Concurrency Theory

Winter 2025/26

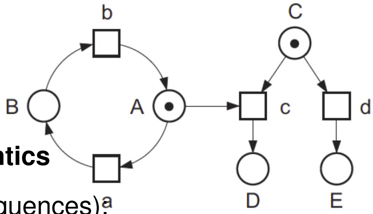
Lecture 16: True Concurrency Semantics of Petri Nets: Branching Processes

Thomas Noll, Peter Thiemann
Programming Languages Group
University of Freiburg

https://proglang.github.io/teaching/25ws/ct.html

Thomas Noll, Peter Thiemann

Winter 2025/26

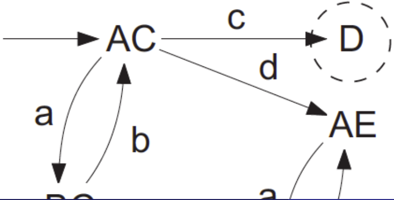


Interleaving semantics

Sequential runs (step sequences):

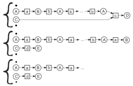
$$AC \xrightarrow{a} BC \xrightarrow{b} AC \xrightarrow{c} D$$
$$AC \xrightarrow{d} AE \xrightarrow{a} BE \xrightarrow{b} AE \xrightarrow{a} \cdots$$

Marking graph (LTS):

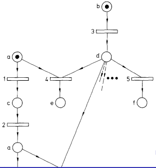


True concurrency semantics

Distributed runs (causal nets):



(Max.) Distributed process (occurrence net):



Outline of Lecture 16

Actions

A distributed run of a net is a partial order represented as a net whose basic building blocks are simple nets denoted as actions.

Definition (Action)

An action is a labelled net $A = (Q, \{v\}, G)$ with ${}^\bullet v \cap v^\bullet = \emptyset$ and ${}^\bullet v \cup v^\bullet = Q$.

Actions represent transition occurrences of elementary system nets. If A represents transition t, then the elements of Q are labelled with in- and output places of t, and v is labelled with t.

Causal Nets Informally

A causal net constitutes the basis of a “distributed” run.

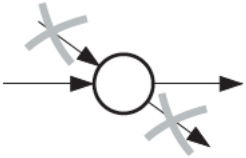
It is a (possibly infinite) elementary system net with the following properties:

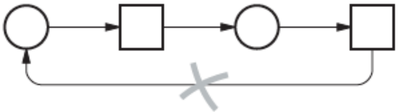
(1) It has no place branches: at most one arc ends or starts in a place.

(2) It is acyclic, i.e., no sequence of arcs forms a loop.

(3) Each sequence of arcs (flows) has a unique first element.

(4) The initial marking contains all places without incoming arcs.





Boundedness of Causal Nets

Lemma

Let $K = (Q, V, G, M_0)$ be a causal net. Then every step sequence

$$M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \ldots\ldots \xrightarrow{t_k} M_k$$

of K satisfies $M_j \cap t_k^\bullet = \emptyset$ for all $j \in \{0, \ldots, k-1\}$.

Theorem (Boundedness of causal nets)

Every causal net is one-bounded, i.e., in every marking every place will hold at most one token.

Proof.

Follows directly from the fact that the initial marking M_0 is one-bounded, and by Lemma 15.9. □

What Is a Distributed Run?

Definition (Distributed run)

A distributed run of a one-bounded elementary system net $N = (P, T, F, M_0)$ is

(1) a labelled causal net $K_N = (Q, V, G, M)$

(2) in which each transition $t \in V$ (with ${}^\bullet t$ and $t^\bullet$) is an action of N.

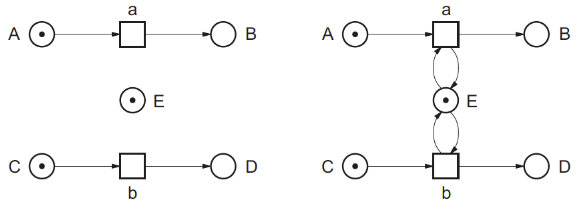
A distributed run K_N of N is complete if the marking $M = {}^\circ K_N$ represents the initial marking M_0 of N and the marking K_N° does not enable any transition in N.

If N is clear from the context, we just write K for K_N.

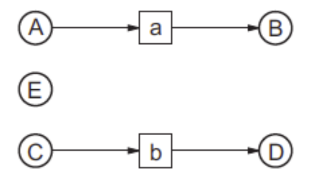
Causality Revisited

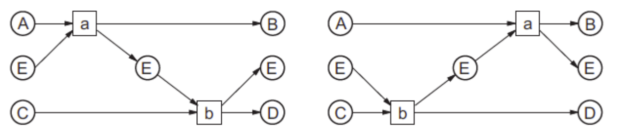
In contrast to sequential runs, distributed runs show the causal order of actions.

