Compositional Reachability in Petri Nets

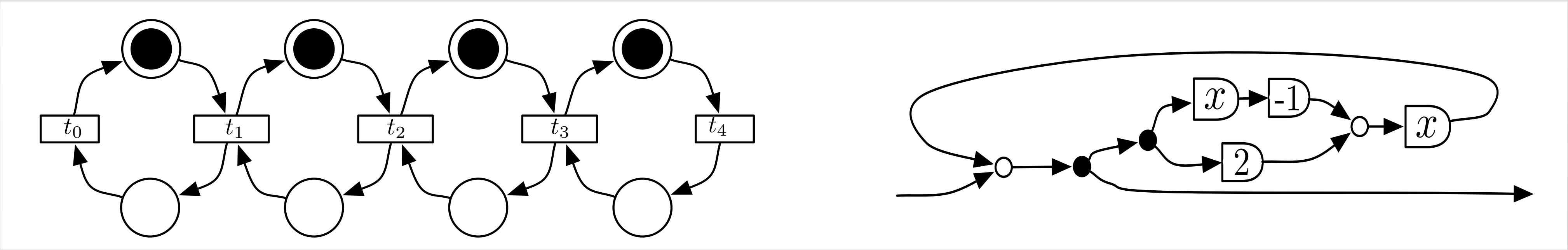
Julian Rathke,
Pawel Sobocinski,
Owen Stephens
University of Southampton

NII Shonan
Sept 2015

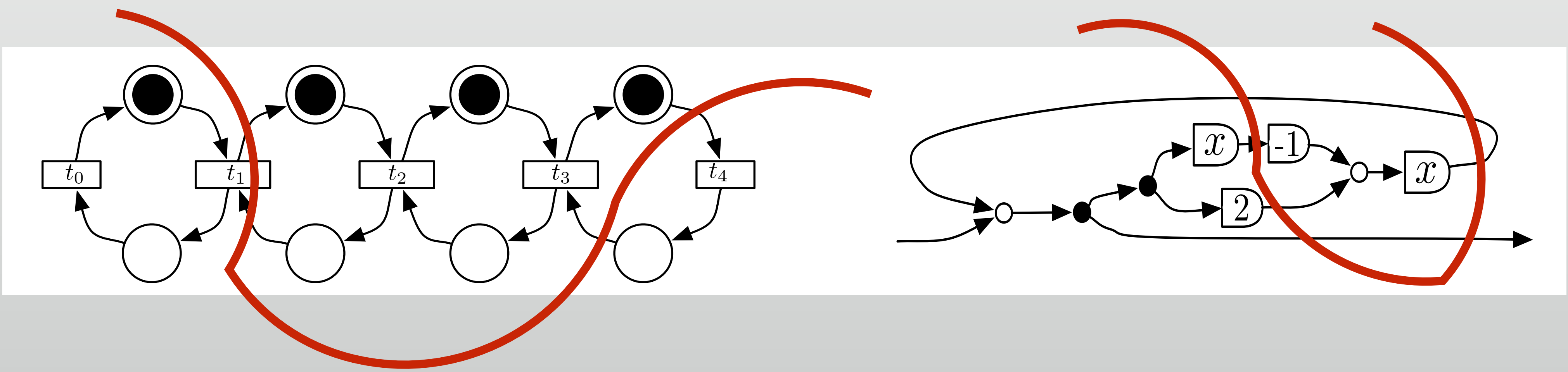
Interaction is key

- Interaction between entities is fundamental to understanding their semantics.
- Labelled transitions are usefully understood as descriptions of a contribution to an interaction rather than as structural properties of a term
- We can use these to study algebraic properties of systems
- This principle underpins all manner of computation …

… even for Petri nets

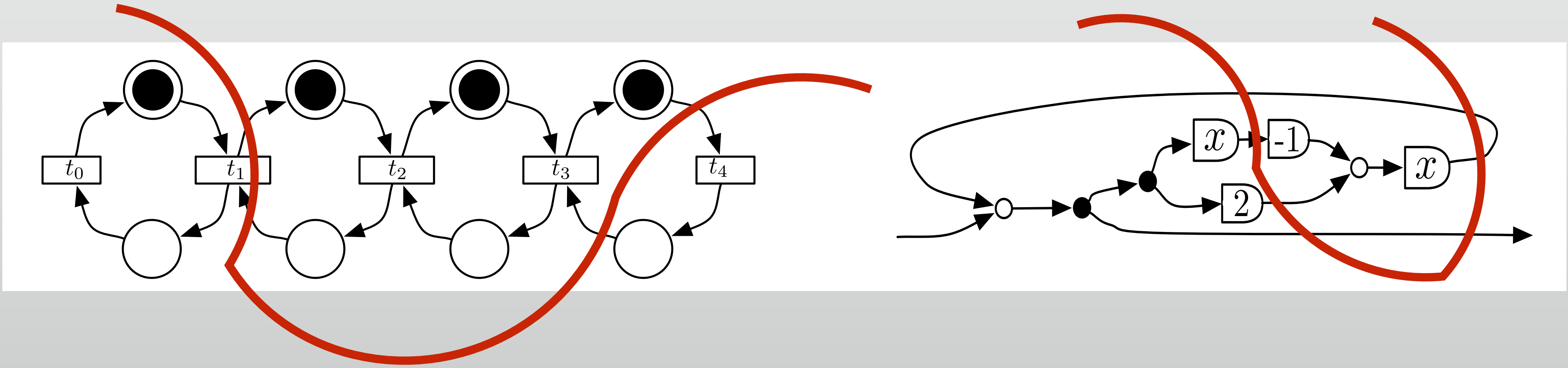


… even for Petri nets



… even for Petri nets

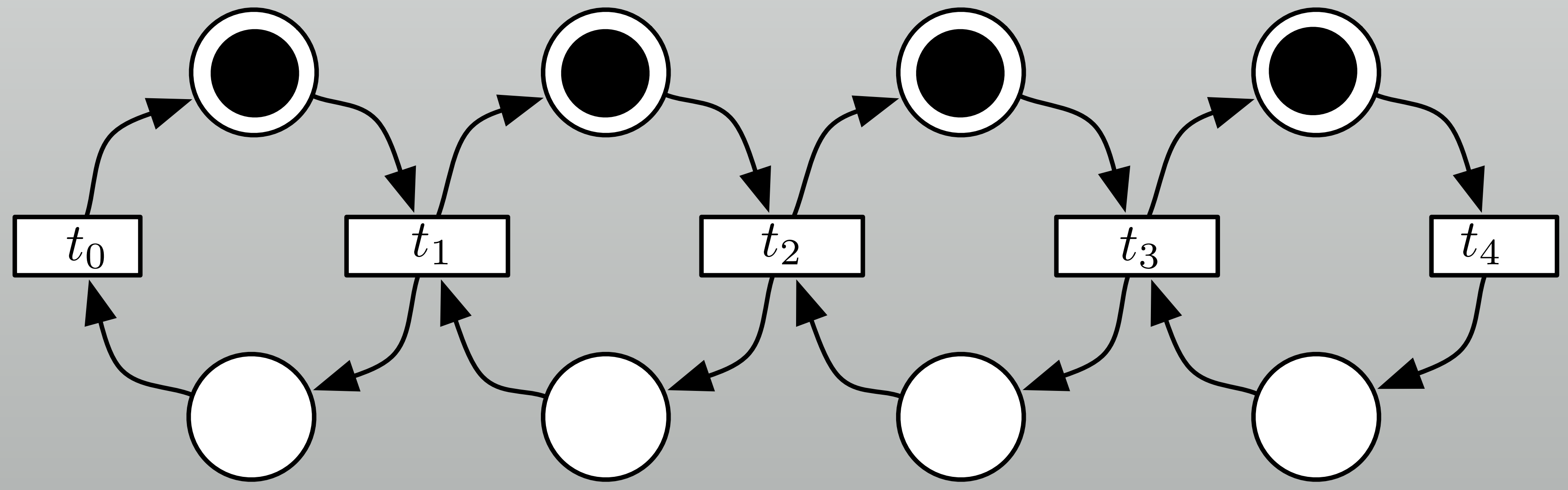
- We look for **compositional** algebras of systems



- Once we have a compositional model, we can use it for
  - parametric verification
  - algorithmic improvements in reachability / coverability checking through divide and conquer
- This talk: we focus on Elementary Net Systems (1-bounded nets)

