Formalisation of metamorph Reinforcement Learning

Iago Bonnici, Abdelkader Gouaich, Fabien Michel

▶ **To cite this version:**

Iago Bonnici, Abdelkader Gouaich, Fabien Michel. Formalisation of metamorph Reinforcement Learning. [Technical Report] LIRMM (UM, CNRS). 2018. hal-01924642

HAL Id: hal-01924642

https://hal-lara.archives-ouvertes.fr/hal-01924642

Submitted on 5 Dec 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

LIRMM
M. Iago BONNICI
Supervisors: Dr. Abdelkader GOUAICH
Dr. Fabien MICHEL
November 22, 2018
Montpellier, FRANCE

Formalisation of metamorph Reinforcement Learning

technical report

This technical report describes the formalisation of a particular Reinforcement Learning (RL) situation that we call "metamorph" (mRL). In this situation, the signature of the learner agent, *i.e.* its set of inputs, outputs and feedback slots, can change over the course of learning.

RL can be viewed as *signal processing*, because the learner agent transforms the inputs/feedbacks signals it is continuously fed with into output signals. The following formalisation is therefore concerned with signals description and the transformation from one signal to another. Also, since the signature of the agent is expected to change, we get concerned in the definition of what is a "signature" and a "signature change".

In the first part, we describe mRL learning context, or how the metamorph agent is embedded into its environment and interacts with it. In the second part, we describe one generic example of a metamorph learner agent: a dynamical computational graph that could theoretically be used in controlling the agent. In the last part, we reformulate the classical problem of RL, a.k.a. "maximizing feedback" in terms of this formalised mRL.

Contents

Learning Context

1 Values and their domains

We call *domain* a tuple $d = (d.i,\ d.\sigma)$, where:

- $d.\sigma$ is a set
- $i.\sigma$ is an index, so that $d.\sigma = d'.\sigma \implies d = d'$

Note: within this whole document, the dot notation $a.b$ does not represent a product between a and b, but the b attribute of object a.

Let $\mathscr{D}$ be the set of all domains.
Let $\underline{\mathscr{D}}$ be its flattened version containing all values from all domains:

$$\underline{\mathscr{D}} = \bigcup_{d \in \mathscr{D}} d.\sigma \tag{1}$$

Let $\perp$ be a joker, null value belonging to none of the domains:

$$\perp \notin \underline{\mathscr{D}} \tag{2}$$

2 Flows

A *flow* is a function associating a value to each real *date* t:

$$h : \begin{cases} \mathbb{R}^+ \to D \\ t \mapsto h(t) \end{cases} \tag{3}$$

3 Sequence approximation

For any precision $\varepsilon \in \mathbb{R}^{+*}$, a flow h may be approximated by a sequence $({}^{\varepsilon}h_n)_{n \in \mathbb{N}}$ such that:

$$\forall n \in \mathbb{N}, \quad {}^{\varepsilon}h_n = h(n\varepsilon) \tag{4}$$

Note that the initial value is the same for all precisions:

$$\forall \varepsilon \in \mathbb{R}^{+*},\ {}^{\varepsilon}h_0 = h(0) = h_0 \tag{5}$$

In the next, we will often work with sequences and discrete time $n \in \mathbb{N}$, with the idea that they approximate continuous flows as $\varepsilon \to 0$.

4 Flow determination

Let $(u_n)_{n\in\mathbb{N}}$ be a sequence in D. For any n, its "past" values live in $D^{[\mathbb{N}]}$ with:

$$D^{[\mathbb{N}]} = \bigcup_{i\in\mathbb{N}} D^i \tag{6}$$

$$\text{so that } \forall n \in \mathbb{N},\ (u_0,\ \ldots,\ u_n) \in D^{[\mathbb{N}]} \tag{7}$$

Let f be a function associating a value in D to each precision and each past.

$$f : \begin{cases} \mathbb{R}^{+*} \times D^{[\mathbb{N}]} & \to\ D \\ (\varepsilon,\ (u_0,\ \ldots,\ u_n)) & \mapsto\ f_\varepsilon(u_0,\ \ldots,\ u_n) \end{cases} \tag{8}$$

Let h,k be two flows in D,D': we say that *f determines k from h* when all sequences that f generates from approximations of h are approximations of k:

$$\forall \varepsilon \in \mathbb{R}^{+*},\ \forall n \in \mathbb{N}^*,\quad {}^{\varepsilon}k_n = f_\varepsilon({}^{\varepsilon}h_0,\ \ldots,\ {}^{\varepsilon}h_n) \tag{9}$$

