Online appendix for the paper

Well-Definedness and Efficient Inference for Probabilistic Logic Programming under the Distribution Semantics

published in Theory and Practice of Logic Programming

FABRIZIO RIGUZZI

ENDIF – University of Ferrara
Via Saragat 1, I-44122, Ferrara, Italy
E-mail: fabrizio.riguzzi@unife.it

TERRANCE SWIFT

CENTRIA – Universidade Nova de Lisboa
E-mail: tswift@cs.suysb.edu

submitted 22 November 2010; revised 15 April 2011; accepted 14 June 2011

Appendix A Proof of Well-Definedness Theorems (Section 4.1)

To prove Theorem 1 we start with a lemma that states one half of the equivalence, and also describes an implication of the bounded term-size property for computation.

Lemma 1
Let P be a normal program with the bounded term-size property. Then

1. Any atom in $WFM(P)$ has a finite stratum, and was computed by a finite number of applications of $True^P$.
2. There are a finite number of true atoms in $WFM(P)$.

Proof
For 2), note that bounding the size of θ as used in Definition 2 bounds the size of the ground clause $B \leftarrow L_1, ..., L_n$, and so bounds the size of $True_I^P(Tr)$ for any $I, Tr \subseteq \mathcal{H}_P$. Since the true atoms in $WFM(P)$ are defined as a fixed-point of $True_I^P$ for a given I, there must be a finite number of them.

Similarly, since the size of θ is bounded by an integer L, and since $True_I^P$ is monotonic for any I $True_I^P(\emptyset)$ reaches its fixed point in a finite number of applications, and in fact only a finite number of applications of $True_I^P$ are required to compute true atoms in $WFM(P)$. In addition, it can be the case that $\mathcal{T}_I^P \neq I$ only a finite number of times, so that $WFM(P)$ can contain only a finite number of strata. □

Theorem 1
Let P be a normal program. Then $WFM(P)$ has a finite number of true atoms iff P has the bounded term-size property.

Proof
The $\Leftarrow$ implication was shown by the previous Lemma, so that it remains to prove that if $WFM(P)$ has a finite number of true atoms, then P has the bounded term-size property. To show this, since the number of true atoms in $WFM(P)$ is finite, all derivations of true atoms using $True_I^P(Tr)$ of Definition 2 can be constructed using only a finite set of ground clauses. For this to be possible, the maximum term size of any literal in any such clause is finitely bounded, so that P has the bounded term-size property. □

Theorem 2
Let T be a sound bounded term-size LPAD, and let $A \in \mathcal{H}_T$. Then A has a finite set of finite explanations that is covering.

Proof
Let T be an LPAD and w be a world of T. Each clause C_{ground} in w is associated with a choice (C, θ, i), for which C and i can both be taken as finite integers. We term (C, θ, i) the generators of C_{ground}. By Theorem 1 each world w of T has a finite number of true atoms, and a maximum size L_w of any atom in such a world. We prove that the maximum L_T of all such worlds has a finite upper bound.

We first consider the case in which T does not contain negation. Consider a world w whose well-founded model has the finite bound L_w on the size of the largest atoms. We show that L_w can not be arbitrarily large.

Since L_w is finite, all facts in T must be ground and all clauses range-restricted: otherwise some possible world of T would contain an infinite number of true atoms and so would not be bounded term-size by Theorem 1. There must be some set G of generators which acts on a chain of interpretations $I_0 \subset I_1 \subset I_n \subset WFM(w)$, where I_0 is some superset of the facts in w, and the maximum size of any atom in I_i is strictly increasing. Because $WFM(w)$ is finite and T is definite, the set of generators G must be finite.

We first show that G must contain generators (C', θ, i) and (C', θ', j) for at least one disjunctive clause C'. If not, then either 1) L_w would be infinite as there would be some recursion in which term size increases indefinitely; or 2) if there is no such recursion that indefinitely increases the size of terms and no disjunctive clauses, L_w could not be arbitrarily large and this would prove the property. In fact, without disjunction the set of clauses causing the recursion would produce an infinite model. With disjunction, eventually a different head is chosen and the recursion is stopped.

Consider then, for some set D of disjunctive clauses, the set D_{expand} of generators must be used to derive (perhaps indirectly) atoms whose size is strictly greater than the maximal size of an atom in I_n, while another set of generators D_{stop} must be used to stop the production of larger atoms, since $WFM(w)$ is finite. However, if such a situation were the case, there must also be a world w_{inf} in which for ground clauses for D whose grounding substitution is over a certain size, only the set D_{expand} of generators is chosen and D_{stop} is never chosen. The well-founded model for w_{inf} would then be infinite, against the hypothesis that T is bounded term-size.

