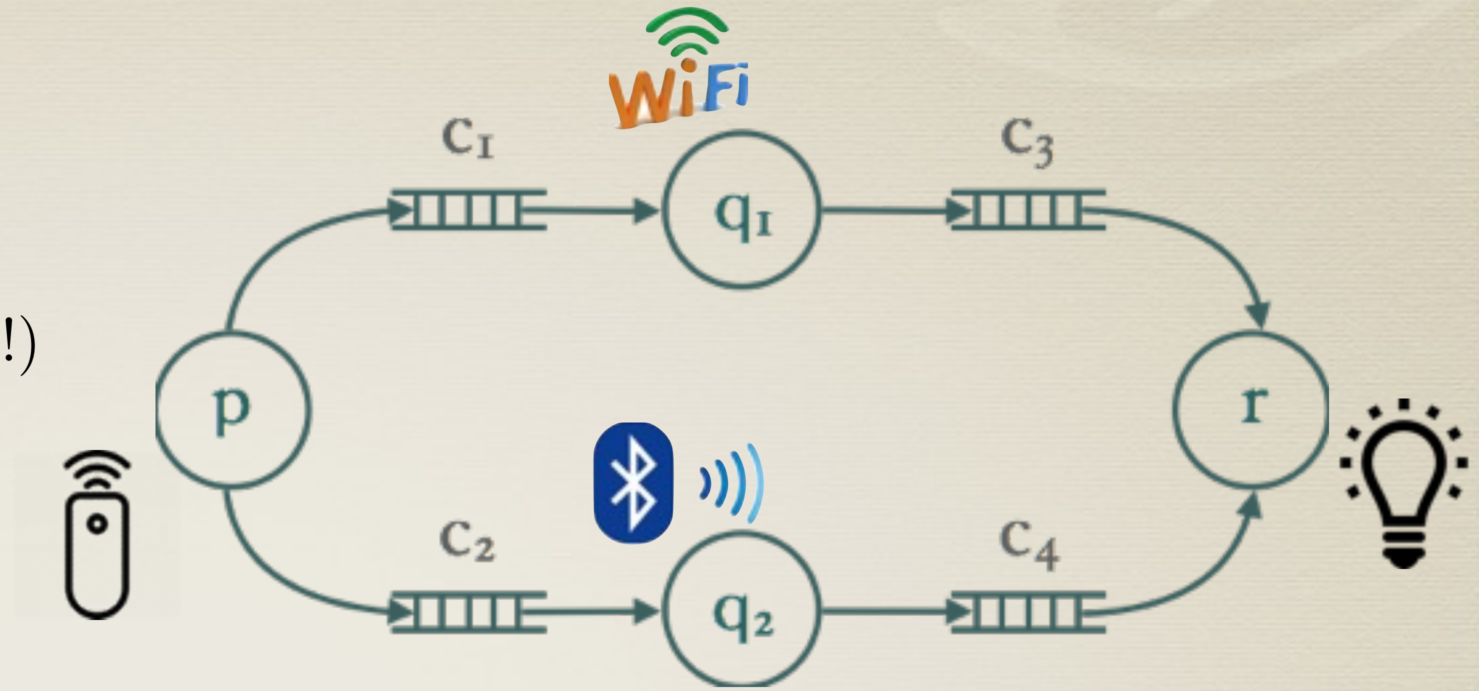
Reasoning about Distributed Systems: WYSIWYG

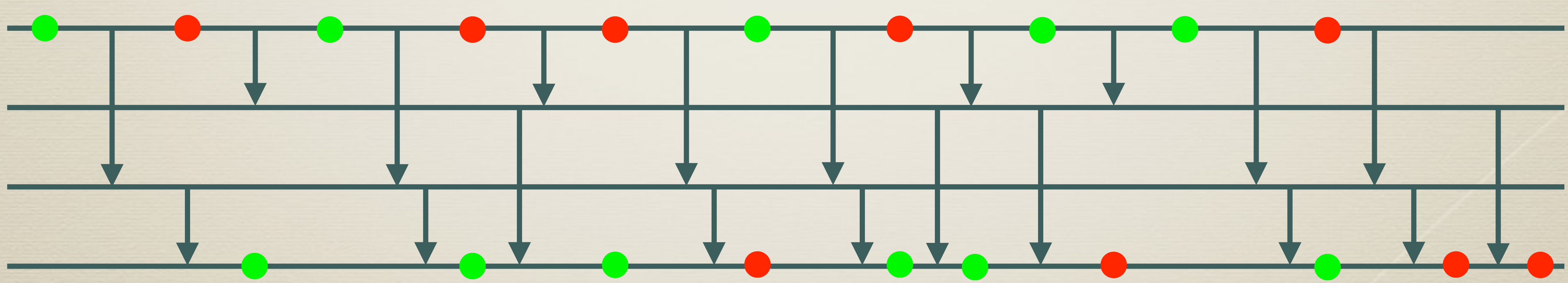
Paul Gastin
LSV, ENS Cachan, France
C. Aiswarya
CMI, Chennai, India

Proceedings of FSTTCS'14 (Invited talk)

Introduction

$(p, \mathsf{on})(p, c_2!)(p, \mathsf{off})(q_2, c_2?)(p, c_1!)(q_1, c_1?)$
$(q_2, c_4!)(p, \mathsf{on})(p, c_2!)(p, \mathsf{off})(r, c_4?)(r, \mathsf{on})$
$(q_1, c_3!)(p, c_1!)(q_1, c_1?)(q_1, c_3!)(q_2, c_2?)(q_2, c_4!)$
$(r, c_4?)(r, \mathsf{on})(r, c_3?)(r, \mathsf{off}) \cdots$

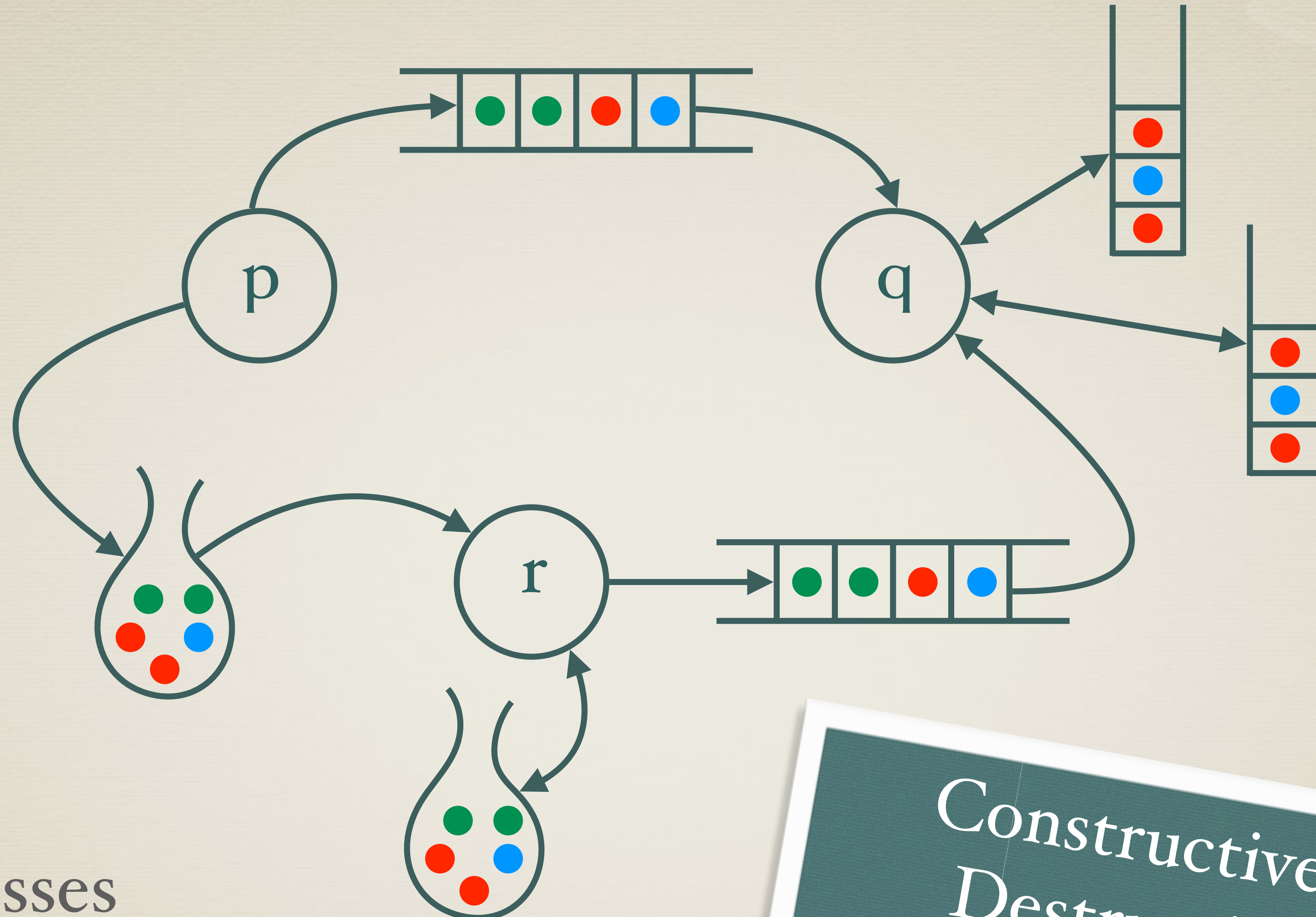




Outline

* Concurrent Processes with Data Structures
* Behaviors as Graphs
* Specifications
* Verification with Graphs and under-approximations
* Split-width and tree interpretation
* Conclusion

System: Concurrent Processes with Data-Structures

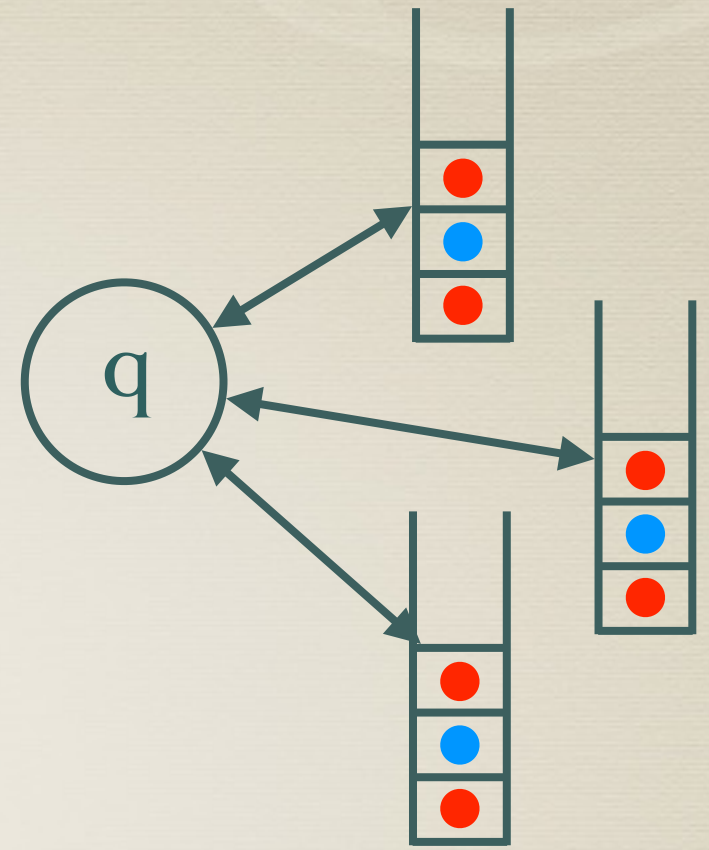


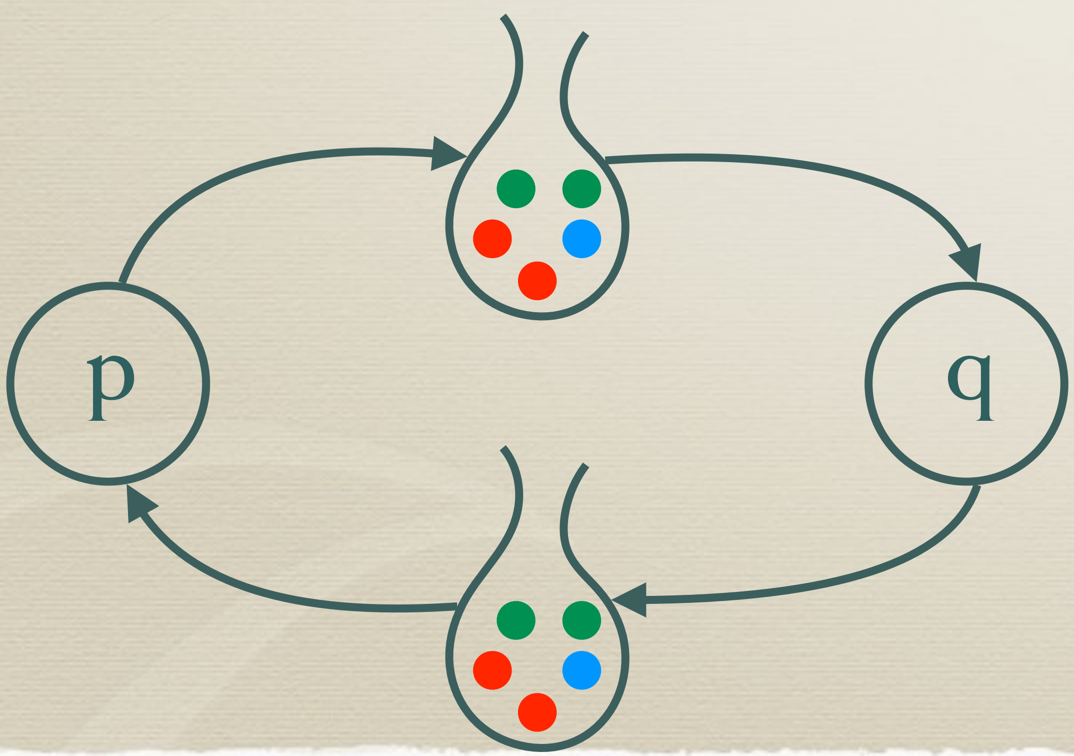
Constructive writes
Destructive reads

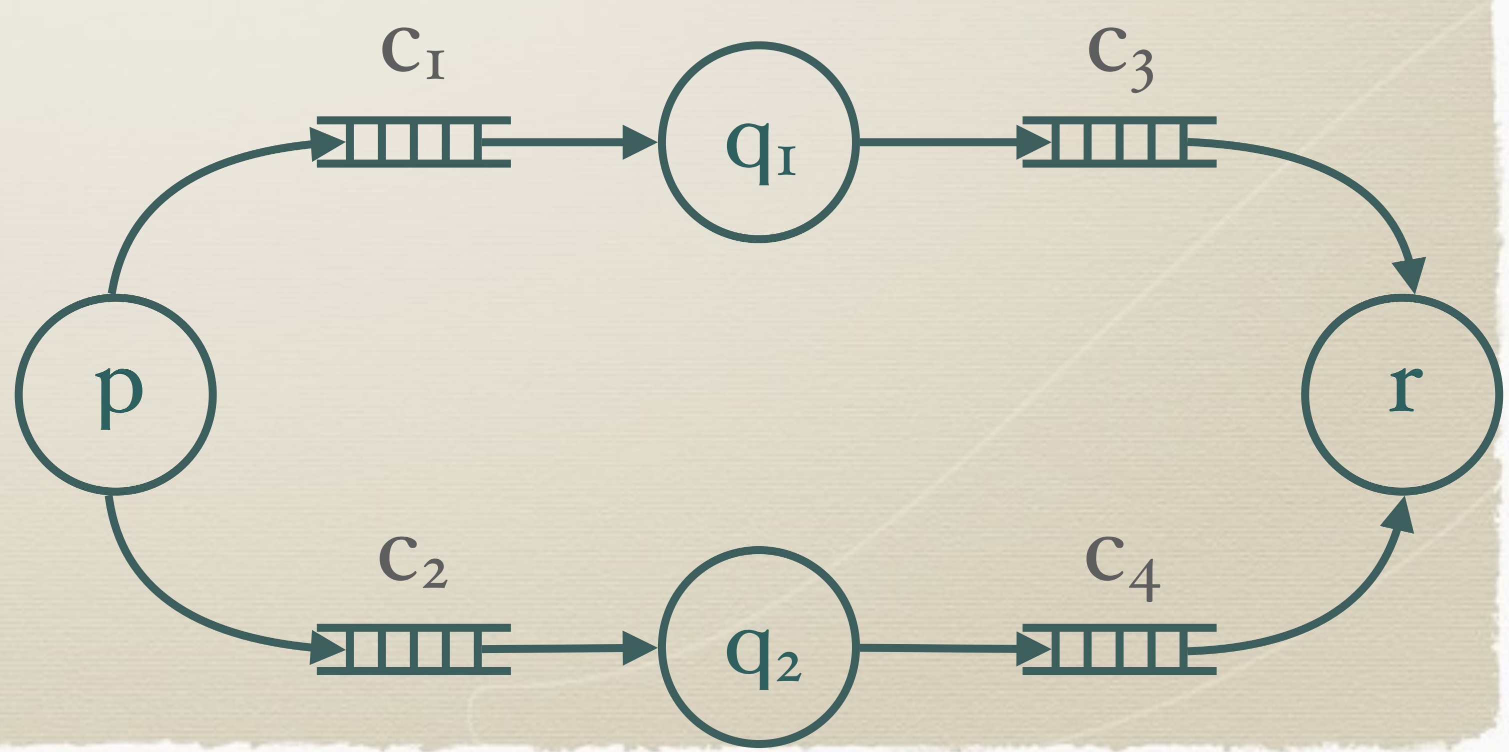
- Processes
- Data structures
  - Stacks: recursive programs, multithreaded
  - Queues: communication (FIFO)
  - Bags: communication (unordered)

Architectures: Special cases

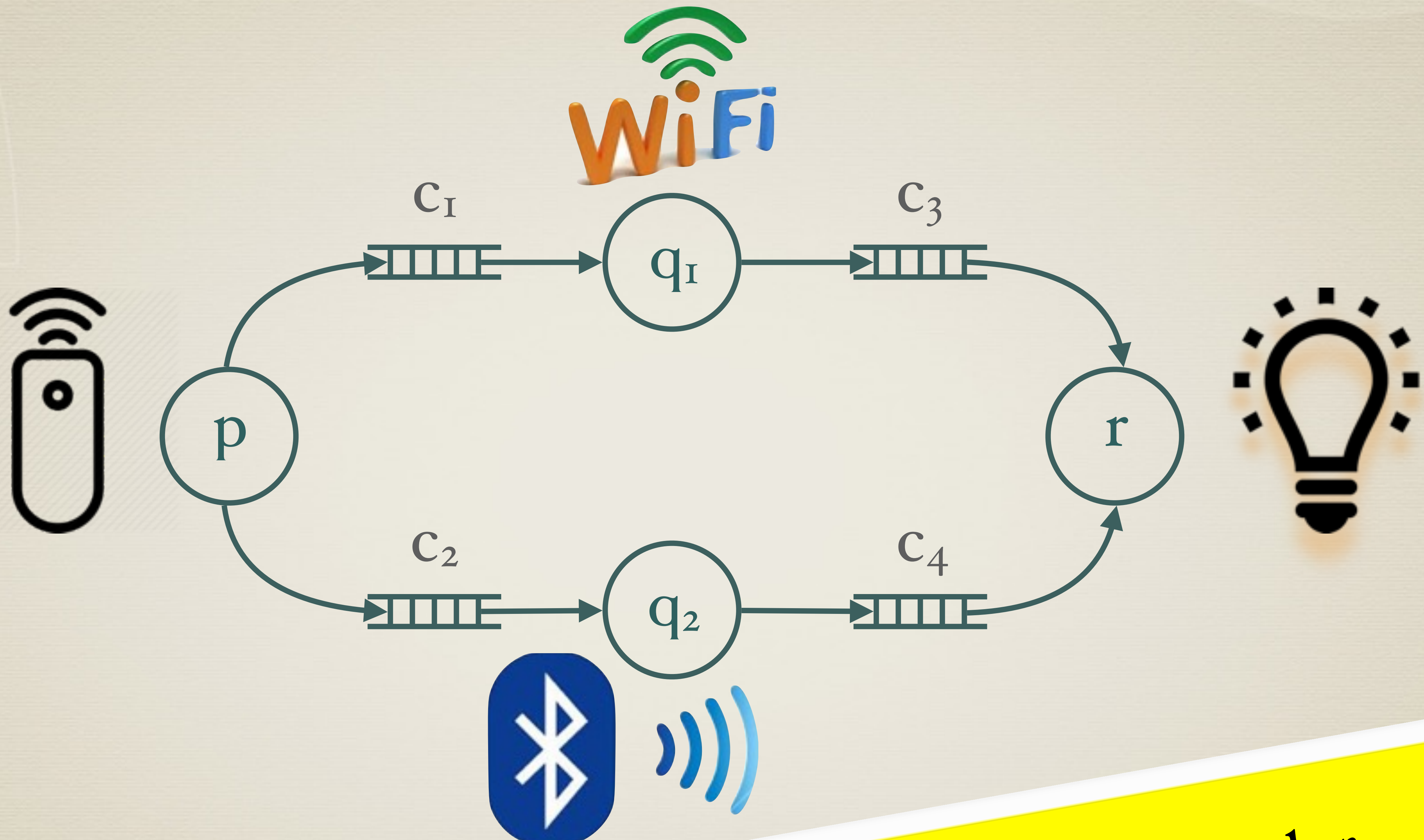
- PDA: Pushdown automata
  Recursive programs
- MPDA: Multi-pushdown automata
  Multi-threaded recursive programs
- MPA: Message passing automata
  Communicating finite state machines
- PN: Petri Nets : Only bags





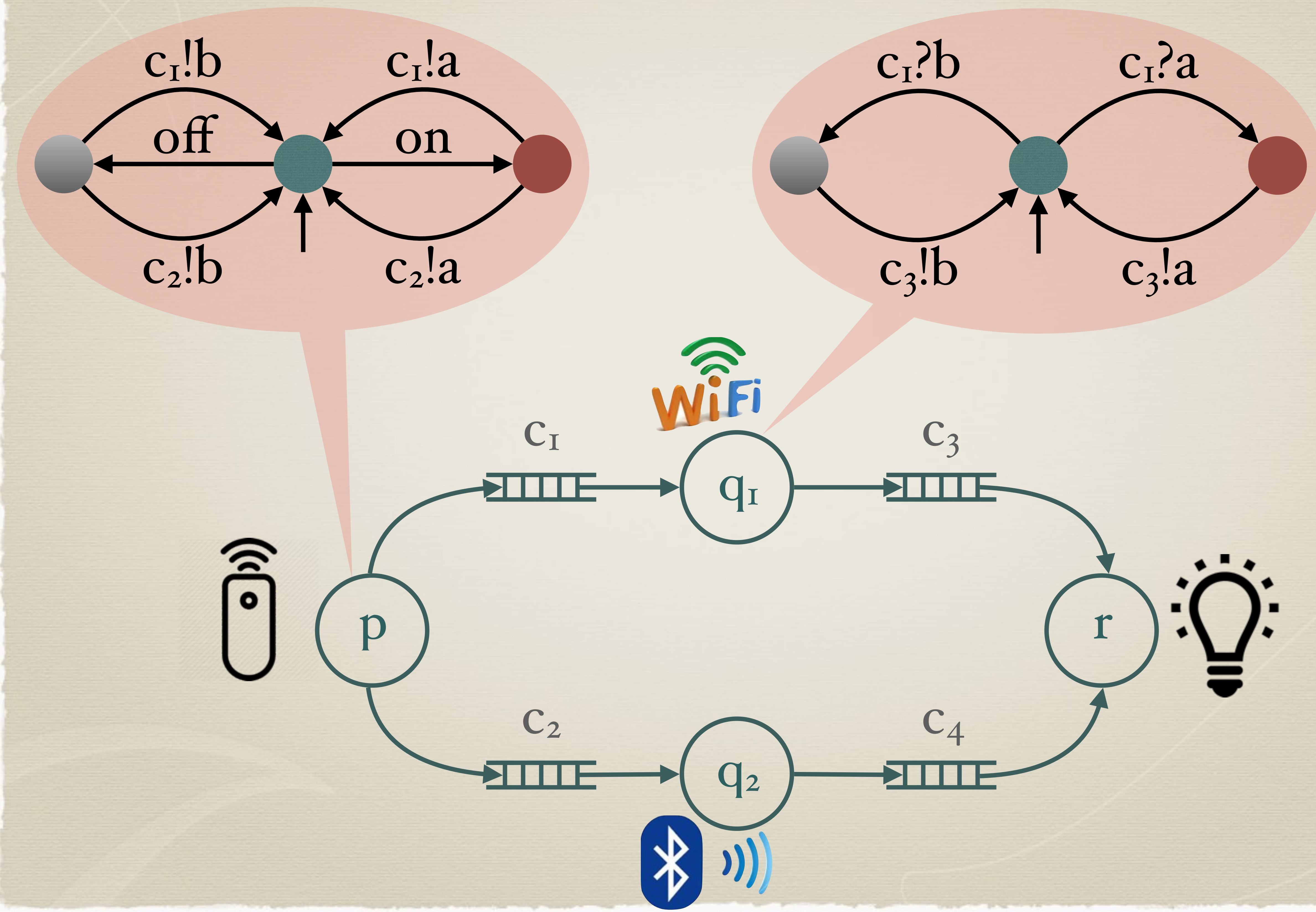


Remote on-off via 2 channels

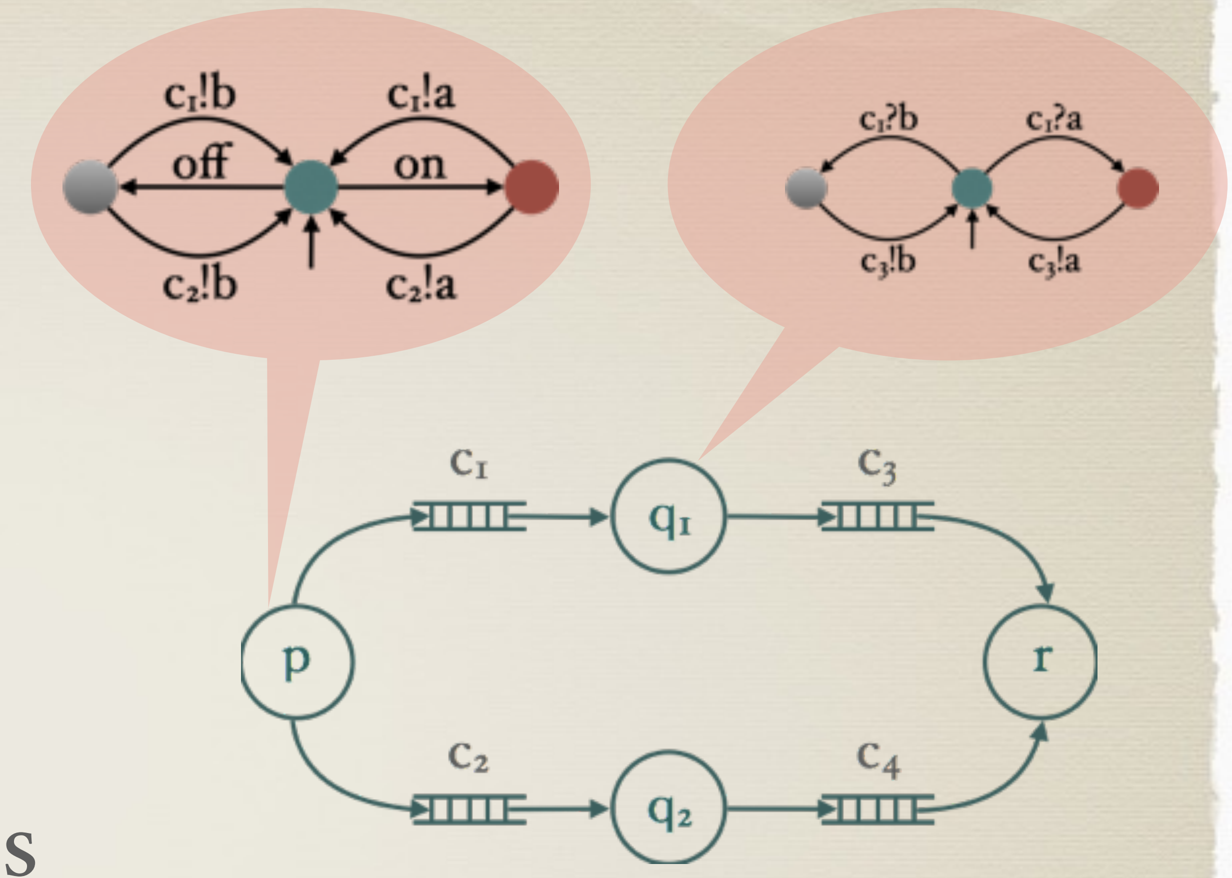


Obey the latest order

System: Architecture + Boolean Programs



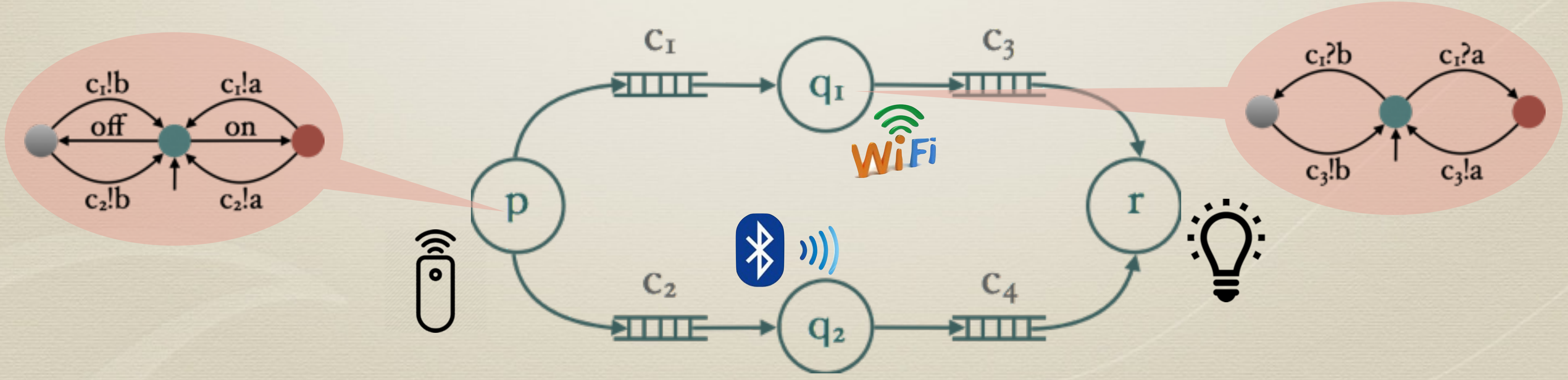
Operational semantics



* Transition system TS
  * Configurations (infinite)
    * local states of processes
    * contents of data structures
  * Transitions
    * Induced by the boolean programs
* Linear traces: abstractions of runs of TS

Linear Traces

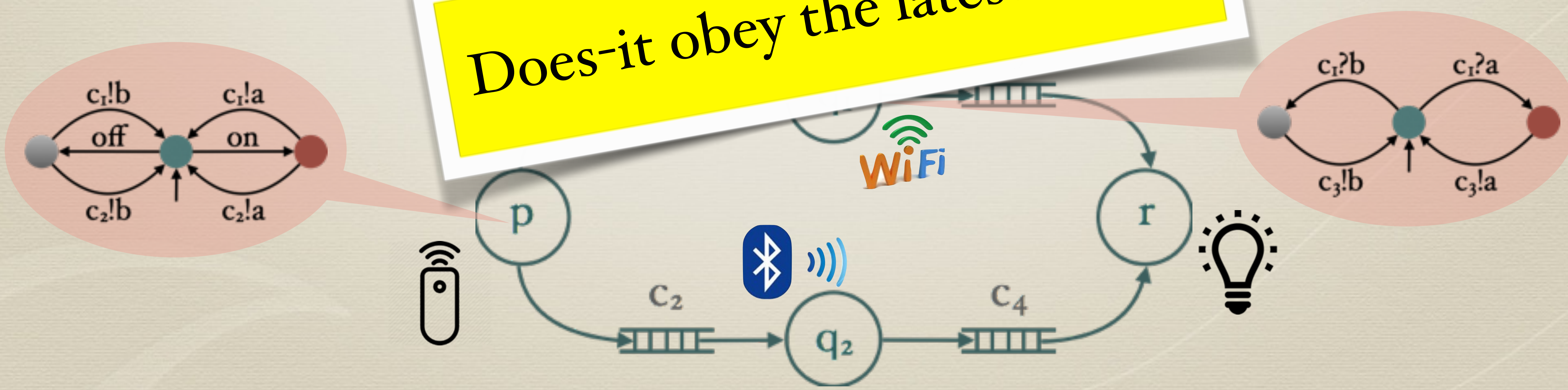
$$(p, \mathsf{on})(p, c_2!)(p, \mathsf{off})(q_2, c_2?)(p, c_1!)(q_1, c_1?)$$
$$(q_2, c_4!)(p, \mathsf{on})(p, c_2!)(p, \mathsf{off})(r, c_4?)(r, \mathsf{on})$$
$$(q_1, c_3!)(p, c_1!)(q_1, c_1?)(q_1, c_3!)(q_2, c_2?)(q_2, c_4!)$$
$$(r, c_4?)(r, \mathsf{on})(r, c_3?)(r, \mathsf{off}) \cdots$$



Linear Traces

$$(p,\mathsf{on})(p,c_2!)(p,\mathsf{off})(q_2,c_2?)(p,c_1!)(q_1,c_1?)$$
$$(q_2,c_4!)(p,\mathsf{on})(p,c_2!)(p,\mathsf{off})(r,c_4?)(r,\mathsf{on})$$
$$(q_1,c_3!)(p,c_1!)(q_1,c_1?)(q_1,c_3!)(q_2,c_2?)(q_2,c_4!)$$
$$(r,c_4?)(r,\mathsf{on})(r,c_3?)(r,\mathsf{off})\cdots$$

Does-it obey the latest order?



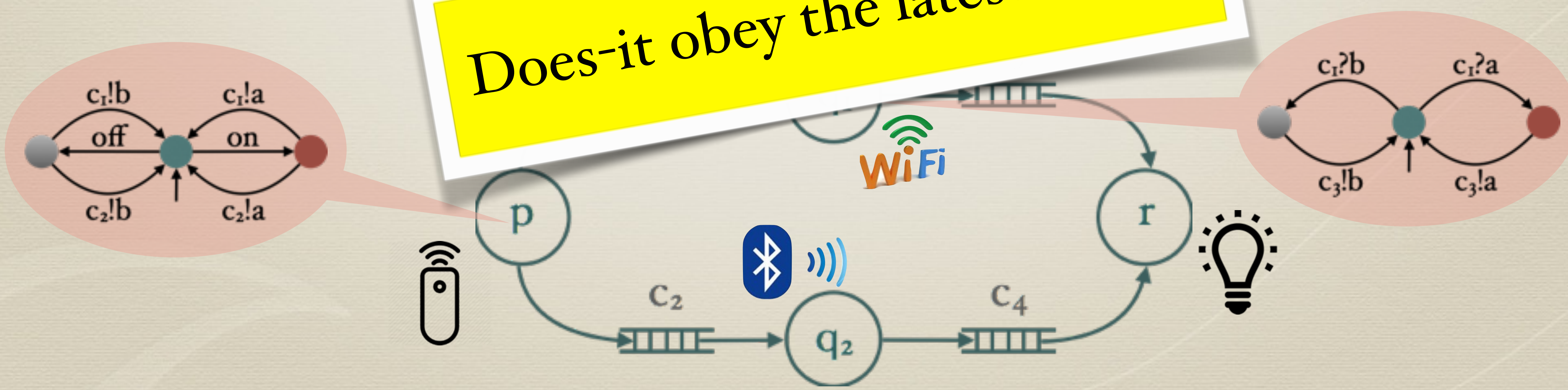
Outline

- ☑ Concurrent Processes with Data Structures
- Behaviors as Graphs
- Specifications
- Verification with Graphs and under-approximations
- Split-width and tree interpretation
- Conclusion

Linear Traces

$(p, \mathsf{on})(p, c_2!)(p, \mathsf{off})(q_2, c_2?)(p, c_1!)(q_1, c_1?)$
$(q_2, c_4!)(p, \mathsf{on})(p, c_2!)(p, \mathsf{off})(r, c_4?)(r, \mathsf{on})$
$(q_1, c_3!)(p, c_1!)(q_1, c_1?)(q_1, c_3!)(q_2, c_2?)(q_2, c_4!)$
$(r, c_4?)(r, \mathsf{on})(r, c_3?)(r, \mathsf{off}) \cdots$

Does-it obey the latest order?



