Application areas
Authentication protocols
multicore programming
wireless sensor networks...

Fundamental models of computation
Turing machines
lambda-calculus
pi-calculus
modal logics...

GAP
Fundamental models of computation

Application areas

GAP

pi-calculus

applied pi calculus
spi-calculus
concurrent constraint pi calculus
others...

Authentication protocols
wireless sensor networks...
The Solution: Psi-calculi

Applications

- Cryptography
- Eq Logics
- many other exts ...

+ Data structures

Reusable

Correct

framework for mobile process calculi
Part 1

Psi-Calculi

Introduction

Jesper Bengtson, Magnus Johansson, Joachim Parrow, and Björn Victor

Department of Information Technology, Uppsala University, Sweden

e-mail address: {jesper.bengtson,magnus.johansson,joachim.parrow,bjorn.victor}@it.uu.se

Abstract. The framework of psi-calculi extends the pi-calculus with nominal datatypes for data structures and for logical assertions and conditions. These can be transmitted between processes and their names can be statically scoped as in the standard pi-calculus. Psi-calculi can capture the same phenomena as other extensions of the pi-calculus such as the applied pi-calculus, the spi-calculus, the fusion calculus, the concurrent constraint pi-calculus, and calculi with polyadic communication channels or pattern matching. Psi-calculi can be even more general, for example by allowing structure channels, higher-order formalisms such as the lambda calculus for data structures, and predicate logic for assertions.

We provide ample comparisons to related calculi and discuss a few significant applications. Our labelled operational semantics and definition of bisimulation is straightforward, without a structural congruence. We establish minimal requirements on the nominal data and logic in order to prove general algebraic properties of psi-calculi, all of which have been checked in the interactive theorem prover Isabelle. Expressiveness of psi-calculi significantly exceeds that of other formalisms, while the purity of these semantics is comparable with the original pi-calculus.

1. Introduction

The pi-calculus [MPW92] has a multitude of extensions where higher-level data structures and operations on them are given as primitive. To mention only two there are the spi-calculus by Abadi and Gordon [AG99] focusing on cryptographic primitives, and the applied pi-calculus of Abadi and Fournet [AF01] where agents can introduce statically scoped aliases of names for data, used e.g. to express how knowledge of an encryption is restricted. It is also possible to express how knowledge of an encryption is restricted.

LICS’09, TPHOLs’09, LMCS’11, SOS’09, LICS’10, SEFM’11, JLAP’12, two PhD theses...

Authors:

Jesper Bengtson, Johannes Borgström, Shuqin Huang, Magnus Johansson, Joachim Parrow, Palle Raabjerg, Björn Victor, Johannes Åman Pohjola, …
Introduction

Psi-calculi extend the pi-calculus with nominal data types for data structures and for logical assertions and conditions. These can be transmitted between processes and their names can be statically scoped as in the standard pi-calculus. Psi-calculi can capture the same phenomena as other process calculi such as the applied pi-calculus, the spi-calculus, the fusion calculus, the concurrent constraint pi-calculus, and calculi with polyadic communication channels or pattern matching.

Psi-calculi can be even more general, for example by allowing structure channels, higher-order formalisms such as the lambda calculus for data structures, and predicate logic for assertions.

We provide ample comparisons to related calculi and discuss a few significant applications. Our labelled operational semantics and definition of bisimulation is straightforward, without a structural congruence. We establish minimal requirements on the nominal data and logic in order to prove general algebraic properties of psi-calculi, all of which have been checked in the interactive theorem prover Isabelle. Expressiveness of psi-calculi significantly exceeds that of other formalisms, while the purity of these semantics is in line with the original pi-calculus.

1. Introduction

The pi-calculus [MPW92] has a multitude of extensions where higher-level data structures and operations on them are given as primitive. To mention only two: there are the spi-calculus by Abadi and Gordon [AG99] focusing on cryptographic primitives, and the applied pi-calculus of Abadi and Fournet [AF01] where agents can introduce statically scoped aliases of names for data, used e.g. to express how knowledge of an encryption is restricted.

