Nondeterministic Asynchronous Dataflow in Isabelle/HOL

Rafael Castro G. Silva, Laouen Fernet and, Dmitriy Traytel

Department of Computer Science
University of Copenhagen

14/05/2025

Motivation

Motivation

Context:

- Stream Processing: programs that compute (possibly) unbounded sequences of data (streams)
- Frameworks: Google Cloud Dataflow, Apache Flink, and Timely Dataflow

Motivation

Context:

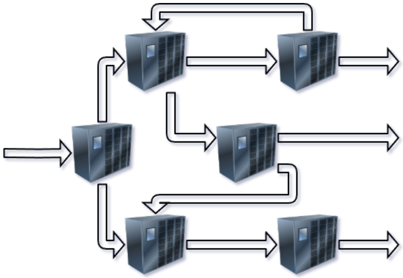
- Stream Processing: programs that compute (possibly) unbounded sequences of data (streams)
- Frameworks: Google Cloud Dataflow, Apache Flink, and Timely Dataflow

Our long term goal:

Mechanically Verify Timely Dataflow algorithms

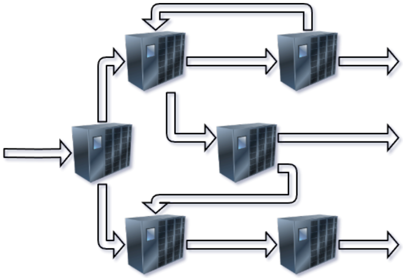
A Good Foundation

A Good Foundation



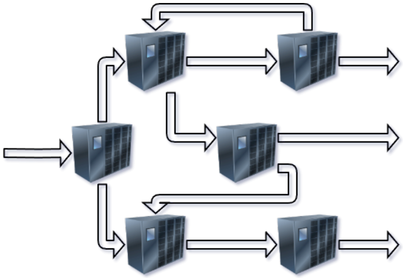
- Nondeterministic Asynchronous Dataflow

A Good Foundation



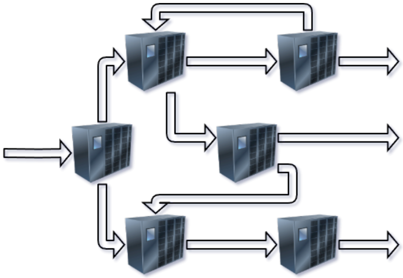
- Nondeterministic Asynchronous Dataflow
  - Dataflow: Directed graph of interconnected operators

A Good Foundation



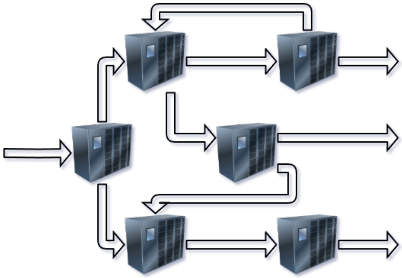
- Nondeterministic Asynchronous Dataflow
  - Dataflow: Directed graph of interconnected operators
  - Asynchronous:
    - Operators execute independently: processes without an orchestrator
    - Operators can freely communicate with the network (read/write); do silent computation steps
    - Networks are unbounded FIFO queues

A Good Foundation



- Nondeterministic Asynchronous Dataflow
  - Dataflow: Directed graph of interconnected operators
  - Asynchronous:
    - Operators execute independently: processes without an orchestrator
    - Operators can freely communicate with the network (read/write); do silent computation steps
    - Networks are unbounded FIFO queues
  - Nondeterministic:
    - Operators can make nondeterministic choices

A Good Foundation



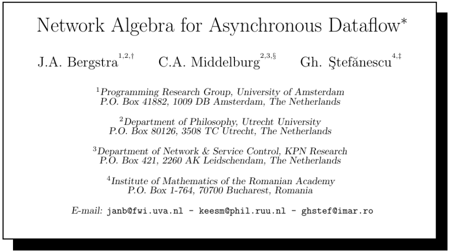
- Nondeterministic Asynchronous Dataflow
  - Dataflow: Directed graph of interconnected operators
  - Asynchronous:
    - Operators execute independently: processes without an orchestrator
    - Operators can freely communicate with the network (read/write); do silent computation steps
    - Networks are unbounded FIFO queues
  - Nondeterministic:
    - Operators can make nondeterministic choices

Question

How do we know something is Nondeterministic Asynchronous Dataflow?

The Algebra for Nondeterministic Asynchronous Dataflow

- Bergstra et al. presents an algebra for Nondeterministic Asynchronous Dataflow
- Primitives:
  sequential and parallel composition; feedback loop...
- 52 axioms
- A process calculus instance



Network Algebra for Asynchronous Dataflow*

J.A. Bergstra[1,2,†] C.A. Middelburg[2,3,§] Gh. Ştefănescu[4,‡]

[1]*Programming Research Group, University of Amsterdam*
P.O. Box 41882, 1009 DB Amsterdam, The Netherlands

[2]*Department of Philosophy, Utrecht University*
P.O. Box 80126, 3508 TC Utrecht, The Netherlands

[3]*Department of Network & Service Control, KPN Research*
P.O. Box 421, 2260 AK Leidschendam, The Netherlands

[4]*Institute of Mathematics of the Romanian Academy*
P.O. Box 1-764, 70700 Bucharest, Romania

E-mail: janb@fwi.uva.nl - keesm@phil.ruu.nl - ghstef@imar.ro

Main Contributions

- An Isabelle/HOL instance of Nondeterministic Asynchronous Dataflow
    - Operators as a shallow embedding as codatatypes
    - 51 axioms proved

Isabelle/HOL Preliminaries

What is Isabelle/HOL?

What is Isabelle/HOL?

- Isabelle: A generic proof assistant

- Classical higher-order logic (HOL):
  Simple Typed Lambda Calculus + axiom of choice + axiom of infinity + rank-1 polymorphism

What is Isabelle/HOL?

- Isabelle: A generic proof assistant

- Classical higher-order logic (HOL):
  Simple Typed Lambda Calculus + axiom of choice + axiom of infinity + rank-1 polymorphism
- Isabelle/HOL: Isabelle's flavor of HOL

What is Isabelle/HOL?

- Isabelle: A generic proof assistant

- Classical higher-order logic (HOL):
  Simple Typed Lambda Calculus + axiom of choice + axiom of infinity + rank-1 polymorphism
- Isabelle/HOL: Isabelle's flavor of HOL

Why Isabelle/HOL?

- Codatatypes: (possibly) infinite data structures (e.g., lazy lists, streams)
- Corecursion: always eventually produces some codatatype constructor
- Coinductive predicates: infinite number of introduction rule applications
- Coinduction: reason about coinductive predicates

Operators as a Codatatype

Operators

Operators in Isabelle/HOL

$$
\begin{array}{l}
\textbf{codatatype}\ ('i,'o,'d)\ op = \\
\quad \mathrm{Read}\ 'i\ ('d \Rightarrow ('i,'o,'d)\ op) \mid \mathrm{Write}\ (('i,'o,'d)\ op)\ 'o\ 'd \\
\quad \mathrm{Silent}\ ('i,'o,'d)\ op \mid \mathrm{Choice}\ (('i,'o,'d)\ op)\ cset
\end{array}
$$

Operators

Operators in Isabelle/HOL

$$
\begin{array}{l}
\textbf{codatatype}\ ('i,'o,'d)\ \mathit{op} = \\
\quad \mathsf{Read}\ 'i\ ('d \Rightarrow ('i,'o,'d)\ \mathit{op}) \mid \mathsf{Write}\ (('i,'o,'d)\ \mathit{op})\ 'o\ 'd \\
\quad \mathsf{Silent}\ ('i,'o,'d)\ \mathit{op} \mid \mathsf{Choice}\ (('i,'o,'d)\ \mathit{op})\ \mathit{cset}
\end{array}
$$

- Type parameters:
  inputs/output ports; data
- Operator's actions
- Possibly infinite trees

Operators

Operators in Isabelle/HOL

$$\textbf{codatatype}\ ('i,'o,'d)\ op =$$
$$\mathsf{Read}\ 'i\ ('d \Rightarrow ('i,'o,'d)\ op) \mid \mathsf{Write}\ (('i,'o,'d)\ op)\ 'o\ 'd$$
$$\mathsf{Silent}\ ('i,'o,'d)\ op \mid \mathsf{Choice}\ (('i,'o,'d)\ op)\ cset$$

- Type parameters: inputs/output ports; data
- Operator's actions
- Possibly infinite trees

Uncommunicative operators

$$\oslash = \mathsf{Choice}\ \{\}_c$$

$$\odot = \mathsf{Silent}\ \odot$$

$$\otimes = \mathsf{Choice}\ \{\otimes\}_c$$

Operators

Operators in Isabelle/HOL

$$\textbf{codatatype}\ ('i,'o,'d)\ op = $$
$$\quad \mathsf{Read}\ 'i\ ('d \Rightarrow ('i,'o,'d)\ op) \mid \mathsf{Write}\ (('i,'o,'d)\ op)\ 'o\ 'd$$
$$\quad \mathsf{Silent}\ ('i,'o,'d)\ op \mid \mathsf{Choice}\ (('i,'o,'d)\ op)\ cset$$

- Type parameters: inputs/output ports; data
- Operator's actions
- Possibly infinite trees

Uncommunicative operators

$\oslash = \mathsf{Choice}\ \{\}_c$

$\odot = \mathsf{Silent}\ \odot$

$\otimes = \mathsf{Choice}\ \{\otimes\}_c$

More examples

$\mathsf{ex1} = \mathsf{Choice}\ \{\mathsf{Write}\ \mathsf{ex1}\ 1\ 42, \oslash\}_c$

$\mathsf{ex2} = \mathsf{Choice}\ \{\mathsf{Write}\ \mathsf{ex2}\ 1\ 42, \mathsf{ex2}\}_c$

$\mathsf{ex3} = \mathsf{Choice}\ \{\mathsf{Write}\ \mathsf{ex3}\ 1\ 42, \mathsf{Silent}\ \mathsf{ex3}\}_c$

Operators Equivalences: Weak Bisimilarity

An ideal equivalence relation

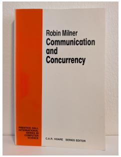
- Only equate operators that **behave** the same
- Useful reasoning principle

Operators Equivalences: Weak Bisimilarity

An ideal equivalence relation

- Only equate operators that **behave** the same
- Useful reasoning principle

- Milner's approach!
  The classic chapter on **weak bisimilarity**

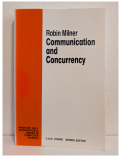


- Based on labeled transition systems (LTS)
- **Weak**: silent computations are abstracted away

Operators Equivalences: Weak Bisimilarity

An ideal equivalence relation

- Only equate operators that **behave** the same
- Useful reasoning principle



- Milner's approach!
  The classic chapter on **weak bisimilarity**
- Based on labeled transition systems (LTS)
- **Weak**: silent computations are abstracted away

