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

$$x \geq 0 \qquad y \geq 0 \qquad x \geq 42 \leftarrow y < 42 \quad (1)$$

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

$$x \geq 1 \qquad x \geq 42 \leftarrow \neg(x \leq 1) \quad (2)$$

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

$p \leftarrow \neg p$. Then, Aziz' approach yields a solution where p holds instead of no solution as expected in terms of ASP. To this end, we present in the following a logical reconstruction of Aziz' idea of foundedness 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, short HT_{LB}, along with its equilibrium models. We elaborate upon the formal properties of HT_{LB} regarding persistence, negation and strong equivalence. Furthermore, we point out the relation of HT_{LB} to HT, and show that our approach corresponds to a straightforward extension of Ferraris' stable model semantics (Ferraris 2005). 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, to point out the benefits of HT_{LB} and sketch future work.

2 Background

Let $\mathcal{A}$ be the set of propositional atoms. A formula φ is a combination of atoms by logical connectives $\bot$, $\wedge$, $\vee$, and $\leftarrow$. As usual, we define $\top \stackrel{\text{def}}{=} \bot \rightarrow \bot$ and $\neg\varphi \stackrel{\text{def}}{=} \varphi \rightarrow \bot$. 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 $\langle H, T\rangle$ 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 $[\![\cdot]\!]_{\mathcal{A}} : \mathcal{A} \rightarrow 2^{\mathcal{A}}$ mapping atoms in $\mathcal{A}$ to sets of interpretations over $\mathcal{A}$. Accordingly, $[\![p]\!]_{\mathcal{A}} \stackrel{\text{def}}{=} \{I \mid p \in I\}$ represents the sets of interpretations where atom p holds.

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

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

1. $\langle H, T\rangle \not\models \bot$
2. $\langle H, T\rangle \models p$ *iff* $H \in [\![p]\!]_{\mathcal{A}}$ *for propositional atom* $p \in \mathcal{A}$
3. $\langle H, T\rangle \models \varphi_1 \wedge \varphi_2$ *iff* $\langle H, T\rangle \models \varphi_1$ *and* $\langle H, T\rangle \models \varphi_2$
4. $\langle H, T\rangle \models \varphi_1 \vee \varphi_2$ *iff* $\langle H, T\rangle \models \varphi_1$ *or* $\langle H, T\rangle \models \varphi_2$
5. $\langle H, T\rangle \models \varphi_1 \rightarrow \varphi_2$ *iff* $\langle I, T\rangle \not\models \varphi_1$ *or* $\langle I, T\rangle \models \varphi_2$ *for both* $I \in \{H, T\}$

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

2': $\langle H, T\rangle \models p$ *iff* $p \in H$ *for propositional atom* $p \in \mathcal{A}$

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

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

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

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 $\mathcal{A}_{\mathcal{X}}$ comprising variables, $\mathcal{X}$, and constants over an ordered domain $(\mathcal{D}, \succeq)$. For simplicity, we assume that each element of $\mathcal{D}$ is uniquely represented by a constant and abuse notation by using $\mathcal{D}$ to refer to the set of constants. Similarly, we identify $\succeq$ with its syntactic representative. The specific syntax of atoms is left open but assumed to refer to elements of $\mathcal{X}$ and $\mathcal{D}$. The only requirement is that we assume that an atom depends on a distinguished subset of variables of $\mathcal{X}$. An atoms 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 $(\mathbb{Z}, \geq)$, respectively; corresponding atoms are $x \geq 42$ and $x = y$. A formula φ is a propositional combination of atoms and logical connectives $\bot, \wedge, \vee, \rightarrow$. As usual, we define $\top \stackrel{\text{def}}{=} \bot \rightarrow \bot$ and $\neg\varphi \stackrel{\text{def}}{=} \varphi \rightarrow \bot$. A theory is a set of formulas. For instance, '$y < 42 \wedge \neg(x = y) \rightarrow x \geq 42$' is a formula. Let $vars(\varphi) \subseteq \mathcal{X}$ be the set of variables and $atoms(\varphi) \subseteq \mathcal{A}_{\mathcal{X}}$ the atoms occurring in a formula φ.

For capturing partiality, we introduce a special domain element $\boldsymbol{u}$, standing for *undefined*, and extend $(\mathcal{D}, \succeq)$ to $(\mathcal{D}_{\boldsymbol{u}}, \succeq_{\boldsymbol{u}})$ where $\mathcal{D}_{\boldsymbol{u}} \stackrel{\text{def}}{=} \mathcal{D} \cup \{\boldsymbol{u}\}$ and $\succeq_{\boldsymbol{u}} \stackrel{\text{def}}{=} \succeq \cup \{(c, \boldsymbol{u}) \mid c \in \mathcal{D}\}$. With it, we define a (partial) *valuation* over $\mathcal{X}, \mathcal{D}$ as a function $v : \mathcal{X} \rightarrow \mathcal{D}_{\boldsymbol{u}}$ mapping each variable to a domain value or undefined. For comparing valuations by set-based means, we alternatively represent them by subsets of $\mathcal{X} \times \mathcal{D}$. Basically, any function v is a set of pairs (x, c) such that $v(x) = c$ for $c \in \mathcal{D}$. In addition, we view a pair (x, c) as $x \succeq c$ and add its downward closure $(x{\downarrow}c) \stackrel{\text{def}}{=} \{(x, d) \mid c, d \in \mathcal{D}, c \succeq d\}$. Given this, a valuation v is represented by the set $\bigcup_{v(x)=c, x\in\mathcal{X}}(x{\downarrow}c)$.[1] As an example, consider variables x and y over domain $(\{0, 1, 2, 3\} \cup \{\boldsymbol{u}\}, \geq_{\boldsymbol{u}})$. The valuation $v = \{x \mapsto 2, y \mapsto 0\}$ can be represented by $v = (x{\downarrow}2) \cup (y{\downarrow}0) = \{(x, 0), (x, 1), (x, 2), (y, 0)\}$. Then, $v' = \{x \mapsto 1, y \mapsto \boldsymbol{u}\}$, 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 $\succeq$. We let $\mathfrak{V}_{\mathcal{X},\mathcal{D}}$ stand for the set of valuations over $\mathcal{X}$ and $\mathcal{D}$.

