Using Graph Decompositions to Verify Concurrent Recursive Programs

K Narayan Kumar

Chennai Mathematical Institute, India.

IMS, Singapore, September 2016

Concurrent Recursive Programs

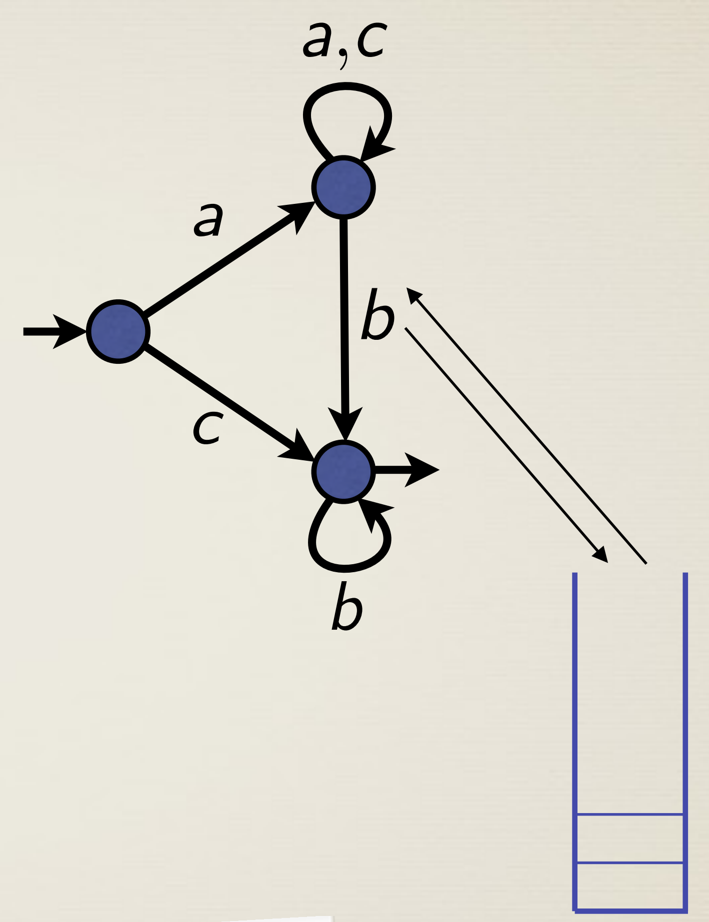
Multi-threaded or Distributed

Variables range over finite domains

Functions can be recursive

Modeling Recursion

```
func f1
{while <true>
{call f1 OR
a OR
exit;}
return;}
```



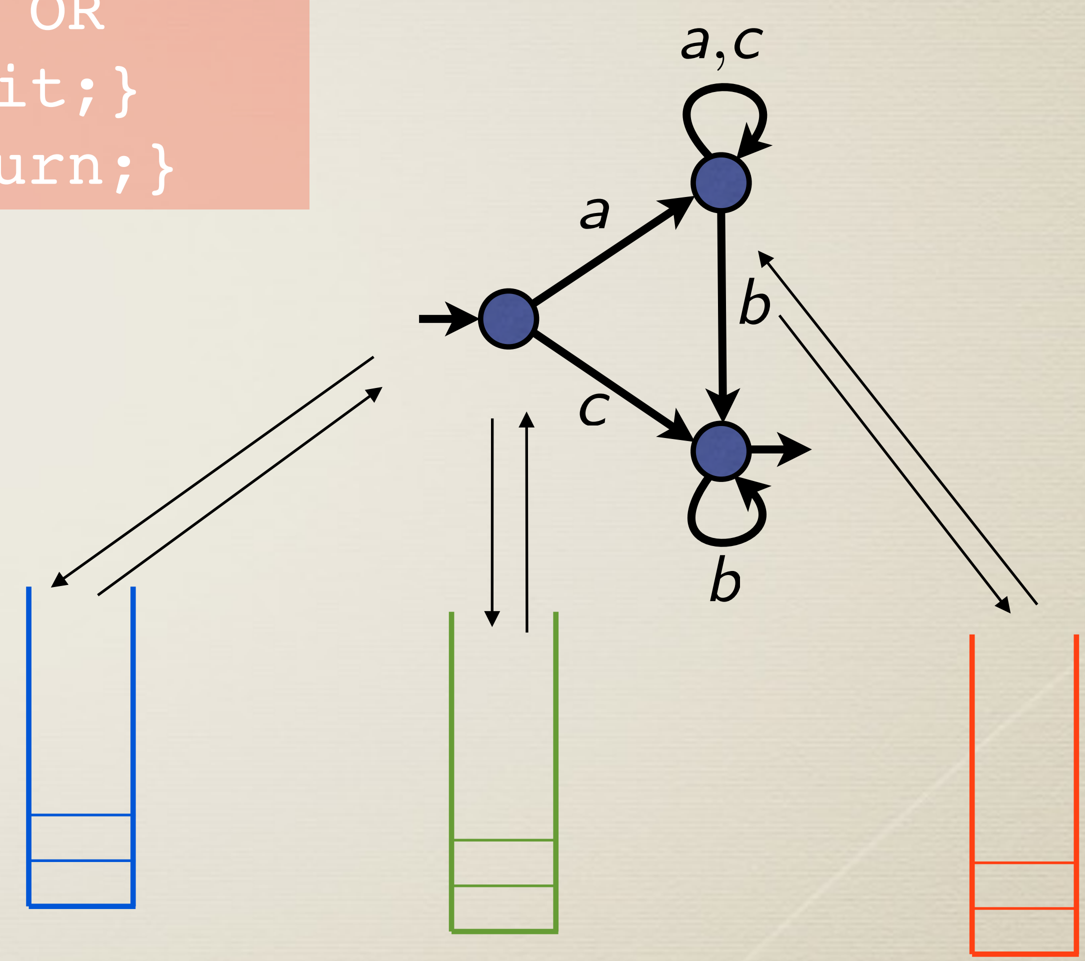
Recursive Programs are Pushdown Systems

Modeling Recursion …

```
func f1
{while <true>
{call f1 OR
a OR
exit;}
return;}
```

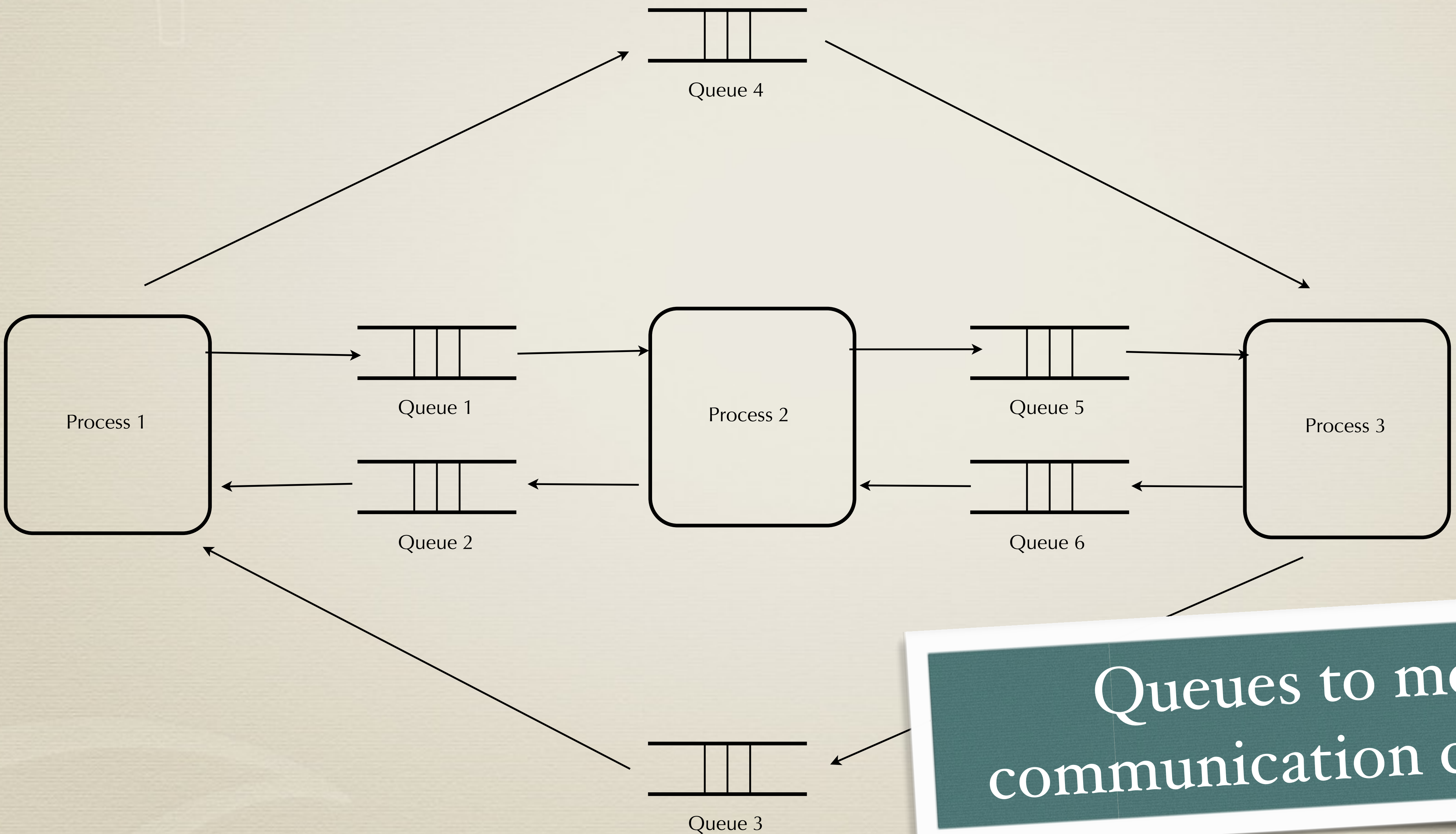
```
func f2
{while <true>
{call f2 OR
a OR
exit;}
return;}
```

```
func f3
{while <true>
{call f3 OR
a OR
exit;}
return;}
```



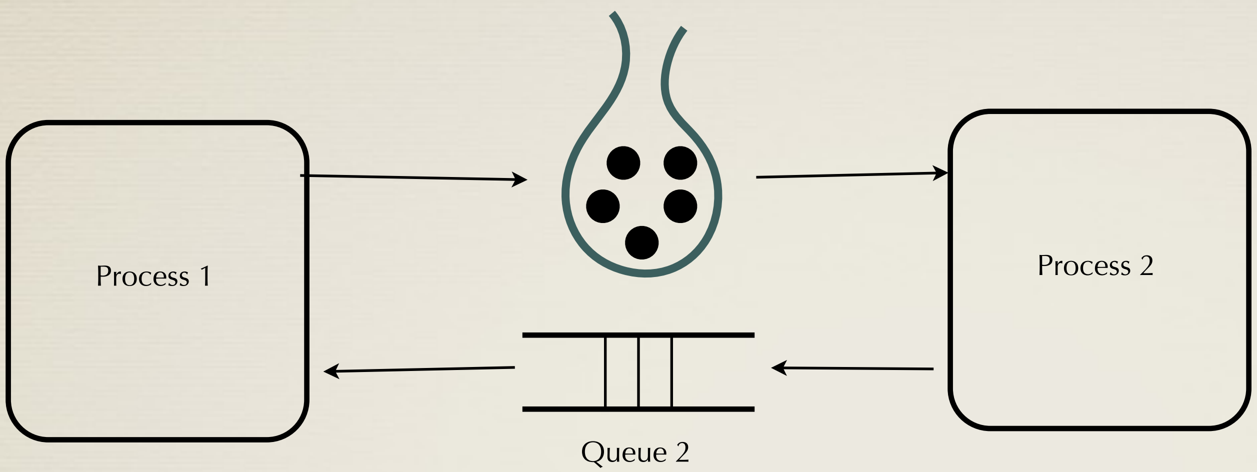
Multi-threaded Programs are Multi-Pushdown Systems

Concurrent Communicating Programs



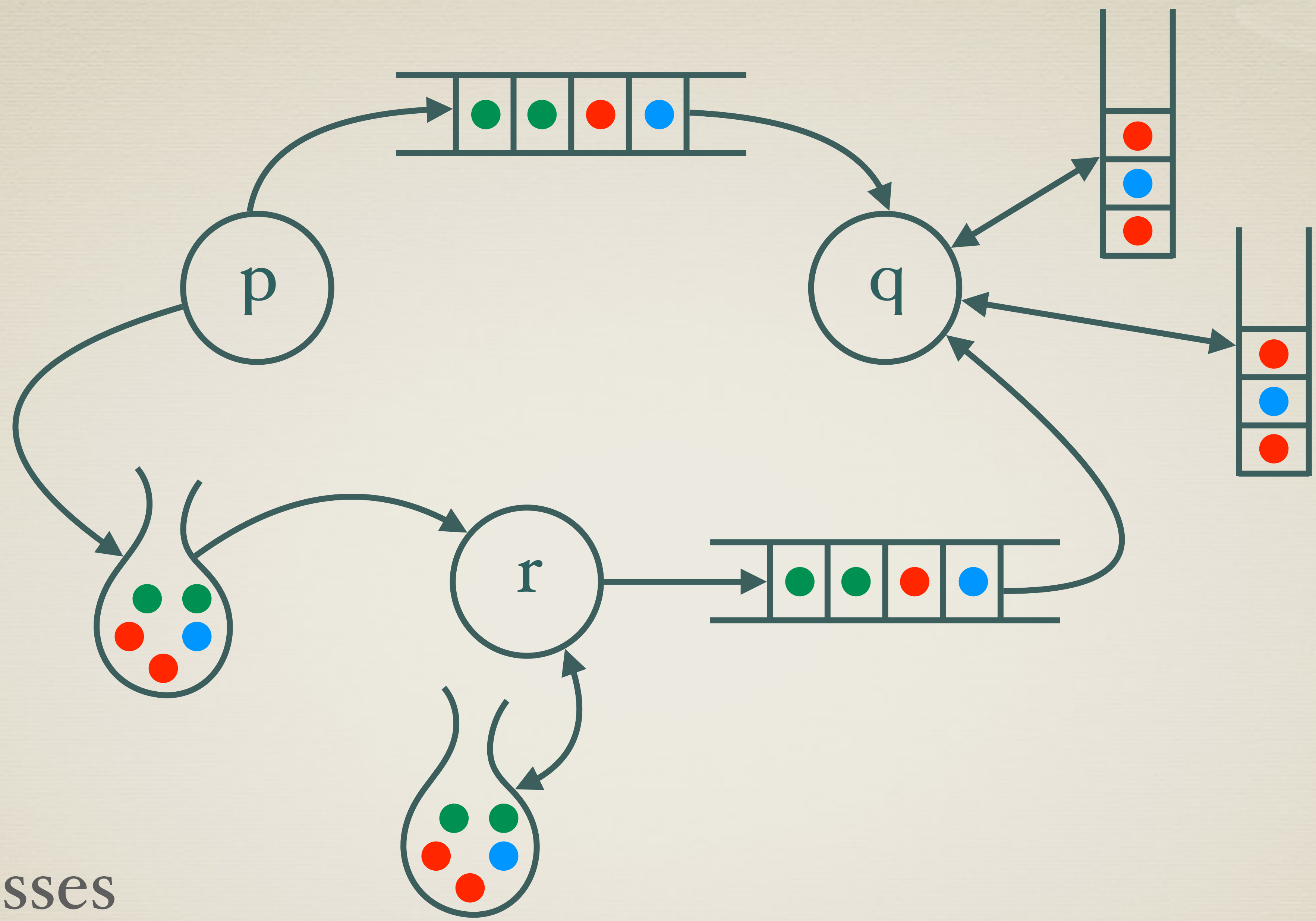
Queues to model communication channels

Unordered Channels



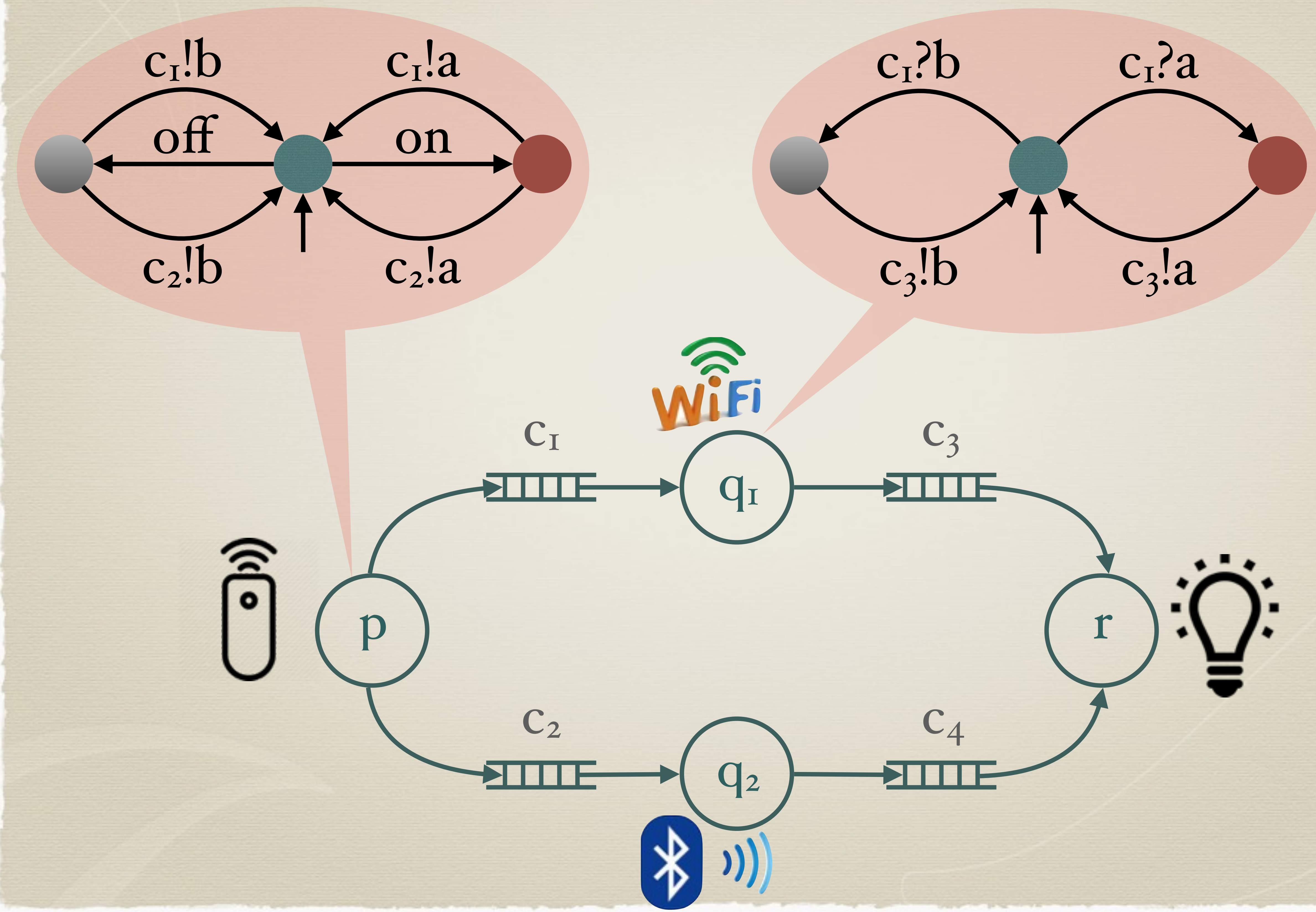
Bags to model unordered communication channels

System: Concurrent Processes with Data-Structures



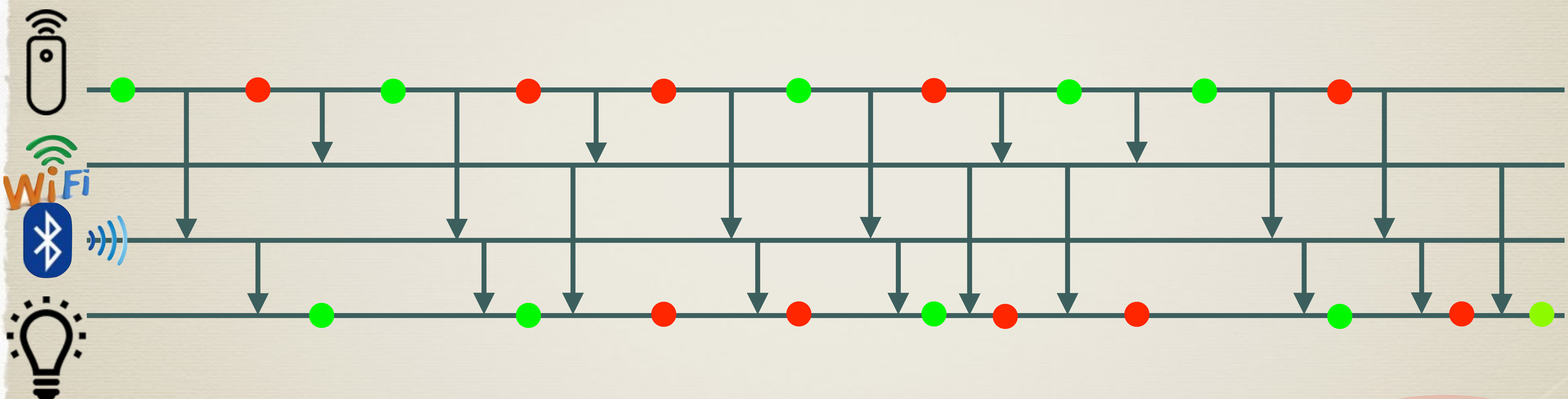
- Processes
- Data structures
  - Stacks: recursive programs, multithreaded
  - Queues: communication (FIFO)
  - Bags: communication (unordered)

