TPLP: Page 1–8. © The Author(s), 2021. Published by Cambridge University Press 2021
doi:10.1017/xxxxx

Supplementary Material
Quantifying over Optimum Answer Sets

GIUSEPPE MAZZOTTA
University of Calabria, Italy

FRANCESCO RICCA
University of Calabria, Italy

MIREK TRUSZCZYNSKI
University of Kentucky

submitted xx xx xxxx; revised xx xx xxxx; accepted xx xx xxxx

1 Preliminaries on complexity classes

In this section, we recall some basic definitions of complexity classes that are used to study the complexity of the introduced formalism. For further details about NP-completeness and complexity theory we refer the reader to dedicated literature (Papadimitriou 1994). We recall that the classes Δ_k^P, Σ_k^P, and Π_k^P of the polynomial time hierarchy (PH)(Stockmeyer 1976) are defined as follows (rf. Garey and Johnson (1979)):

$$\Delta_0^P = \Sigma_0^P = \Pi_0^P = P$$

and, for all $k > 0$

$$\Delta_{k+1}^P = P^{\Sigma_k^P}, \;\; \Sigma_{k+1}^P = NP^{\Sigma_k^P}, \;\; \Pi_{k+1}^P = coNP^{\Sigma_k^P},$$

where, $NP = \Sigma_1^P$, $coNP = \Pi_1^P$, and $\Delta_2 = P^{NP}$.

In general, P^C (resp. NP^C) denotes the class of problems that can be solved in polynomial time on a deterministic (resp. nondeterministic) Turing machine with an oracle in the class C. Note that, the usage of an oracle $O \in C$ for solving a problem π is referred to as a subroutine call, during the evaluation of π, to O. The latter is evaluated in a unit of time. Among such complexity classes, the classes Δ_k^P, with $k \geq 2$, have been refined by the class $\Delta_k^P[O(\log n)]$ (also called Θ_k^P), where the number of oracle calls is bounded by $O(\log n)$, with n being the size of the input (Krentel 1992; Wagner 1990).

2 Modeling Δ_2^P in ASP$^\omega$(Q)

According to the complexity study carried out by Eiter and Gottlob (1995), given a PAP $\mathcal{A} = \langle V, T, H, M\rangle$ and a set $S \subseteq H$, the task of verifying if $S \in sol(\mathcal{A})$ is in Δ_2^P. In particular, this task can be modeled with an ASP$^\omega$(Q) Π of the form $\exists P : C$, where P is not plain. More in detail, the program C contains only one constraint, which is $\leftarrow notEntail$ whereas the program P is defined as follows:

$$P = \left\{ \begin{array}{rcll} v(x) & \leftarrow & & \forall\, x \in V \\ lit(C_i, a, t) & \leftarrow & & \forall\, a \in V \mid a \in C_i \\ lit(C_i, a, f) & \leftarrow & & \forall\, a \in V \mid \sim a \in C_i \\ h(x) & \leftarrow & & \forall\, x \in H \\ m(x) & \leftarrow & & \forall\, x \in M \\ s(x) & \leftarrow & & \forall\, x \in S \\ cl(X) & \leftarrow & lit(X, \_, \_) & \\ \{tau(X,t); tau(X,f)\} = 1 & \leftarrow & v(X) & \\ satCl(C) & \leftarrow & lit(C, A, V), tau(A, V) & \\ unsatTS & \leftarrow & cl(C), \sim satCl(C) & \\ unsatTS & \leftarrow & s(X), tau(X, f) & \\ & \leftarrow & unsatTS & \\ \{tau'(X,t); tau'(X,f)\} = 1 & \leftarrow & v(X) & \\ satCl'(C) & \leftarrow & lit(C, A, V), tau'(A, V) & \\ unsatTS' & \leftarrow & cl(C), \sim satCl'(C) & \\ unsatTS' & \leftarrow & s(X), \sim tau'(X, f) & \\ satTS & \leftarrow & \sim unsatTS' & \\ notEntail & \leftarrow & satTS, m(X), tau'(X, f) & \\ & \leftarrow_w & \sim notEntail \;\; [1@1] & \end{array} \right\}$$

Intuitively, the program P is used to verify that $T \cup S$ is consistent and $T \cup S$ entails the manifestation M. Verifying that $T \cup S \models M$ requires checking that for every truth assignment M is satisfied whenever $T \cup S$ is satisfied. Thus, the main intuition is that, in this case, we can simulate this entailment check by means of weak constraints. More precisely, an answer set $M \in AS(P)$ contains a pair of truth assignments τ and τ' such that τ guarantee the consistency of $T \cup S$, whereas τ' either violates the entailment $T \cup S \models M$ and so $notEntail \in M$ or satisfies the entailment and so $notEntail \notin M$. According to the weak constraint in P, an answer set containing *notEntail* is preferred to an answer set not containing *notEntail*. Thus, if there exists an answer set $M \in OptAS(P)$ such that $notEntail \in M$ then $T \cup S \not\models M$, and so, S is not a solution for $\mathcal{A}$. Moreover, since M is optimal then does not exist $M' \in OptAS(P)$ such that $notEntail \notin M'$, and so, Π is incoherent. Conversely, if there exists an answer set $M \in OptAS(P)$ such that $notEntail \notin M$ then every answer set $M' \in OptAS(P)$ is such that $notEntail \notin M'$. This means that $T \cup S \models$ holds and so S is a solution for $\mathcal{A}$. Since $notEntail \notin M$ then the program C is coherent and so, Π is coherent.

3 Stratified definition assumption

In this section, we demonstrate that we can assume without loss of generality that ASP$^\omega$(Q) programs satisfy the *stratified definition assumption.*

Definition 1
Let $\Pi = \Box_1 P_1 \ldots \Box_n P_n$ be an ASP$^\omega$(Q) program. Π satisfies the stratified definition assumption if for each $1 \leq i \leq n$, $\mathcal{H}(P_i) \cap at(P_j) = \emptyset$, with $1 \leq j < i$.

We demonstrate in the following that any ASP$^\omega$(Q) program Π can be transformed into a program Π' such that Π' satisfies the stratified definition assumption and is coherent whenever Π is coherent.

To this end, we recall that for an ASP expression ϵ and an alphanumeric string s, $clone^s(\epsilon)$ is the expression obtained by substituting all occurrences of each predicate p in ϵ with p^s that is a fresh predicate p^s of the same arity.

It is easy to see that there is a one-to-one correspondence between answer sets of a program P and its clone program $clone^s(P)$.

Proposition 1
Let P be an ASP program, and s be an alphanumeric string. Then $M \in AS(P)$ if and only if $clone^s(M) \in AS(clone^s(P))$.

We now introduce the *remap* function, which will be used to modify the signature of a subprogram.

Definition 2
Let P_1 and P_2 be two ASP programs, then $remap(P_2, P_1)$ is the program $clone^c(P_2) \cup \{clone^c(a) \leftarrow a \mid a \in \mathcal{B}_{P_1}\} \cup \{\leftarrow clone^c(a), {\sim}a \mid a \in \mathcal{B}_{P_1}\}$

Observation 1
Let P_1 and P_2 be two ASP programs, and $P = remap(P_2, P_1)$, then $\mathcal{H}(P) \cap at(P_1) = \emptyset$.

The above transformation has some important properties that follow from the *splitting set* (Lifschitz and Turner (1994)) theorem.

Given a program P, a *splitting set* (Lifschitz and Turner (1994)) for P is a set of atoms $U \subseteq \mathcal{B}_P$, such that for every $r \in P$ such that $H_r \cap U \neq \emptyset$, $at(B_r) \subseteq U$. Let U be a splitting set for P, $bot_U(P)$ denotes the set of rules $r \in P$ such that $at(r) \subseteq U$. Given two sets of atoms, U and X, $e_U(P, X)$ denotes the set of rules obtained from rules $r \in P$ such that $(B_r^+ \cap U) \subseteq X$ and $X \setminus (at(B_r^-) \cap U) = X$, by removing from B_r all those literals whose atom is in U.

Theorem 1 (*Lifschitz and Turner (1994)*)
Let P be an ASP program and U be a splitting set for P, then $M \in AS(P)$ if and only if $M = X \cup Y$, where $X \in AS(bot_U(P))$, and $Y \in AS(e_U(P \setminus bot_U(P))$

Now, we observe some properties of any pair of ASP programs.

Lemma 1
Let P_1, P_2 be two ASP programs such that $at(P_1) \cap \mathcal{H}(P_2) = \emptyset$, and $M_1 \in AS(P_1)$. Then, each model M such that $M \in AS(P_2 \cup \mathit{fix}_{P_1}(M_1))$ is of the form $M = M_1 \cup M_2$, where $M_2 \in AS(e_{at(P_1)}(P_2, M_1))$.

