Lifted Reasoning Meets Weighted Model Integration - Supplementary Material

Jonathan Feldstein[1] **Vaishak Belle**[1]

[1]University of Edinburgh, United Kingdom

A EXAMPLE OF WMI

Example 3. *We can extend Example 1 to the hybrid setting the following way. Let us assume the propositional theory* $\triangle$ *Belle et al. [2015]:*

$$p \vee (0 \leq x \leq 10),$$

with $w(p) = 0.1$, $w(\neg p) = 2x$, $w(q) = 1$ *and* $w(\neg q) = 0$, *where* q *is the propositional abstraction of* $(0 \leq x \leq 10)$. *Then there are three models* $M_1 = \{p, \neg q\}, M_2 = \{\neg p, q\}, M_3 = \{p, q\}$, *and the model weight is*

$$\text{WMI}(\triangle, w) = 0 + \int_0^{10} 2x dx + \int_0^{10} 0.1 dx = 101$$

B KNOWLEDGE COMPILATION

The input of the compilation algorithm which builds a FO-sda-DNNF ciruit is a FO-CNF formula. The algorithm consist of a set of operations on formulas in CNF, each operation equipped with a precondition. The algorithm repeatedly applies these auxiliary operations while their preconditions are satisfied, ensuring that the preconditions of the operations do not hold in the output, and transforms the sets of clauses into a directed-acyclic graph by caching and reusing nodes. The basis of the algorithm is the well-known unification based first-order resolution algorithm, which computes the most general unifier (mgu) of two FO atoms.

Definition 5 (Most general unifier in FOL)**.** *A unifier of two atoms is a substitution that, when applied to both atoms, makes them identical. Let* $\mathcal{U}(a, b)$ *denote the set of unifiers of atoms* a *and* b. *Then the most general unifier (mgu) of FO atoms* a *and* b *is* ψ, *iff* $\forall \phi \in \mathcal{U}(a, b), \exists \phi' : \phi = \psi\phi'$.

The unification algorithm is used in many operations over formulas, such as SPLIT, CONDITIONING, UNIT PROPAGATION and SHATTERING. These auxiliary operations each help to achieve the different requirements (i.e. smoothness, determinism, automorphism and decomposability). For example, the goal of SPLITTING a clause γ w.r.t a constrained atom a is to divide a clause into an equivalent set of clauses such that for each atom a_γ in each clause, either the atom is independent from a or it is subsumed by it. This independence leads to the decomposability of the circuit.

Example 4 (Van den Broeck [2013])**.**

$$\gamma = (\forall X, X \neq kiwi \wedge X \in Animal : flies(X) \vee \neg haswings(x))$$
$$a = (\forall X, X \neq penguin \wedge X \in Bird : flies(X))$$

SPLIT(γ, a) *results in*

$$\gamma_{mgu} = (\forall X, X \neq kiwi \wedge X \neq penguin \wedge X \in Animal \wedge X \in Bird : flies(x) \vee \neg haswings(penguins))$$
$$\gamma^1_{rest} = (flies(penguin) \vee \neg haswings(penguins))$$
$$\gamma^2_{rest} = (\forall X, X \neq kiwi \wedge X \in Animal \wedge X \notin Bird : flies(x) \vee \neg haswings(X))$$

Both γ^1_{rest} *and* γ^2_{rest} *are independent from* a.

For the other operations, we refer the reader to Van den Broeck [2013] for details. The output is, as stated above an sda-DNNF circuit with four internode operators, which allow the computation of the WFOMC.

In d-DNNF, for formulas a and b, the inter-node operators are the following:

- Decomposable conjunction: $a \bigwedge b$ with $a \perp\!\!\!\perp b$, meaning a and b are independent.
- Deterministic disjunction: $a \bigvee b$ with $a \wedge b \vDash \bot$, meaning a and b are contradictory.

The WFOMC of these operators is computed in the same way, as the WMC for propositional logic, namely

$$\text{WFOMC}(a \bigwedge b) = \text{WFOMC}(a) \times \text{WFOMC}(b)$$
$$\text{WFOMC}(a \bigvee b) = \text{WFOMC}(a) + \text{WFOMC}(b)$$

The first restriction, needed for polytime inference, is that we compute the model count over a constrained domain.