This may be written another, graphical way:

$$h \text{ —— } (f) \longrightarrow k \tag{10}$$

A slightly different case is needed to bootstrap circular situations: we say that *f initially determines k from h* when the sequence determination is shifted towards zero:

$$\forall \varepsilon \in \mathbb{R}^{+},\ \begin{cases} {}^{\varepsilon}k_0 = f_\varepsilon(\varnothing) \\ \forall n \in \mathbb{N}^*,\ {}^{\varepsilon}k_n = f_\varepsilon\left({}^{\varepsilon}h_0,\ \ldots,\ {}^{\varepsilon}h_{n-1}\right) \end{cases} \tag{11}$$

Which is written:

$$h \text{ —— } (f^*) \longrightarrow k \tag{12}$$

5 Graphical aliases

Fleshing this graphical notation, we will use the following alias when k is determined by the past of h and h' combined:

$$\begin{array}{ccc} h & & \\ & \searrow & \\ & (f) \longrightarrow & k \\ & \nearrow & \\ h' & & \end{array} \tag{13}$$

$$\forall \varepsilon \in \mathbb{R}^{+*},\ \forall n \in \mathbb{N}^*, \quad {}^{\varepsilon}k_n = f_{\varepsilon}\left(\left({}^{\varepsilon}h_0,\ {}^{\varepsilon}h'_0\right),\ \ldots,\ \left({}^{\varepsilon}h_n,\ {}^{\varepsilon}h'_n\right)\right) \tag{14}$$

And the following alias when f determines both k and k' from h:

$$\begin{array}{ccccc} & & & & k \\ & & & \nearrow & \\ h & \text{——} & (f) & & \\ & & & \searrow & \\ & & & & k' \end{array} \tag{15}$$

$$\forall \varepsilon \in \mathbb{R}^{+*},\ \forall n \in \mathbb{N}^*, \quad \left({}^{\varepsilon}k_n,\ {}^{\varepsilon}k'_n\right) = f_{\varepsilon}({}^{\varepsilon}h_0,\ \ldots,\ {}^{\varepsilon}h_n) \tag{16}$$

6 Flow determination determination

When the function f determining one flow from another is itself determined by another flow, we will use this construct:

$$\begin{array}{ccccc} h' & \text{——} & (F) & & \\ & & \downarrow & & \\ h & \text{——} & (f) & \longrightarrow & k \end{array} \tag{17}$$

Meaning that there exists a flow of determining functions $f : \mathbb{R}^+ \to \mathscr{F}\left(D^{[\mathbb{N}]} \to D\right)$, approximated for each precision ε by a sequence $({}^{\varepsilon}f_n)_{n\in\mathbb{N}}$ so that:

$$\forall \varepsilon \in \mathbb{R}^{+*},\ \forall n \in \mathbb{N},\ \begin{cases} {}^{\varepsilon}f_n = F\left({}^{\varepsilon}h'_0,\ \ldots,\ {}^{\varepsilon}h'_n\right) \\ {}^{\varepsilon}k_n = {}^{\varepsilon}f_n(h_0,\ \ldots,\ h_n) \end{cases} \tag{18}$$

7 States and Multiple flows

We call *state* a tuple $s = (s.\Delta,\ s.\nu)$, where:

- $s.\Delta$ is a subset of $\mathscr{D}$ that we call the state *signature*
- $(s.\nu_d)_{d \in s.\Delta}$ is a family of *values* indexed by $s.\Delta$, such that each value belongs to the set of the corresponding domain:

$$\forall d \in s.\Delta, \quad s.\nu_d \in d.\sigma \tag{19}$$

Let $\mathscr{S}$ be the set of all states.

A flow of states $h : \mathbb{R}^+ \to \mathscr{S}$ is called a *multiple flow*. It carries two interesting informations:

- a flow of domains that we call the *signature* of h. We write it $h.\Delta$:

$$h.\Delta : \begin{cases} \mathbb{R}^+ \to 2^{\mathscr{D}} \\ t \mapsto h.\Delta(t) = h(t).\Delta \end{cases} \tag{20}$$

- a flow of values $h.\nu$, at any time consistent with the signature:

$$h.\nu : \begin{cases} \mathbb{R}^+ \to \underline{\mathscr{D}}^{[\mathbb{N}]} \\ t \mapsto h.\nu(t) = h(t).\nu \end{cases} \tag{21}$$

