# On Propositional Definability

Jérôme Lang [a] Pierre Marquis [b]

[a] *IRIT - CNRS / Université Paul Sabatier*
*118 route de Narbonne - 31062 Toulouse - France*
lang@irit.fr

[b] *CRIL - CNRS / Université d'Artois*
*rue Jean Souvraz - S.P. 18 - 62307 Lens - France*
marquis@cril.univ-artois.fr

**Abstract**

In standard propositional logic, logical definability is the ability to derive the truth value of some propositional symbols given a propositional formula and the truth values of some propositional symbols. Although appearing more or less informally in various AI settings, a computation-oriented investigation of the notion is still lacking, and this paper aims at filling the gap. After recalling the two definitions of definability, which are equivalent in standard propositional logic (while based on different intuitions), and defining a number of related notions, we give several characterization results, and many complexity results for definability. We also show close connections with hypothesis discriminability and with reasoning about action and change.

*Key words:* Knowledge Representation, Computational Complexity.

## 1 Introduction

When reasoning about knowledge represented in propositional logic, exhibiting structure can be of a great help. By "structure" we mean some relationships

⋆ This paper is an extended and revised version of some parts of two papers: "*Complexity results for independence and definability in propositional logic*, appeared in the proceedings of the *Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 356–367; and "*Two forms of Dependence in Propositional Logic: Controllability and Definability*", appeared in the proceedings of the *Fifteenth National Conference on Artificial Intelligence (AAAI'98)*, pages 268-273.

which exist between some sets of propositional symbols and/or formulas within a propositional formula $\Sigma$. Such relationships are known under various names, including dependency, relevance, novelty, controllability, and some of them have been investigated, see among others [1,2].

In this paper we focus on an additional form of dependency, called *definability*. Definability captures two different intuitions: *implicit definability* and *explicit definability*. A propositional symbol $y$ can be implicitly defined in a given formula $\Sigma$ in terms of a set $X$ of propositional symbols if and only if the knowledge of the truth values of the propositional symbols of $X$ (whatever they are) enables concluding about the truth value of $y$, while $y$ can be explicitly defined in $\Sigma$ in terms of $X$ when there exists a formula $\Phi_X$ built up from $X$ only, such that $\Phi_X$ is equivalent to $y$ in $\Sigma$.

Definability is acknowledged as an important logical concept for decades. It is closely related to the Craig/Lyndon interpolation theorem [3]. Many studies in logic are about determining whether a given logic (standard or modal, propositional or first-order) satisfies the "basic" Beth property (whenever a theory implicitly defines a symbol in terms of all others, there is an explicit definition of that symbol in terms of all others), or even the (stronger) projective Beth property, (when implicit definability and explicit definability coincide). Thus classical first-order logic satisfies the "basic" Beth property (this is the famous Beth's theorem [4]), while for instance first-order logic on finite structures does not (see e.g., [5]).

Standard propositional logic has been known to satisfy the projective Beth property. In this paper, we consider definability in standard propositional logic from a *computational* point of view. We present several characterization and complexity results which prove useful for several AI applications, including hypothesis discrimination and reasoning about actions and change.

From a computational point of view, our results concern both time and space complexity. As to time complexity, we mainly considered the decision problem DEFINABILITY which consists in determining whether a given formula $\Sigma$ defines a given symbol $y$ (or more generally a given set $Y$ of symbols) in terms of a given set $X$ of symbols. We identified its complexity both in the general case and under restrictions induced by a number of propositional fragments (formally defined in Section 2) that proved of interest in many AI contexts (see [6–9]); the results are summarized in Table 1.

While the table shows the definability problem intractable in the general case (unless $\mathsf{P} = \mathsf{NP}$), it also shows that:

- the main propositional fragments which are tractable for SAT are also tractable for DEFINABILITY. Indeed, DNNF contains (among others) all DNF formulas and all OBDD "formulas", while q-HornCNF contains all renamable

| **Fragment $\mathcal{C}$** | DEFINABILITY |
|---|---|
| $\text{PROP}_{\text{PS}}$ (general case) | coNP-c |
| DNNF | in P |
| q-HornCNF | in P |
| IP | coNP-c |

Table 1
The complexity of DEFINABILITY.

Horn CNF formulas. The fact that large propositional fragments (including complete ones, i.e., fragments into which any propositional formula has an equivalent, as DNNF is) is of great value from a practical perspective.

- nevertheless,tractability for SAT is not enough for ensuring tractability for DEFINABILITY. Thus the Blake fragment IP is tractable for SAT but likely not for DEFINABILITY. We also identified some sufficient conditions (referred to as stability conditions) under which a propositional fragment is tractable for SAT if and only if it is tractable for DEFINABILITY.

About space complexity, we focused on the size of definitions; we showed that in the general case, the size of any explicit definition of a symbol $y$ in terms of a set of symbols $X$ in $\Sigma$ is not polynomially bounded in the input size. We identified some sufficient conditions (polytime conditioning and polytime forgetting) on propositional fragments for ensuring that definitions can be computed in polynomial time (hence are of polynomial size) when such definitions exist. Interestingly, the influential DNNF fragment satisfies them, as well as the Blake fragment IP. The result for IP shows that it can be the case that computing an explicit definition of $y$ on $X$ in $\Sigma$ is easy when one knows that such a definition exists, while deciding whether it exists is hard.

The rest of the paper is organized as follows. In Section 2, we give some necessary background about propositional logic and computational complexity. In Section 3 the notion of definability is presented, as well as a number of related notions (including the notions of minimal defining family (or base), undefinable symbol, necessary symbol and relevant symbol, as well as the notion of unambiguous definability). We also show how such notions relate one another and are connected to previous concepts, especially variable forgetting (see [10,2]) as well as the notions of weakest sufficient and strongest necessary conditions [11]. In Section 4, we give a number of complexity results for definability and the related notions. We identify a number of tractable restrictions of the decision problems under consideration. We also report some complexity results about the size of explicit definitions and present an algorithm for computing a base. In Section 5, we show that definability is closely related to hypothesis discriminability. In Section 6, we explain how many important issues in reasoning about action and change can be characterized in terms of

definability. In Section 7, we briefly sketch how definability can prove useful to automated reasoning. In Section 8, we relate our results to the literature. Finally, Section 9 concludes the paper.

## 2 Formal Preliminaries

### *2.1 Propositional logic*

Let PS be a finite set of propositional propositional symbols (also called variables). $\mathrm{PROP}_{\mathrm{PS}}$ is the propositional language built up from PS, the connectives $\neg$, $\vee$, $\wedge$, $\Rightarrow$, $\Leftrightarrow$ and the Boolean constants true and false in the usual way. Subsets of PS are denoted $X$, $Y$, etc. For every $X \subseteq \mathrm{PS}$, $\mathrm{PROP}_X$ denotes the sublanguage of $\mathrm{PROP}_{\mathrm{PS}}$ generated from the propositional symbols of $X$ only.

From now on, $\Sigma$ denotes a finite set of propositional formulas from $\mathrm{PROP}_{\mathrm{PS}}$. $\mathrm{Var}(\Sigma)$ is the set of propositional symbols appearing in $\Sigma$ and $|\Sigma|$ is the size of $\Sigma$, i.e., the number of symbols used to write it. Elements of PS are denoted $x$, $y$, etc. Specific formulas from $\mathrm{PROP}_{\mathrm{PS}}$ are of interest: a literal is a symbol $x$ of PS (positive literal) or a negated one $\neg x$ (negative literal). $x$ and $\neg x$ are two complementary literals. A clause (resp. term) is a disjunction (resp. conjunction) of literals, or the constant false (resp. true). A Conjunctive Normal Form formula (for short, a CNF formula) is a conjunction of clauses. A Disjunctive Normal Form formula (for short, a DNF formula) is a disjunction of terms. A CNF formula is Krom [12] if and only if each clause in it contains at most two literals. A Krom formula is also said to be a 2-CNF formula or a quadratic formula. A CNF formula is Horn [13] if and only if each clause in it contains at most one positive literal. A CNF formula $\Sigma$ is renamable Horn [14] if and only if there exists a Horn renaming for it, i.e., a set $V$ of symbols $v$ such that replacing every occurrence of $v \in V$ (resp. $\neg v$) in $\Sigma$ by the complementary literal $\neg v$ (resp. $v$) leads to a Horn CNF formula. A CNF formula $\Sigma$ has a QH-partition [6] if and only if there exists a partition $\{Q, H\}$ of $\mathrm{Var}(\Sigma)$ s.t. for every clause $\delta$ of $\Sigma$, the following conditions hold:

- $\delta$ contains no more than two variables from $Q$;
- $\delta$ contains at most one positive literal from $H$;
- if $\delta$ contains a positive literal from $H$, then it contains no variable from $Q$.

A CNF formula $\Sigma$ is q-Horn [6] if and only if there exists a q-Horn renaming for it, i.e., a set $V$ of symbols $v$ such that replacing in $\Sigma$ every occurrence of a positive literal $v$ (resp. a negative literal $\neg v$) by the complementary literal $\neg v$ (resp. $v$) leads to a CNF formula having a QH-partition $\{Q, H\}$. The propo-

sitional fragment q-HornCNF is the set all q-Horn formulas from $\mathrm{PROP}_{\mathrm{PS}}$; it includes both the Krom formulas ($H = \emptyset$) and the renamable Horn CNF formulas ($Q = \emptyset$) as proper subsets.

A Negation Normal Form formula (for short, an NNF formula) is any formula $\Sigma$ built up from PS, the connectives $\neg$, $\vee$, $\wedge$ and the Boolean constants true and false, such that the scope of any occurrence of $\neg$ in $\Sigma$ is a symbol or a Boolean constant. Thus, every CNF (resp. every DNF) formula also is an NNF formula. An NNF formula $\Sigma$ is decomposable (i.e., a DNNF formula) [7,9] if and only if every subformula in $\Sigma$ of the form $\varphi \wedge \psi$ is such that $\mathrm{Var}(\varphi) \cap \mathrm{Var}(\psi) = \emptyset$. Obviously, every DNF formula also is a DNNF formula, but the converse does not hold. DNNF is the propositional fragment containing all DNNF formulas from $\mathrm{PROP}_{\mathrm{PS}}$.

Formulas from $\mathrm{PROP}_{\mathrm{PS}}$ are interpreted in the standard, usual way. Full instantiations of propositional symbols of PS on $\mathrm{BOOL} = \{0, 1\}$ (worlds) are denoted by $\vec{\omega}$ and their set is denoted by $\Omega$. Any world satisfying a given formula $\varphi$ is said to be a model of $\varphi$. Full instantiations of propositional symbols of $X \subseteq \mathrm{PS}$ are denoted by $\vec{x}$ and called $X$-worlds; their set is denoted by $\Omega_X$. We shall identify $\vec{x}$ with the corresponding canonical conjunction of literals over $X$ in order to simplify the notations; for instance, if $X = \{a, b\}$ and $\vec{x} = (a = 1, b = 0)$ then we also write $\vec{x} = a \wedge \neg b$. We shall also identify any finite set of formulas with the conjunction of all formulas from the set. $\models$ denotes logical entailment and $\equiv$ denotes logical equivalence. If $\Sigma, \Phi, \Psi \in \mathrm{PROP}_{\mathrm{PS}}$, $\Phi$ and $\Psi$ are said to be $\Sigma$-equivalent if and only if $\Sigma \models \Phi \Leftrightarrow \Psi$.

Assuming that worlds are represented by the subset of all variables they satisfy (i.e., $\vec{\omega}$ is given by $\{x \in \mathrm{PS} \mid \vec{\omega}(x) = 1\}$), the Horn envelope of a Horn CNF formula $\Sigma$ is the smallest set of models of $\Sigma$ (over $\mathrm{Var}(\Sigma)$) whose intersection closure [1] is the whole set of models of $\Sigma$. A q-Horn envelope of a q-Horn CNF formula $\Sigma$ which has a QH-partition is any smallest set of models of $\Sigma$ (over $\mathrm{Var}(\Sigma)$) whose QH-convolution closure is the whole set of models of $\Sigma$ (see [15] for details).

In order to avoid heavy notations, we sometimes abuse notations and write $x$ instead of $\{x\}$. For every formula $\Phi \in \mathrm{PROP}_{\mathrm{PS}}$ and every propositional symbol $x \in \mathrm{PS}$, $\Phi_{x \leftarrow 0}$ (resp. $\Phi_{x \leftarrow 1}$) is the formula obtained by replacing in $\Phi$ every occurence of $x$ by the constant false (resp. true). More generally, if $\gamma$ is a satisfiable conjunction of literals then the conditioning $\Sigma_\gamma$ of $\Sigma$ by $\gamma$ is the formula obtained by replacing in $\Sigma$ every occurrence of each positive literal $x$ of $\gamma$ by true and every occurrence of each negative literal $\neg x$ of $\gamma$ by false.

An implicate (resp. implicant) of a formula $\Sigma$ is a clause $\delta$ (resp. a conjunction

[1] The intersection closure $C$ of a set $S$ is the smallest set w.r.t. $\subseteq$ such that $S \subseteq C$ and $\forall e_1, e_2 \in C, e_1 \cap e_2 \in C$.

of literals $\gamma$) which is a logical consequence of $\Sigma$ (resp. such that $\Sigma$ is a logical consequence of $\gamma$). A prime implicate (resp. prime implicant) of $\Sigma$ is one of its logically strongest implicates (resp. one of its logically weakest implicants). A formula $\Sigma$ is in prime implicates normal form (or a Blake formula or a prime formula) [16] if and only if it is a CNF formula whose clauses are the prime implicates of $\Sigma$ (one representative per equivalence class, only). IP is the propositional fragment containing all Blake formulas.

**Example 1**

- $(a \vee b) \wedge (a \vee (\neg b \wedge c))$ *is an NNF formula but neither a DNNF formula nor a CNF formula.*
- $(a \vee b) \wedge (c \vee (\neg c \wedge d))$ *is a DNNF formula but neither a DNF formula nor a CNF formula.*
- $(a \wedge b) \vee (\neg a \wedge d)$ *is a DNF formula.*
- $(a \vee b \vee c) \wedge (\neg a \vee \neg b \vee \neg c) \wedge (\neg a \vee d)$ *is a CNF formula but neither a DNNF one nor a q-Horn CNF one nor a Blake one.*
- $(a \vee b \vee c) \wedge (\neg a \vee \neg b \vee \neg c)$ *is a Blake formula but neither a DNNF one nor a q-Horn CNF one.*
- $(\neg a \vee \neg b \vee c) \wedge (a \vee \neg b \vee \neg c) \wedge (\neg a \vee b \vee \neg c) \wedge (\neg a \vee \neg d \vee \neg e) \wedge (\neg b \vee \neg d \vee e) \wedge (\neg c \vee d \vee \neg e) \wedge (d \vee e \vee \neg f)$ *is a q-Horn CNF formula but neither a DNNF one nor a Blake one nor a renamable Horn CNF one nor a Krom one.*
- $(a \vee b) \wedge (\neg a \vee \neg c \vee d)$ *is a renamable Horn CNF formula but neither a DNNF one nor a Horn CNF one nor a Krom one nor a Blake one.*
- $(a \vee \neg b) \wedge (b \vee \neg c \vee \neg d)$ *is a Horn CNF formula but neither a DNNF one nor a Krom one nor a Blake one.*
- $(a \vee b) \wedge (\neg b \vee c)$ *is a Krom formula but neither a DNNF one nor a Horn CNF one nor a Blake one.*

For each of the propositional fragments listed in this section, the recognition problem is tractable (i.e., there exists a (deterministic) polynomial time algorithm for determining whether any given propositional formula belongs to the fragment). This is obvious for most of those fragments, except qHornCNF (and its subset consisting of all renamable Horn CNF formulas) and to a lesser extent, IP. For qHornCNF, see [6,17]; for IP, this comes from the correctness of any resolution-based prime implicates algorithm (like Tison's one [18]): a CNF formula $\Sigma$ is Blake if and only if whenever two clauses of it have a resolvent, there exists a clause in $\Sigma$ which implies it, and no clause of $\Sigma$ is implied by another clause of $\Sigma$.

Unlike $\mathrm{PROP}_{\mathrm{PS}}$ and some of its subsets (as the set of all CNF formulas), qHornCNF, DNNF and IP are known as tractable for the satisfiability problem SAT; this means that for each of these fragments, there exists a (deterministic) polynomial time algorithm for determining whether any given formula from the fragment is satisfiable. For instance, in order to determine whether a Blake

formula is satisfiable, it is enough to check that it does not reduce to false (the empty clause) (this is a direct consequence of the definition of a Blake formula). For the qHornCNF and DNNF fragments, see respectively [6] and [7,9].

### *2.2 Computational complexity*

We assume that the reader is familiar with some basic notions of computational complexity, especially the complexity classes P, NP, and coNP, as well as the basic decision problems SAT and UNSAT (and their restrictions to CNF formulas, noted CNF-SAT and CNF-UNSAT) and the classes $\Delta_k^p$, $\Sigma_k^p$ and $\Pi_k^p$ of the polynomial hierarchy $\mathsf{PH} = \bigcup_{k\geq 0} \Delta_k^p = \bigcup_{k\geq 0} \Sigma_k^p = \bigcup_{k\geq 0} \Pi_k^p$ (see [19] for details).

