Random Thoughts on Abstract Machines

Peter Hines

University of York (2006)

http://www-users.cs.york.ac.uk/∼phines/AbstractMachine.ps

Motivation ?

We want a 'model of computation' that is :

- physically motivated – the important point!
- as abstract as possible,
- but concrete enough to calculate with.
- *useful* — but we can't have everything ...

We do not care about :

- computing as *calculating a result*, rather than as a *physical process*.
- the program / data distinction.

Ideally :

- we recover *familiar concepts* or *useful tools*.
- we can *generalise* or *restrict* to other settings
  - reversible, asynchronous, quantum, &c.

Physically, what is a computer ?

i.e. how abstract can we go ?

An **Abstract Computing Machine** is:

1. A set X of configurations.
2. An evolution rule $\mathcal{R}$ that takes

   configurations to *next configurations.*

and that's all ...

(For the moment) we also require

The 'evolution rule' $\mathcal{R}$ is :

(i) *Deterministic*

(ii) *Possibly partial*

(iii) *Fixed — it does not change over time!*

If we want time-dependence, we need to build in a *clock*, by modifying both the configuration set X and the evolution rule $\mathcal{R}$.

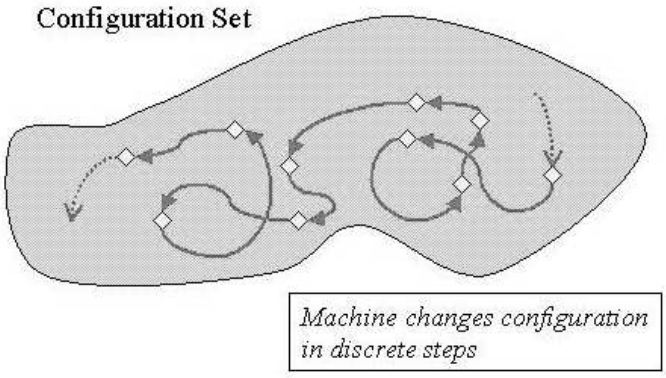
Notation

For an Abstract Computing Machine $\mathcal{M} = (X, \mathcal{R})$, write *"The next configuration of x is y"* as either

- $\mathcal{R}(x) = y$ *(functional notation)*
- $x \rightarrowtail y$ *(relational notation)*

Mathematically : an ACM is trivial

— is is just a **partial function** acting on a **set**.



- Also known as a **deterministic, unlabelled, state transition system**
- However, we study it from a different perspective :
- . . . as first a *physical* and then a *computing* system.

As a Physical System :

Possible problems

1. Partiality
   - what does it mean for a configuration to have no next configuration, under a physical evolution rule ?
2. Irreversibility
   - How can physical time-evolution be irreversible ?

Possible solutions

1. **Partiality :**
   - **Partial information** : The configuration set X is part of a larger set Y

     ... but we can only observe the configurations in X.
   - **A Halting convention** : For example, we only consider evolutions that *change configuration* — we rule out

     $$x \rightarrowtail x \rightarrowtail x \rightarrowtail x \rightarrowtail x \rightarrowtail \ldots$$

     Alternatively, some subset $H \subseteq X$ is chosen as the *halting subset*, and we terminate the experiment when this subset is entered.
2. **Irreversibility**
   - **Classically**, this is fine — we can 'dump information to the environment' with no side-effects.
   - **Quantum-mechanically**, this is *not* the case !

As a Computational System

Motivating examples Turing Machines, Cellular automata, the von Neumann architecture, λ-terms (with fixed reduction strategies), finite state automata (with specified input string), Procedural subroutines, &c.

A natural operation

Given the **Next** relation $x \rightarrowtail y$, consider its *transitive closure*.

- $x \rightarrowtail y$ implies $x \rightsquigarrow y$
- $a \rightsquigarrow b$ and $b \rightsquigarrow c$ implies $a \rightsquigarrow c$

Call this the **Leads To** relation $x \rightsquigarrow z$

Important : unlike transition systems, we do not look at the *reflexive* transitive closure.

Interpretation

Given a machine $\mathcal{M}$ in configuration x, with $x \rightsquigarrow z$, then the machine $\mathcal{M}$ will at some later point be in configuration z.

