Safety Preserving Transformations 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. Many answer set solvers deal with programs with variables by requiring a safety condition on rules: any variable in a rule must appear in its positive body. This idea of safety has recently been extended to cover more general kinds of rules or first-order formulas that might be accepted by existing or future generation ASP systems [4, 12, 7]. In this paper we continue the study of the generalised safety concept recently proposed in [7]. In particular, we show that safety is preserved under a major subset of the transformations that reduce universal theories to disjunctive rules in ASP.

1 Introduction

1.1 Extensions of answer set semantics

Answer set programming (ASP) has become established as a vibrant new sub-field of logic programming and knowledge representation. There are now several rival implementations of ASP, many different kinds of language extensions, and a growing catalogue of practical applications.[4] While ASP systems continue to eliminate variables from programs by means of a grounding process, there is currently much interest in issues involving first order languages and programs. One important line of work in this direction concerns extending the basic language of disjunctive programs to embrace more general kinds of first order formulas.

Answer set semantics can be defined for general logical formulas by regarding answer sets or stable models as minimal models in a non-classical logic called *here-and-there*. This was shown for propositional theories in [16] and for first order theories in [19–21]. Subsequently, equivalent characterisations of answer sets using alternative logical frameworks were provided by [15, 8] in the propositional case and in [9] for first-order logic. However the former approach to answer set semantics remains in our

⋆ Partially supported by the MEC (now MICINN) projects TIN2006-15455-(C01,C02,C03) and CSD2007-00022, Junta de Andalucia project P6-FQM-02049 and Xunta de Galicia project INCITE08-PXIB105159PR.

[4] The recent LPNMR conferences provide a good source of references, eg [3, 2].

view the most natural and intuitive one. A key point in its favour is that here-and-there logic precisely captures the robust notion of strong equivalence for theories or programs under answer set semantics [13, 14]. That is to say, under answer set semantics one theory can be replaced by another in any context without loss if and only if the theories are equivalent in the logic of here-and-there. We denote this logic by $\mathbf{HT}$ in the propositional and by $\mathbf{QHT}$ in the quantified, first order case.

Besides ordinary disjunctive rules and general first order formulas, certain intermediate classes of formulas are also of special interest in ASP. Examples are general disjunctive rules where negation '$\neg$' is allowed in the heads as well as the bodies of rules, and rules with nested expressions where the rule body and head can be any compound expression involving $\wedge, \vee, \neg$ [15]. Recently [4] has studied a syntactically restricted subclass of the latter programs, called *normal form nested* or NFN programs.

Following these extensions of answer set semantics to more general syntactic classes of formulas, one further line of research in ASP has been to study program transformations that reduce a program from a more expressive syntactic class to one belonging to a simpler class. Already [15] showed how nested programs could be transformed into equivalent general disjunctive programs. [6] later showed that any propositional theory is strongly equivalent to a general disjunctive program in the same vocabulary, while [5] provided a complete set of transformations that effectively carries out this reduction.

In the first order case, the situation is briefly described as follows. As usual a first order sentence is said to be in *prenex* form if it has the following shape, for some $n \geq 0$:

$$Q_1 x_1 \dots Q_n x_n \psi \tag{1}$$

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. In [20] it is shown that in the logic $\mathbf{QHT}$ every sentence is logically equivalent to a sentence in prenex form. Without loss of generality we can therefore focus on sentences in prenex form. Since the matrix ψ in a prenex form is quantifier-free, we can apply equivalences from propositional logic to convert ψ into a special reduced form using the transformations described in [5]. They allow us to convert ψ into a logically equivalent general disjunctive rule. In this paper we shall focus on universal theories so that the transformations are all of a type that reduce (1) to a logic program of this general type.[5]

1.2 Safe formulas

The aim of this paper is to re-examine the transformations described in [5] from the point of view of *safety*. This fundamental concept in ASP is applied to rules of ordinary logic programs in the following way. A rule is said to be *safe* if each variable in it appears in the positive body of the rule. Many ASP implementations impose this condition by accepting only safe rules. There are three main properties of safe rules that we should distinguish. The first is that the answer sets of safe rules do not contain

[5] We postpone to future work the study of transformations that apply to an arbitrary prenex sentence or other kind of sentences involving existential quantifiers.

unnamed individuals. This condition is already fulfilled by formulas that we call *semi-safe*. Secondly there is the property usually called *domain independence* which says that grounding a program with respect to any superset of the program's constants will not change the class of answer sets. The third property satisfied by safe formulas is that the collection of their answer sets is first order definable. Like the other properties, this one is relevant for computational purposes, being exploited for instance by the method of loop formulas.

Recently, the concept of safety has been extended to more general formulas, for example to NFN programs in [4] and to arbitrary first-order formulas in [12]. Our own approach also covers arbitrary formulas and is described in [7]. It generalises the safety concept from [12] by re-classifying some kinds of formulas as safe that are unsafe according to [12]. At the same time, our concept still satisfies the three mentioned desiderata for safe formulas.

It is important to notice that safety is defined at the level of single formulas and is an inherently syntactical condition. It is therefore unreasonable to expect that the safety of a formula will be promulgated to arbitrary equivalent formulas. In particular safety may be gained or lost by removing or adding some redundant subformulas. For instance, a rule like $p \rightarrow q(x)$ is unsafe and in principle, may easily have different stable models depending on the domain we use for grounding (just add a fact p). However, if it is included in any program containing a constraint like $p \rightarrow \bot$, the unsafe rule becomes irrelevant. Similarly, with nested expressions, any safe rule $F \rightarrow G$ may become unsafe by a simple addition of a $\mathbf{QHT}$ tautology or inconsistency, as in $F \rightarrow G \vee (p(x) \wedge \neg p(x))$.

On the other hand, if we start with a general expression that is safe and apply certain kinds of logical re-writing steps such as those used in [5] to simplify formulas and reduce them to sets of general disjunctive rules, it might be reasonable to expect that an adequate concept of safety should be preserved under the transformations. In other words, while we cannot replace a safe formula by any arbitrary formula logically equivalent to it without losing safety, we can transform it into a possibly simpler expression while still maintaining safety. This is the problem that we shall study in the remainder of the paper.

