Advancing Concurrent System Verification

Type Based Approach and Tools

Ramūnas Gutkovas

Licentiate Seminar

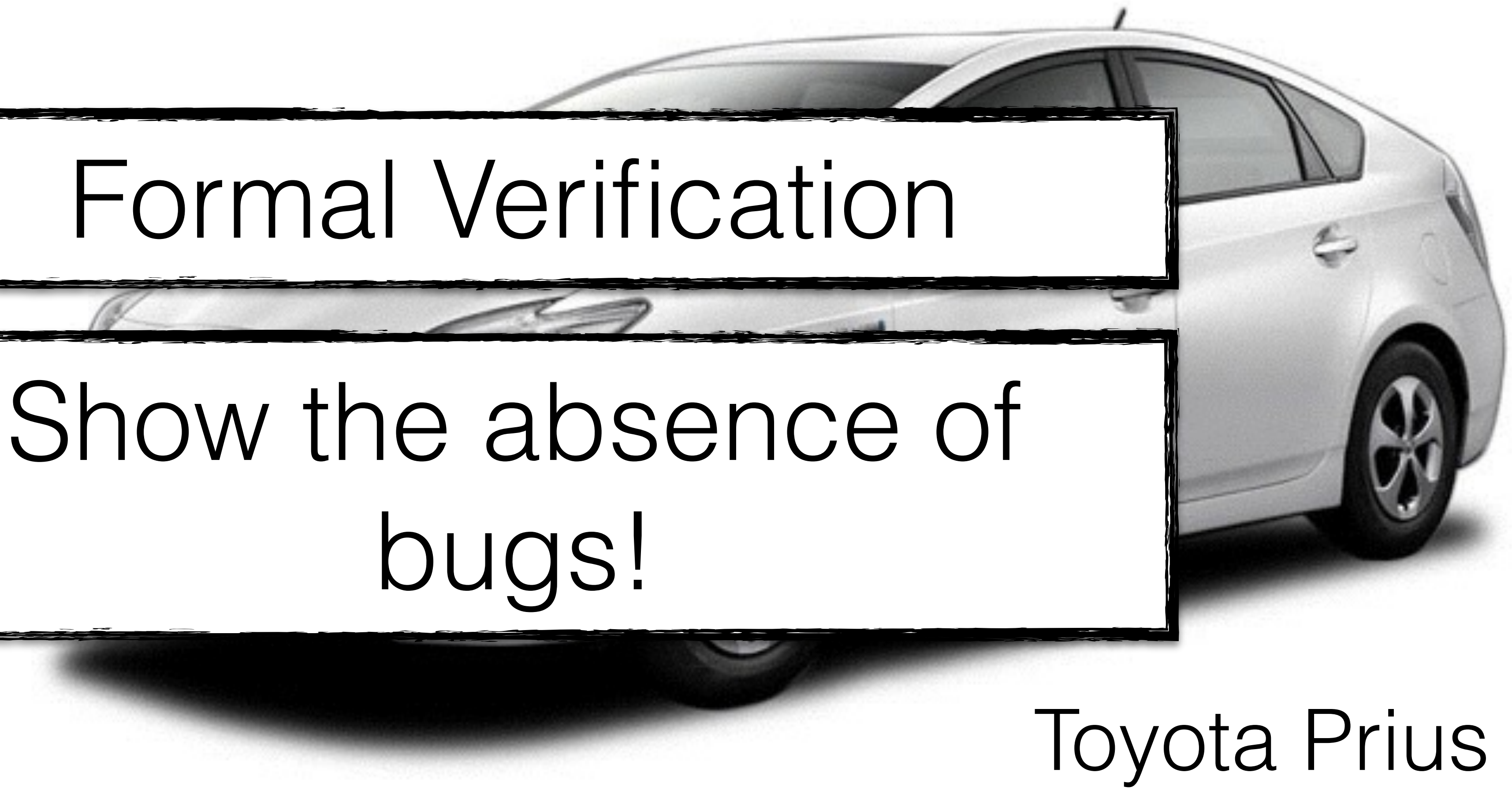
2014 October 20

Uppsala University

Supported by ProFuN

In 2010, Toyota **recalled** 400,000 vehicles

to correct a **software** "**glitch**" in ABS



Formal Verification

Show the absence of bugs!

