# Learning From Interpretation Transitions with Unknowns

Tony Ribeiro[1,3,4], Maxime Folschette[2], Morgan Magnin[1,3], Kotaro Okazaki[4], Kuo-Yen Lo[4], Antoine Roquilly[5,6], Jérémie Poschmann[6], and Katsumi Inoue[3,1]

[1] Centrale Nantes, CNRS, Laboratoire des Sciences du Numérique de Nantes, LS2N, UMR 6004, F-44000 Nantes, France
[2] Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France
[3] National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
[4] SONAR Inc., 8-16-6, Ginza, Chuo-Ku, Tokyo 104-0061, Japan
[5] Nantes Université, CHU Nantes, INSERM, Anesthesie Réanimation, F-44000 Nantes, France
[6] Nantes Université, CHU Nantes, Center for Research in Transplantation and Translational Immunology, UMR 1064, ITUN2, F-44000, Nantes, France

**Abstract.** One major challenge when learning dynamical models from actual time series data consists in tackling partial data. Learning from interpretation transition (LFIT) automatically constructs a model of the dynamics of a system from the observation of its state transitions. In this paper, we extend the LFIT framework to learn from transitions between partial states where some variable values are unknown. By modeling the unknown, we achieve an overestimation of the real system regarding both its dynamics and variables interactions. We show through theoretical results the correctness of our approaches and its effectiveness through an experimental evaluation on benchmarks from biological literature.

**Keywords:** XAI · logical modeling · uncertainty · dynamic systems

## 1 Introduction

One major challenge when learning dynamical models from actual time series data consists in tackling partial data. For example, in Systems Biology, when trying to contribute to the identification of biological markers critical to the development of hospital-acquired pneumonia [10], a major challenge for modelers relies in overcoming the absence of some data in existing cohorts. Some data may be missing, because some variables were not accessible by experimental means (or were simply not considered of interest at the time of collecting data on real cohorts). Learning from partial data implies to extend existing learning approaches to data with unknowns among the state variables. This challenge relates to a well-known problem in the field of discrete-event systems, which is

The 4th International Joint Conference on Learning & Reasoning, Nanjing University International Conference Center, Nanjing, China.

opacity [8]. Since two decades, opacity has raised much interest in cryptography [9] and model-checking [1,6]. In these contexts, systems are analyzed to ensure that certain states or sequences of states remain hidden or protected from an external agent. For example, this implies that a system's secret state cannot be inferred by observing its behavior. When dealing with models of biological networks, opacity is relevant to understand how certain genes or pathways can be hidden from experimental observations. Despite being key for making a bridge between logical approaches and real-life data, opacity has been the subject of very few studies in the field of Boolean networks [15] and even less in the field of logic programs.

While the possibility of learning from transitions of partial interpretations has been briefly illustrated in [5], there has been neither details about its formal semantics (in particular for a multi-valued setting) nor implemented systems in the literature. Research regarding learning from incomplete transitions of (complete but possibly noisy) interpretations, in the context of neurosymbolic learning [3,4,12,13,14] exists, but their settings are not the same as learning from transitions of partial interpretations. In this paper, we aim to bring a first milestone to such a topic, opening the way to various applications hereafter. To achieve this goal, we propose to extend the Learning From Interpretation Transition (LFIT) framework [5,16], an inductive logic programming paradigm that automatically builds a model of the dynamics of a system from the observation of its state-transitions.

Figure 1 illustrates the general LFIT learning process. Given some raw data, like time-series of gene expression, a discretization of those data in the form of state transitions is assumed. From those state transitions, according to the semantics of the system dynamics, several inference algorithms modeling the system as a logic program have been proposed. In this paper, we extend the LFIT framework to learn from transitions between partial states where some variable values are unknown. We propose a formal modeling for learning from transitions of partial multi-valued interpretations, to realize the framework to incorporate into the General Usage LFIT Algorithm (GULA) [16]. By modeling the unknown, we achieve an overestimation of the real system regarding both its dynamics and variables interactions. We show through theoretical results the correctness of our approaches and its effectiveness through an experimental evaluation on benchmarks from biological literature.

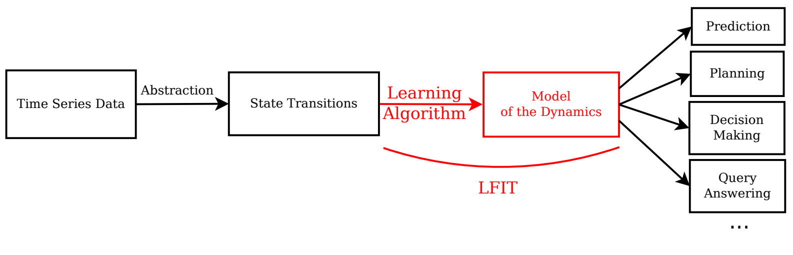

Fig. 1: Assuming a discretization of time series data of a system as state transitions, we propose a method to automatically model the system dynamics.

## 2 Dynamical Multi-Valued Logic Program

In this section, the concepts necessary to understand the modeling we propose in this paper are formalized. Let $\mathcal{V} = \{\mathrm{v}_1, \cdots, \mathrm{v}_n\}$ be a finite set of $n \in \mathbb{N}$ variables, $\mathcal{V}al$ the set in which variables take their values and $\mathsf{dom} : \mathcal{V} \to \{d \in \wp(\mathcal{V}al) \mid |d| \geq 2\}$ a function associating a domain (with at least two values) to each variable, with $\wp$ the power set. The atoms of multi-valued logic ($\mathcal{M}$VL) are of the form $\mathrm{v}^{val}$ where $\mathrm{v} \in \mathcal{V}$ and $val \in \mathsf{dom}(\mathrm{v})$. The set of such atoms is denoted by $\mathcal{A} = \{\mathrm{v}^{val} \in \mathcal{V} \times \mathcal{V}al \mid val \in \mathsf{dom}(\mathrm{v})\}$. Let $\mathcal{F}$ and $\mathcal{T}$ be a partition of $\mathcal{V}$, that is: $\mathcal{V} = \mathcal{F} \cup \mathcal{T}$ and $\mathcal{F} \cap \mathcal{T} = \emptyset$. $\mathcal{F}$ is called the set of feature variables, which values represent the state of the system at the previous time step $(t-1)$, and $\mathcal{T}$ is called the set of target variables, which values represent the state of the system at the current time step $(t)$. A $\mathcal{M}$VL rule $R$ is defined by:

$$R \quad = \quad \mathrm{v}_0^{val_0} \leftarrow \mathrm{v}_1^{val_1} \wedge \cdots \wedge \mathrm{v}_m^{val_m}$$

where $m \in \mathbb{N}$, and $\forall i \in [\![0; m]\!], \mathrm{v}_i^{val_i} \in \mathcal{A}$; furthermore, every variable is mentioned at most once in the right-hand part: $\forall j, k \in [\![1; m]\!], j \neq k \Rightarrow \mathrm{v}_j \neq \mathrm{v}_k$. The rule $R$ has the following meaning: the variable $\mathrm{v}_0$ can take the value $val_0$ in the next dynamical step if for each $i \in [\![1; m]\!]$, variable $\mathrm{v}_i$ has value $val_i$ in the current dynamical step. The atom on the left side of the arrow is called the head of $R$, denoted $\mathrm{head}(R) := \mathrm{v}_0^{val_0}$, and is made of a target variable: $\mathrm{v}_0 \in \mathcal{T}$. The notation $\mathrm{var}(\mathrm{head}(R)) := \mathrm{v}_0$ denotes the variable that occurs in $\mathrm{head}(R)$. The conjunction on the right-hand side of the arrow is called the body of $R$, written $\mathrm{body}(R)$, and all variables in the body are feature variables: $\forall i \in [\![1; m]\!], \mathrm{v}_i \in \mathcal{F}$. In the following, the body of a rule is assimilated to the set $\{\mathrm{v}_1^{val_1}, \cdots, \mathrm{v}_m^{val_m}\}$; we thus use set operations such as $\in$ and $\cap$ on it, and we denote $\emptyset$ an empty body. A dynamical multi-valued logic program ($\mathcal{DM}$VLP) is a set of $\mathcal{M}$VL rules.

**Definition 1 (Rule Domination).** *Let $R_1$, $R_2$ be $\mathcal{M}$VL rules. $R_1$ dominates $R_2$, written $R_1 \geq R_2$ if* $\mathrm{head}(R_1) = \mathrm{head}(R_2)$ *and* $\mathrm{body}(R_1) \subseteq \mathrm{body}(R_2)$.

The dynamical system we want to learn the rules of, is represented by a succession of states as formally given by Definition 2. We also define the "compatibility" of a rule with a state in Definition 5, and with a transition in Definition 6.

**Definition 2 (Discrete state).** *A complete discrete state $s$ on a set of variables $\mathcal{X}$ is a function from $\mathcal{X}$ to* $\mathsf{dom}(\mathrm{v})$. *It can be equivalently represented by the set of atoms $\{\mathrm{v}^{s(\mathrm{v})} \mid \mathrm{v} \in \mathcal{X}\}$ and thus we can use classical set operations on it. If $s$ is a complete discrete state, $s'$ is a partial discrete state iff $s' \subsetneq s$. We write $\mathcal{S}^{\mathcal{X}}$ to denote the set of all (partial or complete) discrete states on $\mathcal{X}$.*

Often, $\mathcal{X} \in \{\mathcal{F}, \mathcal{T}\}$. Moreover, we note $\mathrm{vars}(s) = \{\mathrm{v} \in \mathcal{X} \mid \exists val \in \mathsf{dom}(\mathrm{v}), \mathrm{v}^{val} \in s\}$ the set of variables that appear in a discrete state.

