Colored Petri nets (CPNs)

dr. Tamás Bartha

BME Department of Measurement and Information Systems
Motivation

• Petri net model of Dining Philosophers
Motivation

- Why not this way?
Motivation

- Distinction of tokens: colored Petri net

val n = 5;
colset PH = index ph with 1..n;
colset CS = index cs with 1..n;
var p: PH;

fun Chopsticks(ph(i)) =
  1`cs(i) ++
  1`cs(if i=n then 1 else i+1);
Motivation

• Meaning of colored tokens
A more complex example (see later)
Colored Petri nets

- Colored Petri net (CPN)
  - Extension of uncolored Petri nets with:
    - Flexible data structures
    - Data manipulation language
  - Colored Petri nets unite:
    - Graphical representation $\rightarrow$ clarity
    - Well-defined semantics $\rightarrow$ formal analysis
  - CPN model = net structure + declarations + net markings, expressions + initialization
Main components of CPNs (overview)

• Extensions of tokens
  – Data value: colored token
  – Data type: color set

• Extensions of places
  – Type of place: data type of accepted tokens
  – Initial marking inscription: initial tokens
  – Current marking: multiset of tokens matching the place’s type

• Extensions of arcs
  – Arc expression: tokens moved (with variables to be bound)

• Extensions of transitions
  – Guard for firing
  – To fire: arc expressions shall be bound to colored tokens
Comparison of colored and uncolored Petri nets

Uncolored (P-T) Petri nets:
- Uncolored tokens
- Set of tokens (cardinality)
- Token manipulation
- Initial marking
- Inhibitor edges
- Edge weights
- Transition can be enabled
- Conflict between different enabled transitions
- ~ assembly

Colored Petri nets:
- Colored tokens
- Multiset of tokens
- Data manipulation
- Initial marking inscription
- Guards
- Arc expressions (+variables)
- Binding can be enabled
- Conflict between different bindings of the same transition
- ~ high-level programming lang.
Structure of colored Petri nets
Extensions of tokens

- Colored token
  - Represents a data value

- Color set:
  - Defines the data type
    - E.g., enumeration (with), base type (int, bool, string, ...)
  - Can be complex (compound)
    - E.g., color P = product U * I

- Declaration: in formal language
  - Standard ML

```
color U = with p | q;
color I = int;
color P = product U * I;
color E = with e;
var x : U;
var i : I;
```
Extensions of PN places

- **Color set inscription**: type (color) of the place
  - Type of tokens accepted by the place (one of the declared types)
  - Visualization: written next to the place, in italic

- **Initial marking inscription**
  - Defines the initial marking
  - A multiset of the accepted color set (may be more than one token per color)
  - Visualization: written next to the place, underlined

- **Current marking**
  - Description of current tokens
  - Visualization: written next to the place, number of tokens in circle and detailed description
Extensions of PN transitions

- **Arc expression**
  - Precondition of enablement (removed tokens) and the result of firing (placed tokens)
  - Type: type of the place connected to the arc (one transition have arcs with different types)
  - Visualization: next to the arc

- **Variable can be used in the expression**
  - Can be bound to data values (colored tokens)
  - Shall have a type (the color set of tokens that can be bound to it)

- **Guard**
  - Boolean expression, needs to be true to enable the transition
  - Visualization: next to the transition, within [ ]
Structure of colored Petri nets: Summary

- **Net structure:**
  - Represents the control and data flow structure of the system
  - Places, transitions, arcs

- **Declarations:**
  - Define the data structures and used functions
  - Color sets, variables, arc expressions

- **Markings, naming:**
  - Define the syntactic and data manipulation items
  - Names, color sets, in/out arc expressions, guards, current state

- **Initializing expression:**
  - Defines the initial state of the model (constants)
color U = with p | q;
color I = int;
color P = product U * I;
color E = with e;
var x : U;
var i : I;

- **Elements of CPNs:**
  - **Places**
    - Name
    - Color set
    - Initial marking
    - Current marking
  - **Transitions**
    - Name
    - Guard
  - **Arcs**
    - Arc expressions (incoming, outgoing)
Example: Control structures 1

