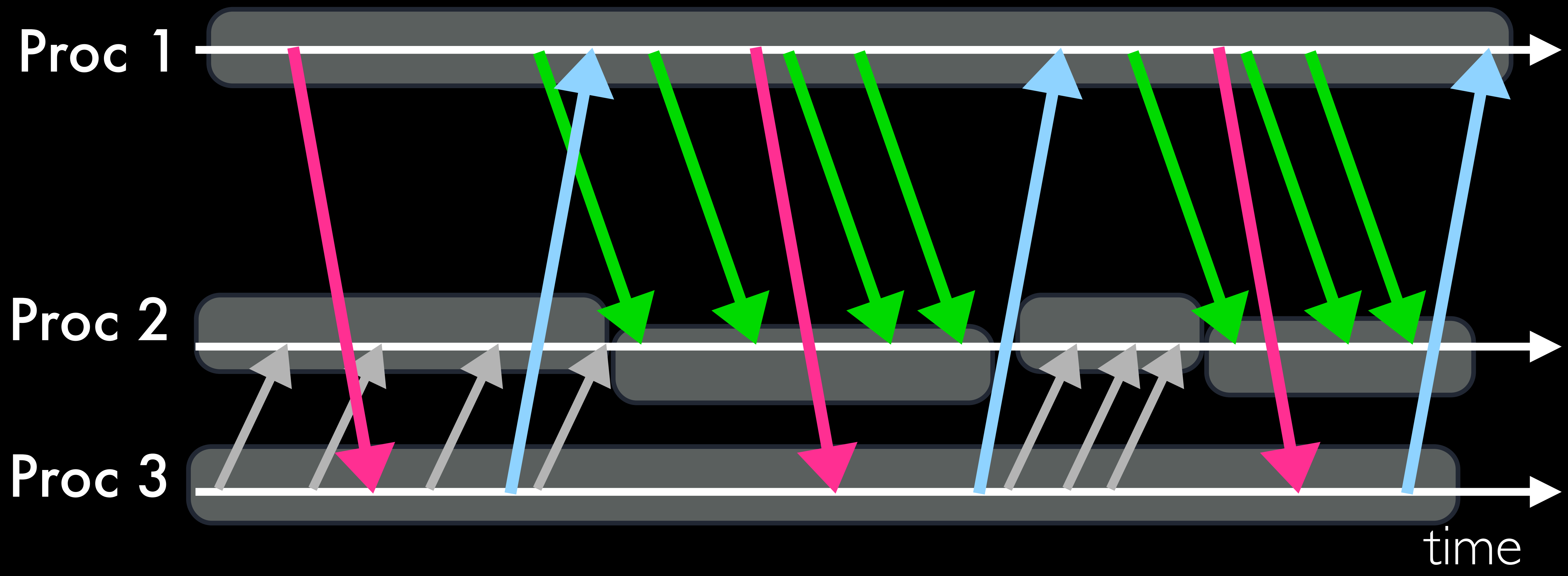


PHASE

Receive from one process, send to all processes

k-BOUNDED PHASE

1. At most k phases on each process
2. No cycles

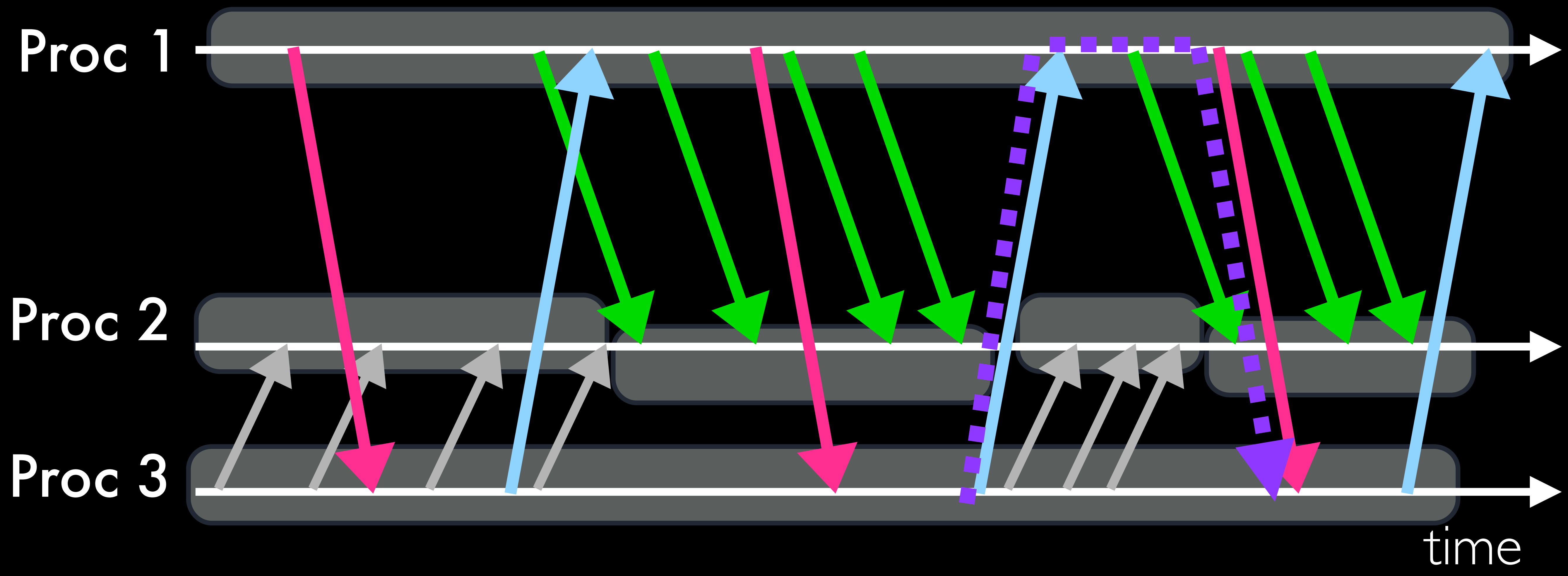


PHASE

Receive from one process, send to all processes

k-BOUNDED PHASE

1. At most k phases on each process
2. No cycles

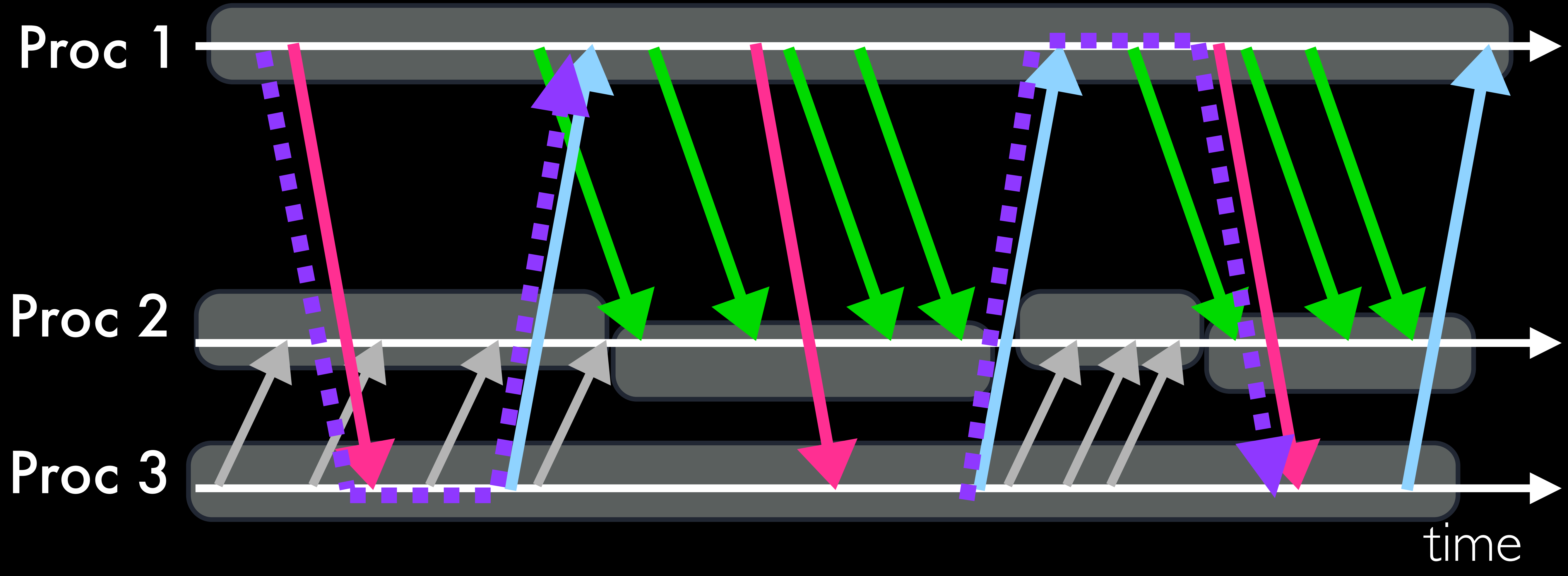


PHASE

Receive from one process, send to all processes

k-BOUNDED PHASE

1. At most k phases on each process
2. No cycles

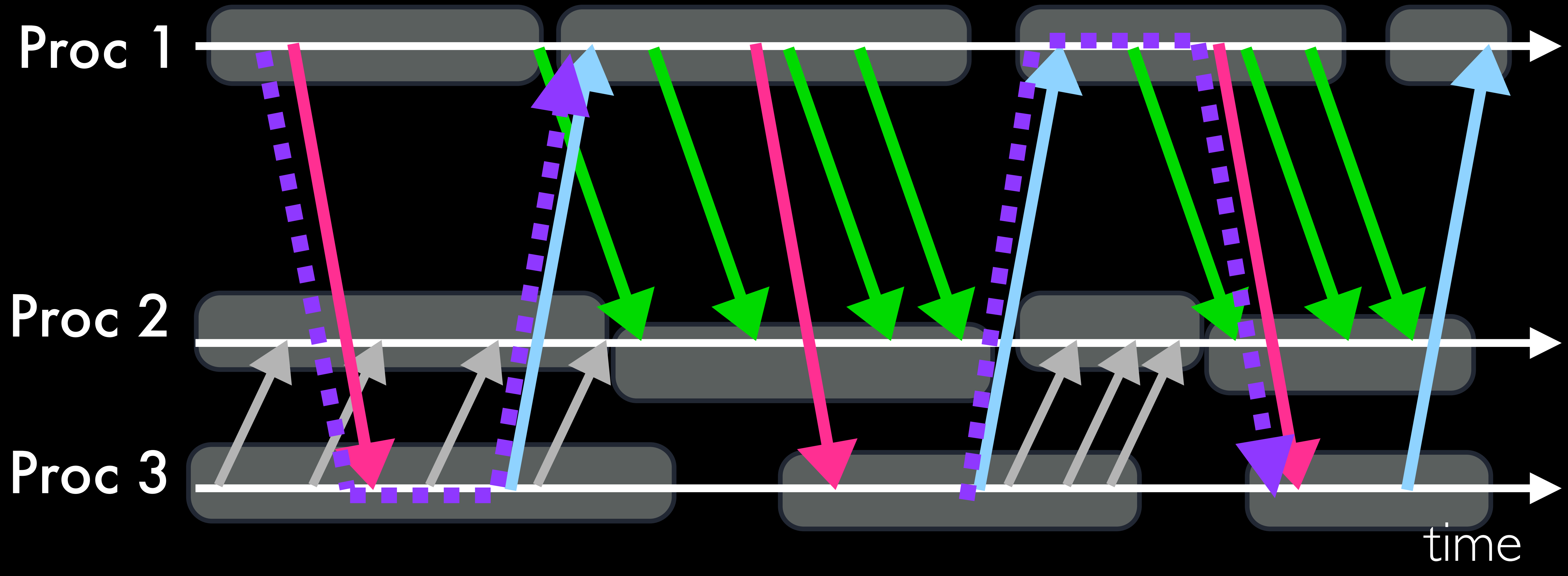


PHASE

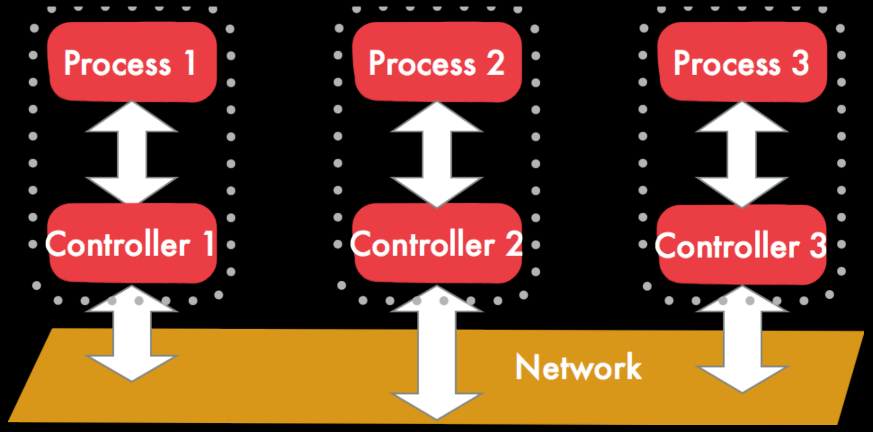
Receive from one process, send to all processes

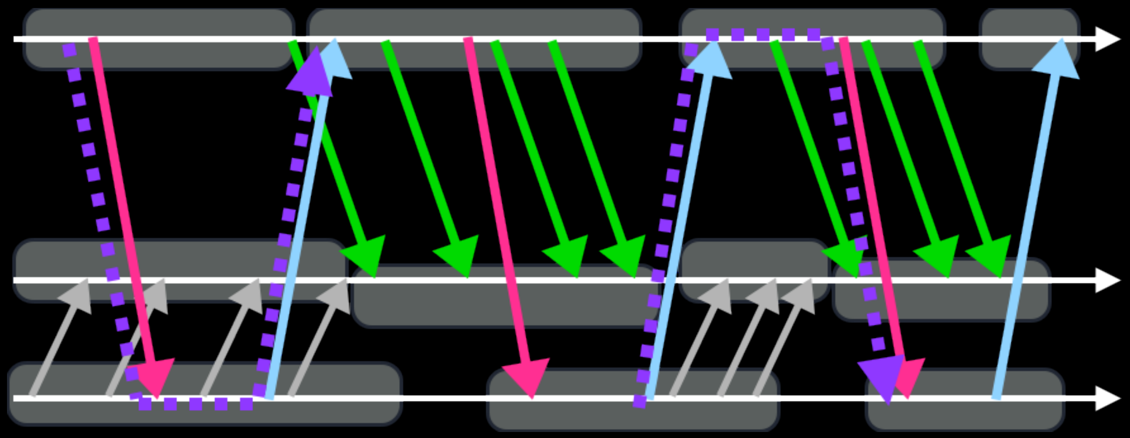
k-BOUNDED PHASE

1. At most k phases on each process
2. No cycles



DISTRIBUTED CONTROLLER FOR K-BOUNDED PHASE U-A





A local controller for each process

State

Has a Phase Counter

Remembers current sender

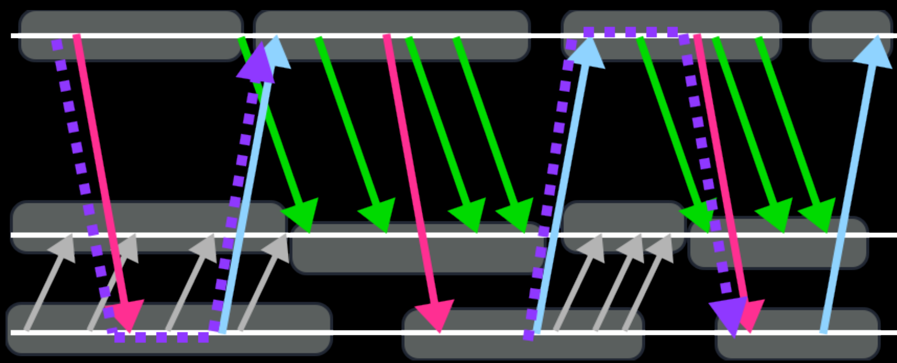
Transitions

Different sender?

Detect Cycle?

Increment counter,
Update sender

DISTRIBUTED CONTROLLER FOR K-BOUNDED PHASE U-A



Detect Cycle?

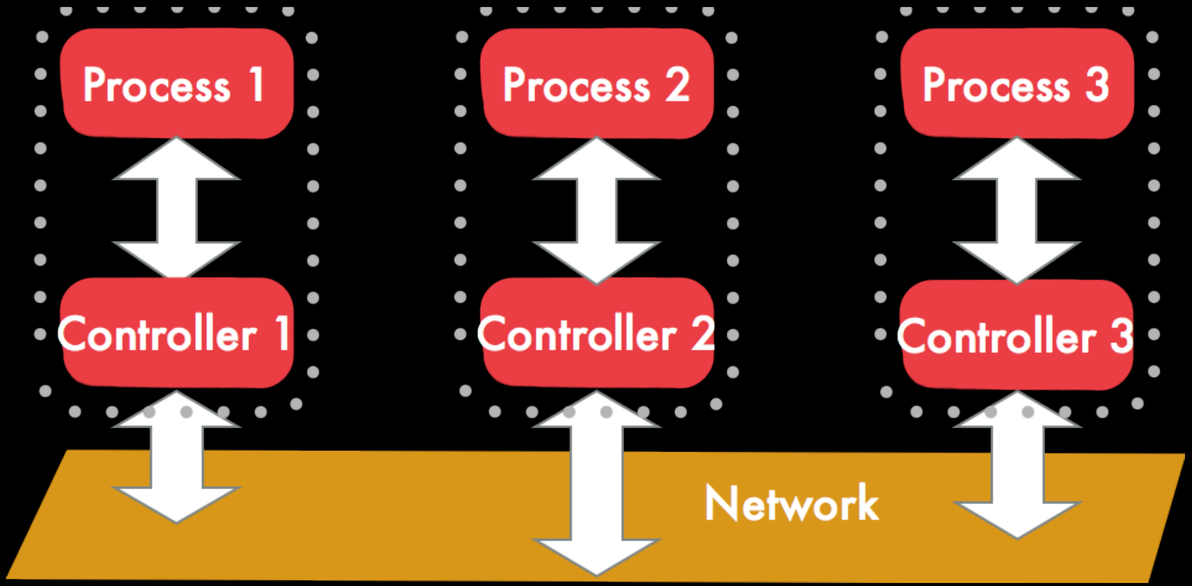
Phase Vectors

best info about phase number of other processes

Sends: tag with phase vector

Receives: update phase vector by taking MAX

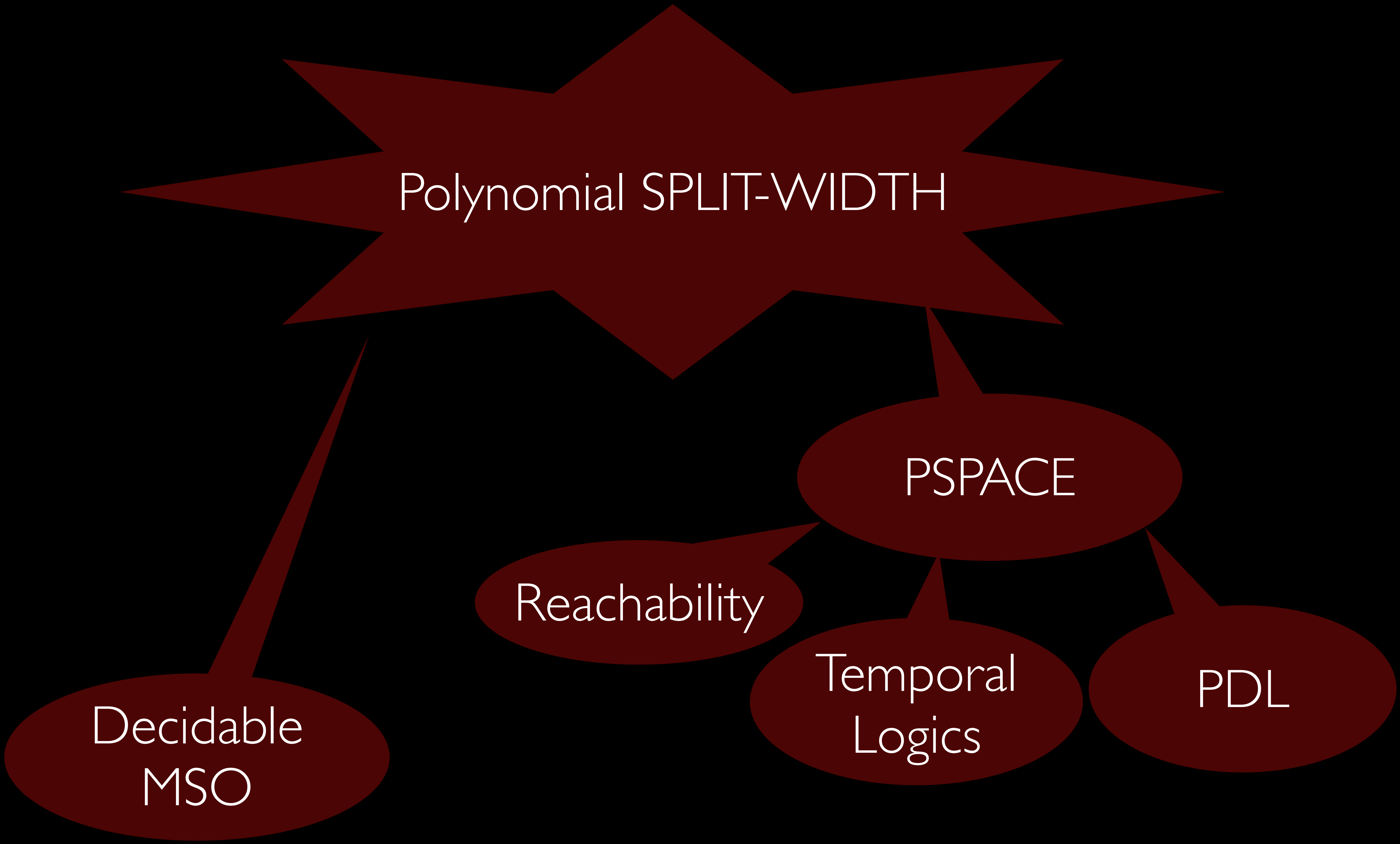
CONTROLLERS FOR BOUNDED PHASE DISTRIBUTED SYSTEMS



- Collection of local controllers
- Communication via piggy-backing
- Privacy: Do NOT read states/messages

