A Free Logic for Stable Models with Partial Intensional Functions*

P. Cabalar[1], L. Fariñas[2], D. Pearce[3], and A. Valverde[4]

[1] Department of Computer Science
University of Corunna, Spain
cabalar@udc.es
[2] University of Toulouse IRIT, CNRS, France
farinas@irit.fr
[3] Universidad Politécnica de Madrid, Spain
david.pearce@upm.es
[4] Universidad de Málaga, Spain
a_valverde@ctima.uma.es

Abstract. In this paper we provide a new logical characterisation of stable models with partial functions that consists in a free-logic extension of Quantified Equilibrium Logic (QEL). In so-called "free" logics, terms may denote objects that are outside the domain of quantification, something that can be immediately used to capture partial functions. We show that this feature can be naturally accommodated in the monotonic basis of QEL (the logic of Quantified Here-and-There, QHT) by allowing variable quantification domains that depend on the world where the formula is being interpreted. The paper provides two main contributions: (i) a correspondence with Cabalar's semantics for stable models with partial functions; and (ii) a Gentzen system for free QHT, the monotonic basis of free QEL.

1 Introduction: Functions in ASP

Answer Set Programming (ASP) [21, 22, 5] constitutes nowadays one of the most popular paradigms for practical Knowledge Representation (KR) and problem solving, being regularly present in mainstream conferences on KR and Artificial Intelligence (AI). This popularity can be attributed not only to its practical applicability, with available state-of-the-art solvers[5] and an increasing number of applications, but also to its robust formal basis, relying on the *stable model* semantics for logic programs [15]. Although stable models were originally defined for propositional logic programs, their logical characterisation in terms of *Equilibrium Logic* [23] paved the way for their extension to more general syntactic

* This research was partially supported by: European French-Spanish Lab IREP; MEC project TIN2012-39353-C04; Junta de Andalucía project TIC115; Xunta de Galicia, Spain, grant GPC2013/070; and Universidad de Málaga, Campus de Excelencia Internacional Andalucía Tech.

[5] See, for instance the report from the fourth ASP Competition [1]

classes. In particular, the first-order extension of this logic, *Quantified Equilibrium Logic* (QEL) [24], allows the definition of stable models for any arbitrary first-order theory [13] and became a powerful theoretical tool for analysing fundamental properties such as strong equivalence [19], safety [9], interpolation [14] or synonymy [25], being in this way a salient, successful case of Logics in AI.

The extension of stable models to an arbitrary first-order syntax has brought into focus a feature traditionally excluded from ASP: the treatment of functions. Although most ASP solvers are propositional, their input language allows the use of variables that, in an initial *grounding* phase, are replaced by their possible ground instantiations, under the assumption (inherited from logic programming) of an Herbrand domain. Due to grounding limitations, ASP has traditionally forbidden the use of functions because the simple introduction of one function symbol makes the Herbrand universe infinite. This distinctive difference between ASP and Prolog has been overcome with `DLV-complex` [11], a tool that allows the grounding of ASP programs with arbitrarily nested Herbrand functions that satisfy a given property of being *finitely-ground* [10] (although checking that property is undecidable).

Apart from Herbrand functions, a less explored possibility that has recently attracted attention is the use of *evaluable* functions in ASP. While an Herbrand function is expected to act as a *syntactic constructor* for defining objects in the universe, such as a tuple or a list, an evaluable function is expected to behave with its usual mathematical meaning, that is, as an *operator*[6] that returns a value, as, for instance, the standard arithmetic operations for integer numbers. Dealing with evaluable functions may have two main advantages. First, from the KR perspective, the use of nested functions usually allows a more compact and natural reading, avoiding the introduction of auxiliary variables that may become a potential source of error. To give an example, saying that X is a patrilineal great grandfather of Y could be naturally represented as $X = father(father(father(Y)))$ whereas in predicate notation, we would need a rule body of the form $father(Y, Z), father(Z, T), father(T, X)$ whose meaning is not so easily recognisable at a first sight, apart from requiring two extra auxiliary variables. Second, evaluable functions can be computationally exploited both at the grounding phase, reducing the ground program size, and at the solving phase, avoiding an overload of constraints.

An immediate interpretation for evaluable functions in ASP was already provided by QEL, since this logic was not necessarily restricted to Herbrand functions. As shown in [20], QEL semantics for evaluable functions[7] can be exploited for a more efficient grounding on scenarios with functional dependences, if we replace propositional ASP solvers by a CSP tool as a backend. Unfortunately, the other potential advantage of using functions, namely, their adequacy for a flexible KR, is not achieved by this approach. In particular, functions in QEL

[6] This distinction between constructors and operators is, in fact, part of the motivation from the area of *Functional Logic Programming* [16].

[7] Although Lin and Wang's approach was independently established, its correspondence to QEL was proven in [7].

are somehow asymmetrical with respect to predicates, since they do not allow non-monotonic reasoning (NMR). A reasonable requirement for a functional semantics is that replacing all predicates by Boolean functions should have no particular effect on the results excepting the minor changes in notation – each atom $p(X)$ would be replaced by the expression $p(X) = true$. However, predicates in ASP are *intensional*: we can just provide the rules for which they hold, assuming that anything else is false. Furthermore, thanks to default negation, we can further specify default rules for a predicate that are applied in the absence of exceptions. As an example, a graph can be described by merely asserting a fact $edge(i, j)$ for each edge, while remaining atoms for that predicate will be false by default. Moreover, we can inductively define a reachability predicate with the pair of rules:

$$reach(X, Y) \leftarrow edge(X, Y) \qquad reach(X, Y) \leftarrow edge(X, Z), reach(Z, Y)$$

something that is well-known to be non-representable in classical first-order logic. Unfortunately, under QEL semantics, functions behave "classically" and *there is no way of defining a function default value* without resorting to predicate-based representations. In our example, if we replace predicates *edge* and $reach$ by Boolean functions, the stable models we obtain correspond to the *classical* models of the original predicate-based theory.

