A Revised Concept of Safety for General Answer Set Programs

Pedro Cabalar[1], David Pearce[2*], and Agustín Valverde[3⋆]

[1] Universidade da Coruña
cabalar@udc.es
[2] Universidad Politécnica de Madrid, Spain
david.pearce@upm.es
[3] Universidad de Málaga, Málaga, Spain
a_valverde@ctima.uma.es

Abstract. To ensure a close relation between the answer sets of a program and those of its ground version, some answer set solvers deal with variables by requiring a safety condition on program rules. If we go beyond the syntax of disjunctive programs, for instance by allowing rules with nested expressions, or perhaps even arbitrary first-order formulas, new definitions of safety are required. In this paper we consider a new concept of safety for formulas in quantified equilibrium logic where answer sets can be defined for arbitrary first-order formulas. The new concept captures and generalises two recently proposed safety definitions: that of Lee, Lifschitz and Palla (2008) as well as that of Bria, Faber and Leone (2008). We study the main metalogical properties of safe formulas.

1 Introduction

In answer set programming (ASP) more and more tools are being created to enable a fuller use of first-order languages and logic. Recent developments include efforts to extend the syntax of programs as well as to deal more directly with variables in a full, first-order context. In several cases, assumptions such as standard names (SNA) are being relaxed and issues involving programming in open domains are being addressed.

A stable model semantics for first-order structures and languages was defined in the framework of equilibrium logic in [17–19]. As a logical basis the non-classical logic of quantified here-and-there, **QHT**, is used (see also [15]). By expanding the language to include new predicates, this logic can be embedded in classical first-order logic, [20], and this permits an alternative but equivalent formulation of the concept of stable model for first-order formulas, expressed in terms of classical, second-order logic [7]. The latter definition of answer set has been further studied in [11, 12] where the basis of a first-order programming language, RASPL-1, is described. An alternative approach to a first-order ASP language is developed in [8].

Several implementations of ASP deal with variables by requiring a safety condition on program rules. In the language of disjunctive LPs, this condition is usually expressed

⋆ Partially supported by the MEC (now MICINN) projects TIN2006-15455-(C01,C02,C03) and CSD2007-00022, and the Junta de Andalucia project P6-FQM-02049.

by saying that a rule is safe if any variable in the rule also appears in its positive body – this condition will be referred here as *DLP safety*. Programs are safe if all their rules are safe. The safety of a program ensures that its answer sets coincide with the answer sets of its ground version and thus allows ASP systems to be based on computations at the level of propositional logic which may include for example the use of SAT-solvers.

What if we go beyond the syntax of disjunctive programs? Adding negation in the heads of program rules will not require a change in the definition of safety. But for more far reaching language extensions, such as allowing rules with nested expressions, or perhaps even arbitrary first-order formulas, new definitions of safety are required. Again the desideratum is that safe programs should be in a certain sense reducible to their ground versions, whether or not these can be compiled into an existing ASP-solver.

Recently, there have been two new extensions of the notion of safety to accommodate different syntactic extensions of the ASP language in the first-order case. In [2], Bria, Faber and Leone introduce an extension to a restricted variant of the language of programs with nested expressions (or nested programs, for short) [16]. The rules are called *normal form nested* or NFN rules. The safety of such rules which will be referred here as *BFL-safety* is designed to guarantee domain independence and permit their efficient translation into the language of disjunctive programs. In this manner, NFN programs can be compiled into the standard DLV system. A second example is the definition of safety introduced by Lee, Lifschitz and Palla in [12] (we will call it here *LLP-safety*) defined essentially for arbitrary first-order formulas in such a way that the stable model of such a formula can be obtained by considering its ground version. It is also shown that the stable models of safe formulas form a (first-order) definable class. Here the motivation is not so much to compile safe programs into existing ASP languages, but rather to find tractable, first-order extensions of the customary languages.[4]

On the face of it, the approach followed in [12] is much more general than that of [2] and so we might expect that the safety of general formulas from [12] embraces also the safe NFN rules of [2]. But this is not the case: safe NFN rules need not be safe formulas in the sense of [12]. In fact, there are simple examples of formulas that would normally be regarded as safe that fail to be so under the [12] definition. Consider the following example.

$$p \leftarrow q(X) \vee r \tag{1}$$

This rule belongs to the syntactic class of programs with nested expressions as introduced in [16], although (1) contains a variable, something not considered in that work. Still, this rule is strongly equivalent to the conjunction of rules

$$p \leftarrow q(X)\ ;\ \ p \leftarrow r$$

and both are DLP safe (note that this is now a normal logic program) and so, they are BFL-safe and LLP-safe as well. Therefore, it is reasonable to consider (1) as safe. According to [12], however, for (1) to be safe, X must occur in some implication $G \rightarrow H$ such that X belongs to what are called the restricted variables of G. In this case, the only option for G is the rule body $q(X) \vee r$ whose set of restricted variables is empty,

[4] *Open* answer set programming studies decidable language extensions via guarded fragments rather than safety conditions,[8].

implying that the rule is not safe. [2] also uses a similar concept of restricted variable but only imposes restrictions on variables occurring in the head of the rule or within the scope of negation; so according to this account, (1) is safe.

In this paper we propose a new concept of safety that, while defined for arbitrary first order formulas, is strictly weaker than LLP-safety, while it captures BFL-safety when restricted to the same syntactic class of NFN-programs.

