Artificial Intelligence 275 (2019) 1–27

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Advanced SMT techniques for weighted model integration[☆]

Paolo Morettin[*], Andrea Passerini[*], Roberto Sebastiani[*]

DISI – Dipartimento di Ingegneria e Scienza dell'Informazione, Università degli Studi di Trento, Via Sommarive 5, I-38123 Povo, TN, Italy

ARTICLE INFO

Article history:
Received 5 March 2018
Received in revised form 20 November 2018
Accepted 8 April 2019
Available online 12 April 2019

Keywords:
Probabilistic inference
Satisfiability modulo theories
Weighted model counting
Weighted model integration

ABSTRACT

Weighted model integration (WMI) is a recent formalism generalizing weighted model counting (WMC) to run probabilistic inference over hybrid domains, characterized by both discrete and continuous variables and relationships between them. WMI is computationally very demanding as it requires to explicitly enumerate all possible truth assignments to be integrated over. Component caching strategies which proved extremely effective for WMC are difficult to apply in this formalism because of the tight coupling induced by the arithmetic constraints. In this paper we present a novel formulation of WMI, which allows to exploit the power of SMT-based predicate abstraction techniques in designing efficient inference procedures. A novel algorithm combines a strong reduction in the number of models to be integrated over with their efficient enumeration. Experimental results on synthetic and real-world data show drastic computational improvements over the original WMI formulation as well as existing alternatives for hybrid inference.

1. Introduction

The development of efficient exact and approximate algorithms for probabilistic inference is a long standing goal of artificial intelligence research. Whereas substantial progress has been made in dealing with purely discrete or purely continuous domains, probabilistic inference on *hybrid domains*, characterized by both discrete and continuous variables and their relationships, is still a challenge. Hybrid domains are however extremely common in real-world scenarios, from transport modeling and traffic forecasting [27] to probabilistic robotics [48] and cyber-physical systems [32].

Weighted model counting (WMC) is the task of computing the weighted sum of all satisfying assignments of a propositional formula, where weights are associated to models and are typically factorized into the product of weights of individual variables. In recent years, WMC was shown to be an effective solution for addressing probabilistic inference in a wide spectrum of formalisms [13,15,23,47]. Exact WMC solvers are based on knowledge compilation [17,37] or exhaustive DPLL search [41]. In both cases, substantial efficiency gains can be obtained by leveraging component caching techniques [3,40], in which the weighted model counts of disjoint components of a formula are cached once and reused whenever the formula is encountered again in the computation.

An inherent limitation of WMC is that it can only deal with discrete distributions. In order to overcome this restriction, weighted model integration (WMI) [7] was recently introduced as a formulation generalizing WMC to deal with hybrid domains. The formalism relies on satisfiability modulo theories (SMT) [5] technology, which allows to reason about the satisfiability of formulas involving e.g. linear constraints over integers or reals. WMI works by replacing the weighted sum of truth assignments satisfying a propositional formula with a sum of integrals over weight functions defined over the truth

[☆] A preliminary and much shorter version of this work was presented at IJCAI'17 [36].
[*] Corresponding authors.
E-mail addresses: paolo.morettin@unitn.it (P. Morettin), andrea.passerini@unitn.it (A. Passerini), roberto.sebastiani@unitn.it (R. Sebastiani).

https://doi.org/10.1016/j.artint.2019.04.003
0004-3702/© 2019 Elsevier B.V. All rights reserved.

assignments satisfying an SMT formula. Weight functions here play the role of (unnormalized) densities, whereas logic formulas in the model define the integration domain. WMI is computationally very demanding as it requires to explicitly enumerate all possible truth assignments to be integrated over. Component caching strategies which proved extremely effective for WMC are difficult to apply in this formalism because of the tight coupling induced by the arithmetic constraints.

In this paper, we elaborate on the notion of WMI and provide a refined formalization which allows to design more efficient inference procedures. In particular, our novel formulation:

- easily captures the previous definition;
- allows for defining arbitrary weight functions (involving e.g., arbitrary combinations of sums, products, if-then-elses, case-splits...) without requiring to specify them in terms of combinations of weights over literals;
- allows for exploiting state-of-the-art SMT-based techniques.

Building on the properties of this novel formulation, we devise an efficient algorithm combining a strong reduction in the number of models to be generated and integrated over, with efficiency in enumerating these models. The key ingredient is the use of SMT-based *predicate abstraction* techniques [11,26,30] to efficiently and effectively generate the set of models needed to compute the exact integral. Our experimental evaluation confirms that the approach is drastically faster than existing alternatives over synthetic and real-world problems, and that both aspects contribute to the gain.

The paper is organized as follows. We start by introducing relevant background, including the original definition of WMI (§2). We then proceed by identifying its shortcomings and proposing a revised and general formulation, proving some essential properties (§3). In order to show the expressiveness and features of the novel definition, we propose a case study coming from a real-world application (§4). Then we show how to efficiently compute WMI by exploiting SMT-based predicate abstraction techniques (§5). In §6 we summarize some related work. In §7 we present an empirical evaluation on synthetic and real-world problems comparing the algorithm proposed in §5 with existing alternatives. In §8 we conclude, and describe some future research directions.

2. Background

We provide the necessary background notions on SMT, AllSMT and Predicate Abstraction (§2.1), and on WMC and the original formulation of WMI (§2.2).

2.1. SMT, AllSMT and predicate abstraction

We assume the reader is familiar with the basic syntax, semantics and results of propositional and first-order logics. We adopt some terminology and concepts from Satisfiability Modulo Theories (SMT), which we briefly summarize below (see [5,44] for details).

Our context is that of SMT on quantifier-free formulas in the theory of *linear arithmetic over the reals*, $\mathcal{LRA}$. $\mathbb{R}$ denotes the set of real values and $\mathbb{B} \stackrel{\text{def}}{=} \{\top, \bot\}$ the set of Boolean values. $\mathcal{LRA}$ formulas are combinations by means of the standard Boolean operators $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ of atomic propositions $A_i \in \mathbb{B}$ (aka Boolean atoms/variables) and of $\mathcal{LRA}$ atomic formulas (aka $\mathcal{LRA}$ atoms) in the form $(\sum_i c_i x_i \bowtie c)$, s.t. x_i are variables in $\mathbb{R}$, c_i are rational values and $\bowtie \in \{=, \neq, \geq, \leq, >, <\}$, with their usual semantics. A Boolean/$\mathcal{LRA}$ *literal* is either a Boolean/$\mathcal{LRA}$ atom (a *positive literal*) or its negation (a *negative literal*).

"$\models_{\mathcal{LRA}}$" denotes entailment in $\mathcal{LRA}$ (e.g. $(x \geq 2) \models_{\mathcal{LRA}} (x \geq 1)$), whereas "$\models_{\mathbb{B}}$" denotes tautological entailment (e.g. $A_1 \wedge (x \geq 2) \models_{\mathbb{B}} (A_1 \vee (x \leq 1)) \wedge (\neg A_1 \vee (x \geq 2))$). Note that $\models_{\mathbb{B}}$ is strictly stronger than $\models_{\mathcal{LRA}}$, that is, if $\varphi_1 \models_{\mathbb{B}} \varphi_2$ then $\varphi_1 \models_{\mathcal{LRA}} \varphi_2$, but not vice versa. We say that φ_1 implies φ_2 in $\mathcal{LRA}$, written $\varphi_1 \Rightarrow_{\mathcal{LRA}} \varphi_2$, iff $\varphi_1 \models_{\mathcal{LRA}} \varphi_2$. We say that φ_1, φ_2 are $\mathcal{LRA}$-equivalent, written $\varphi_1 \Leftrightarrow_{\mathcal{LRA}} \varphi_2$, iff both $\varphi_1 \models_{\mathcal{LRA}} \varphi_2$ and $\varphi_2 \models_{\mathcal{LRA}} \varphi_1$.

We frequently use the following formula and term abbreviations, all written in the form "$[\![\langle expression \rangle]\!]$", denoting the $\mathcal{LRA}$-encoding of $\langle expression \rangle$, the latter being some mathematical concept which is not an $\mathcal{LRA}$ formula or term. Let t, t_i be $\mathcal{LRA}$ terms, φ, φ_i be $\mathcal{LRA}$ formulas, and $I = [l, u]$ be some interval; then we use "$[\![t \in I]\!]$" as a shortcut for the formula $(t \geq l) \wedge (t \leq u)$, possibly with ">" or "<" if some end of the interval is open[1]; we use "$[\![\mathsf{OneOf}\{\varphi_1, \ldots, \varphi_n\}]\!]$" as a shortcut for the formula $(\bigvee_{i=1}^n \varphi_i) \wedge \bigwedge_{1 \leq i < j \leq n} \neg(\varphi_i \wedge \varphi_j)$, i.e., exactly one φ_i holds; we use "$[\![\mathsf{If}\ \varphi\ \mathsf{Then}\ t_1\ \mathsf{Else}\ t_2]\!]$" to represent an if-then-else expression, that is, $t = [\![\mathsf{If}\ \varphi\ \mathsf{Then}\ t_1\ \mathsf{Else}\ t_2]\!]$ is equivalent to $(\varphi \rightarrow (t = t_1)) \wedge (\neg\varphi \rightarrow (t = t_2))$; we use "$[\![\mathsf{Case}\ \varphi_1 : t_1;\ \varphi_2 : t_2;\ \ldots]\!]$" to generalize the if-then-else to the case of multiple mutually-exclusive and exhaustive conditions, that is, $t = [\![\mathsf{Case}\ \varphi_1 : t_1;\ \varphi_2 : t_2;\ \ldots]\!]$ is equivalent to $\bigwedge_i (\varphi_i \rightarrow (t = t_i))$, under the assumption that the conditions φ_i are exhaustive –that is, $\models_{\mathcal{LRA}} \bigvee_i \varphi_i$– and mutually exclusive – that is, $\varphi_i \wedge \varphi_j \models_{\mathcal{LRA}} \bot$ for every i, j.[2]

Given a set of $\mathcal{LRA}$ formulas $\mathbf{\Psi} \stackrel{\text{def}}{=} \{\psi_1, \ldots, \psi_K\}$, we call a *total [resp. partial] truth assignment* μ for $\mathbf{\Psi}$ any total [resp. partial] map from $\mathbf{\Psi}$ to $\mathbb{B}$. With a little abuse of notation, we represent μ either as a set or a conjunction:

[1] We often represent strict inequalities "$(t_1 < t_2)$" as negated non-strict ones "$\neg(t_1 \geq t_2)$", see [44].

[2] Note that the mutual exclusion guarantees that the semantics of $[\![\mathsf{Case}\ \varphi_1 : t_1;\ \varphi_2 : t_2;\ \ldots]\!]$ is not sequential, and it does not depend on the order of the conditions $\varphi_1, \varphi_2, \ldots$.

$$\mu \stackrel{\text{def}}{=} \{\psi \mid \psi \in \mathbf{\Psi},\ \mu(\psi) = \top\} \cup \{\neg\psi \mid \psi \in \mathbf{\Psi},\ \mu(\psi) = \bot\},$$

$$\mu \stackrel{\text{def}}{=} \bigwedge_{\psi \in \mathbf{\Psi}, \mu(\psi)=\top} \quad \psi \wedge \bigwedge_{\psi \in \mathbf{\Psi}, \mu(\psi)=\bot} \neg\psi,$$

and we write "$\psi_i \in \mu_1$" and "$\mu_1 \subseteq \mu_2$" as if μ_1, μ_2 were represented as sets (i.e. we write "$\psi_1 \wedge \neg\psi_2 \subseteq \psi_1 \wedge \neg\psi_2 \wedge \psi_3$" meaning "$\{\psi_1, \neg\psi_2\} \subseteq \{\psi_1, \neg\psi_2, \psi_3\}$"). In the latter case, we say that μ_1 is a *sub-assignment* of μ_2, and that μ_2 is a *super-assignment* of μ_1. We denote by $\mathbb{B}^K$ the set of all total truth assignments over $\mathbf{\Psi}$.

Let $\mathbf{x} \stackrel{\text{def}}{=} \{x_1, \dots, x_N\} \in \mathbb{R}^N$ and $\mathbf{A} \stackrel{\text{def}}{=} \{A_1, \dots, A_M\} \in \mathbb{B}^M$ for some N and M. Consider a generic $\mathcal{LRA}$ formula φ on (subsets of[3]) $\mathbf{x}$ and $\mathbf{A}$, and let $\mathbf{\Psi} \stackrel{\text{def}}{=} Atoms(\varphi)$, i.e. the set of propositional and $\mathcal{LRA}$ atoms occurring in φ. Given a truth assignment μ for $Atoms(\varphi)$, we denote by $\mu^{\mathbf{A}}$ and $\mu^{\mathcal{LRA}}$ its two components on the Boolean atoms in $\mathbf{A}$ and on the $\mathcal{LRA}$ atoms respectively, so that $\mu = \mu^{\mathbf{A}} \wedge \mu^{\mathcal{LRA}}$. (E.g., if $\mu = A_1 \wedge \neg A_2 \wedge (x \geq 1) \wedge \neg(x \geq 3)$, then $\mu^{\mathbf{A}} = A_1 \wedge \neg A_2$ and $\mu^{\mathcal{LRA}} = (x \geq 1) \wedge \neg(x \geq 3)$). Importantly, and unlike with pure propositional logic, μ can be $\mathcal{LRA}$-unsatisfiable due to its $\mu^{\mathcal{LRA}}$ component (e.g. $\mu \stackrel{\text{def}}{=} \neg A_1 \wedge (x_1 + x_2 = 3) \wedge \neg(x_1 + x_2 \geq 2)$). A (partial) truth assignment μ *propositionally satisfies* φ iff $\mu \models_{\mathbb{B}} \varphi$. The SMT problem for φ in $\mathcal{LRA}$ is the problem of checking the existence of a $\mathcal{LRA}$-satisfiable assignment μ s.t. $\mu \models_{\mathbb{B}} \varphi$.

We denote by $\mathcal{TTA}(\varphi) \stackrel{\text{def}}{=} \{\mu_1, \dots, \mu_j, \dots\}$ the set of all $\mathcal{LRA}$-satisfiable *total* truth assignments μ_j on $Atoms(\varphi)$ propositionally satisfying φ. $\mathcal{TTA}(\varphi)$ is unique. The disjunction of the μ_j's in $\mathcal{TTA}(\varphi)$ is $\mathcal{LRA}$-equivalent to φ (see e.g. [44]):

$$\varphi \Leftrightarrow_{\mathcal{LRA}} \bigvee_{\mu_j \in \mathcal{TTA}(\varphi)} \mu_j. \tag{1}$$

We denote by $\mathcal{TA}(\varphi) \stackrel{\text{def}}{=} \{\mu_1, \dots, \mu_j, \dots\}$ any set of $\mathcal{LRA}$-satisfiable *partial* truth assignments μ_j propositionally satisfying φ, s.t. (i) every total truth assignment $\eta \in \mathcal{TTA}(\varphi)$ is a super-assignment of one μ_j in $\mathcal{TA}(\varphi)$, and (ii) every pair μ_i, μ_j assigns opposite truth values to at least one element, i.e., $\mu_i \wedge \mu_j \models_{\mathbb{B}} \bot$ (hence $\mu_i \wedge \mu_j \models_{\mathcal{LRA}} \bot$). $\mathcal{TA}(\varphi)$ is not unique, and $\mathcal{TTA}(\varphi)$ is a particular case of $\mathcal{TA}(\varphi)$. The disjunction of the μ_j's in $\mathcal{TA}(\varphi)$ is $\mathcal{LRA}$-equivalent to φ (see e.g. [44]):

$$\varphi \Leftrightarrow_{\mathcal{LRA}} \bigvee_{\mu_j \in \mathcal{TA}(\varphi)} \mu_j. \tag{2}$$

Intuitively, a set $\mathcal{TA}(\varphi)$ is a (much) more compact representation of $\mathcal{TTA}(\varphi)$, because every partial assignment μ_i in $\mathcal{TA}(\varphi)$ implicitly represents up to $2^{|Atoms(\varphi)|-|\mu_i|}$ of its total super-assignments in $\mathcal{TTA}(\varphi)$. (It is "up to" rather than "exactly" because not all the super-assignments of μ_i are necessarily $\mathcal{LRA}$-satisfiable.)

The *AllSMT problem* for φ in $\mathcal{LRA}$, hereafter denoted as AllSMT(φ), is the problem of enumerating one set $\mathcal{TTA}(\varphi)$—or $\mathcal{TA}(\varphi)$ if and only if we allow for partial assignments—matching the above definition.[4]

Example 1. Consider the $\mathcal{LRA}$-formula $\varphi \stackrel{\text{def}}{=} (x \leq 0) \vee (x \geq 1)$. Then $\mathcal{TTA}(\varphi) \stackrel{\text{def}}{=} \{(x \leq 0) \wedge \neg(x \geq 1), \neg(x \leq 0) \wedge (x \geq 1)\}$. Note that ($\mathcal{TTA}(\varphi)$ does not contain $(x \leq 0) \wedge (x \geq 1)$ because the latter is not $\mathcal{LRA}$-satisfiable.) The other admissible $\mathcal{TA}(\varphi)$'s are: $\{(x \leq 0), \neg(x \leq 0) \wedge (x \geq 1)\}$ and $\{(x \geq 1), (x \leq 0) \wedge \neg(x \geq 1)\}$. Instead, the set of unary truth assignments $\{(x \leq 0), (x \geq 1)\}$ is not an admissible $\mathcal{TA}(\varphi)$ because there is no atom to which they assign different truth values, so that $(x \leq 0) \wedge (x \geq 1) \not\models_{\mathbb{B}} \bot$. $\diamond$

We recall a notion of *predicate abstraction*, which is widely used in formal verification for automatically computing finite-state abstractions for systems with potentially infinite state spaces [11,26,30].

Definition 1. Let $\varphi(\mathbf{x}, \mathbf{A})$ be a $\mathcal{LRA}$-formula on $\mathbf{x}$ and $\mathbf{A}$; let $\mathbf{\Psi} \stackrel{\text{def}}{=} \{\psi_1, \dots, \psi_K\}$ be a set of $\mathcal{LRA}$-formulas over $\mathbf{x}$ and $\mathbf{A}$, and $\mathbf{B} \stackrel{\text{def}}{=} \{B_1, \dots, B_K\}$ be a set of fresh atomic propositions s.t. $\mathbf{A} \cap \mathbf{B} = \emptyset$. Then we call a **Predicate Abstraction** of φ with respect to $\mathbf{\Psi}$ on $\mathbf{B}$, namely $\mathsf{PredAbs}_{[\varphi,\mathbf{\Psi}]}(\mathbf{B})$, any propositional formula equivalent to the formula[5]

$$\exists \mathbf{A} \exists \mathbf{x}. \left(\varphi(\mathbf{x}, \mathbf{A}) \wedge \bigwedge_{k=1}^{K} (B_k \leftrightarrow \psi_k(\mathbf{x}, \mathbf{A})) \right). \tag{3}$$

[3] Note that it is not necessarily the case that all elements of $\mathbf{x}$ and $\mathbf{A}$ actually occur in φ.

[4] We remark that in the SMT literature the word "AllSMT" is used with slightly distinct meanings, as either $\mathcal{TTA}(\varphi)$, or as $\mathcal{TA}(\varphi)$, or even as predicate abstraction as in Definition 1 [16,30,38].

[5] In principle we should existentially quantify only the variables x_n and A_m which actually occur in either φ or $\mathbf{\Psi}$. Nevertheless, this is not restrictive since $\exists A_n.\varphi \Leftrightarrow_{\mathcal{LRA}} \varphi$ if A_n does not occur in φ; the same holds for x_i.

We define $\mathsf{PredAbs}_{[\varphi]}(\mathbf{\Psi}) \stackrel{\text{def}}{=} \mathsf{PredAbs}_{[\varphi,\mathbf{\Psi}]}(\mathbf{B})[\mathbf{B} \leftarrow \mathbf{\Psi}]$, that is, the $\mathcal{LRA}$-formula obtained from the propositional formula $\mathsf{PredAbs}_{[\varphi,\mathbf{\Psi}]}(\mathbf{B})$ by substituting each B_k with its corresponding ψ_k.

Note that in Definition 1 the formulas ψ_k are neither necessarily atomic, nor necessarily sub-formulas of φ. $\mathsf{PredAbs}_{[\varphi,\mathbf{\Psi}]}(\mathbf{B})$ defines an equivalence class of propositional formulas over $\mathbf{B}$, i.e., (3) may represent many syntactically-different albeit logically-equivalent propositional formulas.

