Lezione 31 MSC
Petri Nets:
Some Behavioral Equivalences and Nonpermissive nets

Roberto Gorrieri

Which equivalences?

- **Net isomorphism** (decidable for finite nets: exponential) – distributed semantics
- **Interleaving bisimilarity** (decidable for bounded nets, but undecidable in general) – sequential semantics
- **Step bisimilarity** (decidable for bounded nets, but undecidable in general) – parallel semantics
- Many others (e.g. causal-net bisimulation) …. we will not discuss

Net isomorphism (1)

Definition 3.15. (Net isomorphism) Given two P/T nets $N_1 = (S_1, A, T_1)$ and $N_2 = (S_2, A, T_2)$, we say that N_1 and N_2 are *isomorphic* – denoted $N_1 \cong N_2$ – if there exists a bijection $f : S_1 \to S_2$, homomorphically extended to markings,[3] such that $(m, \ell, m') \in T_1$ if and only if $(f(m), \ell, f(m')) \in T_2$.

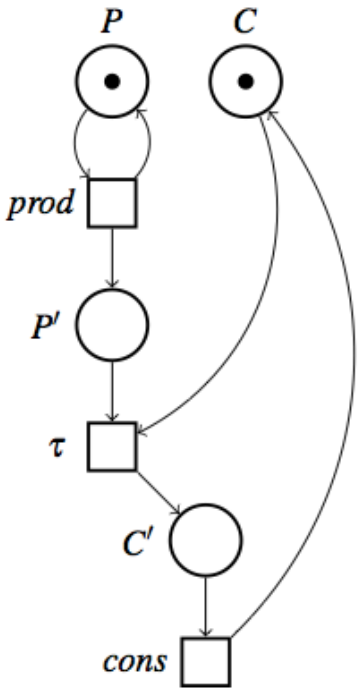
Two P/T net systems $N_1(m_1)$ and $N_2(m_2)$ are *rooted isomorphic* – denoted $N_1(m_1) \cong_r N_2(m_2)$ – if the isomorphism $f : S_1 \to S_2$ ensures, additionally, that $f(m_1) = m_2$. □

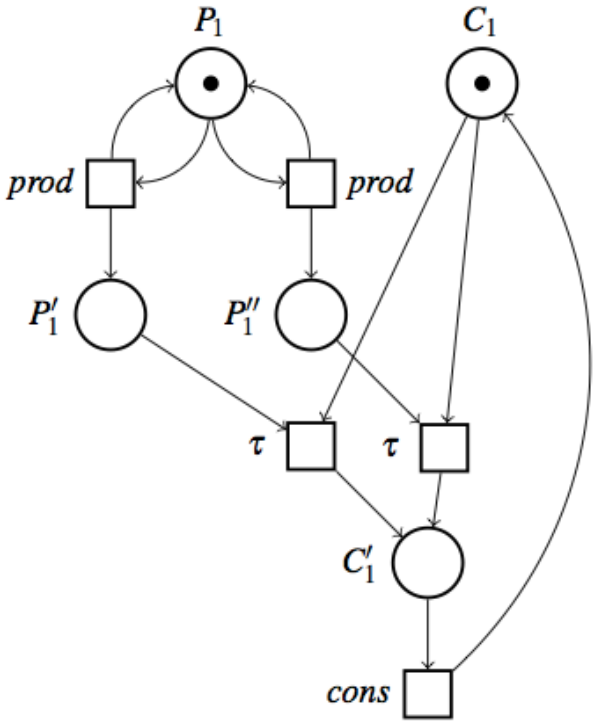
[3] This means that f is applied element-wise to each component of the marking, i.e., $f(m_1 \oplus m_2) = f(m_1) \oplus f(m_2)$.

(Rooted) net isomorphism is an equivalence relation, i.e., reflexive, symmetric and transitive.

Net isomorphism (2)

- Net isomorphism is often a too-concrete equivalence relation.
- Two non-isomorphic nets: it seems that there is no observable reason to tell them apart.





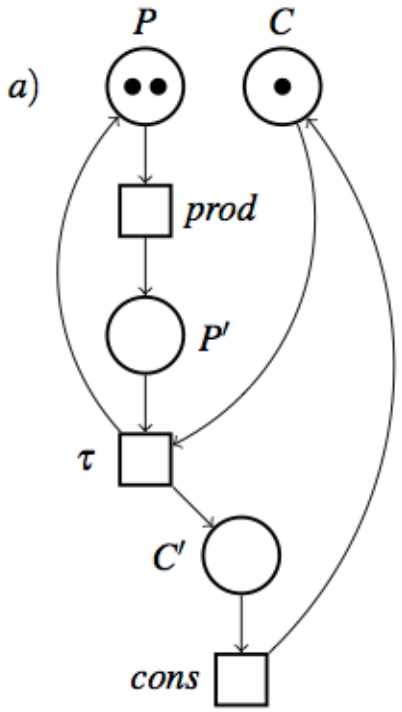
Net isomorphism (3)

Net isomorphism is **decidable** for finite P/T nets.

The problem of determining whether two finite net systems are (rooted) isomorphic is a generalization of the problem of checking whether two finite graphs are isomorphic, as Petri nets can be equivalently represented as bipartite graphs (i.e., graphs with two kinds of nodes – places and transitions – as in the pictorial representation of nets). This problem is in the class NP, which is not known to be in P, and the best known algorithm is exponential in the number of nodes.

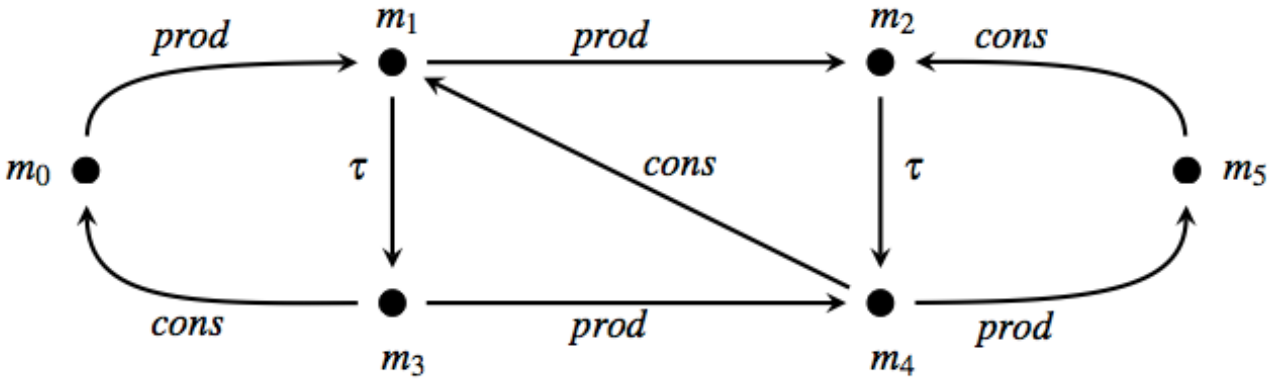
Interleaving semantics

Definition 3.16. (Interleaving marking graph) The *interleaving marking graph* of $N(m_0) = (S, A, T, m_0)$ is the rooted LTS $IMG(N(m_0)) = ([m_0\rangle, A, \rightarrow, m_0)$, where m_0 is the initial state, the set of states is given by the set $[m_0\rangle$ of reachable markings, and the transition relation $\rightarrow \subseteq \mathcal{M}_{fin}(S) \times A \times \mathcal{M}_{fin}(S)$ is defined by $m \xrightarrow{a} m'$ if and only if there exists a transition $t \in T$ such that $m[t\rangle m'$ and $l(t) = a$. □



