Theory Synthesis based on Experience

Yannick Chevalier

Institut de Recherche en Informatique de Toulouse
CNRS - INP - UT3 - UT1 - UT2J

Theory Synthesis based on Experience

Yannick Chevalier*
IRIT, Toulouse University, CNRS, INP, UT3, Toulouse, France
Yannick.Chevalier@irit.fr

* contact author

October 3rd, 2022

Technical report No. IRIT/RR–2022–08–FR
(version 1)

Institut de Recherche en Informatique de Toulouse
CNRS - INP - UT3 - UT1 - UT2J

Theory Synthesis based on Experience

Yannick Chevalier*
IRIT, Toulouse University, CNRS, INP, UT3, Toulouse, France
Yannick.Chevalier@irit.fr

* contact author

October 3rd, 2022

Abstract. We present in this report a novel approach to learning that results in a first-order logic model. The construction proceeds in two steps.

In the first step we interpret first-order function symbols as strict Scott-continuous functions over algebraic domains, a category that encompasses among others sets of traces and sets of sets. First-order terms are composition of functions, and a "flight envelope" of the system is constructed through the construction of new terms and the recording of the maximal values encountered for the terms during the analysis of a set of traces. We prove that this approach has an Angluin-like property for learning `wrt` false positives and negatives. This approach is also practical and was applied in previous works to construct efficient intrusion detection systems.

In the second step we consider an ordered set of cognitive states. Each cognitive state is a representation model equiped with a set of ground object terms denoting the facts on which reasoning is based and a set of representation terms denoting the tests performed on these object terms. Under an additional condition of stability we construct for each cognitive state a set of predicates over a constructed domain such that elements of the domains are the equivalence classes of the object terms relative to the predicates while predicates are defined by sets of object terms. We prove that the ordered set of models derived from the cognitive states forms an inverse system of first-order logic model.

This construction fills the gap in previous work by Achourioti and van Lanbalgem were inverse systems of first-order logic models are employed to formalize Kant's approach to reasoning in the Critique of the Pure Reason.

Keywords: Algebraic domains ; Geometric Logic ; Knowledge Representation; Learning

Technical report No. IRIT/RR–2022–08–FR
(version 1)

Institut de Recherche en Informatique de Toulouse
CNRS - INP - UT3 - UT1 - UT2J

Synthèse de Théories basée sur l'Expérience

Yannick Chevalier*
IRIT, Université de Toulouse, CNRS, INP, UT3, Toulouse, France
Yannick.Chevalier@irit.fr

* contact author

3 octobre 2022

Résumé. Ce document constitue la documentation du style LaTeX IRIT à utiliser pour les rapports techniques ou les rapports de recherche. Il explique notamment les différentes options de la classe de document et leur impact sur son rendu final.

Il requiert un numéro qui vous sera fourni par le service de documentation de l'IRIT (écrire à serdoc@irit.fr).

Mots-clés : documentation, LaTeX, IRIT.

Rapport technique N° IRIT/RR–2022–08–FR
(version 1)

1 Introduction

Context. This report is the first step of an attempt to formalize algorithmic approaches to machine learning, and to describe in what sense learning leads to the construction of *logic theories* to describe the *outside world*. We postulate that the world is only accessible through, and thus can be modeled as, a sequential observation of events or the sharing of such sequences. For the sake of simplicity we thus assume that the outside world is a (possibly infinite) set of (possibly infinite) sequences of events (each of a possibly infinite kind).

Let us first circumscribe the problem and note that two unrelated traditions have arrived to a similar conclusion. Computer scientists have recognised the critical role of geometric logic to construct logical representations of programs based on their semantics [18] (see also [13, 21] for a gentler introduction to the topic). In the philosophical approach, [22] presented a setting in which the definition of a logic from experience is achieved based on the novel analysis of Kant's Critique of the Pure Reason [10] in [3]. The evolution of knowledge through cognitions is captured as an inverse system of models. The formulas whose truth is preserved through inverse morphisms are called *objectively valid*, and those that are true in the inverse limit of the inverse system of models are the *transcendental formulas*. [22] proved that geometric logic formulas are objectively valid. This presentation of learning through cognitions is extremely attractive but the inverse system of models relied upon is taken as granted. We present in this report the construction one such inverse system of models.

Organisation.. We present in Section 2 our model of ML algorithms and introduce the $\mathbf{ALG}_{\perp!}$ category. We introduce sets of traces of events, basic definitions and prove that the basic transformations are continuous in Section 3. and prove that the construction of new objects and representations, as well as the addition of new algorithms to the signature is continuous. We introduce in Section 4 a first-order functional signature to represent these functions, and prove that the addition of new terms to better represent the world observed is continuous. We prove in Section 5 that the valuation of a set of representations is continuous, and that this continuity leads to an Angluin-like property to guide changes when there are false positives (the monitor is over-constrained) and false negatives (the monitor is too lax). We present in Section 6 the relation between this construction and the more general notion of knowledge that was built in the recent decades from a more modern reading of Kant's Critique, resulting in notions of knowledge based on inverse systems of models, and we present how the initial construction is extended in the rest of this report with these philosophical considerations. *Cognitive states* are introduced in Section 7 and we construct for each *stable* cognitive state a first-order logic model representing the predicates, domain elements, and their interpretation in that state. Finally we prove in Section 8 that when considering a family of ordered stable cognitive states, the first-order logic models of these cognitive states forms an inverse system of models.

2 Modeling Learning

2.1 Informal considerations

Per the Oxford dictionary, *learning* is the acquisition of knowledge or skills through *study*, *experience*, or *being taught*.

Without going into the definitions of knowledge and skills it can be infered

that these things increase through the three stated activities. Let d, d' be datasets representing observations of the world, and assume that $d \subseteq d'$. Then a learning algorithm f applied on d' should return a better result than when applied on d, something we denote with $f(d) \sqsubseteq f(d')$. If the possible inputs are partially ordered–*e.g.* with set inclusion–and D_f is the set of the values that can be returned by f we infer that D_f must be partially ordered, and that f must be monotonic *wrt* to the input data. Reasoning along the same lines, we conclude that the result of learning must be monotonic `wrt` *cognitions*–inferences that are drawn from the currently computed value–and *teachings*–user-given information. Also, assuming the value returned denotes knowledge and that the function is unbiased, the value $f(\emptyset)$ should be the least element of D_f.

Monotonicity and strictness are not sufficient as *e.g.* a function returning the number of events in an observation is monotonic and strict with the usual ordering on $\mathbb{N}$. A proper learning function shall converge towards a value representing the best possible knowledge of the system given any dataset. Assume that the observations are bounded by a set d of infinite possible observations. Then for all sets of worlds $(d_n)_{n\in\mathbb{N}}$ that converges towards d, *i.e.* such that $\cup_{n\in\mathbb{N}} d_n = d$, we shall also have $\bigsqcup_{n\in\mathbb{N}} f(d_n) = f(d)$, even if the latter can only be asserted to exist and not computed explicitely. Since f maps a partially ordered set (poset) to another poset, this condition means that f is *Scott-continuous*, and the assumption that d is the upper bound of a set of finite (compact) observations indicates that the domain of f must be algebraic.

2.2 Model for machine learning

Domains that have certain properties are the *objects* of a category and we are interested in the functions between these objects, the morphisms.

Domains. A *poset* is a partially ordered set $(P, \sqsubseteq)$. It is *pointed* if it has a minimal element. A subset $X \subseteq P$ is *directed* whenever two elements $x, y \in X$ have a *least upper bound*, denoted $x \sqcup y$, in X. This notation is extended to set of elements, and $\bigsqcup X$ is the least upper bound of the elements in the set X. A poset is a *chain-complete partial order* (cpo) if every directed set has a least upper bound. An element x of a cpo P is *compact* whenever, for all directed subset $X \subseteq P$ we have $x \sqsubseteq \bigsqcup X$ implies there exists $y \in X$ such that $x \sqsubseteq y$. Compactness is useful for the trivial consequence of the definition that if x is compact and $x = \sqcup_{y\in X} y$ then $x \in X$. Given an element x we denote $\downarrow x$ the set of compact elements $y \sqsubseteq x$.

Morphisms. The morphisms between directed-complete cpo are the *Scott-continuous* functions, *i.e.* the monotonic functions $f : A \to B$ such that for every directed subset $X \subseteq A$ we have $\bigsqcup_{x\in X} f(x) = f(\bigsqcup_{x\in X} x)$. Furthermore if $f(\bot_A) = \bot_B$ and f maps compact elements of A to compact elements of B we say that f is *strict*.

The $\mathbf{ALG}_{\bot!}$ *category.* Let D be a pointed dcpo and $d \in D$. A domain D is *algebraic* if for all $d \in D$ we have $d = \sqcup_{x\in d\downarrow} x$. The objects of the $\mathbf{ALG}_{\bot!}$ category are pointed algebraic domains and its morphisms are the strict continuous functions.

Example 1. The usual boolean lattice $\mathbb{B} = \{0, 1\}$ is an algebraic domain. We chose in the rest of this paper to order it with $1 \sqsubseteq 0$. The Belnap four-valued lattice [15] $\{U, T, F, C\}$ (unknown, true, false, contradictory) with $U \sqsubset T, F$ and $T, F \sqsubseteq C$ is also an algebraic domain. Any finite concept lattice [17] employed to classify hierarchically data is an algebraic domain. Given our definitions, any lattice where arbitrary joins are defined and which is algebraic is a representation domain.

3 Properties of the $\mathbf{ALG}_{\perp!}$ Category

We first define *worlds* as sets of traces in Sec. Section 3.1, and prove that they are objects in the $\mathbf{ALG}_{\perp!}$ category in Sec. Section 3.2. We use the fact that the set of morphims between two objects of $\mathbf{ALG}_{\perp!}$ is also an object in $\mathbf{ALG}_{\perp!}$ to introduce in Sec. Section 3.3 *split functions* that allow case-based and time-based reasoning. Representation functions that determine the characteristics of objects are introduced as a special case of morphisms in Sec. Section 3.4, and extended into World domain morphisms to allow for a more uniform treatment in the rest of this paper.

3.1 World Domains

We represent a sequence of events with words over an alphabet denoting the possible values of records in that table. As usual word concatenation is denoted with a dot, and the empty word is denoted ϵ. Given an alphabet A, we denote respectively A^* and A^ω the sets of finite and infinite words over A.

Any observable trace is a word in A^*, while words in A^ω represent the observations of infinite executions of a system. A system may be observed multiple times. Accordingly we define a *world* as a set of traces. We assume that learning is performed on a finite set of finite words. In contrast the actual possible traces of a system are infinite words and usually are in infinite numbers, *e.g.* in order to take into consideration random events.

The words of $A^* \cup A^\omega$ are ordered with the prefix order $u \sqsubseteq v$ iff u is a prefix of v. We let $+$ denote the union of sets of words, with the additional rule that for two words u and v, $u+v = v$ whenever $u \sqsubseteq v$, and with again the neutral element ϵ denoting a sum with just an empty word. From now on we assume every sum is written in a normal form, *i.e.* if $u = \Sigma_{v \in B \subseteq A^* \cup A^\omega} v$ then for $v, v' \in B$, we have $v \sqsubseteq v'$ implies $v = v'$. This leads to our definition of *worlds*.

Definition 1. (World) Given an alphabet A we let T_A be the set of sums of words of $A^\star \cup A^\omega$ in normal form, and call it the *world domain of* A.

3.2 Properties of World Domains

The following statements are trivial but may help the reader getting acquainted with these definitions.

Lemma 1. *The compact elements of a world domain are the finite sums of finite words.*

Proof. By definition if u is a finite sum of finite words then $u \in \downarrow u$, and it is then trivial that $u = \sqcup_{v \in \downarrow u} v$. Conversely an infinite word is the lub of its set of finite prefixes, and an infinite sum is the lub of the set of its distinct words. □

In the rest of this paper we call a compact element of a world domain an *observation*. A direct consequence of Lemma 1 is that world domains are algebraic.

Lemma 2. *Let T_A be a world domain on A. For every element $u \in T_A$ we have $u = \sqcup_{v \in \downarrow u} v$.*

Proof. Assume u is not compact for the statement is otherwise trivial. If u is an infinite word then $\downarrow u$ is the set of finite prefixes of u. By definition of infinite words we then have $u = \sqcup_{v \in \downarrow u} v$.

Finally if u is an infinite sum of words, say $u = \Sigma_{a \in I \subseteq A^* \cup A^\omega} a$, then by the two former cases we have $u = \bigsqcup_{a \in I \subseteq A^* \cup A^\omega} \bigsqcup_{v \in \downarrow u} v$ and thus $u \sqsubseteq \downarrow u$. The other direction is trivial, and thus $u = \bigsqcup \downarrow u$. □

World domains have a minimal element ϵ and thus are *pointed*. The following proposition is a directy consequence of Lemmas 2 and 1 for the object part, and of the definition of strict for the morphism part.

Proposition 1. *World domains are object of the $\mathbf{ALG}_{\perp !}$ category, and strict functions on world domains are morphisms of that category.*

We refer to [19], Sections 3.2 and 3.3 for the detailed results and their proofs. In the case of infinite products, the compact elements are those for which all but finitely coordinates have a bottom value, and the non-bottom values are compact. The part on morphisms is Prop. 4.2.4 of the same reference.

Proposition 2. *The $\mathbf{ALG}_{\perp !}$ category is closed for limits and colimits. In particular infinite products of objects in $\mathbf{ALG}_{\perp !}$ is also in $\mathbf{ALG}_{\perp !}$. If D, E are objects in $\mathbf{ALG}_{\perp !}$ then the domain of morphisms $[D \to E]$ is also in $\mathbf{ALG}_{\perp !}$.*

3.3 Split functions

Observations are analysed in a loop involving first the discovery of properties of events, then the classification of events according to these properties. Each class can then again be analysed and further classified, thereby providing a hierarchy of classes each with its own properties. While event manipulations and properties discovery are performed by generic morphisms, classification is the mapping of a world domain to a product of a denumerable number of copies of itself, each containing the events in a given class. Beyond classifying events based on their properties they can also be classified according to their time of occurence, *e.g.* to analyze the evolution of the different messages in a given class. Again each occurence can be stored in its own class so that time- or occurence-dependent characteristics can be observed.

