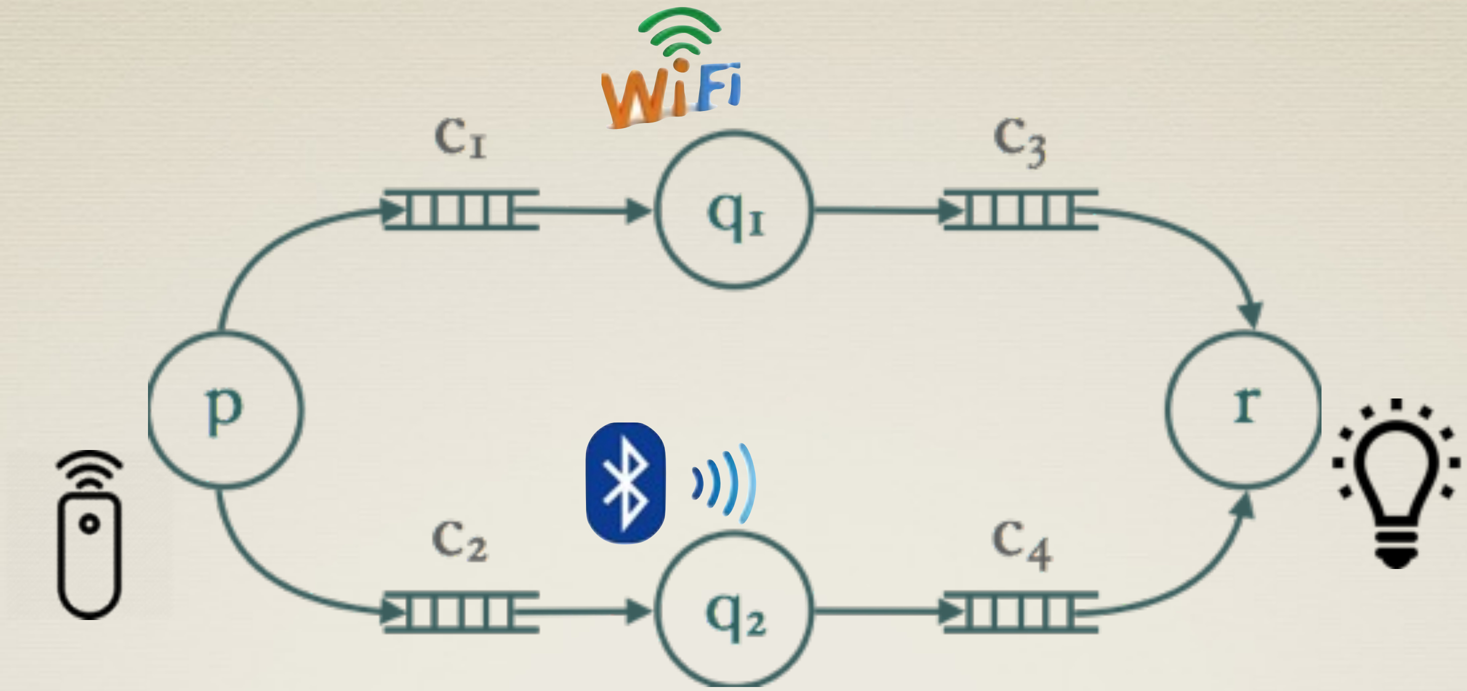
Reasoning about Distributed Systems: WYSIWYG

Paul Gastin
LSV, ENS Cachan, France
C. Aiswarya
Uppsala University, Sweden

Introduction



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

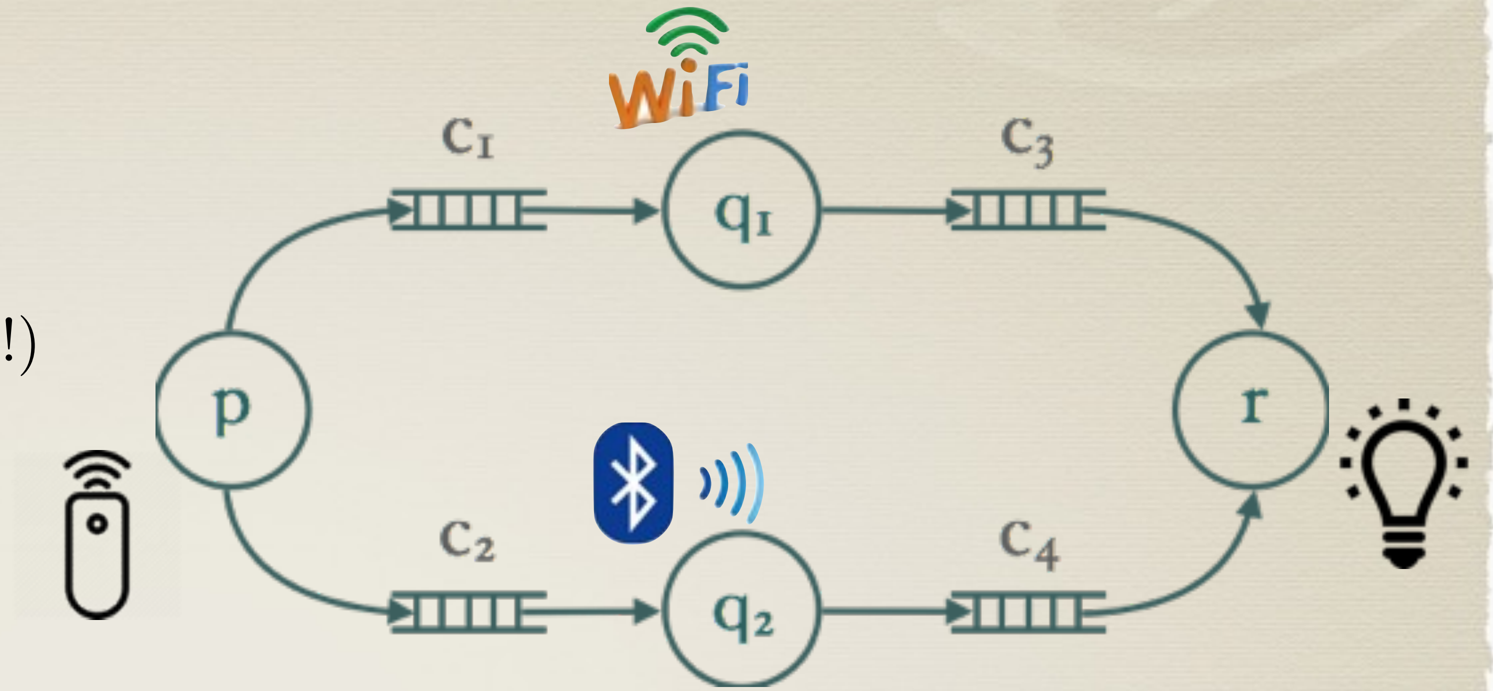
$$(q_2, c_4!)(p, \mathsf{on})(p, c_2!)(p, \mathsf{off})(r, c_4?)(r, \mathsf{on})$$

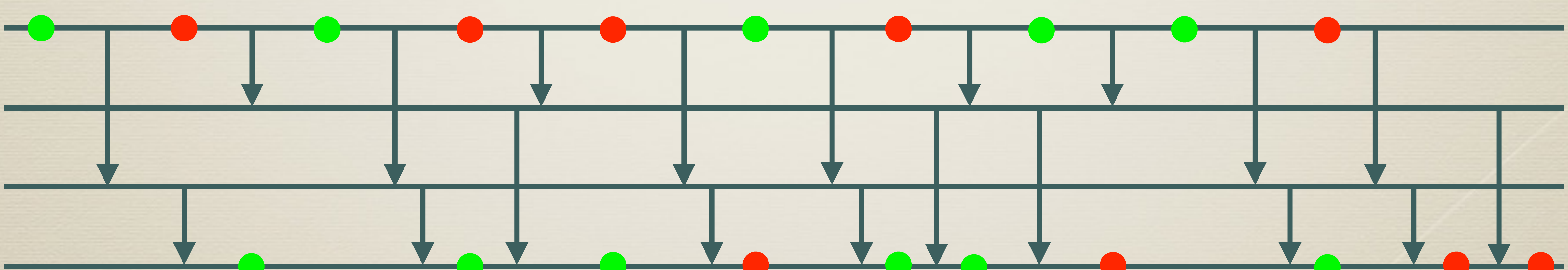
$$(q_1, c_3!)(p, c_1!)(q_1, c_1?)(q_1, c_3!)(q_2, c_2?)(q_2, c_4!)$$

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

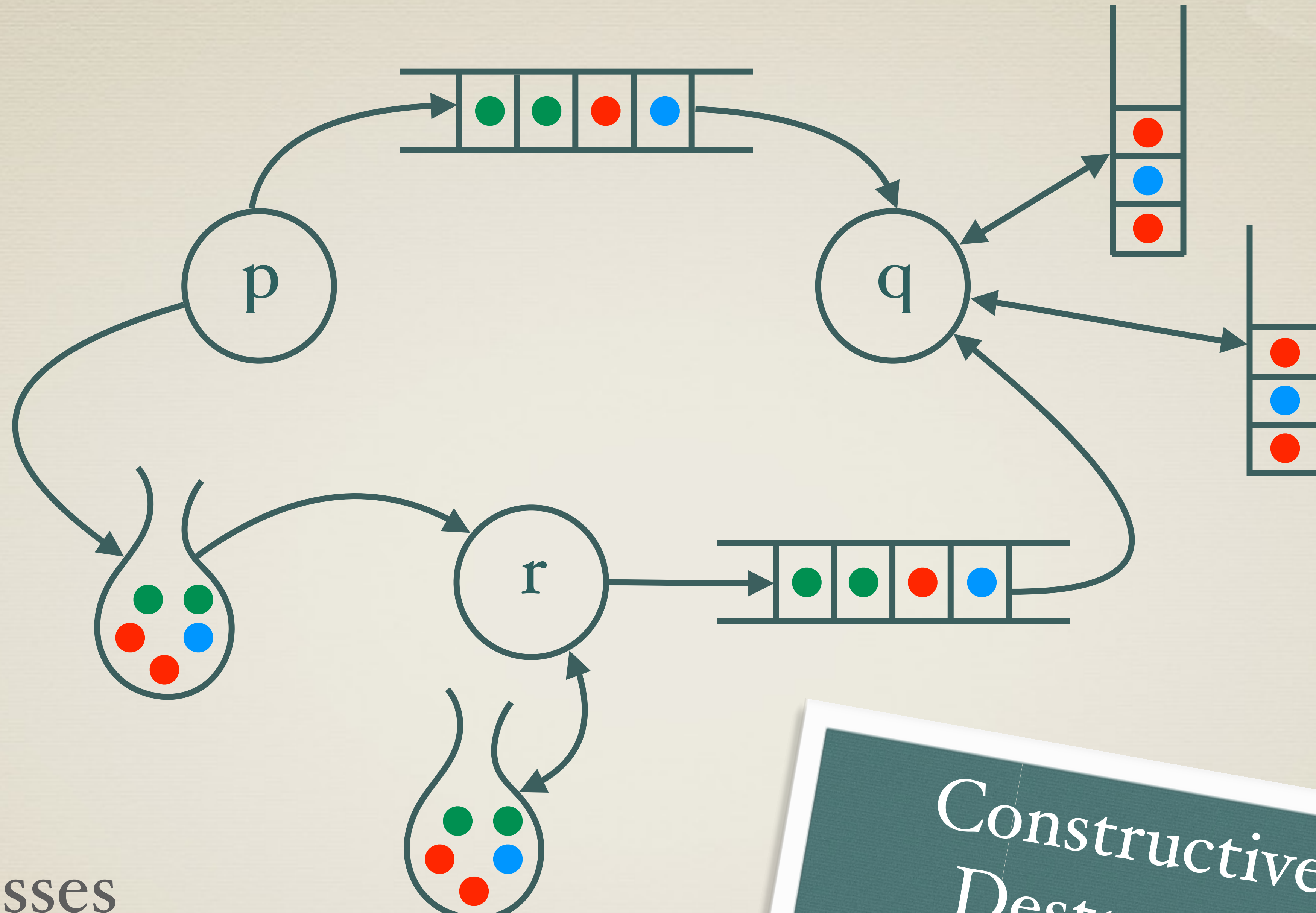
Introduction

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





System: Concurrent Processes with Data-Structures

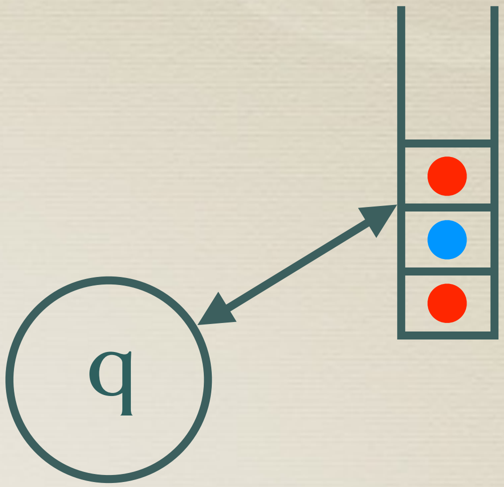


Constructive writes
Destructive reads

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

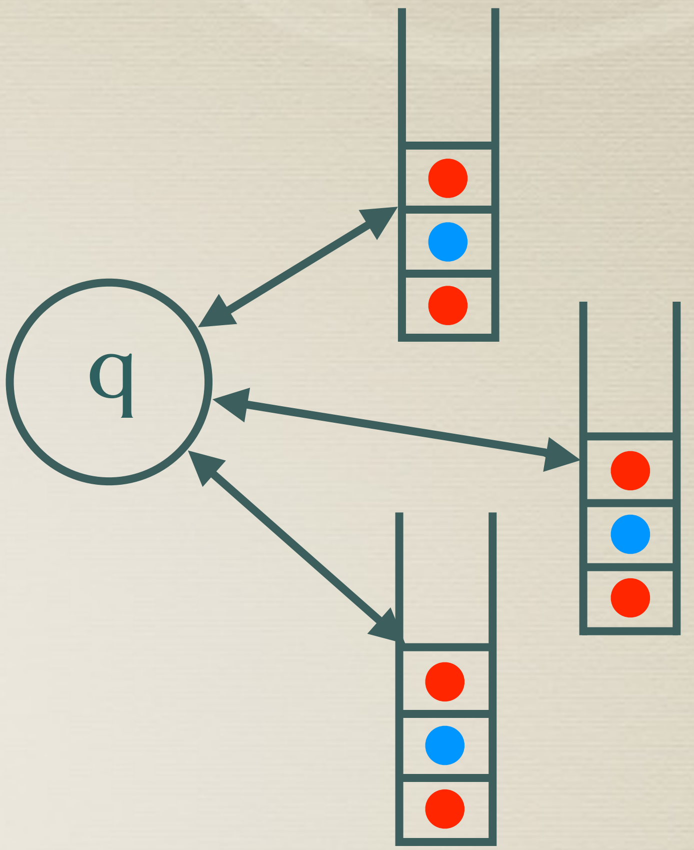
Architectures: Special cases

- PDA: Pushdown automata
  Recursive programs



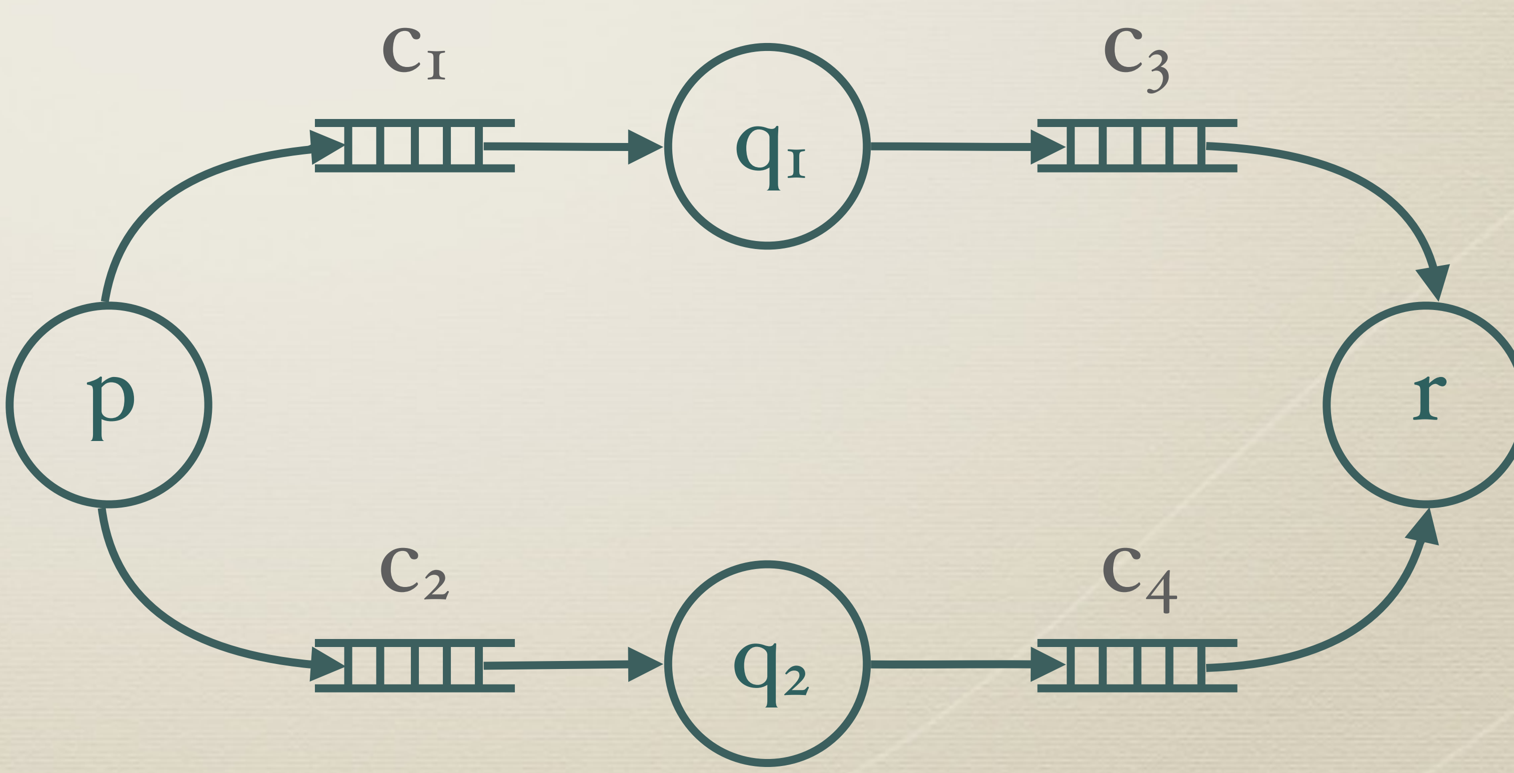
Architectures: Special cases

- PDA: Pushdown automata
  Recursive programs
- MPDA: Multi-pushdown automata
  Multi-threaded recursive programs



