Analysis of Distributed Systems

Máté Tejfel

September 19, 2013

Theme I

Part I/a

Agenda

Basic Definitions

Definition 1 (Bipartite Graph)

Bipartite Graph is a graph of which nodes can be divided into two disjoint sets such that can not exists edge between two elements of the same set.

Definition 2 (Petri net)

Petri net is a tuple (N, M_0), where

- *the underlying graph $N = (P, T, R, v)$ is a directed, weighted, bipartite graph consisting of two kinds of nodes, called places and transitions.*
  - *P is the (finite) set of places,*
  - *T is the (finite) set of transitions, $(P \cup T \neq \emptyset,\ P \cap T = \emptyset)$*
  - *$R \subseteq (P \times T) \cup (T \times P)$ gives the edges,*
  - *$v : R \rightarrow \mathcal{N}$ gives the weights of the edges.*
- *$M_0 : P \rightarrow \mathcal{N}_0$ is the initial marking (the initial state). Places may contain a discrete number of marks called tokens.*

Basic Definitions (Some Notations)

- The initial marking of a Petri net (containing n places) can be considered as an n-tuple $M_0 = (M_0(p_1), M_0(p_2), .., M_0(p_n))$.
- $^\bullet p ::= R^{(-1)}(p)$ is the *preset* of place p (the set of transitions connected to p)
- $p^\bullet ::= R(p)$ is the *postset* of place p (the set of transitions p is connected to)
- $^\bullet t ::= R^{(-1)}(t)$ and $t^\bullet ::= R(t)$ are similarly the preset and postset of transition t

Basic Definitions

Definition 3 (Firing rule)

Let $N = (P, T, R, v)$ be a Petri net with some marking M.

1. *A transition $t \in T$ is enabled if $\forall p \in {}^{\bullet}t$: $M(p) \geq v(p,t)$.*
2. *During one execution step one of the enabled transitions will fire.*
3. *The firing of an enabled transition t produces a new marking M' (the successor marking), where $\forall p \in P : M'(p) = M(p) + v(t,p) - v(p,t)$*

Basic Definitions

- t is a source transition if ${}^\bullet t = \emptyset$,
- t is a sink transition if $t^\bullet = \emptyset$,
- (p, t) is a self-loop if $p \in {}^\bullet t \land p \in t^\bullet$,
- a Petri net is pure if it has no self-loops,
- a Petri net is ordinary if all of its arc weights are 1's $(\forall r \in R : v(r) = 1)$.

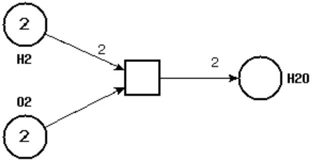
Graphical representation

Petri Nets have clear graphical representation, where

- places are denoted by cirles,
- transitions are denoted by squeres,
- and marking is denoted by flecks or numbers.

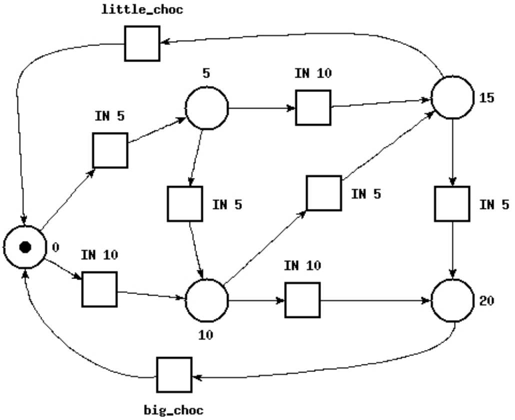
Examples

Example 1 (Synthesis of water)



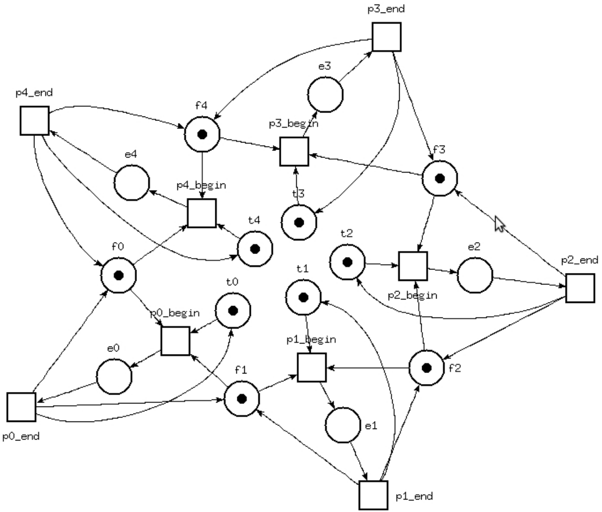
Examples

Example 2 (Vending Machine)



Examples

Example 3 (Dining Philosophers)



Finite capacity net

Definition 4

A Petri net (N, M_0) is a finite capacity net if each place p has an associated capacity $k(p)$, the upper bound for marking of p $(M(p))$.

Finite capacity net

Definition 5 (Strict firing rule)

Let $N = (P, T, R, v)$ be a finite capacity net with some marking M.

1'. *A transition $t \in T$ is enabled if $\forall p \in {}^\bullet t$: $M(p) \geq v(p, t)$ and $\forall p \in t^\bullet$: $M'(p) \leq k(p)$, where $M'(p) = M(p) + v(t, p) - v(p, t)$.*

2. *During one execution step one of the enabled transitions will fire.*
3. *The firing of an enabled transition t produces a new marking M' (the successor marking), where $\forall p \in P : M'(p) = M(p) + v(t, p) - v(p, t)$*

Finite capacity net

Definition 6 (Weak firing rule)

Let $N = (P, T, R, v)$ be a finite capacity net with some marking M.

1. *A transition $t \in T$ is enabled if $\forall p \in {}^{\bullet}t$: $M(p) \geq v(p, t)$.*
2. *During one execution step one of the enabled transitions will fire.*

3'. *The firing of an enabled transition t produces a new marking M'' (the successor marking), where $\forall p \in P : M''(p) = min\big(k(p), M(p) + v(t, p) - v(p, t)\big)$*

Finite capacity net

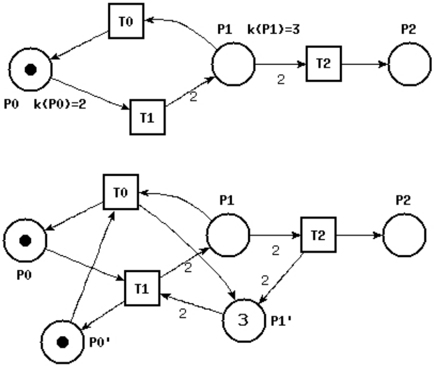
Definition 7 (Complementary place transformation)

Let $N = (P, T, R, v)$ be a finite capacity net with some marking M.

1. $\forall p \in P : k(p) < \infty$ *we create a complementary place* p'*, where* $M`_0(p`) = k(p) - M_0(p)$*,*
2. $\forall t \in T$:
   - *if there exists an edge* $(p, t) \in R$*, we create a new edge* (t, p') *so that* $v(t, p') = v(p, t)$ *will hold,*
   - *if there exists an edge* $(t, p) \in R$*, we create a new edge* (p', t) *so that* $v(p', t) = v(t, p)$ *will hold.*

Finite capacity net

Example 4 (Complementary place transformation)



Finite capacity net

Definition 8 (Enabled firing sequences)

Let $N = (P, T, R, v)$ be a Petri net with some marking M_0.
$M_0\,[t_1{>}M_1$ signs that t_1 is enabled in (N, M_0) and the firing of t_1 produces marking M_1.
The firing sequence $\varsigma = t_1, t_2, \ldots, t_n$ is enabled in (N, M_1), if there exist markings $M_1, M_2, \ldots, M_n$, such that
$M_0\,[t_1{>}M_1\,[t_2{>}M_2, \ldots, M_{n-1}\,[t_n{>}M_n$.
(Short notation: $M_0\,[\varsigma{>}M_n$.)

Finite capacity net

Theorem 5 (Finite capacity elimination)

Let (N, M_0) be a pure finite capacity net and (N', M_0') the result of the complementary place transformation applied to (N, M_0). If the strict firing rule is applied to (N, M_0) and the original one (defined in Definition 3) to (N', M_0') the set of the enabled firing sequences will be the same (the two nets will be equivalent in this manner).

As a conclusion of the previous theorem we only need consider infinite capacity nets with original firing rule.

Agenda

Behavioral properties

Two types of properties can be studied:

- depend on initial marking (marking dependent / behavioral properties)
- independent of the initial marking (structural properties)

Now we will discuss only the marking dependent properties!

Reachability/1

Definition 9

A marking M_n is reachable from M_0 (in short: $M_0\,[\varsigma{>}M_n$), if there exists a sequence of firings ($\varsigma = t_1, t_2, .., t_n$) that transforms M_0 to M_n.

Long version: $M_0\,[t_1{>}M_1\,[t_2{>}M_2..M_{n-1}\,[t_n{>}M_n$

Notations:

- $L(N, M_0)$: the set of all possible firing sequence from M_0 in a net (N, M_0)
- $R(N, M_0)$: the set of all possible markings reachable from M_0 in a net (N, M_0)

If N is given: $L(M_0)$, $R(M_0)$

Reachability/2

Generally: $R(N, M) = \{M' | \exists \varsigma \in L(N, M) : M [\varsigma{>}M'\}$

Reachability problem: $M_n \in R(M_0)$ for a given marking M_n?

Note 1

The reachability problem is decidable.

Note 2

However the equality problem is undecidable.
$L(N, M_0) = L(N', M_0')$ *for any two Petri nets* N *and* N'

Boundedness

Definition 10 (k-bounded Petri nets)

A Petri net is k-bounded, if
$\forall M \in R(N, M_0) : \forall p \in P : M(p) \leq k$. *($k \in \mathcal{N}$)*

There is no marking reachable from M_0, which has more than k tokens in one place.

Definition 11 (Safe Petri nets)

A Petri net is safe, if it is 1-bounded.

Places can be used as buffers and registers for storing intermediate data. Boundedness, safeness means: overflow can not happen, no matter what firing sequence is taken.

Liveness/1

Liveness $\approx$ deadlock free

Notation: $\#(\varsigma, t)$: the number of occurrences of t in ς.

Definition 12 (Liveness)

A transition t in a Petri net N with the initial marking M_0 is said to be:

- *L_0-live (dead):* $\forall \varsigma \in L(N, M_0) : t \notin \varsigma$,
- *L_1-live (potentially fireable):* $\exists \varsigma \in L(N, M_0) : t \in \varsigma$,
- *L_2-live:* $\forall k \in \mathcal{N} : \exists \varsigma \in L(N, M_0) : \#(\varsigma, t) \geq k$,
- *L_3-live:* $\exists \varsigma \in L(N, M_0) : \#(\varsigma, t) = \infty$,
- *L_4-live: if $t \in T$ is L_1-live for $\forall M \in R(M_0)$ marking.*

Liveness/2

Definition 13 (L_k-live)

A Petri net (N, M_0) is L_k-live if $\forall t \in T : t$ is L_k-live

Definition 14 (Strict liveness)

Strictly L_k live: L_k live, but not L_{k+1}-live

Note 3

$L_4 \Rightarrow L_3 \Rightarrow L_2 \Rightarrow L_1$,
$\neg L_0 \Leftrightarrow L_1$

Reversibility and Home state

Definition 15 (Reversibility)

A (N,M_0) Petri net is reversible if $\forall M \in R(M_0) : M_0 \in R(M)$.

In a reversible net one can always get back to the initial marking or state.

Generalization: not just M_0, but any reachable marking can be examined

Definition 16 (Home state)

M' *is a Home state, if* $\forall M \in R(M_0) : M' \in R(M)$.

Coverability

Definition 17 (Coverability)

A marking M in a Petri net (N, M_0) is coverable, if $\exists M' \in R(M_0) : \forall p \in P : M'(p) \geq M(p)$.

Coverability and L_1-liveness are closely related!

Note 4 (Liveness and coverability)

Let M be the minimum marking, which enables transition t:

- *t is dead (L_0-live) $\Leftrightarrow$ M is not coverable*
- *t is L_1-live $\Leftrightarrow$ M is coverable*

Persistence

Definition 18 (Persistence)

(N, M_0) is persistent, if $\forall t1, t2 \in T$:, if both t_1 and t_2 is enabled, then firing one of them will not disable the other.

In short: A transition in a persistent net, once it is enabled, will stay enabled until it fires.

Synchronic distance

- a metric
- related to a degree of mutual dependence between two events
- defined between two transitions (1) or two sets of transitions (2)

Definition 19 (Synchronic distance (1))

In case of $(N, M_0) : t_1, t_2 \in T$

$$d_{1,2} := \max_{\varsigma \in L(N,M), M \in R(M_0)} |\#(\varsigma, t_1) - \#(\varsigma, t_2)|$$

Definition 20 (Synchronic distance (2))

In case of $(N, M_0) : E_1, E_2 \subseteq T$

$$d_{E_1,E_2} := \max_{\varsigma \in L(N,M), M \in R(M_0)} |\#(\varsigma, E_1) - \#(\varsigma, E_2)|$$

Fairness/1

Definition 21 (Bounded-fair or B-fair)

1. *Two transitions t_1 and t_2 are in a bounded-fair relation if the maximum number of times that either one can fire while the other is not firing is bounded* $(\exists K \in \mathcal{N} : d_{i,j} < K)$.
2. *A Petri net (N, M_0) is a bounded-fair net if* $\forall t_i, t_j \in T : \exists K \in \mathcal{N} : d_{i,j} < K$.

Definition 22 (Unconditionally fair)

3. *In case of* $\forall \varsigma \in L(N, M) : \forall M \in R(M_0)$ *: a ς firing sequence is unconditionally fair, if* $\forall t_j \in T : \#(\varsigma, t_j) = \infty$ *or ς is finite.*
4. *A Petri net (N, M_0) is unconditionally fair, if* $\forall M \in R(M_0) : \forall \varsigma \in L(N, M) :$ *ς is unconditionally fair.*

Fairness/2

Theorem 6
(2) $\Rightarrow$ (4)

Every B-fair net is an unconditionally-fair net.

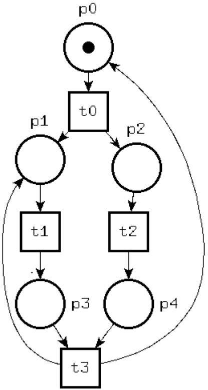
Theorem 7
A Petri net is bounded and fulfills (4) $\Rightarrow$ (2) is fulfilled too.