A *split function* maps a world $u \in T_S$ to an element $\bar{u} \in T_S^{\mathbb{N}} = \Pi_{n \in \mathbb{N}} T_S$. As $T_S^{\mathbb{N}}$ is also the set of mappings from $\mathbb{N} \to T_S$, we denote $\bar{u}(n)$ the events in u that are in the class indexed by n. It is clear that if u is compact, $\bar{u}(n) \neq \varepsilon$ only for a finite number of coordinates.

Definition. Let T_A be a world domain. A mapping $c : T_A \to \mathbb{N}$ is a *choice function on T_A*. Given a choice function c, the *split function for c* is denoted Split_c and is

defined over compact elements as:

$$\begin{array}{rrcl} \text{Split}_c: & T_A & \to & (\mathbb{N} \to T_A) \\ & d & \mapsto & \{n \mapsto \pi_n(d)\} \\ where: & & & \\ \pi_n: & T_A & \to & T_A \\ u \mapsto & \left\{ \begin{array}{ll} \epsilon & \text{if } u = \epsilon \\ \pi_n(u') \cdot a & \text{if } u = u' \cdot a \text{ and } c(u) = n \\ \pi_n(u') & \text{if } u = u' \cdot a \text{ and } c(u) \neq n \\ \pi_n(v_1) + \pi_n(v_2) & \text{if } u = v_1 + v_2 \end{array} \right. & & \end{array}$$

More generally for $u \in T_S$ we define

$$\text{Split}_c(u) = \bigsqcup_{v \in \downarrow u} \text{Split}_c(v)$$

The split functions are continuous and strict.

Proposition 3. *Let T_A be a world domain, and c be a choice function on T_A. The split function for c is strict and continuous, and $\{(\mathbb{N} \to T_A)\}$ is an object of* $\mathbf{ALG}_{\perp!}$.

Proof. We construct $(\mathbb{N} \to T_A)$ as a limit of the functions on the finite prefixes of $\mathbb{N}$. By Prop. 2 this limit is still in $\mathbf{ALG}_{\perp!}$. Looking at the construction in the proof, its basis is the set of elements whose value at every coordinate is $\perp$, but for a finite number of coordinates for which it is a compact world. The Split_c function clearly maps compact elements to compact elements, and ϵ to the bottom element $(n \mapsto \epsilon)$. Thus it defines a unique strict and continuous mapping. □

3.4 Representations

The elementary brick of learning is the construction of a representation of observations. We do not put any restriction on the what representations are, and they may be numbers, subsets of $\mathbb{R}$, formulas, etc. By assumption we only impose that the representation is computed by a strict continuous function between pointed algebraic domains. In practice representations are ordered according to their generality, *i.e.* $a \sqsubseteq b$ if a is a particular case of b. The bottom represents a value so specific as to be impossible to achieve, while when it exists the top element represents an always true generalisation. In order to simplify the notations in the rest of this paper, we note that if T_S is a world domain, any morphism $f : T_S \to E$ computing a representation of T_S can be turned into a morphism $\hat{f} : T_S \to T_E$ as follows:

- The function f is first extended on traces with:

$$\hat{f}(u) = \left\{ \begin{array}{ll} \varepsilon & \text{if } u = \varepsilon \\ \hat{f}(v) \cdot f(a) & \text{if } u = v \cdot a \end{array} \right.$$

- The extension on world as set of traces is defined with:

$$\hat{f}(\Sigma_{a \in A \subseteq S^* \cup S^\omega} a) = \Sigma_{a \in A \subseteq S^* \cup S^\omega} \hat{f}(a)$$

The following lemma is trivial, but the fact that no hypothesis is needed on f actually shows that this construction is lacking.

Lemma 3. *Let T_S be a world domain and E be any object of* $\mathbf{ALG}_{\perp!}$. *For any function $f : T_S \to E$, the function $\hat{f} : T_S \to T_E$ defined above is a morphism in* $\mathbf{ALG}_{\perp!}$.

Reusing the denotations of Lemma 3 we say that $\hat{f} : T_S \to T_E$ is a *representation function* if f is an $\mathbf{ALG}_{\perp!}$ morphism, and that T_E in that case is a *representation domain*. A world domain which is not a representation domain is called on *object domain*. The *value* is the least upper bound of all values in the traces in the image.

Example 2. Any world domain whose alphabet is an algebraic domain as in Ex. 1 is a representation domain.

Definition 2. (Valuation) Let T_B be a representation domain. The *valuation* on T_B is the function:

$$\begin{array}{rccl} \mathrm{Val}^B : & T_B & \to & B \\ & \left\{ \begin{array}{rcl} \epsilon & \mapsto & \bot \\ u_1 \cdot \ldots \cdot u_n & \mapsto & \bigsqcup_{1 \leq i \leq n} u_i \\ u + v & \mapsto & \mathrm{Val}^B(u) \bigsqcup_B \mathrm{Val}^B(v) \end{array} \right. & \end{array}$$

Since the $x, y \mapsto x \sqcup y$ function is always continuous on a domain, the valuation function is always continuous.

Note. While no technical requirements on B is needed for the rest of this paper, we found that in naturally occuring cases the domain B in Def. 2 has a maximum element $\top_B$ which is compact. The former allows for the generalisation of a constraint $f(x) \sqsubseteq d$ into an always true constraint $f(x) \sqsubseteq \top_B$. The compacity of an element d allows one to consider besides $f(x) \sqsubseteq d$ the constraints $f(x) \sqsubset d$. In particular when $\top_B$ denotes the absence of useful representation, $f(x) \sqsubset \top_B$ indicates that a useful representation has been found.

4 Continuity of the Learning Process

This presentation is the generalisation of [25] that relies on the resolution of simple Data Exchange (DX) problems to fill tables in a database and on *filters* to learn integrity contraints. This generalisation follows the approach to DX of [5] in which rules are modeled as morphisms between objects that represent tables. Accordingly a database is modeled with a set of terms, each term denoting either the composition of functions filling a table (an object term) or a filter (a representation term). Beyond the update of the value of terms when presented with new observations, a second aspect of learning consists in computing more representations on existing tables and more tables for further analysis.

4.1 Notations

We let $D_0, D_1, \ldots, D_n$ be world domains, and we consider a set F of strict continuous $f_1, \ldots, f_m$ between these domains, *i.e.*:

$$f : D_{\alpha_f 1} \times \cdots \times D_{\alpha_f(k_f)} \to D_{\alpha_f(0)}$$

for a mapping $\alpha_f : \{0, \ldots, k_f\} \to \{1, \ldots, n\}$. We also let S be a set of split functions $\left\{\mathrm{Split}_{c,i}\right\}_{1 \leq i \leq n_s}$ over the world domains $D_{\alpha_c(i)}$ for $\alpha_c : \{1, \ldots, n_s\} \to \{1, \ldots, n\}$

First-order signature. To each domain D_i we associate a `sort` τ_i. The first-order sorted functional signature $\Sigma_{F,S}$ comprises

1. The set of function symbols $\mathrm{f}_1, \ldots, \mathrm{f}_m$ with sorts
$$\mathrm{f} : \tau_{\alpha_f(1)} \times \cdots \times \tau_{\alpha_f(k_f)} \to \tau_{\sigma_f(0)}$$
2. A constant "X":τ_0 denoting the world domain under analysis;
3. For each $\mathrm{Split}_{c,i} : D_{\alpha_c(i)} \to D_{\alpha_c(i)} \in S$ and each $n \in \mathbb{N}$, a function symbol $\mathrm{c}^n : \tau_{\alpha_c(i)} \to \tau_{\alpha_c(i)}$.

The set of *ground terms over* $\Sigma_{F,S}$ is as usual the least set $\mathrm{T}(\Sigma_{F,S})$ such that:

- "X":$\tau_0 \in \mathrm{T}(\Sigma)$;
- if $t_1{:}\tau_1, \ldots, t_{k_f}{:}\tau_{k_f} \in \mathrm{T}(\Sigma)$ and $\mathrm{f} : \sigma_f(1) \times \cdots \times \sigma_f(k_f) \to \sigma_f(0)$ then $f(t_1, \ldots, t_n){:}\sigma_f(0) \in \mathrm{T}(\Sigma)$;
- If $t : \tau \in \mathrm{T}(\Sigma)$, $t \neq \mathrm{c}^n(t')$ for some $n \in \mathbb{N}$, then for all $m \in \mathbb{N}$, $\mathrm{c}^m(t) \in \mathrm{T}(\Sigma)$.

The last rule prohibits the application of a split function on a class that is the result of the application of the same split function. In most cases the specific set of continuous functions and split functions is not relevant to the analysis, and we shall denote simply Σ a first-order signature as defined above. We shall also denote T_0 the world domain over which the variable "X" is interpreted.

Positions and subterms. A *position* is a finite sequence of integers, with ϵ denoting the empty sequence. Positions in and subterms of a term t are defined recursively as follows:

- t is a subterm of t at position ϵ;
- If $\mathrm{f}(t_1, \ldots, t_n)$ is a subterm at position p in t, then each t_i is a subterm at position $p \cdot i$ in t.

Given a term t we denote $\mathrm{Sub}(t)$ its set of subterms.

Representation and object terms. The set F of available morphisms can be partitioned into two sets F_r of *representation functions* and $F_o = F \setminus F_r$ of *object functions*. The terms $t : \tau$ such that $\tau = \alpha_f(0)$ for a representation function f are called *(ground) representation terms*. Otherwise they are called *(ground) object terms*. Given a set of terms S we denote $\mathrm{Repr}(S)$ the subset of S of representation terms.

Interpretation over a world $u \in D_0$. The interpretation of a function symbol f is denoted $[\mathrm{f}]$ and is f. The interpretation of a term t over an observation $u \in T_0$ is denoted $[t]_u$ and defined inductively as expected:

$$\begin{cases} [\text{"X"}]_u = u \\ [\mathrm{f}(t_1, \ldots, t_n)]_u = f([t_1]_u, \ldots, [t_n]_u) \\ [\mathrm{c}^n(t)]_u = \mathrm{Split}_c([t]_u)(n) \end{cases}$$

This interpretation is extended by continuity to worlds in D_0. If t is a representation term we denote $\mathrm{coDom}(t)$ the alphabet of the codomain of the interpretation of t, *i.e.* if $t = f(t_1, \ldots, t_n)$ and $[\mathrm{f}] : D_1 \times \ldots \times D_n \to D_0 = T_A$ then $\mathrm{coDom}(t) = A$. Finally, if t is a representation term of codomain T_B, its *value* on u is $\mathrm{Val}^B([t]_u)$ and is denoted more simply $\mathrm{Val}_u(t)$.

4.2 Data Exchange Systems (DXS)

A DXS is a set of terms over a signature $\Sigma = \Sigma_{F,S}$. with a few conditions that we explicit in this section.

Definition 3. (Data Exchange System) Let Σ be a signature. A Σ-*Data Exchange System* (Σ-*DXS*) is a non-empty set of terms $S \subseteq \mathrm{T}(\Sigma)$ such that:

- for each term $t' \in S$ and each $t' \in \mathrm{Sub}(t)$ we have $t' \in S$;
- If $\mathrm{c}^n(t) \in S$, then for all $m \in \mathbb{N}$, $\mathrm{c}^m(t) \in S$.

We denote $\mathcal{D}(\Sigma)$ the set of Σ-Data Exchange Systems.

The *initial DXS, i.e.* $\bot_{\mathcal{D}(\Sigma)}$ is {"X"}. Given two possible DXS S,S' we denote $S \sqsubseteq S'$ if $S \subseteq S'$ as sets. $\mathcal{D}(\Sigma)$ with the ordering $\sqsubset$ and $\Delta \bigsqcup \Delta' = \Delta \cup \Delta'$ is a domain with a maximal element, $\mathrm{T}(\Sigma)$ and a minimal element, the initial state.

Lemma 4. $\mathcal{D}(\Sigma)$ *is an object in* $\mathbf{DCPO}_{\bot!}$.

Proof. It clearly has a bottom element, so it suffices to prove it is a DCPO.

Assume there exists a signature Σ and a directed set $X \subseteq \mathcal{D}(\Sigma)$ such that $\sqcup_{S\in X} S \notin \mathcal{D}(\Sigma)$ Since DXS are bounded, there exists a minimal element (for inclusion) $S_X \in \mathcal{D}(\Sigma)$ such that $\sqcup_{S\in X} S \sqsubseteq S_X$. By contradiction assume that $S_X \neq \sqcup_{S\in X} S$, and thus that $S_X \setminus \sqcup_{S\in X} S = S' \neq \emptyset$. Since the subterm ordering on terms is well-founded, S' has a minimal element t for the subterm relation. Two cases are possible:

- If $t = \mathrm{f}(t_1, \ldots, t_n)$ then by minimality all $t_1, \ldots, t_n$ are in S_X and $\sqcup_{S\in X} S$. By minimality of S_X there is no term $t' \in S_X$ such that $t \in \mathrm{Sub}(t')$. Thus $S_X \setminus \{t\}$ is also a state, which contradicts the minimality of S_X;
- If $t = \mathrm{c}^n(t')$, the same reasoning applies, but with $S_X \setminus \{\mathrm{c}^n(t')\}_{n\in\mathbb{N}}$.

□

We now proceed to prove the algebraicity of the DXS domains. Since the split function adds an infinite number of terms we have to differentiate between finite and compact DXS, with the intention to define the latter as the result of a finite number of function applications. First they are redefined as (continuous) functions on $\mathcal{D}(\Sigma)$: For each ground term $t = \mathrm{f}(t_1, \ldots, t_n)$ we define the function:

$$\mathrm{Add}_t : \mathcal{D}(\Sigma) \to \mathcal{D}(\Sigma)$$
$$S \mapsto \begin{cases} S \cup \{t\} & \text{if } \{t_1, \ldots, t_n\} \subseteq S \\ S & \text{otherwise} \end{cases}$$

Similarly, given a term t and a choice function c the application of a function Split_c on a term t is extended to DXS with:

$$\mathrm{Split}_{c,t} : \mathcal{D}(\Sigma) \to \mathcal{D}(\Sigma)$$
$$S \mapsto \begin{cases} S \cup \{\mathrm{c}^n(t) \,|\, n \in \mathbb{N}\} & \text{if } t \in S \\ S & \text{otherwise} \end{cases}$$

We let $\mathcal{I}(\Sigma)$ be the denumerable set of functions Add_t and $\mathrm{Split}_{c,t}$ for all ground terms $t \in \mathrm{T}(\Sigma)$ and split function c. All these functions are by definition monotonous on compact elements, and extended by continuity on all DXS. An element $S \in \mathcal{D}(\Sigma)$ is *finitely reachable* if there exists a finite composition $f_n \circ \ldots \circ f_1$ of functions in $\mathcal{I}(\Sigma)$ such that $S = (f_n \circ \ldots \circ f_1)(\text{"X"})$. It is *reachable* if a finite or infinite such composition exists. The proof of algebraicity proceeds first by verifying that every element in $\mathbf{ALG}_{\perp!}$ is reachable and then by proving that every reachable element is the least upper bound of a set of finitely reachable elements.