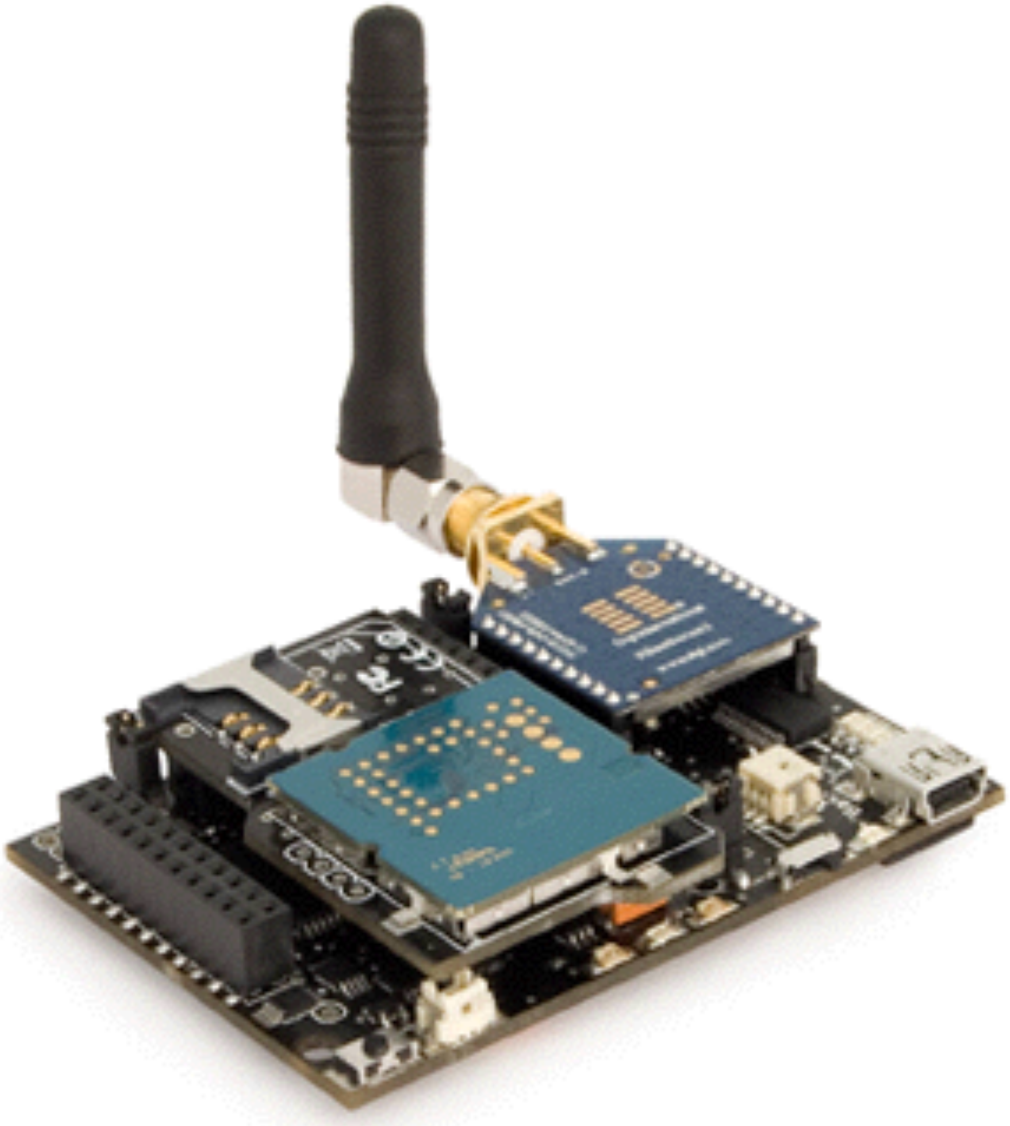
Toyota Prius

Testing shows the presence, not the absence of bugs!
- E. W. Dijkstra

Background

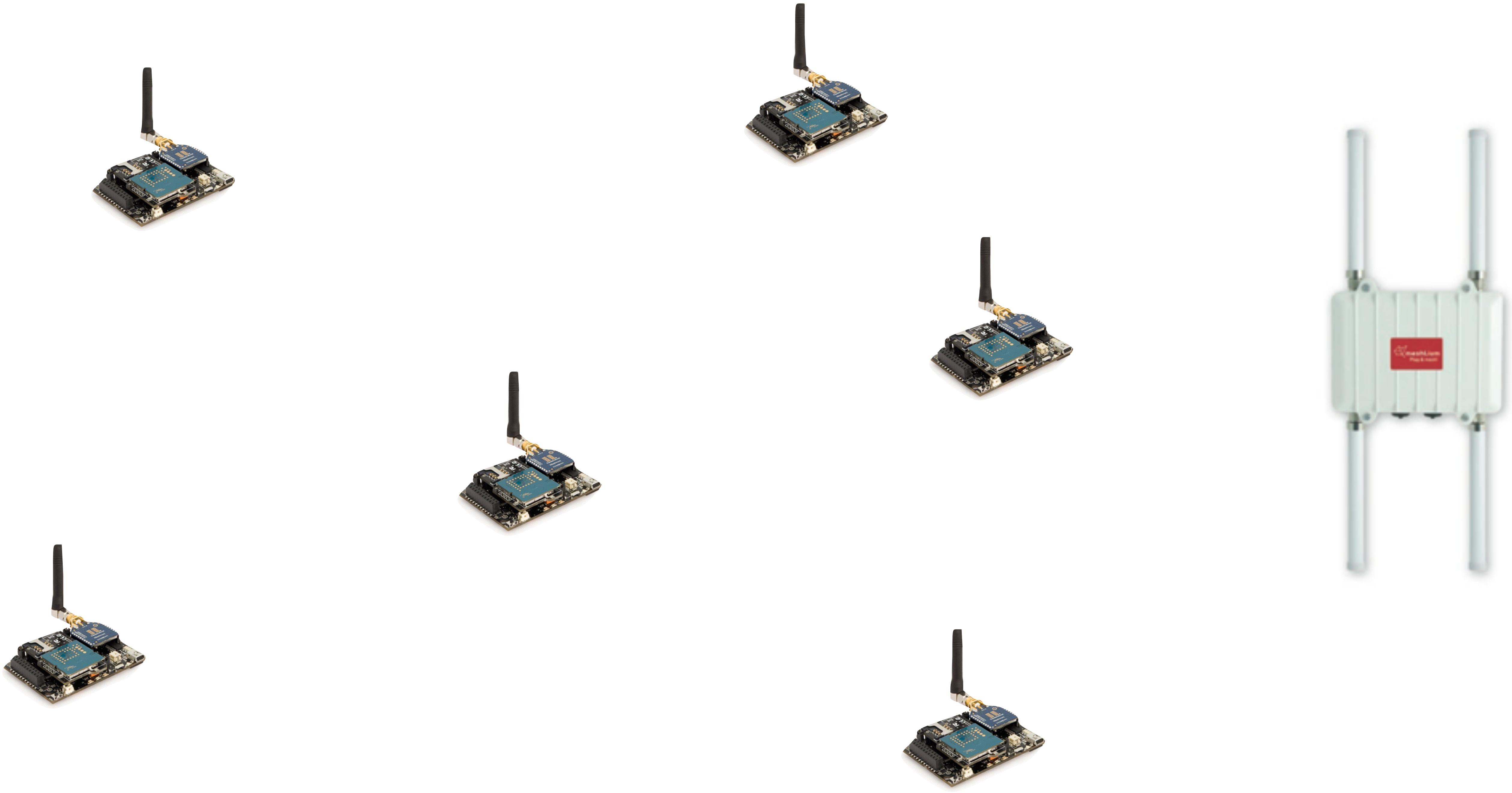
Wireless Sensor Network

ProFuN project



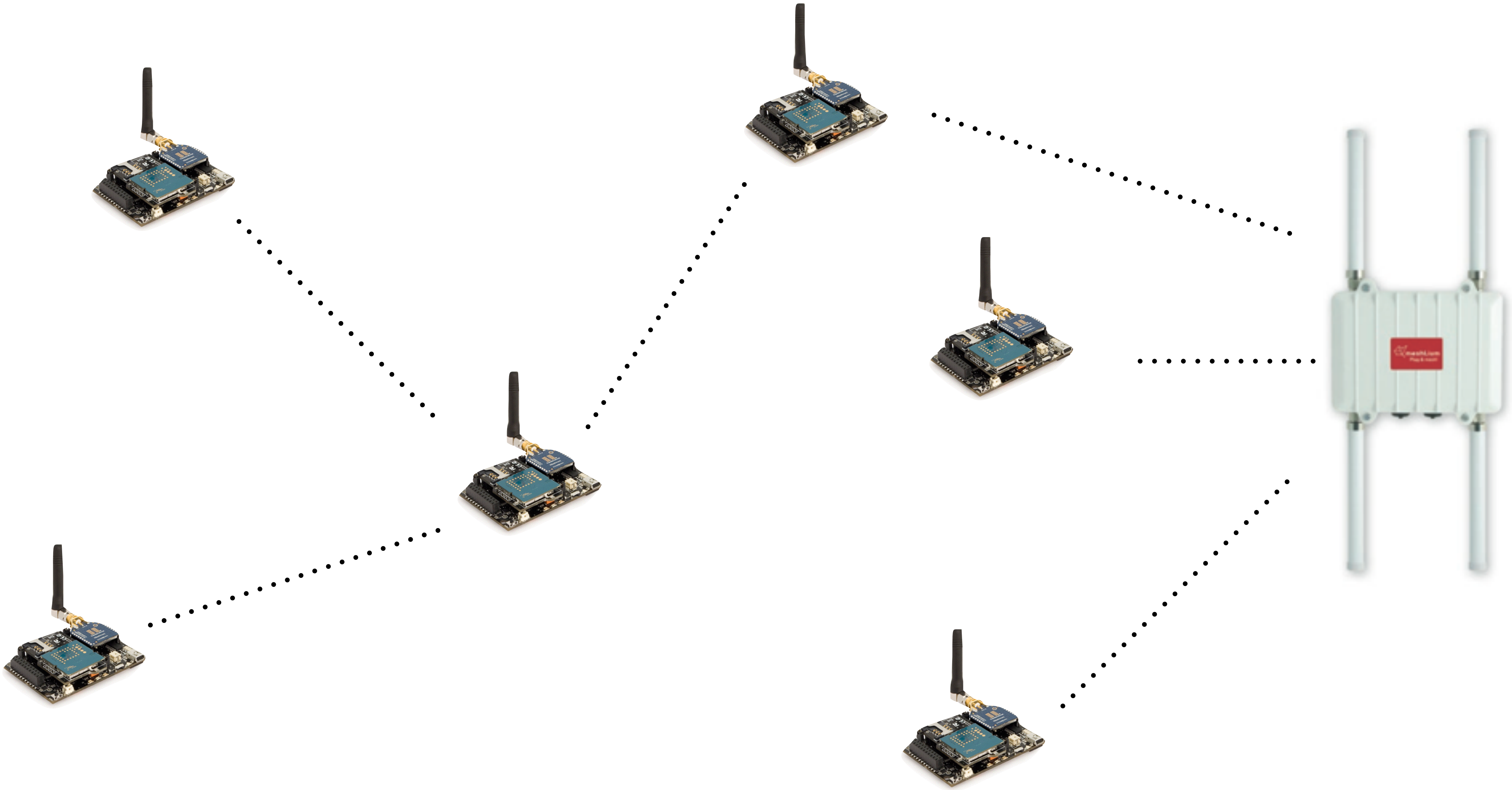
Wireless Sensor Network

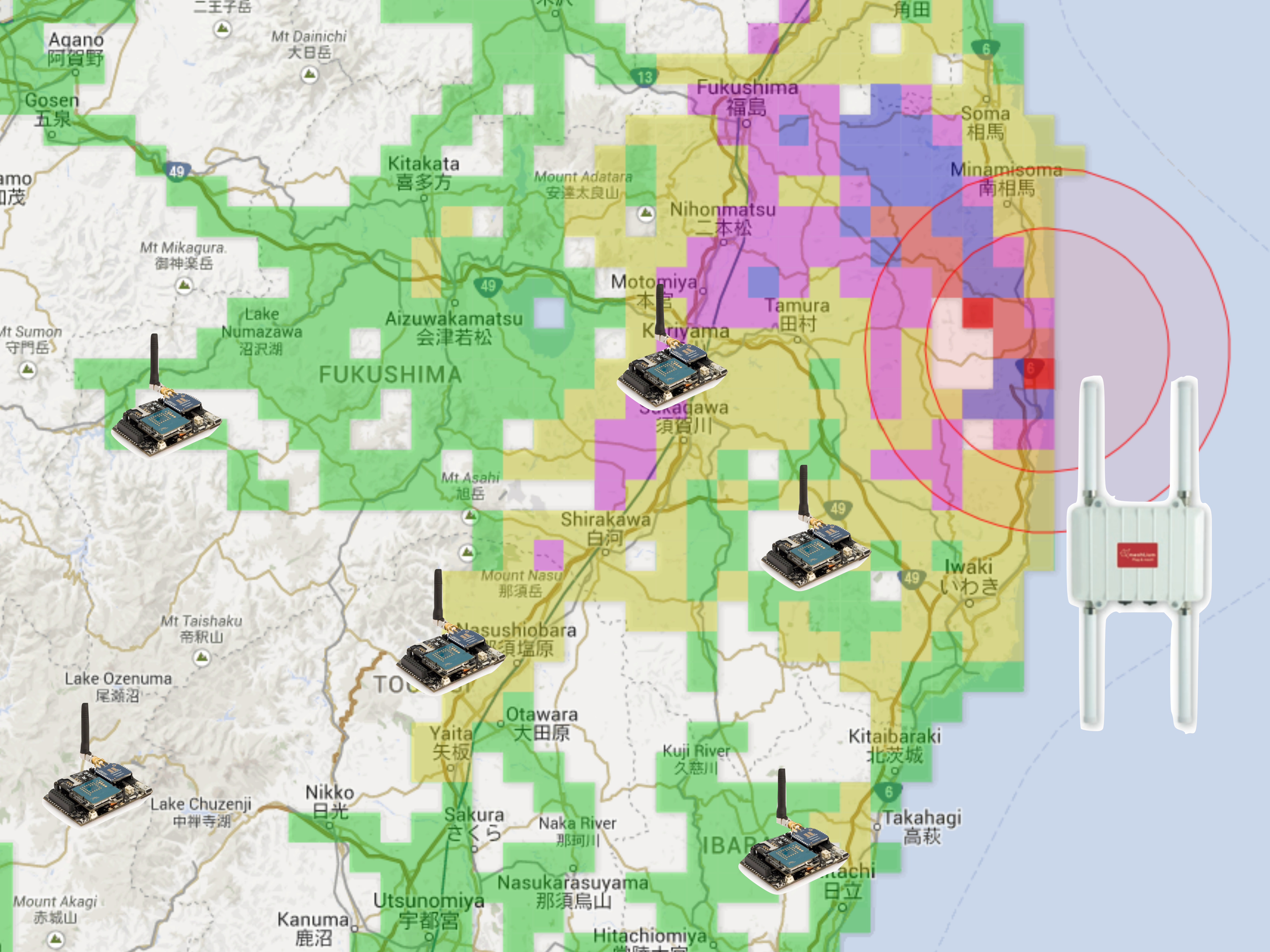
ProFuN project



Wireless Sensor Network

ProFuN project




二王子岳
Mt Dainichi
大日岳
角田
Agano
阿賀野
Gosen
五泉
13
6
Fukushima
福島
Soma
相馬
Kitakata
喜多方
Mount Adatara
安達太良山
49
Minamisoma
南相馬
Nihonmatsu
二本松
Mt Mikagura
御神楽岳
Motomiya
本宮
49
Aizuwakamatsu
会津若松
Tamura
田村
Lake
Numazawa
沼沢湖
Koriyama
Mt Sumon
守門岳
FUKUSHIMA
Sukagawa
須賀川
6
Mt Asahi
旭岳
Shirakawa
白河
49
Mount Nasu
那須岳
49
Iwaki
いわき
Mt Taishaku
帝釈山
Nasushiobara
那須塩原
Lake Ozenuma
尾瀬沼
Otawara
大田原
Yaita
矢板
Kuji River
久慈川
Kitaibaraki
北茨城
Nikko
日光
Lake Chuzenji
中禅寺湖
Sakura
さくら
Naka River
那珂川
6
Takahagi
高萩
Nasukarasuyama
那須烏山
Hitachi
日立
Mount Akagi
赤城山
Utsunomiya
宇都宮
Kanuma
鹿沼
Hitachiomiya

Psi-calculi

Psi-

- A family of languages, known as process calculi, for modelling concurrent systems
- A framework for mobile process calculi ("pi-calculus extensions") for **applications**
- Straightforward semantics, **reusable** theory (holds in all psi-calculi)
- **Correct**: machine-checked proofs! (Isabelle with Nominal Package)

Syntax

Parameters:

M, N: **T** (terms)

φ : **C** (conditions)

Ψ : **A** (assertions)

object

output $\overline{M}\, N.P$

subjects

pattern

input $\underline{M}(\lambda\widetilde{x})\boxed{N}.P$

assertion $(\!|\Psi|\!)$ “facts”

condition

$\mathbf{case}\ \varphi_1 : P_1\ [\!]\ \cdots\ [\!]\ \boxed{\varphi_n} : P_n$

like guarded commands, **if-then-else**

the usual: $\mathbf{0}$ $(\nu a)P$ $P \mid Q$ $!\,P$

Cook a psi-calculus

Define terms **T** (e.g. data terms, channels) M, N

conditions **C** (e.g. for if-then-else) φ

assertions **A** (statements about e.g. terms) Ψ

can be practically anything

Cook a psi-calculus

Define terms **T**, conditions **C,** assertions **A**

M, N

$\varphi \quad \Psi$

Define substitution on these (satisfy axioms)

$[\widetilde{a} := \widetilde{M}]$

Define operators:

$\dot{\leftrightarrow} : \mathbf{T} \times \mathbf{T} \to \mathbf{C}$	Channel equivalence
$\otimes : \mathbf{A} \times \mathbf{A} \to \mathbf{A}$	Composition
$\mathbf{1} : \mathbf{A}$	Unit assertion
$\vdash \subseteq \mathbf{A} \times \mathbf{C}$	Entailment

(practically anything)

$\dot{\prec} : \mathbf{T} \times \mathbf{T} \to \mathbf{C}$	Broadcast Output Connectivity
$\dot{\succ} : \mathbf{T} \times \mathbf{T} \to \mathbf{C}$	Broadcast Input Connectivity

Example

$M \in \mathbf{T}$

$\varphi \in \mathbf{C}$

$\Psi \in \mathbf{A}$

$$\begin{aligned}
M &::= \mathsf{init}(M) \mid a \mid i \in \mathbb{N} \\
\varphi &::= M = M' \mid M \prec M' \\
\Psi &::= M \prec M', \Psi \mid \epsilon
\end{aligned}$$

$$\begin{aligned}
\overline{\mathsf{init}(1)}123.\mathbf{0} &\mid \\
\underline{\mathsf{init}(2)}(\lambda x)x.\mathbf{0} &\mid \\
\underline{\mathsf{init}(3)}(\lambda y)y.\mathbf{case}\ y = 3 : P &\mid \\
(\!| 1 \prec 2, 1 \prec 3 |\!) &
\end{aligned}$$

Example

$M \in \mathbf{T}$

$\varphi \in \mathbf{C}$

$\Psi \in \mathbf{A}$

$$
\begin{aligned}
M &::= \mathsf{init}(M) \mid a \mid i \in \mathbb{N} \\
\varphi &::= M = M' \mid M \prec M' \\
\Psi &::= M \prec M', \Psi \mid \epsilon
\end{aligned}
$$