1.1 Approaches to intensional functions

Although the idea of default values for functions is not new [8], Lifschitz suggested the name *intensional functions* [18] to refer to evaluable functions that allow NMR features analogous to those obtained with intensional predicates. There currently exist two different ways of understanding intensional functions. On the one hand, Bartholomew and Lee introduced a variant [3] (we will call BL semantics) that repairs some counterintuitive features of Lifschitz's approach. Like the latter, BL semantics exclusively deals with total functions defining their "stability" in terms of value *uniqueness* among values stemming from possible models. On the other hand, a previous definition[8] by Cabalar [7] considers instead a minimal-information criterion for partial functions. To understand the difference, take the example formula:

$$father(abel) = adam \tag{1}$$

assuming *abel* and *adam* are Herbrand-constants. Under Cabalar's semantics this formula has a unique stable model where *abel*'s father is *adam* and *adam*'s father, in turn, is left *undefined by default*. Notice how this interpretation is aligned to the idea of minimal information from predicate-based representations. If we just had a predicate fact $father(abel, adam)$ the unique stable model would satisfy $\neg\exists x\ father(adam, x)$ underlining that *adam*'s father is undefined. In this sense,

[8] As shown in [4], the recent approach by Balduccini [2] for logic programs with partial functions is actually equivalent to Cabalar's semantics.

Cabalar's semantics can be seen as a "conversion" of predicate-based ASP into functional notation whose main advantage is nesting functions: for instance, we can conclude that Abel's grandfather $father(father(abel)) = father(adam)$ is also undefined.

Under BL semantics, however, (1) has no stable models since the value of $father(adam)$ is not uniquely defined – in principle, with those two persons in the domain, the possibilities are $father(adam) = abel$ or $father(adam) = adam$ himself. In this way, the intuition behind BL intensional functions is clearly different from predicate-based ASP and relies on an idea of selecting a function value when there is *no other way* to vary that value. This idea was captured in [4] and [12] defining in the latter a *flexible* extension of QEL together with a Gentzen calculus for the "flexible" version of its monotonic basis, the so-called logic of *Quantified Here-and-There* (QHT).

Apart from their different understandings for functions, one important difference in the behaviour of BL and Cabalar's semantics has to do with the treatment of nested functions. In particular, Cabalar's semantics satisfies:

$$\varphi(f(x)) \equiv \exists y \ (f(x) = y \wedge \varphi(y)) \tag{2}$$

for any term $f(x)$ occurring in formula φ, where x is free in φ and y is not free in φ. As a result, nested functions can be safely "unfolded" until all atoms involving functions eventually have the form $f(t) = t'$ where t and t' are function-free terms. This syntactic form is called *c-plain* in [4] and there it was shown that both BL and Cabalar's semantics coincide for this form of theory, under the assumption of total functions.[9] Unfortunately, the unfolding transformation (2) is not safe in BL semantics and the question whether any theory can be equivalently reduced to c-plain form under BL is still unanswered.

1.2 Contribution of the paper

Although, as explained above, Cabalar's semantics seems a promising alternative for interpreting intensional functions, there was no axiomatisation for this logic yet, and so its properties could only be proved at a semantic level. In this paper, we consider an equivalent reformulation of Cabalar's semantics in terms of a *free-logic* extension of Quantified Equilibrium Logic (QEL). The term "free" logic refers to a family of formalisms where syntactic terms may denote objects that are outside the domain of quantification, something that can be used to capture partial functions.[10] We show that this feature can be naturally accommodated

[9] In fact, as explained in [4], the difference total/partial between the two semantics is not essential. In Cabalar's semantics, any function can always be forced to be total by adding an axiom $\neg\neg\exists y \ f(x) = y$. In BL semantics, we can always define a special constant *none* to represent the fact that the function has no value. A comparison like $none = none$ would become true, but under c-plain syntax, such comparisons never occur.

[10] A useful reference is [26] that presents various approaches to free logic over intuitionistic logic.

in the monotonic basis of QEL (the logic of Quantified Here-and-There, QHT) by allowing variable quantification domains that depend on the world where the formula is being interpreted. Apart from capturing Cabalar's semantics, this free-logic characterisation also opens new possibilities for interpreting partial intensional functions that will be explored in the future.

The main contributions of the paper are as follows. First, in Section 2 we describe the free quantified logic of here-and-there, **FHT**, the monotonic basis of free QEL. In Section 3 we then show that **FHT**-models are equivalent to the semantics of Cabalar's partial functions. And in Section 4 we present a Gentzen calculus for **FHT** with corresponding completeness theorems.

2 The Free (Quantified) Here-and-There logic

We consider a first-order language with signature $\Sigma = \langle C, F, P \cup \{=\}\rangle$, where C is the set of constants (or 0-ary functions), F is the set of function symbols and P is the set of predicate symbols. We assume that each predicate $p \in P$ has an associated *arity*, an integer denoting the number of arguments $arity(p) \geq 0$. Similarly, each function $f \in F$ is associated with an $arity(f) > 0$.

First-order formulas are built up in the usual way, with the same syntax of classical predicate calculus with equality $=$. Formally, we assume a countably infinite set of variables, the constant $\bot$, the connectives, '$\vee$', '$\wedge$', '$\rightarrow$', '$\exists$', '$\forall$' and auxiliary parentheses. Negation is defined by $\neg\varphi \stackrel{\text{def}}{=} \varphi \rightarrow \bot$ and double implication is denoted by $\varphi \leftrightarrow \psi \stackrel{\text{def}}{=} (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$. We use letters x, y, z and their capital versions to denote variables, τ to denote terms, c to denote constants and d objects in the domain. Overlined letters like $\overline{x}, \overline{\tau}, \overline{c}, \overline{d}, \ldots$ represent tuples (in this case of variables, terms, constants, and objects respectively). An atom like $\tau = \tau'$ is called an *equality atom*, whereas an atom like $p(\tau_1, \ldots, \tau_n)$ with $n \geq 0$ for any predicate p different from equality is called a*predicate atom*. We denote by $\mathrm{At}(C, F, P)$, or At for short, the set of ground predicate atoms over the language. We also write $Terms(C, F)$ to stand for the set of ground terms formed with constants in C and functions in F.

We will be exclusively interested in closed formulas or sentences, that is, those where each variable is bound by some quantifier. For the sake of readability, however, we will sometimes allow free variables, but as an abbreviation for their universal quantification. A set of sentences is called a *theory*.