IF $b$ THEN stat1 ELSE stat2

WHILE $b$ DO stat
Example: Control structures 2

REPEAT stat UNTIL b

Subroutine call

Start of a process

[−b] [b]
Toolset of colored Petri nets
CPN: Definition of color sets

- Simple color sets
  - Uncolored tokens: `unit`
  - Base types: `int, bool, real, string`
  - Subset: `with 1..4;`
  - Enumeration: `with true | false;`
  - Indexing (vector): `index d with 1..4;`

- Can be used in the definitions of the following:
  - Compound color sets
  - Variables, constants
  - Functions, operators
Compound color sets

- Ways to create compound color sets:
  - Union:
    ```
    union S + T;
    ```
  - Cross product (construction of tuples):
    ```
    product P * Q * R;
    ```
  - Record (labelled tuples):
    ```
    record p:P * q:Q * r:R;
    ```
  - List:
    ```
    list int with 2..6;
    ```
Additional CPN elements: Variables

- **Variables**
  - Symbolic names of tokens
    - Variable declaration:
      ```
      var proc : P;
      ```

- **Constants**
  - With fixed values
    - Constant declaration:
      ```
      val n = 10;
      val d1 = d(1):D;
      ```

- **In the following expr.'s:**
  - Arc expressions
  - Guards

- **In the following decl.'s:**
  - Color sets
  - Functions, operators
  - Arc expressions, guards, initialization expressions
Additional CPN elements: Functions

• Functions
  Side effect-free functions in SML language
  
  - Example:
    ```
    fun Chopsticks(ph(i)) = 
      1`cs(i) ++ 
      1`cs(if i=n then 1 else i+1);
    ```

• Operations, operators
  Infix notation

• In the following decl.'s:
  - Color sets
  - Functions, operators, constants
  - Arc expressions, guards, initialization expressions
Additional CPN elements: Expressions

• Net expressions
  – Value: evaluated with a specific binding of the variables
  – Type: set of all possible evaluations
  – Examples:
    \[
    \begin{align*}
    \text{x=q} \\
    2\cdot(x,i) \\
    \text{if } x=q \text{ then } 2\cdot i \text{ else empty} \\
    \text{Mes}(s)
    \end{align*}
    \]

• Usage in:
  – Arc expressions, guards, initialization expressions
Expressions: Operations with multisets

Addition: $a_1 + a_2$

Comparison: $a_1 \leq a_2$, $a_1 \neq a_2$

Size: $|a_1|$

Scalar multiplication: $n \cdot a_1$

Subtraction: $a_1 - a_2$ (only if $a_2 \leq a_1$)
Behavior of colored Petri nets (informal semantic)
Marking and binding

- **Marking:**
  - Distribution of tokens (count, by color) on the places

- **Binding the arc expressions of a transition:**
  - The variables are bound to data values (colored tokens)
  - For a given transition each occurrence of a variable will be bound to the same value
  - Unbound variable on outgoing arc: Can be bound to any value of its type
  - The bindings of different transitions are independent
Enabling of transitions

- **Transition enabled with a given marking and binding:**
  - Each input arc’s expression evaluates to a multiset of tokens that is present on the corresponding input place
  - The guard is true
  - If a transition is enabled with a binding, it can fire

- **Binding item for firing:**
  - A pair (transition, binding), e.g., (T1, <x=p>)
  - Can be enabled with a marking → can fire
  - In case of one transition: many bindings, many enabled binding items may be constructed; they can fire
Firing

• Transition fires with a binding (i.e., a binding item fires):
  – Removes tokens from the input places according to the arc expressions and the firing binding
  – Adds tokens from the output places according to the arc expressions and the firing binding

• Step (effect of firing on the state space):
  – The marking of the CPN changes
Reachability graph

- **Node:**
  - A marking: count and color of tokens for each place
  - May have an ID, predecessor node and successor node

- **Edge:**
  - The firing binding item: the transition and the binding
  - By definition only one firing binding item is shown in the reachability graph
CPN Tools demo

- Model of dining philosophers
- Simulation
- Reachability graph
Formal definition and semantics of colored Petri nets
Multisets

- **Multiset**: may contain several of the same element
  - **Mapping**: $Bag(A)$, to the domain of $A$, $a \in [A \rightarrow \mathbb{N}]$
  - **Formally**: $a = \sum_{x \in A} a(x) \cdot x$, alternative notation: $a = \sum_{x \in A} a(x)'x$

