Communicating ASP and the Polynomial Hierarchy

Kim Bauters[1,⋆], Steven Schockaert[1,⋆⋆], Dirk Vermeir[2], and Martine De Cock[1]

[1] Department of Applied Mathematics and Computer Science
Universiteit Gent, Krijgslaan 281, 9000 Gent, Belgium
{kim.bauters,steven.schockaert,martine.decock}@ugent.be
[2] Department of Computer Science
Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussel, Belgium
dvermeir@vub.ac.be

Abstract. Communicating answer set programming is a framework to represent and reason about the combined knowledge of multiple agents using the idea of stable models. The semantics and expressiveness of this framework crucially depends on the nature of the communication mechanism that is adopted. The communication mechanism we introduce in this paper allows us to focus on a sequence of programs, where each program in the sequence may successively eliminate some of the remaining models. The underlying intuition is that of leaders and followers: each agent's decisions are limited by what its leaders have previously decided. We show that extending answer set programs in this way allows us to capture the entire polynomial hierarchy.

1 Introduction

Communicating answer set programming is an extension of answer set programming (ASP) in which a number of logic programs, with their own knowledge and reasoning capabilities, can communicate and cooperate with each other to solve the problem at hand. A number of different flavors of Communicating ASP have already been proposed in the literature (*e.g.* [1,8,21]). Communicating ASP is also closely related to the research on multi-context systems, where a group of agents can cooperate to find the solution of a global problem [4,20]. We start off with an introductory example.

Example 1. An employee ('E') needs a new printer ('P'). She has a few choices (loud or silent, stylish or dull), preferring silent and stylish. Her boss ('B') does not want an expensive printer, *i.e.* one that is both silent and stylish. We can

⋆ Funded by a joint Research Foundation-Flanders (FWO) project.
⋆⋆ Postdoctoral fellow of the Research Foundation-Flanders (FWO).

consider the communicating program $\mathcal{P} = \{E, B\}$ with:

$$\begin{aligned} P\!:\!\mathit{stylish} &\leftarrow \mathit{not}\ P\!:\!\mathit{dull} & P\!:\!\mathit{dull} &\leftarrow \mathit{not}\ P\!:\!\mathit{stylish} & (1)\\ P\!:\!\mathit{silent} &\leftarrow \mathit{not}\ P\!:\!\mathit{loud} & P\!:\!\mathit{loud} &\leftarrow \mathit{not}\ P\!:\!\mathit{silent} & (2)\\ E\!:\!\mathit{undesired} &\leftarrow P\!:\!\mathit{dull} & E\!:\!\mathit{undesired} &\leftarrow P\!:\!\mathit{loud} & (3)\\ B\!:\!\mathit{expensive} &\leftarrow P\!:\!\mathit{stylish}, P\!:\!\mathit{silent}. & & & (4) \end{aligned}$$

Intuitively, the rule '$B\!:\!\mathit{expensive} \leftarrow P\!:\!\mathit{stylish}, P\!:\!\mathit{silent}$' expresses that the agent B believes that the printer is '*expensive*' when according to agent P it is '*stylish*' and '*silent*'. The rules in (1) and (2) encode the four possible printers, the rules in (3) and (4) encode the inclinations of the employee and boss, respectively. The answer sets of this program, *i.e.* those with global minimality, are [1]

$$\begin{aligned} M_1 &= \{P\!:\!\mathit{sty}, P\!:\!\mathit{silent}, B\!:\!\mathit{exp}\} & M_2 &= \{P\!:\!\mathit{sty}, P\!:\!\mathit{loud}, E\!:\!\mathit{und}\}\\ M_3 &= \{P\!:\!\mathit{dull}, P\!:\!\mathit{loud}, E\!:\!\mathit{und}\} & M_4 &= \{P\!:\!\mathit{dull}, P\!:\!\mathit{silent}, E\!:\!\mathit{und}\} \end{aligned}$$

where for compactness we write $P\!:\!\mathit{sty}, E\!:\!\mathit{und}$ and $B\!:\!\mathit{exp}$ instead of $P\!:\!\mathit{stylish}$, $E\!:\!\mathit{undesired}$ and $B\!:\!\mathit{expensive}$, respectively. The answer sets with minimality for the agent B are M_2, M_3 and M_4, *i.e.* the answer sets that do not contain $B\!:\!\mathit{expensive}$. The only answer set with minimality for agent E is M_1, *i.e.* the one that does not contain $E\!:\!\mathit{undesired}$. Hence when we determine local minimality for communicating ASP, the order in which we determine the local minimality is important and induces a preference over the agents, *i.e.* it makes some agents more important than others. In this example, if the boss comes first, the employee no longer has the choice to pick M_1. This leaves her with the choice of either a dull or a loud printer, among which she has no preferences.

Answer set semantics is based on the idea of stable minimal models. When dealing with agents that can communicate, it becomes unclear how we should interpret the notion of minimality as will become clear later on. One option is to assume global minimality, *i.e.* we minimize over the conclusions of all the agents in the network. Another option is to assume minimality on the level of a single agent. Since it is not always possible to find a model that is minimal for all individual agents, the order in which we minimize over the agents matters, as the preceding example illustrates.

In this paper we introduce the notion of multi-focused answer sets of communicating ASP programs, which allow us to successively focus (*i.e.* minimize) on different agents. We study the resulting expressiveness and hence complete the picture sketched in [1] on the effect that adding communication has on the expressiveness of ASP. In [1], it was shown that the ability to communicate allows for simple programs (programs without negation-as-failure, which alone are only capable of expressing problems in P) to simulate normal programs (which do have negation-as-failure and are capable of expressing problems in NP). Furthermore, [1] introduced a new communication mechanism where one can "focus" on a single agent, allowing to express problems in Σ_2^{P}. In this paper, we go a step further by introducing multi-focused answer sets. Multi-focused answer sets allow us to focus successively on a number of agents instead of focusing on just one agent, and is therefore a generalization of the idea of focusing. As it turns out, using multi-focused answer set programs it is possible to express any problem in PSPACE. This means in particular that communicating ASP could be used to solve problems that are above the second level of the polynomial hierarchy, such as some forms of abductive reasoning [10] as well as PSPACE-complete problems such as STRIPS planning [6].