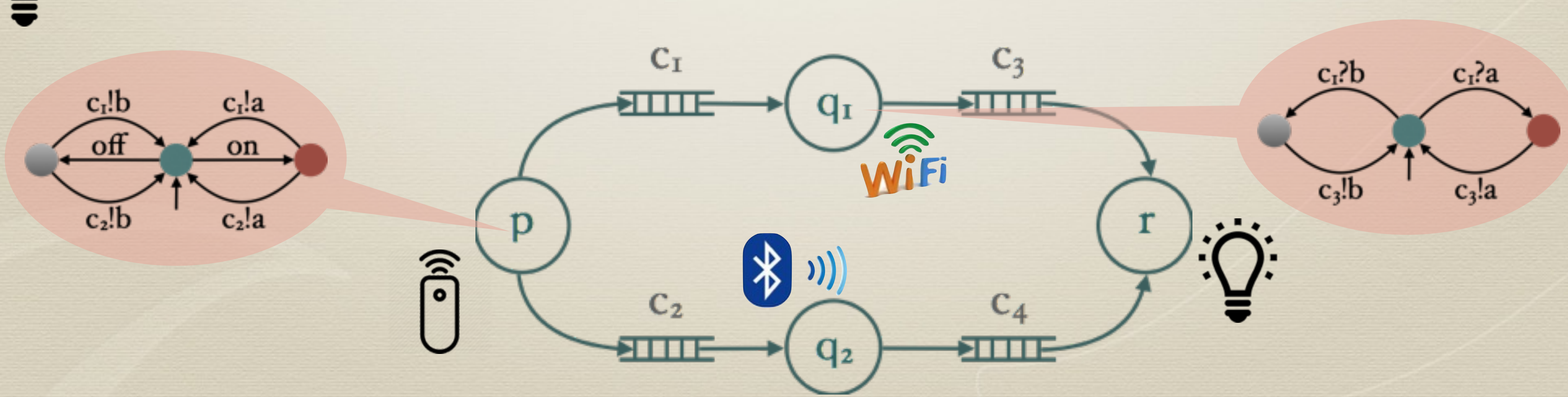
System: An Example



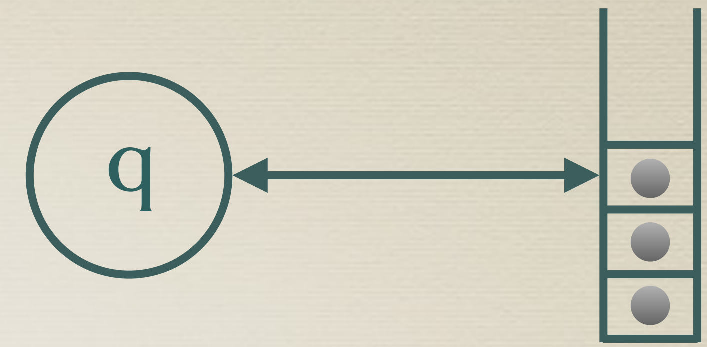
Behaviours as Graphs

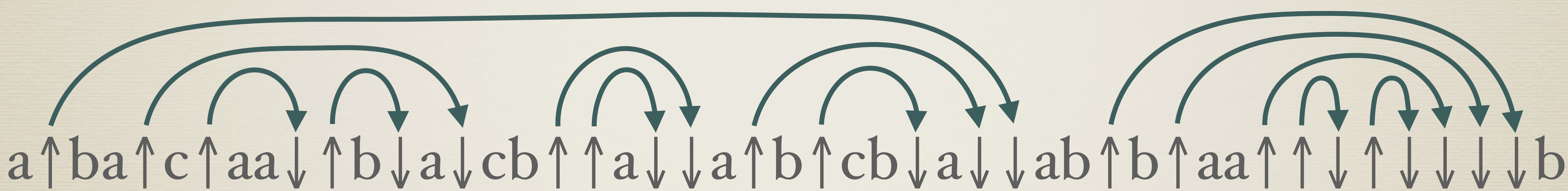
Message Sequence Charts





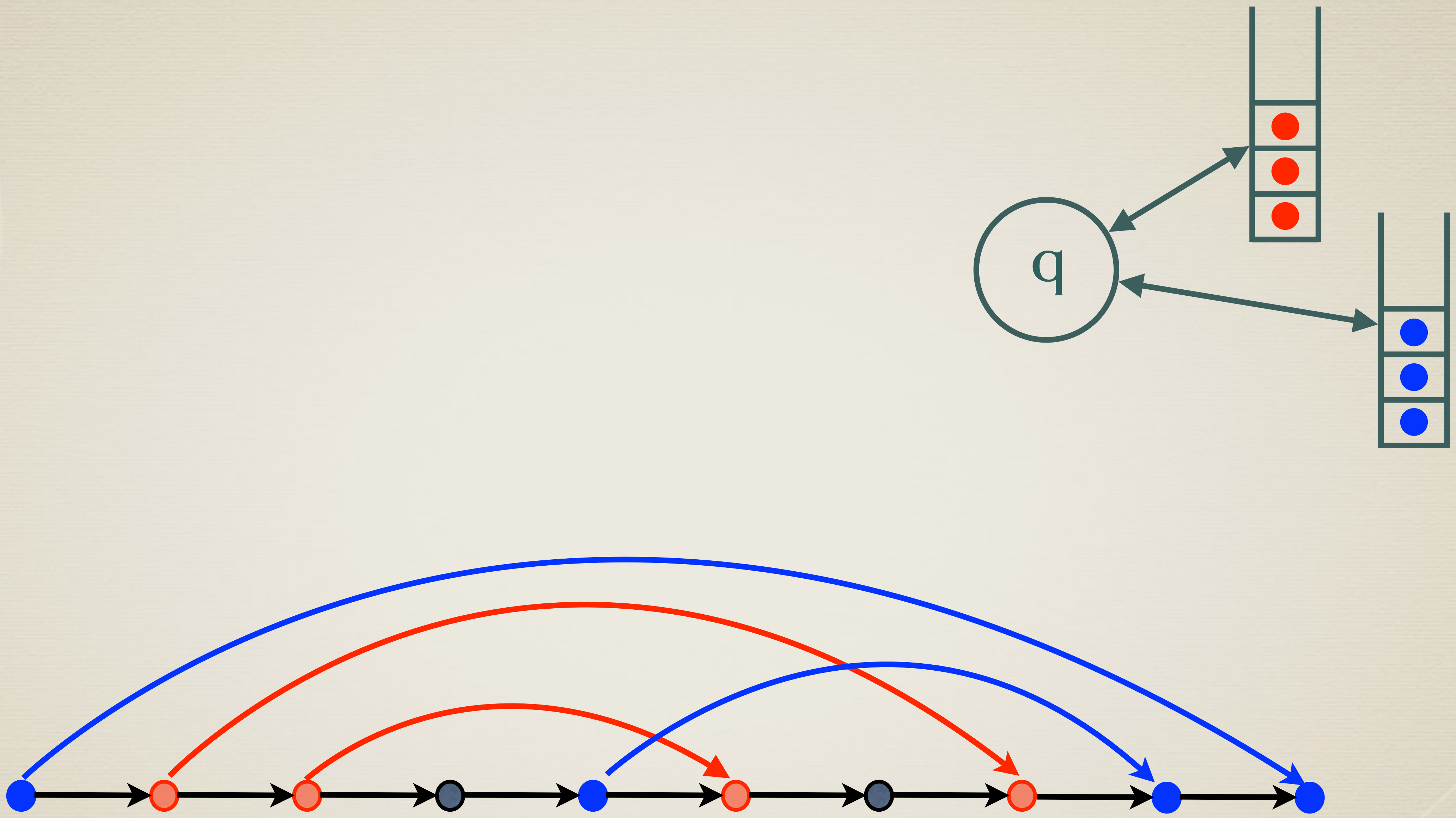
Graphs for Sequential Systems





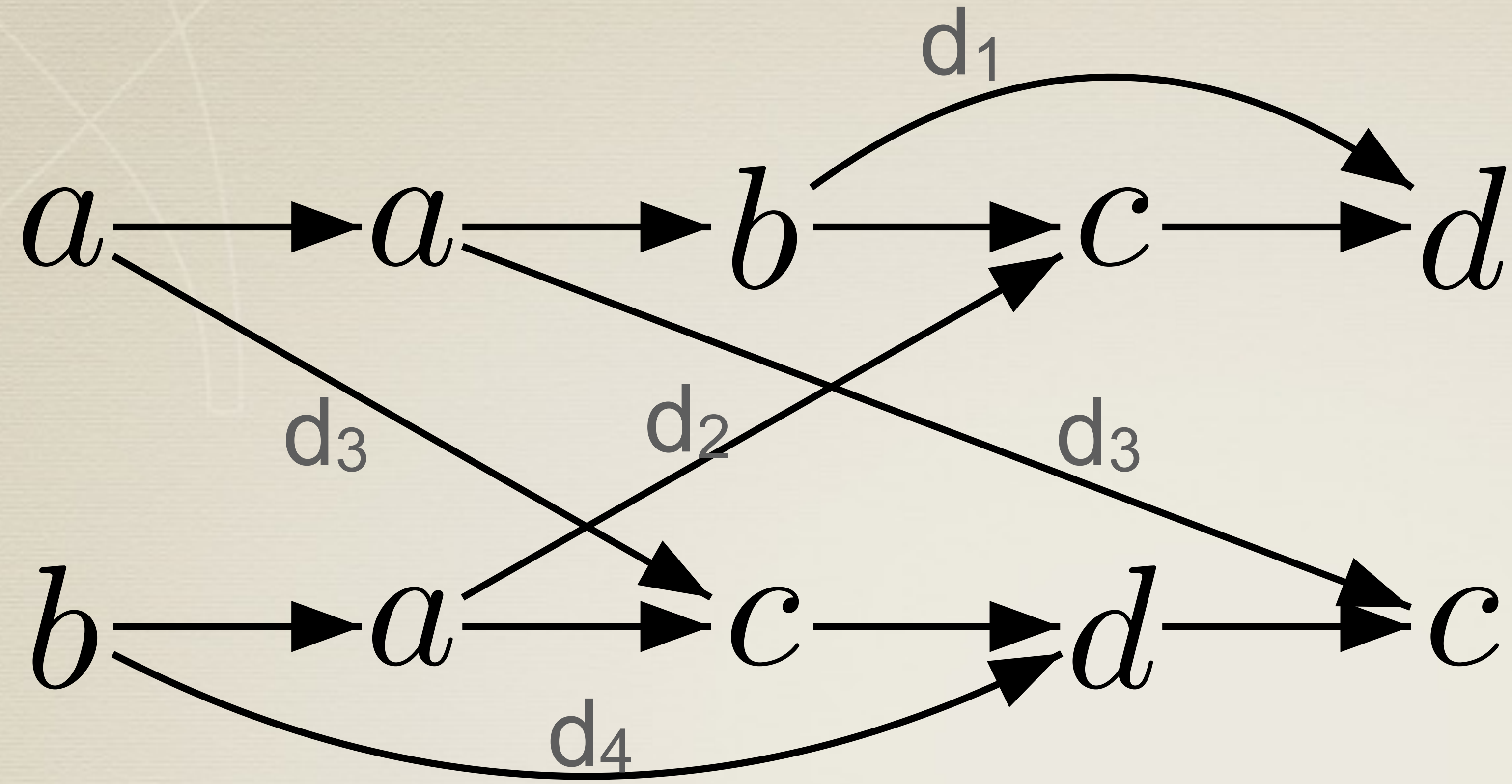
Nested Words
Alur, Madhusudan, 2009

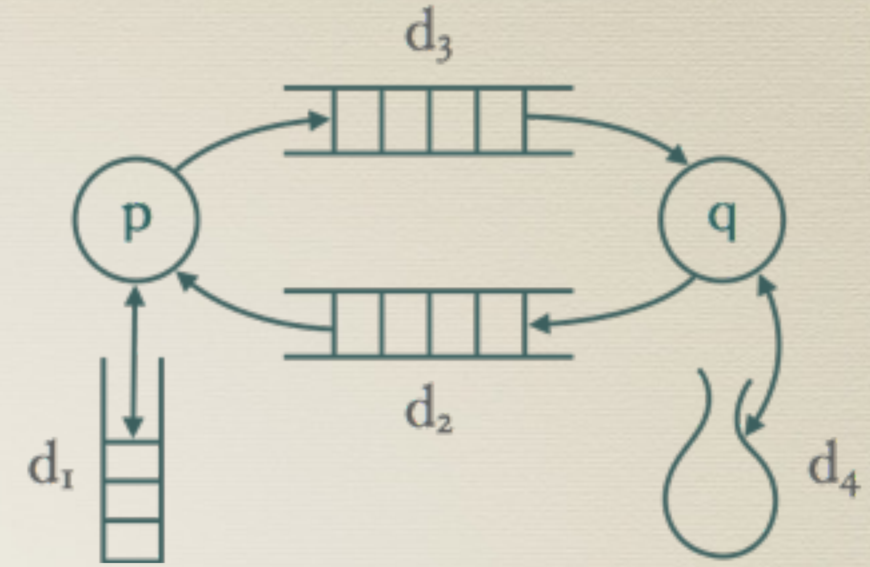
Graphs for Multi-threaded Systems



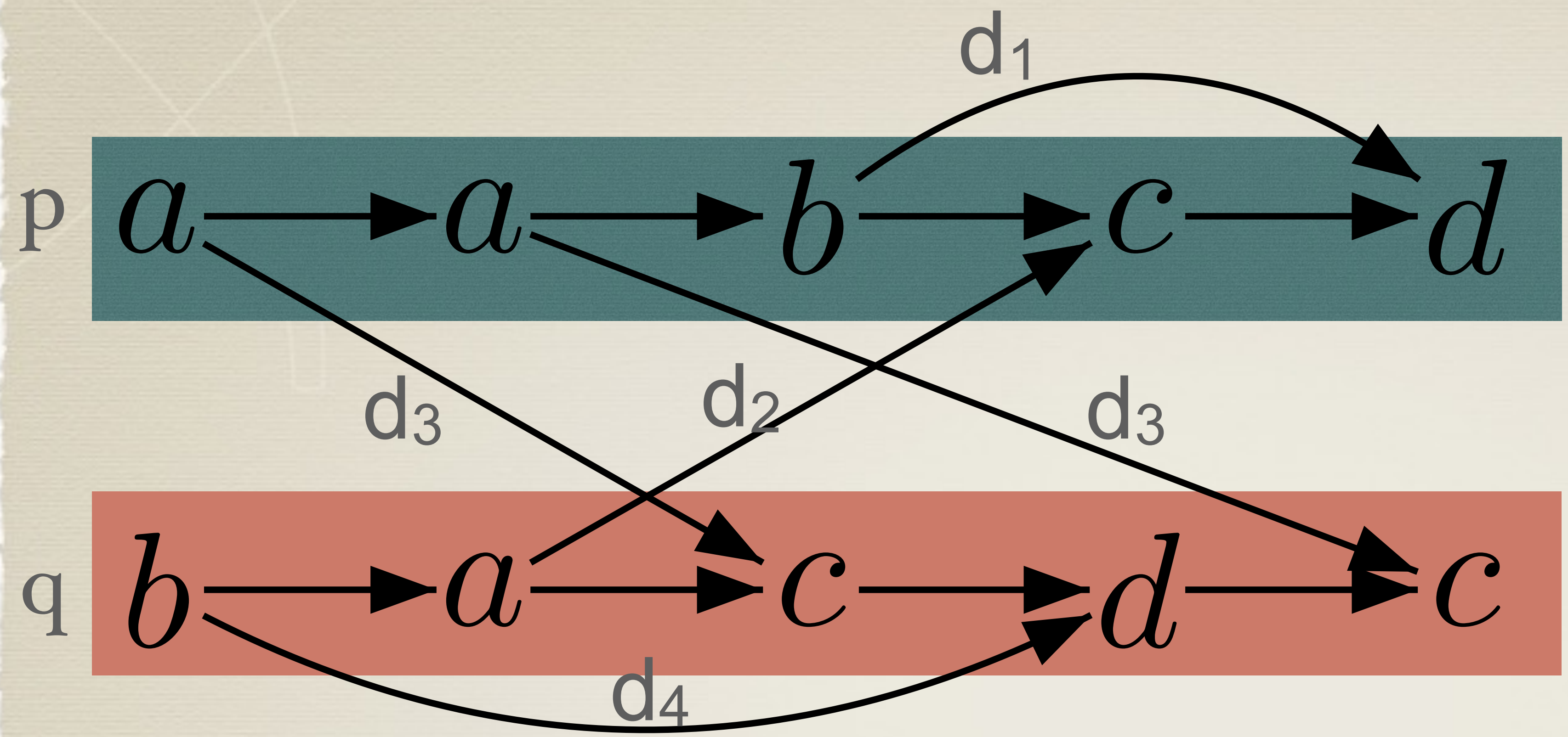
Multiply Nested Words

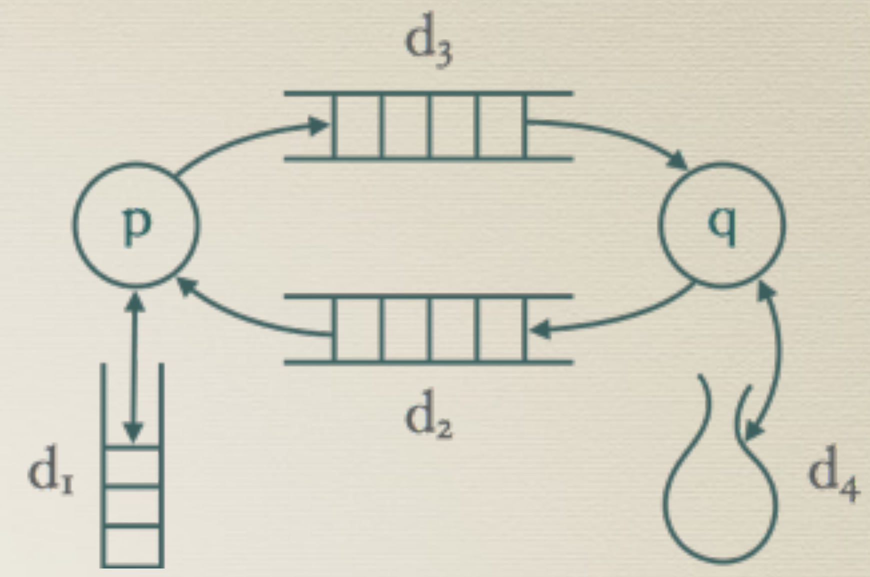
Concurrent Behaviour with Matching (CBM)





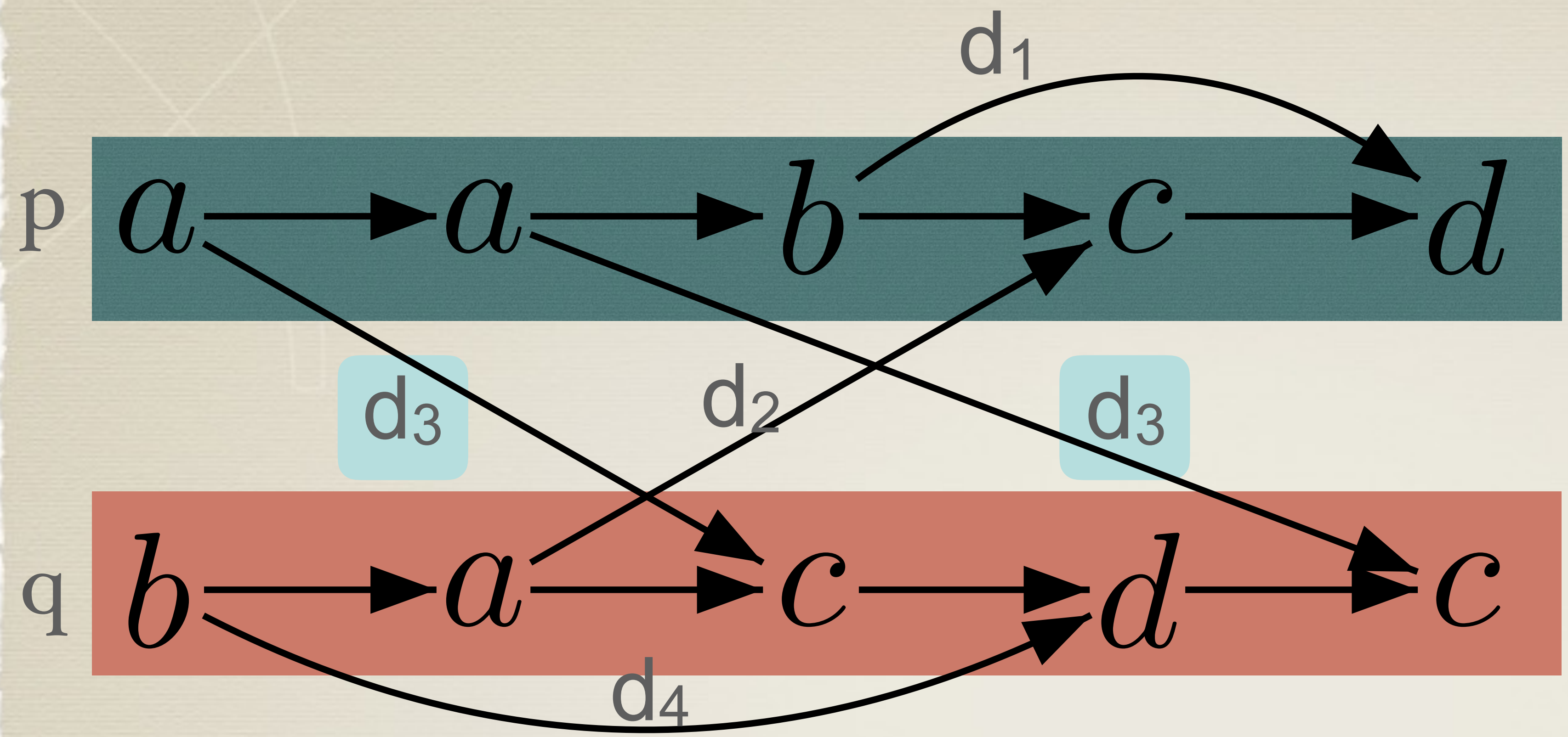
Concurrent Behaviour with Matching (CBM)

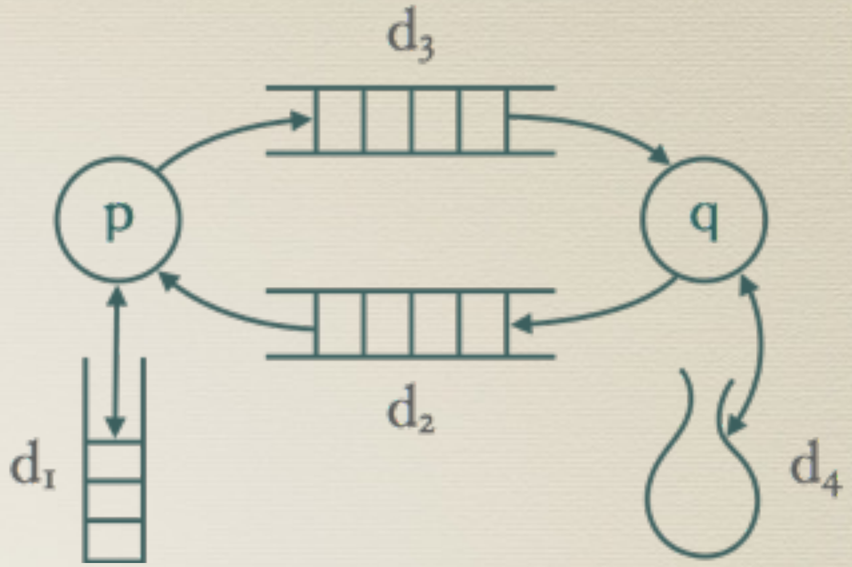




- A linear order (or path) for each process
- Edges labeled with data structures

Concurrent Behaviour with Matching (CBM)



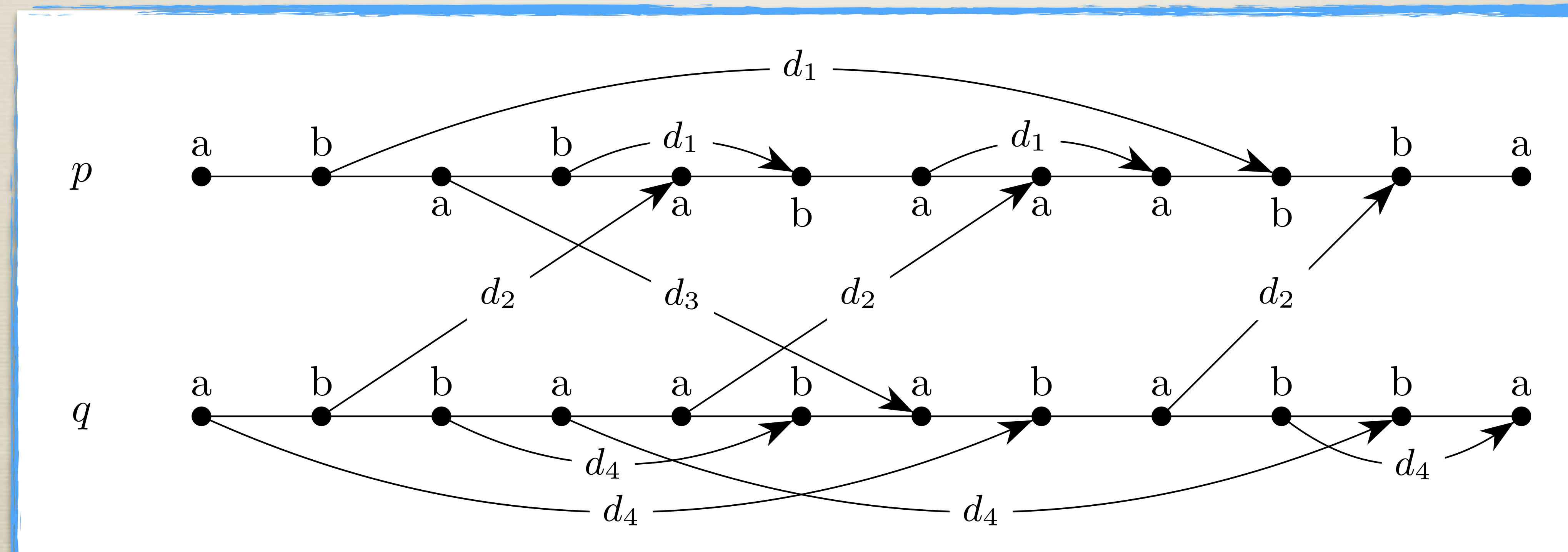


- A linear order (or path) for each process
- Edges labeled with data structures
  - Communication edges form a matching
  - Edge labelled d relates the writer and reader of d
  - Edges follow the discipline of the data structure
    - LIFO/FIFO/Bag

Specification over CBMs

MSO: Monadic Second Order Logic

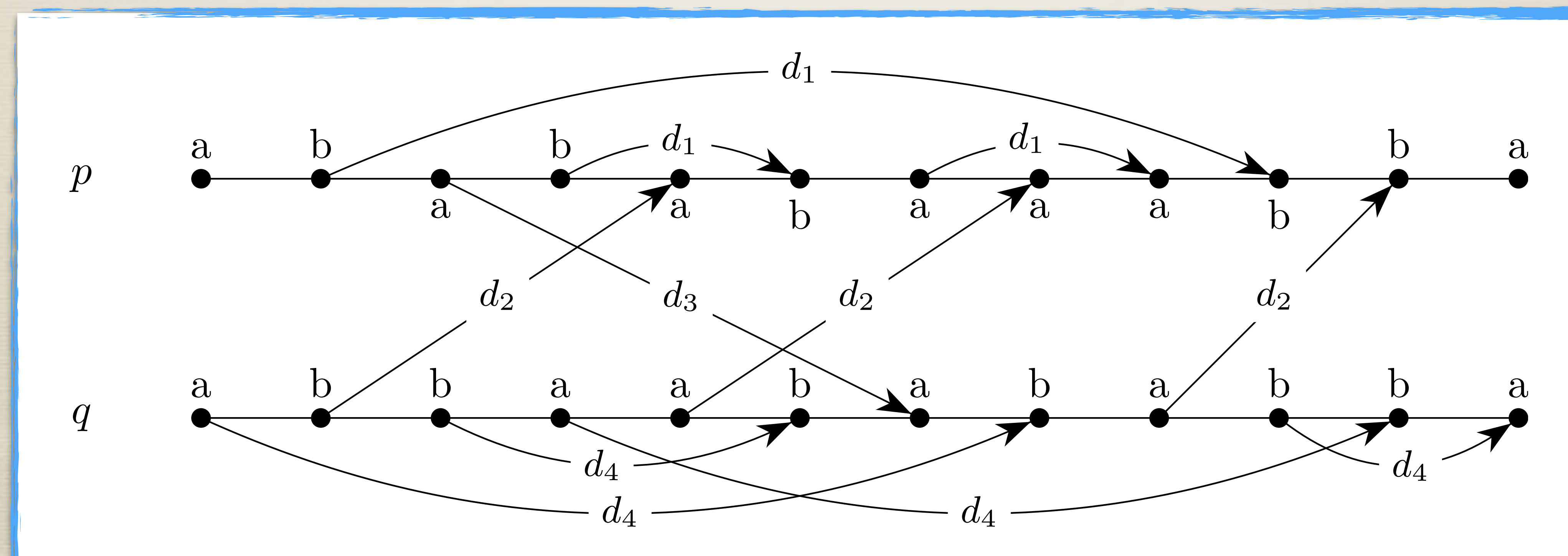
$$\varphi ::= \mathsf{false} \mid a(x) \mid p(x) \mid x \le y \mid x \rhd^{d} y \mid x \to y$$
$$\mid x \in X \mid \varphi \vee \varphi \mid \neg\varphi \mid \exists x\, \varphi \mid \exists X\, \varphi$$



Specification over CBMs

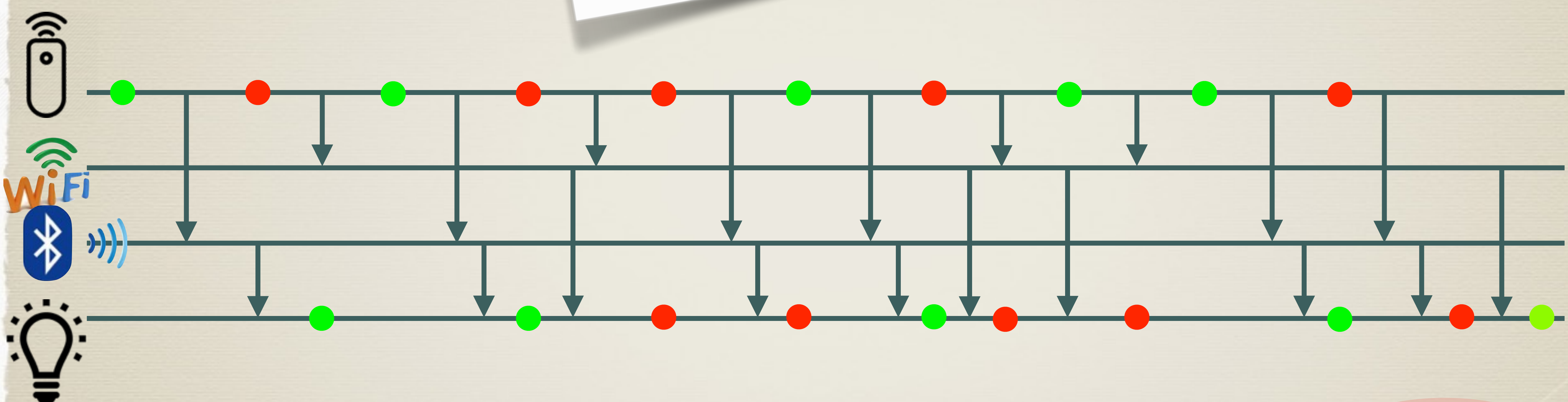
MSO: Monadic Second Order Logic

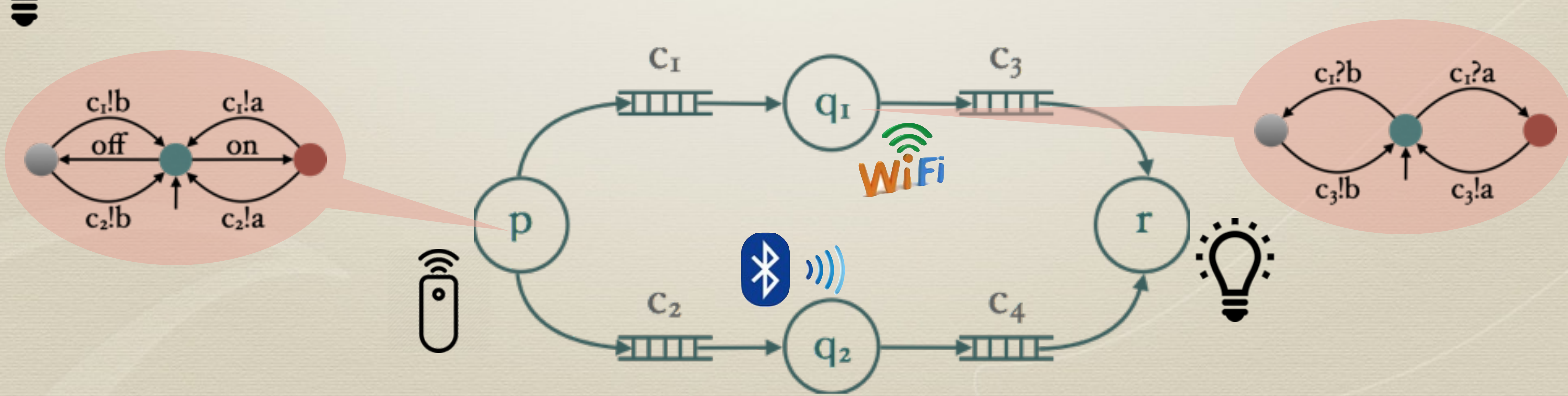
$$\varphi ::= \mathsf{false} \mid a(x) \mid p(x) \mid x \leq y \mid x \rhd^{d} y \mid x \rightarrow y$$
$$\mid x \in X \mid \varphi \vee \varphi \mid \neg\varphi \mid \exists x\, \varphi \mid \exists X\, \varphi$$



Behaviours as Graphs…

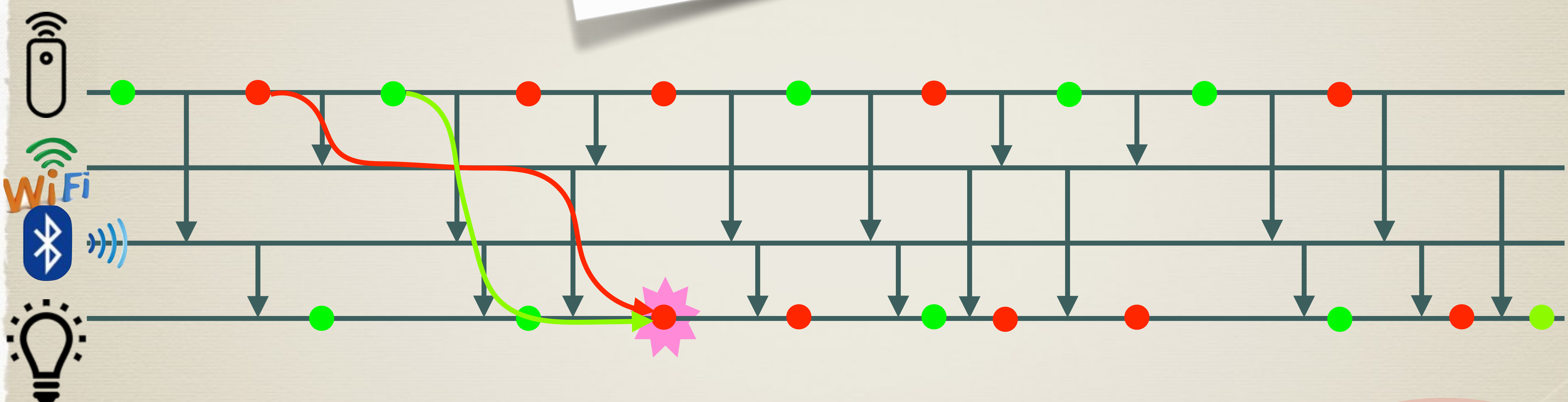
Does it obey the latest order?

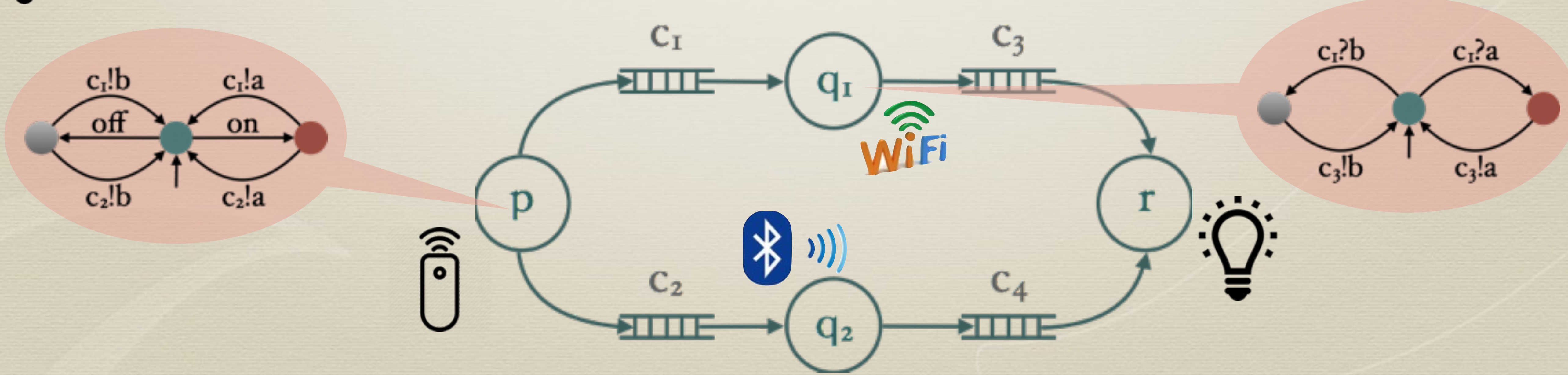




Behaviours as Graphs…

Does it obey the latest order?

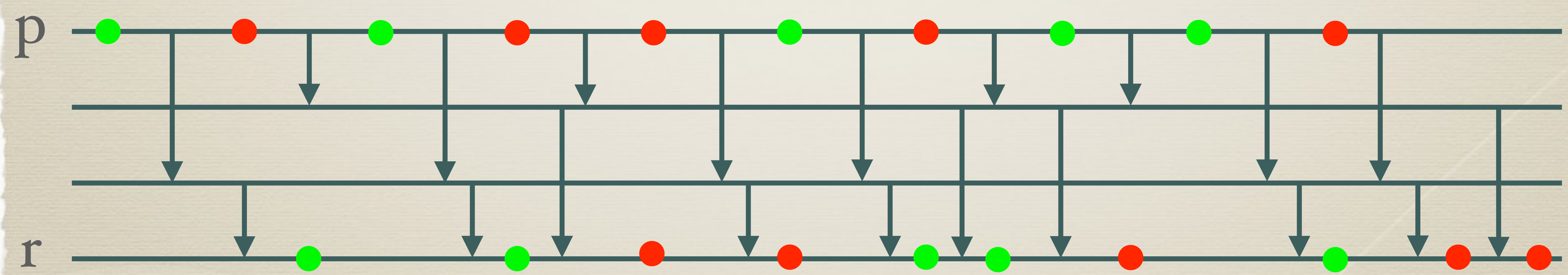




Obey the latest order

FO

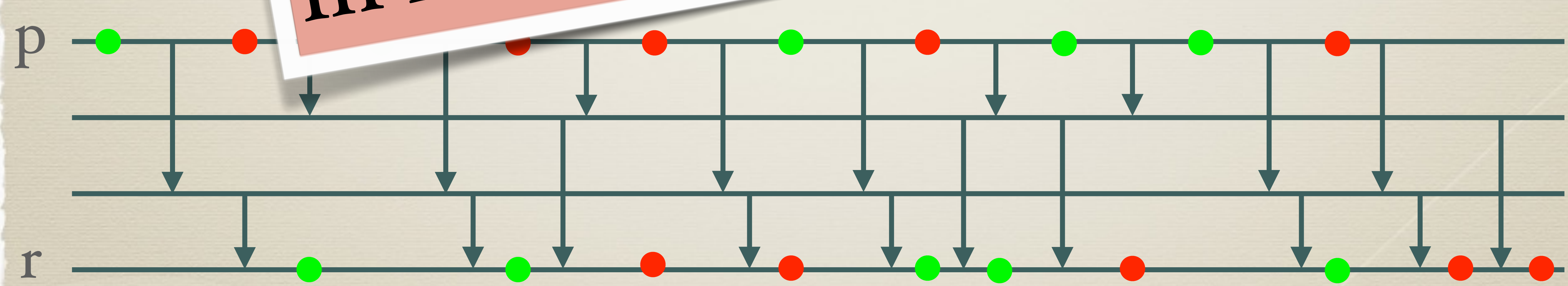
$$\forall z\,(r(z) \wedge \mathsf{on}(z)) \Rightarrow \exists y\,(p(y) \wedge y < z$$
$$\wedge\, \forall x\,(x < z \wedge p(x) \Rightarrow x \leq y)$$
$$\wedge\, \exists x\,(x \rightarrow y \wedge \mathsf{on}(x)))$$



Obey the latest order

FO

Obey the latest order
not expressible
in MSO over Linear Traces



Verification problems

- Emptiness or Reachability
- Satisfiability ϕ: Is there a CBM that satisfies ϕ?
- Model Checking: $S \vDash \phi$
  - Temporal logics
  - Propositional dynamic logics
  - Monadic second order logic

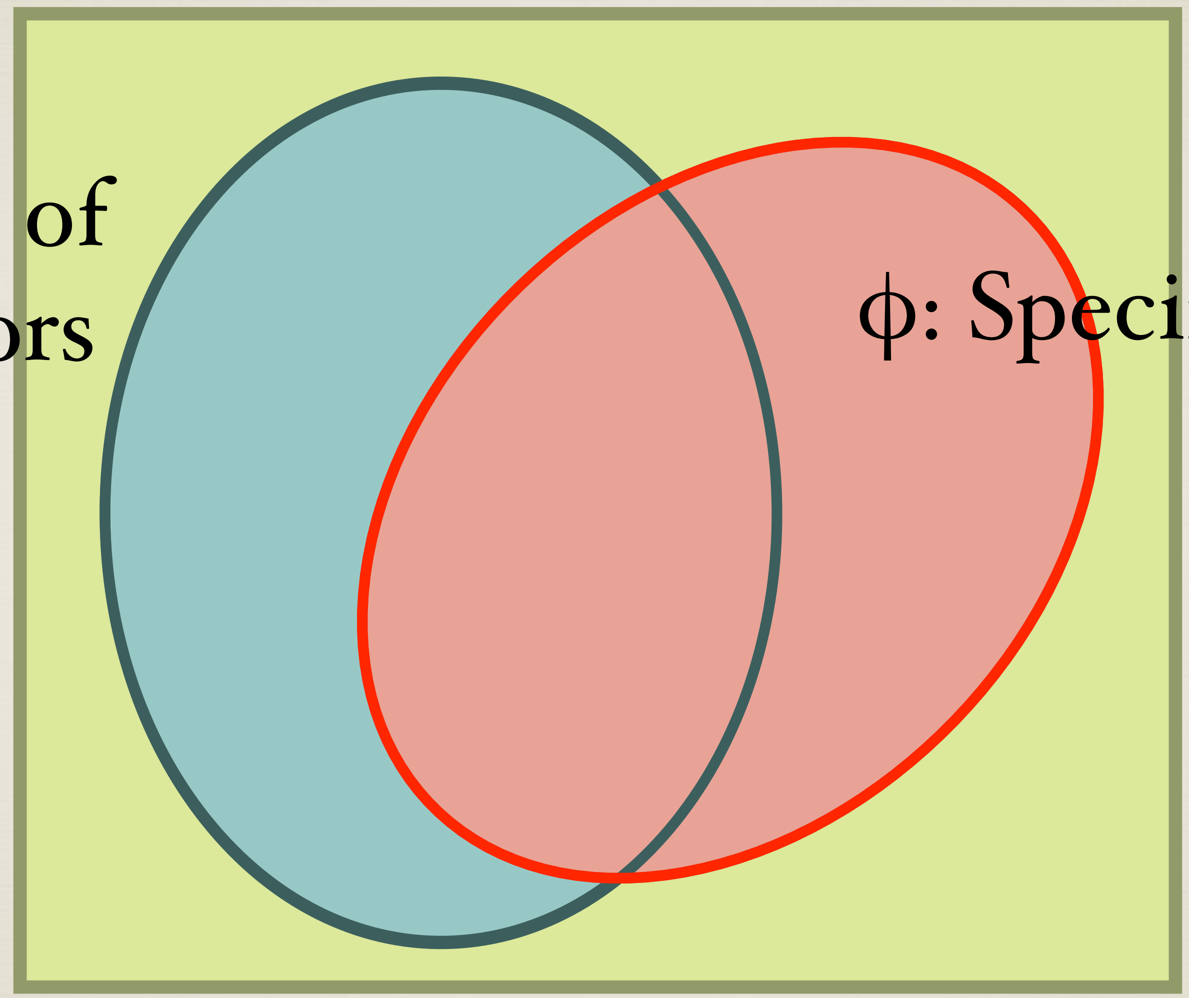
Verification problems

- Emptiness or Reachability
- Satisfiability φ: Is there a CBM that satisfies φ?
- Model Checking: $S \models \phi$
  - Temporal logics
  - Propositi
  - Mo

undecidable in general

Under-approximate Verification

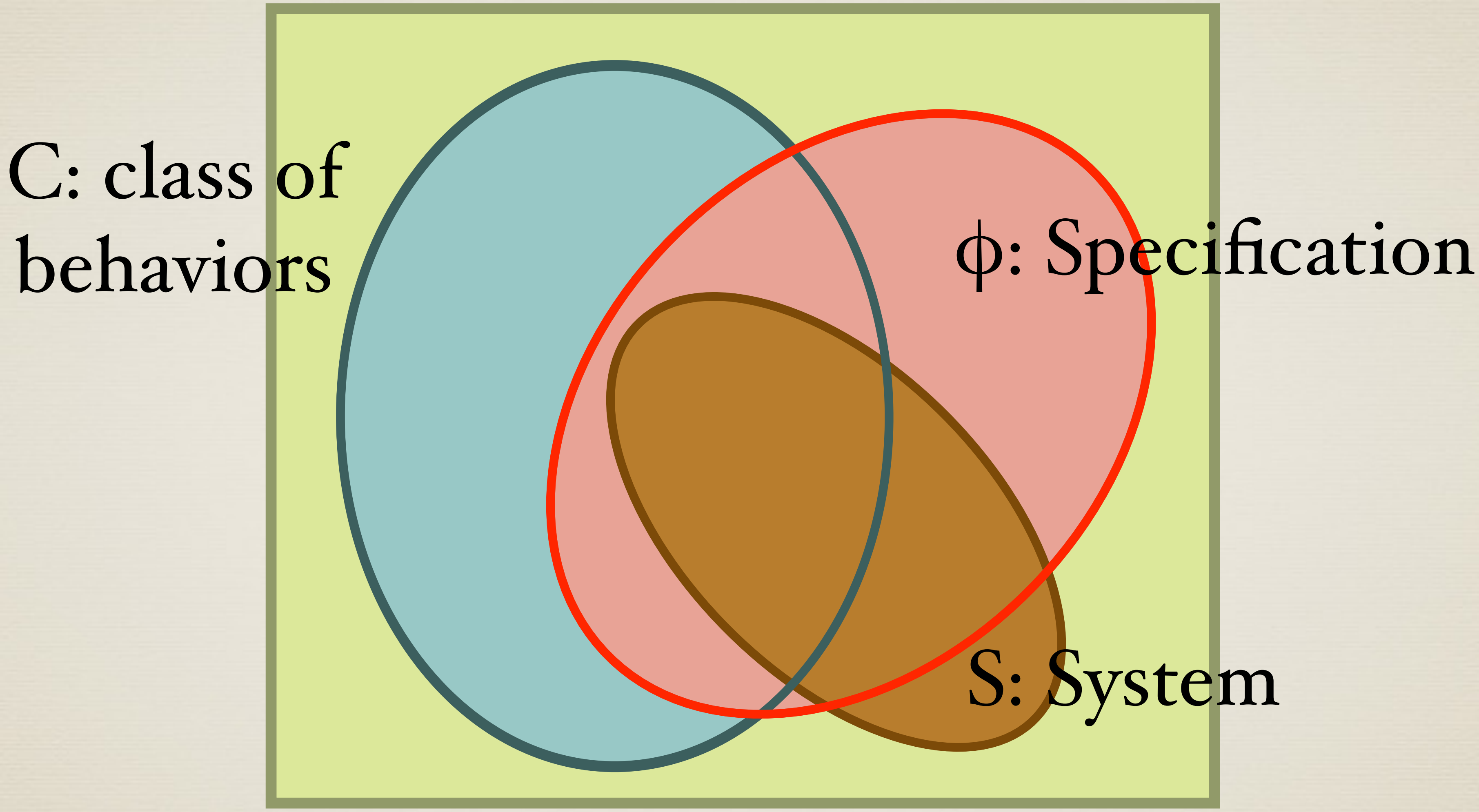
Satisfiability problem:



Is ϕ satisfiable in C?