Linear Traces

WYSIWYG:
Make visible what is important

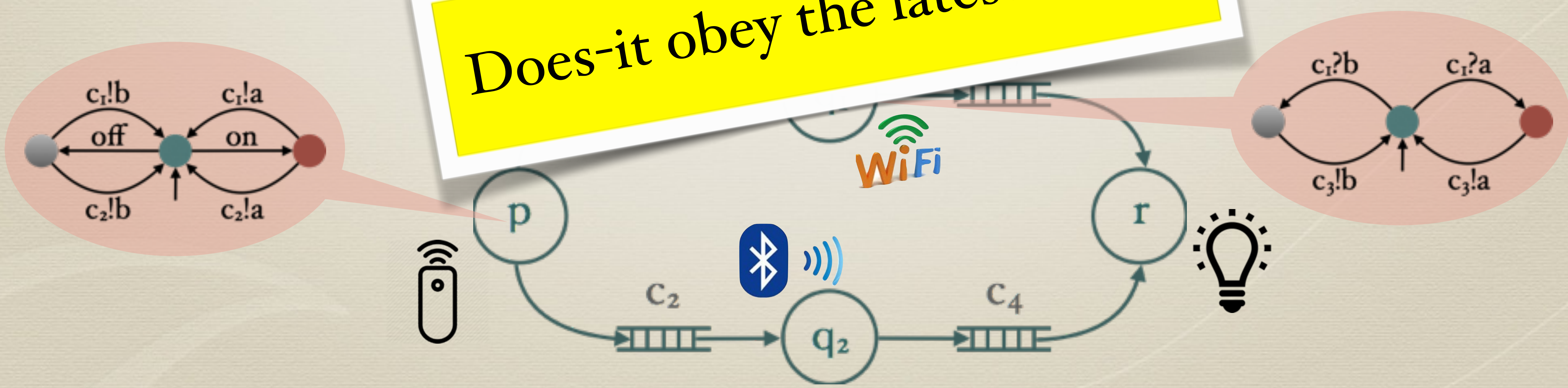
$(p, \mathsf{on})(p, c_2!)(p$ … $)(p, c_1!)(q_1, c_1?)$

$(q_2, c_4!)(p, \mathsf{on})(p, c_2!)(p, \mathsf{off})(r, c_4?)(r, \mathsf{on})$

$(q_1, c_3!)(p, c_1!)(q_1, c_1?)(q_1, c_3!)(q_2, c_2?)(q_2, c_4!)$

$(r, c_4?)(r, \mathsf{on})(r, c_3?)(r, \mathsf{off}) \cdots$

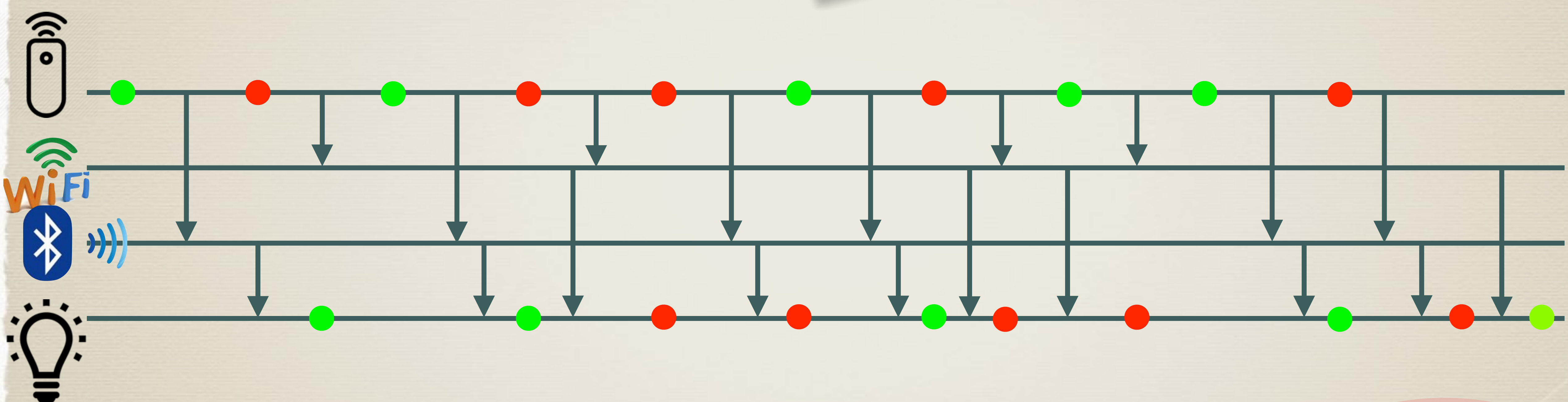
Does-it obey the latest order?

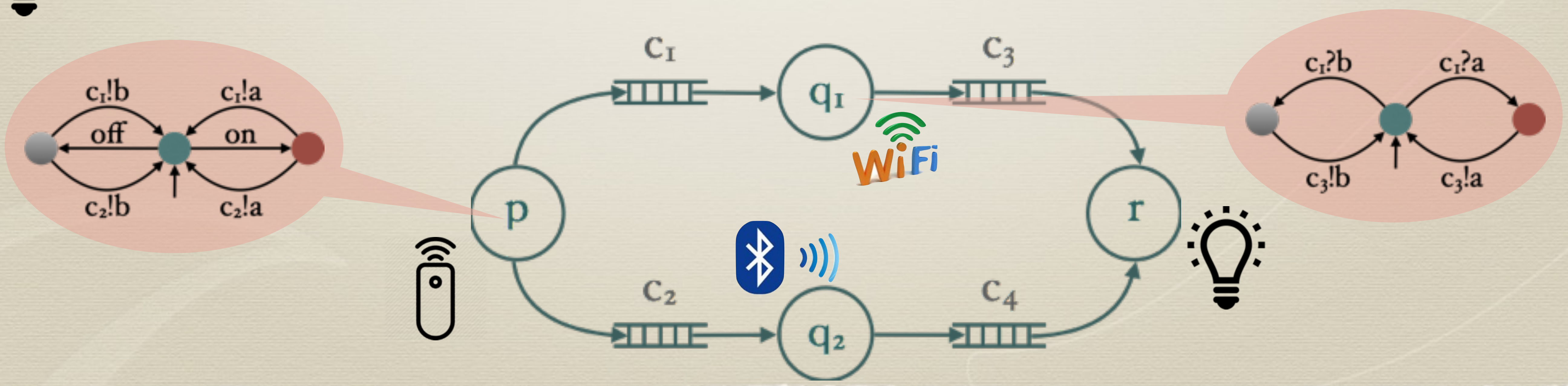


Linear Traces vs. Graphs

$(p,\mathsf{on})(p,c_2!)(p,\mathsf{off})(q_2,c_2?)(p,c_1!)(q_1,c_1?)$
$(q_2,c_4!)(p,\mathsf{on})(p,c_2!)(p,\mathsf{off})(r,c_4?)(r,\mathsf{on})$
$(q_1,c_3!)(p,c_1!)(q_1,c_1?)(q_1,c_3!)(q_2,c_2?)(q_2,c_4!)$
$(r,c_4?)(r,\mathsf{on})(r,c_3?)(r,\mathsf{off})\cdots$

Message Sequence Charts
ITU Standard

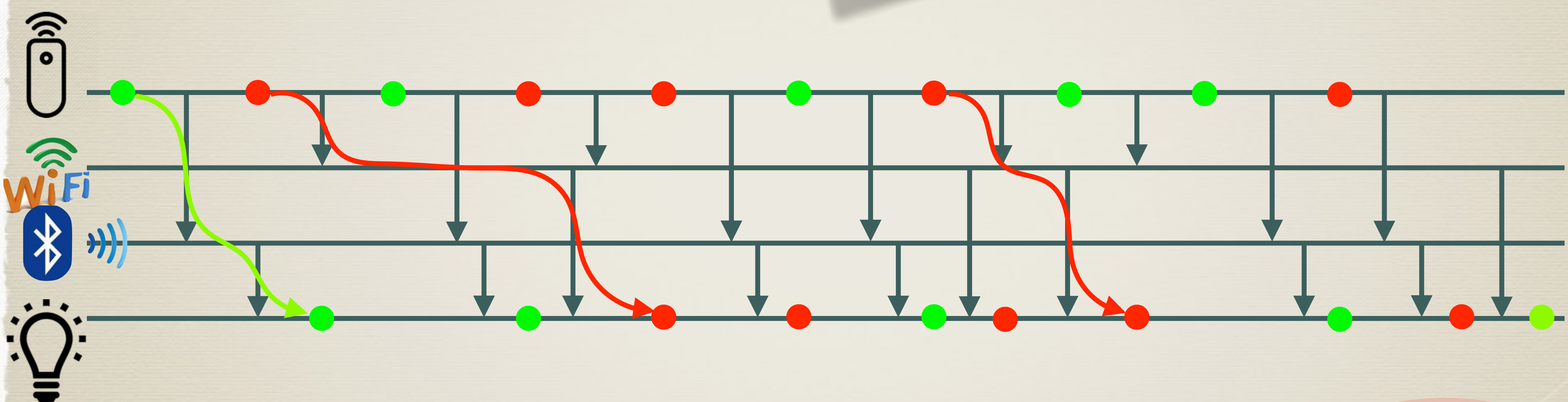


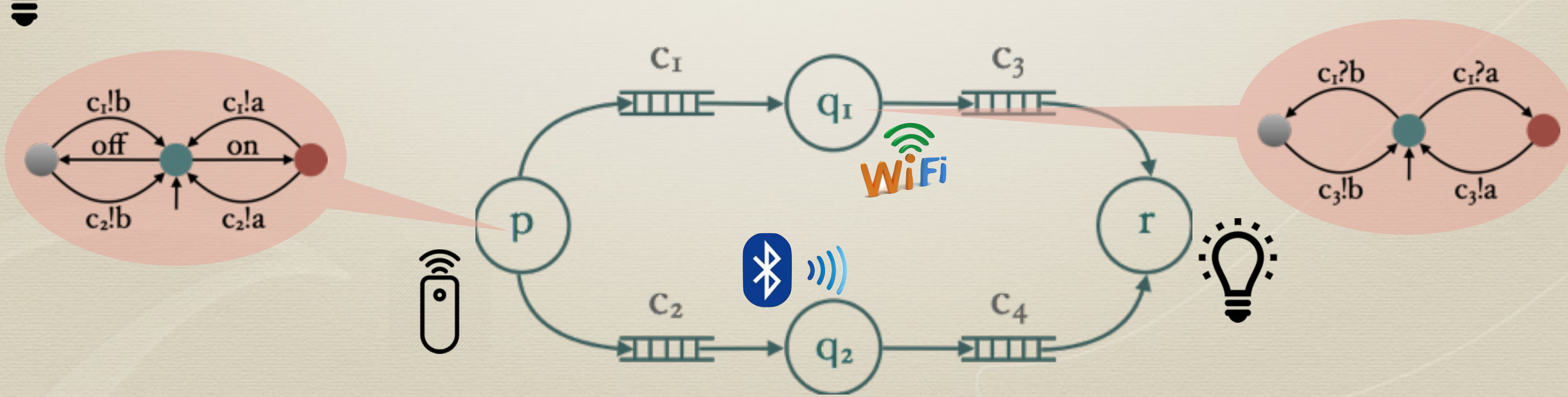


Linear Traces vs. Graphs

$(p, \mathsf{on})(p, c_2!)(p, \mathsf{off})(q_2, c_2?)(p, c_1!)(q_1, c_1?)$
$(q_2, c_4!)(p, \mathsf{on})(p, c_2!)(p, \mathsf{off})(r, c_4?)(r, \mathsf{on})$
$(q_1, c_3!)(p, c_1!)(q_1, c_1?)(q_1, c_3!)(q_2, c_2?)(q_2, c_4!)$
$(r, c_4?)(r, \mathsf{on})(r, c_3?)(r, \mathsf{off}) \cdots$

Does-it obey the latest order?

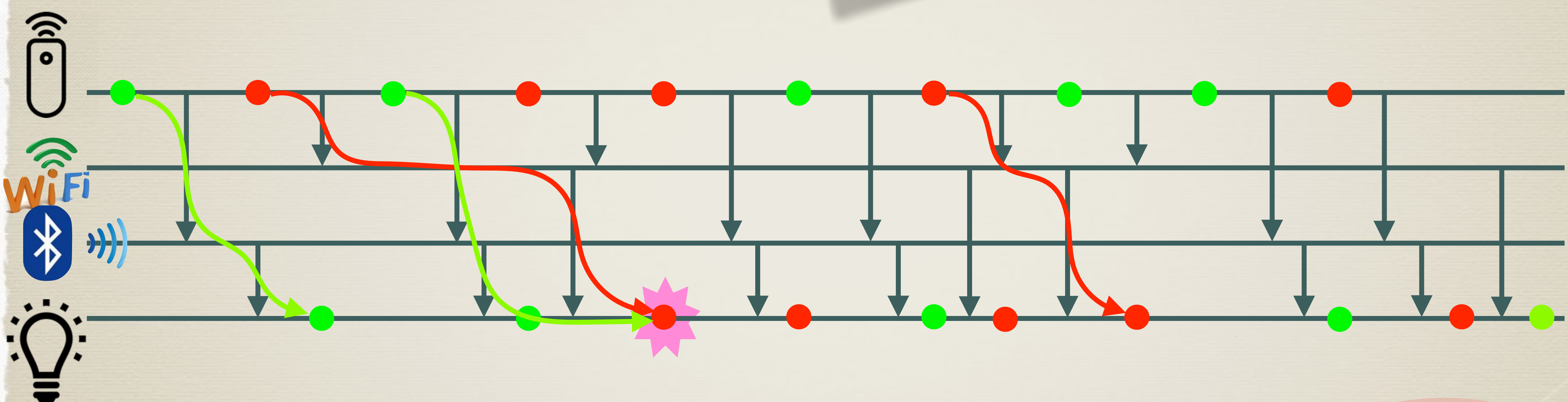


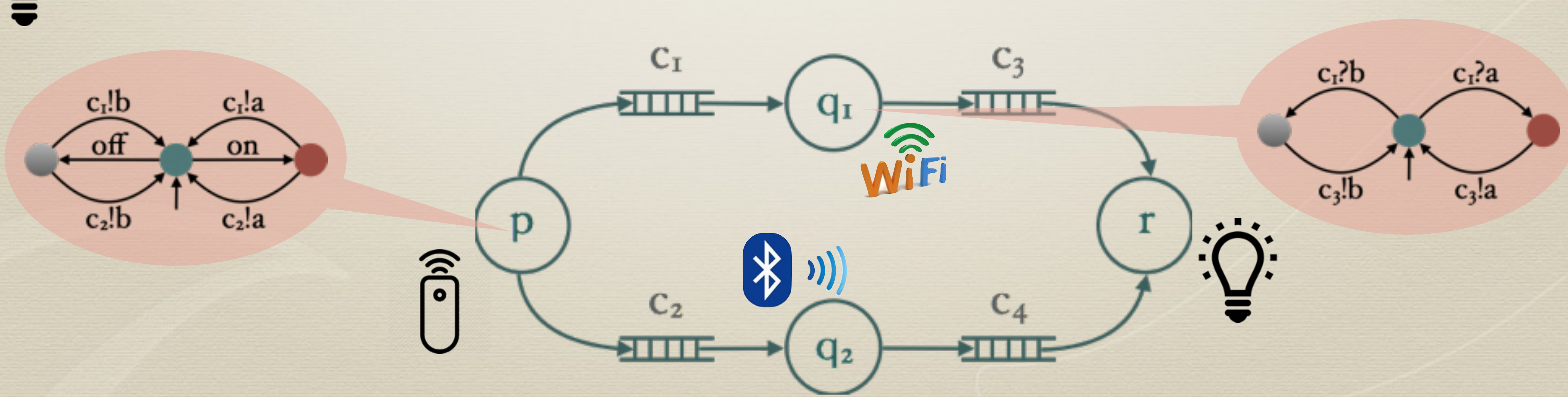


Linear Traces vs. Graphs

$(p,\mathsf{on})(p,c_2!)(p,\mathsf{off})(q_2,c_2?)(p,c_1!)(q_1,c_1?)$
$(q_2,c_4!)(p,\mathsf{on})(p,c_2!)(p,\mathsf{off})(r,c_4?)(r,\mathsf{on})$
$(q_1,c_3!)(p,c_1!)(q_1,c_1?)(q_1,c_3!)(q_2,c_2?)(q_2,c_4!)$
$(r,c_4?)(r,\mathsf{on})(r,c_3?)(r,\mathsf{off})\cdots$

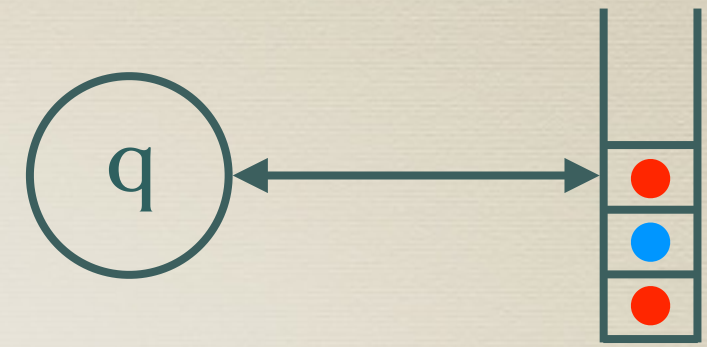
Does-it obey the latest order?





Graphs for Sequential Systems

Answer the correct client for topmost requests

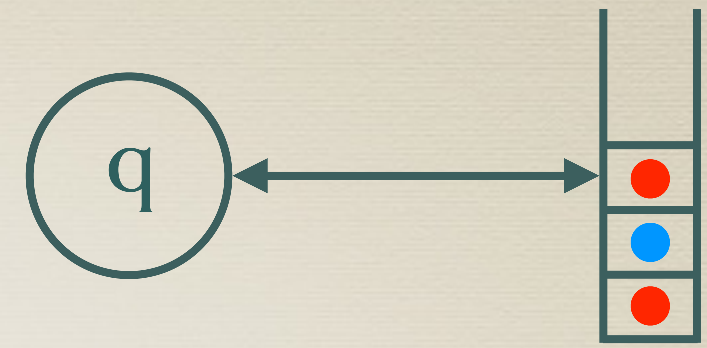


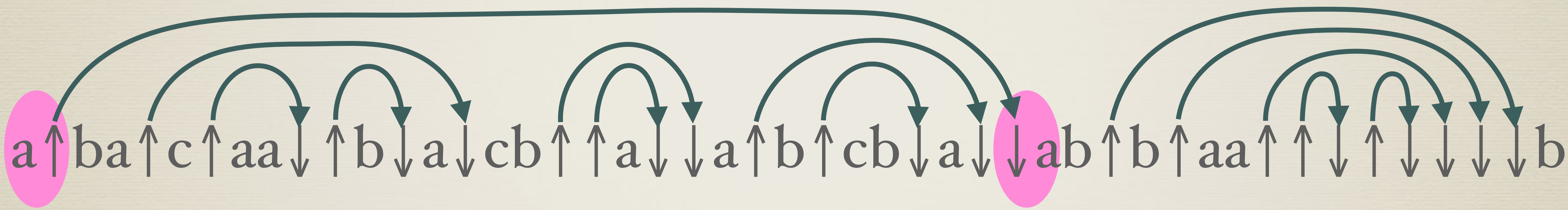
a↑ba↑c↑aa↓↑b↓a↓cb↑↑a↓↓a↑b↑cb↓a↓↓ab↑b↑aa↑↑↓↑↓↓↓↓b

WYSIWYG:
Make visible what is important

Graphs for Sequential Systems

Answer the correct client for topmost requests





WYSIWYG:
Make visible what is important

Nested Words
Alur, Madhusudan, 2009

WYSIWYG

Understanding Behaviors

Linear Traces	Graphs (CBMs)
• Interleaved sequence of events. Interactions are obfuscated and very difficult to recover. • Successor relation not meaningful • Combinatorial explosion single distributed behavior results in a huge number of linear traces	• Visual description of behavior • Interactions are visible • no combinatorial explosion

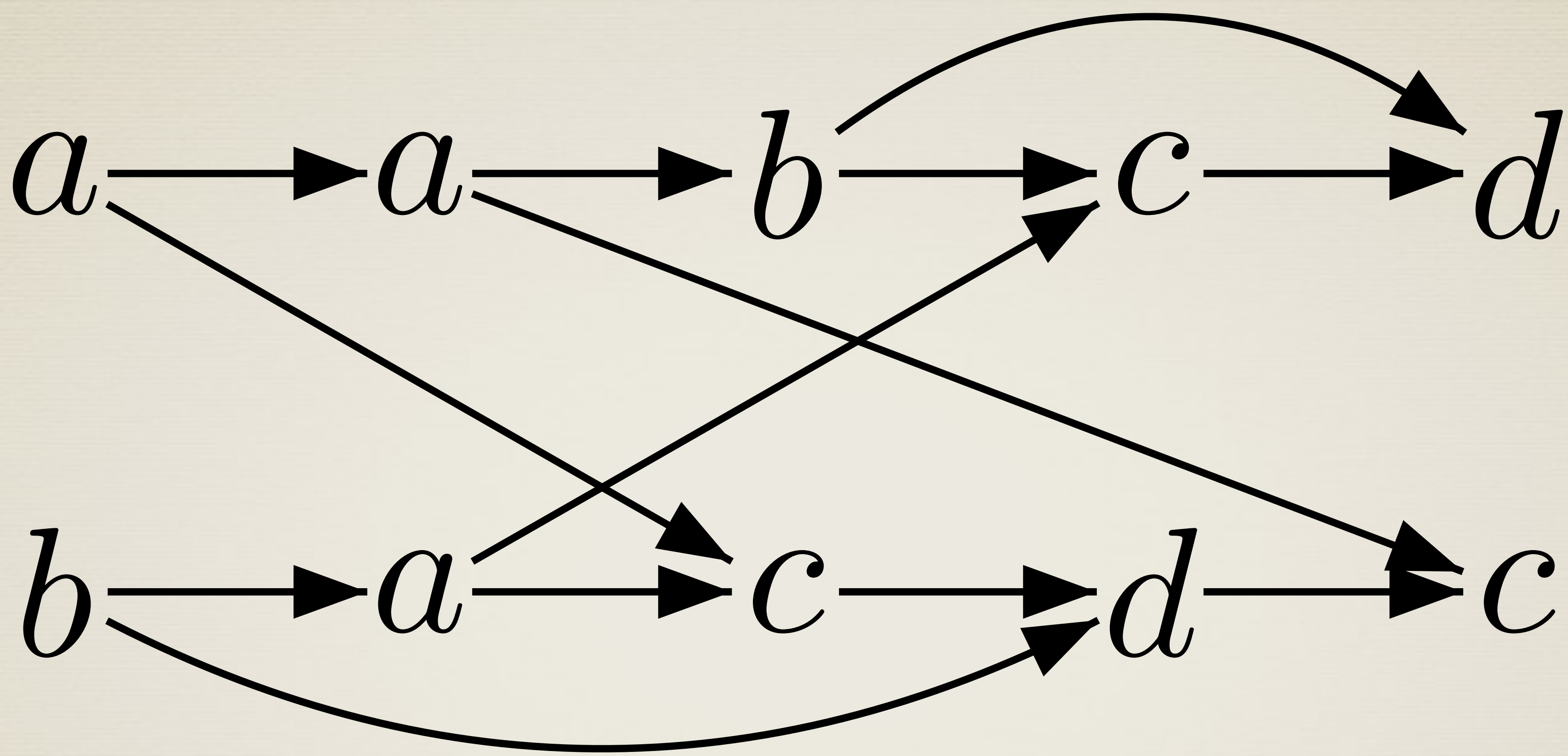
WYSIWYG

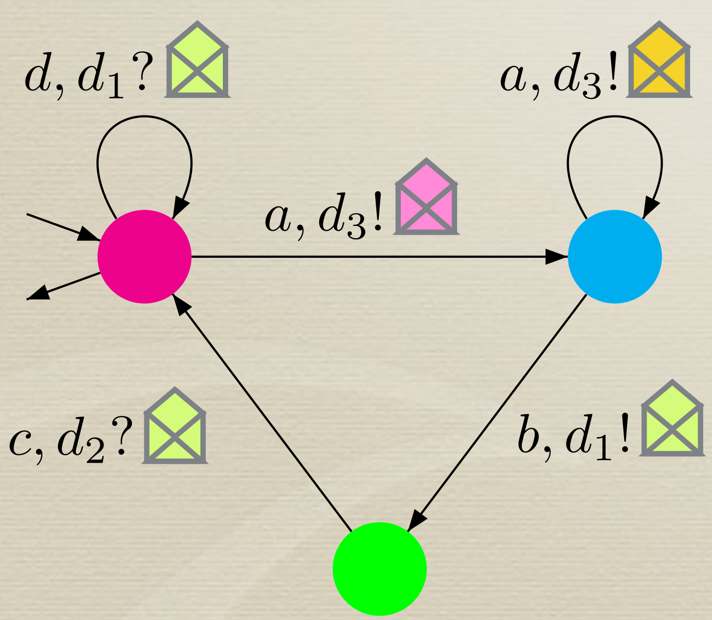
Understanding Behaviors

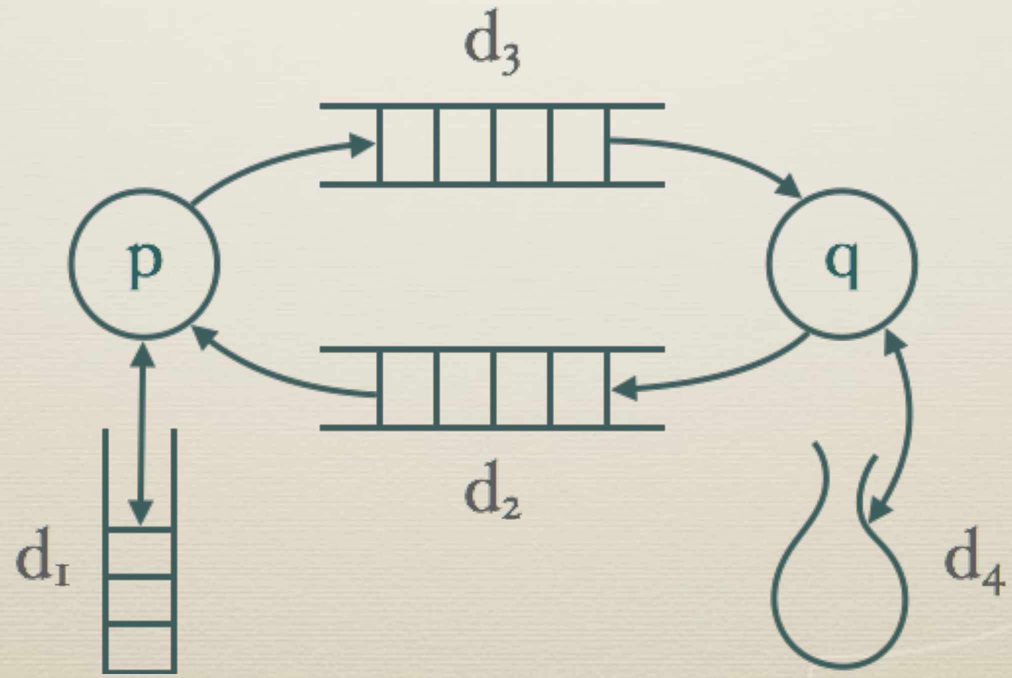
Linear Traces	Graphs (CBMs)
• Interleaved sequence of events. Interactions are obfuscated and very difficult to recover • Successor … • Com… singl… …havior results in a huge number of linear traces	• …avior • …s are visible • no combinatorial explosion

