NETS WITH BOUNDARIES

Pawel Sobocinski, University of Southampton

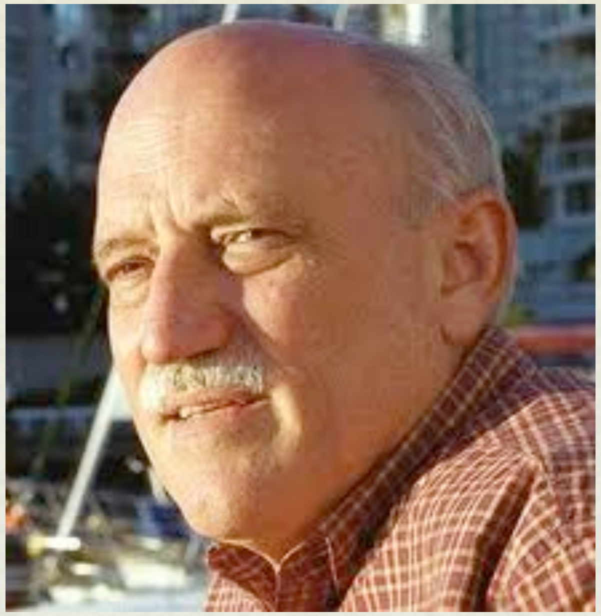
IFIP WG2.2, Lisbon, 24/09/2013

joint work with R. Bruni, H. Melgratti, U. Montanari, J.Rathke, O. Stephens

ABOUT ME

- Undergraduate at Sydney Uni, worked with RFC Walters and Steve Lack
- PhD (2004) at BRICS, Aarhus, supervised by Mogens Nielsen. Thesis on adhesive categories and relative pushouts for deriving LTS semantics from reduction semantics.
- **Keywords**: Concurrency, process calculi, graph transformation, semantics of programming languages, category theory, model checking, concurrent programming

THIS TALK



- RFC Walters
    - in concurrency, what is important is to discover the right **algebra**

- Robin Milner
    - in concurrency, what is important is the notion of **process**

ROADMAP

- **Automata as model of concurrency - Span(Graph)**
- Nets with boundaries
- Application to model checking
- Work in progress and future work

AUTOMATA AS MODEL OF CONCURRENCY

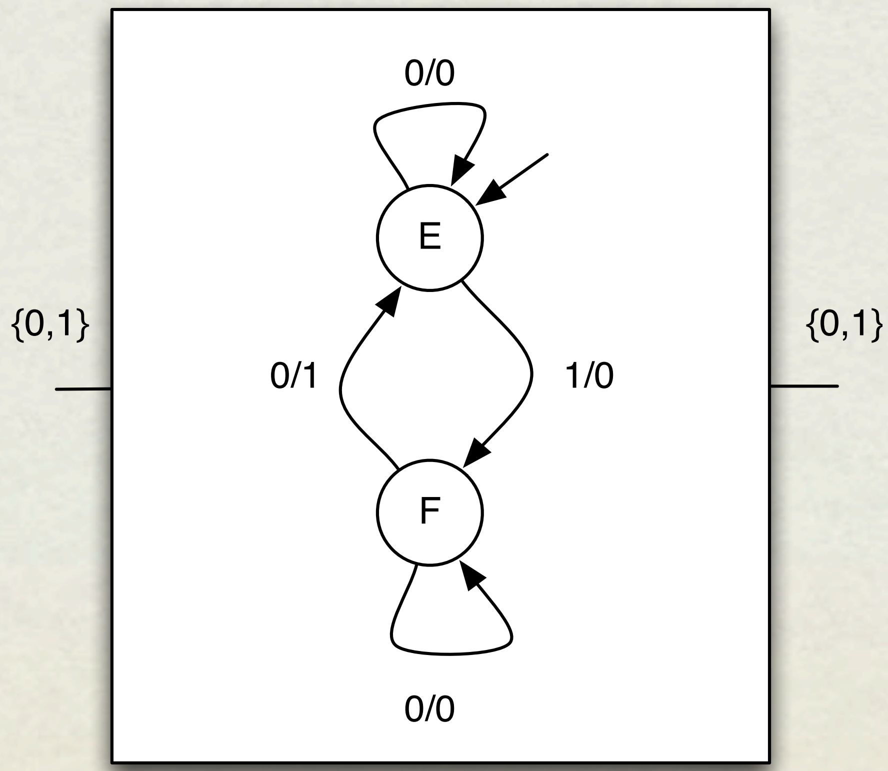
Nivat’s processes and their synchronization

André Arnold

Universite de Bordeaux I, LABRI, CNRS UMR 5800, 351 cours de la Liberation, F-33405 Talence, France

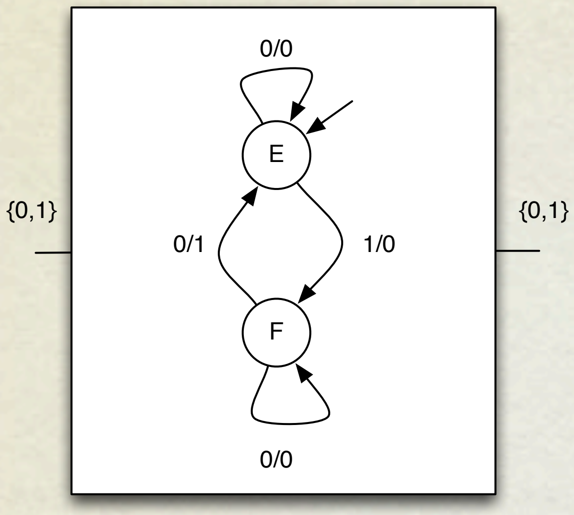
Maurice Nivat, André Arnold

Span(Graph) algebra - RFC Walters

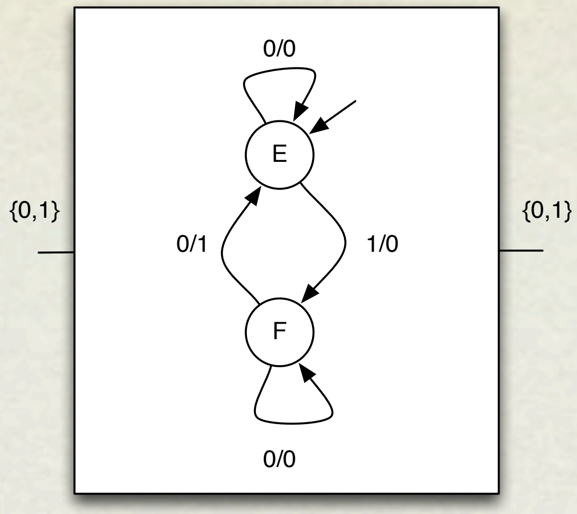


$B_E : (1,1)$

SYNCHRONISATION

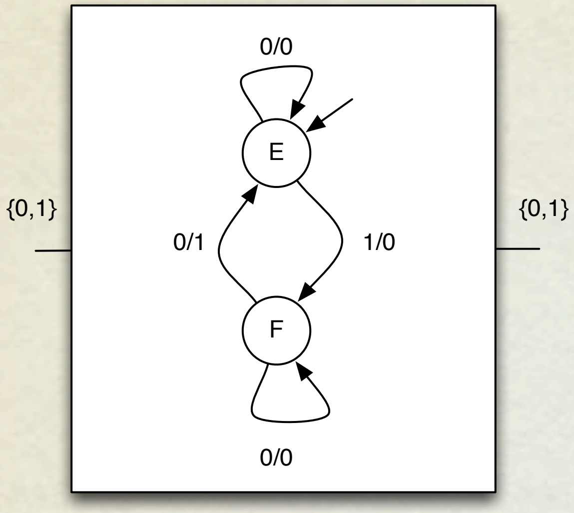


B_E : (1,1)

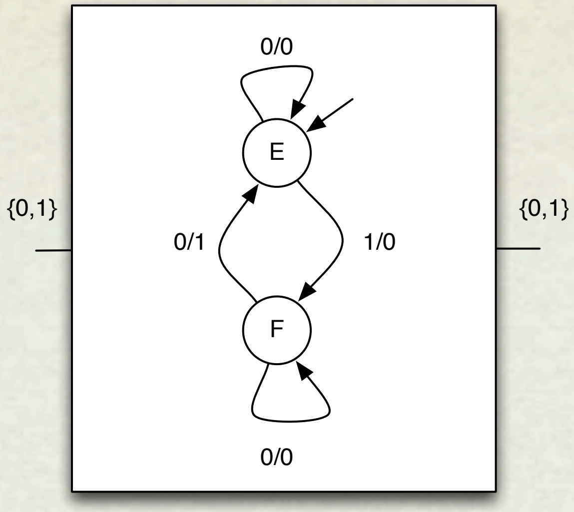


B_E : (1,1)

SYNCHRONISATION



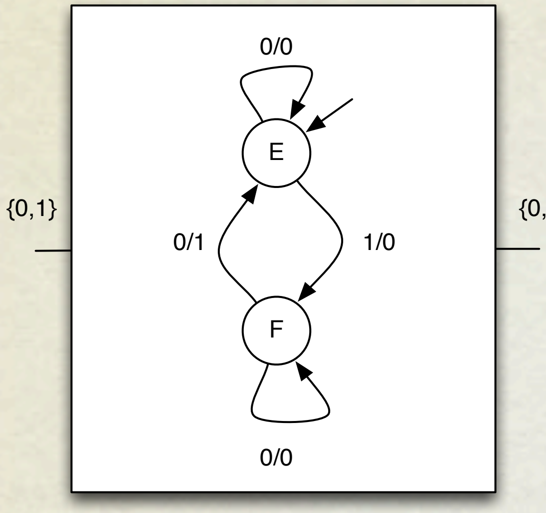
$B_E : (1,1)$



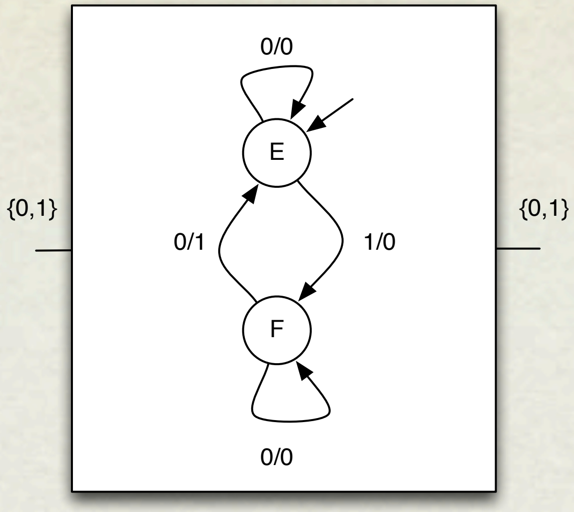
$B_E : (1,1)$

$$\frac{P \xrightarrow[\vec{c}]{\vec{a}} Q \qquad R \xrightarrow[\vec{b}]{\vec{c}} S}{P;R \xrightarrow[\vec{b}]{\vec{a}} Q;S} \; (\textsc{Cut})$$

SYNCHRONISATION

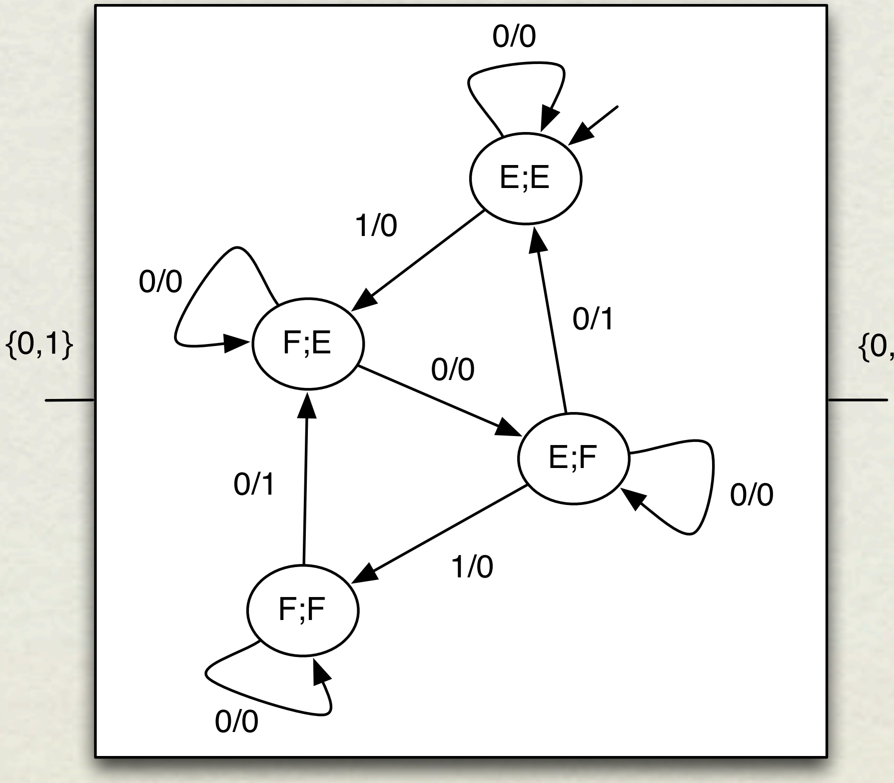


B_E : (1,1)



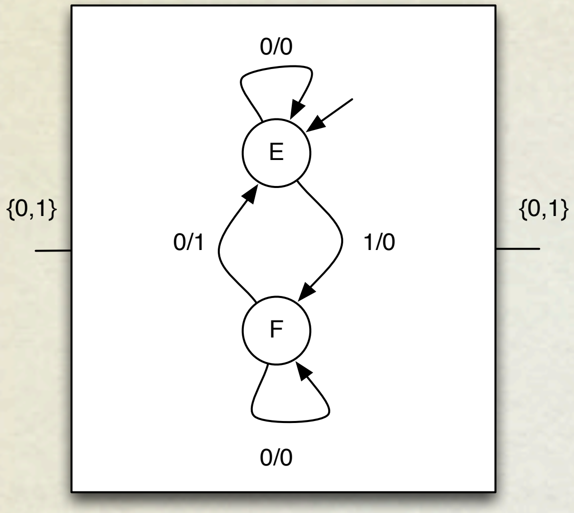
B_E : (1,1)

$$\frac{P \xrightarrow[\vec{c}]{\vec{a}} Q \qquad R \xrightarrow[\vec{b}]{\vec{c}} S}{P;R \xrightarrow[\vec{b}]{\vec{a}} Q;S} \;(\textsc{Cut})$$

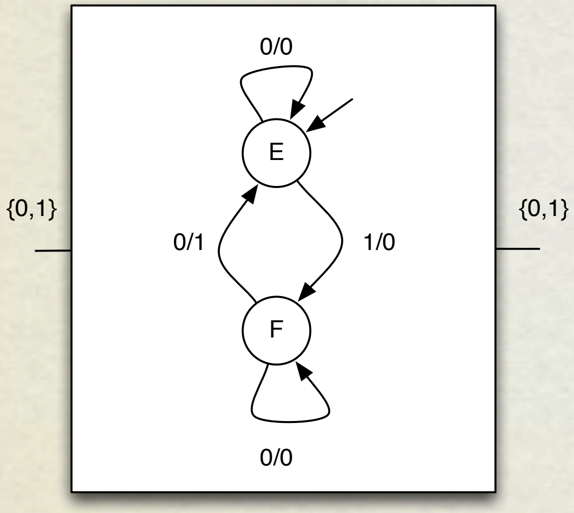


B_E ; B_E: (1,1)

TENSOR PRODUCT

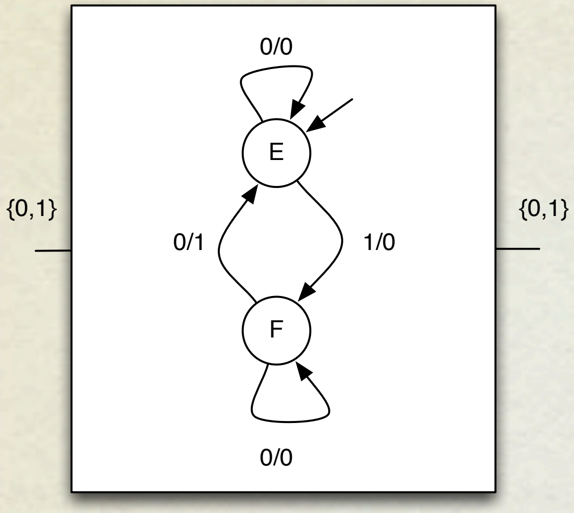


$B_E : (1,1)$

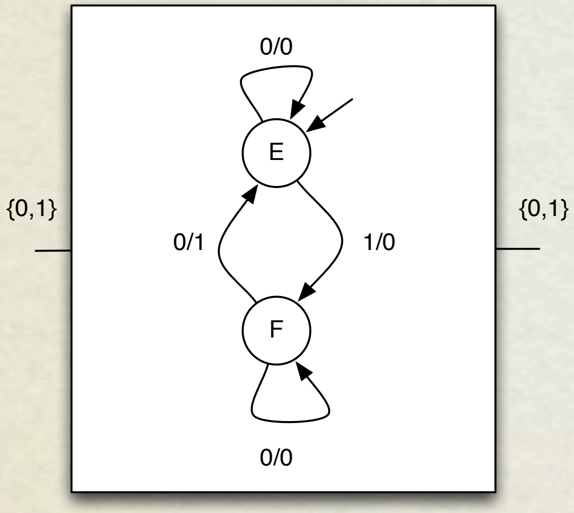


$B_E : (1,1)$

TENSOR PRODUCT



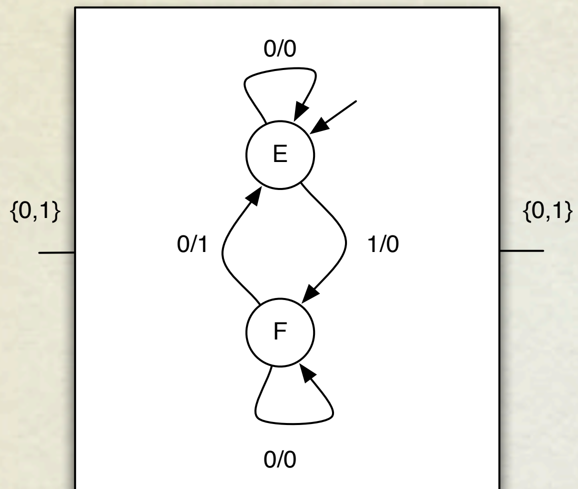
$B_E : (1,1)$



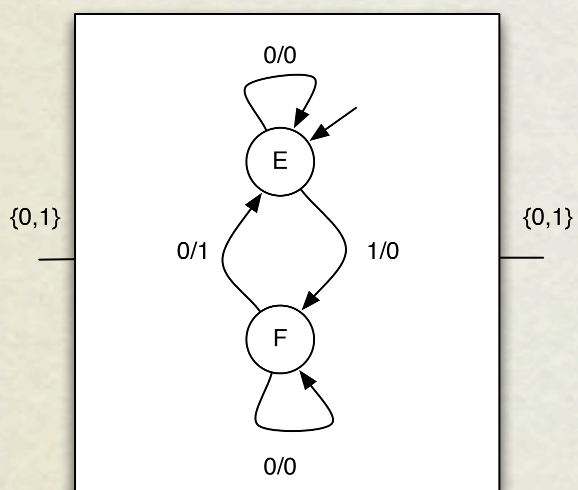
$B_E : (1,1)$

$$\frac{P \xrightarrow[\vec{b}]{\vec{a}} Q \qquad R \xrightarrow[\vec{d}]{\vec{c}} S}{P \otimes R \xrightarrow[\vec{b}\,\vec{d}]{\vec{a}\,\vec{c}} Q \otimes S} \ (\textsc{Ten})$$