Proof
Let $M_1 \in AS(P_1)$, since P_1, P_2 are such that $at(P_1) \cap \mathcal{H}(P_2) = \emptyset$, then $U = at(P_1)$ is a splitting set for program $P = P_2 \cup \mathit{fix}_{P_1}(M_1)$. According to Theorem 1, $M \in AS(P)$ if and only if $M = X \cup Y$, where $X \in AS(bot_U(P))$, and $Y \in AS(e_U(P \setminus bot_U(P), X))$. Now, we have that program $bot_U(P) = \mathit{fix}_{P_1}(M_1)$ that by definition admits only on model M_1. Also, observe that $P \setminus bot_U(P) = P_2$, thus $M = M_1 \cup M_2$ where $M_2 \in AS(e_U(P_2, M_1))$. $\square$

Lemma 2
Let P_1 and P_2 be two ASP programs, $M_1 \in AS(P_1)$, and $P_2' = P_2 \cup \mathit{fix}_{P_1}(M_1)$. Then $M_2 \in AS(P_2')$ if and only if $M = M_1 \cup clone^c(M_2)$ is an answer set of $P' = remap(P_2, P_1) \cup \mathit{fix}_{P_1}(M_1)$.

Proof
Let P_1 and P_2 be two ASP programs and $M_1 \in AS(P_1)$. For simplicity, we denote by $P = remap(P_2, P_1)$, and $P' = P \cup \mathit{fix}_{P_1}(M_1)$, where $M_1 \in AS(P_1)$. Applying Observation 1 to P, we have that $\mathcal{H}(P) \cap at(P_1) = \emptyset$, and so from Lemma 1 we have that each $M \in AS(P')$ is such that $M = M_1 \cup M_2'$, where $M_2' \in AS(e_U(P, M_1))$, where $U = at(P_1)$. By definition $e_U(P, M_1) = e_U(remap(P_2, P_1), M_1) = clone^c(P_2) \cup \{clone^c(a) \leftarrow \mid a \in M_1\} \cup \{\leftarrow clone^c(a) \mid a \in \mathcal{B}_{P_1} \setminus M_1\} = clone^c(P_2 \cup \mathit{fix}_{P_1}(M_1))$. Since $P_2' = P_2 \cup \mathit{fix}_{P_1}(M_1)$ then $e_U(P, M_1) = clone^c(P_2')$. Thus, each answer set $M_2' \in AS(clone^c(P_2'))$ is such that $M_2' = clone^c(M_2)$ with $M_2 \in AS(P_2')$. Thus, $M = M_1 \cup clone^c(M_2)$ iff $M_2 \in AS(P_2')$. □

Corollary 1.1
Let P_1 and P_2 be two ASP programs, possibly with weak constraints, and $M_1 \in AS(P_1)$, then $M \in OptAS(remap(P_2, P_1) \cup \mathit{fix}_{P_1}(M_1))$ if and only if $M = M_1 \cup clone^c(M_2)$, where $M_2 \in OptAS(P_2 \cup \mathit{fix}_{P_1}(M_1))$.

Proof
Let P_1 and P_2 be two ASP programs, possibly with weak constraints, $M_1 \in AS(P_1)$, $P = remap(P_2, P_1) \cup \mathit{fix}_{P_1}(M_1)$, and $P_2' = P_2 \cup \mathit{fix}_{P_1}(M_1)$.

From Lemma 2, $M_2 \in AS(P_2')$ if and only if $M = M_1 \cup clone^c(M_2) \in AS(P)$.

If $M_2 \notin OptAS(P_2')$ this means that there exists $M_2' \in AS(P_2')$ such that M_2' dominates M_2, and so M is dominated by $M' = M_1 \cup clone^c(M_2')$. Thus $M \notin OptAS(P)$.

Conversely, if $M_2 \in OptAS(P_2')$ then for every $M_2' \in AS(P_2')$, M_2 is not dominated by M_2' and so for every $M' = M_1 \cup clone(M_2') \in AS(P)$, M is not dominated by M' and so $M \in OptAS(P)$. □

We are now ready to introduce our main program transformation.

Definition 3
Given an ASP$^\omega$(Q) program Π of the form $\Box_1 P_1 \ldots \Box_n P_n : C$, and $1 < i \leq n$ we define the function $remap(\Pi, i)$ that computes the ASP$^\omega$(Q) program: $remap(\Pi, i) = \Box_1 P_1' \ldots \Box_n P_n' : C'$, and $1 \leq j \leq n$ such that:

$$P_j' = \begin{cases} P_j & j <= i \\ remap(P_j, P_i) & j = i + 1 \\ clone^c(P_j) & j > i + 1 \end{cases}$$

and $C' = clone^c(C)$.

Proposition 2
Let Π be an ASP$^\omega$(Q) program of the form $\Box_1 P_1 \ldots \Box_n P_n : C$, and $1 \leq i \leq n$, then Π is coherent if and only if $remap(\Pi, i)$ is coherent.

Proof
The thesis follows from the definition of quantified answer set, Proposition 1.1 and Lemma 2. □

Proposition 3

Given an ASP$^\omega$(Q) program Π. We define the sequence of ASP$^\omega$(Q) programs

$$\Pi_j = \begin{cases} remap(\Pi, 2) & j = 2 \\ remap(\Pi_{j-1}, j) & 2 < j \leq n \end{cases}$$

Then Π_n is coherent iff Π is coherent.

Corollary 1.2
Given an ASP$^\omega$(Q) program Π such that $\Box_1 = \exists$, then $QAS(\Pi_n) = QAS(\Pi)$.

Proof
It follows from Proposition 3 once we observe that the first programs of Π_n and Π are the same. $\Box$

Thus, without loss of generality, we assume that an ASP$^\omega$(Q) program is of the form $\Box_1 P_1 \dots \Box_n P_n$, where for each $1 \leq i \leq n$, $\mathcal{H}(P_i) \cap at(P_j) = \emptyset$, with $1 \leq j < i$.

4 Rewriting into plain ASP(Q)

In this section, we are going to prove the correctness of the transformation outlined in the corresponding section of the main paper. In what follows we assume that Π is an ASP$^\omega$(Q) program of the form

$$\Box_1 P_1 \; \Box_2 P_2 \; \cdots \; \Box_n P_n : C : C^w, \tag{1}$$

where, for each $i = 1, \dots, n$, $\Box_i \in \{\exists^{st}, \forall^{st}\}$, P_i is an ASP program possibly with weak constraints, C is a (possibly empty) stratified program with constraints, and C^w is a (possibly empty) set of weak constraints such that $B_{C^w} \subseteq B_{P_1}$.

We recall some useful definitions introduced in the main paper. Given program Π of the form (1) we say that two consecutive subprograms P_i and P_{i+1} are *alternating* if $\Box_i \neq \Box_{i+1}$, and are *uniform* otherwise. A program Π is *quantifier-alternating* if $\Box_i \neq \Box_{i+1}$ for $1 \leq i < n$. A subprogram P_i is *plain* if it contains no weak constraints $\mathcal{W}(P_i) = \emptyset$, and Π is *plain* if both all P_i are plain, and $C^w = \emptyset$.

4.1 Rewriting uniform plain subprograms.

First of all, we show how two plain uniform subprograms can be absorbed in a single equi-coherent subprogram. This is done by the transformation $col_1(\cdot)$ as follows.

Lemma 3 (Correctness $col_1(\cdot)$ transformation)
Let program Π be such that $n \geq 2$ and the first two subprograms are plain and uniform, i.e., $\Box_1 = \Box_2$, and $\mathcal{W}(P_1) = \mathcal{W}(P_2) = \emptyset$, then Π is coherent if and only if $col_1(\Pi) = \Box_1 P_1 \cup P_2 \Box_3 P_3 \dots \Box_n P_n : C$ is coherent.

Proof
The proof follows from the stratified definition assumption. In particular, since $at(P_1) \cap \mathcal{H}(P_2) = \emptyset$ then $U = at(P_1)$ is a splitting set for $P = P_1 \cup P_2$, where $bot_U(P) = P_1$ and $P \backslash bot_U(P) = P_2$. From Theorem 1, $M \in AS(P)$ if and only if $M = M_1 \cup M_2$, where $M_1 \in AS(P_1)$ and $M_2 \in AS(e_U(P_2, M_1))$, and so, from Lemma 1, $M \in AS(P_2 \cup \mathit{fix}_{P_1}(M_1))$.

Thus, accordingly, the program P preserves all the answer sets of P_1 and P_2. Since no weak constraints appear in P_1 and P_2, then $AS(P) = OptAS(P)$ and so the coherence is preserved. $\square$

4.2 Rewriting uniform notplain-plain subprograms.

Next transformations apply to pairs of uniform subprograms P_1, P_2 such that P_1 is not plain and P_2 is plain. To this end, we recall the definition of $or(\cdot, \cdot)$ transformation.