Let us recall that a decision problem is said to be at the $k^{th}$ level of PH if and only if if it belongs to $\Delta_{k+1}^p$, and is either $\Sigma_k^p$-hard or $\Pi_k^p$-hard.

It is well-known that if there exists $i > 0$ such that $\Sigma_i^p = \Pi_i^p$ then for every $j > i$, we have $\Sigma_j^p = \Pi_j^p = \Sigma_i^p$: PH is said to collapse to level $i$. It is strongly believed that PH does not collapse (to any level), i.e., it is a truly infinite hierarchy (for every integer $k$, $\mathsf{PH} \neq \Sigma_k^p$).

$\mathsf{BH}_2$ (also known as DP) is the class of all languages $L$ such that $L = L_1 \cap L_2$, for some $L_1$ in NP and $L_2$ in coNP.

The canonical $\mathsf{BH}_2$-complete problem is SAT-UNSAT: a pair of formulas $\langle \varphi, \psi \rangle$ is in SAT-UNSAT if and only if $\varphi$ is satisfiable and $\psi$ is not. This class belongs to the *Boolean hierarchy*; unless NP = coNP, $\mathsf{BH}_2$ strictly contains both NP and coNP.

An advice-taking Turing machine is a Turing machine that has associated with it a special "advice oracle" $A$, which can be any function (not necessarily a recursive one). On input $s$, a special "advice tape" is automatically loaded with $A(|s|)$ and from then on the computation proceeds as normal, based on the two inputs, $s$ and $A(|s|)$.

An advice-taking Turing machine uses polynomial advice if its advice oracle $A$ satisfies $|A(n)| \leq p(n)$ for some fixed polynomial $p$ and all non-negative integers $n$; finally, P/poly is the class of all languages which can be decided in polynomial time by deterministic Turing machines augmented by polynomial advice. It is believed that NP $\cap$ coNP is not included in P/poly.

## 3 Definability: Definitions, Properties and Characterizations

### *3.1 Implicit and explicit definability*

*Definability* is a strong form of dependence: while dependent propositional symbols interact *in some situations*, definability imposes that some propositional symbols are fixed *whenever* some other propositional symbols are fixed as well.

**Definition 2 ((implicit) definability)** *Let $\Sigma \in PROP_{PS}$, $X, Y \subseteq PS$ and $y \in PS$.*

- $\Sigma$ **defines** *$y$ in terms of $X$ (denoted by $X \sqsubseteq_\Sigma y$) if and only if $\forall \vec{x} \in \Omega_X, \vec{x} \wedge \Sigma \models y$ or $\vec{x} \wedge \Sigma \models \neg y$.*
- *$X \sqsubseteq_\Sigma Y$ if and only if $X \sqsubseteq_\Sigma y$ for every $y \in Y$.*

Note that requiring $\vec{x} \wedge \Sigma$ to be satisfiable would be useless since $\vec{x} \wedge \Sigma \models y$ holds whenever $\vec{x} \wedge \Sigma$ is unsatisfiable. When no $X$-world consistent with $\Sigma$ can be found, $\Sigma$ is unsatisfiable. In this case, definability trivializes, i.e., $X \sqsubseteq_\Sigma y$ holds for every $X$ and $y$.

**Example 3** *Let $l$ stand for "leap year", and $d4$ (resp. $d25$, $d100$, $d400$) for "divisible by 4" (resp. by 25, 100, 400). Let $\Sigma = \{d400 \Rightarrow l, (d100 \wedge \neg d400) \Rightarrow \neg l, (d4 \wedge \neg d100) \Rightarrow l, \neg d4 \Rightarrow \neg l, d100 \Leftrightarrow (d4 \wedge d25), d400 \Rightarrow d100\}$ a set of formulas making precise some connections between those symbols.*
*We have $\{d4, d25\} \sqsubseteq_\Sigma d100$; $\{d4, d100, d400\} \sqsubseteq_\Sigma l$; $\{d4, d25, d400\} \sqsubseteq_\Sigma l$;*
*$\Sigma$ does not define $l$ in terms of $\{d25, d100, d400\}$, because the joint falsity of these three propositional symbols does not enable telling whether $l$ is true or false, since we do not know whether $d4$ holds or not.*
*Other definability relations hold; in particular, $\{l, d100, d400\} \sqsubseteq_\Sigma d4$;*
*$\{l, d100\} \sqsubseteq_\Sigma d4$; $\{l, d100\} \sqsubseteq_\Sigma \{d4, d100, d400\}$.*

When $X \sqsubseteq_\Sigma y$ holds, one can state equivalently that the *functional dependency* $X \rightarrow y$ holds in $\Sigma$. This notion of functional dependency is the well-known one from the relational database theory restricted to binary domains (see [20,21]).

Definability satisfies the following easy properties (which we give without proofs):

(1) $\sqsubseteq_\Sigma$ is transitive.
(2) If $X' \subseteq X$, then $X \sqsubseteq_\Sigma X'$. In particular, $\sqsubseteq_\Sigma$ is reflexive.
(3) If $X \sqsubseteq_\Sigma Y$ and $X \sqsubseteq_\Sigma Y'$, then $X \sqsubseteq_\Sigma Y \cup Y'$.
(4) If $X \sqsubseteq_\Sigma Y$ and $\Sigma' \models \Sigma$, then $X \sqsubseteq_{\Sigma'} Y$.
(5) If $X \sqsubseteq_\Sigma Y$ and $X' \sqsubseteq_\Sigma Y'$, then $X \cup X' \sqsubseteq_\Sigma Y \cup Y'$.

(1), (2) and (3) correspond to the famous Armstrong's rules of inference (and known respectively as the transitivity rule, the inclusion rule and the augmentation rule) [20]. (4) is a monotonicity property; (5) is a derived rule of inference in Armstrong's system (and is known as the addition rule or the composition rule).

It is also easy to show that if $\Sigma$ is satisfiable and $y \notin \mathrm{Var}(\Sigma) \cup X$, then $X \not\sqsubseteq_\Sigma y$. Similarly, if $\Sigma$ is valid then $X \sqsubseteq_\Sigma Y$ holds if and only if $Y \subseteq X$.[2] Other properties that can be shown when $\{x\} \sqsubseteq_\Sigma Y$ are reported in Lemma 2.3 from [22].

Now, another notion of definability can be easily defined, relating a set of propositional symbols $X$ to a propositional symbol $y$ given a formula $\Sigma$; it requires the existence of an explicit definition of $y$ in $\Sigma$ using propositional symbols of $X$, only. While the previous form of definability is typically referred to as *implicit* definability, the latter one is called *explicit definability*.

**Definition 4 (explicit definability; definition of a propositional symbol)** *Let $\Sigma \in PROP_{PS}$, $X \subseteq PS$ and $y \in PS$. $\Sigma$* **explicitly defines** *$y$ in terms of $X$ if and only if there exists a formula $\Phi_X \in PROP_X$ s.t. $\Sigma \models \Phi_X \Leftrightarrow y$. In such a case, $\Phi_X$ is called a* definition of $y$ on $X$ in $\Sigma$.

As a corollary of Craig's interpolation theorem [3] (stated in the more general framework of first-order logic), the equivalence between the *implicit* form of definability (as given above) and the *explicit* form can be stated. This result is known as the projective Beth's theorem in propositional logic. We give a proof for this basic result since it enables for pointing out a first, simple (explicit) definition.

**Theorem 5 (propositional projective Beth's theorem)** *Let $\Sigma \in PROP_{PS}$, $X \subseteq PS$ and $y \in PS$. $\Sigma$ explicitly defines $y$ in terms of $X$ if and only if $X \sqsubseteq_\Sigma y$.*

**Proof:** The ($\Rightarrow$) direction is obvious. As to the ($\Leftarrow$) direction, suppose $\Sigma$ implicitly defines $y$ in terms of $X$. For each world $\vec{x}$ satisfying $\Sigma$, let $\varphi_{\vec{x}}$ be the conjunction of all literals over $X$ true in $\vec{x}$. Since the truth value of $y$ in a world satisfying $\Sigma$ depends only on the truth value of $X$, we have that $\Sigma \wedge \varphi_{\vec{x}} \models y$. It follows that the disjunction $\Phi$ of all $\varphi_{\vec{x}}$'s for $\vec{x}$ a world satisfying $\Sigma \Rightarrow y$ is an explicit definition of $y$ on $X$. Similarly, the negation of the disjunction $\Psi$ of all $\varphi_{\vec{x}}$'s for $\vec{x}$ a world satisfying $\Sigma \Rightarrow \neg y$ is an explicit definition of $y$ on $X$ (indeed, we have $\Sigma \models \Phi \Leftrightarrow \neg\Psi$). □

In Lemma 5.1 from [15], one can find representations, based on the prime implicates of $\Sigma$, of the two explicit definitions $\Phi$ and $\neg\Psi$ given in the proof

[2] This shows the system of rules above complete when $\Sigma$ is valid since every definability relation is an instance of axiom schema (3).

of Theorem 5. The first one is noted $f_{(X,y)}$ and the second one is noted $\bar{f}_{(X,\bar{y})}$. Clearly enough, such representations are not always the more succinct one from the spatial efficiency point of view (both of them can be exponential in the size of $\Sigma$), and since any formula $\Sigma$-equivalent to $\Phi$ (resp. $\Psi$) is an explicit definition of $y$ on $X$ in $\Sigma$, there is no specific need to focus on prime implicates representations.

**Example 3 (continued)** *The following explicit definitions hold:*
$\Sigma \models (d4 \wedge (\neg d100 \vee d400)) \Leftrightarrow l$;
$\Sigma \models (d4 \wedge (\neg d25 \vee d400)) \Leftrightarrow l$;
$\Sigma \models (l \vee d100) \Leftrightarrow d4$;
$\Sigma \models (d100 \wedge l) \Leftrightarrow d400$;
$\Sigma \models ((l \wedge \neg d25) \vee d100) \Leftrightarrow d4$.

Theorem 5 shows that $\Sigma$ defines $y$ in terms of $X$ if and only if there exists a definition $\Phi_X$ of $y$ in $\Sigma$ such that $X = \mathrm{Var}(\Phi_X)$. Now, what about the *unicity* of the definition of $y$ on $X$ in $\Sigma$, when $\Sigma$ defines $y$ in terms of $X$? As suggested in the proof of Theorem 5, there are several possible definitions of $y$ on $X$ in $\Sigma$, which are generally not logically equivalent, but which are nevertheless $\Sigma$-equivalent: if $\Phi$ and $\Psi$ are both definitions of $y$ in $\Sigma$, we have $\Sigma \models \Phi \Leftrightarrow \Psi$, and additionally, $\Phi \vee \Psi$ and $\Phi \wedge \Psi$ are also definitions of $y$ in $\Sigma$; thus, the set of all definitions of $y$ on $X$ in $\Sigma$, quotiented by logical equivalence, is a finite lattice. The least (resp. uppest) element of this lattice is called the *strongest* (resp. *weakest*) definition of $y$ in $\Sigma$, and is denoted by $\mathrm{Def}_\Sigma^{X,l}(y)$ (resp. $\mathrm{Def}_\Sigma^{X,u}(y)$). Note that $\mathrm{Def}_\Sigma^{X,l}(y)$ and $\mathrm{Def}_\Sigma^{X,u}(y)$ are defined only when $X \sqsubseteq_\Sigma y$ holds.

Now, the previous notion of definability of a propositional symbol can be easily turned into a more general notion of formula definability. Formally:

**Definition 6 (formula definability)** *Let $\Sigma, \Psi \in PROP_{PS}$ and $X \subseteq PS$. $\Sigma$* **defines** *$\Psi$ in terms of $X$ (noted $X \sqsubseteq_\Sigma \Psi$) if and only if $\forall \vec{x} \in \Omega_X, \vec{x} \wedge \Sigma \models \Psi$ or $\vec{x} \wedge \Sigma \models \neg\Psi$.*

While formula definability extends propositional symbol definability (since every propositional symbol $y$ can be also viewed as the formula $y$), it can be recovered from it easily:

**Lemma 7** *Let $\Sigma, \Psi \in PROP_{PS}$ and $X \subseteq PS$. Let $z$ be a (fresh) propositional symbol of $PS \setminus (X \cup Var(\Sigma) \cup Var(\Psi))$. $X \sqsubseteq_\Sigma \Psi$ if and only if $X \sqsubseteq_{\Sigma \wedge (\Psi \Leftrightarrow z)} z$.*

**Proof:** The proof comes straightforwardly from the following equivalence: for any $\vec{x}$, $\vec{x} \wedge \Sigma \models \Psi$ or $\vec{x} \wedge \Sigma \models \neg\Psi$ is equivalent to $\vec{x} \wedge \Sigma \wedge (\Psi \Leftrightarrow z) \models z$ or $\vec{x} \wedge \Sigma \wedge (\Psi \Leftrightarrow z) \models \neg z$. □

Thus, there is no gap of generality between propositional symbol definability and formula definability; also, in the rest of the paper, for the sake of simplicity

we restrict to propositional symbol definability without any loss of generality.

### *3.2 Characterizations of definability*

The proof of Theorem 5 gives a first, semantical, expression of a definition of $y$ on $X$ in $\Sigma$ (when it makes sense, i.e., when $X \sqsubseteq_\Sigma y$ holds), namely, any formula from $\mathrm{PROP}_X$ whose set of models is $\{\vec{x} | \vec{x} \wedge \Sigma \models y\}$. The next results aim at giving more syntactical characterizations, which will provide us with some pratical ways of computing definitions.

Before presenting them, we need to recall a few basic notions and results about independence and forgetting (see [2] for more details). Let $X$ be a subset of PS. A formula $\Sigma \in \mathrm{PROP}_{\mathrm{PS}}$ is *independent of* $X$ if and only if there exists a formula $\Phi$ s.t. $\Phi \equiv \Sigma$ holds and $\mathrm{Var}(\Phi) \cap X = \emptyset$. When $X = \{x\}$, we say that $\Sigma$ is independent of $x$. It can be easily shown ([2]) that $\Sigma$ is independent of $X$ if and only if $\Sigma$ is independent of each propositional symbol of $X$. The set of propositional symbols on which a formula $\Sigma$ depends is denoted by $\mathrm{DepVar}(\Sigma)$. For instance, if $\Sigma = a \wedge (b \vee \neg b)$ then $\mathrm{DepVar}(\Sigma) = \{a\}$.

Let $\Sigma \in \mathrm{PROP}_{\mathrm{PS}}$ and $X \subseteq \mathrm{PS}$. The *forgetting* of $X$ in $\Sigma$, denoted $\exists X.\Sigma$, is the formula from $\mathrm{PROP}_{\mathrm{PS}}$ inductively defined as follows [10]:

- $\exists \emptyset.\Sigma = \Sigma$,
- $\exists \{x\}.\Sigma = \Sigma_{x \leftarrow 1} \vee \Sigma_{x \leftarrow 0}$,
- $\exists \{x\} \cup Y.\Sigma = \exists Y.(\exists \{x\}.\Sigma)$.

For instance, with $\Sigma = (\neg a \vee b) \wedge (a \vee c)$, we have $\exists \{a\}.\Sigma \equiv b \vee c$.

Clearly enough, $\exists X.\Sigma$ corresponds to a quantified Boolean formula, usually with free variables ($\exists$ is second-order quantification, i.e., it bears on propositional atoms).

It can be shown [2] that $\exists X.\Sigma$ is the logically strongest consequence of $\Sigma$ that is independent of $X$ (up to logical equivalence). Thus, if $\varphi$ is independent of $X$, then $\Sigma \models \varphi$ if and only if $\exists X.\Sigma \models \varphi$. Accordingly, $\Sigma$ is independent of $X$ if and only if $\Sigma \equiv \exists X.\Sigma$ holds.

Now, the *projection* of a formula $\Sigma$ on a set of propositional symbols $X$ is the result of forgetting everything in $\Sigma$ except $X$:

$$\mathrm{Proj}(\Sigma, X) = \exists (\mathrm{Var}(\Sigma) \setminus X).\Sigma.$$

Taking advantage of the notion of projection, the following result gives a characterization of the definitions of a propositional symbol $y$ definable in

terms of a set of propositional symbols $X$ in a formula $\Sigma$.

**Theorem 8** *Let $\Sigma \in PROP_{PS}$ and $X \subseteq PS$. Let $\Phi_X \in PROP_X$ and $y \in PS$. $\Phi_X$ is a definition of $y$ on $X$ in $\Sigma$ if and only if*

$$Proj(\Sigma \wedge y, X) \models \Phi_X \models \neg Proj(\Sigma \wedge \neg y, X).$$

**Proof:** We have $\Sigma \models \Phi_X \Leftrightarrow y$ if and only if $\Sigma \models \Phi_X \Leftarrow y$ and $\Sigma \models \Phi_X \Rightarrow y$ if and only if $\Sigma \wedge \neg y \models \neg\Phi_X$ and $\Sigma \wedge y \models \Phi_X$ if and only if $\exists(PS \setminus X).(\Sigma \wedge \neg y) \models \neg\Phi_X$ and $\exists(PS \setminus X).(\Sigma \wedge y) \models \Phi_X$ (since $\Phi_X$ is independent of $PS \setminus X$) if and only if $\mathrm{Proj}(\Sigma \wedge y, X) \models \Phi_X \models \neg\mathrm{Proj}(\Sigma \wedge \neg y, X)$. □

As a direct corollary, we obtain the following characterizations of the strongest and weakest definitions of $y$, as well as a further characterization of definability:

**Corollary 9** *Let $\Sigma \in PROP_{PS}$, $X \subseteq PS$ and $y \in PS$.*

- *If $X \sqsubseteq_\Sigma y$ then $Def_\Sigma^{X,l}(y) \equiv Proj(\Sigma \wedge y, X)$.*
- *If $X \sqsubseteq_\Sigma y$ then $Def_\Sigma^{X,u}(y) \equiv \neg Proj(\Sigma \wedge \neg y, X)$.*
- *$X \sqsubseteq_\Sigma y$ if and only if $Proj(\Sigma \wedge y, X) \models \neg Proj(\Sigma \wedge \neg y, X)$.*

**Example 3 (continued)** *Here are the weakest and the strongest definitions (up to logical equivalence) of $d4$ on $\{l, d25, d100\}$ in $\Sigma$:*

- $Def_\Sigma^{\{l,d25,d100\},l}(d4) \equiv (l \vee d100)$.
- $Def_\Sigma^{\{l,d25,d100\},u}(d4) \equiv (d25 \wedge d100) \vee (l \wedge \neg d25 \vee \neg d100)$.