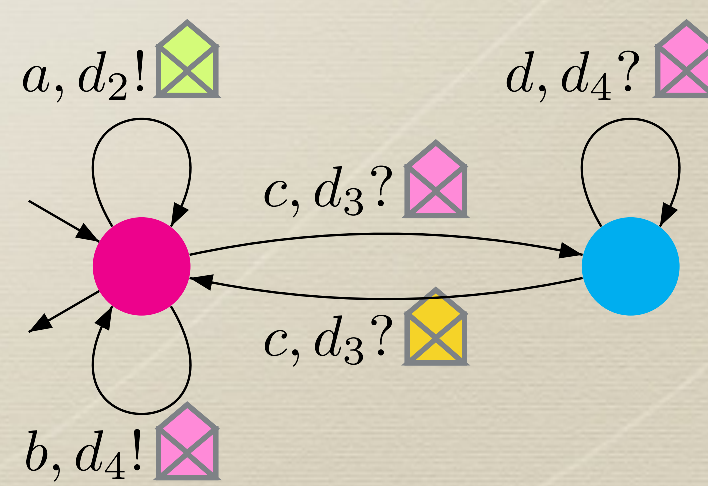
Behaviors should be graphs

Semantics of CPDS on Graphs

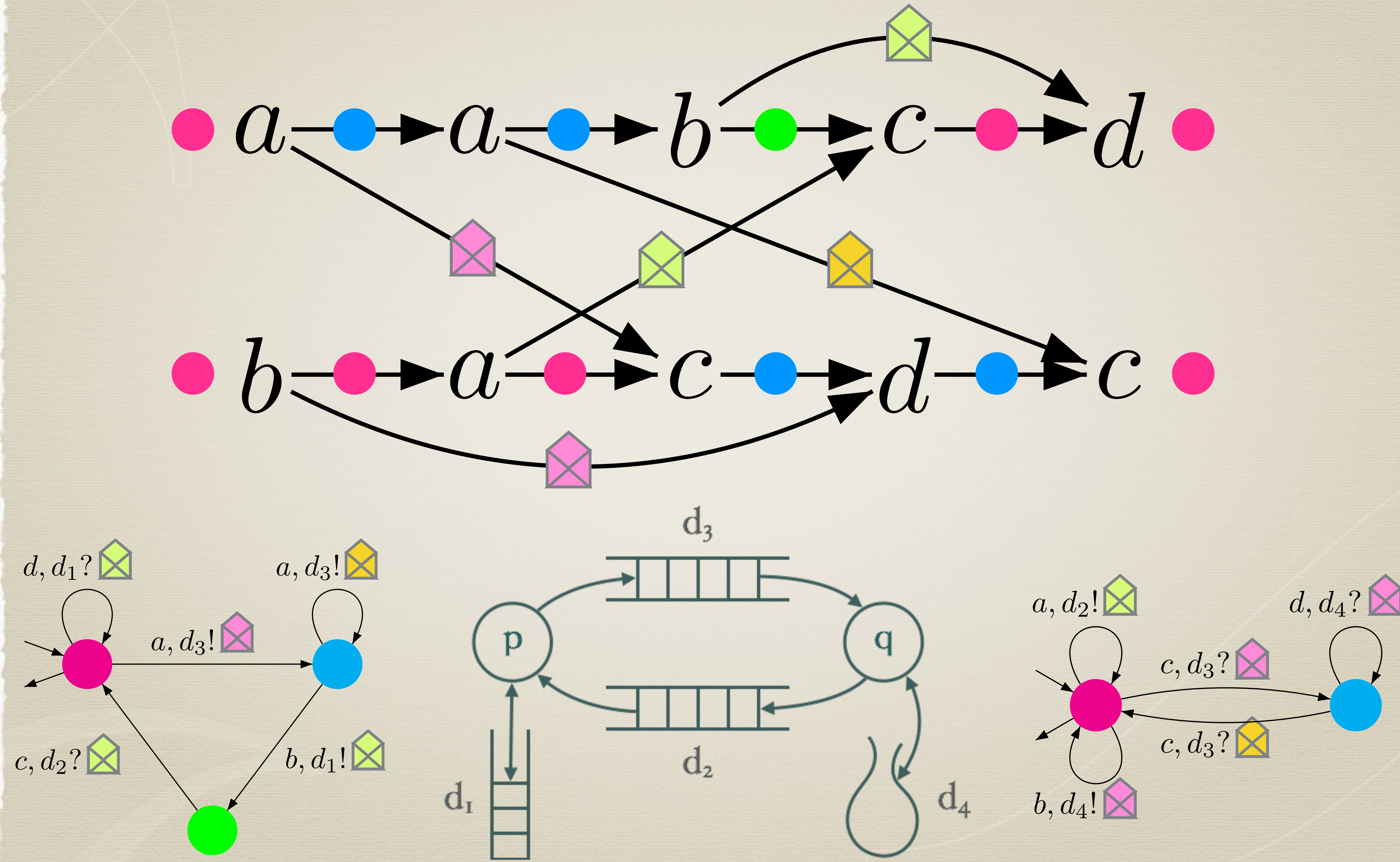




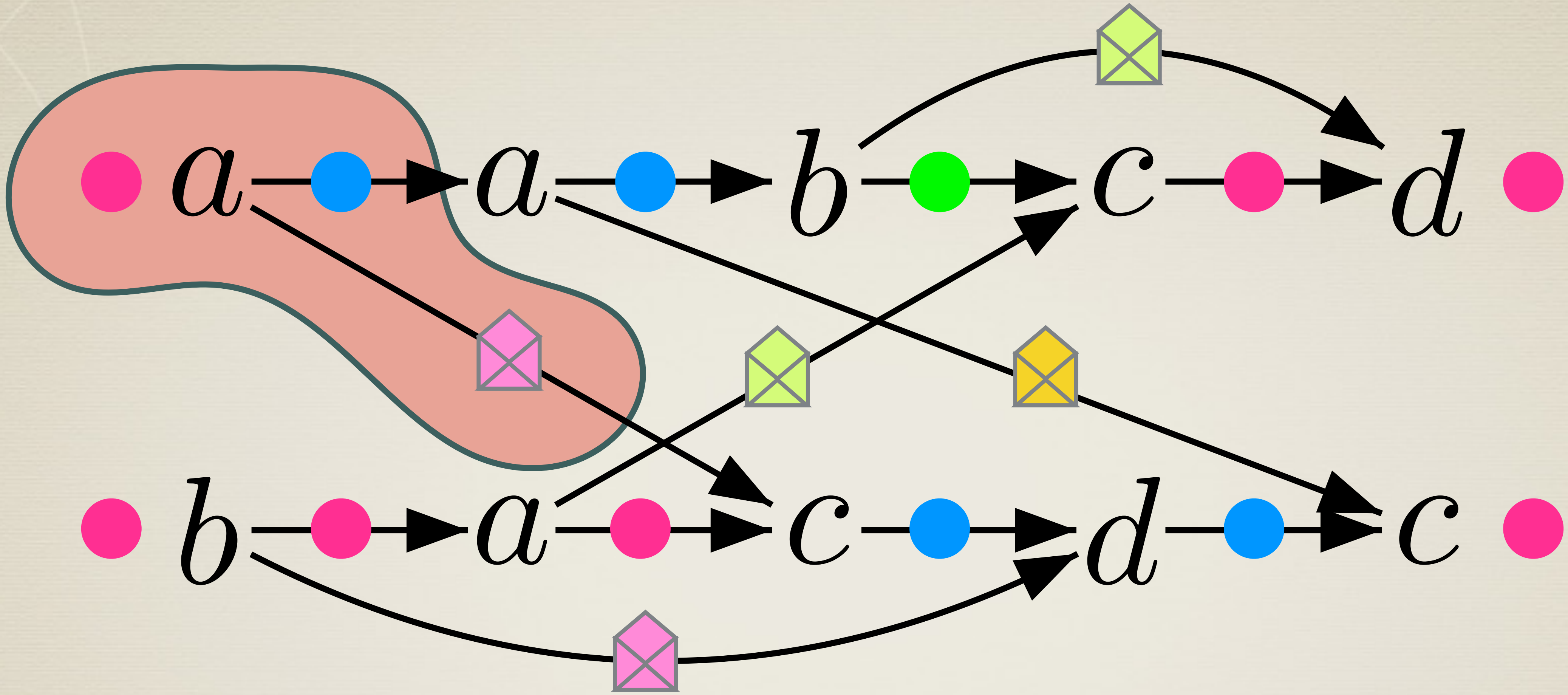


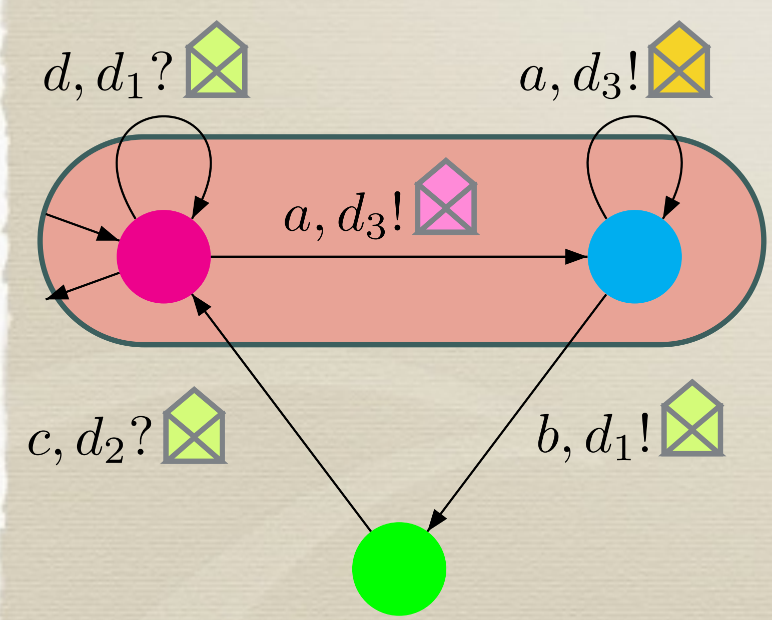


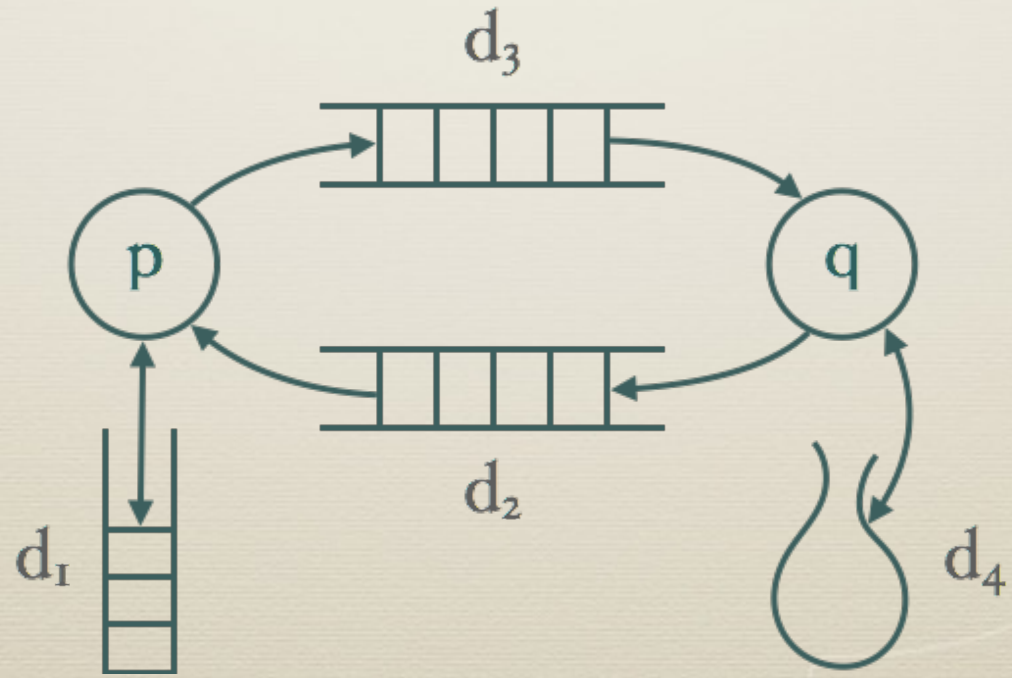
Semantics of CPDS on Graphs

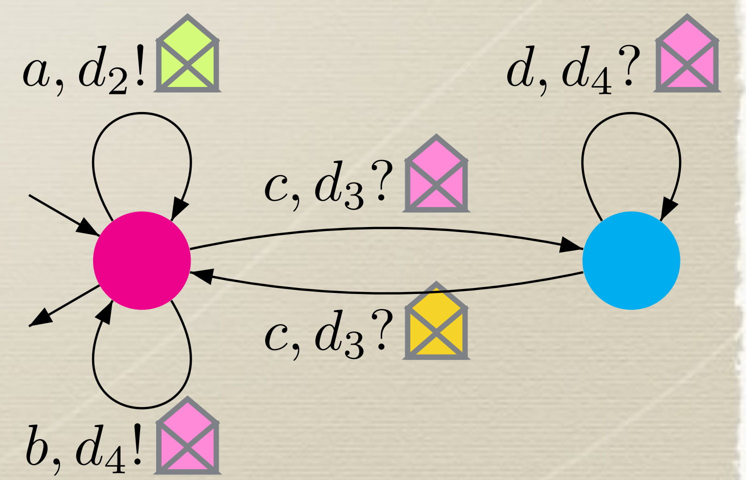


Semantics of CPDS on Graphs

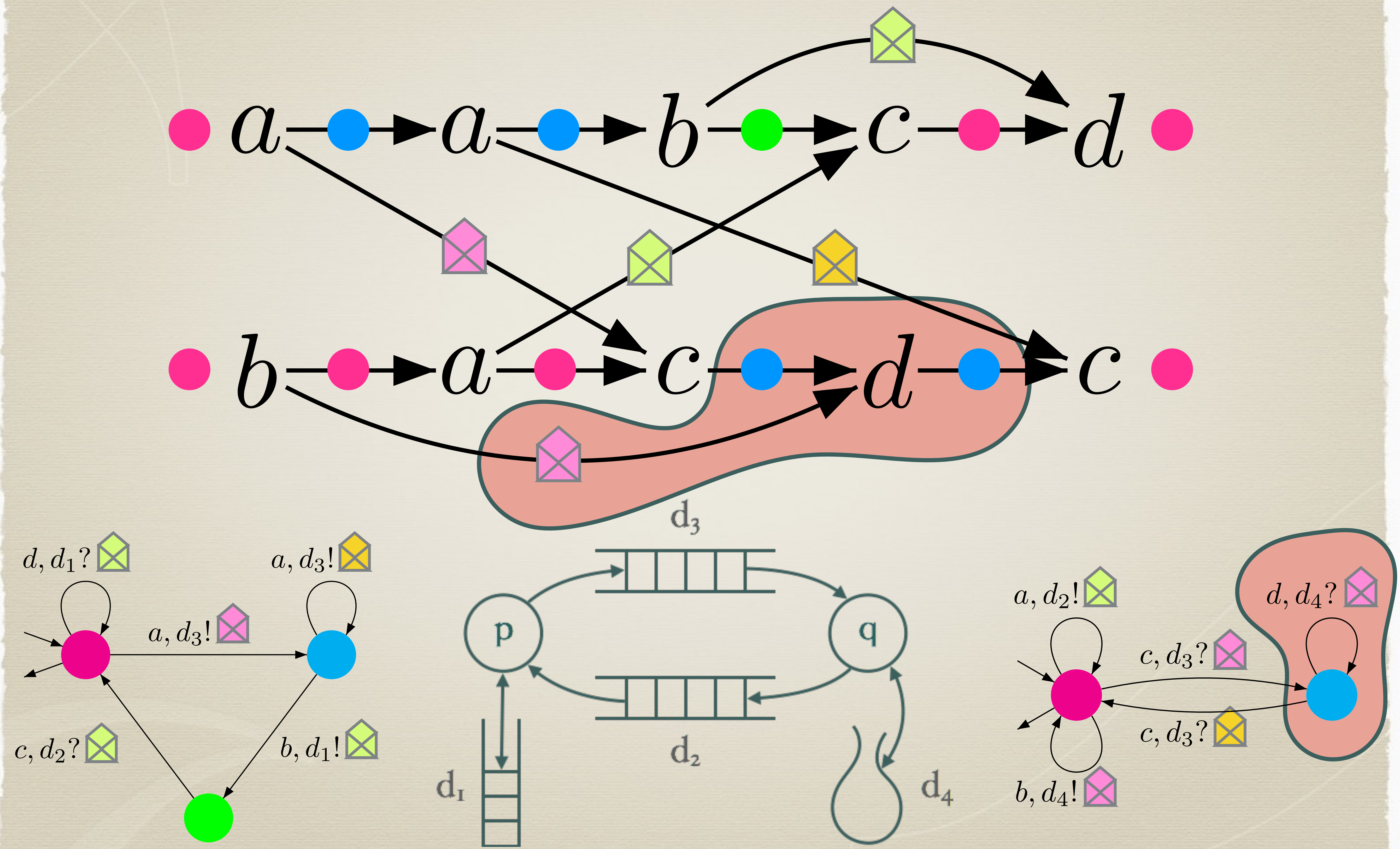








Semantics of CPDS on Graphs



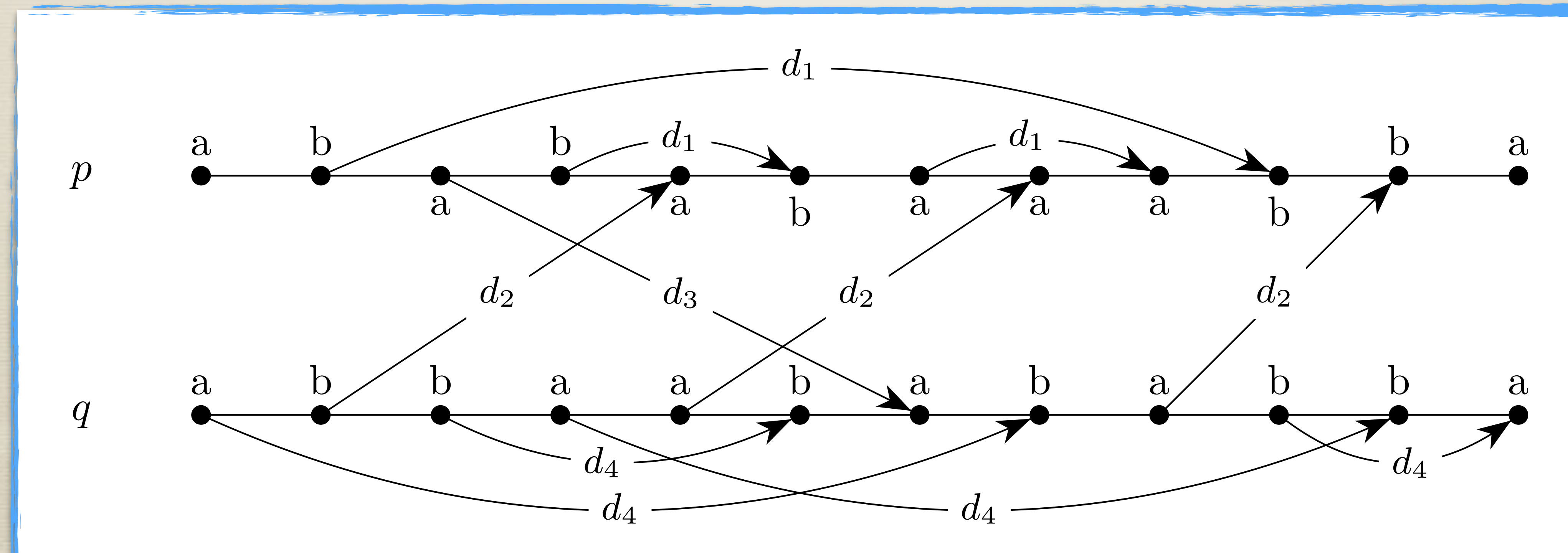
Outline

- ☑ Concurrent Processes with Data Structures
- ☑ Behaviors as Graphs
- Specifications
- Verification with Graphs and under-approximations
- Split-width and tree interpretation
- Conclusion

Specification over Graphs

MSO: Monadic Second Order Logic

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



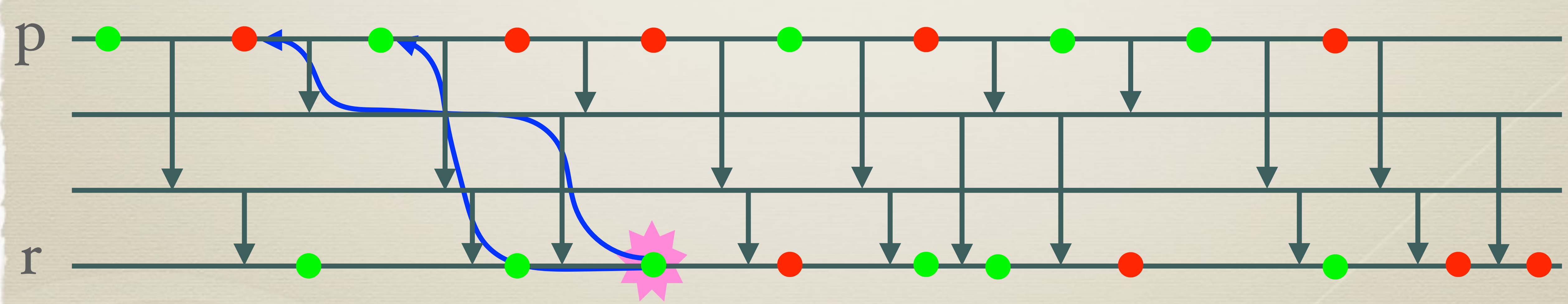
Specification over Graphs
Obey the latest order

TL $\mathsf{G}(r \wedge \mathsf{on} \Rightarrow \mathsf{Latest}_p\, \mathsf{Y}_p\, \mathsf{on})$

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

$\wedge\, \forall x\, (x < z \wedge p(x) \Rightarrow x \leq y)$

$\wedge\, \exists x\, (x \rightarrow y \wedge \mathsf{on}(x)))$



Specification over Linear Traces

$$(p, \mathsf{on})(p, c_2!)(p, \mathsf{off})(q_2, c_2?)(p, c_1!)(q_1, c_1?)$$
$$(q_2, c_4!)(p, \mathsf{on})(p, c_2!)(p, \mathsf{off})(r, c_4?)(r, \mathsf{on})$$
$$(q_1, c_3!)(p, c_1!)(q_1, c_1?)(q_1, c_3!)(q_2, c_2?)(q_2, c_4!)$$
$$(r, c_4?)(r, \mathsf{on})(r, c_3?)(r, \mathsf{off}) \cdots$$

- Based on the word successor relation, and the word total order
- LTL over words, MSO over words

Process successor can be recovered
Data edges cannot in general

Specification over Linear Traces

$(p, \mathsf{on})(p, c_2!)(p, \mathsf{off})(q_2, c_2?)(p, c$ …

$(q_2, c_4!)(p, \mathsf{on})($ …

Obey the latest order
not expressible
in MSO over Linear Traces

* Based o … word total order
* LTL ove … ds, MSO over words

Process successor can be recovered
Data edges cannot in general

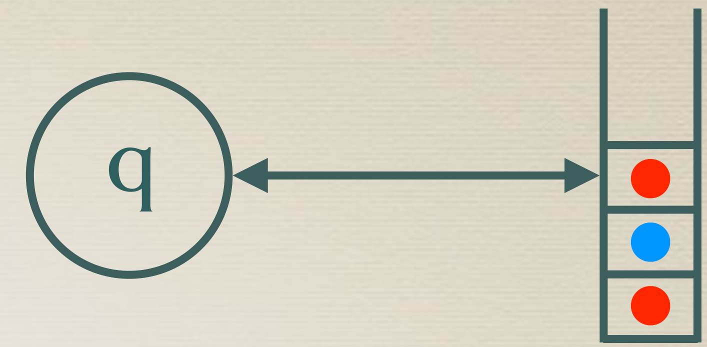
Specification over Linear Traces

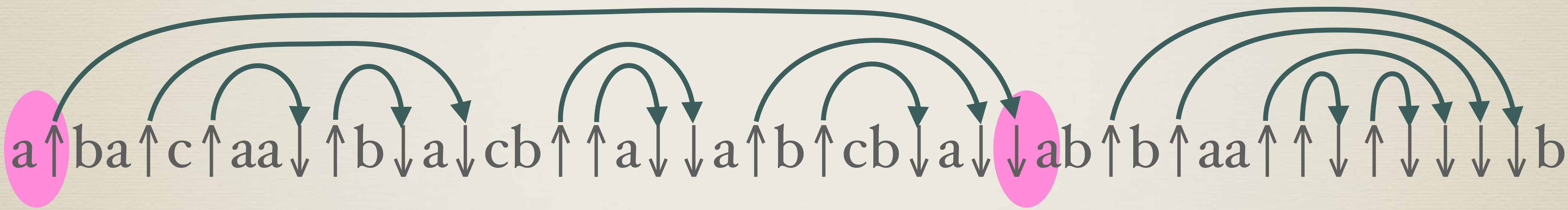
$$(p, \mathsf{on})(p, c_2!)(p, \mathsf{off})(q_2, c_2?)(p, c_1!)(q_1, c_1?)$$
$$(q_2, c_4!)(p, \mathsf{on})(p, c_2!)(p, \mathsf{off})(r, c_4?)(r, \mathsf{on})$$
$$(q_1, c_3!)(p, c_1!)(q_1, c_1?)(q_1, c_3!)(q_2, c_2?)(q_2, c_4!)$$
$$(r, c_4?)(r, \mathsf{on})(r, c_3?)(r, \mathsf{off}) \cdots$$

- Based on the word successor relation, and the word total order
- LTL over words, MSO over words
- LTL specification are not always meaningful
  LTL \ X, Closure properties, ...
- Natural properties of graphs are difficult or impossible to express on linear traces

Graphs for Sequential Systems

Answer the correct client for topmost requests

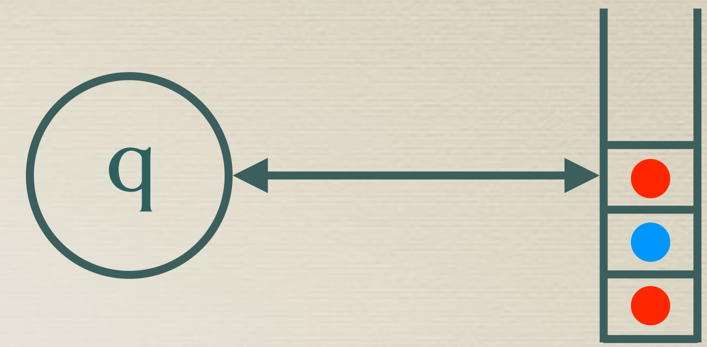




$$\forall x, y \begin{pmatrix} a(x-1) \wedge x \rhd y \wedge \\ \neg\exists z, z' \, (z \rhd z' \wedge z < x < z') \end{pmatrix} \Rightarrow a(y+1)$$

Graphs for Sequential Systems

Answer the correct client
for topmost requests



a↑ba↑c↑aa↓↑b↓a↓cb↑↑a↓↓a↑b↑cb↓a↓↓ab↑b↑aa↑↑↓↑↓↓↓↓b

Not expressible in MSO over Linear Traces
without nesting relation
even with visible alphabet

WYSIWYG

Expressiveness of Specifications

Linear Traces	Graphs (CBMs)
• Too weak for many natural specifications • Difficult to write/understand • Requires syntactical or semantical restrictions to be meaningful	• Powerful specifications • Interactions are built-in • Meaningful • Easy to write/understand

WYSIWYG

Expressiveness of Specifications

Linear Traces	Graphs (CBMs)
• Too weak f… specifica… • Difficult… • Requires … cal restrictio… be meaningful	• … ons • … built-in Meaningful • Easy to write/understand

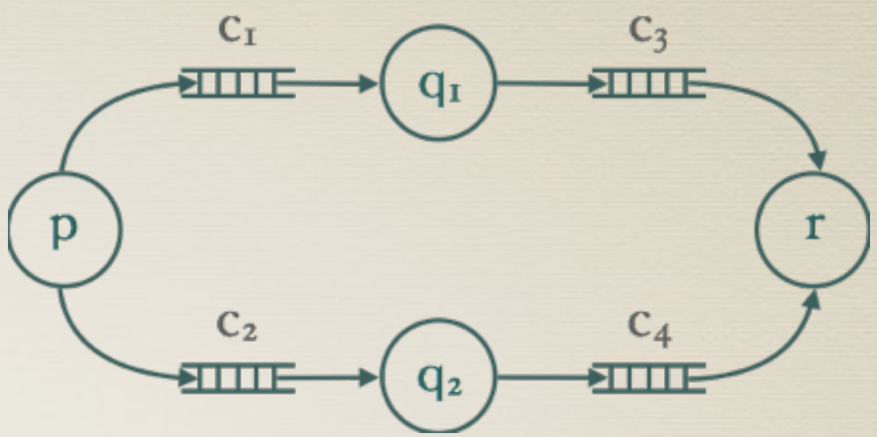
Specifications should be on graphs

Outline

- ☑ Concurrent Processes with Data Structures
- ☑ Behaviors as Graphs
- ☑ Specifications
- * Verification with Graphs and under-approximations
- * Split-width and tree interpretation
- * Conclusion

Verification problems

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



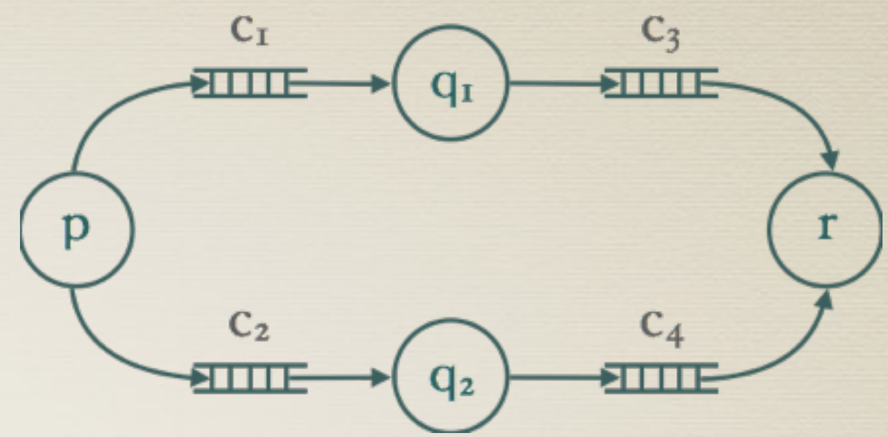
Obey the latest order

$$\mathsf{G}(r \wedge \mathsf{on} \Rightarrow \mathsf{Latest}_p\, \mathsf{Y}_p\, \mathsf{on})$$

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

Verification problems

* Emptiness or Reachability
* Inclusion or Universality
* Satisfiability ϕ
* Model Checking: S ⊨ ϕ
* Temporal logics
* Prop
* Mona logic



Obey the latest order

undecidable in general

$\mathsf{t}_p\ \mathsf{Y}_p$ on)

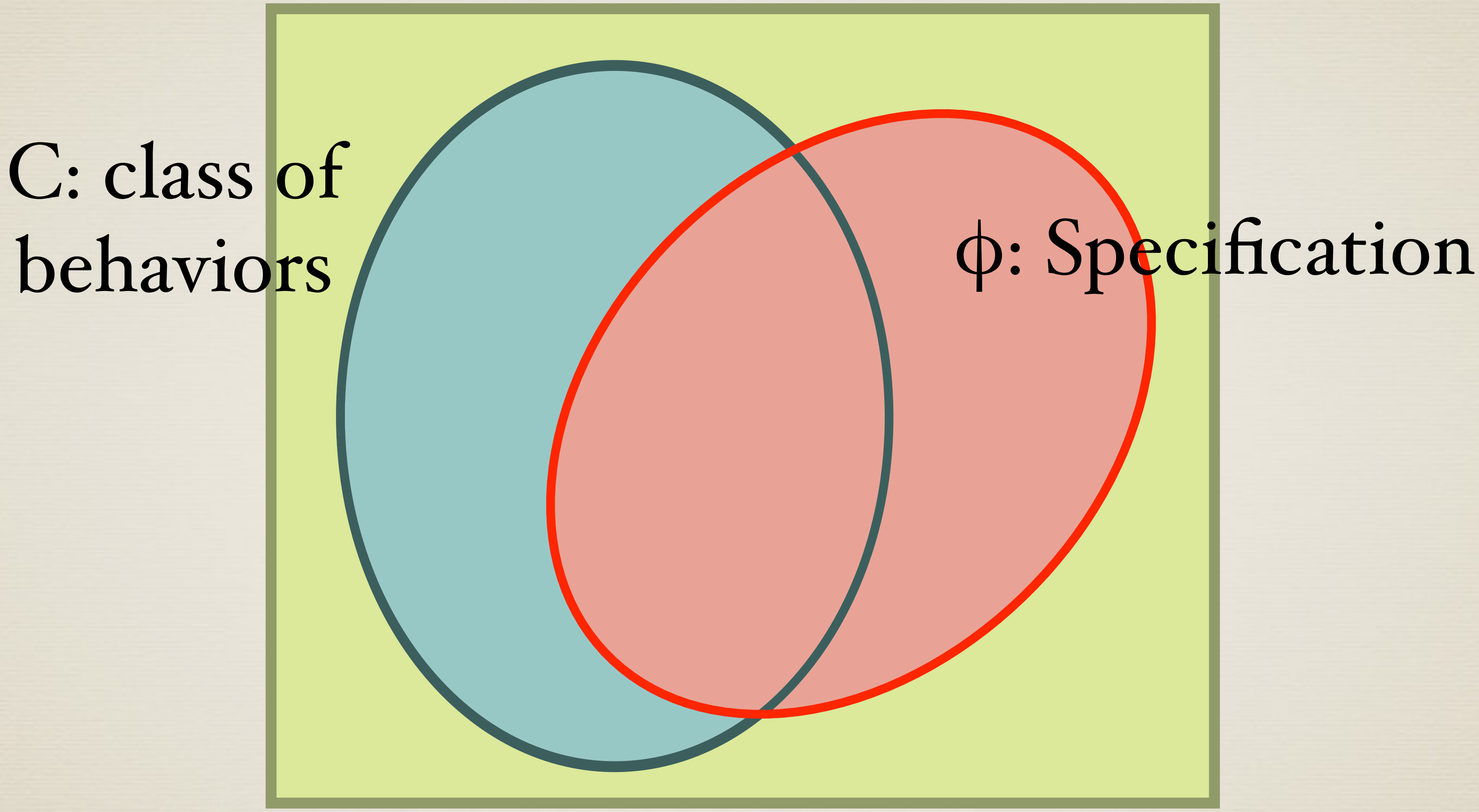
$(z) \wedge \mathsf{on}(z)) \Rightarrow \exists y\,(p(y) \wedge y < z$

$\wedge\, \forall x\,(x < z \wedge p(x) \Rightarrow x \leq y)$

$\wedge\, \exists x\,(x \rightarrow y \wedge \mathsf{on}(x)))$

Under-approximate Verification

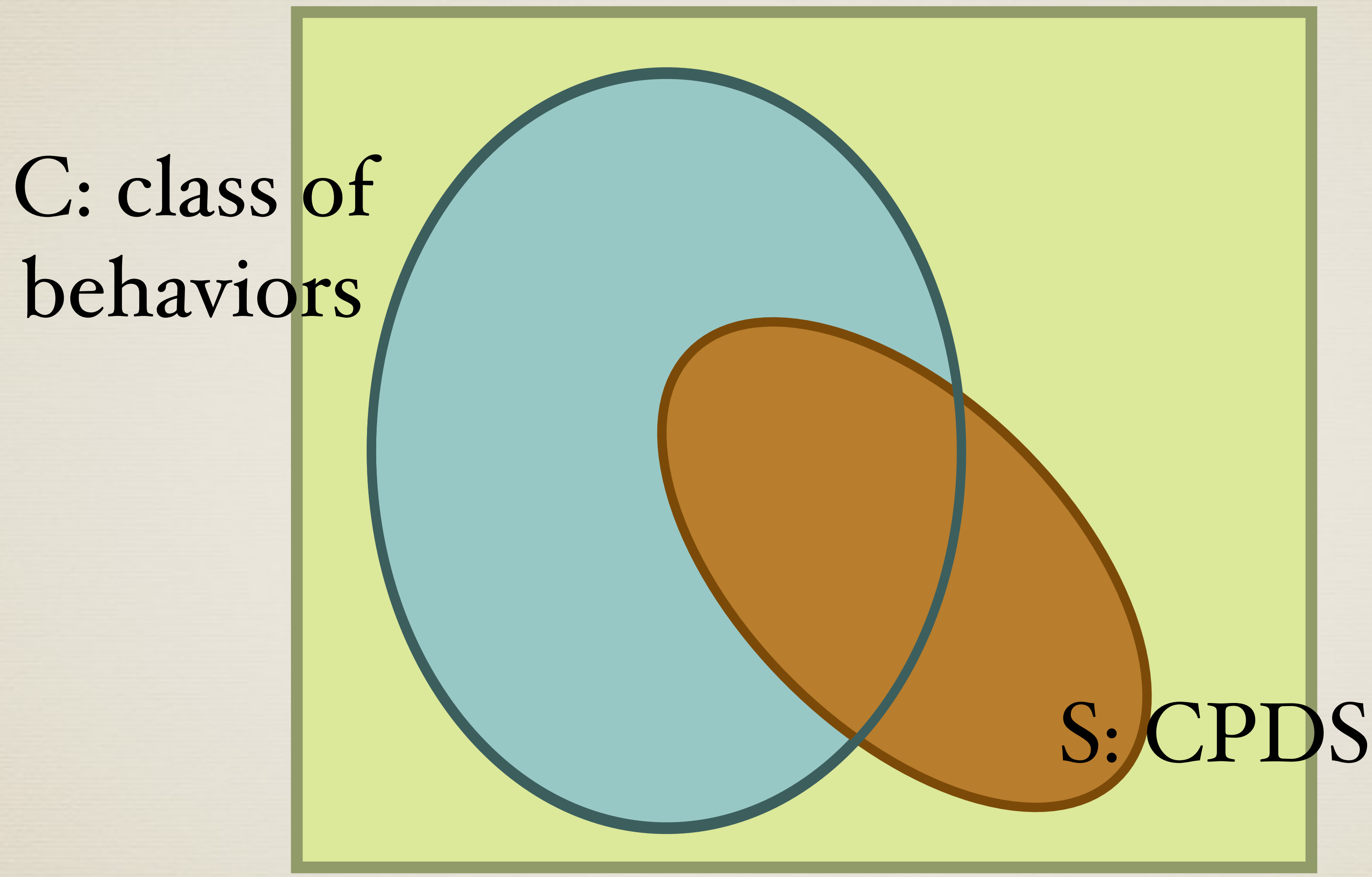
Satisfiability problem:



Is ϕ satisfiable in C?

Under-approximate Verification

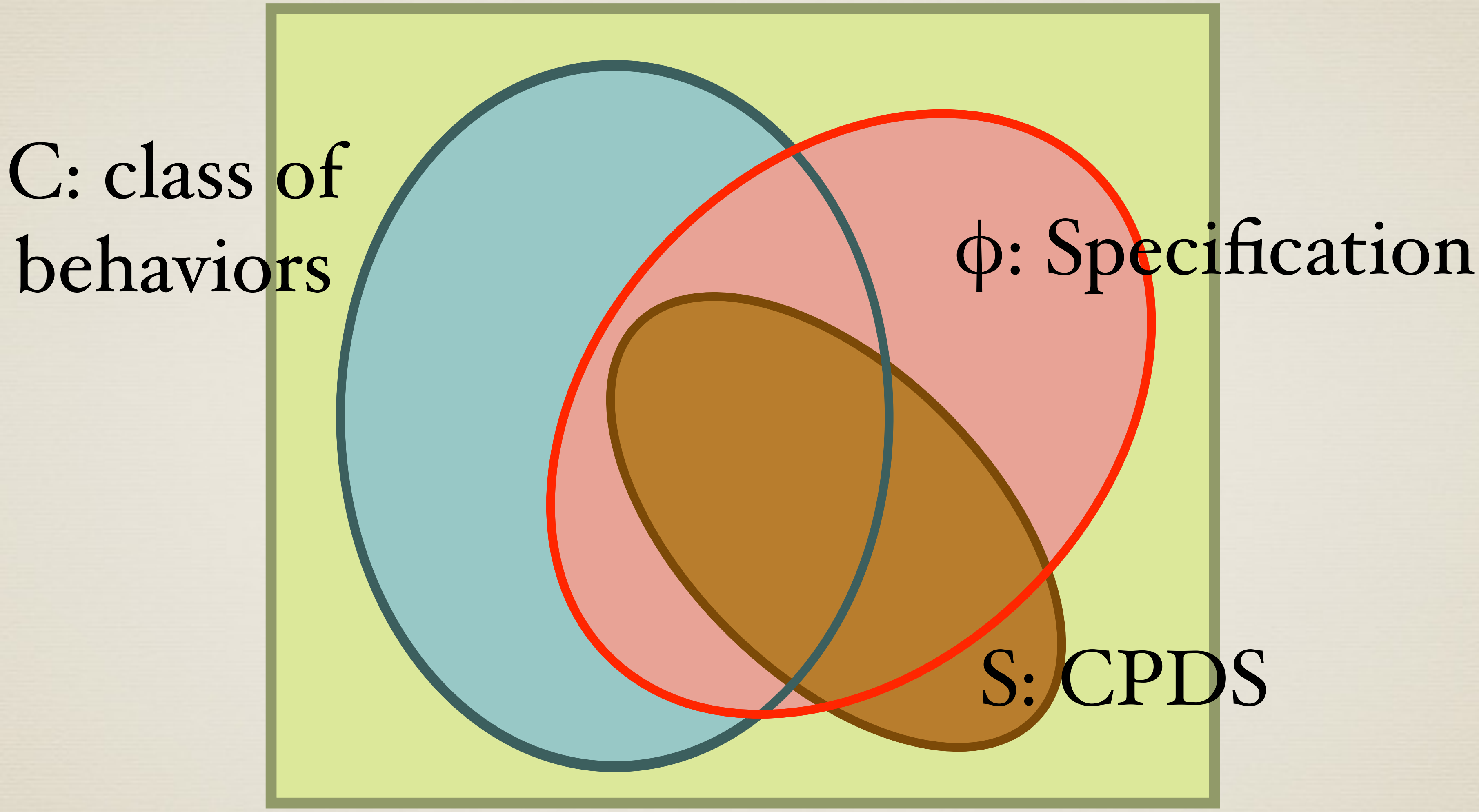
Emptiness or reachability problem:



Is-there a run of S
on some behavior from C?

Under-approximate Verification

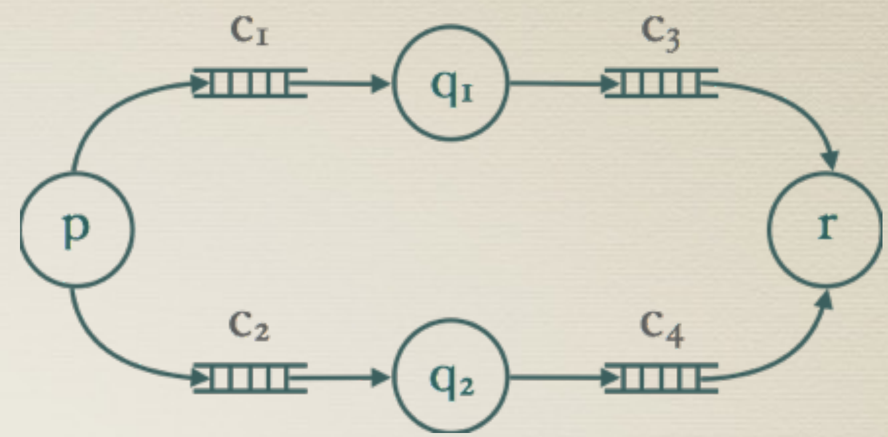
Model checking problem: $S \vDash_C \phi$



Do all behaviors from C accepted by S satisfy ϕ?