The two new intensional operators, compared to WMC, are $\forall$ and $\exists$, where

$$\text{gr}(\forall \mathbf{X}, cs : \phi) \equiv \text{gr}(\phi\theta_1) \wedge \cdots \wedge \text{gr}(\phi\theta_n) \quad (8)$$

$$\text{gr}(\exists \mathbf{X}, cs : \phi) \equiv \text{gr}(\phi\theta_1) \vee \cdots \vee \text{gr}(\phi\theta_n), \quad (9)$$

with gr$(\cdot)$ being the grounding of the sentence $\phi\theta$ and $\{\theta_1, \ldots, \theta_n\} \in$ solutions$(cs, \mathbf{X})$. The equations above show that any WFOMC problem can be reduced to WMC problem through grounding. However, the logical formula would grow exponentially in the worst case, which is why we want to reason about indistinguishable objects as one object. The decomposability and determinism from $\bigwedge$ and $\bigvee$ applies by extension to the $\forall$ and $\exists$ nodes. This leads to

$$\text{WFOMC}(\forall \mathbf{X}, cs : \phi) = \text{WFOMC}(\phi\theta)^n \quad (10)$$

$$\text{WFOMC}(\exists \mathsf{D}, cs : \phi) = \sum_n |\Theta_n| \text{WFOMC}(\phi\theta_n), \quad (11)$$

where D is a subset of the domain, $\Theta_n = \{[\mathsf{D}/d] : [\mathsf{D}/d] \in \text{solutions}(cs, \mathsf{D}) \wedge (|d| = n)\}$ and $\theta_n \in \Theta_n$. Note, that even for this restricted class of sentences the algorithm is only guaranteed to succeed for the 2 variable fragment, i.e. $|\mathbf{X}| = 2$ [Van den Broeck, 2013]. However, in Van den Broeck [2013] it has been shown to work in practice for many problems of interest, and remains an active area of research [Gehrke et al., 2018, Braun and Möller, 2018].

C EXAMPLE OF ALGORITHM 5

Example 5 (Algorithm 5)**.** *For readability, let us define $b \equiv BMI(x)$ and $a \equiv age(x)$. If we have*

(wfs_0, $bounds_0$)
$= ([2a + 3b, 4a], [\{a : [0, 45]; b : [10, 35]\}, \{a : [0, 90]\}])$

and

(wfs_1, $bounds_1$) $= ([10b, 4a], [\{b : [25, 45]\}, \{a : [18, 90]\}])$

then line 3 in Alg. 5 returns

$$\begin{aligned} &wfs = \\ &[(2a + 3b) \cdot 10b, \\ &(2a + 3b) \cdot 4a, \\ &4a \cdot 10b, \\ &4a \cdot 4a] \end{aligned}$$

and lines 6-12 return

$$\begin{aligned} &bounds = \\ &[\{a : [0, 45], b : [25, 35]\}, \\ &\{a : [18, 45], b : [10, 35]\}, \\ &\{a : [0, 90], b : [25, 45]\}, \\ &\{a : [18, 90]\}]. \end{aligned}$$

D WFOMI ALGORITHM WITH EFFICIENT INTEGRATION

The second algorithm we propose reduces the number of integrations to be solved.

