

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

Context:

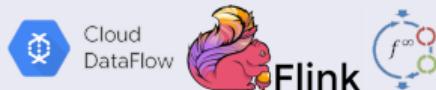
- Stream Processing: programs that compute (possibly) unbounded sequences of data (streams)
- Frameworks: Google Cloud Dataflow, Apache Flink, and Timely Dataflow



Motivation

Context:

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



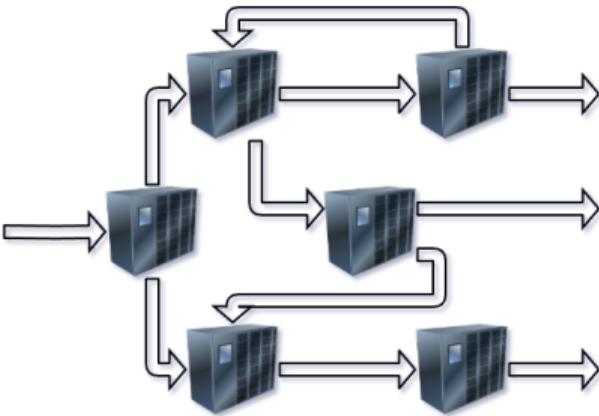
Our long term goal:

Mechanically Verify Timely Dataflow algorithms

A Good Foundation

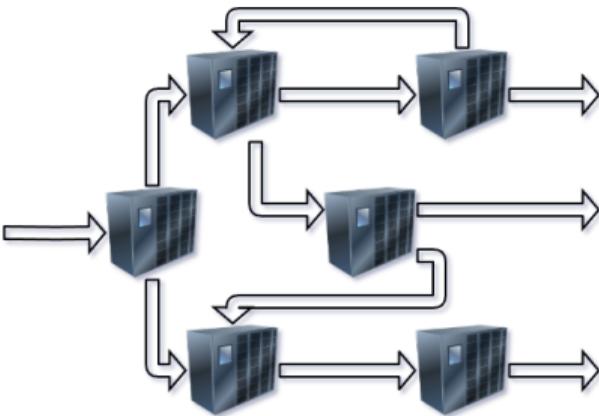
A Good Foundation

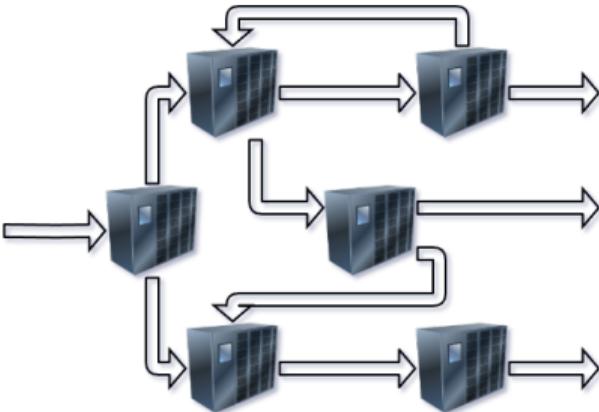
- Nondeterministic Asynchronous Dataflow



