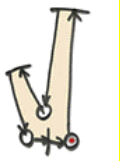
Workshop on the Definition, Implementation and Application of a Standard Interchange Format for Petri Nets

Bologna, Italy, 21-25 June, 2004



Operational PNML: Towards a PNML Support for Model Construction and Modification

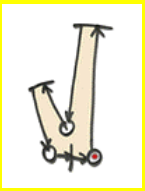
João Paulo Barros[1,2]
jpb@uninova.pt

Luís Gomes[1]
lugo@uninova.pt

[1]**Universidade Nova de Lisboa**
Faculdade de Ciências e Tecnologia
Dep. de Engenharia Electrotécnica
Campus da FCT-UNL
2825 Monte de Caparica

[2]**Instituto Politécnico de Beja**
Escola Superior de Tecnologia e Gestão
Área Departamental de Engenharia
Rua Afonso III, n.º 1
7800-050 Beja

PORTUGAL



The Objective

To allow a PNML specification for model composition and model modification

Why?

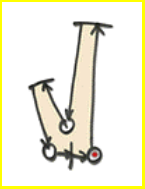
—To fight *model-size explosion*

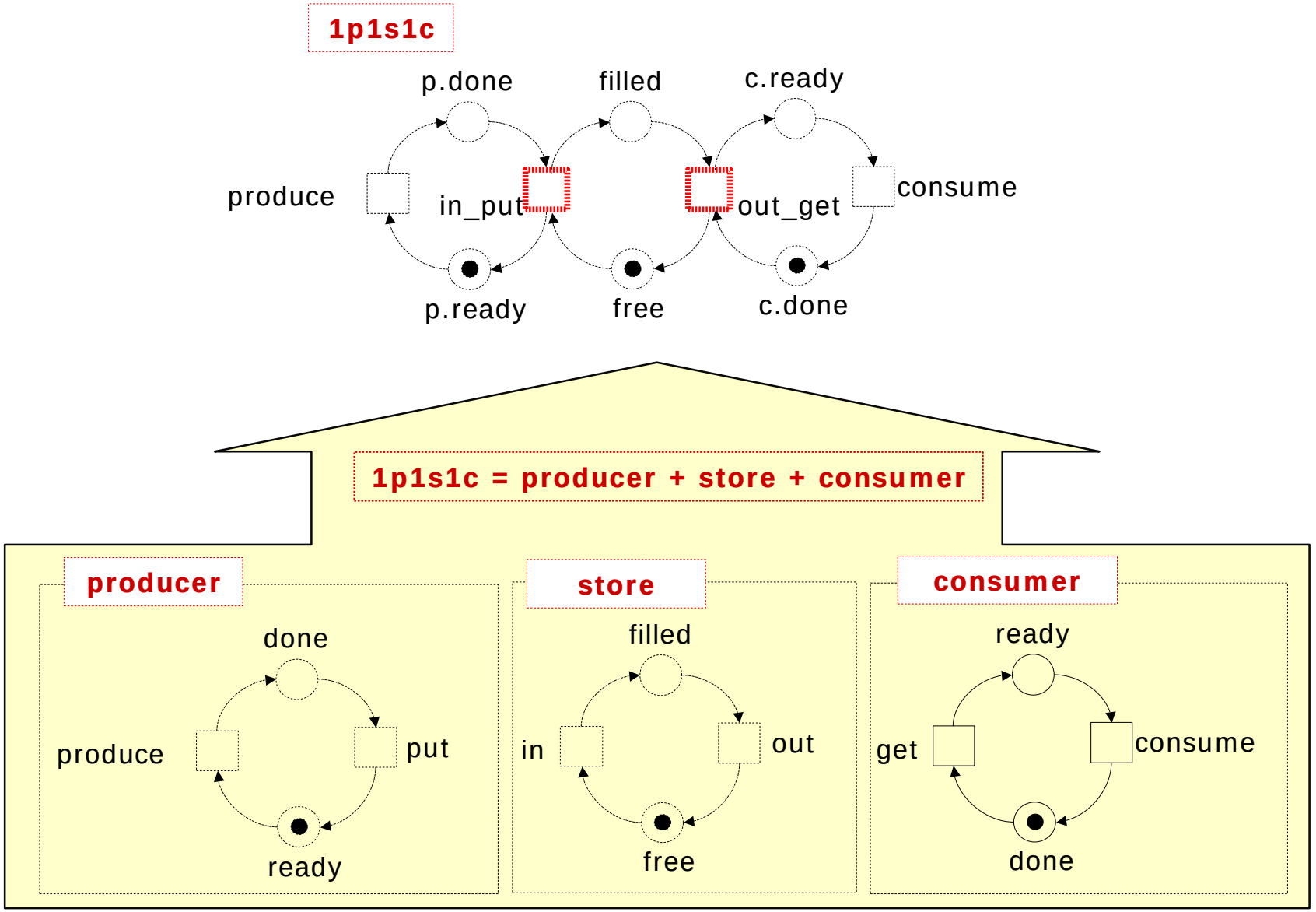
—To specify model structuring through PNML specifications

—To specify model modifications through PNML specifications

Workshop on the Definition, Implementation and Application of a Standard Interchange Format for Petri Nets

