Abstract

Weighted Model Integration (WMI) is a popular formalism aimed at unifying approaches for probabilistic inference in hybrid domains, involving logical and algebraic constraints. Despite a considerable amount of recent work, allowing WMI algorithms to scale with the complexity of the hybrid problem is still a challenge. In this paper we highlight some substantial limitations of existing state-of-the-art solutions, and develop an algorithm that combines SMT-based enumeration, an efficient technique in formal verification, with an effective encoding of the problem structure. This allows our algorithm to avoid generating redundant models, resulting in substantial computational savings. An extensive experimental evaluation on both synthetic and real-world datasets confirms the advantage of the proposed solution over existing alternatives.

1 INTRODUCTION

Weighted Model Integration [Belle et al., 2015] recently emerged as a unifying formalism for probabilistic inference in hybrid domains, characterized by both continuous and discrete variables and their relationships. The paradigm extends Weighted Model Counting (WMC) [Chavira and Darwiche, 2008], which is the task of computing the weighted sum of the set of satisfying assignments of a propositional formula, to deal with SMT formulas (e.g. [Barrett et al., 2009]). Whereas WMC can be made extremely efficient by leveraging component caching techniques [Sang et al., 2004] Bacchus et al., 2009], these strategies are hard to apply for WMI because of the tight coupling induced by the arithmetic constraints. Indeed, component caching approaches for WMI are restricted to fully factorized densities with few dependencies among continuous variables [Belle et al., 2016]. Another direction specifically targets acyclic [Zeng and Van den Broeck, 2019] or loopy [Zeng et al., 2020] pairwise models.

Exact solutions for more general classes of densities and constraints mainly leverage advancements in SMT technology or in knowledge compilation (KC) [Darwiche and Marquis, 2002]. WMI-PA [Morettin et al., 2017, 2019] relies on SMT-based Predicate Abstraction (PA) [Lahiri et al., 2006] to reduce the number of models to be generated and integrated over, and was shown to achieve substantial improvements over previous solutions. However, we show how WMI-PA has the major drawback of ignoring the structure of the weight function when pruning away redundant models. This seriously affects its simplification power when dealing with symmetries in the density. The use of KC for hybrid probabilistic inference was pioneered by Sanner and Abbasnejad [2012] and further refined in a series of later works [Kolb et al., 2018, Dos Martires et al., 2019, Kolb et al., 2020, Feldstein and Belle, 2021]. By compiling a formula into an algebraic circuit, KC techniques can exploit the structure of the problem to reduce the size of the resulting circuit, and are at the core of many state-of-the-art approaches for WMC [Chavira and Darwiche, 2008]. However, even the most recent solutions for WMI [Dos Martires et al., 2019, Kolb et al., 2020] have serious troubles in dealing with densely coupled problems, resulting in exponentially large circuits.

In this paper we introduce a novel algorithm for WMI that aims to combine the best of both worlds, by introducing weight-structure awareness into PA-based WMI. The main idea is to iteratively build a formula which mimics the conditional structure of the weight function, so as to drive the SMT-based enumeration algorithm preventing it from generating redundant models. An extensive experimental evaluation on synthetic and real-world datasets shows substantial computational advantages of the proposed solution over existing alternatives for the most challenging settings.

Our main contributions can be summarized as follows:
• We identify major efficiency issues of existing state-of-the-art WMI approaches, both PA and knowledge-compilation based.
• We introduce SA-WMI-PA, a novel WMI algorithm that combines PA with weight-structure awareness.
• We show how SA-WMI-PA achieves substantial computational improvements over existing solutions in both synthetic and real-world settings.

2 BACKGROUND

2.1 SMT AND PREDICATE ABSTRACTION

Satisfiability Modulo Theories (SMT) (see Barrett et al. [2009]) consists in deciding the satisfiability of first-order formulas over some given theory. For the context of this paper, we will refer to quantifier-free SMT formulas over linear real arithmetic (LRA), possibly combined with uninterpreted function symbols (LRA ∪ EUF). We adopt the notation and definitions in Morettin et al. [2019]. We use B ∋ {T, ⊥} to indicate the set of Boolean value, whereas R indicates the set of real values. SMT(LRA) formulas combines Boolean variables A_i ∈ B and LRA atoms in the form (∑A_i) c_i x_i (where c_i are rational values, x_i are real variables in R and ▷◁ is one of the standard algebraic operators {=, ≠, <, >, ≤, ≥}) by using standard Boolean operators {¬, ∨, ∧, →, ↔}. In SMT(LRA ∪ EUF), LRA terms can be interleaved with uninterpreted function symbols. Some shortcuts are provided to simplify the reading. The formula (x_i ≥ l) ∧ (x_i ≤ u) is shortened into [x_i ∈ [l, u]].

Given an SMT formula φ, a total truth assignment μ is a function that maps every atom in φ to a truth value in B. A partial truth assignment maps only a subset of atoms to B.