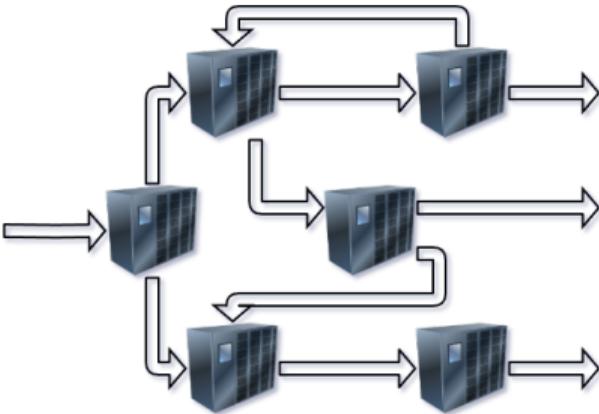
A Good Foundation

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



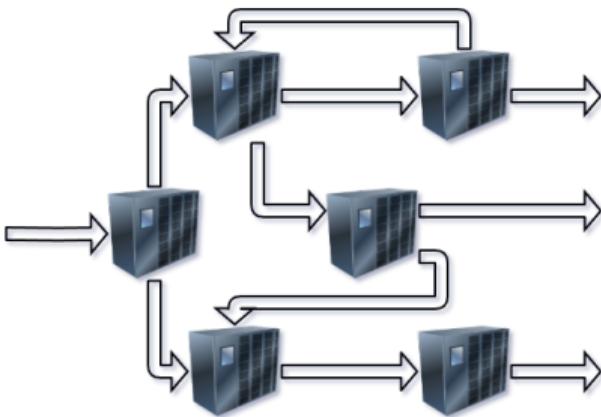


- Nondeterministic Asynchronous Dataflow
 - Dataflow: Directed graph of interconnected operators
 - Asynchronous:
 - Operators execute independently: processes without an orchestrator
 - Operators can freely communicate with the network (read/write); do silent computation steps
 - Networks are unbounded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



- Nondeterministic 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. Ștefănescu^{4,‡}

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

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

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

⁴Institute 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
- Executable via code extraction to Haskell

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

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

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

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

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

Uncommunicative operators

\emptyset = Choice $\{\}^c$

\odot = Silent \odot

\otimes = Choice $\{\otimes\}^c$

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

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

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

Uncommunicative operators

\emptyset = Choice $\{\}$ _c

\odot = Silent \odot

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

More examples

ex1 = Choice {Write ex1 1 42, \emptyset }_c

ex2 = Choice {Write ex2 1 42, ex2}_c

ex3 = Choice {Write ex3 1 42, 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)
- Labels (actions): Read, Write, Silent (τ)

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)
- Labels (actions): Read, Write, Silent (τ)
- Weak bisimilarity of operators (\approx): LTS + weak simulation + greatest weak bisimulation
- \approx has a useful coinduction principle
- $\emptyset \approx \otimes \approx \odot$ and $\text{ex1} \approx \text{ex2} \approx \text{ex3}$

Asynchronous Dataflow Operators

Auxiliary Definitions

Auxiliary Definitions

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

Buffer functions

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

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

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

Auxiliary Definitions

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

Buffer functions

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

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

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

choices type

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

Auxiliary Definitions

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

Buffer functions

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

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

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

choices type

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

Mapping ports functions

map_op :: $('i_1 \Rightarrow 'i_2) \Rightarrow ('o_1 \Rightarrow 'o_2) \Rightarrow ('i_1, 'o_1, 'd) \ op \Rightarrow ('i_2, 'o_2, 'd) \ op$

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

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

Equation of the identity operator

$\text{id_op } buf = \text{Choice}$

\cup_c

Equation of the identity operator

$\text{id_op } buf = \text{Choice}$

$((((\lambda p. \text{Read } p (\lambda x. \text{id_op} (\text{BENQ } p \times buf)))) \ `_c \ \mathfrak{U}_c)$
 \cup_c

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

Equation of the identity operator

$\text{id_op } buf = \text{Choice}$

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

\cup_c

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

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

Equation of the identity operator

$\text{id_op } buf = \text{Choice}$

$((\lambda p. \text{Read } p (\lambda x. \text{id_op} (\text{BENQ } p \times buf))) \ `_c \ \mathfrak{U}_c)$

\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

Equation of the identity operator

$\text{id_op } buf = \text{Choice}$

$((\lambda p. \text{Read } p (\lambda x. \text{id_op} (\text{BENQ } p \times buf))) \ `_c \ \mathfrak{U}_c)$

\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 delay: always can read, and it may eventually output

Identity operator with an empty buffer

$\mathcal{I} = \text{id_op} (\lambda_. \ [])$

Composition operator type

```
comp_op :: ('o1 ⇒ 'i2 option) ⇒ ('i2 ⇒ 'd list) ⇒ ('i1, 'o1, 'd) op ⇒ ('i2, 'o2, 'd) op ⇒  
      ('i1 + 'i2, 'o1 + 'o2, 'd) op
```

