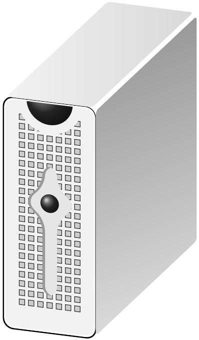
# Reversing Your Computation, but Why?

## Meeting on Foundations of Security and Concurrency

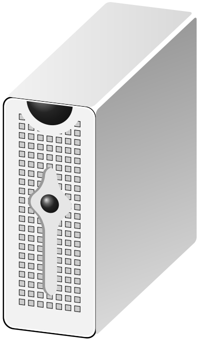
Clément Aubert

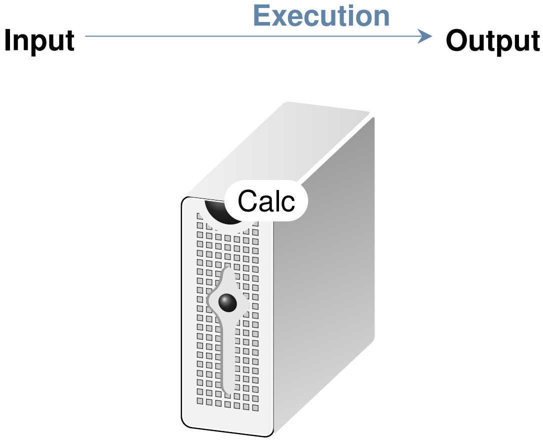
Augusta University, School of Computer & Cyber Sciences, GA, USA

