Lower Bound Founded Logic of Here-and-There: A Preliminary Report

Pedro Cabalar  
University of Corunna, Spain  
cabalar@udc.es

Jorge Fandinno  
University of Toulouse, France  
jorge.fandinno@irit.fr

Torsten Schaub and Sebastian Schellhorn  
University of Potsdam, Germany  
{torsten, seschell}@cs.uni-potsdam.de

Abstract

A distinguishing feature of Answer Set Programming is that all atoms belonging to a stable model must be founded. That is, an atom must not only be true but provably true. This can be made precise by means of the constructive logic of Here-and-There, whose equilibrium models correspond to stable models. One way to look at-foundedness is to regard Boolean truth values as ordered by letting true be greater than false. Then, each Boolean variable takes the smallest truth values that can be proven for it. This idea was generalized by Aziz to ordered domains and applied to constraint satisfaction problems. As before, the idea is that a, say integer, variable gets only assigned to the smallest integer that can be justified.

In this paper, we present a logical reconstruction of Aziz’ idea in the setting of the logic of Here-and-There. More precisely, we start by defining the logic of Here-and-There with lower bound founded variables along with its equilibrium models and elaborate upon their formal properties. We then define a logic program fragment dealing with linear constraints over integers and analyze it in terms of concepts from logic programming. Finally, we compare our approach with related ones and sketch future work.

1 Motivation

A distinguishing feature of Answer Set Programming (ASP; Baral 2003) is that all atoms belonging to a stable model must be founded. That is, an atom must not only be true but provably true. This can be made precise by means of the constructive logic of Here-and-There (HT; Heyting 1930), whose equilibrium models correspond to stable models (Pearce 2006). One way to look at-foundedness is to regard Boolean truth values as ordered by letting true be greater than false. Then, each Boolean variable takes the smallest truth values that can be proven for it. This idea was generalized in (Aziz 2015) to ordered domains and applied to constraint satisfaction problems. As before, the idea is that a, say integer, variable gets only assigned to the smallest integer that can be justified. We refer to this idea by calling it foundedness.

The literature of ASP contains several approaches dealing with atoms containing variables over non-Boolean domains, among them (Baselice, Bonatti, and Gelfond 2005), (Janhunen et al. 2017) and (Cabalar et al. 2016), but these approaches do not address foundedness in our sense. For instance, Constraint ASP (CASP) approaches like (Baselice, Bonatti, and Gelfond 2005) allow atoms with variables over non-Boolean domains in the body of a rule only. Thus, these atoms and the values of non-Boolean variables cannot be founded in terms of ASP.

Approaches like (Janhunen et al. 2017) and (Cabalar et al. 2016) allow any kind of atoms in heads and bodies. This allows atoms with variables over non-Boolean domains to be founded but their variables are not necessarily assigned to the smallest value that can be justified. Since in the approach of (Cabalar et al. 2016) atoms are founded and defaults are possible, one could think about to use defaults or minimization to achieve foundedness. For instance, \( x = 3 \leftarrow \neg(x \neq 3) \) assigns value 3 to \( x \) by default. If we add fact \( x = 1 \), then we deactivate the default and assign value 1 to \( x \). Similarly, \( x \geq 0 \leftarrow \neg(x < 0) \) assigns some arbitrary value greater or equal than 0 by default. However, assigning a minimal value by default cannot be done by rules as the above. To point out the difference of foundedness and founded atoms, the following examples illustrate that minimizing assigned values does not restore foundedness either. Consider the rules

\[
\begin{align*}
x & \geq 0 \\
y & \geq 0 \\
x & \geq 42 \leftarrow y < 42
\end{align*}
\]

The approach of (Cabalar et al. 2016) leads to solutions that assign values greater or equal than 42 to \( x \) and values greater or equal than 0 to \( y \) or vice versa, respectively. Thus, the two solutions with minimal values assign 42 to \( x \) and 0 to \( y \) and the other way around. Note that only the first one respects foundedness, since there is no reason to assign a value greater than 0 to \( y \). Now, consider the rules

\[
\begin{align*}
x & \geq 1 \\
x & \geq 42 \leftarrow \neg(x \leq 1)
\end{align*}
\]

We expect two solutions in terms of foundedness. One assigns the value 1 to \( x \) and the other assigns value 42 to \( x \), since a value greater than 1 forces the derivation of value 42. In general, the rules of (2) give us no reason to derive a value greater than 42. In contrast, the approach presented in (Cabalar et al. 2016) yield an intuitive understanding assigning value 1 or a value greater or equal than 42 to \( x \). That is, the corresponding solution with the minimal value assigned to \( x \) assigns 1 to \( x \). The second equally founded solution is not obtained.

The existing approach regarding foundedness of (Aziz 2015) behaves counter intuitive. For instance, consider rule
Let $\mathcal{A}$ be the set of propositional atoms. A formula $\varphi$ is a combination of atoms by logical connectives $\top, \land, \lor$, and $\iff$. As usual, we define $\top \iff \bot \iff \bot$ and $\neg \varphi \iff \varphi \iff \bot$. A theory is a set of formulas.

We denote an interpretation over $\mathcal{A}$ by $I \subseteq \mathcal{A}$ and an $HT$-interpretation over $\mathcal{A}$ by $(H, T)$ where $H \subseteq T \subseteq \mathcal{A}$ are interpretations. Since we want to abstract from the specific form of atoms in the following sections, we rely upon denotations for fixing their semantics. A denotation of atoms in $\mathcal{A}$ is a function $\| \cdot \|_A : \mathcal{A} \rightarrow 2^A$ mapping atoms in $\mathcal{A}$ to sets of interpretations over $\mathcal{A}$. Accordingly, $\| p \|_A \equiv \{ I \mid p \in I \}$ represents the sets of interpretations where atom $p$ holds.

With it, we next define satisfaction of formulas in $HT$.

**Definition 1** Let $(H, T)$ be an $HT$-interpretation over $\mathcal{A}$ and $\varphi$ a propositional formula over $\mathcal{A}$. Then, $(H, T)$ satisfies $\varphi$, written $(H, T) \models \varphi$, if the following conditions hold:

1. $(H, T) \not\models \bot$
2. $(H, T) \models p$ if $p \in \| p \|_A$ for propositional atom $p \in \mathcal{A}$
3. $(H, T) \models \varphi_1 \land \varphi_2$ if $(H, T) \models \varphi_1$ and $(H, T) \models \varphi_2$
4. $(H, T) \models \varphi_1 \lor \varphi_2$ if $(H, T) \models \varphi_1$ or $(H, T) \models \varphi_2$
5. $(H, T) \models \varphi_1 \rightarrow \varphi_2$ if $(I, T) \not\models \varphi_1$ or $(I, T) \models \varphi_2$ for both $I \in \{ H, T \}$

As usual, we call $(H, T)$ an $HT$-model of a theory $\Gamma$, if $(H, T) \models \varphi$ for all $\varphi$ in $\Gamma$. The usual definition of $HT$ satisfaction (cf. Pearce 2006) is obtained by replacing Condition 2 above by

2'. $(H, T) \models p$ if $p \in H$ for propositional atom $p \in \mathcal{A}$

It is easy to see that both definitions of $HT$ satisfaction coincide.

**Proposition 1** Let $(H, T)$ be an $HT$-interpretation and $\varphi$ a formula over $\mathcal{A}$. Then, $(H, T) \models \varphi$ if $(H, T) \models \varphi$ by replacing Condition 2 by 2'.

As usual, an equilibrium model of a theory $\Gamma$ is a (total) $HT$-interpretation $(T, T)$ such that $(T, T) \models \Gamma$ and there is no $H \subset T$ such that $(H, T) \models \Gamma$.

### 3 Lower Bound Founded Logic of Here-and-There

In what follows, we introduce the logic of Here-and-There with lower bound founded variables, short $HT_{LB}$ and elaborate on some formal properties regarding satisfaction. We discuss the relation of complements of atoms regarding negation and we point out the relation between $HT_{LB}$ and $HT$ as well as a straightforward extension of Ferraris’ stable model semantics.

