Principles of Reversible Computation

Robert Glück
DIKU, University of Copenhagen, Denmark

joint work with
Holger Bock Axelsen
Stéphane Burignat
Alex De Vos
Michael Kirkedal Thomsen
Tetsuo Yokoyama
(Denmark, Belgium, Japan)

Computation is Physical

- Computation is a physical process.
  - paper and pencil
  - electronic device
  - biological system (such as the brain)

- Any computation is subject to physical laws.
  - Physical laws govern all information processing, no matter how it is accomplished.

- Every logical state of a computing device must correspond to a physical state.

Electricity Use by End-Use

At data center level, only ½ of electricity bill goes to computing – other ½ goes to air conditioning...

Thermodynamical Minimum for Irreversible Computation

He then computed the thermodynamical minimum of energy per elementary act of information from the formula $k T \log N$ ergs, where $k$ is Boltzmann's constant (1.38 x 10^{-16} ergs per degree), $T$ is the temperature in absolute units, and $N$ is the number of alternatives. For a binary set $N = 2$, and taking the temperature to be about 300 degrees absolute, he obtained $3 \times 10^{-18}$ ergs for the thermodynamical minimum.

(1 erg = 10^{-7} joule)

Information Loss has a Physical Cost

- Physical states cannot be destroyed, so any bits that are discarded logically turn into entropy.
- This irreversibility directly contributes and therefore power consumption.
- Mismatch

- Our best models of computation are irreversible.
- Carries over to actual implementations in computers.

- Good news: Computation can be organized reversibly.
- Information destruction is not intrinsic to the concept of a computation process.

Reversible Computation

- Adopts a physical principle of reality into a computing model without information loss.

- Consequently, a large part of the power cost associated with destructive computations can theoretically be avoided.

- Sometimes RC a necessity, but also interesting in its own right – regardless of physical motivation.
### Reversible Computing Systems

- Denmark group
- Japan
- Belgium group

### Reversible Logic Gates

**Two essential properties:**
1. The number of input lines is equal to the number of output lines (written $n \times n$).
2. Its Boolean function $B^n \rightarrow B^n$ is bijective.

### Toffoli Gate [Toffoli '80]

**Controlled-Controlled Not gate**

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1</td>
</tr>
<tr>
<td>1 0</td>
<td>1 0</td>
</tr>
<tr>
<td>1 1</td>
<td>1 1</td>
</tr>
</tbody>
</table>

**Universal:** any rev. logic can be built by Toffoli gates

### A Set of Reversible Logic Gates

1. **Not gate**
2. **Feynman gate**
3. **Fredkin gate**
4. **Toffoli gate**

### Reversible Abstract Machine

- **Control logic**
- **Registers**

**Reversible von Neumann architecture**

Shown reversible [AxelsenGlückYokoyama '07]

### Fully Reversible Arithmetic-Logic Unit (ALU) [Thomsen et al. '10]

- Gent's dual-line pass-transistors CMOS technology
- 256 transistors (0.35 µm), 117x77 µm size w/bonding pads
Computability

For all computing devices:

- Conventional irreversible languages: Turing-complete equivalent to Turing-machine. [Turing 1936]

For all reversible computing devices:

- Reversible languages: r-Turing-complete equivalent to reversible Turing-machine. [Bennett 1974]

Injectivization & Reversibilization

We distinguish between two approaches:

1. **Injectivization of non-injective function f**:
   - Necessary for implementing f in any reversible language.
   - NB: This changes type and functionality of the original f.
   - Example: f(x) => (x, f(x))

2. **Reversibilization of irreversible program p**:
   - Transformation of irreversible p into reversible program p'.
   - Obtain reversible algorithm from irreversible one.
   - Examples: compile C program into Janus, turn standard Flowchart into reversible Flowchart, ...

Two Main Steps – Example

1. **Specification**: Injectivization of non-injective function.

   - Fibonacci-Fibonacci-pair classic injective
   - \( f(n) \Rightarrow [f(n), f(n+1)] \)
   - Recall: \( f(1) = 1 \), \( f(2) = 1 \)

2. **Implementation** in a reversible programming language.
   - Reversible algorithm design, coding, optimization, ...
   - Janus Example: Fibonacci-Pairs

Forward & Backward Computation

Example: Program Inversion
Reversible Computing

- **Algorithms design**
  - Examples: FFT, linear transforms, lossless compression

- **Programming languages & tools**
  - Programming language design and implementation
  - Compilation, transformation, optimization
  - Inversion, specialization, composition of programs

- **Abstract machines & computing models**

- **Computability and complexity**

Potential Research Applications

**Technology**
- Low-power CMOS circuits in silicon
- Quantum computing – reversibility is a necessity!
- Optical computing, superconducting circuits

**Software Design & Construction**
- Reliable software construction (reversible languages)
- Automatic generation of software (program inversion)
- Foundations of computation (computability, complexity)

**Computing Applications**
- Simulation of physical systems (law are reversible)
- Lossless encoding (example: zip/unzip)
- Audio/video transformations

References


... and related references therein.