2.2 WEIGHTED MODEL INTEGRATION (WMI)

Let x ≡ {x_1, ..., x_N} ∈ R^N and A ≡ {A_1, ..., A_M} ∈ R^M for some integers N and M. φ(x, A) denotes an SMT(LRA, A) formula over variables in A and x (subgroup of variables are admissible), while w(x, A) denotes a non-negative weight function s.t. R^N × R^M → R^+. Intuitively, w encodes a (possibly unnormalized) density function over A × x. Hereafter μ^A denotes a truth assignment on A, μ^LRA denotes a truth assignment on the LRA-atoms of φ, μ^LRA[φ] denotes (any formula equivalent to) the formula obtained from φ by substituting every Boolean value A_i with its truth value in μ^A and propagating the truth values through Boolean operators, and w[μ^A](x) = w(x, μ^A) is w restricted to the truth values of μ^A.

Given a theory T ∈ {LRA, LRA ∪ EUF}, the nomenclature TTA(φ) ≡ {μ_1, ..., μ_j} defines the set of T-consistent total assignments over both propositional and T atoms that propositionally satisfy φ; TTA(φ) = {μ_1, ..., μ_j} represents one set of T partial assignments over both propositional and T atoms that propositionally satisfy φ, s.t. every total assignment in TTA(φ) is a super-assignment of some of the partial ones in TTA(φ). Given by φ(x, A), TTA(∃x.φ) we mean the set of all total truth assignment μ^A on A s.t. φ[μ^A](x) is T-satisfiable, and by TTA(∀x.φ) a set of partial assignments s.t. every total assignment in TTA(∃x.φ) is a super-assignment of some of the partial ones in TTA(∃x.φ).

The Weighted Model Integral of w(x, A) over φ(x, A) is defined as follows [Morettin et al. 2019]:

\[ WMI(φ, w | x, A) \equiv \sum_{μ^A ∈ B^M} WMI_{nb}(φ[μ^A], w[μ^A] | x), \]

\[ = \sum_{μ^A ∈ TTA(∃x.φ)} WMI_{nb}(φ[μ^A], w[μ^A] | x) \]

\[ WMI_{nb}(φ, w | x) \equiv \int_{φ(x)} w(x) \, dx, \]

\[ = \sum_{μ^LRA ∈ TTA(φ)} \int_{μ^LRA} w(x) \, dx, \]

where the μ^A's are all total truth assignments on A, WMI_{nb}(φ, w | x) is the integral of w(x) over the set \{x | φ(x) is true\} (“nb” means “no-Booleans”).

We call a support of a weight function w(x, A) any subset of R^N × R^M out of which w(x, A) = 0, and we represent it as a LRA-formula χ(x, A). We recall that, consequently,

\[ WMI(φ ∧ χ, w | x, A) = WMI(φ, w | x, A). \]

We consider the class of feasibly integrable on LRA (FI(LRA)) functions w(x), which contain no conditional component, and for which there exists some procedure able to compute WMI_{nb}(μ^LRA, w | x) for every set of LRA literals on x. (E.g., polynomials are FI(LRA).) Then we call a weight function w(x, A), feasibly integrable under LRA conditions (FIUC(LRA)) iff it can be described in terms of a support LRA-formula χ(x, A) (T if not provided), a set Ψ ≡ {ψ_i(x, A)}_{i=1}^K of LRA conditions, in such a way that, for every total truth assignment μ^Ψ on A, w[μ^Ψ] is total and FI(LRA) in the domain given by the values of (x, A) which satisfy (χ ∧ μ^Ψ)[μ^A]. FIUC(LRA) functions are all the weight functions which can be described by means of arbitrary combinations of nested if-then-else’s on conditions in A and Ψ, s.t. each branch μ^Ψ results into a FI(LRA) weight function. Each μ^Ψ describes a portion of the domain of w, inside which w[μ^Ψ] is FI(LRA), and we say that μ^Ψ identifies w[μ^Ψ] in w.

In what follows we assume w.l.o.g. that FIUC(LRA) functions are described as combinations of constants, variables, standard mathematical operators +, −, ∗, / un-conditioned mathematical functions (e.g., exp, sin, ...), conditional expressions in the form (If ψ_i Then t_{i1} Else t_{i2}) whose conditions ψ_i are LRA formulas and terms t_{i1}, t_{i2} are FIUC(LRA).
Algorithm 1 WMI-PA(φ, w, x, A)

1: (φ*, w*, A*) ← LabelConditions(φ, w, x, A)
2: \( \mathcal{M}^{A*} ← \mathcal{TTA}(\text{PredAbs}_{\phi^*}|(A^*)) \)
3: \( vol ← 0 \)
4: \( \text{for } \mu^{A*} ∈ \mathcal{M}^{A*} \text{ do} \)
5: \( \text{Simplify}(\phi_{[\mu^{A*}]}^*) \)
6: \( \text{if } \text{LiteralConjunction}(\phi_{[\mu^{A*}]}^*) \text{ then} \)
7: \( \text{vol } ← \text{vol } + \text{WMI}_{\text{nb}}(\phi_{[\mu^{A*}]}^*, w_{[\mu^{A*}]}^*, x) \)
8: \( \text{else} \)
9: \( \mathcal{M}^{LRA} ← \mathcal{TTA}(\text{PredAbs}_{\phi_{[\mu^{A*}]}^*}(\text{Atoms}(\phi_{[\mu^{A*}]}^*))) \)
10: \( \text{for } \mu^{LRA} ∈ \mathcal{M}^{LRA} \text{ do} \)
11: \( \text{vol } ← \text{vol } + \text{WMI}_{\text{nb}}(\mu^{LRA}, w_{[\mu^{A*}]}^*, x) \)
12: return \( \text{vol} \)