It is also parametrised by an arbitrary signature for expressing data and an equation system for expressing data equalities. The impact of these enriched calculi is considerable with hundreds of papers applying or developing the formalisms. As Abadi and Fournet rightly observe there is a trade-off between "purity", meaning the simplicity and elegance of these models, and generality.

Received by the editors April 1, 2011.

1998 ACM Subject Classification: F.1.2, F.3.1, F.3.2.

Key words and phrases: pi-calculus, nominal sets, bisimulation, operational semantics, theorem prover.
Part 1
Psi-Calculi
Introduction
Psi by Example
pi-calculus (Informal)

<p>| nil          | 0               |
| output      | 'a&lt;b&gt;.P         |
| input       | a(x).P          |
| parallel    | P | Q         |
| replication | !P              |
| restriction | (new a)P        |
| match       | [a = b]P        |
| summation   | P + Q           |</p>
<table>
<thead>
<tr>
<th>Nil</th>
<th>Output</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel</td>
<td>Replication</td>
<td>Restriction</td>
</tr>
<tr>
<td>Match</td>
<td>Summation</td>
<td></td>
</tr>
</tbody>
</table>

### pi-calculus (Informal)

- `0`
- `'a < b`.P
- `a (x).P`
- `P | Q`
- `!P`
- `(new a)P`
- `case [aa == bb] P: P`
- `case P true Q: P`
- `[ ] true : Q`
Input prefix not binding

\[ \begin{align*}
\text{a b. } P & \quad \text{'}a\text{<c>}.Q \\
\tau & \quad a(e) \quad (|b=c|) \quad P \quad \text{'}a\text{<c>}.Q \\
|b=c| & \quad Q \quad (|b=c|) \quad P \quad Q
\end{align*} \]

1-1 transition correspondence

Equator in \(\Psi\) (a la Merro):

Communication yields \text{explicit fusion}
Crypto (A Closer Look)

Structured data: tuples
with projection equations
\[ \text{fst}(t(x,y)) = x \]
\[ \text{snd}(t(x,y)) = y \]

Facts about data: the secret \( s \) and message \( M \) hashes to \( x \)

Structured channels

Equation as condition: checks if message and hash received via \( a \) is hashed with \( s \)
Psi-calculi

Parameterized Calculi

\( P, Q ::= \) (processes in ASCII notation)

Output: \( 'M<N>.P \)

Input: \( M(x).P \)

Case: \( \text{case } cnd_1: P_1 [] ... [] cnd_n: P_n \)

Assertion: \( (|\Psi|) \)
Psi-calculi
Parameterized Calculi

\[ P, Q ::= \text{(processes in ASCII notation)} \]

Output: `\textcolor{green}{M<N>}.P`  \hspace{1cm} Input: `M(x).P`

Case: \textcolor{blue}{case cnd_1:P_1} [] ... [] \textcolor{blue}{cnd_n:P_n}

Assertion: `(|\textcolor{red}{\text{Psi}}|)`

Nil: `0`

Parallel: `P|Q`

Replication: `!P`

Restriction: `(\text{new} \ a)P`
Psi-calculi

Defining a calculus

- \( \text{subs} : T \times \text{List}(n) \rightarrow T \)  
- \( \text{chaneq} : T \rightarrow T \)  
- \( \text{compose} : A \times A \rightarrow A \)  
- \( \text{unit} : A \)  
- \( \text{entails} : A \times C \rightarrow \text{bool} \)

Satisfy simple axioms, practically anything

Provide:

- \( T \) terms
- \( C \) conditions
- \( A \) assertions

Terms are not necessarily defined by a signature
Example Psi-calculus
the pi-calculus (1)

\( T = \text{names} \)  