Lemma 5. *Every $S \in \mathcal{D}(\Sigma)$ is reachable.*

Proof. Let $S \in \mathcal{D}(\Sigma)$. Let X be the set of reachable $S' \in \mathcal{D}(\Sigma)$ such that $S' \sqsubseteq S$. For $i \in \{1, 2\}$, if S_i is reachable by a composition F_i, then $S_1 \cup S_2$ is reachable by the composition in which functions of F_1 and F_2 are alternatively chosen. Thus the set X is directed. Let $R_S = \sqcup_{S'' \in X} S''$. We clearly have $R_S \subseteq S$.

By contradiction assume $S \setminus R_S \neq \emptyset$, and let t be in this difference. By induction on terms, there exists a finite composition that reaches a state containing $\mathrm{Sub}(t)$ from the initial state. By the above paragraph, this implies $t \in R_S$, a contradiction. □

Lemma 6. *A DXS $S \in \mathcal{D}(\Sigma)$ is compact `iff` it is finitely reachable.*

Proof. If S is finitely reachable then it is trivially compact. For the `if` part, if it is not finitely reachable, it is still reachable by Lemma 5, and thus is the sup of a strictly increasing chain of the states reached after a finite number of function composition, but equal to none of these states, and thus is not compact. □

Proposition 4. *$\mathcal{D}(\Sigma)$ is an object in $\mathbf{ALG}_{\perp!}$*

Proof. We already know it is in $\mathbf{DCPO}_{\perp!}$ by Lemma 4. By Lemma 5 every element is reachable. Given a state S let F be a function composition such that $S = F(\text{"X"})$. If F is finite, S is compact. Otherwise, finite prefixes construct a sequence of compact elements (by Lemma 6) whose sup is S. Since every element of $\mathcal{D}(\Sigma)$ is either a compact or the sup of compact elements below it, $\mathcal{D}(\Sigma)$ is algebraic. □

To continue our exploration of the $\mathbf{ALG}_{\perp!}$ category, we note that the construction of DXS can be formulated as a morphism between two objects of that category.

Proposition 5. *The function*

$$\mu : T_{\mathcal{I}(\Sigma)} \to \mathcal{D}$$
$$u \mapsto \begin{cases} \{\text{"X"}\} & \text{if } u = \epsilon \\ a(\sigma') & \text{if } \begin{cases} u = u' \cdot a \\ \mu(u') = \sigma' \end{cases} \\ \mu(u') \cup \mu(v') & \text{if } u = u' + v' \end{cases}$$

is continuous and strict.

On a philosophical note, while Prop. 5 has a low technical content, but it can be reformulated “Data Exchange Systems are representations of the possible observations”. It also implies a possible introspection mechanism since DXS can recursively be reasoned upon by other DXS, thereby laying the ground for a hierarchy of analyses of a system. We present two such examples in Sec. Appendix B.1 in the appendix.

Proposition 6. *Let Σ, Σ' be two representation signatures such that $\Sigma \subseteq \Sigma'$. Then the identity injection: $\iota : \mathcal{D}(\Sigma) \to \mathcal{D}(\Sigma')$ is an* $\mathbf{ALG}_{\perp!}$ *morphism.*

Thus a user can “teach” the system by giving it new functions, *e.g.* based on the analysis of the results with a prior set of functions.

5 Learning and Monitoring

This section presents how the system can learn from the analyzed world. The valuation of a DXS is introduced and shown to be continuous *wrt* both world and DXS in Sec. Section 5.1. A system monitor is constructed from this valuation in Sec. Section 5.2, and an ideal representation of the system is defined. It is also proved that the continuity properties and the algebraicity imply the convergence of the valuation learned to that ideal representation, and that the monitoring process itself is continuous. This latter property is employed in Sec. Section 5.3 to prove that the system presented enjoys an Angluin-like learning process, and thus that it is possible to decide whether to learn more traces of the network or add terms to the DXS in case of false positives or false negatives.

5.1 Continuity of the Valuation

First we prove that the valuation of terms in a domain is continuous as it is the composition of continuous functions.

Lemma 7. *For all terms t the mapping $u \in T_S \mapsto \mathrm{Val}^{D(t)}([t]_u)$ is an* $\mathbf{ALG}_{\perp!}$ *morphism.*

Proof. Given its definition as a least upper bound the $\mathrm{Val}^{D(t)}$ is continuous. It thus suffices to prove that the interpretation is continuous. By contradiction assume the set Ω of terms t such that the mapping $u \in T_S \mapsto [t]_u$ is not continuous is not empty. Let t be minimal for the well-founded subterm relation in Ω. We must have $t \neq$ "X". If $t = f(t_1, \ldots, t_n)$, then by minimality of t the functions $u \mapsto [t_i]_u$ are continuous, and thus by function composition the function $u \mapsto [t]_u$ must be continuous, a contradiction. □

We assume the product topology on the Cartesian products of domains, which is consistent with their construction as limits of expanding sequences (see [19], Theorem 3.3.7). In the following proposition, since the mapping to each coordinate is continuous by Lemma 7, the mapping to the product of interpretations is continuous.

Proposition 7. *Let Σ be a signature, and $S \in \mathcal{D}(\Sigma)$. Denote $D(t)$ denote the codomain of $\mathrm{Val}_{\_}(t)$. Then the function:*

$$\begin{array}{rccl} \mathrm{Val}^S : & T_0 & \to & \Pi_{t\in \mathrm{Repr(T(\Sigma))}} D(t)) \\ & u & \mapsto & \Pi_{t\in \mathrm{Repr(T(\Sigma))}} d_t \textit{ with} \\ & & & d_t = \begin{cases} \mathrm{Val}^{D(t)}([t]_u) & \textit{if } t \in S \\ \bot_{D(t)} & \textit{Otherwise} \end{cases} \end{array}$$

is continuous. Furthermore if S is compact then it is strict.

Proof. We already know by Lemma 7 its projection to each coordinate is continuous. Thus by the universal property of the product topology this function is continuous. Also it clearly maps ϵ to the product of the minimal elements, which is the minimal element of the product.

Let us prove that it is strict when S is compact. Let u be an observation in T_0. Since u is compact it contains a finite number of events. Since S is compact, it is finitely reachable and thus contains only a finite number of non-split terms, and a finite number of splits (each of which adding a denumerable number of terms). By Prop. 3 and since u is compact for each split only a finite number of terms have a non-bottom valuation.

Thus if S is compact, all but finitely elements of the product have the $\bot$ valuation, the remaining having a compact value. It is well known that these elements are the compact elements of the product. □

In order to simplify notation, we denote elements of a product $d = \Pi_{a\in\mathcal{A}} d_a$. The valuation is also continuous if the DXS changes with a fix world u.

Proposition 8. *Let Σ be a signature. Denote $D(t)$ denote the codomain of $\mathrm{Val}_{\_}(t)$. Then the function:*

$$\begin{array}{rccl} \mathrm{Val}_u : & \mathcal{D}(\Sigma) & \to & \Pi_{t\in \mathrm{Repr(T(\Sigma))}} D(t)) \\ & S & \mapsto & \mathrm{Val}_u^S \end{array}$$

is continuous. Furthermore if u is compact then it is strict.

Proof. We remark that "X" is not a representation term and thus Val_u maps the bottom element of $\mathcal{D}(\Sigma)$ to the bottom element of the product. Its continuity again is derived from the fact that the mapping to each coordinate t is continuous as it can only change once from $\bot_{D(t)}$ to $\mathrm{Val}^{D(t)}([t]_u)$.

When u is compact, that fact that it maps compact DXS to compact elements of the product is already proved in the proof of Prop 7. □

A function continuous in each argument being continuous on the product, we obtain the following theorem that characterizes our learning approach.

Theorem 1. *Let Σ be a signature, and denote $D(t)$ denote the codomain of $\mathrm{Val}_{\_}(t)$. Then the function:*

$$\begin{array}{rccl} \mathrm{Val} : & \mathcal{D}(\Sigma) & \to & \Pi_{t\in \mathrm{Repr(T(\Sigma))}} D(t)) \\ & (u, S) & \mapsto & \Pi_{t\in \mathrm{Repr(T(\Sigma))}} \mathrm{Val}_u^S(t) \end{array}$$

is strict and continuous.

Proof. Continuity stems from the continuity on each argument as in Propositions 7 and 8. Strictness also, remembering that the compact elements of a finite product are the products of compact elements. □

5.2 Monitoring

First, a trivial remark already used in the preceding proofs. Given a signature Σ we have $\mathrm{T}(\Sigma) \in \mathcal{D}(\Sigma)$ and thus, by the algebraicity of the latter, $\mathrm{T}(\Sigma) = \bigsqcup_{S \in \downarrow \mathrm{T}(\Sigma)} S$. In practical terms, Proposition 8 implies that the perfect representation of a world u can be acquired by examining all terms in the limit of the representations that are computed on compact elements. Second, it is natural to assume that not all world in T_0 are possible, but instead that only a subset of T_0 can be observed even if we could extend each trace to infinity. Let $u_{\lim} \in T_0$ be the world corresponding to that subset.

These two remarks lead to the definition of a limit element in $\Pi_{t \in \mathrm{Repr}(\mathrm{T}(\Sigma))} D(t)$ which is the top among the reachable elements.

Definition 4. (Limit valuation) Let Σ be a signature, and assume that all possible worlds of a system are bounded by $u_{\lim} \in T_0$. Then the *limit valuation* (of that system) is $\mathrm{Val}_{u_{\lim}}^{\mathrm{Repr}(\mathrm{T}(\Sigma))}$.

This valuation on all possible representation terms give a "*flight envelope*" to the system, in the sense that any behaviour outside of this envelope is an anomaly. We define *monitoring* as the act of checking that a world u is within the limits learned.

Definition 5. (Ground Literal) Let Σ be a signature. A *ground literal* is an expression $t \sqsubseteq d$ where $t \in \mathrm{T}(\Sigma)$ is a representation term, and $d \in \mathrm{coDom}(t)$.

A monitor is a conjunction of literals. It is convenient to order boolean values representing truth 1 and falsehood 0 with $1 \sqsubset 0$, thereby obtaining a finite domain $\mathbb{B}$ in $\mathbf{ALG}_{\perp!}$.

Definition 6. (Monitor) Let Σ be a signature and $D(t)$ denote the codomain of $\mathrm{Val}\ (()t)$ for a representation term t. Then for $r \in \Pi_{t \in \mathrm{Repr}(\mathrm{T}(\Sigma))} D(t)$, the r-monitor is the mapping:

$$\begin{array}{rccl} \mathcal{M}_r : & T_0 & \to & \mathbb{B} \\ & u & \mapsto & \sqcup_{t \in \mathrm{Repr}(\mathrm{T}(\Sigma))} \mathrm{Val}_u(t) \sqsubseteq r_t \end{array}$$

In the following lemma strictness is trivial as the two elements of $\mathbb{B}$ are compact, and continuity follows from the continuity of the valuation function (Lemma 7).

Lemma 8. *Let Σ be a signature, and d be in $\mathrm{coDom}(t)$ for a representation term t. Then the function $\mathcal{M}^{t,d} : u \mapsto \mathrm{Val}_u(t) \sqsubseteq d$ is strict and continuous. If furthermore d is compact then the function: $\mathcal{M}_s^{t,d} : u \mapsto \mathrm{Val}_u(t) \sqsubset d$ is continuous.*

Proof. Only the last statement is not trivial. Assume $u = \bigsqcup_{v \in V} v$ where V contains

only compact elements. We have to prove that $\mathrm{Val}_u(t) \sqsubset d = \bigsqcup_{v \in V} \mathrm{Val}_v(t) \sqsubset d$. By monotony of the valuation and $1 \sqsubset 0$ this is true if $\mathrm{Val}_u(t) \sqsubset d = 1$. Let us now assume this is not the case, and thus that $\mathrm{Val}_u(t) \sqsubset d = 0$ or equivalently that $d \sqsubseteq \mathrm{Val}_u(t)$.

The function $w \mapsto \mathrm{Val}_w(t)$ is continuous by Lemma 7 and thus $\mathrm{Val}_u(t) = \bigsqcup_{v \in \downarrow u} \mathrm{Val}_v(t)$. Since this function maps compact elements to compact elements all the $\mathrm{Val}_v(t)$ values are compact. By definition of compactness if $d \sqsubseteq \mathrm{Val}_u(t) = \bigsqcup_{v \in V} \mathrm{Val}_v(t)$ there exists $v \in V$ such that $d \sqsubseteq \mathrm{Val}_v(t)$ and thus such that $\mathrm{Val}_v(t) \sqsubset d = 0$. □

The proof of the following proposition is trivial since the value on each coordinate of the product is either 0 or 1. Thus the *set* of values occurring in the coordinates is directed, and thus its supremum is again 0 if it occurs at least once, or 1. It is thus an infinite conjunction.

Proposition 9. *Let Σ be a signature and $D(t)$ denote the codomain of $\mathrm{Val}\ (())t$ for a representation term t. Let $r \in \Pi_{t \in \mathrm{Repr(T(\Sigma))}} D(t)$, and $\mathcal{M}_r$ be the r-monitor. Then the mapping $u \in T_0 \mapsto \mathcal{M}_r(u) \in \mathbb{B}$ is strict and continuous.*

Note that in Proposition 9 the r-monitor is fixed. We leave to the reader that reusing the notations of Prop. 9 the function: $(u, r) \mapsto \mathcal{M}_r(u)$ is not continuous. This fact is exploited in the next section.