2.3 WMI VIA PREDICATE ABSTRACTION

WMI-PA is an efficient WMI algorithm presented in Morettin et al. [2017, 2019] which exploits SMT-based predicate abstraction. Let \( w(x, A) \) be a FLU\( C^{LRA} \) function as above. WMI-PA is based on the fact that

\[
\text{WMI}(\phi, w|x, A) = \sum_{\mu^{A*} ∈ \mathcal{TTA}(\exists x. \phi^*)} \text{WMI}_{\text{nb}}(\phi_{[\mu^{A*}]}^*, w_{[\mu^{A*}]}^*, x) \quad (6)
\]

\[
\phi^* \overset{\text{def}}{=} \phi ∧ χ ∧ \bigwedge_{k=1}^{K} (B_k \leftrightarrow \psi_k) \quad (7)
\]

\( A^* \overset{\text{def}}{=} A ∪ B \) s.t. \( B \overset{\text{def}}{=} \{B_1, ..., B_K\} \) are fresh propositional atoms and \( w^*(x, A ∪ B) \) is the weight function obtained by substituting in \( w(x, A) \) each condition \( \psi_i \) with \( B_k \).

The pseudocode of WMI-PA is reported in Algorithm 1. First, the problem is transformed (if needed) by labeling all conditions \( \Psi \) occurring in \( w(x, A) \) with fresh Boolean variables \( B \). After this preprocessing stage, the set \( \mathcal{M}^{A*} \overset{\text{def}}{=} \mathcal{TTA}(\exists x. \phi^*) \) is computed by invoking SMT-based predicate abstraction [Lahiri et al. 2006], namely \( \mathcal{TTA}(\text{PredAbs}_{\phi^*}(A^*)) \). Then, the algorithm iterates over each Boolean assignment \( \mu^{A*} ∈ \mathcal{M}^{A*} \). \( \phi_{[\mu^{A*}]}^* \) is simplified by the Simplify procedure. Then, if \( \phi_{[\mu^{A*}]}^* \) is already a conjunctive form, the algorithm directly computes its contribution to the volume by calling \( \text{WMI}_{\text{nb}}(\phi_{[\mu^{A*}]}^*, w_{[\mu^{A*}]}^*, x) \). Otherwise, \( \mathcal{TTA}(\phi_{[\mu^{A*}]}^*) \) is computed as \( \mathcal{TTA}(\text{PredAbs}_{\phi_{[\mu^{A*}]}^*}(\text{Atoms}(\phi_{[\mu^{A*}]}^*))) \) to produce partial assignments, and the algorithm iteratively computes contributions to the volume for each \( \mu^{LRA} \). We refer the reader to [Morettin et al. 2019] for more details.

Notice that in the actual implementation the potentially large sets \( \mathcal{M}^{A*} \) and \( \mathcal{M}^{LRA} \) are not generated explicitly. Rather, their elements are generated, integrated and then dropped one-by-one, so that to avoid space blowup.

3 EFFICIENCY ISSUES

3.1 KNOWLEDGE COMPILATION

We start our analysis of WMI techniques by noticing a major problem with existing KC approaches for WMI [Dos Santos et al. 2019, Kolb et al. 2020], in that they tend easily to blow up in space even with simple weight functions. Consider, e.g., the case in which

\[
w(x, A) = \prod_{i=1}^{N} (\text{if } \psi_i \text{ then } w_{i1}(x) \text{ else } w_{i2}(x)) \quad (8)
\]

where the \( \psi_i \)s are \( LRA \) conditions on \( \{x, A\} \) and the \( w_{i1}, w_{i2} \) are generic functions on \( x \). First, the decision diagrams do not interleave arithmetical and conditional operators, rather they push all the arithmetic operators below the conditional ones. Thus with (8) the resulting decision diagrams consist of \( 2^N \) branches on the \( \psi_i \), s.t. they do not eliminate a priori the useless branches consisting in \( LRA \)-inconsistent combinations of \( \psi_i \), which can be up to exponentially many.

With WMI-PA, instead, the representation of (8) does not grow in size, because FLU\( C^{LRA} \) functions allow for interleaving arithmetical and conditional operators. Also, the SMT-based enumeration algorithm does not generate \( LRA \)-inconsistent assignments on the \( \psi_i \). We stress the fact that (8) is not an artificial scenario: rather, e.g., this is the case of the real-world logistics problems in [Morettin et al. 2019].

3.2 WMI-PA

We continue our analysis by noticing a major deficiency also of the WMI-PA algorithm, that is, it fails to leverage the structure of the weight function to prune the set of models to integrate over. We illustrate the issue by means of a simple example (see Figure 1).