Desiderata for safety For our purposes there are three key desiderata associated with safety that we should keep distinct. The first property is very simple and does not refer to grounding. It says merely that a stable model should not contain unnamed individuals. This condition is formally expressed by Theorem 1 below and is fulfilled by formulas that we call *semi-safe*. Secondly we have the property usually most associated with safety. This is called *domain independence* and it does refer to grounding. It says that grounding a program with respect to any superset of the program's constants will not change the class of stable models. This is satisfied by formulas that we call *safe* and is expressed by Proposition 4 and Theorem 3. The third property, also satisfied by safe formulas, is expressed in Theorem 4: it says that the class of stable models should be first-order definable. This may be relevant for establishing properties such as interpolation and for computational purposes, being exploited for instance by the method of loop formulas.[5]

Let us consider some inherent limitations of a safety definition. Arbitrary theories are, in general, not domain independent. Even simple normal rules like, for instance:

$$p \leftarrow \mathit{not}\ q(X) \tag{2}$$

are not domain independent. To see why, take (2) plus the fact $q(1)$. If we ground this program on the domain $\{1\}$ we get the stable model $\{q(1)\}$, but if we use the extended domain $\{1, 2\}$, the only stable model becomes now $\{q(1), p\}$. Unfortunately, directly checking domain independence of an arbitrary formula does not seem a trivial task. It seems much more practical to find a (computationally) simple syntactic condition, like safety, that suffices to guarantee domain independence. However, due to its syntactic nature, a condition of this kind will just be *sufficient* but, most probably, not *necessary*. For instance, while as we saw above, (2) cannot be considered safe under any definition, if we consider the conjunction of (2) with the (variable-free) constraint:

$$\bot \leftarrow q(1) \tag{3}$$

the resulting program is *domain independent*. The reason is that (2) is strongly equivalent to the formula $(\exists x\ \neg q(x)) \rightarrow p$ whereas (3) is equivalent to $\neg q(1)$ that, in its turn, implies $(\exists x\ \neg q(x))$ in $\mathbf{QHT}$. This example shows that checking domain independence may require considering semantic interrelations of the formula or program as a whole. On the other hand, the fact that safety is sensitive to redundancies or that the safety of a formula may vary under a strongly equivalent transformation is unavoidable. For

[5] Of course there are other definable classes of stable models, such as those determined by formulas with a finite complete set of *loops* [13].

instance, the program consisting of (2) and the fact p is clearly domain independent, as in the presence of p, (2) becomes redundant and can be removed.

As regards our methodology, it is important to note that in order to handle explicit quantifiers and possible relaxations of the SNA it is indispensable to use a genuine first-order (or higher-order) semantics. Our approach here will be to analyse safety from the standpoint of (quantified) equilibrium logic in which a first-order definition of stable or equilibrium model can be given in the logic **QHT**. One specific aim is to generalise the safety concept from [12] so that rules such as (1), and indeed all the safe rules from [2], can be correctly categorised as safe. At least as regards simplicity and intuitiveness, we feel that our approach has some advantages over the methodology of [12] that uses a mixture of **QHT** and classical second-order logic.[6]

2 Review of Quantified Equilibrium Logic and Answer Sets

In this paper we restrict attention to function-free languages with a single negation symbol, '$\neg$', working with a quantified version of the logic *here-and-there*. In other respects we follow the treatment of [19]. We consider first-order languages $\mathcal{L} = \langle C, P\rangle$ built over a set of *constant* symbols, C, and a set of *predicate* symbols, P. The sets of $\mathcal{L}$-formulas, $\mathcal{L}$-sentences and atomic $\mathcal{L}$-sentences are defined in the usual way. We work here mainly with *sentences*. If D is a non-empty set, we denote by $At(D, P)$ the set of ground atomic sentences of the language $\langle D, P\rangle$. By an $\mathcal{L}$-interpretation I over a set D we mean a subset of $At(D, P)$. A *classical* $\mathcal{L}$-structure can be regarded as a tuple $\mathcal{M} = \langle (D, \sigma), I\rangle$ where I is an $\mathcal{L}$-interpretation over D and $\sigma\colon C \cup D \rightarrow D$ is a mapping, called the *assignment*, such that $\sigma(d) = d$ for all $d \in D$. If $D = C$ and $\sigma = id$, $\mathcal{M}$ is an *Herbrand structure*.

A *here-and-there* $\mathcal{L}$-structure with static domains, or $\mathbf{QHT}(\mathcal{L})$*-structure*, is a tuple $\mathcal{M} = \langle (D, \sigma), I_h, I_t\rangle$ where $\langle (D, \sigma), I_h\rangle$ and $\langle (D, \sigma), I_t\rangle$ are classical $\mathcal{L}$-structures such that $I_h \subseteq I_t$. We can think of a here-and-there structure $\mathcal{M}$ as similar to a first-order classical model, but having two parts, or components, h and t, that correspond to two different points or "worlds", 'here' and 'there', in the sense of Kripke semantics for intuitionistic logic [6], where the worlds are ordered by $h \leq t$. At each world $w \in \{h, t\}$ one verifies a set of atoms I_w in the expanded language for the domain D. We call the model static, since, in contrast to, say, intuitionistic logic, the same domain serves each of the worlds. Since $h \leq t$, whatever is verified at h remains true at t. The satisfaction relation for $\mathcal{M}$ is defined so as to reflect the two different components, so we write $\mathcal{M}, w \models \varphi$ to denote that φ is true in $\mathcal{M}$ with respect to the w component. The recursive definition of the satisfaction relation forces us to consider formulas from $\langle C \cup D, P\rangle$. Evidently we should require that an atomic sentence is true at w just in case it belongs to the w-interpretation. Formally, if $p(t_1, \ldots, t_n) \in At(C \cup D, P)$ and $w \in \{h, t\}$ then

$$\begin{aligned} \mathcal{M}, w \models p(t_1, \ldots, t_n) \quad &\text{iff} \quad p(\sigma(t_1), \ldots, \sigma(t_n)) \in I_w. \\ \mathcal{M}, w \models t = s \quad &\text{iff} \quad \sigma(t) = \sigma(s) \end{aligned}$$

[6] For reasons of space, in the sequel some proofs have been shortened or omitted. For a more detailed version, see `http://www.ia.urjc.es/~dpearce`.