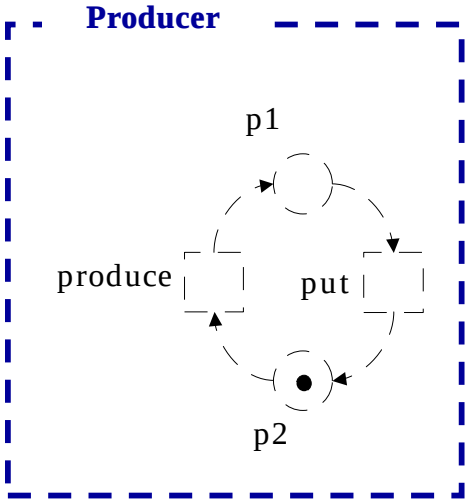
Bologna, Italy, 21-25 June, 2004



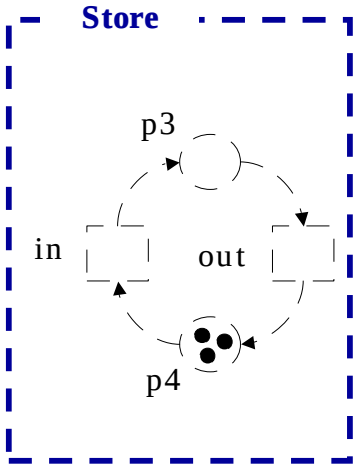


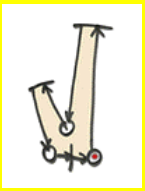
How?

```
<pnml xmlns="http://www.informatik.hu-berlin.de/top/pnml/ptNetb">
 <net id="Producer" type="http://www.informatik.hu-berlin.de/top/pntd/ptNetb">
  <place id="p1">
    <initialMarking>
      <text>0</text>
    </initialMarking>
  </place>

   ...

 </net>
</pnml>
```



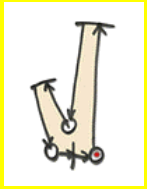

```
<pnml xmlns="http://www.informatik.hu-berlin.de/top/pnml/ptNetb">
 <net id="Store" type="http://www.informatik.hu-berlin.de/top/pntd/ptNetb">
  <name>
   <text>Store net</text>
  </name>
  <place id="p3">
   <name>
    <text>p3</text>
   </name>

    . . .

 </net>
</pnml>
```



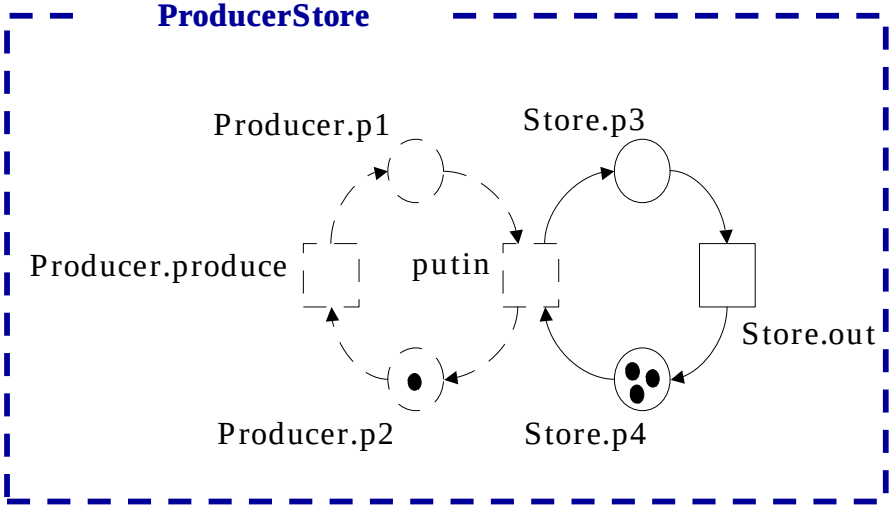


Workshop on the Definition, Implementation and Application of a Standard Interchange Format for Petri Nets

Bologna, Italy, 21-25 June, 2004

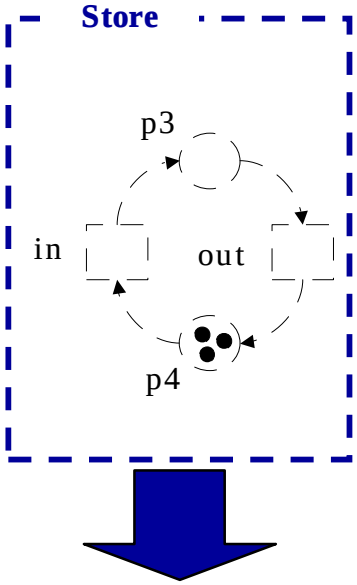


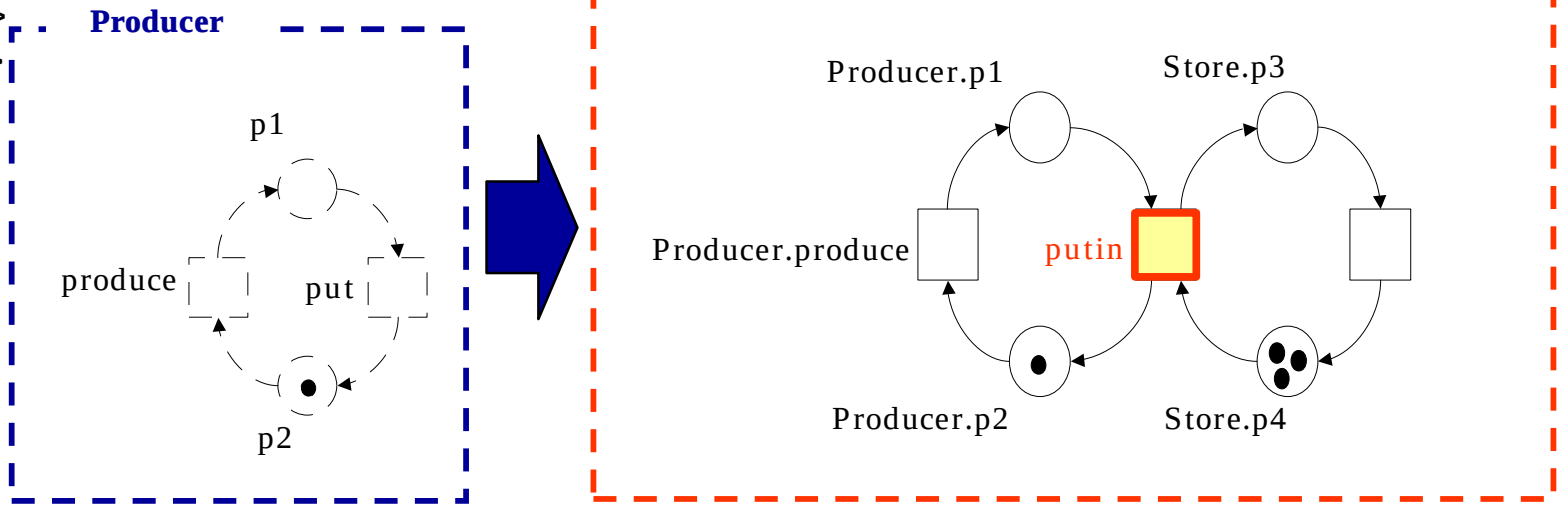
```
<pnml xmlns='http://www.informatik.hu-berlin.de/top/pnml/ptNetb'>
 <net type='http://www.informatik.hu-berlin.de/top/pntd/ptNetb' id='ProducerStore'>
  <place id='Producer.p1'>
   <initialMarking>
    <text>0</text>
   </initialMarking>
  </place>
  <place id='Producer.p2'>
  ...
  <place id='Store.p3'>
  ...
  <place id='Store.p4'>
  ...
  <transition id='Producer.produce'>
  ...
  </transition><transition id='Store.out'>
  ...
  <arc id='Producer.a1' source='Producer.produce' target='Producer.p1'>
  <inscription><text>1</text></inscription>
  </arc>
  <arc id='Producer.a2' source='Producer.p1' target='putin'>
  ...
  <arc id='Producer.a3' source='putin' target='Producer.p2'>
  ...
  <arc id='Producer.a4' source='Producer.p2' target='Producer.produce'>
  ...
  <arc id='Store.a1' source='putin' target='Store.p3'>
  ...
  <arc id='Store.a2' source='Store.p3' target='Store.out'>
  ...
  <arc id='Store.a3' source='Store.out' target='Store.p4'>
  ...
  <arc id='Store.a4' source='Store.p4' target='putin'>
  ...
  <transition id='putin'>
  ...
 </net>
</pnml>
```