Kripke semantics for intermediate logics relies on the idea of possible worlds with an accessibility relation $\leq$ among them that, at least, satisfies reflexivity and transitivity. The simplest case of intermediate logic strictly below classical logic is known as the Logic of Here-and-There (**HT**) [17] where only two worlds are considered, h ("here") and t ("there"), so that $h \leq t$. Apart from being reflexive and transitive, the relation $\leq$ in intermediate logics must also satisfy an important property called *persistence* or *inheritance* so that any accessible world $w' \geq w$ must have at least as much information (true assertions) as the current one w. In the propositional case, this implies that the true atoms in w are a subset of those in w'. In the first-order case, this is naturally extrapolated so that the

extent of any predicate $p(\overline{x})$ in w is a subset of its extent in w'. For instance, in **HT** we could have $\{p(0), p(1)\}$ true in world h and $\{p(0), p(1), p(2), p(3)\}$ true in world t. When thinking of logic programs, it is somehow natural that all worlds share a common domain, normally the Herbrand Universe, that in our example would correspond to $\{0, 1, 2, 3\}$. When this happens, we say that the intermediate logic has a *static domain*. This was, in fact, the choice taken in the original definition of Quantified Here-and-There with Static domains [24], or **SQHT** for short, where worlds h and t shared the same universe. In a more general setting, however, each world w could have its own domain D_w provided that, for any accessible world $w' \geq w$, we guarantee that the domain has at least as many objects as in w, that is, $D_w \subseteq D_{w'}$. In our example a possible situation could be, for instance, $D_h = \{0, 1, 2\}$ and $D_t = \{0, 1, 2, 3\}$. This immediately introduces a way of representing the idea of *undefined elements*: for instance, 3 is undefined in world h but becomes defined in world t whereas 4 is undefined in both worlds. Using non-static domains has immediate consequences for quantification and functional terms, since there may exist elements that cannot be denoted in the current world, but that become denotable in an accessible world instead. To be more precise, we use the Meinongiam approach for free semantics in intuitionistic logic [6, 26], in which an outer domain $D \supseteq D_t$ is considered. This is exactly the semantic structure we introduce next for defining the *Free logic of Quantified Here-and-There*, or **FHT** for short.

Given a function σ and a tuple of terms $\overline{\tau} = \tau_1, \ldots, \tau_n$ we write $\sigma(\overline{\tau})$ to stand for the tuple $\sigma(\tau_1), \ldots, \sigma(\tau_n)$.

Definition 1 (FHT-interpretation). *An* **FHT**-interpretation M*, is a tuple* $M = \langle D_h, D_t, D, \equiv, I, \sigma \rangle$ *verifying the following conditions.*

(F1) $D_h \subseteq D_t \subseteq D$ *are a triple of increasing domains.*
(F2) $\equiv$ *is an equivalence relation on* D_t*, such that:*
  (a) *There is no pair of elements* $d \neq d'$ *from* D_h *such that* $d \equiv d'$*;*
  (b) *For all* $d \in D_t$*, there exists* $d' \in D_h$ *such that* $d \equiv d'$*.*

$\sigma \colon \text{Terms}(C \cup D, F) \to D$*, the interpretation for terms, is a mapping recursively defined and verifying:*

(F3) $\sigma(d) = d$ *if* $d \in D$*.*
(F4) *For any world* $w \in \{h, t\}$*, if* $\sigma(\tau) \in D_w$ *then* $\sigma(\tau') \in D_w$ *for each subterm* τ' *of* τ*.*
(F5) *If* $d_i \equiv d'_i$ *for every* i*, then* $\sigma(f(d_1, \ldots, d_n)) \equiv \sigma(f(d'_1, \ldots, d'_n))$*.*

I is an interpretation for predicates that assigns to each predicate p with arity n at each world $w \in \{h, t\}$ a set of tuples of elements $I(p, w)$ following the rules:

(F6) $I(p, w) \subseteq D_w^n$
(F7) *if* $d, d' \in D_t$*,* $d \equiv d'$*, and* $(\ldots, d, \ldots) \in I(p, t)$*, then* $(\ldots, d', \ldots) \in I(p, t)$
(F8) $I(p, h) \subseteq I(p, t)$. □

Condition (F1) is standard for dynamic domains in intermediate logics – as we explained before, they must contain an increasing set of elements to satisfy the inheritance condition. Condition (F2) is necessary for capturing Cabalar's treatment of the equality predicate. While in the h world, an equality atom $\tau_1 = \tau_2$ will just be interpreted by checking whether $\sigma(\tau_1)$ and $\sigma(\tau_2)$ coincide, in the t world we will use instead a separate equivalence relation '$\equiv$' among elements in D_t. In this way, two different elements $d \neq d'$ can be equivalent $d \equiv d'$ and so, they can be interpreted as "equal" in the t world. Given some $d \in D_t$, we write $[d]$ to represent its $\equiv$-equivalence class. However, (F2) specifies two strong restrictions: (a) says that if these two different elements $d \neq d'$ are in D_h they cannot become equivalent in D_t. Intuitively, this will mean that if we have two *defined* terms in h with a different value, they must remain defined and different (equality is false) in world t. On the other hand, (b) means that all the elements we use in $D_t \setminus D_h$ must have some "purpose" with respect to D_h. More formally, any $d \in D_t \setminus D_h$ must be equivalent to some element in $D_h \subseteq D_t$. This restriction allows us to capture an important condition in Cabalar's semantics: *if a function is defined in world h, its value is maintained in world t*. We will see an example of this, once the satisfaction of formulas is defined. Since the $\equiv$ relation must behave as a kind of equality, it must additionally satisfy (F5) and (F7), so that replacement of equivalent terms preserves function values and truth for predicates. Conditions (F4) and (F6) mean that evaluation of predicates and terms at world w is "fixed" to elements in that world w, even through subterms. Notice that the expanded language includes a constant for each object in D and that (F3) evaluates any object constant to itself; for simplicity we do not make a notational difference between the domain element and its name. (F8) is the usual condition of persistence for predicate atoms from here to there.

We define when a **FHT**-interpretation $M = \langle D_h, D_t, D, \equiv, I, \sigma \rangle$ *satisfies* a formula φ at world $w \in \{h, t\}$, written $M, w \models \varphi$, recursively as follows:

- $M, w \not\models \bot$
- $M, w \models p(\overline{\tau})$ iff $\sigma(\overline{\tau}) \in I(p, w)$.
- $M, h \models \tau_1 = \tau_2$ iff $\sigma(\tau_1) = \sigma(\tau_2) \in D_h$.
- $M, t \models \tau_1 = \tau_2$ iff $\sigma(\tau_1), \sigma(\tau_2) \in D_t$ and $\sigma(\tau_1) \equiv \sigma(\tau_2)$.
- $M, w \models \varphi \wedge \psi$ iff $M, w \models \varphi$ and $M, w \models \psi$,
- $M, w \models \varphi \vee \psi$ iff $M, w \models \varphi$ or $M, w \models \psi$,
- $M, h \models \varphi \rightarrow \psi$ iff $M, t \models \varphi \rightarrow \psi$ and $M, h \not\models \varphi$ or $M, h \models \psi$,
- $M, t \models \varphi \rightarrow \psi$ iff, $M, t \not\models \varphi$ or $M, t \models \psi$,
- $M, w \models \forall x\ \varphi(x)$ iff $M, w \models \varphi(d)$ for all $d \in D_w$.
- $M, w \models \exists x\ \varphi(x)$ iff $M, w \models \varphi(d)$ for some $d \in D_w$.