We define the satisfaction of formulas over $\mathcal{A}_{\mathcal{X}}$ wrt *atom denotations* over $\mathcal{X}, \mathcal{D}$, which are functions $[\![\cdot]\!]_{\mathcal{X},\mathcal{D}} : \mathcal{A}_{\mathcal{X}} \rightarrow$

[1] Note that $(x{\downarrow}\boldsymbol{u}) = \emptyset$, since $\boldsymbol{u} \notin \mathcal{D}$.

$2^{\mathfrak{V}_{\mathcal{X},\mathcal{D}}}$ mapping atoms to sets of valuations. Let a be an atom of $\mathcal{A}_\mathcal{X}$ and $[\![a]\!]_{\mathcal{X},\mathcal{D}}$ its denotation. Then, $[\![a]\!]_{\mathcal{X},\mathcal{D}}$ is the set of valuations v so that a holds. Since a depends on variables $vars(a) \subseteq \mathcal{X}$, we have for each $v \in [\![a]\!]$ and valuation v' with $v(x) = v'(x)$ for $x \in vars(a)$ that $v' \in [\![a]\!]$. Intuitively, values of $\mathcal{X} \setminus vars(a)$ can vary freely without changing the membership of a valuation to $[\![a]\!]$. For simplicity, we drop indices $\mathcal{X}, \mathcal{D}$ whenever clear from context.

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

$$[\![x \geq 42]\!] \stackrel{\text{def}}{=} \{v \mid v(x) \geq 42\}$$
$$[\![42 \geq 0]\!] \stackrel{\text{def}}{=} \mathfrak{V}$$
$$[\![0 \geq 42]\!] \stackrel{\text{def}}{=} \emptyset.$$

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

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

Definition 2 *Let $\langle h, t \rangle$ be an HT_{LB}-valuation over $\mathcal{X}, \mathcal{D}$ and φ be a formula over $\mathcal{A}_\mathcal{X}$. Then, $\langle h, t \rangle$ satisfies φ, written $\langle h, t \rangle \models \varphi$, if the following holds:*

1. $\langle h, t \rangle \not\models \bot$
2. $\langle h, t \rangle \models a$ *iff* $v \in [\![a]\!]_{\mathcal{X},\mathcal{D}}$ *for atom* $a \in \mathcal{A}_\mathcal{X}$ *and for both* $v \in \{h, t\}$
3. $\langle h, t \rangle \models \varphi_1 \wedge \varphi_2$ *iff* $\langle h, t \rangle \models \varphi_1$ *and* $\langle h, t \rangle \models \varphi_2$
4. $\langle h, t \rangle \models \varphi_1 \vee \varphi_2$ *iff* $\langle h, t \rangle \models \varphi_1$ *or* $\langle h, t \rangle \models \varphi_2$
5. $\langle h, t \rangle \models \varphi_1 \rightarrow \varphi_2$ *iff* $\langle v, t \rangle \not\models \varphi_1$ *or* $\langle v, t \rangle \models \varphi_2$ *for both* $v \in \{h, t\}$

As usual, we call $\langle h, t \rangle$ an HT_{LB}-model of a theory Γ, if $\langle h, t \rangle \models \varphi$ for all φ in Γ. For a simple example, consider the theory containing atom $x \geq 42$ only. Then, every HT_{LB}-valuation $\langle h, t \rangle$ with $h, t \in [\![x \geq 42]\!]$ 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 $\langle h, t \rangle$ and $\langle t, t \rangle$ be HT_{LB}-valuations over $\mathcal{X}, \mathcal{D}$, and φ be a formula over $\mathcal{A}_\mathcal{X}$. Then,*

1. $\langle h, t \rangle \models \varphi$ *implies* $\langle t, t \rangle \models \varphi$, *and*
2. $\langle h, t \rangle \models \varphi \rightarrow \bot$ *iff* $\langle t, t \rangle \not\models \varphi$.

Persistence implies that all atoms satisfied by $\langle h, t \rangle$ are also satisfied by $\langle t, t \rangle$. To make this precise, let $At(\langle h, t \rangle) \stackrel{\text{def}}{=} \{a \in \mathcal{A}_\mathcal{X} \mid h \in [\![a]\!] \text{ and } t \in [\![a]\!]\}$ be the set of atoms satisfied by $\langle h, t \rangle$.

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

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

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

We refer an HT_{LB}-equilibrium model $\langle t, t \rangle$ of Γ as an HT_{LB}-stable model t of Γ. 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 $t \in [\![x \geq 42]\!]$ and there is no $h \subset t$ with $h \in [\![x \geq 42]\!]$. In contrast, neither HT_{LB}-model $\langle t', t' \rangle$ with $t' = (x \downarrow 42) \cup (y \downarrow 0)$ nor $\langle t'', t'' \rangle$ with $t'' = (x \downarrow 53)$ are HT_{LB}-stable models since t is a proper subset of both and $\langle t, t' \rangle \models x \geq 42$ as well as $\langle t, t'' \rangle \models x \geq 42$ 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 Γ_1, Γ_2 and Γ be theories over $\mathcal{A}_\mathcal{X}$. Then, theories $\Gamma_1 \cup \Gamma$ and $\Gamma_2 \cup \Gamma$ have the same HT_{LB}-stable models for every theory Γ iff Γ_1 and Γ_2 have the same HT_{LB}-models.*