Unfortunately the 'leads to' relation contains strictly less

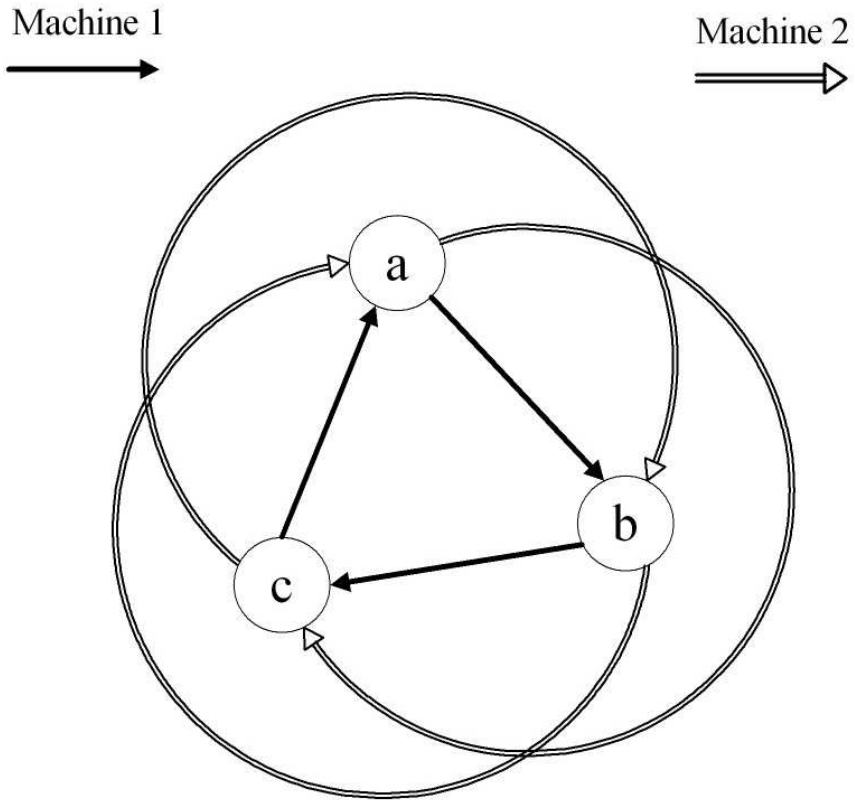
information about a computation than the 'next' relation.

A trivial example

Wherever cycles may occur, the 'leads to' relation loses information about causal ordering (i.e. "p then q then r ...")

Consider two distinct machines $\mathcal{M}_1$ and $\mathcal{M}_2$ with configuration set $X = \{a\ ,\ b\ ,\ c\}$

$$\left.\begin{array}{l} a \rightarrowtail_1 b \rightarrowtail_1 c \rightarrowtail_1 a \\ a \rightarrowtail_2 c \rightarrowtail_2 b \rightarrowtail_2 a \end{array}\right\} \text{'Next' relations for } \mathcal{M}_1\ ,\ \mathcal{M}_2$$



The 'leads to' relation is the same, universal, relation for $\mathcal{M}_1$ and $\mathcal{M}_2$. We can only recover information about ordering when there are no cycles!

Cycle-free machines

An Abstract Computing Machine $\mathcal{M} = (X, \rightarrowtail)$ is **cycle-free** when :

"There does not exist a configuration x such that $x \rightsquigarrow x$"

For finite configuration sets, this is equivalent to **nilpotency** :

$$\mathcal{R}^N = 0 \quad \text{for some} \quad N \in \mathbb{N}$$

For infinite configuration sets, this is **undecidable**.

Advantages of cycle-free machines

- Termination (in the finite case) is guaranteed — the computer never gets stuck in an infinite loop.
- We can recover causal ordering and the $Next$ relation from the 'Leads to' relation.
- the $\rightsquigarrow$ relation is a **strict partial order**
  - Irreflexive : $x \not\rightsquigarrow x$ for any $x \in X$.
  - Transitive : $x \rightsquigarrow y$ and $y \rightsquigarrow z$ implies $x \rightsquigarrow z$.
  - Antisymmetric : $x \rightsquigarrow y$ implies $x \neq y$ and $y \not\rightsquigarrow x$.
- Hence, configurations form a **Directed Acyclic Graph**.

