A False History of True Concurrency

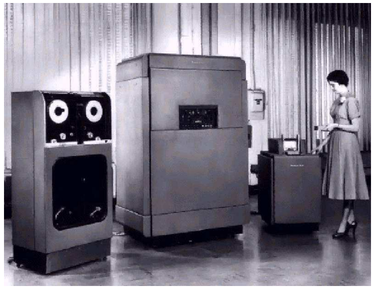
Javier Esparza

Sofware Reliability and Security Group

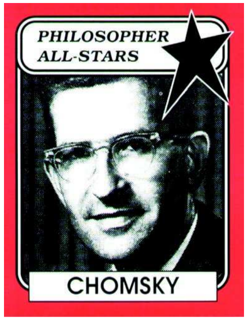
Institute for Formal Methods in Computer Science
University of Stuttgart

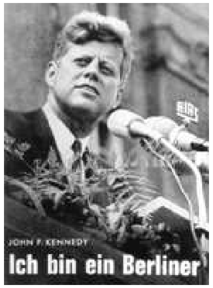
The early 60s

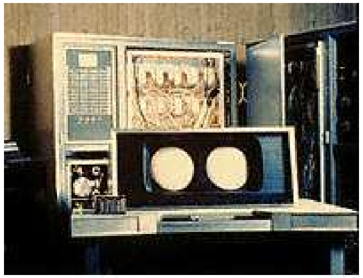










Abstract Models of Computation in the early 60s

Lambda calculus (Church 35)

Turing machines (Turing 36)

Finite automata (Kleene 56, Moore 56, Mealy 56, Scott and Rabin 59)

Pushdown automata (Oettinger 61, Chomsky 62, Evey 63, Schutzenberger 63)

Semantics: executions

States: current configurations of the machine

One or more initial states

Possibly some distinguished final states

Transitions: moves between configurations

Lambda calculus	$(\lambda x.xx)(\lambda y.y)$	$\longrightarrow$	$(\lambda y.y)(\lambda z.z)$
Turing machine	$0010q_1011$	$\longrightarrow$	$001q_201011$
Finite automaton	q_1	$\xrightarrow{a}$	q_2
Pushdown automaton	$(q_1, XYYZ)$	$\xrightarrow{a}$	$(q_2, XYXYYZ)$

Executions: alternating sequences of states and transitions

Physics and Computation

Abstract machines are implemented as physical systems

Physics and Computation

Abstract machines	are implemented as can simulate	physical systems

SIMULA project (Nygaard and Dahl) started in 1962

Physics and Computation

Abstract machines	are implemented as can simulate	physical systems

SIMULA project (Nygaard and Dahl) started in 1962

A plane (physical system)

Physics and Computation

Abstract machines	are implemented as can simulate	physical systems

SIMULA project (Nygaard and Dahl) started in 1962

A plane (physical system)

can be simulated by a plane simulator (abstract machine)

Physics and Computation

Abstract machines	are implemented as can simulate	physical systems

SIMULA project (Nygaard and Dahl) started in 1962

A plane (physical system)

can be simulated by a plane simulator (abstract machine)

which can be implemented in a video console (physical system)

Physics and Computation

Abstract machines	are implemented as can simulate	physical systems

SIMULA project (Nygaard and Dahl) started in 1962

A plane (physical system)

can be simulated by a plane simulator (abstract machine)

which can be implemented in a video console (physical system)

which can be simulated by a hardware simulator (abstract machine)

Physics and Computation

Abstract machines	are implemented as can simulate	physical systems

SIMULA project (Nygaard and Dahl) started in 1962

A plane (physical system)

can be simulated by a plane simulator (abstract machine)

which can be implemented in a video console (physical system)

which can be simulated by a hardware simulator (abstract machine)

which is implemented in a PC (physical system) . . .

Petri's question

C.A. Petri points out a discrepancy between how Theoretical Physics and Theoretical Computer Science described systems in 1962:

Theoretical Physics describes systems as a collection of interacting particles (subsystems), without a notion of global clock or simultaneity

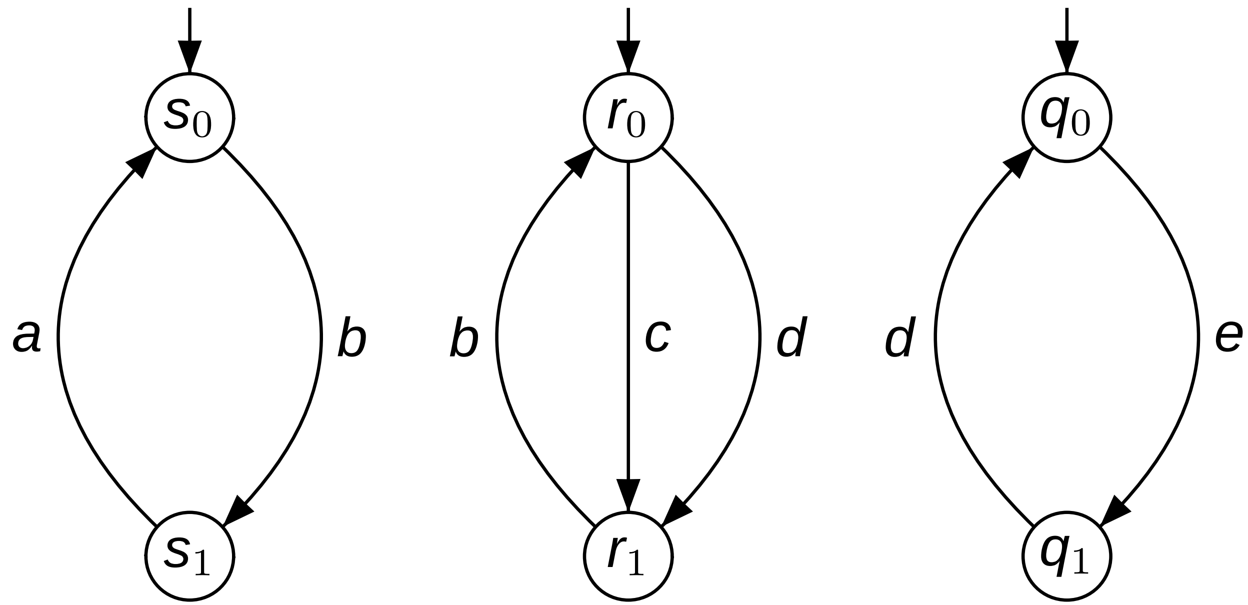
Theoretical Computer Science describes systems as sequential virtual machines going through a temporally ordered sequence of global states

Petri's question:

Which kind of abstract machine should be used to describe the physical implementation of a Turing machine?

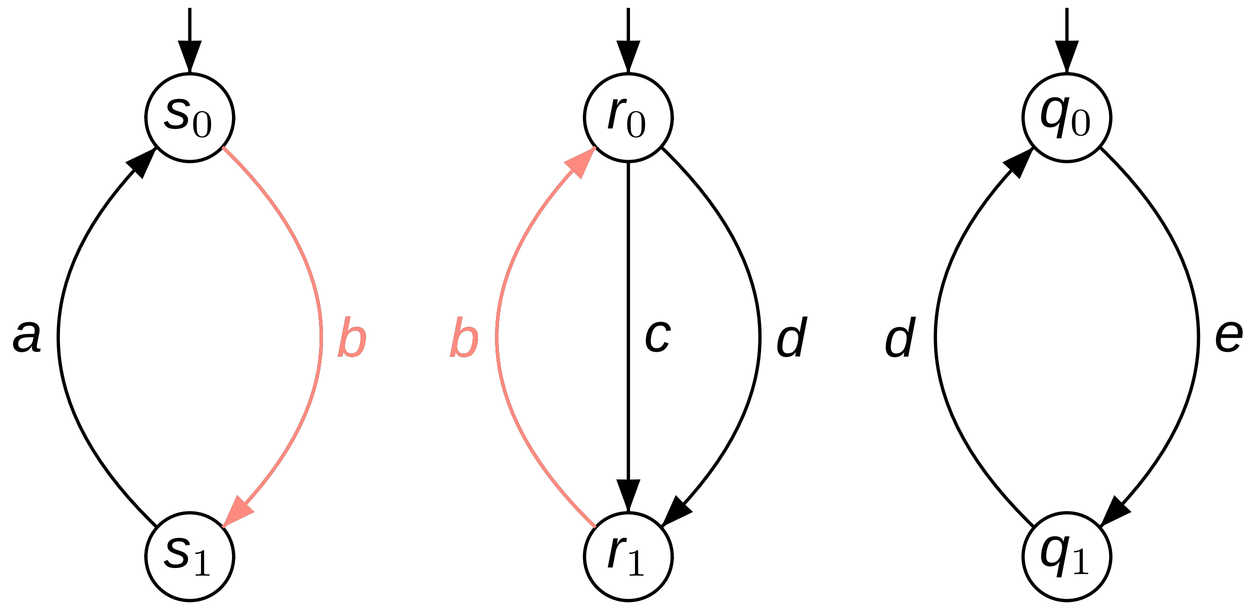
Petri Nets

A graphical representation of interacting finite automata:



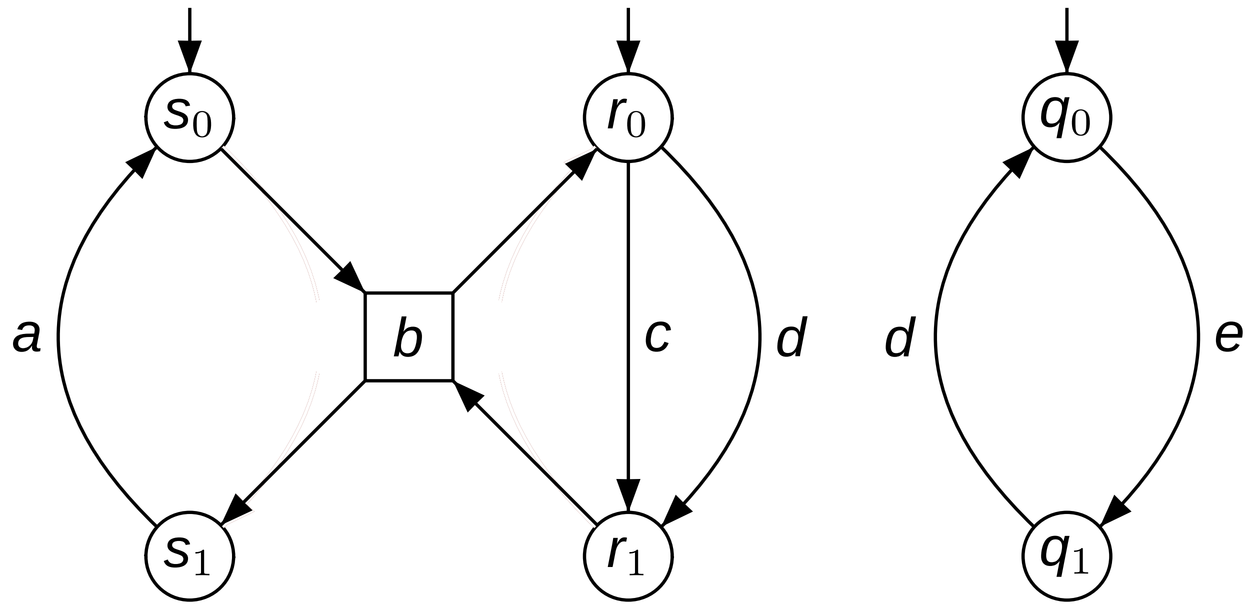
Petri Nets

A graphical representation of interacting finite automata:



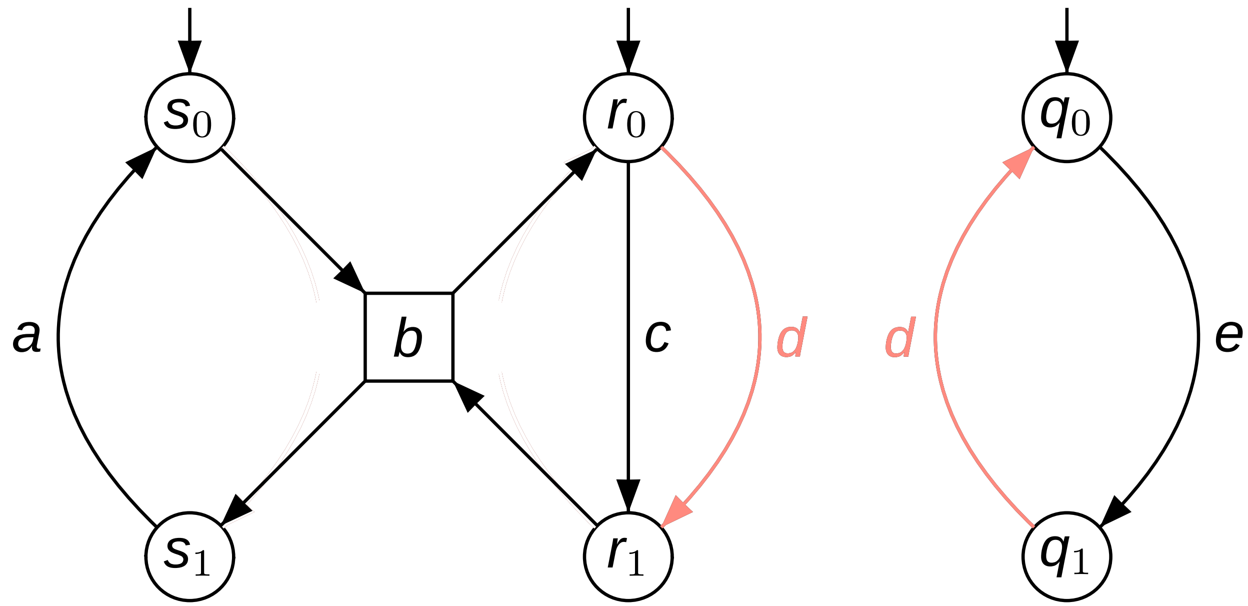
Petri Nets

A graphical representation of interacting finite automata:



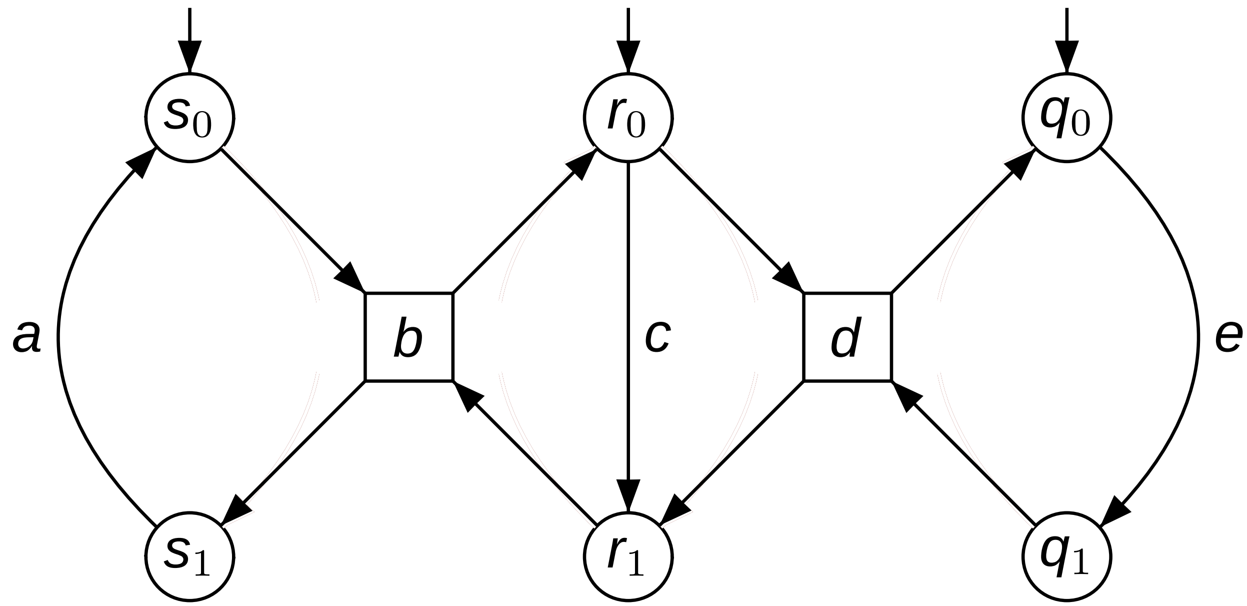
Petri Nets

A graphical representation of interacting finite automata:



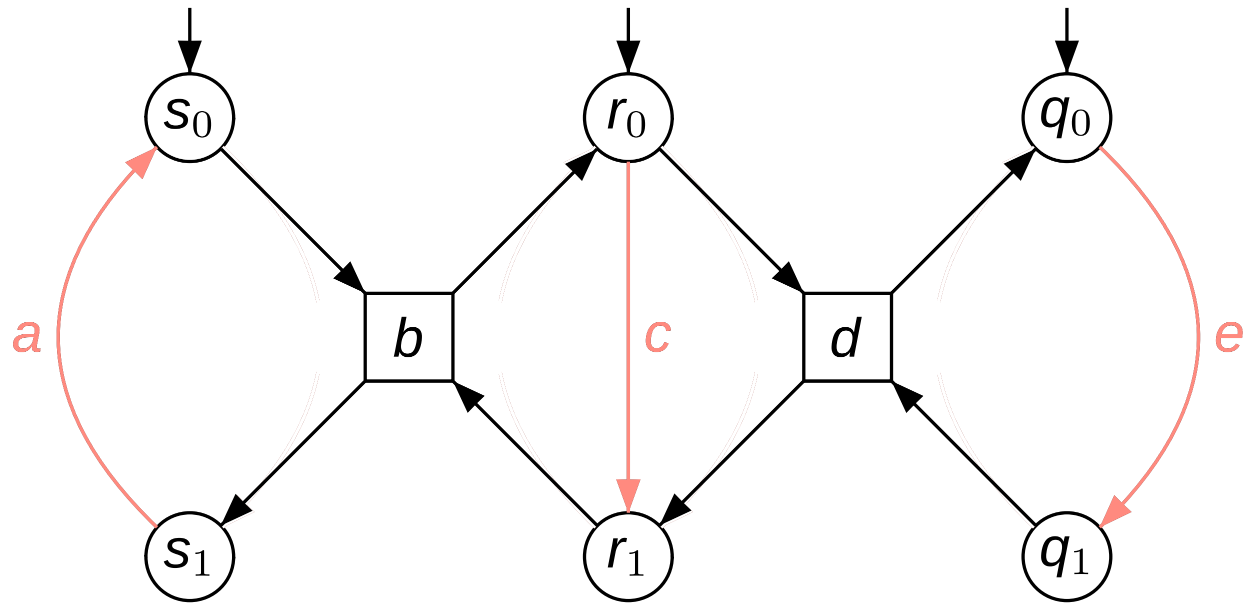
Petri Nets

A graphical representation of interacting finite automata:



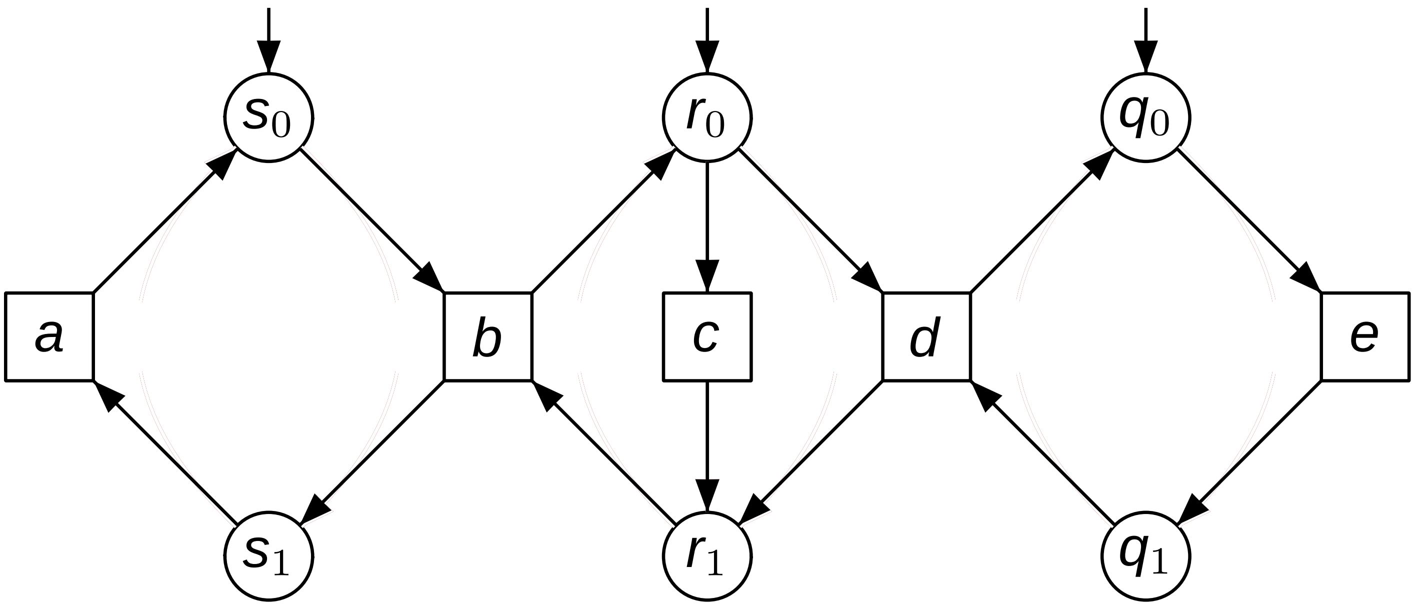
Petri Nets

A graphical representation of interacting finite automata:



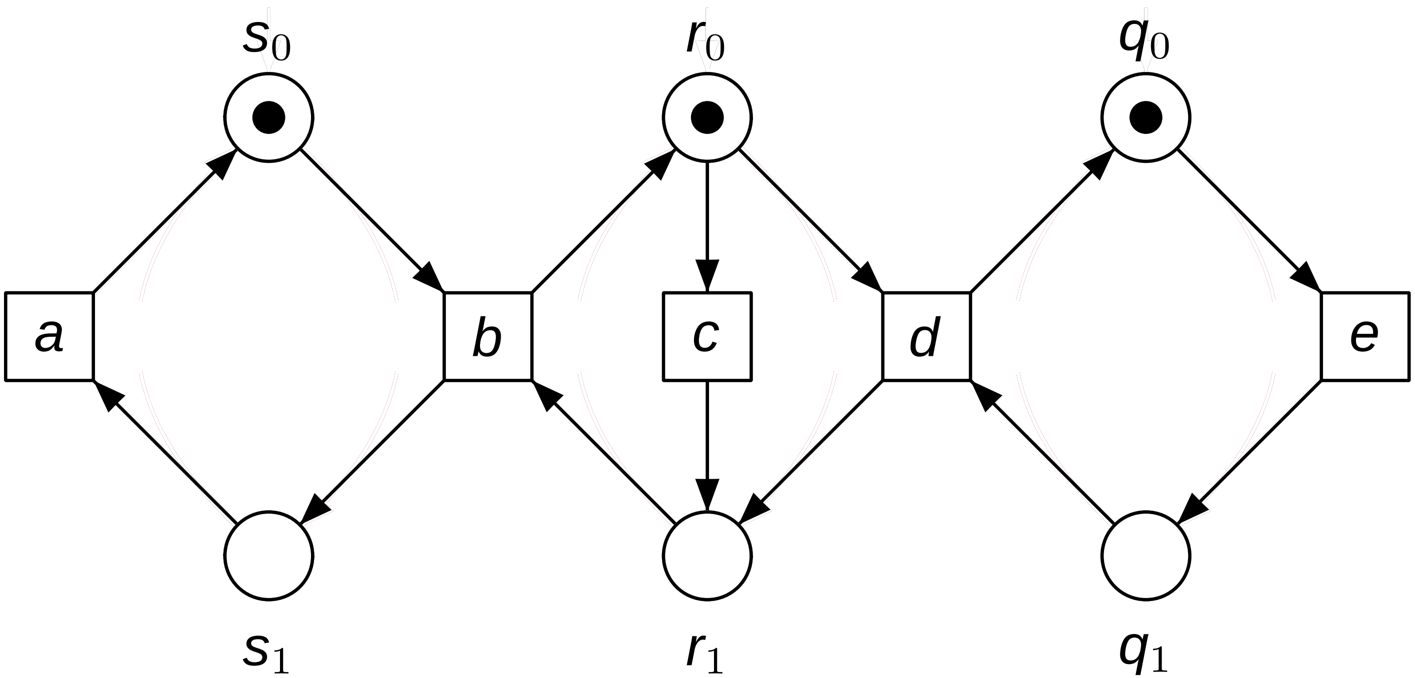
Petri Nets

A graphical representation of interacting finite automata:



Petri Nets

A graphical representation of interacting finite automata:



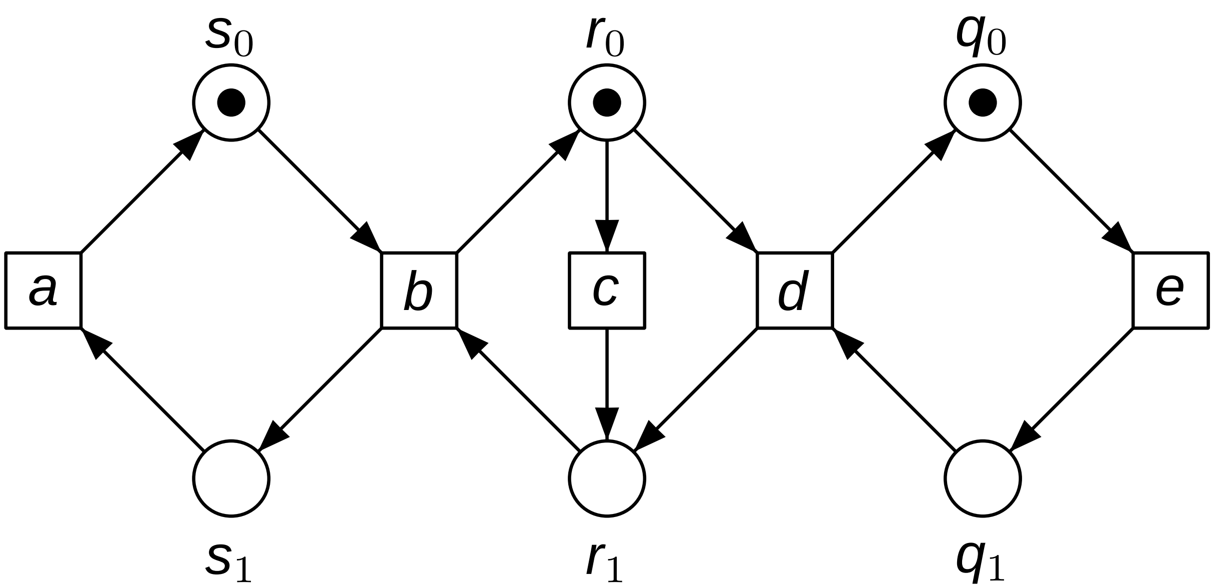
The interleaving semantics of Petri nets

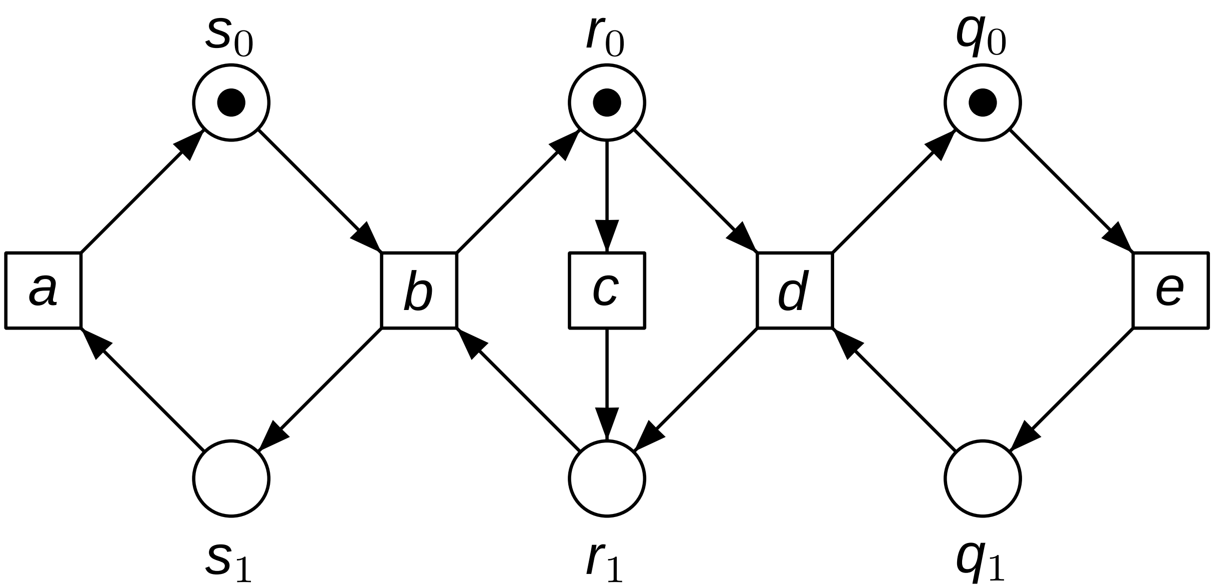
An execution semantics

State: marking (distribution of tokens)

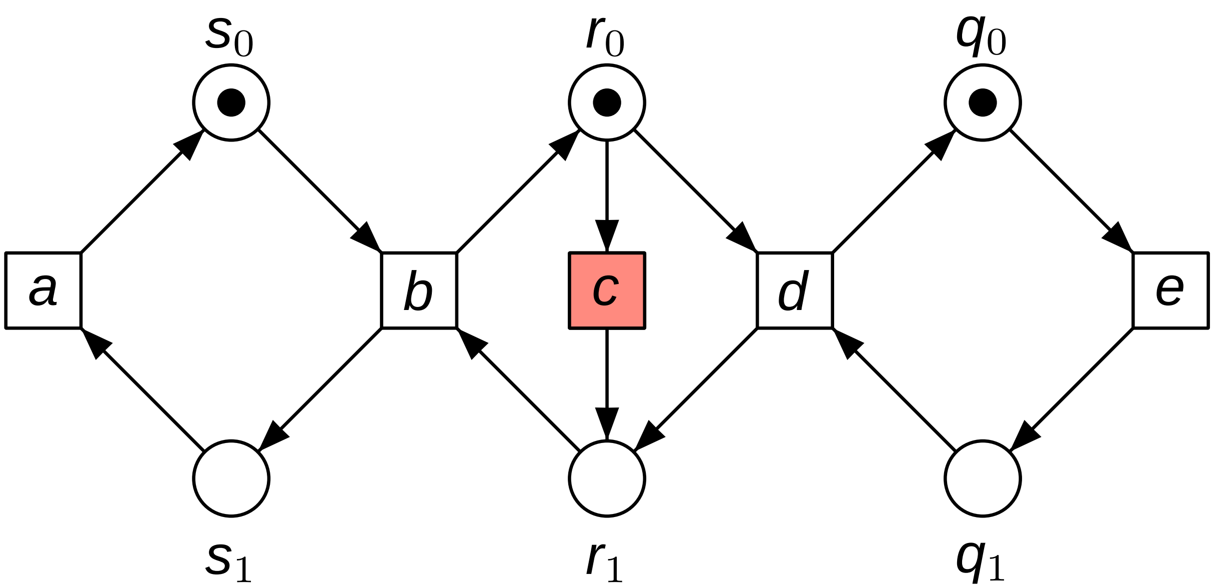
Transitions: $M \xrightarrow{a} M'$

Executions: $M_0 \xrightarrow{a_0} M_1 \xrightarrow{a_1} M_2 \ldots$

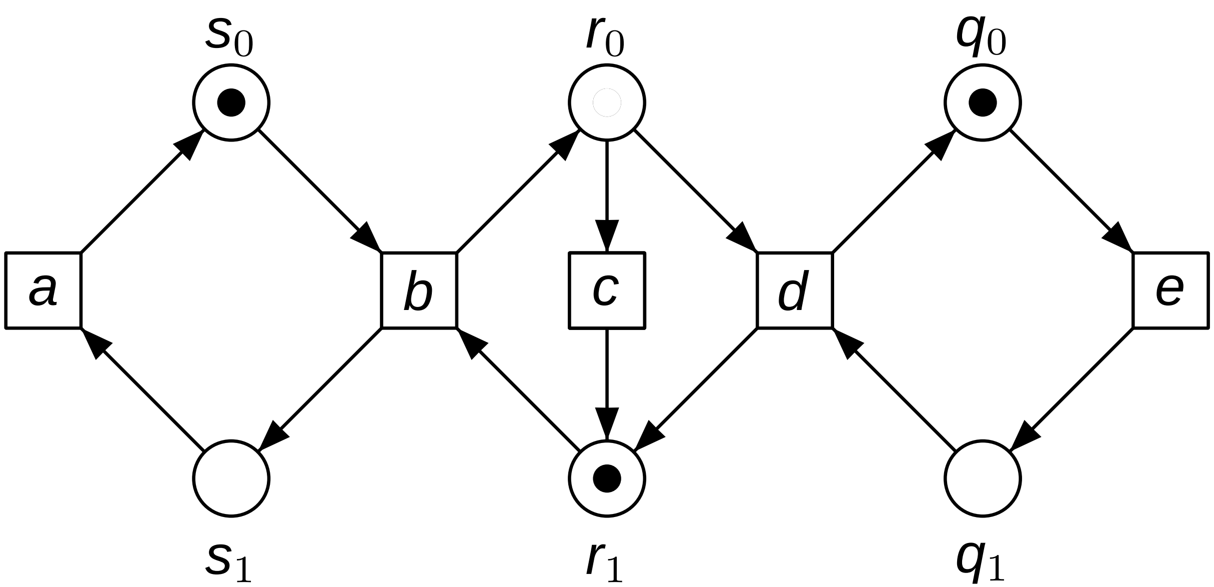

s_0
r_0
q_0
a
b
c
d
e
s_1
r_1
q_1



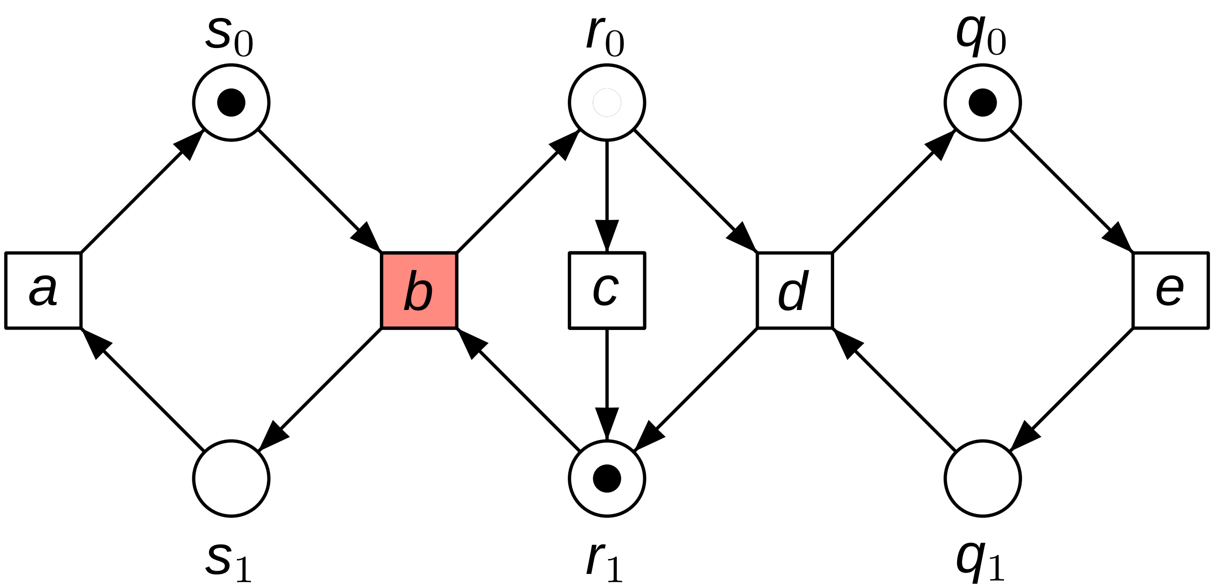
$$\begin{matrix} s \\ r \\ q \end{matrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$