The remainder of this paper is organized as follows. In Section 2 we give the necessary background on answer set programming. In Section 3, we recall the syntax and semantics of communicating ASP. In Section 4 we introduce a generalization of focused answer sets [1], capable of expressing any Σ_n^{P} problem. We finish with Section 5 and Section 6 where we discuss related work and present our conclusion.

2 Background on Answer Set Programming

We first recall the basic concepts and results from ASP that are used in this paper. To define ASP programs, we start from a countable set of atoms and we define a *literal* l as an atom a or its classical negation $\neg a$. If L is a set of literals, we use $\neg L$ to denote the set $\{\neg l \mid l \in L\}$ where, by definition, $\neg\neg a = a$. A set of literals L is consistent if $L \cap \neg L = \emptyset$. An *extended literal* is either a literal or a literal preceded by *not* which we call the negation-as-failure operator. Intuitively we say that *not* l is true when we have no proof to support l. For a set of literals L, we use $not(L)$ to denote the set $\{not\ l \mid l \in L\}$.

A *normal rule* is an expression of the form $l \leftarrow (\alpha \cup not(\beta))$ with 'l' a literal called the head of the rule and $(\alpha \cup not(\beta))$ (interpreted as a conjunction) the body of the rule with α and β sets of literals. When the body is empty, the rule is called a *fact*. When the head is empty, the rule is called a *constraint*. In this paper, we do not consider constraints as they can readily be simulated[1]. A *normal program* P is a finite set of normal rules. The *Herbrand base* $\mathcal{B}_P$ of P is the set of atoms appearing in program P. A (partial) *interpretation* I of P is any consistent set of literals $L \subseteq (\mathcal{B}_P \cup \neg\mathcal{B}_P)$. I is total iff $I \cup \neg I = \mathcal{B}_P \cup \neg\mathcal{B}_P$. A *simple rule* is a normal rule without negation-as-failure in the body. A *simple program* P is a finite set of simple rules.

Answer sets are defined using the *immediate consequence operator* T_P for a simple program P *w.r.t.* an interpretation I as

$$T_P(I) = I \cup \{l \mid ((l \leftarrow \alpha) \in P) \wedge (\alpha \subseteq I)\}\,. \tag{5}$$

We use $P^\star$ to denote the fixpoint which is obtained by repeatedly applying T_P starting from the empty interpretation, *i.e.* the least fixpoint of T_P *w.r.t.* set inclusion. The interpretation $P^\star$ is the minimal model of P and is called the *answer set* of the simple program P.

The *reduct* P^I of a normal program P *w.r.t.* the interpretation I is defined as $P^I = \{l \leftarrow \alpha \mid (l \leftarrow \alpha \cup not(\beta)) \in P, \beta \cap I = \emptyset\}$. It is easy to see that the reduct P^I is a simple program. We say that I is an answer set of the normal program P iff $\left(P^I\right)^\star = I$, *i.e.* if I is the answer set of the reduct P^I.

[1] The constraint $(\leftarrow body)$ is simulated by $(fail \leftarrow not\ fail, body)$ with $fail$ a fresh atom.

3 Communicating Programs

Communication between ASP programs is based on a new kind of literal '$Q{:}l$' as in [1,3,16,20], where the underlying intuition is that of a function call or, in terms of agents, asking questions to other agents. If the literal l is not in the answer set of program Q then $Q{:}l$ is false; otherwise $Q{:}l$ is true. We present the semantics from [1] which are closely related to the minimal semantics in [3] and the semantics in [5].

Let $\mathcal{P}$ be a finite set of program names. A *$\mathcal{P}$-situated literal* is an expression of the form $Q{:}l$ with $Q \in \mathcal{P}$ and l a literal. A $\mathcal{P}$-situated literal $Q{:}l$ is called *Q-local*. For a set of $\mathcal{P}$-situated literals X and $Q \in \mathcal{P}$, we use $X_{\downarrow Q}$ to denote $\{l \mid Q{:}l \in X\}$, *i.e.* the projection of X on Q. An *extended $\mathcal{P}$-situated literal* is either a $\mathcal{P}$-situated literal or a $\mathcal{P}$-situated literal preceded by *not*. For a set of $\mathcal{P}$-situated literals X, we use $not(X)$ to denote the set $\{not\ Q{:}l \mid Q{:}l \in X\}$.

A *$\mathcal{P}$-situated normal rule* is an expression of the form $Q{:}l \leftarrow (\alpha \cup not(\beta))$ where $Q{:}l$ is called the head of the rule, and $\alpha \cup not(\beta)$ is called the body of the rule with α and β sets of $\mathcal{P}$-situated literals. A $\mathcal{P}$-situated normal rule $Q{:}l \leftarrow (\alpha \cup not\ (\beta))$ is called Q-local. A *$\mathcal{P}$-component normal program Q* is a finite set of Q-local $\mathcal{P}$-situated normal rules. Henceforth we shall use $\mathcal{P}$ both to denote the set of program names and to denote the set of actual $\mathcal{P}$-component normal programs. A *communicating normal program $\mathcal{P}$* is then a finite set of $\mathcal{P}$-component normal programs. A *$\mathcal{P}$-situated simple rule* is an expression of the form $Q{:}l \leftarrow \alpha$, *i.e.* a $\mathcal{P}$-situated normal rule without negation-as-failure in the body. A *$\mathcal{P}$-component simple program Q* is a finite set of Q-local $\mathcal{P}$-situated simple rules. A *communicating simple program $\mathcal{P}$* is a finite set of $\mathcal{P}$-component simple programs.