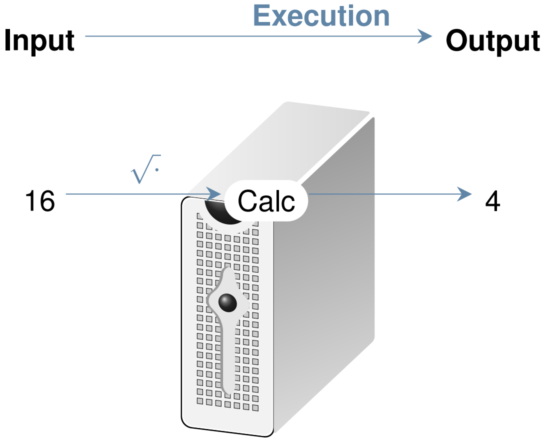
July 4th, 2024

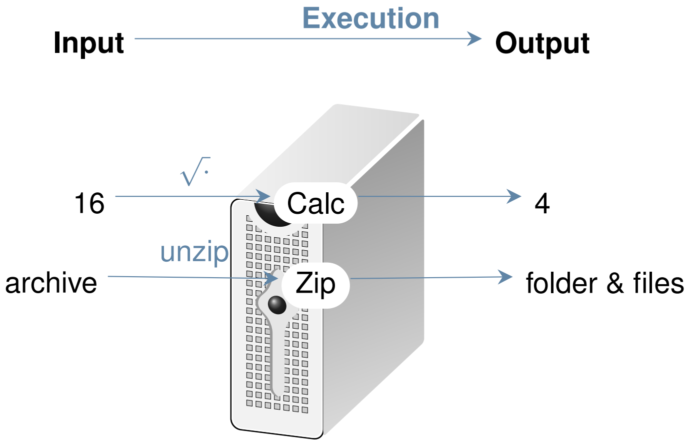


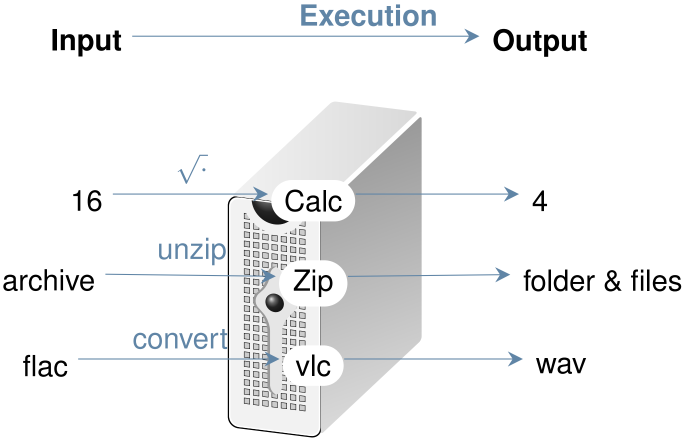
**Input** → **Execution** → **Output**

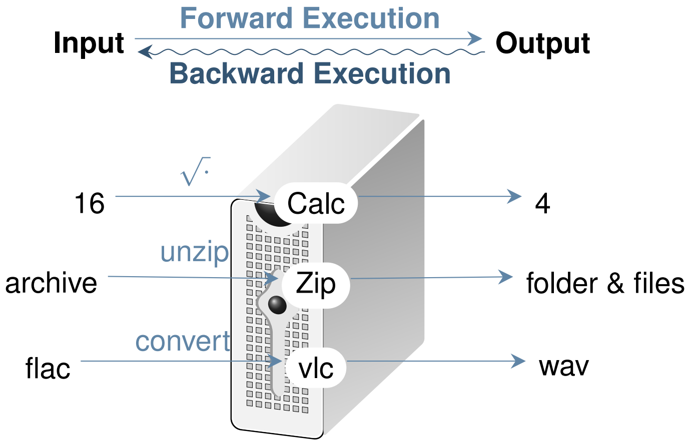


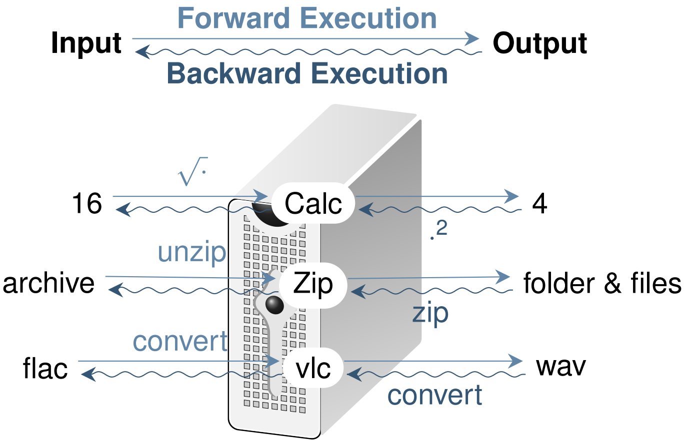

Execution
Input
Output
Calc

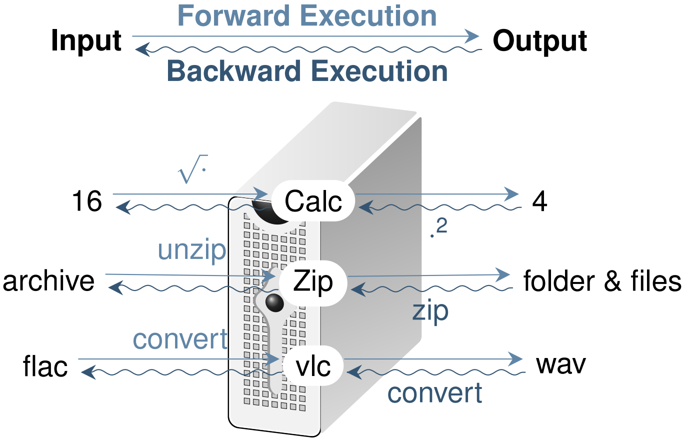

Execution
Input
Output
$\sqrt{\cdot}$
16
Calc
4


Execution
Input
Output
$\sqrt{\cdot}$
16
Calc
4
unzip
archive
Zip
folder & files


Execution
Input
Output
$\sqrt{\cdot}$
16
Calc
4
unzip
archive
Zip
folder & files
convert
flac
vlc
wav

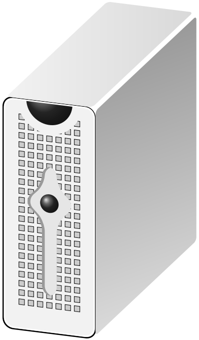

Input
Forward Execution
Backward Execution
Output
16
√·
Calc
4
unzip
archive
Zip
folder & files
convert
flac
vlc
wav


Forward Execution
Input
Output
Backward Execution
√·
16
Calc
4
.2
unzip
archive
Zip
folder & files
zip
convert
flac
vlc
wav
convert

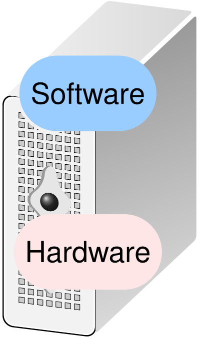


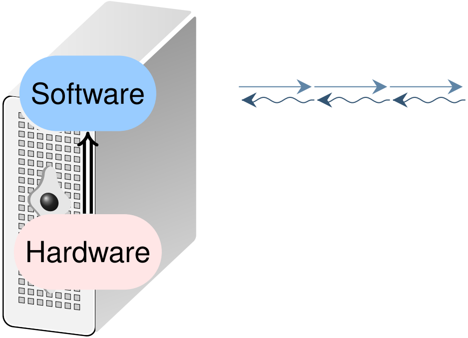

## First observation

We reverse computations (“uncompute”) frequently actually!



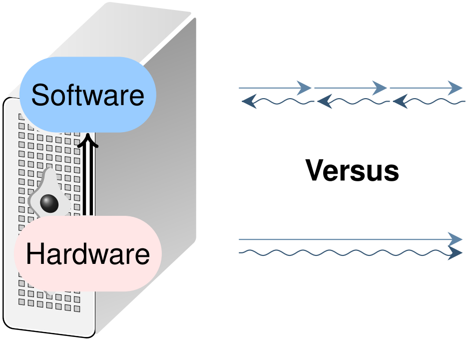
# Reversing Your Computation, but Why?– What is “reversible”?






## Observation

Reversible hardware can only execute reversible software.




## Observations

Reversible hardware can only execute reversible software.
Reversible programs can be compiled into two (forward-only) programs.

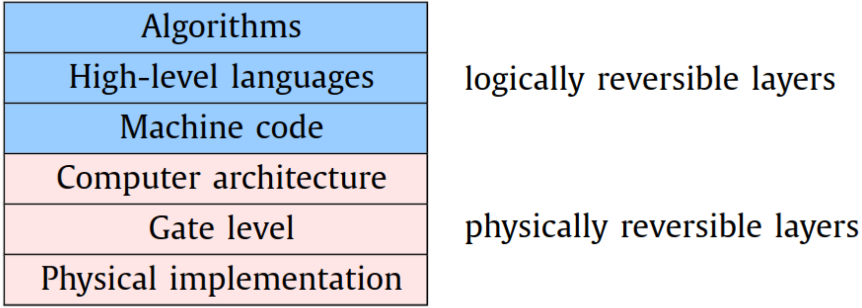



**Fig. 2.** The hardware and software stack of a reversible computing system.

Obs

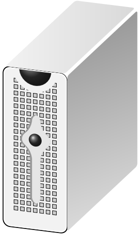
Rev

Rev ... progra... be compiled ... (forward only) program...