Under-approximate Verification

Model checking problem: $S \vDash_C \phi$



Do all behaviors from C accepted by S satisfy ϕ?

Decidable Under-approximate Verification

* Bounded data structures
* Existentially bounded [Genest et al.]
* Acyclic Architectures [La Torre et al., Heußner 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]
* ...

Decidable Under-approximate Verification

* Bounded data structures
* Existentially bounded [Genest et al.]
* Acyclic Architectures [La Torre et al., Heußner 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]
* ...

Reduction to MSO/ Automata over trees.

Bounded-phase to Tree-width

LMP-lics07.pdf (page 10 of 10)

Search

Several future directions are interesting. First, the class of multiple nested word languages with a bounded number of phases is of bounded tree-width (this is the property that allows us to embed them in trees). It would be interesting to characterize naturally the exact class of multiple nested words that have bounded tree-width. Secondly, we believe that our results have applications to other areas in verification, for instance in checking parallel programs that communicate with each other using unbounded FIFO queues, as multiple stacks can be used to simulate queues.

Under-approximate Verification

The Tree Width of Auxiliary Storage

P. Madhusudan
University of Illinois at Urbana-Champaign, USA
madhu@illinois.edu

Gennaro Parlato
LIAFA, CNRS and University of Paris Diderot, France.
gennaro@liafa.jussieu.fr

ostract

e propose a generalization of results on the decidability of empti-
ss for several restricted classes of sequential and distributed au-
nata with auxiliary storage (stacks, queues) that have recently
en proved. Our generalization relies on reducing emptiness of
se automata to finite-state *graph automata* (without storage)
tricted to monadic second-order (MSO) definable graphs of
unded tree-width, where the graph structure encodes the mech-

However, the various identified decidable restrictions on the
automata are, for the most part, *awkward* in their definitions
e.g. emptiness of multi-stack pushdown automata where push
to any stack is allowed at any time, but popping is restricted
the first non-empty stack is decidable! [8]. Yet, relaxing the
definitions to more natural ones seems to either destroy decidabil
or their power. It is hence natural to ask: why do these automa
have decidable emptiness problems? Is there a common underlyi

Tree-width bounds for other Under-approximations

MadhuParlato.pdf (page 2 of 28)

- Multi-stack pushdown automata with bounded context-switching: This is the class of multi-stack automata where each computation of the automaton can be divided into k stages, where in each stage the automaton touches only one stack (proved decidable first in [14]). We show that they can be simulated by graph automata on graphs of tree-width $O(k)$.

- Multi-stack pushdown automata with bounded phases: These are automata that generalize the bounded-context-switching ones: the computations must be dividable into k phases, for a fixed k, where in each phase the automaton can push onto any stack, but can pop only from one stack (proved decidable recently in [11]). We show that graph automata on graphs of tree-width $O(2^k)$ can simulate them.

- Ordered multi-stack pushdown automata: The restriction here is that there a finite number of stacks that are ordered, and at any time, the automaton can push onto any stack, but pop only from the first non-empty stack. Note that the computation is not cut into phases, as in the above two restrictions. We show that automata on graphs of tree-width $O(n \cdot 2^n)$ (where n is the number of stacks) can simulate them.

- Distributed queue automata on polyforest architectures: Distributed queue automata is a model where finite-state processes at n sites work by communicating to each other using FIFO channels, modeled as queues. It was shown recently, that when the architecture is a polyforest (i.e. the underlying undirected graph is a forest), the emptiness problem is decidable (and for other architectures, it is undecidable) [12]. We

Why Tree-width?

Corollary to Seese's Theorem:
If C is any MSO definable family of graphs then, for any k, checking MSO satisfiability among graphs in C with tree-width at most k is decidable.

Why Tree-width?

Corollary to Seese's Theorem:
If C is any MSO definable family of graphs then, for any k, checking MSO satisfiability among graphs in C with tree-width at most k is decidable.

Interpretation over trees.

ENCYCLOPEDIA OF MATHEMATICS AND ITS APPLICATIONS

Graph Structure and Monadic Second-Order Logic

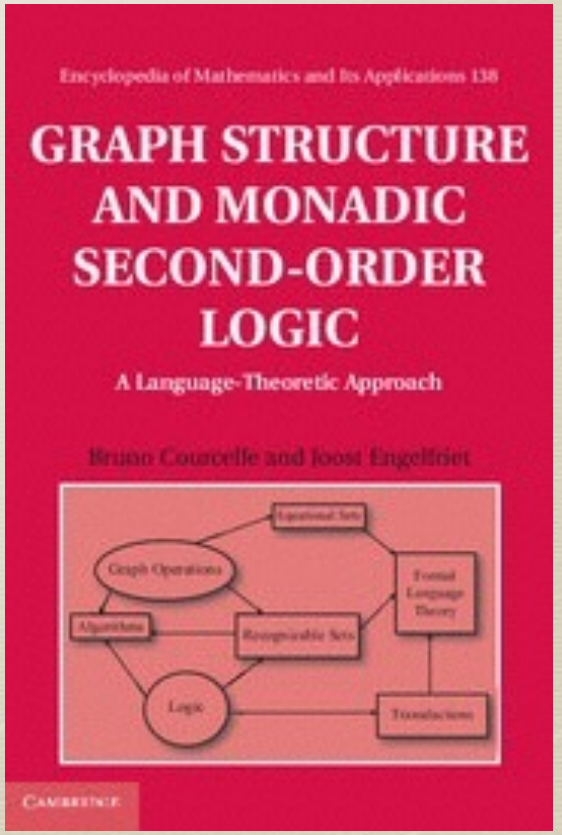
A Language-Theoretic Approach

BRUNO COURCELLE

Université de Bordeaux

JOOST ENGELFRIET

Universiteit Leiden



Co-graphs: An example

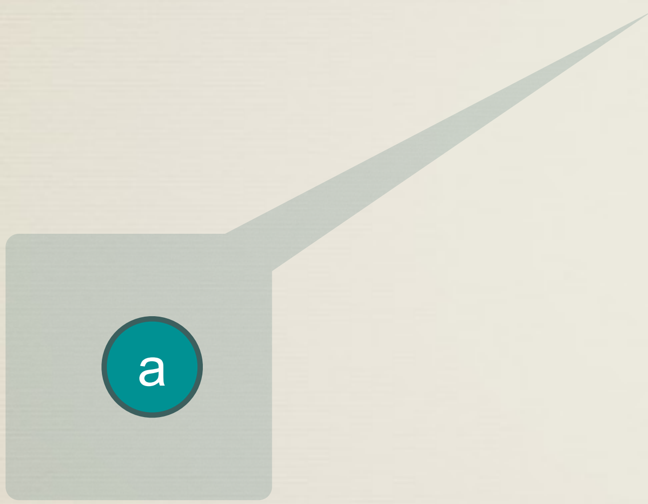
* Family of graphs generated by the following algebra:

$$G ::= \quad a \in \Sigma \mid G \oplus G \mid G \otimes G$$

Co-graphs: An example

- Family of graphs generated by the following algebra:

$$G ::= \ a \in \Sigma \mid G \oplus G \mid G \otimes G$$

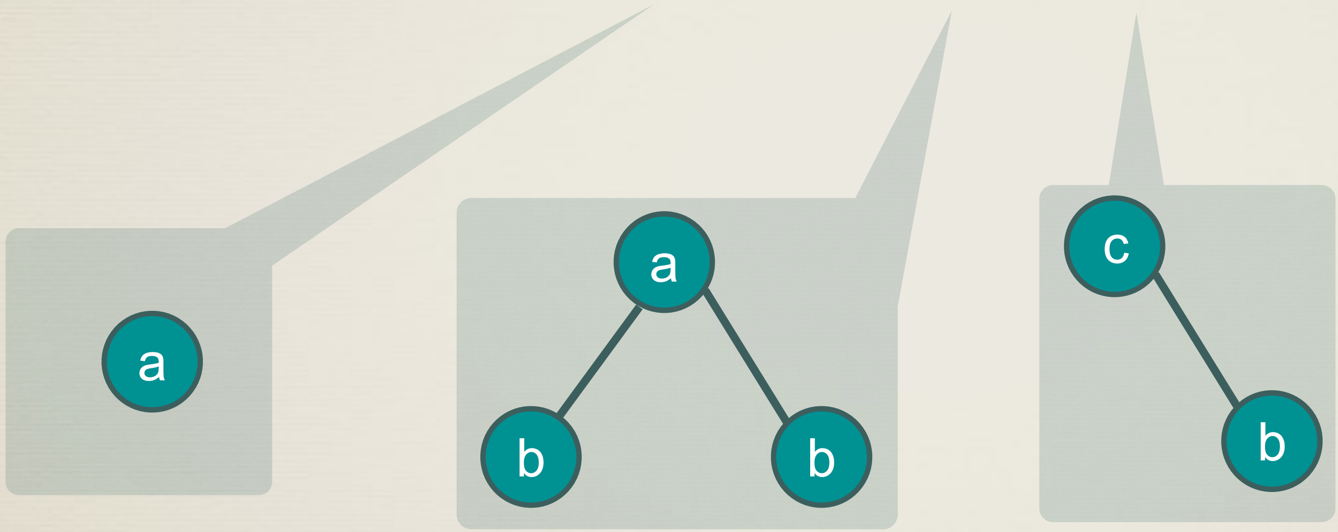


Single vertex
labelled a

Co-graphs: An example

* Family of graphs generated by the following algebra:

$$G ::= \; a \in \Sigma \mid G \oplus G \mid G \otimes G$$



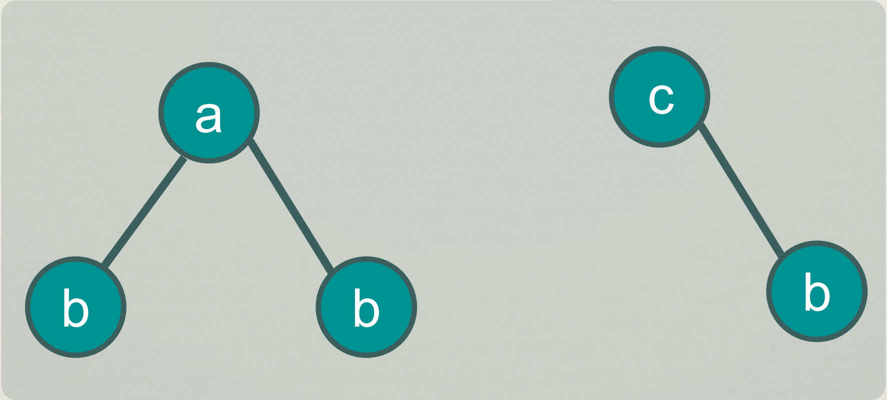
Single vertex labelled a

Co-graphs: An example

* Family of graphs generated by the following algebra:

$$G ::= \ a \in \Sigma \ | \ G \oplus G \ | \ G \otimes G$$

Single vertex labelled a

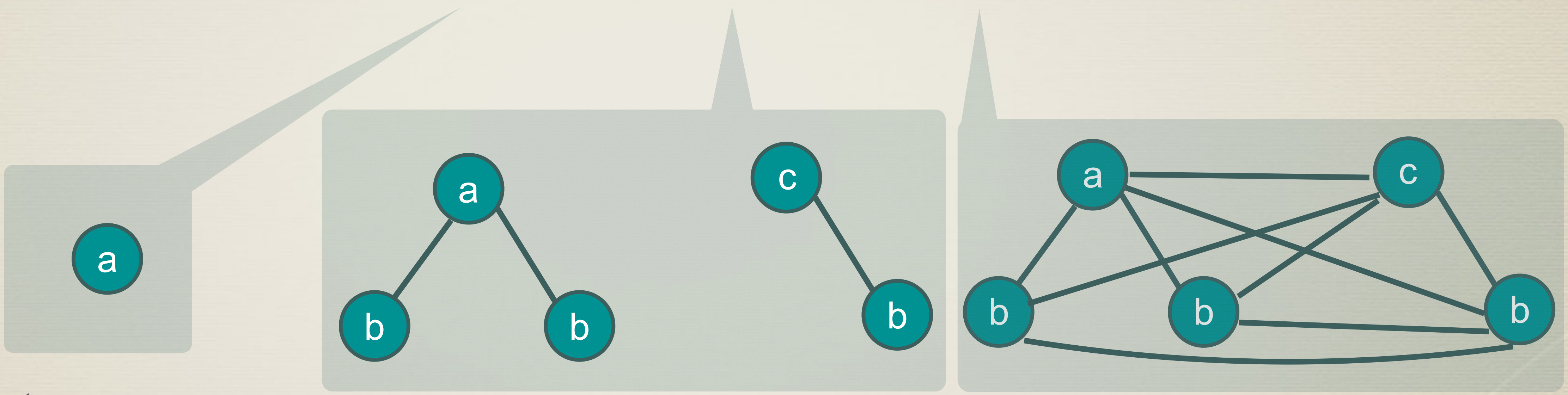


Disjoint union

Co-graphs: An example

* Family of graphs generated by the following algebra:

$$G ::= \; a \in \Sigma \mid G \oplus G \mid G \otimes G$$



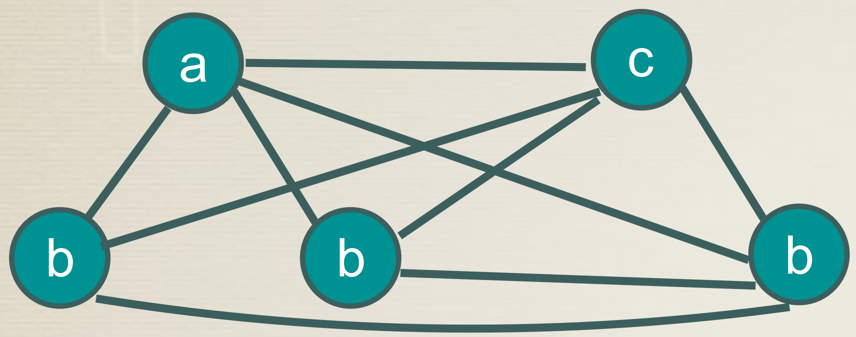
Single vertex labelled a

Disjoint union

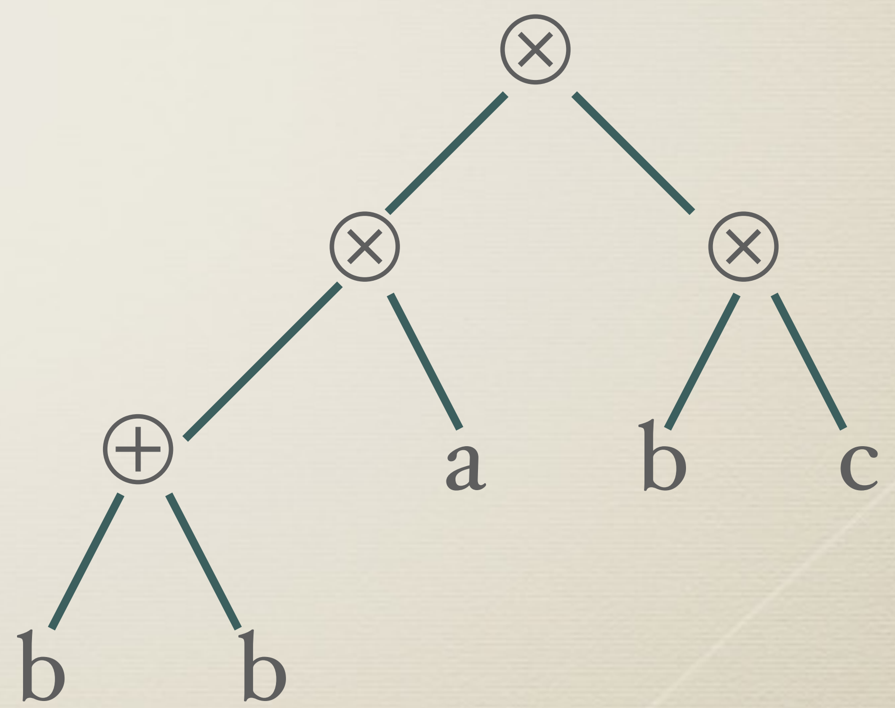
Disjoint union and connect all pairs

Co-graphs to Trees

Every co-graph has an expression generating it.

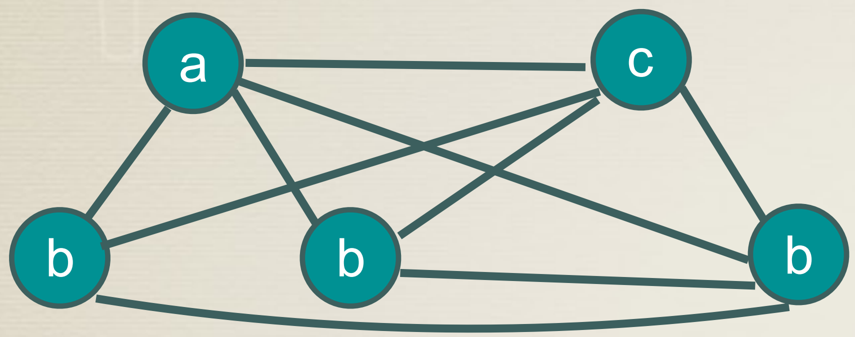


$((b \oplus b) \otimes a) \otimes (b \otimes c)$

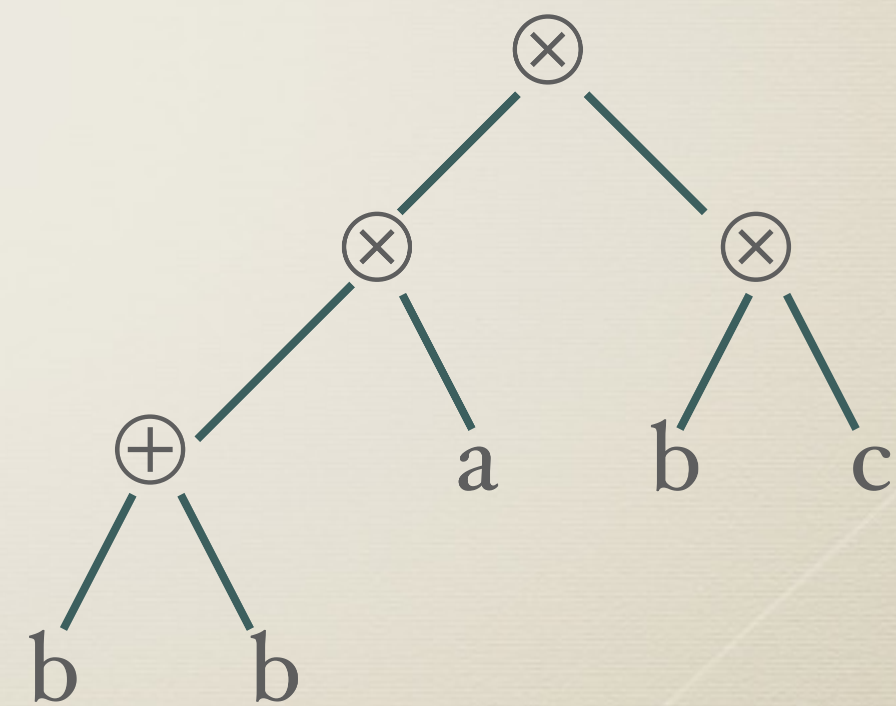


Co-graphs to Trees

Every co-graph has an expression generating it.



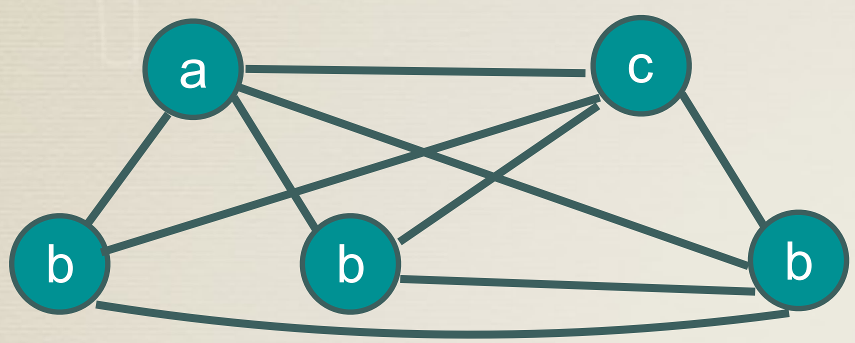
$((b \oplus b) \otimes a) \otimes (b \otimes c)$



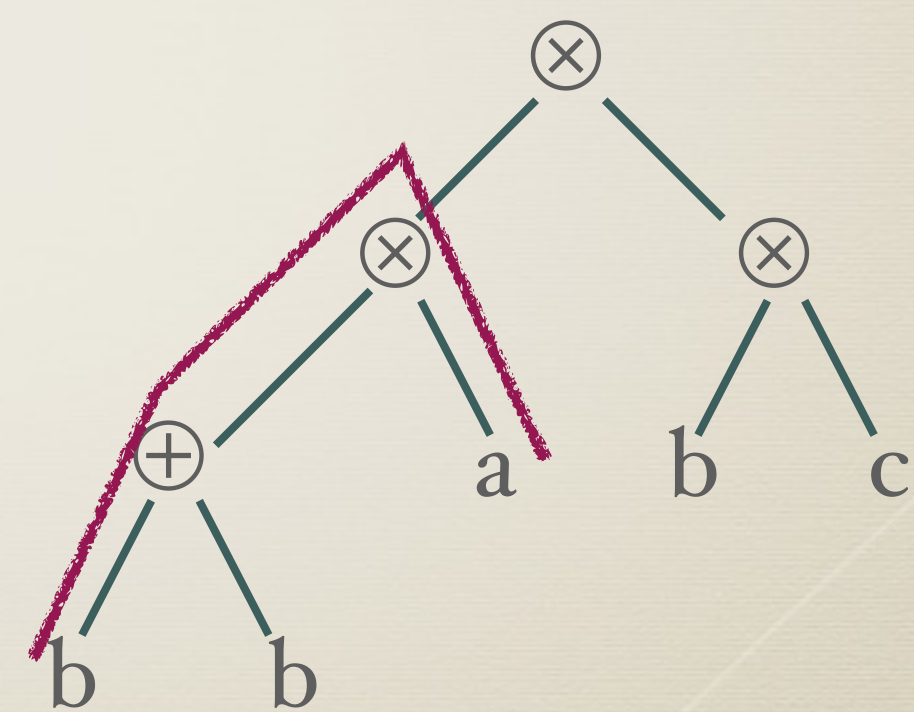
Vertices of the graph correspond to leaves of the expression tree.

Co-graphs to Trees

Every co-graph has an expression generating it.



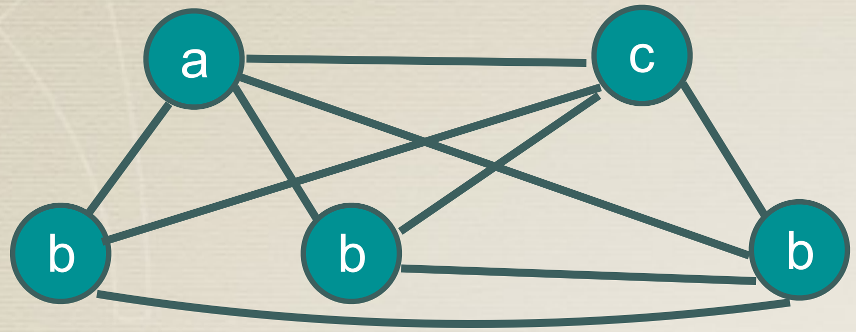
$((b \oplus b) \otimes a) \otimes (b \otimes c)$



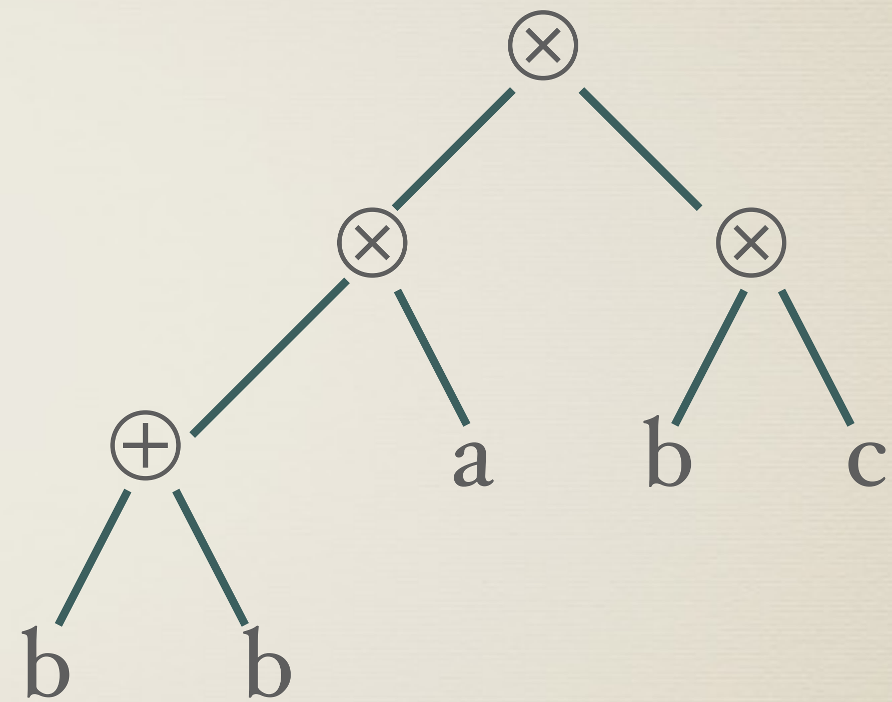
Vertices of the graph correspond to leaves of the expression tree.

Edges are introduced by $\otimes$ nodes between leaves in its two subtrees

Interpretation on Trees

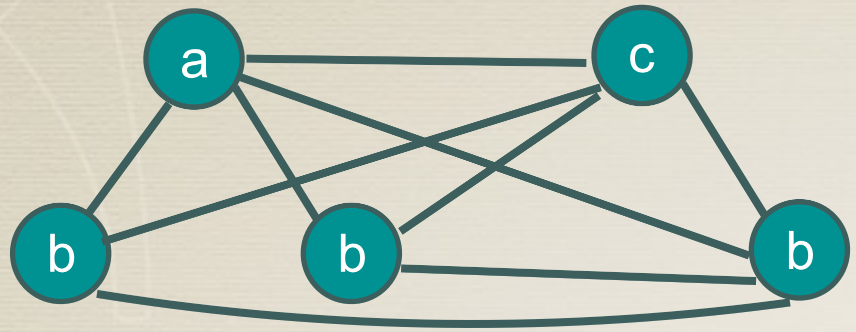


$$((b \oplus b) \otimes a) \otimes (b \otimes c)$$

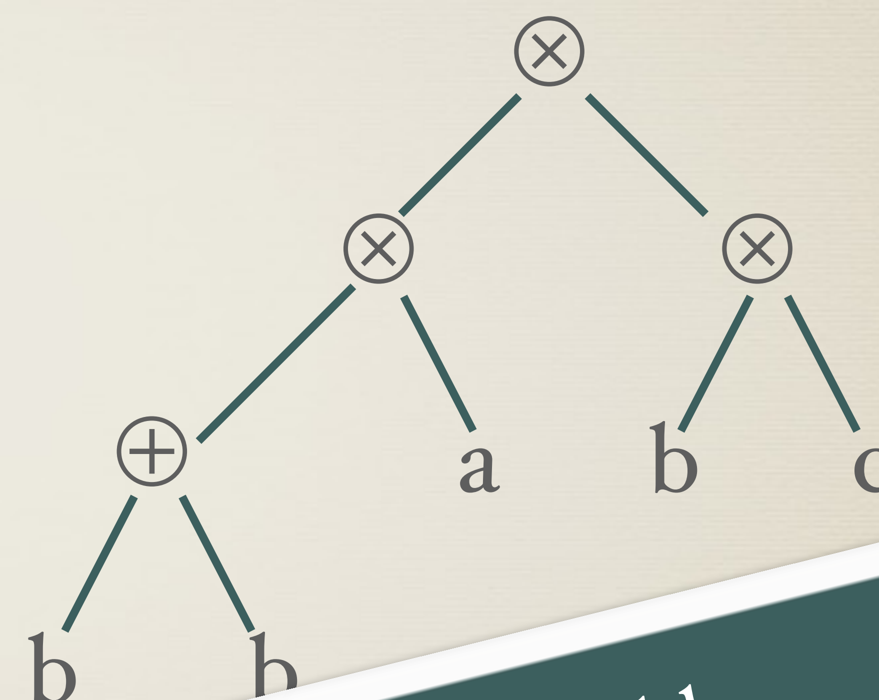


Graph	Tree
There is a vertex x	There is a leaf x
There is a set of vertices X	There is a set of leaves X
a(x)	a(x)
E(x,y)	There is path from x to y whose highest node is a $\otimes$

Interpretation on Trees



$((b \oplus b) \otimes a) \otimes (b \otimes c)$



Graph	Tree
There is a vertex x	There is a leaf x
There is a set of vertices X	There is a set of leaves X
a(x)	a(x)
E(x,y)	There is path f wh

MSO Theory of Co-graphs is decidable

Advantages of the algebraic approach

Advantages of the algebraic approach

* Membership checking
  * Are all paths co-graphs?

Advantages of the algebraic approach

* Membership checking
  * Are all paths co-graphs?

Advantages of the algebraic approach

* Membership checking
  * Are all paths co-graphs?

* What labels the root?

$\otimes$ or $\oplus$

Advantages of the algebraic approach

* Membership checking
  * Are all paths co-graphs?

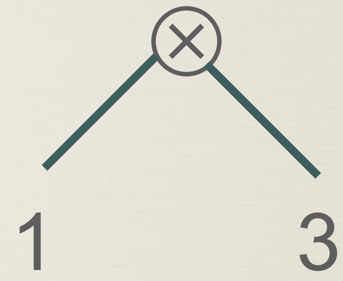
* What labels the root?

$\otimes$

Advantages of the algebraic approach

* Membership checking
  * Are all paths co-graphs?

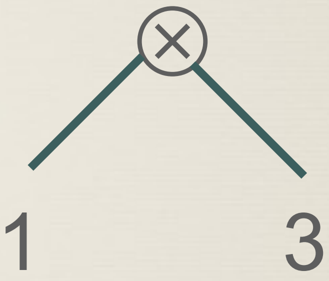
* What labels the root?



Advantages of the algebraic approach

* Membership checking
    * Are all paths co-graphs?

* What labels the root?

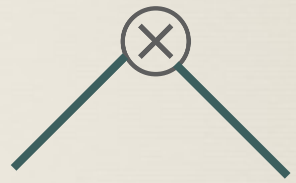


No vertex with degree 3!

Advantages of the algebraic approach

* Membership checking
    * Are all paths co-graphs?

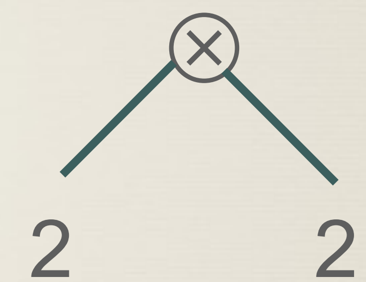
* What labels the root?



Advantages of the algebraic approach

* Membership checking
    * Are all paths co-graphs?

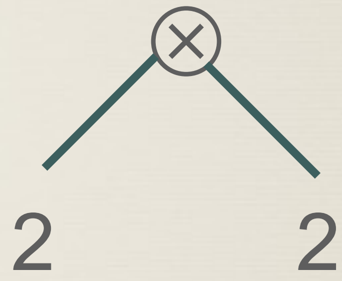
* What labels the root?



Advantages of the algebraic approach

* Membership checking
  * Are all paths co-graphs?

* What labels the root?

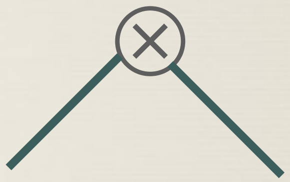


No 2 x 2 perfect matching!

Advantages of the algebraic approach

* Membership checking
  * Are all paths co-graphs?

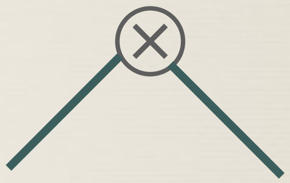
* What labels the root?



Advantages of the algebraic approach

* Membership checking
  * Are all paths co-graphs?

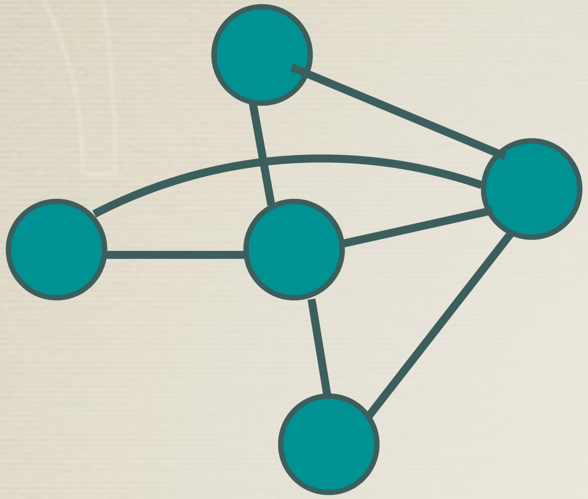
* What labels the root?