$m_0 = 2 \cdot P \oplus C$ $\quad m_1 = P \oplus C \oplus P'$ $\quad m_2 = C \oplus 2 \cdot P'$

$m_3 = 2 \cdot P \oplus C'$ $\quad m_4 = P \oplus C' \oplus P'$ $\quad m_5 = C' \oplus 2 \cdot P'$



Properties

Proposition 3.7. *Given a P/T net system $N(m_0)$, the following hold:*

1. *if $N(m_0)$ is finite and bounded, then $IMG(N(m_0))$ is a finite-state LTS;*
2. *if $N(m_0)$ is finite but not bounded, then $IMG(N(m_0))$ is not finite-state;*
3. *if $N(m_0)$ is bounded and $IMG(N(m_0))$ is not finite-state, then N is infinite;*
4. *if $IMG(N(m_0))$ is finite-state, then the dynamically reachable subnet $Net_d(N(m_0))$ is finite and bounded;*
5. *if $N(m_0)$ is distinct, then $IMG(N(m_0))$ is deterministic.* □

Trace equivalence

Definition 3.17. (Interleaving trace equivalence) Given a net system $N(m_0) = (S, A, T, m_0)$, we write the set of traces of $N(m_0)$ as the set

$$Tr(IMG(N(m_0))) = \{\sigma \in A^* \mid \exists m \in [m_0\rangle.\ m_0 \xrightarrow{\sigma}{}^* m\}.$$

With abuse of notation, we often write such a set as $Tr(N(m_0))$ or simply $Tr(m_0)$ when the net N is clear from the context. Two P/T net systems $N_1(m_1)$ and $N_2(m_2)$ are trace equivalent if $Tr(N_1(m_1)) = Tr(N_2(m_2))$.

This definition can be extended to any state of $IMG(N(m_0))$ as follows: given a marking $m \in [m_0\rangle$, we write $Tr(m) = \{\sigma \in A^* \mid \exists m' \in [m\rangle.\ m \xrightarrow{\sigma}{}^* m'\}$. Two markings $m_1, m_2 \in [m_0\rangle$ are trace equivalent if $Tr(m_1) = Tr(m_2)$. □

Example

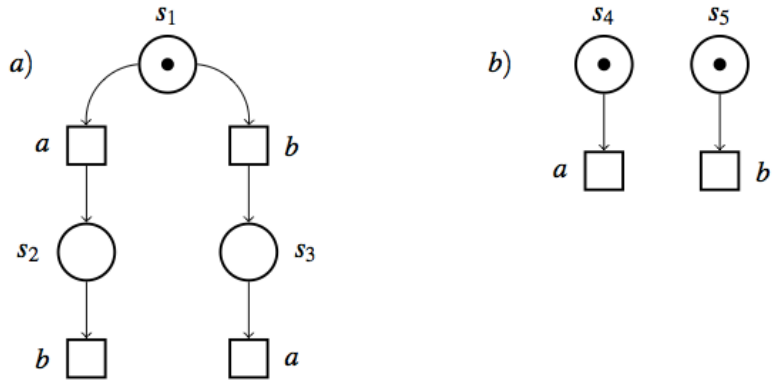


Fig. 3.13 Two interleaving trace equivalent net systems

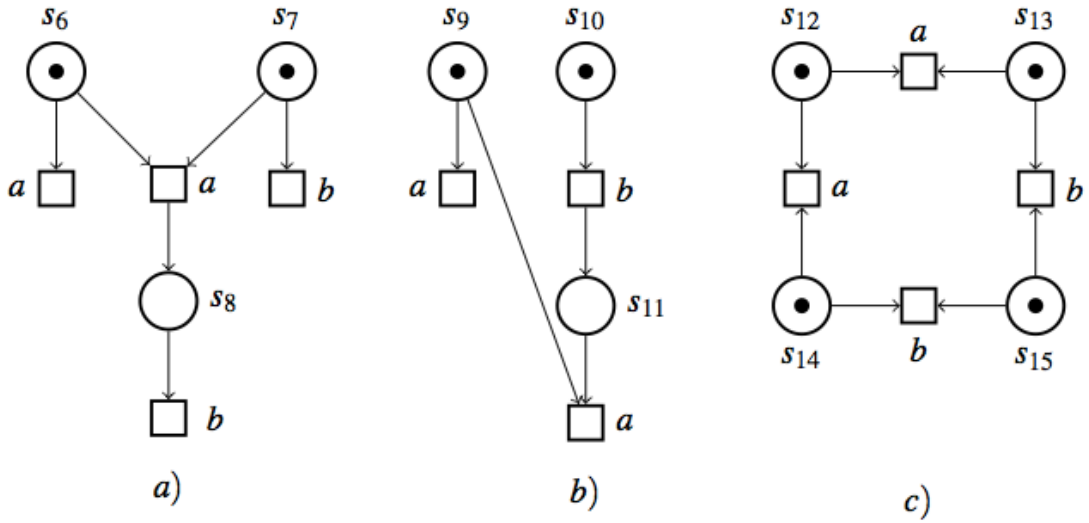


Fig. 3.14 Some other, rather different, interleaving trace equivalent net systems [GV87]