Then $\models$ is extended recursively as follows, conforming to the usual Kripke semantics for intuitionistic logic given our assumptions about the two worlds h and t and the single domain D, see eg. [6]. We shall assume that $\mathcal{L}$ contains the constants $\top$ and $\bot$ and regard $\neg\varphi$ as an abbreviation for $\varphi \to \bot$.

- $\mathcal{M}, w \models \top,\ \mathcal{M}, w \not\models \bot$
- $\mathcal{M}, w \models \varphi \wedge \psi$ iff $\mathcal{M}, w \models \varphi$ and $\mathcal{M}, w \models \psi$.
- $\mathcal{M}, w \models \varphi \vee \psi$ iff $\mathcal{M}, w \models \varphi$ or $\mathcal{M}, w \models \psi$.
- $\mathcal{M}, t \models \varphi \to \psi$ iff $\mathcal{M}, t \not\models \varphi$ or $\mathcal{M}, t \models \psi$.
- $\mathcal{M}, h \models \varphi \to \psi$ iff $\mathcal{M}, t \models \varphi \to \psi$ and $\mathcal{M}, h \not\models \varphi$ or $\mathcal{M}, h \models \psi$.
- $\mathcal{M}, t \models \forall x \varphi(x)$ iff $\mathcal{M}, t \models \varphi(d)$ for all $d \in D$.
- $\mathcal{M}, h \models \forall x \varphi(x)$ iff $\mathcal{M}, t \models \forall x \varphi(x)$ and $\mathcal{M}, h \models \varphi(d)$ for all $d \in D$.
- $\mathcal{M}, w \models \exists x \varphi(x)$ iff $\mathcal{M}, w \models \varphi(d)$ for some $d \in D$.

Truth of a sentence in a model is defined as follows: $\mathcal{M} \models \varphi$ iff $\mathcal{M}, w \models \varphi$ for each $w \in \{h, t\}$. In a model $\mathcal{M}$ we also use the symbols H and T, possibly with subscripts, to denote the interpretations I_h and I_t respectively; so, an $\mathcal{L}$-structure may be written in the form $\langle U, H, T\rangle$, where $U = (D, \sigma)$. A structure $\langle U, H, T\rangle$ is called *total* if $H = T$, whence it is equivalent to a classical structure. We shall also make use of an equivalent semantics based on many-valued logic; the reader is referred to [18].

The resulting logic is called *Quantified Here-and-There Logic with static domains and decidable equality*, and denoted in [15] by $\mathbf{SQHT}^{=}$, where a complete axiomatisation can be found. In terms of satisfiability and validity this logic is equivalent to the logic previously introduced in [18]. To simplify notation we drop the labels for static domains and equality and refer to this logic simply as quantified here-and-there, $\mathbf{QHT}$. In the context of logic programs, the following often play a role. Let $\sigma|_C$ denote the restriction of the assignment σ to constants in C. In the case of both classical and $\mathbf{QHT}$ models, we say that the *parameter names assumption* (PNA) applies in case $\sigma|_C$ is surjective, i.e., there are no unnamed individuals in D; the *unique names assumption* (UNA) applies in case $\sigma|_C$ is injective; in case both the PNA and UNA apply, the *standard names assumption* (SNA) applies, i.e. $\sigma|_C$ is a bijection. We will speak about PNA-, UNA-, or SNA-models, respectively, depending on σ.

As in the propositional case, quantified equilibrium logic, or QEL, is based on a suitable notion of minimal model.

Definition 1. *Let φ be an $\mathcal{L}$-sentence. An* equilibrium *model of φ is a total model $\mathcal{M} = \langle (D, \sigma), T, T\rangle$ of φ such that there is no model of φ of the form $\langle (D, \sigma), H, T\rangle$ where H is a proper subset of T.*

2.1 relation to answer sets

We assume the reader is familiar with the usual definitions of answer set based on *Herbrand* models and ground programs, eg. [1]. Two variations of this semantics, the open [8] and generalised open answer set [9] semantics, consider non-ground programs and open domains, thereby relaxing the PNA.

For the present version of QEL the correspondence to answer sets can be summarised as follows (see [18, 19, 3]). If φ is a universal sentence in $\mathcal{L} = \langle C, P\rangle$ (see §3

below), a total **QHT** model $\langle U, T, T\rangle$ of φ is an equilibrium model of φ iff $\langle T, T\rangle$ is a propositional equilibrium model of the grounding of φ with respect to the universe U.

By the usual convention, when Π is a logic program with variables we consider the models of its universal closure expressed as a set of logical formulas. It follows that if Π is a logic program (of any form), a total **QHT** model $\langle U, T, T\rangle$ of Π is an equilibrium model of Π iff it is a generalised open answer set of Π in the sense of [9]. If we assume all models are UNA-models, we obtain the version of QEL found in [18]. There, the following relation of QEL to (ordinary) answer sets for logic programs with variables was established. If Π is a logic program, a total UNA-**QHT** model $\langle U, T, T\rangle$ of Π is an equilibrium model of Π iff it is an open answer set of Π.