#### 3.1 $HT_{LB}$ and its Properties

The language of $HT_{LB}$ is defined over a set of atoms $\mathcal{A}_X$ comprising variables, $X$, and constants over an ordered domain $(D, \geq)$. For simplicity, we assume that each element of $D$ is uniquely represented by a constant and we refer to the set of constants. Similarly, we identify $\geq$ with its syntactic representative. The specific syntax of atoms is left open but assumed to refer to elements of $X$ and $D$. The only requirement is that an atom depends on a distinguished subset of variables of $X$. An atom can be understood to hold or not once all variables depending on it are substituted by domain elements. Intuitively, variables not occurring in an atom are understood as irrelevant for the atom evaluation. Examples of ordered domains are $\{(0, 1, 2, 3), \geq\}$ and $\{(Z, \geq), \}$; corresponding atoms are $x \geq 42$ and $x = y$. A formula $\varphi$ is a propositional combination of atoms and logical connectives $\land, \lor, \rightarrow$. As usual, we define $\top \rightarrow \bot \rightarrow \bot$ where $\bot \equiv \bot$.

For capturing partiality, we introduce a special domain element $u$, standing for undefined, and extend $(D, \geq) \cup \{ u \}$ to $(D_u, \geq_u)$ where $D_u \equiv D \cup \{ u \}$ and $\geq_u \equiv \geq \cup \{ \{ c, u \} \mid c \in D \}$. With it, we define a (partial) valuation over $X, D$ as a function $v : X \rightarrow D_u$, mapping each variable to a domain value or undefined. For comparing valuations by set-based means, we alternatively represent them by subsets of $X \times D$. Basically, any function $v$ is a set of pairs $(x, c)$ such that $v(x) = c$ for $c \in D$. In addition, we view a pair $(x, c)$ as $x \geq c$ and add its downward closure $(x, c) \equiv \{ (x, d) \mid c, d \in D, c \geq d \}$. Given this, a valuation $v$ is represented by the set $\bigcup_{v(x) = c} \{ (x, c) \}$. As an example, consider variables $x$ and $y$ over domain $\{(0, 1, 2, 3) \cup \{ u \}, \geq_u\}$. The valuation $v = \{ x \rightarrow 2, y \rightarrow 0 \}$ can be represented by $v' = \{ (x, 2) \cup (y, 0) \equiv \{ (x, 0), (x, 1), (x, 2), (y, 0) \} \}. Then, v' = \{ x \rightarrow 1, y \rightarrow u \}$, viz. $\{ (x, 0), (x, 1) \}$ in set notation, can be regarded as “smaller” than $v$ because $v' \subseteq v$. The comparison of two valuations $v$ and $v'$ by their set-based means using $\subseteq$ amounts to a twofold comparison. That is, $v$ and $v'$ are compared regarding the occurrence of variables and their particular values wrt $\geq$. We let $D_{X,D}$ stand for the set of valuations over $X$ and $D$.

We define the satisfaction of formulas over $A_X$ wrt atom denotations over $X, D$, which are functions $\| \cdot \|_{X,D} : A_X \rightarrow \{ 0, 1 \}$.
2^\Omega x \phi$ mapping atoms to sets of valuations. Let $a$ be an atom of $\mathcal{A}_X$ and $[a]_{X,D}$ its denotation. Then, $[a]_{X,D}$ is the set of valuations $v$ so that $a$ holds. Since $a$ depends on variables $\text{vars}(a) \subseteq X$, we have for each $v \in [a]$ and valuation $v'$ with $v(x) = v'(x)$ for $x \in \text{vars}(a)$ that $v' \in [a]$. Intuitively, values of $X \setminus \text{vars}(a)$ can vary freely without changing the membership of a valuation to $[a]$. For simplicity, we drop indices $X,D$ whenever clear from context.

For instance, interpreting the atoms $x \geq 42, 0 \geq 42$ and $0 \geq 42$ over $(\mathbb{Z}, \geq)$ yields the following denotations:

$$
\begin{align*}
\llbracket x \geq 42 \rrbracket & \equiv \{ v \mid v(x) \geq 42 \} \\
\llbracket 0 \geq 42 \rrbracket & \equiv \emptyset \\
\llbracket 0 \geq 0 \rrbracket & \equiv \emptyset.
\end{align*}
$$

In particular, $\llbracket x \geq 42 \rrbracket$ is the set of valuations where $x$ is assigned to a value greater or equal to 42 and all variables in $X \setminus \text{vars}(x \geq 42)$ take any value of $\mathcal{D}_u$, eg $(x \downarrow 45)$ and $(x \downarrow 45) \cup (y \downarrow 0)$ for $y \in X \setminus \text{vars}(x \geq 42)$ are possible valuations. Interestingly, atoms like $x \geq 42$ with $\llbracket x \geq x \rrbracket = \{ v \mid v(x) \neq u \}$ force variables to be defined over $\mathcal{D}$ per definition of $\geq$. A valuation $v$ is defined for a set of variables $\mathcal{Y} \subseteq X$ if $v(x) \neq u$ for all $x \in \mathcal{Y}$.

We define an $HT_{LB}$-valuation over $X,D$ as a pair $(h,t)$ of valuations over $X,D$ with $h \subseteq t$. We define satisfaction of a formula wrt an $HT_{LB}$-valuation as follows.

**Definition 2** Let $(h,t)$ be an $HT_{LB}$-valuation over $X,D$ and $\phi$ be a formula over $\mathcal{A}_X$. Then, $(h,t)$ satisfies $\phi$, written $(h,t) \models \phi$, if the following holds:

1. $(h,t) \not\models \bot$
2. $(h,t) \models a$ iff $v \in [a]_{X,D}$ for atom $a \in \mathcal{A}_X$ and for both $v \in \{h,t\}$
3. $(h,t) \models \phi_1 \land \phi_2$ iff $(h,t) \models \phi_1$ and $(h,t) \models \phi_2$
4. $(h,t) \models \phi_1 \lor \phi_2$ iff $(h,t) \models \phi_1$ or $(h,t) \models \phi_2$
5. $(h,t) \models \phi_1 \rightarrow \phi_2$ iff $(v,t) \not\models \phi_1$ or $(v,t) \models \phi_2$ for both $v \in \{h,t\}$

As usual, we call $(h,t)$ an $HT_{LB}$-model of a theory $\Gamma$, if $(h,t) \models \phi$ for all $\phi \in \Gamma$. For a simple example, consider the theory containing $x \geq 42$ only. Then, every $HT_{LB}$-valuation $(h,t)$ with $h,t \in \llbracket x \geq 42 \rrbracket$ is an $HT_{LB}$-model of $x \geq 42$. Note that, different to $HT$, satisfaction of atoms in $HT_{LB}$ forces satisfaction in both $h$ and $t$, instead of $h$ only. We discuss this in detail in Section 3.4.

Our first result shows that the characteristic properties of persistence and negation hold as well when basing satisfaction on valuations and denotations.

**Proposition 2** Let $(h,t)$ and $(t,t)$ be $HT_{LB}$-valuations over $X,D$, and $\phi$ be a formula over $\mathcal{A}_X$. Then,

1. $(h,t) \models \phi$ implies $(t,t) \models \phi$, and
2. $(h,t) \models \phi \rightarrow \perp$ iff $(t,t) \not\models \phi$.

Persistence implies that all atoms satisfied by $(h,t)$ are also satisfied by $(t,t)$. To make this precise, let $A(h,t) \equiv \{ a \in \mathcal{A}_X \mid h \in [a] \wedge t \in [a] \}$ be the set of atoms satisfied by $(h,t)$.

**Proposition 3** Let $(h,t)$ and $(t,t)$ be $HT_{LB}$-valuations over $X,D$. Then, $A((h,t)) \subseteq A((t,t))$.

Finally, we define an equilibrium model in $HT_{LB}$.

**Definition 3** An $HT_{LB}$-valuation $(t,t)$ over $X,D$ is an $HT_{LB}$-equilibrium model of a theory $\Gamma$ iff $(t,t) \models \Gamma$ and there is no $h \subset t$ such that $(h,t) \models \Gamma$.