*Example 1.* Let $\mathcal{X} = \{a, b, c\}, dom(a) = dom(b) = \{0, 1\}$ and $dom(c) = \{0, 1, 2\}$.
- $\emptyset$ is a partial state, all variables value is unknown.

- $\{a^0\}$ is a partial state, both $b$ and $c$ value is unknown.
- $\{a^1, b^1\}$ is a partial state, $c$ value is unknown.
- $\{a^1, b^0, c^2\}$ is a complete state, all variable have a value.
- $\{a^0, a^1, c^0\}$ is not a discrete state, $a$ has multiple values.
- $\{a^0, b^0, c^0, c^2\}$ is not a discrete state, $c$ has multiple values.

**Definition 3 (States with Masking).** *We call state with masking a couple $s = (s_p, s_c) \in \mathcal{S}^{\mathcal{X}} \times \mathcal{S}^{\mathcal{X}}$ so that $s_c$ is complete and $s_p$ is partial so that $s_p \subsetneq s_c$. We denote $\mathcal{S}^{\mathcal{X}}_m$ the set of all states with masking. Moreover, we define:* $\mathsf{observed}(s) = s_p$ *and* $\mathsf{actual}(s) = s_c$ *and we extend these notations to a pair of states and a set of states:* $\mathsf{observed}((s, s')) = (\mathsf{observed}(s), \mathsf{observed}(s'))$ *and* $\mathsf{actual}((s, s')) = (\mathsf{actual}(s), \mathsf{actual}(s'))$*;* $\mathsf{observed}(S) = \{\mathsf{observed}(s) \mid s \in S\}$ *and* $\mathsf{actual}(S) = \{\mathsf{actual}(s) \mid s \in S\}$.

In other words, the state with masking $s = (s_p, s_c)$ represents a state that we can choose to observe as the result of an partial/imperfect observation, or as the ground truth. In particular, a couple of (complete, partial or with masking) discrete states $(s, s') \in \mathcal{S}^{\mathcal{F}} \times \mathcal{S}^{\mathcal{T}}$ is called a transition.

In practice, the ground truth is not accessible, but interesting knowledge about the system dynamics could still be derived from the partial observation. Providing some interesting guaranties over these derived dynamics is the goal of the following formalization. To simplify the graphical representation of states, each discrete state will be depicted by a sequence of numbers corresponding to the value of each variable. For instance, the complete state $\{a^0, b^0, c^0\}$ is represented by the sequence 000, while $\{a^1, b^0, c^0\}$ is represented as 100, and $\{a^1, b^1, c^1\}$ as 111. Unknown values will be symbolized by '?', allowing us to see at a glance which variables are unknown in each state. For example, the state $\{a^0\}$ is represented as 0??, and $\{c^1\}$ as ??1. The transition graphs in Figure 2 shows an illustration of a set $T$ of state transitions involving three boolean variables, highlighting both complete and partial states. On the left the observed state transitions $\mathsf{observed}(T)$ with some unknown values and on the right the corresponding $\mathsf{actual}(T)$ with no unknown.

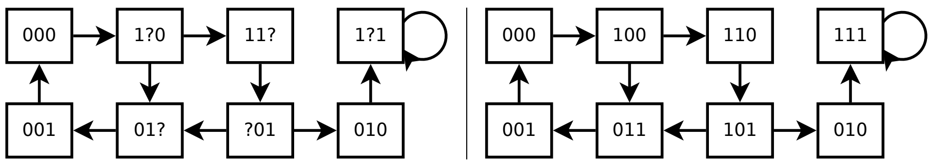

Fig. 2: Observed state transitions of a system with three boolean variables: discrete states represented by labeled boxes, arrows show transitions, and '?' denotes unknown values (left); the actual complete hidden state transitions (right).

**Definition 4 (State uncertain equality).** *Let $s, s' \in \mathcal{S}^{\mathcal{X}}$ be discrete states. The two states are uncertain equal, denoted $s \overset{\mathcal{X}}{\backsimeq} s'$ when*

$$\forall \mathrm{v}^{val} \in s, \forall \mathrm{w}^{val'} \in s', \mathrm{v} = \mathrm{w} \implies val = val'$$

States with different values on the same variable cannot be the same since a variable has only one value at a time. But states without such conflicting atoms can potentially be the same state, missing values causing the uncertainty. Figure 3 show examples of equality relationship under unknown using graphical representation of discrete state.

*Example 2.* Let $\mathcal{X} = \{a, b, c\}, dom(a) = dom(b) = \{0, 1\}$ and $dom(c) = \{0, 1, 2\}$.

- $\{a^0, b^0, c^0\} \overset{\mathcal{X}}{=} \{a^0, b^0, c^0\}$, exactly the same state, usual equality.
- $\{a^0, b^0, c^0\} \overset{\mathcal{X}}{\neq} \{a^0, b^1, c^0\}$, one difference on $b$ value, usual inequality.
- $\{a^0, b^0\} \overset{\mathcal{X}}{\backsimeq} \{a^0, b^0, c^0\}$, $c$ is unknown in first state, it could be $c^0$ or $c^1$.
- $\{a^0, b^0\} \overset{\mathcal{X}}{\backsimeq} \{a^0, b^0\}$, $c$ is unknown in both, can be same or different value.
- $\{a^0, b^0\} \overset{\mathcal{X}}{\neq} \{a^1, b^0\}$, $a$ is different, cannot be the same hidden state.
- $\{a^0, b^1\} \overset{\mathcal{X}}{\backsimeq} \{b^1, c^1\}$, $c$ is unknown in first state and $a$ is unknown in second state, no different value observed thus could be the same hidden state.

*Property 1.* Let $s1, s2$ be two discrete states with masking. From Definition 4 the following holds:

- $\mathsf{actual}(s1) \overset{\mathcal{X}}{=} \mathsf{actual}(s2) \implies \mathsf{observed}(s1) \overset{\mathcal{X}}{\backsimeq} \mathsf{observed}(s2)$
  - but $\mathsf{observed}(s1) \overset{\mathcal{X}}{\backsimeq} \mathsf{observed}(s2) \not\Longrightarrow \mathsf{actual}(s1) \overset{\mathcal{X}}{=} \mathsf{actual}(s2)$
- $\mathsf{observed}(s1) \overset{\mathcal{X}}{\not\backsimeq} \mathsf{observed}(s2) \implies \mathsf{actual}(s1) \overset{\mathcal{X}}{\neq} \mathsf{actual}(s2)$
  - but $\mathsf{actual}(s1) \overset{\mathcal{X}}{\neq} \mathsf{actual}(s2) \not\Longrightarrow \mathsf{observed}(s1) \overset{\mathcal{X}}{\not\backsimeq} \mathsf{observed}(s2)$

When two complete states are identical, their partial states are at least uncertain equal, as the introduction of unknowns cannot create differences. But observed states being uncertain equal does not mean that the actual states are

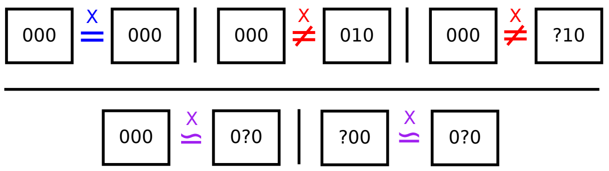

Fig. 3: Illustration of the concept of uncertain equality between discrete states. 000 and 000 are identical while 000 and 010 differ due to the second variable's value. Same for 000 and ?10, the unknown value does not matter here. States 000 and 0?0 are uncertain equal, as the '?' could be 0. Similarly, ?00 and 0?0 may represent the same state if both '?' are actually hiding 0.

certain equal since the unknown part can hide an actual difference. On the other hand, partial state inequality always corresponds to actual state inequality, because differences in observed variables are still present in the actual state. Once again, the reciprocal is not true since the difference between two complete states can be hidden by the masking.

**Definition 5 (Rule-state matching).** *Let $s \in \mathcal{S}^{\mathcal{F}}$. The $\mathcal{M}$VL rule $R$ matches $s$, written $R \sqcap s$, if* $\text{body}(R) \subseteq s$.

The final 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). Regarding the actual hidden observations, the optimal program should concurrently: (1) match them all, tolerating spurious transitions due to unknown values; (2) ensure that all original minimal rules are dominated, and none exceeds their complexity, allowing spurious rules only when unknowns render them indistinguishable from true optimal ones. The idea behind this is that when learning from observations that include unknowns, the obtained program should be an over-approximation of the behaviors allowed by the actual system. Moreover, revealing the real value of some variables (that is, adding new observations that are less hidden than the original), we should be able to iterate the learning part and get closer to the optimal program of the actual transitions. The following definitions formalize these desired properties.

**Definition 6 (Rule and program realization).** *Let $R$ be a $\mathcal{M}$VL rule and $(s, s') \in \mathcal{S}^{\mathcal{F}} \times \mathcal{S}^{\mathcal{T}}$. The rule $R$ realizes the transition $(s, s')$ if $R \sqcap s \wedge \text{head}(R) \in s'$. A $\mathcal{DM}$VLP $P$ realizes $(s, s')$ if $\forall \mathrm{v}^{val} \in s', \exists R \in P, \text{head}(R) = \mathrm{v}^{val} \wedge R$ realizes $(s, s')$. $P$ realizes a set of transitions $T \subseteq \mathcal{S}^{\mathcal{F}} \times \mathcal{S}^{\mathcal{T}}$ if $\forall (s, s') \in T, P$ realizes $(s, s')$.*

*Property 2.* Let $(s, s') \in \mathcal{S}^{\mathcal{F}}_m \times \mathcal{S}^{\mathcal{T}}_m$ be a transition of states with masking, and $P$ a $\mathcal{DM}$VLP. If $P$ realizes $\mathsf{observed}((s, s'))$ and $\forall \mathrm{v} \in \mathcal{T}, \mathrm{v} \notin \text{vars}(s') \implies \forall val \in \mathsf{dom}(\mathrm{v}), \exists R \in P, R$ realizes $(\mathsf{observed}(s), \mathsf{observed}(s') \cup \{\mathrm{v}^{val}\})$, then $P$ realizes $\mathsf{actual}((s, s'))$.