- **Operations on multisets**:
  - **Comparison**: $a_2 \neq a_1$ if $\exists x \in A, a_2(x) \neq a_1(x)$
    $a_2 \leq a_1$ if $\forall x \in A, a_2(x) \leq a_1(x)$
  - **Size**: $|a| = \sum_{x \in A} a(x)$
  - **Addition**: $a_1 + a_2 = \sum_{x \in A} (a_1(x) + a_2(x)) \cdot x$
  - **Subtraction**: $a_1 - a_2 = \sum_{x \in A} (a_1(x) - a_2(x)) \cdot x$ if $a_2 \leq a_1$
  - **Scalar multiplication**: $n \cdot a = \sum_{x \in A} (n \cdot a(x)) \cdot x$
Operations with multisets

Addition: $a_1 + a_2$

Comparison: $a_1 \leq a_2, a_1 \neq a_2$

Size: $|a_1|$

Scalar multiplication: $n \cdot a_1$

Subtraction: $a_1 - a_2$ (only if $a_2 \leq a_1$)
Multisets (continued)

- **Union of multisets:** \(a_1 \cup a_2 \cup \ldots \cup a_m\)
  - Domain: \(A_1 \cup A_2 \cup \ldots \cup A_m\)
  - Item: \(e_i \in \bigcup_{k=1}^{m} A_k\) if \(\exists A_j, e_i \in A_j\)

- **Construction of tuples:** \(\langle A_1, A_2, \ldots, A_n \rangle\)
  - Domain: \(A_1 \times A_2 \times \ldots \times A_2\)
  - Item: \(\langle e_1, e_2, \ldots, e_n \rangle \in \Diamond_{1}^{n} A_j\) if \(\forall e_i \in A_i\)
  - Generalization: \(\langle a_1, a_2, \ldots, a_n \rangle\)
Formal definition of CPNs

\[ \text{CPN} = (\Sigma, P, T, A, C, G, E, M_0) \]

**Color sets:** \( \Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_k\} \)

**Places:** \( P = \{p_1, p_2, \ldots, p_\pi\} \)

**Transitions:** \( T = \{t_1, t_2, \ldots, t_\tau\} \)

\[ P \cap T = \emptyset \]

**Arcs:** \( A \subseteq (P \times T) \cup (T \times P) \)

**Color set func.:** \( C : P \mapsto \Sigma \)

**Guards:** \( G : \forall t \in T, \left[ \text{Type}(G(t)) = B \land \text{Type}(\text{Var}(G(t))) \subseteq \Sigma \right] \)

**Arc expressions:** \( E : \forall a \in A, \left[ \text{Type}(E(a)) = C(p)_{\text{MS}} \land \text{Type}(\text{Var}(E(a))) \subseteq \Sigma \right] \)

**Initial marking:** \( M_0 : \forall p \in P, \left[ \text{Type}(M_0(p)) = C(p)_{\text{MS}} \right] \)
Notations used in the formal definition

- The type (color set) of variable \( v \): \( \text{Type}(v) \)
- The type of expression \( expr \): \( \text{Type}(expr) \)
- The set of variables in expression \( expr \): \( \text{Var}(expr) \)
- A binding of variable \( v \): \( b(v) \in \text{Type}(v) \)
- Evaluation (value) of expression \( expr \) in binding \( b \): \( expr \langle b \rangle \) where \( v \in \text{Var}(expr) \) and \( b(v) \in \text{Type}(v) \)
Arc expressions

• May use variables
  – Variables have types (color sets): $\text{Type}(v)$
  – Their value is an element of their types’ multiset

• Closed arc expression: does not contain variables

• Open arc expression: contains variables that have to be bound to values
  – Binding: a specific value assignment to each variable
    • Arc expression can be evaluated with the given binding
  – Has type: $\text{Type}(expr) = C(p)_{MS}$
    • The color set (type) to which it is evaluated
  – Set of variables in the expression: $\text{Var}(expr)$
Bound and unbound variables

**Bound variables**
- Value binding is determined by the incoming arcs
- Consistency: a variable has only one value in each binding
  - For all in-arcs of the transition the same variable name denotes the same value