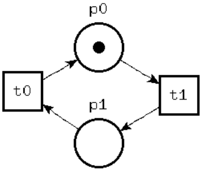
Every bounded unconditionally-fair net is a B-fair net.

Fairness/3

Example 1

(4) but not (2) —— (2) and (4)





Agenda

Reduction rules for analysis

Motivation:

- Analysis of large systems can be tedious
- Reduce to a simple one, while properties are preserved

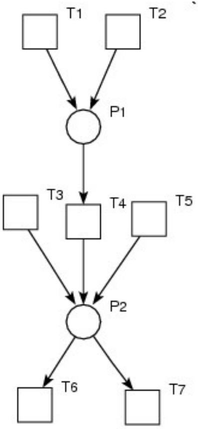
Theorem 8 (Behavioral preserving)

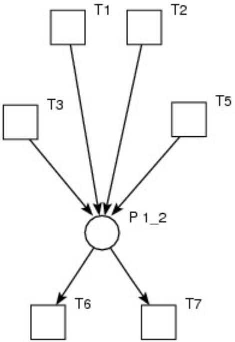
Let (N, M_0) and (N', M_0') be the Petri nets before and after one of the succeeding six simple transformations.

(N', M_0') is live, safe or bounded $\Leftrightarrow$ (N, M_0) is live, safe or bounded

Fusion of series places

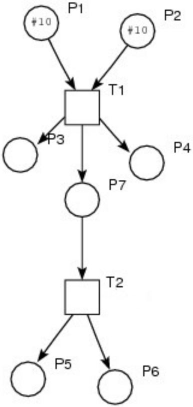
Example 2

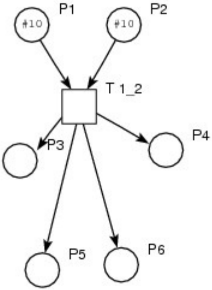




Fusion of series transitions

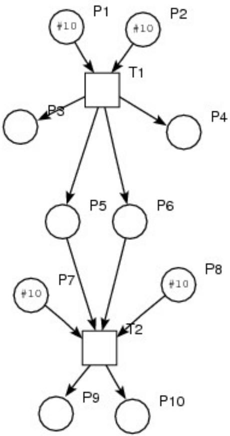
Example 3

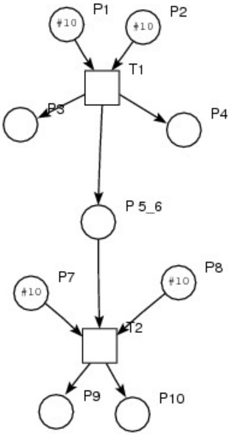




Fusion of parallel places

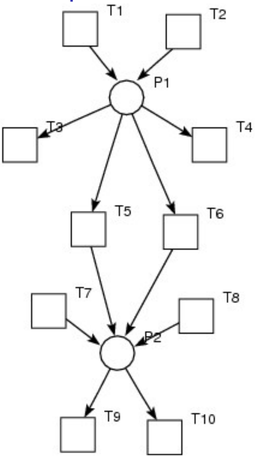
Example 4

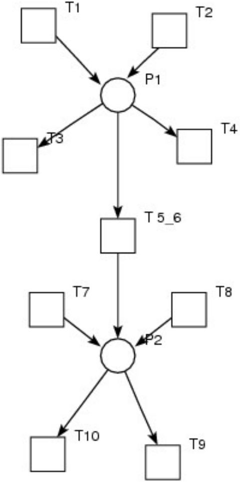




Fusion of parallel transitions

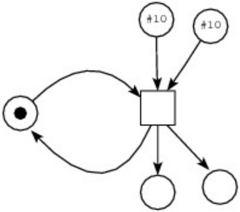
Example 5

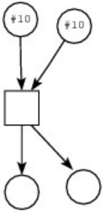




Elimination of self-loop places

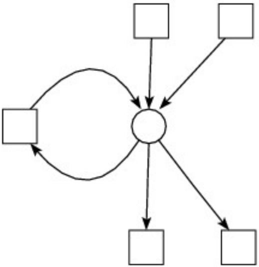
Example 6

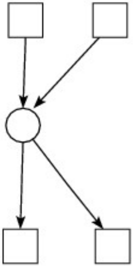




Elimination of self loop transitions

Example 7





Coverability/reachability tree

Given (N, M_0) Petri net

- from M_0 we can reach as many "new" markings as the number of the enabled transitions
- from each marking we can again reach more markings
- result: tree representation of the markings

Definition 23

The reachability / coverability tree of an (N, M_0) Petri net is a graph, where the nodes are labeled with markings and the edges are labeled with firing transitions.

Note 5

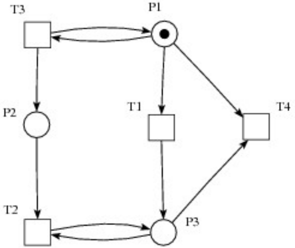
The tree will grow inifinitely large if the net is unbounded. A special ω symbol is introduced as "inifinity" to keep the tree finite.

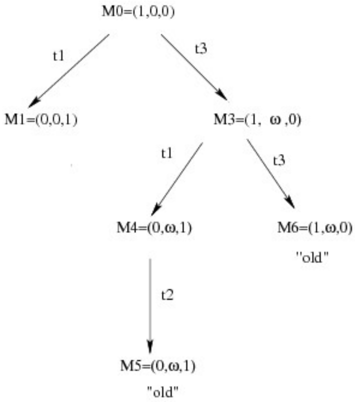
Construction (reachability tree)

1. The initial marking M_0 is the root, and labeled as "new"
2. While "new" markings exists, do the following:
   2.1 Select a "new" marking (M).
   2.2 If M is on the path from the root to M, than label it as "old" and start with another "new" marking.
   2.3 If no transitions are enabled at M, then tag it as "dead-end".
   2.4 While there are enabled transitions at M, do the following for each enabled "t" transition:
      2.4.1 Fire t, which transforms M marking to M' marking.
      2.4.2 If $\exists M''$ marking on the path from the root to M, such that $\forall p : M'(p) \geq M''(p)$ and $M' \neq M''$ then replace $M'(p)$ by ω for $\forall p : M'(p) > M''(p)$.
      2.4.3 Introduce M' as a node, connect it with an edge to M and label the edge with "t". Tag the M' as "new".

Coverability tree/1

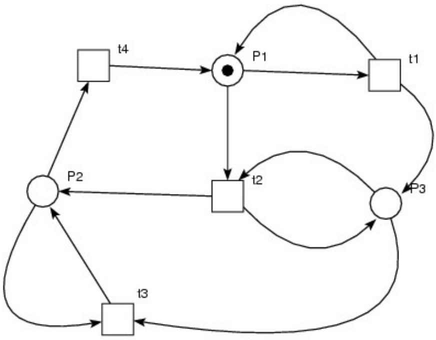
Example 8

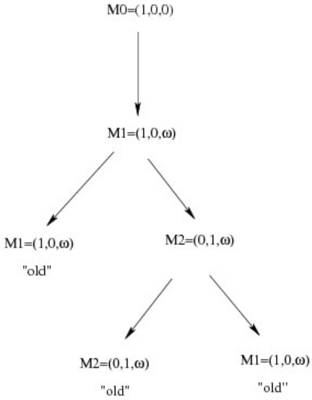




Coverability tree/2

Example 9





Coverability/reachability tree

Theorem 9

G is the coverability tree of the (N, M_0) Petri net

- *the Petri net is bounded $\Leftrightarrow$ there is no ω in G.*
- *the Petri net is safe (1-bounded) $\Leftrightarrow$ only 0,1 appears in the nodes of G.*
- *t dead (L_0-live) $\Leftrightarrow$ $\nexists$ edge labeled with t in G.*

Theorem 10

$M \in R(M_0) \Rightarrow \exists M'$ *in* G : M' *covers* M.

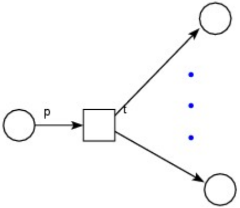
Note 6

By merging the identical nodes (markings), we can transform the coverability tree into a coverability graph.

About places

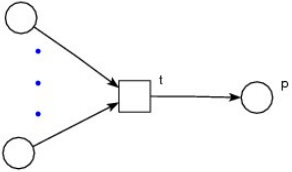
Theorem 11

$^{\bullet}p = \emptyset \Rightarrow t$ *not live*



Theorem 12

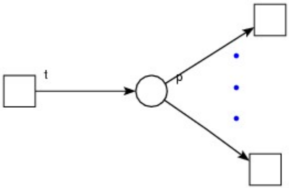
$p^{\bullet} = \emptyset \wedge t$ *live* $\Rightarrow p$ *not safe*



About transitions

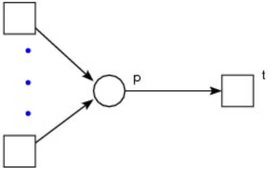
Theorem 13

$t^{\bullet} = \emptyset \Rightarrow p$ *not safe*



Theorem 14

$t^{\bullet} = \emptyset \wedge p$ *safe* $\Rightarrow t$ *not live*



Strongly connected Petri nets

Theorem 15

If (N, M_0) is live and safe $\Rightarrow \forall x \in P \cup T : x^{\bullet} \neq \emptyset \neq {}^{\bullet}x$

Theorem 16

Connected, live and safe Petri nets $\Rightarrow$ *strongly connected.*

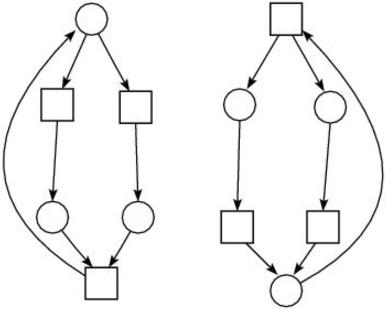
Note 7

The previous theorem is not reversible!

Example for non-reversibility

Example 10

The first net hasn't got any live initial markings, while the second hasn't got any safe and not empty initial markings.



Agenda

Introduction, reminder

We define subclasses of Petri nets by adding some restrictions on their structure.

Reminder:

- we work with *ordinary* (edge weights are 1) Petri nets.
- ${}^\bullet t = \{p | (p, t) \in F\}$ = the set of t's input places.
- $t^\bullet = \{p | (t, p) \in F\}$ = the set of t's output places.
- ${}^\bullet p = \{t | (t, p) \in F\}$ = the set of p's input transitions.
- $p^\bullet = \{t | (p, t) \in F\}$ = the set of p's output transitions.

Classification, subclasses/1

Definition 24 (State Machine (SM))

$\forall t \in R : |\, {}^{\bullet}t \,| = |\, t^{\bullet} \,| = 1$

Each transition t has exactly one input place and exactly one output place.

Definition 25 (Marked Graph (MG))

$\forall p \in P : |\, {}^{\bullet}p \,| = |\, p^{\bullet} \,| = 1$

Each place p has exactly one input transition and exactly one output transition.

Note 8 (About MGs)

- *conflict free Petri net $\not\Rightarrow$ MG*
- *MG $\Rightarrow$ persistent*
- *Persistent, safe Petri net is transformable to MG*

Classification, subclasses/2

Definition 26 (Free Choice (FC))

$\forall p \in P : \mid p^\bullet \mid \leq 1 \vee {}^\bullet(p^\bullet) = p$

Equivalent definition:

$\forall p_1, p_2 \in P : p_1^\bullet \cap p_2^\bullet \neq \emptyset \Rightarrow |p_1^\bullet| = |p_2^\bullet| = 1$

Every edge from a place is either a unique outgoing edge or a unique incoming edge to a transition.

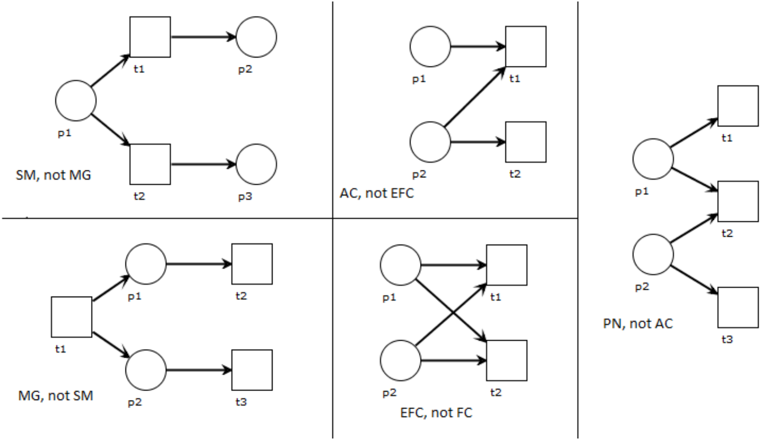
Definition 27 (Extended Free Choice (EFC))

$\forall p_1, p_2 \in P : p_1^\bullet \cap p_2^\bullet \neq \emptyset \Rightarrow p_1^\bullet = p_2^\bullet$

Definition 28 (Asymmetric Choice (AC))

$\forall p_1, p_2 \in P : p_1^\bullet \cap p_2^\bullet \neq \emptyset \Rightarrow p_1^\bullet \subseteq p_2^\bullet \vee p_2^\bullet \subseteq p_1^\bullet$

Key structures - Overview



Properties - Overview

SM : no syncronization

MG : no conflict

FC : no confusion

AC : allow asymetric confusion, but disallow symmetric confusion

Note 9

In case of FC, EFC : $\exists p \in ({}^\bullet t_1 \cap {}^\bullet t_2) \Rightarrow \nexists M$ *marking such that only* t_1 *or only* t_2 *enabled. Thus we have "free-choice" about which transition to fire. An EFC can be converted to it's FC equivalent.*

Source, Sink, Siphon, Trap

Definition 29 (Source place (transition))

A place p (transition t) is a source place (source transition) if $^\bullet p = \emptyset$ $(^\bullet t = \emptyset)$

Definition 30 (Sink place (transition))

A place p (transition t) is a sink place (sink transition) if $p^\bullet = \emptyset$ $(t^\bullet = \emptyset)$

Definition 31 (Siphon (deadlock))

S is a set of places, S is siphon if $^\bullet S \subseteq S^\bullet$

If a siphon place is unmarked, then it remains so.

Definition 32 (Trap)

S is a set of places, S is trap if $S^\bullet \subseteq {^\bullet S}$

If a trap place is marked, then it remains so.

