SM-based Semantics for Answer Set Programs Containing Conditional Literals and Arithmetic

Zachary Hansen[0000−0002−8447−4048] and Yuliya Lierler[0000−0002−6146−623X]

University of Nebraska Omaha, Omaha NE 68106, USA

Abstract. Modern answer set programming solvers such as CLINGO support advanced language constructs that improve the expressivity and conciseness of logic programs. Conditional literals are one such construct. They form “subformulas” that behave as nested implications within the bodies of logic rules. Their inclusion brings the form of rules closer to the less restrictive syntax of first-order logic. These qualities make conditional literals useful tools for knowledge representation. In this paper, we propose a semantics for logic programs with conditional literals and arithmetic based on the SM operator. These semantics do not require grounding, unlike the established semantics for such programs that relies on a translation to infinitary propositional logic. The main result of this paper establishes the precise correspondence between the proposed and existing semantics.

Keywords: Answer Set Programming · Conditional Literals · Semantics.

1 Introduction

Answer Set Programming (ASP) [24,25] is a declarative programming paradigm that has been applied within a variety of challenging and high-consequence systems such as explainable donor-patient matching [6], space shuttle decision support systems [2,3], train scheduling [1], robotics [16], and automated fault diagnosis [29]. ASP programs are concise, human-readable, and benefit from well-defined semantics rooted in mathematical logic – these qualities make ASP programs attractive candidates for formal verification [5]. Providing high levels of assurance regarding program behavior is particularly crucial for safety-critical applications. This paper is part of a research stream with the long-term goal of supporting rigorous verification of ASP systems.

Conditional literals are powerful language features for knowledge representation. Originating in the LPARSE grounder [27], these languages features are now also supported by the answer set solver CLINGO [14,15]. Intuitively, they represent a nested implication within the body of an ASP rule [17]. Rules with conditional literals concisely express knowledge that may be difficult to otherwise encode. For instance, conditional literals are widely employed in meta-programming – Listings 4-7 in “How to build your own ASP-based system?!" by Kaminski et al. [20] define meta encodings which compute the classical and supported models of reified logic programs; these encodings rely heavily on conditional literals.

Conditional literals may also make programs easier to formally verify by reducing the number of auxiliary or inessential predicates in a program. Consider Listing 1.1, which contains a typical encoding of the graph coloring problem.

Listing 1.1: Graph coloring problem encoding.

```
1 {asg(V, C)} :- vtx(V), col(C).
2 :- asg(V, C1), asg(V, C2), C1 != C2.
3 colored(V) :- asg(V,C).
4 :- vtx(V), not colored(V).
5 :- asg(V1, C), asg(V2, C), edge(V1, V2).
```

The program in this listing can be simplified by replacing lines 3-4 with the following *constraint* (a rule with an empty head) containing a conditional literal:

$$\text{:- } \textit{not } asg(V, C) \ : \ col(C);\ vtx(V). \qquad (1)$$

This simplification is attractive since it eliminates the auxiliary predicate $colored/1$ (introduced in line 3 for the sole purpose of stating the subsequent constraint in line 4). We will use rule (1) as a running example in the remainder of the paper.

Typically, the semantics of programs with variables are defined indirectly, via a procedure called *grounding*. Grounding turns a given program with variables into a propositional one. Then, semantics are defined for the resulting propositional program. This hampers our ability to reason about the behavior of programs independently of a specific grounding context. In 2011, Ferraris, Lee, and Lifschitz proposed semantics for answer set programs that bypasses grounding [13]. They introduced the SM operator, which turns a program (or, rather, the first-order logic formula associated with the considered program) into a classical second-order formula. The Herbrand models of this formula coincide with the answer sets of the original program.

Since then, that approach has been generalized to cover such features of ASP input languages as aggregates [7,10], arithmetic [8,21], and conditional literals [17]. Yet, all of these features were addressed independently of the others. This paper helps to close that gap. Here, we introduce grounding-free SM operator-based semantics for logic programs containing *both* conditional literals and arithmetic, combining ideas from earlier work on these features [8,17,21]. We also show that the proposed characterization coincides with the existing semantics for such programs based on grounding to infinitary propositional logic [15]; these are the semantics adhered to by CLINGO.