TENSOR PRODUCT

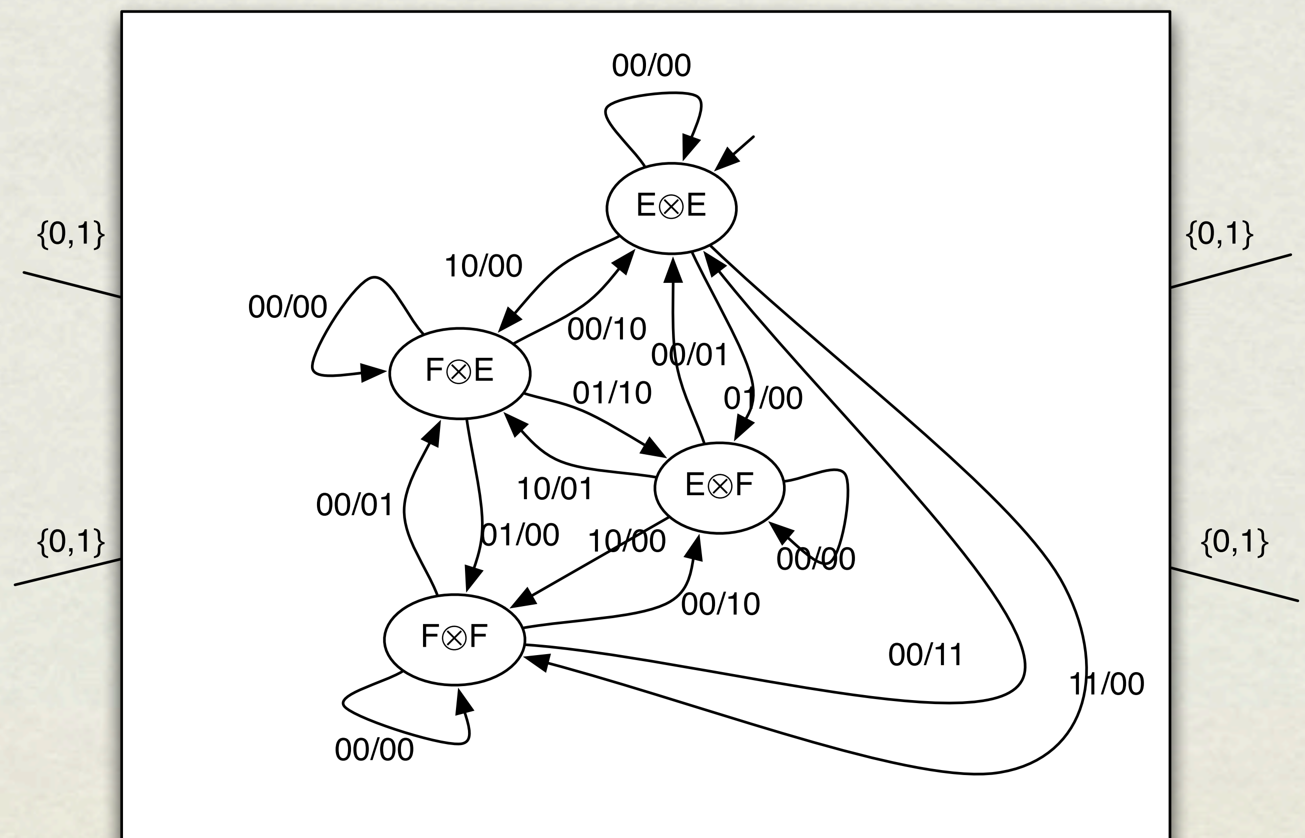


B_E : (1,1)



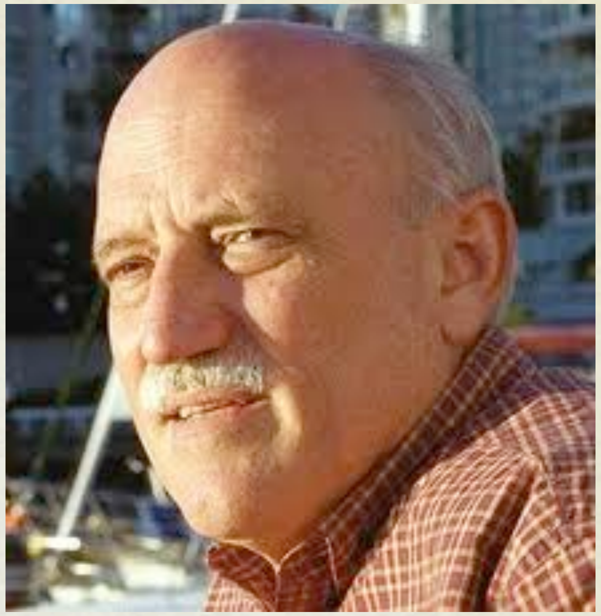
B_E : (1,1)

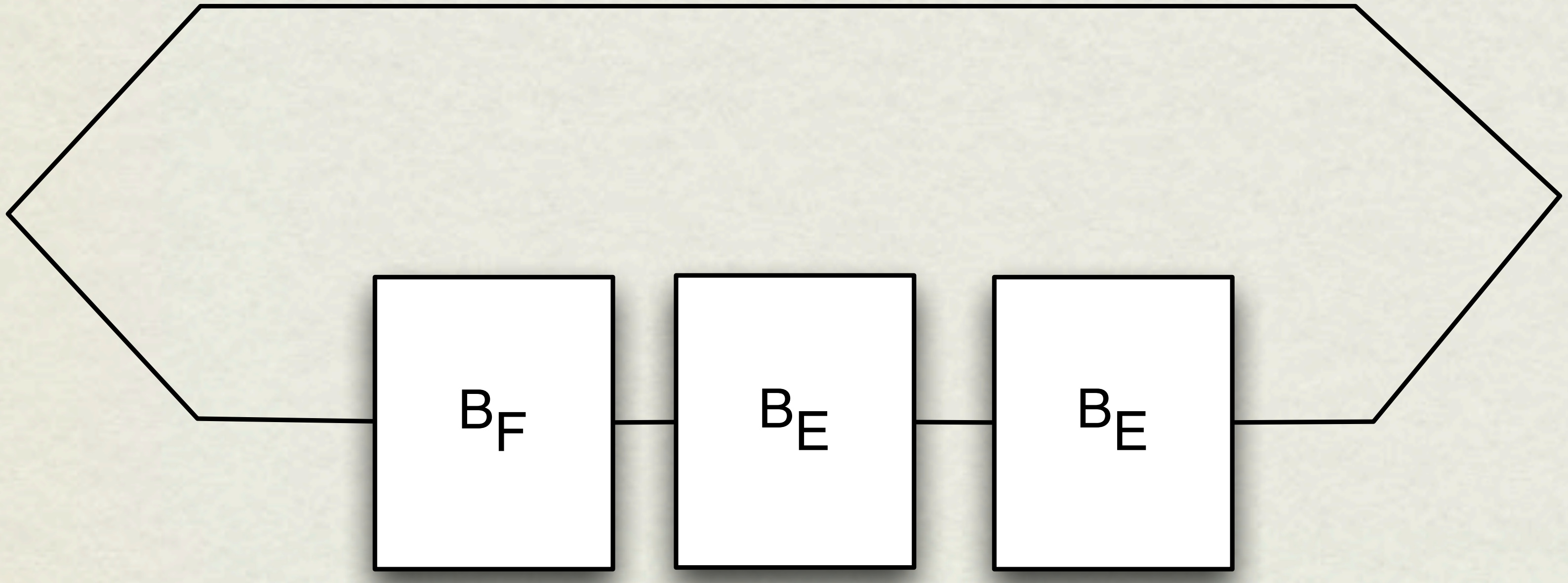
$$\frac{P \xrightarrow[\vec{b}]{\vec{a}} Q \qquad R \xrightarrow[\vec{d}]{\vec{c}} S}{P \otimes R \xrightarrow[\vec{b}\,\vec{d}]{\vec{a}\,\vec{c}} Q \otimes S} \ (\textsc{Ten})$$



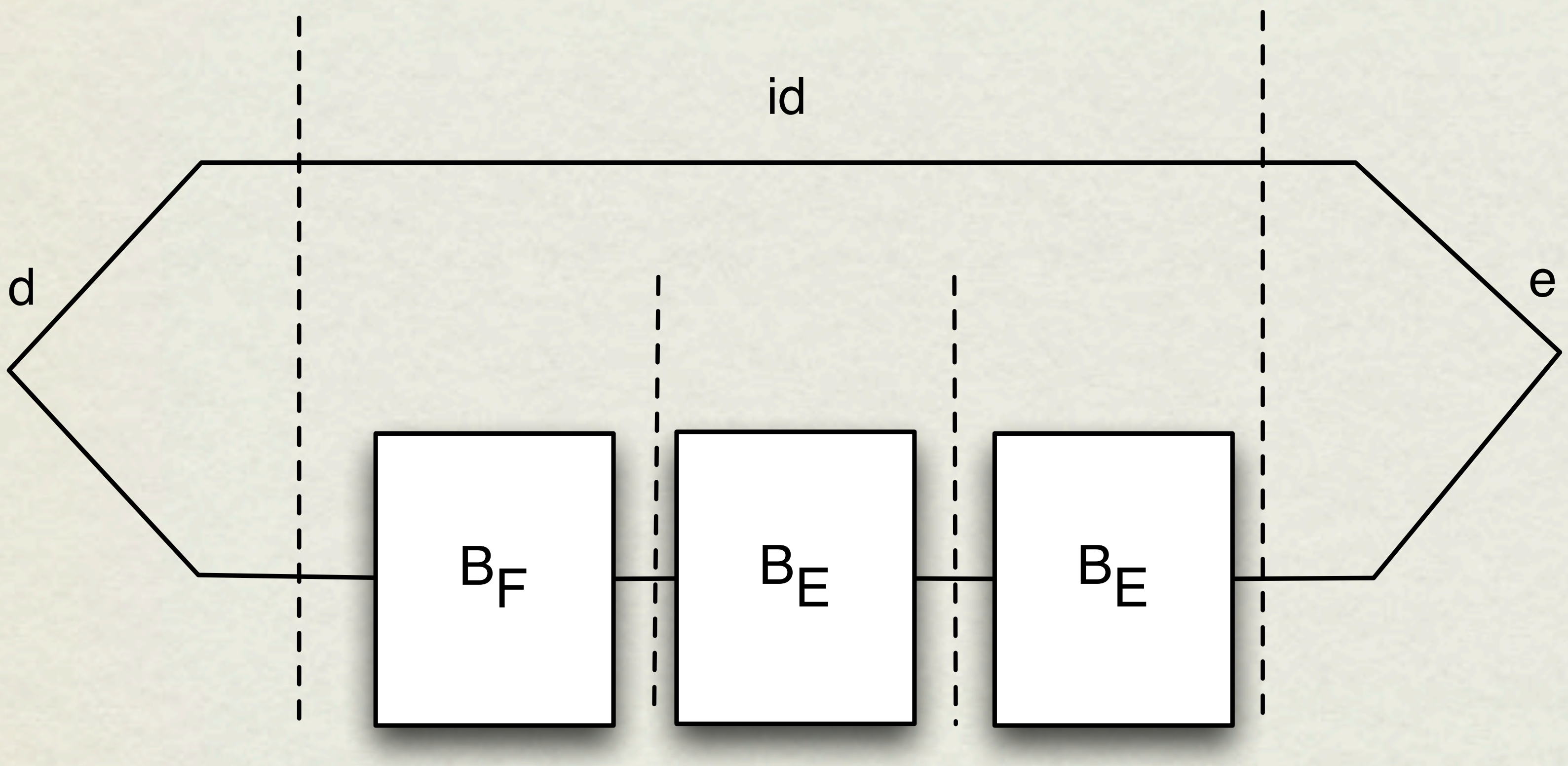
$B_E \otimes B_E$: (2,2)

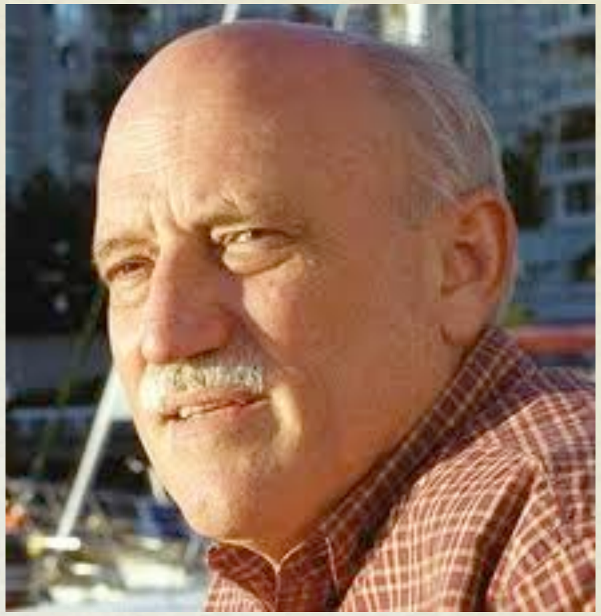
ALGEBRA OF PROCESSES



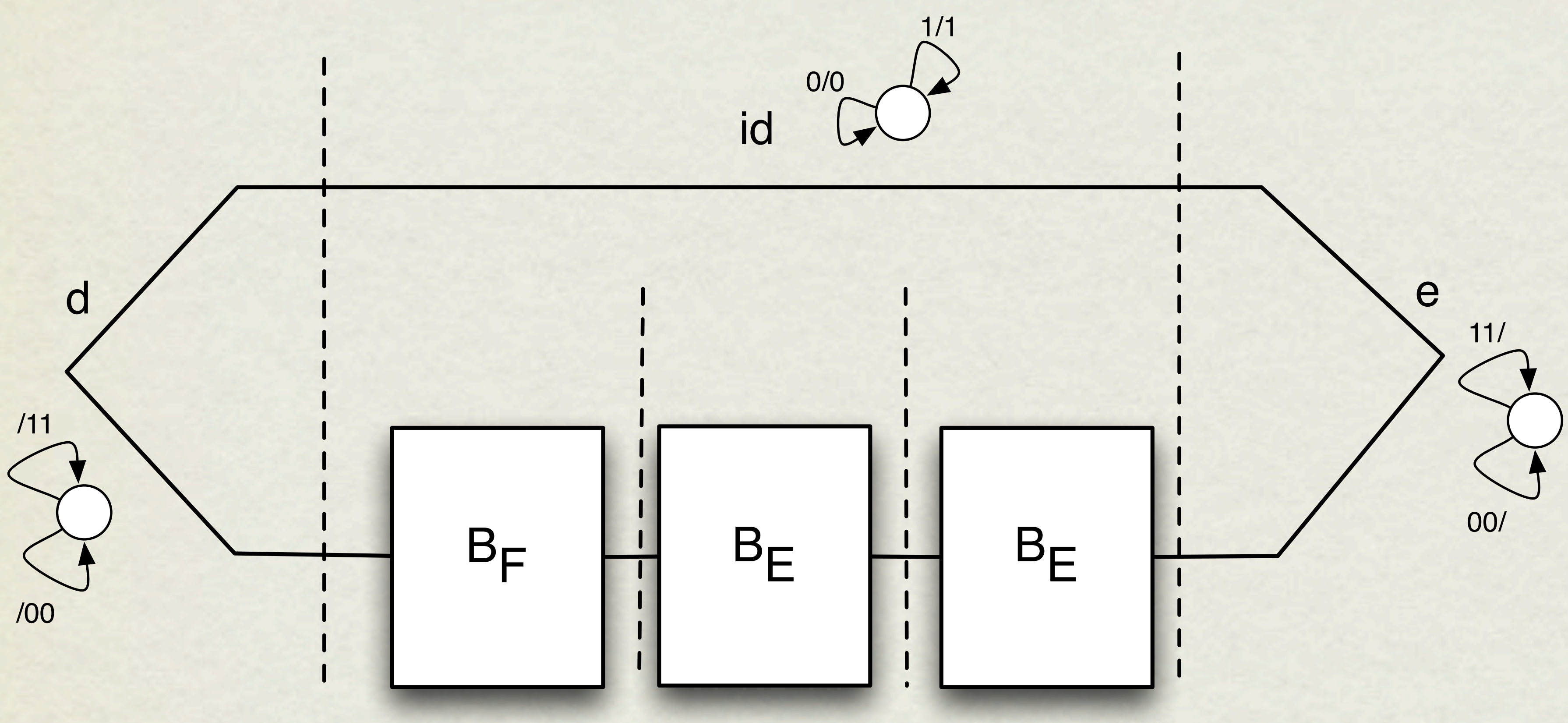


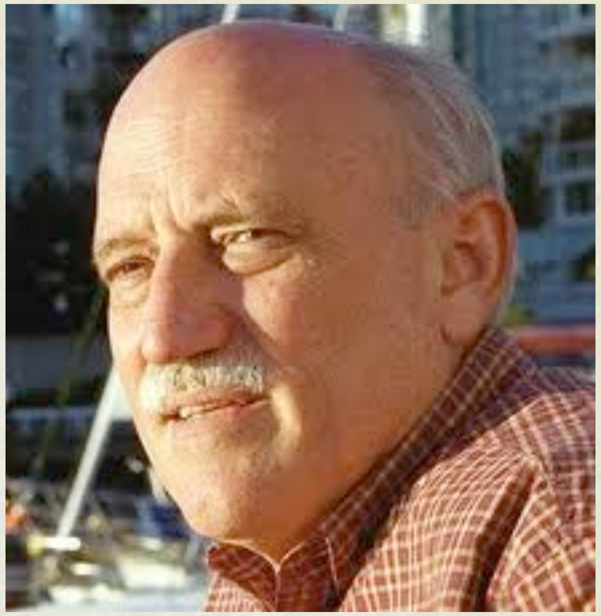
ALGEBRA OF PROCESSES



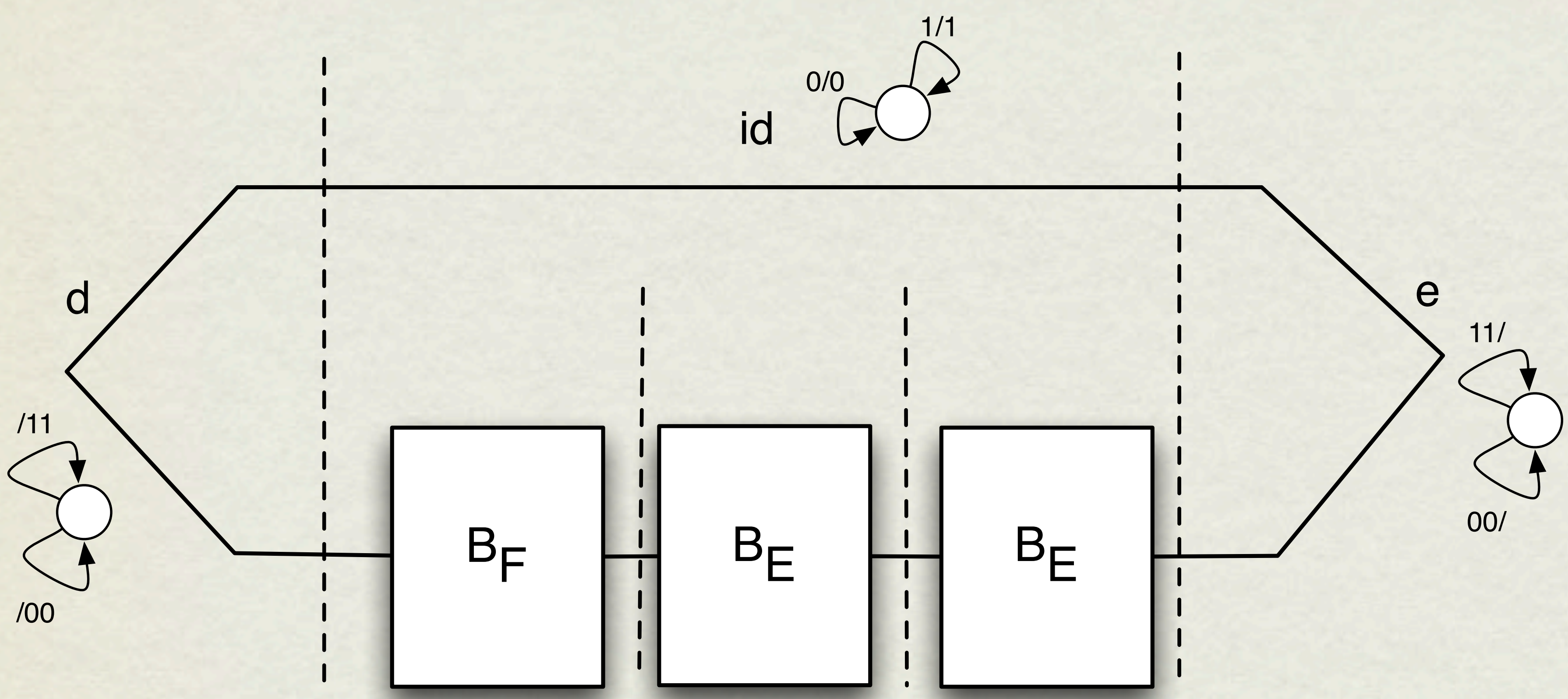


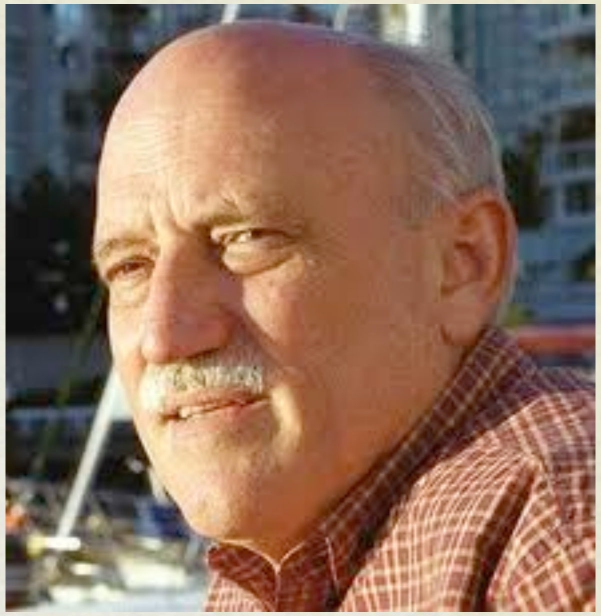
ALGEBRA OF PROCESSES





ALGEBRA OF PROCESSES





$d ; (id \otimes (B_F ; B_E ; B_E)) ; e : (0,0)$

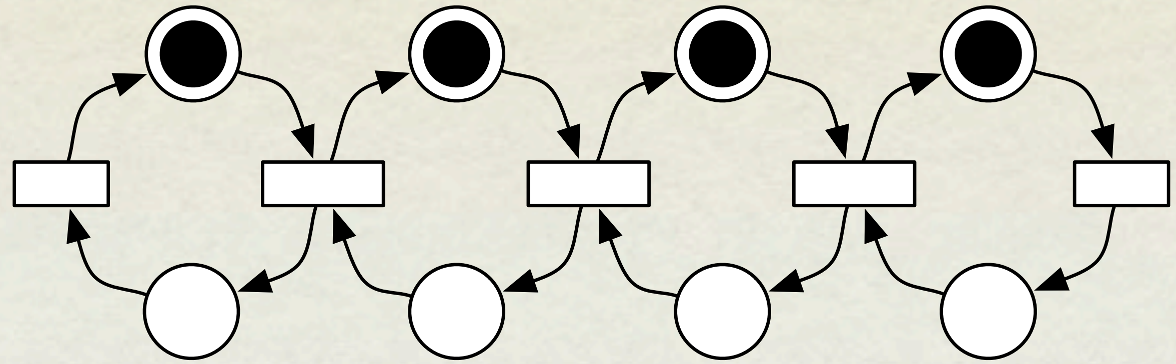
PROS AND CONS

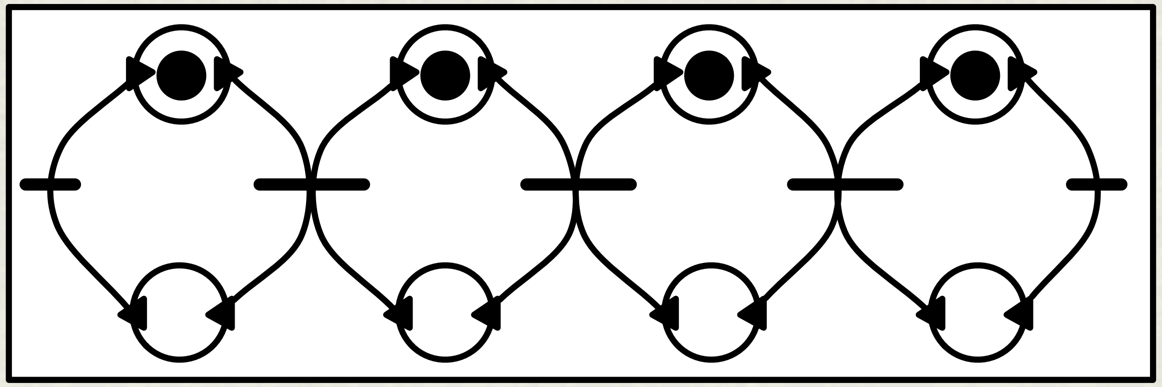
- Pros
  - Algebra with formal semantics
  - Compositional, reasonable equivalences are congruences
  - Syntax has close correspondence with geometry of systems
- Cons
  - Automata hide concurrency