  We can induce a partial ordering by

  $$x \leq y \quad \Leftrightarrow \quad x \rightsquigarrow y \quad \text{or} \quad x = y$$

  - and hope the *order-preserving* or *monotonic* functions may give an interesting theory!

Why not restrict ourselves ?

Should we take this approach ?

Disadvantages of cycle-free machines

- Cycle-freeness is **undecidable** in general.
- We rule out most interesting examples (Turing machines, von Neumann architecture, &c.)
- We lose any hope of compositionality
  - Any *reasonable notion* of 'plugging one machine into another' will allow for the creation of cycles by composing two cycle-free machines.
- The quantum case (pure states & unitary evolution) is long gone!

... in any case, cycle-freeness is less about machine evolution, and more about choice of halting scheme.

An alternative approach

We will work with, and order :

- *partial functions* on the configuration set,
- rather than *configurations themselves.*

Q : Why *partial functions* on configurations ?
A : Simply because the evolution rule $\mathcal{R}$ is a partial function.

More definitions !

Given $\mathcal{M} = (X, \rightarrowtail)$, a **machine evolution** is a partial function $\eta : X \to X$ where

$$\eta(x) = y \quad \Rightarrow \quad x \rightsquigarrow y$$

Interpretation :
When $\mathcal{M}$ is in configuration x then *at some later point*, $\mathcal{M}$ will be in configuration y.

We study the set of *all* machine evolutions $[\mathcal{M}]$, and call this the **machine semantics** of $\mathcal{M}$.

What do we know about $[\mathcal{M}]$?

At least some properties are immediate :

- $[\mathcal{M}]$ is a **semigroup** *(by transitivity of $\rightsquigarrow$)*
- The semigroup $[\mathcal{M}]$ contains a **zero element**
  - *(the nowhere-defined function $0_X : X \to X$)*
- The semigroup $[\mathcal{M}]$ has a **partial summation :**
  - *given η, μ with disjoint domains, then*

$$(\eta + \mu)(x) \begin{cases} \eta(x) & x \in dom(\eta) \\ \mu(x) & x \in dom(\mu) \\ \textit{undefined} & \textit{otherwise} \end{cases}$$

  - *For category theorists, $[\mathcal{M}]$ is enriched over partially additive monoids.*
  - *In this setting, addition is just 'putting together bits of a jigsaw'. In other settings, we must be careful about interpretations!!*

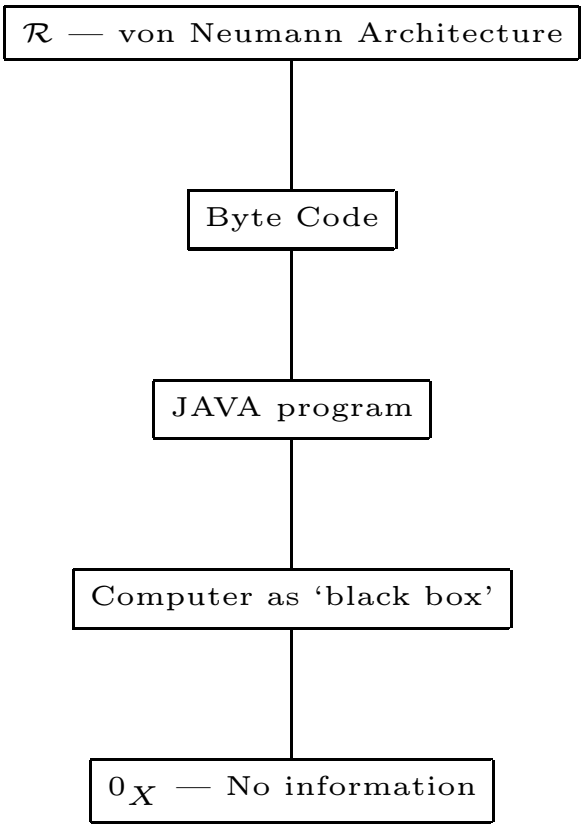
Comparing evolutions

Motivation : "Evolutions $\eta, \mu \in [\mathcal{M}]$ are simply ways of looking at the same machine, at different levels of abstractness."