Theorem 8 shows that definability is related to the notions of weakest sufficient condition and strongest necessary condition from [11]. Indeed, let $X \subseteq \mathrm{PS}$ and $y \in \mathrm{PS}$. A formula $\Phi$ of $\mathrm{PROP}_X$ is a *strongest necessary condition (SNC)* of $y$ on $X$ given $\Sigma$ if $\Sigma \models y \Rightarrow \Phi$ holds (i.e., $\Phi$ is a necessary condition (NC) of $y$ on $X$ given $\Sigma$), and for any formula $\Psi$ of $\mathrm{PROP}_X$, if $\Sigma \models y \Rightarrow \Psi$ holds, then $\Sigma \models \Phi \Rightarrow \Psi$ holds. $\Phi \in \mathrm{PROP}_X$ is a *weakest sufficient condition (WSC)* of $y$ on $X$ given $\Sigma$ if $\Sigma \models \Phi \Rightarrow y$ holds (i.e., $\Phi$ is a sufficient condition (SC) of $y$ on $X$ given $\Sigma$), and for any formula $\Psi$ of $\mathrm{PROP}_X$, if $\Sigma \models \Psi \Rightarrow y$ holds, then $\Sigma \models \Psi \Rightarrow \Phi$ holds. Note that both the strongest necessary and the weakest sufficient conditions of $y$ on $X$ are unique up to $\Sigma$-equivalence [11] (but not up to logical equivalence in the general case).

The following theorem shows how SNC and WSC can be characterized using the notion of projection. It extends Theorem 2 from [11] by relaxing the assumption that $y \in \mathrm{Var}(\Sigma)$ and $y \notin X$, and focus on the logically strongest (resp. weakest) SNC (resp. WSC) of $y$ on $X$ w.r.t. $\Sigma$, *up to logical equivalence*:

**Theorem 10** *Let $\Sigma \in PROP_{PS}$, $X \subseteq PS$ and $y \in PS$.*

- *Proj*$(\Sigma \wedge y, X)$ *is (up to logical equivalence) the logically strongest SNC of* $y$ *on* $X$ *given* $\Sigma$.
- $\neg$*Proj*$(\Sigma \wedge \neg y, X)$ *is (up to logical equivalence) the logically weakest WSC of* $y$ *on* $X$ *given* $\Sigma$.

**Proof:** We just prove the first point (the second one is similar by duality between SNC and WSC). Let $\Phi_X$ be an SNC of $y$ on $X$ given $\Sigma$. By definition, we have $\Sigma \models y \Rightarrow \Phi_X$. This is equivalent to $\Sigma \wedge y \models \Phi_X$, and equivalent again to $\mathrm{Proj}(\Sigma \wedge y, X) \models \Phi_X$ since $\mathrm{Var}(\Phi_X) \subseteq X$. Hence every SNC of $y$ on $X$ given $\Sigma$ is a logical consequence of $\mathrm{Proj}(\Sigma \wedge y, X)$. It remains to show that $\mathrm{Proj}(\Sigma \wedge y, X)$ is an NC of $y$ on $X$ given $\Sigma$, which is easy since by definition of forgetting, $\Sigma \wedge y \models \mathrm{Proj}(\Sigma \wedge y, X)$ for any $X$ and $\mathrm{Proj}(\Sigma \wedge y, X)$ is independent of every symbol which does not belong to $X$. □

From this theorem, one can show that Theorem 8 generalizes Proposition 2 from [11] by providing not only a characterization of definability in terms of SNC and WSC, but also a characterization of all the *definitions* of $y$ on $X$ w.r.t. $\Sigma$ in terms of SNC and WSC.

Finally, the following lemma shows that, when checking whether $X \sqsubseteq_\Sigma Y$, every propositional symbol can be forgotten from $\Sigma$ except the *definiens* $X$ and the *definiendum* $Y$:

**Lemma 11** *Let* $\Sigma \in PROP_{PS}$ *and* $X, Y \subseteq PS$. $X \sqsubseteq_\Sigma Y$ *if and only if* $X \sqsubseteq_{Proj(\Sigma, X \cup Y)} Y$.

**Proof:**

($\Rightarrow$) Let $y \in Y$. We have $X \sqsubseteq_\Sigma y$ if and only if there exists a formula $\Psi$ s.t. $\mathrm{Var}(\Psi) \subseteq X$ and $\Sigma \models (\Psi \Leftrightarrow y)$. Clearly enough, $(\Psi \Leftrightarrow y)$ is independent of every propositional symbol which does not occur in $X \cup \{y\}$. Especially, $(\Psi \Leftrightarrow y)$ is independent of $\mathrm{Var}(\Sigma) \setminus (X \cup \{y\})$. Since $\mathrm{Proj}(\Sigma, X \cup \{y\}) = \exists(\mathrm{Var}(\Sigma) \setminus (X \cup \{y\})).\Sigma$ is the most general consequence of $\Sigma$ that is independent of $\mathrm{Var}(\Sigma) \setminus (X \cup \{y\})$, we have $\Sigma \models (\Psi \Leftrightarrow y)$ if and only if $\mathrm{Proj}(\Sigma, X \cup \{y\}) \models (\Psi \Leftrightarrow y)$. Hence, $X \sqsubseteq_{\mathrm{Proj}(\Sigma, (X \cup \{y\})} \{y\}$. This is true for any $y \in Y$, hence we have $X \sqsubseteq_{\mathrm{Proj}(\Sigma, X \cup Y)} Y$.

($\Leftarrow$) As explained in Section 3.1, $\sqsubseteq_\Sigma$ is monotonic in $\Sigma$ in the sense that, for every $X, Y, \Sigma, \Sigma'$, if $X \sqsubseteq_\Sigma Y$ and $\Sigma' \models \Sigma$, then $X \sqsubseteq_{\Sigma'} Y$. The fact that $\exists(\mathrm{Var}(\Sigma) \setminus (X \cup Y)).\Sigma$ is a logical consequence of $\Sigma$ completes the proof.

□

A practical interest of this lemma lies in the fact that $\mathrm{Proj}(\Sigma, X \cup Y)$ may belong to a fragment which is computationally easier than $\Sigma$ for the definability issues. For instance, consider $\Sigma = (a \vee (\neg b \wedge c)) \wedge (a \wedge (\neg a \vee d))$, $X = \{b, c\}$ and $Y = \{d\}$. While $\Sigma$ belongs to the NNF fragment for which DEFINABILITY is

not tractable (unless $\mathsf{P} = \mathsf{NP}$) (see Theorem 22), $\mathrm{Proj}(\Sigma, X \cup Y) = \exists\{a\}.\Sigma$ belongs to the DNNF fragment for which DEFINABILITY is tractable (see Lemma 27).

### *3.3 Minimal definability*

In many AI applications (some of them will be presented in Sections 6 and 7), one is interested in pointing out a set of propositional symbols $X$ in terms of which $\Sigma$ defines every symbol of a given formula $\Sigma$. Indeed, it is enough to assign truth values to the symbols from such a set $X$ to determine the truth value of $\Sigma$. Thus, one is especially interested in the minimal sets $X$:

**Definition 12 (base)** *Let $\Sigma \in PROP_{PS}$ and $X, Y \subseteq PS$. $X$ is a* **minimal defining family***, or for short a* **base***, for $Y$ w.r.t. $\Sigma$, if and only if $X \sqsubseteq_\Sigma Y$ holds and there is no proper subset $X'$ of $X$ such that $X' \sqsubseteq_\Sigma Y$. The set of all bases for $Y$ w.r.t. $\Sigma$ is denoted by $BS_\Sigma(Y)$.*

**Example 3 (continued)** $\Sigma = \{d400 \Rightarrow l, (d100 \wedge \neg d400) \Rightarrow \neg l, (d4 \wedge \neg d100) \Rightarrow l, \neg d4 \Rightarrow \neg l, d100 \Leftrightarrow (d4 \wedge d25)\}$.
$\{d4, d25\}$ *is a base for* $d100$*; the two sets* $\{d4, d100, d400\}$ *and* $\{d4, d25, d400\}$ *are bases for* $l$*;*
$\Sigma$ *defines* $d4$ *in terms of* $\{l, d100, d400\}$*, but not minimally, since* $\{l, d100\}$ *is a base for* $d4$*; the latter also is a base for* $\{d4, d100, d400\}$.

The following results can be derived easily (we give them without proofs):

(1) $\exists Y \in \mathrm{BS}_\Sigma(X)$ such that $Y \subseteq X$ (and, *a fortiori*, we have $\mathrm{BS}_\Sigma(X) \neq \emptyset$);
(2) $\mathrm{BS}_\Sigma$ is antimonotonic, i.e., $\forall X, Y \subseteq \mathrm{PS}$, if $X \subseteq Y$ then $\mathrm{BS}_\Sigma(X) \succeq \mathrm{BS}_\Sigma(Y)$, where $\succeq$ is the partial order defined by $\mathcal{S}_1 \succeq \mathcal{S}_2$ if and only if $\forall A \in \mathcal{S}_2\ \exists B \in \mathcal{S}_1$ such that $B \subseteq A$.
(3) $\mathrm{BS}_\Sigma(X) = \{\emptyset\}$ if and only if for all $x \in X$ we have $\Sigma \models x$ or $\Sigma \models \neg x$.
(4) $\forall B \in \mathrm{BS}_\Sigma(X)$, we have $B \subseteq \mathrm{Var}(\Sigma) \cup X$.

As to defining a set of propositional symbols, not only we know (from the definition) that $X \sqsubseteq_\Sigma Y$ if and only if $\forall y \in Y, X \sqsubseteq_\Sigma y$, but the following theorem shows that the set of all bases for a set of propositional symbols can be computed from the set of all bases for propositional symbols taken individually by performing pointwise unions and then minimizing the obtained sets. [3]

[3] The operator $*$ such that $\mathrm{BS}_\Sigma(\{x, y\}) = \mathrm{BS}_\Sigma(\{x\}) * \mathrm{BS}_\Sigma(\{y\})$ is sometimes called "unionist product" [23]; it is commutative, associative and idempotent – and as a consequence, it makes sense to write $\mathrm{BS}_\Sigma(X) = *_{x \in X} \mathrm{BS}_\Sigma(\{x\}) = min(\{\cup_{x \in X} B_x | B_x \in \mathrm{BS}_\Sigma(\{x\})\}, \subseteq)$.

**Theorem 13** *Let $\Sigma \in PROP_{PS}$ and $Y = \{y_1, \ldots, y_p\} \subseteq PS$.*

$$BS_\Sigma(Y) = \min\left(\{\bigcup_{i=1}^{p} B_i \mid B_i \in BS_\Sigma(\{y_i\})\}, \subseteq\right).$$

**Proof:** Let $X \subseteq$ PS; we prove that $X \sqsubseteq_\Sigma Y$ if and only if $\exists X_1, \ldots, X_p \subseteq$ PS s.t. $X = X_1 \cup \ldots \cup X_p$ and $X_i \sqsubseteq_\Sigma y_i$ for every $i \in \{1, \ldots, p\}$. Then the theorem follows immediately.

($\Rightarrow$) $X \sqsubseteq_\Sigma \{y_1, \ldots, y_p\}$ means that $X \sqsubseteq_\Sigma y_i$ for every $i \in \{1, \ldots, p\}$. Therefore, taking $X_i = X$ for every $i$ proves the result.

($\Leftarrow$) Assume that $\exists X_1, \ldots, X_p$ such that $X = X_1 \cup \ldots \cup X_p$ and $X_i \sqsubseteq_\Sigma y_i$ for every $i \in \{1, \ldots, p\}$. Since $X_i \sqsubseteq_\Sigma y_i$ and $X_i \subseteq X$, we have $X \sqsubseteq_\Sigma y_i$ for every $i \in \{1, \ldots, p\}$. Therefore $X \sqsubseteq_\Sigma \{y_1, \ldots, y_p\}$.

□

Consequently, it will be enough to compute sets of bases for single propositional symbols only. Note however that a similar result does not hold for *shortest* bases (in terms of cardinality), i.e., a shortest base for $\{x, y\}$ cannot always be written as the union of a shortest base for $\{x\}$ and a shortest base for $\{y\}$.

Note also that it is not the case in general that $\mathrm{Var}(\mathrm{Def}_\Sigma^{\mathrm{Var}(\Sigma)\cup\{y\},l}(y))$ (or $\mathrm{Var}(\mathrm{Def}_\Sigma^{\mathrm{Var}(\Sigma)\cup\{y\},u}(y))$) belongs to $\mathrm{BS}_\Sigma(\{y\})$; such sets are defining sets but they are not necessarily minimal w.r.t. $\subseteq$ (just consider $\Sigma = a \Leftrightarrow b$ and $y = b$ as a counter-example). The conclusion still holds if we consider only the variables $\Sigma$ is not independent of them (i.e., if we replace Var by DepVar in the previous statement) (the same counter-example works).

Note finally that there is no guarantee in the general case that the number of bases for $\{y\}$ w.r.t. $\Sigma$ is polynomial in $|\Sigma|$; for instance, for the following formula $\Sigma$ (equivalent to a Horn CNF formula), $\{y\}$ has $2^n + 1$ bases: $\Sigma = ((\bigwedge_{i=1}^n x_i) \Leftrightarrow y) \wedge \bigwedge_{i=1}^n (x_i \Leftrightarrow x_i')$.

### *3.4 Undefinable propositional symbols*

Because $X \sqsubseteq_\Sigma X$ trivially holds, such instances of the definability relation are typically of little interest. In the theory of relational databases, functional dependencies of the form $X \rightarrow X$ are said to be trivial. In the following, a propositional symbol for which every definition in $\Sigma$ is trivial as such is said to be undefinable.

**Definition 14 (undefinable propositional symbols)** *Let $\Sigma \in PROP_{PS}$ and $y \in PS$. $y$ is* **undefinable** *in $\Sigma$ if and only if $Var(\Sigma) \setminus \{y\} \not\sqsubseteq_\Sigma y$. Otherwise, $y$ is said to be* **definable** *in $\Sigma$.*

We have the following easy connection between undefinable symbols and bases:

**Lemma 15** *Let $\Sigma \in PROP_{PS}$ and $y \in PS$. $y$ is undefinable in $\Sigma$ if and only if $BS_\Sigma(\{y\}) = \{\{y\}\}$.*

**Proof:**

**(1 ⇒ 2)** If $y$ is undefinable in $\Sigma$, then $\mathrm{Var}(\Sigma) \setminus \{y\} \not\sqsubseteq_\Sigma y$. As a consequence, $\emptyset \not\sqsubseteq_\Sigma y$. Hence, $\{y\}$ is a base for $y$ w.r.t. $\Sigma$. Now, let $B \in \mathrm{BS}_\Sigma(\{y\})$. We have $B \subseteq \mathrm{Var}(\Sigma) \cup \{y\}$. If $y \notin B$, then $B \subseteq \mathrm{Var}(\Sigma) \setminus \{y\}$, but this contradicts the fact that $\mathrm{Var}(\Sigma) \setminus \{y\} \not\sqsubseteq_\Sigma y$. Hence, $y \in B$, and therefore $\mathrm{BS}_\Sigma(\{y\}) = \{\{y\}\}$.
**(2 ⇒ 1)** If $\mathrm{BS}_\Sigma(\{y\}) = \{\{y\}\}$, then for every $X \subseteq \mathrm{PS}$, we have $X \sqsubseteq_\Sigma y$ if and only if $y \in X$, which concludes the proof.

□

### *3.5 Necessary and relevant propositional symbols*

Given a formula $\Sigma$ and a set $Y$ of propositional symbols, all propositional symbols in $\mathrm{Var}(\Sigma)$ can be classified according to their usefulness for defining $Y$. The most (resp. the least) important ones are the propositional symbols which are necessary (resp. irrelevant) for defining $Y$, defined as those symbols which belong to all bases for $Y$ (resp. to none of the bases for $Y$). Computing necessary propositional symbols in a preliminary step can also prove valuable for improving the computation of the set of all bases for $Y$ w.r.t. $\Sigma$.