Example 1 Let \( \phi \overset{\text{def}}{=} \top, \chi \overset{\text{def}}{=} \left[ x_1 \in [0, 2] \right] \land \left[ x_2 \in [0, 3] \right] \) (Figure 7(a)) and let \( w(x, A) \) be a tree-structured weight function defined as in Figure 7(b). To compute WMI(\( \varphi \land \chi, w|x, A \)), six integrals have to be computed:

\[
\begin{align*}
x_1^2x_2 & \text{ on } \left[ x_1 \in [1, 2] \right] \land \left[ x_2 \in [1, 3] \right] \quad (\text{if } A_1 = \top) \\
x_1^2x_3 & \text{ on } \left[ x_1 \in [1, 2] \right] \land \left[ x_2 \in [0, 1] \right] \quad (\text{if } A_1 = \top) \\
x_1x_2^2 & \text{ on } \left[ x_1 \in [0, 1] \right] \land \left[ x_2 \in [2, 3] \right] \quad (\text{if } A_1 = \top) \\
x_1x_2^2 & \text{ on } \left[ x_1 \in [0, 1] \right] \land \left[ x_2 \in [0, 2] \right] \quad (\text{if } A_1 = \top) \\
x_1x_2 & \text{ on } \left[ x_1 \in [0, 2] \right] \land \left[ x_2 \in [0, 3] \right] \quad (\text{if } A_1 = \bot, A_2 = \top) \\
x_1x_2 & \text{ on } \left[ x_1 \in [0, 2] \right] \land \left[ x_2 \in [0, 3] \right] \quad (\text{if } A_1 = \bot, A_2 = \bot) 
\end{align*}
\]

When WMI-PA is used (Algorithm 1), applying LabelCon-
Figure 1: Example highlighting the efficiency issues of the WMI-PA algorithm. (a) definition of formula $\varphi$ (trivially true) and support $\chi$. (b) definition of the weight function $w(x, A)$. Round nodes indicate if-then-else conditions, with true and false cases on the right and left outgoing edges respectively. Squared nodes indicate $F^L_{\mathcal{RA}}$ weight functions. (c) novel version of the formula $\varphi^*(x, A \cup B)$ after the application of the $\text{LabelConditions}(..)$ step of WMI-PA. (d) novel version of the weight function $w^*(x, A \cup B)$, where all $LRA$ conditions have been replaced with the fresh Boolean variables introduced in $\varphi^*(x, A \cup B)$. (e) List of assignments obtained by WMI-PA on $A \cup B$. Notice the amount of assignments sharing the same $F^L_{\mathcal{RA}}$ weight function.

ditions(...) we obtain Figure 1(c):

$$\varphi^*(x, A \cup B) \overset{\text{def}}{=} \varphi \land \chi \land B_1 \leftrightarrow (x_1 \geq 1) \land B_2 \leftrightarrow (x_2 \geq 1) \land B_3 \leftrightarrow (x_2 \geq 2)$$

and the weight function $w^*(x, A \cup B)$ shown in Figure 1(d). Then, by applying $TTA(PredAbs_\varphi(A^*))$ (row 2) we obtain 24 total assignments $\mathcal{M}^*_{\varphi^*}$ on $A \cup B$, as shown in Figure 1(e). Notice that WMI-PA uselessly splits into 2 parts the integrals on $x_1^2x_2$ and $x_1x_2^2$ and into 6 parts the integral on $2x_1x_2$ and on $3x_1x_2$. Also, it repeats the very same integrals for $\{A_1, A_2, \ldots\}$ and $\{A_1, \neg A_2, \ldots\}$.

We highlight two facts. First, WMI-PA enumerates total truth assignments on the Boolean atoms $A \cup B$ in $TTA(\exists x.\varphi^*)$ (row 2 in Algorithm 1), assigning also unnecessary values. Second, WMI-PA labels $LRA$ conditions in $w$ by means of fresh Boolean atoms $B$ (row 1 in Algorithm 1). This forces the enumerator to assign all their values in every assignment, even when not necessary.

The key issue about WMI-PA is that the enumeration of $TTA(\exists x.\varphi^*)$ in (a) and of $TTA(\varphi^*_{\mu})$ in (b) (rows 2 and 11 in Algorithm 1) is not aware of the conditional structure of the weight function $w$, in particular, it is not aware of the fact that often partial assignments to the set of conditions in $w^*$ (both Boolean and $LRA$) are sufficient to identify the value of a $\text{FIUC}^L_{\mathcal{RA}}$ function (e.g. $\{A_1, B_1, B_2\}$ suffices to identify $x_1^2x_2$, or $\{A_1, A_2\}$ suffices to identify $2x_1x_2$), so that it is forced to enumerate all total assignments extending them (e.g. $\{A_1, A_2, B_1, B_2, B_3\}$ and $\{A_1, A_2, B_1, B_2, \neg B_3\}$).

Thus, to cope with this issue, we need to modify WMI-PA to make it aware of the conditional structure of $w$.

4 MAKING WMI-PA WEIGHT-STRUCTURE AWARE

The key idea to prevent total enumeration works as follows. We do not rename with $B$ the conditions $\Psi$ in $w$ and, rather than enumerating total truth assignments for $\exists x.\varphi^*$ as in (a), (b), we enumerate partial assignments for $\exists x.\varphi^{**}$ where $\varphi^{**} \equiv \varphi \land \chi \land \text{sk}(w)$ and $\text{sk}(w)$ – which we call the conditional skeleton of $w$ – is a $LRA$ formula s.t.:

(a) its atoms are all and only the conditions in $\Psi$,
(b) is $LRA$-valid, so that $\varphi^* \chi$ is equivalent to $\varphi \land \chi \land \text{sk}(w)$,
(c) any partial truth value assignment $\mu$ to the conditions $\Psi$ which makes $\text{sk}(w)$ true is such that $w_{[\mu]}$ is $F^L_{\mathcal{RA}}$.

