# Learning any memory-less discrete semantics for dynamical systems represented by logic programs

Tony Ribeiro[1,2,4], Maxime Folschette[3], Morgan Magnin[2,4], and Katsumi Inoue[4]

[1] Independant Researcher
[2] Université de Nantes, Centrale Nantes, CNRS, LS2N, F-44000 Nantes, France
[3] Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France
[4] National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan

In biology, learning dynamics can corresponds to the identification of influence of genes, signals, proteins and molecules that can help biologists to understand their interactions. To tackle the problem of learning dynamical systems, we propose a method called LFIT [1], which stands for Learning from Interpretation Transition. This method constructs a model of the dynamics of a system from the observation of its state transitions (Figure 1). We assume that the system under study follows certain rules and that its observation, abstracted as state transitions, can be described by these rules. By observing the system's transitions, we can infer the rule structure of the system and construct a logical representation. The method and the algorithm **GULA** we propose in [2] allow to learn discrete multi-valued systems dynamics under a large range of update semantics including synchronous, asynchronous and more complex ones. The learned model being human readable, it can be used to get insight about the dynamical relations between the system components.

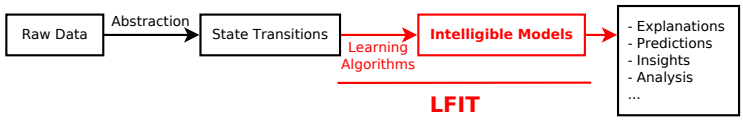


Fig. 1: LFIT automatically models a system dynamics from its state transitions.

LFIT relies on a representation of the system consisting of a logic program, which is a set of rules. Each rule has the following form:

$$a^{val_a} \leftarrow b^{val_b} \wedge c^{val_c} \wedge \cdots \qquad \text{or equivalently:} \qquad a^{val_a} \leftarrow \{b^{val_b}, c^{val_c}, \cdots\}$$

where $a$, $b$, $c$, ..., are variables and $val_a$, $val_b$, $val_c$, ..., are values assigned to the variables, constrained by their domains. The rule above has the following meaning: variable $a$ can take the value $val_a$ in the next dynamical step if variable $b$ (resp. $c$, ...) has value $val_b$ (resp. $val_c$, ...) in the current dynamical step. The actual dynamics depends on the chosen semantics.

Regarding a set of observations, the optimal program we want to learn should both: (1) match the observations in a complete (all transitions are learned) and correct (no spurious transition) way; (2) represent only minimal necessary interactions (no overly-complex rules). **GULA** [2] guarantees to learn the optimal

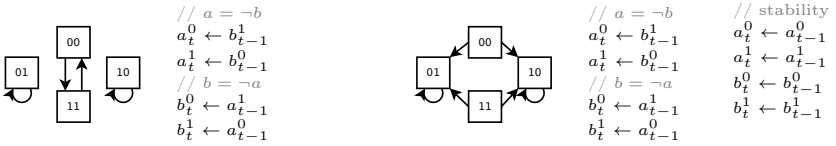


Fig. 2: (Left) State transitions diagram and its optimal program for an example with synchronous semantics. (Right) Same with asynchronous semantics.

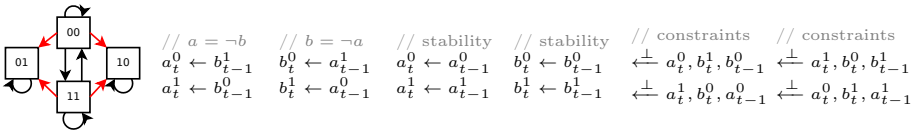


Fig. 3: State transitions diagram featuring observations (black) and transitions prevented by constraints (red), and its optimal program.

program of a set of transitions. For this, it starts from the program allowing any behavior in any situation (rules of the form $x \leftarrow \emptyset$ for each variable $x$) and performs minimal refinements on it (addition of atoms in conjunction) based on negative examples. Using the same update semantics that produces the observations, the optimal program reproduces the exact same transitions under a certain formal condition (detailed in Theorem 1 of [2]). For example, in Figure 2, both state transitions diagrams respect this condition and thus the observed dynamics can be represented by a logic program that is learnable by **GULA**. When the semantics that produce the observations is unknown or do not respect the condition mentioned above, we also propose a second algorithm named **Synchronizer** that additionally learns constraints allowing to reproduce any state transitions diagram. Figure 3 gives an example of such a case. Here, **GULA** is used to learn an over-set of transitions (black and red), and another special call to **GULA** (with pre-processing) allows to learn constraints preventing the spurious transitions (in red).

In [2], we also provided heuristics to use this approach on large and noisy, along with theoretical results that show the correctness of our approaches and practical evaluation performed on benchmarks from biological literature. The source code of all our LFIT algorithms are available as free software at `https://github.com/Tony-sama/pylfit` under the GPL-3.0 license. A user-friendly API allows to easily use LFIT on different kinds of datasets and is already being used in several research collaborations.

## References

1. Inoue, K., Ribeiro, T., Sakama, C.: Learning from interpretation transition. Machine Learning **94**(1), 51–79 (2014)
2. Ribeiro, T., Folschette, M., Magnin, M., Inoue, K.: Learning any memory-less discrete semantics for dynamical systems represented by logic programs. Machine Learning (2022)