Concurrency Theory

Lecture 20: McMillan Prefixes

Joost-Pieter Katoen and Thomas Noll

Lehrstuhl für Informatik 2
Software Modeling and Verification Group

http://moves.rwth-aachen.de/teaching/ws-1718/ct

January 8, 2018

Overview

Overview

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
- Partial-order semantics of Petri nets = set of distributed runs
  - a distributed run is an acyclic (causal) net which contains no choices
  - a distributed run is 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
- Partial-order semantics of Petri nets = set of distributed runs
  - a distributed run is an acyclic (causal) net which contains no choices
  - a distributed run is a partial ordering of transition occurrences
- Today: the set of all distributed runs can be represented by a finite prefix of the unfolding of the net.

Branching process: preamble

- A branching process represents a set of distributed runs

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

Branching process: preamble

- A branching process represents 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 represents a set of distributed runs
- It explicitly represents each possible resolution of each choice[1]
- It is an acyclic (occurrence) net containing choices.

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

Branching process: preamble

- A branching process represents a set of distributed runs
- It explicitly represents each possible resolution of each choice[1]
- It is an acyclic (occurrence) net containing choices.
- 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 represents a set of distributed runs
- It explicitly represents each possible resolution of each choice[1]
- It is an acyclic (occurrence) net containing choices.
- 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 represents a set of distributed runs
- It explicitly represents each possible resolution of each choice[1]
- It is an acyclic (occurrence) net containing choices.
- It is a partial ordering with conflicts of transition occurrences.
- The true concurrency semantics of a net is a specific branching process, called unfolding.
- A net 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 represents a set of distributed runs
- It explicitly represents each possible resolution of each choice[1]
- It is an acyclic (occurrence) net containing choices.
- It is a partial ordering with conflicts of transition occurrences.
- The true concurrency semantics of a net is a specific branching process, called unfolding.
- A net unfolding is the true concurrency counterpart of a marking graph.
- It is the unique maximal branching process in a complete lattice.

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

Branching process: preamble

- A branching process represents a set of distributed runs
- It explicitly represents each possible resolution of each choice[1]
- It is an acyclic (occurrence) net containing choices.
- It is a partial ordering with conflicts of transition occurrences.
- The true concurrency semantics of a net is a specific branching process, called unfolding.
- A net unfolding is the true concurrency counterpart of a marking graph.
- It is the unique maximal branching process in a complete lattice.
- The reachable markings of a 1-bounded net are covered by a finite prefix of this maximal branching process.

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

Overview

Conflicts

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

Conflicts

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

Conflict

Let $N = (P, T, F, M)$ be a net. Nodes x_1 and x_2 are in conflict, denoted $x_1 \# x_2$, if there exist distinct transitions $t_1, t_2 \in T$ such that ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \varnothing$ and $(t_1, x_1) \in F^*$ and $(t_2, x_2) \in F^*$.

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

Conflicts

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

Conflict

Let $N = (P, T, F, M)$ be a net. Nodes x_1 and x_2 are in conflict, denoted $x_1 \# x_2$, if there exist distinct transitions $t_1, t_2 \in T$ such that ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \varnothing$ and $(t_1, x_1) \in F^*$ and $(t_2, x_2) \in F^*$.

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

Note that in a causal net $\# = \varnothing$ as ${}^\bullet t_1 \cap {}^\bullet t_2 = \varnothing$ for any two distinct transitions t_1 and t_2.

Occurrence net

Occurrence net

A net $K = (Q, V, G, M)$ is an occurrence net iff:

1. for each $q \in Q$, $|{}^{\bullet}q| \leqslant 1$
2. the transitive closure G^+ of G is irreflexive
3. for each node $x \in Q \cup V$ we have $\{ y \mid (y, x) \in G^+ \}$ is finite
4. no transition $v \in V$ is in self-conflict
5. $M_0 = {}^{\circ}K \ = \ \{ q \in Q \mid \ {}^{\bullet}q = \varnothing \}$.