Definition 4
Let P be an ASP program, and l be a fresh atom not appearing in P, then $or(P, l) = \{H_r \leftarrow B_r, {\sim}l \mid r \in P\}$

Observation 2 (Trivial model existence)
Let P be an ASP program, and l be a fresh literal not appearing in P, then the following hold: $\{l\}$ is the unique answer set of $or(P, l) \cup \{l \leftarrow\}$; and $AS(or(P, l) \cup \{\leftarrow l\}) = AS(P)$.

Intuitively, if the fact $l \leftarrow$ is added to $or(P, l)$ then the interpretation $I = \{l\}$ trivially satisfies all the rules and is minimal, thus it is an answer set. On the other hand, if we add the constraint $\leftarrow l$, requiring that l is false in any answer set, then the resulting program behaves precisely as P since literal ${\sim}l$ is trivially true in all the bodies of the rules of the program.

We are now ready to introduce the next rewriting function $col_2(\cdot)$.

Definition 5 (Collapse notplain-plain existential subprograms)
Let Π be an ASP$^\omega$(Q) program of the form $\exists P_1 \exists P_2 \ldots \Box_n P_n : C$, where $\mathcal{W}(P_1) \neq \emptyset$, $\mathcal{W}(P_i) = \emptyset$, with $1 < i \leq n$, and $\Box_i \neq \Box_{i+1}$ with $1 < i < n$, then:

$$col_2(\Pi) = \begin{cases} \exists P_1 \cup or(P_2, unsat) \cup W : C \cup \{\leftarrow unsat\} & n = 2 \\ \exists P_1 \cup or(P_2, unsat) \cup W \; \forall P_3' : C \cup \{\leftarrow unsat\} & n = 3 \\ \exists P_1 \cup or(P_2, unsat) \cup W \; \forall P_3' \; \exists P_4 \cup \{\leftarrow unsat\} \ldots \Box_n P_n : C & n > 3 \end{cases}$$

where $W = \{\{unsat\} \leftarrow\} \cup \{\leftarrow_w unsat \; [1@l_{min} - 1]\}$, with l_{min} be the lowest level in $\mathcal{W}(P_1)$ and $unsat$ is a fresh symbol not appearing anywhere else, and $P_3' = or(P_3, unsat)$.

Lemma 4 (Correctness $col_2(\cdot)$ transformation)
Let Π be an ASP$^\omega$(Q) program of the form $\exists P_1 \exists P_2 \ldots \Box_n P_n : C$, where $\mathcal{W}(P_1) \neq \emptyset$, $\mathcal{W}(P_i) = \emptyset$, with $1 < i \leq n$, and $\Box_i \neq \Box_{i+1}$ with $1 < i < n$. Then Π is coherent if and only if $col_2(\Pi)$ is coherent.

Proof (sketch)
First observe that if P_1 is not coherent then both Π (by definition) and $\exists P_1 \cup or(P_2, unsat)$ (for the splitting theorem) are not coherent, and thus (by definition of quantified answer set) also $col_2(\Pi)$ is not coherent. Next observe that, if P_1 is coherent, an optimal answer set of $P_1 \cup or(P_2, unsat)$ contains $unsat$ only if any optimal answer set m of P_1 is such that $P_2 \cup fix_{P_1}(m)$ is incoherent. In this case, Π is not coherent, and since the subprogram following P_2 contains the constraint $\leftarrow unsat$, also $col_2(\Pi)$ is not coherent. On the other hand, if $unsat$ is false in any optimal answer set of the first subprogram of $col_2(\Pi)$, then (if $n \geq 3$) $P_3\prime$ behaves as P_3, and the constraint $\leftarrow unsat$ occurring in the next

subprogram (i.e., P_4 or C) is trivially satisfied. Thus (by the definition of quantified answer set) $col_2(\Pi)$ is coherent whenever Π is coherent. $\square$

Proof

Let $P = P_1 \cup or(P_2, unsat) \cup W$, where $W = \{\{unsat\} \leftarrow\} \cup \{\leftarrow_w unsat\ [1@l_{min} - 1]\}$, with l_{min} being the lowest level in $\mathcal{W}(P_1)$ and *unsat* is a fresh symbol not appearing in Π.

Since P_1 and P_2 satisfy the stratified definition assumption, then, from the splitting theorem, each answer set of P can be computed by fixing any answer set $M_1 \in AS(P_1)$ in the program $or(P_2, unsat) \cup W$. In turn, $M \in AS(P)$ if and only if $\exists M_1 \in AS(P_1)$ and $M \in AS(or(P_2, unsat) \cup W \cup fix_{P_1}(M_1))$.

From the definition of quantified answer set, if P_1 is incoherent then Π is incoherent. Analogously, P is incoherent and so, also $col_2(\Pi)$ is incoherent.

On the other hand, i.e. P_1 is coherent, then $AS(P) = \{M_1 \cup \{unsat\} \mid M_1 \in AS(P_1)\} \cup \{M \mid \exists M_1 \in AS(P_1) \wedge M \in AS(P_2 \cup fix_{P_1}(M_1))\}$.

Let M in $AS(P)$ such that M is obtained from $M_1 \in AS(P_1) \setminus OptAS(P_1)$ (i.e. either $M = M_1 \cup \{unsat\}$ or $M \in AS(P_2 \cup fix_{P_1}(M_1)))$. In this case, we know that there exists $M' = M'_1 \cup \{unsat\} \in AS(P)$, where $M'_1 \in OptAS(P_1)$, and so, since weak constraints in P_1 are defined at the highest priority level, then M is dominated by M'. In turn $M \notin OptAS(P)$.

Let $M = M_1 \cup \{unsat\} \in AS(P)$, with $M_1 \in OptAS(P)$. In this case, according to the weak constraint $\leftarrow_w unsat\ [1@l_{min} - 1]$, M is dominated only by any $M' \in AS(P)$ such that $M' \in AS(P_2 \cup fix_{P_1}(M'_1))$, with $M'_1 \in OptAS(P_1)$, since M' does not violate the weak constraint $\leftarrow_w unsat\ [1@l_{min} - 1]$.

Thus, if there exists $M \in OptAS(P)$ such that $unsat \in M$ then $OptAS(P) = \{M_1 \cup \{unsat\} \mid M_1 \in OptAS(P_1)\}$. This means that for every $M_1 \in OptAS(P_1)$ the program $P_2 \cup fix_{P_1}(M_1)$ is incoherent and so, from the definition of quantified answer set, Π is incoherent. Since *unsat* is true, in this case, in every optimal answer set of P and the subprogram following P_2 contains the constraint $\leftarrow unsat$ then also $col_2(\Pi)$ is incoherent.

On the other hand, if there exists $M \in OptAS(P)$ such that $unsat \notin M$ then $OptAS(P) = \{M \mid M \in AS(P_2 \cup fix_{P_1}(M_1)) \wedge M_1 \in OptAS(P_1)\}$. In this case, since *unsat* is false in any optimal answer set of P, then (if $n \geq 3$) $P_{3'}$ behaves as P_3, and the constraint $\leftarrow unsat$ occurring in the next subprogram (i.e., P_4 or C) is trivially satisfied. Thus (by the definition of quantified answer set) $col_2(\Pi)$ is coherent whenever Π is coherent. $\square$

Definition 6 (*Collapse notplain-plain universal subprograms*)

Let Π be an ASP$^{\omega}$(Q) program of the form $\forall P_1 \forall P_2 \ldots \Box_n P_n : C$, where $\mathcal{W}(P_1) \neq \emptyset$, $\mathcal{W}(P_i) = \emptyset$, with $1 < i \leq n$, and $\Box_i \neq \Box_{i+1}$ with $1 < i < n$, then:

$$col_3(\Pi) = \begin{cases} \forall P_1 \cup or(P_2, unsat) \cup W : or(C, unsat) & n = 2 \\ \forall P_1 \cup or(P_2, unsat) \cup W\ \exists P'_3 : or(C, unsat) & n = 3 \\ \forall P_1 \cup or(P_2, unsat) \cup W\ \exists P'_3\ \forall P_4 \cup \{\leftarrow unsat\} \ldots \Box_n P_n : C & n > 3 \end{cases}$$

where $W = \{\{unsat\} \leftarrow\} \cup \{\leftarrow_w unsat\ [1@l_{min} - 1]\}$, with l_{min} be the lowest level in $\mathcal{W}(P_1)$ and *unsat* is a fresh symbol not appearing anywhere else, and $P'_3 = or(P_3, unsat)$.

Lemma 5 (*Correctness* $col_3(\cdot)$ *transformation*)
Let Π be an ASP$^\omega$(Q) program of the form $\forall P_1 \forall P_2 \ldots \Box_n P_n : C$, where $\mathcal{W}(P_1) \neq \emptyset$, $\mathcal{W}(P_i) = \emptyset$, with $1 < i \leq n$, and $\Box_i \neq \Box_{i+1}$ with $1 < i < n$. Then Π is coherent if and only if $col_3(\Pi)$ is coherent.