5.3 Angluin-like Learning

Recall that we let u_{lim} denote a world that is the least upper bound of all the possible worlds. A world $u \not\sqsubseteq u_{\mathrm{lim}}$ is an *anomaly*. An anomaly u is Σ-detectable if $\mathcal{M}_{\mathrm{Val}_{u_{\mathrm{lim}}}^{\mathrm{Repr(T(\Sigma))}}}(u) = 0$. It is a r-false negative if $\mathcal{M}_r(u) = 1$. Finally we say that $u \sqsubseteq u_{\mathrm{lim}}$ is a r-false positive if $\mathcal{M}_r(u) = 0$.

Angluin-like [4] denotes the possibility to converge to the right solution whenever false positive and false negatives to a proposed solution are given. We do not address in the following Theorem the computation of the set S' as it depends on enriching the functional signature with a theory using which one can compute representations that can differentiate two terms when they exist.

Theorem 2. *(Angluin-like Machine Learning) Let Σ be a signature, and $(u, S) \in \mathcal{D}(\Sigma)$ be a DXS. Let $r = \mathrm{Val}_u^{\mathrm{Repr}(S)}$ be the result of the learning phase, and $v \in T_0$ be a world.*

1. *if v is a detectable anomaly and a r-false negative, there exists S' such that $S \subseteq S'$ and $\mathcal{M}_{\mathrm{Val}_u^{\mathrm{Repr}(S')}}(v) = 0$;*
2. *if v is a normal behavior and a r-false positive, there exists u' such that $u \sqsubseteq u'$ and $\mathcal{M}_{\mathrm{Val}_{u'}^{\mathrm{Repr}(S)}}(v) = 1$.*

Proof. By strict continuity of the monitor fonction. We prove the first case, the second one can either be proved similarly or by taking $u' = u + v$.

Since v is Σ-detectable, $\mathcal{M}_{\mathrm{Val}_{u_{\mathrm{lim}}}^{\mathrm{Repr(T(\Sigma))}}}(v) = 0$. Since it is a r-false negative, $\mathcal{M}_{\mathrm{Val}_u^{\mathrm{Repr}(S)}}(v) = 1$. Since the function Val is strict continuous by Theorem 1, by

considering an increasing chain of compacts (u', S') above (u, S) and whose supremum is $(u_{\lim}, \mathrm{T}(\Sigma))$, there exists a compact DXS (u', S') such that $u \sqsubseteq u'$, $S \subseteq S'$, and $\mathcal{M}_{\mathrm{Val}_{u'}^{\mathrm{Repr}(S')}}(v) = 0$. Since we take the supremum of the values on each term, there exists a term $t \in S'$ such that $\mathrm{Val}_v(t) \not\sqsubseteq \mathrm{Val}_{u'}(t)$. Let us consider the possibilities for t.

Since the valuation is increasing on each term, for all $t \in S$ we have $\mathrm{Val}_u(t) \sqsubseteq \mathrm{Val}_{u'}(t)$, and thus $\mathrm{Val}_v(t) \not\sqsubseteq \mathrm{Val}_{u'}(t)$ implies $\mathrm{Val}_v(t) \not\sqsubseteq \mathrm{Val}_u(t)$. Since v is a false negative, we have for all $t \in S$ that $\mathrm{Val}_v(t) \sqsubseteq \mathrm{Val}_u(t)$. Thus there exists $t \in S' \setminus S$ such that $\mathrm{Val}_v(t) \not\sqsubseteq \mathrm{Val}_{u'}(t)$, and thus $\mathrm{Val}_v(t) \not\sqsubseteq \mathrm{Val}_u(t)$. Thus we have $\mathcal{M}_{\mathrm{Val}_u^{\mathrm{Repr}(S')}}(v) = 0$. □

Conclusion on learning and monitoring. The continuity of the learning process established in Theorem 1 implies the eventual convergence of the result of our learning algorithm towards an ideal description of the system. Theorem 2 is more precise in the sense that we know how to eliminate false positives–by learning more traces–and false negatives–by adding new terms to the DXS.

6 Relation with the Notion of Knowledge

We break from the tradition of taking machine learning inspiration from biological systems and turn instead to philosophy. Since Frege's initial formalisation of first-order logic, it has been the common view that Kant's approach to logic in *CPR* merely consists of–and for some authors at best–syntactical and grammatical distinctions. However interest in this logic has been raised in recent decades, starting with Longuenesse [3] reinterpretation of Kant's work. In [22], inverse systems of models have been proposed to formalize Kant's transcendental logic given its relation with models built from experience according to Kant. In particular the authors concluded that transcendental formulas should be expressed in geometric logic [20].

However no practical construction of such a system is provided. Furthermore it is assumed that predicates do not evolve with experience, only the domain does, though we can observe that often birds are defined as *good-looking animals that fly* while bugs are defined as *ugly, hairy animals that fly* before penguins and bats are considered. We close this gap in the rest of this report. Furthermore we argue that inverse systems of models assume a fixed experience, a limitation that cognitive states do not have.

6.1 Relation with Previous Formalisation

We advise the reader to start with [22] for a correct presentation, and provide here only a flawed summary. There are three different kinds of objects amenable to reasoning. The *objects of appearance* are those that are constructed from our perceptions of the world, and are accessible to reasoning through the "manifold" (multiplicity and arrangement) of their representations. The *objects of experience* are constructed from objects of appearance through *cognitions*. Both these objects may evolve as new experiences and new deductions are made. Finally, the *transcendental objects* can be *a priori* such as the functions in the signature or some schema guiding the construction of DXS, or they can be formed as an idealization of the objects of experience that does not need to be rooted in experience but needs to always hold. To take an example from [23], seeing or hearing a droplet is an object of appearance, the droplet object of experience

is formed from these droplets experienced by considering their common qualities observed. The transcendental object *droplet* is devised by imagining all the possible experiences involving a droplet.

Beyond that summary, a characteristic of Kant's approach to reasoning is that these constructions go in both directions: transcendental objects inform on the possible objects of experiences, transcendental objects and objects of experience guide the construction of objects of appearance to associate perceptions with the known transcendental objects or the objects of experience, and possibly modify the latter.

The main result of [22] is the definition of the relation between objects of experience and transcendental objects, and the constraints on the transcendental logic steming from that construction. Namely, the invariance is expressed by modeling transcendental objects as *threads* of an inverse system of models. Each thread intersect each "state of the mind" with an object of experience. A logic then is provided to reason on these transcendental objects.

Our approach.. In this report, the perceptions are the logs, while the object terms represent the *objects of appearance*. Representation terms constructed on an object term t are the *manifold of representations of* t, while literals are the judgements on that object. Cognitions as presented in [22] encompass without being limited to the construction of objects of appearance, of representations for these objects, and the forming of judgements from these representations, and thus the construction of a DXS is the result of cognitions.

Let (u, S) be a representation state. To build objects of experience, this state is enriched with a set R of non-ground representation terms and a subset $O \subseteq S$ of ground object terms. Adding terms to R and O is another form of cognition, just as reasoning on the cognitive state (u, S, O, R) introduced below to construct *objects of experiences*.

We propose two constructions in the rest of this section. In Sec. Section 6.2 objects of experience are constructed as unary predicates defined by representation terms and instantiated by object terms. In Sec. Section 7 they are defined by predicates composed of *pure* representation terms. The latter is employed in the rest of this report to construct an inverse system of first-order models, while the former is mostly relevant to our discussion on future works.

6.2 Learning Objects from Experience

We consider now a denumerable set $\mathcal{X}$ of variables indexed by a sort and an integer, and denote $\mathrm{T}(\Sigma, \mathcal{X})$ the set of well-sorted terms with variables in $\mathcal{X}$ over the signature Σ. The set of variables occuring in t is denoted $\mathrm{Var}(()t)$. A term $t \in T(\Sigma, \mathcal{X})$ is *abstract* if "X" $\notin \mathrm{Sub}(t)$. Let t, t' be two ground terms. If there exists a term t'' such that $\mathrm{Var}(t'') = \{x\}$ and $t''\{x \leftarrow t'\} = t$ we say that the abstraction of t' in t is defined, and denote t'' with $\mathrm{Abs}(t, t')$

Definition 7. (Pertinence relation) A ground representation term t *pertains to* a ground object term $t' \in \mathrm{Sub}(t)$, and we denote $\mathrm{Pert}(t) = t'$, if t' is maximal for the subterm relation such that the abstraction of t' in t is defined.

Proposition 10. *For every ground representation term* t*, there exists a unique term* t' *such that* $\mathrm{Pert}(t) = t'$*.*

Proof. Let $\Omega \subseteq \mathrm{Sub}(t)$ be the set of object terms t' such that $\mathrm{Abs}(t, t')$ is defined.

Since t is ground, we have "X" $\in \mathrm{Sub}(t)$, and thus "X" $\in \Omega$ which then is not empty. Thus it has a non-empty subset $\partial\Omega$ of maximal elements. Assume now that $\partial\Omega$ contains two distinct elements t_1, t_2. By maximality we can have neither $t_1 \in \mathrm{Sub}(t_2)$ nor $t_2 \in \mathrm{Sub}(t_1)$. Let $t'' = \mathrm{Abs}(t, t_2)$. Since $t = t''\{x \leftarrow t_2\}$ we have $\mathrm{Sub}(t) = \mathrm{Sub}(t'')\{x \leftarrow t_2\} \cup \mathrm{Sub}(t_2)$. Since $t_1 \notin \mathrm{Sub}(t_2)$ we have $t_1 \in \mathrm{Sub}(t'')\{x \leftarrow t_2\}$. Since $t_2 \notin \mathrm{Sub}(t_1)$ we have $t_1 \in \mathrm{Sub}(t'')$. Since t_1 is ground we have "X" $\in \mathrm{Sub}(t_1)$, and thus "X" $\in \mathrm{Sub}(t'')$. Thus t'' is not abstract, which contradicts the assumption $t_2 \in \Omega$. □

Definition 8. (Manifold of represention of an object term) Let S be a DXS, and $t \in \mathrm{T}(\Sigma)$ be a ground object term. The *manifold of representations of* t is denoted $\mathrm{Man}^S(t)$ and is the set $\{t' \in \mathrm{Repr}(S) \mid \mathrm{Pert}\, t' = t\}$.

If $t \notin S$, since DXS are closed for the subterm relation we have $\mathrm{Man}^S(t) = \emptyset$. The manifold of representations of an object term t defines an *object of experience* that gathers the judgements on t.

Definition 9. (Learned Object of Experience) Let $\lambda = (u, S)$ be a representation state, and t be a ground object term. The *object of experience learned from* t *in* (u, S) is denoted $\mathrm{Learn}_\lambda(t)$ and is:

$$\mathrm{Learn}_\lambda(t) = \begin{cases} 1 & \text{If } \mathrm{Man}^S(t) = \emptyset \\ \bigwedge_{t' \in \mathrm{Man}^S(t)} \mathrm{Abs}(t', t) \sqsubseteq \mathrm{Val}_u(t') & \\ & \text{Otherwise} \end{cases}$$

A particular is to consider for representation terms the actions (verbs) applicable on an object (noun). This approach was introduced in [16] for the similar task of learning concepts (objects of experience) from a corpus. Following that comparison the learned objects of experience are concepts.

7 Construction of First-Order Logic Models

Given a representation state $\lambda = (u, S)$ the set of objects of experience that can be learned in λ from any term t can be easily computed, but results only in unary predicates in the sense of formulas that have only one variable. Thus they only capture the intrisic properties of objects whereas typically objects are defined also in terms of their relations with other objects [8]. This section introduces a construction in which predicates can be constructed without limit on their arity (in Sec. Section 7.1), followed by the construction of a domain to interpret these predicates.

7.1 Construction of the relational signature

A literal of the form $t \sqsubseteq d$ where t is a pure representation term is called a *pure literal*.

Definition 10. (Predicate) A *predicate* $\varphi = \bigwedge_{t \in T_\varphi} t \sqsubseteq d_t$ is a conjunction of pure literals. Its *arity* is $|\cup_{t \in T_\varphi} \mathrm{Var}(t)|$.

Since "X" is an object constant, pure literals are necessarily abstract terms. The truth of a pure literal l in relation with a log u is defined through one or several substitutions σ such that $l\sigma$ is ground and satisfied by u.

Definition 11. (Grounding) Let O be a set of object terms and t be a pure representation term. A substitution θ *O-grounds* t if:

$$\begin{cases} \mathrm{coDom}(\theta) \subseteq O \\ t\theta \text{ is ground} \end{cases}$$

We denote $\mathrm{Gr}_O(t)$ the set of substitutions that O-grounds the term t.

Given a set of object terms O we denote $\mathrm{Subst}(O)$ the set of substitutions whose codomain is included in O. As a matter of convenience this notion is extended to literals and we denote $\mathrm{Gr}_O(t \sqsubseteq d)$ the set $\mathrm{Gr}_O(t)$ when t is a pure representation term.

Definition 12. (Support of a literal) Let O be a set of object terms and $u \in T_0$ be a log. A substitution σ *(u, O)-supports a pure literal* $l = r \sqsubseteq d$ if $\sigma \in \mathrm{Gr}_O(l)$ and $\mathrm{Val}_u(r\sigma) \sqsubseteq d$. We denote this fact with $\sigma \models_{(u,O)} l$, and denote $\mathrm{Supp}_u^O(l)$ the subset of $\mathrm{Gr}_O(l)$ of substitutions that (u, O)-supports the literal l.

In a representation state the DXS only reflects the observations of a system through the different processing steps performed on its log. We introduce *cognitive states* to model the reasoning steps that can be based on a representation state, *e.g.* to recognize that two object terms are instances of the same formula or are related through an observation.