The main result we establish is that when applied to universal sentences all but one of the transformation rules from [5] preserves safety. This means that a large class of safe first order formulas can be converted into strongly equivalent general disjunctive programs each of whose rules is safe. This collection includes the important class of all programs with nested expressions. While studying this problem we also found a slight generalisation of the safety concept from [7] which we apply here for the first time. The usual properties of safe formulas remain true for this revised concept.

2 Logical Background

In this paper we restrict attention to function-free first order languages $\mathcal{L} = \langle C, P \rangle$ built over a set of *constant* symbols, C, and a set of *predicate* symbols, P. We assume a single negation symbol, '$\neg$', together with the usual connectives and quantifiers, $\wedge, \vee, \rightarrow, \exists, \forall$. We shall also assume that $\mathcal{L}$ contains the constants $\top$ and $\bot$ and, where convenient, we regard $\neg\varphi$ as an abbreviation for $\varphi \rightarrow \bot$. In other respects we follow the treatment of [21]. The sets of $\mathcal{L}$-formulas, $\mathcal{L}$-sentences and atomic $\mathcal{L}$-sentences are defined in the usual way. The set of (free) variables of a formula φ will be denoted as $\mathrm{VARS}(\varphi)$.

We work in a non-classical logic called *Quantified Here-and-There Logic with static domains and decidable equality*. For reasons of space we give here just a short summary. A complete axiomatisation and more detailed description of this logic can be found in [14] where the logic is denoted by $\mathbf{SQHT}^{=}$. In terms of satisfiability and validity this logic is equivalent to the logic previously introduced in [20]. To simplify notation we drop the labels for static domains and equality and refer to this logic simply as quantified here-and-there, $\mathbf{QHT}$.

The semantics of $\mathbf{QHT}$ is given in terms of intuitionistic Kripke models, see [23], with two notable exceptions. One concerns equality: we regard equality as decidable and as satisfying the axiom $\forall x \forall y((x = y) \vee \neg(x = y))$. Furthermore, we suppose a logic with constant or static domains; in other words within a given Kripke model the same set of individuals populates each world. In addition, $\mathbf{QHT}$ is complete for very simple Kripke models, those possessing just two worlds, sometimes labelled h ("here") and t ("there"), ordered by $h \leq t$.

We use the following notation. If D is a non-empty set, we denote by $At(D, P)$ the set of ground atomic sentences of $\langle D, P\rangle$. By an $\mathcal{L}$-interpretation I over a set D we mean a subset of $At(D, P)$. A $\mathbf{QHT}(\mathcal{L})$-*structure* can therefore be regarded as a tuple $\mathcal{M} = \langle (D, \sigma), I_h, I_t\rangle$. where I_h, I_t are $\mathcal{L}$-interpretations over D such that $I_h \subseteq I_t$ and $\sigma\colon C \cup D \rightarrow D$ is a mapping, called the *assignment*, such that $\sigma(d) = d$ for all $d \in D$. Evidently, $\langle (D, \sigma), I_h\rangle$ and $\langle (D, \sigma), I_t\rangle$ are classical $\mathcal{L}$-structures. Given an interpretation we let $\sigma|_C$ denote the restriction of the assignment σ to constants in C.

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 $U = (D, \nu)$. A structure $\langle U, H, T\rangle$ is called *total* if $H = T$, whence it is equivalent to a classical structure.

An answer semantics for arbitrary first-order formulas can be defined using the quantified variant of equilibrium logic [16, **?**] that we denote by $\mathbf{QEL}$. As in the propositional case, this is based on a suitable notion of minimal model as follows.

Definition 1 ([19, 20]). *Let Γ be a set of $\mathcal{L}$-sentences. An* equilibrium *model or* answer set *of Γ is a total model $\mathcal{M} = \langle (D, \sigma), T, T\rangle$ of Γ such that there is no model of Γ of the form $\langle (D, \sigma), H, T\rangle$ where H is a proper subset of T.*

An equivalent characterisation of stable model or answer set for a finite set of first-order formulas is given in [9].

The study of strong equivalence for logic programs and nonmonotonic theories was initiated in [13]. It has since become an important tool in ASP as a basis for program transformation and optimisation. In equilibrium logic we say that two (first-order) theories Π_1 and Π_2 are strongly equivalent if and only if for any theory Π, $\Pi_1 \cup \Pi$ and $\Pi_2 \cup \Pi$ have the same equilibrium models [14, 21]. Under this definition we have:

Theorem 1 ([14, 21]). *Two (first-order) theories Π_1 and Π_2 are strongly equivalent if and only if they are equivalent in* $\mathbf{QHT}$.

Below we shall treat reductions that transform a formula into a logically equivalent set of formulas. These transformations therefore preserve strong equivalence.

3 Safety

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

- If φ is a non-equality atom: $\mathrm{RV}(\varphi) = \mathrm{VARS}(\varphi)$;
- $\mathrm{RV}(\varphi_1 \wedge \varphi_2) = \mathrm{RV}(\varphi_1) \cup \mathrm{RV}(\varphi_2)$;
- $\mathrm{RV}(\varphi_1 \vee \varphi_2) = \mathrm{RV}(\varphi_1) \cap \mathrm{RV}(\varphi_2)$;
- For the rest of cases: $\mathrm{RV}(\bot) = \mathrm{RV}(\varphi_1 \to \varphi_2) = \mathrm{RV}(t_1 = t_2) = \varnothing$.

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.[6] 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,* $\mathrm{NSS}(\varphi) = \mathrm{VARS}(\varphi)$.
2. $\mathrm{NSS}(\bot) = \varnothing$.
3. $\mathrm{NSS}(\varphi_1 \wedge \varphi_2) = \mathrm{NSS}(\varphi_1 \vee \varphi_2) = \mathrm{NSS}(\varphi_2) \cup \mathrm{NSS}(\varphi_1)$.
4. $\mathrm{NSS}(\varphi_1 \to \varphi_2) = \mathrm{NSS}(\varphi_2) \smallsetminus \mathrm{RV}(\varphi_1)$.