Occurrence net

Occurrence net

A net $K = (Q, V, G, M)$ is an occurrence net iff:

1. for each $q \in Q$, $|^{\bullet}q| \leqslant 1$
2. the transitive closure G^+ of G is irreflexive
3. for each node $x \in Q \cup V$ we have $\{ y \mid (y, x) \in G^+ \}$ is finite
4. no transition $v \in V$ is in self-conflict
5. $M_0 = {}^{\circ}K = \{ q \in Q \mid {}^{\bullet}q = \varnothing \}$.

Remark

Since $\# = \varnothing$ in a causal net, and each causal net fulfils the remaining conditions, every causal net is an occurrence net.

Branching process

Branching process [Engelfriet 1991]

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

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

Branching process

Branching process **[Engelfriet 1991]**

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

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

Every distributed run is a branching process. The reverse is not true.

Branching process

Branching process [Engelfriet 1991]

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

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

Every distributed run is a branching process. The reverse is not true.

Examples

On the black board.

Overview

Relating branching processes

Homomorphisms and isomorphisms 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 $h_2 \circ h = h_1$.[2]

[2] The composition of two net homomorphisms is a net homomorphism.

Relating branching processes

Homomorphisms and isomorphisms 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 $h_2 \circ h = h_1$.[2]

An isomorphism is a bijective homomorphism. B_1 and B_2 are isomorphic if there is an isomorphism from B_1 to B_2.

[2] The composition of two net homomorphisms is a net homomorphism.

Relating branching processes

Homomorphisms and isomorphisms 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 $h_2 \circ h = h_1$.[2]

An isomorphism is a bijective homomorphism. B_1 and B_2 are isomorphic if there is an isomorphism from B_1 to B_2.

Being isomorphic is an equivalence relation. Its equivalence classes are called isomorphism classes.

[2] The composition of two net homomorphisms is a net homomorphism.

Approximation of branching processes

Approximation

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

Approximation of branching processes

Approximation

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

Intuition

B_1 approximates B_2 whenever 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.

Approximation of branching processes

Approximation of branching processes

Approximation

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

Approximation of branching processes

Approximation

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

Lemma

Approximation is preserved by isomorphism: if B'_i is isomorphic to B_i (for $i = 1, 2$), then $B_1 \sqsubseteq B_2$ implies $B'_1 \sqsubseteq B'_2$. Thus, $\sqsubseteq$ can be extended to a partial order on isomorphism classes (of branching processes).

Approximation of branching processes

Approximation

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

Lemma

Approximation is preserved by isomorphism: if B_i' is isomorphic to B_i (for $i = 1, 2$), then $B_1 \sqsubseteq B_2$ implies $B_1' \sqsubseteq B_2'$. Thus, $\sqsubseteq$ can be extended to a partial order on isomorphism classes (of branching processes).

Proof.

Home exercise. Basically juggling with homomorphisms. □

Engelfriet's theorem

Engelfriet's theorem

Engelfriet's branching process theorem

The set of isomorphism classes of branching processes of net N is a complete lattice with respect to the approximation relation $\sqsubseteq$. Formally, $(\mathbb{B}, \sqsubseteq)$ is a complete partial order, where $\mathbb{B}$ is the set of isomorphism classes of branching processes.

Engelfriet's theorem

Engelfriet's branching process theorem

The set of isomorphism classes of branching processes of net N is a complete lattice with respect to the approximation relation $\sqsubseteq$. Formally, $(\mathbb{B}, \sqsubseteq)$ is a complete partial order, where $\mathbb{B}$ is the set of isomorphism classes of branching processes.

Complete lattice

Recall that a complete lattice is a partial order $(\mathbb{B}, \sqsubseteq)$ such that all subsets of $\mathbb{B}$ have LUBs and GLBs.

The true concurrency semantics of a net

Corollary: the unfolding of a net

Every one-bounded net has a unique maximal (with respect to $\sqsubseteq$) branching process up to isomorphism. This is called the unfolding or true concurrency semantics of net N.