Example (cf. Example 14.20)



- Both nets have identical sequential runs (a occurs before b, or vice versa).
- But the left net only has the left distributed run below, the right net both ones:





Outline of Lecture 16

Net Homomorphisms I

Definition 16.1 (Net homomorphism)

A homomorphism from net $N_1 = (P_1, T_1, F_1, M_1)$ to net $N_2 = (P_2, T_2, F_2, M_2)$ is a mapping $h : P_1 \cup T_1 \to P_2 \cup T_2$ such that

Net Homomorphisms I

Definition 16.1 (Net homomorphism)

A **homomorphism** from net $N_1 = (P_1, T_1, F_1, M_1)$ to net $N_2 = (P_2, T_2, F_2, M_2)$ is a mapping $h : P_1 \cup T_1 \to P_2 \cup T_2$ such that

(1) $h(P_1) \subseteq P_2$ and $h(T_1) \subseteq T_2$,

Net Homomorphisms I

Definition 16.1 (Net homomorphism)

A **homomorphism** from net $N_1 = (P_1, T_1, F_1, M_1)$ to net $N_2 = (P_2, T_2, F_2, M_2)$ is a mapping $h : P_1 \cup T_1 \to P_2 \cup T_2$ such that

(1) $h(P_1) \subseteq P_2$ and $h(T_1) \subseteq T_2$,

(2) for every $t \in T_1$, the restriction of h to ${}^\bullet t$ is a bijection between ${}^\bullet t$ (in N_1) and ${}^\bullet h(t)$ (in N_2), and similarly for $t^\bullet$ and $h(t)^\bullet$, and

Definition 16.1 (Net homomorphism)

A homomorphism from net $N_1 = (P_1, T_1, F_1, M_1)$ to net $N_2 = (P_2, T_2, F_2, M_2)$ is a mapping $h : P_1 \cup T_1 \to P_2 \cup T_2$ such that

(1) $h(P_1) \subseteq P_2$ and $h(T_1) \subseteq T_2$,

(2) for every $t \in T_1$, the restriction of h to ${}^\bullet t$ is a bijection between ${}^\bullet t$ (in N_1) and ${}^\bullet h(t)$ (in N_2), and similarly for $t^\bullet$ and $h(t)^\bullet$, and

(3) the restriction of h to M_1 is a bijection between M_1 and M_2.[a]

[a] Due to one-boundedness, a marking M is a subset of the set P of places.

Net Homomorphisms I

Definition 16.1 (Net homomorphism)

A homomorphism from net $N_1 = (P_1, T_1, F_1, M_1)$ to net $N_2 = (P_2, T_2, F_2, M_2)$ is a mapping $h : P_1 \cup T_1 \to P_2 \cup T_2$ such that

(1) $h(P_1) \subseteq P_2$ and $h(T_1) \subseteq T_2$,

(2) for every $t \in T_1$, the restriction of h to ${}^\bullet t$ is a bijection between ${}^\bullet t$ (in N_1) and ${}^\bullet h(t)$ (in N_2), and similarly for $t^\bullet$ and $h(t)^\bullet$, and

(3) the restriction of h to M_1 is a bijection between M_1 and M_2.[a]

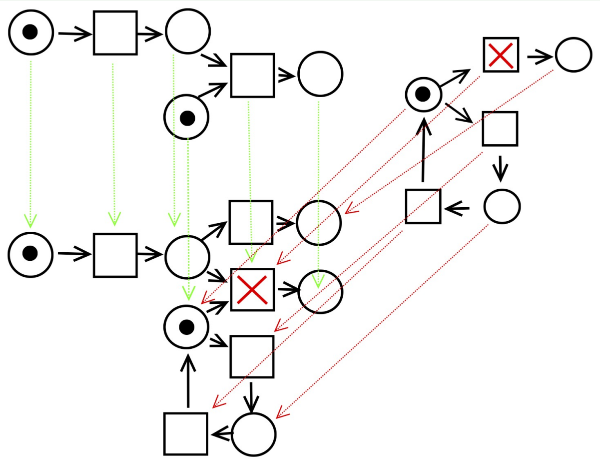
[a] Due to one-boundedness, a marking M is a subset of the set P of places.

Intuition

- A homomorphism is a mapping between nets that preserves
  (1) the kind of a node,
  (2) the neighborhood of transitions (but not necessarily that of places), and
  (3) the initial marking.
- A homomorphism from N_1 to N_2 means that N_1 can be folded onto a part of N_2, or, vice versa, that N_1 can be obtained by unfolding a part of N_2.

Net Homomorphisms II

Example 16.2 (Net homomorphism)



Homomorphic/non-homomorphic net mappings

Characterisation of Distributed Runs by Homomorphisms

Definition 16.3 (Distributed run; cf. Definition 15.14) (Best & Fernandez, 1988)

A distributed run of an elementary system net N is a pair (K, h) where K is a causal net and h is a homomorphism from K to N.

