

Computing with Isomorphisms

Roshan P. James and Amr Sabry

School of Informatics and Computing
Indiana University.
`rpjames@indiana.edu`

26th June, 2012

Quote

The Fabric of Reality, David Deutsch (1997):

*“Turing hoped that his **abstracted-paper-tape model** was so simple, so transparent and well defined, that it **would not depend on any assumptions about physics** that could conceivably be falsified, and therefore that it could become the basis of an abstract theory of computation that was independent of the underlying physics. ‘**He thought,**’ as Feynman once put it, ‘**that he understood paper.**’ But he was mistaken. Real, quantum-mechanical paper is wildly different from the abstract stuff that the Turing machine uses. The Turing machine is entirely classical...”*

Computing with the Physical World

- Quantum Physics suggests a universe in which
 - ▶ fundamental interactions are all **reversible**.
 - ▶ various quantities such as mass, energy, angular momentum, spin etc are **conserved**.

Computing with the Physical World

- Abstract models of computation **shield** us from the underlying technology that realize computation in the real world.
- Irreversible classical worldview.
 - ▶ **Logic gates** such as **nand** cannot recover inputs from outputs.
 - ▶ **Turing machines** rely on overwriting of symbols on a tape.
 - ▶ **λ -calculus** relies on β reduction.

Closed Systems and Interactions as Effects

- **Closed systems** are the basic notion and the basic unit of study in Physics.
 - ▶ Closed systems are reversible and obey various conservation laws.
- **Open systems**, ones that interact with their environment, are a derived notion.
 - ▶ Open systems do not share these properties.

Interactions as Effects

Interactions with the environment that change the properties of a closed system, are much like side effects .

Embedding Irreversibility within Reversibility

Toffoli (1980)

For every finite function $\phi : \text{bool}^m \rightarrow \text{bool}^n$ there exists an invertible finite function $\phi^R : \text{bool}^{r+m} \rightarrow \text{bool}^{r+m}$, with $r \leq n$, such that $\phi(x_1, \dots, x_m) = (y_1, \dots, y_n)$ iff

$$\phi^R(x_1, \dots, x_m, \overbrace{\text{false}, \dots, \text{false}}^r) = (\overbrace{\dots}^{m+r-n}, y_1, \dots, y_n)$$

Compiling AND

- Consider the irreversible function $\text{and} : \text{bool} \times \text{bool} \rightarrow \text{bool}$
- According to Toffoli (1980), “and” can be compiled to

<i>F</i>	<i>F</i>	F		F	F	<i>F</i>
<i>T</i>	<i>F</i>	F		T	F	<i>F</i>
<i>F</i>	<i>T</i>	F		F	T	<i>F</i>
<i>T</i>	<i>T</i>	F		T	T	<i>T</i>
F	F	T		F	F	T
T	F	T		T	F	T
F	T	T		F	T	T
T	T	T		T	T	F

Thesis: Information Effects

Thesis

By embodying irreversible physical primitives, conventional abstract models of computation have also inadvertently included some *implicit* computational effects, which we call *information effects*.

Thesis

By embodying irreversible physical primitives, conventional abstract models of computation have also inadvertently included some *implicit* computational effects, which we call *information effects*.

Technically:

- 1 A function is **information preserving** if its input entropy matches its output entropy.
- 2 **Logically reversible functions are information preserving.**
- 3 **Irreversibility** is a derived concept and **is captured by a type-and-effect system.**
 - ▶ Information effects are modeled as interactions with a surrounding information environment.

Contributions

- We develop such a “information preserving” reversible computational model called Π^o (also a strong normalizing fragment called Π).
- Reversibility is captured by (partial) type isomorphisms.
- Interactions with the heap and garbage are effects that are tracked by the type system.

Computing with Isomorphisms

Π^0 : a semantic foundation for computation

- Π^0 is based on
 - 1 type isomorphisms.
 - 2 trace operators from category theory.

Π^0 : a semantic foundation for computation

- Π^0 is based on
 - 1 type isomorphisms.
 - 2 trace operators from category theory.
- Information can **neither be created nor deleted** in Π^0 much like
 - ▶ Restrictions on weakening and contraction in Linear Logic
 - ▶ The no-cloning and no-deletion theorems of Quantum Mechanics.

Type Isomorphisms for Finite types

base types, b ::= 0 | 1 | *b* + *b* | *b* × *b*
values, v ::= () | *left v* | *right v* | (*v*, *v*)

Type Isomorphisms for Finite types

base types, b ::= 0 | 1 | $b + b$ | $b \times b$
values, v ::= () | *left* v | *right* v | (v, v)

$0 + b \Rightarrow b$ *identity for +*
 $b_1 + b_2 \Rightarrow b_2 + b_1$ *commutativity for +*
 $b_1 + (b_2 + b_3) \Rightarrow (b_1 + b_2) + b_3$ *associativity for +*

$1 \times b \Rightarrow b$ *identity for \times*
 $b_1 \times b_2 \Rightarrow b_2 \times b_1$ *commutativity for \times*
 $b_1 \times (b_2 \times b_3) \Rightarrow (b_1 \times b_2) \times b_3$ *associativity for \times*

$0 \times b \Rightarrow 0$ *distribute over 0*
 $(b_1 + b_2) \times b_3 \Rightarrow (b_1 \times b_3) + (b_2 \times b_3)$ *distribute over +*

