

Pi Markup Language

Dagmar Köhn
Mathias John

(University of Rostock)

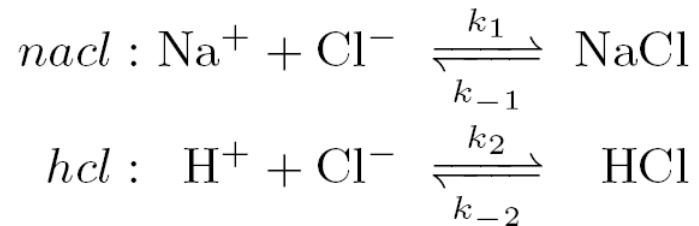
Challenges

- pi-calculus is basis for many modeling languages:
 - stochastic Pi, Beta-Binders, BioAmbients, attributed Pi, SpacePi, SPiCO
- different modeling tools exist (SpiM, BioSPi, SPiCO, JamesII)
- finding a common format
- tools should only need to consider those languages they support
- proposal: "plug-able" modules

Pi Syntax

Process	$P ::= P_1 \parallel P_2$	Parallel Composition
	$(\nu c).P$	ν Operator
	$\sum_i S_i$	Summation
	$A(\tilde{x})$	Application
Summation	$S ::= x!(\tilde{y}).P$	Send
	$x?(\tilde{y}).P$	Receive
Definition	$D ::= A(\tilde{y}) = P$	

Pi Example



Channel Definitions

nacl

hcl

Process Definitions

$$Na() = (\nu \text{ toCl}). \text{nacl}!(\text{toCl}). NaBound(\text{toCl})$$
$$NaBound(\text{free}) = \text{free}?(). Na()$$
$$H() = (\nu \text{ toCl}). \text{hcl}!(\text{toCl}). HBound(\text{toCl})$$
$$HBound(\text{free}) = \text{free}?(). H()$$
$$Cl() = \text{nacl}?(\text{toNa}). ClBound(\text{toNa}) + \text{hcl}?(\text{toH}). ClBound(\text{toH})$$
$$Cl(\text{decay}) = \text{decay}!(). Cl()$$