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. Let us remark that any negated formula is semi-safse, because $\mathrm{NSS}(\neg\varphi) = \varnothing$.

Example 1. Suppose that if x requests to y some item z and y is not a subprocess of x that has not item z then y rejects x's request and x gets unattended. We can represent this with the following rule with nested expressions:

$$request(x, y, z) \wedge \neg(subproc(x, y) \wedge \neg has(y, z)) \to ignore(y, x) \wedge unatt(x) \quad (2)$$

□

The formula (2) is semi-safe: all variables x, y and z occur in an implication (the main one) whose variables are restricted in the antecedent, $\mathrm{RV}(request(x, y, z) \wedge \neg(subproc(x, y) \wedge \neg has(y, z))) = \mathrm{RV}(request(x, y, z)) = \{x, y, z\}$.

The following results establish the main property of semi-safe formulas: their equilibrium models only refer to constants in the original language.

[6] Notice that while we study the effect of program transformations on *universal* sentences, safety and semi-safety are actually defined for arbitrary prenex sentences, so we give the general definition here.

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

Theorem 2. *If φ is semi-safe, and $\langle (D,\sigma), T, T\rangle$ is an equilibrium model of φ, then $T|_C = T$.*

The concept of safety relies on semi-safety plus an additional condition on variable occurrences. As a technical device we can define this condition using Kleene's three-valued logic [10]. Given a three-valued interpretation $\nu\colon \mathit{Atoms} \rightarrow \{0, 1/2, 1\}$, we extend it to evaluate arbitrary formulas $\nu(\varphi)$ as follows:

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

from which we can derive $\nu(\neg\varphi) = \nu(\varphi \rightarrow \bot) = 1 - \nu(\varphi)$ 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 A:*

$$\nu_x(A) \stackrel{\text{def}}{=} \begin{cases} 0 & \textit{if } x \in \mathrm{VARS}(A) \\ 1/2 & \textit{otherwise} \end{cases}$$

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 $\nu_x(p(x) \vee r(y) \rightarrow q(x)) = \nu_x(0 \vee 1/2 \rightarrow 0) = \max(1 - \max(0, 1/2), 0) = 1/2$.

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 **(safety).** *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, notice that (2) introduced in Example 1 is safe. All variables are universally quantified and all (positive) occurrences of x, y and z occur in a positive subformula, (2) itself, for which $\nu_x((2)) = \nu_y((2)) = \nu_z((2)) = 1$.

Theorem 3 establish the main property of safe formulas. The grounding over C of a sentence φ, denoted by $\mathrm{Gr}_C(\varphi)$, is defined recursively: the operator does not modify ground formulas, commutes with propositional connectives and

$$\mathrm{Gr}_C(\forall x \varphi(x)) = \bigwedge_{c \in C} \mathrm{Gr}_C \varphi(c) \qquad \mathrm{Gr}_C(\exists x \varphi(x)) = \bigvee_{c \in C} \mathrm{Gr}_C \varphi(c)$$

Theorem 3. *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 $\mathrm{Gr}_C(\varphi)$.*

Notice that, although in [7] Theorems 2 and 3 were established under a slightly different safety concept, it is easy to see that they continue to hold for the revised concept used here.

4 Negation Normal Form

The transformations introduced in [5] are top-down processes that rely on the successive application of several rewriting rules that operate on sets (conjunctions) of implications. A rewriting takes place whenever one of those implications does not yet have the form of a (non-nested) program rule.

Two sets of transformations are described next. A formula is said to be in *negation normal form* (NNF) when negation is only applied to literals. As a first step, we describe a set of rules that move negations inwards until a NNF is obtained:

$$\neg\top \Longleftrightarrow \bot \tag{N1}$$

$$\neg\bot \Longleftrightarrow \top \tag{N2}$$

$$\neg\neg\neg\alpha \Longleftrightarrow \neg\alpha \tag{N3}$$

$$\neg(\alpha \wedge \beta) \Longleftrightarrow \neg\alpha \vee \neg\beta \tag{N4}$$

$$\neg(\alpha \vee \beta) \Longleftrightarrow \neg\alpha \wedge \neg\beta \tag{N5}$$

$$\neg(\alpha \rightarrow \beta) \Longleftrightarrow \neg\neg\alpha \wedge \neg\beta \tag{N6}$$

Lemma 1. *For any pair $\gamma \Longleftrightarrow \gamma'$ in transformations* (N1)-(N6) *we have:*

1. $\mathrm{NSS}(\gamma) = \mathrm{NSS}(\gamma')$.
2. $\mathrm{RV}(\gamma) = \mathrm{RV}(\gamma')$.

Proof. Both properties are trivial, because for every transformation the application of the operators on both sides returns the empty set. □

By an inductive application of this lemma, we immediatelly conclude that NNF transformations preserve the semi-safe property, as stated below:

Proposition 2. *For any sentence φ and for every pair $\gamma \Longleftrightarrow \gamma'$ in transformations* (N1)-(N6) *we have that* $\mathrm{NSS}(\varphi) = \mathrm{NSS}(\varphi[\gamma/\gamma'])$. □

So, if φ' is an NNF formula obtained from φ by the application of the rules (N1)-(N6), then φ' is semi-safe if and only if φ is semi-safe. We prove now that the NNF conversion also preserves safety. To this aim, we first provide a pair of properties.