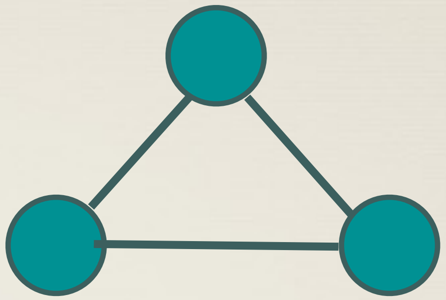


The path of length 3 is not a co-graph

Using the Co-graph Algebra

Let C = { G | {u,v} is an edge then N(u) ⊆N(v) or N(v)⊆N(u) }





The class is clearly MSO definable.

Using the Co-graph Algebra

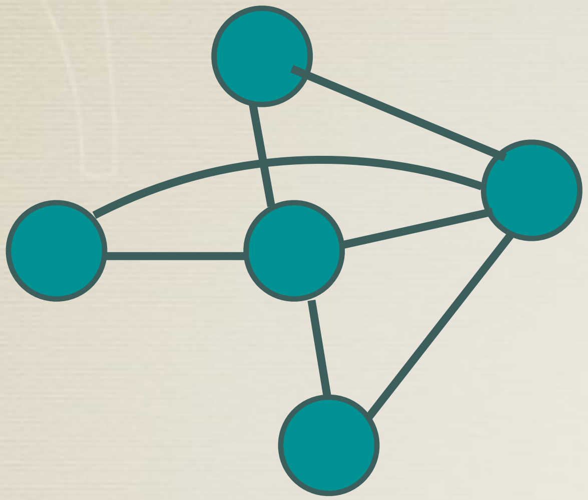
Let C = { G | {u,v} is an edge then N(u) ⊆N(v) or N(v)⊆N(u) }

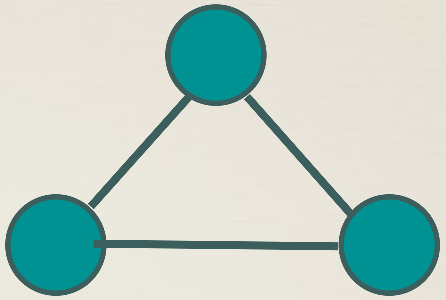
The class is clearly MSO definable.

Is the MSO theory of C decidable?

Using the Co-graph Algebra

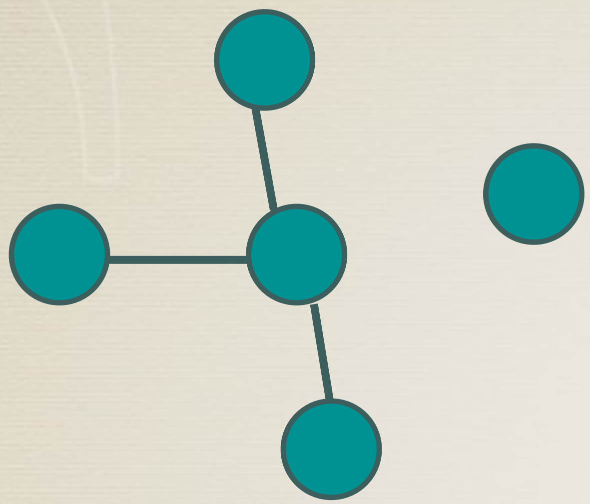
Let C = { G | {u,v} is an edge then N(u) $\subseteq$N(v) or N(v)$\subseteq$N(u) }

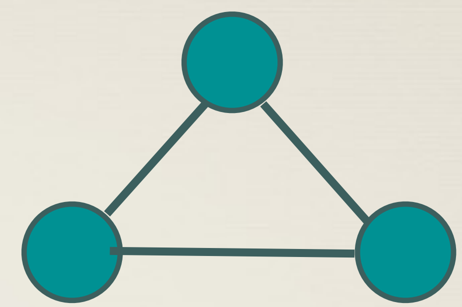




Using the Co-graph Algebra

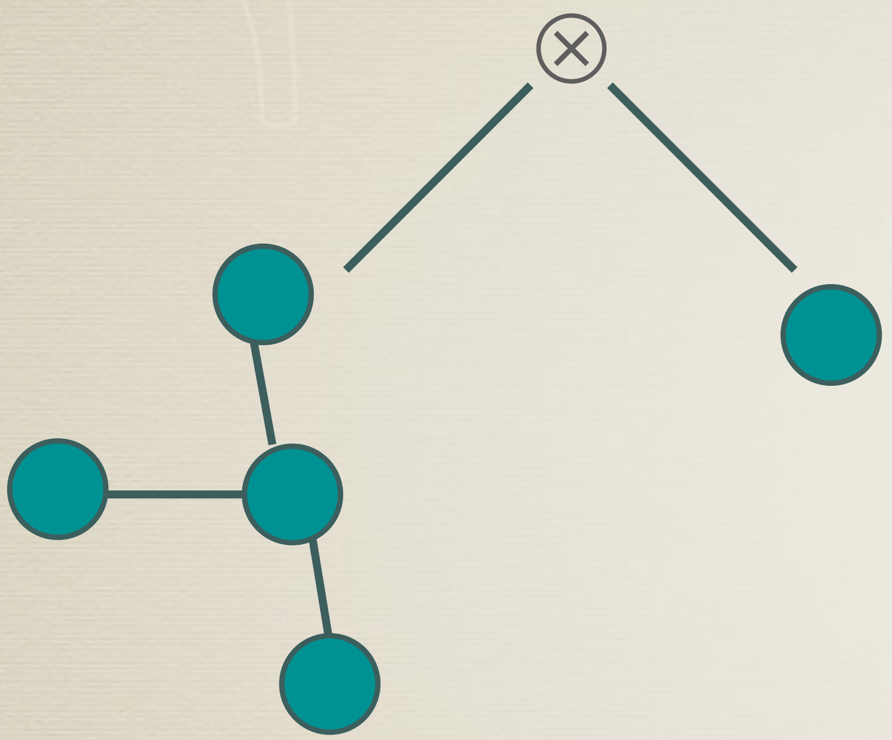
Let C = { G | {u,v} is an edge then $N(u) \subseteq N(v)$ or $N(v) \subseteq N(u)$ }

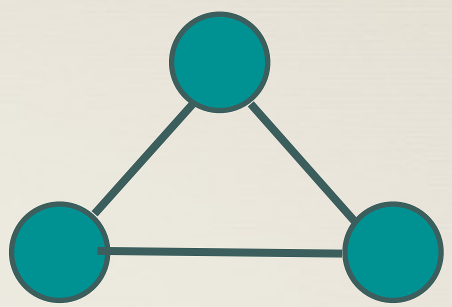




Using the Co-graph Algebra

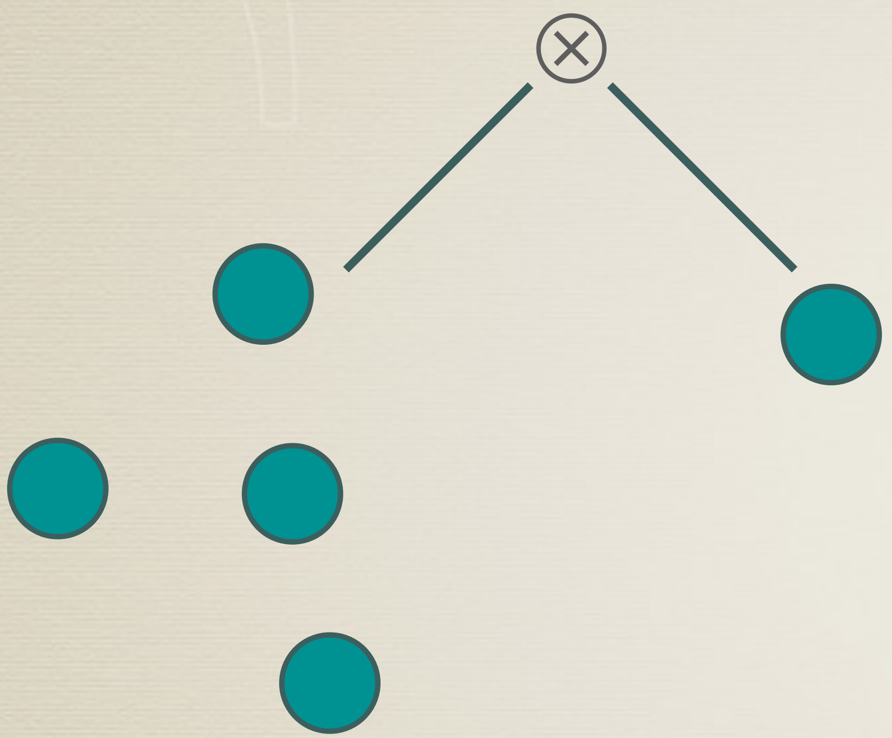
Let C = { G | {u,v} is an edge then N(u) ⊆N(v) or N(v)⊆N(u) }

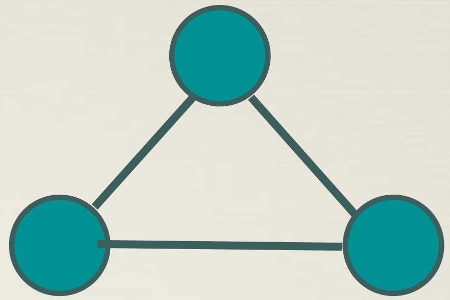




Using the Co-graph Algebra

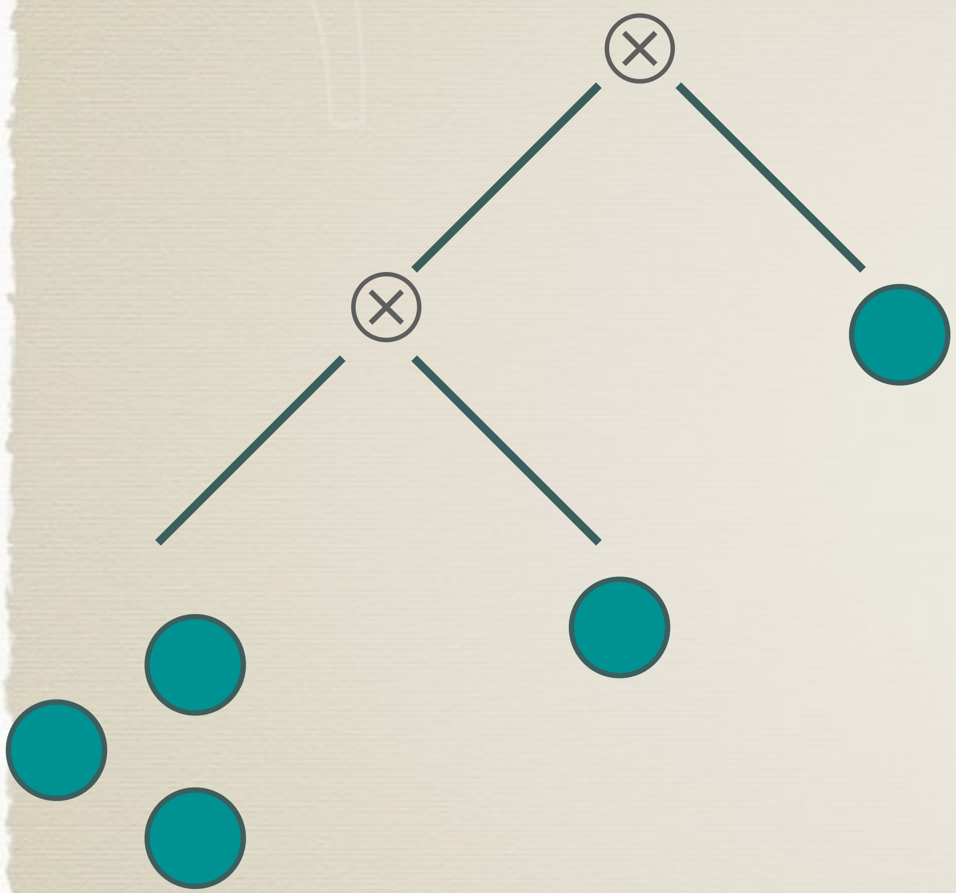
Let C = { G | {u,v} is an edge then N(u) ⊆N(v) or N(v)⊆N(u) }

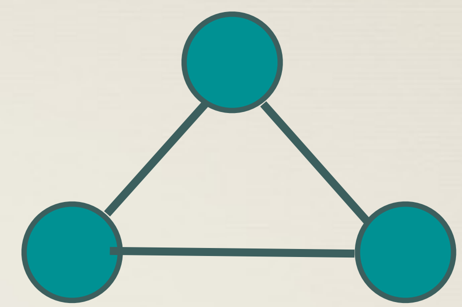




Using the Co-graph Algebra

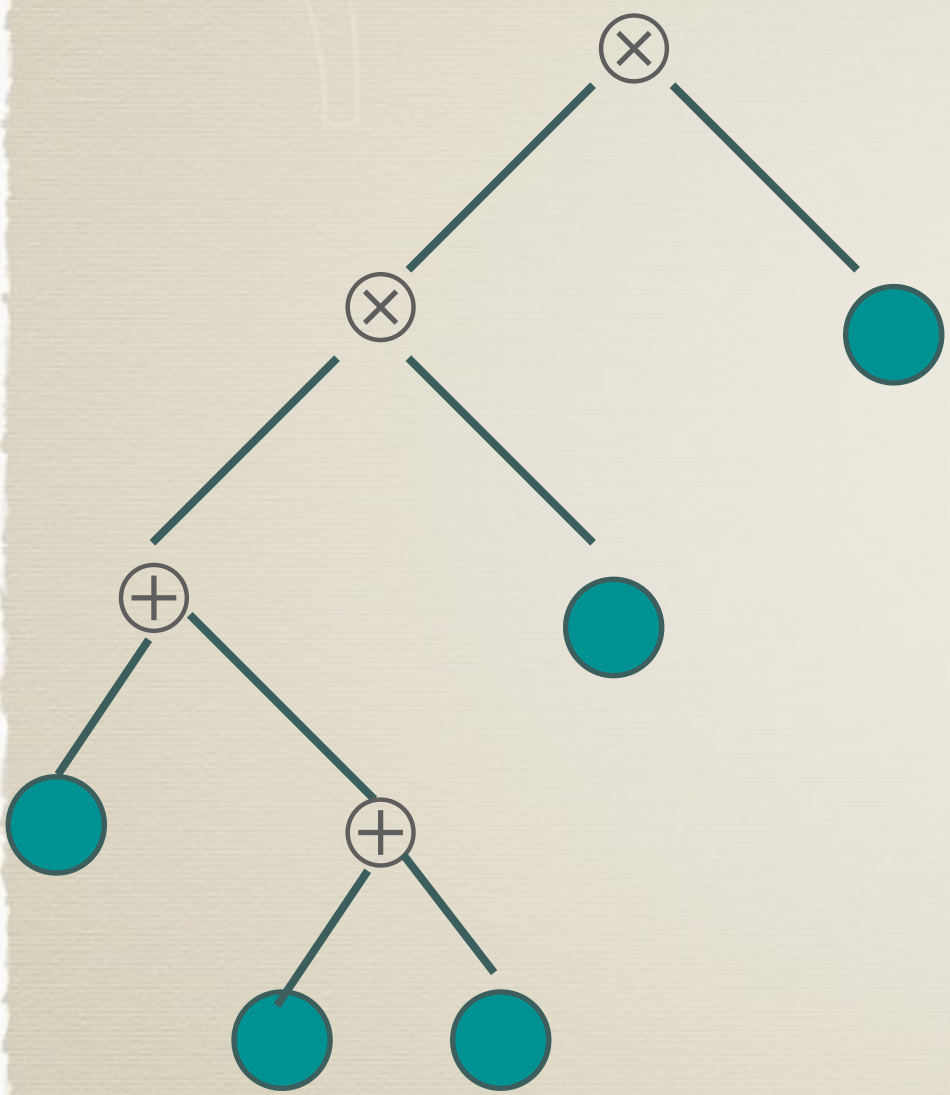
Let C = { G | {u,v} is an edge then N(u) ⊆N(v) or N(v)⊆N(u) }

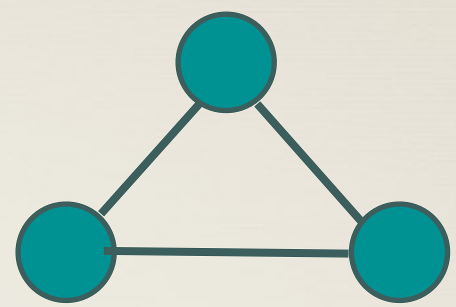




Using the Co-graph Algebra

Let C = { G | {u,v} is an edge then $N(u) \subseteq N(v)$ or $N(v) \subseteq N(u)$ }





Using the Co-graph Algebra

Let C = { G | {u,v} is an edge then $N(u) \subseteq N(v)$ or $N(v) \subseteq N(u)$ }

Using the Co-graph Algebra

Let C = { G | {u,v} is an edge then $N(u) \subseteq N(v)$ or $N(v) \subseteq N(u)$ }

Size one graphs in C are co-graphs.

Using the Co-graph Algebra

Let $C = \{ G \mid \{u,v\}$ is an edge then $N(u) \subseteq N(v)$ or $N(v) \subseteq N(u) \}$

- Size one graphs in C are co-graphs.
- Case 1: If the graph is not connected then we inductively construct expressions for each part.

Using the Co-graph Algebra

Let C = { G | {u,v} is an edge then N(u) ⊆N(v) or N(v)⊆N(u) }

Size one graphs in C are co-graphs.

Case 1: If the graph is not connected then we inductively construct expressions for each part.

Case 2: If the graph is connected. ⊗ at the top.

Using the Co-graph Algebra

Let $C = \{ G \mid \{u,v\}$ is an edge then $N(u) \subseteq N(v)$ or $N(v) \subseteq N(u) \}$

Size one graphs in C are co-graphs.

Case 1: If the graph is not connected then we inductively construct expressions for each part.

Case 2: If the graph is connected. $\otimes$ at the top.

Divide it into two parts so that the complete bipartite graph on the division is a subgraph

Using the Co-graph Algebra

Let C = { G | {u,v} is an edge then $N(u) \subseteq N(v)$ or $N(v) \subseteq N(u)$ }

Case 2: If the graph is connected. $\otimes$ at the top.

Using the Co-graph Algebra

Let C = { G | {u,v} is an edge then N(u) ⊆N(v) or N(v)⊆N(u) }

Case 2: If the graph is connected. ⊗ at the top.

u be a maximal degree vertex.

Using the Co-graph Algebra

Let C = { G | {u,v} is an edge then N(u) ⊆N(v) or N(v)⊆N(u) }

Case 2: If the graph is connected. ⊗ at the top.

u be a maximal degree vertex.

v be the closest vertex that is not a neighbour

Using the Co-graph Algebra

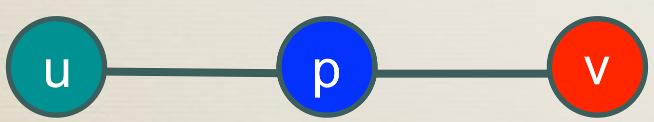
Let C = { G | {u,v} is an edge then N(u) ⊆N(v) or N(v)⊆N(u) }

Case 2: If the graph is connected. ⊗ at the top.

u be a maximal degree vertex.

v be the closest vertex that is not a neighbour

Then



Using the Co-graph Algebra

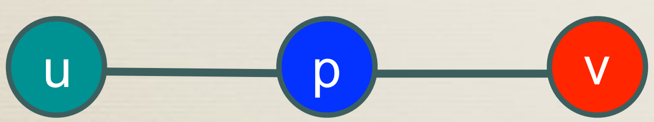
Let C = { G | {u,v} is an edge then $N(u) \subseteq N(v)$ or $N(v) \subseteq N(u)$ }

Case 2: If the graph is connected. $\otimes$ at the top.

u be a maximal degree vertex.

v be the closest vertex that is not a neighbour

Then



$N(u) \supseteq N(p)$ and so v in N(p).

Using the Co-graph Algebra

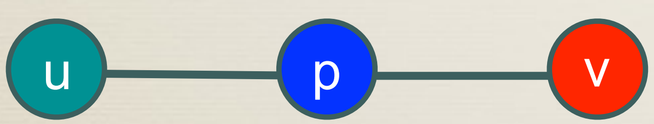
Let C = { G | {u,v} is an edge then $N(u) \subseteq N(v)$ or $N(v) \subseteq N(u)$ }

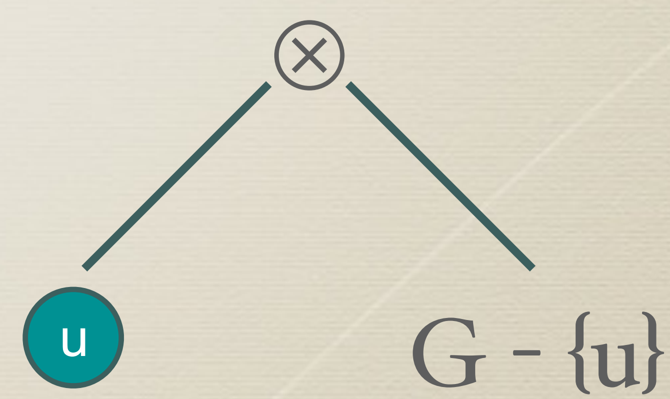
Case 2: If the graph is connected. $\otimes$ at the top.

u be a maximal degree vertex.

v be the closest vertex that is not a neighbour

Then





$N(u) \supseteq N(p)$ and so v in $N(p)$.

u is a neighbour of every vertex in G

Using the Co-graph Algebra

Let C = { G | {u,v} is an edge then N(u) ⊆N(v) or N(v)⊆N(u) }

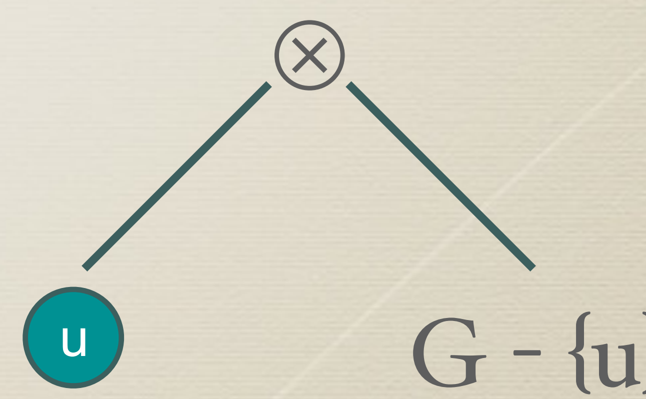
Case 2: If the graph is connected. ⊗ at the top.

u be a maximal degree vert…

v be…

Every graph in C is a co-graph. MSO theory of C is decidable

Then



N(u) ⊇ N(p) and so v in N(p).

u is a neighbour of every vertex in G

Using the Co-graph Algebra

Let C = { G | {u,v} is an edge then $N(u) \subseteq N(v)$ or $N(v) \subseteq N(u)$ }

Case 2: If the graph is connected. $\otimes$ at the top.

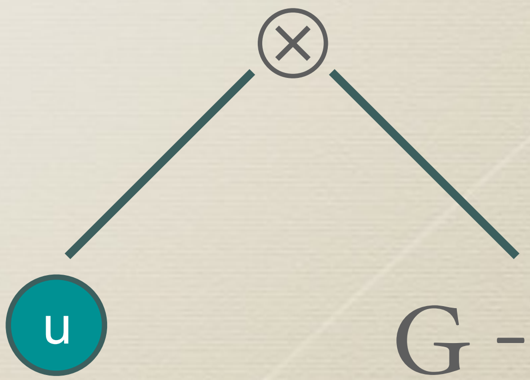
u be a maximal degree vert…

v be…

MSO theory of quasi-threshold graphs is decidable.

Then

$N(u) \supseteq N(p)$ and so v in N(p).



u is a neighbour of every vertex in G

Advantages of the algebraic approach

* The tree interpretation is quite transparent.
* Helps in establishing membership/containment in class.

Advantages of the algebraic approach

* The tree interpretation is quite transparent.
* Helps in establishing membership/containment in class.
* Fewer operators might help.
    * For quasi-threshold graphs, our search was guided by the limited set of operations available.

Split-width

* A way to decompose graphs to obtain a tree interpretation.
* Specifically designed for CBMs
* CBMs have bounded degree

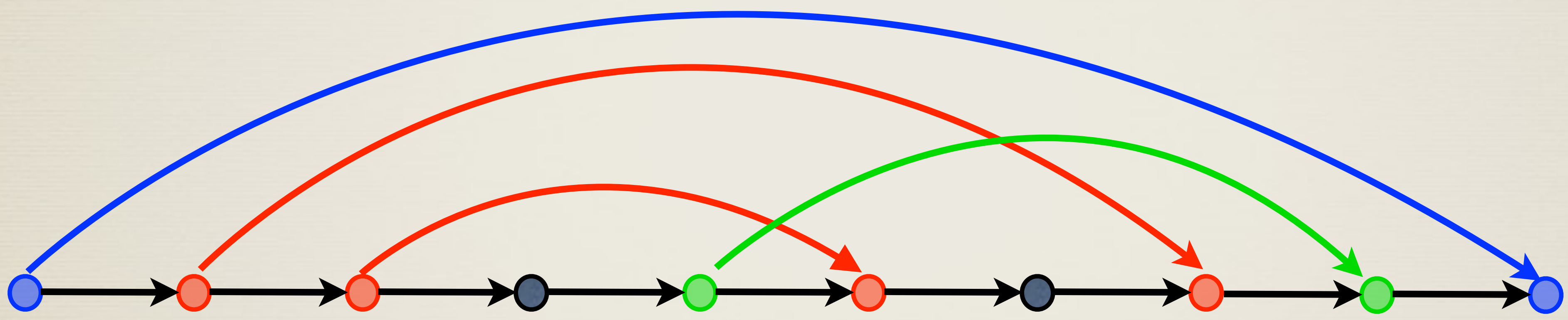
Let C be MSO definable class of CBMs
TFAE

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

Split-width Operations

The Cut Operation:

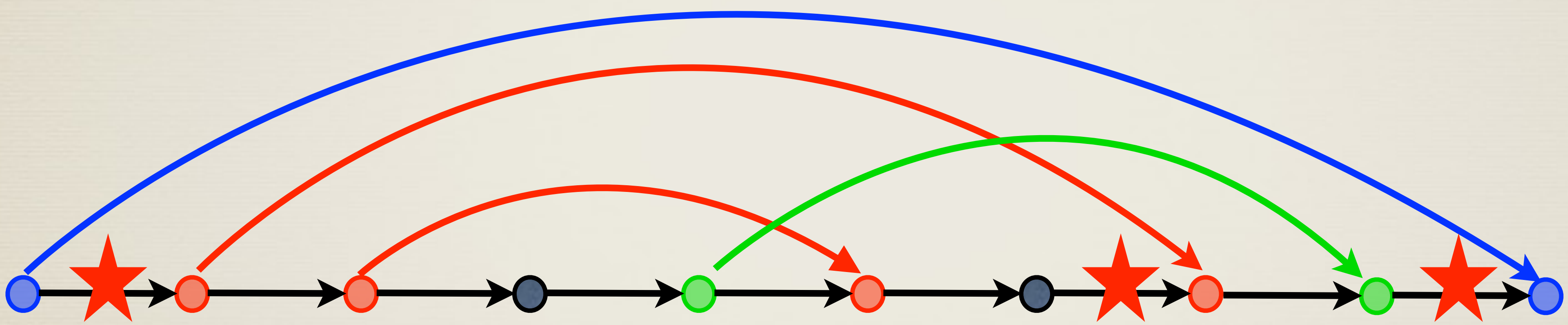
Pick a set of process edges and delete them.



Split-width Operations

The Cut Operation:

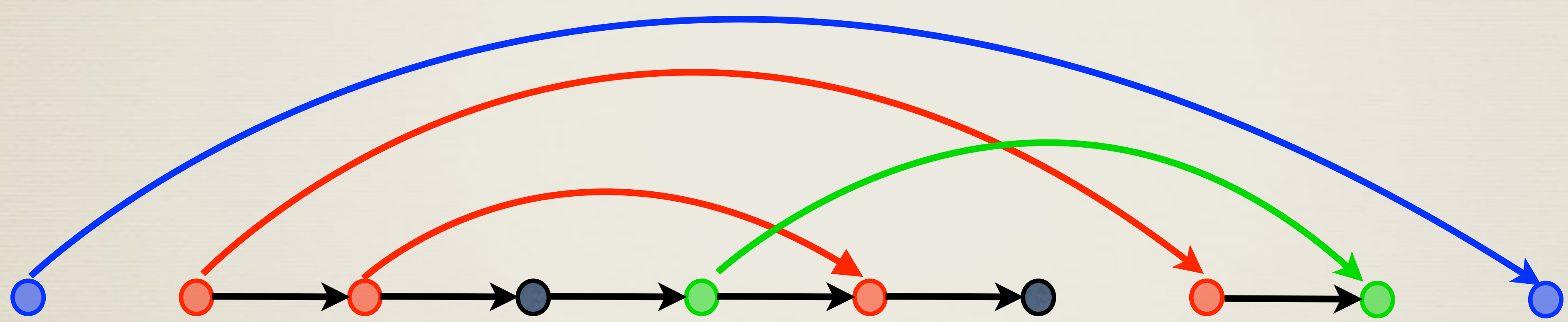
Pick a set of process edges and delete them.



Split-width Operations

The Cut Operation:

Pick a set of process edges and delete them.

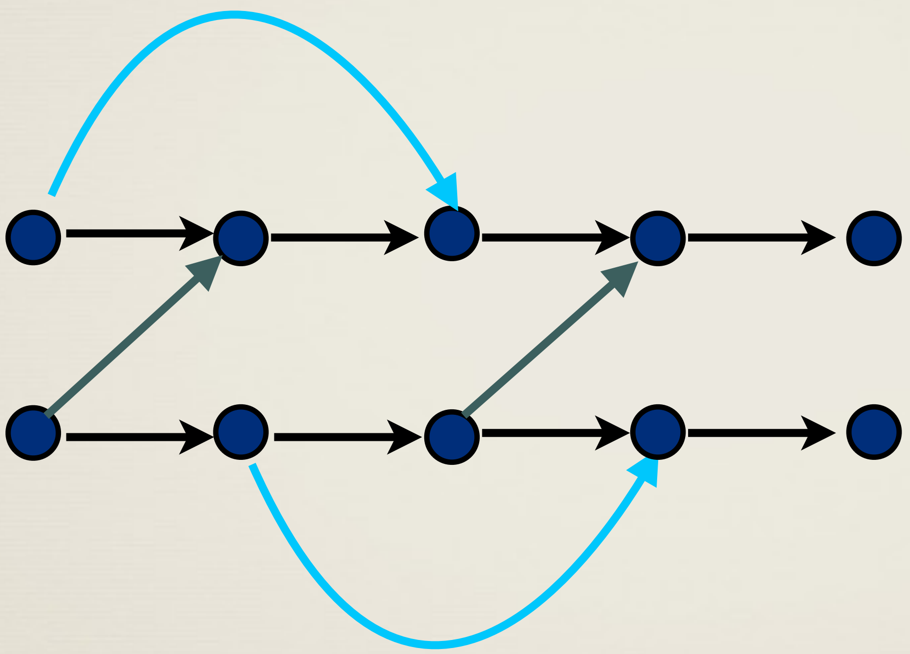


Split CBM with 3 holes

Split-width Operations

The Cut Operation:

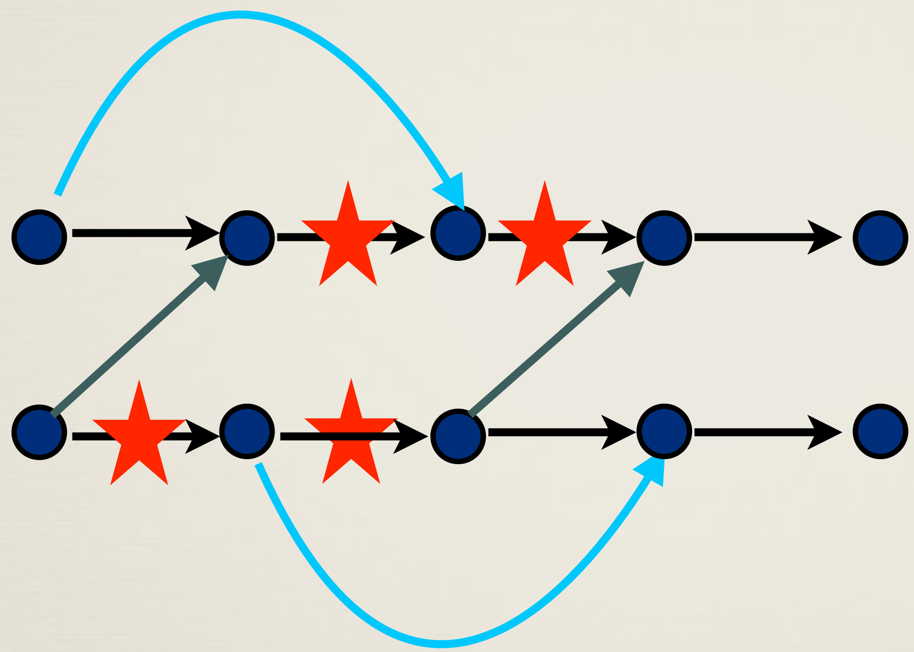
Pick a set of process edges and delete them.



Split-width Operations

The Cut Operation:

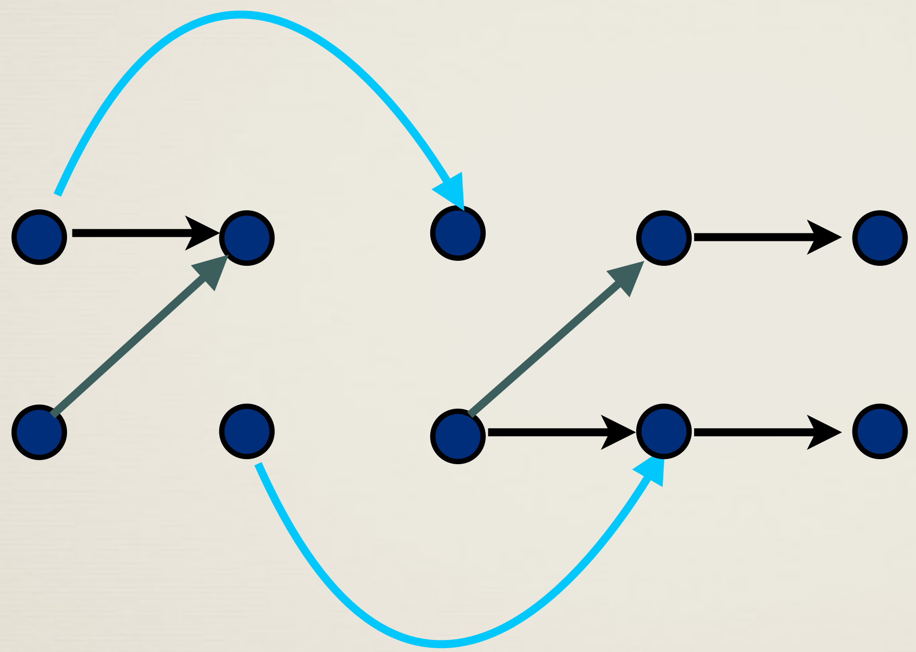
Pick a set of process edges and delete them.