Example 2. Consider $\mathbf{A} \stackrel{\text{def}}{=} \{A_1\}$, $\mathbf{x} \stackrel{\text{def}}{=} \{x_1, x_2\}$, $\varphi \stackrel{\text{def}}{=} A_1 \wedge (x_1 + x_2 > 12)$, $\psi_1 \stackrel{\text{def}}{=} (x_1 + x_2 = 2)$, $\psi_2 \stackrel{\text{def}}{=} (x_1 - x_2 < 10)$. Then we have that:

$$\mathsf{PredAbs}_{[\varphi,\mathbf{\Psi}]}(\mathbf{B}) \stackrel{\text{def}}{=} \exists A_1.\exists x_1 x_2. \left(\begin{array}{l} A_1 \wedge (x_1 + x_2 > 12) \wedge \\ (B_1 \leftrightarrow (x_1 + x_2 = 2)) \wedge \\ (B_2 \leftrightarrow (x_1 - x_2 < 10)) \end{array} \right) \tag{4}$$

$$= (\neg B_1 \wedge \neg B_2) \vee (\neg B_1 \wedge B_2) \tag{5}$$

$$= \neg B_1. \tag{6}$$

$$\mathsf{PredAbs}_{[\varphi]}(\mathbf{\Psi}) = \begin{array}{l} (\neg(x_1 + x_2 = 2) \wedge \neg(x_1 - x_2 < 10)) \vee \\ (\neg(x_1 + x_2 = 2) \wedge (x_1 - x_2 < 10)) \end{array} \tag{7}$$

$$= \neg(x_1 + x_2 = 2). \tag{8}$$

Note that both the equivalent propositional formulas (5) and (6) match the definition of $\mathsf{PredAbs}_{[\varphi,\mathbf{\Psi}]}(\mathbf{B})$: (5) is built as the disjunction of *total* assignments on $\mathbf{B}$, whereas (6) is built as the disjunction of *partial* ones s.t.:

$$\mathcal{TTA}(\mathsf{PredAbs}_{[\varphi,\mathbf{\Psi}]}(\mathbf{B})) = \{(\neg B_1 \wedge \neg B_2), (\neg B_1 \wedge B_2)\} \tag{9}$$

$$\mathcal{TA}(\mathsf{PredAbs}_{[\varphi,\mathbf{\Psi}]}(\mathbf{B})) = \{(\neg B_1)\} \tag{10}$$

$$\mathcal{TTA}(\mathsf{PredAbs}_{[\varphi]}(\mathbf{\Psi})) = \left\{ \begin{array}{l} (\neg(x_1 + x_2 = 2) \wedge \neg(x_1 - x_2 < 10)), \\ (\neg(x_1 + x_2 = 2) \wedge (x_1 - x_2 < 10)) \end{array} \right\} \tag{11}$$

$$\mathcal{TA}(\mathsf{PredAbs}_{[\varphi]}(\mathbf{\Psi})) = \{(\neg(x_1 + x_2 = 2))\} \tag{12}$$

Note also that the other two total assignments, $B_1 \wedge B_2$ and $B_1 \wedge \neg B_2$, do not occur in (5) because they would both force the formula to be $\mathcal{LRA}$-unsatisfiable because of the contradictory conjuncts $(x_1 + x_2 > 12) \wedge (x_1 + x_2 = 2)$. ◇

Intuitively, given a set of "relevant" formulas $\mathbf{\Psi}$—typically referred to as "predicates", from which the name "predicate abstraction" follows [26]— and a set of Boolean atoms $\mathbf{B}$ labeling them, $\mathsf{PredAbs}_{[\varphi,\mathbf{\Psi}]}(\mathbf{B})$ is a Boolean formula whose satisfying assignments represent the truth assignments to (the propositional atoms $\mathbf{B}$ labeling) the predicates in $\mathbf{\Psi}$ which are consistent with φ, so that $\mathsf{PredAbs}_{[\varphi]}(\mathbf{\Psi})$ is a $\mathcal{LRA}$-formula characterizing the truth assignments to the predicates in $\mathbf{\Psi}$ which share some solution with φ. Thus, $\mathsf{PredAbs}_{[\varphi,\mathbf{\Psi}]}(\mathbf{B})$ "abstracts" (aka "projects") φ onto the Boolean space $\mathcal{B}^K$ by considering the truth values of (the propositional atoms $\mathbf{B}$ labeling) the predicates in $\mathbf{\Psi}$ as the only relevant information to consider in the resulting propositional formula, and by existentially-quantifying away all other information.

Note that in the typical usage of predicate abstraction the number of "relevant" predicates $|\mathbf{\Psi}|$ is smaller or even much smaller than $|Atoms(\varphi)|$, so that to drastically reduce the size of the search space for the SMT solver, from (up to) $2^{|Atoms(\varphi)|}$ to (up to) $2^{|\mathbf{\Psi}|}$, so that $2^{|\mathbf{\Psi}|} \ll 2^{|Atoms(\varphi)|}$.

We highlight a few facts about predicate abstraction.

(*a*) If $\mathbf{\Psi}$ is $\mathbf{A}$, then $\mathsf{PredAbs}_{[\varphi]}(\mathbf{A})$ reduces to $\exists \mathbf{x}.\varphi(\mathbf{x}, \mathbf{A})$.[6]
(*b*) If $\mathbf{\Psi}$ is $Atoms(\varphi)$, then $\mathsf{PredAbs}_{[\varphi]}(Atoms(\varphi))$ is equivalent to φ.[7] Therefore $\mathcal{TTA}(\mathsf{PredAbs}_{[\varphi]}(Atoms(\varphi)))$ [resp. $\mathcal{TA}(\mathsf{PredAbs}_{[\varphi]}(Atoms(\varphi)))$ iff we admit partial assignments] is the same as $\mathcal{TTA}(\varphi)$ [resp. $\mathcal{TA}(\varphi)$], that is, $\mathsf{AllSMT}(\varphi)$.
(*c*) If $\mathbf{\Psi} \subset Atoms(\varphi)$ and $|\mathbf{\Psi}|$ is significantly smaller than $|Atoms(\varphi)|$, then typically $|\mathcal{TTA}(\mathsf{PredAbs}_{[\varphi]}(\mathbf{\Psi}))| \ll |\mathcal{TTA}(\varphi)|$.[8]

Very effective SMT-based techniques for computing $\mathcal{TTA}(\mathsf{PredAbs}_{[\varphi,\mathbf{\Psi}]}(\mathbf{B}))$ —and hence for $\mathcal{TTA}(\mathsf{PredAbs}_{[\varphi]}(\mathbf{\Psi}))$— have been proposed in the literature (e.g. [11,30]) and are implemented in modern SMT solvers like MathSAT5 [16]. Very importantly for our purposes, these techniques work by iteratively producing a set of propositional truth assignments on $\mathbf{B}$, which

[6] In fact, $\mathsf{PredAbs}_{[\varphi]}(\mathbf{A}) = \mathsf{PredAbs}_{[\varphi,\mathbf{A}]}(\mathbf{B})[\mathbf{B} \leftarrow \mathbf{A}] = \{\exists \mathbf{A} \exists \mathbf{x}.(\varphi(\mathbf{x}, \mathbf{A}) \wedge \bigwedge_{k=1}^{K}(B_k \leftrightarrow A_k)\}\ [\mathbf{B} \leftarrow \mathbf{A}]) = \{\exists \mathbf{x}.\varphi(\mathbf{x}, \mathbf{B})\}\ [\mathbf{B} \leftarrow \mathbf{A}] = \exists \mathbf{x}.\varphi(\mathbf{x}, \mathbf{A})$.

[7] In fact, $\mathsf{PredAbs}_{[\varphi]}(Atoms(\varphi)) = \mathsf{PredAbs}_{[\varphi, Atoms(\varphi)]}(\mathbf{B})[\mathbf{B} \leftarrow Atoms(\varphi)] = \{\exists \mathbf{A} \exists \mathbf{x}.(\varphi(\mathbf{x}, \mathbf{A}) \wedge \bigwedge_{\psi_k \in Atoms(\varphi)}(B_k \leftrightarrow \psi_k)\}\ [\mathbf{B} \leftarrow Atoms(\varphi)])$, which is equivalent to $\varphi(\mathbf{x}, \mathbf{A})$.

[8] In fact, for every assignment $\mu \in \mathcal{TTA}(\mathsf{PredAbs}_{[\varphi]}(\mathbf{\Psi}))$ there exist from 1 to $2^{|Atoms(\varphi)|-|\mathbf{\Psi}|}$ assignments $\mu \wedge \mu' \in \mathcal{TTA}(\varphi)$.

are then disjoined as in (2). Therefore, MATHSAT5 computes $\mathsf{PredAbs}_{[\varphi,\mathbf{\Psi}]}(\mathbf{B})$ directly in the form $\mathcal{TTA}(\mathsf{PredAbs}_{[\varphi,\mathbf{\Psi}]}(\mathbf{B}))$ [resp. $\mathsf{PredAbs}_{[\varphi]}(\mathbf{\Psi})$ in the form $\mathcal{TTA}(\mathsf{PredAbs}_{[\varphi]}(\mathbf{\Psi}))$].

In particular MATHSAT5, on demand, can produce either the set of *total* assignments on $\mathbf{B}$, $\mathcal{TTA}(\mathsf{PredAbs}_{[\varphi,\mathbf{\Psi}]}(\mathbf{B}))$, or a set of *partial* ones, $\mathcal{TA}(\mathsf{PredAbs}_{[\varphi,\mathbf{\Psi}]}(\mathbf{B}))$, by means of assignment-reduction techniques. To this extent, we recall that MATHSAT5 does not provide an explicit command for AllSMT(φ); rather, the user has to set explicitly the definitions $\bigwedge_k (B_k \leftrightarrow \psi_k)$ and specify the (sub)set of $\mathbf{B}$ of interest.[9]

2.2. Weighted model counting and weighted model integration

WMC is the task of computing the weighted sum of all satisfying assignments of a propositional formula, with weights typically factorized as a product of weights over literals.

Definition 2 *(Weighted Model Count).* Let φ be a propositional formula on the set of Boolean atoms $\mathbf{A} \overset{\text{def}}{=} \{A_1, \ldots, A_M\}$ and let w be a function associating a non-negative weight to each literal whose atom occurs in φ. Then the **Weighted Model Count** of φ is defined as:

$$\mathsf{WMC}(\varphi, w) = \sum_{\mu \in \mathcal{TTA}(\varphi)} \prod_{\ell \in \mu} w(\ell). \tag{13}$$

Example 3. Let $\varphi \overset{\text{def}}{=} ((A_1 \wedge A_2) \leftrightarrow B_1) \wedge ((A_1 \wedge \neg A_2) \leftrightarrow B_2) \wedge ((\neg A_1 \wedge A_2) \leftrightarrow B_3) \wedge ((\neg A_1 \wedge \neg A_2) \leftrightarrow B_4)$. Let the weights of all literals be equal to 1 apart from $w(A_1) = w(\neg A_1) = 0.5$, $w(B_1) = 0.8$, $w(B_2) = 0.2$, $w(B_3) = 0.4$, $w(B_4) = 0.6$. Then $\mathcal{TTA}(\varphi \wedge A_2)$ has two assignments $\mu_1 \overset{\text{def}}{=} A_1 \wedge A_2 \wedge B_1 \wedge \neg B_2 \wedge \neg B_3 \wedge \neg B_4$ and $\mu_2 \overset{\text{def}}{=} \neg A_1 \wedge A_2 \wedge \neg B_1 \wedge \neg B_2 \wedge B_3 \wedge \neg B_4$, so that the weighted model count of $\varphi \wedge A_2$ is computed as:

$$\begin{aligned}\mathsf{WMC}(\varphi \wedge A_2, w) = & \sum_{\mu \in \mathcal{TTA}(\varphi \wedge A_2)} \prod_{\ell \in \mu} w(\ell) \\ = & \, w(\ \ A_1) \cdot w(\ \ A_2) \cdot w(\ \ B_1) \cdot w(\neg B_2) \cdot w(\neg B_3) \cdot w(\neg B_4) \\ + & \, w(\neg A_1) \cdot w(\ \ A_2) \cdot w(\neg B_1) \cdot w(\neg B_2) \cdot w(\ \ B_3) \cdot w(\neg B_4) \\ = & \, 0.5 \cdot 1 \cdot 0.8 \cdot 1 \cdot 1 \cdot 1 + 0.5 \cdot 1 \cdot 1 \cdot 1 \cdot 0.4 \cdot 1 = 0.6.\end{aligned}$$

Note that the (φ, w) pair here models a joint probability distribution $P(A_1, A_2) = P(A_2|A_1)P(A_1)$ over binary variables A_1 and A_2. $P(A_1)$ is encoded in the weights associated to A_1 and $P(A_2|A_1)$ is encoded in the weights associated to the auxiliary variables B_1, B_2, B_3 and B_4, each labeling one of the four truth value combinations. $\mathsf{WMC}(\varphi \wedge A_2, w)$ here computes the marginal probability of $A_2 = \top$, i.e. $P(A_2 = \top) = \sum_{A_1 = \{\bot, \top\}} P(A_2 = \top|A_1)P(A_1)$. $\diamond$

Remark 1. Note that the above definition of WMC(φ,w) implicitly assumes that the Boolean domain is specified by means of some given background set of atomic propositions (namely $\mathbf{A}$) such that $Atoms(\varphi) \subseteq \mathbf{A}$ for every formula φ considered, and that always truth assignments assign all atoms in $\mathbf{A}$, including those which do not occur in $Atoms(\varphi)$. (If not so, and we considered instead assignments only to the atoms occurring in φ, we could have situations where the WMC of two logically-equivalent formulas φ_1 and φ_2 under the same weight function $w$produces different values: $\varphi_1 \Leftrightarrow \varphi_2$ and $\mathsf{WMC}(\varphi_1, w) \neq \mathsf{WMC}(\varphi_2, w)$.[10]) In our framework we will make domain assumptions explicit (§3 and §5).

WMI generalizes WMC to hybrid domains. Following is the original definition of WMI [7], hereafter denoted with $\mathsf{WMI}_{\mathsf{old}}$, which serves as a starting point for our revised formulation. The definition assumes $\mathcal{LRA}$ formulas, for which efficient solvers exist, albeit the concept could in principle accommodate other theories over continuous domains.

Definition 3 *(Weighted Model Integral).* Let φ be a $\mathcal{LRA}$ formula on the set of $\mathcal{LRA}$ variables $\mathbf{x} \overset{\text{def}}{=} \{x_1, \ldots, x_N\}$ and Boolean atoms $\mathbf{A} \overset{\text{def}}{=} \{A_1, \ldots, A_M\}$. Let w be a function associating an expression (possibly constant) over $\mathbf{x}$ to each literal whose atom occurs in φ. The **Weighted Model Integral** of φ is defined as:

$$\mathsf{WMI}_{\mathsf{old}}(\varphi, w) = \sum_{\mu \in \mathcal{TTA}(\varphi)} \int_{\mu^{\mathcal{LRA}}} \prod_{\ell \in \mu} w(\ell) \, d\mathbf{x}. \tag{14}$$

[9] See http://mathsat.fbk.eu/smt2examples.html. See also Footnote 4.

[10] E.g., if $\varphi_1 \overset{\text{def}}{=} A_1$, $\varphi_2 \overset{\text{def}}{=} (A_1 \vee A_2) \wedge (A_1 \vee \neg A_2)$, and $w(l) = 1$ for every literal l, then $\mathsf{WMC}(\varphi_1, w) = 1$ and $\mathsf{WMC}(\varphi_2, w) = 2$.

Example 4. Consider the $\mathcal{LRA}$-formula

$$\varphi \overset{\text{def}}{=} (A_2 \to ((1 \le x) \wedge (x \le 3))) \wedge (A_3 \to (\neg(x \le 3) \wedge (x \le 5)))$$
$$\wedge\ (A_1 \leftrightarrow (\neg A_2 \wedge \neg A_3)) \wedge (1 \le x) \wedge (x \le 5).$$

Let the weights of all literals be 1 except for $w(A_1) = 0.1$, $w(A_2) = 0.25 \cdot x - 0.25$ and $w(A_3) = 1.25 - 0.25 \cdot x$. First, it is easy to see that

$$\mathcal{TTA}(\varphi) = \left\{ \begin{array}{lllll} A_1 \wedge \neg A_2 \wedge \neg A_3 \wedge & (1 \le x) \wedge & (x \le 5) \wedge & (x \le 3), \\ A_1 \wedge \neg A_2 \wedge \neg A_3 \wedge & (1 \le x) \wedge & (x \le 5) \wedge \neg(x \le 3), \\ \neg A_1 \wedge \quad A_2 \wedge \neg A_3 \wedge & (1 \le x) \wedge & (x \le 5) \wedge & (x \le 3), \\ \neg A_1 \wedge \neg A_2 \wedge \quad A_3 \wedge & (1 \le x) \wedge & (x \le 5) \wedge \neg(x \le 3) \end{array} \right\}.$$

Then we have:[11]

$$\begin{aligned} \mathsf{WMI}_{\mathsf{old}}(\varphi, w) = & \int_{(1 \le x) \wedge (x \le 5) \wedge (x \le 3)} w(A_1)\, dx + \int_{(1 \le x) \wedge (x \le 5) \wedge \neg(x \le 3)} w(A_1)\, dx \\ & + \int_{(1 \le x) \wedge (x \le 5) \wedge (x \le 3)} w(A_2)\, dx + \int_{(1 \le x) \wedge (x \le 5) \wedge \neg(x \le 3)} w(A_3)\, dx \\ & = \int_{[1,3]} 0.1\, dx + \int_{(3,5]} 0.1\, dx \\ & + \int_{[1,3]} 0.25 \cdot x - 0.25\, dx + \int_{(3,5]} 1.25 - 0.25 \cdot x\, dx \\ & = [0.1 \cdot x]_1^3 + [0.1 \cdot x]_3^5 \\ & + \Big[0.125 \cdot x^2 - 0.25 \cdot x\Big]_1^3 + \Big[1.25 \cdot x - 0.125 \cdot x^2\Big]_3^5 \\ & = 0.3 - 0.1 + 0.5 - 0.3 + 0.375 + 0.125 + 3.125 - 2.625 = 1.4 \end{aligned}$$

This example models an unnormalized distribution over x ranging from one to five, which is uniform if A_1 is true, and is modeled as a triangular distribution with mode at $x = 3$ otherwise. $\diamond$

Remark 2. The main motivation behind the introduction of WMI was that of enabling probabilistic inference in hybrid domains. In that scenario, it was implicitly assumed that a pair $\langle \varphi, w \rangle$ defines the unnormalized distribution, and that any additional formula representing evidence or queries does not introduce any additional Boolean or continuous variables with respect to those in φ, and that the weight of any literal of atoms not in φ has a constant weight of 1.

Example 5. Let φ and w be as in the previous example. Suppose we are interested in computing the probability that $x \le 2$ (query), given the unnormalized distribution represented by the $\langle \varphi, w \rangle$ pair and the information that $A_1 = \bot$ (evidence). This probability can be computed as:

$$P_{\langle \varphi, w \rangle}(x \le 2 | A_1 = \bot) = \frac{\mathsf{WMI}_{\mathsf{old}}(\varphi \wedge \neg A_1 \wedge (x \le 2), w)}{\mathsf{WMI}_{\mathsf{old}}(\varphi \wedge \neg A_1, w)} = \frac{0.125}{1.0} = 0.125$$

because:

$$\begin{aligned} \mathsf{WMI}_{\mathsf{old}}(\varphi \wedge \neg A_1, w) = & \int_{(1 \le x) \wedge (x \le 5) \wedge (x \le 3)} w(A_2)\, dx + \int_{(1 \le x) \wedge (x \le 5) \wedge \neg(x \le 3)} w(A_3)\, dx \\ & = \int_{[1,3]} 0.25 \cdot x - 0.25\, dx + \int_{(3,5]} 1.25 - 0.25 \cdot x\, dx \\ & = \Big[0.125 \cdot x^2 - 0.25 \cdot x\Big]_1^3 + \Big[1.25 \cdot x - 0.125 \cdot x^2\Big]_3^5 \\ & = 0.375 + 0.125 + 3.125 - 2.625 = 1.0 \end{aligned}$$

[11] For better readability, here and henceforth we drop from products the weights $w(l)$ which are equal to 1.

$$
\begin{aligned}
\mathsf{WMI}_{\mathsf{old}}(\varphi \wedge \neg A_1 \wedge (x \le 2), w) &= \int_{(1 \le x) \wedge (x \le 5) \wedge (x \le 3) \wedge (x \le 2)} w(A_2)\, \mathrm{d}x \\
&= \int_{[1,2]} 0.25 \cdot x - 0.25\, \mathrm{d}x \\
&= \Big[0.125 \cdot x^2 - 0.25 \cdot x\Big]_1^2 \\
&= 0.0 + 0.125 = 0.125 \quad \diamond
\end{aligned}
$$

3. Weighted model integration, revisited

Definition 3 is a very direct and intuitive generalization of WMC to the hybrid case. However, it is very abstract and directly turning it into a computational procedure, as was done in all previous implementations [7–9], can result in major inefficiencies, even if using non-naive techniques like AllSMT, as will be shown in our experimental evaluation. In the following we present a revised formulation of WMI that:

- easily captures the previous definition;
- decouples the specification of the formula and of the weight function from that of the variables on which WMI is to be computed, removing all implicit assumptions of the original formulation (see Remark 2);
- is not restricted to weight functions in the form of products of weights *over literals*, but allows for much more general forms (§3.3). This gives a remarkable flexibility in designing efficient encodings for hybrid domains, as shown with the case study on modeling journey times on road networks (§4);
- makes it easier to develop algorithms for WMI computation that fully exploit the potential advantage of advanced SMT techniques like predicate abstraction, as will be shown in §5.

We start by introducing a revisited and very general definition of WMI, starting from the Boolean-free case (§3.1) and then covering the general case (§3.2), and finally we describe a very general class of weight functions s.t. WMI is computable (§3.3).

3.1. Basic case: WMI without atomic propositions

We investigate first the simple case where no atomic proposition comes into play. Let $\mathbf{x} \overset{\text{def}}{=} \{x_1, \ldots, x_N\} \in \mathbb{R}^N$. We consider a *generic* total weight function $w(\mathbf{x})$ s.t. $w : \mathbb{R}^N \longmapsto \mathbb{R}^+$, and $\mathcal{LRA}$ formulas $\varphi(\mathbf{x})$ s.t. $\varphi : \mathbb{R}^N \longmapsto \mathbb{B}$.

Definition 4. Assume φ does not contain atomic propositions and $w : \mathbb{R}^N \longmapsto \mathbb{R}^+$. Then we define the **Weighted Model Integral** of w over φ on $\mathbf{x}$ as:

$$
\mathsf{WMI}_{\mathsf{nb}}(\varphi, w|\mathbf{x}) \overset{\text{def}}{=} \int_{\varphi(\mathbf{x})} w(\mathbf{x})\, \mathrm{d}\mathbf{x}, \tag{15}
$$

"$_{\mathsf{nb}}$" meaning "no-Booleans", that is, as the integral of $w(\mathbf{x})$ over the set $\{\mathbf{x} \mid \varphi(\mathbf{x})\ \textit{is true}\}$.