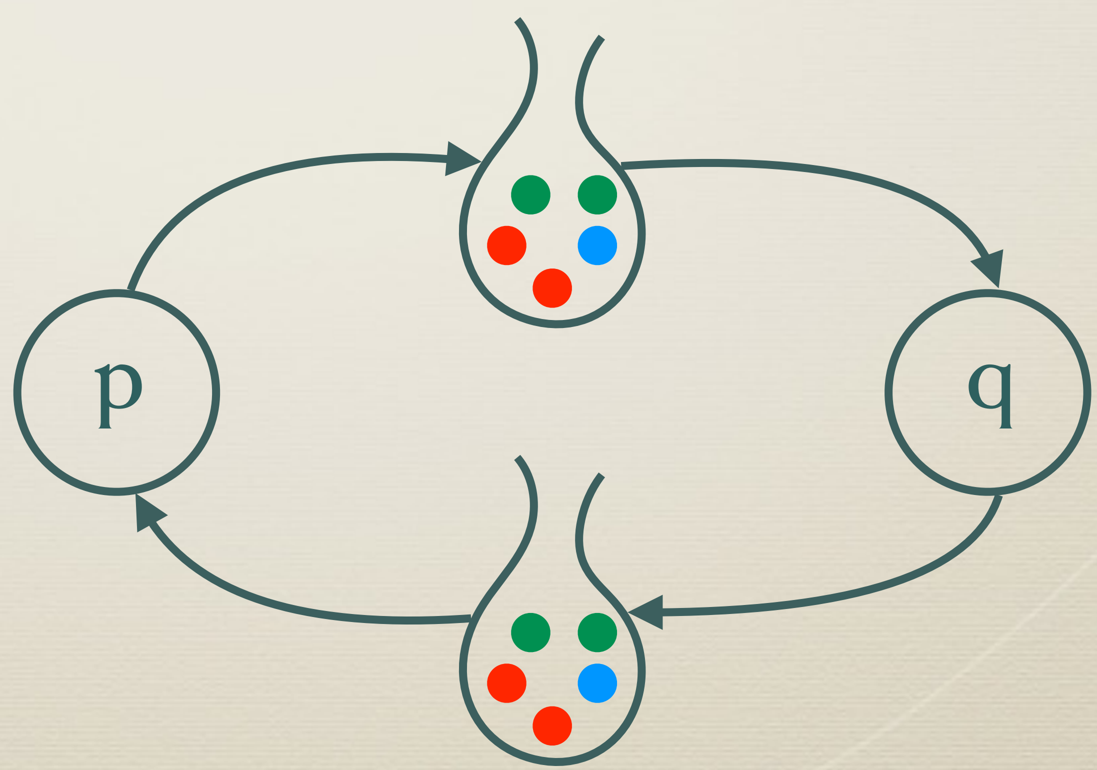
Architectures: Special cases

- PDA: Pushdown automata
  Recursive programs
- MPDA: Multi-pushdown automata
  Multi-threaded recursive programs
- MPA: Message passing automata
  Communicating finite state machines

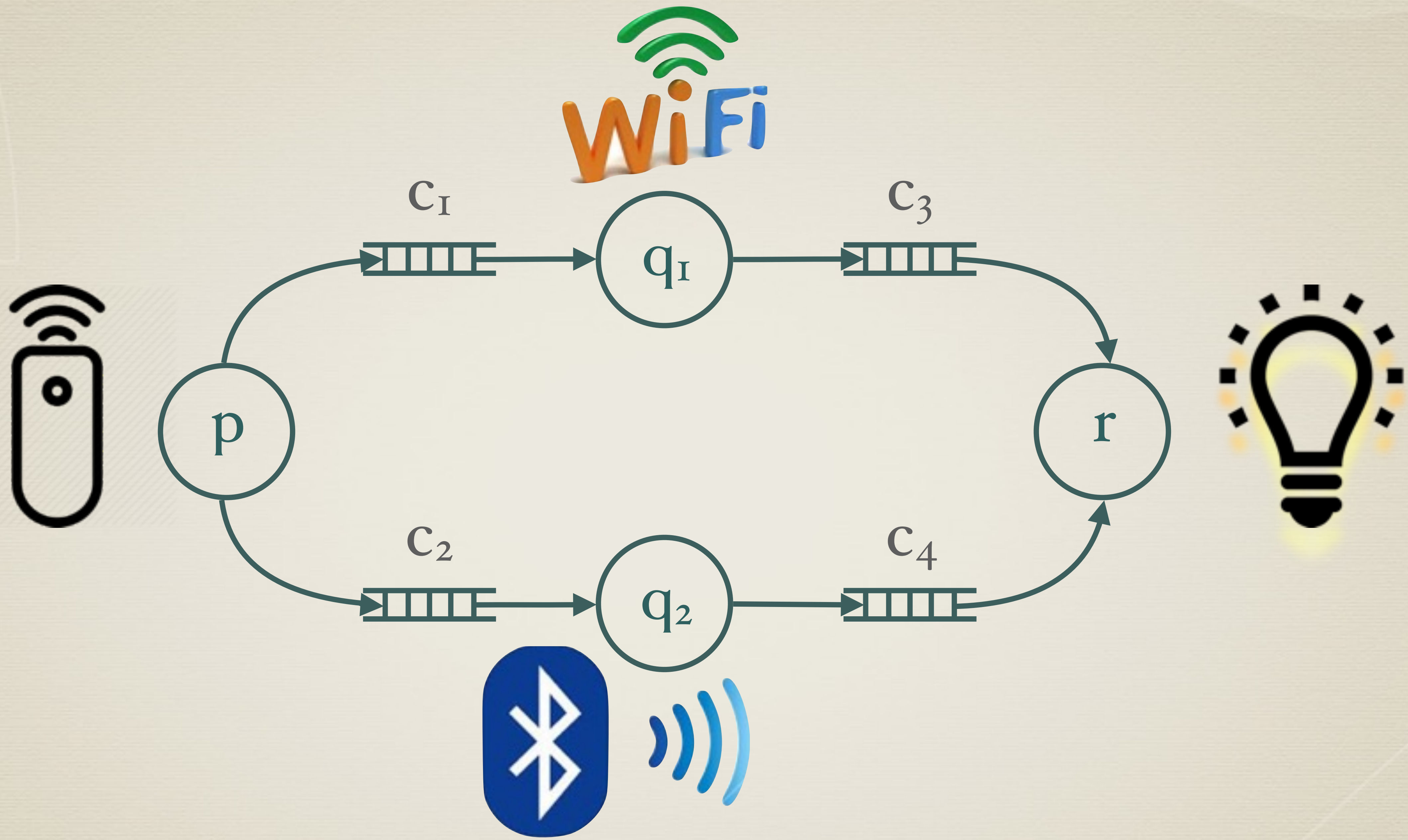


Architectures: Special cases

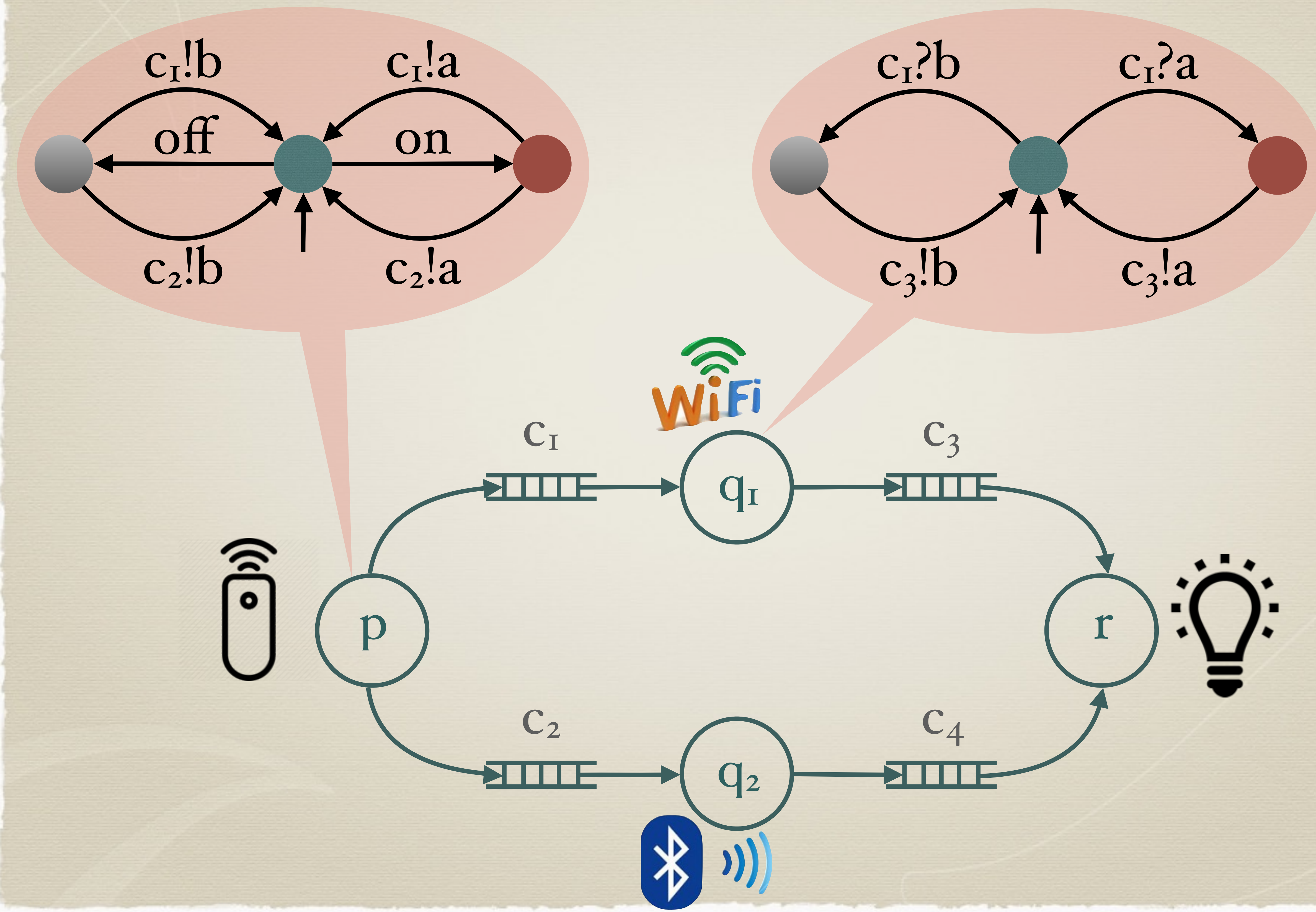
- PDA: Pushdown automata
  Recursive programs
- MPDA: Multi-pushdown automata
  Multi-threaded recursive programs
- MPA: Message passing automata
  Communicating finite state machines
- PN: Petri Nets
  Only bags



Remote on-off via 2 channels

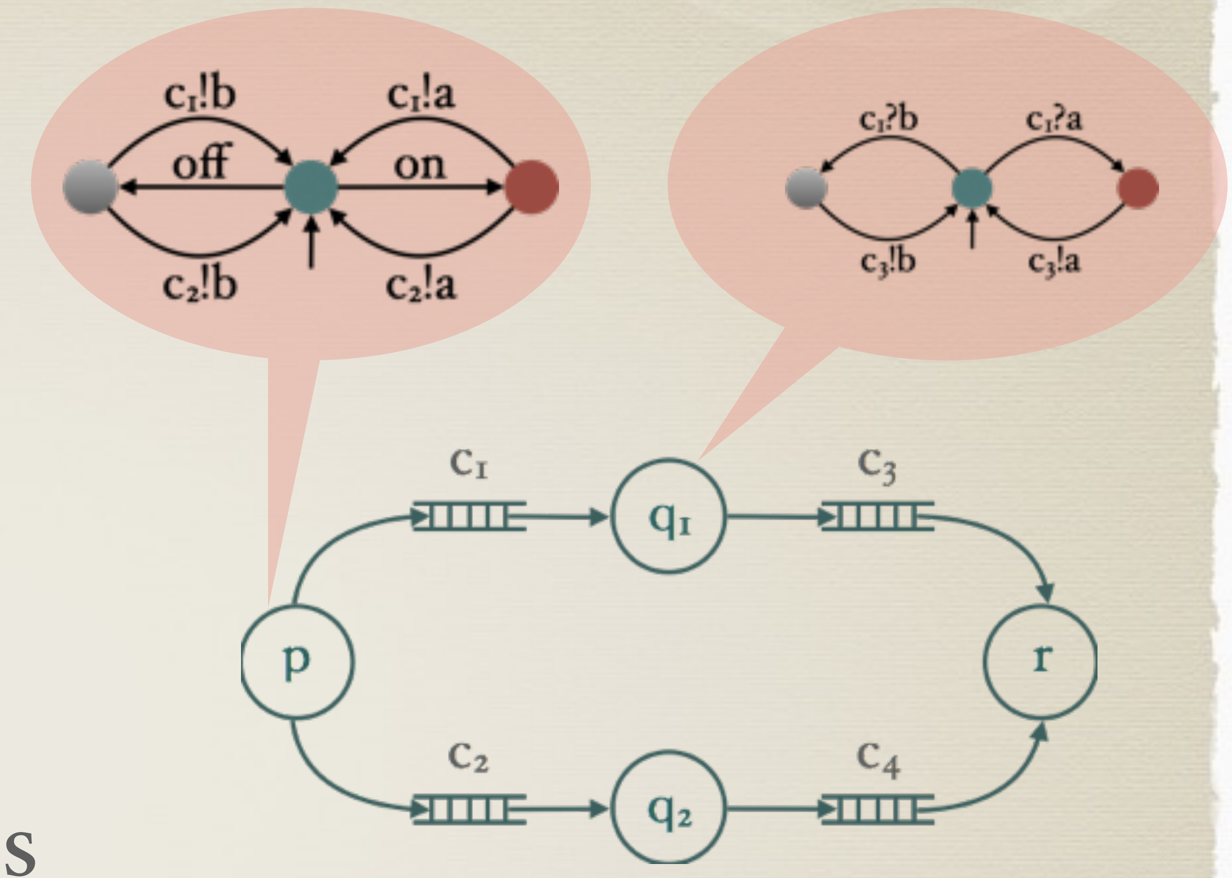


System: Architecture + Boolean Programs



Operational semantics

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



Linear Traces

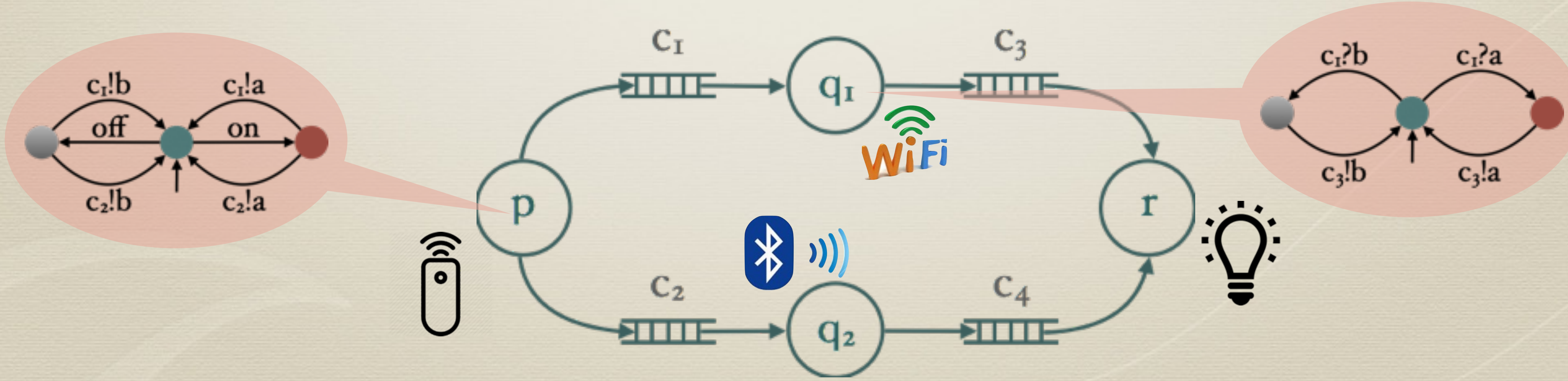
WYSIWYG:
Make visible what is important

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

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

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

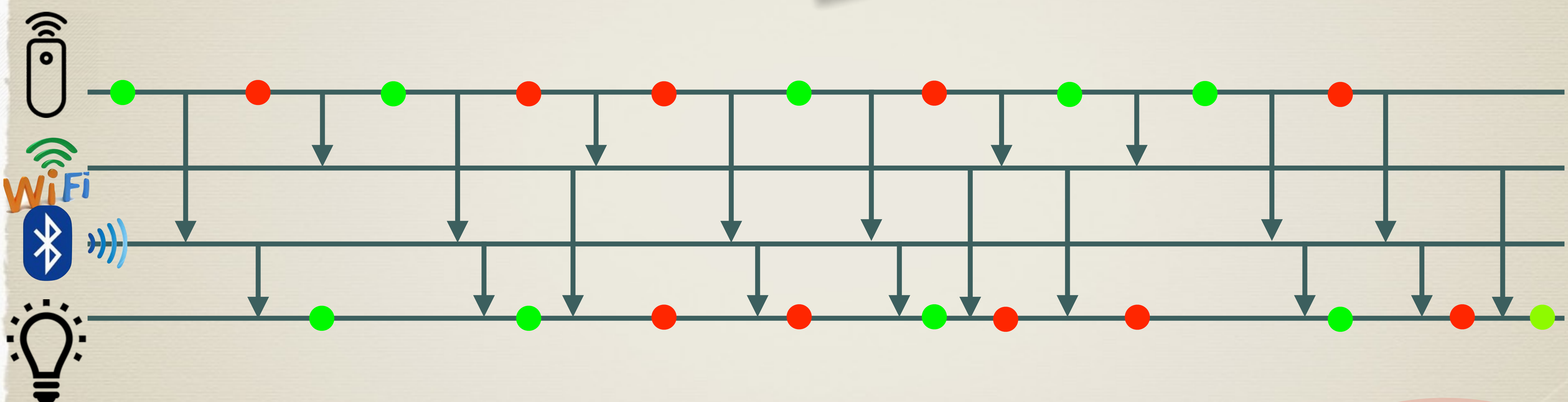
$(r, c_4?)(r, \mathsf{on})(r, c_3?)(r, \mathsf{off}) \cdots$