- System independent
- Generic
- Deterministic
- Finite state

DECIDABILITY OF K BOUNDED PHASE



Polynomial SPLIT-WIDTH

PSPACE

Reachability

Temporal Logics

PDL

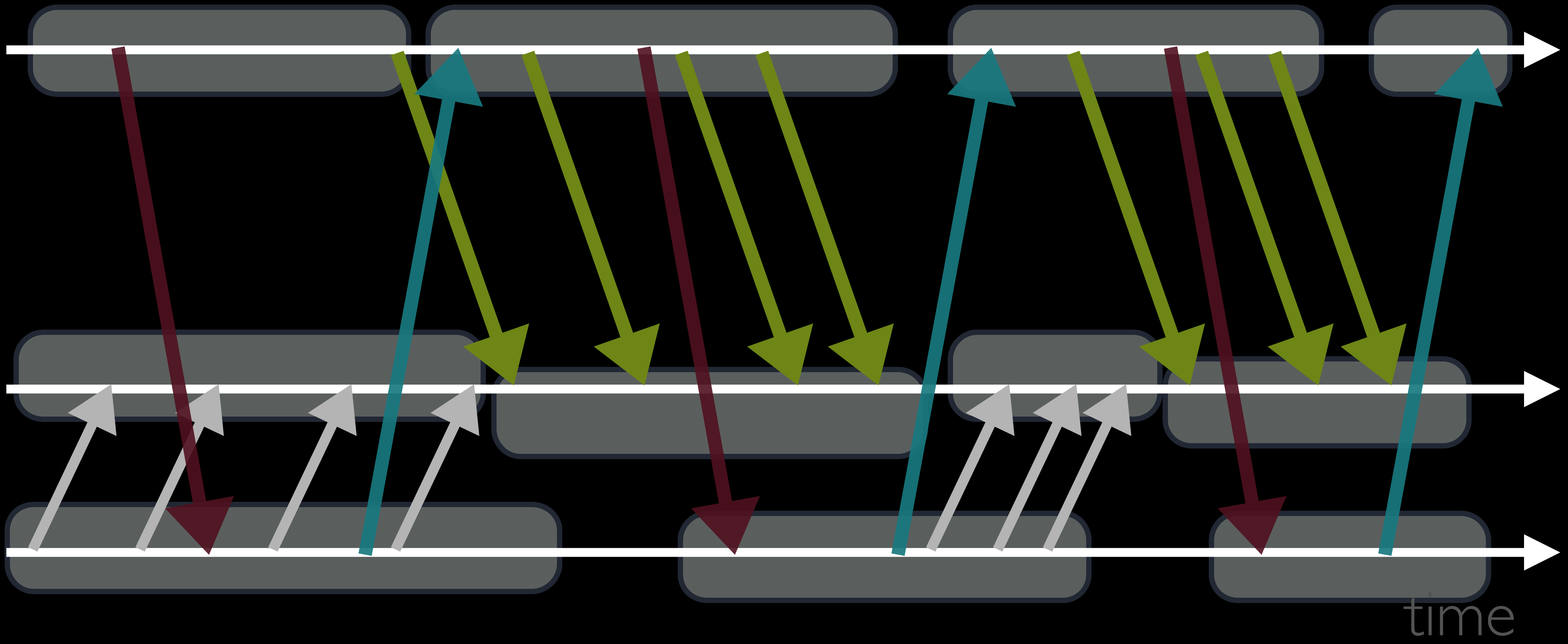
Decidable MSO

Polynomial SPLIT-WIDTH

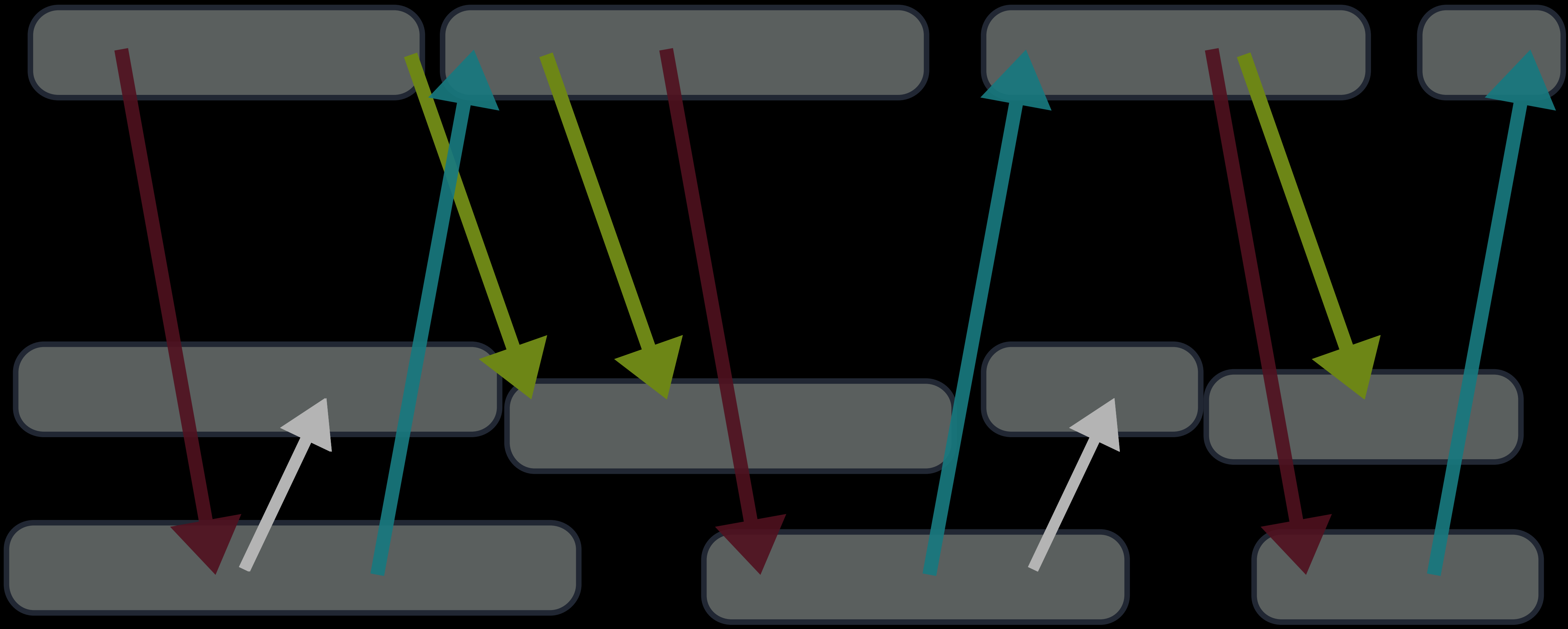
Refine phases to tree-like

bound split-width

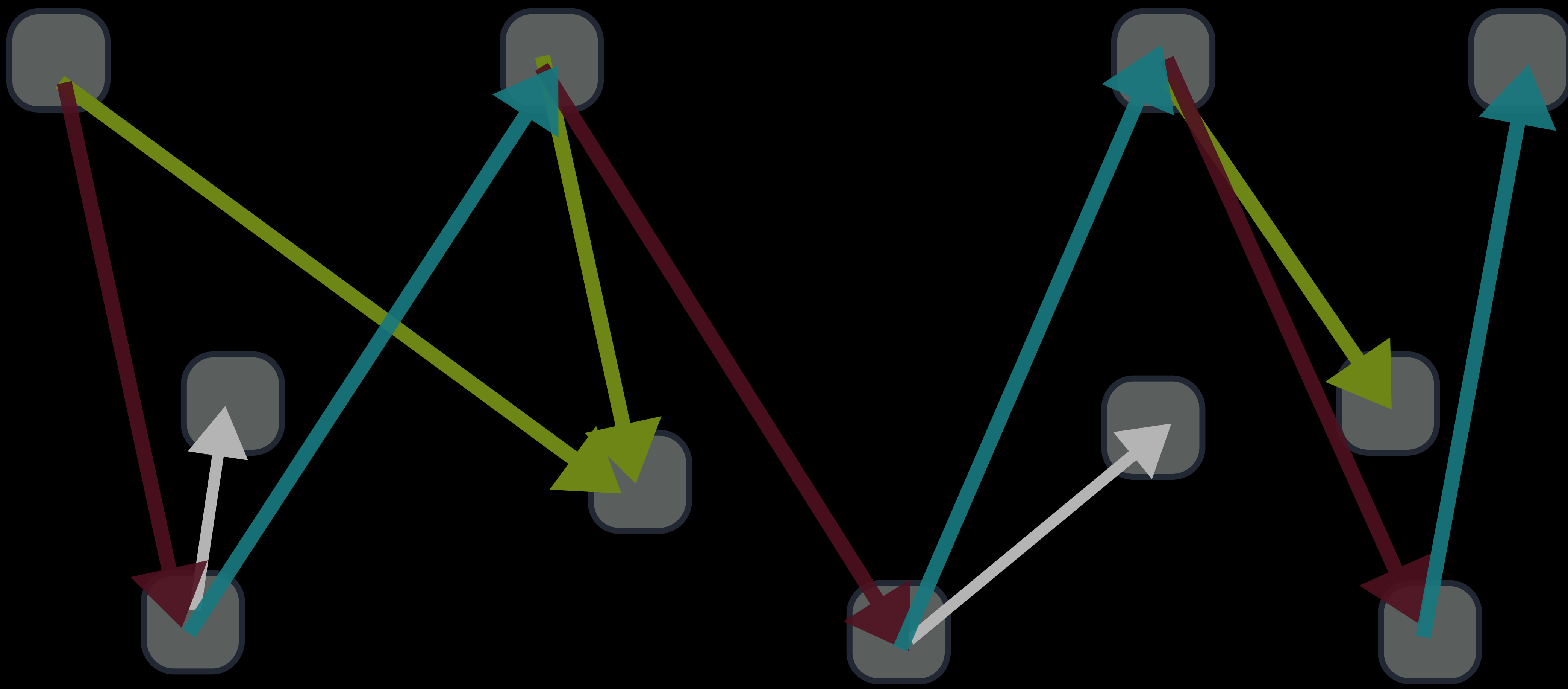
ACYCLIC PHASE DECOMPOSITION



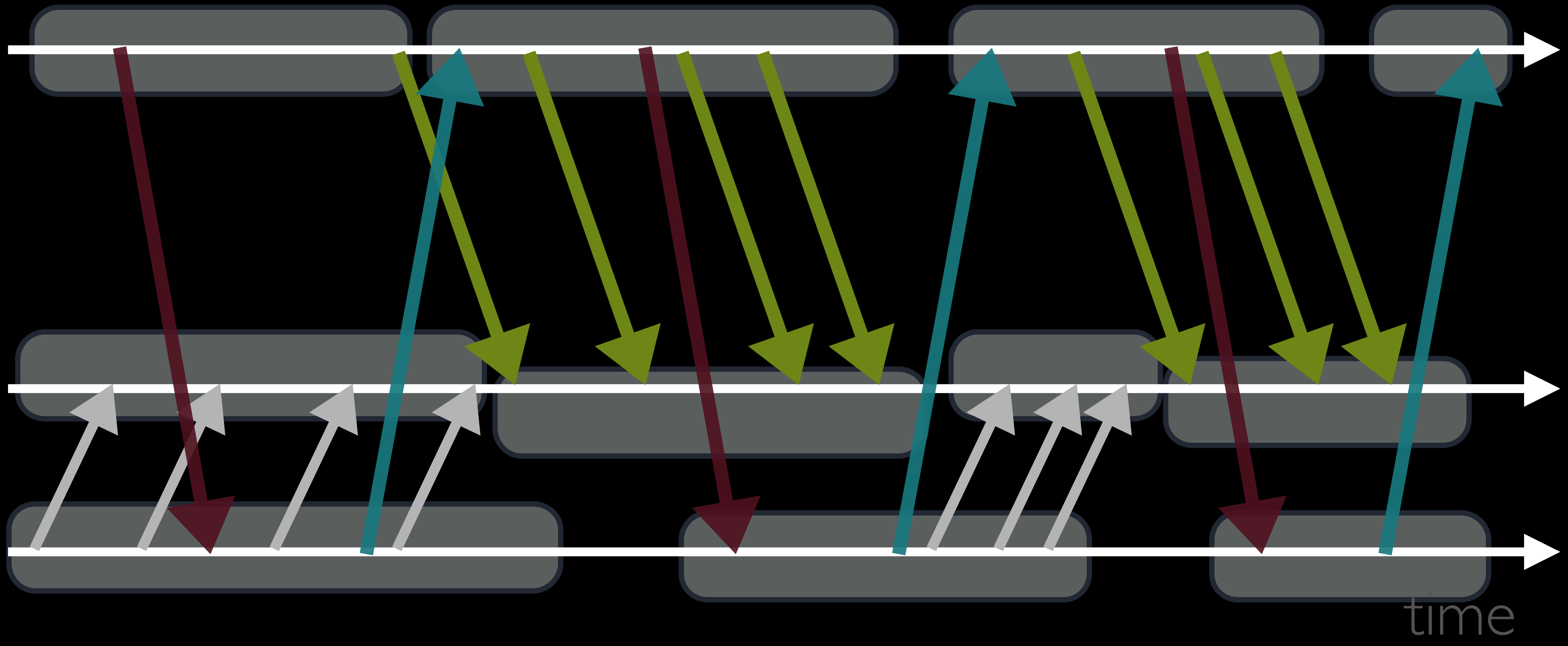
INDUCED GRAPH ON PHASE



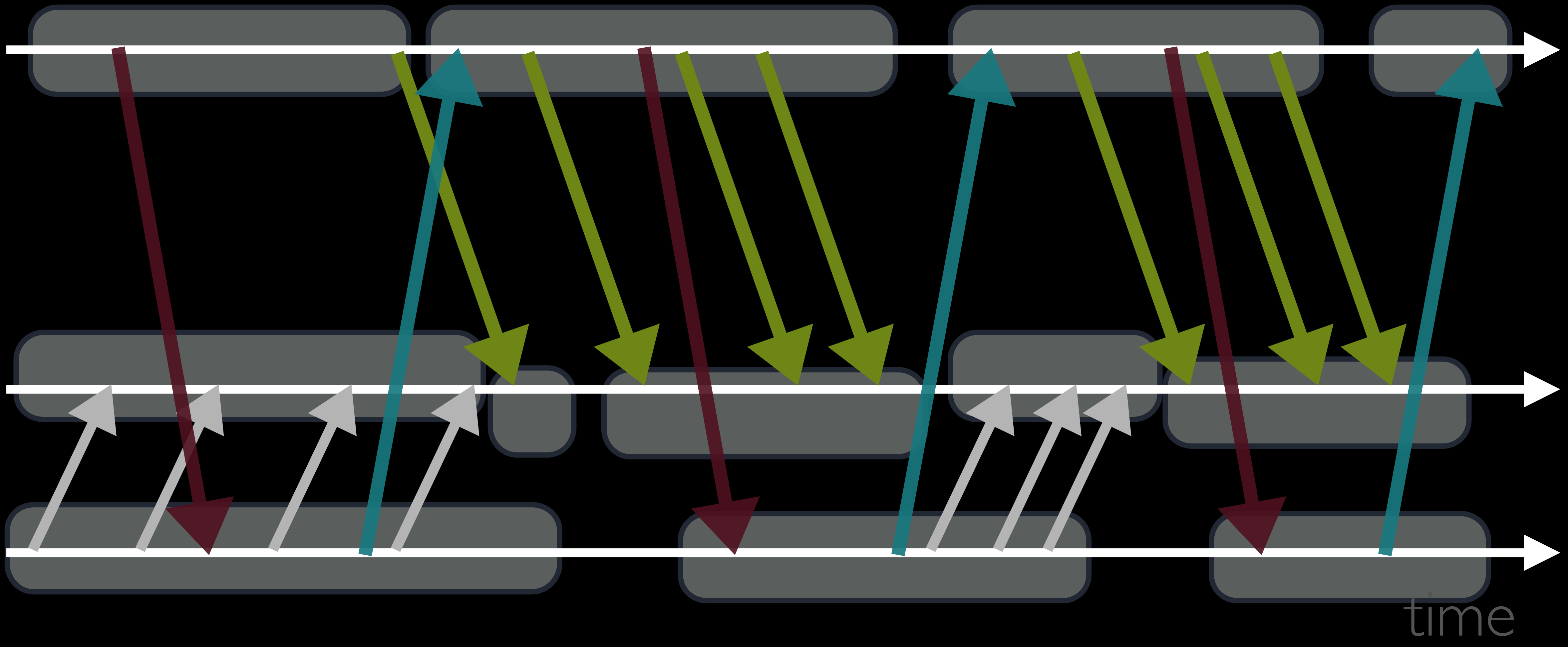
INDUCED GRAPH ON PHASE



PHASE DECOMPOSITION



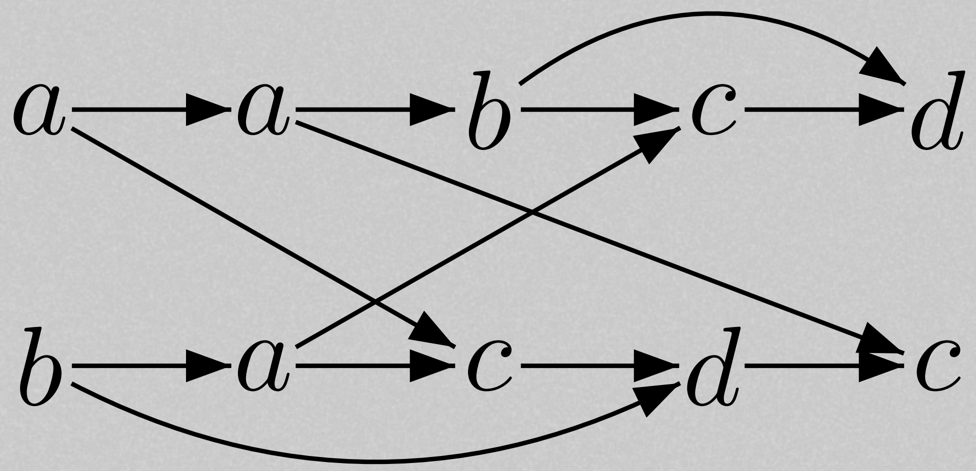
PHASE DECOMPOSITION



Tree-like

Polynomial SPLIT-WIDTH

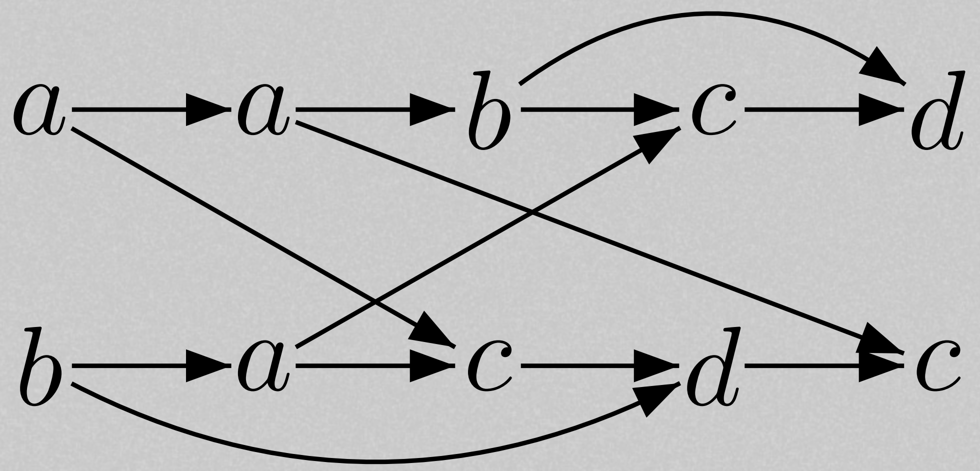
Split-width

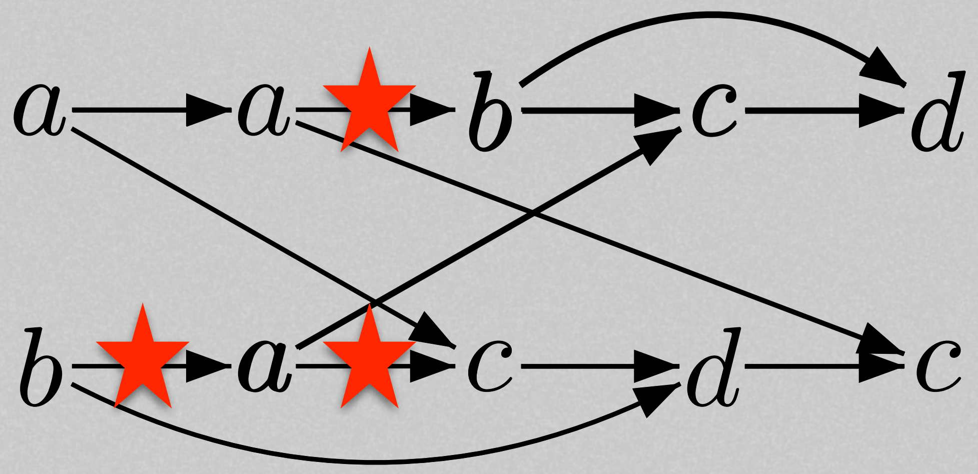


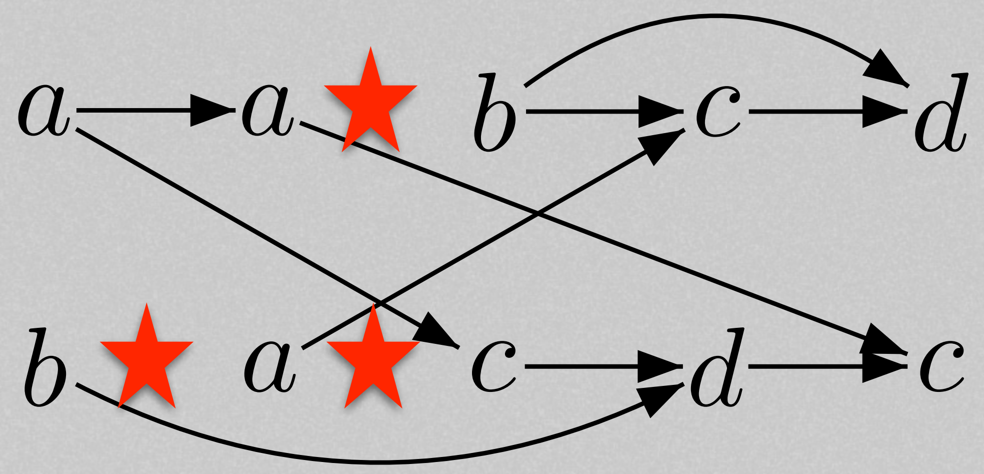
Split-width

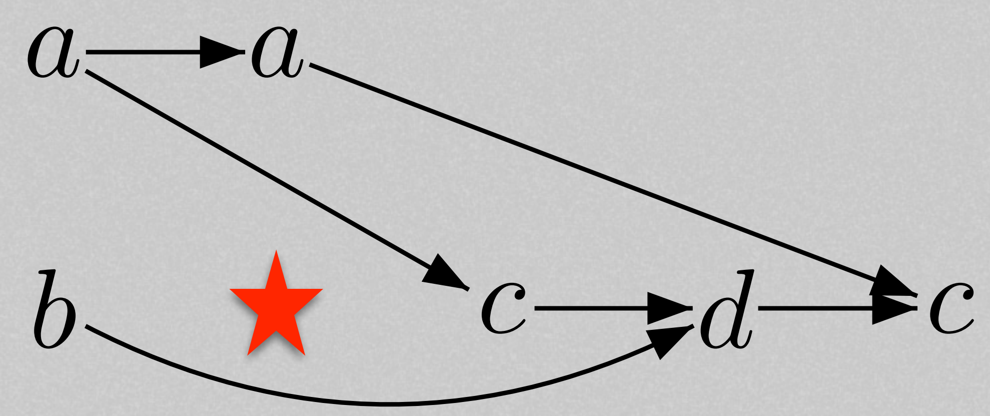
★ ★ ★

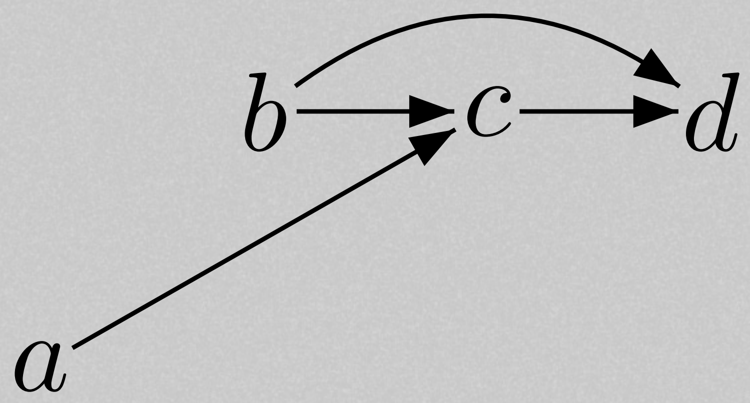
BUDGET