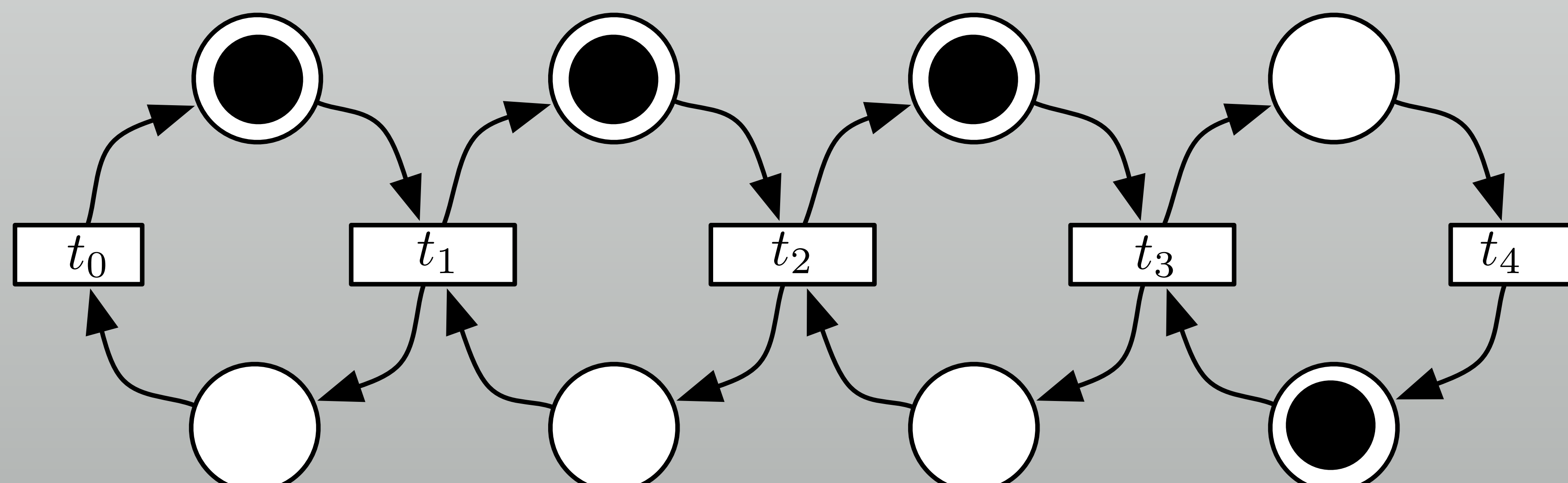
Reachability

Given a (1-bounded) Petri Net with a particular marking, Is there an execution sequence that will leave the net in some other specified marking?



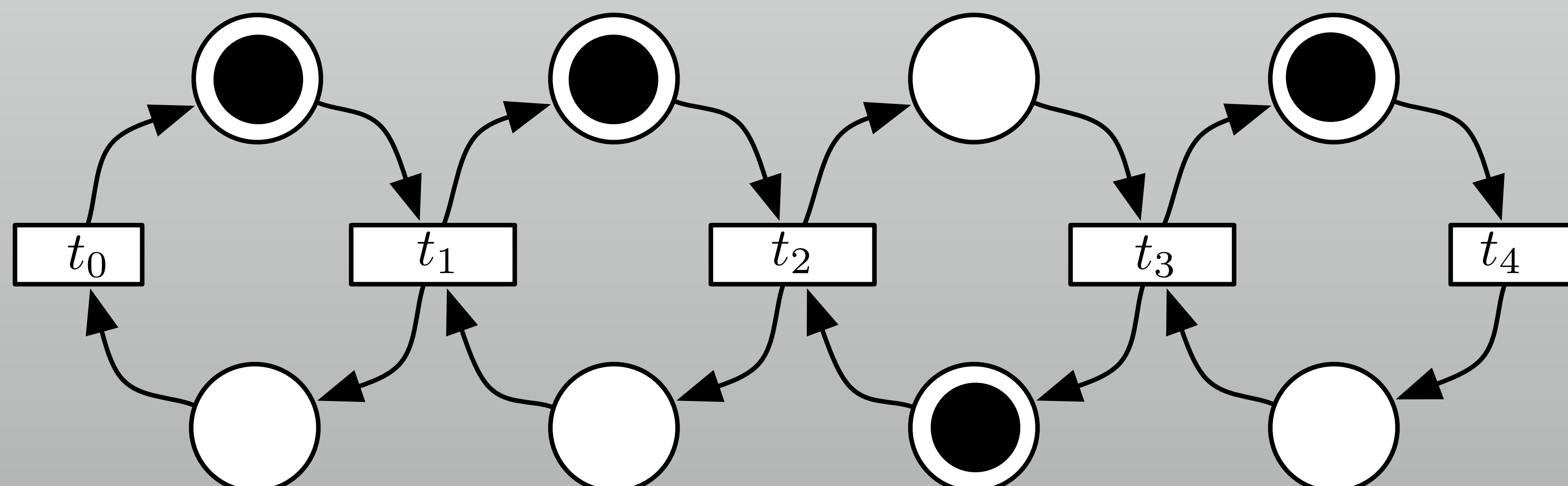
Reachability

Given a (1-bounded) Petri Net with a particular marking, Is there an execution sequence that will leave the net in some other specified marking?



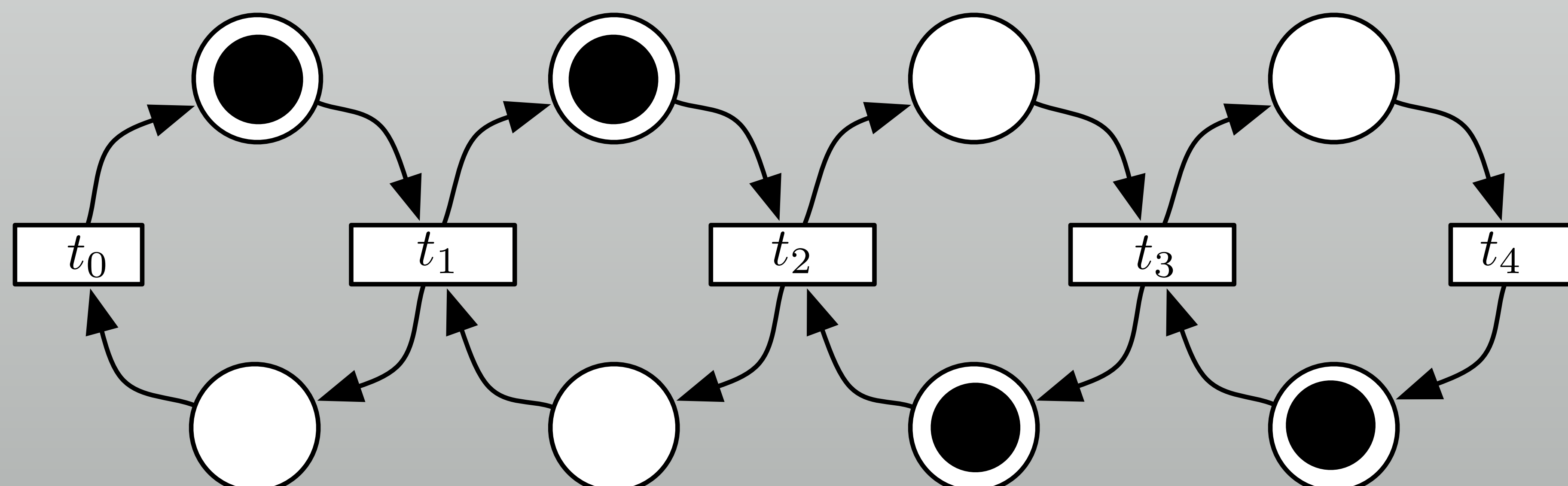
Reachability

Given a (1-bounded) Petri Net with a particular marking, Is there an execution sequence that will leave the net in some other specified marking?



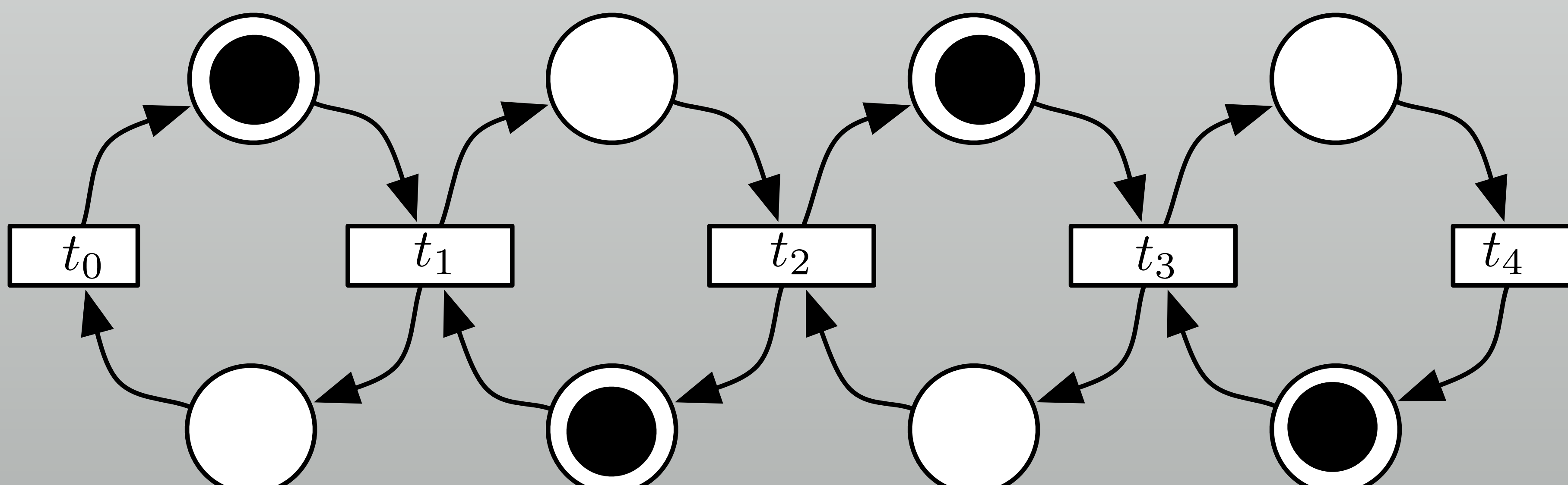
Reachability

Given a (1-bounded) Petri Net with a particular marking, Is there an execution sequence that will leave the net in some other specified marking?



