arXiv:1109.2434v1 [cs.LO] 12 Sep 2011

Expressiveness of Communication in Answer Set Programming

KIM BAUTERS, STEVEN SCHOCKAERT

Dept. of Applied Mathematics and Computer Science,
Krijgslaan 281 (WE02), Universiteit Gent, 9000 Gent, Belgium
(e-mail: `(kim.bauters, steven.schockaert)@ugent.be`*)*

JEROEN JANSSEN, DIRK VERMEIR

Dept. of Computer Science, Vrije Universiteit Brussel,
Pleinlaan 2, 1050 Brussel, Belgium
(e-mail: `(jeroen.janssen, dirk.vermeir)@vub.ac.be`*)*

MARTINE DE COCK

Dept. of Applied Mathematics and Computer Science,
Krijgslaan 281 (WE02), Universiteit Gent, 9000 Gent, Belgium
(e-mail: `martine.decock@ugent.be`*)*

Abstract

Answer set programming (ASP) is a form of declarative programming that allows to succinctly formulate and efficiently solve complex problems. An intuitive extension of this formalism is communicating ASP, in which multiple ASP programs collaborate to solve the problem at hand. However, the expressiveness of communicating ASP has not been thoroughly studied. In this paper, we present a systematic study of the additional expressiveness offered by allowing ASP programs to communicate. First, we consider a simple form of communication where programs are only allowed to ask questions to each other. For the most part, we deliberately only consider simple programs, *i.e.* programs for which computing the answer sets is in P. We find that the problem of deciding whether a literal is in some answer set of a communicating ASP program using simple communication is NP-hard. In other words: we move up a step in the polynomial hierarchy due to the ability of these simple ASP programs to communicate and collaborate. Second, we modify the communication mechanism to also allow us to focus on a sequence of communicating programs, where each program in the sequence may successively remove some of the remaining models. This mimics a network of leaders, where the first leader has the first say and may remove models that he or she finds unsatisfactory. Using this particular communication mechanism allows us to capture the entire polynomial hierarchy. 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 as well as PSPACE-complete problems such as STRIPS planning.

KEYWORDS: logic programming, answer set programming, multi-agent reasoning

1 Introduction

Answer set programming (ASP) is a form of non-monotonic reasoning based on the stable model semantics (Gelfond and Lifzchitz 1988). ASP has proven successful as an elegant and convenient vehicle for commonsense reasoning in discrete domains and to encode combinatorial optimization problems in a purely declarative way. It has been applied in, for example, plan generation (Lifschitz 2002), diagnosis (Eiter et al. 1999) and biological networks (Gebser et al. 2010). Being an active field of research, a large body of extensions have been proposed that improve upon the basics of ASP and offer, for example, cardinality constraints (Niemelä and Simons 2000) or nested expressions (Lifschitz et al. 1999). Not all of these extensions provide an increase in terms of computational expressiveness and some are merely convenient syntactic sugar.

One particularly interesting extension of ASP is called communicating ASP. It allows for a number of ASP programs to communicate, *i.e.* share information about their knowledge base, giving them the ability to cooperate with each other to solve the problem at hand. Each ASP program involved, called a component program, has individual beliefs and reasoning capabilities. One of the benefits of this extension is that it eases the declarative formulation of a problem originating in a multi-agent context. Quite a number of different flavours of communicating ASP have been proposed in the literature, both in the narrow domain of ASP and in the broader domain of logic programming, where each of these papers presents intriguing examples that highlight the usefulness of communicating ASP, for example (Dell'Acqua et al. 1999), (Brain and De Vos 2003), (Roelofsen and Serafini 2005), (De Vos et al. 2005) and (Van Nieuwenborgh et al. 2007). In particular, all the examples involve multi-dimensional problems (*e.g.* a police investigation with multiple witnesses) where each agent only has the knowledge contained in one or a few of the dimensions (*e.g.* the witness only saw the burglar enter the building).

A standard example to illustrate the usefulness in this area is shown in Figure 1.[1] The figure depicts two people who are looking at a box. The box is called magic because neither agent can make out its depth. The information the agents know is further limited because parts of the box are blinded. By cooperation, both agents can pinpoint the exact location of the ball. Indeed, *Mr.2* sees a ball on his left side. From this information *Mr.1* knows that there is a ball and that it must therefore be on his left (since he cannot see a ball on his right). This knowledge can be relayed back to *Mr.2*. Both agents now know the exact position and depth of the ball.

Complexity results from (Brewka et al. 2007) show that computing the answer sets of a communicating logic program is in NP. In general, however, only few results exist regarding the expressiveness of such communicating ASP programs. In addition, for many of the known results, it is not clear whether an increase in expressiveness is due to the type of component programs considered (*i.e.* the expressiveness of the individual programs involved) or due to the use of communication.

In communicating ASP, the notion of an answer set can be defined in different

---

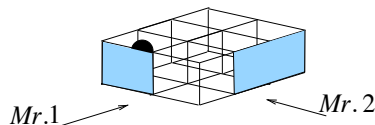[1] Illustration from (Roelofsen and Serafini 2005) used with permission from the authors.

Fig. 1. A magic box.

ways; we refer to the specific definition that is used as the communication mechanism. 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. One option is to assume global minimality, *i.e.* we minimise over the conclusions of all the agents in the network. Another option is to assume minimality on the level of a single agent. Since in general it is not possible to find a model that is minimal for all individual agents, the order in which we minimise over the agents matters.

The main contributions of this paper are as follows. We present a systematic study of the additional expressiveness offered by allowing ASP programs to communicate, where, for the most part, we deliberately only consider component programs that are simple, *i.e.* programs for which computing the answer sets is in P. Clearly any step we move up in the polynomial hierarchy is then due to the ability of these simple ASP programs to communicate and collaborate with each other. We show that the problem of deciding whether a literal is in any answer set of a communicating ASP program using simple communication is in NP. We also provide a simulation using a network of simple programs that is capable of modeling any program with negation-as-failure. These results are extended with complexity results for when the network consists of disjunctive programs. In all cases, we examine the complexity of determining whether an answer set exists as well as whether a literal is true in any (brave reasoning) or all answer sets (cautious reasoning). Furthermore, we introduce the notion of multi-focused answer sets of communicating ASP programs, which allows us to successively focus (*i.e.* minimise) on different agents. 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 (Eiter and Gottlob 1995) as well as PSPACE-complete problems such as STRIPS planning (Bylander 1994). We show that these results hold for any choice of program in the network, being it either simple, normal or disjunctive, and we provide the complexity of determining whether a multi-focused answer set exists.

This paper aggregates and extends our work from (Bauters et al. 2010; Bauters et al. 2011). The first work is extended with completeness results and an elaborate overview of the simulation of negation-as-failure. Furthermore, we provide more detailed complexity results including the complexity of determining whether an answer set exists and the complexity of determining whether some literal is true in some or all of the answer sets. The focused answer sets introduced in Section 5

are from (Bauters et al. 2011) and are more general than the corresponding notion that we introduced in (Bauters et al. 2010). Additional results are provided, such as the complexity of determining whether a multi-focused answer set exists, as well as new results concerning the use of disjunctive programs as component programs. We also provide the proofs of all results in the appendix.

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 *disjunctive rule* is an expression of the form $\gamma \leftarrow (\alpha \cup not(\beta))$ where γ is a set of literals (interpreted as a disjunction, denoted as $l_1; \ldots; l_n$) called the head of the rule and $(\alpha \cup not(\beta))$ (interpreted as a conjunction) is 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 using extended literals.[2] A *positive disjunctive rule* is a disjunctive rule without negation-as-failure in the body, *i.e.* with $\beta = \emptyset$. A *disjunctive program P* is a finite set of disjunctive 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 $I \subseteq (\mathcal{B}_P \cup \neg\mathcal{B}_P)$. I is *total* iff $I \cup \neg I = \mathcal{B}_P \cup \neg\mathcal{B}_P$.

A *normal rule* is a disjunctive rule with exactly one literal l in the head. A *normal program P* is a finite set of normal rules. 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.

The satisfaction relation $\models$ is defined for an interpretation I as $I \models l$ iff $l \in I$, otherwise $I \not\models l$. For an interpretation I and L a set of literals, we define $I \models L$ iff $\forall l \in L \cdot I \models l$. An interpretation I is a *model* of a positive disjunctive rule $r = \gamma \leftarrow \alpha$, denoted $I \models r$, if $I \not\models \neg\gamma$ or $I \not\models \alpha$, *i.e.* the body is false or at least one of the literals in the head can be true. An interpretation I of a positive disjunctive program P is a *model* of P iff $\forall r \in P \cdot I \models r$.

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

$$T_P(I) = I \cup \{l \mid ((l \leftarrow \alpha) \in P) \wedge (\alpha \subseteq I)\}. \tag{1}$$

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. An interpretation I is an *answer set* of a simple program P iff $I = P^\star$.

---

[2] We can simulate a constraint $(\leftarrow body)$ by $(fail \leftarrow not\ fail, body)$ with *fail* a fresh atom.

The *reduct* P^I of a disjunctive program P *w.r.t.* the interpretation I is defined as $P^I = \{\gamma \leftarrow \alpha \mid (\gamma \leftarrow \alpha \cup not(\beta)) \in P, \beta \cap I = \emptyset\}$. We say that I is an answer set of the disjunctive program P when I is a minimal model of P^I *w.r.t.* set inclusion.

In the specific case of normal programs, answer sets can also be characterised in terms of fixpoints. Specifically, it is easy to see that in this case the reduct P^I is a simple program. We say that I is an answer set of a normal program P iff $\left(P^I\right)^\star = I$, *i.e.* if I is the answer set of the reduct P^I.

3 Communicating programs

The underlying intuition of communication between ASP programs is that of a function call or, in terms of agents, asking questions to other agents. This communication is based on a new kind of literal '$Q{:}l$', as in (Giunchiglia and Serafini 1994; Roelofsen and Serafini 2005; Brewka and Eiter 2007). If the literal l is not in the answer set of program Q then $Q{:}l$ is false; otherwise $Q{:}l$ is true. The semantics presented in this section are closely related to the minimal semantics of (Brewka and Eiter 2007) and especially the semantics of (Buccafurri et al. 2008).

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. For $R \in \mathcal{P}$, a $\mathcal{P}$-situated literal $Q{:}l$ is called R-*local* if $Q = R$. For a set of literals L, we use $Q{:}L$ as a shorthand for $\{Q{:}l \mid l \in L\}$. For a set of $\mathcal{P}$-situated literals X and $Q \in \mathcal{P}$, we use X_Q to denote $\{l \mid Q{:}l \in X\}$, i.e. the projection of X on Q. A set of $\mathcal{P}$-situated literals X is *consistent* iff X_Q is consistent for all $Q \in \mathcal{P}$. By $\neg X$ we denote the set $\{Q{:}\neg l \mid Q{:}l \in X\}$ where we define $Q{:}\neg\neg l = Q{:}l$. 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 $\{not\ Q{:}l \mid Q{:}l \in X\}$. For a set of extended $\mathcal{P}$-situated literals X we denote by X_{pos} the set of $\mathcal{P}$-situated literals in X, *i.e.* those extended $\mathcal{P}$-situated literals in X that are not preceded by negation-as-failure, while $X_{\mathrm{neg}} = \{Q{:}l \mid not\ Q{:}l \in X\}$.

A $\mathcal{P}$-*situated disjunctive rule* is an expression of the form $Q{:}\gamma \leftarrow (\alpha \cup not(\beta))$ where γ is a set of literals, 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 disjunctive rule $Q{:}\gamma \leftarrow (\alpha \cup not\ (\beta))$ is called R-local whenever $Q = R$. A $\mathcal{P}$-*component disjunctive program* Q is a finite set of Q-local $\mathcal{P}$-situated disjunctive 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 disjunctive programs. A *communicating disjunctive program* $\mathcal{P}$ is then a finite set of $\mathcal{P}$-component disjunctive programs.