*Reversible computing from a programming language perspective*, R. Glück, T. Yokoyama

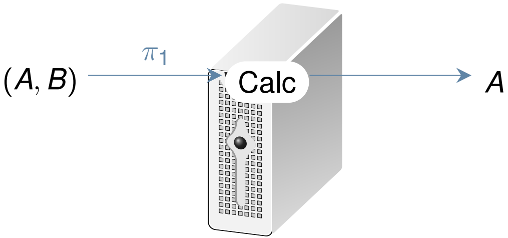
## Question

*Can* any program be reversed?



## Question

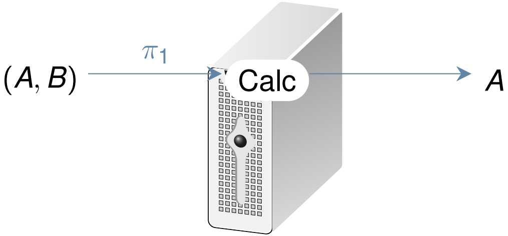
*Can* any program be reversed? **No!**



## Questions

*Can* any program be reversed? **No!**

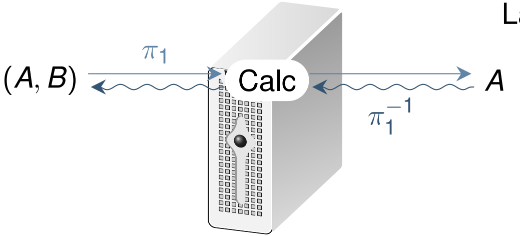
Can any program have an “equivalent” reversible program?



## Questions

*Can* any program be reversed? **No!**

Can any program have an “equivalent” reversible program? **Yes!**

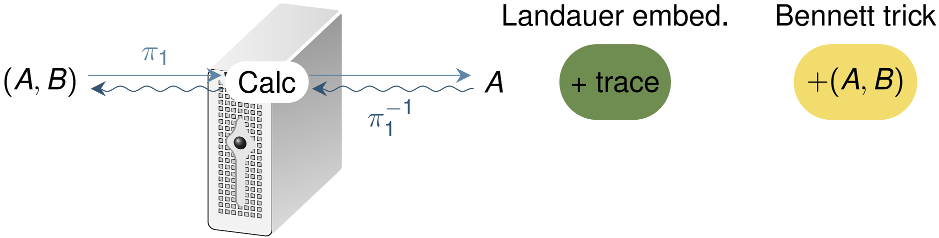



Landauer embed.

+ trace

## Questions

*Can* any program be reversed? **No!**
Can any program have an “equivalent” reversible program? **Yes! (twice!)**



## Questions

*Can* any program be reversed? **No!**

Can any program have an “equivalent” reversible program? **Yes! (twice!)**

*Bennett method* Another way to reversibilize a program $p$ is to preserve its input in the output. The reversibilized version $\overline{p}^{ben}$ then returns the original output $[\![p]\!]_L\ x$ together with the input $x$:

$$[\![\overline{p}^{ben}]\!]_R\ x = ([\![p]\!]_L\ x, x). \qquad (15)$$

Although adding $x$ to the output of $\overline{p}^{ben}$ may seem like a simple change from a functional viewpoint, implementing it in a reversible language is not. We should keep in mind that $\overline{p}^{ben}$ must be implemented in a reversible language $R$, which means it cannot be built from destructive (non-injective) statements like $p$, only from reversible statements. It is not sufficient to add a statement to $\overline{p}^{ben}$ that copies $x$ to the output and otherwise run $p$ with its irreversible statements.

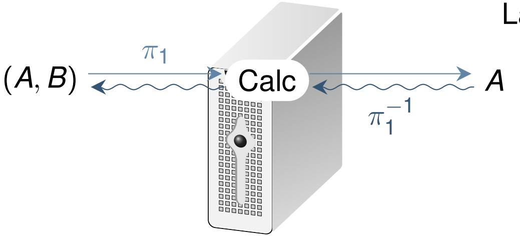
*Reversible computing from a programming language perspective*, R. Glück, T. Yokoyama

## Questions

*Can* any program be reversed? **No!**

Can any program have an “equivalent” reversible program? **Yes! (twice!)**

*Should* any program be reversible?




Landauer embed.

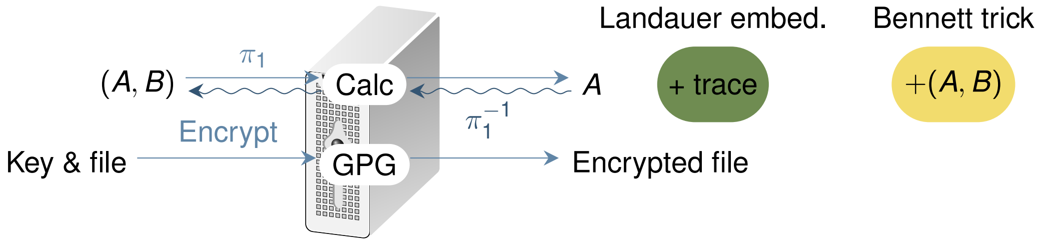
+ trace

Bennett trick

$+(A, B)$

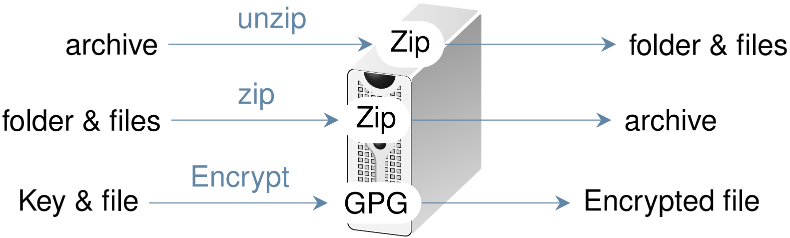
## Questions

*Can* any program be reversed? **No!**
Can any program have an "equivalent" reversible program? **Yes! (twice!)**
*Should* any program be reversible? **No!**