Characterisation of Distributed Runs by Homomorphisms

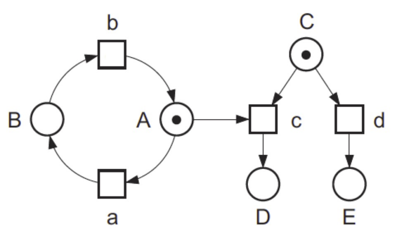
Definition 16.3 (Distributed run; cf. Definition 15.14) (Best & Fernandez, 1988)

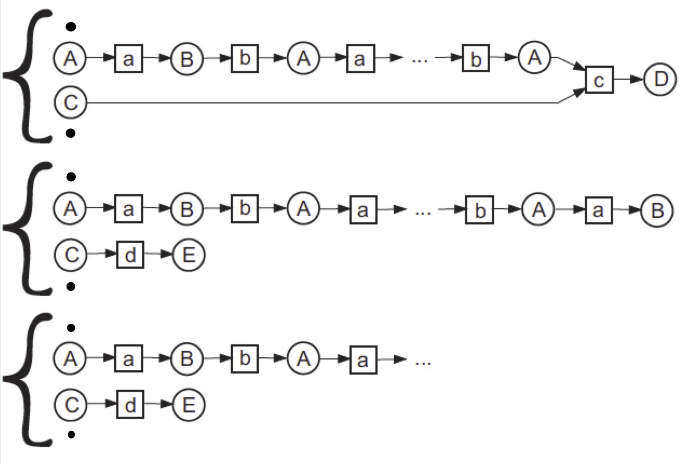
A distributed run of an elementary system net N is a pair (K, h) where K is a causal net and h is a homomorphism from K to N.

Intuition

- A distributed run (K, h) of N may be viewed as a net K whose places and transitions are labelled by places and transitions of N such that the labelling h forms a net homomorphism from K to N.
- Here, requirement (2) of Definition 16.1 ensures that K is composed of actions of N.
- Thus, Definitions 15.14 and 16.3 are equivalent.

Example 16.4 (cf. Example 15.17)





Two finite distributed runs (first complete, second incomplete) and the only infinite and complete distributed run of a net (homomorphisms given by labellings)

Outline of Lecture 16

Introduction

- Interleaving semantics of Petri nets = set of sequential runs.
  - a sequential run is a total ordering of transition occurrences

Introduction

- Interleaving semantics of Petri nets = set of sequential runs.
  - a sequential run is a total ordering of transition occurrences
- The set of all sequential runs can be represented by a marking graph.

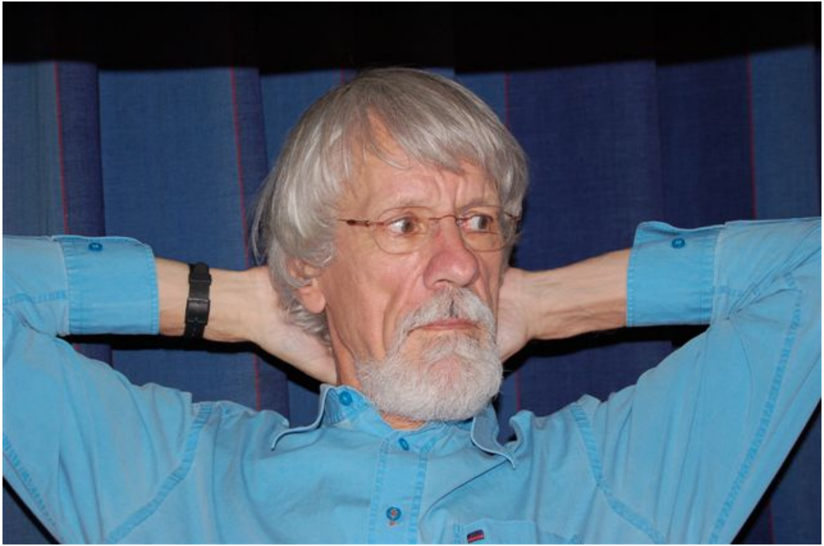
Introduction

- Interleaving semantics of Petri nets = set of sequential runs.
  - a sequential run is a total ordering of transition occurrences
- The set of all sequential runs can be represented by a marking graph.
- True concurrency semantics of Petri nets = set of distributed runs
  - a distributed run is an acyclic (causal) net which contains no choices
  - a distributed run gives a partial ordering of transition occurrences

Introduction

- Interleaving semantics of Petri nets = set of sequential runs.
  - a sequential run is a total ordering of transition occurrences
- The set of all sequential runs can be represented by a marking graph.
- True concurrency semantics of Petri nets = set of distributed runs
  - a distributed run is an acyclic (causal) net which contains no choices
  - a distributed run gives a partial ordering of transition occurrences
- Today: The set of all distributed runs can be represented by a specific branching process, the unfolding of the net.

Joost Engelfriet



Joost Engelfriet, Leiden University (NL), retired

Branching Process: Preamble

- A branching process depicts a set of distributed runs.

[1] In net jargon, a choice is called a conflict.

Branching Process: Preamble

- A branching process depicts a set of distributed runs.
- It explicitly represents each possible resolution of each choice[1].

[1] In net jargon, a choice is called a conflict.

Branching Process: Preamble

- A branching process depicts a set of distributed runs.
- It explicitly represents each possible resolution of each choice[1].
- It is a partial ordering with conflicts of transition occurrences.