In the remainder of this paper we drop the $\mathcal{P}$-prefix whenever the set $\mathcal{P}$ is clear from the context. Whenever the name of the component normal program Q is clear, we write l instead of $Q{:}l$ for Q-local situated literals. Note that a communicating normal (resp. simple) program with only one component program thus trivially corresponds to a normal (resp. simple) program.

Similar as for a classical program, we can define the *Herbrand base* of a component program Q as the set of atoms occurring in Q-local situated literals in Q, which we denote as $\mathcal{B}_Q$. We then define $\mathcal{B}_{\mathcal{P}} = \left\{Q{:}a \mid Q \in \mathcal{P} \text{ and } a \in \bigcup_{R \in \mathcal{P}} \mathcal{B}_R\right\}$ as the Herbrand base of the communicating program $\mathcal{P}$.

Example 2. Consider the communicating simple program $\mathcal{P} = \{Q, R\}$ with the following situated rules:

$$Q{:}a \leftarrow R{:}a \qquad\qquad Q{:}b \leftarrow \qquad\qquad R{:}a \leftarrow Q{:}a.$$

$Q{:}a$, $Q{:}b$ and $R{:}a$ are situated literals. The situated simple rules $Q{:}a \leftarrow R{:}a$ and $Q{:}b \leftarrow$ are Q-local since we have $Q{:}a$ and $Q{:}b$ in the heads of these rules. The situated simple rule $R{:}a \leftarrow Q{:}a$ is R-local. Hence $Q = \{a \leftarrow R{:}a, b \leftarrow\}$ and $R = \{a \leftarrow Q{:}a\}$. Furthermore, we have that $\mathcal{B}_Q = \{a, b\}$, $\mathcal{B}_R = \{a\}$ and $\mathcal{B}_{\mathcal{P}} = \{Q{:}a, Q{:}b, R{:}a, R{:}b\}$.

We say that a (partial) interpretation I of a communicating program $\mathcal{P}$ is any consistent subset $I \subseteq (\mathcal{B}_\mathcal{P} \cup \neg\mathcal{B}_\mathcal{P})$. Given an interpretation I of a communicating normal program $\mathcal{P}$, the reduct Q^I for $Q \in \mathcal{P}$ is the component simple program obtained by deleting

- each rule with an extended situated literal '*not* $R{:}l$' such that $R{:}l \in I$;
- each remaining extended situated literal of the form '*not* $R{:}l$';
- each rule with a situated literal '$R{:}l$' that is not Q-local such that $R{:}l \notin I$;
- each remaining situated literal $R{:}l$ that is not Q-local.

The underlying intuition of the reduct is clear. Analogous to the definition of the reduct for normal programs [15], the reduct of a communicating normal program defines a way to reduce a program relative to some guess I. The reduct of a communicating normal program is a communicating simple program that only contains component simple programs Q with Q-local situated literals. That is, each component simple program Q corresponds to a classical simple program.

Definition 1. *We say that an interpretation I of a communicating normal program $\mathcal{P}$ is an* answer set *of $\mathcal{P}$ if and only if* $\forall Q \in \mathcal{P} \cdot (Q{:}I_{\downarrow Q}) = \left(Q^I\right)^\star$.

In other words: an interpretation I is an answer set of a communicating normal program $\mathcal{P}$ if and only if for every component normal program Q we have that the projection of I on Q is an answer set of the component normal program Q under the classical definition.

Example 3. Let us once again consider the communicating simple program $\mathcal{P} = \{Q, R\}$ from Example 2. Given the interpretation $I = \{Q{:}a, Q{:}b, R{:}a\}$ we find that $Q^I = \{a \leftarrow, b \leftarrow\}$ and $R^I = \{a \leftarrow\}$. We can easily treat Q^I and R^I separately since they now correspond to classical programs. It then readily follows that $Q{:}I_{\downarrow Q} = \left(Q^I\right)^\star$ and $R{:}I_{\downarrow R} = \left(R^I\right)^\star$, hence the interpretation I is an answer set of $\mathcal{P}$. In total the communicating simple program $\mathcal{P}$ has two answer sets, namely $\{Q{:}b\}$ and $\{Q{:}a, Q{:}b, R{:}a\}$.

Proposition 1. *[1] Let $\mathcal{P}$ be a communicating simple program. Determining whether a given situated literal $Q{:}l$ occurs in any answer set of $\mathcal{P}$ (*i.e.* brave reasoning) is in* NP*. Determining whether a literal is true in all the answer sets of $\mathcal{P}$ (*i.e.* cautious reasoning) is in* coNP*. Furthermore, the same complexity results hold when $\mathcal{P}$ is a communicating normal program.*

4 Multi-focused Answer Sets

We extend the semantics of communicating programs in such a way that it becomes possible to focus on a sequence of component programs. As such, we can indicate that we are only interested in those answer sets that are successively minimal with respect to each respective component program. The underlying intuition is that of leaders and followers, where the decisions that an agent can make are limited by what its leaders have previously decided.

Definition 2. *Let $\mathcal{P}$ be a communicating program and $\{Q_1, \ldots, Q_n\} \subseteq \mathcal{P}$ a set of component programs. A $(Q_1, \ldots, Q_n)$-*focused answer set *of $\mathcal{P}$ is defined as:*

$$\begin{aligned}&M \textit{ is a } (Q_1, \ldots, Q_{n-1})\textit{-focused answer set of } \mathcal{P} \textit{ and}\\ &\quad \textit{for each } (Q_1, \ldots, Q_{n-1})\textit{-focused answer set } M' \textit{ of } \mathcal{P}\\ &\quad \textit{we do not have that } M'_{\downarrow Q_n} \subset M_{\downarrow Q_n}\end{aligned}$$

where a ()-focused answer set of $\mathcal{P}$ is any answer set of $\mathcal{P}$.