Algorithm 6: Efficient Lifted WFOMI

if *node.type* = $\exists$ **then**
  result = 0;
  forall $d \in$ *range(set)* **do**
    (wfs, cst, bounds) = `WFOMI`(child, (set, d, set-d));
    result += $\binom{|\text{set}|}{d}$ cst;
else if *node.type* = $\forall$ **then**
  terms = [];
  wfs, bounds, cst = `WFOMI`(child, sets);
  uniques = find_uniques(hash(wfs));
  symbolic = solve_symbolically(uniques);
  forall *(wf, bounds, cst)* $\in$ *`WFOMI`(child, sets)* **do**
    terms.append(cst $\times$ solve_definite(wf, symbolic, bounds));
  cst = (sum(terms))$^{|\text{node.set\_type(sets)}|}$;
  wfs = [1];
  bounds = [{}];
else if *node.type* = $\wedge$ **then**
  (wfs_0, cst_0, bounds_0) = WFOMI(node.child[0], sets);
  (wfs_1, cst_1, bounds_1) = WFOMI(node.child[1], sets);
  wfs = flatten($\text{wfs}_0^\top \cdot \text{wfs}_1$);
  cst = flatten($\text{cst}_0^\top \cdot \text{cst}_1$);
  bounds = [];
  forall $bounds_i \in bounds_0$ **do**
    forall $bounds_j \in bounds_1$ **do**
      new_bounds = bounds_i ;
      forall $p \in pred(bound_i) \cap pred(bound_j)$ **do**
        new_bounds[p] = bounds_i[p] $\cap$ bounds_j[p];
      forall $p \in pred(bounds_j) \setminus pred(bounds_j)$ **do**
        new_bounds[p] = bounds_j[p];
      bounds.append(new_bounds);
else if *node.type* = $\vee$ **then**
  (wfs_0, cst_0, bounds_0) = WFOMI(node.child[0], sets);
  (wfs_1, cst_1, bounds_1) = WFOMI(node.child[1], sets);
  wfs = concat(wfs_0, wfs_1);
  bounds = concat(bounds_0, bounds_1);
  cst = concat(cst_0, cst_1);
return (wfs, cst, bounds)

The first difference to Alg. 1, mentioned previously in section 4.4, is that we split the information related to the weight functions into a three parts data structure, and carry, in addition to the weight function and the bounds, the constants separately. This is where the extra operations on lines 34

and 49 stem from.

The second difference to Alg. 1, is the check for duplicate terms in the weight function array before integration. We first hash the weight function array and check for unique values. In Python, this can be achieved with `set()`, which uses a hashing function, to find unique values and create a set from it. Then, we only need to integrate symbolically the unique values, and compute the numeric values for all terms in the weight function array, depending on the symbolic value, and its bounds. Finally, as we carry around the constants separately, the integration result is multiplied by the constant value. The result is a numeric value and is thus stored in the constant part of the data structure instead of the weight function part. This improves also the integration speed, as large constant values are multiplied in separately, instead of during integration. The speed up is, as suggested by Theorem 3, $\frac{1}{2^{(m+n)}}$, which the empirical results show as well.

E PROOFS

Proof sketch Lemma 3. Let us, first, consider the case where we have only one logical variable in the weight functions. Let $p(x)$ be a predicate, with a weight function of the form $w(p(x)) = f(a(x), b(x), c(x), \dots)$, with an arbitrary number of function symbols $a(x), b(x), ...$, and $\phi \equiv \forall x, x \in \{x_1, \dots, x_n\} : p(x)$. Let us simplify the notation, s.t. $a(x_i) \equiv a_i$ and $b(x_j) \equiv b_j$. Then,

$$
\begin{aligned}
&\text{WFOMI}(\phi) = w(p(x_1)) \times \dots \times w(p(x_n)) \\
&= \int_0^1 \dots \int_0^1 f(a_1, b_1, ...) f(a_2, b_2, ...) \dots f(a_n, b_n, ...) \\
&da_1 db_1 ... da_2 db_2 ... \dots da_n db_n ... \\
&= \int_0^1 \dots \int_0^1 f(a_1, b_1, ...) da_1 db_1 ... \\
&\int_0^1 \dots \int_0^1 f(a_2, b_2, ...) da_2 db_2 ... \\
&\int_0^1 \dots \int_0^1 f(a_n, b_n, ...) da_n db_n ... \\
&= \left(\int_0^1 \dots \int_0^1 f(a_1, b_1, ...) da_1 db_1 ... \right)^n ,
\end{aligned}
$$

where we used the separability of integrals in the second line. Then, in the case where solving the integral of $f(a(x), b(x), c(x), \dots)$ itself is in polytime, the WFOMI is still computable in polytime.

If the function symbols have two logical variables, this still holds, and the proof follows the same steps, as above.