8 Subflows

A multiple flow $h : \mathbb{R}^+ \to \mathscr{S}$ can also be seen as / is isomorphic to a family of subflows $(h_d)_{d \in \mathscr{D}}$, where each flow $h_d : \mathbb{R}^+ \to d.\sigma \cup \{\perp\}$ associates the value taken by h in the domain d on each date, and $\perp$ if d is not part of the signature of h on this date:

$$\forall d \in \mathscr{D},\ \forall t \in \mathbb{R}^+, \quad h_d(t) = \begin{cases} h.\nu(t)_d \text{ if } d \in h.\Delta(t) \\ \perp \text{ if not} \end{cases} \tag{22}$$

In other words, h can represents many flows and h_d "selects" the subflow corresponding to one domain d.

9 Partial flow determination

Let k be a multiple flow, one such construct will be used:

$$h \text{ —— } (f) \xrightarrow{k} \overset{\rightarrow k.\Delta \qquad k.\nu \leftarrow}{} (f') \text{ —— } h' \tag{23}$$

It states that the signature of k and its values are determined by two different sources $(h,\ f)$ and $(h',\ f')$:

$$\forall \varepsilon \in \mathbb{R}^{+*},\ \forall n \in \mathbb{N},\ \begin{cases} {}^{\varepsilon}k.\Delta_n = f_{\varepsilon}({}^{\varepsilon}h_0,\ \ldots,\ {}^{\varepsilon}h_n) \\ {}^{\varepsilon}k.\nu_n = f'_{\varepsilon}({}^{\varepsilon}h'_0,\ \ldots,\ {}^{\varepsilon}h'_n) \end{cases} \tag{24}$$

10 Bit flows and Contiguous multiple flows

Let $h_d : \mathbb{R}^+ \to d.\sigma \cup \{\perp\}$ be a subflow in domain d.

h_d is said to be a *bit* flow if and only if its value is always $\perp$ except on one contiguous window in $\mathbb{R}^+$. *I.e.*, there exists $t_0,\ t_1 \in \overline{\mathbb{R}}^+$ (t_1 may be infinite) such that:

$$\forall t \in \mathbb{R}^+,\ \begin{cases} \text{if } t \in\ ]t_0,\ t_1[,\ \ h_d(t) \in d.\sigma \\ \qquad\qquad \text{else } h_d(t) = \perp \end{cases} \tag{25}$$

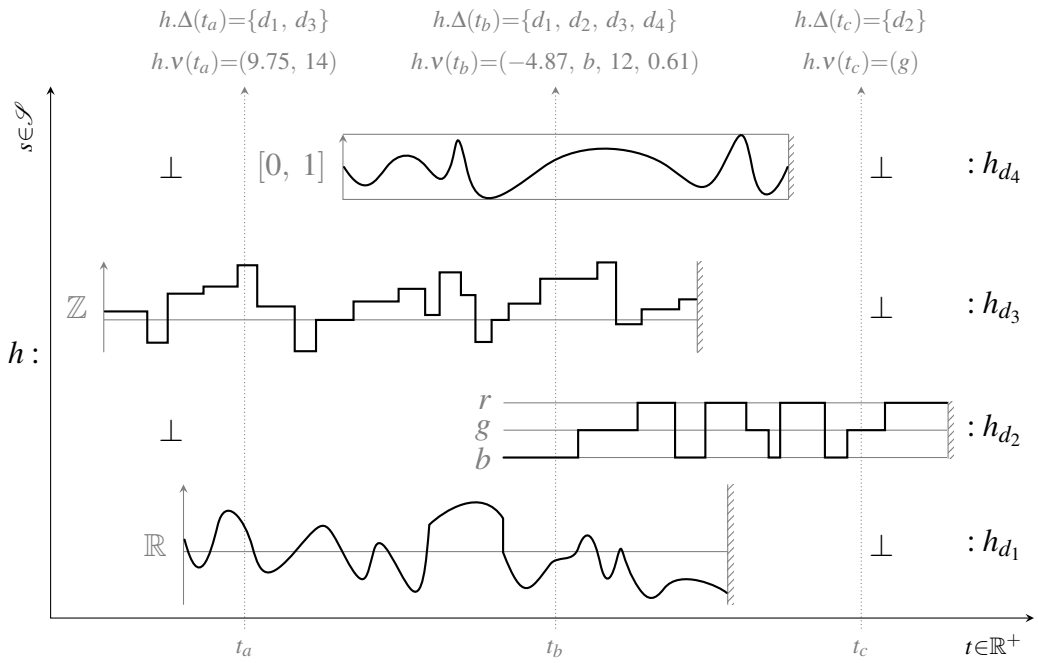
The domain d is said to be *active* at all dates $t \in]t_0,\ t_1[$, regarding to the subflow h_d, or else it is *inactive*.