One of the advantages of the SM-based characterization is that it enables us to construct proofs of correctness in a modular way that does not rely on grounding the program with respect to a specific instance of input data. For instance, this style of verification – which exploits the SM operator's ability to divide programs into modules – has been employed to demonstrate the adherence of Graph Coloring, Hamiltonian Cycle, and Traveling Salesman problems to natural language specifications [5,9]. This paper extends the class of programs for which such arguments can be constructed.

Section 2 defines the language of logic programs considered, and Section 3 provides the essence of the SM characterization for logic programs with conditional literals and arithmetic. Section 4 reviews the established semantics for programs with conditional literals and arithmetic, which relies on a translation from logic programs to infinitary propositional formulas. The main results of this paper –Theorems 1 and 2 – are given in Section 5, connecting our SM-based semantics to the established semantics.

2 Syntax of Logic Programs

We now present the language of logic programs considered in this paper. It can be viewed as a fragment of the Abstract Gringo (AG) language [15]; equivalently, it can be viewed as an extension of the mini-GRINGO language [21] to rules whose bodies may contain conditional literals.

We assume a *(program) signature* with three countably infinite sets of symbols: *numerals*, *symbolic constants* and *variables*. We also assume a 1-to-1 correspondence between numerals and integers; the numeral corresponding to an integer n is denoted by $\overline{n}$. A syntactic expression is *ground* if it contains no variables. A ground expression is *precomputed* if it contains no operation names. *Terms* are defined recursively:

- Numerals, symbolic constants, variables, or either of the special symbols *inf* and *sup* are terms;
- if t_1, t_2 are program terms and $\circ$ is one of the *operation names*
$$+ \quad - \quad \times \quad / \quad \backslash \quad .. \tag{2}$$
then $t_1 \circ t_2$ is a term (we write $-t$ to abbreviate the term $\overline{0} - t$);
- if t_1 is a program term, then $|t_1|$ is a term.

We assume that a total order on ground terms is chosen such that

- *inf* is its least element and *sup* is its greatest element,
- for any integers m and n, $\overline{m} < \overline{n}$ iff $m < n$, and
- for any integer n and any symbolic constant c, $\overline{n} < c$.

A comparison is an expression of the form $t_1 \prec t_2$, where t_1 and t_2 are terms and $\prec$ is one of the comparison symbols:
$$= \quad \neq \quad < \quad > \quad \leq \quad \geq \tag{3}$$

An *atom* is an expression of the form $p(\mathbf{t})$, where p is a symbolic constant and $\mathbf{t}$ is a list of program terms. A *basic literal* is an atom possibly preceded by one or two occurrences of *not*. A *conditional literal* is an expression of the form
$$H : l_1, \ldots, l_m,$$
where H is either a comparison, a basic literal, or the symbol $\bot$ (denoting falsity) and $l_1, \ldots, l_m$ is a list of basic literals and comparisons. We often abbreviate such an expression as $H : \mathbf{L}$. If $m = 0$, then the preceding ":" is dropped (so that the program stays CLINGO-compliant [15]). We view basic literals and comparisons as conditional literals with an empty list of conditions, i.e., $m = 0$. A *rule* is an expression of the form
$$Hd \text{ :- } B_1, \ldots, B_n, \tag{4}$$
where

- Hd is either an atom (a normal rule), or an atom in braces (a choice rule), or the symbol $\bot$ (a constraint);

– each B_i ($1 \leq i \leq n$) is a conditional literal.

We call the symbol :- a *rule operator*. We call the left hand side of the rule operator the *head*, the right hand side of the rule operator the *body*. The symbol $\bot$ may be omitted from the head, resulting in an empty head. Such rules are called *constraints*. If the body of the rule is empty, the rule operator will be omitted, resulting in a *fact*. A *program* is a finite set of rules.

3 Semantics of Logic Programs via the SM Operator

Here, we introduce the SM operator-based semantics for logic programs written in the syntax of Section 2. Subsections reviewing necessary concepts are prefixed by the word *preliminaries*. The introduction of these semantics is split into two parts. The first part is given in Section 3.1, where a translation from a logic program to a many-sorted first order theory is provided. This section builds on earlier translations by Fandinno et al. [8,12] and Hansen and Lierler [17]. Section 3.2 provides us with the details of the second part, where we start by reviewing the SM operator and conclude with the definition of answer sets for the considered logic programs. These programs are translated into first-order theories and then the SM operator is applied. Certain models of the resulting formula that we call "standard" correspond to answer sets.