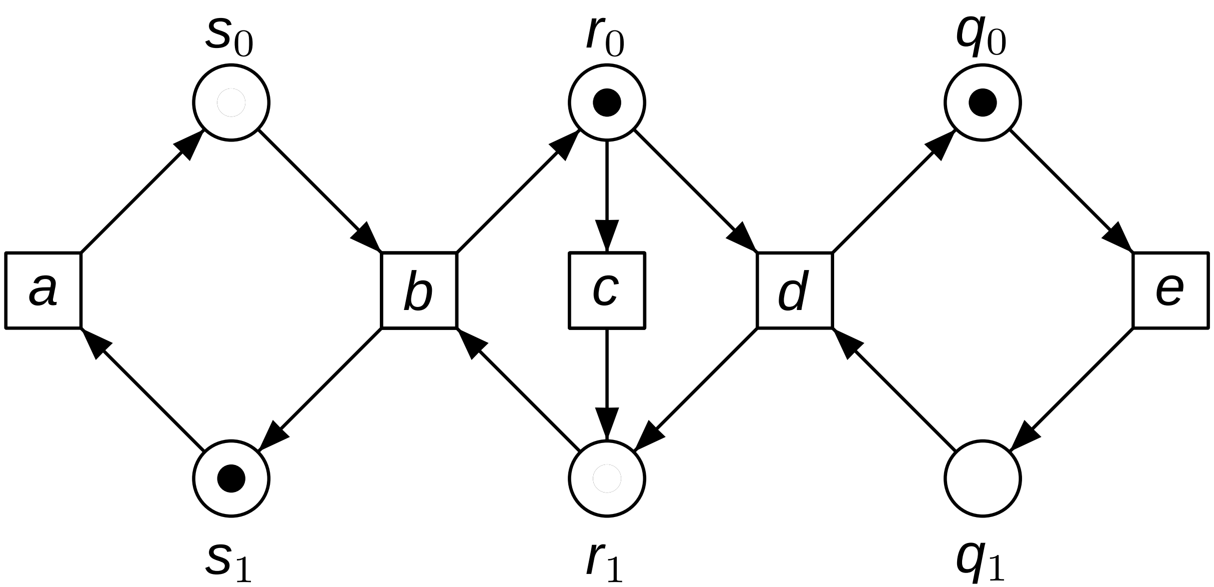
$$\begin{matrix} s \\ r \\ q \end{matrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$



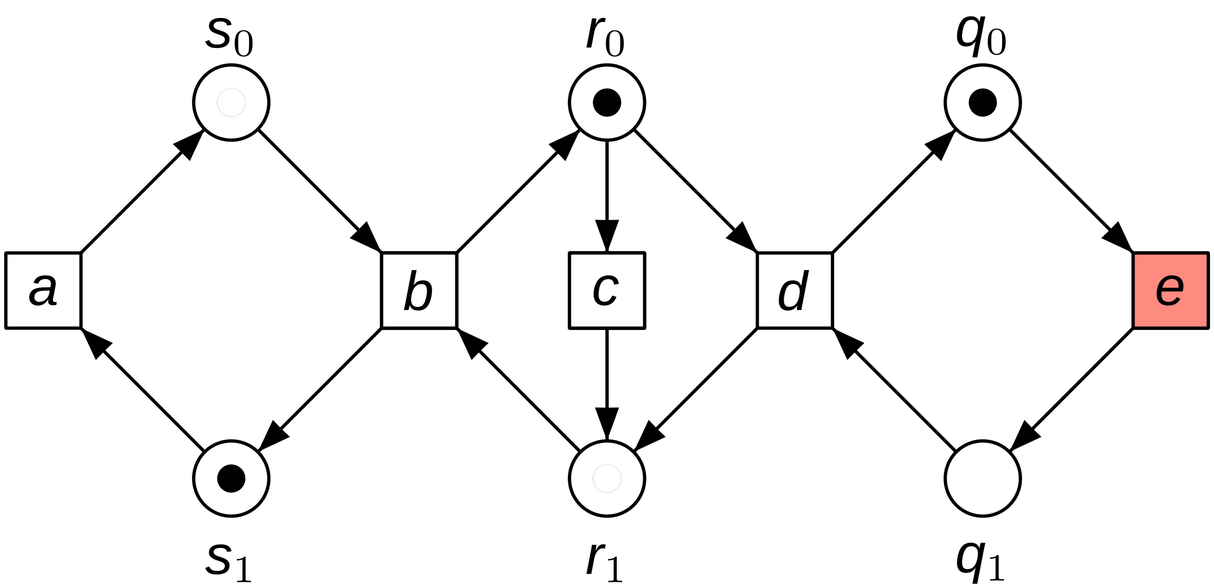
$$\begin{matrix} s \\ r \\ q \end{matrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{c} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$



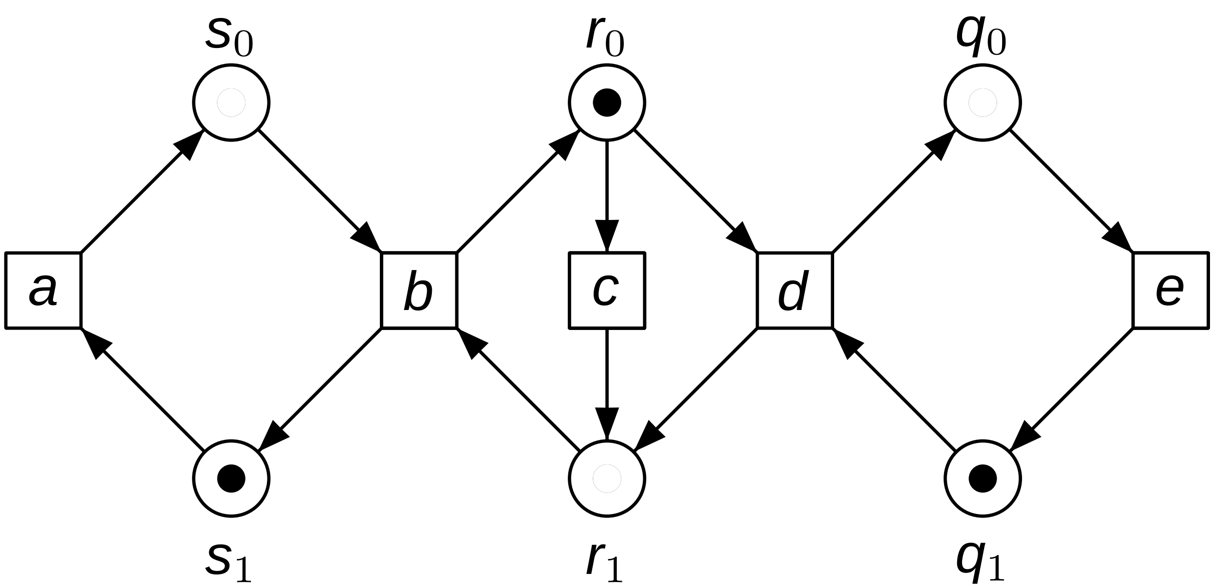
$$\begin{matrix} s \\ r \\ q \end{matrix} \quad \begin{bmatrix}0\\0\\0\end{bmatrix} \xrightarrow{c} \begin{bmatrix}0\\1\\0\end{bmatrix}$$



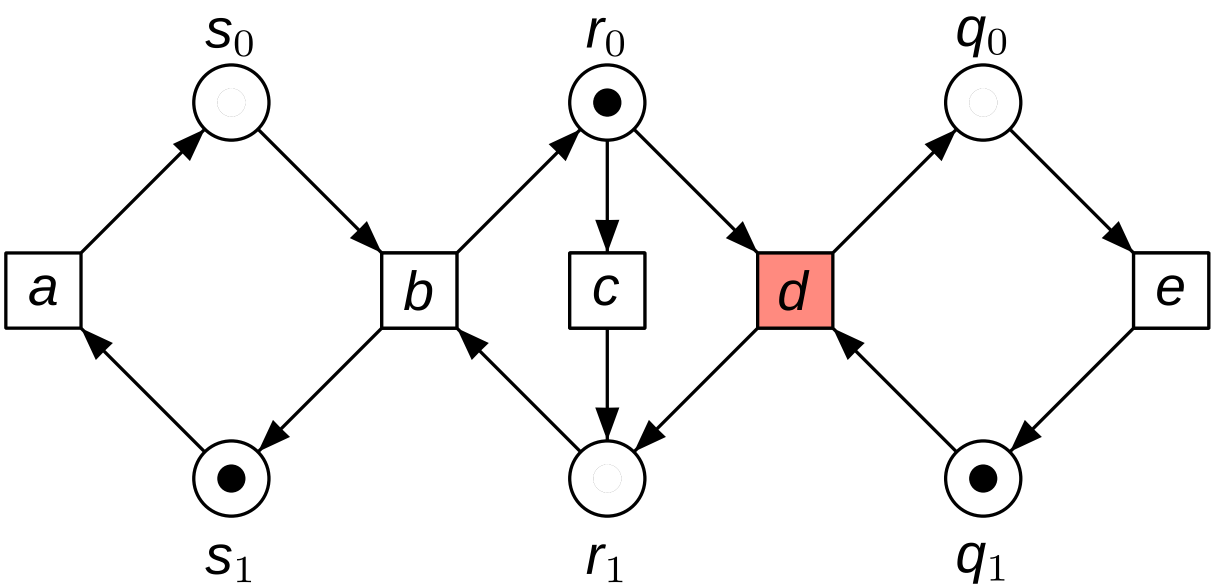
$$\begin{matrix} s \\ r \\ q \end{matrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{c} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{b} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$



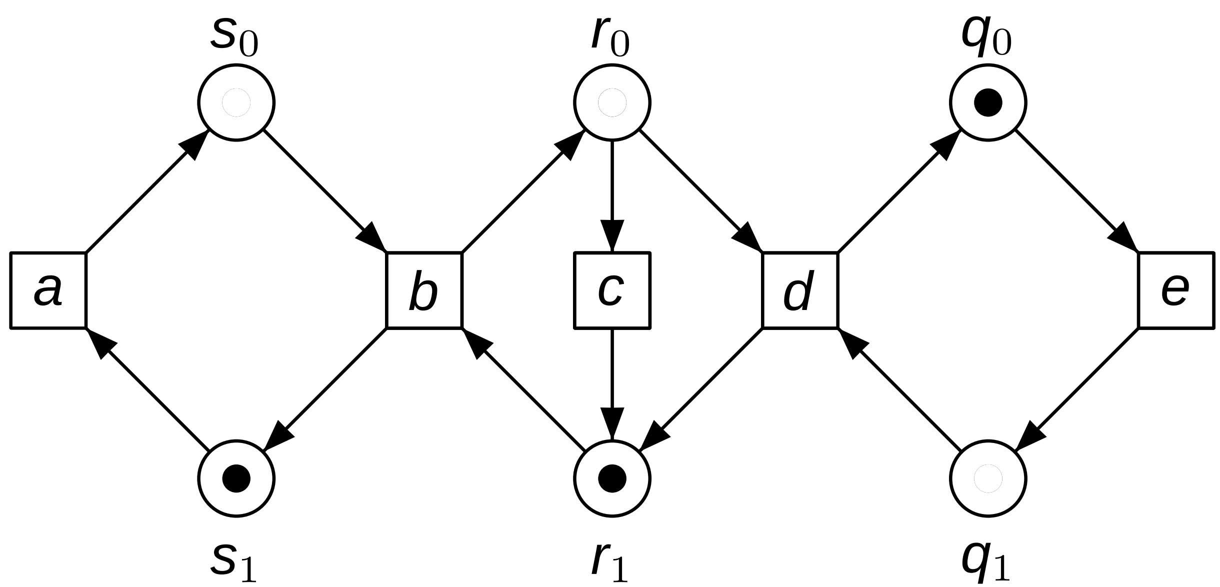
$$\begin{matrix} s \\ r \\ q \end{matrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{c} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{b} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$



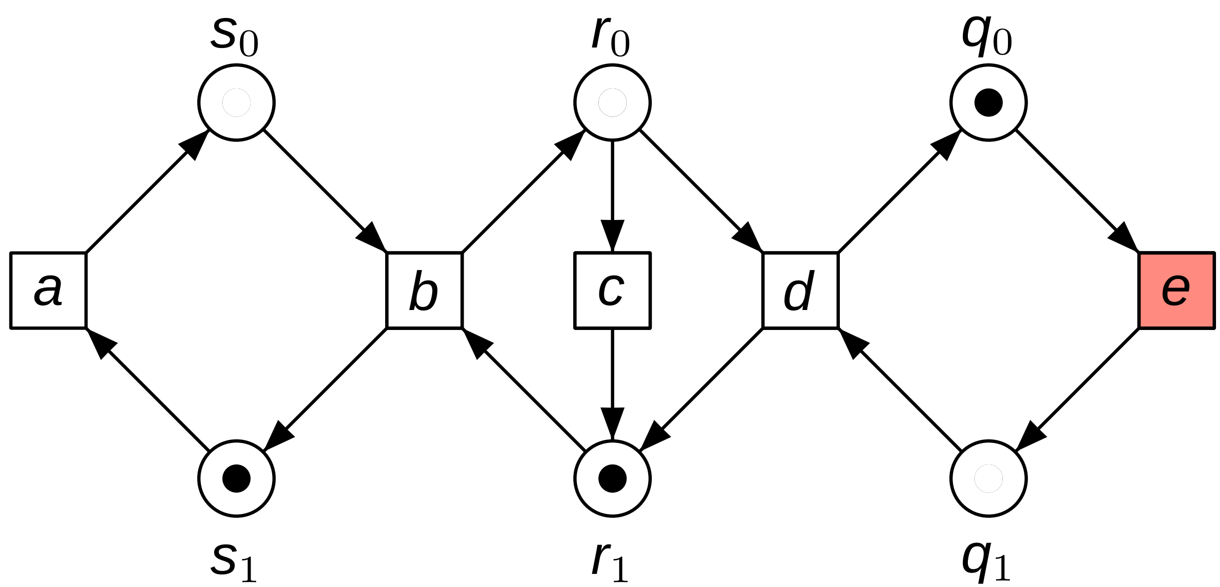
$$\begin{matrix} s \\ r \\ q \end{matrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{c} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{b} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{e} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$