In other words, we say that M is a $(Q_1, \ldots, Q_n)$-focused answer set of $\mathcal{P}$ if and only if M is minimal among all $(Q_1, \ldots, Q_{n-1})$-focused answer sets *w.r.t.* the projection on Q_n.

Example 4. Consider the communicating program $\mathcal{P}_4 = \{Q, R, S\}$ with the rules

$$\begin{array}{lll} Q\!:\!a \leftarrow & R\!:\!b \leftarrow S\!:\!c & S\!:\!a \leftarrow \\ Q\!:\!b \leftarrow not\ S\!:\!d & R\!:\!a \leftarrow S\!:\!c & S\!:\!c \leftarrow not\ S\!:\!d, not\ R\!:\!c \\ Q\!:\!c \leftarrow R\!:\!c & R\!:\!a \leftarrow S\!:\!d & S\!:\!c \leftarrow not\ S\!:\!c, not\ R\!:\!c \\ & R\!:\!c \leftarrow not\ R\!:\!a & \end{array}$$

The communicating program $\mathcal{P}_4$ has three answer sets, namely

$$\begin{aligned} M_1 &= Q\!:\!\{a, b, c\} \cup R\!:\!\{c\} \cup S\!:\!\{a\} \\ M_2 &= Q\!:\!\{a, b\} \cup R\!:\!\{a, b\} \cup S\!:\!\{a, c\} \\ M_3 &= Q\!:\!\{a\} \cup R\!:\!\{a\} \cup S\!:\!\{a, d\}. \end{aligned}$$

The only (R, S)-focused answer set of $\mathcal{P}_4$ is M_1. Indeed, since $\{a\} = (M_3)_{\downarrow R} \subset (M_2)_{\downarrow R} = \{a, b\}$ we find that M_2 is not a (R)-focused answer set. Furthermore $\{a\} = (M_1)_{\downarrow S} \subset (M_3)_{\downarrow S} = \{a, d\}$, hence M_3 is not an (R, S)-focused answer set.

We now show how the validity of quantified boolean formulas (QBF) can be checked using multi-focused answer sets of communicating ASP programs.

Definition 3. *Let $\phi = \exists X_1 \forall X_2 ... \Theta X_n \cdot p(X_1, X_2, \cdots X_n)$ be a QBF where $\Theta = \forall$ if n is even and $\Theta = \exists$ otherwise, and $p(X_1, X_2, \cdots X_n)$ is a formula of the form $\theta_1 \vee \ldots \vee \theta_m$ in disjunctive normal form over $X_1 \cup \ldots \cup X_n$ with X_i, $1 \leq i \leq n$, sets of variables and where each θ_t is a conjunction of propositional literals. We define Q_0 as follows:*

$$\begin{aligned} Q_0 = &\{x \leftarrow not\ \neg x, \neg x \leftarrow not\ x \mid x \in X_1 \cup \ldots \cup X_n\} && (6)\\ &\cup \{sat \leftarrow Q_0\!:\!\theta_t \mid \theta_t, 1 \leq t \leq m\} && (7)\\ &\cup \{\neg sat \leftarrow not\ sat\}\,. && (8) \end{aligned}$$

For $1 \leq j \leq n-1$ we define Q_j as follows:

$$\begin{aligned} Q_j = &\{x \leftarrow Q_0\!:\!x, \neg x \leftarrow Q_0\!:\!\neg x \mid x \in (X_1 \cup \ldots \cup X_{n-j})\} && (9)\\ &\cup \begin{cases} \{\neg sat \leftarrow Q_0\!:\!\neg sat\} & \textit{if } (n-j) \textit{ is even} \\ \{sat \leftarrow Q_0\!:\!sat\} & \textit{if } (n-j) \textit{ is odd.} \end{cases} && (10) \end{aligned}$$

The communicating normal program corresponding with ϕ is $\mathcal{P} = \{Q_0, ..., Q_{n-1}\}$.

For a QBF of the form $\phi = \forall X_1 \exists X_2 ... \Theta X_n \cdot p(X_1, X_2, \cdots X_n)$ where $\Theta = \exists$ if n is even and $\Theta = \forall$ otherwise and $p(X_1, X_2, \cdots X_n)$ once again a formula in disjunctive normal form, the simulation only changes slightly. Indeed, only the conditions in (10) are swapped.

Example 5. Given the QBF $\phi = \exists x \forall y \exists z \cdot (x \wedge y) \vee (\neg x \wedge y \wedge z) \vee (\neg x \wedge \neg y \wedge \neg z)$, the communicating program $\mathcal{P}$ corresponding with the QBF ϕ is defined as follows:

$$\begin{array}{rrr}
Q_0 : x \leftarrow not\ \neg x & Q_0 : y \leftarrow not\ \neg y & Q_0 : z \leftarrow not\ \neg z \\
Q_0 : \neg x \leftarrow not\ x & Q_0 : \neg y \leftarrow not\ y & Q_0 : \neg z \leftarrow not\ z \\
Q_0 : sat \leftarrow x, y & Q_0 : sat \leftarrow \neg x, y, z & Q_0 : sat \leftarrow \neg x, \neg y, \neg z \\
Q_0 : \neg sat \leftarrow not\ sat & &
\end{array}$$

$$\begin{array}{rrr}
Q_1 : x \leftarrow Q_0 : x & Q_1 : y \leftarrow Q_0 : y & \\
Q_1 : \neg x \leftarrow Q_0 : \neg x & Q_1 : \neg y \leftarrow Q_0 : \neg y & Q_1 : \neg sat \leftarrow Q_0 : \neg sat
\end{array}$$

$$\begin{array}{rrr}
Q_2 : x \leftarrow Q_0 : x & Q_2 : \neg x \leftarrow Q_0 : \neg x & Q_2 : sat \leftarrow Q_0 : sat
\end{array}$$

The communicating program in Example 5 can be used to determine whether the QBF ϕ is satisfiable. First, note that the rules in (6) generate all possible truth assignments of the variables, *i.e.* all possible propositional interpretations. The rules in (7) ensure that '*sat*' is true exactly for those interpretations that satisfy the formula $p(X_1, X_2, \ldots, X_n)$, *i.e.* we check, for this particular assignment of the variables, whether $p(X_1, \ldots, X_n)$ is satisfied.