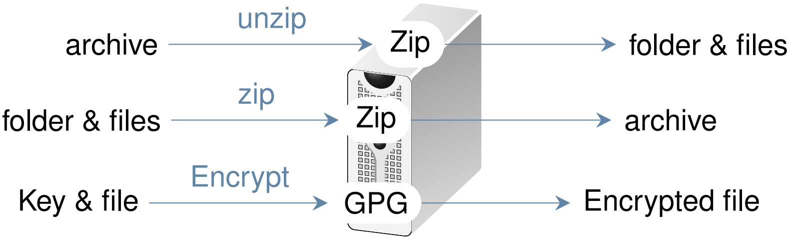
## Question

But . . . why would we want to reverse our computation(s)?



## Question

But . . . why would we want to reverse our computation(s)?




## Mottos

1. Futur(e|istic) hardware will need it.
2. Constraint software = better control = more guarantees.
3. Interesting links to concurrency.

## Computing paradigms connected to reversibility (and their fields)

1. Low-power electronics (Thermodynamic)
2. Quantum computing (Linear algebra)
3. Chemical computing (Theoretical chemistry)

*And remember that we can adopt only a part of a paradigm.*

### Landauer's principle

Logically irreversible transformation dissipates heat.

# Landauer's prin...

Logically irrevers...

Letter | Published: 07 March 2012

## Experimental verification of Landauer's principle linking information and thermodynamics

Antoine Bérut, Artak Arakelyan, Artyom Petrosyan, Sergio Ciliberto, Raoul Dillenschneider & Eric Lutz

*Nature* **483**, 187–189 (2012) | Cite this article



### Abstract

In 1961, Rolf Landauer argued that the erasure of information is a dissipative process[1]. A minimal quantity of heat, proportional to the thermal energy and called the Landauer bound, is necessarily produced when a classical bit of information is deleted. A direct consequence of this logically irreversible transformation is that the entropy of the environment increases by a finite amount. Despite its fundamental importance for information theory and computer science[2,3,4,5], the erasure principle has not been verified experimentally so far, the main obstacle being the difficulty of doing single-particle experiments in the low-dissipation regime. Here we experimentally show the existence of the Landauer bound in a generic model of a one-bit memory. Using a system of a single colloidal particle trapped in a modulated double-well potential, we establish that the mean dissipated heat saturates at the Landauer bound in the limit of long erasure cycles. This result demonstrates the intimate link between information theory and thermodynamics. It further highlights the ultimate physical limit of irreversible computation.

## Landauer’s principle

Logically irreversible transformation dissipates heat.

## Landauer’s principle – extrapolated

Logically *reversible* transformation uses no energy.

### Landauer’s principle

Logically irreversible transformation dissipates heat.

### Landauer’s principle – extrapolated

Logically *reversible* transformation uses no energy.

### Hope

Reversible computers could use less energy.

## Unitary matrices [Wikipedia]

An invertible complex square matrix $U$ is unitary if its matrix inverse $U^{-1}$ equals its conjugate transpose $U^*$, that is, if

$$U^*U = UU^* = \mathrm{id}$$

## Unitary matrices [Wikipedia]

An invertible complex square matrix $U$ is unitary if its matrix inverse $U^{-1}$ equals its conjugate transpose $U^*$, that is, if

$$U^* U = UU^* = \mathrm{id}$$

## Conclusion

Quantum circuits[a] *have* to be reversible.

[a] Without e.g., measurement.

# Reversing Your Computation, but Why?– Quantum computing

Uni

An i … s its

conj

Cor

Qua

a

VOLUME 80, NUMBER 15 PHYSICAL REVIEW LETTERS 13 APRIL 1998

## Experimental Implementation of Fast Quantum Searching

Isaac L. Chuang,[1,*] Neil Gershenfeld,[2] and Mark Kubinec[3]

[1] *IBM Almaden Research Center K10/D1, 650 Harry Road, San Jose, California 95120*

[2] *Physics and Media Group, MIT Media Lab, Cambridge, Massachusetts 02139*

[3] *College of Chemistry, D7 Latimer Hall, University of California, Berkeley, Berkeley, California 94720-1460*

(Received 21 November 1997; revised manuscript received 29 January 1998)

Using nuclear magnetic resonance techniques with a solution of chloroform molecules we implement Grover's search algorithm for a system with four states. By performing a tomographic reconstruction of the density matrix during the computation good agreement is seen between theory and experiment. This provides the first complete experimental demonstration of loading an initial state into a quantum computer, performing a computation requiring fewer steps than on a classical computer, and then reading out the final state. [S0031-9007(98)05850-5]