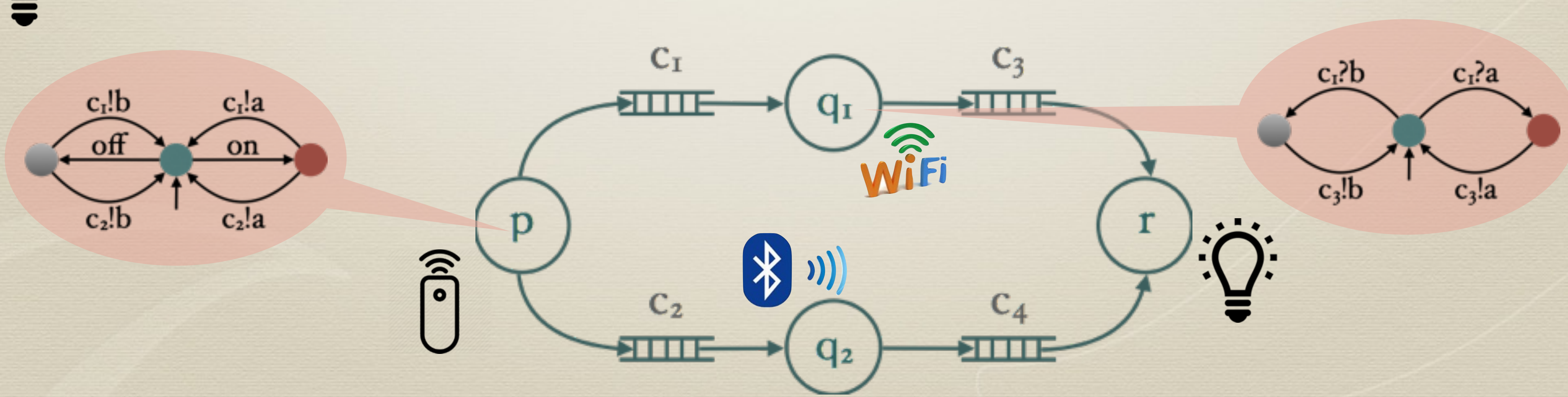


Linear Traces vs. Graphs

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

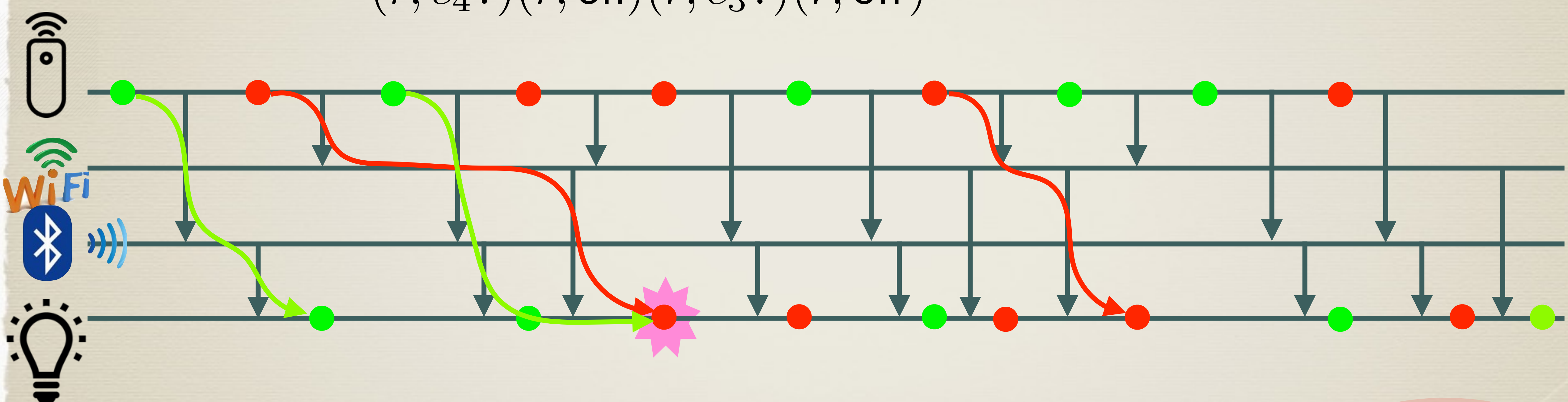
Message Sequence Charts
ITU Standard

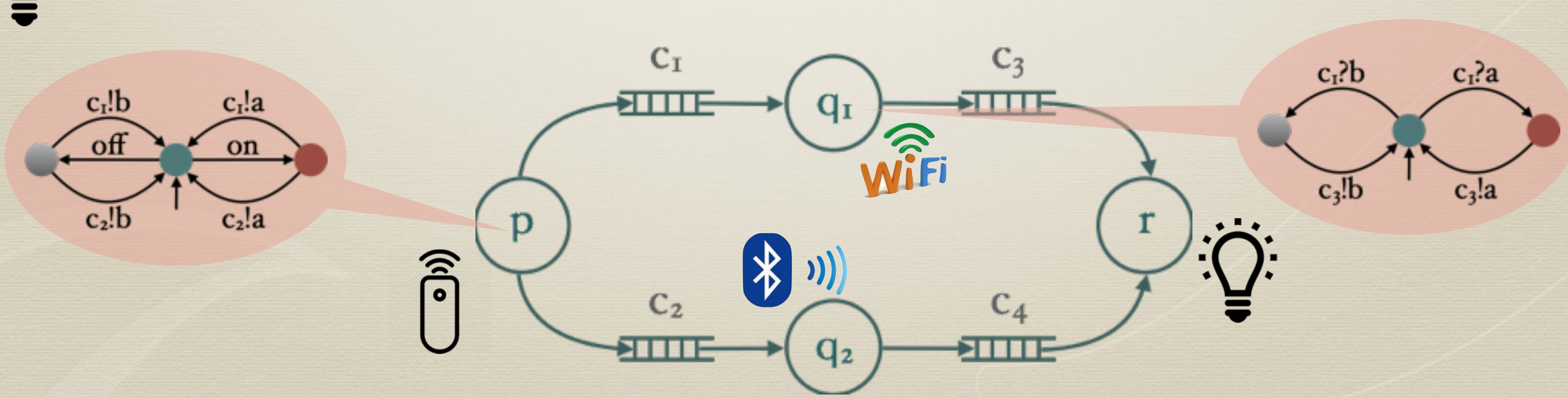




Obey the Latest Order

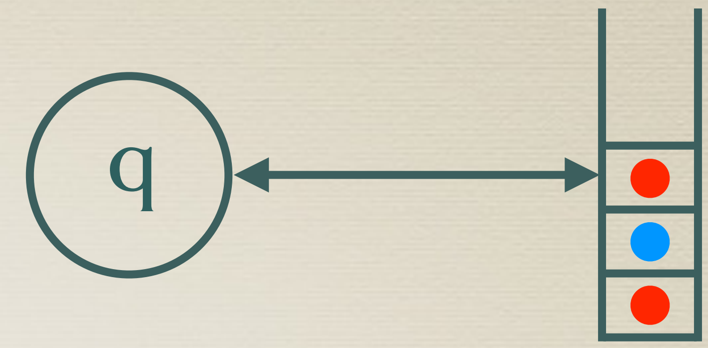
$$(p,\mathsf{on})(p,c_2!)(p,\mathsf{off})(q_2,c_2?)(p,c_1!)(q_1,c_1?)$$
$$(q_2,c_4!)(p,\mathsf{on})(p,c_2!)(p,\mathsf{off})(r,c_4?)(r,\mathsf{on})$$
$$(q_1,c_3!)(p,c_1!)(q_1,c_1?)(q_1,c_3!)(q_2,c_2?)(q_2,c_4!)$$
$$(r,c_4?)(r,\mathsf{on})(r,c_3?)(r,\mathsf{off})\cdots$$





Graphs for Sequential Systems

Answer the correct client
for topmost requests

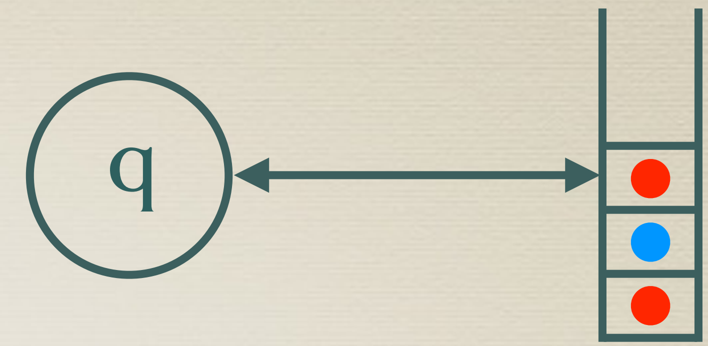


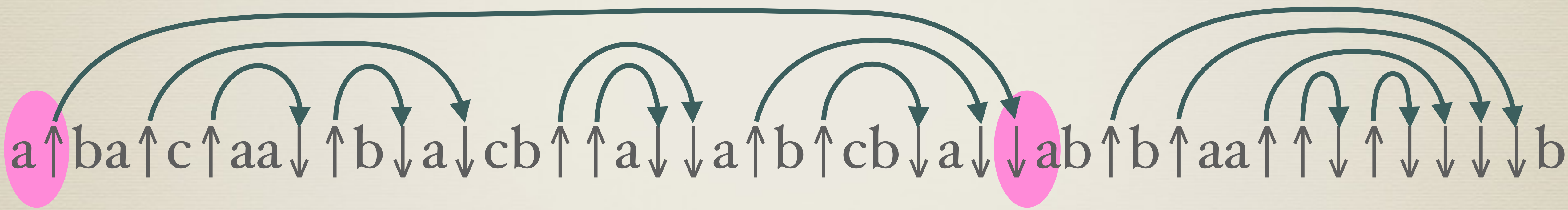
a↑ba↑c↑aa↓↑b↓a↓cb↑↑a↓↓a↑b↑cb↓a↓↓ab↑b↑aa↑↑↓↑↓↓↓↓b