Composition: Preliminaries

Composition operator type

```
comp_op :: ('o1 ⇒ 'i2 option) ⇒ ('i2 ⇒ 'd list) ⇒ ('i1, 'o1, 'd) op ⇒ ('i2, 'o2, 'd) op ⇒ ('i1 + 'i2, 'o1 + 'o2, 'd) 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

`comp_op wire buf op1 op2 =`

Composition: Equation

Equation of the composition operator

`comp_op wire buf op1 op2 = Choice`

\cup_c

Composition: Equation

Equation of the composition operator

`comp_op wire buf op1 op2 = Choice`

$\backslash_c \text{ choices } op_1) \cup_c$

Composition: Equation

Equation of the composition operator

$\text{comp_op } \text{wire } \text{buf } \text{op}_1 \text{ op}_2 = \text{Choice } (((\lambda \text{op}. \underline{\text{case}} \text{ op } \underline{\text{of}} \text{ Read } p f \Rightarrow \text{Read} (\text{Inl } p) (\lambda x. \text{comp_op } \text{wire } \text{buf } (f x) \text{ op}_2)$

$\backslash_c \text{ choices } \text{op}_1) \cup_c$

Composition: Equation

Equation of the composition operator

$$\begin{aligned} \text{comp_op wire buf } op_1 \text{ } op_2 &= \text{Choice } (((\lambda op. \underline{\text{case}} \text{ } op \text{ } \underline{\text{of}}}) \\ &\quad \text{Read } p \text{ } f \Rightarrow \text{Read } (\text{Inl } p) (\lambda x. \text{comp_op wire buf } (f x) \text{ } op_2) \\ | \text{ Write } op \text{ } p \text{ } x &\Rightarrow (\underline{\text{case}} \text{ } \text{wire } p \text{ } \underline{\text{of}}) \\ &\quad \text{None} \Rightarrow \text{Write } (\text{comp_op wire buf } op \text{ } op_2) (\text{Inl } p) \text{ } x \\ | \text{ Some } q &\Rightarrow \text{Silent } (\text{comp_op wire } (\text{BENQ } q \text{ } x \text{ } \text{buf}) \text{ } op \text{ } op_2)) \\ &\quad \backslash_c \text{ choices } op_1) \cup_c \end{aligned}$$

Composition: Equation

Equation of the composition operator

$\text{comp_op wire buf } op_1 \ op_2 = \text{Choice (((}\lambda op. \underline{\text{case}} \ op \ \underline{\text{of}} \ \text{Read } p \ f \Rightarrow \text{Read} (\text{Inl } p) (\lambda x. \text{comp_op wire buf } (f \ x) \ op_2)$

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

 None $\Rightarrow \text{Write} (\text{comp_op wire buf } op \ op_2) (\text{Inl } p) \ x$

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

| Silent $op \Rightarrow \text{Silent} (\text{comp_op wire buf } op \ op_2)) \ \cup_c \ \text{choices } op_1 \ \cup_c$

Composition: Equation

Equation of the composition operator

$$\begin{aligned} \text{comp_op wire buf } op_1 \text{ } op_2 &= \text{Choice } (((\lambda op. \underline{\text{case}} \text{ } op \text{ } \underline{\text{of}}}) \\ &\quad \text{Read } p \text{ } f \Rightarrow \text{Read } (\text{Inl } p) (\lambda x. \text{comp_op wire buf } (f x) \text{ } op_2) \\ | \text{ Write } op \text{ } p \text{ } x &\Rightarrow (\underline{\text{case}} \text{ } \text{wire } p \text{ } \underline{\text{of}}) \\ &\quad \text{None} \Rightarrow \text{Write } (\text{comp_op wire buf } op \text{ } op_2) (\text{Inl } p) \text{ } x \\ | \text{ Some } q &\Rightarrow \text{Silent } (\text{comp_op wire } (\text{BENQ } q \text{ } x \text{ } \text{buf}) \text{ } op \text{ } op_2)) \\ | \text{ Silent } op &\Rightarrow \text{Silent } (\text{comp_op wire buf } op \text{ } op_2) \text{ } `_{\text{c}} \text{ choices } op_1 \cup_{\text{c}} \\ &\quad \text{Silent } op \text{ } op_2 \end{aligned}$$

(choices op_2)))

Composition: Equation

Equation of the composition operator

$$\begin{aligned} \text{comp_op wire buf } op_1 \text{ } op_2 &= \text{Choice (((}\lambda \text{op. } \underline{\text{case}} \text{ } op \text{ } \underline{\text{of}} \text{)}} \\ &\quad \text{Read } p \text{ } f \Rightarrow \text{Read (Inl } p) (\lambda x. \text{comp_op wire buf } (f \text{ } x) \text{ } op_2) \\ | \text{ Write } op \text{ } p \text{ } x &\Rightarrow (\underline{\text{case}} \text{ } \text{wire } p \text{ } \underline{\text{of}} \text{)}} \\ &\quad \text{None} \Rightarrow \text{Write (comp_op wire buf } op \text{ } op_2) (\text{Inl } p) \text{ } x \\ | \text{ Some } q &\Rightarrow \text{Silent (comp_op wire (BENQ } q \text{ } x \text{ } \text{buf}) \text{ } op \text{ } op_2))} \\ | \text{ Silent } op &\Rightarrow \text{Silent (comp_op wire buf } op \text{ } op_2)) \text{ } \backslash_c \text{ choices } op_1 \text{) } \cup_c \end{aligned}$$