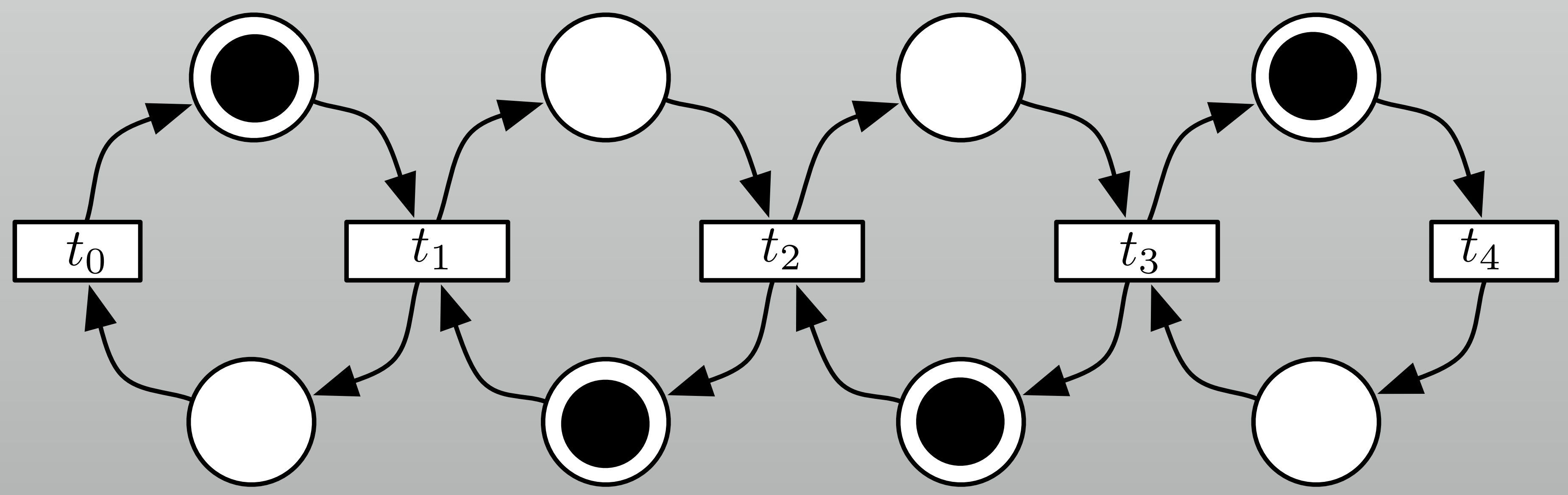
Reachability

Given a (1-bounded) Petri Net with a particular marking, Is there an execution sequence that will leave the net in some other specified marking?



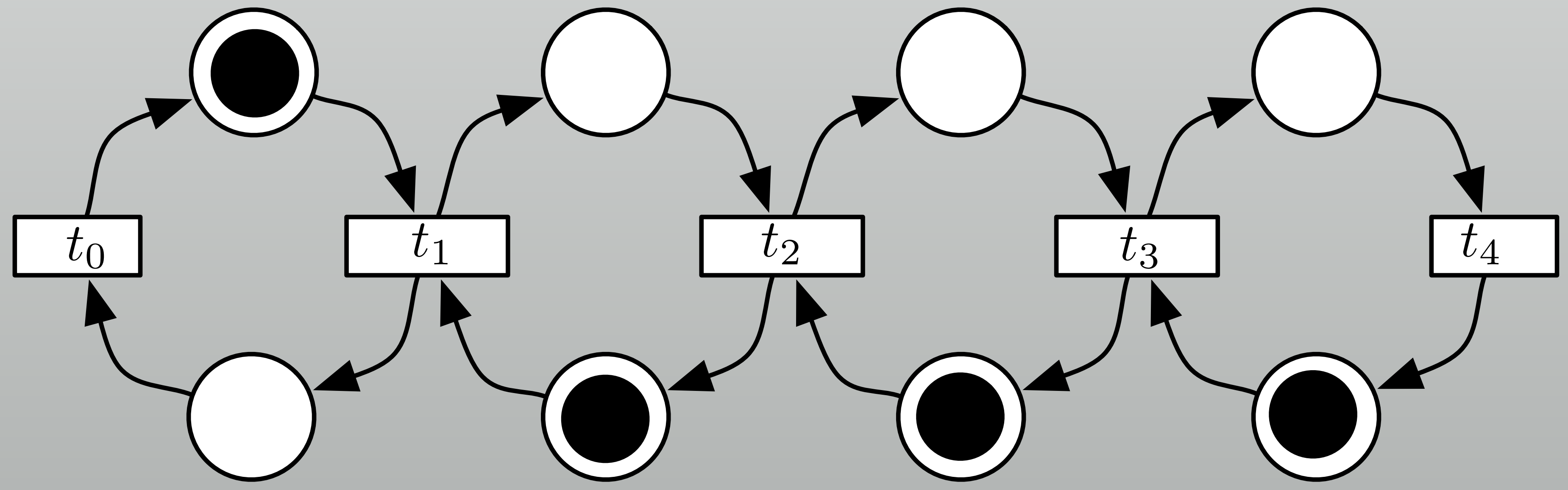
Reachability

Given a (1-bounded) Petri Net with a particular marking, Is there an execution sequence that will leave the net in some other specified marking?



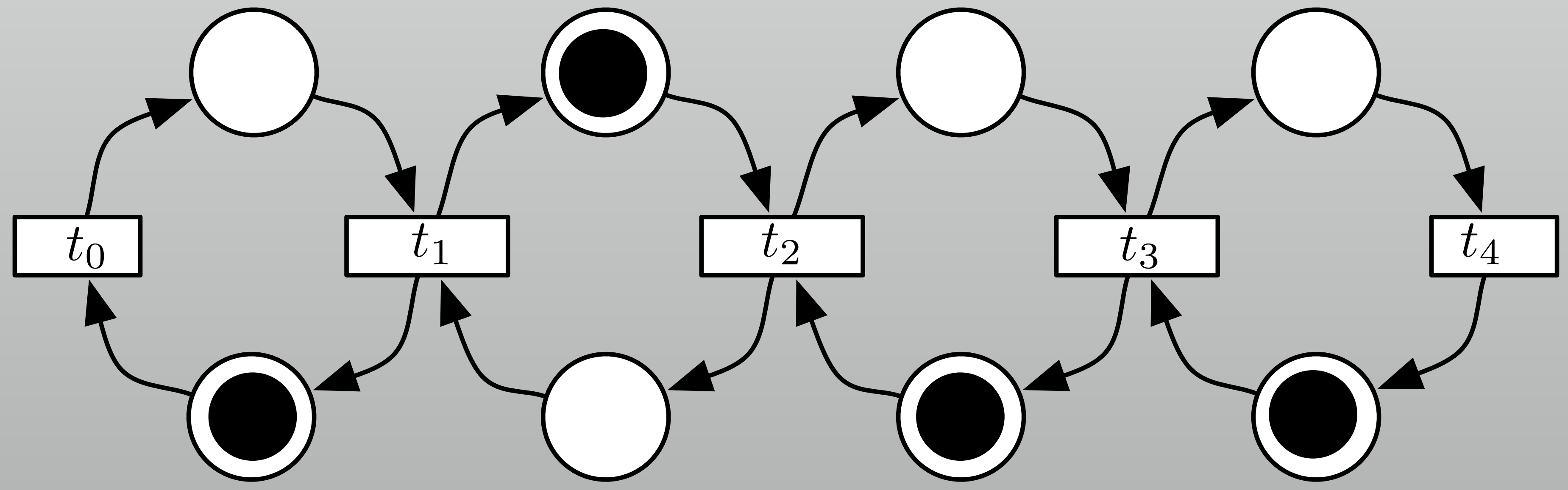
Reachability

Given a (1-bounded) Petri Net with a particular marking,
Is there an execution sequence that will leave the net in
some other specified marking?