A $\mathcal{P}$-*situated normal rule* is an expression of the form $Q{:}l \leftarrow (\alpha \cup not(\beta))$ where $Q{:}l$ is a single $\mathcal{P}$-situated literal. 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. A $\mathcal{P}$-*component normal (resp. simple) program* Q is a finite set of Q-local $\mathcal{P}$-situated normal (resp. simple) rules. A *communicating normal (resp. simple) program* $\mathcal{P}$ is then a finite set of $\mathcal{P}$-component normal (resp. 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 disjunctive program Q is

clear, we write l instead of $Q\!:\!l$ for Q-local situated literals. Note that a communicating disjunctive (resp. normal, simple) program with only one component program thus trivially corresponds to a classical disjunctive (resp. normal, simple) program. Finally, for notational convenience, we write communicating program when it is clear from the context whether the program is a communicating simple program or a communicating normal program.

Example 1
Consider the communicating normal program $\mathcal{P} = \{Q, R\}$ with the following situated rules:

$$Q\!:\!a \leftarrow R\!:\!a \qquad\qquad Q\!:\!b \leftarrow \qquad\qquad\qquad Q\!:\!c \leftarrow Q\!:\!c$$
$$R\!:\!a \leftarrow Q\!:\!a \qquad\qquad R\!:\!b \leftarrow \textit{not } Q\!:\!c.$$

$Q\!:\!a$, $Q\!:\!b$, $Q\!:\!c$, $R\!:\!a$ and $R\!:\!b$ are situated literals. The situated simple rules on the top line are Q-local since we respectively have $Q\!:\!a$, $Q\!:\!b$ and $Q\!:\!c$ in the head of these rules. The situated normal rules on the bottom line are R-local. Hence $Q = \{a \leftarrow R\!:\!a, b \leftarrow, c \leftarrow c\}$ and $R = \{a \leftarrow Q\!:\!a, b \leftarrow \textit{not } Q\!:\!c\}$.

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

Example 2
Given the communicating normal program $\mathcal{P} = \{Q, R\}$ from Example 1 we have that $\mathcal{B}_Q = \{a, b, c\}$, $\mathcal{B}_R = \{a, b\}$ and $\mathcal{B}_{\mathcal{P}} = \{Q\!:\!a, Q\!:\!b, Q\!:\!c, R\!:\!a, R\!:\!b, R\!:\!c\}$.

We say that a (partial) interpretation I of a communicating disjunctive 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 disjunctive program $\mathcal{P}$, the reduct Q^I for $Q \in \mathcal{P}$ is the component disjunctive 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.

Note that this definition actually combines two types of reducts together. On the one hand, we remove the negation-as-failure according to the given knowledge. On the other hand, we also remove situated literals that are not Q-local, again according to the given knowledge. The underlying intuition of the reduct remains unchanged compared to the classical case: we take the information into account which is encoded in the guess I and we simplify the program so that we can easily verify whether or not I is stable, *i.e.* whether or not I is a minimal model of the reduct. Analogous to the definition of the reduct for disjunctive programs (Gelfond and Lifschitz 1991), the reduct of a communicating disjunctive program thus defines a way to reduce a program relative to some guess I. The reduct of a communicating

disjunctive program is a communicating disjunctive program (without negation-as-failure) that only contains component disjunctive programs Q with Q-local situated literals. That is, each remaining component disjunctive program Q corresponds to a classical disjunctive program.

Example 3
Let us once again consider the communicating normal program $\mathcal{P} = \{Q, R\}$ from Example 1. Given $I = \{Q\!:\!a, Q\!:\!b, R\!:\!a, R\!:\!b\}$ we find that $Q^I = \{a \leftarrow, b \leftarrow, c \leftarrow c\}$ and $R^I = \{a \leftarrow, b \leftarrow\}$. We can easily treat Q^I and R^I separately since they now correspond to classical programs.

Definition 1
We say that an interpretation I of a communicating disjunctive program $\mathcal{P}$ is an *answer set* of $\mathcal{P}$ if and only if $\forall Q \in \mathcal{P} \cdot (Q\!:\!I_Q)$ is the minimal model *w.r.t.* set inclusion of Q^I. In other words: an interpretation I is an answer set of a communicating disjunctive program $\mathcal{P}$ if and only if for every component program Q we have that the projection of I on Q is an answer set of the component program Q^I under the classical definition.

In the specific case of a communicating normal program $\mathcal{P}$ we can equivalently say that I is an answer set of $\mathcal{P}$ if and only if we have that $\forall Q \in \mathcal{P} \cdot (Q\!:\!I_Q) = \left(Q^I\right)^{\star}$.

Example 4
The communicating normal program $\mathcal{P} = \{Q, R\}$ from Example 1 has two answer sets, namely $\{Q\!:\!b, R\!:\!b\}$ and $\{Q\!:\!a, Q\!:\!b, R\!:\!a, R\!:\!b\}$.

Note that while most approaches do not allow self-references of the form $Q\!:\!a \leftarrow Q\!:\!a$, in our approach this poses no problems as it is semantically equivalent to $Q\!:\!a \leftarrow a$. Also note that our semantics allow for "mutual influence" as in (Brewka and Eiter 2007; Buccafurri et al. 2008) where the belief of an agent can be supported by the agent itself, via belief in other agents, *e.g.* $\{Q\!:\!a \leftarrow R\!:\!a, R\!:\!a \leftarrow Q\!:\!a\}$. Furthermore we want to point out that the belief between agents is the belief as identified in (Lifschitz et al. 1999), *i.e.* the situated literal $Q\!:\!l$ is true in our approach whenever "$\neg not\ Q\!:\!l$" is true in the approach introduced in (Lifschitz et al. 1999) for nested logic programs and treating $Q\!:\!l$ as a fresh atom.

Before we introduce our first proposition, we generalise the immediate consequence operator for (classical) normal programs to the case of communicating simple programs. Specifically, the operator $T_{\mathcal{P}}$ is defined *w.r.t.* an interpretation I of $\mathcal{P}$ as

$$T_{\mathcal{P}}(I) = I \cup \{Q\!:\!l \mid (l \leftarrow \alpha) \in Q, Q \in \mathcal{P}, \alpha \subseteq I\}$$

where α is a set of $\mathcal{P}$-situated literals. It is easy to see that this operator is monotone. Together with a result from (Tarski 1955) we know that this operator has a least fixpoint. We use $\mathcal{P}^{\star}$ to denote this fixpoint obtained by repeatedly applying $T_{\mathcal{P}}$ starting from the empty interpretation. Clearly, this fixpoint can be computed in polynomial time. Furthermore, just like the immediate consequence operator for (classical) normal programs, this generalised operator only derives the information that is absolutely necessary, *i.e.* the fixpoint $\mathcal{P}^{\star}$ is globally minimal.

Proposition 1
Let $\mathcal{P}$ be a communicating simple program. We then have that:

- there always exists at least one answer set of $\mathcal{P}$;
- there is always a unique answer set of $\mathcal{P}$ that is globally minimal;
- we can compute this unique globally minimal answer set in polynomial time.

Example 5
Consider the communicating simple program $\mathcal{P}$ with the rules

$$Q\!:\!a \leftarrow R\!:\!a \qquad\qquad R\!:\!a \leftarrow Q\!:\!a \qquad\qquad Q\!:\!b \leftarrow .$$

This communicating simple program has two answer sets, namely $\{Q\!:\!a, Q\!:\!b, R\!:\!a\}$ and $\{Q\!:\!b\}$. We have that $\mathcal{P}^{\star} = \{Q\!:\!b\}$, *i.e.* $\{Q\!:\!b\}$ is the answer set that can be computed in polynomial time. Intuitively, this is the answer set of the communicating simple program $\mathcal{P}$ where we treat every situated literal as an ordinary literal. For example, if we replace the situated literal $Q\!:\!a$ (resp. $Q\!:\!b$, $R\!:\!a$) by the literals qa (resp. qb, ra) we obtain the simple program

$$qa \leftarrow ra \qquad\qquad qb \leftarrow \qquad\qquad ra \leftarrow qa$$

which has the unique answer set $\{qb\}$, with qb the literal that replaced $Q\!:\!b$. Note that the procedure involving the generalised fixpoint does not allow us to derive the second answer set. In general, no polynomial procedure will be able to verify whether there is some answer set in which a given literal is true (unless $\mathsf{P}=\mathsf{NP}$).

Although finding an answer set of a communicating simple program can be done in polynomial time, we will see in the next section that brave reasoning (the problem of determining whether a given situated literal $Q\!:\!l$ occurs in any answer set of a communicating simple program) is NP-hard. Consequently, cautious reasoning (the problem of determining whether a given literal $Q\!:\!l$ occurs in all answer sets of a communicating simple program) is coNP-hard.

4 Simulating Negation-as-Failure with Communication

The addition of communication to ASP programs can provide added expressiveness over simple programs and a resulting increase in computational complexity for brave reasoning and cautious reasoning. To illustrate this observation, in this section we show that a communicating simple program can simulate normal programs.[3] Furthermore, we illustrate that, surprisingly, there is no difference in terms of computational complexity between communicating simple programs and communicating normal programs; a communicating simple program can be constructed which simulates any given communicating normal program.

---

[3] Recall that simple programs are P-complete and normal programs are NP-complete (Baral 2003).

For starters, we recall some of the notions of complexity theory. The complexity classes Δ_n^P, Σ_n^P and Π_n^P are defined as follows, for $i \in \mathbb{N}$ (Papadimitriou 1994):

$$\Delta_0^P = \Sigma_0^P = \Pi_0^P = P$$
$$\Delta_{i+1}^P = P^{\Sigma_i^P}$$
$$\Sigma_{i+1}^P = NP^{\Sigma_i^P}$$
$$\Pi_{i+1}^P = co\left(\Sigma_{i+1}^P\right)$$

where $NP^{\Sigma_i^P}$ (resp. $P^{\Sigma_i^P}$) is the class of problems that can be solved in polynomial time on a non-deterministic machine (resp. deterministic machine) with an Σ_i^P oracle, *i.e.* assuming a procedure that can solve Σ_i^P problems in constant time. For a general complexity class C, a problem is C-hard if any other problem in C can be efficiently reduced to this problem A problem is said to be C-complete if the problem is in C and the problem is C-hard. Deciding the validity of a QBF $\phi = \exists X_1 \forall X_2 ... \Theta X_n \cdot p(X_1, X_2, \cdots X_n)$ with $\Theta = \exists$ if n is odd and $\Theta = \forall$ otherwise, is the canonical Σ_n^P-complete problem. 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. Moreover, these results also hold when we restrict ourselves to problems with $p(X_1, X_2, \cdots X_n)$ in disjunctive normal form, except when the last quantifier is an $\exists$.[4] Brave reasoning as well as answer set existence for simple, normal and disjunctive programs is P-complete, NP-complete and Σ_2^P-complete, respectively (Baral 2003). Cautious reasoning for simple, normal and disjunctive programs is coP-complete, $coNP$-complete and $co\Sigma_2^P$-complete (Baral 2003).

In this section we start by giving an example of the transformation that allows us to simulate (communicating) normal programs using communicating simple programs. A formal definition of the simulation is given below in Definition 2. The correctness is proven by Propositions 2 and 3.