We refer an $HT_{LB}$-equilibrium model $(t,t)$ of $\Gamma$ as an $HT_{LB}$-stable model $t$ of $\Gamma$. Let us reconsider the theory containing atom $x \geq 42$ only. Then, $t = (x \downarrow 42)$ is an $HT_{LB}$-stable model of $x \geq 42$, since $t \in \llbracket x \geq 42 \rrbracket$ and there is no $h \subset t$ with $h \in \llbracket x \geq 42 \rrbracket$. In contrast, neither $HT_{LB}$-model $(t',t')$ with $t' = (x \downarrow 42) \cup (y \downarrow 0)$ nor $(t'',t'')$ with $t'' = (x \downarrow 53)$ are $HT_{LB}$-stable models since $t$ is a proper subset of both and $(t,t') \models x \geq 42$ as well as $(t,t'') \models x \geq 42$ holds. Hence, $HT_{LB}$-stable models make sure that each variable is assigned to its smallest founded value and does not take any value of possible valuations of corresponding denotations.

Note that $HT_{LB}$-equilibrium models induce the non-monotonic counterpart of the monotonic logic of $HT_{LB}$. Following well-known patterns, we show that $HT_{LB}$ allows us to decide strong equivalence wrt $HT_{LB}$-equilibrium models.

**Proposition 4 (Strong Equivalence)** Let $\Gamma_1, \Gamma_2$ and $\Gamma$ be theories over $\mathcal{A}_X$. Then, theories $\Gamma_1 \cup \Gamma_2$ and $\Gamma_2 \cup \Gamma$ have the same $HT_{LB}$-stable models for every theory $\Gamma$ iff $\Gamma_1$ and $\Gamma_2$ have the same $HT_{LB}$-models.
3.2 Negation in $HT_{LB}$

In the following, we elaborate on complements of atoms and its relation to negation, since $A_X$ may contain atoms like $x \geq 42$ and $x < 42$. Intuitively, one could expect that the strong negation of an atom holds whenever the atom itself does not hold. This can be easily expressed by defining the complement of valuations of an atom denotation. More formally, we characterize the complement $\pi$ of an atom $a$ by its denotation $[\pi] = \{v \mid v(a) = \mathbf{f}\}$.

To illustrate that the simple complement of an atom is not sufficient to yield something similar to strong negation let us take a closer look on propositional atoms in $HT_{LB}$. For mimicking Boolean truth values, we consider the domain $\{\{t, f\}, \{t \geq f\}\}$. Then, the denotation of propositional atoms in $HT_{LB}$ can be defined as follows: $[p = t]_{A_X(t, f)} \equiv \{v \mid v(p) = t\}$ and $[p = f]_{A_X(t, f)} \equiv \{v \mid v(p) = f\}$. Note that $p = t$ and $p = f$ are regarded as strong negations of each other, as in standard case (Gelfond and Lifschitz 1990); its weak negation are given by $\neg(p = t)$ and $\neg(p = f)$, respectively. For instance, the complement $p = t$ is characterized by denotation $[p = t]_\tau = 2^{V} \setminus \{p = t\} = \{v \mid v(p) \neq t\}$. Note that this complement allows valuations $v$ with $v(p) = u$, which does not match $p = f$.

To this end, we define another complement to exclude assignments undefined to variables of the atom. First, we define a denotation $[a]$ of an atom $a$ as strict if each $v \in [a]$ is defined for $vars(a)$. Then, we characterize the strict complement $\pi^s$ of atom $a$ by the strict denotation $[\pi^s] = 2^{V} \setminus \bigcup \{a \mid \{v \mid v(x) = u \text{ for some } x \in vars(a)\}\}$. In-formally, the strict complement of an atom holds whenever all variables are defined and the atom itself does not hold. That is, atoms $p = f$ and $p = t$ are strict complements of each other.

More generally, an atom with strict denotation and its strict complement can be regarded as being strongly negated to each other. For instance, consider atom $x \geq 42$ and its strict denotation $[x \geq 42] = \{v \mid v(x) \geq 42\}$. Then, its strict counterpart $x \geq 42^s$ is defined by $[x \geq 42^s] = \{v \mid v(x) < 42\}$. As in the Boolean case, the strict counterpart $x \geq 42^s$ can be seen as the strong negation of $x \geq 42$.

To make the relation of complements and negation precise, let us define entailments. A theory (or a single formula) $\Gamma$ over $A_X$ entails a formula $\varphi$ over $A_X$, written $\Gamma \models \varphi$, when all $HT_{LB}$-models of $\Gamma$ are $HT_{LB}$-models of $\varphi$. Then, we have the following result.

**Proposition 5** Let $a$ be an atom over $A_X$, and $\pi$ and $\pi^s$ its complement and its strict complement over $A_X$, respectively. Then, $\pi^s \models \pi$ and $\pi \models \sim a$.

This implies that the strict complement $\pi^s$ of an atom $a$ implies its negation $\sim a$, just as strong negation implies weak negation in the standard case (Pearce 2006). To illustrate that in general the negation of an atom does not entail its complement ($\sim a \not\models \pi$), let us consider atom $x \leq 42$ with strict denotation $[x \leq 42] = \{v \mid v(x) \leq 42\}$. Then, the complement $x \leq 42^s$ is defined by denotation $[x \leq 42^s] = 2^{V} \setminus \{x \leq 42\} = \{v \mid v(x) = u \text{ or } v(x) > 42\}$. For valuations $h = (x \downarrow 42)$ and $t = (x \downarrow 50)$ we have that $\langle h, t \rangle \models \sim(x \leq 42)$ since $(x \downarrow 50) \not\in [x \leq 42]$. In contrast, $\langle h, t \rangle \models x \leq 42$ does not hold, since $(x \downarrow 42) \not\subseteq [x \leq 42]$. Thus, the complement $\pi$ of an atom $a$ can be seen as a kind of negation in between of strong and weak negation.

3.3 $HT_{LB}$ versus $HT$

Analogously to (Cabalar et al. 2016), we next show that $HT$ can be seen as a special case of $HT_{LB}$.

Note that both types of denotations $[p]_A$ and $[p = t]_{A(t, f)}$ of a propositional atom $p$ collect interpretations and valuations assigning true to $p$, respectively. To this end, we define a transformation $\tau$ relating each propositional atom $p$ with corresponding atom $p = t$ by $\tau(p) \equiv p = t$. Let $\Gamma$ be a propositional theory, then $\tau(\Gamma)$ is obtained by substituting each $p \in atoms(\Gamma)$ by $\tau(p)$. Moreover, we extend $\tau$ to interpretations $I$ by $\tau(I) \equiv \{(p, t) \mid p \in I\}$ to obtain a corresponding valuation over $A_X, \{t\}$. The next proposition establishes that $HT$ can be seen as a special case of $HT_{LB}$.

**Proposition 6** Let $\Gamma$ be a theory over propositional atoms $A$ and $(H, T)$ an $HT$-interpretation over $A$. Let $\tau(\Gamma)$ be a theory over atoms $\{p = t \mid p \in A\}$ and $\tau(\Gamma)$ an $HT_{LB}$-valuation over $A_X, \{t\}$. Then, $(H, T) \models \Gamma$ iff $(\tau(H), \tau(T)) \models \tau(\Gamma)$.

This can be generalized to any arbitrary singleton domain $\{d\}$ and corresponding atoms $p = d$ and the relationship still holds.

We obtain the following results relating $HT_{LB}$ and $HT$:

**Proposition 7** Let $\Gamma$ be a theory over $A_X$ and $(h, t)$ an $HT_{LB}$-model of $\Gamma$ over $X, D$. Then, $(\mathcal{A}(h, t), \mathcal{A}(t, t)))$ is an $HT$-model of $\Gamma$ over $A_X$.

That is, the collected atoms satisfied by an $HT_{LB}$-model of $\Gamma$ can be seen as an $HT$-model of $\Gamma$ by interpreting $A_X$ as propositional atoms. For instance, consider the theory containing only atom $x \neq y$ and its denotation $[x \neq y] = \{v \mid v(x) \neq v(y) \neq u\}$. Let $h = (x \downarrow 0) \cup (y \downarrow 4)$ and $t = (x \downarrow 0) \cup (y \downarrow 42)$ be valuations and hence $A(h, t) = A(t, t) = \{x \neq y\}$ interpretations. Then, $(h, t) \models x \neq y$ in $HT_{LB}$ and $(\mathcal{A}(h, t), \mathcal{A}(t, t))) \models x \neq y$ in $HT$.