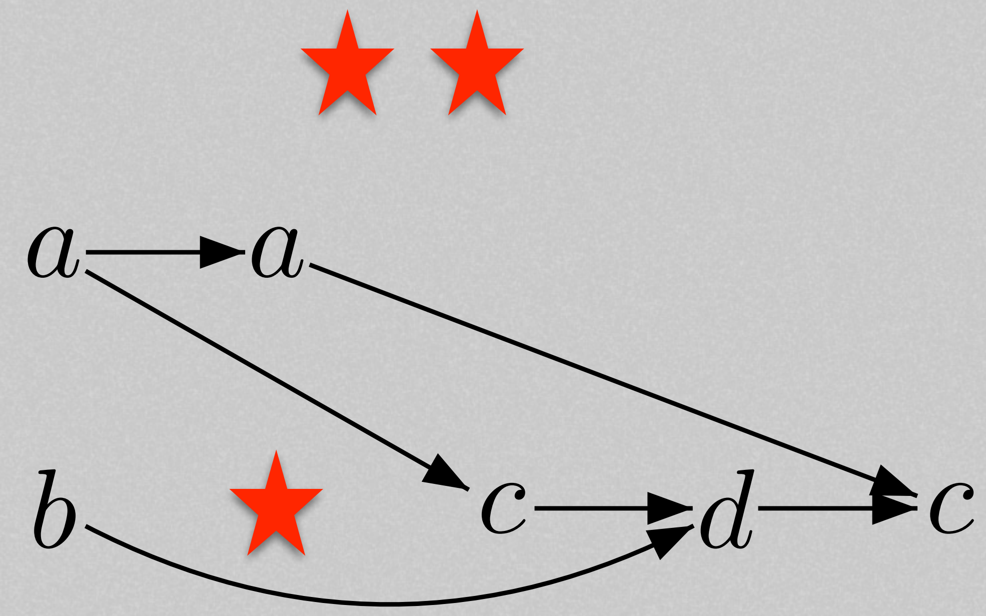


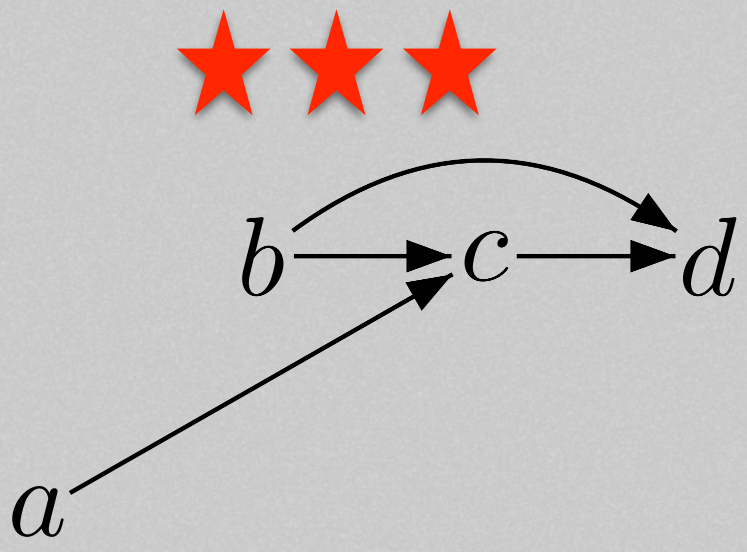

a
a
b
c
d
b
a
c
d
c


a a b c d
b a c d c

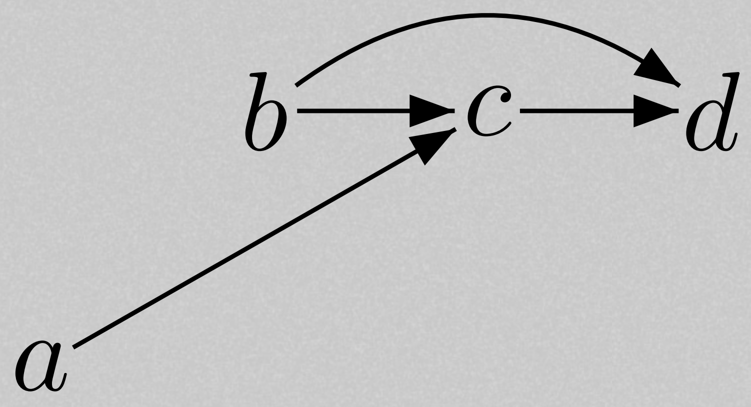

a
a
b
c
d
c

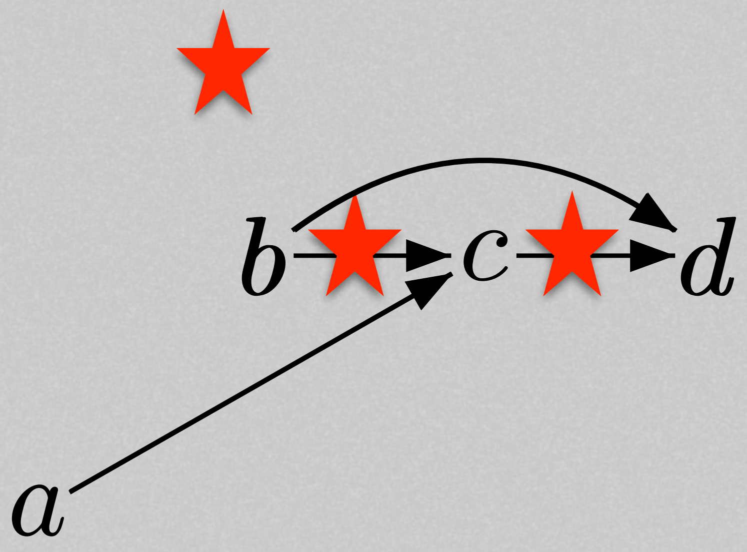

b
c
d
a

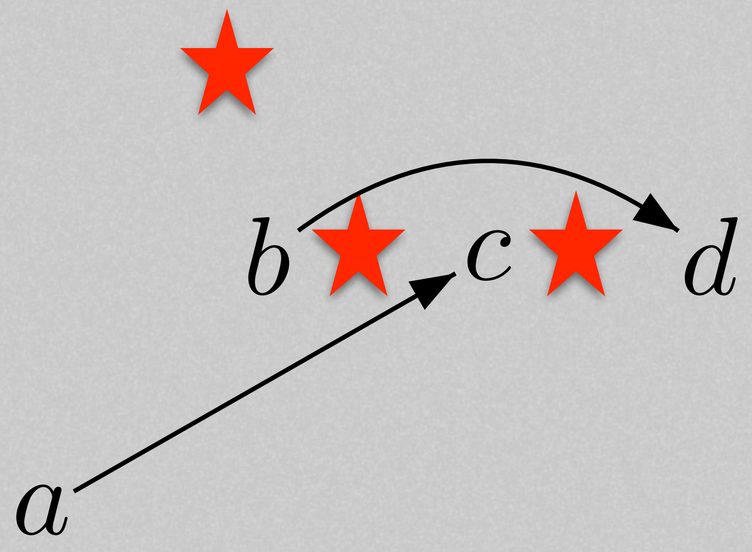

★ ★
a
a
b
★
c
d
c

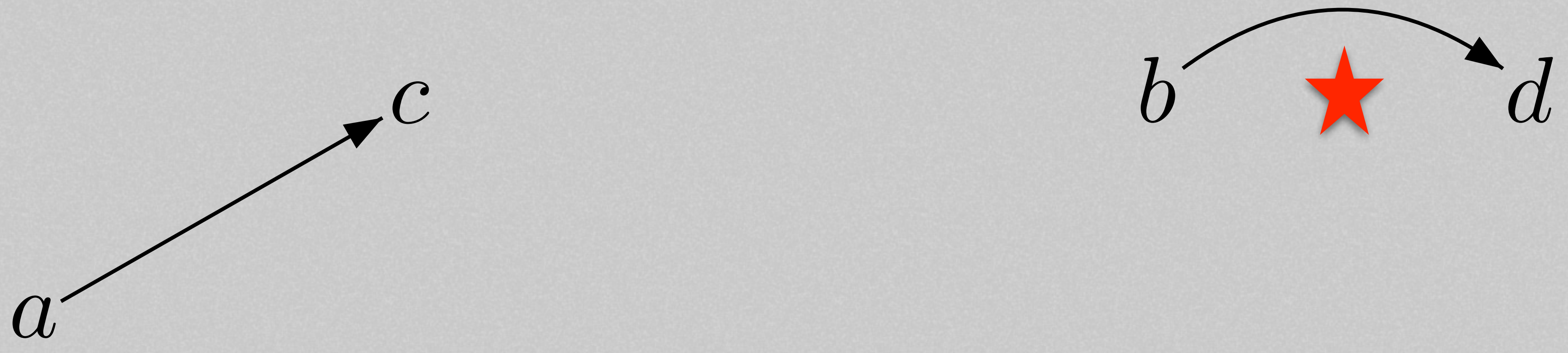

★ ★ ★
b
c
d
a

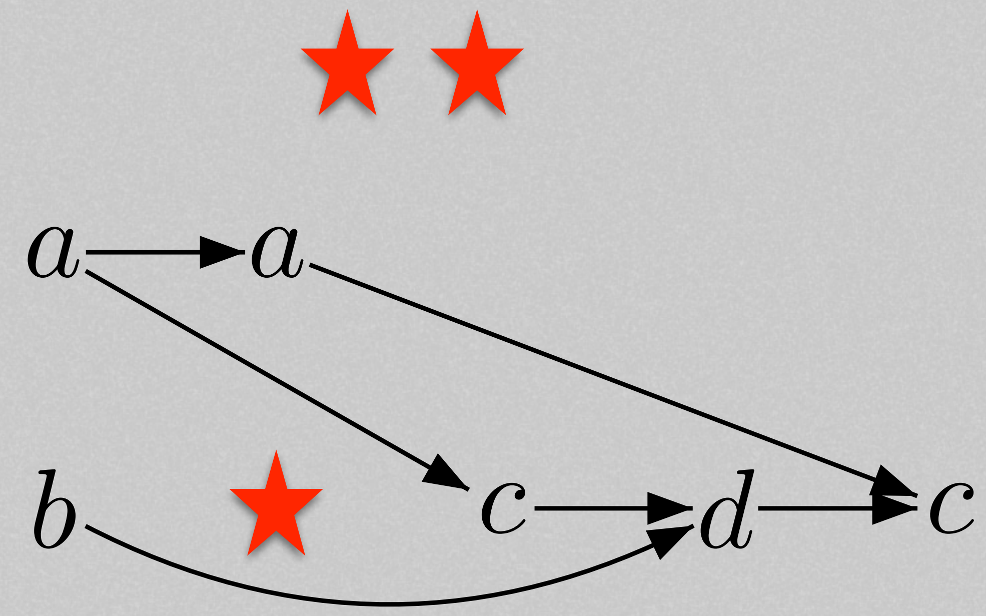
★★★

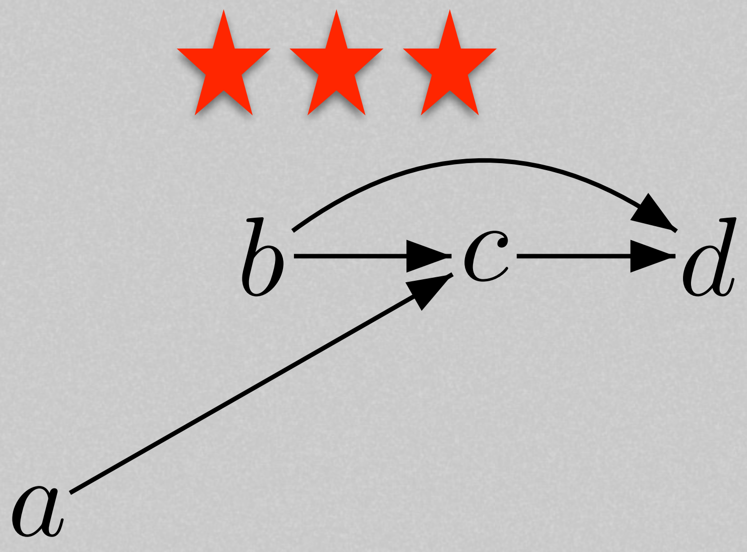

b
c
d
a

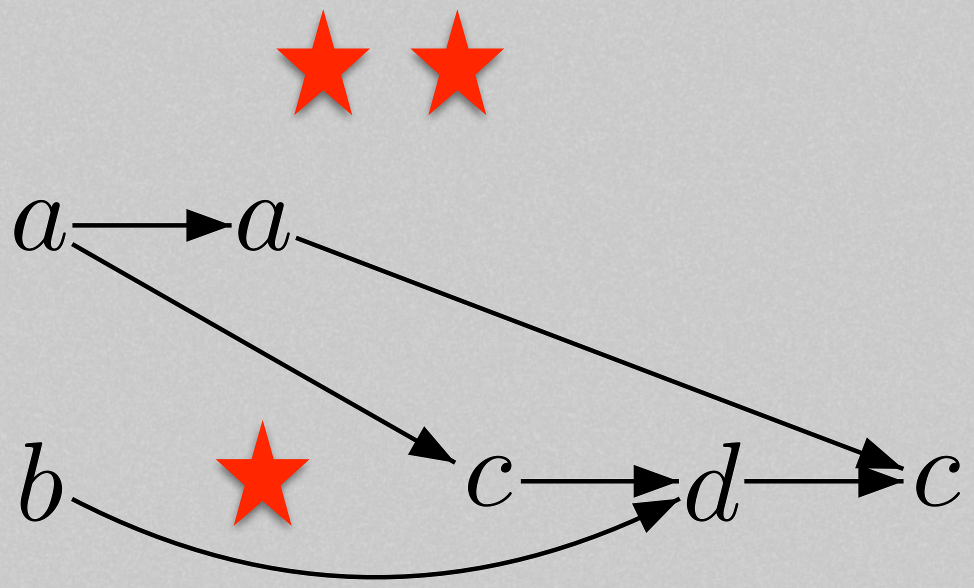

b
c
d
a

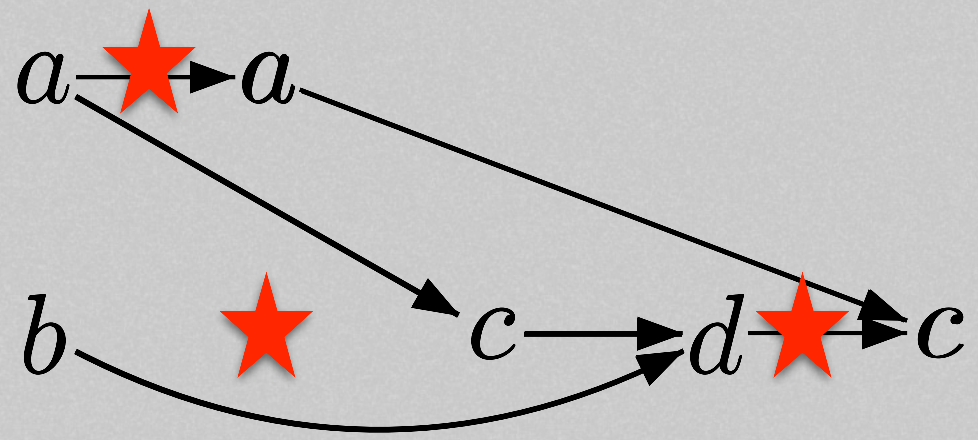

b
c
d
a

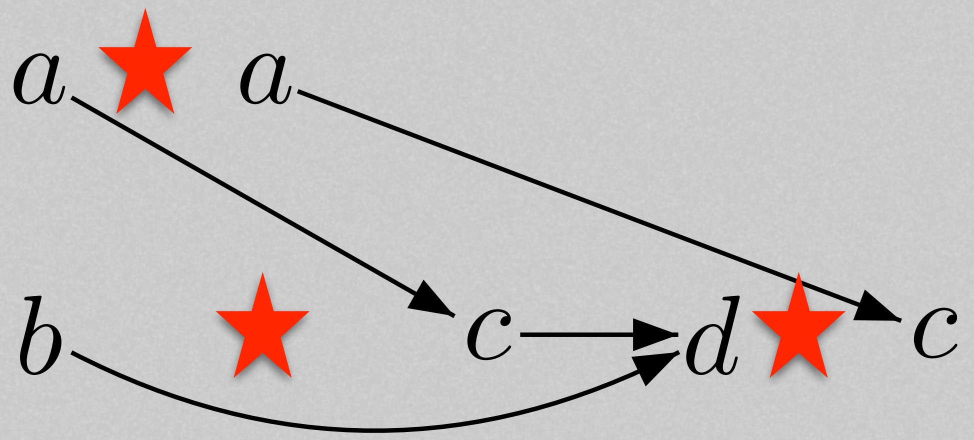

a
c
b
d


★★
a
a
b
★
c
d
c

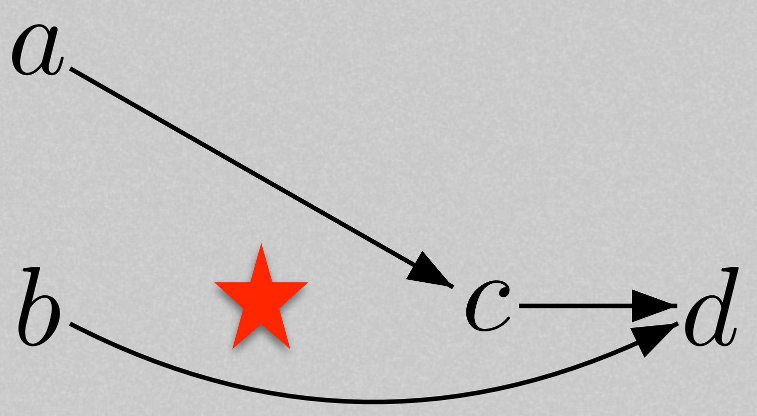

★★★
b
c
d
a

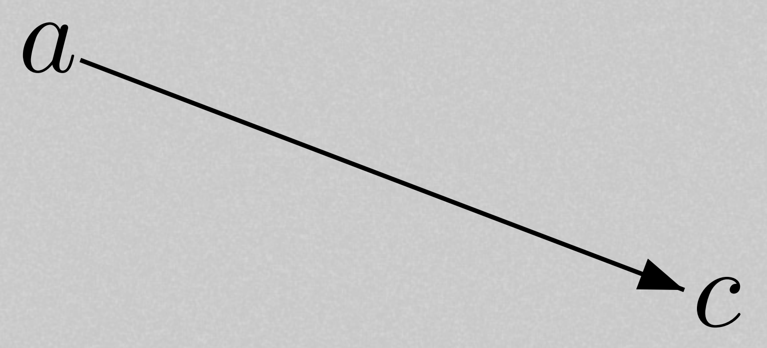

a
a
b
c
d
c

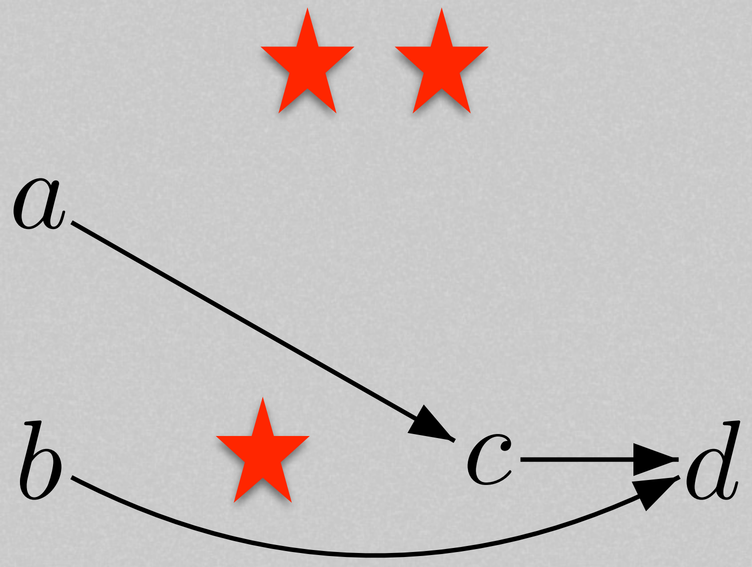

a
a
b
c
d
c

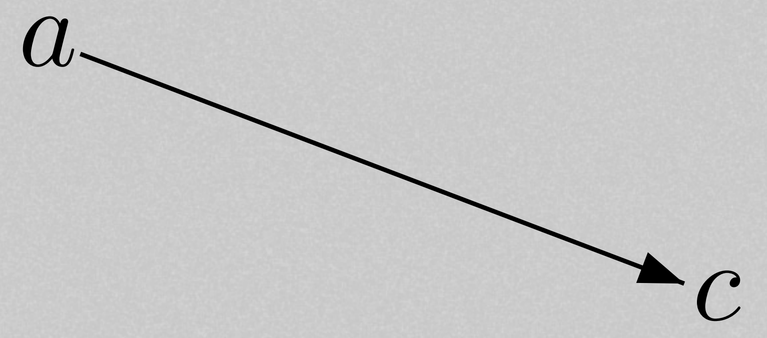


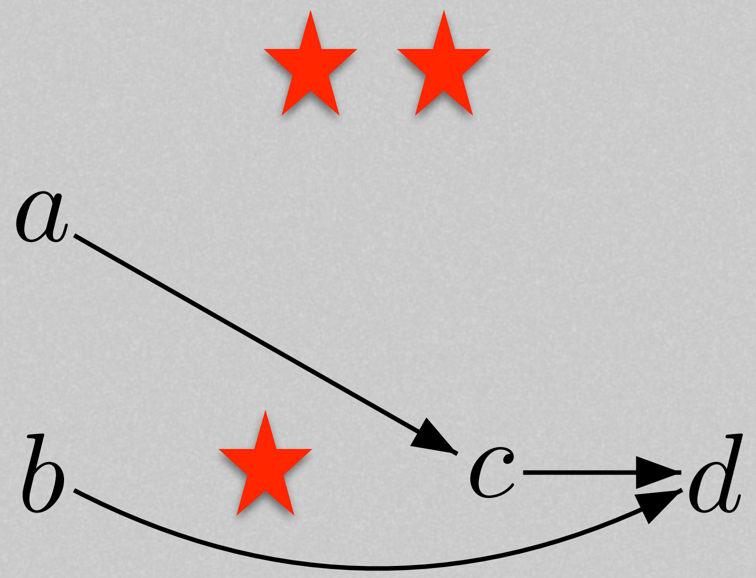
a
a
b
c
d
c

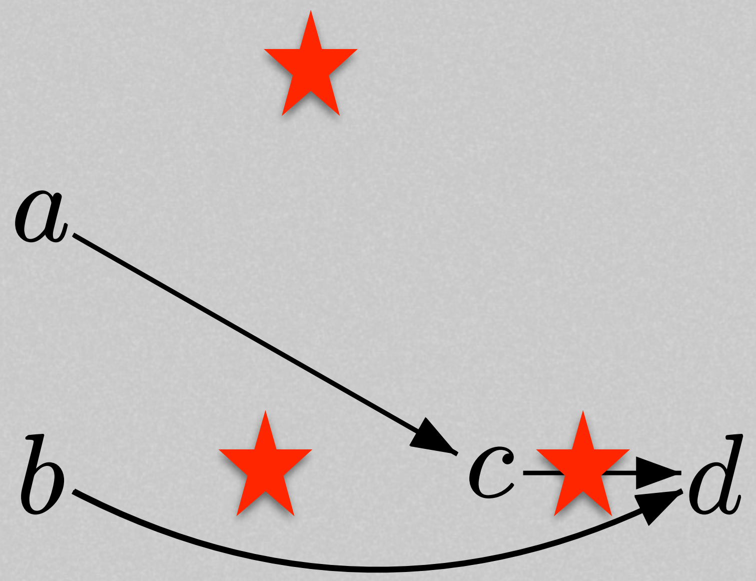