Type Isomorphisms for Finite types

$\text{base types, } b ::= 0 \mid 1 \mid b + b \mid b \times b$
 $\text{values, } v ::= () \mid \text{left } v \mid \text{right } v \mid (v, v)$

$0 + b \Rightarrow b$ *identity for +*
 $b_1 + b_2 \Rightarrow b_2 + b_1$ *commutativity for +*
 $b_1 + (b_2 + b_3) \Rightarrow (b_1 + b_2) + b_3$ *associativity for +*

$1 \times b \Rightarrow b$ *identity for \times*
 $b_1 \times b_2 \Rightarrow b_2 \times b_1$ *commutativity for \times*
 $b_1 \times (b_2 \times b_3) \Rightarrow (b_1 \times b_2) \times b_3$ *associativity for \times*

$0 \times b \Rightarrow 0$ *distribute over 0*
 $(b_1 + b_2) \times b_3 \Rightarrow (b_1 \times b_3) + (b_2 \times b_3)$ *distribute over +*

$$\frac{}{b_1 \Rightarrow b_1} \quad \frac{b_1 \Rightarrow b_2}{b_2 \Rightarrow b_1} \quad \frac{b_1 \Rightarrow b_2 \quad b_2 \Rightarrow b_3}{b_1 \Rightarrow b_3}$$

$$\frac{b_1 \Rightarrow b_3 \quad b_2 \Rightarrow b_4}{(b_1 + b_2) \Rightarrow (b_3 + b_4)} \quad \frac{b_1 \Rightarrow b_3 \quad b_2 \Rightarrow b_4}{(b_1 \times b_2) \Rightarrow (b_3 \times b_4)}$$

Type Isomorphisms with Recursive Types and Trace

$\text{base types, } b ::= 0 \mid 1 \mid b + b \mid b \times b \mid \textcolor{red}{x} \mid \mu x. b$
 $\text{values, } v ::= () \mid \text{left } v \mid \text{right } v \mid (v, v) \mid \langle v \rangle$

$\mu x. b \quad \rightleftharpoons \quad b[\mu x. b / x]$ *isorecursive types*

$0 + b \quad \rightleftharpoons \quad b$ *identity for +*
 $b_1 + b_2 \quad \rightleftharpoons \quad b_2 + b_1$ *commutativity for +*
 $b_1 + (b_2 + b_3) \quad \rightleftharpoons \quad (b_1 + b_2) + b_3$ *associativity for +*

$1 \times b \quad \rightleftharpoons \quad b$ *identity for \times*
 $b_1 \times b_2 \quad \rightleftharpoons \quad b_2 \times b_1$ *commutativity for \times*
 $b_1 \times (b_2 \times b_3) \quad \rightleftharpoons \quad (b_1 \times b_2) \times b_3$ *associativity for \times*

$0 \times b \quad \rightleftharpoons \quad 0$ *distribute over 0*
 $(b_1 + b_2) \times b_3 \quad \rightleftharpoons \quad (b_1 \times b_3) + (b_2 \times b_3)$ *distribute over +*

$$\frac{}{b \rightleftharpoons b} \quad \frac{b_1 \rightleftharpoons b_2}{b_2 \rightleftharpoons b_1} \quad \frac{b_1 \rightleftharpoons b_2 \quad b_2 \rightleftharpoons b_3}{b_1 \rightleftharpoons b_3}$$

$$\frac{b_1 \rightleftharpoons b_3 \quad b_2 \rightleftharpoons b_4}{(b_1 + b_2) \rightleftharpoons (b_3 + b_4)} \quad \frac{b_1 \rightleftharpoons b_3 \quad b_2 \rightleftharpoons b_4}{(b_1 \times b_2) \rightleftharpoons (b_3 \times b_4)} \quad \frac{\textcolor{red}{b_1 + b_2} \rightleftharpoons \textcolor{red}{b_1 + b_3} \quad \textcolor{red}{b_2} \rightleftharpoons \textcolor{red}{b_3}}{\textcolor{red}{b_1 + b_2} \rightleftharpoons \textcolor{red}{b_1 + b_3}}$$

Witnesses for Type Isomorphisms with Trace : Π^0

$\text{base types, } b ::= 0 \mid 1 \mid b + b \mid b \times b \mid x \mid \mu x.b$
 $\text{values, } v ::= () \mid \text{left } v \mid \text{right } v \mid (v, v) \mid \langle v \rangle$

unfold : $\mu x.b \rightleftharpoons b[\mu x.b/x]$: *fold*

zeroe : $0 + b \rightleftharpoons b$: *zeroi*

swap⁺ : $b_1 + b_2 \rightleftharpoons b_2 + b_1$: *swap*⁺

assocl⁺ : $b_1 + (b_2 + b_3) \rightleftharpoons (b_1 + b_2) + b_3$: *assocr*⁺

unite : $1 \times b \rightleftharpoons b$: *uniti*

swap[×] : $b_1 \times b_2 \rightleftharpoons b_2 \times b_1$: *swap*[×]

assocl[×] : $b_1 \times (b_2 \times b_3) \rightleftharpoons (b_1 \times b_2) \times b_3$: *assocr*[×]

*distrib*₀ : $0 \times b \rightleftharpoons 0$: *factor*₀

distrib : $(b_1 + b_2) \times b_3 \rightleftharpoons (b_1 \times b_3) + (b_2 \times b_3)$: *factor*