The following property of $\mathsf{WMI}_{\mathsf{nb}}(\varphi, w|\mathbf{x})$ derives directly from Definition 4.[12]

Property 1. Given $\mathbf{x}$, w, φ, and φ' as above,

a. if φ is $\mathcal{LRA}$-unsatisfiable, then $\mathsf{WMI}_{\mathsf{nb}}(\varphi, w|\mathbf{x}) = 0$.
b. if $\varphi \Rightarrow_{\mathcal{LRA}} \varphi'$, then $\mathsf{WMI}_{\mathsf{nb}}(\varphi, w|\mathbf{x}) \le \mathsf{WMI}_{\mathsf{nb}}(\varphi', w|\mathbf{x})$
c. if $\varphi \Leftrightarrow_{\mathcal{LRA}} \varphi'$, then $\mathsf{WMI}_{\mathsf{nb}}(\varphi, w|\mathbf{x}) = \mathsf{WMI}_{\mathsf{nb}}(\varphi', w|\mathbf{x})$
d. for every $\mathcal{LRA}$-formula $\psi(\mathbf{x})$,

$$
\mathsf{WMI}_{\mathsf{nb}}(\varphi, w|\mathbf{x}) = \mathsf{WMI}_{\mathsf{nb}}(\varphi \wedge \psi, w|\mathbf{x}) + \mathsf{WMI}_{\mathsf{nb}}(\varphi \wedge \neg\psi, w|\mathbf{x}).
$$

[12] We understand that Property 1 holds also for $\mathsf{WMI}_{\mathsf{old}}$, provided the implicit assumptions of Remark 2.

Remark 3. We stress the fact that in the definition of $\mathsf{WMI_{nb}}(\varphi, w|\mathbf{x})$ specifying the domain "$|\mathbf{x}$" is of primary importance. In fact, even if some x_n does not occur in φ,[13] $\mathsf{WMI_{nb}}(\varphi, w|\mathbf{x}) = \int_{\mathbb{R}} \mathsf{WMI_{nb}}(\varphi, w|\mathbf{x}\backslash\{x_n\})\ dx_n \neq \mathsf{WMI_{nb}}(\varphi, w|\mathbf{x}\backslash\{x_n\})$. "$|\mathbf{x}$" defines the dimensions of the space we are integrating on, which must be stated. (E.g., integrating on volumes differs from integrating on surfaces.)

The above definition of $\mathsf{WMI_{nb}}(\varphi, w|\mathbf{x})$ does not (yet) imply a practical technique for computing it, because "$\int_{\varphi(\mathbf{x})} \dots d\mathbf{x}$" cannot be directly computed by most integration solvers, which typically can handle only *conjunctions* of $\mathcal{LRA}$-literals, not arbitrary Boolean combinations of them. To cope with this fact, we need decomposing φ into conjunctions of $\mathcal{LRA}$-literals. This is where $\mathcal{TTA}()$, $\mathcal{TA}()$ and the SMT-based techniques to compute them come into play, as described in the following.

The following property of $\mathsf{WMI_{nb}}(\varphi, w|\mathbf{x})$ derives directly from the definition of $\mathcal{TTA}(\varphi)$ and $\mathcal{TA}(\varphi)$ and from (2), by recalling that the domains of the assignments $\mu^{\mathcal{LRA}}$ in $\mathcal{TTA}(\varphi)$ and $\mathcal{TA}(\varphi)$ are pairwise disjoint.

Proposition 1. *Given* $\mathbf{x}$, $w(\mathbf{x})$, $\varphi(\mathbf{x})$, $\mathcal{TTA}(\varphi)$ *and* $\mathcal{TA}(\varphi)$ *as above,*

$$\mathsf{WMI_{nb}}(\varphi, w|\mathbf{x}) = \sum_{\mu^{\mathcal{LRA}} \in \mathcal{TTA}(\varphi)} \mathsf{WMI_{nb}}(\mu^{\mathcal{LRA}}, w|\mathbf{x}) \tag{16}$$

$$= \sum_{\mu^{\mathcal{LRA}} \in \mathcal{TA}(\varphi)} \mathsf{WMI_{nb}}(\mu^{\mathcal{LRA}}, w|\mathbf{x}). \tag{17}$$

Proof. We prove only (17), because $\mathcal{TTA}(\varphi)$ is a subcase of $\mathcal{TA}(\varphi)$:

$$\begin{aligned} & \mathsf{WMI_{nb}}(\varphi, w|\mathbf{x}) \\ \{\text{by (2)}\} = {} & \mathsf{WMI_{nb}}\Big(\bigvee_{\mu^{\mathcal{LRA}} \in \mathcal{TA}(\varphi)} \mu^{\mathcal{LRA}}, w|\mathbf{x}\Big) \\ \{\text{disjoint domains of the } \mu^{\mathcal{LRA}}\text{'s}\} = {} & \sum_{\mu^{\mathcal{LRA}} \in \mathcal{TA}(\varphi)} \mathsf{WMI_{nb}}(\mu^{\mathcal{LRA}}, w|\mathbf{x}). \qquad \square \end{aligned}$$

Importantly, the $\mu^{\mathcal{LRA}}$s in both (16) and (17) are conjunctions of literals, so that $\mathsf{WMI_{nb}}(\mu^{\mathcal{LRA}}, w|\mathbf{x})$ is computable by standard integration solvers under reasonable hypotheses (e.g., w is a polynomial) which will be discussed in §4. Note that if (i) w is in the form of products of weights over a set of literals as in Definition 3 and (ii) φ is defined only over such literals and (iii) φ contains no Boolean atom, then (16) corresponds to (14).

3.2. General case: WMI with atomic propositions

We investigate now the general case, where atomic propositions come into play and both $w$and φ depend also on them. Let $\mathbf{A} \overset{\text{def}}{=} \{A_1, \dots, A_M\} \in \mathbb{B}^M$. We consider thus a generic total weight function $w(\mathbf{x}, \mathbf{A})$ s.t. $w : \mathbb{R}^N \times \mathbb{B}^M \longmapsto \mathbb{R}^+$, and $\mathcal{LRA}$ formulas $\varphi(\mathbf{x}, \mathbf{A})$ s.t. $\varphi : \mathbb{R}^N \times \mathbb{B}^M \longmapsto \mathbb{B}$.

In what follows, $\mu^{\mathbf{A}}$ denotes a total truth assignment on $\mathbf{A}$, $\varphi_{[\mu^{\mathbf{A}}]}(\mathbf{x})$ denotes (any formula equivalent to) the formula obtained from φ by substituting every Boolean value A_i with its truth value in $\mu^{\mathbf{A}}$, and $w_{[\mu^{\mathbf{A}}]}(\mathbf{x})$ is w computed on $\mathbf{x}$ and on the truth values of $\mu^{\mathbf{A}}$. Thus, $\varphi_{[\mu^{\mathbf{A}}]} : \mathbb{R}^N \longmapsto \mathbb{B}$ and $w_{[\mu^{\mathbf{A}}]} : \mathbb{R}^N \longmapsto \mathbb{R}^+$.

Definition 5. Given $\mathbf{x}$, $\mathbf{A}$, the **Weighted Model Integral** of w over φ is defined as follows:

$$\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A}) \overset{\text{def}}{=} \sum_{\mu^{\mathbf{A}} \in \mathbb{B}^M} \mathsf{WMI_{nb}}(\varphi_{[\mu^{\mathbf{A}}]}, w_{[\mu^{\mathbf{A}}]}|\mathbf{x}), \tag{18}$$

where the $\mu^{\mathbf{A}}$'s are all total truth assignments on $\mathbf{A}$.

Example 6. Let $\varphi \overset{\text{def}}{=} (A \leftrightarrow (x \geq 0)) \wedge (x \geq -1) \wedge (x \leq 1)$, and $w(x, A) \overset{\text{def}}{=} [\![\mathsf{If}\ A\ \mathsf{Then}\ x\ \mathsf{Else}\ -x]\!]$. If $\mu^{\mathbf{A}} \overset{\text{def}}{=} \{(\neg A)\}$, then $\varphi_{[\mu^{\mathbf{A}}]} = \neg(x \geq 0) \wedge (x \geq -1) \wedge (x \leq 1)$ and $w_{[\mu^{\mathbf{A}}]} = -x$. Note that $\varphi_{[\mu^{\mathbf{A}}]}$ can be simplified into the equivalent formula $\neg(x \geq 0) \wedge (x \geq -1)$. Similarly, if $\mu^{\mathbf{A}} \overset{\text{def}}{=} \{(A)\}$, then $\varphi_{[\mu^{\mathbf{A}}]}$ can be simplified into $(x \geq 0) \wedge (x \leq 1)$ and $w_{[\mu^{\mathbf{A}}]} = x$. Thus,

[13] It may be the case that x_n does not occur in φ even though $w$depends on x_n: e.g., $w$may be a Gaussian on x_n, so that no restriction on the domain of x_n is expressed by φ.

$$\begin{aligned}\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A}) &\overset{\text{def}}{=} \mathsf{WMI}_{\mathsf{nb}}(\varphi_{[\{\neg A\}]}, w_{[\{\neg A\}]}|x) \;+\; \mathsf{WMI}_{\mathsf{nb}}(\varphi_{[\{A\}]}, w_{[\{A\}]}|x)\\ &= \int_{[-1,0)} -x\,\mathrm{d}x \;+\; \int_{[0,1]} x\,\mathrm{d}x\\ &= \frac{1}{2}+\frac{1}{2}=1. \quad\diamond\end{aligned}$$

Note that in Definition 5 the truth assignments $\mu^{\mathbf{A}}$ of practical interest are only those for which $\varphi_{[\mu^{\mathbf{A}}]}$ is $\mathcal{LRA}$-satisfiable, because for the others $\mathsf{WMI}_{\mathsf{nb}}(\varphi_{[\mu^{\mathbf{A}}]}, w_{[\mu^{\mathbf{A}}]}|\mathbf{x}) = 0$ by Property 1.a. We address this issue in §5.

The following property of $\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A})$ derives directly from Definition 5, by applying Property 1 to $w_{[\mu^{\mathbf{A}}]}$, $\varphi_{[\mu^{\mathbf{A}}]}$, $\varphi'_{[\mu^{\mathbf{A}}]}$, $(\varphi\wedge\psi)_{[\mu^{\mathbf{A}}]}$, and $(\varphi\wedge\neg\psi)_{[\mu^{\mathbf{A}}]}$, for every $\mu^{\mathbf{A}}$.

Property 2. Given $\mathbf{x}$, $\mathbf{A}$, w, φ, and φ' as above,

a. if φ is $\mathcal{LRA}$-unsatisfiable, then $\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A}) = 0$.
b. if $\varphi \Rightarrow_{\mathcal{LRA}} \varphi'$, then $\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A}) \le \mathsf{WMI}(\varphi', w|\mathbf{x}, \mathbf{A})$
c. if $\varphi \Leftrightarrow_{\mathcal{LRA}} \varphi'$, then $\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A}) = \mathsf{WMI}(\varphi', w|\mathbf{x}, \mathbf{A})$
d. for every $\mathcal{LRA}$-formula $\psi(\mathbf{x}, \mathbf{A})$,

$$\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A}) = \mathsf{WMI}(\varphi\wedge\psi, w|\mathbf{x}, \mathbf{A}) + \mathsf{WMI}(\varphi\wedge\neg\psi, w|\mathbf{x}, \mathbf{A}).$$

Remark 4. As with Remark 3, in $\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A})$, specifying not only "$|\mathbf{x}$", but also "$|\mathbf{x}, \mathbf{A}$" is of primary importance. In fact, even if some of the A_m does not occur in φ,

$$\begin{aligned}&\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A})\\ &= \mathsf{WMI}(\varphi, w_{[\{A_m\}]}|\mathbf{x}, \mathbf{A}\backslash\{A_m\}) + \mathsf{WMI}(\varphi, w_{[\{\neg A_m\}]}|\mathbf{x}, \mathbf{A}\backslash\{\neg A_m\})\\ &\ne \mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A}\backslash\{A_m\}).\end{aligned}$$

To this extent, hereafter and if not explicitly specified otherwise, we implicitly assume w.l.o.g. that $\mathbf{A}$ and φ are such that each Boolean atom in $\mathbf{A}$ occurs in φ. (If this were not the case, we could rewrite φ into the equivalent formula $\varphi \wedge \bigvee_k (A_k \vee \neg A_k)$, s.t. the A_k's are the atoms in $\mathbf{A}$ not occurring in φ.) Consequently, each truth assignment in $\mathcal{TTA}(\varphi)$ assigns every atom in $\mathbf{A}$. We make the same assumption w.l.o.g. for the formula $\exists\mathbf{x}.\varphi$.

The following properties of $\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A})$ derive from the definition of $\mathcal{TTA}(...)$.

Proposition 2. *Given* $\mathbf{x}$, $\mathbf{A}$, *w(*$\mathbf{x}$,$\mathbf{A}$*)*, $\varphi(\mathbf{x}, \mathbf{A})$ *and* $\mathcal{TTA}(\varphi)$ *as above, we have that:*

$$\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A}) = \sum_{\mu^{\mathbf{A}}\wedge\mu^{\mathcal{LRA}}\in\mathcal{TTA}(\varphi)} \mathsf{WMI}_{\mathsf{nb}}(\mu^{\mathcal{LRA}}, w_{[\mu^{\mathbf{A}}]}|\mathbf{x}) \tag{19}$$

Proof. By applying (16) to Definition 5 we have that:

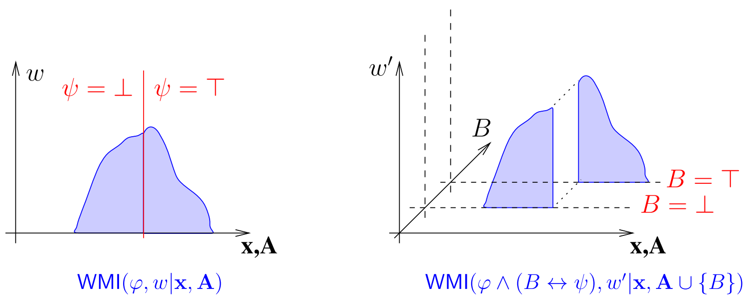
$$\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A}) = \sum_{\mu^{\mathbf{A}}\in\mathbb{B}^M} \;\sum_{\mu^{\mathcal{LRA}}\in\mathcal{TTA}(\varphi_{[\mu^{\mathbf{A}}]})} \mathsf{WMI}_{\mathsf{nb}}(\mu^{\mathcal{LRA}}, w_{[\mu^{\mathbf{A}}]}|\mathbf{x}). \tag{20}$$

In order to pass from (20) to (19), consider $\mu^{\mathbf{A}} \wedge \mu^{\mathcal{LRA}}$ s.t. $\mu^{\mathbf{A}} \in \mathbb{B}^M$ and $\mu^{\mathcal{LRA}} \in \mathcal{TTA}(\varphi_{[\mu^{\mathbf{A}}]})$. By construction $\mu^{\mathbf{A}} \wedge \mu^{\mathcal{LRA}} \models_{\mathbb{B}} \varphi$ (otherwise $\mu^{\mathcal{LRA}} \notin \mathcal{TTA}(\varphi_{[\mu^{\mathbf{A}}]})$).

If $\mu^{\mathbf{A}} \wedge \mu^{\mathcal{LRA}} \notin \mathcal{TTA}(\varphi)$, then $\mu^{\mathbf{A}} \wedge \mu^{\mathcal{LRA}}$ is not $\mathcal{LRA}$-satisfiable by the definition of $\mathcal{TTA}(\varphi)$, so that $\mathsf{WMI}_{\mathsf{nb}}(\mu^{\mathcal{LRA}}, w_{[\mu^{\mathbf{A}}]}|\mathbf{x}) = 0$. Hence (20) equals (19). $\square$

Proposition 3. *Given* $\mathbf{x}$, $\mathbf{A}$, *w(*$\mathbf{x}$,$\mathbf{A}$*)*, $\varphi(\mathbf{x}, \mathbf{A})$ *and* $\mathcal{TTA}(\varphi)$ *as above, we have that:*

$$\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A}) = \sum_{\mu^{\mathbf{A}}\in\mathcal{TTA}(\exists\mathbf{x}.\varphi)} \mathsf{WMI}_{\mathsf{nb}}(\varphi_{[\mu^{\mathbf{A}}]}, w_{[\mu^{\mathbf{A}}]}|\mathbf{x}) \tag{21}$$



w
$\psi = \bot$
$\psi = \top$
x,A
$\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A})$
w'
B
$B = \top$
$B = \bot$
x,A
$\mathsf{WMI}(\varphi \wedge (B \leftrightarrow \psi), w'|\mathbf{x}, \mathbf{A} \cup \{B\})$

Fig. 1. Graphical representation of Lemma 1.

Proof. Compare (18) with (21). Let $\mu^{\mathbf{A}} \in \mathbb{B}^M$ s.t. $\mu^{\mathbf{A}} \notin \mathcal{TTA}(\exists \mathbf{x}.\varphi)$. Then $\varphi_{[\mu^{\mathbf{A}}]} \Leftrightarrow_{\mathcal{LRA}} \bot$, so that $\mathsf{WMI}_{\mathsf{nb}}(\varphi_{[\mu^{\mathbf{A}}]}, w_{[\mu^{\mathbf{A}}]}|\mathbf{x}) = 0$. Hence (18) equals (21). $\square$

Predicate abstraction in Definition 1 and other forms of frequently-used formula manipulations require the introduction of fresh propositions B "labelling" sub-formulas ψ. The next result shows that this does not affect the value of WMI.

Lemma 1. *Let* $\mathbf{x}$, $\mathbf{A}$, w, *and* φ *be as in Definition 5; let* $\psi(\mathbf{x}, \mathbf{A})$ *be some* $\mathcal{LRA}$*-formula; let* $\varphi' \stackrel{\text{def}}{=} \varphi \wedge (B \leftrightarrow \psi)$, *where* $B \notin \mathbf{A}$*; let* w' *extend* w *s.t.* $w'(\mathbf{x}, \mathbf{A} \cup \{B\}) = w(\mathbf{x}, \mathbf{A})$ *for every* $\mathbf{x}$, $\mathbf{A}$ *and* B*. Then we have that*

$$\mathsf{WMI}(\varphi', w'|\mathbf{x}, \mathbf{A} \cup \{B\}) = \mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A}). \tag{22}$$

Proof. We note that $\varphi'_{[\mu^{\mathbf{A}} \wedge B]} \Leftrightarrow_{\mathcal{LRA}} (\varphi \wedge \psi)_{[\mu^{\mathbf{A}}]}$ and that $\varphi'_{[\mu^{\mathbf{A}} \wedge \neg B]} \Leftrightarrow_{\mathcal{LRA}} (\varphi \wedge \neg\psi)_{[\mu^{\mathbf{A}}]}$. Also, we have that $w'_{[\mu^{\mathbf{A}} \wedge B]}(\mathbf{x}) = w'_{[\mu^{\mathbf{A}} \wedge \neg B]}(\mathbf{x}) = w_{[\mu^{\mathbf{A}}]}(\mathbf{x})$. Thus:

$$\begin{aligned}
&\mathsf{WMI}(\varphi', w'|\mathbf{x}, \mathbf{A} \cup \{B\}) \\
&= \sum_{\mu^{\mathbf{A}} \in \mathbb{B}^M} (\mathsf{WMI}_{\mathsf{nb}}(\varphi'_{[\mu^{\mathbf{A}} \wedge B]}, w'_{[\mu^{\mathbf{A}} \wedge B]}|\mathbf{x}) + \mathsf{WMI}_{\mathsf{nb}}(\varphi'_{[\mu^{\mathbf{A}} \wedge \neg B]}, w'_{[\mu^{\mathbf{A}} \wedge \neg B]}|\mathbf{x})) \\
&= \sum_{\mu^{\mathbf{A}} \in \mathbb{B}^M} (\mathsf{WMI}_{\mathsf{nb}}((\varphi \wedge \psi)_{[\mu^{\mathbf{A}}]}, w_{[\mu^{\mathbf{A}}]}|\mathbf{x}) + \mathsf{WMI}_{\mathsf{nb}}((\varphi \wedge \neg\psi)_{[\mu^{\mathbf{A}}]}, w_{[\mu^{\mathbf{A}}]}|\mathbf{x})) \\
&= \sum_{\mu^{\mathbf{A}} \in \mathbb{B}^M} (\mathsf{WMI}_{\mathsf{nb}}(\varphi_{[\mu^{\mathbf{A}}]} \wedge \psi_{[\mu^{\mathbf{A}}]}, w_{[\mu^{\mathbf{A}}]}|\mathbf{x}) + \mathsf{WMI}_{\mathsf{nb}}(\varphi_{[\mu^{\mathbf{A}}]} \wedge \neg\psi_{[\mu^{\mathbf{A}}]}, w_{[\mu^{\mathbf{A}}]}|\mathbf{x})) \\
&= \sum_{\mu^{\mathbf{A}} \in \mathbb{B}^M} \mathsf{WMI}_{\mathsf{nb}}(\varphi_{[\mu^{\mathbf{A}}]}, w_{[\mu^{\mathbf{A}}]}|\mathbf{x}) \\
&= \mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A}). \qquad \square
\end{aligned}$$

The intuitive meaning of Lemma 1 is represented in Fig. 1. (For graphical convenience, we abstract the whole space $\mathbf{x}, \mathbf{A}$ into only one horizontal dimension.) Suppose Fig. 1 (left) represents $\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A})$. The formula ψ cuts the space $\mathbf{x}, \mathbf{A}$, and thus $\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A})$, into two parts. In Fig. 1 (right) we add a new Boolean dimension B, and we represent $\mathsf{WMI}(\varphi \wedge (B \leftrightarrow \psi), w'|\mathbf{x}, \mathbf{A} \cup \{B\})$, which is split into the sum of two parts, for $B = \bot$ and $B = \top$, corresponding respectively to $\psi = \bot$ and $\psi = \top$. Thus $\mathsf{WMI}(\varphi'), w'|\mathbf{x}, \mathbf{A} \cup \{B\})$ is identical to the sum of the two pieces of $\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A})$, for $\psi = \bot$ and $\psi = \top$ respectively.