Property 2 states that when in a transition a target variable value is unknown, a $\mathcal{DM}$VLP needs to have a rule matching for each possible value of the variable in order to ensure to realize the corresponding complete hidden target state.

*Example 3.* Let $T$ be the set of transitions of Figure 2 (left).
- $a^1 \leftarrow \emptyset$ realizes all transition to $a^1$.
- $a^1 \leftarrow a^0$ realizes 000 to 1?0 and 010 to 1?1.
- $c^0 \leftarrow a^0, b^0, c^0$ also realizes 000 to 1?0.
- $b^1 \leftarrow a^1, b^1, c^1$ realizes no transitions of $T$, never 111 is observed.
- $b^0 \leftarrow a^1, c^1$ realizes no transitions of $T$, never $b^0$ is observed after $\{a^1, c^1\}$.

**Definition 7 (Conflict and Consistency).** *A $\mathcal{M}$VL rule $R$ conflicts with a set of transitions $T \subseteq \mathcal{S}^{\mathcal{F}} \times \mathcal{S}^{\mathcal{T}}$ when $\exists (s, s') \in T, \big(R \sqcap s \wedge \forall (s1, s2) \in T, s \overset{\mathcal{F}}{\backsimeq}$*

*$s1, \text{var}(\text{head}(R)) \in \text{vars}(s2) \land \text{var}(\text{head}(R))^{val} \neq \text{head}(R)$. Otherwise, $R$ is said to be consistent with $T$.*

*Example 4.* Let $T$ be the set of transitions of Figure 2.
- $a^1 \leftarrow \emptyset$ conflicts with $T$, since $a^1$ is not possible from 001.
- $a^1 \leftarrow a^0$ conflicts with $T$ for the same reason.
- $a^1 \leftarrow c^0$ is consistent with $T$, as 000, 1?0, 010 all have a transition to $a^1$.
- $b^0 \leftarrow a^1, b^1, c^1$ is consistent with $T$, as it matches no state and thus there is no contradiction.

A rule is considered in conflict when its condition matches a feature state $s$, but the rule's outcome is not present in the target state of any transition from $s$ or any other state that is uncertain equal to $s$ (i.e., both states could be partial observations of the same actual state). However, if the variable in the rule's head is unknown in the target state, it's not considered a conflict, as the unknown could be the rule head. Furthermore, if a feature state that matches the rule's condition never transitions to the rule's head, yet has uncertain equality with a feature state that does, it's not considered a conflict. This is because the states might be actually the same, thus making the rule correct.

*Property 3.* Let $R$ be a $\mathcal{M}$VL rule and $T \subseteq \mathcal{S}^{\mathcal{F}}_m \times \mathcal{S}^{\mathcal{T}}_m$ be a set of transitions with masking. It holds:
- $R$ conflicts with $\mathsf{observed}(T) \implies R$ conflicts with $\mathsf{actual}(T)$.
  - but $R$ conflicts with $\mathsf{actual}(T) \not\Longleftarrow R$ conflicts with $\mathsf{observed}(T)$.
- $R$ consistent with $\mathsf{actual}(T) \implies R$ consistent with $\mathsf{observed}(T)$.
  - but $R$ consistent with $\mathsf{observed}(T) \not\Longrightarrow R$ consistent with $\mathsf{actual}(T)$.

A rule consistent with a set of transitions remains consistent even when those transitions are masked by unknowns. The introduction of missing values cannot alter the original states, potentially reducing matches but not introducing new ones, thus preserving consistency. However, consistency with the observed transitions does not guarantee consistency with the actual ones, as different values might be concealed by unknowns.

A rule that conflicts with a set of transitions masked by unknowns will also conflict with the actual, corresponding hidden transitions. Revealing the missing values doesn't alter the visible atoms, so the rule condition will continue to be a subset of the feature state, maintaining the conflict. However, a rule conflicting with the actual transitions might not conflict with a masked version of them, introducing missing values in feature states can disrupt matching and remove conflict, while inserting unknowns in the target state could render the rule's head a feasible hidden value, rendering it consistent.

**Definition 8 (Suitable and optimal program).** *Let $T \subseteq \mathcal{S}^{\mathcal{F}}_m \times \mathcal{S}^{\mathcal{T}}_m$ a set of transitions with masking. A $\mathcal{DM}$VLP $P$ is suitable for $T$ if: $P$ is consistent with $\mathsf{observed}(T)$, $P$ realizes $\mathsf{observed}(T)$, $P$ realizes $\mathsf{actual}(T)$, and for any possible $\mathcal{M}$VL rule $R'$ consistent with $\mathsf{observed}(T)$, there exists $R \in P$ s.t. $R' \geq R$. If, in addition, for all $R \in P$, all the $\mathcal{M}$VL rules $R'$ belonging to $\mathcal{DM}$VLP suitable for $T$ are such that $R' \geq R$ implies $R \geq R'$, then $P$ is called optimal and denoted $P_{\mathcal{O}}(T)$.*

*Property 4.* Let $T \subseteq \mathcal{S}^{\mathcal{F}}_m \times \mathcal{S}^{\mathcal{T}}_m$ a set of transitions with masking.

$$\forall R \in P_{\mathcal{O}}(\mathsf{actual}(T)), \exists R' \in P_{\mathcal{O}}(\mathsf{observed}(T)), R' \geq R$$

$$\forall R' \in P_{\mathcal{O}}(\mathsf{observed}(T)), \forall R \in P_{\mathcal{O}}(\mathsf{actual}(T)), R \geq R' \implies R = R'$$

The optimal program accounting for unknowns represents an over-approximation of the actual optimal program without unknowns, encompassing both dynamics and actual rules. All transitions realized by the optimal program of the ground truth are also realized by the optimal program considering unknowns. Actual optimal rules or their generalizations can be discovered among unknowns, as all optimal rules on the ground truth are specializations of those in the optimal program considering unknowns.

Additionally, the optimal program accounting for unknowns doesn't contain overly specific rules: none are more specific that the optimal rules on the ground truth. But it can also include rules not found in the optimal set on the ground truth, that were generated because of the unknown values. These additional rules cannot be discarded without risking the loss of actual optimal rules or their parts, as they remain indistinguishable from true optimal ones due to unknown values observed.

*Example 5.* Optimal $\mathcal{DM}$VLP of the actual transitions from Figure 2 (right).

$$\begin{array}{lllll}
a^1_t \leftarrow c^0_{t-1} & a^0_t \leftarrow a^0_{t-1}, c^1_{t-1} & b^1_t \leftarrow a^1_{t-1}, b^0_{t-1} & b^0_t \leftarrow a^0_{t-1}, b^0_{t-1} & c^1_t \leftarrow a^1_{t-1} \\
a^1_t \leftarrow a^1_{t-1}, b^1_{t-1} & a^0_t \leftarrow a^1_{t-1}, b^0_{t-1} & b^1_t \leftarrow a^1_{t-1}, c^1_{t-1} & b^0_t \leftarrow a^0_{t-1}, c^1_{t-1} & c^1_t \leftarrow b^1_{t-1} \\
 & a^0_t \leftarrow b^0_{t-1}, c^1_{t-1} & b^1_t \leftarrow a^0_{t-1}, b^1_{t-1}, c^0_{t-1} & b^0_t \leftarrow a^1_{t-1}, b^1_{t-1}, c^0_{t-1} & c^0_t \leftarrow b^0_{t-1}
\end{array}$$

Optimal $\mathcal{DM}$VLP of the transitions with unknowns from Figure 2 (left).

$$\begin{array}{llllll}
a^1_t \leftarrow c^0_{t-1} & a^0_t \leftarrow a^1_{t-1} & b^1_t \leftarrow \emptyset & b^0_t \leftarrow \emptyset & c^1_t \leftarrow a^1_{t-1} & c^0_t \leftarrow a^1_{t-1} \\
a^1_t \leftarrow a^1_{t-1} & a^0_t \leftarrow b^1_{t-1} & & & c^1_t \leftarrow b^1_{t-1} & c^0_t \leftarrow b^0_{t-1} \\
a^1_t \leftarrow b^1_{t-1} & a^0_t \leftarrow c^1_{t-1} & & & c^1_t \leftarrow c^1_{t-1} & c^0_t \leftarrow c^1_{t-1}
\end{array}$$

In this scenario, some actual optimal rules, such as $a^1_t \leftarrow c^0_{t-1}$ and $c^1_t \leftarrow a^1_{t-1}$, remain decipherable despite the presence of unknowns. Furthermore, we can see that all rules of the actual optimal program are dominated, as in $a^1_t \leftarrow a^1_{t-1}, b^1_{t-1}$, which manifests partially through $a^1_t \leftarrow a^1_{t-1}$ and $a^1_t \leftarrow b^1_{t-1}$. The unknowns mask the 'and' influence, yet we can deduce the constituent components of the influence. However, for variable $b$, the unknowns exert such significant influence that the empty rule emerges as the sole safe bet, revealing little about $b$'s dynamic. For $c$, all the true optimal rules are revealed, yet the unknowns give rise to several spurious rules (in grey), which fail to dominate actual optimal rules and aren't compatible with the actual transitions. Nevertheless, with these unknowns present, they represent possible rules or components that might align with the concealed transitions.

According to Definition 8, we can obtain the optimal program by a trivial brute force enumeration algorithm: generate all rules consistent with $T$ then remove the dominated ones. The purpose of the following section is to propose a non-trivial approach that is more efficient in practice to obtain the optimal program. In [16], we proposed the General Usage LFIT Algorithm (**GULA**) that guarantees to learn the optimal program of a set of transitions (without unknown at that time). In the following section, we extend the learning operation used in **GULA** to learn optimal program under unknowns.