When approximated with precision ε, the activation window translates to the corresponding discrete window $[\![{}^{\varepsilon}n_0,\ {}^{\varepsilon}n_1[\![\subset \overline{\mathbb{N}}$, ${}^{\varepsilon}n_0$ being the date of first non-$\perp$ value in ${}^{\varepsilon}h$, and ${}^{\varepsilon}n_1$ the date after the last non-$\perp$ value or ∞.

Now let $h : \mathbb{R}^+ \to \mathscr{S}$ be a multiple flow.

h is said to be a *contiguous* multiple flow if and only if all its subflows h_d are bit flows.

A contigous multiple flow can be represented as figure 1.

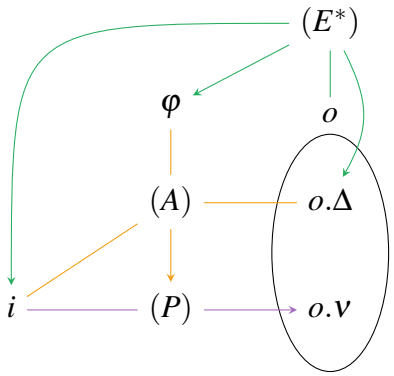


subflows \ evaluated in	t_a	t_b	t_c
$h_{d_4}:\mathbb{R}^+\to[0,\ 1]\cup\{\perp\}$	$h_{d_4}(t_a)=\perp$	$h_{d_4}(t_b)=0.61$	$h_{d_4}(t_c)=\perp$
$h_{d_3}:\mathbb{R}^+\to\mathbb{Z}\cup\{\perp\}$	$h_{d_3}(t_a)=14$	$h_{d_3}(t_b)=12$	$h_{d_3}(t_c)=\perp$
$h_{d_2}:\mathbb{R}^+\to\{r,\ g,\ b\}\cup\{\perp\}$	$h_{d_2}(t_a)=\perp$	$h_{d_2}(t_b)=r$	$h_{d_2}(t_c)=g$
$h_{d_1}:\mathbb{R}^+\to\mathbb{R}\cup\{\perp\}$	$h_{d_1}(t_a)=9.75$	$h_{d_1}(t_b)=-4.87$	$h_{d_1}(t_c)=\perp$

Figure 1: An example contiguous multiple flow h and its bit subflows $h_{d_1},\ \ldots,\ h_{d_4}$. Notice how each bit subflow is only active on a contiguous window of $\mathbb{R}^+$. In this example, the first four domains describe the following set of values for subflows: $d_1.\sigma=\mathbb{R},\ d_2.\sigma=\{r,\ g,\ b\},\ d_3.\sigma=\mathbb{Z},\ d_4.\sigma=[0,\ 1]$. The gray dotted lines correspond to three samples $t_a,\ t_b,\ t_c$ where the multiple flow h and its subflows are evaluated. Dashed areas mark the end of each bit subflow active window.

11 High-level agent representation

At the highest level, our learner agent can be represented by 4 contiguous multiple flows $(i,\ o,\ \varphi,\ P)$, with the following determination scheme:



(26)

Here:

- E represents the environment in which the agent is immersed
- i represents the agent's inputs or *sensors*: their nature may change in time as the signature $i.\Delta$ evolves.
- o represents the agent's outputs or *actuators*: their nature may also change in time as $o.\Delta$ evolves.
- φ represents the agent's *objectives*, a continuously fed evaluation of the actions it undertakes. They also may change in nature as $\varphi.\Delta$ evolves.

Note that the environment determines i, φ and $o.\Delta$, so the agent cannot decide the data it is fed with or its output signature.

- P represents the agent instant behaviour. It is an inner computational procedure determining the output values based on all input history.
- A is the abstract agent strategy, constantly adapting its behaviour based on environmental information.

Only two objects are not depending on time here: the environment E and the inner agent strategy A. E will be defined by user, and it is our job to design A (see objectives later).

Computational graph: an example of metamorph agent

In this chapter, we describe a graph process that can be translated into the inner reaction P of our learner agent.

Computational graphs are elements of Γ, that we will progressively describe hereafter.