\( C = \{a \equiv b : a, b \text{ in names}\} \)  

conditions are equalities between names

\( A = \{\text{unit}\} = \{1\} \)  

no facts in the environment
Example Psi-calculus

the pi-calculus (2)

\[
\text{chaneq} = \lambda (a,b).a == b
\]

channel equivalence is formation of equality conditions

\[
\text{entails} = \lambda (\psi,a == b).a = b
\]

name equality is entailed whenever the names are the same
Example Psi-calculus
the pi-calculus (2)

\[ chaneq = \lambda(a,b).a == b \]
channel equivalence is formation of equality conditions

\[ entails = \lambda(\psi,a == b).a = b \]
name equality is entailed whenever the names are the same

\[ unit = 1 \quad compose = \lambda(\psi_1,\psi_2).1 \]
assertions are trivial
Example Psi-calculus
the pi-calculus

\[ T = \text{names} \]
\[ C = \{ a \equiv b : a, b \in \text{names} \} \]
\[ A = \{ \text{unit} \} = \{ 1 \} \]

\[ \text{chaneq} = \lambda (a, b). a \equiv b \]
\[ \text{compose} = \lambda (\psi_1, \psi_2). 1 \]
\[ \text{unit} = 1 \]

That's it!
Psi instance defined

1-1 correspondence of
\sim coincides

\[ T \sim A \rightarrow C \]

That instance of Psi

That instance

That instance

machine-checked
proofs

That instance

That instance
Example Psi-calculus

Fusion Calculus

\[ T = \text{names} \]
\[ C = \{a = b : a, b \in \text{names}\} \]
\[ A = \text{pow}_{\text{fin}}(\{a = b : a, b \in \text{names}\}) \]

\[ \text{chaneq} = \lambda(a, b). a = b \]
\[ \text{entails} = \lambda(\psi, a = b). (a, b) \in \text{EQ}(\psi) \]
\[ \text{compose} = \lambda(\psi_1, \psi_2). \psi_1 \cup \psi_2 \]
\[ \text{unit} = \emptyset \]
Example Psi-calculus

crypto

$$\Sigma = \{ \text{hash}(\cdot), \text{enc}(\cdot,\cdot), \text{dec}(\cdot,\cdot), \text{pk}(\cdot), \text{sk}(\cdot), \ldots \}$$

$$E = \{ \text{dec(\text{enc}(x,y),y)} = x, \ldots \}$$

$$T = \{ f(M_1,\ldots,M_n) : M_1 \in T \text{ & } \psi \in \Sigma \} \cup \text{names}$$

$$C = \{ M=N : M,N \in T \}$$

$$A = \text{pow}\{ M=N : M,N \in T \}$$

$$\text{chaneq} = \lambda (M,N) \ . \ M=N$$

$$\text{entails} = \lambda (\psi,M=N). \ E \cup \psi \ \vdash_{\text{eq}} \ M=N$$

$$\text{compose} = \lambda (\psi_1,\psi_2) \ . \ \psi_1 \cup \psi_2$$

unit = $\emptyset$
Psi-Calculi

Expressiveness

- Psi-calculi can capture
  - Cryptography, like applied pi-calculus (Abadi, Fournet 2001)
  - Fusion, like the explicit fusion calculus (Wischik, Gardner 2005)
  - Concurrent constraints, like Concurrent constraint pi (Buscemi, Montanari 2007)
  - Polyadic synchronisation (Carbone, Maffeis 2003)
  - Pattern matching, higher order values (various)

- Extensions
  - Higher Order
  - Broadcast
  - Sorts, match
  - Types (Hüttel, CONCUR ’11)
Psi-Calculi: A Framework for Mobile Processes with Nominal Data and Logic

Jesper Bengtson, Magnus Johansson, Joachim Parrow, and Björn Victor

Department of Information Technology, Uppsala University, Sweden
e-mail address: \{jesper.bengtson, magnus.johansson, joachim.parrow, bjorn.victor\}@it.uu.se

Abstract. The framework of psi-calculi extends the pi-calculus with nominal datatypes for data structures and for logical assertions and conditions. These can be transmitted between processes and their names can be statically scoped as in the standard pi-calculus. Psi-calculi can capture the same phenomena as other extensions of the pi-calculus such as the applied pi-calculus, the spi-calculus, the fusion calculus, the concurrent constraint pi-calculus, and calculi with polyadic communication channels or pattern matching.

