Nondeterministic Asynchronous Dataflow in Isabelle/HOL

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

Department of Computer Science
University of Copenhagen

28/09/2025

Motivation

Motivation

Context:

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

Cloud DataFlow Flink

Motivation

Context:

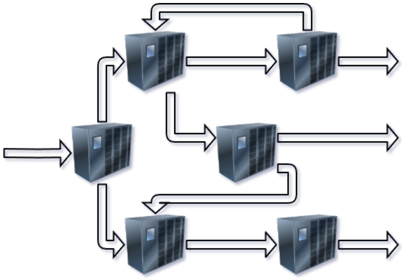
- Stream Processing: programs that compute with (possibly) unbounded sequences of data
- Frameworks: Google Cloud Dataflow, Apache Flink, and Timely Dataflow

Our long term goal:

Mechanically Verify Timely Dataflow algorithms

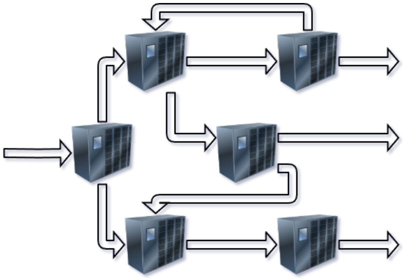
A Good Foundation

A Good Foundation



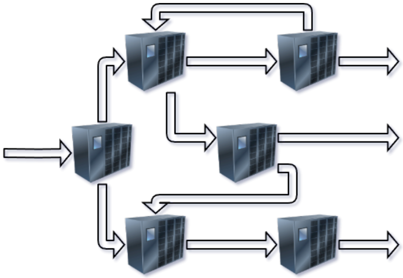
- Nondeterministic Asynchronous Dataflow

A Good Foundation



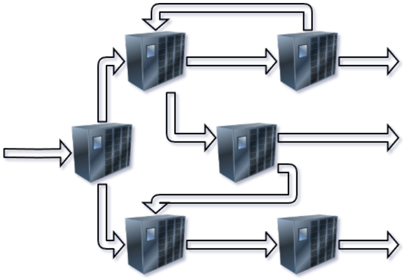
- Nondeterministic Asynchronous Dataflow
  - Dataflow: Directed graph of interconnected operators
    - Operators are (possibly) non-terminating processes that communicate with the network
    - Networks are unbounded FIFO queues

A Good Foundation



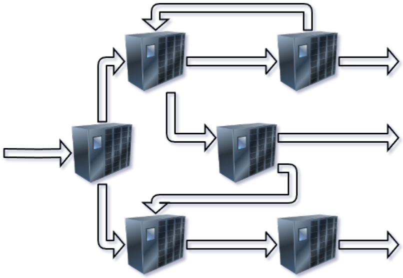
- Nondeterministic Asynchronous Dataflow
  - Dataflow: Directed graph of interconnected operators
    - Operators are (possibly) non-terminating processes that communicate with the network
    - Networks are unbounded FIFO queues
  - Asynchronous:
    - Operators execute independently: processes without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

A Good Foundation



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

A Good Foundation



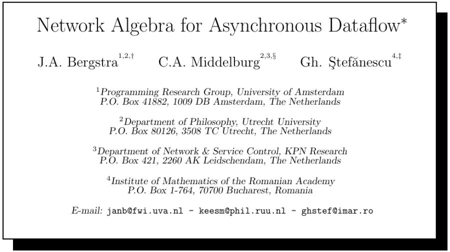
- Nondeterministic Asynchronous Dataflow
  - Dataflow: Directed graph of interconnected operators
    - Operators are (possibly) non-terminating processes that communicate with the network
    - Networks are unbounded FIFO queues
  - Asynchronous:
    - Operators execute independently: processes without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

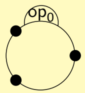
Operators as a Codatatype

Operators

Operators in Isabelle/HOL

codatatype $('i, 'o, 'd)\ op =$
Read $'i\ ('d \Rightarrow ('i, 'o, 'd)\ op)$ | Write $(('i, 'o, 'd)\ op)\ 'o\ 'd$ |
Silent $(('i, 'o, 'd)\ op)$ | Choice $(('i, 'o, 'd)\ op)\ cset$