Let us consider a predicate $p(x, y)$, with a weight function of the form $w(p(x, y)) = f(a(x, y), b(x, y), c(x, y), \dots)$, with an arbitrary number of function symbols $a(x, y), b(x, y), ...$ as described above, and $\phi \equiv \forall x, y$ s.t. $x \in \{d_1, d_2\} y \in \{d_3, \dots, d_4\} : p(x, y)$, and we use the same simplified notation as above, i.e. $a(d_i, d_j) \equiv a_{ij}$, then

$$
\begin{aligned}
&\text{WFOMI}(\phi) = \\
&w(p(d_1, d_3)) \times w(p(d_1, d_4)) \times w(p(d_2, d_3)) \times w(p(d_2, d_4)) \\
&= \int_0^1 \dots \int_0^1 f(a_{13}, b_{13}, ...) f(a_{14}, b_{14}, ...) \dots f(a_{24}, b_{24}, ...) \\
&da_{11} ... da_{24} db_{11} ... db_{24} ... \\
&= \int_0^1 \dots \int_0^1 f(a_{13}, b_{13}, ...) da_{13} db_{13} ... \ \dots \\
&\int_0^1 \dots \int_0^1 f(a_{24}, b_{24}, ...) da_{24} db_{24} ... \\
&= \left(\int_0^1 \dots \int_0^1 f(a_{13}, b_{13}, ...) da_{13} db_{13} ... \right)^{2 \times 2} ,
\end{aligned}
$$

where we used the separability of integrals in the second line. It is important to remark that $a(x_i, x_j)$ are function symbols, and that $a(x_i, x_j)$ and $a(x_i, x_k)$ need to be independent. Then, in the case where solving the integral of $f(a(x, y), b(x, y), c(x, y), \dots)$ itself is in polytime, the WFOMI is still computable in polytime. □

Proof sketch Theorem 3. Assume a theory of m literals, s.t. positive and negative literal have the same weight function, only with different boundaries. Furthermore, the theory has n atoms, with constant weight functions. Lastly, the theory has l predicates with different weight functions for both the positive and negative atoms. Then, from Definition 4, it follows there are 2^{m+l+n} integrals to solve. However, if we factorize out the constants and only integrate symbolically, there are only 2^n different integrals to solve. Thus, the number of integrals that need to be solved symbolically is reduced by a factor of $\frac{1}{2^{m+l}}$. □

Proof sketch Theorem 4. We denote by $p(l_i^+)$ the predicates that the SMT literals l_i^+ are instances of, and $i \in \{1...m\}$, for m different SMT atoms in the model. Further, we denote by $p(l_j^-)$ the predicates that the relational literals l_j^- are instances of, and $j \in \{1...n\}$, for n different relational atoms in the model. This closely follows the notation in Definition 4, with the difference that we clearly distinguish between SMT and relational atoms. Theorem 4 states that $w(p(l_i^+)) = f_i(p(l_i^+))$ and $w(p(l_i^-)) = c_j$, where c_j are

constants. Then from Definition 4 it follows

$$\begin{aligned}
&\text{WFOMI}(\triangle, w) \\
&= \sum_{M \vDash \triangle} \int_{l^+ \in M} \prod_{l_i^+ \in M} w(p(l_i^+)) \prod_{l_i^- \in M} w(p(l_j^-)) \\
&= \sum_{M \vDash \triangle} \int_{l^+ \in M} \prod_{l_i^+ \in M} f_i(p(l_i^+)) \prod_{l_i^- \in M} c_j \\
&= \sum_{M \vDash \triangle} \prod_{l_i^- \in M} c_j \prod_{l_i^+ \in M} \int_{l_i^+ \in M} f_i(p(l_i^+)) \,,
\end{aligned} \tag{12}$$

where we split up the product in the first step into two products one for the SMT atoms and one for the relational atoms. In the second step we wrote the weight functions explicitly. In the third step, we factored the constants out of the integrals and then inverted the integrals and the product for the SMT atoms, which we can do, as the weights are all functions of only one predicate, and independent of each other by construction.

On the other hand, integrating over the leaf nodes, and then using the WFOMC solver leads to the following WFOMI

$$\begin{aligned}
&\text{WFOMI}(\triangle, w) \\
&= \sum_{M \vDash \triangle} \prod_{l \in M} \int_{l^+ \in M} w(p(l)) \\
&= \sum_{M \vDash \triangle} \prod_{l_i^- \in M} \prod_{l_i^+ \in M} \int_{l^+ \in M} w(p(l_i^+)) w(p(l_j^-)) \\
&= \sum_{M \vDash \triangle} \prod_{l_i^- \in M} \prod_{l_i^+ \in M} \int_{l^+ \in M} f_i(p(l_i^+)) \cdot c_j \\
&= \sum_{M \vDash \triangle} \prod_{l_i^- \in M} c_j \prod_{l_i^+ \in M} \int_{l^+ \in M} f_i(p(l_i^+)) \,,
\end{aligned} \tag{13}$$

where the first line stems from integrating at the leaf nodes and then using the definition for WFOMC, provided in the preliminaries. This shows that integrating at the leaf nodes, under the assumptions made in Theorem 4 and using a WFOMC solver leads to the correct WFOMI. This has also been shown empirically with the Forclift WFOMC solver Van den Broeck et al. [2011]. □