**Unbound variables**
- They can only be present in outgoing arc expressions
- Enablement did not assign (bound) any value to them
- Have to be bound at firing:
  - Can take any value from its color set
  - Number of possible bindings = cardinality of the color set
  - Non-deterministic choice
Guards

- Each guard is assigned to a transition
  - Expression over multisets
  - Evaluated to Boolean value
- The transition is enabled only if the guard is evaluated to “true”
  - “Filters” the enabled bindings
Enabling in colored Petri nets

- **Binding of transitions**
  - Valid binding: \( \forall v \in \text{Var}(t): b(v) \in \text{Type}(v) \land G(t)\langle b \rangle \)
    \[
    \text{Var}(t) = \{ v \mid v \in \text{Var}(G(t)) \lor \exists a \in A(t) : v \in \text{Var}(E(a)) \}
    \]
  - Set of all valid bindings: \( B(t) \)

- **A valid binding is enabled if**
  - Guard is true
  - The input places contain enough colored tokens (cf. arc expressions \( E^{-}(p,t)\langle b \rangle \)) and the inhibitor arcs do not inhibit the firing (cf. arc expressions \( E^{h}(p,t)\langle b \rangle \)):

\[
\forall p \in \bullet t : E^{-}(p,t)\langle b \rangle \leq M(p) \land E^{h}(p,t)\langle b \rangle > M(p)
\]
Firing in colored Petri nets

• An enabled transition can fire if there is no enabled transition with higher priority, i.e.
  – The transitions with higher priority do not have enough tokens in their input places (see arc expressions $E^{-}(p,t')(b')$) or their inhibitor arcs disable the firing (see arc expressions $E^{h}(p,t')(b')$),
    \[ \forall t', \pi(t') > \pi(t) : \exists p \in \bullet t' : \]
    \[ E^{-}(p,t')(b') > M(p) \lor E^{h}(p,t')(b') \leq M(p) \]
  – Or their guards are not satisfied (not evaluated to true)
    \[ \neg G(t')(b') \]
Firing in colored Petri nets

• Steps of firing:
  – Finding enabled bindings
    • Determined by incoming arc expressions and guards
  – Transition enabled with a given binding \( \rightarrow \) it can fire
  – Firing: removal of colored tokens from incoming places, adding colored tokens to outgoing places

\[
\forall p \in P : M'(p) = M(p) - \sum_{p \in \bullet t} E^-(p, t)\langle b \rangle + \sum_{p \in t \bullet} E^+(t, p)\langle b \rangle
\]

– Then \( M' \) directly reachable from \( M : M \ [ (t, b) \rangle \ M' \)
Dynamic properties of colored Petri nets
Reachability graph (excerpt)

Sent, Received, Acknowledged
Dynamic properties of CPNs

• Extension of the uncolored Petri net properties to multisets

• Boundedness

  A place is bounded if the number of tokens in any state is bounded
  – $n$ is an upper integer bound for $p$ if $\forall M \in [M_0]: |M(p)| < n$
  – $m$ is an upper multiset bound for $p$ if $\forall M \in [M_0]: M(p) < m$

• Reversibility (home state)

  It is always possible to get back to a home state
  – $M$ is a home state if $\forall M' \in [M_0]: M \in [M']$
  – $X$ is a home group if $\forall M' \in [M_0]: X \cap [M'] \neq \emptyset$
Dynamic properties of CPNs

- **Liveness**
  
  Liveness guarantees that some of the binding items remain active
  
  - **Dead state (deadlock):** no binding item is enabled
    \[ \forall b \in BE : \neg M[b] \]
  
  - **Dead transition:** none of its bindings may become enabled
    \[ \forall M' \in [M], b \in B(t) : \neg M'[b] \]
  
  - **Live transition:** from each reachable state there is at least one trajectory starting where the transition is not dead (at least one binding will become active)
    \[ \forall M' \in [M_0], \exists M'' \in [M'], \exists b \in B(t) : M''[b] \]
Dynamic properties of CPNs

• Fairness

  Fairness represents how often can a binding item fire

  – Impartial transition: fires infinitely often
    \[ \forall b \in B(t), \quad |\sigma| = \infty : \quad OC_b(\sigma) = \infty \]