Intuitively, the component programs $\{Q_1, \ldots, Q_{n-1}\}$ successively bind fewer and fewer variables. In particular, focusing on $Q_1, \ldots, Q_{n-1}$ allows us to consider the binding of the variables in $X_{n-1}, \ldots, X_1$, respectively. Depending on the rules from (10), focusing on Q_i allows us to verify that either some or all of the assignments of the variables in X_{n-j} make the formula $p(X_1, \ldots, X_n)$ satisfied, given the bindings that have already been determined by the preceding components. We now prove that the QBF ϕ is satisfiable iff $Q_0 : sat$ is true in some $(Q_1, \ldots, Q_{n-1})$-focused answer set.

Proposition 2. *Let ϕ and $\mathcal{P}$ be as in Definition 3. We have that a QBF ϕ of the form $\phi = \exists X_1 \forall X_2 ... \Theta X_n \cdot p(X_1, X_2, \cdots X_n)$ is satisfiable if and only if $Q_0 : sat$ is true in some $(Q_1, \ldots, Q_{n-1})$-focused answer set of $\mathcal{P}$. Furthermore, we have that a QBF ϕ of the form $\phi = \forall X_1 \exists X_2 ... \Theta X_n \cdot p(X_1, X_2, \cdots X_n)$ is satisfiable if and only if $Q_0 : sat$ is true in all $(Q_1, \ldots, Q_{n-1})$-focused answer sets of $\mathcal{P}$.*

Proof. We give a proof by induction. Assume we have a QBF ϕ_1 of the form $\exists X_1 \cdot p(X_1)$ with $\mathcal{P}_1 = \{Q_0\}$ the communicating normal program corresponding with ϕ_1 according to Definition 3. If the formula $p_1(X_1)$ of the QBF ϕ_1 is satisfiable then we know that there is a ()-focused answer set M of $\mathcal{P}_1$ such that $Q_0 : sat \in M$. Otherwise, we know that $Q_0 : sat \notin M$ for all ()-answer sets M of $\mathcal{P}_1$. Hence the induction hypothesis is valid for $n = 1$.

Assume the result holds for any QBF ϕ_{n-1} of the form $\exists X_1 \forall X_2 \ldots \Theta X_{n-1} \cdot p_n(X_1, X_2, \ldots, X_{n-1})$. We show in the induction step that it holds for any QBF ϕ_n of the form $\exists X_1 \forall X_2 \ldots \overline{\Theta} X_n \cdot p_{n-1}(X_1, X_2, \ldots, X_n)$. Let $\mathcal{P} = \{Q_0, \ldots, Q_{n-1}\}$ and $\mathcal{P}' = \left\{Q'_0, \ldots, Q'_{n-2}\right\}$ be the communicating normal programs that correspond with ϕ_n and ϕ_{n-1}, respectively. Note that the component programs $Q_2, \ldots, Q_{n-1}$ are defined in exactly the same way as the component programs $Q'_1, \ldots, Q'_{n-2}$, the only difference being the name of the component programs. What is of importance in the case of ϕ_n is therefore only the additional rules in Q_0 and the new component program Q_1. The additional rules in Q_0 merely generate the corresponding interpretations, where we now need to consider the possible interpretations of the variables from X_n as well. The rules in the new component program Q_1 ensure that $Q_1 : x \in M$ whenever $Q_0 : x \in M$ and $Q_1 : \neg x \in M$ whenever $Q_0 : \neg x \in M$ for every M an answer set of $\mathcal{P}$ and $x \in (X_1 \cup \ldots \cup X_{n-1})$. Depending on n being even or odd, we get two distinct cases:

- if n is even, then we have $(sat \leftarrow Q_0 : sat) \in Q_1$ and we know that the QBF ϕ_n has the form $\exists X_1 \forall X_2 \ldots \forall X_n \cdot p_n(X_1, X_2, \ldots, X_n)$. Let us consider what happens when we determine the (Q_1)-focused answer sets of $\mathcal{P}$. Due to the construction of Q_1, we know that $M'_{\downarrow Q_1} \subset M_{\downarrow Q_1}$ can only hold for two answer sets M' and M of $\mathcal{P}$ if M' and M correspond to identical interpretations of the variables in $X_1 \cup \ldots \cup X_{n-1}$. Furthermore, $M'_{\downarrow Q_1} \subset M_{\downarrow Q_1}$ is only possible if $Q_1 : sat \in M$ while $Q_1 : sat \notin M'$.