[7] provides a new definition of stable model for arbitrary first-order formulas, defining the property of being a stable model syntactically via a second-order condition. However [7] also shows that the new notion of stable model is equivalent to that of equilibrium model defined here. In a sequel to this paper, [11] applies the new definition and makes the following refinements. The *stable models* of a formula are defined as in [7] while the *answer sets* of a formula are those Herbrand models of the formula that are stable in the sense of [7]. Using this new terminology, it follows that in general stable models and equilibrium models coincide, while answer sets are equivalent to SNA-**QHT** models that are equilibrium models.

3 Safety

We work with the same concept of restricted variable as used in [11, 12]. To every quantifier-free formula φ the set $\mathrm{RV}(\varphi)$ of its *restricted variables* is defined as follows:

- For atomic φ, if φ is an equality between two variables then $\mathrm{RV}(\varphi) = \varnothing$; otherwise, $\mathrm{RV}(\varphi)$ is the set of all variables occurring in φ;
- $\mathrm{RV}(\bot) = \varnothing$;
- $\mathrm{RV}(\varphi_1 \wedge \varphi_2) = \mathrm{RV}(\varphi_1) \cup \mathrm{RV}(\varphi_2)$;
- $\mathrm{RV}(\varphi_1 \vee \varphi_2) = \mathrm{RV}(\varphi_1) \cap \mathrm{RV}(\varphi_2)$;
- $\mathrm{RV}(\varphi_1 \to \varphi_2) = \varnothing$.

A sentence is said to be in *prenex* form if it has the following shape, for some $n \geq 0$:

$$Q_1 x_1 \ldots Q_n x_n \psi \tag{4}$$

where Q_i is $\forall$ or $\exists$ and ψ is quantifier-free. A sentence is said to be *universal* if it is in prenex form and all quantifiers are universal. A universal theory is a set of universal sentences. For **QHT**, normal forms such as prenex and Skolem forms were studied in [18]. In particular it is shown there that in quantified here-and-there logic every sentence is logically equivalent to a sentence in prenex form. A similar observation regarding first-order formulas under the new stable model semantics of [7] was made in [14]. Thus from a logical point of view there is no loss of generality in defining safety for prenex formulas.

3.1 Semi-safety

A variable assignment ξ in a universe (D, σ) is a mapping from the set of variables to D. If $\varphi \in \mathcal{L}$ has free-variables, φ^{ξ} is the closed formula obtained by replacing every free variable x by $\xi(x)$. On the other hand, in the following, if $T \subset At(D, P)$, we denote by $T|_C$ the subset of T whose atoms contain terms only from $\sigma(C)$.

Lemma 1. *Let φ be a quantifier-free formula and ξ a variable assignment in a universe (D, σ). If $\langle (D, \sigma), T|_C, T\rangle \models \varphi^{\xi}$, then $\xi(x) \in \sigma(C)$ for all $x \in \mathrm{RV}(\varphi)$.*

As in [12], we define a concept of *semi-safety* of a prenex form sentence φ in terms of the *semi-safety* of all its variable occurrences. Formally, this is done by defining an operator NSS that collects the variables that have *non-semi-safe* occurrences in a formula φ.

Definition 2 (NSS and semi-safety).

1. *If φ is an atom,* $\mathrm{NSS}(\varphi)$ *is the set of variables in φ.*
2. $\mathrm{NSS}(\varphi_1 \wedge \varphi_2) = \mathrm{NSS}(\varphi_1) \cup \mathrm{NSS}(\varphi_2)$
3. $\mathrm{NSS}(\varphi_1 \vee \varphi_2) = \mathrm{NSS}(\varphi_1) \cup \mathrm{NSS}(\varphi_2)$
4. $\mathrm{NSS}(\varphi_1 \to \varphi_2) = \mathrm{NSS}(\varphi_2) \setminus \mathrm{RV}(\varphi_1)$.

A sentence φ is said to be semi-safe *if* $\mathrm{NSS}(\varphi) = \varnothing$. □

In other words, a variable x is semi-safe in φ if every occurrence is inside some subformula $\alpha \to \beta$ such that, either x is restricted in α or x is semi-safe in β. This condition of semi-safety is a relaxation of that of [12], where the implication $\alpha \to \beta$ should always satisfy that x is restricted in α. As a result, (1) is semi-safe, but it is *not* considered so under the definition in [12]. Similarly, our definition implies, for instance, that any occurrence of a variable x in a negated subformula, $\neg\alpha(x)$, will be semi-safe – it corresponds to an implication $\alpha(x) \to \bot$ with no variables in the consequent. Other examples of semi-safe formulas are, for instance:

$$\neg p(x) \to (q(x) \to r(x)) \tag{5}$$

$$p(x) \vee q \to \neg r(x) \tag{6}$$

Note how in (6), x is not restricted in $p(x) \vee q$ but the consequent $\neg r(x)$ is semi-safe and thus the formula itself. On the contrary, the following formulas are not semi-safe:

$$p(x) \vee q \to r(x) \tag{7}$$

$$\neg\neg p(x) \wedge \neg r(x) \to q(x) \tag{8}$$

Lemma 2. *Let φ be a quantifier-free formula and ξ a variable assignment in (D, σ) s.t. $\xi(x) \in \sigma(C)$ for all $x \in \mathrm{NSS}(\varphi)$. If $\langle (D, \sigma), T, T\rangle \models \varphi^{\xi}$, then $\langle (D, \sigma), T|_C, T\rangle \models \varphi^{\xi}$.*

