Online appendix for the paper

Complexity and Compilation of GZ-Aggregates in Answer Set Programming

published in Theory and Practice of Logic Programming

MARIO ALVIANO and NICOLA LEONE
Department of Mathematics and Computer Science, University of Calabria, Italy

submitted 30 June 2015; revised 04 July 2015; accepted 14 July 2015

Appendix A Proofs of Section 3

Lemma 1

Let Π be in ASP(M). The least fixpoint of $T_\Pi(I)$ exists and is polytime computable. Let I be the least fixpoint of T_Π, and J be the least fixpoint of $T_{G(\Pi,I)}$. If $I \neq J$ then Π is G-incoherent, otherwise $GSM(\Pi) = \{I\}$.

Proof

We first show that the least fixpoint of T_Π is polytime computable. Let Π be a program in ASP(M), and I be an interpretation. Computing $T_\Pi(I)$ requires to iterate over every rule r of Π and check whether $I \models B(r)$. Checking $I \models B(r)$ can be done in polynomial-time if aggregates are polynomial-time computable functions, as it is assumed in this section. Hence, a single application of T_Π is polynomial-time computable. The least fixpoint of T_Π is computed, by definition, starting from $\emptyset$ and repeatedly applying T_Π. Define $I_0 = \emptyset$, $I_{i+1} = T_\Pi(I_i)$ (for $i \geq 0$). For each $i \geq 0$, either $I_{i+1} \setminus I_i \neq \emptyset$ or I_i is the least fixpoint of T_Π. Since atoms in $I_{i+1} \setminus I_i$ are among those in $At(\Pi)$, we have that $I_{|At(\Pi)|} = I_{|At(\Pi)|+1}$.

We now show the second part of the lemma. $I \models \Pi$ by construction. Note that $G(\Pi, I)$ is a plain Datalog program. It is unique minimal model is the least fixpoint of $T_{G(\Pi,I)}$, i.e., interpretation J. Hence, $I \in GSM(\Pi)$ if and only if $I = J$. To complete the proof is enough to show that no other interpretation is a G-stable model of Π. Let K be an interpretation such that $K \neq I$ and $K \models \Pi$. Therefore, $K \supset I$ because I is the least fixpoint of T_Π. To prove that $K \notin GSM(\Pi)$ note that $I \models G(\Pi, K)$. $\square$

Theorem 1

G-coherence testing is in P for ASP(M).

Proof
Let I be the least fixpoint of T_Π. I is computable in polynomial-time because of Lemma 1. Actually, I is the only candidate to be a G-stable model of Π because of Lemma 1. To check whether $I \in GSM(\Pi)$, build $G(\Pi, I)$ and compute the least fixpoint of $T_{G(\Pi,I)}$, again in polynomial-time because of Lemma 1. If the two least fixpoints are equal then Π is G-coherent, otherwise it is G-incoherent. □

Theorem 2
G-coherence testing is in NP for programs in ASP(¬, M, C, N).

Proof
Let Π be in ASP(¬, M, C, N), and I be an interpretation. We provide a polynomial-time procedure for checking that I is a G-stable model of Π. The procedure first checks that $I \models \Pi$ in polynomial-time. If it is the case, the procedure builds the reduct $G(\Pi, I)$, again in polynomial-time. Program $G(\Pi, I)$ is in ASP($-$) and therefore Lemma 1 can be applied to obtain the unique minimal model of $G(\Pi, I)$, say J, in polynomial-time. If $I = J$ then the procedure accepts I as a G-stable model, otherwise it rejects I. □

Lemma 2
Let Π be in ASP(¬, ∨). Then, $GSM(\Pi) \equiv_{At(\Pi)} GSM(C(\Pi)) \equiv_{At(\Pi)} GSM(N(\Pi))$.