WYSIWYG:
Make visible what is important

Graphs for Sequential Systems

Answer the correct client for topmost requests





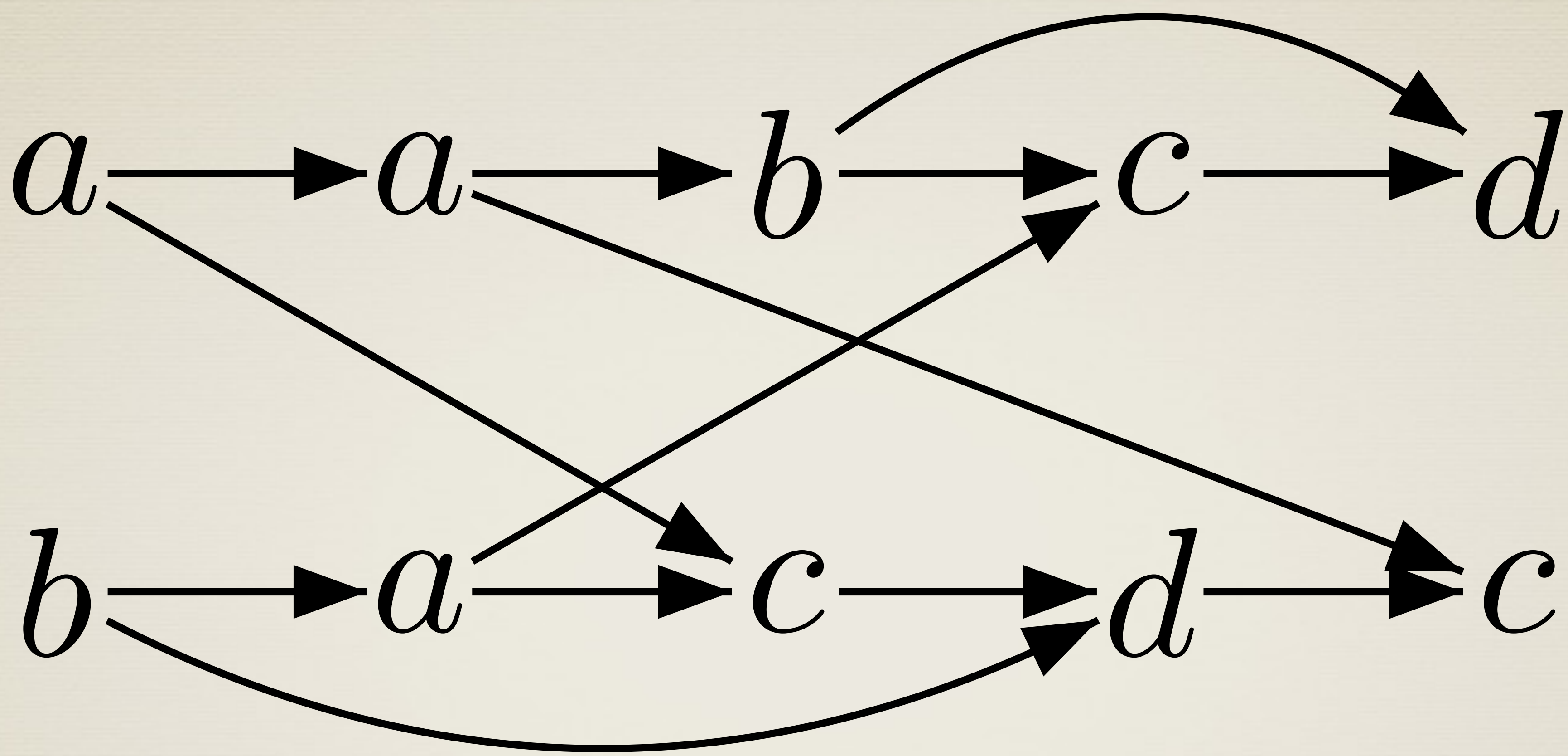
Behaviors should be graphs

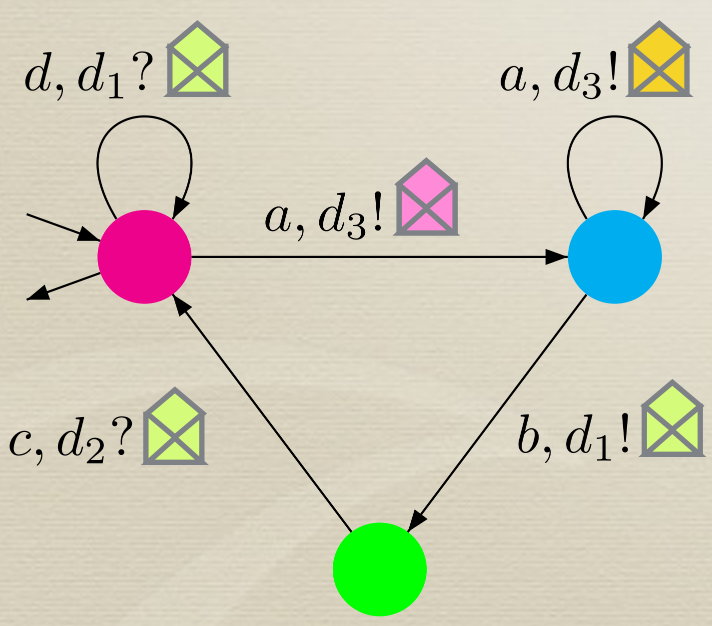
Make visible what is important

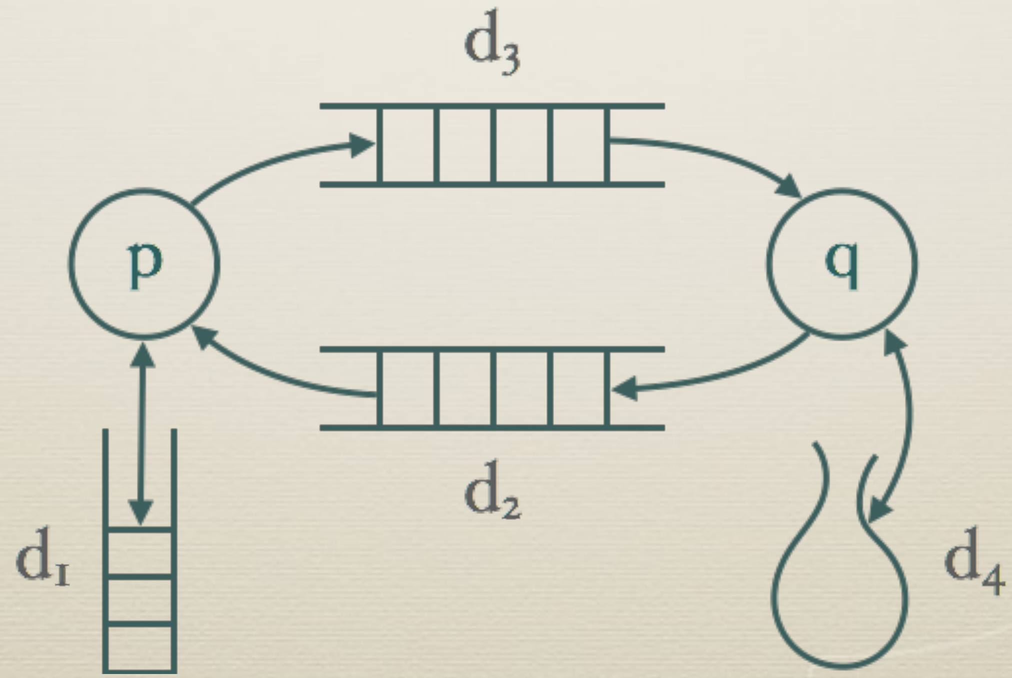
Nested Words

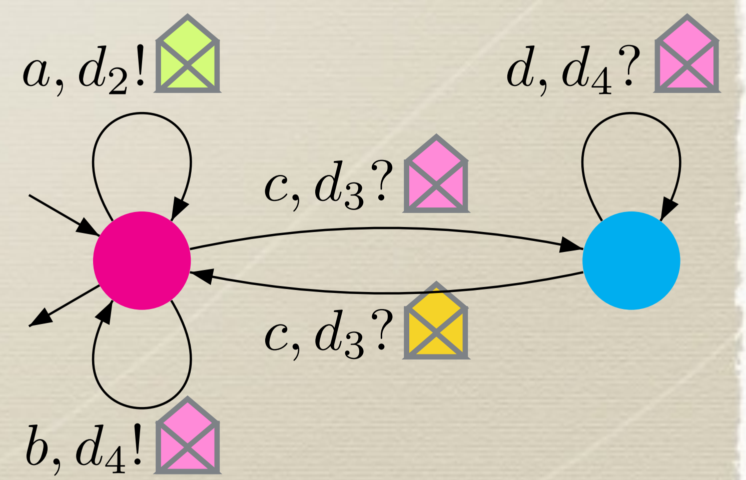
Alur, Madhusudan, 2009

Semantics of CPDS on Graphs

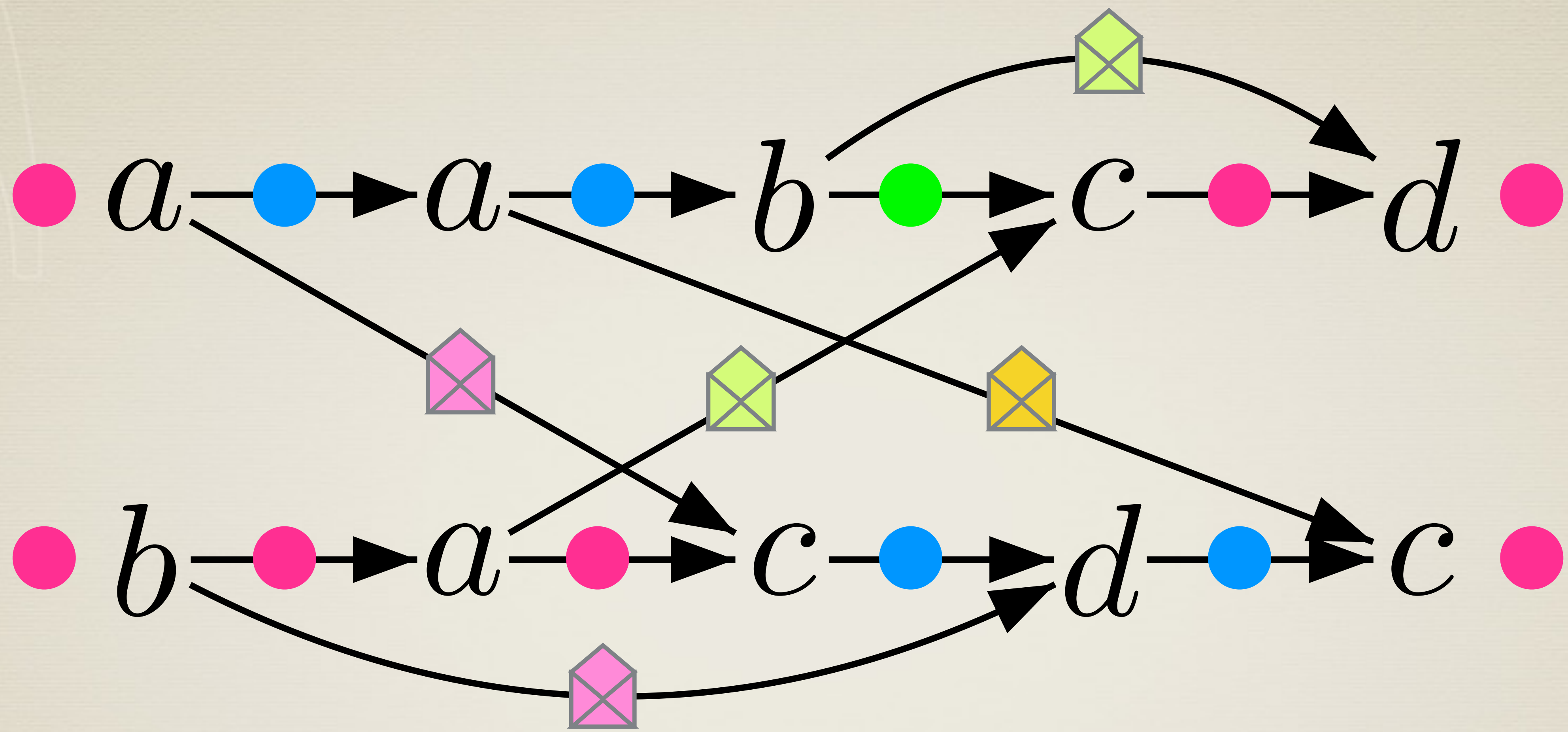


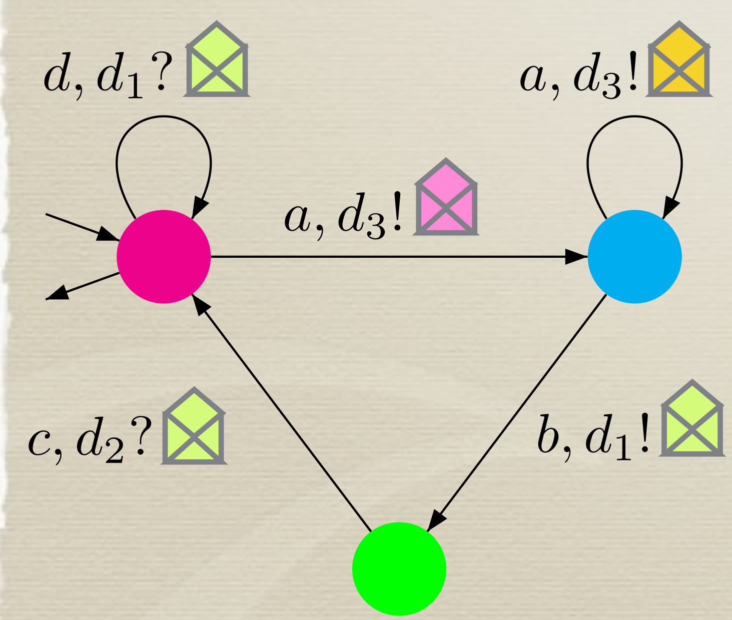


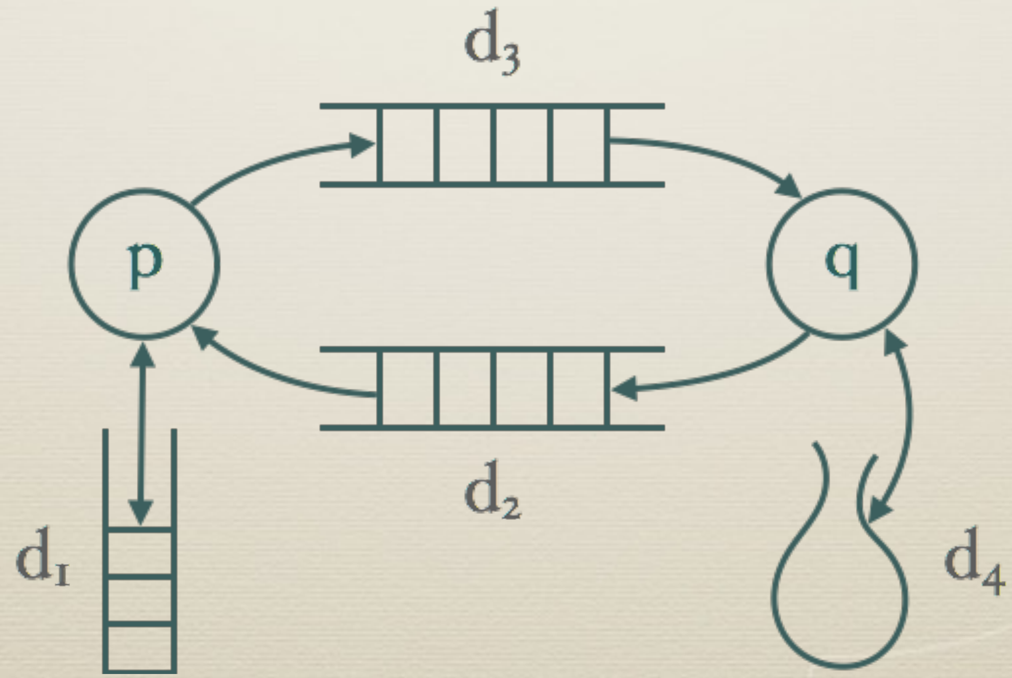


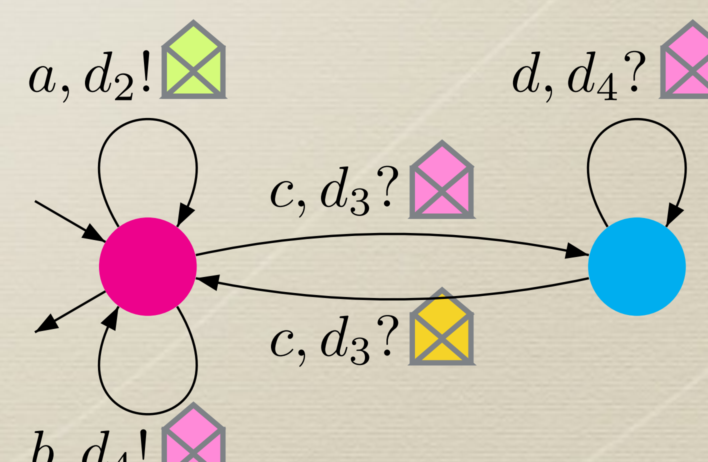


Semantics of CPDS on Graphs

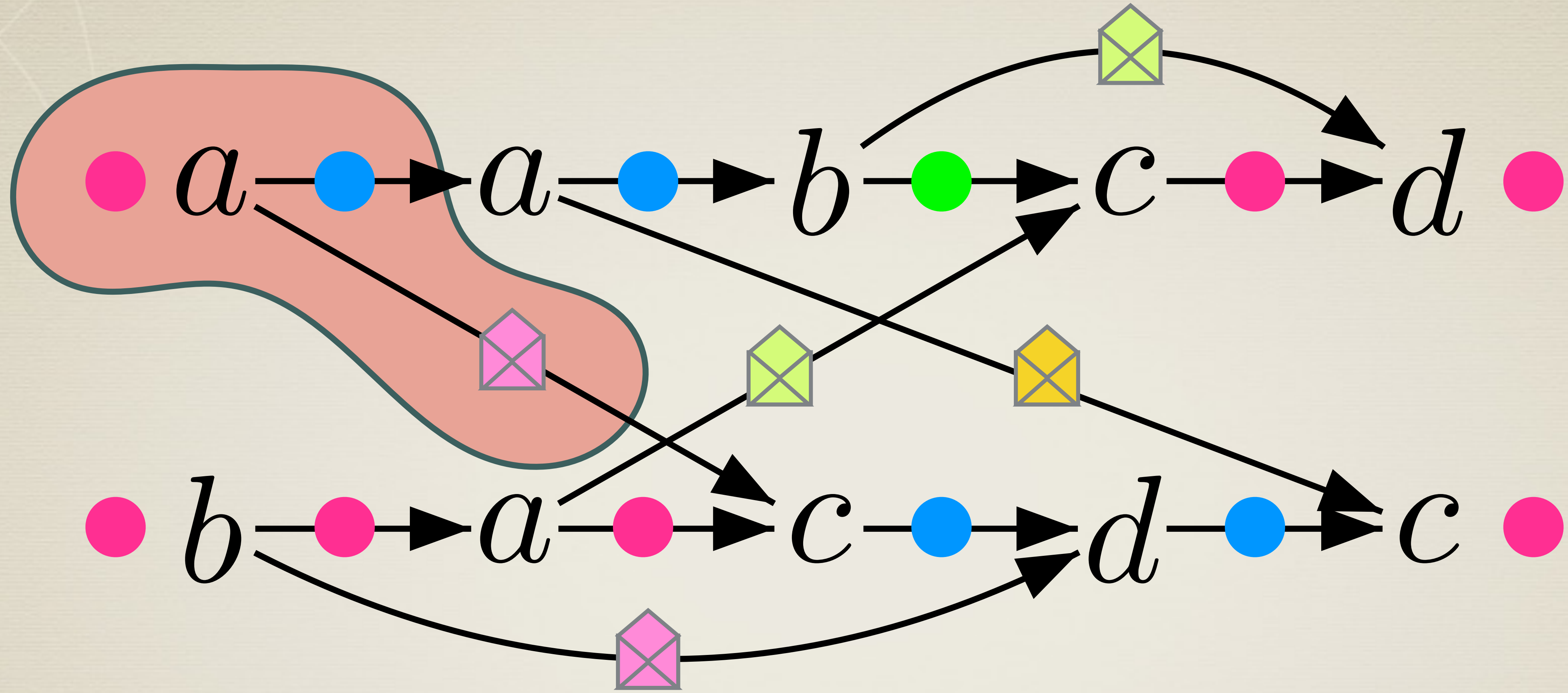


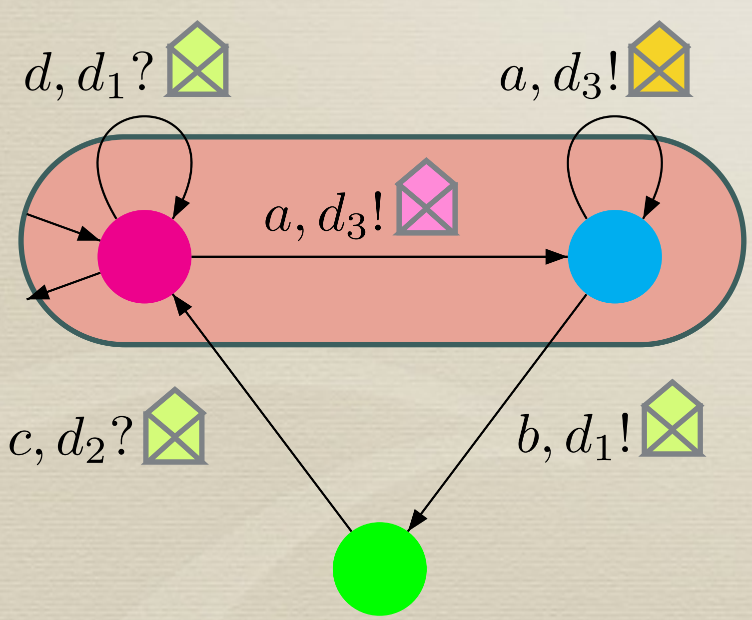


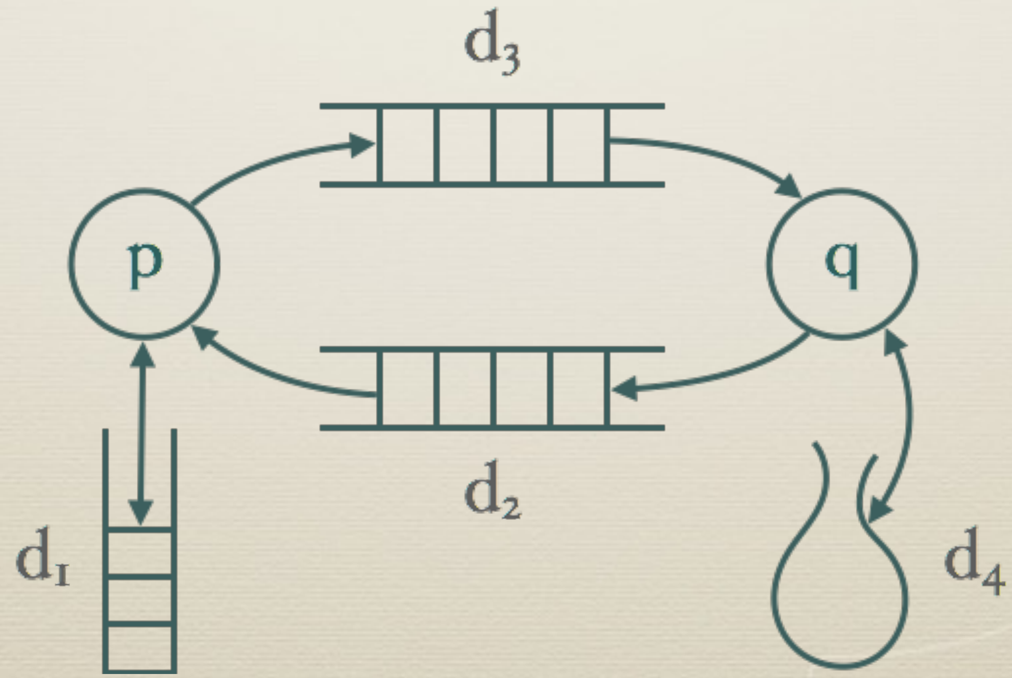


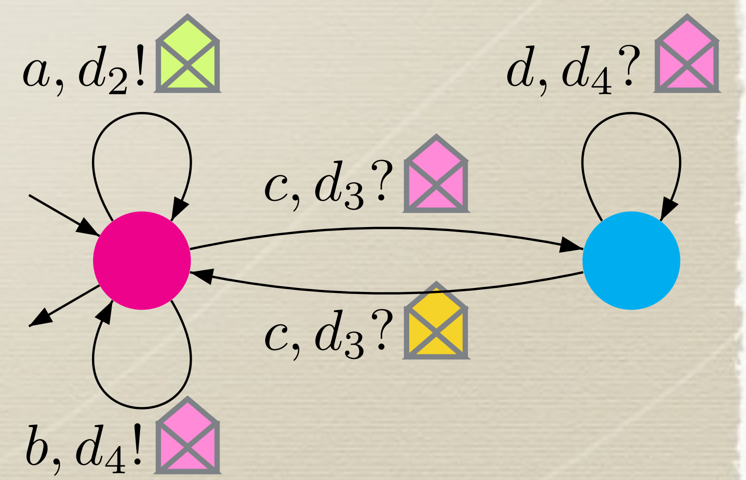


Semantics of CPDS on Graphs

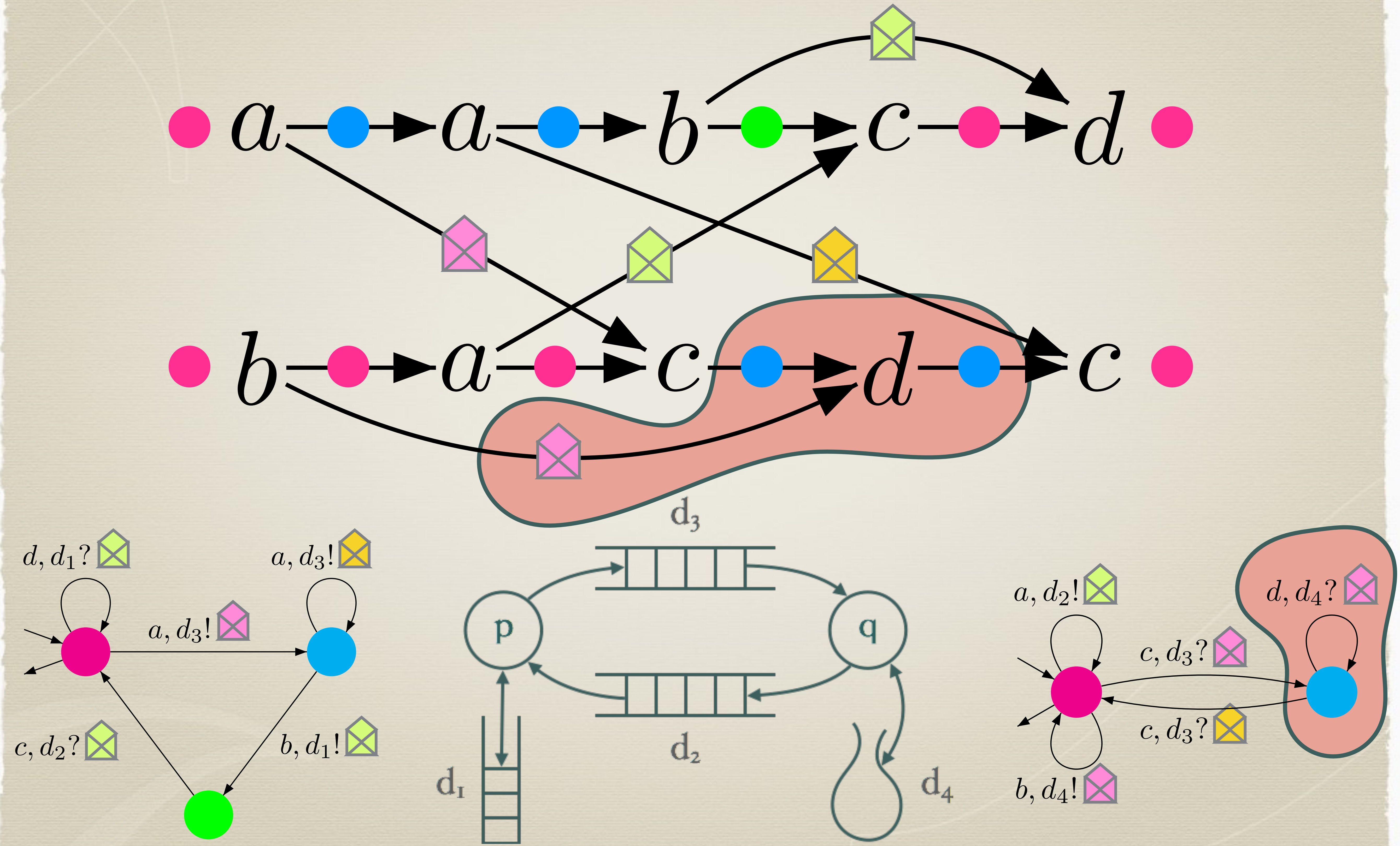








Semantics of CPDS on Graphs



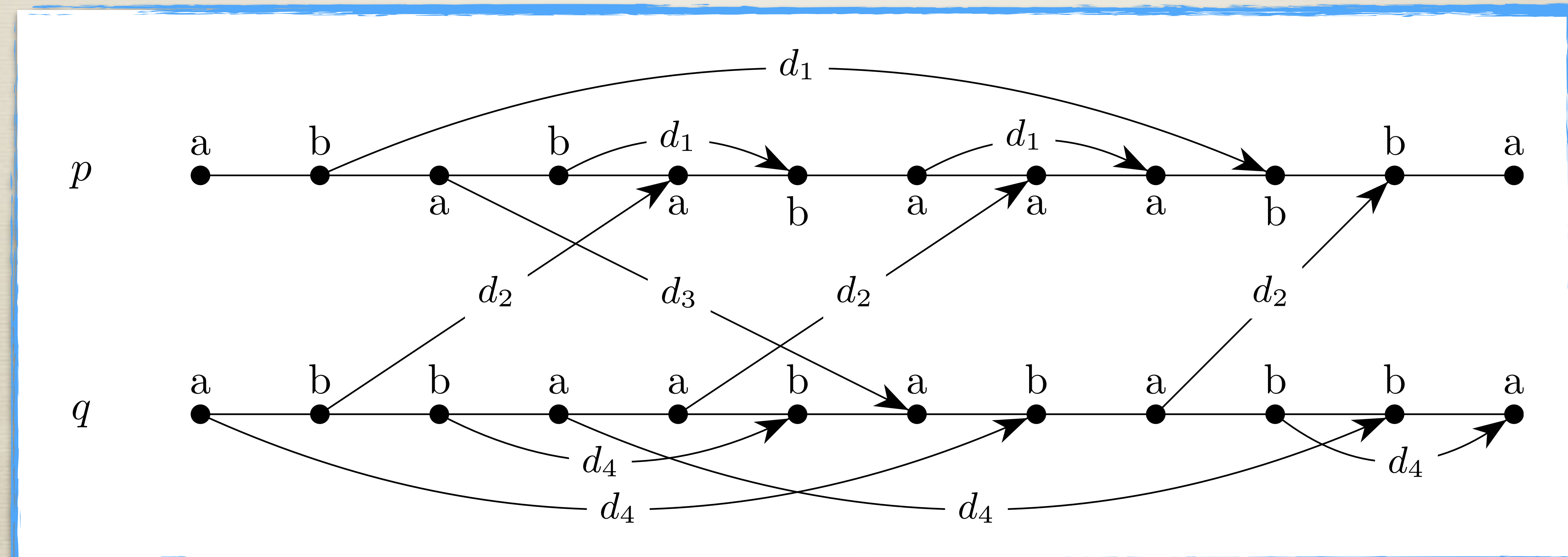
Outline

- ☑ 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

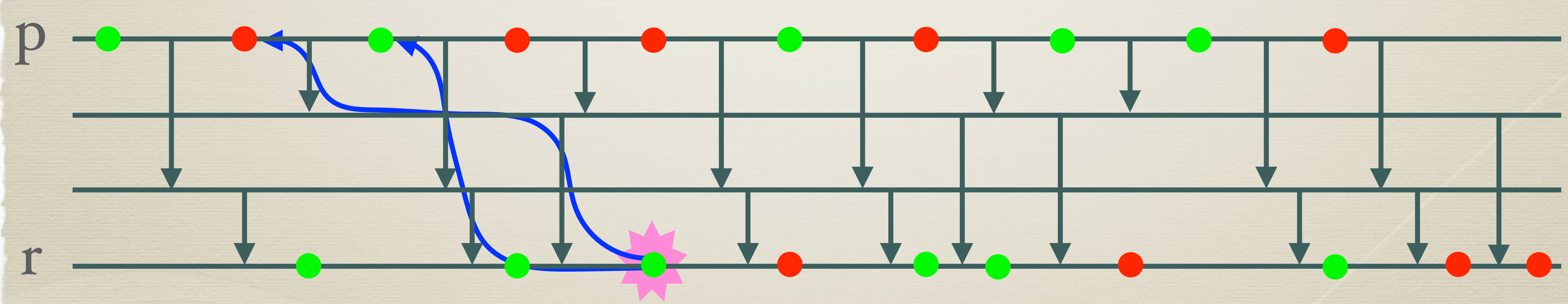
$$\varphi ::= \mathsf{false} \mid a(x) \mid p(x) \mid x \leq y \mid x \rhd^{d} y \mid x \rightarrow y$$
$$\mid x \in X \mid \varphi \vee \varphi \mid \neg\varphi \mid \exists x\, \varphi \mid \exists X\, \varphi$$



Specification over Graphs
Obey the latest order

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

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



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
* LTL specification are not always meaningful
  LTL \ X, Closure properties, ...
* Natural properties of graphs are difficult or impossible to express on linear traces

Specification over Linear Traces

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

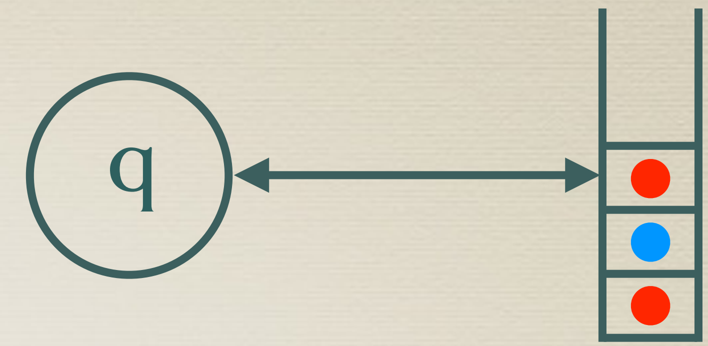
$(q_2, c_4!)(p, \mathsf{on}) ...$

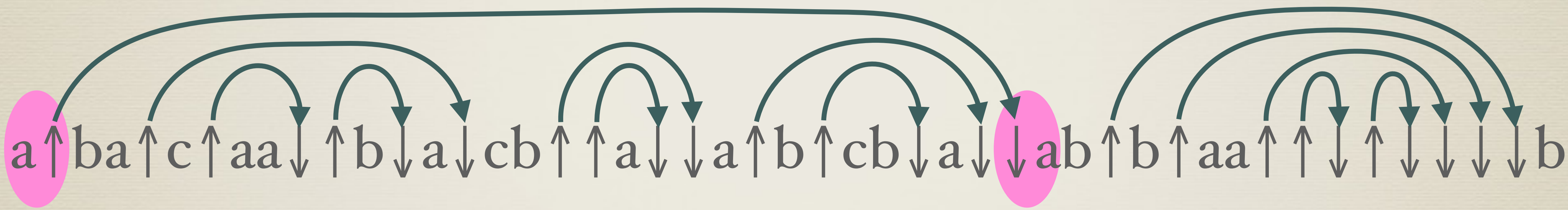
Obey the latest order
not expressible
in MSO over Linear Traces

* Based o ... word total order
* LTL ove ... 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

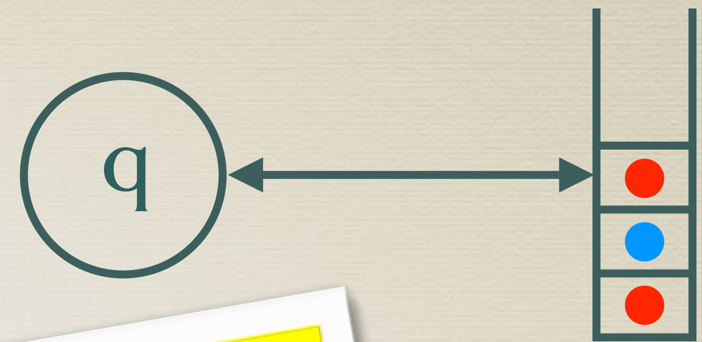




$$\forall x, y \left(\begin{array}{c} a(x-1) \wedge x \rhd y \wedge \\ \neg \exists z, z' \, (z \rhd z' \wedge z < x < z') \end{array} \right) \Rightarrow a(y+1)$$

Graphs for Sequential Systems

Answer the correct client
for topmost requests



Specifications should be on graphs

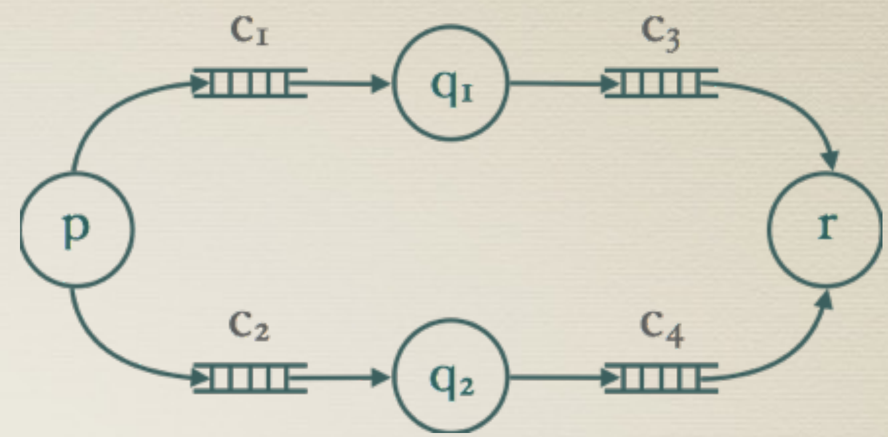