The concepts of validity, equivalence and semantic consequence are defined as usual. To understand how these definitions work for undefined functions, let us extend our Biblical genealogy example.

Example 1. Assume we have the Herbrand constants *adam*, *cain*, *abel* and take the following situation that is compatible with Cabalar's semantics. Suppose that $M, w \models father(abel) = adam$ in worlds $w \in \{h, t\}$, whereas $father(cain)$

is undefined in world h, $M, h \models \neg\exists x\, father(cain) = x$ taking value $adam$ in world t, $M, t \models father(cain) = adam$. Besides, in both worlds, we still have $father(adam)$ undefined. To represent this situation in **FHT** we would fix $D_h = \{abel, cain, adam\}$, $D_t = D_h \cup \{cf\}$ and $D = D_h \cup \{af\}$ with $cf \equiv adam$ where cf and af are unnamed elements that respectively stand for "Cain's father" and "Adam's father." Then $\sigma(father(abel)) = adam \in D_h$, $\sigma(father(cain)) = cf \in D_t \setminus D_h$ but $cf \equiv adam$ and, finally, $\sigma(father(adam)) = af \in D \setminus D_t$. □

To define equilibrium models, we say that an interpretation $M = \langle D_h, D_t, D, \equiv, I, \sigma\rangle$ is *smaller than* $M' = \langle D'_h, D'_t, D', \equiv', I', \sigma'\rangle$, written $M \leq M'$, when $D \setminus D_t = D' \setminus D'_t$ (elements that represent functions undefined both here and there must coincide in both interpretations), $I(p,t) = I'(p,t)$ and $I(p,h) \subseteq I'(p,h)$ for every predicate p, and finally, for every τ, one of these three cases holds: (1) $\sigma(\tau) \notin D_t$ and $\sigma'(\tau) \notin D'_t$; or (2) $\sigma(\tau) = \sigma'(\tau) \in D_h \cap D'_h$; or (3) $\sigma(\tau) \in D_t \setminus D_h$, $\sigma'(\tau) \in D'_h \cap D_h$, $\sigma(\tau) \equiv \sigma'(\tau)$. Then a model M of a theory Γ is said to be an *equilibrium model* iff there is no other model $M' \neq M$, $M' \leq M$ of Γ.

3 Relation to Cabalar's partial functions

In this section we recall the basic definitions from Cabalar's extension [7] of **SQHT** for dealing with partial functions. The main idea of this semantics relies on keeping a static domain D, common for both worlds h and t, but the interpretation of terms may map now to a special object $\mathtt{u} \notin D$ that stands for "undefined." Let us denote this variant as $\mathbf{SQHT}_{\mathtt{u}}$ and recall[11] next its main semantic definitions.

Definition 2 ($\mathbf{SQHT}_{\mathtt{u}}$-interpretation). *A* $\mathbf{SQHT}_{\mathtt{u}}$-interpretation *is a tuple* $M = \langle D, I, \sigma_h, \sigma_t\rangle$ *where* σ_w *with* $w \in \{h,t\}$ *are functions* $\sigma_w : Terms(C \cup D, F) \to D \cup \{\mathtt{u}\}$ *with* $\mathtt{u}$ *some new element* $\mathtt{u} \notin D$ *(standing for "undefined") and satisfying:*

(U1) $\sigma_w(d) = d$ *for all* $d \in D$.
(U2) *The mappings* σ_w *are recursive and verify* $\sigma_w(f(\overline{\tau})) = \mathtt{u}$ *if* $\sigma_w(\tau_i) = \mathtt{u}$ *for some* τ_i *in* $\overline{\tau}$.
(U3) $\sigma_h(\tau) = \sigma_t(\tau)$ *or* $\sigma_h(\tau) = \mathtt{u}$ *for all* $\tau \in Terms(C \cup D, F)$.

and I *is an interpretation for predicates satisfying:*

(U4) $I(p,w) \subseteq D^n$, *if* $arity(p) = n$, *and*
(U5) $I(p,h) \subseteq I(p,t)$. □

An interpretation $M = \langle D, I, \sigma_h, \sigma_t\rangle$ is *total* iff $\sigma_h = \sigma_t$ and $I(p,h) = I(p,t)$ for every predicate p. We say that $M = \langle D, I, \sigma_h, \sigma_t\rangle$ is smaller than $M' = \langle D, I', \sigma'_h, \sigma'_t\rangle$, written $M \leq M'$, when $I(p,t) = I'(p,t)$ and $I(p,h) \subseteq I'(p,h)$ for every predicate p, $\sigma_t = \sigma'_t$, and $\sigma_h(\tau) = \mathtt{u}$ or $\sigma_h(\tau) = \sigma'_h(\tau)$ for every term τ.

[11] For simplicity, we omit the distinction between Herbrand and non-Herbrand functions made in the original definition of [7].

Definition 3 (Equilibrium model). *A total* $\mathbf{SQHT_u}$ *interpretation* M *is an* equilibrium model *of a theory* Γ *iff* $M, h \models \alpha$ *for all* $\alpha \in \Gamma$ *and there is no strictly smaller* $M' < M$ *such that* $M', h \models \alpha$ *for all* $\alpha \in \Gamma$.

The satisfaction relation in $\mathbf{SQHT_u}$, written $\models_u$, is defined as follows.

- $M, w \models_u p(\overline{\tau})$ iff $\sigma_w(\overline{\tau}) \in I(p, w)$;
- $M, w \models_u \tau_1 = \tau_2$ iff $\sigma_w(\tau_1) = \sigma_w(\tau_2) \neq \mathtt{u}$;
- $\bot$, $\wedge$ and $\vee$ are interpreted as usual;
- $M, h \models_u \varphi \to \psi$ iff $M, t \models_u \varphi \to \psi$ and either $M, h \not\models_u \varphi$ or $M, h \models_u \psi$;
- $M, t \models_u \varphi \to \psi$ iff either $M, t \not\models_u \varphi$ or $M, t \models_u \psi$;
- $M, w \models_u \forall x\ \varphi(x)$ iff $M, w \models_u \varphi(d)$ for all $d \in D$;
- $M, w \models_u \exists x\ \varphi(x)$ iff $M, w \models_u \varphi(d)$ for some $d \in D$.