3.1 Translation τ^* Extended

In this section, we introduce an extension of the τ^* translation from logic programs to many-sorted first-order theories (we refer to the introduced extension with the same symbol τ^*). This extension combines elements of the most recent incarnation of the τ^* transformation [12] with the translation ϕ for conditional literals [17]. Following the example of past work [7], we extend the τ^B component of the τ^* translation with special treatment for global variables.

Preliminaries: The Target Language of τ^* A *signature* σ consists of *function* and *predicate* constants in addition to a set of *sorts*. For every sort s, a many-sorted interpretation $\mathscr{I}$ has a non-empty universe $|\mathscr{I}|^s$ (we further assume that there are infinitely many variables for each sort). A reflexive and transitive *subsort* relation $\prec$ is defined on the set of sorts such that when sort $s_1 \prec s_2$, an interpretation $\mathscr{I}$ satisfies the condition $|\mathscr{I}|^{s_1} \subseteq |\mathscr{I}|^{s_2}$. The *function signature* of every function constant f consists of a tuple of *argument sorts* $s_1, \ldots, s_n$, and *value sort* s, denoted by $s_1 \times \cdots \times s_n \to s_{n+1}$.

Object constants are function constants with $n = 0$, their function signature contains only a value sort. Similarly, the *predicate signature* of every predicate constant p is a tuple of argument sorts $s_1 \times \cdots \times s_n$. A predicate constant whose predicate signature is the empty tuple is called a *proposition*. The *arity* of a function or predicate signature with n argument sorts is n.

Terms of a signature σ are constructed recursively from function constants. Atomic formulas are built similar to the standard unsorted logic with the restriction that in a term $f(t_1, \ldots, t_n)$ (an atom $p(t_1, \ldots, t_n)$, respectively), the sort of term t_i must be a subsort of the i-th argument of f (of p, respectively). In addition, $t_1 = t_2$ is an atomic formula if the sorts of t_1 and t_2 have a common supersort. The notion of satisfaction is analogous to the unsorted case with the restriction that an interpretation maps a term to an element in the universe of its associated sort.

Our translation τ^* transforms a logic program Π written in the syntax of Section 2 into a first-order sentence with equality over a signature σ_Π of *two sorts*. The first sort is called the *program sort* (denoted s_p); all program terms are of this sort. The second sort is called the *integer sort* (denoted s_i); it is a subsort of the program sort. Specifically, the variables of s_i range over numerals. Variants of X, Y, Z will denote variables of sort s_p and variants of I, J, M, N will denote integer variables. Bold face variants will denote lists of variables. To define the remainder of this signature, we must introduce the concepts of *occurrences* and *global variables*.

A *predicate symbol* is a pair p/n, where p is a symbolic constant and n is a non-negative integer. About a program or other syntactic expression, we say that a predicate symbol p/n *occurs* in it if it contains an atom of the form $p(t_1,\dots,t_n)$.

A variable is *global* in a conditional literal $H : \mathbf{L}$ if it occurs in H but not in $\mathbf{L}$. Thus, any variables in basic literals or comparisons are also global. A variable is global in a rule if it is global in any of the rule's expressions. All non-global variables in an expression are called *local*.

For a program Π, signature σ_Π contains:

1. all precomputed terms as object constants of the program sort; a precomputed constant is assigned the sort s_i iff it is a numeral;
2. all predicate symbols occurring in Π as predicate constants with all arguments of the sort s_p;
3. the comparison symbols other than equality and inequality as predicate constants with predicate signature $s_p \times s_p$ (we will use infix notation for constructing these atoms);
4. function constants $+$, $-$, and $\times$ with function signature $s_i \times s_i \rightarrow s_i$, and function constant $|\cdot|$ with function signature $s_i \rightarrow s_i$;

Preliminaries: Values of Terms In the language of Section 2, a term may have one value (as in $3+5$), many values (as in $3+1..5$), or no values (as in $a+3$). Thus, for every program term t, we define a formula $val_t(Z)$, where Z is a program variable with no occurrences in t. It indicates that Z is a value of t.