Transition relation ~ semantics

action/behaviour

$$
\begin{array}{r}
\mathsf{init}(1)123.\mathbf{0} \mid \\
\mathsf{init}(2)(\lambda x)x.\mathbf{0} \mid \\
\mathsf{init}(3)(\lambda y)y.\mathbf{case}\ y = 3 : P \mid \\
\llparenthesis 1 \prec 2, 1 \prec 3 \rrparenthesis
\end{array}
\xrightarrow{\alpha}
\begin{array}{r}
\mathbf{0} \mid \\
\mathbf{0} \mid \\
\mathbf{case}\ 123 = 3 : P[y := 123] \mid \\
\llparenthesis 1 \prec 2, 1 \prec 3 \rrparenthesis
\end{array}
$$

Example

$M \in \mathbf{T}$

$\varphi \in \mathbf{C}$

$\Psi \in \mathbf{A}$

$$\begin{array}{rl} M & ::= \mathsf{init}(M) \mid a \mid i \in \mathbb{N} \\ \varphi & ::= M = M' \mid M \prec M' \\ \Psi & ::= M \prec M', \Psi \mid \epsilon \end{array}$$

User defined logic

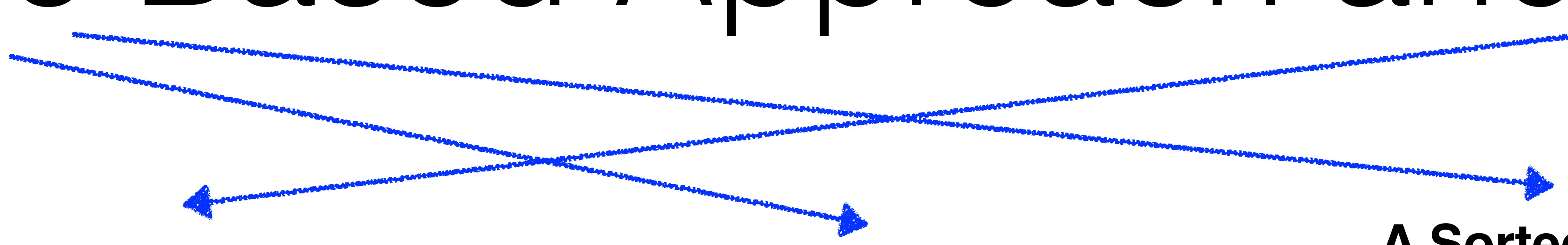
$$\boxed{\begin{array}{r} \Psi, M \prec M' \vdash \mathsf{init}(M) \prec \mathsf{init}(M') \\ \Psi \vdash M = M' \text{ if } M = M' \end{array}}$$

$$\begin{array}{r} \overline{\mathsf{init}(1)}123.\mathbf{0} \mid \\ \underline{\mathsf{init}(2)}(\lambda x)x.\mathbf{0} \mid \\ \underline{\mathsf{init}(3)}(\lambda y)y.\mathbf{case}\ y = 3 : P \mid \\ (\!| 1 \prec 2, 1 \prec 3 |\!) \end{array} \xrightarrow{\alpha} \begin{array}{r} \mathbf{0} \mid \\ \mathbf{0} \mid \\ \mathbf{case}\ 123 = 3 : P[y := 123] \mid \\ (\!| 1 \prec 2, 1 \prec 3 |\!) \end{array}$$

Advancing Concurrent System Verification

- A tool factory the Psi-Calculi Workbench for concurrent system verification
- Session types for broadcast communication and unreliable systems
- More expressivity: generalised pattern-matching and sorts for psi-calculi

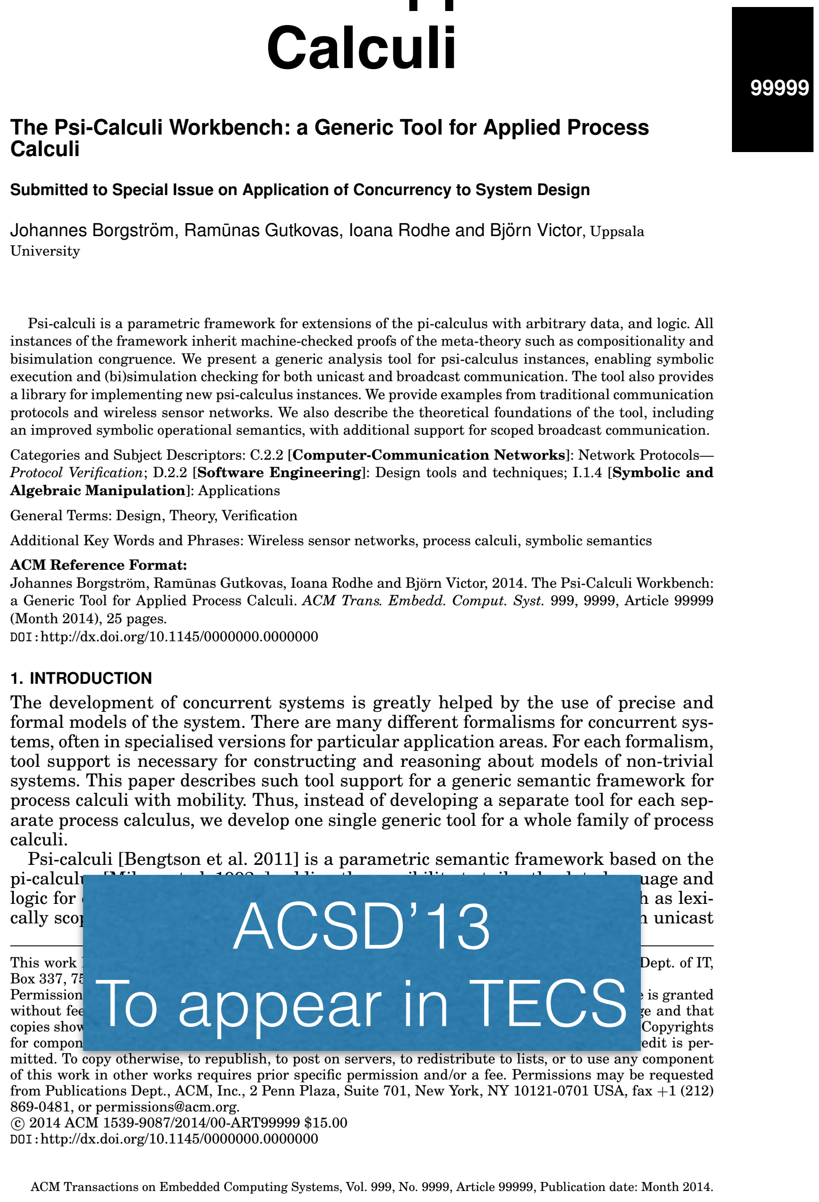
Type Based Approach and Tools



The Psi-Calculi Workbench: a Generic Tool for Applied Process Calculi

Session Types for Broadcasting

A Sorted Semantic Framework for Applied Process Calculi



99999

The Psi-Calculi Workbench: a Generic Tool for Applied Process Calculi

Submitted to Special Issue on Application of Concurrency to System Design

Johannes Borgström, Ramūnas Gutkovas, Ioana Rodhe and Björn Victor, Uppsala University

Psi-calculi is a parametric framework for extensions of the pi-calculus with arbitrary data, and logic. All instances of the framework inherit machine-checked proofs of the meta-theory such as compositionality and bisimulation congruence. We present a generic analysis tool for psi-calculus instances, enabling symbolic execution and (bi)simulation checking for both unicast and broadcast communication. The tool also provides a library for implementing new psi-calculus instances. We provide examples from traditional communication protocols and wireless sensor networks. We also describe the theoretical foundations of the tool, including an improved symbolic operational semantics, with additional support for scoped broadcast communication.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Protocol Verification*; D.2.2 [**Software Engineering**]: Design tools and techniques; I.1.4 [**Symbolic and Algebraic Manipulation**]: Applications

General Terms: Design, Theory, Verification

Additional Key Words and Phrases: Wireless sensor networks, process calculi, symbolic semantics

ACM Reference Format:
Johannes Borgström, Ramūnas Gutkovas, Ioana Rodhe and Björn Victor, 2014. The Psi-Calculi Workbench: a Generic Tool for Applied Process Calculi. *ACM Trans. Embedd. Comput. Syst.* 999, 9999, Article 99999 (Month 2014), 25 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

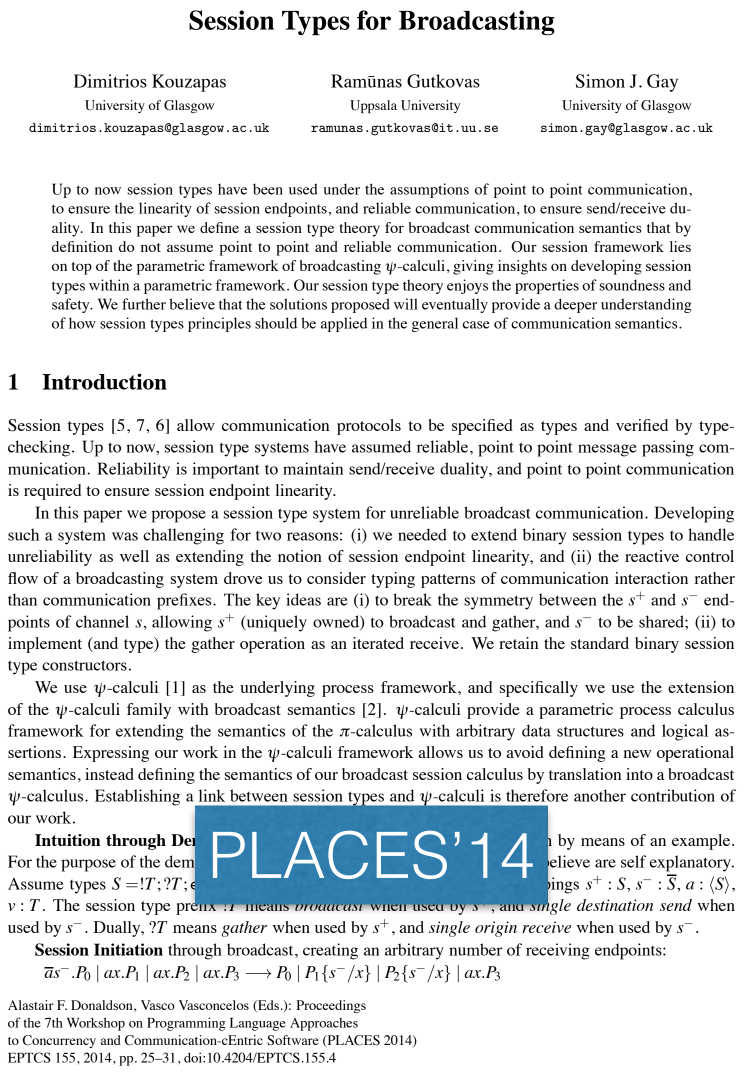
The development of concurrent systems is greatly helped by the use of precise and formal models of the system. There are many different formalisms for concurrent systems, often in specialised versions for particular application areas. For each formalism, tool support is necessary for constructing and reasoning about models of non-trivial systems. This paper describes such tool support for a generic semantic framework for process calculi with mobility. Thus, instead of developing a separate tool for each separate process calculus, we develop one single generic tool for a whole family of process calculi.

Psi-calculi [Bengtson et al. 2011] is a parametric semantic framework based on the
pi-calculu uage and
logic for h as lexi-
cally scop unicast

This work Dept. of IT,
Box 337, 75
Permission is granted
without fee ge and that
copies show Copyrights
for compon edit is per-
mitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2014 ACM 1539-9087/2014/00-ART99999 $15.00
DOI:http://dx.doi.org/10.1145/0000000.0000000

ACM Transactions on Embedded Computing Systems, Vol. 999, No. 9999, Article 99999, Publication date: Month 2014.