Furthermore, we relate tautologies in $HT$ and $HT_{LB}$.

**Proposition 8** Let $\varphi$ be a tautology over $A$ and $\varphi'$ a formula over $A_X$ obtained by replacing all atoms in $\varphi$ by atoms of $A_X$. Then, $\varphi'$ is a tautology in $HT_{LB}$.

That is, tautologies in $HT$ are independent of any form of atoms.

3.4 $HT_{LB}$-stable versus Ferraris-style stable models

As mentioned, in Definition 2 satisfaction of atoms differs from $HT$ by forcing satisfaction in both $h$ and $t$, instead of $h$ only. This is necessary to satisfy persistence in $HT_{LB}$. In fact, let $HT_{LB}$-valuation $(h, t)$ satisfy atom $a$ in $A_X$, and by persistence $HT_{LB}$-valuation $(t, t)$ satisfies $a$ as well, but not necessarily each $HT_{LB}$-valuation $(v, t)$ with $h \subset v \subset t$ satisfies $a$. For instance, consider atom $x \not\in 42$ with $[x \not\in 42] = \{v \mid v(x) \neq 12\}$. Let $h = (x \downarrow 0)$ and $t = (x \downarrow 53)$ be valuations. Then, $(h, t) \models x \not\in 42$ and $(t, t) \models x \not\in 42$, but for $v = (x \downarrow 42)$ with $h \subset v \subset t$ we have $(v, t) \not\models x \not\in 42$.
A question that arises now from the above is whether \( HT_{LB} \) behaves as expected in terms of stable models semantics. To this end, we give a straightforward definition of classical satisfaction and of the reduct put by Ferraris in (Ferraris 2005) in our setting and show that equilibrium models correspond to stable models according to the resulting Ferraris’-like stable model semantics. We define the counterpart of classical satisfaction as follows.

**Definition 4** Let \( t \) be a valuation over \( \mathcal{X}, \mathcal{D} \) and \( \varphi \) a formula over \( \mathcal{A}_X \). Then, \( t \) satisfies \( \varphi \), written \( t \models_{cl} \varphi \), if the following holds:

1. \( t \not|_{cl} \bot \)
2. \( t \models_{cl} \varphi \) iff \( t \models \varphi \) for atom \( a \in \mathcal{A}_X \)
3. \( t \models_{cl} \varphi_1 \land \varphi_2 \) iff \( t \models_{cl} \varphi_1 \) and \( t \models_{cl} \varphi_2 \)
4. \( t \models_{cl} \varphi_1 \lor \varphi_2 \) iff \( t \models_{cl} \varphi_1 \) or \( t \models_{cl} \varphi_2 \)
5. \( t \models_{cl} \varphi_1 \rightarrow \varphi_2 \) iff \( t \not|_{cl} \varphi_1 \) or \( t \models_{cl} \varphi_2 \)

We call \( t \) a classical model of a theory \( \Gamma \), if \( t \models_{cl} \varphi \) for all \( \varphi \in \Gamma \). We define a Ferraris-like reduct, short F-reduct, wrt atoms \( \mathcal{A}_X \) as follows.

**Definition 5** Let \( \varphi \) be a formula over \( \mathcal{A}_X \) and \( t \) a valuation over \( \mathcal{X}, \mathcal{D} \). Then, the F-reduct of \( \varphi \) over \( t \), written \( \varphi^t \), is given by

\[
\varphi^t = \begin{cases} 
\bot & \text{if } t \not|_{cl} \varphi \\
 a & \text{if } t \models_{cl} \varphi \text{ and } \varphi = a \text{ a atom of } \mathcal{A}_X \\
 \varphi_1^t \otimes \varphi_2^t & \text{if } t \models_{cl} \varphi_1 \land \varphi_2 \text{ and } \varphi = (\varphi_1 \land \varphi_2) \\
 & \text{for } \otimes \in \{ \land, \lor, \rightarrow \} 
\end{cases}
\]

For theory \( \Gamma \) and \( HT_{LB} \)-valuation \( t \), we define \( \Gamma^t \equiv \{ \varphi^t \mid \varphi \in \Gamma \} \). Note that in case of propositional atoms the F-reduct corresponds to Ferraris’ reduct. We define an F-stable model as expected according to classical satisfaction and the F-reduct above.

**Definition 6** A valuation \( t \) over \( \mathcal{X}, \mathcal{D} \) is an F-stable model of theory \( \Gamma \) over \( \mathcal{A}_X \) iff \( t \models_{cl} \Gamma^t \) and there is no \( h \subset t \) such that \( h \models_{cl} \Gamma^t \).

The next propositions show that models in \( HT_{LB} \) can be alternatively characterized in the style of Ferraris, rephrasing (Ferraris 2005, Lemma 1):

**Proposition 9** Let \( \langle h, t \rangle \) be an \( HT_{LB} \)-valuation over \( \mathcal{X}, \mathcal{D} \) and \( \Gamma \) a theory over \( \mathcal{A}_X \). Then, \( h \models_{cl} \Gamma^t \) iff \( \langle h, t \rangle \models \Gamma \).

As a special case, we obtain that every \( HT_{LB} \)-stable model corresponds to an F-stable model and vice versa.

**Corollary 1** Let \( t \) be a valuation over \( \mathcal{X}, \mathcal{D} \) and \( \Gamma \) a theory over \( \mathcal{A}_X \). Then, \( t \) is an \( HT_{LB} \)-stable model of \( \Gamma \) iff \( t \) is an F-stable model of \( \Gamma \).

The last two results have shown that our logic follows well known patterns wrt different representations of stable models.

### 4 Bound Founded Programs with Linear Constraints

In this section, we focus on atoms representing linear constraints over integers and analyze them in terms of concepts known from ASP. Due to space limitations, we present proofs and some preliminaries needed for the following results in an extended version of this work. We illustrate the modelling capabilities of this fragment of \( HT_{LB} \) on an example of error diagnosis.

#### 4.1 Programs and its Properties

Reconsider the ordered domain of integers \((\mathbb{Z}, \geq)\). We define a linear constraint atom as

\[
\sum_{i=1}^{m} w_i x_i < k
\]

where \( w_i, k \in \mathbb{Z} \), \( x_i \in \mathcal{X} \) are distinct variables, and \( \leq \in \{\geq, \leq, \neq\} \) is a binary relation. By \( \mathcal{L}_X \) we denote the set of linear constraint atoms wrt \( \mathcal{X} \) and \( \mathbb{Z} \). The denotation of a linear constraint atom is given by

\[
\left\{ \sum_{i=1}^{m} w_i x_i < k \right\} \equiv \{ v \mid \sum_{i=1}^{m} w_i v(x_i) < k, \; v(x_i) \neq u \}.
\]

A linear constraint atom \( a \) and its negation \( \lnot a \) are called linear constraint literals. In the following, we just say atoms and literals.

We define logic programs as follows.

**Definition 7** A formula over \( \mathcal{L}_X \) is called a rule if it is of form

\[
a_1 \lor \ldots \lor a_n \leftarrow l_1 \land \ldots \land l_n', \quad \text{(3)}
\]

where \( a_i \) is an atom for \( 1 \leq i \leq n \) and \( l_j \) is a literal for \( 1 \leq j \leq n' \) both over \( \mathcal{L}_X \).

A logic program is a theory of rules of form (3). Following logic programming syntax, we use \( \lor \) and \( \land \) as alternative representations of \( \lor \) and \( \land \), respectively. Moreover, in this context we write \( \varphi_1 \leftarrow \varphi_2 \) for \( \varphi_2 \rightarrow \varphi_1 \) for formulas \( \varphi_1 \) and \( \varphi_2 \). Examples of programs over \( \mathcal{L}_X \) are given in the introduction.

Let \( r \) be a rule of form (3). Then, we define by \( \text{head}(r) \equiv \{ a_i \mid 1 \leq i \leq n \} \) and \( \text{body}(r) \equiv \{ l_j \mid 1 \leq j \leq n' \} \) the set of literals of the left and right hand side of \( r \), respectively. Whenever \( \text{body}(r) = \emptyset \), we then drop \( \leftarrow \) and call \( r \) fact. If \( \text{head}(r) = \emptyset \) we write \( 
abla \leftarrow l_1, \ldots, l_n \). Rules of latter form are called integrity constraints; they eliminate all models satisfying their body. The following result is related to integrity constraints.