Example 6

Consider the communicating normal program $\mathcal{P}$ with the rules

$$Q_1 : a \leftarrow not\ Q_2 : b$$
$$Q_2 : b \leftarrow not\ Q_1 : a.$$

Note that if we were to take $Q_1 = Q_2$ then this example corresponds to a normal program. In our simulation, the communicating normal program $\mathcal{P}$ is transformed

---

[4] Given a QBF with the last quantifier an $\exists$ and a formula in disjunctive normal form, we can reduce the problem in polynomial time to a new QBF without the last quantifier. To do this, for every variable quantified by this last quantifier we remove those clauses in which both the quantified variable and its negation occur (contradiction) and then remove all occurrences of the quantified variables in the remaining clauses as well as the quantifier itself. The new QBF is then valid if and only if the original QBF is valid.

into the following communicating simple program $\mathcal{P}' = \{Q'_1, Q'_2, N_1, N_2\}$:

$$Q'_1 : a \leftarrow N_2 : \neg b^\dagger \qquad\qquad N_1 : a^\dagger \leftarrow Q'_1 : a$$
$$Q'_2 : b \leftarrow N_1 : \neg a^\dagger \qquad\qquad N_2 : b^\dagger \leftarrow Q'_2 : b$$
$$Q'_1 : \neg a^\dagger \leftarrow N_1 : \neg a^\dagger \qquad\qquad N_1 : \neg a^\dagger \leftarrow Q'_1 : \neg a^\dagger$$
$$Q'_2 : \neg b^\dagger \leftarrow N_2 : \neg b^\dagger \qquad\qquad N_2 : \neg b^\dagger \leftarrow Q'_2 : \neg b^\dagger.$$

The transformation creates two types of component programs or *'worlds'*, namely Q'_i and N_i. The component program Q'_i is similar to Q_i but occurrences of extended situated literals of the form *not* $Q_i : l$ are replaced by $N_i : \neg l^\dagger$, with $l^\dagger$ a fresh literal. The non-monotonicity associated with negation-as-failure is simulated by introducing the rules $\neg l^\dagger \leftarrow N_i : \neg l^\dagger$ and $\neg l^\dagger \leftarrow Q'_i : \neg l^\dagger$ in Q'_i and N_i, respectively. Finally, we add rules of the form $l^\dagger \leftarrow Q'_i : l$ to N_i, creating an inconsistency when N_i believes $\neg l^\dagger$ and Q'_i believes l.

The resulting communicating simple program $\mathcal{P}'$ is an equivalent program in that its answer sets correspond to those of the original communicating normal program, yet without using negation-as-failure. Indeed, the answer sets of $\mathcal{P}$ are $\{Q_1 : a\}$ and $\{Q_2 : b\}$ and the answer sets of $\mathcal{P}'$ are $\{Q'_1 : a\} \cup \{Q'_2 : \neg b^\dagger, N_2 : \neg b^\dagger, N_1 : a^\dagger\}$ and $\{Q'_2 : b\} \cup \{Q'_1 : \neg a^\dagger, N_1 : \neg a^\dagger, N_2 : b^\dagger\}$.

Note that the simulation given in Example 6 can in fact be simplified. Indeed, in this particular example there is no need to have two additional component programs N_1 and N_2 since Q_1 and Q_2 do not share literals. Also, in this particular example, we need not use '$a^\dagger$' and '$b^\dagger$' since the simulation would work just as well if we simply considered 'a' and 'b' instead. Nonetheless, for the generality of the simulation such technicalities are necessary. Without adding an additional component program N_i for every original component program Q_i the simulation would in general not work when two component programs shared literals, *e.g.* $Q_1 : a$ and $Q_2 : a$. Furthermore, we need to introduce fresh literals as otherwise the simulation would in general not work when we had true negation in the original program, *e.g.* $Q : \neg a$. We now give the definition of the simulation which works in the general case.

Definition 2
Let $\mathcal{P} = \{Q_1, \ldots, Q_n\}$ be a communicating normal program. The communicating simple program $\mathcal{P}' = \{Q'_1, \ldots, Q'_n, N_1, \ldots, N_n\}$ with $1 \le i, j \le n$ that simulates $\mathcal{P}$ is defined by

$$Q'_i = \left\{ l \leftarrow \alpha'_{\text{pos}} \cup \left\{ N_j : \neg k^\dagger \mid Q_j : k \in \alpha_{\text{neg}} \right\} \mid (l \leftarrow \alpha) \in Q_i \right\} \tag{2}$$
$$\cup \quad \left\{ \neg b^\dagger \leftarrow N_i : \neg b^\dagger \mid Q_i : b \in \mathcal{E}_{\text{neg}} \right\} \tag{3}$$
$$N_i = \left\{ \neg b^\dagger \leftarrow Q'_i : \neg b^\dagger \mid Q_i : b \in \mathcal{E}_{\text{neg}} \right\} \tag{4}$$
$$\cup \quad \left\{ b^\dagger \leftarrow Q'_i : b \mid Q_i : b \in \mathcal{E}_{\text{neg}} \right\} \tag{5}$$

with $\alpha' = \{Q'_j : l \mid Q_j : l \in \alpha\}$, $\mathcal{E}_{\text{neg}} = \bigcup_{i=1}^n \left(\bigcup_{(a \leftarrow \alpha) \in Q_i} \alpha_{\text{neg}} \right)$ and with α_{pos} and α_{neg} as defined before. Note how this is a polynomial transformation with at most $3 \cdot |\mathcal{E}_{\text{neg}}|$ additional rules. This is important when later we use the NP-completeness results from normal programs to show that communicating simple programs are

NP-complete as well. Recall that both $\neg b^\dagger$ and $b^\dagger$ are fresh literals that intuitively correspond to $\neg b$ and b. We use Q'_i+ to denote the set of rules in Q'_i defined by (2) and Q'_i- to denote the set of rules in Q'_i defined by (3).

The intuition of the simulation in Definition 2 is as follows. The simulation uses the property of mutual influence to mimic the choice induced by negation-as-failure. This is obtained from the interplay between rules (3) and (4). As such, we can use the new literal '$\neg b^\dagger$' instead of the original extended (situated) literal '*not b*', allowing us to rewrite the rules as we do in (2). In order to ensure that the simulation works even when the program we want to simulate already contains classical negation, we need to specify some additional bookkeeping (5).

As will become clear from Proposition 2 and Proposition 3, the above transformation preserves the semantics of the original program. Since we can thus rewrite any normal program as a communicating normal program, the importance is twofold. On one hand, we reveal that communicating normal programs do not have any additional expressive power over communicating simple programs. On the other hand, it follows that communicating simple programs allow us to solve NP-complete problems. Before we show the correctness of the simulation in Definition 2, we introduce a lemma.

Lemma 1
Let $\mathcal{P} = \{Q_1, \ldots, Q_n\}$ and let $\mathcal{P}' = \{Q'_1, \ldots, Q'_n, N_1, \ldots, N_n\}$ with $\mathcal{P}$ a communicating normal program and $\mathcal{P}'$ the communicating simple program that simulates $\mathcal{P}$ defined in Definition 2. Let M be an answer set of $\mathcal{P}$ and let the interpretation M' be defined as:

$$
\begin{aligned}
M' = & \{Q'_i{:}a \mid Q_i{:}a \in M\} \\
& \cup \ \{Q'_i{:}\neg b^\dagger \mid Q_i{:}b \notin M\} \\
& \cup \ \{N_i{:}\neg b^\dagger \mid Q_i{:}b \in M\} \\
& \cup \ \{N_i{:}a^\dagger \mid Q_i{:}a \in M\}.
\end{aligned}
\tag{6}
$$

For each $i \in \{1, \ldots, n\}$ it holds that $(Q'_i+)^{M'} = \{l \leftarrow \alpha' \mid l \leftarrow \alpha \in Q_i^M\}$ with Q'_i+ the set of rules defined in (2) with $\alpha' = \{Q'_i{:}b \mid Q_i{:}b \in \alpha\}$.

Using this lemma, we can prove that M' as defined in Lemma 1 is indeed an answer set of the communicating simple program that simulates the communicating normal program $\mathcal{P}$ when M is an answer set of $\mathcal{P}$.

Proposition 2
Let $\mathcal{P} = \{Q_1, \ldots, Q_n\}$ and let $\mathcal{P}' = \{Q'_1, \ldots, Q'_n, N_1, \ldots, N_n\}$ with $\mathcal{P}$ a communicating normal program and $\mathcal{P}'$ the communicating simple program that simulates $\mathcal{P}$ as defined in Definition 2. If M is an answer set of $\mathcal{P}$, then M' is an answer set of $\mathcal{P}'$ with M' defined as in Lemma 1.

Next we introduce Lemma 2, which is similar to Lemma 1 in approach but which states the converse.

Lemma 2

Let $\mathcal{P} = \{Q_1, \ldots, Q_n\}$ and let $\mathcal{P}' = \{Q'_1, \ldots, Q'_n, N_1, \ldots, N_n\}$ with $\mathcal{P}$ a communicating normal program and $\mathcal{P}'$ the communicating simple program that simulates $\mathcal{P}$. Assume that M' is an answer set of $\mathcal{P}'$ and that $(M')_{N_i}$ is total *w.r.t.* $\mathcal{B}_{N_i}$ for all $i \in \{1, \ldots, n\}$. Let M be defined as

$$M = \left\{ Q_i\!:\!b \mid Q'_i\!:\!b \in \left((Q'_i+)^{M'} \right)^{\star} \right\} \tag{7}$$

For each $i \in \{1, \ldots, n\}$, it holds that $(Q'_i+)^{M'} = \{l \leftarrow \alpha' \mid l \leftarrow \alpha \in Q_i^M\}$ with $\alpha' = \{Q'_i\!:\!b \mid Q_i\!:\!b \in \alpha\}$.

Proposition 3

Let $\mathcal{P} = \{Q_1, \ldots, Q_n\}$ and let $\mathcal{P}' = \{Q'_1, \ldots, Q'_n, N_1, \ldots, N_n\}$ with $\mathcal{P}$ a communicating normal program and $\mathcal{P}'$ the communicating simple program that simulates $\mathcal{P}$. Assume that M' is an answer set of $\mathcal{P}'$ and that $(M')_{N_i}$ is total *w.r.t.* $\mathcal{B}_{N_i}$ for all $i \in \{1, \ldots, n\}$. Then the interpretation M defined in Lemma 2 is an answer set of $\mathcal{P}$.

It is important to note that Lemma 2 and, by consequence, Proposition 3 require (part of) the answer set M' to be total. This is a necessary requirement, as demonstrated by the following example.

Example 7

Consider the normal program $R = \{a \leftarrow not\ a\}$ which has no answer sets. The corresponding communicating simple program $\mathcal{P}' = \{Q', N\}$ has the following rules:

$$Q'\!:\!a \leftarrow N\!:\!\neg a^{\dagger} \qquad\qquad N\!:\!\neg a^{\dagger} \leftarrow Q'\!:\!\neg a^{\dagger}$$
$$Q'\!:\!\neg a^{\dagger} \leftarrow N\!:\!\neg a^{\dagger} \qquad\qquad N\!:\!a^{\dagger} \leftarrow Q'\!:\!a.$$

It is easy to see that $I = \emptyset$ is an answer set of $\mathcal{P}'$ since we have $Q'^I = N^I = \emptyset$. Notice that I does not correspond with an answer set of R, which is due to $I_N = \emptyset$ not being total and hence we cannot apply Proposition 3.

Regardless, it is easy to see that the requirement for the answer set to be total can be built into the simulation program. Indeed, it suffices to introduce additional rules to every N_i with $1 \leq i \leq n$ in the simulation defined in Definition 2. These rules are

$$\left\{ N_i\!:\!a \leftarrow N_i\!:\!a^{\dagger}, N_i\!:\!a \leftarrow N_i\!:\!\neg a^{\dagger} \mid a^{\dagger} \in \mathcal{B}_{N_i} \right\}$$
$$\cup \left\{ N_i\!:\!total \leftarrow \beta \right\} \text{ with } \beta = \left\{ N_i\!:\!a \mid a^{\dagger} \in \mathcal{B}_{N_i} \right\}.$$

Thus the requirement that (part of) the answer set must be total can be replaced by the requirement that the situated literals $N_i\!:\!total$ must be true in the answer set. Hence, if we want to check whether a literal $Q\!:\!l$ is true in at least one answer set of a (communicating) normal program, it suffices to check whether $Q\!:\!l$ and $N_i\!:\!total$ can be derived in the communicating simple program that simulates it. Clearly we find that brave reasoning for communicating simple programs is NP-hard.