a
b
c
d


a
c

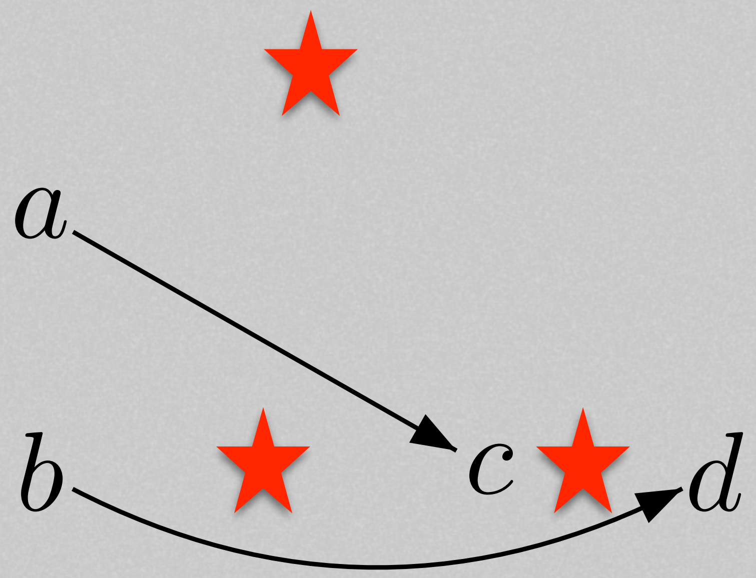

a
b
c
d


a
c


a
b
c
d

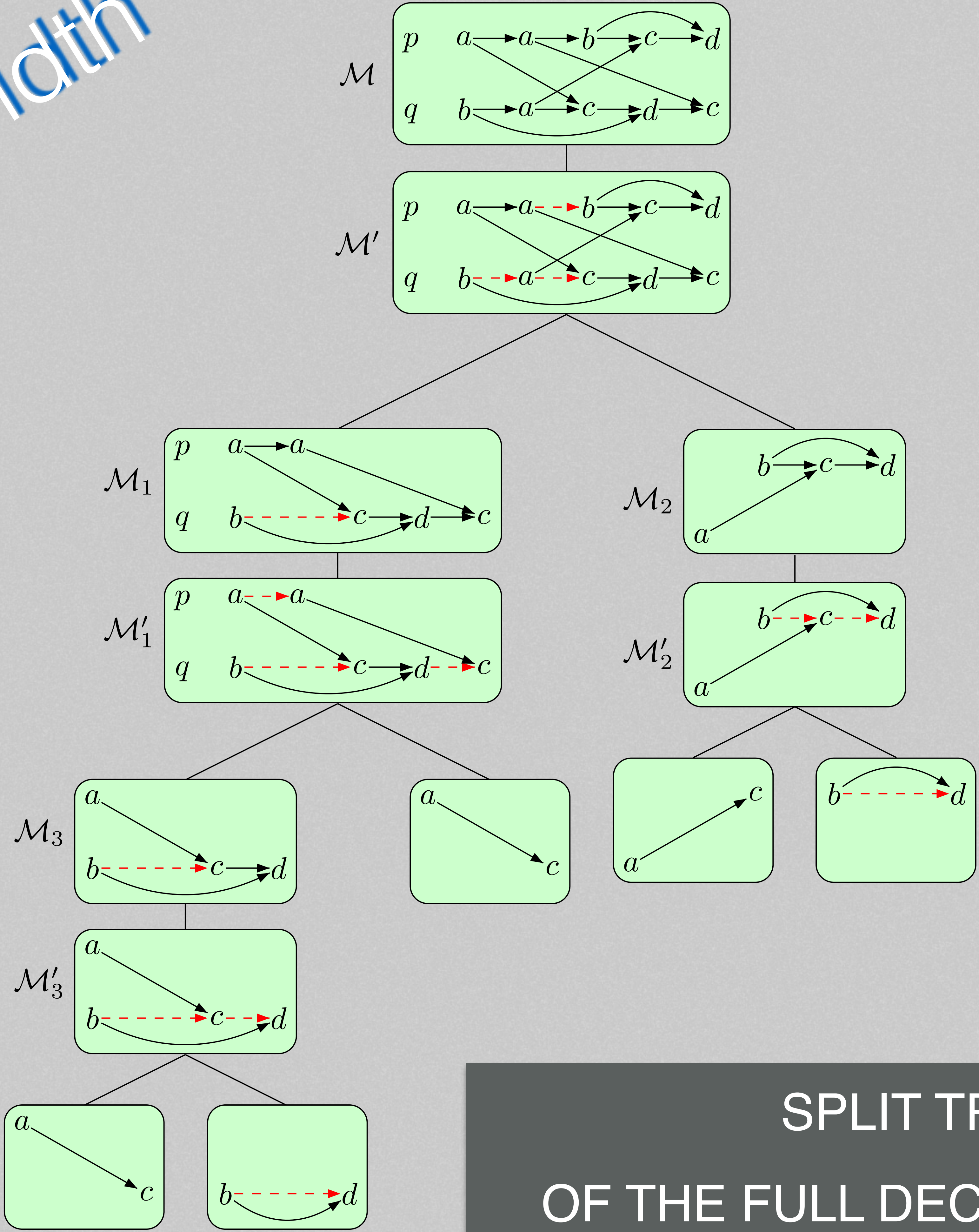


a
b
c
d


a
b
c
d

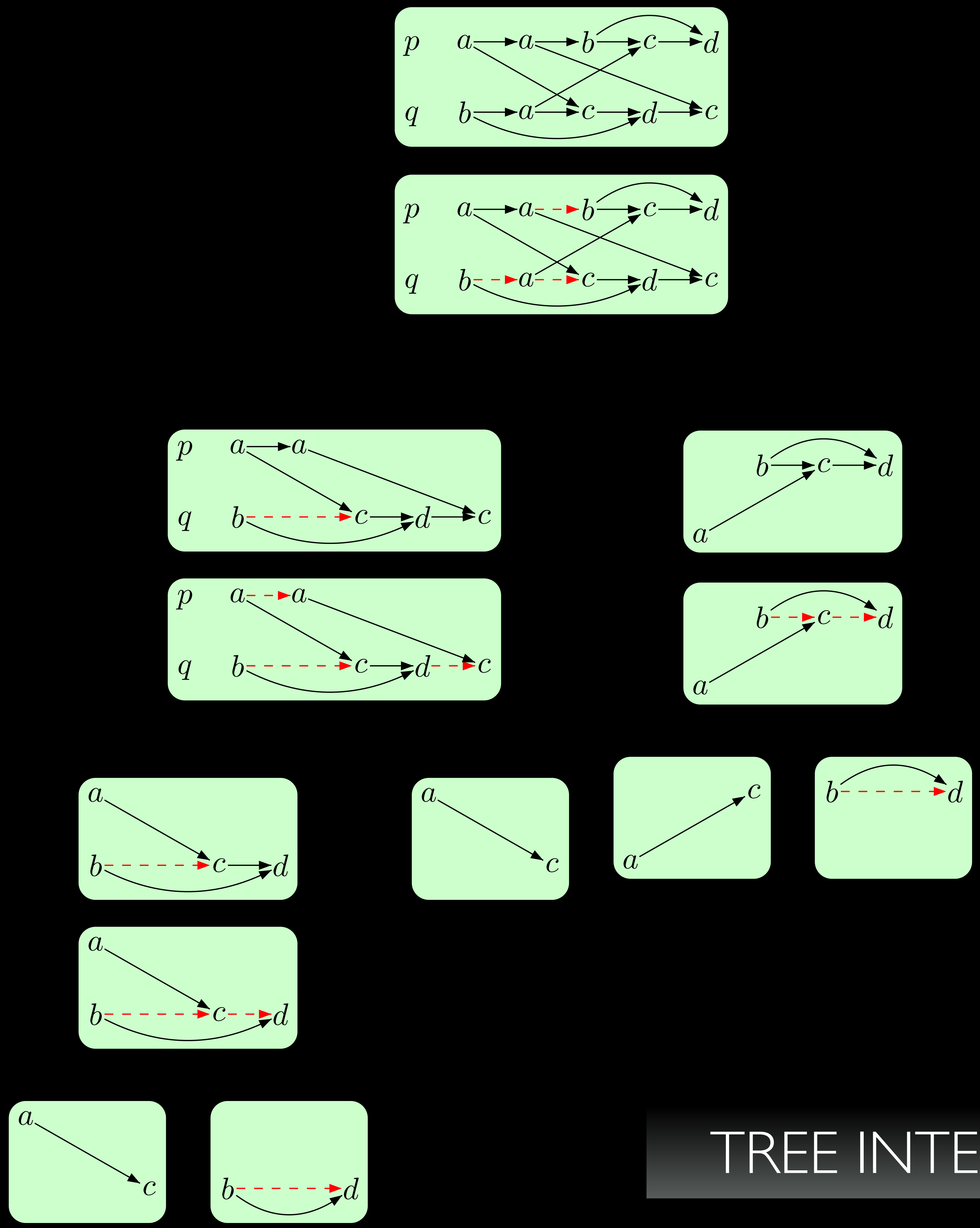
a
c
b
d

Split-width

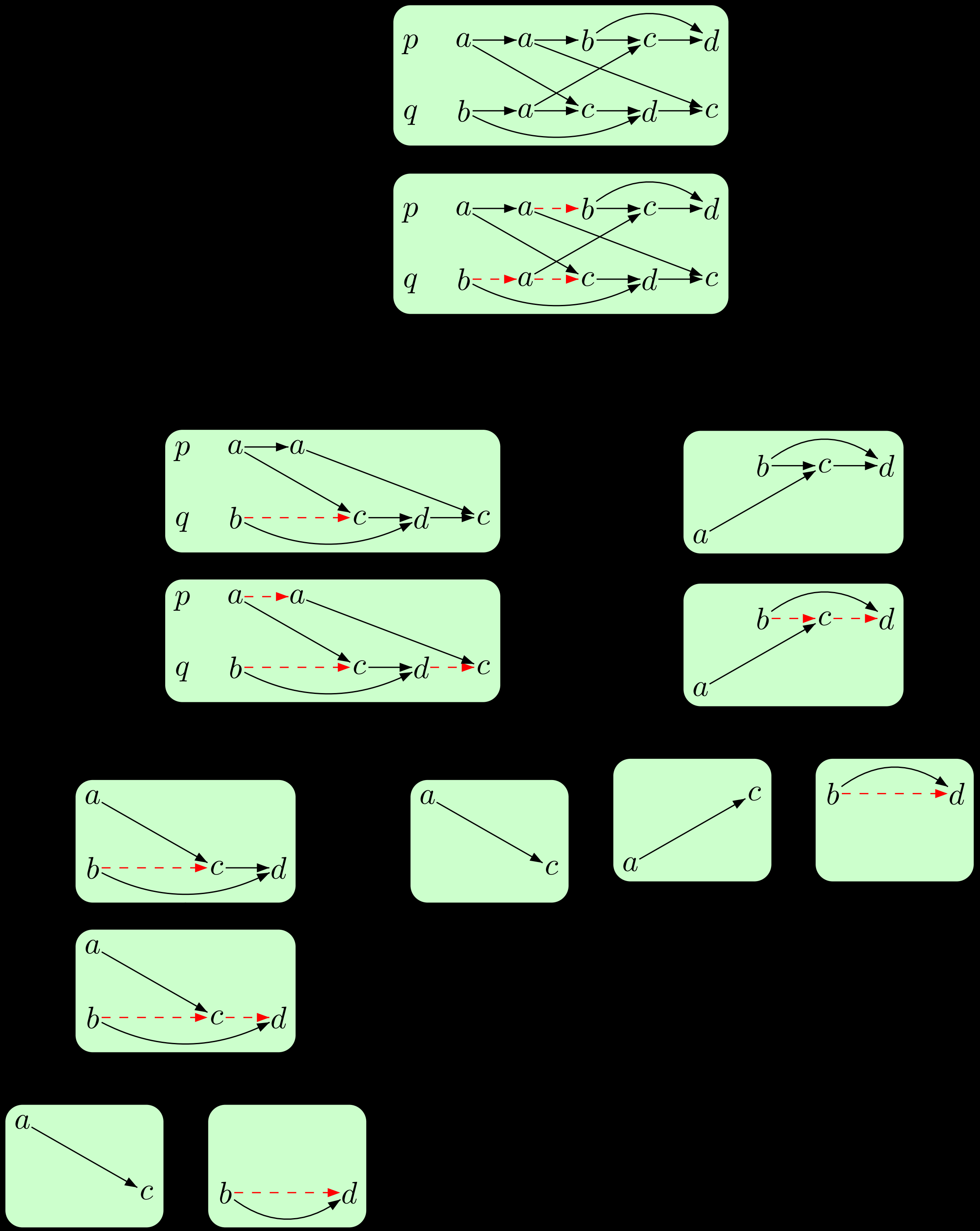


SPLIT TREE

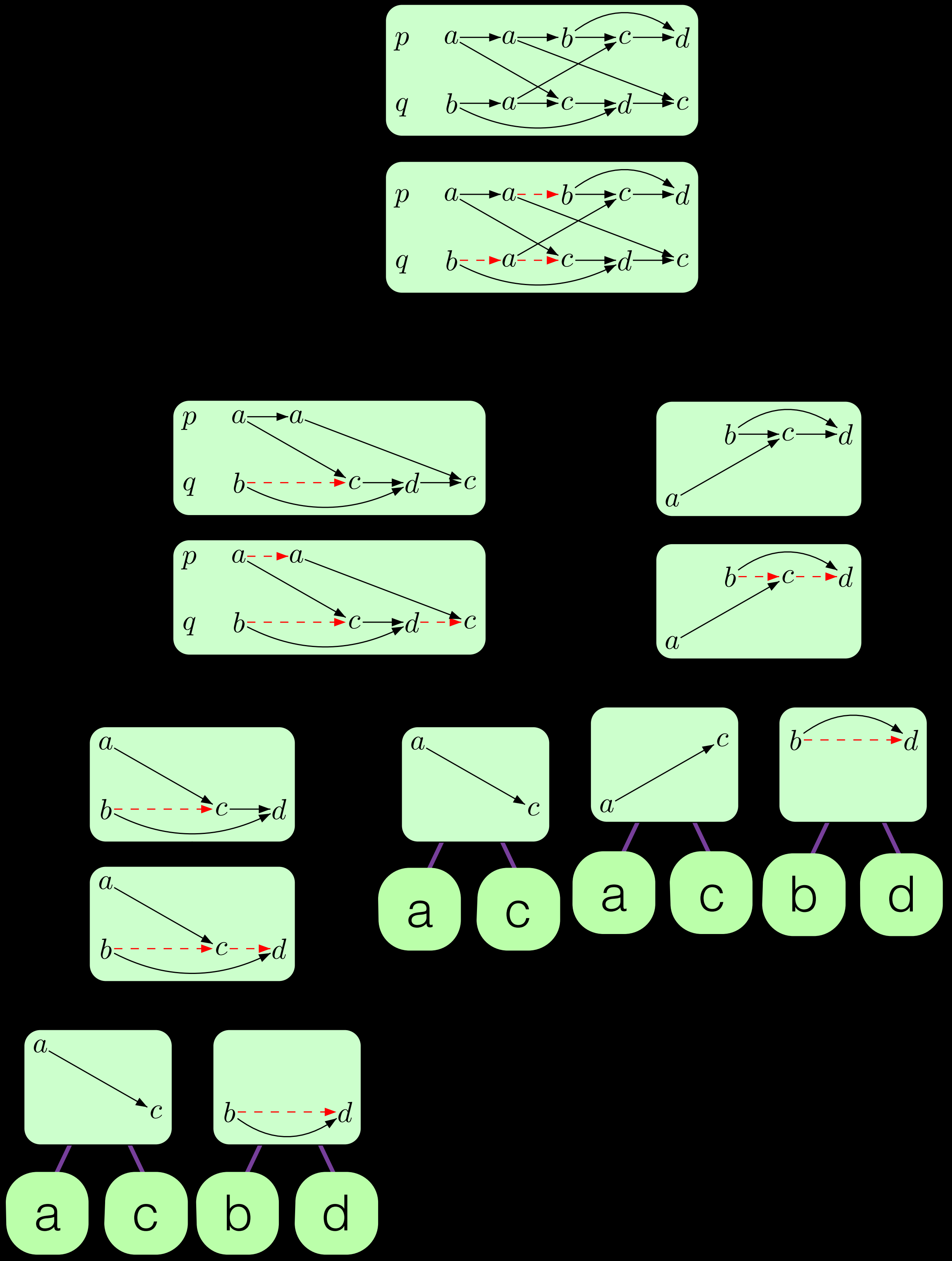
OF THE FULL DECOMPOSITION



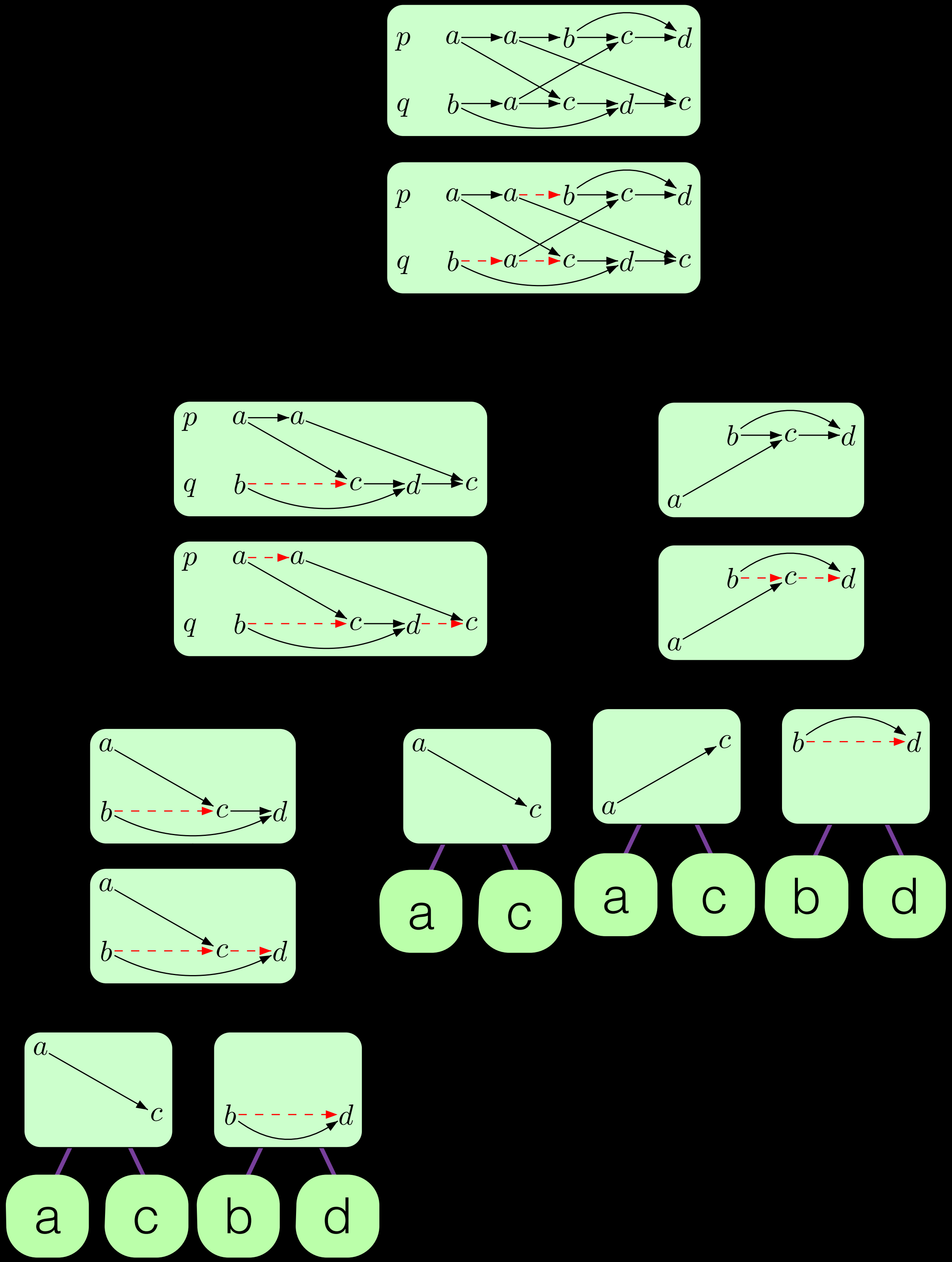
TREE INTERPRETATION

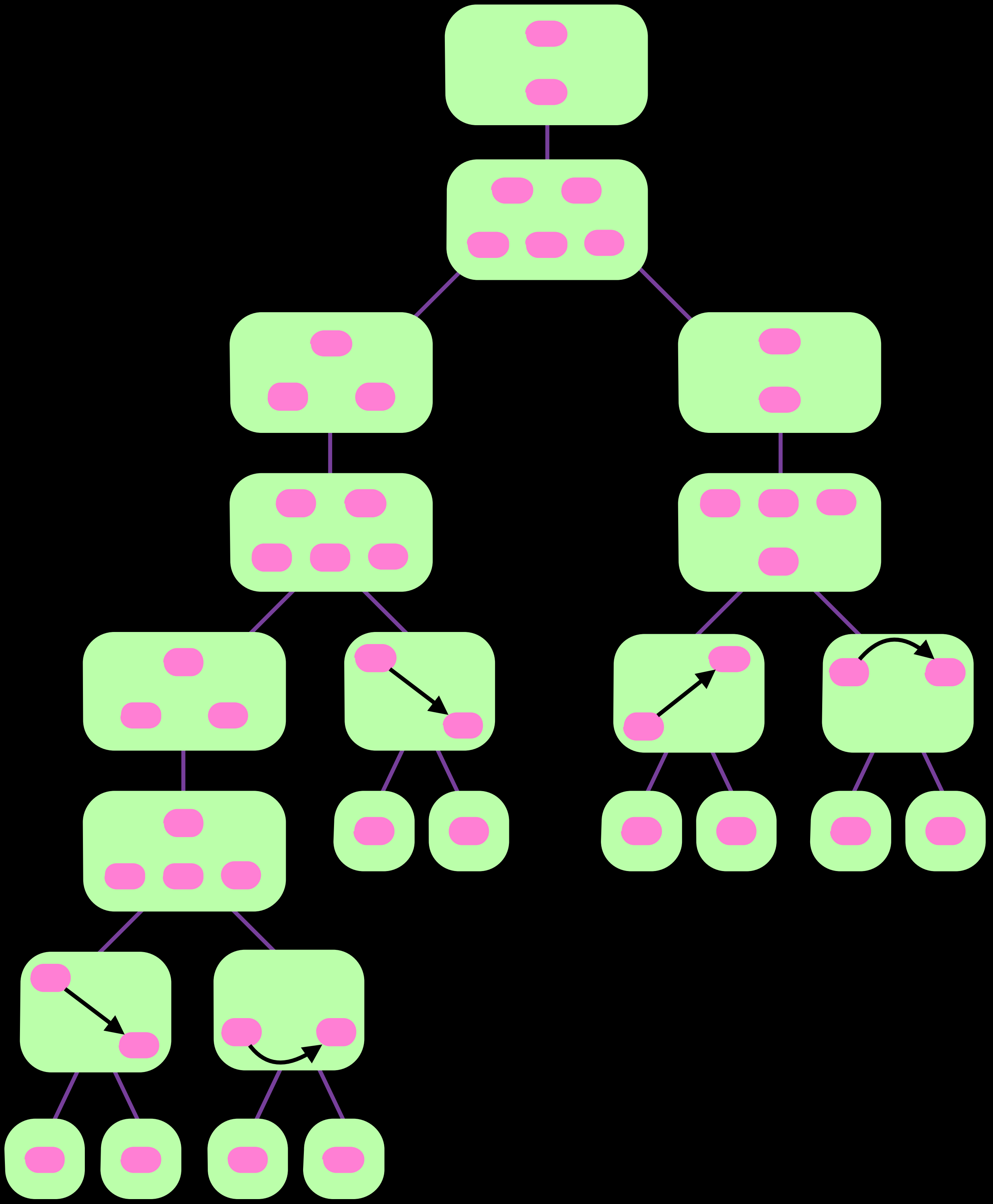


TREE INTERPRETATION

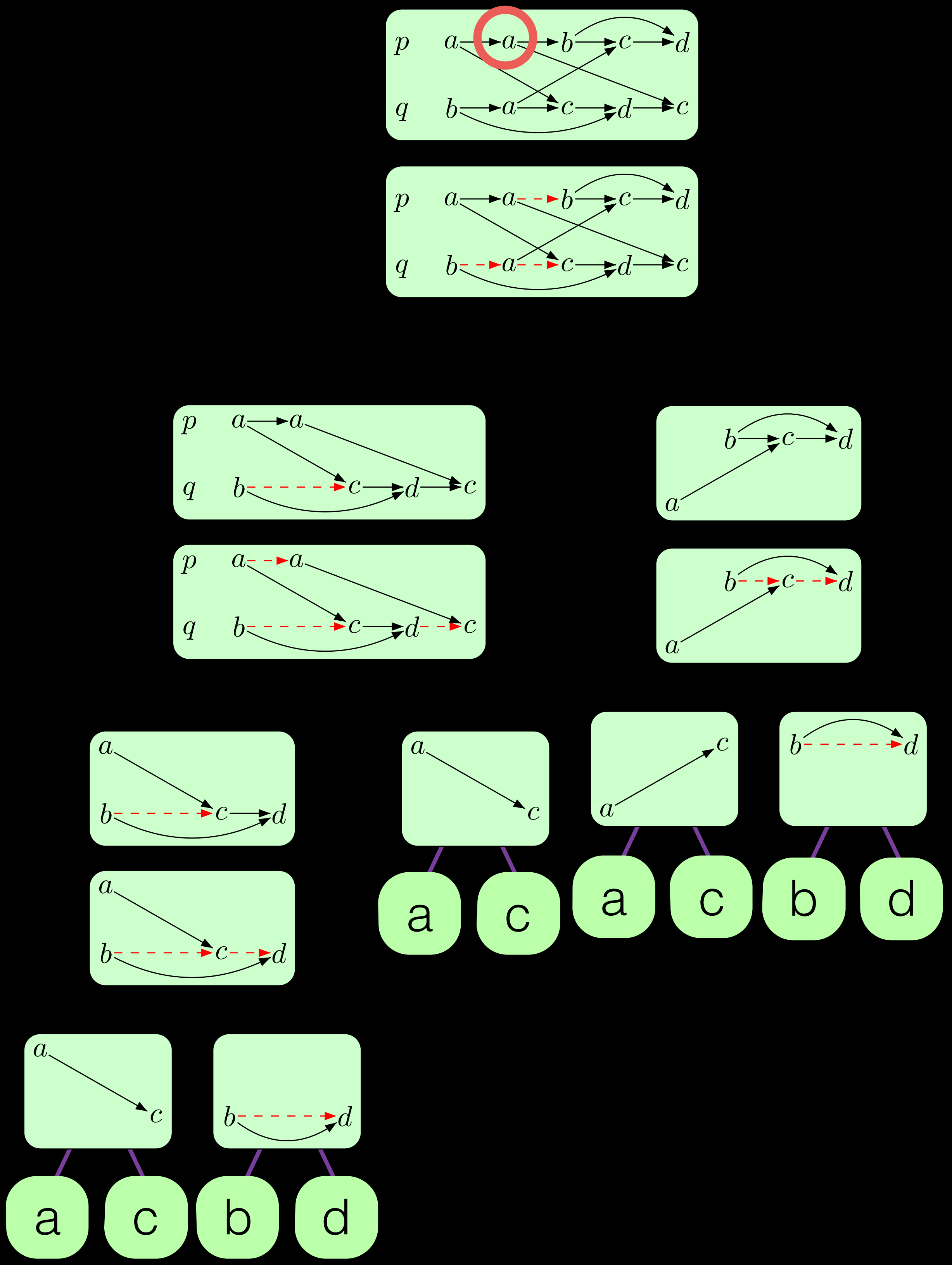


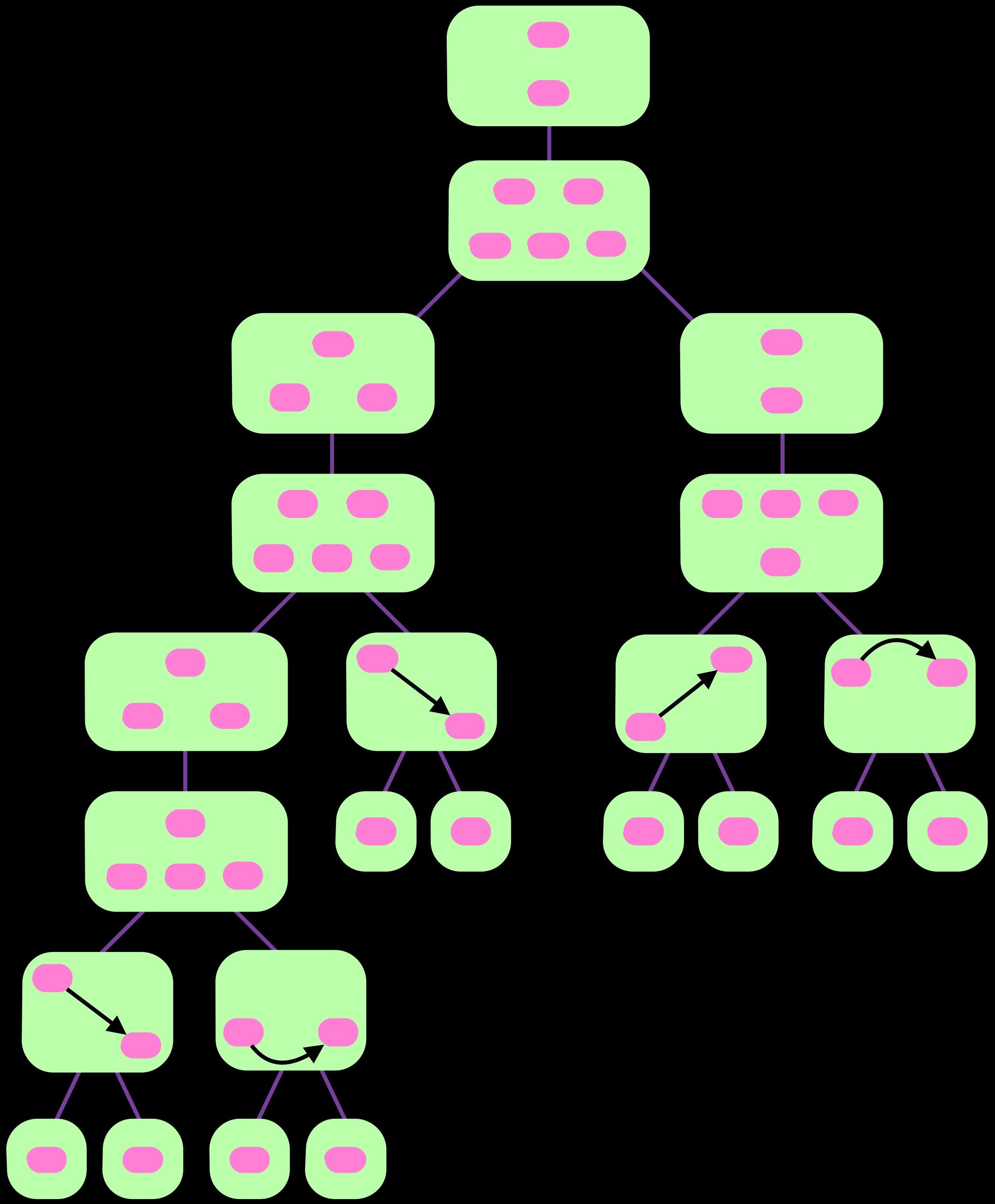
TREE INTERPRETATION



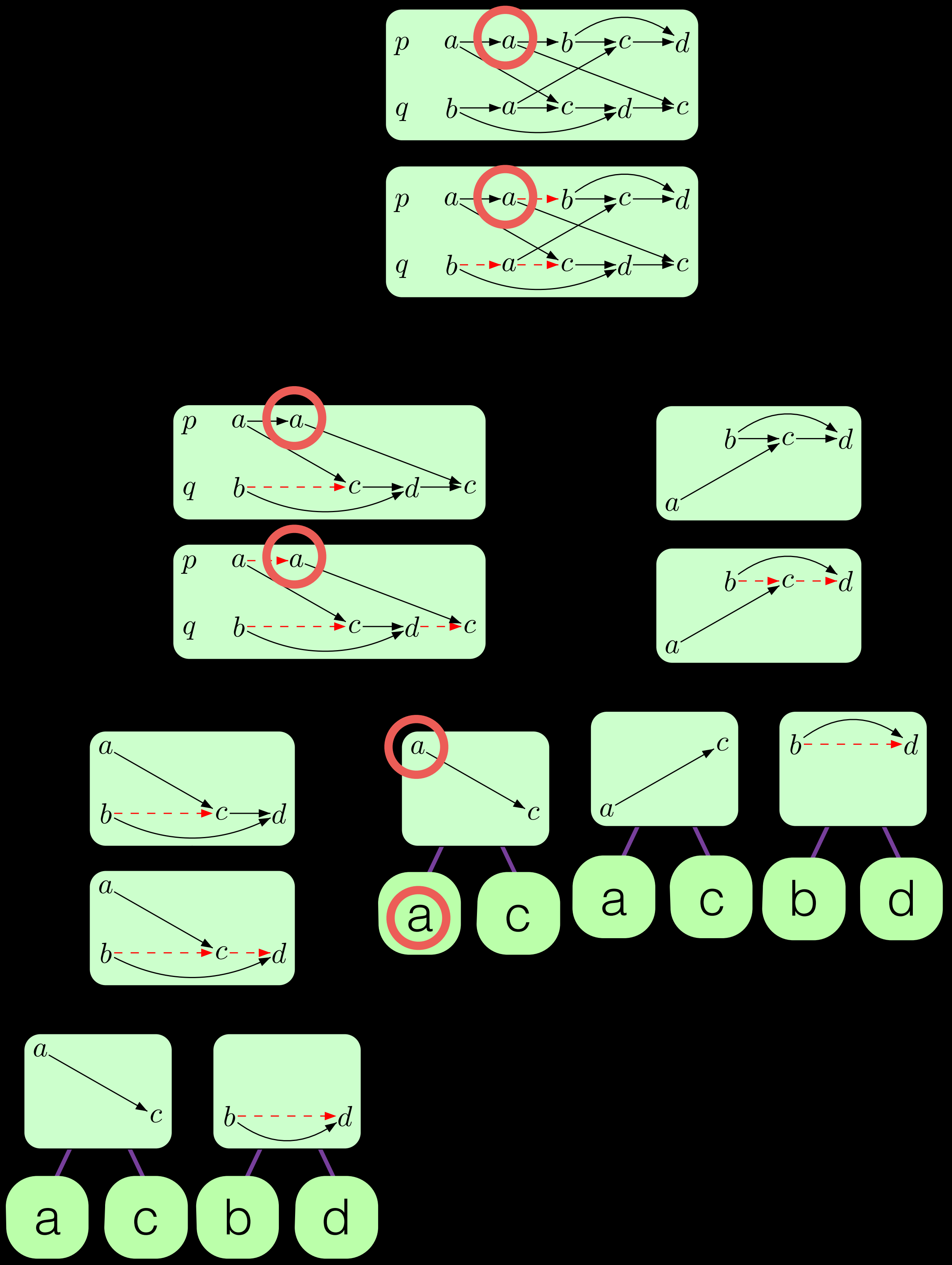


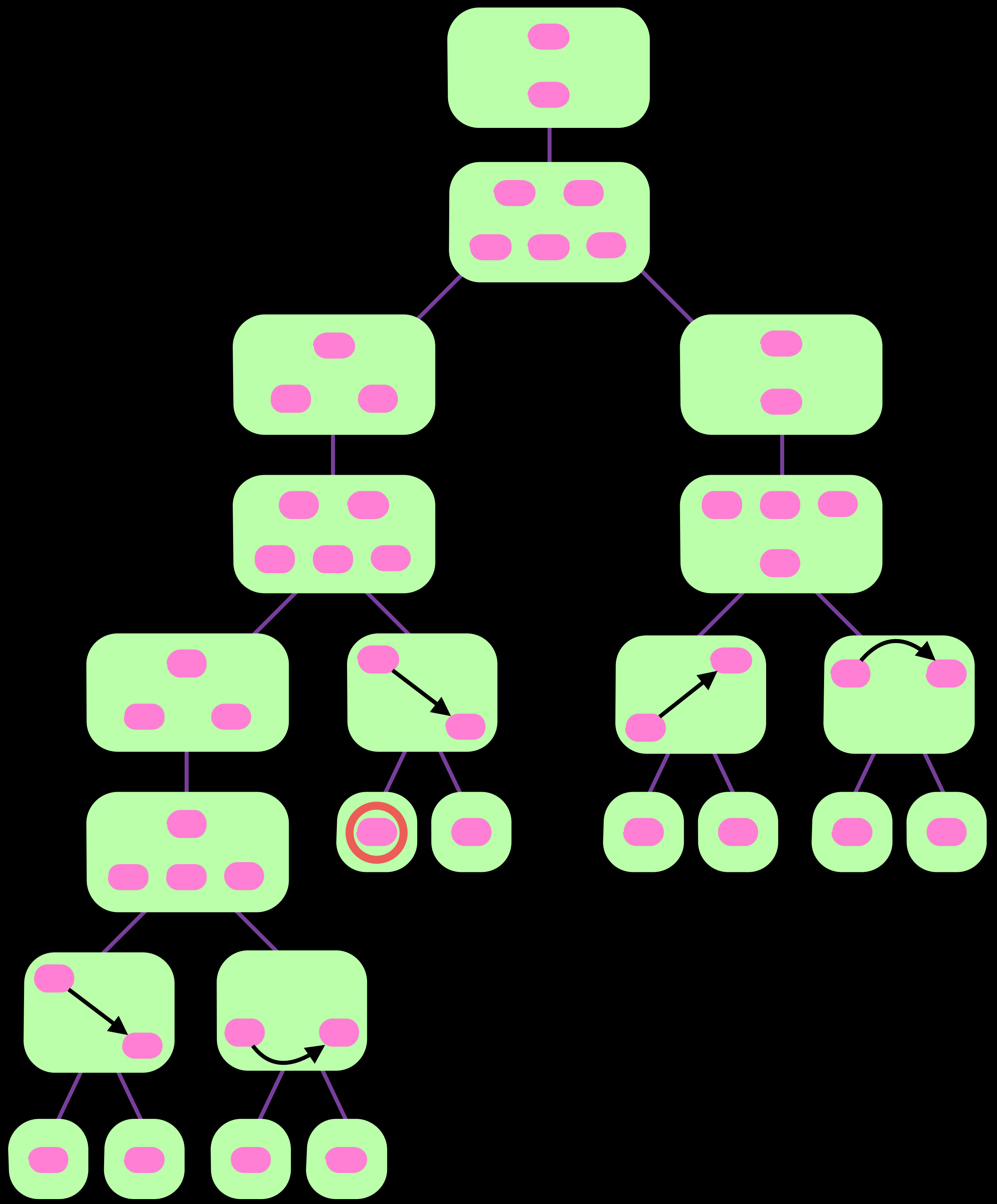
TREE INTERPRETATION



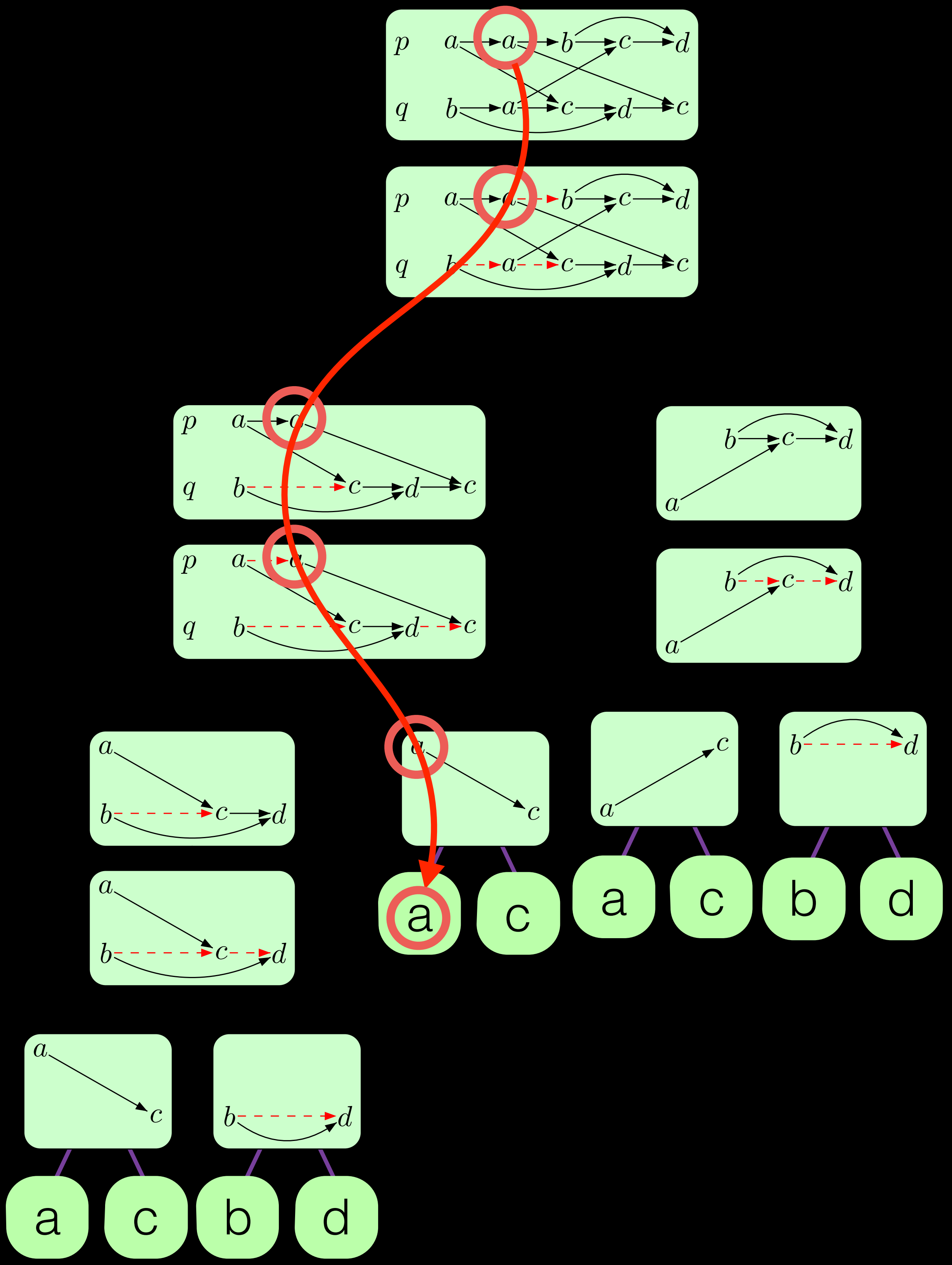


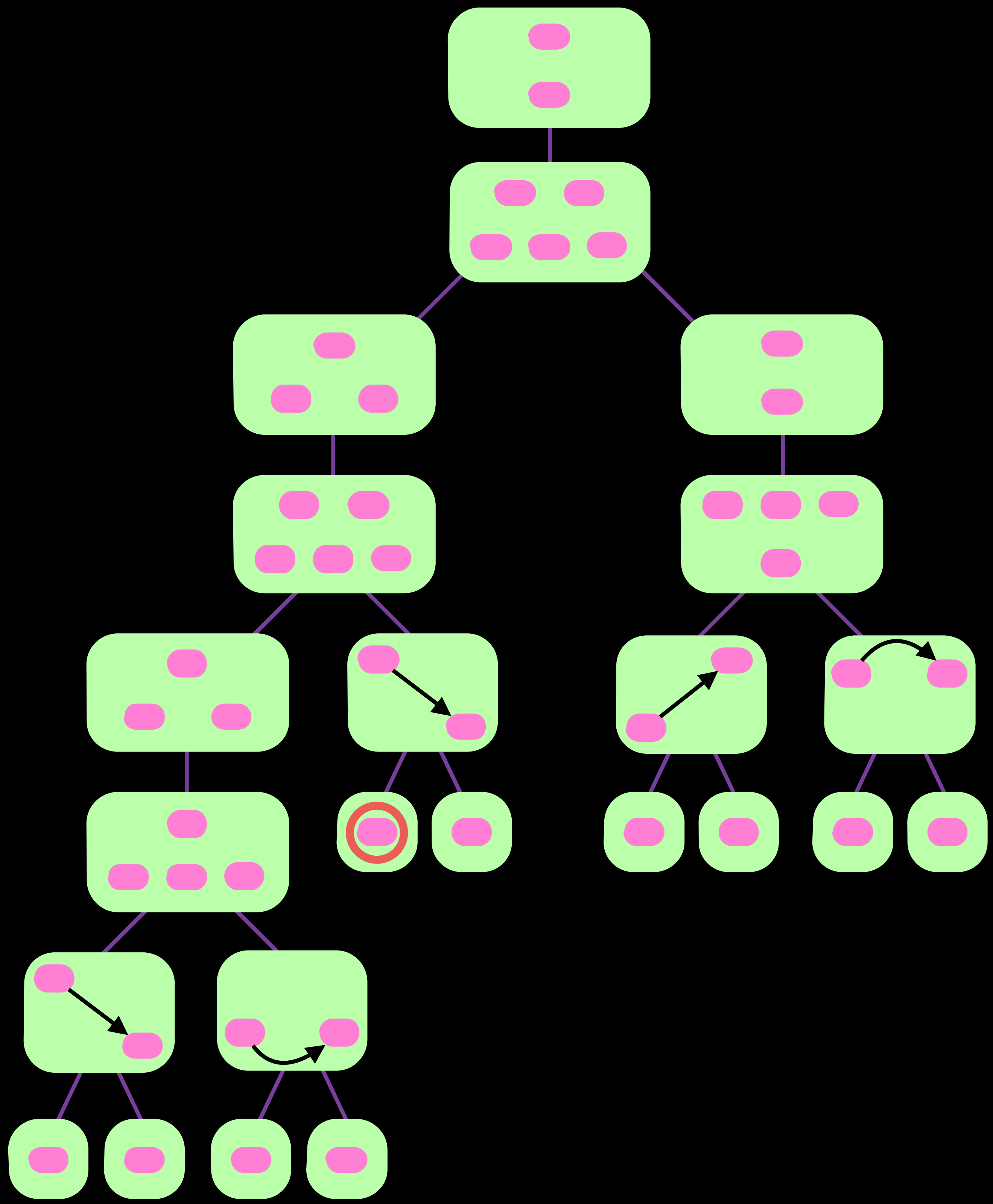
TREE INTERPRETATION



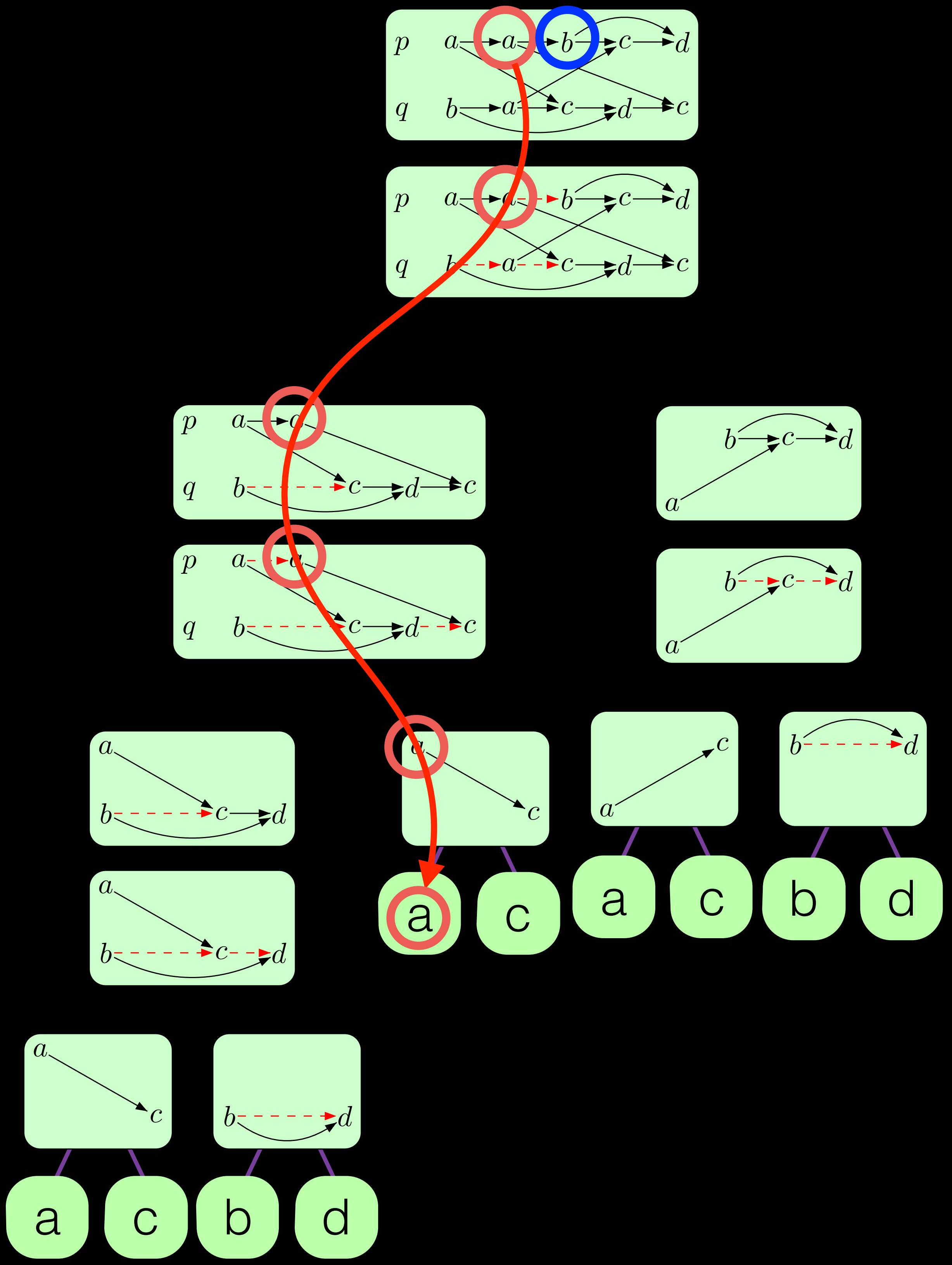


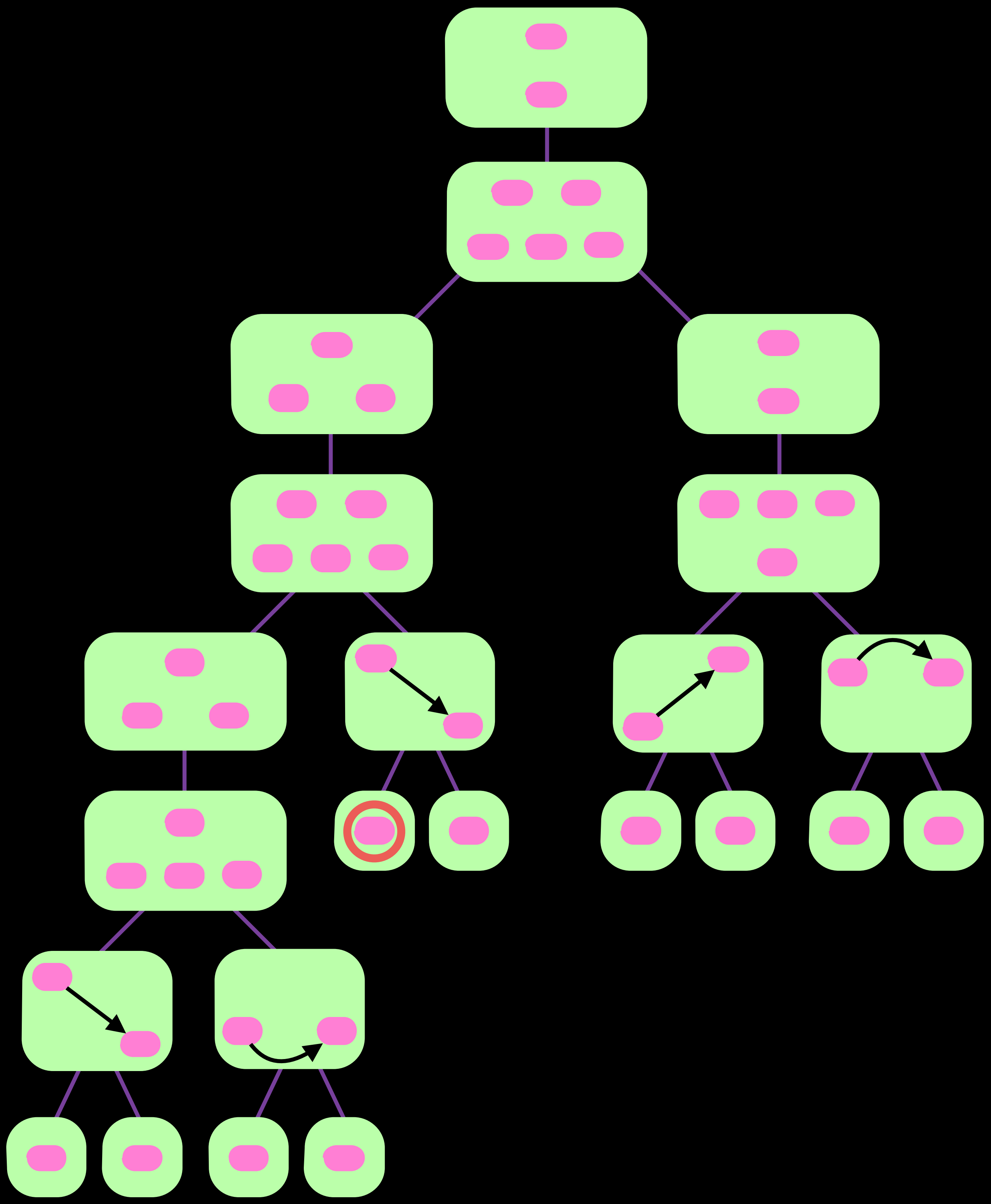
TREE INTERPRETATION