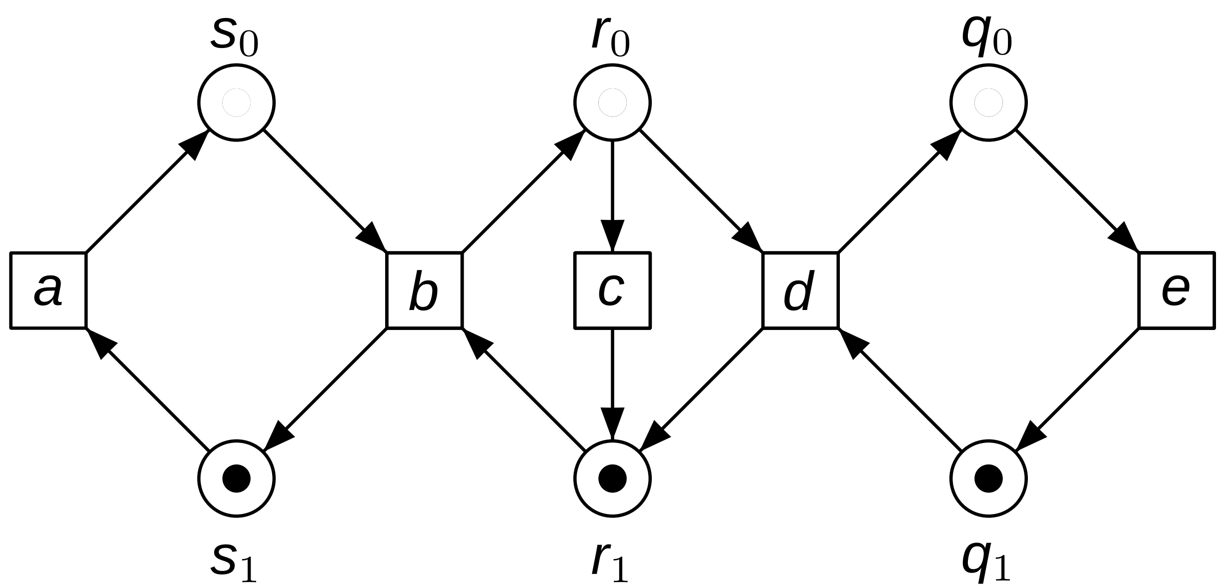
$$\begin{matrix} s \\ r \\ q \end{matrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{c} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{b} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{e} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$



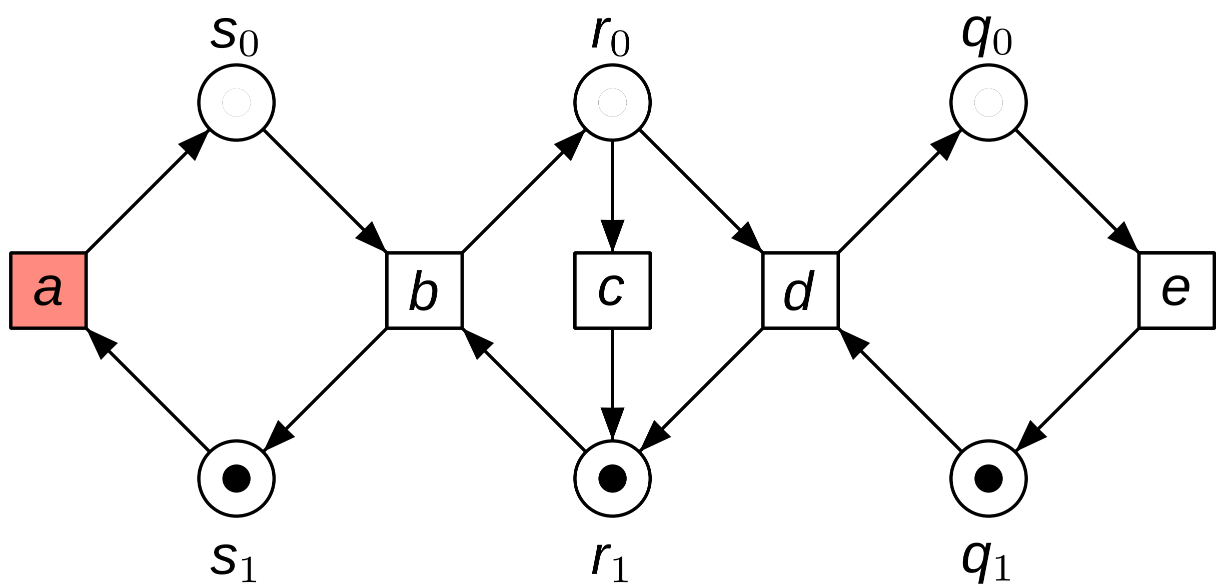
$$\begin{matrix} s \\ r \\ q \end{matrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{c} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{b} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{e} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \xrightarrow{d} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$



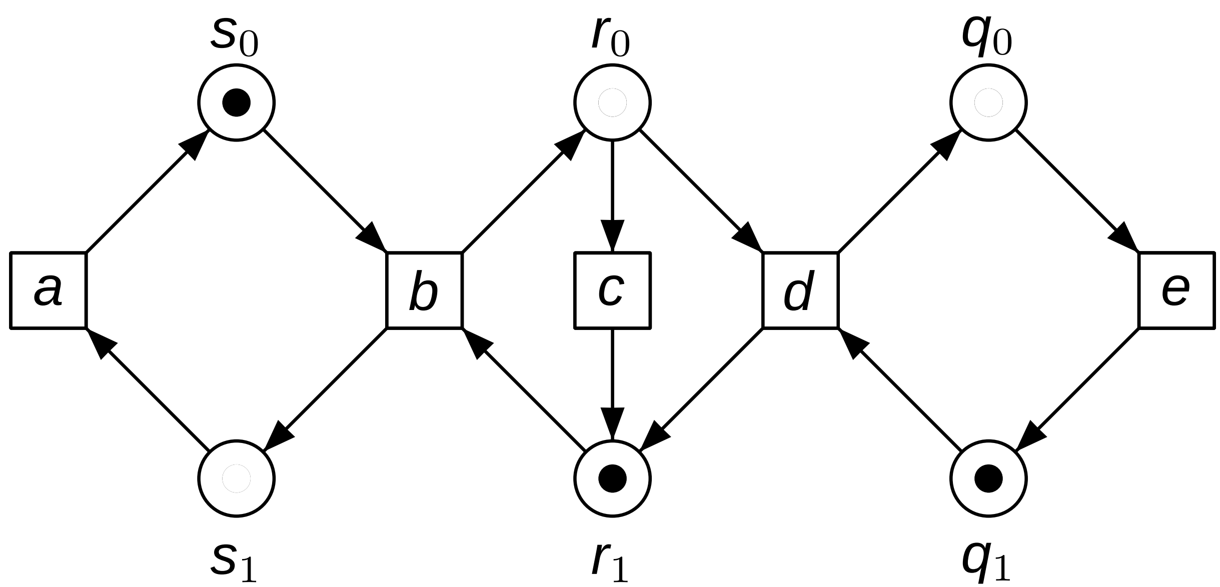
$$\begin{matrix} s \\ r \\ q \end{matrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{c} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{b} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{e} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \xrightarrow{d} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$



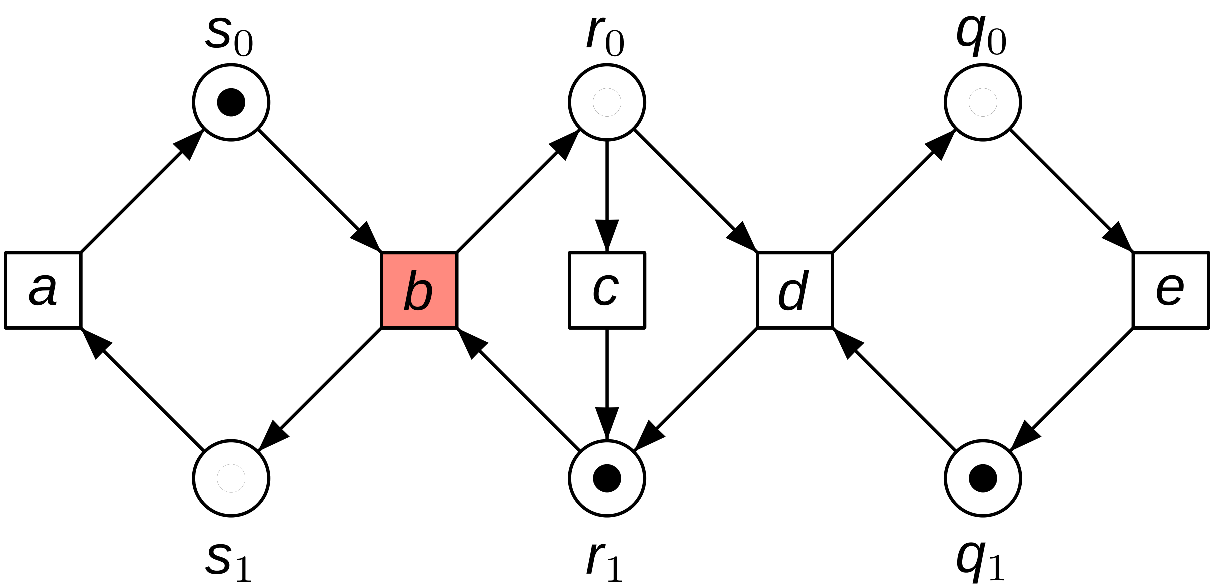
$$\begin{matrix} s \\ r \\ q \end{matrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{c} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{b} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{e} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \xrightarrow{d} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{e} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$



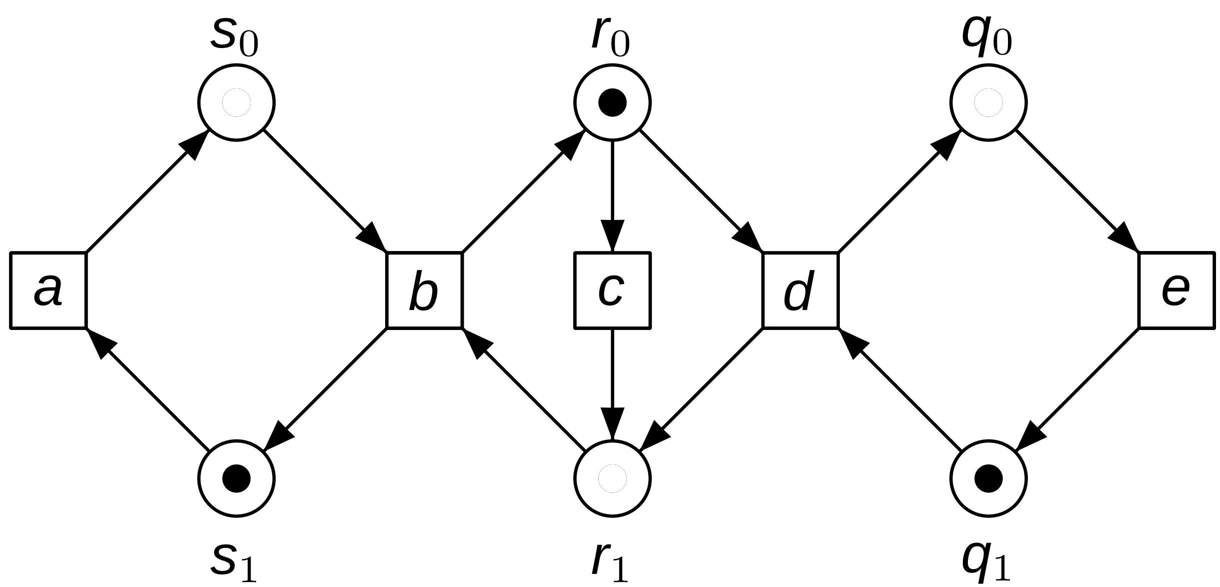
$$\begin{matrix} s \\ r \\ q \end{matrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{c} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{b} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{e} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \xrightarrow{d} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{e} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$



$$\begin{matrix} s \\ r \\ q \end{matrix} \quad \begin{bmatrix}0\\0\\0\end{bmatrix} \xrightarrow{c} \begin{bmatrix}0\\1\\0\end{bmatrix} \xrightarrow{b} \begin{bmatrix}1\\0\\0\end{bmatrix} \xrightarrow{e} \begin{bmatrix}1\\0\\1\end{bmatrix} \xrightarrow{d} \begin{bmatrix}1\\1\\0\end{bmatrix} \xrightarrow{e} \begin{bmatrix}1\\1\\1\end{bmatrix} \xrightarrow{a} \begin{bmatrix}0\\1\\1\end{bmatrix}$$

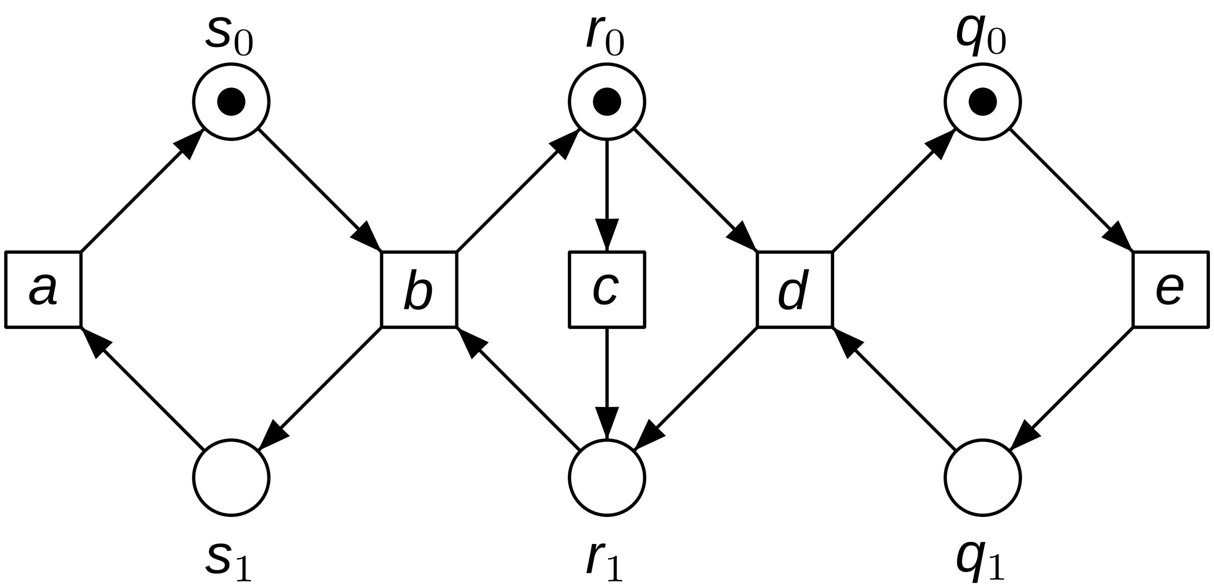


$$\begin{matrix} s \\ r \\ q \end{matrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{c} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{b} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{e} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \xrightarrow{d} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{e} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \xrightarrow{a} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

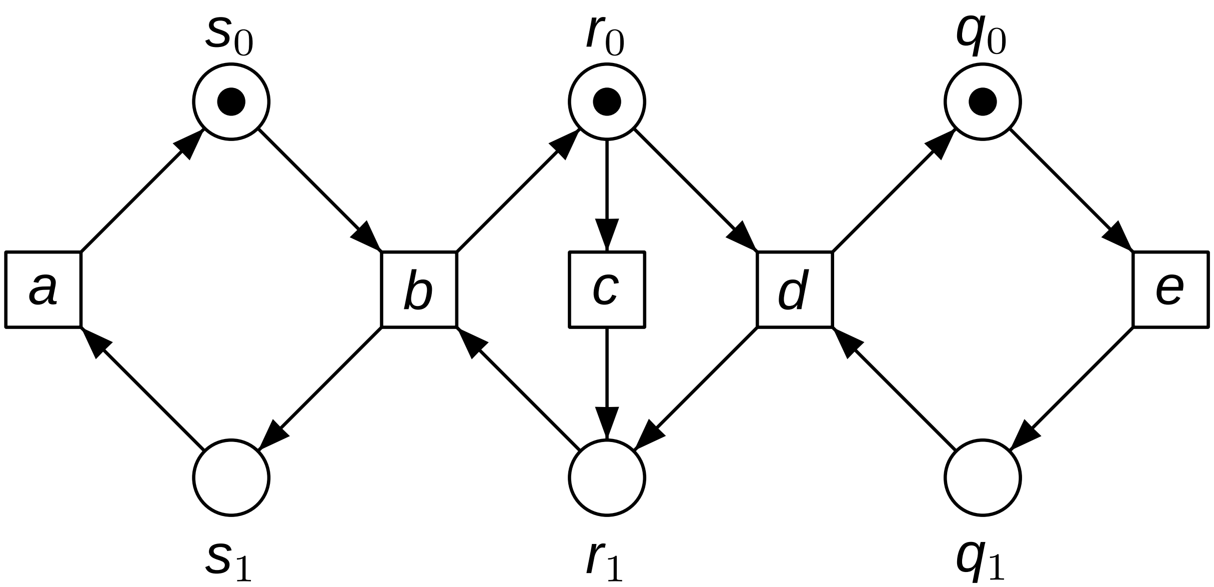


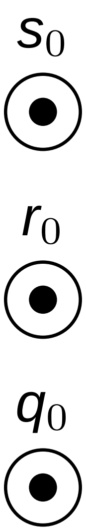
$$
\begin{matrix} s \\ r \\ q \end{matrix}
\quad
\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}
\xrightarrow{c}
\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}
\xrightarrow{b}
\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}
\xrightarrow{e}
\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}
\xrightarrow{d}
\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}
\xrightarrow{e}
\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}
\xrightarrow{a}
\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}
\xrightarrow{b}
\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}
$$