Thus, we have that (2) can be rewritten as:

$$\text{WMI}(\varphi, w|x, A) = \sum_{\mu \in \mathcal{T}_A(\exists x.\varphi^{**})} 2^{\mathcal{A}[\mu]} \cdot \text{WMI}_{nb}(\varphi^*_{[\mu]}, w_{[\mu]}|x) \quad (9)$$

$$\varphi^{**} \overset{\text{def}}{=} \varphi \land \chi \land \text{sk}(w) \quad (10)$$

where $\mathcal{A}[\mu]$ is the number of Boolean atoms in $A$ that are not assigned by $\mu$. Condition (c) guarantees that $\text{WMI}_{nb}(\varphi^{**}_{[\mu]}, w_{[\mu]}|x)$ in (9) can be directly computed, without further partitioning. The $2^{\mathcal{A}[\mu]}$ factor in (9) resembles the fact that, if some Boolean atom $A_i \in A$ is not assigned in $\mu$, then $\text{WMI}_{nb}(\varphi^{**}_{[\mu]}, w_{[\mu]}|x)$ should be counted twice because $\mu$ represents two assignments $\mu \cup \{A_i\}$ and $\mu \cup \{\neg A_i\}$ which would produce two identical integrals.

Notice that logic-wise $\text{sk}(w)$ is non-informative because it is a valid formula. Nevertheless, the role of $\text{sk}(w)$ is to mimic the structure of $w$ so that to “make the enumerator aware of the presence of the conditions $\Psi$”, forcing every

\[1\]E.g., the partial assignment $\mu \overset{\text{def}}{=} \{A_1, (x_1 \geq 1), (x_2 \geq 1)\}$ in Example 1 is such that $w_{[\mu]} = x_1^2x_2$, which is $F^L_{\mathcal{RA}}$.\[1\]
assignment $\mu$ to assign truth values also to these conditions which are necessary to make $w_\mu[F]_{CRA}$ and hence make $\text{WM}_{\text{base}}(\varphi^*_{\mu}, w_\mu, x)$ directly computable, without further partitioning.

An important issue is to avoid $\text{sk}(w)$ blow up in size. E.g., one could use as $\text{sk}(w)$ a formula encoding the conditional structure of an XADDs or (F)XSDSs, but this may cause a blow up in size, as discussed in Section 3.1.

In order to prevent such problems, we do not generate $\text{sk}(w)$ explicitly. Rather, we build it as a disjunction of partial assignments over $\Psi$ which we enumerate progressively. To this extent, we define $\text{sk}(w) \overset{\text{def}}{=} \exists y. [y = w]$ where $[y = w]$ is a formula on $A, x, y$ s.t. $y \overset{\text{def}}{=} \{y_1, y_2, \ldots, y_k\}$ is a set of fresh variables. Thus, $\text{TA}(\exists x. (\varphi \land \Psi \land [y = w]))$ can be computed as $\text{TA}(\exists x. (\varphi \land \Psi \land [y = w]))$ because the $y$'s do not occur in $\varphi \land \Psi$, with no need to generate $sk(w)$ explicitly. The enumeration of $\text{TA}(\exists x. (\varphi \land \Psi \land [y = w]))$ is performed by the very same SMT-based procedure used in [Morettin et al., 2019].

$[y = w]$ is obtained by taking $(y = w)$, s.t. $y$ is fresh, and recursively substituting bottom-up every conditional term (If $\psi$ Then $t_1$ Else $t_2$) in it with a fresh variable $y_i \in y$, adding the definition of $(y_i = (If \psi_i \ Then \ t_1 \ Else \ t_2))$ as

$$(-\psi_i \lor y_i = t_1) \land (\psi_i \lor y_i = t_2). \quad (11)$$

This labeling&rewriting process, which is inspired to labeling CNF-ization [seitin, 1968], guarantees that the size of $[y = w]$ is linear wrt. that of $w$. E.g., if $[8]$ holds, then $[y = w]$ is $(y = \prod_{i=1}^N y_i) \land \bigwedge_{i=1}^N (\neg \psi_i \lor y_i = w_i(x)) \land (\psi_i \lor y_i = w_i(x)).$

One problem with the above definition of $[y = w]$ is that it is not a $\text{LRA}$ formula, because $w$ may include multiplications or even transcendental functions out of the conditions $\Psi$, which makes SMT reasoning over it dramatically hard or even undecidable. We notice, however, that when computing $\text{TA}(\exists x. (\varphi \land \Psi \land [y = w]))$ the arithmetical functions (including operators $+, -, \cdot, /$) occurring in $w$ out of the conditions $\Psi$ have no role, since the only fact that we need to guarantee for the validity of sk(w) is that they are indeed functions, so that $\exists y. (y = f(\ldots))$ is always valid. In substance, during the enumeration we are interested only in the truth values of the conditions $\Psi$ in $\mu$ which make $w_\mu[F]_{CRA}$, regardless the actual values of $w_\mu$. Therefore we can safely substitute condition-less arithmetical functions (including operators $+, -, \cdot, /$) with some fresh uninterpreted function symbols, obtaining a $\text{LRA} \cup \text{ELUF}$-formula $[y = w]_{ELUF}$, which is relatively easy to solve by standard SMT solvers [Barrett et al., 2009]. It is easy to see that a partial assignment $\mu$ evaluating $[y = w]$ to true is $\text{LRA}$-satisfiable iff its corresponding assignment $\mu_{\text{ELUF}}$ is $\text{LRA} \cup \text{ELUF}$-satisfiable. Therefore, we can modify the enumeration procedure into $\text{TA}(\exists x. (\varphi \land \Psi \land [y = w]_{ELUF})).$