To prove equivalence between **SQHT** and **FHT** we will use the next observation.

Proposition 1. *Let* $\mathcal{L}_1$ *and* $\mathcal{L}_2$ *be two different Kripke logics for a common syntax and set of worlds* W*, and let* c *be a correspondence assigning an* $\mathcal{L}_2$ *interpretation* M^c *to any* $\mathcal{L}_1$ *interpretation* M*. If* c *is such that, at any world* $w \in W$*, both* M, w *and* M^c, w *satisfy the same set of formulas, then* $\mathcal{L}_2 \subseteq \mathcal{L}_1$.

We provide next a pair of correspondences that satisfy the conditions in Proposition 1: mapping '$*$' from **SQHT** interpretations into **FHT** interpretations, and mapping '†' in the opposite direction. In the sequel, if $\tau \in \text{Terms}(D \cup C, F)$, we write $\tau(d_1, \ldots, d_n)$ to indicate that $d_1, \ldots, d_n$ are the elements of D occurring in τ. Given an **FHT** interpretation $M = \langle D_h, D_t, D, \equiv, I, \sigma \rangle$ and assuming $\mathtt{u} \notin D$, we define an $\mathbf{SQHT_u}$ *interpretation* $M^* = \langle D^*, I, \sigma_h^*, \sigma_t^* \rangle$ as:

- $D^* \stackrel{\text{def}}{=} D_t / \equiv$
- If $\tau([d_1], \ldots, [d_n]) \in \text{Terms}(D^* \cup C, F)$ with $d_i \in D_h$ and $\sigma(\tau(d_1, \ldots, d_n)) \in D_h$, then $\sigma_h^*(\tau([d_1], \ldots, [d_n])) = [\sigma(\tau(d_1, \ldots, d_n))]$;
  otherwise, $\sigma_h^*(\tau([d_1], \ldots, [d_n])) = \mathtt{u}$.
- If $\sigma(\tau(d_1, \ldots, d_n)) \in D_t$, then $\sigma_t^*(\tau([d_1], \ldots, [d_n])) = [\sigma(\tau(d_1, \ldots, d_n))]$;
  otherwise, $\sigma_t^*(\tau([d_1], \ldots, [d_n])) = \mathtt{u}$.
- If $d_i \in D_h$ for every i, $([d_1], \ldots, [d_n]) \in I^*(p, h)$ iff $(d_1, \ldots, d_n) \in I(p, h)$.
- $I^*(p, t) = \{([d_1], \ldots, [d_n]) \mid (d_1, \ldots, d_n) \in I(p, t)\}$, if $n = \mathit{arity}(p)$.

The mappings σ_w are well defined, because if $d \equiv d'$ and $\tau(d)$ is a term containing d, then by condition (F5), $\sigma(\tau(d)) \equiv \sigma(\tau(d'))$. The interpretation I^* is also well defined by conditions (F2) and (F6).

As an example, consider $D_h = \{0, 1, 2\}$, $D_t = \{0, 1, 2, 3\}$ and $D = \mathbb{N}$. Any σ in an **FHT**-interpretation will assign $\sigma(i) = i$ for any natural number $i \in \mathbb{N}$. Then $\sigma_h^*(i) = i$ for $i \in \{0, 1, 2\}$ and $\sigma_h^*(i) = \mathtt{u}$ for all the rest. Similarly $\sigma_t^*(i) = i$ for $i \in \{0, 1, 2, 3\}$ and $\sigma_t^*(i) = \mathtt{u}$ otherwise.

Proposition 2. *If* M *is an* **FHT***-interpretation, then* M^* *is a well-formed* **SQHT***-interpretation.* □

Theorem 1. *Let M be an* **FHT** *interpretation and α an arbitrary sentence. Then $M, w \models \alpha$ iff $M^*, w \models_u \alpha$ for any $w \in \{h, t\}$.* □

Given an $\mathbf{SQHT_u}$-interpretation $M = \langle D, I, \sigma_h, \sigma_t \rangle$ we provide now the correspondence for the other direction, defining the associated **FHT**-interpretation $M^\dagger = (D_h, D_t, D^\dagger, \equiv, I^\dagger, \sigma)$ as follows:

- $D^\dagger = \text{Terms}(D \cup C, F)/\equiv_h$, where $\tau_1 \equiv_h \tau_2$ if either $\tau_1 = \tau_2$, or $\sigma_h(\tau_1) = \sigma_h(\tau_2) \neq u$.
- $D_h = \{[\tau] \mid \sigma_h(\tau) \neq u\}$
- $D_t = \{[\tau] \mid \sigma_t(\tau) \neq u\}$
- $[\tau_1] \equiv [\tau_2]$ iff $\sigma_t(\tau_1) = \sigma_t(\tau_2)$.
- $\sigma([\tau]) = [\tau], \quad \sigma(f([\tau_1], \ldots, [\tau_2])) = [f(\tau_1, \ldots, \tau_2)]$.
- $I^\dagger(p, w) = \{([\tau_1], \ldots, [\tau_n]) \mid (\sigma_w(\tau_1), \ldots, \sigma_w(\tau_n)) \in I(p, w)\}$, if $n = \mathit{arity}(p)$.

The mapping σ is well defined, because if $\sigma_h(\tau_1) = \sigma_h(\tau_i')$ for every i, then, by recursion, $\sigma_h(f(\tau_1, \ldots, \tau_n)) = \sigma_h(f(\tau_1', \ldots, \tau_n'))$. On the other hand, $\equiv$ is well defined: if $\sigma_h(\tau_1) = \sigma_h(\tau_2) \neq u$, then $\sigma_t(\tau_1) = \sigma_h(\tau_1) = \sigma_h(\tau_2) = \sigma_t(\tau_2)$.

Proposition 3. *Let $M = \langle D, I, \sigma_h, \sigma_t \rangle$ be an $\mathbf{SQHT_u}$-interpretation. Then $M^\dagger = \langle D_h, D_t, D^\dagger, \equiv, I^\dagger, \sigma \rangle$ is a well-formed* **FHT***-interpretation.* □

Theorem 2. *Let M be an $\mathbf{SQHT_u}$-interpretation and α an arbitrary sentence. Then $M, w \models_u \alpha$ iff $M^\dagger, w \models \alpha$ for any $w \in \{h, t\}$.* □

4 Gentzen calculus FHTG

In this section we introduce a Gentzen Calculus **FHTG** with multi-consequent sequents of the form $\Gamma \vdash \Delta$ where, Γ and Δ are sets of formulas (respectively understood as a conjunction and a disjunction). The soundness of the system is guaranteed if the rules preserve the following property: for a rule $\frac{\Gamma_0 \vdash \Delta_0}{\Gamma_1 \vdash \Delta_1}$ if M is a countermodel of $\Gamma_0 \vdash \Delta_0$ then it is also a countermodel of $\Gamma_1 \vdash \Delta_1$; and M is a countermodel of $\Gamma \vdash \Delta$ if $M \models \varphi$ for every $\varphi \in \Gamma$ and $M \not\models \psi$ for every $\psi \in \Delta$. We begin by introducing the axioms and the rules of the basic system.