3.3. *Conditional weight functions*

We call a (non-minimal) *support* of a weight function $w(\mathbf{x}, \mathbf{A})$ any subset of $\mathbb{R}^N \times \mathbb{B}^M$ out of which $w(\mathbf{x}, \mathbf{A}) = 0$.[14] In many situations it is useful to provide explicitly the representation of a support of $w(\mathbf{x}, \mathbf{A})$ as a $\mathcal{LRA}$-formula $\chi(\mathbf{x}, \mathbf{A})$. (When this is not the case, then we implicitly set $\chi(\mathbf{x}, \mathbf{A}) \stackrel{\text{def}}{=} \top$.) For instance, it allows to cut part of the domain where a polynomial function is negative (see e.g. Example 7 below).

The following property follows trivially.

[14] Note that a support is not unique and it is not necessarily minimal with respect to $\subseteq$, that is, it may be the case that $w(\mathbf{x}, \mathbf{A}) = 0$ also if $\langle \mathbf{x}, \mathbf{A} \rangle$ is in the support. This definition allows to deal with cases in which the minimal support is not known or hard to characterize.

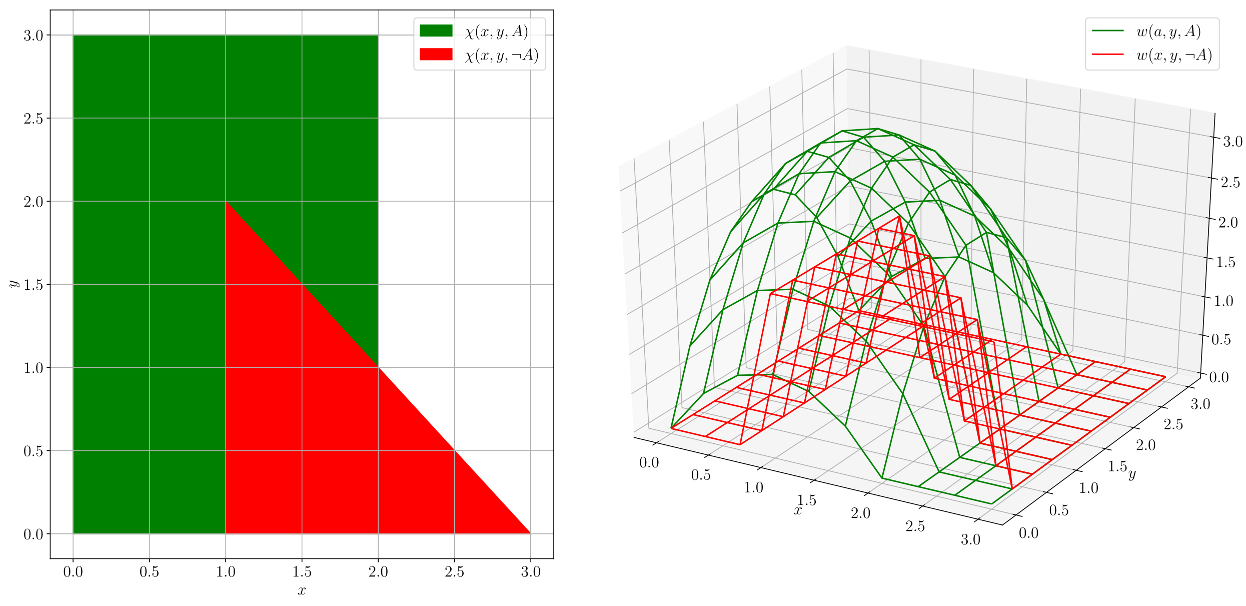


Fig. 2. Graphical representation of the support $\chi(\mathbf{x},\mathbf{A})$ (left) and of the weight $w(\mathbf{x},\mathbf{A})$ (right) in Example 7.

Property 3. Let φ and w be as above. If $\chi(\mathbf{x},\mathbf{A})$ is a $\mathcal{LRA}$-formula representing a support of w, then:

$$\mathsf{WMI}(\varphi, w|\mathbf{x},\mathbf{A}) = \mathsf{WMI}(\varphi \wedge \chi, w|\mathbf{x},\mathbf{A}). \tag{23}$$

Example 7. Let $\mathbf{x} \overset{\text{def}}{=} \{x, y\}$, $\mathbf{A} \overset{\text{def}}{=} \{A\}$,

$$\chi(\mathbf{x},\mathbf{A}) \overset{\text{def}}{=} (A \to [\![x \in [1,2]]\!]) \wedge (\neg A \to ([\![x \in [1,3]]\!] \wedge (x + y \le 3))) \wedge [\![y \in [1,3]]\!]$$

$$w(\mathbf{x},\mathbf{A}) \overset{\text{def}}{=} [\![\text{If } A \text{ Then } (-x^2 - y^2 + 2x + 3y) \text{ Else } (-2x - 2y + 6)]\!].$$

The support $\chi(\mathbf{x},\mathbf{A})$ defines two sub-spaces of $x \times y$ conditioned on the truth value of A, as depicted in Fig. 2 (left). The weight function $w(\mathbf{x},\mathbf{A})$ is an if-then-else on condition A such that its two branches are two polynomials which are forced to be zero outside the domain defined by the support formula, as depicted in Fig. 2 (right). Note that outside such domains the two polynomials may acquire negative values. ◇

We introduce a novel kind of weight function, which can be defined also in terms of $\mathcal{LRA}$ conditions. (See §4 for an example application). We consider first the generic class of functions $w(\mathbf{x})$, which we call *feasibly integrable on* $\mathcal{LRA}$ ($\mathsf{FI}^{\mathcal{LRA}}$), which contain no combinatorial component, and for which there exists some procedure able to compute $\mathsf{WMI}_{\mathsf{nb}}(\mu^{\mathcal{LRA}}, w|\mathbf{x})$ for every set of $\mathcal{LRA}$ literals on $\mathbf{x}$. (E.g., polynomials are $\mathsf{FI}^{\mathcal{LRA}}$ [4]). Such background procedure, which we use as a black-box, is the basic building block of our WMI calculations.

Definition 6. We call a total weight function $w(\mathbf{x},\mathbf{A})$, **feasibly integrable under $\mathcal{LRA}$ conditions** ($\mathsf{FIUC}^{\mathcal{LRA}}$) iff it can be described in terms of

- a support $\mathcal{LRA}$-formula $\chi(\mathbf{x},\mathbf{A})$ (if no support description is provided, then $\chi \overset{\text{def}}{=} \top$),
- a set $\boldsymbol{\Psi} \overset{\text{def}}{=} \{\psi_1(\mathbf{x},\mathbf{A}), \ldots, \psi_K(\mathbf{x},\mathbf{A})\}$ of *$\mathcal{LRA}$-formulas* (**conditions**),

in such a way that, for every total truth assignment $\mu^{\mathbf{A}}$ to $\mathbf{A}$ and for every total truth assignment $\mu^{\boldsymbol{\Psi}}$ to $\boldsymbol{\Psi}$, $w_{[\mu^{\mathbf{A}}\mu^{\boldsymbol{\Psi}}]}(\mathbf{x})$ is $\mathsf{FI}^{\mathcal{LRA}}$ in the domain given by the values of $\langle \mathbf{x},\mathbf{A}\rangle$ which satisfy $(\chi \wedge \mu^{\boldsymbol{\Psi}})_{[\mu^{\mathbf{A}}]}$. We denote such $\mathsf{FI}^{\mathcal{LRA}}$ functions by $f_{\mu^{\mathbf{A}}\mu^{\boldsymbol{\Psi}}}(\mathbf{x})$, s.t. for every $\langle \mu^{\mathbf{A}}, \mu^{\boldsymbol{\Psi}}\rangle$,

$$\text{if } \mu^{\mathbf{A}} \wedge \mu^{\boldsymbol{\Psi}} \text{ holds, then } w(\mathbf{x}) = f_{\mu^{\mathbf{A}}\mu^{\boldsymbol{\Psi}}}(\mathbf{x}). \tag{24}$$

(Note that a plain $\mathsf{FI}^{\mathcal{LRA}}$ weight function is a subcase in which $\chi \overset{\text{def}}{=} \top$ and $\boldsymbol{\Psi} \overset{\text{def}}{=} \emptyset$.)

A very relevant subcase of $\mathsf{FIUC}^{\mathcal{LRA}}$ functions, which we denote by $\mathsf{P}^{\mathcal{LRA}}$ ("Polynomials under $\mathcal{LRA}$ conditions"), is given by arbitrary combinations of polynomials with $\mathcal{LRA}$ conditions, such that each $f_{\mu^{\mathbf{A}}\mu^{\boldsymbol{\Psi}}}(\mathbf{x})$ in (24) is a polynomial whose value is non-negative in the domain defined by $\mu^{\mathbf{\Psi}}$. $\mathsf{P}^{\mathcal{LRA}}$ functions are $\mathsf{FIUC}^{\mathcal{LRA}}$ because polynomials can always be integrated on the domains given by a set of $\mathcal{LRA}$ literals [4]. The syntax of $\mathsf{P}^{\mathcal{LRA}}$ weight functions can be defined by the following grammar, expressed in standard Backus-Naur form:[15]

$$\begin{aligned} w ::= & \ c \mid x \mid \\ & -w \mid (w+w) \mid (w-w) \mid (w \cdot w) \mid \\ & [\![\text{If } \varphi \text{ Then } w \text{ Else } w]\!] \mid \\ & [\![\text{Case } \varphi : w;\ \varphi : w;\ \ldots]\!] \end{aligned} \tag{25}$$

$$\chi ::= \varphi \tag{26}$$

where c denotes a real value, x denotes a real variable, w denotes a $\mathsf{P}^{\mathcal{LRA}}$ weight function, φ denotes an $\mathcal{LRA}$ formula. We recall from §2.1 that the conditions φ in the case-terms must be mutually exclusive and exhaustive. (In practice, it suffices that the conditions are exhaustive *within the domain described by the support* χ, that is, for the term $[\![\text{Case } \varphi_1 : w_1;\ \varphi_2 : w_2;\ \ldots\ \varphi_k : w_k]\!]$ we must have that $\chi \models_{\mathcal{LRA}} \bigvee_{i=1}^{k} \varphi_i$.) In short, w can be any arbitrary combination of sums, products and $\mathcal{LRA}$-conditions.

Note that all the $\mathcal{LRA}$-formulas φ occurring as conditions in the weight function or in the support formula must be such to restrict the domains of the polynomials to areas where they are non-negative.

Example 8. Let $\mathbf{x} \stackrel{\text{def}}{=} \{x_1, x_2\}$, $\mathbf{A} \stackrel{\text{def}}{=} \{A\}$, and

$$\chi(\mathbf{x}, \mathbf{A}) \stackrel{\text{def}}{=} [\![x_1 \in [-1,1)]\!] \wedge [\![x_2 \in [-1,1)]\!] \wedge (A \leftrightarrow (x_2 \geq 0))$$
$$w(\mathbf{x}, \mathbf{A}) \stackrel{\text{def}}{=} [\![\text{If } x_1 \geq 0 \text{ Then } x_1^3 \text{ Else } -2x_1]\!] + [\![\text{If } A \text{ Then } 3x_2 \text{ Else } -x_2^5]\!].$$

w is $\mathsf{P}^{\mathcal{LRA}}$, and hence $\mathsf{FIUC}^{\mathcal{LRA}}$. In fact, its value depends on the combination of the truth values of the conditions $\mathbf{\Psi} \stackrel{\text{def}}{=} \{(x_1 \geq 0)\}$ and $\mathbf{A} \stackrel{\text{def}}{=} \{A\}$, so that:

$$\begin{aligned} f_{\{A,(x_1 \geq 0)\}} &= x_1^3 + 3x_2 && s.t.\ x_1 \in [0,1),\ x_2 \in [0,1), \\ f_{\{A,\neg(x_1 \geq 0)\}} &= -2x_1 + 3x_2, && s.t.\ x_1 \in [-1,0),\ x_2 \in [0,1), \\ f_{\{\neg A,(x_1 \geq 0)\}} &= x_1^3 - x_2^5, && s.t.\ x_1 \in [0,1),\ x_2 \in [-1,0), \\ f_{\{\neg A,\neg(x_1 \geq 0)\}} &= -2x_1 - x_2^5 && s.t.\ x_1 \in [-1,0),\ x_2 \in [-1,0). \end{aligned}$$

All four $f_{\mu^{\mathbf{A}}\mu^{\mathbf{\Psi}}}$ are positive polynomials in their respective domain and as such they can be integrated. ◇

Intuitively, Definition 6 captures the class of all the weight functions which can be described by means of arbitrary combinations of nested if-then-elses on conditions in $\mathbf{A}$ and $\mathbf{\Psi}$, s.t. each branch $\langle \mu^{\mathbf{A}}, \mu^{\mathbf{\Psi}} \rangle$ results into a $\mathsf{FI}^{\mathcal{LRA}}$ weight function. Each pair $\langle \mu^{\mathbf{A}}, \mu^{\mathbf{\Psi}} \rangle$ describes a portion of the domain of w, inside which w is the $\mathsf{FI}^{\mathcal{LRA}}$ function $f_{\mu^{\mathbf{A}}\mu^{\mathbf{\Psi}}}$.

The expressivity of $\mathsf{FIUC}^{\mathcal{LRA}}$ weight functions allows for the direct encoding of a number of probabilistic models or density estimators into the weighted model integration framework. For instance, it is possible to readily perform WMI inference on a trained *Mixed Sum-Product Network* (MSPN) [35] with piecewise-polynomial leaves, whose internal nodes are product or weighted sums. In this case, the circuit is already a $\mathsf{FIUC}^{\mathcal{LRA}}$ function and the support corresponds to the disjunction of the domains of its polynomial leaves. This procedure allows MSPNs to answer complex probability queries that couldn't normally be computed by their inference algorithms, such as those involving hard constraints. *Density Estimation Trees* (DETs) [39] are hybrid non-parametric[16] density estimators composed of internal univariate split nodes and constant leaves. Also in this case, the tree can be represented by a $\mathsf{FIUC}^{\mathcal{LRA}}$ function without additional processing and probabilistic queries can be performed using the estimator's bounding box as the support, thus enabling probabilistic queries on DETs.

Theorem 1. *Let $w(\mathbf{x}, \mathbf{A})$, $\mathbf{\Psi}$ and χ be as in Definition 6. Let $\mathbf{B} \stackrel{def}{=} \{B_1, \ldots, B_K\}$ be fresh propositional atoms and let $w^*(\mathbf{x}, \mathbf{A} \cup \mathbf{B})$ be the weight function obtained by substituting in $w(\mathbf{x}, \mathbf{A})$ each condition ψ_k with B_k, for every $k \in [1..K]$. Let $\varphi^* \stackrel{def}{=} \varphi \wedge \chi \wedge \bigwedge_{k=1}^{K} (B_k \leftrightarrow \psi_k)$. Then:*

$$\mathsf{WMI}(\varphi \wedge \chi, w | \mathbf{x}, \mathbf{A}) = \mathsf{WMI}(\varphi^*, w^* | \mathbf{x}, \mathbf{A} \cup \mathbf{B}). \tag{27}$$

[15] The obvious standard syntactic simplifications apply, e.g., "$((1 \cdot x_1) - (4 \cdot x_2)) + -(5 \cdot x_3)$" is rewritten as "$(x_1 - 4x_2 - 5x_3)$".
[16] In this context, the intended meaning of "non-parametric" is that no distributional assumption is made by the model.

Proof. To every truth assignment $\mu^{\mathbf{\Psi}}$ to $\mathbf{\Psi}$ we associate the corresponding truth assignment $\mu^{\mathbf{B}}$ to $\mathbf{B}$ s.t. $\mu^{\mathbf{B}}(B_k) = \mu^{\mathbf{\Psi}}(\psi_k)$, for every $k \in [1..K]$. We note that, for every $\mu^{\mathbf{A}} \in \mathbb{B}^M$ and $\mu^{\mathbf{B}} \in \mathbb{B}^K$ (with its corresponding $\mu^{\mathbf{\Psi}}$):

$$\varphi^*_{[\mu^{\mathbf{A}} \wedge \mu^{\mathbf{B}}]} \Leftrightarrow_{\mathcal{LRA}} (\varphi \wedge \chi \wedge \mu^{\mathbf{\Psi}})_{[\mu^{\mathbf{A}} \wedge \mu^{\mathbf{B}}]}, \tag{28}$$

because every ψ_k is forced by $\mu^{\mathbf{\Psi}}$ to assume the same truth value B_k assumes in $\mu^{\mathbf{B}}$. Let w' extend w s.t. $w'(\mathbf{x}, \mathbf{A} \cup \mathbf{B}) = w(\mathbf{x}, \mathbf{A})$ for every $\mathbf{x}$, $\mathbf{A}$ and $\mathbf{B}$. Then, since φ^* forces every B_k to hold if and only if Ψ_k holds, we have:

$$\begin{aligned} & \mathsf{WMI_{nb}}(\varphi^*_{[\mu^{\mathbf{A}} \wedge \mu^{\mathbf{B}}]}, w'_{[\mu^{\mathbf{A}} \wedge \mu^{\mathbf{B}}]} | \mathbf{x}) \\ = & \mathsf{WMI_{nb}}((\varphi \wedge \chi \wedge \mu^{\mathbf{\Psi}})_{[\mu^{\mathbf{A}} \wedge \mu^{\mathbf{B}}]}, w'_{[\mu^{\mathbf{A}} \wedge \mu^{\mathbf{B}}]} | \mathbf{x}) \\ = & \mathsf{WMI_{nb}}((\varphi \wedge \chi \wedge \mu^{\mathbf{\Psi}})_{[\mu^{\mathbf{A}} \wedge \mu^{\mathbf{B}}]}, f_{\mu^{\mathbf{A}}\mu^{\mathbf{\Psi}}} | \mathbf{x}) \\ = & \mathsf{WMI_{nb}}((\varphi \wedge \chi \wedge \mu^{\mathbf{\Psi}})_{[\mu^{\mathbf{A}} \wedge \mu^{\mathbf{B}}]}, w^*_{[\mu^{\mathbf{A}} \wedge \mu^{\mathbf{B}}]} | \mathbf{x}) \\ = & \mathsf{WMI_{nb}}(\varphi^*_{[\mu^{\mathbf{A}} \wedge \mu^{\mathbf{B}}]}, w^*_{[\mu^{\mathbf{A}} \wedge \mu^{\mathbf{B}}]} | \mathbf{x}). \end{aligned} \tag{29}$$

Then, by applying K times Lemma 1, and then (29):

$$\begin{aligned} & \mathsf{WMI}(\varphi \wedge \chi, w | \mathbf{x}, \mathbf{A}) \\ = & \mathsf{WMI}(\varphi \wedge \chi \wedge \bigwedge_{k=1}^{K} (B_k \leftrightarrow \psi_k), w' | \mathbf{x}, \mathbf{A} \cup \mathbf{B}) \\ = & \sum_{\mu^{\mathbf{A}} \in \mathbb{B}^M \mu^{\mathbf{B}} \in \mathbb{B}^K} \mathsf{WMI_{nb}}(\varphi^*_{[\mu^{\mathbf{A}} \wedge \mu^{\mathbf{B}}]}, w'_{[\mu^{\mathbf{A}} \wedge \mu^{\mathbf{B}}]} | \mathbf{x}) \\ = & \sum_{\mu^{\mathbf{A}} \in \mathbb{B}^M \mu^{\mathbf{B}} \in \mathbb{B}^K} \mathsf{WMI_{nb}}(\varphi^*_{[\mu^{\mathbf{A}} \wedge \mu^{\mathbf{B}}]}, w^*_{[\mu^{\mathbf{A}} \wedge \mu^{\mathbf{B}}]} | \mathbf{x}) \\ = & \mathsf{WMI}(\varphi^*, w^* | \mathbf{x}, \mathbf{A} \cup \mathbf{B}). \qquad \square \end{aligned}$$

Example 9. Let $\mathbf{A} = \emptyset$, $\chi \overset{\text{def}}{=} [\![x \in [-1, 1]]\!]$, $\varphi \overset{\text{def}}{=} \top$, $\psi \overset{\text{def}}{=} (x \geq 0)$, and the weight $w(x) \overset{\text{def}}{=} [\![\mathsf{If}\ (x \geq 0)\ \mathsf{Then}\ x\ \mathsf{Else}\ -x]\!]$. (I.e., $w(x) \overset{\text{def}}{=} |x|$.) Then $\mathsf{WMI}(\varphi, w | x, \emptyset) = \mathsf{WMI_{nb}}(\varphi, w | \mathbf{x}) = \int_{[-1,1]} |x|\ dx = 1$. By Lemma 1, $\varphi^* = [\![x \in [-1, 1]]\!] \wedge (B \leftrightarrow (x \geq 0))$ and $w^* = [\![\mathsf{If}\ B\ \mathsf{Then}\ x\ \mathsf{Else}\ -x]\!]$, which are the same formula and weight function as in Example 6 (modulo some reordering and variable renaming), s.t. $\mathsf{WMI}(\varphi^*, w^* | \mathbf{x}, \mathbf{B}) = 1$. $\diamond$

Theorem 1 allows to compute the WMI with complicated $\mathsf{FIUC}^{\mathcal{LRA}}$ weight functions by substituting with a fresh Boolean variable B_k each condition ψ_k in the if-then-else and case constructs and by adding $\bigwedge_{k=1}^{K} (B_k \leftrightarrow \psi_k)$ to $\varphi \wedge \chi$. Intuitively, during the computation of the WMIs, Theorem 1 allows for extracting out of the integrals the conditional component on $\mathcal{LRA}$ conditions, which are labeled by Boolean atoms and can be thus handled externally. Note that the pairs of truth assignments $\langle \mu^{\mathbf{A}}, \mu^{\mathbf{\Psi}} \rangle$ of practical interest are only those for which $(\chi \wedge \mu^{\mathbf{\Psi}})_{[\mu^{\mathbf{A}}]}$ is $\mathcal{LRA}$-satisfiable. We will address this issue in §5.