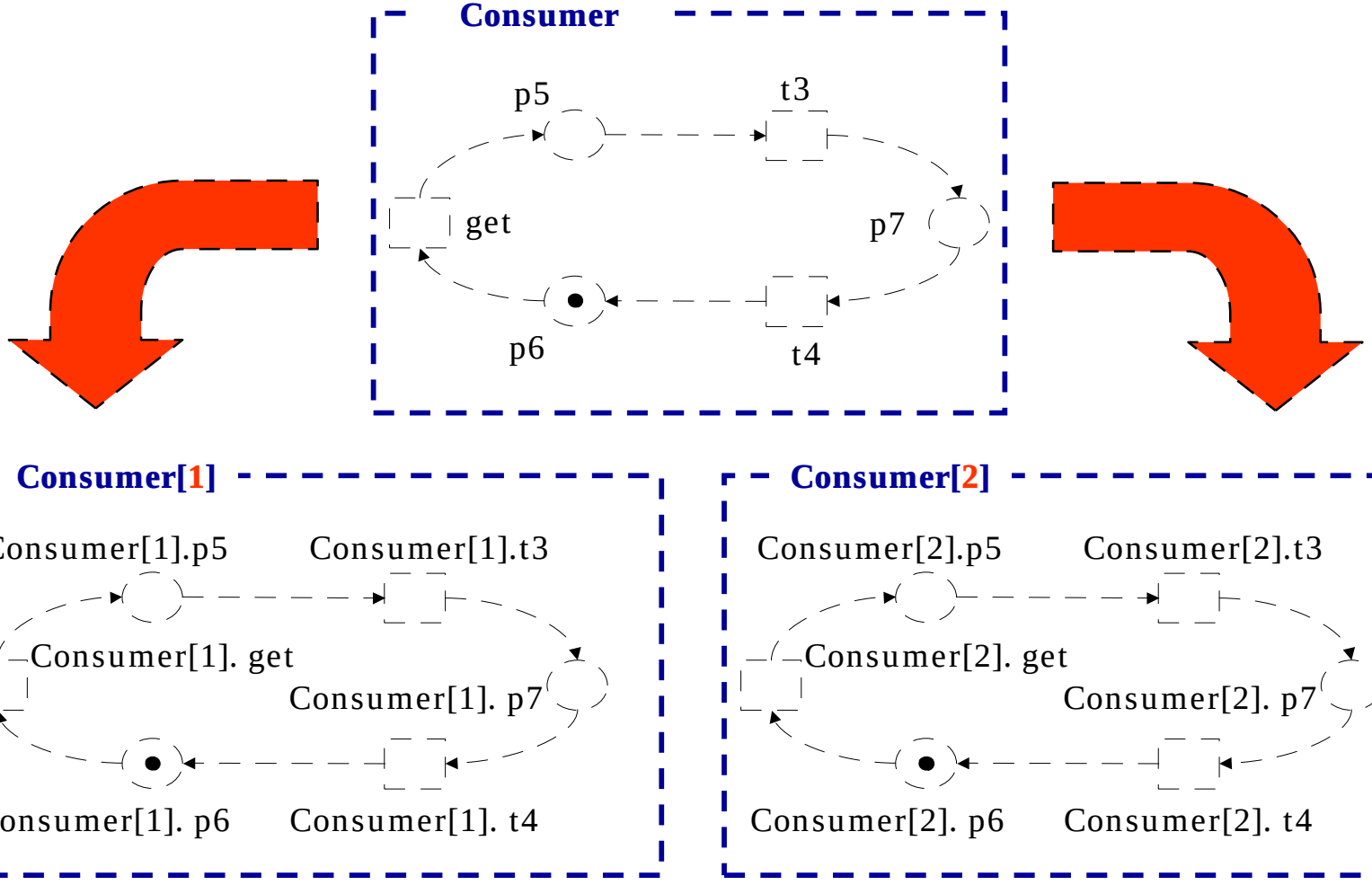
An alternative way to define a net

```
<pnml xmlns="http://www.informatik.hu-berlin.de/top/pnml/ptNetb">
  <net id="ProducerStore"
       type="http://www.informatik.hu-berlin.de/top/pntd/ptNetb">
    <operations>
      <addition result="return">
        <net netID="Producer"/>
        <net netID="Store"/>
        <merge>
          <transitionFusion nodeID="putin">
            <transition netID="Producer" nodeID="put"/>
            <transition netID="Store" nodeID="in"/>
          </transitionFusion>
        </merge>
      </addition>
    </operations>
  </net>
</pnml>
```