$\frac{}{id : b \rightleftharpoons b}$ $\frac{c : b_1 \rightleftharpoons b_2}{sym\ c : b_2 \rightleftharpoons b_1}$ $\frac{c_1 : b_1 \rightleftharpoons b_2 \quad c_2 : b_2 \rightleftharpoons b_3}{(c_1 \circ c_2) : b_1 \rightleftharpoons b_3}$

$\frac{c_1 : b_1 \rightleftharpoons b_3 \quad c_2 : b_2 \rightleftharpoons b_4}{(c_1 + c_2) : (b_1 + b_2) \rightleftharpoons (b_3 + b_4)}$ $\frac{c_1 : b_1 \rightleftharpoons b_3 \quad c_2 : b_2 \rightleftharpoons b_4}{(c_1 \times c_2) : (b_1 \times b_2) \rightleftharpoons (b_3 \times b_4)}$ $\frac{c : b_1 + b_2 \rightleftharpoons b_1 + b_3}{trace\ c : b_2 \rightleftharpoons b_3}$

Π^0 examples : Booleans, Negation

base types, $b = 0 \mid 1 \mid b + b \mid b \times b \mid x \mid \mu x.b$

Π^0 examples : Booleans, Negation

base types, $b = 0 \mid 1 \mid b + b \mid b \times b \mid x \mid \mu x.b$

- We encode booleans in Π^0 by $bool = 1 + 1$

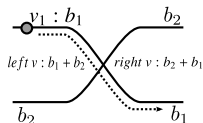
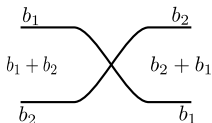
And thus we have

$true = left()$

$false = right()$

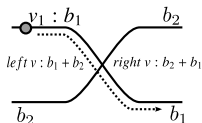
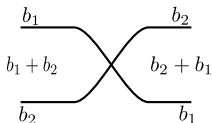
Π^0 examples : Booleans, Negation

$$\text{swap}^+ : b_1 + b_2 \rightleftharpoons b_2 + b_1$$



Π^0 examples : Booleans, Negation

$$\text{swap}^+ : b_1 + b_2 \rightleftharpoons b_2 + b_1$$



Thus

$$\text{not} : \text{bool} \rightleftharpoons \text{bool}$$

$$\text{not} = \text{swap}^+$$

and we can verify that:

$$\text{not true} \mapsto \text{false}$$

$$\text{not false} \mapsto \text{true}$$

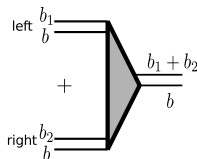
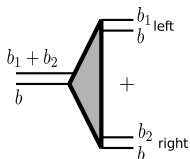
- Computation is modeled as the flow of particles in a circuit.
 - ▶ Geometry of Interaction
 - ▶ Proof Nets
 - ▶ Penrose diagrams for categories etc.

Π^0 examples : Conditionals

- $\mathbf{if}_c (flag, b) = \mathbf{if} (flag == true) \mathbf{then} (flag, c(b)) \mathbf{else} (flag, b)$

Π^0 examples : Conditionals

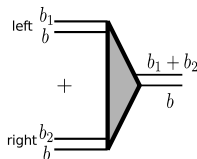
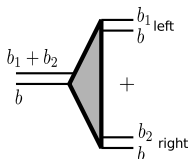
- $\text{if}_c (flag, b) = \text{if } (flag == \text{true}) \text{ then } (flag, c(b)) \text{ else } (flag, b)$



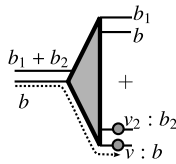
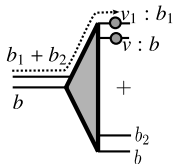
$$\text{distrib} : (b_1 + b_2) \times b \rightleftharpoons (b_1 \times b) + (b_2 \times b) : \text{factor}$$

Π^0 examples : Conditionals

- $\text{if}_c(\text{flag}, b) = \text{if}(\text{flag} == \text{true}) \text{ then } (\text{flag}, c(b)) \text{ else } (\text{flag}, b)$



$\text{distrib} : (b_1 + b_2) \times b \rightleftharpoons (b_1 \times b) + (b_2 \times b) : \text{factor}$

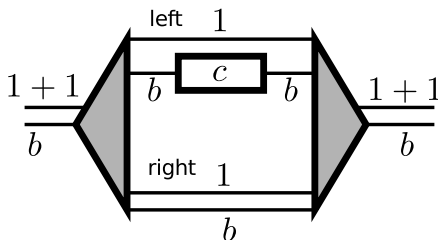


Π^0 examples : Conditionals

- $\mathbf{if}_c (flag, b) = \mathbf{if} (flag == true) \mathbf{then} (flag, c(b)) \mathbf{else} (flag, b)$

Π^0 examples : Conditionals

- $\mathbf{if}_c (flag, b) = \mathbf{if} (flag == true) \mathbf{then} (flag, c(b)) \mathbf{else} (flag, b)$
- $\mathbf{if}_c : bool \times b \rightleftharpoons bool \times b$
 $\mathbf{if}_c = distrib \circ ((id \times c) + id) \circ factor$