  – Fair transition: infinitely many enabling \( \Rightarrow \) infinitely many firing
    \[ \forall b \in B(t), \quad |\sigma| = \infty : \quad EN_b(\sigma) = \infty \Rightarrow OC_b(\sigma) = \infty \]

  – Just transition: persistent enabling \( \Rightarrow \) firing
    (there is no persistent enabling without firing)
    \[ \forall b \in B(t), \quad \forall i \geq 1 : \]
    \[ \left[ \text{EN}_{b,i}(\sigma) \neq 0 \Rightarrow \exists k \geq i : \left[ \text{EN}_{b,k}(\sigma) = 0 \lor \text{OC}_{b,k}(\sigma) \neq 0 \right] \right] \]
Structural properties of colored Petri nets
T invariant in CPNs

- Transition invariant

A firing sequence $\sigma$ that does not affect the state:

$$M'(p) = M(p) - \sum_{p \in t, b \in \sigma} E^-(p, t)\langle b \rangle + \sum_{p \in t, b \in \sigma} E^+(t, p)\langle b \rangle$$

where $M'(p) - M(p) = 0$ for all $p$

then $\sum_{p \in t, b \in \sigma} E^-(p, t)\langle b \rangle = \sum_{p \in t, b \in \sigma} E^+(t, p)\langle b \rangle$
P invariant in CPNs

- **Place invariant**

  Idea: Equation that is satisfied in every reachable state

  - Weighted token sum is constant:
    \[ W_{p_1}(M(p_1)) + W_{p_2}(M(p_2)) + \ldots + W_{p_n}(M(p_n)) = m_{inv} \]

  - Weight function: maps the color sets of the places to a common multiset

  - \( W_p \) is a P invariant:
    \[ \forall M \in [M_0]: \sum_{p \in P} W_p(M(p)) = \sum_{p \in P} W_p(M_0(p)) \]
Unfolding colored Petri nets
Possibilities to construct a CPN

• **CPNs: information in both structure and data**

• **Extremities**
  – Pure structural information, no data:
    • Uncolored (P/T) net (can be build as a CPN)
  – No structure, only data (data and control information):
    • 1 place + 1 transition, complex color sets and arc expressions

• **We need the golden mean**
  – To have a clean, readable CPN
Example: Modeling possibilities

Control flow expressed by the structure

The same in code ("folded")
Unfolding

- Expressivity of CPNs (with priorities) equals to the expressivity of uncoloured PNs with inhibitor edges (and with priorities)
  - Each CPN has a corresponding uncolored PN with equivalent behavior (in the automaton theoretical sense $\rightarrow$ bisimulation for the steps)
  - Equivalent uncolored net: unfolded net
  - Unfolding:
    - Information of colored tokens is represented by the structure
    - Each event of the CPN has exactly one corresponding event in the unfolded net
Simple colored net

color A = with apple | pear;
color B = with red | yellow;
color C = with fresh | stale;
color BC = product B*C declare mult;
var x: A;
var y: B;
var z: C;
Unfolded, uncolored net

apple \quad p_1 \quad pear

red \quad p_2 \quad yellow

(red, fresh) \quad (yellow, fresh) \quad (red, stale) \quad (yellow, stale)
Example: A simple commit protocol

Problem description:

• The system consists of three components: $c_1$, $c_2$, és $c_3$
• One of them randomly becomes the coordinator which sends a request to the other two
• The response of another component is either an abort or commit vote
• Based on the vote of the two components the coordinator decides: the decision is commit if the two other components voted for commit, abort otherwise.
Example: Model of the simple commit protocol

- Three color sets are defined in the CPN model. Two of them are simple color sets:
  \[ C = \{0, c_1, c_2, c_3\} \] representing components,
  \[ D = \{\text{commit, abort}\} \] representing votes/decisions.
- One compound color set:
  \[ M = C \times C \] for requests (originator and target);
  the \((0, x)\)-like token represents that the coordinator does not receive a request.
- Five variables are used, their types: \( x, y, z \in C \);
  and \( d_1, d_2 \in D \)
- The if in the arc expression has the common intuitive meaning (as in programming languages).
- In the initial state the place \( p_1 \) has 3 tokens:
  \[ M(p_1) = c_1 + c_2 + c_3, \] the other places are empty.
- Empty set is denoted by \( \emptyset \)
Example: Model of the simple commit protocol