Weak bisimilarity of operators

- Labels (actions): read, write, silent (τ)
- Weak bisimilarity of operators ($\approx$): LTS + weak simulation + greatest weak bisimulation
- $\approx$ has a useful coinduction principle
- $\oslash \approx \otimes \approx \otimes$ and ex1 $\approx$ ex2 $\approx$ ex3

Asynchronous Dataflow Operators

Auxiliary Definitions

Auxiliary Definitions

- Buffers: $'p \Rightarrow 'd\ list$

Buffer functions

$\mathrm{BHD}\ p\ buf = \mathrm{hd}\ (buf\ \ p)$

$\mathrm{BTL}\ p\ buf = buf(p := \mathrm{tl}\ (buf\ \ p))$

$\mathrm{BENQ}\ p\ x\ buf = buf(p := buf\ \ p\ @\ [x])$

Auxiliary Definitions

- Buffers: $'p \Rightarrow 'd\ \mathit{list}$
- choices: computes the set of operators that immediately perform next actions

Buffer functions

$\mathrm{BHD}\ p\ \mathit{buf}\ = \mathrm{hd}\ (\mathit{buf}\ \ p)$

$\mathrm{BTL}\ p\ \mathit{buf}\ = \mathit{buf}(p := \mathrm{tl}\ (\mathit{buf}\ \ p))$

$\mathrm{BENQ}\ p\ x\ \mathit{buf}\ = \mathit{buf}(p := \mathit{buf}\ \ p\ @\ [x])$

choices type

$\mathrm{choices} :: ('i, 'o, 'd)\ \mathit{op} \Rightarrow (('i, 'o, 'd)\ \mathit{op})\ \mathit{set}$

Auxiliary Definitions

- Buffers: $'p \Rightarrow 'd\ list$
- choices: computes the set of operators that immediately perform next actions

Buffer functions

$\mathrm{BHD}\ p\ buf\ = \mathrm{hd}\ (buf\ \ p)$

$\mathrm{BTL}\ p\ buf\ = buf(p := \mathrm{tl}\ (buf\ \ p))$

$\mathrm{BENQ}\ p\ x\ buf\ = buf(p := buf\ \ p\ @\ [x])$

choices type

$\mathrm{choices} :: ('i, 'o, 'd)\ op \Rightarrow (('i, 'o, 'd)\ op)\ set$

Mapping ports functions

$\mathrm{map\_op} :: ('i_1 \Rightarrow 'i_2) \Rightarrow ('o_1 \Rightarrow 'o_2) \Rightarrow ('i_1, 'o_1, 'd)\ op \Rightarrow ('i_2, 'o_2, 'd)\ op$

$\curvearrowright :: ('a + 'b) + 'c \Rightarrow 'a + 'b + 'c$

$\curvearrowleft :: 'a + 'b + 'c \Rightarrow ('a + 'b) + 'c$

Identity

Equation of the identity operator

$$\text{id\_op}\ buf = \text{Choice}$$

$$\cup_c$$

Identity

Equation of the identity operator

$$\text{id\_op}\ \mathit{buf} = \text{Choice}$$
$$(((\lambda p.\ \text{Read}\ p\ (\lambda x.\ \text{id\_op}\ (\text{BENQ}\ p\ x\ \mathit{buf})))\ \grave{}_c\ \mathfrak{U}_c)$$
$$\cup_c$$

- $\mathfrak{U}_c$ is the set of usable ports provide by its type

Identity

Equation of the identity operator

$$\text{id\_op}\ \mathit{buf} = \text{Choice}$$
$$(((\lambda p.\ \text{Read}\ p\ (\lambda x.\ \text{id\_op}\ (\text{BENQ}\ p\ x\ \mathit{buf})))\ \grave{}_c\ \mathfrak{U}_c)$$
$$\cup_c$$
$$((\lambda p.\ \text{Write}\ (\text{id\_op}\ (\text{BTL}\ p\ \mathit{buf}))\ p\ (\text{BHD}\ p\ \mathit{buf}))\ \grave{}_c\ \{p \in_c \mathfrak{U}_c \mid \mathit{buf}\ \ p \neq []\}))$$

- $\mathfrak{U}_c$ is the set of usable ports provide by its type

Identity

Equation of the identity operator

$$\mathrm{id\_op}\ \mathit{buf} = \mathrm{Choice}$$
$$(((\lambda p.\ \mathrm{Read}\ p\ (\lambda x.\ \mathrm{id\_op}\ (\mathrm{BENQ}\ p\ x\ \mathit{buf})))\ `_c\ \mathfrak{U}_c)$$
$$\cup_c$$
$$((\lambda p.\ \mathrm{Write}\ (\mathrm{id\_op}\ (\mathrm{BTL}\ p\ \mathit{buf}))\ p\ (\mathrm{BHD}\ p\ \mathit{buf}))\ `_c\ \{p \in_c \mathfrak{U}_c \mid \mathit{buf}\ p \neq []\}))$$

- $\mathfrak{U}_c$ is the set of usable ports provide by its type
- Stream delayer: always can read, and it may eventually output

Identity

Equation of the identity operator

$$\text{id\_op } \mathit{buf} = \text{Choice}$$
$$(((\lambda p.\ \text{Read } p\ (\lambda x.\ \text{id\_op } (\text{BENQ } p\ x\ \mathit{buf})))\ `_c\ \mathfrak{U}_c)$$
$$\cup_c$$
$$((\lambda p.\ \text{Write } (\text{id\_op } (\text{BTL } p\ \mathit{buf}))\ p\ (\text{BHD } p\ \mathit{buf}))\ `_c\ \{p \in_c \mathfrak{U}_c \mid \mathit{buf}\ \ p \neq []\}))$$

- $\mathfrak{U}_c$ is the set of usable ports provide by its type
- Stream delayer: always can read, and it may eventually output

Identity operator with an empty buffer

$$\mathcal{I} = \text{id\_op } (\lambda\_.\ [])$$

Composition: Preliminaries

Composition operator type

$$\text{comp\_op} :: ('o_1 \Rightarrow 'i_2\ \mathit{option}) \Rightarrow ('i_2 \Rightarrow 'd\ \mathit{list}) \Rightarrow ('i_1, 'o_1, 'd)\ \mathit{op} \Rightarrow ('i_2, 'o_2, 'd)\ \mathit{op} \Rightarrow ('i_1 + 'i_2, 'o_1 + 'o_2, 'd)\ \mathit{op}$$

Composition: Preliminaries

Composition operator type

$$\text{comp\_op} :: ('o_1 \Rightarrow 'i_2\ \mathit{option}) \Rightarrow ('i_2 \Rightarrow 'd\ \mathit{list}) \Rightarrow ('i_1, 'o_1, 'd)\ \mathit{op} \Rightarrow ('i_2, 'o_2, 'd)\ \mathit{op} \Rightarrow ('i_1 + 'i_2, 'o_1 + 'o_2, 'd)\ \mathit{op}$$

Sequential and parallel composition

$$op_1 \parallel op_2 = \text{comp\_op}\ (\lambda\_.\ \text{None})\ (\lambda\_.\ [])\ op_1\ op_2$$

$$op_1 \bullet op_2 = \text{map\_op projl projr}\ (\text{comp\_op Some}\ (\lambda\_.\ [])\ op_1\ op_2)$$

Composition: Equation

Equation of the composition operator

$$\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ \mathit{op}_2 =$$

Composition: Equation

Equation of the composition operator

$\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ \mathit{op}_2 = \text{Choice}$

$\cup_c$

Composition: Equation

Equation of the composition operator

comp_op *wire* *buf* op_1 op_2 = Choice

$`_c$ choices op_1) $\cup_c$

Composition: Equation

Equation of the composition operator

comp_op *wire buf* op_1 op_2 = Choice ((($\lambda op.$ <u>case</u> *op* <u>of</u>

Read *p f* $\Rightarrow$ Read (Inl *p*) ($\lambda x.$ comp_op *wire buf* (*f x*) op_2)

$`_c$ choices op_1) $\cup_c$

Composition: Equation

Equation of the composition operator

$$
\begin{array}{l}
\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ \mathit{op}_2 = \text{Choice}\ (((\lambda \mathit{op}.\ \underline{\text{case}}\ \mathit{op}\ \underline{\text{of}} \\
\quad \text{Read}\ p\ f\ \ \Rightarrow \text{Read}\ (\text{Inl}\ p)\ (\lambda x.\ \text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ (f\ x)\ \mathit{op}_2) \\
\mid\ \text{Write}\ \mathit{op}\ p\ x \Rightarrow (\underline{\text{case}}\ \mathit{wire}\ p\ \underline{\text{of}} \\
\qquad \text{None} \Rightarrow \text{Write}\ (\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}\ \mathit{op}_2)\ (\text{Inl}\ p)\ x \\
\quad\ \mid\ \text{Some}\ q \Rightarrow \text{Silent}\ (\text{comp\_op}\ \mathit{wire}\ (\text{BENQ}\ q\ x\ \mathit{buf})\ \mathit{op}\ \mathit{op}_2)) \\
\hfill \grave{}_c\ \text{choices}\ \mathit{op}_1)\ \cup_c
\end{array}
$$

Composition: Equation

Equation of the composition operator

$$
\begin{array}{l}
\text{comp\_op } \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ \mathit{op}_2 = \text{Choice } (((\lambda \mathit{op}.\ \underline{\text{case}}\ \mathit{op}\ \underline{\text{of}} \\
\quad \text{Read } p\ f \ \Rightarrow \text{Read } (\text{Inl } p)\ (\lambda x.\ \text{comp\_op } \mathit{wire}\ \mathit{buf}\ (f\ x)\ \mathit{op}_2) \\
\mid \text{Write } \mathit{op}\ p\ x \Rightarrow (\underline{\text{case}}\ \mathit{wire}\ p\ \underline{\text{of}} \\
\qquad \text{None} \Rightarrow \text{Write } (\text{comp\_op } \mathit{wire}\ \mathit{buf}\ \mathit{op}\ \mathit{op}_2)\ (\text{Inl } p)\ x \\
\quad \mid \text{Some } q \Rightarrow \text{Silent } (\text{comp\_op } \mathit{wire}\ (\text{BENQ } q\ x\ \mathit{buf})\ \mathit{op}\ \mathit{op}_2)) \\
\mid \text{Silent } \mathit{op} \Rightarrow \text{Silent } (\text{comp\_op } \mathit{wire}\ \mathit{buf}\ \mathit{op}\ \mathit{op}_2))\ \grave{}_c\ \text{choices } \mathit{op}_1)\ \cup_c
\end{array}
$$