ACSD'13
To appear in TECS



Session Types for Broadcasting

Dimitrios Kouzapas
University of Glasgow
dimitrios.kouzapas@glasgow.ac.uk

Ramūnas Gutkovas
Uppsala University
ramunas.gutkovas@it.uu.se

Simon J. Gay
University of Glasgow
simon.gay@glasgow.ac.uk

Up to now session types have been used under the assumptions of point to point communication, to ensure the linearity of session endpoints, and reliable communication, to ensure send/receive duality. In this paper we define a session type theory for broadcast communication semantics that by definition do not assume point to point and reliable communication. Our session framework lies on top of the parametric framework of broadcasting ψ-calculi, giving insights on developing session types within a parametric framework. Our session type theory enjoys the properties of soundness and safety. We further believe that the solutions proposed will eventually provide a deeper understanding of how session types principles should be applied in the general case of communication semantics.

1 Introduction

Session types [5, 7, 6] allow communication protocols to be specified as types and verified by type-checking. Up to now, session type systems have assumed reliable, point to point message passing communication. Reliability is important to maintain send/receive duality, and point to point communication is required to ensure session endpoint linearity.

In this paper we propose a session type system for unreliable broadcast communication. Developing such a system was challenging for two reasons: (i) we needed to extend binary session types to handle unreliability as well as extending the notion of session endpoint linearity, and (ii) the reactive control flow of a broadcasting system drove us to consider typing patterns of communication interaction rather than communication prefixes. The key ideas are (i) to break the symmetry between the s^+ and s^- endpoints of channel s, allowing s^+ (uniquely owned) to broadcast and gather, and s^- to be shared; (ii) to implement (and type) the gather operation as an iterated receive. We retain the standard binary session type constructors.

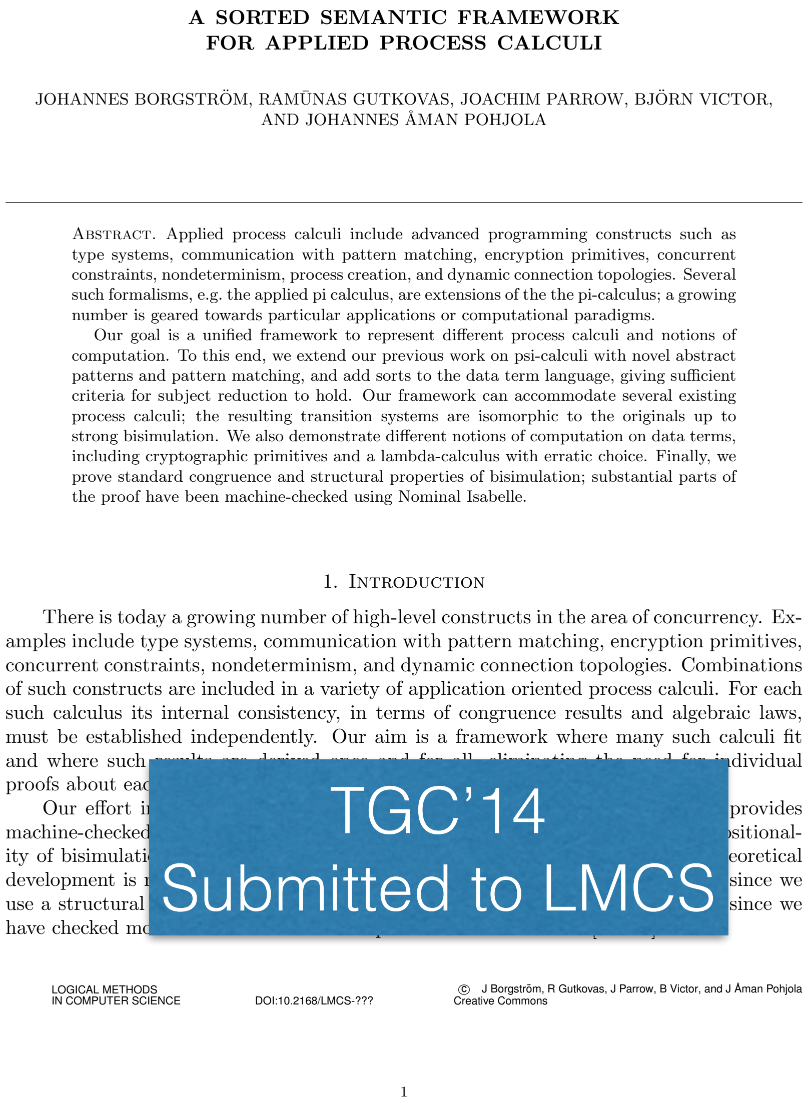
We use ψ-calculi [1] as the underlying process framework, and specifically we use the extension of the ψ-calculi family with broadcast semantics [2]. ψ-calculi provide a parametric process calculus framework for extending the semantics of the π-calculus with arbitrary data structures and logical assertions. Expressing our work in the ψ-calculi framework allows us to avoid defining a new operational semantics, instead defining the semantics of our broadcast session calculus by translation into a broadcast ψ-calculus. Establishing a link between session types and ψ-calculi is therefore another contribution of our work.

Intuition through De by means of an example.
For the purpose of the dem elieve are self explanatory.
Assume types $S = !T;?T;$ e ings $s^+ : S$, $s^- : \overline{S}$, $a : \langle S \rangle$,
$v : T$. The session type prefix $!T$ means *broadcast* when used by s^+, and *single destination send* when used by s^-. Dually, $?T$ means *gather* when used by s^+, and *single origin receive* when used by s^-.

Session Initiation through broadcast, creating an arbitrary number of receiving endpoints:
$$\overline{a}s^-.P_0 \mid ax.P_1 \mid ax.P_2 \mid ax.P_3 \longrightarrow P_0 \mid P_1\{s^-/x\} \mid P_2\{s^-/x\} \mid ax.P_3$$

Alastair F. Donaldson, Vasco Vasconcelos (Eds.): Proceedings
of the 7th Workshop on Programming Language Approaches
to Concurrency and Communication-cEntric Software (PLACES 2014)
EPTCS 155, 2014, pp. 25–31, doi:10.4204/EPTCS.155.4

PLACES'14



**A SORTED SEMANTIC FRAMEWORK
FOR APPLIED PROCESS CALCULI**

JOHANNES BORGSTRÖM, RAMŪNAS GUTKOVAS, JOACHIM PARROW, BJÖRN VICTOR,
AND JOHANNES ÅMAN POHJOLA

Abstract. Applied process calculi include advanced programming constructs such as type systems, communication with pattern matching, encryption primitives, concurrent constraints, nondeterminism, process creation, and dynamic connection topologies. Several such formalisms, e.g. the applied pi calculus, are extensions of the the pi-calculus; a growing number is geared towards particular applications or computational paradigms.

Our goal is a unified framework to represent different process calculi and notions of computation. To this end, we extend our previous work on psi-calculi with novel abstract patterns and pattern matching, and add sorts to the data term language, giving sufficient criteria for subject reduction to hold. Our framework can accommodate several existing process calculi; the resulting transition systems are isomorphic to the originals up to strong bisimulation. We also demonstrate different notions of computation on data terms, including cryptographic primitives and a lambda-calculus with erratic choice. Finally, we prove standard congruence and structural properties of bisimulation; substantial parts of the proof have been machine-checked using Nominal Isabelle.

1. Introduction

There is today a growing number of high-level constructs in the area of concurrency. Examples include type systems, communication with pattern matching, encryption primitives, concurrent constraints, nondeterminism, and dynamic connection topologies. Combinations of such constructs are included in a variety of application oriented process calculi. For each such calculus its internal consistency, in terms of congruence results and algebraic laws, must be established independently. Our aim is a framework where many such calculi fit and where such results are derived once and for all, eliminating the need for individual
proofs about eac
Our effort i provides
machine-checked ositional-
ity of bisimulati eoretical
development is r since we
use a structural since we
have checked mo

LOGICAL METHODS
IN COMPUTER SCIENCE

DOI:10.2168/LMCS-???

© J Borgström, R Gutkovas, J Parrow, B Victor, and J Åman Pohjola
Creative Commons

1

TGC'14
Submitted to LMCS

Contributions

Tools

Tool is essential for verifying non-trivial systems!

Many tools

mCRL2

ABC

SBC

PiET

ProVerif

Concurrency Workbench

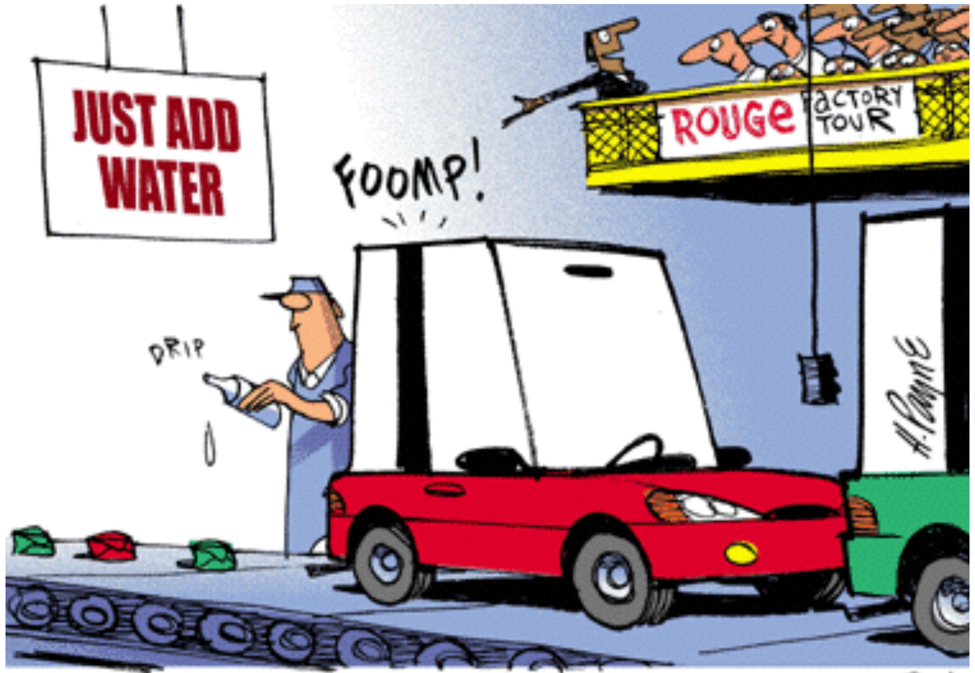
But specialised!

Mobility Workbench

Petruchio

PAT3

Psi-Calculi Workbench



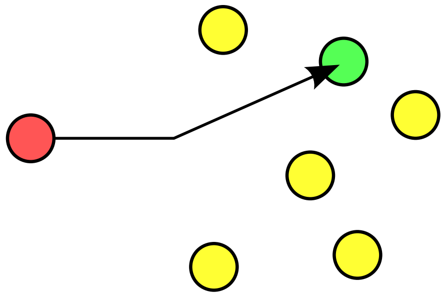
EMAIL: hpayne@detnews.com ©04

- Tool factory: define your own tool!
- Based on the parametric psi-calculi framework

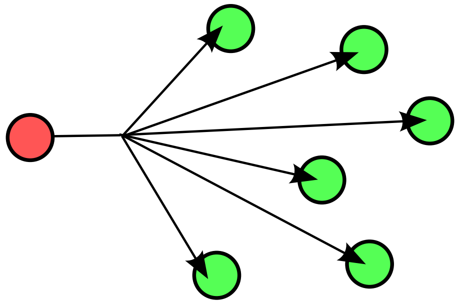
Features

Communication Primitives

Unicast



Wireless Broadcast



Parametric On

Data Structures

e.g., Names, Bits, Vectors, ADTs, Trees, ...

Logics

e.g., EUF, FOL, Equational Theory, ...

Logical Assertions

e.g., Knows a secret, Connectivity, Constraints...