12 Abuse of notation

In the next, three objects will essentially represent the same thing: the inner behaviour of the agent:

- One flow $P : \mathbb{R}^+ \to \Gamma$, assigning one computational graph to each date.
- One flow $P : \mathbb{R}^+ \to \mathscr{F}\left(\mathscr{S}^{[\mathbb{N}]} \to \underline{\mathscr{D}}^{[\mathbb{N}]}\right)$ of determining functions, determining output values $o.\nu$ from input pasts.
- One contiguous multiple flow $P : \mathbb{R}^+ \to \mathscr{S}$ describing the evolution of every intermediate value in the flowing computational graph.

Since they are conceptually related but still different in nature, we think that using the same symbol for all of them will cause less confusion than using one different symbol for each. As such, we will refer to each of them with the same letter P, and let the reader rely on the context to figure them out.

Also, considering P as a multiple flow and P_d its subflow in domain d, we will refer to the sequence approximation $({}^{\varepsilon}P_{dn})_{n\in\mathbb{N}}$ with the simpler expression $(d_n)_{n\in\mathbb{N}}$ so that double subscript is avoided.

As a consequence, in such a context, the symbol d refers both to the domain $d \in \mathscr{D}$ and to the corresponding subflow sequence approximation $d = {}^{\varepsilon}P_d$.

13 Operations

A computational graph in Γ is a structure that typically contains "operations".

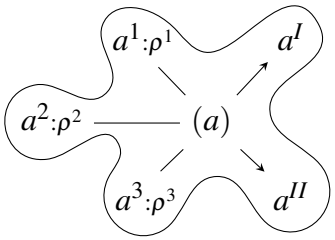
We call *operation* an object of the form:

$$\big(a,\ \big((a^1,\ \rho^1),\ \ (a^2,\ \rho^2),\ \ldots,\ (a^p,\ \rho^p)\big),\ (a^I,\ \ a^{II},\ \ldots,\ a^q)\big) \tag{27}$$

Where:

- All $a^i \in \mathscr{D}$ are domains, and a P_{a^i} subflow corresponds to each of them.
- $\big((a^1,\ \rho^1),\ \ldots,\ (a^p,\ \rho^p)\big)$ are called the operation *inputs*
- $\big(a^I,\ \ldots,\ a^q\big)$ are called the operation *outputs*.
- All $\rho^i \in a^i.\sigma$ are called *rest values*, and belong to their corresponding input domain.
- $a : \underline{\mathscr{D}}^{p^{[\mathbb{N}]}} \to \underline{\mathscr{D}}^q$ is a determining function called the operation *formula*. Its exact signature is $a : (a^1.\sigma \times \cdots \times a^p.\sigma)^{[\mathbb{N}]} \to a^I.\sigma \times \cdots \times a^q.\sigma$.

The computational graph process P, like any flow, can be approximated by a sequence ${}^{\varepsilon}P$. At any date $n \in \mathbb{N}$, the instant computational graph ${}^{\varepsilon}P_n \in \Gamma$ may contain such an operation, so it may read somewhere:



$$\in {}^{\varepsilon}P_n = P(n\varepsilon) \tag{28}$$

This means that all domains a^i are being active within the multiple flow P at this date. In other words: the bit subflows P_{a^i} are in their non-$\bot$ window:

$$\begin{aligned} &\forall n \in \mathbb{N},\ \text{operation "a" present in } {}^{\varepsilon}P_n \iff \\ &\qquad \forall i \in \{1,\ \ldots,\ p,\ I,\ \ldots,\ q\},\quad a^i_n \neq \bot \implies a^i_n \in a^i.\sigma \end{aligned} \tag{29}$$

Also, the outputs of the operation are determined by its inputs histories:

$$\big(a^I_n,\ \ldots,\ a^q_n\big) = a\big((a^1_0,\ \ldots,\ a^p_0),\ \ldots,\ (a^1_n,\ \ldots,\ a^p_n)\big) \tag{30}$$

Operations are *solid* in that no input $(a^1,\ \rho^1),\ \ldots,\ (a^p,\ \rho^p)$, no output $a^I,\ \ldots,\ a^q$ and no formula a can be present in the graph if not all other elements of the operation are present.

Note: since the multiple flow P is contiguous, P_{a^i} are bit subflows. Thus, as a consequence of operation solidity, every operation can only appear once in the graph process P during a $]t_0,\ t_1[$ activity window. And this activity window is the same for all its related P_{a^i} bit subflows.