The true concurrency semantics of a net

Corollary: the unfolding of a net

Every one-bounded net has a unique maximal (with respect to $\sqsubseteq$) branching process up to isomorphism. This is called the unfolding or true concurrency semantics of net N.

We denote by $B_{\mathsf{max}} = ((P_{\mathsf{max}}, T_{\mathsf{max}}, F_{\mathsf{max}}), h_{\mathsf{max}})$ a representative of the isomorphism class of the maximal branching process of N.

The true concurrency semantics of a net

Corollary: the unfolding of a net

Every one-bounded net has a unique maximal (with respect to $\sqsubseteq$) branching process up to isomorphism. This is called the unfolding or true concurrency semantics of net N.

We denote by $B_{\mathrm{max}} = ((P_{\mathrm{max}}, T_{\mathrm{max}}, F_{\mathrm{max}}), h_{\mathrm{max}})$ a representative of the isomorphism class of the maximal branching process of N.

Example

On the black board.

The true concurrency semantics of Petri nets

The true concurrency semantics of a Petri net is given by its unfolding.

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

Overview

Finite prefix

The maximal branching process under $\sqsubseteq$ may be infinite.

Finite prefix

The maximal branching process under $\sqsubseteq$ may be infinite.

Prefix of maximal branching process

Branching process $B = (P, T, F, M_0)$ is a prefix of $B_{\max}$ if $B \sqsubseteq B_{\max}$ and $P \subseteq P_{\max}$ and $T \subseteq T_{\max}$. B is finite whenever P and T are finite.

Finite prefix

The maximal branching process under $\sqsubseteq$ may be infinite.

Prefix of maximal branching process

Branching process $B = (P, T, F, M_0)$ is a prefix of B_{max} if $B \sqsubseteq B_{\text{max}}$ and $P \subseteq P_{\text{max}}$ and $T \subseteq T_{\text{max}}$. B is finite whenever P and T are finite.

Finite prefix existence theorem [McMillan, 1992]

For every finite one-bounded net N, there exists a finite prefix B_{fin} of B_{max} that covers all reachable markings of N. The size of the finite prefix can maximally be exponential in the size of N.

Finite prefix

The maximal branching process under $\sqsubseteq$ may be infinite.

Prefix of maximal branching process

Branching process $B = (P, T, F, M_0)$ is a prefix of B_{max} if $B \sqsubseteq B_{\text{max}}$ and $P \subseteq P_{\text{max}}$ and $T \subseteq T_{\text{max}}$. B is finite whenever P and T are finite.

Finite prefix existence theorem **[McMillan, 1992]**

For every finite one-bounded net N, there exists a finite prefix B_{fin} of B_{max} that covers all reachable markings of N. The size of the finite prefix can maximally be exponential in the size of N.

Proof.

Follows directly from two facts:

1. Every reachable marking is represented by some cut of B_{max}, and
2. The set of reachable markings of a finite one-bounded net is finite.

□

Configurations

Configurations

Configurations

Let $K = (Q, V, G, M_0)$ be an occurrence net, $\prec = G^+$ and $\preceq = G^*$.

Configurations

Configurations

Let $K = (Q, V, G, M_0)$ be an occurrence net, $\prec = G^+$ and $\preceq = G^*$.
The set $C \subseteq V$ is a configuration of K whenever:

1. $x \in C$ implies $y \in C$, for all $y \preceq x$ (downward-closed wrt. $\preceq$)
2. $\forall x, y \in C. \neg(x \# y)$ (conflict-free)

Configurations

Configurations

Let $K = (Q, V, G, M_0)$ be an occurrence net, $\prec = G^+$ and $\preceq = G^*$.
The set $C \subseteq V$ is a configuration of K whenever:

1. $x \in C$ implies $y \in C$, for all $y \preceq x$ (downward-closed wrt. $\preceq$)
2. $\forall x, y \in C. \neg(x \# y)$ (conflict-free)

Intuition and examples

A configuration can be seen as the set of transitions that have occurred so far in a distributed run.