Pwb Functionality

Symbolic Execution

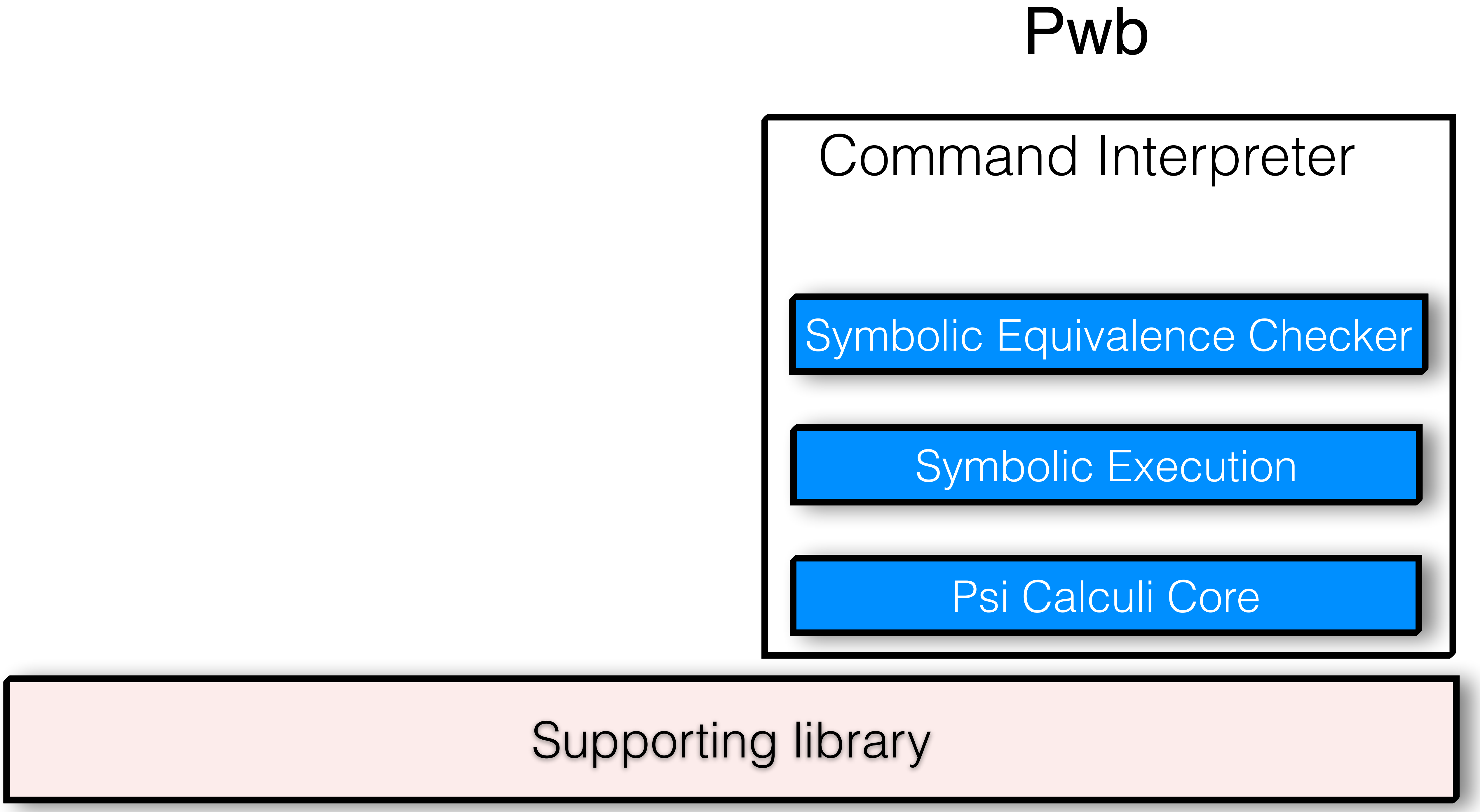
$$\Psi \rhd P \xrightarrow[C]{\alpha} P'$$

Symbolic Constraints

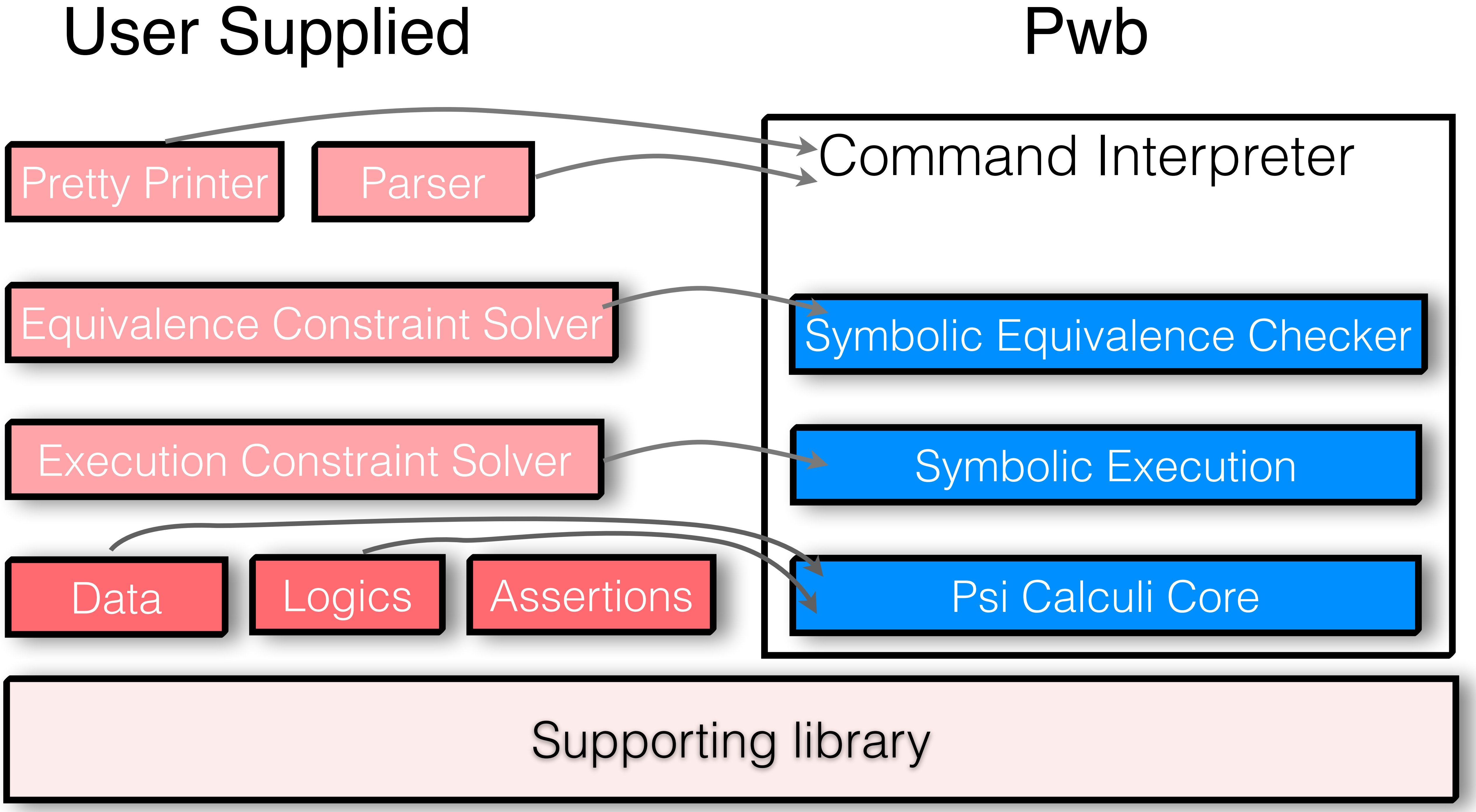
Symbolic Behavioral Equivalence Checking