ROADMAP

- Automata as model of concurrency - Span(Graph)
- **Nets with boundaries**
- Application to model checking
- Work in progress and future work

NETS AS STRING DIAGRAMS

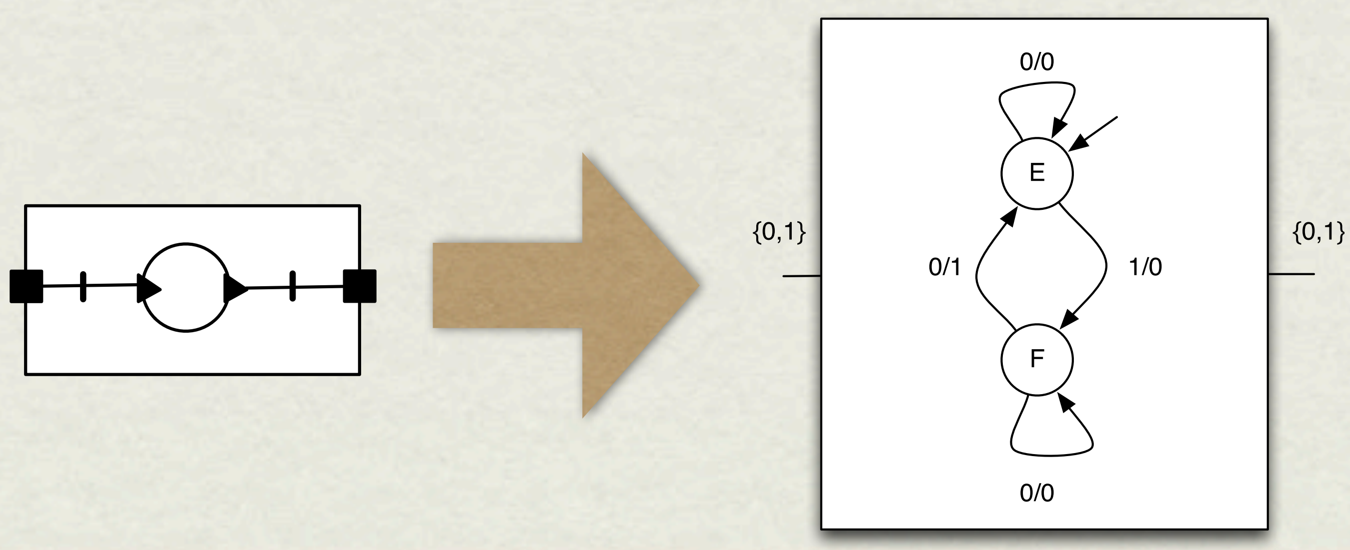




- places drawn with in-port and out-port
- transitions are undirected and simply connect a set of ports

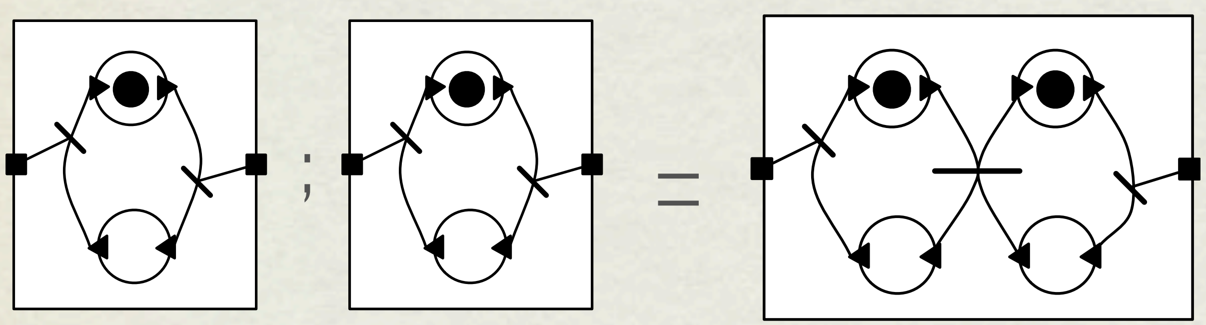
(1 BOUNDED) NETS WITH BOUNDARIES

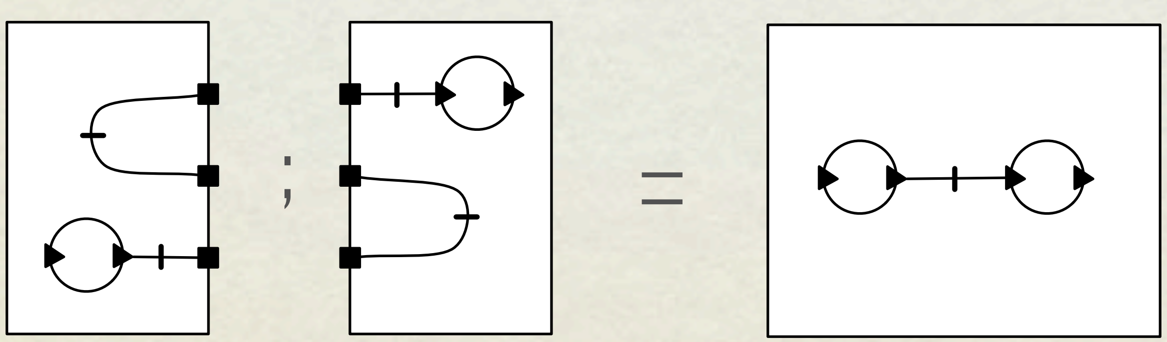
- add boundary ports
- transitions can connect also to boundary ports
- step semantics

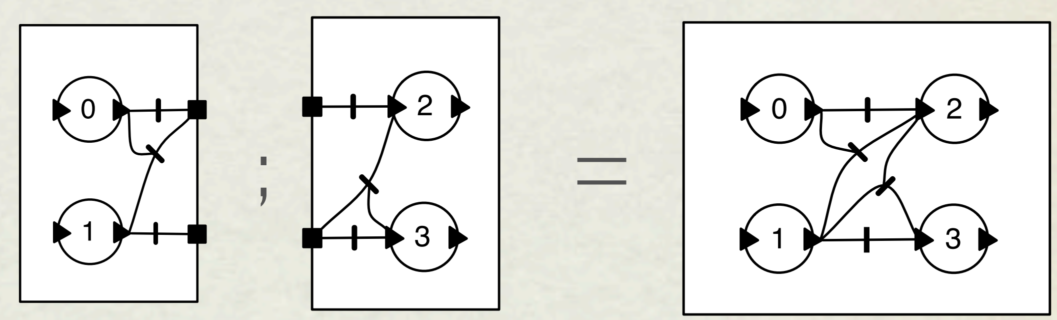


COMPOSING NETS

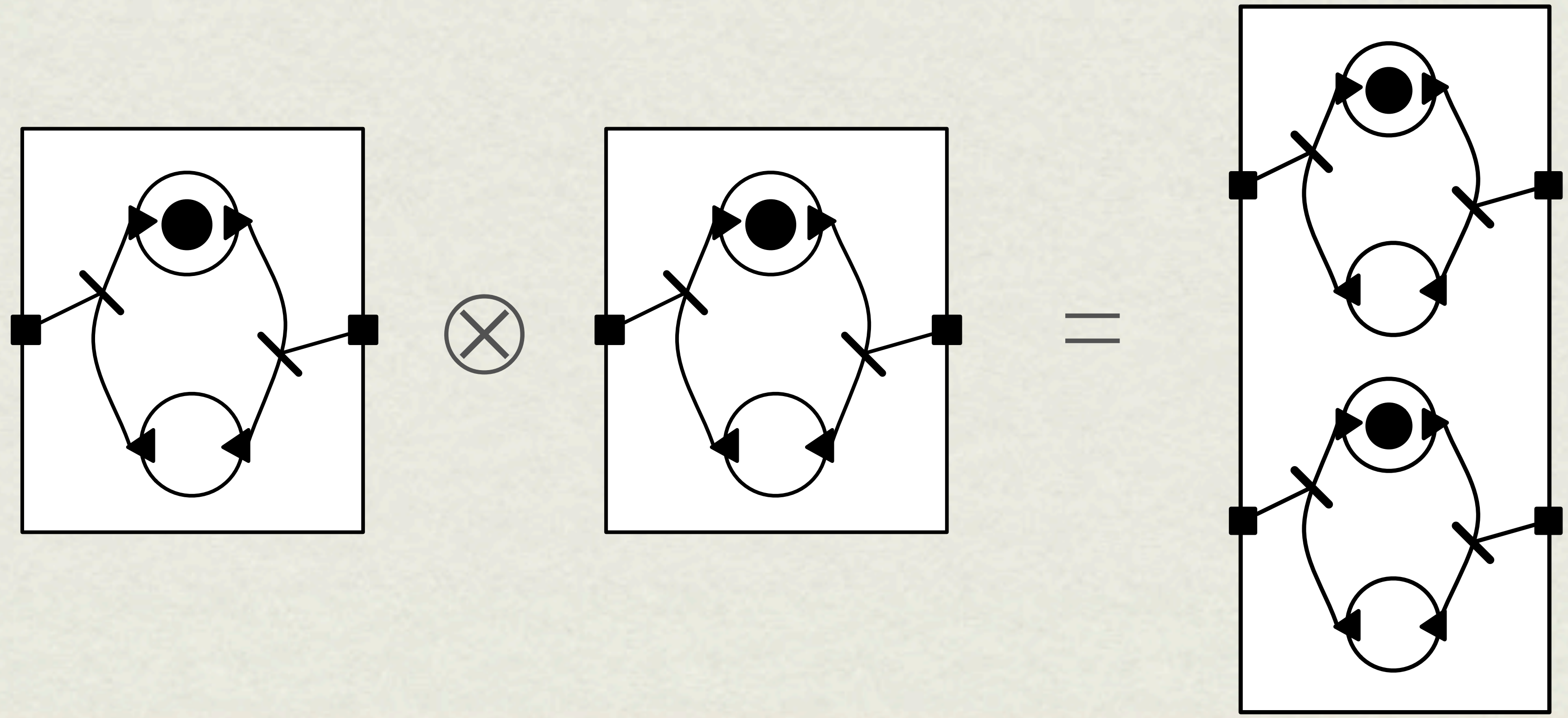
- Nets are composed in a “geometrically obvious” way
- Two or more transitions connected to a boundary port is a simple way of including nondeterminism in components





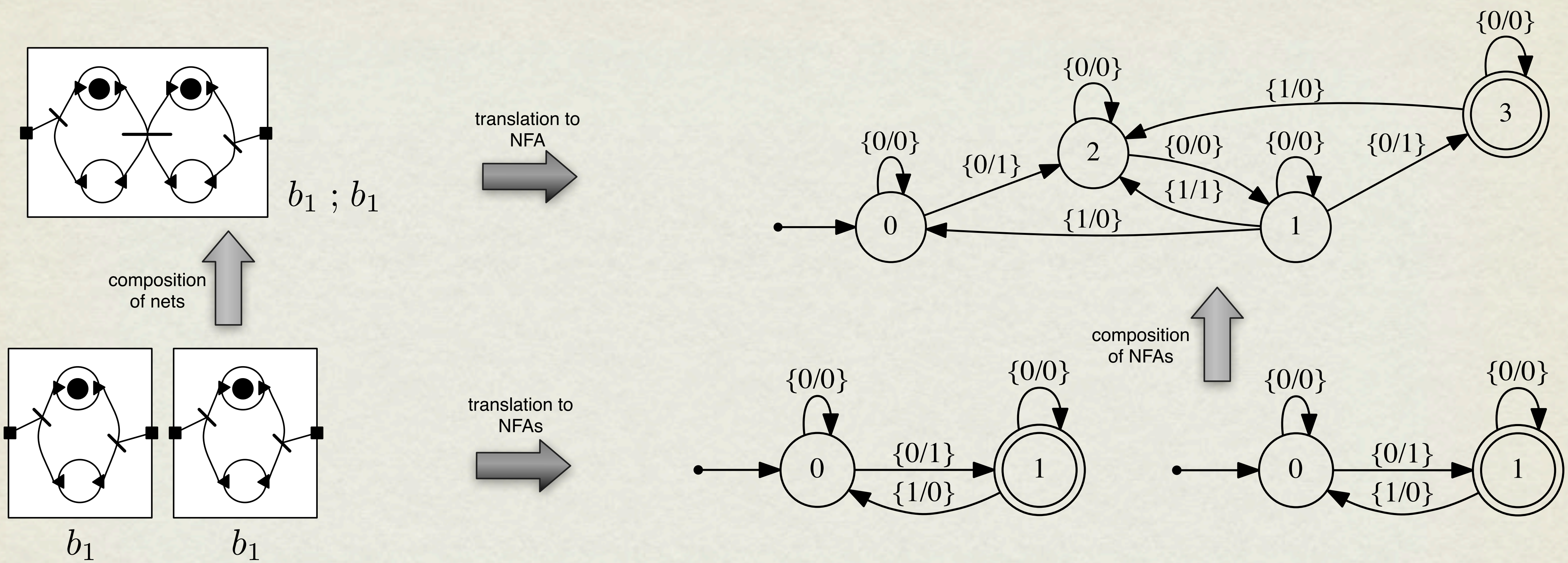


TENSOR PRODUCT



COMPOSITIONALITY

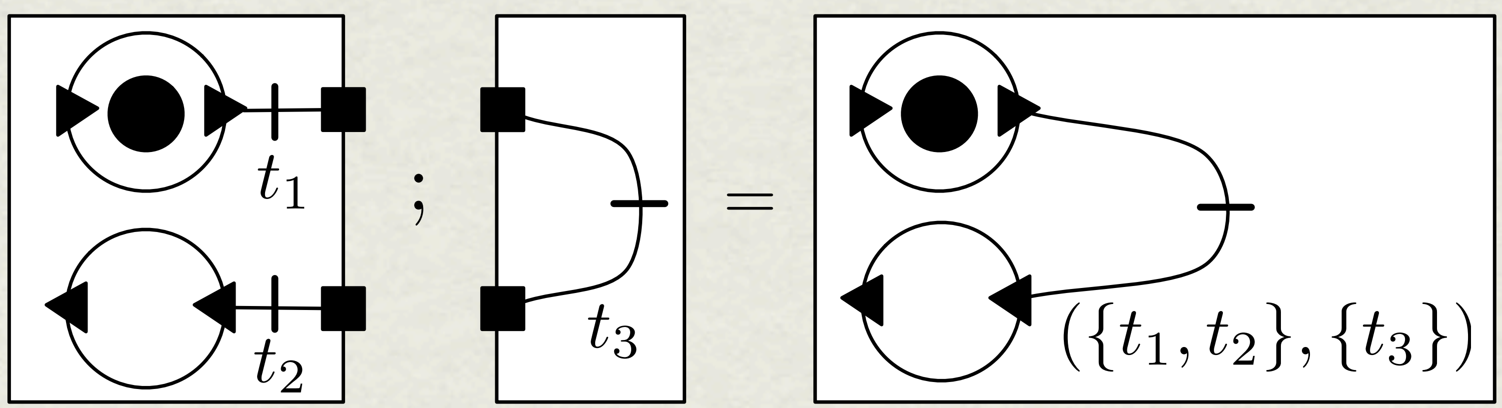
The following diagram always commutes



Moreover, all "reasonable equivalences" are congruences

WHY STEP SEMANTICS?

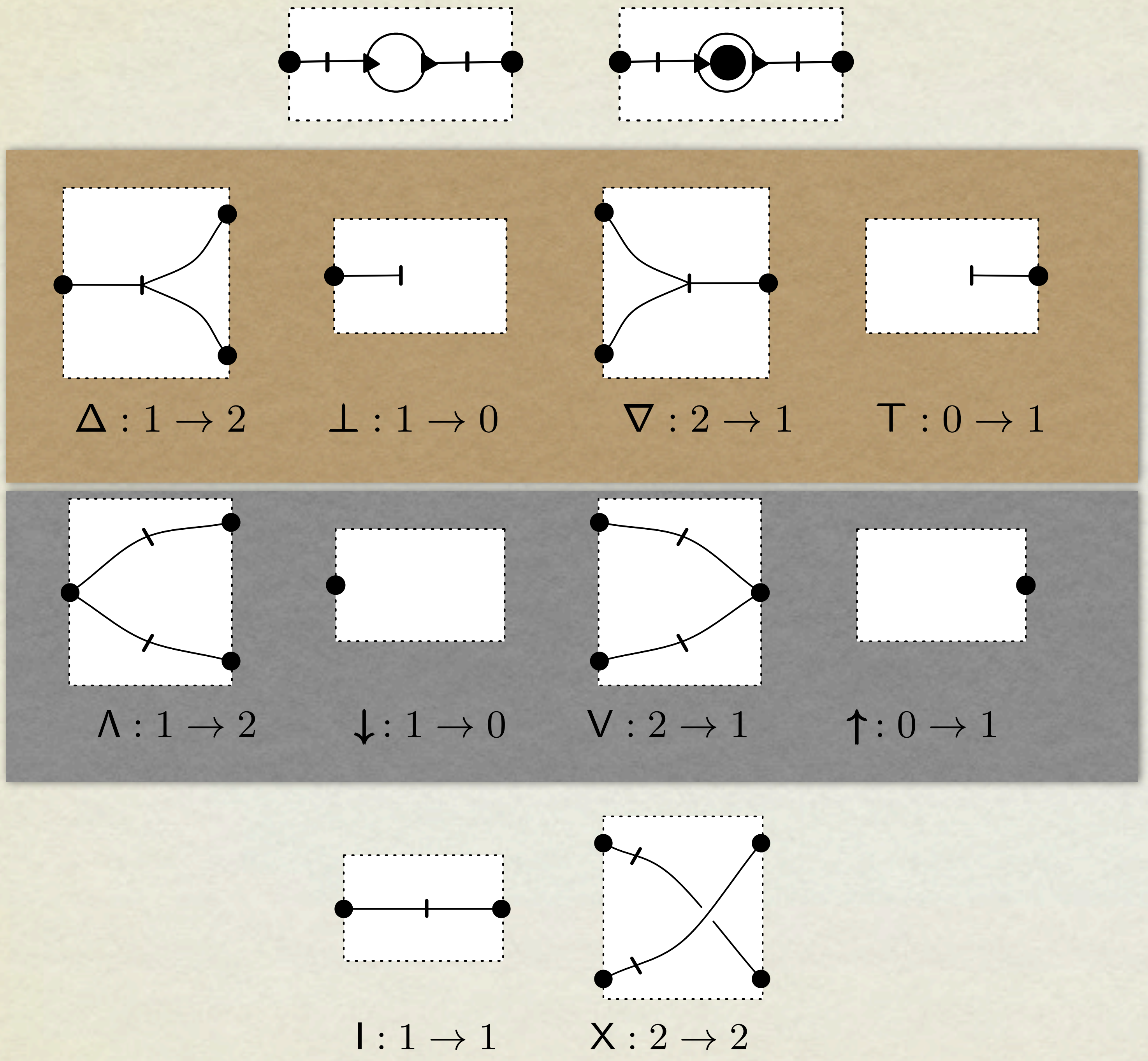
- Interleaving would not be compositional!