Observation 1 *For any pair $\gamma \Longleftrightarrow \gamma'$ in transformations* (N1)-(N6)*, if ψ is a subformula of α or β, then the sign of ψ in γ is equal to the sign in γ'.* □

Lemma 2. *For any pair $\gamma \Longleftrightarrow \gamma'$ in transformations* (N1)-(N6) *we have $\nu_x(\gamma) = \nu_x(\gamma')$ and thus, $\nu_x(\psi) = \nu_x(\psi[\gamma/\gamma'])$ for any formula ψ.*

Proof. It can be easily checked that, for each pair, $\gamma \iff \gamma'$, formulas γ and γ' are semantically equivalent in Kleene's three-valued logic, that is $\nu(\gamma) = \nu(\gamma')$ for any three-valued interpretation ν. □

Theorem 4. *Consider a semi-safe universal sentence $\forall x_1 \ldots \forall x_n \varphi$ and any pair $\gamma \iff \gamma'$ in transformations* (N1)-(N6) *such that γ is a subformula of φ: if x_i is safe in φ then it is also safe in $\varphi[\gamma/\gamma']$. Therefore, if φ is safe and φ' is an NNF formula obtained from φ by applying the transformations* (N1)-(N6)*, then φ' is also safe.*

Proof. To prove the result, we must analyse every occurrence of every variable in $\varphi[\gamma/\gamma']$ to check if it is made weakly-restricted. It is important to note that each one of these occurrences corresponds in a natural way to a specific occurrence of the same variable in the formula φ, because the transformations do not modify either the number or the relative situation of the variables. Note also that, by Observation 1, the transformation does not modify the sign of the variable occurrences. We proceed with the proof, distinguishing several cases.

Since the formula is universal, let us consider a positive occurrence of x_i and ψ the subformula of φ making the ocurrence weakly-restricted.

- If the occurrence is outside γ and γ is not a subformula of ψ, then ψ directly makes the corresponding ocurrence weakly-restricted in $\varphi[\gamma/\gamma']$.
- If γ is a subformula of ψ, then $\psi[\gamma/\gamma']$ makes the ocurrence weakly-restricted in $\varphi[\gamma/\gamma']$, because $\nu_x(\psi) = \nu_x(\psi[\gamma/\gamma'])$ (by Lemma 2) and the sign of ψ in φ is equal to the sign of $\psi[\gamma/\gamma']$ in $\varphi[\gamma/\gamma']$.
- If ψ is a strict subformula of γ, then we need to analyse every rule. For (N1), (N2) and (N3) the result is trivial. For rules (N4), (N5) and (N6), if ψ is subformula of α or β, then ψ makes the ocurrence weakly restricted in $\varphi[\gamma/\gamma']$, because the rule preserves the sign of subformulas of α and β.
- If $\psi = \alpha \wedge \beta$ then $\psi' = \neg\alpha \vee \neg\beta$ makes the corresponding ocurrence weakly restricted in $\varphi[\gamma/\gamma']$, because the sign of ψ' is the opposite of the sign of ψ and $\nu_x(\neg\alpha \vee \neg\beta) = \nu_x(\neg(\alpha \wedge \beta))$
- If $\psi = \alpha \vee \beta$ then $\psi' = \neg\alpha \wedge \neg\beta$ makes the corresponding ocurrence weakly restricted in $\varphi[\gamma/\gamma']$, because the sign of ψ' is the opposite of the sign of ψ and $\nu_x(\neg\alpha \wedge \neg\beta) = \nu_x(\neg(\alpha \vee \beta))$
- If $\psi = \alpha \to \beta$ then $\psi' = \neg\neg\alpha \wedge \neg\beta$ makes the corresponding ocurrence weakly restricted in $\varphi[\gamma/\gamma']$, because the sign of ψ' is the opposite of the sign of ψ and $\nu_x(\neg\neg\alpha \wedge \neg\beta) = \nu_x(\neg(\alpha \to \beta))$. □

Following with Example 1, after applying the NNF transformations, we obtain the safe rule:

$$request(x, y, z) \wedge (\neg subproc(x, y) \vee \neg\neg has(y, z)) \to ignore(y, x) \wedge unatt(x) \quad (3)$$

□

5 Transformations with implications

In the second set of transformations, as in [5] we deal with sets (conjunctions) of implications (the empty conjunction corresponds to $\top$). Each step replaces one of the implications by new implications to be included in the set. If φ is the (matrix of the) original formula, the initial set of implications is the singleton $\{\top \to \varphi\}$. Without loss of generality, we assume that any implication $\alpha \to \beta$ to be replaced has been previously transformed into NNF. Furthermore, we always consider that α is a conjunction and β a disjunction (if not, we just take $\alpha \wedge \top$ or $\beta \vee \bot$, respectively), and we implicitly apply commutativity of conjunction and disjunction as needed.

Left side rules:

$$\top \wedge \alpha \to \beta \iff \{\, \alpha \to \beta \,\} \tag{L1}$$

$$\bot \wedge \alpha \to \beta \iff \varnothing \tag{L2}$$

$$\neg\neg\varphi \wedge \alpha \to \beta \iff \{\, \alpha \to \neg\varphi \vee \beta \,\} \tag{L3}$$

$$(\varphi \vee \psi) \wedge \alpha \to \beta \iff \left\{\begin{matrix} \varphi \wedge \alpha \to \beta \\ \psi \wedge \alpha \to \beta \end{matrix}\right\} \tag{L4}$$

$$(\varphi \to \psi) \wedge \alpha \to \beta \iff \left\{\begin{matrix} \neg\varphi \wedge \alpha \to \beta \\ \psi \wedge \alpha \to \beta \\ \alpha \to \varphi \vee \neg\psi \vee \beta \end{matrix}\right\} \tag{L5}$$

Right side rules

$$\alpha \to \bot \vee \beta \iff \{\, \alpha \to \beta \,\} \tag{R1}$$

$$\alpha \to \top \vee \beta \iff \varnothing \tag{R2}$$

$$\alpha \to \neg\neg\varphi \vee \beta \iff \{\, \neg\varphi \wedge \alpha \to \beta \,\} \tag{R3}$$

$$\alpha \to (\varphi \wedge \psi) \vee \beta \iff \left\{\begin{matrix} \alpha \to \varphi \vee \beta \\ \alpha \to \psi \vee \beta \end{matrix}\right\} \tag{R4}$$

$$\alpha \to (\varphi \to \psi) \vee \beta \iff \left\{\begin{matrix} \varphi \wedge \alpha \to \psi \vee \beta \\ \neg\psi \wedge \alpha \to \neg\varphi \vee \beta \end{matrix}\right\} \tag{R5}$$