What we have done so far is important for two reasons. First, we have shown that

the complexity of brave reasoning for communicating normal programs is no harder than brave reasoning for communicating simple programs. Indeed, the problem of brave reasoning for communicating normal programs can be reduced in polynomial time to the problem of brave reasoning for communicating simple programs. Second, since normal programs are a special case of communicating normal programs and since we know that brave reasoning for normal programs is an NP-complete problem, we have successfully shown that brave reasoning for communicating simple programs is NP-hard. In order to show that brave reasoning for communicating simple programs is NP-complete, we need to additionally show that this is a problem in NP. To this end, consider the following algorithm to find the answer sets of a communicating simple program $\mathcal{P}$:

> guess an interpretation $I \subseteq (\mathcal{B}_\mathcal{P} \cup \neg\mathcal{B}_\mathcal{P})$
> verify that this interpretation is an answer set as follows:
>   calculate the reduct Q^I of each component program Q
>   calculate the fixpoint of each simple component program Q^I
>   verify that $Q : I_Q = (Q^I)^\star$ for each component program Q

The first step of the algorithm requires a choice, hence the algorithm is non-deterministic. Next we determine whether this guess is indeed a communicating answer set, which involves taking the reduct, computing the fixpoint and verifying whether this fixpoint coincides with our guess. Clearly, verifying whether the interpretation is an answer set can be done in polynomial time and thus the algorithm to compute the answer sets of a communicating simple program is in NP, and thus NP-complete, regardless of the number of component programs. These same results hold for communicating normal programs since the reduct also removes all occurrences of negation-as-failure.

For communicating disjunctive programs it is easy to see that the Σ_2^P-completeness of classical disjunctive ASP carries over to communicating disjunctive programs. Cautious reasoning is then coNP and $\mathsf{co}\Sigma_2^\mathsf{P}$ for communicating normal programs and communicating disjunctive programs, respectively, since this decision problem is the complement of brave reasoning. Finally, the problem of answer set existence is carried over from normal programs and disjunctive programs (Baral 2003) and is NP-hard and $\mathsf{co}\Sigma_2^\mathsf{P}$-hard, respectively. Most of these complexity results correspond with classical ASP, with the results from communicating simple programs being notable exceptions; indeed, for communicating simple programs the communication aspect clearly has an influence on the complexity. Table 1 summarises the main complexity results.

To conclude, given a communicating normal program $\mathcal{P}$, there is an easy (linear) translation that transforms $\mathcal{P}$ into a normal program P' such that the answer sets of P' corresponds to the answer sets of $\mathcal{P}$.

Proposition 4

Let $\mathcal{P}$ be a communicating normal program. Let P' be the normal program defined

Table 1. Completeness results for the main reasoning tasks for a communicating program $\mathcal{P} = \{Q_1, \ldots, Q_n\}$

reasoning task → component programs ↓	answer set existence	brave reasoning	cautious reasoning
simple program	P	NP	coNP
normal program	NP	NP	coNP
disjunctive program	Σ_2^P	Σ_2^P	$co\Sigma_2^P$

as follows. For every $Q\!:\!a \in \mathcal{B}_\mathcal{P}$ we add the following rules to P':

$$guess(Q\_a) \leftarrow not\ not\_guess(Q\_a) \qquad not\_guess(Q\_a) \leftarrow not\ guess(Q\_a)$$
$$\leftarrow guess(Q\_a), not\ Q\_a \qquad\qquad \leftarrow not\_guess(Q\_a), Q\_a. \qquad (8)$$

Furthermore, for every normal communicating rule $\mathcal{P}$ which is of the form $r = Q\!:\!a \leftarrow body$, we add the rule $Q\_a \leftarrow body'$ to P' with Q_a a literal. We define $body'$, which is a set of extended literals, as follows:

$$body' = \{Q\_b \mid Q\!:\!b \in body\}$$
$$\cup \{guess(R\_c) \mid R\!:\!c \in body, Q \neq R\}$$
$$\cup \{not\ S\_d \mid (not\ S\!:\!d) \in body\}. \qquad (9)$$

We have that $M = \{Q\!:\!a \mid Q\_a \in M'\}$ is an answer set of $\mathcal{P}$ if and only if M' is an answer set of P.

Proof
(sketch) The essential difference between a normal program and a communicating program is in the reduct. More specifically, the difference is in the treatment of situated literals of the form $R\!:\!l$ which are not Q-local. Indeed, such literals can, like naf-literals, be guessed and verified whether or not they are stable, *i.e.* verified whether or not the minimal model of the reduct corresponds to the initial guess. It can readily be seen that this behaviour is mimicked by the rules in (8). The first two rules guess whether or not some situated literal $Q\!:\!a$ is true, while the last two rules ensure that our guess is stable; *i.e.* we are only allowed to guess $Q\!:\!a$ when we are later on actually capable of deriving $Q\!:\!a$. The purpose of (9) is then to ensure that guessing of situated literals is only used when the situated literal in question is not Q-local and is not preceded by negation-as-failure. $\square$

5 Multi-Focused Answer Sets

Answer set semantics are 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. One option is to assume global minimality, *i.e.* we minimise over the conclusions of all the agents in the network. This is the approach that

was taken in Section 4. 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 minimise over the agents matters, as the next example illustrates.

Example 8
An employee ('E') needs a new printer ('P'). She has a few choices (loud or silent, stylish or dull), preferring silent and stylish. Her manager ('M') insists that it is a silent printer. Her boss ('B') does not want an expensive printer, *i.e.* one that is both silent and stylish. We can consider the communicating normal program $\mathcal{P} = \{E, M, B\}$ with:

$$P\!:\!stylish \leftarrow not\ P\!:\!dull \qquad\qquad P\!:\!dull \leftarrow not\ P\!:\!stylish \qquad (10)$$

$$P\!:\!silent \leftarrow not\ P\!:\!loud \qquad\qquad P\!:\!loud \leftarrow not\ P\!:\!silent \qquad (11)$$

$$E\!:\!undesired \leftarrow P\!:\!dull \qquad\qquad E\!:\!undesired \leftarrow P\!:\!loud \qquad (12)$$

$$M\!:\!undesired \leftarrow P\!:\!loud \qquad\qquad\qquad\qquad (13)$$

$$B\!:\!expensive \leftarrow P\!:\!stylish, P\!:\!silent. \qquad\qquad\qquad (14)$$

The rules in (10) and (11) encode the four possible printers and the rules in (12), (13) and (14) encode the inclinations of the employee, manager and boss, respectively. The answer sets of this program, *i.e.* those with global minimality, are

$$M_1 = \{P\!:\!stylish, P\!:\!silent, B\!:\!expensive\}$$
$$M_2 = \{P\!:\!stylish, P\!:\!loud, E\!:\!undesired, M\!:\!undesired\}$$
$$M_3 = \{P\!:\!dull, P\!:\!loud, E\!:\!undesired, M\!:\!undesired\}$$
$$M_4 = \{P\!:\!dull, P\!:\!silent, E\!:\!undesired\}$$

The answer sets that are minimal for agent B are M_2, M_3 and M_4, *i.e.* the answer sets that do not contain $B\!:\!expensive$. The only answer set that is minimal for agent E is M_1, *i.e.* the one that does not contain $E\!:\!undesired$. Hence when we determine local minimality for communicating ASP, the order in which we consider the agents is important as it induces a priority over them, *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. Since the manager prefers a silent printer, when we first minimise over 'B' and then minimise over 'M' and 'E' (we may as well minimise over 'E' and then 'M', as 'E' and 'M' have no conflicting preferences) we end up with the unique answer set M_4.

In this section, we formalise such a communication mechanism. 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 3
Let $\mathcal{P}$ be a communicating normal 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 recursively as follows:

- M is a $(Q_1, \ldots, Q_n)$-focused answer set of $\mathcal{P}$ and there are no $(Q_1, \ldots, Q_{n-1})$-focused answer sets M' of $\mathcal{P}$ such that $M'_{Q_n} \subset M_{Q_n}$;

- 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 9
Consider the communicating normal program $\mathcal{P} = \{Q, R, S\}$ with the rules

$$Q\!:\!a \leftarrow \qquad\qquad R\!:\!b \leftarrow S\!:\!c \qquad\qquad S\!:\!a \leftarrow$$
$$Q\!:\!b \leftarrow not\ S\!:\!d \qquad R\!:\!a \leftarrow S\!:\!c \qquad\qquad S\!:\!c \leftarrow not\ S\!:\!d, not\ R\!:\!c$$
$$Q\!:\!c \leftarrow R\!:\!c \qquad\qquad R\!:\!a \leftarrow S\!:\!d \qquad\qquad S\!:\!c \leftarrow not\ S\!:\!c, not\ R\!:\!c$$
$$\qquad\qquad\qquad\quad R\!:\!c \leftarrow not\ R\!:\!a$$

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

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

The only (R, S)-focused answer set of $\mathcal{P}_9$ is M_1. Indeed, since $\{a\} = (M_3)_R \subset (M_2)_R = \{a, b\}$ we find that M_2 is not a (R)-focused answer set. Furthermore $\{a\} = (M_1)_S \subset (M_3)_S = \{a, d\}$, hence M_3 is not an (R, S)-focused answer set.

Proposition 5
Let $\mathcal{P}$ be a communicating simple program. We then have:

- there always exists at least one $(Q_1, ..., Q_n)$-focused answer set of $\mathcal{P}$;
- we can compute this $(Q_1, ..., Q_n)$-focused answer set in polynomial time.

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

Definition 4
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:

$$Q_0 = \{x \leftarrow not\ \neg x, \neg x \leftarrow not\ x \mid x \in X_1 \cup \ldots \cup X_n\} \qquad (15)$$
$$\cup\ \{sat \leftarrow Q_0\!:\!\theta_t \mid 1 \leq t \leq m\} \qquad\qquad\qquad\qquad (16)$$
$$\cup\ \{\neg sat \leftarrow not\ sat\}\,. \qquad\qquad\qquad\qquad\qquad\qquad (17)$$

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

$$Q_j = \{x \leftarrow Q_0 : x, \neg x \leftarrow Q_0 : \neg x \mid x \in (X_1 \cup \ldots \cup X_{n-j})\} \tag{18}$$

$$\cup \begin{cases} \{\neg sat \leftarrow Q_0 : \neg sat\} & \text{if } (n-j) \text{ is even} \\ \{sat \leftarrow Q_0 : sat\} & \text{if } (n-j) \text{ is odd.} \end{cases} \tag{19}$$

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

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 (19) are swapped.

Example 10
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 normal program $\mathcal{P}$ corresponding with the QBF ϕ is defined as follows:

$$Q_0 : x \leftarrow not \ \neg x \qquad Q_0 : y \leftarrow not \ \neg y \qquad Q_0 : z \leftarrow not \ \neg z$$
$$Q_0 : \neg x \leftarrow not \ x \qquad Q_0 : \neg y \leftarrow not \ y \qquad Q_0 : \neg z \leftarrow not \ z$$
$$Q_0 : sat \leftarrow x, y \qquad Q_0 : sat \leftarrow \neg x, y, z \qquad Q_0 : sat \leftarrow \neg x, \neg y, \neg z$$
$$Q_0 : \neg sat \leftarrow not \ sat$$

$$Q_1 : x \leftarrow Q_0 : x \qquad Q_1 : y \leftarrow Q_0 : y$$
$$Q_1 : \neg x \leftarrow Q_0 : \neg x \qquad Q_1 : \neg y \leftarrow Q_0 : \neg y \qquad Q_1 : \neg sat \leftarrow Q_0 : \neg sat$$

$$Q_2 : x \leftarrow Q_0 : x \qquad Q_2 : \neg x \leftarrow Q_0 : \neg x \qquad Q_2 : sat \leftarrow Q_0 : sat$$

The communicating normal program in Example 10 can be used to determine whether the QBF ϕ is valid. First, note that the rules in (15) generate all possible truth assignments of the variables, *i.e.* all possible propositional interpretations. The rules in (16) ensure that 'sat' is true exactly for those interpretations that satisfy the formula $p(X_1, X_2, \ldots, X_n)$.