3.4. *From* WMI *to* $\mathsf{WMI_{old}}$ *and vice versa*

We can now compare the original definition of WMI in [7] (Definition of $\mathsf{WMI_{old}}$ in §2.2) with our new notion of WMI applied to $\mathsf{FIUC}^{\mathcal{LRA}}$ weight functions. One key difference is that in the former the weight w is defined as a *product of weights on literals in* φ, whereas with the latter the weight w is a $\mathsf{FIUC}^{\mathcal{LRA}}$ function over the $\mathcal{LRA}$ domain $\langle \mathbf{x}, \mathbf{A} \rangle$ (and hence it does not depend on the $\mathcal{LRA}$-atoms in φ).

To this extent, we note that we can easily express and compute $\mathsf{WMI_{old}}$ (14) as WMI in the following way, by using an equivalent $\mathsf{FIUC}^{\mathcal{LRA}}$ weight function:

$$\mathsf{WMI}(\varphi, \prod_{\psi \in Atoms(\varphi)} [\![\mathsf{If}\ \psi\ \mathsf{Then}\ w(\psi)\ \mathsf{Else}\ w(\neg\psi)]\!] \ | \mathbf{x}, \mathbf{A}).$$

The vice versa is tricky, in the sense that, to the best of our knowledge and understanding, there is no obvious general way to encode an arbitrary $\mathsf{FIUC}^{\mathcal{LRA}}$ weight function into a $\mathsf{WMI_{old}}$ one while always preventing an explosion in the size of its representation. In order to understand the difficulty in finding such a general encoding, consider a *generic* $\mathsf{FIUC}^{\mathcal{LRA}}$ weight function $w(\mathbf{x}, \mathbf{A})$. In order to write it as a $\mathsf{WMI_{old}}$ weight function, one should find an integer K, a set of conditions

$\{\psi_k(\mathbf{x},\mathbf{A})\}_{k=1}^K$ and a set of positive functions $\{f_{\psi_k}(\mathbf{x}), f_{\neg\psi_k}(\mathbf{x})\}_{k=1}^K$ so that $w(\mathbf{x},\mathbf{A})$ could be written into the $\mathsf{WMI}_{\mathsf{old}}$-equivalent form:

$$w(\mathbf{x},\mathbf{A}) = \prod_{k=1}^{K} [\![\mathsf{If}\ \psi_k(\mathbf{x},\mathbf{A})\ \mathsf{Then}\ f_{\psi_k}(\mathbf{x})\ \mathsf{Else}\ f_{\neg\psi_k}(\mathbf{x})]\!] \tag{30}$$

where the conditions ψ_k can be either (a) Boolean atoms in $\mathbf{A}$, (b) $\mathcal{LRA}$-atoms on $\mathbf{x}$, (c) $\mathcal{LRA}$-formulas on atoms in the form (a) and (b) by labeling them with fresh Boolean atoms, so that their truth values derive deterministically from the values of $\mathbf{x}, \mathbf{A}$, written "$\psi_k(\mathbf{x},\mathbf{A})$".

Since $w(\mathbf{x},\mathbf{A})$, $f_{\psi_k}(\mathbf{x})$ and $f_{\neg\psi_k}(\mathbf{x})$ are positive for every k, and the *log* function is continuous, strictly increasing and invertible, noticing that $[\![\mathsf{If}\ A\ \mathsf{Then}\ b\ \mathsf{Else}\ c]\!] = [\![\mathsf{If}\ A\ \mathsf{Then}\ b\ \mathsf{Else}\ 0]\!] + [\![\mathsf{If}\ A\ \mathsf{Then}\ 0\ \mathsf{Else}\ c]\!]$, we see that (30) is equivalent to:

$$log(w(\mathbf{x},\mathbf{A})) = \sum_{k=1}^{K} \begin{array}{l} [\![\mathsf{If}\ \psi_k(\mathbf{x},\mathbf{A})\ \mathsf{Then}\ log(f_{\psi_k}(\mathbf{x}))\ \mathsf{Else}\ 0]\!] + \\ [\![\mathsf{If}\ \psi_k(\mathbf{x},\mathbf{A})\ \mathsf{Then}\ 0\ \mathsf{Else}\ log(f_{\neg\psi_k}(\mathbf{x}))]\!] \end{array} \tag{31}$$

for every $\mathbf{x},\mathbf{A}$. Thus, if we fix the value for $\mathbf{x}$ and call $y_{k\top} \stackrel{\text{def}}{=} log(f_{\psi_k}(\mathbf{x}))$ and $y_{k\bot} \stackrel{\text{def}}{=} log(f_{\neg\psi_k}(\mathbf{x}))$ s.t. $y_{k\top}, y_{k\bot} \in \mathbb{R} \cup \{-\infty\}$, then (31) can be represented as a system of $2^{|\mathbf{A}|}$ linear equalities, one for each total truth assignment on $\mathbf{A}$, on $2K$ variables $\{y_{k\top}, y_{k\bot}\}_{k=1}^K$ whose $\{0,1\}$-coefficients are given by the truth values of $\psi_k(\mathbf{x},\mathbf{A})$. Thus, for every value of $\mathbf{x}$, we have $2^{|\mathbf{A}|}$ linear equations with $2K$ real-valued variables. This suggests that, in order (30) to hold, the size K of the product may blow up in size with $|\mathbf{A}|$.

For instance, a trivial general solution consists in first converting the problem into $\mathsf{WMI}(\varphi^*, w^*|\mathbf{x},\mathbf{A}^*)$ as in Theorem 1 and then, *for every total truth assignment* μ *in* $\mathcal{TTA}(\varphi^*)$, introducing a fresh new Boolean atom B_μ adding $B_\mu \leftrightarrow \mu$ to the formula, and defining $w(B_\mu) \stackrel{\text{def}}{=} w_\mu(\mathbf{x})$, $w(\neg B_\mu) \stackrel{\text{def}}{=} 1$, $w(l) \stackrel{\text{def}}{=} 1$ for every other literal l. This solution is obviously not practical for non-trivial size of $\mathbf{A}^*$ because it generates an exponential growth in the size of the formula.

4. A case study

Consider modeling journey time on a road network for e.g. a delivery agency. In order to safely organize priority deliveries, the agency could be interested in knowing well in advance the probability of completing the journey within a certain time, given the time of departure. An accurate estimate requires to consider how travel duration between locations can change according to the time of the day, and combine these duration distributions over the entire route. A different encoding for the same problem was presented in the original WMI work [7].

Suppose that (the part of interest of) the day is partitioned into $\{I^1, \dots, I^M\}$ disjoint and consecutive intervals such that, for each adjacent location l_i and l_j in the road network and for each $I^m \stackrel{\text{def}}{=} [c_m, c_{m+1})$, we know the distribution of the journey time from location l_i to location l_j given that we move at time $t \in I^m$. Let $f^m_{l_i,l_j} : \mathbb{R} \mapsto \mathbb{R}^+$ denote such distribution and let the interval $R^m_{l_i,l_j} \stackrel{\text{def}}{=} [a^m_{l_i,l_j}, b^m_{l_i,l_j})$ be its support. (Note that the I^ms are intervals in absolute time and are all disjoint whereas the $R^m_{l_i,l_j}$s are intervals in relative time and are typically not disjoint.)

4.1. Modeling a journey with a fixed path

Given a path $(l_0, ..., l_N)$ and departure and arrival times t_{dep} and t_{arr}, we are interested in answering queries of the form $P(t_N \le t_{\text{arr}} \mid t_0 = t_{\text{dep}}, \{l_i\}_{i=0}^N)$. We can encode the problem as follows. Let t_n be the time at step n and x_n the journey time between l_{n-1} and l_n. Let $\mathbf{x} \stackrel{\text{def}}{=} \{x_1, \dots, x_N\}$. (Here $\mathbf{A} \stackrel{\text{def}}{=} \emptyset$.) Then:

$$\begin{aligned}
\chi(\mathbf{x}) &\stackrel{\text{def}}{=} \bigwedge_{n=0}^{N} [\![t_n \in [c_1, c_{M+1}]]\!] \ \wedge \ \bigwedge_{n=1}^{N} [\![\mathsf{OneOf}\{[\![t_{n-1} \in I^m]\!]\}_{m=1}^M]\!] \\
&\quad \wedge \bigwedge_{n=1}^{N}\bigwedge_{m=1}^{M} ([\![t_{n-1} \in I^m]\!] \rightarrow [\![x_n \in R^m_{l_{n-1},l_n}]\!]) \\
w(\mathbf{x}) &\stackrel{\text{def}}{=} \prod_{n=1}^{N} \left[\!\!\left[\mathsf{Case}\ \ [\![t_{n-1} \in I^1]\!] : f^1_{l_{n-1},l_n}(x_n);\ \dots\ [\![t_{n-1} \in I^M]\!] : f^M_{l_{n-1},l_n}(x_n)\right]\!\!\right] \\
\varphi(\mathbf{x}) &\stackrel{\text{def}}{=} \top,
\end{aligned}$$

where for $n > 0$, "t_n" is a shortcut for the term "$\sum_{i=1}^n x_i + t_0$", so that "$[\![t_{n-1} \in I^m]\!]$" is a shortcut for the formula "$(\sum_{i=1}^{n-1} x_i + t_0 \ge c_m) \wedge \neg(\sum_{i=1}^{n-1} x_i + t_0 \ge c_{m+1})$", and "$[\![x_n \in R^m_{l_{n-1},l_n}]\!]$" is a shortcut for the formula "$(x_n \ge a^m_{l_{n-1},l_n}) \wedge \neg(x_n \ge b^m_{l_{n-1},l_n})$".

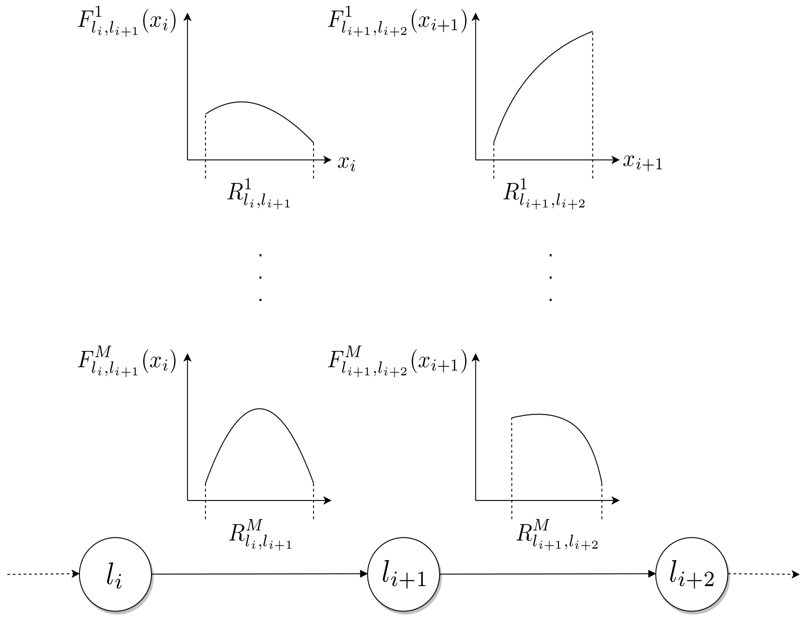


Fig. 3. This figure shows the journey time densities for a pair of consecutive time steps, from location l_i to l_{i+2}. Each edge shows the corresponding journey time distribution for each of the intervals.

This encoding allows us to answer the up-mentioned queries as follows:

$$P(t_N \le t_{\text{arr}} \mid t_0 = t_{\text{dep}}, \{l_i\}_{i=0}^N) = \frac{\mathsf{WMI}_{\text{nb}}(\chi(\mathbf{x}) \wedge (t_N \le t_{\text{arr}}) \wedge (t_0 = t_{\text{dep}}), w(\mathbf{x})|\mathbf{x})}{\mathsf{WMI}_{\text{nb}}(\chi(\mathbf{x}) \wedge (t_0 = t_{\text{dep}}), w(\mathbf{x})|\mathbf{x})}$$

where the locations $\{l_i\}_{i=0}^N$ are used to generate a query-specific encoding for $\chi(\mathbf{x})$ and $w(\mathbf{x})$.

Under the assumption that each distribution $f^m_{l_i,l_j}(x)$ is feasibly integrable if $x \in R^m_{l_i,l_j}$, then $w(\mathbf{x})$ is $\mathsf{FIUC}^{\mathcal{LRA}}$ with $N \cdot M$ conditions $\psi^m_n \stackrel{\text{def}}{=} [\![t_n \in I^m]\!]$. Thus we can introduce $N \cdot M$ fresh Boolean atoms B^m_n and apply Theorem 1, obtaining:

$$\varphi^*(\mathbf{x}, \mathbf{B}) \stackrel{\text{def}}{=} \varphi(\mathbf{x}) \wedge \chi(\mathbf{x}) \wedge \bigwedge_{n=1}^{N} \bigwedge_{m=1}^{M} (B^m_{n-1} \leftrightarrow [\![t_{n-1} \in I^m]\!])$$

$$w^*(\mathbf{x}, \mathbf{B}) \stackrel{\text{def}}{=} \prod_{n=1}^{N} \left[\!\!\left[\mathsf{Case}\ B^1_{n-1} : f^1_{l_{n-1},l_n}(x_n); \ldots B^M_{n-1} : f^M_{l_{n-1},l_n}(x_n)\right]\!\!\right].$$

Each distribution $f^m_{l_{n-1},l_n}$ is thus associated to B^m_n. Note that, for each step n, exactly one condition variable B^m_n is true, representing the fact that the n-th location is reached during the m-th interval. Intuitively, this allows to select at each step the distribution corresponding to the interval in which the location is reached, as shown in Fig. 3.

Example 10. Consider an instance of our case study where $\mathbf{A} \stackrel{\text{def}}{=} \emptyset$, $N = 2$, $M = 3$,

$$\begin{aligned}
\chi(\mathbf{x}) \stackrel{\text{def}}{=} & [\![t_0 \in [7, 10)]\!] \\
& \wedge [\![t_0 + x_1 \in [7, 10)]\!] \\
& \wedge [\![\mathsf{OneOf}\{[\![t_0 \in [7, 8)]\!], \ldots, [\![t_0 \in [9, 10)]\!]\}]\!] \\
& \wedge [\![\mathsf{OneOf}\{[\![t_0 + x_1 \in [7, 8)]\!], \ldots, [\![t_0 + x_1 \in [9, 10)]\!]\}]\!] \\
& \wedge [\![t_0 \in [7, 8)]\!] \rightarrow [\![x_1 \in [0.5, 1)]\!] \\
& \wedge [\![t_0 \in [8, 9)]\!] \rightarrow [\![x_1 \in [1, 1.5)]\!] \\
& \wedge [\![t_0 \in [9, 10)]\!] \rightarrow [\![x_1 \in [1, 2)]\!] \\
& \wedge [\![t_0 + x_1 \in [7, 8)]\!] \rightarrow [\![x_2 \in [1, 1.5)]\!] \\
& \wedge [\![t_0 + x_1 \in [8, 9)]\!] \rightarrow [\![x_2 \in [1.5, 2)]\!] \\
& \wedge [\![t_0 + x_1 \in [9, 10)]\!] \rightarrow [\![x_2 \in [1, 2)]\!]
\end{aligned}$$

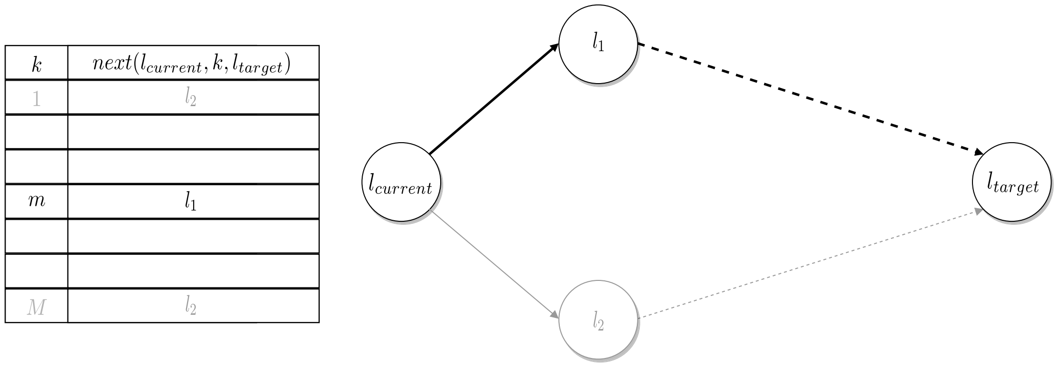


Fig. 4. The figure shows two alternative (sub)paths from l_{curr} to l_{target}: the successor of l_{curr} is selected according to the time interval at which the node is reached (m in the figure).

$$w(\mathbf{x}) \stackrel{\text{def}}{=} \left[\!\!\left[\begin{array}{ll} \text{Case} & \\ [\![t_0 \in [7,8)]\!] & : f^1_{l_0 l_1}(x_1); \\ [\![t_0 \in [8,9)]\!] & : f^2_{l_0 l_1}(x_1); \\ [\![t_0 \in [9,10)]\!] & : f^3_{l_0 l_1}(x_1); \end{array} \right]\!\!\right] \cdot \left[\!\!\left[\begin{array}{ll} \text{Case} & \\ [\![t_0 + x_1 \in [7,8)]\!] & : f^1_{l_1 l_2}(x_2); \\ [\![t_0 + x_1 \in [8,9)]\!] & : f^2_{l_1 l_2}(x_2); \\ [\![t_0 + x_1 \in [9,10)]\!] & : f^3_{l_1 l_2}(x_2); \end{array} \right]\!\!\right]$$

$$\varphi(\mathbf{x}) \stackrel{\text{def}}{=} \top$$

where the $f^m_{l_{n-1}l_n}(x_n)$ are functions which are integrable and positive in their respective domain stated in $\chi(\mathbf{x})$ (e.g., $f^1_{l_0 l_1}(x_1)$ is integrable and positive in $[\![x_1 \in [0.5, 1)]\!]$).

Then, by applying Theorem 1, we can introduce 6 Boolean variables B^m_n and reformulate the problem as follows:

$$\begin{aligned} \varphi^*(\mathbf{x}, \mathbf{B}) \stackrel{\text{def}}{=} & \ \varphi(\mathbf{x}) \wedge \chi(\mathbf{x}) \\ & \wedge (B^1_0 \leftrightarrow [\![t_0 \in [7,8)]\!]) \\ & \wedge \ldots \\ & \wedge (B^3_1 \leftrightarrow [\![t_0 + x_1 \in [9,10)]\!]) \end{aligned} \tag{32}$$

$$w^*(\mathbf{x}, \mathbf{B}) \stackrel{\text{def}}{=} \left[\!\!\left[\begin{array}{ll} \text{Case} & \\ B^1_0 & : f^1_{l_0 l_1}(x_1); \\ B^2_0 & : f^2_{l_0 l_1}(x_1); \\ B^3_0 & : f^3_{l_0 l_1}(x_1); \end{array} \right]\!\!\right] \cdot \left[\!\!\left[\begin{array}{ll} \text{Case} & \\ B^1_1 & : f^1_{l_1 l_2}(x_2); \\ B^2_1 & : f^2_{l_1 l_2}(x_2); \\ B^3_1 & : f^3_{l_1 l_2}(x_2); \end{array} \right]\!\!\right]$$

4.2. Modeling a journey under a conditional plan

We generalize the previous scenario to the case in which the path is not given in advance. Rather, they are provided only a maximum path length N, a final target location l_{target} and a *conditional plan*, establishing the successor location for every location at each time slot. Intuitively, the conditional plan mimics the empirical knowledge of a driver that, given his/her current location and time of the day, chooses the next step towards the final destination. (E.g., if one road passes aside a school entrance, the driver arriving there from 8 am to 9 am knows it is better to choose an alternative path to avoid the queues of cars leaving the children there.)