$$P \sim Q$$

Parametric Architecture



Parametric Architecture



User Supplied

Pwb

Pretty Printer

Parser

Command Interpreter

Equivalence Constraint Solver

Symbolic Equivalence Checker

Execution Constraint Solver

Symbolic Execution

Data

Logics

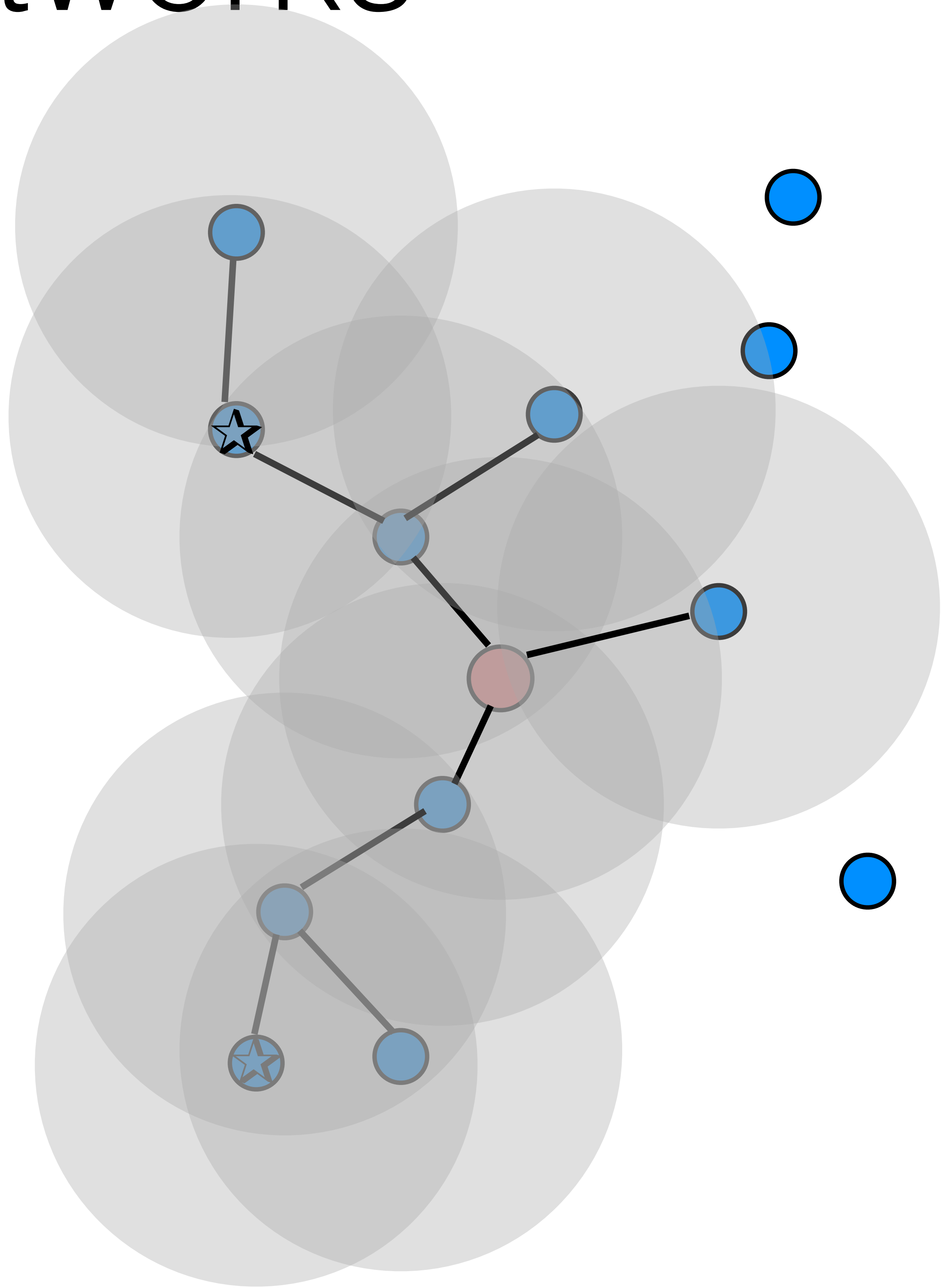
Assertions

Psi Calculi Core

Supporting library

Data Collection in Wireless Sensor Networks

1. Routing tree
2. Data collection



Specification in Pwb

Node Behavior

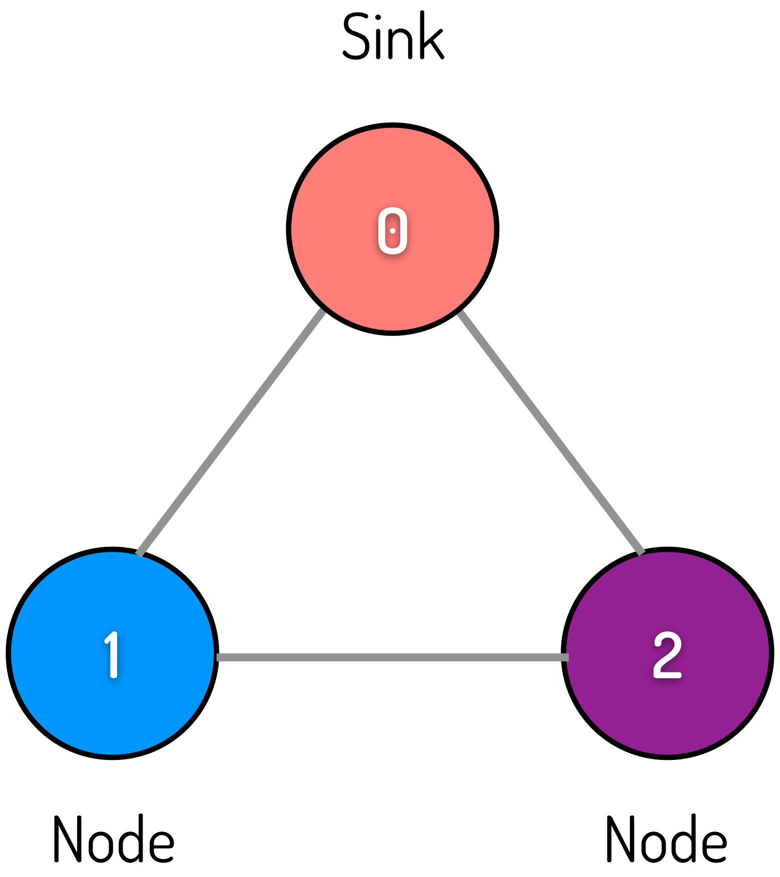
```
Sink(nodeId, sinkChan) <=
    '"init(nodeId)"! <sinkChan> .
    ! "data(sinkChan)"(x). ProcData<x> ;

Node(nodeId, nodeChan, datum) <=
    "init(nodeId)"? (chan) .
    '"init(nodeId)"! <nodeChan> .
    '"data(chan)"<datum> .
    ! "data(nodeChan)"(x).
     '"data(chan)"<x>  ;
```

System

```
(new sinkChan)  Sink<0, sinkChan>      |
(new chan1)     Node<1, chan1, datum1> |
(new chan2)     Node<2, chan2, datum2>
```

Node Connectivity for Broadcasting



graph represented as edge list

```
(0,1), (0,2), (1,2)
```

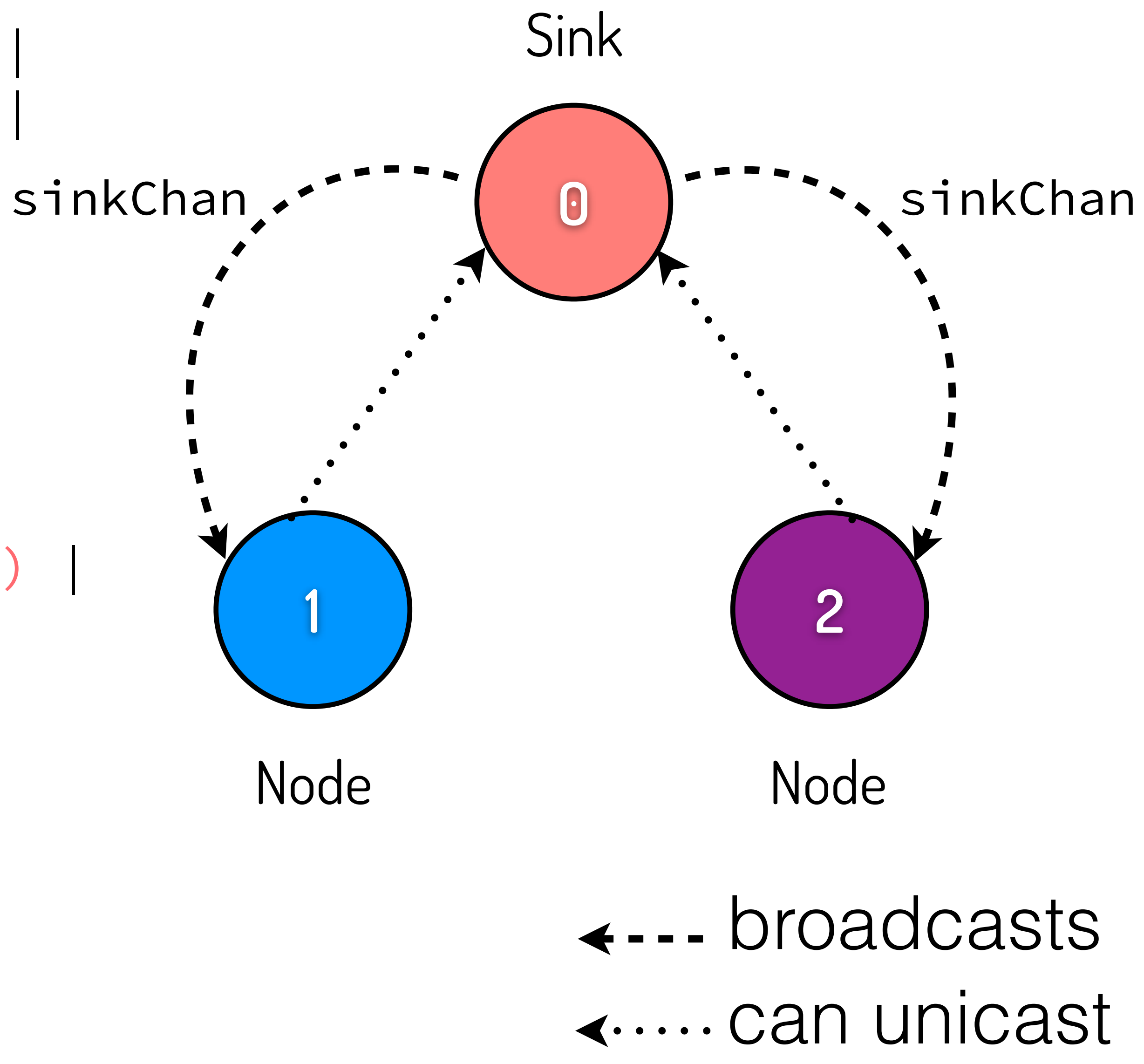
Example Transition

```
(new sinkChan)  Sink<0, sinkChan>      |
(new chan1)     Node<1, chan1, datum1> |
(new chan2)     Node<2, chan2, datum2>
```

"init(0)"!(new sinkChan)sinkChan

true

```
(!("data(sinkChan)"(gnb). ProcData<gnb>)) |
 (((new chan1)(
   '"init(1)"!<chan1>.
     '"data(sinkChan)"<datum1>.
       !("data(chan1)"(gnb).
         '"data(sinkChan)"<gnb>))) |
   ((new chan2)(
     '"init(2)"!<chan2>.
       '"data(sinkChan)"<datum2>.
         !("data(chan2)"(gnb).
           '"data(sinkChan)"<gnb>))))
```



Example Summary

- Executable model of an aggregation-tree building protocol
- Connectivity graph expressed as an assertion (possible to add and remove edges at runtime)
- Mix of wireless broadcast and reliable unicast communication

Session Types

Specification of process that checks equality over a channel of type

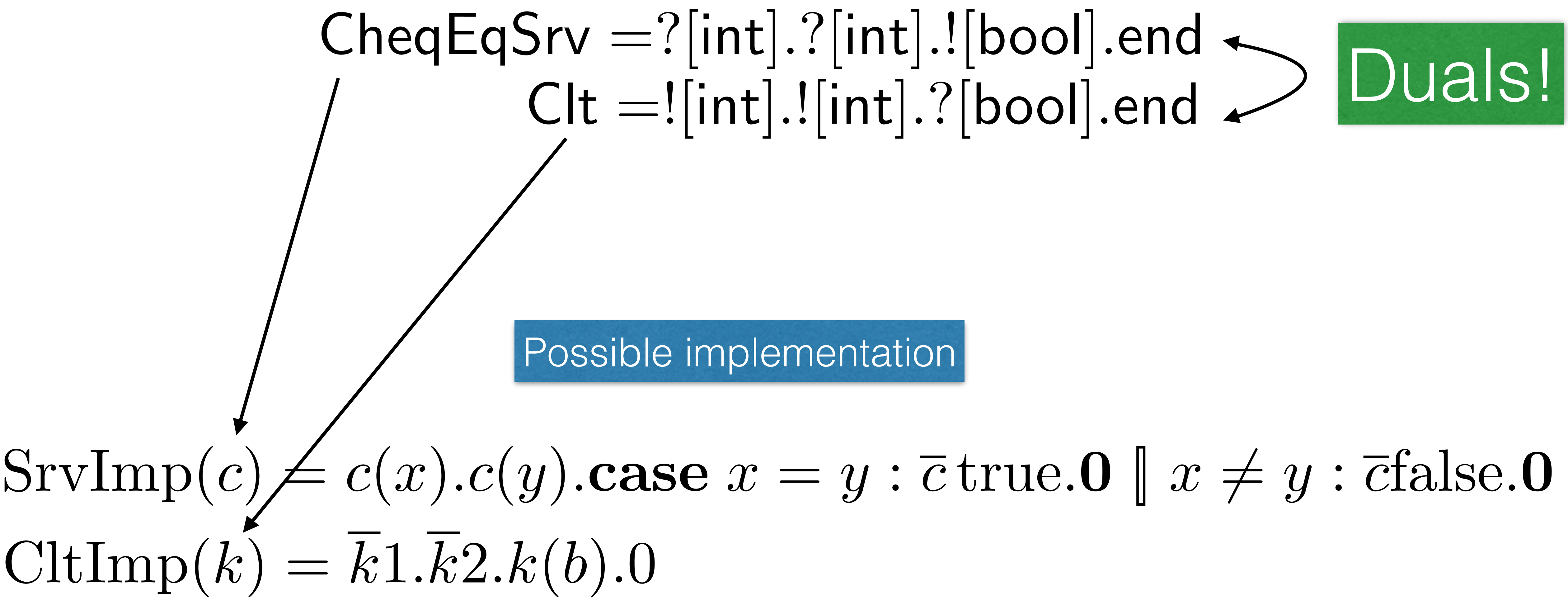
$$\mathsf{CheqEqSrv} = ?[\mathsf{int}].?[\mathsf{int}].![\mathsf{bool}].\mathsf{end}$$

Possible implementation

$$\mathrm{SrvImp}(c) = c(x).c(y).\mathbf{case}\ x = y : \overline{c}\,\mathrm{true}.\mathbf{0} \talloblong x \neq y : \overline{c}\mathrm{false}.\mathbf{0}$$

Session Types

Specification of process that checks equality over a channel of type



$$\mathsf{CheqEqSrv} = ?[\mathsf{int}].?[\mathsf{int}].![\mathsf{bool}].\mathsf{end}$$

$$\mathsf{Clt} = ![\mathsf{int}].![\mathsf{int}].?[\mathsf{bool}].\mathsf{end}$$

Duals!