Theorem 5. $\mathrm{NSS}(\gamma) = \mathrm{NSS}(\gamma')$ *for any transformation of the form* $\gamma \iff \gamma'$ *in (L1)-(L4), (R1)-(R5), where* γ' *is the conjunction of the resulting formulas. Therefore, if* γ *is semi-safe, then* γ' *is also semi-safe.*

Proof. We prove case by case:

(L1) Trivially, $\mathrm{RV}(\bot \wedge \alpha) = \mathrm{RV}(\alpha)$ and thus,

$$\mathrm{NSS}(\top \wedge \alpha \to \beta) = \mathrm{NSS}(\beta) \setminus \mathrm{RV}(\top \wedge \alpha) = \mathrm{NSS}(\beta) \setminus \mathrm{RV}(\alpha) = \mathrm{NSS}(\alpha \to \beta)$$

(R1) Trivially, $\mathrm{NSS}(\top \vee \beta) = \mathrm{NSS}(\beta)$ and thus,

$$\mathrm{NSS}(\alpha \to \bot \vee \beta) = \mathrm{NSS}(\bot \vee \beta) \setminus \mathrm{RV}(\alpha) = \mathrm{NSS}(\beta) \setminus \mathrm{RV}(\alpha) = \mathrm{NSS}(\alpha \to \beta)$$

(L3) In the equality $(*)$ of the following sequence, we use that $\mathrm{RV}(\neg\neg\varphi) = \varnothing = \mathrm{NSS}(\neg\varphi)$:

$$\begin{aligned}
\mathrm{NSS}(\neg\neg\varphi \wedge \alpha \to \beta) &= \mathrm{NSS}(\beta) \smallsetminus \mathrm{RV}(\neg\neg\varphi \wedge \alpha) \\
&= \mathrm{NSS}(\beta) \smallsetminus (\mathrm{RV}(\neg\neg\varphi) \cup \mathrm{RV}(\alpha)) \\
&= (\mathrm{NSS}(\neg\varphi) \cup \mathrm{NSS}(\beta)) \smallsetminus \mathrm{RV}(\alpha) && (*) \\
&= (\mathrm{NSS}(\neg\varphi \vee \beta)) \smallsetminus \mathrm{RV}(\alpha) \\
&= \mathrm{NSS}(\alpha \to \neg\varphi \vee \beta)
\end{aligned}$$

(R3) In the equality $(*)$ of the following sequence, we use that $\mathrm{RV}(\neg\neg\varphi) = \varnothing = \mathrm{NSS}(\neg\varphi)$:

$$\begin{aligned}
\mathrm{NSS}(\alpha \to \neg\neg\varphi \vee \beta) &= \mathrm{NSS}(\neg\neg\varphi \vee \beta) \smallsetminus \mathrm{RV}(\alpha) \\
&= (\mathrm{NSS}(\neg\neg\varphi) \cup \mathrm{NSS}(\beta)) \smallsetminus \mathrm{RV}(\alpha) \\
&= \mathrm{NSS}(\beta) \smallsetminus (\mathrm{RV}(\neg\varphi) \cup \mathrm{RV}(\alpha)) && (*) \\
&= \mathrm{NSS}(\beta) \smallsetminus (\mathrm{RV}(\neg\varphi \wedge \alpha)) \\
&= \mathrm{NSS}(\neg\varphi \wedge \alpha \to \beta)
\end{aligned}$$

(L4) The main steps are properties of naive set theory

$$\begin{aligned}
&\mathrm{NSS}((\varphi \vee \psi) \wedge \alpha \to \beta) = \mathrm{NSS}(\beta) \smallsetminus \mathrm{RV}((\varphi \vee \psi) \wedge \alpha) \\
&\quad = \mathrm{NSS}(\beta) \smallsetminus ((\mathrm{RV}(\varphi) \cap \mathrm{RV}(\psi)) \cup \mathrm{RV}(\alpha)) \\
&\quad = \mathrm{NSS}(\beta) \smallsetminus ((\mathrm{RV}(\varphi) \cup \mathrm{RV}(\alpha)) \cap (\mathrm{RV}(\psi) \cup \mathrm{RV}(\alpha))) \\
&\quad = \mathrm{NSS}(\beta) \cap ((\mathrm{RV}(\varphi) \cup \mathrm{RV}(\alpha)) \cap (\mathrm{RV}(\psi) \cup \mathrm{RV}(\alpha)))^{\mathsf{C}} \\
&\quad = \mathrm{NSS}(\beta) \cap ((\mathrm{RV}(\varphi) \cup \mathrm{RV}(\alpha))^{\mathsf{C}} \cup (\mathrm{RV}(\psi) \cup \mathrm{RV}(\alpha))^{\mathsf{C}}) \\
&\quad = (\mathrm{NSS}(\beta) \cap (\mathrm{RV}(\varphi) \cup \mathrm{RV}(\alpha))^{\mathsf{C}}) \cup (\mathrm{NSS}(\beta) \cap (\mathrm{RV}(\psi) \cup \mathrm{RV}(\alpha))^{\mathsf{C}}) \\
&\quad = \mathrm{NSS}(\varphi \wedge \alpha) \to \beta) \cup \mathrm{NSS}(\psi \wedge \alpha \to \beta)
\end{aligned}$$