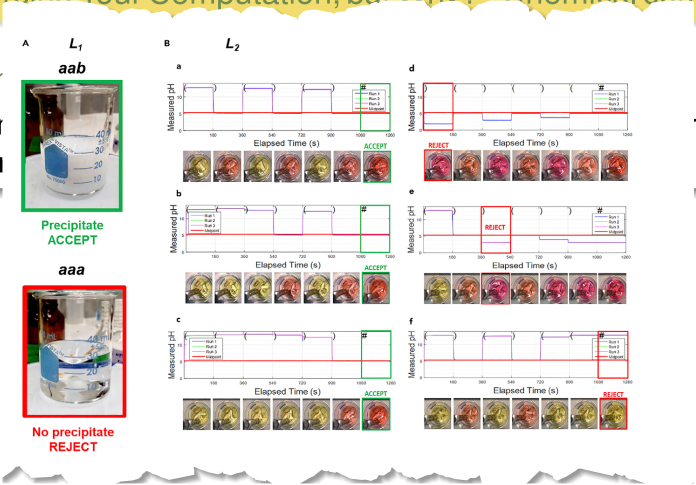
PACS numbers: 89.70.+c, 03.65.–w

### Chemical computer [Wikipidea]

A chemical computer is an unconventional computer where data are represented by varying concentrations of chemicals.

## Chemical cor

A chemical cor … represented by varying concen…




*How Chemistry Computes: Language Recognition by Non-Biochemical Chemical Automata. From Finite Automata to Turing Machines*, M. Dueñas-Díez, J. Pérez-Mercader

## Chemical computer [Wikipidea]

A chemical computer is an unconventional computer where data are represented by varying concentrations of chemicals.

## Reversible reaction [Wikipedia]

$$aA + bB \rightleftharpoons cC + dD$$

### Chemical computer [Wikipidea]

A chemical computer is an unconventional computer where data are represented by varying concentrations of chemicals.

### Reversible reaction [Wikipedia]

$$aA + bB \rightleftharpoons cC + dD$$

### Hope

Reversible chemical computers?

## Software benefits provided by reversibility

1. Development
2. Verification
3. Security

*And remember that we can execute reversible programs on irreversible hardware.*

### Motto

Constraining the programmer can be *a good thing.*

# Reversing Your Computation, but Why?– Software development

Mot

Con

imaginable algorithm as long as it works in polynomial time. Implicit computational complexity theory studies classes of functions (problems, languages) that are defined without imposing explicit resource bounds on machine models, but rather by imposing linguistic constraints on the way algorithms can be formulated. When we explicitly restrict our language for formulating algorithms, that is, our programming language, then we may implicitly restrict the computational resources needed to execute algorithms. If we manage to find a restricted programming language that captures a complexity class, then we will have a so-called implicit characterization. A seminal example is Bellantoni & Cook's [3] characterization of

*Reversible computing and implicit computational complexity*, L. Kristiansen

## Motto

Constraining the programmer can be *a good thing.*

## Reminder

Reversible programs can be compiled into two (forward-only) programs.

Mo

Co

Re

Re

# Source codes in RC2024 paper

The following are complete implementation of the algorithms in the reversible language Janus, appeared in [1].

- LZW: A reversible implementation of LZW running in O(n) time and using O(n) space.
- BWT: A reversible implementation of BWT running in O(n^3) time and using O(n) space.
- source code

[1] Lyngby, T., Nylandsted, R.R., Glück, R., Yokoyama, T. (2024). Towards Clean Reversible Lossless Compression: A Reversible Programming Experiment with Zip. In: Mogensen, T.Æ., Mikulski, Ł. (eds) Reversible Computation. RC 2024. Lecture Notes in Computer Science, vol 14680. Springer, Cham. https://doi.org/10.1007/978-3-031-62076-8_7

## Verify a zip program

1. Write the zip routine.
2. Write the unzip routine.
3. Verify
   1. $\text{zip} \circ \text{unzip} = \mathrm{id}$,
   2. $\text{unzip} \circ \text{zip} = \mathrm{id}$.

## Verify a zip program

1. Write the zip routine.
2. Write the unzip routine.
3. Verify
    1. $\text{zip} \circ \text{unzip} = \mathrm{id}$,
    2. $\text{unzip} \circ \text{zip} = \mathrm{id}$.

## Verify a reversible zip program

1. Write the zip routine.

### Data integrity [Wikipedia]

Ensuring that the data remains the same as when it was originally recorded.

## Data in

Ensurin d.

more efficiently. In this research work, we offer an error detection and correction module using reversible logic which has a reduced power consumption and also boosts efficiency which is integrated onto the AHB-APB interface to detect and rectify faults that may occur in data transmission. The proposed model contains a ECC

*Improving Data Integrity with Reversible Logic-based Error Detection and Correction Module on AHB-APB Bridge*, S. Anant Edidi; R. Marada; T. Ali Khan & Chitra E

### Data integrity [Wikipedia]

Ensuring that the data remains the same as when it was originally recorded.

### Computer forensics

Reversible watermarking is required to produce forensic evidence.

## Data in

Ensurin … d.