$\backslash_c \text{ sound_reads wire buf (choices } op_2\text{))}$

Composition: Equation

Equation of the composition operator

`comp_op wire buf op1 op2` = Choice ((($\lambda op.$ case op of

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

| Write $op\ p\ x \Rightarrow (\text{case } wire\ p\ \text{ of}$

$\text{None} \Rightarrow \text{Write} (\text{comp_op } wire \text{ } buf \text{ } op \text{ } op_2) (\text{Inl } p) \text{ } x$

| Some $q \Rightarrow$ Silent (comp op wire (BENQ $q \times buf$) op op_2))

| Silent $op \Rightarrow$ Silent (comp_op wire buf op op_2)) `c choices op_1) \cup_c

((λ op. case op of

| Write $op\ p\ x \Rightarrow$ Write (comp op wire buf $op_1\ op$) (lnr $p\ x$)

`sound reads wire buf (choices op2))`

Composition: Equation

Equation of the composition operator

$\text{comp_op wire buf } op_1 \ op_2 = \text{Choice (((}\lambda op. \underline{\text{case}} \ op \ \underline{\text{of}}$

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

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

None $\Rightarrow \text{Write} (\text{comp_op wire buf } op \ op_2) (\text{Inl } p) \ x$

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

| Silent $op \Rightarrow \text{Silent} (\text{comp_op wire buf } op \ op_2) \ \backslash_c \ \text{choices } op_1 \cup_c$

(($\lambda op. \underline{\text{case}} \ op \ \underline{\text{of}}$

Read $p \ f \Rightarrow \underline{\text{if }} p \in \text{ran wire}$

then $\text{Silent} (\text{comp_op wire } (\text{BTL } p \ \text{buf}) \ op_1 (f (\text{BHD } p \ \text{buf})))$

else $\text{Read} (\text{Inr } p) (\lambda x. \text{comp_op wire buf } op_1 (f \ x))$

| Write $op \ p \ x \Rightarrow \text{Write} (\text{comp_op wire buf } op_1 \ op) (\text{Inr } p) \ x$

$\backslash_c \ \text{sound_reads wire buf } (\text{choices } op_2))$

Composition: Equation

Equation of the composition operator

$\text{comp_op wire buf } op_1 \ op_2 = \text{Choice (((}\lambda op. \underline{\text{case}} \ op \ \underline{\text{of}}$

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

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

None $\Rightarrow \text{Write} (\text{comp_op wire buf } op \ op_2) (\text{Inl } p) \ x$

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

| Silent $op \Rightarrow \text{Silent} (\text{comp_op wire buf } op \ op_2) \ `_c \ \text{choices } op_1 \cup_c$

(($\lambda op. \underline{\text{case}} \ op \ \underline{\text{of}}$

Read $p \ f \Rightarrow \underline{\text{if }} p \in \text{ran wire}$

then $\text{Silent} (\text{comp_op wire (BTL } p \ \text{buf)} \ op_1 (f (\text{BHD } p \ \text{buf})))$

else $\text{Read} (\text{Inr } p) (\lambda x. \text{comp_op wire buf } op_1 (f \ x))$

| Write $op \ p \ x \Rightarrow \text{Write} (\text{comp_op wire buf } op_1 \ op) (\text{Inr } p) \ x$

| Silent $op \Rightarrow \text{Silent} (\text{comp_op wire buf } op_1 \ op) \ `_c \ \text{sound_reads wire buf (choices } op_2))$

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

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

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

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

Sink operator type

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

Split operator type

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

Asynchronous Dataflow Operators

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

Feedback operator type ($op \uparrow$)

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

Transpose operator type

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

Dummy source operator type

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

Sink operator type

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

Split operator type

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

Merge operator type

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

Asynchronous Dataflow Operators

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

Feedback operator type ($op \uparrow$)

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

Transpose operator type

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