Composition: Equation

Equation of the composition operator

$$
\begin{array}{l}
\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ \mathit{op}_2 = \text{Choice}\ (((\lambda \mathit{op}.\ \underline{\text{case}}\ \mathit{op}\ \underline{\text{of}} \\
\quad \text{Read}\ p\ f\ \Rightarrow \text{Read}\ (\text{Inl}\ p)\ (\lambda x.\ \text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ (f\ x)\ \mathit{op}_2) \\
\ \mid \text{Write}\ \mathit{op}\ p\ x \Rightarrow (\underline{\text{case}}\ \mathit{wire}\ p\ \underline{\text{of}} \\
\qquad \text{None} \Rightarrow \text{Write}\ (\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}\ \mathit{op}_2)\ (\text{Inl}\ p)\ x \\
\quad\ \mid \text{Some}\ q \Rightarrow \text{Silent}\ (\text{comp\_op}\ \mathit{wire}\ (\text{BENQ}\ q\ x\ \mathit{buf})\ \mathit{op}\ \mathit{op}_2)) \\
\ \mid \text{Silent}\ \mathit{op} \Rightarrow \text{Silent}\ (\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}\ \mathit{op}_2))\ \grave{}_c\ \text{choices}\ \mathit{op}_1)\ \cup_c \\
\hfill (\text{choices}\ \mathit{op}_2)))
\end{array}
$$

Composition: Equation

Equation of the composition operator

$$
\begin{array}{l}
\text{comp\_op } \mathit{wire}\ \mathit{buf}\ op_1\ op_2 = \text{Choice } (((\lambda op.\ \underline{\text{case}}\ op\ \underline{\text{of}} \\
\quad \text{Read } p\ f \ \Rightarrow \text{Read } (\text{Inl } p)\ (\lambda x.\ \text{comp\_op } \mathit{wire}\ \mathit{buf}\ (f\ x)\ op_2) \\
\mid \text{Write } op\ p\ x \Rightarrow (\underline{\text{case}}\ \mathit{wire}\ p\ \underline{\text{of}} \\
\qquad \text{None} \Rightarrow \text{Write } (\text{comp\_op } \mathit{wire}\ \mathit{buf}\ op\ op_2)\ (\text{Inl } p)\ x \\
\quad \mid \text{Some } q \Rightarrow \text{Silent } (\text{comp\_op } \mathit{wire}\ (\text{BENQ } q\ x\ \mathit{buf})\ op\ op_2)) \\
\mid \text{Silent } op \Rightarrow \text{Silent } (\text{comp\_op } \mathit{wire}\ \mathit{buf}\ op\ op_2))\ \grave{}_c\ \text{choices } op_1)\ \cup_c \\
\hspace{20em} \grave{}_c\ \text{sound\_reads } \mathit{wire}\ \mathit{buf}\ (\text{choices } op_2)))
\end{array}
$$

Composition: Equation

Equation of the composition operator

$$
\begin{aligned}
&\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ op_1\ op_2 = \text{Choice}\ (((\lambda op.\ \underline{\text{case}}\ op\ \underline{\text{of}} \\
&\quad \text{Read}\ p\ f\ \Rightarrow \text{Read}\ (\text{Inl}\ p)\ (\lambda x.\ \text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ (f\ x)\ op_2) \\
&\quad |\ \text{Write}\ op\ p\ x \Rightarrow (\underline{\text{case}}\ \mathit{wire}\ p\ \underline{\text{of}} \\
&\qquad \text{None} \Rightarrow \text{Write}\ (\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ op\ op_2)\ (\text{Inl}\ p)\ x \\
&\qquad |\ \text{Some}\ q \Rightarrow \text{Silent}\ (\text{comp\_op}\ \mathit{wire}\ (\text{BENQ}\ q\ x\ \mathit{buf})\ op\ op_2)) \\
&\quad |\ \text{Silent}\ op \Rightarrow \text{Silent}\ (\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ op\ op_2))\ `_c\ \text{choices}\ op_1)\ \cup_c \\
&\quad ((\lambda op.\ \underline{\text{case}}\ op\ \underline{\text{of}} \\
&\quad |\ \text{Write}\ op\ p\ x \Rightarrow \text{Write}\ (\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ op_1\ op)\ (\text{Inr}\ p)\ x \\
&\qquad\qquad `_c\ \text{sound\_reads}\ \mathit{wire}\ \mathit{buf}\ (\text{choices}\ op_2)))
\end{aligned}
$$

Composition: Equation

Equation of the composition operator

$$
\begin{aligned}
&\text{comp\_op } \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ \mathit{op}_2 = \text{Choice } (((\lambda \mathit{op}.\ \underline{\text{case}}\ \mathit{op}\ \underline{\text{of}} \\
&\quad \text{Read } p\ f \ \Rightarrow \text{Read } (\text{Inl } p)\ (\lambda x.\ \text{comp\_op } \mathit{wire}\ \mathit{buf}\ (f\ x)\ \mathit{op}_2) \\
&\quad |\ \text{Write } \mathit{op}\ p\ x \Rightarrow (\underline{\text{case}}\ \mathit{wire}\ p\ \underline{\text{of}} \\
&\qquad \text{None} \Rightarrow \text{Write } (\text{comp\_op } \mathit{wire}\ \mathit{buf}\ \mathit{op}\ \mathit{op}_2)\ (\text{Inl } p)\ x \\
&\qquad |\ \text{Some } q \Rightarrow \text{Silent } (\text{comp\_op } \mathit{wire}\ (\text{BENQ } q\ x\ \mathit{buf})\ \mathit{op}\ \mathit{op}_2)) \\
&\quad |\ \text{Silent } \mathit{op} \Rightarrow \text{Silent } (\text{comp\_op } \mathit{wire}\ \mathit{buf}\ \mathit{op}\ \mathit{op}_2))\ `_c\ \text{choices } \mathit{op}_1)\ \cup_c \\
&\quad ((\lambda \mathit{op}.\ \underline{\text{case}}\ \mathit{op}\ \underline{\text{of}} \\
&\quad \text{Read } p\ f \Rightarrow \underline{\text{if}}\ p \in \text{ran } \mathit{wire} \\
&\qquad \underline{\text{then}}\ \text{Silent } (\text{comp\_op } \mathit{wire}\ (\text{BTL } p\ \mathit{buf})\ \mathit{op}_1\ (f\ (\text{BHD } p\ \mathit{buf}))) \\
&\qquad \underline{\text{else}}\ \text{Read } (\text{Inr } p)\ (\lambda x.\ \text{comp\_op } \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ (f\ x)) \\
&\quad |\ \text{Write } \mathit{op}\ p\ x \Rightarrow \text{Write } (\text{comp\_op } \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ \mathit{op})\ (\text{Inr } p)\ x \\
&\qquad\qquad\qquad\qquad `_c\ \text{sound\_reads } \mathit{wire}\ \mathit{buf}\ (\text{choices } \mathit{op}_2)))
\end{aligned}
$$

Composition: Equation

Equation of the composition operator

$$
\begin{array}{l}
\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ \mathit{op}_2 = \text{Choice}\ (((\lambda \mathit{op}.\ \underline{\text{case}}\ \mathit{op}\ \underline{\text{of}} \\
\quad \text{Read}\ p\ f\ \Rightarrow \text{Read}\ (\text{Inl}\ p)\ (\lambda x.\ \text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ (f\ x)\ \mathit{op}_2) \\
\mid \text{Write}\ \mathit{op}\ p\ x \Rightarrow (\underline{\text{case}}\ \mathit{wire}\ p\ \underline{\text{of}} \\
\qquad \text{None} \Rightarrow \text{Write}\ (\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}\ \mathit{op}_2)\ (\text{Inl}\ p)\ x \\
\quad \mid \text{Some}\ q \Rightarrow \text{Silent}\ (\text{comp\_op}\ \mathit{wire}\ (\text{BENQ}\ q\ x\ \mathit{buf})\ \mathit{op}\ \mathit{op}_2)) \\
\mid \text{Silent}\ \mathit{op} \Rightarrow \text{Silent}\ (\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}\ \mathit{op}_2))\ \grave{}_c\ \text{choices}\ \mathit{op}_1)\ \cup_c \\
((\lambda \mathit{op}.\ \underline{\text{case}}\ \mathit{op}\ \underline{\text{of}} \\
\quad \text{Read}\ p\ f \Rightarrow \underline{\text{if}}\ p \in \text{ran}\ \mathit{wire} \\
\qquad \underline{\text{then}}\ \text{Silent}\ (\text{comp\_op}\ \mathit{wire}\ (\text{BTL}\ p\ \mathit{buf})\ \mathit{op}_1\ (f\ (\text{BHD}\ p\ \mathit{buf}))) \\
\qquad \underline{\text{else}}\ \text{Read}\ (\text{Inr}\ p)\ (\lambda x.\ \text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ (f\ x)) \\
\mid \text{Write}\ \mathit{op}\ p\ x \Rightarrow \text{Write}\ (\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ \mathit{op})\ (\text{Inr}\ p)\ x \\
\mid \text{Silent}\ \mathit{op} \Rightarrow \text{Silent}\ (\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ \mathit{op}))\ \grave{}_c\ \text{sound\_reads}\ \mathit{wire}\ \mathit{buf}\ (\text{choices}\ \mathit{op}_2)))
\end{array}
$$

Asynchronous Dataflow Operators

- Identity: $\mathcal{I}$
- Parallel composition: $op_1 \parallel op_2$
- Sequential composition: $op_1 \bullet op_2$

Asynchronous Dataflow Operators

- Identity: $\mathcal{I}$
- Parallel composition: $op_1 \parallel op_2$
- Sequential composition: $op_1 \bullet op_2$

Feedback operator type ($op \uparrow$)

$\uparrow :: ('i + 'm, 'o + 'm, 'd)\ op \Rightarrow ('i, 'o, 'd)\ op$

Asynchronous Dataflow Operators

- Identity: $\mathcal{I}$
- Parallel composition: $op_1 \parallel op_2$
- Sequential composition: $op_1 \bullet op_2$

Feedback operator type ($op\uparrow$)

$\uparrow :: ('i + 'm, 'o + 'm, 'd)\ op \Rightarrow ('i, 'o, 'd)\ op$

Transpose operator type

$\mathcal{X} :: ('n + 'm, 'm + 'n, 'd)\ op$

Asynchronous Dataflow Operators

- Identity: $\mathcal{I}$
- Parallel composition: $op_1 \parallel op_2$
- Sequential composition: $op_1 \bullet op_2$

Feedback operator type ($op \uparrow$)

$\uparrow$ :: $('i + 'm, 'o + 'm, 'd)\ op \Rightarrow ('i, 'o, 'd)\ op$

Transpose operator type

$\mathcal{X}$ :: $('n + 'm, 'm + 'n, 'd)\ op$

Dummy source operator type

i :: $('i, 'o, 'd)\ op$

Asynchronous Dataflow Operators

- Identity: $\mathcal{I}$
- Parallel composition: $op_1 \parallel op_2$
- Sequential composition: $op_1 \bullet op_2$

Feedback operator type ($op\uparrow$)

$\uparrow\ ::\ ('i + 'm, 'o + 'm, 'd)\ op \Rightarrow ('i, 'o, 'd)\ op$

Transpose operator type

$\mathcal{X}\ ::\ ('n + 'm, 'm + 'n, 'd)\ op$

Dummy source operator type

$\mathrm{i}\ ::\ ('i, 'o, 'd)\ op$

Sink operator type

$!\ ::\ ('i, 'o, 'd)\ op$

Asynchronous Dataflow Operators

- Identity: $\mathcal{I}$
- Parallel composition: $op_1 \parallel op_2$
- Sequential composition: $op_1 \bullet op_2$