The preceding argument has shown that since there is an overall bound on the

size of the largest atom in any world for T, T has a finite number of different models, each of which is finite. As each model is finite, there is a finite number of ground clauses that determine each model by deriving the positive atoms in the model. Each such clause is associated with an atomic choice, and the set of these clauses corresponds to a finite composite choice. The set of these composite choices corresponding to models in which the query A is true represent a finite set of finite explanations that is covering for A.

Although the preceding paragraph assumed that T did not contain negation, the assumption was made only for simplicity, so that details of strata need not be considered. The argument for normal programs is essentially the same, constituting an induction where the above argument is made for each stratum. Because Definition **??** specifies that an atom can be added to an interpretation only once, there can only be a finite number of strata in which some true atom is added, so that there will be only a finite number of strata overall. Since there are only a finite number of strata, each of which has a finite number of applications of $True_I^P(Tr)$, a finite bound L can be constructed so that T fulfills the definition of bounded term-size. □

Appendix B Proof of the Termination Theorem for Tabling (Section 6)

Theorem 3
Let P be fixed-order dynamically stratified normal program, and Q a bounded term-size query to P. Then there is an SLG evaluation of Q to P using term-depth abstraction that finitely terminates.

Proof
SLG has been proven to terminate for other notions of bounded term-size queries, so here we only sketch the termination proof.

First, we note that (Sagonas et al. 2000) guarantees that if P is a fixed-order stratified program, then there is an an SLG evaluation $\mathcal{E}$ of P that does not require the use of the SLG Delaying, Simplification or Answer Completion operations, and by implication no forest of $\mathcal{E}$ contains a conditional answer. Such an evaluation is termed *delay-minimal*. Note that Definition **??** constrains only the bindings used in $True_I^P$, and these constraints may not apply ground atoms that are undefined in the $WFM(P)$. As a result, condition answers, if they are not simplified or removed by Simplification or Answer Completion may not have a bounded term-size. This situation is avoided by delay-minimal evaluations. Next, we assume that all negative selected literals are ground. This assumption causes no loss of generality as the evaluation will flounder and so terminate finitely if a non-ground negative literal is selected. Given this context, the proof uses the forest-of-trees model (Swift 1999) of SLG (Chen and Warren 1996).

- We consider as an induction basis the case when Q is in stratum 0 – that is, when Q can be derived without clauses that contain negative literals, or is part of an unfounded set $\mathcal{S}$ of atoms and clauses for atoms in $\mathcal{S}$ do not contain negative literals. As argued in Section 6, the use of term-depth abstraction

ensures that an SLG evaluation $\mathcal{E}$ of a query Q to a program with bounded term-size has only a finite number of trees. In addition, since SLG works on the original clauses of a program P and P is finite, (although $ground(P)$ may not be), there can be only a finite number of clauses resolvable against the root of any tree via PROGRAM CLAUSE RESOLUTION, and so the root of each SLG tree can contain only a finite number of children. Finally, to show that each interior node has a finite number of children, we consider that there can only be a finite number of answers to any subgoal upon which Q depends. This follows from the fact that $\mathcal{E}$ is delay-minimal and so produces no conditional answers, together with the the bound of Definition 4 that ensures a program is bounded term-size. As a result, there are only a finite number of nodes that are produced through ANSWER RETURN. These observations together ensure that each tree in any SLG forest of $\mathcal{E}$ is finite. Since each operation (including the SLG COMPLETION operation, which does not add nodes to a forest) is applicable only one time to a given node or set of nodes in an evaluation (i.e. executing an SLG operation removes the conditions for its applicability) the evaluation $\mathcal{E}$ itself must be finite and statement holds for the induction basis.

- For the induction step, we assume the statement holds for queries whose (fixed-order) dynamic strata is less than N to show that the statement will hold for a query Q at stratum N as well. As indicated above, we use a delay-minimal SLG evaluation $\mathcal{E}$ that does not require DELAYING, SIMPLIFICATION or ANSWER COMPLETION operations. For the induction case, the various SLG operations that do not include negation will only produce a finite number of trees and a finite number of nodes in each tree as described in the induction basis. However if there is a node N in a forest with a selected negative literal $\neg A$, the SLG operation NEGATION RETURN is applicable. In this case, a single child will be produced for N and no further operations will be applicable to N. Thus any forest in $\mathcal{E}$ will have a finite number of finite trees, and since all operations can be applied once to each node, as before $\mathcal{E}$ will be finite, so that the statement holds by induction.