- **Colored Petri net model:**
  - $p_1$: Participants (tokens $c_1$, $c_2$, $c_3$ in initial state)
  - $p_2$: Requests
  - $p_3$: Votes
  - $p_4$: Decision
Example: Model of the simple commit protocol

- Partially unfolded (uncolored PN) model: $c_1$ is the coordinator
- Simple optimizations were done in the structure and events (firings)
Example: Model of the simple commit protocol

Similar nets needed for these parts too
Hierarchical colored Petri nets
Hierarchical colored Petri nets

- Integration of subnets into a complex CPN hierarchically
  - Pages: Colored Petri net models (subnets)
    - Page number, page name: alternatives to refer to the subnet
    - The pages can be instantiated (on any level of the hierarchy)
    - The marking (token distribution) is unique for each instance
  - Hierarchy: Structure of the pages
    - Main (prime) page: topmost level
    - Secondary page instances (subpages)
      - Identification: page-instance ID number
      - Page-hierarchy graph
Tools of hierarchical composition

1. **Coarse (substitute) transition**
   - Representation of a subpage
   - Interfaces between pages: places
     1. On main page: “Socket” places → insertion point of subnets
     2. On subpage: “Port” places → connection points of the subnet,
        port type: input, output, input-output (bidirectional), general

2. **Fusion places**
   - Places with same name, multiple instances,
     denoting the same place at different locations
   - Tokens are added / removed simultaneously
     to / from each instance
Example: hierarchical version of the simple protocol

```
color INT = int;
color DATA = string;
color INTxDATA = product INT*DATA;
color INTxINT = product INT * INT;
var n, k, n1, n2: INT;
var p, str: DATA;
val stop = "##########";

color Ten0 = int with 0..10;
color Ten1 = int with 1..10;
var s: Ten0; var r, r1, r2: Ten1;
fun Ok(s:Ten0, r:Ten1) = (r<s);
```
Example CPN:
Distributed database manager
Specification of the distributed database manager

- n different servers; local copy on each server, managed by a local database manager
  \[ \text{DBM} = \{d_1, d_2, \ldots, d_n\}, \ n \geq 3 \]

- Database operations:
  - Modification of local data
  - Change notification of the other database managers which will update

- State of the system:
  - Active: handling the update is in progress
  - Passive: handling the update is finished

- States of database managers:
  - Inactive, Performing (updating), Waiting (for acknowledgement)

- Notification about changes: with messages
  - Message header: sender and receiver database manager
    \[ \text{MES} = \{(s,r) \mid s,r \in \text{DBM} \land s \neq r\}, \quad \text{Mes}(s) = \sum_{r \in \text{DBM-\{s\}}} 1 \ `(s,r) \]
  - Message states: Unused, Sent, Received, Acknowledged
Distributed database: Declarations

**Declaration field**

val \( n = 4; \)
color \( DBM \) = index \( d \) with \( 1..n; \)
color \( PR \) = product \( DBM \times DBM; \)
fun \( \text{diff}(x,y) = (x<>y); \)
color \( MES \) = subset \( PR \) by \( \text{diff}; \)
color \( E \) = with \( e; \)
fun \( \text{Mes}(s) = \text{mult}^\prime PR(1`s, DBM--1`s) \)
var \( s, r : DBM; \)

**Meaning:**

\[
\text{DBM} = \{d_1, d_2, \ldots, d_n\}
\]

\[
\text{MES} = \{(s, r) \mid s, r \in \text{DBM} \land s \neq r\}
\]

\[
\text{Mes}(s) = \sum_{r \in \text{DBM}-\{s\}} 1'(s, r)
\]

- **DBM:** database managers
- **PR:** DBM pairs
- **MES:** possible messages (headers)
- **Mes(s):** messages that can be sent by the DBM \( s \)
- **E:** simple token (uncolored)
Distributed database: System component

- System states denoted by a single token, initially ‘Passive’
Distributed database: Database managers

