# Reversible Computation

**Classroom Presentation at Appalachian State University**

Clément Aubert

06/02/2017

**Reversible uoıʇʇndɯoƆ**
**Computation ǝʃqısɹǝʌǝɹ**

---


## Abstract

Students know that the flow of a program is a combination of sequential processing, branches, and loops. The introduction of exceptions and their handling, as well as of parallel threads, gives a more fine-grained view on the variations in a program's execution. There is one last variation, of critical impact, that won't be treated in the CS2440 lecture: reversibility. Indeed, recently emerged a completely different way of handling the flow of a program, by allowing the computation to go *back* ***and*** *forth*.

This requires every operation, i.e., statement, in the program to be *invertible*, so that any function, for instance, can seamlessly go from the input to the output, and from the output to the input. Allowing a program to go back and forth offers several advantages:

1. As odd as it may seems, it *saves energy*, due to Landauer's principle that states that "*If no information is erased, computation may in principle be achieved which is thermodynamically reversible, and require no release of heat*", i.e., no consumption of energy.
2. It forces to adopt a programming discipline where no data is ever lost: given an output, one may always 'undo' the computation to read back the input, so that the preservation of information is guaranteed.
3. It allows to re-use code: for instance, the program that computes the $n$th element of the Fibonacci sequence is the same as the one that, given a Fibonacci number, gives you its position!
4. It also open the door to a completely new way of writing and thinking algorithms.
5. Bug tracking becomes easy!

I will provide a quick tour of the motivations and fundamentals of reversible computing and sketch one of my contribution during ~30 minutes, and will happily answer your questions for the rest of the lecture. Some material will be posted at https://lacl.fr/~caubert/ASU/cp.html.

## When and Where?

### When

Monday, February 6, 2017, 10:00 PM – 10:50 PM

### Where

Appalachian State University, Anne Belk Hall, Room 325

---

## Code Shown During the Lecture

The Fibonacci Pairs code, written in `Janus`, is a canonical example:

```
procedure fib(int x1, int x2, int n)
  if n=0 then x1 += 1
              x2 += 1
         else n -= 1
              call fib(x1, x2, n)
              x1 += x2
              x1 <=> x2
  fi x1=x2
```

| Register | x1 | x2 | n | (Comment) |
|---|---|---|---|---|
| Step 1 | 0 | 0 | 4 | (call fib(0, 0 4)) |
| Step 2 | 0 | 0 | 3 | (call fib(0, 0, 3)) |
| Step 3 | 0 | 0 | 2 | (call fib(0, 0, 2)) |
| Step 4 | 0 | 0 | 1 | (call fib(0, 0, 1)) |
| Step 5 | 0 | 0 | 0 | (call fib(0, 0, 0)) |
| Step 6 | 1 | 1 | 0 | (terminate call fib(0, 0, 0)) |
| Step 7 | 1 | 2 | 0 | (terminate call fib(0, 0, 1)) |

| Register | x1 | x2 | n | (Comment) |
|---|---|---|---|---|
| Step 8 | 2 | 3 | 0 | (terminate call fib(0, 0, 2)) |
| Step 9 | 3 | 5 | 0 | (terminate call fib(0, 0, 3)) |
| Step 10 | 5 | 8 | 0 | (terminate call fib(0, 0, 4)) |

## To Go Further

### Reversible Programming Languages

- `Janus` is probably the oldest and most robust reversible programming language. Its playground is unfortunately broken, but should be fixed soon.
- `Joule` is an object-oriented variation on `Janus`.
- `rfun` is an experimental, functional and reversible programming language, with an interpreter for `Haskell`.
- `Boomerang` is a 'bidirectional programming language for ad-hoc, textual data'.
- `JsonGrammar` is a bidirectional '`Haskell` library for converting between `Haskell` datatypes and `JSON ASTs`'.

### Libraries

Code for reversible programming languages is hard to find, with one notable exception: Sarah Vang Nøhr, published the `Janus` code that resulted from her Master's thesis (*Reversible Graph Algorithms*, January 2015). Her pioneer work in the adaptation of graph algorithms for reversible computation is well-documented, solid, and enlightening.

### Readings and Viewings

**Video** Holger Bock Axelsen, from the University of Copenhagen, gave an excellent 10-minutes introduction to Reversible Computing.

### Textbook

- *Introduction to Reversible Computing*, by Kalyan S. Perumalla, is 'envisioned to be suitable at the senior undergraduate and graduate levels.'

The same author gave a tutorial in 2014, that gives a rough idea of the extend of the topic, along with some useful references.

- An excellent introduction and panorama of the field is covered by Michael Kirkedal Carøe's Ph.D. thesis *Design of Reversible Computing Systems*.

### Research Papers

- *Reversible Computation and Reversible Programming Languages* is a clear and accessible tutorial to reversible programming, presenting and using `Janus` through simple examples (but you should probably skip Section 2.3, which is a bit difficult).
- *Elements of a Reversible Object-Oriented Language* gives elements to extend `Janus` with object-oriented features.
- *Time, space, and energy in reversible computing* covers the broad and complex topic of how to measure complexity on reversible machines, along with an excellent introduction that surveys results of a quarter century of work on reversible computation.
- *Interpretation and Programming of the Reversible Functional Language RFUN* lays the foundation of the first purely reversible and functional programming language.

### Traveling

Make sure to submit your work to the international conference Reversible Computation, whose next edition will take place in India!

---

## Misc.

- Download a `pdf` version or the `md` source of this page
- HTML5 and CSS3 valid
- 
- Contact: aubertc@appstate.edu
- Created with debian, pandoc, latex and emacs.