Proof. By induction over $\psi = \varphi^{\xi}$. If ψ is atomic the result is trivial and the induction step is also trivial for conjunctive and disjunctive formulas. So, let us assume that $\varphi =$

$\varphi_1 \to \varphi_2$ where, by the induction hypothesis, φ_2 is such that, if $\langle (D,\sigma), T, T\rangle \models \varphi_2^\xi$ and $\xi(x) \in \sigma(C)$ for all $x \in \mathrm{NSS}(\varphi_2)$, then $\langle (D,\sigma), T|_C, T\rangle \models \varphi_2^\xi$. Assume

$$\langle (D,\sigma), T, T\rangle \models \varphi_1^\xi \to \varphi_2^\xi \tag{9}$$

and $\xi(x) \in \sigma(C)$ for all $x \in \mathrm{NSS}(\varphi_1 \to \varphi_2)$. We must prove that

$$\langle (D,\sigma), T|_C, T\rangle \models \varphi_1^\xi \to \varphi_2^\xi. \tag{10}$$

First, suppose that $\langle (D,\sigma), T|_C, T\rangle \not\models \varphi_1^\xi$. If $\langle (D,\sigma), T|_C, T\rangle \models \neg\varphi_1^\xi$ then clearly $\langle (D,\sigma), T|_C, T\rangle \models \varphi_1^\xi \to \varphi_2^\xi$ and we are done. Otherwise, $\langle (D,\sigma), T|_C, T\rangle \models \neg\neg\varphi_1^\xi$. Then, by (9) $\langle (D,\sigma), T|_C, T\rangle, t \models \varphi_1^\xi \to \varphi_2^\xi$, and since $\langle (D,\sigma), T|_C, T\rangle, h \not\models \varphi_1^\xi$, (10) follows. Suppose therefore that $\langle (D,\sigma), T|_C, T\rangle \models \varphi_1^\xi$, then $\langle (D,\sigma), T, T\rangle \models \varphi_1^\xi$ and thus

$$\langle (D,\sigma), T, T\rangle \models \varphi_2^\xi \tag{11}$$

On the other hand, if $\langle (D,\sigma), T|_C, T\rangle \models \varphi_1^\xi$, by Lemma 1, $\xi(x) \in \sigma(C)$ for all $x \in \mathrm{RV}(\varphi_1)$ and by hypothesis, $\xi(x) \in \sigma(C)$ for all $x \in \mathrm{NSS}(\varphi_1 \to \varphi_2)$; in particular, for $\mathrm{NSS}(\varphi_1 \to \varphi_2) \cup \mathrm{RV}(\varphi_1) \supseteq \mathrm{NSS}(\varphi_2)$ we have:

$$\xi(x) \in \sigma(C) \text{ for all } x \in \mathrm{NSS}(\varphi_2) \tag{12}$$

By the ind. hyp., (11) and (12), we conclude that $\langle (D,\sigma), T|_C, T\rangle \models \varphi_2^\xi$. $\square$

This lemma allows us to conclude the main property of semi-safe formulas: their equilibrium models only refer to objects from the language.

Proposition 1. *If φ is semi-safe, and $\langle (D,\sigma), T, T\rangle \models \varphi$, then $\langle (D,\sigma), T|_C, T\rangle \models \varphi$.*

Proof. (sketch) Assume the formula is in prenex form and proceed by induction over the length of the prefix. $\square$

Theorem 1. *If φ is semi-safe, and $\langle (D,\sigma), T, T\rangle$ is an equilibrium model of φ, then $T|_C = T$.*

3.2 safe formulas

The concept of safety relies on semi-safety plus an additional condition on variable occurrences that can be defined in terms of Kleene's three-valued logic [10]. Given a three-valued interpretation $\nu\colon At \to \{0, 1/2, 1\}$, we extend it to evaluate arbitrary formulas $\nu(\varphi)$ as follows:

$$\begin{array}{ll} \nu(\varphi \wedge \psi) = \min(\nu(\varphi), \nu(\psi)) & \nu(\bot) = 0 \\ \nu(\varphi \vee \psi) = \max(\nu(\varphi), \nu(\psi)) & \nu(\varphi \to \psi) = \max(1 - \nu(\varphi), \nu(\psi)) \end{array}$$

from which we can derive $\nu(\neg\varphi) = \nu(\varphi \to \bot) = 1 - \nu(\varphi)$ and $\nu(\top) = \nu(\neg\bot) = 1$.

Definition 3 (ν_x operator). *Given any quantifier-free formula φ and any variable x, we define the three-valued interpretation so that for any atom α, $\nu_x(\alpha) = 0$ if x occurs in α and $\nu_x(\alpha) = 1/2$ otherwise.*

Intuitively, $\nu_x(\varphi)$ fixes all atoms containing the variable x to 0 (falsity) leaving all the rest undefined and then evaluates φ using Kleene's three-valued operators, that is nothing else but exploiting the defined values 1 (true) and 0 (false) as much as possible. For instance, $\nu_x(p(x) \rightarrow q(x))$ would informally correspond to $\nu_x(0 \rightarrow 0) = \max(1-0,0) = 1$ whereas $\nu_x(p(x) \vee r(y) \rightarrow q(x)) = \nu_x(0 \vee 1/2 \rightarrow 0) = \max(1 - \max(0, 1/2), 0) = 1/2$.

For the following definition, we need to recall the following terminology: a subexpression of a formula is said to be positive in it if the number of implications that contain that subexpression in the antecedent is even, and negative if it is odd.