The true concurrency semantics of Petri nets

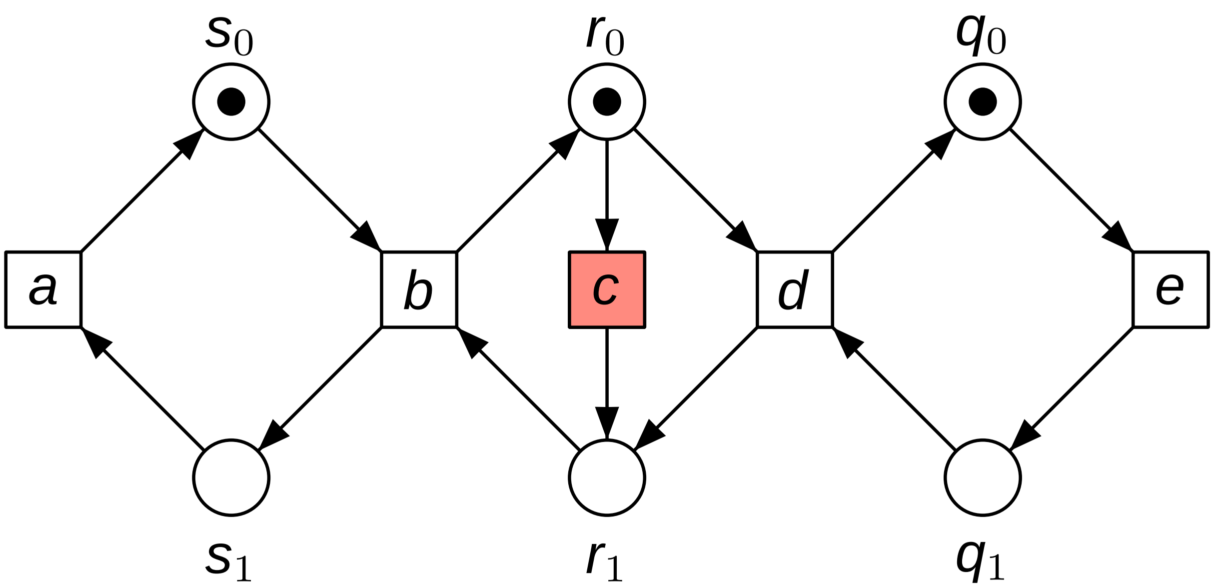


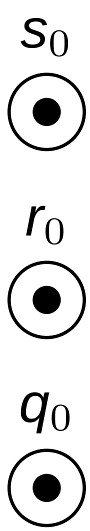
The true concurrency semantics of Petri nets



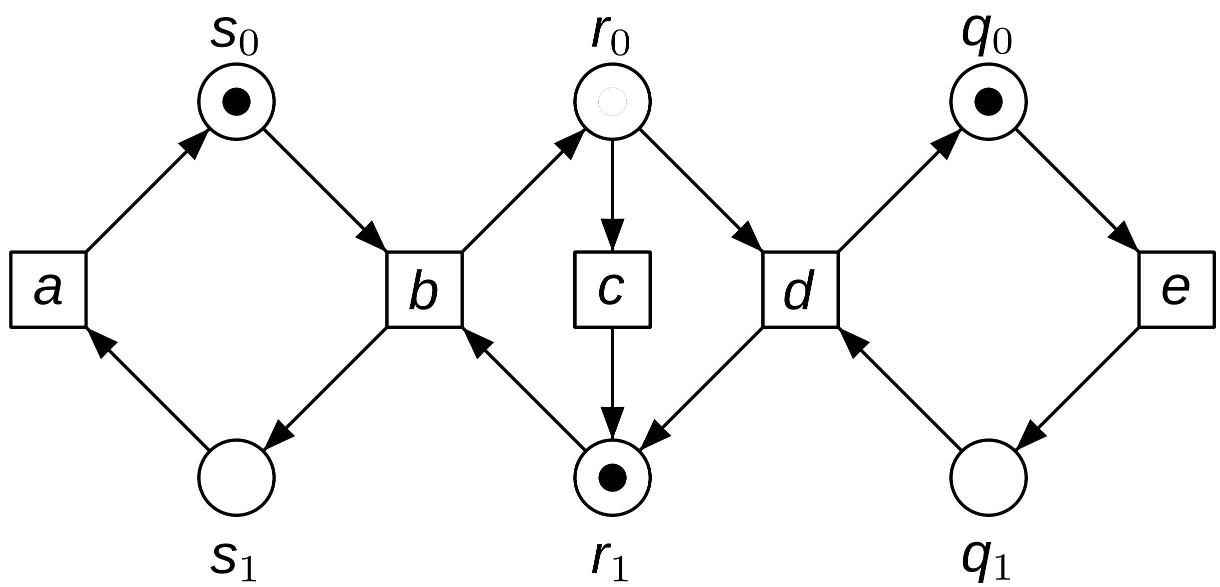


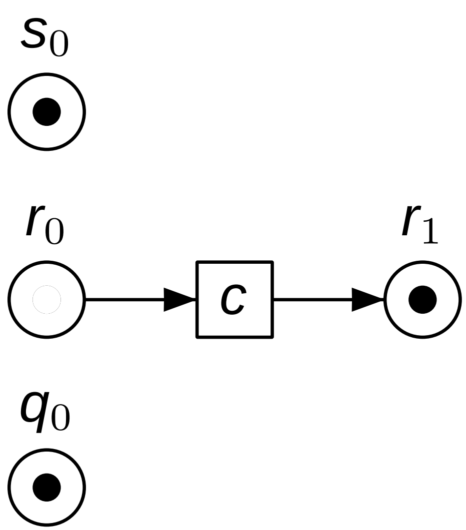
The true concurrency semantics of Petri nets



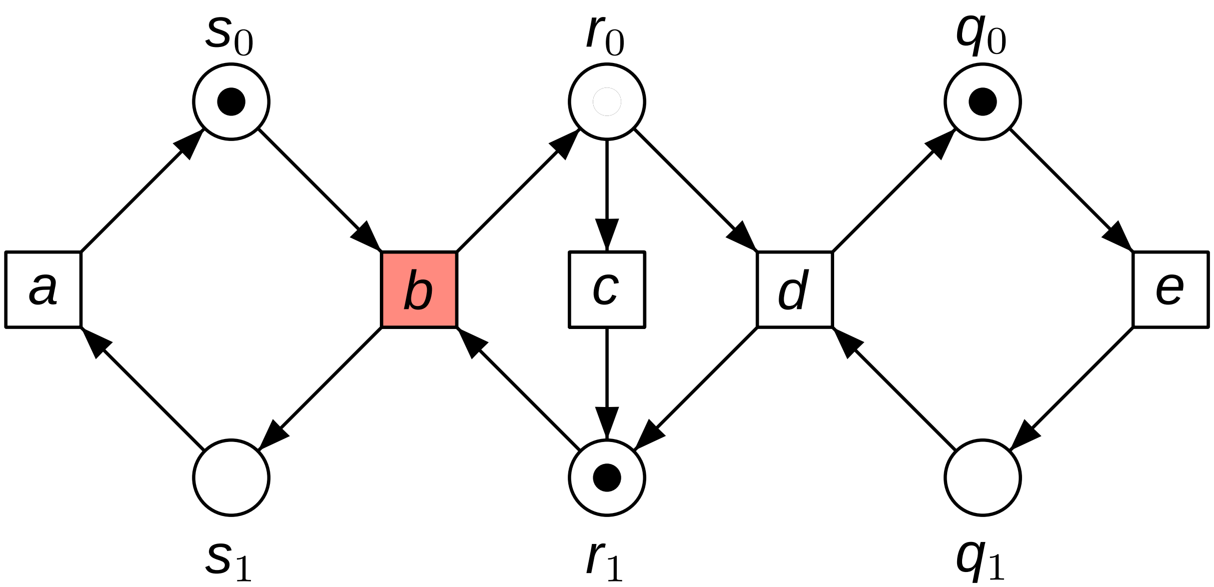


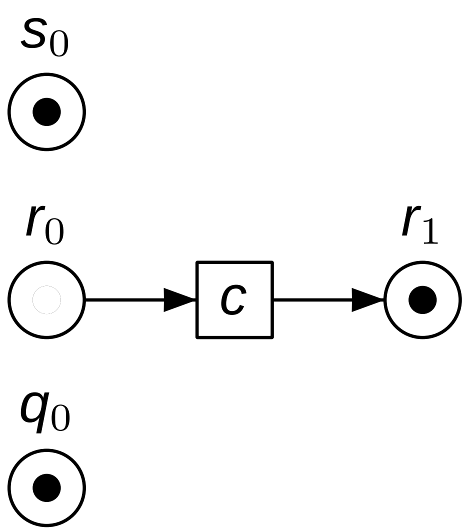
The true concurrency semantics of Petri nets



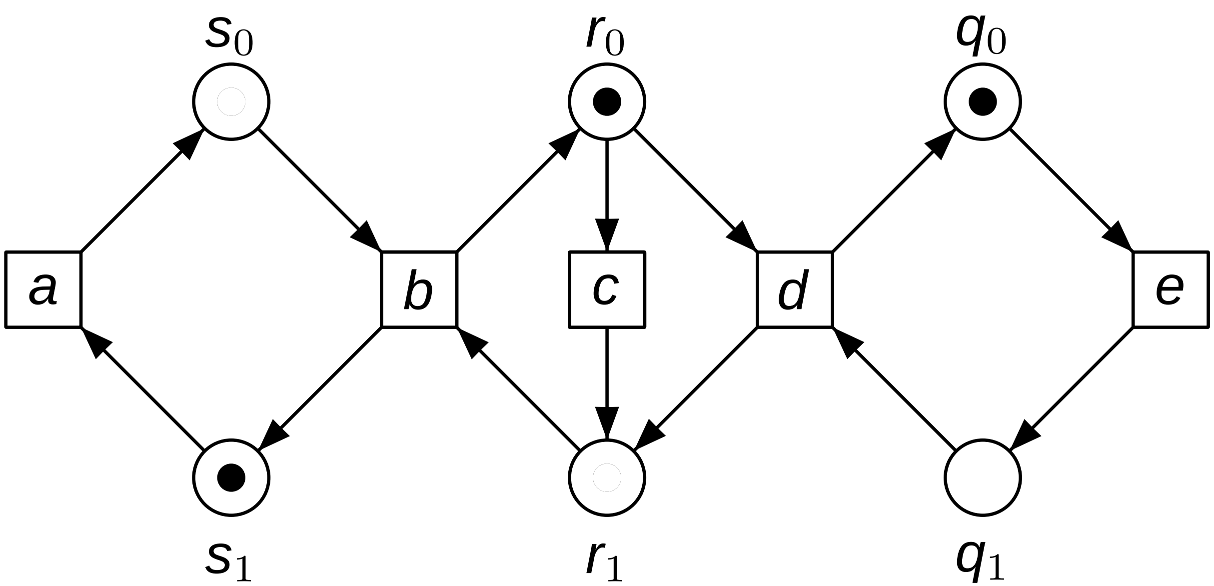


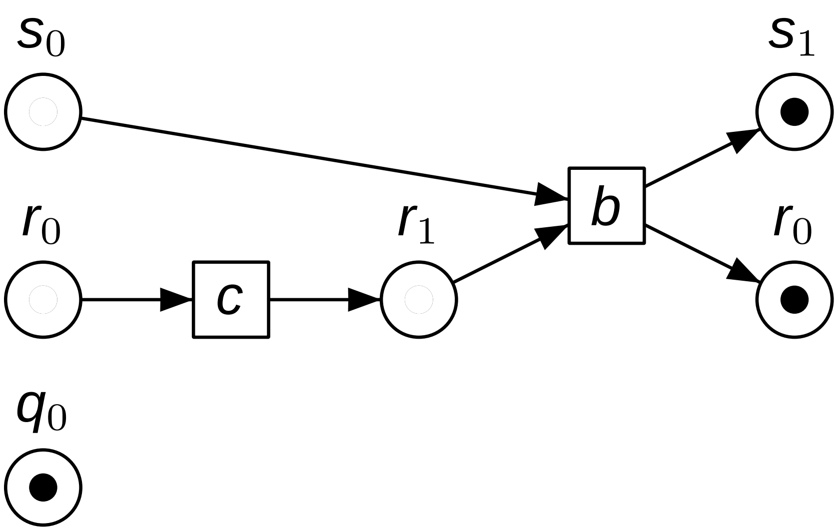
The true concurrency semantics of Petri nets



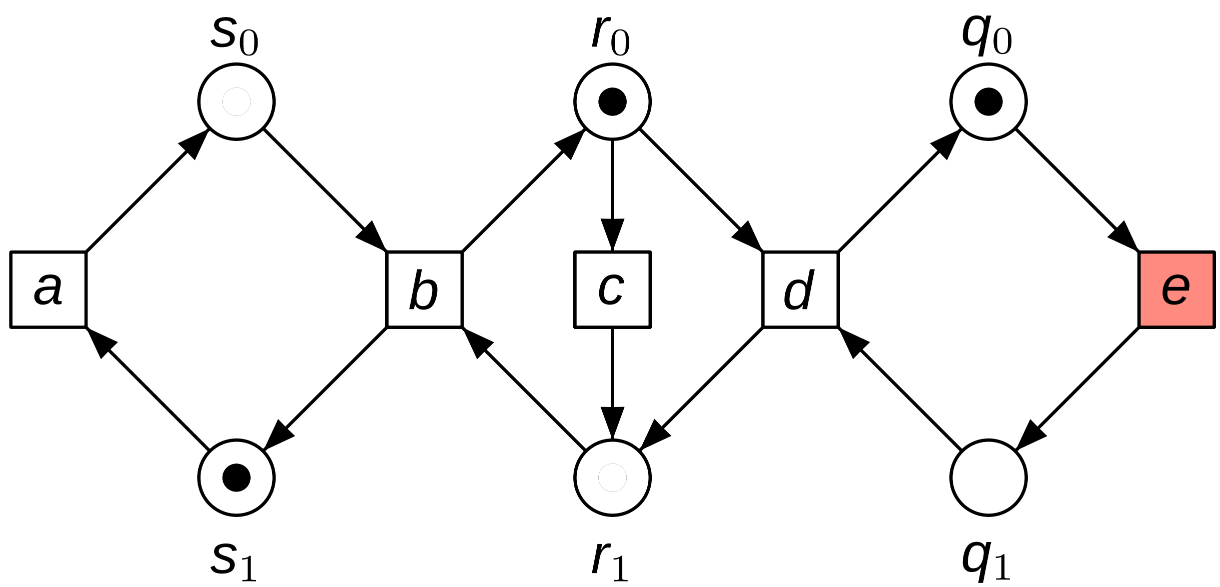


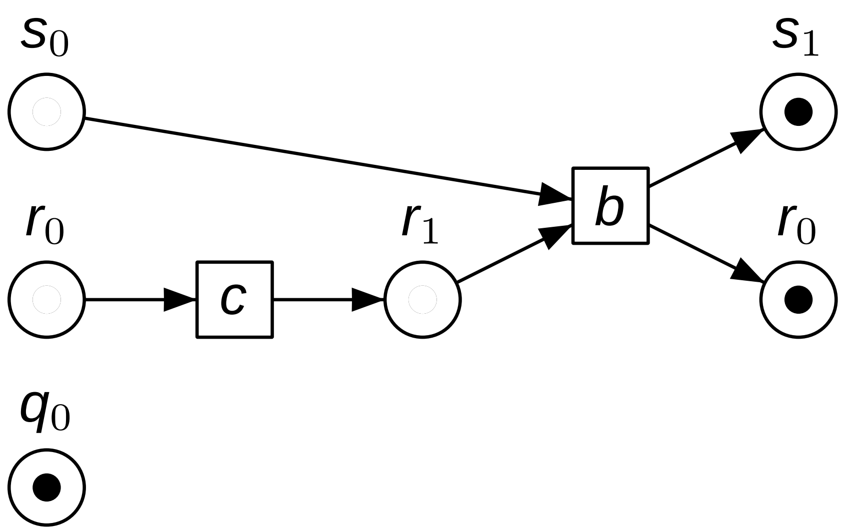
The true concurrency semantics of Petri nets