Possible implementation

$$\mathrm{SrvImp}(c) = c(x).c(y).\textbf{case}\ x = y : \overline{c}\,\mathrm{true}.\mathbf{0} \parallel x \neq y : \overline{c}\mathrm{false}.\mathbf{0}$$

$$\mathrm{CltImp}(k) = \overline{k}1.\overline{k}2.k(b).0$$

Session Types

Specification of process that checks equality over a channel of type

$$\mathsf{CheqEqSrv} = ?[\mathsf{int}].?[\mathsf{int}].![\mathsf{bool}].\mathsf{end}$$
$$\mathsf{Clt} = ![\mathsf{int}].![\mathsf{int}].?[\mathsf{bool}].\mathsf{end}$$

Possible implementation

$$\mathrm{SrvImp}(c) = c(x).c(y).\mathbf{case}\ x = y : \overline{c}\,\mathrm{true}.\mathbf{0} \parallel x \neq y : \overline{c}\mathrm{false}.\mathbf{0}$$

$$\mathrm{CltImp}(k) = \overline{k}1.\overline{k}2.k(b).0$$

$$c^{+} : \mathsf{CheqEqSrv}$$
$$c^{-} : \mathsf{Clt} = \overline{\mathsf{CheqEqSrv}}$$

System

$$(\nu c)(\mathrm{SrvImp}(c^{+}) \mid \mathrm{CltImp}(c^{-}))$$

Session Types

- Structured Description of a protocol
- Specifies direction and data carried over channel
- Abstract specification
- Safety: progress, session fidelity

Broadcast Session Types

- First Application of session types to **Unreliable** and **Broadcast** communication systems
- Types for **scatter** & **gather** communication pattern

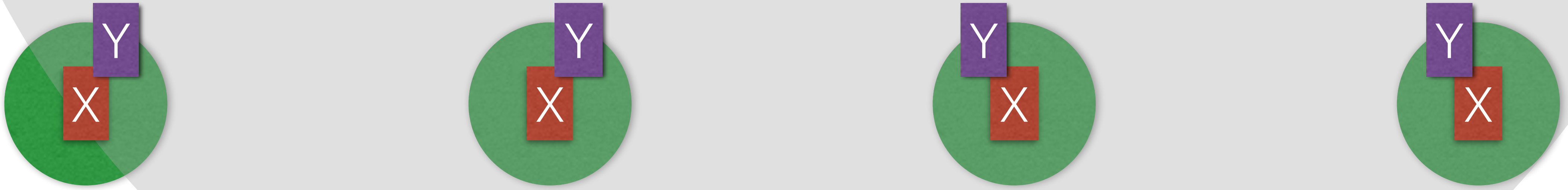
Scatter & Gather

Type

$c^+ :![\text{int}].?[\text{int}].T$

$\overline{c^+}x.c^+(y).P$

- Runtime tracking of session state
- Extended notion of duality



$c^-(x).\overline{c^-}y.Q_i$

Unreliability

Let process recover

$$P \bowtie R$$

(νc)

↓

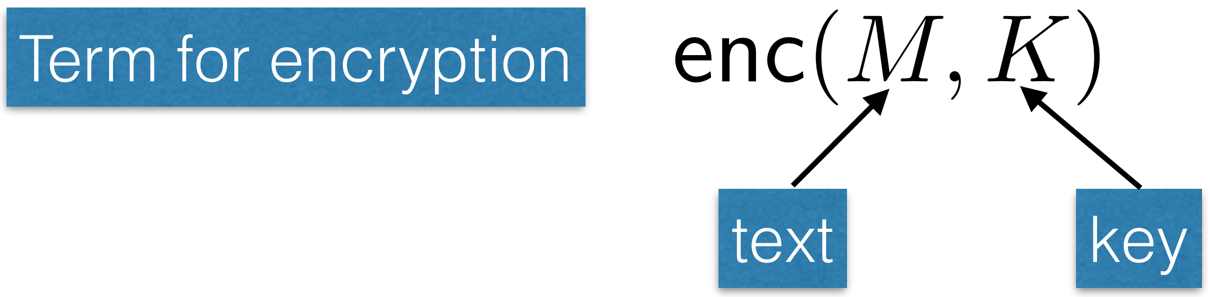
$$(\nu c)(c^{+}(x).c^{+}(y).\mathbf{0} \quad | \quad \overline{c^{-}}2.\mathbf{0})$$

Process no longer consistent with the type!

Results

- We are the first to introduce session types to unreliable and broadcast systems
- Well-typed processes always transition to well-typed processes
- Well-typed process does not reduce to an error

Crypto Example



Term for encryption

$\mathsf{enc}(M, K)$

$$(\nu k)(\overline{M}\mathsf{enc}(a, k).P) \mid M(\lambda x, y)\mathsf{enc}(x, y).Q)$$

$$\to (\nu k)(P \mid Q[x := a, y := k])$$

$$(\nu k)(\overline{M}\mathsf{enc}(a, k).P \mid M(\lambda x)\mathsf{enc}(x, k).Q)$$

We need a way to control what are pattern variables

$$\to (\nu k)(P \mid Q[x := a])$$

Knowledge of the key

Computation

All names of $\tilde{L}$

must be in $M[\tilde{x} := \tilde{L}]$ if $\tilde{x} \subseteq \mathrm{n}(M)$

Useful computation to have as part of substitution

$$\mathsf{dec}(\mathsf{enc}(M, K), K) \rightarrow M$$

However, the substitutions are **not** allowed to **lose names**

$$\mathsf{dec}(\mathsf{enc}(a, b), b)[b := k] \rightarrow a$$

k does not appear in the result

Generalised Pattern Matching

User defined pattern matchin.
Relaxes requirement on the substitution.

$\mathbf{X}$ *patterns, ranged over by X, Y*

$\underline{M}(\lambda\tilde{x})X.P$
well-formed if
$\tilde{x} \in \text{VARS}(X)$

$$\begin{array}{rcll} \text{MATCH} & : & \mathbf{T} \times \mathcal{N}^* \times \mathbf{X} \to \mathcal{P}(\mathbf{T}^*) & \textit{Pattern matching} \\ \text{VARS} & : & \mathbf{X} \to \mathcal{P}(\mathcal{P}(\mathcal{N})) & \textit{Pattern variables} \end{array}$$

Signifies which names are patterns

Ex:

$$\text{VARS}(\mathsf{enc}(m, k)) = \{\{m\}\}$$

$\underline{M}(\lambda m)\mathsf{enc}(m, k).P$

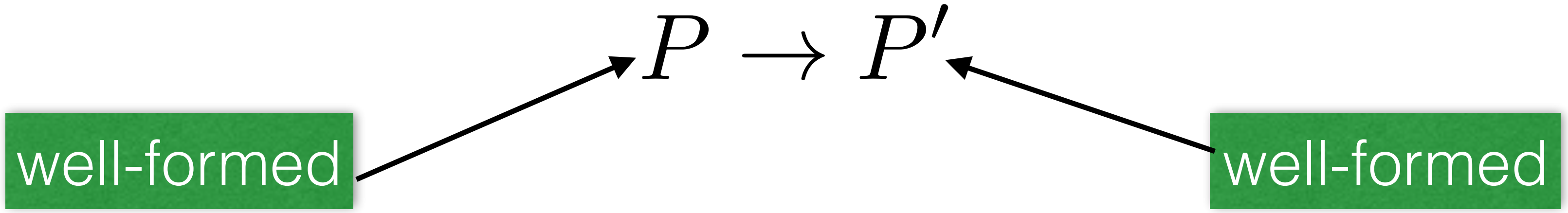
$\underline{M}(\lambda m, k)\mathsf{enc}(m, k).P$

Results

did not break psi

- Previous Psi results hold: compositional semantics, behavioural equivalence is a congruence

- well-formedness of processes is preserved by transitions



$$P \to P'$$

well-formed

well-formed

Polyadic communication

Polyadic pi-calculus

$$\begin{array}{ll} & a(x_1,\ldots,x_n).P \\ \mid & \overline{a}b_1,\ldots,b_n.Q \end{array} \longrightarrow P\{b_1,\ldots,b_n/x_1,\ldots,x_n\} \mid Q$$

Should be easy to express in Psi

Let’s take $\mathbf{T} = \mathcal{N}^*$ ⟵ sequences of names

Substitution needs to be a **total** function

$$(a, b, c)[a := (c, d)] = ((c, d), b, c) \qquad \notin \mathcal{N}^*$$

Junk

Solution

Allow $((c, d), b, c)$ $\mathbf{T} = \mathbf{T}^* \cup \mathcal{N}$

Set to error

$(a, b, c)[a := (c, d)] =$ error $\mathbf{T} = \mathcal{N}^* \cup \{$error$\}$

Allow substitution to be a **partial** function

Better yet! Type to disallow ‘bad’ substitutions from arising.

Sorts

a.k.a. Types

Goal: flexible definition of "well-formed"

$\textsc{sort} \quad : \quad \mathcal{N} \cup \mathbf{T} \cup \mathbf{X} \to \mathcal{S}$ name, term, and pattern sorting

		is well-sorted iff
substitution	$[\widetilde{a} := \widetilde{N}]$	$\textsc{sort}(a_i) \prec \textsc{sort}(N_i)$
restriction	$(\nu a)P$	$\textsc{sort}(a) \in \mathcal{S}_\nu$
output	$\overline{M}\ \ N.P$	$\textsc{sort}(M)\ \overline{\propto}\ \textsc{sort}(N)$
input	$\underline{M}(\lambda\widetilde{x})X.P$	$\textsc{sort}(M)\ \underline{\propto}\ \textsc{sort}(X)$

Polyadic Pi-calculus

$$\begin{aligned} \textsc{sort}(a) &= \texttt{chan} \\ \textsc{sort}(\tilde{a}) &= \texttt{tup} \\ \overline{\propto} = \underline{\propto} &= \{(\texttt{chan}, \texttt{tup})\} \end{aligned}$$

a channel can send/ receive a tuple

$$\textsc{vars}(\langle \tilde{a} \rangle) = \{\tilde{a}\}$$

all names in input pattern must be bound

$$\underline{a}(\lambda \tilde{x})\langle \tilde{x} \rangle.P$$

$$\textsc{match}(\langle \tilde{a} \rangle, \tilde{x}, \langle \tilde{x} \rangle) = \{\tilde{a}\} \text{ if } |\tilde{a}| = |\tilde{x}|$$

$\langle \tilde{a} \rangle$matches the pattern $\langle \tilde{x} \rangle$ binding $\tilde{x}$, then substituting $\tilde{a}$ for $\tilde{x}$

$$\underline{c}(\lambda \tilde{x})\langle \tilde{x} \rangle.P \xrightarrow{\underline{c}\ \tilde{a}} P[\tilde{x} := \tilde{a}]$$

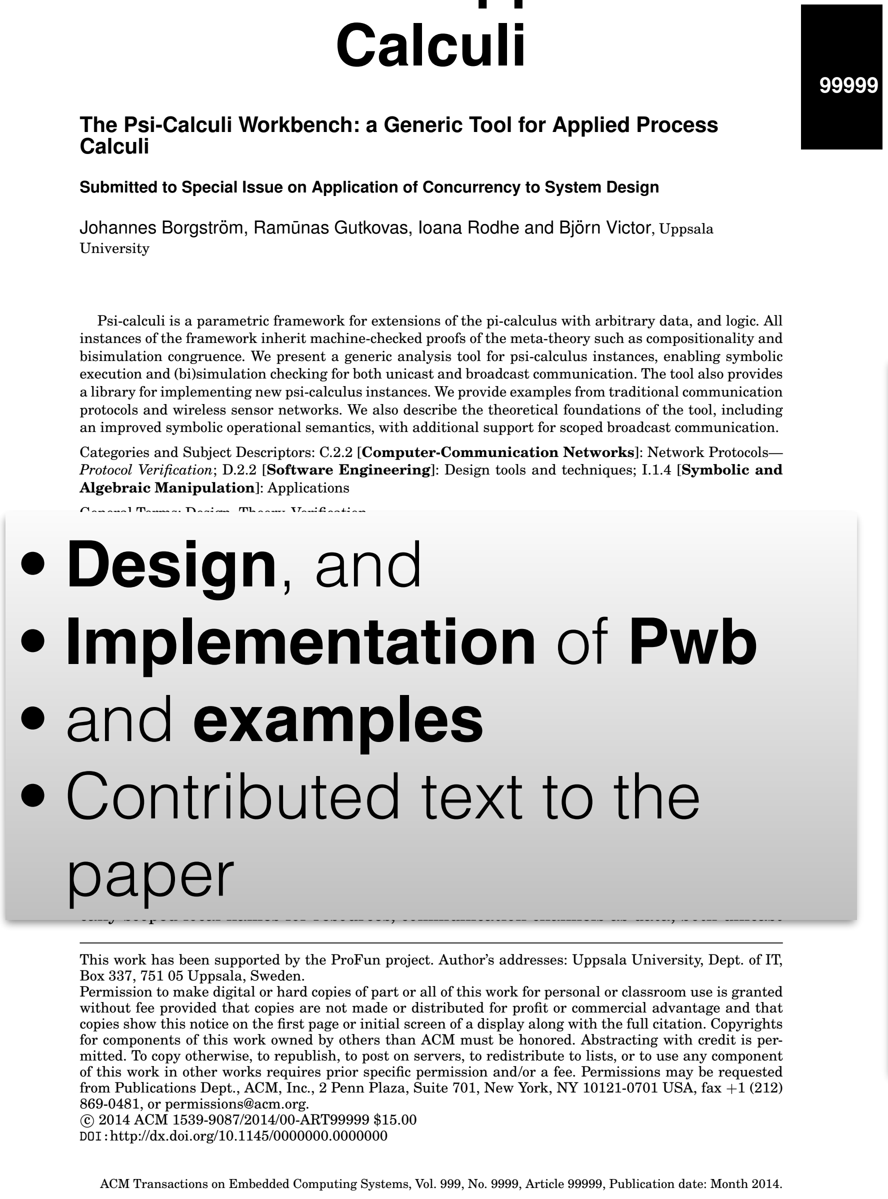
Formal correspondence of transitions and equivalence