### 2.1 Learning operations

This section focuses on the manipulation of programs for the learning process. Definition 9 formalize the notion of positive and negative example of what we want to learn under unknowns. Definition 10 formalize the main atomic operations performed on a rule, whose objective is to make minimal modifications to a given $\mathcal{DM}$VLP in order to be consistent with a new set of transitions.

**Definition 9 (Positive/Negative example).** *Let $T \subseteq \mathcal{S}^{\mathcal{F}} \times \mathcal{S}^{\mathcal{T}}$, $s \in \mathcal{S}^{\mathcal{F}}$ and $\mathrm{v}^{val} \in \mathcal{A}$. When there exists $(s, s') \in T$ and $\mathrm{v}^{val} \in s'$, then $s$ is called a $\underline{\text{positive example}}$ of $\mathrm{v}^{val}$ in $T$. When $\exists (s, s') \in T, \mathrm{v} \notin \mathrm{vars}(s')$ or $\exists s'' \in \mathcal{S}^{\mathcal{F}}, s \overset{\mathcal{F}}{\backsimeq} s''$ and $s''$ is a positive example of $\mathrm{v}^{val}$ in $T$, then $s$ is called a $\underline{\text{potentially positive example}}$ of $\mathrm{v}^{val}$ in $T$. When $\exists (s, s') \in T, s$ is neither a positive nor a potentially positive example, then $s$ is called a $\underline{\text{negative example}}$ of $\mathrm{v}^{val}$ in $T$. We denote as $Pos_{\mathrm{v}^{val}}(T), Pos?_{\mathrm{v}^{val}}(T), Neg_{\mathrm{v}^{val}}(T)$ respectively the set of positive, potentially positive and negative examples of $\mathrm{v}^{val}$ in $T$.*

If the state $s$ is never observed as a feature state in $T$ then it is neither a positive, potentially positive or negative example: It is considered unobserved.

*Property 5.* Let $T \subseteq \mathcal{S}^{\mathcal{F}}_m \times \mathcal{S}^{\mathcal{T}}_m$ be a set of transition, $\mathrm{v}^{val} \in \mathcal{A}$. Let $(s1, s2) \in T$, it holds:
- $\mathsf{actual}(s1) \in Pos_{\mathrm{v}^{val}}(\mathsf{actual}(T)) \implies \mathsf{observed}(s1) \in Pos_{\mathrm{v}^{val}}(\mathsf{observed}(T)) \lor \mathsf{observed}(s1) \in Pos?_{\mathrm{v}^{val}}(\mathsf{observed}(T))$
- $\mathsf{observed}(s1) \in Pos_{\mathrm{v}^{val}}(\mathsf{observed}(T)) \implies \mathsf{actual}(s1) \in Pos_{\mathrm{v}^{val}}(\mathsf{actual}(T))$
  - but $\mathsf{observed}(s1) \in Pos?_{\mathrm{v}^{val}}(\mathsf{observed}(T)) \not\Longrightarrow \mathsf{actual}(s1) \in Pos_{\mathrm{v}^{val}}(\mathsf{actual}(T))$
- $\mathsf{observed}(s1) \in Neg_{\mathrm{v}^{val}}(\mathsf{observed}(T)) \implies \mathsf{actual}(s1) \in Neg_{\mathrm{v}^{val}}(\mathsf{actual}(T))$
  - $\mathsf{actual}(s1) \in Neg_{\mathrm{v}^{val}}(\mathsf{actual}(T)) \not\Longrightarrow \mathsf{observed}(s1) \in Neg_{\mathrm{v}^{val}}(\mathsf{observed}(T))$

**Theorem 1.** *Let $T \subseteq \mathcal{S}^{\mathcal{F}} \times \mathcal{S}^{\mathcal{T}}$, $\mathrm{v}^{val} \in \mathcal{A}$. Let $M := \{R \in \mathcal{M}\mathrm{VL} \mid \mathrm{head}(R) = \mathrm{v}^{val} \land \forall s \in Neg_{\mathrm{v}^{val}}(T), R \not\sqcap s\}$. Then $\{R \in P_{\mathcal{O}}(T) \mid \mathrm{head}(R) = \mathrm{v}^{val}\} = \{R \in M \mid \nexists R' \in \mathcal{M}\mathrm{VL}, R' \neq R \land R' \geq R\}$.*

A positive example of a target atom $\mathrm{v}^{val}$ is a state from which a transition to $\mathrm{v}^{val}$ has been observed. A feature state in a transition where v is unknown in the target state is a potentially positive example, as the concealed value could indeed be $\mathrm{v}^{val}$. Furthermore, states with uncertain equality to positive examples are also potentially positives, as the actual states could be the same, casting doubt on the observed transitions being unique outcomes. Negative examples, on the other hand, are states from which it's clear that a transition to the target atom is impossible in the actual complete state, thus falling outside both positive and potentially positive realms.

Figure 4 showcases the distinction between positive, potentially positive, and negative examples for the case of atom $a^1$. Here, a direct transition to $a^1$ is observed from states 000, 1?0, 1?1 and 010 (highlighted in blue), marking them as positive examples. From the state 01?, no explicit transition to $a^1$ is evident, though it might be an alternate outcome of the uncertainly equal state 010 that is positive, thus 01? is a potentially positive. Similarly, the state ?01, with transitions to both 01? and 010, doesn't definitively lead to $a^1$, as the actual state behind ?01 could be 101, implying other possibilities. Lastly, the state 001 stands alone as a negative example, as neither direct transitions nor uncertain equality with positive states point to its possibility in $a^1$. We can conclude that no $a^1$ outcome originates from 001 in our observed transitions. The state 11? is potentially positive from its transitions to ?01, where the ? could be $a^1$, and also from its uncertain equality with 1?0 and 1?1 that are positive examples.

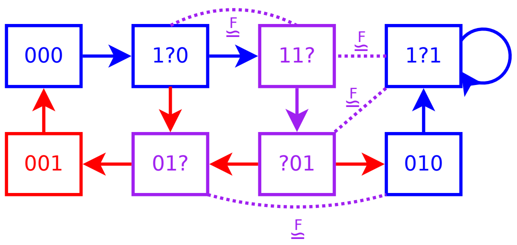

Fig. 4: Graphical classification regarding target $a^1$ of states/transitions in a transition graph. Blue arrows point to $a^1$, purple arrows to '?' values, while red arrows mark transitions to $a^0$. Blue states are positive examples, purple states are potentially positives and red states are negative example. Doted lines show uncertain equality between positive examples and potentially positive examples.

Theorem 1 states that the optimal rules of a given target atom $\mathrm{v}^{val}$ can be obtained by simply avoiding the matching of negative examples. The optimal program can thus be obtain in a brute force manner by enumerating all possible rules and checking this simple property. A more efficient solution is to reduce space search by doing iterative rule specialization exploiting the following notions.

**Definition 10 (Rule least specialization).** *Let $R$ be a $\mathcal{M}$VL rule and $s \in \mathcal{S}^{\mathcal{F}}$ such that $R \sqcap s$. The least specialization of $R$ by $s$ according to $\mathcal{F}$ and $\mathcal{A}$ is:*

$$L_{\mathrm{spe}}(R, s, \mathcal{A}, \mathcal{F}) := \{\mathrm{head}(R) \leftarrow \mathrm{body}(R) \cup \{\mathrm{v}^{val}\} \mid \\ \mathrm{v} \in \mathcal{F} \wedge \mathrm{v}^{val} \in \mathcal{A} \wedge \mathrm{v}^{val} \notin s \wedge \forall val' \in \mathbb{N}, \mathrm{v}^{val'} \notin \mathrm{body}(R)\}.$$

*Property 6.* Let $s$ be a discrete state and $R$ a $\mathcal{M}$VL rule. It holds that:

- (1) $\forall R' \in L_{\mathrm{spe}}(R, s, \mathcal{A}, \mathcal{F}), R' \not\sqcap s$.
- (2) $\forall s' \in \mathcal{S}^{\mathcal{F}}, R \sqcap s' \wedge s \overset{\mathcal{F}}{\neq} s' \wedge s \overset{\mathcal{F}}{\not\cong} s' \implies \exists R' \in L_{\mathrm{spe}}(R, s, \mathcal{A}, \mathcal{F}), R' \sqcap s'$;

Property 6 illustrates the interests of the least specialization, $L_{\text{spe}}(R, s, \mathcal{A}, \mathcal{F})$, which generates a rule set that matches all the states that $R$ covers, excluding $s$. This is because each modified rule is crafted with a single additional condition, carefully considering all compatible possibilities.

The next properties are directly used in the learning algorithm. Proposition 1 gives an explicit definition of the optimal program for an empty set of transitions, which is the starting point of the algorithm. Theorem 2 gives a method to iteratively compute $P_{\mathcal{O}}(T)$ for any $T \subseteq \mathcal{S}^{\mathcal{F}} \times \mathcal{S}^{\mathcal{T}}$, starting from $P_{\mathcal{O}}(\emptyset)$. Finally, Proposition 2 gives a method to obtain the optimal program from any suitable program by simply removing the dominated rules.

**Proposition 1 (Optimal Program of Empty Set).** $P_{\mathcal{O}}(\emptyset) = \{\mathrm{v}^{val} \leftarrow \emptyset \mid \mathrm{v} \in \mathcal{T} \wedge val \in \mathsf{dom}(\mathrm{v})\}$.