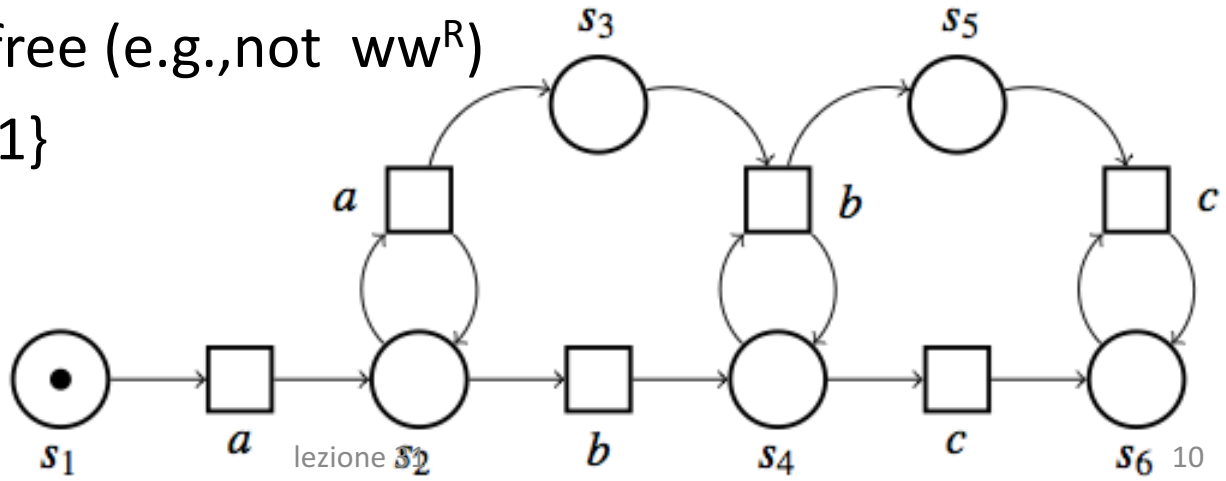
Petri Net Language

Definition 3.18. (Petri net language) Given a P/T net system $N(m_0) = (S, A, T, m_0)$, such that $B = A \setminus \{\tau\}$, we write the set of its weak completed traces as

$$WCTr(IMG(N(m_0))) = \{\sigma \in B^* \mid \exists m \in [m_0\rangle.\ m_0 \stackrel{\sigma}{\Longrightarrow} m \stackrel{\ell}{\not\Rightarrow} \text{ for all } \ell \in B\}.$$

The Petri net language $L[N(m_0)]$ recognized by the net system $N(m_0)$ is exactly the set $WCTr(IMG(N(m_0)))$ of its weak completed traces. □

- Included in the class of context-dependent
- Not all context-free (e.g.,not ww^R)
- $\{a^n b^m c^m \mid n \geq m \geq 1\}$



Interleaving bisimilarity

Definition 3.19. (Interleaving bisimulation equivalence) The P/T systems $N_1(m_1)$ and $N_2(m_2)$ are *interleaving bisimilar* – denoted $N_1(m_1) \sim N_2(m_2)$, or $m_1 \sim m_2$ when the nets are clear from the context – if and only if there exists a strong bisimulation $R \subseteq [m_1\rangle \times [m_2\rangle$ over the LTSs $IMG(N_1(m_1))$ and $IMG(N_2(m_2))$ such that $(m_1, m_2) \in R$.

Given a P/T net system $N(m_0)$ and a rooted LTS $TS(q_0) = (Q, A, \rightarrow, q_0)$, we say that N and TS are interleaving bisimilar, denoted $N(m_0) \sim TS(q_0)$ or even $m_0 \sim q_0$, if and only if there exists a strong bisimulation $R \subseteq [m_0\rangle \times Q$ over the LTSs $IMG(N(m_0))$ and $TS(q_0)$ such that $(m_0, q_0) \in R$. □

- All the five nets on slide 9 are pairwise interleaving bisimilar

Decidability issues

Interleaving bisimilarity coincides with trace equivalence over *distinct* P/T nets, because a distinct P/T net generates a deterministic *IMG*, and over deterministic LTSs the two equivalences coincide. Since trace equivalence is decidable over distinct finite P/T nets, then also interleaving bisimilarity is decidable over such a restricted class of finite P/T nets.

However, bisimilarity and trace eq. are **undecidable in general** for finite P/T nets, even with only two unbounded places. This negative results holds also for finite CCS nets, because this is the case for finite-net CCS (FNC).

Of course, they are **decidable for bounded finite P/T nets**, as the associated *IMG*s are finite-state. In particular, for safe nets interleaving bisimilarity is DEXPTIME-complete; it is also decidable for BBP nets.

Step Transition System

Definition 2.22. (Step labels) Given a set *Lab* of labels, as in Definition 2.1, the set of *step labels* is $SLab = \mathscr{M}_{fin}(Lab) \setminus \{\emptyset\}$, i.e., the set of all finite, nonempty multisets over *Lab*. □

Definition 2.23. (Step transition systems) A step transition system (STS for short) is a triple $STS = (Q, \mathscr{B}, \rightarrow)$ where

- Q is the nonempty, countable set of *states*, ranged over by q (possibly indexed);
- $\mathscr{B} \subseteq SLab$ is the countable set of *step labels*, ranged over by M (possibly indexed);
- $\rightarrow \subseteq Q \times \mathscr{B} \times Q$ is the *step transition relation*.

For economy's sake, we assume that for any $M \in \mathscr{B}$ there exists a step transition $(q, M, q') \in \rightarrow$. A *rooted* step transition system is a pair (STS, q_0) where $STS = (Q, A, \rightarrow)$ is an STS and $q_0 \in Q$ is the *initial state* (or *root*). Sometimes we write $STS = (Q, \mathscr{B}, \rightarrow, q_0)$ for a rooted STS. □