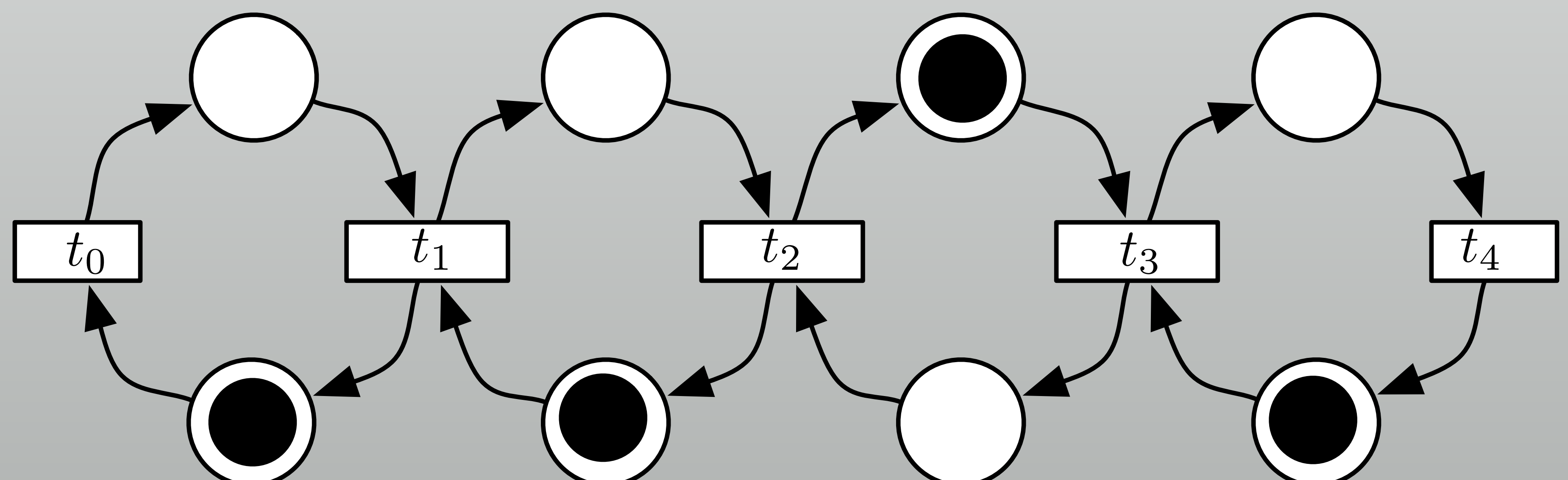
Reachability

Given a (1-bounded) Petri Net with a particular marking, Is there an execution sequence that will leave the net in some other specified marking?



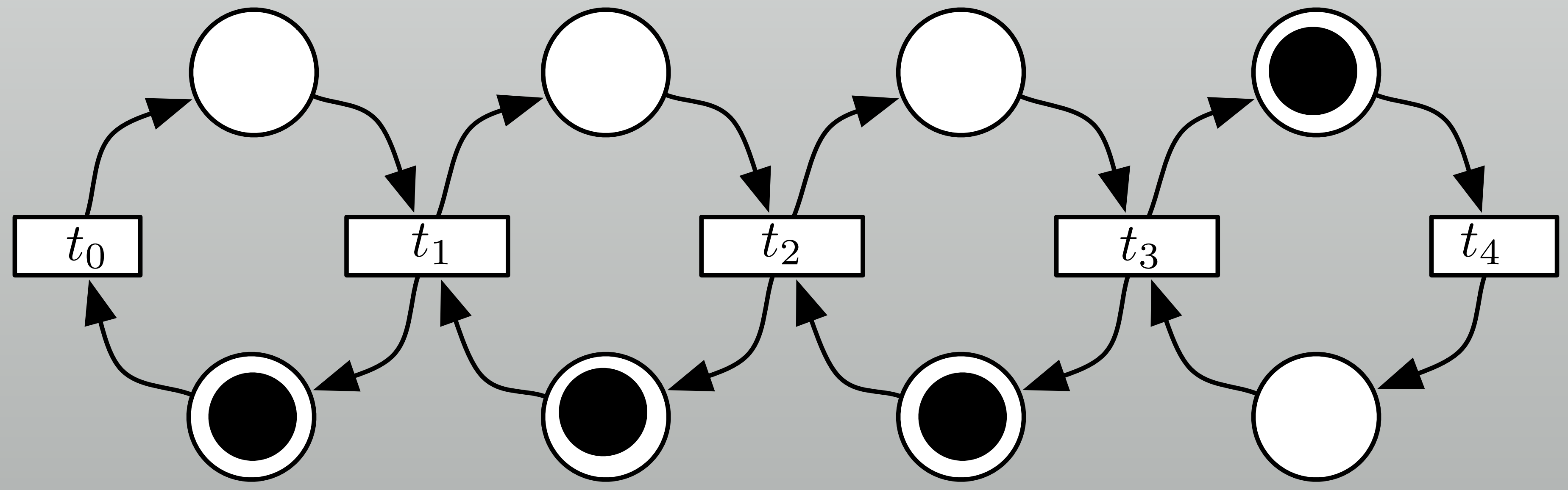
Reachability

Given a (1-bounded) Petri Net with a particular marking, Is there an execution sequence that will leave the net in some other specified marking?



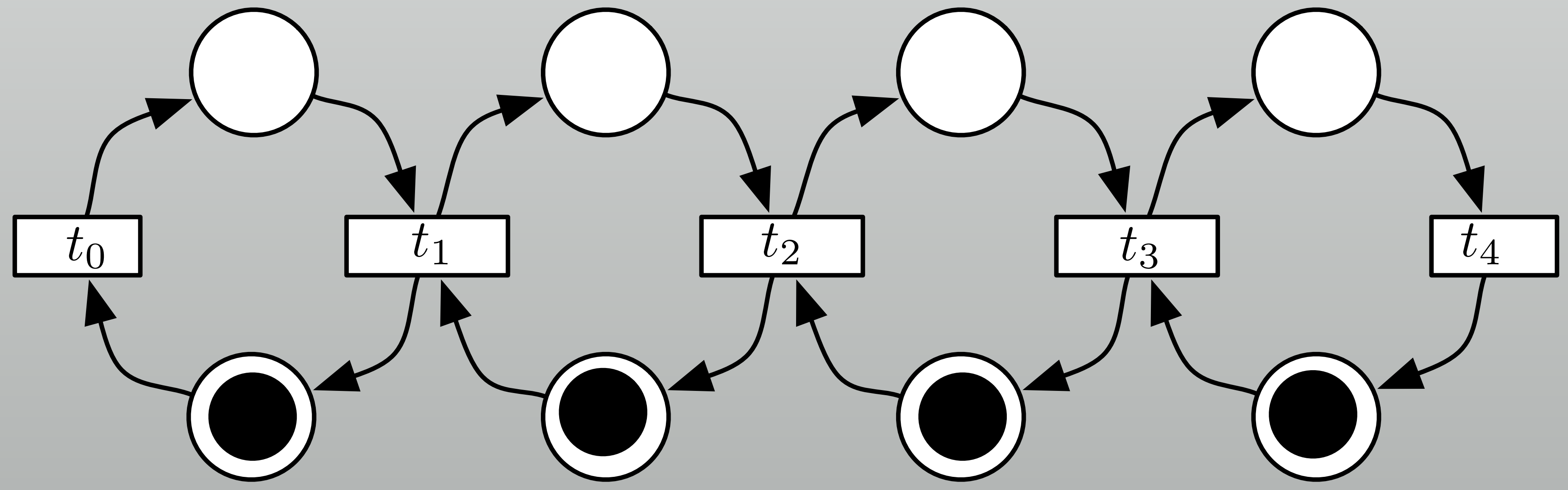
Reachability

Given a (1-bounded) Petri Net with a particular marking, Is there an execution sequence that will leave the net in some other specified marking?

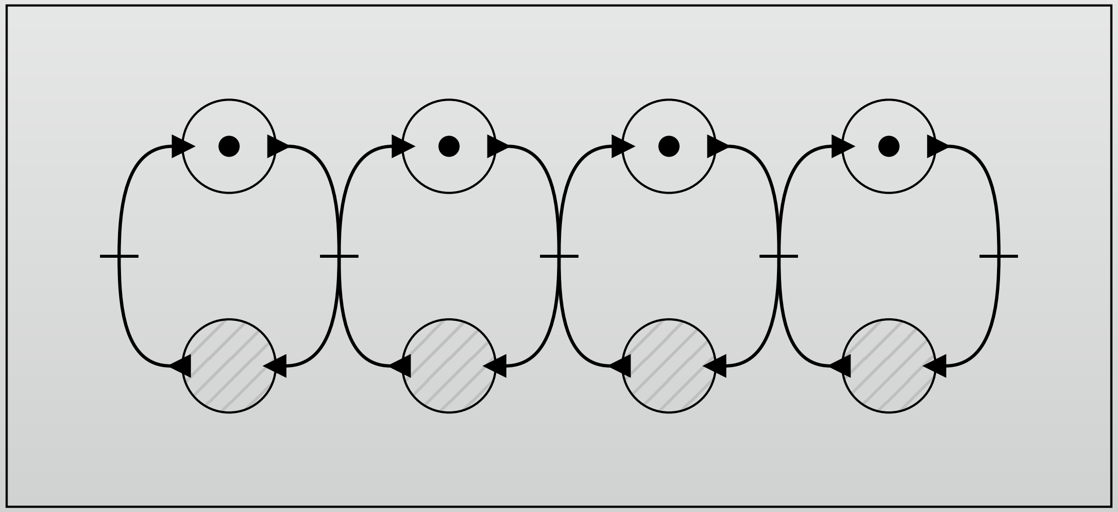


Reachability

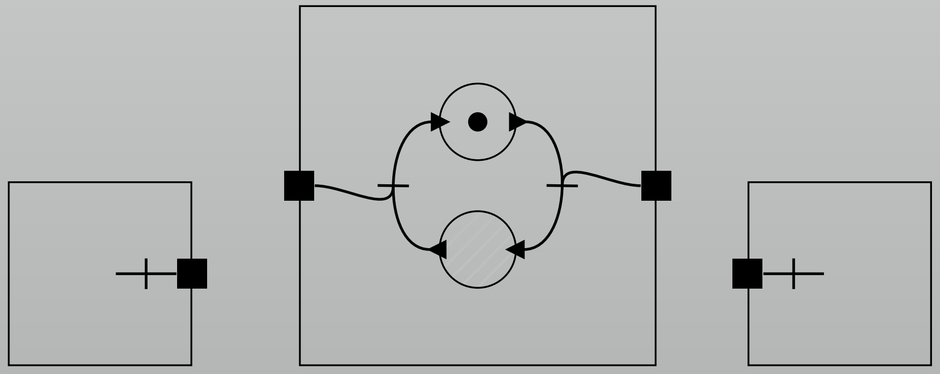
Given a (1-bounded) Petri Net with a particular marking, Is there an execution sequence that will leave the net in some other specified marking?