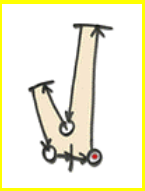
Net instances and Net vectors



Consumer[1..2]

Multiple operations

```
<pnml xmlns="http://www.informatik.hu-berlin.de/top/pnml/ptNetb">
  <net id="PSC" type="http://www.informatik.hu-berlin.de/top/pntd/ptNetb">
    <operations>
      <addition result="PS">
        <net netID="Producer"/>
        <netVector netID="Store" first="1" last="3"/>
        <merge>
          <transitionFusionVector nodeID="putin[i]"
                                  iterator="i" first="1" last="3">
            <transition netID="Producer" nodeID="put"/>
            <transition netID="Store[i]" nodeID="in"/>
          </transitionFusionVector>
        </merge>
      </addition>
      <addition result="return">
        <net netID="PS" />
        <netVector netID="Consumer" first="1" last="3"/>
        <merge>
          <transitionFusionVector nodeID="outget[i]"
                    iterator="i" first="1" last="3">
            <transition netID="PS" nodeID="Store[i].out"/>
            <transition netID="Consumer[i]" nodeID="get"/>
          </transitionFusionVector>
        </merge>
      </addition>
    </operations>
  </net>
</pnml>
```



Net Vectors and Fusion Vectors

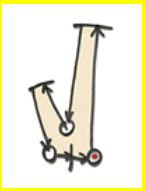
```
<pnml xmlns="http://www.informatik.hu-berlin.de/top/pnml/ptNetb">
  <net id="PSC" type="http://www.informatik.hu-berlin.de/top/pntd/ptNetb">
    <operations>
      <addition result="PS">
        <net netID="Producer"/>
        <netVector netID="Store" first="1" last="3"/>
        <merge>
          <transitionFusionVector nodeID="putin[i]"
                                  iterator="i" first="1" last="3">
            <transition netID="Producer" nodeID="put"/>
            <transition netID="Store[i]" nodeID="in"/>
          </transitionFusionVector>
        </merge>
      </addition>
      <addition result="return">
        <net netID="PS"/>
        <netVector netID="Consumer" first="1" last="3"/>
        <merge>
          <transitionFusionVector nodeID="outget[i]"
                    iterator="i" first="1" last="3">
            <transition netID="PS" nodeID="Store[i].out"/>
            <transition netID="Consumer[i]" nodeID="get"/>
          </transitionFusionVector>
        </merge>
      </addition>
    </operations>
  </net>
</pnml>
```

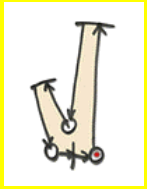
Producer
Store[1]
Store[2]
Store[3]

PS
Consumer[1]
Consumer[2]
Consumer[3]

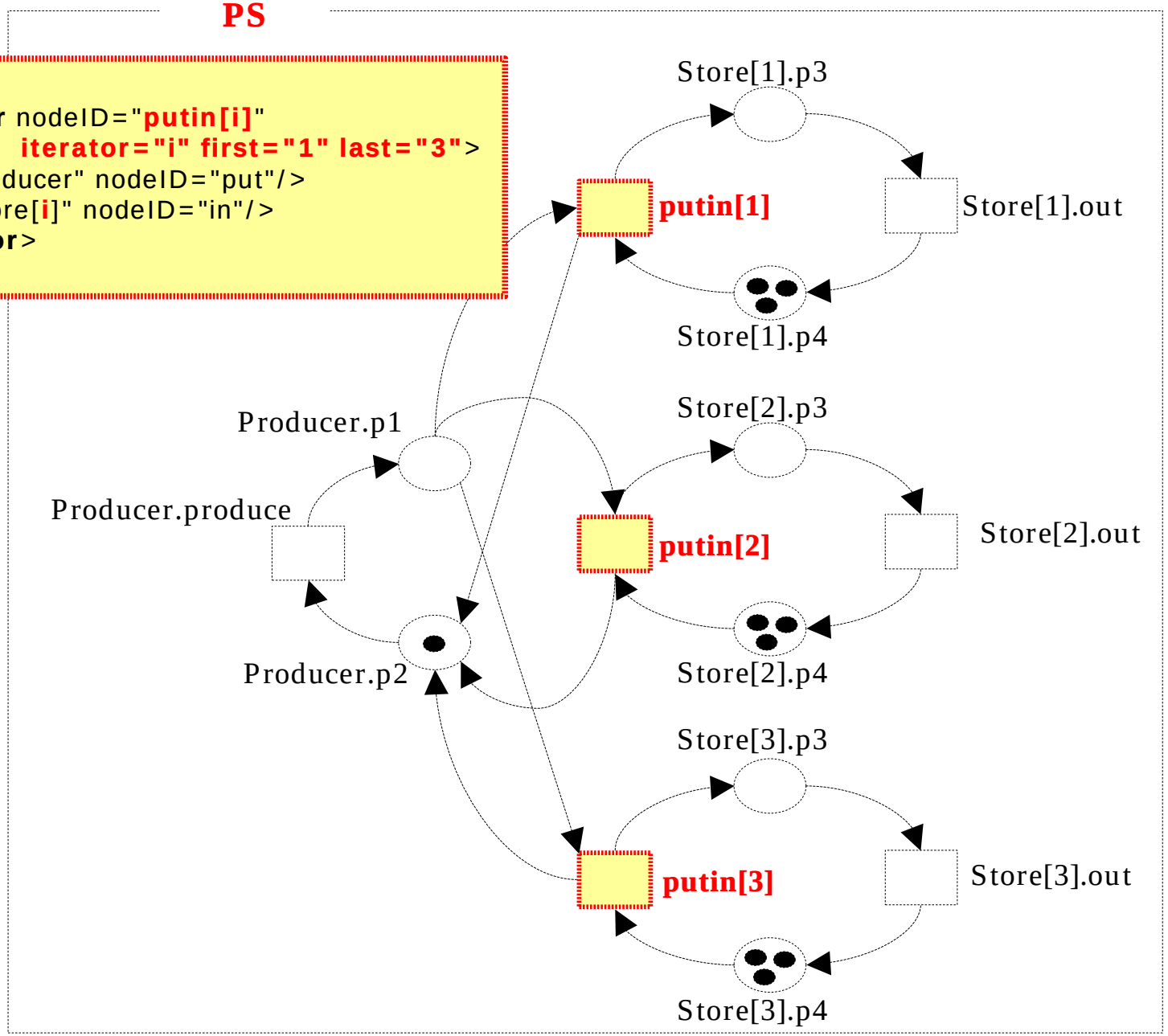
with i = 2:

```
<transitionFusion nodeID="outget[2]">
<transition netID="PS" nodeID="Store[2].out"/>
<transition netID="Consumer[2]" nodeID="get"/>
</transitionFusion>
```