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 (19), 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 of the corresponding program.

Proposition 6
Let ϕ and $\mathcal{P}$ be as in Definition 4. 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 $\mathcal{P}$.

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

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

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

Proposition 7
Let $\mathcal{P}$ be a communicating normal program with $Q_i \in \mathcal{P}$. The problem of deciding whether there exists a $(Q_1, \ldots, Q_n)$-focused answer set M of $\mathcal{P}$ such that $Q_i : l \in M$ (brave reasoning) is in $\Sigma^{\mathsf{P}}_{n+1}$.

Since cautious reasoning is the complementary problem of brave reasoning it readily follows that cautious reasoning is in $\mathsf{co}\Sigma^{\mathsf{P}}_{n=1}$. 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 with $Q_i \in \mathcal{P}$. 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 $\Sigma^{\mathsf{P}}_{n+1}$-complete.

The next corollary shows that the complexity remains the same when going from normal component programs to simple component programs.

Proposition 8
Let $\mathcal{P}$ be a communicating simple program with $Q_i \in \mathcal{P}$. The problem of deciding whether there exists a $(Q_1, \ldots, Q_n)$-focused answer set M of $\mathcal{P}$ such that $Q_i : l \in M$ (brave reasoning) is in $\Sigma^{\mathsf{P}}_{n+1}$.

Finally, we also have a result for communicating disjunctive programs instead of communicating normal programs.

Proposition 9
Let $\mathcal{P}$ be a communicating disjunctive program with $Q_i \in \mathcal{P}$. 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 in $\Sigma^{\mathsf{P}}_{n+2}$.

Table 2 summarises the membership results for brave reasoning that were discussed in this section.

Due to the extra expressiveness and complexity of multi-focused answer sets, it is clear that no translation to classical ASP is possible. Possible future implementations may, however, be based on a translation to other PSPACE complete problems such as QBF formulas or modal logics. A translation to QBF formulas seems to be the most natural, especially since the proof of the complexity of multi-focused

Table 2. Membership results for brave reasoning with (multi-focused) answer sets of the communicating program $\mathcal{P} = \{Q_1, \ldots, Q_n\}$

form of communication → type of component program ↓	none	situated literals	multi-focused
simple program	P	NP	Σ^P_{n+1}
normal program	NP	NP	Σ^P_{n+1}
disjunctive program	Σ^P_2	Σ^P_2	Σ^P_{n+2}

answer sets involves reducing QBF formulas to multi-focused answer sets. However, any such translation falls beyond the scope is this paper and is the subject of future research.

6 Case Study: subset-minimal abductive diagnosis

In this section we work out an example that highlights the usefulness of multi-focused answer sets. The use of multi-focused answer sets has already proven itself useful in modeling problems where one can use a negotiation paradigm, *e.g.* in (Bauters 2011). However, the main goal in (Bauters 2011) was to show that such a paradigm is possible, rather than actually trying to encode a problem that is known to be more complex that Σ^P_2. Though a lot of interesting problems are indeed in P, NP or Σ^P_2, there are still some important problems that are even higher up in the polynomial hierarchy. One such a problem is a special form of abductive diagonistics. An abductive diagnostic problem is encoded as a triple $\langle H, T, O \rangle$ (Eiter et al. 1999), where H is a set of atoms referred to as hypotheses, T is an ASP program referred to as the theory and O is a set of literals referred to as observations. Intuitively, the theory T describes the dynamics of the system, the observations O describe the observed state of the system and the hypotheses H try to explain these observations within the theory. The goal in subset-minimal abductive diagnosis is to find the minimal set of hypotheses that explain the observation. That is, we want to find the minimal set of hypotheses such that $O \subseteq M$ with M an answer set of $T \cup H$. Subset-minimal abductive diagnostics over a theory consisting of a disjunctive program is a problem in Σ^P_3 and hence we cannot rely on classical ASP to find the solutions to this problem. However, as we will see in the next example, we can easily solve this problem using multi-focused answer sets.

Example 11 (Adapted from (Eiter et al. 1999))
Consider an electronic circuit, as in Figure 2, where we have a power source, a control lamp, three hot-plates wired in parallel and a fuse to protect each hot-plate. It is known that some of the fuses are sensitive to high current and may consequently blow, but it is not known which fuses. Furthermore, plate A sits near a source of water (*e.g.* a tap). If water comes into contact with plate A, this causes a short circuit which blows the nearest fuse, *i.e.* fuse A, to prevent any damage.
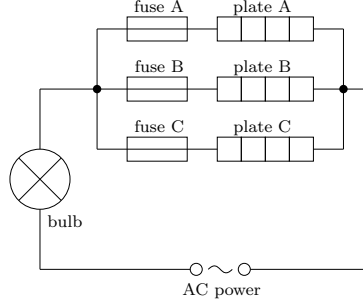
Fig. 2. Schematics of the electronic circuit we want to diagnose.

Upon inspection, we find that the control lamp is on and that plate A feels cold to the touch. We want to find the subset minimal diagnoses that would explain the problem, *i.e.* we want to find the minimal causes that can explain this situation.

First we need to describe the theory, *i.e.* the schematics. The theory describes the dynamics of the system and thus also how the system may fail. We can describe the theory as follows. For starters, a melted fuse can be caused by a high current, or, for fuse A, due to a hazardous water leak:

$$Q : melted\_A; Q : melted\_B; Q : melted\_C \leftarrow Q : high$$
$$Q : melted\_A \leftarrow Q : leak.$$

Furthermore, under a number of conditions the control light will be off:

$$Q : light\_off \leftarrow Q : power\_off$$
$$Q : light\_off \leftarrow Q : broken\_bulb$$
$$Q : light\_off \leftarrow Q : melted\_A, Q : melted\_B, Q : melted\_C.$$

Then we describe under what conditions each plate will be hot:

$$Q : hot\_plateA \leftarrow not\ Q : melted\_A, not\ Q : power\_off$$
$$Q : hot\_plateB \leftarrow not\ Q : melted\_B, not\ Q : power\_off$$
$$Q : hot\_plateC \leftarrow not\ Q : melted\_C, not\ Q : power\_off.$$

We now encode the hypotheses. We have a number of causes, each of which may by itself or in conjunction with other causes explain our observation. In total, we have four causes. The power can be off (*power_off*), the light bulb might be broken (*broken_bulb*), there may have been a high current (*high*) and/or a water leak may have occurred (*leak*). We describe all these hypotheses as follows:

$$Q : power\_off \leftarrow not\ Q : no\_power\_off$$
$$Q : no\_power\_off \leftarrow not\ Q : power\_off$$
$$Q : broken\_bulb \leftarrow not\ Q : no\_broken\_bulb$$
$$Q : no\_broken\_bulb \leftarrow not\ Q : broken\_bulb$$
$$Q : high \leftarrow not\ Q : no\_high$$

$$Q\!:\!no\_high \leftarrow not\ Q\!:\!high$$
$$Q\!:\!leak \leftarrow not\ Q\!:\!no\_leak$$
$$Q\!:\!no\_leak \leftarrow not\ Q\!:\!leak.$$

It is easy to see that these rules in Q encode all possible subsets of hypotheses that may have occurred. We then add rules to a separate component program H which merely relays the information on the set of hypotheses that we chose. The reason for this separate component program H is that we can now minimise over the set of hypotheses that is assumed, simply by focusing on H.

$$H\!:\!power\_off \leftarrow Q\!:\!power\_off$$
$$H\!:\!broken\_bulb \leftarrow Q\!:\!broken\_bulb$$
$$H\!:\!high\_current \leftarrow Q\!:\!high$$
$$H\!:\!water\_leak \leftarrow Q\!:\!leak$$

Finally, we model the observation. We observe that the control light is on and that plate A is cold. In other words, we obtain the rules (which encode constraints):

$$Q\!:\!contradiction \leftarrow not\ Q\!:\!contradiction, Q\!:\!light\_off$$
$$Q\!:\!contradiction \leftarrow not\ Q\!:\!contradiction, Q\!:\!hot\_plateA$$

which intuitively tell us that we cannot have that the light is off, nor can we have that plate A is hot.

The (H)-focused answer sets give us the subset minimal abductive diagnoses. It is easy to see that the focus on H is needed to minimise over the hypotheses. The program $\mathcal{P} = \{Q, H\}$ has two (H)-focused answer sets M_1 and M_2, both containing $M_{\text{shared}} = \{Q\!:\!no\_power\_off, Q\!:\!no\_broken\_bulb, Q\!:\!hot\_plateB, Q\!:\!hot\_plateC\}$:

$$M_1 = M_{\text{shared}} \cup \{Q\!:\!melted\_A, Q\!:\!no\_leak, Q\!:\!high, H\!:\!high\}$$
$$M_2 = M_{\text{shared}} \cup \{Q\!:\!melted\_A, Q\!:\!leak, Q\!:\!no\_high, H\!:\!leak\}.$$

Hence the minimal sets of hypotheses that support our observation, *i.e.* M_H with M an (H)-focused answer set, are that either there was a high current (which melted fuse A) or there was a water leak (which also melted fuse A).

7 Related Work

A large body of research has been devoted to combining logic programming with multi-agent or multi-context ideas, with various underlying reasons. One reason for such a combination is that the logic can be used to describe the (rational) behaviour of the agents in a multi-agent network, as in (Dell'Acqua et al. 1999). Alternately, it can be used to combine different flavours of logic programming languages (Luo et al. 2005; Eiter et al. 2008). It can also be used to externally solve tasks for which ASP is not suited, while remaining in a declarative framework (Eiter et al. 2006). As a final example, it can be used as a form of cooperation, where multiple agents or contexts collaborate to solve a difficult problem (De Vos et al. 2005; Van Nieuwenborgh et al. 2007). The approach in this paper falls in the last category

and is concerned with how the collaboration of different ASP programs affects 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 discuss some of the more prominent work in these areas.

The work of (Roelofsen and Serafini 2005) proposes an extension of multi-context systems or MCSs (Giunchiglia and Serafini 1994) that allows MCSs to reason about absent information, *i.e.* they introduce non-monotonicity in the context of MCSs. The idea of a MCS is that we have a number of contexts that each have access to only a subset of the available information. Each context has a local model and reasoning capabilities, but there is also an information flow defined by the system between the different contexts. It is this idea that was later adopted in the ASP community and in our paper in particular.

Our paper has a comparable syntax as (Roelofsen and Serafini 2005) but rather different semantics. The semantics in (Roelofsen and Serafini 2005) are closely related to the well-founded semantics (Gelder et al. 1991), while our semantics are closer in spirit to the stable model semantics (Gelfond and Lifzchitz 1988). Another point where our semantics differ is that we allow a restricted circular explanation of why a literal is true, if that circular explanation is due to our reliance on other component programs. This particular form of circular reasoning has been identified in (Buccafurri et al. 2008) as a requirement in the representation of social reasoning.

The work in (Brewka et al. 2007) further extends the work in (Roelofsen and Serafini 2005) and addresses a number of problems and deficiencies. The paper is, to the best of our knowledge, the first to offer a syntactical rather than semantical description of communication in multi-context systems, making it easier to implement an actual algorithm. A number of interesting applications of contextual frameworks, including information fusion, game theory and social choice theory are highlighted in the paper. Lastly, the paper identifies that the complexity of the main reasoning task is on the second level of the polynomial hierarchy.

Along similar lines (Brewka and Eiter 2007) combines the non-monotonicity from (Roelofsen and Serafini 2005) with the heterogeneous approach presented in (Giunchiglia and Serafini 1994) into a single framework for heterogenous non-monotonic multi-context reasoning. The work in (Brewka and Eiter 2007) introduces several notions of equilibria, including minimal and grounded equilibria. In our approach, local reasoning is captured by grounded equilibria (which does not allow circular explanations) while communicating with other component programs is captured by the weaker minimal equilibria. The work in (Brewka and Eiter 2007) offers various membership results on deciding the existence of an equilibrium and is one of the first to note that multi-context systems, due to the nature of the bridge rules/situated literals, can be non-monotonic even if all the logics in the component programs themselves are monotonic.