Example

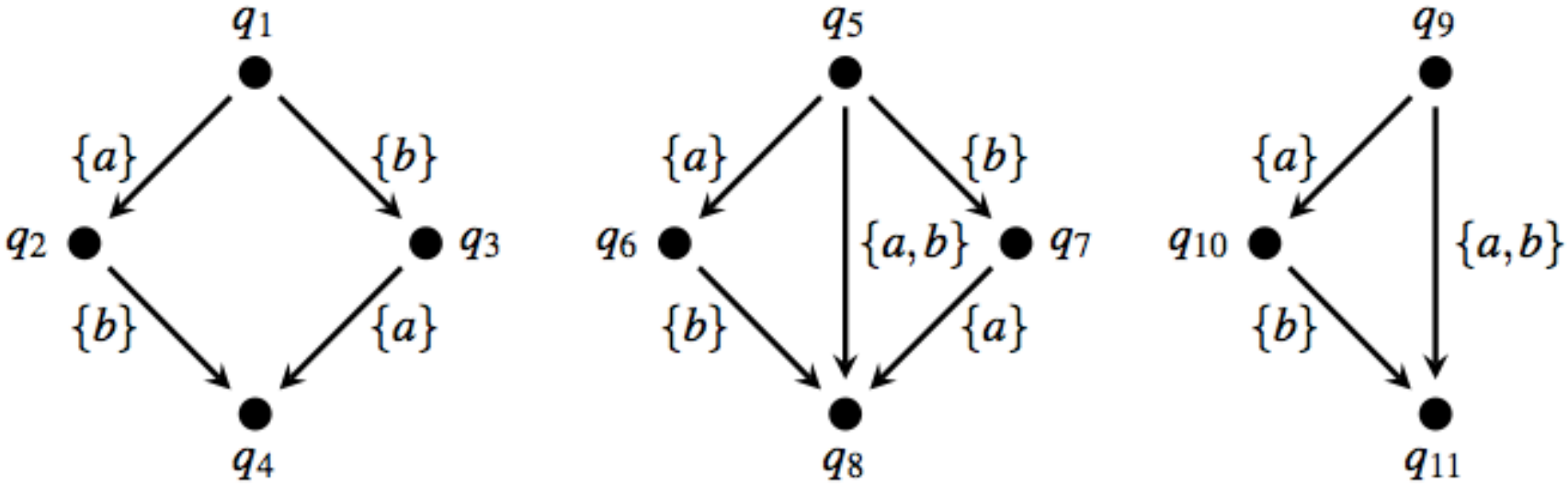


Fig. 2.10 Three step transition systems

Fully-concurrent STS

Definition 2.25. (Fully concurrent STS) An STS $(Q, \mathscr{B}, \rightarrow)$ is *fully concurrent* if whenever $q \xrightarrow{M_1 \oplus M_2} q'$, with $M_i \neq \emptyset$ for $i = 1, 2$, then there exists two states q_1 and q_2 such that

$$q \xrightarrow{M_1} q_1 \xrightarrow{M_2} q' \quad \text{and} \quad q \xrightarrow{M_2} q_2 \xrightarrow{M_1} q'. \qquad \square$$

Therefore, a fully concurrent STS $(Q, \mathscr{B}, \rightarrow)$ is such that, whenever $q \xrightarrow{M} q'$, with $M = \{\ell_1, \ell_2, \ldots, \ell_n\}$, then for any linearization $\ell'_1 \ell'_2 \ldots \ell'_n$ of the step M (i.e., for any permutation of the string $\ell_1 \ell_2 \ldots \ell_n$), there is a path

$$q_1 \xrightarrow{\{\ell'_1\}} q_2 \xrightarrow{\{\ell'_2\}} \ldots q_n \xrightarrow{\{\ell'_n\}} q_{n+1}$$

where $q_1 = q$, $q_{n+1} = q'$. By looking at Figure 2.10, it is clear that the STSs rooted in q_1 and q_5 are fully concurrent, while the STS rooted in q_9 is not.

Step of a P/T net and concurrent token game

A step $G : T \to \mathbb{N}$ is a multiset of transitions. G is enabled at marking m if ${}^\bullet G \subseteq m$, where

${}^\bullet G = \oplus_{t \in T} G(t) \cdot {}^\bullet t$ and $G(t)$ denotes the number of occurrences of transition t in the step G.

The execution of a step G enabled at m produces the marking $m' = (m \ominus {}^\bullet G) \oplus G^\bullet$, where

$G^\bullet = \oplus_{t \in T} G(t) \cdot t^\bullet$. This is written $m[G\rangle m'$.

We sometimes refer to this as the ***concurrent token game***, in opposition to the ***sequential* token game**.

Step Marking Graph

Proposition 3.8. *Given a P/T net N, a marking m and a step $G = G_1 \oplus G_2$, where $G_i \neq \emptyset$ for $i = 1,2$, if $m[G_1 \oplus G_2\rangle m'$, then there exist two markings m_1 and m_2 such that $m[G_1\rangle m_1[G_2\rangle m'$ and $m[G_2\rangle m_2[G_1\rangle m'$.*

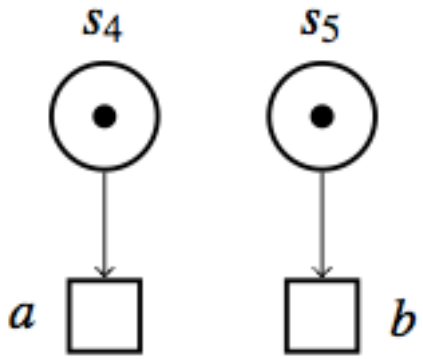
Definition 3.20. (Step marking graph) The *step marking graph* associated with a P/T net system $N(m_0) = (S, A, T, m_0)$ is the rooted STS

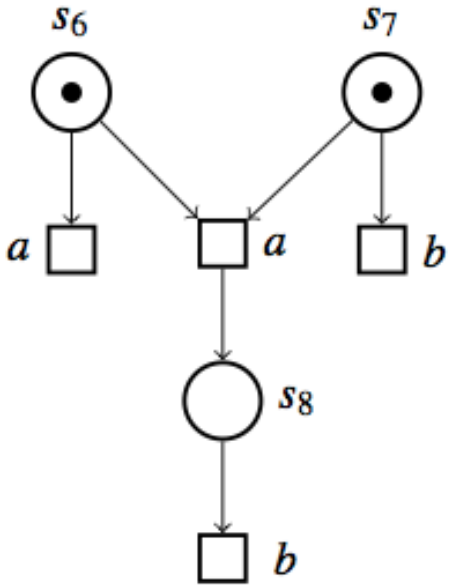
$$SMG(N(m_0)) = ([m_0\rangle, \mathscr{M}_{fin}(A), \longrightarrow_s, m_0)$$

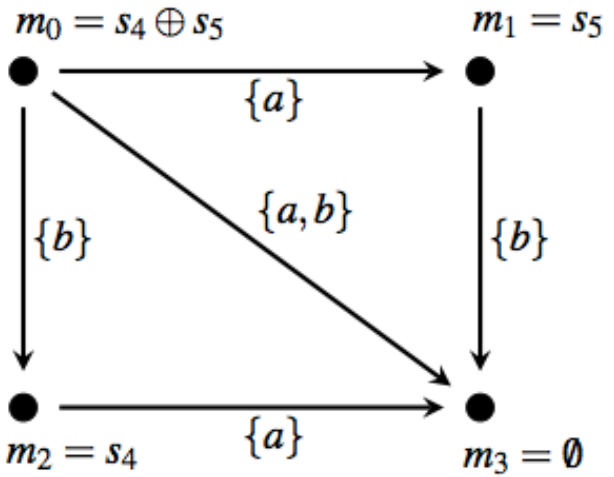
where $\longrightarrow_s \subseteq \mathscr{M}_{fin}(S) \times \mathscr{M}_{fin}(A) \times \mathscr{M}_{fin}(S)$ is defined by $m \xrightarrow{B}_s m'$ iff there exists a step G such that $m[G\rangle m'$ and $B = l(G)$, and the labeling function l is extended to multisets of transitions in the obvious way: $l(G)(a) = \sum_{t_i \in dom(G).l(t_i)=a} G(t_i)$. □