NETS WITH BOUNDARIES

- Algebra with formal semantics
- Compositional, reasonable equivalences are congruences
- Syntax has close correspondence with geometry of systems
- Evident concurrency

GENERATORS



The resulting algebraic theory can be studied using category theoretical machinery (PROPS) - some initial results reported at CALCO `13

WHAT ABOUT P/T NETS?

- Very similar algebra available for infinite state nets
  - in particular, for P/T nets we have the same generators
- Both algebras can be understood as certain process calculi
  - passing from bounded to unbounded nets is particularly easy from the point of view of process algebra, essentially one adds one new SOS rule:

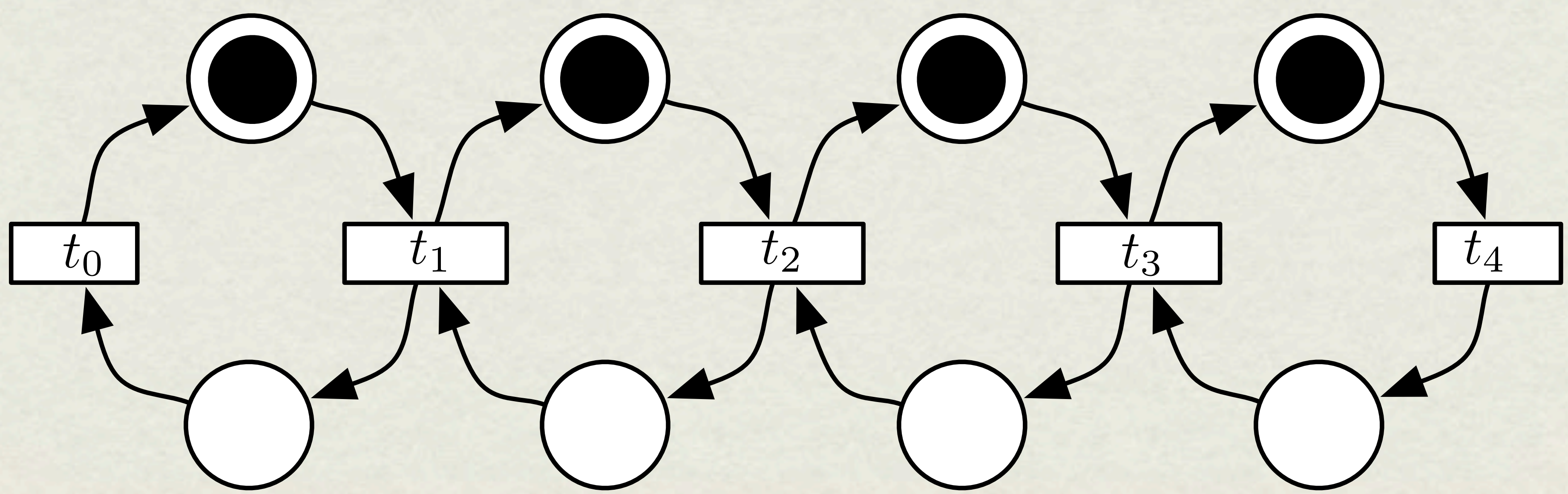
$$\frac{P \xrightarrow[\beta_1]{\alpha_1} R \qquad R \xrightarrow[\beta_2]{\alpha_2} Q}{P \xrightarrow[\beta_1+\beta_2]{\alpha_1+\alpha_2} Q} \ (\textsc{Weak*})$$

ROADMAP

- Automata as model of concurrency - Span(Graph)
- Nets with boundaries
- **Application to model checking**
- Work in progress and future work

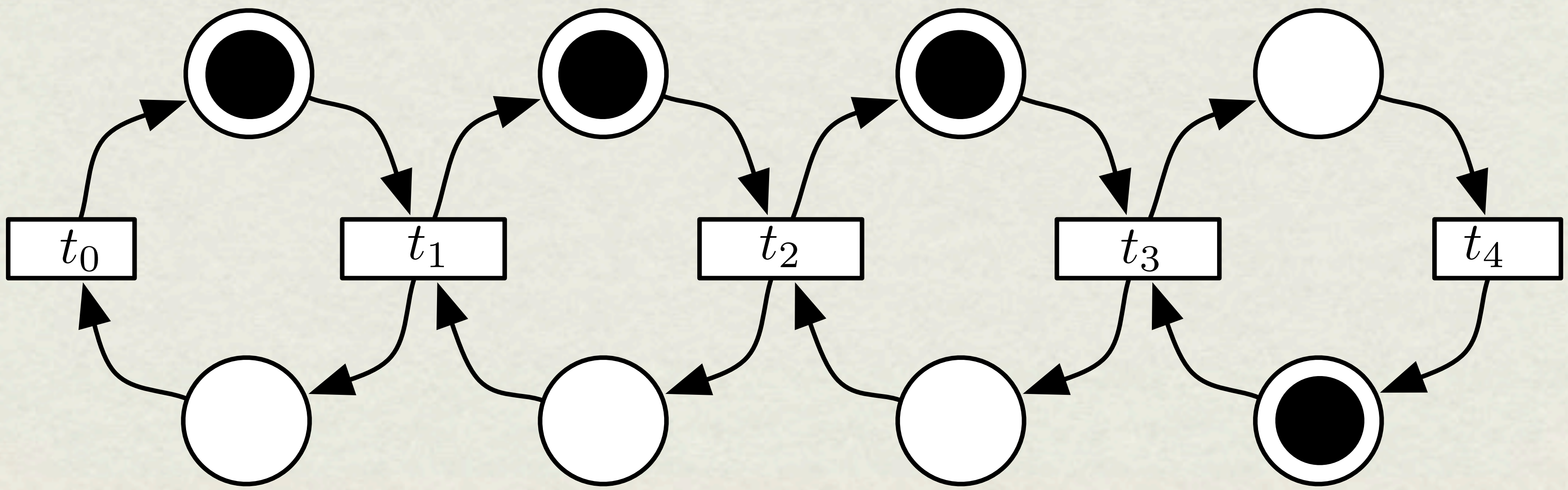
APPLICATION: REACHABILITY

- Reachability in 1-bounded nets is PSPACE-complete
- most "real" systems are quite modular - can we exploit this?



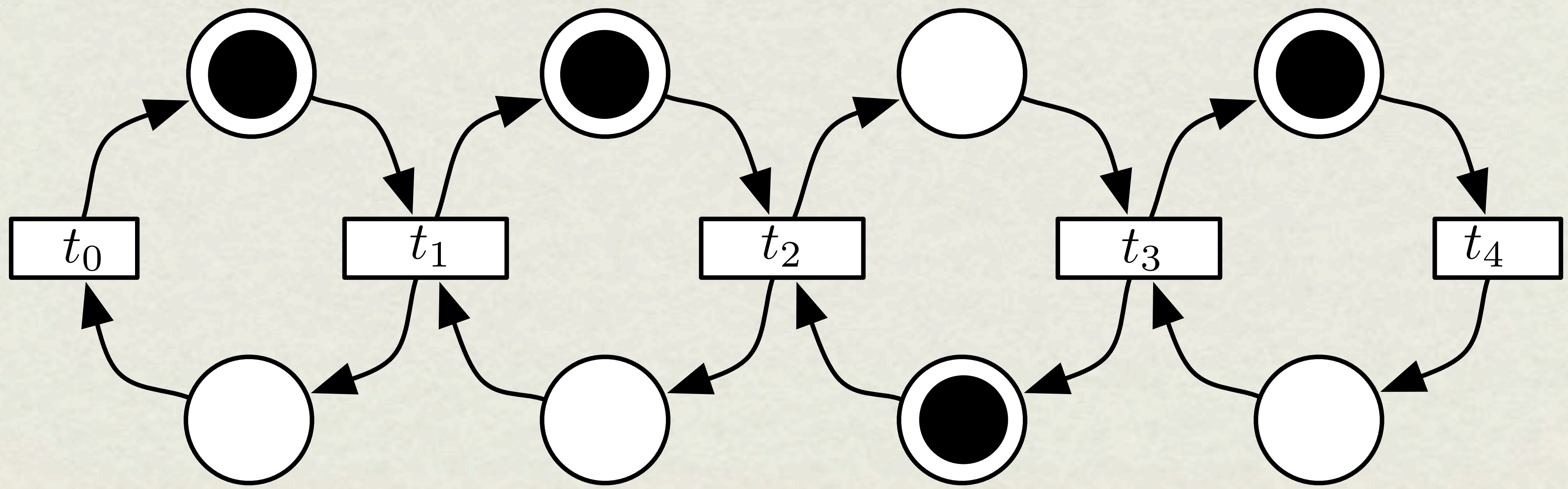
APPLICATION: REACHABILITY

- Reachability in 1-bounded nets is PSPACE-complete
- most “real” systems are quite modular - can we exploit this?



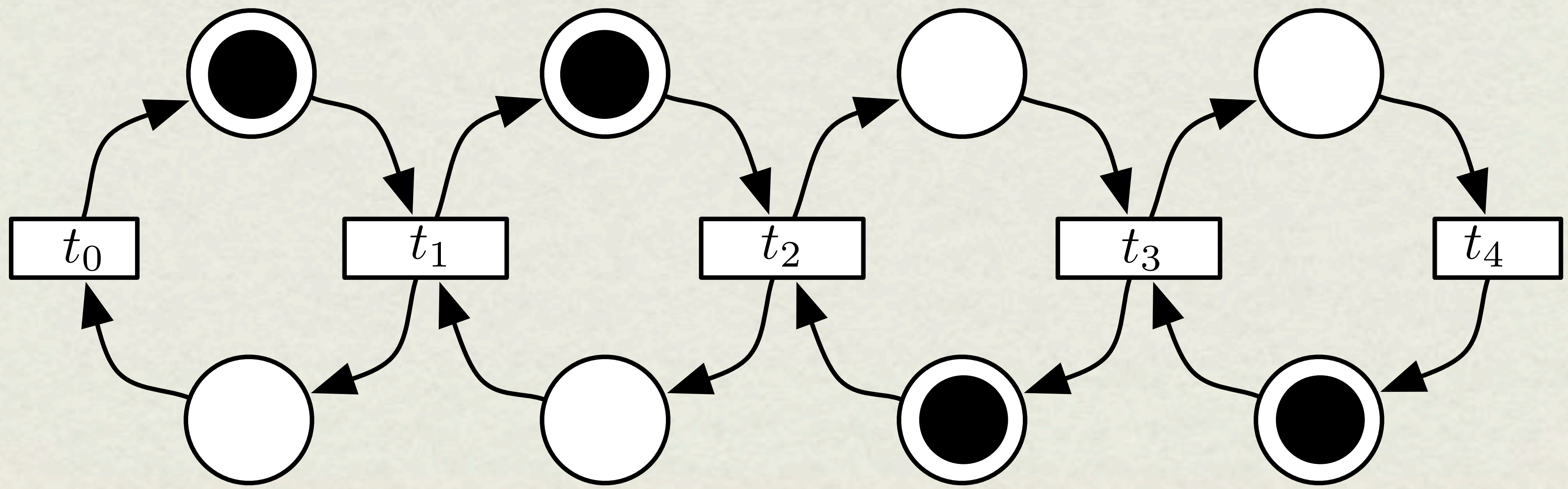
APPLICATION: REACHABILITY

- Reachability in 1-bounded nets is PSPACE-complete
- most "real" systems are quite modular - can we exploit this?



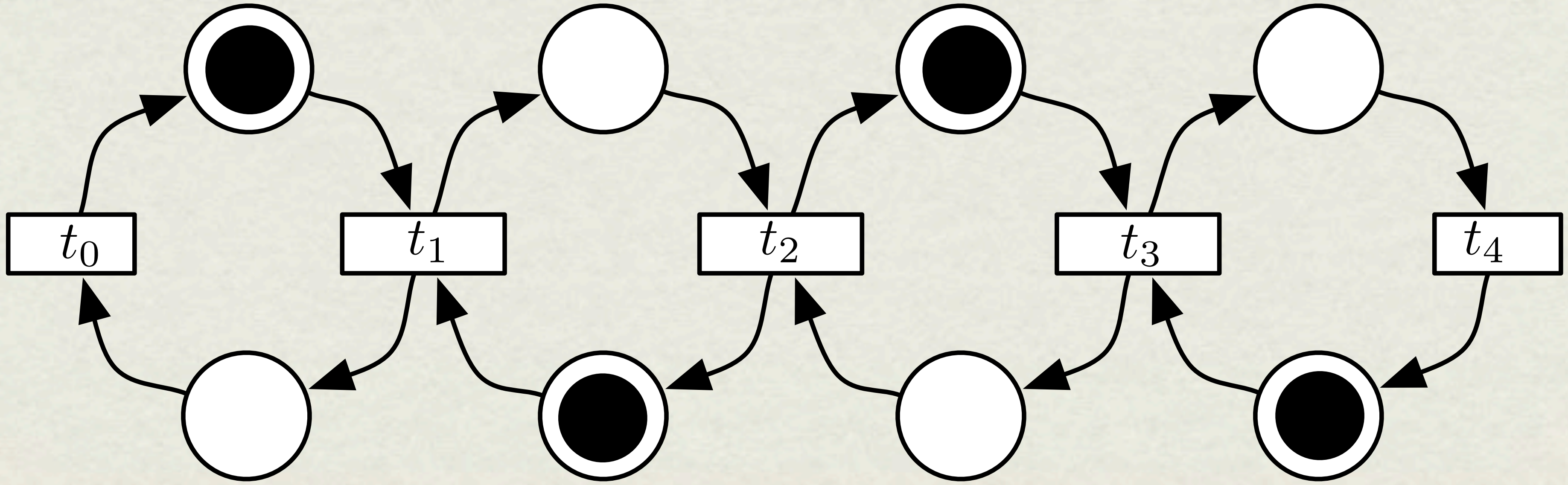
APPLICATION: REACHABILITY

- Reachability in 1-bounded nets is PSPACE-complete
- most "real" systems are quite modular - can we exploit this?



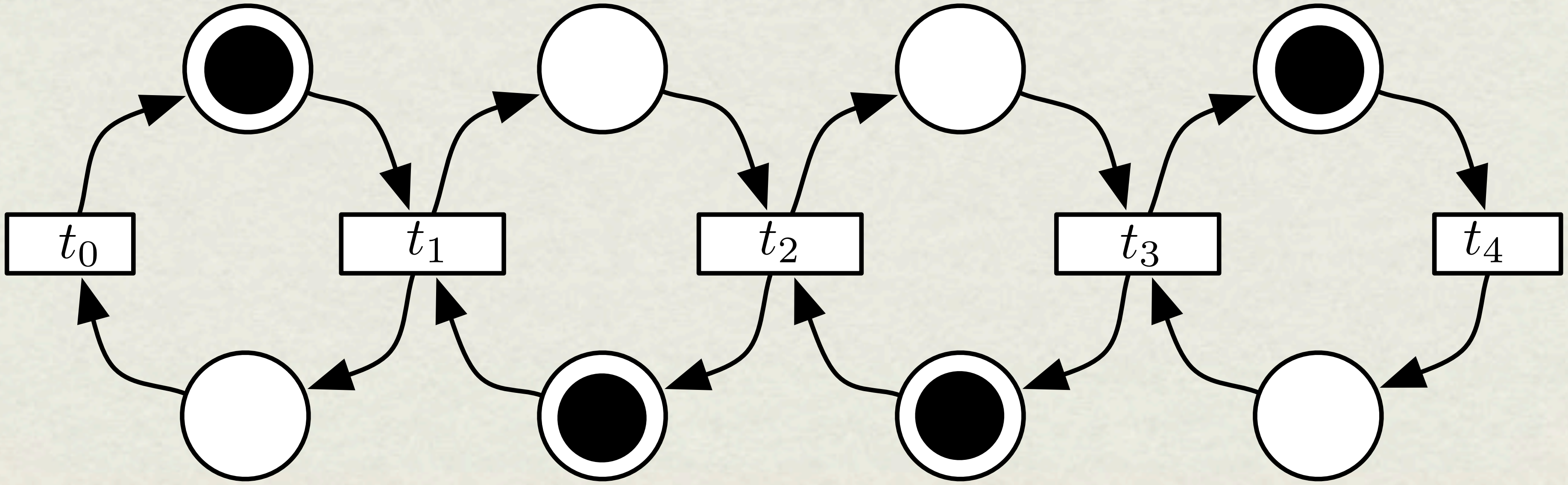
APPLICATION: REACHABILITY

- Reachability in 1-bounded nets is PSPACE-complete
- most “real” systems are quite modular - can we exploit this?



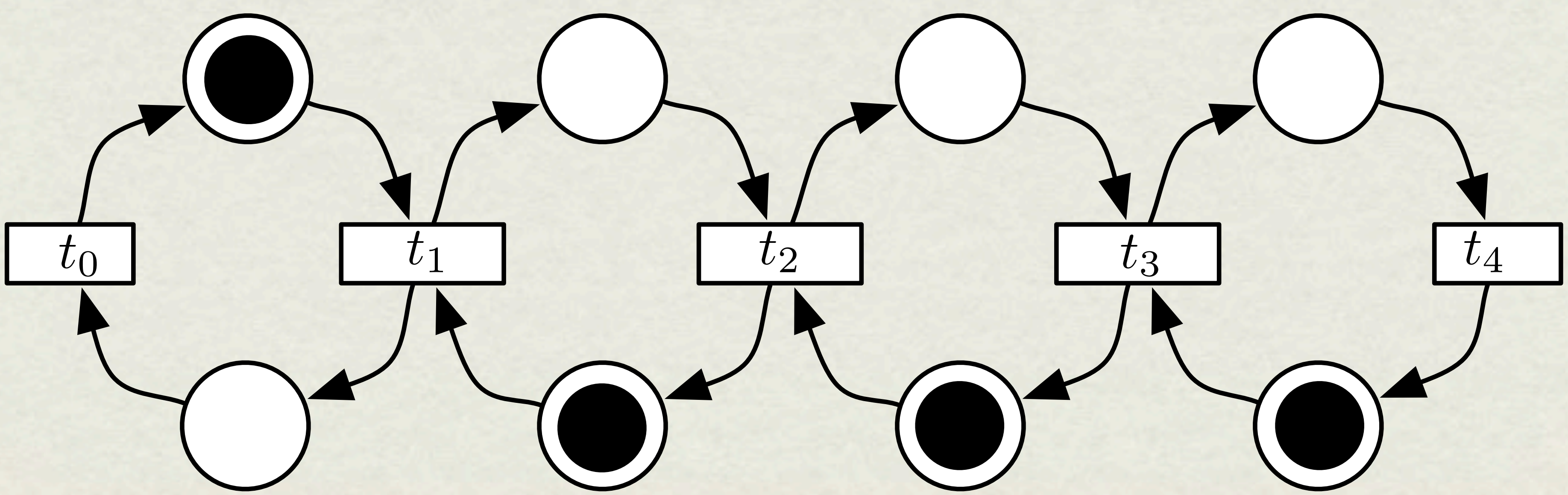
APPLICATION: REACHABILITY

- Reachability in 1-bounded nets is PSPACE-complete
- most “real” systems are quite modular - can we exploit this?



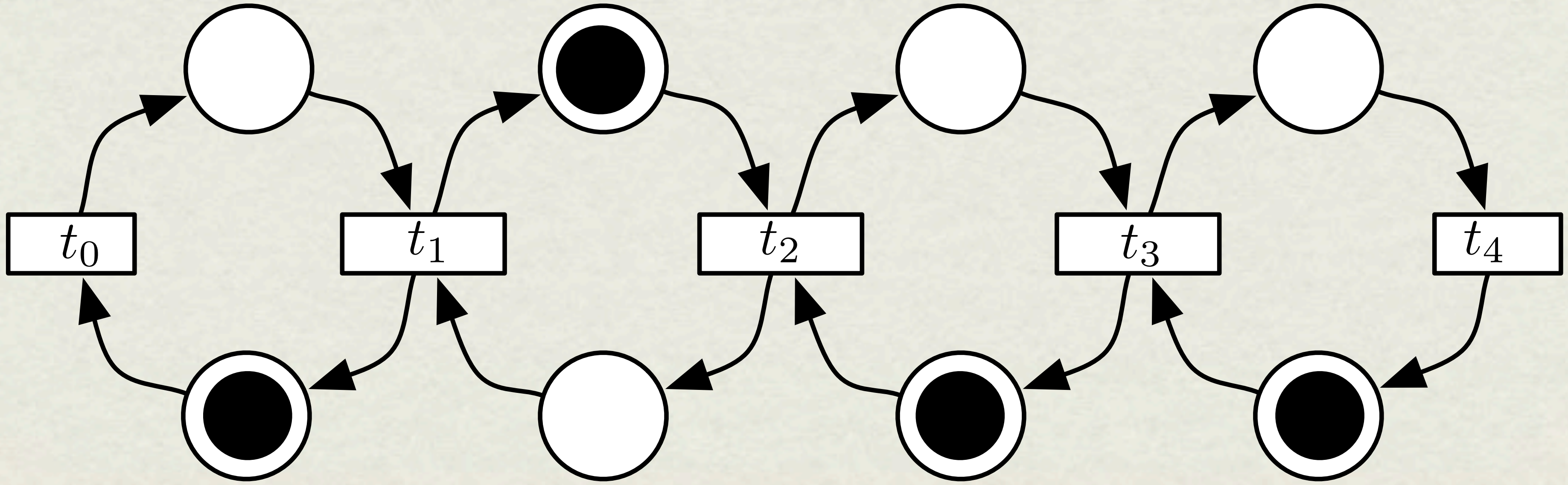
APPLICATION: REACHABILITY

- Reachability in 1-bounded nets is PSPACE-complete
- most “real” systems are quite modular - can we exploit this?