Let $\mathcal{I} = \{1, \ldots, M\}$ be the set of indices of the time intervals. Let $\mathcal{L} = \{1, \ldots, L\}$ be the set of indices of each location. Given a target destination $l_{\text{target}} \in \mathcal{L}$, a *conditional plan* is a function $\text{next} : \mathcal{L} \times \mathcal{I} \times \mathcal{L} \to \mathcal{L}$ such that, for any current location $l \in \mathcal{L}$ and time interval index $m \in \mathcal{I}$, $\text{next}(l, m, l_{\text{target}})$ is the next location in the path, as shown in Fig. 4. To handle the special case of the final location l_{target}, we set $\text{next}(l_{\text{target}}, m, l_{\text{target}}) \stackrel{\text{def}}{=} l_{\text{target}}$ and $R^m_{l_{\text{target}}, l_{\text{target}}} \stackrel{\text{def}}{=} [0, 0]$, so that $[\![x \in R^m_{l_{\text{target}}, l_{\text{target}}}]\!] \stackrel{\text{def}}{=} (x = 0)$. The queries we want to address are of the form

$$P(t_N \le t_{\text{arr}} \mid t_0 = t_{\text{dep}}, l_{\text{dep}}, l_{\text{target}}, \text{next}).$$

The encoding generalizes the previous encoding for fixed paths. This time we need to introduce N sets of L mutually-exclusive Boolean variables $A_{n,l}$ which encode the location visited at each time step – with the intended meaning that $A_{n,l}$

is true iff the location at step n is the one indexed by l – and $[\![\mathsf{OneOf}\{A_{n,l} \mid l \in [1,L]\}]\!]$ is added to the formula for every n.[17] Unless otherwise specified, in the following we use the same notation and shortcuts of the case study presented in §4 (in particular, we recall that for $n > 0$, "t_n" is a shortcut for the term "$\sum_{i=1}^{n} x_i + t_0$"):

$$\begin{aligned}
\chi(\mathbf{x},\mathbf{A}) \overset{\text{def}}{=} & \bigwedge_{n=0}^{N} [\![t_n \in [c_1, c_{M+1}]]\!] \wedge \bigwedge_{n=1}^{N} [\![\mathsf{OneOf}\{[\![t_{n-1} \in I^m]\!] \mid m \in [1,M]\}]\!] \\
& \wedge \bigwedge_{n=0}^{N} [\![\mathsf{OneOf}\{A_{n,l} \mid l \in [1,L]\}]\!] \\
& \wedge \bigwedge_{n=1}^{N} \Big(\bigwedge_{l=1}^{L} \Big(A_{n-1,l} \rightarrow \bigwedge_{m=1}^{M} ([\![t_{n-1} \in I^m]\!] \rightarrow [\![x_n \in R^m_{l,\mathsf{next}(l,m,l_{\mathsf{target}})}]\!]) \Big) \Big), \\
\varphi(\mathbf{x},\mathbf{A}) \overset{\text{def}}{=} & A_{0,l_0} \wedge \bigwedge_{n=1}^{N} \Big(\bigwedge_{l=1}^{L} \Big(A_{n-1,l} \rightarrow \bigwedge_{m=1}^{M} ([\![t_{n-1} \in I^m]\!] \rightarrow A_{n,\mathsf{next}(l,m,l_{\mathsf{target}})}) \Big) \Big)
\end{aligned}$$

$$w(\mathbf{x},\mathbf{A}) \overset{\text{def}}{=} \prod_{n=1}^{N} \left[\!\!\left[\begin{array}{l}
\mathsf{Case} \\
(A_{n-1,l_1} \wedge A_{n,l_2}): \\
\quad [\![\mathsf{Case}\ [\![t_{n-1} \in I^1]\!] : f^1_{l_1,l_2}(x_n);\ \ldots\ ;\ [\![t_{n-1} \in I^M]\!] : f^M_{l_1,l_2}(x_n)\ ]\!]; \\
(A_{n-1,l_1} \wedge A_{n,l_3}): \\
\quad [\![\mathsf{Case}\ [\![t_{n-1} \in I^1]\!] : f^1_{l_1,l_3}(x_n);\ \ldots\ ;\ [\![t_{n-1} \in I^M]\!] : f^M_{l_1,l_3}(x_n)\ ]\!]; \\
\ldots \\
(A_{n-1,l_L} \wedge A_{n,l_{L-1}}): \\
\quad [\![\mathsf{Case}\ [\![t_{n-1} \in I^1]\!] : f^1_{l_L,l_{L-1}}(x_n);\ \ldots\ ;\ [\![t_{n-1} \in I^M]\!] : f^M_{l_L,l_{L-1}}(x_n)\ ]\!];
\end{array} \right]\!\!\right]$$

Note that in the definition of $w(\mathbf{x},\mathbf{A})$ the pairs of locations of interest in the outer case-expression are only those connected by an edge. Moreover, we can safely consider only the edges which appear in the conditional plan for the desired l_{target}, i.e. pairs in $\mathcal{L}_{\mathsf{reach}} = \{\langle l_i, l_j \rangle \mid l_i \neq l_j \in \mathcal{L}, \exists m \in [1,M] \,.\, \mathsf{next}(l_i, m, l_{\mathsf{target}}) = l_j\}$.

If we compare the description of $\chi(\mathbf{x})$, $w(\mathbf{x})$ and $\varphi(\mathbf{x})$ for the fixed-path-setting in §4.1 with these of $\chi(\mathbf{x},\mathbf{A})$, $w(\mathbf{x},\mathbf{A})$ and $\varphi(\mathbf{x},\mathbf{A})$ for the conditional-plan setting described above, we note that the former can be seen as a particular subcase of the latter.[18]

This encoding allows us to answer the queries of interest as follows:

$$P(t_N \le t_{\mathsf{arr}} \mid t_0 = t_{\mathsf{dep}}, l_{\mathsf{dep}}, l_{\mathsf{target}}, \mathsf{next}) = \frac{\mathsf{WMI}(\varphi(\mathbf{x},\mathbf{A}) \wedge \chi(\mathbf{x},\mathbf{A}) \wedge (t_N \le t_{\mathsf{arr}}) \wedge (t_0 = t_{\mathsf{dep}}), w(\mathbf{x},\mathbf{A}) | \mathbf{x},\mathbf{A})}{\mathsf{WMI}(\varphi(\mathbf{x},\mathbf{A}) \wedge \chi(\mathbf{x},\mathbf{A}) \wedge (t_0 = t_{\mathsf{dep}}), w(\mathbf{x},\mathbf{A}) | \mathbf{x},\mathbf{A})}$$

where $l_{\mathsf{dep}}, l_{\mathsf{target}}$ and next are used to generate a query-specific encoding for $\varphi(\mathbf{x},\mathbf{A})$, $\chi(\mathbf{x},\mathbf{A})$ and $w(\mathbf{x},\mathbf{A})$.

4.3. Efficiency of the encodings

Note that, with both encodings in §4.1 and §4.2, in the formula χ the constraints "$[\![\mathsf{OneOf}\{[\![t_{n-1} \in I^m]\!]\}_{m=1}^{M}]\!]$" (see §2.1) are not strictly necessary from the logical perspective because the I^m's are mutually exclusive by construction and $[\![t_n \in [c_1, c_{M+1}]]\!]$ for $n \in [0,N]$. Nevertheless, adding such constraints may improve the performances of the SMT solver the formula is fed to, because they allow the solver to infer the disjunction and the mutual exclusion directly via Boolean constraint propagation instead of using less efficient $\mathcal{LRA}$-deduction steps (see e.g. *static learning* in [44]).

For the same reason, adding the following logically-redundant constraints to χ may improve the performances of the SMT solver:

$$\begin{aligned}
(\sum_{i=1}^{n-1} x_i + t_0 \ge c_{m+1}) \rightarrow (\sum_{i=1}^{n-1} x_i + t_0 \ge c_m) \qquad & \forall n \in [1,N],\ m \in [1, M-1] \\
(x_n \ge v_i) \rightarrow (x_n \ge v_j) \qquad & \textit{if } v_i \ge v_j,\ \forall n \in [1,N],
\end{aligned}$$

where the v_i, v_j are among the upper- and lower-bound values of the intervals $R^m_{l_{n-1},l_n}$.[19]

[17] Alternatively, this can be encoded by a distinct truth assignment to $\lceil log_2(L) \rceil$ Boolean variables representing the binary encoding of the index l, using overall $N \cdot \lceil log_2(L) \rceil$ Boolean variables. However, since L is quite small, this is not worth doing in our case.

[18] In fact, if we impose a given path $l_0, l_1, \ldots, l_N$ by substituting the part "$A_{0,l_0} \wedge \bigwedge_{n=1}^{N}(\ldots)$" in $\varphi(\mathbf{x},\mathbf{A})$ with "$\bigwedge_{n=0}^{N} A_{n,l_n}$", then it is easy to see that $\chi(\mathbf{x},\mathbf{A}) \wedge \varphi(\mathbf{x},\mathbf{A})$ simplifies by unit-propagation into $\mu^{\mathbf{A}} \wedge \chi(\mathbf{x}) \wedge \varphi(\mathbf{x})$, where $\mu^{\mathbf{A}} \overset{\text{def}}{=} \bigwedge_{n=0}^{N}(A_{n,l_n} \wedge \bigwedge_{l \neq l_n} \neg A_{n,l})$, and that $w_{[\mu^{\mathbf{A}}]}(\mathbf{x})$ simplifies into $w(\mathbf{x})$ because only one condition $(A_{n-1,l_{n-1}} \wedge A_{n,l_n})$ holds for the n-th external Case.

[19] In practice, we do not need adding such constraints for every pair $\langle v_i, v_j \rangle$; rather and more efficiently, it suffices to sort all such values for each x_n, and to add one constraint only for pairs of consecutive values, because the others are obtained implicitly by transitivity.

5. Efficient WMI computation

We address the problem of computing efficiently the WMI of a $\mathsf{FIUC}^{\mathcal{LRA}}$ weight function $w(\mathbf{x}, \mathbf{A})$, with support formula χ and set of conditions $\mathbf{\Psi}$, over a formula $\varphi(\mathbf{x}, \mathbf{A})$.

The first step (if needed) is a preprocessing step in which the problem is transformed by labeling all conditions $\mathbf{\Psi}$ with fresh Boolean atoms $\mathbf{B}$, as in Theorem 1. Let $\varphi^*, w^*, \mathbf{x}, \mathbf{A}^*$ be the result of such process, where $\varphi^* \stackrel{\text{def}}{=} \varphi \wedge \chi \wedge \bigwedge_{k=1}^{K}(B_k \leftrightarrow \psi_k)$, $w^* \stackrel{\text{def}}{=} w[\mathbf{B} \leftarrow \mathbf{\Psi}]$, and $\mathbf{A}^* \stackrel{\text{def}}{=} \mathbf{A} \cup \mathbf{B}$. Consequently, for every $\mu^{\mathbf{A}^*}$, $w^*_{[\mu^{\mathbf{A}^*}]}$ is feasibly integrable on $\varphi^*_{[\mu^{\mathbf{A}^*}]}$.

Remark 5. Following up with Remark 4, hereafter we assume w.l.o.g. that each Boolean atom in $\mathbf{A}$ occurs in $\varphi \wedge \chi$ or in $\mathbf{\Psi}$, so that every atom in $\mathbf{A}^*$ occurs in φ^*. Consequently, each truth assignment in $\mathcal{TTA}(\varphi^*)$ assigns every atom in $\mathbf{A}^*$. The same assumption applies to $\exists \mathbf{x}.\varphi^*$.[20]

5.1. *The procedure* WMI-AllSMT

Consider $\mu = \mu^{\mathbf{A}^*} \wedge \mu^{\mathcal{LRA}} \in \mathcal{TTA}(\varphi^*)$. Then $\mu^{\mathcal{LRA}} \in \mathcal{TTA}(\varphi^*_{[\mu^{\mathbf{A}^*}]})$, so that we can compute $\mathsf{WMI}_{\mathsf{nb}}(\mu^{\mathcal{LRA}}, w^*_{[\mu^{\mathbf{A}^*}]}|\mathbf{x})$. Combining Theorem 1 with Proposition 2 allows us to compute the WMI as follows:

$$\begin{aligned} &\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A}) \\ &= \mathsf{WMI}(\varphi^*, w^*|\mathbf{x}, \mathbf{A}^*). \\ &= \sum_{\mu^{\mathbf{A}^*} \wedge \mu^{\mathcal{LRA}} \in \mathcal{TTA}(\varphi^*)} \mathsf{WMI}_{\mathsf{nb}}(\mu^{\mathcal{LRA}}, w^*_{[\mu^{\mathbf{A}^*}]}|\mathbf{x}). \end{aligned} \tag{33}$$

The set $\mathcal{TTA}(\varphi^*)$ is computed by an AllSMT procedure implemented on top of an SMT solver like MathSAT5—i.e., as $\mathcal{TTA}(\mathsf{PredAbs}_{[\varphi^*]}(Atoms(\varphi^*)))$, without the assignment-reduction technique (see fact (b) in §2.1). Each $\mathsf{WMI}_{\mathsf{nb}}(\mu^{\mathcal{LRA}}, w^*_{[\mu^{\mathbf{A}^*}]}|\mathbf{x})$ is computed by invoking our background integration procedure for $\mathsf{FI}^{\mathcal{LRA}}$ functions of §3.3. We call this algorithm WMI-AllSMT.

5.2. *The procedure* WMI-PA

A much more efficient technique, which we call WMI-PA because it exploits SMT-based predicate abstraction in its full pruning power rather than simply as AllSMT,[21] can be implemented by noticing that, combining Theorem 1 with Proposition 3, we have that:

$$\begin{aligned} &\mathsf{WMI}(\varphi, w|\mathbf{x}, \mathbf{A}) \\ &= \mathsf{WMI}(\varphi^*, w^*|\mathbf{x}, \mathbf{A}^*). \\ &= \sum_{\mu^{\mathbf{A}^*} \in \mathcal{TTA}(\exists \mathbf{x}.\varphi^*)} \mathsf{WMI}_{\mathsf{nb}}(\varphi^*_{[\mu^{\mathbf{A}^*}]}, w^*_{[\mu^{\mathbf{A}^*}]}|\mathbf{x}) \end{aligned} \tag{34}$$

and that, due to Proposition 1, each $\mathsf{WMI}_{\mathsf{nb}}(\varphi^*_{[\mu^{\mathbf{A}^*}]}, w^*_{[\mu^{\mathbf{A}^*}]}|\mathbf{x})$ can be computed as:

$$\sum_{\mu^{\mathcal{LRA}} \in \mathcal{TA}(\varphi^*_{[\mu^{\mathbf{A}^*}]})} \mathsf{WMI}_{\mathsf{nb}}(\mu^{\mathcal{LRA}}, w^*_{[\mu^{\mathbf{A}^*}]}|\mathbf{x}). \tag{35}$$

Note that in (34) we must use $\mathcal{TTA}(...)$ instead of $\mathcal{TA}(...)$ because by construction $w^*_{[\mu^{\mathbf{A}^*}]}$ requires each $\mu^{\mathbf{A}^*}$ to be total, whereas in (35) we can use $\mathcal{TA}(...)$ because there is no need for the $\mu^{\mathcal{LRA}}$s to be total (Proposition 1).

The pseudocode of WMI-PA is reported in Algorithm 1. First, the problem is transformed (if needed) by labeling conditions $\mathbf{\Psi}$ with fresh Boolean variables $\mathbf{B}$, as in Theorem 1. After this preprocessing stage, the set $\mathcal{M}^{\mathbf{A}^*} \stackrel{\text{def}}{=} \mathcal{TTA}(\exists \mathbf{x}.\varphi^*)$ is computed by invoking $\mathcal{TTA}(\mathsf{PredAbs}_{[\varphi^*]}(\mathbf{A}^*))$ (see §2.1). Then, the algorithm iterates over each Boolean assignment $\mu^{\mathbf{A}^*}$ in $\mathcal{M}^{\mathbf{A}^*}$. $\varphi^*_{[\mu^{\mathbf{A}^*}]}$ can be simplified by the $\mathsf{Simplify}$ procedure, by propagating truth values (e.g., $\varphi_1 \wedge (\top \vee \varphi_2) \wedge (\bot \vee \varphi_3) \wedge (\neg\varphi_3 \vee \varphi_4) \Rightarrow \varphi_1 \wedge \varphi_3 \wedge \varphi_4$) and by applying arithmetical simplifications like $\mathcal{LRA}$ theory propagation [5] (e.g.,

[20] We note that this assumption is necessary for our basic procedure WMI-AllSMT (see §5.1) but it is not necessary with our much more efficient procedure WMI-PA (see §5.2) because the SMT-based procedure we use for computing predicate abstraction, $\mathcal{TTA}(\mathsf{PredAbs}_{[\varphi]}(\mathbf{A}))$ (see §2.1), allows for forcing the branches even on atoms $\mathbf{A}_i$ which do not actually occur in the input formula φ. Nevertheless, this assumption makes the explanation simpler.

[21] To this extent, compare facts (c) and (b) in §2.1.

Algorithm 1 WMI-PA(φ, w, $\mathbf{x}$, $\mathbf{A}$).

```
$\langle\varphi^*, w^*, \mathbf{A}^*\rangle \leftarrow$ LabelConditions($\varphi, w, \mathbf{x}, \mathbf{A}$)
$\mathcal{M}^{\mathbf{A}^*} \leftarrow \mathcal{TTA}(\mathsf{PredAbs}_{[\varphi^*]}(\mathbf{A}^*))$
$vol \leftarrow 0$
for $\mu^{\mathbf{A}^*} \in \mathcal{M}^{\mathbf{A}^*}$ do
  Simplify($\varphi^*_{[\mu^{\mathbf{A}^*}]}$)
  if LiteralConjunction($\varphi^*_{[\mu^{\mathbf{A}^*}]}$) then
    $vol \leftarrow vol + \mathsf{WMI}_{\mathsf{nb}}(\varphi^*_{[\mu^{\mathbf{A}^*}]}, w^*_{[\mu^{\mathbf{A}^*}]}|\mathbf{x})$
  else
    $\mathcal{M}^{\mathcal{LRA}} \leftarrow \mathcal{TA}(\mathsf{PredAbs}_{[\varphi^*_{[\mu^{\mathbf{A}^*}]}]}(Atoms(\varphi^*_{[\mu^{\mathbf{A}^*}]})))$
    for $\mu^{\mathcal{LRA}} \in \mathcal{M}^{\mathcal{LRA}}$ do
      $vol \leftarrow vol + \mathsf{WMI}_{\mathsf{nb}}(\mu^{\mathcal{LRA}}, w^*_{[\mu^{\mathbf{A}^*}]}|\mathbf{x})$
    end for
  end if
end for
return $vol$
```

$(x \geq 1) \wedge (\neg(x \geq 0) \vee \varphi_1) \wedge ((x \geq 0) \vee \varphi_2) \Rightarrow (x \geq 1) \wedge \varphi_1)$. This improves the chances of reducing $\varphi^*_{[\mu^{\mathbf{A}^*}]}$ to a conjunction of literals, and allows for reducing the size of $Atoms(\varphi^*_{[\mu^{\mathbf{A}^*}]})$ to feed to PredAbs (see below). Then, if $\varphi^*_{[\mu^{\mathbf{A}^*}]}$ is already a conjunction of literals, then the algorithm directly computes its contribution to the volume by calling $\mathsf{WMI}_{\mathsf{nb}}(\varphi^*_{[\mu^{\mathbf{A}^*}]}, w^*_{[\mu^{\mathbf{A}^*}]}|\mathbf{x})$. Otherwise, $\mathcal{TA}(\varphi^*_{[\mu^{\mathbf{A}^*}]})$ is computed as $\mathcal{TA}(\mathsf{PredAbs}_{[\varphi^*_{[\mu^{\mathbf{A}^*}]}]}(Atoms(\varphi^*_{[\mu^{\mathbf{A}^*}]})))$, using the assignment-reduction technique to produce partial assignments (see §2.1), and the algorithm iteratively computes the contributions to the volume for each $\mu^{\mathcal{LRA}}$.

Example 11. Consider the problem described by φ^* and w^* in Example 10. Since $\mathbf{A} = \emptyset$, then $\mathbf{A}^* = \mathbf{B}$.

Suppose first we generically want to leave l_0 no earlier than 7 and no later than 10, and arrive to l_2 strictly before 11. These constraints correspond to conjoining

$$[\![t_0 \in [7, 10)]\!] \wedge (t_0 + x_1 + x_2 < 11)$$

to φ^*. In such case, $\mathsf{PredAbs}_{[\varphi^*]}(\mathbf{A}^*)$ is the following formula:

$$(\quad B_0^1 \wedge \neg B_0^2 \wedge \neg B_0^3 \wedge \quad B_1^1 \wedge \neg B_1^2 \wedge \neg B_1^3) \tag{36}$$
$$\vee(\quad B_0^1 \wedge \neg B_0^2 \wedge \neg B_0^3 \wedge \neg B_1^1 \wedge \quad B_1^2 \wedge \neg B_1^3) \tag{37}$$
$$\vee(\neg B_0^1 \wedge \quad B_0^2 \wedge \neg B_0^3 \wedge \neg B_1^1 \wedge \neg B_1^2 \wedge \quad B_1^3) \tag{38}$$

so that $\mathcal{M}^{\mathbf{A}^*} \overset{\text{def}}{=} \mathcal{TTA}(\mathsf{PredAbs}_{[\varphi^*]}(\mathbf{A}^*))$ is the set of the three disjuncts (36)-(38). Importantly, note that the other 6 assignments, which would make φ^* $\mathcal{LRA}$-unsatisfiable causing $\mathsf{WMI}_{\mathsf{nb}}$ to return 0, *are not generated by* $\mathcal{TTA}(\mathsf{PredAbs}_{[\varphi^*]}(\mathbf{A}^*))$. (E.g., if $B_0^1 = \top$ then l_1 is necessarily reached strictly before 9, which forces $B_1^3 = \bot$, s.t. the assignment $(B_0^1 \wedge \neg B_0^2 \wedge \neg B_0^3 \wedge \neg B_1^1 \wedge \neg B_1^2 \wedge B_1^3)$ is not generated.)

Now suppose instead that we fix t_0 to some value $t_{\mathsf{dep}} \in [7, 10)$ by conjoining $(t_0 = t_{\mathsf{dep}})$ to φ^* (see §7). Depending on the value t_{dep}, we distinguish four cases:

$t_{\mathsf{dep}} \in [7, 7.5)$: forces $B_0^1 = \top$ and $\mathcal{TTA}(\mathsf{PredAbs}_{[\varphi^*]}(\mathbf{A}^*))$ reduces to (36) and (37);
$t_{\mathsf{dep}} \in [7.5, 8)$: forces $B_0^1 = \top$ and $\mathcal{TTA}(\mathsf{PredAbs}_{[\varphi^*]}(\mathbf{A}^*))$ reduces to (37) because (36) cannot be extended with any $\mathcal{LRA}$-satisfiable $\mu^{\mathcal{LRA}}$;
$t_{\mathsf{dep}} \in [8, 9)$: forces $B_0^2 = \top$ and $\mathcal{TTA}(\mathsf{PredAbs}_{[\varphi^*]}(\mathbf{A}^*))$ reduces to (38);
$t_{\mathsf{dep}} \in [9, 10)$: makes the whole formula $\mathcal{LRA}$-unsatisfiable, s.t. $\mathcal{TTA}(\mathsf{PredAbs}_{[\varphi^*]}(\mathbf{A}^*))$ is empty.