[1] In net jargon, a choice is called a conflict.

Branching Process: Preamble

- A branching process depicts a set of distributed runs.
- It explicitly represents each possible resolution of each choice[1].
- It is a partial ordering with conflicts of transition occurrences.
- The true concurrency semantics of a net is a specific branching process, called unfolding.

[1] In net jargon, a choice is called a conflict.

Branching Process: Preamble

- A branching process depicts a set of distributed runs.
- It explicitly represents each possible resolution of each choice[1].
- It is a partial ordering with conflicts of transition occurrences.
- The true concurrency semantics of a net is a specific branching process, called unfolding.
- The unfolding is the true concurrency counterpart of a marking graph.

[1] In net jargon, a choice is called a conflict.

Branching Process: Preamble

- A branching process depicts a set of distributed runs.
- It explicitly represents each possible resolution of each choice[1].
- It is a partial ordering with conflicts of transition occurrences.
- The true concurrency semantics of a net is a specific branching process, called unfolding.
- The unfolding is the true concurrency counterpart of a marking graph.
- It is the greatest branching process in a complete lattice.

[1] In net jargon, a choice is called a conflict.

Outline of Lecture 16

Conflicts I

- A distributed run is based on a causal net.
- A branching process is based on an occurrence net.
- Main difference: the presence of conflicts (choices).

Conflicts I

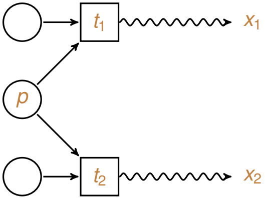
- A distributed run is based on a causal net.
- A branching process is based on an occurrence net.
- Main difference: the presence of conflicts (choices).

Definition 16.5 (Conflict)

Let $N = (P, T, F, M_0)$ be an elementary system net. Nodes $x_1, x_2 \in P \cup T$ are in conflict, denoted $x_1 \# x_2$, if there exist distinct transitions $t_1, t_2 \in T$ such that

$$^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset \quad \text{and} \quad (t_1, x_1) \in F^* \quad \text{and} \quad (t_2, x_2) \in F^*.$$

Node $x \in P \cup T$ is in self-conflict whenever $x \# x$.



Conflicts I

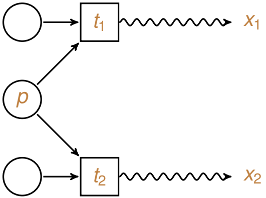
- A distributed run is based on a causal net.
- A branching process is based on an occurrence net.
- Main difference: the presence of conflicts (choices).

Definition 16.5 (Conflict)

Let $N = (P, T, F, M_0)$ be an elementary system net. Nodes $x_1, x_2 \in P \cup T$ are in conflict, denoted $x_1 \# x_2$, if there exist distinct transitions $t_1, t_2 \in T$ such that

$$^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset \quad \text{and} \quad (t_1, x_1) \in F^* \quad \text{and} \quad (t_2, x_2) \in F^*.$$

Node $x \in P \cup T$ is in self-conflict whenever $x \# x$.



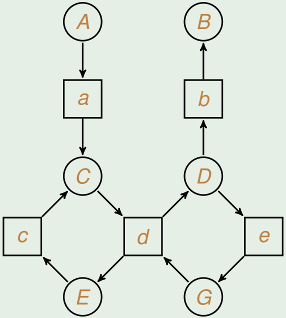
Notes:

- Conflicts are structural properties of nets, and as such independent of concrete markings.
- In a causal net, $\# = \emptyset$ as $^\bullet t_1 \cap {}^\bullet t_2 = \emptyset$ for any two distinct transitions t_1 and t_2 (since there is no place branching).

Conflicts II

Recall: $x_1 \# x_2$ if $\exists t_1, t_2 \in T : {}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset, (t_1, x_1) \in F^*, (t_2, x_2) \in F^*$.

Example 16.6

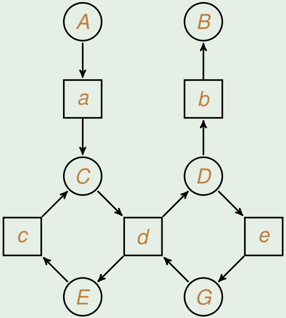


Two threads (left/right sub-net) that cycle around and synchronize from time to time (via transition d).
Right thread can "opt-out" (via transition b), which leads to a global deadlock.

Conflicts II

Recall: $x_1 \# x_2$ if $\exists t_1, t_2 \in T : {}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset, (t_1, x_1) \in F^*, (t_2, x_2) \in F^*$.

Example 16.6



Two threads (left/right sub-net) that cycle around and synchronize from time to time (via transition d).
Right thread can "opt-out" (via transition b), which leads to a global deadlock.