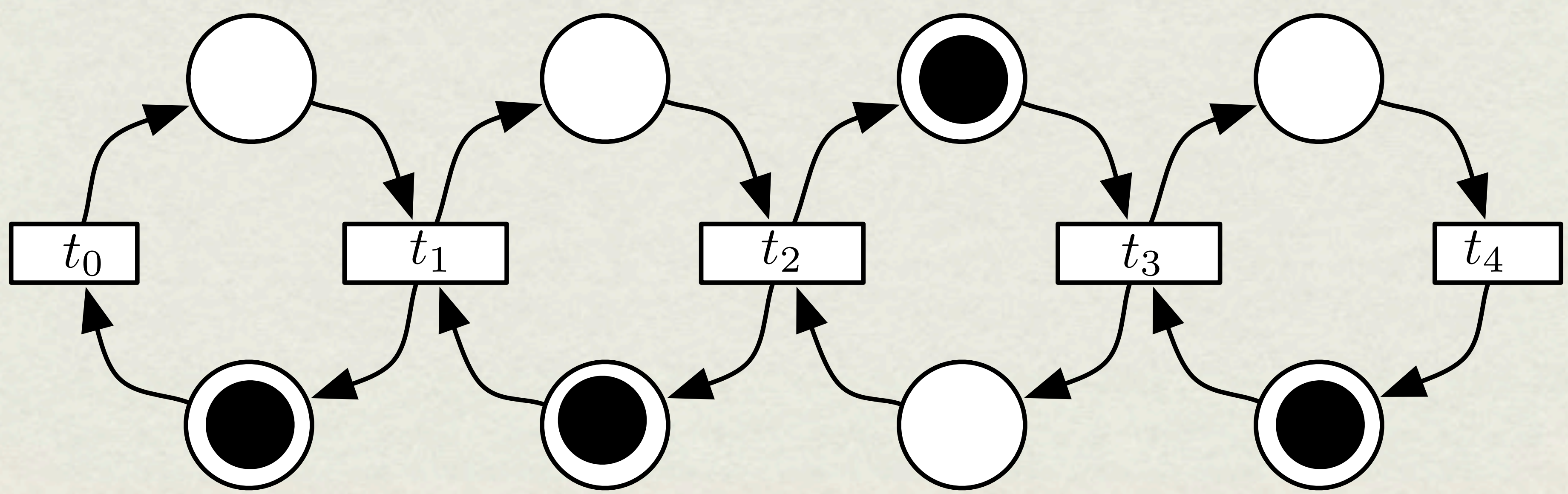
APPLICATION: REACHABILITY

- Reachability in 1-bounded nets is PSPACE-complete
- most "real" systems are quite modular - can we exploit this?



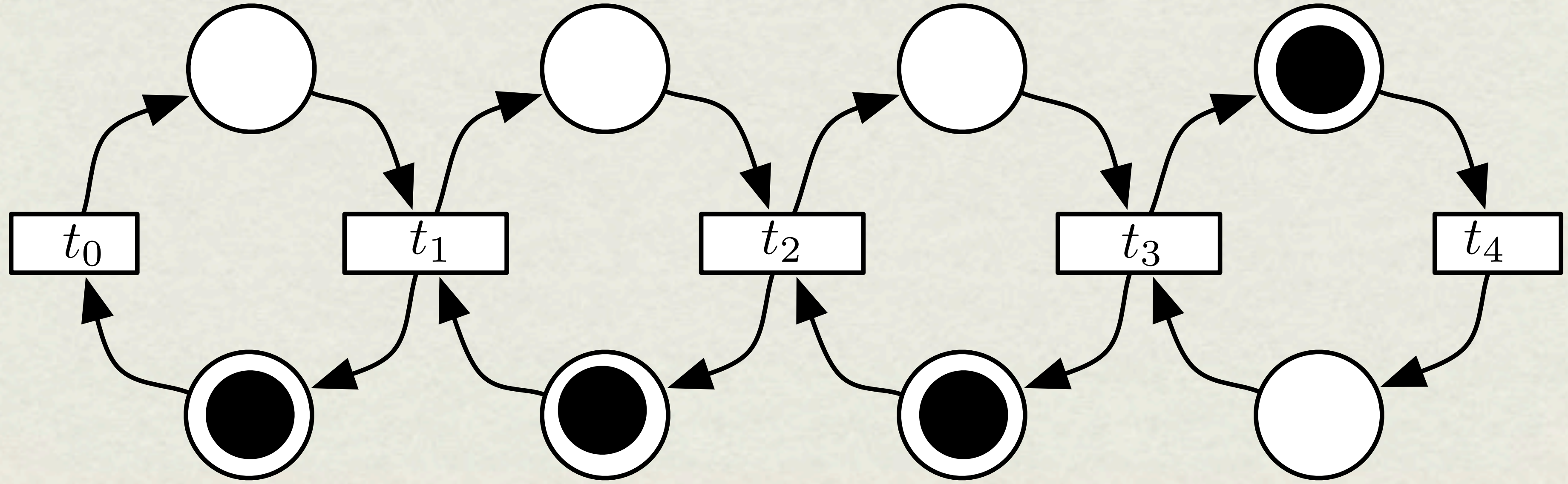
APPLICATION: REACHABILITY

- Reachability in 1-bounded nets is PSPACE-complete
- most "real" systems are quite modular - can we exploit this?



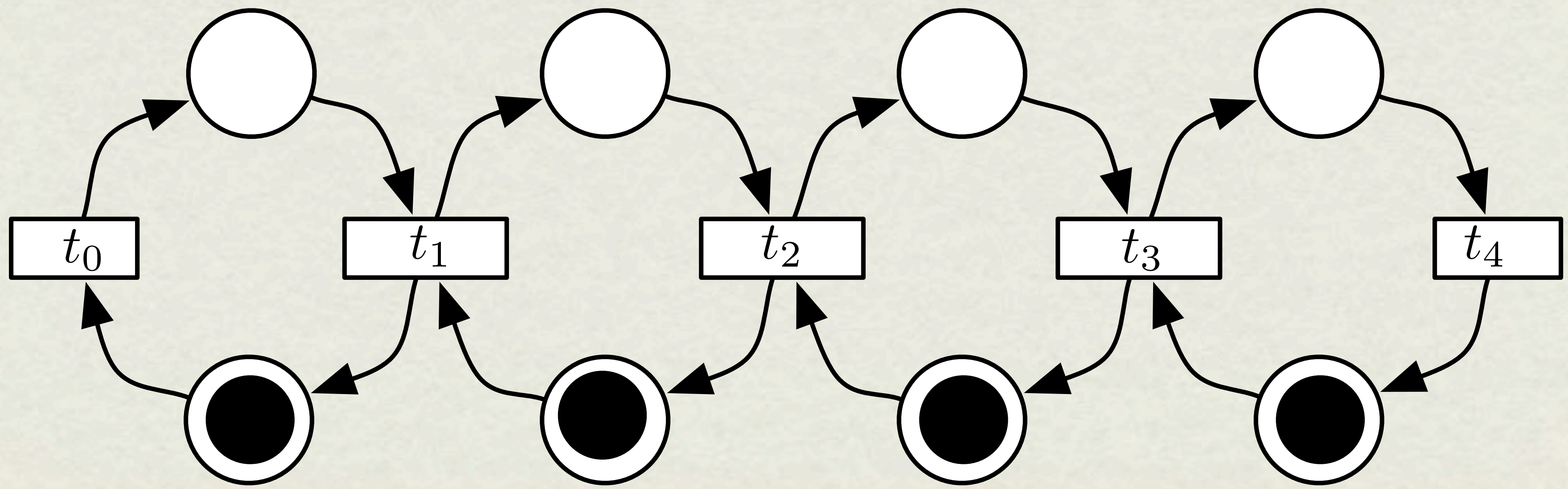
APPLICATION: REACHABILITY

- Reachability in 1-bounded nets is PSPACE-complete
- most “real” systems are quite modular - can we exploit this?

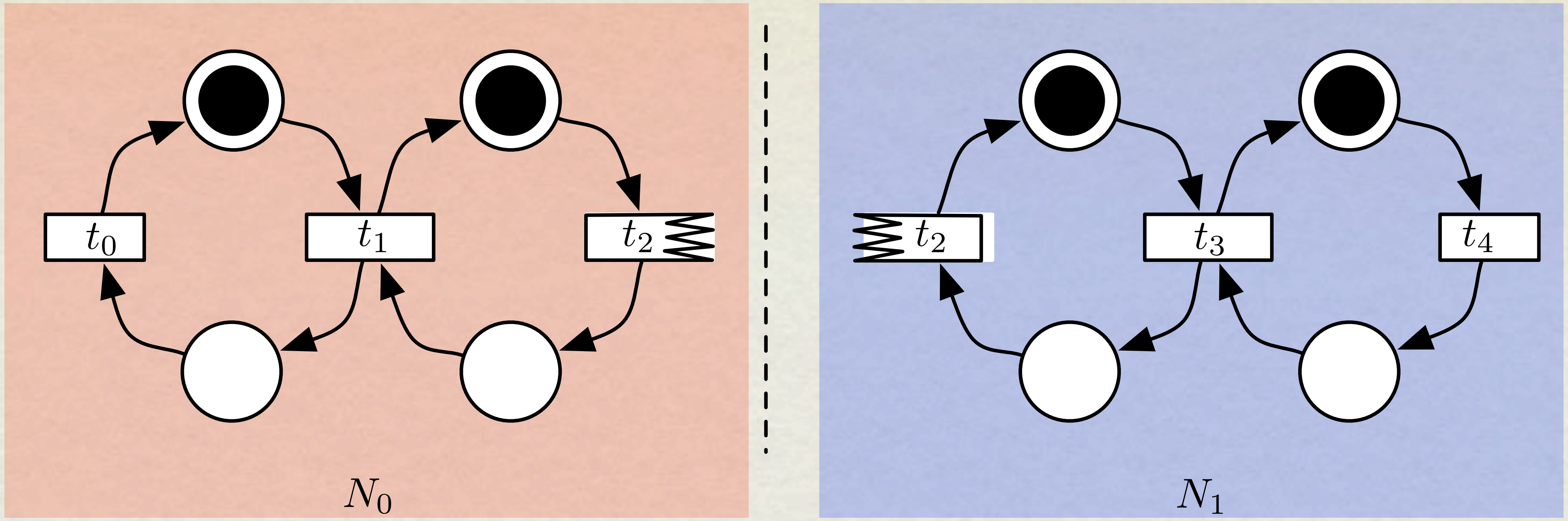


APPLICATION: REACHABILITY

- Reachability in 1-bounded nets is PSPACE-complete
- most “real” systems are quite modular - can we exploit this?



DECOMPOSING



"synchronisation policies"

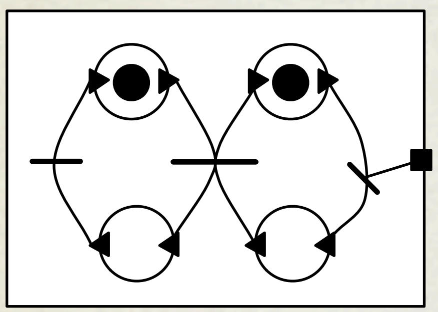
N_2 can reach desired local after firing t_2 twice, after which it can be fired an arbitrary additional number of times

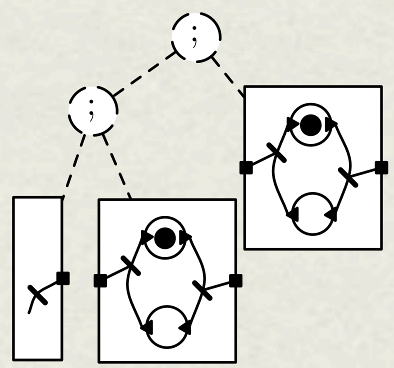
N_1 can reach desired local marking and fire t_2 an arbitrary number of times

INTERACTION IS WHAT MATTERS

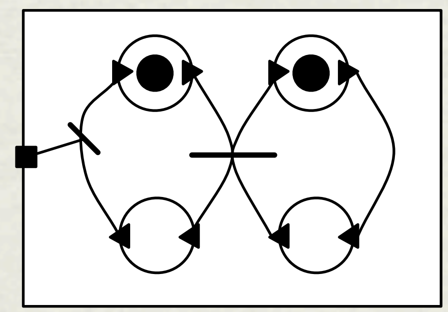
- in concurrency, what is important is the notion of **process**
    - ie. can throw away unnecessary local state and keep only the minimal amount of information necessary to express communication with environment

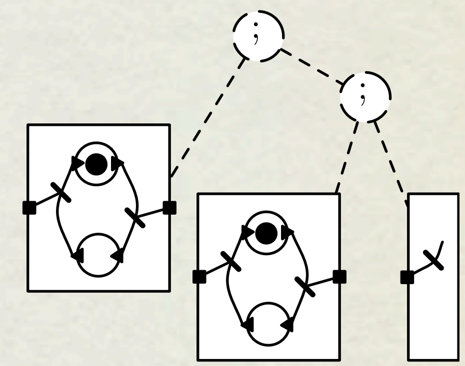
N_2 can reach desired local after firing t_2 twice, after which it can be fired an arbitrary additional number of times



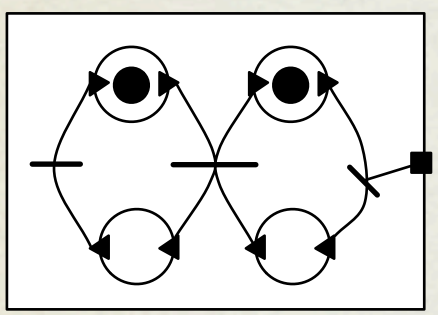


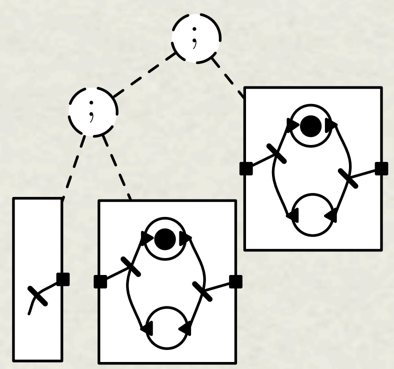
N_1 can reach desired local marking and fire t_2 an arbitrary number of times

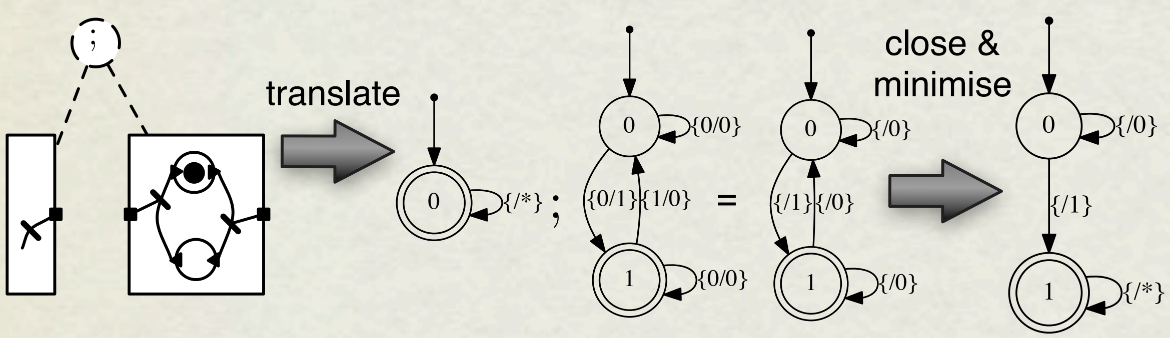




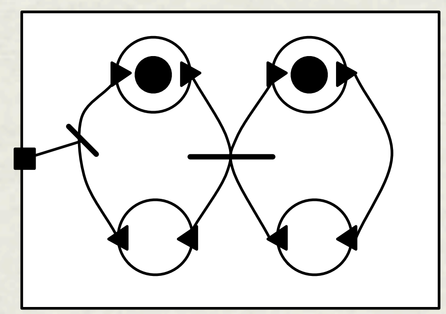
N_2 can reach desired local after firing t_2 twice, after which it can be fired an arbitrary additional number of times

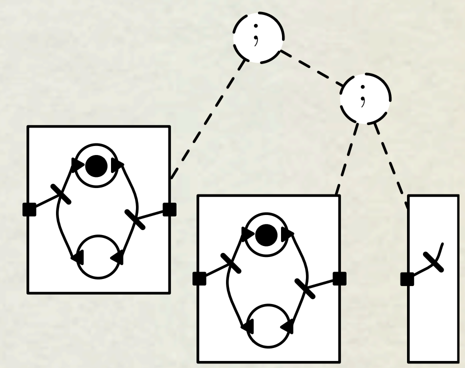




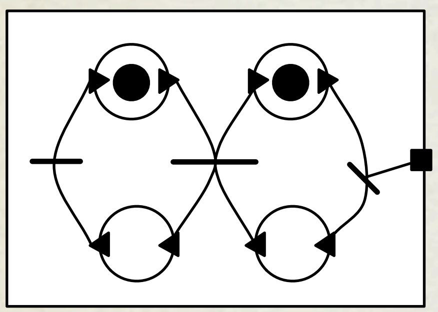


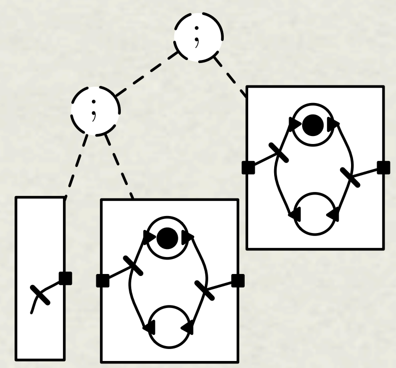
N_1 can reach desired local marking and fire t_2 an arbitrary number of times

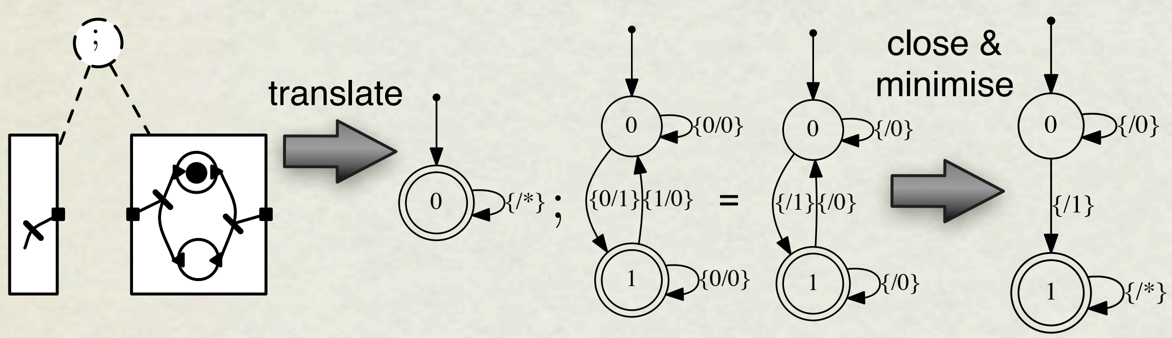


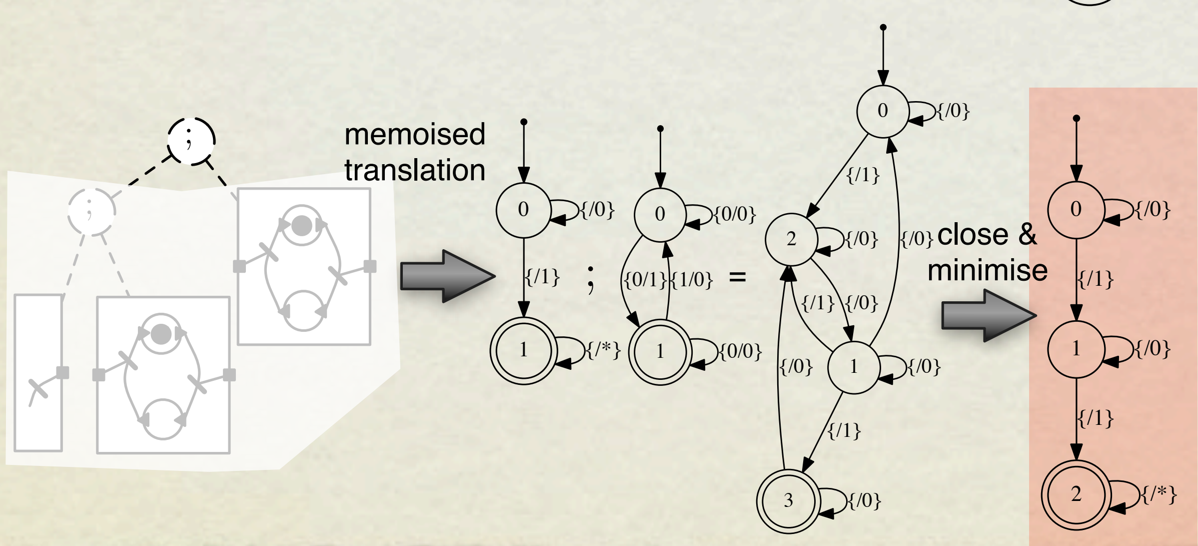


N$_2$ can reach desired local after firing t$_2$ twice, after which it can be fired an arbitrary additional number of times