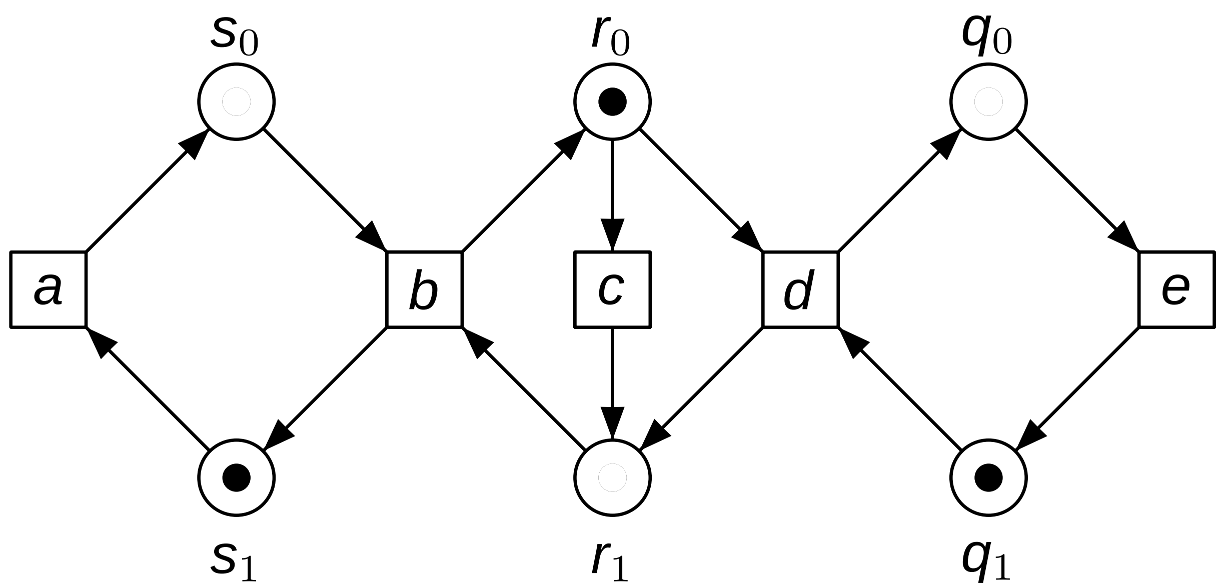


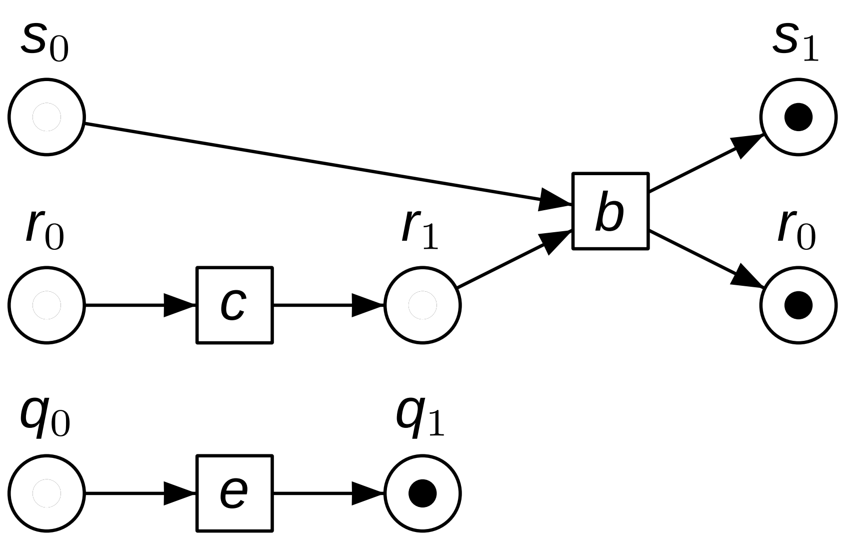
The true concurrency semantics of Petri nets



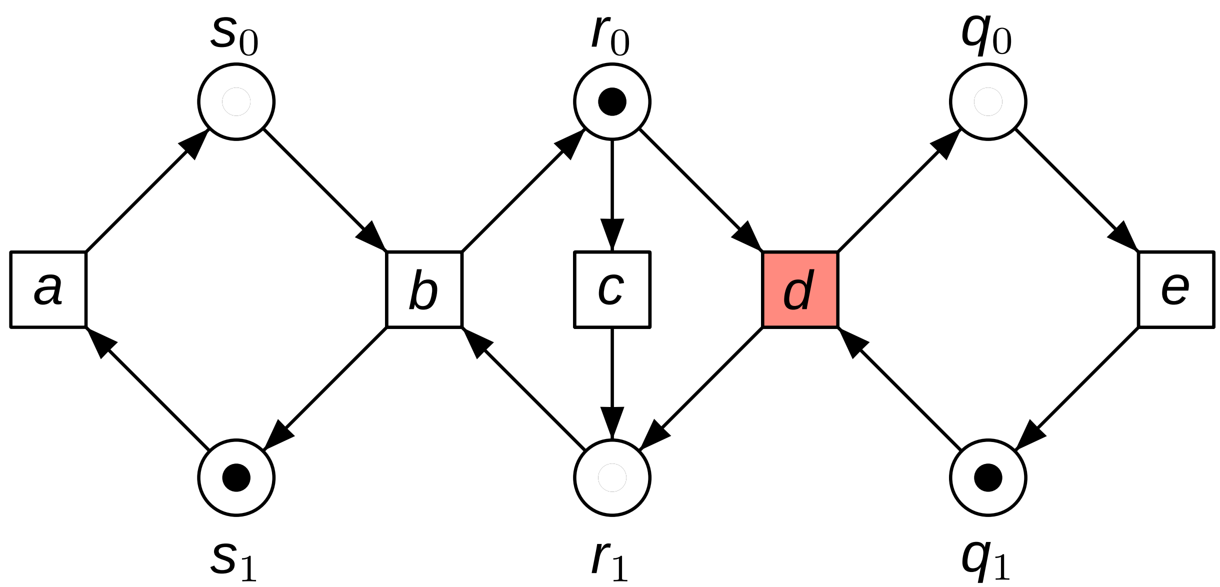


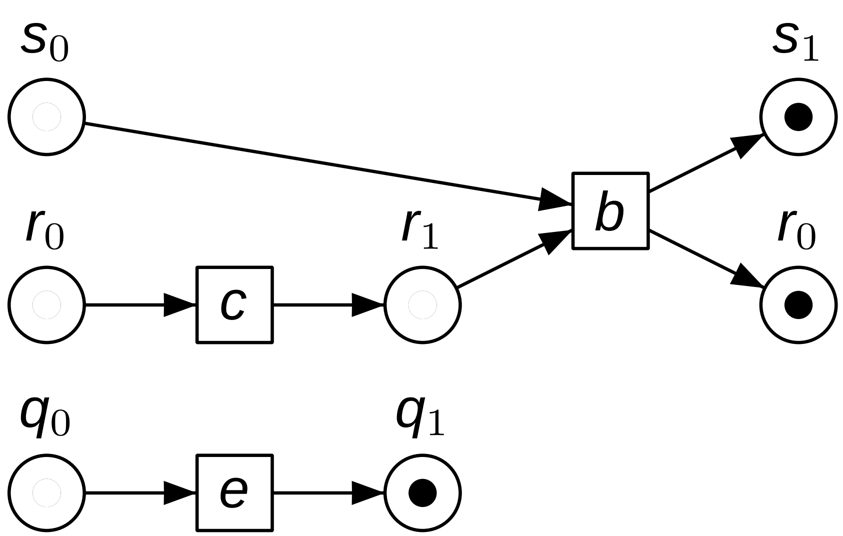
The true concurrency semantics of Petri nets



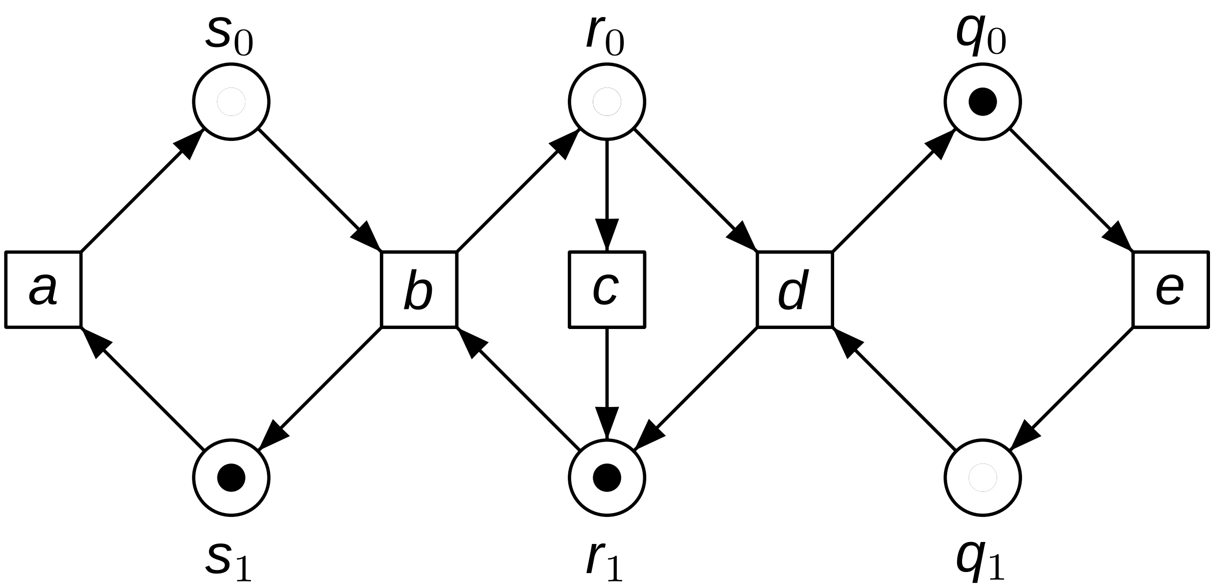


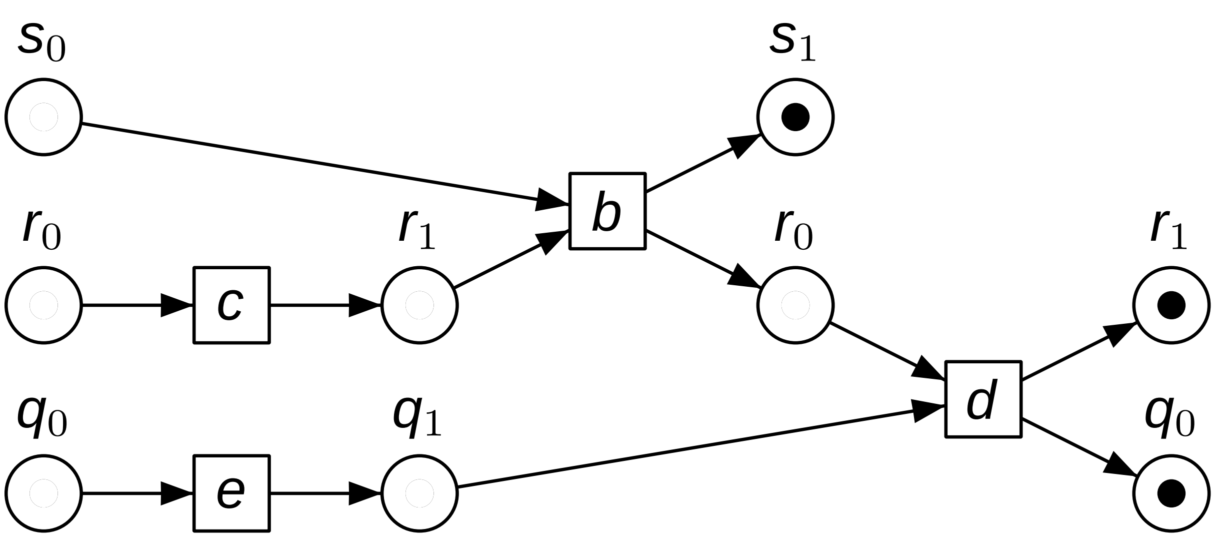
The true concurrency semantics of Petri nets



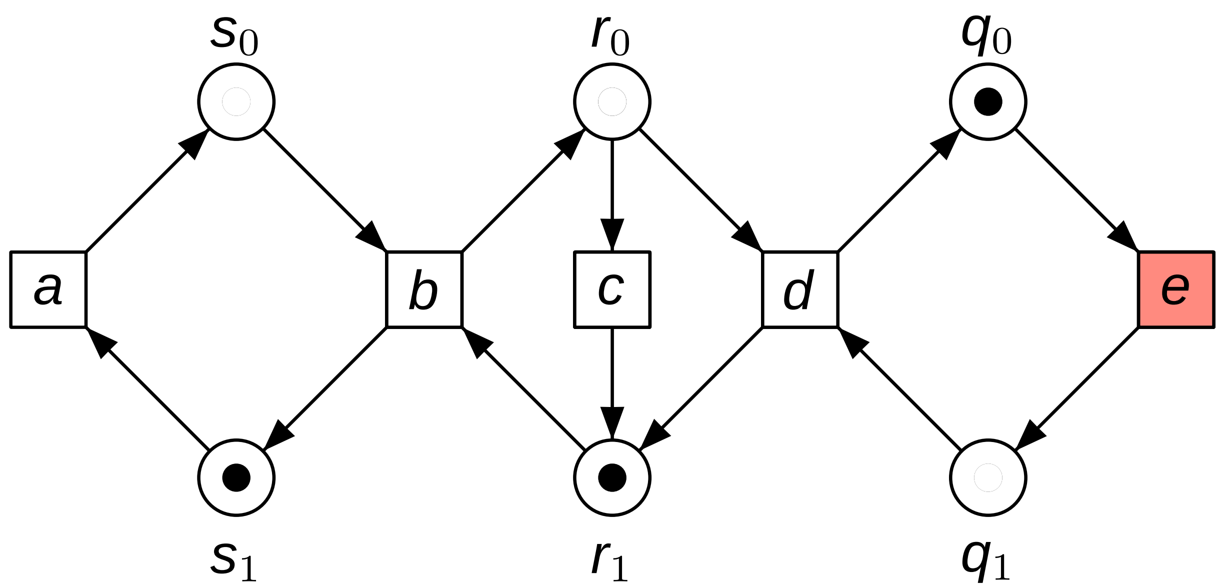


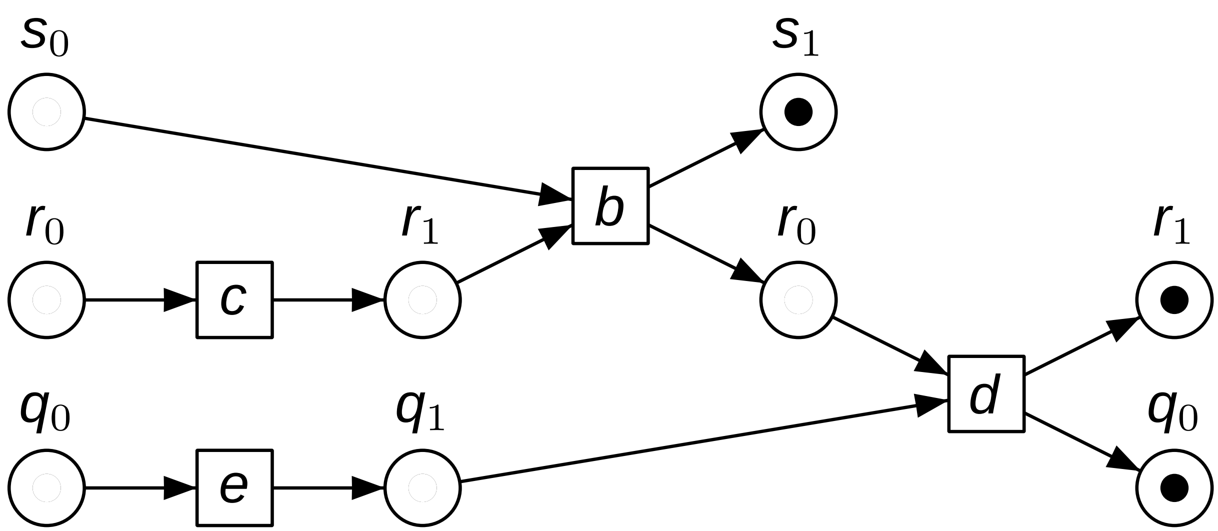
The true concurrency semantics of Petri nets