State Machine - Liveness, safeness

Definition 33 (State machine - reminder)

$\forall t \in T : |\, {}^\bullet t \,| = |\, t^\bullet \,| = 1$

Theorem 17

A (N, M_0) SM is live $\Leftrightarrow$ N strongly connected, and M_0 has at least one token.

Theorem 18

A (N, M_0) SM is safe $\Leftrightarrow$ M_0 has at most one token.

Theorem 19

A live (N, M_0) SM is safe $\Leftrightarrow$ M_0 has exactly one token, and N is strongly connected.

Marked Graph/1

Definition 34 (Marked graph - reminder)

$\forall p \in P : | {}^{\bullet}p | = | p^{\bullet} | = 1$

Theorem 20

For a MG, the token count in a directed circuit is invariant under any firing, i.e., $\forall M \in R(M_0) : \forall C : M_0(C) = M(C)$, where C is the set of nodes of the directed circuit.

By the previous theorem:
If a transition t is L_0-live (dead) in a strongly connected MG $\Rightarrow$ there is a tokenless directed circuit, which contains t.

Theorem 21

Strongly connected MG (N, M_0) is live $\Leftrightarrow$ M_0 places at least one token on each directed circuit in N.

Marked Graph/2

Theorem 22 (Mini-max)

The maximum number of tokens that an edge can have in a MG (N, M_0) is equal to the minimum number of tokens placed by M_0 on a directed circuit containing this edge.

Theorem 23

A live MG (N, M_0) is safe $\Leftrightarrow$ every edge (place) belongs to a directed circuit C with $M_0(C)=1$.

Theorem 24

There exists a live and safe marking in MG (N, M_0) $\Leftrightarrow$ N is strongly connected.

Feedback Arc Set (FAS)/1

Definition 35

A subset of edges E' in a directed graph $G = (V, E)$ is a feedback arc set if $G' = (V, E - E')$ is acyclic.

Definition 36

FAS is minimal if no proper subset of the FAS is a FAS.

Definition 37

FAS is minimum if no other FAS contains a smaller number of edges.

Note 10

A FAS is not necessarily unambiguous.

Feedback Arc Set (FAS)/2

Theorem 25

A subset of marked edges of a live MG's is a FAS.

Conversely, if each edge in a FAS of a directed graph is marked, we have a live MG.

Theorem 26

A strongly connected live MG is safe $\Leftrightarrow \forall M \in R(M_0)$ *: the set of marked edges is a minimal FAS.*

Note 11

A minimum FAS does not necessary yield a safe marking.

Liveness, safeness in FC nets

Theorem 27 (FC's liveness)

An FC (N, M_0) is live $\Leftrightarrow$ every siphon in N contains a marked trap.

Theorem 28 (Live FC's safeness)

A live FC (N, M_0) is safe $\Leftrightarrow$ N is covered by strongly-connected SM components each of which has exactly one token at M_0.

Theorem 29

Let (N, M_0) be a live and safe FC. Then, N is covered by strongly-connected MG components. $\exists M \in R(M_0) : \forall (N_i, M_i)$ component is a live and safe MG, where M_i is M restricted to N_i.

SM/MG component

Definition 38 (SM-component (MG-component))

An SM-component (MG-component) N_1 of a net N is defined as a subnet generated by places (transitions) in N_1 having the following two properties:

- $\forall t(p) \in N_1$ *has at most one incoming edge and at most one outgoing edge*
- *a subnet generated by places (transitions) is the net consisting of these places (transitions), all of their input and output transitions (places), and their connecting edges.*

Note 12

A live and safe FC can be viewed as an interconnection of live and safe SMs (MGs).

Liveness, safeness in AC nets

Theorem 30 (AC's liveness)

An AC (N, M_0) is live $\Rightarrow$ every siphon in N contains a marked trap.

Theorem 31 (AC's liveness (2))

An AC (N, M_0) is live $\Leftrightarrow$ place-live.

Definition 39 (Place-liveness)

$\forall M_i \in R(M_0), \forall p \in N : \exists M \in R(M_i) : M(p) > 0.$

Agenda

Introduction to CP-nets

An ordinary Petri net (PT-net) has no types and no modules:

- Only one kind of tokens and the net is flat.

With Coloured Petri Nets (CP-nets) it is possible to use data types and complex data manipulation:

- Each token has attached a data value called the token colour.
- The token colours can be investigated and modified by the occurring transitions.

Coloured Petri Nets

Declarations:

- Types, functions, operations and variables.

Each *place* has the following inscriptions:

- Name (for identification).
- Colour set (specifying the type of tokens which may reside on the place).
- Initial marking (multi-set of token colours).

Each *transition* has the following inscriptions:

- Name (for identification).
- Guard (boolean expression containing some of the variables).

Each *arc* has the following inscriptions:

- Arc expression (containing some of the variables). When the arc expression is evaluated it yields a multi-set of token colours.

Enabling and occurrence

A *binding* assigns a colour (i.e., a value) to each variable of a transition.
A *binding element* is a pair (t, b) where t is a transition while b is a binding for the variables of t.
Example: $(T2, < x = p, i = 2 >)$.

A binding element is enabled if and only if:

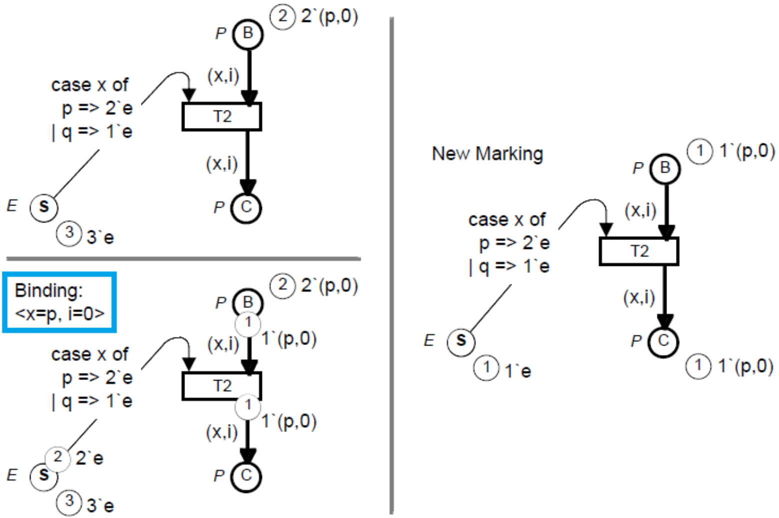
- There are enough tokens (of the correct colours on each input-place).
- The guard evaluates to true.

When a binding element is enabled it may occur:

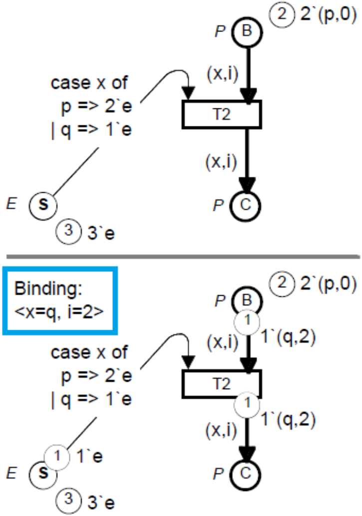
- A multi-set of tokens is removed from each input-place.
- A multi-set of tokens is added to each output-place.

A binding element may occur concurrently to other binding elements ⇔ each binding element can get its "own share".

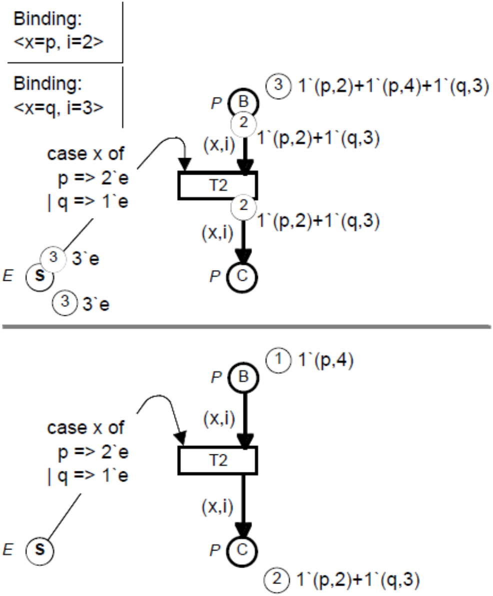
Enabled binding



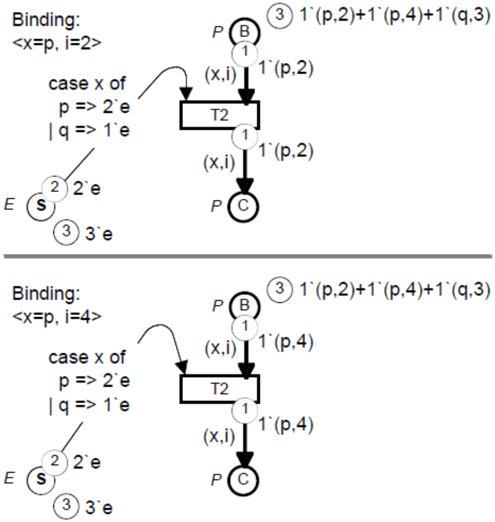
Not enabled binding



Concurrency



Conflict


Binding:
<x=p, i=2>
case x of
p => 2`e
| q => 1`e
3 1`(p,2)+1`(p,4)+1`(q,3)
P B
1 1`(p,2)
(x,i)
T2
1 1`(p,2)
(x,i)
P C
E S
2 2`e
3 3`e
Binding:
<x=p, i=4>
case x of
p => 2`e
| q => 1`e
3 1`(p,2)+1`(p,4)+1`(q,3)
P B
1 1`(p,4)
(x,i)
T2
1 1`(p,4)
(x,i)
P C
E S
2 2`e
3 3`e

Formal definition of CP-nets

Definition 40 (Coloured Petri Net)

is a tuple $CPN = (\Sigma, P, T, A, N, C, G, E, I)$ satisfying the following requirements:

(a) Σ is a finite set of non-empty types, called *colour sets*.

(b) P is a finite set of *place*s.

(c) T is a finite set of *transition*s.

(d) A is a finite set of *arc*s such that:
$P \cap T = P \cap A = T \cap A = \emptyset$

(e) N is a *node* function. $(N :: A \rightarrow P \times T \cup T \times P)$

(f) C is a *colour* function. $(C :: P \rightarrow \Sigma)$

Formal definition of CP-nets/2

(g) G is a *guard* function. It is defined from T into expressions such that:
$\forall t \in T : [Type(G(t)) = Bool \wedge Type(Var(G(t))) \subseteq \Sigma]$

(h) E is an *arch expression* function. It is defined from A into expressions such that: $\forall a \in A$:
$[Type(E(a)) = C(p(a))_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma]$
where $p(a)$ is the place of $N(a)$.

(i) I is an *initialization* function. It is defined from P into closed expressions such that:
$\forall p \in P : [Type(I(p)) = C(p)_{MS}]$

Note 13

$_{MS}$ *means multi-set.*

Formal definition of behaviour

Definition 41 (Step)

A step is a multi-set of binding elements.

Definition 42 (Enabled step)

A step Y is enabled in a marking $M \Leftrightarrow$ the following property is satisfied:

$$\forall p \in P : \sum_{(t,b) \in Y} E(p,t)\langle b \rangle \leq M(p)$$

Definition 43

When a step Y is enabled in a marking M_1 it may occur by changing to marking M_2:

$$\forall p \in P : M_2(p) =$$
$$(M_1(p) - \sum_{(t,b) \in Y} E(p,t)\langle b \rangle) + \sum_{(t,b) \in Y} E(t,p)\langle b \rangle$$

Formal definition of behaviour/2

Definition 44 (Directly reachable)

M_2 is directly reachable from M_1 by the step Y:
$M_1\,[Y{>}M_2$

Definition 45 (Occurrence sequence)

is a sequence of markings and steps:
$M_1\,[Y_1{>}M_2\,[Y_2{>}M_2..M_n\,[Y_n{>}M_{n+1}$

Definition 46 (Reachable)

M_{n+1} is reachable from M_1:
$\forall i \in [1..n] : \exists Y_i : M_i\,[Y_i{>}M_{i+1}$

Theme II

Part I/b

Agenda

Introduction to Labelled Petri nets

We assume a set Lab of *actions* to be given.

Definition 47 (relabelling)

ρ is a relabelling relation: $\rho \subseteq (mult(Lab)) \times Lab$ such that $(\emptyset, \alpha) \in \rho$ if and only if $\rho = \{(\emptyset, \alpha)\}$

Special relabellings:

- constant: $\rho_\alpha = \{(\emptyset, \alpha)\}$ where $\alpha \in Lab$
- transformational: $\rho_{Lab'} = \{(\{\alpha\}, \alpha) | \alpha \in Lab'\} : Lab' \subseteq Lab$
- identity: $\rho_{id} = \{(\{\alpha\}, \alpha) | \alpha \in Lab\}$

Labelled Petri net

Definition 48 (Labelled Petri net)

$\Sigma = (S, T, W, \lambda, M)$, *where* S *is a set of places,* T *is a set of transitions,* W *describes the edges,* λ *is a labelling function and* M *gives the marking.*

$S \cap T = \emptyset$,
$W : ((S \times T) \cup (T \times S)) \rightarrow N_0$,
$\forall s \in S : \lambda(s) \in \{e, i, x\}$,
$\forall t \in T : \lambda(t)$ *is a relabelling,*
$M : S \times N_0$

Definition 49 (Action counter - Bag function)

$\mu : A \longrightarrow \mathcal{N}_0$. $\mathcal{M}_F(A) = \{\mu | \mu : A \rightarrow \mathbf{N}_0 \}$. *The bag function gives the number of occurrence for an element (of the bag).*

Example 11

If μ *is the* $\{aabccc\}$ *bag, then* $\mu(a) = 2, \mu(b) = 1, \mu(c) = 3$.

Definition 50 (Pair definer function)

$^\wedge : A \rightarrow A : a \neq \hat{a}$, *bijection,* $^\wedge =^{\wedge(-1)}$, *defines pairs over* A.

Note 14

Notation: A *is given as a set of actions and* $^\wedge$ *is given as pair definer function over* A.
$\hat{\mu}(a) ::= \mu(\hat{a})$.

Semantics

$$\Sigma h(a) = \Sigma \mu(a) * h(a),$$
$$\cup(\mu_1, \mu_2) = \max \circ (\mu_1, \mu 2),$$
$$\cap(\mu_1, \mu_2) = \min \circ (\mu_1, \mu 2),$$
$$\mu_1 + \mu_2 = + \circ (\mu_1, \mu 2),$$
$$\mu_1 - \mu_2 = \text{difference or } 0,$$

Notations

$\Sigma = (S, T, W, \lambda, M)$. Given $s \in S$. If
$\lambda(s) = \{e\}$, then s is an entry place,
$\lambda(s) = \{x\}$, then s is an exit place,
$\lambda(s) = \{i\}$, then s is an internal place.