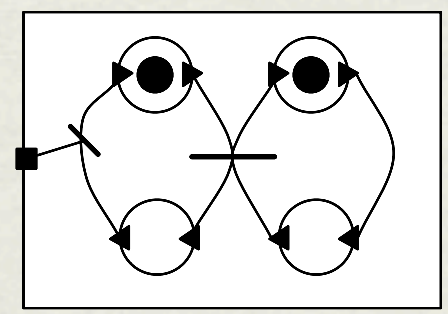


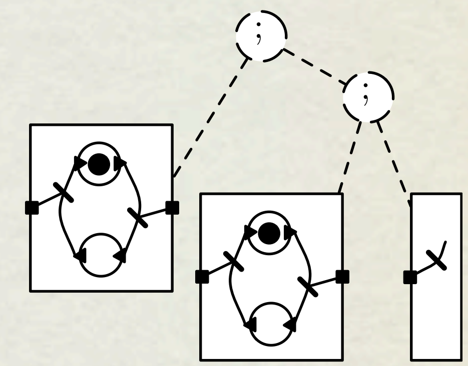




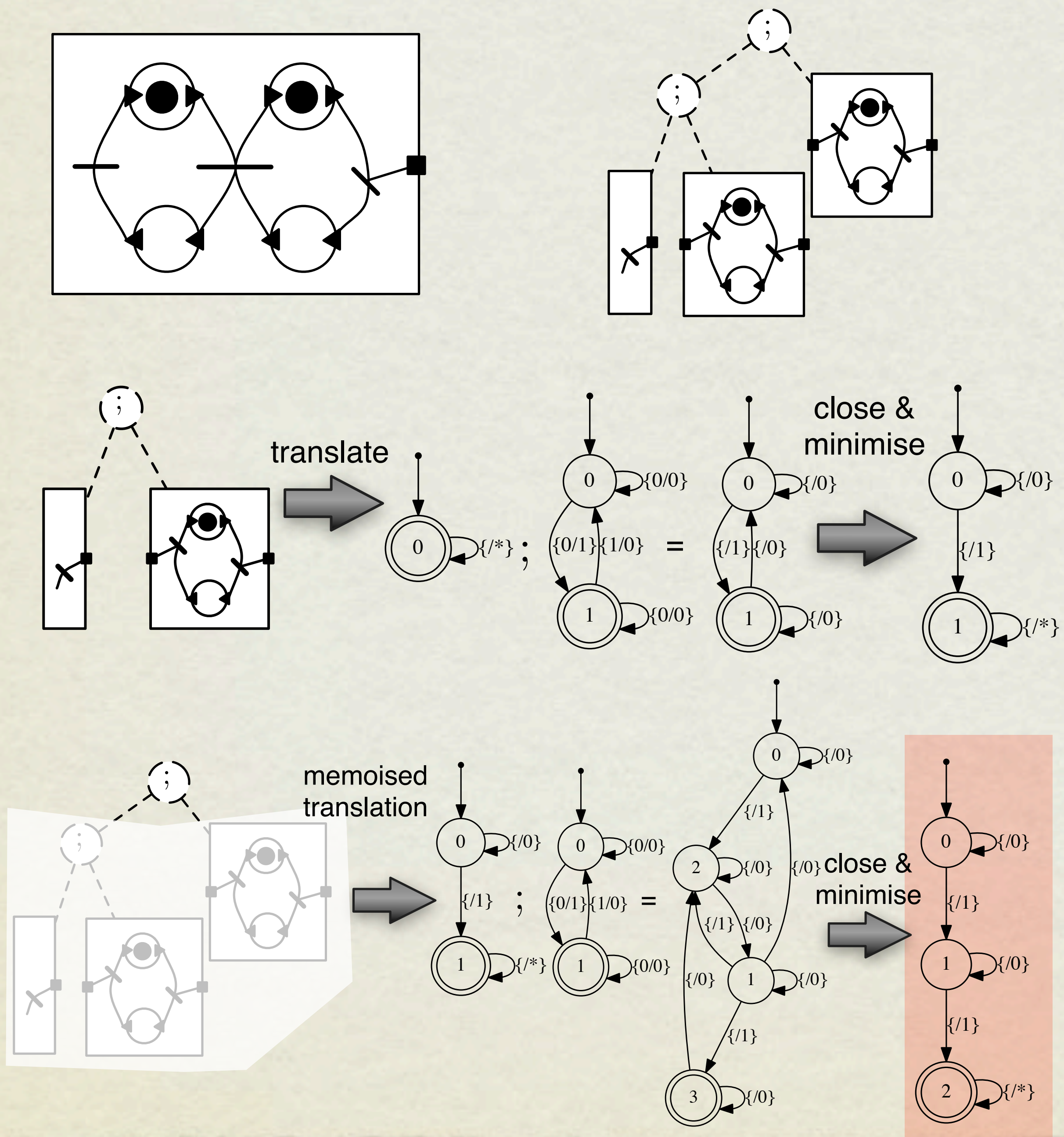


N$_1$ can reach desired local marking and fire t$_2$ an arbitrary number of times

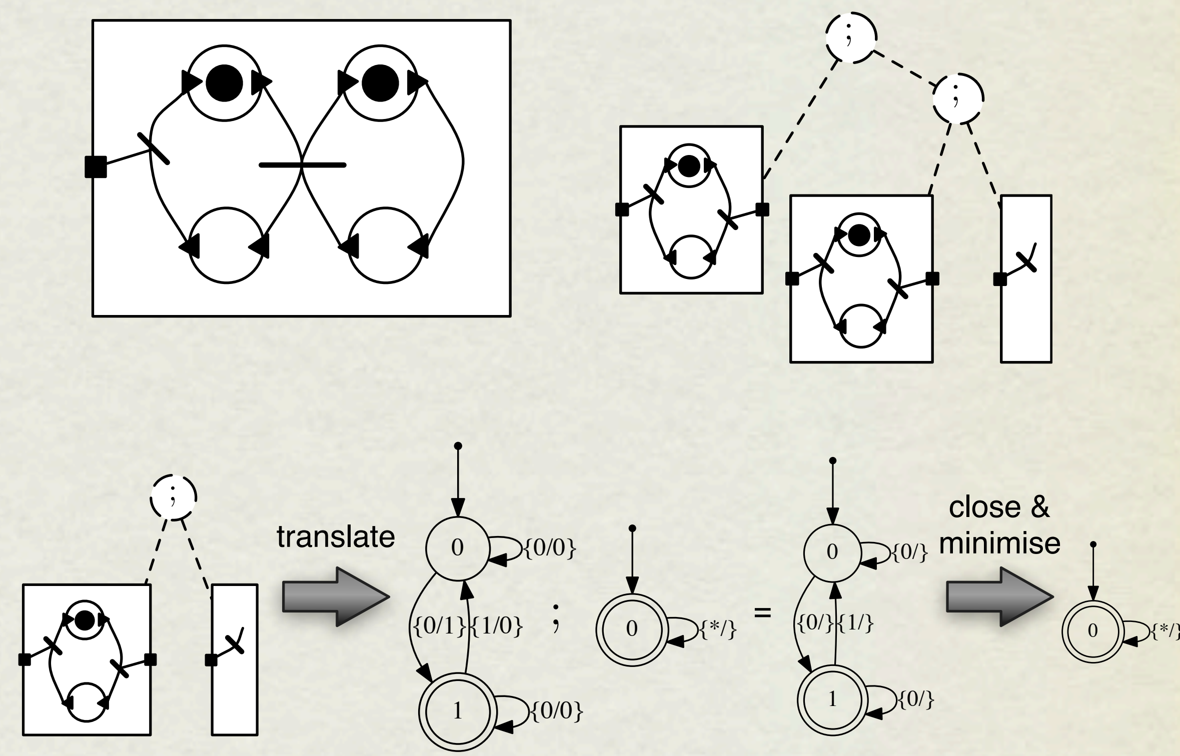




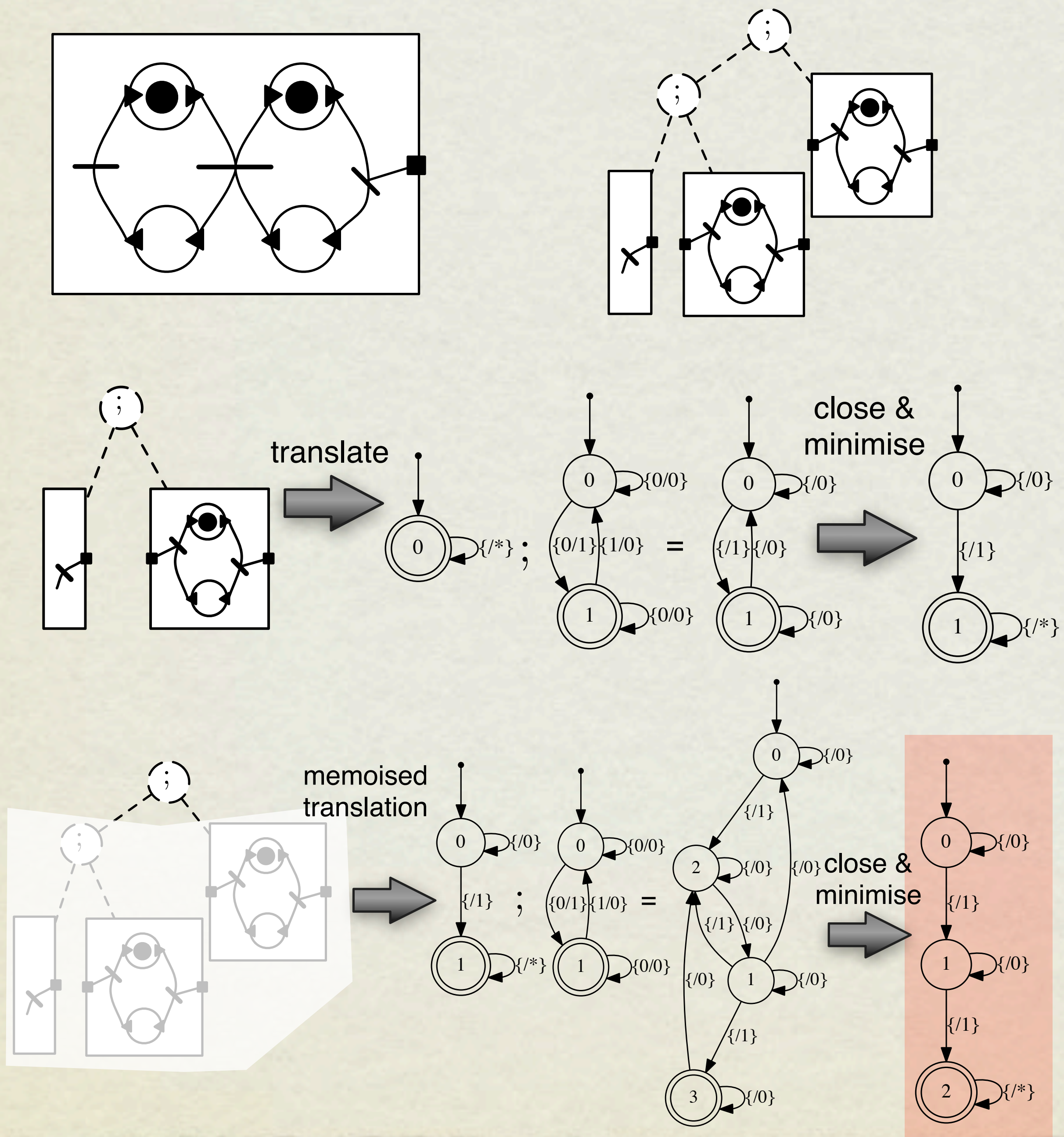
N_2 can reach desired local after firing t_2 twice, after which it can be fired an arbitrary additional number of times



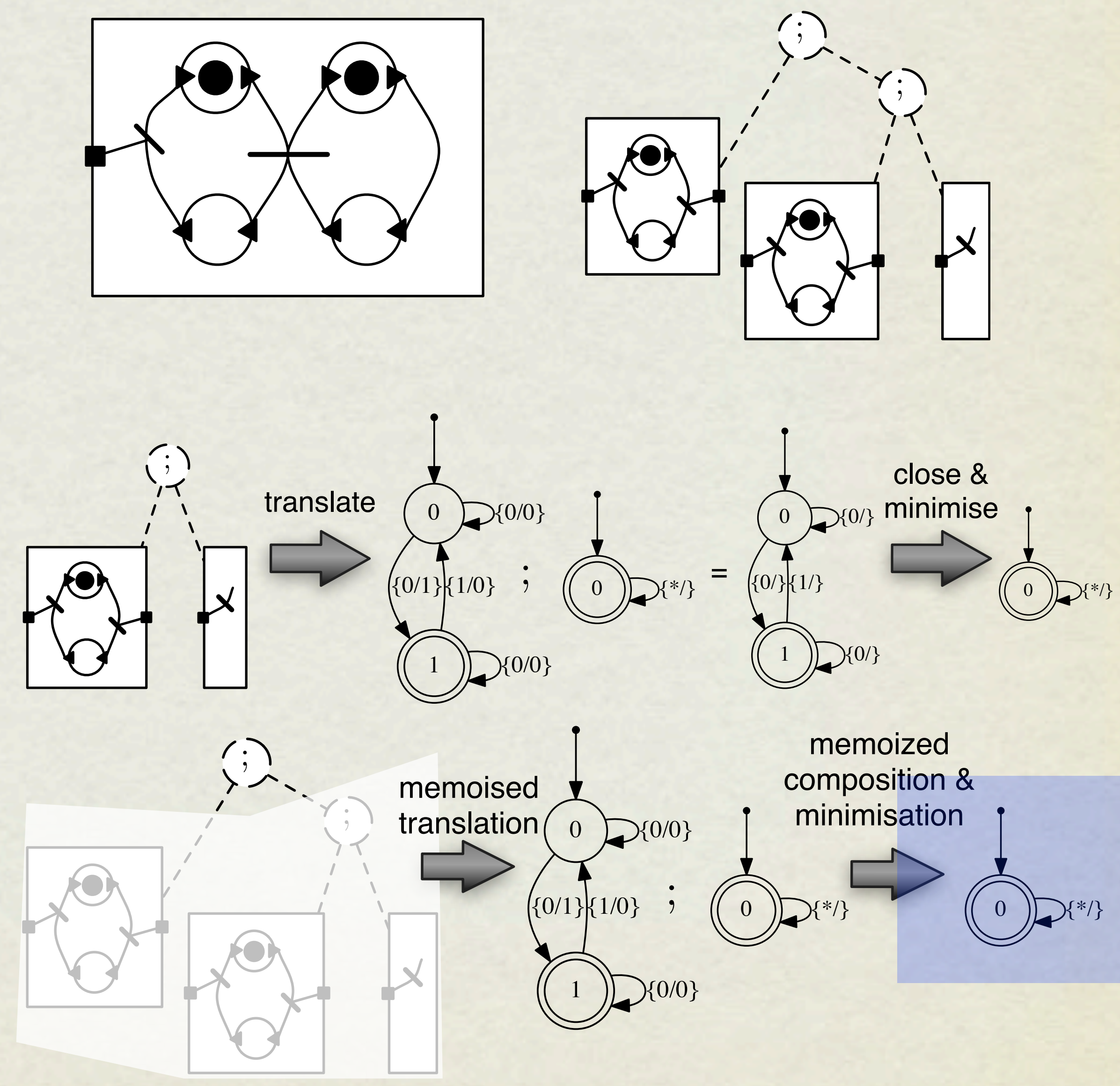
N_1 can reach desired local marking and fire t_2 an arbitrary number of times



N_2 can reach desired local after firing t_2 twice, after which it can be fired an arbitrary additional number of times

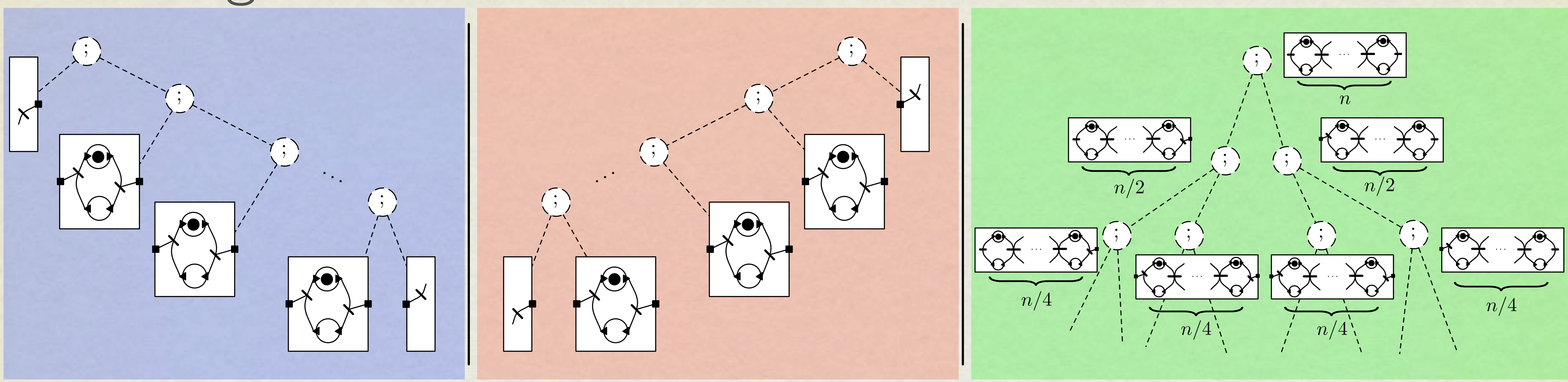


N_1 can reach desired local marking and fire t_2 an arbitrary number of times



PERFORMANCE IS NOT ASSOCIATIVE

right | left | balanced



n	min # firing sequence	Time [s]		
		right	left	balanced
16	136	0.000	0.020	0.008
32	528	0.000	0.140	0.024
64	2080	0.000	1.108	0.172
128	8256	0.000	12.597	2.954
256	32896	0.000	-	74.737
65536	2147516416	0.228	-	-