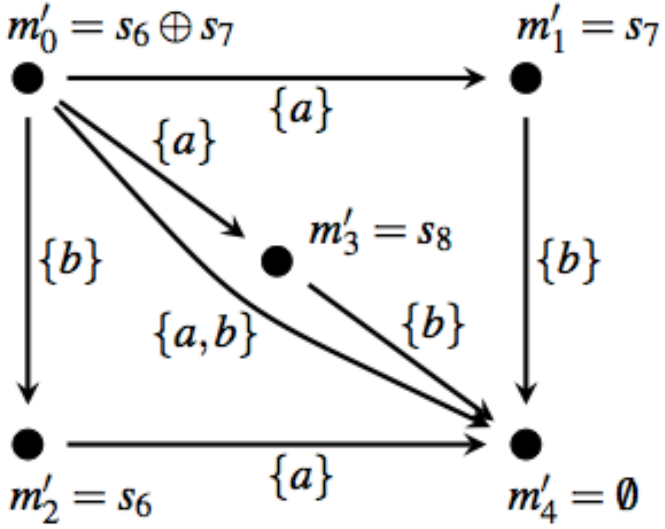
Proposition 3.9. (Fully concurrent) *For any P/T net system $N(m_0)$, its associated step marking graph $SMG(N(m_0))$ is fully concurrent.*

Examples


s_4
s_5
a
b


s_6
s_7
a
a
b
s_8
b

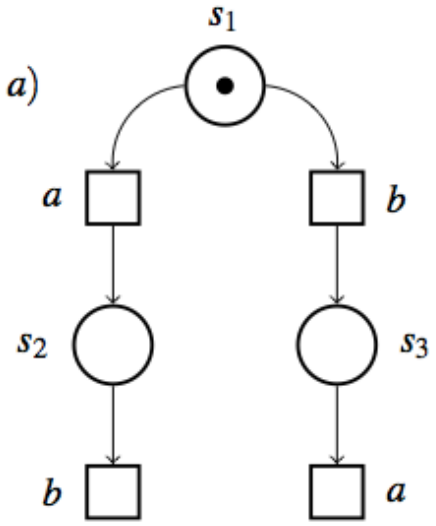

$m_0 = s_4 \oplus s_5$
$m_1 = s_5$
$\{a\}$
$\{b\}$
$\{a,b\}$
$\{b\}$
$m_2 = s_4$
$\{a\}$
$m_3 = \emptyset$

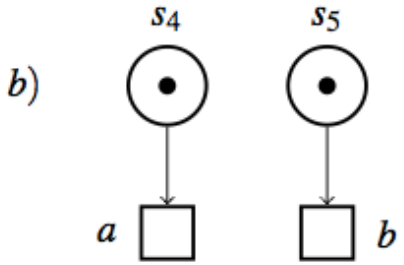

$m'_0 = s_6 \oplus s_7$
$m'_1 = s_7$
$\{a\}$
$\{a\}$
$m'_3 = s_8$
$\{b\}$
$\{b\}$
$\{a,b\}$
$\{b\}$
$m'_2 = s_6$
$\{a\}$
$m'_4 = \emptyset$

Step bisimilarity

Definition 3.21. (Step bisimilarity) Two P/T systems $N_1(m_1)$ and $N_2(m_2)$ are *step bisimilar* (denoted by $N_1(m_1) \sim_{step} N_2(m_2)$ or simply $m_1 \sim_{step} m_2$) if and only if there exists a strong bisimulation $R \subseteq [m_1\rangle \times [m_2\rangle$ over the STSs $SMG(N_1(m_1))$ and $SMG(N_2(m_2))$ such that $(m_1, m_2) \in R$. □

- The two nets in the previous slide are step bisimilar. The two below are not step bisimilar.





Properties

Proposition 3.10. *Given two net systems* $N_1(m_1)$ *and* $N_2(m_2)$, *if* $N_1(m_1) \sim_{step} N_2(m_2)$, *then* $N_1(m_1) \sim N_2(m_2)$.

- Step bisimilarity is also undecidable for finite P/T nets with at least two unbounded places.
- For BPP nets, the problem of checking step bisimilarity is decidable
- On bounded finite P/T nets, step bisimilarity is decidable, because the associated *SMG*s are finite-state.

Nonpermissive Petri nets (NP/T)

- The main feature is that a transition *t* enabled at *m* may be disabled at a larger marking *m'* (i.e., such that $m \subseteq m'$) because *m'* can contain additional tokens inhibiting the firability of *t*: Nonpermissiveness.
- Testable places: a place that can be tested for the precise number of tokens in it.
- Neg-set of a transition: set of places to be tested for absence of additional tokens, apart from those required by the pre-set.
- Enabling condition: all the tokens in the pre-set must be available and no additional token is present in the places of the neg-set.

NP/T nets

Definition 3.22. (Nonpermissive net) A *Nonpermissive P/T Petri net* (or NP/T net, for short) is a tuple $N = (S, D, A, T)$ where

- S is the countable set of *places*, ranged over by s (possibly indexed),
- $D \subseteq S$ is the set of *testable* places,
- $A \subseteq Lab$ is the countable set of *labels*, ranged over by ℓ (possibly indexed), and
- $T \subseteq (\mathscr{M}_{fin}(S) \setminus \{\emptyset\}) \times \mathscr{P}_{fin}(D) \times A \times \mathscr{M}_{fin}(S)$ is the countable set of *transitions*, ranged over by t (possibly indexed), such that for each $\ell \in A$ there exists a transition $t \in T$ of the form (m, I, ℓ, m').

Given a transition $t = (m, I, \ell, m')$, we use the notation $({}^\bullet t, {}^\circ t) \xrightarrow{l(t)} t^\bullet$.

- ${}^\bullet t$ to denote its *pre-set* m (which cannot be an empty multiset) of tokens to be consumed;
- ${}^\circ t$ to denote its *neg-set* I (which can be an empty set) of testable places where no further tokens can be present besides those required by ${}^\bullet t$;
- $l(t)$ for its *label* ℓ, and
- $t^\bullet$ to denote its *post-set* m' of tokens to be produced.

Firing of a transition

A transition t is enabled at m, written $m[t\rangle$, if

$\bullet t(s) \leq m(s)$ for all $s \in dom(\bullet t)$ (the tokens to be consumed are available, as for P/T nets) and, additionally,

$\bullet t(s) = m(s)$ for all $s \in \circ t$.