Definition 4 (Weakly-restricted variable). *An occurrence of a variable x in $Qx\ \varphi$ is* weakly-restricted *if it occurs in a subformula ψ of φ such that:*

- *$Q = \forall$, ψ is positive and $\nu_x(\psi) = 1$*
- *$Q = \forall$, ψ is negative and $\nu_x(\psi) = 0$*
- *$Q = \exists$, ψ is positive and $\nu_x(\psi) = 0$*
- *$Q = \exists$, ψ is negative and $\nu_x(\psi) = 1$*

In all cases, we further say that ψ makes the ocurrence weakly restricted in φ.

Definition 5. *A semi-safe sentence is said to be* safe *if all its positive occurrences of universally quantified variables, and all its negative occurrences of existentially quantified variables are weakly restricted.*

For instance, the formula $\varphi = \forall x(\neg q(x) \rightarrow (r \vee \neg p(x)))$ is safe: the occurrence of x in $p(x)$ is negative, whereas the occurrence in $q(x)$ is inside a positive subformula, φ itself, for which x is weakly-restricted, since $\nu_x(\varphi) = \neg 0 \rightarrow (1/2 \vee \neg 0) = 1$. The occurrence of x in $\neg q(x)$ is not restricted and as a consequence the formula is not LLP-safe. The other aspect where Definition 5 is more general than LLP-safety results from the fact that to be safe x can occur negatively in φ given $Q_i = \forall$ or positively given $Q_i = \exists$. Another example of a safe formula is $\forall x((\neg\neg p(x) \wedge q(x)) \rightarrow r)$.

Lemma 3. *Let $\varphi(x)$ a prenex formula that has no free variables other than x. Let $\langle (D,\sigma), H, T\rangle$ be a model such that $T \subset \mathrm{At}(\sigma(C), P)$. Then:*

1. *If $\forall x\varphi(x)$ is safe:* $\langle (D,\sigma), H, T\rangle \models \forall x\varphi(x)$ *iff* $\langle (D,\sigma), H, T\rangle \models \bigwedge_{c\in C} \varphi(c)$.
2. *If $\exists x\varphi(x)$ is safe:* $\langle (D,\sigma), H, T\rangle \models \exists x\varphi(x)$ *iff* $\langle (D,\sigma), H, T\rangle \models \bigvee_{c\in C} \varphi(c)$.

Proof. (sketch) The proof makes use of the monotonic properties of the many-valued assignments with respect to the positive and negative ocurrences of subformulas. □

4 Grounding

Let (D,σ) be a domain and $D' \subseteq D$ a finite subset; the grounding over D' of a sentence φ, denoted by $\mathrm{Gr}_{D'}(\varphi)$, is defined recursively: the operator does not modify ground formulas, it propagates through propositional connectives and:

$$\mathrm{Gr}_{D'}(\forall x\varphi(x)) = \bigwedge_{d\in D'} \mathrm{Gr}_{D'}\varphi(d) \qquad \mathrm{Gr}_{D'}(\exists x\varphi(x)) = \bigvee_{d\in D'} \mathrm{Gr}_{D'}\varphi(d)$$

The following property of grounding holds (cf [18] for a version for universal theories.)

Proposition 2. *$\langle (D,\sigma), H, T\rangle \models \varphi$ if and only if $\langle (D,\sigma), H, T\rangle \models \mathrm{Gr}_D(\varphi)$*

In particular, if we work with PNA-models ($D = \sigma(C)$), we can ground using the initial constants, but we need the first-order semantics because with the mapping σ two constants may denote the same element of the domain; then we would obtain:

$$\langle (D,\sigma), H, T\rangle \models \varphi \text{ iff } \langle (D,\sigma), H, T\rangle \models \mathrm{Gr}_C(\varphi) \tag{13}$$

Lemma 4. *If $T \subset At(\sigma(C), P)$ and there exists $a \in D \setminus \sigma(C)$ then,*

1. *$\langle (D,\sigma), H, T\rangle \models \forall x\varphi(x)$ if and only if $\langle (D,\sigma), H, T\rangle \models \bigwedge_{d\in\sigma(C)\cup\{a\}} \varphi(d)$*
2. *$\langle (D,\sigma), H, T\rangle \models \exists x\varphi(x)$ if and only if $\langle (D,\sigma), H, T\rangle \models \bigvee_{d\in\sigma(C)\cup\{a\}} \varphi(d)$*

Proof. (sketch) If $T \subset At(\sigma(C), P)$, then the role of any object outside of $\sigma(C)$ is the same. So, we can choose just one element to represent all of them. □

Proposition 3. *If $T \subset At(\sigma(C), P)$ and $a \in D \setminus \sigma(C)$, then $\langle (D,\sigma), H, T\rangle \models \varphi$ if and only if $\langle (D,\sigma), H, T\rangle \models \mathrm{Gr}_{C\cup\{a\}}(\varphi)$.*

Theorem 2. *Let φ be a safe prenex formula. Let $\langle (D,\sigma), H, T\rangle$ be a model such that $T \subset \mathrm{At}(\sigma(C), P)$. Then:*

$$\langle (D,\sigma), H, T\rangle \models \varphi \text{ if and only if } \langle (D,\sigma), H, T\rangle \models \mathrm{Gr}_C(\varphi)$$

Proof. By induction on the length of the prefix. □

We can now show as promised the property that safe formulas satisfy domain independence. The following is immediate from Theorems 1 and 2 and the definition of equilibrium model.

Proposition 4. *Let φ be a safe prenex formula, then: $\langle (D,\sigma), T, T\rangle$ is an equilibrium model of φ if and only if it is an equilibrium model of $\mathrm{Gr}_C(\varphi)$.*

Here is an alternative formulation using language expansions.