E.g., in the first case, if we set t_{dep} to 7.4 by conjoining $(t_0 = 7.4)$ to φ^*, then $\mathcal{TTA}(\mathsf{PredAbs}_{[\varphi^*]}(\mathbf{A}^*))$ contains only (36) and (37). Let (36) be the first assignment selected in the "for" loop, that is, $\mu^{\mathbf{A}^*} \overset{\text{def}}{=} (B_0^1 \wedge \neg B_0^2 \wedge \neg B_0^3 \wedge B_1^1 \wedge \neg B_1^2 \wedge \neg B_1^3)$. Propagating its truth values inside φ^* and w^* in (32) and simplifying the truth values by means of Simplify(), we get rid of most $\mathcal{LRA}$-literals in φ^*, obtaining thus[22]:

$$\varphi^*_{[\mu^{\mathbf{A}^*}]} = (t_0 = 7.4) \wedge [\![t_0 \in [7, 10)]\!]$$
$$\wedge (t_0 + x_1 + x_2 < 11)$$

[22] Note that $[\![t_0 \in [7, 10)]\!]$ and $[\![t_0 \in [7, 8)]\!]$ are made redundant by $(t_0 = 7.4)$; however, they do not affect the result.

$$\begin{aligned} &\wedge [\![t_0 \in [7,8)]\!] \wedge [\![x_1 \in [0.5,1)]\!] \\ &\wedge [\![t_0 + x_1 \in [7,8)]\!] \wedge [\![x_2 \in [1,1.5)]\!] \\ w^*_{[\mu^{\mathbf{A}^*}]} &= f^1_{l_0 l_1}(x_1) \cdot f^1_{l_1 l_2}(x_2) \end{aligned}$$

$\varphi^*_{[\mu^{\mathbf{A}^*}]}$ is a conjunction of $\mathcal{LRA}$-literals, so that the condition of the "if" is verified, then $\mathsf{WMI}_{\mathsf{nb}}$ can be invoked on it directly at the cost of one integration only, without further invoking another predicate abstraction and hence without running the internal "for", which would cost one integration for every internal loop. ◇

5.3. WMI-PA *vs.* WMI-AllSMT

As a general remark, comparing (34) with (18)—even if $\varphi^*, w^*, \mathbf{x}, \mathbf{A}^*$ were respectively $\varphi, w, \mathbf{x}, \mathbf{A}$—we note that in WMI-PA the restriction of the sum to $\mathcal{TTA}(\exists \mathbf{x}.\varphi^*)$ in (34) removes *a priori* all the assignments $\mu^{\mathbf{A}^*}$ which cannot be expanded by any assignment $\mu^{\mathcal{LRA}}$ s.t. $\mu^{\mathbf{A}^*} \wedge \mu^{\mathcal{LRA}}$ propositionally satisfies φ^* and $\mu^{\mathcal{LRA}}$ is $\mathcal{LRA}$-satisfiable, whose integrals would be 0-valued.

We argue that WMI-PA produces much less calls to the background integration procedure $\mathsf{WMI}_{\mathsf{nb}}(\mu^{\mathcal{LRA}}, w^*_{[\mu^{\mathbf{A}^*}]}|\mathbf{x})$ than WMI-AllSMT for two main reasons.

First, the size of $Atoms(\varphi^*_{[\mu^{\mathbf{A}^*}]})$ which is fed to PredAbs in (35) can be made much smaller than the number of $\mathcal{LRA}$-atoms in $Atoms(\varphi^*)$ fed to PredAbs in (33), since many $\mathcal{LRA}$-atoms are simplified out by $\mu^{\mathbf{A}^*}$. (E.g., $(x \le 1) \wedge (A_2 \vee (x \ge 0)))_{[A_2]}$ is simplified into $(x \le 1)$, so that $(x \ge 0)$ is eliminated.) Thus, for each $\mu^{\mathbf{A}^*}$, the number of assignments in the form $\mu^{\mathbf{A}^*} \wedge \mu^{\mathcal{LRA}}$ which are enumerated in (34)-(35) can be drastically reduced with respect to those enumerated in (33).

Second, with (35) it is possible to search for a set $\mathcal{TA}(...)$ of *partial* assignments, each of which substitutes 2^i total ones, i being the number of unassigned $\mathcal{LRA}$-atoms. Note that, unlike with Boolean atoms, we can safely produce partial assignments on $\mathcal{LRA}$-atoms because $w(\mathbf{x}, \mathbf{A})$ does not depend directly on them, since the integrals can be computed also on a partial assignment of the $\mathcal{LRA}$-atoms. (E.g., if $\varphi^*_{[\mu^{\mathbf{A}^*}]} \overset{\text{def}}{=} (x \ge 0) \wedge ((x \le 2) \vee (x \le 1))$, the partial assignment $\mu^{\mathcal{LRA}} \overset{\text{def}}{=} (x \ge 0) \wedge (x \le 2)$ prevents enumerating the two total ones $\mu^{\mathcal{LRA}} \wedge (x \le 1)$ and $\mu^{\mathcal{LRA}} \wedge \neg(x \le 1)$, computing one integral rather than two.

6. Related work

Most works on probabilistic inference in hybrid graphical models are either limited to joint Gaussian distributions [31], or perform approximate inference [1,24]. A recent line of research focused on developing exact inference algorithms for graphical models with mixtures of polynomials [42,45,49], using techniques like generalized joint-tree algorithms or symbolic variable elimination. The WMI formalism extends these approaches allowing to represent constraints in terms of arbitrary combinations of Boolean connectives.

The first solver for exact WMI [7], which is based on the original definition $\mathsf{WMI}_{\mathsf{old}}$ in §2.2, was a proof-of-concept relying on a simple block-clause strategy (WMI-BC in the following), which iteratively generates new models by adding the negation of the latest model to the formula for the following iteration. In the propositional (WMC) case, substantial efficiency gains can be obtained by leveraging component caching techniques [3,40], in which the weighted model counts of disjoint components of a formula are cached once and reused whenever the formula is encountered again in the computation. Unfortunately, these strategies are difficult to apply in the WMI case, because of the additional coupling induced by algebraic constraints. A recent work [9] did show substantial computational savings in adapting #DPLL with component caching from the WMC to the WMI case. The approach however works with purely piecewise polynomial densities, with no additional constraints between theory variables. In this setting, the reasoning part boils down to plain SAT, and an eager encoding of the piecewise structure allows to apply component caching at the propositional level and leave the integration of densities to the final steps of the decomposition procedure. Albeit effective, this solution cannot be applied whenever algebraic constraints exist between variables, e.g. their sum being within a certain range, a rather common situation in many practical cases (see §4). In the same paper, an approach equivalent to WMI-AllSMT applied to the original WMI formulation was shown to improve over the WMI-BC baseline. Nevertheless, further improvements are not possible without revising the formulation as we do in this work.

Closest to WMI are the work by Sanner and Abbasnejad [42] and the probabilistic inference modulo theories [19] framework. The former is the first approach for closed-form exact inference on hybrid graphical models with piecewise polynomial functions. It relies on a symbolic extension of the variable elimination algorithm implemented using extended algebraic decision diagrams (XADD) [43]. We will refer to this approach as SVE-XADD in the rest of the paper. A major limitation of SVE-XADD is that it requires to enumerate all paths from the root to the leaves in the XADD during computation, thus producing a blow-up both in memory requirements and number of alternatives to be evaluated. Although variable-elimination-based algorithms can cache intermediate results in principle, this is apparently not the case for the SVE-XADD algorithm. Indeed, our experimental evaluation (§7) shows that SVE-XADD has the worst performance among all competing approaches in terms of query execution times, and quickly exhausts available memory for increasing problem size. Probabilistic inference modulo theories [19] is a general framework where a combination of DPLL and symbolic variable elimination allows to perform probabilistic inference on a variety of theories. While initially focused on integer arithmetic, the system developed by the authors (called PRAiSE) was recently provided with support for arithmetic over the reals and polynomial densities. Our experimental evaluation (§7) shows that PRAiSE is always more efficient than SVE-XADD, and substantially outperforms WMI-BC and WMI-AllSMT when the relations between the continuous variables allow to efficiently decompose the integral by means of variable elimination steps, but it is largely inferior to our novel formulation in all tested scenarios.

7. Experiments

We have evaluated the performance of WMI-PA on both synthetic (§7.1) and real-world (§7.2 and §7.3) problems, comparing it with $\mathrm{WMI}_{\mathrm{old}}$ techniques and alternative symbolic approaches. These problems have been chosen also because they can be suitably encoded to be fed to all tools under test; in particular, they are very naturally encoded into $\mathrm{WMI}_{\mathrm{old}}$ without the exponential blowup described in §3.4.

In our empirical evaluation we compared the following tools:

- WMI-BC is our re-implementation of the $\mathrm{WMI}_{\mathrm{old}}$ procedure in [7];
- WMI-AllSMT and WMI-PA are our implementations of the procedures described in §5.1 and §5.2 respectively;
- SVE-XADD is the implementation of the algorithm in [43] provided by the authors, adapted in order to parse our input format;
- PRAiSE[23] is the implementation of Probabilistic Inference Modulo Theories [19] provided by the authors.

The implementations of WMI-BC, WMI-AllSMT and WMI-PA use MathSAT5[24] [16] to perform SMT reasoning and LattE Integrale[25] [33] to compute integrals of polynomials. To perform internal manipulations of the weight components, we used SymPy,[26] a Python library for symbolic mathematics. The software implementation of all algorithms, as well as all data and scripts for replicating the experiments in this paper are publicly available online.[27]

All experiments were run on a Virtual Machine with 7 cores running at a frequency of 2.2 GHz and 94 GB of RAM. The timeout was set at 10,000 seconds for each ⟨query, tool⟩ job pair. Importantly, comparing the numerical results of the tests it turned out that, when terminating, all tools returned the same values on the same queries (modulo roundings).

7.1. Synthetic setting

The synthetic setting is conceived in order to test the performance of the different tools on generic WMI problems. The setting we use here is more elaborate than the one we employed in [36], with the aim of making it more challenging for WMI-PA, in particular to force WMI-PA to enter more frequently its inner loop (the loop in the "else" case in Algorithm 1). Note however that the results in the simpler setting reported in [36] are qualitatively similar to the ones we report here, with an even more pronounced advantage of the WMI-based approaches over the symbolic alternatives (see the supplementary material for the results on that setting).

In what follows, let $A^{\mathbb{B}}$ denote a random Boolean atom drawn from $\mathbf{A}$, let $A^{\mathbb{R}}$ denote a random $\mathcal{LRA}$-atom over variables in $\mathbf{x}$, let $A^{\mathbb{B}/\mathbb{R}}$ denote a random Boolean or $\mathcal{LRA}$-atom. In this experiment, we used two recursive procedures to generate random formulas and nested weight functions with a given depth d:

$$\mathsf{rand}_{\varphi}(d) = \begin{cases} \bigoplus_{q=1}^{Q} \mathsf{rand}_{\varphi}(d-1) & \text{if } d > 0 \\ [\neg]A^{\mathbb{B}/\mathbb{R}} & \text{otherwise} \end{cases}$$

$$\mathsf{rand}_{w}(d) = \begin{cases} [\![\text{If } \mathsf{rand}_{\varphi}(d) \text{ Then } \mathsf{rand}_{w}(d-1) \text{ Else } \mathsf{rand}_{w}(d-1)]\!] & \\ \text{or} & \\ \mathsf{rand}_{w}(d-1) \otimes \mathsf{rand}_{w}(d-1) & \text{if } d > 0 \\ P_{\mathsf{random}}(\mathbf{x}, max_{\mathsf{deg}}) & \text{otherwise} \end{cases}$$

where $\bigoplus \in \{\bigvee, \bigwedge, \neg\bigvee, \neg\bigwedge\}$, $\otimes \in \{+, \cdot\}$ are randomly-picked Boolean and arithmetical operators respectively, "[¬]" means that a negation is added at random, "w_1 or w_2" means that one of the two alternative functions w_1 and w_2 is chosen randomly, and $P_{\mathsf{random}}(\mathbf{x}, max_{\mathsf{deg}})$ are random polynomials over $\mathbf{x}$ with maximum degree max_{deg}.

[23] http://aic-sri-international.github.io/aic-praise/.
[24] http://mathsat.fbk.eu/.
[25] https://www.math.ucdavis.edu/~latte/.
[26] http://www.sympy.org/.
[27] https://github.com/unitn-sml/wmi-pa.

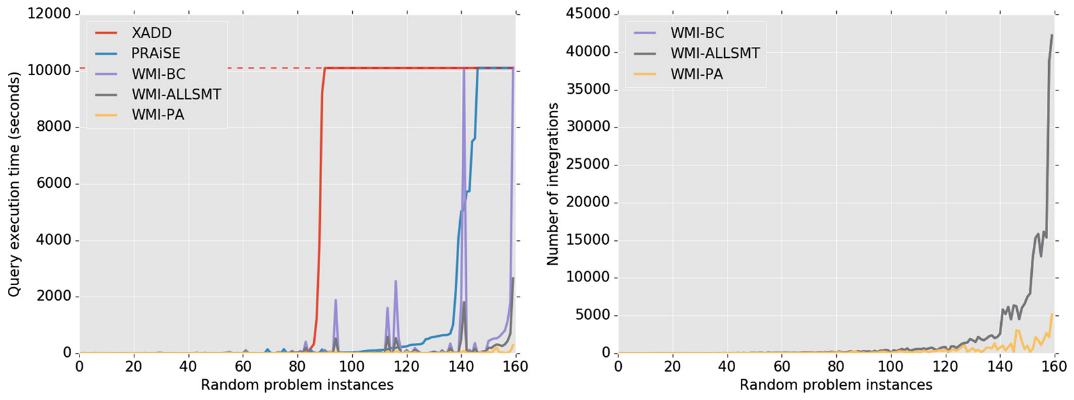


Fig. 5. Query execution times in seconds for all methods on the synthetic experiment (left); number of integrals (right) for WMI-BC, WMI-AllSMT and WMI-PA on the same instances.

Using the procedures above, we generated the problem instances as follows:

$$\chi(\mathbf{x}, \mathbf{A}) = \mathsf{rand}_{\varphi}(D) \wedge \bigwedge_{x \in \mathbf{x}} [\![x \in [l_x, u_x]]\!]$$

$$w(\mathbf{x}, \mathbf{A}) = \mathsf{rand}_{w}(D)$$

$$\varphi_{\mathsf{query}}(\mathbf{x}, \mathbf{A}) = \mathsf{rand}_{\varphi}(D)$$

where D is a parameter that control the depth of χ, w and φ_{query}, and $[l_x, u_x]$ are random lower and upper bounds for each variable $x \in \mathbf{x}$.

Fig. 5 (left) shows the query execution times on the randomly generated problem instances for all the methods. Instances are ordered by increasing hardness, measured as the running time of the slowest method. For instances in which the slowest method reaches the timeout, the second slowest method is used to order instances, and so on. (Here and in next figures, a value of 10,000 s denotes the fact that the procedure under test reached the timeout without producing a solution.)

In this experimental setting, the WMI methods achieve better performance with respect to the symbolic approaches, suggesting that the latter struggle with combinatorial reasoning, in contrast with the WMI approaches which rely on the full reasoning power of a state-of-the-art SMT solver.

Whereas WMI-AllSMT performs better than the baseline WMI-BC for the most difficult cases, WMI-PA achieves drastic speedups with respect to both the alternatives. Fig. 5 (right) reports the number of integrals computed by the three WMI methods. The curves for WMI-BC and WMI-AllSMT are indistinguishable, an expected result as the two formulations enumerate the same set of total truth assignments, with WMI-AllSMT doing it more efficiently. Conversely, the predicate abstraction steps of WMI-PA allow it to drastically reduce the number of assignments, and thus integrals to be computed. Note that a comparison with the symbolic approaches in terms of number of integrals is not possible because of their complex combination of variable elimination and integration steps.

7.2. *Strategic road network with fixed path*

In order to show the applicability of our method to real world tasks, we implemented the case study described in §4.1. The data was taken from the Strategic Road Network Dataset,[28] which provides a record of journey times on all the motorways managed by the English Highways Agency. From this dataset we extracted a graph between junctions whose edges were labeled with distributions of average journey times for each time interval (15 minutes long). For our experiments, we considered the largest strongly connected component of this graph, shown in Fig. 6. In this setting, the task is to perform queries of the form:

$$P((t_N \leq t_{\mathsf{arr}}) \mid t_0 = t_{\mathsf{dep}}, \{l_i\}_{i=0}^{N}),$$

that is, computing the probability of completing a fixed path $l_1, \ldots, l_N$ within t_{arr}, given the departure time t_{dep}. We encoded an equivalent formulation for PRAiSE and compared it with the WMI approaches. SVE-XADD was not considered in this setting because its execution times are prohibitive for all but the smallest path lengths. Another issue we encountered with SVE-XADD is that it often runs out of memory due to the size of the underlying XADDs.

The results in Fig. 7 show median, first and third quartiles of the query execution times, computed over 10 randomly generated queries for each path length. Whereas WMI-BC and WMI-AllSMT cannot scale to the path lengths handled by

[28] https://data.gov.uk/dataset/dft-eng-srn-routes-journey-times.

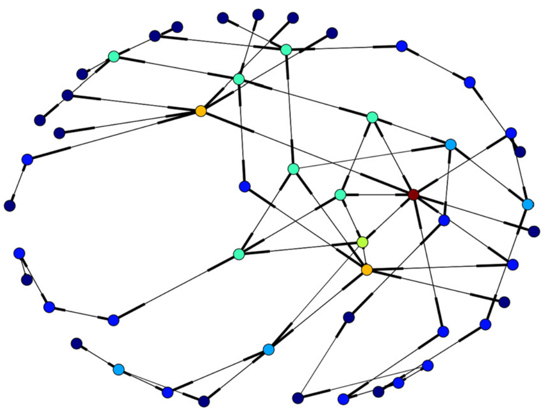
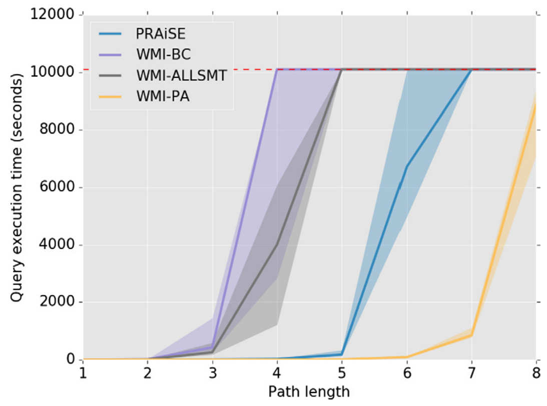


Fig. 6. The subgraph of the Strategic Road Network used in our experiments. Locations are colored according to their out degree. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)



#	PRAiSE	WMI		
		BC	AllSMT	PA
1	2	1	0	0
2	3	10	8	0
3	7	425	253	0
4	22	> 10000	3994	2
5	174	> 10000	> 10000	8
6	6722	> 10000	> 10000	86
7	> 10000	> 10000	> 10000	850
8	> 10000	> 10000	> 10000	8884

Fig. 7. Query execution times in seconds (1st quartile, median and 3rd quartile) in the Strategic Road Network setting with fixed path (left). Table showing the medians for each length (right).

PRAiSE, our approach is much faster than all alternatives, being able to compute queries up to two steps longer than PRAiSE without reaching the timeout.[29]

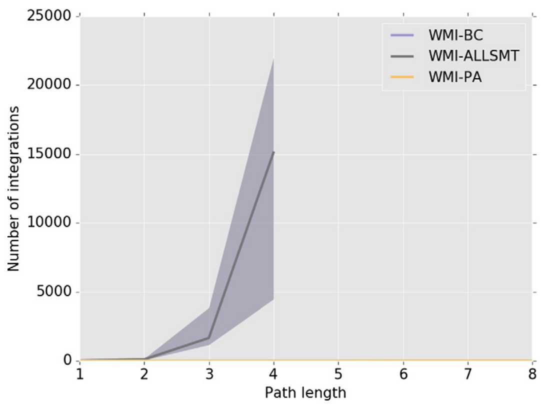
In contrast with the synthetic experiment, in this setting PRAiSE performs much better than WMI-BC and WMI-AllSMT. On the one hand, PRAiSE seems to benefit from the deterministic relationships between the journey time variables, being able to symbolically decompose the integration much more efficiently with relation to the synthetic experiment, in which the continuous variables can relate to each other in diverse and entangled ways. On the other hand, the number of overlapping intervals in this encoding makes the enumeration of total truth assignments performed by WMI-BC and WMI-AllSMT prohibitive (see §7.4).

Fig. 8 shows the number of integrals computed by the three WMI techniques. (As before, this data cannot be provided with PRAiSE.) Note that the plots for WMI-BC and WMI-AllSMT coincide, whereas that for WMI-PA cannot be distinguished from the x axis. From this, we observe that predicate abstraction techniques used in WMI-PA allow to drastically reduce the number of integrations.