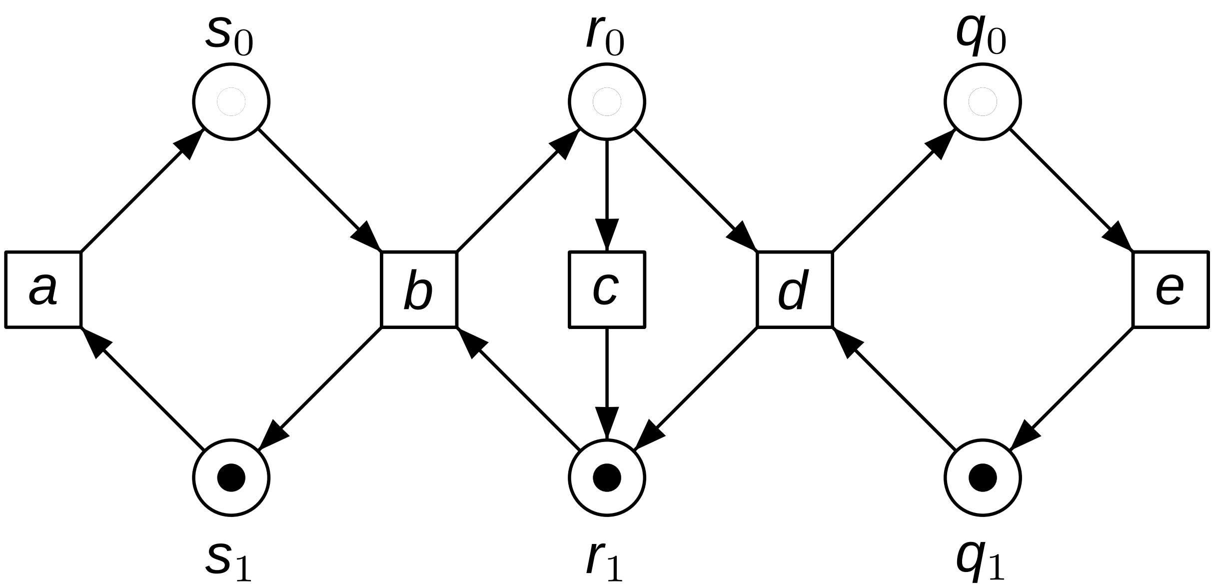


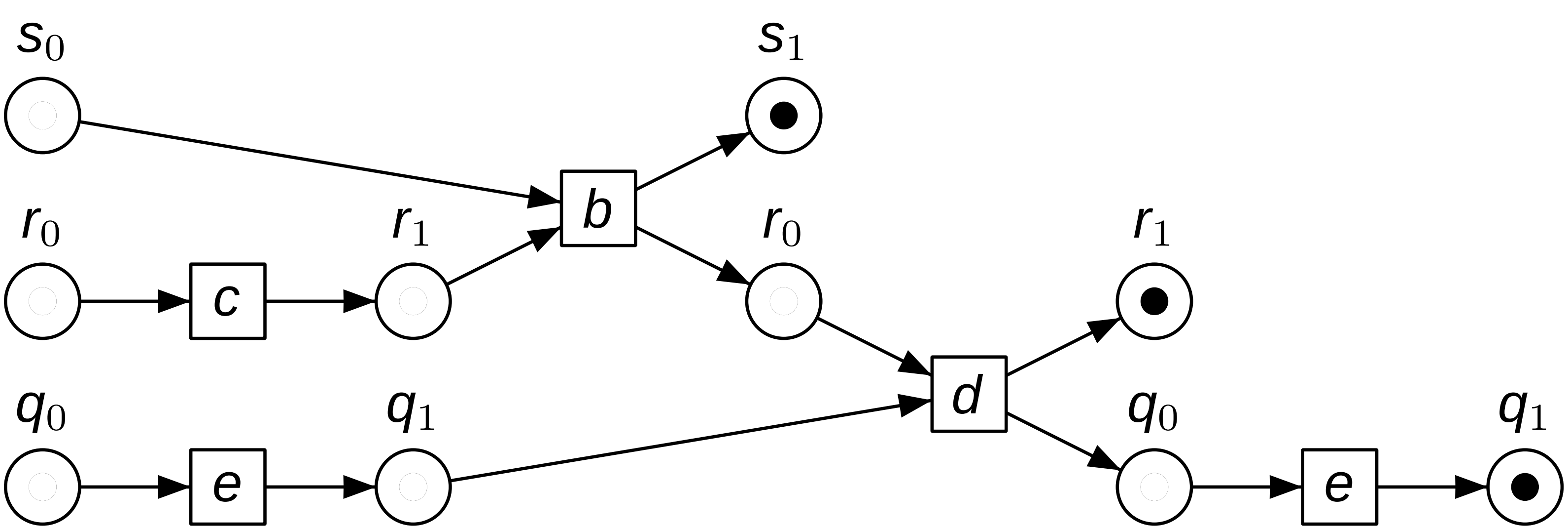
The true concurrency semantics of Petri nets



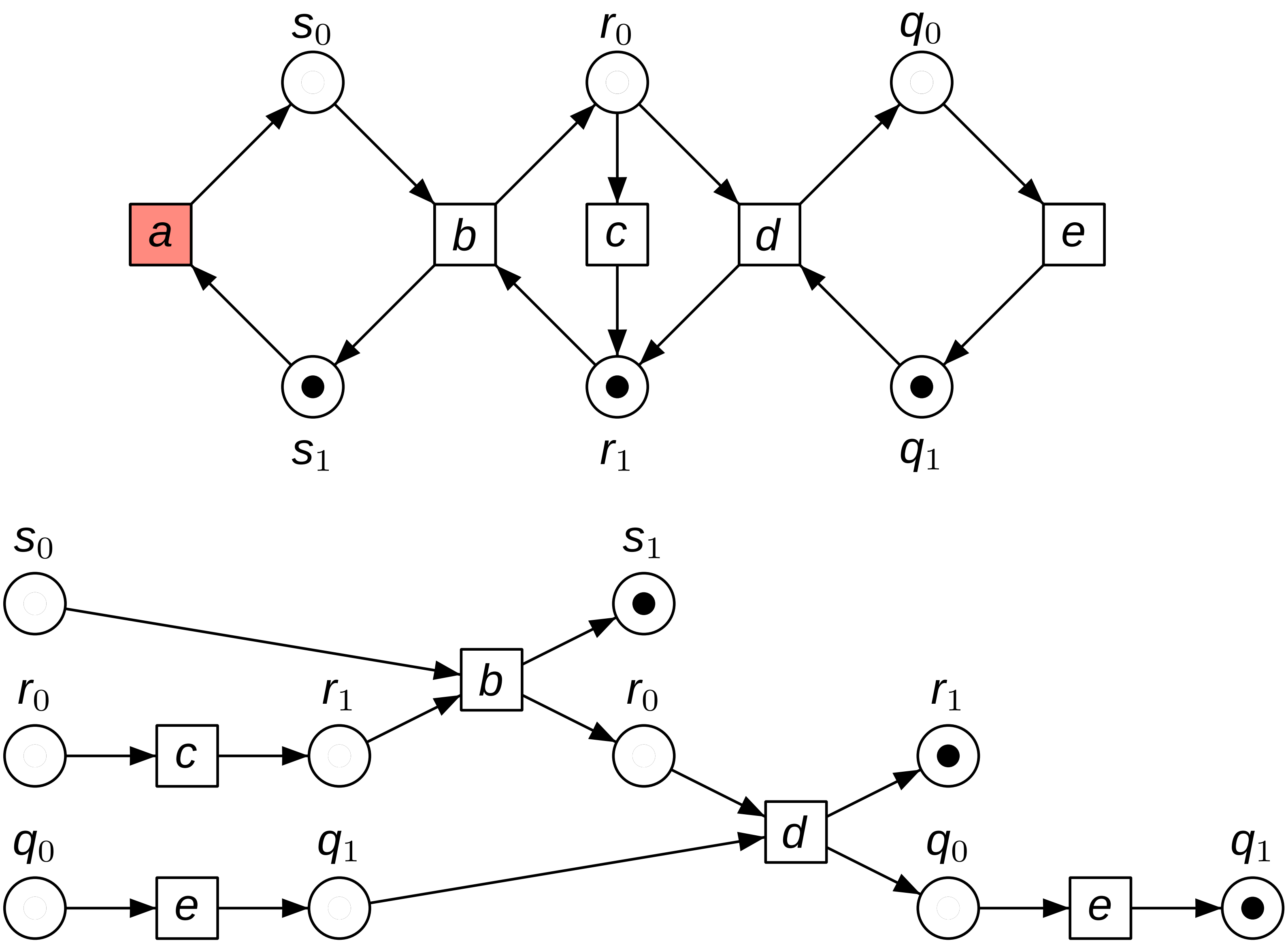


The true concurrency semantics of Petri nets

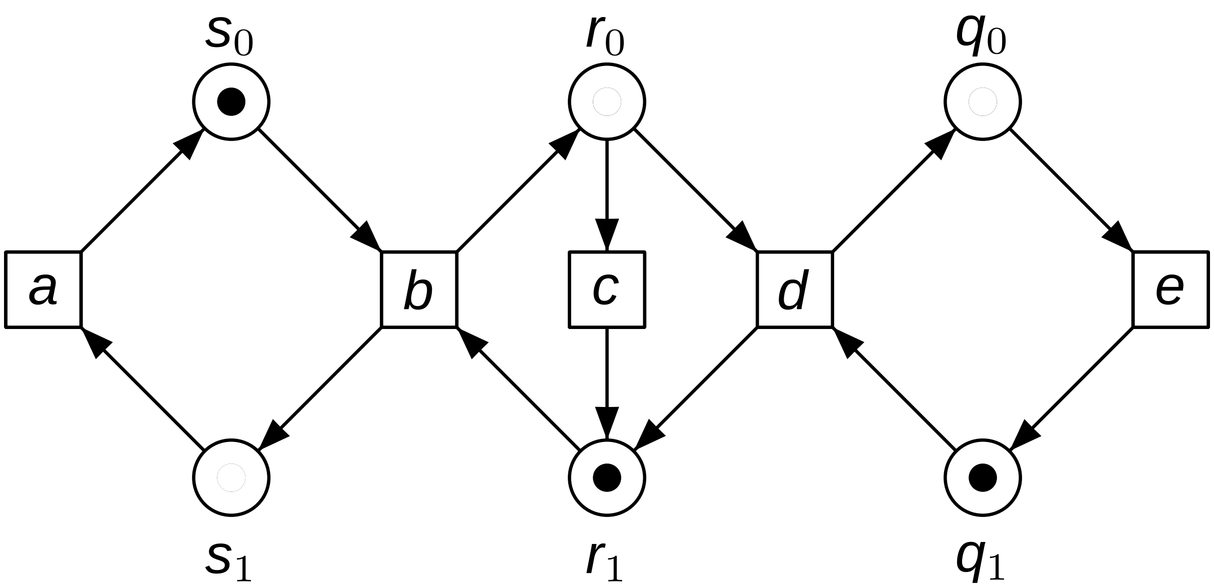


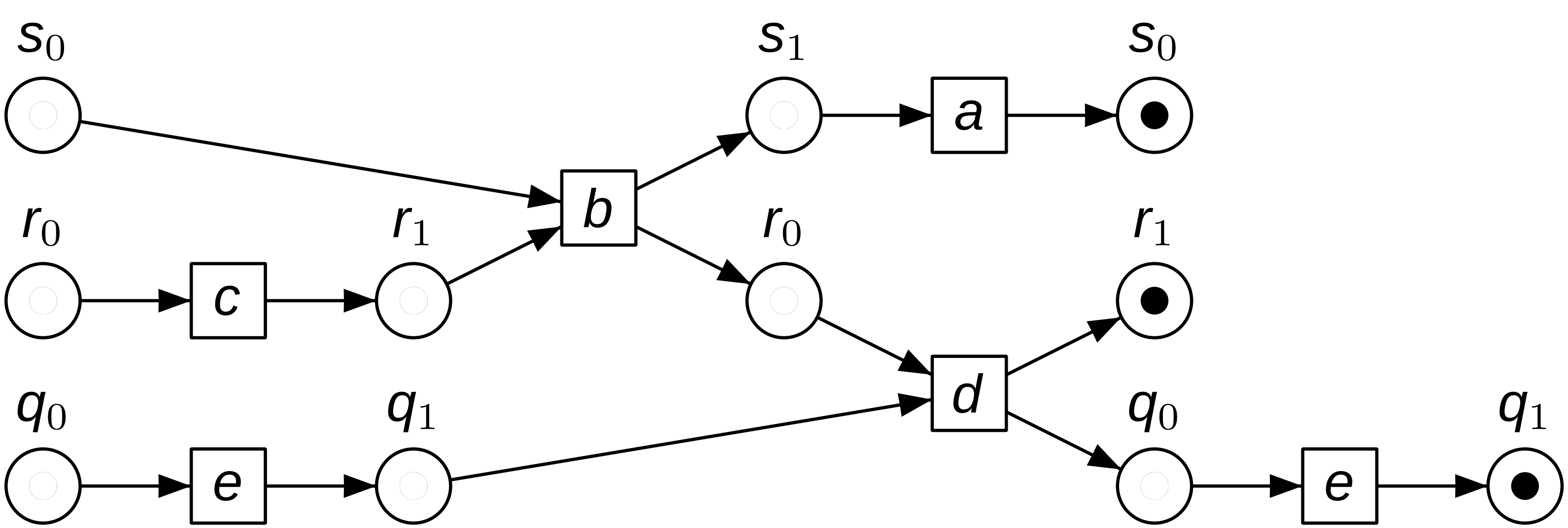


The true concurrency semantics of Petri nets

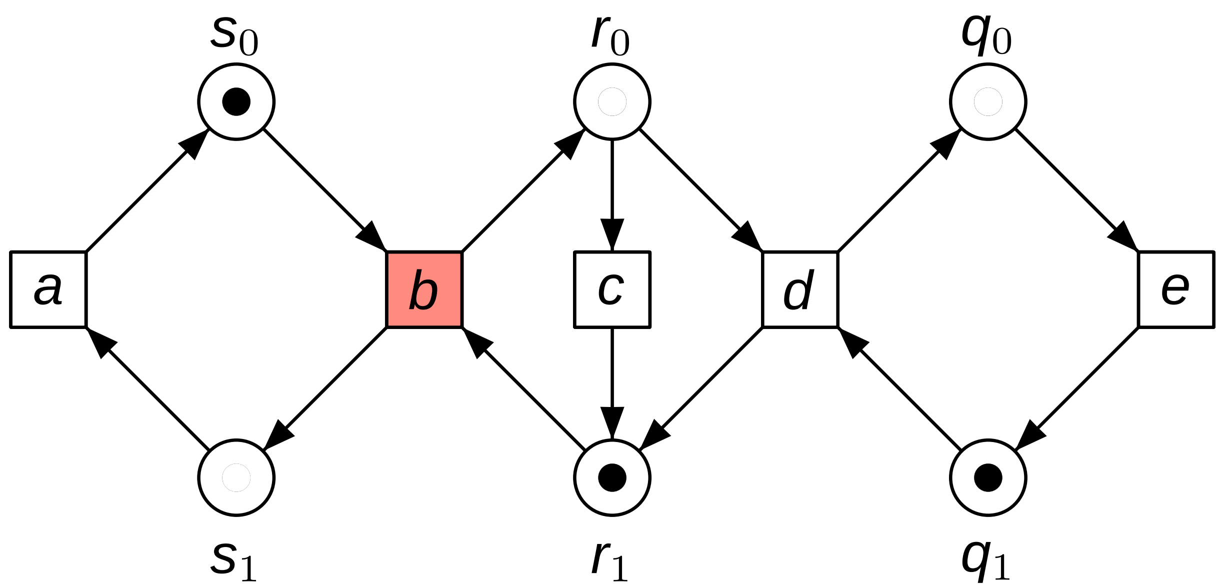


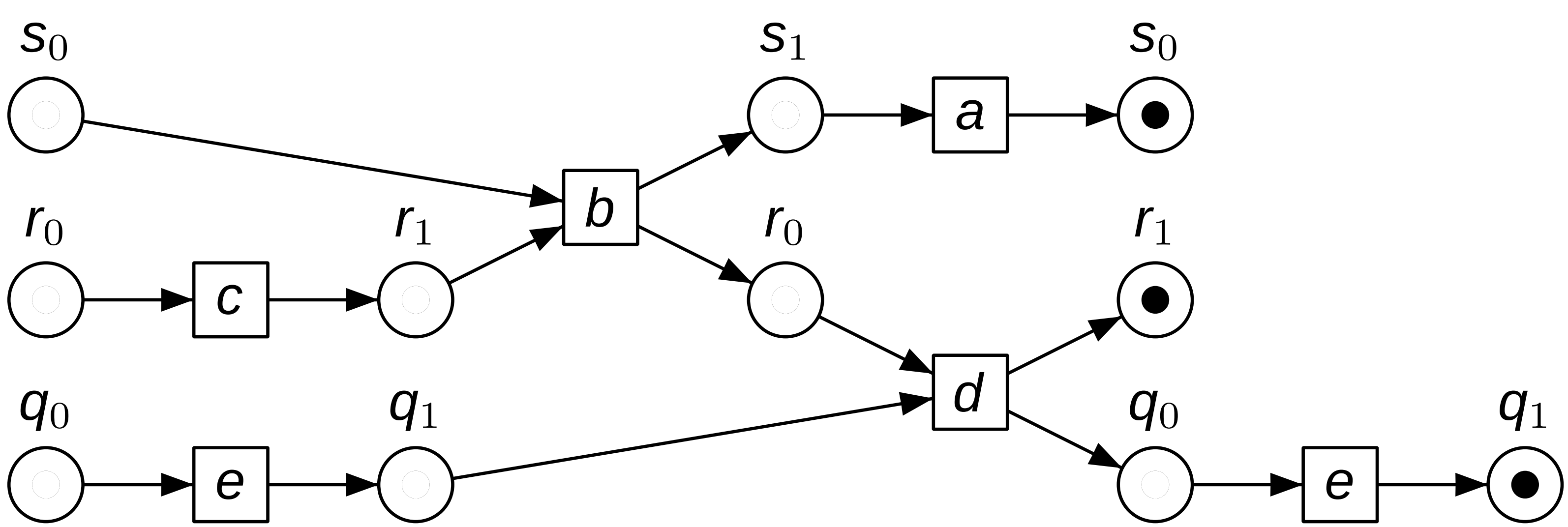
The true concurrency semantics of Petri nets