Split-width Operations

The Cut Operation:

Pick a set of process edges and delete them.

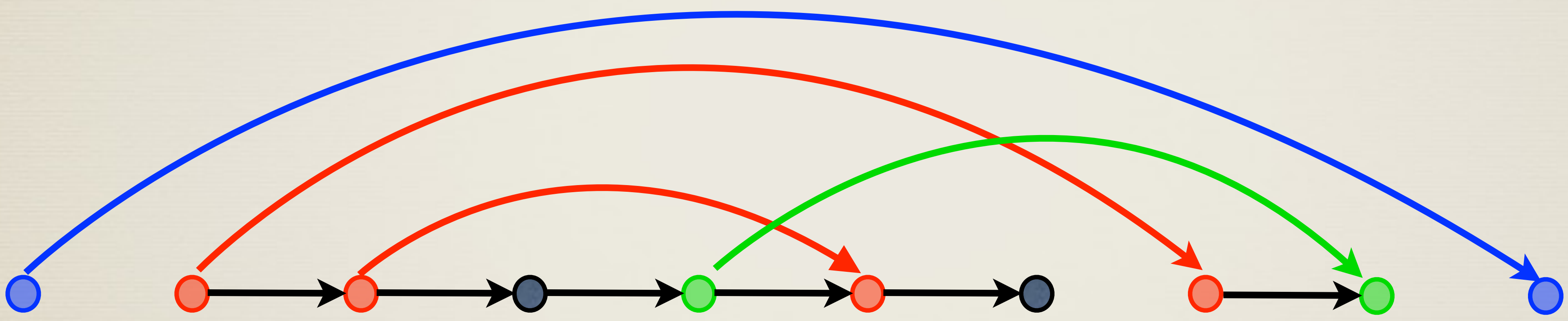


Split CBM with 4 holes

Split-width Operations

The Split Operation:

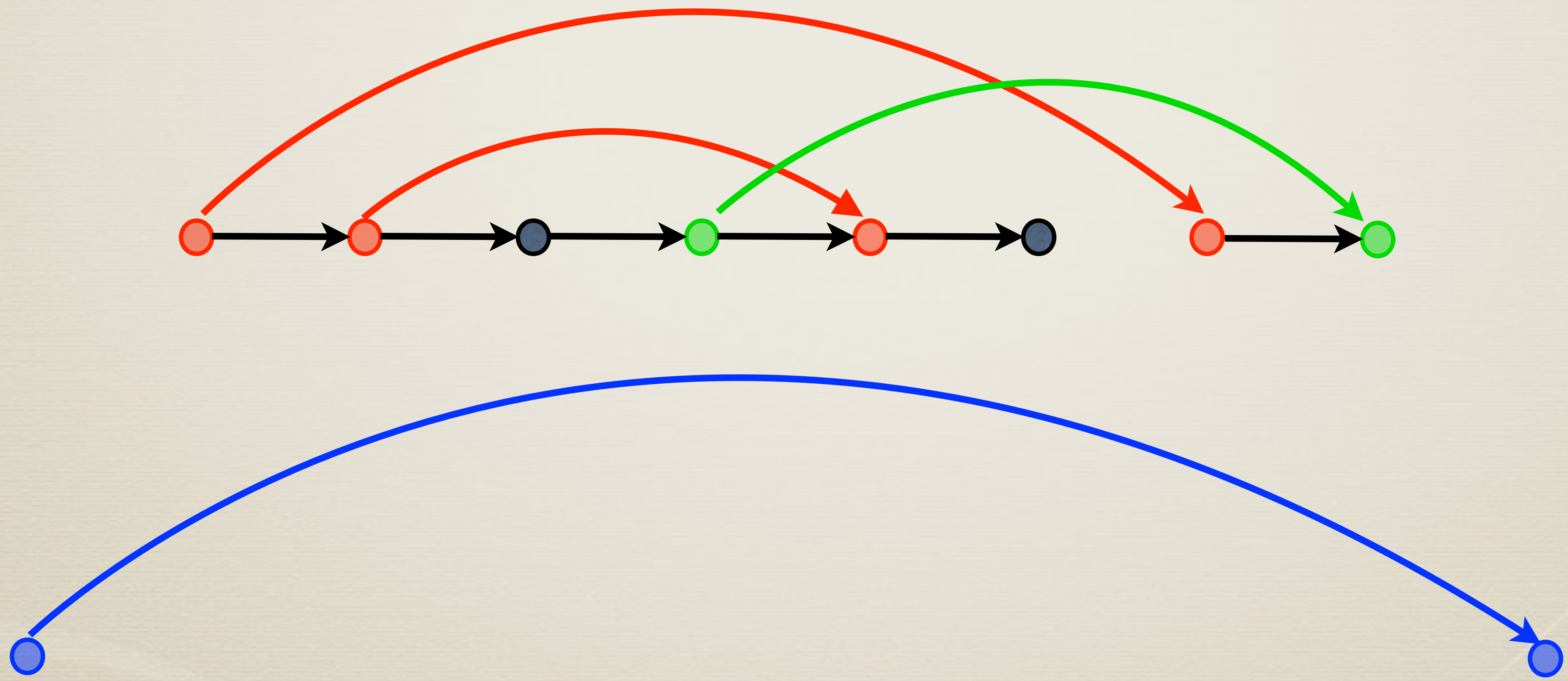
Separate out two disconnected parts into 2 split-CBMs



Split-width Operations

The Split Operation:

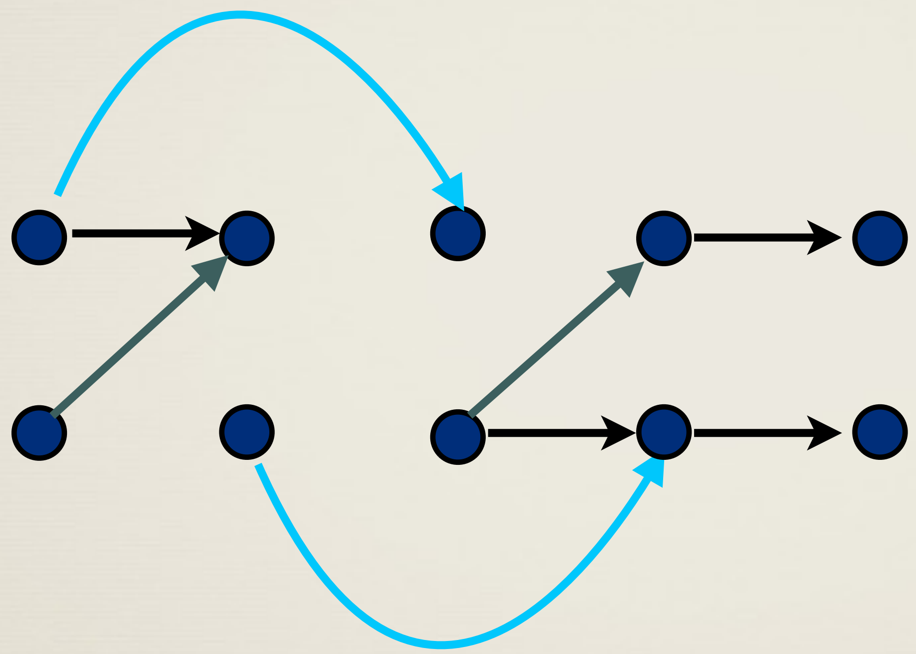
Separate out two disconnected parts into 2 split-CBMs



Split-width Operations

The Split Operation:

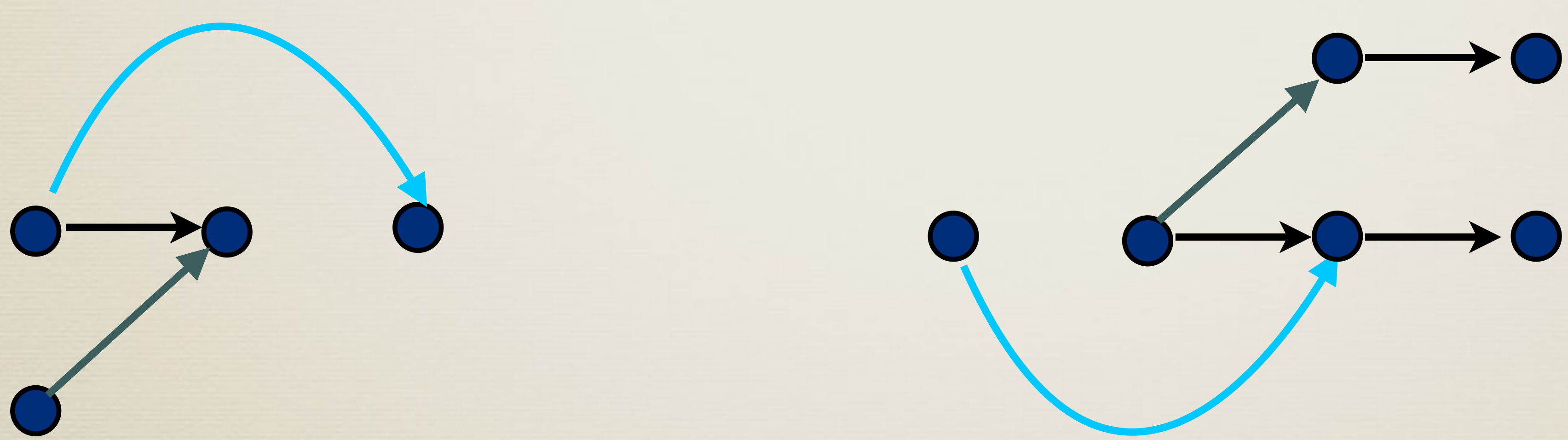
Separate out two disconnected parts into 2 split-CBMs



Split-width Operations

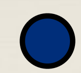
The Split Operation:

Separate out two disconnected parts into 2 split-CBMs

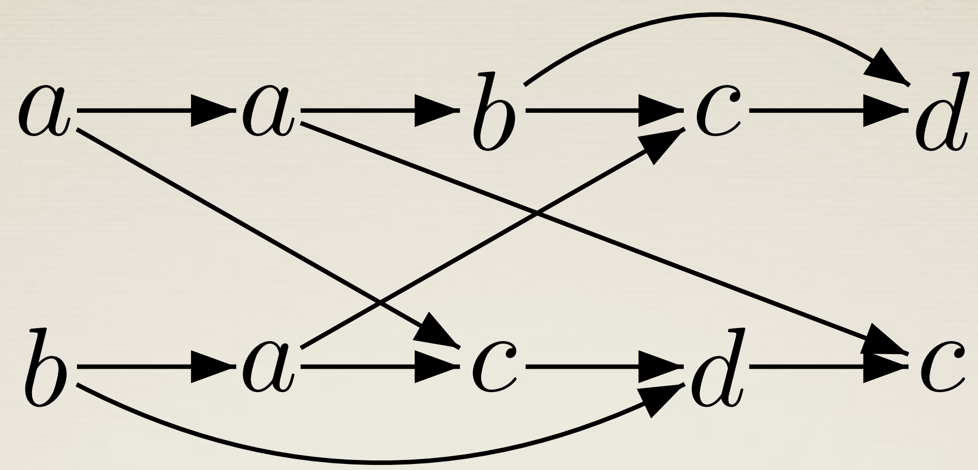


The Basic Split CBMs

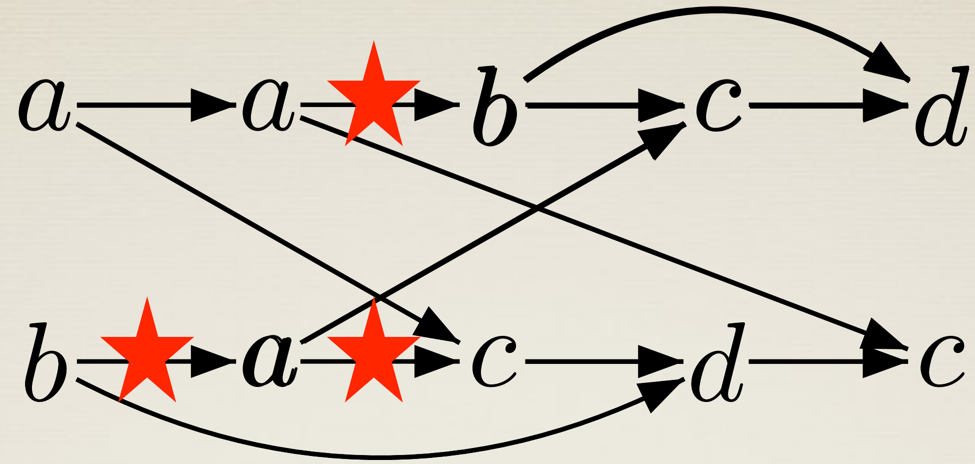
An Internal event:



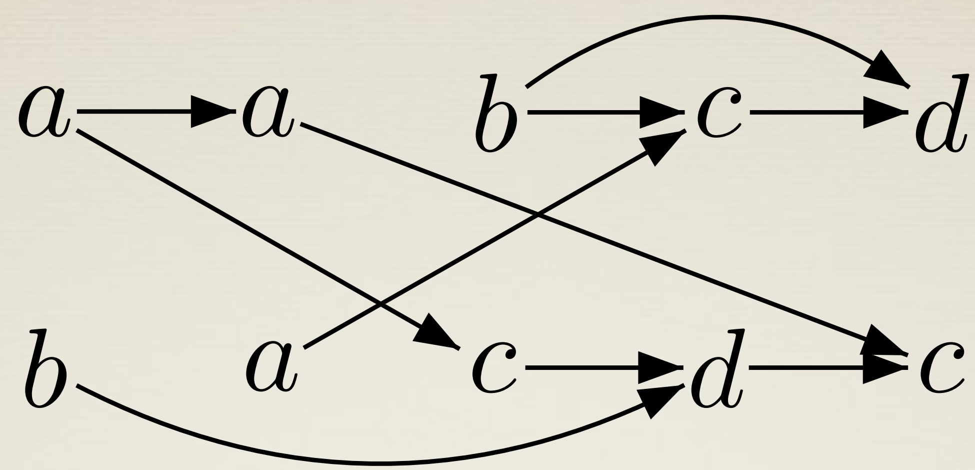
A Communication edge:



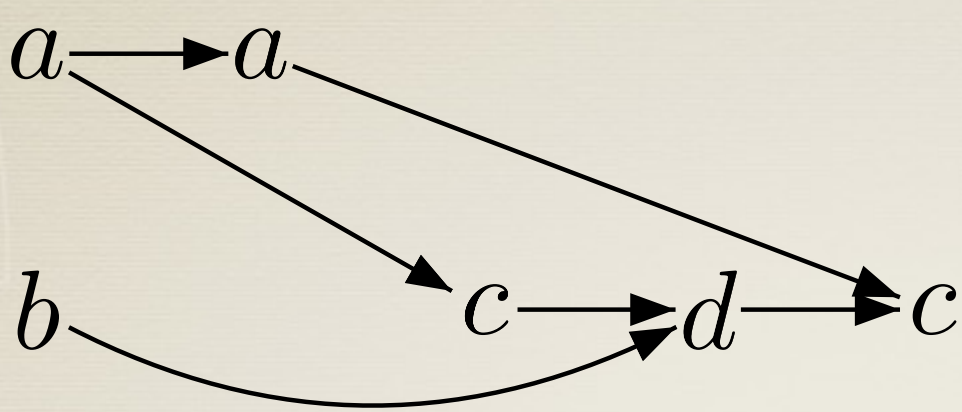
SPLIT DECOMPOSITION OF CBMs

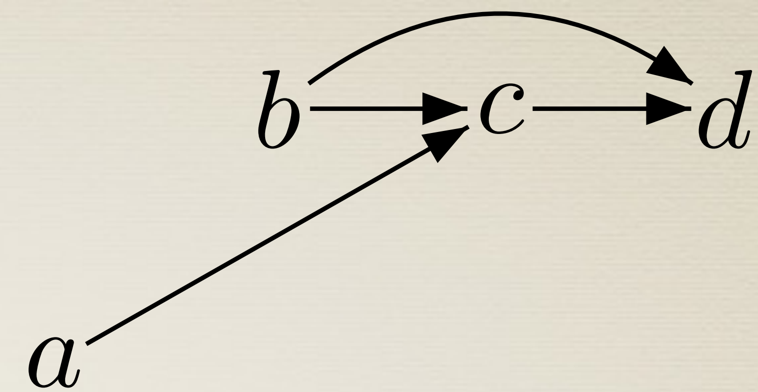


SPLIT DECOMPOSITION OF CBMs

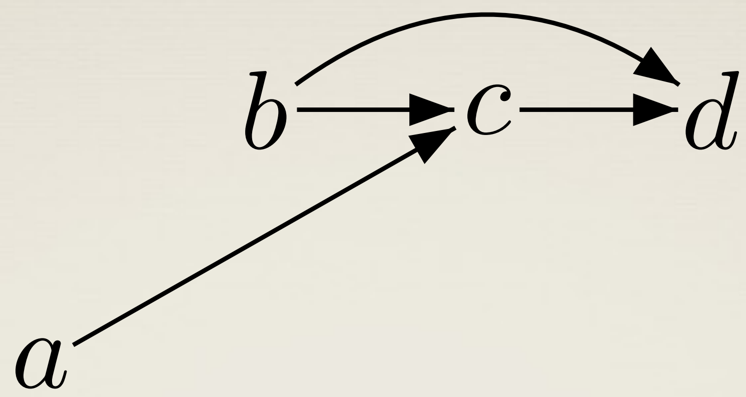


SPLIT DECOMPOSITION OF CBMs

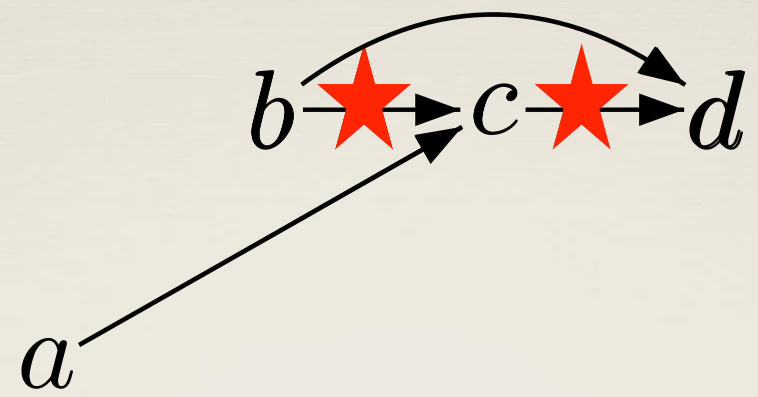




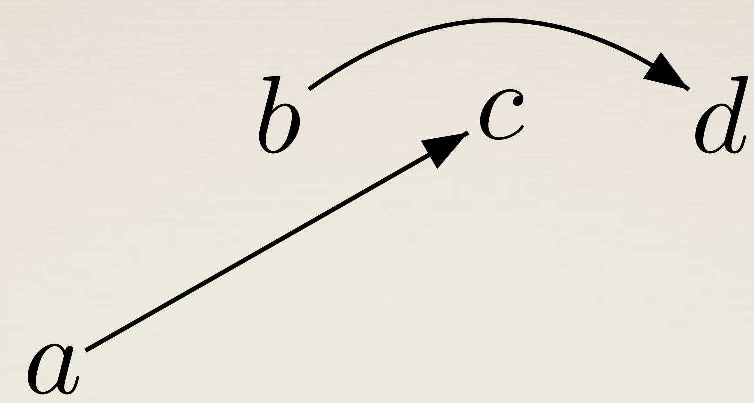
SPLIT DECOMPOSITION OF CBMs



SPLIT DECOMPOSITION OF CBMs

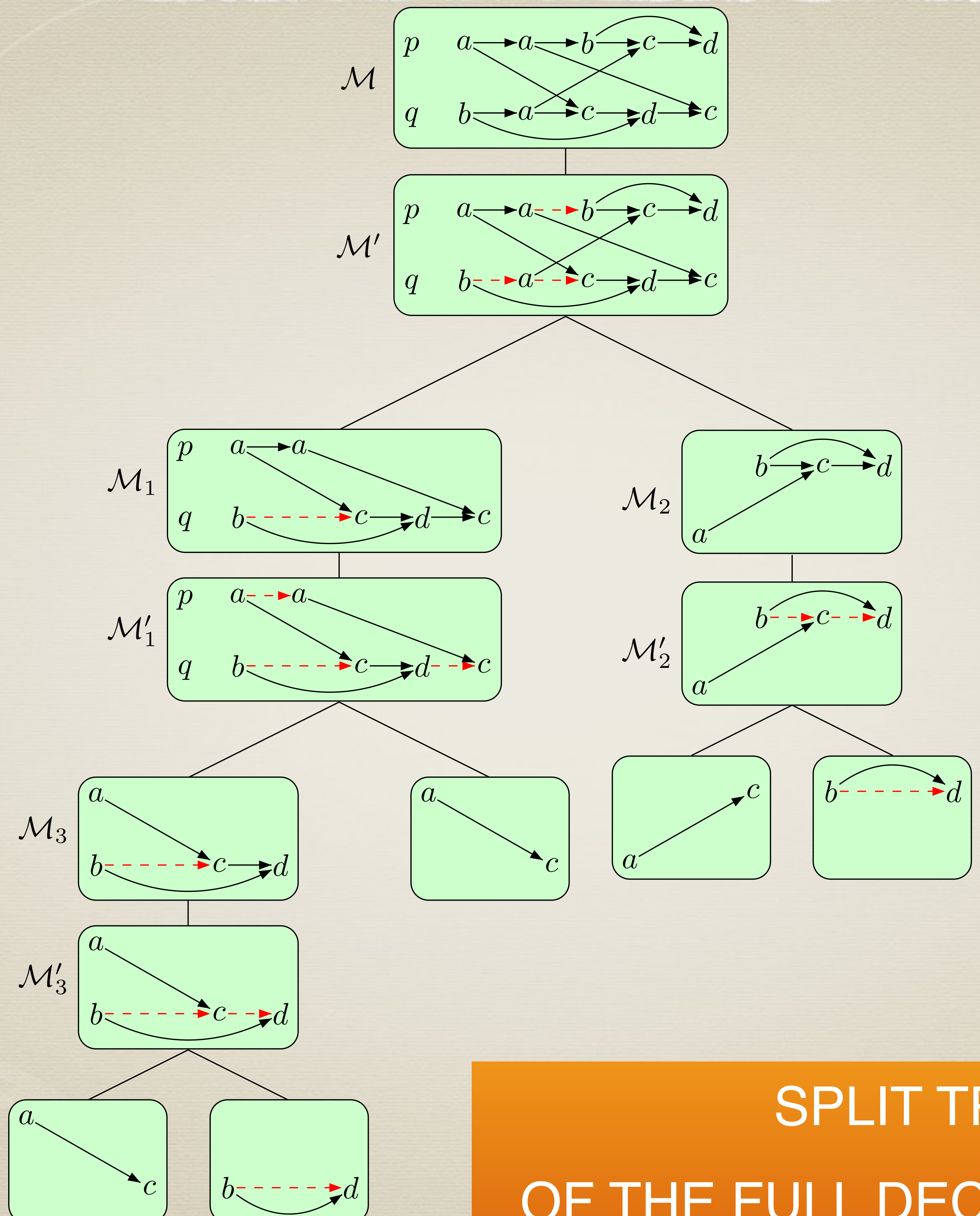


SPLIT DECOMPOSITION OF CBMs



SPLIT DECOMPOSITION OF CBMs

SPLIT DECOMPOSITION OF CBMs

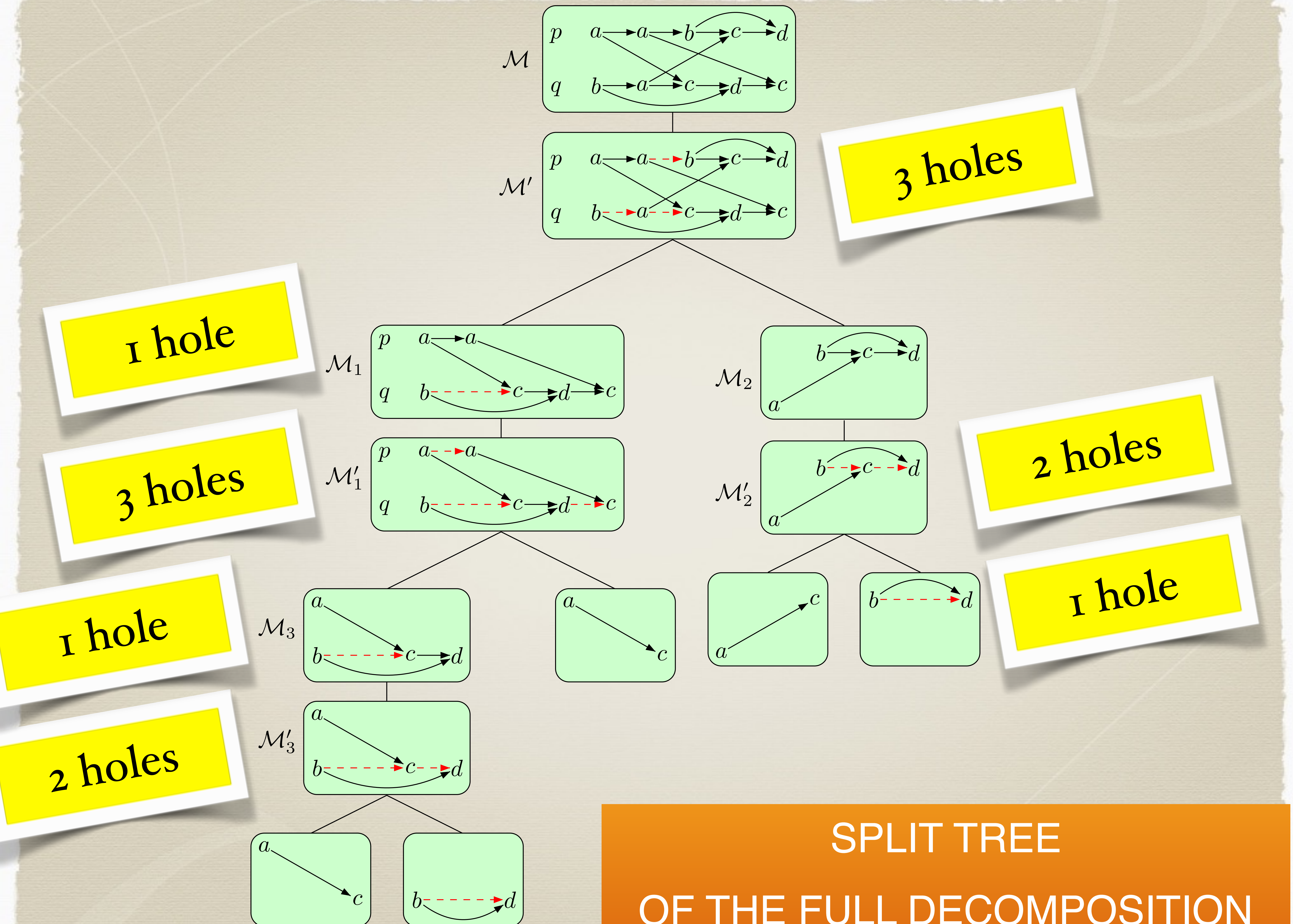


SPLIT TREE

OF THE FULL DECOMPOSITION

SPLIT TREE

OF THE FULL DECOMPOSITION



Split-width

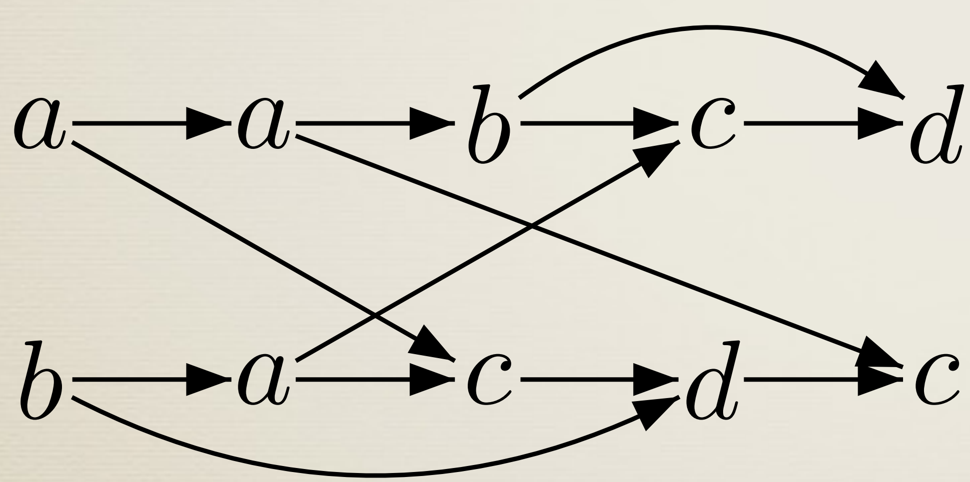
* The width of a decomposition is the maximum number of holes in any split-CBM in the decomposition.

* Split-width of a CBM is the minimum of the widths of its decompositions.

Split-width

* The width of a decomposition is the maximum number of holes in any split-CBM in the decomposition.

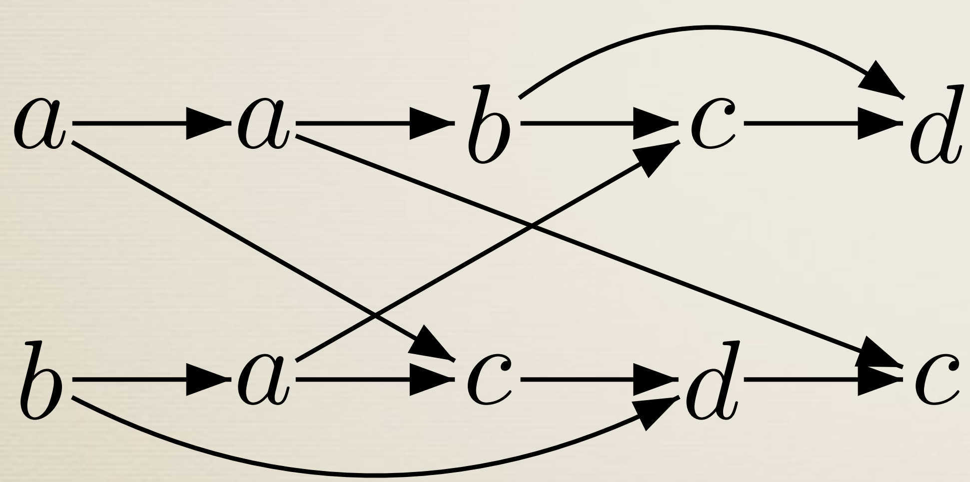
* Split-width of a CBM is the minimum of the widths of its decompositions.



A CBM with split-width = 3

Split-width

- The width of a decomposition is the maximum number of holes in any split-CBM in the decomposition.
- Split-width of a CBM is the minimum of the widths of its decompositions.

