Machine Spaces: Axioms and Metrics

Jörg Zimmermann and Armin B. Cremers

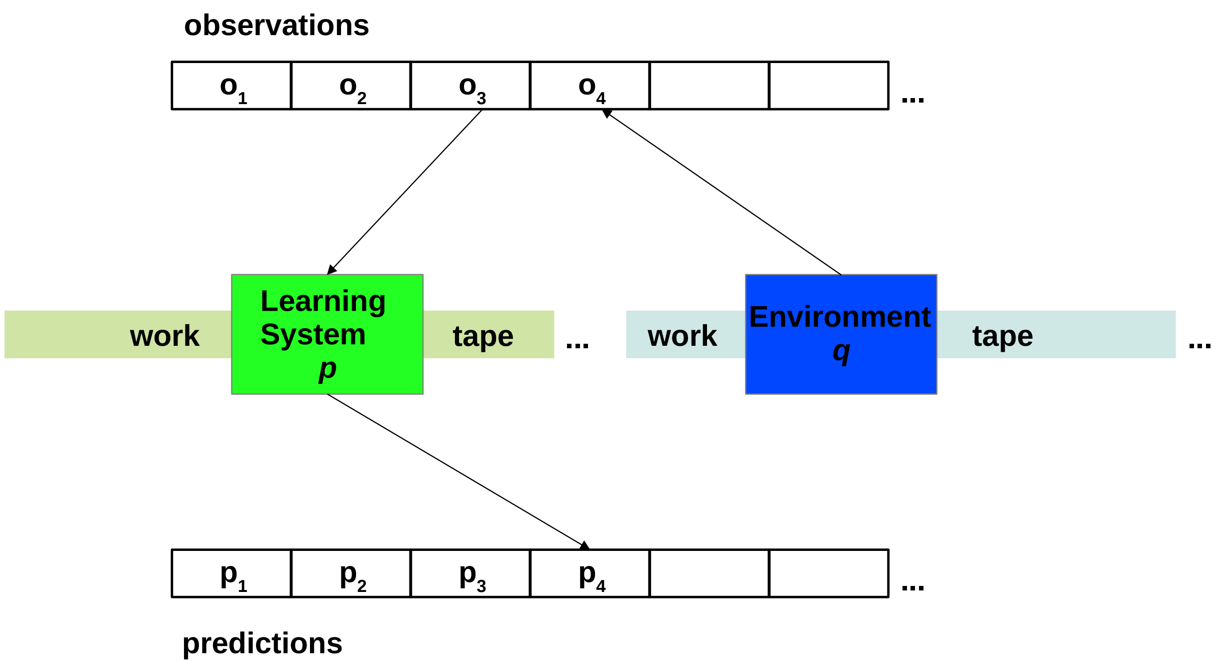
Institute of Computer Science
University of Bonn, Germany

Machine Spaces: Motivation and Context

- Defining a standard reference machine for universal induction.
- Investigation of the physical Church-Turing Thesis.

Machine Spaces: Motivation and Context

A learning system observing and predicting an environment:



Solomonoff Induction

- Bayesian learning in program space.
- Prior $\sim 2^{-|p|}$, $\quad |p| =$ length of program p in bits.
- p is executed on a fixed *universal Turing machine* U, which is called the *reference machine*.

But on finite data x, the choice of a universal reference machine can manipluate the posterior probability of a program consistent with x between ϵ and $1 - \epsilon$.

$\rightsquigarrow$ "natural" reference machines. But how can one define "natural" for machines?

$\rightsquigarrow$ axiomatic investigation of the "Machine Space".

Time Axioms

Structure of time *from a computational point of view*:

Thesis: time structure can be modelled by a *totally ordered monoid*:

(Associativity) $\forall t_1, t_2, t_3 : (t_1 + t_2) + t_3 = t_1 + (t_2 + t_3)$.

(Neutral Element) $\forall t : t + 0 = 0 + t = t$.

(Compatibility) $\forall t_1, t_2, t_3 : t_1 \leq t_2 \Rightarrow t_1 + t_3 \leq t_2 + t_3$ and $t_3 + t_1 \leq t_3 + t_2$.

$\rightsquigarrow$ time structures can be discrete, continuous, or transfinite.

$\rightsquigarrow$ ordinal numbers modeling a transfinite time structure have a non-commutative addidtion: $1 + \omega \neq \omega + 1$.

Machine Axioms

The "ontology" of the machine space:

- State space Σ
- Input space I
- Output space O
- Program space P
- *initializer*: a mapping $init$ from $P \times I$ to Σ
- *output operator*: a mapping out from Σ to O

Here "space" is used only figuratively. In the basic version of our formalization these "spaces" are just sets.

Machine Axioms

A machine wrt. a time structure T and a state space Σ is a mapping M from $\Sigma \times T$ to Σ (denoted by $M_t(s)$).

- Subset $HALT$ of Σ. States in $HALT$ will be used to signal termination of a computation.
- $TERM_M(s)$: denotes the set of time points t with $M_t(s) \in HALT$.