## Compu

Reversi

*Review Article*

# Reversible Watermarking Techniques: An Overview and a Classification

**Roberto Caldelli, Francesco Filippini, and Rudy Becarelli**

*MICC, University of Florence, Viale Morgagni 65, 50134 Florence, Italy*

Correspondence should be addressed to Roberto Caldelli, roberto.caldelli@unifi.it

Received 23 December 2009; Accepted 17 May 2010

Academic Editor: Jiwu W. Huang



An overview of reversible watermarking techniques appeared in literature during the last five years

## Concurrency and reversibility

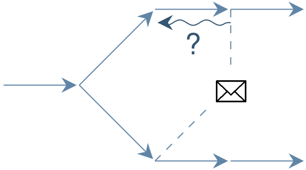
Consider two symmetric labeled transition systems $\xrightarrow{\cdot}$, $\overset{\cdot}{\rightsquigarrow}$ enforcing:

$$R \xrightarrow{a} S \iff S \overset{a}{\rightsquigarrow} R \qquad \text{(Loop Lemma)}$$

1. Defining history
2. Defining independence
3. Defining dependence

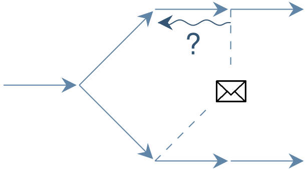
## Problem

How to organize reversible concurrency?



## Problem

How to organize reversible concurrency?




## Solution

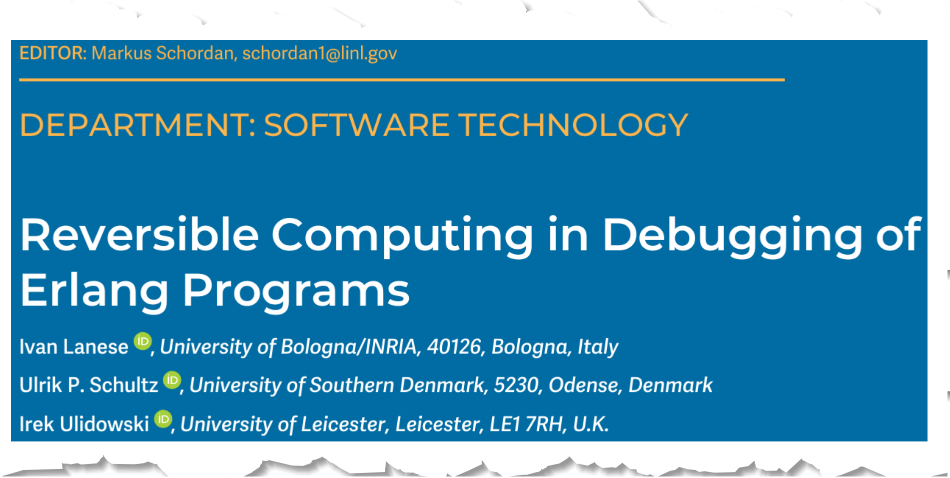
Each thread carries its own (causal) history.

Pro

How

Solu

Eac

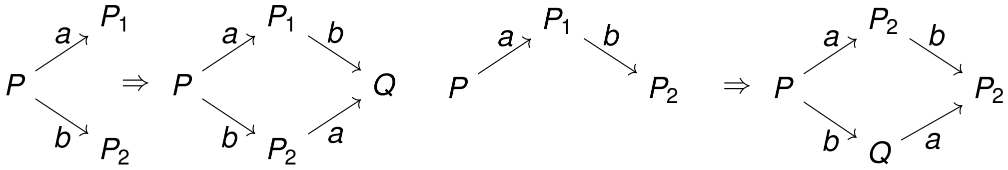

EDITOR: Markus Schordan, schordan1@linl.gov

DEPARTMENT: SOFTWARE TECHNOLOGY

# Reversible Computing in Debugging of Erlang Programs

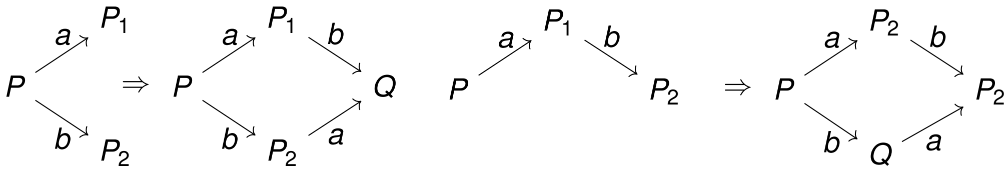
Ivan Lanese, *University of Bologna/INRIA, 40126, Bologna, Italy*
Ulrik P. Schultz, *University of Southern Denmark, 5230, Odense, Denmark*
Irek Ulidowski, *University of Leicester, Leicester, LE1 7RH, U.K.*

## Defining independence (concurrency)

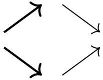


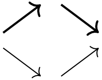

*Drawn as:*

## Defining independence (concurrency)




*Drawn as:*





## Observation (forward-only)

Co-initial and composable definitions *do not have* to be related.

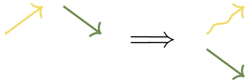
### Observation (reversible)

Co-initial and composable, forward and backward definitions *can easily* be related.

## Observation (reversible)

Co-initial and composable, forward and backward definitions *can easily* be related.

$$R \xrightarrow{a} S \iff S \overset{a}{\rightsquigarrow} R \qquad \text{(Loop Lemma)}$$



(Reversing preserves independence)[1]

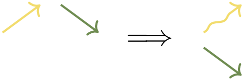
---

1 *An Axiomatic Approach to Reversible Computation*, I. Lanese, I. Phillips, I. Ulidowski

## Observation (reversible)

Co-initial and composable, forward and backward definitions *can easily* be related.