$a\uparrow ba\uparrow c\uparrow$ … $\downarrow\downarrow ab\uparrow b\uparrow aa\uparrow\uparrow\downarrow\uparrow\downarrow\downarrow\downarrow\downarrow b$

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

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

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

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

Model Checking vs Reachability

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

  ... when specifications can be translated to automata

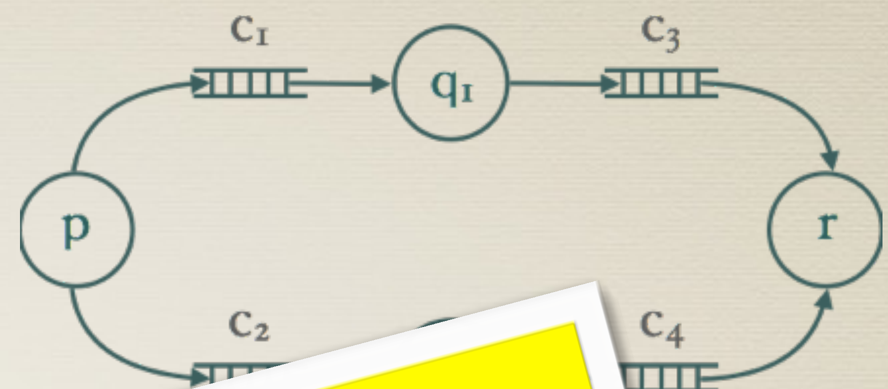
  ... this is **not possible** in general for graphs

$$S \models \phi$$

$$\mathcal{A}_S \qquad\qquad \mathcal{A}_{\neg\varphi}$$

$$\mathcal{A}_S \cap \mathcal{A}_{\neg\varphi} = \emptyset$$

Verification problems



* Emptiness or Reachability
* Inclusion or Universality
* Satisfiability ϕ
* Mod
* Tempo

undecidable in general

rder

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

* Propositional dynamic logics
* Monadic second order logic

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

Under-approximate Verification

Mainly for reachability

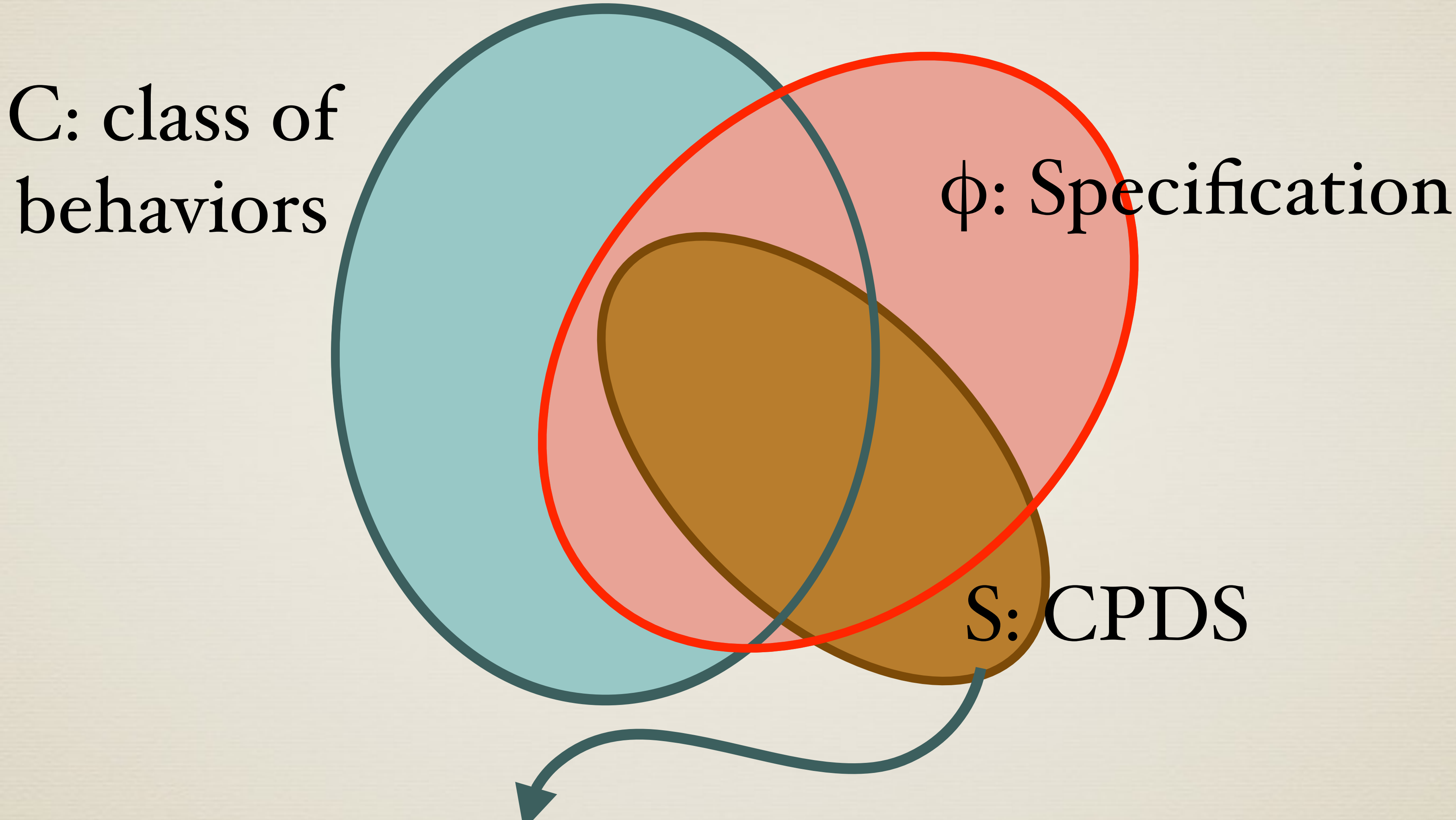
- Emptiness or Reachability
- Inclusion or Univ…
- …
- …g. $s \models \phi$
- Temporal logics
- Propositional dynamic logics
- Monadic second order logic

undecidable

- 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.]

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

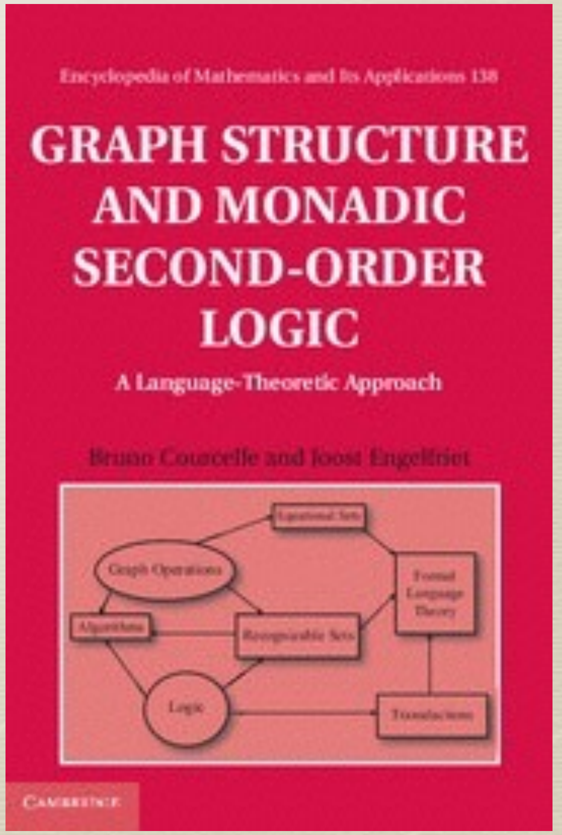
A Language-Theoretic Approach

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 Universality
* Satisfiability
* Model Checking: S
* 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.
* 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

bstract

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

$$\begin{tikzcd}
& \text{Split-Width } k \arrow[dl, "t \leq 2(k + |Procs|) - 1"'] \arrow[dr, "c \leq 2(k + |Procs|) + 1"] & \\
\text{Tree-Width } t & & \text{Clique-Width } c
\end{tikzcd}$$

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

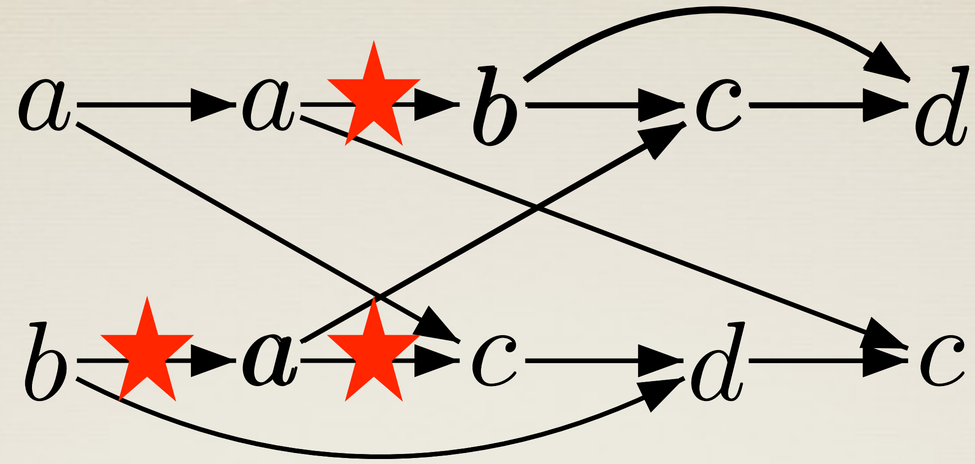
Let C be a class of bounded degree MSO definable graphs.
TFAE

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)

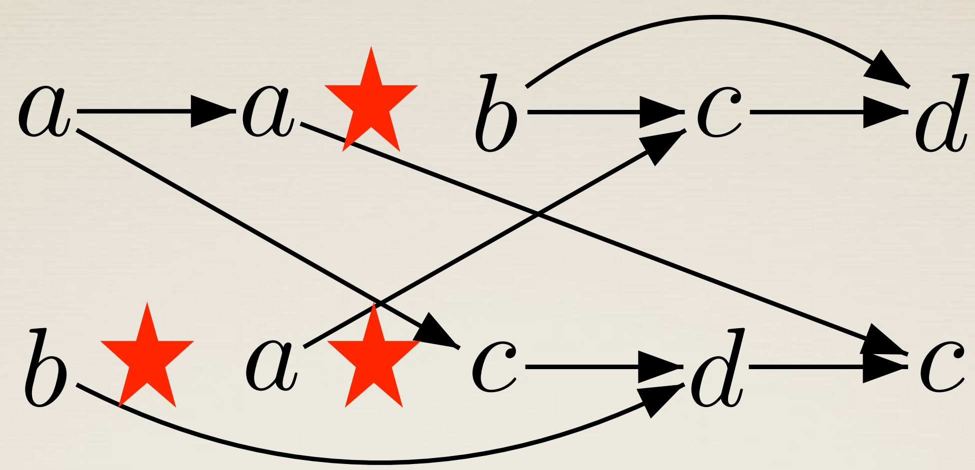
BUDGET

$$\begin{tikzcd}
a \arrow[r] \arrow[drr] & a \arrow[r] \arrow[drrr] & b \arrow[r] \arrow[rr, bend left] & c \arrow[r] & d \\
b \arrow[r] \arrow[rrr, bend right] & a \arrow[r] \arrow[urr] & c \arrow[r] & d \arrow[r] & c
\end{tikzcd}$$