Finally, we enforce the fact that the two branches of an if-then-else are alternative by adding to (11) a mutual-exclusion constraint $\neg(y_i = t_1) \lor \neg(y_i = t_2)$, so that the choice of the function is uniquely associated to the list of decisions on the $\psi_i$s. The procedure producing $[y = w]_{ELUF}$ is described in detail in Appendix, Algorithm 1.

Example 2 Consider the problem in Example 4 Figure 2 shows the relabeling process applied to the weight function $w$. The resulting $[y = w]_{ELUF}$ formula is:

$$[y = w](x \cup y, A) \overset{\text{def}}{=} \left(\begin{array}{ll}
(-A_1 \lor -\psi_1 \land \chi \land y_2 \geq 1) \lor (\neg A_2 \lor -\psi_2 \land \chi \land y_2 \geq 1) \\
\land (-A_1 \lor -\psi_1 \land \chi \land y_2 \geq 1) \lor (\neg A_2 \lor -\psi_2 \land \chi \land y_2 \geq 1) \\
\land (-A_1 \lor -\psi_1 \land \chi \land y_2 \geq 1) \lor (\neg A_2 \lor -\psi_2 \land \chi \land y_2 \geq 1) \\
\land (-A_1 \lor -\psi_1 \land \chi \land y_2 \geq 1) \lor (\neg A_2 \lor -\psi_2 \land \chi \land y_2 \geq 1)
\end{array}\right) \land
\left(\begin{array}{ll}
(A_1 \lor -\psi_1 \land \chi \land y_2 \geq 1) \lor (y_1 \lor f_1(x)) \\
\land (A_1 \lor -\psi_1 \land \chi \land y_2 \geq 1) \lor (y_1 \lor f_1(x)) \\
\land (A_1 \lor -\psi_1 \land \chi \land y_2 \geq 1) \lor (y_1 \lor f_1(x)) \\
\land (A_1 \lor -\psi_1 \land \chi \land y_2 \geq 1) \lor (y_1 \lor f_1(x))
\end{array}\right).$$

Figure 3 illustrates a possible enumeration process. The algorithm enumerates partial assignments satisfying $\varphi \land 
\chi \land [y = w]_{ELUF}$, restricted on the conditions $\Psi \overset{\text{def}}{=} \{A_1, A_2, (x_1 \geq 1), (x_2 \geq 1), (x_2 \geq 2)\}$, which is equivalent to enumerate $\text{TA}(\exists x. (\varphi \land \Psi \land [y = w]_{ELUF}))$. Assuming the enumeration procedure picks nondeterministic choices following the order of the above set and assigning positive values first, then in the first branch the following satisfying partial assignment is generated, in order: $\psi_i \overset{\text{def}}{=} (\neg y_3 \lor y_4 \lor \chi) \land (y_4 = f_3(x)) \lor (y_4 = f_3(x)) \lor (y_4 = f_3(x))$.

(Notice that, following the chains of true equalities, we have $y = y_3 = y_5 = y_1 = f_1(\chi)$.) Then the SMT

This boils down to the fact that $y$ occurs only in the top equation and as such it is free to assume arbitrary values, and that all arithmetic functions are total in the domain so that, for every possible values of $x$ a value for $y$ always exists iff there exists in the $\text{ELUF}$ version.

Like in Algorithm 2 we pick Boolean conditions first.

Here nondeterministic choices are underlined. The atoms in $\chi$ are assigned deterministically.
Overall, the algorithm enumerates the following weight functions (the leaves of the tree, highlighted in red) with \( \text{EUF} \) producing the assignment: the SMT-based enumeration algorithm.

Atoms ordered collection of partial assignments restricted to the clause \( \neg \phi \) restricted on the conditions in \( \Psi \). Then the blocking clause \( \neg A_1 \lor \neg (x_1 \geq 1) \lor \neg (x_2 \geq 1) \) is added to the formula, which prevents to enumerate the same subset again. This forces the algorithm to backtrack and generate \( \chi \cup \{(y = y_1), A_1, (y_2 = y_3), \neg (y_3 = y_4), (x_1 \geq 1), (y_3 = y_1), \neg (y_3 = y_2), \neg (x_2 \geq 1), (y_1 = f_{12}(x))\), \neg (y_1 = f_{11}(x))\}. 

producing the assignment: \( \{A_1, (x_1 \geq 1), \neg (x_2 \geq 1)\} \)