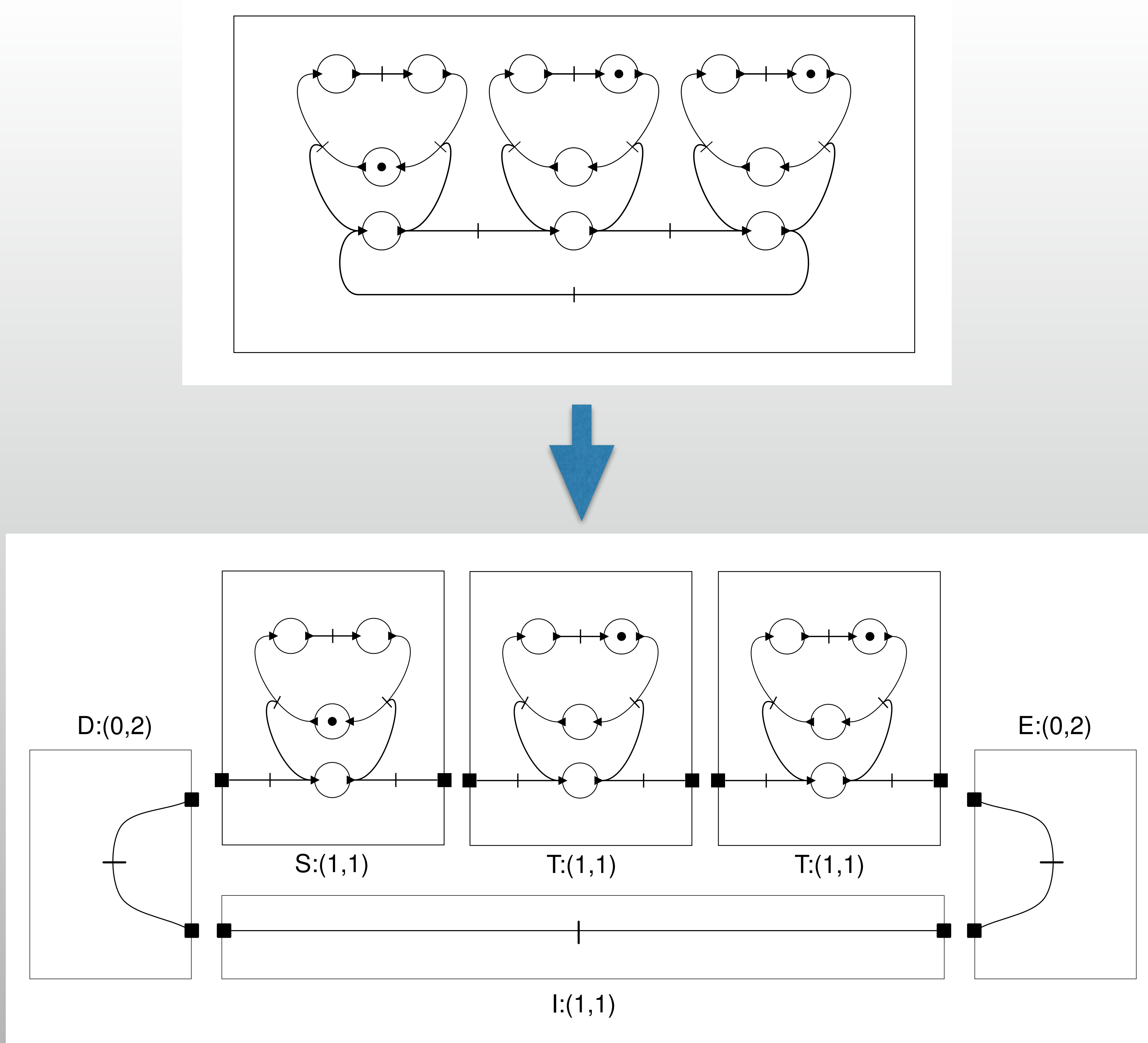
Petri nets with Boundaries (PNB)



$$\top \ ; b_1 \ ; b_1 \ ; b_1 \ ; b_1 \ ; \bot$$

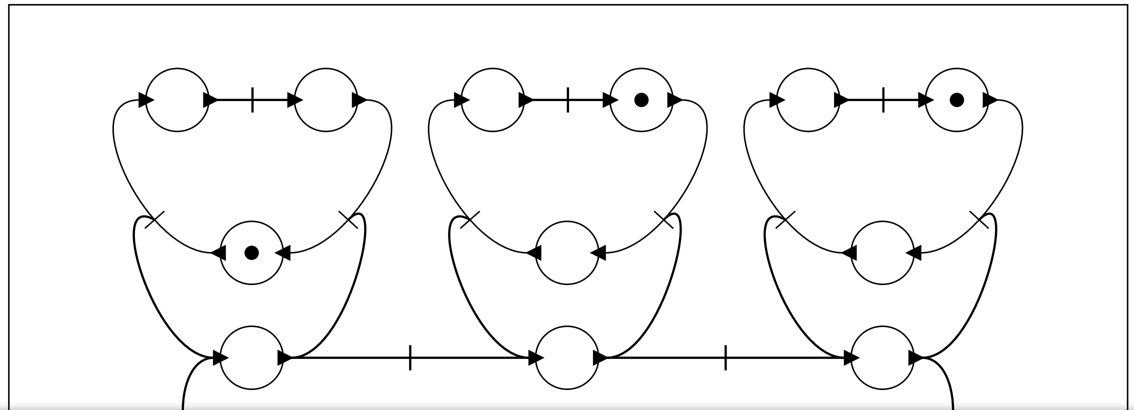


$\top$ b_1 $\bot$



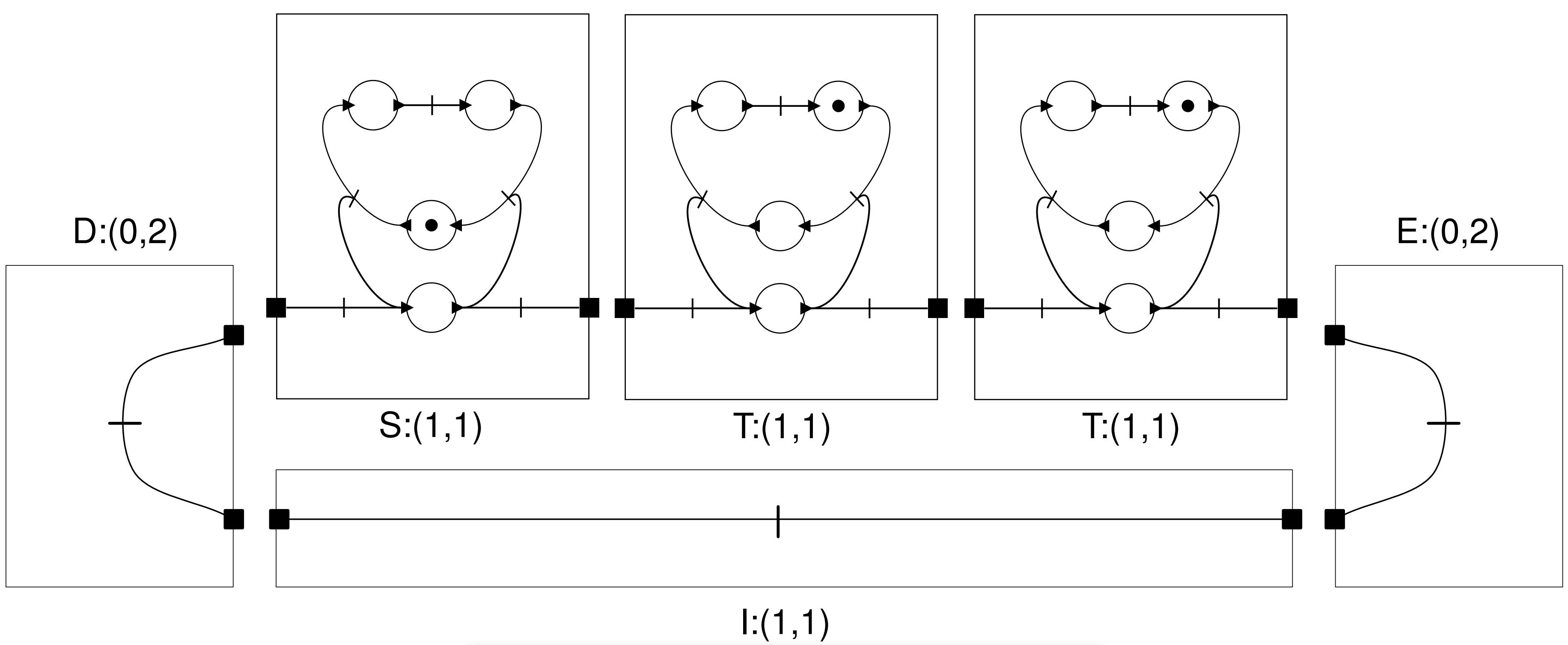
$$D \,;\, ((S \,;\, T \,;\, T) \otimes I) \,;\, E \qquad (\dagger)$$