$(2, 1, nat)\ op$ shape



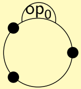
- Type parameters:
  inputs/output ports; data
- Operator's actions
- Possibly infinite trees

Operators

Operators in Isabelle/HOL

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

$(2, 1, nat)\ op$ shape



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

Uncommunicative operators

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

$\odot = \text{Silent}\ \odot$

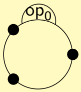
$\otimes = \text{Choice}\ \{\otimes\}_c$

Operators

Operators in Isabelle/HOL

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

$(2, 1, nat)$ *op* shape



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

Uncommunicative operators

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

$$\odot = \mathrm{Silent}\ \odot$$

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

More examples

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

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

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

Operators Equivalences: Weak Bisimilarity

An ideal equivalence relation

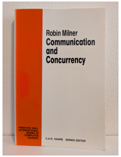
- Only equate operators that **behave** the same
- Useful reasoning principle

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**

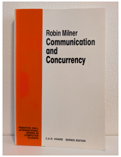


- Based on labeled transition systems (LTS)
- **Weak**: silent computations are abstracted away

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
- $\oslash \approx \odot$, $\odot \approx \otimes$, ex1 $\approx$ ex2, and ex2 $\approx$ ex3

Asynchronous Dataflow Operators

Auxiliary Definitions

Auxiliary Definitions

- Buffers: $'p \Rightarrow {'d}\ \mathit{list}$

Buffer functions

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

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

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

Auxiliary Definitions

- Buffers: $'p \Rightarrow 'd\ list$
- choices: flattens the choices of an operator (no top-level Choice constructor in the returned set)

Buffer functions

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

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

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

choices function

$$\text{choices} :: ('i, 'o, 'd)\ op \Rightarrow (('i, 'o, 'd)\ op)\ cset$$

Auxiliary Definitions

- Buffers: $'p \Rightarrow 'd \ \mathit{list}$
- choices: flattens the choices of an operator (no top-level Choice constructor in the returned set)

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 function

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

Mapping ports functions

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

Auxiliary Definitions

- Buffers: $'p \Rightarrow 'd\ list$
- choices: flattens the choices of an operator (no top-level Choice constructor in the returned set)

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 function

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

Mapping ports functions

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

Auxiliary Definitions

- Buffers: $'p \Rightarrow 'd\ list$
- choices: flattens the choices of an operator (no top-level Choice constructor in the returned set)

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 function

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

Mapping ports functions

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

Auxiliary Definitions

- Buffers: $'p \Rightarrow 'd\ list$
- choices: flattens the choices of an operator (no top-level Choice constructor in the returned set)

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 function

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

Mapping ports functions

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

Auxiliary Definitions

- Buffers: $'p \Rightarrow 'd\ list$
- choices: flattens the choices of an operator (no top-level Choice constructor in the returned set)

Buffer functions

$$\text{BHD}\ p\ buf = \text{hd}\ (buf\ p)$$
$$\text{BTL}\ p\ buf = buf(p := \text{tl}\ (buf\ p))$$
$$\text{BENQ}\ p\ x\ buf = buf(p := buf\ p\ @\ [x])$$

choices function

$$\text{choices} :: ('i, 'o, 'd)\ op \Rightarrow (('i, 'o, 'd)\ op)\ cset$$

Mapping ports functions

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

Auxiliary Definitions

- Buffers: $'p \Rightarrow 'd\ list$
- choices: flattens the choices of an operator (no top-level Choice constructor in the returned set)

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 function

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

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

$$\text{id\_op} :: ('p \Rightarrow 'd\ list) \Rightarrow ('p, 'p, 'd)\ op$$

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

$$\cup_c$$

Identity

Equation of the identity operator

$$\text{id\_op} :: (\text{'}p \Rightarrow \text{'}d\ \mathit{list}) \Rightarrow (\text{'}p, \text{'}p, \text{'}d)\ \mathit{op}$$
$$\text{id\_op}\ \mathit{buf} = \text{Choice}$$
$$(((\lambda p.\ \text{Read}\ p\ (\lambda x.\ \text{id\_op}\ (\text{BENQ}\ p\ x\ \mathit{buf})))\ \grave{}_c\ \mathfrak{U}_c)$$
$$\cup_c$$