Proof sketch Theorem 5. We begin by introducing $p(a_{ij})$ auxiliary predicates of SMT atoms i and relational atoms j, and we keep the notation introduced in the previous proof, with the difference that $w(p(l_j^-)) = 1$ by construction. Using auxiliary predicates, as discussed above, which have weight functions $w(p(a_{ij})) = f_j(p(l_i^+))$ leads to

$$\begin{aligned}
&\text{WFOMI}(\triangle, w) \\
&= \sum_{M \vDash \triangle} \int_{l^+ \in M} \prod_{l_i^- \in M} \prod_{l_i^+ \in M} \prod_{a_{ij} \in M} w(p(l_i^+)) w(p(l_j^-)) w(p(a_{ij})) \\
&= \sum_{M \vDash \triangle} \int_{l^+ \in M} \prod_{l_i^+ \in M} \prod_{a_{ij} \in M} w(p(l_i^+)) w(p(a_{ij})) \\
&= \sum_{M \vDash \triangle} \int_{l^+ \in M} \prod_{l_i^+ \in M} w(p(l_i^+)) \int_{l^+ \in M} \prod_{a_{ij} \in M} w(p(a_{ij})) \\
&= \sum_{M \vDash \triangle} \prod_{l_i^+ \in M} \prod_{a_{ij} \in M} \int_{l^+ \in M} w(p(l_i^+)) \int_{l^+ \in M} w(p(a_{ij})) \,,
\end{aligned} \tag{14}$$

where the first step stems from the fact that $w(p(l_j^-)) = 1$. The second step is true due to the fact that $w(p(l_i^+))$ are step distributions with the boundaries equal to the boundaries of $w(p(a_{ij}))$, and thus, evaluating

$$\int_{l^+ \in M} \prod_{a_{ij} \in M} \prod_{l_i^+ \in M} w(p(l_i^+)) w(p(a_{ij}))$$

is equivalent to

$$\int_{l^+ \in M} \prod_{l_i^+ \in M} w(p(l_i^+)) \int_{l^+ \in M} \prod_{a_{ij} \in M} w(p(a_{ij})) \,.$$

Finally, the last step in Eq. 14 holds due to the independence of the weight functions.

The fact that Eq. 14 is equal to integrating at the leaf nodes and then using a WFOMC solver follows the same reasoning as in Eq. 13 in the previous proof. □

F SDA-DNNF CIRCUITS

In this section, we provide two more complex circuits. First, in Fig. 7, we present the circuit, where we use additional auxiliary predicates. In Fig. 8 we show the circuit for the second experiment, where we introduced family relations.