Configurations

Configurations

Let $K = (Q, V, G, M_0)$ be an occurrence net, $\prec = G^+$ and $\preceq = G^*$.
The set $C \subseteq V$ is a configuration of K whenever:

1. $x \in C$ implies $y \in C$, for all $y \preceq x$ (downward-closed wrt. $\preceq$)
2. $\forall x, y \in C. \neg(x \# y)$ (conflict-free)

Intuition and examples

A configuration can be seen as the set of transitions that have occurred so far in a distributed run. Examples on the black board.

Configurations

Configurations

Let $K = (Q, V, G, M_0)$ be an occurrence net, $\prec = G^+$ and $\preceq = G^*$.
The set $C \subseteq V$ is a configuration of K whenever:

1. $x \in C$ implies $y \in C$, for all $y \preceq x$ (downward-closed wrt. $\preceq$)
2. $\forall x, y \in C. \neg(x \# y)$ (conflict-free)

Intuition and examples

A configuration can be seen as the set of transitions that have occurred so far in a distributed run. Examples on the black board.

Fact

For configuration C of $B_{\max}$ (of net N), and $x_1 \ldots x_n$ a linearisation of the transitions in C (respecting $\preceq$), the sequence $h_{\max}(x_1) \ldots h_{\max}(t_n)$ is a sequential run of the original net N.

Cuts

Cuts

Cuts

Let C be a finite configuration of a branching process $B = (K, h)$.

Cuts

Cuts

Let C be a finite configuration of a branching process $B = (K, h)$. Then:

$$Cut(C) \ = \ ({}^{\circ}K \cup C^{\bullet}) \setminus {}^{\bullet}C.$$

Cuts

Cuts

Let C be a finite configuration of a branching process $B = (K, h)$. Then:

$$Cut(C) \ = \ ({}^{\circ}K \cup C^{\bullet}) \setminus {}^{\bullet}C.$$

If B is a branching process of N, then $h(Cut(C))$ is a reachable marking of net N. We denote $h(Cut(C))$ by $M(C)$, the marking of configuration C.

Cuts

Cuts

Let C be a finite configuration of a branching process $B = (K, h)$. Then:

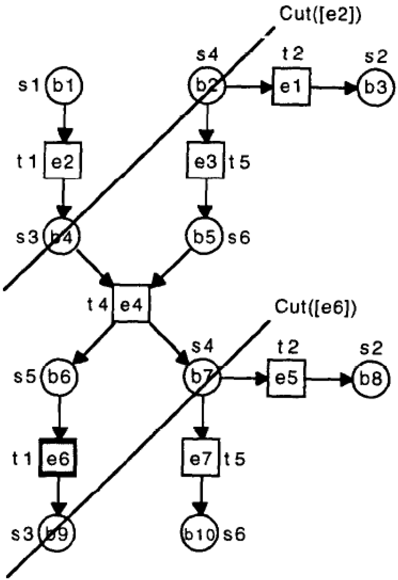
$$Cut(C) \;=\; ({}^{\circ}K \cup C^{\bullet}) \setminus {}^{\bullet}C.$$

If B is a branching process of N, then $h(Cut(C))$ is a reachable marking of net N. We denote $h(Cut(C))$ by $M(C)$, the marking of configuration C.

Intuition

Cuts correspond to markings reached by firing all transitions in a given finite configuration.

Example


Cut([e2])
s4
t2
s2
s1
b1
b2
e1
b3
t1
e2
e3
t5
s3
b4
b5
s6
t4
e4
Cut([e6])
s4
t2
s2
s5
b6
b7
e5
b8
t1
e6
e7
t5
s3
b9
b10
s6

Transition causes

Transition causes

Let $K = (Q, V, G)$ be an occurrence net and $v \in V$. The set $[v]$ of causes of v is defined by:

$$[v] \;=\; \{\, v' \in V \mid v' \preceq v \,\}.$$

(Recall that $\preceq$ denotes G^*, the reflexive and transitive closure of G.)