An example : Recall motivation from von Neumann computers :

> Let $\mathcal{M}$ be a **desktop P.C.**, running a **JAVA program**,
> that is compiled into **Interpreted Byte Code**,
> and executed as **Intel 68000 machine code**

we can look at this in a number of different ways!



Comparing evolutions (cont.)

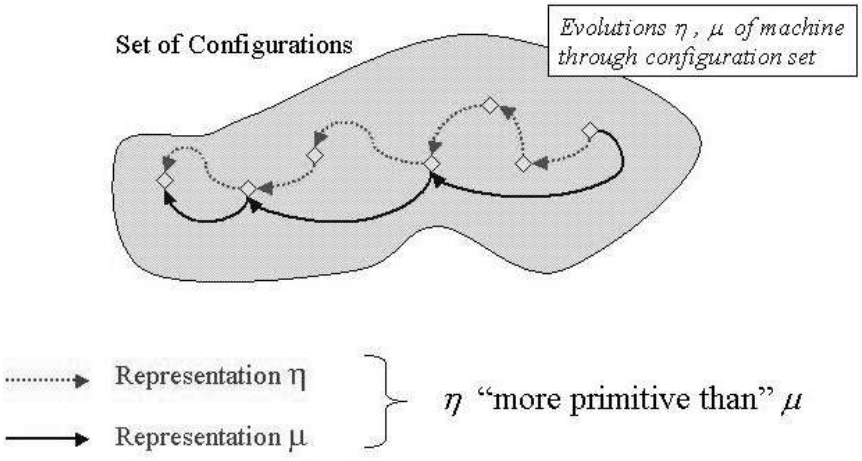
For $\eta, \mu \in [\mathcal{M}]$, say η is **more primitive than** μ when :

- When $\mu(c) = d$, there exists $K > 0$, with $\eta^K(c) = d$.

Write this as $\mu \prec \eta$

Interpretation

- μ is a (partial) description of the behaviour of $\mathcal{M}$.
- η is a more complete description of $\mathcal{M}$
  - because we can recover μ by *restricting* or *iterating* η."



Important ! K is not a *fixed* integer:

— Each Virtual Machine instruction does not take the same number of clock cycles.

— Each Java language command does not take the same number of Virtual Machine instructions.

— The time a computer takes to terminate depends on the input.

What do we know about this relation ?

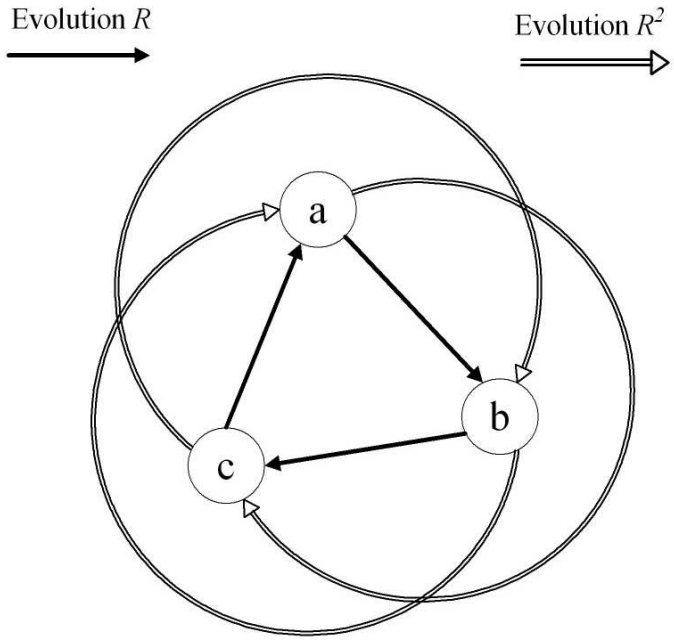
In any Abstract Machine $\mathcal{M} = (X, \mathcal{R})$, we know that

- $\eta \prec \eta$, for all $\eta \in [\mathcal{M}]$
- $0_X \prec \eta \prec \mathcal{R}$.
- When $\eta + \mu$ exists, then $\eta \prec \eta + \mu$.