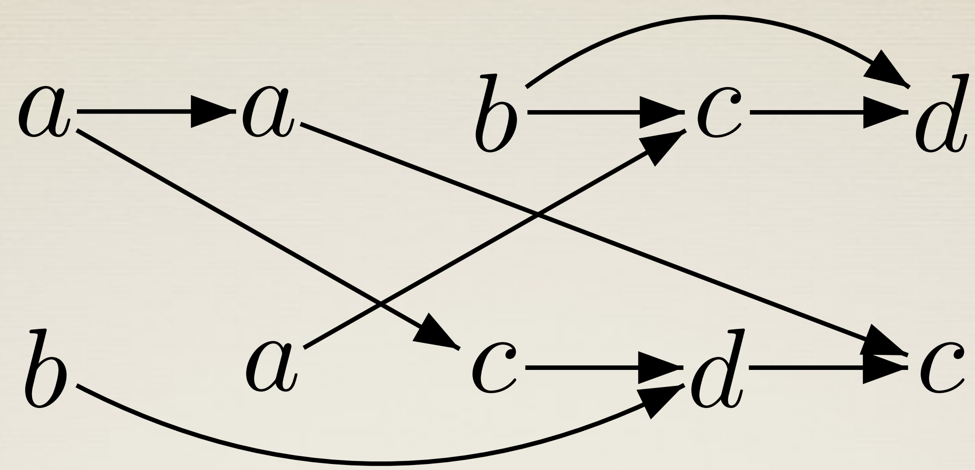
SPLIT DECOMPOSITION OF CBMs



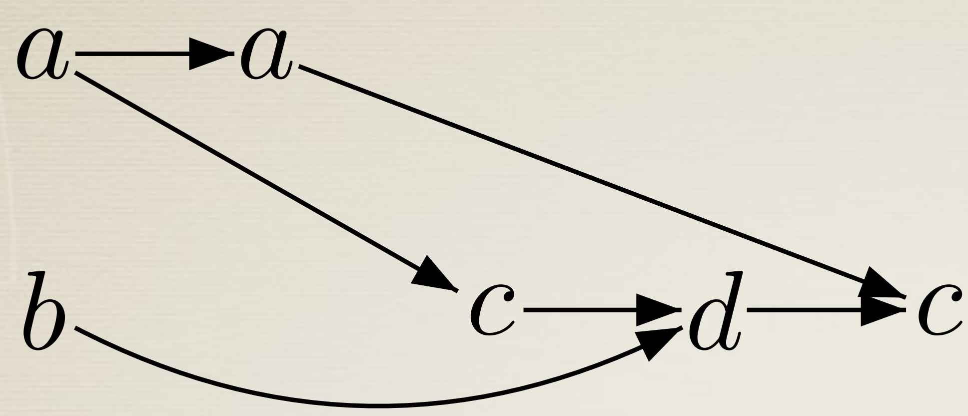
SPLIT DECOMPOSITION OF CBMs

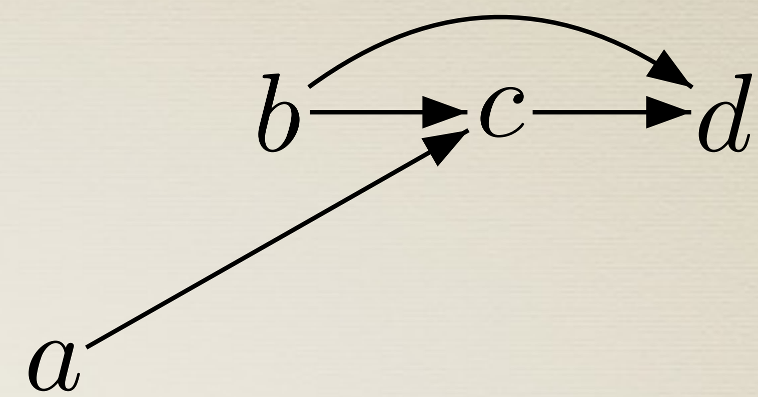


SPLIT DECOMPOSITION OF CBMs

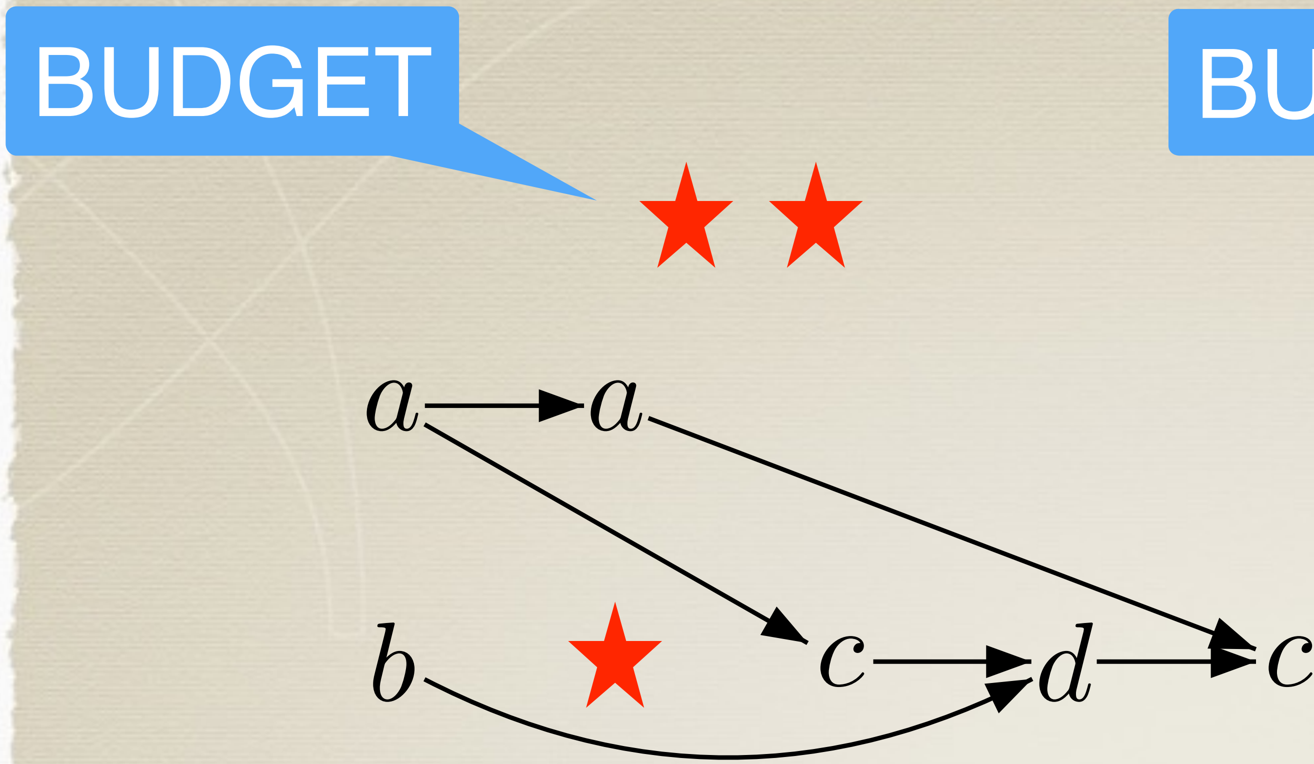


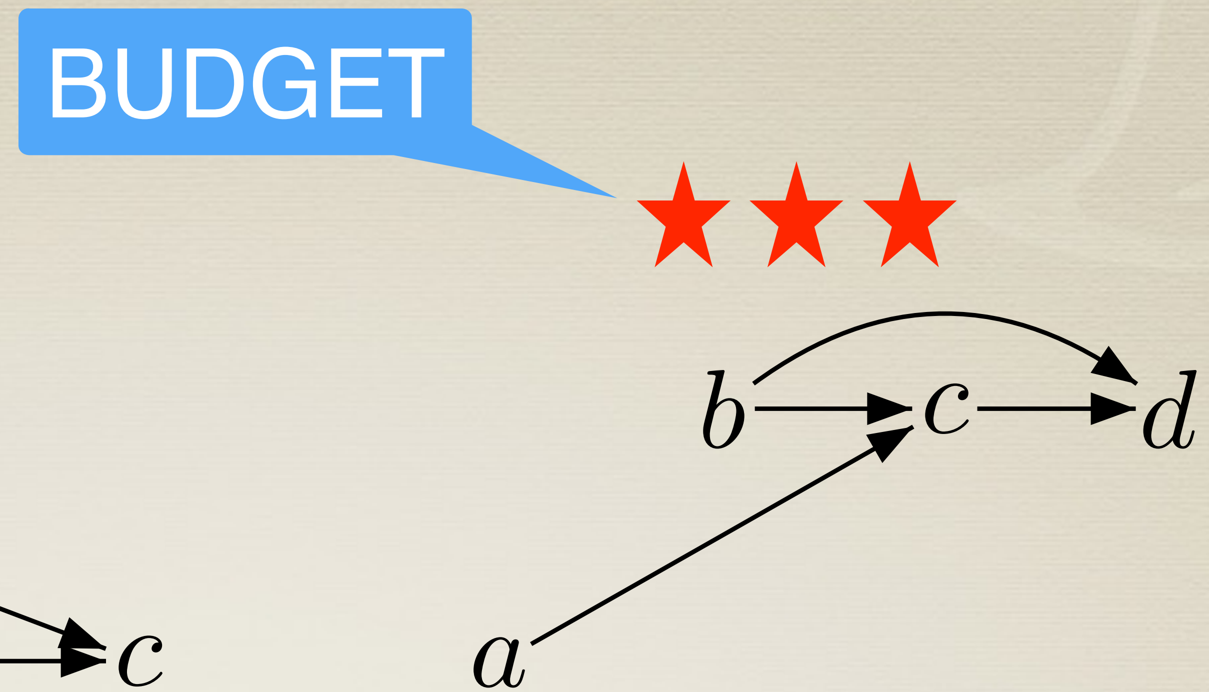
SPLIT DECOMPOSITION OF CBMs



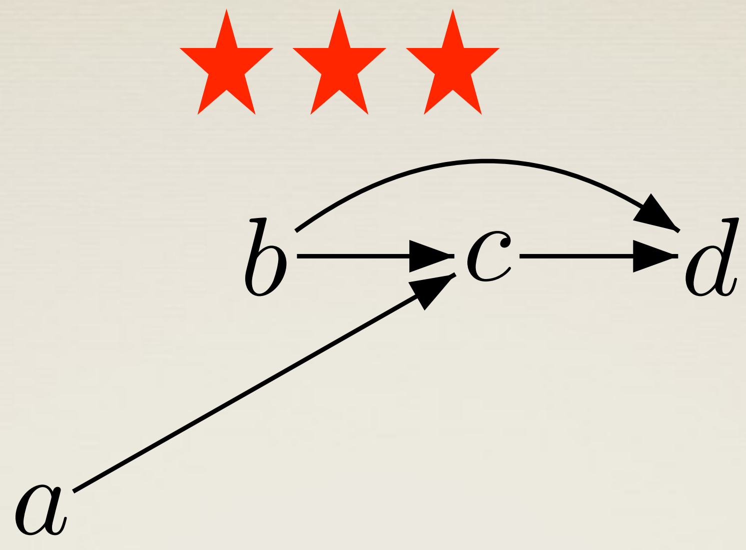


SPLIT DECOMPOSITION OF CBMs

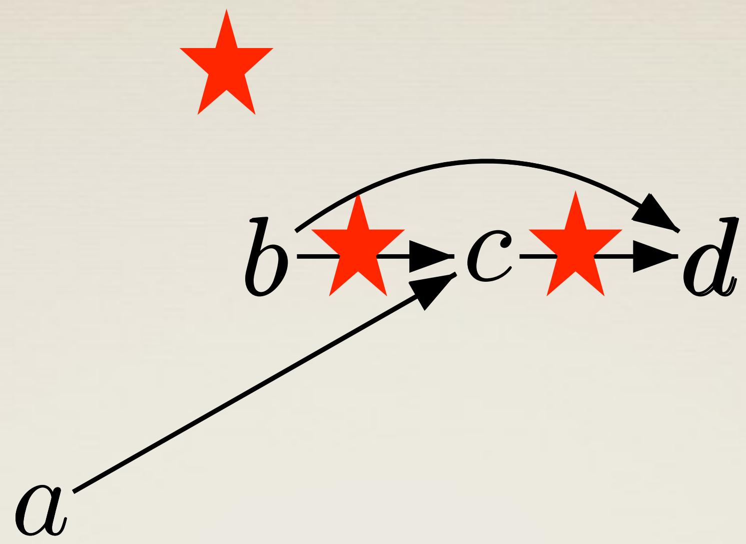




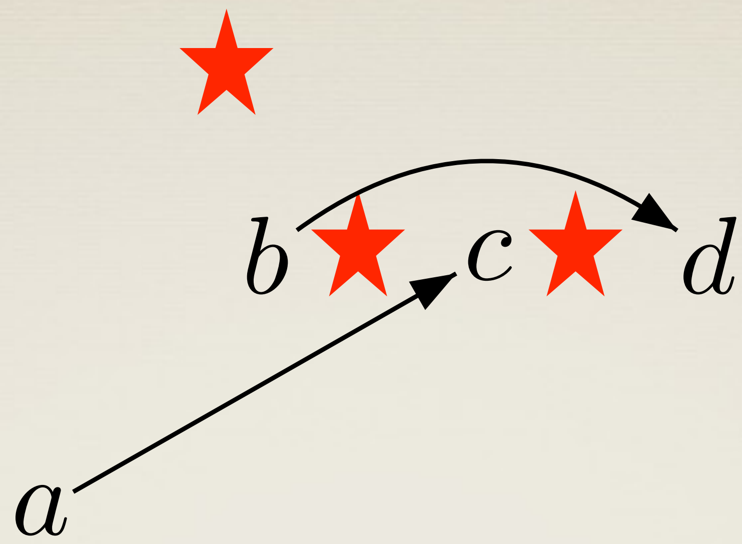
SPLIT DECOMPOSITION OF CBMs



SPLIT DECOMPOSITION OF CBMs

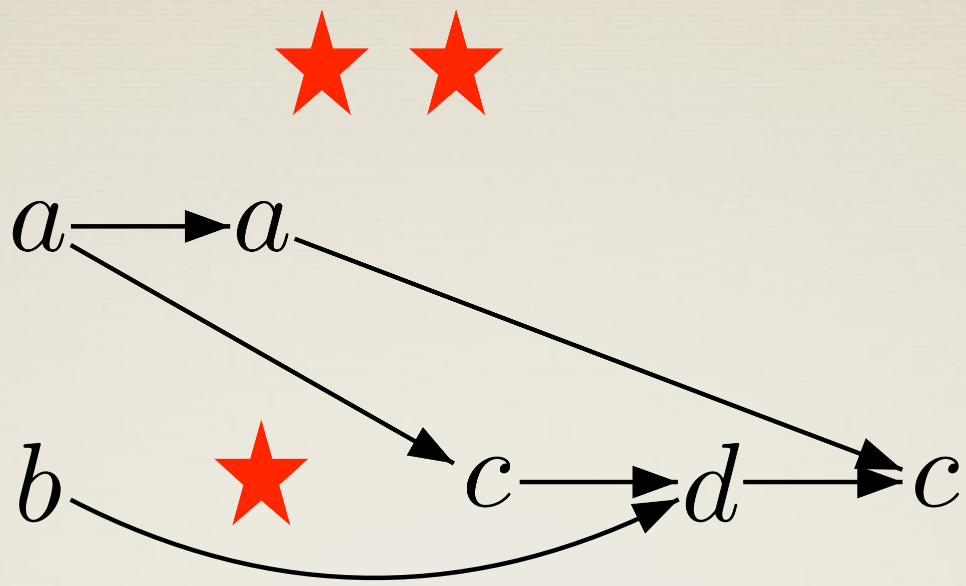


SPLIT DECOMPOSITION OF CBMs

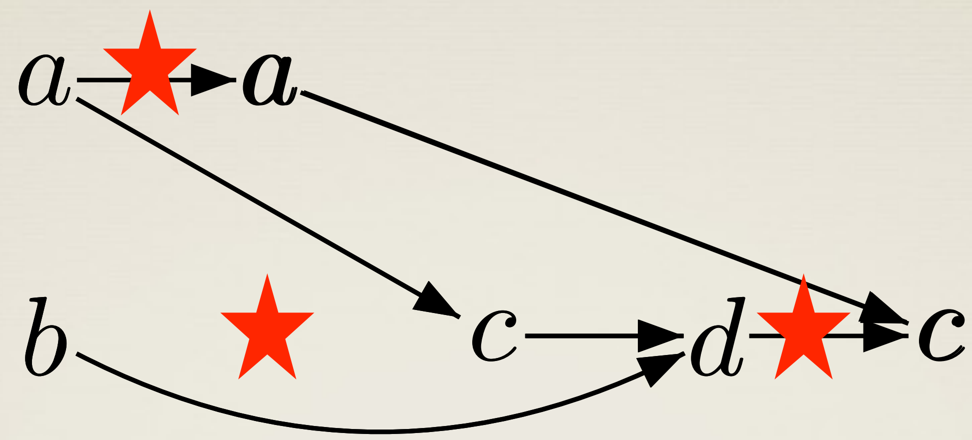


SPLIT DECOMPOSITION OF CBMs