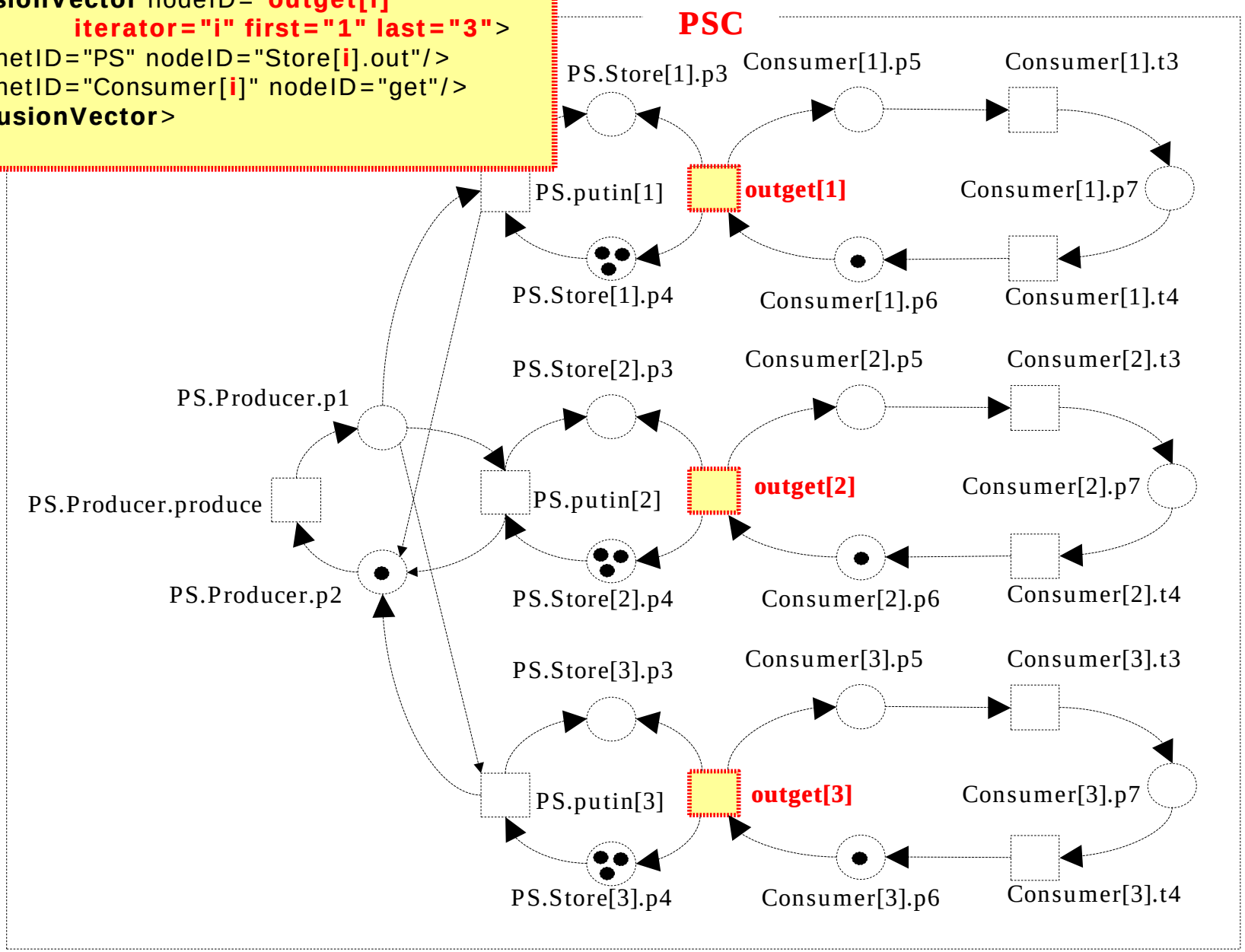


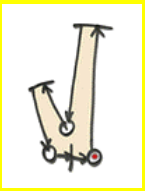


```
...
<transitionFusionVector nodeID="putin[i]"
                        iterator="i" first="1" last="3">
   <transition netID="Producer" nodeID="put"/>
   <transition netID="Store[i]" nodeID="in"/>
</transitionFusionVector>
...
```



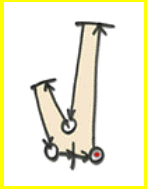
```
...
<transitionFusionVector nodeID="outget[i]"
                        iterator="i" first="1" last="3">
  <transition netID="PS" nodeID="Store[i].out"/>
  <transition netID="Consumer[i]" nodeID="get"/>
 </transitionFusionVector>
...
```