The idea is to prove the if direction by proving its contraposition, and the only if direction by proving its straightforward implication. The contraposition assumes that there exists an HT_{LB}-valuation that satisfies Γ_1 but not Γ_2 which implies that the stable models of $\Gamma_1 \cup \Gamma$ and $\Gamma_2 \cup \Gamma$ do not coincide. There are two cases to construct Γ in a way that $\Gamma_1 \cup \Gamma$ has a stable model which is not a stable model of $\Gamma_2 \cup \Gamma$ and the other way around, respectively. Let us consider an example to illustrate the idea of the construction of Γ. Let $h = (x \downarrow 0)$ and $t = (x \downarrow 2) \cup (y \downarrow 0)$ be HT_{LB}-valuation over $\{x, y\}, \{0, 1, 2, 3\}$ with $\langle h, t \rangle \models \Gamma_1$ and $\langle h, t \rangle \not\models \Gamma_2$. For the first case assume that $\langle t, t \rangle \not\models \Gamma_2$. Since t cannot be a model of $\Gamma_2 \cup \Gamma$ by assumption, we construct Γ in a way that t is a stable model of $\Gamma_1 \cup \Gamma$. Hence, let $\Gamma = \{z \succeq c \mid (z, c) \in t\} = \{x \succeq 0, x \succeq 1, x \succeq 2, y \succeq 0\}$ be the theory with the only stable model t. By persistence of $\langle h, t \rangle$ wrt Γ_1 and construction of Γ we get that t is a stable model of $\Gamma_1 \cup \Gamma$ but not of $\Gamma_2 \cup \Gamma$. For the second case we assume that $\langle t, t \rangle \models \Gamma_2$. Now we construct Γ in a way that t is a stable model of $\Gamma_2 \cup \Gamma$ but not of $\Gamma_1 \cup \Gamma$. By assumption we have that $\langle h, t \rangle \models \Gamma_1$ and $\langle h, t \rangle \not\models \Gamma_2$ as well as $\langle t, t \rangle \models \Gamma_2$, thus we want to have $\langle h, t \rangle$ and $\langle v, v' \rangle$ with $t \subseteq v \subseteq v'$ as the only models of Γ. Hence, let $\Gamma = \Gamma' \cup \Gamma''$ with $\Gamma' = \{z \succeq c \mid (z, c) \in h\} = \{x \succeq 0\}$ the theory that is satisfied by everything that is greater or equal than h, and $\Gamma'' = \{z \succeq t(z) \rightarrow z' \succeq t(z'),\ z \succeq c \rightarrow z \succeq t(z) \mid (z, c), (z, t(z)), (z', t(z')) \in t \setminus h, z \neq z'\} = \{x \succeq 2 \rightarrow y \succeq 0,\ y \succeq 0 \rightarrow x \succeq 2,\ x \succeq 1 \rightarrow x \succeq 2,\ x \succeq 2 \rightarrow x \succeq 2\}$ the theory which derives values of t for each v'' with $h \subset v'' \subset t$. Since $\langle h, t \rangle \not\models \Gamma_2$ and by construction of Γ we get that t is a stable model of $\Gamma_2 \cup \Gamma$ but not of $\Gamma_1 \cup \Gamma$.

3.2 Negation in HT_{LB}

In the following, we elaborate on complements of atoms and its relation to negation, since $\mathcal{A}_\mathcal{X}$ 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 $\overline{a}$ of atom a by its denotation $[\![\, \overline{a} \,]\!] \stackrel{\text{def}}{=} 2^\mathfrak{V} \setminus [\![\, a \,]\!]$.

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 $(\{\mathbf{t}, \mathbf{f}\}, \{\mathbf{t} \succeq \mathbf{f}\})$. Then, the denotation of propositional atoms in HT_{LB} can be defined as follows: $[\![\, p = \mathbf{t} \,]\!]_{\mathcal{A},\{\mathbf{t},\mathbf{f}\}} \stackrel{\text{def}}{=} \{v \mid v(p) = \mathbf{t}\}$ and $[\![\, p = \mathbf{f} \,]\!]_{\mathcal{A},\{\mathbf{t},\mathbf{f}\}} \stackrel{\text{def}}{=} \{v \mid v(p) = \mathbf{f}\}$. Note that $p = \mathbf{t}$ and $p = \mathbf{f}$ are regarded as strong negations of each other, as in standard case (Gelfond and Lifschitz 1990); its weak negations are given by $\neg(p = \mathbf{t})$ and $\neg(p = \mathbf{f})$, respectively. For instance, the complement $\overline{p = \mathbf{t}}$ is characterized by denotation $[\![\, \overline{p = \mathbf{t}} \,]\!] = 2^\mathfrak{V} \setminus [\![\, p = \mathbf{t} \,]\!] = \{v \mid v(p) \neq \mathbf{t}\}$. Note that this complement allows valuations v with $v(p) = \boldsymbol{u}$, which does not match $p = \mathbf{f}$.

To this end, we define another complement to exclude assigning value 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 $\overline{a}^s$ of atom a by the strict denotation $[\![\, \overline{a}^s \,]\!] \stackrel{\text{def}}{=} 2^\mathfrak{V} \setminus ([\![\, a \,]\!] \cup \{v \mid v(x) = \boldsymbol{u} \text{ for some } x \in vars(a)\})$. Informally, the strict complement of an atom holds whenever all variables are defined and the atom itself does not hold. That is, atoms $p = \mathbf{f}$ and $p = \mathbf{t}$ 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 complement $\overline{x \geq 42}^s$ is defined by $[\![\, \overline{x \geq 42}^s \,]\!] = \{v \mid \boldsymbol{u} \neq v(x) < 42\}$. As in the Boolean case, the strict complement $\overline{x \geq 42}^s$ 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) Γ over $\mathcal{A}_\mathcal{X}$ entails a formula φ over $\mathcal{A}_\mathcal{X}$, written $\Gamma \models \varphi$, when all HT_{LB}-models of Γ are HT_{LB}-models of φ. Then, we have the following result.