Feedback operator type ($op\uparrow$)

$\uparrow :: ('i + 'm, 'o + 'm, 'd)\ op \Rightarrow ('i, 'o, 'd)\ op$

Transpose operator type

$\mathcal{X} :: ('n + 'm, 'm + 'n, 'd)\ op$

Dummy source operator type

$\mathrm{i} :: ('i, 'o, 'd)\ op$

Sink operator type

$! :: ('i, 'o, 'd)\ op$

Split operator type

$\wedge :: ('n, 'n + 'n, 'd)\ op$

Asynchronous Dataflow Operators

- Identity: $\mathcal{I}$
- Parallel composition: $op_1 \parallel op_2$
- Sequential composition: $op_1 \bullet op_2$

Feedback operator type ($op \uparrow$)

$\uparrow$ :: $('i + 'm, 'o + 'm, 'd)$ $op \Rightarrow ('i, 'o, 'd)$ op

Transpose operator type

$\mathcal{X}$ :: $('n + 'm, 'm + 'n, 'd)$ op

Dummy source operator type

i :: $('i, 'o, 'd)$ op

Sink operator type

! :: $('i, 'o, 'd)$ op

Split operator type

Λ :: $('n, 'n + 'n, 'd)$ op

Merge operator type

$\mathcal{V}$ :: $('n + 'n, 'n, 'd)$ op

Asynchronous Dataflow Operators

- Identity: $\mathcal{I}$
- Parallel composition: $op_1 \parallel op_2$
- Sequential composition: $op_1 \bullet op_2$

Feedback operator type ($op \uparrow$)

$\uparrow :: ('i + 'm, 'o + 'm, 'd)\ op \Rightarrow ('i, 'o, 'd)\ op$

Transpose operator type

$\mathcal{X} :: ('n + 'm, 'm + 'n, 'd)\ op$

Dummy source operator type

$\text{¡} :: ('i, 'o, 'd)\ op$

Sink operator type

$! :: ('i, 'o, 'd)\ op$

Split operator type

$\Lambda :: ('n, 'n + 'n, 'd)\ op$

Merge operator type

$\mathcal{V} :: ('n + 'n, 'n, 'd)\ op$

Copy operator type

$\mathcal{C} :: ('n, 'n + 'n, 'd)\ op$

Asynchronous Dataflow Operators

- Identity: $\mathcal{I}$
- Parallel composition: $op_1 \parallel op_2$
- Sequential composition: $op_1 \bullet op_2$

Feedback operator type ($op \uparrow$)

$\uparrow :: ('i + 'm, 'o + 'm, 'd)\ op \Rightarrow ('i, 'o, 'd)\ op$

Transpose operator type

$\mathcal{X}\ :: ('n + 'm, 'm + 'n, 'd)\ op$

Dummy source operator type

$¡\ :: ('i, 'o, 'd)\ op$

Sink operator type

$!\ :: ('i, 'o, 'd)\ op$

Split operator type

$\Lambda\ :: ('n, 'n + 'n, 'd)\ op$

Merge operator type

$\mathcal{V}\ :: ('n + 'n, 'n, 'd)\ op$

Copy operator type

$\mathcal{C}\ :: ('n, 'n + 'n, 'd)\ op$

Equality test operator type

$\mathcal{Q}\ :: ('n + 'n, 'n, 'd\ option)\ op$

Asynchronous Dataflow Properties

Basic Network Algebra Properties

B1: $op_1 \parallel (op_2 \parallel op_3) \approx \text{map\_op} \curvearrowright \curvearrowright (op_1 \parallel op_2) \parallel op_3$

B2_1: $op \parallel (\mathcal{I} :: (0, 0, 'd)\ op) \approx \text{map\_op Inl Inl}\ op$

B2_2: $(\mathcal{I} :: (0, 0, 'd)\ op) \parallel op \approx \text{map\_op Inr Inr}\ op$

B3: $(op_1 \bullet op_2) \bullet op_3 \approx op_1 \bullet (op_2 \bullet op_3)$

B4_1: $op \bullet \mathcal{I} \approx op$ B4_2: $\mathcal{I} \bullet op \approx op$

B5: $(op_1 \parallel op_2) \bullet (op_3 \parallel op_4) \approx (op_1 \bullet op_3) \parallel (op_2 \bullet op_4)$

B6: $\mathcal{I} \parallel \mathcal{I} \approx \mathcal{I}$ B7: $\mathcal{X} \bullet \mathcal{X} \approx \mathcal{I}$

B8: $(\mathcal{X} :: ('i + 0, 0 + 'i, 'd)\ op) \approx \text{map\_op id (case\_sum Inr Inl)}\ \mathcal{I}$

B9: $\mathcal{X} \approx \text{map\_op} \curvearrowright \curvearrowright (\mathcal{X} \parallel \mathcal{I}) \bullet \text{map\_op id} \curvearrowleft (\mathcal{I} \parallel \mathcal{X})$

B10: $(op_1 \parallel op_2) \bullet \mathcal{X} \approx \mathcal{X} \bullet (op_2 \parallel op_1)$

F1: $\mathcal{I} \uparrow \approx (\mathcal{I} :: (0, 0, 'd)\ op)$ F2: $\mathcal{X} \uparrow \approx \mathcal{I}$

R1: $op_2 \bullet (op_1 \uparrow) \approx ((op_2 \parallel \mathcal{I}) \bullet op_1) \uparrow$

R2: $(op_1 \uparrow) \bullet op_2 \approx (op_1 \bullet (op_2 \parallel \mathcal{I})) \uparrow$

R3: $op_1 \parallel (op_2 \uparrow) \approx (\text{map\_op} \curvearrowleft \curvearrowleft (op_1 \parallel op_2)) \uparrow$

R4: $(op_1 \bullet (\mathcal{I} \parallel op_2)) \uparrow \approx ((\mathcal{I} \parallel op_2) \bullet op_1) \uparrow$

R5: $\text{map\_op Inl Inl}\ ((op :: ('i + 0, 'o + 0, 'd)\ op) \uparrow) \approx op$

R6: $(op \uparrow) \uparrow \approx (\text{map\_op} \curvearrowright \curvearrowright op) \uparrow$

Basic Network Algebra Properties

B1: $op_1 \parallel (op_2 \parallel op_3) \approx \text{map\_op} \curvearrowright \curvearrowright (op_1 \parallel op_2) \parallel op_3$

B2_1: $op \parallel (\mathcal{I} :: (0, 0, 'd)\ op) \approx \text{map\_op Inl Inl}\ op$

B2_2: $(\mathcal{I} :: (0, 0, 'd)\ op) \parallel op \approx \text{map\_op Inr Inr}\ op$

B3: $(op_1 \bullet op_2) \bullet op_3 \approx op_1 \bullet (op_2 \bullet op_3)$

B4_1: $op \bullet \mathcal{I} \approx op$ B4_2: $\mathcal{I} \bullet op \approx op$

B5: $(op_1 \parallel op_2) \bullet (op_3 \parallel op_4) \approx (op_1 \bullet op_3) \parallel (op_2 \bullet op_4)$

B6: $\mathcal{I} \parallel \mathcal{I} \approx \mathcal{I}$ B7: $\mathcal{X} \bullet \mathcal{X} \approx \mathcal{I}$

B8: $(\mathcal{X} :: ('i + 0, 0 + 'i, 'd)\ op) \approx \text{map\_op id (case\_sum Inr Inl)}\ \mathcal{I}$

B9: $\mathcal{X} \approx \text{map\_op} \curvearrowright \curvearrowright (\mathcal{X} \parallel \mathcal{I}) \bullet \text{map\_op id} \curvearrowleft (\mathcal{I} \parallel \mathcal{X})$

B10: $(op_1 \parallel op_2) \bullet \mathcal{X} \approx \mathcal{X} \bullet (op_2 \parallel op_1)$

F1: $\mathcal{I} \uparrow \approx (\mathcal{I} :: (0, 0, 'd)\ op)$ F2: $\mathcal{X} \uparrow \approx \mathcal{I}$

R1: $op_2 \bullet (op_1 \uparrow) \approx ((op_2 \parallel \mathcal{I}) \bullet op_1) \uparrow$

R2: $(op_1 \uparrow) \bullet op_2 \approx (op_1 \bullet (op_2 \parallel \mathcal{I})) \uparrow$

R3: $op_1 \parallel (op_2 \uparrow) \approx (\text{map\_op} \curvearrowleft \curvearrowleft (op_1 \parallel op_2)) \uparrow$

R4: $(op_1 \bullet (\mathcal{I} \parallel op_2)) \uparrow \approx ((\mathcal{I} \parallel op_2) \bullet op_1) \uparrow$

R5: $\text{map\_op Inl Inl}\ ((op :: ('i + 0, 'o + 0, 'd)\ op) \uparrow) \approx op$

R6: $(op \uparrow) \uparrow \approx (\text{map\_op} \curvearrowright \curvearrowright op) \uparrow$

Basic Network Algebra Properties

B1: $op_1 \parallel (op_2 \parallel op_3) \approx \text{map\_op} \curvearrowright \curvearrowright (op_1 \parallel op_2) \parallel op_3$

B2_1: $op \parallel (\mathcal{I} :: (0, 0, 'd)\ op) \approx \text{map\_op Inl Inl}\ op$

B2_2: $(\mathcal{I} :: (0, 0, 'd)\ op) \parallel op \approx \text{map\_op Inr Inr}\ op$

B3: $(op_1 \bullet op_2) \bullet op_3 \approx op_1 \bullet (op_2 \bullet op_3)$

B4_1: $op \bullet \mathcal{I} \approx op$ B4_2: $\mathcal{I} \bullet op \approx op$

B5: $(op_1 \parallel op_2) \bullet (op_3 \parallel op_4) \approx (op_1 \bullet op_3) \parallel (op_2 \bullet op_4)$

B6: $\mathcal{I} \parallel \mathcal{I} \approx \mathcal{I}$ B7: $\mathcal{X} \bullet \mathcal{X} \approx \mathcal{I}$

B8: $(\mathcal{X} :: ('i + 0, 0 + 'i, 'd)\ op) \approx \text{map\_op id (case\_sum Inr Inl)}\ \mathcal{I}$

B9: $\mathcal{X} \approx \text{map\_op} \curvearrowright \curvearrowright (\mathcal{X} \parallel \mathcal{I}) \bullet \text{map\_op id} \curvearrowleft (\mathcal{I} \parallel \mathcal{X})$

B10: $(op_1 \parallel op_2) \bullet \mathcal{X} \approx \mathcal{X} \bullet (op_2 \parallel op_1)$

F1: $\mathcal{I} \uparrow \approx (\mathcal{I} :: (0, 0, 'd)\ op)$ F2: $\mathcal{X} \uparrow \approx \mathcal{I}$