Proof
Let I be an interpretation. $I \models \Pi$ if and only if $I \models C(\Pi)$. In particular, if $\sim p$ is replaced by an aggregate A in a rule r, we have $I \models \sim p$ if and only if $I \models A$. Note that $I \not\models \sim p$ implies that r is removed in the reducts $G(\Pi, I), G(C(\Pi), I)$, while $I \models \sim p$ implies that both $\sim p$ and A are replaced by the empty set in the rules obtained from r in the reducts. We therefore conclude that $G(\Pi, I) = G(\Pi, C(I))$, from which we obtain $GSM(\Pi) \equiv_{At(\Pi)} GSM(C(\Pi))$.

The proof of $GSM(\Pi) \equiv_{At(\Pi)} GSM(N(\Pi))$ is similar. We have just to additionally note that $\bot \notin I$ holds for every $I \in GSM(\Pi) \cup GSM(N(\Pi))$. □

Theorem 3
G-coherence testing is Σ_2^P-hard for both ASP(∨, C) and ASP(∨, N). It is NP-hard for both ASP(C) and ASP(N).

Proof
G-coherence testing is Σ_2^P-hard for ASP(¬, ∨), and it is NP-hard for ASP(¬) (Eiter and Gottlob 1995). G-coherence of Π can be reduced to G-coherence testing of $C(\Pi)$ or of $N(\Pi)$ because of Lemma 2. Since $C(\Pi)$ and $N(\Pi)$ can be computed in polynomial-time, do not introduce disjunction, eliminate negation, and only have convex and non-convex aggregates, respectively, the proof is complete. □

Theorem 4
G-coherence testing is P-hard for ASP(M).

Proof
G-cautious reasoning over Datalog programs is P-hard (Eiter and Gottlob 1995). We reduce this problem to G-coherence testing of disjunction- and negation-free programs with monotone aggregates. Let Π be in ASP$(-)$, and p be a propositional atom. Program $\Pi' = \Pi \cup \{p \leftarrow A\}$, where $dom(A) = \{p\}$ and $A(I) = |\{p\} \cap I| \geq 0$, can be built using only logarithmic space. Since Π is a Datalog program, it has a unique G-stable model, say I. If $p \in I$ then p belongs to the least fixpoint of T_Π because of Lemma 1, and therefore it belongs to the least fixpoint of $T_{\Pi'}$ too because of monotonicity. On the other hand, if $p \notin I$ then any model J of Π' is such that $J \supset I$ because of rule $p \leftarrow A$ (note that A is always true). We conclude that $G(\Pi', J) = G(\Pi, J) \cup \{p \leftarrow p\}$, and therefore the least fixpoint of $T_{G(\Pi',J)}$, which is equal to the least fixpoint of $T_{G(\Pi,J)}$, is a subset of I. We conclude that J is not a G-stable model of Π' and hence Π' is G-incoherent. $\square$

Lemma 3
Let Π be in ASP$(\neg, \vee)$. The following relation holds: $GSM(\Pi) \equiv_{At(\Pi)} GSM(M(\Pi))$.

Proof
Without loss of generality, let us assume that all atoms in $At(\Pi)$ occur negated in Π at least once. Let I be a G-stable model of Π. Define $I^F = I \cup \{p^F \mid p \notin I\}$. We have $I^F \models M(\Pi)$. Concerning $G(M(\Pi), I^F)$ note that for each $p \in At(\Pi)$ rule $p \vee p^F \leftarrow A$ is either replaced by

$$p \vee p^F \leftarrow$$

in case $p \notin I$, or by

$$p \vee p^F \leftarrow p$$

if $p \in I$. In the first case, the rule guarantees that every model J of $G(M(\Pi), I^F)$ such that $J \subseteq I$ satisfies $p^F \in J$. Hence, rules of $G(M(\Pi), I^F)$ containing p^F can be simplified by removing p^F, which essentially results into $G(\Pi, I)$ (plus rules obtained from $p \vee p^F \leftarrow A$). In the second case, the rule is trivially satisfied by all interpretations, and therefore it can be removed from $G(M(\Pi), I^F)$. Since I is a minimal model of $G(\Pi, I^F)$, we have that I^F is a minimal model of $G(M(\Pi), I^F)$, i.e., $I^F \in GSM(M(\Pi))$.