(R4) The main steps are properties of naive set theory

$$\begin{aligned}
&\mathrm{NSS}(\alpha \to (\varphi \wedge \psi) \vee \beta) = \mathrm{NSS}((\varphi \wedge \psi) \vee \beta) \cap \mathrm{RV}(\alpha)^{\mathsf{C}} \\
&\quad = (\mathrm{NSS}(\varphi) \cup \mathrm{NSS}(\psi) \cup \mathrm{NSS}(\beta)) \cap \mathrm{RV}(\alpha)^{\mathsf{C}} \\
&\quad = (\mathrm{NSS}(\varphi) \cup \mathrm{NSS}(\beta) \cup \mathrm{NSS}(\psi) \cup \mathrm{NSS}(\beta)) \cap \mathrm{RV}(\alpha)^{\mathsf{C}} \\
&\quad = ((\mathrm{NSS}(\varphi) \cup \mathrm{NSS}(\beta)) \cap \mathrm{RV}(\alpha)^{\mathsf{C}}) \cup ((\mathrm{NSS}(\psi) \cup \mathrm{NSS}(\beta)) \cap \mathrm{RV}(\alpha)^{\mathsf{C}}) \\
&\quad = \mathrm{NSS}(\alpha \to \varphi \vee \beta) \cup \mathrm{NSS}(\alpha \to \psi \vee \beta)
\end{aligned}$$

(R5) In the equality $(*)$ of the following sequence, we use that $\mathrm{RV}(\neg\varphi) = \varnothing = \mathrm{RV}(\neg\psi)$:

$$\begin{aligned}
&\mathrm{NSS}(\alpha \to (\varphi \to \psi) \vee \beta) = (\mathrm{NSS}(\varphi \to \psi) \cup \mathrm{NSS}(\beta)) \cap \mathrm{RV}(\alpha)^{\complement} \\
&\quad = ((\mathrm{NSS}(\psi) \cap \mathrm{RV}(\varphi)^{\complement}) \cup \mathrm{NSS}(\beta)) \cap \mathrm{RV}(\alpha)^{\complement} \\
&\quad = (\mathrm{NSS}(\psi) \cup \mathrm{NSS}(\beta)) \cap (\mathrm{RV}(\varphi)^{\complement} \cup \mathrm{NSS}(\beta)) \cap \mathrm{RV}(\alpha)^{\complement} \\
&\quad = (\mathrm{NSS}(\psi) \cup \mathrm{NSS}(\beta)) \cap ((\mathrm{RV}(\varphi)^{\complement} \cap \mathrm{RV}(\alpha)^{\complement}) \cup (\mathrm{NSS}(\beta) \cap \mathrm{RV}(\alpha)^{\complement})) \\
&\quad = ((\mathrm{NSS}(\psi) \cup \mathrm{NSS}(\beta)) \cap (\mathrm{RV}(\varphi)^{\complement} \cap \mathrm{RV}(\alpha)^{\complement})) \cup \\
&\quad\quad \cup ((\mathrm{NSS}(\psi) \cup \mathrm{NSS}(\beta)) \cap (\mathrm{NSS}(\beta) \cap \mathrm{RV}(\alpha)^{\complement})) \\
&\quad = \mathrm{NSS}(\varphi \wedge \alpha \to \psi \vee \beta) \cup (\mathrm{NSS}(\beta) \cap \mathrm{RV}(\alpha)^{\complement}) \\
&\quad = \mathrm{NSS}(\varphi \wedge \alpha \to \psi \vee \beta) \cup ((\mathrm{NSS}(\neg\varphi) \cup \mathrm{NSS}(\beta)) \cap (\mathrm{RV}(\alpha) \cup \mathrm{RV}(\neg\psi))^{\complement}) \\
&\quad = \mathrm{NSS}(\varphi \wedge \alpha \to \psi \vee \beta) \cup \mathrm{NSS}(\neg\psi \wedge \alpha \to \neg\varphi \vee \beta)
\end{aligned}$$

□

It is important to note that (L5) *does not preserve* semi-safety. If $(\varphi \to \psi) \wedge \alpha \to \beta$ is semi-safe then, although we can easily see that the two first rules resulting from (L5) are semi-safe:

$$\begin{aligned}
\varnothing = \mathrm{NSS}((\varphi \to \psi) \wedge \alpha \to \beta) &= \mathrm{NSS}(\beta) \setminus \mathrm{RV}((\varphi \to \psi) \wedge \alpha) \\
&= \mathrm{NSS}(\beta) \setminus \mathrm{RV}(\alpha) \\
&= \mathrm{NSS}(\beta) \setminus (\mathrm{RV}(\neg\varphi) \cup \mathrm{RV}(\alpha)) \\
&= \mathrm{NSS}(\neg\varphi \wedge \alpha \to \beta)
\end{aligned}$$

$$\begin{aligned}
\varnothing = \mathrm{NSS}((\varphi \to \psi) \wedge \alpha \to \beta) &= \mathrm{NSS}(\beta) \setminus \mathrm{RV}(\alpha) \\
&\supseteq \mathrm{NSS}(\beta) \setminus (\mathrm{RV}(\psi) \cup \mathrm{RV}(\alpha)) \\
&= \mathrm{NSS}(\psi \wedge \alpha \to \beta)
\end{aligned}$$

the third rule $\alpha \to \varphi \vee \neg\psi \vee \beta$, in the general case, is not semi-safe. As a counterexample, take the formula

$$(p(x) \to q) \to \neg r(x) \tag{4}$$

This formula is semi-safe and in fact, is safe. However, after applying (L5) to (4), we get the implication $\top \to p(x) \vee \neg q \vee \neg r(x)$, which is not semi-safe (in particular, the first of occurrence of x is not semi-safe) and thus, is not safe either. As we will see next, our definition of safety is indeed preserved for all transformations involving nested expressions (N1)-(N6), (L1)-(L4), (R1)-(R4), but fails for some cases dealing with nested implications.