  Now note that given an interpretation of the variables in $X_1 \cup \ldots \cup X_{n-1}$, there is exactly one answer set for each choice of X_n. When we have M' with $Q_1 : sat \notin M'$ this implies that there is an interpretation such that, for some choice of X_n, this particular assignment of values of the QBF does not satisfy the QBF. Similarly, if we have M with $Q_1 : sat \in M$ then the QBF is satisfied for that particular choice of X_n. Determining (Q_1)-focused answer sets of $\mathcal{P}$ will eliminate M since $M'_{\downarrow Q_1} \subset M_{\downarrow Q_1}$. In other words, for identical interpretations of the variables in $X_1 \cup \ldots \cup X_{n-1}$, the answer set M' encodes a counterexample that shows that for these interpretations it does not hold that the QBF is satisfied for all choices of X_n. Focusing thus eliminates those answer sets that claim that the QBF is satisfiable for the variables in $X_1 \cup \ldots \cup X_{n-1}$. When we cannot find such $M'_{\downarrow Q_1} \subset M_{\downarrow Q_1}$ this is either because none of the interpretations satisfy the QBF or all of the interpretations satisfy the QBF. In both cases, there is no need to eliminate any answer sets. We thus effectively mimic the requirement that the QBF ϕ_n should hold for $\forall X_n$.
- if n is odd, then $(\neg sat \leftarrow Q_0 : \neg sat) \in Q_1$ and we know that the QBF ϕ_n has the form $\exists X_1 \forall X_2 \ldots \exists X_n \cdot p_n(X_1, X_2, \ldots, X_n)$. As before, we know that $M'_{\downarrow Q_1} \subset M_{\downarrow Q_1}$ can only hold for two answer sets M' and M of $\mathcal{P}$ if M' and M correspond to identical interpretations of the variables in $X_1 \cup \ldots \cup X_{n-1}$. However, this time $M'_{\downarrow Q_1} \subset M_{\downarrow Q_1}$ is only possible if $Q_1 : \neg sat \in M$ while $Q_1 : \neg sat \notin M'$.

  If we have M with $Q_1 : \neg sat \in M$ then the QBF is not satisfied for that particular choice of X_n, whereas when M' with $Q_1 : \neg sat \notin M'$ this implies that there is an interpretation such that, for some choice of X_n, this particular assignment of the variables does satisfy the QBF. Determining (Q_1)-focused answer sets of $\mathcal{P}$ will eliminate M since $M'_{\downarrow Q_1} \subset M_{\downarrow Q_1}$. For identical interpretations of the variables in $X_1 \cup \ldots \cup X_{n-1}$, the answer set M' encodes a counterexample that shows that for these interpretations there is some choice of X_n such that the QBF is satisfied. Focusing thus eliminates those answer sets that claim that the QBF is not satisfiable for the variables in $X_1 \cup \ldots \cup X_{n-1}$. When we cannot find such $M'_{\downarrow Q_1} \subset M_{\downarrow Q_1}$ this is either because none of the interpretations satisfy the QBF or all of the interpretations satisfy the QBF. In both cases, there is no need to eliminate any answer sets. We effectively mimic the requirement that the QBF ϕ_n should hold for $\exists X_n$.

For a QBF of the form $\forall X_1 \exists X_2 \ldots \Theta X_n \cdot p(X_1, X_2, \ldots, X_n)$, with $\Theta = \exists$ if n is even and $\Theta = \forall$ otherwise, the proof is analogous. In the base case, we know that a QBF ϕ_1 of the form $\forall X_1 \cdot p(X_1)$ is satisfiable only when for every ()-focused answer set M of $\mathcal{P}_1 = \{Q_0\}$ we find that $Q_0 : sat \in M$. Otherwise, we know that there exists some ()-focused answers sets M of $\mathcal{P}_1$ such that $Q_0 : sat \notin M$. Hence the induction hypothesis is valid for $n = 1$. The induction step is then entirely analogous to what we have proven before, with the only difference being that the cases for n being even or odd are swapped. Finally, since the first quantifier is $\forall$, we need to verify that $Q_0 : sat$ is true in every $(Q_1, \ldots, Q_{n-1})$-focused answer set of $\mathcal{P}$. □

Before we discuss the computational complexity, we recall some of the notions of complexity theory. We have $\Sigma_0^{\mathsf{P}} = \mathsf{P}$ and $\Sigma_1^{\mathsf{P}} = \mathsf{NP}$ where the complexity class $\Sigma_n^{\mathsf{P}} = \mathsf{NP}^{\Sigma_{n-1}^{\mathsf{P}}}$ is the class of problems that can be solved in polynomial time on a non-deterministic machine with an $\Sigma_{n-1}^{\mathsf{P}}$ oracle, *i.e.* assuming a procedure that can solve $\Sigma_{n-1}^{\mathsf{P}}$ problems in constant time [19]. Deciding the validity of a QBF $\phi = \exists X_1 \forall X_2 ... \Theta X_n \cdot p(X_1, X_2, \cdots X_n)$ is the canonical Σ_n^{P}-complete problem. Furthermore $\Pi_n^{\mathsf{P}} = \mathsf{co}(\Sigma_n^{\mathsf{P}})$. Deciding the validity of a QBF $\phi = \forall X_1 \exists X_2 ... \Theta X_n \cdot p(X_1, X_2, \cdots X_n)$ with $\Theta = \forall$ if n is odd and $\Theta = \exists$ otherwise, is the canonical Π_n^{P}-complete problem.

Corollary 1. *Let $\mathcal{P}$ be a communicating program, $Q_i \in \mathcal{P}$. The problem of deciding whether $Q_i : l \in M$ (brave reasoning) with M a $(Q_1, \ldots, Q_n)$-focused answer set of $\mathcal{P}$ is $\Sigma_{n+1}^{\mathsf{P}}$-hard.*

Corollary 2. *Let $\mathcal{P}$ be a communicating program, $Q_i \in \mathcal{P}$. The problem of deciding whether all $(Q_1, \ldots, Q_n)$-focused answer sets contain $Q_i : l$ (cautious reasoning) is Π_{n+1}^{P}-hard.*

In addition to these hardness results, we can also establish the corresponding membership results.