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

Identity

Equation of the identity operator

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

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

Identity

Equation of the identity operator

$$\text{id\_op} :: ('p \Rightarrow 'd\ \textit{list}) \Rightarrow ('p, 'p, 'd)\ \textit{op}$$

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

$$(((\lambda p.\ \text{Read}\ p\ (\lambda x.\ \text{id\_op}\ (\text{BENQ}\ p\ x\ \textit{buf})))\ `_c\ \mathfrak{U}_c)$$

$$\cup_c$$

$$((\lambda p.\ \text{Write}\ (\text{id\_op}\ (\text{BTL}\ p\ \textit{buf}))\ p\ (\text{BHD}\ p\ \textit{buf}))\ `_c\ \{p \in_c \mathfrak{U}_c \mid \textit{buf}\ \ p \neq []\}))$$

- $\mathfrak{U}_c$ 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

$$\text{id\_op} :: ('p \Rightarrow 'd\ list) \Rightarrow ('p, 'p, 'd)\ op$$
$$\text{id\_op}\ buf = \text{Choice}$$
$$(((\lambda p.\ \text{Read}\ p\ (\lambda x.\ \text{id\_op}\ (\text{BENQ}\ p\ x\ buf)))\ `_c\ \mathfrak{U}_c)$$
$$\cup_c$$
$$((\lambda p.\ \text{Write}\ (\text{id\_op}\ (\text{BTL}\ p\ buf))\ p\ (\text{BHD}\ p\ buf))\ `_c\ \{p \in_c \mathfrak{U}_c \mid buf\ \ p \neq []\}))$$

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

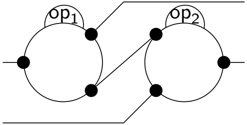
Identity operator with an empty buffer

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

Composition: Preliminaries

Composition operator type

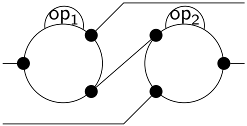
$$\text{comp\_op} :: ('o_1 \Rightarrow 'i_2\ option) \Rightarrow ('i_2 \Rightarrow 'd\ list) \Rightarrow ('i_1, 'o_1, 'd)\ op \Rightarrow ('i_2, 'o_2, 'd)\ op \Rightarrow ('i_1 + 'i_2, 'o_1 + 'o_2, 'd)\ op$$



Composition: Preliminaries

Composition operator type

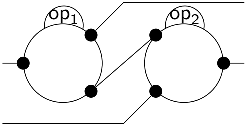
$$\text{comp\_op} :: ('o_1 \Rightarrow 'i_2\ option) \Rightarrow ('i_2 \Rightarrow 'd\ list) \Rightarrow ('i_1, 'o_1, 'd)\ op \Rightarrow ('i_2, 'o_2, 'd)\ op \Rightarrow ('i_1 + 'i_2, 'o_1 + 'o_2, 'd)\ op$$



Composition: Preliminaries

Composition operator type

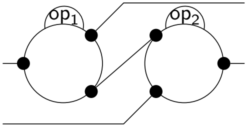
$$\text{comp\_op} :: ('o_1 \Rightarrow 'i_2\ option) \Rightarrow ('i_2 \Rightarrow 'd\ list) \Rightarrow ('i_1, 'o_1, 'd)\ op \Rightarrow ('i_2, 'o_2, 'd)\ op \Rightarrow ('i_1 + 'i_2, 'o_1 + 'o_2, 'd)\ op$$



Composition: Preliminaries

Composition operator type

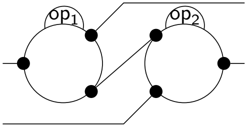
$$\text{comp\_op} :: ('o_1 \Rightarrow 'i_2\ option) \Rightarrow ('i_2 \Rightarrow 'd\ list) \Rightarrow ('i_1, 'o_1, 'd)\ op \Rightarrow ('i_2, 'o_2, 'd)\ op \Rightarrow ('i_1 + 'i_2, 'o_1 + 'o_2, 'd)\ op$$