As a consequence, for each operation, there exists one maximal "arrival date" in the graph process, $t_0 \in \overline{\mathbb{R}}^+$, and a corresponding maximal "approximated arrival date" ${}^{\varepsilon}n_0 \in \overline{\mathbb{N}}$, such that:

$$\forall t < t_0, \quad P_{a^i}(t) = \bot \tag{31}$$

$$\forall n <\, {}^{\varepsilon}n_0, \quad a^i_n = P_{a^i}(n\varepsilon) = \bot \tag{32}$$

Operations are *autonomous* (in the sense of dynamical systems) in that their determining function a ignores the arrival date of the operation. In other words, their result is independent on the number of initial $\bot$ values, so $\forall n \geqslant {}^{\varepsilon}n_0$:

$$\begin{aligned} a\big((\bot,\ \ldots,\ \bot),\ \ldots,\ (\bot,\ \ldots,\ \bot),\ (a^1_{n_0},\ \ldots,\ a^p_{n_0}),\ \ldots,\ (a^1_n,\ \ldots,\ a^p_n)\big) \\ = a\big((a^1_{n_0},\ \ldots,\ a^p_{n_0}),\ \ldots,\ (a^1_n,\ \ldots,\ a^p_n)\big) \end{aligned} \tag{33}$$

This makes the operation capable of processing flowing information, gifted with a memory, but unaware of the absolute date.

14 Fluxes and delays

Operations in a computational graph in Γ are typically connected together via "fluxes". Operation formulae determine outputs subflows, and fluxes determine input subflows.

We call *flux* an object of the form:

$$\omega = (\delta,\ \mathrm{o},\ (\iota,\ \rho)) \tag{34}$$

Where

- $\delta \in \mathbb{R}^{+*}$ is a *delay* value
- o is an output
- $(\iota,\ \rho)$ is an input

At any date, the computational graph may contain such a flux, so it reads:

$$\mathrm{o} \xrightarrow{\ \delta\ } \iota{:}\rho \quad \in {}^{\varepsilon}P_n \quad \text{or} \quad \omega \in {}^{\varepsilon}P_n \tag{35}$$

This means that the subflow P_ι copies the subflow P_o with a delay δ as long as ω is part of the graph. However, it cannot copy values of P_o anterior to the apparition of ω in the graph, and will use its rest value ρ instead.

More formally, let $t_0 \in \mathbb{R}^+$ be the arrival date of ω into the graph:

$$\forall t > t_0, \quad P_\iota(t) = \begin{cases} \text{if } t - \delta \leqslant t_0 : \rho \\ \text{else} : P_\mathrm{o}(t-\delta) \end{cases} \tag{36}$$

Inputs cannot have more that one flux attached at any given date.

For any other input $(\iota,\ \rho)$ present in the graph, but with no flux attached, their corresponding subflow has the rest value ρ: $\forall t \in \mathbb{R}^+$,

$$\varnothing \quad \iota{:}\rho \qquad \in P(t) \quad \Longrightarrow \quad P_\iota(t) = \rho \tag{37}$$

With fluxes, operations can be connected to each other so that information gets processed by their formulae from one subflow to another.

Since each flux transfers information with a positive delay δ, no circular configuration of operations yields undefined subflows, and we can always find a precision $\varepsilon < \delta$ yielding approximated subflows ${}^{\varepsilon}P_d$ that are defined for all $n \in \mathbb{N}$ (see example section 15).

15 Embedding into environment: one example

At any time, the computational graph is connected to the agent input and output flows.

Each active subflow in the agent inputs corresponds to one free output in the graph, so that operations inputs can feed from them.

Each active subflow in the agent outputs corresponds to one free input in the graph, so that operations outputs can feed them.

For instance consider the following initial situation:

$$\begin{cases} (i_0,\ \varphi_0,\ o.\Delta_0) = E(\varnothing) \\ \qquad\qquad i.\Delta_0 = \{A,\ B,\ C\} \subset \mathscr{D} \\ \qquad\qquad i.\nu_0 = (A_0,\ B_0,\ C_0) \in A.\sigma \times B.\sigma \times C.\sigma \\ \qquad\qquad o.\Delta_0 = \{D,\ E,\ F\} \subset \mathscr{D} \end{cases} \tag{38}$$

Then, the minimal computational graph $P_0 \in \Gamma$ would be:

$$P_0 = \begin{matrix} A & & D{:}\rho^D \\ B & \varnothing & E{:}\rho^E \\ C & & F{:}\rho^F \end{matrix} \tag{39}$$