Work on a distributed solver for heterogenous multi-context systems commenced in (Dao-Tran et al. 2010). While solvers exist to compute multi-context systems locally, this is the first work to consider an algorithm which is both distributed (*i.e.* no shared memory) and modular (*i.e.* computation starting from partial mod-

els). When the context under consideration uses *e.g.* ASP, loop formulas can be devised which allow bridge rules to be compiled into local classical theories. It is then possible to use SAT solvers to compute the grounded equilibria of the heterogenous multi-context system. Later work in (Drescher et al. 2011) improved on the idea by offering a mechanism to identify and break symmetries (*i.e.* permutations of belief which result in identical knowledge). As such, the solver need never visit two points in the search space that are symmetric, thus potentially offering a considerable speedup. Experimental results show that the solution space can indeed be (significantly) compressed. A similar idea might be used to compute answer sets of a communicating ASP program in a distributed fashion. Indeed, such answer sets are closely related to the idea of minimal equilibria from (Brewka and Eiter 2007). A few modifications should nonetheless be made. For example, the Herbrand base needs to be redefined in a way that is safe in such a distributed setting, *e.g.* by only taking situated literals into account that occur in a given component program. Optimizations to the distributed algorithm also seem likely to be applicable to the setting of communicating ASP. On the other hand, it does not seem to be straightforward to extend these ideas to compute multi-focused answer sets in a distributed fashion.

One of the most recent extensions to multi-context systems are managed multi-context systems or mMCS (Brewka et al. 2011). Normally, bridge rules can only be used to pass along information which allows for *e.g.* selection and abstraction of information between contexts. In an mMCS, however, additional operations on knowledge bases can be freely defined. For example, operations may be defined that remove or revise information. Such operations are performed by the context itself, *i.e.* by the legacy system that is used such as ASP, but mMCS allow to cope with this additional functionality in a principled way. As one would expect, adding such complex operations increases the expressiveness of the resulting system considerably. Our work, on the other hand, only allows for information to be passed along. By varying the way that the communication works, we achieved a comparable expressiveness.

We now direct our attention to work done within the ASP community. The ideas presented in this paper are related to HEX programs (Eiter et al. 2005) in which ASP is extended with higher-order predicates and external atoms. These external atoms allow to exchange knowledge in a declarative way with external sources that may implement functionality which is inconvenient or impossible to encode using current answer set programming paradigms. Application-wise, HEX is mainly proposed as a tool for non-monotonic semantic web reasoning under the answer set semantics. Hence 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 (De Vos et al. 2005) and (Van Nieuwenborgh et al. 2007). In both (De Vos et al. 2005) and (Van Nieuwenborgh et al. 2007) a multi-agent system is developed in which multiple agents/component programs can communicate with each other. Most importantly from the point of view of our work, both approaches use ASP and have agents that are quite expressive in their own right. Indeed, in (De Vos et al. 2005) each agent

is an Ordered Choice Logic Program (OCLP) (Brain and De Vos 2003) and in (Van Nieuwenborgh et al. 2007) each agent uses the extended answer set semantics.

The framework introduced in (De Vos et al. 2005) is called LAIMA. Each of the agents is an OCLP. The agents can communicate with whoever they want and circular communication is allowed (where agent A tells something to agent B which tells something to A ...). However, only positive information can be shared and the authors do not look at the actual expressiveness of the framework. In (Van Nieuwenborgh et al. 2007) each agent uses the extended answer set semantics. The network is a linear "hierarchical" network (*i.e.* information only flows in one direction), yet they employ the idea of a failure feedback mechanism. Intuitively, a failure feedback mechanism allows the previous agent in a network to revise his conclusion when the conclusion 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 simple and normal 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 also mention (Dao-Tran et al. 2009) where recursive modular non-monotonic logic programs (MLP) under the ASP semantics are considered. The main difference between MLP and our work is that our communication mechanism 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 behaviour of the networks presented in (Dao-Tran et al. 2009). Our expressiveness results therefore do not directly apply to MLPs.

Finally, there is an interesting resemblance between multi-focused answer sets and the work on multi-level integer programming (Jeroslow 1985). In multi-level integer 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 minimise the set of literals they have to accept as being true, rather than a linear objective function. Though there is an intuitive link, further research is required to make this link between multi-focused answer sets and the work on multi-level integer programming explicit.

8 Conclusions

We have systematically studied the effect of adding communication to ASP in terms of expressiveness and computational complexity. We start from simple programs, *i.e.* definite programs extended with true negation. Determining whether a literal belongs to an answer set of a simple program is a problem in P. A network of these simple programs, which we call communicating simple programs, is however expres-

sive enough to simulate normal programs. In other words, determining whether a literal belongs to an answer set of a communicating simple program is NP-hard. Furthermore, communicating simple programs can also simulate communicating normal programs provided that the resulting answer sets are partially complete, thus showing that adding negation-as-failure to communicating simple programs does not further increase the expressiveness.

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, in addition to the expressiveness of the individual agents. Table 2 highlights the membership results for brave reasoning obtained in Section 5.

Appendix A Result proofs

Proposition 1
Let $\mathcal{P}$ be a communicating simple program. We then have that:

- there always exists at least one answer set of $\mathcal{P}$;
- there is always a unique answer set of $\mathcal{P}$ that is globally minimal;
- we can compute this unique globally minimal answer set in polynomial time.

Proof
We can easily generalise the immediate consequence operator for (classical) simple programs to the case of communicating simple programs. Specifically, the operator $T_{\mathcal{P}}$ is defined *w.r.t.* an interpretation I of $\mathcal{P}$ as

$$T_{\mathcal{P}}(I) = I \cup \{Q\,{:}\,l \mid (Q\,{:}\,l \leftarrow \alpha) \in Q, Q \in \mathcal{P}, \alpha \subseteq I\}$$

where α is a set of $\mathcal{P}$-situated literals. It is easy to see that this operator is monotone. Together with a result from (Tarski 1955) we know that this operator has a least fixpoint. We use $\mathcal{P}^{\star}$ to denote this fixpoint obtained by repeatedly applying $T_{\mathcal{P}}$ starting from the empty interpretation. Clearly, this fixpoint can be computed in polynomial time.

We need to verify that $\mathcal{P}^{\star}$ is indeed an answer set. Since $\mathcal{P}$ is a communicating simple program, we know that the reduct $Q^{\mathcal{P}^{\star}}$ will only remove rules that contain situated literals $R\,{:}\,l$ that are not Q-local with $R\,{:}\,l \notin \mathcal{P}^{\star}$. In other words, rules that are not applicable ($\alpha \not\subseteq \mathcal{P}^{\star}$) and that contain non-$Q$-local situated literals are removed. Furthermore, remaining situated literals of the form $R\,{:}\,l$ that are not Q-local (*i.e.* those where $R\,{:}\,l \in \mathcal{P}^{\star}$) are removed from the body of the remaining rules.

Hence the remaining rules are all Q-local. Notice that the operator $T_{\mathcal{P}}$ is clearly an extension of the operator T_Q. Indeed, for a component simple program Q' that is Q'-local it is easy to verify that if $(Q')^{\star} = M'$ then $((\mathcal{P}')^{\star})_{Q'} = M'$ with $\mathcal{P}' = \{Q'\}$. It then readily follows, since all rules are Q-local and therefore independent of all other component programs, that $\left(Q^{\mathcal{P}^{\star}}\right)^{\star} = (\mathcal{P}^{\star})_Q$ for all $Q \in \mathcal{P}$.

So far we found that an answer set exists and that it can be computed in polynomial time. All that remains to be shown is that this answer set is globally minimal. This trivially follows from the way we defined the operator $T_{\mathcal{P}}$ since it only makes true the information that is absolutely necessary, *i.e.* the information that follows directly from the facts in the communicating simple program. Hence this is the minimal amount of information that needs to be derived for a set of situated literals to be a model of the communicating simple program at hand and thus the fixpoint $\mathcal{P}^{\star}$ is the globally minimal answer set. $\quad\square$

Lemma 1
Let $\mathcal{P} = \{Q_1, \ldots, Q_n\}$ and let $\mathcal{P}' = \{Q'_1, \ldots, Q'_n, N_1, \ldots, N_n\}$ with $\mathcal{P}$ a communicating normal program and $\mathcal{P}'$ the communicating simple program that simulates $\mathcal{P}$ as defined in Definition 2. Let M be an answer set of $\mathcal{P}$ and let the interpretation M' be defined as:

$$\begin{aligned}
M' = \; & \{Q'_i : a \mid a \in M_{Q_i}, Q_i \in \mathcal{P}\} \\
& \cup \; \{Q'_i : \neg b^{\dagger} \mid b \notin M_{Q_i}, Q_i \in \mathcal{P}\} \\
& \cup \; \{N_i : \neg b^{\dagger} \mid b \notin M_{Q_i}, Q_i \in \mathcal{P}\} \\
& \cup \; \{N_i : a^{\dagger} \mid a \in M_{Q_i}, Q_i \in \mathcal{P}\}.
\end{aligned} \tag{A1}$$

For each $i \in \{1, \ldots, n\}$ it holds that $(Q'_i+)^{M'} = \{l \leftarrow \alpha' \mid l \leftarrow \alpha \in Q_i^M\}$ with Q'_i+ the set of rules defined in (2) with $\alpha' = \{Q'_i : b \mid Q_i : b \in \alpha\}$.

Proof
To prove this, we first show that any rule of the form $(l \leftarrow \alpha) \in Q_i^M$ reappears in $(Q'_i+)^{M'}$ under the form $(l \leftarrow \alpha')$ for any $i \in \{1, \ldots, n\}$. The second step, showing that the converse also holds, can then be done in an analogous way.

Suppose $(l \leftarrow \alpha) \in Q_i^M$ for some $i \in \{1, \ldots, n\}$. By the definition of the reduct we know that there is some rule of the form $(l \leftarrow \alpha \cup \mathit{not}\ \beta \cup \gamma) \in Q_i$ such that $\beta \cap M = \emptyset$ and $\gamma \subseteq M$ is a set of situated literals of the form $Q_j : d$ with $i \neq j$, $1 \leq j \leq n$. From Definition 2, we know that the communicating normal rule $(Q_i : l \leftarrow \alpha \cup \mathit{not}\ \beta \cup \gamma)$ is transformed into the rule $(Q'_i : l \leftarrow \alpha' \cup \beta' \cup \gamma')$ with $\alpha' = \{Q'_i : b \mid Q_i : b \in \alpha\}$, $\beta' = \{N_k : \neg c^{\dagger} \mid Q_k : c \in \beta, k \in \{1, \ldots, n\}\}$ and $\gamma' = \{Q'_j : d \mid Q_j : d \in \gamma, j \in \{1, \ldots, n\}\}$. We show that, indeed, $(l \leftarrow \alpha) \in Q_i^M$ reappears in $(Q'_i+)^{M'}$ under the form $(l \leftarrow \alpha')$.

First, whenever $Q_k : c \in \beta$, we know that $Q_k : c \notin M$ since $\beta \cap M = \emptyset$. From the construction of M' we have that $N_k : \neg c^{\dagger} \in M'$. Similarly, since $\gamma \subseteq M$ we know from the construction of M' that $Q'_j : d \in M'$ whenever $Q_j : d \in \gamma$. Hence when determining the reduct $(Q'_i+)^{M'}$, the extended situated literals in β' and γ' will be deleted.