Verification problems

* Emptiness or Reachability
* Inclusion or Universality
* Satisfiability ϕ
* Model Checking: S ⊨ ϕ
* Temporal logics
* Prop
* Mona ... ogic



Obey the latest order

undecidable in general

$t_p \, \mathsf{Y}_p \, \mathsf{on})$

$\ldots(z) \wedge \mathsf{on}(z)) \Rightarrow \exists y \, (p(y) \wedge y < z$

$\wedge \, \forall x \, (x < z \wedge p(x) \Rightarrow x \leq y)$

$\wedge \, \exists x \, (x \rightarrow y \wedge \mathsf{on}(x)))$

Under-approximate Verification

Mainly for reachability

* Emptiness or Reachability
* Inclusion or Univ

undecidable

g. S ⊨ ϕ

* Temporal logics
* Propositional dynamic logics
* Monadic second order logic

* Bounded data structures
* 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.]

Model Checking vs Reachability

* Reachability reduces to model checking
* Model checking reduces to Reachability ...

  ... when specifications can be translated to systems

  ... this is **not possible** in general for graphs

$$S \models \phi$$

$$S_{\neg\phi}$$

$$S \cap S_{\neg\phi} = \emptyset$$

Model Checking vs Reachability

- Reachability reduces to model checking
- Model checking reduces to Reachability ...

  ... when specifications can be translated to systems

  ... this is **not possible** in general for graphs

Obey the latest order

Answer the correct client for topmost requests

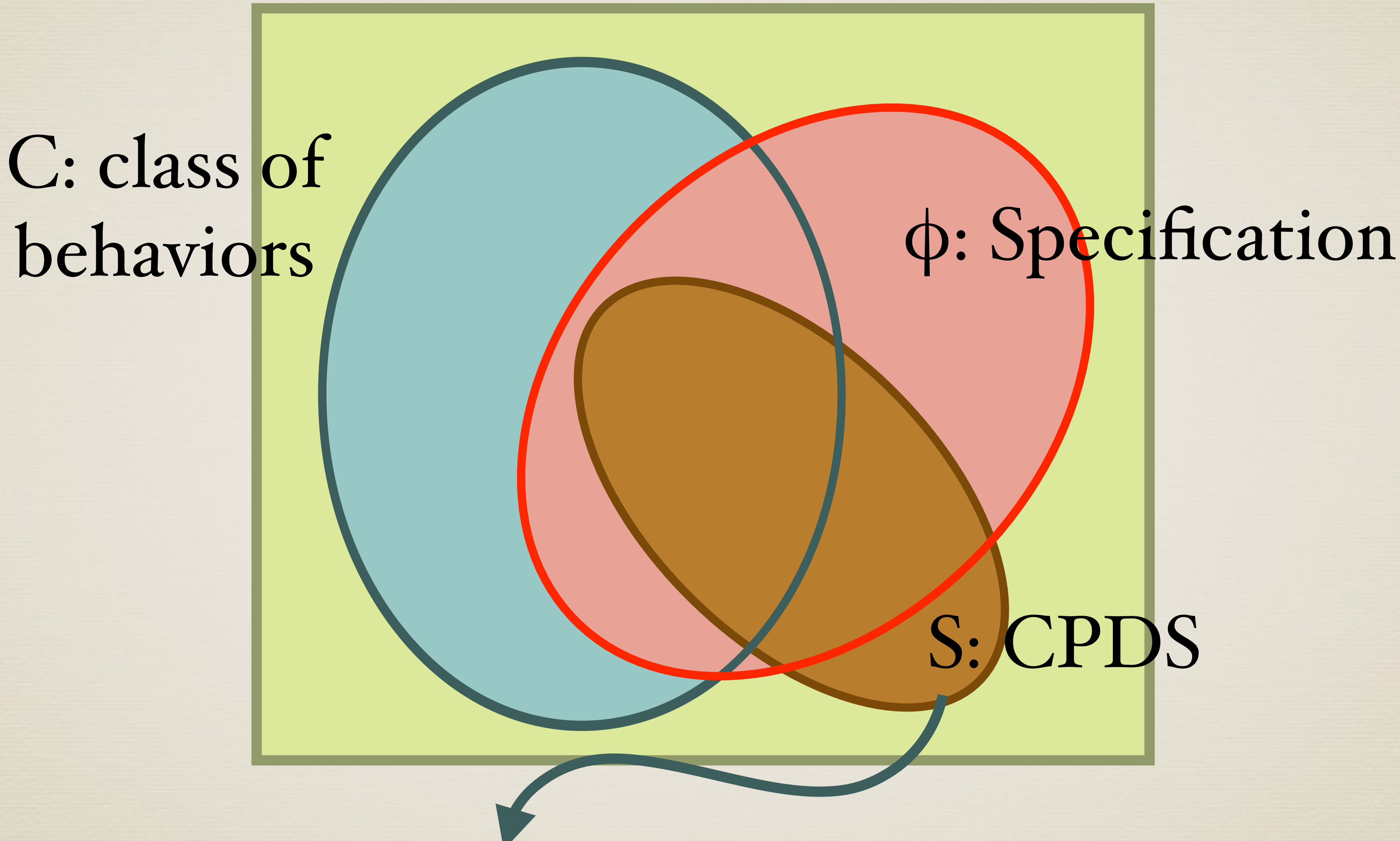
$$S \models \phi$$

$$S_{\neg\phi}$$

$$S \cap S_{\neg\phi} = \emptyset$$

Under-approximate Verification

Model checking problem: $S \models_C \phi$



$S \models_C \phi$ iff $\phi_S \Rightarrow \phi$ is valid in C

ENCYCLOPEDIA OF MATHEMATICS AND ITS APPLICATIONS

Graph Structure and Monadic Second-Order Logic

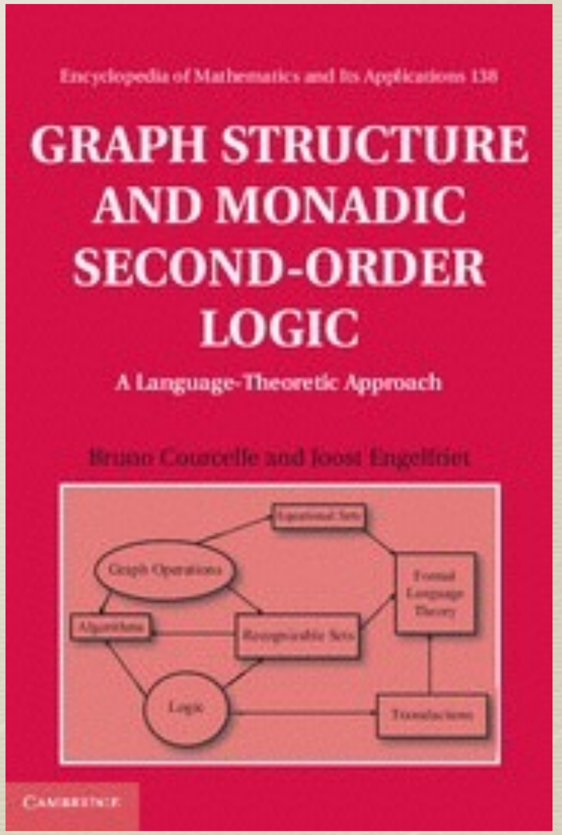
A Language-Theoretic Approach

BRUNO COURCELLE

Université de Bordeaux

JOOST ENGELFRIET

Universiteit Leiden



Decidability of MSO theory

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 CBMs)

Reduction to the theory of
Tree Automata

Under-approximate Verification

Mainly for reachability

undecidable

* Emptiness or Reachability
* Inclusion or Univ
* g. S ⊨ φ
* Temporal logics
* Propositional dynamic logics
* Monadic second order logic

* 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.]

Under-approximate Verification

Mainly for reachability

undecidable

* Emptiness or Reachability
* Inclusion or Universality
* Satisfiability ϕ
* Model Checking: S ⊨ ϕ
* Temporal logics
* Propositional dynamic logics
* Monadic second order logic

* 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.]

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

Outline

- ☑ Concurrent Processes with Data Structures
- ☑ Behaviors as Graphs
- ☑ Specifications
- ☑ Verification with Graphs and under-approximations
- Split-width and tree interpretation
- Conclusion

joint work with
C. Aiswarya
K. Narayan Kumar

Width: split vs tree vs clique

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 CBMs)

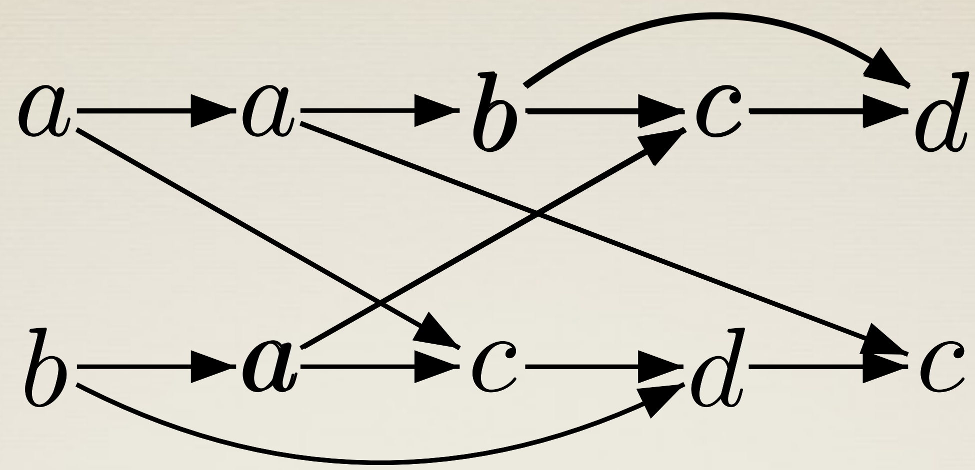
Width: split vs tree vs clique

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

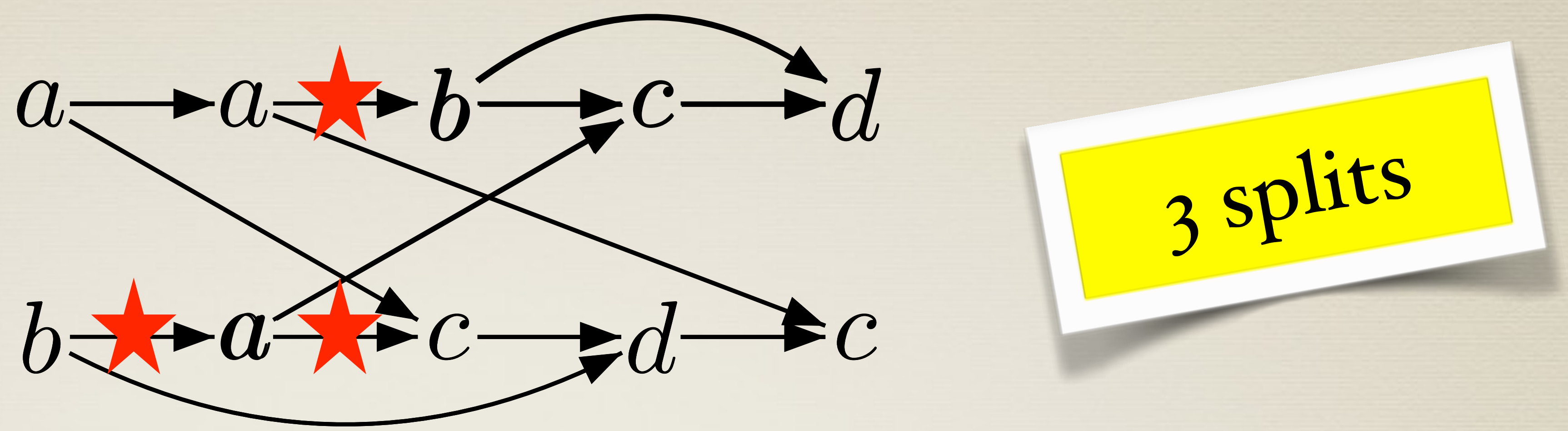
1. C has a decidable MSO theory
2. C can be interpreted in binary trees
3. C has bounded tree-width
4. C has bounded clique-width
5. C has bounded split-width (for CBMs)

Decomposition game

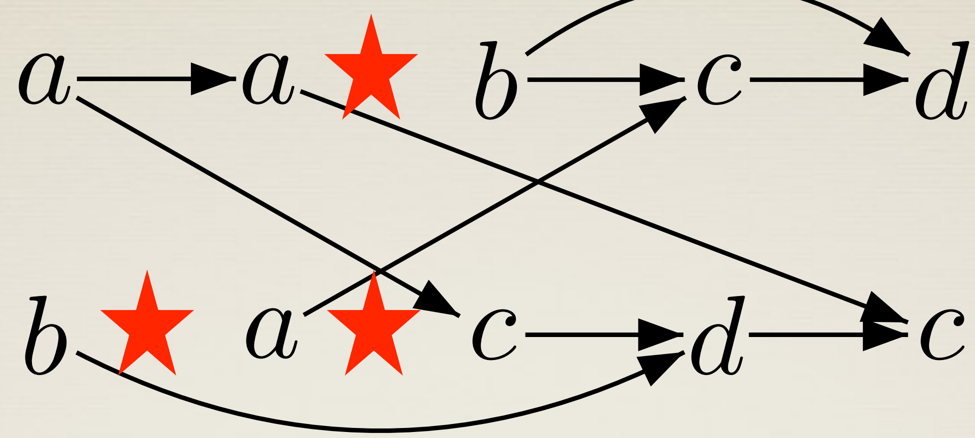
* Eve: Disconnect the graph by cutting edges
* Adam: Choose a connected component
* Split game: Eve cuts process edges only (CBM)
* Width: Maximal number of holes in graphs (CBMs) along a play until reaching an atomic graph