Composition: Preliminaries

Composition operator type

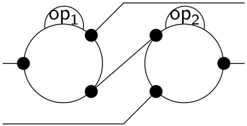
$$\text{comp\_op} :: ('o_1 \Rightarrow 'i_2\ option) \Rightarrow ('i_2 \Rightarrow 'd\ list) \Rightarrow ('i_1, 'o_1, 'd)\ op \Rightarrow ('i_2, 'o_2, 'd)\ op \Rightarrow$$
$$('i_1 + 'i_2, 'o_1 + 'o_2, 'd)\ op$$



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

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

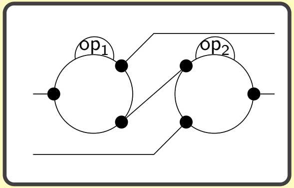
$\cup_c$

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>

$`_c$ choices $op_1) \cup_c$

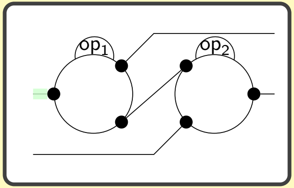


Composition: Equation

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

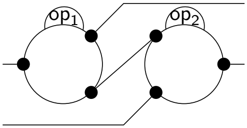
Equation of the composition operator

$\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ \mathit{op}_2 = \text{Choice}\ (((\lambda \mathit{op}.\ \underline{\text{case}}\ \mathit{op}\ \underline{\text{of}}$

$\quad \text{Read}\ p\ f\ \ \Rightarrow \text{Read}\ (\text{Inl}\ p)\ (\lambda x.\ \text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ (f\ x)\ \mathit{op}_2)$

$\mid \text{Write}\ \mathit{op}\ p\ x \Rightarrow (\underline{\text{case}}\ \mathit{wire}\ p\ \underline{\text{of}}$

$`_c\ \ \text{choices}\ \mathit{op}_1)\ \cup_c$



Composition: Equation

Equation of the composition operator

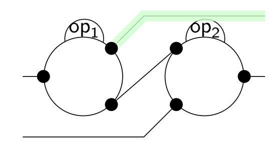
$\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ \mathit{op}_2 = \text{Choice}\ (((\lambda \mathit{op}.\ \underline{\text{case}}\ \mathit{op}\ \underline{\text{of}}$

$\quad \text{Read}\ p\ f\ \Rightarrow \text{Read}\ (\text{Inl}\ p)\ (\lambda x.\ \text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ (f\ x)\ \mathit{op}_2)$

$\mid\ \text{Write}\ \mathit{op}\ p\ x \Rightarrow (\underline{\text{case}}\ \mathit{wire}\ p\ \underline{\text{of}}$

$\qquad \text{None} \Rightarrow \text{Write}\ (\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}\ \mathit{op}_2)\ (\text{Inl}\ p)\ x$

$`_c$ choices op_1) $\cup_c$



Composition: Equation

Equation of the composition operator

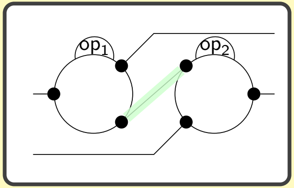
comp_op *wire buf* op_1 op_2 = Choice ((($\lambda op.$ <u>case</u> *op* <u>of</u>

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

| Write *op* p x $\Rightarrow$ (<u>case</u> *wire* p <u>of</u>

None $\Rightarrow$ Write (comp_op *wire buf op* op_2) (Inl p) x

| Some q $\Rightarrow$ Silent (comp_op *wire* (BENQ q x *buf*) *op* op_2))



$`_c$ choices op_1) $\cup_c$

Composition: Equation

Equation of the composition operator

$\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ \mathit{op}_2 = \text{Choice}\ (((\lambda \mathit{op}.\ \underline{\text{case}}\ \mathit{op}\ \underline{\text{of}}$

$\quad \text{Read}\ p\ f \ \Rightarrow \text{Read}\ (\text{Inl}\ p)\ (\lambda x.\ \text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ (f\ x)\ \mathit{op}_2)$

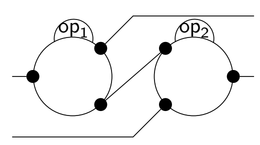
$\mid \text{Write}\ \mathit{op}\ p\ x \Rightarrow (\underline{\text{case}}\ \mathit{wire}\ p\ \underline{\text{of}}$