Definition 13. (Cognitive State) A *cognitive state* is a tuple $K = (u, S, O, R)$ such that:

- (u, S) is a DXS;
- $O \subseteq \mathrm{Obj}^P(S)$ and for all $o \in O$ we have $[o]_u^S \neq \varepsilon$;
- R is a set of pure representation terms such that for $t, t' \in R$ we have $\mathrm{Var}(t) \cap \mathrm{Var}(t') = \emptyset$;

Given a cognitive state $K = (u, S, O, R)$ and $t \in R$ we denote $\mathrm{Gr}_K(t)$ the set of substitutions θ such that $\mathrm{Var}(t) \subseteq \mathrm{Dom}(\theta) \subseteq \mathrm{Var}(R)$, $\mathrm{coDom}(\theta) \subseteq O$, and $t\sigma \in S$ The set of substitutions *defined* in a cognitive state K is denoted $\mathrm{Subst}(K)$ and is the set $\cup_{t \in R} \mathrm{Gr}_K(t)$. Given a substitution $\theta \in \mathrm{Subst}(K)$, we let $\mathrm{Obs}_K(\theta)$ be the set of *observations* that can be made on the objects in the image of θ:

$$\mathrm{Obs}_K(\theta) = \{t \,|\, \theta \in \mathrm{Gr}_K(t)\}$$

We extend this notation to sets of substitutions with:

$$\mathrm{Obs}_K(\Theta) = \cap_{\theta \in \Theta} \mathrm{Obs}_K(\theta)$$

Example 3. Let $K = (u, S, O, R)$ be a cognitive state, $f : T \to T_D$ be a representation function, and $\mathrm{eq} : T_D \times T_D \to \{0, 1\}$ be an equality function, *i.e.* such that

$\mathrm{eq}(u,v)=1$ if and only if $u=v$. The representation term $\mathrm{eq}(f(x),f(y))$ is a binary predicate. Let Θ be the set of substitutions $\theta=\{x\mapsto t_1, y\mapsto t_2\}$ with $t_1,t_2\in O$ such that $\mathrm{eq}(f(t_1),f(t_2))=1$. By definition we have $\mathrm{eq}(f(x),f(y))\in\mathrm{Obs}_K(\Theta)$. Knowing that the predicate must be reflexive, symmetric, and transitive can help in the computation of Θ, and symmetrically the knowledge of a maximal set of substitutions Θ leads to the learning from experience that the predicate is reflexive, symmetric, and transitive. Since this report only aims at establishing a sound semantics, these aspects are out of its scope.

Since the set $\mathrm{Obs}_K(\Theta)$ represents the common qualities of the objects related by the substitutions in Θ it defines a *predicate of experience*.

Definition 14. (Predicate of Experience) Let K be a cognitive state, and $\Theta\subseteq\mathrm{Subst}(K)$. The *predicate of experience defined by* Θ *in* K is denoted $\mathrm{Obj}_K(\Theta)$ and is the formula:

$$\mathrm{Obj}_K(\Theta)=\begin{cases}1 & \text{If } \mathrm{Obs}_K(\Theta)=\emptyset\\ \bigwedge_{t\in\mathrm{Obs}_K(\Theta)} t\sqsubseteq\bigsqcup_{\theta\in\Theta}\mathrm{Val}_u(t\theta) & \\ & \text{Otherwise}\end{cases}$$

Its *arity* is the cardinal of $\cup_{t\in\mathrm{Obs}_K(\Theta)}\mathrm{Var}(t)$.

Def. 14 can be refined by assuming that some of the variables are existentially quantified to model ontologies [8], but this would lead to additional complexity in this report. We denote $\mathrm{Obj}^E(K)$ the set of predicates of experience $\mathrm{Obj}_K(\Theta)$ for $\Theta\subseteq\mathrm{Subst}(K)$. As in the case of Gr_O, Gr_K is extended to literals, and to predicates with $\mathrm{Gr}_K(\bigwedge_{t\in T} t\sqsubseteq d_t)=\cap_{t\in T}\mathrm{Gr}_K(t)$. We give the usual semantics to the conjunction by defining the *support of* $\varphi=\bigwedge_{t\in T} t\sqsubseteq d_t$ *in* K as $\cap_{t\in T}\mathrm{Supp}^O_u(t\sqsubseteq d_t)$, and denote it $\mathrm{Supp}^O_u(\varphi)$. By construction $\Theta\subseteq\mathrm{Supp}^O_u(\mathrm{Obj}_K(\Theta))$: the predicate defined by the examples in Θ is satisfied by these examples.

That a set of substitutions may entail the object formula of another set of substitutions yields a pre-order on these sets. In turns the pre-order yields an equivalence relation between sets of substitutions.

Definition 15. (Specialisation) Let K be a cognitive state, and $\Theta,\Theta'\subseteq\mathrm{Subst}(K)$. We say that Θ' *specialises* Θ, and denote $\Theta\preceq_K\Theta'$, if $\Theta'\subseteq\mathrm{Supp}^O_u(\mathrm{Obj}_K(\Theta))$.

The sets $\Theta,\Theta'\subseteq\mathrm{Subst}(K)$ are K*-equivalent*, and we denote $\Theta\equiv_K\Theta'$, if $\Theta\preceq_K\Theta'$ and $\Theta'\preceq_K\Theta$.

The equivalence classes for $\equiv_K$ are (pre-order) isomorphic with the predicates of experience on K.

Lemma 9. *Let* $K=(u,S,O,R)$ *be a cognitive state, and* $\Theta,\Theta'\subseteq\mathrm{Subst}(K)$ *be such that* $\Theta\equiv_K\Theta'$. *Then* $\mathrm{Obj}_K(\Theta)=\mathrm{Obj}_K(\Theta')$.

Proof. Let $\Theta,\Theta'\subseteq\mathrm{Subst}(K)$ be such that $\Theta\preceq_K\Theta'$.

By definition $\Theta\preceq_K\Theta'$ implies $\mathrm{Obs}_K(\Theta')\subseteq\mathrm{Obs}_K(\Theta)$, and thus by double inclusion $\Theta\preceq_K\Theta'$ implies $\mathrm{Obs}_K(\Theta)=\mathrm{Obs}_K(\Theta')$.

By definition of $\preceq_K$ we have for each $\theta'\in\Theta'$ that $u\models\mathrm{Obj}_K(\Theta)\theta'$. For each $r\in\mathrm{Obs}_K(\Theta)$, $u\models\mathrm{Obj}_K(\Theta)\theta'$ implies $\mathrm{Val}_u(r\theta')\sqsubset\bigsqcup_{\theta\in\Theta}\mathrm{Val}_u(r\theta)$. Inversing the roles

of θ, θ' we also get that for all $r \in \mathrm{Obs}_K(\Theta')$, and all $\theta \in \Theta$ we have $\mathrm{Val}_u(r\theta) \sqsubset \bigsqcup_{\theta' \in \Theta'} \mathrm{Val}_u(r\theta')$. Taken together these inequations yield for all $r \in \mathrm{Obs}_K(\Theta)$:

$$\bigsqcup_{\theta \in \Theta} \mathrm{Val}_u(r\theta) = \bigsqcup_{\theta' \in \Theta'} \mathrm{Val}_u(r\theta')$$

Given the two preceding paragraphs and the definition of Obj_K we have:

$$\mathrm{Obj}_K(\Theta) = \mathrm{Obj}_K(\Theta')$$

Conversely, $\mathrm{Obj}_K(\Theta) = \mathrm{Obj}_K(\Theta')$ implies, together with the fact that by construction $u \models \mathrm{Obj}_K(\Theta)\theta'$ for all $\theta' \in \Theta'$, and symmetrically when reversing the roles of Θ and Θ', that $\Theta \equiv_K \Theta'$. □

This tight coupling is employed to transfer the predicates of experience of a cognitive state to those of another through the sets of substitutions that generate them.

7.1.1 Construction of the domain

The construction of a domain from a cognitive state $K = (u, S, O, R)$ proceeds by considering the elements of O modulo an equivalence generated by $\mathrm{Obj}^E(K)$. This approach has two obstacles:

- A first modeling problem is whether it suffices to consider all comparisons between terms in O in the image of the possible substitutions, or if all the replacement of one term by another in O have to be considered. Both choices lead to an equivalence relation on O, the latter making it a congruence on terms. In this report we consider the former as it makes the exposition simpler. The former is non-trivial, but can be computed as a consequence of the termination of the Knuth-Bendix procedure on ground term rewriting systems [6]. It has been left out of this report out of space constraints;
- A second problem is that pairwise comparison of terms does not yield a transitive relation as some terms may be missing in S.

Considering closure under replacements. The following lemma allows for the transfer of a replacement on an instantiated term to a replacement on the substitution, and is used without references when considering ground instances of pure representation terms.

Lemma 10. *Let t be an object term, r be a pure representation, and θ be a substitution such that $r\theta$ is ground. Then if $(r\theta)_{|p} = t$ and $p \in \mathrm{Pos}(r)$ then $r_{|p}$ is a variable.*

Proof. Since t is an object term the symbol at position ϵ in t is in F_o. Since r is a pure representation term, this symbol cannot occur in r. Thus t can occur in $r\theta$ only at a position of a variable or below. □

Definition 16. A set of ground terms O is *replacement saturated* if for all $t, t_1, t_2 \in O$ and position $p \in \mathrm{Pos}(t)$, we have either $t_{|p} = t_1$ implies $t[p \leftarrow t_2] \in O$ or $t_{|p} = t_2$ implies $t[p \leftarrow t_1] \in O$.

The replacement saturation of a set of terms can be computed effectively and is finite.

Lemma 11. *Let $K = (u, S, O, R)$ be a compact cognitive state. Then there exists a finite and minimal set of ground term O' such that $O \subseteq O'$ and O' is replacement saturated.*

Proof. Let $<$ be a simplification ordering [14] on terms. We consider the ground term rewriting system $T = \{t \to t' \,|\, t, t' \in O \text{ and } t' < t\}$. The Knuth-Bendix completion of T always succeed (Corollary 6.2 in [6]) and yields a finite equivalent term rewriting sytem T'. Let O' be the minimal set of terms that contains O and such that $t \in O'$ and $t \to^{T'} t'$ implies $t' \in O$. Since T' is always terminating this set if finite, and it is replacement saturated by definition. □

To address the second problem we introduce variations of a substitution in which a term in its image is replaced by another term. Given a substitution θ of domain X, for $Y \subseteq X$ and t, t' terms we denote $\theta[t \leftarrow t']_Y$ the substitution:

$$\theta[t \leftarrow t']_Y(x) = \begin{cases} \theta(x) & \text{if } x \notin Y \\ t' & \text{if } x \in Y \text{ and } \theta(x) = t \end{cases}$$

Construction of the domain. We discussed earlier about the two choices when trying to compare two terms. The following definition assumes only the replacement in substitutions and not within terms. It can be adapted by furthermore assuming O is replacement saturated and considering the orbits for the symmetric closure of the $t \to t'$ rewriting rule: the terms t and t' are equivalent if and only if the valuation is constant over the orbit of each representation term.

Definition 17. (Pre-order and equivalence on O) Let $K = (u, S, O, R)$ be a compact cognitive state. Given $t, t' \in O$ we write $t \preceq^o_K t'$ if for all $\theta \in \text{Subst}(K)$, for all $X \subseteq \text{Dom}(\theta)$, and for all $r \in R$ we have $\text{Val}^S_u(r\theta[t \leftarrow t']_X) \sqsubseteq \text{Val}^S_u(r\theta)$.

We denote $t \equiv^o_K t'$ the fact that $t \preceq^o_K t'$ and $t' \preceq^o_K t$.

The closure by partial replacement may seem too stringent but is technically necessary to prove transitivity in the following lemma, which justifies the notations employes.

Lemma 12. *$t \preceq^o_K t'$ is pre-order on O.*

Proof. It is trivially reflexive. Assume $t, t', t'' \in O$ be such that $t \preceq^o_K t'$ and $t' \preceq^o_K t''$.

Let $\theta \in \text{Subst}(K)$, $X \subseteq \text{coDom}(\theta)$, and $r \in R$. We remark that $\theta[t \leftarrow t'']_X = \theta[t \leftarrow t'']_{X \cap \theta^{-1}(t)}$. It thus suffices to prove $\text{Val}^S_u(r\theta[t \leftarrow t'']_{X \cap \theta^{-1}(t)}) \sqsubseteq \text{Val}^S_u(r\theta)$.

By composition we have $r\theta[t \leftarrow t'']_{X \cap \theta^{-1}(t)} = r\theta[t \leftarrow t']_{X \cap \theta^{-1}(t)}[t' \leftarrow t'']_{X \cap \theta^{-1}(t)}$ and conclude with the hypothesis and the transitivity of $\sqsubseteq$. □

Lemma 12 imply that $\equiv^o_K$ is an equivalence relation on O.

The behaviour of this equivalence is problematic when considering the addition of representation terms in S. Assume $O = \{t_1, t_2\}$, $[t_1]^S_u = [t_2]^S_u$ and $R = \{f(x)\}$. If S contains only one of $f(t_1), f(t_2)$ we necessarily have $t_1 \not\equiv^o_K t_2$ but if it contains either none or both of them we have $t_1 \equiv^o_K t_2$. Starting from an empty set O and adding successively t_1 and t_2 makes the equivalence classes are thus not

monotonic. Accordingly *stable* cognitive states are those states such that S contains enough terms to properly evaluate all replacements of one term in O with another.

Definition 18. (Stable cognitive state) A cognitive state $K = (u, S, O, R)$ is *stable* if $\mathrm{Subst}(K)$ contains all the substitutions grounding a term in R and whose co-domain is included in$O$.

The essence of the following proposition is that the equivalence class $\equiv^o_K$ on O defines a finer equivalence than $\equiv_K$ on $\mathcal{P}(\mathrm{Subst}(K))$ (the first part), but that it is the coarsest equivalence on O that induces an equivalence finer than $\equiv_K$ on $\mathcal{P}(\mathrm{Subst}(K))$ (the second part).

Proposition 11. *Let K be a stable cognitive state, and $t, t' \in O$. If $t \equiv^o_K t'$ then for all $\Theta \subseteq \mathrm{Subst}(K)$ and $X \subseteq \cap_{\theta\in\Theta} \mathrm{Dom}(\theta)$ we have*

$$\mathrm{Obj}_K(\Theta) = \mathrm{Obj}_K(\{\theta[t \leftarrow t']_X \mid \theta \in \Theta\}$$

Conversely if for all $\Theta \subseteq \mathrm{Subst}(K)$ and $X \subseteq \mathrm{Dom}(\Theta)$ we have

$$\mathrm{Obj}_K(\Theta) = \mathrm{Obj}_K(\{\theta[t \leftarrow t']_X \mid \theta \in \Theta, X \subseteq \mathrm{Dom}(\theta)\})$$

then $t \equiv^o_K t'$.