R1: $op_2 \bullet (op_1 \uparrow) \approx ((op_2 \parallel \mathcal{I}) \bullet op_1) \uparrow$

R2: $(op_1 \uparrow) \bullet op_2 \approx (op_1 \bullet (op_2 \parallel \mathcal{I})) \uparrow$

R3: $op_1 \parallel (op_2 \uparrow) \approx (\text{map\_op} \curvearrowleft \curvearrowleft (op_1 \parallel op_2)) \uparrow$

R4: $(op_1 \bullet (\mathcal{I} \parallel op_2)) \uparrow \approx ((\mathcal{I} \parallel op_2) \bullet op_1) \uparrow$

R5: $\text{map\_op Inl Inl}\ ((op :: ('i + 0, 'o + 0, 'd)\ op) \uparrow) \approx op$

R6: $(op \uparrow) \uparrow \approx (\text{map\_op} \curvearrowright \curvearrowright op) \uparrow$

Basic Network Algebra Properties

B1: $op_1 \parallel (op_2 \parallel op_3) \approx \text{map\_op} \curvearrowright \curvearrowright (op_1 \parallel op_2) \parallel op_3$
B2_1: $op \parallel (\mathcal{I} :: (0, 0, 'd)\ op) \approx \text{map\_op Inl Inl}\ op$
B2_2: $(\mathcal{I} :: (0, 0, 'd)\ op) \parallel op \approx \text{map\_op Inr Inr}\ op$
B3: $(op_1 \bullet op_2) \bullet op_3 \approx op_1 \bullet (op_2 \bullet op_3)$
B4_1: $op \bullet \mathcal{I} \approx op$ B4_2: $\mathcal{I} \bullet op \approx op$
B5: $(op_1 \parallel op_2) \bullet (op_3 \parallel op_4) \approx (op_1 \bullet op_3) \parallel (op_2 \bullet op_4)$
B6: $\mathcal{I} \parallel \mathcal{I} \approx \mathcal{I}$ B7: $\mathcal{X} \bullet \mathcal{X} \approx \mathcal{I}$
B8: $(\mathcal{X} :: ('i + 0, 0 + 'i, 'd)\ op) \approx \text{map\_op id (case\_sum Inr Inl)}\ \mathcal{I}$
B9: $\mathcal{X} \approx \text{map\_op} \curvearrowright \curvearrowright (\mathcal{X} \parallel \mathcal{I}) \bullet \text{map\_op id} \curvearrowleft (\mathcal{I} \parallel \mathcal{X})$
B10: $(op_1 \parallel op_2) \bullet \mathcal{X} \approx \mathcal{X} \bullet (op_2 \parallel op_1)$
F1: $\mathcal{I} \uparrow \approx (\mathcal{I} :: (0, 0, 'd)\ op)$ F2: $\mathcal{X} \uparrow \approx \mathcal{I}$
R1: $op_2 \bullet (op_1 \uparrow) \approx ((op_2 \parallel \mathcal{I}) \bullet op_1) \uparrow$
R2: $(op_1 \uparrow) \bullet op_2 \approx (op_1 \bullet (op_2 \parallel \mathcal{I})) \uparrow$
R3: $op_1 \parallel (op_2 \uparrow) \approx (\text{map\_op} \curvearrowleft \curvearrowleft (op_1 \parallel op_2)) \uparrow$
R4: $(op_1 \bullet (\mathcal{I} \parallel op_2)) \uparrow \approx ((\mathcal{I} \parallel op_2) \bullet op_1) \uparrow$
R5: $\text{map\_op Inl Inl}\ ((op :: ('i + 0, 'o + 0, 'd)\ op) \uparrow) \approx op$
R6: $(op \uparrow) \uparrow \approx (\text{map\_op} \curvearrowright \curvearrowright op) \uparrow$

Basic Network Algebra Properties

B1: $op_1 \parallel (op_2 \parallel op_3) \approx \text{map\_op} \curvearrowright \curvearrowright (op_1 \parallel op_2) \parallel op_3$

B2_1: $op \parallel (\mathcal{I} :: (0, 0, 'd)\ op) \approx \text{map\_op Inl Inl}\ op$

B2_2: $(\mathcal{I} :: (0, 0, 'd)\ op) \parallel op \approx \text{map\_op Inr Inr}\ op$

B3: $(op_1 \bullet op_2) \bullet op_3 \approx op_1 \bullet (op_2 \bullet op_3)$

B4_1: $op \bullet \mathcal{I} \approx op$

B4_2: $\mathcal{I} \bullet op \approx op$

B5: $(op_1 \parallel op_2) \bullet (op_3 \parallel op_4) \approx (op_1 \bullet op_3) \parallel (op_2 \bullet op_4)$

B6: $\mathcal{I} \parallel \mathcal{I} \approx \mathcal{I}$

B7: $\mathcal{X} \bullet \mathcal{X} \approx \mathcal{I}$

B8: $(\mathcal{X} :: ('i + 0, 0 + 'i, 'd)\ op) \approx \text{map\_op id (case\_sum Inr Inl)}\ \mathcal{I}$

B9: $\mathcal{X} \approx \text{map\_op} \curvearrowright \curvearrowright (\mathcal{X} \parallel \mathcal{I}) \bullet \text{map\_op id} \curvearrowleft (\mathcal{I} \parallel \mathcal{X})$

B10: $(op_1 \parallel op_2) \bullet \mathcal{X} \approx \mathcal{X} \bullet (op_2 \parallel op_1)$

F1: $\mathcal{I} \uparrow \approx (\mathcal{I} :: (0, 0, 'd)\ op)$

F2: $\mathcal{X} \uparrow \approx \mathcal{I}$

R1: $op_2 \bullet (op_1 \uparrow) \approx ((op_2 \parallel \mathcal{I}) \bullet op_1) \uparrow$

R2: $(op_1 \uparrow) \bullet op_2 \approx (op_1 \bullet (op_2 \parallel \mathcal{I})) \uparrow$

R3: $op_1 \parallel (op_2 \uparrow) \approx (\text{map\_op} \curvearrowleft \curvearrowleft (op_1 \parallel op_2)) \uparrow$

R4: $(op_1 \bullet (\mathcal{I} \parallel op_2)) \uparrow \approx ((\mathcal{I} \parallel op_2) \bullet op_1) \uparrow$

R5: $\text{map\_op Inl Inl}\ ((op :: ('i + 0, 'o + 0, 'd)\ op) \uparrow) \approx op$

R6: $(op \uparrow) \uparrow \approx (\text{map\_op} \curvearrowright \curvearrowright op) \uparrow$

Basic Network Algebra Properties

B1: $op_1 \parallel (op_2 \parallel op_3) \approx \text{map\_op} \curvearrowright \curvearrowright (op_1 \parallel op_2) \parallel op_3$
B2_1: $op \parallel (\mathcal{I} :: (0, 0, 'd)\ op) \approx \text{map\_op Inl Inl}\ op$
B2_2: $(\mathcal{I} :: (0, 0, 'd)\ op) \parallel op \approx \text{map\_op Inr Inr}\ op$
B3: $(op_1 \bullet op_2) \bullet op_3 \approx op_1 \bullet (op_2 \bullet op_3)$
B4_1: $op \bullet \mathcal{I} \approx op$ B4_2: $\mathcal{I} \bullet op \approx op$
B5: $(op_1 \parallel op_2) \bullet (op_3 \parallel op_4) \approx (op_1 \bullet op_3) \parallel (op_2 \bullet op_4)$
B6: $\mathcal{I} \parallel \mathcal{I} \approx \mathcal{I}$ B7: $\mathcal{X} \bullet \mathcal{X} \approx \mathcal{I}$
B8: $(\mathcal{X} :: ('i + 0, 0 + 'i, 'd)\ op) \approx \text{map\_op id (case\_sum Inr Inl)}\ \mathcal{I}$
B9: $\mathcal{X} \approx \text{map\_op} \curvearrowright \curvearrowright (\mathcal{X} \parallel \mathcal{I}) \bullet \text{map\_op id} \curvearrowleft (\mathcal{I} \parallel \mathcal{X})$
B10: $(op_1 \parallel op_2) \bullet \mathcal{X} \approx \mathcal{X} \bullet (op_2 \parallel op_1)$
F1: $\mathcal{I} \uparrow \approx (\mathcal{I} :: (0, 0, 'd)\ op)$ F2: $\mathcal{X} \uparrow \approx \mathcal{I}$
R1: $op_2 \bullet (op_1 \uparrow) \approx ((op_2 \parallel \mathcal{I}) \bullet op_1) \uparrow$
R2: $(op_1 \uparrow) \bullet op_2 \approx (op_1 \bullet (op_2 \parallel \mathcal{I})) \uparrow$
R3: $op_1 \parallel (op_2 \uparrow) \approx (\text{map\_op} \curvearrowleft \curvearrowleft (op_1 \parallel op_2)) \uparrow$
R4: $(op_1 \bullet (\mathcal{I} \parallel op_2)) \uparrow \approx ((\mathcal{I} \parallel op_2) \bullet op_1) \uparrow$
R5: $\text{map\_op Inl Inl}\ ((op :: ('i + 0, 'o + 0, 'd)\ op) \uparrow) \approx op$
R6: $(op \uparrow) \uparrow \approx (\text{map\_op} \curvearrowright \curvearrowright op) \uparrow$

Properties of Equality Test, Merge, Copy, Split, Source and Sink operators

$A1: (\mathcal{Q} \parallel \mathcal{I}) \bullet \mathcal{Q} \approx \mathsf{map\_op} \curvearrowleft \mathsf{id}\ ((\mathcal{I} \parallel \mathcal{Q}) \bullet \mathcal{Q})$

$A2: \mathcal{X} \bullet \ogreaterthan \approx \ogreaterthan \quad \text{for } \ogreaterthan \in \{\mathcal{Q}, \mathcal{V}\}$ $\qquad$ $A6: \olessthan \bullet \mathcal{X} \approx \olessthan \quad \text{for } \olessthan \in \{\mathcal{C}, \Lambda\}$

$A3_{\mathcal{Q}}: ((\mathsf{i} :: (0, 'a, 'd)\ op) \parallel \mathcal{I}) \bullet \mathcal{Q} \approx (! :: (0 + 'a, 0, 'd)\ op) \bullet \mathsf{i}$

$A3_{\mathcal{V}}: ((\mathsf{i} :: (0, 'a, 'd)\ op) \parallel \mathcal{I}) \bullet \mathcal{V} \approx \mathsf{map\_op\ Inr\ id}\ \mathcal{I}$

$A4: \ogreaterthan \bullet ! \approx ! \parallel ! \quad \text{for } \ogreaterthan \in \{\mathcal{Q}, \mathcal{V}\}$ $\qquad$ $A8: \mathsf{i} \bullet \olessthan \approx \mathsf{i} \parallel \mathsf{i} \quad \text{for } \olessthan \in \{\mathcal{C}, \Lambda\}$