Proposition 5 *Let a be an atom over $\mathcal{A}_\mathcal{X}$, and $\overline{a}$ and $\overline{a}^s$ its complement and its strict complement over $\mathcal{A}_\mathcal{X}$, respectively. Then, $\overline{a}^s \models \overline{a}$ and $\overline{a} \models \neg a$.*

This implies that the strict complement $\overline{a}^s$ of an atom a implies its negation $\neg a$, 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 ($\neg a \not\models \overline{a}$), let us consider atom $x \leq 42$ with strict denotation $[\![\, x \leq 42 \,]\!] = \{v \mid \boldsymbol{u} \neq v(x) \leq 42\}$. Then, the complement $\overline{x \leq 42}$ is defined by denotation $[\![\, \overline{x \leq 42} \,]\!] = 2^\mathfrak{V} \setminus [\![\, x \leq 42 \,]\!] = \{v \mid v(x) = \boldsymbol{u} \text{ or } v(x) > 42\}$. For valuations $h = (x \downarrow 42)$ and $t = (x \downarrow 50)$ we have that $\langle h, t\rangle \models \neg(x \leq 42)$ since $(x{\downarrow}50) \notin [\![\, x \leq 42 \,]\!]$. In contrast, $\langle h, t\rangle \models \overline{x \leq 42}$ does not hold, since $(x{\downarrow}42) \notin [\![\, \overline{x \leq 42} \,]\!]$. Thus, the complement $\overline{a}$ 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 $[\![\, p \,]\!]_\mathcal{A}$ and $[\![\, p = \mathbf{t} \,]\!]_{\mathcal{A},\{\mathbf{t}\}}$ of a propositional atom p collect interpretations and valuations assigning true to p, respectively. To this end, we define a transformation τ relating each propositional atom p with corresponding atom $p = \mathbf{t}$ by $\tau(p) \stackrel{\text{def}}{=} p = \mathbf{t}$. Let Γ be a propositional theory, then $\tau(\Gamma)$ is obtained by substituting each $p \in atoms(\Gamma)$ by $\tau(p)$. Moreover, we extend τ to interpretations I by $\tau(I) \stackrel{\text{def}}{=} \{(p, \mathbf{t}) \mid p \in I\}$ to obtain a corresponding valuation over $\mathcal{A}, \{\mathbf{t}\}$. The next proposition establishes that HT can be seen as a special case of HT_{LB}.

Proposition 6 *Let Γ 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 $\{p = \mathbf{t} \mid p \in \mathcal{A}\}$ and $\langle \tau(H), \tau(T)\rangle$ an HT_{LB}-valuation over $\mathcal{A}, \{\mathbf{t}\}$. Then, $\langle H, T\rangle \models \Gamma$ iff $\langle \tau(H), \tau(T)\rangle \models \tau(\Gamma)$.*

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 Γ be a theory over $\mathcal{A}_\mathcal{X}$ and $\langle h, t\rangle$ an HT_{LB}-model of Γ over $\mathcal{X}, \mathcal{D}$. Then, $\langle At(\langle h, t\rangle), At(\langle t, t\rangle)\rangle$ is an HT-model of Γ over $\mathcal{A}_\mathcal{X}$.*

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

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

Proposition 8 *Let φ be a tautology over $\mathcal{A}$ and φ' a formula over $\mathcal{A}_\mathcal{X}$ obtained by replacing all atoms in φ by atoms of $\mathcal{A}_\mathcal{X}$. Then, φ' 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 $\langle h, t\rangle$ satisfy atom a in $\mathcal{A}_\mathcal{X}$, and by persistence HT_{LB}-valuation $\langle t, t\rangle$ satisfies a as well, but not necessarily each HT_{LB}-valuation $\langle v, t\rangle$ with $h \subset v \subset t$ satisfies a. For instance, consider atom $x \neq 42$ with $[\![\, x \neq 42 \,]\!] \stackrel{\text{def}}{=} \{v \mid \boldsymbol{u} \neq v(x) \neq 42\}$. Let $h = (x \downarrow 0)$ and $t = (x{\downarrow}53)$ be valuations. Then, $\langle h, t\rangle \models x \neq 42$ and $\langle t, t\rangle \models x \neq 42$, but for $v = (x \downarrow 42)$ with $h \subset v \subset t$ we have $\langle v, t\rangle \not\models x \neq 42$.

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 φ a formula over $\mathcal{A}_\mathcal{X}$. Then, t satisfies φ, written $t \models_{cl} \varphi$, if the following holds:*

1. $t \not\models_{cl} \bot$
2. $t \models_{cl} a$ *iff* $t \in [\![a]\!]_{\mathcal{X},\mathcal{D}}$ *for atom* $a \in \mathcal{A}_\mathcal{X}$
3. $t \models_{cl} \varphi_1 \wedge \varphi_2$ *iff* $t \models_{cl} \varphi_1$ *and* $t \models_{cl} \varphi_2$
4. $t \models_{cl} \varphi_1 \vee \varphi_2$ *iff* $t \models_{cl} \varphi_1$ *or* $t \models_{cl} \varphi_2$
5. $t \models_{cl} \varphi_1 \rightarrow \varphi_2$ *iff* $t \not\models_{cl} \varphi_1$ *or* $t \models_{cl} \varphi_2$.

We call t a classical model of a theory Γ, if $t \models_{cl} \varphi$ for all φ in Γ. We define a Ferraris-like reduct, short F-reduct, wrt atoms $\mathcal{A}_\mathcal{X}$ as follows.