Proof. If $t \equiv^o_K t'$ for all $\Theta \subseteq \mathrm{Subst}(K)$ and $X \subseteq \cap_{\theta\in\Theta} \mathrm{Dom}(\theta)$ we have for all $\theta \in \Theta$ that $\theta \equiv_K \theta[t \leftarrow t']_X$ by definition of $\equiv^o_K$, and thus $\Theta \equiv_K \Theta[t \leftarrow t']_X$ by definition of $\equiv_K$.

Conversely assume that for all $\Theta \in \mathrm{Subst}(K)$ and all $X \subseteq \cap_{\theta\in\Theta} \mathrm{Dom}(\theta)$ we have

$$\mathrm{Obj}_K(\Theta) = \mathrm{Obj}_K(\{\theta[t \leftarrow t']_X \mid \theta \in \Theta\})$$

By Lemma 9 $\mathrm{Obj}_K(\Theta) = \mathrm{Obj}_K(\{\theta\delta_{t,t'} \mid \theta \in \Theta\} \cap \mathrm{Subst}(K))$ is equivalent to $\Theta \equiv_K \{\theta[t \leftarrow t']_X \mid \theta \in \Theta\}$.

In particular for all $\theta \in \Theta$ and for all $r \in R$ such that $\theta \in \mathrm{Gr}_K(r)$ that $\mathrm{Val}^S_{r\theta[t\leftarrow t']_X}(\sqsubseteq) \mathrm{Val}^S_u(r\theta)$, and thus again by Definition 17 that $t \equiv^o_K t'$. □

Let $K = (u, S, O, R)$ be a stable cognitive state. By Prop. 11 for all $\varphi \in \mathrm{Obj}^E(K)$ of domain $x_1, \ldots, x_n$ and all $t_1, \ldots, t_n, t'_1, \ldots, t'_n \in O$ then if for all $1 \leq i \leq n$ we have $t_i \equiv^o_K t'_i$ then $u \models \varphi\{x_i \mapsto t_i\}_{1\leq i\leq n}$ if and only if $u \models \varphi\{x_i \mapsto t'_i\}_{1\leq i\leq n}$, and thus the valuation of the *predicates on experiences of* K is well-defined on the $O/\equiv^o_K$. This justifies the definition of the $\equiv^o_K$ equivalence classes as the *domain of* K.

Definition 19. (Domain defined by a cognitive state) Let $K = (u, S, O, R)$ be a stable cognitive state. The *domain of* K is the set $O/\equiv^o_K$.

Though there are no function symbols in the constructed models, we note that given two different constants c, c', there exists a predicate p and a substitution θ such that exactly one of $u \models p\theta$ and $u \models p\theta[c \leftarrow c']$ is valid. Conversely, if p, p' are two different predicates there exists a substitution θ such that exactly one of $u \models p\theta$ and $u \models p'\theta$ is valid.

Domain of cognitive states.. Given two cognitive states $K = (u, S, O, R)$ and $K' = (u', S', O', R')$ we say that K' is an extension K, and denote it $K \sqsubseteq K'$, if $u \sqsubseteq u'$, $S \subseteq S'$, $O \subseteq O'$, and $R \subseteq R'$.

Given a family of cognitive states $(K_f)_{f \in F}$ the supremum of the family is denoted $\bigsqcup_{f \in F} K_f$ and is the cognitive state $K = (u, S, O, R)$ where:

$$\begin{cases} u & = \cup_{f \in F} u_f \\ S & = \cup_{f \in F} S_f \\ R & = \cup_{f \in F} R_f \\ O & = \cup_{f \in F} O_f \end{cases}$$

A cognitive state is *compact* if u and S are compact, and R and O are finite.

8 Inverse systems of models

The construction of an inverse system on objects of experience in [22] relies on the existence for $K' \sqsubseteq K$ of functions:

1. $h_{K',K} : \mathrm{Obj}^E(K) \to \mathrm{Obj}^E(K')$ relating a predicate of experience in a state K to its less precise formulation in a preceding state K';
2. from the domain of K to the domain of K'.

The first requirement fails in our approach as in addition to cognitions, we consider passive observations: For any $u \sqsubseteq u'$ and any representation terms t, t' we may have $\mathrm{Val}_u(t) \neq \mathrm{Val}_u(t')$ but $\mathrm{Val}_{u'}(t) = \mathrm{Val}_{u'}(t')$, and conversely. In particular two object terms may be temporarily considered distinct before realising that they have the same properties. As a consequence we consider cognitive states with the same observed logs u. The second requirement fails whenever two equivalence classes of $\equiv^o_{K'}$ are merged into the same equivalence class of $\equiv^o_K$. This justifies the restriction in this section to stable cognitive states.

Given a *stable* cognitive state $K = (u, S, O, R)$ we denote $K\downarrow$ the set of stable cognitive states $K' = (u', S', O', R')$ such that $K' \sqsubseteq K$ and $u' = u$.

Definition 20. (Thread) Let $K = (u, S, O, R)$ be a stable cognitive state. A K-thread is a function $\xi : K\downarrow \to \mathcal{P}(O)$ such that:

(i) $\xi(K')$ is a $\equiv^o_{K'}$ equivalence class;

(ii) For all $K', K'' \in K\downarrow$, if $K'' \sqsubseteq K'$, then $\xi(K') \subseteq \xi(K'')$.

Let $K = (u, S, O, R)$ be a stable cognitive state and $K' = (u, S', O', R'), K'' = (u, S', O', R') \in K\downarrow$ with $K'' \sqsubseteq K'$. Threads have a few properties that need to be mentioned. We have $O'' \subseteq O'$ and thus $\mathrm{Subst}(K'') \subseteq \mathrm{Subst}(K')$. Since we additionally have $R'' \subseteq R'$, for each $\Theta \subseteq \mathrm{Subst}(K'')$ we have $\mathrm{Obs}_{K''}(\Theta) \subseteq \mathrm{Obs}_{K'}(\Theta)$. Since the log u does not change by definition we have the first-order logic syntactic tautology $\mathrm{Obj}_{K'}(\Theta) \Rightarrow \mathrm{Obj}_{K''}(\Theta)$. Thus for all $\Theta_1, \Theta_2 \subseteq \mathrm{Subst}(K'')$ we have $\Theta_1 \equiv_{K'} \Theta_2$ implies $\Theta_1 \equiv_{K''} \Theta_2$ by Lemma 9. Thus by Proposition 11 for all $t_1, t_2 \in O''$ we have $t_1 \equiv^o_{K'} t_2$ implies $t_1 \equiv^o_{K''} t_2$. The constraint on the chosen equivalence classes in Def. 20 can thus always be satisfied.

Also, we note that if ξ is a K-thread then $\xi_{|K'\downarrow}$ is a K'-thread. Together with the preceding remark we obtain that for every K-thread ξ we have $\xi(K'') =$

$\xi_{|K'\downarrow}(\xi(K'))$. *I.e.*, let Ξ_K be the set of K-threads. This set is in bijection with the $\equiv^o_K$ equivalence classes, *i.e.* the domain of K. In that sense the elements of the domain of K are the inverse limits of the K-threads.

We note that all the elements of the domain of a cognitive state K' are visited by at least one K-thread. Thus for all $K', K'' \in K\downarrow$ and $K'' \sqsubseteq K'$ there exists a projection:

$$\begin{array}{rrl} g_{K',K''}: & \mathrm{Dom}(K') & \to \mathrm{Dom}(K'') \\ & \xi(K') & \mapsto \xi(K'') \\ & & \text{for all } K' - \text{threads } \xi \in \Xi_{K'} \end{array}$$

Per Lemma 9 we have $\Theta \equiv_K \Theta'$ if, and only if, $\mathrm{Obj}_K(\Theta) = \mathrm{Obj}_K(\Theta')$. Thus $\mathrm{Obj}^E(K)$ is in bijection with $\mathcal{P}(\mathrm{Subst}(K))/\equiv_K$. Reusing the same argument as for the threads, there exists a mapping from $\mathcal{P}(\mathrm{Subst}(K))/\equiv_K$ to $\mathcal{P}(\mathrm{Subst}(K'))/\equiv_{K'}$ that maps each equivalence class C to an equivalence class that contains $C \cap \mathrm{Subst}(K')$.

Definition 21. (Coherence projection) Let K, K' be two DXS such that $K' \in K\downarrow$. The *coherence projection* of K into K' is the function $h_{K,K'} : \mathrm{Obj}^E(K) \to \mathrm{Obj}^E(K')$ defined on predicates of experience by:

$$h_{K,K'}(\mathrm{Obj}_K(\Theta)) = \begin{cases} 1 & \text{If } \Theta \cap \mathrm{Subst}(K') = \emptyset \\ \mathrm{Obj}_{K'}(\Theta \cap \mathrm{Subst}(K')) & \\ & \text{Otherwise} \end{cases}$$

Proposition 12. *Reusing the notations of Def. 21, the function $h_{K',K}$ is well-defined.*

Proof. [1] Let $\varphi \in \mathrm{Obj}^E(K')$. We have to prove that the value $h_{K',K}(\varphi)$ is uniquely defined. If is clearly the case if $\mathrm{Gr}_{K'}(\varphi) \cap \Sigma_K = \emptyset$ or if $\mathrm{Gr}_{K'}(\varphi) \cap \Sigma_K = \{\sigma\}$. Let us thus assume there exists two distinct substitutions $\sigma_1, \sigma_2 \in \mathrm{Gr}_{K'}(\varphi) \cap \Sigma_K$. Since $\mathrm{Obj}_{K'}(\sigma_1) = \mathrm{Obj}_{K'}(\sigma_2)$ we have $\mathrm{Obs}_{K'}(\sigma_1) = \mathrm{Obs}_{K'}(\sigma_2)$ and for all r in this set $\mathrm{Val}_u(r\sigma_1) = \mathrm{Val}_u(r\sigma_2)$. Also, since $K \sqsubseteq K'$ we have $\mathrm{Obs}_K(\sigma_i) \subseteq \mathrm{Obs}_{K'}(\sigma_i)$, for $i \in \{1, 2\}$ and by definition $\mathrm{Obs}_K(\sigma_1) \subseteq \mathrm{Obs}_K(\sigma_2)$. Since the log u has not changed, for all $r \in \mathrm{Obs}_K(\sigma_1)$ we have $\mathrm{Val}_u(r\sigma_1) = \mathrm{Val}_u(r\sigma_2)$, and thus $\mathrm{Obj}_K(\sigma_1) = \mathrm{Obj}_K(\sigma_2)$, which completes the proof. □

Proposition 13. *Let K, K' be two DXS such that $K \in K'\downarrow$, and let $h_{K',K} : \mathrm{Obj}^E(K') \to \mathrm{Obj}^E(K)$ be the coherence projection of K' into K. Then for all $\varphi \in \mathrm{Obj}^E(K')$ we have the first-order logic entailment $\models \varphi \Rightarrow h_{K',K}(\varphi)$.*

Proof. $h_{K',K}(\varphi)$ is a conjunction of a subset of the literals occuring in φ. □

Each cognitive state $K = (u, S, O, R)$ can be viewed as a first-order model on the signature $(\mathrm{Dom}(K), \mathrm{Obj}^E(K))$ with the interpretation that an atom $\varphi(a_1, \ldots, a_n)$ is true, with $\varphi \in \mathrm{Obj}^E(K)$ having variables $x_1, \ldots, x_n$, and $a_i \in \mathrm{Dom}(K)$ for $1 \leq i \leq n$ if, and only if, $u \models \varphi\{x_i \mapsto a_i\}$.

We have inverse morphisms on the domains and on the predicate of experience. By construction the inverse morphisms on equivalence classes preserve

[1] old notations

the satisfaction of inverse morphisms on objects of experience, that are interpreted as relations. These two sets of inverse morphisms define an *inverse system of models* as follows. We adapt the Def. 4 in [22] to take into account that the set of predicates may change from one model to another.

Definition 22. (Inverse system of models, Def. 4 in) Let $\{\mathcal{M}_s \,|\, s \in T\}$ be a family of first-order models indexed by a directed set T. Let D_s be the domain of $\mathcal{M}_s$, and $\mathcal{R}_s = \left\{R_1^s, \ldots, R_{n_s}^s\right\}$ be its set of predicate symbols.

Let $\mathcal{F}$ be a family of model homomorphisms

$$\{(h_{st}, g_{st}) \,|\, t \sqsubseteq s, h_{st} : \mathcal{R}_s \to \mathcal{R}_t \text{ and } g_{st} : D_s \to D_t\}$$

that satisfies the two following properties:

coherence: For all $t \sqsubseteq s \sqsubseteq r$ we have $h_{tr} \circ h_{st} = h_{sr}$ and $g_{tr} \circ g_{st} = g_{sr}$;

model homomorphism: For all $t \sqsubseteq s$ we have:

$$\mathcal{M}_s \models R_i^s(a_1, \ldots, a_n) \text{ implies } \mathcal{M}_t \models h_{st}(R_i^s)(g_{st}(a_1), \ldots, g_{st}(a_n))$$

Then $\{\mathcal{M}_s \,|\, s \in T\}$ together with the family $\mathcal{F}$ of homomorphisms is an *inverse system of models*.

The construction given yields trivially our main theorem.

Theorem 3. *Let $K = (u, S, O, R)$ be a cognitive state. For $K' = (u', S', O', R') \in K\downarrow$, let $\mathcal{R}_{K'} = \mathrm{Obj}^E(K')$, $D_{K'} = / \equiv_{K'}$.*

Then $(K\downarrow, \left\{(\mathrm{Dom}(K'), \mathrm{Obj}^E(K')) \,|\, K' \in K\downarrow\right\}, \mathcal{F})$, with $\mathcal{F}$ the family of inverse model homomorphisms

$$\{(h_{K',K''}, g_{K',K''} \,|\, K', K'' \in K\downarrow \text{ and } K'' \sqsubseteq K'\}$$

is an inverse system of models of inverse limit K.

Consequence.. The connection for a fixed u with the work of [22] shown by Theorem 3 implies that we can define *objective validity*, and that objectively valid formulas are either existentially quantified geometric formulas or universally quantified geometric implications.