Proposition 3. *Let $\mathcal{P}$ be a communicating program, $Q_i \in \mathcal{P}$. Deciding whether $Q_i : l \in M$ with M a $(Q_1, \ldots, Q_n)$-focused answer set of $\mathcal{P}$ is in $\Sigma_{n+1}^{\mathsf{P}}$.*

Proof. (sketch) This proof is by induction on n. When $n = 1$ we guess a (Q_1)-focused answer set M of $\mathcal{P}$ in polynomial time and verify that this is indeed a (Q_1)-focused answer set in coNP as in classical ASP, *i.e.* finding a (Q_1)-focused answer set is in Σ_2^{P}. Given an algorithm to compute the $(Q_1, \ldots, Q_{n-1})$-focused answer sets of $\mathcal{P}$ in Σ_n^{P}, we can guess a $(Q_1, \ldots, Q_n)$-focused answer set and verify there is no $(Q_1, \ldots, Q_n)$-focused answer set M' of $\mathcal{P}$ such that $M'_{\downarrow Q_n} \subset M_{\downarrow Q_n}$ using a Σ_n^{P} oracle, *i.e.* the algorithm is in $\Sigma_{n+1}^{\mathsf{P}}$. $\square$

Now that we have both hardness and membership results, we readily obtain the following corollary.

Corollary 3. *Let $\mathcal{P}$ be a communicating normal program, $Q_i \in \mathcal{P}$. The problem of deciding whether $Q_i\!:\!l \in M$ with M a $(Q_1, \ldots, Q_n)$-focused answer set of $\mathcal{P}$ is $\Sigma_{n+1}^{\mathsf{P}}$-complete.*

The next corollary provides a result for communicating simple programs instead of communicating normal programs.

Corollary 4. *Let $\mathcal{P}$ be a communicating simple program, $Q_i \in \mathcal{P}$. The problem of deciding whether $Q_i\!:\!l \in M$ with M a $(Q_1, \ldots, Q_n)$-focused answer set of $\mathcal{P}$ is $\Sigma_{n+1}^{\mathsf{P}}$-complete.*

5 Related Work

A lot of research has been done, for various reasons, on the subject of combining logic programming with the multi-agent paradigm. One reason for such a combination is that logics can be used to describe the (rational) behavior of the agents [9]. Another reason is that it can be used to combine different flavors of logic programming languages [11,18]. Such an extension of logic programming can be used to externally solve tasks for which ASP is not suited, while remaining in a declarative framework [13]. It can also be used as a form of cooperation, where multiple agents or contexts collaborate to solve a difficult problem [8,21]. The approach in this paper falls in the last category and is concerned with how the collaboration of different ASP programs affect the expressiveness of the overall system.

Important work has been done in the domain of multi-context systems (MCS) and multi-agent ASP to enable collaboration between the different contexts/ASP programs. We briefly discuss some of the more prominent work in these areas.

The work of [20] discusses an extension of MCSs [16] that allows MCSs to reason about absent information. Each context only has access to a subset of the available information. In order to share this information, an information flow is defined by the system between the different contexts. This idea was later adopted in the ASP community and in our work in particular.

The current paper has the same syntax as [20] but rather different semantics. The semantics in [20] are closely related to the well-founded semantics [14], whereas ours are closer to the spirit of stable models [15]. Another point where our semantics differ is that the motivation for accepting a literal as being true can be circular if that explanation relies on other component programs. In [5] this circularity is identified as a requirement for the representation of social reasoning.

The work in [4] further extends the work in [20], introducing a multi-context variant of default logic. This extension guarantees that a number of conclusions from default logic come "for free". The paper is the first to offer a syntactical description of the communication rather than a semantic one, making it easier to implement an actual algorithm. Some of interesting applications of contextual frameworks are shown, *e.g.* information fusion, game theory and social choice.

Along similar lines [3] combines the non-monotonicity from [20] with the heterogeneous approach from [16] into a single framework for heterogenous non-monotonic multi-context reasoning. The work in [3] introduces several notions of equilibria, including minimal and grounded equilibria. In our approach, local reasoning is captured by grounded equilibria (no circularity allowed) while communicating with other component programs is captured by the weaker concept of minimal equilibria. The work in [3] also offers various membership results on checking the existence of an equilibrium. Most notably, [3] is – to the best of our knowledge – the first to explicitly remark that multi-context systems can be non-monotonic even if all the logics in the component programs are monotonic.

We now direct our attention to work done within the ASP community. The ideas presented in this paper are related to HEX programs [12] in which ASP is extended by higher-order predicates and external atoms. Through these external atoms, knowledge can be exchanged with external sources while remaining within the declarative paradigm. Applicability-wise, HEX is proposed as a tool for non-monotonic semantic web reasoning under the answer set semantics. Because of this setting, HEX is not primarily targeted at increasing the expressiveness, but foremost at extending the applicability and ease of use of ASP.

Two other important works in the area of multi-agent ASP are [8] and [21]. In both [8] and [21] a multi-agent system is developed in which multiple agents communicate with each other. The communication channel is uni-directional, allowing information to be pushed to the next agent. Both approaches use ASP and have agents that are quite expressive in their own right. Indeed, in [8] each agent is an Ordered Choice Logic Program [2] and in [21] each agent uses the extended answer set semantics.

In [8] the agents can communicate with whomever they want and circular communication is allowed (agent A tells B whom tells $A \ldots$), which is similar to our approach. However, in [8] only positive information can be shared and the authors do not look at the actual expressiveness of the framework. In [21] Hierarchical Decision Making is introduced where each agent uses the extended answer set semantics. The network is a linear "hierarchical" network and the framework employs the idea of a failure feedback mechanism. Intuitively, this mechanism allows the previous agent in a network to revise his conclusion when it leads to an unresolvable inconsistency for the next agent in the network. It is this mechanism that gives rise to a higher expressiveness, namely Σ_n^{P} for a hierarchical network of n agents. Our work is different in that we start from normal and simple ASP programs for the agents. Our communication mechanism is also quite simple and does not rely on any kind of feedback. Regardless, we obtain a comparable expressiveness.