Dummy source operator type

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

Sink operator type

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

Split operator type

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

Merge operator type

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

Copy operator type

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

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

$$B1: op_1 \parallel (op_2 \parallel op_3) \approx \text{map_op} \curvearrowleft \curvearrowleft (op_1 \parallel op_2) \parallel op_3$$

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

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

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

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

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

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

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

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

$$B8: (\mathcal{X} :: ('i + 0, 0 + 'i, 'd) op) \approx \text{map_op} \text{id} (\text{case_sum} \text{Inr} \text{Inl}) \mathcal{I}$$

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

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

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

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

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

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

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

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

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

$$R6: (op \uparrow) \uparrow \approx (\text{map_op} \curvearrowleft \curvearrowleft op) \uparrow$$

Basic Network Algebra Properties

$$B1: op_1 \parallel (op_2 \parallel op_3) \approx map_op \curvearrowleft \curvearrowleft (op_1 \parallel op_2) \parallel op_3$$

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

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

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

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

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

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

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

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

$$B8: (\mathcal{X} :: ('i + 0, 0 + 'i, 'd) op) \approx map_op \text{id} (\text{case_sum Inr Inl}) \mathcal{I}$$

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

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

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

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

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

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

$$R3: op_1 \parallel (op_2 \uparrow) \approx (map_op \curvearrowleft \curvearrowleft (op_1 \parallel op_2)) \uparrow$$

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

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

$$R6: (op \uparrow) \uparrow \approx (map_op \curvearrowleft \curvearrowleft op) \uparrow$$

Basic Network Algebra Properties

$$B1: op_1 \parallel (op_2 \parallel op_3) \approx \text{map_op} \curvearrowleft \curvearrowleft (op_1 \parallel op_2) \parallel op_3$$

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

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

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

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

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

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

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

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

$$B8: (\mathcal{X} :: ('i + 0, 0 + 'i, 'd) op) \approx \text{map_op} \text{id} (\text{case_sum} \text{Inr} \text{Inl}) \mathcal{I}$$

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

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

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

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

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

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

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