Proof (sketch)
The proof follows the same idea used for proving Lemma 4. In this case, if P_1 is not coherent then Π (by definition) is coherent. At the same time $\forall P_1 \cup or(P_2, unsat)$ (for the splitting theorem) is also incoherent, and thus (according to the coherence of ASP$^\omega$(Q)) also $col_2(\Pi)$ is coherent. On the other hand, if P_1 is coherent then an optimal answer set of $\forall P_1 \cup or(P_2, unsat)$ contains $unsat$ only if any optimal answer set m of P_1 is such that $P_2 \cup fix_{P_1}(m)$ is incoherent. In this case, Π is coherent, and, according to the definition of optimal answer set, every optimal answer set of $\forall P_1 \cup or(P_2, unsat)$ contains $unsat$. Thus, the subprogram following P_2 are trivially satisfied ($or(C, unsat)$, $or(P_3, unsat)$ or $\forall P_4 \cup \{\leftarrow unsat\}$), and so also $col_2(\Pi)$ is coherent. On the other hand, if $unsat$ is false in any optimal answer set of the first subprogram of $col_2(\Pi)$, then (if $2 \leq n \leq 3$) $or(C, unsat)$ behaves as C, (if $n \geq 3$) P'_3 behaves as P_3, and the constraint $\leftarrow unsat$ occurring in the next subprogram (i.e. P_4) is trivially satisfied. Thus (according to the coherence of ASP$^\omega$(Q)) $col_2(\Pi)$ is coherent whenever Π is coherent. $\Box$

Proof
Let $P = P_1 \cup or(P_2, unsat) \cup W$, where $W = \{\{unsat\} \leftarrow\} \cup \{\leftarrow_w unsat\ [1@l_{min} - 1]\}$, with l_{min} being the lowest level in $\mathcal{W}(P_1)$ and $unsat$ is a fresh symbol not appearing in Π.

As it has been observed in the proof of Lemma 4, since P_1 and P_2 satisfy the stratified definition assumption, then, from the splitting theorem, each answer set of P can be computed by fixing any answer set $M_1 \in AS(P_1)$ in the program $or(P_2, unsat) \cup W$. In turn, $M \in AS(P)$ if and only if $\exists M_1 \in AS(P_1)$ and $M \in AS(or(P_2, unsat) \cup W \cup fix_{P_1}(M_1))$.

According to the coherence of ASP$^\omega$(Q) programs, if P_1 is incoherent then Π is coherent. Analogously, P is incoherent and so, also $col_2(\Pi)$ is coherent.

On the other hand, i.e. P_1 is coherent, then $AS(P) = \{M_1 \cup \{unsat\} \mid M_1 \in AS(P_1)\} \cup \{M \mid \exists M_1 \in AS(P_1) \wedge M \in AS(P_2 \cup fix_{P_1}(M_1))\}$.

Thus, from the observation made in the proof of Lemma 4, if there exists $M \in OptAS(P)$ such that $unsat \in M$ then $OptAS(P) = \{M_1 \cup \{unsat\} \mid M_1 \in OptAS(P_1)\}$. This means that for every $M_1 \in OptAS(P_1)$ the program $P_2 \cup fix_{P_1}(M_1)$ is incoherent and so, from the coherence of ASP$^\omega$(Q), Π is coherent. Since $unsat$ is true in every optimal answer set of P then, in this case, the subprogram following P_2 (i.e. $or(C, unsat)$, $or(P_3, unsat)$ or $\forall P_4 \cup \{\leftarrow unsat\}$) are trivially satisfied, and so also $col_2(\Pi)$ is coherent.

On the other hand, if there exists $M \in OptAS(P)$ such that $unsat \notin M$ then $OptAS(P) = \{M \mid M \in AS(P_2 \cup fix_{P_1}(M_1)) \wedge M_1 \in OptAS(P_1)\}$. In this case since $unsat$ is false in any optimal answer set of P, then (if $2 \leq n \leq 3$) $or(C, unsat)$ behaves as C, (if $n \geq 3$) P'_3 behaves as P_3, and the constraint $\leftarrow unsat$ occurring in the next subprogram (i.e. P_4) is trivially satisfied. Thus (according to the coherence of ASP$^\omega$(Q)) $col_2(\Pi)$ is coherent whenever Π is coherent. $\Box$

4.3 Rewrite subprograms with weak constraints.

The following transformations have the role of eliminating weak constraints from a subprogram by encoding the optimality check in the subsequent subprograms. To this end, we recall the definition of $check(\cdot)$ transformation that is useful for simulating the cost comparison of two answer sets of an ASP program P.

Definition 7 (*Transform weak constraints*)
Let P be an ASP program with weak constraints, then

$$check(P) = \begin{cases} v_c(w,l,T) \leftarrow b_1,\dots,b_m & \forall c: \ \leftarrow_w b_1,\dots,b_m[w@l,T] \in P \\ cl_P(C,L) \leftarrow level(L), C = \#sum\{w_{c_1};\dots;w_{c_n}\} & \\ clone^o(v_c(w,l,T) \leftarrow b_1,\dots,b_m) & \forall c: \ \leftarrow_w b_1,\dots,b_m[w@l,T] \in P \\ clone^o(cl_P(C,L) \leftarrow level(L), C = \#sum\{w_{c_1};\dots;w_{c_n}\}) & \\ diff(L) \leftarrow cl_P(C1,L), cl^o_P(C2,L), C1 \neq C2 & \\ hasHigher(L) \leftarrow diff(L), diff(L1), L < L1 & \\ higest(L) \leftarrow diff(L), \sim hasHigher(L) & \\ dom_P \leftarrow higest(L), cl_P(C1,L), cl^o_P(C2,L), C2 < C1 & \end{cases}$$

where each w_{c_i} is an aggregate element of the form $W,T : v_{c_i}(W,L,T)$.

Thus, the first two rules compute in predicate cl_P the cost of an answer set of P w.r.t. its weak constraints, and the following two rules do the same for $clone^o(P)$. Then, the last four rules derive dom_P for each answer set of P that is dominated by $clone^o(P)$.

Observation 3
Let P be an ASP program with weak constraints, and $M_1, M_2 \in AS(P)$, then M_1 is dominated by M_2 if and only if $check(P) \cup fix_P(M_1) \cup clone^o(fix_P(M_2))$ admits an answer set M such that $dom_P \in M$.

Definition 8 (*Transform existential not-plain subprogram*)
Let Π be an existential alternating ASP$^\omega$(Q) program such that all subprograms are plain except the first one (i.e. $\mathcal{W}(P_1) \neq \emptyset$, $\mathcal{W}(P_i) = \emptyset$, $1 < i \leq n$), then

$$col_4(\Pi) = \begin{cases} \exists \mathcal{R}(P_1) \forall clone^o(\mathcal{R}(P_1)) \cup check(P_1) : \{\leftarrow dom_{P_1}\} \cup C & n = 1 \\ \exists \mathcal{R}(P_1) \forall P_2' : \{\leftarrow dom_{P_1}\} \cup C & n = 2 \\ \exists \mathcal{R}(P_1) \forall P_2' \exists P_3 \cup \{\leftarrow dom_{P_1}\} \dots \Box_n P_n : C & n \geq 3 \end{cases}$$

where $P_2' = clone^o(\mathcal{R}(P_1)) \cup check(P_1) \cup or(P_2, dom_{P_1})$.

Lemma 6 (*Correctness $col_4(\cdot)$ transformation*)
Let Π be an existential alternating ASP$^\omega$(Q) program such that all subprograms are plain except the first, then Π is coherent if and only if $col_4(\Pi)$ is coherent.

Proof (sketch)
Intuitively, $col_4(\Pi)$ is structured in such a that if there exists an answer set m_1 of P_1 (any, also not optimal ones), such that for any other answer set of m^o_i of P_1 (computed by cloning P_1 in the second subprogram of $col_4(\Pi)$) either dom_{P_1} is derived or m_1 is optimal. In the first case, if $n \geq 2$ dom_{P_1} inhibits the rules of P_2, and the next subprogram discards m_1, as expected since it is not optimal. In the second case, the next subprograms of $col_4(\Pi)$ behave as those of Π, and the constraint $\leftarrow dom_{P_1}$ occurring next is trivially satisfied. Thus, m_1 is a quantified answer set of $col_4(\Pi)$ only if m_1 is a quantified answer set of Π. □

Proof

Let Π be an existential alternating ASP$^{\omega}$(Q) program such that all subprograms are plain except the first. First of all we observe that $AS(\mathcal{R}(P_1)) = AS(P_1) \subseteq OptAS(P_1)$. Thus, if P_1 is incoherent then $AS(\mathcal{R}(P_1)) = AS(P_1) = \emptyset$, and so, also $\mathcal{R}(P_1)$ is incoherent. Indeed, from the definition of quantified answer set, both Π and $col_4(\Pi)$ are incoherent.