Axioms: $\qquad \Gamma, \varphi \vdash \Delta, \varphi; \qquad \Gamma, \varphi, \neg\varphi \vdash \Delta;$

Rules for propositional connectives:

$$\frac{\Gamma, \alpha, \beta \vdash \Delta}{\Gamma, \alpha \wedge \beta \vdash \Delta}\ (\text{L-}\wedge) \qquad \frac{\Gamma \vdash \Delta, \alpha \qquad \Gamma \vdash \Delta, \beta}{\Gamma \vdash \Delta, \alpha \wedge \beta}\ (\text{R-}\wedge)$$

$$\frac{\Gamma, \alpha \vdash \Delta \qquad \Gamma, \beta \vdash \Delta}{\Gamma, \alpha \vee \beta \vdash \Delta}(\text{L-}\vee) \qquad \frac{\Gamma \vdash \Delta, \alpha, \beta}{\Gamma \vdash \Delta, \alpha \vee \beta}(\text{R-}\vee)$$

$$\frac{\Gamma, \neg\alpha \vdash \Delta \quad \Gamma \vdash \Delta, \alpha, \neg\beta \quad \Gamma, \beta \vdash \Delta}{\Gamma, \alpha \to \beta \vdash \Delta}\ (\text{L-}\to) \qquad \frac{\Gamma, \alpha \vdash \Delta, \beta \quad \Gamma, \neg\beta \vdash \Delta, \neg\alpha}{\Gamma \vdash \Delta, \alpha \to \beta}\ (\text{R-}\to)$$

$$\frac{\Gamma, \neg\alpha, \neg\beta \vdash \Delta}{\Gamma, \neg(\alpha \vee \beta) \vdash \Delta}\ (\text{L-}\neg\vee) \qquad \frac{\Gamma \vdash \Delta, \neg\alpha \qquad \Gamma \vdash \Delta, \neg\beta}{\Gamma \vdash \Delta, \neg(\alpha \vee \beta)}\ (\text{R-}\neg\vee)$$

$$\frac{\Gamma, \neg\alpha \vdash \Delta \qquad \Gamma, \neg\beta \vdash \Delta}{\Gamma, \neg(\alpha \wedge \beta) \vdash \Delta}\ (\text{L-}\neg\wedge) \qquad \frac{\Gamma \vdash \Delta, \neg\alpha, \neg\beta}{\Gamma \vdash \Delta, \neg(\alpha \wedge \beta)}\ (\text{R-}\neg\wedge)$$

$$\frac{\Gamma, \neg\beta \vdash \Delta, \neg\alpha}{\Gamma, \neg(\alpha \to \beta) \vdash \Delta}\ (\text{L-}\neg\to) \qquad \frac{\Gamma, \neg\alpha \vdash \Delta \qquad \Gamma \vdash \Delta, \neg\beta}{\Gamma \vdash \Delta, \neg(\alpha \to \beta)}\ (\text{R-}\neg\to)$$

$$\frac{\Gamma \vdash \Delta, \neg\alpha}{\Gamma, \neg\neg\alpha \vdash \Delta}\ (\text{L-}\neg\neg) \qquad \frac{\Gamma, \neg\alpha \vdash \Delta}{\Gamma \vdash \Delta, \neg\neg\alpha}\ (\text{R-}\neg\neg)$$

Rules for quantified formulas: The Gentzen system works over the domain V, a denumerable set of variables (or parameters); that is, the introduction of quantifiers is always made from variables, not from terms of the original language. In the following rules, y is a *fresh* variable, i.e. a variable which does not occur free in $\Gamma \cup \Delta$ and $\tau \in Terms(C \cup V, F)$:

$$\frac{\Gamma, y = \tau, \varphi(y) \vdash \Delta}{\Gamma, \exists x\varphi(x) \vdash \Delta}(\text{R-}\exists), \qquad \frac{\Gamma, y = \tau \vdash \Delta, \varphi(y)}{\Gamma \vdash \Delta, \forall x\varphi(x)}(\text{L-}\forall),$$

The atoms $y = \tau$ in the left-hand side introduce the elements y of the domain D_h. In the following rules, y may be any variable in V (not necessarily fresh), but we also need to include the atom $y = \tau$ in the left-hand side.

$$\frac{\Gamma, y = \tau, \varphi(y), \forall x\varphi(x) \vdash \Delta}{\Gamma, y = \tau, \forall x\varphi(x) \vdash \Delta}(\text{R-}\forall), \qquad \frac{\Gamma, y = \tau \vdash \Delta, \varphi(y), \exists x\varphi(x)}{\Gamma, y = \tau \vdash \Delta, \exists x\varphi(x)}(\text{L-}\exists)$$

Substitution rules: If τ_1, τ_2 are terms in $\mathcal{T}erms(C \cup V, F)$:

$$\frac{\Gamma, \tau_1 = \tau_2, \varphi(\tau_1) \vdash \Delta}{\Gamma, \tau_1 = \tau_2, \varphi(\tau_2) \vdash \Delta}; \quad \frac{\Gamma, \tau_1 = \tau_2 \vdash \Delta, \varphi(\tau_1)}{\Gamma, \tau_1 = \tau_2 \vdash \Delta, \varphi(\tau_2)};$$

$$\frac{\Gamma, \neg\varphi(\tau_1) \vdash \Delta, \neg(\tau_1 = \tau_2)}{\Gamma, \neg\varphi(\tau_2) \vdash \Delta, \neg(\tau_1 = \tau_2)}; \quad \frac{\Gamma \vdash \Delta, \neg(\tau_1 = \tau_2), \neg\varphi(\tau_1)}{\Gamma \vdash \Delta, \neg(\tau_1 = \tau_2), \neg\varphi(\tau_2)};$$

Strictness rule (left side): The property (F4) for interpretations establishes the strictness of the assignment mapping, i.e. if a term τ is defined, every subterm τ' is also defined. The syntactic rule for this property is the following one:

$$\frac{\Gamma, x = \tau, y = \tau' \vdash \Delta}{\Gamma, x = \tau \vdash \Delta} \tag{3}$$

The previous set of rules is basic for systems built to characterize free logics. The rest of the rules are specific for our system.