□

Appendix C Proof of the Correctness Theorems for PITA (Section 8)

The next theorem addresses the correctness of the PITA evaluation. As discussed in Section 8, the BDDs of the PITA transformation are represented as ground terms, while BDD operations, such as *and/3*, *or/3* etc. are infinite relations on such terms. The PITA transformation also uses the predicate *get_var_n/4* whose definition in Section 7 is:

$$\begin{array}{l} get\_var\_n(R, S, Probs, Var) \leftarrow \\ \quad (var(R, S, Var) \rightarrow true; \\ \quad length(Probs, L), add\_var(L, Probs, Var), assert(var(R, S, Var))). \end{array}$$

This definition uses a non-logical update of the program, and so without modifications, it is not suitable for our proofs below. Alternately, we assume that $ground(T)$

is augmented with a (potentially infinite) number of facts of the form $var(R, [], Var)$ for each ground rule R (note that no variable instantiation is needed in the second argument of $var/3$ if it is indexed on ground rule names). Clearly, the augmentation of T by such facts has the same meaning as *get_var_n/4*, but is simply done by an a priori program extension rather than during the computation as in the implementation.

Lemma 1
Let T be an LPAD and Q a bounded term-size query to T. Then the query $PITA_H(Q)$ to $PITA(T)$ has bounded term-size.

Proof
Although T_Q (Definition 6) has bounded term-size, we also need to ensure that $PITA(T_Q)$ has bounded term-size, given the addition of the BDD relations *and/3*, *or/3*, etc. along with the *var/3* relations mentioned above.

Both *var/3* and the BDD relations are functional on their input arguments (i.e. the first two arguments of *var/3*, *and/3*, *or/3*. etc. (cf. Section 7). Therefore, for the body of a clause C that was true in an application of $True_I^{T_Q}$ there are exactly n bodies that are true in an application of $True_I^{PITA(T_Q)}$, where n is the number of heads of C. Thus the size of every ground substitutions in every iteration of $True_I^{PITA(T_Q)}$ is bounded as well.

Note that since $PITA(T)$ and $PITA_H(Q)$ are both syntactic transformations, the theorem applies even if the LPAD isn't sound. □

Theorem 4
Let T be a fixed-order dynamically stratified LPAD and Q a ground bounded term-size atomic query. Then there is an SLG evaluation $\mathcal{E}$ of $PITA_H(Q)$ against $PITA(T_Q)$, such that answer subsumption is declared on $PITA_H(Q)$ using BDD-disjunction where $\mathcal{E}$ finitely terminates with an answer Ans for $PITA_H(Q)$ and $BDD(Ans)$ represents a covering set of explanations for Q.

Proof
(Sketch) The proof uses the forest-of-trees model (Swift 1999) of SLG (Chen and Warren 1996).

Because T is fixed-order dynamically stratified, queries to T can be evaluated using SLG without the DELAYING, SIMPLIFICATION or ANSWER COMPLETION operations. Instead, as (Sagonas et al. 2000) shows, only the SLG operations NEW SUBGOAL, PROGRAM CLAUSE RESOLUTION, ANSWER RETURN and NEGATIVE RETURN are needed. Since T is fixed-order dynamically stratified, it is immediate from inspecting the transformations of Section **??** together with the fact that the BDD relations are functional that $PITA(T)$ is also fixed-order dynamically stratified as is $PITA(T)_Q$.

However, Theorem 3 must be extended to evaluations that include answer subsumption, which we capture with a new operation ANSWER JOIN to perform answer subsumption over an upper semi-lattice L. Without loss of generality we assume that a given predicate of arity $m > 0$ has had answer subsumption declared on its m^{th} argument and we term the first $m - 1$ arguments *non-subsuming arguments*.

We recall that a node N is an answer in an SLG tree T if N has no unresolved goals and is a leaf in T. Accordingly, creating a child of N with a special marker $fail$ is a method to effectively delete an answer (cf. (Swift 1999)).

- ANSWER JOIN: Let an SLG forest $\mathcal{F}_n$ contain an answer node

$$N = Ans \leftarrow$$

where the predicate for Ans has been declared to use answer subsumption over a lattice L for which the join operation is decidable, and let the arity of Ans be $m > 0$. Further, let $\mathcal{A}$ be the set of all answers in $\mathcal{F}_n$ that are in the same tree, T_N, as N and for which the non-subsuming arguments are the same as Ans. Let $Join$ be the L-join of all the final arguments of all answers in $\mathcal{A}$.

— If $(Ans \leftarrow)\{arg(m, Ans)/Join\}$ is not an answer in T_N, add it as a child of N, and add the child $fail$ to all other answers in $\mathcal{A}$.
— Otherwise, if $(Ans \leftarrow)\{arg(m, Ans)/Join\}$ is answer in T_N, create a child $fail$ for N.