**Definition 16 (necessary and relevant propositional symbols)** *Let $\Sigma \in PROP_{PS}$, $Y \subseteq PS$ and $x \in PS$.*

- *$x$ is a* **necessary propositional symbol** *for $Y$ w.r.t. $\Sigma$ if and only if $x$ belongs to all bases for $Y$ w.r.t. $\Sigma$.*
- *$x$ is a* **relevant propositional symbol** *for $Y$ w.r.t. $\Sigma$ if and only if $x$ belongs to at least one base for $Y$ w.r.t. $\Sigma$ (otherwise, $x$ is an* **irrelevant symbol** *for $Y$ w.r.t. $\Sigma$).*

Since both $Y$ and $\Sigma$ are finite, the set of all bases for $Y$ w.r.t. $\Sigma$ is never empty ($Y \sqsubseteq_\Sigma Y$ always holds). As a consequence, any necessary propositional symbol for $Y$ is a relevant propositional symbol for $Y$. Moreover, it is obvious that any propositional symbol $x$ is relevant to itself whenever $\Sigma \not\models x$ and $\Sigma \not\models \neg x$. The following results are simple characterizations of necessary and relevant

propositional symbols:

**Lemma 17** *Let $\Sigma \in PROP_{PS}$, $Y \subseteq PS$ and $x \in PS$.*

*(1) $x$ is necessary for $Y$ w.r.t. $\Sigma$ if and only if $x \in Y$ and $x$ is undefinable in $\Sigma$.*
*(2) $x$ is relevant for $Y$ w.r.t. $\Sigma$ if and only if it is relevant for some $y \in Y$ w.r.t. $\Sigma$.*
*(3) $x$ is necessary for $Y$ w.r.t. $\Sigma$ if and only if it is necessary for some $y \in Y$ w.r.t. $\Sigma$.*

**Proof:**

**(1, $\Rightarrow$)** Assume that $x$ is necessary for $Y$ w.r.t. $\Sigma$. Since $Y \sqsubseteq_\Sigma Y$, there exists a $B \in \mathrm{BS}_\Sigma(Y)$ such that $B \subseteq Y$. Therefore, since $x \in B$, we have $x \in Y$. Now, suppose that $x$ is definable in $\Sigma$, which means that there exists $Z \subseteq \mathrm{Var}(\Sigma)$ such that $x \notin Z$ and $Z \sqsubseteq_\Sigma x$. Let $B \in \mathrm{BS}_\Sigma(Y)$ and $B' = (B \setminus \{x\}) \cup Z$. From what precedes, we have $B' \sqsubseteq_\Sigma Y$, therefore there is a $B'' \in \mathrm{BS}_\Sigma(Y)$ such that $B'' \subseteq B'$, and since $x$ does not belong to $B''$, it cannot be necessary for $Y$ w.r.t. $\Sigma$.
**(1, $\Leftarrow$)** Assume that $x \in Y$ and $x$ is undefinable in $\Sigma$. $x$ being undefinable in $\Sigma$ is equivalent to $\mathrm{BS}_\Sigma(\{x\}) = \{\{x\}\}$, therefore, as a consequence of Theorem 13 and the fact that $x \in Y$, any $B \in \mathrm{BS}_\Sigma(Y)$ contains $x$, which means that $x$ is necessary for $Y$ w.r.t. $\Sigma$.
**(2, $\Rightarrow$)** If $x$ is relevant for $Y$ w.r.t. $\Sigma$ then there is a $B \in \mathrm{BS}_\Sigma(Y)$ containing $x$, and by Theorem 13, there is a $y \in Y$ and a $B' \in \mathrm{BS}_\Sigma(\{y\})$ such that $y \in B'$; hence $x$ is relevant for $y$ w.r.t. $\Sigma$.
**(2, $\Leftarrow$)** Immediate consequence of Theorem 13.
**(3)** Comes easily from point (1): $x$ is necessary for $Y = \{y_1, \ldots, y_p\}$ w.r.t. $\Sigma$ if and only if $\exists i \in 1 \ldots p$, $x = y_i$ and $x$ is undefinable in $\Sigma$ if and only if $\exists i \in 1 \ldots p$, ($x = y_i$ and $x$ is undefinable in $\Sigma$) if and only if $\exists i \in 1 \ldots p$, $x$ is necessary for $y_i$ w.r.t. $\Sigma$.

□

Point (1) expresses that the propositional symbols necessary for $\mathrm{Var}(\Sigma)$ – hence the "key propositional symbols", by analogy with data bases, are all those that cannot be defined otherwise. Point (2) expresses that it is enough to consider the relation "being relevant for" between propositional symbols instead of sets of propositional symbols. Point (3) expresses the same result for the relation "being necessary for".

As a direct corollary, we obtain the following easy connection between necessary symbols and undefinable ones:

**Corollary 18** *Let $\Sigma \in PROP_{PS}$ and $y \in PS$. $y$ is undefinable in $\Sigma$ if and*

*only if $y$ is necessary for $\{y\}$ w.r.t. $\Sigma$.*

**Example 3 (continued)**

- $BS_\Sigma(Var(\Sigma)) = \{\{d4, d25, d400\}, \{l, d25, d100\}, \{l, d4, d25\}\}$*; therefore, only* $d25$ *is necessary for* $Var(\Sigma)$ *w.r.t.* $\Sigma$*; furthermore,* $BS_\Sigma(\{d25\}) = \{\{d25\}\}$ *and* $d25$ *is undefinable in* $\Sigma$*.*
- $BS_\Sigma(\{l, d4, d100, d400\}) = \{\{d4, d100, d400\}, \{d4, d25, d400\}, \{l, d100, d400\}, \{l, d4, d25\}, \{l, d4, d100\}\}$*; therefore, no propositional symbol is necessary for* $\{l, d4, d100, d400\}$ *w.r.t.* $\Sigma$*, and all propositional symbols of* $Var(\Sigma)$ *are relevant for* $\{l, d4, d100, d400\}$ *w.r.t.* $\Sigma$*.*

Note that the relation "being relevant for" between single propositional symbols is not symmetric. For instance, let $\Sigma = (c \Leftrightarrow (a \vee b))$. $a$ is relevant for $c$, but $c$ is not relevant for $a$. [4]

### *3.6 Unambiguous definability*

In the beginning of this section we wrote that definability imposes that some propositional symbols are fixed *whenever* some other propositional symbols are fixed as well, or in other terms, that the value of $y$ is a function of the values of the variables in $X$. Formally, this is not entirely true, as we can see on the following example: let $\Sigma = (a \Rightarrow b) \wedge ((a \Leftrightarrow b) \Leftrightarrow c)$, $X = \{a, b\}$, and $y = c$. Clearly, $X \sqsubseteq_\Sigma y$. Is the value of $c$ *unambiguously* defined from the values of $a$ and $b$? No, because of the situation $\vec{x}$ where $a$ is true and $b$ false. This situation being inconsistent with $\Sigma$, it trivially holds that $\Sigma \wedge \vec{x} \models y$ *and* $\Sigma \wedge \vec{x} \models \neg y$, thus in this situation the value of $y$ is not unambiguously defined, and we cannot formally say that the value of $y$ is a function of the values of $a$ and $b$. However, in practice, this makes little difference provided that $\Sigma$ is interpreted as a hard constraint (that is, any countermodel of $\Sigma$ is an impossible world that does not need to be considered): in this case, we can safely neglect those $\vec{x}$-worlds that are inconsistent with $\Sigma$, and say that in every *possible* situation, the value of $y$ is a function of the values of $a$ and $b$. Still, in some contexts (especially reasoning about action and change – see Section 6), it is important to know whether such inconsistent $X$-assignments exist or not.

**Definition 19** *Let* $\Sigma \in PROP_{PS}$ *and* $X \subseteq PS$*. We say that* $\Sigma$ *is* strongly $X$-consistent *if and only if for every* $\vec{x} \in \Omega_X$*,* $\vec{x} \wedge \Sigma$ *is consistent. We say*

[4] The relation "being necessary for" between single propositional symbols is of no interest since $y \neq x$ is never necessary for $x$, and $x$ is necessary for $x$ if and only if $x$ is undefinable.

*that* $\Sigma$ **unambiguously defines** $Y$ *in terms of* $X$ *if and only if* $\Sigma$ *is strongly* $X$*-consistent and* $X \sqsubseteq_\Sigma Y$.

Requiring $\Sigma$ to be strongly $X$-consistent has a strong impact on the characterization of explicit definitions. Indeed, the strong $X$-consistency of $\Sigma$ is a necessary and sufficient condition for the unicity (up to logical equivalence) of explicit definitions on $X$ in $\Sigma$:

**Theorem 20** *Let* $\Sigma \in PROP_{PS}$, $X \subseteq PS$ *and* $y \in PS$ *such that* $X \sqsubseteq_\Sigma y$. *Then* $\Sigma$ *is strongly* $X$*-consistent if and only if for any two definitions* $\varphi$, $\psi$ *of* $y$ *on* $X$ *in* $\Sigma$, *we have* $\varphi \equiv \psi$.

**Proof:**

($\Rightarrow$) Assume there exist two non-equivalent formulas $\varphi$ and $\psi$ of $\mathrm{PROP}_X$ such that (a) $\Sigma \models y \Leftrightarrow \varphi$ and (b) $\Sigma \models y \Leftrightarrow \psi$. (a) and (b) imply (c) $\Sigma \models \varphi \Leftrightarrow \psi$. Since $\varphi$ and $\psi$ are not logically equivalent, there exists a $\vec{x} \in \Omega_X$ such that $\vec{x} \models \neg(\varphi \Leftrightarrow \psi)$, which, together with (c), implies that $\vec{x} \wedge \Sigma$ is inconsistent, therefore $\Sigma$ is not strongly $X$-consistent.

($\Leftarrow$) Assume $\Sigma$ is not strongly $X$-consistent. Let then be $\vec{x} \in \Omega_X$ such that $\vec{x} \wedge \Sigma$ is inconsistent. Let $\varphi$ be a definition of $y$ on $X$ in $\Sigma$. If $\vec{x} \models \varphi$ (respectively, $\vec{x} \models \neg\varphi$), then let $\psi$ be the formula of $\mathrm{PROP}_X$, unique up to logical equivalence, whose set of models are exactly the models of $\varphi$ *except* $\vec{x}$ (respectively, the models of $\varphi$ *plus* $\vec{x}$). $\psi$ is also a definition of $y$ on $X$ in $\Sigma$, and $\psi$ is not logically equivalent to $\varphi$.

□

## 4 Computational Aspects

### *4.1 Definability*

The following result is the restriction to propositional logic of a property, which holds in first-order logic, and is due to Padoa [24]. It consists of an entailment-based characterization of (implicit) definability and is useful for identifying tractable restrictions of definability in the propositional case. We give a simple proof which holds for propositional logic: for any $\Sigma$ and any $X \subseteq \mathrm{PS}$, let $rename(\Sigma, X)$ be the formula obtained by replacing in $\Sigma$ in a uniform way every propositional symbol $z$ from $\mathrm{Var}(\Sigma) \setminus X$ by a new propositional symbol $z'$. We have:

**Theorem 21 (Padoa's method)** *[24]*
*If* $y \notin X$, *then* $X \sqsubseteq_\Sigma y$ *if and only if* $(\Sigma \wedge rename(\Sigma, X)) \models y \Rightarrow y'$.

**Proof:** From Theorem 8, we get that $X \sqsubseteq_\Sigma y$ if and only if $\mathrm{Proj}(\Sigma \wedge y, X) \models \neg\mathrm{Proj}(\Sigma \wedge \neg y, X)$. Equivalently, $X \sqsubseteq_\Sigma y$ if and only if $\exists(PS \setminus X).(\Sigma \wedge y) \wedge \exists(PS \setminus X).(\Sigma \wedge \neg y)$ is unsatisfiable. Since quantified variables are dummy ones, when $y \notin X$, $\exists(PS \setminus X).(\Sigma \wedge y) \wedge \exists(PS \setminus X).(\Sigma \wedge \neg y)$ is equivalent to $\exists(PS \setminus X).(\Sigma \wedge y) \wedge \exists(PS' \setminus X').(rename(\Sigma, X) \wedge \neg y')$ where for any subset $Z$ of PS we have $Z' = \{x' \mid x \in Z\}$. This quantified Boolean formula is also equivalent to the following prenex one: $\exists(PS \setminus X) \cup (PS' \setminus X').(\Sigma \wedge y \wedge rename(\Sigma, X) \wedge \neg y')$, which is unsatisfiable if and only if $\Sigma \wedge y \wedge rename(\Sigma, X) \wedge \neg y'$ is unsatisfiable if and only if $(\Sigma \wedge rename(\Sigma, X)) \models y \Rightarrow y'$.

□

Accordingly, whenever $y$ does not belong to $X$, checking definability comes down to a standard deduction check. Since $X \sqsubseteq_\Sigma y$ trivially holds in the remaining case (i.e., $y \in X$), we can conclude that a set-membership test plus a deduction check are always sufficient to decide definability.

We now give the complexity of definability in the general case, as well as in some restricted cases:

**Theorem 22** DEFINABILITY *is* coNP*-complete even under the restriction when* $\Sigma$ *is a Blake formula.*

**Proof:**

- *Membership*: Membership of DEFINABILITY to coNP comes directly from Theorem 21 which gives a polynomial reduction from DEFINABILITY to UNSAT, which is in coNP and coNP is well-known as closed under such reductions.
- *Hardness*: As to hardness, let us exhibit a polynomial reduction from CNF-UNSAT to the restriction of DEFINABILITY to the Blake fragment: let $\varphi = \bigwedge_{i=1}^{m} \gamma_i$ be a CNF formula from $\mathrm{PROP}_{\mathrm{PS}}$ such that $\mathrm{Var}(\varphi) = \{x_1, \ldots, x_n\}$; w.l.o.g., we assume that $\varphi$ does not contain any clause implied by another clause (if it is not the case, we first remove every properly implied clause from it; this can be easily achieved in polynomial time). To $\varphi$ we associate in polynomial time the formula $\Sigma = \bigwedge_{i=1}^{m} (\gamma_i \vee new \vee y) \wedge (\gamma_i \vee \neg new \vee \neg y)$ where $new$ is a fresh variable from $PS \setminus (\mathrm{Var}(\varphi) \cup \{y\})$. We take advantage of the following property, which results directly from the correctness of resolution-based prime implicates algorithms (like Tison's one [18]): a CNF formula $\Sigma$ contains all its prime implicates if and only if whenever two clauses from it have a resolvent $\delta$, there exists a clause $\epsilon \in \Sigma$ s.t. $\epsilon \models \delta$. By construction, every binary resolvent from clauses of $\Sigma$ is tautologous, hence implied by any clause of $\Sigma$. As a consequence, $\Sigma$ contains all its prime implicates, and since it does not contain properly implied clauses, it is a prime implicates formula. Now, from Theorem 8, we have that $X \sqsubseteq_\Sigma y$ if and only if $\mathrm{Proj}(\Sigma \wedge y, X) \models \neg\mathrm{Proj}(\Sigma \wedge \neg y, X)$ if and only if $\exists \bar{X}.(\Sigma \wedge y) \wedge \exists \bar{X}.(\Sigma \wedge \neg y)$ is

unsatisfiable. With $X = \mathrm{Var}(\varphi) \cup \{new\}$, we have that $X \sqsubseteq_\Sigma y$ if and only if $\exists \bar{X}.(\bigwedge_{i=1}^{m}(\gamma_i \vee new \vee y) \wedge (\gamma_i \vee \neg new \vee \neg y) \wedge y) \wedge \exists \bar{X}.(\bigwedge_{i=1}^{m}(\gamma_i \vee new \vee y) \wedge (\gamma_i \vee \neg new \vee \neg y) \wedge \neg y)$ is unsatisfiable. The latter formula is equivalent to $\exists \bar{X}.(\bigwedge_{i=1}^{m}((\gamma_i \vee \neg new) \wedge y)) \wedge \exists \bar{X}.(\bigwedge_{i=1}^{m}((\gamma_i \vee new) \wedge \neg y))$, which is itself equivalent to $\bigwedge_{i=1}^{m}(\gamma_i \vee \neg new) \wedge \bigwedge_{i=1}^{m}(\gamma_i \vee new)$ since $\bar{X} = \{y\}$ and $y \notin \mathrm{Var}(\varphi) \cup \{new\}$. But this formula is also equivalent to $\varphi$ (it is enough to compute all its resolvents over *new* and remove the implied clauses to get $\varphi$). Hence $\varphi$ is unsatisfiable if and only if $X \sqsubseteq_\Sigma y$ and this completes the proof.

□

This theorem generalizes Theorem 2.2 from [22]: we relax here the (useless) assumption that $\Sigma$ is a CNF formula for proving the membership to coNP and constrain $\Sigma$ to belong to the Blake fragment for the hardness part.

Interestingly, it shows that constraining $\Sigma$ to belong to a propositional fragment that is tractable for SAT (as it it the case for IP) does not necessarily lead a tractable restriction of DEFINABILITY.