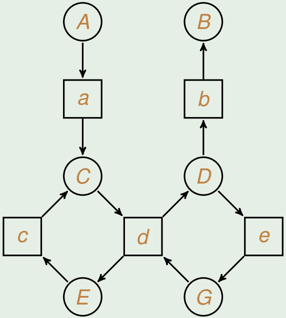
Some conflicts:

$b \# e$: for $t_1 = b$ and $t_2 = e$, ${}^\bullet b \cap {}^\bullet e = \{D\} \neq \emptyset$, $(b, b) \in F^*$, $(e, e) \in F^*$

Conflicts II

Recall: $x_1 \# x_2$ if $\exists t_1, t_2 \in T : {}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset, (t_1, x_1) \in F^*, (t_2, x_2) \in F^*$.

Example 16.6



Two threads (left/right sub-net) that cycle around and synchronize from time to time (via transition d).
Right thread can "opt-out" (via transition b), which leads to a global deadlock.

Some conflicts:

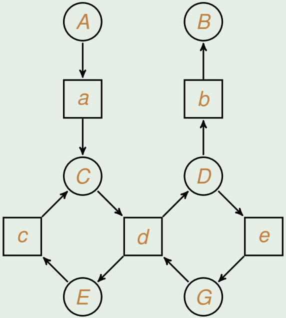
$b\#e$: for $t_1 = b$ and $t_2 = e$, ${}^\bullet b \cap {}^\bullet e = \{D\} \neq \emptyset$, $(b, b) \in F^*$, $(e, e) \in F^*$

$b\#c$: for $t_1 = b$ and $t_2 = e$, ${}^\bullet b \cap {}^\bullet e = \{D\} \neq \emptyset$, $(b, b) \in F^*$, $(e, c) \in F^*$

Conflicts II

Recall: $x_1 \# x_2$ if $\exists t_1, t_2 \in T : {}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset, (t_1, x_1) \in F^*, (t_2, x_2) \in F^*$.

Example 16.6



Two threads (left/right sub-net) that cycle around and synchronize from time to time (via transition d).
Right thread can "opt-out" (via transition b), which leads to a global deadlock.

Some conflicts:

$b \# e$: for $t_1 = b$ and $t_2 = e$, ${}^\bullet b \cap {}^\bullet e = \{D\} \neq \emptyset$, $(b, b) \in F^*$, $(e, e) \in F^*$

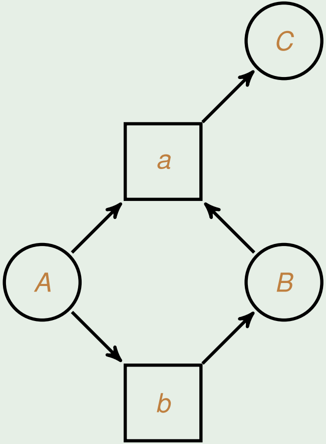
$b \# c$: for $t_1 = b$ and $t_2 = e$, ${}^\bullet b \cap {}^\bullet e = \{D\} \neq \emptyset$, $(b, b) \in F^*$, $(e, c) \in F^*$

$B \# G$: for $t_1 = b$ and $t_2 = e$, ${}^\bullet b \cap {}^\bullet e = \{D\} \neq \emptyset$, $(b, B) \in F^*$, $(e, G) \in F^*$

Conflicts III

Recall: $x_1 \# x_2$ if $\exists t_1, t_2 \in T : {}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset, (t_1, x_1) \in F^*, (t_2, x_2) \in F^*$.

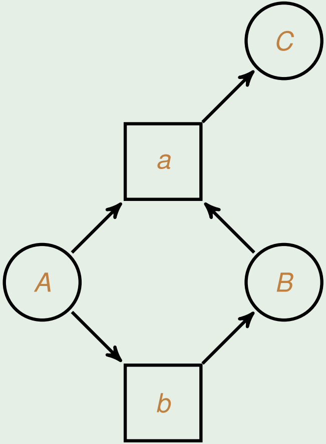
Example 16.7



Conflicts III

Recall: $x_1 \# x_2$ if $\exists t_1, t_2 \in T : {}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset, (t_1, x_1) \in F^*, (t_2, x_2) \in F^*$.

Example 16.7



A self-conflict:

$$a \# a$$

as for $t_1 = a$ and $t_2 = b$, ${}^\bullet a \cap {}^\bullet b = \{A\} \neq \emptyset$, $(a, a) \in F^*$, $(b, a) \in F^*$

Outline of Lecture 16

Occurrence Nets I

Definition 16.8 (Occurrence net)

A net $K = (Q, V, G, M_0)$ is an occurrence net if

(1) for each place $q \in Q$, $|{}^\bullet q| \leq 1$,

(2) the transitive closure G^+ of G is irreflexive,

(3) for each node $x \in Q \cup V$, the set $\{y \mid (y, x) \in G^+\}$ is finite,

(4) no transition $v \in V$ is in self-conflict, and

(5) $M_0 = {}^\circ K$.[a]

[a] Reminder: ${}^\circ K = \{q \in Q \mid {}^\bullet q = \emptyset\}$.