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

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

$$R6: (op \uparrow) \uparrow \approx (\text{map_op} \curvearrowleft \curvearrowleft op) \uparrow$$

Basic Network Algebra Properties

$$B1: op_1 \parallel (op_2 \parallel op_3) \approx \text{map_op} \curvearrowleft \curvearrowleft (op_1 \parallel op_2) \parallel op_3$$

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

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

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

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

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

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

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

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

$$B8: (\mathcal{X} :: ('i + 0, 0 + 'i, 'd) op) \approx \text{map_op} \text{id} (\text{case_sum} \text{Inr} \text{Inl}) \mathcal{I}$$

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

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

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

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

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

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

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

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

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

$$R6: (op \uparrow) \uparrow \approx (\text{map_op} \curvearrowleft \curvearrowleft op) \uparrow$$

Basic Network Algebra Properties

$$B1: op_1 \parallel (op_2 \parallel op_3) \approx \text{map_op} \curvearrowleft \curvearrowleft (op_1 \parallel op_2) \parallel op_3$$

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

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

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

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

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

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

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

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

$$B8: (\mathcal{X} :: ('i + 0, 0 + 'i, 'd) op) \approx \text{map_op} \text{id} (\text{case_sum} \text{Inr} \text{Inl}) \mathcal{I}$$

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

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

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

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

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

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

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

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

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

$$R6: (op \uparrow) \uparrow \approx (\text{map_op} \curvearrowleft \curvearrowleft op) \uparrow$$

Basic Network Algebra Properties

$$B1: op_1 \parallel (op_2 \parallel op_3) \approx \text{map_op} \curvearrowleft \curvearrowleft (op_1 \parallel op_2) \parallel op_3$$

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

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

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

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

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

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

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

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

$$B8: (\mathcal{X} :: ('i + 0, 0 + 'i, 'd) op) \approx \text{map_op} \text{id} (\text{case_sum} \text{Inr} \text{Inl}) \mathcal{I}$$

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

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

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

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

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

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

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

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

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

$$R6: (op \uparrow) \uparrow \approx (\text{map_op} \curvearrowleft \curvearrowleft op) \uparrow$$