Q: Is 'primitiveness' a partial order ?

A: No ! (or, *not yet* !)

Counterexample :



In this case,

$$R \prec R^2 \quad , \quad R^2 \prec R$$

Taking the *induced quotient* does not help — we will identify everything!

What saves us ?

Fact : Even when a machine $\mathcal{M} = (X, \rightarrowtail)$ contains cycles , it has cycle-free evolutions.

Cycle-free evolutions are those where :

$$\eta^K(x) \neq x \quad \forall x \in X, \quad K \in \mathbb{N}$$

We define $[\![\mathcal{M}]\!] \subseteq [\mathcal{M}]$, the **cycle-free semantics** for $\mathcal{M}$, to be the set of *all cycle-free evolutions* :

$$[\![\mathcal{M}]\!] = \{\eta \in [\mathcal{M}] \ : \ \eta^K(x) \neq x \quad \forall x \in X, \quad K \in \mathbb{N}\}$$

Motivation : We think of these as restrictions of the machine evolution that actually *compute something* ... possibly by setting Starting & Halting criteria

— alternatively, these are the machine evolutions that give a well-behaved mathematical theory!

What do we know about cycle-free semantics ?

Let $\mathcal{M} = (X, \mathcal{R} : X \to X)$ be an abstract machine.

When we consider the cycle-free semantics $[\![\mathcal{M}]\!]$,

We gain :

- **(Reflexivity) :** $\eta \prec \eta$ for all $\eta \in [\![\mathcal{M}]\!]$.
  - **Proof :** by definition of $\prec$.
- **(Anti-symmetry) :** $\eta \prec \mu$ and $\mu \prec \eta$ implies $\eta = \mu$.
  - **Proof :** a consequence of cycle-freeness.
- **(Transitivity) :** $\eta \prec \mu$ and $\mu \prec \zeta$ implies $\eta \prec \zeta$.
  - **Proof :** by definition of $\prec$.
- **(Bottom element) :** The nowhere-defined arrow 0_X is a bottom element, so $0_X \prec \eta$.
  - **Proof :** Trivial, by definition of 0_X.
- **(Additivity) :** When $\eta + \mu$ is defined and is cycle-free, then $\eta \prec \eta + \mu$.
  - **Proof :** Trivial, by definition of summation of partial functions.
  - **Warning :** Even when defined, the sum of two cycle-free elements might not be cycle-free.

We lose :

- **Semigroup structure** $[\![\mathcal{M}]\!]$ need not be closed under composition.
  - η and μ cycle-free does not imply that $\eta\mu$ is cycle-free.
  - although $\eta \in [\![\mathcal{M}]\!]$ implies $\eta^2, \eta^3, \eta^4, \ldots \in [\![\mathcal{M}]\!]$.
- **'Top' element** $[\![\mathcal{M}]\!]$ might not contain $\mathcal{R}$
  - The 'primitive evolution' need not be cycle-free.
  - As a corollary, the partial order $\prec$ need not have a top element.

What these posets look like – the simple case.

The special case where $\mathcal{M} = (X, \mathcal{R} : X \to X)$ is cycle-free.

$$\mathcal{R}^N(x) \neq x \quad \forall\, x \in X \quad N \in \mathbb{N}$$

Trivially :

- Every evolution is cycle-free, so $[\![\mathcal{M}]\!] = [\mathcal{M}]$.
- $[\mathcal{M}], \prec$ is a partially ordered semigroup, enriched over a partial addition, with top and bottom elements, $\mathcal{R}$ and 0_X, and compatibility between the summation and the ordering.

How about Meets and Joins ?

$[\mathcal{M}], \prec$ is closed under **finite meets**, and **arbitrary joins**.