9 Conclusion

The approach to learning presented in this report formalises the existing *ensemble learning* technique in a discrete setting where the real numbers usually associated with Machine Learning (ML) are replaced by values in domains, and ML algorithms are modeled as continuous functions over these values. It builds a model of the system that represents a *flight envelope* for its possible behaviors and monitors the network with literals denoting that the value observed in the network must be more specific than a value fixed after the learning phase. In practical terms the approach presented in this report is sufficiently explainable to allow for guiding an expert user reviewing the results into either constructing new algorithms or proposing novel representation construction as in [25].

We then extended this construction to construct an inverse system of models. The implications of this extension are best presented in [22] but we sum them up here. First there is a clear delineation between a transcendental logic formula that is evaluated on the threads of an inverse system and a formula of experience that is evaluated locally in each model. By Theorem 3 they coincide in the limit though it usually cannot be effectively computed. However a user can always add formulas that are known to be true as transcendental formulas. These formulas are satisfied in all stable states and form a background theory with which it is possible to reason in each stable state. A result of [22] is that geometric implications [18, 21] are well-behaved *wrt* the evaluation on an inverse system of models.

References

[1] Rakesh Agrawal, Dimitrios Gunopulos, and Frank Leymann. Mining process models from workflow logs. In Hans-Jörg Schek, Gustavo Alonso, Felix Saltor, and Isidro Ramos, editors, *Advances in Database Technology — EDBT'98*, pages 467–483, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[2] AUTOSAR. Specification of can transport layer. Technical Report 014, AUTOSAR, December 2017.

[3] Béatrice Longuenesse. *Kant and the Capacity to Judge: Sensibility and Discursivity in the Transcendental Analytic of the Critique of Pure Reason*. Princeton University Press, Princeton, NJ, USA, 2001.

[4] Dana Angluin. Learning Regular Sets from Queries and Counterexamples. *Inf. Comput.*, 75(2):87–106, 1987.

[5] David I. Spivak and Ryan Wisnesky. On The Relational Foundations Of Functorial Data Migration. *CoRR*, abs/1212.5303, 2012.

[6] Deepak Kapur and Paliath Narendran. The Knuth-Bendix Completion Procedure and Thue Systems. *SIAM J. Comput.*, 14(4):1052–1072, 1985.

[7] Community Effort. Obd-ii semantics of can frames, 2019. Owners' assessment of the frame ID semantics in a Tesla Model 3 CAN Bus, available at https://docs.google.com/spreadsheets/d/1ijvNE4lU9Xoruvcg5AhUNLKr7xYyHcxa8YSkTxAERUw/edit#gid=0.

[8] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, Cambridge, UK, 2003.

[9] Jeremy Gunawardena. Deducing causal relationships in CCS. In C. E.Veni Madhavan, editor, *Foundations of Software Technology and Theoretical Computer Science*, pages 161–170, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg.

[10] Immanuel Kant. *Critique of the Pure Reason ; Translated from the German by Paul Guyer and Allen W. Wood*. Cambridge University Press, 1998.

[11] ISO. Road vehicles — diagnostic communication over controller area network (docan) — part 2: Transport protocol and network layer services. Technical Report 15765-2, ISO, 2016.

[12] Abir Laraba, Jérôme François, Shihabur Rahman Chowdhury, Isabelle Chrisment, and Raouf Boutaba. Mitigating TCP protocol misuse with programmable data planes. *IEEE Trans. Netw. Serv. Manag.*, 18(1):760–774, 2021.

[13] Mark V. Lawson. 1. Classical Stone Duality. 2018.

[14] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite Systems. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 243–320. Elsevier and MIT Press, 1990.

[15] Nuel D. Belnap Jr. A Useful Four-Valued Logic. In *Modern Uses of Multiple-Valued Logic*, pages 5–37. D. Reidel Publishing Company, 1977.

[16] Philipp Cimiano, Andreas Hotho, and Steffen Staab. Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis. *J. Artif. Intell. Res.*, 24:305–339, 2005.

[17] Rudolf Wille. Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts. In Ivan Rival, editor, *Ordered Sets*, volume 83 of *NATO Advanced Study Institutes Series*, pages 445–470. Springer Netherlands, 1982.

[18] Samson Abramsky. Domain Theory in Logical Form. *Ann. Pure Appl. Log.*, 51(1-2):1–77, 1991.

[19] Samson Abramsky, Dov M. Gabbay, and T. S. E. Maibaum. *Handbook of logic in computer science. Volume 3. Semantic Structures*. Clarendon Press, 1994.

[20] Steve Vickers. Geometric Logic in Computer Science. In Geoffrey Burn, Simon Gay, and Mark D. Ryan, editors, *Theory and Formal Methods 1993*, pages 37–54, London, 1993. Springer London.

[21] Steven Vickers. Continuity and Geometric Logic. *J. Appl. Log.*, 12(1):14–27, 2014.

[22] Theodora Achourioti and Michiel van Lambalgen. A Formalization of Kant's Transcendental Logic. *Rev. Symb. Log.*, 4(2):254–289, 2011.

[23] Theodora Achourioti and Michiel van Lambalgen. Kant's Logic Revisited. *FLAP*, 4(4), 2017.

[24] J. Wardell. Log of a tesla model 3 car, 2019. available at https://www.dropbox.com/s/vpz5b0c78qmqlt8/Model3Log2019-10-02v10.asc.zip?dl=0&file_subpath=%2FModel3Log2019-10-02v10.asc.

[25] Yannick Chevalier. Data Exchange for Anomaly Detection: The Case of the CAN Bus. In *Conference on Artificial Intelligence for Defense (CAID 21)*, pages 25–32, Rennes, France, November 2021. DGA : Direction Générale de l'Armement - Ministère français des Armées.

[26] Yannick Chevalier and Michaël Rusinowitch. Implementing Security Protocol Monitors. In Temur Kutsia, editor, *Proceedings of the 9th International Symposium on Symbolic Computation in Software Science, SCSS 2021, Hagenberg, Austria, September 8-10, 2021*, volume 342 of *EPTCS*, pages 22–34, 2021.

A Proofs

The proof of Lemma 1 is a rephrasing of the case analysis of the proof of Lemma 2.

Lemma 2. *Let T_A be a log domain on A. For every element $u \in T_A$ we have $u = \sqcup_{v \in \downarrow u} v$.*

Proof. By definition if u is a finite sum of finite words then $u \in \downarrow u$, and it is then trivial that $u = \sqcup_{v \in \downarrow u} v$.

Also if u is an infinite word then $\downarrow u$ is the set of finite prefixes of u. By definition of infinite words we then have $u = \sqcup_{v \in \downarrow u} v$.

Finally if u is an infinite sum of words, say $u = \Sigma_{a \in I \subseteq A^* \cup A^\omega} a$, then by the two former cases we have $u = \bigsqcup_{a \in I \subseteq A^* \cup A^\omega} \bigsqcup_{v \in \downarrow u} v$ and thus $u \sqsubseteq \downarrow u$. The other direction is trivial, and thus $u = \sqsubseteq \downarrow u$. □

Proposition 3. *Let T_A be a world domain, and c be a choice function on T_A. The split function for c is strict and continuous, and $\{(\mathbb{N} \to T_A)\}$ is an object of* $\mathbf{ALG}_{\perp !}$.

Proof. We construct $(\mathbb{N} \to T_A)$ as a limit of the functions on the finite prefixes of $\mathbb{N}$. By Prop. 2 this limit is still in $\mathbf{ALG}_{\perp !}$. Looking at the construction in the proof, its basis is the set of elements whose value at every coordinate is $\perp$, but for a finite number of coordinates for which it is a compact world. The Split_c function clearly maps compact elements to compact elements, and ϵ to the bottom element $(n \mapsto \epsilon)$. Thus it defines a unique strict and continuous mapping. □

Lemma 4. *$\mathcal{D}(\Sigma)$ is an object in* $\mathbf{DCPO}_{\perp !}$.

Proof. It clearly has a bottom element, so it suffices to prove it is a DCPO.

Assume there exists a signature Σ and a directed set $X \subseteq \mathcal{D}(\Sigma)$ such that $\sqcup_{S \in X} S \notin \mathcal{D}(\Sigma)$ Since DXS are bounded, there exists a minimal element (for inclusion) $S_X \in \mathcal{D}(\Sigma)$ such that $\sqcup_{S \in X} S \sqsubseteq S_X$. By contradiction assume that $S_X \neq \sqcup_{S \in X} S$, and thus that $S_X \setminus \sqcup_{S \in X} S = S' \neq \emptyset$. Since the subterm ordering on terms is well-founded, S' has a minimal element t for the subterm relation. Two cases are possible:

- If $t = \mathrm{f}(t_1, \ldots, t_n)$ then by minimality all $t_1, \ldots, t_n$ are in S_X and $\sqcup_{S \in X} S$. By minimality of S_X there is no term $t' \in S_X$ such that $t \in \mathrm{Sub}(t')$. Thus $S_X \setminus \{t\}$ is also a state, which contradicts the minimality of S_X;
- If $t = \mathrm{c}^n(t')$, the same reasoning applies, but with $S_X \setminus \{\mathrm{c}^n(t')\}_{n \in \mathbb{N}}$.

□

Lemma 5. *Every $S \in \mathcal{D}(\Sigma)$ is reachable.*

Proof. Let $S \in \mathcal{D}(\Sigma)$. Let X be the set of reachable $S' \in \mathcal{D}(\Sigma)$ such that $S' \sqsubseteq S$. For $i \in \{1, 2\}$, if S_i is reachable by a composition F_i, then $S_1 \cup S_2$ is reachable by the composition in which functions of F_1 and F_2 are alternatively chosen. Thus the set X is directed. Let $R_S = \sqcup_{S'' \in X} S''$. We clearly have $R_S \subseteq S$.

By contradiction assume $S \setminus R_S \neq \emptyset$, and let t be in this difference. By induction on terms, there exists a finite composition that reaches a state containing $\mathrm{Sub}(t)$ from the initial state. By the above paragraph, this implies $t \in R_S$, a contradiction. □

Lemma 6. *A DXS $S \in \mathcal{D}(\Sigma)$ is compact `iff` it is finitely reachable.*

Proof. If S is finitely reachable then it is trivially compact. For the `if` part, if it is not finitely reachable, it is still reachable by Lemma 5, and thus is the sup of a strictly increasing chain of the states reached after a finite number of function composition, but equal to none of these states, and thus is not compact. □

Proposition 4. *$\mathcal{D}(\Sigma)$ is an object in* $\mathbf{ALG}_{\perp!}$.

Proof. We already know it is in $\mathbf{DCPO}_{\perp!}$ by Lemma 4. By Lemma 5 every element is reachable. Given a state S let F be a function composition such that $S = F(\text{"X"})$. If F is finite, S is compact. Otherwise, finite prefixes construct a sequence of compact elements (by Lemma 6) whose sup is S. Since every element of $\mathcal{D}(\Sigma)$ is either a compact or the sup of compact elements below it, $\mathcal{D}(\Sigma)$ is algebraic. □

Lemma 8. *Let Σ be a signature, and d be in $\mathrm{coDom}(t)$ for a representation term t. Then the function $\mathcal{M}^{t,d} : u \mapsto \mathrm{Val}_u(t) \sqsubseteq d$ is strict and continuous. If furthermore d is compact then the function: $\mathcal{M}_s^{t,d} : u \mapsto \mathrm{Val}_u(t) \sqsubset d$ is continuous.*

Proof. Only the last statement is not trivial. Assume $u = \bigsqcup_{v \in V} v$ where V contains only compact elements. We have to prove that $\mathrm{Val}_u(t) \sqsubset d = \bigsqcup_{v \in V} \mathrm{Val}_v(t) \sqsubset d$. By monotony of the valuation and $1 \sqsubset 0$ this is true if $\mathrm{Val}_u(t) \sqsubset d = 1$. Let us now assume this is not the case, and thus that $\mathrm{Val}_u(t) \sqsubset d = 0$ or equivalently that $d \sqsubseteq \mathrm{Val}_u(t)$.

The function $w \mapsto \mathrm{Val}_w(t)$ is continuous by Lemma 7 and thus $\mathrm{Val}_u(t) = \bigsqcup_{v \in V} \mathrm{Val}_v(t)$. Since this function maps compact elements to compact elements all the $\mathrm{Val}_v(t)$ values are compact. By definition of compactness if $d \sqsubseteq \mathrm{Val}_u(t) = \bigsqcup_{v \in V} \mathrm{Val}_v(t)$ there exists $v \in V$ such that $d \sqsubseteq \mathrm{Val}_v(t)$ and thus such that $\mathrm{Val}_v(t) \sqsubset d = 0$. □

B Example application: Detection of Network Protocols

B.1 Detection of purely parallel processes

While process mining [1] usually focuses on the discovery of workflow graphs encoding conditionals and loops, the detection of synchronisation between messages is more often, as *e.g.* in the case of a protocol, a sequential activity without holes: Messages a and b in an order are always followed by messages c and d, etc. In the case of the CAN Bus these sequences occur in two contexts. In a MFM protocol execution that starts with an initial message, replied to with an ack, and followed by a sequence of messages with a counter. Or in the case of a proper design of the CAN Bus in which such synchronisation is introduced to avoid collisions between messages. These are examples of *purely parallel* processes [9] that are built with sequential and parallel composition only, *i.e.* with no non-deterministic choice.

Let S be a DXS. A subset $S_c \subseteq S$ of terms which are applications of split functions on "X" is called a *pre-classification*. If it is maximal then it is the classification of the DXS. If the codomain of "X" is the set of possible CAN frames, then the codomain of each t in the classification S_c is also the set of CAN frames. We assume that all the events in the log are distinct *e.g.* by adding a time of capture. To simplify notations we denote $e \in u$ if the event e occurs at some position in the word u, and $|u|$ the length of the word u.

