Reversible Models for Programming Molecular Computers

William Earley · w.earley@damtp.cam.ac.uk · Department of Applied Mathematics and Theoretical Physics · University of Cambridge

`alethe` & The ℵ Calculus

As well as promising lower energy costs, reversible computing is of particular interest to molecular computing because it better exploits the physics of this domain, namely microscopic reversibility. Hence, we propose a model of reversible computation, the ℵ (*aleph*) calculus, with features desirable in a reversible molecular computing context. We also introduce the associated programming language, `alethe`, meaning *not forgotten*.

Key insights

As far as we are aware, all existing reversible programming languages keep program and data separate. Implementations of such models can be very complicated, as the computers must maintain an often complicated representation of where in the program it currently is. This makes direct molecular implementation of such models especially challenging.

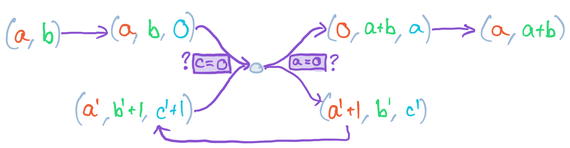
Our model circumvents this by being a term-rewriting system (TRS). In a TRS, there is no distinction between program and data. Instead, they are combined into a 'term', which gives a complete[1] representation of the state of computation at any point along its worldline. As the below listing shows, this representation is concise, and our reversible TRS makes it particularly easy to step forward and backward through this computational worldline.

```
-- Peano definition of natural numbers
data Z;
data S n;

-- reversible Peano addition
      + a b _ = + a b Z +;
+ (S a) b c + = + a (S b) (S c) +;
  + Z a+b a + = _ a a+b +;

-- termination conditions
! + a b   _;
! _ a a+b +;
```

```
+ 3 4 _
    ↕
+ 3 0 4 +
    ↕
+ 4 1 3 +
    ↕
+ 5 2 2 +
    ↕
+ 6 3 1 +
    ↕
+ 7 4 0 +
    ↕
_ 7 4 +
```



top-left: An example implementation of Peano addition written in `alethe`.

top-right: Each term encountered during the reversible addition of 4 to 3. Note that the result necessarily includes one of the addends.

bottom: A schematic of the control flow of the addition algorithm.

The choice of a TRS makes particularly good sense from a molecular implementation point of view. Terms keep the relevant bits of program next to the relevant bits of data, allowing computation to be effected by local operations and manipulation. In contrast, other models often need to synchronise activity over large domains. Their structure also is strongly suggestive of a molecular representation.

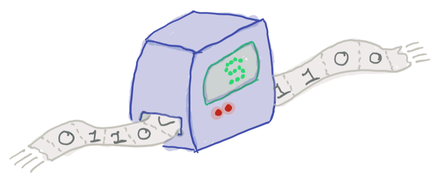
-- Some simple examples:

```
-- a simpler, recursive infix implementation of Peano addition
a `+ Z` a;
a `+ (S b)` (S c):
    a `+ b` c.

-- the Cantor pairing function π : ℕ² ↔ ℕ
-- this definition makes use of the comparison operator >
a b `Pair` n:
    ! ~Go n Z True = ~Go b (S a+b) False.
    ~Go n a+b True = ~Go n' (S a+b) p:
        n' `+ a+b` n.
        `> n' a+b` p.
    a `+ b` a+b.

-- the Factorial function, with no garbage
-- this is only a partial bijection, and will not compute 0!
(S n) `Fac` m:
    ! ~Go (S n)     1 = ~Go 1     m.
      ~Go (S (S n)) a = ~Go (S n) (S (S a')):
          a `* (S (S n))` (S (S a')).
```

-- A proof of Turing completeness:




```
-- a one hole context representation of a bi-infinite tape
data Tape l x r;
data Symb x;
data Blank;

[Blank    x . xs] `Pop` Blank    [x . xs];
[(Symb x)   . xs] `Pop` (Symb x) xs;
[]                `Pop` Blank    [];

--- tape movement
x `Id` x;

(Tape l x r) `Left` (Tape l' x' r'):
  l  `Pop` x' l'.
  r' `Pop` x  r.

t `Right` t':
  t' `Left` t.

--- tape initialisation
xs `MkTape` (Tape [] Blank xs'):
  x ~ (Symb x);
  xs `Map ~` xs'.


-- a four-tape reversible Turing Machine (RTM), per Bennett [1]

--- halting states
! Start t1 t2 t3 t4;
! Stop  t1 t2 t3 t4;

--- the rule S1 [C / D /] ⟷ [B - F +] S2
  S1 (Tape l1 C r1) t2  (Tape l3 D r3) t4
= S2 (Tape l1 B r1) t2' (Tape l3 F r3) t4':

    t2 `Left`  t2'.
    t4 `Right` t4'.
```

Definition

The ℵ calculus has a very simple and concise definition. In BNF notation, this is:

$$\begin{aligned} \text{(pattern term)} \quad & \tau ::= \text{ATOM} \mid \text{VAR} \mid (\tau^*) \\ \text{(rule)} \quad & \rho ::= \tau^* \leftrightarrow \tau^* \\ \text{(definition)} \quad & \delta ::= \rho : \rho^* \mid \vdash \tau^* \end{aligned}$$

`alethe` is essentially the same except for some syntactic sugar for common programming motifs. The model is perhaps best understood by example (see the central listing).

Essentially, we define rules that map terms to other terms. Each rule has two patterns, and can thus convert a term between these two forms. Patterns need not match the term as a whole – they can match anywhere in a term. We can express more complex maps via subrules, which are the primary means of composition.

Additional constraints on this definition, as well as semantics, will be presented in the full forthcoming paper.

Select Features

- The ℵ calculus is microscopically reversible by design
  - This makes it directly compatible with the laws of physics without the need for an external source of free energy.
  - It may even make it easier to implement molecularly.
  - It encourages an economical programming style. Often one finds that garbage data can be recycled somehow or avoided altogether.
- Automatic parallelisation
  - When possible, subrules or subterms can be automatically evaluated in parallel.
  - Subrules can even be evaluated in both directions when needed.
    - This includes Bennett's algorithm [1] as a special case.
- Effects and interactions are easily accommodated, e.g.
  - Walking along a lattice
  - Actuating molecular machinery

Extensions

- If run in a thermally coupled environment, the ℵ calculus can be extended to handle non-determinism and irreversibility by allowing for ambiguous patterns. This can be done whilst preserving microscopic reversibility.
- The ℵ calculus can also be easily extended to support first class concurrency, in which terms can freely fission and fuse. The semantics and consequences of this will be discussed in the full paper.

Both of these extensions permit entropy-generating behaviour and so should be used with caution.

Future work

Though the ℵ calculus is motivated by reversible macromolecular computing systems, it is perhaps too high level for a direct implementation. We plan next on exploring simpler models more amenable to molecular implementation.

[1] Rewriting rules are defined externally, but require no extra hidden state and are intended to be shared between programs. In effect, they define a DSL (domain-specific language).

UNIVERSITY OF CAMBRIDGE

Department of Applied Mathematics and Theoretical Physics (DAMTP)

MICKLEM LAB

EPSRC The author would like to acknowledge the support of the Engineering and Physical Sciences Research Council (EPSRC), project reference 1781682 for "*Modelling approaches to molecular computation*".

References

[1] Charles H Bennett. 'Logical Reversibility of Computation'. In: *IBM J. Res. Dev.* 17.6 (Nov. 1973), pp. 525–532.