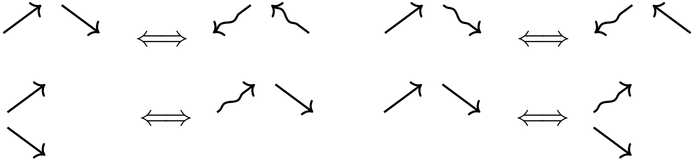
$$R \xrightarrow{a} S \iff S \overset{a}{\rightsquigarrow} R \qquad \text{(Loop Lemma)}$$



(Reversing preserves independence)[1]

[1] *An Axiomatic Approach to Reversible Computation*, I. Lanese, I. Phillips, I. Ulidowski

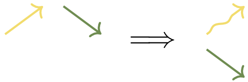
## Inter-defining



## Observation (reversible)

Co-initial and composable, forward and backward definitions *can easily* be related.

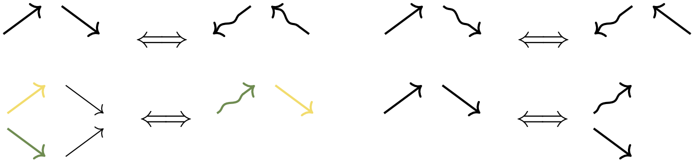
$$R \xrightarrow{a} S \iff S \overset{a}{\rightsquigarrow} R \quad \text{(Loop Lemma)}$$



(Reversing preserves independence)[1]

[1] *An Axiomatic Approach to Reversible Computation*, I. Lanese, I. Phillips, I. Ulidowski

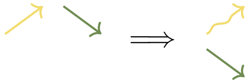
## Inter-defining



## Observation (reversible)

Co-initial and composable, forward and backward definitions *can easily* be related.

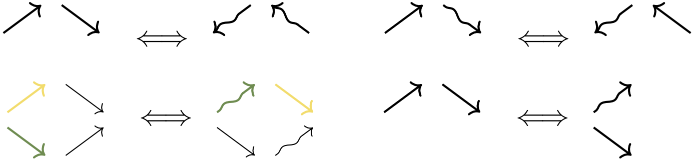
$$R \xrightarrow{a} S \iff S \overset{a}{\rightsquigarrow} R \quad \text{(Loop Lemma)}$$



(Reversing preserves independence)[1]

[1] *An Axiomatic Approach to Reversible Computation*, I. Lanese, I. Phillips, I. Ulidowski

## Inter-defining



## Observation (reversible)

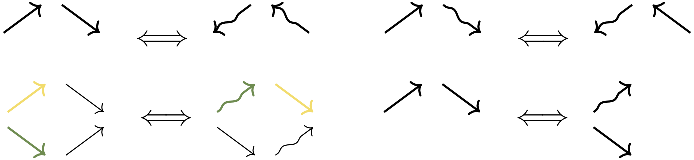
Co-initial and composable, forward and backward definitions *can easily* be related.

$$R \xrightarrow{a} S \iff S \overset{a}{\rightsquigarrow} R \quad \text{(Loop Lemma)}$$

is the *loop lemma*, that states that any transition in a reversible system $t : X \xrightarrow{\theta} Y$ can be reversed[3] as $t^{\bullet} : Y \overset{\theta}{\rightsquigarrow} X$ with $(t^{\bullet})^{\bullet} = t$. From there, a correctness criterion linking $\smile_i$ and $\smile_c$ can easily be formulated:

$$(t_1 \smile_i t_2 \text{ for } t_1 : X \xrightarrow{\theta_1} X_1, t_2 : X \xrightarrow{\theta_2} X_2)$$
$$\iff (t_1^{\bullet} \smile_c t_2 \text{ for } t_1^{\bullet} : X_1 \overset{\theta_1}{\rightsquigarrow} X, t_2 : X \xrightarrow{\theta_2} X_2) \quad \text{(Correctness of Concurrencies)}$$

nter-defin



## Notation

Given two transitions $t$, $u$, we write

- $t \iota u$ if $t$ and $u$ are *independent*,
- $t \times u$ if $t$ and $u$ are *dependent*.

## Notation

Given two transitions $t$, $u$, we write

- $t \iota u$ if $t$ and $u$ are *independent*,
- $t \times u$ if $t$ and $u$ are *dependent*.

## Definition (forward-only)

$t \iota u$ iff neither $t \times u$ nor $t \times u$ hold.

## Notation

Given two transitions $t$, $u$, we write

- $t \iota u$ if $t$ and $u$ are *independent*,
- $t \times u$ if $t$ and $u$ are *dependent*.

## Definition (forward-only)

$t \iota u$ iff neither $t \times u$ nor $t \times u$ hold.

## Theorem (reversible)

$t \iota u$ iff not $t \times u$.

## Notatio

Given tw

— $t \iota$

— $t \times$

## Definiti

$t \iota u$ iff r

## Theore

$t \iota u$ iff r

### Dependence Relation

**Action**

$$\frac{}{\alpha[k] \times \theta}\,\mathrm{A}^1 \qquad \frac{\theta \text{ is not a prefix}}{\theta \times \alpha[k]}\,\mathrm{A}^2$$

**Choice**

$$\frac{\theta \times \theta'}{+_{\mathrm{d}}\theta \times +_{\mathrm{d}}\theta'}\,\mathrm{C}^1 \qquad \frac{}{+_{\mathrm{d}}\theta \times +_{\overline{\mathrm{d}}}\theta'}\,\mathrm{C}^2$$

**Parallel**

$$\frac{\theta \times \theta'}{|_{\mathrm{d}}\theta \times |_{\mathrm{d}}\theta'}\,\mathrm{P}^1 \qquad \frac{\not{k}(\theta) = \not{k}(\theta')}{|_{\mathrm{d}}\theta \times |_{\overline{\mathrm{d}}}\theta'}\,\mathrm{P}^2_k$$