Algorithm. Let S be a DXS with a classification set S_c. A representation function nb_occ is applied on each $t \in S_c$ to count the number of occurrences of messages in the interpretation of the term t. The state of the network after seeing a trace u is represented by a boolean square matrix$P_{S_c}(u)$ indexed by terms in S_c and defined inductively on traces with:

$$P^u_{S_c}(t,t') = \begin{cases} 1 & \text{If } u = \varepsilon \\ P^v_{S_c}(t,t') \wedge (\text{nb\_occ}(t') \neq \text{nb\_occ}(t') & \\ P^u_{S_c}(t,t') & \text{Otherwise} \end{cases}$$

When analyzing a log $\Sigma_{u\in U} u$ we let

$$P^{\Sigma_{u\in U} u}_{S_c} = \Sigma_{u\in U} P^u_{S_c}$$

be the coordinate-wise conjunctions of the matrice $P^u_{S_c}$ for $u \in U$. Ordering the matrice with the extension on all coordinates of the ordering on booleans, one easily checks that the conjunction being the sup of boolean values, this function is monotonic and strict and thus can be extended by continuity to infinite traces. We note it is defined on a DXS built when analyzing the network, see Prop. 5.

In the following definition, the semi-colon denotes sequential composition, a set of term denotes a parallel composition of these terms, and the while *true* do denotes an unbounded iteration of the process.

Definition 23. (Purely Parallel Process—PPP Let S be a DXS with a classification set S_c. An *iterated purely parallel process* is a process of the while *true* do $(A_1; \ldots; A_n)$ where $A_1, \ldots, A_n$ is a partition of a subset of S_c.

A PPP $P =$ while *true* do $(A_1; \ldots; A_n)$ is *unambiguous* on trace $u = (e_i)_{1\leq i\leq N}$ if furthermore for all $1 \leq i \leq N$ if e_i occurs in $[t]_u$ and $intput'$ for $t, t' \in \cup_{i=1}^n A_i$ then $t = t'$.

Lemma 13. *Assume the purely parallel process* $P =$ while *true* do $(A_1; \ldots; A_n)$ *is executed on the network, that the trace* u *is observed, and that* P *is unambiguous for* u*. Then for all* $1 \leq i < j \leq n$ *and for all* $t' \in A_i, t \in A_j$ *we have:*

$$|[t']_u| = |[t]_u| \text{ or } |[t']_u| = |[t]_u| + 1$$

Proof. We first prove the following statement that is satisfied by the less constrained process:

$$P = \text{while } \textit{true} \text{ do } (\cup_{i=1}^{n} A_i)$$

Claim. For every trace u there exists L_u, H_u, and E_u such that:

- L_u, H_u is a partition of $\cup_{i=1}^{n} A_i$ and $L_u \neq$;
- $E_u \subseteq \cup_{i=1}^{n} A_i$ is the subset of the terms of the process that have not been observed in the current iteration of the process;
- The following equalities are satisfied:

$$\begin{cases} \forall t, t' \in L_u, & |[t]_u| & = & |[t']_u| & (A) \\ \forall t, t' \in H_u, & |[t]_u| & = & |[t']_u| & (B) \\ \forall t \in L_u, \forall t' \in H_u, & |[t]_u| + 1 & = & |[t']_u| & (C) \\ E_u = L_u & & & & (D) \end{cases}$$

Proof of the claim. By induction on the length of the trace u.

- If $u = \varepsilon$ let $H_\varepsilon = \emptyset$ and $L_\varepsilon = E_\varepsilon = \cup_{i=1}^{n} A_i$. Since $H_u = \emptyset$ the equations (C) and (D) are trivially satisfied. The interpretation is strict so the equation (A) is satisfied (and the length is equal to 0). The equation (D) is satisfied by definition.
- Assume the claim stands for a trace u and consider the trace $u \cdot e$.
  - If $e \notin \cup_{i=1}^{n} \cup_{t \in A_i} [t]_{u \cdot e}$, *i.e.* the observed event e is not associated with a term in the process, then setting $L_{u \cdot e} = L_u$, $H_{u \cdot e} = H_u$, and $E_{u \cdot e} = E_u$ satifies the constraints of the claim;
  - Otherwise there exists $t \in \cup_{i=1}^{n} A_i$ such that $e \in [t]_{u \cdot e}$. Since the process is unambiguous the term t has not been observed in the current iteration of the loop, and thus $t \in E_u$. By induction this implies $t \in L_u$. One then easily checks that the equalities are all satisfied by setting:
    - If $L_u = \{t\}$: $L_{u \cdot e} = E_{u \cdot e} = \cup_{i=1}^{n} A_i$ and $H_{u \cdot e} = \emptyset$;
    - Otherwise $L_{u \cdot e} = L_u \setminus \{t\}$, $E_{u \cdot e} = L_{u \cdot e}$, and $H_{u \cdot e} = H_u \cup \{t\}$;

□

Claim. For any trace u, there exists $1 \leq i_t \leq n$ such that $H_u \subseteq \cup_{i=1}^{i_t} A_i$, $L_u \subseteq \cup_{i=i_t}^{n} A_i$, and if $E_u \neq E_{u \cdot e}$ there exists $t \in L_u \cap A_{i_t}$ such that $e \in [t]_u$.

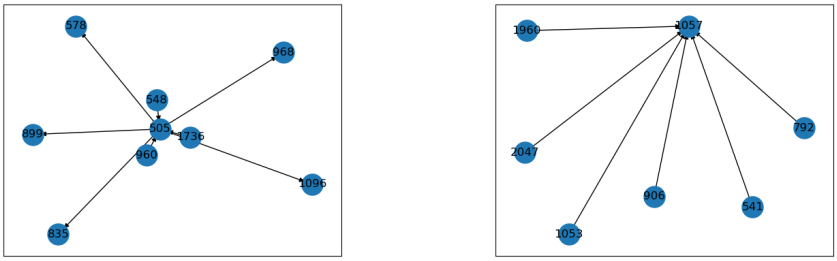
The proof is a copy of the one of the first claim, but it has been separated for clarity. One has to observe that to respect the sequentiality of the process, when $t \in E_u \cap A_j$ is chosen we must have $E_u \cap \cup_{i=1}^{j-1} A_i = \emptyset$.

The Lemma follows from the two claims by case analysis on the repartition of t, t' in H_u and L_u using the second claim to transform it into a case analysis on the indice. □

Proposition 14. *Let S be a DXS with a classification set S_c, and u be a trace. Assume the purely parallel process $P =$* while *true* do $(A_1; \ldots; A_n)$ *is executed on the network, and that P is unambiguous for u. Then for every $t \in A_i, t' \in A_j$ with $i > j$ we have $P^u_{S_c}(t, t') = 1$.*

The other direction cannot be proven as a purely parallel process can be found by coincidence when there are few iterations of the loop.

We present in Fig. 1 examples of the analysis result on a Model 3 CAN Bus log [24] representing 418s of execution and 879682 frames.



(a) The star PPP is the most common one on the Tesla Model 3: Two sets of *id*s have one of the set of nodes which is separated by a single node. This PPP is iterated 4181 times in the log.

(b) A degeneracy of the star PPP is to have one of the set of nodes which is empty. This PPP is also iterated 4181 times in the log.

Figure 1: Examples of purely parallel processes found on a Tesla Model 3 log. In total 29 CAN frames *id* were isolated, 32 were discovered in a request-response process, and 126 involved were in a star process either proper as in Fig. 1a or degenerate as in Fig. 1b.

B.2 Discovering Protocol Instances

Setting.. A *protocol* is a program running among multiple participants in a network and observed through the messages exchanged by these participants. It is specified with a set of *roles* that are programs and the execution of a role is a process called an *actor*. In addition to the control rules defined in the role an actor is playing, the actor has a local memory that is updated while running the protocol. We note that as in [26] under this definition a multi-step attack in which an adversary communicates with different actors on the network is also a protocol.

As in [12] each role is modeled with an *Extended Finite State Machine* (EFSM), a finite state automaton enriched with rules updating the state of an actor playing that role. This automaton is *deterministic* in the sense that an actor playing a role and receiving a message can either reject that message as non-conforming with the protocol specification and leaves its state unchanged, or change its state according to the rules set in the rule in exactly one possible way. Thus to each state of the automaton we associate a couple of functions (g, p) where g returns whether a message received in the current participant's state is acceptable, and p that performs the update of the memory and of the automaton state. A group of actors playing the roles defined in the protocol and communicating one with another is called a *session*.

More formally let X be a finite set of sorted variables, $\mathcal{R}$ be a finite set of

constants (the roles), $\mathcal{Q}$ be a finite state of automaton states, and $\mathcal{E}$ be the set of the possibly observed events (the messages). A *memory state* is a ground substitution of domain X. We denote $\mathcal{M}$ the set of possible memory states. A *state* is a couple $(q, \sigma) \in \mathcal{Q} \times \mathcal{M}$. A *guard* is a function $\mathcal{Q} \times \mathcal{M} \times \mathcal{E} \to \mathcal{B}$. A *post-transition* is a function $\mathcal{Q} \times \mathcal{M} \times \mathcal{E} \to \mathcal{Q} \times \mathcal{M} \times \mathcal{E}$. If p is a transition function and $f(q, m, e)) = (q', m', e')$ then upon receiving the message e, the actor with a state (q, m) updates it to (q', m') and sends the message e'.

An *EFSM* E is a set of tuples $\{(q, g_q, f_q)\}_{q \in \mathcal{Q}_r \subseteq \mathcal{Q}}$ where g_q is a guard and f_q is a post-transition function. A *protocol* P is a finite set of tuples $\{(r, \iota_r, E_r)\}_{r \in \mathcal{R}}$ where each E_r is an EFSM, $\iota_r \in \mathcal{Q}$ is the *initial state* of the role r, and if $r \neq r'$ then $\mathcal{Q}_r \cap \mathcal{Q}_{r'} = \emptyset$. The bin role contains one state $\mathcal{Q}_b = \{\iota_b\}$, and one rule (ι_b, g_b, f_b) where the guard g_b is always true and f_b does nothing.

Algorithm.. Again we build a representation of the system inductively on a trace u. Let $P = \{(r, \iota_r, E_r)\}_{r \in \mathcal{R}}$ be a protocol. Participants are identified uniquely and are assumed to always or never play a role of the protocol. Depending on the network type they can be identified by a socket address or in the case of the CAN bus by the *id* of the frame. Actors are participants playing a role in the protocol. We first build a derived object simulating the network after observing a trace u Our representation of the network is based on an internal state $\mathcal{A}_u$ of potential actors with their state. Accordingly we let:

$$\mathcal{A}(\varepsilon) = \{(\iota_{p,r}, m_{p,r}, E_{p,r})\}_{(p,r) \in \mathcal{P} \times \mathcal{R}}$$

assuming initially every participant may play a role in the protocol.

Inductively the internal state of the function is defined as follows:

- If there exists $(q_{p,r}, m_{p,r}, E_{p,r}) \in \mathcal{A}(u)$, a transition $(q_{p,r}, g_{q_{p,r}}, f_{q_{p,r}}) \in E_{p,r}$, and an event $e' \in u$ such that:

$$\begin{cases} g_{q_{p,r}}(e', m_{p,r}) & = & 1 \\ f_{q_{p,r}}(e', q_{p,r}, m_{p,r}) & = & (e, q'_{p,r}, m'_{p,r}) \end{cases}$$

  Then:

$$\begin{array}{rl} \mathcal{A}(u \cdot e) = (\mathcal{A}(u) & \setminus \{(q_{p,r}, m_{p,r}, E_{p,r})\}) \\ & \cup \left\{(q'_{p,r}, m'_{p,r}, E_{p,r})\right\} \end{array}$$

- Otherwise the participant has sent a message that is not conformant with the protocol specification, and thus per our assumption never is an actor, in any role, of that protocol:

$$\mathcal{A}(u \cdot e) = \mathcal{A}(u) \setminus \{(q_{p,r}, m_{p,r}, E_{p,r}) \in \mathcal{A}(u) \,|\, r \in \mathcal{R}\}$$

The representation of the network after observing the trace u is the subset of participants p such that there exists an EFSM indexed by p in $\mathcal{A}(u)$. We order these sets with $A \sqsubseteq B$ if and only if $B \subseteq A$. The top element is the emptyset and the bottom element is the set of all participants, and the join $A \sqcup B$ of two sets is their intersection. This domain clearly is a representation domain. The computation of $\mathcal{A}(u)$ is extended to worlds with $\mathcal{A}(\Sigma_{u \in U} u) = \bigsqcup_{u \in U} \mathcal{A}(u)$, and is clearly continuous. Though it possible for a participant to play different roles concurrently and be correctly detected, playing concurrently the same role may (as in the case of a server in [12]) make the participant appear as not playing along the rules of the protocols. The benefit of this simplification is the low computational and memory cost in practice.

Our implementation of the Multi-Frame Message protocol [11] has correctly detected on the Tesla Model 3 log the similar CAN-TP [2] protocol according to owners' common findings [7].

Institut de Recherche en Informatique de Toulouse
CNRS - INP - UT3 - UT1 - UT2J

ASR – Architecture, Systems and Networks

RMESS – Networks, Mobile, Embedded, Wireless, Sattelites

SEPIA – Operating systems, distributed systems, from Middleware to Architecture

SIERA – Service IntEgration and netwoRk Administration

T2RS – Real-Time in networks and systems

TRACES – Trace stands for research groups in architecture and compilation for embedded systems

CISO – HPC, Simulation, Optimization

APO – Parallel Algorithms and Optimisation

REVA – Real Expression Artificial Life

FSL – Reliability Systems and Software

ACADIE – Assistance for certification of distributed and embedded applications

ARGOS – Advancing Rigorous Software and System Engineering

ICS – Interactive Critical Systems

SM@RT – Smart Modeling for softw@re Research and Technology

GD – Data Management

IRIS – Information Retrieval and Information Synthesis

PYRAMIDE – Dynamic Query Optimization in large-scale distributed environments

SIG – Generalized information systems

IA – Artificial Intelligence

ADRIA – Argumentation, Decision, Reasoning, Uncertainty and Learning methods

LILaC – Logic, Interaction, Language and Computation

MELODI – Methods and Engineering of Language, Ontology and Discourse

ICI – Interaction, Collective Intelligence

ELIPSE – Human computer interaction

SMAC – Cooperative multi-agents systems

TALENT – Teaching And Learning Enhanced by Technologies

SI – Signals and Images

MINDS – coMputational Imaging anD viSion

SAMoVA – Structuration, Analysis, Modeling of Video and Audio documents

SC – Signal and Communications

STORM – Structural Models and Tools in Computer Graphics

TCI – Images processing and understanding