On the other hand, let $M_1 \in AS(P_1)$ and P denotes the second subprogram of $col_4(\Pi)$ ($P = clone^o(\mathcal{R}(P_1)) \cup check(P_1)$, if $n = 1$, otherwise $P = clone^o(\mathcal{R}(P_1)) \cup check(P_1) \cup or(P_2, dom_{P_1})$).

If $M_1 \notin OptAS(P_1)$ then M_1, from the definition of quantified answer set, is not a quantified answer set for Π. Since M_1 is not optimal then we know that there exists $M_1' \in OptAS(P_1)$ such that M_1 is dominated by M_1', and so there exists $M \in AS(P \cup fix_{\mathcal{R}(P_1)}(M_1))$ such that $clone^o(M_1') \subseteq M$ and $dom_{P_1} \in M$. Thus, M violates the strong constraint $\leftarrow dom_{P_1}$ in the following subprograms and so, from the definition of quantified answer set, M_1 is not a quantified answer set of $col_4(\Pi)$.

Conversely, if $M_1 \in OptAS(P_1)$ then for every $M \in AS(P \cup fix_{\mathcal{R}(P_1)}(M_1))$, $dom_{P_1} \notin M$ and so the constraint $\leftarrow dom_{P_1}$ added in the subsequent subprograms is trivially satisfied. Since atoms in $clone^o(\mathcal{R}(P_1))$ and $check(P_1)$ do not appear anywhere else then they do not affect the coherence of $col_4(\Pi)$ and so M_1 is a quantified answer set of $col_4(\Pi)$ whenever M_1 is a quantified answer set for Π. □

Definition 9 (*Transform universal not-plain subprogram*)

Let Π be a universal alternating ASP$^{\omega}$(Q) program such that all subprograms are plain except the first one (i.e. $\mathcal{W}(P_1) \neq \emptyset$, $\mathcal{W}(P_i) = \emptyset$ $1 < i \leq n$), then

$$col_5(\Pi) = \begin{cases} \forall \mathcal{R}(P_1) \exists clone^o(\mathcal{R}(P_1)) \cup check(P_1) : or(C, dom_{P_1}) & n = 1 \\ \forall \mathcal{R}(P_1) \exists P_2' : or(C, dom_{P_1}) & n = 2 \\ \forall \mathcal{R}(P_1) \exists P_2' \forall P_3 \cup \{\leftarrow dom_{P_1}\} \ldots \Box_n P_n : C & n \geq 3 \end{cases}$$

where $P_2' = clone^o(\mathcal{R}(P_1)) \cup check(P_1) \cup or(P_2, dom_{P_1})$.

Lemma 7 (*Correctness $col_5(\cdot)$ transformation*)

Let Π be a universal alternating ASP$^{\omega}$(Q) program such that all subprograms are plain except the first, then Π is coherent if and only if $col_5(\Pi)$ is coherent.

The proof of Lemma 7 can be established using a dual argument with respect to that employed for Lemma 6.

4.4 Translate ASP$^{\omega}$(Q) to ASP(Q).

Algorithm 1 defines a procedure for rewriting an ASP$^{\omega}$(Q) program Π into an ASP(Q) program Π', made of at most $n + 1$ alternating quantifiers, such that Π is coherent if and only if Π' is coherent. We recall that in Algorithm 1, we make use of some (sub)procedures and dedicated notation. More precisely, for a program Π of the form (1), $\Pi^{\geq i}$ denotes the *i-th suffix program* $\Box_i P_i \ldots \Box_n P_n : C$, with $1 \leq i \leq n$. (i.e., the one obtained from Π removing the first $i - 1$ quantifiers and subprograms). Moreover, procedure $removeGlobal(\Pi)$ builds an ASP(Q) program from a plain one in input (roughly, it removes the global constraint program C^w). Given two programs Π_1 and Π_2, $replace(\Pi_1, i, \Pi_2)$ returns the ASP$^{\omega}$(Q) program obtained from Π_1 by replacing program

Algorithm 1 Rewrite from ASP$^\omega$(Q) to ASP(Q)

Input : An ASP$^\omega$(Q) program Π
Output: A quantifier-alternating ASP(Q) program

1 **begin**
2 $\quad s := 0; \quad \Pi_0 := \Pi$
3 $\quad$ **do**
4 $\quad\quad$ $stop := \top$
5 $\quad\quad$ **for all** $ProgramType \in [1,5]$ **do**
6 $\quad\quad\quad$ Let $i \in [1,n]$ be the largest index such that $\Pi_s^{\geq i}$ is of the type $ProgramType$
7 $\quad\quad\quad$ **if** $i \neq \bot$ **then**
8 $\quad\quad\quad\quad$ $\Pi_{s+1} := replace(\Pi_s, i, col_{ProgramType}(\Pi_s^{\geq i}))$
9 $\quad\quad\quad\quad$ $s := s + 1;$
10 $\quad\quad\quad\quad$ $stop := \bot$
11 $\quad\quad\quad\quad$ *break* // go to line 12
12 $\quad$ **while** $stop \neq \top$;
13 $\quad$ **return** $removeGlobal(\Pi_s)$

$\Pi_1^{\geq i}$ by Π_2, for example $replace(\exists P_1 \forall P2 \exists P_3 : C, 2, \exists P_4 : C)$ returns $\exists P_1 \exists P_4 : C$. With a little abuse of notation, we write that a program is of type $T \in [1,5]$ if it satisfies the conditions for applying the rewriting col_T defined above (cfr., Lemmas 3-7). For example, when type $T = 1$ we check that the first two subprograms of Π are plain and uniform so that col_1 can be applied to program Π.

In order to obtain a quantifier alternating ASP(Q) program from the input Π, Algorithm 1 generates a sequence of programs by applying at each step one of the col_T transformations. In particular, at each iteration s, the innermost suffix program, say $\Pi_s^{\geq i}$, that is of current type T is identified. Then the next program Π_{s+1} is built by replacing $\Pi_s^{\geq i}$ by $col_T(\Pi_s^{\geq i})$. Algorithm terminates when no transformation can be applied, and returns the program $removeGlobal(\Pi_s)$.

Theorem 2 (*ASP$^\omega$(Q) to ASP(Q) convergence and correctness*)
Given program Π, Algorithm 1 terminates and returns an alternating ASP(Q) program Π' that is Π' is coherent iff Π is coherent, and $nQuant(\Pi') \leq nQuant(\Pi) + 1$.

Proof. (Sketch)
Algorithm 1 repeatedly simplifies the input by applying $col_T(\cdot)$ procedures ($T \in [1,5]$) until none can be applied. So, the results follow from the Lemmas 3-7 that ensures the input can be converted to an equi-coherent plain ASP$^\omega$(Q) program. Note that, unless the innermost subprogram of Π is not plain, no additional quantifier is added during the execution of Algorithm 1 (if anything, some may be removed), so $nQuant(\Pi') \leq nQuant(\Pi) + 1$. $\square$

Proof
At each step s, Algorithm 1 searches for the innermost suffix subprogram $\Pi_s^{\geq i}$ such that either (i) $\Pi_s^{\geq i}$ begins with two consecutive quantifiers of the same type (i.e., it is of type 1,2 or 3), or (ii) $\Pi_s^{\geq i}$ begins with a not plain subprogram followed by a quantifier alternating sequence of plain subprograms (i.e., it is of type 4 or 5). In case (i), one of the subprocedures col_1, col_2, or col_3 is applied, which results in the computation of program Π_{s+1} having one less pair of uniform subprograms (i.e., $nQuant(\Pi_{s+1}) = nQuant(\Pi_s) - 1$). In case (ii), one of the subprocedures col_4, col_5 is applied, which results in the compu-

tation of program Π_{s+1} such that its i-th subprogram is plain. After applying col_4, col_5 we have that $nQuant(\Pi_{s+1}) \leq nQuant(\Pi_s) + 1$, indeed if $i = nQuant(\Pi_s)$ one more quantifier subprogram is added. So the algorithm continues until neither condition (i) nor (ii) holds. This happens when Π_s is a plain quantifier alternating program. Note that, unless the innermost subprogram of Π is not plain, no additional quantifier is added during the execution of Algorithm 1 (if anything, some may be removed), so $nQuant(\Pi') \leq nQuant(\Pi) + 1$. □

5 Complexity issues

In this section, we recall the complexity results related to verifying the coherence of ASP$^\omega$(Q) programs and provide full proof for completeness results.