Overall, the algorithm enumerates the following ordered collection of partial assignments restricted to \( \text{Atoms}(\varphi \land \chi) \cup \Psi \):

\[
\chi \cup \{ A_1, (x_1 \geq 1), (x_2 \geq 1) \}, /\!/ y = ... = f_{11}(x) \\
\chi \cup \{ A_1, (x_1 \geq 1), \neg (x_2 \geq 1) \}, /\!/ y = ... = f_{12}(x) \\
\chi \cup \{ A_1, \neg(x_1 \geq 1), (x_2 \geq 2) \}, /\!/ y = ... = f_{21}(x) \\
\chi \cup \{ A_1, \neg(x_1 \geq 1), \neg(x_2 \geq 2) \}, /\!/ y = ... = f_{22}(x) \\
\chi \cup \{ \neg A_1, A_2 \}, /\!/ y = ... = f_{3}(x) \\
\chi \cup \{ \neg A_1, \neg A_2 \}, /\!/ y = ... = f_{4}(x) \\
\]

which correspond to the six integrals of Example 2. Notice that according to (2) the first four integrals have to be multiplied by 2, because the partial assignment \( \{A_1\} \) covers two total assignments \( \{A_1, A_2\} \) and \( \{A_1, \neg A_2\} \). Notice also that the disjunction of the six partial assignments, \( (A_1 \land (x_1 \geq 1) \land (x_2 \geq 1)) \lor \ldots \lor (\neg A_1 \land \neg A_2) \), matches the definition of \( \text{sk}(w) \), which we have computed by progressive enumeration rather than encoded a priori. 

We refer the reader to Lahiri et al. [2006] for more details on the SMT-based enumeration algorithm.

Based on the previous ideas, we develop SA-WMI-PA, a novel “weight structure aware” variant of WMI-PA. The
As with WMI-PA, we enumerate the assignments in two main steps: in the first loop (rows 3-7) we generate a set \( M^{\varphi} \) of partial assignments \( \mu^{\varphi} \) over the Boolean variables \( A \), s.t. \( \varphi^{\mu^{\varphi}} A \) is \( LRA \)-satisfiable and does not contain Boolean variables anymore. In Example 2, \( M^{\varphi} = \{ \{ A_1 \}, \{ \neg A_1, A_2 \}, \{ \neg A_1, \neg A_2 \} \} \). In the second loop (rows 11-19), for each \( \mu^{\varphi} \) in \( M^{\varphi} \) we enumerate the set \( M^{LRA} \) of \( LRA \)-satisfiable partial assignments satisfying \( \varphi^{\mu^{\varphi}} A \) (that is, on \( LRA \) atoms in \( \text{Atoms}(\varphi \land \chi) \cup \Psi \)), we compute the integral \( \text{WMI}_{\chi \cup \Psi}(\mu^{LRA}, w_{\mu^{\varphi} A}[x]) \), multiply it by the \( 2^{|\mu^{\varphi} A|} \) factor and add it to the result. In Example 2 if e.g. \( \mu^{\varphi} A = \{ A_1 \} \), \( TA(\text{PredAbs}_{\varphi^{\mu^{\varphi}} A}(\text{Atoms}(\varphi^{\mu^{\varphi}} A))) \) computes the four partial assignments \( \chi \cup \{ (x_1 \geq 1), (x_2 \geq 1) \}, \ldots, \chi \cup \{ (x_1 \geq 1), (x_2 \geq 2) \} \).

In detail, in row 2 we extend \( \varphi \land \chi \) with \( \| y = w \|_{J, LRA} \) to provide structure awareness. (We recall that, unlike with WMI-PA, we do not label \( LRA \) conditions with fresh Boolean variables.) Next, in row 3 we perform \( TA(\text{PredAbs}_{\varphi^{\mu^{\varphi}} A}(A)) \) to obtain a set \( M^{\varphi} \) of partial assignments restricted on Boolean atoms \( A \). Then, for each assignment \( \mu^{\varphi} A \in M^{\varphi} \) we build the (simplified) residual \( \varphi^{\mu^{\varphi}} A \). Since \( \mu^{\varphi} A \) is partial, \( \varphi^{\mu^{\varphi}} A \) is not guaranteed to be free of Boolean variables \( A \), as shown in Example 3. If this is the case, we simply add \( \mu^{\varphi} A \) to \( M^{\varphi} \), otherwise we invoke \( TA(\text{PredAbs}_{\varphi^{\mu^{\varphi}} A}(A)) \) to assign the remaining variables and conjoin each assignment \( \mu^{\varphi}_{\text{residual}} \) to \( \mu^{\varphi} A \), ensuring that the residual now contains only \( LRA \) atoms (rows 9-10). The second loop (rows 11-19) resembles the main loop in WMI-PA, with the only relevant difference that, since \( \mu^{\varphi} A \) is partial, the integral is multiplied by a \( 2^{|A \setminus \mu^{\varphi} A|} \) factor.

Notice that in general the assignments \( \mu^{\varphi} A \) are partial even if the steps in rows 9-10 are executed; the set of residual Boolean variables \( \varphi^{\mu^{\varphi}} A \) are a (possibly much smaller) subset of \( A \setminus \mu^{\varphi} A \) because some of them do not occur anymore in \( \varphi^{\mu^{\varphi}} A \) after the simplification, as shown in the following example.