For the proof, the first item to note is that since T_Q is bounded term-size, any clauses on which Q depends that give rise to true atoms in the well-founded model of any world of T must be be range-restricted – otherwise since T has function symbols, T_Q would have an infinite model and not be bounded term-size. Given this, it is then straightforward to show that $PITA(T)_Q$ is also range-restricted and that any answer A of $PITA_H(Q)$ will be ground (cf. (Muggleton 2000)). Accordingly, the operation ANSWER JOIN will be applicable to any subgoal with a non-empty set of answers.

We extend Theorem 3 and Lemma 1 to show that since $PITA(T)_Q$ has the bounded term-size property, a SLG evaluation of a query $PITA_H(Q)$ to $PITA(T)_Q$ will terminate. Because the join operation for L is decidable, computation of the join will not affect termination properties. Let T_N be a tree whose root subgoal is a predicate that uses answer subsumption. Then each time a new answer node N is added to T_N there will be one new ANSWER JOIN operation that becomes applicable for N. Let $\mathcal{A}$ be a set of answers in T_N as in the definition of ANSWER JOIN. Then applying the ANSWER JOIN operation will either 1) create a child of N that is a new answer and "delete" $|\mathcal{A}|$ answers by creating children for them of the form $fail$; or 2) "delete" the answer N by creating a child $fail$ of N. Clearly any answer can be deleted at most once, and each application of the ANSWER JOIN operation will delete at least one answer in T_N. Accordingly, if T_N contains Num answers, there can be at most Num applications of ANSWER JOIN for answers in T_N. Using these considerations it is straightforward to show that termination of bounded term-size programs holds for SLG evaluations extended with answer subsumption [1].

[1] As an aside, note that due to the fact that ANSWER JOIN deletes all answers in $\mathcal{A}$ except the join, it can be shown by induction that immediately after an ANSWER JOIN operation is applied to Ans in a tree T_N, there will be only one "non-deleted" answer in T_N with the same

Thus, the bounded term-size property of $PITA(T)_Q$ together with Theorem 2 imply that there will be a finite set of finite explanations for $PITA_H(Q)$, and the preceding argument shows that SLG extended with ANSWER JOIN will terminate on the query $PITA_H(Q)$. It remains to show that an answer Ans for $PITA_H(Q)$ in the final state of $\mathcal{E}$ is such that $BDD(Ans)$ represents a covering set of explanations for Q. That $BDD(Ans)$ contains a covering set of explanations can be shown by induction on the number of BDD operations. For the induction basis it is easy to see that the operations *zero/1* and *one/1* are covering for false and true atoms respectively.

- Consider an "and" operation in the body of a clause. For the inductive assumption, BB_{i-1} and B_i both represent finite set of explanations covering for $L_1, \ldots, L_{i-1}$ and L_i respectively. Let F_{i-1}, F'_i, and F_i be the formulas expressed by BB_{i-1}, B_i, and BB_i respectively. These formulas can be represented in disjunctive normal form, in which every disjunct represents an explanation. F_i is obtained by multiplying F_{i-1} and F'_i so, by algebraic manipulation, we can obtain a formula in disjunctive normal form in which every disjunct is the conjunction of two disjuncts, one from F_{i-1} and one from F'_i. Every disjunct is thus an explanation for the body prefix up to and including L_i. Moreover, every disjunct for F_i is obtained by conjoining a disjunct for F_{i-1} with a disjunct for F'_i.
- In the case of a "not" operation in the body of a clause, let L_i be the negative literal $\neg D$. Then for BN_i the BDD produced by D, $not(BN_i, B_i)$ simply negates this BDD to produce a covering set of explanations for $\neg D$.
- In the case of an "or" operation between two answers, the resulting BDD will represents the union of the set of explanations represented by the BDDs that are joined.

Since the property holds both for the induction basis and the induction step, the set of explanations represented by $BDD(Ans)$ is covering for the query. □

References

CHEN, W. AND WARREN, D. S. 1996. Tabled evaluation with delaying for general logic programs. *Journal of the Association for Computing Machinery 43,* 1, 20–74.

MUGGLETON, S. 2000. Learning stochastic logic programs. *Electronic Transactions on Artificial Intelligence 4,* B, 141–153.

SAGONAS, K., SWIFT, T., AND WARREN, D. S. 2000. The limits of fixed-order computation. *Theoretical Computer Science 254,* 1-2, 465–499.

SWIFT, T. 1999. A new formulation of tabled resolution with delay. In *Recent Advances in Artifiial Intelligence*. LNAI, vol. 1695. Springer, 163–177.

non-subsuming bindings as *Ans*. Accordingly, if the cost of computing the join is constant, the total cost of *Num* ANSWER JOIN operations will be *Num*. Based on this observation, the implementation of PITA can be thought of as applying an ANSWER JOIN operation immediately after a new answer is derived in order to avoid returning answers that are not optimal given the current state of the computation.