Finally, whenever $\alpha \cap M \neq \emptyset$ we know from the construction of M' that $Q_i':b \in M'$ whenever $Q_i:b \in \alpha$. Clearly, when determining the reduct, none of these extended situated literals will be deleted as they are Q_i'-local. Hence it is clear that the reduct of the communicating rule $(Q_i':l \leftarrow \alpha' \cup \beta' \cup \gamma')$ is the rule $Q_i':l \leftarrow \alpha'$. This completes the first part of the proof. As indicated, the second part of the proof is completely analogous. $\square$

Proposition 2
Let $\mathcal{P} = \{Q_1, \ldots, Q_n\}$ and let $\mathcal{P}' = \{Q_1', \ldots, Q_n', N_1, \ldots, N_n\}$ with $\mathcal{P}$ a communicating normal program and $\mathcal{P}'$ the communicating simple program that simulates $\mathcal{P}$ as defined in Definition 2. If M is an answer set of $\mathcal{P}$, then M' is an answer set of $\mathcal{P}'$ with M' defined as in Lemma 1.

Proof
This proof is divided into two parts. In part 1 we only consider the component programs Q_i' with $i \in \{1, \ldots, n\}$ and show that $\left((Q_i')^{M'}\right)^{\star} = (M')_{Q_i'}$. In part 2 we do the same, but we only consider the component programs N_i with $i \in \{1, \ldots, n\}$. As per Definition 1 we have then shown that M' is indeed an answer set of $\mathcal{P}'$.

Consider a component program Q_i' with $i \in \{1, \ldots, n\}$. By Definition 2 we have that $Q_i' = (Q_i'+) \cup (Q_i'-)$ and thus

$$(Q_i')^{M'} = (Q_i'+)^{M'} \cup (Q_i'-)^{M'}. \tag{A2}$$

For $Q_i'-$ we know by construction that it only contains rules that are of the form $(Q_i':\neg b^{\dagger} \leftarrow N_i:\neg b^{\dagger})$ and that the only rules of this form are in $Q_i'-$. Therefore, due to the definition of the reduct, we have

$$(Q_i'-)^{M'} = \left\{\neg b^{\dagger} \leftarrow \mid N_i:\neg b^{\dagger} \in M'\right\}$$

and because of the construction of M', see (A1), we obtain

$$(Q_i'-)^{M'} = \left\{\neg b^{\dagger} \leftarrow \mid b \notin M_{Q_i}\right\}. \tag{A3}$$

Hence $(Q_i'-)^{M'}$ only contains facts about literals that, by construction of Q_i', do not occur in $Q_i'+$. This means that from (A2) and (A3) we obtain

$$\left((Q_i')^{M'}\right)^{\star} = \left((Q_i'+)^{M'}\right)^{\star} \cup \left\{\neg b^{\dagger} \leftarrow \mid b \notin M_{Q_i}\right\}. \tag{A4}$$

From Lemma 1 we know that $(Q_i'+)^{M'} = \{l \leftarrow \alpha' \mid l \leftarrow \alpha \in Q_i^M\}$ where $\alpha' = \{Q_i':b \mid Q_i:b \in \alpha\}$. Because of the definition of an answer set of a communicating program we have

$$M_{Q_i} = \left(Q_i^M\right)^{\star} = \left((Q_i'+)^{M'}\right)^{\star}. \tag{A5}$$

Combining this with (A4) we get

$$\left((Q_i')^{M'}\right)^{\star} = M_{Q_i} \cup \left\{\neg b^{\dagger} \mid b \notin M_{Q_i}\right\}$$

$$= (M')_{Q_i'} \qquad \text{(definition of } M', \text{ see (A1))}$$

This concludes the first part of the proof.

In the second part of the proof, we only consider the component programs N_i' with $i \in \{1, \ldots, n\}$. By construction of N_i we know that all the rules of the form $\neg b^\dagger \leftarrow Q_i': \neg b^\dagger$ and $b^\dagger \leftarrow Q_i': b$ are in N_i and that all the rules in N_i are of this form. We have

$$(N_i)^{M'} = \{\neg b^\dagger \leftarrow \;|\; Q_i': \neg b^\dagger \in M'\} \cup \{b^\dagger \leftarrow \;|\; Q_i': b \in M'\}$$

which, due to the definition of M' can be written as

$$= \{\neg b^\dagger \leftarrow \;|\; b \notin M_{Q_i}\} \cup \{b^\dagger \leftarrow \;|\; b \in M_{Q_i}\}$$

from which it follows that $\left((N_i)^{M'}\right)^\star = (M')_{N_i}$. $\square$

Lemma 2

Let $\mathcal{P} = \{Q_1, \ldots, Q_n\}$ and let $\mathcal{P}' = \{Q_1', \ldots, Q_n', N_1, \ldots, N_n\}$ with $\mathcal{P}$ a communicating normal program and $\mathcal{P}'$ the communicating simple program that simulates $\mathcal{P}$. Assume that M' is an answer set of $\mathcal{P}'$ and that $(M')_{N_i}$ is total *w.r.t.* $\mathcal{B}_{N_i}$ for all $i \in \{1, \ldots, n\}$. Let M be defined as

$$M = \left\{Q_i: b \;\middle|\; Q_i': b \in \left((Q_i'+)^{M'}\right)^\star\right\} \tag{A6}$$

For each $i \in \{1, \ldots, n\}$, it holds that $(Q_i'+)^{M'} = \{l \leftarrow \alpha' \;|\; l \leftarrow \alpha \in Q_i^M\}$ with $\alpha' = \{Q_i': b \;|\; Q_i: b \in \alpha\}$.

Proof

To prove this, we first show that any rule of the form $(l \leftarrow \alpha') \in (Q_i'+)^{M'}$ reappears in Q_i^M under the form $l \leftarrow \alpha$ for any $i \in \{1, \ldots, n\}$. We then show that the converse also holds, which is rather analogous to the proof of the first part of Lemma 1. Due to some technical subtleties in the second part of the proof, however, we present the proof in detail.

Suppose $(l \leftarrow \alpha') \in (Q_i'+)^{M'}$. By the definition of the reduct of a communicating simple program we know that there is some communicating simple rule of the form $(l \leftarrow \alpha' \cup \beta' \cup \gamma') \in Q_i'+$ such that $\beta' \subseteq M'$ is a set of situated literals of the form $N_k: \neg c^\dagger$ and $\gamma' \subseteq M'$ is a set of situated literals of the form $Q_j': d$ with $i \neq j$ and $1 \leq j, k \leq n$.

From the definition of $Q_i'+$, we know that $(Q_i': l \leftarrow \alpha' \cup \beta' \cup \gamma')$ corresponds to a rule $(l \leftarrow \alpha \cup not\ \beta \cup \gamma) \in Q_i$ where we have that $\alpha = \{Q_i: b \;|\; Q_i': b \in \alpha'\}$, $\beta = \{Q_k: c \;|\; N_k: \neg c^\dagger \in \beta'\}$ and $\gamma = \{Q_j: d \;|\; Q_j': d \in \gamma'\}$. We show that, indeed, $(l \leftarrow \alpha') \in (Q_i'+)^{M'}$ reappears in Q_i^M under the form $(l \leftarrow \alpha)$.

First, since $\beta' \subseteq M'$, whenever $N_k: \neg c^\dagger \in \beta'$ we know that $N_k: \neg c^\dagger \in M'$. Since M' is a model (indeed, it is an answer set) it is an interpretation (and thus consistent). Therefore, if $N_k: \neg c^\dagger \in M'$ then surely $N_k: c^\dagger \notin M'$. Now, if we were to have $Q_k': c \in M'$, then applying the immediate consequence operator on the rule $N_k: c^\dagger \leftarrow Q_k': c$ found in the component program N_k would force us to have $N_k: c^\dagger \in M'$ which results in a contradiction. Hence we find that $Q_k': c \notin M'$.

By Definition 2 we know that $Q'_k = (Q'_k+) \cup (Q'_k-)$ and thus, by the definition of the reduct, we know that $(Q'_k)^{M'} = (Q'_k+)^{M'} \cup (Q'_k-)^{M'}$. Then we find that $((Q'_k)^{M'})^\star = ((Q'_k+)^{M'})^\star \cup ((Q'_k-)^{M'})^\star$ since all the rules in Q'_k- have fresh literals in the head and literals from N_k in the body and hence cannot interact with the rules from Q'_k+ which only depend on information derived from Q'_k+ and N_k in their bodies. Recall from the definition of an answer set of a communicating program that $\forall Q'_k \in \mathcal{P}' \cdot (Q'_k : M'_{Q'_k}) = \left((Q'_k)^{M'} \right)^\star$. Since we already found that $Q'_k : c \notin M'$ we must have $Q'_k : c \notin \left((Q'_i+)^{M'} \right)^\star$, or, because of the definition of M, that $Q_k : c \notin M$. Hence when determining the reduct $(l \leftarrow \alpha \cup not\ \beta \cup \gamma)^M$, the extended situated literals in $not\ \beta$ will be deleted.

In a similar way of reasoning, since $\gamma' \subseteq M'$ and because $\gamma = \{Q_j : d \mid Q'_j : d \in \gamma'\}$ we know from the construction of M that $\gamma \subseteq M$. Hence when determining the reduct, the situated literals in γ will be deleted. Finally, since $\alpha' \subseteq M'$ and because $\{Q_i : b \mid Q'_i : b \in \alpha'\} \subseteq M$ we know from the construction of M that $\alpha \in M$. Clearly, when determining the reduct, none of the situated literals in α will be deleted as they are Q_i-local. Hence the reduct of the communicating rule $(Q_i : l \leftarrow \alpha \cup not\ \beta \cup \gamma)$ is the rule $Q_i : l \leftarrow \alpha$. This completes the first part of the proof.

We now come to the second part. This time we show that any rule of the form $(l \leftarrow \alpha) \in Q_i^M$ reappears in $(Q'_i+)^{M'}$ under the form $(l \leftarrow \alpha')$ for any $i \in \{1, \ldots, n\}$.

Suppose $(l \leftarrow \alpha) \in Q_i^M$ for some $i \in \{1, \ldots, n\}$. By the definition of the reduct we know that there is some rule of the form $(l \leftarrow \alpha \cup not\ \beta \cup \gamma) \in Q_i$ such that $\beta \cap M = \emptyset$ and $\gamma \subseteq M$ is a set of situated literals of the form $Q_j : d$ with $i \neq j$, $1 \leq j \leq n$. From Definition 2, we know that the communicating normal rule $(Q_i : l \leftarrow \alpha \cup not\ \beta \cup \gamma)$ is transformed into the rule $(Q'_i : l \leftarrow \alpha' \cup \beta' \cup \gamma')$ with $\alpha' = \{Q'_i : b \mid Q_i : b \in \alpha\}$, $\beta' = \{N_k : \neg c^\dagger \mid Q_k : c \in \beta, k \in \{1, \ldots, n\}\}$ and $\gamma' = \{Q'_j : d \mid Q_j : d \in \gamma, j \in \{1, \ldots, n\}\}$. We show that, indeed, $(l \leftarrow \alpha) \in Q_i^M$ reappears in $(Q'_i+)^{M'}$ under the form $(l \leftarrow \alpha')$ when M'_{N_i} is total w.r.t. $\mathcal{B}_{N_i}$ for all $i \in \{1, \ldots, n\}$.

First, since $\gamma \subseteq M$ we know from the construction of M that $Q_j : d \in M$ whenever $Q'_j : d \in \gamma'$. Also, when $Q_k : c \in \beta$, we know that $Q_k : c \notin M$ since $\beta \cap M = \emptyset$. From the construction of M we then know that $Q'_k : c \notin M'$ and since M' is an answer set we readily obtain that $N_k : c^\dagger \notin M'$ due to the construction of N_k. Together with the requirement that M'_{N_k} is total w.r.t. $\mathcal{B}_{N_k}$ we then must have that $N_k : \neg c^\dagger \in M'$. Hence when determining the reduct $(Q'_i+)^{M'}$, the extended situated literals in β' and γ' will be deleted.