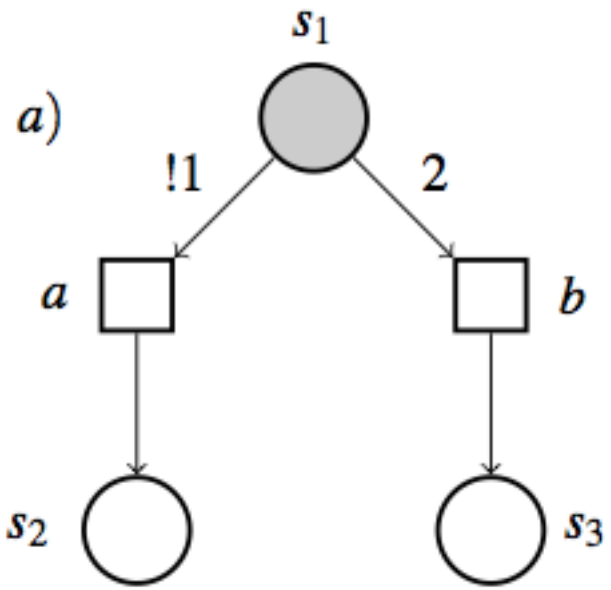
As a matter of fact, this additional constraint means that, for all $s \in \circ t$, the tokens in s for m are exactly as many as are required by t; note that if $\bullet t(s) = 0$, then the condition $m(s) = \bullet t(s) = 0$ corresponds to the classic inhibiting condition (test for absence of tokens in place s).

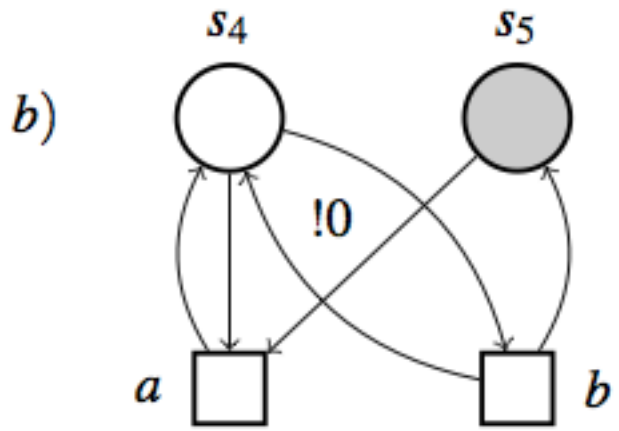
The firing of a transition t enabled at m produces the marking $\boldsymbol{m' = (m \ominus \bullet t) \oplus t\bullet}$, as for P/T nets; this is denoted $m[t\rangle m'$, as usual.

Drawing conventions

- A testable place is represented by a grey-colored circle.
- An arc connecting a testable place to a transition may be labeled not only by a number (1 is the default value, in case the number is omitted), but also by a number prefixed by the special symbol !.
- **If $s \in {\bullet}t$, $s \notin t{\circ}$**, then the arc from s to t is normally labeled with the number $\bullet t(s)$, even if $s \in D$.
- **If $s \in {\circ}t \cap dom(\bullet t)$**, then the arc from s to t is labeled by $!\bullet t(s)$, i.e., the number $\bullet t(s)$ is prefixed by the symbol ! to express that *exactly* that number of tokens are to be present in the current marking m to enable t.
- **If $s \in {\circ}t$ but $s \notin dom(\bullet t)$**, then a **!0-labeled arc** is included from s to t, meaning that no tokens can be present in s in the current marking m to enable t.

Examples





Turing-completeness (1)

Finite NP/T nets are a Turing-complete formalism, as it is possible to represent any counter machine (CM, for short) as a finite NP/T net. We assume the reader is familiar with CMs [ER64, Min67], and so we give a short presentation of this computational model; a didactical presentation can be found in [GV15]. A CM M is a pair (I,n), where

$$I = \{(1 : I_1), \ldots, (m : I_m)\}$$

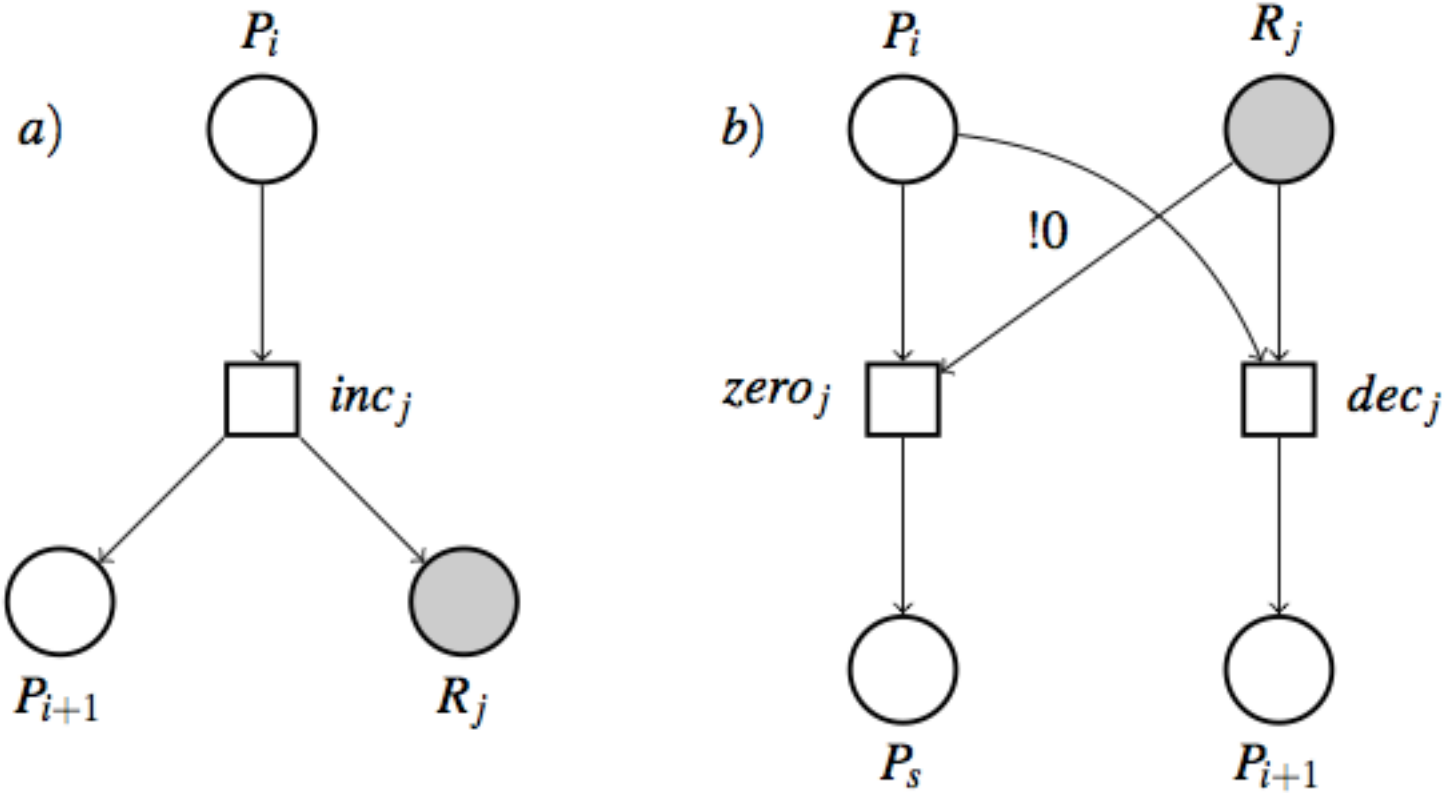
is the set of indexed instructions of M, with $|I| = m$, n is the number of registers r_j of M, and each instruction I_i is of two possible kinds:

- $I_i = Inc(r_j) \quad 1 \leq j \leq n$ (increment register r_j and go to the next instruction of index $i+1$);
- $I_i = DecJump(r_j, s)) \quad 1 \leq j \leq n$ (if r_j holds value 0, then jump to the instruction of index s; otherwise, decrement register r_j and go to the next instruction of index $i+1$).