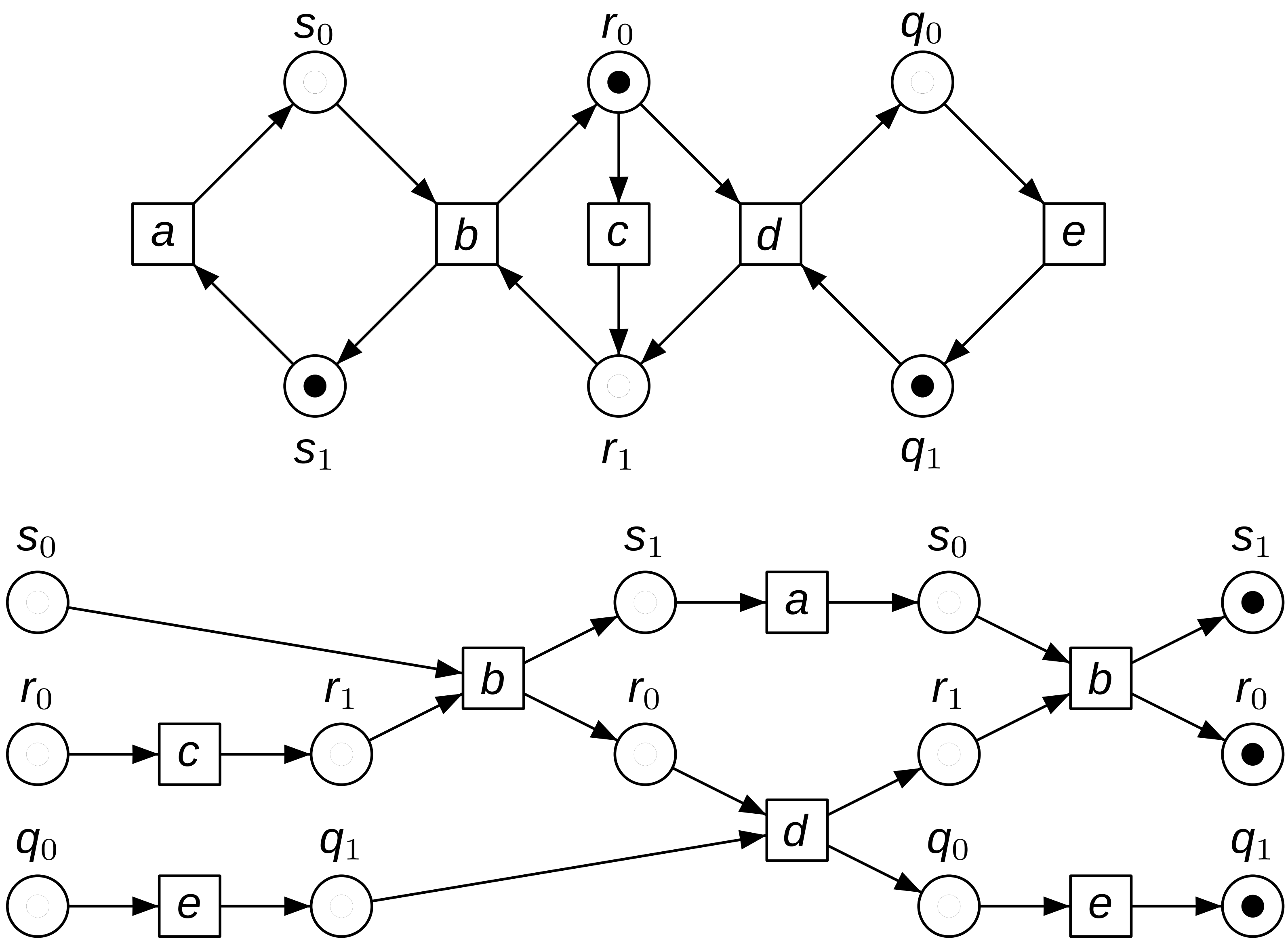


The true concurrency semantics of Petri nets





The true concurrency semantics of Petri nets



Interleaving vs. true concurrency

The interleaving thesis:

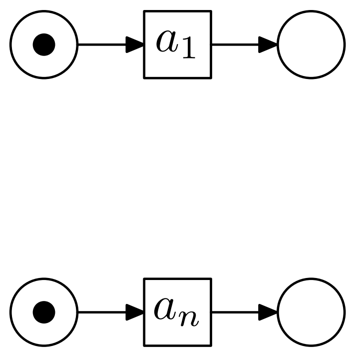
> The total order assumption is a reasonable abstraction, adequate for practical purposes, and leading to nice mathematics

The true concurrency thesis:

> The total order assumption does not correspond to physical reality and leads to awkward representations of simple phenomena

The standard example

In interleaving semantics, a system composed of n independent components



has $n!$ different executions

The automaton accepting them has 2^n states

In true concurrency semantics, it has only one nonsequential execution

30 years of concurrency theory in one slide

Interleaving semantics

Petri nets/vector addition systems (Hack, Kosaraju, Mayr, . . . 70s–80s)
Process algebras (Milner 80)
Temporal logic (Pnueli 77)
Model checking (Clarke, Emerson, Queille, Sifakis 81)

True concurrency

Axiomatic concurrency theory (Best, Fernandez, Petri . . . 70s–80s)
Trace theory (formal languages, Mazurkiewicz 77)
Event structures (domain theory, Winskel 80)
True concurrency semantics of process algebras (Montanari, Winskel . . . 80s)
Partial order model checking (E., Godefroid, Peled, Wolper 90s)
Temporal logics for true concurrency (90s–00s)

Temporal Logics for True Concurrency

LTL: a temporal logic for sequential runs

Syntax: $\varphi ::= \mathbf{true} \mid \neg\varphi \mid \varphi \vee \psi \mid \langle a \rangle \varphi \mid \mathbf{F}\,\varphi \mid \mathbf{G}\,\varphi \mid \varphi \mid \varphi\,\mathbf{U}\,\psi$
where a belongs to a finite set Act of actions

Formulas interpreted on runs over Act : elements of Act^ω

Semantics:

$\rho \models \langle a \rangle \varphi$	if	$\rho = a\rho'$ and $\rho' \models \varphi$
$\rho \models \mathbf{F}\,\varphi$	if	$\rho' \models \varphi$ for some suffix ρ' of ρ
$\rho \models \mathbf{G}\,\varphi$	if	$\rho' \models \varphi$ for all suffixes ρ' of ρ
$\rho \models \varphi\,\mathbf{U}\,\psi$	if	$\rho' \models \psi$ for some suffix ρ' of ρ and $\rho'' \models \phi$ for all suffixes ρ'' between ρ and ρ'

Examples

Invariants: $\mathbf{G}\,\varphi$

$\mathbf{G}(\langle a_1\rangle\mathbf{true} \vee \ldots \vee \langle a_n\rangle\mathbf{true})$ deadlock freedom

Response, recurrence: $\mathbf{G}(\varphi \Rightarrow \mathbf{F}\,\psi)$

$\mathbf{G}(\langle \mathit{request}\rangle\mathbf{true} \Rightarrow \mathbf{F}\langle \mathit{taken}\rangle\mathbf{true})$ eventual access to a resource

$\mathbf{G}\,\mathbf{F}\langle \mathit{active}\rangle\mathbf{true}$ process remains active

Reactivity: $\mathbf{G}\,\mathbf{F}\,\varphi \Rightarrow \mathbf{G}\,\mathbf{F}\,\psi$

$\mathbf{G}\,\mathbf{F}(\langle \mathit{request}_1\rangle\mathbf{true} \wedge \neg\langle \mathit{taken}_2\rangle\mathbf{true}) \Rightarrow$
$\mathbf{G}\,\mathbf{F}\langle \mathit{taken}_1\rangle\mathbf{true}$ strong fairness

Model checking

Fix a system S with action alphabet Act

We use LTL over Act to specify properties of S

S satisfies a formula φ, denoted $S \models_{\mathrm{LTL}} \varphi$, if all its executions satisfy φ

The model checking problem: given S and φ, decide if $T \models_{\mathrm{LTL}} \varphi$

Results on LTL

Kamp's theorem: LTL has the same expressivity as the first-order theory of runs

$$FO(Act) ::= R_a(x) \mid x \leq y \mid \neg\varphi \mid \phi \vee \psi \mid \exists x.\varphi$$

The satisfiability and model-checking problems are PSPACE-complete

Construct a Büchi automaton of size $2^{O(|\varphi|)}$ accepting the runs satisfying φ

(Intersect it with an automaton accepting all executions of S)

Check for emptiness

LTrL: Interpreting LTL on nonsequential runs

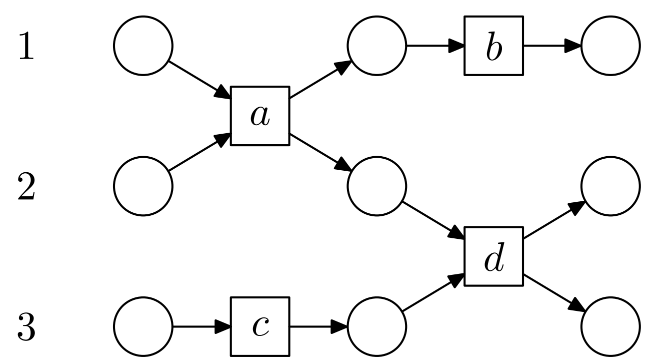
Fix a distributed alphabet $Act = (Act_1, \ldots, Act_n)$ of actions

$a \in Act_i \cap Act_j$ means that a is a joint action of the i-th and the j-th agents

Denote by $NS(Act)$ the set of nonsequential runs over Act

- The line of the i-th component only contains actions of Act_i
- Joint actions 'synchronize' the lines of its agents

Example: $Act_1 = \{a, b\}$, $Act_2 = \{a, d\}$, $Act_3 = \{c, d\}$



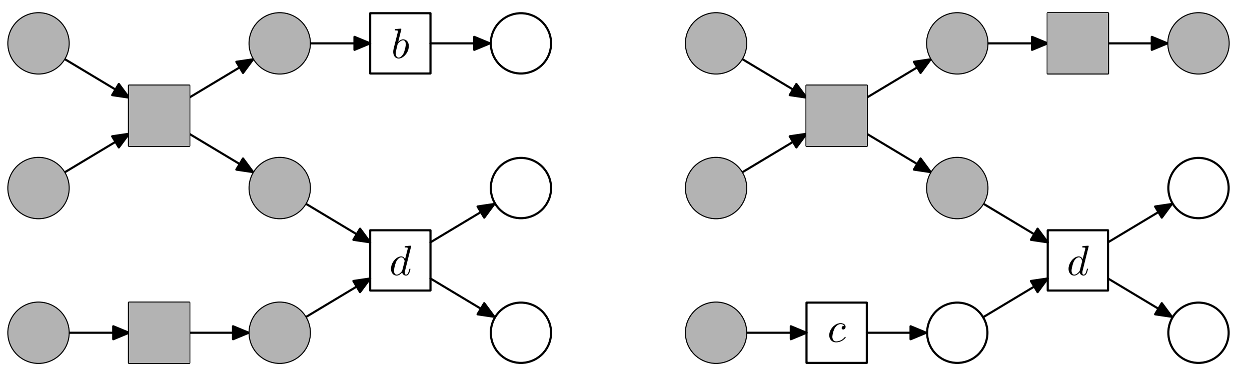
LTL now interpreted on $NS(Act)$

Same semantics as LTL, but with a new notion of suffix

Prefixes of a nonsequential run:

- All minimal places belong to the prefix
- If a transition belongs to the prefix, so do its output places

Suffixes: ‘complements’ of prefixes



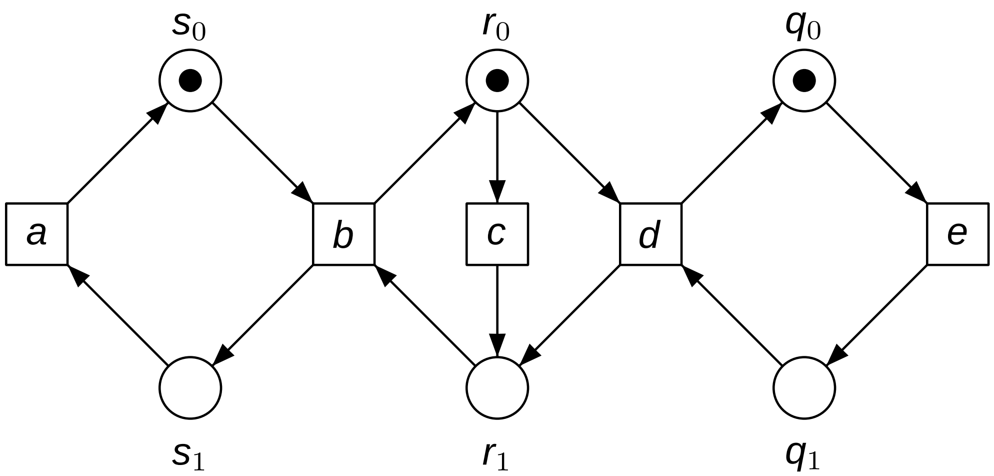
Semantics:

Semantics:	$\nu \models \langle a \rangle \varphi$	if	ν can be extended by an a-labelled event e such that $\nu \cup \{e\} \models \varphi$
	$\nu \models \mathbf{F}\, \varphi$	if	$\nu \models \varphi$ for some suffix ν' of ν
	$\nu \models \mathbf{G}\, \varphi$	if	$\nu \models \varphi$ for all suffixes ν' of ν
	$\nu \models \varphi\, \mathbf{U}\, \psi$	if	$\nu' \models \psi$ for some suffix ν' of ρ and $\nu'' \models \phi$ for all suffixes ν'' between ν and ν'

Where is the difference?

$\langle a \rangle \mathbf{true} \wedge \langle b \rangle \mathbf{true}$ unsatisfiable in LTL, satisfiable in LTrL

Example satisfies $\mathbf{G}\,\mathbf{F}(\langle a \rangle \langle e \rangle \mathbf{true})$ as a formula of LTrL, but not as a formula of LTL



Better specification of ‘reset states’

Model checking

Fix a system $S = (S_1, \dots, S_n)$ with distributed alphabet Act,

We use LTL over Act to specify properties of S

S satisfies a formula φ if all its nonsequential executions satisfy φ

The model checking problem: given S and φ, decide if $T \models_{\mathrm{LTrL}} \varphi$

Results on LTrL

Expressively complete for the first order theory of nonsequential runs

$FO(Act) ::= R_a(x) \mid x \leq y \mid \neg\varphi \mid \phi \vee \psi \; \exists x.\varphi$

$\leq$ interpreted on partial orders over Act that respect the distribution

Thiagarajan and Walukiewicz, LICS '97: LTrL + $\mathbf{P}_a$ modalities

Diekert and Gastin, CSL '99: LTrL + X_A^* modalities

Diekert and Gastin, ICALP '00

Non-elementary satisfiability and model checking problems (Walukiewicz ICALP'98)

LTL allows to specify 2^n-counters with formulas of length $O(n)$

LTrL allows to specify $Tower(2, n)$-counters with formulas of length $O(n)$

Local LTL: interpreting LTL on local states

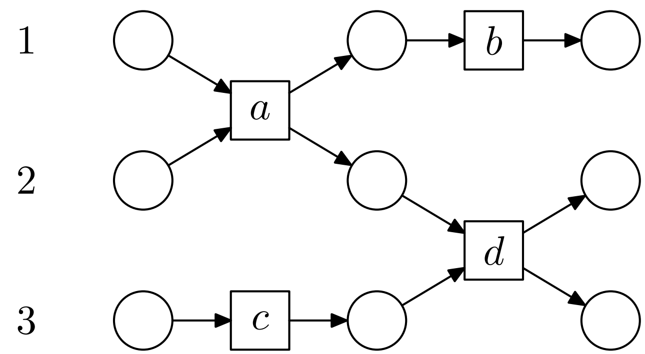
Local state of a component: 'a position in its time line'

Identify a local state (place) with the prefix determined by all its predecessors

A component always has complete information about its causal past

Components exchange full information when they synchronize

Example: $Act_1 = \{a, b\}$, $Act_2 = \{a, d\}$, $Act_3 = \{c, d\}$



Syntax: $\varphi ::= \mathbf{true} \mid \neg\varphi \mid \varphi \vee \psi \mid \langle a \rangle^i \varphi \mid \mathbf{F}^i\, \varphi \mid \mathbf{G}^i\, \varphi \mid \varphi\, \mathbf{U}^i\, \psi$

where a is an action and $1 \leq i \leq n$

Semantics:

$\langle a \rangle^i \varphi$	means	φ holds at i's next local state
$\mathbf{F}^i\, \varphi$	means	φ holds eventually at i's timeline
$\mathbf{G}^i\, \varphi$	means	φ holds always along i's timeline
$\varphi\, \mathbf{U}_i\, \psi$	means	φ holds until ψ holds along i's timeline

Gets interesting when formulas use several indices: $\mathbf{F}^1\, \mathbf{G}^2 \langle a \rangle^1 \mathbf{true}$

Results on Local LTL

PSPACE-complete satisfiability and model checking problems
(Thiagarajan, LICS '94)

Generalization of the Büchi automaton construction

Technical problem: to keep track of the latest gossip

Expresiveness still unclear

Completeness results for logics with similar flavours
(Gastin, Mukund, Kumar MFCS '03)

Difficult to specify with

Outlook

Interleaving semantics 'default' semantics in practice

True concurrency brought in when interleaving 'fails'

Challenge: automatic synthesis of distributed systems

- Interleaving logics insensitive to distribution requirements
- Maybe the 'killer application' for true concurrency?