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

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

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

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

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

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

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

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

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

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

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

$$A13: i \approx i \parallel i$$

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

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

$$A15: \otimes \approx \text{map_op } \curvearrowleft \curvearrowleft \curvearrowleft (\text{map_op } \curvearrowleft \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \bullet (\otimes \parallel \otimes) \quad \text{for } \otimes \in \{\mathcal{Q}, \mathcal{V}\}$$

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

$$A19: \otimes \approx (\otimes \parallel \otimes) \bullet \text{map_op } \curvearrowleft \curvearrowleft \curvearrowleft (\text{map_op } \curvearrowleft \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \quad \text{for } \otimes \in \{\mathcal{C}, \Lambda\}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$A15: \otimes \approx \text{map_op } \curvearrowleft \curvearrowleft \curvearrowleft (\text{map_op } \curvearrowleft \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \bullet (\otimes \parallel \otimes) \quad \text{for } \otimes \in \{\mathcal{Q}, \mathcal{V}\}$$

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

$$A19: \otimes \approx (\otimes \parallel \otimes) \bullet \text{map_op } \curvearrowleft \curvearrowleft \curvearrowleft (\text{map_op } \curvearrowleft \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \quad \text{for } \otimes \in \{\mathcal{C}, \Lambda\}$$

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

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

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

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

$$A1: (\mathcal{Q} \parallel \mathcal{I}) \bullet \mathcal{Q} \approx \text{map_op } \curvearrowright \text{id } ((\mathcal{I} \parallel \mathcal{Q}) \bullet \mathcal{Q})$$

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

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

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

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

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

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

$$A5: \mathcal{C} \bullet (\mathcal{C} \parallel \mathcal{I}) \approx \text{map_op id } \curvearrowright (\mathcal{C} \bullet (\mathcal{I} \parallel \mathcal{C}))$$

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

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

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

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

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

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

$$A13: i \approx i \parallel i$$

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

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

$$A15: \otimes \approx \text{map_op } \curvearrowright \curvearrowright (\text{map_op } \curvearrowright \curvearrowright (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \bullet (\otimes \parallel \otimes) \quad \text{for } \otimes \in \{\mathcal{Q}, \mathcal{V}\}$$

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

$$A19: \otimes \approx (\otimes \parallel \otimes) \bullet \text{map_op } \curvearrowright \curvearrowright (\text{map_op } \curvearrowright \curvearrowright (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \quad \text{for } \otimes \in \{\mathcal{C}, \Lambda\}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$A15: \otimes \approx \text{map_op } \curvearrowleft \curvearrowleft \curvearrowleft (\text{map_op } \curvearrowleft \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \bullet (\otimes \parallel \otimes) \quad \text{for } \otimes \in \{\mathcal{Q}, \mathcal{V}\}$$