**Synchronisation**

$$\frac{\theta \times \theta_{\mathrm{d}}}{|_{\mathrm{d}}\theta \times \langle |_{\mathrm{L}}\theta_{\mathrm{L}}, |_{\mathrm{R}}\theta_{\mathrm{R}}\rangle}\,\mathrm{S}^1 \qquad \frac{\theta_{\mathrm{d}} \times \theta}{\langle |_{\mathrm{L}}\theta_{\mathrm{L}}, |_{\mathrm{R}}\theta_{\mathrm{R}}\rangle \times |_{\mathrm{d}}\theta}\,\mathrm{S}^2$$

$$\frac{\theta_i \times \theta'_i \qquad \theta_j \curlyvee \theta'_j \qquad i, j \in \{1, 2\}, i \neq j}{\langle |_{\mathrm{L}}\theta_1, |_{\mathrm{R}}\theta_2\rangle \times \langle |_{\mathrm{L}}\theta'_1, |_{\mathrm{R}}\theta'_2\rangle}\,\mathrm{S}^3$$

### Independence Relation

**Action**

*(empty)*

**Choice**

$$\frac{\theta \iota \theta'}{+_{\mathrm{d}}\theta \iota +_{\mathrm{d}}\theta'}\,\mathrm{C}^1$$

**Parallel**

$$\frac{\theta \iota \theta'}{|_{\mathrm{d}}\theta \iota |_{\mathrm{d}}\theta'}\,\mathrm{P}^1 \qquad \frac{\not{k}(\theta) \neq \not{k}(\theta')}{|_{\mathrm{d}}\theta \iota |_{\overline{\mathrm{d}}}\theta'}\,\mathrm{P}^2_k$$

**Synchronisation**

$$\frac{\theta \iota \theta_{\mathrm{d}}}{|_{\mathrm{d}}\theta \iota \langle |_{\mathrm{L}}\theta_{\mathrm{L}}, |_{\mathrm{R}}\theta_{\mathrm{R}}\rangle}\,\mathrm{S}^1 \qquad \frac{\theta_{\mathrm{d}} \iota \theta}{\langle |_{\mathrm{L}}\theta_{\mathrm{L}}, |_{\mathrm{R}}\theta_{\mathrm{R}}\rangle \iota |_{\mathrm{d}}\theta}\,\mathrm{S}^2$$

$$\frac{\theta_1 \iota \theta'_1 \qquad \theta_2 \iota \theta'_2}{\langle |_{\mathrm{L}}\theta_1, |_{\mathrm{R}}\theta_2\rangle \iota \langle |_{\mathrm{L}}\theta'_1, |_{\mathrm{R}}\theta'_2\rangle}\,\mathrm{S}^3$$

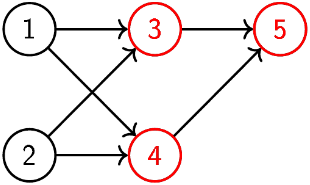
*Work in progress*, C. Aubert, I. Phillips, I. Ulidowski

## Thanks!

Feel free to reach out to

`caubert@augusta.edu`

## Sjouke Mauw




input nodes: $\{1, 2\}$
output nodes: $\{5\}$

- Pebble node 1: $\mathbf{l}(1) = \mathbf{h}(1)$ [erase]
- Pebble node 2: $\mathbf{l}(2) = \mathbf{h}(2)$ [erase]
- Pebble node 3: $\mathbf{l}(3) = \mathbf{h}(3\|\mathbf{l}(1)\|\mathbf{l}(2))$
- Pebble node 4: $\mathbf{l}(4) = \mathbf{h}(4\|\mathbf{l}(1)\|\mathbf{l}(2))$
- Pebble node 5: $\mathbf{l}(5) = \mathbf{h}(5\|\mathbf{l}(3)\|\mathbf{l}(4))$

## Krzysztof Ziemianski & Uli Fahrenberg

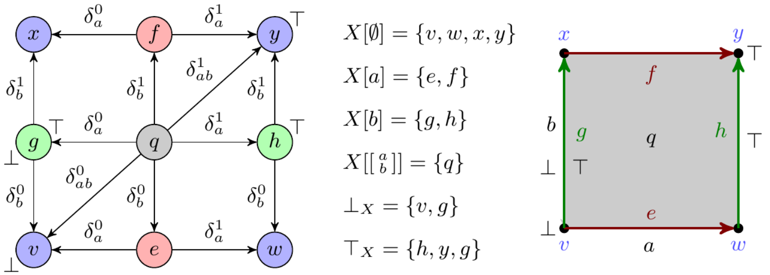


Fig. 5: A two-dimensional HDA $X$ on $\Sigma = \{a, b\}$, see Ex. 7.

terminate some events (*upper* faces) or "unstart" some events (*lower* faces), *i.e.*,

## Ross Horne & Christian Johansen

### 4.3 History-Preserving similarity: preserving causality

Besides observing the duration of events as in ST semantics, History-Preserving semantics observe also the partial order of causal dependencies between events. We define here HP-similarity as a strengthening of our definition of ST-similarity such that we observe not only independence but also dependence,

## Ross Horne & Christian Johansen & Rob van Glabbeek!

### 4.3 History-Preserving similarity: preserving causality

Besides observing the duration of events as in ST semantics, History-Preserving semantics observe also the partial order of causal dependencies between events. We define here HP-similarity as a strengthening of our definition of ST-similarity such that we observe not only independence but also dependence,