We also identified the complexity of the minimal definability problem:

**Theorem 23** *Let $\Sigma \in PROP_{PS}$, $X \subseteq PS$ and $y \in PS$. Checking whether $X$ is a minimal defining family for $y$ (*MINIMAL DEFINING FAMILY*) w.r.t. $\Sigma$ is $\mathsf{BH}_2$-complete.*

**Proof:**

- *Membership*: $X$ is a minimal defining family for $y$ w.r.t. $\Sigma$ if and only if $X \sqsubseteq_\Sigma y$ and $\forall X' \subset X$, $X' \not\sqsubseteq_\Sigma y$. Now, $\forall X' \subset X$,$X' \not\sqsubseteq_\Sigma y$ holds if and only if $\forall x \in X$, $X \setminus \{x\} \not\sqsubseteq_\Sigma y$. Thus MINIMAL DEFINING FAMILY is the intersection of a language in coNP and of a language in NP (since the intersection of a linear number of a languages in NP is in NP), which proves membership to $\mathsf{BH}_2$.
- *Hardness*: let $\varphi$ and $\psi$ be two propositional formulas; we associate to them in polynomial time the tuple $L(\langle \varphi, \psi \rangle) = \langle \Sigma, X, y \rangle$ where
  · $\Sigma = ((\neg\psi \wedge x) \Rightarrow y) \wedge ((\neg\psi \wedge \neg x) \Rightarrow \neg y) \wedge ((\neg\varphi) \Rightarrow y)$;
  · $X = \{x\}$;
  · $x$ and $y$ are new propositional symbols, not appearing in $\varphi$ or $\psi$.
  It is easy to check that $\{x\} \sqsubseteq_\Sigma y$ if and only if $\psi$ is unsatisfiable or $\varphi$ is unsatisfiable. Now, $\emptyset \sqsubseteq_\Sigma y$ if and only if $\varphi$ is unsatisfiable. This means that $\{x\}$ is a minimal defining family for $y$ w.r.t. $\Sigma$ if and only if $\psi$ is unsatisfiable and $\varphi$ is satisfiable, i.e., if and only if $\langle \varphi, \psi \rangle$ is an instance of SAT-UNSAT. Thus $L$ is a polynomial (Karp) reduction from SAT-UNSAT to MINIMAL DEFINING FAMILY.

□

When $\Sigma$ is such that deciding whether $X \sqsubseteq_\Sigma y$ holds for any $X \subseteq \mathrm{PS}$ and $y \in \mathrm{PS}$ is tractable, deciding whether $X$ is a minimal defining family for $y$ w.r.t. $\Sigma$ for any $X \subseteq \mathrm{PS}$ and $y \in \mathrm{PS}$ is tractable as well (since $X$ is a minimal defining family for $y$ w.r.t. $\Sigma$ if and only if $X \sqsubseteq_\Sigma y$ and $\forall x \in X, X \setminus \{x\} \not\sqsubseteq_\Sigma y$).

On the other hand, as Theorem 23 suggests it, when $X \sqsubseteq_\Sigma y$ is known to hold, deciding whether $X$ is a minimal defining family for $y$ w.r.t. $\Sigma$ remains computationally hard (unless $\mathsf{P} = \mathsf{NP}$):

**Theorem 24** *Let $\Sigma \in PROP_{PS}$, $X \subseteq PS$ and $y \in PS$ such that $X \sqsubseteq_\Sigma y$. Checking whether $X$ is a minimal defining family for $y$ w.r.t. $\Sigma$ is* NP*-complete.*

**Proof:**

- *Membership*: Membership consists in checking that $\forall x \in X, X \setminus \{x\} \not\sqsubseteq_\Sigma y$, which requires to solve $Card(X)$ (independent) instances of DEFINABILITY. Since DEFINABILITY is in $\mathsf{NP}$, this is also the case of the problem under consideration.
- *Hardness*: By reduction from SAT. Let $\varphi \in \mathrm{PROP_{PS}}$ such that $\mathrm{Var}(\varphi) = \{x_1, \ldots, x_n\}$ a non-empty set. To $\varphi$ we associate in polynomial time $\Sigma = (\varphi \wedge \bigwedge_{i=1}^n (x_i \Leftrightarrow x'_i)) \Leftrightarrow y$ (where $x'_1, \ldots, x'_n$ are fresh atoms from $PS \setminus \{x_1, \ldots, x_n, y\}$) and $X = \{x_1, \ldots, x_n, x'_1, \ldots, x'_n\}$. By construction $\varphi \wedge \bigwedge_{i=1}^n (x_i \Leftrightarrow x'_i)$ is a definition of $y$ on $X$ in $\Sigma$, hence $X \sqsubseteq_\Sigma y$. Now, $\varphi$ is satisfiable if and only if $X$ is a minimal defining family for $y$ w.r.t. $\Sigma$. Indeed if $\varphi$ satisfiable then depends on all its variables $X$ (i.e., there does not exist a formula $\psi$ such that $\psi \equiv \varphi \wedge \bigwedge_{i=1}^n (x_i \Leftrightarrow x'_i)$ and $\mathrm{Var}(\psi) \subset X$). This means that there does not exist a definition of $y$ on a proper subset of $X$ in $\Sigma$, hence $X$ is a minimal defining family for $y$ w.r.t. $\Sigma$. If $\varphi$ is unsatisfiable, then $\Sigma \equiv \neg y$ and $\emptyset \sqsubseteq_\Sigma y$, hence $X$ does not minimally defines $y$ w.r.t. $\Sigma$.

□

Since the transformation from formula definability to propositional symbol definability given by Lemma 7 can be achieved in polynomial time and since propositional symbol definability is a restriction of formula definability, these complexity results apply as well to formula definability.

Now, some tractable restrictions for DEFINABILITY (hence for MINIMAL DEFINING FAMILY) can be easily derived from Theorem 21. We first need to make precise the conditions under which such restrictions are based:

**Definition 25 (stability conditions)** *Let $\mathcal{C}$ be a propositional fragment, i.e., a subset of $PROP_{PS}$.*

- $\mathcal{C}$ *is* **stable by expansion for partial renaming** *if and only if for every* $\Sigma \in \mathcal{C}$ *and for every* $X \subseteq PS$*, we have* $\Sigma \wedge rename(\Sigma, X) \in \mathcal{C}$*.*
- $\mathcal{C}$ *is* **stable by conditioning** *if and only if for every* $\Sigma \in \mathcal{C}$ *and* $\gamma$ *is a satisfiable conjunction of literals, then the conditioning* $\Sigma_\gamma$ *of* $\Sigma$ *by* $\gamma$ *also belongs to* $\mathcal{C}$*.*

**Theorem 26** *Let* $\mathcal{C}$ *be a propositional fragment satisfying the stability conditions listed in Definition 25.* $\mathcal{C}$ *is tractable for* SAT *if and only if the restriction of* DEFINABILITY *when* $\Sigma$ *belongs to* $\mathcal{C}$ *is tractable.*

**Proof:** Let us first show that if $\mathcal{C}$ is tractable for SAT then the restriction of DEFINABILITY is tractable. The key is Theorem 21; there are two cases: if $y \in X$ (which can be obviously decided in polynomial time), then any $\Sigma$ defines $y$ in terms of $X$; otherwise, Theorem 21 shows that $X \sqsubseteq_\Sigma y$ if and only if $\Sigma \wedge rename(\Sigma, X) \models y \Rightarrow y'$. This is equivalent to determine whether $(\Sigma \wedge rename(\Sigma, X))_\gamma$ is inconsistent where $\gamma$ is $y \wedge \neg y'$. By construction, such a formula $(\Sigma \wedge rename(\Sigma, X))_\gamma$ belongs to $\mathcal{C}$ whenever $\Sigma$ belongs to $\mathcal{C}$, because $\mathcal{C}$ is stable by conditioning and expansion by partial renaming; hence the satisfiability of it can be decided in polynomial time.

Conversely, if the restriction of DEFINABILITY when $\Sigma$ belongs to $\mathcal{C}$ is tractable then deciding whether $\emptyset \sqsubseteq_\Sigma new$ (with $new \in Ps \setminus \mathrm{Var}(\Sigma)$) can be achieved in polynomial time. But $\emptyset \sqsubseteq_\Sigma new$ if and only if $\Sigma$ is unsatisfiable. Hence the satisfiability of $\Sigma$ can be decided in polynomial time. □

Note that stability by expansion with partial renaming is strictly less demanding than stability by (bounded) conjunction; for instance, the class of renamable Horn CNF formulas is stable by expansion for partial renaming, but it is not stable by bounded conjunction.

Interestingly, some quite general propositional fragments satisfy the stability conditions given in Definition 25. This is the case for the class of q-Horn formulas (which includes both Krom CNF formulas, Horn CNF formulas and renamable Horn CNF formulas as specific cases) [6] and the class of Decomposable Negation Normal Form (DNNF) formulas (which includes several other important fragments, namely the DNF formulas and the Ordered Binary Decision Diagrams, $\mathrm{OBDD}_<$) [7,9].

**Lemma 27** *The restrictions of* DEFINABILITY *for which* $\Sigma$ *is a q-Horn CNF formula or a DNNF formula are in* P*.*

**Proof:** It is known that the class of q-Horn CNF formulas is tractable for SAT [6]; and it is obvious that it is stable by conditioning; now, stability by expansion with partial renaming comes from the fact that if $V$ is a q-Horn renaming for $\Sigma$, then the set of symbols $V \cup \{rename(x, \emptyset) \mid x \in V \setminus X\}$ is a q-Horn renaming for $\Sigma \wedge rename(\Sigma, X)$. Finally, as to the DNNF class, the

result comes immediately from Propositions 4.1 and 5.1 from [25]. □

Lemma 27 generalizes Theorem 3.1 and Corollary 3.2 from [22], which concern Horn CNF formulas, as well as Theorem 7.1 and Corollary 7.2 from [15], which concern q-Horn CNF formulas. It does not generalize Theorem 3.5 and Corollary 3.6 from [22] (resp. Theorem 7.3 and Corollary 7.4 from [15]), showing the tractability of the restrictions of DEFINABILITY when $\Sigma$ is equivalent to a Horn CNF formula (resp. a q-Horn CNF formula) but is given by its (disjunctively interpreted) Horn (resp. q-Horn) envelope.

Note that Theorem 21 can prove helpful for deciding in polynomial time whether $X \sqsubseteq_\Sigma y$ under restrictions on $\Sigma$ that are outside the scope of Lemma 27. For instance, if $\Sigma = (\varphi \Rightarrow y) \wedge (y \Rightarrow \psi)$ where $\varphi, \psi$ are Horn CNF formulas such that $y \notin \mathrm{Var}(\varphi) \cup \mathrm{Var}(\psi)$, then $X \sqsubseteq_\Sigma y$ can be decided in polynomial time since it amounts to determining whether $\varphi \models \psi$ holds. However, $\Sigma$ is neither a q-Horn CNF formula, nor a DNNF one.

It is interesting to observe that the stability conditions given in Definition 25 are not satisfied by every propositional fragment that is tractable for SAT (for instance the Blake fragment (formulas in prime implicates normal form) does not satisfy any of them).

Now, what about the complexity of unambiguous definability? Checking that $\Sigma$ is strongly $X$-consistent being significantly harder than checking definability, this carries over to unambiguous definability:

**Theorem 28** *Let $\Sigma \in PROP_{PS}$, $X \subseteq PS$ and $y \in PS$.*

- *Deciding whether $\Sigma$ is strongly $X$-consistent is $\Pi_2^p$-complete.*
- *Deciding whether $\Sigma$ unambiguously defines $y$ in terms of $X$ is $\Pi_2^p$-complete.*

**Proof:** Membership is easy in both cases. For deciding whether $\Sigma$ is strongly $X$-consistent, hardness comes from this trivial reduction from $\mathrm{QBF}_{2,\forall}$: $\exists A \forall B \Sigma$ is a valid instance of $\mathrm{QBF}_{2,\forall}$ if and only if $\Sigma$ is strongly $A$-consistent. As for deciding whether $\Sigma$ unambiguously defines $y$ in terms of $X$, it suffices to remark that $\Sigma$ unambiguously defines $X$ in terms of $X$ if and only if $\Sigma$ is strongly $X$-consistent. □

Finally, knowing that $\Sigma$ is strongly $X$-consistent does not change the complexity of definability:

**Theorem 29** *Let $\Sigma \in PROP_{PS}$, $X \subseteq PS$ and $y \in PS$. Given that $\Sigma$ is strongly $X$-consistent, deciding whether $X \sqsubseteq_\Sigma y$ is* coNP*-complete.*

**Proof:** Membership is obvious. Hardness comes from the following reduction from UNSAT: let $\varphi$ be a propositional formula and $z$ a fresh variable, not

appearing in $\varphi$; then $\varphi \in$ UNSAT if and only if $\varphi \vee z \models z$, that is, if $\emptyset \sqsubseteq_{\varphi \vee z} z$, and clearly, $\varphi \vee z$ is strongly $\emptyset$-consistent, because $\varphi \vee z$ is consistent. □

### *4.2 Undefinability, necessity and relevance*

From Theorem 21, we can easily derive the following characterization of undefinable propositional symbols, which is surprisingly simple:

**Lemma 30** *Let $\Sigma \in PROP_{PS}$ and $y \in PS$. $y$ is undefinable in $\Sigma$ if and only if $\Sigma_{y \leftarrow 0} \wedge \Sigma_{y \leftarrow 1}$ is satisfiable.*

**Proof:** By definition, we have that $y$ is undefinable in $\Sigma$ if and only if $\mathrm{Var}(\Sigma) \setminus \{y\} \not\sqsubseteq_\Sigma y$. Since $y \notin \mathrm{Var}(\Sigma) \setminus \{y\}$, from Theorem 21, we get that $y$ is undefinable in $\Sigma$ if and only if $(\Sigma \wedge rename(\Sigma, \mathrm{Var}(\Sigma) \setminus \{y\})) \not\models (y \vee \neg y')$. This is equivalent to state that $(\Sigma \wedge rename(\Sigma, \mathrm{Var}(\Sigma) \setminus \{y\})) \wedge \neg y \wedge y'$ is satisfiable. This is again equivalent to state that the conditioning of $\Sigma \wedge rename(\Sigma, \mathrm{Var}(\Sigma) \setminus \{y\})$ by the satisfiable conjunction of literals $\neg y \wedge y'$ is satisfiable. Now since $y'$ (resp. $y$) does not occur in $\Sigma$ (resp. $rename(\Sigma, \mathrm{Var}(\Sigma) \setminus \{y\})$), this conditioning is equivalent to $\Sigma_{\neg y} \wedge rename(\Sigma, \mathrm{Var}(\Sigma) \setminus \{y\})_{y'}$. Since $rename(\Sigma, \mathrm{Var}(\Sigma) \setminus \{y\})_{y'}$ is equivalent to $\Sigma_y$ (since $y$ is the unique symbol that has been renamed), we obtain that $y$ is undefinable in $\Sigma$ if and only if $\Sigma_{\neg y} \wedge \Sigma_y$ is satisfiable.

□

Necessary propositional symbols can be characterized by means of prime implicants in this simple and elegant way:

**Lemma 31** *Let $\Sigma \in PROP_{PS}$ and $x \in PS$. $x$ is definable in $\Sigma$ if and only if every prime implicant of $\Sigma$ contains $x$ or $\neg x$.*

**Proof:** The prime implicants of $\Sigma$, or equivalently of $(\neg x \wedge \Sigma_{x \leftarrow 0}) \vee (x \wedge \Sigma_{x \leftarrow 1})$, that contain neither $x$ nor $\neg x$, are the prime implicants of $\Sigma_{x \leftarrow 0} \wedge \Sigma_{x \leftarrow 1}$, see e.g., [8]. Since the latter formula is unsatisfiable whenever $x$ is definable in $\Sigma$ (cf. Lemma 30), every prime implicant of $\Sigma$ contains $x$ or $\neg x$ in this situation (and only if $x$ is definable in $\Sigma$). □

We have also derived the following complexity results:

**Theorem 32** *Let $\Sigma \in PROP_{PS}$, $X, Y \subseteq PS$, and $x, y \in PS$.*

*(1) Deciding whether $y$ is undefinable in $\Sigma$ (*UNDEFINABILITY*) is* NP*-complete.*
*(2) Deciding whether $x$ is necessary for $Y$ w.r.t. $\Sigma$ (*NECESSITY*) is* NP*-complete. Hardness still holds if $Y$ is a singleton.*
*(3) Deciding whether $x$ is relevant to $Y$ w.r.t. $\Sigma$ (*RELEVANCE*) is in $\Sigma_2^p$ and*

*both* NP*-hard and* coNP*-hard, hence not in* NP ∪ coNP *(unless the polynomial hierarchy collapses to the first level). Hardness still holds if* $Y$ *is a singleton.*[5]

**Proof:**

(1) Membership is a corollary of Lemma 30. Hardness comes from the following polynomial reduction from SAT: for any propositional formula $\varphi$, let $\Sigma = \varphi \vee y$ where $y \notin \mathrm{Var}(\varphi)$; now, $\varphi$ is satisfiable if and only if $y$ is undefinable in $\Sigma$.

(2) Membership is a corollary of Point (1) above and Point (2) of Lemma 17. Hardness is a consequence of Point (1) above and the equivalence between (1) and (4) in Corollary 18.