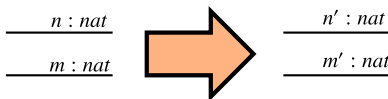


Π^0 examples : Abstract Machines

Numbers, n, m = $0 \mid n + 1$
Machine states = $\langle n, n \rangle$

Start state = $\langle n, 0 \rangle$
Stop State = $\langle 0, n \rangle$

$$\langle n + 1, m \rangle \mapsto \langle n, m + 1 \rangle$$



Π^0 examples : Abstract Machines

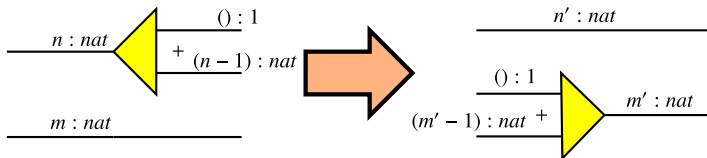
Numbers, n, m = $0 \mid n + 1$

Machine states = $\langle n, n \rangle$

Start state = $\langle n, 0 \rangle$

Stop State = $\langle 0, n \rangle$

$$\langle n + 1, m \rangle \mapsto \langle n, m + 1 \rangle$$



$$\begin{aligned} fold_{nat} : 1 + nat &\rightleftharpoons nat : unfold_{nat} \\ nat &= \mu x. (1 + x) \end{aligned}$$

Π^0 examples : Abstract Machines

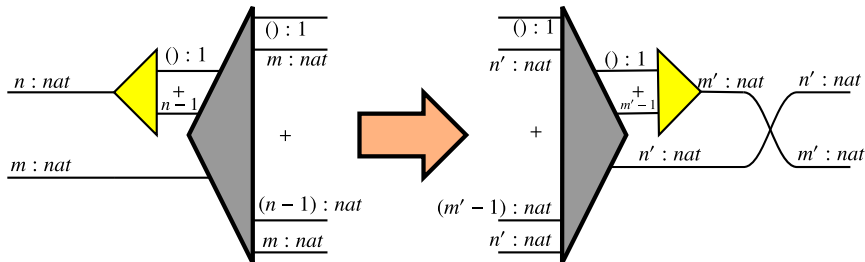
Numbers, $n, m = 0 \mid n + 1$

Machine states = $\langle n, n \rangle$

Start state = $\langle n, 0 \rangle$

Stop State = $\langle 0, n \rangle$

$$\langle n + 1, m \rangle \mapsto \langle n, m + 1 \rangle$$



Π^0 examples : Abstract Machines

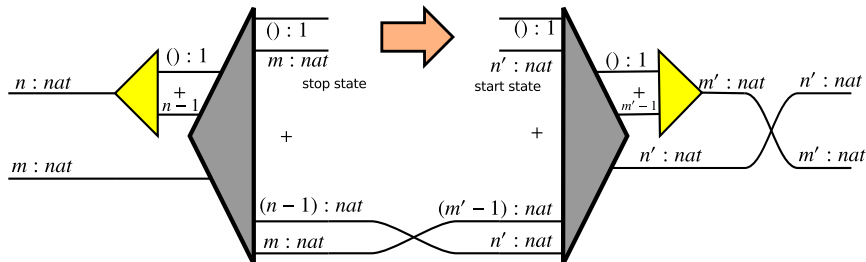
Numbers, $n, m = 0 \mid n + 1$

Machine states = $\langle n, n \rangle$

Start state = $\langle n, 0 \rangle$

Stop State = $\langle 0, n \rangle$

$$\langle n + 1, m \rangle \mapsto \langle n, m + 1 \rangle$$



Π^0 examples : Abstract Machines

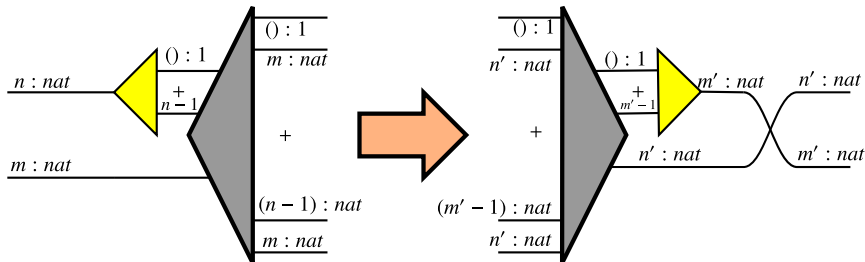
Numbers, $n, m = 0 \mid n + 1$

Machine states = $\langle n, n \rangle$

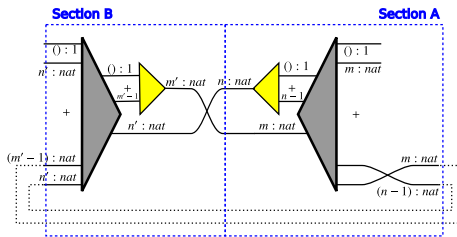
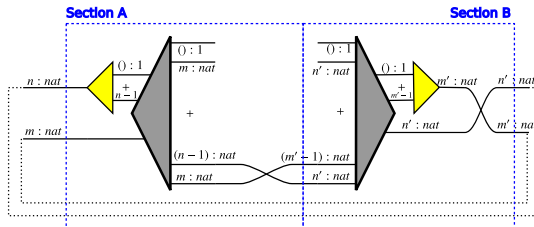
Start state = $\langle n, 0 \rangle$