We briefly mention [7], in which recursive modular non-monotonic logic programs (MLP) under the ASP semantics are considered. The main difference between MLP and our simple communication is that our communication is parameter-less, *i.e.* the truth of a situated literal is not dependent on parameters passed by the situated literal to the target component program. Our approach is clearly different and we cannot readily mimic the behavior of the network as presented in [7]. Our complexity results therefore do not directly apply to MLPs.

Finally, we like to point out the resemblance between multi-focused answer sets and the work on multi-level linear programming [17]. In multi-level linear programming, different agents control different variables that are outside of the control of the other agents, yet are linked by means of linear inequalities (constraints). The agents have to fix the values of the variables they can control in a predefined order, such that their own linear objective function is optimized. Similarly, in communicating ASP, literals belong to different component programs (agents), and their values are linked through constraints, which in this case take the form of rules. Again the agents act in a predefined order, but now they try to minimize the set of literals they have to accept as being true, rather than a linear objective function.

6 Conclusion

We have introduced multi-focused answer sets for communicating programs. The underlying intuition is that of leaders and followers, where the choices available to the followers are limited by what the leaders have previously decided. On a technical level, the problem translates to establishing local minimality for some of the component programs in the communicating program. Since in general it is not possible to ensure local minimality for all component programs, an order must be defined among component programs on which to focus. The result is an increase in expressiveness, where the problem of deciding whether $Q_i : l \in M$ with M a $(Q_1, \ldots, Q_n)$-focused answer set of $\mathcal{P}$ is $\Sigma_{n+1}^{\mathsf{P}}$-complete. In our work we thus find that the choice of the communication mechanism is paramount *w.r.t.* the expressiveness of the overall system, irrespective of the expressiveness of the individual agents.

References

1. Bauters, K., Janssen, J., Schockaert, S., Vermeir, D., De Cock, M.: Communicating answer set programs. In: Tech. Comm. of ICLP 2010, vol. 7, pp. 34–43 (2010)
2. Brain, M., De Vos, M.: Implementing OCLP as a front-end for answer set solvers: From theory to practice. In: Proc. of ASP 2005 (2003)
3. Brewka, G., Eiter, T.: Equilibria in heterogeneous nonmonotonic multi-context systems. In: Proc. of AAAI 2007, pp. 385–390 (2007)

4. Brewka, G., Roelofsen, F., Serafini, L.: Contextual default reasoning. In: Proc. of. IJCAI 2007, pp. 268–273 (2007)
5. Buccafurri, F., Caminiti, G., Laurendi, R.: A logic language with stable model semantics for social reasoning. In: Garcia de la Banda, M., Pontelli, E. (eds.) ICLP 2008. LNCS, vol. 5366, pp. 718–723. Springer, Heidelberg (2008)
6. Bylander, T.: The computational complexity of propositional STRIPS planning. Artificial Intelligence 69, 165–204 (1994)
7. Dao-Tran, M., Eiter, T., Fink, M., Krennwallner, T.: Modular nonmonotonic logic programming revisited. In: Hill, P.M., Warren, D.S. (eds.) ICLP 2009. LNCS, vol. 5649, pp. 145–159. Springer, Heidelberg (2009)
8. De Vos, M., Crick, T., Padget, J., Brain, M., Cliffe, O., Needham, J.: LAIMA: A multi-agent platform using ordered choice logic programming. In: Baldoni, M., Endriss, U., Zhang, S.-W., Torroni, P. (eds.) DALT 2005. LNCS (LNAI), vol. 3904, pp. 72–88. Springer, Heidelberg (2006)
9. Dell'Acqua, P., Sadri, F., Toni, F.: Communicating agents. In: Proc. of MASL 1999 (1999)
10. Eiter, T., Gottlob, G.: The complexity of logic-based abduction. Journal of the ACM 42, 3–42 (1995)
11. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. Artifial Intelligence 172(12-13), 1495–1539 (2008)
12. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In: Proc. of IJCAI 2005, pp. 90–96 (2005)
13. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: dlvhex: A tool for semantic-web reasoning under the answer-set semantics. In: Proc. of ALPSWS 2006, pp. 33–39 (2006)
14. Gelder, A.V., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. Journal of the ACM 38(3), 620–650 (1991)
15. Gelfond, M., Lifzchitz, V.: The stable model semantics for logic programming. In: Proc. of ICLP 1988, pp. 1081–1086 (1988)
16. Giunchiglia, F., Serafini, L.: Multilanguage hierarchical logics or: How we can do without modal logics. Artifial Intelligence 65(1), 29–70 (1994)
17. Jeroslow, R.: The polynomial hierarchy and a simple model for competitive analysis. Mathematical Programming 32, 146–164 (1985)
18. Luo, J., Shi, Z., Wang, M., Huang, H.: Multi-agent cooperation: A description logic view. In: Lukose, D., Shi, Z. (eds.) PRIMA 2005. LNCS, vol. 4078, pp. 365–379. Springer, Heidelberg (2009)
19. Papadimitriou, C.: Computational complexity. Addison-Wesley, Reading (1994)
20. Roelofsen, F., Serafini, L.: Minimal and absent information in contexts. In: Proc. of IJCAI 2005, pp. 558–563 (2005)
21. Van Nieuwenborgh, D., De Vos, M., Heymans, S., Hadavandi, E.: Hierarchical decision making in multi-agent systems using answer set programming. In: Inoue, K., Satoh, K., Toni, F. (eds.) CLIMA 2006. LNCS (LNAI), vol. 4371, pp. 20–40. Springer, Heidelberg (2007)