(3) Membership is easy: guess $B \subseteq \mathrm{Var}(\Sigma) \cup Y$ and check using a linear number of calls to an NP oracle that $B$ is a minimal defining family for $Y$ w.r.t. $\Sigma$. NP-hardness comes from the following polynomial reduction from SAT: for any propositional formula $\varphi$, let $\Sigma = \varphi \wedge y$ where $y \notin \mathrm{Var}(\varphi)$ and let $Y = \{y\}$; now, $\varphi$ is satisfiable if and only if $y$ is relevant to $Y$ w.r.t. $\Sigma$. coNP-hardness comes from the following polynomial reduction from UNSAT: for any propositional formula $\varphi$ over $X = \{x_1, \ldots, x_n\}$, let $\Sigma = (z \Leftrightarrow y) \wedge (((z \wedge \varphi) \vee x) \Leftrightarrow y)$; if $\varphi$ is unsatisfiable, then $x$ is relevant to $Y = \{y\}$ w.r.t. $\Sigma$ since $\{x\}$ is a base for $Y$ w.r.t. $\Sigma$ in such a case; if $\varphi$ is satisfiable, then $x$ is not relevant to $Y$ w.r.t. $\Sigma$; indeed, $\Sigma$ does not define $Y$ in terms of $X \cup \{x\}$: let $\vec{x}$ be any $X$-model of $\varphi$; we have $\vec{x} \wedge \neg x \wedge \Sigma \equiv (z \Leftrightarrow y)$ showing that instantiating $z$ is necessary to derive the truth value of $y$. Hence, every base for $Y$ w.r.t. $\Sigma$ must contain $z$; since, by construction, $\{z\}$ is a base for $Y$ w.r.t. $\Sigma$, we conclude that $\{z\}$ is the unique base for $Y$ w.r.t. $\Sigma$ when $\varphi$ is satisfiable, and this is enough to conclude the proof.

□

From the definition of undefinable symbols and Lemma 17, it immediately follows that the restrictions of UNDEFINABILITY and of NECESSITY for which $\Sigma$ satisfies the stability conditions listed in Definition 25 are also in P. Such restrictions also make the complexity of RELEVANCE belonging to NP.

### *4.3 Computing explicit definitions*

Theorem 8 and its corollary give us several ways of computing explicit definitions. In particular, they show that when $X \sqsubseteq_\Sigma y$, then the strongest definition of $y$ on $X$ in $\Sigma$ is $\mathrm{Proj}(\Sigma \wedge y, X)$, or equivalently in the case $y \notin X$,

[5] We conjecture that this problem is $\Sigma_2^p$-complete.

$\mathrm{Proj}(\Sigma_{y\leftarrow 1}, X)$.

Such a characterization proves particularly helpful when $\Sigma$ is from a propositional fragment allowing polytime forgetting and conditioning [26]. As a consequence of Theorem 8, we get:

**Lemma 33** *Let $\mathcal{C}$ be any propositional fragment, which is stable by conditioning and enables polytime forgetting (i.e., there exists a polytime algorithm for deriving a formula from $\mathcal{C}$ equivalent to $Proj(\Sigma, X)$ for any formula $\Sigma \in \mathcal{C}$ and a set of symbols $X$). Then for any $\Sigma \in \mathcal{C}$, $X \subseteq PS$ and $y \in PS$ such that $X \sqsubseteq_\Sigma y$, an explicit definition $\Phi_X$ of $y$ on $X$ in $\Sigma$ can be computed in time polynomial in $|\Sigma| + |X|$.*

**Proof:** If $y \in X$ then $y \Leftrightarrow y$ is an explicit definition of $y$ on $X$ in $\Sigma$. Otherwise, from Theorem 8, we have that $\mathrm{Proj}(\Sigma_{y\leftarrow 1}, X)$ is an explicit definition of $y$ on $X$ in $\Sigma$. Under the assumptions of the lemma, a propositional formula equivalent to $\mathrm{Proj}(\Sigma_{y\leftarrow 1}, X)$ can be computed in polynomial time. □

Among the influential propositional fragments enabling both operations in polynomial time are the DNNF one [7,9] and the prime implicates one (see [26]). For instance, $\mathrm{Proj}(\Sigma \wedge y, X)$ can be computed efficiently by selecting from the set $IP(\Sigma)$ of prime implicates of $\Sigma \wedge y$ those belonging to $\mathrm{PROP}_X$ (see e.g., Lemma 8 from [27]). Once this formula has been computed, the truth value of $y$ for any $\vec{x} \in \Omega_X$ can be computed in linear time as the truth value of $\mathrm{Proj}(\Sigma \wedge y, X)_{y\leftarrow 1}$. It is interesting to note that, while both fragments enable the computation of definitions of polynomial size *when $\Sigma$ is known to define $y$ in terms of $X$*, the restrictions of DEFINABILITY they induce do not have the same complexity (unless $\mathsf{P} = \mathsf{NP}$). Thus determining whether $X \sqsubseteq_\Sigma y$ is tractable when $\Sigma$ is a DNNF formula and $\mathsf{coNP}$-complete when $\Sigma$ is a prime implicates formula (see Lemma 27 and Theorem 22).

The $\mathrm{OBDD}_<$ fragment [28,29], a famous subset of DNNF, can also be considered, provided that the variables to be forgotten (i.e., all the variables except those of $X$) are the final variables w.r.t. the total, strict ordering $<$ on variables, associated to the fragment [30,31]. Accordingly, the previous corollary completes some results reported in [22] (resp. in [15]), showing that when $\Sigma$ is a Horn CNF formula (resp. a q-Horn CNF formula), then every variable $y$ has an explicit definition in $\Sigma$ that is equivalent to a positive conjunction of literals (resp. a conjunction of literals or a clause) (hence, is of polynomial size w.r.t. the input).

While the possibility to compute an explicit definition $\Phi_X$ of $y$ on $X$ in $\Sigma$ in polynomial time in some restricted cases (and to determine in polynomial time that no such definition exists otherwise), ensures that the size of this definition is polynomially bounded, this cannot be guaranteed in the general case, unless $\mathsf{P} = \mathsf{NP}$ (this is a direct consequence of Theorem 22).

Actually, the situation is even computationally worse in the general case, since we can prove that there is no way to compute definitions in *polynomial space* in the general case (under the usual assumptions of complexity theory).

**Theorem 34** *Let $\Sigma$ be a formula from $PROP_{PS}$. Let $X \subseteq PS$ and let $y \in PS$. In the general case, the size of any explicit definition $\Phi_X$ of $y$ in $\Sigma$ is not polynomially bounded in $|\Sigma| + |X|$ unless* $\mathsf{NP} \cap \mathsf{coNP} \subseteq \mathsf{P/poly}$.

**Proof:** We exploit a close connection between the definability problem and the interpolation one.

Let $\varphi$, $\psi$ be two formulas from $\mathrm{PROP}_{\mathrm{PS}}$. A formula $\alpha$ from $\mathrm{PROP}_{\mathrm{PS}}$ is an interpolant of $\langle \varphi, \psi \rangle$ if and only if $\mathrm{Var}(\alpha) \subseteq \mathrm{Var}(\varphi) \cap \mathrm{Var}(\psi)$ and $\varphi \models \alpha$ and $\alpha \models \psi$ hold.

Indeed, it is known that in the general case the size of any interpolant $\alpha$ of $\langle \varphi, \psi \rangle$ is not polynomially bounded in $|\varphi| + |\psi|$ unless $\mathsf{NP} \cap \mathsf{coNP} \subseteq \mathsf{P/poly}$ [32].

To every pair $\langle \varphi, \psi \rangle$, we can associate in polynomial time the pair $\langle \Sigma, new \rangle$ where $\Sigma = (\psi \Rightarrow new) \wedge (new \Rightarrow \varphi)$, $new \in \mathrm{PS} \setminus (\mathrm{Var}(\varphi) \cup \mathrm{Var}(\psi))$. The point is that $\varphi \models \psi$ if and only if $X \sqsubseteq_\Sigma new$. Moreover, every interpolant of $\langle \varphi, \psi \rangle$ is a definition of $new$ on $\mathrm{Var}(\varphi) \cap \mathrm{Var}(\psi)$ w.r.t. $\Sigma$ and the converse also holds. Indeed, from Craig's interpolation theorem in propositional logic, $\varphi \models \psi$ holds if and only if there exists an interpolant of $\langle \varphi, \psi \rangle$. Now:

- If way. Let $\Phi_X$ be any explicit definition of $new$ on $X = \mathrm{Var}(\varphi) \cap \mathrm{Var}(\psi)$ w.r.t. $\Sigma$. We have $\Sigma \models new \Leftrightarrow \Phi_X$. This is equivalent to state that (1) $(\psi \Rightarrow new) \wedge (new \Rightarrow \varphi) \models new \Rightarrow \Phi_X$, and (2) $(\psi \Rightarrow new) \wedge (new \Rightarrow \varphi) \models \Phi_X \Rightarrow new$. (1) is equivalent to $(\psi \Rightarrow new) \wedge (new \Rightarrow \varphi) \wedge new \wedge \neg\Phi_X$ is unsatisfiable, or equivalently to $new \wedge \varphi \wedge \neg\Phi_X$ is unsatisfiable. Since $new \notin \mathrm{Var}(\psi) \cup \mathrm{Var}(\varphi)$, we have $new \notin X$. Accordingly, (1) is equivalent to $\varphi \wedge \neg\Phi_X$ is unsatisfiable, i.e., $\varphi \models \Phi_X$. From (2), it is easy to derive in a similar way that $\Phi_X \models \psi$. Hence, any $\Phi_X$ is an interpolant of $\langle \varphi, \psi \rangle$.
- Only-if way. Let $\alpha_X$ be any interpolant of $\langle \varphi, \psi \rangle$. By definition, we have $\models (\varphi \Rightarrow \alpha_X) \wedge (\alpha_X \Rightarrow \psi)$. Subsequently, $\Sigma \equiv (\psi \Rightarrow new) \wedge (new \Rightarrow \varphi) \wedge (\varphi \Rightarrow \alpha_X) \wedge (\alpha_X \Rightarrow \psi)$. We immediately obtain that $\Sigma \models new \Leftrightarrow \alpha_X$. Thus, $X \sqsubseteq_\Sigma new$ and every interpolant of $\langle \varphi, \psi \rangle$ is an explicit definition of $new$ on $X = \mathrm{Var}(\varphi) \cap \mathrm{Var}(\psi)$ w.r.t. $\Sigma$.

□

In this section, we present an algorithm for generating a base $X$ for a propositional symbol $y$ w.r.t. a formula $\Sigma$, if any. $X$ will be required to be contained in a fixed set of "acceptable" propositional symbols $V^*$. We called such a base a $V^*$- base. The role of $V^*$ is to focus on interesting bases, only; for instance, in a discriminability problem, $V^*$ will be the set of testable propositional symbols. In particular, if one wants to know whether $y$ is undefinable or not in $\Sigma$, then $V^*$ is set to $\mathrm{Var}(\Sigma) \setminus \{y\}$.

This algorithm (described by the function `Find-A-Base` below) is a greedy algorithm which considers all the propositional symbols of $V^*$ in any order (nevertheless the use of heuristics for determining this order may reduce the search time) and throw them away when they are not necessary for forming a base from the current set of acceptable propositional symbols. The inputs of `Find-A-Base` are $V^*$, $y$ and $\Sigma$ and its output is a subset of $V^*$ or "failure".

This algorithm calls a function `Defines` which checks whether a given subset of propositional symbols defines $y$ w.r.t. $\Sigma$. How $\Sigma$ is represented and how the function `Defines` is implemented will be discussed separately.

**if not** `Defines`($V^*$, $y$, $\Sigma$) **then**
  **return** "failure"
**else**
  $X \leftarrow V^*$
  **for** $x \in V^*$ **do**
    **if** `Defines`($X \setminus \{x\}$, $y$, $\Sigma$) **then**
      $X \leftarrow X \setminus \{x\}$
    **end if**
  **end for**
  **return** $X$
**end if**

The following easy lemma states that the algorithm `Find-A-Base` is correct:

**Lemma 35** *Provided that* `Defines`$(X, y, \Sigma)$ *returns true if and only if* $X \sqsubseteq_\Sigma y$, `Find-A-Base` *returns a* $V^*$*-base for* $y$ *w.r.t.* $\Sigma$ *if there exists such a base, "failure" otherwise.*

**Proof:** Straightforward. □

This algorithm can readily be extended to an algorithm for generating a base $X$ for a set $Y$ of propositional symbols w.r.t. a formula $\Sigma$. It suffices to replace $y$ by $Y$ within each call to `Defines`, and to extend the latter function to such sets $Y$ (this is obvious given the definition of implicit definability).

It can also be extended to an algorithm for deriving all $V^*$-bases for a set $Y$, through a judicious way to search the whole set $2^{V^*}$ (see the set enumeration tree algorithm in [33]). This task is clearly more computationally expensive than computing a single base, especially due to the number of such bases (which can be exponential, as explained before); however, as Theorem 22 suggests it, its computational cost is not solely due to the number of bases:

**Theorem 36** *Unless* $\mathsf{P} = \mathsf{NP}$*, there exists no polynomial time algorithm for computing a* $V^*$*-base for a propositional symbol* $y$ *w.r.t. a CNF formula* $\Sigma$*.*

**Proof:** Let $\varphi$ be a CNF formula and let $y \notin \mathrm{Var}(\varphi)$; let $V^* = \mathrm{Var}(\varphi) \cup \{y\}$; let $\Sigma = \varphi \wedge y$. If $\varphi$ is unsatisfiable (resp. satisfiable), then $\emptyset$ (resp. $\{y\}$) is the unique $V^*$-base for $y$ w.r.t. $\Sigma$. If a polynomial time algorithm for computing a $V^*$-base for $y$ would exist, then after running it on $y$ and $\Sigma$, there are two possibilities: either the computed base is $\emptyset$ and in this case, $\varphi$ is unsatisfiable, or it is $\{y\}$ and in this case $\varphi$ is satisfiable. But this would be a polynomial time algorithm for deciding whether a CNF formula $\varphi$ is satisfiable. Hence, SAT would belong to $\mathsf{P}$. □

Since Theorem 36 also holds when $y$ has a single $V^*$-base w.r.t. $\Sigma$, it strengthens Theorem 3.1 from [34] showing that there exists no polynomial total time (i.e., polynomial in the size of the input plus the size of the output) for computing all the minimal functional dependencies which hold in $\Sigma$, unless $\mathsf{P} = \mathsf{NP}$ when $\Sigma$ is a CNF formula.[6]

In practice, the task of deriving all $V^*$-bases for a set $Y$ w.r.t. $\Sigma$ can be improved in some situations by computing first the set of all necessary variables and the set of all relevant variables for $Y$ w.r.t. $\Sigma$; all irrelevant variables can be removed from $V^*$ before running the algorithm, and subsets of $2^{V^*}$ which do not contain all necessary variables can be skipped during the search.

Clearly enough, the simple algorithm `Find-A-Base` above does not run in polynomial time in the worst case (since this is not the case for the function `Defines`, unless $\mathsf{P} = \mathsf{NP}$). This coheres with Theorem 36 showing that no such algorithm exists, unless $\mathsf{P} = \mathsf{NP}$.

Now, there are several possible ways to implement the function `Defines` depending on the propositional fragment $\Sigma$ belongs to. If $\Sigma$ is a CNF formula, then one can easily implement `Defines` by taking advantage of a SAT solver (and many such solvers with impressive performances are available nowadays).

[6] In the same paper, the authors also showed that, when $\Sigma$ is given by the set of its models (over $\mathrm{Var}(\Sigma)$) this task is polynomially equivalent to the problem of dualizing a positive theory (or, equivalently, of computing the transversals of a hypergraph), for which no polynomial time algorithm is known but a pseudo-polynomial algorithm exists.

In the case when the syntactic restrictions on $\Sigma$ makes definability polynomial, then the search for a $V^*$-base is itself polynomial because it consists in $|V^*|+1$ definability tests. Additionally, when $\Sigma$ belongs to a propositional fragment satisfying the conditions listed in Definition 25, and $X$ is a base for $y$ w.r.t. $\Sigma$, the truth value of $y$ of can be computed in time polynomial in $|\Sigma|+|X|$ for every $X$-world $\vec{x}$. Indeed, checking whether $\Sigma_{\vec{x},\neg y}$ is satisfiable can be done in polynomial time and the truth value of this test gives the truth value of $y$.

## 5 Definability and Hypothesis Discriminability

In this section, we investigate a notion which is closely related to definability and which also has many practical applications ranging from fault isolation in diagnosis to decision under partial observability. Intuitively, given a set of propositional formulas $H = \{h_1 \ldots h_n\}$, which represent mutually exclusive and exhaustive hypotheses w.r.t. a knowledge base $\Sigma$ (i.e., $\forall h, h' \in H$, if $h \neq h'$ then $\Sigma \models \neg(h \wedge h')$ and $\Sigma \models \bigvee_{i=1}^{n} h_i$) and a set $X$ of available binary tests (encoded as propositional symbols), $X$ discriminates $H$ w.r.t. $\Sigma$ if the knowledge of the truth values of propositional symbols of $X$ helps finding out which one of the $h_i$ is true.

**Definition 37 (discrimination)**

- *An input of a* **discrimination problem** *is a triple $\langle \Sigma, X, H \rangle$ which consists of a consistent formula $\Sigma$, a set of test variables $X$ s.t. $X \subseteq Var(\Sigma)$ and a set $H = \{h_1, \ldots h_n\}$ of formulas which are mutually exclusive and exhaustive w.r.t. $\Sigma$.*
- *$X$* **discriminates** *$H$ w.r.t. $\Sigma$ if and only if $\forall \vec{x} \in \Omega_X \ \exists h \in H$ s.t. $\vec{x} \wedge \Sigma \models h$.*
- *$X$* **discriminates minimally** *$H$ w.r.t. $\Sigma$ if and only if $X$ discriminates $H$ w.r.t. $\Sigma$ and no proper subset of $X$ does it.*

There are many activities (including diagnosis and decision under uncertainty) where one wishes to discriminate among a set of hypotheses $h_i, i = 1, \ldots, n$ given a set of available tests. Let us illustrate it, focusing on the consistency-based diagnosis setting [35] (things are similar in the abductive diagnosis setting with respect to the discrimination issue).