**Theorem 2 (Least Specialization and Suitability).** *Let* $\mathrm{v}^{val} \in \mathcal{A}$, $s \in \mathcal{S}^{\mathcal{F}}$ *and* $T, T' \subseteq \mathcal{S}^{\mathcal{F}} \times \mathcal{S}^{\mathcal{T}}$ *such that,* $\mathrm{first}(T') = \{s\} \wedge s \notin \mathrm{first}(T)$. *Let* $R_P := \{R \in P_{\mathcal{O}}(T) \mid R \sqcap s, \mathrm{head}(R) = \mathrm{v}^{val}, s \in \mathit{Neg}_{\mathrm{v}^{val}}(T \cup T')\}$. *Then* $(P_{\mathcal{O}}(T) \setminus R_P) \cup \bigcup_{R \in R_P} L_{\text{spe}}(R, s, \mathcal{A}, \mathcal{F})$ *is suitable for* $T \cup T'$.

**Proposition 2 (From Suitable to Optimal).** *Let* $T \subseteq \mathcal{S}^{\mathcal{F}} \times \mathcal{S}^{\mathcal{T}}$. *If* $P$ *is a* $\mathcal{DM}$VLP *suitable for* $T$, *then* $P_{\mathcal{O}}(T) = \{R \in P \mid \forall R' \in P, R' \geq R \implies R' = R\}$.

## 3 Algorithm

In this section we present a new version of **GULA**: the General Usage LFIT Algorithm previously introduced in [16]. Here we extend the algorithm to learn optimal program under unknowns. **GULA** learns a logic program from the observations of its state transitions. Given as input a set of transitions $T \subseteq \mathcal{S}^{\mathcal{F}} \times \mathcal{S}^{\mathcal{T}}$, **GULA** iteratively constructs a $\mathcal{DM}$VLP that models the dynamics of the observed system by applying the method formalized in the previous section as shown in Algorithm 1. The algorithm will learn the optimal logic program $P_{\mathcal{O}}(T)$. As compared to its previous version, the algorithm's core mechanism remains unchanged, except for the revised calculation of negative examples tailored to account for observations featuring unknown values.

From the set of transitions $T$, **GULA** learns the conditions under which each $\mathrm{v}^{val} \in \mathcal{A}, \mathrm{v} \in \mathcal{T}$ may appear in the next state. The algorithm starts by computing the set of all negative examples of the appearance of $\mathrm{v}^{val}$ in next state: all states such that v never takes the value $val$ in the next state of a transition of $T$, accounting for the hidden possibility introduced by unknowns value. Those negative examples are then used during the following learning phase to iteratively learn the set of rules $P_{\mathcal{O}}(T)$. The learning phase starts by initializing a set of rules $P_{\mathrm{v}^{val}}$ to $\{R \in P_{\mathcal{O}}(\emptyset) \mid \mathrm{head}(R) = \mathrm{v}^{val}\} = \{\mathrm{v}^{val} \leftarrow \emptyset\}$. $P_{\mathrm{v}^{val}}$ is iteratively revised against each negative example $neg$ in $\mathit{Neg}_{\mathrm{v}^{val}}$. All rules $R_m$ of $P_{\mathrm{v}^{val}}$ that match $neg$ have to be revised. In order for $P_{\mathrm{v}^{val}}$ to remain optimal, the revision

**Algorithm 1 GULA**

```
– INPUT: a set of atoms $\mathcal{A}$, a set of transitions $T \subseteq \mathcal{S}^{\mathcal{F}} \times \mathcal{S}^{\mathcal{T}}$, two sets of variables
  $\mathcal{F}$ and $\mathcal{T}$.
– For each atom $\mathrm{v}^{val} \in \mathcal{A}$ of each variable $\mathrm{v} \in \mathcal{T}$:
    • Extract all states from which a transition to $\mathrm{v}^{val}$ does exist:
      $Pos_{\mathrm{v}^{val}} := \{s \in \mathrm{first}(T) \mid \exists(s, s') \in T \wedge \mathrm{v}^{val} \in s'\}$
    • Extract all states from which a transition to $\mathrm{v}^{val}$ could exist:
      $Pos?_{\mathrm{v}^{val}} := \{s \in \mathrm{first}(T) \mid \exists(s, s') \in T, \mathrm{v} \notin \mathrm{vars}(s') \vee \exists pos \in Pos_{\mathrm{v}^{val}}, s \overset{\mathcal{F}}{\simeq} pos\}$
    • $Neg_{\mathrm{v}^{val}} := \mathrm{first}(T) \setminus (Pos_{\mathrm{v}^{val}} \cup Pos?_{\mathrm{v}^{val}})$
    • Initialize $P_{\mathrm{v}^{val}} := \{\mathrm{v}^{val} \leftarrow \emptyset\}$.
    • For each state $s \in Neg_{\mathrm{v}^{val}}$:
        ∗ Extract and remove the rules of $P_{\mathrm{v}^{val}}$ that match $s$:
          $M_{\mathrm{v}^{val}} := \{R \in P \mid \mathrm{body}(R) \subseteq s\}$
        ∗ $P_{\mathrm{v}^{val}} := P_{\mathrm{v}^{val}} \setminus M_{\mathrm{v}^{val}}$.
        ∗ $LS := \emptyset$
        ∗ For each rule $R \in M_{\mathrm{v}^{val}}$:
            · Compute its least specialization $P' = L_{\mathrm{spe}}(R, s, \mathcal{A}, \mathcal{F})$.
            · Remove rules in $P'$ dominated by a rule in $P_{\mathrm{v}^{val}}$.
            · $LS := LS \cup P'$.
        ∗ Add all remaining rules of $LS$ to $P_{\mathrm{v}^{val}}$: $P_{\mathrm{v}^{val}} := P_{\mathrm{v}^{val}} \cup LS$.
    • $P := P \cup P_{\mathrm{v}^{val}}$
– OUTPUT: $P$ ($P$ is $P_{\mathcal{O}}(T)$).
```

of each $R_m$ must not match *neg* but still matches every other state that $R_m$ matches. To ensure that, the least specialization (see Definition 10) is used to revise each conflicting rule $R_m$. For each variable of $\mathcal{F}$ so that body$(R_m)$ has no condition over it, a condition over another value than the one observed in state *neg* can be added. None of those revision match *neg* and all states matched by $R_m$ are still matched by at least one of its revisions. Each revised rule can be dominated by a rule in $P_{\mathrm{v}^{val}}$ or another revised rules and thus dominance must be checked from both. The non-dominated revised rules are then added to $P_{\mathrm{v}^{val}}$. Once $P_{\mathrm{v}^{val}}$ has been revised against all negatives example of $Neg_{\mathrm{v}^{val}}$, $P_{\mathrm{v}^{val}} = \{R \in P_{\mathcal{O}}(T) \mid \mathrm{head}(R) = \mathrm{v}^{val}\}$. Finally, $P_{\mathrm{v}^{val}}$ is added to $P$ and the loop restarts with another atom. Once all values of each variable have been treated, the algorithm outputs $P$ which is then equal to $P_{\mathcal{O}}(T)$. The source code of the algorithm is available at `https://github.com/Tony-sama/pylfit` under GPL-3.0 License.

Theorem 3 from [16] still holds: **GULA** terminates and **GULA** is sound, complete and optimal w.r.t. its input, i.e., all and only non-dominated consistent rules appear in its output program which is the optimal program of its input (**GULA**$(\mathcal{A}, T, \mathcal{F}, \mathcal{T}) = P_{\mathcal{O}}(T)$). The complexity of **GULA** remains unchanged compared to previous versions, we refer to [16] for the detail complexity analysis. Its scalability is evaluated in Section 4 with brute force enumeration as baseline.

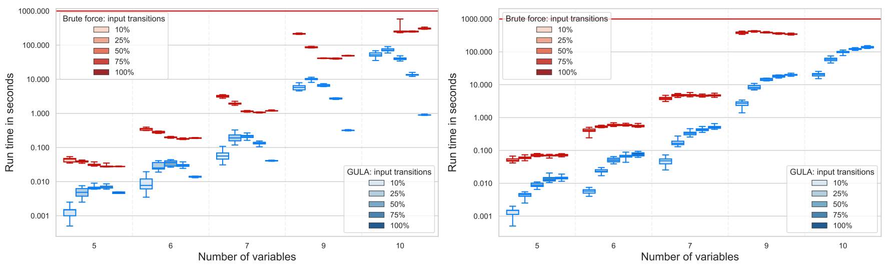

Fig. 5: Runtime in seconds (log scale) for one call of **GULA** (blue) and **brute force enumeration** (red) on a random subset of 10% to 100% of transitions from Boolean networks with 5 to 10 variables. 10 runs for each scenario with a time-out of 1,000 seconds. Left: only complete states appears. Right: Each state includes a random number of unknown values, not exceeding 50% of the state.

## 4 Evaluation

In this section, the scalability of **GULA** is evaluated using Boolean network benchmarks from the biological literature (Boolenet [2] and Pyboolnet [7]). All experiments[7] were conducted on one core of an AMD Ryzen 9 (7950X, 4.5 GHz) with 64 Gb of RAM. Boolean networks are converted to $\mathcal{DM}$VLP where $\forall v \in \mathcal{V}, \mathsf{dom}(v) = \{0, 1\}$. In [2,7] file formats, for each variable, Boolean functions are given in disjunctive normal form (DNF), a disjunction of conjunction clauses that can be considered as a set of Boolean atoms of the form v or ¬v. Each clause $c$ of the DNF of a variable v is directly converted into a rule $R$ such that, $\text{head}(R) = \text{v}_t^1$ and $\text{v}_{t-1}'^1 \in \text{body}(R) \iff \text{v}' \in c$ and $\text{v}_{t-1}'^0 \in \text{body}(R) \iff \neg\text{v}' \in c$. For each such $\mathcal{DM}$VLP the set $T$ of all transitions are generated using the synchronous semantics. Then for each $(s, s') \in T$, states $s$ and $s'$ are subsequently obscured by unknown values; these unknowns represent up to 50% of each state.