– if t is a numeral, symbolic constant, program variable, *inf*, or *sup*, then $val_t(Z)$ is $Z = t$;
– if t is $|t_1|$, then $val_t(Z)$ is $\exists I(val_{t_1}(I) \wedge Z = |I|)$;
– if t is $(t_1 \circ t_2)$, where $\circ$ is one of $+$, $-$, or $\times$, then $val_t(Z)$ is

$$\exists IJ(Z = I \circ J \wedge val_{t_1}(I) \wedge val_{t_2}(J))$$

where I, J are fresh integer variables;
– if t is t_1/t_2 then $val_t(Z)$ is

$$\exists IJK(val_{t_1}(I) \wedge val_{t_2}(J) \wedge F_1(IJK) \wedge F_2(IJKZ))$$

where $F_1(IJK)$ is

$$K \times |J| \leq |I| < (K + \overline{1}) \times |J|$$

and $F_2(IJKZ)$ is

$$(I \times J \geq \overline{0} \wedge Z = K) \vee (I \times J < \overline{0} \wedge Z = -K)$$

– if t is $t_1 \setminus t_2$ then $val_t(Z)$ is

$$\exists IJK(val_{t_1}(I) \wedge val_{t_2}(J) \wedge F_1(IJK) \wedge F_3(IJKZ))$$

where $F_3(IJKZ)$ is

$$(I \times J \geq \overline{0} \wedge Z = I - K \times J) \vee (I \times J < \overline{0} \wedge Z = I + K \times J)$$

– if t is $t_1..t_2$ then $val_t(Z)$ is

$$\exists IJK(Z = K \wedge I \leq K \leq J \wedge val_{t_1}(I) \wedge val_{t_2}(J))$$

where $I \leq K \leq J$ is an abbreviation for $I \leq K \wedge K \leq J$.

Translation τ^* We now describe a translation τ^* that converts a program into a finite set of first-order sentences. It will be helpful to consider additional notation. For a tuple of terms $t_1, \dots, t_k$, abbreviated as $\mathbf{t}$, and a tuple of variables $V_1, \dots, V_k$, abbreviated as $\mathbf{V}$, we use $val_{\mathbf{t}}(\mathbf{V})$ to denote the formula

$$val_{t_1}(V_1) \wedge \cdots \wedge val_{t_k}(V_{t_k}).$$

Now we introduce $\tau^B_{\mathbf{Z}}$. It extends the τ^B translation [12] with a translation for conditional literals. We use the $\mathbf{Z}$ subscript to denote the set of global variables present in a rule. Given a list $\mathbf{Z}$ of global variables in some rule R, we define $\tau^B_{\mathbf{Z}}$ for all elements of R as follows:

1. $\tau^B_{\mathbf{z}}(\bot)$ is $\bot$;
2. $\tau^B_{\mathbf{z}}(p(\mathbf{t}))$ is $\exists \mathbf{V}(val_{\mathbf{t}}(\mathbf{V}) \wedge p(\mathbf{V}))$ for every basic literal $p(\mathbf{t})$;
3. $\tau^B_{\mathbf{z}}(not\ p(\mathbf{t}))$ is $\exists \mathbf{V}(val_{\mathbf{t}}(\mathbf{V}) \wedge \neg p(\mathbf{V}))$ for every basic literal *not* $p(\mathbf{t})$;
4. $\tau^B_{\mathbf{z}}(not\ not\ p(\mathbf{t}))$ is $\exists \mathbf{V}(val_{\mathbf{t}}(\mathbf{V}) \wedge \neg\neg p(\mathbf{V}))$ for every basic literal *not not* $p(\mathbf{t})$
5. $\tau^B_{\mathbf{z}}(t_1 \prec t_2)$ is $\exists Z_1 Z_2(val_{t_1}(Z_1) \wedge val_{t_2}(Z_2) \wedge Z_1 \prec Z_2)$ for every comparison $t_1 \prec t_2$;
6. $\tau^B_{\mathbf{z}}(\mathbf{L})$ is $\tau^B_{\mathbf{z}}(l_1) \wedge \cdots \wedge \tau^B_{\mathbf{z}}(l_m)$ for a list $\mathbf{L}$ of basic literals and comparisons;
7. $\tau^B_{\mathbf{z}}(H : \mathbf{L})$ is
$$\forall \mathbf{x} \left(\tau^B_{\mathbf{z}}(\mathbf{L}) \to \tau^B_{\mathbf{z}}(H)\right)$$
for every conditional literal $H : \mathbf{L}$ with local variables $\mathbf{x}$ occurring in the body of R.