Fig. 6: A token ring network as a PNB expression



The algebraic expression reflects the system communication topology

(we can see how the net is wired up from looking at the term!)



$$D \mathbin{;} ((S \mathbin{;} T \mathbin{;} T) \otimes I) \mathbin{;} E \qquad (\dagger)$$

Fig. 6: A token ring network as a PNB expression

Describing interaction

- LTS labels indicate when a transition that is connected to a boundary port has been fired

This assignment is **functorial**
(the LTS of a composed net is the composition of component LTSs)

Describing interaction

- LTS labels indicate when a transition that is connected to a boundary port has been fired

This assignment is **functorial**
(the LTS of a composed net is the composition of component LTSs)

Describing interaction

- LTS labels indicate when a transition that is connected to a boundary port has been fired

This assignment is **functorial**
(the LTS of a composed net is the composition of component LTSs)

Describing interaction

- LTS labels indicate when a transition that is connected to a boundary port has been fired

This assignment is **functorial**
(the LTS of a composed net is the composition of component LTSs)

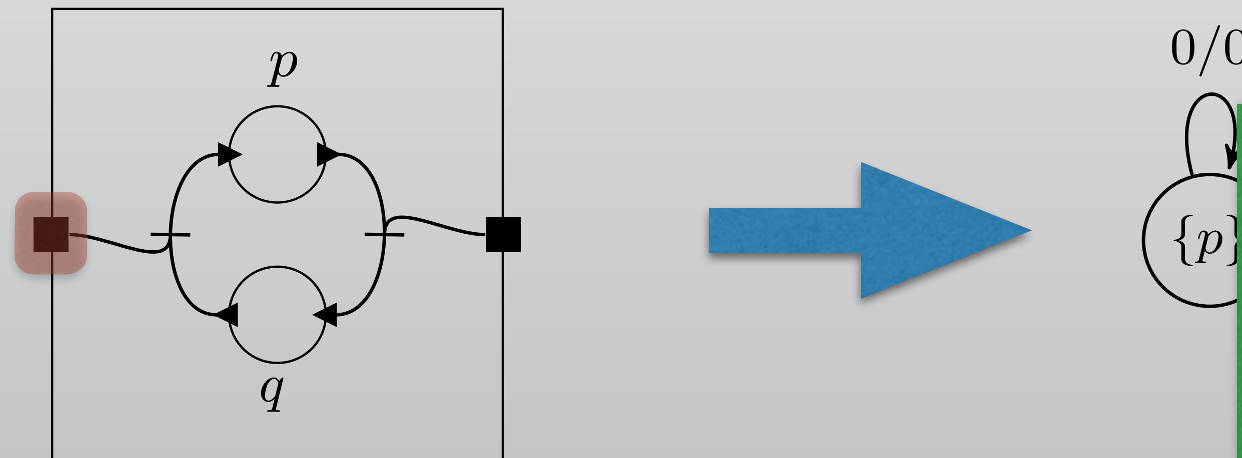
Describing interaction

- LTS labels indicate when a transition that is connected to a boundary port has been fired

This assignment is **functorial**
(the LTS of a composed net is the composition of component LTSs)

Describing interaction

- LTS labels indicate when a transition that is connected to a boundary port has been fired



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

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

This assignment is **fu**…

(the LTS of a c…posed net is the

composition o… component LTSs)

Reachability, compositionally

- NFA captures the **interaction patterns** which allow the subnet to reach the desired local marking

Reachability, compositionally

- NFA captures the **interaction patterns** which allow the subnet to reach the desired local marking

Any language equivalent NFA describes the same interaction patterns. e.g. a minimal NFA

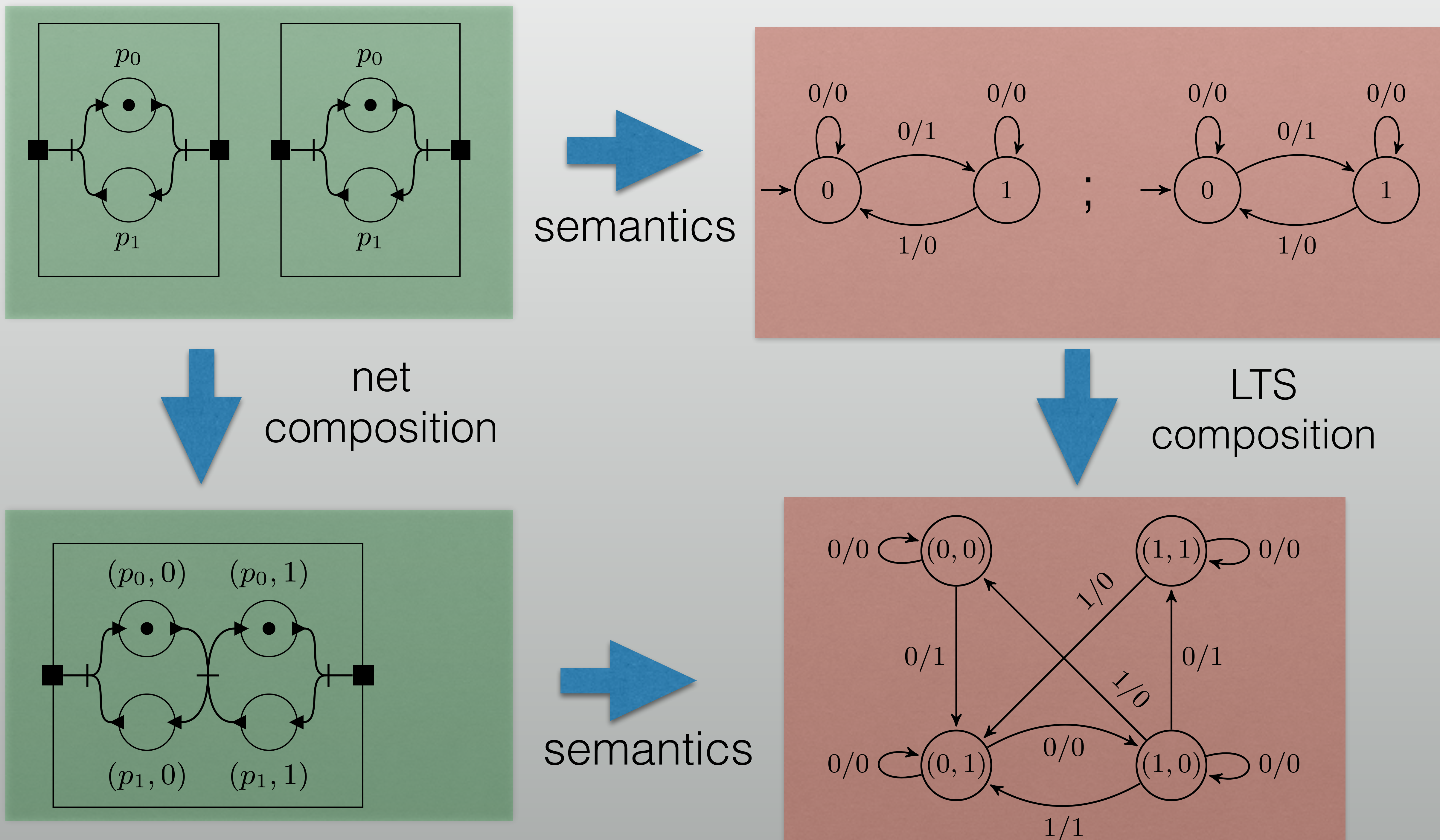
Reachability, compositionally

- NFA captures the **interaction patterns** which allow the subnet to reach the desired local marking

Any language equivalent NFA describes the same interaction patterns. e.g. a minimal NFA

This assignment is **functorial**
(the NFA of a composed net is the composition of component NFAs)

Compositionality as Functoriality



Main Theorem

- **Weak language equivalence is a congruence wrt to composition operations**
  - weak here means regarding internal moves (ie firing of transitions that are not connected to a boundary port) in a net as tau-moves or epsilon-moves
  - up-to-weak-language-equivalence means that we can discard irrelevant local state
  - in essence, we only care about how a component net *interacts*
  - Reachability reduces to language emptiness for nets with no boundaries!