Meets : For evolutions $\eta, \mu \in [\mathcal{M}]$

- $(\eta \wedge \mu)(x)$ is *undefined* when either $\eta(x)$ or $\mu(x)$ is undefined.
- Otherwise, there exists unique $p, q > 0$ with

$$\eta(x) = \mathcal{R}^p(x) \quad \text{and} \quad \mu(x) = \mathcal{R}^q(x)$$

  - In this case, define

$$(\eta \wedge \mu)(x) = \mathcal{R}^{lcm(p,q)}(x)$$

Joins : For evolutions, $\{\eta_i\}_{i\in I} \subseteq [\mathcal{M}]$

- $\left(\bigvee_{i\in I} \eta_i\right)(x)$ is *undefined* when $\eta_i(x)$ is undefined for <u>all</u> $i \in I$.
- Otherwise, consider the subset $\{\eta_j\}_{j\in I\subseteq I}$ of evolutions where $\eta_j(x)$ is defined.
  -
  - There exists <u>unique</u> $\{p_j > 0\}$ with

$$\eta_j(x) = \mathcal{R}^{p_j}(x)$$

  - We then define $\left(\bigvee_{i\in I} \eta_i\right)(x) = \mathcal{R}^{gcd(\{p_j\})}(x)$

Anything else ?

Distributivity : <u>Finite</u> meets distribute over <u>arbitrary</u> joins.

$$\mu \wedge \left(\bigvee_{i\in I} \eta_i\right) = \bigvee_{i\in I} (\mu \wedge \eta_i)$$

— this follows from the definition of $\wedge, \vee$ and simple properties of $gcd(\ )$ and $lcm(\ )$

Relative Pseudocomplements :

- For any element η, μ, the set $\{\zeta : \eta \wedge \mu \leq \zeta\}$ has a *least upper bound*, $\eta \Rightarrow \mu$.

— this follows from *distributivity & cycle-freeness*.

We can identify $[\mathcal{M}], \prec$ as a **Heyting Algebra**

— these play the same rôle for **intuitionistic logic** that *Boolean algebras* play for *classical logic*.

What these posets look like – the complicated case!

The general case : $\mathcal{M}$ is not cycle-free.

In $[\![\mathcal{M}]\!]$, we no longer have :

- A semigroup structure — athough integer powers of evolutions are well-defined.
- Meets and Joins : $\eta \wedge \mu$ and $\eta \vee \mu$ need not exist.
- Arbitrary suprema, relative pseudocomplements, &c.

We do still have:

- A partial ordering.
- A bottom element, $0_X : X \to X$.
- Suprema of some sets.

Given a **chain** of evolutions

$$\mathcal{C} = \{ \ldots \prec \eta_i \prec \eta_{i+1} \prec \eta_{i+2} \prec \ldots \} \subseteq [\![\mathcal{M}]\!]$$

We claim this has a supremum, $sup(C) \in [\![\mathcal{M}]\!]$

Consider η_i defined at a configuration $x \in X$.

By definition, $\eta_{i+1}, \eta_{i+2}, \ldots$ are all defined at x.

— **Assume** that $\eta_j(x) \neq \eta_{j+1}(x)$:

This interprets as

$$\begin{array}{ccccccc} \eta_i(x) & = & \eta_{i+1}^{k_1}(x) & & & & \\ & & \eta_{i+1}(x) & = & \eta_{i+2}^{k_2}(x) & & \\ & & & & \eta_{i+2}(x) & = & \eta_{i+3}^{k_3}(x) \\ & & & & & & \vdots \end{array}$$

Where $\{k_1, k_2, k_3 \ldots\}$ are all greater than 1.