Stop State = $\langle 0, n \rangle$

$$\langle n + 1, m \rangle \mapsto \langle n, m + 1 \rangle$$



Π^0 examples : Abstract Machines



Π^0 examples : Abstract Machines

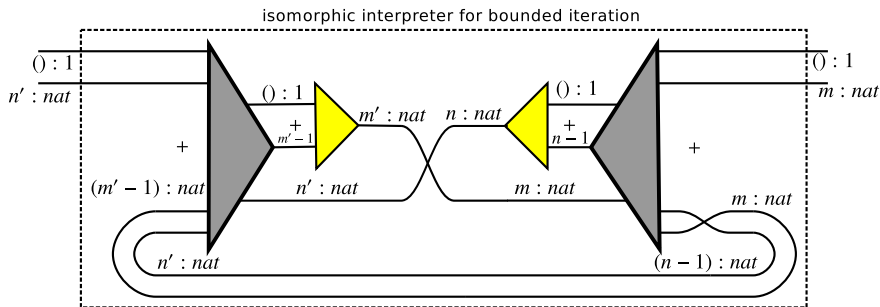
Numbers, $n, m = 0 \mid n + 1$

Machine states = $\langle n, n \rangle$

Start state = $\langle n, 0 \rangle$

Stop State = $\langle 0, n \rangle$

$$\langle n + 1, m \rangle \mapsto \langle n, m + 1 \rangle$$



Π^0 examples : Metacircular Evaluator

Combinators, c = $iso \mid c \circ c \mid c \times c \mid c + c \mid trace \ c$
Combinator Contexts, cc = $\square \mid Fst \ cc \ c \mid Snd \ c \ cc$
 $\mid LeftTimes \ cc \ c \ v \mid RightTimes \ c \ v \ cc$
 $\mid LeftPlus \ cc \ c \mid RightPlus \ c \ cc \mid Trace \ cc$
Values, v = $() \mid (v, v) \mid L \ v \mid R \ v$

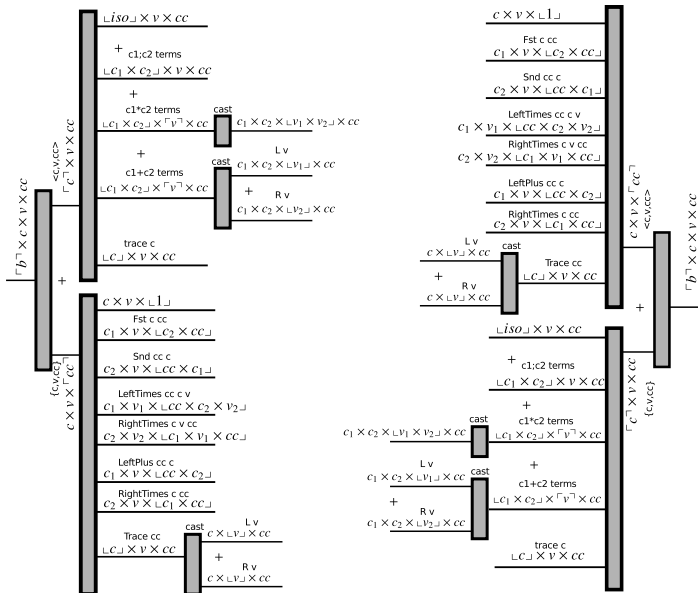
Machine states = $\langle c, v, cc \rangle \mid \{c, v, cc\}$