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

$$\varphi^t \stackrel{\text{def}}{=} \begin{cases} \bot & \text{if } t \not\models_{cl} \varphi \\ a & \text{if } t \models_{cl} \varphi \text{ and } \varphi = a \text{ atom of } \mathcal{A}_\mathcal{X} \\ {\varphi_1}^t \otimes {\varphi_2}^t & \text{if } t \models_{cl} \varphi \text{ and } \varphi = (\varphi_1 \otimes \varphi_2) \\ & \text{for } \otimes \in \{\wedge, \vee, \rightarrow\} \end{cases}$$

For theory Γ and HT_{LB}-valuation t, we define $\Gamma^t \stackrel{\text{def}}{=} \{\varphi^t \mid \varphi \in \Gamma\}$. 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 Γ over $\mathcal{A}_\mathcal{X}$ iff $t \models_{cl} \Gamma^t$ and there is no $h \subset t$ such that $h \models_{cl} \Gamma^t$.*

The next propositions shows 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 Γ a theory over $\mathcal{A}_\mathcal{X}$. 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 Γ a theory over $\mathcal{A}_\mathcal{X}$. Then, t is an HT_{LB}-stable model of Γ iff t is an F-stable model of Γ.*

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

$$\sum_{i=1}^{m} w_i x_i \prec k$$

where $w_i, k \in \mathbb{Z}$ are constants, $x_i \in \mathcal{X}$ are distinct variables, and $\prec \in \{\geq, \leq, \neq, =\}$[2] is a binary relation. By $\mathcal{L}_\mathcal{X}$ we denote the set of linear constraint atoms wrt $\mathcal{X}$ and $\mathbb{Z}$. The denotation of a linear constraint atom is given by

$$[\![\sum_{i=1}^{m} w_i x_i \prec k]\!] \stackrel{\text{def}}{=} \{v \mid \sum_{i=1}^{m} w_i v(x_i) \prec k,\ v(x_i) \neq \boldsymbol{u}\}.$$

A linear constraint atom a and its negation $\neg a$ 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 $\mathcal{L}_\mathcal{X}$ is called a rule if it is of form*

$$a_1 \vee \cdots \vee a_n \leftarrow l_1 \wedge \cdots \wedge l_{n'} \tag{3}$$

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 $\mathcal{L}_\mathcal{X}$.

A logic program is a theory of rules of form (3). Following logic programming syntax, we use ',' and ';' as alternative representations of $\wedge$ and $\vee$, respectively. Moreover, in this context we write $\varphi_1 \leftarrow \varphi_2$ for $\varphi_2 \rightarrow \varphi_1$ for formulas φ_1 and φ_2. Examples of programs over $\mathcal{L}_\mathcal{X}$ are given in the introduction.

Let r be a rule of form (3). Then, we define by $head(r) \stackrel{\text{def}}{=} \{a_i \mid 1 \leq i \leq n\}$ and $body(r) \stackrel{\text{def}}{=} \{l_j \mid 1 \leq j \leq n'\}$ the set of literals of the left and right hand side of r, respectively. Whenever $body(r) = \emptyset$, then we drop $\leftarrow$ and call r fact. If $head(r) = \emptyset$ we write $\bot \leftarrow l_1, \ldots, l_{n'}$. 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 $\mathcal{L}_\mathcal{X}$ containing a rule of form $a \leftarrow \neg a$ and for each HT_{LB}-stable model v of $P \setminus \{a \leftarrow \neg a\}$ over $\mathcal{X}, \mathbb{Z}$ 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, $w_1x_1, + \cdots + w_nx_n < k$ and $w_1x_1, + \cdots + w_nx_n > k$ can be expressed by $w_1x_1, + \cdots + w_nx_n \leq k - 1$ and $w_1x_1, + \cdots + w_nx_n \geq k + 1$, 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 $v' \subset v_1$ with $v' \in [\![x \geq 1]\!]$ and hence v_1 is a stable model of P. Furthermore, consider valuation $v'' \subset v_2$. Then, $\langle v'', v_2 \rangle \models x \geq 42 \leftarrow \neg x \leq 1$ 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 $v'' \subset v_2$ is contradicted. That is, v_2 is a stable model as well and $(x{\downarrow}1) \subset (x{\downarrow}42)$ 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 $v'(x) \neq \boldsymbol{u}$), where $v(y) = v'(y)$ for all $y \in vars(a) \setminus \{x\}$.[4] We define an atom a as *monotonic* (resp. *anti-monotonic*) if it is monotonic (resp. anti-monotonic) wrt all variables in $vars(a)$, 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. For instance, atom $x \geq 42$ is monotonic, $y < 42$ is anti-monotonic, and $x - y \geq 42$ is non-monotonic, since x is monotonic and y is anti-monotonic, respectively.

Thus, we define normal programs as follows.

Definition 8 *A rule over $\mathcal{L}_\mathcal{X}$ is normal if it is of form*

$$a_0 \leftarrow a_1, \ldots, a_n, \neg a_{n+1}, \ldots, \neg a_{n'} \tag{4}$$

where $|vars(a_0)| = 1$ 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 $\mathcal{L}_\mathcal{X}$. Then, each HT_{LB}-stable model of P over $\mathcal{X}, \mathbb{Z}$ 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:

$$x \geq 0 \qquad\qquad x \geq 42 \leftarrow y < 42$$
$$y \geq 0 \qquad\qquad y \geq 42 \leftarrow x < 42$$

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 $body^+(r) \stackrel{\text{def}}{=} \{a_i \mid 1 \leq i \leq n, a_i \text{ monotonic}\}$ and its negative body as $body^-(r) \stackrel{\text{def}}{=} body(r) \setminus body^+(r)$, 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 $\mathcal{L}_\mathcal{X}$ is positive if $head(r)$ is monotonic and $body^-(r) = \emptyset$.*

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 $\mathcal{L}_\mathcal{X}$. Then, P has exactly one HT_{LB}-stable model over $\mathcal{X}, \mathbb{Z}$.*

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 set 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 $\mathcal{L}_\mathcal{X}$ with monotonic heads only. Then, P has exactly one HT_{LB}-stable model over $\mathcal{X}, \mathbb{Z}$.*

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 $\mathcal{L}_\mathcal{X}$. Then, P has at most one HT_{LB}-stable model over $\mathcal{X}, \mathbb{Z}$.*

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

[4] Note that our definition of monotonicity of atoms differs from Aziz' ones (Aziz 2015), due to different concepts of valuations.