Results

- More **expressive** framework
- Captures many previous process calculi
- Better precision for defining terms
- Well-sortedness is preserved by transitions
- Previous results for psi still hold
- Implemented in Pwb

Personal Contributions

The Psi-Calculi Workbench: a Generic Tool for Applied Process Calculi



99999

The Psi-Calculi Workbench: a Generic Tool for Applied Process Calculi

Submitted to Special Issue on Application of Concurrency to System Design

Johannes Borgström, Ramūnas Gutkovas, Ioana Rodhe and Björn Victor, Uppsala University

Psi-calculi is a parametric framework for extensions of the pi-calculus with arbitrary data, and logic. All instances of the framework inherit machine-checked proofs of the meta-theory such as compositionality and bisimulation congruence. We present a generic analysis tool for psi-calculus instances, enabling symbolic execution and (bi)simulation checking for both unicast and broadcast communication. The tool also provides a library for implementing new psi-calculus instances. We provide examples from traditional communication protocols and wireless sensor networks. We also describe the theoretical foundations of the tool, including an improved symbolic operational semantics, with additional support for scoped broadcast communication.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Protocol Verification*; D.2.2 [**Software Engineering**]: Design tools and techniques; I.1.4 [**Symbolic and Algebraic Manipulation**]: Applications

General Terms: Design, Theory, Verification

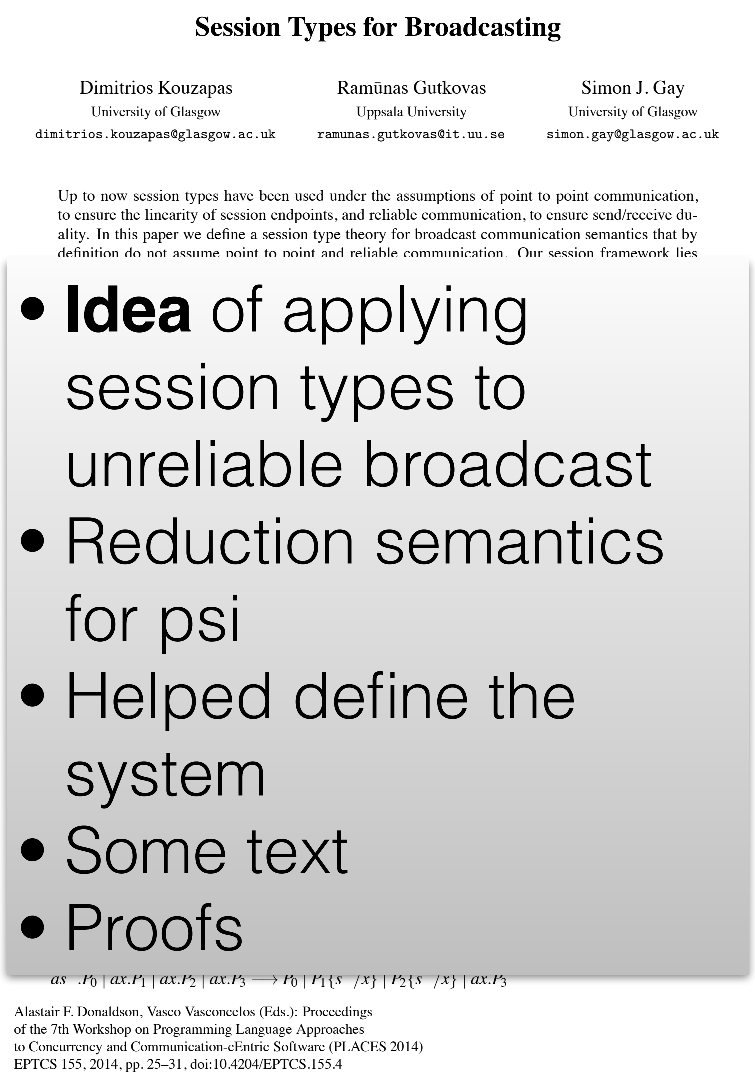
- **Design**, and
- **Implementation** of **Pwb**
- and **examples**
- Contributed text to the paper

This work has been supported by the ProFun project. Author's addresses: Uppsala University, Dept. of IT, Box 337, 751 05 Uppsala, Sweden.

DOI:http://dx.doi.org/10.1145/0000000.0000000

ACM Transactions on Embedded Computing Systems, Vol. 999, No. 9999, Article 99999, Publication date: Month 2014.

Session Types for Broadcasting



Session Types for Broadcasting

Dimitrios Kouzapas
University of Glasgow
dimitrios.kouzapas@glasgow.ac.uk

Ramūnas Gutkovas
Uppsala University
ramunas.gutkovas@it.uu.se

Simon J. Gay
University of Glasgow
simon.gay@glasgow.ac.uk

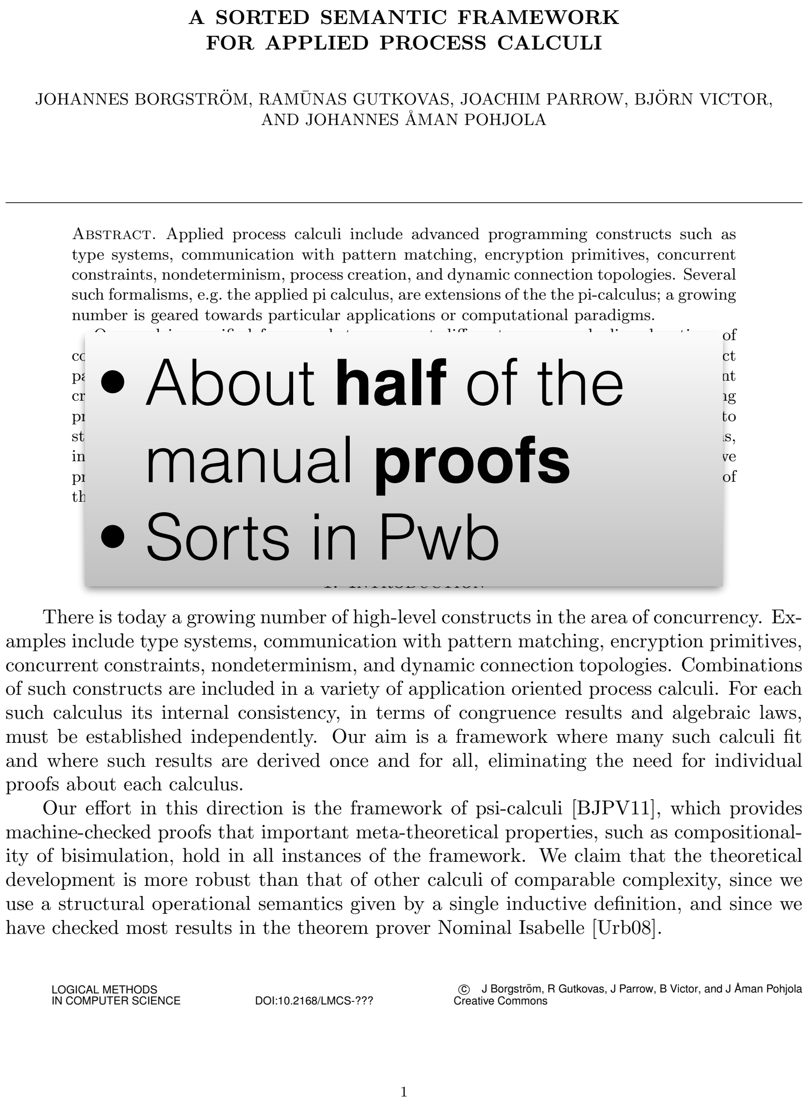
Up to now session types have been used under the assumptions of point to point communication, to ensure the linearity of session endpoints, and reliable communication, to ensure send/receive duality. In this paper we define a session type theory for broadcast communication semantics that by definition do not assume point to point and reliable communication. Our session framework lies

- **Idea** of applying session types to unreliable broadcast
- Reduction semantics for psi
- Helped define the system
- Some text
- Proofs

$$a\overline{s}.P_0 \mid ax.P_1 \mid ax.P_2 \mid ax.P_3 \longrightarrow P_0 \mid P_1\{s/x\} \mid P_2\{s/x\} \mid ax.P_3$$

Alastair F. Donaldson, Vasco Vasconcelos (Eds.): Proceedings
of the 7th Workshop on Programming Language Approaches
to Concurrency and Communication-cEntric Software (PLACES 2014)
EPTCS 155, 2014, pp. 25–31, doi:10.4204/EPTCS.155.4

A Sorted Semantic Framework for Applied Process Calculi



A SORTED SEMANTIC FRAMEWORK
FOR APPLIED PROCESS CALCULI

JOHANNES BORGSTRÖM, RAMŪNAS GUTKOVAS, JOACHIM PARROW, BJÖRN VICTOR,
AND JOHANNES ÅMAN POHJOLA

Abstract. Applied process calculi include advanced programming constructs such as type systems, communication with pattern matching, encryption primitives, concurrent constraints, nondeterminism, process creation, and dynamic connection topologies. Several such formalisms, e.g. the applied pi calculus, are extensions of the the pi-calculus; a growing number is geared towards particular applications or computational paradigms.

- About **half** of the manual **proofs**
- Sorts in Pwb

There is today a growing number of high-level constructs in the area of concurrency. Examples include type systems, communication with pattern matching, encryption primitives, concurrent constraints, nondeterminism, and dynamic connection topologies. Combinations of such constructs are included in a variety of application oriented process calculi. For each such calculus its internal consistency, in terms of congruence results and algebraic laws, must be established independently. Our aim is a framework where many such calculi fit and where such results are derived once and for all, eliminating the need for individual proofs about each calculus.

Our effort in this direction is the framework of psi-calculi [BJPV11], which provides machine-checked proofs that important meta-theoretical properties, such as compositionality of bisimulation, hold in all instances of the framework. We claim that the theoretical development is more robust than that of other calculi of comparable complexity, since we use a structural operational semantics given by a single inductive definition, and since we have checked most results in the theorem prover Nominal Isabelle [Urb08].

LOGICAL METHODS
IN COMPUTER SCIENCE
DOI:10.2168/LMCS-???

1

Conclusion

Made it possible to model more complicated systems

- A parametric ication tool the Psi-Calculi bench

Straightforward protocol specification with safety guarantees

- Session nication and unreliable systems
- More expressivity: generalised pattern-matching and sorts

Made psi-calculi more expressive

Future Work

Build more complex models out of simpler

- Algebras of Psi-calculi

Factory of tool factory

- Nominal transition system specification

Fine grained reasoning: safety & liveness

- Modal logics for Psi

Efficient representations

- Models of Psi-calculi

More WSNs

- More case-studies

Thank you for listening