Start state = $\langle c, v, \square \rangle$

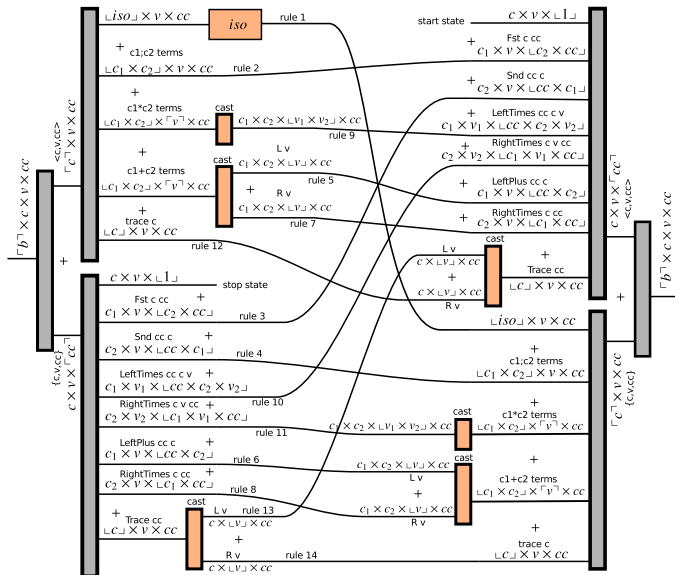
Stop State = $\{c, v, \square\}$

$\langle iso, v, cc \rangle$	$\mapsto \{iso, iso(v), cc\}$	rule 1
$\langle c_1 \circ c_2, v, cc \rangle$	$\mapsto \langle c_1, v, Fst \ cc \ c_2 \rangle$	rule 2
$\{c_1, v, Fst \ cc \ c_2\}$	$\mapsto \langle c_2, v, Snd \ c_1 \ cc \rangle$	rule 3
$\{c_2, v, Snd \ c_1 \ cc\}$	$\mapsto \langle c_1 \circ c_2, v, cc \rangle$	rule 4
$\langle c_1 + c_2, L \ v, cc \rangle$	$\mapsto \langle c_1, v, LeftPlus \ cc \ c_2 \rangle$	rule 5
$\{c_1, v, LeftPlus \ cc \ c_2\}$	$\mapsto \{c_1 + c_2, L \ v, cc\}$	rule 6
$\langle c_1 + c_2, R \ v, cc \rangle$	$\mapsto \langle c_2, v, RightPlus \ c_1 \ cc \rangle$	rule 7
$\{c_2, v, RightPlus \ c_1 \ cc\}$	$\mapsto \{c_1 + c_2, R \ v, cc\}$	rule 8
$\langle c_1 \times c_2, (v_1, v_2), cc \rangle$	$\mapsto \langle c_1, v_1, LeftTimes \ cc \ c_2 \ v_2 \rangle$	rule 9
$\{c_1, v_1, LeftTimes \ cc \ c_2 \ v_2\}$	$\mapsto \langle c_2, v_2, RightTimes \ c_1 \ v_1 \ cc \rangle$	rule 10
$\{c_2, v_2, RightTimes \ c_1 \ v_1 \ cc\}$	$\mapsto \langle c_1 \times c_2, (v_1, v_2), cc \rangle$	rule 11
$\langle trace \ c, v, cc \rangle$	$\mapsto \langle c, R \ v, Trace \ cc \rangle$	rule 12
$\{c, L \ v, Trace \ cc\}$	$\mapsto \langle c, L \ v, Trace \ cc \rangle$	rule 13
$\{c, R \ v, Trace \ cc\}$	$\mapsto \{trace \ c, R \ v, cc\}$	rule 14

Π^0 examples : Metacircular Evaluator



Π^0 examples : Metacircular Evaluator



Extending Π^o with Information Effects

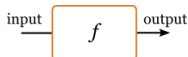
Arrows

Information effects are captured using an **arrow-metalanguage over Π^0** — similar to expressing computational effects over λ -calculus using arrows or monads.

Arrows

Information effects are captured using an **arrow-metalanguage over Π^0** — similar to expressing computational effects over λ -calculus using arrows or monads.

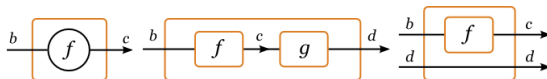
Computations are $f : \text{input} \rightsquigarrow \text{output}$



$$\text{arr} : (b \rightarrow c) \rightarrow (b \rightsquigarrow c)$$

$$\gg : (b \rightsquigarrow c) \rightarrow (c \rightsquigarrow d) \rightarrow (b \rightsquigarrow d)$$

$$\text{first} : (a \rightsquigarrow b) \rightarrow ((b, d) \rightsquigarrow (c, d))$$



We will lift sequencing, parallel composition and trace.

Language of Information Effects : ML_{Π^0}

- 1 ML_{Π^0} extends Π^0 with the arrow type $a : b_1 \rightarrow b_2$.

Language of Information Effects : ML_{Π^0}

- 1 ML_{Π^0} extends Π^0 with the arrow type $a : b_1 \multimap b_2$.
- 2 Lifts traces, sequencing and parallel composition to the new arrow type $b_1 \multimap b_2$.

Language of Information Effects : ML_{Π^0}

- 1 ML_{Π^0} extends Π^0 with the arrow type $a : b_1 \rightarrow b_2$.
- 2 Lifts traces, sequencing and parallel composition to the new arrow type $b_1 \rightarrow b_2$.
- 3 And adds two information effects representing the creation and erasure of information.
 - ▶ *create* : $1 \rightarrow b$
 - ▶ *erase* : $b \rightarrow 1$
- 4 Given the ability to *create* constants, we can construct a *clone* operator.

Entropy Analysis Example : ML_{Π^o}

XOR and NAND example : entropy analysis

$xor(b_1, b_2) = \text{if } b_1 \text{ then (if } b_2 \text{ then } false \text{ else } true) \text{ else } b_2$

$nand(b_1, b_2) = \text{if } b_1 \text{ then (if } b_2 \text{ then } false \text{ else } true) \text{ else } true$

XOR and NAND example : entropy analysis

$xor(b_1, b_2) = \text{if } b_1 \text{ then (if } b_2 \text{ then false else true) else } b_2$

$nand(b_1, b_2) = \text{if } b_1 \text{ then (if } b_2 \text{ then false else true) else true}$

- Entropy of a function is $H_i - H_o$
- Entropy of input, H_i , for $bool \times bool$ is 2 bits.

XOR and NAND example : entropy analysis

$xor(b_1, b_2) = \text{if } b_1 \text{ then (if } b_2 \text{ then false else true) else } b_2$

$nand(b_1, b_2) = \text{if } b_1 \text{ then (if } b_2 \text{ then false else true) else true}$

- Entropy of a function is $H_i - H_o$
- Entropy of input, H_i , for $bool \times bool$ is 2 bits.
- For xor ,
 - ▶ Probability of output being true, $P(true) = 1/2$
 - ▶ and $P(false) = 1/2$.
 - ▶ Therefore $H_o = -\sum p_i \log p_i = 1$ bits.
 - ▶ Information lost is $2 - 1 = 1$ bit

XOR and NAND example : entropy analysis

$xor(b_1, b_2) = \text{if } b_1 \text{ then (if } b_2 \text{ then false else true) else } b_2$

$nand(b_1, b_2) = \text{if } b_1 \text{ then (if } b_2 \text{ then false else true) else true}$