Turing-completeness (2)

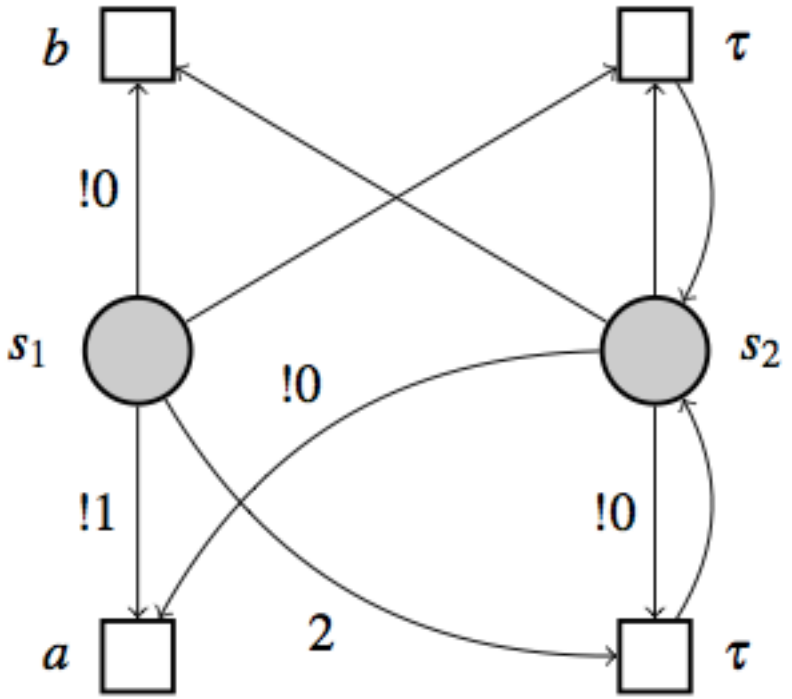
The finite NP/T net system $N(m_0) = (S, D, A, T, m_0)$ modeling the CM M is given as follows. The set of places $S = \{P_1, P_2, \ldots P_m, P_{last}, R_1, R_2, \ldots R_n\}$ is composed of a place P_i for each program-counter/instruction I_i, a place P_{last} to model the case in which the next instruction has index greater than m, and a place R_j for each register/counter r_j. The set D of testable places is $\{R_1, R_2, \ldots R_n\}$. The set A is $\{inc_j, dec_j, zero_j \mid 1 \leq j \leq n\}$. The set T of transitions models the operations of the CM M. In Figure 3.21(a) we show pictorially the transition for instruction $(i : Inc(r_j))$. The program counter flows from P_i to P_{i+1}, while one additional token is deposited into place R_j, to represent the increment of the register; if $i = m$, then P_{i+1} is actually P_{last}. In Figure 3.21(b), we show the net transitions for $(i : DecJump(r_j, s))$. If R_j contains at least one token, then the net transition labeled dec_j may be performed and instruction P_{i+1} is activated; if $i = m$, then P_{i+1} is actually P_{last}. Moreover, if R_j contains no tokens, the net transition labeled $zero_j$ can move the token from P_i to P_s; if $s > m$, then P_s is actually P_{last}. The initial marking m_0 puts one token into place P_1 and v_j tokens into place R_j (for $j = 1, \ldots n$), according to the initial values of the registers.

Turing-completeness (3)



Last Man Standing Problem

- Initially no token is present in s2.
- If one token is in s1, then only a can be performed.
- If n>1 tokens are present in s1, then only b will be performed eventually; first two tokens are removed from s1, producing one token in s2; then one token at a time is consumed from s1; when s1 is empty, b can be performed by s2.



Distributed Computability (1)

- Finite, dynamically acyclic, NP/T nets can solve the LMS problem, while CCS cannot solve the LMS problem
- NPL is the calculus corresponding to finite NP/T nets
- CCS and NPL are two Turing-complete calculi that are not equally expressive:
  - NPL can describe all finite NP/T nets and solve the LMS problem
  - CCS can describe many infinite P/T nets
- Hence, Turing-completeness is not an appropriate criterion for comparing the expressive power of concurrent languages

Distributed Computability (2)

- Therefore, what is the analog of Turing-completeness for concurrency? New definitions are necessary.
- Here we give our own proposal as a possible new foundation for ***distributed* computability theory – as a generalization of classic, *sequential* (or Turing) computability theor**y.
- A ***process*** is a semantic model, up to some behavioral equivalence. For instance, if the chosen models are labeled transition systems and the chosen equivalence is (weak completed) trace equivalence, then a process is nothing but a formal language; if, instead, the chosen models are Petri nets and the chosen equivalence is net isomorphism, then a process is a Petri net, up to net isomorphism.
- In other words, the notion of computable function on the natural numbers for sequential computation is to be replaced by a model with an associated equivalence relation, which we call a process, for concurrent computation.