**Proposition 10** Let \( P \) be a program over \( \mathcal{L}_X \) containing a rule of form \( a \leftarrow \lnot a \) and for each \( HT_{LB} \)-stable model \( v \) of \( P \setminus \{a \leftarrow \lnot a\} \) over \( \mathcal{X}, \mathcal{Z} \) we have that \( \langle v, v \rangle \not\models a \). Then, \( P \) has no \( HT_{LB} \)-stable model.

This proposition seems to be trivial, but we show in Section 5 that Aziz’ original approach does not satisfy this property.

In basic ASP, normal programs are of special interest, since their stable models are subset minimal.\(^3\) In the following, we define and study normal programs in terms of \( HT_{LB} \).

\(^2\)As usual, \( w_1 x_1 + \ldots + w_n x_n < k \) and \( w_1 x_1 + \ldots + w_n x_n > k \) can be expressed by \( w_1 x_1 + \ldots + w_n x_n \leq k - 1 \) and \( w_1 x_1 + \ldots + w_n x_n \geq k + 1 \), respectively.

\(^3\)The fact that stable models are subset minimal is also known as anti-chain property.
Similar to ASP, we force the conclusion of normal rules to be not ambiguous, thus forbidding for instance disjunctive heads. We restrict heads to include exactly one atom and additionally exactly one variable as well. For instance, let \( P \) be a program consisting of fact \( x + y \geq 42 \) over \( \{x, y\}, \mathbb{Z} \) only. Then, \( P \) has infinitely many stable models \( \{v \mid v(x) + v(y) = 42\} \), eg \((x \downarrow 0) \cup (y \downarrow 42)\) and \((x \downarrow 42) \cup (y \downarrow 0)\). Hence, \( P \) should not be a normal program.

To illustrate that it is not enough to restrict heads for defining normal programs, let us reconsider program \( P \) with rules (2) of the introduction. Then, \( P \) has stable models \((x \downarrow 1)\) and \((x \downarrow 42)\). Let us take a closer look on how to get them. First, we note that \( v_1 = (x \downarrow 1) \) and \( v_2 = (x \downarrow 42) \) are candidates of stable models, since both satisfy \( P \). It is easy to see that there is no \( v' \subseteq v_1 \) with \( v' \in \{x \geq 1\} \) and hence \( v_1 \) is a stable model of \( P \). Furthermore, consider valuation \( v'' \subseteq v_2 \). Then, \( \langle v'', v_2 \rangle \models x \geq 42 \iff x \leq 1 \) iff either both \( v'' \in \{x \geq 42\} \) and \( v_2 \in \{x \geq 42\} \) or \( v_2 \in \{x \leq 1\} \) holds. This boils down to \( v'' \in \{x \geq 42\} \), which implies that \( v'' \subseteq v_2 \) is contradicted. That is, \( v_2 \) is a stable model as well and \((x \downarrow 1) \subseteq (x \downarrow 42)\) holds. Hence, the stable models of \( P \) are not subset minimal, \( P \) should not be a normal program.

The issue shown in the previous example arises, due to the monotonicity of atoms. We define an atom \( a \) as monotonic (resp. anti-monotonic) wrt variable \( x \) if \( v \in \{a\} \) implies \( v' \in \{a\} \) for every valuation \( v' \) with \( v \subseteq v' \) (resp. \( v' \subseteq v \) with \( v'(x) \neq u \)), where \( v(y) = v'(y) \) for all \( y \in \text{vars}(a) \setminus \{x\} \).

We define an atom \( a \) as monotonic (resp. anti-monotonic) if it is monotonic (resp. anti-monotonic) wrt all variables in \( \text{vars}(a) \), and non-monotonic otherwise. Analogously, a program \( P \) is monotonic (resp. anti-monotonic) if all atoms occurring in it are monotonic (resp. anti-monotonic). We call a program \( P \) directed if no atom in it is non-monotonic.

Thus, we define normal programs as follows.

**Definition 8** A rule over \( \mathcal{L}_X \) is normal if it is of form
\[
a_0 \leftarrow a_1, \ldots, a_n, \neg a_{n+1}, \ldots, \neg a_{n'}
\]
where \( |\text{vars}(a_0)| = 1 \) and each atom \( a_i \) is monotonic for \( n + 1 \leq i \leq n' \).

A normal program is a set of rules of form (4). As the program in (2) illustrates, programs containing rule bodies with not monotonic atoms in the scope of negation, like \( \neg x \leq 1 \), may lead to stable models which are not subset minimal. As in ASP, we have that stable models of normal programs are subset minimal.

**Proposition 11** Let \( P \) be a normal program over \( \mathcal{L}_X \). Then, each \( HT_{LB} \)-stable model of \( P \) over \( X, \mathbb{Z} \) is subset minimal.

To elaborate more on the influence of atomic monotonicity on programs, let us consider the following example. Let \( P \) be a directed program, in which no atom occurs in the scope of negation:
\[
\begin{align*}
x &\geq 0 \\
y &\geq 0 \\
x &\geq 42 \iff y < 42 \\
y &\geq 42 \iff x < 42
\end{align*}
\]

Then, \( P \) has the two stable models \((x \downarrow 42) \cup (y \downarrow 0)\) and \((x \downarrow 0) \cup (y \downarrow 42)\). Compare this with the ASP program \( \{a \leftarrow \neg b, b \leftarrow \neg a\} \) formulating an “even loop” yielding stable models \( \{a\} \) and \( \{b\} \). Both programs behave similarly, since assigning \( x \) (or \( y \)) to 42 disables the foundedness of 42 for \( y \) (or \( x \)) in the same way as assigning \( a \) (or \( b \)) to true disables the foundedness of true for \( b \) (or \( a \)). That is, not monotonic atoms implicitly involve negation.

The previous example motivates us to define positive programs. To this end, we first define the positive and negative body of a rule. Let \( r \) be a normal rule of form (4), then we define the positive body of \( r \) as \( \text{body}^+(r) \equiv \{a_1 \mid 1 \leq i \leq n, a_i \text{ monotonic}\} \) and its negative body as \( \text{body}^-(r) \equiv \text{body}^+(r) \setminus \text{body}^+(r) \), respectively. That is, atoms like \( x < 42 \) not occurring in the scope of negation belong to the negative body, since they are not monotonic.

Then, we define positive programs as follows.

**Definition 9** A normal rule \( r \) over \( \mathcal{L}_X \) is positive if head \((r)\) is monotonic and \( \text{body}^-(r) = \emptyset \).

A positive program is a set of positive rules.

The following result shows that a positive program has a unique stable model, just as in ASP (Apt, Blair, and Walker 1987).

**Proposition 12** Let \( P \) be a positive program over \( \mathcal{L}_X \). Then, \( P \) has exactly one \( HT_{LB} \)-stable model over \( X, \mathbb{Z} \).

The proof follows the well-known idea of applying a fix point calculation using a continuous and monotonic operator.

In ASP, a program is stratified if it is free of recursion through negation (Apt, Blair, and Walker 1987), also referred to “negative loops”. This idea remains the same in case of \( HT_{LB} \). Note that we drop in this work the preliminaries needed for the following results, due to space limitations. That is, we give the definitions of dependency graph, loop, stratification and splitting in terms of \( HT_{LB} \) in an extended work of this version.

The next results generalize the calculation of a stable model to stratified programs.

**Proposition 13** Let \( P \) be a stratified program over \( \mathcal{L}_X \) with monotonic heads only. Then, \( P \) has exactly one \( HT_{LB} \)-stable model over \( X, \mathbb{Z} \).

Interestingly, allowing not monotonic atoms in the head may eliminate stable models but it does not produce further stable models. That is, if we drop the additional condition on heads, then we can still apply a fix point calculation and get the following result.

**Proposition 14** Let \( P \) be a stratified program over \( \mathcal{L}_X \). Then, \( P \) has at most one \( HT_{LB} \)-stable model over \( X, \mathbb{Z} \).