- Entropy of a function is $H_i - H_o$
- Entropy of input, H_i , for $bool \times bool$ is 2 bits.
- For *xor*,
 - ▶ Probability of output being true, $P(true) = 1/2$
 - ▶ and $P(false) = 1/2$.
 - ▶ Therefore $H_o = -\sum p_i \log p_i = 1$ bits.
 - ▶ Information lost is $2 - 1 = 1$ bit
- For *nand*,
 - ▶ Probability of output being true, $P(true) = 3/4$
 - ▶ and $P(false) = 1/4$.
 - ▶ Therefore $H_o = -\sum p_i \log p_i = 1/4 \log 4 + 3/4(\log 4 - \log 3) = 0.8$ bits.
 - ▶ Information lost is $2 - 0.8 = 1.2$ bits

XOR and NAND example : ML_{Π^0} implementation

- The most optimal implementation of *xor* must erase at least 1 bit i.e. there must be at least one *erase_{bool}*.

```
xor : bool × bool → bool  
xor = distrib >>> (not ⊕ id)  
      >>> factor >>> (erasebool ⊗ id) >>> arr unite
```

- The most optimal implementation of *nand* must erase at least 1.2 bits i.e. at least two *bools* must be erased.

```
nand : bool × bool → bool  
nand = distrib >>> (not ⊕ (erasebool >>> createtrue))  
      >>> factor >>> (erasebool ⊗ id) >>> arr unite
```

- These minimums are captured in the structure of the program.

$$\text{LET}^o \implies \text{ML}_{\Pi^o} \implies \Pi^o$$

Translation 1: $\text{LET}^o \Longrightarrow \text{ML}_{\Pi^o}$

We show the translation from a first-order λ^\rightarrow , with sum and product types, and extended with iteration to ML_{Π^o} .

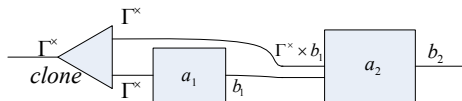
$$\Gamma \vdash e : b \Longrightarrow a : \Gamma^\times \multimap b$$

- The translation exposes the implicit irreversibility of the λ -calculus by requiring explicit *create* and *erase* operations.

Translation 1 : overview

- Case *let* $x = e_1$ in e_2 :

$$\frac{\begin{array}{l} \Gamma \vdash e_1 : b_1 \implies a_1 : \Gamma^\times \multimap b_1 \\ \Gamma, x : b_1 \vdash e_2 : b_2 \implies a_2 : \Gamma^\times \times b_1 \multimap b_2 \end{array}}{\Gamma \vdash \text{let } x = e_1 \text{ in } e_2 : b_2 \implies a : \Gamma^\times \multimap b_2}$$



Translation 1 : overview

- The main work of the translation is to
 - ▶ Make the implicit environment of LET^0 explicit as the Γ^\times input.
 - ▶ Thread the environment through the computation.
 - ▶ Clone/introduce constants as required.
 - ▶ Erase unwanted values.
- Much of the complexity lies in the handling of sums and case.
- The product fragment of this translation is much like the *CCC* semantics for λ^\rightarrow .

Translation 2: $\text{ML}_{\Pi^0} \Longrightarrow \Pi^0$

ML_{Π^0} can be embedded in Π^0 .

$$a : \Gamma^\times \multimap b \Longrightarrow c : h \times \Gamma^\times \Rrightarrow g \times b$$

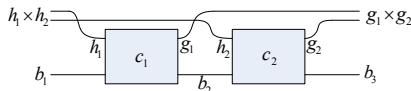
- Information effects manifest as interactions with an information environment.
 - ▶ The artifacts h and g , exposed in the types, represent this information environment.

Translation 2 : overview

- $a_1 \ggg a_2$:

$$\frac{\begin{array}{l} a_1 : b_1 \rightarrow b_2 \implies c_1 : h_1 \times b_1 \Rrightarrow g_1 \times b_2 \\ a_2 : b_2 \rightarrow b_3 \implies c_2 : h_2 \times b_2 \Rrightarrow g_2 \times b_3 \end{array}}{a_1 \ggg a_2 : b_1 \rightarrow b_3 \implies c : h \times b_1 \Rrightarrow g \times b_3}$$

We choose $h = h_1 \times h_2$ and $g = g_1 \times g_2$ and we have



Translation 2 : overview

- The main work of the translation is to shuffle h and g values through the computation.
- In the simplest sense, *create* exposes the input heap.

$$\overline{\text{create} : 1 \multimap b \Longrightarrow c : h \times 1 \Rrightarrow g \times b}$$

We choose $h = b$ and $g = 1$ and we have $c = \text{swap}^\times$.

- The operator *erase* does the dual and is also realized by swap^\times .

Note: The paper refines this further and gives two different treatments of *create*, one for Π° and one for its strong-normalizing fragment Π .

Applications and Connections

- Could serve as better basis of computation than λ -calculus for applications where information manipulation is computationally significant.
 - ▶ Quantitative information-flow security. Sabelfeld and Myers (2003)
 - ▶ Differential privacy. Dwork (2006)
 - ▶ Energy-aware computing. Ma et al. (2008); Zeng et al. (2002)
 - ▶ VLSI design. Macii and Poncino (1996)
 - ▶ Biochemical models of computation. Cardelli and Zavattaro (2008)
- Hiding in the structure of Π^o is a Dagger Symmetric Traced Monoidal Category.
 - ▶ Gol. Girard (1989)
 - ▶ Duality of Computation. Filinski (1989), Curien and Herbelin (2000)
 - ▶ Quantum Computing.
 - ▶ Categorical constructions such as \mathcal{G} and **Int**.