Additional rule for equality: By (F2)-a, two distinct elements in D_h cannot be equivalent in D_t. The property (F2)-a is syntactically characterized by the rule

$$\frac{\Gamma, x = y \vdash \Delta}{\Gamma, x = y \vdash \Delta, \neg(x = y)} \tag{4}$$

On the other hand, by the property (F2)-b, every element of D_t, must be equivalent to one from D_h. The atom $\neg(y = \tau)$ in the right-hand side introduces the element y of the domain D_t, but does not determine any relation with D_h. So, to comply with property (F2)-b, we need to modify the standard rules for negated quantified formulas and strictness.

Rules for negated quantified formulas: In the following rules, y, z are fresh variables and $\tau \in Terms(C \cup V, F)$:

$$\frac{\Gamma, z = \tau, \neg\varphi(y) \vdash \Delta, \neg(y = \tau)}{\Gamma, \neg\forall x\varphi(x) \vdash \Delta}(\text{R-}\neg\forall), \qquad \frac{\Gamma, z = \tau \vdash \Delta, \neg\varphi(y), \neg(y = \tau)}{\Gamma \vdash \Delta, \neg\exists x\varphi(x)}(\text{L-}\neg\exists)$$

The literal $\neg(y = \tau)$ says that y is a new element of D_t equivalent to τ, and the presence of the atom $z = \tau$ in the left-hand side says that τ is an element of D_h, as required by property (F2)-b; if we drop the condition (F2)-b in our models, these atoms in the left-hand sides of these rules must be also dropped.

In the following rules, y may be any variable in V (not necessarily fresh).

$$\frac{\Gamma, \neg\varphi(y), \neg\exists x\varphi(x) \vdash \Delta, \neg(y = \tau)}{\Gamma, \neg\exists x\varphi(x) \vdash \Delta, \neg(y = \tau)}(\text{R-}\neg\exists)$$

$$\frac{\Gamma \vdash \Delta, \neg(y = \tau), \neg\varphi(y), \neg\forall x\varphi(x)}{\Gamma \vdash \Delta, \neg(y = \tau), \neg\forall x\varphi(x)}(\text{L-}\neg\forall)$$

Strictness rule (right side): Let τ' below be a subterm of τ and y, z fresh variables. We add the atom $z = \tau'$ in the left-hand side to comply with (F2)-b.

$$\frac{\Gamma, z = \tau' \vdash \Delta, \neg(x = \tau), \neg(y = \tau')}{\Gamma \vdash \Delta, \neg(x = \tau)} \tag{5}$$

Auxiliary parameters elimination: As we have said, the quantifier rules only work with variables and thus, to prove formulas involving terms, these terms must be assigned to variables. This is done by auxiliary parameters elimination rules we denote as (ParEl). In the following rules, α is either a predicate symbol or the equality, every τ_i is a term in $Terms(C \cup V, F)$, and $x, x_1, \ldots, x_n$ are variables.

$$\frac{\Gamma, \alpha(x_1, \ldots, x_n), x_1 = \tau_1, \ldots, x_n = \tau_n \vdash \Delta}{\Gamma, \alpha(\tau_1, \ldots, \tau_n) \vdash \Delta}$$

$$\frac{\Gamma, x_1 = \tau_1, \ldots, x_n = \tau_n \vdash \alpha(x_1, \ldots, x_n), \Delta \quad \Gamma \vdash \Delta, \exists x(x = \tau_i), i = 1..n}{\Gamma \vdash \alpha(\tau_1, \ldots, \tau_n), \Delta}$$

$$\frac{\Gamma \vdash \neg\alpha(x_1, \ldots, x_n), \neg(x_1 = \tau_1), \ldots, \neg(x_n = \tau_n), \Delta}{\Gamma \vdash \neg\alpha(\tau_1, \ldots, \tau_n), \Delta}$$

$$\frac{\Gamma\neg\alpha(x_1, \ldots, x_n) \vdash \neg(x_1 = \tau_1), \ldots, \neg(x_n = \tau_n), \Delta \quad \Gamma, \neg\exists x(x = \tau_i) \vdash \Delta, i = 1..n}{\Gamma, \neg\alpha(\tau_1, \ldots, \tau_n) \vdash \Delta}$$

Example 2. The inference $p(a) \vdash \exists x(x = a)$ is provable in FHT,

$$\dfrac{\dfrac{p(y), y = a \vdash y = a, \exists x(x = a)}{p(y), y = a \vdash \exists x(x = a)}\ (\text{L-}\exists)}{p(a) \vdash \exists x(x = a)}\ (\text{ParEl})\ (\text{Axiom})$$

because the truth of the atom $p(a)$ in a model requires that a is defined. This is a consequence of the condition (F6), $I(p,h) \subseteq D_h$, and syntactically of the auxiliary parameters elimination rules. However, the inference $\neg p(a) \vdash \exists x(x = a)$ is not provable. If we try to construct a proof applying the rules upwards we can deduce how to build a counterexample.

$$\dfrac{\dfrac{\dfrac{\neg p(a) \vdash \exists x(x = a)}{y = y, \neg p(y) \vdash \neg(y = a), \exists x(x = a)}}{y = y, \neg p(y) \vdash \neg(y = a), y = a, \exists x(x = a)} \qquad \dfrac{}{\neg\exists x(x = a) \vdash \exists x(x = a)}$$

In the first step, we apply the parameter elimination rule; we would need to add the atom $y = y$ because y is a fresh variable and we need to define it as an element of D_h. In the second step we apply R-$\exists$; note that we would need the presence of the atom $y = y$ in the left-hand side to apply this rule. The sequent in the left branch is not open and it can not be generated from other sequents (the rule R-$\exists$ has been applied using the unique parameter in the sequent). Moreover, it is easy to construct a countermodel of this sequent,

$$D_h = \{y\}, \quad D_t = \{y, a\}, \quad y \equiv a, \quad I(p,h) = I(p,t) = \varnothing$$

which also is a countermodel of $\neg p(a) \vdash \exists x(x = a)$.