SPLIT DECOMPOSITION OF CBMs

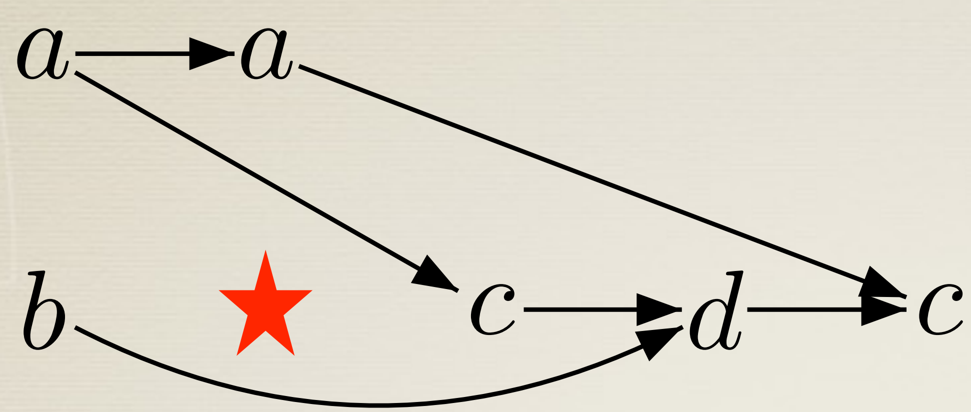


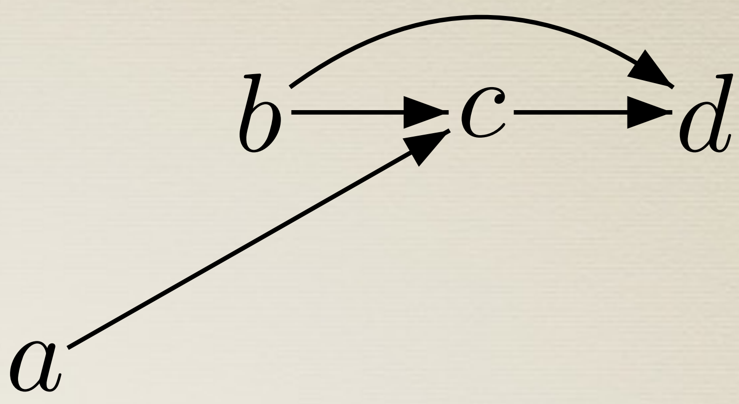
SPLIT DECOMPOSITION OF CBMs



3 holes

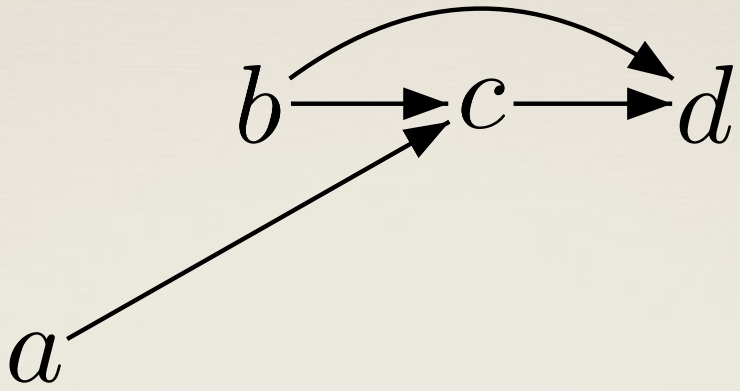
SPLIT DECOMPOSITION OF CBMs



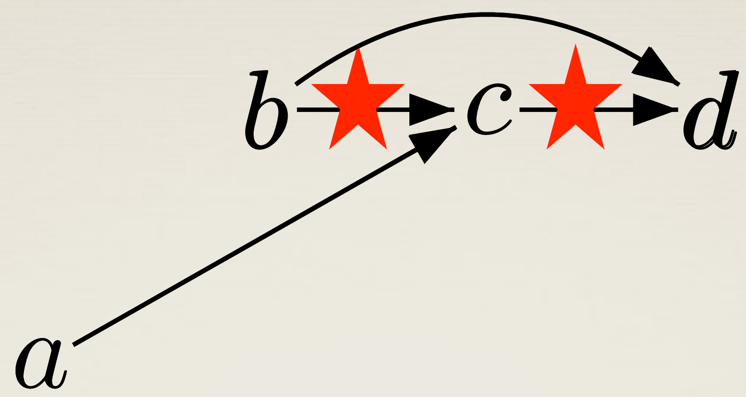


1 hole left

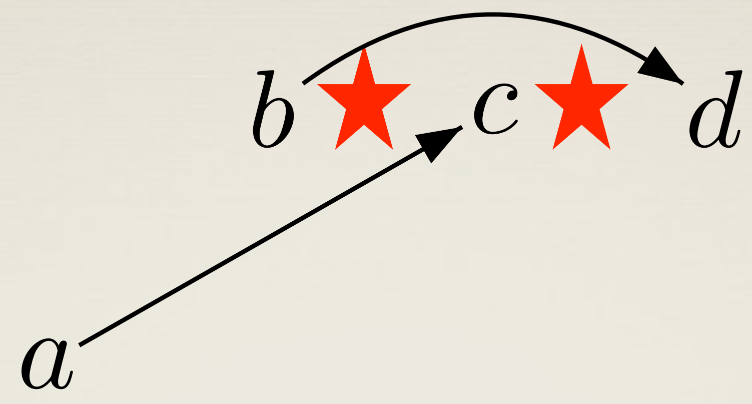
SPLIT DECOMPOSITION OF CBMs



SPLIT DECOMPOSITION OF CBMs



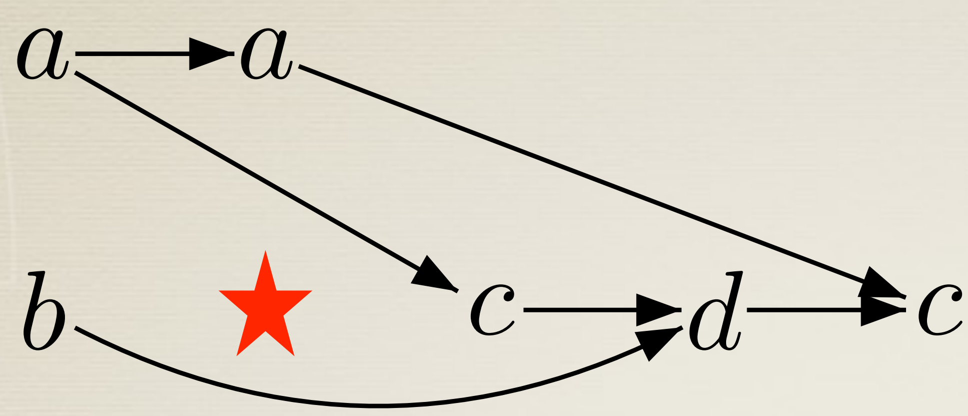
SPLIT DECOMPOSITION OF CBMs

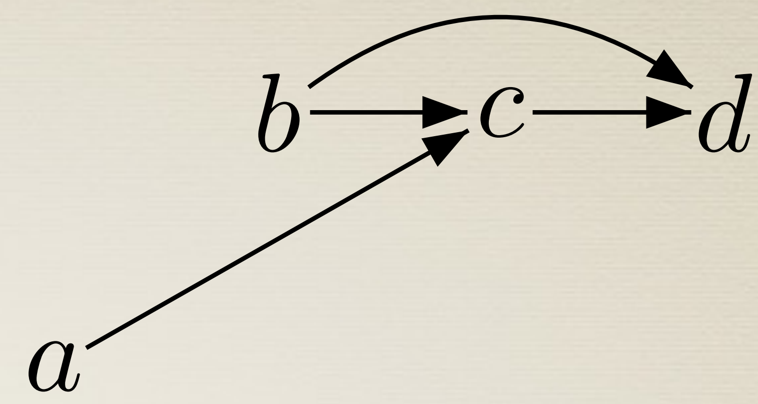


2 holes

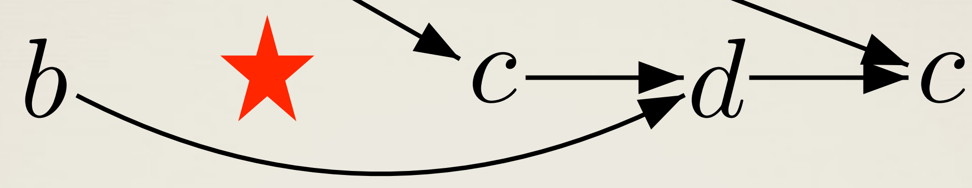
SPLIT DECOMPOSITION OF CBMs

SPLIT DECOMPOSITION OF CBMs

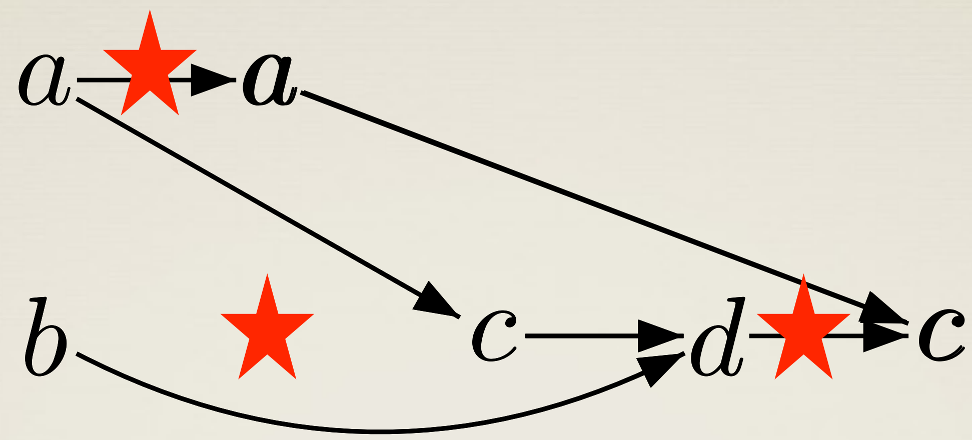




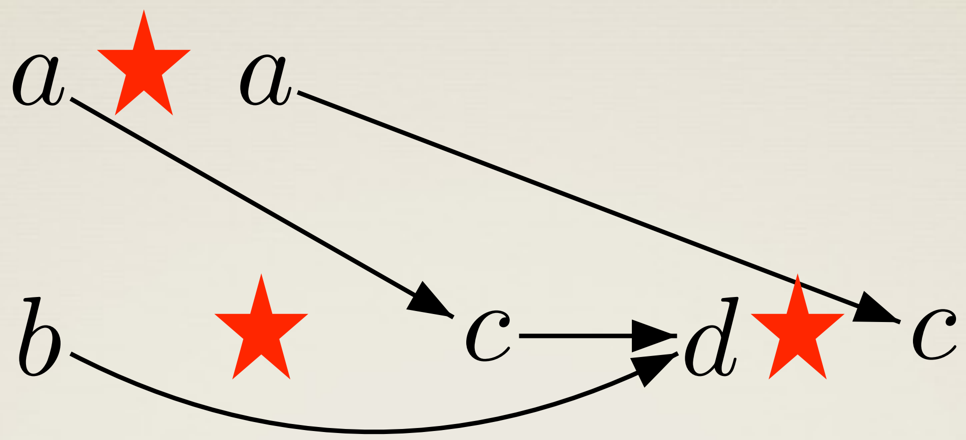
SPLIT DECOMPOSITION OF CBMs



SPLIT DECOMPOSITION OF CBMs

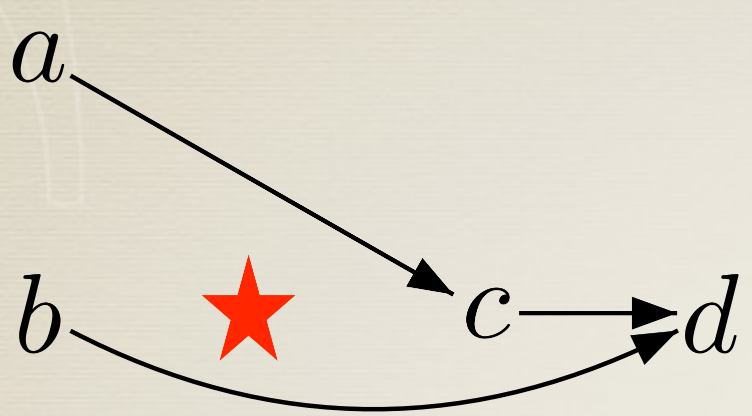


SPLIT DECOMPOSITION OF CBMs

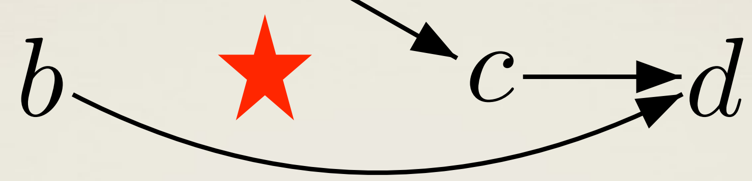


3 holes

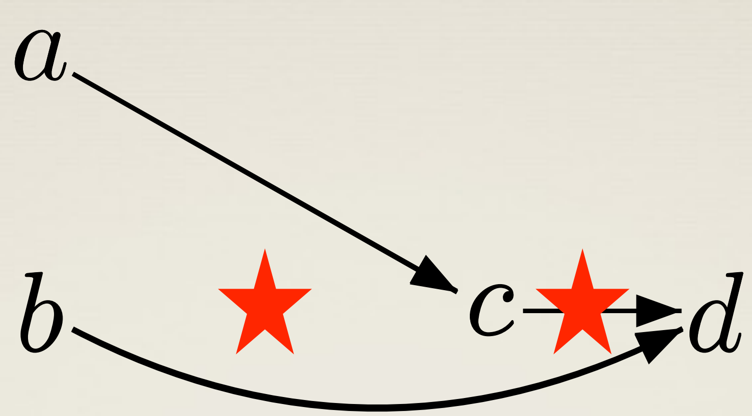
SPLIT DECOMPOSITION OF CBMs



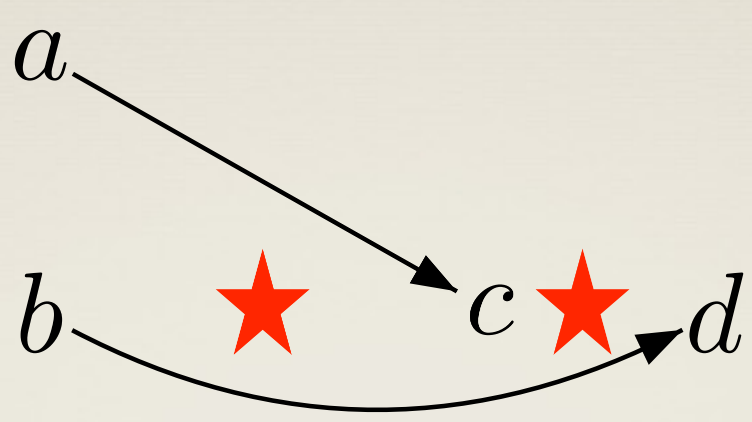
SPLIT DECOMPOSITION OF CBMs



SPLIT DECOMPOSITION OF CBMs



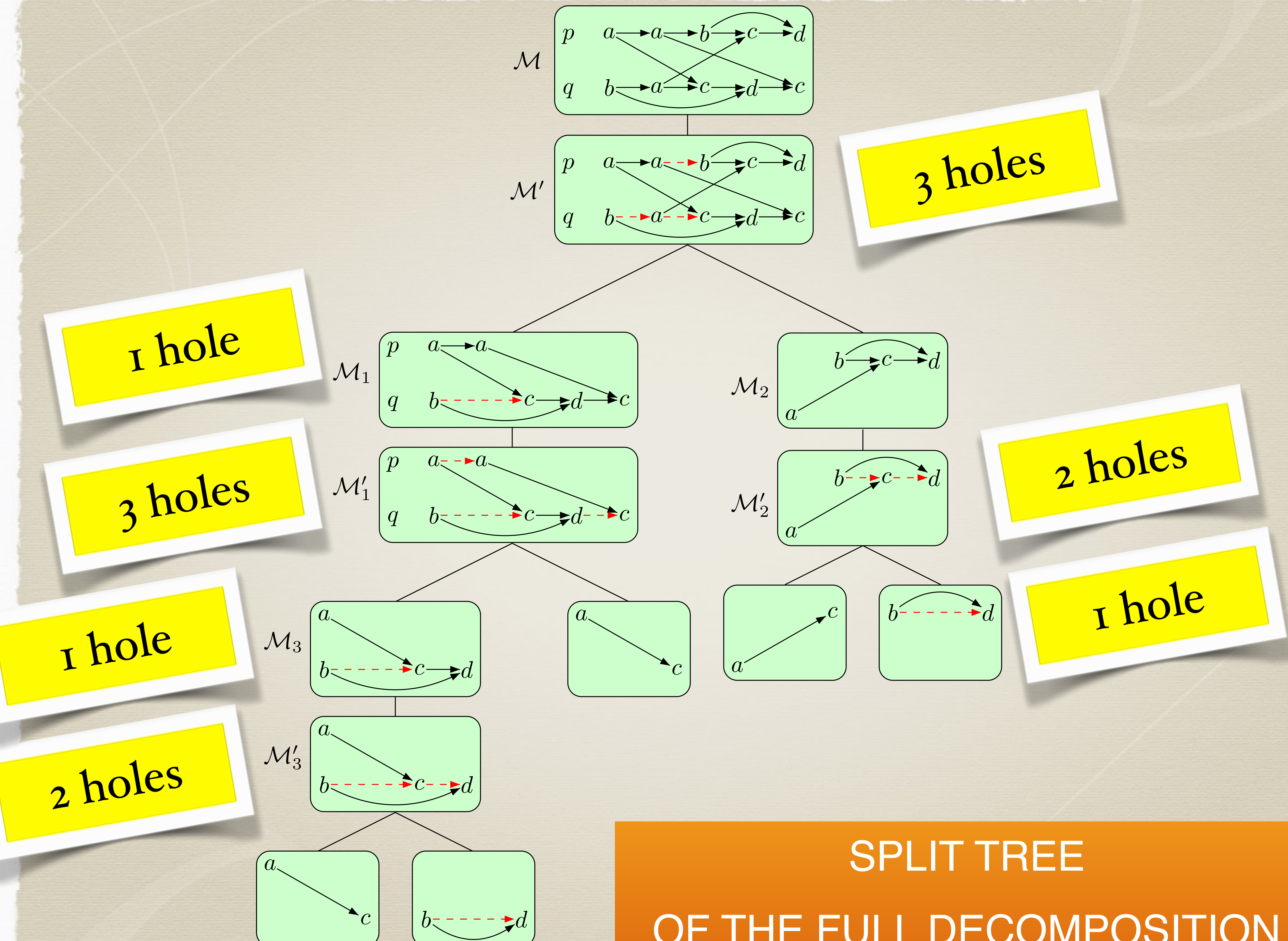
SPLIT DECOMPOSITION OF CBMs



2 holes

SPLIT DECOMPOSITION OF CBMs

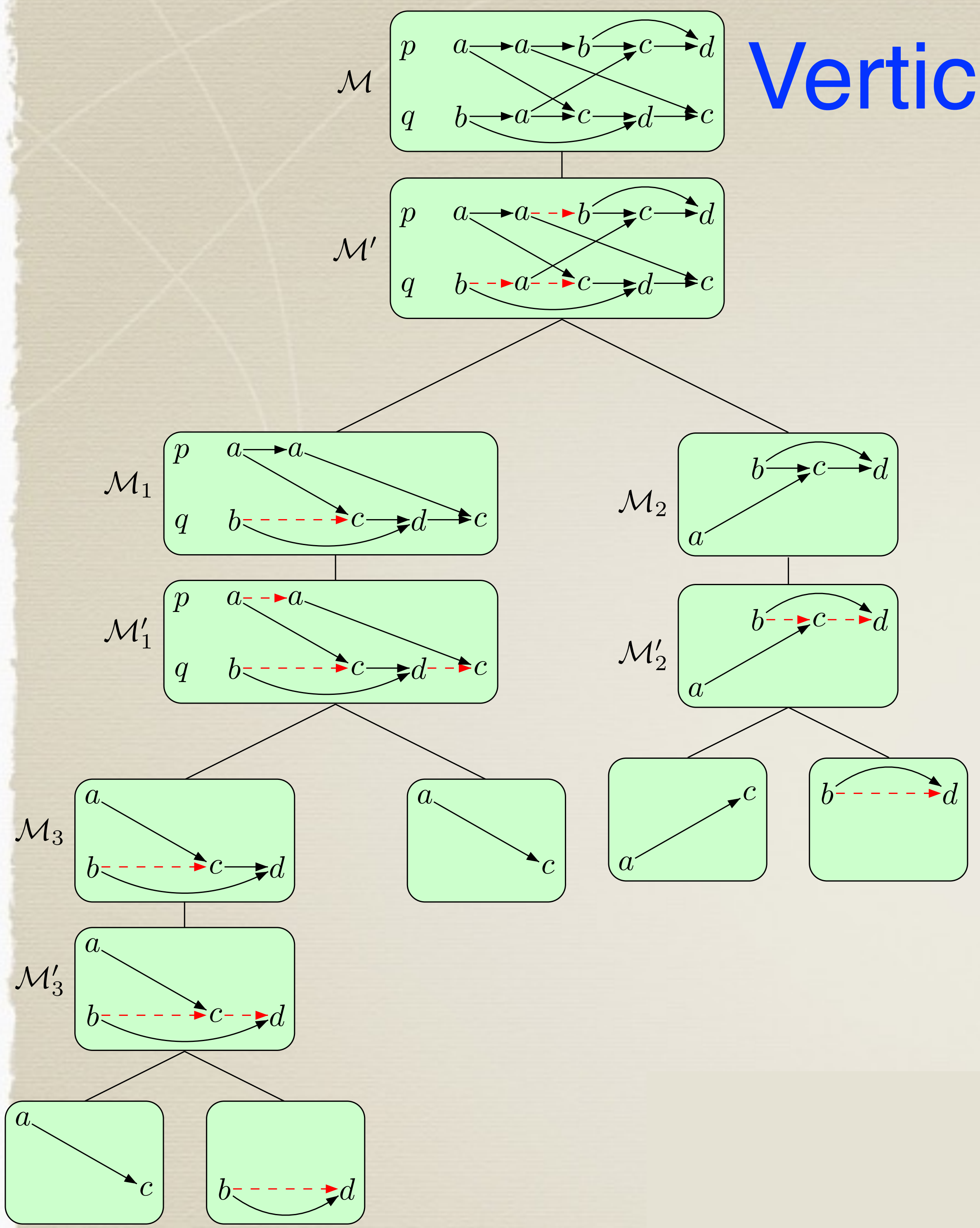