Theorem 3. *Let φ be a safe prenex formula. Suppose we expand the language $\mathcal{L}$ by considering a set of constants $C' \supset C$. A total* **QHT***-model $\langle (D,\sigma), T, T\rangle$ is an equilibrium model of $\mathrm{Gr}_{C'}(\varphi)$ if an only if it is an equilibrium model of $\mathrm{Gr}_C(\varphi)$.*

The concept of domain independence from [2] is obtained as a special case if we assume that the unique name assumption applies.

5 Definability

An important property of safe formulas is that they form a definable class in first-order logic.[7] Let φ be a ground sentence and $(\alpha_1, \ldots, \alpha_n)$ the sequence of atoms in φ. If $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$ is another sequence of ground atoms, the formula $\varphi[\boldsymbol{\beta}]$ is built recursively:

[7] Shown for LLP-safety in [12]. An alternative approach to exhibiting definable classes of stable models, using loop formulas, can be found in [13].

- $\alpha_i[\boldsymbol{\beta}] = \beta_i$
- $\bot[\boldsymbol{\beta}] = \bot$
- $(\psi_1 \wedge \psi_2)[\boldsymbol{\beta}] = \psi_1[\boldsymbol{\beta}] \wedge \psi_2[\boldsymbol{\beta}]$
- $(\psi_1 \vee \psi_2)[\boldsymbol{\beta}] = \psi_1[\boldsymbol{\beta}] \vee \psi_2[\boldsymbol{\beta}]$
- $(\psi_1 \rightarrow \psi_2)[\boldsymbol{\beta}] = (\psi_1[\boldsymbol{\beta}] \rightarrow \psi_2[\boldsymbol{\beta}]) \wedge (\varphi \rightarrow \psi)$
- $(\neg\psi)[\boldsymbol{\beta}] = \neg\psi[\boldsymbol{\beta}] \wedge \neg\psi$

Lemma 5. *(i) Let φ be a ground sentence and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ the sequence of atoms in φ. Let $\mathbf{u}$ be a sequence $(u_1, \ldots, u_n)$ such that, for every i, either $u_i = \bot$ or $u_i = \alpha_i$ and there is some i such that $u_i = \bot$. (ii) Then $\langle (D, \sigma), T, T \rangle$ is an equilibrium model of φ if and only if T is a classical model of $\varphi \wedge \neg \bigvee_{\mathbf{u} \in U} \varphi[\mathbf{u}]$.*

Satisfaction in classical logic can be encoded by satisfaction by total models in here-and-there logic [20]; so we can characterise the classical models of the formula in the previous lemma by the total models of a formula in **QHT**, as follows:

Corollary 1. *Given statement (i) of Lemma 5, $\langle (D, \sigma), T, T \rangle$ is an equilibrium model of φ if and only if it is a* **QHT***-model of*

$$\bigwedge_{1 \leq i \leq n} (\neg\neg\alpha_i \rightarrow \alpha_i) \wedge \varphi[\neg\neg\boldsymbol{\alpha}] \wedge \neg \bigvee_{\mathbf{u} \in U} \varphi[\neg\neg\mathbf{u}]$$

As a consequence of this and Theorem 2 we obtain:

Theorem 4. *For every safe sentence φ, there exists a ground formula ψ in the same language such that $\langle (D, \sigma), T, T \rangle$ is an equilibrium model of φ if and only if it is a* **QHT***-model of ψ.*

6 Discussion and relation to other concepts of safety

As remarked in the introduction, one of our aims is to generalise the safety concept of [11, 12] and at the same time capture logic programs that are safe according to [2]. Although for reasons of space we cannot present here the full definition of LLP-safety from [12], we explained after Definition 5 the main features of our concept that generalise that of [12] (see also the examples discussed below). To verify the second objective we need to consider the class of logic programs treated by Bria, Faber and Leone in [2]. These programs are called *normal form nested*, or NFN for short. They comprise rules of a special form; in logical notation they are formulas that have the shape:

$$\bigwedge_{1 \leq i \leq m} \Delta_i \rightarrow \bigvee_{\leq i \leq n} \Gamma_j \tag{14}$$

where each Δ_i is a disjunction of literals and each Γ_j is a conjunction of atoms. A Δ_i is called a *basic* disjunction and is said to be *positive* if it contains only positive literals or atoms. In a rule r of form (14) the antecedent is called the *body* and the consequent the *head* of the rule. According to [2] r is safe if each variable in its head literals and its negative body literals is safe, where a variable x is said to be safe in r if if there exists

a positive basic disjunction Δ_i in the body of r such that x is a variable of α for every atom $\alpha \in \Delta_i$. It is easy to check that the BFL-safe variables in r are weakly restricted in r, therefore:

Proposition 5. *All rules of an NFN program that are safe according to [2] are safe formulas in the sense of Definition 5.*

Let us consider some examples showing that the safety concept introduced in Definition 5 is quite general and, in many cases, is better behaved with respect to strongly equivalent transformations than other definitions presented before. There are more examples where our definition detects safe sentences that are not LLP-safe. Eg, while

$$\neg\neg p(x) \rightarrow q \tag{15}$$

is LLP-safe, once we add $\neg r(x)$ to the body

$$\neg\neg p(x) \wedge \neg r(x) \rightarrow q \tag{16}$$

it becomes an LLP-unsafe sentence, whereas it still preserves safety under the current definition. Notice that the new x occurs negatively in the body but it is "made safe" by $\neg\neg p(x)$ and the fact that x does not occur in the head. In fact, if the head becomes $q(x)$ like in (8), the formula is not domain independent and so is not LLP-safe nor safe under our definition (in fact, as we saw, it is not even semi-safe).