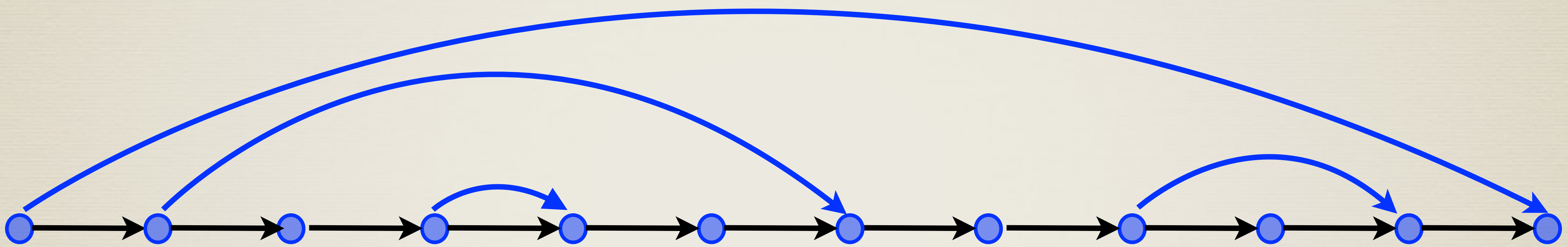


A CBM with split-width = 3

Split-width of a set of CBMs is the maximum of their split-widths

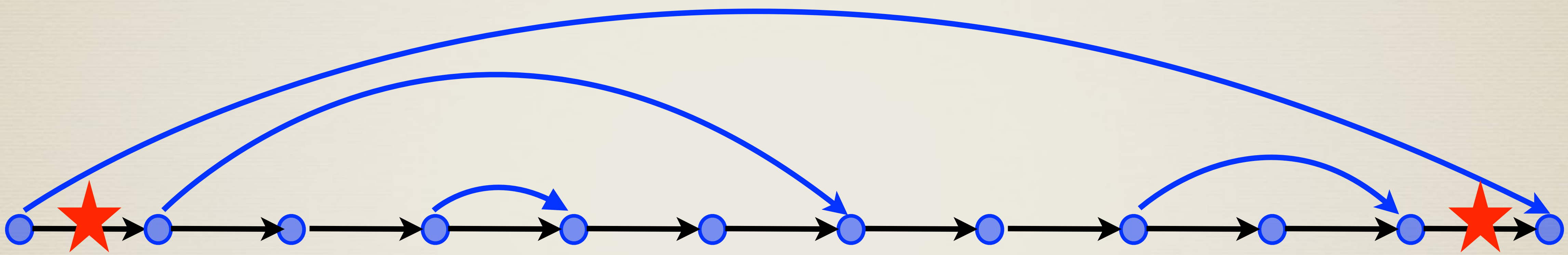
Split-width of nested words

* The class of nested words has split-width bounded by 2



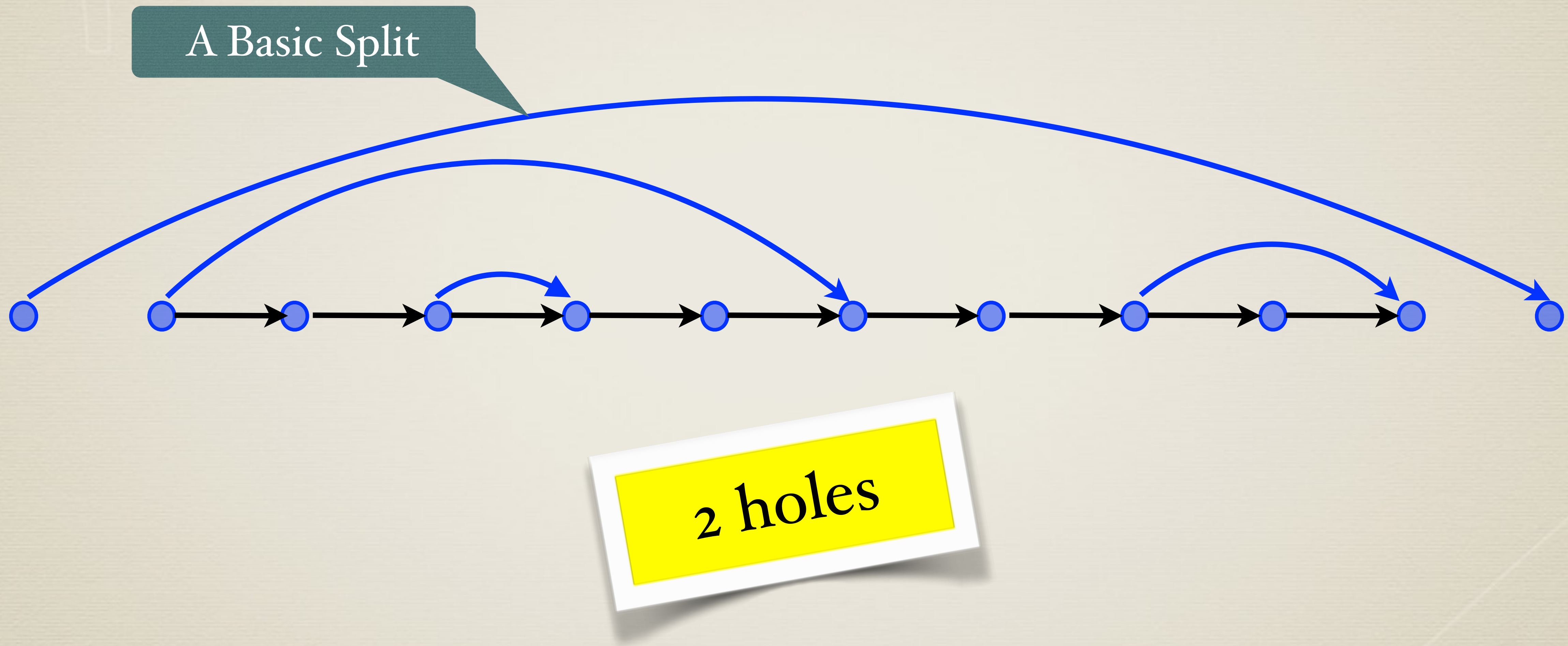
Split-width of nested words

* The class of nested words has split-width bounded by 2



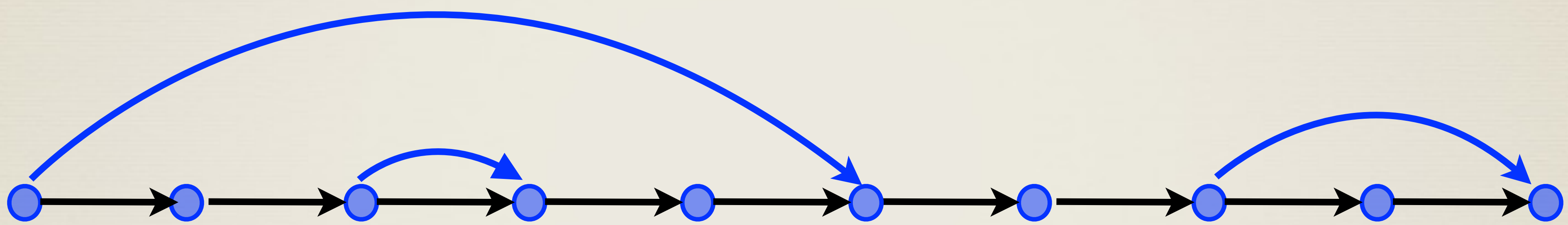
Split-width of nested words

* The class of nested words has split-width bounded by 2



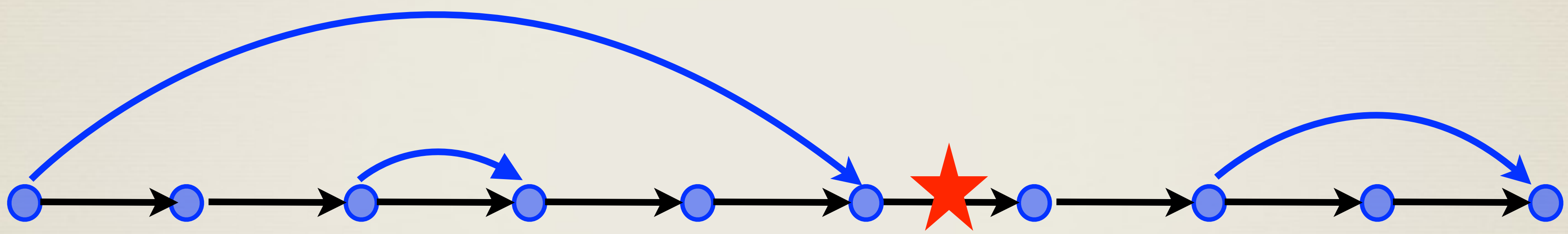
Split-width of nested words

* The class of nested words has split-width bounded by 2



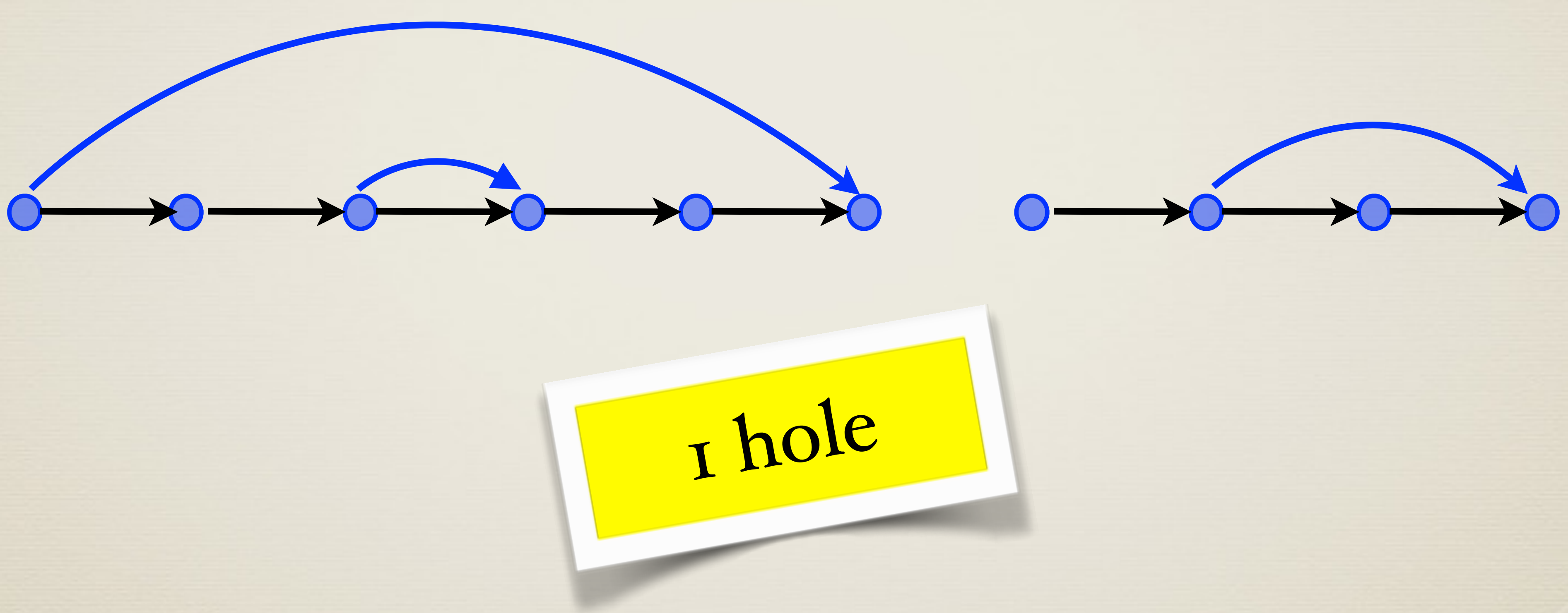
Split-width of nested words

* The class of nested words has split-width bounded by 2



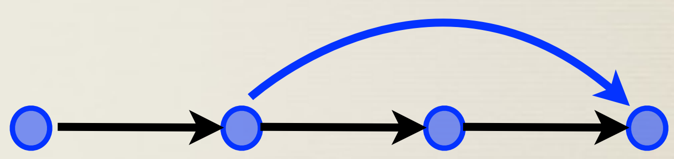
Split-width of nested words

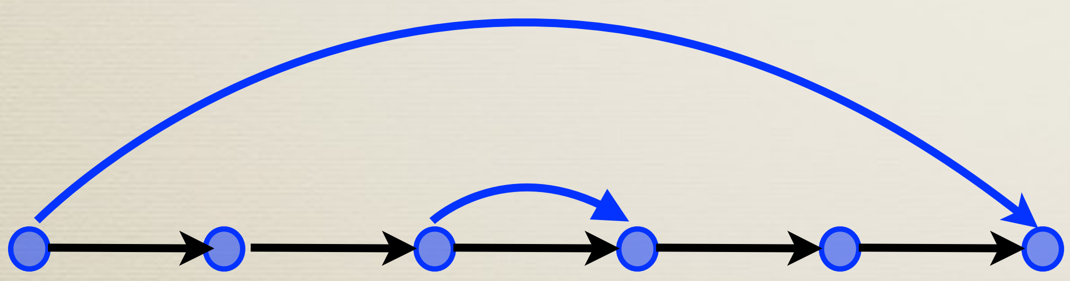
* The class of nested words has split-width bounded by 2



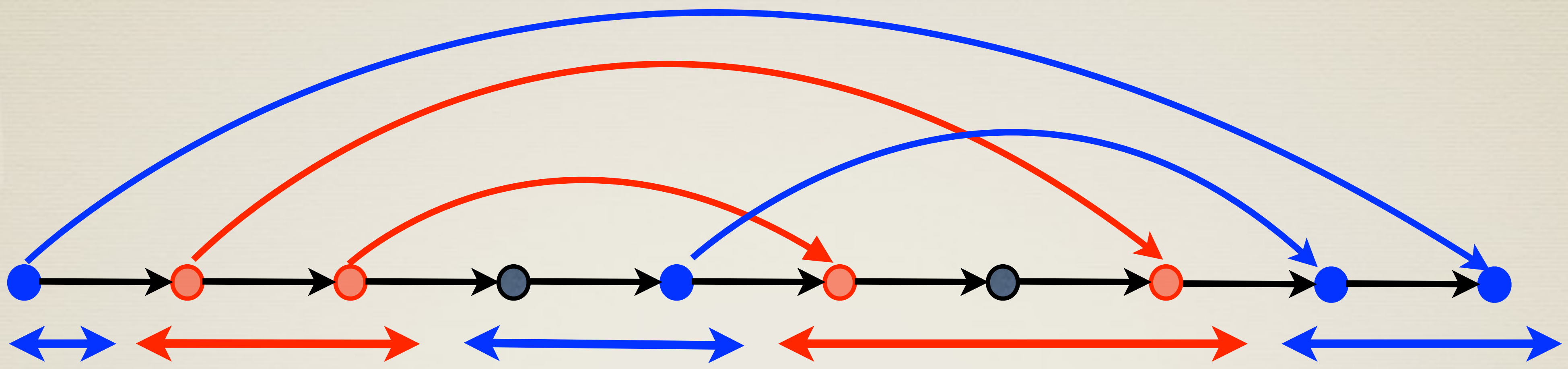
Split-width of nested words

* The class of nested words has split-width bounded by 2

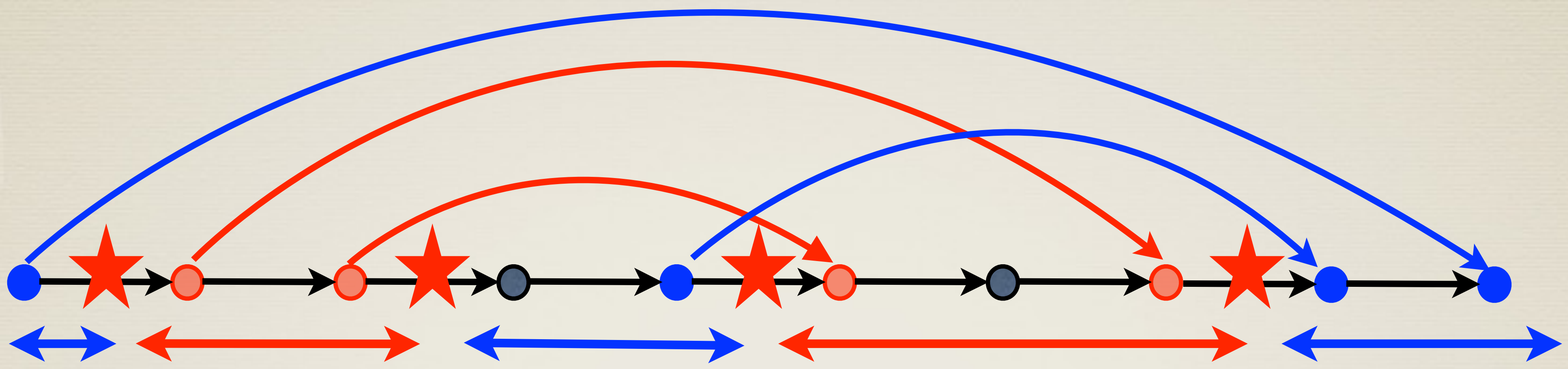




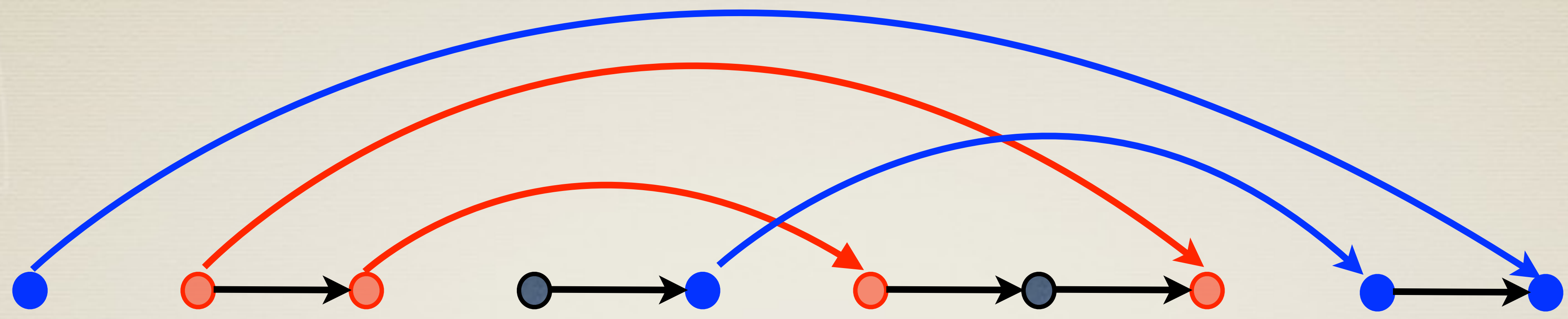
Bounded-context Runs



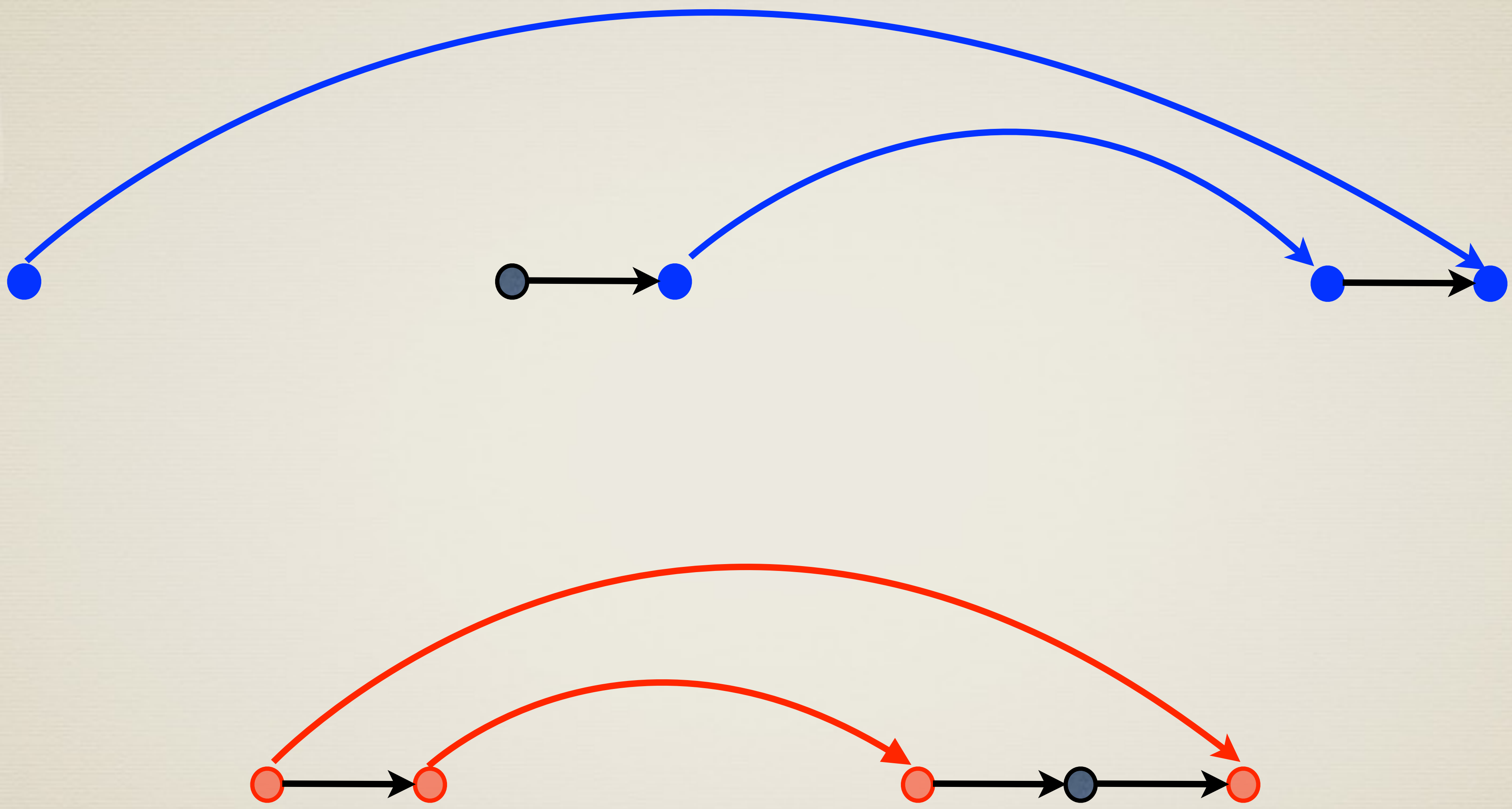
Bounded-context Runs



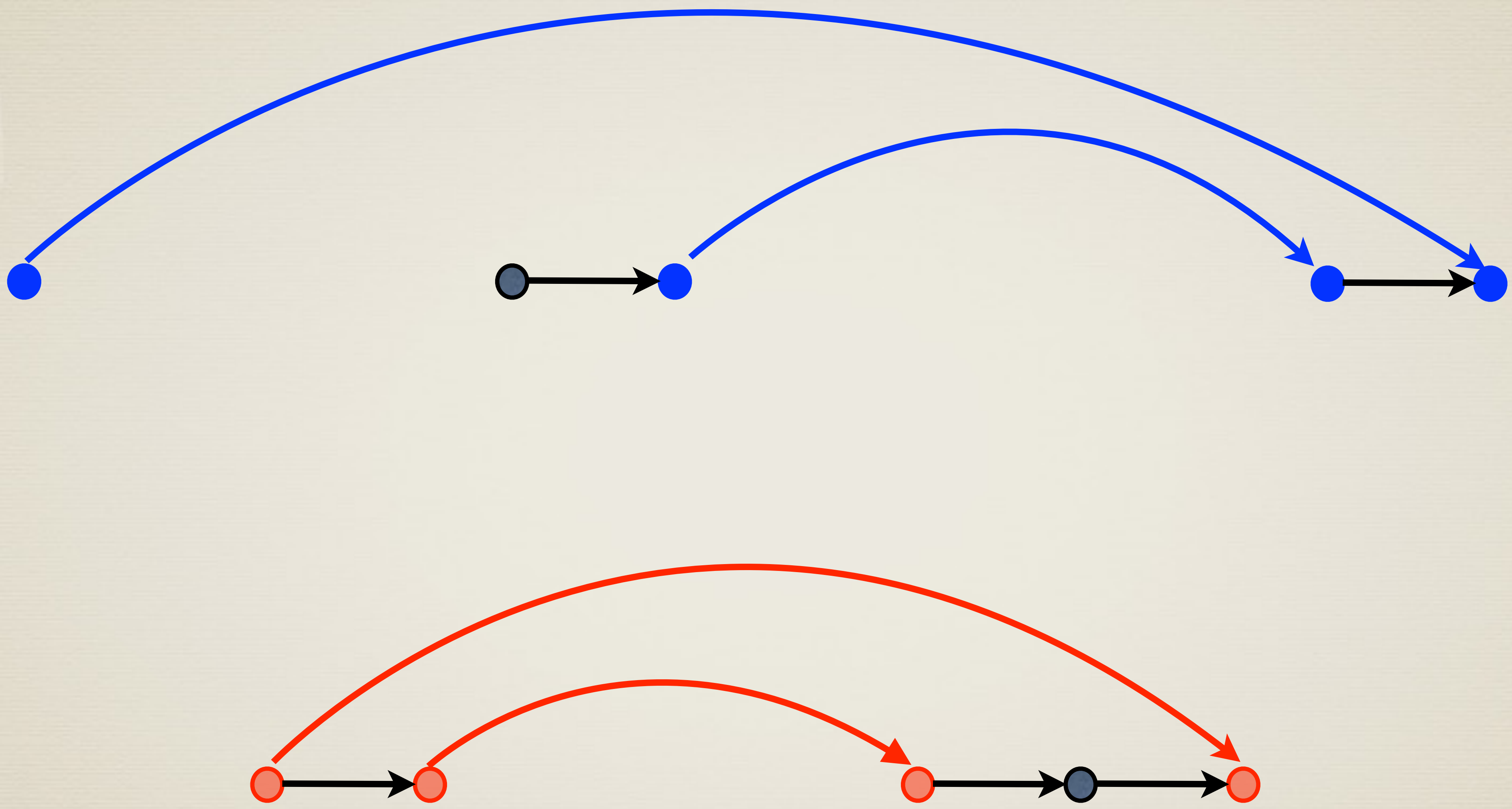
Bounded-context Runs



Bounded-context Runs



Bounded-context Runs



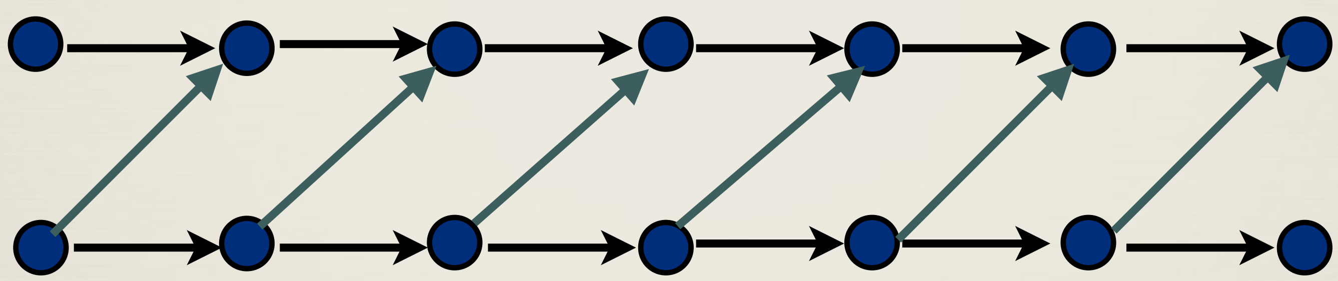
A k context CBM has split-width k+1

Existentially k bounded MSCs

There is a linearisation where no channel contains more than k values at any point along the linearization.

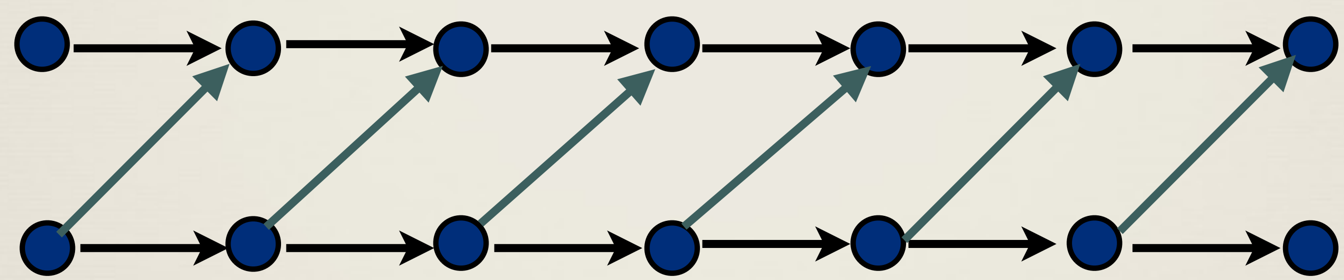
Existentially k bounded MSCs

There is a linearisation where no channel contains more than k values at any point along the linearization.



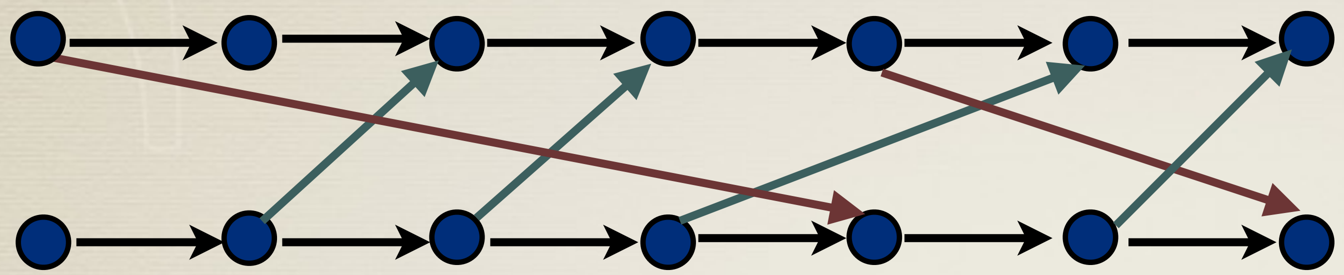
Existentially k bounded MSCs

There is a linearisation where no channel contains more than k values at any point along the linearization.

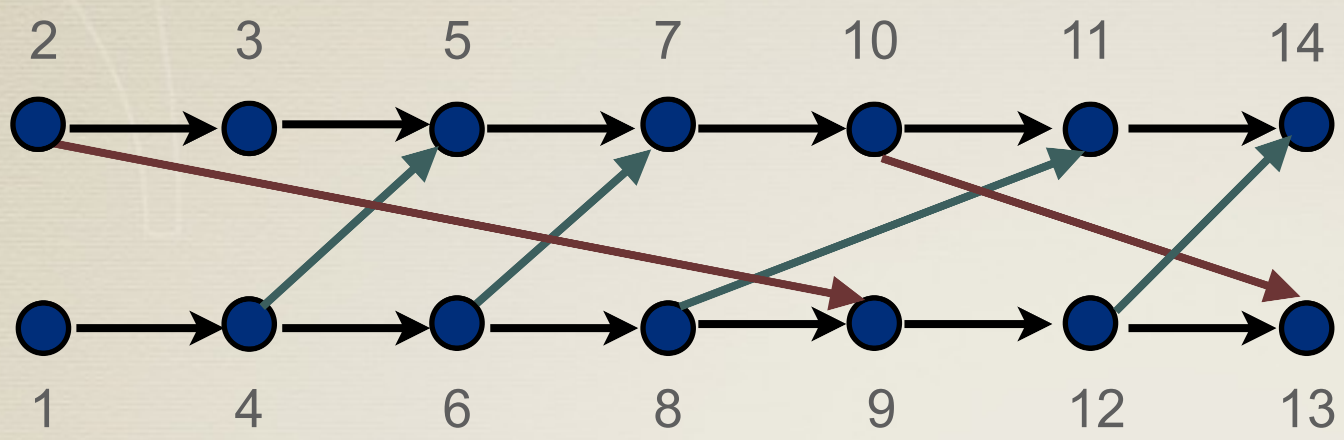


An existentially 1 bounded behaviour.

Existentially k bounded MSCs

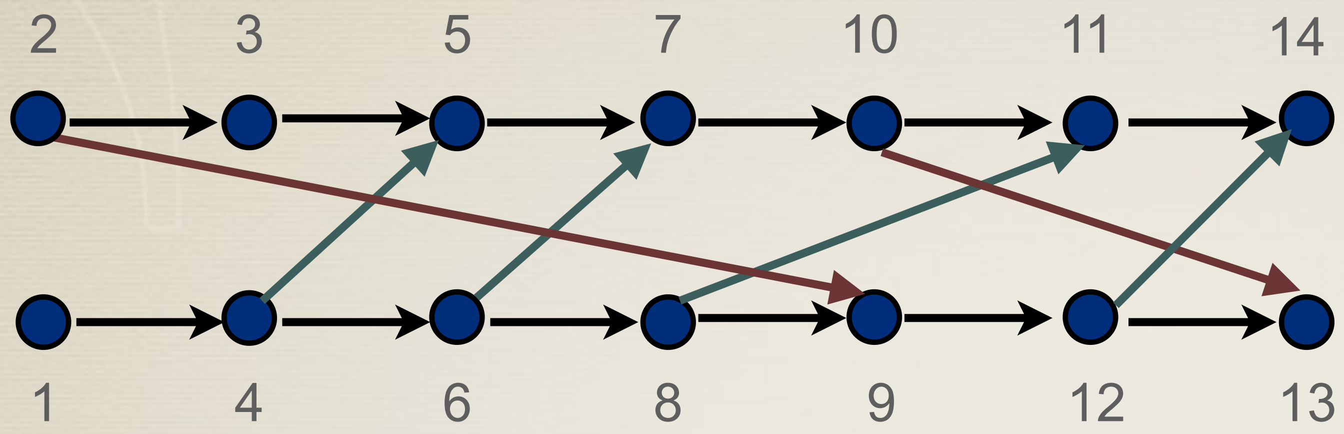


Existentially k bounded MSCs

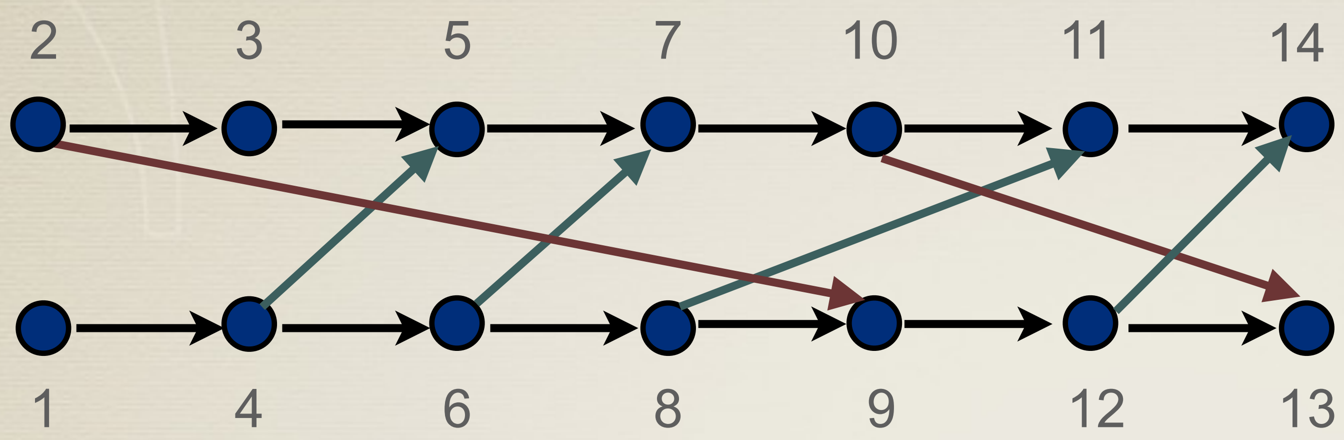


a 2-bounded sequentialisation

Existentially k bounded MSCs

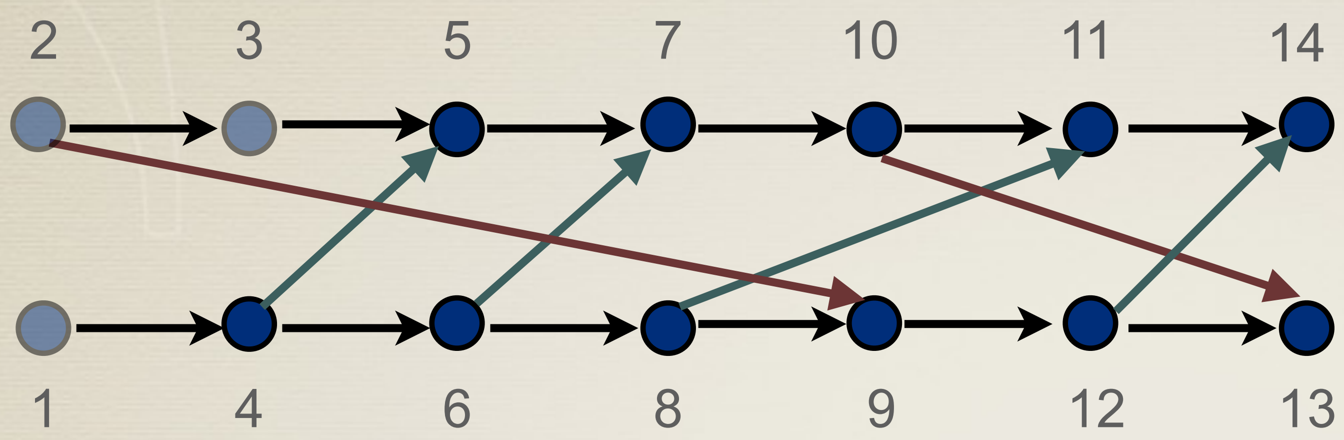


Existentially k bounded MSCs



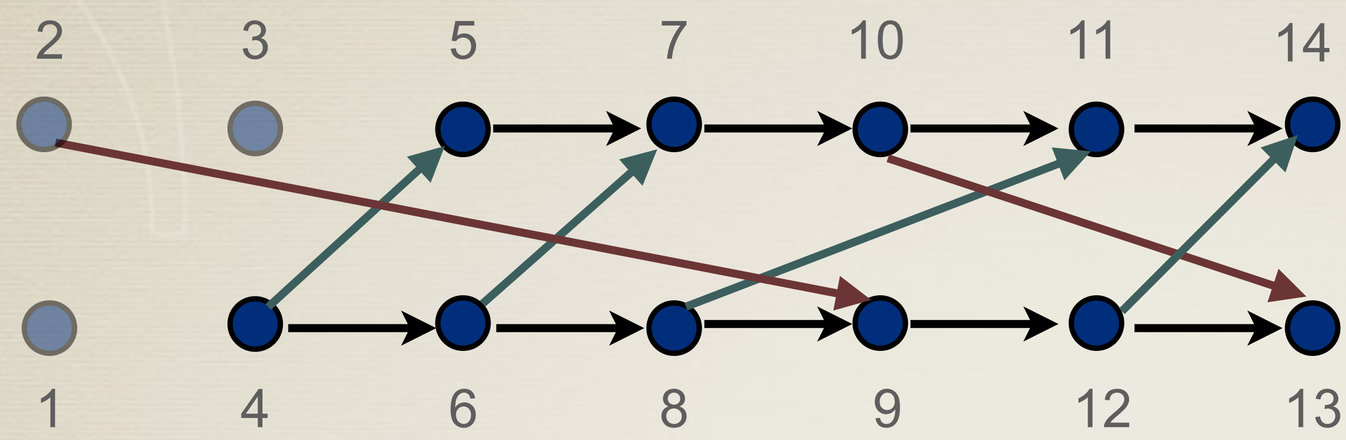
* Cut process edges from the first k+1 events in the k bounded sequentialisation.
* Remove message edges, internal events that can be.
* Expand to have k+1 events and repeat this process.