${}^\bullet\Sigma = \{s \in S | \lambda(s) = \{e\}\}$ entry places
$\Sigma^\bullet = \{s \in S | \lambda(s) = \{x\}\}$ exit places
$\ddot{\Sigma} = \{s \in S | \lambda(i) = \{i\}\}$ internal places

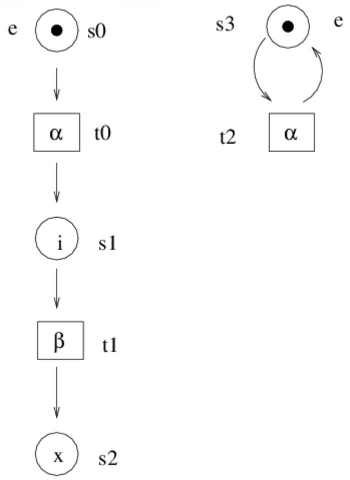
Example - A labelled Petri Net

$\Sigma_0 = (S_0, T_0, W_0, \lambda_0, M_0)$
$S_0 = \{s_0, s_1, s_2, s_3\}$
$T_0 = \{t_0, t_1, t_2\}$
$W_0 = ((TS \cup ST) \times \{1\}) \cup (((S \times T) \setminus ST \cup (T \times S) \setminus TS) \times \{0\})$
$\lambda_0 = \{(s_0, e), (s_1, i), (s_2, x), (s_3, e), (t_0, \alpha), (t_1, \beta), (t_2, \alpha)\}$
$M_0 = \{(s_0, 1), (s_1, 0), (s_2, 0), (s_3, 1)\}$

where

$TS = \{(t_0, s_1), (t_1, s_2), (t_2, s_3)\}$ and
$ST = \{(s_0, t_0), (s_1, t_1), (s_3, t_2)\}$

Previous example again



Step sequence

Definition 51 (Step)

$\Sigma = (S, T, W, \lambda, M)$ *A finite multiset of transitions* $U \in mult(t)$*, called a step is enabled by* Σ *if* $\forall s \in S : M(s) \geq \Sigma_{t \in U}(U(t) * W(s,t))$

Note 15

Notation: $M\,[U\!>$ *or* $\Sigma\,[U\!>$

This means that every place has enough marking to perform every transition in a simultaneous way.

Step execution

Definition 52 (Step execution - semantics)

$\forall s \in S : M'(s) = M(s) + \Sigma_{t \in U} U(t) * (W(t,s) - W(s,t))$

Note 16

Notation: $M\,[U{>}M'$ *or* $\Sigma\,[U{>}\Theta$*, where*

$\Theta = (S, T, W, \lambda, M')$

Definition 53 (Step sequence)

of Σ *is a possibly empty sequence of step,* $\rho = U_1 ... U_k$*, such that* $\exists \Sigma_1 ... \Sigma_k$ *satisfying* $\Sigma = \Sigma_0$ *and* $\forall \in [1..k] : \Sigma_{i-1} [U_i > \Sigma_i$*.*

Note 17

Notation:

- $\Sigma [\rho > \Sigma_k$
- Σ_k *is derivable from* Σ
- *and its marking* M_{Σ_k}*, reachable from* M_Σ

Agenda

Properties

Definition 54 (T-restricted)

$\Sigma = (S, T, W, \lambda, M)$ *labelled Petri net is T-restricted, if*
$\forall t \in T : {}^\bullet t \neq \emptyset \wedge \ t^\bullet \neq \emptyset$,
namely there is not any transition which has empty preset or postset.
(In what follows, every analysed net is supposed to satisfy this property.)

Definition 55 (ex-restricted)

$\Sigma = (S, T, W, \lambda, M)$ *labelled Petri net is ex-restricted, if*
${}^\bullet\Sigma \neq \emptyset \wedge \ \Sigma^\bullet \neq \emptyset$,
namely there exists at least one entry and one exit place.

Properties

Definition 56 (e-directed)

$\Sigma = (S, T, W, \lambda, M)$ *labelled Petri net is e-directed, if*
$\forall s \in {}^{\bullet}\Sigma$: $\forall t \in T : W(t, s) = 0$,
namely entry places have not incoming arcs.

Definition 57 (x-directed)

$\Sigma = (S, T, W, \lambda, M)$ *labelled Petri net is x-directed, if*
$\forall s \in \Sigma^{\bullet}$: $\forall t \in T : W(s, t) = 0$,
namely exit places have not outgoing arcs.

Definition 58 (ex-directed)

A labelled Petri net is ex-directed, if e-directed and x-directed.

Definition 59

Let $\Sigma = (S, T, W, \lambda, M)$ *be a labelled Petri net.*

$$\forall s \in S : M_{\bullet\Sigma}(s) = \begin{cases} 1 & \textit{if } s \in {}^{\bullet}\Sigma \\ 0 & \textit{otherwise} \end{cases}$$

$$\forall s \in S : M_{\Sigma\bullet}(s) = \begin{cases} 1 & \textit{if } s \in \Sigma^{\bullet} \\ 0 & \textit{otherwise} \end{cases}$$

Definition 60 (ex-exclusive)

$\Sigma = (S, T, W, \lambda, M_0)$ *labelled Petri net is ex-exclusive, if for every marking* M *reachable from* M_0, $M_{\bullet\Sigma}$ *or* $M_{\Sigma\bullet}$ *:*
$M \cap M_{\bullet\Sigma} = \emptyset$ *or* $M \cap M_{\Sigma\bullet} = \emptyset$*.*
Namely it is not possible to mark simultaneously an entry and an exit place.

Definition 61 (ex-asymmetric)

Let be $\Sigma = (S, T, W, \lambda, M)$ a labelled Petri net. A $t \in T$ transition is ex-asymmetric, if $({}^\bullet t \cap {}^\bullet\Sigma \neq \emptyset) \ \wedge \ ({}^\bullet t \cap \Sigma^\bullet \neq \emptyset)$ or $(t^\bullet \cap {}^\bullet\Sigma \neq \emptyset) \ \wedge \ (t^\bullet \cap \Sigma^\bullet \neq \emptyset)$.

Note 18

Let be $\Sigma = (S, T, W, \lambda, M)$ a labelled Petri net. If there exists a $t \in T$ transition which is ex-asymmetric, then Σ is ex-restricted but it is not ex-directed. And if t is executable, then Σ is not ex-exclusive.

Definition 62 (independence relation)

$$ind_\Sigma = \{(t,u) \in T \times T \mid ({}^\bullet t \cup t^\bullet) \cap ({}^\bullet u \cup u^\bullet) = \emptyset\}$$

Note 19

If $\Sigma = (S, T, W, \lambda, M)$ is safe (1-bounded), then any two transitions occurring in the same step are independent.

Definition 63 (Notations)

Let be $\Sigma = (S, T, W, \lambda, M)$. We can use the following notations.

$\lfloor \Sigma \rfloor = (S, T, W, \lambda, \emptyset)$

$\overline{\Sigma} = (S, T, W, \lambda, M_{\bullet \Sigma})$

$\underline{\Sigma} = (S, T, W, \lambda, M_{\Sigma \bullet})$

Definition 64 (Petri box)

Σ labelled Petri net is a Petri box, if it is ex-restricted, ex-directed and T-restricted.

Definition 65 (plain box)

$\Sigma = (S, T, W, \lambda, M)$ Petri box is a plain box if for every $t \in T$ transition $\lambda(t)$ is a constant relabelling.

Definition 66 (clean marking)

M marking is clean if it is neither a proper super-multiset of $M_{\bullet\Sigma}$ nor of $M_{\Sigma\bullet}$. Namely, if $M_{\bullet\Sigma} \subseteq M$, then $M_{\bullet\Sigma} = M$ and if $M_{\Sigma\bullet} \subseteq M$, then $M_{\Sigma\bullet} = M$.

Definition 67 (static box)

$\Sigma = (S, T, W, \lambda, M)$ *plain Petri box is a static box if* $M_\Sigma = \emptyset$ *and every marking reachable from* $M_{\bullet\Sigma}$ *and* $M_{\Sigma\bullet}$ *is safe and clean.*

Definition 68 (dinamic box)

$\Sigma = (S, T, W, \lambda, M)$ *plain Petri box is a dinamic box if it is marked* ($M_\Sigma \neq \emptyset$) *and every marking reachable from* $M_{\bullet\Sigma}$, $M_{\Sigma\bullet}$ *and* M *is safe and clean.*

Note 20

If Σ *and* Θ *are Petri boxes,* Σ *is a static box and* Θ *is derivable from* $\overline{\Sigma}$, *then* Θ *is a dinamic box. (Accordingly* $\overline{\Sigma}$ *is a dinamic box too.)*

Definition 69 (entry box)

$\Sigma = (S, T, W, \lambda, M)$ *dinamic Petri box is entry box if* $M = M_{\bullet\Sigma}$.

Definition 70 (exit box)

$\Sigma = (S, T, W, \lambda, M)$ *dinamic Petri box is exit box if* $M = M_{\Sigma\bullet}$.

Definition 71 (Notations)

Box^s *is the set of static boxes,*
Box^d *is the set of dinamic boxes,*
Box^e *is the set of entry boxes,*
Box^x *is the set of exit boxes.*

Theorem 32

Let $\Sigma = (S, T, W, \lambda, M)$ be a dinamic Petri box and U be a step enabled by Σ.

- *If $\Theta = (S_2, T_2, W_2, \lambda_2, M_2)$ is a Petri box, derivable from Σ, then Θ is a dinamic box.*
- *U is a set of mutually independent transitions. Namely $U \times U \subseteq ind_\Sigma \cup id_T$, where $id_X = \{(x, x) | x \in X\}$.*
- *Every arcs connected to transitions in U are unitary, namely $W(U \times S) \cup W(S \times U) \subseteq \{0, 1\}$.*

Proof.

- If Θ is derivable from Σ, then Θ is marked since Σ is marked and $T - restricted$ (namely there is not sink transition in Σ). Every marking reachable from $M_{\bullet\Theta}$, $M_{\Theta\bullet}$ and M_2 is safe and clean since they are reachable from $M_{\bullet\Sigma}$, $M_{\Sigma\bullet}$ or M (M_2 is reachable from M, $M_{\bullet\Theta} = M_{\bullet\Sigma}$) and $M_{\Theta\bullet} = M_{\Sigma\bullet}$.
- Every marking reachable from M is safe, that is $\forall t \in U : \forall s \in ({}^{\bullet}t \cup t^{\bullet}) : M(s) \leq 1$. This means if there are two transitions in U, which are not independent, then U can not be enabled.
- The proof follows from the proof of the previous item.

$\square$

Agenda

Definition 72 (operator box)

$\Omega = (S, T, W, \lambda, M)$ *Petri box is an operator box if for every* $t \in T$ *transition* $\lambda(t)$ *is a transformational relabelling.*

Definition 73 (complex marking)

Let be Ω *an operator box. A complex marking of* Ω *is a pair* $\mathcal{M} = (M, Q)$*, where* M *is a normal marking of* Ω *and* Q *is a final multiset of activated transitions of* Ω*.*

Note 21

A normal marking M *of an operator box can be represented as a complex marking* $(M, \emptyset)$*.*

Note 22

Complex markings are useful for operator boxes, since a transition of an operator box can represent complex program part (even infinite loop) so their execution can take measurable time.

Definition 74 (step – using complex markings)

Let be $\mathcal{M} = (M, Q)$ a complex marking. A step U is enabled in $\mathcal{M}$ if it is enabled in M. Notation: $\mathcal{M}[U\!>$.

Definition 75 (complete step execution)

Let be U an enabled step in $\mathcal{M} = (M, Q)$ complex marking. The complete execution of U produces the complex marking $\mathcal{M}' = (M', Q)$, where

$$\forall s \in S : \; M'(s) = M(s) + \sum_{t \in U} U(t) * (W(t,s) - W(s,t)).$$

Notation: $\mathcal{M}[U\!> \mathcal{M}'$

Definition 76 (step activization)

Let be U an enabled step in $\mathcal{M} = (M, Q)$ complex marking. The complete execution of U produces the complex marking $\mathcal{M}' = (M', Q + U)$, where

$$\forall s \in S : \; M'(s) = M(s) - \sum_{t \in U} U(t) * W(s, t).$$

Notation: $\mathcal{M}[U^{+} > \mathcal{M}'$

Definition 77 (step completion)

Let be $U \subseteq Q$ an activated step in $\mathcal{M} = (M, Q)$ complex marking. The completion of U produces the complex marking $\mathcal{M}' = (M', Q - U)$, where

$$\forall s \in S : \; M'(s) = M(s) + \sum_{t \in U} U(t) * W(t, s).$$

Notation: $\mathcal{M}[U^{-} > \mathcal{M}'$

Definition 78 (direct reachability)

$\mathcal{M}' = (M', Q')$ complex marking is directly reachable from $\mathcal{M} = (M, Q)$, if there exists finite multisets of transitions U, V and Y such that $Y \subseteq Q$, $Q' = Q + V - Y$, $\forall s \in S$:
$M(s) \geq \sum_{t \in U+V} (U(t) + V(t)) * W(s,t)$ and

$$M'(s) = M(s) + \sum_{t \in U+Y} (U(t) + Y(t)) * W(t,s) - \sum_{t \in U+V} (U(t) + V(t)) * W(s,t)$$

Notation: $\mathcal{M}[U : V^+ : Y^- > \mathcal{M}'$

Definition 79 (properties – using complex markings)

$\mathcal{M} = (M, Q)$ complex marking is safe, k-bounded and clean, if correspondingly M is safe, k-bounded and clean.

Definition 80 (Ω-tuple)

Let be $\Omega = (S, T, W, \lambda, M)$ an operator box. $\Sigma : T \to Box$ function is an Ω-tuple.

Definition 81 (notations)

Let be $\Omega = (S, T, W, \lambda, M)$ an operator box and Σ an Ω-tuple.
$\forall v \in T :$ let Σ_v denote $\Sigma(v)$.
If T is finite we can assume their exists a fixed ordering $T = \{v_1, ..., v_n\}$. In this case we can use notation $\Sigma = \{\Sigma_{v_1}, ..., \Sigma_{v_n}\}$ or $\Sigma = \{\Sigma_1, ..., \Sigma_n\}$.