7.3. Strategic road network with conditional plans

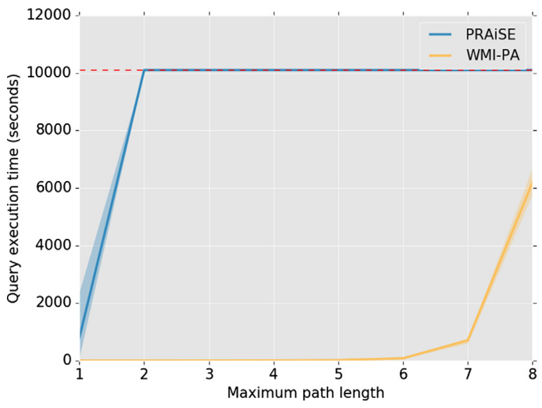
In order to further investigate the impact of combinatorial reasoning on the performance of WMI-PA and PRAiSE, we generalized the previous experiment to the case in which the path is not given in advance, using the encoding described in §4.2. In this experiment, we precomputed the conditional plan for each triple $\langle l_i, m, l_j \rangle$ using a greedy procedure based on expected journey time between adjacent locations. WMI-BC and WMI-AllSMT were not considered in this experiment,

[29] Note that the complexity of the query is due to the combination of the path length and the number of time intervals in which the time horizon is divided ($M = 12$ in these experiments). For paths of length 8, the total number of potential cases is $12^8 = 429,981,696$. Clearly, most of these cases are unfeasible and are thus ruled out by the SMT solver before the integration.



#	WMI		
	BC	AllSMT	PA
1	14	14	2
2	96	96	2
3	1653	1653	2
4	15105	15105	2
5	-	-	2
6	-	-	5
7	-	-	6
8	-	-	6

Fig. 8. Number of integrations (1st quartile, median and 3rd quartile) computed by the WMI methods in the Strategic Road Network setting with fixed path (left). Table showing the medians for each method and path length (right).



#	PRAiSE	WMI-PA
1	799	1
2	> 10000	2
3	> 10000	4
4	> 10000	6
5	> 10000	14
6	> 10000	77
7	> 10000	708
8	> 10000	6203

Fig. 9. Query execution times in seconds (1st quartile, median and 3rd quartile) in the Strategic Road Network setting with conditional plan (left). Table showing the medians for each maximum length (right).

given their inability to scale on the simpler fixed path experiment. Recall that in this setting, the task is answering queries of the form:

$$P(t_N \leq t_{\mathrm{arr}} \mid t_0 = t_{\mathrm{dep}}, l_{\mathrm{dep}}, l_{\mathrm{target}}, \mathsf{next}),$$

that is, computing the probability of reaching l_{target} within t_{arr}, leaving from l_{dep} at time t_{dep} and using the conditional plan encoded in next to make local decisions on the route to follow.

The results displayed in Fig. 9 show that with the conditional-plan setting WMI-PA drastically outperforms PRAiSE, the performance gaps being even superior than that with the fixed-plan setting in Fig. 7.[30] These results suggest that PRAiSE struggles with the heavier combinatorial aspect of this generalization. On the other hand, WMI-PA can prune the combinatorial space much more efficiently.

7.4. Discussion

The remarkable performance gaps of WMI-PA with respect to its competitors, in particular with respect to WMI-AllSMT and WMI-BC, can be explained in terms of what discussed in §5.3. In particular, we analyze the Strategic Road Network with Fixed Path setting of §7.2 by generalizing the scenario of Example 11. (In what follows we have omitted the literals from the query, which can simply be conjoined to each truth assignment.)

Consider $\varphi^*(\mathbf{x}, \mathbf{B})$ as in §4.1, and consider some $\mu^{\mathbf{A}^*} \in \mathcal{M}^{\mathbf{A}^*}$ as in Algorithm 1. Then, for every n, only one B^m_{n-1} is true (say, $B^{m_n}_{n-1}$) and all others are false in $\mu^{\mathbf{A}^*}$, so that $\bigwedge_{m=1}^{M}(B^m_{n-1} \leftrightarrow [\![t_n \in I^m]\!])$ forces $[\![t_n \in I^{m_n}]\!]$ to be true and all the others

[30] Comparing Fig. 9 with Fig. 7 one may get the (false) impression that the fixed plan problem is comparable or even harder for WMI-PA than the conditional-plan one. We note, however, that the two plots cannot be compared because in Fig. 7 the x axis represents the length of the (fixed) plan whereas in Fig. 9 it represents the *maximum* plan length, which can be bigger than the length of the actual plans.

to be false, so that $\bigwedge_{m=1}^{M}([\![t_{n-1} \in I^m]\!] \rightarrow [\![x_n \in R^m_{l_{n-1},l_n}]\!])$ forces $[\![x_n \in R^{m_n}_{l_{n-1},l_n}]\!]$ to be true and satisfies all other constraints $([\![t_{n-1} \in I^m]\!] \rightarrow [\![x_n \in R^m_{l_{n-1},l_n}]\!])$ for $m \neq m_n$, with no need to assign truth values to the other constraints $[\![x_n \in R^m_{l_{n-1},l_n}]\!]$.

With WMI-AllSMT (and WMI-BC) only *total* truth assignments $\mu^{\mathbf{A}^*} \wedge \mu^{\mathcal{LRA}}$ are generated from $\mathcal{TTA}(\varphi^*)$. We note that there can be up to $(2M-1)^N$ such assignments sharing the same $\mu^{\mathbf{A}^*}$ with different $\mu^{\mathcal{LRA}}$ part, each of which must be integrated separately. In fact, consider one $\mu^{\mathbf{A}^*}$ as above. WMI-AllSMT is forced to enumerate all total assignments $\mu^{\mathbf{A}^*} \wedge \mu_1^{\mathcal{LRA}}, \mu^{\mathbf{A}^*} \wedge \mu_2^{\mathcal{LRA}}, \ldots$ extending $\mu^{\mathbf{A}^*}$ which cover all possible truth value combinations of the atoms in $[\![x_n \in R^m_{l_{n-1},l_n}]\!]$ with $m \neq m_n$ which are $\mathcal{LRA}$-consistent with $[\![x_n \in R^{m_n}_{l_{n-1},l_n}]\!]$, although a truth value assignment to these atoms is not necessary to satisfy the formula, as pointed out above. (Recall that the intervals $\{R^m_{l_{n-1},l_n}\}_m$ are not disjoint.) Depending on the possible overlappings of $R^{m_n}_{l_{n-1},l_n}$ with the other intervals $R^m_{l_{n-1},l_n}$ with $m \neq m_n$, there are up to $(2M-1)^N$ such potential combinations: the extreme case is where, for every n, the bounds of the intervals $R^m_{l_{n-1},l_n}$ are all different and for every $m \neq m_n$ $R^m_{l_{n-1},l_n} \subset R^{m_n}_{l_{n-1},l_n}$, so that $R^{m_n}_{l_{n-1},l_n}$ is partitioned into $2M-1$ sub-intervals, totaling $(2M-1)^N$ combinations. We stress the fact that this partitioning is unnecessary because inside $R^{m_n}_{l_{n-1},l_n}$ the weight function is not partitioned.

With WMI-PA, instead, the constraints $([\![t_{n-1} \in I^m]\!] \rightarrow [\![x_n \in R^m_{l_{n-1},l_n}]\!])$ for $m \neq m_n$ are removed from $\varphi^*_{[\mu^{\mathbf{A}^*}]}$ by Simplify(), so that $\varphi^*_{[\mu^{\mathbf{A}^*}]}$ is simplified into[31]

$$\bigwedge_{n=1}^{N}\left([\![t_{n-1} \in I^{m_n}]\!] \wedge \bigwedge_{m=1}^{m_n-1}(t_{n-1} \geq c_m) \wedge \bigwedge_{m=m_n+1}^{M} \neg(t_{n-1} \geq c_{m-1}) \wedge [\![x_n \in R^{m_n}_{l_{n-1},l_n}]\!]\right)$$

which is a conjunction of $\mathcal{LRA}$-literals (namely $\mu^{\mathcal{LRA}}$). Thus, with the fixed-path setting, WMI-PA generates only one $\mu^{\mathcal{LRA}}$ to integrate for every $\mu^{\mathbf{A}^*} \in \mathcal{M}^{\mathbf{A}^*}$.

A direct theoretical comparison of the WMI techniques with respect to the symbolic techniques in SVE-XADD [42] and PRAiSE [19] is not possible, because of the very different nature of such procedures—and of the fact that the code of PRAiSE is much more complex and sophisticated than the general algorithm described in [19]—so that we limit to express a few conjectures.

Concerning the performance of SVE-XADD, as mentioned in §6, we conjecture that its major limitation is that it requires to enumerate all paths from the root to the leaves in the XADD during computation, thus producing a blow-up both in memory requirements and number of alternatives to be evaluated.

An analysis of the performance difference between PRAiSE and WMI-BC/WMI-AllSMT is more difficult. On the one hand, to the best of our understanding of the algorithm in [19], PRAiSE supports some form of reasoning on partial $\mathcal{LRA}$-subassignments, which we conjecture to provide a good advantage with respect to WMI-BC and WMI-AllSMT on the road-network setting, where this feature is critical. On the other hand, where the above issue is less critical as with the synthetic setting, we conjecture that the usage of variable-elimination techniques might be less efficient than the Boolean decomposition plus numerical integrations, as done by WMI-BC and WMI-AllSMT.

Finally, when comparing PRAiSE and WMI-PA—in addition to what mentioned in the last paragraph—we conjecture that the superiority in performance might be due mostly to the two-step usage of predicate abstraction interleaved with formula simplification, which allows both for getting rid of most $\mathcal{LRA}$-atoms from $\varphi^*_{[\mu^{\mathbf{A}^*}]}$ and for enumerating *partial* assignments on them, so that to drastically prune the number of $\mathcal{LRA}$-assignments $\mu^{\mathcal{LRA}}$ produced and thus the number of integrals performed.

8. Conclusion and future work

In this paper we proposed a revised definition of WMI which addresses some theoretical and practical limitations of the original formulation. Building on the properties of the novel formulation, we developed an efficient WMI algorithm combining a substantial reduction in the number of integrations with their efficient enumeration. Experimental comparisons over synthetic and real-world data confirm the drastic efficiency improvement over existing alternatives.

A number of relevant research directions can be foreseen to improve the current framework. From a computational perspective, further efficiency improvements could be obtained by developing decomposition techniques combined with appropriate caching strategies. This is a highly non-trivial task, as component caching approaches from the WMC literature [3,40] cannot be easily adapted to the WMI case because of the coupling induced by algebraic constraints. Another direction to scale up WMI to larger domains is that of working on approximate strategies. Space decomposition strategies based on random parity constraints [46] have been successfully employed for approximate model counting [6,25] and its weighted alternative [12,21,22]. A first step in this direction has been recently made by Belle et al. [8], but further work is needed in order to fully combine the efficiency of sampling by parity constraints with that of reasoning with theory

[31] Recall from §4.1 that $[\![t_{n-1} \in I^{m_n}]\!] \overset{\text{def}}{=} (t_{n-1} \geq c_{m-1}) \wedge \neg(t_{n-1} \geq c_m)$; consequently, we have that $\neg[\![t_{n-1} \in I^{m_n}]\!] = \neg(t_{n-1} \geq c_{m-1}) \vee (t_{n-1} \geq c_m)$, which is simplified into $(t_{n-1} \geq c_m)$ if $m < m_n$ and into $\neg(t_{n-1} \geq c_{m-1})$ if $m > m_n$.

solvers while retaining approximation guarantees in the weighted setting. From an expressivity viewpoint, an obvious direction is that of generalizing WMI to deal with combinations of theories, like linear arithmetic over reals and integers. This is a rather straightforward extension as it only affects the base step of our algorithm (alternative approaches like PRAISE already provide this functionality). A promising but more challenging direction for further enhancement is that of incorporating function symbols, something which has been recently investigated in both WMC [10] and probabilistic inference modulo theories [18]. A complementary direction in terms of enhanced expressivity is that of extending the class of weight functions supported by the method. WMI currently works for $\mathsf{FI}^{\mathcal{LRA}}$ functions, i.e. functions which are feasibly integrable on every set of $\mathcal{LRA}$ atoms. Polynomials are $\mathsf{FI}^{\mathcal{LRA}}$ [4], and they are the weight functions currently supported by our WMI implementation. Arbitrary weight functions could in principle be tackled using hyper-rectangle decomposition approaches [14]. Here the weight function is approximated as the sum of weight functions over hyper-rectangles, which are easy to compute. The problem is that depending on the $\mathcal{LRA}$ atoms, a very large or even infinite number of hyper-rectangles can be needed. A recent work by Merrell et al. [34] showed that in the special case of independent Gaussian densities, appropriate orthogonal transformations can be used to align one of the faces of the boundary of the formula to one of the axes without affecting the value of the integral, thus reducing the number of hyper-rectangles needed for an accurate approximation.

Finally, WMI has a number of intriguing application scenarios. Probabilistic inference in hybrid graphical models is the original motivation for research on WMI. Probabilistic planning [20,29] on hybrid domains is also a natural application domain. Program analysis is another promising direction, as exemplified by the recent works on quantifying bias in probabilistic programs [2] and probabilistic program abstraction [28].

Funding

This work was partially supported by the EC project CogNet, 671625 (H2020-ICT-2014-2, Research and Innovation action).

Acknowledgements

We would like to thank Alberto Griggio for useful suggestions about the usage of MathSAT5, Samuel Kolb and Rodrigo de Salvo Braz for their support on the SVE-XADD and PRAISE tool respectively.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.artint.2019.04.003.

References

[1] Hadi Mohasel Afshar, Scott Sanner, Christfried Webers, Closed-form Gibbs sampling for graphical models with algebraic constraints, in: AAAI, 2016.
[2] Aws Albarghouthi, Loris D'Antoni, Samuel Drews, Aditya V. Nori, Quantifying program bias, CoRR (2017), arXiv:1702.05437.
[3] Fahiem Bacchus, Shannon Dalmao, Toniann Pitassi, Solving #SAT and Bayesian inference with backtracking search, J. Artif. Intell. Res. 34 (1) (2009) 391–442.
[4] Velleda Baldoni, Nicole Berline, Jesus De Loera, Matthias Köppe, Michéle Vergne, How to integrate a polynomial over a simplex, Math. Comput. 80 (273) (2011) 297–325.
[5] Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, Cesare Tinelli, Satisfiability modulo theories, in: Handbook of Satisfiability, IOS Press, 2009, pp. 825–885, chapter 26.
[6] Mihir Bellare, Oded Goldreich, Erez Petrank, Uniform generation of NP-witnesses using an NP-oracle, Inf. Comput. 163 (2) (2000) 510–526.
[7] Vaishak Belle, Andrea Passerini, Guy Van den Broeck, Probabilistic inference in hybrid domains by weighted model integration, in: IJCAI, 2015.
[8] Vaishak Belle, Guy Van den Broeck, Andrea Passerini, Hashing-based approximate probabilistic inference in hybrid domains, in: Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (UAI), 2015.
[9] Vaishak Belle, Guy Van den Broeck, Andrea Passerini, Component caching in hybrid domains with piecewise polynomial densities, in: AAAI, 2016.
[10] Vaishak Belle, Weighted model counting with function symbols, in: The Conference on Uncertainty in Artificial Intelligence (UAI 2017), 2017.
[11] Roberto Cavada, Alessandro Cimatti, Anders Franzén, Krishnamani Kalyanasundaram, Marco Roveri, R.K. Shyamasundar, Computing predicate abstractions by integrating BDDs and SMT solvers, in: FMCAD, 2007.
[12] Supratik Chakraborty, Daniel J. Fremont, Kuldeep S. Meel, Sanjit A. Seshia, Moshe Y. Vardi, Distribution-Aware Sampling and Weighted Model Counting for SAT, AAAI, 2014.
[13] Mark Chavira, Adnan Darwiche, On probabilistic inference by weighted model counting, Artif. Intell. 172 (6–7) (2008) 772–799.
[14] Dmitry Chistikov, Rayna Dimitrova, Rupak Majumdar, Approximate counting in smt and value estimation for probabilistic programs, in: Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems – vol. 9035, New York, NY, USA, 2015, Springer-Verlag, New York, Inc, 2015, pp. 320–334.
[15] Arthur Choi, Doga Kisa, Adnan Darwiche, Compiling probabilistic graphical models using sentential decision diagrams, in: Symbolic and Quantitative Approaches to Reasoning with Uncertainty, Springer, 2013, pp. 121–132.
[16] Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, Roberto Sebastiani, The MathSAT 5 SMT solver, in: TACAS, 2013.
[17] Adnan Darwiche, New advances in compiling CNF to decomposable negation normal form, in: Proceedings of ECAI, 2004, pp. 328–332.
[18] Rodrigo de Salvo Braz, Ciaran O'Reilly, Exact inference for relational graphical models with interpreted functions: lifted probabilistic inference modulo theories, in: The Conference on Uncertainty in Artificial Intelligence (UAI 2017), 2017.
[19] Rodrigo de Salvo Braz, Ciaran O'Reilly, Vibhav Gogate, Rina Dechter, Probabilistic inference modulo theories, in: IJCAI, 2016.
[20] Carmel Domshlak, Jörg Hoffmann, Probabilistic planning via heuristic forward search and weighted model counting, J. Artif. Intell. Res. 30 (2007) 565–620.

[21] Stefano Ermon, Carla P. Gomes, Ashish Sabharwal, Bart Selman, Embed and project: discrete sampling with universal hashing, in: NIPS, 2013, pp. 2085–2093.
[22] Stefano Ermon, Carla P. Gomes, Ashish Sabharwal, Bart Selman, Low-density parity constraints for hashing-based discrete integration, in: ICML, 2014, pp. 271–279.
[23] Daan Fierens, Guy Van den Broeck, Joris Renkens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, Luc De Raedt, Inference and learning in probabilistic logic programs using weighted Boolean formulas, Theory Pract. Log. Program. 15 (3) (2015) 358–401.
[24] Vibhav Gogate, Rina Dechter, Approximate inference algorithms for hybrid Bayesian networks with discrete constraints, in: UAI, 2005.
[25] Carla P. Gomes, Ashish Sabharwal, Bart Selman, Near-uniform sampling of combinatorial spaces using XOR constraints, in: NIPS, 2006, pp. 481–488.
[26] Susanne Graf, Hassen Saïdi, Construction of abstract state graphs with PVS, in: CAV, 1997.
[27] D.A. Hensher, K.J. Button, Handbook of Transport Modelling, Handbooks in Transport, Emerald Publishing Limited, 2007.
[28] Steven Holtzen, Todd Millstein, Guy Van den Broeck, Probabilistic program abstractions, in: Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI), August 2017.
[29] Nicholas Kushmerick, Steve Hanks, Daniel S. Weld, An algorithm for probabilistic planning, Artif. Intell. 76 (1) (1995) 239–286, Planning and Scheduling.
[30] Shuvendu K. Lahiri, Robert Nieuwenhuis, Albert Oliveras, SMT techniques for fast predicate abstraction, in: CAV, 2006.
[31] Steffen L. Lauritzen, Frank Jensen, Stable local computation with conditional Gaussian distributions, Stat. Comput. 11 (2) (2001) 191–203.
[32] Edward A. Lee, Cyber physical systems: design challenges, in: Proc. IEEE Symposium on Object Oriented Real-Time Distributed Computing, 2008, pp. 363–369.
[33] Jesus De Loera, Brandon Dutra, Matthias Koeppe, Stanislav Moreinis, Gregory Pinto, Jianqiu Wu, Software for exact integration of polynomials over polyhedra, ACM Commun. Comput. Algebra 45 (3/4) (2012) 169–172.
[34] David Merrell, Aws Albarghouthi, Loris D'Antoni, Weighted model integration with orthogonal transformations, IJCAI (2017) 4610–4616.
[35] Alejandro Molina, Antonio Vergari, Nicola Di Mauro, Sriraam Natarajan, Floriana Esposito, Kristian Kersting, Mixed Sum-Product Networks: A Deep Architecture for Hybrid Domains, 2018.
[36] Paolo Morettin, Andrea Passerini, Roberto Sebastiani, Efficient weighted model integration via SMT-based predicate abstraction, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, 2017, pp. 720–728.
[37] Christian Muise, Sheila A. McIlraith, J. Christopher Beck, Eric I. Hsu, Dsharp: fast d-DNNF compilation with sharpSAT, in: Advances in Artificial Intelligence, Springer, 2012, pp. 356–361.
[38] Quoc San Phan, Pasquale Malacaria, All-solution satisfiability modulo theories: applications, algorithms and benchmarks, in: 2015 10th International Conference on Availability, Reliability and Security, IEEE, August 2015, pp. 100–109.
[39] Parikshit Ram, Alexander G. Gray, Density estimation trees, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2011, pp. 627–635.
[40] Tian Sang, Fahiem Bacchus, Paul Beame, Henry A. Kautz, Toniann Pitassi, Combining component caching and clause learning for effective model counting, in: SAT, 2004.
[41] Tian Sang, Paul Beame, Henry A. Kautz, Performing bayesian inference by weighted model counting, in: AAAI, vol. 5, 2005, pp. 475–481.
[42] Scott Sanner, Ehsan Abbasnejad, Symbolic variable elimination for discrete and continuous graphical models, in: AAAI, 2012.
[43] Scott Sanner, Karina Valdivia Delgado, Leliane Nunes de Barros, Symbolic dynamic programming for discrete and continuous state MDPs, in: UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, July 14–17, 2011, 2011, pp. 643–652.
[44] Roberto Sebastiani, Lazy satisfiability modulo theories, J. Satisf. Boolean Model. Comput. 3 (3–4) (2007) 141–224.
[45] Prakash P. Shenoy, James C. West, Inference in hybrid bayesian networks using mixtures of polynomials, Int. J. Approx. Reason. 52 (5) (2011) 641–657.
[46] Michael Sipser, A complexity theoretic approach to randomness, in: STOC, ACM, 1983, pp. 330–335.
[47] Dan Suciu, Dan Olteanu, Christopher Ré, Christoph Koch, Probabilistic databases, Synth. Lect. Data Manag. 3 (2) (2011) 1–180.
[48] Sebastian Thrun, Wolfram Burgard, Dieter Fox, Probabilistic Robotics (Intelligent Robotics and Autonomous Agents), The MIT Press, 2005.
[49] Shenlong Wang, Alex Schwing, Raquel Urtasun, Efficient inference of continuous Markov random fields with polynomial potentials, in: NIPS, 2014.