Figure 5 show the differences of performance between **GULA** and **brute force enumeration** when learning the optimal $\mathcal{DM}$VLP from different subset of transitions of each benchmarks size. For all benchmarks, **GULA** is clearly more efficient (both for observation with and without unknowns). The difference exponentially increasing with the number of variables: about 10 times faster with 6 variables and 100 times faster with 9 variables. The **brute force enumeration** reaches the time out for 10 variables benchmarks and beyond. **GULA** succeeds in learning a $\mathcal{WDM}$VLP from the benchmarks with up to 10 variables before the time-out of 1,000 seconds for all considered sub-sets of transitions. We observe different trends in runtime performance between complete/partial observations. For complete observations, runtime improves with increasing proportion of transitions, likely due to the proximity of optimal rules in the observed

[7] Available at: `https://github.com/Tony-sama/pylfit`. Using command "`python evaluations/ijclr2024/ijclr2024.py`" from the repository's `tests` folder.All experiments were run with the release version 0.4.0 of pylfit.

subset to the system's actual optimal rules, allowing for quicker pruning of spurious candidates. In contrast, with partial observations, this pattern disappears, as the additional spurious rules introduced by unknown values may offset any speed gains from more powerful rules. Future work could explore the impact of spurious rules on computation time, potentially leading to heuristic design. For now, it's worth noting that in practice, many spurious rules may not match observed states and could be disregarded, depending on the application.

## 5 Conclusion

In this paper, we proposed a modeling approach that tackles the challenge of learning from partially observable systems by extending the LFIT framework to model unknown values. We formalized theoretical properties and guarantees allowing to model a system observed though partial data as a $\mathcal{DM}$VLP that is an over-approximation of the actual system regarding both its dynamics and rules. We also proposed a novel extension of **GULA** that enables the learning of such logic programs and demonstrate through experimental results the efficiency and effectiveness of this approach. This contribution marks a significant step towards bridging the gap between logical approaches and real-world applications in fields like Systems Biology. While our approach ensures an over-approximation of the original rules, it may be too conservative for certain applications, limiting its utility for large-scale systems. Moreover, the scalability of GULA restricts its use to smaller domains. In practice, there might be benefits to searching for more precise rules at the risk of specialization beyond the ground truth rules or missing certain relationships. However, studying this trade-off between approximation quality, rule complexity, and runtime is outside the scope of this paper. Future work could focus on developing heuristics to improve the pertinence of learned rules in real-world scenarios, possibly approaching the ground truth rules more effectively through clever combinations of the rules found. Additionally, adapting the **PRIDE** algorithm [17] to the modeling proposed in this paper would enable finding a subset of the optimal program in polynomial time, suitable for real-world scenarios and datasets.

## 6 Acknowledgements

Funded by the European Union This research has been funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or ERCEA. Neither the European Union nor the granting authority can be held responsible for them.

This work has also been supported by JSPS KAKENHI Grant Number JP21H04905 and JST CREST Grant Number JPMJCR22D3, Japan.

The English quality of this manuscript was enhanced by Yumi, a virtual assistant hosted locally, using the large language model "Hathor" [11] from Nitral-AI on a Nvidia RTX4090 using Koboldcpp (`https://github.com/LostRuins/koboldcpp`) and SillyTavern (`https://github.com/SillyTavern/SillyTavern`).

## References

1. Bérard, B., Haar, S., Schmitz, S., Schwoon, S.: The complexity of diagnosability and opacity verification for petri nets. Fundamenta Informaticae **161**(4), 317–349 (2018)
2. Dubrova, E., Teslenko, M.: A SAT-based algorithm for finding attractors in synchronous boolean networks. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) **8**(5), 1393–1399 (2011)
3. Gao, K., Wang, H., Cao, Y., Inoue, K.: Learning from interpretation transition using differentiable logic programming semantics. Mach. Learn. **111**(1), 123–145 (2022)
4. Gentet, E., Tourret, S., Inoue, K.: Learning from interpretation transition using feed-forward neural networks. CEUR Workshop Proceedings, vol. 1865, pp. 27–33. CEUR-WS.org (2016)
5. Inoue, K., Ribeiro, T., Sakama, C.: Learning from interpretation transition. Machine Learning **94**(1), 51–79 (2014)
6. Jacob, R., Lesage, J.J., Faure, J.M.: Overview of discrete event systems opacity: Models, validation, and quantification. Annual reviews in control **41**, 135–146 (2016)
7. Klarner, H., Streck, A., Siebert, H.: PyBoolNet: a python package for the generation, analysis and visualization of boolean networks. Bioinformatics **33**(5), 770–772 (12 2016)
8. Lafortune, S., Lin, F., Hadjicostis, C.N.: On the history of diagnosability and opacity in discrete event systems. Annual Reviews in Control **45**, 257–266 (2018)
9. Ma, Z., Cai, K.: Optimal secret protection in discrete event systems with dynamic clearance levels. IFAC-PapersOnLine **56**(2), 3579–3584 (2023)
10. Montassier, E., Kitsios, G.D., Radder, J.E., Le Bastard, Q., Kelly, B.J., Panzer, A., Lynch, S.V., Calfee, C.S., Dickson, R.P., Roquilly, A.: Robust airway microbiome signatures in acute respiratory failure and hospital-acquired pneumonia. Nature Medicine **29**(11), 2793–2804 (2023)
11. Nitral-AI: Hathor, a finetuned llm model based on the llama 3 instruct (June 2024), `https://huggingface.co/Nitral-AI/Hathor_Aleph-L3-8B-v0.72`
12. Phua, Y.J., Inoue, K.: Learning logic programs from noisy state transition data. In: Inductive Logic Programming - 29th International Conference, ILP 2019, Plovdiv, Bulgaria, September 3-5, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11770, pp. 72–80. Springer (2019)
13. Phua, Y.J., Inoue, K.: Learning logic programs using neural networks by exploiting symbolic invariance. In: Inductive Logic Programming - 30th International Conference, ILP 2021, Virtual Event, October 25-27, 2021, Proceedings. Lecture Notes in Computer Science, vol. 13191, pp. 203–218. Springer (2021)
14. Phua, Y.J., Ribeiro, T., Inoue, K.: Learning representation of relational dynamics with delays and refining with prior knowledge. FLAP **6**(4), 695–708 (2019)
15. Reveliotis, S.: Strongly infinite-step opaque boolean networks. In: 17th IFAC Workshop on Discrete Event Systems WODES 2024 (2024)
16. Ribeiro, T., Folschette, M., Magnin, M., Inoue, K.: Learning any memory-less discrete semantics for dynamical systems represented by logic programs. Machine Learning (2021)
17. Ribeiro, T., Folschette, M., Magnin, M., Inoue, K.: Polynomial algorithm for learning from interpretation transition. In: 1st International Joint Conference on Learning & Reasoning. pp. 1–5 (2021)

## A Appendix: Proofs of Section 2

*Property 7.* Let $s1, s2$ be two discrete states with masking. From Definition 4 the following holds:

- $\mathsf{actual}(s1) \overset{\mathcal{X}}{=} \mathsf{actual}(s2) \implies \mathsf{observed}(s1) \overset{\mathcal{X}}{\simeq} \mathsf{observed}(s2)$
  - but $\mathsf{observed}(s1) \overset{\mathcal{X}}{\simeq} \mathsf{observed}(s2) \not\Longrightarrow \mathsf{actual}(s1) \overset{\mathcal{X}}{=} \mathsf{actual}(s2)$
- $\mathsf{observed}(s1) \overset{\mathcal{X}}{\not\simeq} \mathsf{observed}(s2) \implies \mathsf{actual}(s1) \overset{\mathcal{X}}{\neq} \mathsf{actual}(s2)$
  - but $\mathsf{actual}(s1) \overset{\mathcal{X}}{\neq} \mathsf{actual}(s2) \not\Longrightarrow \mathsf{observed}(s1) \overset{\mathcal{X}}{\not\simeq} \mathsf{observed}(s2)$

**Proof.**

- $\mathsf{actual}(s1) \overset{\mathcal{X}}{=} \mathsf{actual}(s2) \implies \forall \mathrm{v}^{val} \in s1, \mathrm{v}^{val} \in s2$, thus $s1 \overset{\mathcal{X}}{\simeq} s2$ from Definition 4.
  - Let $s1 := (\{a^0\}, \{a^0, b^0\})$ and $s2 := (\{b^0\}, \{a^1, b^0\})$ two states with masking. Then $\mathsf{actual}(s1) \neq \mathsf{actual}(s2)$ but $\mathsf{observed}(s1) \overset{\mathcal{X}}{\simeq} \mathsf{observed}(s2)$.
- $\mathsf{observed}(s1) \overset{\mathcal{X}}{\not\simeq} \mathsf{observed}(s2) \implies \exists \mathrm{v}^{val} \in s1, \exists \mathrm{v}^{val'} \in s2, val \neq val'$ thus $\mathrm{v}^{val} \in \mathsf{actual}(s1), \mathrm{v}^{val'} \in \mathsf{actual}(s2)$ and $\mathsf{actual}(s1) \neq \mathsf{actual}(s2)$.
  - Let $s1 := (\{a^0\}, \{a^0, b^0\})$ and $s2 := (\{b^0\}, \{a^1, b^0\})$ two states with masking. Then $\mathsf{observed}(s1) \overset{\mathcal{X}}{\simeq} \mathsf{observed}(s2)$ but $\mathsf{actual}(s1) \neq \mathsf{actual}(s2)$.