Note 23

Let be $\Omega = (S, T, W, \lambda, M)$ *an operator box with complex marking* $\mathcal{M} = (M, Q)$*. The operation defined by* Ω *applicable for a* Σ Ω*-tuple if for every* $v \in T : \Sigma_v$ *is marked if and only if* $v \in Q$*.*

Definition 82 (interface change – Ω-tuple)

Let be $\Omega = (S, T, W, \lambda, M)$ *an operator box and* Σ *an* Ω*-tuple. Interface change of* Σ *according to* Ω *executes an interface change for every* Σ_v *from* Σ *according to the* $\lambda(v)$ *relabelling of the corresponding* $v \in T$ *transition.*

Definition 83 (notation)

Let be $\rho_\alpha = \{(\emptyset, \alpha)\}$ *a constant relabelling. We can use the following notation:* $\underline{\rho_\alpha} = \alpha$

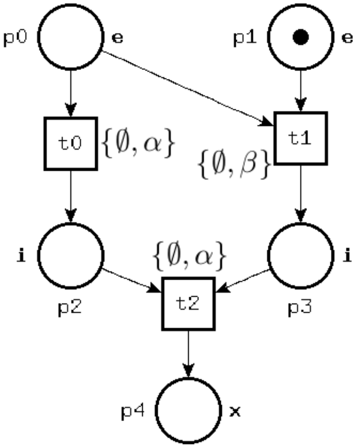
Definition 84 (interface change – plain box)

Let be $\Sigma_v = (S, T, W, \lambda, M)$ a plain box and λ_v a transformational relabelling. Interface change of Σ_v according to relabelling λ_v results the plain box $\Sigma'_v = (S, T', W', \lambda', M)$, where $\forall s \in S : \lambda'(s) = \lambda(s)$ and T', W' and $\forall t' \in T', \lambda(t')$ are created in the following way.
For all set of transitions $U \in \mathcal{P}(T)$: if the bag $U_\lambda = \left(\underset{t \in U}{+} \{\underline{\lambda(t)}\}\right)$ is in the domain of λ_v a new t' is created to T' (as a composition of transitions from set U) in the following way.

- $\lambda'(t') = \{(\emptyset, \lambda_v(U_\lambda))\}$
- $\forall s \in S : W'(s, t') = \underset{t \in U}{+} W(s, t)$
- $\forall s \in S : W'(t', s) = \underset{t \in U}{+} W(t, s)$

Example

Consider the following plain box and the transformational relabelling $\rho = \{(\{\alpha\},\gamma),(\{\alpha,\alpha\},\alpha),(\{\alpha,\beta\},\beta)\}$.

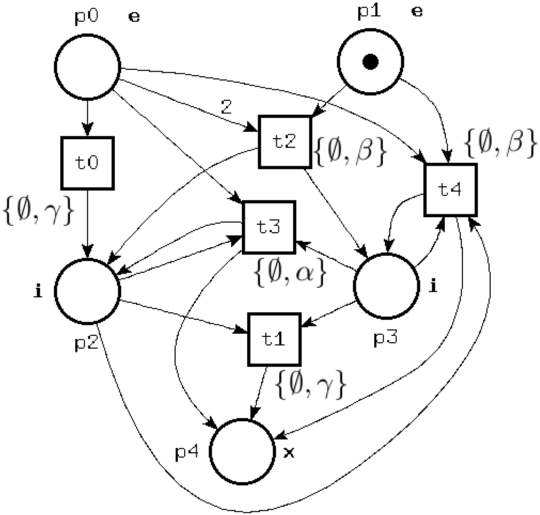


According to definition 84 we can create the following table.

sets of transitions	bags of labels	ρ	transition in the result
$\emptyset$	$\emptyset$	–	–
$\{t0\}$	$\{\alpha\}$	γ	$t0$
$\{t1\}$	$\{\beta\}$	–	–
$\{t2\}$	$\{\alpha\}$	γ	$t1$
$\{t0, t1\}$	$\{\alpha, \beta\}$	β	$t2$
$\{t0, t2\}$	$\{\alpha, \alpha\}$	α	$t3$
$\{t1, t2\}$	$\{\alpha, \beta\}$	β	$t4$
$\{t1, t2\}$	$\{\alpha, \alpha, \beta\}$	–	–

It shows that the plain box created by the interface change will contain 5 various transitions, illustrating five various compositions of the sets of transitions where the domain of function ρ contains the corresponding bag of labels.

The result of the interface change is the following plain box.



Agenda

Definition 85 (transition refinement)

Let be $\Omega = (S, T, W, \lambda, M)$ *an operator box and* Σ *an* Ω*-tuple. Let be* $\Sigma = \{\Sigma_1, \Sigma_2, \ldots, \Sigma_n\}$ *and*

$\Sigma_1 = (S_1, T_1, W_1, \lambda_1, M_1)$,

$\ldots$,

$\Sigma_n = (S_n, T_n, W_n, \lambda_n, M_n)$ *correspondingly.*

Transition refinement of Σ *according to* Ω *creates the plain box* $\Sigma_\Omega = (S_{\Sigma_\Omega}, T_{\Sigma_\Omega}, W_{\Sigma_\Omega}, \lambda_{\Sigma_\Omega}, M_{\Sigma_\Omega})$ *by composing* $\Sigma_1, \Sigma_2, \ldots, \Sigma_n$ *in the following way.*

- $T_{\Sigma_\Omega} = \bigcup\limits_{i \in [1,n]} T_i$
- $\forall t \in T_{\Sigma_\Omega} : \lambda_{\Sigma_\Omega}(t) = \lambda_i(t)$ *if* $t \in T_i$
- $\ddot{\Sigma}_\Omega = \bigcup\limits_{i \in [1,n]} \ddot{\Sigma}_i$

- $\forall s \in \ddot{\Sigma}_\Omega$:
  - $\lambda_{\Sigma_\Omega}(s) = i$,
  - $M_{\Sigma_\Omega}(s) = M_i(s)$ *if* $s \in S_i$
  - $\forall t \in T_{\Sigma_\Omega}$:
    - $W_{\Sigma_\Omega}(t,s) = \begin{cases} W_i(t,s) & \textit{if } t \in T_i \textit{ and } s \in S_i \\ 0 & \textit{if } t \in T_j \textit{ and } s \in S_i, j \neq i \end{cases}$
    - $W_{\Sigma_\Omega}(s,t) = \begin{cases} W_i(s,t) & \textit{if } t \in T_i \textit{ and } s \in S_i \\ 0 & \textit{if } t \in T_j \textit{ and } s \in S_i, j \neq i \end{cases}$
- $S_{\Sigma_\Omega} = \ddot{\Sigma}_\Omega \cup S^{new}_{\Sigma_\Omega}$
- $S^{new}_{\Sigma_\Omega}$ *and* $\forall s \in S^{new}_{\Sigma_\Omega} : \lambda_{\Sigma_\Omega}(s), M_{\Sigma_\Omega}(s)$ *and the connected arcs are created by applying the following method according to every* $p_j \in S$.

Let be p a place from S. Transition refinement of Σ according to p creates new places Σ_p^{new} (with corresponding marking, relabelling and connected arcs) in the following way.
Let us suppose ${}^{\bullet}p = \{v_{i_1}, \ldots, v_{i_k}\}$ *and* $p^{\bullet} = \{v_{j_1}, \ldots, v_{j_m}\}$

$$\Sigma_p^{new} = \Big\{ comp(\{s_{i_1}, \ldots, s_{i_k}, s_{j_1}, \ldots, s_{j_m}\}) \mid s_{i_1} \in \Sigma(v_{i_1})^{\bullet}, \ldots, s_{i_k} \in \Sigma(v_{i_k})^{\bullet}, s_{j_1} \in {}^{\bullet}\Sigma(v_{j_1}), \ldots, s_{j_m} \in {}^{\bullet}\Sigma(v_{j_m}) \Big\}, \text{ where}$$

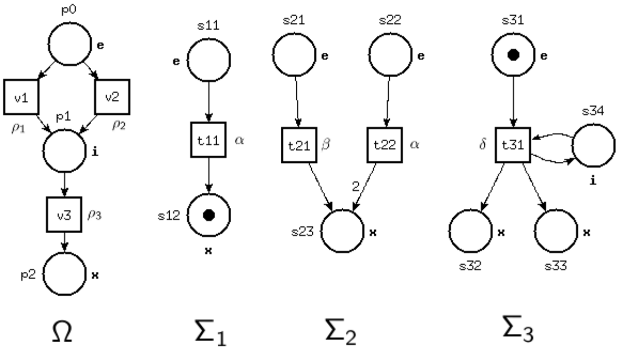
$comp(\{s_{i_1}, \ldots, s_{i_k}, s_{j_1}, \ldots, s_{j_m}\})$ *is a new place with properties*

- $\lambda_{\Sigma_\Omega}(comp(\{s_{i_1}, \ldots, s_{i_k}, s_{j_1}, \ldots, s_{j_m}\})) = \lambda(p)$
- $M_{\Sigma_\Omega}(comp(\{s_{i_1}, \ldots, s_{i_k}, s_{j_1}, \ldots, s_{j_m}\}))$
  $= \Big(\sum_{f=1}^{k} M(s_{i_f}) \Big) + \Big(\sum_{g=1}^{m} M(s_{j_g}) \Big)$

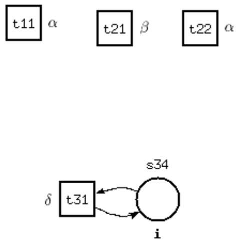
- $\forall t \in T_{\Sigma_\Omega}$:
  let be $l \in [1, n]$ *where* $t \in T_l$
  - $W_{\Sigma_\Omega}(comp(\{s_{i_1}, \ldots, s_{i_k}, s_{j_1}, \ldots, s_{j_m}\}), t)$
    $= \big(\sum_{f=1}^{k} \chi(i_f = l) * W_l(s_{i_f}, t)\big) + \big(\sum_{g=1}^{m} \chi(j_g = l) * W_l(s_{j_g}, t)\big)$
  - $W_{\Sigma_\Omega}(t, comp(\{s_{i_1}, \ldots, s_{i_k}, s_{j_1}, \ldots, s_{j_m}\}))$
    $= \big(\sum_{f=1}^{k} \chi(i_f = l) * W_l(t, s_{i_f})\big) + \big(\sum_{g=1}^{m} \chi(j_g = l) * W_l(t, s_{j_g})\big)$

Example

Consider the following operator box Ω and the Ω-tuple $\Sigma = \{\Sigma_1, \Sigma_2, \Sigma_3\}$.



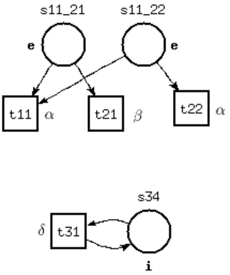
According to definition 85 if we calculate the transition refinement of Σ according to Ω first we can copy all the transitions and internal places of Σ_1, Σ_2 and Σ_3 into the new plain box (with the corresponding relabellings and markings).



Then the described composition method have to be applied according to every place of Ω.

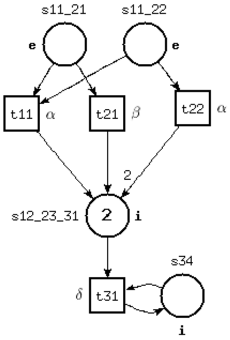
P_0	preset	corresponding plain box	exit places
	–	–	–
	postset	corresponding plain box	entry places
	v1	Σ_1	$s11$
	v2	Σ_2	$s21, s22$

The new composed places are $s11\_21$ and $s11\_22$.



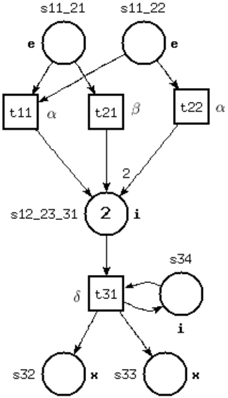
P_1	preset	corresponding plain box	exit places
	v1	Σ_1	$s12$
	v2	Σ_2	$s23$
	postset	corresponding plain box	entry places
	v3	Σ_3	$s31$

The new composed place is $s12\_23\_31$.



P_2	preset	corresponding plain box	exit places	
	v3	Σ_3	$s32, s33$	The
	postset	corresponding plain box	entry places	
	–	–	–	

new composed places are $s32$ and $s33$. (In this case we practicaly just copy the two old places into the new plain box.)



Definition 86 (net refinement)

Let be $\Omega = (S, T, W, \lambda, M)$ an operator box and $\Sigma = \{\Sigma_1, \Sigma_2, \ldots, \Sigma_n\}$ an Ω-tuple. The net refinement of Σ according to Ω in the first step calculates the interface change of Σ according to Ω. And then it calculates the transition refinement of Ω-tuple $\Sigma' = \{\Sigma'_1, \Sigma'_2, \ldots, \Sigma'_n\}$ according to Ω, where $\Sigma'_1, \Sigma'_2, \ldots, \Sigma'_n$ are the results of the first step.

Note 24

Operator boxes can be defined for describing the construction of well-know program structures (sequence, branch, loop, parallel structure) and transformations (renaming, synchronization). This makes it possible to calculate the petri net representation of a complex program by defining the representation of the basic elements (for example the actions) and applying the corresponding operator boxes for the program constructs.

Theme III

Part II

Agenda

Labelled Transition Systems

Definition 87 (Labelled Transition System)

A Labelled Transition System is a triple $(C, A, \rightarrow)$, *where*

- C *is a set of configurations (states),*
- A *is a set of actions, and*
- $\rightarrow$ *is a transition relation* $(\rightarrow \subseteq C \times A \times C)$

Notations

- $c \xrightarrow{a} c' \; : < c, a, c' > \in \rightarrow$
- $\forall a \in A : \xrightarrow{a} = \{(c, c') | < c, a, c' > \in \rightarrow\}$
- $c \rightarrow c' \; : \; \exists a \in A : c \xrightarrow{a} c'$
- $c \xrightarrow{a} \; : \; \exists c' \in C : c \xrightarrow{a} c'$
- $c \not\rightarrow : \; \nexists c' \in C : c \rightarrow c'$
- $\rightarrow^* \subseteq C \times A^* \times C$ is the transitive closure of $\rightarrow$

Labelled Transition Systems

An example.
A is an arbitrary set.
Definition of C is inductive:

- $nil \in C$,
- $ap \in C$, if $a \in A, p \in C$,
- $p + q \in C$, if $p, q \in C$,
- C is the smallest set satisfying the previous 3 rules.