$A5: \mathcal{C} \bullet (\mathcal{C} \parallel \mathcal{I}) \approx \mathsf{map\_op\ id} \curvearrowleft (\mathcal{C} \bullet (\mathcal{I} \parallel \mathcal{C}))$

$A7: \mathcal{C} \bullet (! \parallel \mathcal{I}) \approx \mathsf{map\_op\ id\ Inr}\ \mathcal{I}$ $\qquad$ $A9: \mathsf{i} \bullet ! \approx \mathcal{I}$

$A10: \mathcal{Q} \bullet \mathcal{C} \approx (\mathcal{C} \parallel \mathcal{C}) \bullet (\mathsf{map\_op} \curvearrowright \curvearrowright (\mathsf{map\_op} \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I})) \bullet (\mathcal{Q} \parallel \mathcal{Q})$

$A11: \mathcal{C} \bullet \mathcal{Q} \approx \mathcal{I}$ $\qquad$ $A12: \mathsf{i} \approx (\mathcal{I} :: (0, 0, 'd)\ op)$

$A16: ! \approx (\mathcal{I} :: (0, 0, 'd)\ op)$ $\qquad$ $A13: \mathsf{i} \approx \mathsf{i} \parallel \mathsf{i}$ $\qquad$ $A17: ! \approx ! \parallel !$

$A14: \mathsf{map\_op\ id\ Inl}\ (\ogreaterthan :: (0 + 0, 0, 'd)\ op) \approx \mathcal{I} \quad \text{for } \ogreaterthan \in \{\mathcal{Q}, \mathcal{V}\}$

$A15: \ogreaterthan \approx \mathsf{map\_op} \curvearrowright \curvearrowright (\mathsf{map\_op} \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \bullet (\ogreaterthan \parallel \ogreaterthan) \quad \text{for } \ogreaterthan \in \{\mathcal{Q}, \mathcal{V}\}$

$A18: \mathsf{map\_op\ Inl\ id}\ (\olessthan :: (0, 0 + 0, 'd)\ op) \approx \mathcal{I} \quad \text{for } \olessthan \in \{\mathcal{C}, \Lambda\}$

$A19: \olessthan \approx (\olessthan \parallel \olessthan) \bullet \mathsf{map\_op} \curvearrowright \curvearrowright (\mathsf{map\_op} \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \quad \text{for } \olessthan \in \{\mathcal{C}, \Lambda\}$

$F3: (\mathsf{map\_op\ id\ Inr}\ \ogreaterthan) \uparrow \approx ! \quad \text{for } \ogreaterthan \in \{\mathcal{Q}, \mathcal{V}\}$

$F4: (\mathsf{map\_op\ Inr\ id}\ \olessthan) \uparrow \approx \mathsf{i} \quad \text{for } \olessthan \in \{\mathcal{C}, \Lambda\}$

$F5: ((\mathcal{I} \parallel \mathcal{C}) \bullet \mathsf{map\_op} \curvearrowright \curvearrowright (\mathcal{X} \parallel \mathcal{I}) \bullet (\mathcal{I} \parallel \mathcal{Q})) \uparrow \approx ! \bullet \mathsf{i}$

Properties of Equality Test, Merge, Copy, Split, Source and Sink operators

$A1: (\mathcal{Q} \parallel \mathcal{I}) \bullet \mathcal{Q} \approx \text{map\_op} \curvearrowleft \text{id} ((\mathcal{I} \parallel \mathcal{Q}) \bullet \mathcal{Q})$

$A2: \mathcal{X} \bullet \ogreaterthan \approx \ogreaterthan \quad \text{for } \ogreaterthan \in \{\mathcal{Q}, \mathcal{V}\}$

$A6: \olessthan \bullet \mathcal{X} \approx \olessthan \quad \text{for } \olessthan \in \{\mathcal{C}, \Lambda\}$

$A3_{\mathcal{Q}}: ((\text{i} :: (0, 'a, 'd)\ op) \parallel \mathcal{I}) \bullet \mathcal{Q} \approx (! :: (0 + 'a, 0, 'd)\ op) \bullet \text{i}$

$A3_{\mathcal{V}}: ((\text{i} :: (0, 'a, 'd)\ op) \parallel \mathcal{I}) \bullet \mathcal{V} \approx \text{map\_op Inr id } \mathcal{I}$

$A4: \ogreaterthan \bullet\, ! \approx\, ! \parallel ! \quad \text{for } \ogreaterthan \in \{\mathcal{Q}, \mathcal{V}\}$

$A8: \text{i} \bullet \olessthan \approx \text{i} \parallel \text{i} \quad \text{for } \olessthan \in \{\mathcal{C}, \Lambda\}$

$A5: \mathcal{C} \bullet (\mathcal{C} \parallel \mathcal{I}) \approx \text{map\_op id} \curvearrowleft (\mathcal{C} \bullet (\mathcal{I} \parallel \mathcal{C}))$

$A7: \mathcal{C} \bullet (! \parallel \mathcal{I}) \approx \text{map\_op id Inr } \mathcal{I}$

$A9: \text{i} \bullet\, ! \approx \mathcal{I}$

$A10: \mathcal{Q} \bullet \mathcal{C} \approx (\mathcal{C} \parallel \mathcal{C}) \bullet (\text{map\_op} \curvearrowright \curvearrowright (\text{map\_op} \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I})) \bullet (\mathcal{Q} \parallel \mathcal{Q})$

$A11: \mathcal{C} \bullet \mathcal{Q} \approx \mathcal{I}$

$A12: \text{i} \approx (\mathcal{I} :: (0, 0, 'd)\ op)$

$A16: ! \approx (\mathcal{I} :: (0, 0, 'd)\ op)$

$A13: \text{i} \approx \text{i} \parallel \text{i}$

$A17: ! \approx\, ! \parallel !$

$A14: \text{map\_op id Inl } (\ogreaterthan :: (0 + 0, 0, 'd)\ op) \approx \mathcal{I} \quad \text{for } \ogreaterthan \in \{\mathcal{Q}, \mathcal{V}\}$

$A15: \ogreaterthan \approx \text{map\_op} \curvearrowright \curvearrowright (\text{map\_op} \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \bullet (\ogreaterthan \parallel \ogreaterthan) \quad \text{for } \ogreaterthan \in \{\mathcal{Q}, \mathcal{V}\}$

$A18: \text{map\_op Inl id } (\olessthan :: (0, 0 + 0, 'd)\ op) \approx \mathcal{I} \quad \text{for } \olessthan \in \{\mathcal{C}, \Lambda\}$

$A19: \olessthan \approx (\olessthan \parallel \olessthan) \bullet \text{map\_op} \curvearrowright \curvearrowright (\text{map\_op} \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \quad \text{for } \olessthan \in \{\mathcal{C}, \Lambda\}$

$F3: (\text{map\_op id Inr } \ogreaterthan) \uparrow \approx\, ! \quad \text{for } \ogreaterthan \in \{\mathcal{Q}, \mathcal{V}\}$

$F4: (\text{map\_op Inr id } \olessthan) \uparrow \approx \text{i} \quad \text{for } \olessthan \in \{\mathcal{C}, \Lambda\}$

$F5: ((\mathcal{I} \parallel \mathcal{C}) \bullet \text{map\_op} \curvearrowright \curvearrowright (\mathcal{X} \parallel \mathcal{I}) \bullet (\mathcal{I} \parallel \mathcal{Q})) \uparrow \approx\, ! \bullet \text{i}$

Properties of Equality Test, Merge, Copy, Split, Source and Sink operators

$A1: (\mathcal{Q} \parallel \mathcal{I}) \bullet \mathcal{Q} \approx \text{map\_op} \curvearrowleft \text{id} ((\mathcal{I} \parallel \mathcal{Q}) \bullet \mathcal{Q})$

$A2: \mathcal{X} \bullet \textcircled{>} \approx \textcircled{>} \quad \text{for } \textcircled{>} \in \{\mathcal{Q}, \mathcal{V}\}$ $\qquad$ $A6: \textcircled{<} \bullet \mathcal{X} \approx \textcircled{<} \quad \text{for } \textcircled{<} \in \{\mathcal{C}, \Lambda\}$

$A3_{\mathcal{Q}}: ((\text{i} :: (0, 'a, 'd)\ op) \parallel \mathcal{I}) \bullet \mathcal{Q} \approx (! :: (0 + 'a, 0, 'd)\ op) \bullet \text{i}$

$A3_{\mathcal{V}}: ((\text{i} :: (0, 'a, 'd)\ op) \parallel \mathcal{I}) \bullet \mathcal{V} \approx \text{map\_op Inr id } \mathcal{I}$

$A4: \textcircled{>} \bullet ! \approx ! \parallel ! \quad \text{for } \textcircled{>} \in \{\mathcal{Q}, \mathcal{V}\}$ $\qquad$ $A8: \text{i} \bullet \textcircled{<} \approx \text{i} \parallel \text{i} \quad \text{for } \textcircled{<} \in \{\mathcal{C}, \Lambda\}$

$A5: \mathcal{C} \bullet (\mathcal{C} \parallel \mathcal{I}) \approx \text{map\_op id} \curvearrowleft (\mathcal{C} \bullet (\mathcal{I} \parallel \mathcal{C}))$

$A7: \mathcal{C} \bullet (! \parallel \mathcal{I}) \approx \text{map\_op id Inr } \mathcal{I}$ $\qquad$ $A9: \text{i} \bullet ! \approx \mathcal{I}$

$A10: \mathcal{Q} \bullet \mathcal{C} \approx (\mathcal{C} \parallel \mathcal{C}) \bullet (\text{map\_op} \curvearrowright \curvearrowright (\text{map\_op} \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I})) \bullet (\mathcal{Q} \parallel \mathcal{Q})$

$A11: \mathcal{C} \bullet \mathcal{Q} \approx \mathcal{I}$ $\qquad$ $A12: \text{i} \approx (\mathcal{I} :: (0, 0, 'd)\ op)$

$A16: ! \approx (\mathcal{I} :: (0, 0, 'd)\ op)$ $\qquad$ $A13: \text{i} \approx \text{i} \parallel \text{i}$ $\qquad$ $A17: ! \approx ! \parallel !$

$A14: \text{map\_op id Inl } (\textcircled{>} :: (0 + 0, 0, 'd)\ op) \approx \mathcal{I} \quad \text{for } \textcircled{>} \in \{\mathcal{Q}, \mathcal{V}\}$

$A15: \textcircled{>} \approx \text{map\_op} \curvearrowright \curvearrowright (\text{map\_op} \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \bullet (\textcircled{>} \parallel \textcircled{>}) \quad \text{for } \textcircled{>} \in \{\mathcal{Q}, \mathcal{V}\}$

$A18: \text{map\_op Inl id } (\textcircled{<} :: (0, 0 + 0, 'd)\ op) \approx \mathcal{I} \quad \text{for } \textcircled{<} \in \{\mathcal{C}, \Lambda\}$