Psi-calculi can be even more general, for example by allowing structured channels, higher-order formalisms such as the lambda calculus for data structures, and predicate logic for assertions.

We provide ample comparisons to related calculi and discuss a few significant applications. Our labelled operational semantics and definition of bisimulation is straightforward, without a structural congruence. We establish minimal requirements on the nominal data and logic in order to prove general algebraic properties of psi-calculi, all of which have been checked in the interactive theorem prover Isabelle. Expressiveness of psi-calculi significantly exceeds that of other formalisms, while the purity of these semantics is not with the original pi-calculus.

1. Introduction

The pi-calculus [MPW92] has a multitude of extensions where higher-level data structures and operations on them are given as primitive. To mention only two there are the spi-calculus by Abadi and Gordon [AG99] focusing on cryptographic primitives, and the applied pi-calculus of Abadi and Fournet [AF01] where agents can introduce statically scoped aliases of names for data, used e.g. to express how knowledge of an encryption is restricted.

It is also parametrised by an arbitrary signature for expressing data and an equation system for expressing data equalities. The impact of these enriched calculi is considerable with hundreds of papers applying or developing the formalisms. As Abadi and Fournet rightly observe there is a trade-off between “purity”, meaning the simplicity and elegance.

Received by the editors April 1, 2011.

1998 ACM Subject Classification: F.1.2, F.3.1, F.3.2.

Key words and phrases: pi-calculus, nominal sets, bisimulation, operational semantics, theorem prover.

Publications:
LICS’09, TPHOLs’09, LMCS’11, SOS’09, LICS’10, SEFM’11, JLAP’12, two PhD theses ...

Authors:
Jesper Bengtson, Johannes Borgström, Shuqin Huang, Magnus Johansson, Joachim Parrow, Palle Raabjerg, Björn Victor, Johannes Åman Pohjola, ...

Pwb
Tool for Psi

Workbench
Part 2
Psi-Calculi
Workbench
Tool for Psi
Psi-Calculi
Workbench

Framework for implementing Psi-Calculi instances
Experimental Platform for experimentation with semantics and pi-calculus extensions
Free
Implemented in SML (PolyML)

For every psi implementation includes
Transition simulator
Weak bisim generator
Psi-Calculi Workbench

Architecture

Psi-Calculi Workbench (Pwb)

Implementer

Supp Lib
Nominal PP Parser Comb ...

TCA
chaneq
entails
compose
unit
subst

Transition
Constraint Solver

Symbolic Evaluator

Modular Pluggable
Psi-Calculi Workbench

Constraint Solver

Pwb
Implementer

Process

Symbolic Evaluator → Transition Constraint → Transition Constraint Solver
Psi-Calculi Workbench

Architecture

Psi Core

Transition Constraint Solver

Bisimulation Constraint Solver

Symbolic Evaluator

Bisim Cnstr Generator

Psi

TCA

chaneq
entails
compose
unit
subst

Supp Lib
Nominal PP
Parser Comb ...

Pwb

Implementer

Command Interp

Parser

Pretty Printer
Demo
Psi-Calculi
Extensions (in progress)

+ Nominal Free Algebras

+ Broadcast communication
  Johannes Borgström

+ Hüttel’s Types
  Amin Khorsandi, MSc Thesis
References

Get Pwb at

$ wget www.it.uu.se/research/group/mobility/applied/psiworkbench

Take a look

$ wget www.it.uu.se/research/group/mobility

Read

LICS’09, TPHOLs’09, LMCS’11, SOS’09, LICS’10, SEFM’11, JLAP’12
Exercising Psi-calculi: A Psi-calculi workbench (my MSc Thesis)