Definition of $\rightarrow$ is also inductive:

- $ap \xrightarrow{a} p$, where $a \in A,\ p \in C$,
- $\frac{p \xrightarrow{a} p'}{p+q \xrightarrow{a} p'}$, where $a \in A,\ p, q, p' \in C$,
- $\frac{p \xrightarrow{a} p'}{q+p \xrightarrow{a} p'}$, where $a \in A,\ p, q, p' \in C$,
- $\rightarrow$ is the smallest set satisfying the previous 3 rules.

Labelled Transition Systems

Semantics

- Operational semantics
  - consider the meaning of program steps
  - useful for implementation
- Denotational semantics
  - consider the program as a whole
  - from parts to complete (useful for program synthesis)
- Axiomatic semantics
  - basic properties of the program
  - useful for verification

LTS operational semantics

Consider the process with its environment.

- $(p, e) \in C \times C$
- $p||e : \frac{p \xrightarrow{a} p',\ e \xrightarrow{a} e'}{p||e \xrightarrow{a} p'||e'}$

Definition 88

The process p corresponds to an environment e (p sat e), if and only if $\forall p', e' \in C : \frac{p||e \xrightarrow{a} p'||e' \text{ and } p'||e' \not\to}{e' = nil}$.

(In every case the environment is reduceable into nil.)

Definition 89 (Equivalence in operational semantics)

Two processes p and q are equivalent according to the operational semantics (p equ_o q), if and only if $\forall e \in C : p \ sat \ e \Leftrightarrow q \ sat \ e$

LTS denotational semantics

Definition 90

$\tau : P \to \mathcal{P}(A^*)$

- $\tau(nil) = \epsilon$
- $\forall a \in A, p \in P : \tau(ap) = a\tau(p)$,
  where $aT = \{at | t \in T\}$ *(*$T \subseteq A^*$*)*
  every sequence which can be produced by 'p' with an additional 'a' in the beginning
- $\forall p, q \in P : \tau(p + q) = \tau(p) \cup \tau(q)$

Definition 91 (Equivalence in denotational semantics)

Two processes p and q are equivalent according to the denotational semantics ($p\ equ_d\ q$*), if and only if* $\tau(p) = \tau(q)$

Relationship between different semantics

Theorem 33

The operational and the denotational semantics of LTS are not equivalent, $\exists p, q \in P : (p \ equ_o \ q) \not\Leftrightarrow (p \ equ_d \ q)$

Proof.

$\exists p, q \in P : (p \ \mathrm{equ_d} \ q) \not\Rightarrow (p \ \mathrm{equ_o} \ q)$:

$(a(p+q) \ \mathrm{equ_d} \ ap + aq)$, but $\neg(a(p+q) \ \mathrm{equ_o} \ ap + aq)$, (where $a \in A, p, q \in P$)

$(a(p+q) \ \mathrm{equ_d} \ ap + aq)$:

- $\tau(a(p+q)) = a\tau(p+q) = a(\tau(p) \cup \tau(q)) = a\tau(p) \cup a\tau(q)$
- $\tau(ap+aq) = \tau(ap) \cup \tau(aq) = a\tau(p) \cup a\tau(q)$

Relationship between different semantics

$\neg(a(p+q)\ \mathrm{equ_o}\ ap+aq)$:

- let $p = ap_1$, and $q = bq_1$ (where $p_1, q_1 \in P$, $b \in A$, and $a \neq b$),
- let $e = aanil$,
- $a(p+q)\ \mathrm{sat}\ e$: $(a(ap_1 + bq_1) \parallel aanil) \xrightarrow{a} (ap_1 + bq_1 \parallel anil) \xrightarrow{a} (p_1 \parallel nil)$
- $\neg(ap + aq\ \mathrm{sat}\ e)$: $aap_1 + abq_1 \parallel aanil \xrightarrow{a} bq_1 \parallel anil \not\rightarrow$

Relationship between different semantics

$\exists p, q \in P : (p \ \mathrm{equ_o} \ q) \not\Rightarrow (p \ \mathrm{equ_d} \ q)$:

- $p \ \mathrm{equ_o} \ (p + nil)$
- $p \neq nil \ \Rightarrow \ \neg(p \ \mathrm{equ_d} \ (p + nil))$:
    - $\tau(p + nil) = \tau(p) \cup \tau(nil) = \tau(p) \cup \epsilon$

$\square$

LTS denotational semantics (alternative version)

Definition 92

$\tau' : P \to \mathcal{P}(A^*)$

- $\tau'(nil) = \epsilon$
- $\forall a \in A, p \in P : \tau'(ap) = a\tau'(p) \cup \epsilon,$
- $\forall p, q \in P : \tau'(p + q) = \tau'(p) \cup \tau'(q)$

Note 25

$\tau'(p)$ *is prefix closed.*

LTS denotational semantics (alternative version)

Definition 93 (Equivalence in denotational semantics (alternative version))

Two processes p and q are equivalent according to the modified denotational semantics ($p \; equ'_d \; q$), if and only if $\tau'(p) = \tau'(q)$

Theorem 34

$\forall p, q \in P : (p \; equ_o \; q) \Rightarrow (p \; equ'_d \; q)$:

LTS axiomatic semantics

A1 $p + (q + r) = (p + q) + r$

A2 $p + q = q + p$

A3 $p + p = p$

A4 $p + nil = p$

A5 $a(p + q) = ap + aq$

Definition 94 (Equivalence in axiomatic semantics)

Two processes p and q are equivalent according to the axiomatic semantics ($p \ equ_a \ q$), if and only if p is transformable to q using axioms A1-A5.

Relationship between different semantics

Theorem 35

$\forall p, q \in P : (p \ equ_a \ q) \Leftrightarrow (p \ equ'_d \ q) :$

Definition 95 (Weak equivalence in axiomatic semantics)

Two processes p and q are weak equivalent according to the axiomatic semantics ($p \ equ\_w_a \ q$), if and only if p is transformable to q using axioms A1-A4.

Theorem 36

$\forall p, q \in P : (p \ equ\_w_a \ q) \Rightarrow (p \ equ_o \ q) :$

Note 26

$\forall a, b, c \in A : \ abnil + acnil \ equ_o \ (abnil + acnil) + a(bnil + cnil)$, but $\neg(abnil + acnil \ equ\_w_a \ (abnil + acnil) + a(bnil + cnil))$

Agenda

Syntax of CSP

Definition 96 (Syntax of CSP)

Let Com be the set of the communication events.
Let Id be the set of process identifiers.
The set of CSP processes
$PROC = \{p \mid p \in Rec \ \wedge \ FV(p) = \emptyset\}$, *where*
$FV(expr)$ *is the set of free variables of* $expr$,
Rec *is the minimal set satisfying the following:*

- $STOP \in Rec$ *(deadlock or endpoint),*
- $DIV \in Rec$ *(divergence),*
- $a \to P \in Rec$ *(prefix), where*
  - $a \in Com$ *and*
  - $P \in Rec$,

Syntax of CSP

- $(x_1 \to P_1 | x_2 \to P_2 | \dots | x_n \to P_n) \in Rec$ *(choice) ,where*
  - $n \in \mathcal{N}$,
  - $x_1, x_2, \dots, x_n \in Com$,
  - $x_1 \neq x_2 \neq \dots \neq x_n$ $(x_1, x_2, \dots, x_n$ *are distinct events) and*
  - $P_1, P_2, \dots, P_n \in Rec$,
- $P \sqcap Q$ *(nondeterministic or), where*
  - $P, Q \in Rec$
- $P \Box Q$ *(general choice), where*
  - $P, Q \in Rec$
- $P || Q$ *(concurrency), where*
  - $P, Q \in Rec$

Syntax of CSP

- $rec\ X.P$ *(recursion), where*
  - $X \in Id$
  - $P \in Rec$
- $X \in Rec$ *(variable), where*
  - $X \in Id$
- $f(P) \in Rec$ *(renaming), where*
  - $f : \alpha P \to Com$
  - $P \in Rec$
- $P \setminus C \in Rec$ *(concealment), where*
  - $C \subseteq Com$
  - $P \in Rec$

Alphabet of a CSP process

αP is the alphabet of process P

- the process is equiped with the physical capabilities to engage in these events.

Note 27

- *$STOP_A$ is the process which is equipped with the physical capabilities to engage in the events of A, but it never exercises those capabilities,*
- *$STOP_A \neq STOP_B$ if $A \neq B$,*
- *$\alpha(a \to P) = \alpha P$, $(a \in \alpha P)$,*

Alphabet of a CSP process

- $\alpha(a_1 \to P_1 | a_2 \to P_2 | \ldots | a_n \to P_n) = \alpha P_1 = \cdots = \alpha P_n$, $(\{a_1, a_2, \ldots, a_n\} \subseteq \alpha P_1)$,
- $\alpha(P || Q) = \alpha P \cup \alpha Q$,
- $\alpha(f(P)) = f(\alpha P)$ (where $f : \alpha P \to A$),
- $\alpha(P \sqcap Q) = \alpha P = \alpha Q$,
- $\alpha(P \Box Q) = \alpha P = \alpha Q$,
- $\alpha(P \setminus C) = (\alpha P) \setminus C$.

Menu

Note 28

- *The choice is not an operator on processes, the following are incorrect:*
  - $(P|Q)$
  - $(x \rightarrow P|x \rightarrow Q)$,
  - $(x \rightarrow P|y \rightarrow Q|R)$,
  - $((x \rightarrow P|y \rightarrow Q)|z \rightarrow R))$

Definition 97 (Menu)

$x : B \rightarrow P(x)$, where $B \subseteq COM$ and $\forall x \in B : P(x) \in PROC$ is a generalization of choice. First it offers a choice of any event e in B, and then behaves like $P(e)$.

Menu

Note 29

- $x : B \rightarrow P(x) = y : B \rightarrow P(y)$
- $x : \{\} \rightarrow P = STOP_{\alpha P}$
- $x : \{a\} \rightarrow P = a \rightarrow P$
- $x : B \rightarrow P(x) = (a_1 \rightarrow P_1 | a_2 \rightarrow P_2 | \ldots | a_n \rightarrow P_n),$
  if $B = \{a_1, a_2, \ldots, a_n\}$
  and $P(a_1) = P_1,\ P(a_2) = P_2,\ \ldots,\ P(a_n) = P_n$

Recursion

$P = recX.F(X) \sim P = F(P)$

Example

- $CLOCK = tick \rightarrow CLOCK$, $\alpha CLOCK = \{tick\}$

$$\begin{aligned} & CLOCK \\ = \ & tick \rightarrow CLOCK \\ = \ & tick \rightarrow tick \rightarrow CLOCK \\ = \ & tick \rightarrow tick \rightarrow tick \rightarrow CLOCK \\ = \ & \ldots \end{aligned}$$

Recursion

$X = F(X),\ \alpha X = A$ is well defined if the equation has a unique solution with alphabet A.
$\mu X : A.F(X)$ denotes this solution.

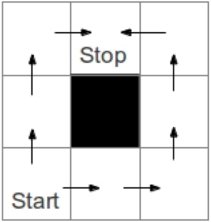
Note 30

- $\mu X : A.F(X) = \mu Y : A.F(Y)$
- $CLOCK = \mu X : \{tick\}.tick \rightarrow X$

Examples

- $P = (up \to up \to right \to STOP$
  $|right \to right \to up \to up \to left \to STOP)$,

  $\alpha P = \{up, right, left, down\}$



Examples

- $VMS = coin \to choc \to VMS, \quad \alpha VMS = \{coin, choc\}$
- $RUN_A = x : A \to RUN_A, \quad \alpha RUN_A = A$
- $P = LEVEL_0$
  $LEVEL_0 = (around \to LEVEL_0 | up \to LEVEL_1)$
  $LEVEL_i = (up \to LEVEL_{i+1} | down \to LEVEL_{i-1})$, where $i \in \mathcal{N}$

  $\alpha P = \{around, up, down\}$

Examples

Mutual recursion.

$TV = (set_{BBC} \rightarrow BBC \mid set_{MTV} \rightarrow MTV)$
$BBC = (watching_{BBC} \rightarrow BBC \mid turn_{off} \rightarrow TV \mid set_{MTV} \rightarrow MTV)$
$MTV = (watching_{MTV} \rightarrow MTV \mid turn_{off} \rightarrow TV \mid set_{BBC} \rightarrow BBC)$

$\alpha TV = \{set_{BBC}, set_{MTV}, watching_{BBC}, watching_{MTV}, turn_{off}\}$

Agenda

Axiomatic semantics (menu and recursion)

Definition 98 (Guarded process)

A process P is guarded if it begins with a prefix.

Ax.1. Let be $Pr_1 = x : A \rightarrow P(x)$ and $Pr_2 = y : B \rightarrow Q(y)$ $Pr_1 = Pr_2$ if, and only if $\alpha Pr_1 = \alpha Pr_2,\ A = B$ and $\forall x \in A : P(x) = Q(x)$.

Ax.2. If $F(X)$ is a guarded expression containing the process name X, and A is the alphabet of X, then $X = F(X)$ has a unique solution with alphabet A.
$(X = F(X) \iff X = \mu Y : A.F(Y))$

Application of Ax.1.

- $STOP \neq (a \rightarrow P)$
  $STOP = (x : \{\} \rightarrow P) \; \neq \; (x : \{a\} \rightarrow P) = (a \rightarrow P)$
- $(a \rightarrow P) \neq (b \rightarrow Q)$, if $a \neq b$
  $(a \rightarrow P) = (x : \{a\} \rightarrow P) \; \neq (x : \{b\} \rightarrow Q) = (b \rightarrow Q)$
  $(\{a\} \neq \{b\})$
- $(a \rightarrow P | b \rightarrow Q) = (b \rightarrow Q | a \rightarrow P)$
  Let be $R(a) = P$ and $R(b) = Q$
  $(a \rightarrow P | b \rightarrow Q) = (x : \{a, b\} \rightarrow R(x)) = (b \rightarrow Q | a \rightarrow P)$
- $(a \rightarrow P) = (a \rightarrow Q) \iff P = Q$

Application of Ax.2.