Transition causes

Transition causes

Let $K = (Q, V, G)$ be an occurrence net and $v \in V$. The set $[v]$ of causes of v is defined by:

$$[v] \;=\; \{\, v' \in V \mid v' \preceq v \,\}.$$

(Recall that $\preceq$ denotes G^*, the reflexive and transitive closure of G.)

Example

On the black board

Transition causes

Transition causes

Let $K = (Q, V, G)$ be an occurrence net and $v \in V$. The set $[v]$ of causes of v is defined by:

$$[v] \; = \; \{ \, v' \in V \mid v' \preceq v \, \}.$$

(Recall that $\preceq$ denotes G^*, the reflexive and transitive closure of G.)

Example

On the black board

Facts

1. For each v, $[v]$ is a finite configuration.
2. For every configuration C of K, either $v \notin C$ or $[v] \subseteq C$.

Cut-off event

Cut-off event

Cut-off event

Let $B_{\max} = ((P_{\max}, T_{\max}, G_{\max}), h_{\max})$. Transition $t \in T_{\max}$ is a cut-off transition if there exists a transition $t' \in T_{\max} \cup \{\bot\}$ such that:

$$\big|[t']\big| < \big|[t]\big| \text{ and } M([t]) = M([t']).$$

Cut-off event

Cut-off event

Let $B_{\max} = ((P_{\max}, T_{\max}, G_{\max}), h_{\max})$. Transition $t \in T_{\max}$ is a cut-off transition if there exists a transition $t' \in T_{\max} \cup \{ \perp \}$ such that:

$$\big|[t']\big| < \big|[t]\big| \text{ and } M([t]) = M([t']).$$

Dummy transition

Remark: $\perp$ is a dummy transition having no input places and ${}^\circ B_{\max}$ as output places, for which we let $[\perp] = \varnothing$. This yields that if $M([t]) = M_0$, then t is a cut-off transition.

Cut-off event

Cut-off event

Let $B_{\max} = ((P_{\max}, T_{\max}, G_{\max}), h_{\max})$. Transition $t \in T_{\max}$ is a cut-off transition if there exists a transition $t' \in T_{\max} \cup \{ \bot \}$ such that:

$$\big|[t']\big| < \big|[t]\big| \text{ and } M([t]) = M([t']).$$

Dummy transition

Remark: $\bot$ is a dummy transition having no input places and $^\circ B_{\max}$ as output places, for which we let $[\bot] = \varnothing$. This yields that if $M([t]) = M_0$, then t is a cut-off transition.

Fact

If $\big|[t']\big| < \big|[t]\big|$ and $M([t]) = M([t'])$, then the "continuations" of $B_{\max}$ from $Cut([t])$ and $Cut([t'])$ are isomorphic.

McMillan prefix

McMillan prefix

McMillan prefix

The McMillan prefix of one-bounded net N is the branching process B_{fin}, the unique prefix of $B_{\max}$ having T_{fin} as set of transitions satisfying for each $t \in T_{\max}$::

$$t \in T_{\text{fin}} \quad \text{iff} \quad \text{no transition } t' \prec t \text{ is a cut-off transition.}$$

Computing the McMillan prefix

Algorithm

1. Start with the empty branching process

Computing the McMillan prefix

Algorithm

1. Start with the empty branching process
2. Add transitions one at the time, in order of increasing size of their sets of causes

Computing the McMillan prefix

Algorithm

1. Start with the empty branching process
2. Add transitions one at the time, in order of increasing size of their sets of causes
3. On adding t, compare $M([t])$ with $M([t'])$ for each t' that was added before t