Theorem 3 (Soundness). *If Γ and Δ are lists of formulas such that $\Gamma \vdash \Delta$ is deducible in* **FHTG***, and $\mathcal{I}$ is a model of Γ, then $\mathcal{I}$ is a model of a formula $\psi \in \Delta$. In particular, if $\Gamma \vdash \varphi$, then $\Gamma \models \varphi$*

As usual, the soundness proof consists in verifying that every rule preserves the satisfiability of sequents.

Theorem 4 (Completeness). *If Γ and Δ are lists of formulas such that for every model $\mathcal{I}$ of Γ there exists $\psi \in \Delta$ such that $\mathcal{I}$ is a model of ψ, then $\Gamma \vdash \Delta$ is deducible in* **FHTG***. In particular, if $\Gamma \models \varphi$, then $\Gamma \vdash \varphi$.*

5 Conclusions

We have provided an alternative characterisation of (the monotonic basis for) Cabalar's semantics for partial intensional functions based on free logic. This characterisation allows us to establish a Gentzen calculus that can be used, for instance, to check strong equivalence properties or make formal analysis for theories involving partial functions. With respect to Cabalar's original approach, the current free-logic variant is more flexible: it can be modified in various ways by relaxing some of the conditions we had to impose to capture Cabalar's approach. Another interesting topic is the comparison to Flexible QHT and its Gentzen calculus presented in [12] whose main differences rely on the treatment of equality. We will study a formal comparison and explore the possibility of capturing both Cabalar's and BL functions in the same formal framework.

References

1. Mario Alviano, Francesco Calimeri, Günther Charwat, Minh Dao-Tran, Carmine Dodaro, Giovambattista Ianni, Thomas Krennwallner, Martin Kronegger, Johannes Oetsch, Andreas Pfandler, Jörg Pührer, Christoph Redl, Francesco Ricca, Patrik Schneider, Martin Schwengerer, Lara Katharina Spendier, Johannes Peter Wallner, and Guohui Xiao. The fourth answer set programming competition: Preliminary report. In Pedro Cabalar and Tran Cao Son, editors, *Proc. of the 12th Intl. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR 2013), Corunna, Spain, September 15-19*, volume 8148 of *Lecture Notes in Computer Science*, pages 42–53. Springer, 2013.
2. Marcello Balduccini. A "conservative" approach to extending answer set programming with nonherbrand functions. In Esra Erdem, Joohyung Lee, Yuliya Lierler, and David Pearce, editors, *Correct Reasoning*, pages 24–39. Springer-Verlag, 2012.
3. Michael Bartholomew and Joohyung Lee. Stable models of formulas with intensional functions. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR'12)*, pages 2–12, 2012.
4. Michael Bartholomew and Joohyung Lee. On the stable model semantics for intensional functions. In *Proceedings of International Conference on Logic Programming (ICLP'13)*, 2013.
5. Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming at a glance. *Commun. ACM*, 54(12):92–103, 2011.
6. Tyler Burge. Truth and singular terms. *Nous*, 8(4):309–325, 1974.
7. Pedro Cabalar. Functional answer set programming. *Theory and Practice of Logic Programming*, 10(2-3):203–233, 2011.
8. Pedro Cabalar and David Lorenzo. Logic programs with functions and default values. In *Proc. of the 9th European Conf. on Logics in AI (JELIA'04) (LNCS 3229)*, pages 294–306, 2004.
9. Pedro Cabalar, David Pearce, and Agustín Valverde. A revised concept of safety for general answer set programs. In Esra Erdem, Fangzhen Lin, and Torsten Schaub, editors, *Proc. of the 10th Intl. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR 2009), Potsdam, Germany, September 14-18*, volume 5753 of *Lecture Notes in Computer Science*, pages 58–70. Springer, 2009.
10. Francesco Calimeri, Susanna Cozza, Giovambattista Ianni, and Nicola Leone. Computable functions in ASP: Theory and implementation. In *24th Intl. Conf. on Logic Programming*, volume 5366 of *Lecture Notes in Computer Science*, pages 407–424. Springer-Verlag, 2008.
11. Francesco Calimeri, Susanna Cozza, Giovambattista Ianni, and Nicola Leone. An ASP system with functions, lists, and sets. In *10th Intl. Conf. on Logic Programming and Nonmonotonic Reasoning*, volume 5753 of *Lecture Notes in Computer Science*, pages 483–489. Springer-Verlag, 2009.
12. Luis Fariñas del Cerro, David Pearce, and Agustín Valverde. FQHT: The logic of stable models for logic programs with intensional functions. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI'13)*, 2013.
13. P. Ferraris, J. Lee, and V. Lifschitz. A new perspective on stable models. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 372–379, 2007.
14. Dov M. Gabbay, David Pearce, and Agustín Valverde. Interpolable formulas in equilibrium logic and answer set programming. *Journal of Artificial Intelligence Research (JAIR)*, 42:917–943, 2011.

15. Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Proc. of the 5th Intl. Conf. on Logic Programming*, pages 1070–1080, 1988.
16. Michael Hanus. The integration of functions into logic programming: from theory to practice. *Journal of Logic Programming*, 19,20:583–628, 1994.
17. Arend Heyting. Die formalen Regeln der intuitionistischen Logik. *Sitzungsberichte der Preussischen Akademie der Wissenschaften, Physikalisch-mathematische Klasse*, pages 42–56, 1930.
18. Vladimir Lifschitz. Logic programs with intensional functions. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR'12)*, 2012.
19. Vladimir Lifschitz, David Pearce, and Agustín Valverde. A characterization of strong equivalence for logic programs with variables. In *Proc. of the 9th Intl. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*, pages 188–200, 2007.
20. Fangzhen Lin and Yisong Wang. Answer set programming with functions. In *Proc. of the 11th Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR'08)*, 2008.
21. V. Marek and M. Truszczyński. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a 25-Year Perspective*, pages 169–181. Springer-Verlag, 1999.
22. I. Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25:241–273, 1999.
23. David Pearce. A new logical characterisation of stable models and answer sets. In *Non monotonic extensions of logic programming. Proc. NMELP'96. (LNAI 1216)*. Springer-Verlag, 1996.
24. David Pearce and Agustín Valverde. Towards a first order equilibrium logic for nonmonotonic reasoning. In *Proc. of the 9th European Conf. on Logics in AI (JELIA'04)*, pages 147–160, 2004.
25. David Pearce and Agustín Valverde. Synonymous theories and knowledge representations in answer set programming. *Journal of Computer and System Sciences*, 78(1):86–104, 2012.
26. Carl J. Posy. A free IPC is a natural logic: Strong completeness for some intuitionistic free logics. *Topoi*, 1(1-2):30–43, 1982.