Naive Algorithm

Naive Algorithm

- **The main theorem suggests the following algorithm for deciding reachability:**

Naive Algorithm

- **The main theorem suggests the following algorithm for deciding reachability:**
  - Write your Petri Net as a composition of subnets

Naive Algorithm

- **The main theorem suggests the following algorithm for deciding reachability:**
  - Write your Petri Net as a composition of subnets
  - Generate the NFA for each of these wrt their desired submarking

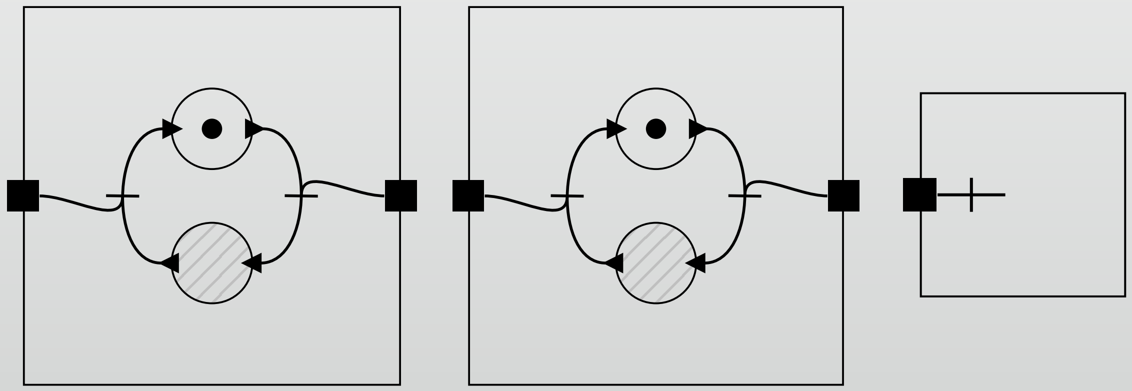
Naive Algorithm

- **The main theorem suggests the following algorithm for deciding reachability:**
  - Write your Petri Net as a composition of subnets
  - Generate the NFA for each of these wrt their desired submarking
  - Check their languages: if any of their languages is already empty then reachability fails

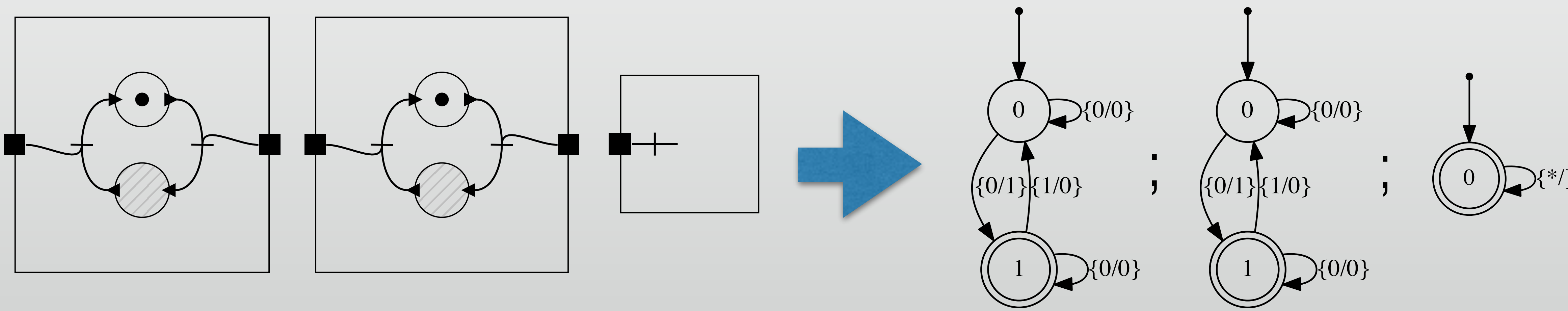
Naive Algorithm

- **The main theorem suggests the following algorithm for deciding reachability:**
  - Write your Petri Net as a composition of subnets
  - Generate the NFA for each of these wrt their desired submarking
  - Check their languages: if any of their languages is already empty then reachability fails
  - Compose the NFAs - and check whether their languages are empty

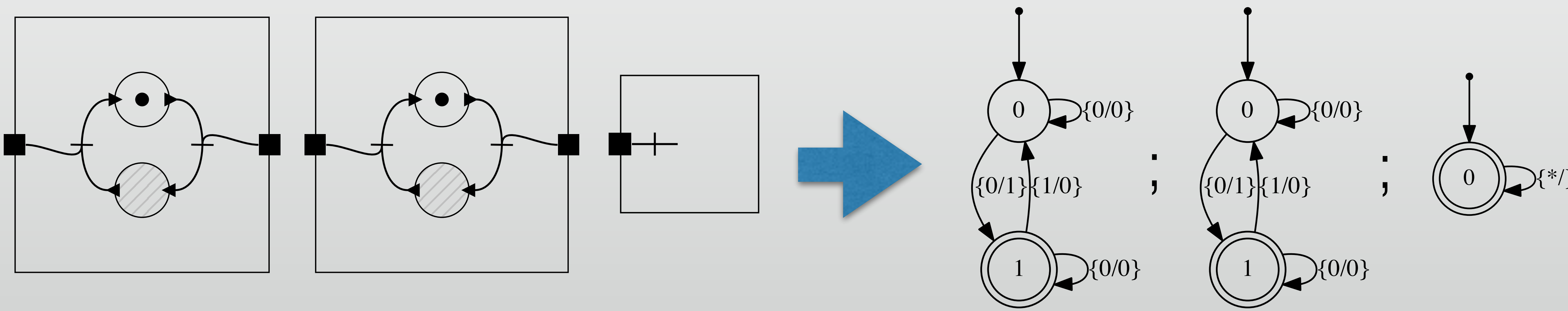
Example - buffer



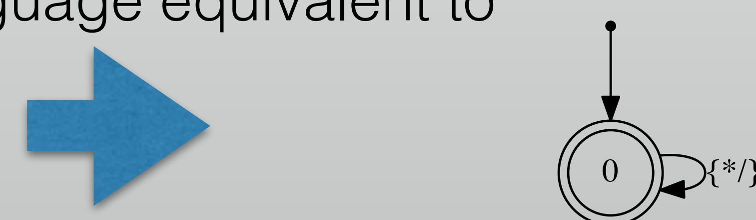
Example - buffer



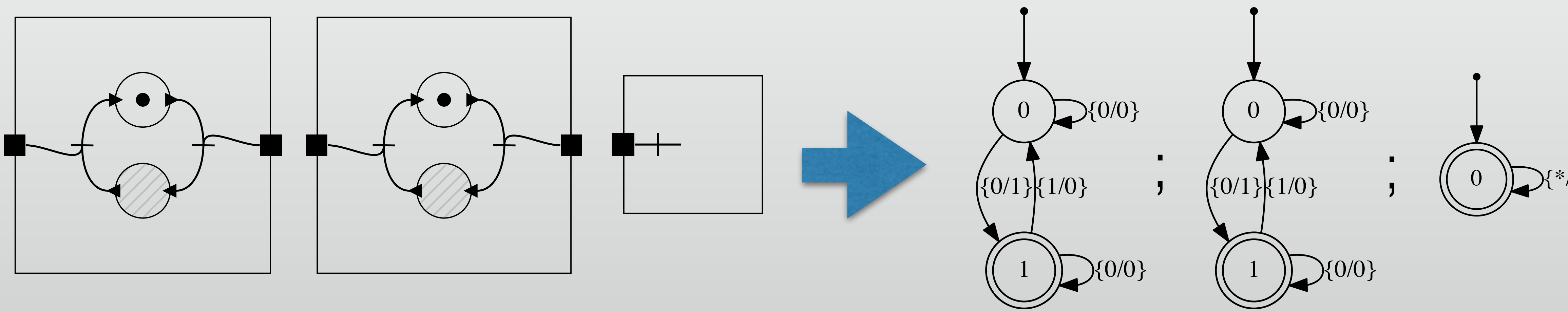
Example - buffer



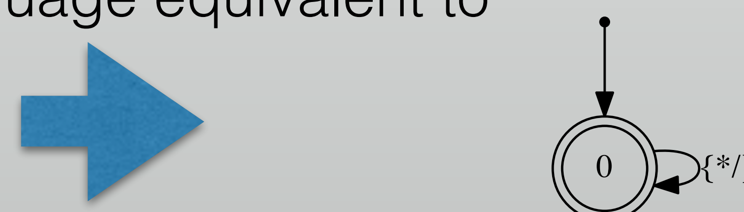
weak language equivalent to



Example - buffer



weak language equivalent to



The trivial accepting automaton is a fixed point of this process: this can also be seen as a proof of parametrised reachability for the buffer example!