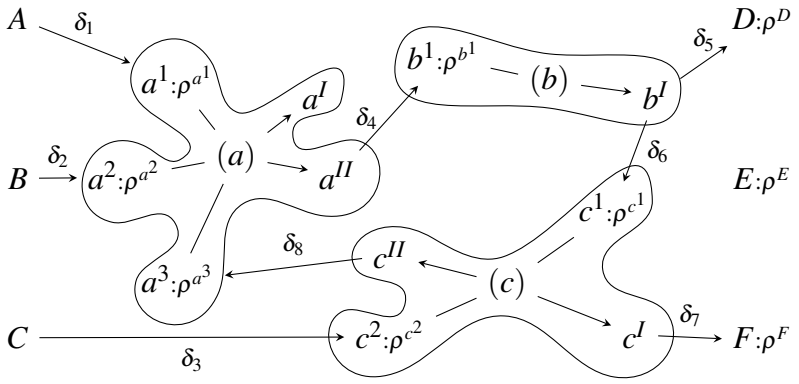
With no operations, only one free output per agent input, one free input per agent output and one rest value for each free input.

In this situation, only the rest values would be used, so there would be, $D_0 = \rho_D,\ E_0 = \rho_E,\ F_0 = \rho_F$. So, with any precision $\varepsilon \in \mathbb{R}^{+*}$, the first outputs values would be determined as:

$$^{\varepsilon}o.\nu_0 = (\rho_D,\ \rho_E,\ \rho_F) \in D.\sigma \times E.\sigma \times F.\sigma \tag{40}$$

However, the initial computational graph could also be more fleshed. In the following example, it contains three operations with their inner formulae a, b, c, connected to each other and to the agent environment *via* 8 fluxes:

$P_0 =$

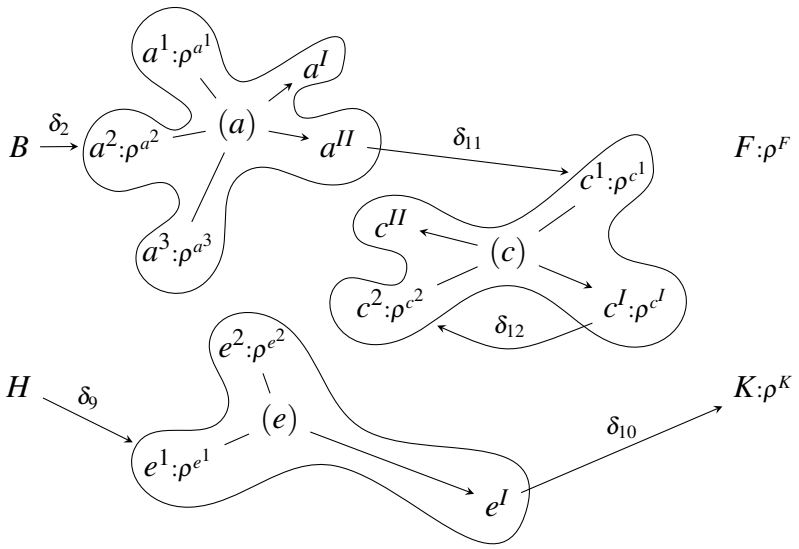


(41)

Then, assuming that the input signature stays constant ($i.\Delta_0 = {}^{\varepsilon}i.\Delta_1 = {}^{\varepsilon}i.\Delta_2$) then evolves into ${}^{\varepsilon}i.\Delta_3 = \{B,\ H\}$, and the output signature stays constant ($o.\Delta_0 = {}^{\varepsilon}o.\Delta_1 = {}^{\varepsilon}o.\Delta_2$) then evolves into ${}^{\varepsilon}o.\Delta_3 = \{F,\ K\}$, the graph could also stay con-

stant ($P_0 = {}^{\varepsilon}P_1 = {}^{\varepsilon}P_2$) then become, for example:

$${}^{\varepsilon}P_3 =$$



(42)

Let's assume that the ε-approximation of continuous time yields that the delays above correspond to these discrete delays:

delay ($\mathbb{R}^{+*}$)	δ_1	δ_2	δ_3	δ_4	δ_5	δ_6	δ_7	δ_8	δ_9	δ_{10}	δ_{11}	δ_{12}
ε-delay($\mathbb{N}^*$)	2	1	1	2	2	3	1	1	1	2	2	3

(43)