Lemma 3. *For any pair $\gamma \iff \gamma'$ in transformations (L1)-(L5), (R1)-(R5) we have $\nu_x(\gamma) = \nu_x(\gamma')$ and thus, $\nu_x(\psi) = \nu_x(\psi[\gamma/\gamma'])$ for any formula ψ.*

Proof. Again, the result follows from semantic equivalences in Kleene's logic. □

Theorem 6. *Consider a semi-safe sentence $\forall x_1 \ldots \forall x_n \gamma$ and a pair $\gamma \Longleftrightarrow \gamma'$ in transformations (L1)-(L4), (R1)-(R5): if x_i is safe in γ then it is also safe in γ'.*

Proof. The proof is similar to that for Theorem 4. To prove the result, we must analyse each occurrence of every variable in γ' to check if it is made weakly restricted. Again, each of these occurrences corresponds, in a natural way, to a specific occurrence of the same variable in the formula φ, although an occurrence in γ may correspond to up to three occurrences in γ'. Also, it is easy to check now that the transformation does not modify the sign of the occurrences of the variables and that, in any pair in transformations (L1)-(L5), (R1)-(R5), if δ is a subformula of α, β, φ or ψ, then the sign of δ in γ is equal to the sign of the corresponding occurrence of δ in γ'.

Since the sentence is universal, if δ is a subformula that makes weakly restricted an ocurrence of x_i then we only need to analyse the cases in which δ is a strict subformula of γ, because the proof for the other situations is the same as for Theorem 4. Finally, for (L1), (L2), (L3), (R1), (R2) and (R3) the result is trivial.

(L4) If $\delta = \varphi \vee \psi$, then $\nu_x(\varphi \vee \psi) = \bot$ and thus $\nu_x(\varphi) = \nu_x(\psi) = \bot$; therefore, φ makes weakly restricted the corresponding ocurrence of x_i in $\varphi \wedge \alpha \to \beta$ and ψ makes weakly restricted the corresponding ocurrence of x_i in $\psi \wedge \alpha \to \beta$.
If $\delta = (\varphi \vee \psi) \wedge \alpha$, then $\nu_x((\varphi \vee \psi) \wedge \alpha) = \bot$ and either $\nu_x(\varphi \vee \psi) = \bot$ or $\nu_x(\alpha) = \bot$; both cases reduce to some of the previous cases.

(R4) If $\delta = \varphi \wedge \psi$, then $\nu_x(\varphi \wedge \psi) = \top$ and thus $\nu_x(\varphi) = \nu_x(\psi) = \top$; therefore, φ makes weakly restricted the corresponding ocurrence of x_i in $\alpha \to \varphi \vee \beta$ and ψ makes weakly restricted the corresponding ocurrence of x_i in $\alpha \to \psi \vee \beta$.
If $\delta = (\varphi \wedge \psi) \vee \beta$, then $\nu_x((\varphi \vee \psi) \wedge \alpha) = \top$ and either $\nu_x(\varphi \wedge \psi) = \top$ or $\nu_x(\alpha) = \top$; both cases reduce to some of the previous cases.

(R5) If $\delta = \varphi \to \psi$, then $\nu_x(\varphi \to \psi) = \top$ and either $\nu_x(\varphi) = \bot$ or $\nu_x(\psi) = \top$. In both cases, $\nu_x(\varphi \wedge \alpha \to \psi \vee \beta) = \nu_x(\neg\psi \wedge \alpha \to \neg\varphi \vee \beta) = \top$ and the complete formulas make the corresponding occurrences weakly restricted.

This result shows that transformations (L1)-(L4) and (R1)-(R4) plus (N1)-(N6), which allow unfolding rules with nested expressions, preserve safety. If we apply these transformations to our running example (3) we obtain the four rules:

$$
\begin{aligned}
request(x,y,z) \wedge \neg subproc(x,y) &\to ignore(y,x)\\
request(x,y,z) \wedge \neg subproc(x,y) &\to unatt(x)\\
request(x,y,z) &\to ignore(y,x) \vee \neg has(y,z)\\
request(x,y,z) &\to unatt(x) \vee \neg has(y,z)
\end{aligned}
$$

all of them safe.

In the case of nested implications, although for (L5) we do not obtain a positive result, we can still establish a sufficient condition for preserving safety, as follows.

Theorem 7. *Consider a semi-safe sentence $\forall x_1 \ldots \forall x_n \varphi$, the pair $\gamma \Longleftrightarrow \gamma'$ in transformation (L5) and suppose that $\alpha \to \varphi$ is semi-safe. Then, if x_i is safe in φ then it is also safe in $\varphi[\gamma/\gamma']$.*

Proof. Semi-safety of rules $(\neg\varphi \wedge \alpha \rightarrow \beta)$ and $(\psi \wedge \alpha \rightarrow \beta)$ was proved before. As for $(\alpha \rightarrow \varphi \vee \neg\psi \vee \beta)$, we get:

$$\begin{aligned}\mathrm{NSS}(\alpha \rightarrow \varphi \vee \neg\psi \vee \beta) &= (\mathrm{NSS}(\beta) \cup \mathrm{NSS}(\varphi) \cup \mathrm{NSS}(\neg\psi)) \smallsetminus \mathrm{RV}(\alpha)\\ &= (\mathrm{NSS}(\beta) \cup \mathrm{NSS}(\varphi)) \smallsetminus \mathrm{RV}(\alpha)\end{aligned}$$

but as $\alpha \rightarrow \varphi$ is semi-safe, $\mathrm{NSS}(\varphi) \smallsetminus \mathrm{RV}(\alpha) = \varnothing$ and we obtain:

$$\begin{aligned}&= \mathrm{NSS}(\beta) \smallsetminus \mathrm{RV}(\alpha)\\ &= \mathrm{NSS}((\varphi \rightarrow \psi) \wedge \alpha \rightarrow \beta) = \varnothing\end{aligned}$$

□

To see how this sufficient condition can be applied, let us consider a variation of (4) where we include in the antecedent an additional atom $dom(x)$ (possibly fixing the "domain" of x):

$$(p(x) \rightarrow q) \wedge dom(x) \rightarrow \neg r(x)$$

This formula is still safe and, furthermore, the implication $dom(x) \rightarrow p(x)$ is semi-safe. Thus, the result of applying (L5) yields the three rules:

$$\neg p(x) \wedge dom(x) \rightarrow \neg r(x) \tag{5}$$
$$q \wedge dom(x) \rightarrow \neg r(x) \tag{6}$$
$$dom(x) \rightarrow p(x) \vee \neg q \vee \neg r(x) \tag{7}$$

that are now safe.