For instance, the program consisting of facts \( x \geq 42 \) and \( x < 42 \) only has no \( HT_{LB} \)-stable model.
4.2 Modelling Capabilities

In this section, we go into an example of error diagnosis to illustrate some modelling features of HTLB in terms of programs. In particular, the following example illustrates foundedness and default valuations.

Let $N = \{1, \ldots, n\} \subseteq \mathbb{Z}$ be an index set. We represent events by constants $e_i$ and identify them with value $i$ for $i \in N$. Consider program $P_{err}$ given by

$$\begin{align*}
error \geq \sum_{i \in X} e_i \quad &\land \quad \bigvee_{i \in X} occur(e_i) = 1 \quad \text{for all } X \subseteq N \quad (5) \\
\text{occur}(e_2) = 1 &\quad \land \quad \text{occur}(e_3) = 1, \quad error \geq 4 \quad (6) \\
\text{occur}(e_4) = 1 \quad &\iff \quad \text{temperature} \leq 42 \quad (7) \\
\text{temperature} = 60 \quad &\iff \quad \neg \text{temperature} \neq 60 \quad (8)
\end{align*}$$

Rules of (5) express that the value of $error$ is greater or equal than the sum of occurred events. The empty sum means that we have no error and is defined by 0. Rule (6) models the dependency of event $e_3$ regarding $e_2$ and the comparison if the value of $error$ is beyond some threshold value 4. If the value of $temperature$ falls below 42 degrees, then event $e_4$ occurs, modelled by Rule (7). Rule (8) sets the default value of $temperature$ to 60 degrees.

To illustrate the behaviour of $P_{err}$, let us consider the specific instance $I_{err}$ containing fact $\text{occur}(e_3) = 1$. Then, we get the single stable model $(\text{temperature} \downarrow 60) \cup (error \downarrow 3) \cup (\text{occur}(e_3) \downarrow 1)$ of $P_{err} \cup I_{err}$. The minimal founded value of $temperature$ is the default value 60. Since $e_3$ is the only event that occurs, by (5) we derive $error \geq e_3$ and thus the minimal founded value for $error$ is 3.

Let us extend $I_{err}$ to $I'_{err}$ by adding $\text{temperature} \geq 42$. Then we get stable models $(\text{temperature} \downarrow 60) \cup (error \downarrow 3) \cup (\text{occur}(e_3) \downarrow 1)$ and $(\text{temperature} \downarrow 42) \cup (error \downarrow 9) \cup (\text{occur}(e_2) \downarrow 1) \cup (\text{occur}(e_3) \downarrow 1) \cup (\text{occur}(e_4) \downarrow 1)$ of $P_{err} \cup I'_{err}$. Note that for one stable model the default valuation of $temperature$ is founded and for the other one not, due to non-monotonic atom $\text{temperature} \neq 60$ in the scope of negation. Hence, we derive $error \geq e_3$ and $error \geq e_2 + e_3 + e_4$, respectively.

5 Related Work

In this section, we compare HTLB to existing formalisms from the literature.

5.1 BFASP

First, let us compare HTLB to Aziz’ bound founded ASP (BFASP; Aziz 2015), since both share the same motivation to generalize the idea of foundedness to ordered domains.

Let us point out some differences of both approaches. In BFASP an arbitrary formula is called constraint and a rule is defined as a pair of a constraint and a variable called head. The constraint needs to be increasing wrt its head variable. A constraint is increasing in one of its variables if the constraint holds for a substitution of its variables by domain values and it holds for each substitution where the value of the particular variable is increased and rest stays the same as before. Note that the definition of increasing is made for constraints and does not differentiate between the monotonicity of atoms and logic connectives. In case of atoms Aziz’ definitions of increasing and ours of monotonic coincide. Stable models are defined in BFASP via a reduct depending on the monotonicity of constraints w.r.t their variables and by applying a fix point operation.

Both, BFASP and HTLB assign variables to their smallest domain value per default. Interestingly, they differ in their understanding of smallest domain values. In HTLB, the smallest domain value is always the value undefined to capture partiality, whereas in BFASP partiality is not considered if the value undefined is not explicitly part of a given domain.

However, the value of the head variable is derived by the constraint even if it contains no implication. For instance, let $Z_{\text{err}}$ be the variable domain of positive integers with 0 and $(x + y \geq 42, x)$ a rule in BFASP. Then, BFASP yields one stable model assigning $x$ to 42 and $y$ to 0. The value of $x$ is derived from the value of $42 - y$, obtained by the smallest value of $y$. The value of $y$ is 0, since $y$ occurs in no head and the default is the minimal domain value of $Z_{\text{err}}$. This is different from HTLB where the fact $x + y \geq 42$ results in two stable models $(x \downarrow 0) \cup (y \downarrow 42)$ and $(x \downarrow 42) \cup (y \downarrow 0)$. In HTLB, the variables of a fact are treated in an equal way instead of an implicatory way by declaring one of them as head.

Now, we show that BFASP does not satisfy the same well-known properties as HTLB. In particular, BFASP does not satisfying Proposition 10 in its turn. That is, in BFASP we may get unintuitive stable models. For instance, consider ASp rule $p \leftarrow \neg p$. This rule has no stable model in ASP and HTLB, since if $p$ holds then we cannot derive $p$ any more and if $p$ not holds then we need to derive $p$. In contrast, BFASP yields the stable model assigning $p$ to true, since the reduct will never replace head variables and produce the rule as it is.

Hence, BFASP yields the stable model assigning $p$ to true, since it is the minimal (and only) model of the rule.

5.2 HTC

Next, we compare our approach to the logic of Here-and-There with constraints (HTC; Cabalar et al. 2016).

At first, we note that both are based on HT and capture theories over (constraint) atoms in a non-monotonic setting and can easily express default values. The key difference is that HTLB inherently minimizes valuations w.r.t foundedness. This is achieved by additionally comparing valuations w.r.t the particular values assigned to the variables. To this end, we represented valuations as a set of tuples together with a downward closure regarding the assignments to yield a comparison of values in a set based mean using standard subset relation. For instance, consider the fact $x \geq 42$ over $\{x\} \subseteq \mathbb{Z}$. Then, for valuations $v$ and $v'$ with $v(x) = 42$ and $v'(x) = 43$ in HTC, we have $v \neq v'$, whereas in HTLB we have $v \subseteq v'$. Hence, both $v$ and $v'$ are stable models.

Note that (5) leads to exponentially many rules; it is also possible to write this in a more compact way using nested expressions, what we not do in this work for reasons of simplicity.

For more details see (Aziz 2015).
in \( HC \), but only the first one is \( HT_{LB} \)-stable model, due to
foundedness.

On a first look, \( HT_{LB} \) seems like \( HC \): with value mini-
ization on top. However, this is insufficient, since it does
generally not work for foundedness. Recall program \( P \) in (2)
with \( HT_{LB} \)-stable models \( (x \downarrow 1) \) and \( (x \downarrow 42) \). In contrast,
the minimal stable model in \( HC \) assigns \( x \) to 1. This elimi-
nates the second \( HT_{LB} \)-stable model. Moreover, program
\( P \) in (1) has the sole \( HT_{LB} \)-stable model \( (x \downarrow 42) \cup (y \downarrow 0) \).
Whereas in \( HC \), we get two stable models with minimal
values: one assigns \( x \) to 42 and \( y \) to 0, and the other \( x \) to 0
and \( y \) to 42. That is in general, minimization on top of stable
models in \( HC \) does not yield \( HT_{LB} \)-stable models.

However, both \( HT_{LB} \) and \( HC \) define atomic satisfaction
in terms of atom denotations. A difference is that in \( HC \)
denotations need to be closed.\(^7\) Informally, a denotation is
closed if for each valuation of the denotation every valuation
which is a superset is in the denotation as well. For \( HT_{LB} \)
this cannot be maintained, due to the additional comparison of
valuations regarding values. For instance, consider atom
\( x \neq 42 \) with \( \{ x \neq 42 \} = \{ v \mid u \neq v(x) \neq 42 \} \) over \( \{ x \}, \mathbb{Z} \).
Then, valuations \( v \) and \( v' \) with \( v(x) = 0 \) and \( v'(x) = 99 \)
are part of the denotation, but \( v'' \) with \( v''(x) = 42 \) and \( v \subset
v'' \) is not. The reason to be closed or not is that \( v, v' \) and
\( v'' \) are different in \( HC \) but subsets in \( HT_{LB} \), respectively.