This gives

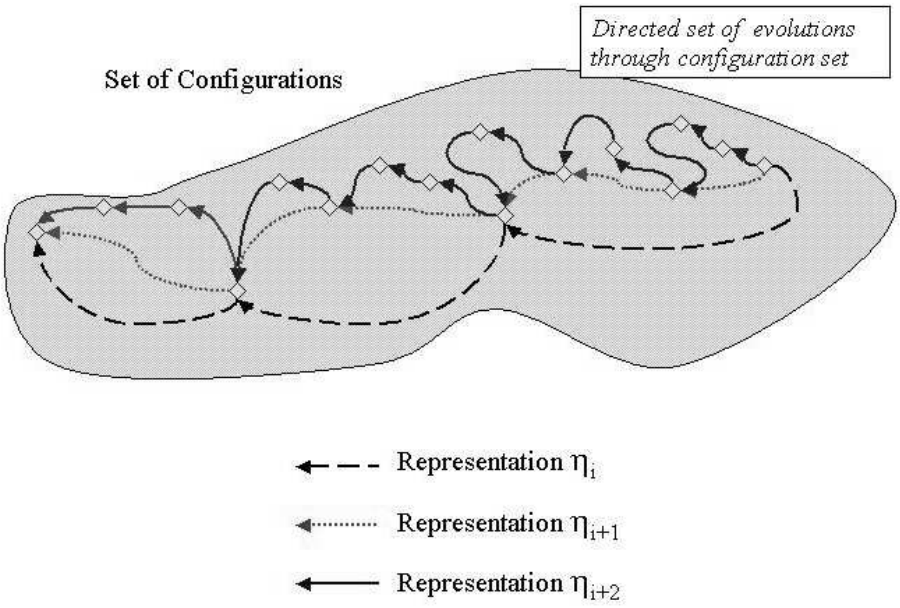
$$\eta_i(x) = \eta_{i+1}^{N_1}(x) = \eta_{i+2}^{N_2}(x) = \eta_{i+3}^{N_3}(x) = \ldots$$

with

$$N_1 < N_2 < N_3 < \ldots$$

Can prove formally this contradicts cycle-freeness & the definition of an evolution !

— a graphical demonstration is more interesting :



$$\eta_i \prec \eta_{i+1} \prec \eta_{i+2} \prec \ldots$$

Given **discrete evolution** and **no repeated configurations** we cannot **infinitely subdivide** a computational path

— Zeno's paradox does not apply to discrete computation!

(despite various proposed schemes for hypercomputation ...)

Completeness of cycle-free semantics

We have seen, for any chain

$$\mathcal{C} = \{ \ldots \prec \eta_i \prec \eta_{i+1} \prec \eta_{i+2} \prec \ldots \} \subseteq [\![\mathcal{M}]\!]$$

and any configuration $x \in X$, there exists some η_M, with

$$\forall\ \eta_j \quad , \quad \exists N > 0, \quad : \quad \eta_j(x) = \eta_M^N(x)$$

Make the obvious definition :

$$Sup(\mathcal{C})(x) = \eta_M(x)$$

(not forgetting M is a function of x).

Small amount of extra work (to prove *cycle-freeness* , &c.), shows that this is the least upper bound of this chain.

Finally :

"A partially ordered set P is a DCPO if and only if each chain in P has a supremum"
— *T. Iwamura (1944) , using* Axiom of Choice

We can identify $[\![\mathcal{M}]\!], \prec$ as a **DCPO** with a bottom element

— can do various things like 'finding least fixed-points of monotone functions', &c.

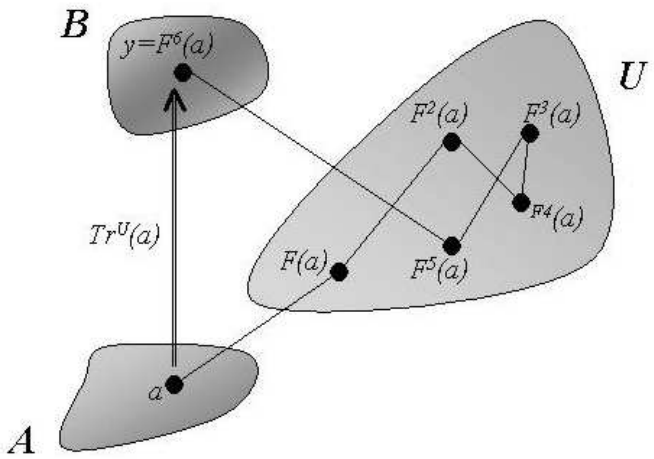
Why is this of interest to a category-theorist ??

