Level Mapping Induced Loop Formulas for Weight Constraint and Aggregate Programs

Guohua Liu

Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada
guohua@cs.ualberta.ca

Abstract. We improve the formulations of loop formulas for weight constraint and aggregate programs by investigating the level mapping characterization of the semantics for these programs. First, we formulate a level mapping characterization of the stable model semantics for weight constraint programs, based on which we define loop formulas for these programs. This approach makes it possible to build loop formulas for programs with arbitrary weight constraints without introducing new atoms. Secondly, we further use level mapping to characterize the semantics and propose loop formulas for aggregate programs. The main result is that for aggregate programs not involving the inequality comparison operator, the dependency graphs can be built in polynomial time. This compares to the previously known exponential time method.

1 Introduction

Logic programming under *stable model* semantics has been extended to incorporate a variety of constraints to facilitate knowledge representation and reasoning. These constraints include weight constraints [12], aggregates [1, 11, 14] and abstract constraints [10, 13]. We refer to logic programs with these constraints as weight constraint, aggregate and abstract constraint programs, respectively.

Lin and Zhao [6] propose to compute stable models of normal logic program as the model of the loop completion of the program. The loop completion consists of the loop formulas and the formulas of the completion of the program. Liu and Truszczyński [8] extend the approach to weight constraint programs where the weight constraints contain only positive literals and weights. However, in order to transform an arbitrary weight constraint to a weight constraint with only positive literals and weights, new propositional atoms are needed [8, 9]. In theory, new atoms enlarge the search space for stable model computation. An interesting question is whether the loop formulas for arbitrary weight constraint programs can be formulated without extra atoms.

The method of level mapping has been studied to characterize stable models [2, 4] of normal programs. We observe that such a characterization is closely related to the formulation of loop formulas. We present level mapping characterization of the stable models of weight constraint programs. The characterization leads to a formulation of loop formulas for arbitrary weight constraint programs without introducing extra atoms.

Aggregate programs are closely related to weight constraint programs, since many aggregates can be encoded by weight constraints [7]. There are different semantics proposed for aggregate programs [3, 5, 14]. Among them, the semantics based on conditional satisfaction is considered the most *conservative* [13], in the sense that any answer set under this semantics is an answer set under others, but the reverse may not hold. We are interested in the formulation of loop formulas for this semantics. To distinguish from the stable model semantics of weight constraint programs, we call the semantics *answer set* semantics.

Loop formulas for answer set semantics are presented in [15]. In the approach, given a program, the construction of the dependency graph requires computing what is called "local power set" for the constraints in the program, to capture conditional satisfaction. The process takes exponential time in the size of the program. We investigate the level mapping characterization of answer sets and find that, for aggregates, the conditional satisfaction checking can be reduced to polynomial time standard satisfaction checking. Based on this finding, we define the levels of aggregates. The definition induces a formulation of loop formulas, where local power sets are not needed and the exponential process to construct the dependency graph is avoided.

2 Level mapping induced loop formulas for weight constraint programs

A *weight constraint* is of the form

$$l\,[a_1{=}w_{a_1}, ..., a_n{=}w_{a_n}, \texttt{not}\ b_1{=}w_{b_1}, ..., \texttt{not}\ b_m{=}w_{b_m}]\,u \tag{1}$$

where each a_i, b_j is an atom. Atoms $a_i's$ and not-atoms $\texttt{not}\ b_i's$ are also called *literals* (*positive* and *negative* literals, respectively). We denote by $lit(W)$ the set of literals in a weight constraint. Each literal in a constraint is associated with a *weight* [1]. The numbers l and u give the lower and upper bounds of the constraint, respectively. The weights and bounds are real numbers. Either of the bounds may be omitted in which case the missing lower bound is taken to be $-\infty$ and the missing upper bound ∞.

A set of atoms M satisfies a weight constraint W of the form (1), denoted $M \models W$, if (and only if) $l \leq w(W, M) \leq u$, where $w(W, M) = \sum_{a_i \in M} w_{a_i} + \sum_{b_i \notin M} w_{b_i}$. M satisfies a set of weight constraints Π if $M \models W$ for every $W \in \Pi$.