Then, the approximated multiple flows ${}^{\varepsilon}i$, ${}^{\varepsilon}P$, ${}^{\varepsilon}o$ can be computed like in the following tables:

	inputs			
n	A	B	C	H
0	A_0	B_0	C_0	$\perp$
1	A_1	B_1	C_1	$\perp$
2	A_2	B_2	C_2	$\perp$
3	$\perp$	B_3	$\perp$	H_3
4	$\perp$	B_4	$\perp$	H_4
5	$\perp$	B_5	$\perp$	H_5

	outputs			
n	D	E	F	K
0	ρ_D	ρ_E	ρ_F	$\perp$
1	ρ_D	ρ_E	c_0^I	$\perp$
2	b_0^I	ρ_E	c_1^I	$\perp$
3	$\perp$	$\perp$	ρ_F	ρ_K
4	$\perp$	$\perp$	ρ_F	ρ_K
5	$\perp$	$\perp$	ρ_F	e_3^I

Table 1: First few values of the example run: each column represents the ε-approximation of one input or output domain. The global agent signature has changed between approximation steps 2 and 3.

inner graph subflows

n	a^1	a^2	a^3	a^I	a^{II}	b^1	b^I	c^1	c^2	c^I	c^{II}	e^1	e^2	e^I
0	ρ_{a^1}	ρ_{a^2}	ρ_{a^3}	a_0^I	a_0^{II}	ρ_{b^1}	b_0^I	ρ_{c^1}	ρ_{c^2}	c_0^I	c_0^{II}	$\perp$	$\perp$	$\perp$
1	ρ_{a^1}	B_0	c_0^{II}	a_1^I	a_1^{II}	ρ_{b^1}	b_1^I	ρ_{c^1}	C_0	c_1^I	c_1^{II}	$\perp$	$\perp$	$\perp$
2	A_0	B_1	c_1^{II}	a_2^I	a_2^{II}	a_0^{II}	b_2^I	ρ_{c^1}	C_1	c_2^I	c_2^{II}	$\perp$	$\perp$	$\perp$
3	ρ_{a^1}	B_2	ρ_{a^3}	a_3^I	a_3^{II}	$\perp$	$\perp$	ρ_{c^1}	ρ_{c^2}	c_3^I	c_3^{II}	ρ_{e^1}	ρ_{e^2}	e_3^I
4	ρ_{a^1}	B_3	ρ_{a^3}	a_4^I	a_4^{II}	$\perp$	$\perp$	ρ_{c^1}	ρ_{c^2}	c_4^I	c_4^{II}	H_3	ρ_{e^2}	e_4^I
5	ρ_{a^1}	B_4	ρ_{a^3}	a_5^I	a_5^{II}	$\perp$	$\perp$	a_3^{II}	ρ_{c^2}	c_5^I	c_5^{II}	H_4	ρ_{e^2}	e_5^I

Table 2: First few values of the example inner multiple flow P. Each column represents the evolution of an operation input or output. The structure of the computational graph has changed between approximation steps 2 and 3.

Notice how input subflows are either constant rest flows or delayed copies of other subflows. For instance, a^1 contains a copy of B, while c^1, whose delays δ_6 then δ_{11} are long, is mostly resting to its rest value ρ_{c^1}.

Objective

In this chapter, we state the last constraints on the system so it can be simulated with ε-approximations and our objective can be expressed.

16 Finite multiple flows

A multiple flow $h : \mathbb{R}^+ \to \mathscr{S}$ is *finite* if and only if its signature is finite at any date:

$$h \text{ finite} \iff \forall t \in \mathbb{R}^+, \quad |h.\Delta(t)| \in \mathbb{N} \tag{44}$$

All constitutive multiple flows $(i,\ o,\ \varphi,\ P)$ of the agents we are interested in are finite.

17 Orderable objectives

In the agents we are interested in, all sets visited by the objective multiple flow φ are numerical, *i.e.* subsets of $\mathbb{R}$.

$$\forall t \in \mathbb{R}^+,\ \forall d \in \varphi.\Delta(t), \quad d.\sigma \text{ numerical} \tag{45}$$

18 Increasing objectives

A collection of E_i is given as problem data (it may be reduced to only one E). Our goal is to design the inner agent strategy A such that all objective subflows φ_d are maximal.

More formally, for any bit subflow φ_d, in any environment E_i, let $]t_0,\ t_1[$ be its activation window. Then, we want:

$$\int_{t_0}^{t_1} \varphi_d(t)\,\mathrm{d}t \text{ maximal} \tag{46}$$

In other words, no matter the experienced environment or the current signature, we want the agent A to be able to find which output signals will make all feedback maximal.
When this is achieved, we say that our agent keeps "learning", even though its objectives, inputs and outputs change over time.