Finally, whenever $\alpha \cap M \neq \emptyset$ we know from the construction of M' that $Q'_i : b \in M'$ whenever $Q_i : b \in \alpha$. Clearly, when determining the reduct, none of these extended situated literals will be deleted as they are Q'_i-local. Hence it is clear that the reduct of the communicating rule $(Q'_i : l \leftarrow \alpha' \cup \beta' \cup \gamma')$ is the rule $Q'_i : l \leftarrow \alpha'$. This completes the second part of the proof.

$\square$

Proposition 3
Let $\mathcal{P} = \{Q_1, \ldots, Q_n\}$ and let $\mathcal{P}' = \{Q'_1, \ldots, Q'_n, N_1, \ldots, N_n\}$ with $\mathcal{P}$ a communi-

cating normal program and $\mathcal{P}'$ the communicating simple program that simulates $\mathcal{P}$. Assume that M' is an answer set of $\mathcal{P}'$ and that $(M')_{N_i}$ is total *w.r.t.* $\mathcal{B}_{N_i}$ for all $i \in \{1, \ldots, n\}$. Then the interpretation M defined in Lemma 2 is an answer set of $\mathcal{P}$.

Proof

Lemma 2 tells us that $(Q_i'+)^{M'} = \{l \leftarrow \alpha' \mid l \leftarrow \alpha \in Q_i^M\}$ where we have $\alpha' = \{Q_i' : b \mid Q_i : b \in \alpha\}$. Hence we have $\left((Q_i'+)^{M'}\right)^{\star} = \left(Q_i^M\right)^{\star}$ since repeatedly applying the immediate consequence operator must conclude the same literals l due to the correspondence of the rules in the reducts and because of the way α' is defined. Since we defined M as

$$\left\{ Q_i : b \mid Q_i' : b \in \left((Q_i'+)^{M'}\right)^{\star} \right\}$$

it follows immediately that M is an answer set of $\mathcal{P}$ since

$$\forall i \in \{1, \ldots, n\} \cdot \left(Q_i^M\right)^{\star} = M_{Q_i} \tag{A7}$$

which completes the proof. $\quad\square$

Proposition 5

Let $\mathcal{P}$ be a communicating simple program. We then have:

- there always exists at least one $(Q_1, ..., Q_n)$-focused answer set of $\mathcal{P}$;
- we can compute this $(Q_1, ..., Q_n)$-focused answer set in polynomial time.

Proof

We know from Proposition 1 that we can always find a globally minimal answer of $\mathcal{P}$ in polynomial time. Due to the way we defined the immediate fixpoint operator $T_{\mathcal{P}}$ this operator only makes true the information that is absolutely necessary, *i.e.* the minimal amount of information that can be derived (for each component program). It is then easy to see that no component program can derive any less information (we have no negation-as-failure) and thus that this globally minimal answer set is also locally minimal and thus a $(Q_1, ..., Q_n)$-focused answer set of $\mathcal{P}$. $\quad\square$

Proposition 6

Let ϕ and $\mathcal{P}$ be as in Definition 4. 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 4. 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 $\mathcal{P}' = \{Q'_0, \ldots, Q'_{n-2}\}$ 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 $M'_{Q_1} \subset M_{Q_1}$ 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, $M'_{Q_1} \subset M_{Q_1}$ 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 $M'_{Q_1} \subset M_{Q_1}$. 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 $M'_{Q_1} \subset M_{Q_1}$ 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 $M'_{Q_1} \subset M_{Q_1}$ 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 $M'_{Q_1} \subset M_{Q_1}$ 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 $M'_{Q_1} \subset M_{Q_1}$. 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 $M'_{Q_1} \subset M_{Q_1}$ 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 $\mathcal{P}$. $\quad\square$

Proposition 7
Let $\mathcal{P}$ be a communicating normal program with $Q_i \in \mathcal{P}$. The problem of deciding whether there exists a $(Q_1, \ldots, Q_n)$-focused answer set M of $\mathcal{P}$ such that $Q_i{:}l \in M$ (brave reasoning) is in $\Sigma^{\mathsf{P}}_{n+1}$.

Proof
We show the proof by induction on n. In the case where $n = 1$, we need to guess a (Q_1)-focused answer set M of $\mathcal{P}$ which can clearly be done in polynomial time. We now need to verify that this is indeed a (Q_1)-focused answer set which is a problem in coNP. Indeed, verifying that M is not a (Q_1)-focused answer set can be done using the following procedure in NP:

- guess an interpretation M'
- verify that M' is an answer set of $\mathcal{P}$
- verify that $M'_{Q_1} \subset M_{Q_1}$.

Hence, to find a (Q_1)-focused answer set, we guess an interpretation, verify that it is an answer set in polynomial time, and we subsequently use an NP oracle to decide whether this answer set is (Q_1)-focused, *i.e.* the problem is in Σ_2^P. Assume that there exists an algorithm to compute the $(Q_1, \ldots, Q_{n-1})$-focused answer sets of $\mathcal{P}$ that is in Σ_n^P. In a similar fashion, 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 $M'_{Q_n} \subset M_{Q_n}$ using a Σ_n^P oracle, *i.e.* the algorithm is in Σ_{n+1}^P. $\quad\square$

Proposition 8
Let $\mathcal{P}$ be a communicating simple program with $Q_i \in \mathcal{P}$. The problem of deciding whether there exists a $(Q_1, \ldots, Q_n)$-focused answer set M of $\mathcal{P}$ such that $Q_i{:}l \in M$ (brave reasoning) is in Σ_{n+1}^P.

Proof
We know from Proposition 2 that one normal program can be simulated by a communicating simple program with two component programs. Since only the program Q_0 in the simulation in Definition 4 includes negation-as-failure, it suffices to add a single simple component program in order to simulate the negation-as-failure. Since the number of component programs is of no importance in Proposition 7, the result readily follows. $\quad\square$

Proposition 9
Let $\mathcal{P}$ be a communicating disjunctive program with $Q_i \in \mathcal{P}$. 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 in Σ_{n+2}^P.

Proof
This result can easily be verified by looking at the proof of Proposition 7 and noticing that the only part of the algorithm that is affected by the use of communicating disjunctive programs $\mathcal{P}$ is the base step. In this base step, we use an oracle in NP to check whether our guess M is indeed an answer set of $\mathcal{P}$. Since M is an answer set of $\mathcal{P}$ iff $\forall i \in \{1, \ldots, n\} \cdot (Q_i^M)^\star = M_{Q_i}$ and since Q_i is a disjunctive component program we know that we will instead need an oracle in Σ_2^P to deal with communicative disjunctive programs. The remainder of the algorithm sketch remains unaffected. $\square$

References

BARAL, C. 2003. *Knowledge, Representation, Reasoning and Declarative Problem Solving.* Cambridge University Press.

BAUTERS, K. 2011. Modeling coalition formation using multi-focused answer sets. In *Proceedings of ESSLLI'11 Student Session.*

BAUTERS, K., JANSSEN, J., SCHOCKAERT, S., VERMEIR, D., AND DE COCK, M. 2010. Communicating answer set programs. In *Technical Communications of ICLP 2010.* Vol. 7. 34–43.

BAUTERS, K., SCHOCKAERT, S., VERMEIR, D., AND DE COCK, M. 2011. Communicating ASP and the polynomial hierarchy. To appear in Proceedings of LPNMR-11.

BRAIN, M. AND DE VOS, M. 2003. Implementing OCLP as a front-end for answer set solvers: From theory to practice. In *Proceedings of ASP'05*.

BREWKA, G. AND EITER, T. 2007. Equilibria in heterogeneous nonmonotonic multi-context systems. In *Proceedings of AAAI'07*. 385–390.

BREWKA, G., EITER, T., FINK, M., AND WEINZIERL, A. 2011. Managed multi-context systems. In *Proceedings of IJCAI'11*. 786–791.

BREWKA, G., ROELOFSEN, F., AND SERAFINI, L. 2007. Contextual default reasoning. In *Proceedings of IJCAI'07*. 268–273.

BUCCAFURRI, F., CAMINITI, G., AND LAURENDI, R. 2008. A logic language with stable model semantics for social reasoning. In *Proceedings of ICLP'08*. 718–723.

BYLANDER, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence 69*, 165–204.

DAO-TRAN, M., EITER, T., FINK, M., AND KRENNWALLNER, T. 2009. Modular non-monotonic logic programming revisited. In *Proceedings of ICLP'09*. 145–159.

DAO-TRAN, M., EITER, T., FINK, M., AND KRENNWALLNER, T. 2010. Distributed non-monotonic multi-context systems. In *Proceedings of KR'10*.

DE VOS, M., CRICK, T., PADGET, J., BRAIN, M., CLIFFE, O., AND NEEDHAM, J. 2005. LAIMA: A multi-agent platform using ordered choice logic programming. In *Proceedings of DALT'05*. 72–88.

DELL'ACQUA, P., SADRI, F., AND TONI, F. 1999. Communicating agents. In *Proceedings of MAS'99*.

DRESCHER, C., EITER, T., FINK, M., KRENNWALLNER, T., AND WALSH, T. 2011. Symmetry breaking for distributed multi-context systems. In *Proceedings of ICLP'10*. Lecture Notes in Computer Science, vol. 6645. 26–39.

EITER, T., FABER, W., LEONE, N., AND PFEIFER, G. 1999. The diagnosis frontend of the dlv system. *AI Communications 12*, 99–111.

EITER, T. AND GOTTLOB, G. 1995. The complexity of logic-based abduction. *ACM 42*, 3–42.

EITER, T., IANNI, G., LUKASIEWICZ, T., SCHINDLAUER, R., AND TOMPITS, H. 2008. Combining answer set programming with description logics for the semantic web. *Artifial Intelligence 172,* 12–13, 1495–1539.

EITER, T., IANNI, G., SCHINDLAUER, R., AND TOMPITS, H. 2005. A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In *Proceedings of IJCAI'05*. 90–96.

EITER, T., IANNI, G., SCHINDLAUER, R., AND TOMPITS, H. 2006. dlvhex: A tool for semantic-web reasoning under the answer-set semantics. In *Proceedings of International Workshop on Proceedings of ALPSWS'06*. 33–39.

GEBSER, M., GUZIOLOWSKI, C., IVANCHEV, M., SCHAUB, T., SIEGEL, A., THIELE, S., AND VEBER, P. 2010. Repair and prediction (under inconsistency) in large biological networks with answer set programming. In *Proceedings of KR'10*.

GELDER, A. V., ROSS, K. A., AND SCHLIPF, J. S. 1991. The well-founded semantics for general logic programs. *Journal of the ACM 38,* 3, 620–650.

GELFOND, M. AND LIFSCHITZ, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing 9*, 365–385.

GELFOND, M. AND LIFZCHITZ, V. 1988. The stable model semantics for logic programming. In *Proceedings of ICLP'88*. 1081–1086.

GIUNCHIGLIA, F. AND SERAFINI, L. 1994. Multilanguage hierarchical logics or: How we can do without modal logics. *Artifial Intelligence 65,* 1, 29–70.

Jeroslow, R. 1985. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical Programming 32*, 146–164.

Lifschitz, V. 2002. Answer set programming and plan generation. *Artificial Intelligence 138*, 39–54.

Lifschitz, V., Tang, L. R., and Turner, H. 1999. Nested expressions in logic programs. *Ann. Math. Artif. Intell. 25,* 3-4, 369–389.

Luo, J., Shi, Z., Wang, M., and Huang, H. 2005. Multi-agent cooperation: A description logic view. In *Proceedings of PRIMA'05*. 365–379.

Niemelä, I. and Simons, P. 2000. Extending the smodels system with cardinality and weight constraints. In *Logic-Based Artificial Intelligence*. Kluwer Academic Publishers, 491–521.

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

Roelofsen, F. and Serafini, L. 2005. Minimal and absent information in contexts. In *Proceedings of IJCAI'05*. 558–563.

Tarski, A. 1955. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics 5,* 2, 285–309.

Van Nieuwenborgh, D., De Vos, M., Heymans, S., and Vermeir, D. 2007. Hierarchical decision making in multi-agent systems using answer set programming. In *Proceedings of CLIMA'07*.