Existentially k bounded MSCs



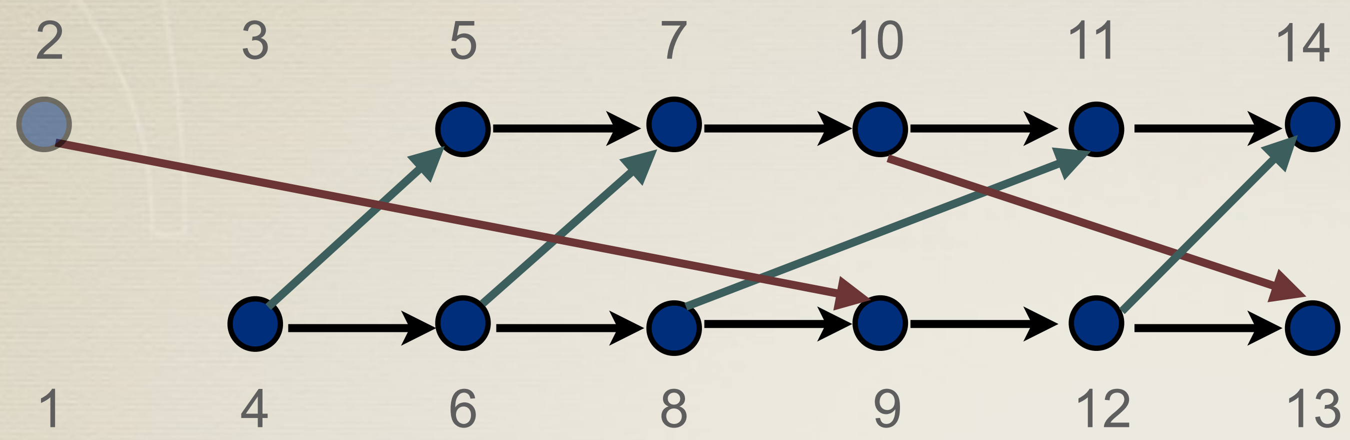
* Cut process edges from the first k+1 events in the k bounded sequentialisation.
* Remove message edges, internal events that can be.
* Expand to have k+1 events and repeat this process.

Existentially k bounded MSCs



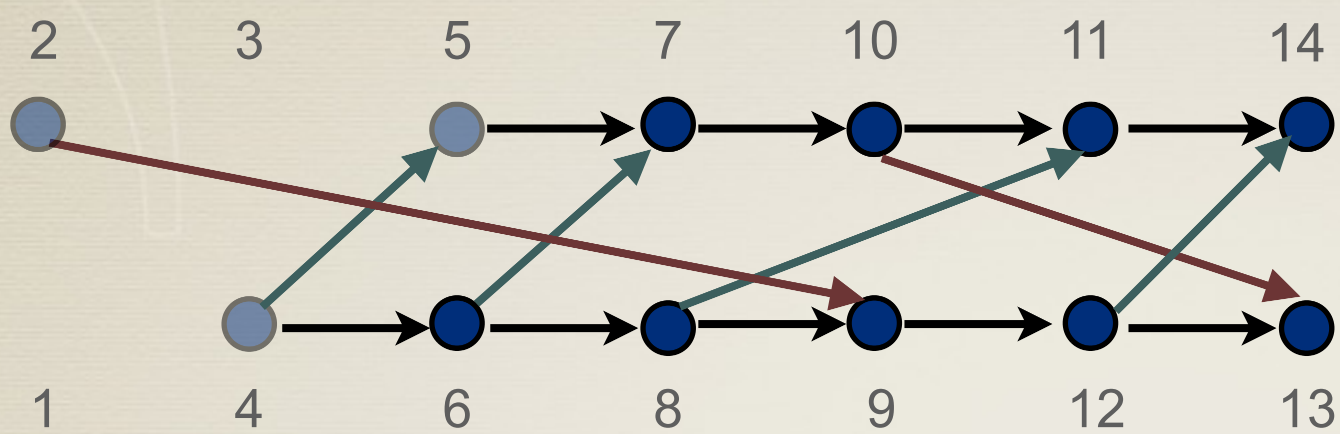
* Cut process edges from the first k+1 events in the k bounded sequentialisation.
* Remove message edges, internal events that can be.
* Expand to have k+1 events and repeat this process.

Existentially k bounded MSCs



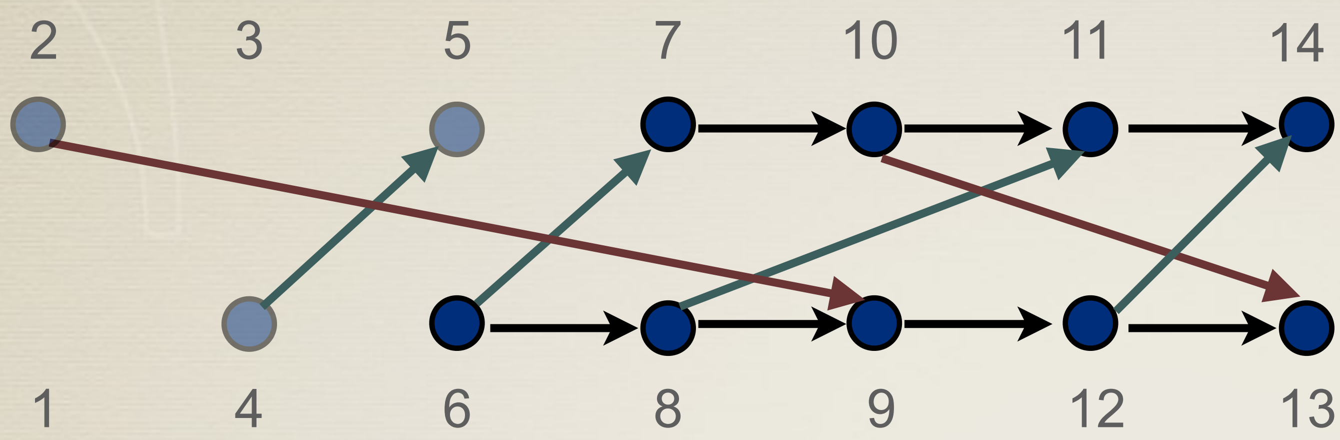
* Cut process edges from the first k+1 events in the k bounded sequentialisation.
* Remove message edges, internal events that can be.
* Expand to have k+1 events and repeat this process.

Existentially k bounded MSCs



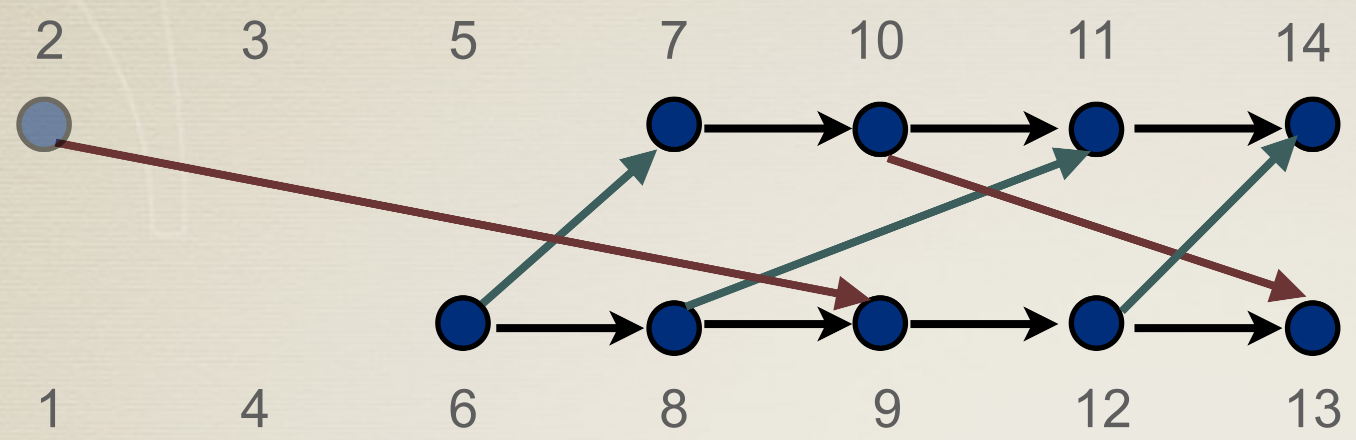
* Cut process edges from the first k+1 events in the k bounded sequentialisation.
* Remove message edges, internal events that can be.
* Expand to have k+1 events and repeat this process.

Existentially k bounded MSCs



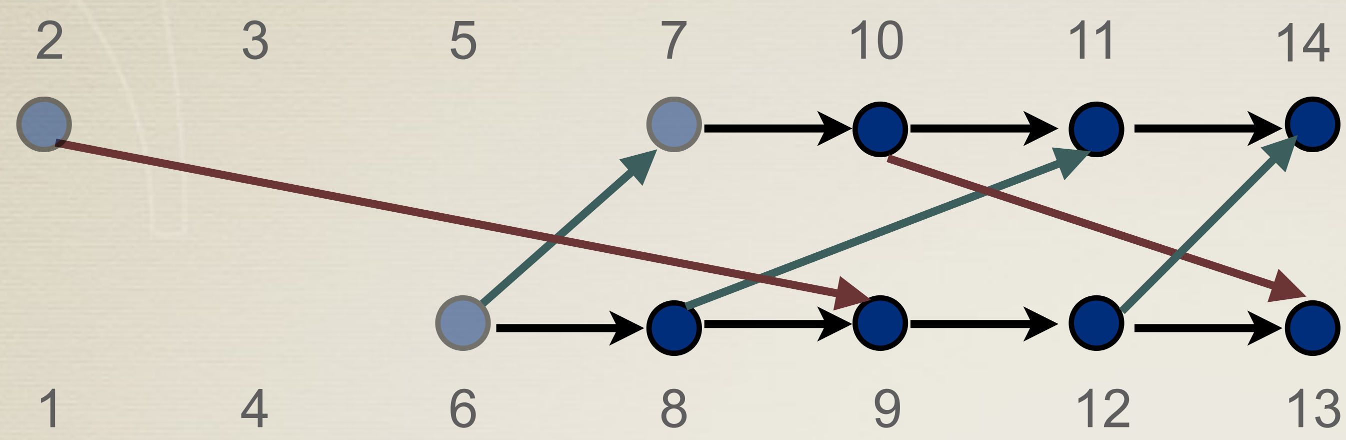
* Cut process edges from the first k+1 events in the k bounded sequentialisation.
* Remove message edges, internal events that can be.
* Expand to have k+1 events and repeat this process.

Existentially k bounded MSCs



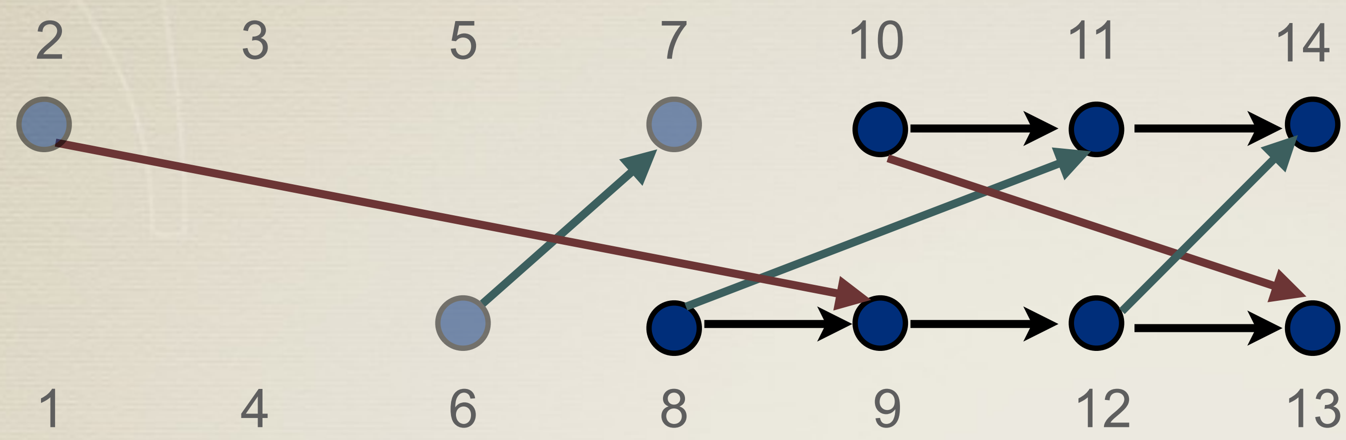
- Cut process edges from the first k+1 events in the k bounded sequentialisation.
- Remove message edges, internal events that can be.
- Expand to have k+1 events and repeat this process.

Existentially k bounded MSCs



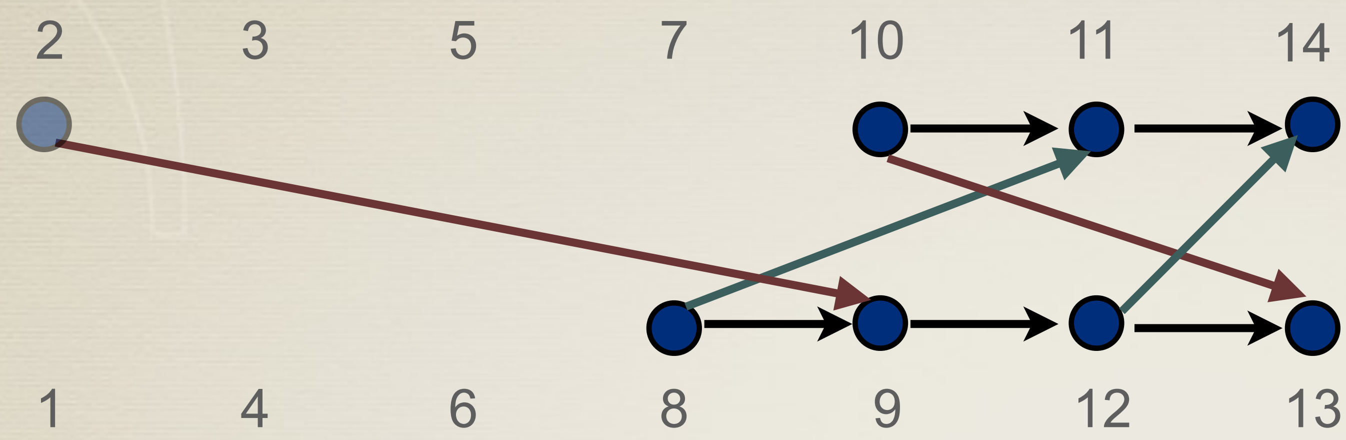
* Cut process edges from the first k+1 events in the k bounded sequentialisation.
* Remove message edges, internal events that can be.
* Expand to have k+1 events and repeat this process.

Existentially k bounded MSCs



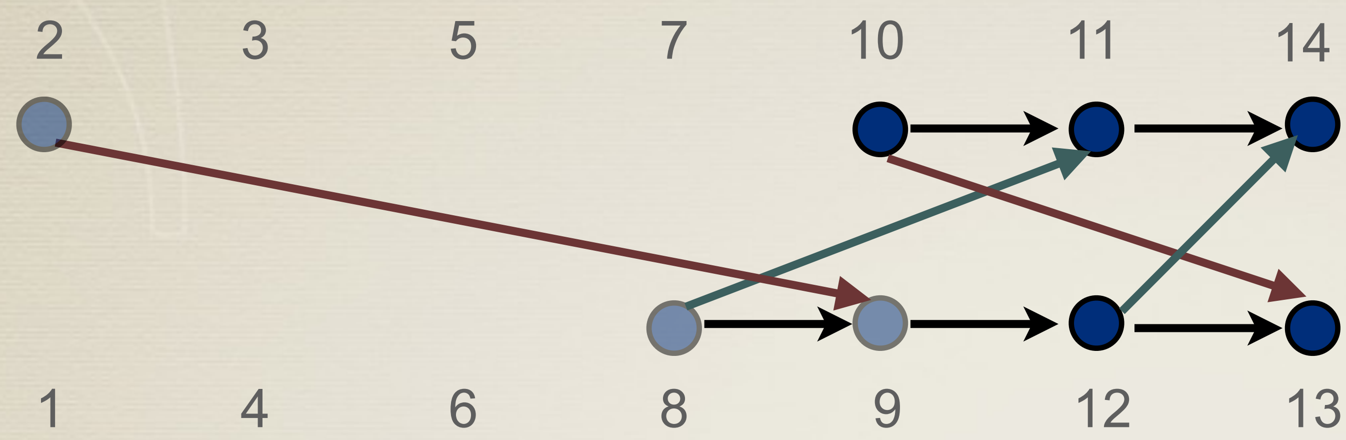
- Cut process edges from the first k+1 events in the k bounded sequentialisation.
- Remove message edges, internal events that can be.
- Expand to have k+1 events and repeat this process.

Existentially k bounded MSCs



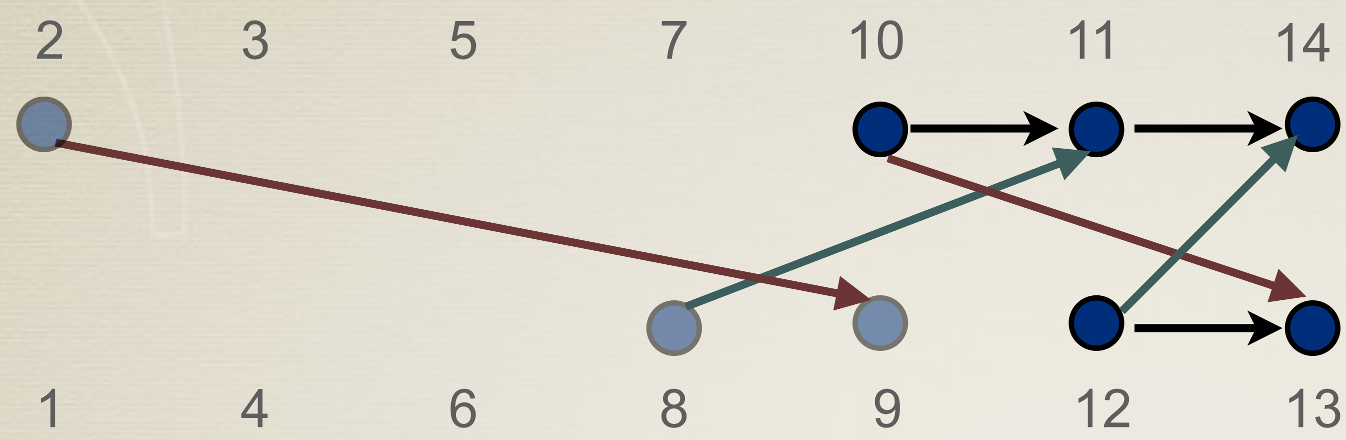
* Cut process edges from the first k+1 events in the k bounded sequentialisation.
* Remove message edges, internal events that can be.
* Expand to have k+1 events and repeat this process.

Existentially k bounded MSCs



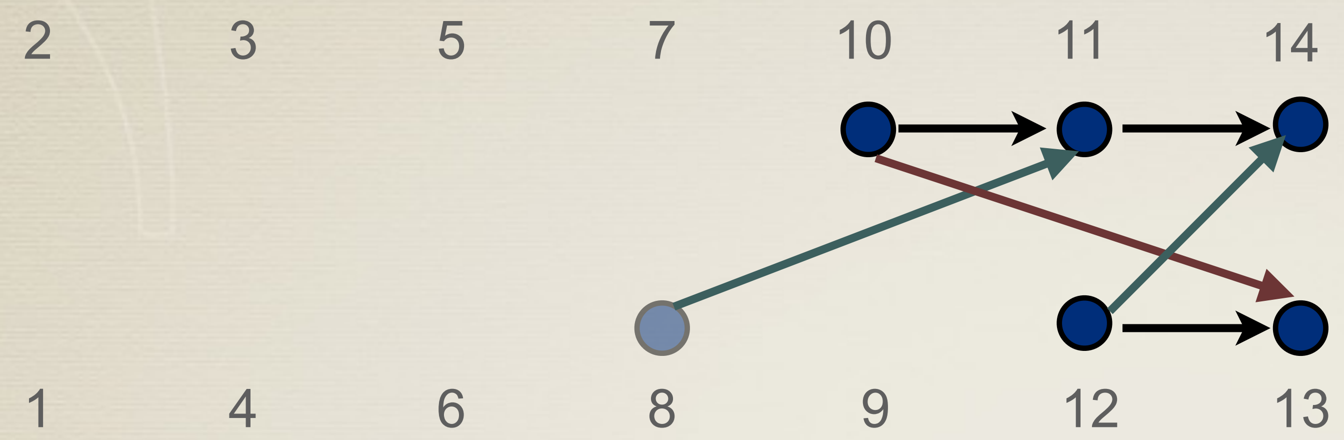
* Cut process edges from the first k+1 events in the k bounded sequentialisation.
* Remove message edges, internal events that can be.
* Expand to have k+1 events and repeat this process.

Existentially k bounded MSCs



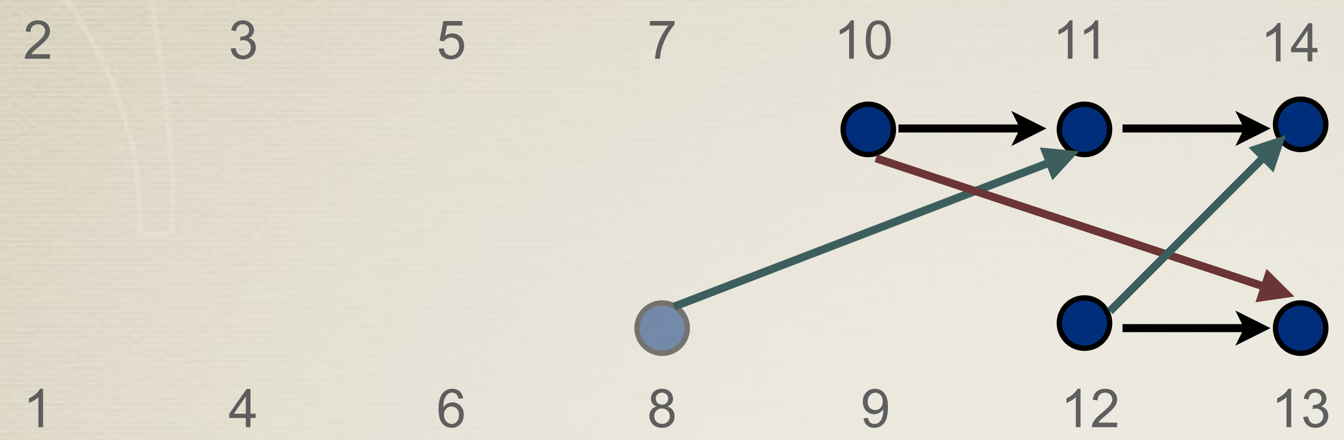
* Cut process edges from the first k+1 events in the k bounded sequentialisation.
* Remove message edges, internal events that can be.
* Expand to have k+1 events and repeat this process.

Existentially k bounded MSCs



- Cut process edges from the first k+1 events in the k bounded sequentialisation.
- Remove message edges, internal events that can be.
- Expand to have k+1 events and repeat this process.

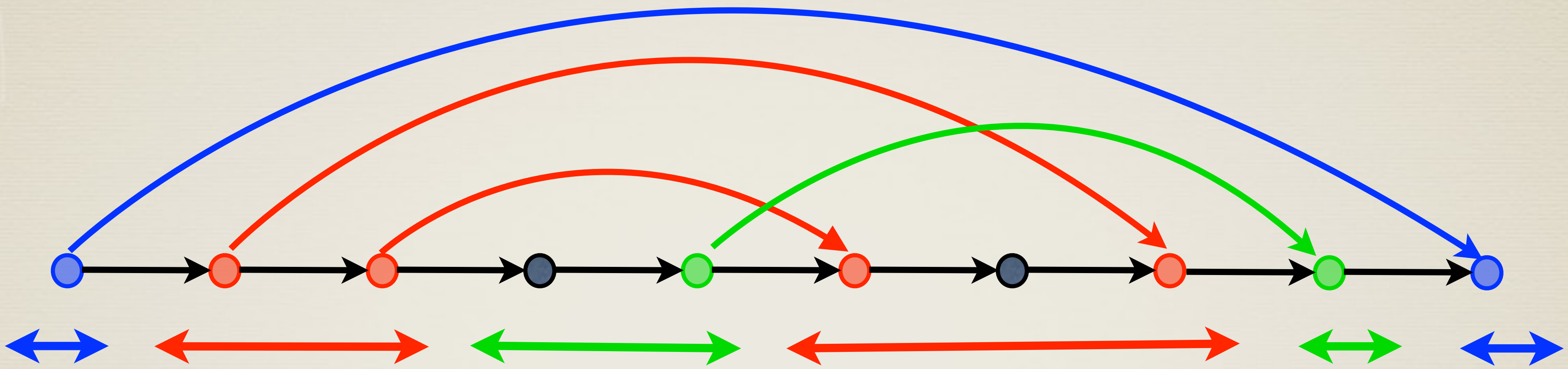
Existentially k bounded MSCs



* Cut process edges from the first k+1 events in the k bounded sequentialisation.
* Remove message edges, internal events that can be.
* Expand to have k+1 events and repeat this process.

A k existentially bounded MSC/CBM has split-width k+1

Bounded Scope Behaviours

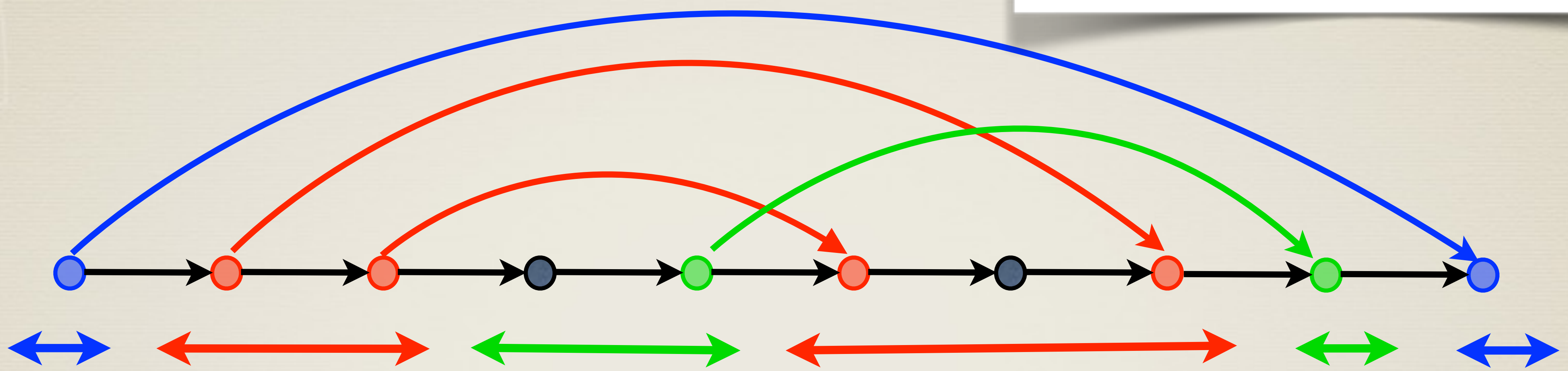


* Between any call and the corresponding return there are at most k context-switches

LaTorreNapoli'11

Bounded Scope Behaviours

a 5 scope bounded cbm

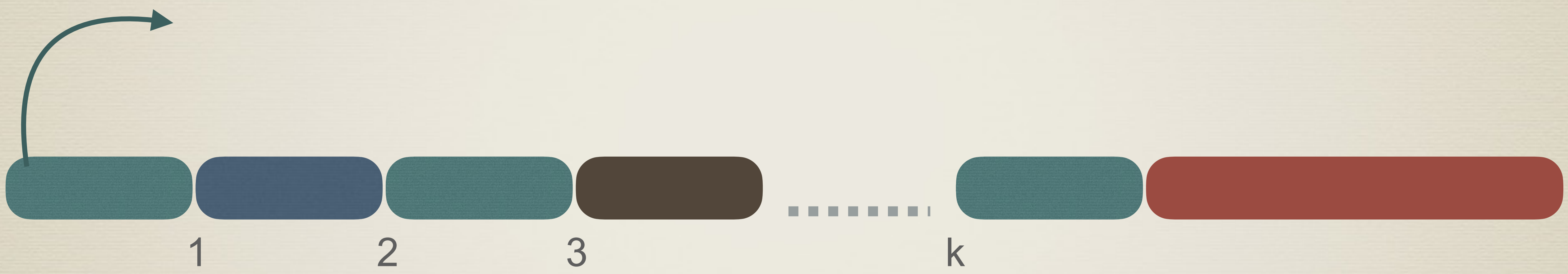


* Between any call and the corresponding return there are at most k context-switches

LaTorreNapoli'11

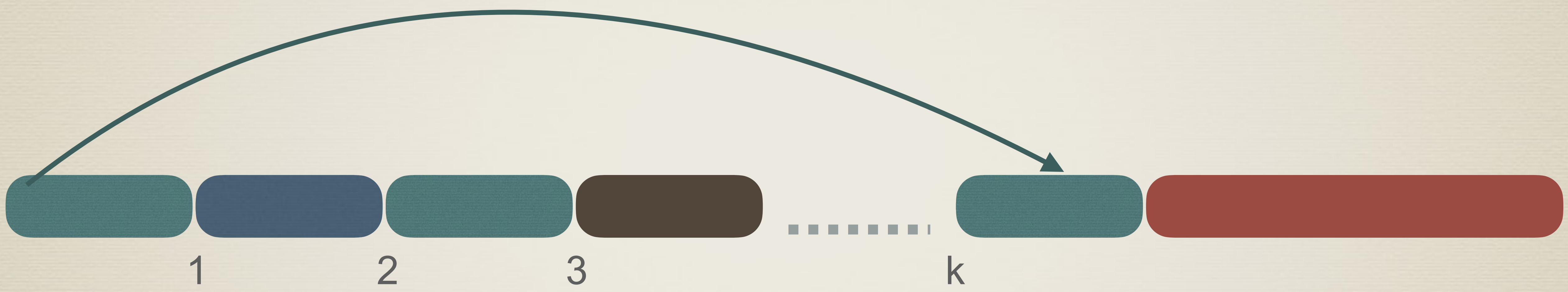
Bounded Scope Behaviours

* Between any call and the corresponding return there are at most k context-switches



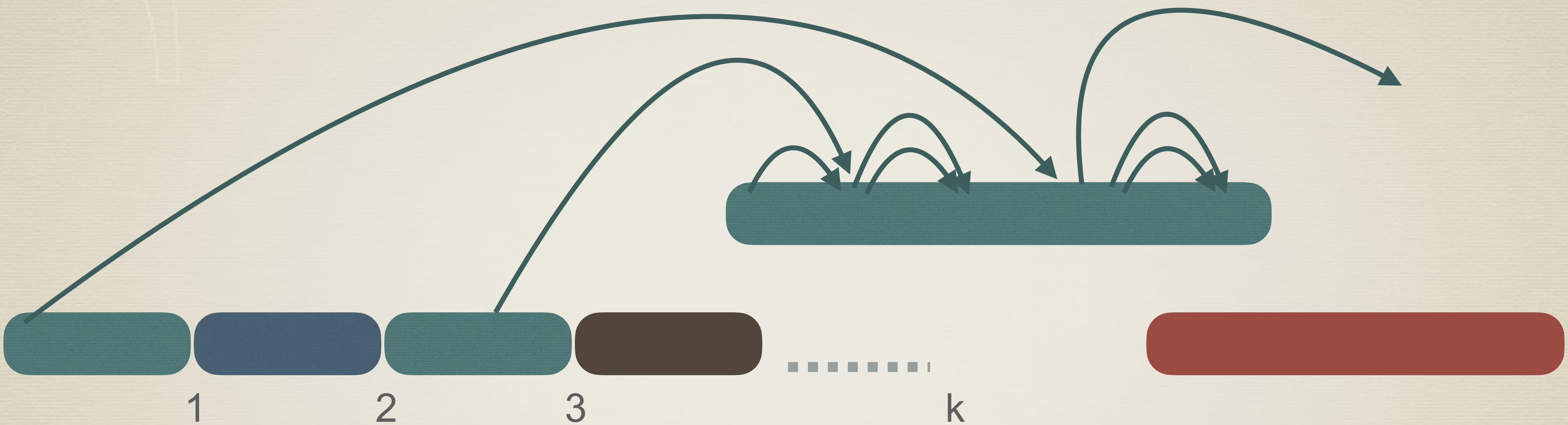
Bounded Scope Behaviours

* Between any call and the corresponding return there are at most k context-switches



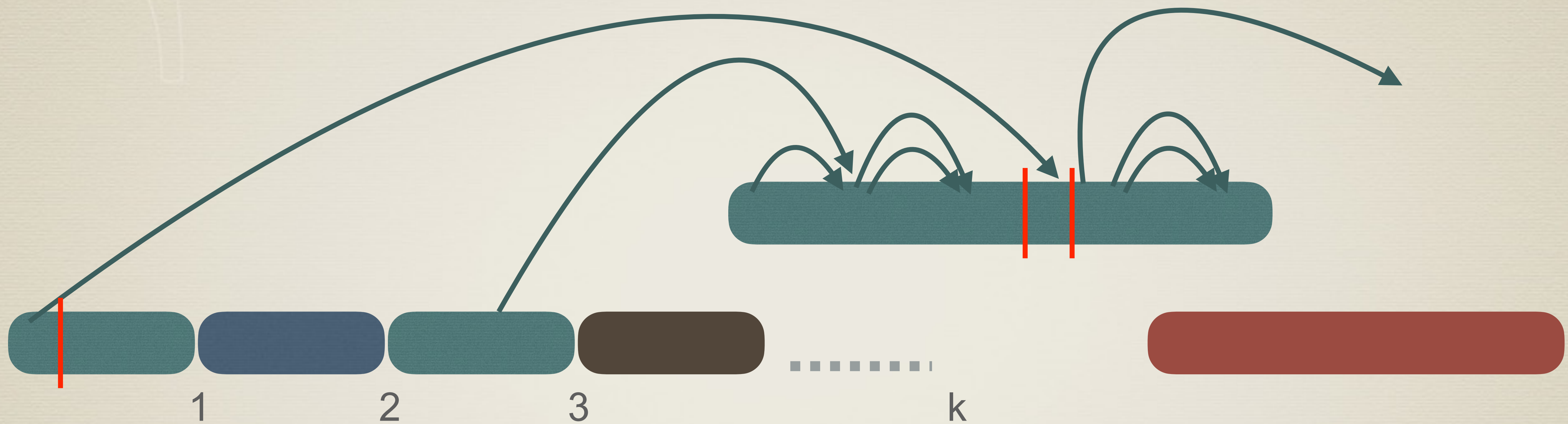
Bounded Scope Behaviours

* Between any call and the corresponding return there are at most k context-switches



Bounded Scope Behaviours

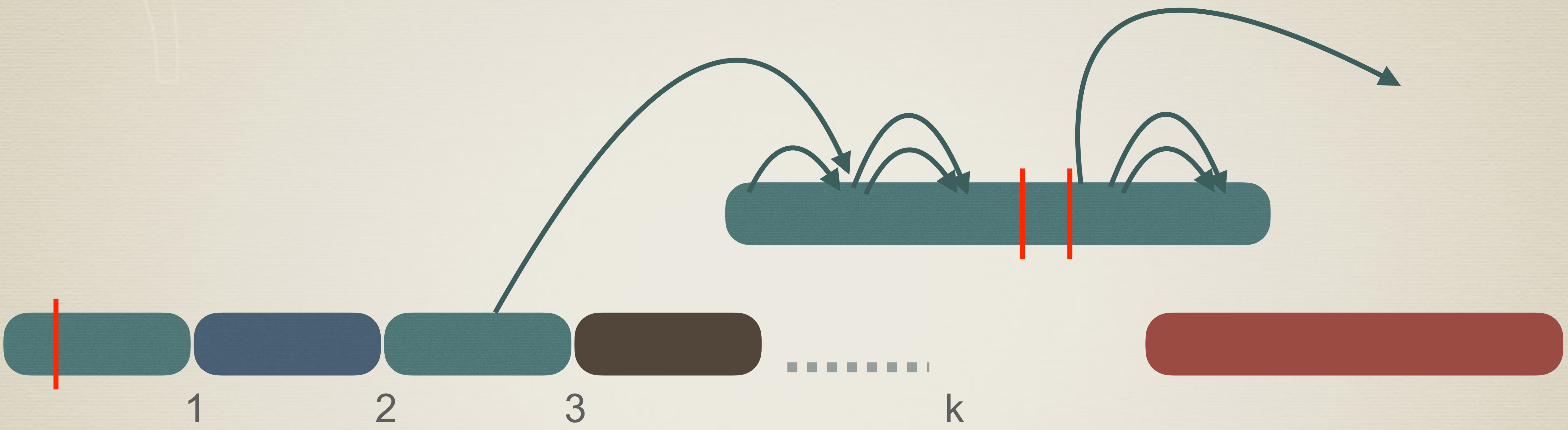
* Between any call and the corresponding return there are at most k context-switches



Cut this call-return pair. Remove.