5.1 Discussion

Taken together, the transformations (N1)-(N6), (L1)-(L5), (R1)-R(5), are sufficient to reduce a universal sentence in prenex form with matrix φ into a prenex formula whose matrix, say φ', has the form of a general disjunctive program rule of shape $\alpha \rightarrow \beta$, where α is a conjunction of literals and β is a disjunction of literals. The resulting transformation therefore has the form of a logic program allowing negation in the heads of rules. As we saw, all transformations preserve the property of safety, except (L5), where safety preservation can be ensured, but at the cost of an additional condition. On the other hand, removing nested occurrences of implication in the heads of rules does not affect safety.

Quantifier-free formulas of the form $\alpha \rightarrow \beta$ where α, β do not contain occurrences of implication, other than in the form '$\rightarrow \bot$', are known in ASP as rules with *nested expressions* and a set of such rules is called a *nested program*, [15]. Taken together, therefore, (N1)-(N6), (L1)-(L4), (R1)-R(4), are sufficient to transform any nested program into a fully equivalent general disjunctive program, preserving the safety of formulas in each case.

6 Related Work and Conclusions

We have defined a safety condition on first order formulas and identified a syntactic class of formulas that can be transformed via rewriting rules that preserve strong equivalence and safety. This syntactic class contains eg. all nested logic programs.

While the condition of safety is in general highly relevant for computational purposes, it should be noted that here we have been concerned with logical issues rather than matters of computation and implementation. In fact, while the transformations we have studied do not introduce any new terms into the language, they are also in general not polynomial in size. Polynomial reductions of nested programs have been studied in [17, 18] and polynomial reductions of arbitrary propositional formulas to logic programs are discussed in [5]. These transformations may lend themselves to a more efficient implementation of the reductions, but they introduce new predicates. In so doing, safety may be lost. In [4] a restricted subclass of nested programs is identified, called *normal form nested* or NFN. A concept of safety for NFN rules is proposed in [4] along with a polynomial algorithm for reducing them to disjunctive programs that can be processed by the DLV system. This reduction method introduces new predicates and other auxiliary devices. It does preserve safety, however [4] does not prove the correctness of the reduction, something that in our case of equivalence preserving transformations is easy to establish. Evidently, our safety concept is also much more widely applicable than that of [4].

It is perhaps worth to mention that, although our concept of safety is weaker than that of [12], that is, it allows classifying more formulas as safe, it is also more robust with respect to transformations. For instance, a nested expression fact like $\neg(p(x) \land \neg q(x))$ is classified as safe both under our definition and the one in [12], whereas the result after applying transformation (N4) to transform it into NNF, $\neg p(x) \lor \neg\neg q(x)$ preserves our concept of safety, while becomes unsafe under [12].

There remain several directions for further study. One is the search for an improved concept of safety along with a complete set of transformations that preserve this property. Another is the investigation of algorithms for a more efficient reduction of general formulas to logic programs while preserving safety. Another topic is the study of transformations on existential sentences and arbitrary formulas involving existential quantifiers.

References

1. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. CUP, Cambridge (2002)
2. Baral, C., Brewka, G. and Schliof, J. (Eds), *Proc. of the 9th International Conference on Logic Programming and Non Monotonic Reasoning (LPNMR 2007)*, Springer LNAI 4483, 2007.
3. Baral, C., Greco, G., Leone, N. and Teracine, G. (Eds), *Proc. of the 8th International Conference on Logic Programming and Non Monotonic Reasoning (LPNMR 2005)*, Springer LNAI 3662, 2005.
4. 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.

5. 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.
6. Cabalar, P. and Ferraris, P.: Propositional Theories are Strongly Equivalent to Logic Programs, *Theory and Practice of Logic Programming* 7 (6), pp. 745-759, 2007.
7. Cabalar, P., Pearce, D., Valverde, A.: A Revised Concept of Safety for General Answer Set Programs (extended version). Technical Report `http://www.ia.urjc.es/~dpearce`.
8. Ferraris, P.: Answer Sets for Propositional Theories In *Proc. of the 8th International Conference on Logic Programming and Non Monotonic Reasoning (LPNMR 2005)*, Springer LNAI 3662, pp. 119-131, 2005.
9. 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.
10. Klenne, S. C., *Introduction to Metamathematics*, North-Holland, Amsterdam, 1952.
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. J. Lee, V. Lifschitz and R. Palla. Safe formulas in the general theory of stable models. (Preliminary report). in Proceedings of ICLP-08, Springer, LNCS, 2008.
13. V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2:526–541, 2001.
14. 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.
15. Lifschitz, V., Tang, L. and Turner, H. Nested Expressions in Logic Programs. *Annals of Mathematics and Artificial Intelligence*, 25(3-4): 369-389, 1999.
16. D. Pearce. A New Logical Characterisation of Stable Models and Answer Sets. In *Proc. of NMELP 96*, 1997.
17. D. Pearce, H. Tompits and S. Woltran. Encodings for equilibrium logic and logic programs with nested expressions. In *Proc. of EPIA 2001*, Springer LNAI 2258, pp. 306-320.
18. D. Pearce, H. Tompits and S. Woltran. Chatacterising equilibrium logic and nested logic programs: reductions and complexity. Technical Report GIA 2007-01-12, Universidad Rey Juan Carlos, 2007; to appear in *Theory and Practice of Logic programming*.
19. 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.
20. Pearce, D. and Valverde, A. A First-Order Nonmonotonic Extension of Constructive Logic. *Studia Logica* 80:321-346, 2005.
21. Pearce, D. and Valverde, A. Quantified Equilibrium Logic. *Tech. report, Univ. Rey Juan Carlos*, 2006. $http://www.matap.uma.es/investigacion/tr/ma06_02.pdf$.
22. Pearce, D. and Valverde, A. Quantified Equilibrium Logic and Foundations for Answer Set Programs. *Proc. ICLP 08*, Springer, LNCS, 2008.
23. van Dalen, D. *Logic and Structure*. Springer, 2004.