For the other direction, let I be a G-stable model of $M(\Pi)$. We shall show that $I \cap At(\Pi)$ is a G-stable model of Π. First of all, note that $I \models A$ for any aggregate A occurring in $M(\Pi)$, and therefore $I \cap \{p, p^F\} \neq \emptyset$ because of rule $p \vee p^F \leftarrow A$, for all $p \in At(\Pi)$. Moreover, since I is a minimal model of $G(M(\Pi), I)$ by assumption, and p^F does not occur in any other rule heads, we have $|I \cap \{p, p^F\}| = 1$. We can therefore argument as in the previous direction and conclude that $I \cap At(\Pi)$ is a minimal model of $G(\Pi, I \cap At(\Pi))$, i.e., $I \cap At(\Pi) \in GSM(\Pi)$.

As a final observation, note that also $|GSM(\Pi)| = |GSM(M(\Pi))|$ holds because in any G-stable model of $M(\Pi)$ truth values for atoms of the form p^F are implied by truth values of atoms of the form p. $\square$

Theorem 5
G-coherence testing is Σ_2^P-hard for ASP($\vee$, M).

Proof
G-coherence testing is Σ_2^P-hard for a program Π in ASP($\neg$, $\vee$) (Eiter and Gottlob 1995). G-coherence of Π can be reduced to G-coherence testing of $M(\Pi)$ because of Lemma 3. Since $M(\Pi)$ can be computed in polynomial-time, eliminates negation, and only has monotone aggregates, the proof is complete. $\square$

Theorem 6
G-cautious reasoning is in P for ASP(M).

Proof
We provide a procedure for checking whether a given propositional atom p is a G-cautious consequence of Π. The procedure first checks G-coherence of Π in polynomial-time (Theorem 1). If Π is G-incoherent then the procedure rejects. Otherwise, because of Lemma 1, the unique G-stable model of Π, say I, is the least fixpoint of T_Π. The procedure then computes I in polynomial-time (Lemma 1), and accepts if $p \in I$, otherwise it rejects. $\square$

Theorem 7
G-cautious reasoning is in co-NP for programs in ASP($\neg$, M, C, N).

Proof
Let Π be in ASP($\neg$, M, C, N), and p a propositional atom. We prove that the complementary problem, checking the existence of a G-stable model I of Π such that $p \notin I$, is in NP. To this aim, let I be an interpretation such that $p \notin I$. The following is a polynomial-time procedure for checking that I is a G-stable model of Π: The procedure first builds $G(\Pi, I)$, which is disjunction-, negation and aggregate-free. Then, it computes the unique G-stable model, say J, of $G(\Pi, I)$, i.e., the least fixpoint of $T_{G(\Pi,I)}$ (Lemma 1), and accepts if $I = J$. $\square$

Theorem 8
G-cautious consequence is Π_2^P-hard for ASP($\vee$, M), ASP($\vee$, C) and ASP($\vee$, N). It is co-NP-hard for ASP(C) and ASP(N).

Proof
G-cautious reasoning is Π_2^P-hard for ASP($\neg$, $\vee$) already for programs in which negation only occurs in a rule of the form $w \leftarrow {\sim}w$ (Eiter and Gottlob 1995). Therefore, let us consider a program $\Pi = \Pi' \cup \{w \leftarrow {\sim}w\}$, where Π' is in ASP($\vee$). From Lemmas 2–3, $GSM(\Pi) \equiv_{At(\Pi)} GSM(M(\Pi)) \equiv_{At(\Pi)} GSM(C(\Pi)) \equiv_{At(\Pi)} GSM(N(\Pi))$. Let p be a propositional atom among those in $At(\Pi)$. It holds that p is a G-cautious consequence of Π if and only if p is a G-cautious consequence of the other programs. Hence, Π_2^P-hardness follows.