SPLIT DECOMPOSITION OF CBMs

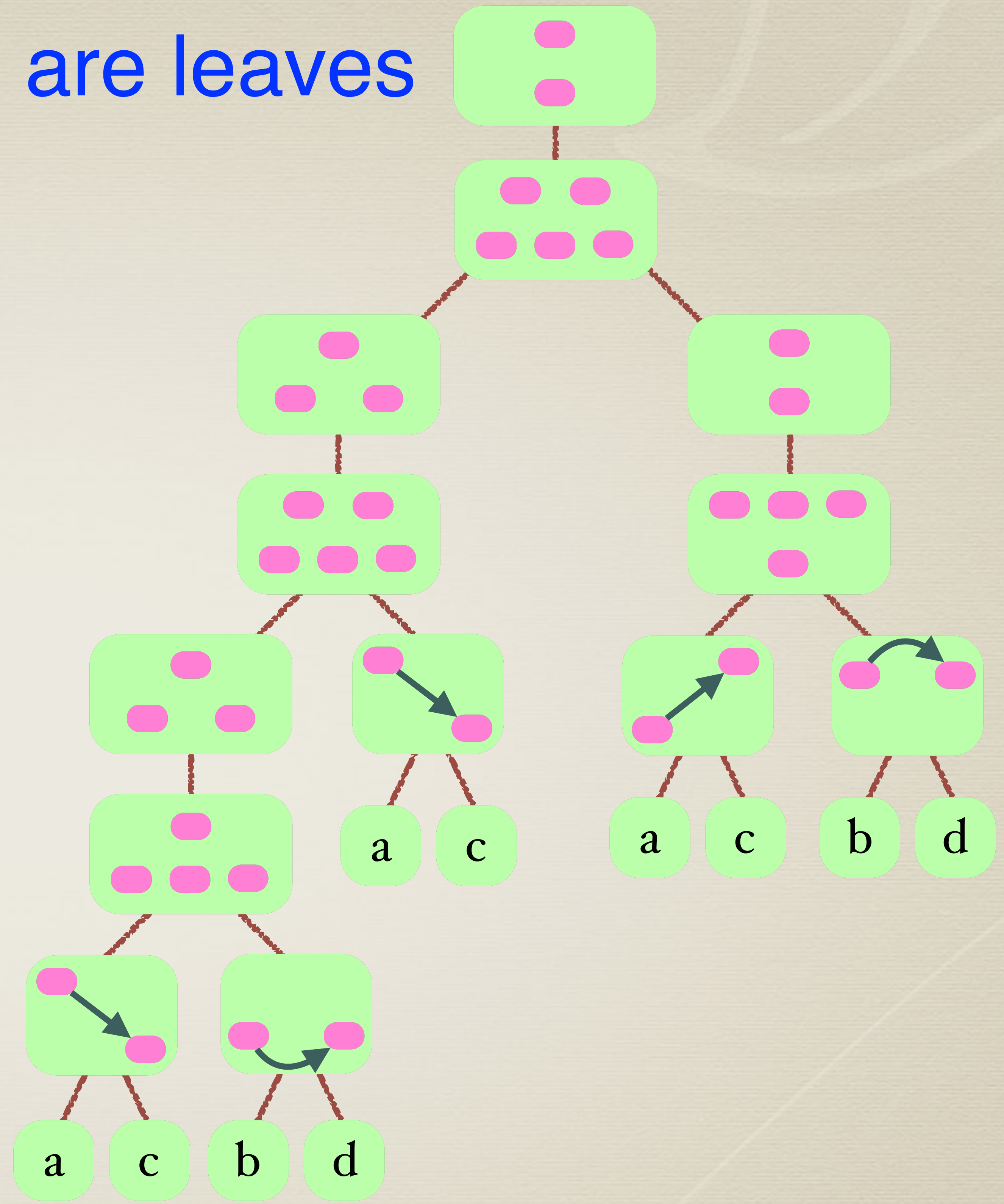


SPLIT TREE

OF THE FULL DECOMPOSITION

Vertices are leaves

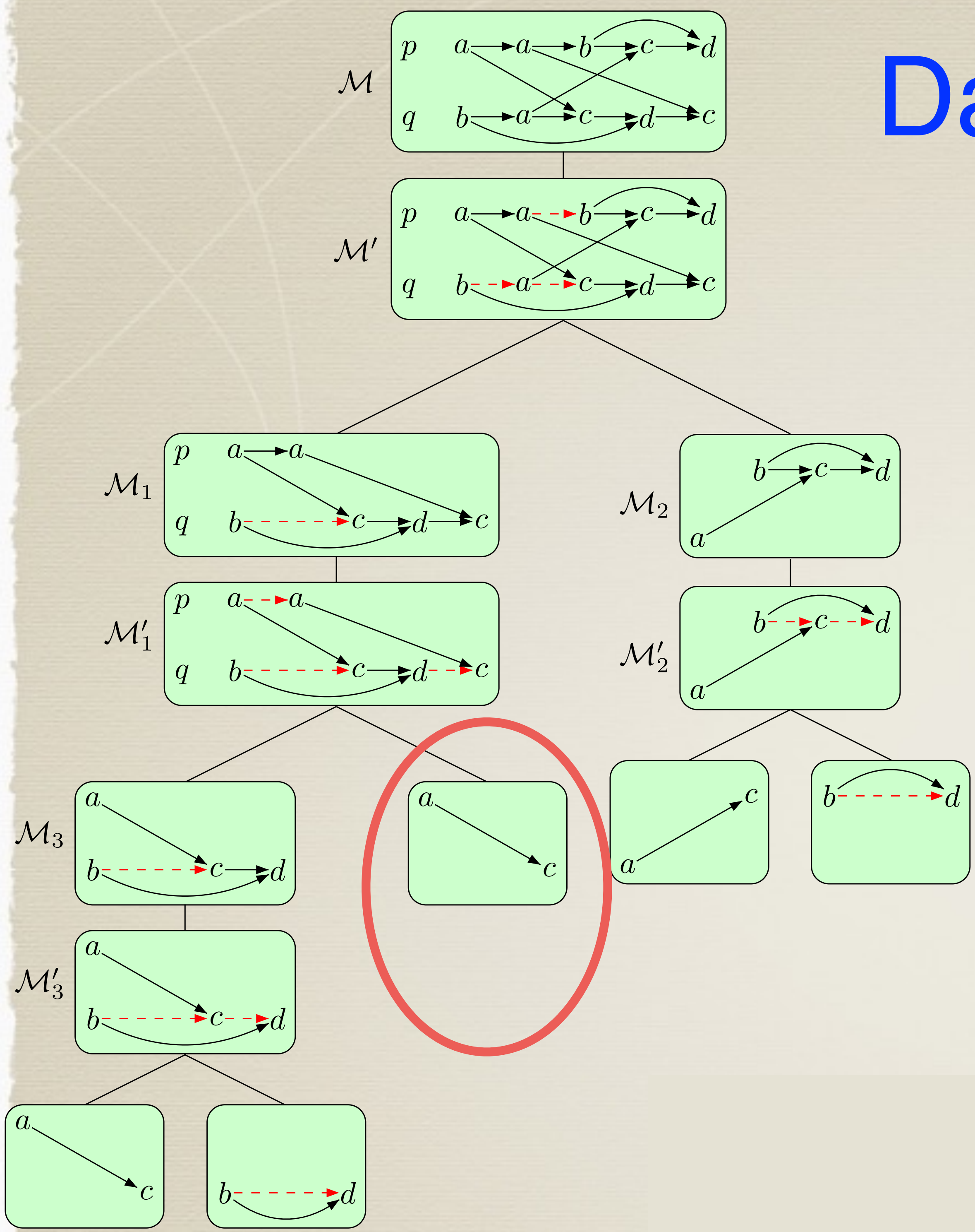


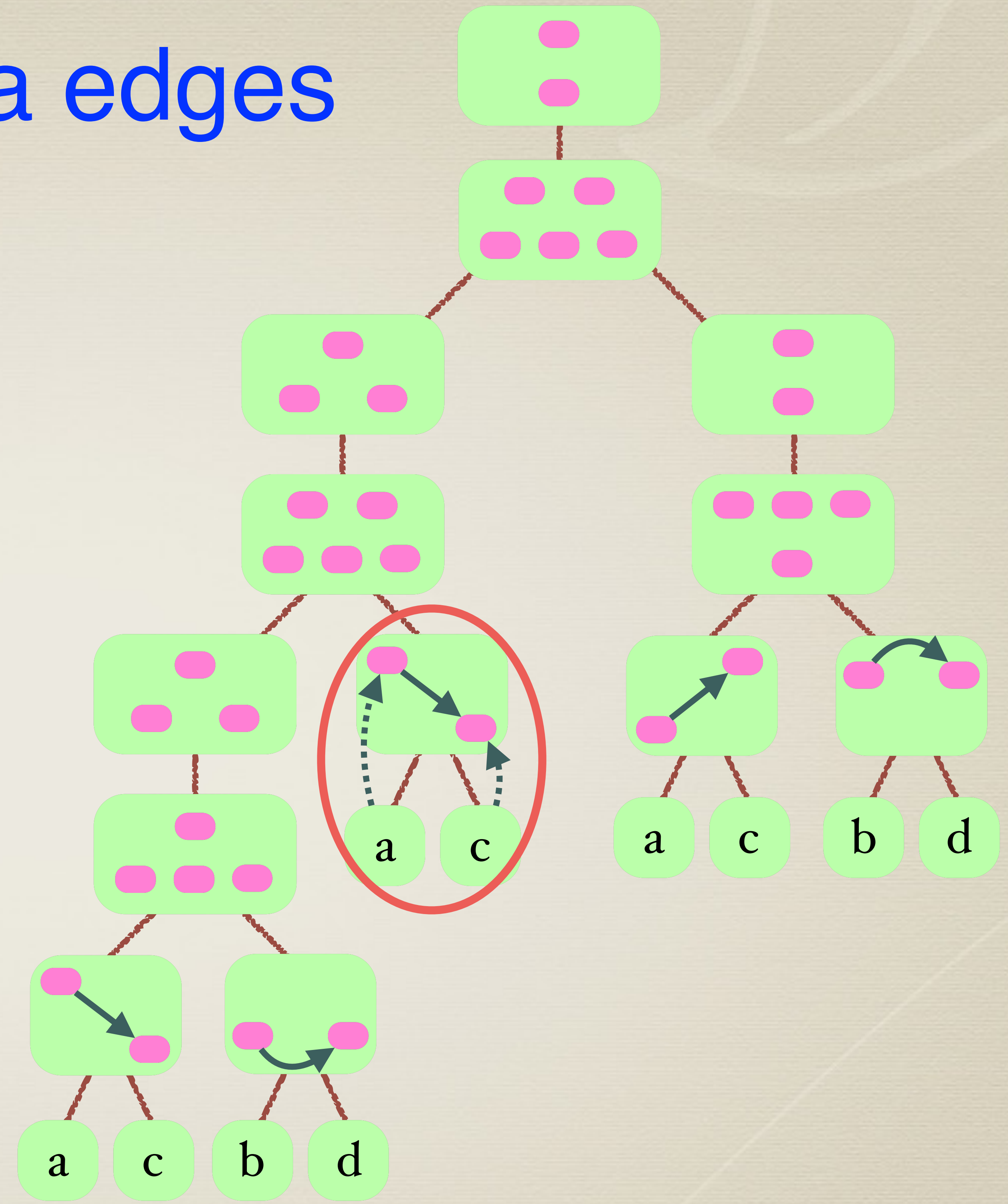


Tree interpretation in

Abstract Tree Decomposition

Data edges

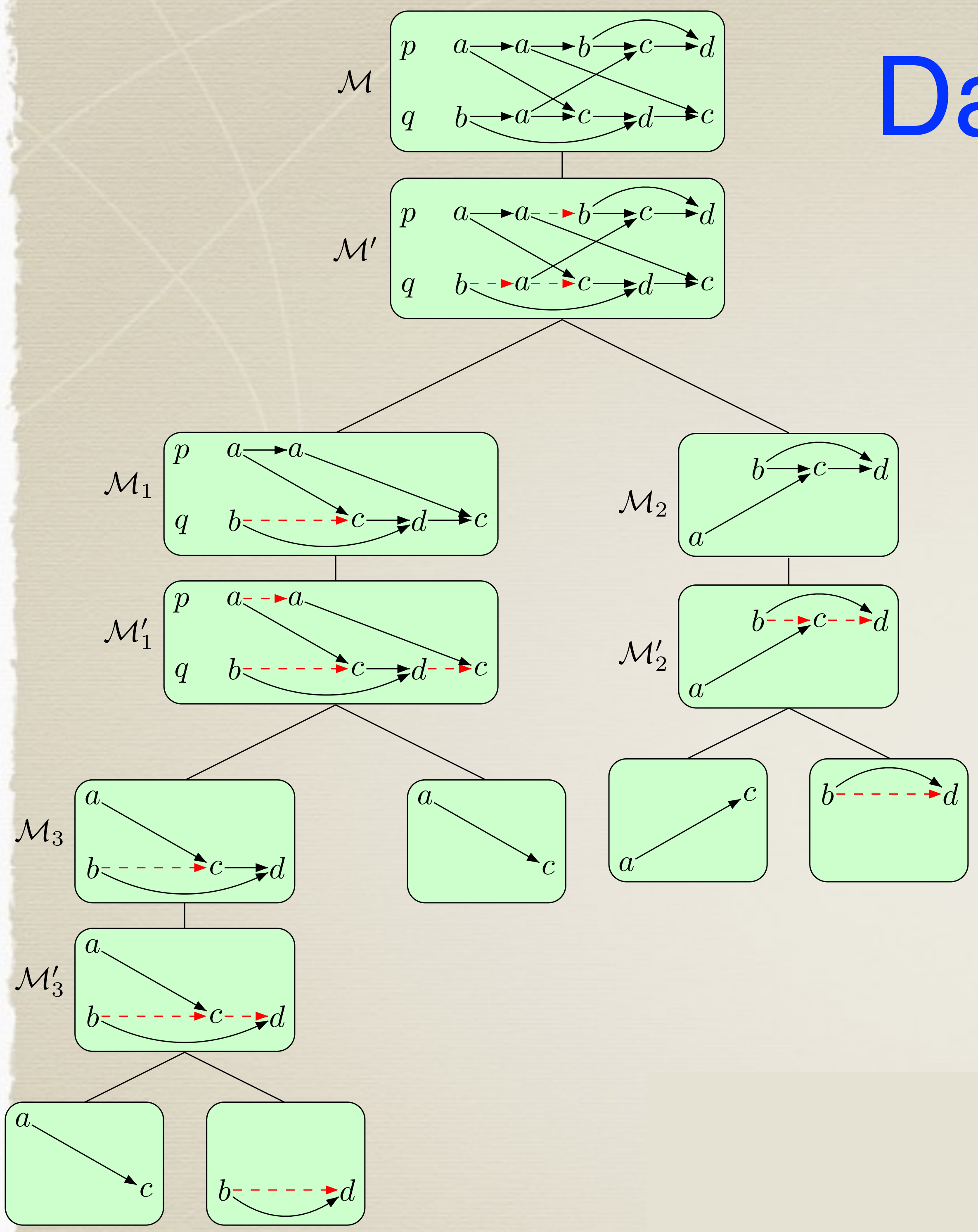


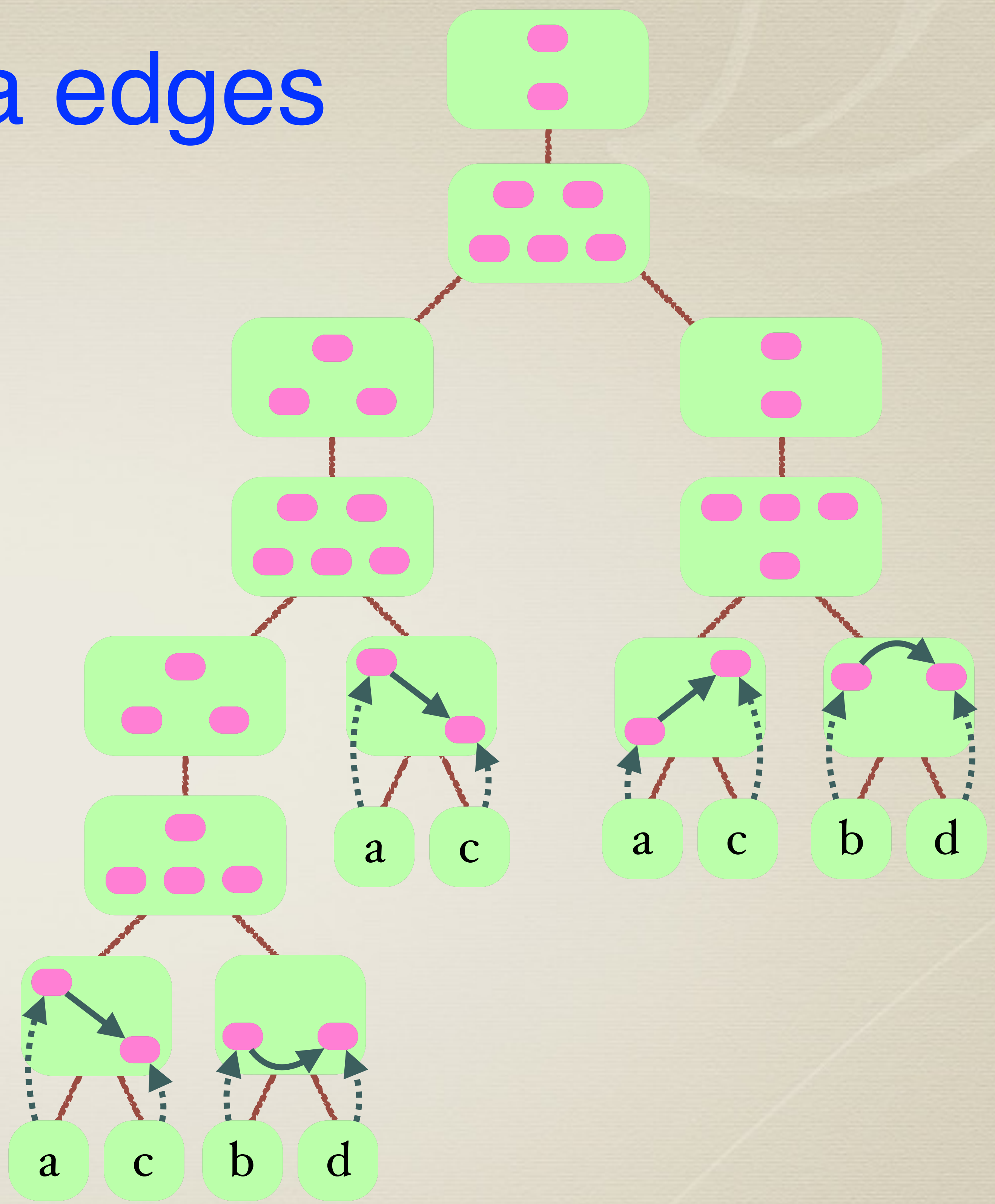


Tree interpretation in

Abstract Tree Decomposition

Data edges

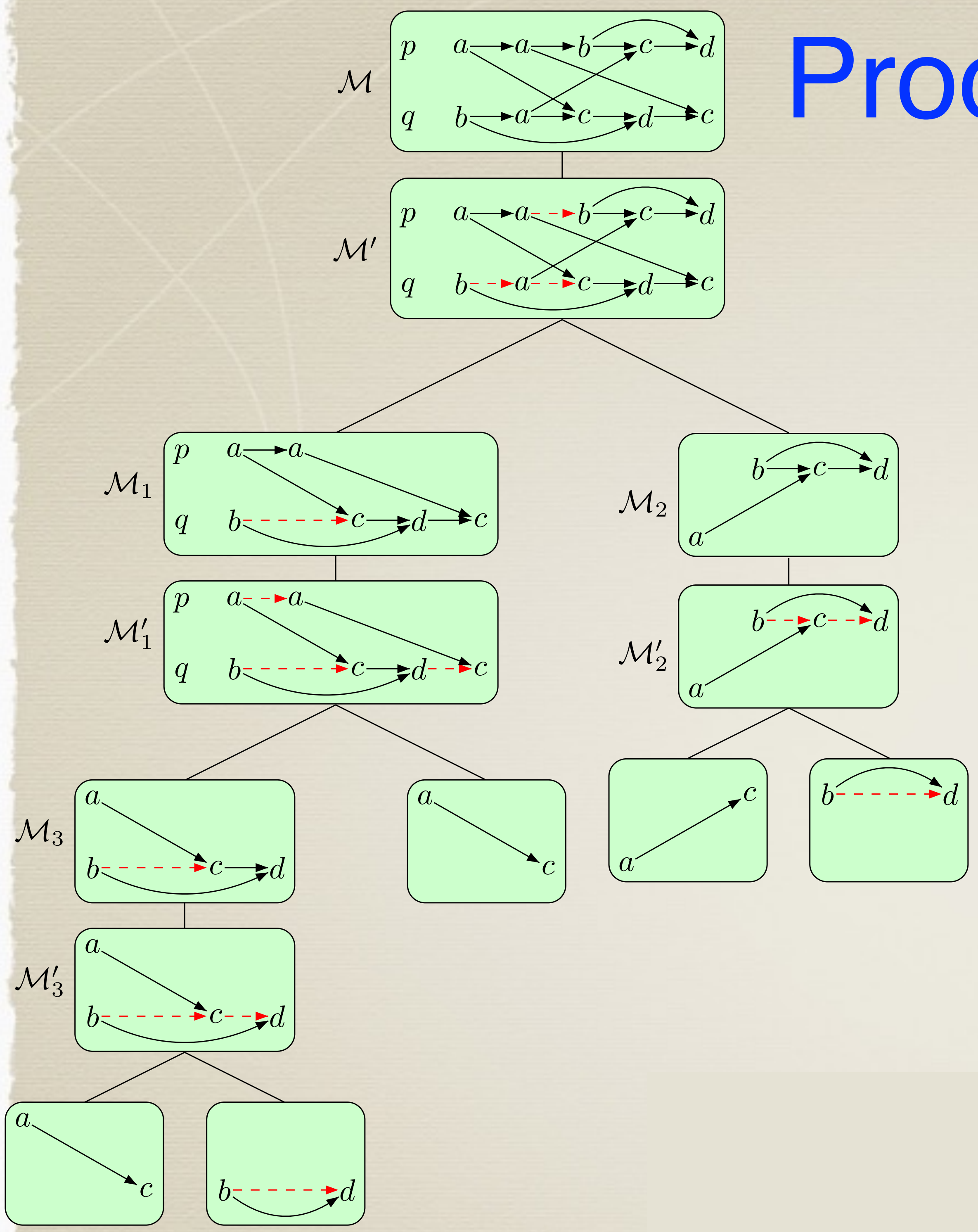


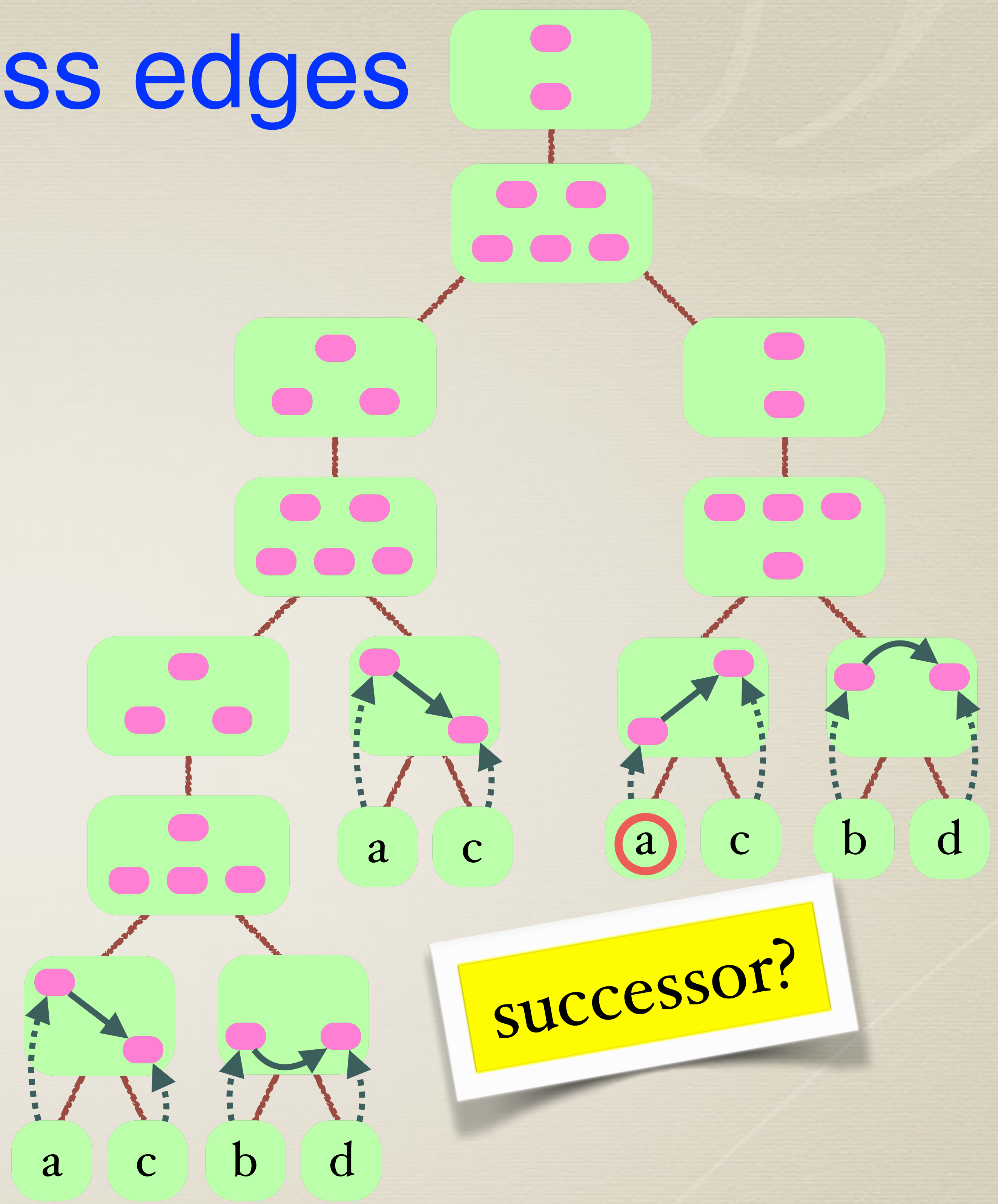


Tree interpretation in

Abstract Tree Decomposition

Process edges

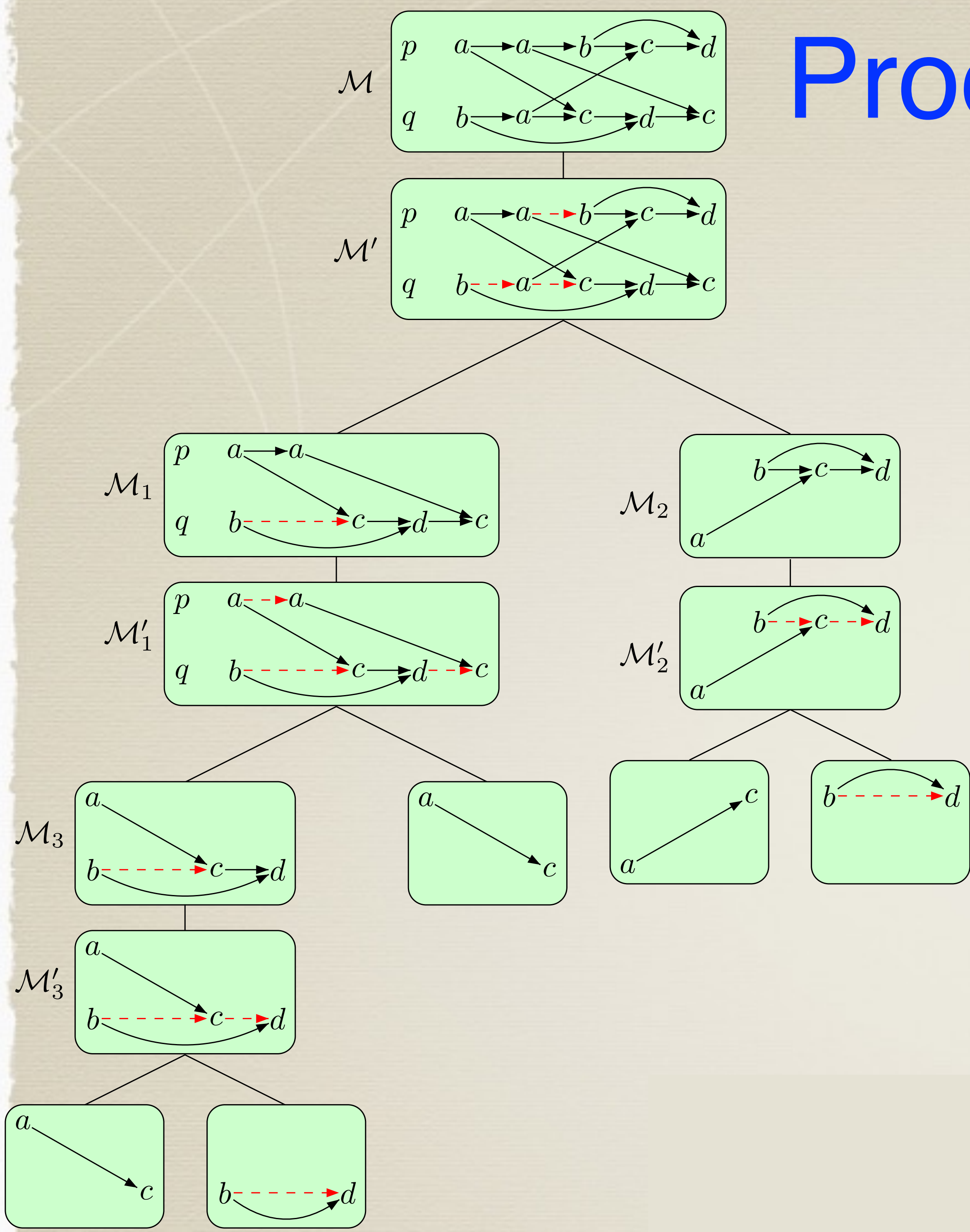


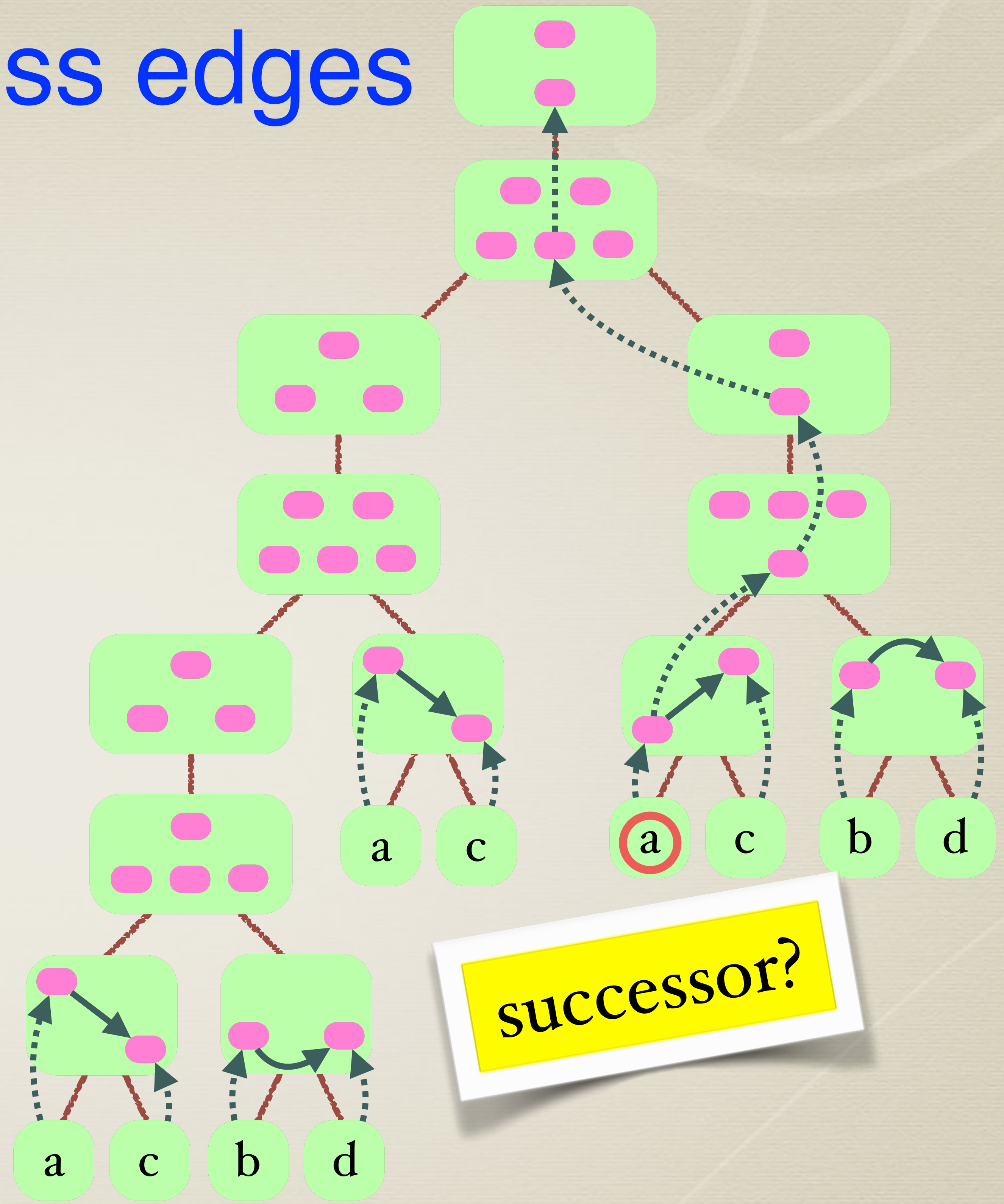


Tree interpretation in

Abstract Tree Decomposition

Process edges

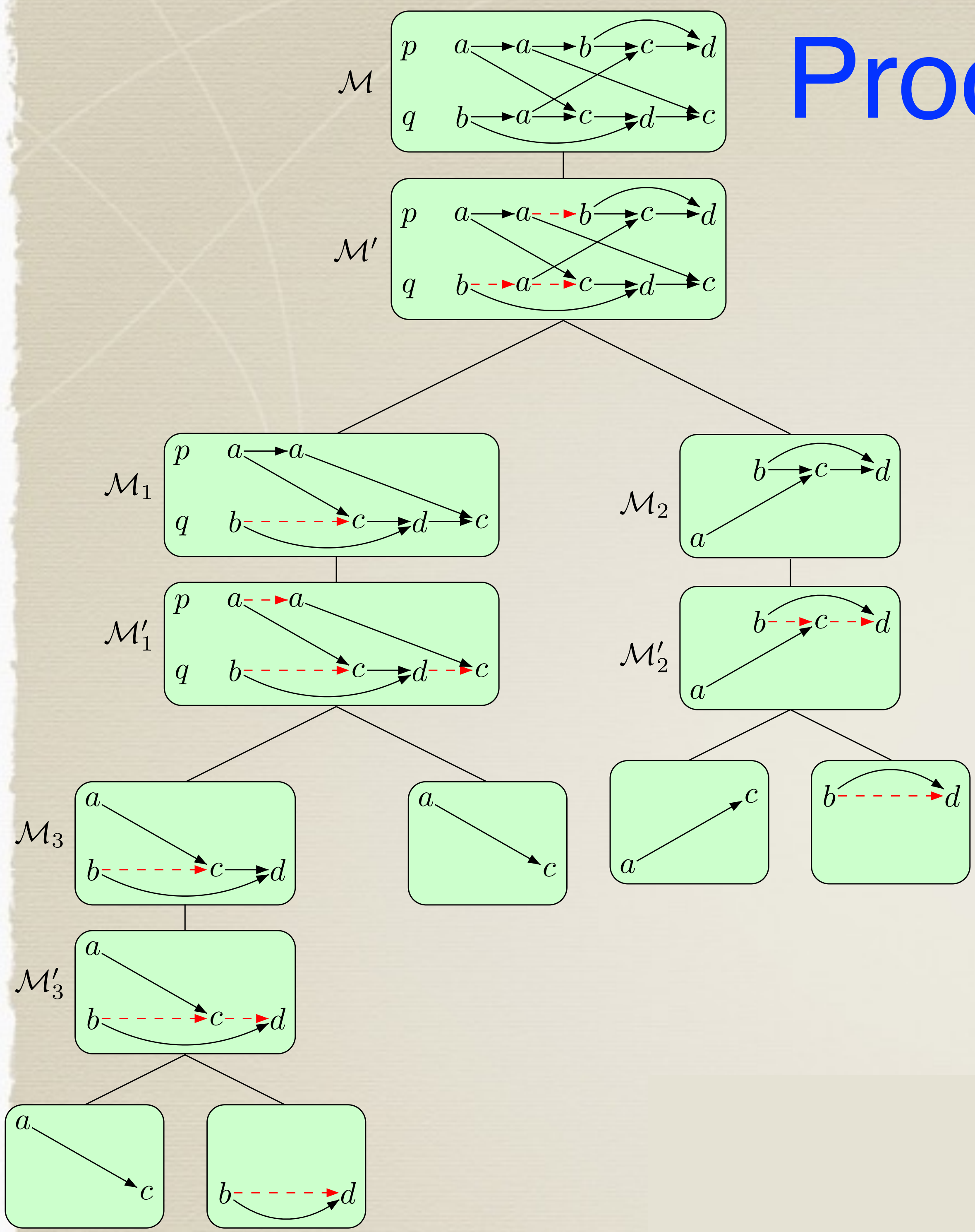


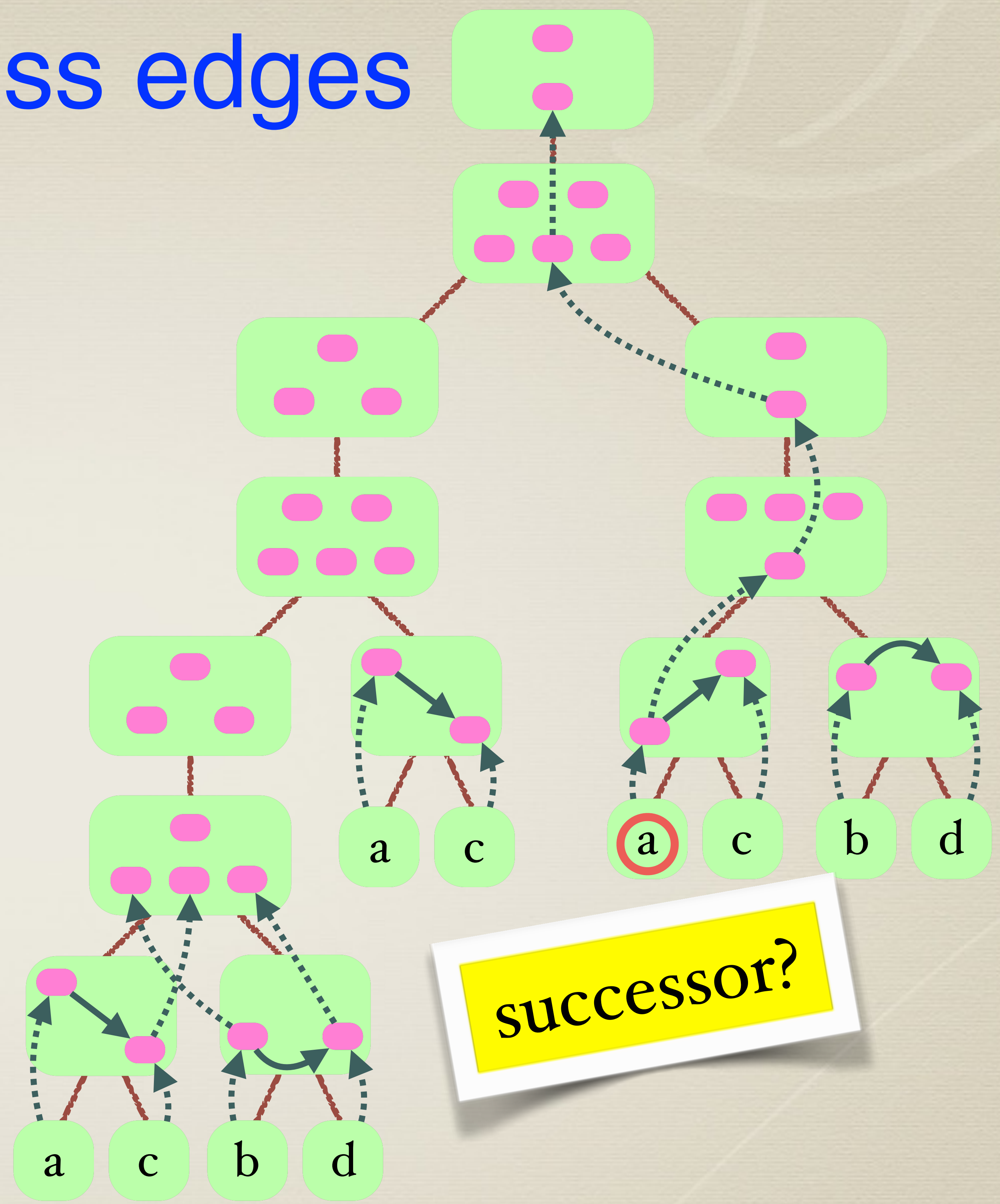


Tree interpretation in

Abstract Tree Decomposition

Process edges

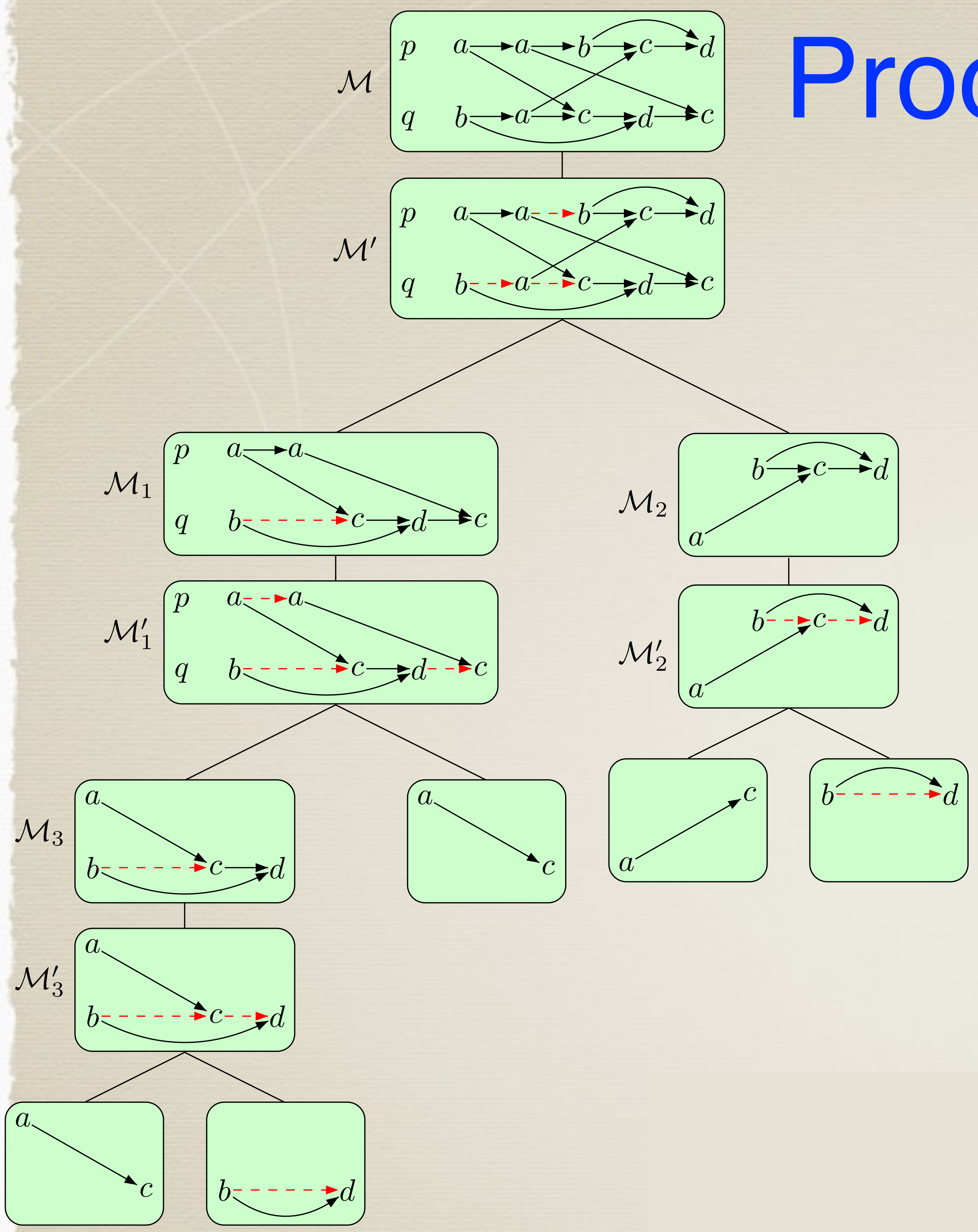


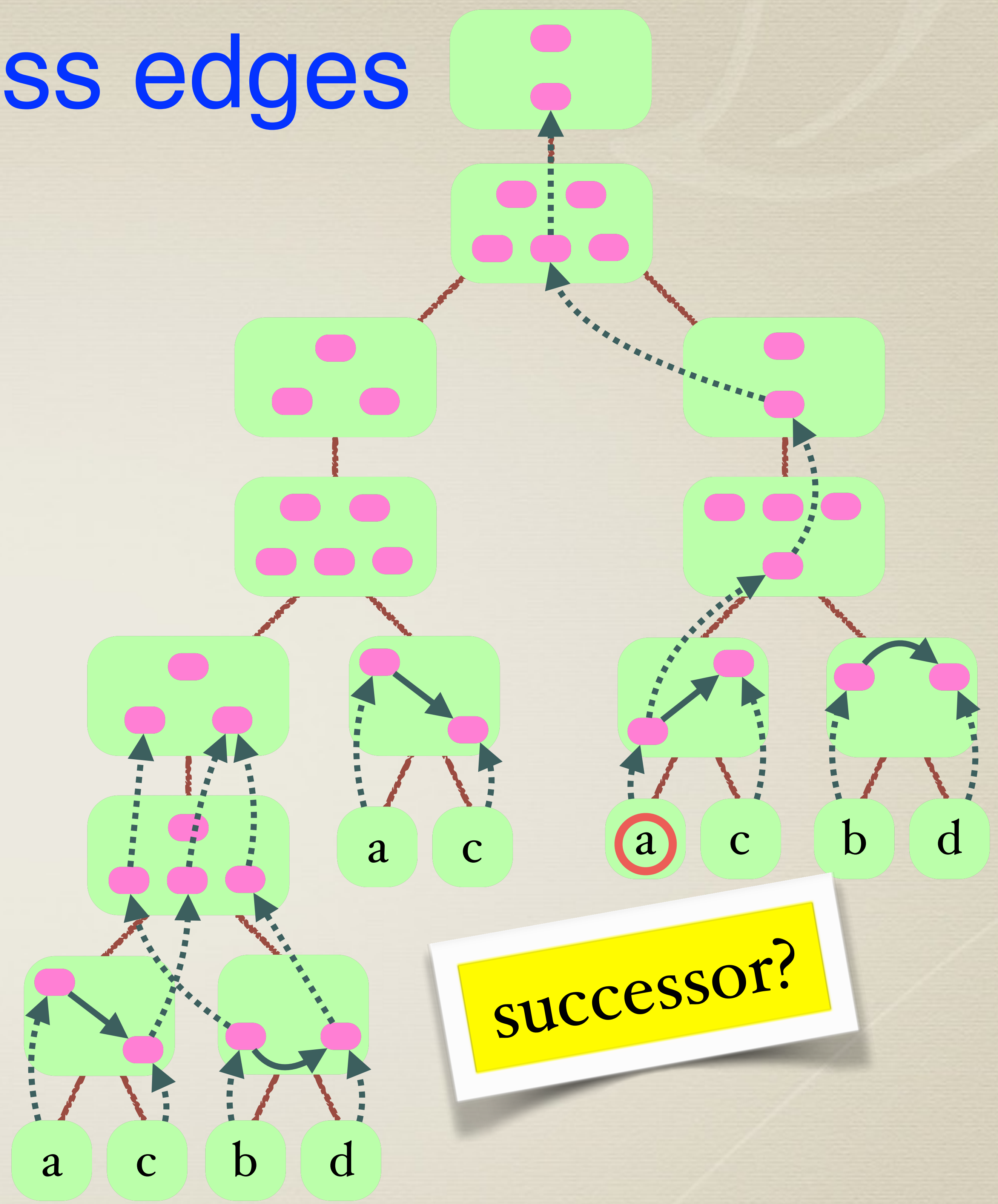


Tree interpretation in

Abstract Tree Decomposition

Process edges

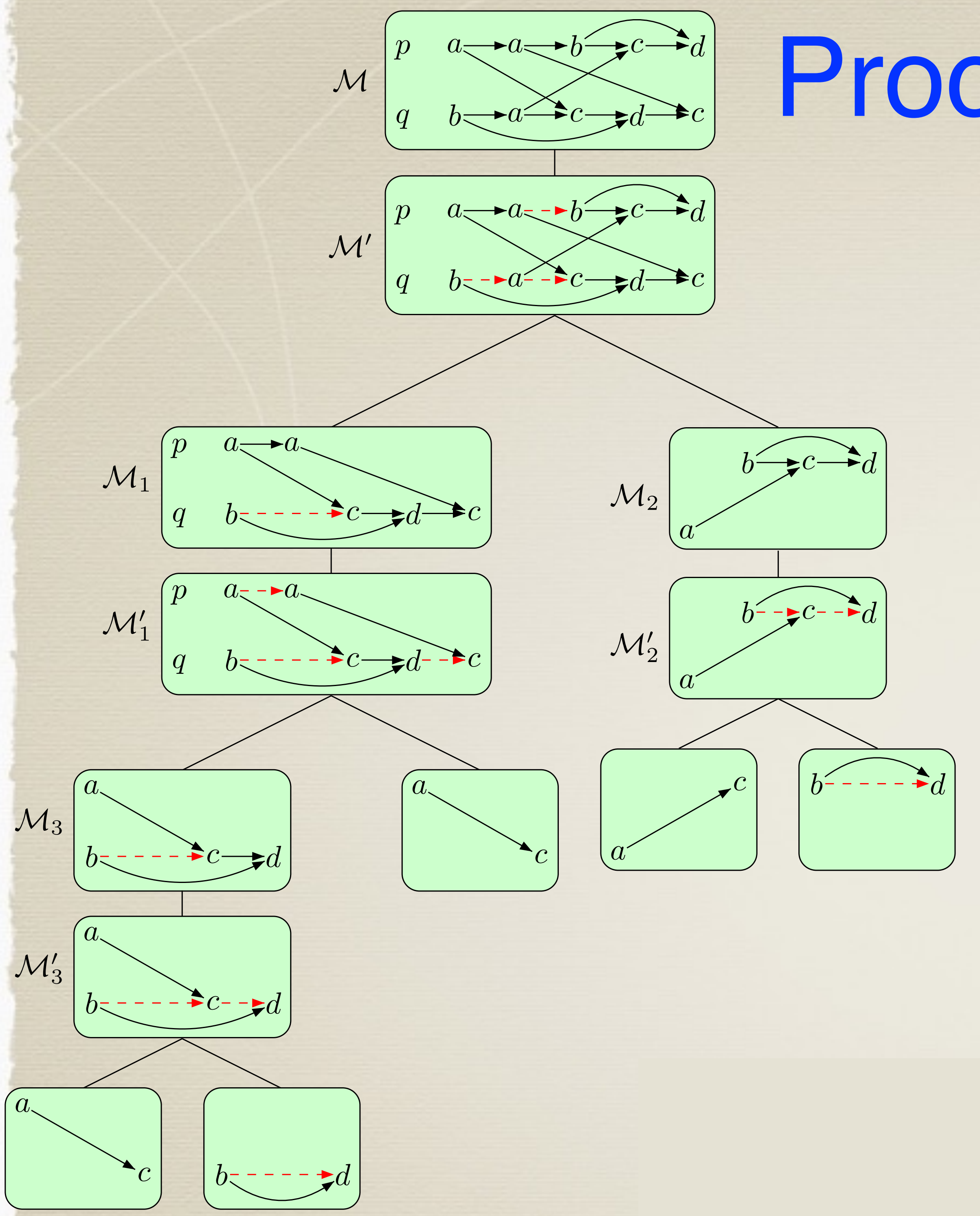


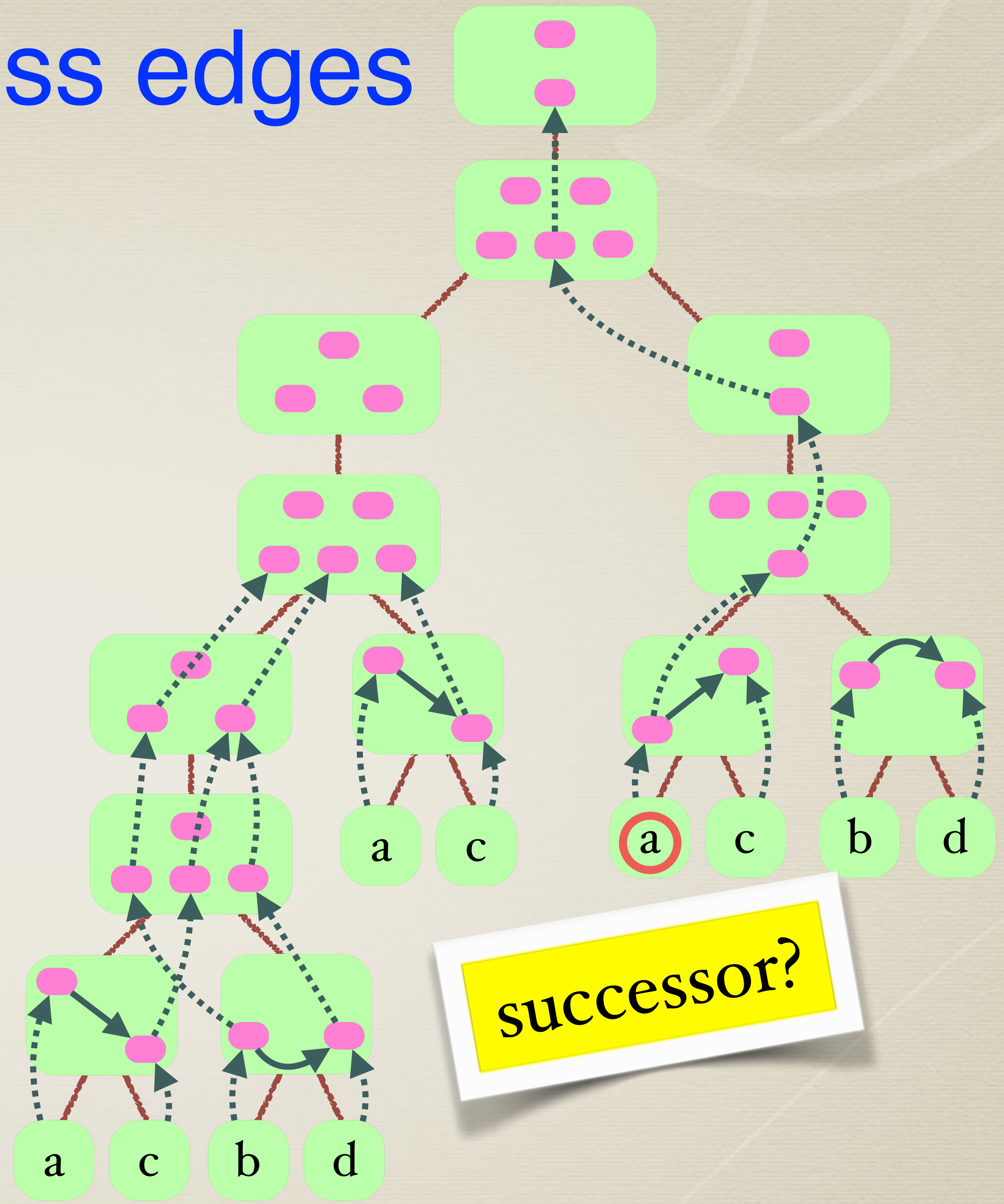


Tree interpretation in

Abstract Tree Decomposition

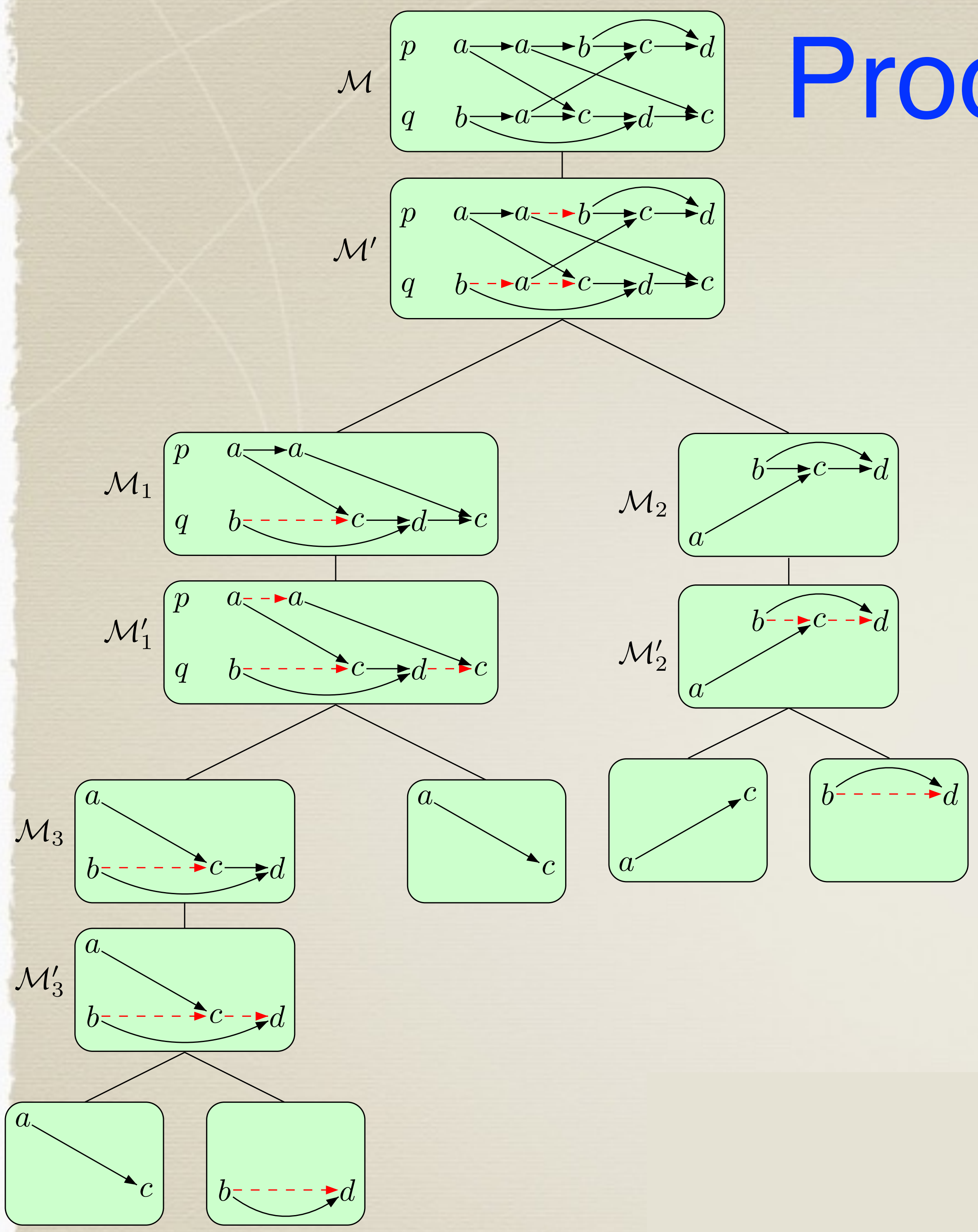