The closure of denotations is significant to satisfy persistence
in \( HC \). In contrast, in \( HT_{LB} \) persistence is maintained by
forcing atomic satisfaction in both \( h \) and \( t \), instead of \( h \) only
as in \( HC \). The corresponding benefit is that this allows us
to consider atoms in \( HT_{LB} \) which are not allowed in \( HC \),
like \( x \leq y \) with \( \{ x \leq y \} \equiv \{ v \mid v(x) = v(y) \} \) which is not
closed in \( HC \) as well.

5.3 Other Formalisms

ILP Let us compare Integer Linear Programming (ILP;
Schrijver 1999) with \( HT_{LB} \).

Note that \( ILP \) is a monotone theory. Hence, compared to
\( ASP \) it is not intuitive to model recursion like reachability using
\( ILP \). For instance, in (Liu, Janhunen, and Niemelä 2012) it is mentioned that it is not easy to represent loop
formulas in \( ILP \) which are needed for this purpose.

To overcome this shortcoming, approaches like \( HT_{LB} \) and
\( HC \) tried to integrate monotone theories as \( ILP \) in a non-
monotonic setting. In other words, these approaches can be
seen as non-monotonic counterparts of \( ILP \) which support
an intuitive modelling of reachability and thus recursion, like
in \( ASP \). That is, the benefit of an intuitive modelling is a key
difference of \( HT_{LB} \) to \( ILP \).

\( ASP \) modulo Theories Now, let us compare \( HT_{LB} \) to
\( ASP \) modulo Theories approaches like in (Janhunen et al.
2017).

The idea of those approaches is to integrate monotone
theories as linear programming in the non-monotonic setting
of \( ASP \). Informally, the theories are wrapped by \( ASP \).

These approaches extend stable model semantics (Gelfond
and Lifschitz 1991) by following the approach of lazy theory
solving (Barrett et al. 2009). The idea is that a stable model is a
set of atoms which needs to be valid regarding the underly-
ing theory. Technically, in (Janhunen et al. 2017) a program
over a theory is extended by rules depending on possible as-
signments wrt the theory to determine the stable models. The
assignments for variables are obtained by particular theory
solvers if the atoms are valid in the theory. It is interesting to
note that there are two ways of interpreting atoms which do
not occur in a model: one way is to assume that the opposite
needs to hold and the other way is to let it open.

Similar to \( HC \), the main difference of \( ASP \) modulo Theory
approaches to \( HT_{LB} \) is that atoms are founded but per
definition foundedness regarding values is not achieved for
its comprised variables, since stable models in \( ASP \) modulo
Theory rely on any possible valid assignment for variables.

Aggregates Aggregates are extensions of \( ASP \) allowing us
to perform set operations like counting and summing on
elements of a respective set. Aggregates can be treated by
translating them into \( ASP \) rules. For instance, sum aggre-
gates can be translated by adapting well-known techniques
translating pseudo-Boolean constraints into SAT, cf (Sinz
2005) and (Bomanson and Janhunen 2013).

The syntax of an aggregate is given by \( \{ c_1 : \varphi_1 , \ldots , c_m : \varphi_m \} \prec k \), where \( i \) is an aggregate symbol, \( c_i \), \( k \) constants,
\( \varphi \) propositional formulas also called conditions with \( 1 \leq
i \leq m \), and \( \prec \in \{ \leq , < , > , \geq , = , \neq \} \) a binary relation.

On semantics side, the community comes up with different
understandings for aggregates like in (Ferraris 2011;
Gelfond and Zhang 2014; Son and Pontelli 2007). Informally,
a constant belongs to the set if its condition holds. The aggre-
gate holds if the relation holds for all constants that belong
to the set.

Obviously, (sum) aggregates are related to (linear con-
straint) atoms of \( HT_{LB} \). As we will show in an extended
version of this work, aggregates under Ferraris’ semantics
(Ferraris 2011) can be represented by atoms in \( HT_{LB} \). To
this end, we restrict conditions of aggregates to propositional
atoms. Note that this is not a very limiting restriction, since
these atoms can be seen as auxiliaries for arbitrary formulas.

This is interesting, since it means that aggregates are no
longer an extension of an existing approach, instead aggreg-
gates under Ferraris’ semantics are now already integrated as
atoms of an approach. Hence, the results shown in this work
allow us to view aggregates in a new setting and give us a
possibly better way to elaborate on their properties like mono-
tonicity. Maybe the view on aggregates as atoms in context
of \( HT_{LB} \) helps us to better understand the existing discussion
of different aggregate semantics and their properties.

6 Conclusion

We presented the idea of foundedness for minimal values
of variables over ordered domains in the setting of the logic
of Here-and-There. We elaborated on important properties
like persistence, negation and strong equivalence and showed
that they hold in our approach. Furthermore, we pointed
out that the base logic \( HC \) can be seen as a special case of
\( HT_{LB} \). To prove if our approach follows well-known patterns,
we showed that $HT_{LB}$-stable models correspond to stable models according to a Ferraris’-like stable model semantics.

To elaborate on our approach in terms of logic programming and modelling, we isolated a fragment dealing with linear constraints over integers. In this context, we analyzed the influence of monotonicity of atoms on programs and concepts like normal, stratified and positive. Moreover, we illustrated the features of foundedness and defaults with the example of error diagnosis.

Finally, we compared our approach to related ones and showed that foundedness is a non-trivial key feature of $HT_{LB}$. We showed that $HT_{LB}$ and $BFASP$ have the same starting motivation but differ in their treatments of undefined and monotonicity. Furthermore, we pointed out that $HT_{LB}$ can be seen as non-monotonic counterpart of monotonic theories. We also mentioned that $HT_{LB}$ offers a new view of aggregates under Ferraris’ semantics as atoms with its corresponding monotonic properties. Thus, aggregates are integrated in $HT_{LB}$ instead of being an extension of an existing approach.

In an extended version we plan to present a fix point operator, dependency graph, (odd and even) loops, stratification, splitting sets, and the relation to aggregates in detail.

References


Appendix of Proofs

Proof of Proposition 1 To prove that $(H, T) \models \varphi$ holds under Definition 1 iff it holds when replacing Condition 2 by 2’ for $(H, T)$ $HT$-interpretation over $A$ and $\varphi$ a propositional formula over $A$, it is enough to prove equivalence of base cases 2 and 2’, since the rest follows directly by structural induction. Per definition of denotation we have for propositional atom $p \in A$ that

$$H \in \llbracket p \rrbracket_A \iff H \in \{I \mid p \in I\} \iff p \in H$$
Proof of Proposition 2  It is enough to prove the proposition for the base case, since the rest follows directly by structural induction for each formula over \( \mathcal{A}_X \). Let \( \langle h, t \rangle \) an \( HT_{LB} \)-valuation over \( \mathcal{X}, \mathcal{D} \) and \( a \) an atom of \( \mathcal{A}_X \).

First, we prove persistence, represented by \( \ell \) of the proposition. We have
\[
\langle h, t \rangle \models a \iff h \in \left[ a \right] \land t \in \left[ a \right] \Rightarrow t \models \left( \ell, t \right) \models a
\]

Subsequently, we prove negation, represented by \( \ell \) of the proposition. We have
\[
\langle h, t \rangle \models \neg a \Rightarrow a \rightarrow \bot \\
\langle h, t \rangle \models a \land \langle t, t \rangle \models \neg a \\
\langle h, t \rangle \models a \land t \not\models a \\
\langle h, t \rangle \not\models a
\]

Proof of Proposition 3  For any \( a \in \text{At}(\langle h, t \rangle) = \{ a \in \mathcal{A}_X \mid h \in [a] \text{ and } t \in [a] \} \) we have \( h \in \left[ a \right] \text{ and } t \in \left[ a \right] \), thus we conclude \( a \in \text{At}(\langle t, t \rangle) = \{ a \in \mathcal{A}_X \mid t \in \left[ a \right] \} \).