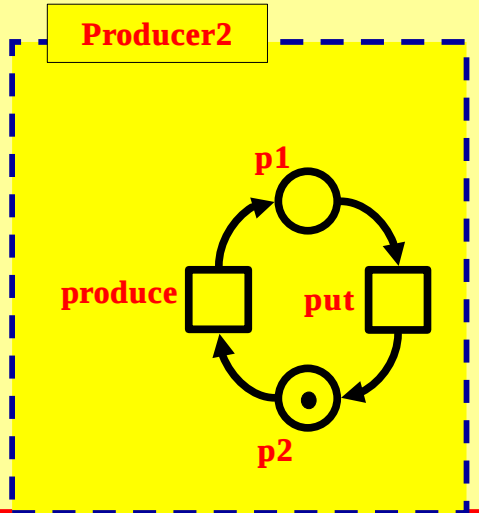
Net subtraction

```
<pnml xmlns="http://www.informatik.hu-berlin.de/top/pnml/ptNetb">
  <net id="Producer2"
  type="http://www.informatik.hu-berlin.de/top/pntd/ptNetb">
    <operations>
      <subtraction result="return">
        <net netID="ProducerStore"/>
        <net netID="Store"/>
        <merge>
          <transitionFusion nodeID="put">
            <transition netID="ProducerStore" nodeID="putin"/>
            <transition netID="Store" nodeID="in"/>
          </transitionFusion>
          <transitionFusion nodeID="produce">
            <transition netID="ProducerStore" nodeID="Producer.produce"/>
          </transitionFusion>
        </merge>
        <removal>
          <placeFusion nodeID="p3">
            <place netID="ProducerStore" nodeID="Store.p3"/>
            <place netID="Store" nodeID="p3"/>
          </placeFusion>
          <placeFusion nodeID="p4">
            <place netID="ProducerStore" nodeID="Store.p4"/>
            <place netID="Store" nodeID="p4"/>
          </placeFusion>
          <transitionFusion nodeID="out">
            <transition netID="ProducerStore" nodeID="Store.out"/>
            <transition netID="Store" nodeID="out"/>
          </transitionFusion>
        </removal>
      </subtraction>
    </operations>
  </net>
</pnml>
```



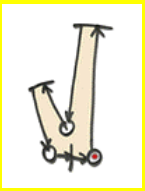
Net subtraction result

```
<pnml xmlns="http://www.informatik.hu-berlin.de/top/pnml/ptNetb">
  <net type="http://www.informatik.hu-berlin.de/top/pntd/ptNetb" id="Producer2">
    <place id="ProducerStore.Producer.p1">
      <initialMarking>
        <text>0</text>
      </initialMarking>
    </place>
    <place id="ProducerStore.Producer.p2">
      <initialMarking>
        <text>1</text>
      </initialMarking>
    </place>
    <arc id="ProducerStore.Producer.a1" source="produce" target="ProducerStore.Producer.p1">
      <inscription>
        <text>1</text>
      </inscription>
    </arc>
    <arc id="ProducerStore.Producer.a2" source="ProducerStore.Producer.p1" target="put">
      <inscription>
        <text>1</text>
      </inscription>
    </arc>
    <arc id="ProducerStore.Producer.a3" source="put" target="ProducerStore.Producer.p2">
      <inscription>
        <text>1</text>
      </inscription>
    </arc>
    <arc id="ProducerStore.Producer.a4" source="ProducerStore.Producer.p2" target="produce">
      <inscription>
        <text>1</text>
      </inscription>
    </arc>
    <transition id="put">
      <name>
        <text>in/put/in</text>
      </name>
    </transition>
    <transition id="produce">
      <name>
        <text>produce</text>
      </name>
    </transition>
  </net>
</pnml>
```



Workshop on the Definition, Implementation and Application of a Standard Interchange Format for Petri Nets

Bologna, Italy, 21-25 June, 2004



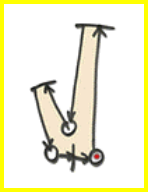
Dealing with annotations

1. Multiple and unknown node and arc's labels

e.g. High-level nets

2. Arcs

- "Fusion"
- Creation
- Removal
- Label modification

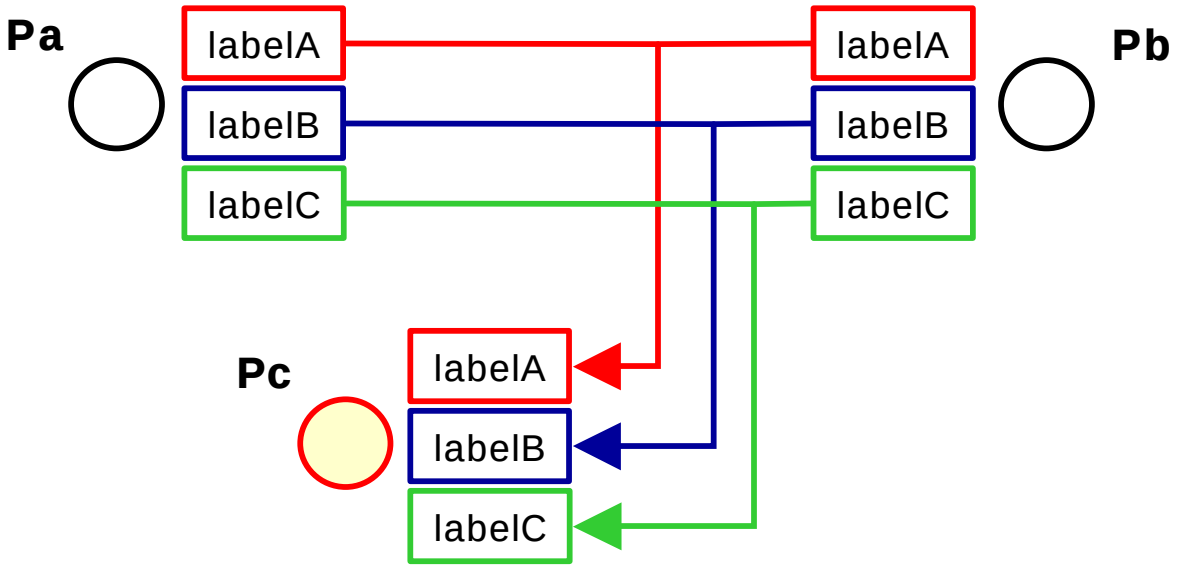


Fusion and Multiple Labels 1 / 3

When two or more nodes are fused
we must define how to fuse the several node labels