Process edges

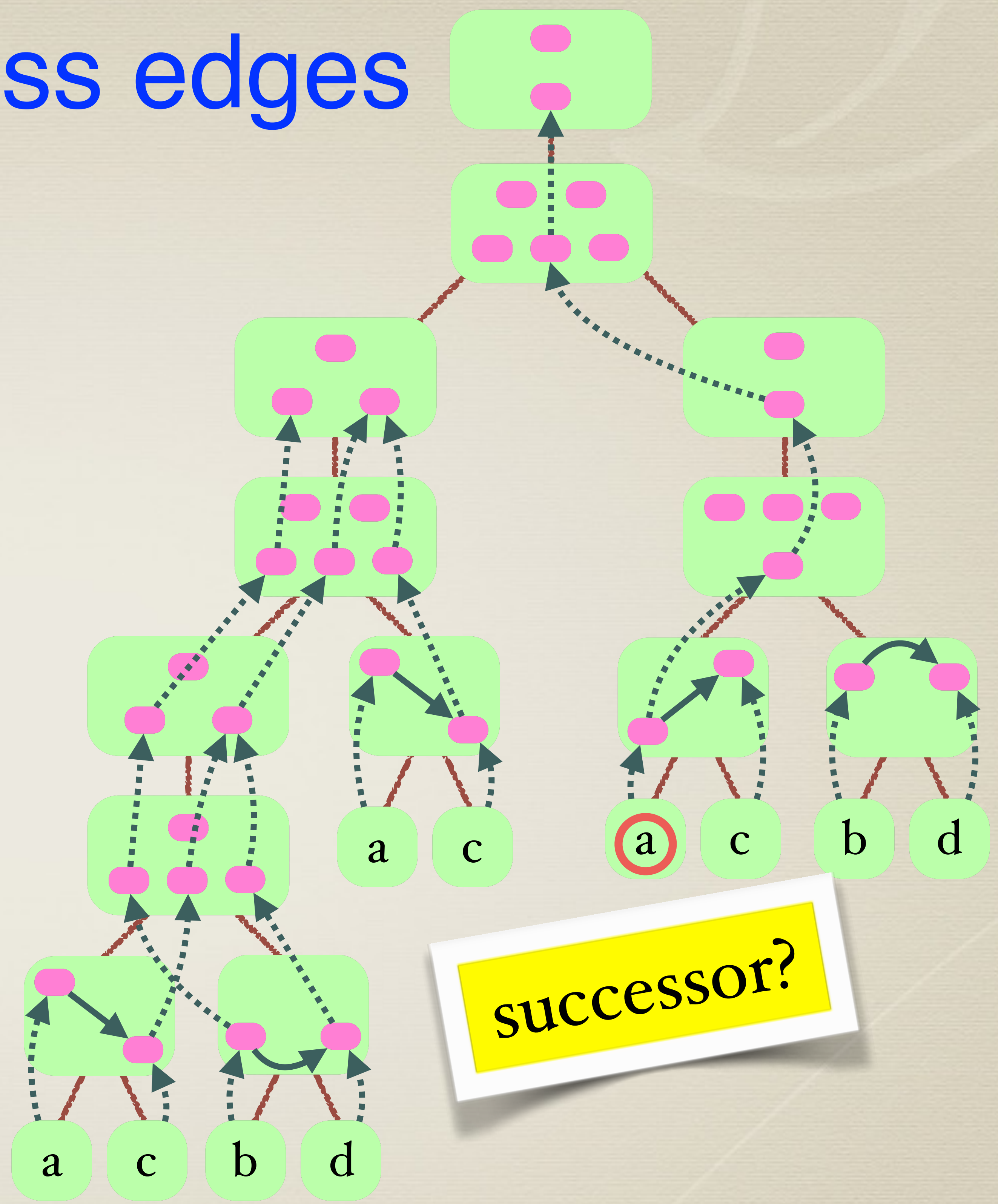




Tree interpretation in Abstract Tree Decomposition

Process edges

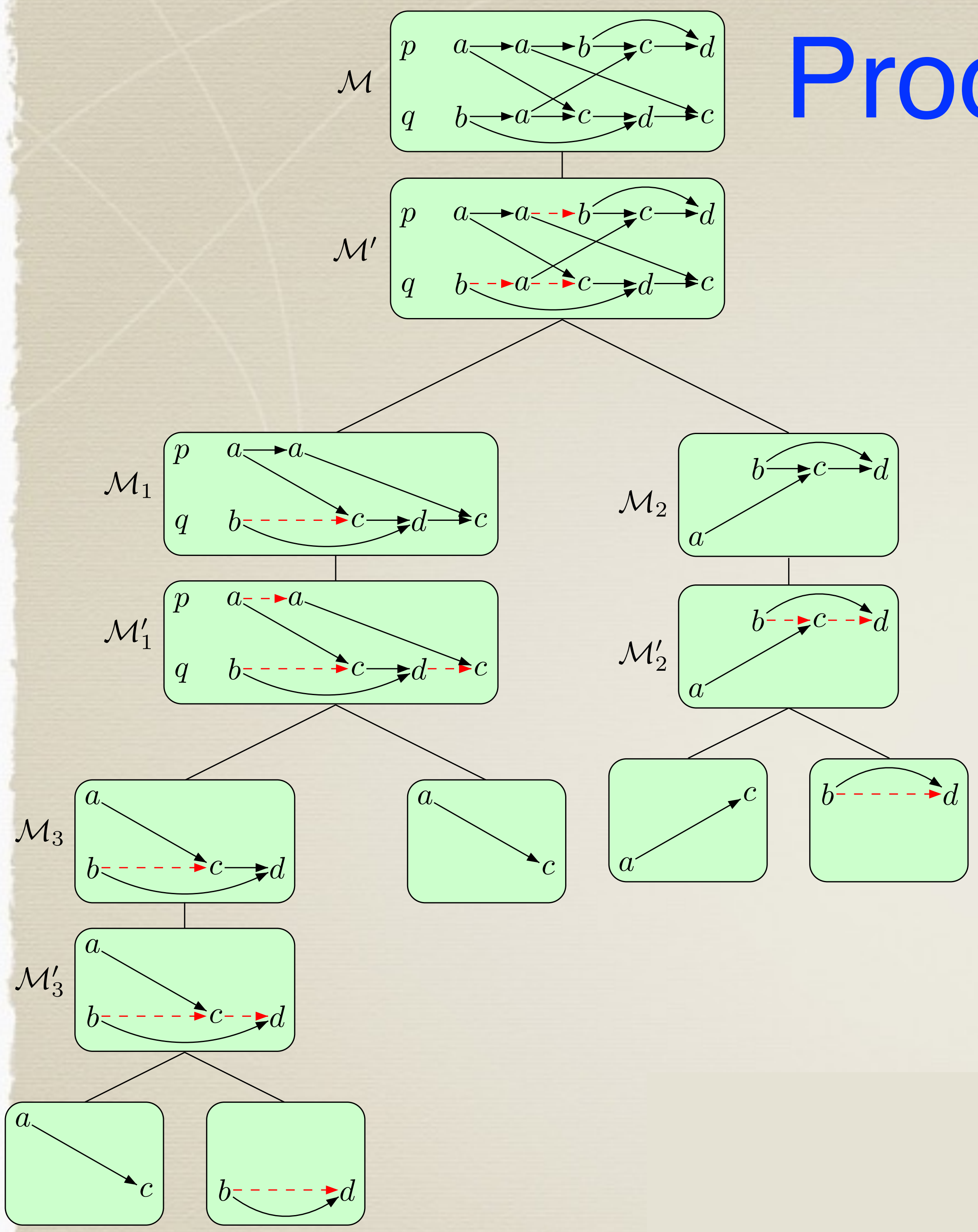


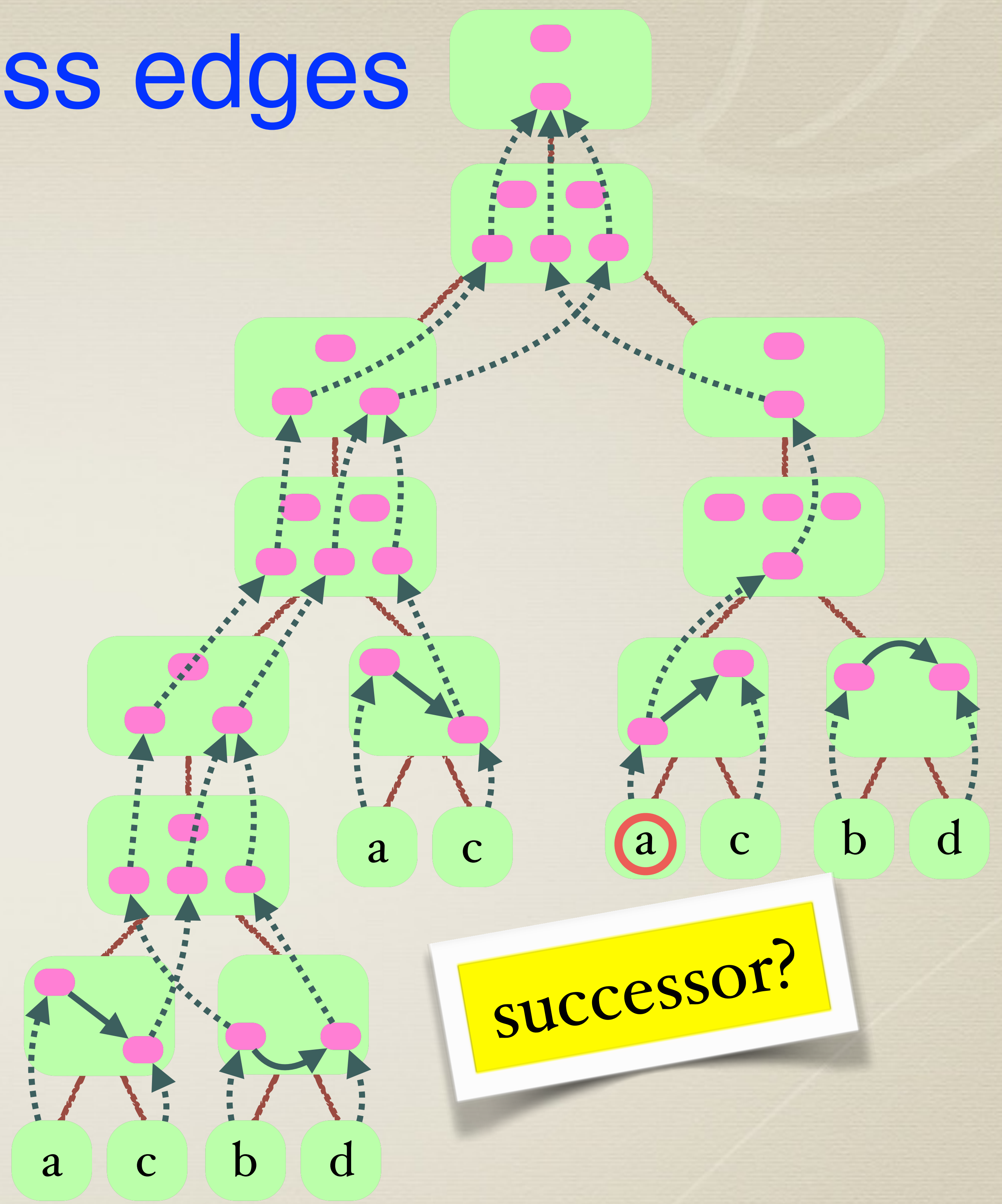


Tree interpretation in

Abstract Tree Decomposition

Process edges

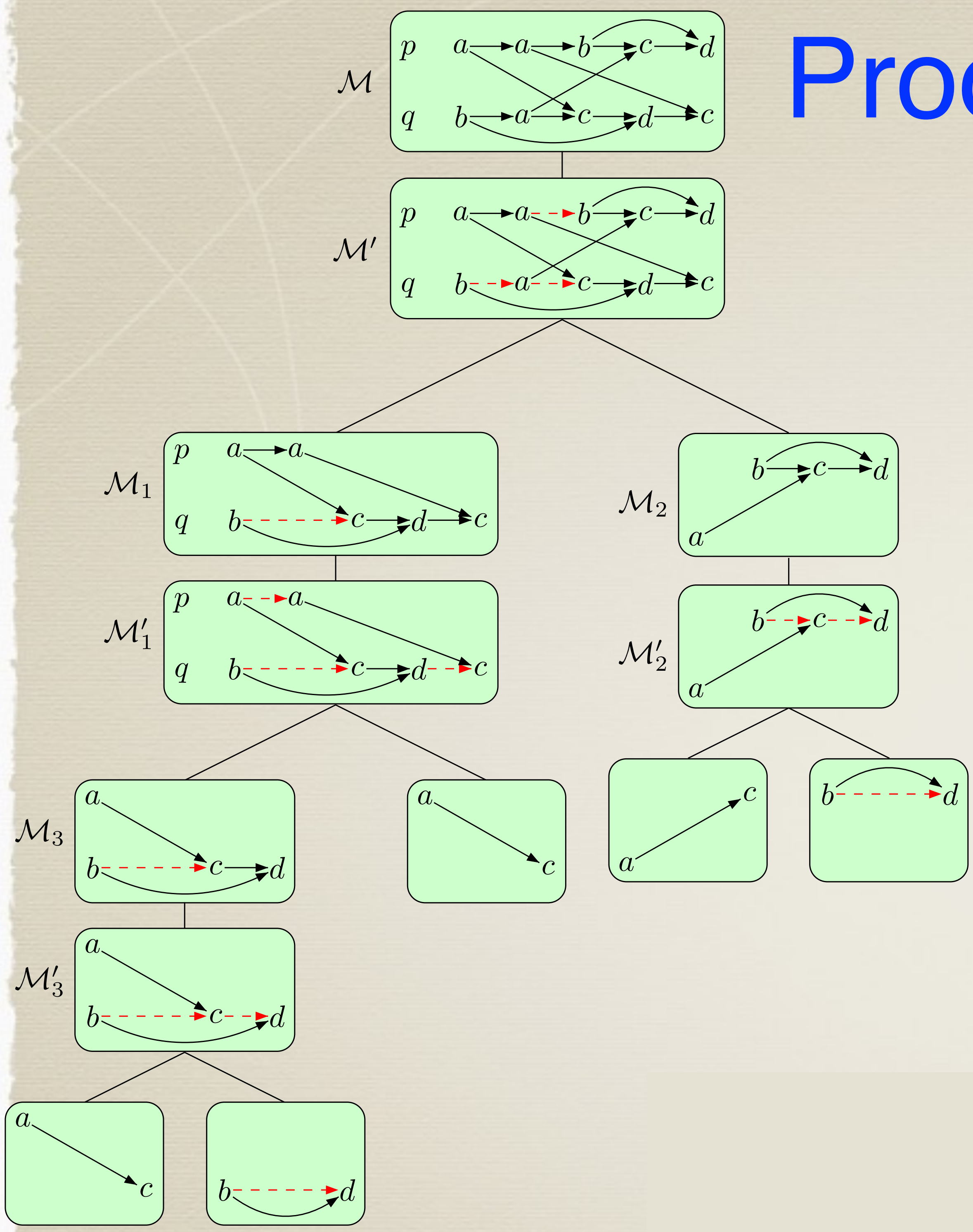


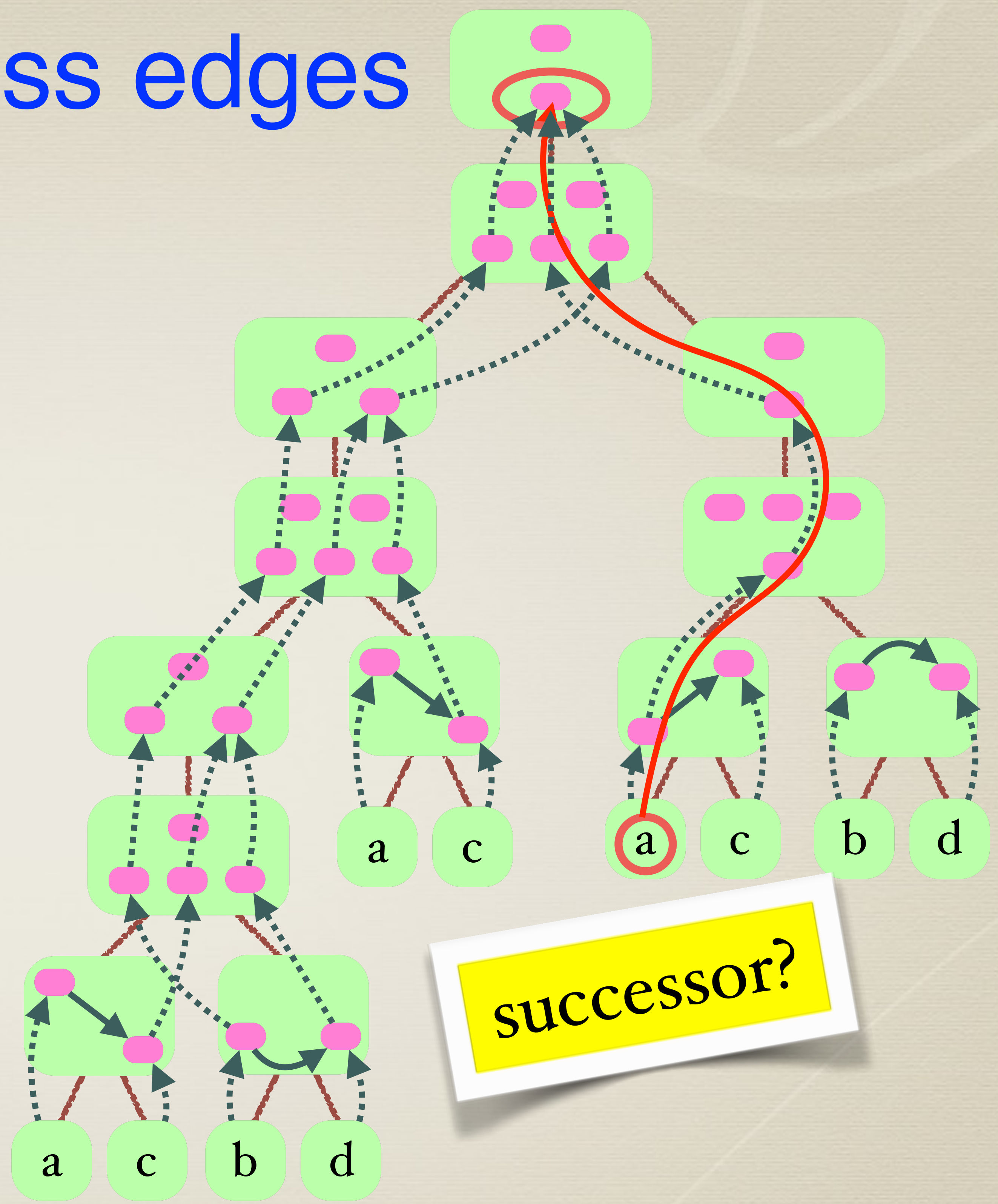


Tree interpretation in

Abstract Tree Decomposition

Process edges

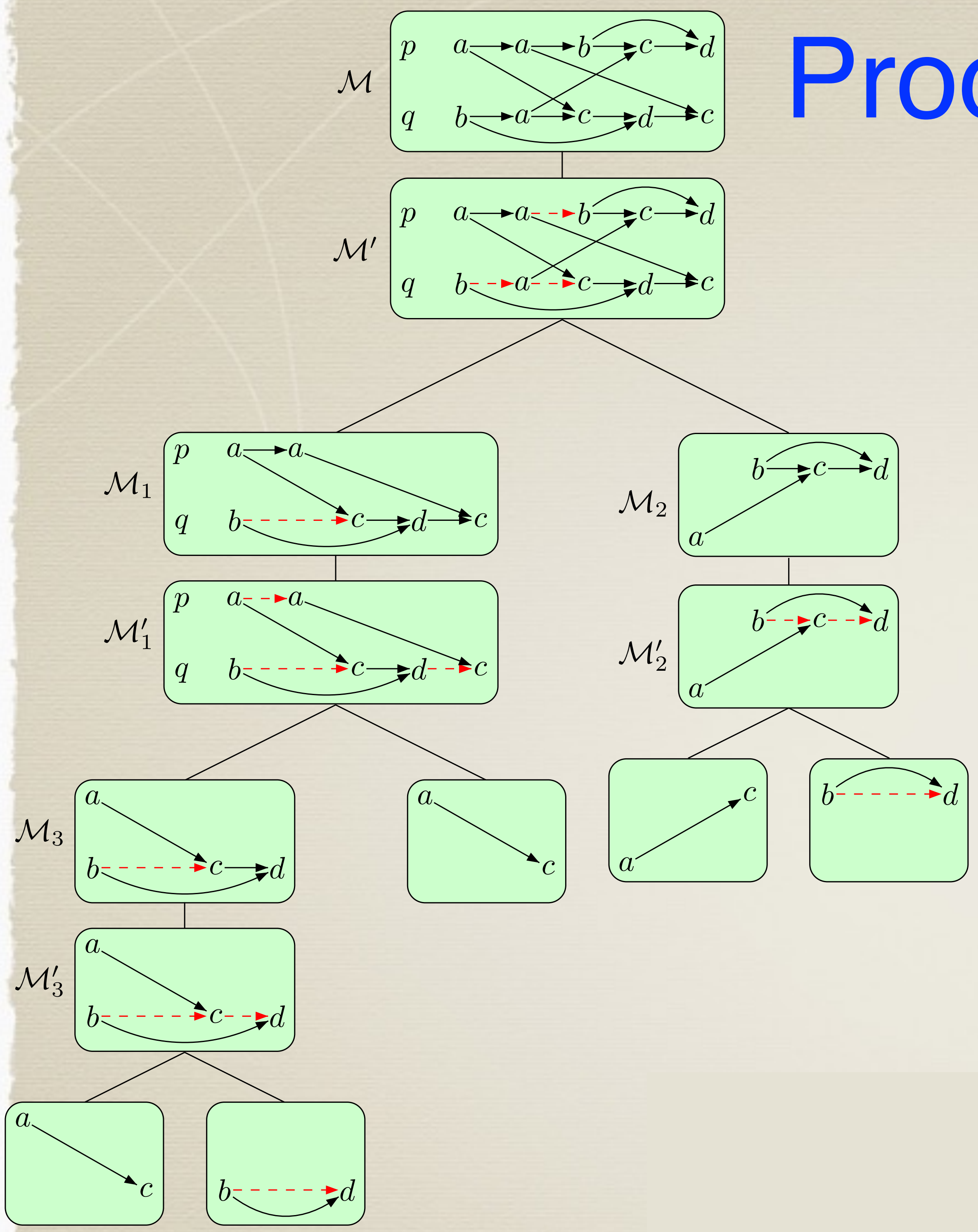


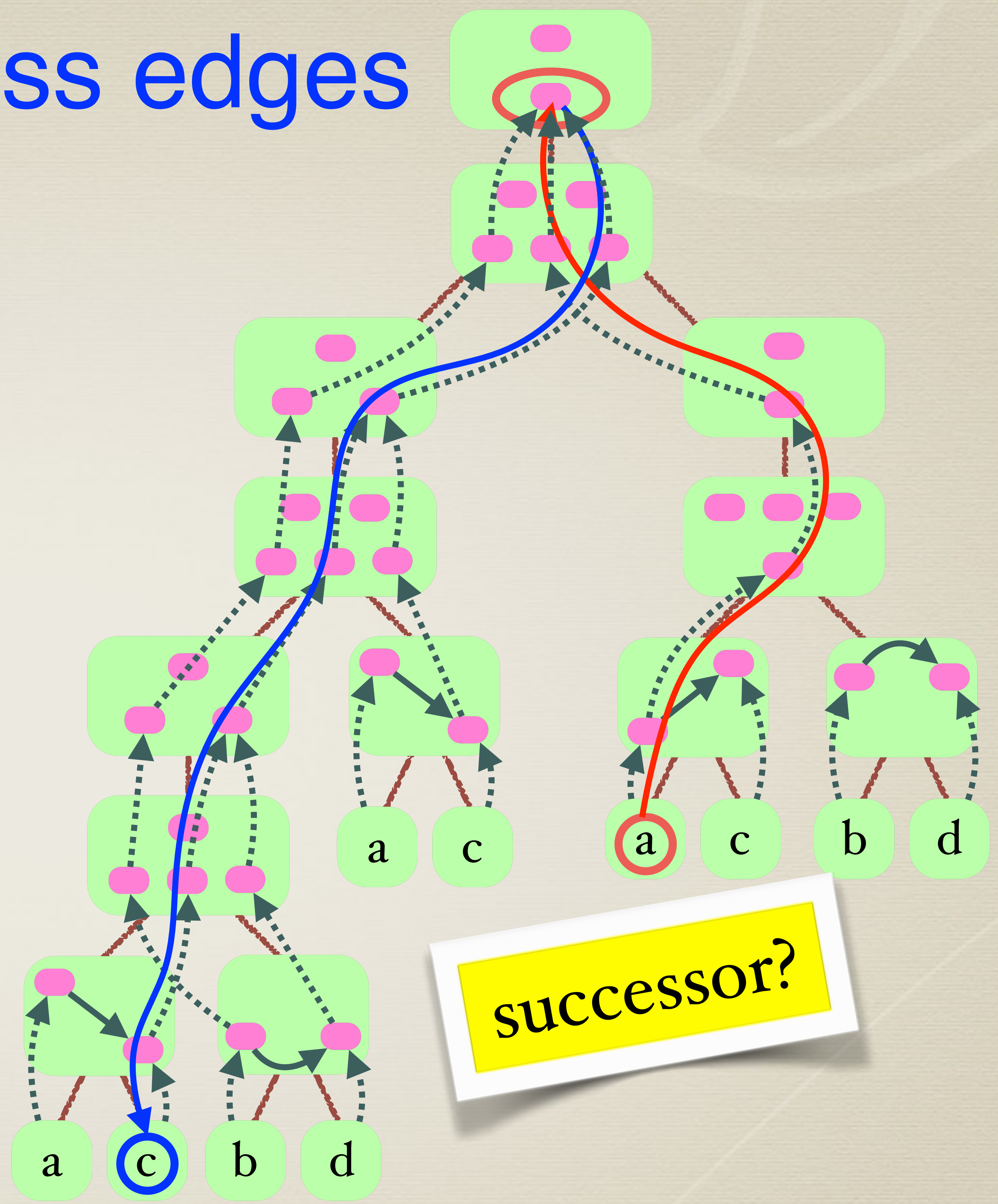


Tree interpretation in

Abstract Tree Decomposition

Process edges

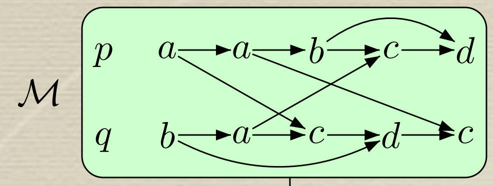




Tree interpretation in

Abstract Tree Decomposition

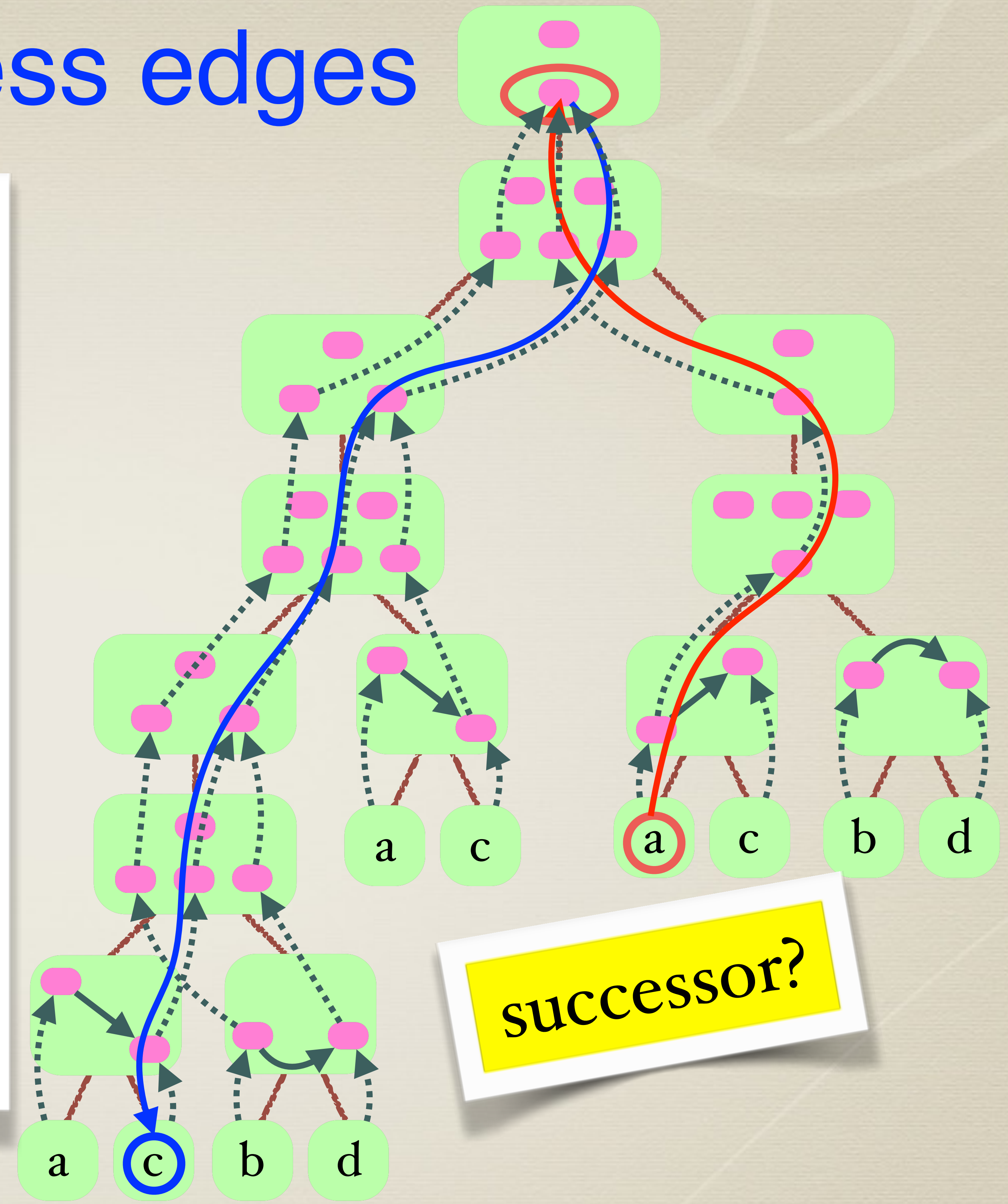
Process edges



In the abstract tree, we can interpret the graph (CBM)

- vertices and labels
- data edges
- process edges

with
tree (walking) automata
PDL or MSO formulas



Tree interpretation in

Abstract Tree Decomposition

Split-width: under-approximations

* Words
* Nested Words
* Mazurkiewicz Traces
* Acyclic Architectures

Constant

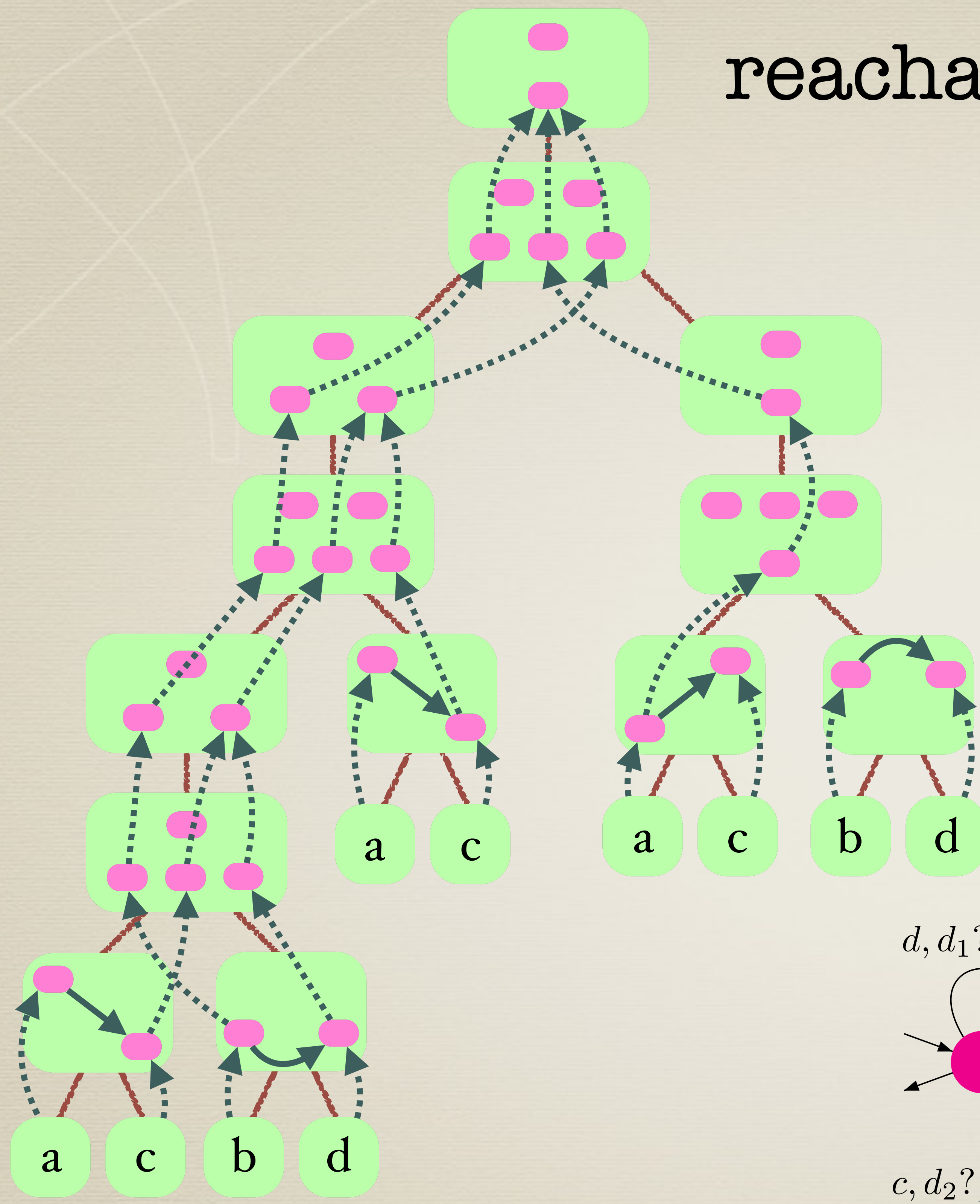
* Bounded channel size
* Existentially bounded
* Bounded context switching
* Bounded scope