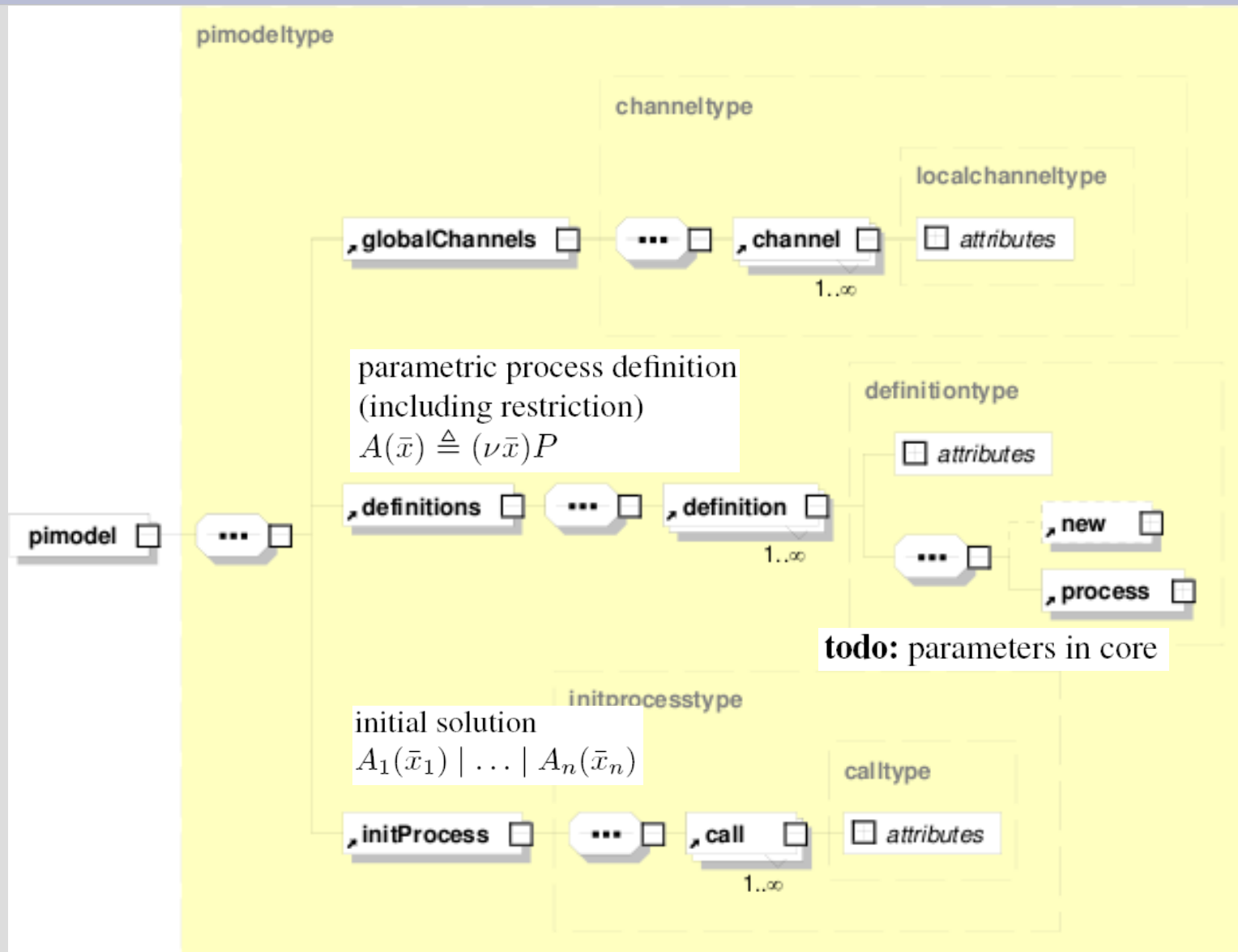
Initial Process

$$(Na() \mid \dots \mid Na() \mid H() \mid \dots \mid H() \mid Cl() \mid \dots \mid Cl())$$