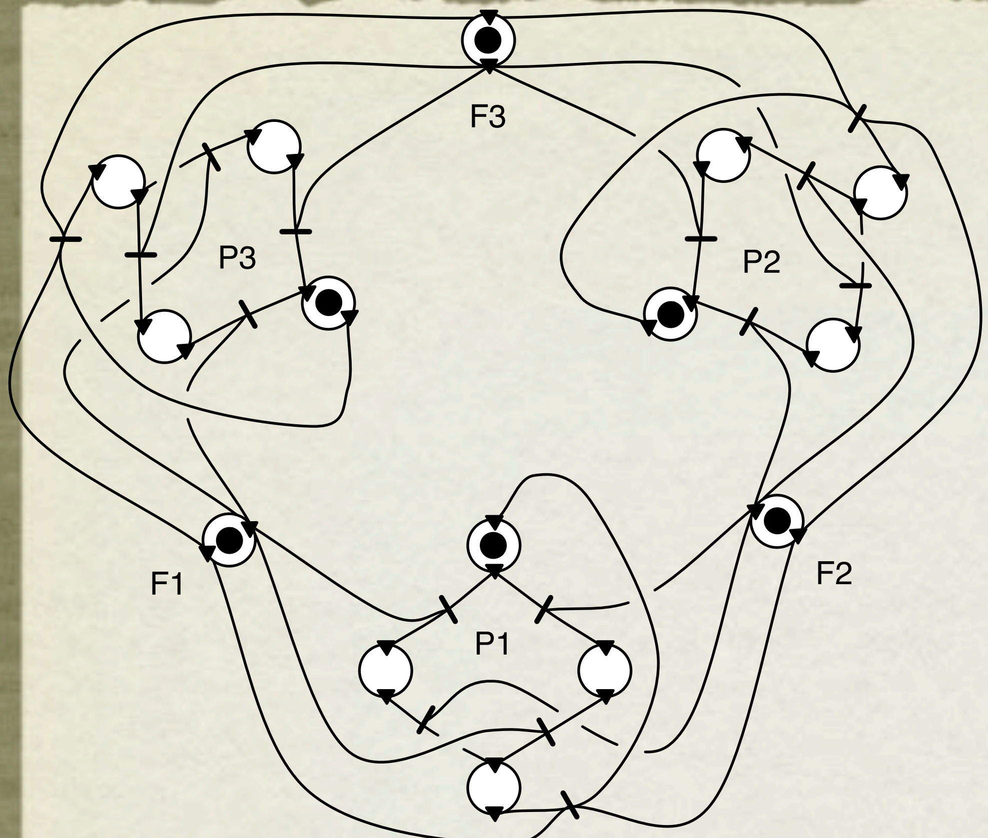
Penrose tool

http://users.ecs.soton.ac.uk/os1v07/Penrose_CALCO13/

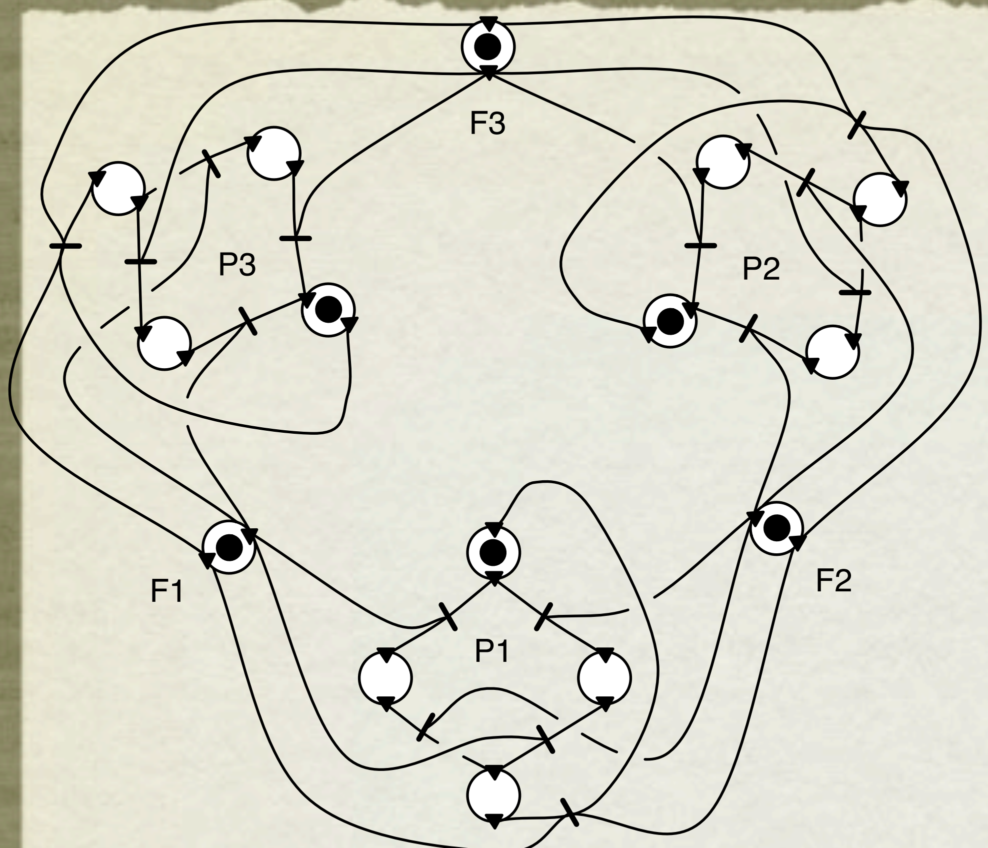
joint work with

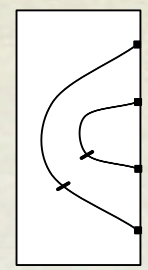
Owen Stephens

PHILOSOPHERS

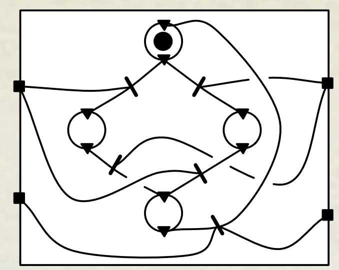


PHILOSOPHERS

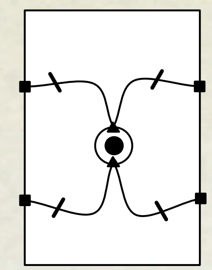




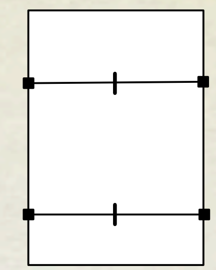
$d_2 : 0 \to 4$



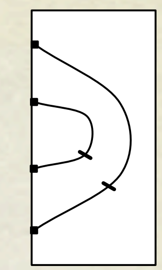
$ph : 2 \to 2$



$fk : 2 \to 2$



$i_2 : 2 \to 2$



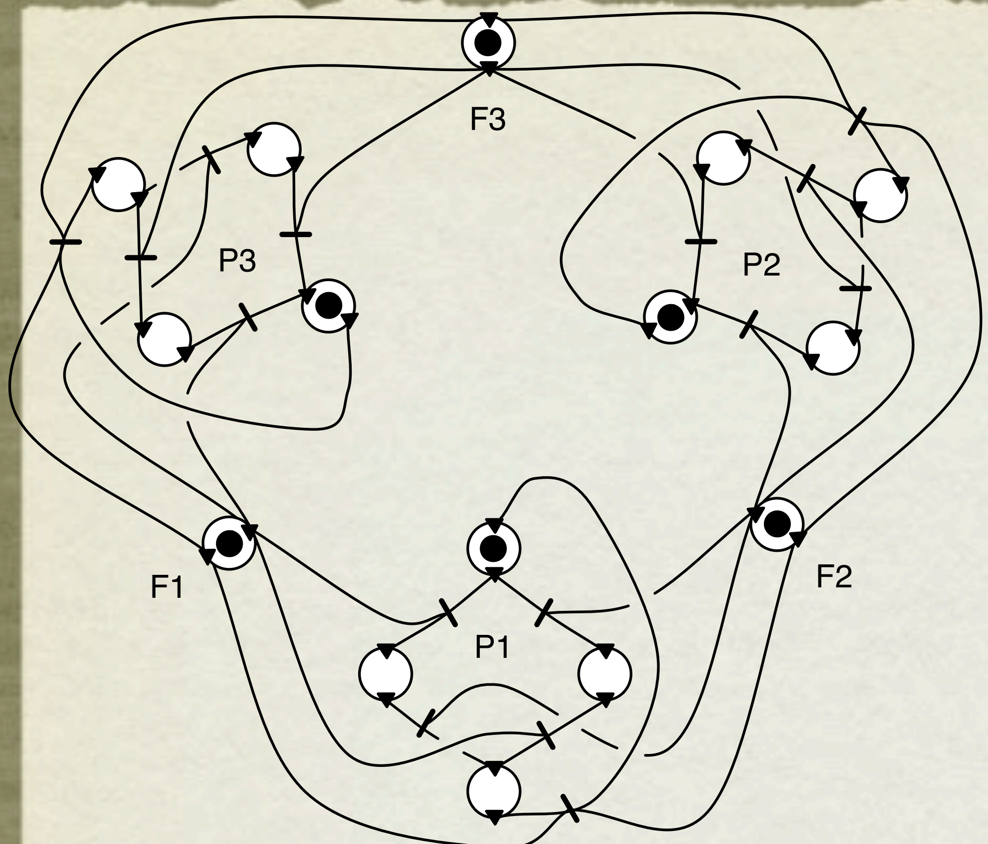
$e_2 : 4 \to 0$

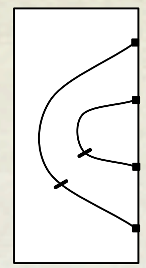
$$PhRow_1 \stackrel{\text{def}}{=} ph \; ; fk$$

$$PhRow_{k+1} \stackrel{\text{def}}{=} ph \; ; fk \; ; PhRow_k$$

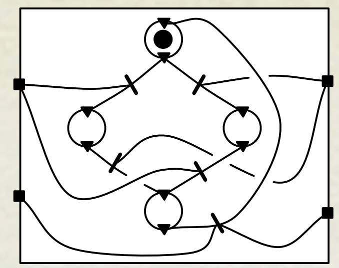
$$Ph_n \stackrel{\text{def}}{=} d_2 \; ; (i_2 \otimes PhRow_n) \; ; e_2$$

PHILOSOPHERS

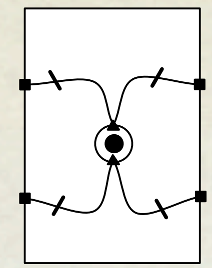




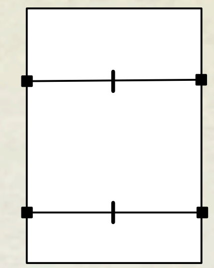
$d_2 : 0 \to 4$



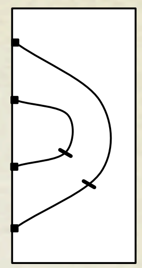
$ph : 2 \to 2$



$fk : 2 \to 2$



$i_2 : 2 \to 2$

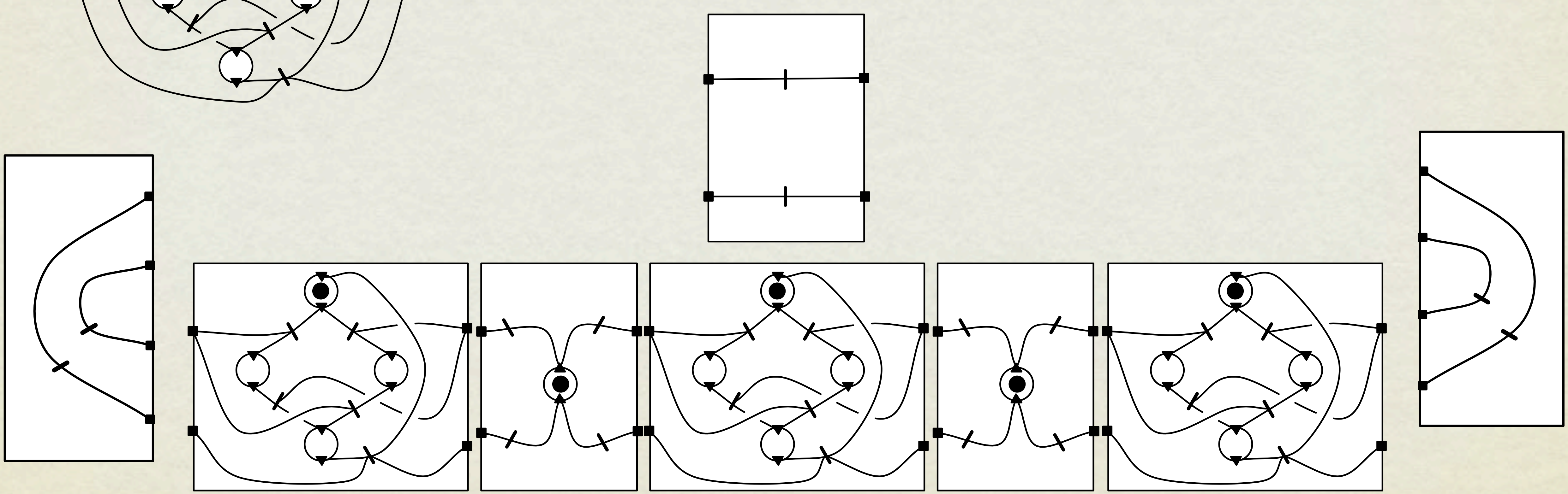


$e_2 : 4 \to 0$

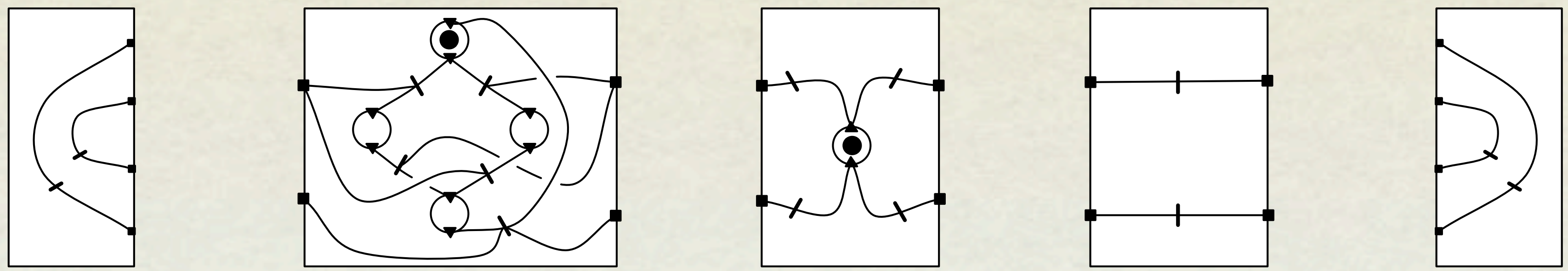
$$PhRow_1 \stackrel{\text{def}}{=} ph \,;\, fk$$

$$PhRow_{k+1} \stackrel{\text{def}}{=} ph \,;\, fk \,;\, PhRow_k$$

$$Ph_n \stackrel{\text{def}}{=} d_2 \,;\, (i_2 \otimes PhRow_n) \,;\, e_2$$



ANALYSING PHILOSOPHERS

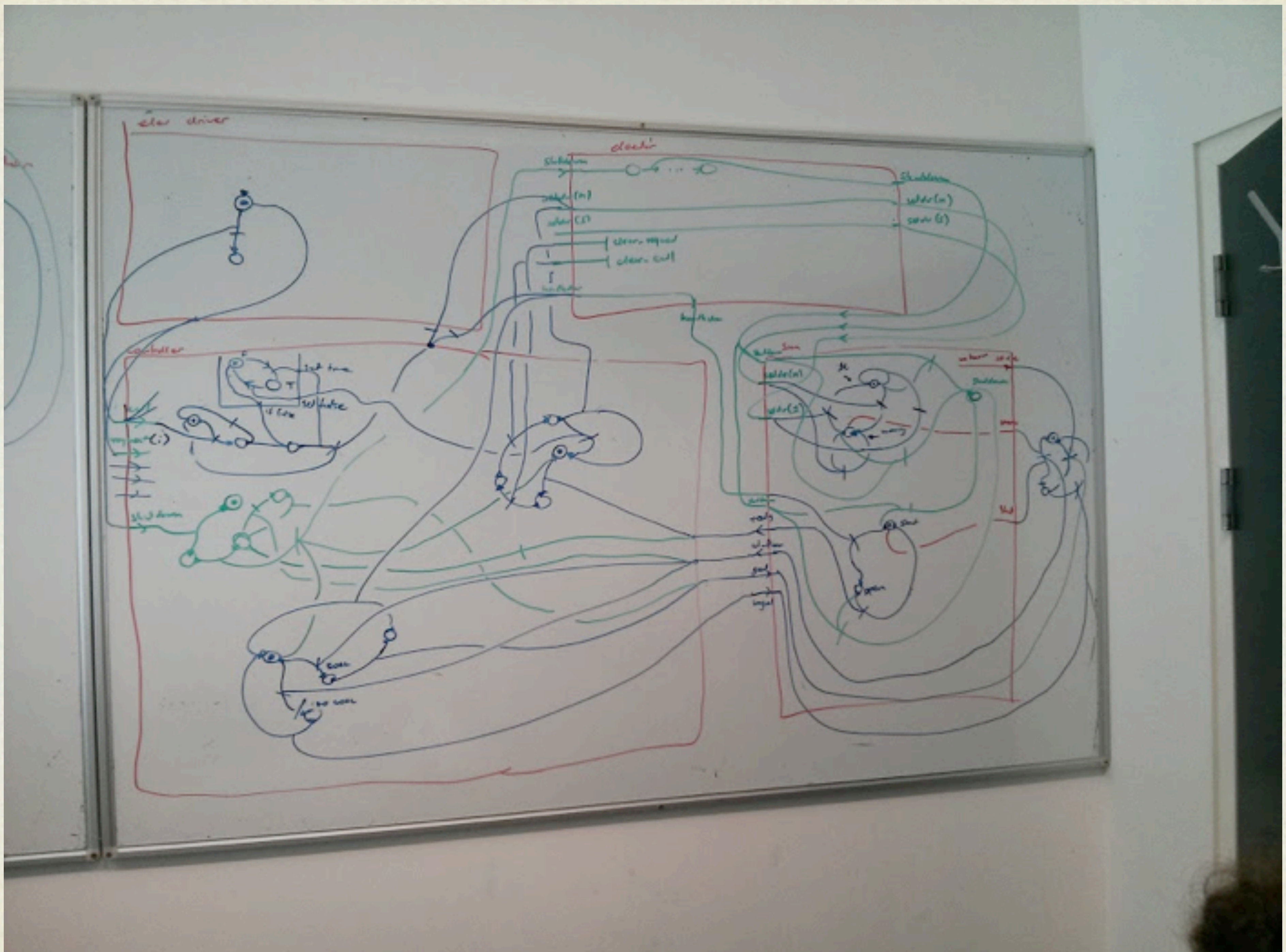


$d_2 : 0 \to 4$ $\quad ph : 2 \to 2$ $\quad fk : 2 \to 2$ $\quad i_2 : 2 \to 2$ $\quad e_2 : 4 \to 0$

$$PhRow_1 \stackrel{\text{def}}{=} ph \; ; \; fk$$
$$PhRow_{k+1} \stackrel{\text{def}}{=} ph \; ; \; fk \; ; \; PhRow_k$$

- Minimization reaches a fixpoint at $PhRow_2$
  - a nice example of when a model-checking technique gives a *proof* for all n.

CORBETT'S ELEVATORS



ROADMAP

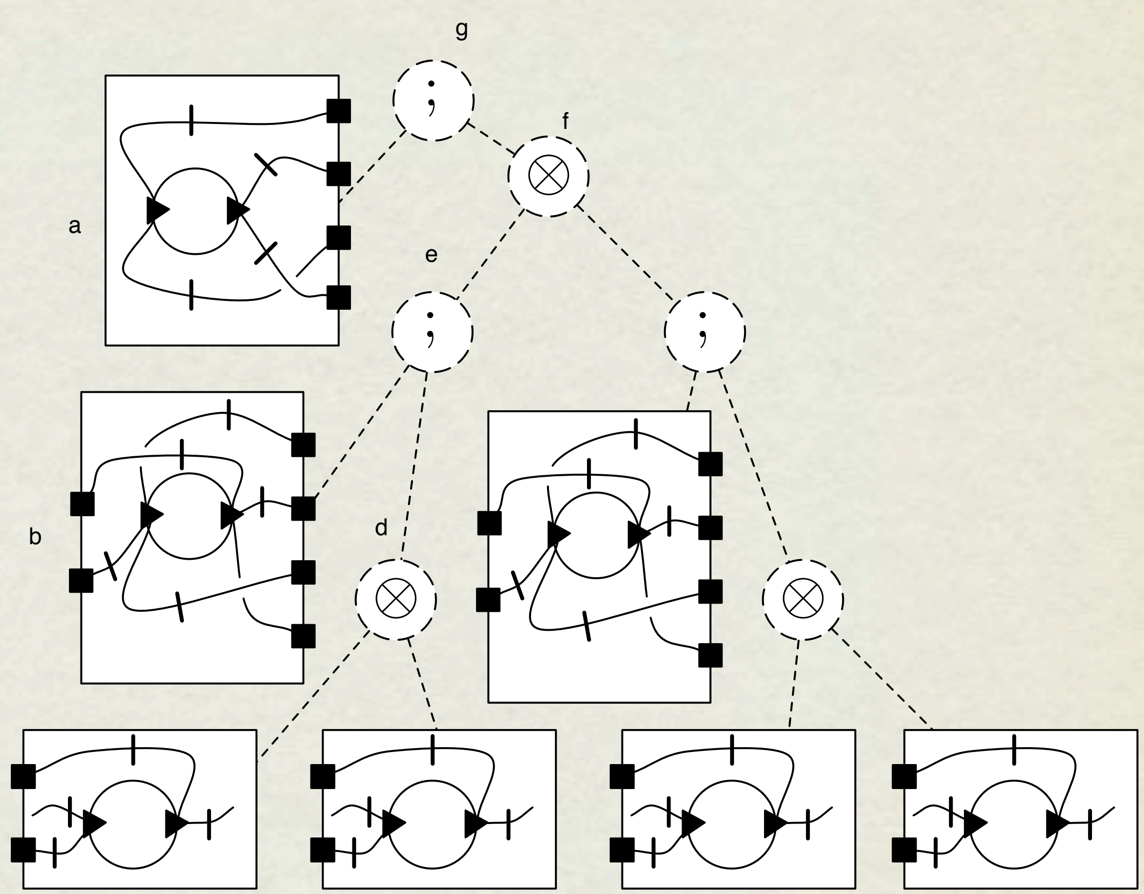
- Automata as model of concurrency - Span(Graph)
- Nets with boundaries
- Application to model checking
- **Work in progress and future work**

WHEN DOES THE TECHNIQUE WORK?