Definition 16.8 (Occurrence net)

A net $K = (Q, V, G, M_0)$ is an occurrence net if

(1) for each place $q \in Q$, $|{}^\bullet q| \leq 1$,

(2) the transitive closure G^+ of G is irreflexive,

(3) for each node $x \in Q \cup V$, the set $\{y \mid (y, x) \in G^+\}$ is finite,

(4) no transition $v \in V$ is in self-conflict, and

(5) $M_0 = {}^\circ K$.[a]

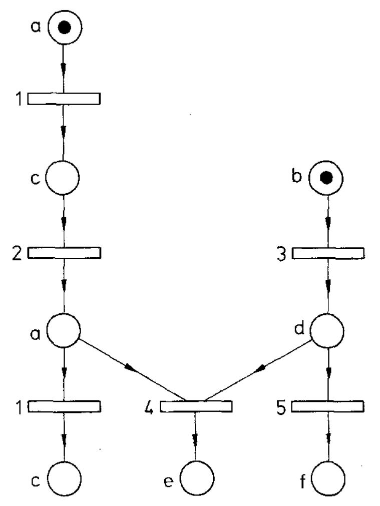
[a]Reminder: ${}^\circ K = \{q \in Q \mid {}^\bullet q = \emptyset\}$.

Remarks:

- In contrast to causal nets (Definition 15.8), occurrence nets additionally admit output (but still no input) branching for places.
- Since $\# = \emptyset$ in a causal net and each causal net by definition fulfils the remaining conditions, every causal net is also an occurrence net.

Occurrence Nets II

Example 16.9



An occurrence net (but not a causal net)

Theorem 16.10 (Boundedness of occurrence nets)

Every occurrence net is one-bounded, i.e., in every reachable marking every place will hold at most one token.

Boundedness of Occurrence Nets

Theorem 16.10 (Boundedness of occurrence nets)

Every occurrence net is one-bounded, i.e., in every reachable marking every place will hold at most one token.

Proof.

Similar to Theorem 15.10 (boundedness of causal nets). Note that input branching for places is also forbidden for occurrence nets. □

Outline of Lecture 16

Branching Processes I

Definition 16.11 (Branching process) (Engelfriet 1991)

A branching process of net N is a pair $B = (K, h)$ where $K = (Q, V, G, M_0)$ is an occurrence net and h a net homomorphism from K to N such that

$$\forall v, v' \in V : \left({}^\bullet v = {}^\bullet v' \text{ and } h(v) = h(v') \quad \text{implies} \quad v = v' \right) .$$

Definition 16.11 (Branching process) (Engelfriet 1991)

A branching process of net N is a pair $B = (K, h)$ where $K = (Q, V, G, M_0)$ is an occurrence net and h a net homomorphism from K to N such that

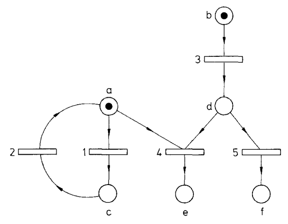
$$\forall v, v' \in V : \left({}^\bullet v = {}^\bullet v' \text{ and } h(v) = h(v') \quad \text{implies} \quad v = v'\right).$$

Remarks:

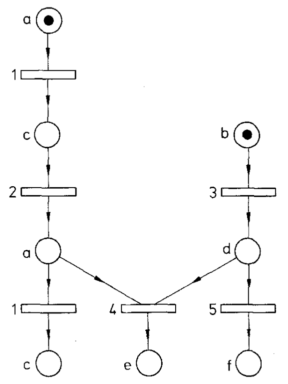
- The condition on h asserts that, in any particular situation, a transition of N can be chosen at most once in K.
- Note that every distributed run is also a branching process. The reverse does not hold.

Example 16.12

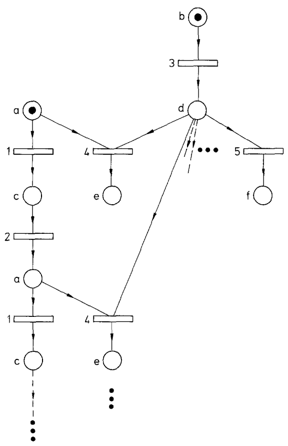
(Homomorphisms are given by labellings.)



Elementary system net



Finite branching process



Infin. branching process

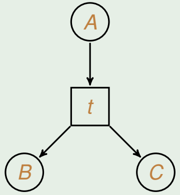
Branching Processes III

Definition (Branching process) (Engelfriet 1991)

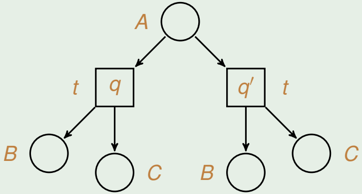
A branching process of net N is a pair $B = (K, h)$ where $K = (Q, V, G, M_0)$ is an occurrence net and h a net homomorphism from K to N such that

$$\forall v, v' \in V : ({}^\bullet v = {}^\bullet v' \text{ and } h(v) = h(v') \quad \text{implies} \quad v = v') \,.$$

Example 16.13



Net N



Occurrence net, but not a branching process of N
(${}^\bullet q = {}^\bullet q'$ and $h(q) = h(q')$ but $q \neq q'$)

Outline of Lecture 16

Definition 16.14 (Homomorphisms between branching processes)

Let $B_1 = (K_1, h_1)$ and $B_2 = (K_2, h_2)$ be two branching processes of net N.