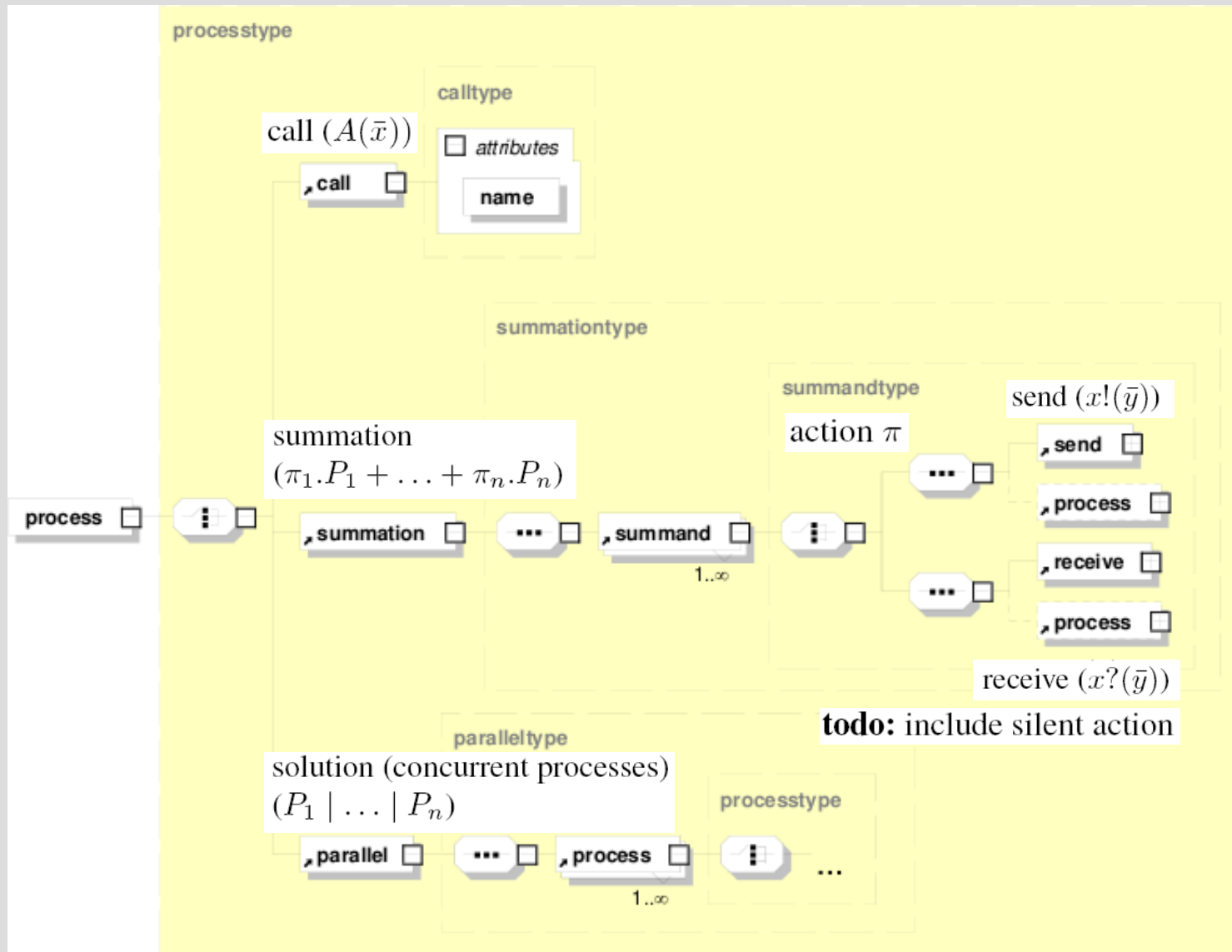
Extension Example: BioAmbients

Process	$P ::= P_1 \parallel P_2$	Parallel Composition
	$(\nu c).P$	ν Operator
	$\sum_i S_i$	Summation
	$[P]$	Ambient
Summation	$S ::= \delta x!(y).P$	Send with Direction
	$\delta x?(y).P$	Receive with Direction
	$enter\ x.P$	Enter
	$accept\ x.P$	Accept
	$exit\ x.P$	Exit
	$expel\ x.P$	Expel
	$merge\ +\ x.P$	Merge+
	$merge\ -\ x.P$	Merge-
Direction	$\delta ::= local$	Processes in Ambient
	$s2s$	Ambients in Ambient
	$p2c$	Ambient to Nested Ambient
	$c2p$	Nested Ambient to Ambient

PiML: elements & types



PiML: elements & types



PiML: example: Repressilator

Globale Kanäle definieren

a, b, c, t

Prozesse definieren

GenA = t?().(ProteinB | GenA) + a?().t?().GenA

GenB = t?().(ProteinC | GenB) + b?().t?().GenB

GenC = t?().(ProteinA | GenC) + c?().t?().GenC

ProteinA = a!().ProteinA + t?().0

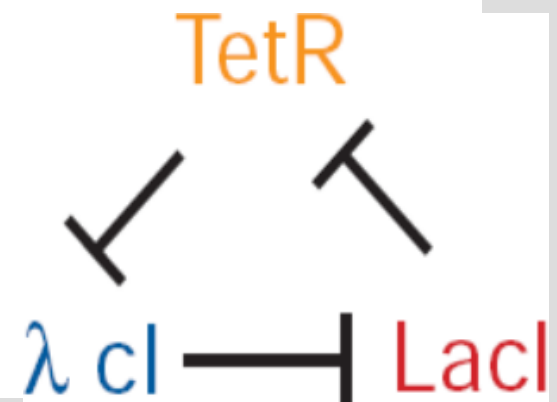
ProteinB = b!().ProteinB + t?().0

Proteinc = c!().ProteinC + t?().0

Timer = t!().Timer

Initialer Prozess

(GenA | GenB | GenC | Timer)



PiML: example: Repressilator

Globale Kanäle definieren

a, b, c, t



```
<globalChannels>  
  <channel name="a"/>  
  <channel name="b"/>  
  <channel name="c"/>  
  <channel name="t"/>  
</globalChannels>
```