□

*Property 8.* Let $(s, s') \in \mathcal{S}^{\mathcal{F}}_m \times \mathcal{S}^{\mathcal{T}}_m$ be a transition of states with masking, and $P$ a $\mathcal{DM}$VLP. If $P$ realizes $\mathsf{observed}((s, s'))$ and $\forall \mathrm{v} \in \mathcal{T}, \mathrm{v} \notin \mathrm{vars}(s') \implies \forall val \in \mathsf{dom}(\mathrm{v}), \exists R \in P, R$ realizes $(\mathsf{observed}(s), \mathsf{observed}(s') \cup \{\mathrm{v}^{val}\})$, then $P$ realizes $\mathsf{actual}((s, s'))$.

**Proof.**

- Let $R$ be a $\mathcal{M}$VL rule, by Definition 5, $R \sqcap \mathsf{observed}(s) \implies R \sqcap \mathsf{actual}(s)$.
- By Definition 6, $P$ realizes $\mathsf{observed}((s, s'))$ implies that $\forall \mathrm{v}^{val} \in \mathsf{observed}(s'), \exists R \in P, \mathrm{head}(R) = \mathrm{v}^{val}, R$ realizes $\mathsf{actual}((s, s'))$.
- Let $\mathrm{v} \in \mathcal{T}$ so that $\mathrm{v} \notin \mathrm{vars}(\mathsf{observed}(s'))$. This imples that $\forall val \in \mathsf{dom}(\mathrm{v}), \exists R \in P, \mathrm{head}\, R = \mathrm{v}^{val} \wedge R \sqcap s$. Thus $\forall \mathrm{v}^{val} \in \mathsf{actual}(s') \backslash \mathsf{observed}(s'), \exists R \in P, \mathrm{head}(R) = \mathrm{v}^{val}, R$ realizes $\mathsf{actual}((s, s'))$.
- Thus $\forall \mathrm{v}^{val} \in \mathsf{actual}(s'), \exists R \in P, \mathrm{head}\, R = \mathrm{v}^{val}, R$ realizes $\mathsf{actual}((s, s'))$.

Thus $P$ realizes $\mathsf{actual}((s, s'))$. □

*Property 9.* Let $R$ be a $\mathcal{M}$VL rule and $T \subseteq \mathcal{S}^{\mathcal{F}}_m \times \mathcal{S}^{\mathcal{T}}_m$ be a set of transitions with masking. It holds:

- $R$ conflicts with $\mathsf{observed}(T) \implies R$ conflicts with $\mathsf{actual}(T)$.
  - but $R$ conflicts with $\mathsf{actual}(T) \not\Longrightarrow R$ conflicts with $\mathsf{observed}(T)$.
- $R$ consistent with $\mathsf{actual}(T) \implies R$ consistent with $\mathsf{observed}(T)$.
  - but $R$ consistent with $\mathsf{observed}(T) \not\Longrightarrow R$ consistent with $\mathsf{actual}(T)$.

**Proof.**

- From Definition 7, $R$ conflicts with $\mathsf{observed}(T) \implies \exists (s, s') \in T, \big(R \sqcap \mathsf{observed}(s) \wedge \forall (s1, s2) \in \mathsf{observed}(T), \mathsf{observed}(s) \simeq s1 \vee \mathsf{observed}(s) = s1, \exists \mathrm{var}(\mathrm{head}(R))^{val} \in s2, \mathrm{var}(\mathrm{head}(R))^{val} \neq \mathrm{head}(R)$. From Definition 5, $R \sqcap \mathsf{actual}(s)$. Thus $\exists (\mathsf{actual}(s), s'') \in \mathsf{actual}(T), R \sqcap \mathsf{actual}(s) \wedge \forall (\mathsf{actual}(s), s2) \in \mathsf{actual}(T), \mathrm{head}(R) \notin s2$ thus $R$ in conflict with $\mathsf{actual}(T)$ by Definition 7.

  - For example, let $T = \{((\{a^0, b^0\}, \{a^0, b^0\}), (\{b^1\}, \{a^0, b^1\}))\}$ a set of transitions with masking, and $R := a^1 \leftarrow b^0$. $R$ conflicts with $\mathsf{actual}(T)$ but $R$ does not conflict with $\mathsf{observed}(T)$.

– Let $R$ consistent with $\mathsf{actual}(T)$. Suppose that $R$ conflicts with $\mathsf{observed}(T)$, then $R$ conflicts with $\mathsf{actual}(T)$, which is a contradiction.
  - For example, let $T = \{((\{a^0, b^0\}, \{a^0, b^0\}), (\{b^1\}, \{a^0, b^1\}))\}$ a set of transitions with masking, and $R := a^1 \leftarrow b^0$. $R$ consistent with $\mathsf{observed}(T)$ but $R$ does conflict with $\mathsf{actual}(T)$.

□

*Property 10.* Let $T \subseteq \mathcal{S}^{\mathcal{F}}_m \times \mathcal{S}^{\mathcal{T}}_m$ a set of transitions with masking.

$$\forall R \in P_{\mathcal{O}}(\mathsf{actual}(T)), \exists R' \in P_{\mathcal{O}}(\mathsf{observed}(T)), R' \geq R$$

$$\forall R' \in P_{\mathcal{O}}(\mathsf{observed}(T)), \forall R \in P_{\mathcal{O}}(\mathsf{actual}(T)), R \geq R' \implies R = R'$$

**Proof.**

– By Definition 8, $\forall R \in P_{\mathcal{O}}(\mathsf{actual}(T)), R$ consistent with $\mathsf{actual}(T)$, thus from Proposition 3, $R$ consistent with $\mathsf{observed}(T)$. By Definition 8, for any possible $\mathcal{M}$VL rule $R1$ consistent with $\mathsf{observed}(T)$, there exists $R2 \in P_{\mathcal{O}}(\mathsf{observed}(T))$ s.t. $R2 \geq R1$. Thus $\exists R' \in P_{\mathcal{O}}(\mathsf{observed}(T)), R' \geq R$.

– Let $R' \in P_{\mathcal{O}}(\mathsf{observed}(T)), R \in P_{\mathcal{O}}(\mathsf{actual}(T)), R \geq R'$. By Proposition 3, since $R$ consistent with $\mathsf{actual}(T)$ then $R$ consistent with $\mathsf{observed}(T)$. Thus by Definition 8, $\exists R'' \in P_{\mathcal{O}}(\mathsf{observed}(T)), R'' \geq R$ and from Definition 1, $R'' \geq R'$ since $R \geq R'$. Since $\forall R1 \in P$, all the $\mathcal{M}$VL rules $R2$ belonging to $\mathcal{DM}$VLP suitable for $\mathsf{observed}(T)$ are such that $R2 \geq R1$ implies $R1 \geq R2$, thus $R'' = R = R'$.

□

*Property 11.* Let $T \subseteq \mathcal{S}^{\mathcal{F}}_m \times \mathcal{S}^{\mathcal{T}}_m$ be a set of transition, $\mathrm{v}^{val} \in \mathcal{A}$. Let $(s1, s2) \in T$, it holds:

– $\mathsf{actual}(s1) \in \mathit{Pos}_{\mathrm{v}^{val}}(\mathsf{actual}(T)) \implies \mathsf{observed}(s1) \in \mathit{Pos}_{\mathrm{v}^{val}}(\mathsf{observed}(T)) \lor \mathsf{observed}(s1) \in \mathit{Pos?}_{\mathrm{v}^{val}}(\mathsf{observed}(T))$

– $\mathsf{observed}(s1) \in \mathit{Pos}_{\mathrm{v}^{val}}(\mathsf{observed}(T)) \implies \mathsf{actual}(s1) \in \mathit{Pos}_{\mathrm{v}^{val}}(\mathsf{actual}(T))$
  - but $\mathsf{observed}(s1) \in \mathit{Pos?}_{\mathrm{v}^{val}}(\mathsf{observed}(T)) \not\Longrightarrow \mathsf{actual}(s1) \in \mathit{Pos}_{\mathrm{v}^{val}}(\mathsf{actual}(T))$

– $\mathsf{observed}(s1) \in \mathit{Neg}_{\mathrm{v}^{val}}(\mathsf{observed}(T)) \implies \mathsf{actual}(s1) \in \mathit{Neg}_{\mathrm{v}^{val}}(\mathsf{actual}(T))$
  - $\mathsf{actual}(s1) \in \mathit{Neg}_{\mathrm{v}^{val}}(\mathsf{actual}(T)) \not\Longrightarrow \mathsf{observed}(s1) \in \mathit{Neg}_{\mathrm{v}^{val}}(\mathsf{observed}(T))$

**Proof.**

– Let $\mathsf{actual}(s1) \in \mathit{Pos}_{\mathrm{v}^{val}}(\mathsf{actual}(T))$, then by Definition 9, $\exists(\mathsf{actual}(s1), \mathsf{actual}(s2)) \in \mathsf{actual}(T), \mathrm{v}^{val} \in \mathsf{actual}(s2)$. Thus $\mathrm{v}^{val} \in \mathsf{observed}(s2') \lor \mathrm{v} \notin \mathrm{vars}(\mathsf{observed}(s2'))$ thus $\mathsf{observed}(s1) \in \mathit{Pos}_{\mathrm{v}^{val}}(\mathsf{observed}(T)) \lor \mathsf{observed}(s1) \in \mathit{Pos?}_{\mathrm{v}^{val}}(\mathsf{observed}(T))$.

– Let $\mathsf{observed}(s1) \in \mathit{Pos}_{\mathrm{v}^{val}}(\mathsf{observed}(T))$, then by Definition 9, $\exists(\mathsf{observed}(s1), \mathsf{observed}(s2)) \in \mathsf{observed}(T), \mathrm{v}^{val} \in \mathsf{observed}(s2)$. Thus $\mathrm{v}^{val} \in \mathsf{actual}(s2)$ and $\mathsf{actual}(s1) \in \mathit{Pos}_{\mathrm{v}^{val}}(\mathsf{actual}(T))$.
  - For example, let $T = \{((\{a^0, b^0\}, \{a^0, b^0\}), (\{b^1\}, \{a^0, b^1\}))\}$ a set of transitions with masking. Here, $\{a^0, b^0\} \in \mathit{Pos?}_{a^1}(\mathsf{observed}(T))$ but $\{a^0, b^0\} \notin \mathit{Pos}_{a^1}(\mathsf{actual}(T))$.