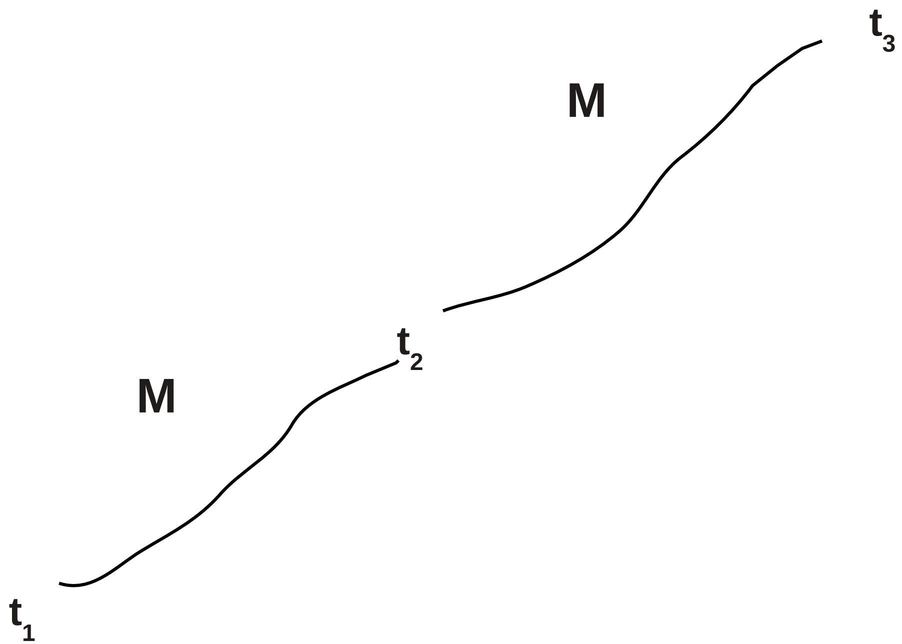
Machine Axioms

(Start) $\forall s \in \Sigma : M_{0^{(t)}}(s) = s$ (i.e., $M_{0^{(t)}} = id_\Sigma$),

(Action) $\forall t_1, t_2 \in T : M_{t_1+t_2} = M_{t_2} \circ M_{t_1}$.

These two axioms state that the time monoid is operating on the state space via machine M.

Machine Axioms



The Action Axiom implies that M traces out trajectories in state space and does not jump from START to STOP

Machine Axioms

(Stop) $\forall s \in \Sigma, t_1, t_2 \in T : t_1 \in TERM_M(s)$ and $t_1 \leq t_2 \Rightarrow M_{t_1}(s) = M_{t_2}(s)$.

That is, after reaching a termination state, nothing changes anymore, i.e., termination states are fixpoints of the machine dynamics.

Machine Axioms

(Well-Termination) $\forall s \in \Sigma : TERM_M(s) \neq \emptyset \Rightarrow \exists t_1 \in TERM_M(s)\ \forall t_2 \in TERM_M(s) : t_1 \leq t_2$.

Well-termination requires that if a machine terminates on s, i.e., reaches $HALT$ for some point in time, then there is a first point in time when this happens.

If $TERM_M(s)$ is non-empty, its least element is denoted by t^*.

Implementation

Definition: A function $f : I \to O$ is *implemented* by $p \in P$ on M iff

$$f(x) = out(M_{t^*}(init(p, x)))$$

for all $x \in I$.

Functions f which are implementable on a machine M are called "M-computable". $[p]_M$ denotes the (partial) function implemented by p on M.

Measuring Resources: Time

Let $time_p^M(x) = min(TERM_M(init(p, x)))$.

Then define a transfer function between machines as follows:

$\tau : T \to T$ is an *admissible time transfer function (attf)* from M_1 to M_2 iff τ is monotone and $\forall p_1 \in P_1 \exists p_2 \in P_2 : [p_1]_{M_1} = [p_2]_{M_2}$ and $\forall x \in I : time_{p_2}^{M_2}(x) \leq \tau(time_{p_1}^{M_1}(x))$.

Transfer functions will be used to measure the "time distance" of two machines in machine space.

M-dependent Computability and Complexity

A machine M defines implicitly a set of functions, the M-computable functions:

$$COMP(M) = \{f | f : I \to O, f \text{ is } M - computable\}$$

But it also defines complexity classes in analogy to the classical complexity classes:

$$TIME_M(g) = \{f | f \in COMP(M), [p]_M = f, time_p^M(x) \leq g(x)\}$$

Metrics on Machine Space

M_1 and M_2 are time-compatible if they operate on the same time structure, input space and output space.

A generalized metric $\Delta^{(t)}$ on machine space is now defined as follows:

$\Delta^{(t)}(M_1, M_2) = \{\tau | \tau$ is an attf from M_1 to $M_2\}$.

This roughly corresponds to statements like: "Machine A can simulate machine B with a logarithmic factor".

Metrics on Machine Space

One can combine and compare sets of functions much like single functions. Let $\alpha, \beta \subseteq T^T$:

$\alpha \circ \beta := \{\tau_1 \circ \tau_2 | \tau_1 \in \alpha, \tau_2 \in \beta\}$.

$\alpha \leq \beta$ iff $\forall \tau_2 \in \beta \exists \tau_1 \in \alpha : \tau_1 \leq \tau_2$.

Metrics on Machine Space

- By these definitions sets of attfs become a directedly ordered monoid (dom).
- Directed monoids can be used as ranges for generalized metrics, allowing many standard constructions of topology.

Our metric can be classified as a *dom-valued directed pseudometric*, satisfying the following triangle inequality:

$$\Delta^{(t)}(M_1, M_3) \leq \Delta^{(t)}(M_2, M_3) \circ \Delta^{(t)}(M_1, M_2).$$

Open Problems

- Additional Axioms?
- How to avoid that all the work is done by input and ouput operators?
- How to define a "Standard Reference Machine" (SRM), which can serve as a anchor point for concrete complexity statements?

Open Problems

- Idea: Define the SRM as the "center" of the smallest ball enclosing current real world computing machines.