- DBMs are grouped by states, each group is represented by one place.
- Initially each DBM is inactive; later it can change or update.
Distributed database: Messages

- Places: message buffers
- A DBM sends notifications to the others; one from the set of possible messages
Distributed database: Complete CPN model

- Active and Passive places: only one DBM performs change at the same time, then waits
Particularities of the model

• **Causality**
  - Update and Send → Receive → Send Ack → Receive Ack

• **Conflict**
  - Update and Send enabled for each binding item $s$, but only one can fire

• **Concurrency**
  - Receive a Message for binding items $(s,r)$ that are concurrent with themselves
Reachability graph for $n=3$

- **Occurrence graph**
- **Abbreviated transition names:**
  - **SM**: Update and Send Messages
  - **RM**: Receive a Message
  - **SA**: Send an Acknowledgment
  - **RA**: Receive all Acknowledgments
Dynamic properties: boundedness

<table>
<thead>
<tr>
<th>State</th>
<th>Multiset</th>
<th>Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inactive</td>
<td>DBM</td>
<td>n</td>
</tr>
<tr>
<td>Waiting</td>
<td>DBM</td>
<td>1</td>
</tr>
<tr>
<td>Performing</td>
<td>DBM</td>
<td>n - 1</td>
</tr>
<tr>
<td>Unused</td>
<td>MES</td>
<td>n*(n - 1)</td>
</tr>
<tr>
<td>Sent, Received, Acknowledged</td>
<td>MES</td>
<td>n - 1</td>
</tr>
<tr>
<td>Passive, Active</td>
<td>E</td>
<td>1</td>
</tr>
</tbody>
</table>
Dynamic properties: Liveness, fairness

• **Liveness Properties**
  – Dead markings: None
  – Dead transition instances: None
  – Live transition instances: All

• **Fairness Properties**
  – Impartial transition instances:
    • Update and Send Messages
    • Receive a Message
    • Send an Acknowledgment
    • Receive all Acknowledgments
  – Fair transition instances: None
  – Just transition instances: None

• Impartial transition: Fires infinitely often
• Fair transition: Infinitely many enabling → infinitely many firing
• Just transition: Persistent enabling → firing
Structural properties: P invariants

- $M(\text{Active}) + M(\text{Passive}) = 1' e$
- $M(\text{Inactive}) + M(\text{Waiting}) + M(\text{Performing}) = DBM$
- $M(\text{Unused}) + M(\text{Sent}) + M(\text{Received}) + M(\text{Acknowledged}) = MES$
- $M(\text{Performing}) - \text{Rec}(M(\text{Received})) = \emptyset$
  - Function $\text{Rec()}$ for token mapping: $\text{Rec}(s,r) = r$
- $M(\text{Sent}) + M(\text{Received}) + M(\text{Acknowledged}) - \text{Mes}(M(\text{Waiting})) = \emptyset$
  - Function $\text{Mes()}$ for token mapping: $\text{Mes}(s)$: the messages can be sent by DBM $s$
- $M(\text{Active}) - \text{Ign}(M(\text{Waiting})) = \emptyset$
  - Function $\text{Ign()}$ turns tokens with any color into token with color $e \in E$
P invariant: the state of the system

\[ M(\text{Active}) + M(\text{Passive}) = 1`e \]
**P invariant: database managers**

\[ M(\text{Inactive}) + M(\text{Waiting}) + M(\text{Performing}) = DBM \]
P invariants: messaging subsystem

\[ M(\text{Unused}) + M(\text{Sent}) + M(\text{Received}) + M(\text{Acknowledged}) = MES \]
P invariants of the model
One of the P invariants

\[ M(\text{Sent}) + M(\text{Received}) + M(\text{Acknowledged}) - \text{Mes}(M(\text{Waiting})) = \emptyset \]
The complete CPN model (reminder)

- **Update and Send Messages**
- **Receive a Message**
- **Receive all Acknowledgments**
- **Send an Acknowledgment**
- **Acknowledged**

States:
- Active
- Passive
- Unused
- Inactive
- Waiting
- Performing

Transitions:
- **Sent**
- **Received**
- **DBM**

Symbols:
- s
- e
- r
- (s,r)

The complete CPN model (reminder)
Messaging unfolded for n=3