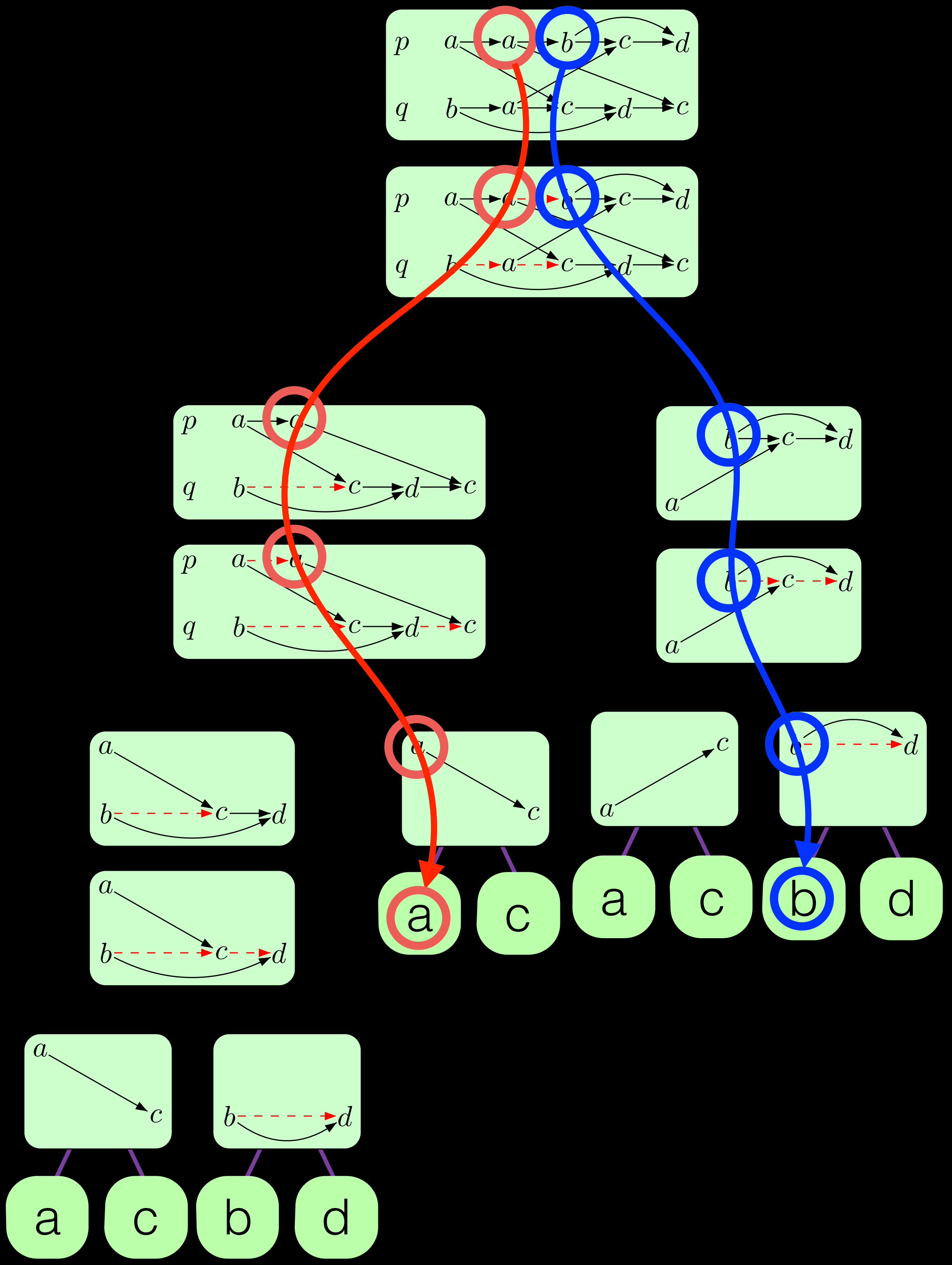


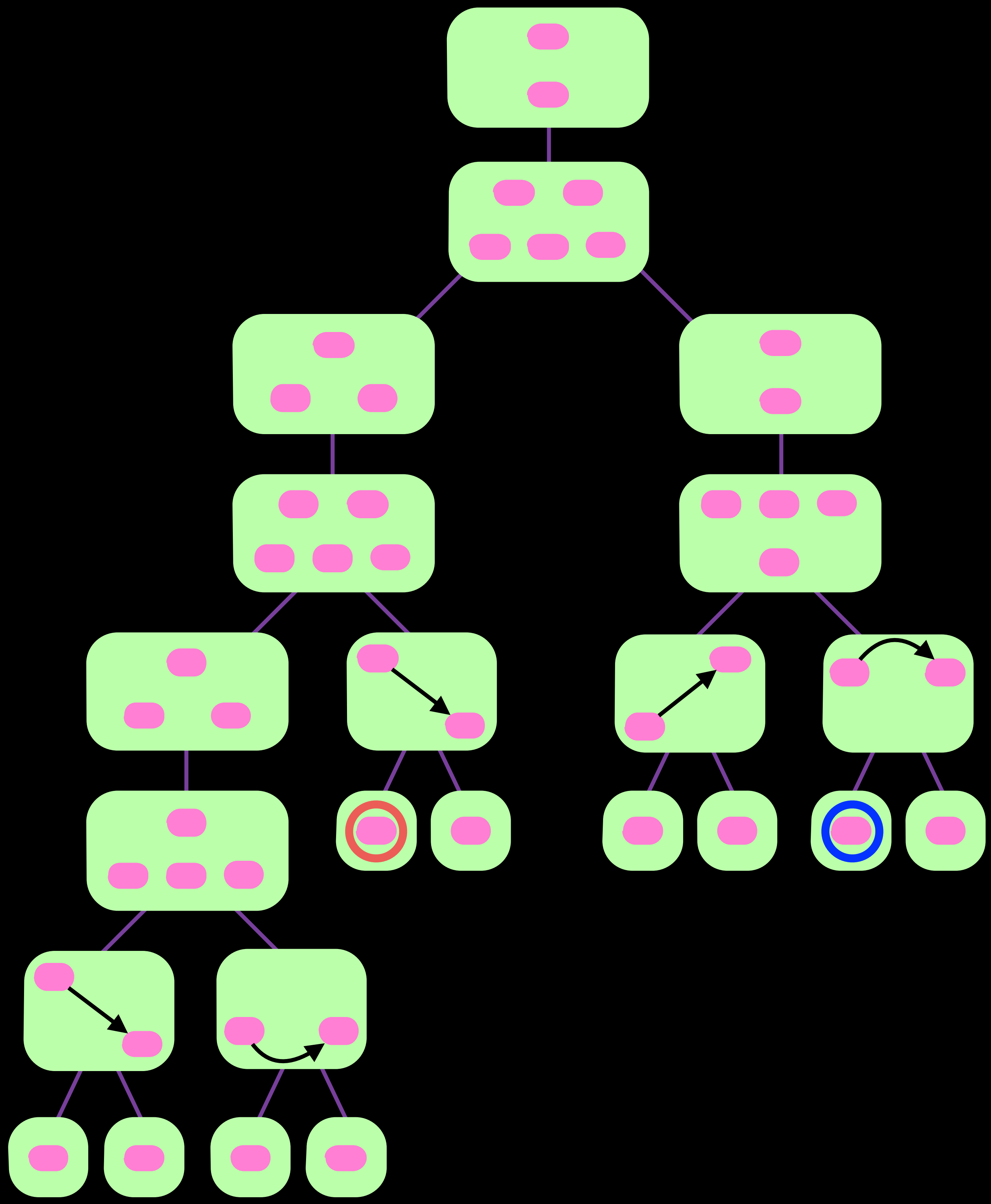
TREE INTERPRETATION



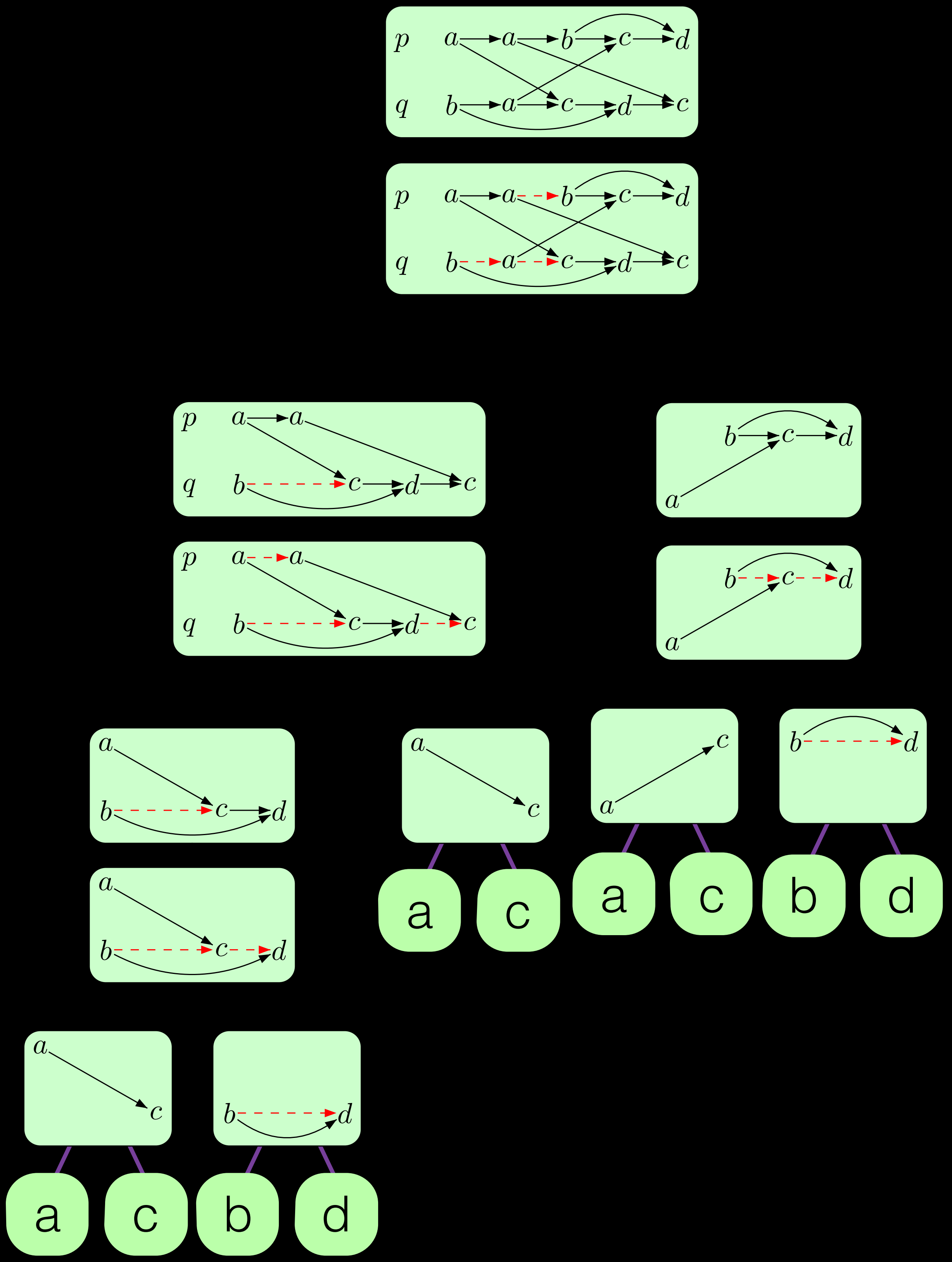


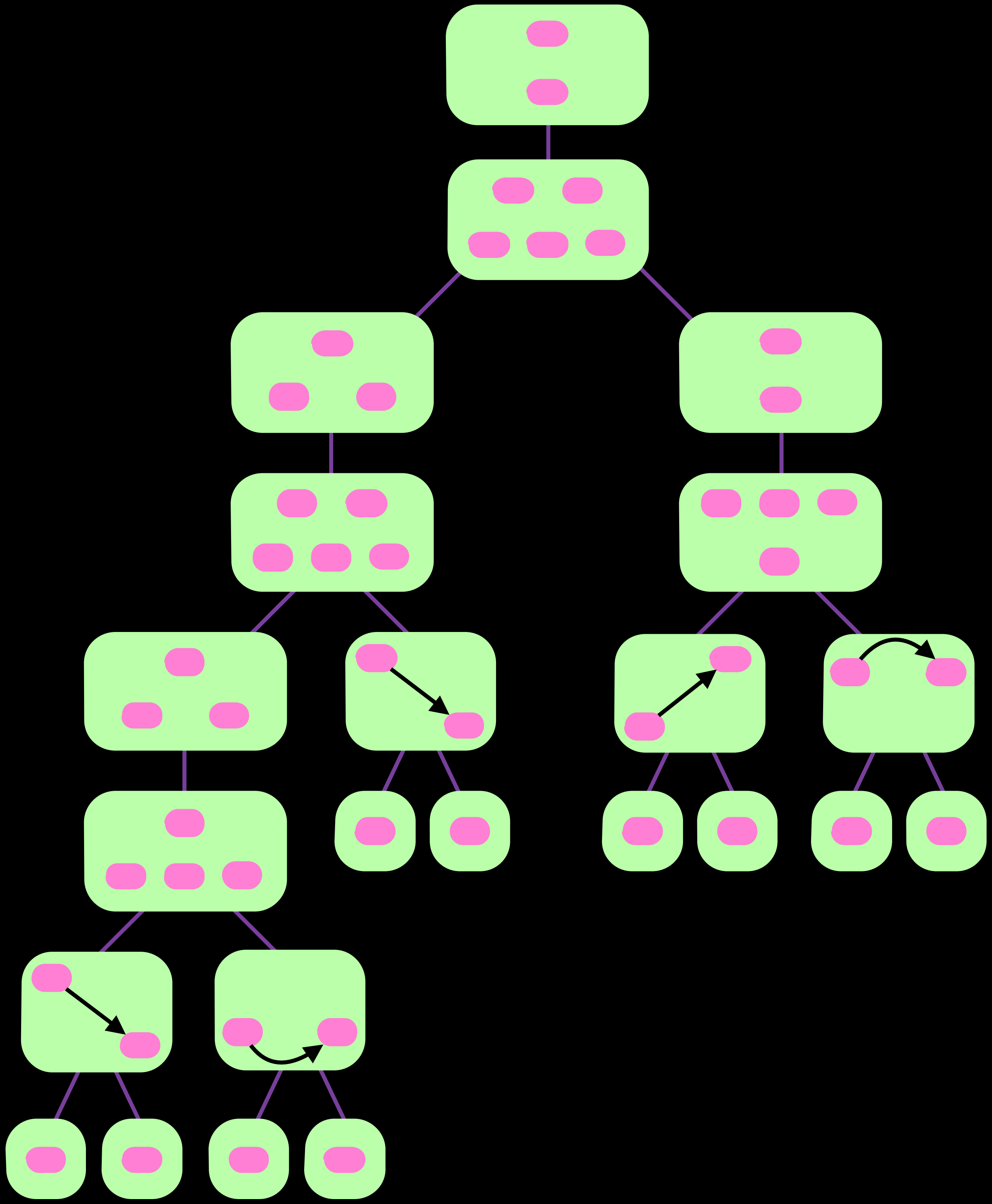
TREE INTERPRETATION



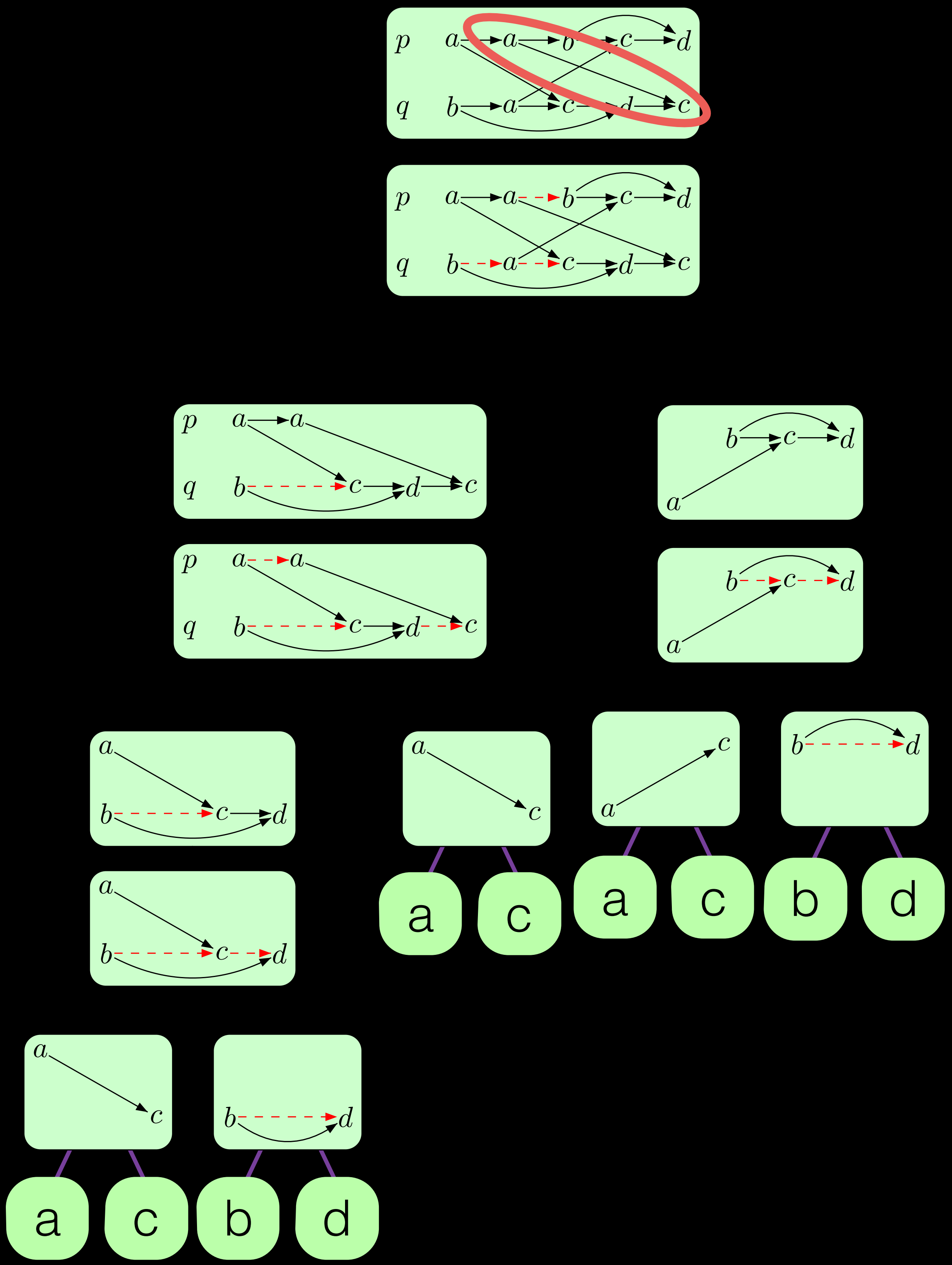


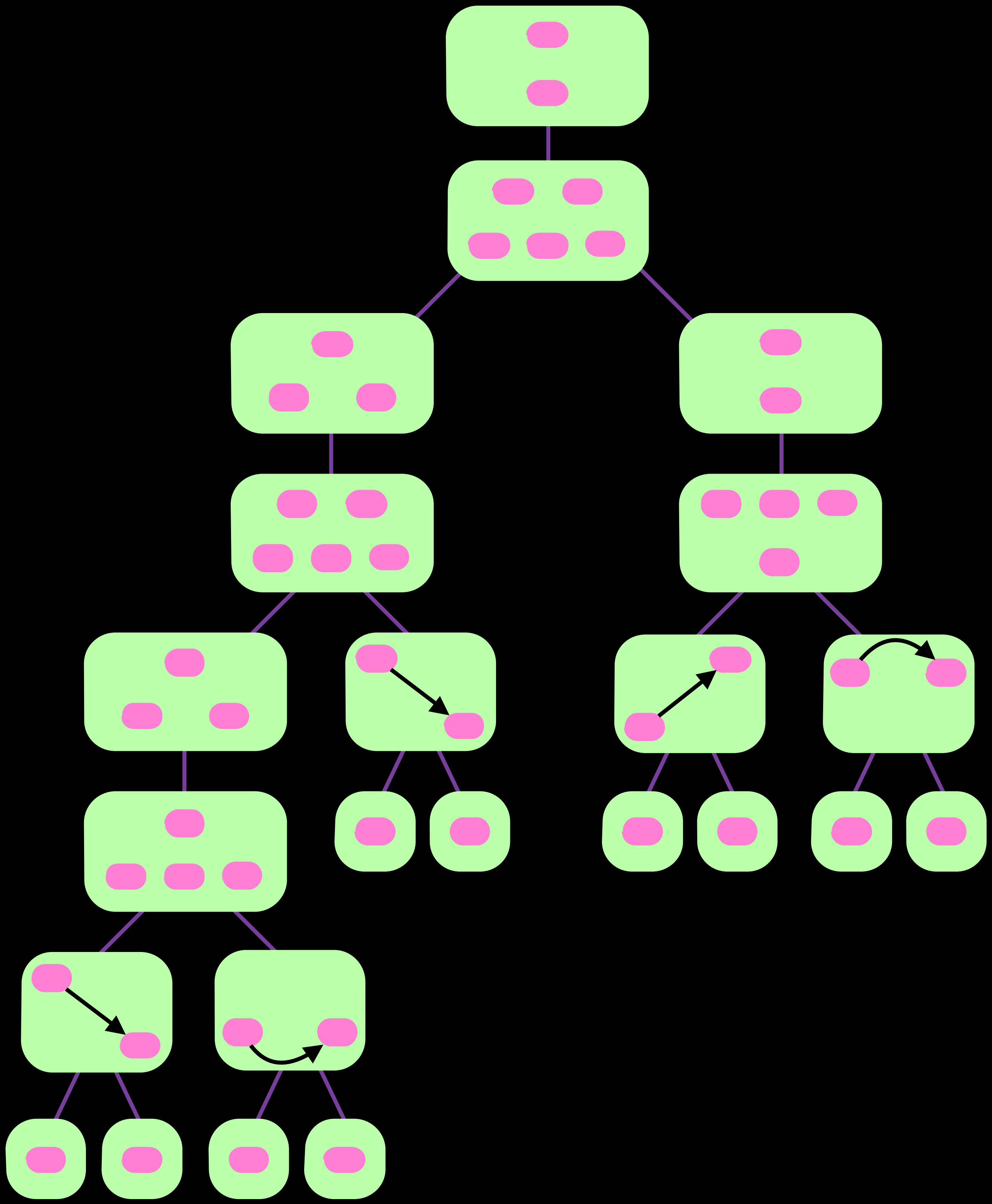
TREE INTERPRETATION



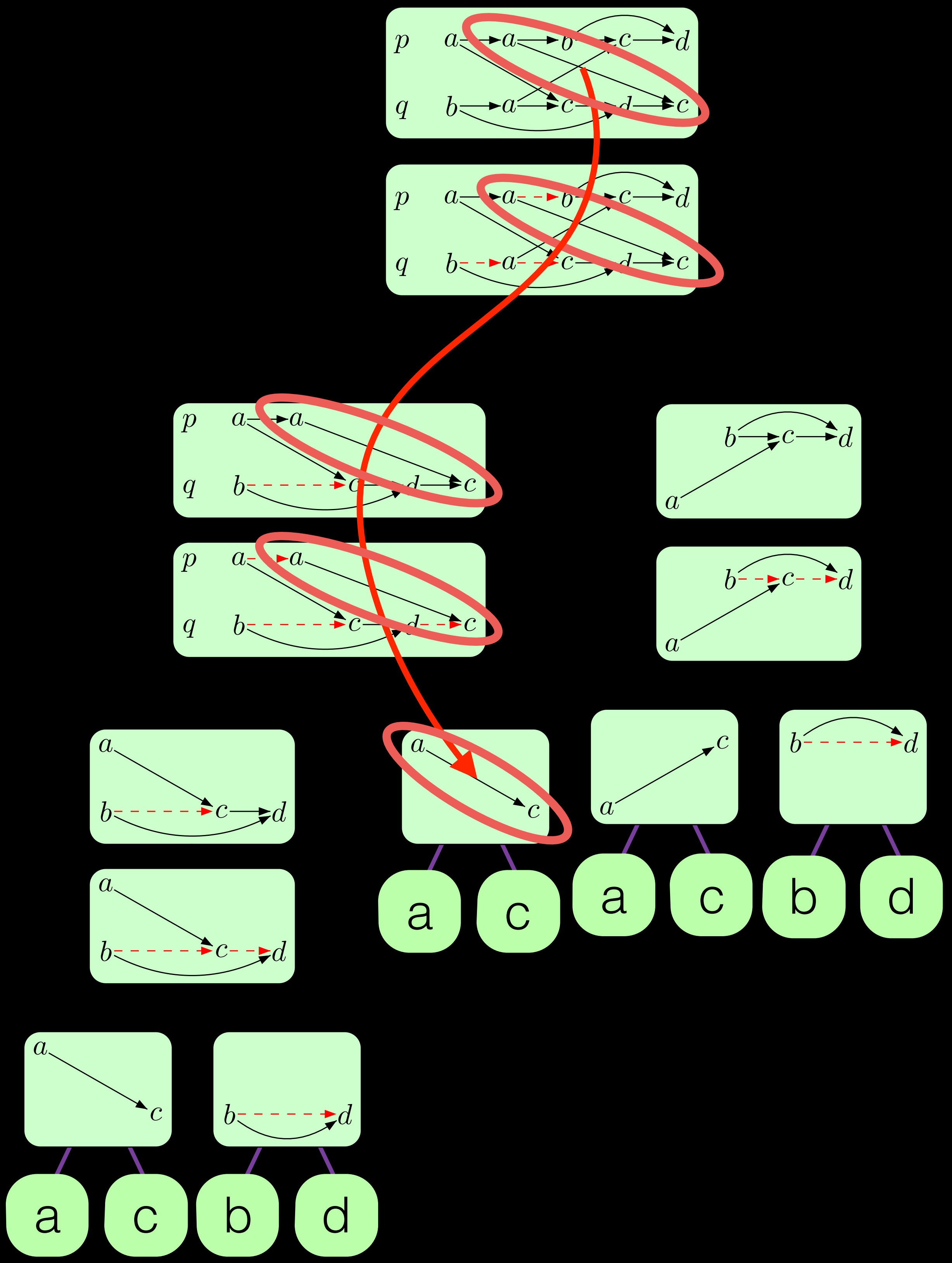


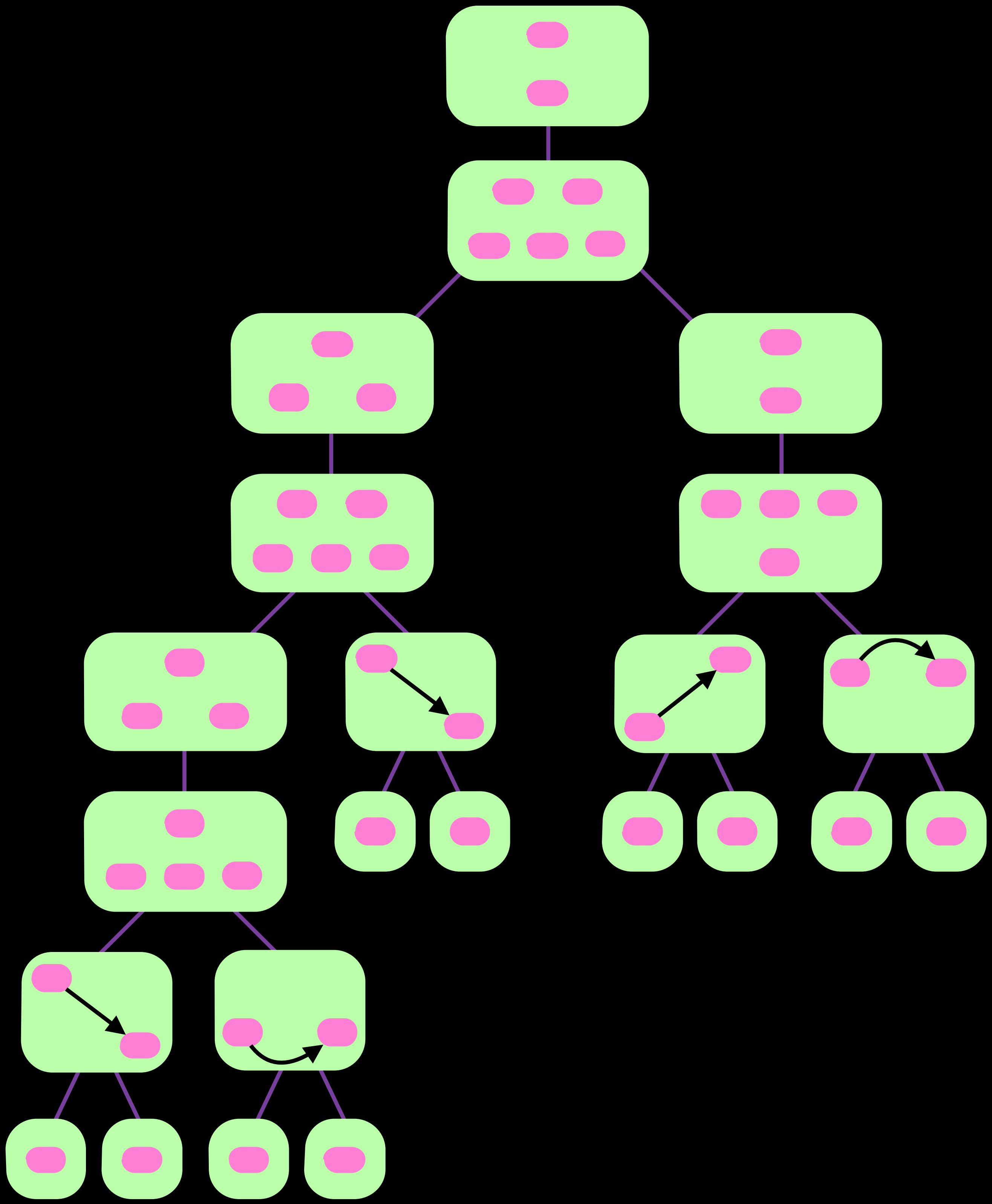
TREE INTERPRETATION



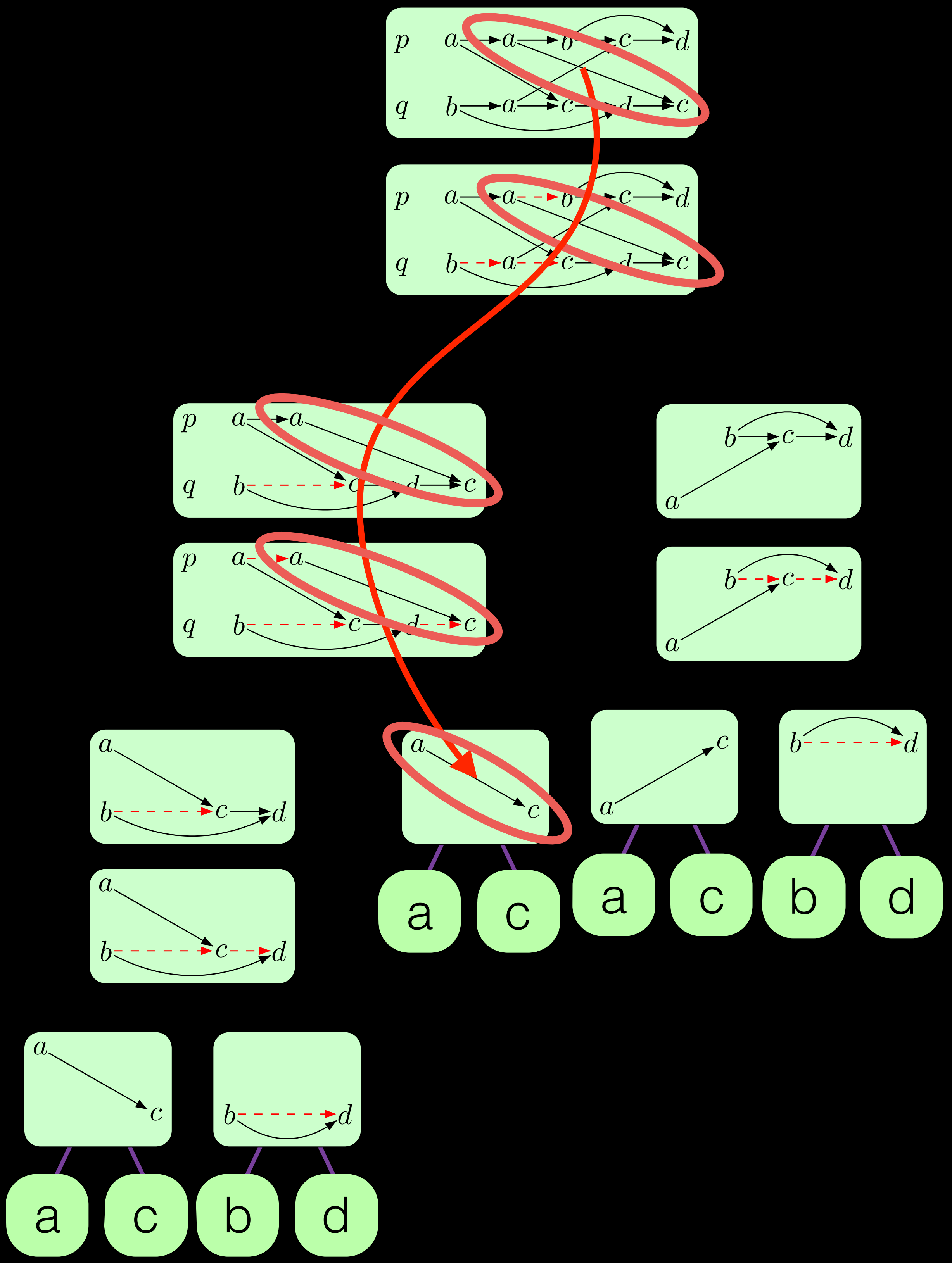


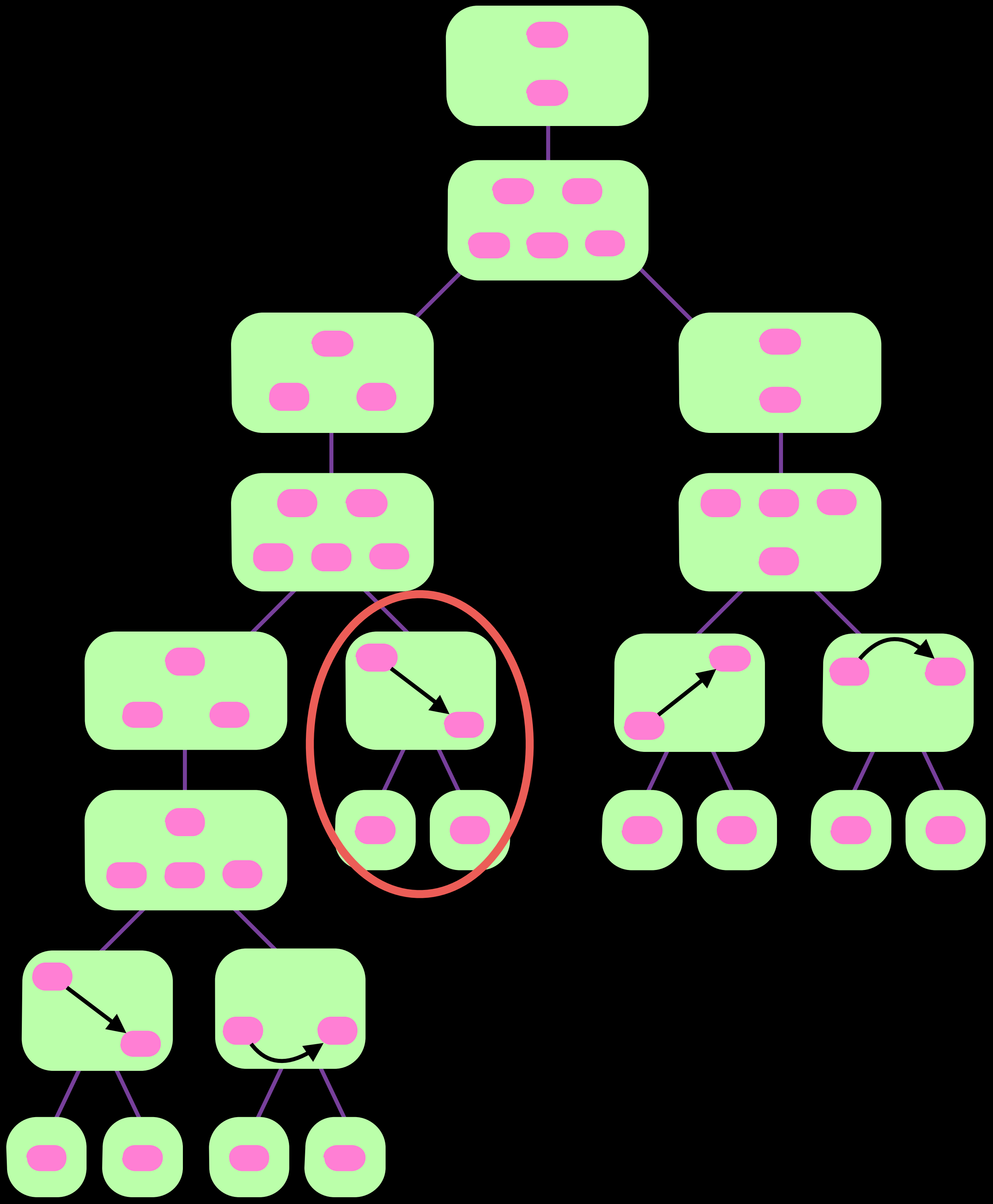
TREE INTERPRETATION



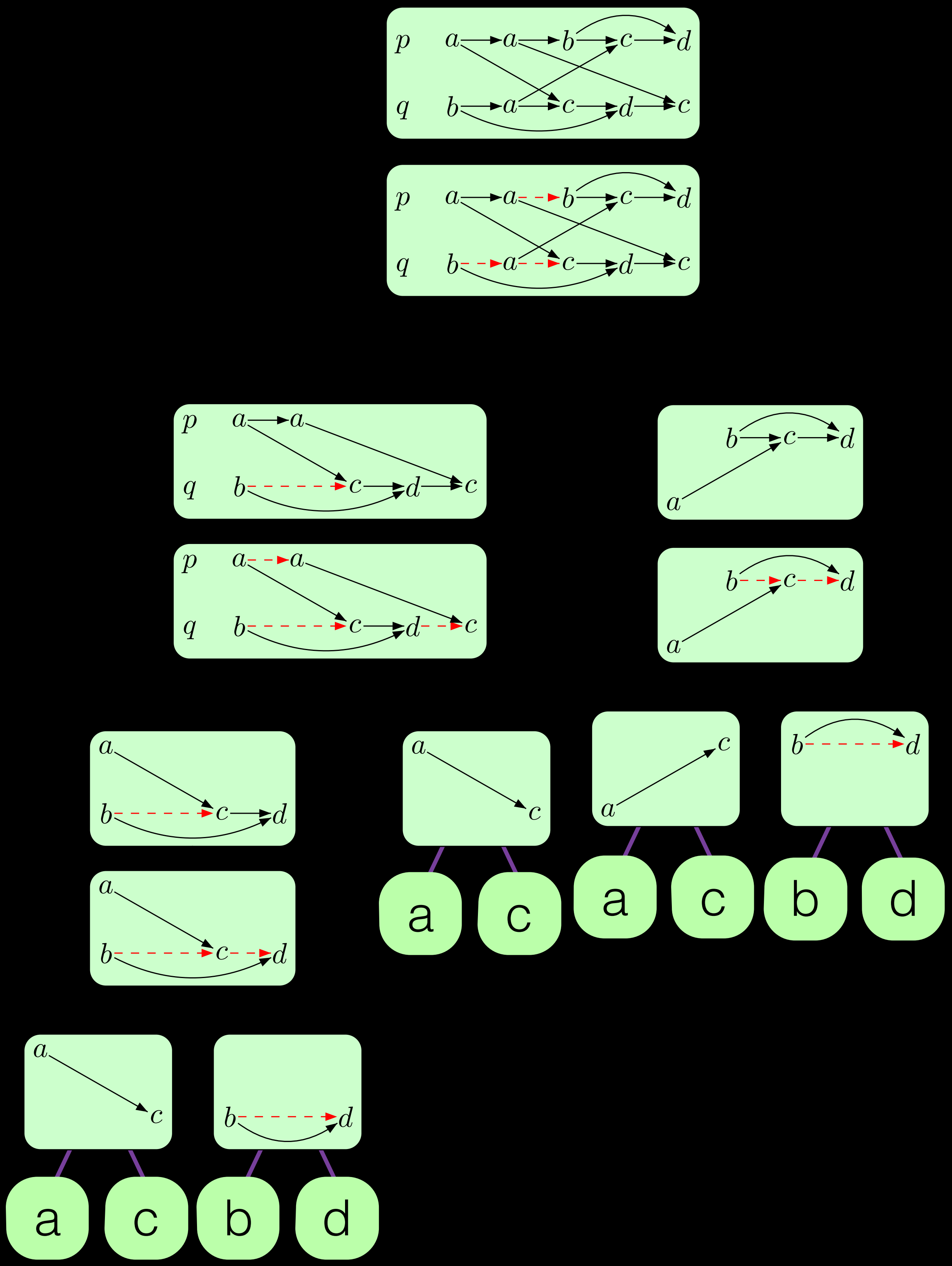


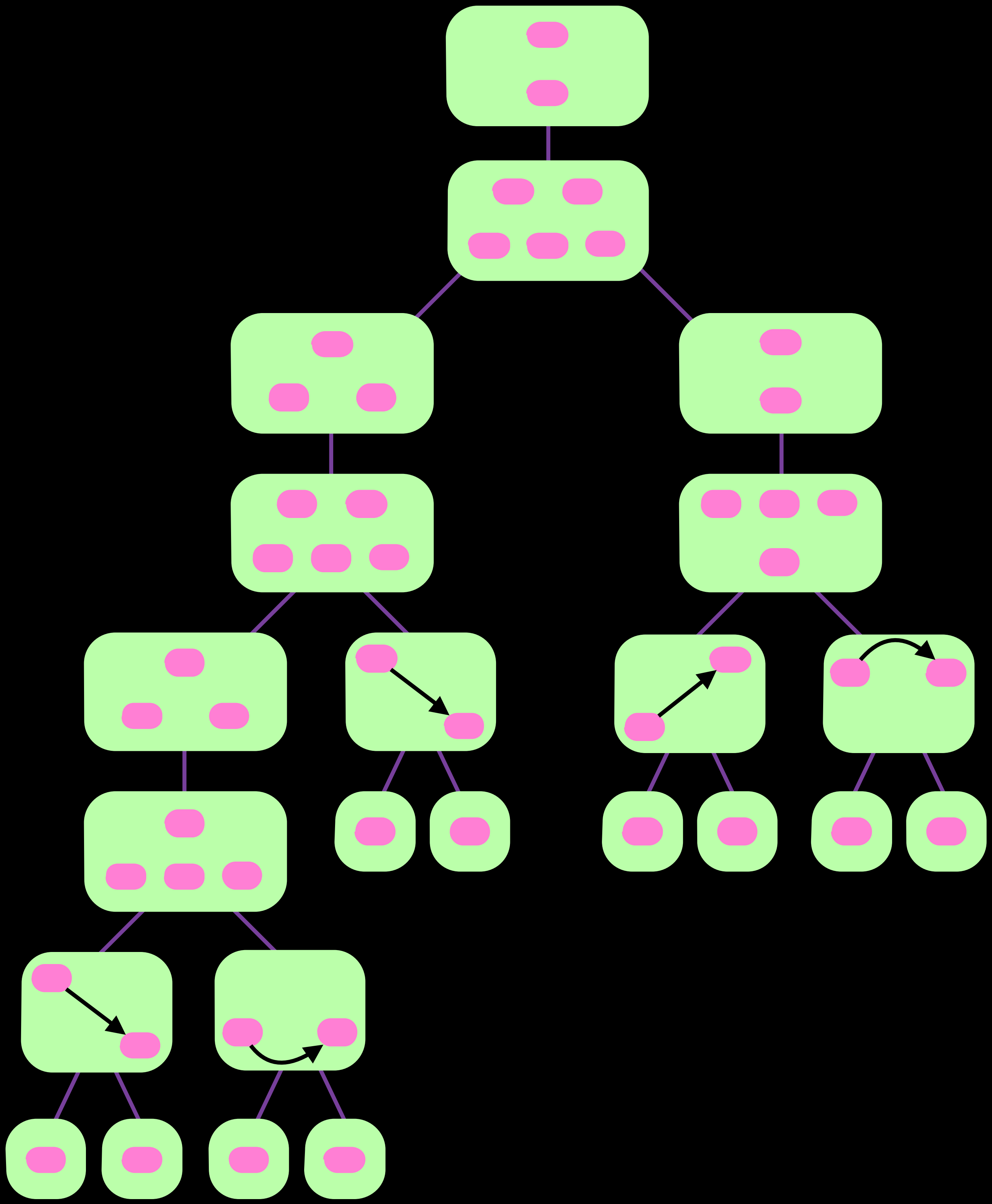
TREE INTERPRETATION



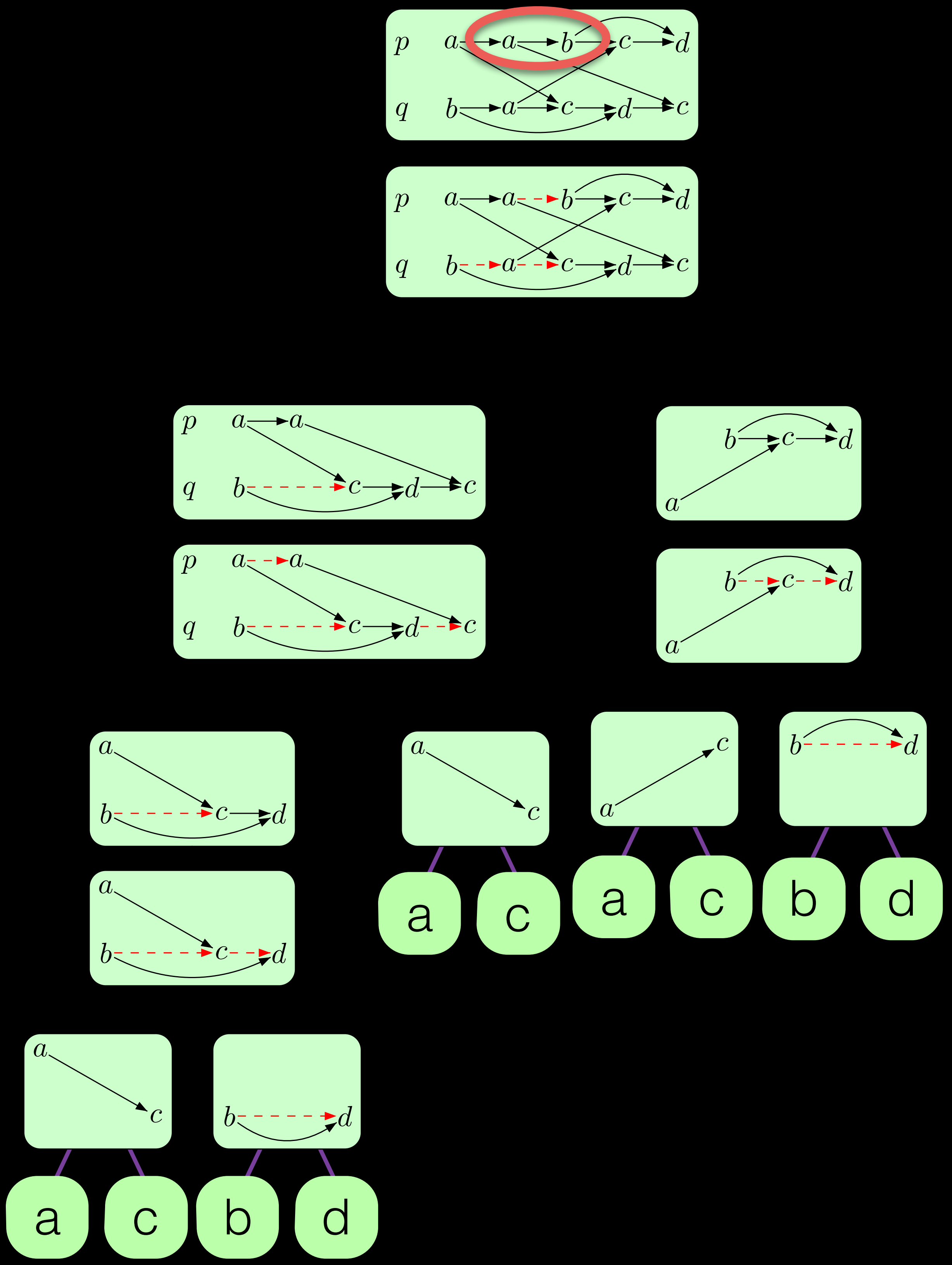


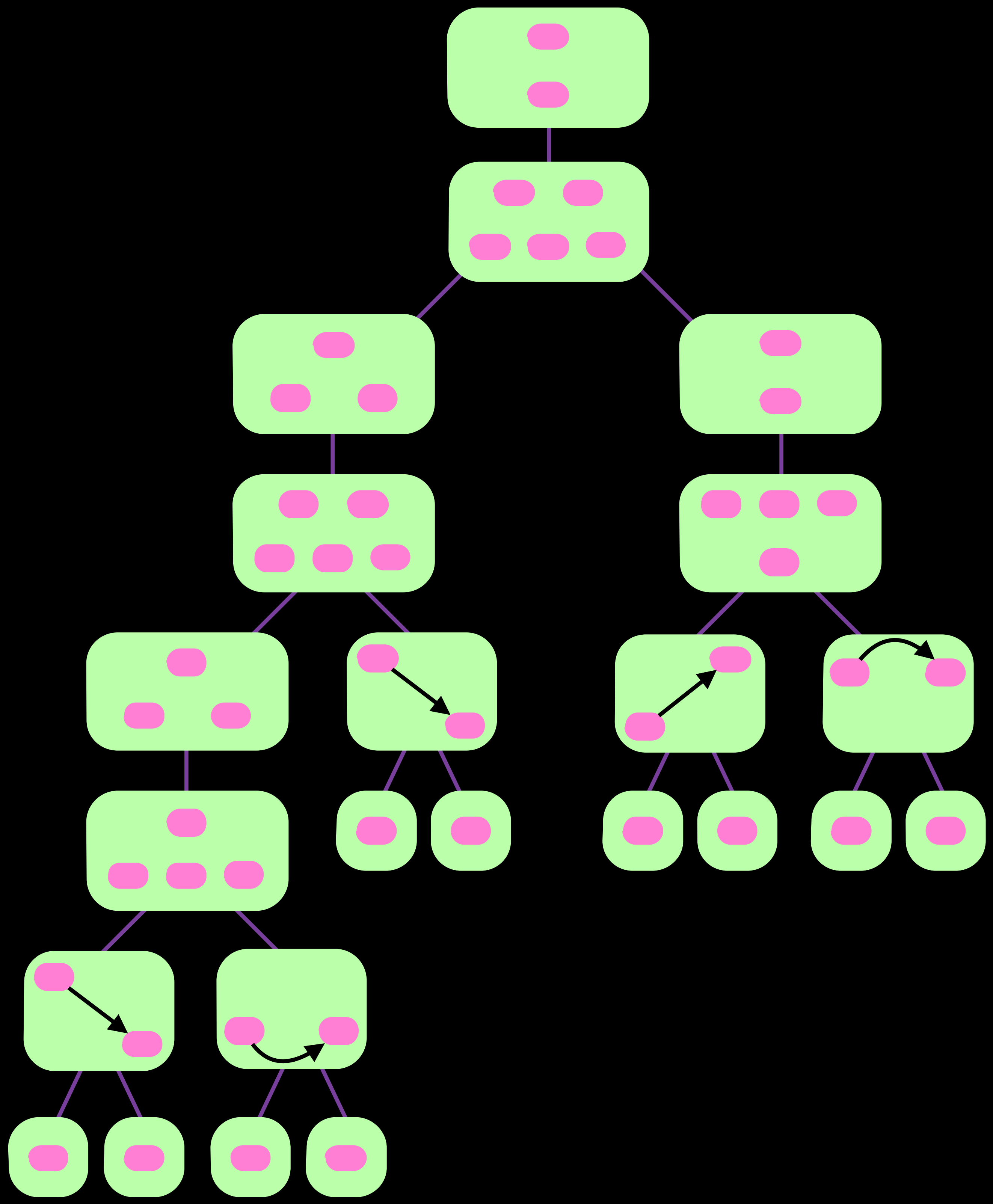
TREE INTERPRETATION



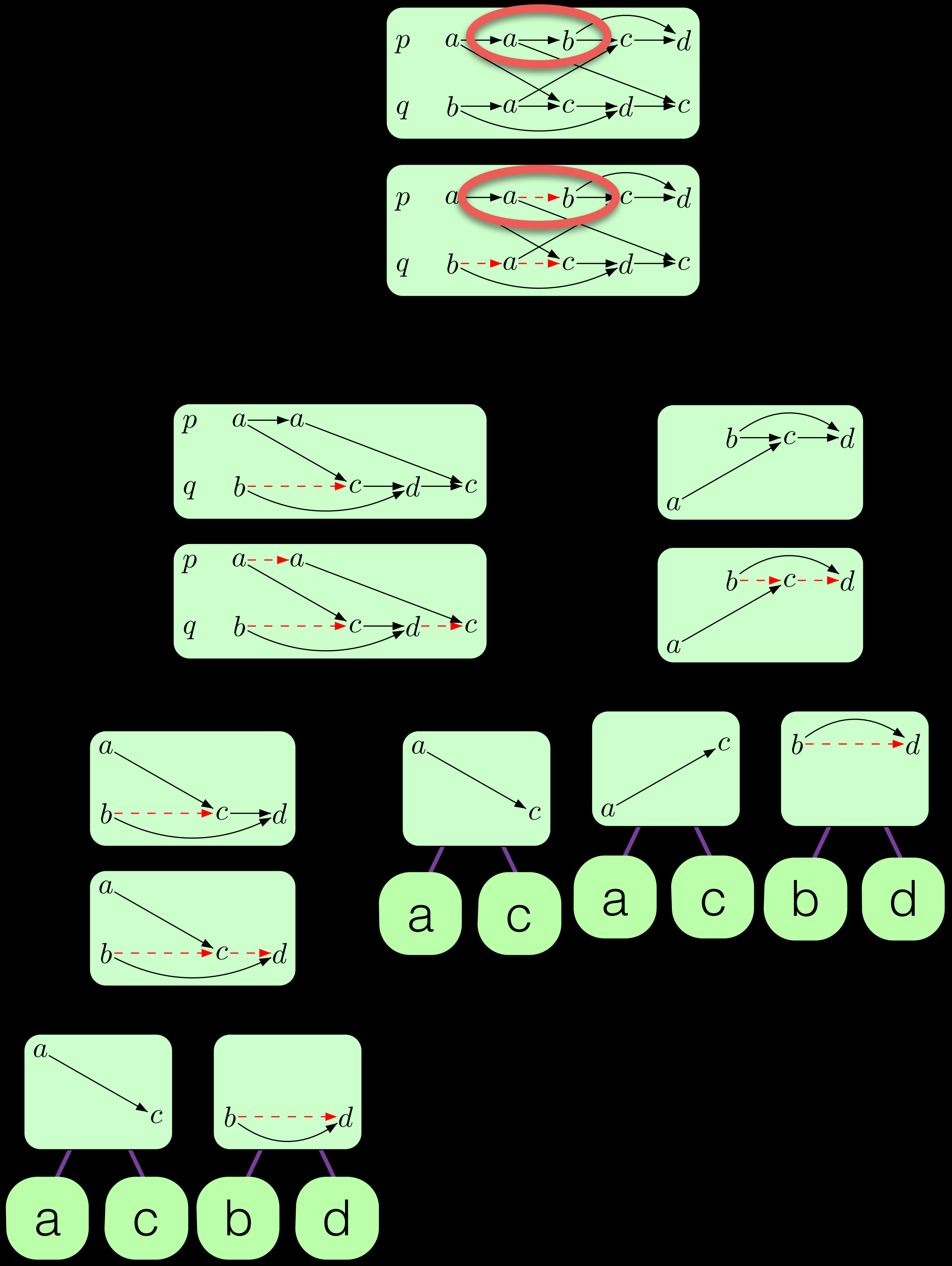


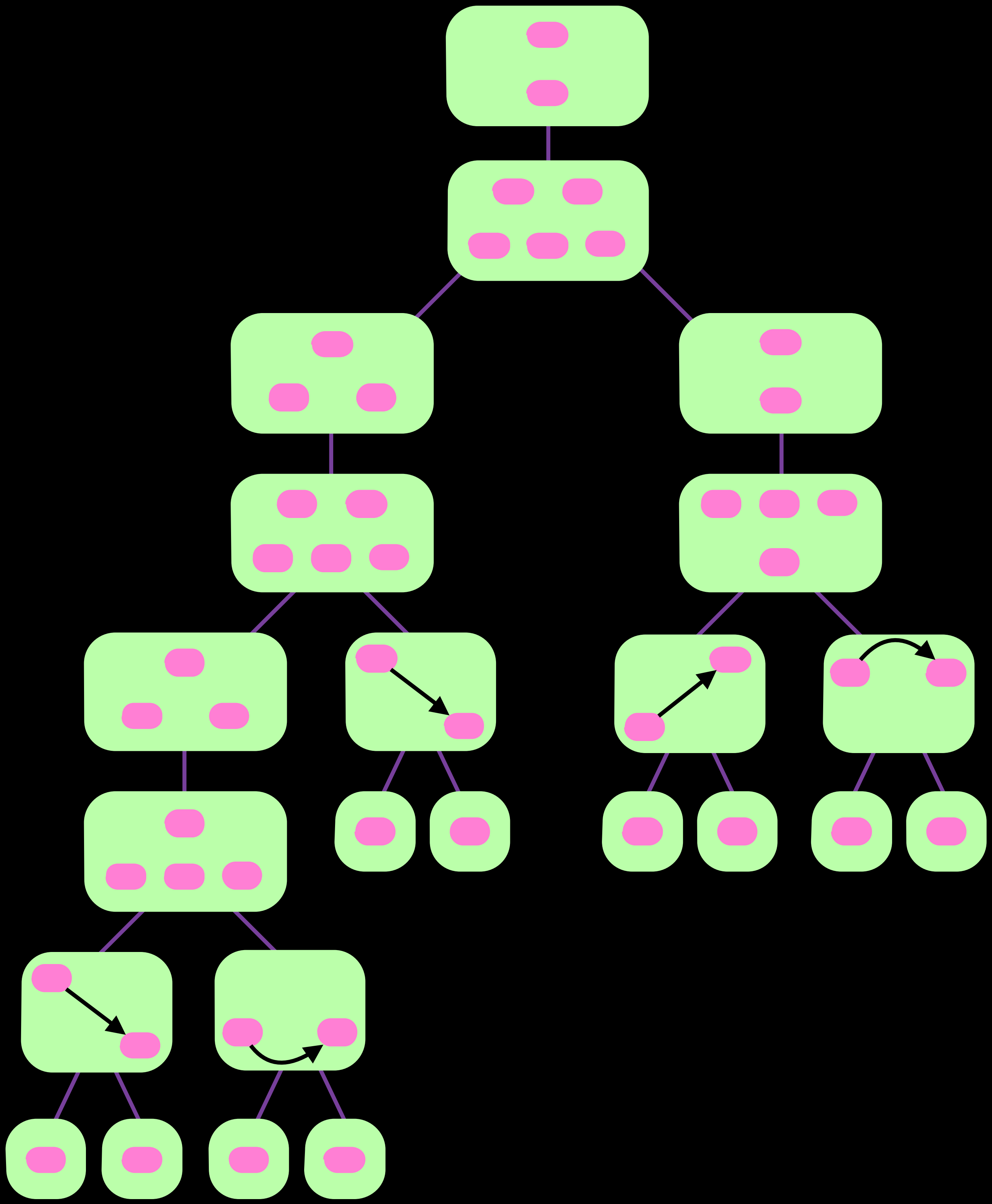
TREE INTERPRETATION



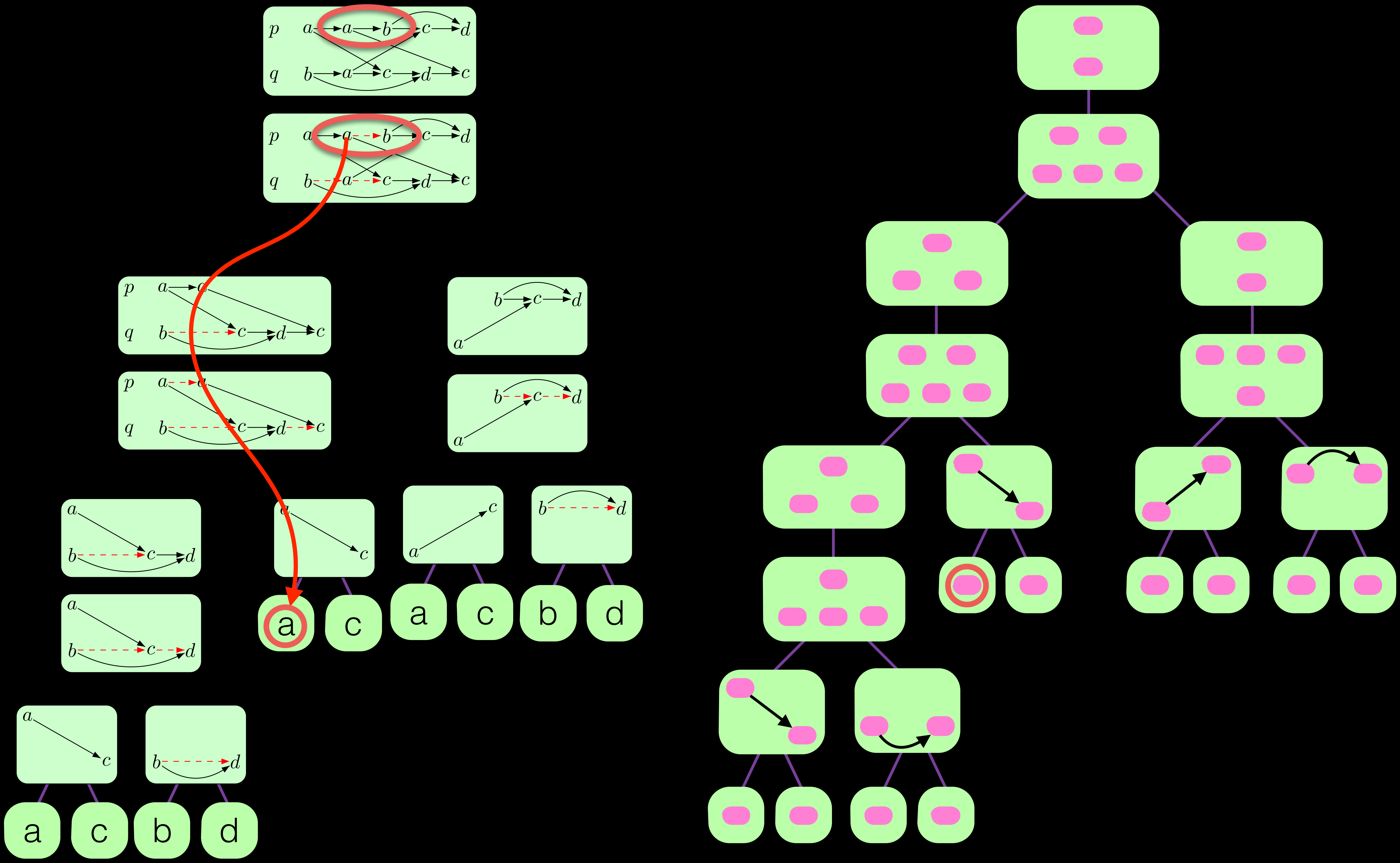


TREE INTERPRETATION