- A homomorphism from B_1 to B_2 is a homomorphism h from K_1 to K_2 such that $h_2 \circ h = h_1$.
- An isomorphism is a bijective homomorphism.
- We write $B_1 \cong B_2$ if there exists an isomorphism between B_1 and B_2.

$$\begin{tikzcd} & N & \\ K_1 \arrow[ur, "h_1"] \arrow[rr, "h"] & & K_2 \arrow[ul, "h_2"'] \end{tikzcd}$$

Relating Branching Processes

Definition 16.14 (Homomorphisms between branching processes)

Let $B_1 = (K_1, h_1)$ and $B_2 = (K_2, h_2)$ be two branching processes of net N.

- A homomorphism from B_1 to B_2 is a homomorphism h from K_1 to K_2 such that $h_2 \circ h = h_1$.
- An isomorphism is a bijective homomorphism.
- We write $B_1 \cong B_2$ if there exists an isomorphism between B_1 and B_2.

$$
\begin{tikzcd}
 & N & \\
K_1 \arrow[ur, "h_1"] \arrow[rr, "h"] & & K_2 \arrow[ul, "h_2"']
\end{tikzcd}
$$

- Relation $\cong$ is an equivalence relation.
- Its equivalence classes are called isomorphism classes.
- The isomorphism quotient, i.e., the set of isomorphism classes of a branching process is denoted by $\mathbb{B}$.

Approximation of Branching Processes I

Definition 16.15 (Approximation)

Let B_1 and B_2 be two branching processes of net N. B_1 approximates B_2, denoted by $B_1 \sqsubseteq B_2$, if there is an injective homomorphism from B_1 to B_2.

Definition 16.15 (Approximation)

Let B_1 and B_2 be two branching processes of net N. B_1 approximates B_2, denoted by $B_1 \sqsubseteq B_2$, if there is an injective homomorphism from B_1 to B_2.

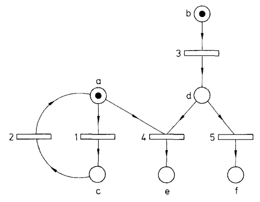
Remarks:

- B_1 approximates B_2 if every (partial) distributed run in B_1 is also contained in B_2. In other words, B_1 is isomorphic to an initial part of B_2.
- Being an approximation on branching processes is the analogue of being a prefix on sequences.
- Obviously, $\sqsubseteq$ is a preorder on branching processes.

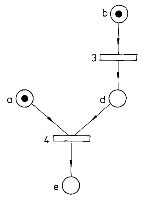
Approximation of Branching Processes II

Example 16.16 (cf. Example 16.12)

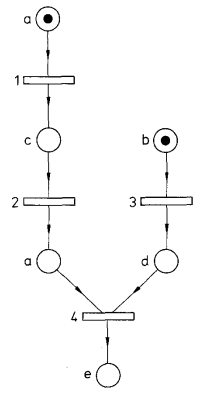
$$B_1 \sqsubseteq B_4,\ B_2 \sqsubseteq B_3 \sqsubseteq B_4,\ B_1 \not\sqsubseteq B_2, \ldots$$



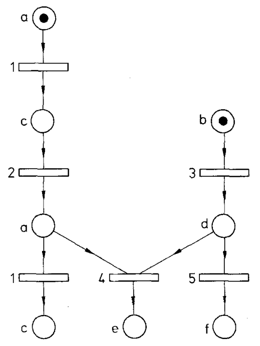
N



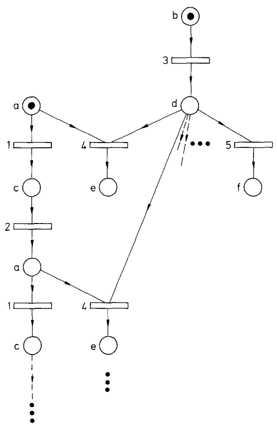
B_1



B_2



B_3



B_4

Definition (Approximation)

Let B_1 and B_2 be two branching processes of net N. B_1 approximates B_2, denoted by $B_1 \sqsubseteq B_2$, if there is an injective homomorphism from B_1 to B_2.

Lemma 16.17

Approximation is preserved by isomorphism: If B'_i is isomorphic to B_i (for $i \in \{1, 2\}$), then $B_1 \sqsubseteq B_2$ implies $B'_1 \sqsubseteq B'_2$. Thus, $\sqsubseteq$ can be extended to a partial order on the isomorphism quotient $\mathbb{B}$.

Proof.

omitted $\square$

Recall: a complete lattice is a partial order such that all subsets of its domain have LUBs and GLBs.

Theorem 16.18 (Engelfriet's branching process theorem)

$(\mathbb{B}, \sqsubseteq)$ *is a complete lattice.*

Proof.

see Joost Engelfriet: *Branching processes of Petri nets*, Acta Informatica 28, 1991 □

The True Concurrency Semantics of a Net I

Corollary 16.19 (Unfolding of a net)

Every net has a greatest (with respect to $\sqsubseteq$) branching process up to isomorphism, which is called its unfolding.