4.2 Modelling Capabilities

In this section, we go into an example of error diagnosis to illustrate some modelling features of HT_{LB} 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

$$error \geq \sum_{i \in X} e_i \leftarrow \bigwedge_{i \in X} occur(e_i) = 1 \quad \text{for all } X \subseteq N \quad (5)$$

$$occur(e_2) = 1 \leftarrow occur(e_3) = 1,\ error \geq 4 \quad (6)$$

$$occur(e_4) = 1 \leftarrow temperature \leq 42 \quad (7)$$

$$temperature = 60 \leftarrow \neg temperature \neq 60 \quad (8)$$

Rules of (5) express that the value of $error$ is greater or equal than the sum of occurred events.[5] The empty sum means that we have no error and is defined by 0. Rule (6) models the dependency of event e_2 regarding e_3 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 $occur(e_3) = 1$. Then, we get the single stable model $(temperature \downarrow 60) \cup (error \downarrow 3) \cup (occur(e_3) \downarrow 1)$ 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 $temperature \geq 42$. Then we get stable models $(temperature \downarrow 60) \cup (error \downarrow 3) \cup (occur(e_3) \downarrow 1))$ and $(temperature \downarrow 42) \cup (error \downarrow 9) \cup (occur(e_2) \downarrow 1) \cup (occur(e_3) \downarrow 1) \cup (occur(e_4) \downarrow 1)$ 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 $temperature \neq 60$ 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 HT_{LB} to existing formalisms from the literature.

5.1 $BFASP$

First, let us compare HT_{LB} 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. [6] 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 wrt their variables and by applying a fix point operation.

Both, $BFASP$ and HT_{LB} assign variables to their smallest domain value per default. Interestingly, they differ in their understanding of smallest domain values. In HT_{LB}, 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 $\mathbb{Z}_0^+$ 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 $\mathbb{Z}_0^+$. This is different from HT_{LB} 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 HT_{LB}, 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 HT_{LB}. 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 HT_{LB}, 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 HT_C

Next, we compare our approach to the logic of Here-and-There with constraints (HT_C; 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 HT_{LB} inherently minimizes valuations wrt foundedness. This is achieved by additionally comparing valuations wrt 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 $\{x\}, \mathbb{Z}$. Then, for valuations v and v' with $v(x) = 42$ and $v'(x) = 43$ in HT_C we have $v \neq v'$, whereas in HT_{LB} we have $v \subseteq v'$. Hence, both v and v' are stable models

[5] 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.

[6] For more details see (Aziz 2015).

in HT_C but only the first one is HT_{LB}-stable model, due to foundedness.

On a first look, HT_{LB} seems like HT_C with value minimization 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 HT_C assigns x to 1. This eliminates 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 HT_C, 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 HT_C does not yield HT_{LB}-stable models.

However, both HT_{LB} and HT_C define atomic satisfaction in terms of atom denotations. A difference is that in HT_C 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 $[\![x \neq 42]\!] = \{v \mid \boldsymbol{u} \neq v(x) \neq 42\}$ 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 $v \subseteq v'' \subseteq v'$ is not. The reason to be closed or not is that v, v' and v'' are different in HT_C but subsets in HT_{LB}, respectively.

The closure of denotions is significant to satisfy persistence in HT_C. In contrast, in HT_{LB} persistence is maintained by forcing atomic satisfaction in both h and t, instead of h only as in HT_C. The corresponding benefit is that this allows us to consider atoms in HT_{LB} which are not allowed in HT_C, like $x \doteq y$ with $[\![x \doteq y]\!] \stackrel{\text{def}}{=} \{v \mid v(x) = v(y)\}$ which is not closed in HT_C 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 HT_C 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 underlying theory. Technically, in (Janhunen et al. 2017) a program over a theory is extended by rules depending on possible assignments 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 HT_C, 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 aggregates 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 $f\{c_1 : \varphi_1, \ldots, c_m : \varphi_m\} \prec k$, where f is an aggregate symbol, c_i, k constants, φ_i 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 aggregate holds if the relation holds for all constants that belong to the set.

Obviously, (sum) aggregates are related to (linear constraint) 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 aggregates 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 monotonicity. 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 HT can be seen as a special case of HT_{LB}. To prove if our approach follows well-known patterns,

[7] Please see (Cabalar et al. 2016) for more details.

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

Apt, K.; Blair, H.; and Walker, A. 1987. Towards a theory of declarative knowledge. In Minker, J., ed., *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann Publishers. chapter 2, 89–148.

Aziz, R. 2015. *Answer Set Programming: Founded Bounds and Model Counting*. Ph.D. Dissertation, University of Melbourne.

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

Barrett, C.; Sebastiani, R.; Seshia, S.; and Tinelli, C. 2009. Satisfiability modulo theories. In Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds., *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press. chapter 26, 825–885.

Baselice, S.; Bonatti, P.; and Gelfond, M. 2005. Towards an integration of answer set and constraint solving. In Gabbrielli, M., and Gupta, G., eds., *Proceedings of the Twenty-first International Conference on Logic Programming (ICLP'05)*, volume 3668 of *Lecture Notes in Computer Science*, 52–66. Springer-Verlag.

Bomanson, J., and Janhunen, T. 2013. Normalizing cardinality rules using merging and sorting constructions. In Cabalar, P., and Son, T., eds., *Proceedings of the Twelfth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'13)*, volume 8148 of *Lecture Notes in Artificial Intelligence*, 187–199. Springer-Verlag.

Cabalar, P.; Kaminski, R.; Ostrowski, M.; and Schaub, T. 2016. An ASP semantics for default reasoning with constraints. In Kambhampati, R., ed., *Proceedings of the Twenty-fifth International Joint Conference on Artificial Intelligence (IJCAI'16)*, 1015–1021. IJCAI/AAAI Press.