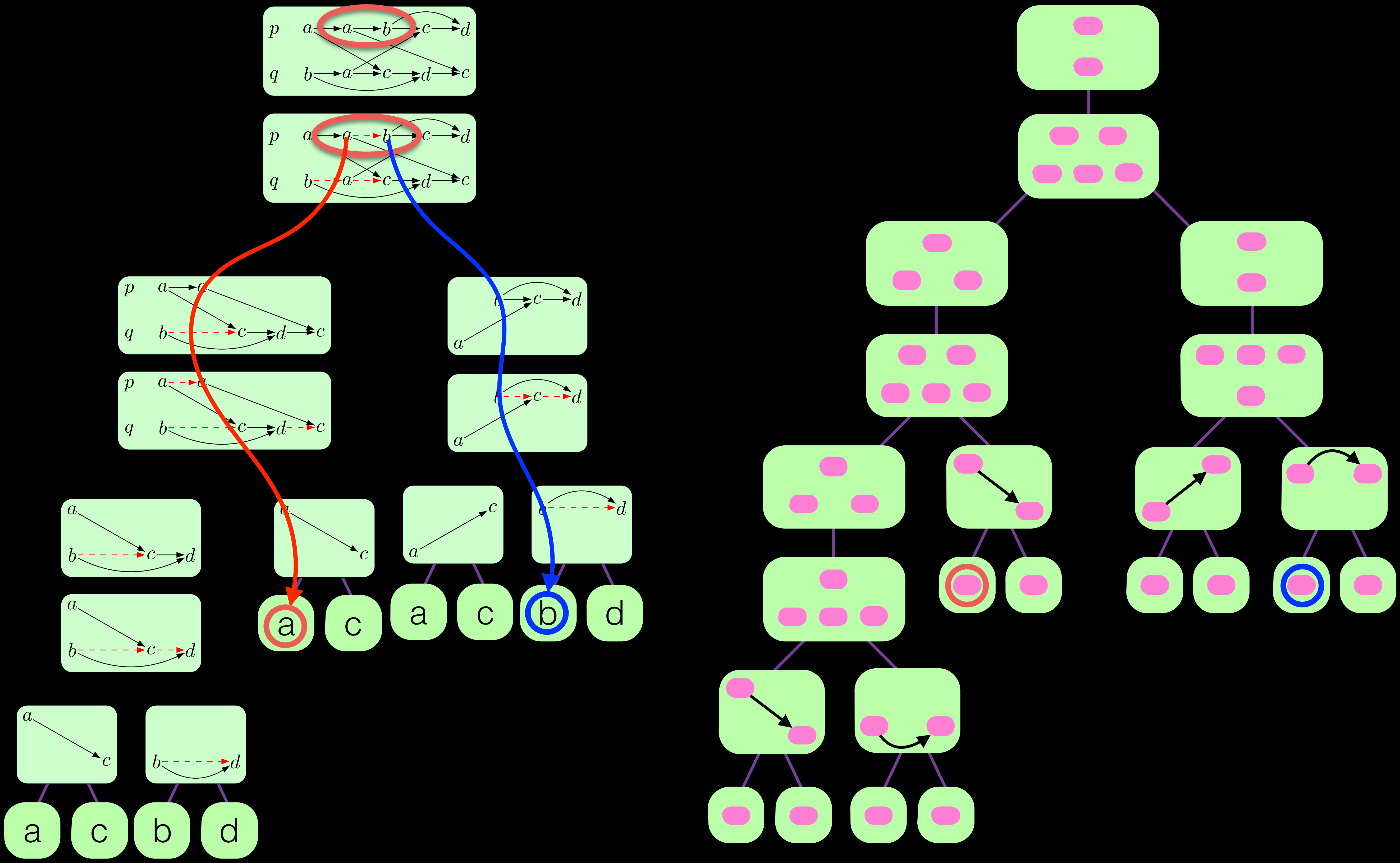




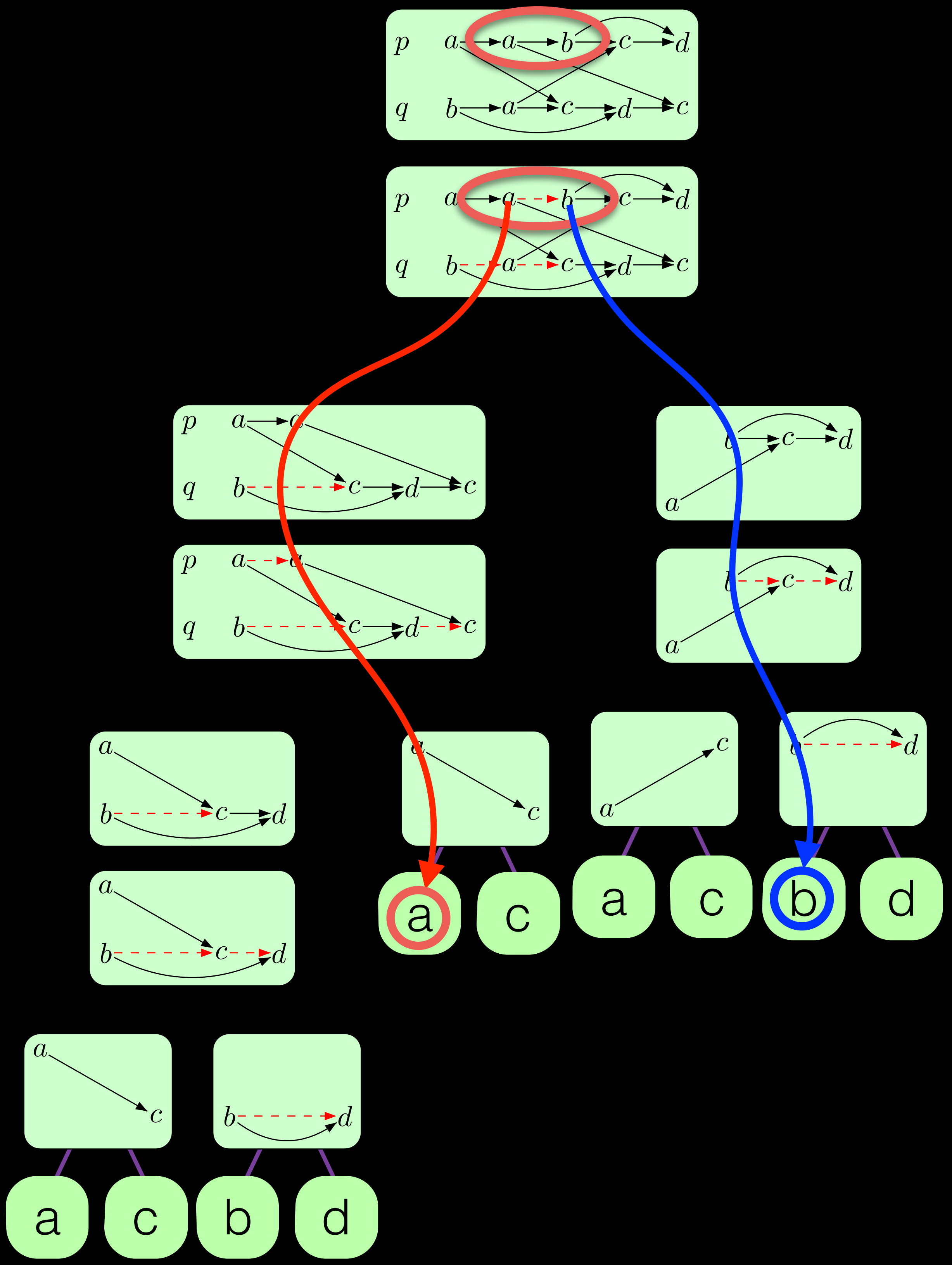
TREE INTERPRETATION

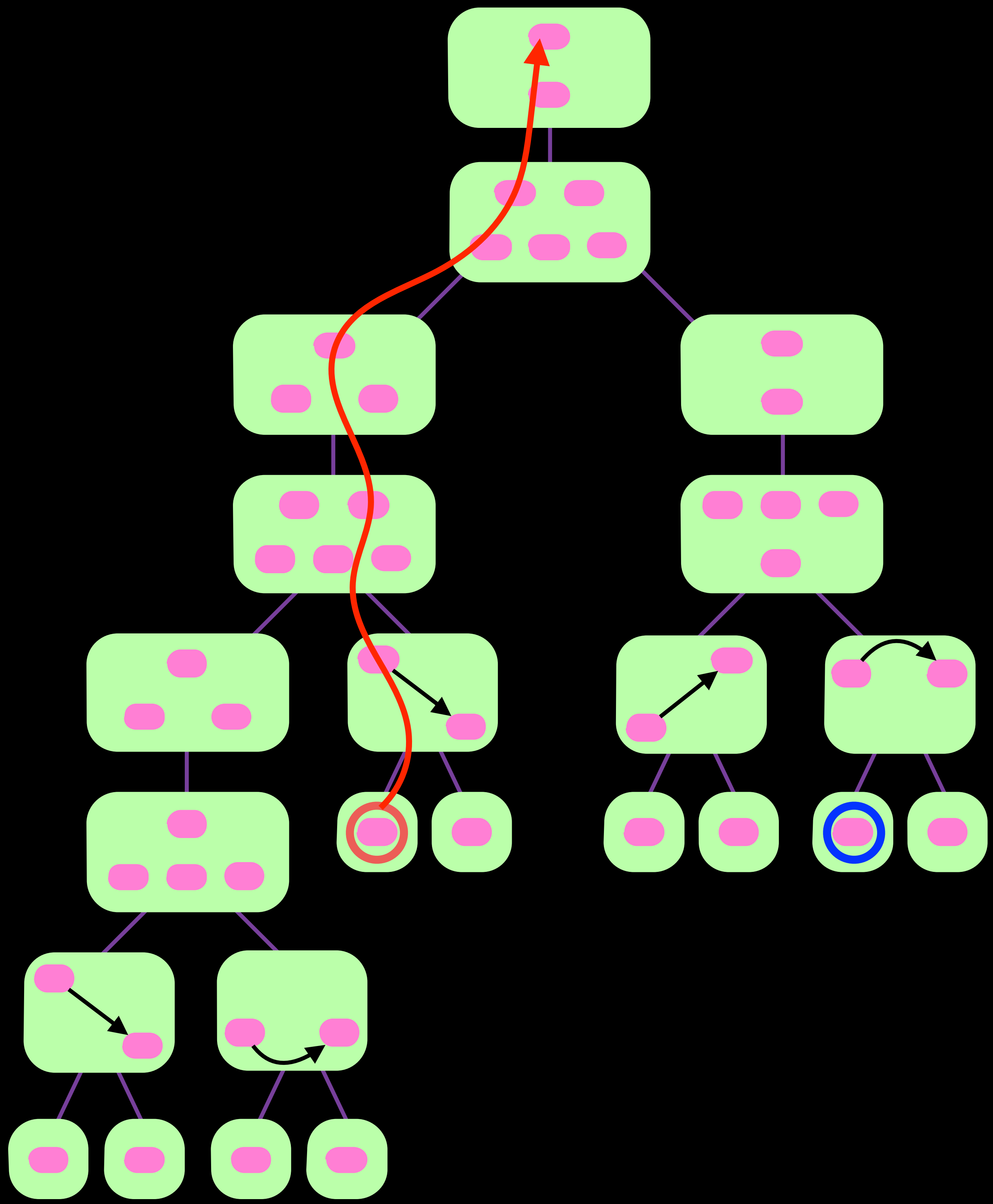


TREE INTERPRETATION

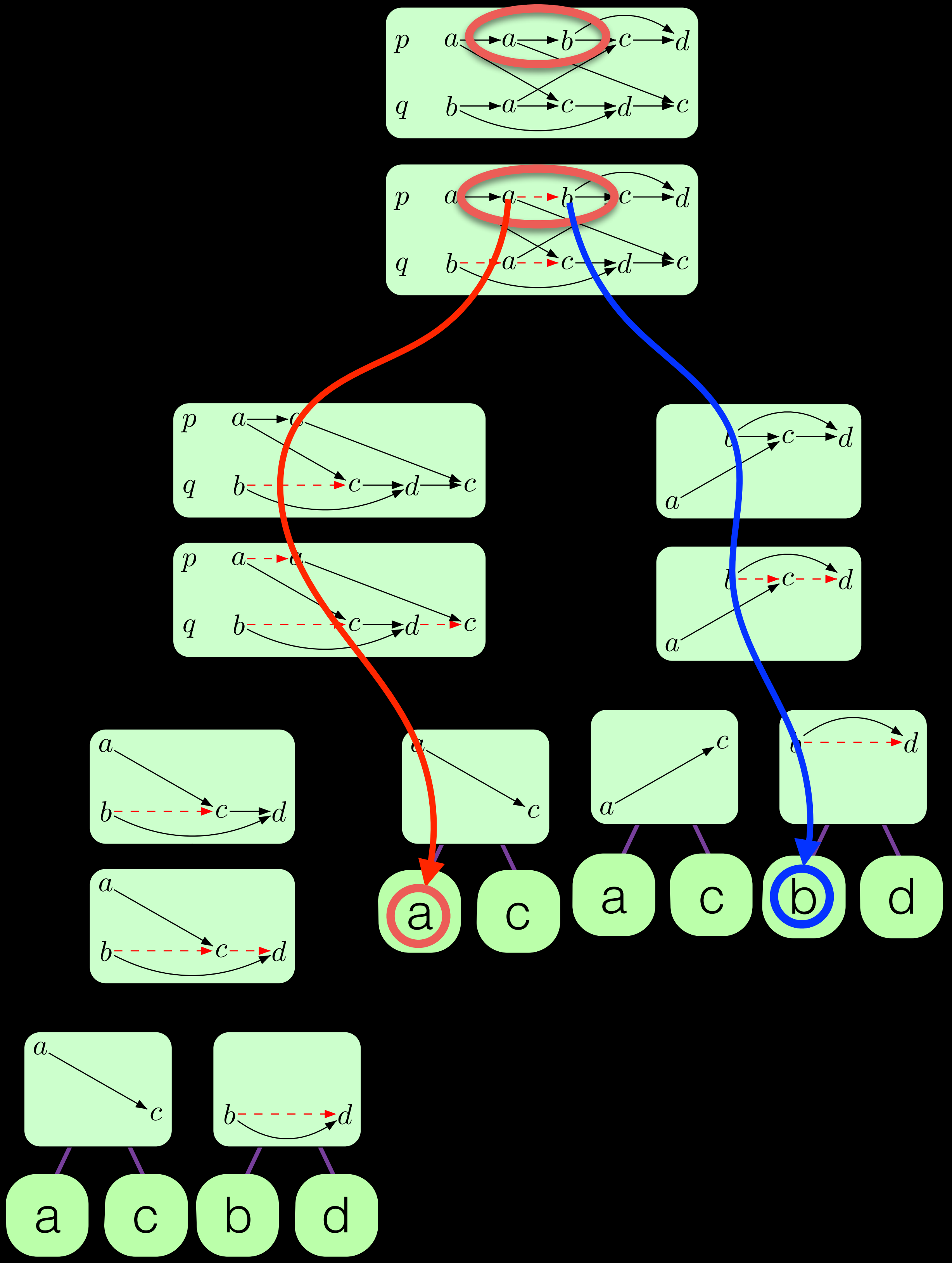


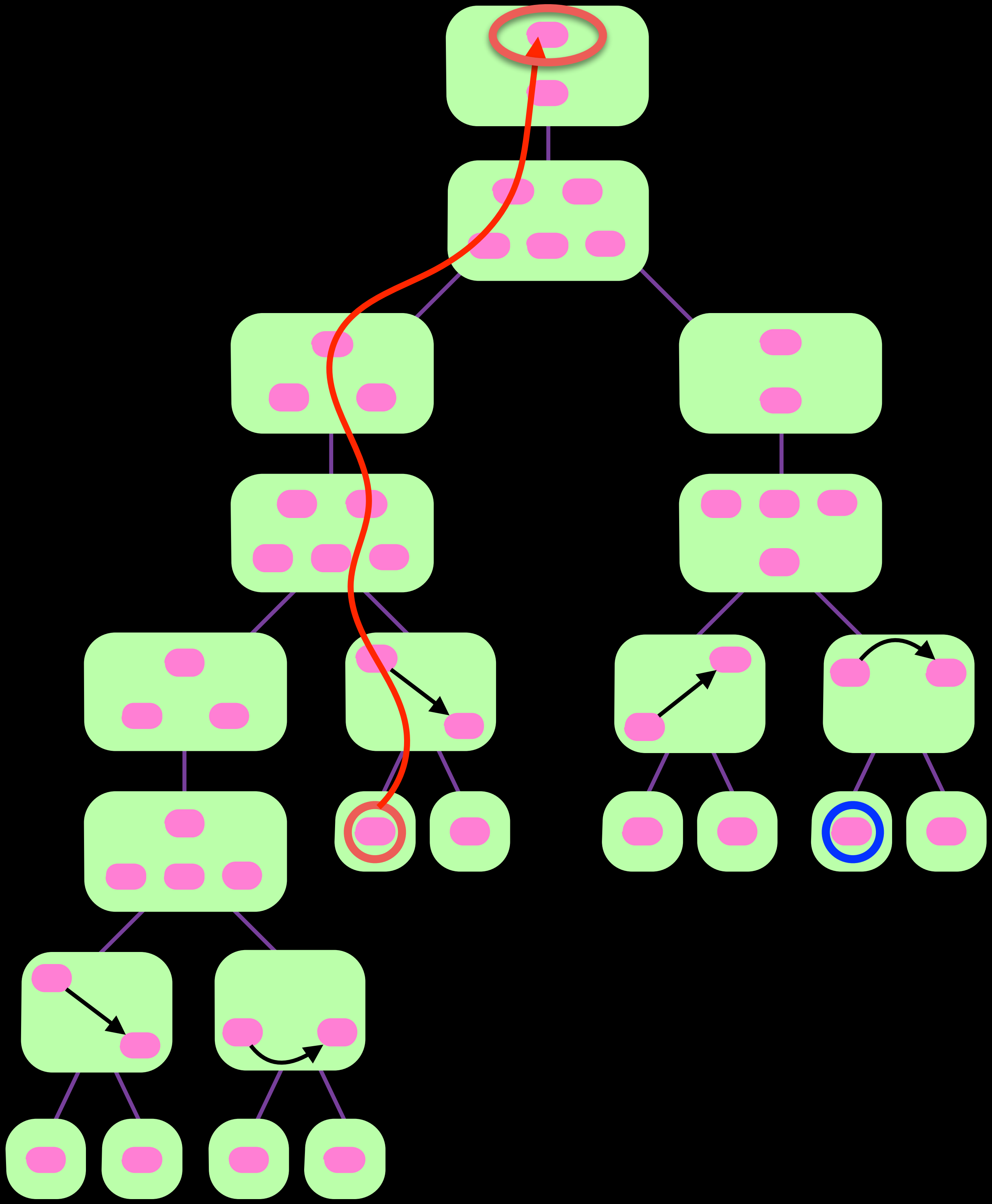
TREE INTERPRETATION



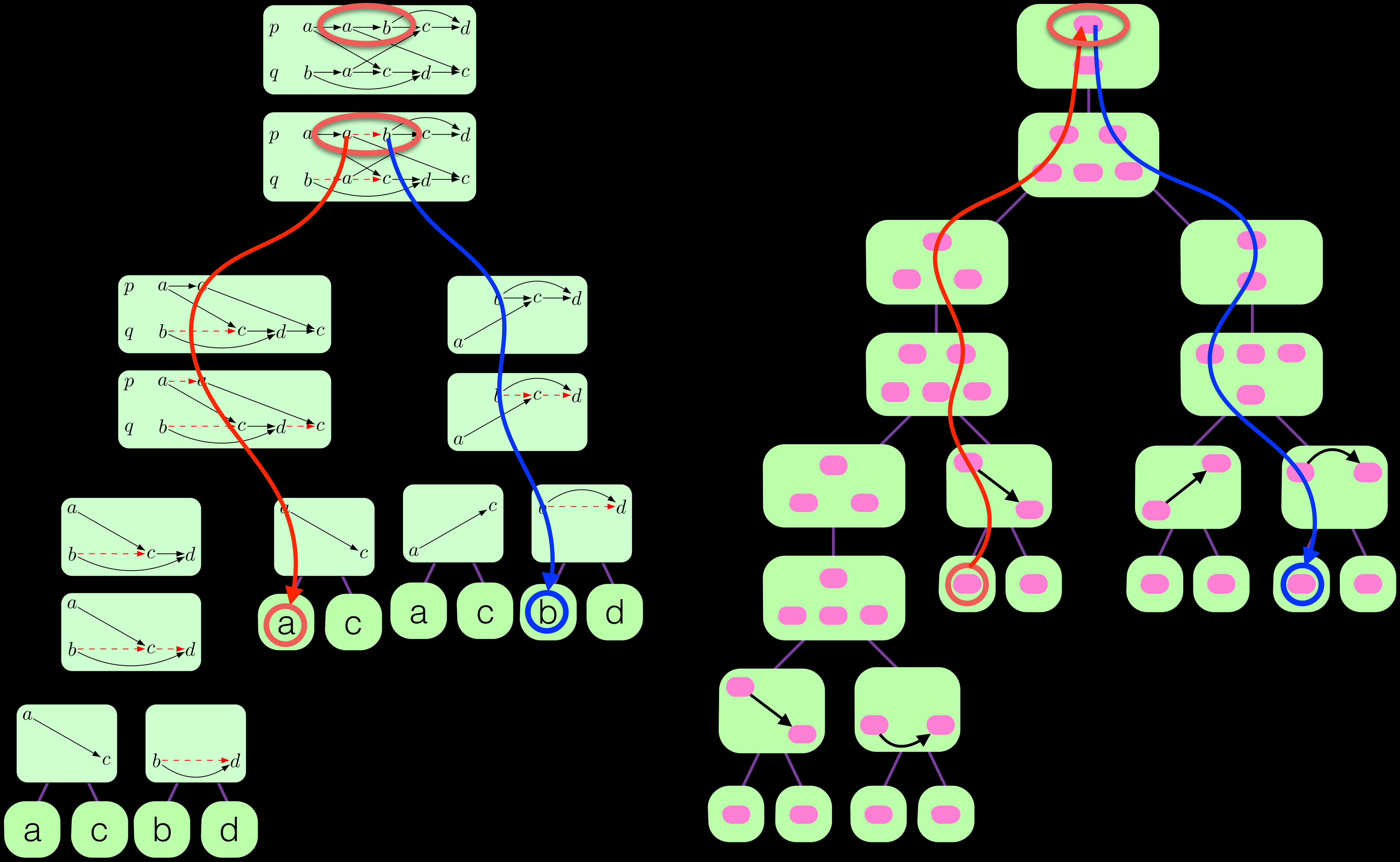


TREE INTERPRETATION



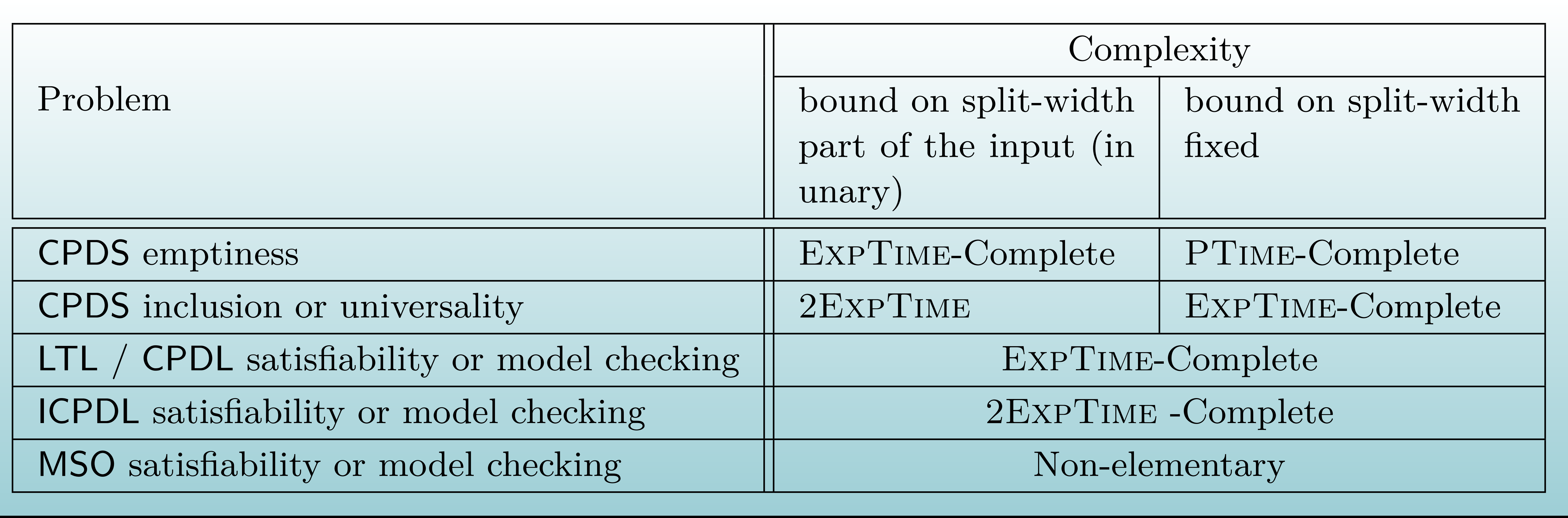


TREE INTERPRETATION



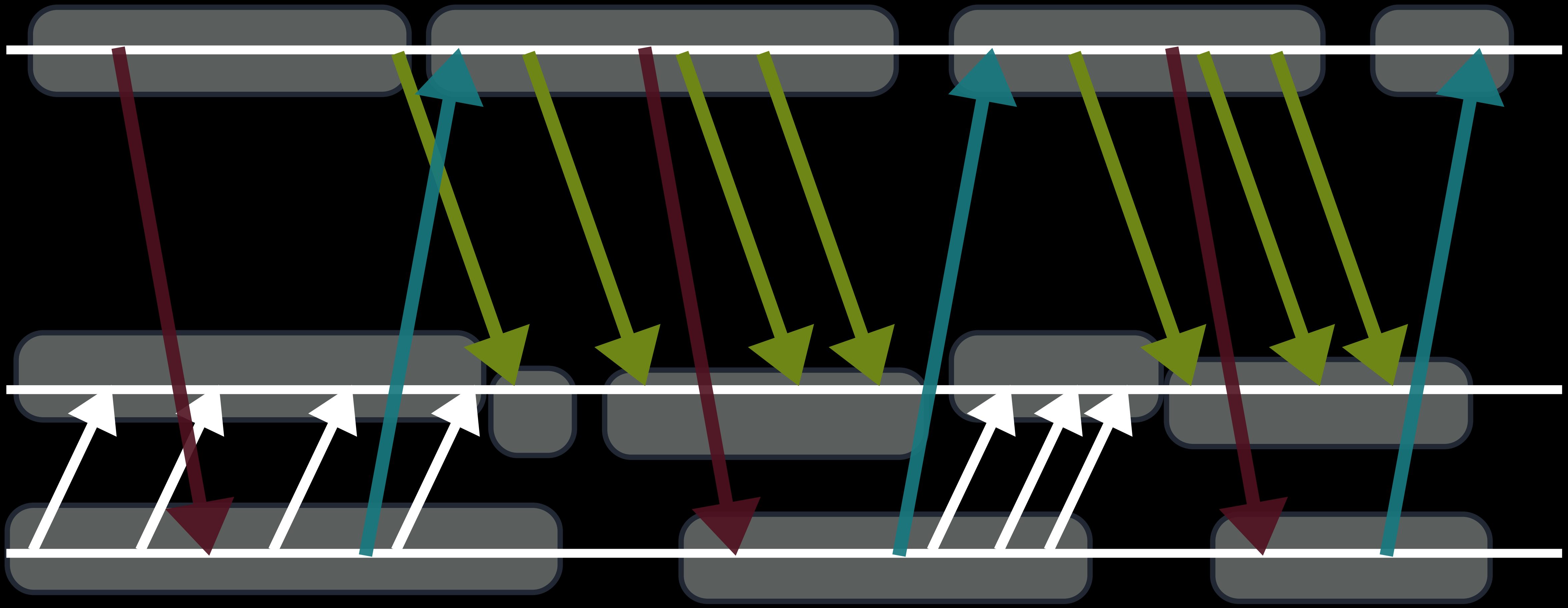
TREE INTERPRETATION

Split-width

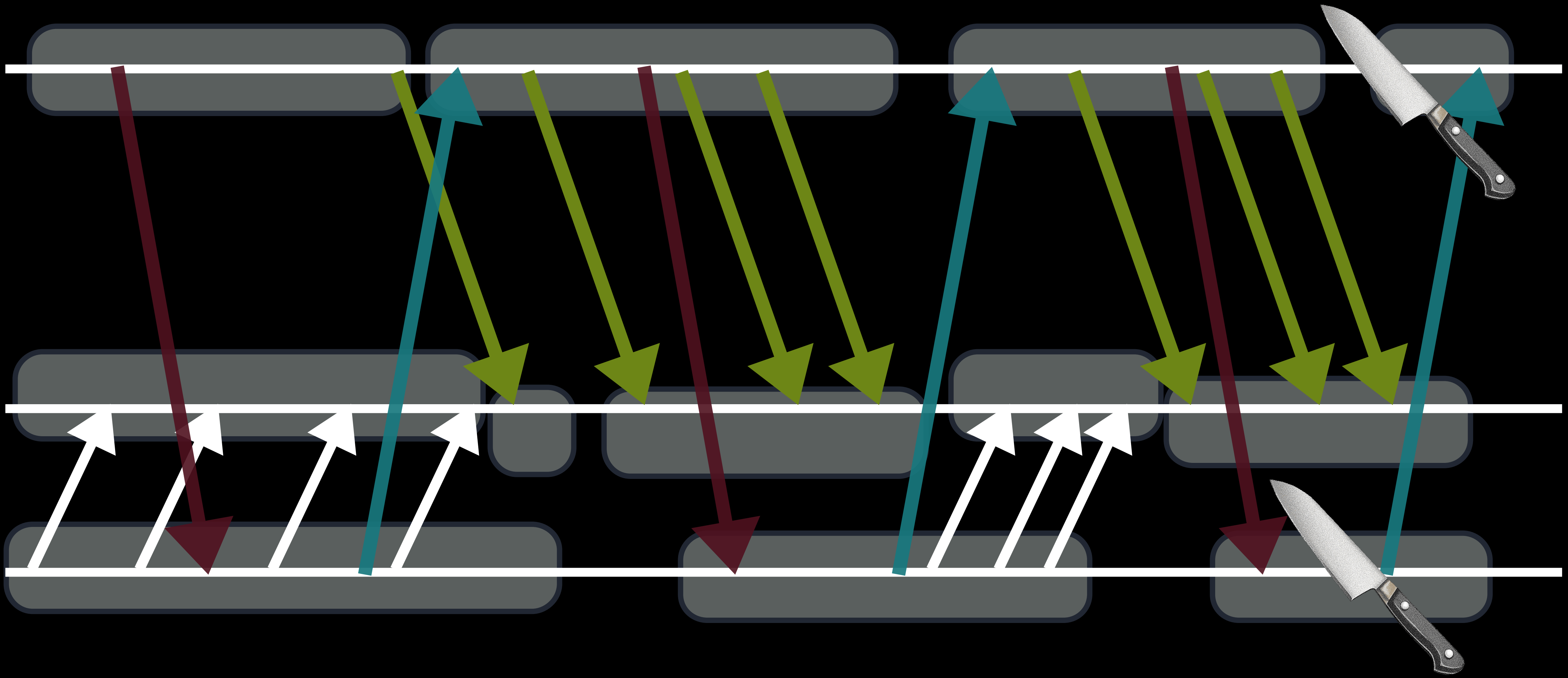


Problem	Complexity	
	bound on split-width part of the input (in unary)	bound on split-width fixed
CPDS emptiness	ExpTime-Complete	PTime-Complete
CPDS inclusion or universality	2ExpTime	ExpTime-Complete
LTL / CPDL satisfiability or model checking	ExpTime-Complete	
ICPDL satisfiability or model checking	2ExpTime -Complete	
MSO satisfiability or model checking	Non-elementary	