$A19: \textcircled{<} \approx (\textcircled{<} \parallel \textcircled{<}) \bullet \text{map\_op} \curvearrowright \curvearrowright (\text{map\_op} \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \quad \text{for } \textcircled{<} \in \{\mathcal{C}, \Lambda\}$

$F3: (\text{map\_op id Inr } \textcircled{>}) \uparrow \approx ! \quad \text{for } \textcircled{>} \in \{\mathcal{Q}, \mathcal{V}\}$

$F4: (\text{map\_op Inr id } \textcircled{<}) \uparrow \approx \text{i} \quad \text{for } \textcircled{<} \in \{\mathcal{C}, \Lambda\}$

$F5: ((\mathcal{I} \parallel \mathcal{C}) \bullet \text{map\_op} \curvearrowright \curvearrowright (\mathcal{X} \parallel \mathcal{I}) \bullet (\mathcal{I} \parallel \mathcal{Q})) \uparrow \approx ! \bullet \text{i}$

Properties of Equality Test, Merge, Copy, Split, Source and Sink operators

A1: $(\mathcal{Q} \parallel \mathcal{I}) \bullet \mathcal{Q} \approx \text{map\_op} \curvearrowleft \text{id} ((\mathcal{I} \parallel \mathcal{Q}) \bullet \mathcal{Q})$

A2: $\mathcal{X} \bullet \circledgtr \approx \circledgtr \quad \text{for } \circledgtr \in \{\mathcal{Q}, \mathcal{V}\}$ A6: $\circledless \bullet \mathcal{X} \approx \circledless \quad \text{for } \circledless \in \{\mathcal{C}, \Lambda\}$

A3$_{\mathcal{Q}}$: $((\text{i} :: (0, 'a, 'd)\ op) \parallel \mathcal{I}) \bullet \mathcal{Q} \approx (! :: (0 + 'a, 0, 'd)\ op) \bullet \text{i}$

A3$_{\mathcal{V}}$: $((\text{i} :: (0, 'a, 'd)\ op) \parallel \mathcal{I}) \bullet \mathcal{V} \approx \text{map\_op Inr id } \mathcal{I}$

A4: $\circledgtr \bullet ! \approx ! \parallel ! \quad \text{for } \circledgtr \in \{\mathcal{Q}, \mathcal{V}\}$ A8: $\text{i} \bullet \circledless \approx \text{i} \parallel \text{i} \quad \text{for } \circledless \in \{\mathcal{C}, \Lambda\}$

A5: $\mathcal{C} \bullet (\mathcal{C} \parallel \mathcal{I}) \approx \text{map\_op id} \curvearrowleft (\mathcal{C} \bullet (\mathcal{I} \parallel \mathcal{C}))$

A7: $\mathcal{C} \bullet (! \parallel \mathcal{I}) \approx \text{map\_op id Inr } \mathcal{I}$ A9: $\text{i} \bullet ! \approx \mathcal{I}$

A10: $\mathcal{Q} \bullet \mathcal{C} \approx (\mathcal{C} \parallel \mathcal{C}) \bullet (\text{map\_op} \curvearrowright \curvearrowright (\text{map\_op} \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I})) \bullet (\mathcal{Q} \parallel \mathcal{Q})$

A11: $\mathcal{C} \bullet \mathcal{Q} \approx \mathcal{I}$ A12: $\text{i} \approx (\mathcal{I} :: (0, 0, 'd)\ op)$

A16: $! \approx (\mathcal{I} :: (0, 0, 'd)\ op)$ A13: $\text{i} \approx \text{i} \parallel \text{i}$ A17: $! \approx ! \parallel !$

A14: $\text{map\_op id Inl } (\circledgtr :: (0 + 0, 0, 'd)\ op) \approx \mathcal{I} \quad \text{for } \circledgtr \in \{\mathcal{Q}, \mathcal{V}\}$

A15: $\circledgtr \approx \text{map\_op} \curvearrowright \curvearrowright (\text{map\_op} \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \bullet (\circledgtr \parallel \circledgtr) \quad \text{for } \circledgtr \in \{\mathcal{Q}, \mathcal{V}\}$

A18: $\text{map\_op Inl id } (\circledless :: (0, 0 + 0, 'd)\ op) \approx \mathcal{I} \quad \text{for } \circledless \in \{\mathcal{C}, \Lambda\}$

A19: $\circledless \approx (\circledless \parallel \circledless) \bullet \text{map\_op} \curvearrowright \curvearrowright (\text{map\_op} \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \quad \text{for } \circledless \in \{\mathcal{C}, \Lambda\}$

F3: $(\text{map\_op id Inr } \circledgtr) \uparrow \approx ! \quad \text{for } \circledgtr \in \{\mathcal{Q}, \mathcal{V}\}$

F4: $(\text{map\_op Inr id } \circledless) \uparrow \approx \text{i} \quad \text{for } \circledless \in \{\mathcal{C}, \Lambda\}$

F5: $((\mathcal{I} \parallel \mathcal{C}) \bullet \text{map\_op} \curvearrowright \curvearrowright (\mathcal{X} \parallel \mathcal{I}) \bullet (\mathcal{I} \parallel \mathcal{Q})) \uparrow \approx ! \bullet \text{i}$

Properties of Equality Test, Merge, Copy, Split, Source and Sink operators

$\mathsf{A1}: (\mathcal{Q} \parallel \mathcal{I}) \bullet \mathcal{Q} \approx \mathsf{map\_op} \curvearrowleft \mathsf{id}\ ((\mathcal{I} \parallel \mathcal{Q}) \bullet \mathcal{Q})$

$\mathsf{A2}: \mathcal{X} \bullet \circledgt \approx \circledgt \quad \text{for } \circledgt \in \{\mathcal{Q}, \mathcal{V}\}$ $\qquad$ $\mathsf{A6}: \circledless \bullet \mathcal{X} \approx \circledless \quad \text{for } \circledless \in \{\mathcal{C}, \Lambda\}$

$\mathsf{A3}_{\mathcal{Q}}: ((\mathsf{i} :: (0, 'a, 'd)\ op) \parallel \mathcal{I}) \bullet \mathcal{Q} \approx (! :: (0 + 'a, 0, 'd)\ op) \bullet \mathsf{i}$

$\mathsf{A3}_{\mathcal{V}}: ((\mathsf{i} :: (0, 'a, 'd)\ op) \parallel \mathcal{I}) \bullet \mathcal{V} \approx \mathsf{map\_op\ Inr\ id}\ \mathcal{I}$

$\mathsf{A4}: \circledgt \bullet ! \approx ! \parallel ! \quad \text{for } \circledgt \in \{\mathcal{Q}, \mathcal{V}\}$ $\qquad$ $\mathsf{A8}: \mathsf{i} \bullet \circledless \approx \mathsf{i} \parallel \mathsf{i} \quad \text{for } \circledless \in \{\mathcal{C}, \Lambda\}$

$\mathsf{A5}: \mathcal{C} \bullet (\mathcal{C} \parallel \mathcal{I}) \approx \mathsf{map\_op\ id} \curvearrowleft (\mathcal{C} \bullet (\mathcal{I} \parallel \mathcal{C}))$

$\mathsf{A7}: \mathcal{C} \bullet (! \parallel \mathcal{I}) \approx \mathsf{map\_op\ id\ Inr}\ \mathcal{I}$ $\qquad$ $\mathsf{A9}: \mathsf{i} \bullet ! \approx \mathcal{I}$

$\mathsf{A10}: \mathcal{Q} \bullet \mathcal{C} \approx (\mathcal{C} \parallel \mathcal{C}) \bullet (\mathsf{map\_op} \curvearrowright \curvearrowright (\mathsf{map\_op} \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I})) \bullet (\mathcal{Q} \parallel \mathcal{Q})$

$\mathsf{A11}: \mathcal{C} \bullet \mathcal{Q} \approx \mathcal{I}$ $\qquad$ $\mathsf{A12}: \mathsf{i} \approx (\mathcal{I} :: (0, 0, 'd)\ op)$

$\mathsf{A16}: ! \approx (\mathcal{I} :: (0, 0, 'd)\ op)$ $\qquad$ $\mathsf{A13}: \mathsf{i} \approx \mathsf{i} \parallel \mathsf{i}$ $\qquad$ $\mathsf{A17}: ! \approx ! \parallel !$

$\mathsf{A14}: \mathsf{map\_op\ id\ Inl}\ (\circledgt :: (0 + 0, 0, 'd)\ op) \approx \mathcal{I} \quad \text{for } \circledgt \in \{\mathcal{Q}, \mathcal{V}\}$

$\mathsf{A15}: \circledgt \approx \mathsf{map\_op} \curvearrowright \curvearrowright (\mathsf{map\_op} \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \bullet (\circledgt \parallel \circledgt) \quad \text{for } \circledgt \in \{\mathcal{Q}, \mathcal{V}\}$

$\mathsf{A18}: \mathsf{map\_op\ Inl\ id}\ (\circledless :: (0, 0 + 0, 'd)\ op) \approx \mathcal{I} \quad \text{for } \circledless \in \{\mathcal{C}, \Lambda\}$

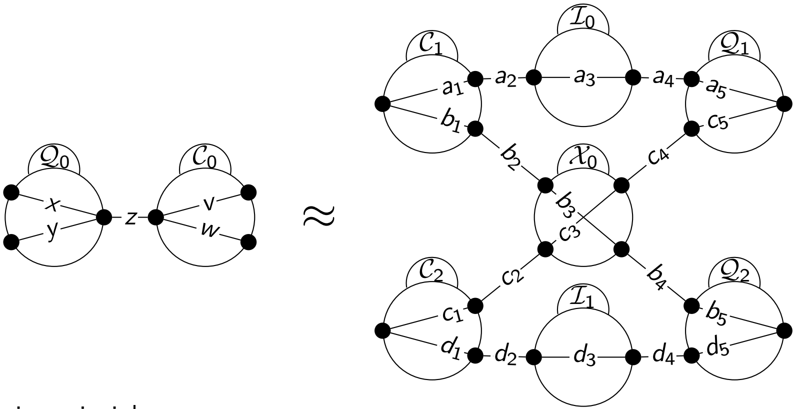
$\mathsf{A19}: \circledless \approx (\circledless \parallel \circledless) \bullet \mathsf{map\_op} \curvearrowright \curvearrowright (\mathsf{map\_op} \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \quad \text{for } \circledless \in \{\mathcal{C}, \Lambda\}$

$\mathsf{F3}: (\mathsf{map\_op\ id\ Inr}\ \circledgt) \uparrow \approx ! \quad \text{for } \circledgt \in \{\mathcal{Q}, \mathcal{V}\}$

$\mathsf{F4}: (\mathsf{map\_op\ Inr\ id}\ \circledless) \uparrow \approx \mathsf{i} \quad \text{for } \circledless \in \{\mathcal{C}, \Lambda\}$

$\mathsf{F5}: ((\mathcal{I} \parallel \mathcal{C}) \bullet \mathsf{map\_op} \curvearrowright \curvearrowright (\mathcal{X} \parallel \mathcal{I}) \bullet (\mathcal{I} \parallel \mathcal{Q})) \uparrow \approx ! \bullet \mathsf{i}$