Ferraris, P. 2005. Answer sets for propositional theories. In Baral, C.; Greco, G.; Leone, N.; and Terracina, G., eds., *Proceedings of the Eighth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'05)*, volume 3662 of *Lecture Notes in Artificial Intelligence*, 119–131. Springer-Verlag.

Ferraris, P. 2011. Logic programs with propositional connectives and aggregates. *ACM Transactions on Computational Logic* 12(4):25.

Gelfond, M., and Lifschitz, V. 1990. Logic programs with classical negation. In Warren, D., and Szeredi, P., eds., *Proceedings of the Seventh International Conference on Logic Programming (ICLP'90)*, 579–597. MIT Press.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9:365–385.

Gelfond, M., and Zhang, Y. 2014. Vicious circle principle and logic programs with aggregates. *Theory and Practice of Logic Programming* 14(4-5):587–601.

Heyting, A. 1930. Die formalen Regeln der intuitionistischen Logik. In *Sitzungsberichte der Preussischen Akademie der Wissenschaften*. Deutsche Akademie der Wissenschaften zu Berlin. 42–56. Reprint in Logik-Texte: Kommentierte Auswahl zur Geschichte der Modernen Logik, Akademie-Verlag, 1986.

Janhunen, T.; Kaminski, R.; Ostrowski, M.; Schaub, T.; Schellhorn, S.; and Wanko, P. 2017. Clingo goes linear constraints over reals and integers. *Theory and Practice of Logic Programming* 17(5-6):872–888.

Liu, G.; Janhunen, T.; and Niemelä, I. 2012. Answer set programming via mixed integer programming. In Brewka, G.; Eiter, T.; and McIlraith, S., eds., *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'12)*, 32–42. AAAI Press.

Pearce, D. 2006. Equilibrium logic. *Annals of Mathematics and Artificial Intelligence* 47(1-2):3–41.

Schrijver, A. 1999. *Theory of linear and integer programming*. Discrete mathematics and optimization. John Wiley & sons.

Sinz, C. 2005. Towards an optimal CNF encoding of Boolean cardinality constraints. In van Beek, P., ed., *Proceedings of the Eleventh International Conference on Principles and Practice of Constraint Programming (CP'05)*, volume 3709 of *Lecture Notes in Computer Science*, 827–831. Springer-Verlag.

Son, T., and Pontelli, E. 2007. A constructive semantic characterization of aggregates in answer set programming. *Theory and Practice of Logic Programming* 7(3):355–375.

Appendix of Proofs

Proof of Proposition 1 To prove that $\langle H, T\rangle \models \varphi$ holds under Definition 1 iff it holds when replacing Condition *2* by *2*' for $\langle H, T\rangle$ HT-interpretation over $\mathcal{A}$ and φ a propositional formula over $\mathcal{A}$, 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 $p \in \mathcal{A}$ that

$$H \in [\![p]\!]_{\mathcal{A}} \Leftrightarrow H \in \{I \mid p \in I\} \Leftrightarrow p \in H$$

□

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 $\mathcal{A}_\mathcal{X}$. Let $\langle h, t\rangle$ an HT_{LB}-valuation over $\mathcal{X}, \mathcal{D}$ and a atom of $\mathcal{A}_\mathcal{X}$.

First, we prove persistence, represented by *1* of the proposition. We have

$$\langle h, t\rangle \models a \Leftrightarrow h \in [\![a]\!] \wedge t \in [\![a]\!] \Rightarrow t \in [\![a]\!] \Leftrightarrow \langle t, t\rangle \models a$$

Subsequently, we prove negation, represented by *2* of the proposition. We have

$\langle h, t\rangle \models a \to \bot$
$\Leftrightarrow (\langle h, t\rangle \models \bot \vee \langle h, t\rangle \not\models a) \wedge (\langle t, t\rangle \models \bot \vee \langle t, t\rangle \not\models a)$
$\Leftrightarrow \langle h, t\rangle \not\models a \wedge \langle t, t\rangle \not\models a$
$\Leftrightarrow (h \notin [\![a]\!] \vee t \notin [\![a]\!]) \wedge (t \notin [\![a]\!])$
$\Leftrightarrow \langle t, t\rangle \not\models a$ □

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

Proof of Proposition 4 Let Γ_1, Γ_2 and Γ be theories over $\mathcal{A}_\mathcal{X}$. First, we prove "$\Leftarrow$" of the proposition. For each HT_{LB}-valuation $\langle h, t\rangle$ over $\mathcal{X}, \mathcal{D}$ we have $\langle h, t\rangle \models \Gamma_1$ iff $\langle h, t\rangle \models \Gamma_2$. This implies that $\langle h, t\rangle \models \Gamma_1 \cup \Gamma$ iff $\langle h, t\rangle \models \Gamma_2 \cup \Gamma$ for any Γ. Hence, $\Gamma_1 \cup \Gamma$ and $\Gamma_2 \cup \Gamma$ have the same HT_{LB}-stable models for every Γ.