In what follows, for each rule R, $\mathbf{Z}$ denotes the list of the global variables of R, and $\mathbf{V}$ denotes a list of fresh, alphabetically first program variables. We now define the translation τ^*.

1. For a basic rule R of the form $p(\mathbf{t})$:- $B_1, \dots, B_n$, its translation $\tau^* R$ is

$$\widetilde{\forall} \left(val_{\mathbf{t}}(\mathbf{V}) \wedge \tau^B_{\mathbf{Z}}(B_1) \wedge \cdots \wedge \tau^B_{\mathbf{Z}}(B_n) \to p(\mathbf{V})\right).$$

2. For a choice rule R of the form $\{p(\mathbf{t})\}$:- $B_1, \ldots, B_n$, its translation $\tau^* R$ is

$$\widetilde{\forall}\big(val_{\mathbf{t}}(\mathbf{V}) \wedge \tau^B_{\mathbf{Z}}(B_1) \wedge \cdots \wedge \tau^B_{\mathbf{Z}}(B_n) \wedge \neg\neg p(\mathbf{V}) \to p(\mathbf{V})\big).$$

3. For a constraint R of the form $\bot$:- $B_1, \ldots, B_n$, its translation $\tau^* R$ is

$$\forall \mathbf{Z}\big(\tau^B_{\mathbf{Z}}(B_1) \wedge \cdots \wedge \tau^B_{\mathbf{Z}}(B_n) \to \bot\big).$$

4. For every program Π, its translation $\tau^* \Pi$ is the first-order theory containing $\tau^* R$ for each rule R in Π.

Example 1. For a list of global variables $\{V\}$, $\tau^B_{\{V\}}(not\ asg(V,\ C)\ :\ col(C))$ is

$$\forall C(\exists Z(Z = C \wedge col(Z)) \to \exists Z Z_1(Z = V \wedge Z_1 = C \wedge \neg asg(Z, Z_1)))$$

Thus, the translation of rule (1) is

$$\forall V(\tau^B_{\{V\}}(not\ asg(V,\ C)\ :\ col(C)) \wedge \exists Z(Z = V \wedge vtx(Z)) \to \bot).$$

3.2 Semantics via Many-sorted SM

Preliminaries: The SM **Operator for Many-sorted Signatures** This subsection reviews an extension of the SM operator [13] to the many-sorted setting [7]. This operator is applied to a set of (many-sorted) first-order sentences corresponding to the *formula representation* of a logic program to obtain a set of (many-sorted) second-order sentences. Models of this set respecting certain assumptions (such as a Herbrand interpretation of symbolic constants) capture the stable models of the original logic program. We use τ^* to obtain the formula representation of a program in a specific many-sorted signature σ_Π, and apply SM to the result to characterize stable models of the program.

If p and u are predicate constants or variables with the same predicate signature, then $u \leq p$ stands for the formula

$$\forall \mathbf{W}(u(\mathbf{W}) \to p(\mathbf{W})),$$

where $\mathbf{W}$ is an n-tuple of distinct object variables. If $\mathbf{p}$ and $\mathbf{u}$ are tuples $p_1, \ldots, p_n$ and $u_1, \ldots, u_n$ of predicate constants or variables such that each p_i and u_i have the same predicate signature, then $\mathbf{u} \leq \mathbf{p}$ stands for the conjunction $(u_1 \leq p_1) \wedge \cdots \wedge (u_n \leq p_n)$, and $\mathbf{u} < \mathbf{p}$ stands for $(\mathbf{u} \leq \mathbf{p}) \wedge \neg(\mathbf{p} \leq \mathbf{u})$. For any many-sorted first-order formula F and a list $\mathbf{p}$ of predicate constants, by $\text{SM}_{\mathbf{p}}[F]$ we denote the second-order formula

$$F \wedge \neg\exists \mathbf{u}\big((\mathbf{u} < \mathbf{p}) \wedge F^*(\mathbf{u})\big)$$

where $\mathbf{u}$ is a list of distinct predicate variables $u_1, \ldots, u_n$ of the same length as $\mathbf{p}$, such that the predicate signature of each u_i is the same as the predicate signature of p_i, and $F^*(\mathbf{u})$ is defined recursively:

- $F^* = F$ for any atomic formula F that does not contain members of $\mathbf{p}$,
- $p_i(\mathbf{t})^* = u_i(\mathbf{t})$ for any predicate symbol p_i belonging to $\mathbf{p}$ and any list $\mathbf{t}$ of terms,

– $(F \wedge G)^* = F^* \wedge G^*$,
– $(F \vee G)^* = F^* \vee G^*$,
– $(F \rightarrow G)^* = (F^* \rightarrow G^*) \wedge (F \rightarrow G)$,
– $(\forall x F)^* = \forall x F^*$,
– $(\exists x F)^* = \exists x F^*$.

Definition 1. *For a many-sorted first-order sentence F from signature σ, the models of $SM_{\mathbf{p}}[F]$ are called the* **p***-stable models of F. For a set Γ of first-order sentences, the* **p***-stable models of Γ are the* **p***-stable models of the conjunction of all formulas in Γ.*

The list **p** of predicates of a **p**-stable model are called *intensional* – “belief” in these predicates is minimized. Predicates that are not intensional are called *extensional.*

Answer Sets via Standard Interpretations Recall from Section 3.1 that τ^* maps a program Π with intensional predicates **p** into a formula within signature σ_Π of two sorts. We now consider a special type of interpretation of σ_Π (a *standard* interpretation) to restrict the models of $\mathrm{SM}_{\mathbf{p}}[\tau^*\Pi]$ to exactly the stable models of Π.

A standard interpretation $\mathscr{I}$ satisfies that

1. the universe $|\mathscr{I}|^{s_p}$ is the set of all precomputed terms;
2. the universe $|\mathscr{I}|^{s_i}$ is the set of all numerals;
3. $\mathscr{I}$ interprets every precomputed term t as t;
4. $\mathscr{I}$ interprets $\overline{m}+\overline{n}$ as $\overline{m+n}$, and similarly for subtraction and multiplication;
5. $\mathscr{I}$ interprets $|\overline{n}|$ as $\overline{|n|}$;
6. $\mathscr{I}$ interprets every comparison $t_1 \prec t_2$, where t_1 and t_2 are precomputed terms, as true iff the relation $\prec$ holds for the pair (t_1, t_2).

For a standard interpretation $\mathscr{I}$, we use $A(\mathscr{I})$ to denote the unique set of precomputed atoms to which $\mathscr{I}$ assigns the value *true*.

Definition 2. *Let Π be a program and let* **p** *be a list of some predicate symbols occurring in Π other than the comparison symbols. Then, for a standard interpretation $\mathscr{I}$, we call $A(\mathscr{I})$ a* **p**-answer set *of Π when $\mathscr{I}$ is a* **p***-stable model of $\tau^*\Pi$. When* **p** *is the list of all predicate symbols occurring in Π other than the comparison symbols, then we simply call a* **p**-answer set *of Π an* answer set *of Π.*

In the sequel, we illustrate how answer sets as defined here coincide with the notion of what we later call gringo answer sets. Yet, it is interesting to note that SM-operator based semantics enable a more flexible understanding of a **p**-answer set that distinguishes intensional (namely, **p**) and extensional predicate symbols.

4 Review: Semantics of Logic Programs via Infinitary Propositional Logic

Truszczyński (2012) provides a characterization of logic program semantics in terms of infinitary propositional logic (IPL) and illustrates the precise relation to the SM-based characterization [28]. IPL is a useful technical device due to its close relation to the grounding procedure implemented by answer set solver CLINGO. In this section, we review IPL and the infinitary logic of here-and-there, which are important tools for establishing the results of this paper. Unless otherwise specified, P and its variants are used to denote IPL formulas, whereas $\mathscr{P}$ and its variants denote sets of IPL formulas. Finally, we introduce *gringo answer sets*, which constitute the established semantic characterization of logic programs with conditional literals and arithmetic.

4.1 Infinitary Formulas

A *propositional signature* Σ is a set of propositional atoms. For every nonnegative integer r, *(infinitary) formulas* over Σ of rank r are defined recursively:

- every atom from Σ is a formula of rank 0,
- if $\mathscr{P}$ is a set of formulas, and r is the smallest nonnegative integer that is greater than the ranks of all elements of $\mathscr{P}$, then $\mathscr{P}^\wedge$ and $\mathscr{P}^\vee$ are formulas of rank r,
- if P and P' are formulas, and r is the smallest nonnegative integer that is greater than the ranks of P and P', then $P \to P'$ is a formula of rank r.