The True Concurrency Semantics of a Net I

Corollary 16.19 (Unfolding of a net)

Every net has a greatest (with respect to $\sqsubseteq$) branching process up to isomorphism, which is called its unfolding.

Definition 16.20 (True concurrency semantics)

Let N be a net, and let $B_{\mathsf{max}} = (K_{\mathsf{max}}, h_{\mathsf{max}})$ denote a representative of the isomorphism class of the greatest branching process of N. Then B_{max} is the true concurrency semantics of N.

The True Concurrency Semantics of a Net I

Corollary 16.19 (Unfolding of a net)

Every net has a greatest (with respect to $\sqsubseteq$) branching process up to isomorphism, which is called its unfolding.

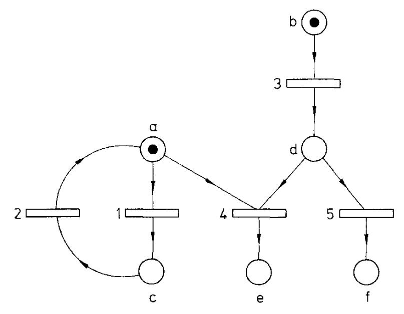
Definition 16.20 (True concurrency semantics)

Let N be a net, and let $B_{\mathrm{max}} = (K_{\mathrm{max}}, h_{\mathrm{max}})$ denote a representative of the isomorphism class of the greatest branching process of N. Then B_{max} is the true concurrency semantics of N.

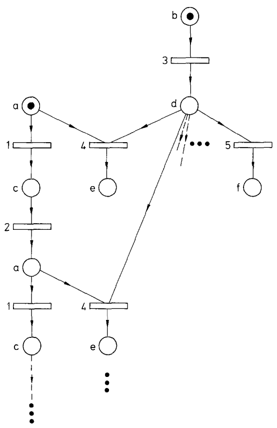
Recall: The interleaving semantics of a net is given by its marking graph.

The True Concurrency Semantics of a Net II

Example 16.21



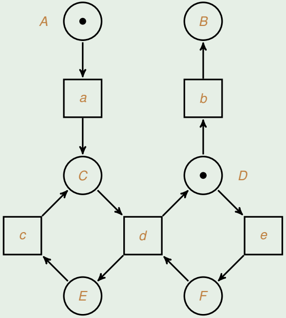
Elementary system net



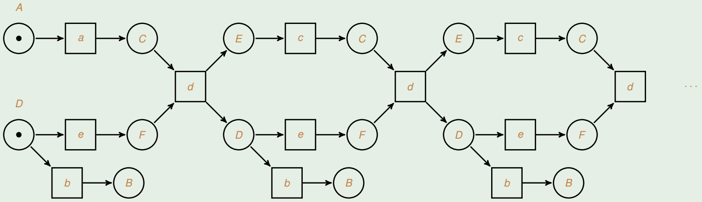
Its unfolding

Example 16.22 (cf. Example 16.6)

Elementary system net:



Its unfolding:



Outline of Lecture 16

Summary

- A branching process captures several distributed runs of a net N.

Summary

- A branching process captures several distributed runs of a net N.
- It is represented by a relaxed notion of causal net, the occurrence net.

Summary

- A branching process captures several distributed runs of a net N.
- It is represented by a relaxed notion of causal net, the occurrence net.
- N maps to its branching processes via homomorphisms.

Summary

- A branching process captures several distributed runs of a net N.
- It is represented by a relaxed notion of causal net, the occurrence net.
- N maps to its branching processes via homomorphisms.
- Approximation (denoted $\sqsubseteq$) is a preorder on branching processes.

Summary

- A branching process captures several distributed runs of a net N.
- It is represented by a relaxed notion of causal net, the occurrence net.
- N maps to its branching processes via homomorphisms.
- Approximation (denoted $\sqsubseteq$) is a preorder on branching processes.
- Isomorphism classes of branching processes with $\sqsubseteq$ form a complete lattice.

Summary

- A branching process captures several distributed runs of a net N.
- It is represented by a relaxed notion of causal net, the occurrence net.
- N maps to its branching processes via homomorphisms.
- Approximation (denoted $\sqsubseteq$) is a preorder on branching processes.
- Isomorphism classes of branching processes with $\sqsubseteq$ form a complete lattice.
- The true concurrency semantics of N is its greatest element (with respect to $\sqsubseteq$).

Summary

- A branching process captures several distributed runs of a net N.
- It is represented by a relaxed notion of causal net, the occurrence net.
- N maps to its branching processes via homomorphisms.
- Approximation (denoted $\sqsubseteq$) is a preorder on branching processes.
- Isomorphism classes of branching processes with $\sqsubseteq$ form a complete lattice.
- The true concurrency semantics of N is its greatest element (with respect to $\sqsubseteq$).
- [For one-bounded nets, it is possible to construct a finite approximating branching process ("McMillan prefix") that covers all reachable markings.]