Proof of Proposition 4  Let \( \Gamma_1, \Gamma_2 \) and \( \Gamma \) be theories over \( \mathcal{A}_X \). First, we prove “\( \Rightarrow \)” of the proposition. For each \( HT_{LB} \)-valuation \( \langle h, t \rangle \) over \( \mathcal{X}, \mathcal{D} \) we have \( \langle h, t \rangle \models \Gamma \iff \Gamma_1 \cup \Gamma_2 \models \Gamma \). This implies that \( \langle h, t \rangle \models \Gamma_1 \cup \Gamma_2 \iff \Gamma_2 \cup \Gamma \models \Gamma_1 \cup \Gamma_2 \) for any \( \Gamma \). Hence, \( \Gamma_1 \cup \Gamma \) and \( \Gamma_2 \cup \Gamma_2 \) have the same \( HT_{LB} \)-stable models for every \( \Gamma \).

Secondly, we prove “\( \Leftarrow \)” by contradiction. Without loss of generality, assume that \( \langle h, t \rangle \) is \( HT_{LB} \)-valuation over \( \mathcal{X}, \mathcal{D} \) with \( \langle h, t \rangle \models \Gamma_1 \) and \( \langle h, t \rangle \not\models \Gamma_2 \). Then, we differ two cases:

Case 1: Let \( \langle t, t \rangle \not\models \Gamma_2 \). We have \( \langle h, t \rangle \models \Gamma_1 \) and thus by persistence (Proposition 2) \( \langle t, t \rangle \models \Gamma_1 \). Let \( \Gamma = \{ x \geq c \mid (x, c) \in t \} \). Then, \( \langle t, t \rangle \models \Gamma_1 \cup \Gamma \) is \( HT_{LB} \)-stable model. But \( \langle t, t \rangle \not\models \Gamma_2 \cup \Gamma \) by assumption.

Case 2: Let \( \langle t, t \rangle \models \Gamma_2 \). Moreover, let \( \Gamma = \Gamma_1 \cup \Gamma_2 \) with \( \Gamma_1 = \{ x \geq c \mid (x, c) \in h \} \) and \( \Gamma_2 = \{ x \geq t(x) \rightarrow y \geq t(y), x \geq c \rightarrow x \geq t(x), (x, c), (x, t(x)), (y, t(y)) \in t \land h \} \). Then, \( \langle t, t \rangle \models \Gamma \) by assumption and \( h \subseteq t \). Note there is no \( v \subseteq t \) with \( \langle v, v \rangle \models \Gamma \) since by \( \langle h, t \rangle \not\models \Gamma_2 \) we get that \( h \subset v \not\subseteq t \) need to hold, and thus there exists at least one pair \( a_1, a_2 \in \text{atoms}(\Gamma) \) with \( v \not\subseteq \left[ a_1 \right] \) and \( v \not\subseteq \left[ a_2 \right] \). Hence, \( \langle v, v \rangle \not\models \Gamma_2 \cup \Gamma_1 \) for \( h \subseteq v \subset t \). Thus, \( \langle t, t \rangle \) is \( HT_{LB} \)-stable model of \( \Gamma_2 \cup \Gamma_1 \). By assumption and construction, we have that \( \langle t, t \rangle \models \Gamma_1 \) and \( \langle t, t \rangle \models \Gamma_1' \). Moreover, we have that \( \langle t, t \rangle \not\models a \) for every \( a \in \text{atoms}(\Gamma_1) \). Hence, \( \langle t, t \rangle \models \Gamma' \) and thus \( \langle h, t \rangle \models \Gamma_1 \cup \Gamma_2 \). Note that since \( \langle h, t \rangle \not\models \Gamma_2 \) and \( \langle t, t \rangle \models \Gamma_2 \), we have \( \langle h, t \rangle \not\models (\ell, t) \), which implies that \( h \subseteq t \). Finally, \( \langle t, t \rangle \) is no \( HT_{LB} \)-stable model of \( \Gamma_1 \cup \Gamma_2 \).

Proof of Proposition 5  Let \( a \) be an atom over \( \mathcal{A}_X \), and \( \pi \) and \( \pi' \) its complement and its strict complement over \( \mathcal{A}_X \), respectively.

First, we prove \( \pi' \models \pi \). For any \( HT_{LB} \)-valuation \( \langle h, t \rangle \) over \( \mathcal{X}, \mathcal{D} \) we have
\[
\langle h, t \rangle \models \pi' \iff h \in \left[ \pi' \right] \land t \in \left[ \pi' \right] \text{ with } \left[ \pi' \right] = 2^{\forall}(\left[ a \right] \cup \{ v \mid v(x) = a \text{ for some } x \in \text{vars}(a) \}) \\
\Rightarrow h \in 2^{\forall}(\left[ a \right] \land t \in 2^{\forall}(\left[ a \right]) \\
\Rightarrow \langle h, t \rangle \models \pi
\]

Secondly, we prove \( \pi \models \neg a \). For any \( HT_{LB} \)-valuation \( \langle h, t \rangle \) over \( \mathcal{X}, \mathcal{D} \) we have
\[
\langle h, t \rangle \models \pi \\
\Rightarrow h \in \left[ \pi \right] \land t \in \left[ \pi \right] \text{ with } \left[ \pi \right] = 2^{\forall}(\left[ a \right]) \\
\Rightarrow \langle h, t \rangle \models \neg a
\]

Proof of Proposition 6  It is enough to prove the proposition for the base case, since the rest follows directly by structural induction for each theory over \( \mathcal{A} \).

Let \( \Gamma \) be a theory over propositional atoms \( \mathcal{A} \) and \( \langle H, T \rangle \) an \( HT \)-interpretation over \( \mathcal{A} \). Let \( \tau(\Gamma) \) be a theory over atoms\( \{ p = t \mid p \in \mathcal{A} \} \) and \( (\tau(H), \tau(T)) \) an \( HT_{LB} \)-interpretation over \( \mathcal{A}, \{ t \} \). Then we have
\[
\langle H, T \rangle \models H \\
\Rightarrow H \in \text{assertions}(\mathcal{A}) \\
\Rightarrow (\tau(H), \tau(T)) \models p = t
\]

Proof of Proposition 7  It is enough to prove the proposition for the base case, since the rest follows directly by structural induction for each theory over \( \mathcal{A}_X \).

First, note that the pair \( (H, T) \) over \( \mathcal{A}_X \) with \( H = \text{At}((h, t)) \) and \( T = \text{At}((t, t)) \) is a well formed \( HT \)-interpretation, since \( H \subseteq T \) holds by \( h \subseteq t \) and Proposition 3. Then we have
\[
\langle h, t \rangle \models a \\
\Rightarrow h \in \left[ a \right]_{\mathcal{X}, \mathcal{D}} \land t \in \left[ a \right]_{\mathcal{X}, \mathcal{D}} \\
\Rightarrow H \models \left[ a \right]_{\mathcal{A}_X} \land T \models \left[ a \right]_{\mathcal{A}_X} \\
\Rightarrow \langle H, T \rangle \models a
\]

Proof of Proposition 8  Let \( \varphi \) over \( \mathcal{A} \) be an arbitrary tautology in \( HT \). This means that for every \( HT \)-interpretation \( \langle H, T \rangle \) over \( \mathcal{A} \) holds \( \langle H, T \rangle \models \varphi \). Thus, we conclude that formula \( \varphi^\prime \) over \( \mathcal{A}_X \) obtained by replacing \( \text{atoms}(\varphi) \) in \( \varphi \) by atoms of \( \mathcal{A}_X \), is a tautology as well (for every \( HT_{LB} \)-valuation \( (h, t) \) over \( \mathcal{X}, \mathcal{D} \) holds \( \langle h, t \rangle \models \varphi^\prime \)), since the semantics of the atoms may change the truth value of a single atom but can not affect the truth of the formula itself.

Proof of Proposition 9  It is enough to prove the proposition for the base case, since the rest follows directly by structural induction for each theory over \( \mathcal{A}_X \).

Let \( \Gamma \) be a theory over \( \mathcal{A}_X \) and \( \langle h, t \rangle \) an \( HT_{LB} \)-valuation over \( \mathcal{X}, \mathcal{D} \). Then, we have
\[
\langle h, t \rangle \models a \\
\Rightarrow h \models a \\
\Rightarrow h \in \left[ a \right] \land t \in \left[ a \right]
\]

\[
\langle h, t \rangle \models a
\]