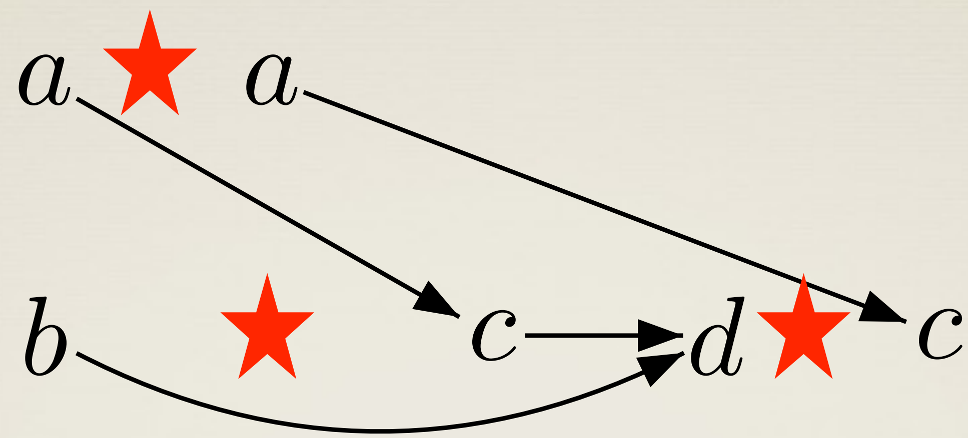
SPLIT DECOMPOSITION OF CBMs



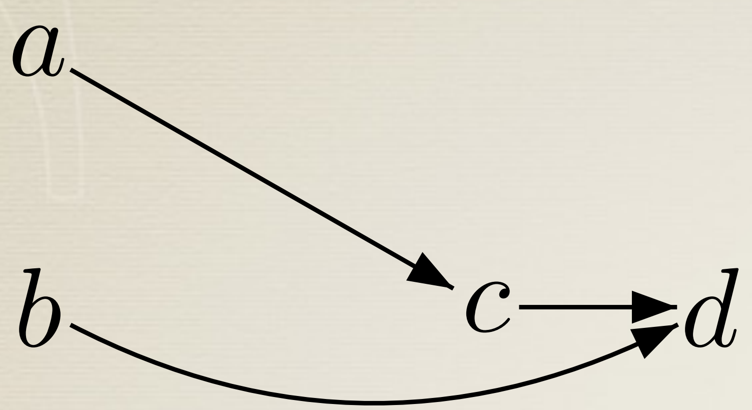
SPLIT DECOMPOSITION OF CBMs



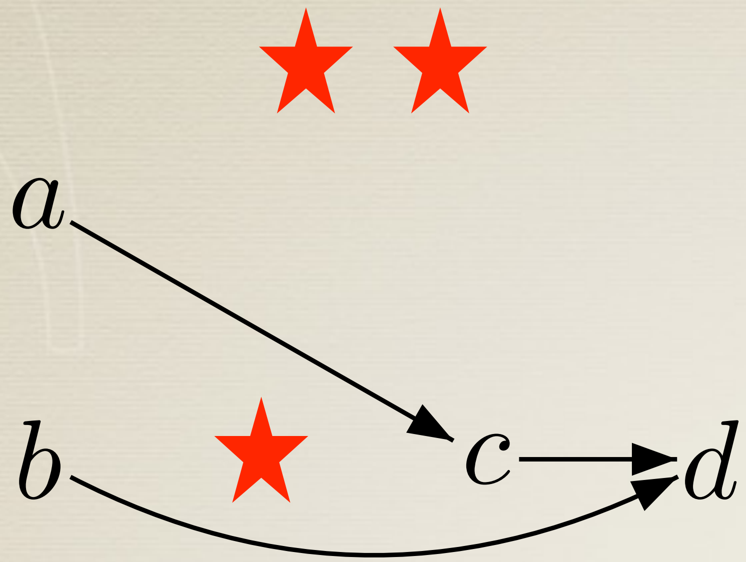
SPLIT DECOMPOSITION OF CBMs



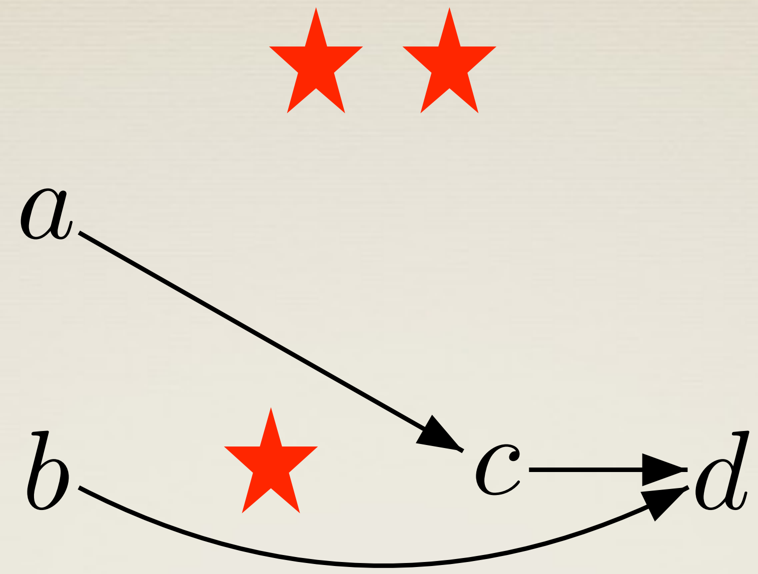
SPLIT DECOMPOSITION OF CBMs



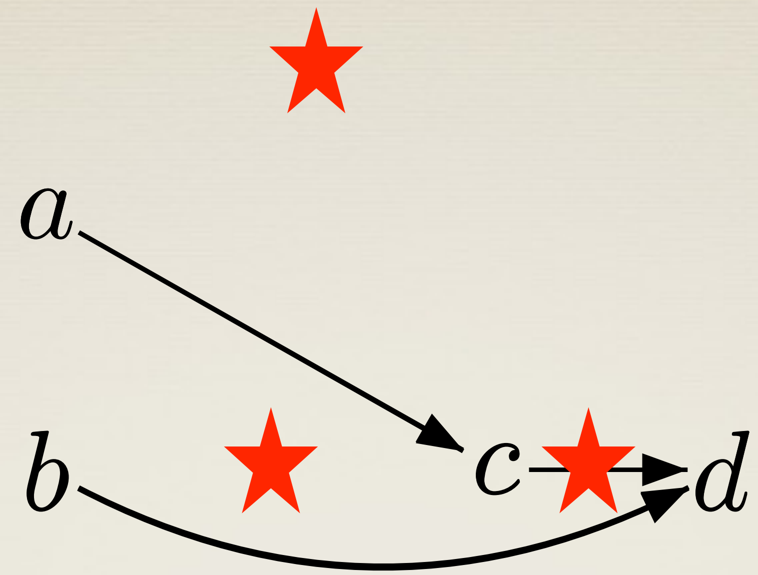
SPLIT DECOMPOSITION OF CBMs



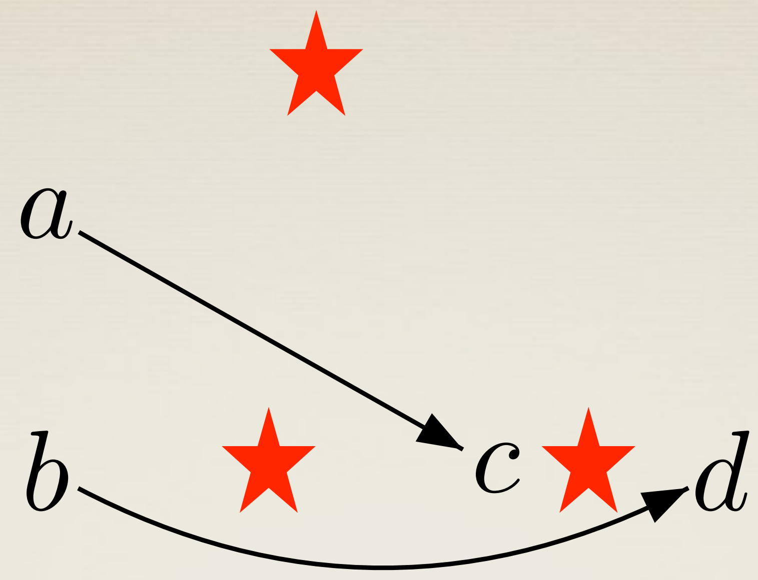
SPLIT DECOMPOSITION OF CBMs



SPLIT DECOMPOSITION OF CBMs



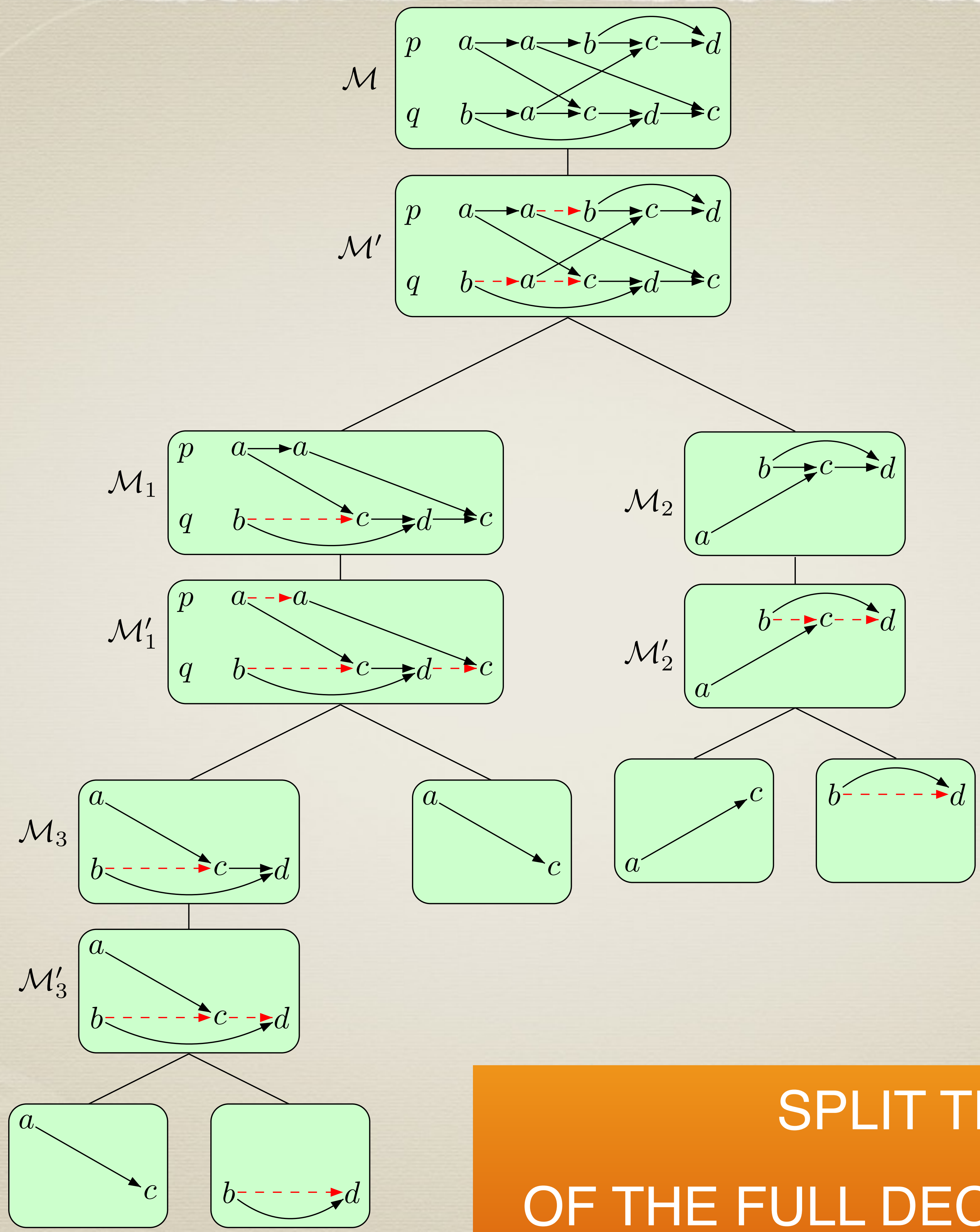
SPLIT DECOMPOSITION OF CBMs



SPLIT DECOMPOSITION OF CBMs

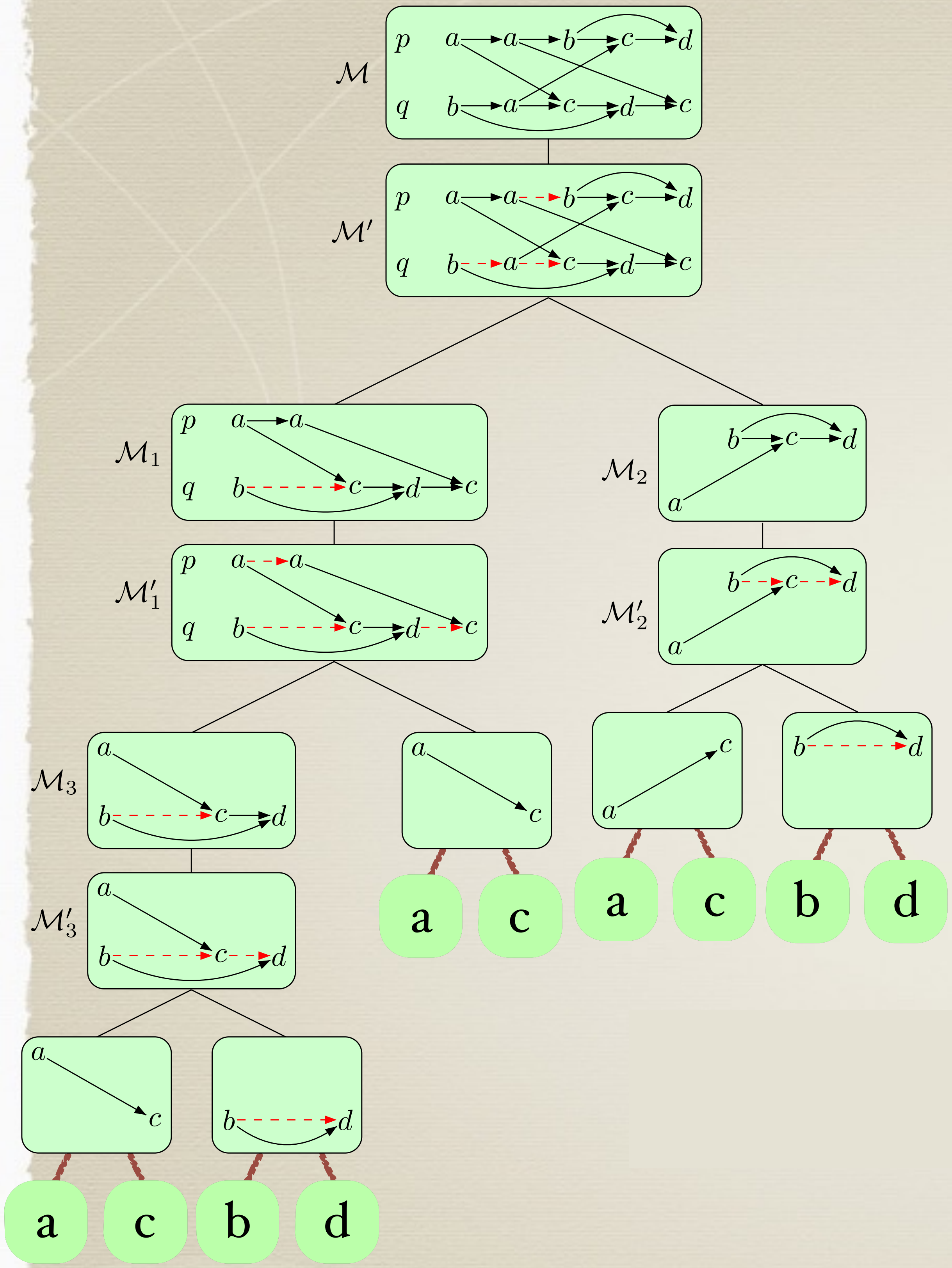
$$\begin{tikzcd}
a \arrow[dr] & & & \\
& c & b \arrow[r, bend right] & d
\end{tikzcd}$$