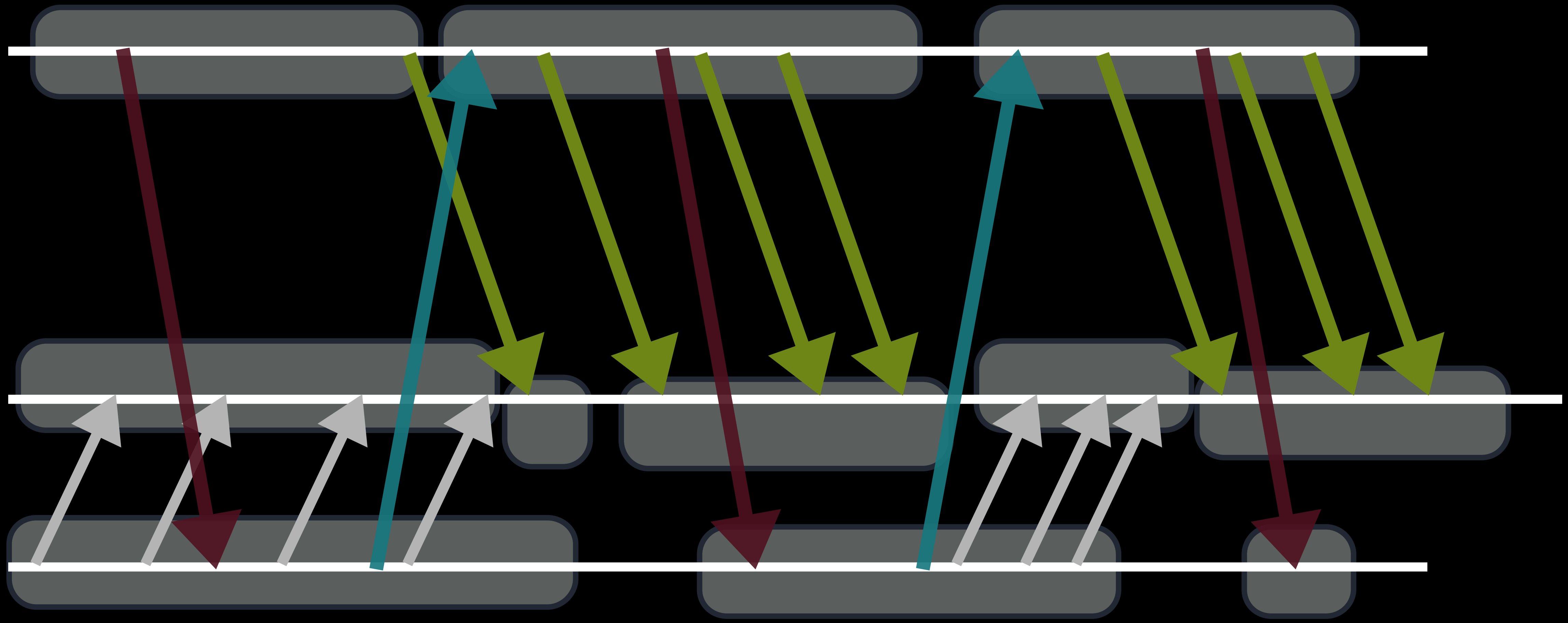
SPLIT-WIDTH



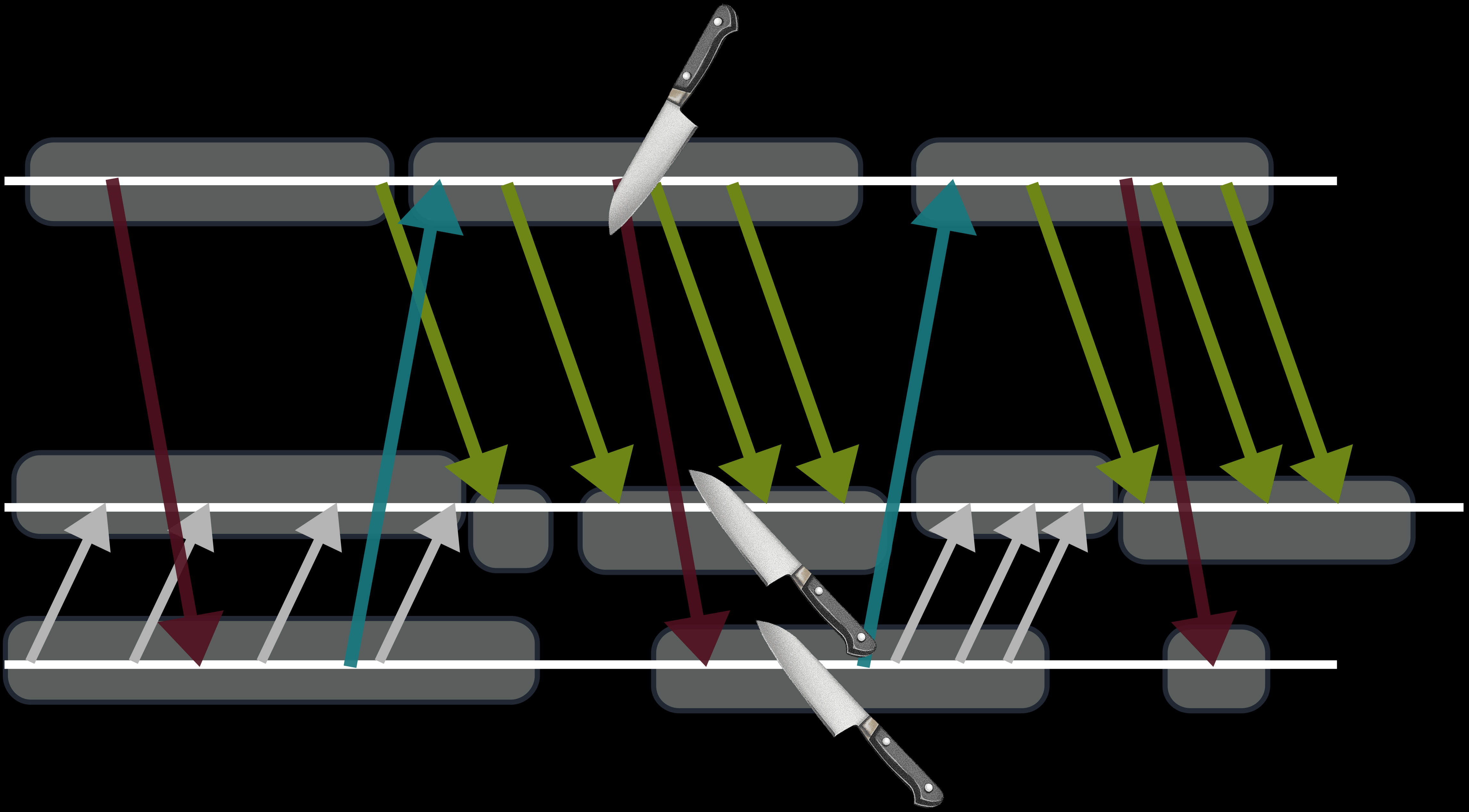
SPLIT-WIDTH



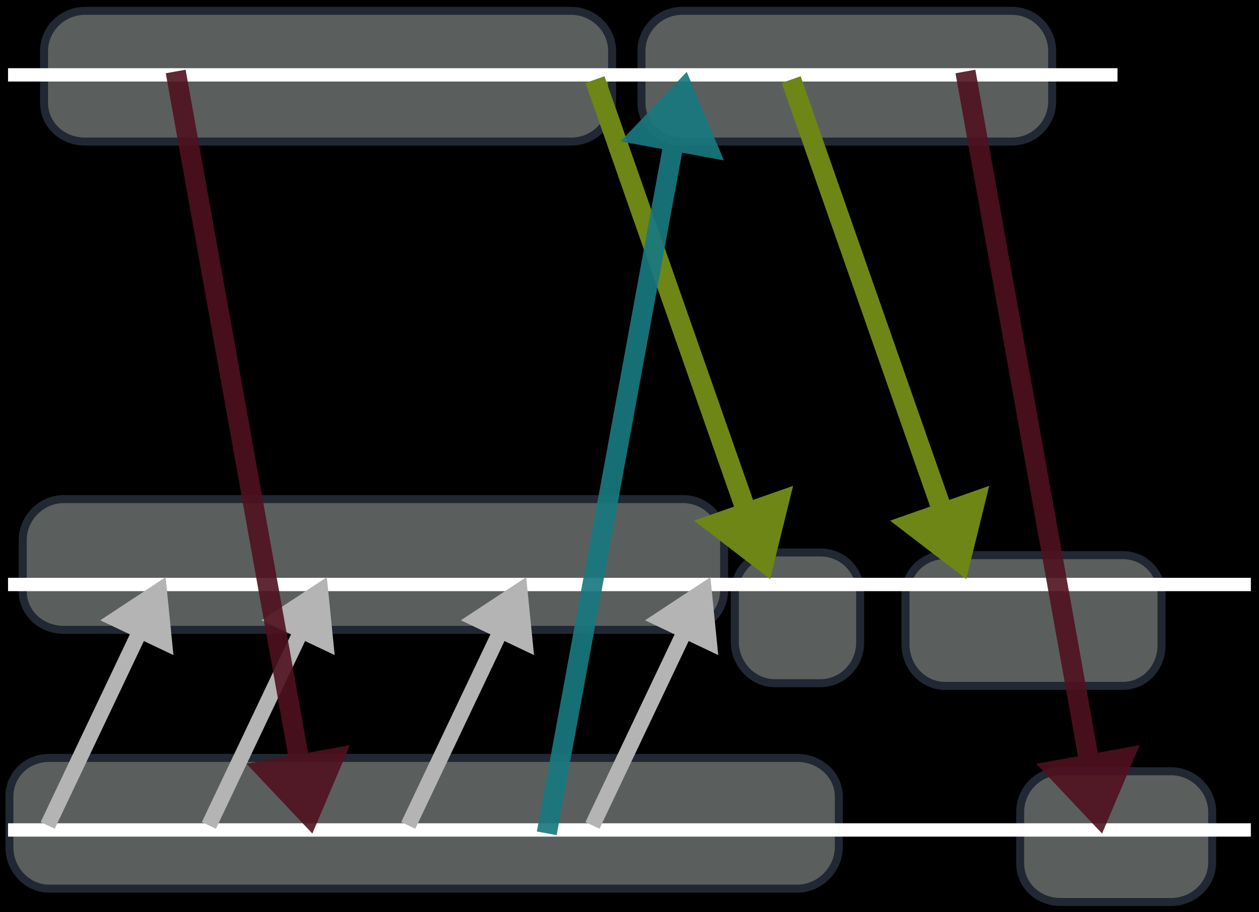
SPLIT-WIDTH 3



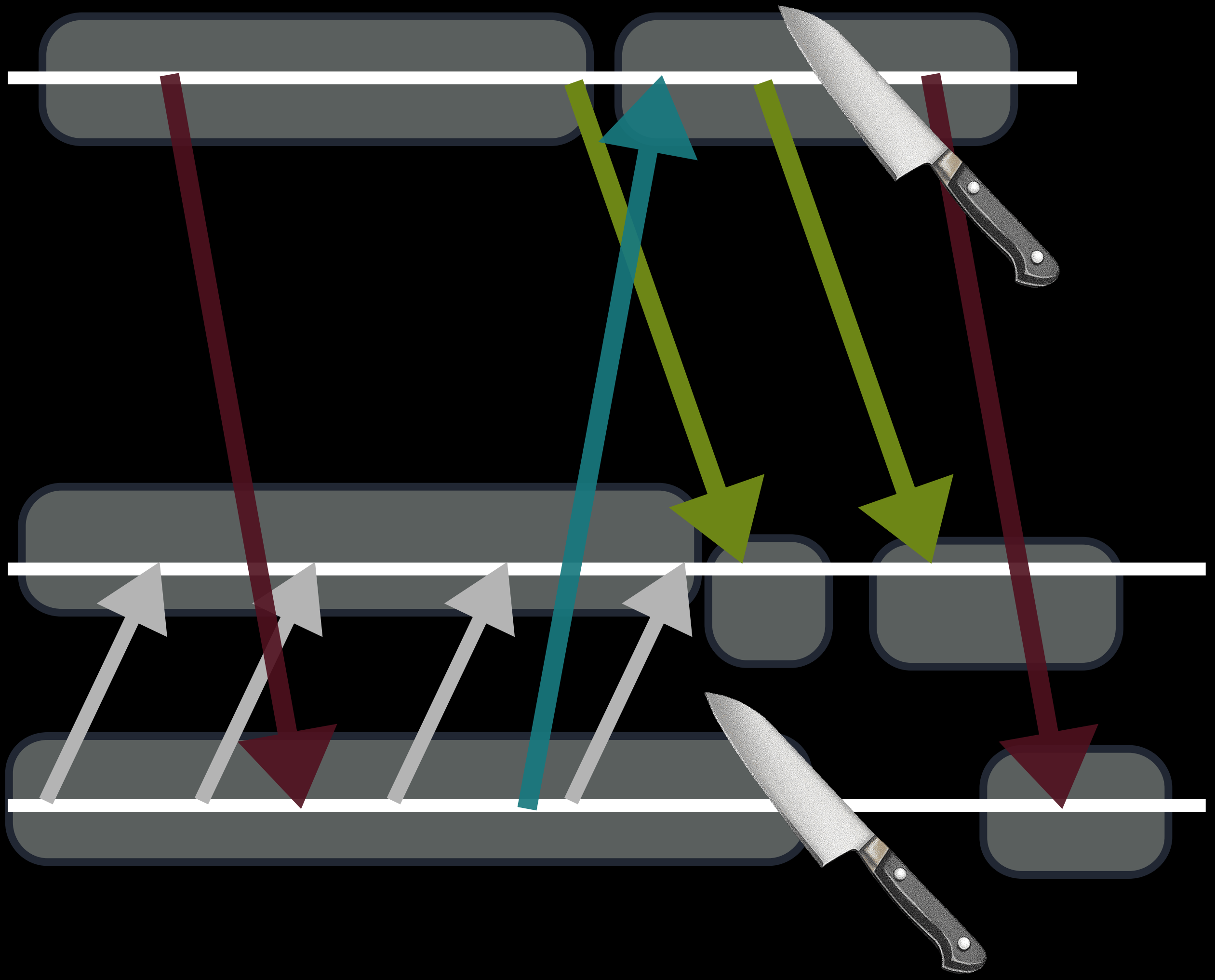
SPLIT-WIDTH 3



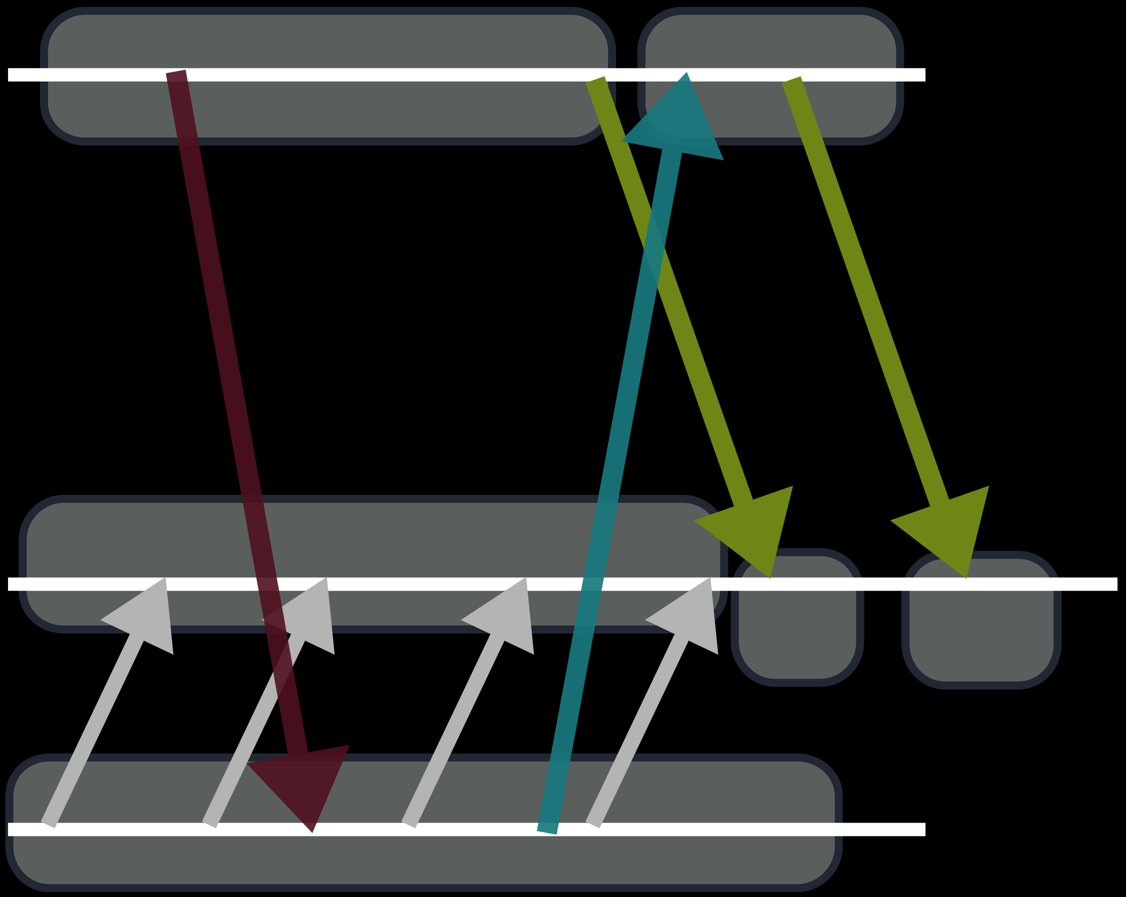
SPLIT-WIDTH 3



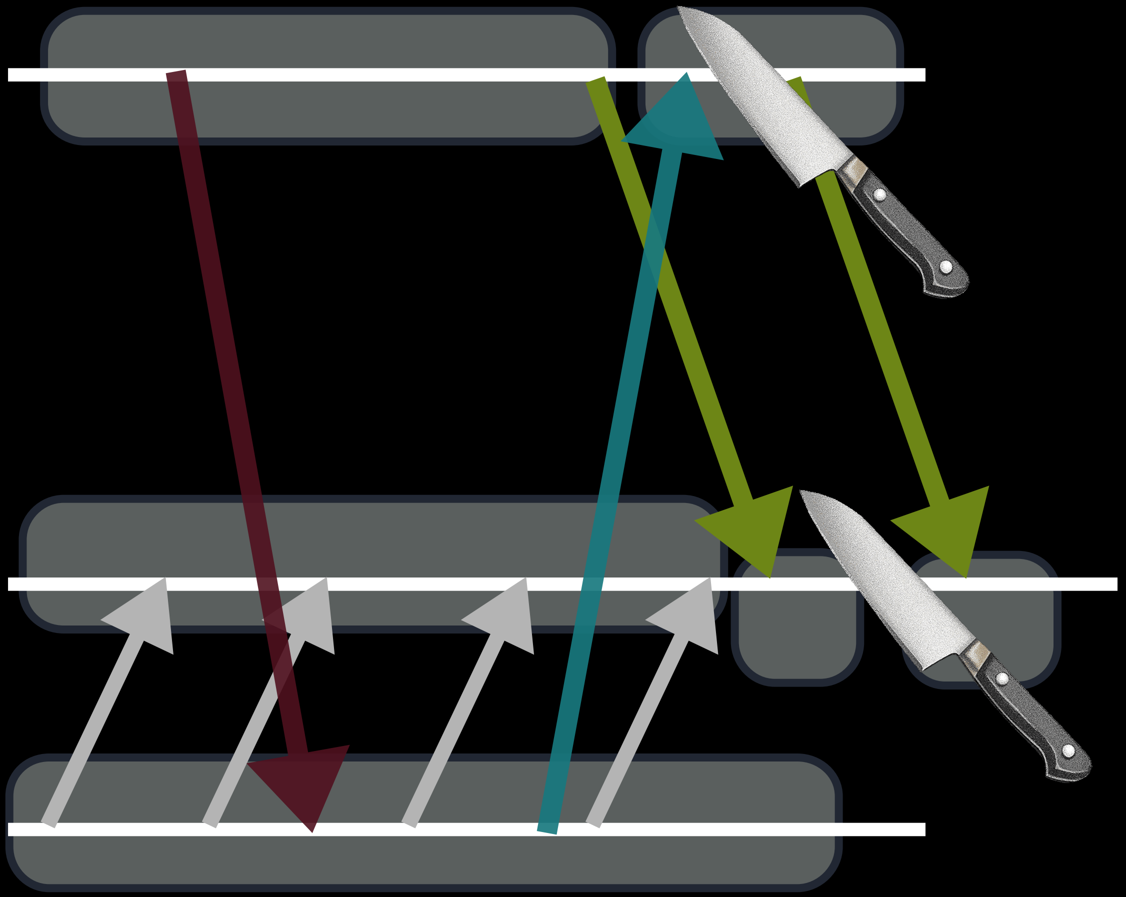
SPLIT-WIDTH 3



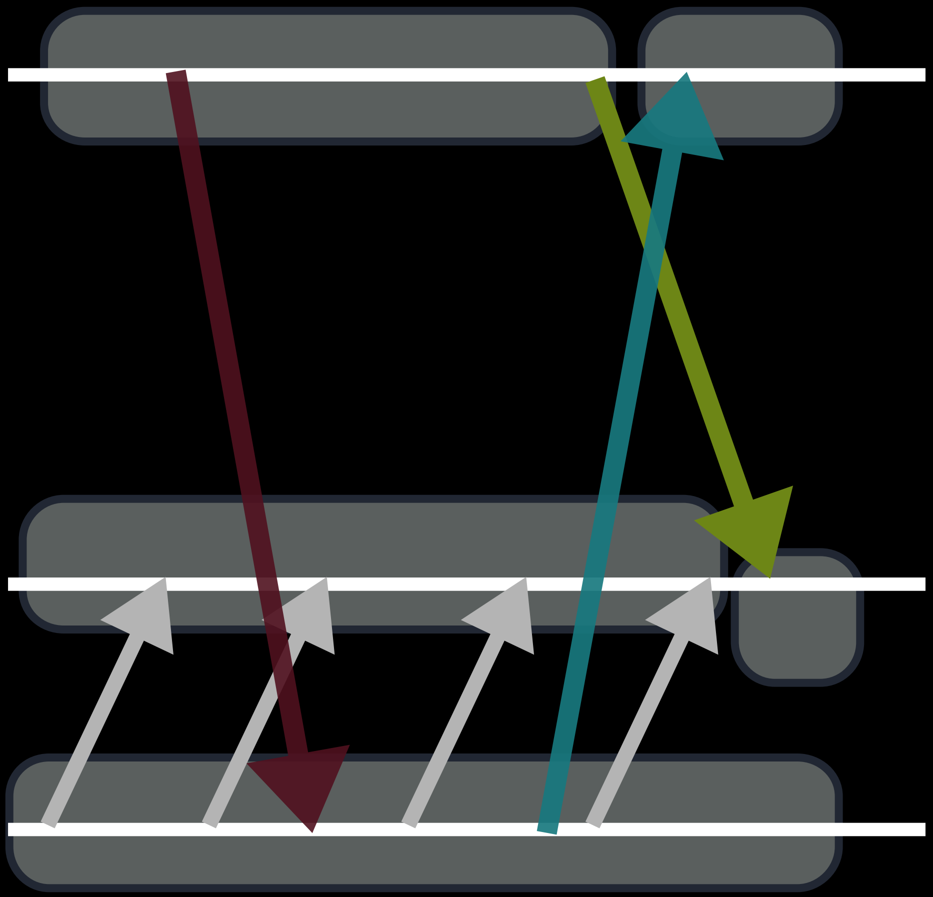
SPLIT-WIDTH 3



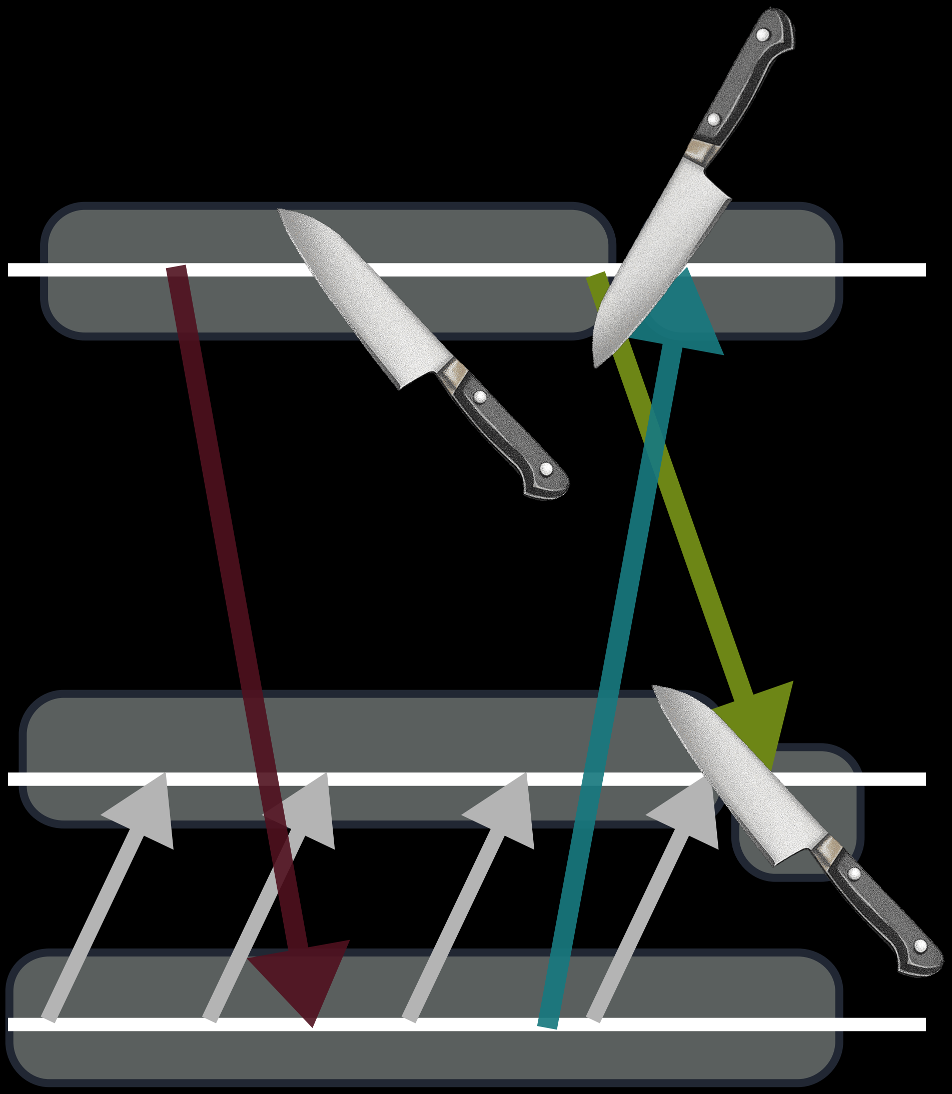
SPLIT-WIDTH 3



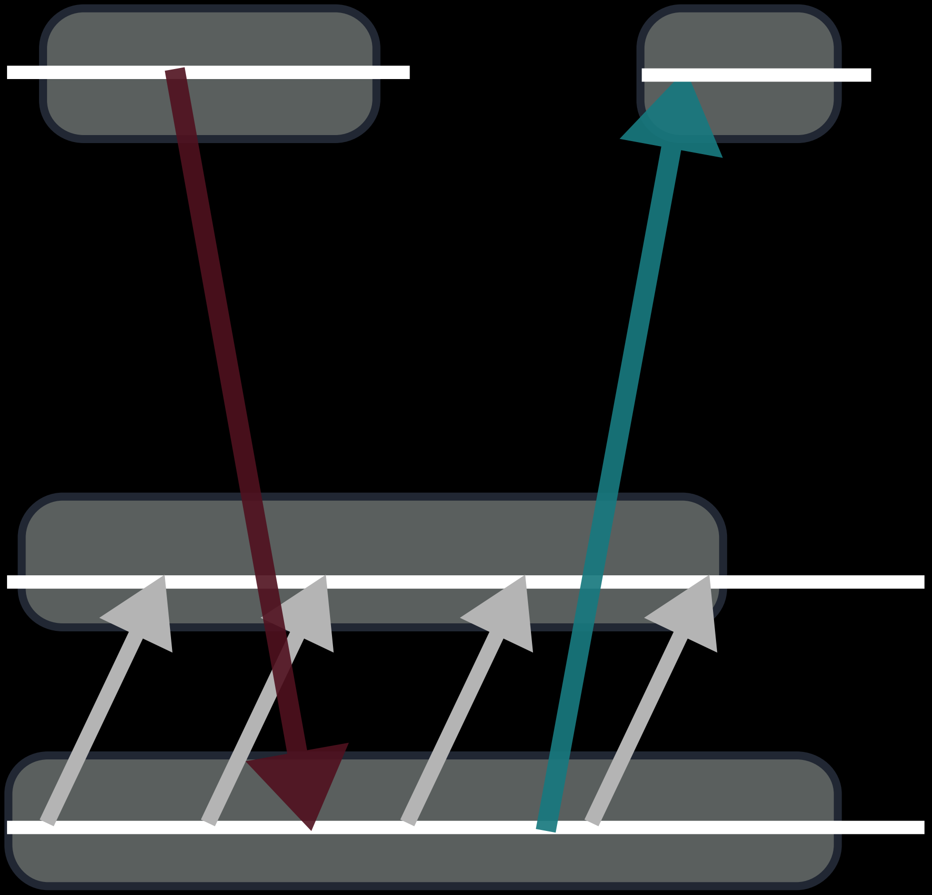
SPLIT-WIDTH 3



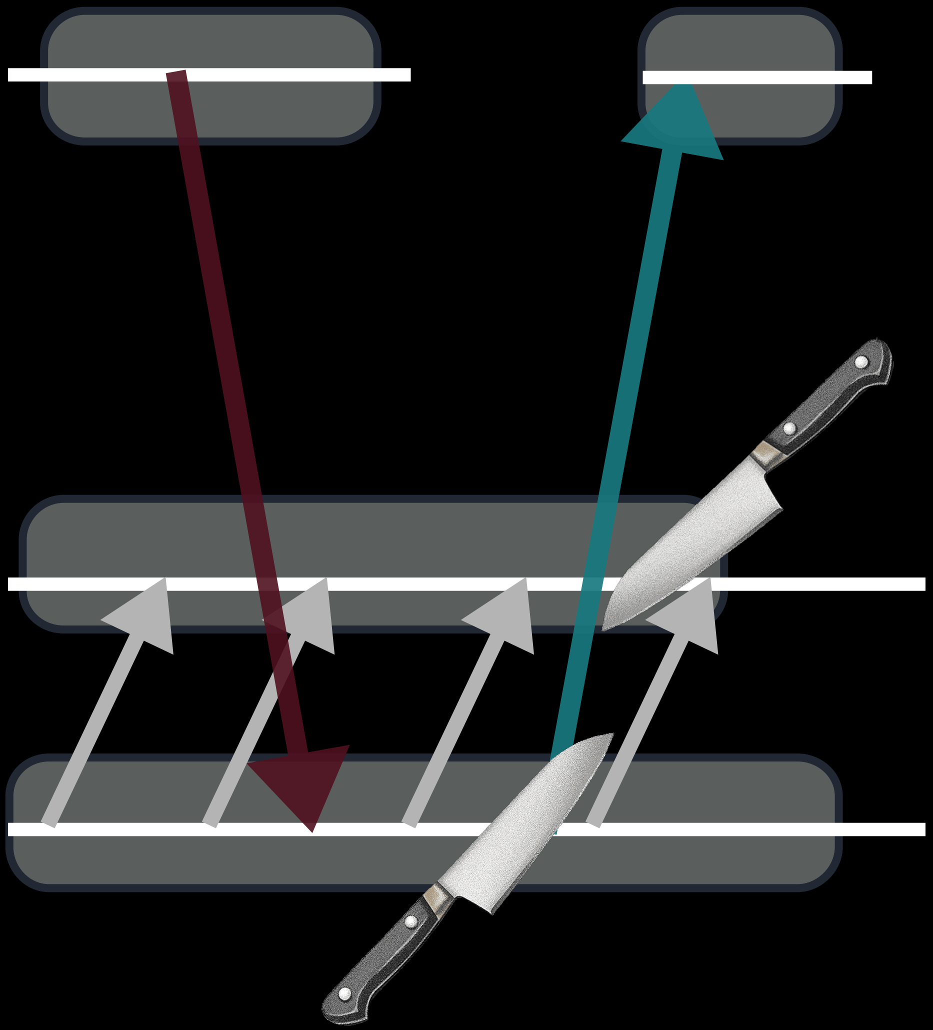
SPLIT-WIDTH 3



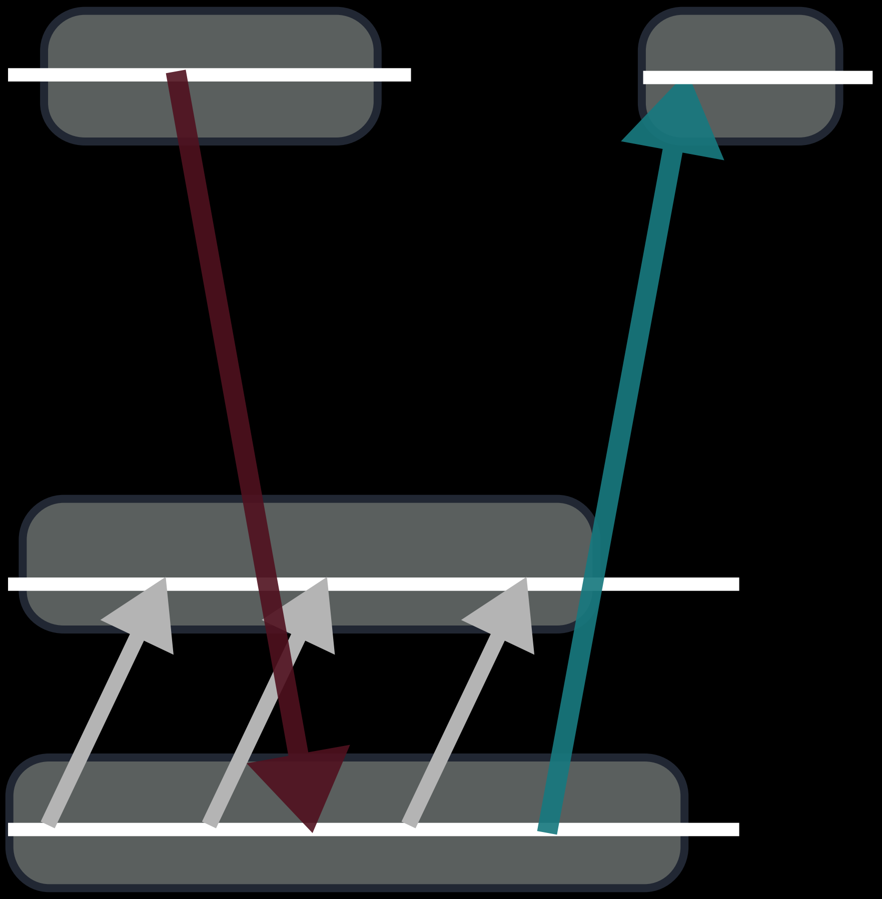
SPLIT-WIDTH 3



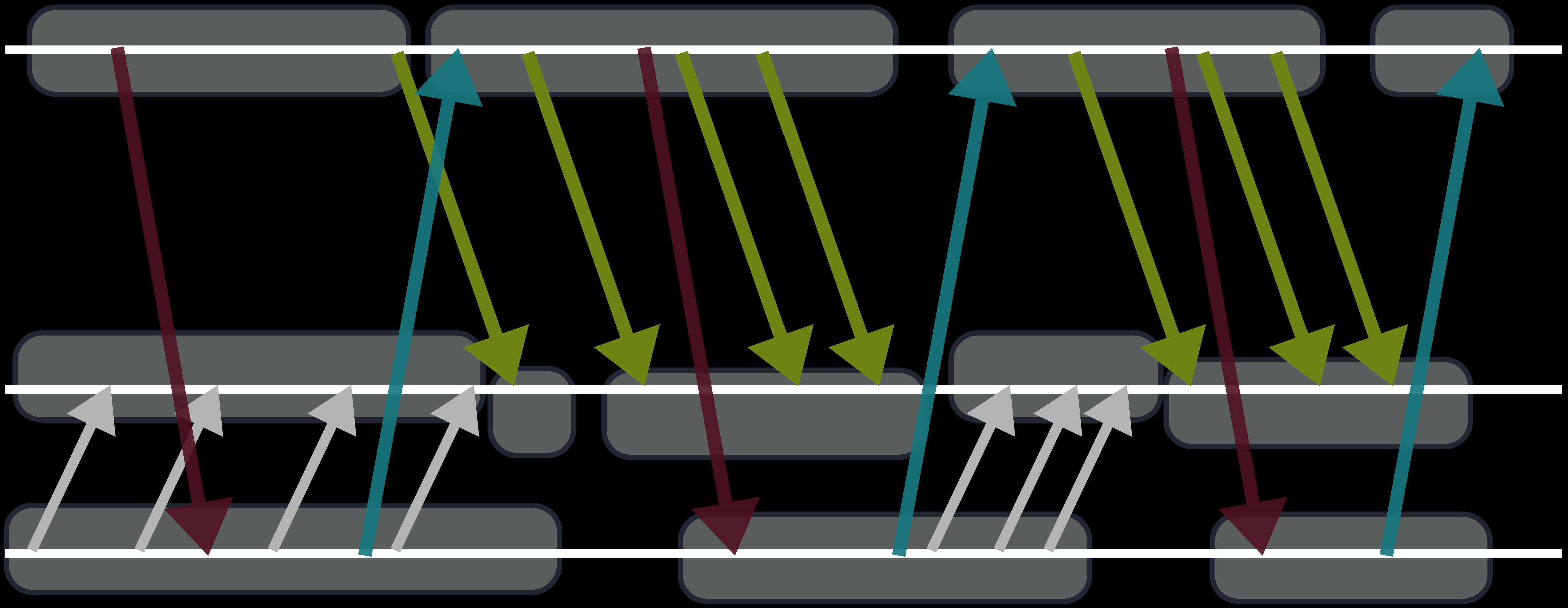
SPLIT-WIDTH 3

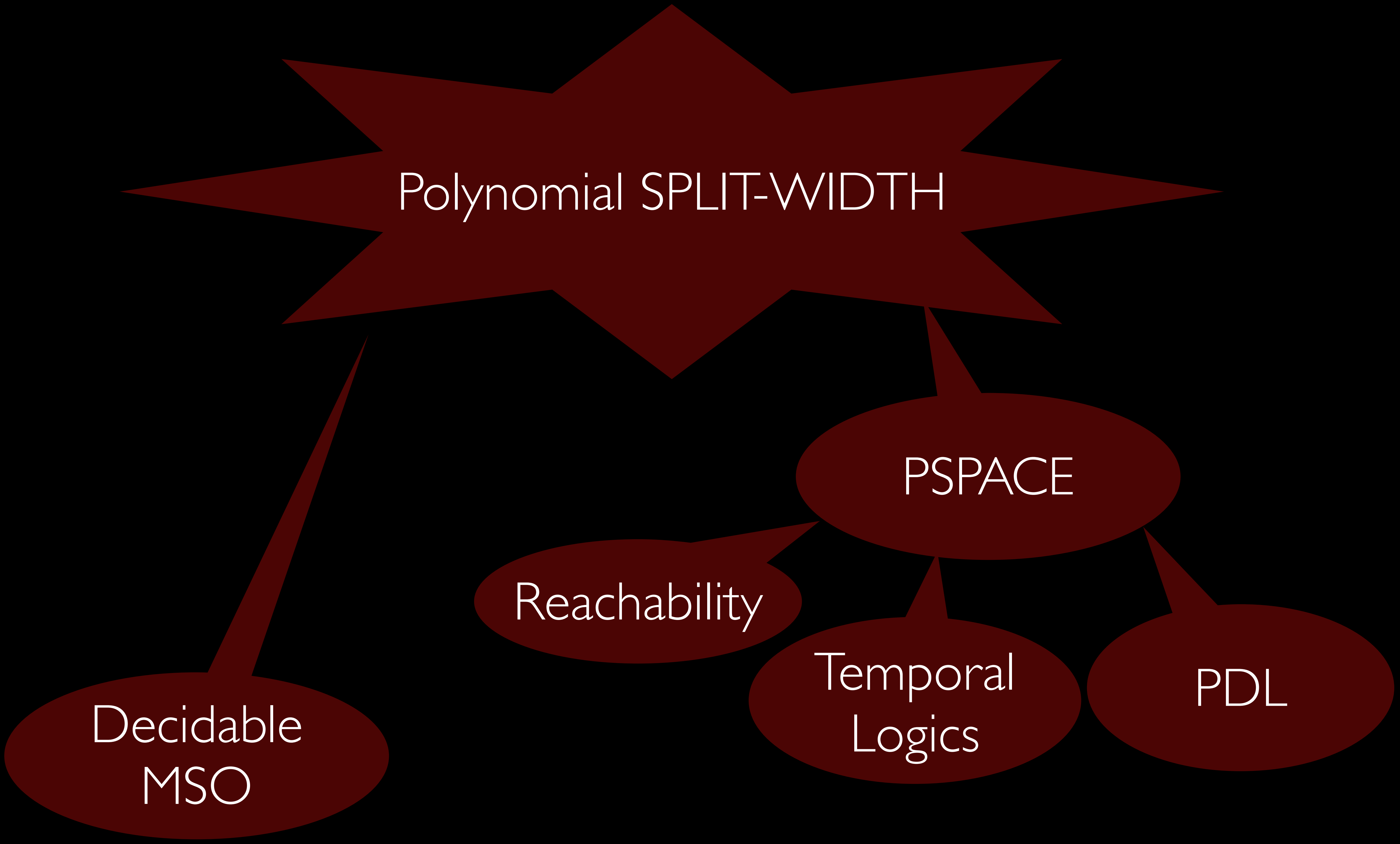


SPLIT-WIDTH 3



SPLIT-WIDTH 3





Polynomial SPLIT-WIDTH

PSPACE

Reachability

Temporal Logics

PDL

Decidable MSO

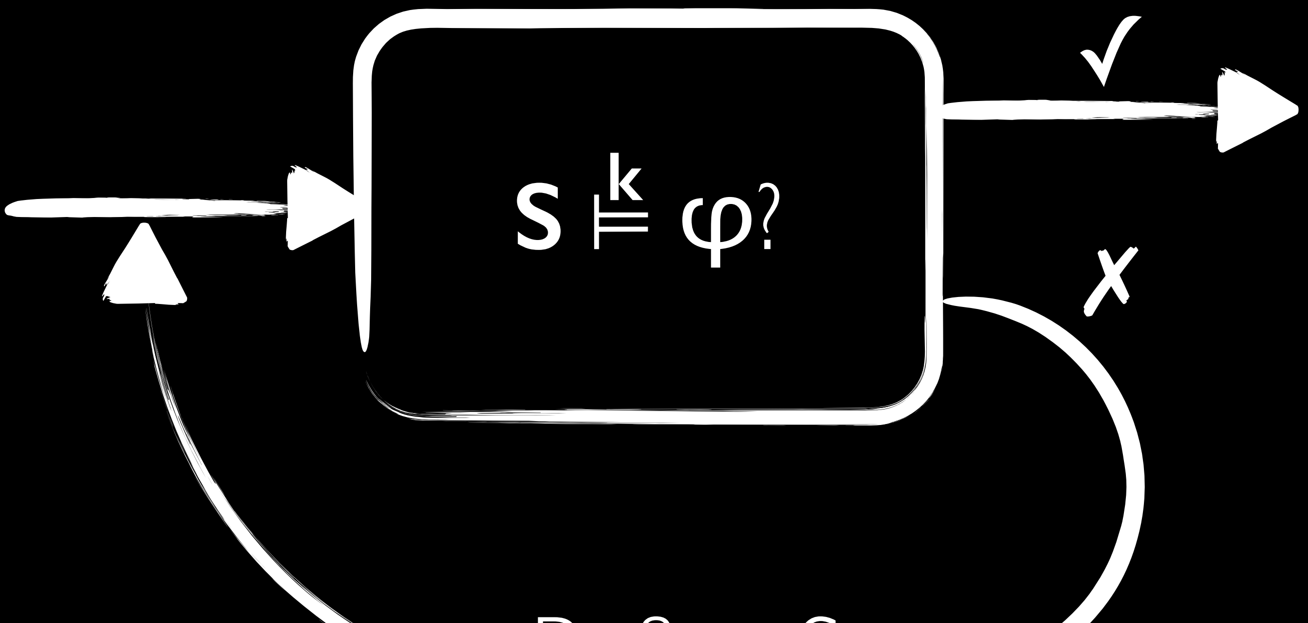
UNDER-APPROXIMATE VERIFICATION

Model Checking

> **Decidable**

System **S**

Specification **φ**



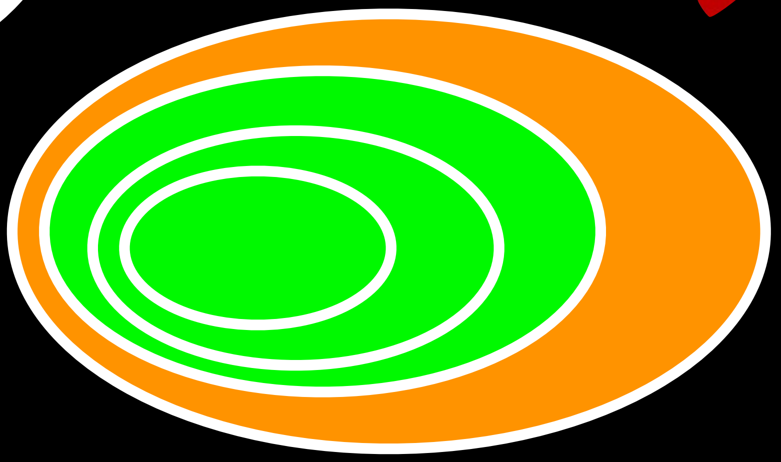
$\mathbf{S} \overset{k}{\models} \varphi$?

✓

✗

Refine S

(Fix bugs)



OTHER UNDER-APPROXIMATIONS

- Bounded channel size
- Existentially bounded [Genest et al.]
- Acyclic Architectures [La Torre et al., Heußner et al. Clemente et al.]
- Bounded context switching [Qadeer, Rehof], [LaTorre et al.], ...
- Bounded phase [LaTorre et al.]
- Bounded scope [LaTorre et al.]
- Priority ordering [Atig et al., Saivasan et al.]

OTHER UNDER-APPROXIMATIONS

- Bounded channel size
- Existentially bounded [Genest et al.]
- Acyclic Architectures [La Torre et al., Heußner et al. Clemente et al.]
- Bounded context switching [Qadeer, Rehof], [LaTorre et al.], ...
- Bounded phase [LaTorre et al.]
- Bounded scope [LaTorre et al.]
- Priority ordering [Atig et al., Saivasan et al.]

Tree-width

OTHER UNDER-APPROXIMATIONS

- Bounded channel size
- Existentially bounded [Genest et al.]
- Acyclic Architectures [La Torre et al., Heußner et al. Clemente et al.]
- Bounded context switching [Qadeer, Rehof], [LaTorre et al.], ...
- Bounded phase [LaTorre et al.]
- Bounded scope [LaTorre et al.]
- Priority ordering [Atig et al., Saivasan et al.]

Tree-width

- Many of the above classes have bounded tree-width [Parlato, Madhusudhan]

OTHER UNDER-APPROXIMATIONS

Split-width

* Acyclic Architectures	Constant
* Bounded channel size * Existentially bounded * Bounded context switching * Bounded scope	Bound + 2
* Bounded phase * Priority ordering	2^{Bound}
* Bounded Tree-width	Linear

Width: split vs tree vs clique

Split-Width k

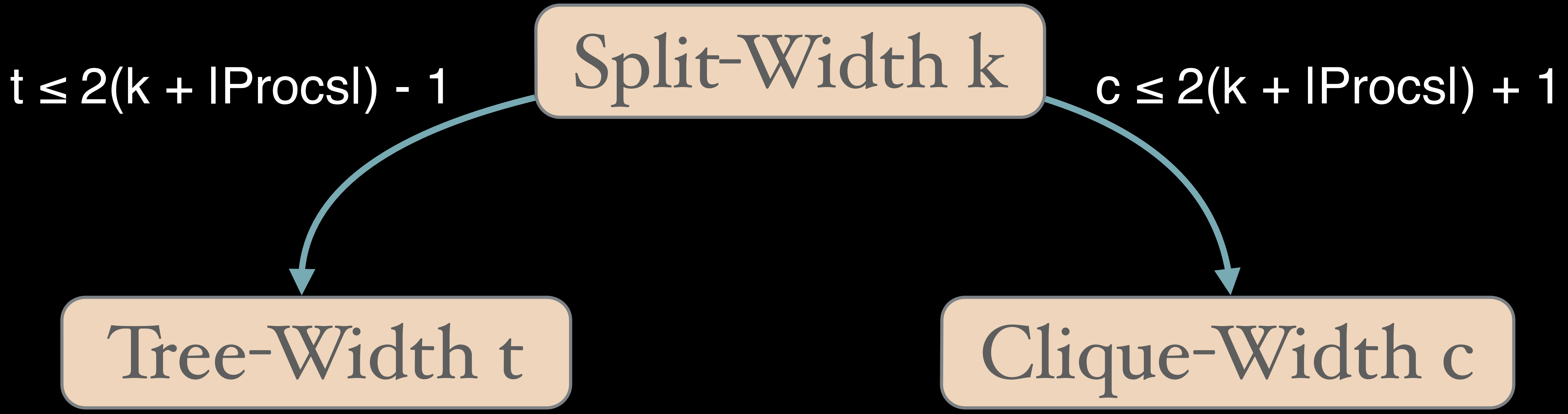
Tree-Width t

Clique-Width c

Let C be a class of bounded degree MSO definable graphs.
TFAE

1. C has a decidable MSO theory
2. C can be interpreted in binary trees