The idea of modifying the definition of semi-safe formulas to check, as in [2], that a variable is restricted only if occurs in the head, can also be applied to existentially quantified formulas. For instance, this formula

$$\exists x\ (\neg\neg(p(x) \vee q)) \tag{17}$$

should be safe. It is strongly equivalent to $\neg\exists x\, p(x) \rightarrow q$ and this can be easily captured by introducing an auxiliary predicate in a safe way as follows:

$$\begin{aligned} aux_1 &\leftarrow p(X) \\ q &\leftarrow \neg aux_1 \end{aligned}$$

In fact, (17) is safe. However, it is not even semi-safe according to the definition in [11]: there is no implication containing x restricted in the antecedent.

7 Conclusion

We have presented a new concept of safety for general first-order formulas under stable model semantics and hence for very general kinds of answer set programs. As a logical formalism we have used quantified equilibrium logic based on the non-classical logic of quantified here-and-there, **QHT**, in which equilibrium or stable models can be easily defined as minimal models and their logical properties readily studied. We showed that the new concept of safety extends and generalises recent definitions of safety from [12] and [2]. Unlike in [2], in our approach we do not have to make the unique name

assumption. We showed that safe formulas satisfy three main properties or criteria of adequacy: their stable models do not contain unnamed individuals, they satisfy the condition of domain independence, and they form a first-order definable class of structures.

A topic of ongoing work concerns program transformations. In a sequel to this paper [5] we have begun to study classes of formulas whose safety is preserved under strong-equivalence preserving program transformations (in particular from [4]). This work may help to guide future improvements in the definition of a general safety concept, both from theoretical and from computational standpoints.

References

1. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. CUP, Cambridge (2002)
2. Bria, A., Faber, W. and Leone, N. Normal Form Nested Programs. S. Hölldobler *et al* (eds), JELIA 2008, Springer LNAI 5293, pp. 76–88, 2008.
3. de Bruijn, J., Pearce, D., Polleres, A., Valverde, A.: Quantified Equilibrium Logic and Hybrid Rules. In: Marchiori, M. *et al* (eds), RR2007, LNCS 4524, pp. 58-72, Springer, 2007
4. Cabalar, P., Pearce, D., Valverde, A.: Reducing Propositional Theories in Equilibrium Logic to Logic Programs. In: Bento, C. *et al* (eds) EPIA 05, LNAI 3808, pp. 4-17 Springer, 2005.
5. Cabalar, P., Pearce, D., Valverde, A.: Safety Preserving Transformations for General Answer Set Programs In: *Logic-Based program Synthesis and Transformation. Proc. LOPSTR 2009*.
6. van Dalen, D. *Logic and Structure*. Springer, 2004.
7. Ferraris, P., Lee, J., Lifschitz, V.: A New Perspective on Stable Models. In: Veloso, M. (ed) 20th International Joint Conference on Artificial Intelligence, IJCAI 2007, pp. 372-379.
8. Heymans, S., van Nieuwenborgh, D., Vermeir, D.: Open Answer Set Programming with Guarded Programs. ACM Trans. Comp. Logic 9, Article 26 (2008)
9. Heymans, S., Predoiu, L., Feier, C., de Bruijn, J., van Nieuwenborgh, D.: G-hybrid Knowledge Bases. In: ALPSWS 2006, CEUR Workshop Proceedings, vol 196 (2006)
10. Kleene, S. C., *On Notation for Ordinal Numbers.* The J. of Symbolic Logic 3(4), 1938.
11. Lee, J., Lifschitz, V., Palla, R.: A Reductive Semantics for Counting and Choice in Answer Set Programming. In: Proceedings AAAI 2008, pp. 472-479.
12. Lee, J., Lifschitz, V. and Palla, R.: Safe formulas in the general theory of stable models. Preliminary report in Proc. ICLP-08, Springer, LNCS 5366, pp. 672-6, 2008; longer version by personal communication (June 2008).
13. Lee, J. and Meng, V.: On Loop Formulas with Variables. Proc. of AAAI-08, AAAI. 2008.
14. Lee, J. Palla, R. Yet Another Proof of the Strong Equivalence between Propositional Theories and Logic Programs. In: CENT 2007, volume 265 of CEUR Workshop Proceedings, Tempe, AZ, May 2007. CEUR-WS.org.
15. Lifschitz, V., Pearce, D.and Valverde, A. A Characterization of Strong Equivalence for Logic Programs with Variables. In *Proc. of LPNMR 2007*, Springer LNAI 4483, pp. 188-200.
16. Lifschitz, V., Tang, L. and Turner, H. Nested Expressions in Logic Programs. *Annals of Mathematics and Artificial Intelligence*, 25(3-4): 369-389, 1999.
17. Pearce, D. and Valverde, A. Towards a first order equilibrium logic for nonmonotonic reasoning. In *Proc. of JELIA 2004*, Springer LNAI 3229, pp. 147-160.
18. Pearce, D. and Valverde, A. A First-Order Nonmonotonic Extension of Constructive Logic. *Studia Logica* 80:321-346, 2005.
19. Pearce, D. and Valverde, A. Quantified Equilibrium Logic. *Tech. report, Univ. Rey Juan Carlos*, 2006. `http://www.matap.uma.es/investigacion/tr/ma06_02.pdf`.
20. Pearce, D. and Valverde, A. Quantified Equilibrium Logic and Foundations for Answer Set Programs. *Proc. ICLP 08*, Springer, LNCS, 2008.