- $\mu X : A.F(X) = F(\mu X : A.F(X))$, (if $F(X)$ is guarded),
- Let be $VM_1 = coin \rightarrow VM_2$ and $VM_2 = choc \rightarrow VM_1$
  $VM_1 = VMS$
  $VM_1 = coin \rightarrow VM_2 = coin \rightarrow choc \rightarrow VM_1$
  $VM_1 = \mu X : \{coin, choc\}.coin \rightarrow choc \rightarrow X = VMS$
- $\mu X.coin \rightarrow (choc \rightarrow X \mid toffee \rightarrow X)$
  $= \mu X.coin \rightarrow (toffee \rightarrow X \mid choc \rightarrow X)$

  $((choc \rightarrow X \mid toffee \rightarrow X) \ = \ (toffee \rightarrow X \mid choc \rightarrow X))$

Axiomatic semantics (concurrency)

Ax.3. $P||Q \;=\; Q||P.$

Ax.4. $P||(Q||R) \;=\; (P||Q)||R.$

Ax.5. $P||STOP_{\alpha P} \;=\; STOP_{\alpha P}.$

Ax.6. $P||RUN_{\alpha P} \;=\; P.$

Ax.7. $(c \rightarrow P) \;||\; (c \rightarrow Q) \;=\; (c \rightarrow (P||Q)).$

Ax.8. $(c \rightarrow P) \;||\; (d \rightarrow Q) \;=\; STOP,$
if $c \neq d$ and $c, d \in (\alpha P \cap \alpha Q).$

Axiomatic semantics (concurrency)

Ax.9. $(a \to P) \parallel (c \to Q) \;=\; a \to (P \parallel (c \to Q))$,
if $a \in (\alpha P \setminus \alpha Q)$ and $c \in (\alpha P \cap \alpha Q)$.

Ax.10. $(c \to P) \parallel (b \to Q) \;=\; b \to ((c \to P) \parallel Q)$,
if $c \in (\alpha P \cap \alpha Q)$ and $b \in (\alpha Q \setminus \alpha P)$.

Ax.11. $(a \to P) \parallel (b \to Q)$
$\;=\; \big(b \to ((a \to P) \parallel Q) \mid a \to (P \parallel (b \to Q))\big)$,
if $a \in (\alpha P \setminus \alpha Q)$ and $b \in (\alpha Q \setminus \alpha P)$.

Ax.12. $(x : A \to P(x)) \parallel (y : B \to Q(y))$
$\;=\; z : (A \cap B) \to \big(P(z) \parallel Q(z)\big)$,
if $\alpha P = \alpha Q$.

Axiomatic semantics (concurrency)

Ax.13. Let be $P = x : A \to R(x)$ and $Q = y : B \to T(y)$ $(A \subseteq \alpha P,\ B \subseteq \alpha Q)$

$$P \mid\mid Q \;=\; z : C \to (P'(z) \mid\mid Q'(z)), \text{ where}$$

$$C \;=\; (A \cap B) \;\cup\; (A \setminus \alpha Q) \;\cup\; (B \setminus \alpha P)$$

$$P'(z) \;=\; \begin{cases} R(z) & \text{if } z \in A \\ P & \text{otherwise} \end{cases}$$

$$Q'(z) \;=\; \begin{cases} T(z) & \text{if } z \in B \\ Q & \text{otherwise} \end{cases}$$

Examples

- $$P = (a \to b \to P \mid b \to P), \quad (\alpha P = \{a, b, c\})$$
  $$Q = (a \to (b \to Q \mid c \to Q)), \quad (\alpha Q = \{a, b, c\})$$
  $$P \mid\mid Q = a \to \big(b \to P \mid\mid (b \to Q \mid c \to Q)\big)$$
  $$= a \to b \to (P \mid\mid Q)$$
  $$= \mu X : \{a, b, c\}.a \to b \to X$$

- $$NOISYVM$$
  $$= coin \to clink \to choc \to clunk \to NOISYVM,$$
  $$(\alpha NOISYVM = \{coin, choc, clink, clunk, toffee\})$$
  $$CUST$$
  $$= coin \to (toffee \to CUST \mid curse \to choc \to CUST),$$
  $$(\alpha CUST = \{coin, choc, curse, toffee\})$$
  $$NOISYVM \mid\mid CUST$$
  $$= \mu X : \{coin, choc, clink, clunk, toffee, curse\}.coin \to$$
  $$(\ clink \to curse \to choc \to clunk \to X$$
  $$\mid curse \to clink \to choc \to clunk \to X)$$

Examples

- $P \;=\; up \rightarrow down \rightarrow P, \quad (\alpha P = \{up, down\})$
  $Q \;=\; (left \rightarrow right \rightarrow Q \mid right \rightarrow left \rightarrow Q),$
  $(\alpha Q = \{left, right\})$

  $P \parallel Q = R_{12}$, where

$$
\begin{aligned}
R_{12} &= (up \rightarrow R_{22} \mid left \rightarrow R_{11} \mid right \rightarrow R_{13}) \\
R_{22} &= (down \rightarrow R_{12} \mid left \rightarrow R_{21} \mid right \rightarrow R_{23}) \\
R_{11} &= (up \rightarrow R_{21} \mid right \rightarrow R_{12}) \\
R_{21} &= (down \rightarrow R_{11} \mid right \rightarrow R_{22}) \\
R_{13} &= (up \rightarrow R_{23} \mid left \rightarrow R_{12}) \\
R_{23} &= (down \rightarrow R_{13} \mid left \rightarrow R_{22})
\end{aligned}
$$

Examples

- $P = a \rightarrow c \rightarrow P, \quad (\alpha P = \{a, c\})$
  $Q = c \rightarrow b \rightarrow Q, \quad (\alpha Q = \{b, c\})$
  $$\begin{aligned} P \parallel Q = \ & (a \rightarrow c \rightarrow P) \parallel (c \rightarrow b \rightarrow Q) \\ = \ & a \rightarrow (c \rightarrow P \parallel c \rightarrow b \rightarrow Q) \\ = \ & a \rightarrow c \rightarrow (P \parallel b \rightarrow Q) \end{aligned}$$
  $$\begin{aligned} P \parallel b \rightarrow Q \ = \ & (a \rightarrow c \rightarrow P) \parallel (b \rightarrow Q) \\ = & \big(a \rightarrow (c \rightarrow P \parallel b \rightarrow Q) \ \mid \ b \rightarrow (a \rightarrow c \rightarrow P \parallel Q)\big) \\ = & \big(a \rightarrow b \rightarrow ((c \rightarrow P) \parallel Q) \ \mid \ b \rightarrow (P \parallel Q)\big) \\ = & \big(a \rightarrow b \rightarrow (c \rightarrow P \parallel c \rightarrow b \rightarrow Q) \\ & \ \mid \ b \rightarrow (a \rightarrow c \rightarrow (P \parallel b \rightarrow Q))\big) \\ = & \big(a \rightarrow b \rightarrow c \rightarrow (P \parallel b \rightarrow Q) \\ & \ \mid \ b \rightarrow a \rightarrow c \rightarrow (P \parallel b \rightarrow Q)\big) \\ = & \mu X : \{a, b, c\}.(a \rightarrow b \rightarrow c \rightarrow X \mid b \rightarrow a \rightarrow c \rightarrow X) \end{aligned}$$
  $$\begin{aligned} P \parallel Q \ = \ a \rightarrow c \rightarrow \big(\mu X : \{a, b, c\}.(& a \rightarrow b \rightarrow c \rightarrow X \\ & \mid b \rightarrow a \rightarrow c \rightarrow X)\big) \end{aligned}$$

Axiomatic semantics (renaming)

Let be $f(A) = \{f(x) \mid x \in A\}$,
where $A \subseteq Com$ and $f : Com \to Com$. Let be f^{-1} is the inverse of f.

Ax.14. $f(STOP_A) = STOP_{f(A)}$.

Ax.15. $f(x : B \to P(x)) = y : (f(B)) \to P(f^{-1}(y))$.

Ax.16. $f(P||Q) = f(P)||f(Q)$.

Ax.17. $f(\mu X : A.F(X)) = \mu Y : f(A).F(f^{-1}(Y))$.

Ax.18. $f(g(P)) = f \circ g(P)$,
where $f \circ g$ is the composition of f and g.

Axiomatic semantics (nondeterministic or)

Ax.19. $P \sqcap P = P.$

Ax.20. $P \sqcap Q = Q \sqcap P.$

Ax.21. $(P \sqcap Q) \sqcap R = P \sqcap (Q \sqcap R).$

Ax.22. $x \rightarrow (P \sqcap Q) = (x \rightarrow P) \sqcap (x \rightarrow Q)$

Ax.23. $x : B \rightarrow (P(x) \;\sqcap\; Q(x))$
$= \; (x : B \rightarrow P(x)) \;\sqcap\; (x : B \rightarrow Q(x)).$

Ax.24. $P \;||\; (Q \sqcap R) \;=\; (P||Q) \;\sqcap\; (P||R).$

Ax.25. $(P \sqcap Q) \;||\; R \;=\; (P||R) \;\sqcap\; (Q||R).$

Ax.26. $f(P \sqcap Q) \;=\; f(P) \;\sqcap\; f(Q).$

Axiomatic semantics (general choice)

Ax.27. $P \,\Box\, P \;=\; P.$

Ax.28. $P \,\Box\, Q \;=\; Q \,\Box\, P.$

Ax.29. $(P \,\Box\, Q) \,\Box\, R \;=\; P \,\Box\, (Q \,\Box\, R).$

Ax.30. $P \,\Box\, STOP \;=\; P.$

Ax.31. $(x : A \to P(x)) \,\Box\, (y : B \to Q(y))$
$\;=\; z : (A \cup B) \to R(z),$

$$\text{where } R(z) \;=\; \begin{cases} P(z) & \text{if } z \in A \setminus B \\ Q(z) & \text{if } z \in B \setminus A \\ P(z) \;\sqcap\; Q(z) & \text{if } z \in A \cap B \end{cases}$$

Ax.32. $P \,\Box\, (Q \sqcap R) \;=\; (P \,\Box\, Q) \;\sqcap\; (P \,\Box\, R).$

Ax.33. $P \sqcap (Q \,\Box\, R) \;=\; (P \sqcap Q) \,\Box\, (P \sqcap R).$

Axiomatic semantics (concealment)

Ax.34. $P \setminus \{\} \ = \ P.$

Ax.35. $(P \setminus B) \setminus C \ = \ P \setminus (B \cup C).$

Ax.36. $(P \sqcap Q) \setminus C \ = \ (P \setminus C) \sqcap (Q \setminus C).$

Ax.37. $(STOP_A) \setminus C \ = \ STOP_{A \setminus C}.$

Ax.38. $(x \to P) \setminus C \ = \ \begin{cases} x \to (P \setminus C) & \text{if } x \notin C \\ (P \setminus C) & \text{if } x \in C \end{cases}$

Ax.39. $(P || Q) \setminus C \ = \ (P \setminus C) \mid\mid (Q \setminus C),$
if $\alpha P \cap \alpha Q \cap C \ = \ \{\}.$

Ax.40. $f(P \setminus C) = f(P) \setminus f(C).$

Axiomatic semantics (concealment)

Ax.41. $(x : B \to P(x)) \setminus C \ = \ x : B \to (P(x) \setminus C)$,
if $B \cap C = \{\}$.

Ax.42. $(x : B \to P(x)) \setminus C \ = \ \underset{x:B}{\sqcap}(P(x) \setminus C)$,
if $B \subseteq C$, and B is finite and not empty.

Ax.42. $(x : B \to P(x)) \setminus C$

$$= \ Q \ \sqcap \ \Big(Q \,\Box\, \big(x : (B \setminus C) \to (P(x) \setminus C)\big)\Big),$$

where $Q \ = \ \underset{x:B\cap C}{\sqcap}(P(x) \setminus C)$,
if $C \cap B$ is finite and not empty.

Note 31

There is no general axiom for $(P \,||\, Q) \setminus C$ and for $(P \,\Box\, Q) \setminus C$.

Agenda

Traces

Definition 99 (Trace of a process)

A trace t of a process P is a finite sequence of events in which the process has engaged up to some moment in time.
($t \in (\alpha P)^*$)

Auxiliary functions.
Let be s, t and u traces. $(s, t, u \in Com^*)$

- $s^\wedge t$ – concatenation of s and t.
- t^n – n times concatenation of t.
  - $t^0 = <>$,
  - $t^{n+1} = t^\wedge t^n$
- $t \uparrow A$ – restriction to A ($A \subseteq Com$).
  - $<> \uparrow A = <>$,
  - $(s^\wedge t) \uparrow A \ = \ (s \uparrow A)^\wedge (t \uparrow A)$,
  - $< x > \uparrow A = \begin{cases} < x > & \text{if } x \in A \\ <> & \text{if} x \notin A \end{cases}$

Traces

- t_0 – head of t,
  - $(< x >^\wedge s)_0 = x$.
- t' – tail of t,
  - $(< x >^\wedge s)' = s$.
- $s \leq t$ – prefix,
  - $s \leq t \;\; = \;\; \big(\exists u : (s^\wedge u) = t\big)$.
- $s \; in \; t$ – infix,
  - $s \; in \; t \;\; = \;\; \big(\exists u, v : (u^\wedge s^\wedge v) = t\big)$.
- $\#t$ – length of t.
- $t \downarrow x$ – the number of occurencies of x in t,
  - $t \downarrow x = \#(t \uparrow \{x\})$.

Denotational semantics of processes

Definition 100 (Equivalence in denotational semantics.)

Two CSP process P and Q are equivalent according to the denotational semantics, if $traces(P) = traces(Q)$ *(they have the same traces), where the formal definition of function traces is the following.*

Definition 101 (Traces of a process)

1. $traces(STOP) = <>$,
2. $traces(x : B \to P(x))$
   $= \{t \mid t = <> \vee \big(t_0 \in B \wedge t' \in traces(P(t_0))\big)\}$,
3. If $F(X)$ is guarded, then
   $traces(\mu X : A.F(X)) \;=\; \bigcup_{n \geq 0} traces(F^n(STOP_A))$,
   where
   - $F^0(X) = X$,
   - $F^{n+1}(X) = F(F^n(X))$,

Denotational semantics of processes

4. $traces(P \mid\mid Q)$
   $= \{t \mid (t \uparrow \alpha P) \in traces(P) \ \wedge \ (t \uparrow \alpha Q) \in traces(Q)$
   $\wedge \ t \in (\alpha P \cup \alpha Q)^*\}$,
   - If $\alpha P = \alpha Q$, then $traces(P \mid\mid Q) = traces(P) \cap traces(Q)$,
5. $traces(f(P)) = \{f^*(s) \mid s \in traces(P)\}$,

   where $f \in \alpha P \rightarrow Com$,
   $f^* \in (\alpha P)^* \rightarrow Com^*$,
   $f^*(<>) = <>$,
   $f^*(< x >) = < f(x) >$,
   $f^*(s^\wedge t) = f^*(s)^\wedge f^*(t)$,

Denotational semantics of processes

6. $traces(P \sqcap Q) = traces(P) \cup traces(Q)$,
7. $traces(P \;\Box\; Q) = traces(P) \cup traces(Q)$,
8. $traces(P \setminus C) = \{t \uparrow (\alpha P \setminus C) \mid t \in traces(P)\}$,
   if $\forall s \in traces(P) : \neg diverges(P/s, C)$,
   where $diverges(P, C)$
   $= \big(\forall n \in \mathbb{N} : (\exists t \in traces(P) \cap C^* : \#t > n)\big)$,
   and P/s is a process which behaves the same as P behaves from the time after it has engaged in all the actions recorded in s, if s is not a trace of P, (P / s) is not defined,
   $traces(P/s) = \{t \mid s^\wedge t \in traces(P)\}$, if $s \in traces(P)$.