Theorem 3 (*Upper bound*)
The coherence problem of an ASP$^\omega$(Q) program Π is in: (i) Σ^p_{n+1} for existential programs, and (ii) Π^p_{n+1} for universal programs, where $n = nQuant(\Pi)$.

Proof
Let Π' be the result of applying Algorithm 1 to Π. Then, Π' is a quantifier-alternating plain program with at most $n = nQuant(\Pi) + 1$ quantifiers that is coherent iff Π is coherent (Theorem 2). Thesis follows from Theorem 3 in the paper by Amendola et al. (2019). □

Theorem 4 (*Lower bound*)
The coherence problem of an ASP$^\omega$(Q) program is hard for (i) Σ^p_n for existential programs, and hard for (ii) Π^p_n for universal programs, where $n = nQuant(\Pi)$.

Proof
The proof trivially follows from the observation that any quantifier-alternating ASP(Q) program with n quantifiers is a plain ASP$^\omega$(Q) program where $C^w = \emptyset$. □

Corollary 4.1 (*First completeness result*)
The coherence problem of an ASP$^\omega$(Q) program where the last subprogram is plain (i.e., $\mathcal{W}(P_n) = \emptyset$) is (i) Σ^p_n-complete for existential programs, and (ii) Π^p_n-complete for universal programs, where $n = nQuant(\Pi)$.

Proof
The statement follows from Theorem 3 of Amendola et al. (2019).
(Membership) Let Π be an ASP$^\omega$(Q) program where the last subprogram is plain. We observe that by applying Algorithm 1 we obtain a quantifier-alternating plain ASP(Q) program Π' with at most $n + 1$ quantifiers such that Π' is coherent if and only if Π is coherent. Since the only case in which an extra quantifier is added is when the last subprogram is not plain (i.e. $\mathcal{W}(P_n) \neq \emptyset$) then $nQuant(\Pi') \leq n$ then the membership follows.
(Hardness) The hardness trivially follows by observing that any quantifier-alternating ASP(Q) program with n quantifiers is trivially encoded as a plain ASP$^\omega$(Q) program where $C^w = \emptyset$. □

Note that, in plain ASP(Q) (as well as in related formalisms (Stockmeyer 1976; Fandinno et al. 2021)), the complexity of coherence correlates directly with the number of quantifier alternations (Amendola et al. 2019). Perhaps somewhat unexpectedly at first glance, it is not the case of ASP$^\omega$(Q), where one can "go up one level" with two consecutive quantifiers of the same kind. This observation is exemplified below.

Theorem 5 (*Second completeness results*)
Deciding coherence of uniform existential ASP$^\omega$(Q) programs with two quantifiers (i.e. $n = 2$) such that P_2 is not plain is Σ_2^p-complete.

Proof. (Sketch)
Membership follows from Theorem 3. Hardness is proved by a reduction of an existential 2QBF in DNF by modifying the QBF encoding in ASP(Q) presented in Theorem 2 of Amendola et al. (2019). In particular, a weak constraint in P_2 simulates the forall quantifier by preferring counterexamples that are later excluded by the final constraint C. □

Proof
(Hardness) Let us consider a QBF formula $\Phi = \exists X_1 \forall X_2 \phi$, where X_1, X_2 are two disjoint sets of propositional variables, and ϕ is a 3-DNF formula over variables in X_1, X_2 of the form $D_1 \vee \ldots \vee D_n$, where each conjunct $D_i = l_1^i \wedge l_2^i \wedge l_3^i$, with $1 \leq i \leq n$. It is known that the task of verifying the satisfiability of Φ is a Σ_2^P-complete problem (Stockmeyer 1976), thus, we reduce Φ to an ASP$^\omega$(Q) program Π of the form $\exists P_1 \exists P_2 : C$ where

$$P_1 = \left\{ \ \{x\} \leftarrow \quad \forall x \in X_1 \ \right\} \qquad P_2 = \left\{ \begin{array}{ll} \{x\} \leftarrow & \forall x \in X_2 \\ sat \leftarrow l_1^i, l_2^i, l_3^i & \forall D_i \in \phi \\ \leftarrow_w sat[1@1] & \end{array} \right\} \qquad C = \{\leftarrow {\sim} sat\}$$

We recall that 2-QBF formula Φ of the form $\exists X_1 \forall X_2 \ \phi$ where is satisfiable if and only if there exists a truth assignment τ_1 for variables in X_1 such that for every truth assignment τ_2 of variables in X_2, the formula ϕ is satisfied w.r.t. τ_1 and τ_2 (i.e. at least a conjunct in ϕ is true w.r.t. τ_1 and τ_2).

To this end, the program P_1 encodes the truth assignments of variables in X_1 by means of a choice rule for each $x \in X_1$. Analogously, P_2 encodes the truth assignments of variables in X_2, by means of a choice rule for each $x \in X_2$, and checks whether ϕ is satisfied or not by means of a rule for each conjunct D_i, that derives the atom sat whenever D_i is true.

Thus, if Φ is satisfiable then there exists $M_1 \in AS(P_1) = OptAS(P_1)$, such that M_1 encodes τ_1, and for every answer set $M_2 \in AS(P_2')$, $sat \in M_2$, where $P_2' = P_2 \cup fix_{P_1}(M_1)$.

Since sat appears in every $M_2 \in AS(P_2')$ then for every $M_2 \in AS(P_2')$, $\mathcal{C}(P_2', M_2, 1) = 1$, and so, $AS(P_2') = OptAS(P_2')$. Thus, for every $M_2 \in OptAS(P_2')$, $C \cup fix_{P_2'}(M_2)$ is coherent and so, $M_1 \in QAS(\Pi)$.

Conversely, if Φ is unsatisfiable then for every truth assignment τ_1 over variables X_1, there exists a truth assignment τ_2 over variables X_2 such that ϕ is unsatisfiable w.r.t. τ_1 and τ_2. This means that each conjunct of ϕ is false w.r.t. τ_1 and τ_2. Thus, for every $M_1 \in AS(P_1)$ there exists $M_2 \in AS(P_2')$ such that $sat \notin M_2$, with $P_2' = P_2 \cup fix_{P_1}(M_1)$.

This means that $\mathcal{C}(P_2', M_2, 1) = 0$, and so, $OptAS(P_2') = \{M_2 \in AS(P_2') \mid sat \notin M_2\}$. Thus, there exists $M_2 \in OptAS(P_2')$, $C \cup \mathit{fix}_{P_2'}(M_2)$ is incoherent and so Π is incoherent.

At this point, it is easy to see that we can trivially add in the program P_1 a weak constraint of the form $\leftarrow_w a\ [1@1]$ where a is a fresh atom not appearing anywhere else without affecting the optimal answer set of P_1. Thus, this hardness holds both if P_1 is plain or not.

(Membership) Let Π be an ASP$^\omega$(Q) program of the form $\exists P_1 \exists P_2 : C$ where P_2 is not plain. By applying Algorithm 1 on Π, no matter if P_1 contains weak constraints or not, we obtain an existential ASP(Q) program Π', made of two alternating quantifiers, such that Π is coherent if and only if Π' is coherent. Since verifying the coherence of Π' is in Σ_2^P (Amendola et al. 2019)-complete then verifying the coherence of Π is also in Σ_2^P. □

The proof provides insights into this phenomenon. Indeed, the second quantifier, the one over optimal answer sets, "hides" a universal quantifier.

Theorem 6

Deciding coherence of uniform existential ASP$^\omega$(Q) programs with at most two quantifiers (i.e. $n \leq 2$) such that only P_1 contains weak constraints (i.e. $\mathcal{W}(P_1) \neq \emptyset$ and (if $n = 2$) $\mathcal{W}(P_2) = \emptyset$) is Δ_2^P-complete.

Proof

From Lemma 4 we observe that each program $\Pi = \exists P_1 \exists P_2 : C$, where P_1 is not plain and P_2 is plain, can be transformed into an ASP$^\omega$(Q) program $\Pi' = col_2(\Pi)$ such that Π is coherent if and only if Π' is coherent. Thus it is sufficient to prove that the statement holds for $n = 1$.

(Hardness) Given a program P, it is known that the task of verifying that an atom $a \in B_P$ appears in some optimal answer sets is Δ_2^P-complete, for a normal program with weak constraints (Buccafurri et al. 2000). Given a normal program P with weak constraints and an atom $a \in B_P$, we can construct an ASP$^\omega$(Q) program Π of the form $\exists P : \{\leftarrow {\sim}a\}$. According to the semantics of ASP$^\omega$(Q), Π is coherent if there exists $M \in OptAS(P)$ such that $C \cup \mathit{fix}_P(M)$ is coherent. By construction, $C \cup \mathit{fix}_P(M)$ if and only if $a \in M$. Thus, $M \in QAS(\Pi)$ (i.e. Π is coherent) if and only if $M \in OptAS(P)$ and $a \in M$.