3. C has bounded tree-width
4. C has bounded clique-width
5. C has bounded split-width (for concurrent recursive behaviors)

Width: split vs tree vs clique


Split-Width k

$t \leq 2(k + |Procs|) - 1$

$c \leq 2(k + |Procs|) + 1$

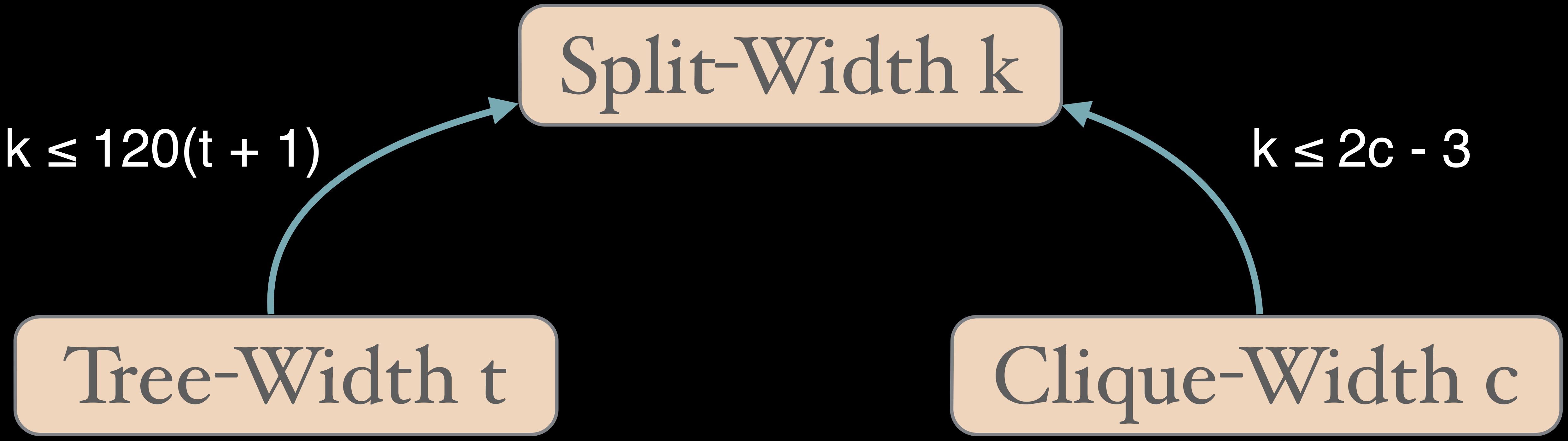
Tree-Width t

Clique-Width c

Let C be a class of bounded degree MSO definable graphs. TFAE

1. C has a decidable MSO theory
2. C can be interpreted in binary trees
3. C has bounded tree-width
4. C has bounded clique-width
5. C has bounded split-width (for concurrent recursive behaviors)

Width: split vs tree vs clique



Split-Width k

k ≤ 120(t + 1)

k ≤ 2c - 3

Tree-Width t

Clique-Width c

Let C be a class of bounded degree MSO definable graphs. TFAE

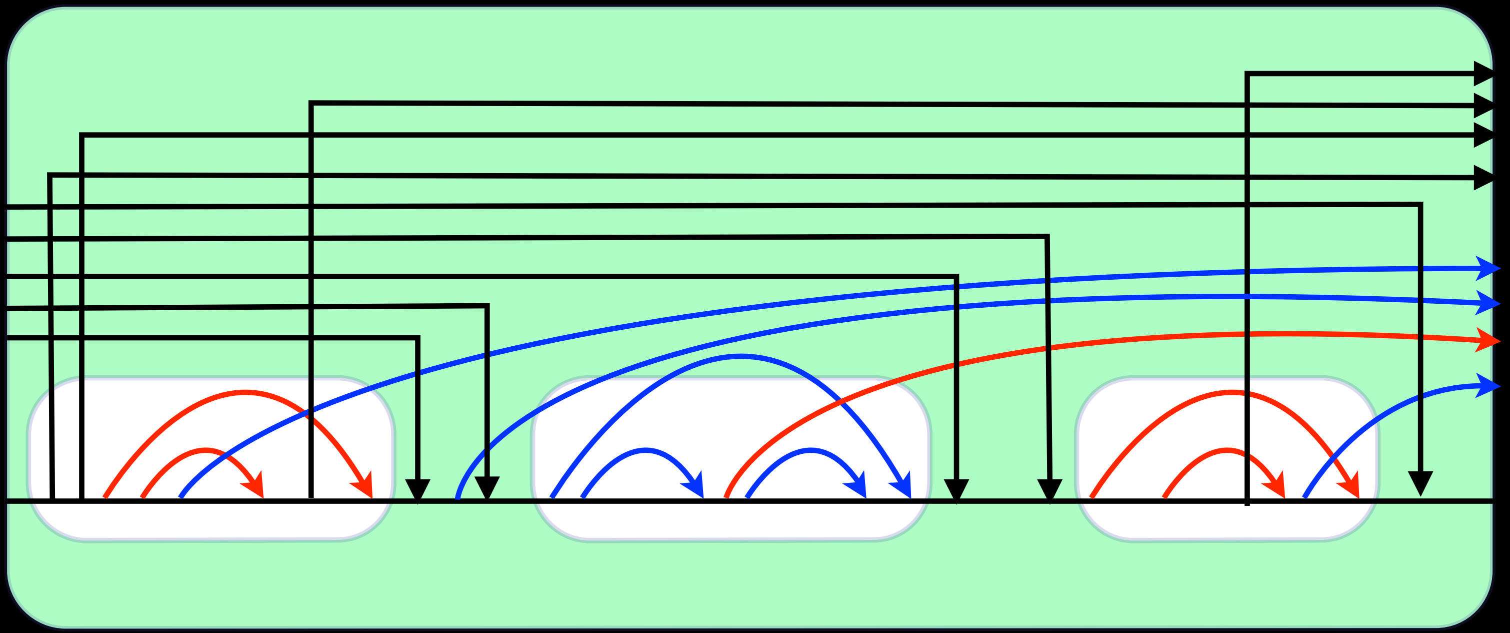
1. C has a decidable MSO theory
2. C can be interpreted in binary trees
3. C has bounded tree-width
4. C has bounded clique-width
5. C has bounded split-width (for concurrent recursive behaviors)

COMMUNICATING RECURSIVE PROGRAMS: CONTROL AND SPLIT-WIDTH

AUTONOMOUS COMPUTATIONS

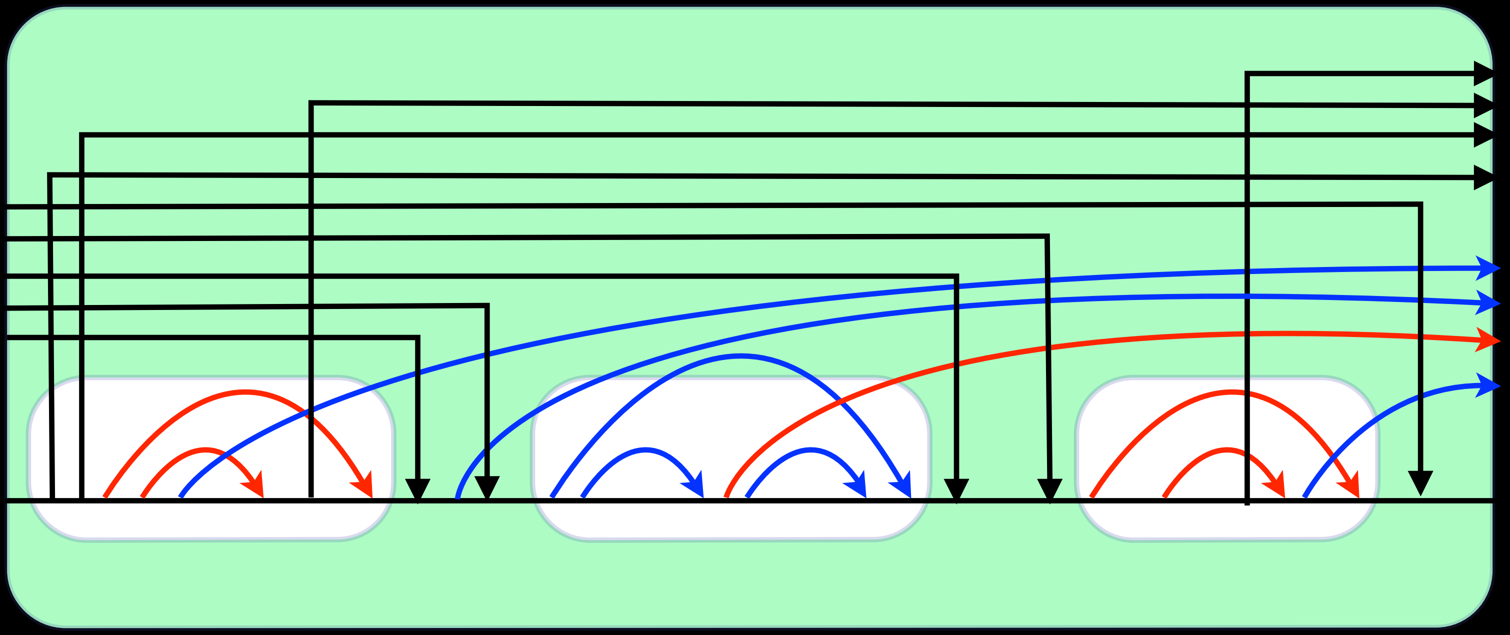
- Recursive computations which does not read from other stacks/queues.
- A stretch of computation in which all incoming edges are on a single stack

AUTONOMOUS COMPUTATIONS

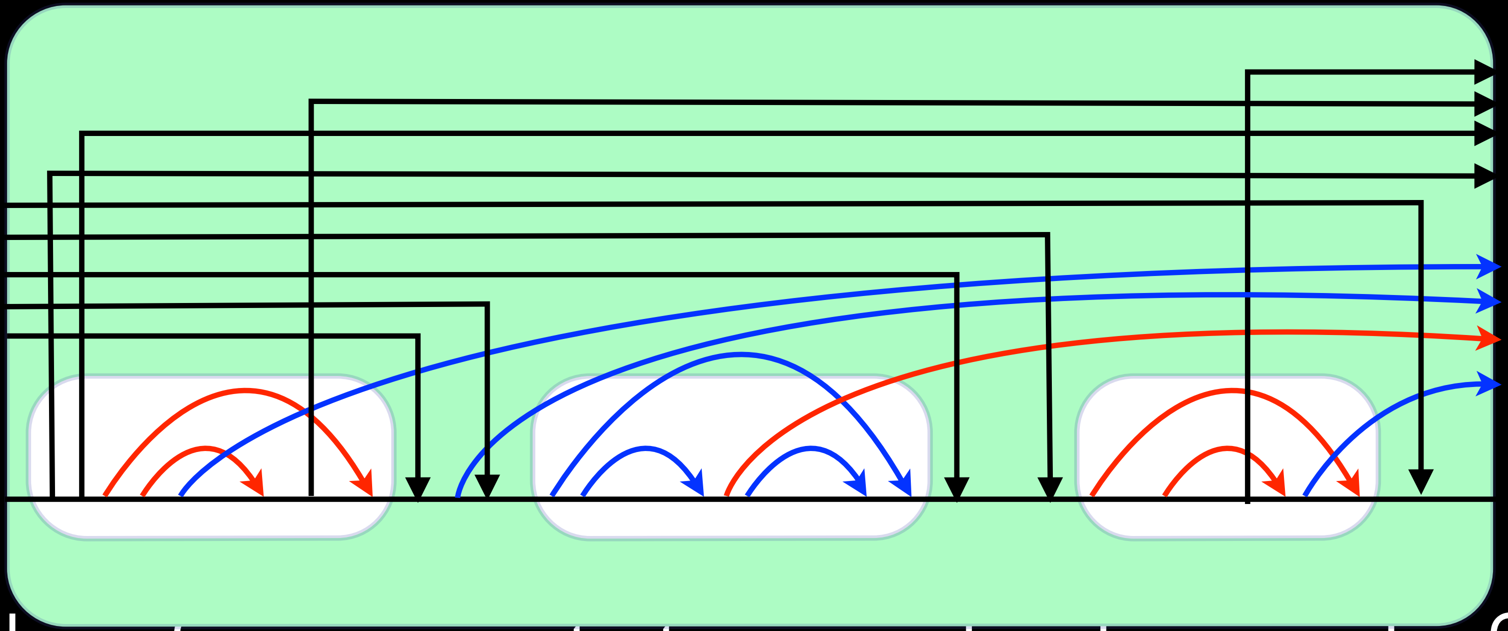


- Recursive computations which does not read from other stacks/queues.
- A stretch of computation in which all incoming edges are on a single stack

PHASE

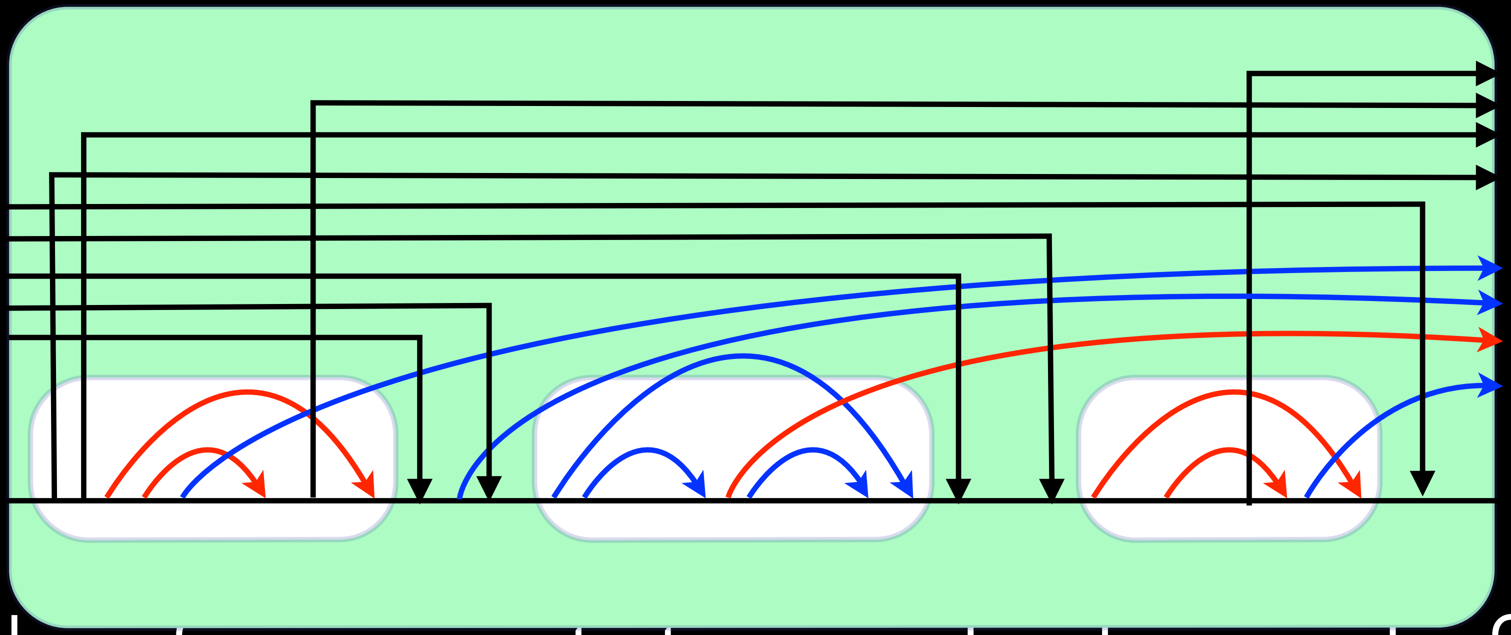


PHASE



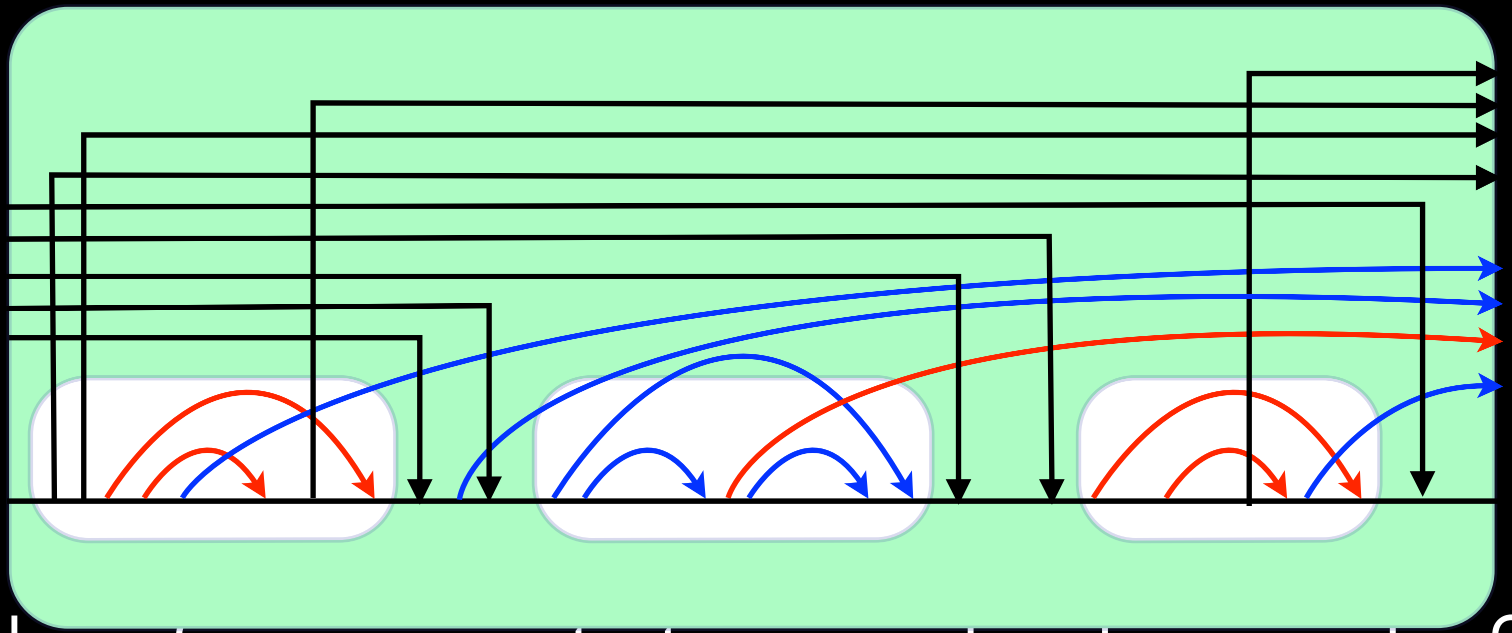
- A stretch of computation which reads from at most one stack/queue

PHASE



- A stretch of computation which reads from at most one stack/queue
- free (unlimited) autonomous computations

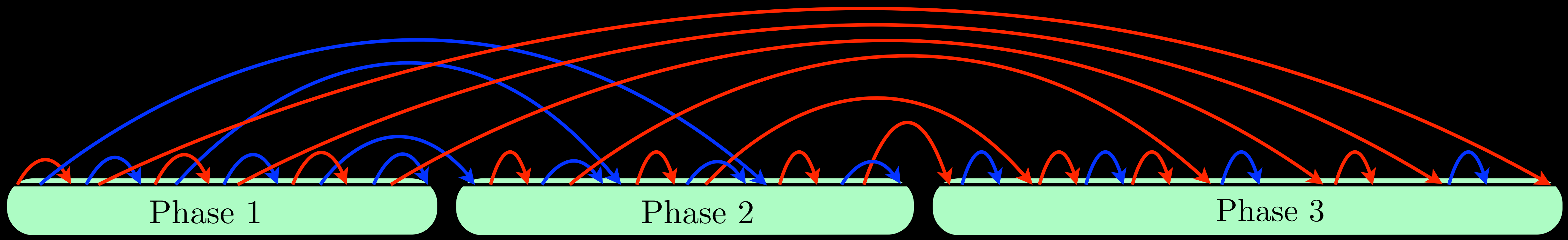
PHASE



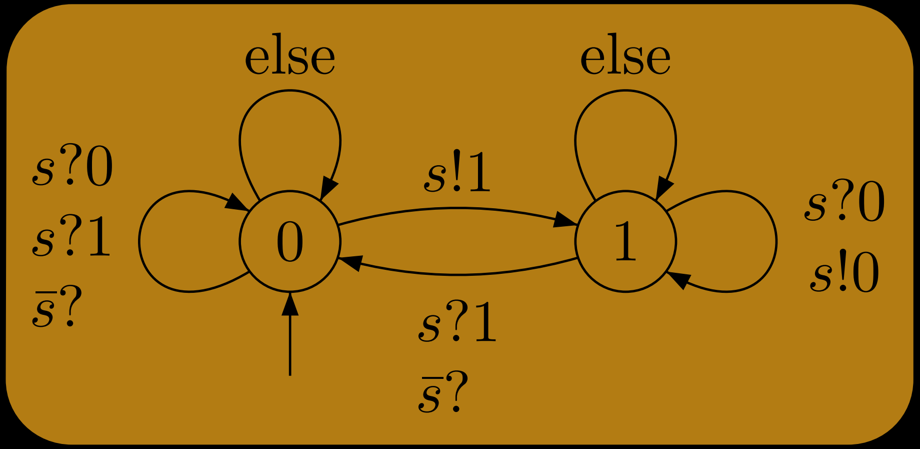
- A stretch of computation which reads from at most one stack/queue
- free (unlimited) autonomous computations
- no loops

K-BOUNDED PHASE

K-BOUNDED PHASE



IDENTIFYING AUTONOMOUS POPS



- Possible by tagging the values on stacks
- Deterministic controller for each stack
- The phase controller simulates one such automaton for each stack.

COMMUNICATING RECURSIVE PROGRAMS: CONTROL AND SPLIT-WIDTH

- C. A., Paul Gastin, and K. Narayan Kumar. Verifying communicating multi pushdown systems via Split-width. In *ATVA* 2014.
- C. A., Paul Gastin, and K. Narayan Kumar. Controllers for the verification of communicating multi-pushdown systems. In *CONCUR* 2014.
- A. C., Paul Gastin, and K. Narayan Kumar. MSO decidability of multi-pushdown systems via Split-width. In *CONCUR* 2012.
- A. C. *Verification of Communicating Recursive Programs via Split-width*. PhD thesis, ENS Cachan, 2014.