A10



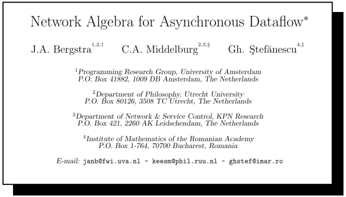
- $\approx$ coinduction principle
- Buffers generalization
- Buffers invariant

Related Work

Related Work

Related Work

- Instance in Algebra of Communicating Processes (ACP)
- Our operators closely follow theirs


Network Algebra for Asynchronous Dataflow*

J.A. Bergstra[1,2,†] C.A. Middelburg[2,3,§] Gh. Ştefănescu[4,‡]

[1]Programming Research Group, University of Amsterdam
P.O. Box 41882, 1009 DB Amsterdam, The Netherlands

[2]Department of Philosophy, Utrecht University
P.O. Box 80126, 3508 TC Utrecht, The Netherlands

[3]Department of Network & Service Control, KPN Research
P.O. Box 421, 2260 AK Leidschendam, The Netherlands

[4]Institute of Mathematics of the Romanian Academy
P.O. Box 1-764, 70700 Bucharest, Romania

E-mail: janb@fwi.uva.nl - keesm@phil.ruu.nl - ghstef@imar.ro

Related Work

- Instance in Algebra of Communicating Processes (ACP)
- Our operators closely follow theirs

Network Algebra for Asynchronous Dataflow*

J.A. Bergstra[1,2,†] C.A. Middelburg[2,3,§] Gh. Ştefănescu[4,‡]

[1]Programming Research Group, University of Amsterdam
P.O. Box 41882, 1009 DB Amsterdam, The Netherlands

[2]Department of Philosophy, Utrecht University
P.O. Box 80126, 3508 TC Utrecht, The Netherlands

[3]Department of Network & Service Control, KPN Research
P.O. Box 421, 2260 AK Leidschendam, The Netherlands

[4]Institute of Mathematics of the Romanian Academy
P.O. Box 1-764, 70700 Bucharest, Romania

E-mail: janb@fwi.uva.nl – keesm@phil.ruu.nl – ghstef@imar.ro

- Our operators can be seen as a domain specific variant of choice trees

Interaction Trees

Representing Recursive and Impure Programs in Coq

LI-YAO XIA, University of Pennsylvania, USA
YANNICK ZAKOWSKI, University of Pennsylvania, USA
PAUL HE, University of Pennsylvania, USA
CHUNG-KIL HUR, Seoul National University, Republic of Korea
GREGORY MALECHA, BedRock Systems, USA
BENJAMIN C. PIERCE, University of Pennsylvania, USA
STEVE ZDANCEWIC, University of Pennsylvania, USA

Choice Trees

Representing Nondeterministic, Recursive, and Impure Programs in Coq

NICOLAS CHAPPE, Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France
PAUL HE, University of Pennsylvania, USA
LUDOVIC HENRIO, Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France
YANNICK ZAKOWSKI, Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France
STEVE ZDANCEWIC, University of Pennsylvania, USA

Related Work

- Instance in Algebra of Communicating Processes (ACP)
- Our operators closely follow theirs

Network Algebra for Asynchronous Dataflow*

J.A. Bergstra[1,2,†] C.A. Middelburg[2,3,§] Gh. Ştefănescu[4,‡]

[1]Programming Research Group, University of Amsterdam
P.O. Box 41882, 1009 DB Amsterdam, The Netherlands

[2]Department of Philosophy, Utrecht University
P.O. Box 80126, 3508 TC Utrecht, The Netherlands

[3]Department of Network & Service Control, KPN Research
P.O. Box 421, 2260 AK Leidschendam, The Netherlands

[4]Institute of Mathematics of the Romanian Academy
P.O. Box 1-764, 70700 Bucharest, Romania

E-mail: janb@fwi.uva.nl - keesm@phil.ruu.nl - ghstef@imar.ro

- **Synchronous** dataflow languages have some proof assistant mechanization

A Formally Verified Compiler for Lustre

Timothy Bourke[1,2] Timothy.Bourke@inria.fr
Lélio Brun[2,1] Lelio.Brun@ens.fr
Pierre-Évariste Dagand[3,4,1] Pierre-Evariste.Dagand@lip6.fr
Xavier Leroy[1] Xavier.Leroy@inria.fr
Marc Pouzet[3,2,1] Marc.Pouzet@ens.fr
Lionel Rieg[5,6] Lionel.Rieg@yale.edu

[1] Inria, Paris, France
[2] Département d'Informatique, École normale supérieure, PSL Research University, Paris, France
[3] Sorbonne Universités, UPMC Univ Paris 06, France
[4] CNRS, LIP6 UMR 7606, Paris, France
[5] Collège de France, Paris, France
[6] Yale University, New Haven, Connecticut, USA

- Our operators can be seen as a domain specific variant of choice trees

Interaction Trees

Representing Recursive and Impure Programs in Coq

LI-YAO XIA, University of Pennsylvania, USA
YANNICK ZAKOWSKI, University of Pennsylvania, USA
PAUL HE, University of Pennsylvania, USA
CHUNG-KIL HUR, Seoul National University, Republic of Korea
GREGORY MALECHA, BedRock Systems, USA
BENJAMIN C. PIERCE, University of Pennsylvania, USA
STEVE ZDANCEWIC, University of Pennsylvania, USA

Choice Trees

Representing Nondeterministic, Recursive, and Impure Programs in Coq

NICOLAS CHAPPE, Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France
PAUL HE, University of Pennsylvania, USA
LUDOVIC HENRIO, Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France
YANNICK ZAKOWSKI, Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France
STEVE ZDANCEWIC, University of Pennsylvania, USA

Related Work

- Instance in Algebra of Communicating Processes (ACP)
- Our operators closely follow theirs

Network Algebra for Asynchronous Dataflow*

J.A. Bergstra[1,2,†] C.A. Middelburg[2,3,§] Gh. Ştefănescu[4,‡]

[1]Programming Research Group, University of Amsterdam
P.O. Box 41882, 1009 DB Amsterdam, The Netherlands

[2]Department of Philosophy, Utrecht University
P.O. Box 80126, 3508 TC Utrecht, The Netherlands

[3]Department of Network & Service Control, KPN Research
P.O. Box 421, 2260 AK Leidschendam, The Netherlands

[4]Institute of Mathematics of the Romanian Academy
P.O. Box 1-764, 70700 Bucharest, Romania

E-mail: janb@fwi.uva.nl - keesm@phil.ruu.nl - ghstef@imar.ro

- **Synchronous** dataflow languages have some proof assistant mechanization

A Formally Verified Compiler for Lustre

Timothy Bourke[1,2] Timothy.Bourke@inria.fr
Lélio Brun[2,1] Lelio.Brun@ens.fr
Pierre-Évariste Dagand[3,4,1] Pierre-Evariste.Dagand@lip6.fr
Xavier Leroy[1] Xavier.Leroy@inria.fr
Marc Pouzet[3,2,1] Marc.Pouzet@ens.fr
Lionel Rieg[5,6] Lionel.Rieg@yale.edu

[1] Inria, Paris, France
[2] Département d'Informatique, École normale supérieure, PSL Research University, Paris, France
[3] Sorbonne Universités, UPMC Univ Paris 06, France
[4] CNRS, LIP6 UMR 7606, Paris, France
[5] Collège de France, Paris, France
[6] Yale University, New Haven, Connecticut, USA

- Our operators can be seen as a domain specific variant of choice trees

Interaction Trees

Representing Recursive and Impure Programs in Coq

LI-YAO XIA, University of Pennsylvania, USA
YANNICK ZAKOWSKI, University of Pennsylvania, USA
PAUL HE, University of Pennsylvania, USA
CHUNG-KIL HUR, Seoul National University, Republic of Korea
GREGORY MALECHA, BedRock Systems, USA
BENJAMIN C. PIERCE, University of Pennsylvania, USA
STEVE ZDANCEWIC, University of Pennsylvania, USA

Choice Trees

Representing Nondeterministic, Recursive, and Impure Programs in Coq

NICOLAS CHAPPE, Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France
PAUL HE, University of Pennsylvania, USA
LUDOVIC HENRIO, Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France
YANNICK ZAKOWSKI, Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France
STEVE ZDANCEWIC, University of Pennsylvania, USA

- Flo: a framework for representing streaming computations originating from different systems like Flink and DBSP

Flo: a Semantic Foundation for Progressive Stream Processing

SHADAJ LADDAD, UC Berkeley, USA
ALVIN CHEUNG, UC Berkeley, USA
JOSEPH M. HELLERSTEIN, UC Berkeley, USA
MAE MILANO, Princeton University, USA

Conclusion

Conclusion

- An Isabelle/HOL instance of Nondeterministic Asynchronous Dataflow

Conclusion

- An Isabelle/HOL instance of Nondeterministic Asynchronous Dataflow
- Isabelle/HOL has a good tool set:
  - Codatatypes, corecursion, coinductive predicates

Conclusion

- An Isabelle/HOL instance of Nondeterministic Asynchronous Dataflow
- Isabelle/HOL has a good tool set:
  - Codatatypes, corecursion, coinductive predicates
- 51/52 axioms proved
  - Axiom A1 for the merge operator does not hold in our instance when equivalence is $\approx$. We conjecture it also does not hold in the process algebra from Bergstra et al.

Conclusion

- An Isabelle/HOL instance of Nondeterministic Asynchronous Dataflow
- Isabelle/HOL has a good tool set:
  - Codatatypes, corecursion, coinductive predicates
- 51/52 axioms proved
  - Axiom A1 for the merge operator does not hold in our instance when equivalence is $\approx$. We conjecture it also does not hold in the process algebra from Bergstra et al.
- Jonsson's trace model, trace equivalence

Conclusion

- An Isabelle/HOL instance of Nondeterministic Asynchronous Dataflow
- Isabelle/HOL has a good tool set:
  - Codatatypes, corecursion, coinductive predicates
- 51/52 axioms proved
  - Axiom A1 for the merge operator does not hold in our instance when equivalence is $\approx$. We conjecture it also does not hold in the process algebra from Bergstra et al.
- Jonsson's trace model, trace equivalence
- Around 28 000 lines of definitions and proofs

Conclusion

- An Isabelle/HOL instance of Nondeterministic Asynchronous Dataflow
- Isabelle/HOL has a good tool set:
  - Codatatypes, corecursion, coinductive predicates
- 51/52 axioms proved
  - Axiom A1 for the merge operator does not hold in our instance when equivalence is $\approx$. We conjecture it also does not hold in the process algebra from Bergstra et al.
- Jonsson's trace model, trace equivalence
- Around 28 000 lines of definitions and proofs
- Future work:
  - Brock–Andersen Time anomaly
  - Formalize Timely Dataflow infrastructure
  - Verify Timely Dataflow algorithms

Questions, comments and suggestions