Computing the McMillan prefix

Algorithm

1. Start with the empty branching process
2. Add transitions one at the time, in order of increasing size of their sets of causes
3. On adding t, compare $M([t])$ with $M([t'])$ for each t' that was added before t
4. If $M([t]) = M([t'])$, then t is a cut-off transition, and its successors are not explored

Computing the McMillan prefix

Algorithm

1. Start with the empty branching process
2. Add transitions one at the time, in order of increasing size of their sets of causes
3. On adding t, compare $M([t])$ with $M([t'])$ for each t' that was added before t
4. If $M([t]) = M([t'])$, then t is a cut-off transition, and its successors are not explored
5. Terminate when no further transitions can be added.

Computing the McMillan prefix

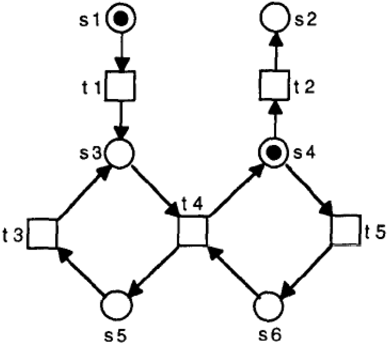
Algorithm

1. Start with the empty branching process
2. Add transitions one at the time, in order of increasing size of their sets of causes
3. On adding t, compare $M([t])$ with $M([t'])$ for each t' that was added before t
4. If $M([t]) = M([t'])$, then t is a cut-off transition, and its successors are not explored
5. Terminate when no further transitions can be added.

Remark

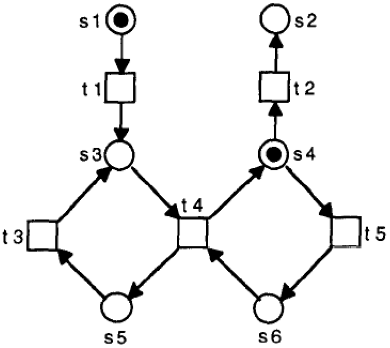
Termination is ensured by the finiteness of the number of reachable markings on N, as N is one-bounded.

Example net and one of its branching processes

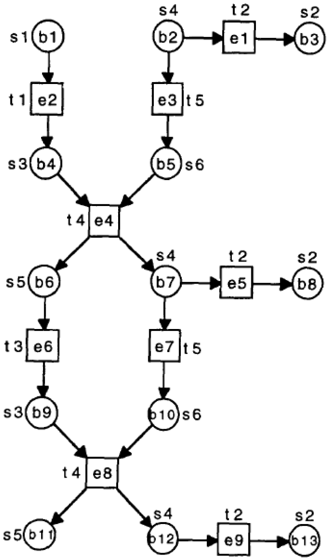


A sample one-bounded elementary system net

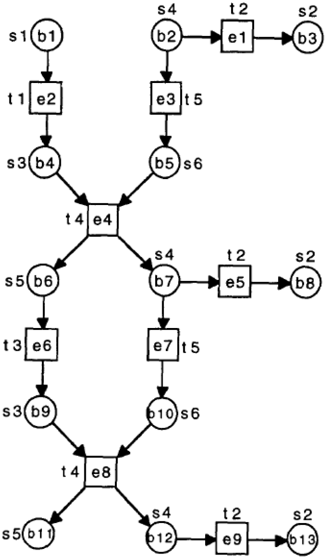
Example net and one of its branching processes



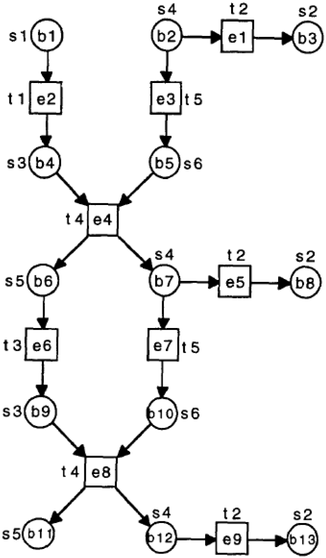
A sample one-bounded elementary system net