- When the net can be “decomposed well”
  - we don’t want too many places in the leaves (# of states is exponential wrt places)
  - we don’t want big boundaries (# of labels is exponential wrt boundary size)
- AND when the state-space “grows slowly” as we recompose

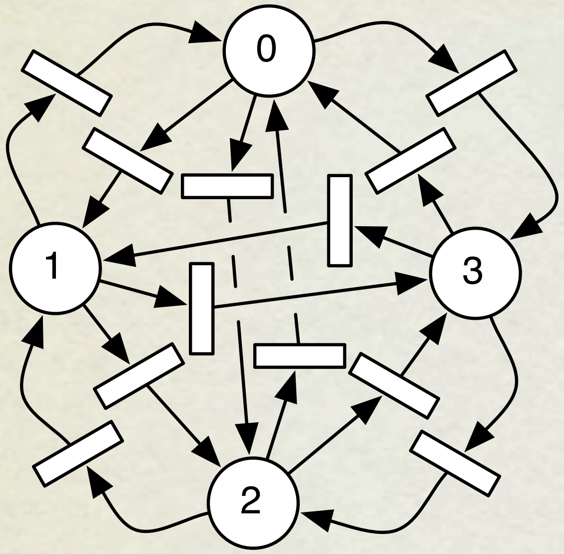
DECOMPOSITION WIDTH

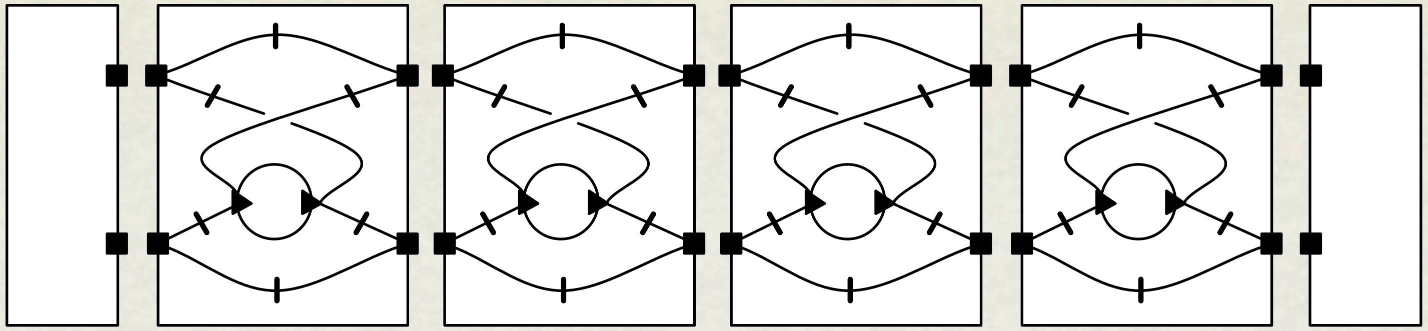
- A decomposition has width k when
  - all leaves have max(#places,boundary)≤k
  - all complete subtrees have boundary size≤k
  - e.g. the composition on the right has width 4



DECOMPOSITION WIDTH

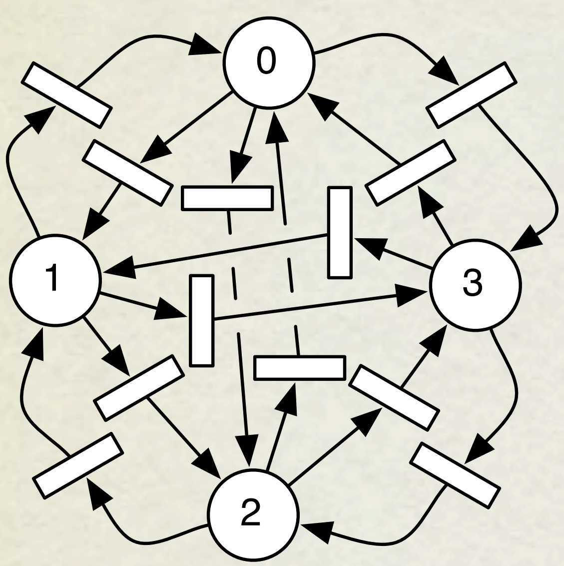
- e.g. cliques

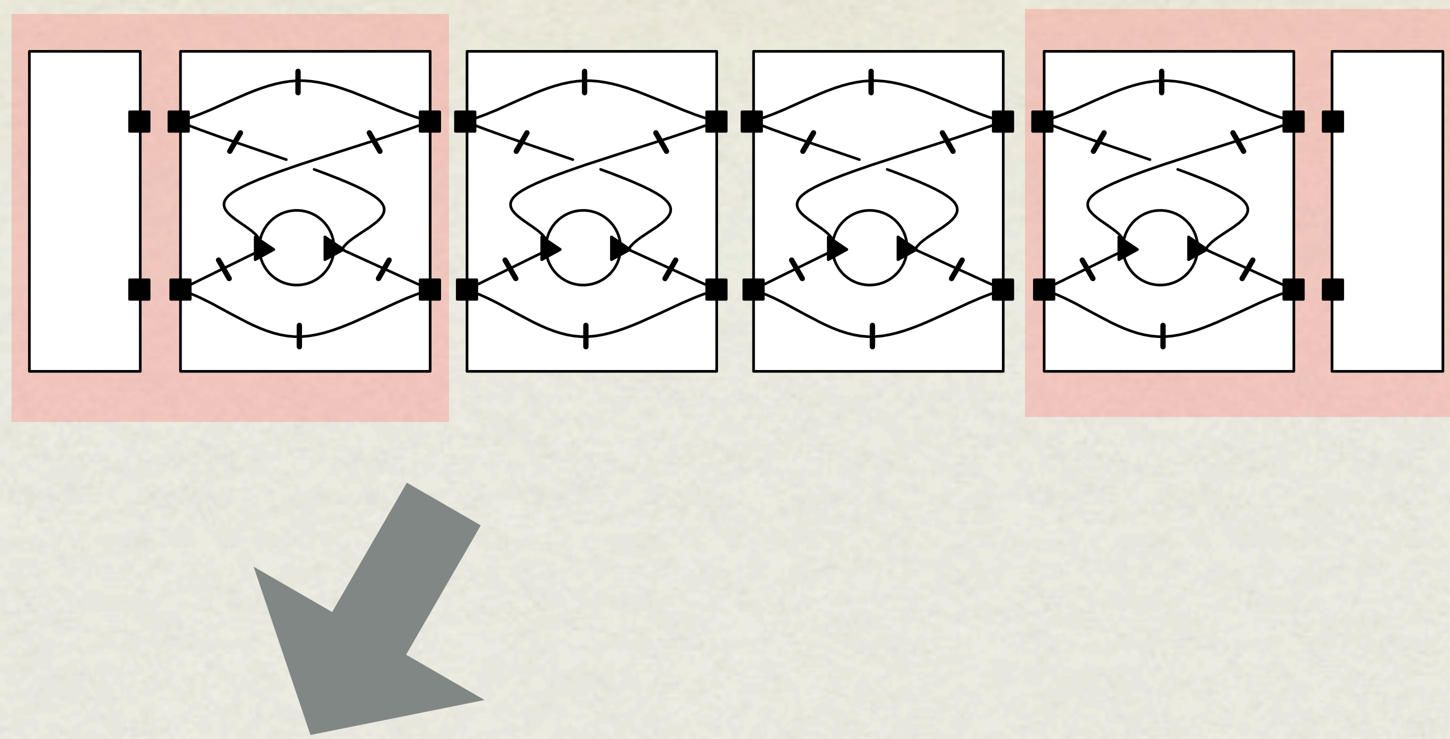


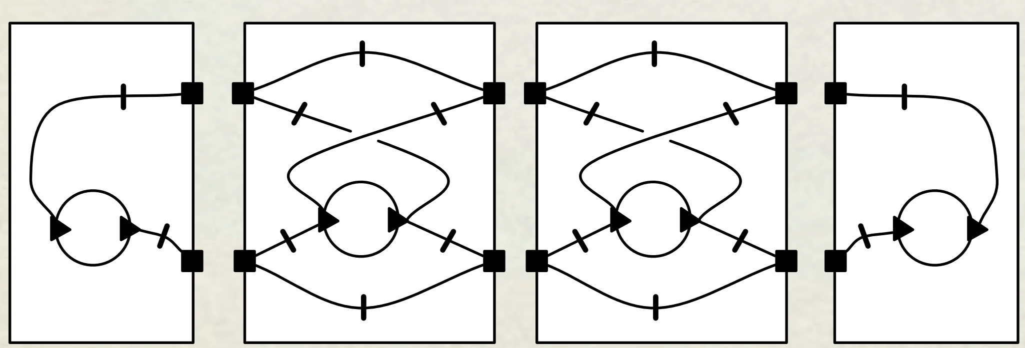


DECOMPOSITION WIDTH

- e.g. cliques

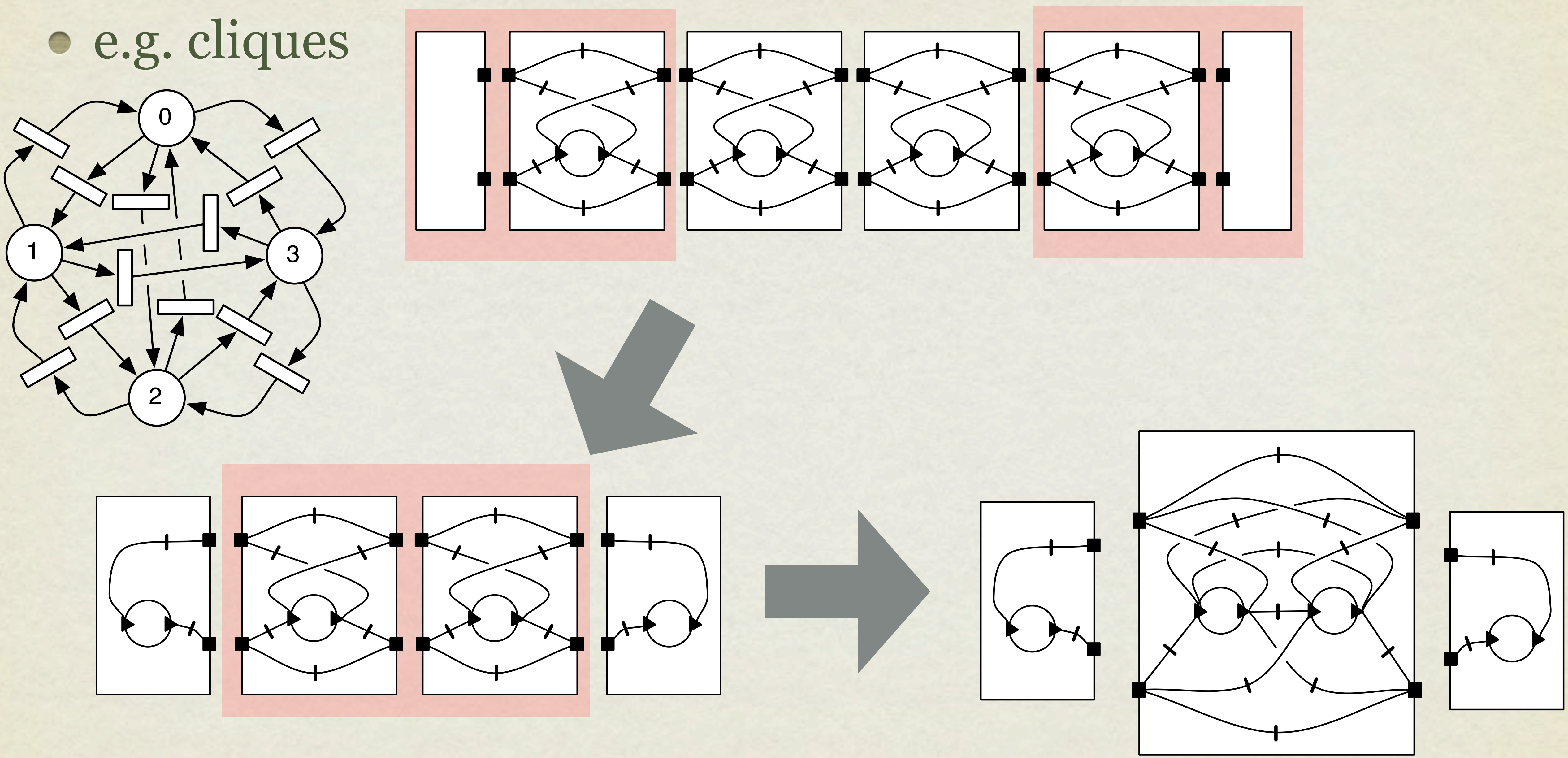






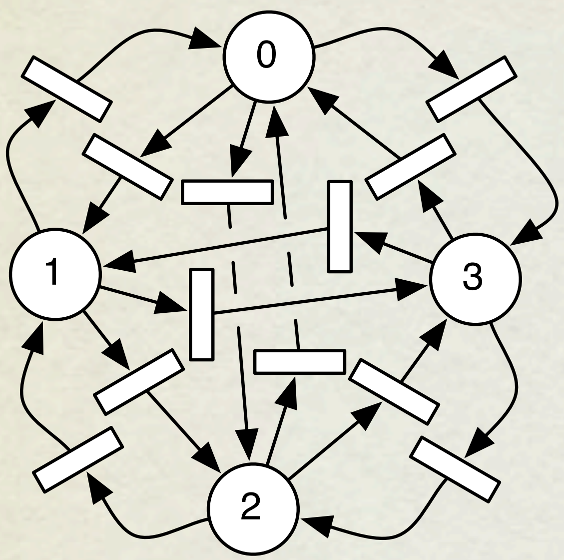
DECOMPOSITION WIDTH

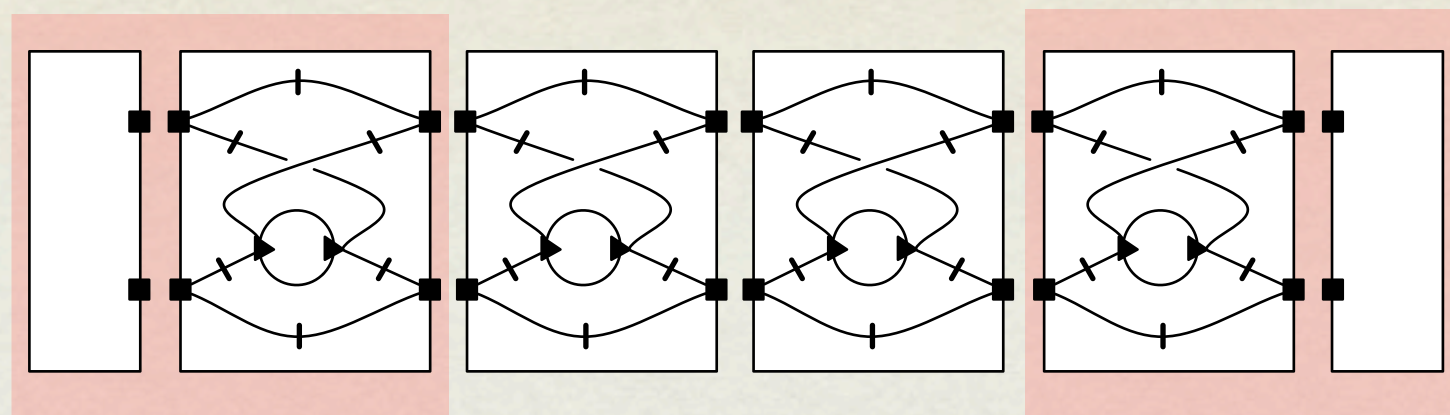
- e.g. cliques

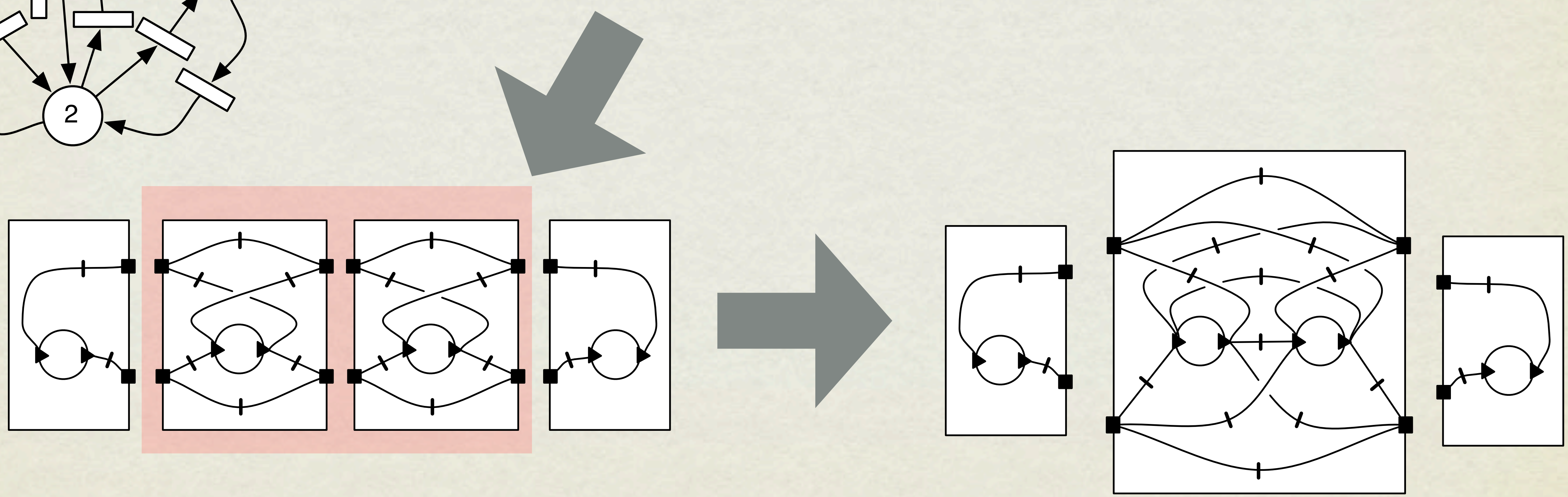


DECOMPOSITION WIDTH

- e.g. cliques

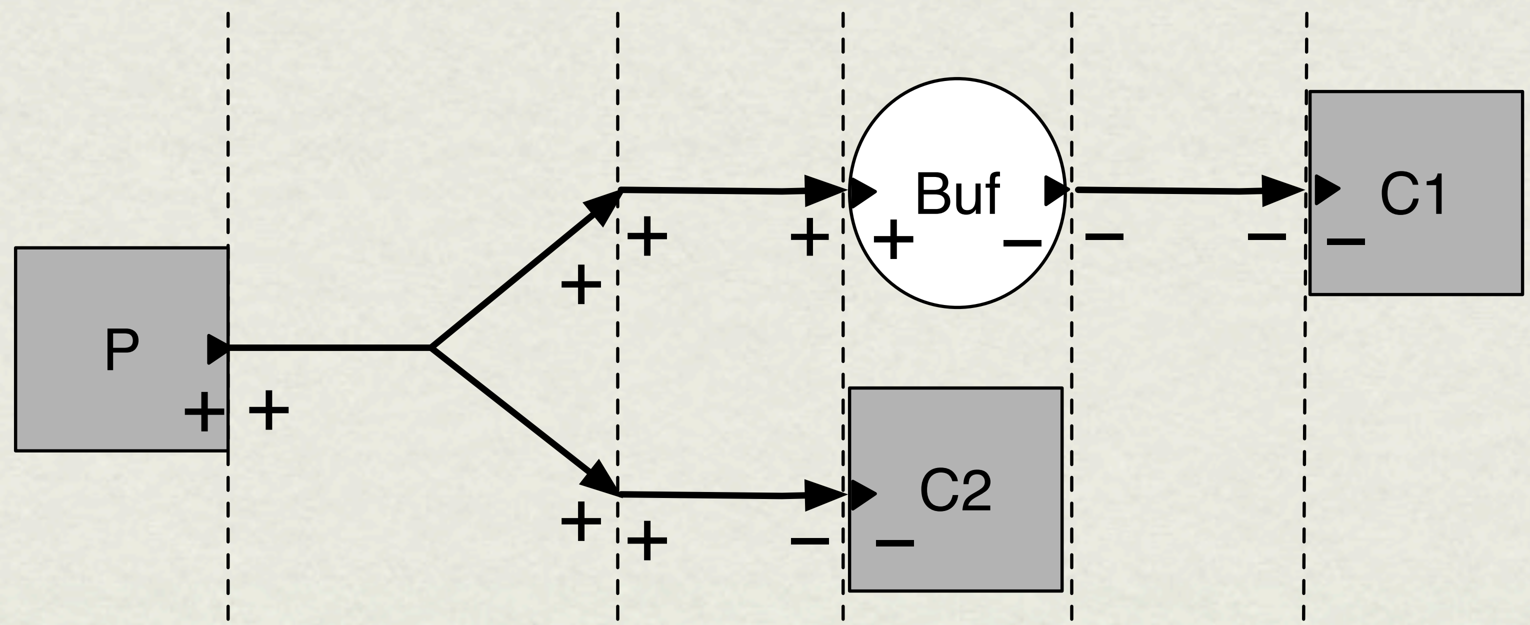






related to rank width of graphs

SYNCHRONISATION AS OO PROGRAMMING PRIMITIVE



RELATED WORK

- Body of work on compositional model checking via interface theories going back to Clarke
- Work on compositional algebras of Petri nets going back to Mazurkiewicz
- Work on reachability in bounded nets using unfolding going back to McMillan
- Body of work on algebraic approaches to nets, including the Petri box calculus of Koutny, Esparza and Best

THE END