(Membership) Given an ASP$^\omega$(Q) program Π of the form $\exists P : C$, we can construct an ASP program P^* such that Π is coherent if and only if P^* admits an optimal answer set M such that $\mathcal{C}(P^*, M, 1) = 0$. It is known that the task of verifying the existence of an optimal answer set whose cost is c at level l is Δ_2^P-complete (Amendola et al. 2024).

Let $\Pi = \exists P : C$, we construct (1) the program P' obtained by uniformly increasing the level of weak constraints in P in such a way that the lowest level is 2; (2) the program C' by translating each strong constraint $r \in C$ is into a normal rule $unsat \leftarrow B_r$, where $unsat$ is a fresh atom not appearing anywhere else.

Let $P^* = P' \cup C' \cup \{\leftarrow_w unsat[1@1]\}$, from the stratified definition assumption we know that $\mathcal{H}(C') \cap at(P') = \emptyset$, and so, $U = at(P)$ is a splitting set for P^* and so each $M \in AS(P^*)$ is of the form $M_1 \cup M_2$ where $M_1 \in AS(bot_U(P^*))$ and $M_2 \in AS(e_U(P^* \setminus bot_U(P^*), M_1))$. In particular, $bot_U(P^*) = P'$ and $P^* \setminus bot_U(P^*) = C'$, and so $M_1 \in AS(P')$ and $M_2 \in AS(e_U(C', M_1))$. Thus, from Lemma 1, $M \in AS(C' \cup \mathit{fix}_{P'}(M_1))$.

By construction, weak constraints in P' have the highest levels and so, M is an optimal answer set of P^* if and only if $M_1 \in OptAS(P')$, and $M \in AS(C' \cup \mathit{fix}_{P'}(M_1))$.

Moreover, since C is a stratified program with strong constraints then the incoherence of C can only be caused by strong constraint violations, that are encoded as normal rules in C' defining the fresh atom *unsat*. Thus, $C' \cup \mathit{fix}_{P'}(M_1)$ admits always a unique answer M. In particular, if $unsat \in M$ then some strong constraints in C are violated and so $C \cup \mathit{fix}_P(M_1)$ is incoherent, otherwise, no strong constraints in C are violated and so, $C \cup \mathit{fix}_P(M_1)$ is coherent.

By construction, P^* contains the weak constraint $\leftarrow_w unsat[1@1]$, and so, if there exists $M \in OptAS(P^*)$ such that $unsat \in M$ then $\mathcal{C}(P^*, M, 1) = 1$ and does not exist $M' \in OptAS(P^*)$ such that $unsat \notin M$ with $\mathcal{C}(P^*, M', 1) = 0$.

Thus, if there exists $M \in OptAS(P^*)$ such that $unsat \notin M$ then P^* admits an optimal answer set that costs 0 at level 1. Accordingly, since $unsat \notin M$ then every constraint in C is satisfied, and so Π is coherent. Conversely, if there exists $M \in OptAS(P^*)$ such that $unsat \in M$ then each $M' \in OptAS(P^*)$ contains *unsat* then P^* does not admit an optimal answer set that costs 0 at level 1. Accordingly, since *unsat* appears in every M' then at least one constraint in C is violated, and so Π is incoherent. □

Proposition 4 (*Third completeness results*)
Deciding coherence of plain uniform ASP$^{\omega}$(Q) programs with 2 quantifiers is (i) NP-complete for existential programs; and (i) coNP-complete for universal programs.

The result follows trivially from Lemma 3, once we observe that one application of col_1 builds an equi-coherent program with one quantifier.

Finally, the suitability of ASP$^{\omega}$(Q) for modeling optimization problems is witnessed by the following.

Lemma 8 (*Krentel (1992)*)
Let $X_1, \ldots, X_n$ be disjoint sets of propositional variables and ϕ be a propositional formulas over $X_1, \ldots, X_n$. Given a pair of truth assignments τ_1, τ_2 over a set of variables $X = x_1, \ldots, x_m$, we say that τ_1 is lexicographically greater that τ_2 if $\tau_1(x_i) = \top$ and $\tau_2(x_i) = \bot$ with $1 \leq i \leq m$ being the smallest index for which $\tau_1(x_i) \neq \tau_2(x_i)$. Let Φ be a QBF formula of the form $\forall X_2 \exists X_3 \ldots \mathcal{Q} X_n\ \phi$, where each $Q \in \{\exists, \forall\}$, and ϕ is a formula in 3-DNF if n is even, otherwise it is in 3-CNF, and $X_1 = \{x_1, \ldots, x_m\}$. Deciding whether the lexicographically minimum truth assignment τ of variables in X_1, such that $\forall X_2 \exists X_3 \ldots \mathcal{Q} X_n\ \phi_\tau$ is satisfied (assuming such τ exists), satisfies the condition $\tau(x_m) = \top$ is a Δ^P_{n+1}-complete problem.

Theorem 7 (*Fourth completeness results*)
Deciding whether an atom a belongs to an optimal quantified answer set of a plain alternating existential ASP$^{\omega}$(Q) program with n quantifiers is Δ^P_{n+1}-complete.

Proof
(Hardness) Starting from the Δ^P_{n+1}-complete problem introduced by Lemma 8, we can construct a plain alternating ASP$^{\omega}$(Q) program with n quantifier Π such that an atom, namely x_m, appears in some optimal quantified answer set of Π if and only if the answer to the problem is “yes”.

Without loss of generality we assume that n is even, ϕ is propositional formula in 3-DNF, and $X_1 = \{x_1, \ldots, x_m\}$.

For simplicity, we introduce some set of rules that will be used in the construction of Π. More precisely, $sat(\phi)$ denotes the set of rules of the form $sat_\phi \leftarrow l_1^i, l_2^i, l_3^i$, where $D_i = l_1^i \wedge l_2^i \wedge l_3^i$ is a conjunct in ϕ; whereas for a set of variables $Z = \{z_1, \ldots, z_k\}$, $choice(Z)$ denotes the program made of a single choice rule of the form $\{z_1; \ldots; z_k\} \leftarrow$.

We are now ready to construct the program Π of the form $\exists P_1 \forall P_2 \ldots \forall P_n : C : C^w$, where $P_i = choice(X_i)$, for each $1 \leq i \leq n$, and the programs C and C^w are of the form:

$$C = \left\{ \begin{array}{rl} & sat(\phi) \\ \leftarrow & \sim sat_\phi \end{array} \right\} \qquad\qquad C^w = \{ \leftarrow_w x_i \; [1@m-i+1] \quad \forall 1 \leq i \leq m \}$$

Intuitively, the program P_1 is used to guess a possible assignment τ over variables in X_1, for which we want to verify the satisfiability of the QBF formula $\Phi : \forall X_2 \exists X_3 \ldots \forall X_n \; \phi_\tau$. The following subprograms P_i, with $2 \leq i \leq n$, precisely match the quantifier alternation of Φ and are used for guessing possible truth assignments for variables in X_i. Once the final constraint program C is reached, we can evaluate the 3-DNF formula ϕ according to the truth assignments guessed by previous subprograms. The rules in $sat(\phi)$ will derive the atom sat if there is at least one conjunct in ϕ that is satisfied. Finally the last constraint in C impose that at least one conjunct must be satisfied.

Thus, there exists a quantified answer set of Π if and only if there exists a assignment of variables in X_1 such that Φ is satisfiable. Since the program C^w contains the set of weak constraints of the form $\leftarrow_w x_i \; [1@m-i+1]$ for each $i \in [1, \ldots, m]$ then the cost of each quantified answer set is given by the true atoms in the guessed τ. Thus, by assigning the highest priority to the atom x_1 (i.e. $m-1+1 = m$) and the lowest priority to x_m (i.e. $m-m+1 = 1$) we can simulate the lexicographical order described above. In conclusion, the optimal quantified answer set Π corresponds to the lexicographically minimum truth assignment τ, such that Φ is coherent. By construction x_m is derived if and only if $\tau(x_m)$ is true, and so the thesis follows.

(Membership) According to Theorem 3 of Amendola et al. (2019), we know that the coherence of an existential plain alternating program with n quantifiers falls within the complexity class Σ_n^P-complete. By following similar observations employed in the proofs by Buccafurri et al. (2000); Simons et al. (2002) an optimal solution can be obtained, by implementing a binary search on the value of k, with a logarithmic number of calls to an oracle in Σ_n^P (checking that no better solution than current exists). A final call to the oracle can ensure the existence of an optimal solution containing a. Since k can be exponential w.r.t. the input size (Buccafurri et al. (2000); Simons et al. (2002)) the thesis follows. □

Theorem 8 (*Fifth completeness results*)
Deciding whether an atom a belongs to an optimal quantified answer set of a plain alternating existential ASP$^\omega$(Q) program with n quantifiers is Θ_{n+1}^P-complete if there is only one level and all the weights are the same.