A *weight constraint program* is a finite set of rules of the form

$$W_0 \leftarrow W_1, ..., W_n \tag{2}$$

where each W_i is a weight constraint. We use $hd(r)$ and $bd(r)$ to denote W_0 and $\{W_1, ..., W_n\}$, respectively. $Atom(P)$ denotes the set of the atoms appearing in program P. For the semantics of weight constraint programs, we refer the reader to [12].

Notations: In the rest of this paper, we will use the following notations: Given a weight constraint W of the form (1) and a set of atoms M, we define $M_a(W) = \{a_i \in$

[1] The weights of literals could be negative. It is pointed out that negative weights can be eliminated by a transformation [12]. We assume that weights are non-negative if not indicated otherwise.

$M \mid a_i \in lit(W)\}$ and $M_b(W) = \{b_i \in M \mid \texttt{not}\ b_i \in lit(W)\}$. Since W is always clear by context, we will simply write M_a and M_b.

In general, an atom may appear both positively and negatively in a weight constraint. We call such an atom a *dual* atom, e.g. atom a is a dual atom in $1[a = 1, \texttt{not}\ a = 2]1$.

Following the notation in [13], for a set of atoms X and a mapping λ from X to positive integers, we define $H(X) = max(\{\lambda(a) \mid a \in X\})$. For the empty set $\emptyset$, we define $\max(\emptyset) = 0$ and $\min(\emptyset) = \infty$.

2.1 Level mapping characterization of stable models

Given a set of atoms X, a *level mapping* of X is a function λ from the atoms in X to positive integers. Let W be a weight constraint of the form (1), M a set of atoms and λ a level mapping of M. The *level* of W w.r.t. M, denoted $L(W, M)$, is defined as:

$$L(W, M) = min(\{H(X_a) \mid X \subseteq M,\ and\ w(W, X_a) \geq l + \sum_{b_i \in M \setminus X_a} w_{b_i}\}). \quad (3)$$

Proposition 1. *Let W be a weight constraint of the form (1), M and X be two sets of atoms. $w(W^M, X_a) \geq l^M$ iff $w(W, X_a) \geq l + \sum_{b_i \in M \setminus X_a} w_{b_i}$, where W^M is the* reduct *of W w.r.t. M as defined in [12] and l^M is the lower bound of W^M.*

Intuitively, the level of W w.r.t. M depends on the levels of atoms in M that are necessary to satisfy W^M and positive in W.

Definition 1. *Let P be a weight constraint program and M a set of atoms. M is said to be* level mapping justified *by P if there is a level mapping λ of M satisfying that for each $b \in M$, there is a rule $r \in P$ such that $b \in lit(hd(r))$, $M \models bd(r)$, and for each $W \in bd(r)$, $\lambda(b) > L(W, M)$.*

By Proposition 1, the following theorem can be proved.

Theorem 1. *Let P be a weight constraint program and M a set of atoms. M is a stable model of P iff M is a model of P and level mapping justified by P.*

2.2 Loop formulas for weight constraint programs

To characterize stable models by loop formulas, we need the concept of *completion* of a program, whose models are the supported models of the program. The definition of completion can be found in [8]. For a program P, we denote its completion $Comp(P)$.

The formulation of loop formulas consists of two steps: constructing a dependency graph and then establishing a formula for each loop in the graph.

Let P be a weight constraint program. The *dependency graph* of P, denoted $G_P = (V, E)$, is a directed graph, where (i). $V = Atoms(P)$ and (ii). (u, v) is a directed edge from u to v in E, if there is a rule of the form (2) in P, such that $u \in lit(W_0)$ and $v \in lit(W_i)$ for some i ($1 \leq i \leq n$). Let $G = (V, E)$ be a directed graph. A set $L \subseteq V$ is a *loop* in G if the subgraph of G induced by L is strongly connected.

Let W be a weight constraint and L be a set of atoms. The *restriction* of W w.r.t. L, denoted $W_{|L}$, is a conjunction of weight constraints $W_{l|L} \wedge W_{u|L}$, where

- $W_{l|L}$ is obtained by removing the upper bound and all positive literals in L and their weights from W;
- $W_{u|L}$ is obtained by removing the lower bound from W.

Let P be a weight constraint program and L be a loop in G_P. The *loop formula* for L, denoted $LF(P, L)$, is defined as

$$LF(P, L) = \bigvee L \rightarrow \bigvee \{ \bigwedge_{W \in bd(r)} W_{|L} \mid r \in P, L \cap lit(hd(r)) \neq \emptyset \} \tag{4}$$

Let P be a weight constraint program. The *loop completion* of P denoted $LComp(P)$ is defined as $LComp(P) = Comp(P) \cup \{LF(P, L) \mid L \; is \; a \; loop \; in \; G_P\}$.