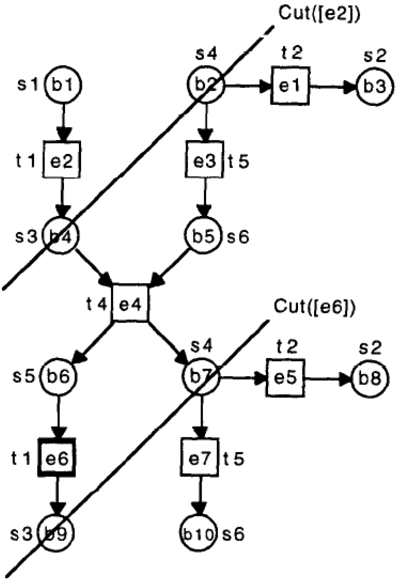


Its McMillan prefix

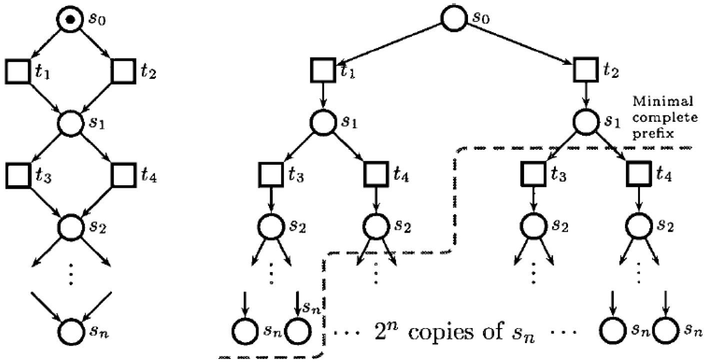


Its McMillan prefix





An exponentially-sized McMillan prefix



For every marking M all the configurations $[t]$ satisfying $M([t]) = M$ have the same size, and therefore there exist no cut-off events [Kishinevsky and Taubin]

Overview

Summary

- A branching process captures several distributed runs of N

Summary

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

Summary

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

Summary

- A branching process captures several distributed runs of N
- It is represented by a relaxed notion of causal net, the occurrence net
- Branching processes are mapped to N via homomorphisms
- A homomorphism is a structure-preserving mapping between two nets

Summary

- A branching process captures several distributed runs of N
- It is represented by a relaxed notion of causal net, the occurrence net
- Branching processes are mapped to N via homomorphisms
- A homomorphism is a structure-preserving mapping between two nets
- Approximation (denoted $\sqsubseteq$) is a partial-order on branching processes

Summary

- A branching process captures several distributed runs of N
- It is represented by a relaxed notion of causal net, the occurrence net
- Branching processes are mapped to N via homomorphisms
- A homomorphism is a structure-preserving mapping between two nets
- Approximation (denoted $\sqsubseteq$) is a partial-order on branching processes
- Isomorphic branching process with $\sqsubseteq$ are a complete lattice

Summary

- A branching process captures several distributed runs of N
- It is represented by a relaxed notion of causal net, the occurrence net
- Branching processes are mapped to N via homomorphisms
- A homomorphism is a structure-preserving mapping between two nets
- Approximation (denoted $\sqsubseteq$) is a partial-order on branching processes
- Isomorphic branching process with $\sqsubseteq$ are a complete lattice
- True concurrency semantics of $N =$ the maximal element (under $\sqsubseteq$)

Summary

- A branching process captures several distributed runs of N
- It is represented by a relaxed notion of causal net, the occurrence net
- Branching processes are mapped to N via homomorphisms
- A homomorphism is a structure-preserving mapping between two nets
- Approximation (denoted $\sqsubseteq$) is a partial-order on branching processes
- Isomorphic branching process with $\sqsubseteq$ are a complete lattice
- True concurrency semantics of $N =$ the maximal element (under $\sqsubseteq$)
- For 1-bounded nets, the McMillan prefix covers all reachable markings