Examples of traces

Note 32

Forall CSP process P:

- $<>\in traces(P)$
- $s^\wedge t \in traces(P) \Rightarrow s \in traces(P)$
- $traces(P) \subseteq (\alpha P)^*$
- $P \sqcap Q$ *and* $P \,\Box\, Q$ *cannot be distinguished by their traces.*

Examples

- $traces(a \rightarrow P) = \{<>\} \ \cup \ \{< a >^\wedge t \mid t \in traces(P)\}.$
- $traces(coin \rightarrow choc \rightarrow STOP)$
  $= \{<>, < coin >, < choc >\}.$
- $traces(a \rightarrow P \mid b \rightarrow Q)$
  $= \{<>\} \ \cup \ \{< a >^\wedge t \mid t \in traces(P)\}$
  $\cup \{< b >^\wedge t \mid t \in traces(Q)\}.$

Examples of traces

- $traces(RUN_A) = A^*$,
  - $RUN_A = \mu X : A.(y : A \to X)$,
    namely here $(F(X) = y : A \to X)$
  - According to the 3. item of definition of function *traces* it is enough to see:
    $\forall n \in \mathbb{N} : traces(F^n(STOP_A)) = \{s \mid s \in A^* \wedge \#s \leq n\}$.
    Using induction:

    n=0 $traces(F^0(STOP_A)) = traces(STOP_A) = \{<>\} = \{s \mid s \in A^* \wedge \#s \leq 0\}$,

    n=k+1 $traces(F^{k+1}(STOP_A))$
    $= traces(F(F^k(STOP_A)))$
    $= traces(y : A \to F^k(STOP_A))$
    $= \{t \mid t =<> \vee \left(t_0 \in A \wedge t' \in traces(F^k(STOP_A))\right)\}$
    $= \left\{t \mid t =<> \vee \left(t_0 \in A \wedge t' \in \{s \mid s \in A^* \wedge \#s \leq k\}\right)\right\}$
    $= \{t \mid t =<> \vee \left(t_0 \in A \wedge t' \in A^* \wedge \#t' \leq k\right)\}$
    $= \{t \mid t \in A \wedge \#t \leq k + 1\}$

Specifications

Definition 102 (Specification)

A specification S of a process P is a requirement for the traces of P. ($S : (\alpha P)^* \rightarrow \{TRUE, FALSE\}$.)

Definition 103 (Satisfaction)

P satisfies S, (P sat S) if $\forall tr \in traces(P) : S(tr)$.

Note 33

Let be S a specification. If there exists any process which satisfies S, then $S(<>)$ has to hold, so $STOP$ satisfies S. Namely we can specify only safety properties. (We can not specify progress properties.)

Properties of satisfaction

1. $P \; sat \; TRUE$,
2. $(\forall n \in \mathbb{N} : P \; sat \; S_n) \Longrightarrow P \; sat \; (\forall n \in \mathbb{N} : S_n)$,
3. $(P \; sat \; S \; \wedge \; S \Rightarrow T) \Longrightarrow P \; sat \; T$,
4. $(\forall x \in B : (P(x) \; sat \; S_x)$ $\Longrightarrow (x : B \rightarrow P(x)) \;\; sat \;\; ((tr = <>) \vee (tr_0 \in B \wedge S_{tr_0}(tr')))$,
5. $F(X)$ is guarded $\wedge(STOP_A \; sat \; S)$ $\wedge\forall X \in PROC, \alpha X = A : ((X \; sat \; S) \Rightarrow (F(X) \; sat \; S))$ $\Longrightarrow \mu X : A.F(X) \; sat \; S$,

Properties of satisfaction

6. $(P \ sat \ S) \ \wedge \ (Q \ sat \ T)$
   $\Longrightarrow P \,||\, Q \ sat \ (S(tr \uparrow \alpha P) \wedge T(tr \uparrow \alpha P))$,
7. $P \ sat \ S \Longrightarrow f(P) \ sat \ S(f^{-1}(tr))$,
8. $(P \ sat \ S) \ \wedge \ (Q \ sat \ T)$
   $\Longrightarrow P \ \sqcap \ Q \ sat \ (S \vee T)$,
9. $(P \ sat \ S) \ \wedge \ (Q \ sat \ T)$
   $\Longrightarrow P \ \Box \ Q \ sat \ (S \vee T)$.

Examples

- $STOP \; sat \; (tr = <>)$,
- $P \; sat \; S$
  $$\Longrightarrow (c \to d \to P \quad sat \; (tr \; \leq < c, d >) \vee (< c, d > \leq \; tr \wedge S((tr')'))$$,
- $P \; sat \; S \; \wedge Q \; sat \; T$
  $$\Longrightarrow (c \to P \mid d \to Q) \quad sat \; (tr \; = <> \vee (tr_0 = c \wedge S(tr')) \vee (tr_0 = d \wedge T(tr'))$$,

Examples

- Let be $VMS = \mu X : \{coin, choc\}.coin \rightarrow choc \rightarrow X$
  $(F(X) = coin \rightarrow choc \rightarrow X)$, and
  $VMSSPEC = (0 \leq ((tr \downarrow coin) - (tr \downarrow choc)) \leq 1)$

  $VMS \ \ sat \ \ VMSSPEC,$
  because

  1. $(tr = <>) \Rightarrow VMSSPEC$

  $\Rightarrow$ $STOP \ sat \ (tr = <>) \Longrightarrow STOP \ sat \ VMSSPEC$

  2. Suppose $X \ sat \ (0 \leq ((tr \downarrow coin) - (tr \downarrow choc)) \leq 1)$

  $\Rightarrow$ $F(X) \ sat \ ((tr \ \leq < coin, choc >)$
  $\vee ((< coin, choc > \leq \ tr)$
  $\wedge (0 \leq ((tr'' \downarrow coin) - (tr'' \downarrow choc)) \leq 1))$

  2.a $(<> \downarrow coin) - (<> \downarrow choc) = 0,$
  $(< coin > \downarrow coin) - (< coin > \downarrow choc) = 1,$
  $(< coin, choc > \downarrow coin) - (< coin, choc > \downarrow choc) = 0$

  $\Rightarrow$ $\forall t \ \leq < coin, choc >: \ (0 \leq ((t \downarrow coin) - (t \downarrow choc)) \leq 1)$

Examples

2.b Suppose for trace u $VMSSPEC(u)$ holds.

$(< coin, choc >^\wedge u \downarrow coin) - (< coin, choc >^\wedge u \downarrow choc)$
$= ((u \downarrow coin) + 1) - ((u \downarrow choc) + 1)$
$= ((u \downarrow coin) - (u \downarrow choc))$

$\Rightarrow$ $VMSSPEC(< coin, choc >^\wedge u)$ holds.

- (2.a) $\wedge$ (2.b)$\Rightarrow$ $F(X)$ sat $VMSSPEC$.
- (1.) $\wedge$ (2.) $\Rightarrow$ VMS sat $VMSSPEC$.

Agenda

Communication events

Using channels.

- Special actions: $c.v$
  - c is the name of the channel,
  - v is the value of the message.
- $channel(c.v) = c$
- $message(c.v) = v$
- $\alpha c(P) = \{v \mid c.v \in \alpha P\}$
  - potential messages on channel c.

Communication events

- Sending a value v on channel c:
  - $c!v \to P = c.v \to P$
- Receving a value from channel c into variable x:
  - $c?x \to P(x) = y : \{y \mid channel(y) = c\} \to P(message(y))$

Example 12

$$COPY = \mu X.(in?y \to out!y \to X)$$

Communication rules

Definition 104 (Communication rules)

1. $(c!v \rightarrow P \parallel c?x \rightarrow Q(x)) = c.v \rightarrow (P \parallel Q(v))$
2. $(c!v \rightarrow P \parallel c?x \rightarrow Q(x)) \setminus C = (P \parallel Q(v)) \setminus C,$
   $\text{where } C = \{y \mid channel(y) = c\}$

Example 13

$INPUT = \mu X.(in!42 \rightarrow X)$

$INPUT || COPY = \mu X.(in.42 \rightarrow out!42 \rightarrow X)$

$$(INPUT || COPY) \setminus \{y \mid channel(y) = in\} \\ = \mu X.(out!42 \rightarrow X)$$

Examples

- Simulating a variable

$$VAR = in?x \rightarrow VAR_x$$
$$VAR_x = (in?y \rightarrow VAR_y \mid out!x \rightarrow VAR_x)$$

- Simulating a dataflow multiplexer

$$MUX = (in_1?x \rightarrow out!x \rightarrow MUX \mid in_2?x \rightarrow out!x \rightarrow MUX)$$

- Simulating a dataflow branch

$$FORK = in?x \rightarrow (out_1!x \rightarrow FORK \mid out_2!x \rightarrow FORK)$$

Examples

- Simulating a buffer

$$BUFFER = P_{<>}$$

$$\begin{aligned} P_{<>} &= (empty \rightarrow P_{<>} \\ &\quad \mid in?x \rightarrow P_{<x>}) \end{aligned}$$

$$\begin{aligned} P_{<x>^\wedge xs} &= (out!x \rightarrow P_{xs} \\ &\quad \mid in?y \rightarrow P_{<x>^\wedge xs^\wedge <y>}) \end{aligned}$$

- Simulating a stack

$$STACK = P_{<>}$$

$$\begin{aligned} P_{<>} &= (empty \rightarrow P_{<>} \\ &\quad \mid in?x \rightarrow P_{<x>}) \end{aligned}$$

$$\begin{aligned} P_{<x>^\wedge xs} &= (out!x \rightarrow P_{xs} \\ &\quad \mid in?y \rightarrow P_{<y>^\wedge <x>^\wedge xs}) \end{aligned}$$

Specifications

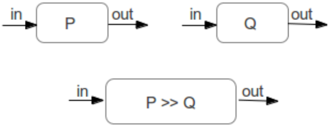
Notations

- $tr \downarrow c = message^*(tr \uparrow \alpha c)$
- Simplification: $c_1 \leq c_2$ in place of $tr \downarrow c_1 \leq tr \downarrow c_2$
- $c_1 \overset{n}{\leq} c_2 = (c_1 \leq c_2 \ \wedge \ \#c_2 \leq \#c_1 + n)$
  - $c_1 \overset{0}{\leq} c_2 \Longleftrightarrow c_1 = c_2$
  - $(c_1 \overset{n}{\leq} c_2) \wedge (c_2 \overset{m}{\leq} c_3) \ \Rightarrow \ (c_1 \overset{m+n}{\leq} c_3)$
  - $(c_1 \leq c_2) \Rightarrow \exists n \in \mathbb{N}_0 : c_1 \overset{n}{\leq} c_2$

Examples

- $MUX\ sat$
  $$(\exists r \in Com^* : out \overset{1}{\leq} r \wedge r \in interleaves(in_1, in_2))$$
- $FORK\ sat$
  $$(\exists r \in Com^* : r \overset{1}{\leq} in \wedge r \in interleaves(out_1, out_2))$$
- $BUFFER\ sat\ out \leq in$

Pipes



$P \gg Q$ is the pipes of P and Q

- $\alpha(P \gg Q) = \alpha in(P) \cup \alpha out(Q)$
- $\alpha out(P) = \alpha in(Q)$

Communication rules of Pipes

Definition 105 (Communication rules of Pipes)

1. $P \gg (Q \gg R) = (P \gg Q) \gg R$
2. $(out!v \to P) \gg (in?x \to Q(x)) = (P \gg Q(v))$
3. $(out!v \to P) \gg (out!w \to Q(x))$
   $= out!w \to ((out!v \to P) \gg Q(v))$
4. $(in?y \to P(y)) \gg (in?x \to Q(x))$
   $= in?y \to (P(y) \gg (in?x \to Q(x)))$

Communication rules of Pipes

5. $$(in?x \to P(x)) \gg (out!w \to Q) \\ = in?x \to (P \gg (out!w \to Q)) \\ \mid out!w \to ((in?x \to P(x)) \gg Q)$$

6. $$(in?x \to P(x)) \gg R \gg (out!w \to Q) \\ = in?x \to (P \gg R \gg (out!w \to Q)) \\ \mid out!w \to ((in?x \to P(x)) \gg R \gg Q)$$

7. If R is a chain of pipes all starting with sending data to channel *out*:
   $R \gg (out!w \to Q) = out!w \to (R \gg Q)$

8. If R is a chain of pipes all starting with waiting data from channel *in*:
   $(in?x \to P(x)) \gg R = in?x \to (P(x) \gg R)$

Examples

- $P = (in?x \rightarrow out!x^2 \rightarrow P)$
  $(P \gg P)\ sat\ \left(out \overset{2}{\leq} power\_four^*(in)\right)$
  ,where $power\_four(y) = y^4$

- $P = (in?x \rightarrow out!(x, x+4) \rightarrow P)$
  $Q = (in?y \rightarrow out!(y_1 * y_2) \rightarrow Q)$
  $(P \gg Q)\ sat\ \left(out \overset{2}{\leq} fv^*(in)\right)$
  ,where $fv(z) = z^2 + 4z$

Agenda

Literature

(1) Murata, Tadao. "Petri nets: Properties, analysis and applications." Proceedings of the IEEE 77.4 (1989): 541-580.

(2) Best, Eike, Raymond Devillers, and Maciej Koutny. "Petri net algebra." Springer-Verlag New York Incorporated, 2001.

(3) Hoare, Charles Antony Richard. "Communicating sequential processes.", Prentice Hall International, 1985.

(4) Jensen, Kurt, Kristensen, Lars Michael and Wells, Lisa. "Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems". Int. J. Softw. Tools Technol. Transf. (2007): 213-254-