Theorem 2. *Let P be a weight constraint program and M a set of atoms. M is a stable model of P iff M is a model of $LComp(P)$.*

3 Level mapping induced loop formulas for aggregate programs

An aggregate is a constraint on sets taking the form $aggr(\{X \mid p(X)\})$ `op` $Result$, where $aggr$ is an *aggregate function*. The standard aggregate functions are those in {SUM, COUNT, AVG, MAX, MIN}. The relational operator op is from $\{=, \neq, <, > , \leq, \geq\}$ and $Result$ is either a variable or a numeric constant.

An aggregate program is a set of rules of the form

$$h \leftarrow A_1, ..., A_n \tag{5}$$

where h is an atom and $A_1, ..., A_n$ are aggregates. The semantics of aggregate programs bases on the notion of *conditional satisfaction*. We refer the reader to [14] for the details.

All of the standard aggregates (without the operator "$\neq$"[2]) can be encoded by weight constraints as shown in [7]. In this section, we focus on programs with aggregate SUM only [3]. For an aggregate A, we denote its weight constraint encoding $W(A)$. A property of aggregates is that their weight constraint encoding contain no dual atoms. This is useful to prove the proposition later.

3.1 Level mapping characterization of answer sets

Let A be an aggregate, M a set of atoms and λ a level mapping of M. The *answer set level* of A w.r.t. M, denoted $L^*(A, M)$, is defined as:

$$L^*(A, M) = min(\{H(X) \mid X \subseteq M, w(W(A), X_a) \geq l + \sum_{b_i \in M} w_{b_i}, \; and \; w(W(A), X_b) \leq u - \sum_{a_i \in M} w_{a_i}\}), \tag{6}$$

[2] We only study the aggregates without the operator "$\neq$", since for programs with $SUM(.) \neq k$, the *answer set existence problem* is at a level higher than NP-Completeness in the complexity hierarchy, according to Son and Pontelli [14].

[3] Note: Aggregates $COUNT$ and AVG are special cases of SUM. Aggregates MAX and MIN can be encoded by SUM [7].

where l and u are the lower and upper bounds of $W(A)$ respectively.

Intuitively, the level of A, w.r.t. M depends on the level of atoms in M that are necessary to conditionally satisfies A, w.r.t. M.

Proposition 2. *Let A be an aggregate and X and M two sets of atoms such that $X \subseteq M$. X conditionally satisfies A w.r.t. M iff $w(W(A), X_a) \geq l + \sum_{b_i \in M} w_{b_i}$ and $w(W(A), X_b) \leq u - \sum_{a_i \in M} w_{a_i}$, where l and u are the lower and upper bounds of $W(A)$, respectively.*

Definition 2. *Let P be an aggregate program and M a set of atoms. M is said to be* strongly level mapping justified *by P if there is a level mapping λ of M satisfying that for each $b \in M$, there is a rule $r \in P$ such that $b = hd(r)$, $M \models bd(r)$, and for each $A \in bd(r)$, $\lambda(b) > L^*(A, M)$.*

Using Proposition 2, we can prove the following theorem.

Theorem 3. *Let P be an aggregate program and M a set of atoms. M is an answer set of P iff M is a model of P and strongly level mapping justified by P.*

3.2 Loop formulas for aggregate programs

The completion of aggregate programs consists of the same set of formulas as that of weight constraint programs, except that the weight constraints in the formulas are weight constraint encoding of aggregates.

Let P be an aggregate program. The *dependency graph* of P, denoted $G^*_P = (V, E)$, is a directed graph, where (i). $V = Atom(P)$ and (ii). (u, v) is a directed edge from u to v in E, if there is a rule of the form (5) in P such that $u = hd(r)$ and either v or `not` $v \in lit(W(A_i))$ for some i ($1 \leq i \leq n$).