$P \wedge P'$ is shorthand for $\{P, P'\}^\wedge$, and $P \vee P'$ is shorthand for $\{P, P'\}^\vee$. We use $\top$ and $\bot$ as abbreviations for $\emptyset^\wedge$ and $\emptyset^\vee$, respectively. Further, $\neg P$ stands for $P \to \bot$, and $P \leftrightarrow P'$ stands for $(P \to P') \wedge (P' \to P)$.

A *propositional interpretation* of Σ is a subset S of Σ. The *satisfaction relation* between a propositional interpretation and an infinitary formula is defined recursively:

- For every atom p from Σ, $S \models p$ if $p \in S$.
- $S \models \mathscr{P}^\wedge$ if, for every formula P in $\mathscr{P}$, $S \models P$.
- $S \models \mathscr{P}^\vee$ if there is a formula P in $\mathscr{P}$ such that $S \models P$.
- $S \models P \to P'$ if $S \not\models P$ or $S \models P'$.

4.2 Infinitary Logic of Here-and-there and Truszczyński Stable Models

An *HT-interpretation* of propositional signature Σ is an ordered pair $\langle S, S' \rangle$ of interpretations of Σ such that $S \subseteq S'$. The *satisfaction relation (ht-satisfaction)* between an HT-interpretation and an infinitary formula is defined recursively:

- For every atom p from Σ, $\langle S, S' \rangle \models p$ if $p \in S$.
- $\langle S, S' \rangle \models \mathscr{P}^\wedge$ if, for every formula P in $\mathscr{P}$, $\langle S, S' \rangle \models P$.
- $\langle S, S' \rangle \models \mathscr{P}^\vee$ if there is a formula P in $\mathscr{P}$ such that $\langle S, S' \rangle \models P$.
- $\langle S, S' \rangle \models P \to P'$ if
  1. $S' \models P \to P'$, and
  2. $\langle S, S' \rangle \not\models P$ or $\langle S, S' \rangle \models P'$.

An *HT-model* of an infinitary formula is an HT-interpretation that satisfies this formula. An *HT-model* of a set $\mathscr{P}$ of infinitary formulas is an HT-interpretation that satisfies all formulas in $\mathscr{P}$. Two infinitary formulas (sets of infinitary formulas) are *equivalent* when they have the same HT-models. An HT-interpretation of the form $\langle S, S \rangle$ is called *total*. An *equilibrium model* of a set $\mathscr{P}$ of infinitary formulas is a total HT-model $\langle S', S' \rangle$ of $\mathscr{P}$ such that for every proper subset S of S', $\langle S, S' \rangle$ is not an HT-model of $\mathscr{P}$.

An interpretation satisfies a set $\mathscr{P}$ of formulas (is a model of $\mathscr{P}$) if it satisfies all formulas in $\mathscr{P}$.

Definition 3. *For a set $\mathscr{P}$ of infinitary propositional formulas, we say that a propositional interpretation S is a* Truszczyński stable model *if $\langle S, S\rangle$ is an equilibrium model of $\mathscr{P}$.*

In the sequel, we may refer to a Truszczyński stable model of an infinitary propositional formula, where we identify this formula with a singleton set containing it.

4.3 Translation τ and Gringo Answer Sets

In this section we review the relevant components of τ [12,15]. It is due to note that the 2015 publication is the "official" source of the Abstract Gringo semantics and contains a τ translation for conditional literals, whereas the 2024 publication is more up-to-date with respect to the definitions of division, modulo, and absolute value adhered to by the latest versions of CLINGO.