Bounded Scope Behaviours

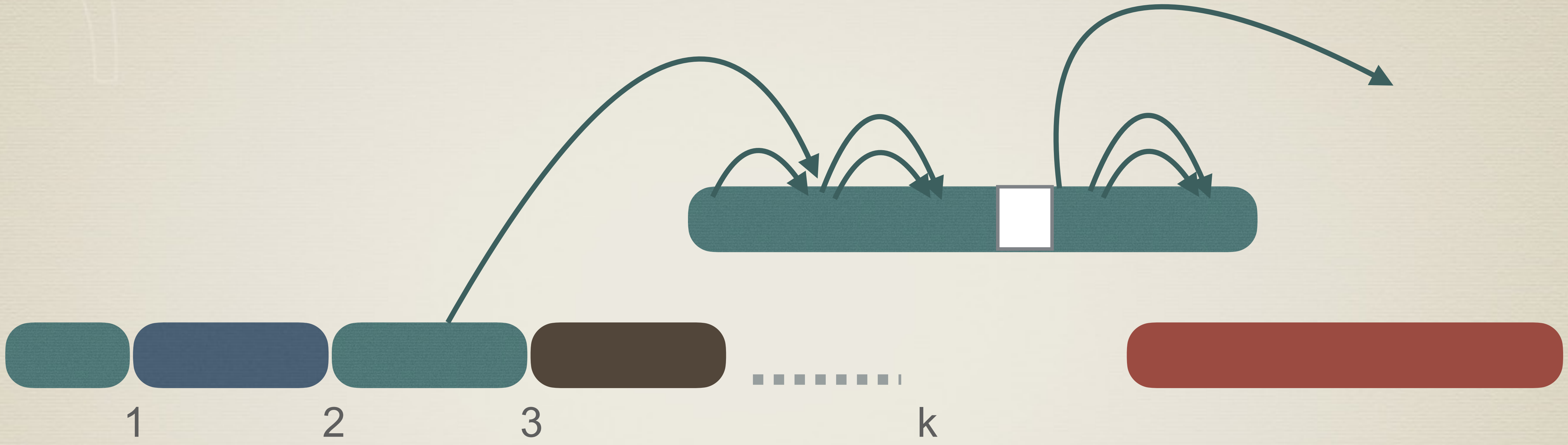
- Between any call and the corresponding return there are at most k context-switches



Cut this call-return pair. Remove.

Bounded Scope Behaviours

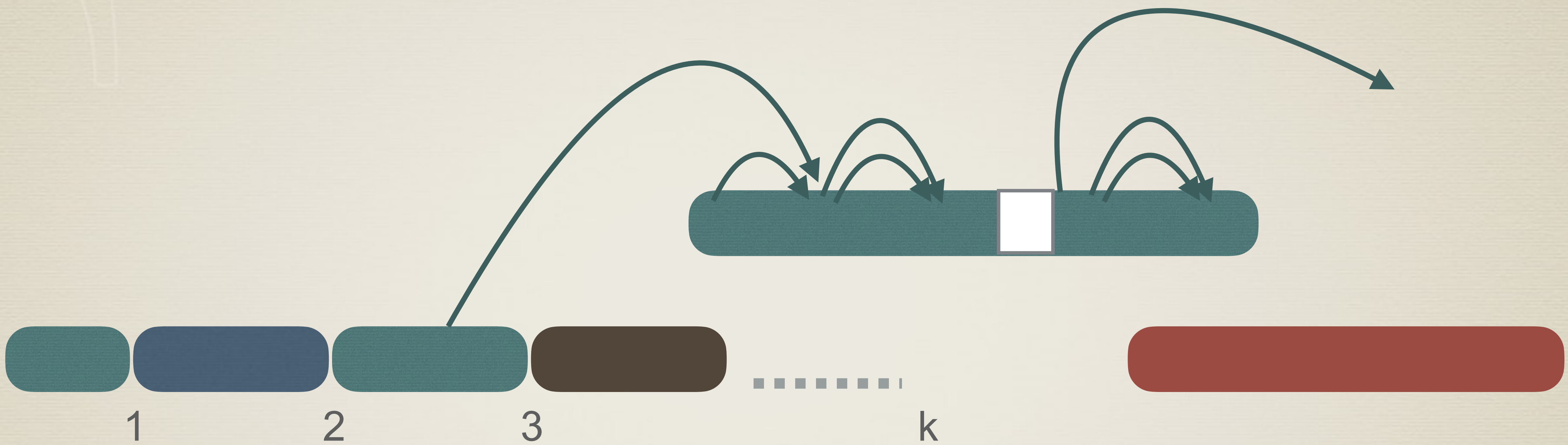
- Between any call and the corresponding return there are at most k context-switches



The called context is left with a hole.

Bounded Scope Behaviours

* Between any call and the corresponding return there are at most k context-switches

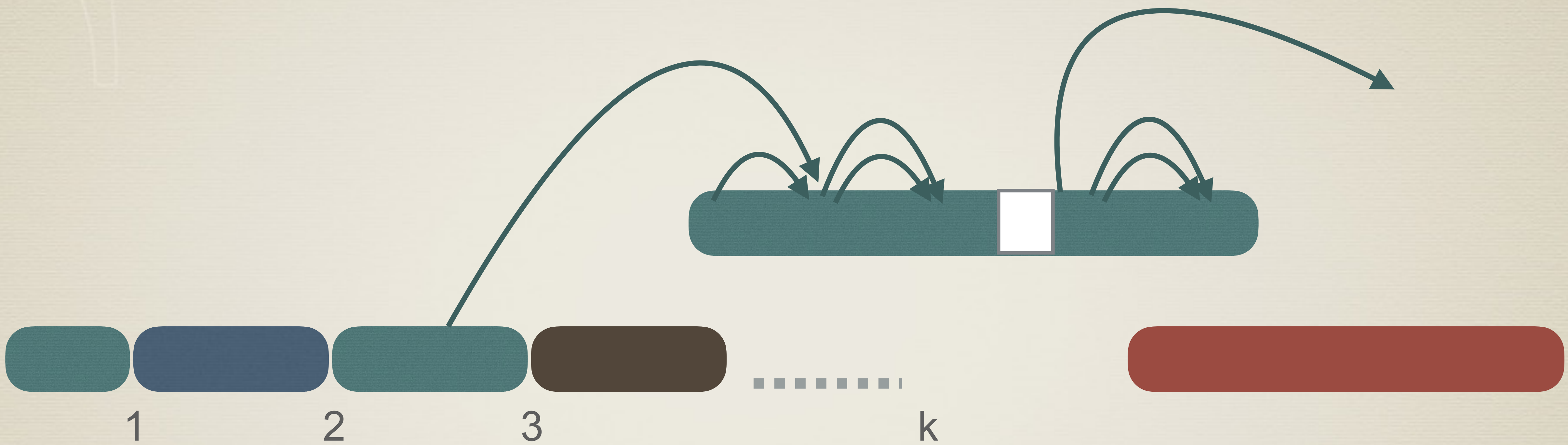


No green nesting edge crosses the hole

The called context is left with a hole.

Bounded Scope Behaviours

* Between any call and the corresponding return there are at most k context-switches



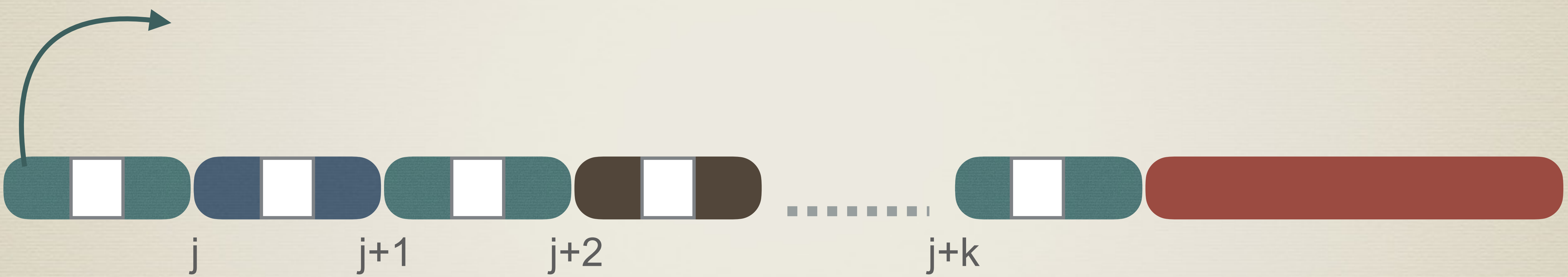
The called context is left with a hole.

Bounded Scope Behaviours

Maintaining invariantly that

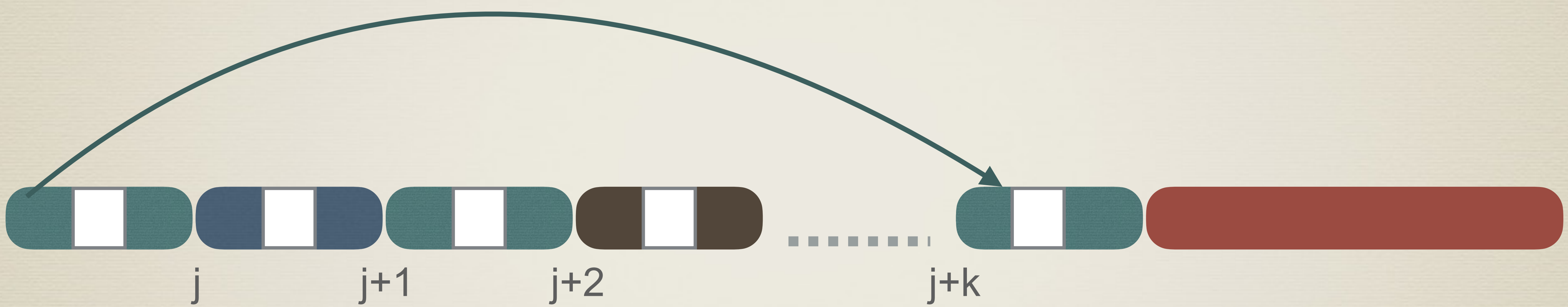
* we have at most 1 hole each in the first k contexts
* no green edge crosses a hole in a green context ...

we will show that we can remove one more edge.



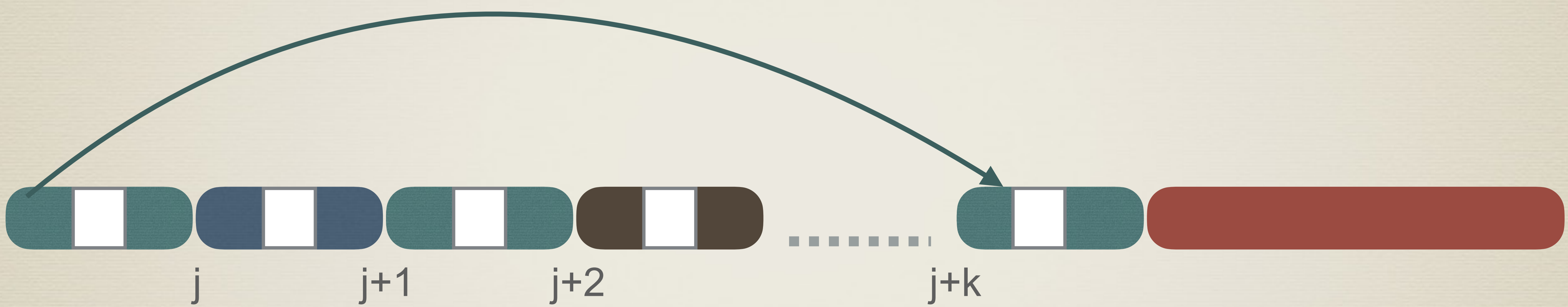
Bounded Scope Behaviours

* we have at most 1 hole each in the first k contexts
* no green edge crosses a hole in a green context



Bounded Scope Behaviours

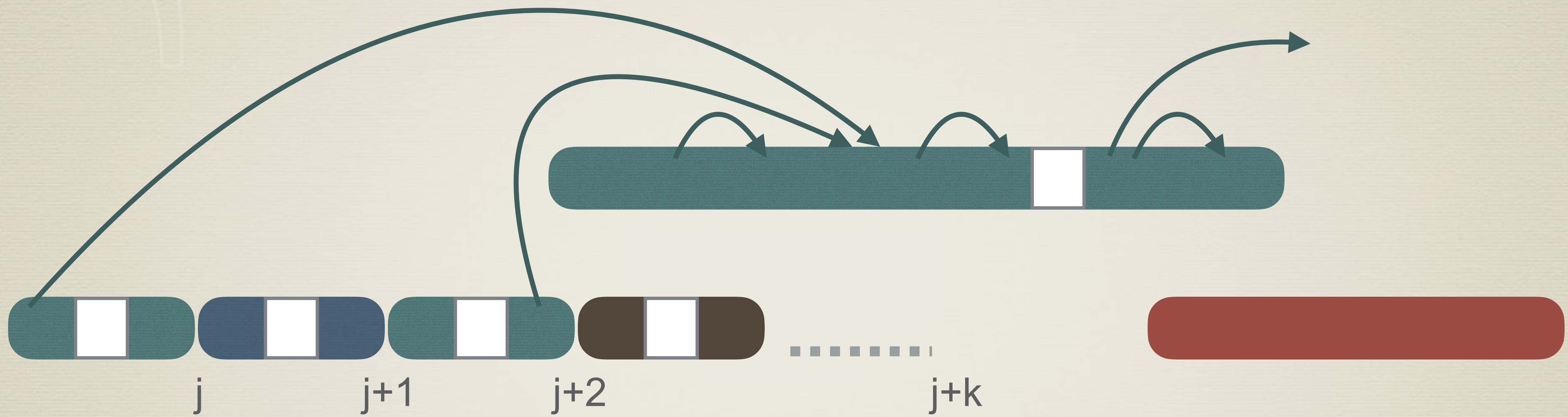
* we have at most 1 hole each in the first k contexts
* no green edge crosses a hole in a green context



Target is before the hole (if any) in that context.

Bounded Scope Behaviours

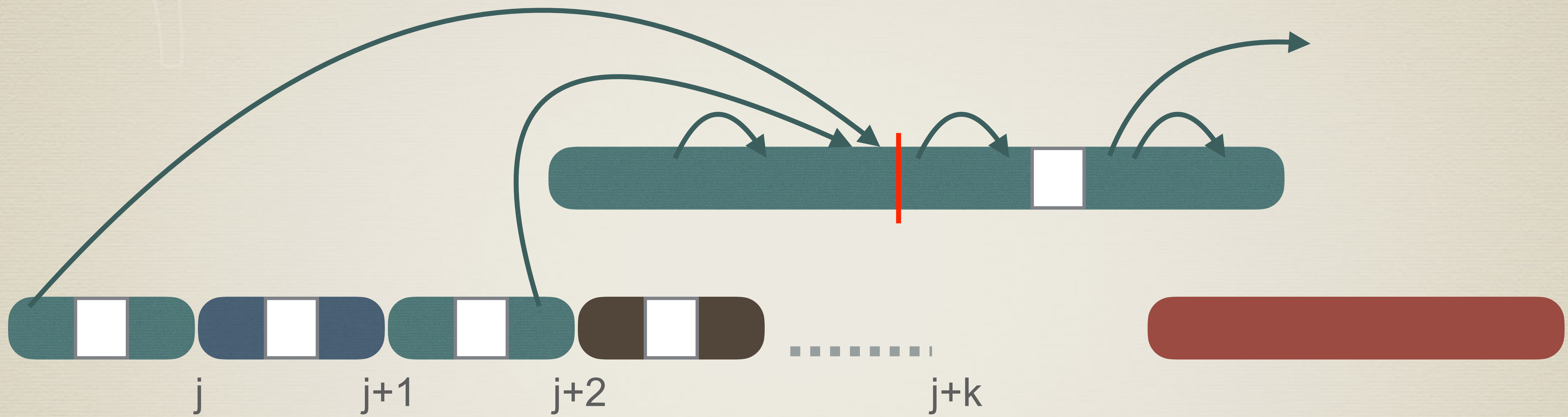
* we have at most 1 hole each in the first k contexts
* no green edge crosses a hole in a green context



Target is before the hole. Between the target and hole is a nested word.

Bounded Scope Behaviours

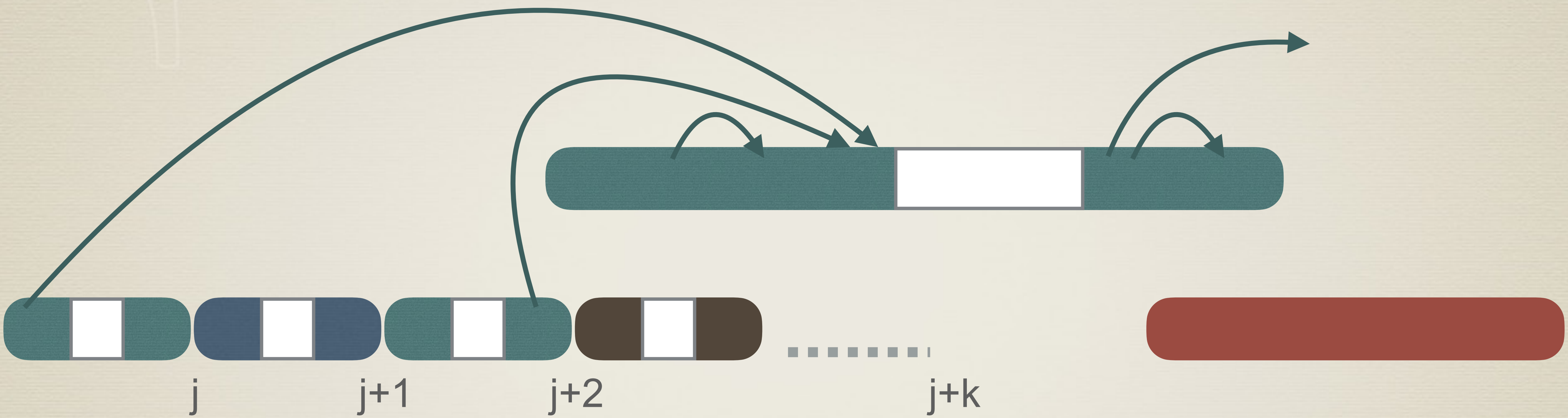
* we have at most 1 hole each in the first k contexts
* no green edge crosses a hole in a green context



Cut and remove nested word to expand the hole

Bounded Scope Behaviours

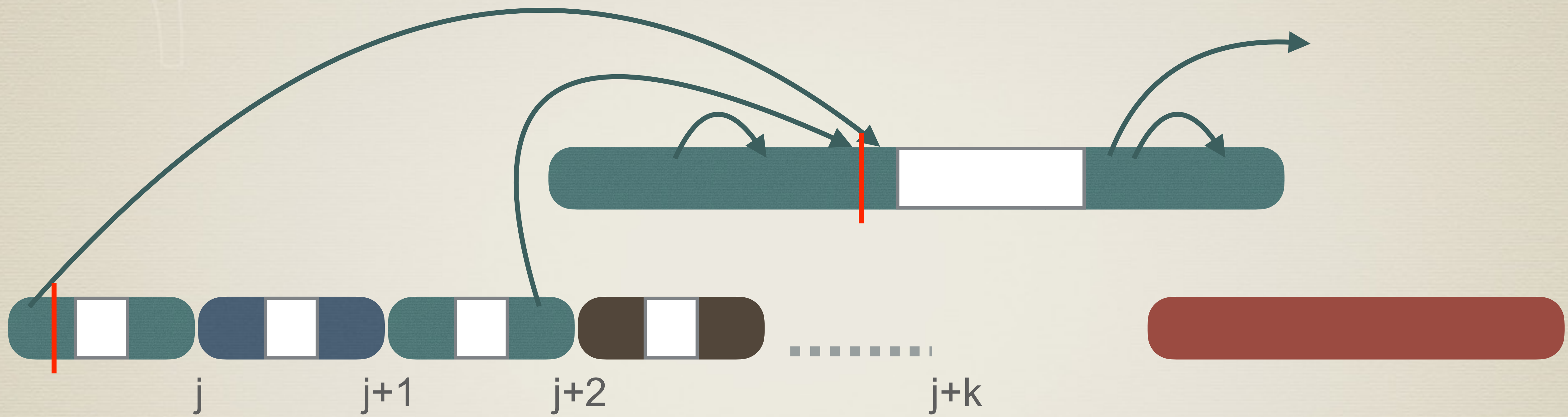
* we have at most 1 hole each in the first k contexts
* no green edge crosses a hole in a green context



Cut and remove nested word to expand the hole

Bounded Scope Behaviours

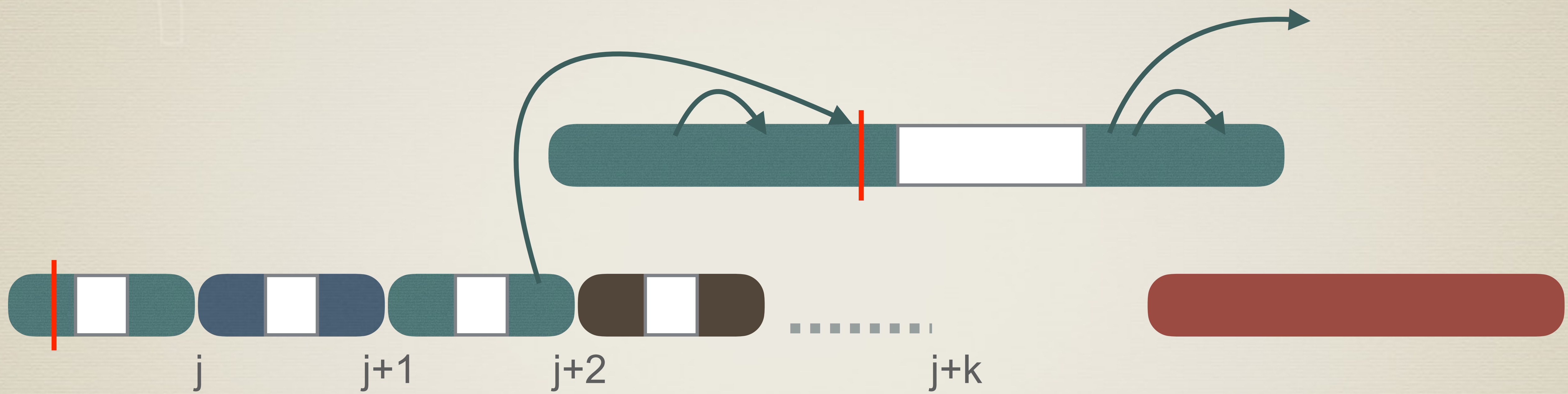
* we have at most 1 hole each in the first k contexts
* no green edge crosses a hole in a green context



If there is a hole: Cut and remove the nesting edge maintaining the invariant

Bounded Scope Behaviours

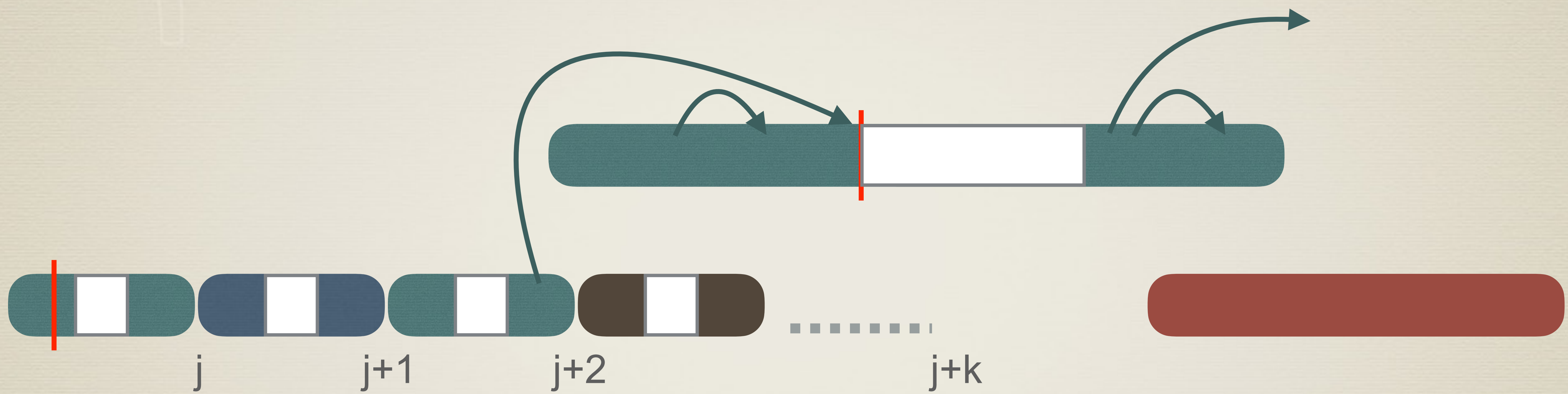
* we have at most 1 hole each in the first k contexts
* no green edge crosses a hole in a green context



If there is a hole: Cut and remove the nesting edge maintaining the invariant

Bounded Scope Behaviours

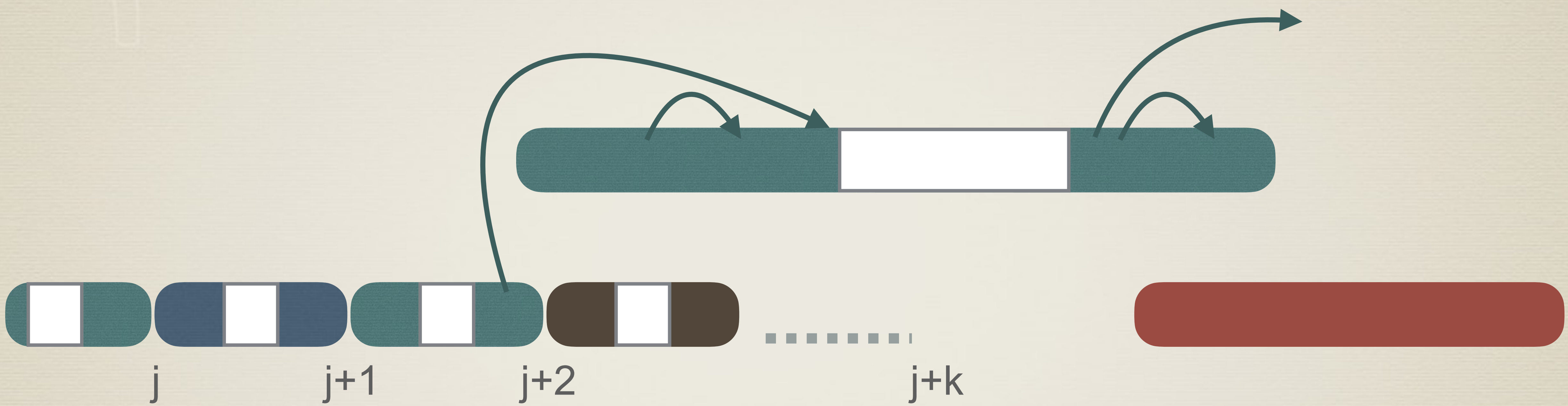
* we have at most 1 hole each in the first k contexts
* no green edge crosses a hole in a green context



If there is a hole: Cut and remove the nesting edge maintaining the invariant

Bounded Scope Behaviours

* we have at most 1 hole each in the first k contexts
* no green edge crosses a hole in a green context



If there is a hole: Cut and remove the nesting edge maintaining the invariant

Existentially Bounded MSCs

* The class of MSCs that are existentially k bounded have split-width k+1
* Existentially k-bounded MSCs form an MSO definable class (GenKusMus07)

Theorem: (GenKusMus07)

1. MSO theory of existentially k bounded MSCs is decidable.
2. MSO model checking for Message Passing Automata w.r.t existentially k-bounded behaviours is decidable.

Bounded Scope Behaviours

* The class of MNWs with scope bound k has split-width k+1
* k bounded-scope MNWs is MSO expressible.

Theorem:

1. MSO theory of k scope-bound MNWs is decidable.
2. MSO model checking for MPDS w.r.t k scope-bounded behaviours is decidable.

Split-width: parametrized verification

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	

Conclusion

* Use graphs to reason about behaviors of systems

  distributed or sequential

* Exploit theory of graph decompositions
  * Tree Interpretations
  * Use “algebraic decompositions”
  * Tailor algebra to the setting to find natural proofs for boundedness.

* Split-width: convenient decomposition technique

  As powerful as tree-width or clique-width for CBMs and yields optimal algorithms

Conclusions…

- Extensions
  - Parameterized systems (size, topology)
    GasFor'16, FOSSACS'16
  - Timed systems
    AksGasKri'16, CONCUR'16
  - Higher-order PDA level 2
    AisGasSaivasan'16
  - Dynamic creation of processes
  - Read from many
  - Infinite behaviors
  - …

Main Sources

P. Madhusudan and G. Parlato

* Tree-width of Auxiliary Storage. In POPL 2011.

C. Aiswarya, P. Gastin, ..

* MSO decidability of multi-pushdown systems via split-width. In CONCUR 2012.
* Verifying Communicating Multi-pushdown Systems via Split-width. In ATVA 2014.

Aiswarya's PhD Thesis

* Many more classes with bounded split-width
* Many more results

C. Aiswarya, P. Gastin

* Reasoning About Distributed Systems: WYSIWYG. In FSTTCS 2014

B. Bollig, P. Gastin

* MPRI Lecture Notes on Non-sequential Theory of Distributed Systems

Main Sources

P. Madhusudan and G. Parlato
* Tree-width of Auxiliary Storage. In POPL 2011.

C. Aiswarya, P. Gastin, ..
* MSO decidability of multi-pushdown systems via split-width. In CONCUR 2012.
* Verifying Communicating Multi-pushdown Systems via Split-width. In ATVA 2014.

Aiswarya's PhD Thesis
* Many more classes with bounded split-width
* Many more results

C. Aiswarya, P. Gastin
* Reasoning About Distributed Systems:

B. Bollig, P. Gastin
* MPRI Lecture Notes on Non-sequenti ... Distributed Systems

THANK YOU