$\qquad \text{None} \Rightarrow \text{Write}\ (\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}\ \mathit{op}_2)\ (\text{Inl}\ p)\ x$

$\quad \mid \text{Some}\ q \Rightarrow \text{Silent}\ (\text{comp\_op}\ \mathit{wire}\ (\text{BENQ}\ q\ x\ \mathit{buf})\ \mathit{op}\ \mathit{op}_2))$

$\mid \text{Silent}\ \mathit{op} \Rightarrow \text{Silent}\ (\text{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}\ \mathit{op}_2))$

$`_c\ \text{choices}\ \mathit{op}_1)\ \cup_c$



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)

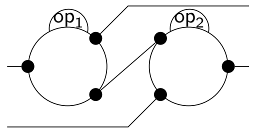
| Write *op p x* $\Rightarrow$ (<u>case</u> *wire p* <u>of</u>

None $\Rightarrow$ Write (comp_op *wire buf op* op_2) (Inl *p*) *x*

| Some *q* $\Rightarrow$ Silent (comp_op *wire* (BENQ *q x buf*) *op* op_2))

| Silent *op* $\Rightarrow$ Silent (comp_op *wire buf op* op_2))

\`$_c$ 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)

| Write *op p x* $\Rightarrow$ (<u>case</u> *wire p* <u>of</u>

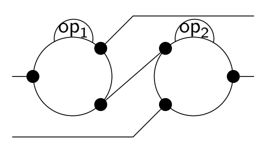
None $\Rightarrow$ Write (comp_op *wire buf op* op_2) (Inl *p*) *x*

| Some *q* $\Rightarrow$ Silent (comp_op *wire* (BENQ *q x buf*) *op* op_2))

| Silent *op* $\Rightarrow$ Silent (comp_op *wire buf op* op_2))

\`$_c$ choices op_1) $\cup_c$

(($\lambda op.$ <u>case</u> *op* <u>of</u>

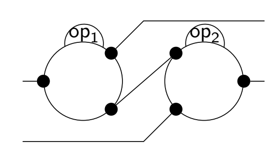


\`$_c$ sound_reads *wire buf* (choices op_2)))

Composition: Equation

Equation of the composition operator

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

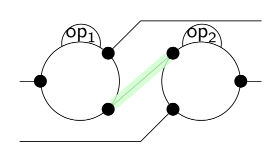


$$`_c\ \ \mathrm{sound\_reads}\ \mathit{wire}\ \mathit{buf}\ (\mathrm{choices}\ \mathit{op}_2)))$$

Composition: Equation

Equation of the composition operator

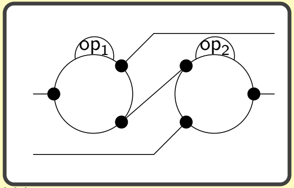
$$
\begin{aligned}
&\mathsf{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ \mathit{op}_2 = \mathsf{Choice}\ (((\lambda \mathit{op}.\ \underline{\mathsf{case}}\ \mathit{op}\ \underline{\mathsf{of}}\\
&\quad \mathsf{Read}\ p\ f\ \ \Rightarrow \mathsf{Read}\ (\mathsf{Inl}\ p)\ (\lambda x.\ \mathsf{comp\_op}\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



Composition: Equation

Equation of the composition operator

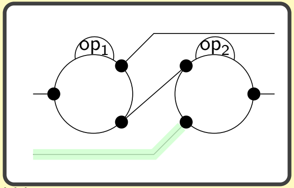
$$
\begin{array}{l}
\mathrm{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ \mathit{op}_2 = \mathrm{Choice}\ (((\lambda \mathit{op}.\ \underline{\mathrm{case}}\ \mathit{op}\ \underline{\mathrm{of}} \\
\quad \mathrm{Read}\ p\ f \ \Rightarrow \mathrm{Read}\ (\mathrm{Inl}\ p)\ (\lambda x.\ \mathrm{comp\_op}\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