Proof
(Hardness) Let a QBF formula Φ be an expression of the form $\mathcal{Q}_1 X_1 \ldots \mathcal{Q}_n X_n \phi$, where $X_1, \ldots, X_n$ are disjoint sets of propositional variables, $\mathcal{Q}_i \in \{\exists, \forall\}$ for all $1 \leq i \leq n$, $\mathcal{Q}_i \neq \mathcal{Q}_{i+1}$ for all $1 \leq i < n$, and ϕ is a 3-DNF formula over variables in $X_1, X_2, \ldots, X_n$

of the form $D_1 \vee \ldots \vee D_n$, where each conjunct $D_i = l_1^i \wedge l_2^i \wedge l_3^i$, with $1 \leq i \leq n$. A k-existential QBF formula Φ is a QBF formula where $n = k$ and $\mathcal{Q}_1 = \exists$.

Given a sequence of m k-existential QBF formulas $\Phi_1, \ldots, \Phi_m$, with k being even and greater than or equal to 2, and such that if Φ_j is unsatisfiable then also Φ_{j+1} is unsatisfiable, where $1 \leq j < m$, deciding whether $v(\Phi_1, \ldots, \Phi_m) = max\{j \mid 1 \leq j \leq m \wedge \Phi_j \text{ is satisfiable}\}$ is odd is Θ_{k+1}-complete (Buccafurri et al. 2000).

The above problem can be encoded into an ASP$^\omega$(Q) program Π such that a literal, namely *odd*, appears in some optimal quantified answer set of Π if and only if $v(\Phi_1, \ldots, \Phi_m)$ is odd. For simplicity, we introduce notation for some sets of rules that will be used in the construction of Π. More precisely, given a QBF formula Φ, $sat(\Phi)$ denotes the set of rules of the form $sat_\Phi \leftarrow l_1^i, l_2^i, l_3^i$, where $D_i = l_1^i \wedge l_2^i \wedge l_3^i$ is a conjunct in ϕ; whereas for a set of variables $X_i = \{x_1^i, \ldots, x_n^i\}$ in Φ, and an atom a, $choice(X_i, a)$ denotes the choice rule $\{x_1^i; \ldots; x_n^i\} \leftarrow a$. We are now ready to construct the program Π.

First of all, we observe that all the formulas $\Phi_1, \ldots, \Phi_m$ have the same alternation of quantifiers. Thus, there is a one-to-one correspondence between the quantifiers in the QBF formulas and those in Π. Let Π be of the form $\Box_1 P_1 \Box_2 P_2 \ldots \Box_k P_k : C : C^w$ where $\Box_i = \exists$ if $\mathcal{Q}_i = \exists$ in a formula Φ_j, otherwise $\Box_i = \forall$. The program P_1 is of the form

$$P_1 = \left\{ \begin{array}{rcll} \{solve(1); \ldots; solve(m)\} = 1 & \leftarrow & & \\ unsolved(i) & \leftarrow & solve(j) & \forall\, j, i \in [1, \ldots, m] s.t.\ i > j \\ odd & \leftarrow & solve(j) & \forall j \in [1, \ldots, m] s.t.\ j \text{ is odd} \\ choice(X_1^j, solve(j)) & & & \forall 1 \leq j \leq m \end{array} \right\},$$

while, for each $2 \leq i \leq k$, the program P_i is of the form

$$P_i = \{\ choice(X_i^j, solve(j)) \quad \forall 1 \leq j \leq m\ \},$$

where each X_i^j denotes the set of variables appearing in the scope of the i-th quantifier of the j-th QBF formula Φ_j. Finally, the programs C and C^w are of the form

$$C = \left\{ \begin{array}{rll} & sat(\Phi_j) & \forall 1 \leq j \leq m \\ \leftarrow & solve(j),\ {\sim} sat_{\Phi_j} & \forall 1 \leq j \leq m \end{array} \right\} C^w = \{\ \leftarrow_w unsolved(i)\ [1@1, i] \quad \forall 1 \leq i \leq m\ \}.$$

Intuitively, the first choice rule in P_1 is used to guess one QBF formula, say Φ_j, among the m input ones, for which we want to verify the satisfiability. The guessed formula is encoded with the unary predicate *solve*, whereas, all the following formulas Φ_i, with $i > j$, are marked as unsolved by means of the unary predicate *unsolved*.

Then, P_1 contains different rules of the form $odd \leftarrow solve(j)$ for each odd index j in $[1, m]$. Thus the literal *odd* is derived whenever a QBF formula Φ_j in the sequence $\Phi_1, \ldots, \Phi_m$ is selected (i.e. $solve(j)$ is true) and j is odd. The remaining part of P_1 shares the same working principle of the following subprograms P_i, with $i \geq 2$. More precisely, for each QBF formula Φ_j in the sequence $\Phi_1, \ldots, \Phi_m$, they contain a choice rule over the set of variables quantified by the i-th quantifier of Φ_j. Note that the atom $solve(j)$ in the body of these choice rules guarantees that only one gets activated, and so the activated choice rule guesses a truth assignment for the variables in the i-th quantifier of Φ_j. Similarly, the constraint program C contains, for each QBF formula Φ_j in the sequence $\Phi_1, \ldots, \Phi_m$, (i) a set of rules that derives an atom sat_{ϕ_j} whenever the truth assignment guessed by the previous subprograms satisfies ϕ_j, and (ii) a strong constraint imposing that is not possible that we selected the formula Φ_j (i.e. $solve(j)$ is true) and ϕ_j is violated (i.e. sat_{Φ_j} is false). Thus, there exists a quantified answer set of Π if and only if there exists a formula Φ_j in the sequence $\Phi_1, \ldots, \Phi_m$ such that Φ_j is satisfiable. Since the program C^w contains the set of weak constraints of the form $\leftarrow_w$ $unsolved(j)$ $[1@1, j]$ for each $j \in [1, \ldots, m]$, the cost of each quantified answer set is given by the index j of the selected formula. Thus, by minimizing the number of unsolved formulas we are maximizing the index of the satisfiable formula Φ_j. Thus, an optimal quantified answer set corresponds to a witness of coherence for a formula Φ_j, s.t. for each $\Phi_{j'}$, with $j' > j$, $\Phi_{j'}$ is unsatisfiable. By construction *odd* is derived whenever j is odd and so the hardness follows.

(Membership) According to Theorem 3 of Amendola et al. (2019), we know that the coherence of an existential plain alternating program with n quantifiers falls within the complexity class Σ_n^P-complete. By following an observation employed in the proofs by Buccafurri et al. (2000), the cost of an optimal solution can be obtained by binary search that terminates in a logarithmic, in the value of the maximum cost, number of calls to an oracle in Σ_n^P that checks whether a quantified answer set with a lower cost with respect to the current estimate of the optimum exists. Once the cost of an optimal solution is determined, one more call to the oracle (for an appropriately modified instance), allows one to decide the existence of an optimal solution containing a. Since each weak constraint has the same weight and the same level, then we can consider as the maximum cost the number of weak constraint violations. Thus, the number of oracle calls is at most logarithmic in the size of the problem and the membership follows. □

References

Amendola, G., Berei, T., Mazzotta, G., and Ricca, F. 2024. Unit testing in ASP revisited: Language and test-driven development environment. *CoRR*, *abs/2401.02153*.

Amendola, G., Ricca, F., and Truszczynski, M. 2019. Beyond NP: quantifying over answer sets. *Theory Pract. Log. Program.*, *19*, 5-6, 705–721.

Buccafurri, F., Leone, N., and Rullo, P. 2000. Enhancing disjunctive datalog by constraints. *TKDE*, *12*, 5, 845–860.

Eiter, T. and Gottlob, G. 1995. The complexity of logic-based abduction. *J. ACM*, *42*, 1, 3–42.

Fandinno, J., Laferrière, F., Romero, J., Schaub, T., and Son, T. C. 2021. Planning with incomplete information in quantified answer set programming. *TPLP*, *21*, 5, 663–679.

Garey, M. R. and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman.

Krentel, M. W. 1992. Generalizations of opt P to the polynomial hierarchy. *Theor. Comput. Sci.*, *97*, 2, 183–198.

Lifschitz, V. and Turner, H. Splitting a logic program. In *Proc. of ICLP 1994* 1994, pp. 23–37.

Papadimitriou, C. H. 1994. *Computational complexity.* Addison-Wesley.

Simons, P., Niemelä, I., and Soininen, T. 2002. Extending and implementing the stable model semantics. *Artif. Intell.*, *138*, 1-2, 181–234.

Stockmeyer, L. J. 1976. The polynomial-time hierarchy. *Theor. Comput. Sci.*, *3*, 1, 1–22.

Wagner, K. W. 1990. Bounded query classes. *SIAM J. Comput.*, *19*, 5, 833–846.