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

$$A19: \otimes \approx (\otimes \parallel \otimes) \bullet \text{map_op } \curvearrowleft \curvearrowleft \curvearrowleft (\text{map_op } \curvearrowleft \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \quad \text{for } \otimes \in \{\mathcal{C}, \Lambda\}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$A13: i \approx i \parallel i$$

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

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

$$A15: \otimes \approx \text{map_op } \curvearrowleft \curvearrowleft \curvearrowleft (\text{map_op } \curvearrowleft \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \bullet (\otimes \parallel \otimes) \quad \text{for } \otimes \in \{\mathcal{Q}, \mathcal{V}\}$$

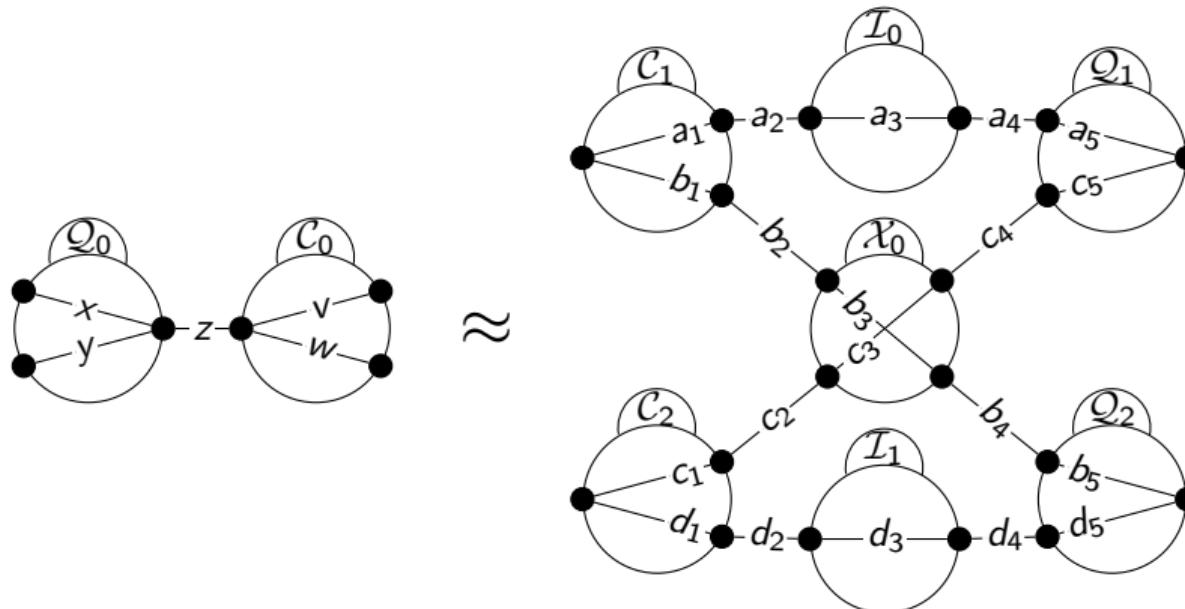
$$A18: \text{map_op Inl id } (\otimes :: (0, 0 + 0, 'd) op) \approx \mathcal{I} \quad \text{for } \otimes \in \{\mathcal{C}, \Lambda\}$$

$$A19: \otimes \approx (\otimes \parallel \otimes) \bullet \text{map_op } \curvearrowleft \curvearrowleft \curvearrowleft (\text{map_op } \curvearrowleft \curvearrowleft \curvearrowleft (\mathcal{I} \parallel \mathcal{X}) \parallel \mathcal{I}) \quad \text{for } \otimes \in \{\mathcal{C}, \Lambda\}$$

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

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

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



- \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. Ștefănescu^{4,||}

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

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

³Department of Network & Service Control, KPN Research
P.O. Box 421, 2200 AK Leidschendam, The Netherlands

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

E-mail: janb@fwi.uva.nl ~ keem@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. Ștefănescu^{4,||}

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

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

³Department of Network & Service Control, KPN Research
P.O. Box 421, 2200 AK Leidschendam, The Netherlands

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

E-mail: janb@fwi.uva.nl ~ keem@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, RedBull Systems, USA
BENJAMIN C. PIERCE, University of Pennsylvania, USA
STEVE ZDANCEWIC, University of Pennsylvania, USA

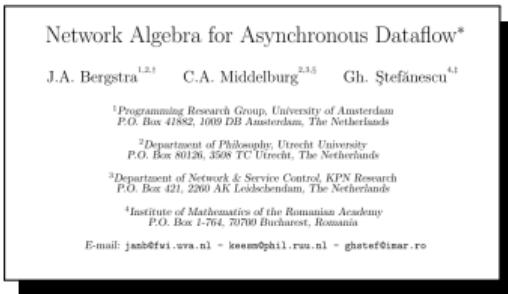
Choice Trees

Representing Nondeterministic, Recursive, and Impure Programs in Coq

NICOLAS CHAPPE, Univ Lyon, EnsL, LCB, 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



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

Interaction Trees

Representing Recursive and Impure Programs in Coq

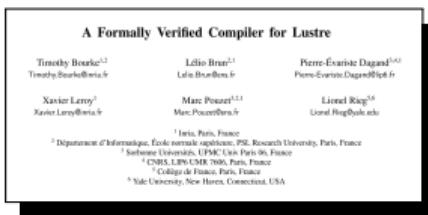
LI-YAO XIA, University of Pennsylvania, USA
YANNICK ZAKOWSKI, University of Pennsylvania, USA
PAUL HE, University of Pennsylvania, USA
CHUNG-KIL HUR, Seoul National University, Republic of Korea
GREGORY MALECHA, RedBull Systems, USA
BENJAMIN C. PIERCE, University of Pennsylvania, USA
STEVE ZDANCEWIC,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

Choice Trees

Representing Nondeterministic, Recursive, and Impure Programs in Coq

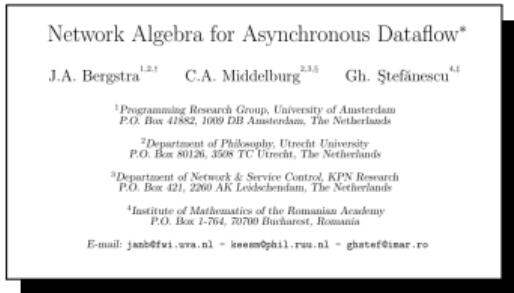
NICOLAS CHAPPE, Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France
PAUL HE, University of Pennsylvania, USA
LUDOVIC HENRIO, Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France
YANNICK ZAKOWSKI, Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France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

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

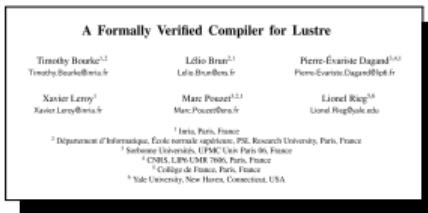


Related Work

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



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



- 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, RedRock 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

SHADA LAJDAD, UC Berkeley, USA
ALVIN CHERNIG, 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

- 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

- 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

- 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