Secondly, we prove "$\Rightarrow$" 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 $\Gamma = \{x \succeq c \mid (x, c) \in t\}$. 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 $\Gamma = \Gamma' \cup \Gamma''$ with $\Gamma' = \{x \succeq c \mid (x, c) \in h\}$ and $\Gamma'' = \{x \succeq t(x) \to y \succeq t(y),\ x \succeq c \to x \succeq t(x) \mid (x, c), (x, t(x)), (y, t(y)) \in t \setminus h, x \neq y\}$. Then, $\langle t, t\rangle \models \Gamma_2 \cup \Gamma$ by assumption and $h \subseteq t$. Note there is no $v \subset t$ with $\langle v, t\rangle \models \Gamma_2 \cup \Gamma$, since by $\langle h, t\rangle \not\models \Gamma_2$ we get that $h \subset v \subset t$ need to hold, and thus there exists at least one pair $a_1, a_2 \in atoms(\Gamma'')$ with $v \in [\![a_1]\!]$ and $v \notin [\![a_2]\!]$. Hence, $\langle v, t\rangle \not\models \Gamma_2 \cup \Gamma$ for $h \subseteq v \subset t$. Thus, $\langle t, t\rangle$ is HT_{LB}-stable model of $\Gamma_2 \cup \Gamma$. By assumption and construction, we have that $\langle h, t\rangle \models \Gamma_1$ and $\langle h, t\rangle \models \Gamma'$, respectively. Moreover, we have that $\langle h, t\rangle \not\models a$ for every $a \in atoms(\Gamma'')$. Hence, $\langle h, t\rangle \models \Gamma''$ and thus $\langle h, t\rangle \models \Gamma_1 \cup \Gamma$. Note that since $\langle h, t\rangle \not\models \Gamma_2$ and $\langle t, t\rangle \models \Gamma_2$ we have $\langle h, t\rangle \neq \langle t, t\rangle$, which implies that $h \subset t$. Finally, $\langle t, t\rangle$ is no HT_{LB}-stable model of $\Gamma_1 \cup \Gamma$. □

Proof of Proposition 5 Let a be an atom over $\mathcal{A}_\mathcal{X}$, and $\overline{a}$ and $\overline{a}^s$ its complement and its strict complement over $\mathcal{A}_\mathcal{X}$, respectively.

First, we prove $\overline{a}^s \models \overline{a}$. For any HT_{LB}-valuation $\langle h, t\rangle$ over $\mathcal{X}, \mathcal{D}$ we have

$\langle h, t\rangle \models \overline{a}^s$
$\Leftrightarrow h \in [\![\overline{a}^s]\!] \wedge t \in [\![\overline{a}^s]\!]$ with $[\![\overline{a}^s]\!] = 2^\mathfrak{V} \setminus ([\![a]\!] \cup \{v \mid v(x) = \boldsymbol{u} \text{ for some } x \in vars(a)\})$
$\Rightarrow h \in 2^\mathfrak{V} \setminus [\![a]\!] \wedge t \in 2^\mathfrak{V} \setminus [\![a]\!]$
$\Leftrightarrow \langle h, t\rangle \models \overline{a}$

Secondly, we prove $\overline{a} \models \neg a$. For any HT_{LB}-valuation $\langle h, t\rangle$ over $\mathcal{X}, \mathcal{D}$ we have

$\langle h, t\rangle \models \overline{a}$
$\Leftrightarrow h \in [\![\overline{a}]\!] \wedge t \in [\![\overline{a}]\!]$ with $[\![\overline{a}]\!] = 2^\mathfrak{V} \setminus [\![a]\!]$
$\Leftrightarrow h \notin [\![a]\!] \wedge t \notin [\![a]\!]$
$\Rightarrow t \notin [\![a]\!]$
$\overset{\text{Proposition 2}}{\Leftrightarrow} \langle h, t\rangle \models \neg a$ □

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 Γ 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 $\{p = \mathbf{t} \mid p \in \mathcal{A}\}$ and $\langle \tau(H), \tau(T)\rangle$ an HT_{LB}-valuation over $\mathcal{A}, \{\mathbf{t}\}$. Then we have

$\langle H, T\rangle \models p$
$\Leftrightarrow H \in [\![p]\!]_\mathcal{A}$
$\overset{H \subseteq T}{\Leftrightarrow} H \in [\![p]\!]_\mathcal{A} \wedge T \in [\![p]\!]_\mathcal{A}$
$\Leftrightarrow \tau(H) \in [\![p = \mathbf{t}]\!]_{\mathcal{A},\{\mathbf{t}\}} \wedge \tau(T) \in [\![p = \mathbf{t}]\!]_{\mathcal{A},\{\mathbf{t}\}}$
$\Leftrightarrow \langle \tau(H), \tau(T)\rangle \models p = \mathbf{t}$ □

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 $\mathcal{A}_\mathcal{X}$.

First, note that the pair $\langle H, T\rangle$ over $\mathcal{A}_\mathcal{X}$ with $H = At(\langle h, t\rangle)$ and $T = At(\langle t, t\rangle)$ is a well formed HT-interpretation, since $H \subseteq T$ holds by $h \subseteq t$ and Proposition 3. Then we have

$\langle h, t\rangle \models a$
$\Leftrightarrow h \in [\![a]\!]_{\mathcal{X},\mathcal{D}} \wedge t \in [\![a]\!]_{\mathcal{X},\mathcal{D}}$
$\Rightarrow H \in [\![a]\!]_{\mathcal{A}_\mathcal{X}} \wedge T \in [\![a]\!]_{\mathcal{A}_\mathcal{X}}$
$\Rightarrow \langle H, T\rangle \models a$ □

Proof of Proposition 8 Let φ 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 φ' over $\mathcal{A}_\mathcal{X}$ obtained by replacing $atoms(\varphi)$ in φ by atoms of $\mathcal{A}_\mathcal{X}$, is a tautology as well (for every HT_{LB}-valuation $\langle h, t\rangle$ over $\mathcal{X}, \mathcal{D}$ holds $\langle h, t\rangle \models \varphi'$), 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 $\mathcal{A}_\mathcal{X}$.

Let Γ be a theory over $\mathcal{A}_\mathcal{X}$ and $\langle h, t\rangle$ an HT_{LB}-valuation over $\mathcal{X}, \mathcal{D}$. Then, we have

$h \models_{cl} a^t$
$\Leftrightarrow h \models_{cl} a \wedge t \models_{cl} a$
$\Leftrightarrow h \in [\![a]\!] \wedge t \in [\![a]\!]$
$\Leftrightarrow \langle h, t\rangle \models a$ □