**Definition 38 (minimal diagnosis)** *[35] Let $\langle SD, COMPS, OBS \rangle$ be the input of a diagnosis problem ($SD$ is a conjunction of propositional formulas representing the system description, $COMPS$ is a set of symbols denoting the components of the system and $OBS$ is a conjunction of literals representing the initial observations). A* **minimal consistency-based diagnosis** *for $\langle SD, COMPS, OBS \rangle$ is a minimal subset $\Delta$ of $COMPS$ such that $SD \wedge OBS \wedge AB(\Delta)$ is consistent, where $AB(\Delta)$ is the formula $\bigwedge_{c \in \Delta} AB\_c \wedge$*

$\bigwedge_{c \in COMPS \setminus \Delta} \neg AB\_c$ *(each* $AB\_c$ *is a propositional symbol meaning that that the corresponding component* $c$ *is "abnormal", i.e., it does not work properly).*

**Definition 39 (fault isolation)** *Let* $\langle SD, COMPS, OBS \rangle$ *be the input of a diagnosis problem, and* $TB = \{t_1, \ldots t_n\}$ *a test base over some of the propositional symbols of the system (set of available measures); we have* $TB \subseteq Var(SD \cup OBS)$*. The input* $\langle SD, COMPS, OBS, TB \rangle$ *of the* **fault isolation problem** *is the input* $\langle \Sigma, TB, HYP \rangle$ *of the discrimination problem defined by* $\Sigma = SD \wedge OBS$*, TB, and* $HYP = \{AB(\Delta) \mid \Delta$ *is a minimal consistency-based diagnosis for* $\langle SD, COMPS, OBS \rangle\} \cup \{\bigwedge_\Delta \neg AB(\Delta) \mid \Delta$ *is a minimal consistency-based diagnosis for* $\langle SD, COMPS, OBS \rangle\}$*.*

By construction, HYP is a set of mutually exclusive and exhaustive hypotheses w.r.t. $\Sigma$.

Interestingly, there is a direct link between hypothesis discriminability and definability:

**Theorem 40** *Let* $\langle \Sigma, X, H = \{h_1, \ldots, h_n\} \rangle$ *be a discrimination problem. Let* $\Sigma' = \Sigma \wedge \bigwedge_{i=1}^{n} (h_i \Leftrightarrow h^i_{new})$*, where each* $h^i_{new} \in PS \setminus Var(\{\Sigma\} \cup H)$ *is a new symbol. Then* $X$ *discriminates* $H$ *w.r.t.* $\Sigma$ *if and only if* $\Sigma'$ *defines* $H_{new} = \{h^i_{new} \mid i \in 1 \ldots n\}$ *in terms of* $X$*.*

**Proof:**

($\Rightarrow$) If $\forall \vec{x} \in \Omega_X \; \exists h_{\vec{x}} \in H$ s.t. $\vec{x} \wedge \Sigma \models h_{\vec{x}}$, then $\forall \vec{x} \in \Omega_X$ ($\exists h_{\vec{x}} \in H$ s.t. $\vec{x} \wedge \Sigma \models h_{\vec{x}}$ and $\forall h \in H \setminus \{h_{\vec{x}}\}$, we have $\vec{x} \wedge \Sigma \models \neg h$; indeed, $\Sigma \models \neg h_{\vec{x}} \vee \neq h$ for every $h \in H \setminus \{h_{\vec{x}}\}$ since $H$ contains mutually exclusive hypotheses given $\Sigma$. Thus, $\forall \vec{x} \in \Omega_X \; \forall h \in H$ ($\vec{x} \wedge \Sigma \models h$ or $\vec{x} \wedge \Sigma \models \neg h$) holds. Hence $\Sigma'$ defines $H_{new}$ in terms of $X$.

($\Leftarrow$) If $\Sigma'$ defines $H_{new}$ in terms of $X$ then $\forall h \in H \; \forall \vec{x} \in \Omega_X$ ($\vec{x} \wedge \Sigma \models h$ or $\vec{x} \wedge \Sigma \models \neg h$) holds. Equivalently, $\forall \vec{x} \in \Omega_X \; \forall h \in H$ ($\vec{x} \wedge \Sigma \models h$ or $\vec{x} \wedge \Sigma \models \neg h$) holds. Assume that $\forall \vec{x} \in \Omega_X \; \forall h \in H \; \vec{x} \wedge \Sigma \models \neg h$. Since $H$ is exhaustive given $\Sigma$, this is possible only if $\Sigma$ is unsatisfiable. In such a case, $X$ trivially discriminates $H$ w.r.t. $\Sigma$. In the remaining case, we have that $\forall \vec{x} \in \Omega_X \; \exists h \in H \; \vec{x} \wedge \Sigma \models h$. Hence $X$ trivially discriminates $H$ w.r.t. $\Sigma$.

□

Clearly enough, one can take advantage of this polynomial reduction and the results reported in the previous sections to compute discriminating sets and minimal discriminating sets. Thus, when dealing with mutually exclusive and exhaustive sets of hypotheses, bases can be used to design minimal test inputs ([36] [37]) in order to isolate faulty components in model-based diagnosis (in this case hypotheses correspond to candidate diagnoses, and testable propositional symbols correspond most often to available measurements). Note that

McIlraith's notions of relevant or necessary tests [37] have some counterparts in our framework (for instance, a necessary test corresponds to a propositional symbol without which the hypothesis space cannot be discriminated). Lastly, the algorithm for computing bases described before can be used to design conditional test policies (where tests are performed sequentially and conditioned by the outcomes of previous tests – see [38] for the case of mutually exclusive hypotheses).

Conversely, the definability problem can be also reduced to the hypothesis discriminability problem (in presence of mutually exclusive hypotheses). Indeed, a consistent formula $\Sigma$ defines $y$ in terms of $X$ if and only if $X$ discriminates $H = \{y, \neg y\}$ w.r.t. $\Sigma$. Since both reductions are polytime ones, this is enough to show that deciding whether $X$ discriminates $H$ w.r.t. $\Sigma$ (HYPOTHESIS DISCRIMINABILITY) is a coNP-complete problem.

## 6 Propositional Definability and Reasoning about Action and Change

### *6.1 Determinism, executability, and successor state axioms*

In this section, we show that definability is also closely related to several issues pertaining to reasoning about action and change.

Let $F$ be a finite set of fluents (i.e., a subset of PS). Define $F_t = \{f_t \mid f \in F\}$ and $F_{t+1} = \{f_{t+1} \mid f \in F\}$, two sets of fluents indexed by time points. Let $\Sigma_\alpha$ be a propositional action theory describing action $\alpha$, that is, a formula of $\mathrm{PROP}_{F_t \cup F_{t+1}}$, such that $(\vec{f_t}, \vec{f'_{t+1}}) \models \Sigma_\alpha$ holds if and only if $\vec{f'}$ is a possible successor state of $\vec{f}$ by $\alpha$. The *transition function* for $\alpha$ is the binary relation $R_\alpha$ on $\Omega_F$ defined by $R_\alpha(\vec{f}, \vec{f'})$ iff $(\vec{f_t}, \vec{f'_{t+1}}) \models \Sigma_\alpha$. Then:

- $\alpha$ is **deterministic** if for every $\vec{f} \in \Omega_F$ there is at most one $\vec{f'} \in \Omega_F$ such that $R_\alpha(\vec{f}, \vec{f'})$.
- $\alpha$ is **fully executable** if for every $\vec{f} \in \Omega_F$ there is a $\vec{f'} \in \Omega_F$ such that $R_\alpha(\vec{f}, \vec{f'})$.

Now, it is easy to check that determinism and full executability are expressed simply within the notions of definability and strong consistency:

**Lemma 41**

- *$\alpha$ is deterministic if and only if $F_t \sqsubseteq_{\Sigma_\alpha} F_{t+1}$.*
- *$\alpha$ is fully executable if and only if $\Sigma_\alpha$ is strongly $F_t$-consistent.*

**Proof:** Straightforward. □

Putting these two points together, $\alpha$ is deterministic and fully executable if and only if $F_{t+1}$ is unambiguously defined from $F_t$ w.r.t. $\Sigma_\alpha$.

Furthermore, even when an action $\alpha$ is not "fully" deterministic, it may be deterministic *for some fluents*. Let us say that $\alpha$ is **deterministic** for $f$ if and only if for any $\vec{f} \in \Omega_F$ and any two states $\vec{f}'_1, \vec{f}'_2 \in \Omega_F$ such that $R_\alpha(\vec{f}, \vec{f}'_1)$ and $R_\alpha(\vec{f}, \vec{f}'_2)$ then $\vec{f}'_1$ and $\vec{f}'_2$ give the same truth value to $f$. Clearly, definability allows for identifying the fluents for which $\alpha$ is deterministic. Moreover, when $\alpha$ is deterministic for $f$, any definition of $f_{t+1}$ on $F_t$ in $\Sigma_\alpha$ corresponds to a *successor state axiom* [39]. (See also the final example of [11]).

**Example 42** *Let $\alpha$, $\beta$ and $\gamma$ be the actions defined by the following theories:*
$\Sigma_\alpha = (a_{t+1} \Leftrightarrow \neg a_t) \wedge (a_t \Rightarrow b_{t+1}) \wedge (b_t \Rightarrow b_{t+1}) \wedge ((\neg a_t \wedge \neg b_t) \Rightarrow \neg b_{t+1})$,
$\Sigma_\beta = (a_{t+1} \Leftrightarrow \neg a_t) \wedge (a_t \Rightarrow b_{t+1}) \wedge (b_t \Rightarrow (a_{t+1} \wedge b_{t+1})) \wedge ((\neg a_t \wedge \neg b_t) \Rightarrow \neg b_{t+1})$,
$\Sigma_\gamma = (a_{t+1} \Leftrightarrow \neg a_t) \wedge (a_t \Rightarrow b_{t+1}) \wedge (b_t \Rightarrow b_{t+1})$.
*$\{a_t, b_t\} \sqsubseteq_{\Sigma_\alpha} \{a_{t+1}, b_{t+1}\}$ holds, therefore $\alpha$ is deterministic, and $\Sigma_\alpha$ is strongly $\{a_t, b_t\}$-consistent, therefore $\alpha$ is fully executable. The successor state axiom of $q$ corresponds to the definition of $q$ (it is unique up to logical equivalence, due to Theorem 20): $b_{t+1} \equiv (a_t \vee b_t)$.*
*$\Sigma_\beta$ is not strongly $\{a_t, b_t\}$-consistent, because $a_t \wedge b_t \wedge \Sigma_\beta$ is inconsistent. Therefore, $\beta$ is not fully executable (but it is deterministic). There are two non-equivalent definitions of fluents at time $t+1$, hence two successor state axioms: $b_{t+1} \equiv (a_t \vee b_t)$ and $b_{t+1} \equiv (a_t \Leftrightarrow \neg b_t)$.*
*$\{a_t, b_t\} \sqsubseteq_{\Sigma_\gamma} \{a_{t+1}, b_{t+1}\}$ does not hold, therefore $\gamma$ is not deterministic (but it is fully executable). However, it is deterministic as far as fluent $a$ is concerned, since $\{a_t, b_t\} \sqsubseteq_{\Sigma_\gamma} a_{t+1}$ holds.*

Note that usually, we are given initially a set of causal rules from which, using some completion process (e.g., [40,41]), we compute successor state axioms and then finally $\Sigma_\alpha$. Computing successor state axioms as definitions is the reverse process of the latter completion process: from an action theory already compiled in its propositional form $\Sigma_\alpha$, we find the successor state axioms (and then possibly a compact description of the effects of $\alpha$ by causal rules).

Due to the connections made precise by Lemma 41, many notions and results of the paper apply to reasoning about action.

For instance, as a direct consequence of Theorem 20 and Lemma 41, when an action is fully executable and deterministic for $f$, there exists only one successor state axiom for $f$ (up to logical equivalence) – it is indeed the case for $\alpha$, but not for $\beta$ in Example 42.

Definability proves also useful for characterizing regression. Given a propositional formula $\psi \in \mathrm{PROP}_{\mathrm{PS}}$, the **(deductive) regression** of $\psi$ by $\alpha$ is the formula $\mathrm{reg}(\psi, \alpha)$ (unique up to logical equivalence) such that $\mathrm{Mod}(\mathrm{reg}(\psi, \alpha)) =$

$\bigcup_{\vec{f'} \models \psi} R_\alpha^{-1}(\vec{f'})$. The **abductive regression** of $\psi$ by $\alpha$ is the formula $\mathrm{Reg}(\psi, \alpha)$ (unique up to logical equivalence) such that $\mathrm{Mod}(\mathrm{Reg}(\psi, \alpha)) = \{\vec{f} \mid R_\alpha(\vec{f}) \subseteq \mathrm{Mod}(\psi)\}$. While we have $\mathrm{Reg}(\psi, \alpha) \models \mathrm{reg}(\psi, \alpha)$ in the general case, $\mathrm{reg}(\psi, \alpha)$ and $\mathrm{Reg}(\psi, \alpha)$ are equivalent when $\alpha$ is deterministic [41].

For any formula $\varphi$ from $\mathrm{PROP}_F$, let us note $\varphi_t$ the formula from $\mathrm{PROP}_{F_t}$ obtained by substituting in a uniform way in $\varphi$ every symbol $a$ by $a_t$; we have:

**Theorem 43** *Let $\alpha$ be a deterministic and fully executable action. For any fluent $f \in F$ and any formula $\psi \in PROP_F$, $reg(\psi, \alpha)_t$ (or equivalently $Reg(\psi, \alpha)_t$) is equivalent to any definition of $z_{t+1}$ on $F_t$ in $\Sigma_\alpha \wedge (\psi_{t+1} \Leftrightarrow z_{t+1})$, where $z$ is a fresh symbol (not in $F$).*

**Proof:** First observe that from Theorem 20 and Lemma 41, it makes sense to consider any definition (since all of them are equivalent). Now, we have $\mathrm{reg}(\psi, \alpha)_t \equiv \mathrm{Proj}(\Sigma_\alpha \wedge \psi_{t+1}, F_t)$ (see Proposition 5 in [41]). The latter formula is also equivalent to $\mathrm{Proj}(\Sigma_\alpha \wedge (\psi_{t+1} \Leftrightarrow z_{t+1}) \wedge z_{t+1}, F_t)$. Finally, Theorem 8 concludes the proof. □

This result can be generalized to actions that are not fully executable. We omit it for the sake of brevity, as well as applications of definability to progression and planning.

### *6.2 Ramification*

Another role of definability in reasoning about change is in the handling of *ramification*, or indirect action effects. A way to address the well-known problem consists in finding out fluents that can be derived from primitive ones (called a frame) within the knowledge base, and to apply change on reduced world descriptions (composed of primitive fluents, only) [42]. Many formalisms for reasoning about change, adhere to this approach that has been implemented in various planning systems (e.g., in the early system BUILD [43]).

Let us describe more formally the role of definability for dealing with the ramification problem. Let $F$ be a set of fluents, and $\Sigma$ be a propositional formula expressing some constraints on the values that fluents may take (at any time point). Finding a partition of $F$ between a set $F_P$ of primary fluents and a set $F_D$ of derived fluents comes down to find a base for $F$ with respect to $\Sigma$. Clearly, several choices are generally possible, since $\mathrm{BS}_\Sigma(F)$ is generally not a singleton. The goal being to come up with action descriptions that are as concise as possible, a good heuristics consists in choosing a base of $F$ of minimum cardinality.

## 7 Yet Another Application to AI: Automated Reasoning

The notion of definability proves valuable in automated reasoning for several tasks. For instance, identifying functionally dependent propositional symbols is a way for finding out variable orderings that may prevent the OBDD representation of a formula from an exponential size blowup [44].

Identifying definability relations between variables can also prove useful for the *satisfiability issue*. [45] have shown how definability can be exploited in local search for the satisfiability problem. The idea is to concentrate the search on undefinable variables, and to handle the remaining ones by exploiting definability relations. They reported some empirical results showing their algorithm DAGSAT valuable. [46] considered the role of definability relations (what they called gates) to reduce the search space explored by complete DPLL-like algorithms for SAT. In a nutshell, the idea is that a definable variable should not be elected by a branching rule before the variables from a base for it have been assigned. Accordingly, the undefinable variables of the input CNF formula $\Sigma$ should be considered first. Since deciding definability relations is a coNP-complete problem, they considered only those relations that can be discovered through (linear time) unit propagation of literals (such relations include equivalent literals which have been considered in several other papers, see e.g., [47]); in [46], the explicit definitions of variables which are discovered take the form of a conjunction of literals or a clause (depending on the sign of the propagated literal); interestingly, once the variables occurring in an explicit definition of $y$ have been assigned, unit propagation in $\Sigma$ proves enough to get $y$ assigned as well. The resulting set of functional dependencies induces a "relevance" graph whose set of vertices is $\mathrm{Var}(\Sigma)$ and the set of arcs contains $(x, z)$ whenever one of the found definitions of variable $z$ bears on variable $x$. When no undefinable variables occur in $\Sigma$ (or the CNF formulas obtained by conditioning and simplifying $\Sigma$ at subsequent steps of the algorithm), the corresponding "relevance graph" contains no source (i.e., a node of incoming degree 0); then polynomial time heuristics for approximating a minimal cycle cutset of the graph are used, and the variables from the resulting set (also known as a strong backdoor) are assigned first. This approach exhibited interesting performances on some benchmarks used during the SAT'02 and SAT'03 competitions, and appeared as the best performer on hand-made instances at the SAT'03 competition. [48] also reported on the possible advantages and drawbacks of taking advantage of such "independent (i.e., undefinable) variable selection" heuristics.