Implementation details

- Penrose tool: implemented in Haskell, with almost no optimisation, but:
- We try to keep automata small

    R. Mayr and L. Clemente. *Advanced Automata Minimization*. In PoPL ’13.

- Memoisation is used to avoid re-minimising and re-composing weak language equivalent automata

    F. Bonchi and D. Pous. *Checking NFA Equivalence with Bisimulations up to Congruence*. In PoPL ’13.

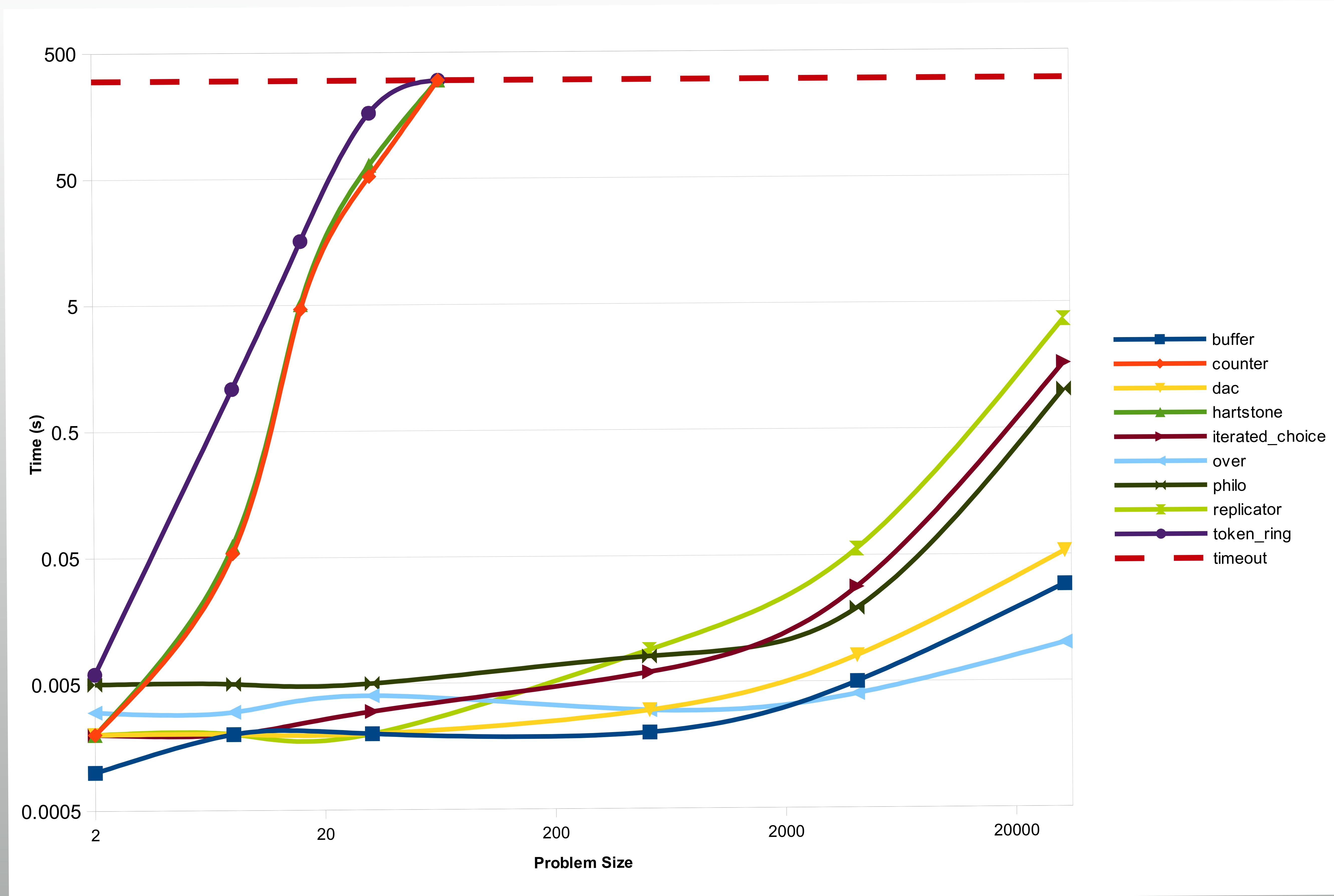
name	size	LOLA	CLP	CNA	Penrose
buffer	8	**0.001**	0.003	0.017	0.002
buffer	32	**0.001**	0.013	0.824	0.002
buffer	512	0.058	T	M	**0.002**
buffer	4096	T	T	M	**0.005**
buffer	32768	T	T	M	**0.029**

Performance on standard benchmarks

over	8	31.039	0.008	1.071	**0.003**	3812.00	37.63	141.85	**15.49**
over	32	*M*	*T*	*M*	**0.004**	*M*	*T*	*M*	**15.50**
over	512	*M*	*T*	*M*	**0.003**	*M*	*T*	*M*	**15.52**
over	4096	*M*	*T*	*M*	**0.004**	*M*	*T*	*M*	**16.04**
over	32768	*M*	*T*	*M*	**0.010**	*M*	*T*	*M*	**20.09**
dac	8	**0.001**	0.003	0.017	0.002	**7.51**	33.28	38.85	14.68
dac	32	**0.001**	0.005	0.028	0.002	**7.50**	34.50	49.45	14.68
dac	512	0.005	*T*	255.847	**0.003**	20.62	*T*	6012.00	**14.80**
dac	4096	2.462	*T*	*M*	**0.008**	166.07	*T*	*M*	**15.92**
dac	32768	*T*	*T*	*M*	**0.053**	*T*	*T*	*M*	**24.24**
philo	8	**0.002**	0.003	0.016	0.005	**8.86**	33.22	38.54	17.34
philo	32	*M*	**0.003**	0.017	0.005	*M*	33.53	40.87	**17.35**
philo	512	*M*	0.020	0.086	**0.008**	*M*	41.69	290.77	**17.39**
philo	4096	*M*	7.853	*M*	**0.019**	*M*	172.76	*M*	**17.58**
philo	32768	*M*	*T*	*M*	**1.014**	*M*	*T*	*M*	**21.32**
iter-choice∗	8	0.006	5.025	19.062	**0.002**	36.37	465.17	1570.64	**14.64**
iter-choice∗	32	*M*	*T*	*T*	**0.003**	*M*	*T*	*T*	**14.64**
iter-choice∗	512	*M*	*T*	*T*	**0.006**	*M*	*T*	*T*	**14.71**
iter-choice∗	4096	*M*	*T*	*T*	**0.028**	*M*	*T*	*T*	**15.22**
iter-choice∗	32768	*M*	*T*	*T*	**1.644**	*M*	*T*	*T*	**20.15**
replicator∗	8	**0.001**	/	0.016	0.002	**7.51**	/	38.15	14.72
replicator∗	32	**0.001**	/	0.017	0.002	**7.51**	/	39.41	14.72
replicator∗	512	**0.002**	/	1.023	0.009	**14.72**	/	77.87	14.82
replicator∗	4096	0.062	/	64.046	**0.056**	86.85	/	3256.00	**15.72**
replicator∗	32768	91.646	/	*M*	**3.660**	1524.50	/	*M*	**21.90**
counter∗	8	**0.001**	/	/	0.054	**7.51**	/	/	19.98
counter∗	16	**0.000**	/	/	4.646	**7.51**	/	/	27.98
counter∗	32	**0.001**	/	/	52.072	**7.51**	/	/	50.25
counter∗	64	**0.001**	/	/	*T*	**8.60**	/	/	*T*
hartstone	8	**0.001**	0.002	/	0.062	**7.51**	33.17	/	20.05
hartstone	16	**0.001**	0.003	/	5.073	**7.51**	33.20	/	24.01
hartstone	32	**0.001**	0.002	/	64.062	**7.51**	33.22	/	38.70
hartstone	64	**0.001**	0.002	/	*T*	**8.54**	33.46	/	*T*
token-ring	8	**0.001**	0.007	0.071	1.085	**7.51**	39.96	89.81	20.89
token-ring	16	**1.824**	*T*	*T*	16.038	318.08	*T*	*T*	**29.41**
token-ring	32	*M*	*T*	*T*	**165.461**	*M*	*T*	*T*	**50.19**
token-ring	64	*M*	*T*	*T*	*T*	*M*	*T*	*T*	*T*

Caveats

- Our tool takes in an algebraic decomposition as input
  - some nets do not allow efficient decompositions because of graph theoretic complexity (high rank width of the underlying hypergraph)


500
50
5
0.5
0.05
0.005
0.0005
Time (s)
2
20
200
2000
20000
Problem Size
buffer
counter
dac
hartstone
iterated_choice
over
philo
replicator
token_ring
timeout

Caveats

- Our tool takes in an algebraic decomposition as input
  - some nets do not allow efficient decompositions because of graph theoretic complexity (high rank width of the underlying hypergraph)
  - even if a net has small rank-width, efficient decompositions may not exist for semantic reasons
  - deriving efficient decompositions automatically is highly non-trivial
  - even after choosing a graph decomposition, the *syntactic* description is important e.g. **associativity matters!**
- But high-level system descriptions are the norm in **real** systems: e.g. decompositions have followed Corbett's high-level Ada descriptions very closely

Conclusions

- Divide and conquer for reachability in 1-bounded nets
  - on many realistic examples, this approach vastly outperforms traditional global approaches
- Speculation and future work
  - examples on which we perform less well can sometimes be determined statically (e.g. by looking at the graph theoretical complexity of the underlying net!)
  - can compositionality help us to understand reachability in the infinite state case?