SPLIT DECOMPOSITION OF CBMs



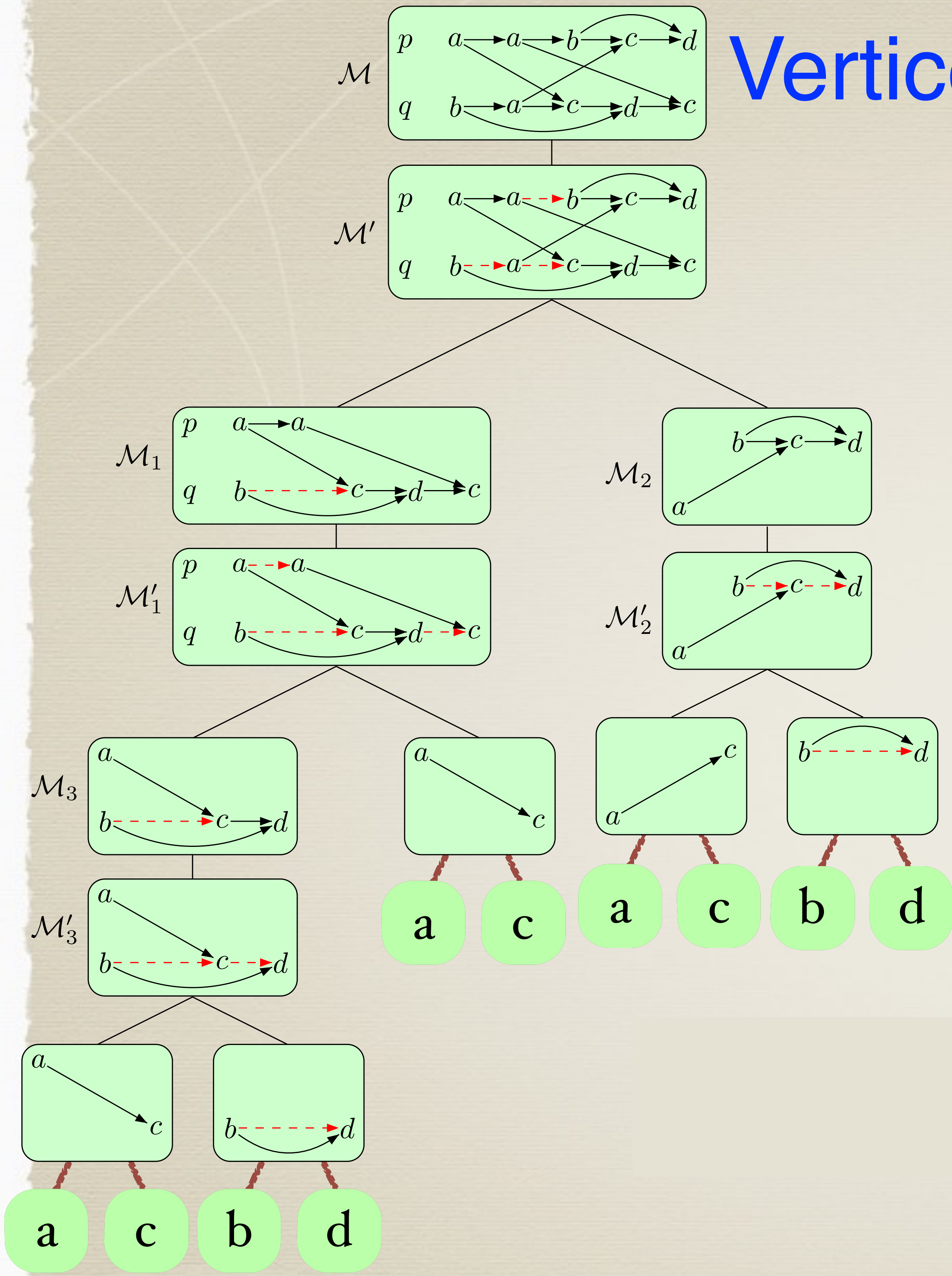
SPLIT TREE

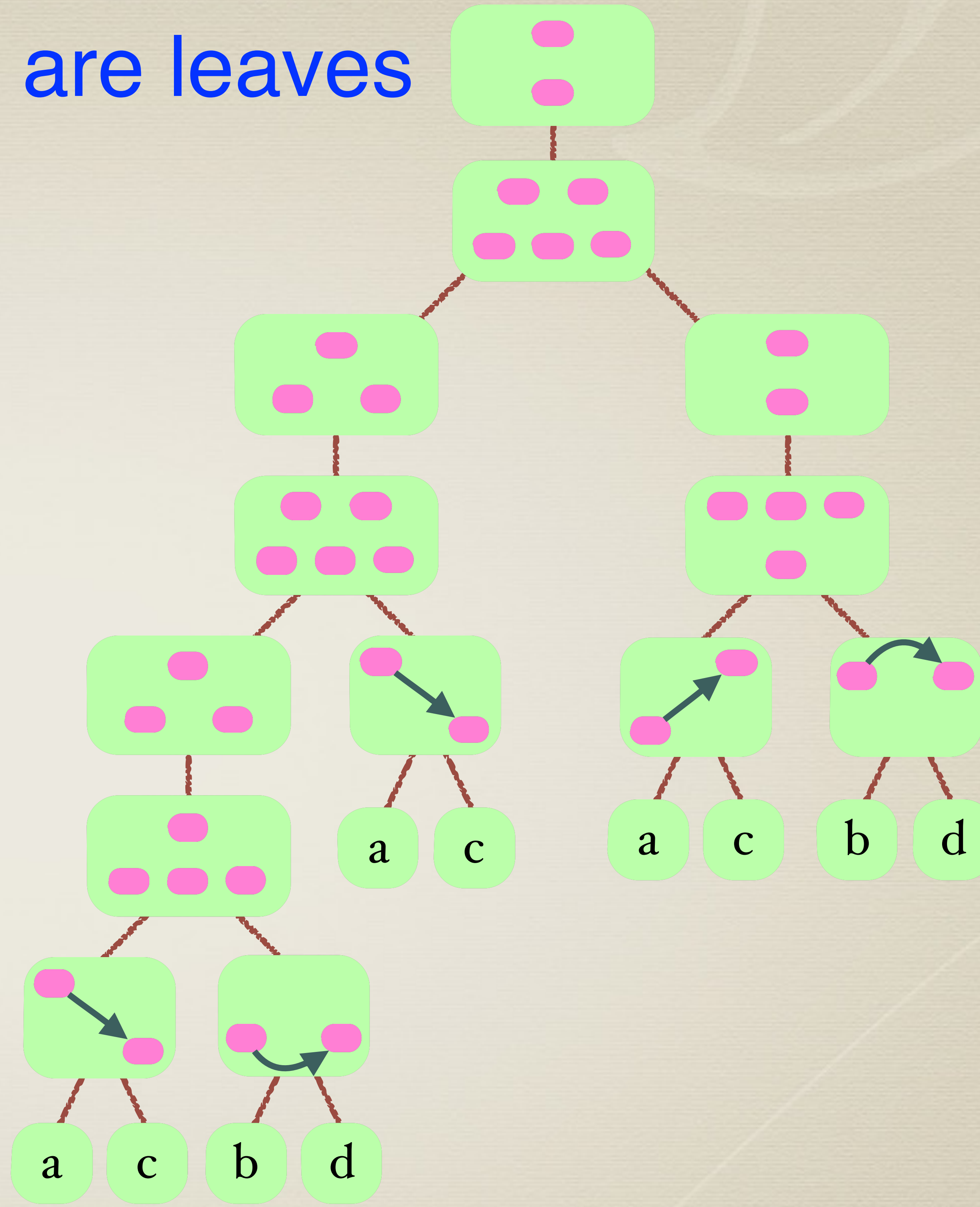
OF THE FULL DECOMPOSITION



Tree interpretation in Abstract Tree Decomposition

Vertices are leaves

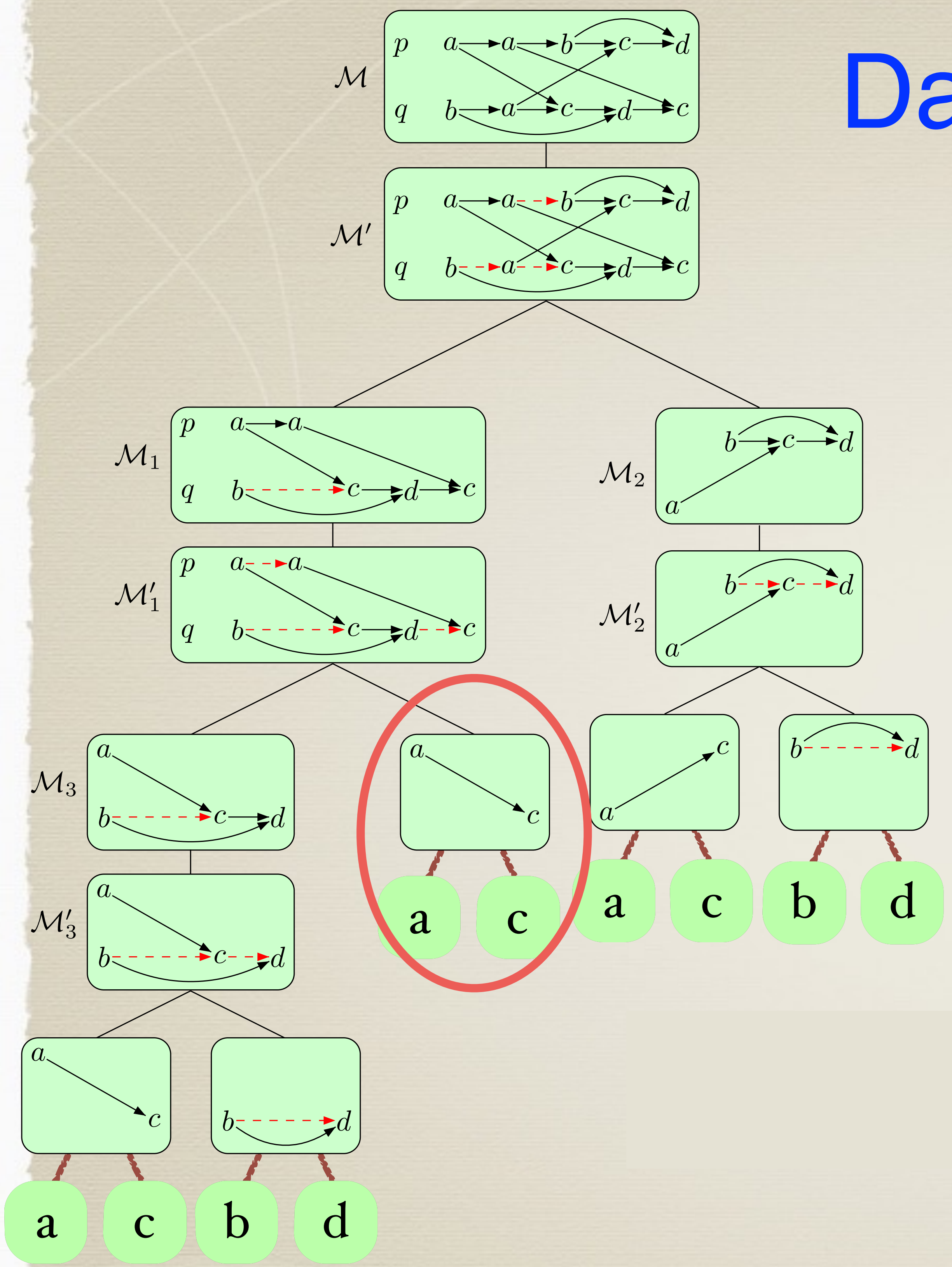


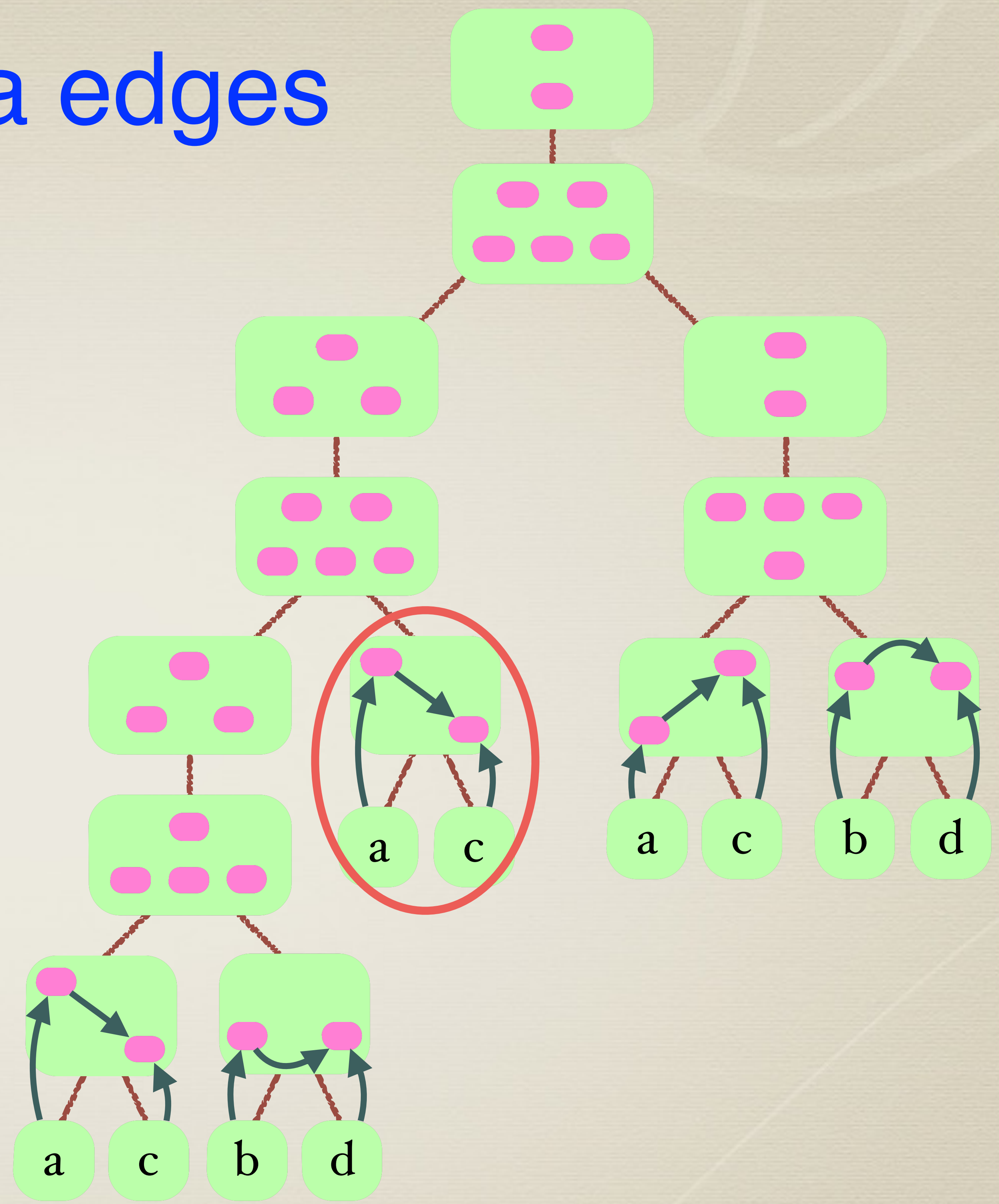


Tree interpretation in

Abstract Tree Decomposition

Data edges

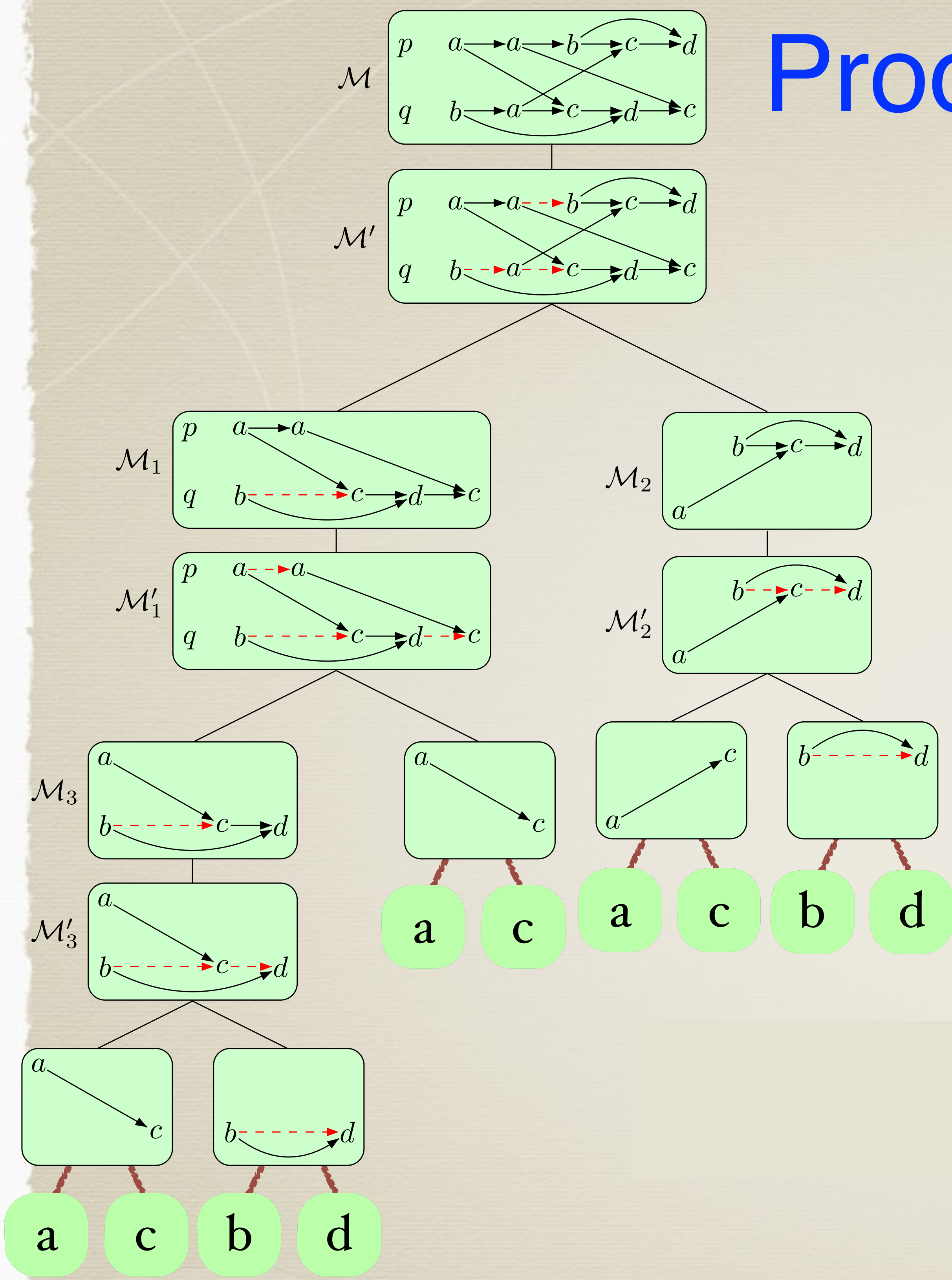


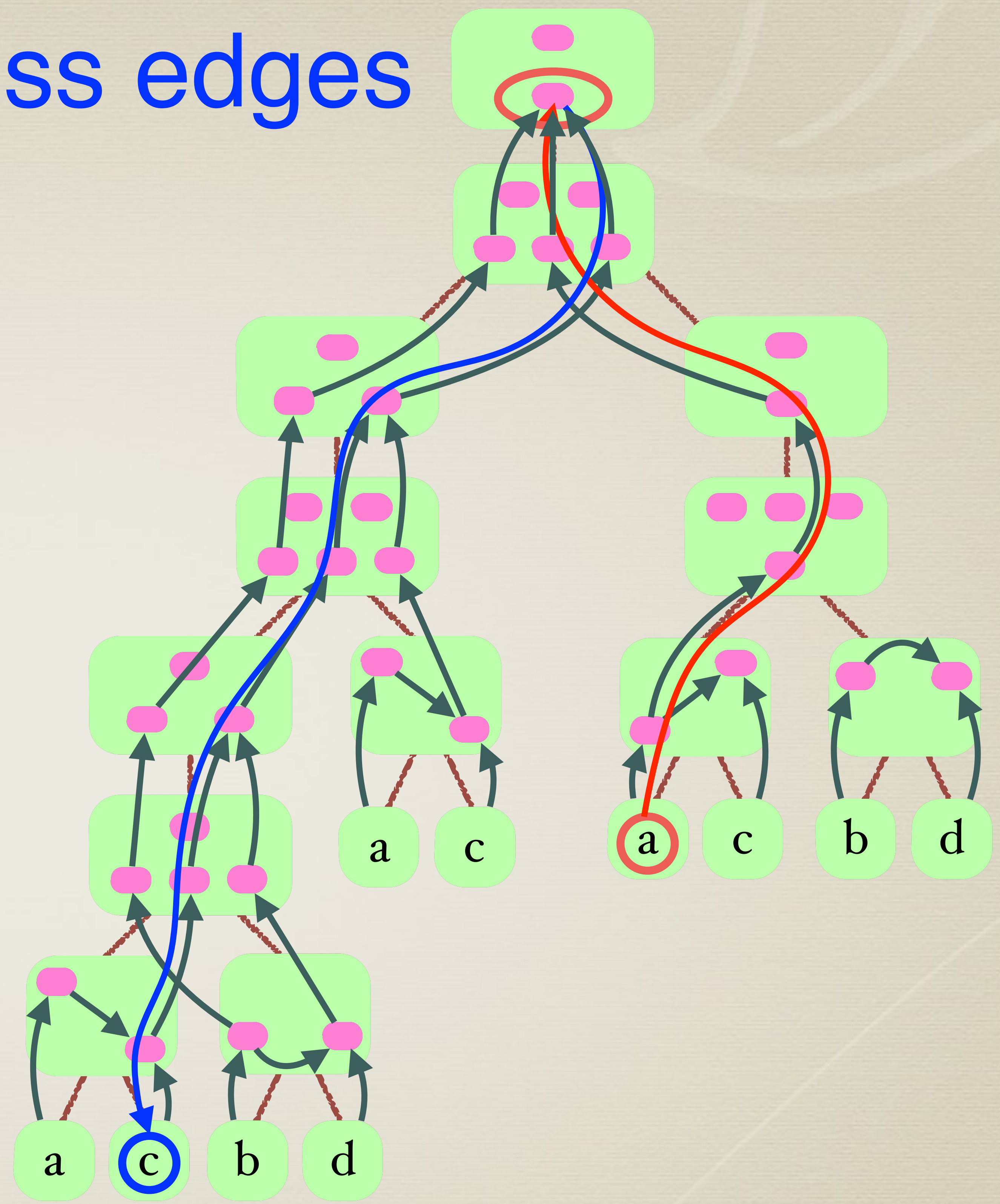


Tree interpretation in

Abstract Tree Decomposition

Process edges





Tree interpretation in

Abstract Tree Decomposition

Nested Words: split-width ≤ 2

$$w ::= \bullet \mid w \longrightarrow w \mid \bullet \longrightarrow w \longrightarrow \bullet \mid \bullet \longrightarrow \bullet$$

Split-width: under-approximations

* Words
* Nested Words
* Acyclic Architectures

Constant

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

Bound + 2

* Bounded phase
* Priority ordering

2^{Bound}

Split-width: parametrized verification

Problem	Complexity: bound on split-width part of the input (in unary)	Complexity: 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 • Requires syntactical or semantical restrictions to be meaningful	• Powerful specifications • Interactions are built-in • Meaningful

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
  * Timed systems
  * Dynamic creation of processes
  * Read from many
  * Infinite behaviors
  * ...
* Tools

THANK YOU