PiML: example: Repressilator

Prozesse definieren

GenA = t?() . (ProteinB | GenA) + a?() . t?() . GenA

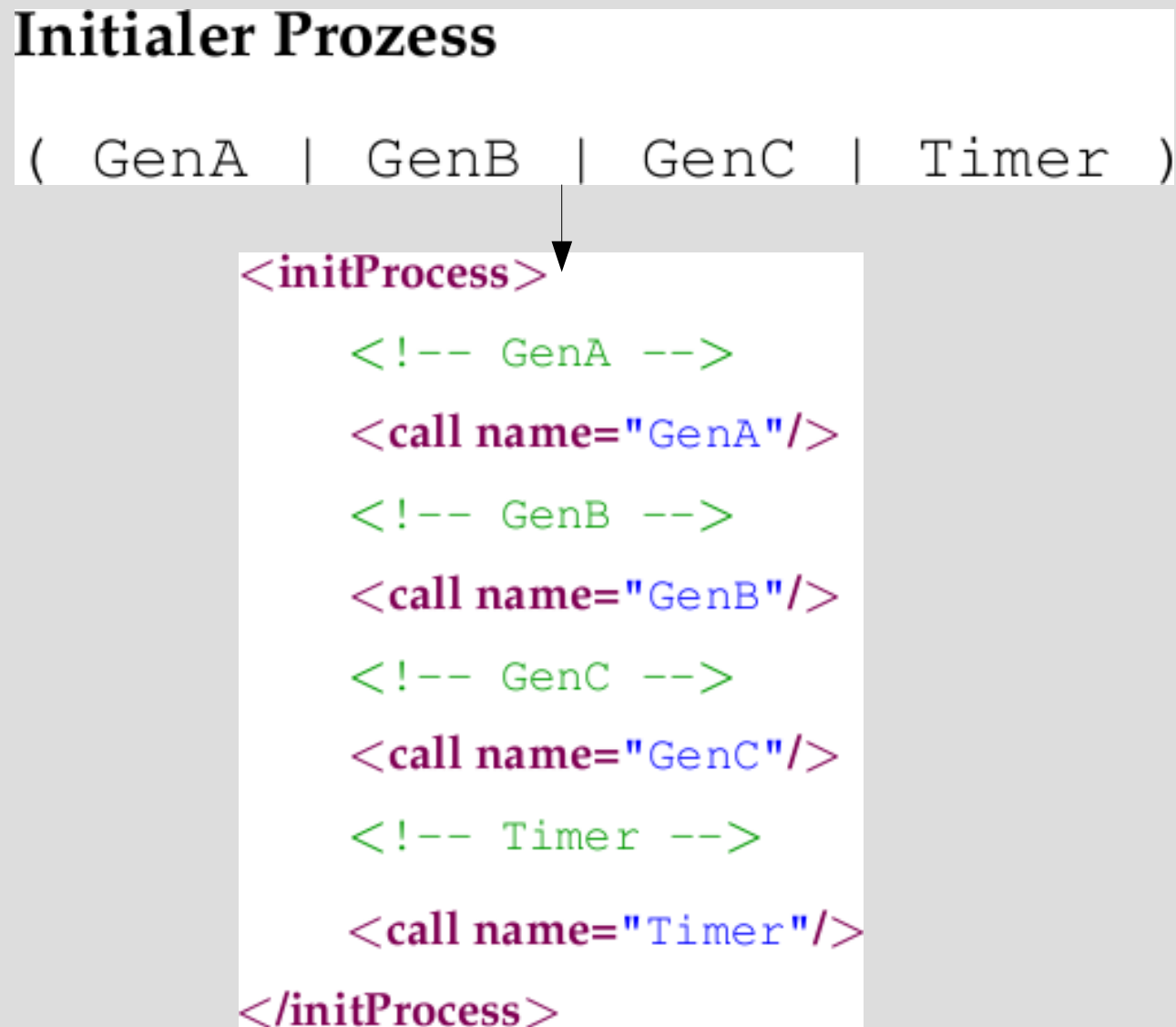
```
<definition name="GenA">
  <process>
    <summation>
      <!-- t?() . ( ProteinB | GenA ) -->
      <summand>
        <receive channel="t" />
        <process>
          <parallel>
            <process>
              <call name="ProteinB" />
            </process>
            <process>
              <call name="GenA" />
            </process>
          </parallel>
        </process>
      </summand>
    </summation>
  </process>
</definition>
```

```
<!-- a?() . t?() . GenA -->
<summand>
  <receive channel="a" />
  <process>
    <summation>
      <summand>
        <receive channel="t" />
        <process>
          <call name="GenA"></call>
        </process>
      </summand>
    </summation>
  </process>
</summand>
```

PiML: example: Repressilator

Initialer Prozess

```
( GenA | GenB | GenC | Timer )
```



```
<initProcess>  
  <!-- GenA -->  
  <call name="GenA"/>  
  <!-- GenB -->  
  <call name="GenB"/>  
  <!-- GenC -->  
  <call name="GenC"/>  
  <!-- Timer -->  
  <call name="Timer"/>  
</initProcess>
```

Modules: example: times

- Times extension

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- created by Robert Kuehn (University Rostock) -->
<!-- Pi-Calculus Times Extension Module -->
<!-- Date: 2008-07-01 -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:redefine schemaLocation="PIML.xsd">
    <xs:complexType name="calltype">
      <xs:complexContent>
        <xs:extension base="calltype">
          <xs:attribute name="times" type="xs:integer" use="optional"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:redefine>
</xs:schema>
```