This definition is specified in a programming language

The label fusion is *strongly tied* to the respective node fusion:

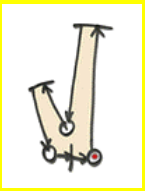


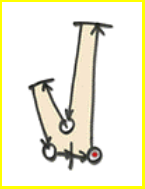
Fusion and Multiple Labels 2 / 3

The result for each node label fusion is specified in a code block in the chosen programming language.

opnml2pnml uses the Ruby programming language
The node label fusion for label **X** is defined by a method named **add_X( )**.
For example, for **marking addition** we can write:

```
def add_initialMarking()
   result = 0
   @labels.each do |label|
      result += label.xml.elements["text"].text.to_i
   end
   return "<text>#{result}</text>"
end
```





Fusion and Multiple Labels 3 / 3

Upon net subtraction the node label fusion for label **X** is defined by a method named **sub_X( )**.

For example, for **marking subtraction** we can write:

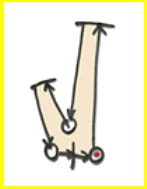
```
def sub_initialMarking()
    result = 0
    @labels.each do |label|
       result += label.sign *
                 label.xml.elements["text"].text.to_i
    end
    return "<text>#{result}</text>"
end
```

```
class Label
    attr_reader :xml, :sign
    ...
end
```

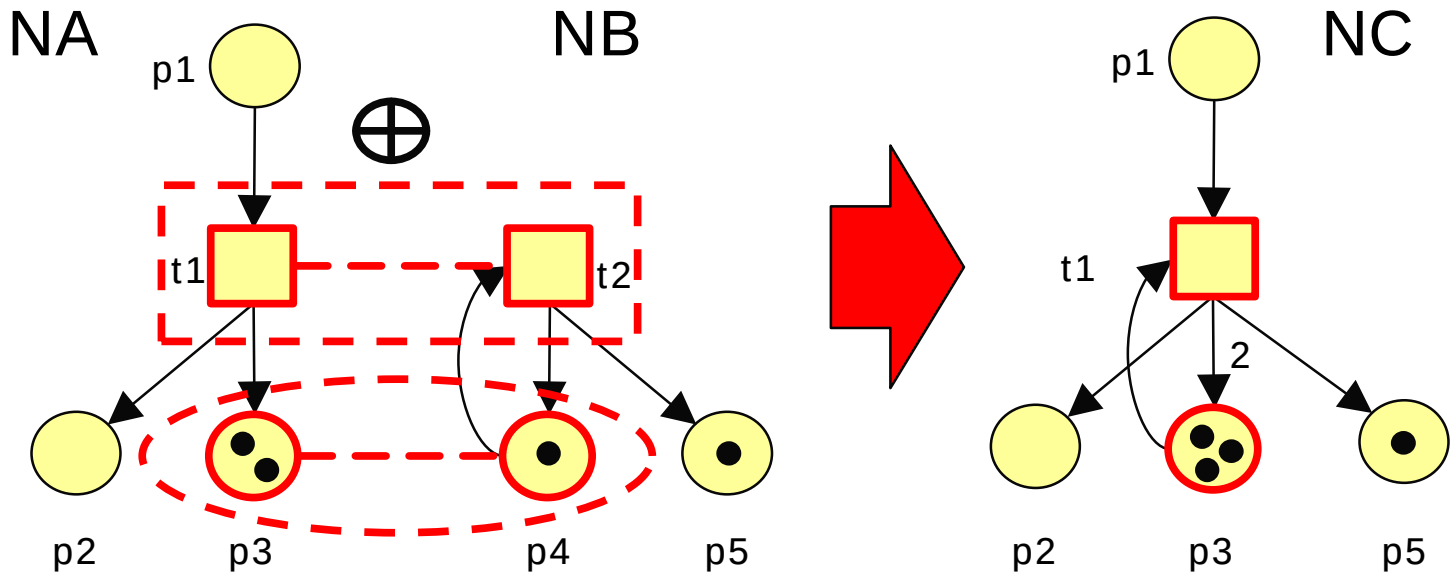
```
...
<subtraction result="return">
  <net netID="NC"/>
  <net netID="NB" sign="minus"/>
  <merge>
  ...
```