The 'particle-style trace' or 'trace by iteration' :

The intuition of 'eliminating a subspace by iteration'

A partial function $F : A \uplus U \to B \uplus U$ may be written as a matrix :

$$F = \begin{pmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{pmatrix} \quad \text{where} \quad \begin{cases} f_{11} : A \to B & f_{12} : U \to B \\ f_{21} : A \to U & f_{22} : U \to U \end{cases}$$



The Particle-style Trace 'eliminates the shared subobject U'

$$Tr^U \begin{pmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{pmatrix} = f_{11} + \sum_{i=0}^{\infty} f_{12} f_{22}^i f_{21} \; : \; A \to B$$

Iterative traces & order theory

Given an abstract machine

$$\mathcal{M} = (X, \mathcal{R} \in \mathbf{pFun}(X, X))$$

assume $X = A \uplus U$

Define the **restriction** of $[\mathcal{M}]$ to $A \subseteq X$ as

$$[\mathcal{M}]_A = \{\ \mu \in [\mathcal{M}]\ :\ dom(\mu) = A = im(\mu)\ \}$$

similarly for the cycle-free semantics :

$$[\![\mathcal{M}]\!]_A = \{\ \mu \in [\![\mathcal{M}]\!]\ :\ dom(\mu) = A = im(\mu)\ \}$$

For *arbitrary* evolutions, $Tr^U_{A,A}(\zeta) \prec \zeta$

(up to the embedding $\mathbf{pFun}(A, A) \hookrightarrow \mathbf{pFun}(X, X)$).

For a given *cycle-free* evolution, $\eta \in [\![\mathcal{M}]\!]$, consider

$$\eta \downarrow_A = \{\ \mu \in [\![\mathcal{M}]\!]_A\ :\ \mu \prec \eta\ \}$$

Can show (up to an embedding of $\mathbf{pFun}(A, A)$ into $\mathbf{pFun}(X, X)$)

- Any $\mu \in \eta \downarrow_A$ is cycle-free.
- $Tr^U_{A,A}(\eta)$ is :
  1. cycle-free, and a member of $\eta \downarrow_A$.
  2. the **supremum** of $\eta \downarrow_A$.

Finding cycle-free evolutions

Good programming practice : every program subroutine has well-defined entry and exit points — *it is 'bad manners' to jump into, or break out of, a subroutine half way through!*

Interpretation : Given an abstract machine $\mathcal{M} = (X, \mathcal{R})$, we chose distinct subsets, S, T

—the (**Starting** and **Terminal**) subsets of the configuration space :

$$S, T \subseteq X \quad , \quad S \cap T = \emptyset$$

and consider all evolutions that take elements of S to elements of T :

$$[M]^S_T \;=\; \{\eta \in [\mathcal{M}] \;:\; dom(\eta) \subseteq S \;,\; im(\eta) \subseteq T\}$$

Trivially :

- $[M]^S_T$ is a *flat semigroup* :
  - it is closed under composition
  - contains the zero arrow
  - $\eta\mu = 0 = \mu\eta$ for all $\eta, \mu \in [M]^S_T$
- All evolutions in $[M]^S_T$ are cycle-free
  - Nilpotency is very easy to prove !
- $[M]^S_T$ has a top element, given by $Tr_{S,T}(\mathcal{R})$
  - again, up to some $\mathbf{pFun}(S, T) \hookrightarrow \mathbf{pFun}(X, X)$.

Interpretation :

We have specified (distinct) starting and halting subsets for the Abstract Machine $\mathcal{M}$

— we are now treating this as a 'black box', that takes inputs to outputs.

Further directions :

This is very much (!) work in progress !

'To Do' list :

- Learn some domain theory!
- Clarify relationship between *trace* and *partial orders* in the general case.
- Give concrete examples :
  - 'Black Box' semantics for computer programs,
  - Algebraic models for state machines (with feedback)
  - Geometry of Interaction -style systems.
- A study of *compositionality*.
- Apply in other categories :
  1. Restriction to partial *reversible* functions goes through without a problem
     — similar theory for reversible computations.

  2. Similarly for Relations
     — and hence non-determinism.

  3. A 'particle-style' trace exists for non-expansive maps on Hilbert space.
     — but preserving unitarity requires a global clock.

     Questions :
     - What is the correct notion of *configuration*, *cycle-freeness* or *termination* ?
     - What is the corresponding order theory, logic, or domain theory?