The Dualities of Computation

Extensions: Duality

- How do we express functions in Π^0 ?
- How do we express continuations in Π^0 ?

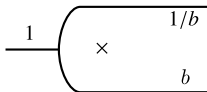
Extensions: Duality

Not one duality, but two: negative and fractional types.

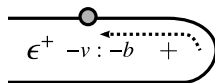
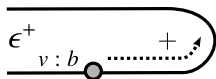
$$\eta^+ : 0 \leftrightarrow (-b) + b : \epsilon^+$$

$$\eta^\times : 1 \leftrightarrow (1/b) \times b : \epsilon^\times$$

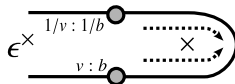
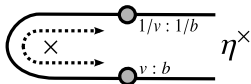
$$\frac{\vdash v : b}{\vdash -v : -b} \quad \frac{\vdash v : b}{\vdash 1/v : 1/b}$$



Extensions: $\Pi^{\eta\epsilon}$



Backward Information Flow.



Creation and Annihilation of entangled particles.

In many ways, we are only beginning to understand these structures.

Extras

The language Π^o

Syntax of Π^o

base types, b ::= $0 \mid 1 \mid b + b \mid b \times b \mid x \mid \mu x.b$

combinator types, t ::= $b \rightleftharpoons b$

values, v ::= $() \mid \text{left } v \mid \text{right } v \mid (v, v) \mid \langle v \rangle$

isomorphisms, iso ::= $\text{swap}^+ \mid \text{assocl}^+ \mid \text{assocr}^+$
| $\text{swap}^\times \mid \text{assocl}^\times \mid \text{assocr}^\times \mid \text{unite} \mid \text{uniti}$
| $\text{distrib} \mid \text{factor} \mid \text{id} \mid \text{fold} \mid \text{unfold}$

combinators, c ::= $iso \mid \text{sym } c \mid c \circ c \mid c \times c \mid c + c \mid \text{trace } c$

- Π is the fragment of Π^o sans recursive types and trace.
 - ▶ i.e. without the red parts.

Logically reversible functions are “information preserving”

- 1 **Logical reversibility**: A function $f : b_1 \rightarrow b_2$ is logically reversible if there exists an inverse function $g : b_2 \rightarrow b_1$ such that for all values $v_1 \in b_1$ and $v_2 \in b_2$, we have: $f(v_1) = v_2$ iff $g(v_2) = v_1$. (Zuliani (2001))
- 2 **Entropy of a variable** : Let ‘ b ’ be a (not necessarily finite) type whose values are labeled b^1, b^2, \dots . Let ξ be a random variable of type b that is equal to b^i with probability p_i . The entropy of ξ is defined as $-\sum p_i \log p_i$.
- 3 **Output entropy of a function**: Consider a function $f : b_1 \rightarrow b_2$ where b_2 is a (not necessarily finite) type whose values are labeled b_2^1, b_2^2, \dots . The output entropy of the function is given by $-\sum q_j \log q_j$ where q_j indicates the probability of the output of the function to have value b_2^j .
- 4 **Information Preservation**: We say a function is *information-preserving* if its output entropy is equal to the entropy of its input.
- 5 **Non-termination is not an observable**.

We extend Toffoli's result in several ways

- Richer types, not truth tables. (Full sum and product types)
- Term language for reversible and irreversible computation.
- Type directed translation.
- Deal with infinite functions (ex. over nats).

References

- Luca Cardelli and Gianluigi Zavattaro. On the computational power of biochemistry. In *Third International Conference on Algebraic Biology*, 2008.
- Pierre-Louis Curien and Hugo Herbelin. The duality of computation. In *ICFP*, pages 233–243, New York, NY, USA, 2000. ACM.
- Cynthia Dwork. Differential privacy. In *ICALP (2)'06*, pages 1–12, 2006.
- Andrzej Filinski. Declarative continuations: an investigation of duality in programming language semantics. In *Category Theory and Computer Science*, pages 224–249, London, UK, 1989. Springer-Verlag.
- J.Y. Girard. Geometry of interaction 1: Interpretation of system f. *Studies in Logic and the Foundations of Mathematics*, 127:221–260, 1989.
- Xiaojun Ma, Jing Huang, and Fabrizio Lombardi. A model for computing and energy dissipation of molecular QCA devices and circuits. *J. Emerg. Technol. Comput. Syst.*, 3(4):1–30, 2008.
- Enrico Macii and Massimo Poncino. Exact computation of the entropy of a logic circuit. In *Proceedings of the 6th Great Lakes Symposium on VLSI*, Washington, DC, USA, 1996. IEEE Computer Society.
- Andrei Sabelfeld and Andrew Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003.
- Tommaso Toffoli. Reversible computing. In *Proceedings of the Colloquium on Automata, Languages and Programming*, pages 632–644. Springer-Verlag, 1980.
- Heng Zeng, Carla S. Ellis, Alvin R. Lebeck, and Amin Vahdat. Ecosystem: managing energy as a first class operating system resource. *SIGPLAN Not.*, 37(10):123–132, 2002.
- P. Zuliani. Logical reversibility. *IBM J. Res. Dev.*, 45:807–818, November 2001.