Composition: Equation

Equation of the composition operator

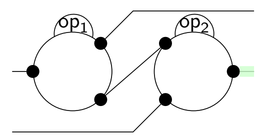
$$
\begin{array}{l}
\mathsf{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ \mathit{op}_2 = \mathsf{Choice}\ (((\lambda \mathit{op}.\ \underline{\mathsf{case}}\ \mathit{op}\ \underline{\mathsf{of}} \\
\quad \mathsf{Read}\ p\ f\ \Rightarrow \mathsf{Read}\ (\mathsf{Inl}\ p)\ (\lambda x.\ \mathsf{comp\_op}\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



Composition: Equation

Equation of the composition operator

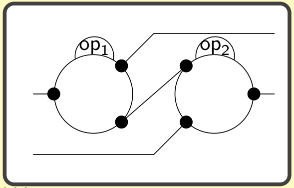
$$
\begin{array}{l}
\mathrm{comp\_op}\ \mathit{wire}\ \mathit{buf}\ \mathit{op}_1\ \mathit{op}_2 = \mathrm{Choice}\ (((\lambda \mathit{op}.\ \underline{\mathrm{case}}\ \mathit{op}\ \underline{\mathrm{of}} \\
\quad \mathrm{Read}\ p\ f\ \Rightarrow \mathrm{Read}\ (\mathrm{Inl}\ p)\ (\lambda x.\ \mathrm{comp\_op}\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



Composition: Equation

Equation of the composition operator

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



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

¡ :: $('i, 'o, 'd)\ op$

Sink operator type

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

Split operator type

Λ :: $('n, 'n + 'n, 'd)\ op$

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

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

¡ :: $('i, 'o, 'd)\ op$

Sink operator type

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

Split operator type

Λ :: $('n, 'n + 'n, 'd)\ op$

Merge operator type

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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: $\mathcal{X} \bullet \ogreaterthan \approx \ogreaterthan$ for $\ogreaterthan \in \{\mathcal{Q}, \mathcal{V}\}$ A6: $\olessthan \bullet \mathcal{X} \approx \olessthan$ 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: $\ogreaterthan \bullet ! \approx ! \parallel !$ for $\ogreaterthan \in \{\mathcal{Q}, \mathcal{V}\}$ A8: $\text{i} \bullet \olessthan \approx \text{i} \parallel \text{i}$ 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: $\text{map\_op id Inl}\ (\ogreaterthan :: (0 + 0, 0, 'd)\ op) \approx \mathcal{I}$ for $\ogreaterthan \in \{\mathcal{Q}, \mathcal{V}\}$

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

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

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

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

F4: $(\text{map\_op Inr id}\ \olessthan)\uparrow \approx \text{i}$ 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: $(\mathcal{Q} \parallel \mathcal{I}) \bullet \mathcal{Q} \approx \mathsf{map\_op} \curvearrowleft \mathsf{id} \ ((\mathcal{I} \parallel \mathcal{Q}) \bullet \mathcal{Q})$

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

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

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

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

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

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

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

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

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

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

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

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

A17: $! \approx ! \parallel !$

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

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

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

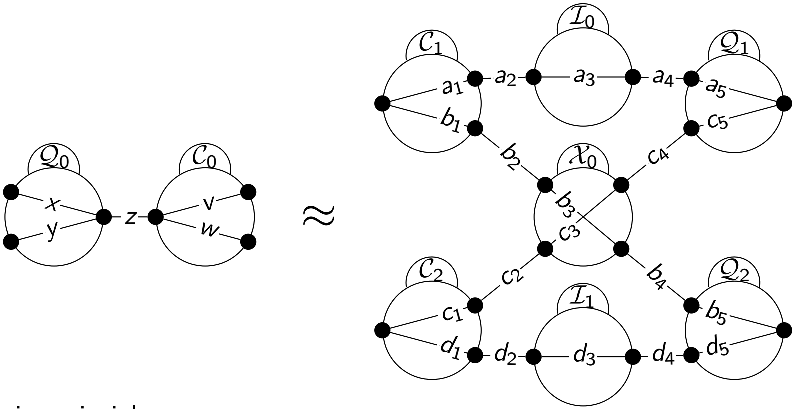
A19: $\circledless \approx (\circledless \parallel \circledless) \bullet \mathsf{map\_op} \curvearrowright \curvearrowright (\mathsf{map\_op} \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \quad \text{for } \circledless \in \{\mathcal{C}, \Lambda\}$