## 8 Other Related Work

As evoked previously, propositional definability is closely related to the notions of strongest necessary and weakest sufficient conditions and to the notion of functional dependencies in propositional logic. In this section, we make precise the main differences between the contribution of the present paper and the (closest) related ones from the literature. Before concluding the paper, we also briefly present some other related work, where definability is considered in more complex logical settings than classical propositional logic.

### *8.1 Functional dependencies*

The closest work to our own one is described in three papers by Ibaraki, Kogan and Makino [22,15,34]. In those papers, Ibaraki, Kogan and Makino presented a number of results related to functional dependencies.

In [22,15], they reported many very interesting results about issues that we mainly ignored here. Among them is the condensation issue: the basic idea comes from the observation that when $X \sqsubseteq_\Sigma y$ and $y \notin X$, then $\Sigma$ can be simplified by "removing" $y$ (i.e., forgetting $y$ in $\Sigma$), while keeping track of an explicit definition of $y$ on $X$ in $\Sigma$; at the semantical level, no loss of information results from such a process; condensing $\Sigma$ consists in repeating it in an iterative way, unless reaching a formula without any non-trivial functional dependency. While the result of the condensing procedure is not unique in general (it depends on the functional dependency chosen at each step), Ibaraki, Kogan and Makino have shown that it is unique when $\Sigma$ is a Horn CNF formula or more generally a q-Horn CNF formula (given as such or by its corresponding envelope), and that the condensing process can be achieved in polynomial time in such a case. In [34], the authors considered the problem of computing all the minimal functional dependencies which hold in $\Sigma$. Among other things, they showed that there exists an incrementally polynomial algorithm for achieving this goal when $\Sigma$ is a Horn CNF formula, or more generally, a q-Horn CNF formula, while the problem is equivalent to the problem of dualizing a positive theory when $\Sigma$ is equivalent to a Horn CNF formula (resp. q-Horn CNF formula) but is given by the Horn (resp. q-Horn) envelope of its models.

A major difference with our present work is that Ibaraki, Kogan and Makino mainly focused on Horn and q-Horn formulas, while our results are mainly about *(unconstrained) propositional formulas*. Actually, the few results from [22,15,34] which are related to unconstrained propositional formulas have been exhaustively listed in Sections 3.1, 4.1, and 4.3. Some of our results generalize their results (e.g., our Theorem 26 gives more tractable classes for the (mini-

mal) definability problem than just the Horn or q-Horn one), and some other results complete them (e.g., the results presented in Section 4.3 – about the computation of explicit definitions – address the general case and, again, give other tractable classes for this issue than just the Horn or q-Horn one).

### *8.2 Strongest necessary and weakest sufficient conditions*

The work by Lin [11] is concerned with strongest necessary and weakest sufficient conditions.

In Section 3.2, we have shown close connections between definability and strongest necessary (SNC) / weakest sufficient conditions (WSC). While Proposition 2 from [11] characterizes definability in terms of WSC and SNC, we have shown how to characterize all the *definitions* of $y$ on $X$ in $\Sigma$ in terms of SNC and WSC.

First, Theorem 10 shows how SNC and WSC can be characterized using the notion of projection. It extends Theorem 2 from [11] by relaxing the assumption that $y \in \mathrm{Var}(\Sigma)$ and $y \notin X$, and focus on the logically strongest (resp. weakest) SNC $\mathrm{Proj}(\Sigma \wedge y, X)$ (resp. WSC) $\neg\mathrm{Proj}(\Sigma \wedge \neg y, X)$ of $y$ on $X$ w.r.t. $\Sigma$, up to logical equivalence. Then Theorem 8 shows that $\Phi_X$ is a definition of $y$ on $X$ in $\Sigma$ if and only if $\mathrm{Proj}(\Sigma \wedge y, X) \models \Phi_X \models \neg\mathrm{Proj}(\Sigma \wedge \neg y, X)$.

### *8.3 Definability in other logical settings*

Since Padoa and Beth, there has been a considerable amount of work on definability and interpolation in various classes of logics. A logic is said to has the projective Beth definability property if and only if implicit definability equals explicit definability. As pointed out in [5], implicit definability being a semantical (model-theoretic) concept whereas explicit definability is a syntactic (proof-theoretic) concept, to say that both forms of definability coincide in a given logic is a good indication that there is a good balance between syntax and semantics in the logic. There are two main streams of works: definability in fragments of first-order logic, and definability in propositional modal logics. We briefly discuss these two streams of work, by pointing to some of the most relevant references. A comprehensive review can be found in Chapter 2 of [5]. See also the excellent book [49] for connections with second order quantification in many propositional logics.

Definability in predicate logic starts with Padoa's work and, later on, Beth's theorem. The latter [4] shows that first-order logic has the definability property. The question is now whether given fragments of first-order logic still have

the property. For instance, the $k$-variable fragment of first-order logic fails to satisfy it [**?**,**?**], as well as in a large number of first-order modal logics (e.g., [**?**]), while it holds in intuitionistic predicate logic [**?**,**?**]. Definability in fragments of first-order logic also has an impact on the database community (e.g., [50].)

As for propositional logics, a large number thereof satisfy the definability property. For instance, Kreisel [**?**] proves that this is the case for any logic between classical propositional logic and intuitionistic propositional logic. A large number of works has concentrated on modal logics (e.g. [**?**,51,52]), and especially (and more AI related) on description logics [53,54] (the latter paper focuses on computational issues; especially, they give bounds on the size of explicit definitions).

Our work, focusing on classical propositional logic, is not of the same nature as most of the abovementioned works (apart of the works about description logics). Our focus is on the computational issues of the problems related to definability, as well as on the applications to artificial intelligence problems.

## 9 Conclusion

This paper is centered on definability in standard propositional logic and reports a number of results issued from our computation-oriented investigation of this notion. Especially, we gave several characterization results, and complexity results for definability and related notions. We also presented a number of applications of such results in several AI problems, including hypothesis discrimination, reasoning about actions and automated reasoning.

This work calls for a number of perspectives. First, an alternative way of characterizing logical definability (and related notions) would consist in expressing it in epistemic logic, remarking that for any propositional formula $\Sigma \in \mathrm{PROP}_{\mathrm{PS}}$, $X \subseteq \mathrm{PS}$, and $y \in \mathrm{PS}$, we have $X \sqsubseteq_\Sigma y$ holds if and only if $(\mathbf{K}\Sigma \wedge \bigwedge_{x \in X}(\mathbf{K}x \vee \mathbf{K}\neg x)) \Rightarrow (\mathbf{K}y \vee \mathbf{K}\neg y)$ is a theorem of $\mathsf{S5}$. From this we can derive characterizations for other notions, such as minimal defining families, undefinable variables, etc. The results stated in the paper would then be easily reformulated (in different terms) in this setting.

Second, the notion of definability studied in this paper is rather strong, and it would be worth to relaxing the notion of definability. Doing so is not easy if the background knowledge $\Sigma$ is still expressed by a mere propositional formula; now, if instead of $\Sigma$ we have a probability distribution over $\Omega$, expressed succinctly for instance by a Bayesian network $N$ whose induced probability distribution is $p_N$, then definability becomes itself a probabilistic, decision-

theoretic notion: defining $\delta(N, X, y) = \sum_{\vec{x} \in \Omega_X} p_N(\vec{x}).\max(p_N(y|\vec{x}), p_N(\neg y|\vec{x}))$, then $\delta(N, X, y)$ can be interpreted as the prior probability of guessing the right value of $y$ after observing the values of variables in $X$, and is probably the most natural generalization of definability (obviously, $\Sigma$ defines $y$ in terms of $X$ in the usual way if and only if $\delta(N, X, y) = 1$). Thus, a natural decision problem in this setting would be: given a Bayesian network $N$, a set $X$ of variables, a variable $y$ and $\alpha \in [0, 1]$, determine whether $\delta(N, X, y) \geq \alpha$. Another notion of probabilistic definability arises when the probabilistic background knowledge is expressed by a set of constraints in probabilistic logic [55] – in this case we do not have a single probability distribution but a set of probability distributions, and the latter notion must be updated accordingly. A computational investigation of these probabilistic notions of definability is left for further research.

## Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments. The second author has been partly supported by the IUT de Lens, the Université d'Artois, the Région Nord/Pas-de-Calais, the IRCICA consortium and by the European Community FEDER Program.

## References

[1] J. Lang, P. Liberatore, P. Marquis, Conditional independence in propositional logic, Artificial Intelligence 141 (1-2) (2002) 79–121.

[2] J. Lang, P. Liberatore, P. Marquis, Propositional independence – formula-variable independence and forgetting, Journal of Artificial Intelligence Research 18 (2003) 391–443.

[3] W. Craig, Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory, Journal of Symbolic Logic 22 (1957) 269–285.

[4] E. Beth, On Padoa's method in the theory of definition, Indagationes mathematicae 15 (1953) 330–339.

[5] E. Hoogland, Definability and interpolation – model-theoretic investigations, Ph.D. thesis, Institute for Logic, Language and Computation, University of Amsterdam (2001).

[6] E. Boros, Y. Crama, P. Hammer, Polynomial-time inference of all valid implications for horn and related formulae, Annals of Mathematics and Artificial Intelligence 1 (1990) 21–32.

[7] A. Darwiche, Compiling devices into decomposable negation normal form, in: Proc. of IJCAI'99, 1999, pp. 284–289.

[8] P. Marquis, Consequence finding algorithms, Vol. 5 of Handbook on Defeasible Reasoning and Uncertainty Management Systems, Kluwer Academic Publisher, 2000, Ch. 2, pp. 41–145.

[9] A. Darwiche, Decomposable negation normal form, Journal of the Association for Computing Machinery 48 (4) (2001) 608–647.

[10] F. Lin, R. Reiter, Forget it!, in: Proc. of the AAAI Fall Symposium on Relevance, New Orleans, 1994, pp. 154–159.

[11] F. Lin, On the strongest necessary and weakest sufficient conditions, Artificial Intelligence 128 (2001) 143–159.

[12] M. Krom, The decision problem for formulas in prenex conjunctive normal form with binary disjunctions, Journal of Symbolic Logic 35 (1970) 210–216.

[13] A. Horn, On sentences which are true of direct unions of algebras, Journal of Symbolic Logic 16 (1951) 14–21.

[14] H. Lewis, Renaming a set of clauses as a horn set, Journal of the Association for Computing Machinery 25 (1978) 134–135.

[15] T. Ibaraki, A. Kogan, K. Makino, On functional dependencies in q-Horn theories, Artificial Intelligence 131 (1–2) (2001) 171–187.

[16] A. Blake, Canonical expressions in boolean algebra, Ph.D. thesis, University of Chicago, Chicago (IL) (1937).

[17] E. Boros, P. Hammer, X. Sun, Recognition of q-Horn formulae in linear time, Discrete Applied Mathematics 55 (1994) 1–13.

[18] P. Tison, Generalization of consensus theory and application to the minimization of boolean functions, IEEE Transactions on Electronic Computers EC–16 (1967) 446–456.

[19] C. H. Papadimitriou, Computational complexity, Addison–Wesley, 1994.

[20] W. Armstrong, Dependency structures of database relationships, in: Proc. of IFIP'74, 1974, pp. 580–583.

[21] R. Fagin, Functional dependencies in a relational database and propositional logic, IBM Journal of Research and Development 21 (1977) 534–544.

[22] T. Ibaraki, A. Kogan, K. Makino, Functional dependencies in Horn theories, Artificial Intelligence 108 (1–2) (1999) 1–30.

[23] T. Castell, Computation of prime implicates and prime implicants by a variant of the Davis and Putnam procedure, in: Proc. of ICTAI'96, IEEE Computer Society, Washington, DC, USA, 1996, p. 428.

[24] A. Padoa, Essai d'une théorie algébrique des nombres entiers, précédé d'une introduction logique à une théorie déductive quelconque, in: Bibliothèque du Congrès International de Philosophie, Paris, 1903, pp. 309–365.

[25] A. Darwiche, P. Marquis, A knowledge compilation map, Journal of Artificial Intelligence Research 17 (2002) 229–264.

[26] A. Darwiche, P. Marquis, A perspective on knowledge compilation, in: Proc. of IJCAI'01, 2001, pp. 175–182.

[27] G. Lakemeyer, A logical account of relevance, in: Proc. of IJCAI'95, 1995, pp. 853–859.

[28] S. Akers, Binary decision diagrams, IEEE Transactions on Computers C-27 (6) (1978) 509–516.

[29] R. Bryant, Graph-based algorithms for boolean function manipulation, IEEE Transactions on Computers 8 (C-35) 677–692.

[30] S. Coste-Marquis, D. Le Berre, F. Letombe, P. Marquis, Propositional fragments for knowledge compilation and quantified boolean formulae, in: Proc. of AAAI'05, 2005, pp. 288–293.

[31] H. Fargier, P. Marquis, On the use of partially ordered decision graphs in knowledge compilation and quantified boolean formulae, in: Proc. of AAAI'06, 2006.

[32] D. Mundici, Tautologies with a unique Craig interpolant, uniform vs. non-uniform complexity, Annals of Pure and Applied Logic 27 (1974) 265–273.

[33] R. Rymon, An SE-tree-based prime implicant generation algorithm, Annals of Mathematics and Artificial Intelligence 11 (1994) 329–349, special issue on model-based diagnosis.

[34] T. Ibaraki, A. Kogan, K. Makino, Inferring minimal functional dependencies in Horn and q-Horn theories, Annals of Mathematics and Artificial Intelligence 38 (2003) 233–255.

[35] R. Reiter, A theory of diagnosis from first principles, Artificial Intelligence 32 (1987) 57–95.

[36] P. Struss, Testing for the discrimination of diagnoses, in: Proc. of DX'94, 1994, pp. 312–320.

[37] S. McIlraith, Generating tests using abduction, in: Proc. of KR'94, 1994, pp. 449–460.

[38] J. Lang, Planning to discriminate diagnoses, in: Proc. of DX'97, 1997, pp. 135–139.

[39] R. Reiter, Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems, MIT Press, 2001.

[40] E. Giunchiglia, V. Lifschitz, An action language based on causal explanation: Preliminary report, in: Proc. of AAAI'98, 1998, pp. 623–630.

[41] J. Lang, F. Lin, P. Marquis, Causal theories of action: a computational core, in: Proc. of IJCAI'03, 2003, pp. 1073–1078.

[42] V. Lifschitz, Frames in the space of situations (research note), Artificial Intelligence 46 (1990) 365–376.

[43] S. Fahlman, A planning system for robot construction tasks, Artificial Intelligence 5 (1974) 1–49.

[44] A. J. Hu, D. L. Dill, Reducing BDD size by exploiting functional dependencies, in: Proc. of $30^{th}$ ACM/IEEE Design Automation Conference, 1993, pp. 266–271.

[45] H. Kautz, D. McAllester, B. Selman, Exploiting variable dependency in local search (abstract), in: Proc. of IJCAI'97 (poster session), 1997, p. 57.

[46] É. Grégoire, R. Ostrowski, B. Mazure, L. Sais, Automatic extraction of functional dependencies, in: Proc. of SAT'04 (Selected Papers), Vol. 3542 of Lecture Notes in Computer Science, Springer, 2005, pp. 122–132.

[47] D. Le Berre, Exploiting the real power of unit propagation lookahead, in: Proc. of SAT'01, Vol. 9 of Electronic Notes in Discrete Mathematics, Elsevier Science Publishers, 2001.

[48] E. Giunchiglia, M. Maratea, A. Tacchella, Dependent and independent variables in propositional satisfiability, in: Proc. of JELIA'02, 2002, pp. 296–307.

[49] S. Ghilardi, M. Zawadowski, Sheaves, games, and model completions, Vol. 14 of Trends in Logic—Studia Logica Library, Kluwer Academic Publishers, Dordrecht, 2002, a categorical approach to nonclassical propositional logics.

[50] L. Segoufin, V. Vianu, Views and queries: determinacy and rewriting, in: Proc. of PODS'05, 2005, pp. 49–60.

[51] A. Urquhart, Beth's definability theorem in relevant logics, in: E. Orlowska (Ed.), Logic at Work: Essays Dedicated to the Memory of Helena Rasiowa, Springer-Verlag, pp. 229–234.

[52] L. Maksimova, Definability and interpolation in non-classical logics., Studia Logica 82 (2) (2006) 271–291.

[53] F. Baader, W. Nutt, Basic Description Logics, The Description Logic Handbokk: Theory, Implementation and Applications, Cambridge University Press, 2003.

[54] B. ten Cate, W. Conradie, M. Marx, Y. Venema, Definitorially complete description logics., in: Proc. of KR'06, 2006, pp. 79–89.

[55] N. Nilsson, Probabilistic logic, Artificial Intelligence 28 (1987) 71–87.