Distributed Computability: A Hierarchy

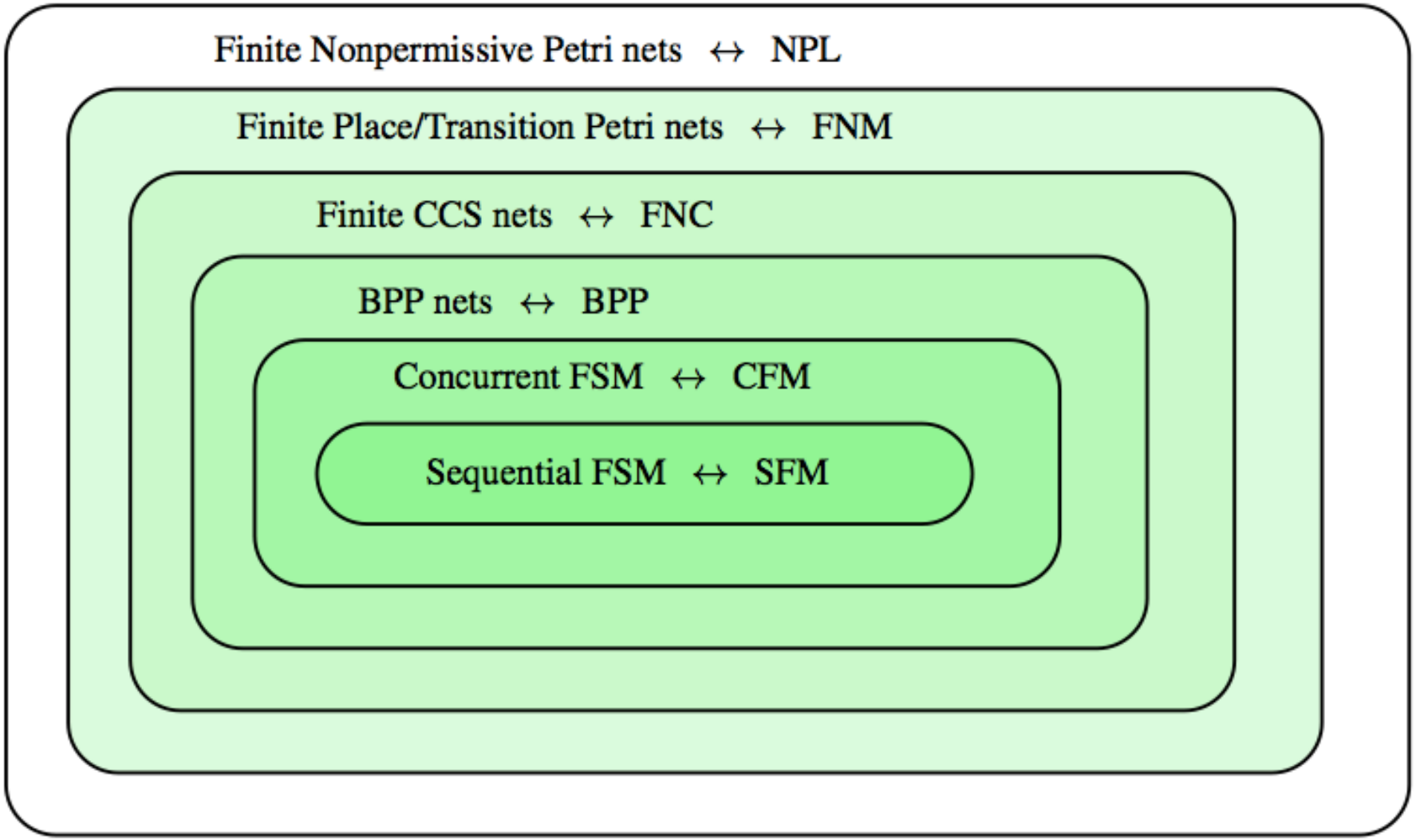


Fig. 1.5 The hierarchy of net classes and process algebras

Sequential Computability: A Hierarchy

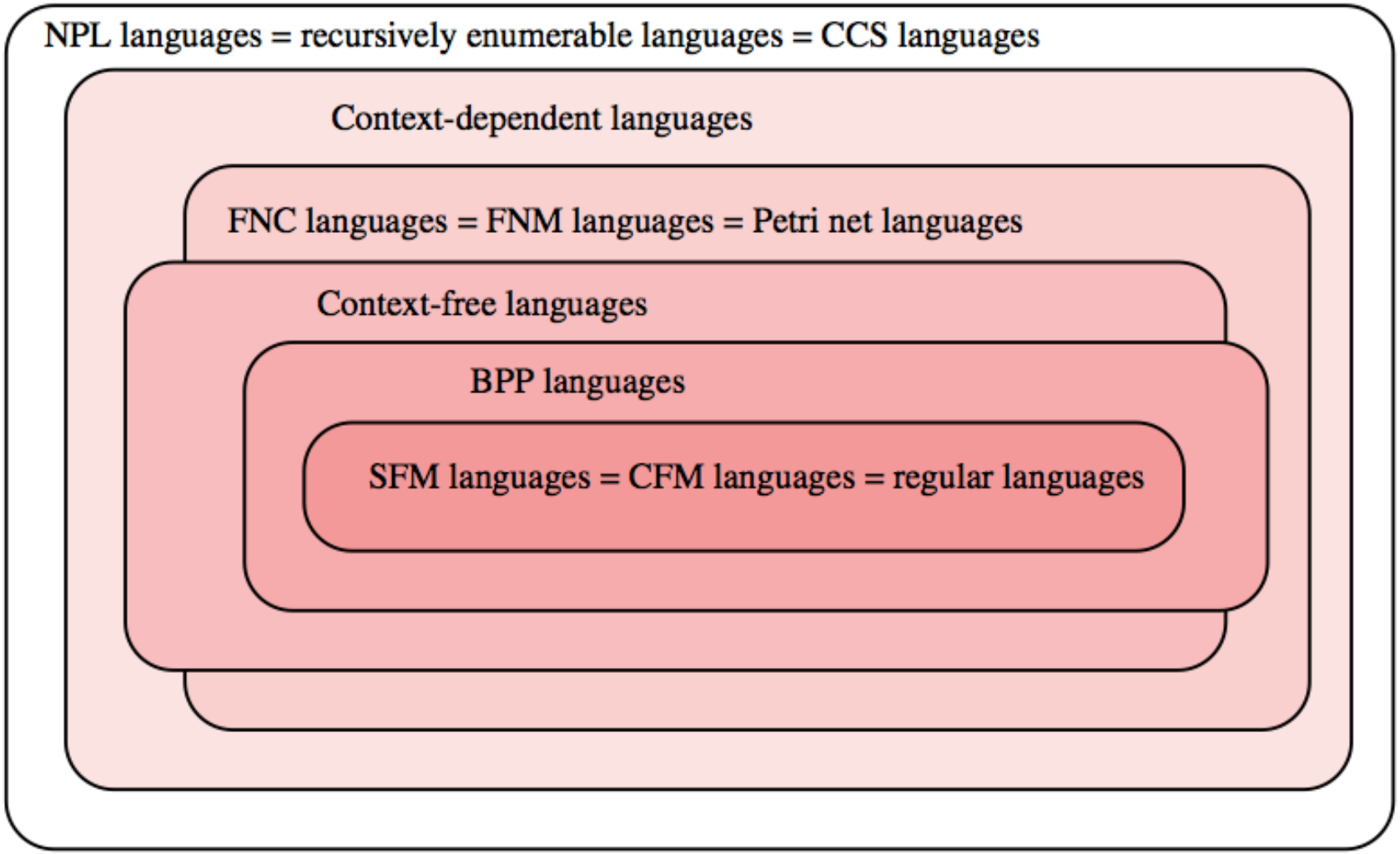


Fig. 1.9 The hierarchy of languages — interleaving semantics