Similarly, G-cautious reasoning for ASP($\neg$) is co-NP-hard already for programs in which negation only occurs in a rule of the form $w \leftarrow {\sim}w$. Since $C(\Pi)$ and $N(\Pi)$ are disjunction-free if Π is disjunction-free, co-NP-hardness follows. $\square$

Appendix B Proofs of Section 4

Theorem 9
Let Π be a program. The following relation holds: $GSM(\Pi) \equiv_{At(\Pi)} FSM(rew(\Pi))$.

Proof
Let I be a G-stable model of Π. We shall show that $I' = I \cup \{p' \mid p \in At(\Pi)\}$ is an F-stable model of $rew(\Pi)$. In fact, $I' \models rew(\Pi)$ because $I \models \Pi$. Consider a model $J \subseteq I$ of the reduct $F(rew(\Pi), I)$. We have $J \cap At(\Pi) \models G(\Pi, I)$, and therefore $J \cap At(\Pi) = I$ holds because I is a G-stable model of Π by assumption. Because of rules of introduced by item 1 in Definition 4, $J \cap At(\Pi) = I$ implies $J = I$, i.e., I is an F-stable model of $rew(\Pi)$.

Let I be an F-stable model of $rew(\Pi)$. We shall show that $I \cap At(\Pi)$ is a G-stable model of Π. First of all, note that $\{p' \mid p \in At(\Pi)\} \subseteq I$ because $I \models \Pi$ and because of rules introduced by item 1 in Definition 4. Therefore, $I \cap At(\Pi) \models \Pi$ follows. Consider a model $J \subseteq I \cap At(\Pi)$ of the reduct $G(\Pi, I)$. We have $J \cup \{p' \mid p \in At(\Pi)\} \models F(rew(\Pi), I)$, and therefore $J \cup \{p' \mid p \in At(\Pi)\} = I$ because I is an F-stable model of $rew(\Pi)$ by assumption. It follows that $J = I \cap At(\Pi)$, i.e., $I \cap At(\Pi)$ is a G-stable model of Π.

Finally, note that also $|GSM(\Pi)| = |FSM(rew(\Pi))|$ holds because the mappings used above are one-to-one. □

Theorem 10
Let Π be a program. The following relation holds: $GSM(\Pi) \equiv_{At(\Pi)} FSM(str(\Pi))$.

Proof
Let I be a G-stable model of Π. We shall show that $I' = I \cup \{p' \mid p \in At(\Pi)\} \cup \{p'' \mid p \in I\}$ is an F-stable model of $str(\Pi)$. In fact, $I' \models str(\Pi)$ because $I \models \Pi$. Consider a model $J \subseteq I$ of the reduct $F(str(\Pi), I)$. We have $J \cap At(\Pi) \models G(\Pi, I)$, and therefore $J \cap At(\Pi) = I$ holds because I is a G-stable model of Π by assumption. Because of rules of the groups (i)–(ii) in Definition 5, $J \cap At(\Pi) = I$ implies $J = I$, i.e., I is an F-stable model of $str(\Pi)$.

Let I be an F-stable model of $str(\Pi)$. We shall show that $I \cap At(\Pi)$ is a G-stable model of Π. First of all, note that $\{p' \mid p \in At(\Pi)\} \subseteq I$ because $I \models \Pi$ and because of rules of the group (i). Moreover, note that $p \in I$ if and only if $p'' \in I$ because of rules of the group (iii), for all $p \in At(\Pi)$. And also note that for each aggregate A'' occurring in $str(\Pi)$, $I \models A''$ if and only if $I \cap At(\Pi) \models A$. Therefore, $I \cap At(\Pi) \models \Pi$ follows. Consider a model $J \subseteq I \cap At(\Pi)$ of the reduct $G(\Pi, I)$, and define $J' = J \cup \{p' \mid p \in At(\Pi)\} \cup \{p'' \mid p \in I\}$. We have $J' \models F(str(\Pi), I)$, and therefore $J' = I$ because I is an F-stable model of $str(\Pi)$ by assumption. It follows that $J = I \cap At(\Pi)$, i.e., $I \cap At(\Pi)$ is a G-stable model of Π.