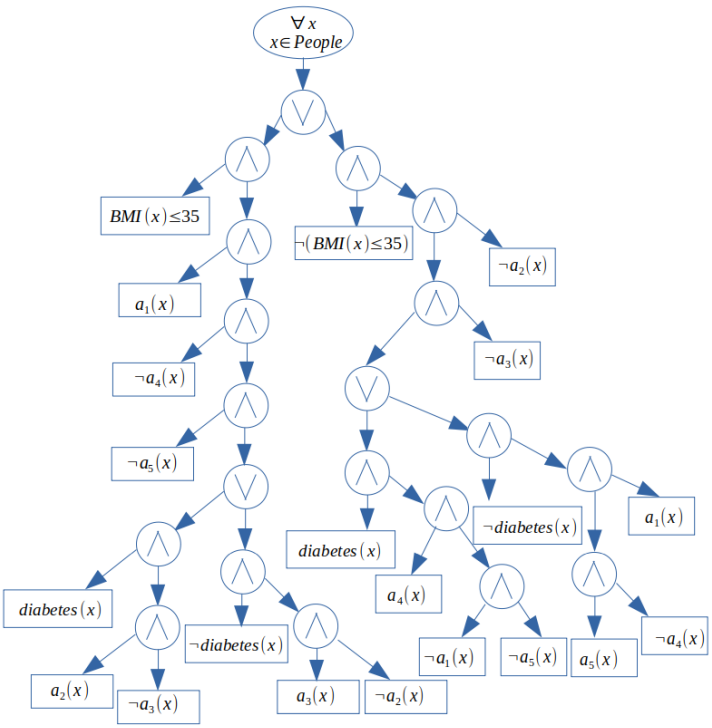

Figure 7: sda-DNNF of the motivational example with auxiliary predicates

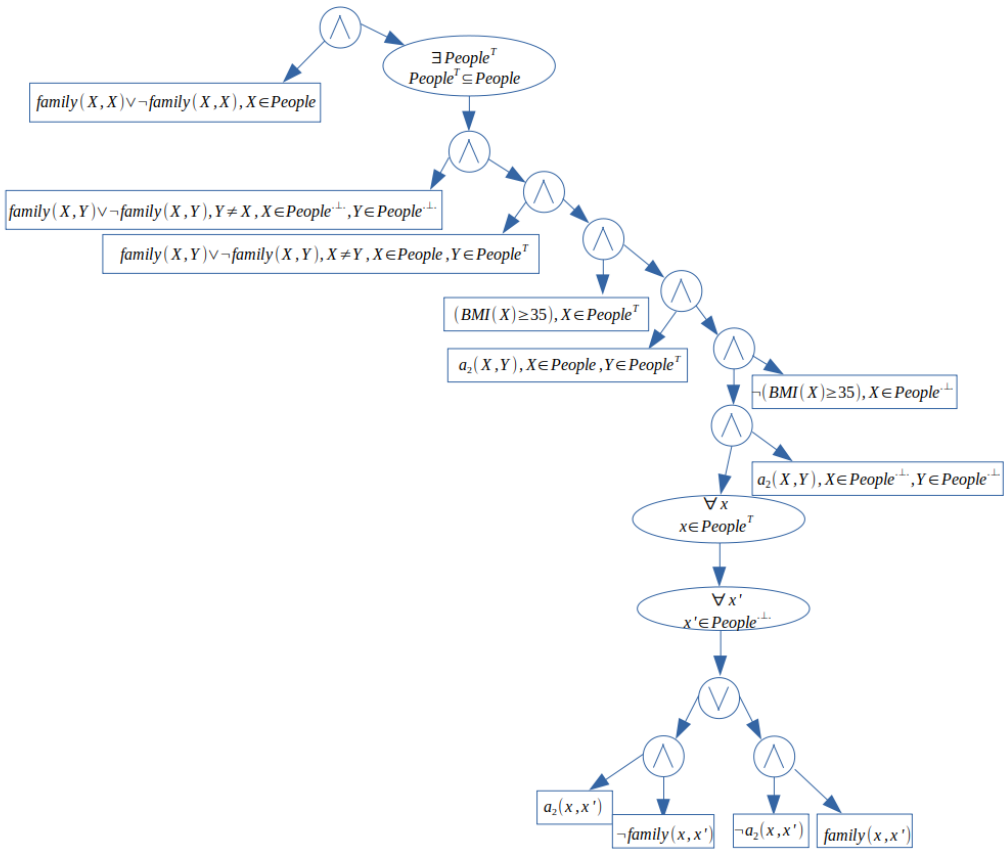

Figure 8: sda-DNNF of the example with the family predicate.

References

Vaishak Belle, Andrea Passerini, and Guy Van den Broeck. Probabilistic inference in hybrid domains by weighted model integration. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Tanya Braun and Ralf Möller. Parameterised queries and lifted query answering. In *IJCAI*, pages 4980–4986, 2018.

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted dynamic junction tree algorithm. In *International Conference on Conceptual Structures*, pages 55–69. Springer, 2018.

Guy Van den Broeck. *Lifted inference and learning in statistical relational models*. PhD thesis, Ph. D. Dissertation, KU Leuven, 2013.

Guy Van den Broeck, Nima Taghipour, Wannes Meert, Jesse Davis, and Luc De Raedt. Lifted probabilistic inference by first-order knowledge compilation. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*, pages 2178–2185, 2011.