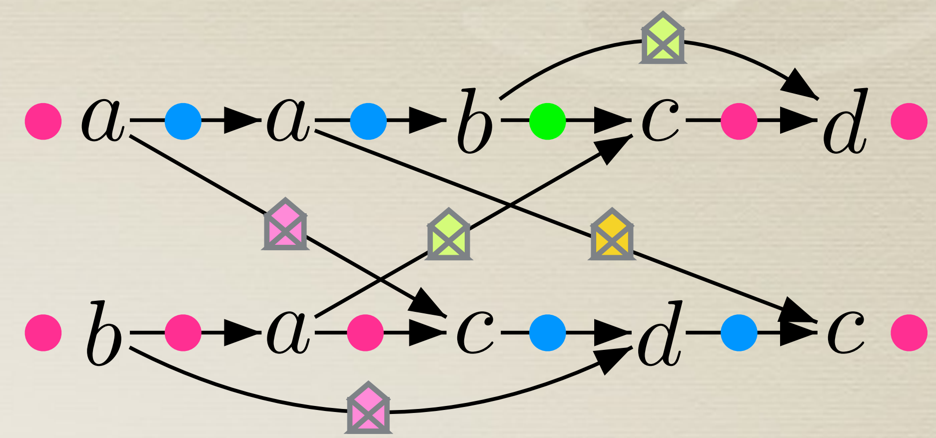
Bound + 2

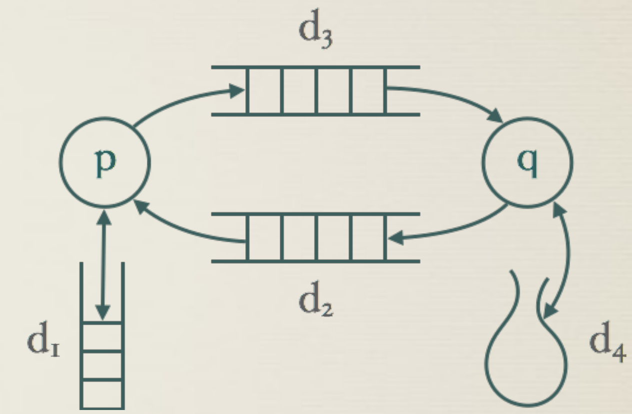
* Bounded phase
* Priority ordering

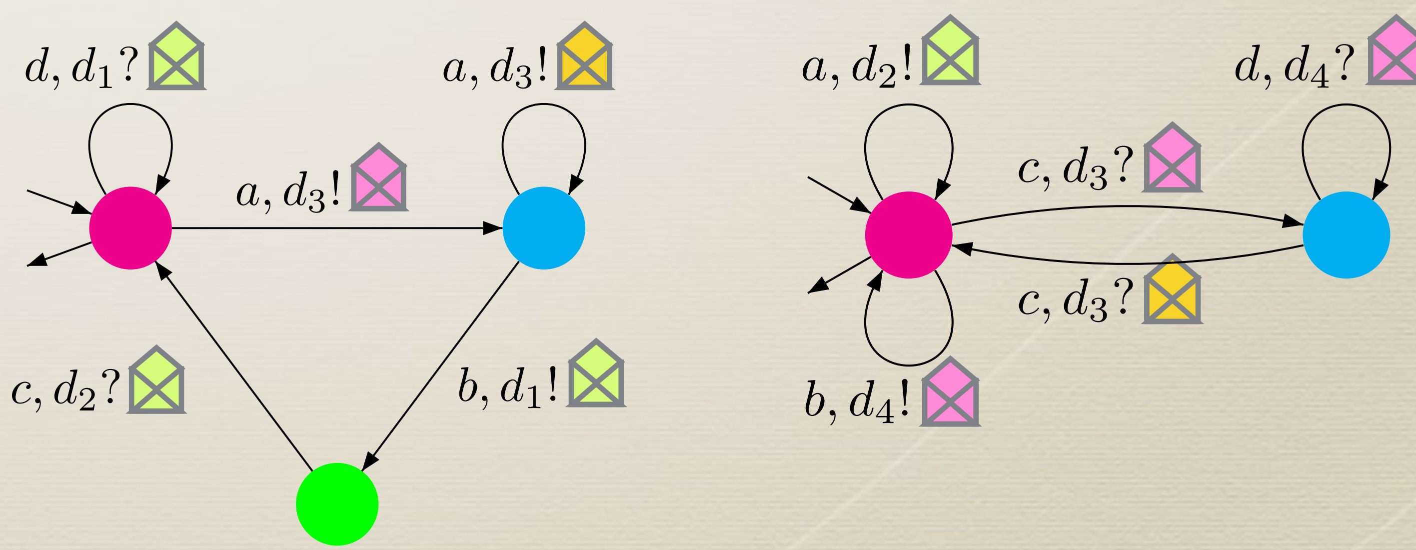
2^{Bound}

reachability/emptiness

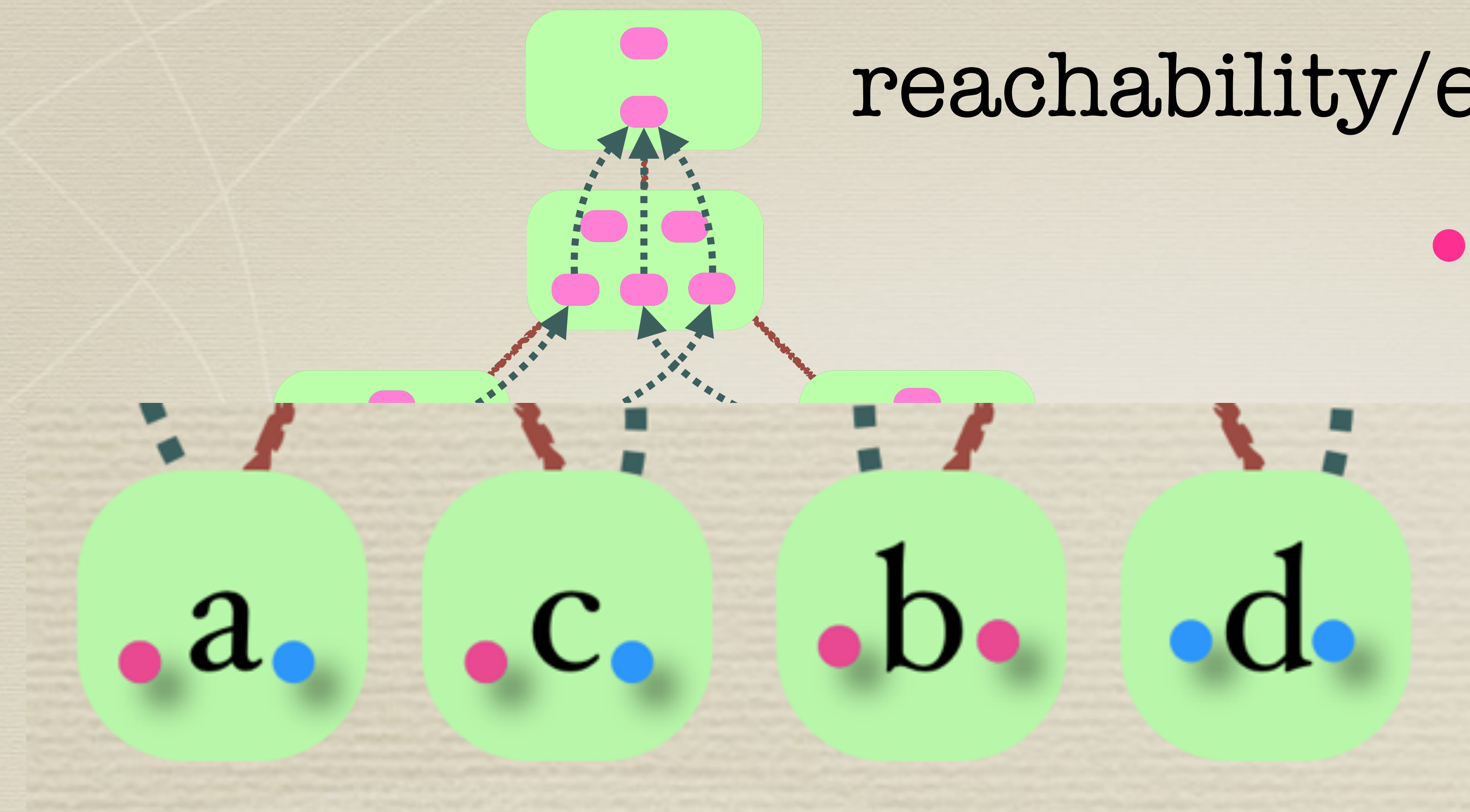


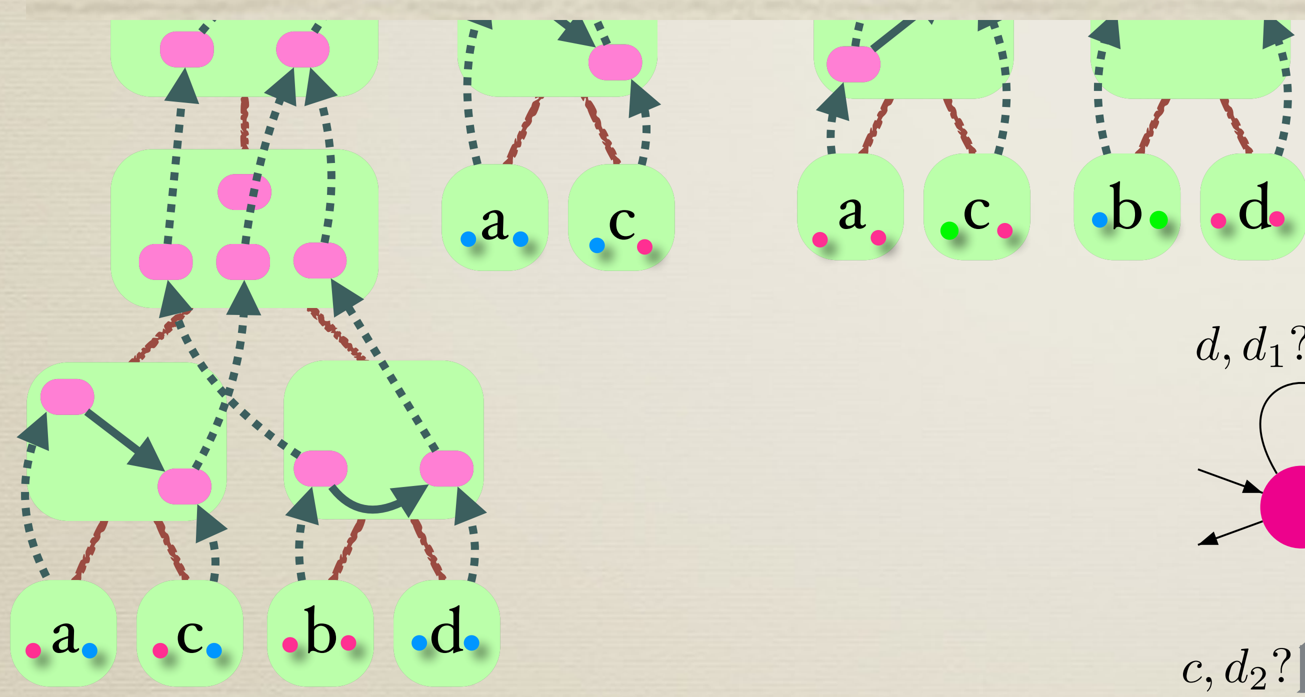


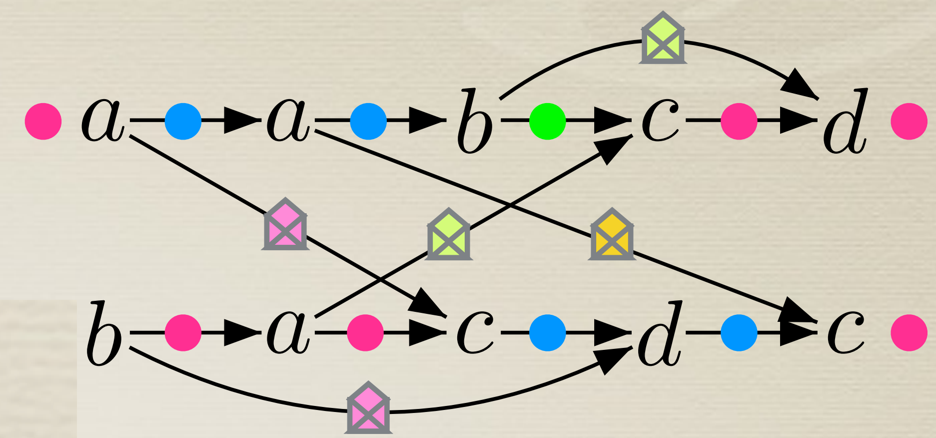


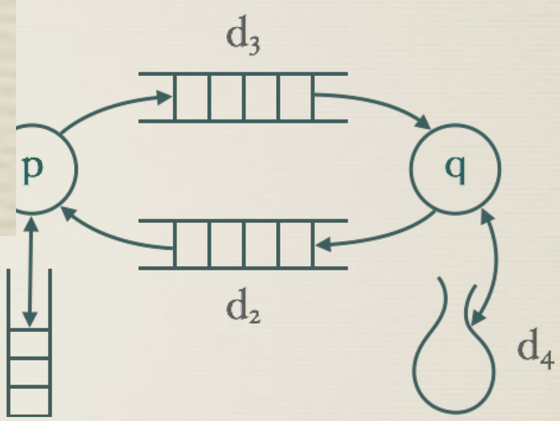


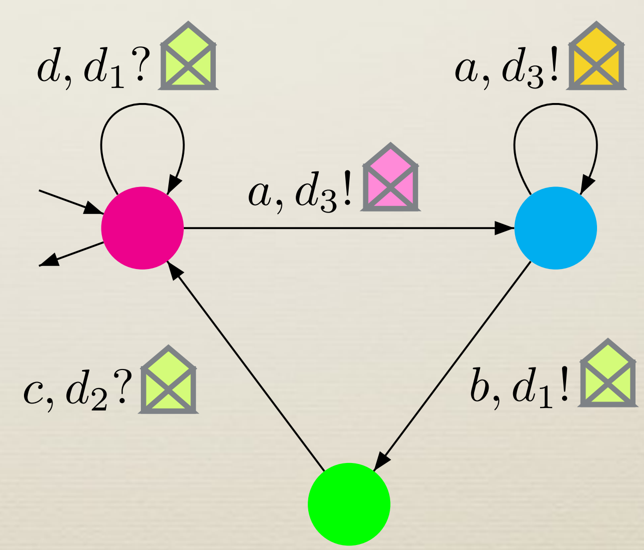
reachability/emptiness

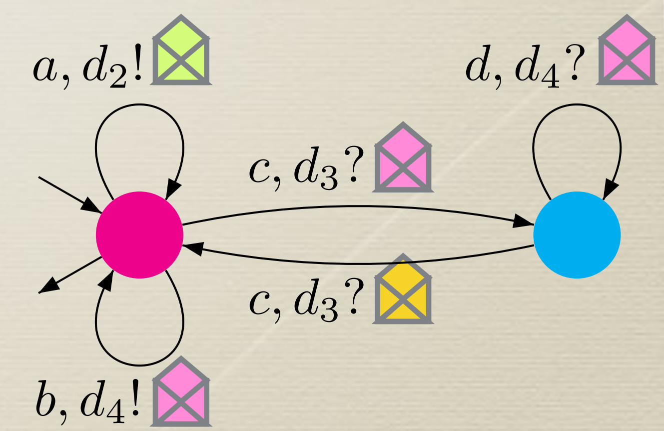




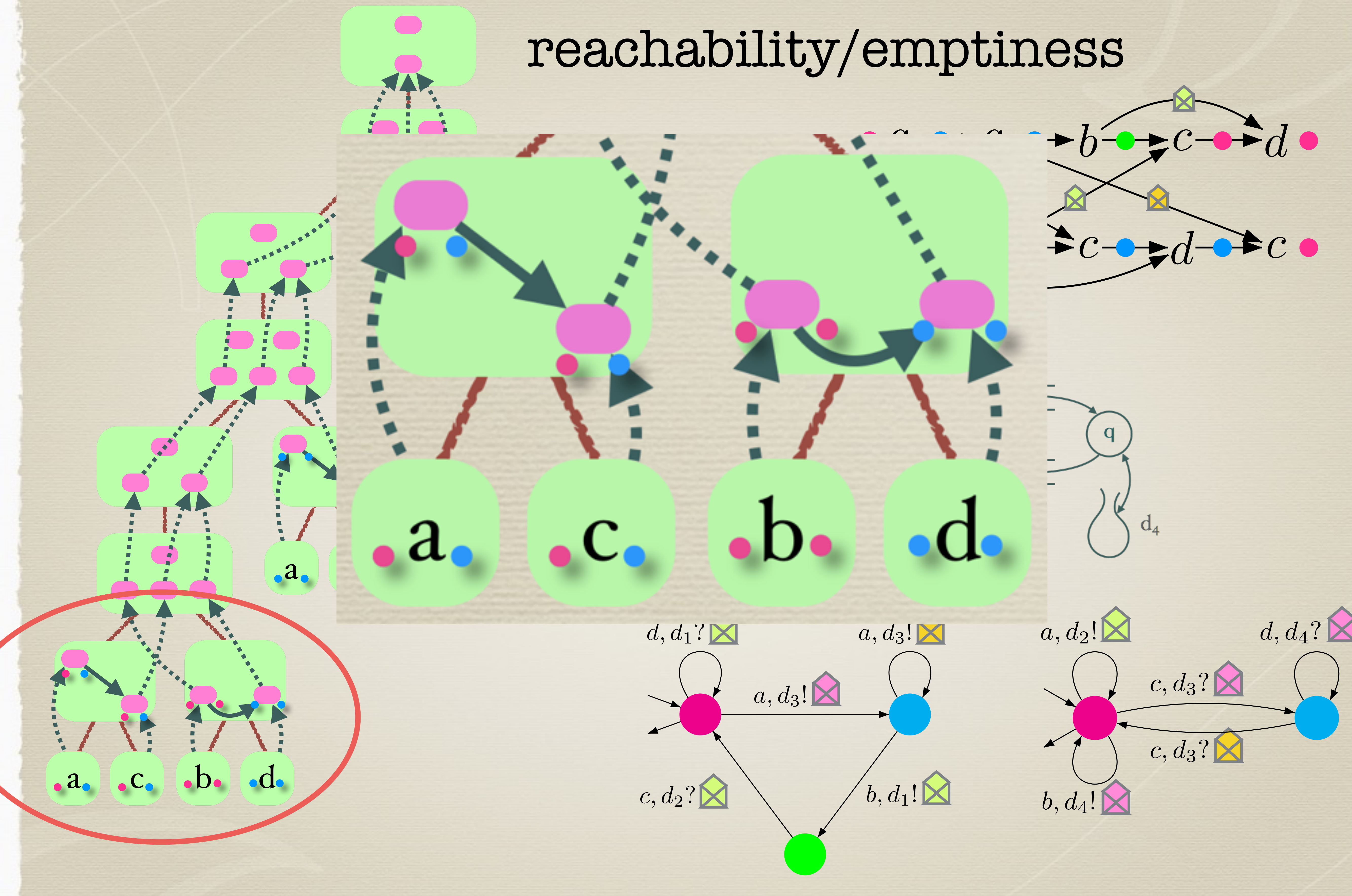




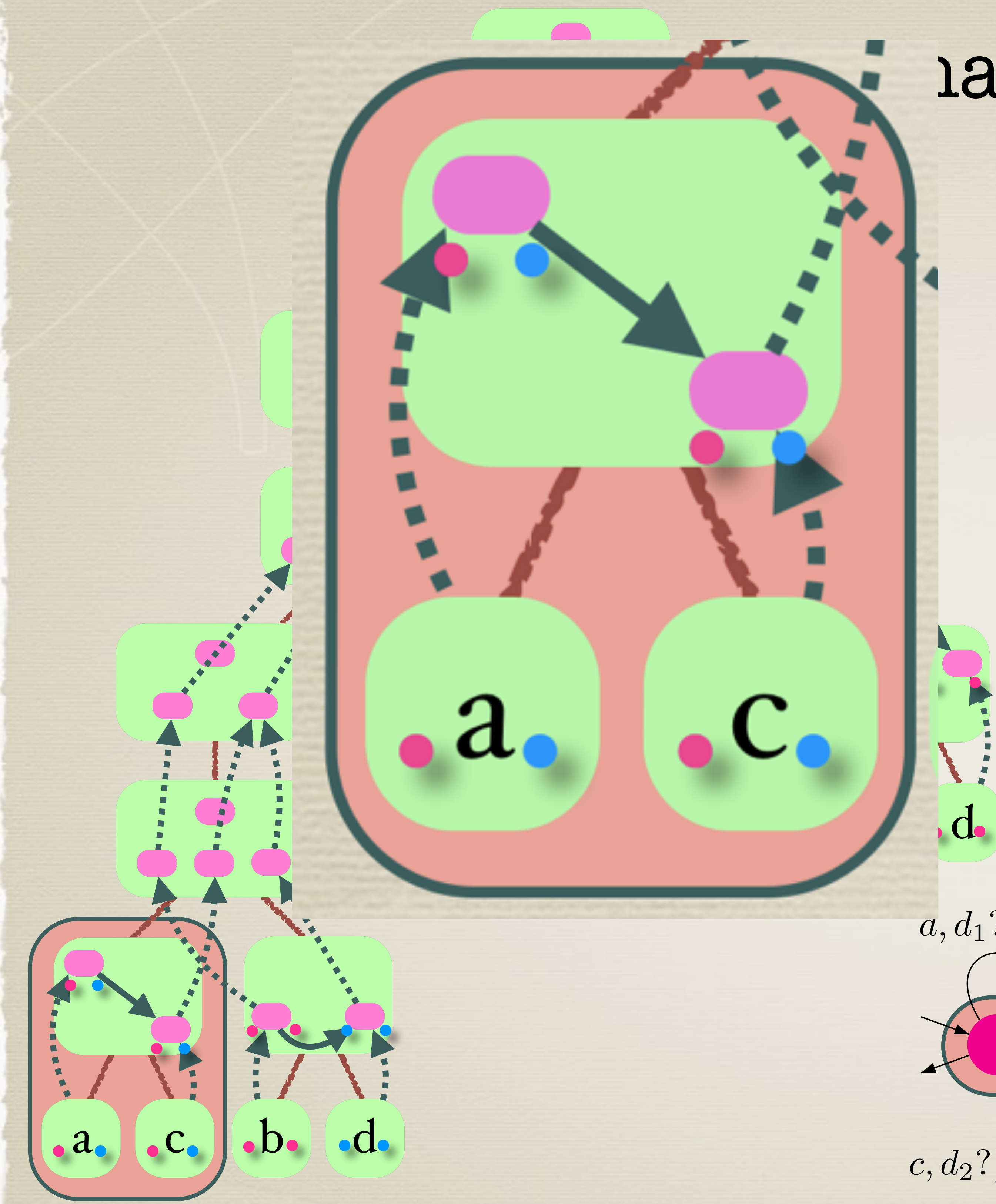


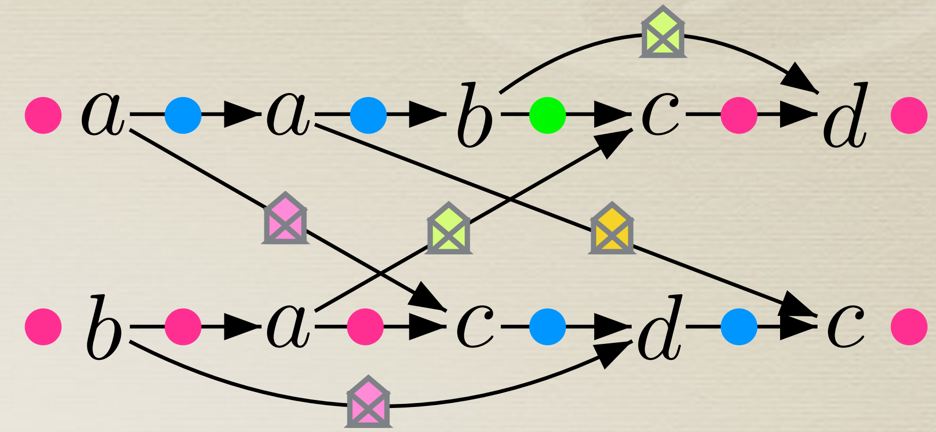


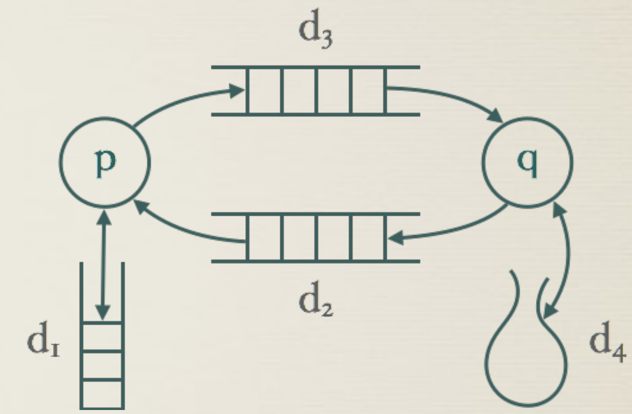
reachability/emptiness

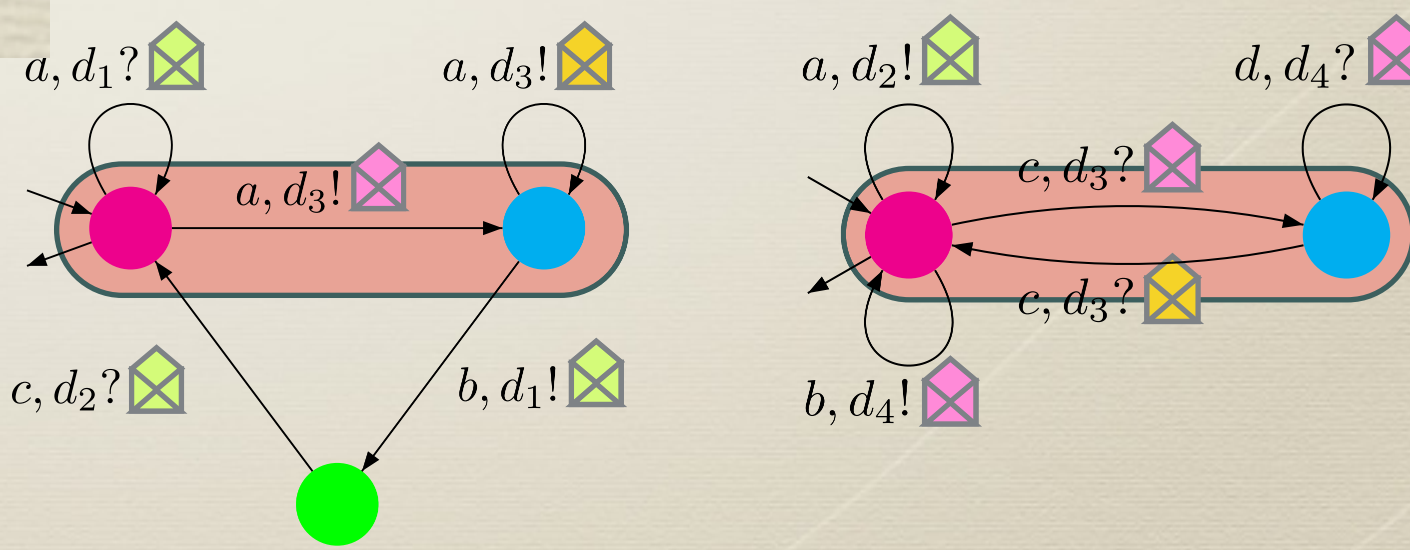


ability/emptiness

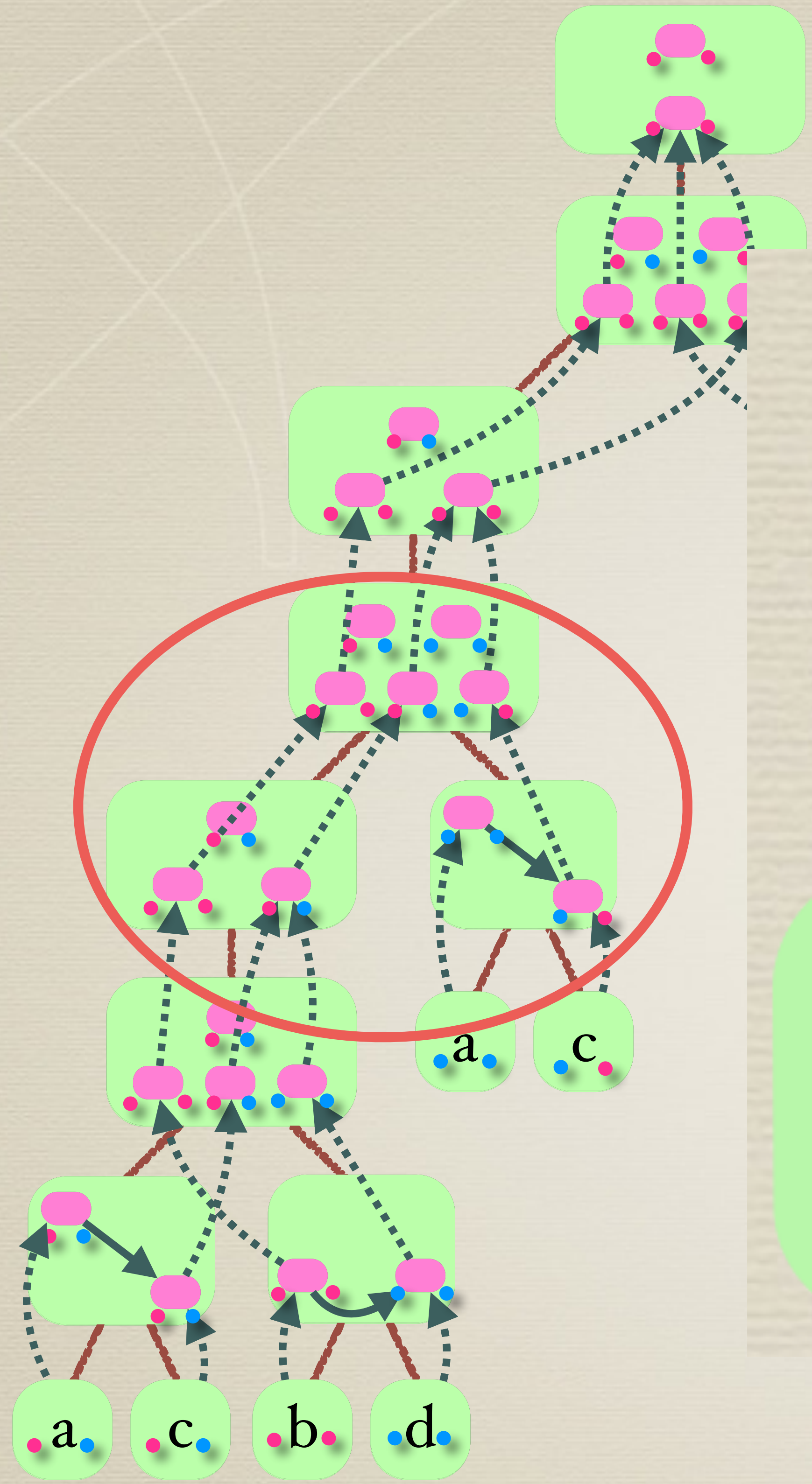


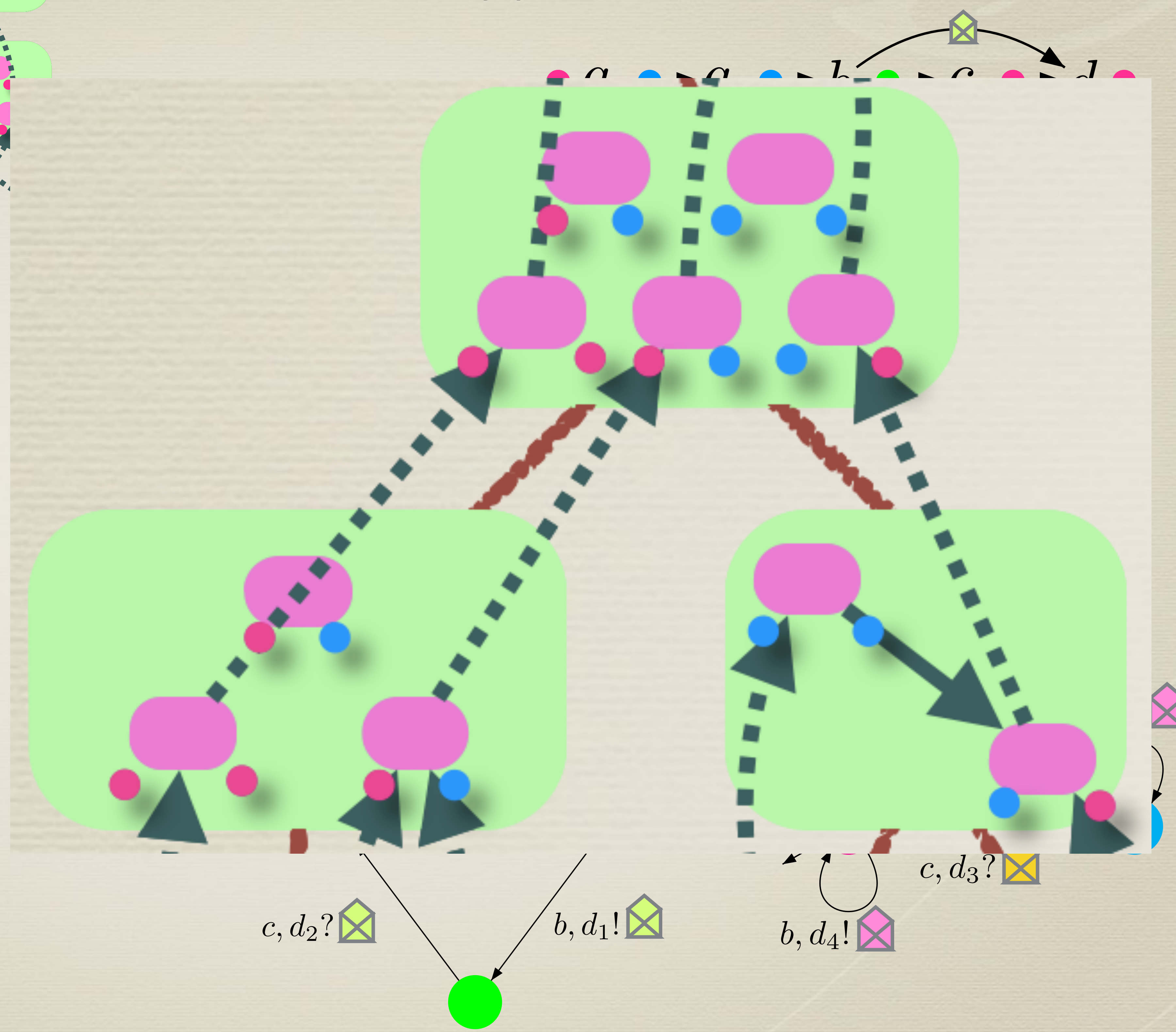




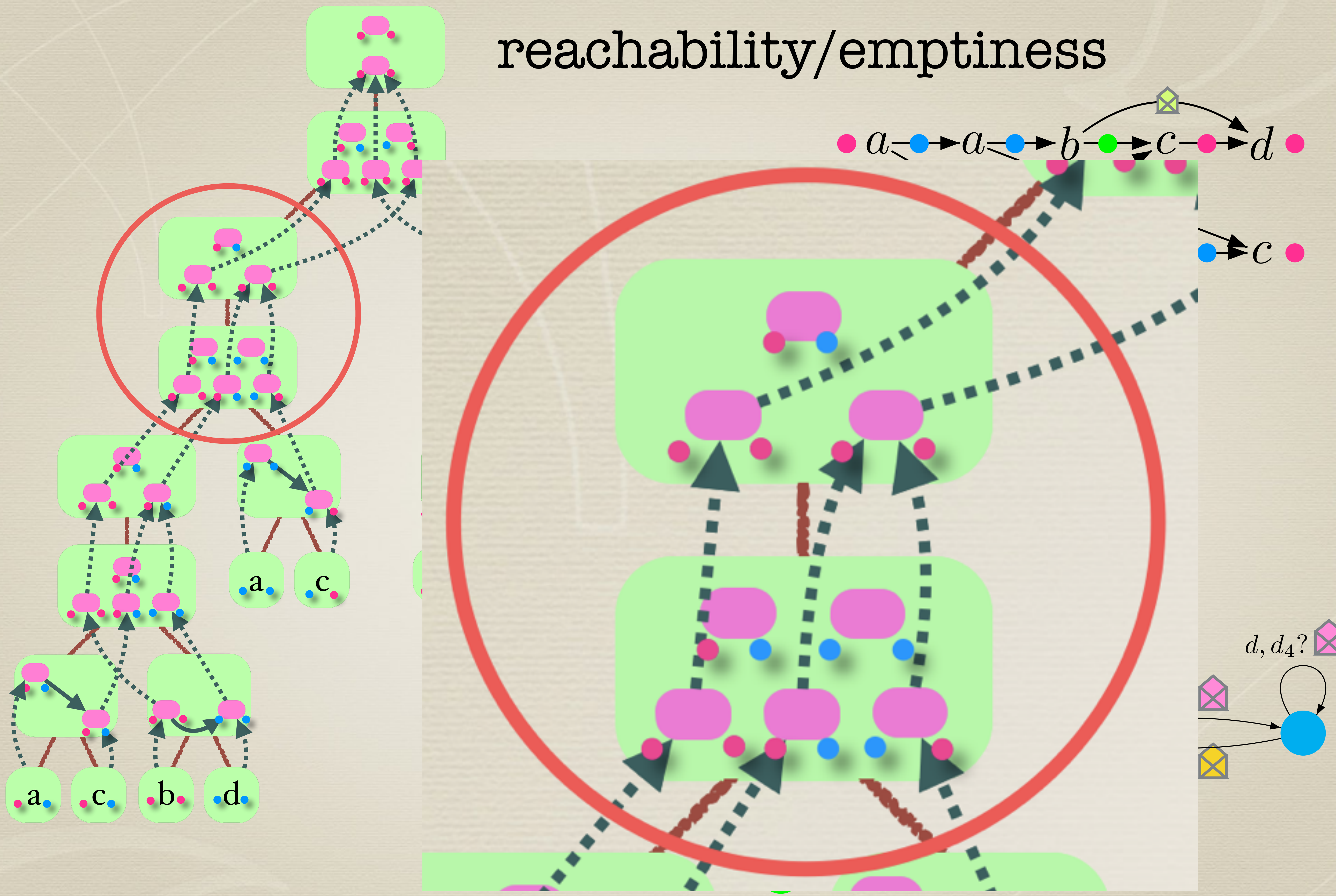


reachability/emptiness

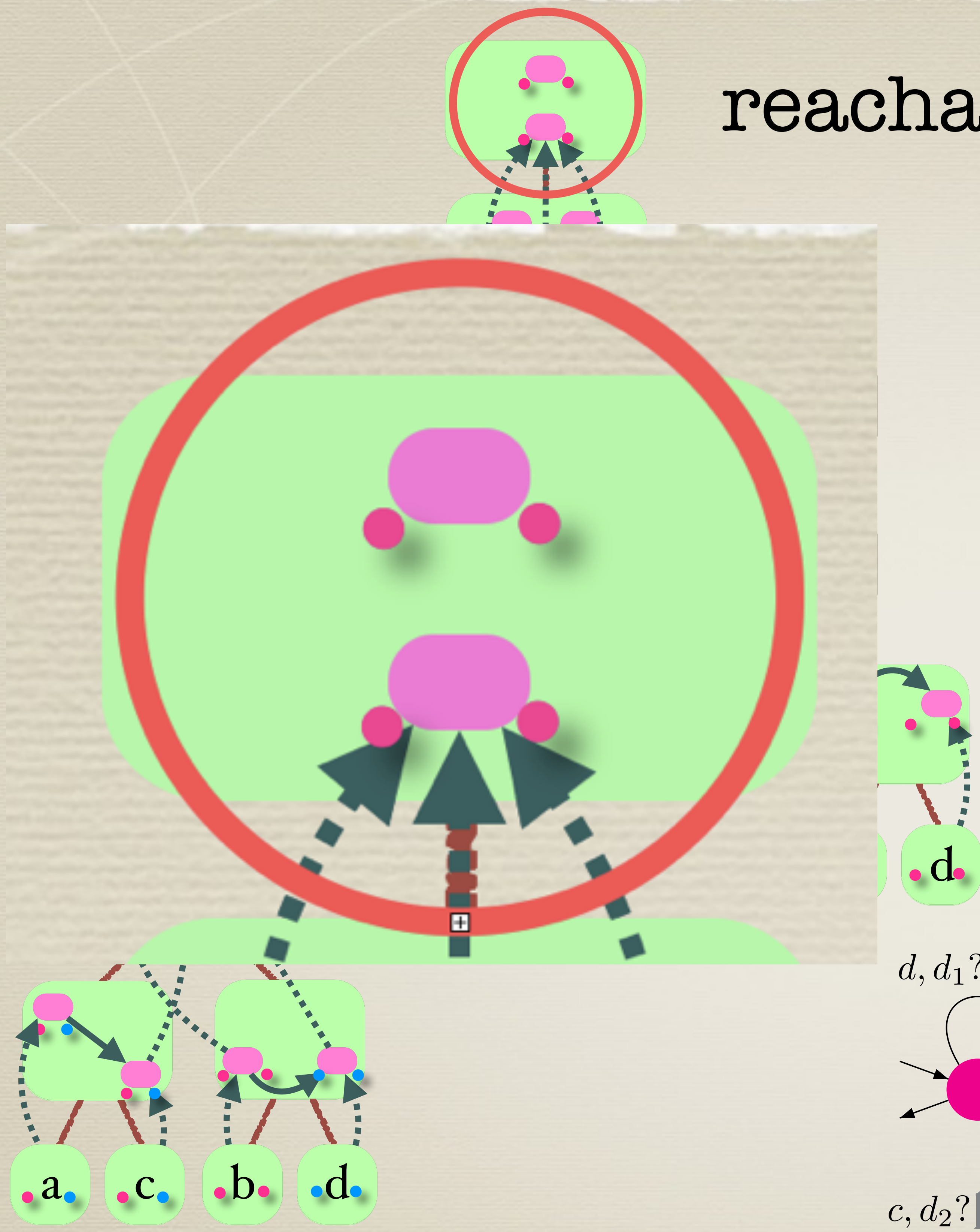


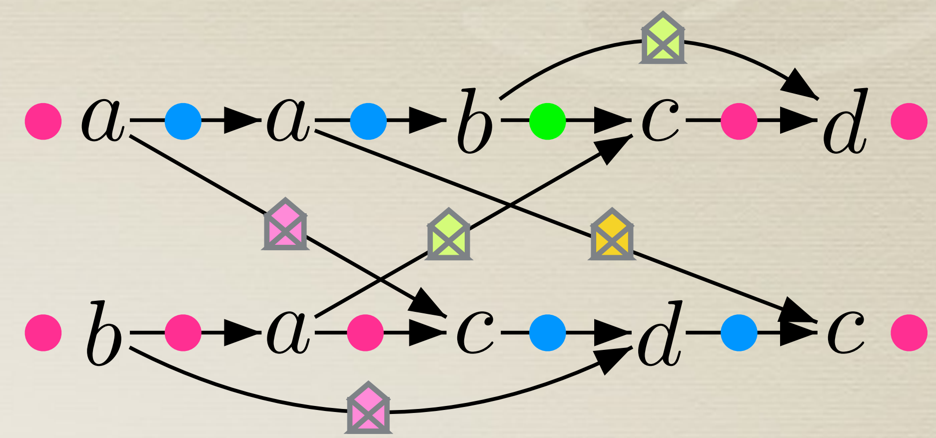


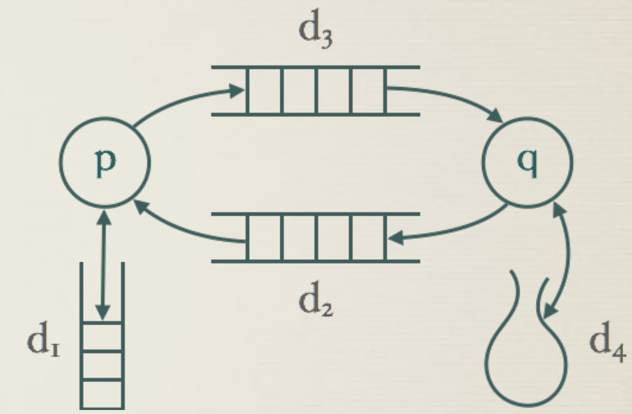
reachability/emptiness

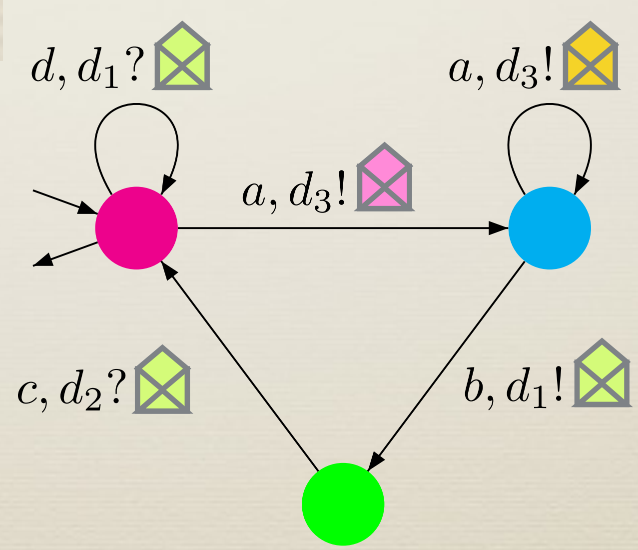


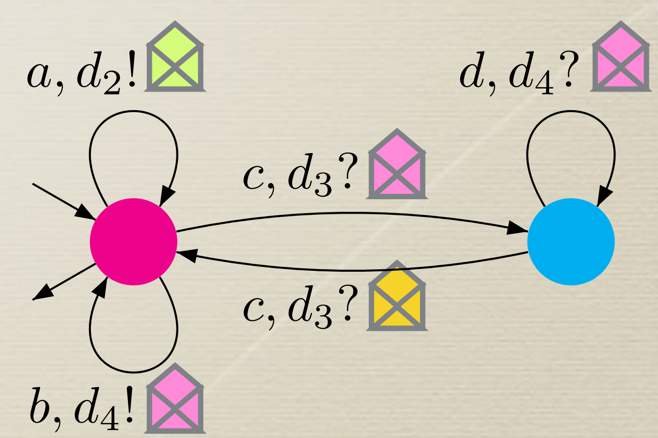
reachability/emptiness











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	

C. Aiswarya, P.G, K. Narayan Kumar

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

Outline

- ☑ Concurrent Processes with Data Structures
- ☑ Behaviors as Graphs
- ☑ Specifications
- ☑ Verification with Graphs and under-approximations
- ☑ Split-width
- Conclusion

WYSIWYG

Understanding Behaviors

Linear Traces	Graphs (CBMs)
• Interleaved sequence of events. Interactions are obfuscated and very difficult to recover. • Successor relation not meaningful • Combinatorial explosion single distributed behavior results in a huge number of linear traces	• Visual description of behavior • Interactions are visible • no combinatorial explosion

WYSIWYG

Expressiveness of Specifications

Linear Traces	Graphs (CBMs)
• Too weak for many natural specifications • Difficult to write/understand • Requires syntactical or semantical restrictions to be meaningful	• Powerful specifications • Interactions are built-in • Meaningful • Easy to write/understand

WYSIWYG

Efficiency of Algorithms

Linear Traces	Graphs (CBMs)
• Undecidable in general • Decidable under restrictions • Reductions to word automata • Good space complexity • Many tools available	• Undecidable in general • Decidable under more lenient restrictions • Reductions to tree automata via tree-interpretations • Good time complexity • Tools to be developed

Conclusion

* Use graphs to reason about behaviors of systems
  distributed or sequential
* Exploit graph theory
  Logics, decompositions, tree interpretations
* Split-width: convenient decomposition technique
  as powerful as tree-width or clique-width for CBMs
  yields optimal algorithms

Perspectives

* Extensions
  * Parameterized systems (size, topology)
    with Marie Fortin, FOSSACS'16
  * Timed systems
    with S. Akshay and S. Krishna, submitted
  * Higher-order PDA
    with C. Aiswarya and P. Saivasan
  * Dynamic creation of processes
  * Read from many
  * Infinite behaviors
  * ...
* Tools

Perspectives

* Extensions
  * Parameterized systems (size, topology)
    with Marie Fortin, FOSSACS'16
  * Timed systems
    with S. Akshay and S. Krishna, submitted
  * Higher-order PDA
    with C. Aiswarya and P. Saivasan
  * Dynamic creation of processes
  * Read from many
  * Infinite behaviors
  * ...
* Tools

THANK YOU