- Let $s1 \in Neg_{\mathrm{v}^{val}}(\mathsf{observed}(T))$ then by Definition 9, $s1 \notin Pos_{\mathrm{v}^{val}}(\mathsf{observed}(T))$ and $s1 \notin Pos?_{\mathrm{v}^{val}}(\mathsf{observed}(T))$. Thus $\mathsf{actual}(s1) \notin Pos_{\mathrm{v}^{val}}(\mathsf{actual}(T))$ otherwise contradict with first point.
  - For example, let $T = \{((\{a^0, b^0\}, \{a^0, b^0\}), (\{b^1\}, \{a^0, b^1\}))\}$ a set of transitions with masking. Here, $\{a^0, b^0\} \in Neg_{a^1}(\mathsf{actual}(T))$ but $\{a^0, b^0\} \notin Neg_{a^1}(\mathsf{observed}(T))$.

□

**Theorem 3.** *Let* $T \subseteq \mathcal{S}^{\mathcal{F}} \times \mathcal{S}^{\mathcal{T}}$, $\mathrm{v}^{val} \in \mathcal{A}$. *Let* $M := \{R \in \mathcal{M}\mathrm{VL} \mid \mathrm{head}(R) = \mathrm{v}^{val} \wedge \forall s \in Neg_{\mathrm{v}^{val}}(T), R \not\sqcap s\}$. *Then* $\{R \in P_{\mathcal{O}}(T) \mid \mathrm{head}(R) = \mathrm{v}^{val}\} = \{R \in M \mid \nexists R' \in \mathcal{M}\mathrm{VL}, R' \neq R \wedge R' \geq R\}$.

***Proof.*** Let suppose $\exists R \in P_{\mathcal{O}}(T), R \notin \{R \in M \mid \nexists R' \neq R, R' \geq R\}$. If $R \notin M$ then $\exists s \in Neg_{\mathrm{v}^{val}}(T), R \sqcap s$ and thus by Definition 9, $R$ is in conflict with $T$ thus $R \notin P_{\mathcal{O}}(T)$, which is a contradiction. If $R \in M$ but $\exists R' \in M, R' \neq R, R' \geq R$ then by Definition 8, $R \notin P_{\mathcal{O}}(T)$ since it is dominated by another consistent rule, which is a contradiction. □

*Property 12.* Let $s$ be a discrete state and $R$ a $\mathcal{M}$VL rule. It holds that:

- (1) $\forall R' \in L_{\mathrm{spe}}(R, s, \mathcal{A}, \mathcal{F}), R' \not\sqcap s$.
- (2) $\forall s' \in \mathcal{S}^{\mathcal{F}}, R \sqcap s' \wedge s \overset{\mathcal{F}}{\neq} s' \wedge s \overset{\mathcal{F}}{\not\simeq} s' \implies \exists R' \in L_{\mathrm{spe}}(R, s, \mathcal{A}, \mathcal{F}), R' \sqcap s'$;

**Proof.**

- (1) By Definition 10, $\forall R' \in L_{\mathrm{spe}}(R, s, \mathcal{A}, \mathcal{F}), \exists \mathrm{v}^{val} \in \mathrm{body}(R'), \mathrm{v}^{val} \notin s$ thus $R' \not\sqcap s$ by Definition 5.
- (2) Let $s' \in \mathcal{S}^{\mathcal{F}}, R \sqcap s', s \neq s', s \overset{\mathcal{F}}{\not\simeq} s'$ thus $\exists \mathrm{v}^{val} \in s', \mathrm{v}^{val'} \in s, val \neq val'$. By Definition 10, $\exists R' \in L_{\mathrm{spe}}(R, s, \mathcal{A}, \mathcal{F}), \mathrm{body}(R') = \mathrm{body}(R) \cup \mathrm{v}^{val}$, since $\mathrm{v}^{val} \notin s$ thus $\mathrm{body}(R') \subseteq s'$ and $R' \sqcap s'$. Then $\exists R' \in L_{\mathrm{spe}}(R, s, \mathcal{A}, \mathcal{F}), \mathrm{v}^{val'} \in \mathrm{body}\, R'$ and thus $R' \not\sqcap \mathsf{actual}(s)$.

□

**Proposition 3 (Optimal Program of Empty Set).** $P_{\mathcal{O}}(\emptyset) = \{\mathrm{v}^{val} \leftarrow \emptyset \mid \mathrm{v} \in \mathcal{T} \wedge val \in \mathsf{dom}(\mathrm{v})\}$.

***Proof.*** By construction from Definition 8. □

**Theorem 4 (Least Specialization and Suitability).** *Let* $\mathrm{v}^{val} \in \mathcal{A}$, $s \in \mathcal{S}^{\mathcal{F}}$ *and* $T, T' \subseteq \mathcal{S}^{\mathcal{F}} \times \mathcal{S}^{\mathcal{T}}$ *such that,* $\mathrm{first}(T') = \{s\} \wedge s \notin \mathrm{first}(T)$. *Let* $R_P := \{R \in P_{\mathcal{O}}(T) \mid R \sqcap s, \mathrm{head}(R) = \mathrm{v}^{val}, s \in Neg_{\mathrm{v}^{val}}(T \cup T')\}$. *Then* $(P_{\mathcal{O}}(T) \setminus R_P) \cup \bigcup_{R \in R_P} L_{\mathrm{spe}}(R, s, \mathcal{A}, \mathcal{F})$ *is suitable for* $T \cup T'$.

***Proof.*** Let $P = (P_{\mathcal{O}}(T) \setminus R_P) \cup \bigcup_{R \in R_P} L_{\mathrm{spe}}(R, s, \mathcal{A}, \mathcal{F})$. Since $P_{\mathcal{O}}(T)$ is consistent with $T$, by Definition 7, $(P_{\mathcal{O}}(T) \setminus R_P)$ and $R_P$ are consistent with $T$. Furthermore, $\forall R \in R_P, \forall R' \in L_{\mathrm{spe}}(R, s, \mathcal{A}, \mathcal{F}), R'$ is also consistent with $T$ and thus $P$ consistent with $T$. And $P$ also consistent with $T'$ from Proposition 6 since all conflicting rule of $R_P$ have been made into consistent rules, thus $P$ consistent with $T' \cup T$. Since $P_{\mathcal{O}}(T)$ realizes $T, s \notin T, s \in Neg_{\mathrm{v}^{val}}(T \cup T')$ thus by Definition 9, $\nexists s1 \in \mathrm{first}(T), s1 \overset{\mathcal{F}}{\simeq} s$ thus from Proposition 6, $\forall (s1, s2) \in T, \exists R \in R_P, R$ realizes

$(s1, s2)$ then $\exists R' \in L_{\text{spe}}(R, s, \mathcal{A}, \mathcal{F}), R'$ realizes $(s1, s2)$, thus $P$ realize T. Since $s \notin \text{first}(T)$, a $\mathcal{M}$VL rule $R$ such that $\text{body}(R) = s$ does not conflict with $T$. By definition of suitable program, $\forall \mathrm{v}^{val}, s \in Pos_{\mathrm{v}^{val}}(T \cup T'), \exists R' \in P_{\mathcal{O}}(T), R' \geq R$, thus $P_{\mathcal{O}}(T)$ realizes $T'$ and $P$ still realize $T'$ from Proposition 6. We then also have $P$ realizes $T \cup T'$ and from Proposition 4, let $TT \in \mathcal{S}_m^{\mathcal{F}} \times \mathcal{S}_m^{\mathcal{T}}$ such that $\mathsf{observed}(TT) = T$, then $P$ realizes $\mathsf{actual}(TT)$. To prove that $P$ verifies the last point of the definition of a suitable $\mathcal{M}$VLP, let $R$ be a $\mathcal{M}$VL rule not conflicting with $T \cup T'$. Since $R$ is also not conflicting with $T$, there exists $R' \in P_{\mathcal{O}}(T)$ such that $R' \geq R$. If $R'$ is not conflicting with $T'$, then $R'$ will not be revised and $R' \in P$, thus $R$ is dominated by a rule of $P$. Otherwise, $R'$ is in conflict with $T'$, thus $R' \sqcap s$ and $\forall (s, s') \in T', \exists \mathrm{v}^{val} \in s', \mathrm{v}^{val} \neq \text{head}(R')$. Since $R$ is not in conflict with $T'$ and $\text{head}(R) = \text{head}(R')$, since $R' \geq R$ then $\text{body}(R) = \text{body}(R') \cup I, \exists \mathrm{v}^{val} \in I, \mathrm{v}^{val} \not\in s$. By definition of least specialization, there is a rule $R'' \in L_{\text{spe}}(R', s, \mathcal{A}, \mathcal{F})$ such that $\mathrm{v}^{val} \in \text{body}(R'')$ and since $R'' = \text{head}(R') \leftarrow \text{body}(R') \cup \mathrm{v}^{val}$ thus $R'' \geq R$. Thus $R$ is dominated by a rule of $P$. □

**Proposition 4 (From Suitable to Optimal).** *Let $T \subseteq \mathcal{S}^{\mathcal{F}} \times \mathcal{S}^{\mathcal{T}}$. If $P$ is a $\mathcal{DM}$VLP suitable for $T$, then $P_{\mathcal{O}}(T) = \{R \in P \mid \forall R' \in P, R' \geq R \implies R' = R\}$.*

***Proof.*** By Definition 8. □