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

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

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

A10



- $\approx$ coinduction principle
- Buffers generalization
- Buffers invariant (details in the paper and artifact)

Related Work

Related Work

Related Work

- Bergstra et al formalization
  - Operators in a process algebra - we use Isabelle/HOL
  - We show formal proofs

Related Work

- Bergstra et al formalization
  - Operators in a process algebra - we use Isabelle/HOL
  - We show formal proofs
- Choice Trees by Chappe et al
  - ctrees share some similarities to our codatatype: coinductive trees, nondeterminism
  - Impure programs in general; our work is domain specific

Related Work

- Bergstra et al formalization
  - Operators in a process algebra - we use Isabelle/HOL
  - We show formal proofs
- Choice Trees by Chappe et al
  - ctrees share some similarities to our codatatype: coinductive trees, nondeterminism
  - Impure programs in general; our work is domain specific
- Flo by Laddad et al:
  - Formal representation for different streaming frameworks (Flink, Timely Dataflow)
  - Our language is Isabelle/HOL, streaming computations can be defined using our codatatype

Related Work

- Bergstra et al formalization
  - Operators in a process algebra - we use Isabelle/HOL
  - We show formal proofs
- Choice Trees by Chappe et al
  - ctrees share some similarities to our codatatype: coinductive trees, nondeterminism
  - Impure programs in general; our work is domain specific
- Flo by Laddad et al:
  - Formal representation for different streaming frameworks (Flink, Timely Dataflow)
  - Our language is Isabelle/HOL, streaming computations can be defined using our codatatype
- Vélus project
  - Verified Lustre compilation
  - Synchronous dataflow language

Conclusion

Conclusion

- An Isabelle/HOL instance of Nondeterministic Asynchronous Dataflow

Conclusion

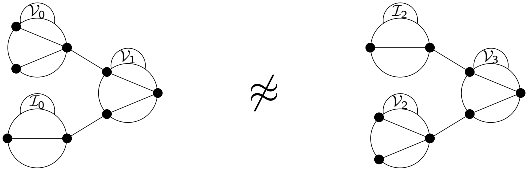
- An Isabelle/HOL instance of Nondeterministic Asynchronous Dataflow
- Isabelle/HOL has a good tool set:
    - Codatatypes, corecursion, coinductive predicates

Conclusion

- An Isabelle/HOL instance of Nondeterministic Asynchronous Dataflow
- Isabelle/HOL has a good tool set:
  - Codatatypes, corecursion, coinductive predicates
- 51/52 axioms proved

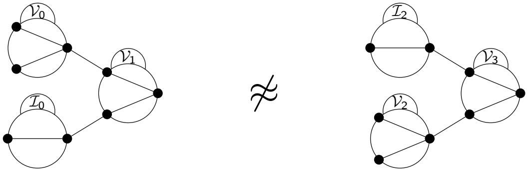
Conclusion

- 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 for us:



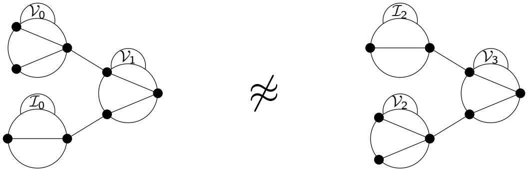
Conclusion

- 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 for us:



Conclusion

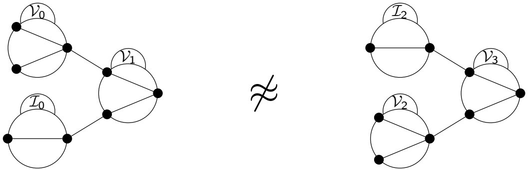
- An Isabelle/HOL instance of Nondeterministic Asynchronous Dataflow
- Isabelle/HOL has a good tool set:
  - Codatatypes, corecursion, coinductive predicates
- 51/52 axioms proved
  - Axiom A1 for the merge operator does not hold for us:



- 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 for us:



- Around 28 000 lines of definitions and proofs
- Future work:
  - Brock–Ackermann Time anomaly
  - Formalize Timely Dataflow infrastructure (finished)
  - Verify Timely Dataflow algorithms (on-going)

Questions, comments and suggestions