Finally, note that also $|GSM(\Pi)| = |FSM(str(\Pi))|$ holds because the mappings used above are one-to-one. □

Theorem 11
Let Π, Π' be programs such that $\Pi \cap \Pi' = \emptyset$. For $tr \in \{rew, str\}$, the following conditions are satisfied: $tr(\Pi \cup \Pi') = tr(\Pi) \cup tr(\Pi')$, and $tr(\Pi) \cap tr(\Pi') = \emptyset$.

Proof
Immediate because the rewritings work on one rule at time. □

Theorem 12
Let Π be a program. The programs $rew(\Pi)$ and $str(\Pi)$ are polynomial-time constructible, and the following relations holds: (i) $\|rew(\Pi)\| \leq 4 \cdot |At(\Pi)| + 2 \cdot \|\Pi\|$; (ii) $\|str(\Pi)\| \leq 10 \cdot |At(\Pi)| + 2 \cdot \|\Pi\|$.

Proof
We first prove relation (i). Program $rew(\Pi)$ contains 2 rules for each atom in $At(\Pi)$, each one of size 2, and a rule for each rule of Π. The number of atoms in these rules is at most two times the number of atoms in the original rules.

We now show relation (ii). Program $rew(\Pi)$ contains 5 rules for each atom in $At(\Pi)$, each one of size 2, and a rule for each rule of Π. The number of atoms in these rules is at most two times the number of atoms in the original rules. □

Theorem 13
Let Π be a program, and I be an interpretation. If $I \models rew(\Pi)$ or $I \models str(\Pi)$ then $\{p' \mid p \in At(\Pi)\} \subseteq I$. Moreover, for each $J \subseteq I$ such that $J \models F(str(\Pi), I)$, it holds that $\{p'' \mid p \in I\} \subseteq J$.

Proof of Theorem 13
If I satisfies rules introduced by item 1 in Definition 4, or equivalently of the group (i) in Definition 5, then $\{p' \mid p \in At(\Pi)\} \subseteq I$. Consider a model $J \subseteq I$ of the reduct $F(str(\Pi), I)$. For each $p'' \in I$, $F(str(\Pi), I)$ contains a rule $p'' \leftarrow$ because of rules of the group (ii) in Definition 5. □

Theorem 14
Let Π be a program. All aggregates in $str(\Pi)$ are stratified, and if Π has no disjunction then both $rew(\Pi)$ and $str(\Pi)$ have no disjunction.

Proof
We first provide a more formal definition of stratified aggregate. The *dependency graph* of Π has a node p for each atom $p \in At(\Pi)$, and an arc from q to p if there is a rule $r \in \Pi$ such that $p \in H(r)$ and q occurs in $B(r)$, either as a possibly negated literal or in the domain of an aggregate. Π is stratified with respect to aggregates if there is no rule $r \in \Pi$ such that $p \in H(r)$ and q occurring in $B(r)$ belong to the same *strongly connected component* of Π.

Let Π be a program, and A be an aggregate in $str(\Pi)$. Hence, by construction, $dom(A) \subseteq \{p'' \mid p \in At(\Pi)\}$. Note that all rules whose head contains some atom in $dom(A)$ belong to the group (ii) in Definition 5, and therefore each atom

$p'' \in dom(A)$ belongs to a singleton strongly connected component. Stratification of aggregates in $str(\Pi)$ is thus proved.

Let Π be a program without disjunction. Program $rew(\Pi)$ and $str(\Pi)$ contain rules of the groups (i)–(iii), which have no disjunction, and rules obtained from those in Π by replacing aggregates. Hence, neither $rew(\Pi)$ nor $str(\Pi)$ has disjunction. □

References

Eiter, T. and Gottlob, G. 1995. On the computational cost of disjunctive logic programming: Propositional case. *Ann. Math. Artif. Intell. 15,* 3-4, 289–323.