Now we give the *strong restriction* of an aggregate w.r.t. a loop by defining the strong restriction of a weight constraint. Let W be a weight constraint and L a set of atoms. The strong restriction of W, w.r.t. L, denoted $W^*_{|L}$, is a conjunction of weight constraints $W^*_{l|L} \wedge W^*_{u|L}$, where

- $W^*_{l|L}$ is obtained by removing from W the upper bound, all positive literals that are in L and their weights;
- $W^*_{u|L}$ is obtained by removing from W the lower bound, `not` $b_i = w_{b_i}$ for each $b_i \in L$, and changing the upper bound to be $u - \sum_{b_i \in L} w_{b_i}$.

The strong restriction of an aggregate A w.r.t. a loop L is defined as the strong restriction of its weight constraint encoding w.r.t. the loop $W^*_{|L}(A)$.

Let P be an aggregate program and L a loop in G^*_P. The loop formula for L, denoted $LF^*(P, L)$, is defined as

$$LF^*(P, L) = \bigvee L \rightarrow \bigvee \{ \bigwedge_{A \in bd(r)} W(A)^*_{|L} \mid r \in P, hd(r) \in L \} \tag{7}$$

Let P be an aggregate program. The loop completion of P, denoted $LComp^*(P)$, is defined as $LComp^*(P) = Comp(P) \cup \{LF^*(P, L) \mid L \; is \; a \; loop \; in \; G^*_P\}$.

Theorem 4. *Let P be an aggregate program and M a set of atoms. M is an answer set of P iff it is a model of $LComp^*(P)$.*

4 Conclusion and future work

We present level mapping characterizations of semantics for weight constraint programs and aggregate programs, respectively. Based on the level mapping characterizations, we improve the formulation of loop formulas for these programs. For arbitrary weight constraint programs, we propose a formulation of loop formulas that does not require introducing any new atoms; For aggregate programs, we show that the dependency graph can be constructed in time polynomial in the size of programs.

The level mapping and loop formulas are defined on aggregate SUM. For the aggregates MAX and MIN, direct definition may be desired for intuitiveness. We leave this for the future work.

References

1. T. Dell'Armi, W. Faber, G. Lelpa, and N. Leone. Aggregate functions in disjunctive logic programming: semantics, complexity, and implementation in DLV. In *Proc. IJCAI'03*, pages 847–852, 2003.
2. E. Erdem and V. Lifschitz. Tight logic programs. *Theory and Practice of Logic Programming*, 3(4):499–518, 2003.
3. W. Faber, N. Leone, and G. Pfeifer. Recursive aggregates in disjunctive logic programs. In *Proc. JELIA'04*, pages 200–212, 2004.
4. F. Fages. Consistency of clark's completion and existence of stable models. *Journal of Methods of Logic in Computer Science,1*, pages 51–60, 1994.
5. P. Ferraris. Answer sets for propositional theories. In *Proc. LPNMR'05*, pages 119–131, 2005.
6. F. Lin and Y. Zhao. ASSAT: Computing answer sets of a logic program by SAT solvers. *Artificial Intelligence*, 157(1-2):115–137, 2004.
7. G. Liu and J. You. Lparse programs revisited: semantics and representation of aggregates. In *Proc. ICLP'08*, pages 347–361, 2008.
8. L. Liu and M. Truszczyński. Properties and applications of programs with monotone and convex constraints. *Journal of Artificial Intelligence Research*, 7:299–334, 2006.
9. V. Marek, I. Niemelä, and M. Truszczyński. Logic programs with monotone abstract constraint atoms. *Theory and Practice of Logic Programming*, 8(2):167–199, 2008.
10. V. W. Marek and J. B. Remmel. Set constraints in logic programming. In *Proc. LPNMR'04*, pages 167–179, 2004.
11. N. Pelov, M. Denecker, and M. Bruynooghe. Well-founded and stable semantics of logic programs with aggregates. *Theory and Practice of Logic Programming*, 7:301–353, 2007.
12. P. Simons, I. Niemelä, and T. Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1-2):181–234, 2002.
13. T. C. Son, E. Pontelli, and P. H. Tu. Answer sets for logic programs with arbitrary abstract constraint atoms. *Journal of Artificial Intelligence Research*, 29:353–389, 2007.
14. T.C. Son and E. Pontelli. A constructive semantic characterization of aggregates in answer set programming. *Theory and Practice of Logic Programming*, 7:355–375, 2006.
15. J. You and G. Liu. Loop formulas for logic programs with arbitrary constraint atoms. In *Proc. AAAI-08*, pages 584–5895, 2008.