**Example 3** Let \( \varphi \overset{\text{def}}{=} (A_1 \lor A_2 \lor A_3) \land (\neg A_1 \lor A_2 \land (x \geq 1)) \land (\neg A_2 \lor (x \geq 2)) \land (\neg A_3 \land (x \leq 3)) \), \( \chi \overset{\text{def}}{=} \| x_1 \in [0, 4] \|_{J, LRA} \) and \( w(x, A) \overset{\text{def}}{=} 1.0 \). Suppose \( TA(\text{PredAbs}_{\varphi^{\mu^{\varphi}} A}(A)) \) finds the partial assignment...
We highlight two main differences wrt. WMI-PA. First, XSDD and FXSDD [Kolb et al., 2020]. Each of these methods on the synthetic experiments (left); number of integrals (right) for WMI-PA and SA-WMI-PA.

GRALE for computing integrals. For the KC algorithms we use PSiPSI [Gehr et al., 2016] as symbolic computer algebra backend. All experiments are performed on an Intel Xeon Gold 6238R @ 2.20GHz 28 Core machine with 128GB of ram and running Ubuntu Linux 20.04. The code of SA-WMI-PA is freely available at https://github.com/unitn-sml/wmi-pa.

For improved readability, in both the experiments we report runtime using cactus plots, i.e. the single problem instances are increasingly sorted by runtime for each algorithm separately. We highlight how, by construction, problem instances of the same tick of the x-axis are not guaranteed to be the same for different algorithms. Steeper slopes of an algorithm curve means less efficiency.

5.1 SYNTHETIC EXPERIMENTS

We first evaluate our algorithm on random formulas and weights, following the experimental protocol of [Morettin et al., 2019]. We define two recursive procedures to generate LRA formulae and weight functions with respect to a positive integer number D, named depth:

\[
\begin{align*}
\varphi(D) & = \begin{cases} 
\bigoplus_{q=1}^{Q} r_{\varphi}(D-1) & \text{if } D > 0 \\
[-]_{A^{B/R}} & \text{otherwise}
\end{cases} \\
r_w(D) & = \begin{cases} 
\begin{cases} 
[\text{If } r_{\varphi}(D) \text{ Then } r_w(D-1) \text{ Else } r_w(D-1)] \\
[\text{or } r_w(D-1) \otimes r_w(D-1)]
\end{cases} & \text{if } D > 0 \\
P_{\text{random}}(x) & \text{otherwise}
\end{cases}
\end{align*}
\]

where \( \bigoplus \in \{\vee, \wedge, \neg \vee, \neg \wedge\} \), \( \otimes \in \{+, \cdot\} \) and \( P_{\text{random}}(x) \) is a random polynomial function. Using these procedures we generate instances of synthetic problems:

\[
\begin{align*}
\chi(x, A) & = r_{\varphi}(D) \land \bigwedge_{x \in x \in [l_x, u_x]} \\
w(x, A) & = r_w(D) \\
\varphi_{\text{query}}(x, A) & = r_{\varphi}(D)
\end{align*}
\]

where \( l_x, u_x \) are real numbers such that \( \forall x, l_x < u_x \).
We considered a selection of hybrid datasets from the UCI repository [Dua and Graff 2017], reported in Table 1 in the Appendix. Following the approach of Morettin et al. [2020], discrete numerical features were relaxed into continuous variables, while $n$-ary categorical features are one-hot encoded with $n$ binary variables.

After learning a DET on each dataset, we generated a benchmark of increasingly complex queries, 5 for each dataset, involving a ratio $H \in \{0.25, 0.5, 0.75, 1\}$. More specifically, the queries are linear inequalities involving a number of variables $k = \max(1, \lfloor H \cdot |x| \rfloor)$. Figure 5 depicts the runtime of the algorithms for $H \in \{0.25, 0.5, 0.75, 1\}$. Timeout was set to 1200 seconds. KC approaches have an edge for the simplest cases ($H \leq 0.5$) in which substantial factorization of the integrals is possible. Contrarily to many other probabilistic models, which are akin to the case in Section 5.1 DETs are well-suited for KC-based inference, due to the absence of arithmetic operations in the internal nodes. When the coupling between variables increases, however, the advantage of decomposing the integrals is overweight by the combinatorial reasoning capabilities of SA-WMI-PA. We remark that SA-WMI-PA is agnostic of the underlying integration procedure, and thus in principle it could also incorporate a symbolic integration component.

6 CONCLUSION

We presented the first SMT-based algorithm for WMI that is aware of the structure of the weight function. This is particularly beneficial when the piecewise density defined on top of SMT(LRA) constraints is deep, as it is often the case with densities learned from data. We evaluated our algorithmic ideas on both synthetic and real-world problems, obtaining state-of-the-art results in many settings of practical interest. Providing unprecedented scalability in the evaluation of complex probabilistic SMT problems, this contribution directly impacts the use of WMI for the probabilistic verification of systems. While the improvements described in this work drastically reduce the number of integrations required to compute a weighted model integral, the integration itself remains an obstacle to scalability. Combining the fast combinatorial reasoning offered by SMT solvers with symbolic integration is a promising research direction.

Acknowledgements

This research was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215 and from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 694980) SYNTH: Synthesising Inductive Data Models). The authors would like to thank Pedro Zuidberg Dos Martires for his help with XSDDs and FXSDDs.
References