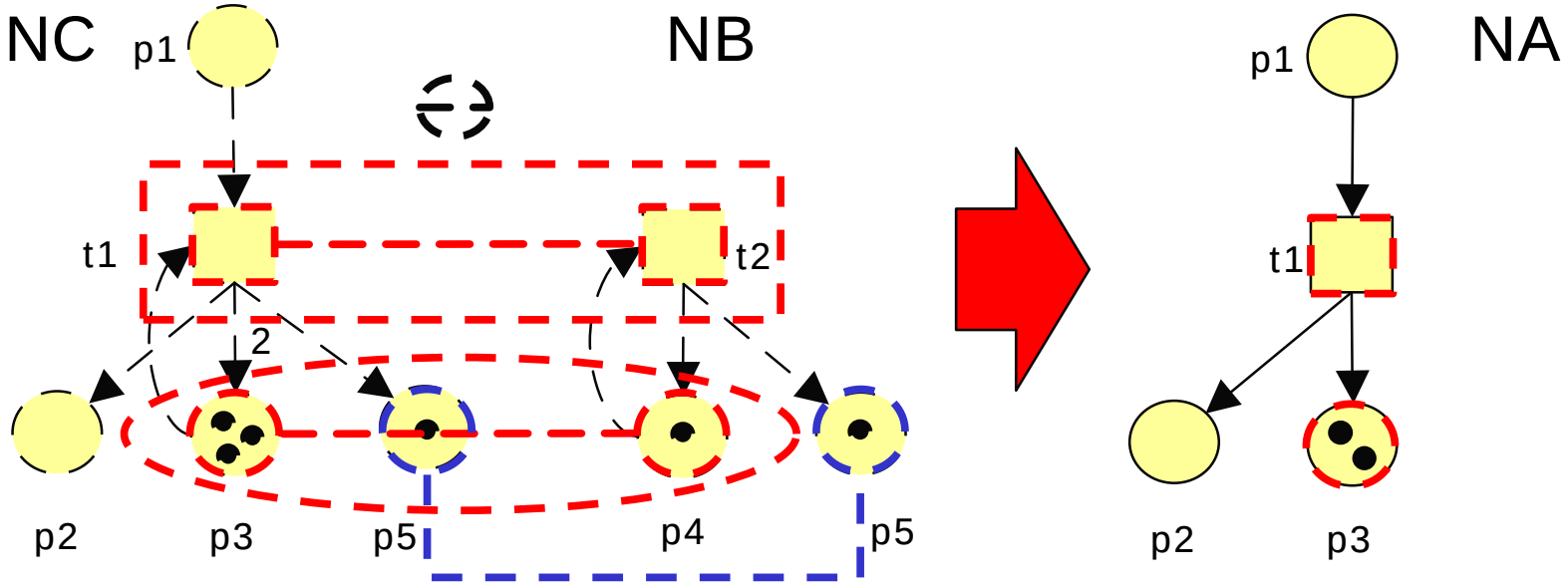
Workshop on the Definition, Implementation and Application of a Standard Interchange Format for Petri Nets

Bologna, Italy, 21-25 June, 2004



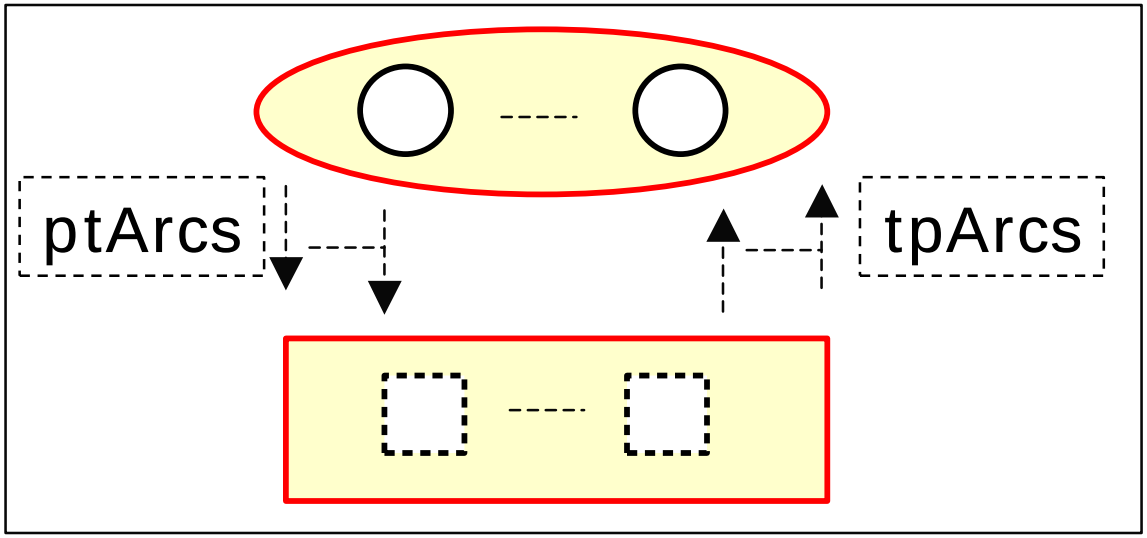
Arc Fusion



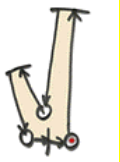


Arc Fusion

Based on the concept of *Interface Pair* (`IPair`)

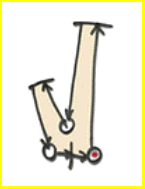


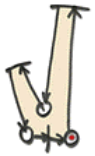
Upon net addition and net subtraction
all the interface pairs are iterated allowing the arcs inside interface pairs to be manipulated.



Arc Fusion

```
def add() # this one merges the several arcs between each interface pair
        weightPT = 0
        weightTP = 0
        @iPairsList.list.each do |ip| # for each interface pair
            ip.ptArcs.each do |ptArc| #for each place->transition arc
                weightPT = weightPT +
                           ptArc.xml.elements['inscription'].elements['text'].text.to_i
                deleteArc(ptArc)
            end
            ip.tpArcs.each do |tpArc| #for each transition->place arc
                weightTP = weightTP +
                           tpArc.xml.elements['inscription'].elements['text'].text.to_i
                deleteArc(tpArc)
            end
            p = ip.placeFE.final
            t = ip.transitionFE.final
            if weightPT > 0
                # add p->t arc element
                a = @net.add_element('arc', {'id' => p + t, 'source'=> p, 'target' => t})
                a.add_element('inscription').add_element('text').add_text(weightPT.to_s)
            end
            if weightTP > 0
                # add t->p arc element
                a = @net.add_element('arc', {'id' => t + p, 'source'=> t, 'target' => p})
                a.add_element('inscription').add_element('text').add_text(weightTP.to_s)
            end
        end
    end
```





Concluding...

We proposed:

The use of an extension to PNML allowing the specifications of Petri net models' composition and modification.

Complemented by the use of a programming language to specify the necessary ad-hoc transformations inside each Petri net class: node labels and arc labels.

Is this a generic way to compose and modify models in any Petri net class?...