An expression is *ground* if it does not contain variables. For every ground term t, the set of precomputed terms $[t]$ of its *values* is defined as follows:

- if t is a numeral, symbolic constant, or *inf* or *sup* then $[t]$ is $\{t\}$;
- if t is $|t_1|$, then $[t]$ is the set of numerals $\overline{|n|}$ for all integers n such that $\overline{n} \in [t_1]$;
- if t is $(t_1 \circ t_2)$, where $\circ$ is one of $+, -, \times$, then $[t]$ is the set of numerals $\overline{n_1 \circ n_2}$ such that $\overline{n_1} \in [t_1]$, $\overline{n_2} \in [t_2]$;
- if t is (t_1/t_2), then $[t]$ is the set of numerals $\overline{round(n_1/n_2)}$ for all integers n_1, n_2 such that $\overline{n_1} \in [t_1]$, $\overline{n_2} \in [t_2]$ and $n_2 \neq 0$;
- if t is $(t_1 \setminus t_2)$, then $[t]$ is the set of numerals $\overline{n_1 - n_2 \cdot round(n_1/n_2)}$ for all integers n_1, n_2 such that $\overline{n_1} \in [t_1]$, $\overline{n_2} \in [t_2]$ and $n_2 \neq 0$;
- if t is $(t_1..t_2)$, then $[t]$ is the set of numerals $\overline{m}$ for all integers m such that, for some integers n_1, n_2

$$\overline{n_1} \in [t_1], \quad \overline{n_2} \in [t_2], \quad n_1 \leq m \leq n_2.$$

- if $\mathbf{t}$ is a tuple of terms $t_1, \dots, t_n$ ($n > 0$) then $[\mathbf{t}]$ is the set of tuples $\langle r_1, \dots, r_n\rangle$ for all $r_1 \in [t_1]$, $\dots$, $r_n \in [t_n]$.

The function *round* is defined as:

$$round(n) = \begin{cases} \lfloor n \rfloor & \text{if } n \geq 0 \\ \lceil n \rceil & \text{if } n < 0 \end{cases} \tag{5}$$

A rule (or any other expression in a rule) is called *closed* if it contains no global variables. An *instance* of a rule R is any rule that can be obtained from R by substituting precomputed terms for all global variables.

To transform a closed rule R into a set of infinitary propositional formulas, translation τ is defined as follows:

1. $\tau(\bot)$ is $\bot$;
2. $\tau(p(\mathbf{t}))$ is the disjunction of atoms $p(\mathbf{r})$ over all tuples $\mathbf{r}$ in $[\mathbf{t}]$ for any ground atom $p(\mathbf{t})$, thus, $\tau(p(\mathbf{t}))$ is $p(\mathbf{t})$ if $\mathbf{t}$ is a tuple of precomputed terms;
3. $\tau(\textit{not } A)$ is $\neg\tau A$ for any ground atom A;
4. $\tau(\textit{not not } A)$ is $\neg\neg\tau A$ for any ground atom A;

5. $\tau(t_1 \prec t_2)$ is $\top$ if the relation $\prec$ holds between the terms r_1, r_2 for some r_1, r_2 such that $r_1 \in [t_1]$ and $r_2 \in [t_2]$, and $\bot$ otherwise;
6. $\tau(\mathbf{L})$ is $\tau(l_1) \wedge \cdots \wedge \tau(l_m)$ for a list $\mathbf{L}$ of basic or conditional literals;
7. for a closed conditional literal $H : \mathbf{L}$ occurring in the body of rule R, $\tau(H : \mathbf{L})$ is the conjunction of the formulas $\tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \rightarrow \tau(H^{\mathbf{x}}_{\mathbf{r}})$ where $\mathbf{x}$ is the list of variables occurring in the conditional literal, over all tuples $\mathbf{r}$ of precomputed terms of the same length as $\mathbf{x}$;
8. for an instance ρ of
   - a basic rule $p(\mathbf{t})$:- $B_1, \ldots, B_n$, its translation $\tau\rho$ is
   $$\tau(B_1 \wedge \cdots \wedge B_n) \rightarrow \bigwedge_{\mathbf{r} \in [\mathbf{t}]} p(\mathbf{r}); \tag{6}$$
   - a choice rule $\{p(\mathbf{t})\}$:- $B_1, \ldots, B_n$, its translation $\tau\rho$ is
   $$\tau(B_1 \wedge \cdots \wedge B_n) \rightarrow \bigwedge_{\mathbf{r} \in [\mathbf{t}]} (p(\mathbf{r}) \vee \neg p(\mathbf{r})); \tag{7}$$
   - a constraint $\bot$:- $B_1, \ldots, B_n$, its translation $\tau\rho$ is
   $$\neg\tau(B_1 \wedge \cdots \wedge B_n).$$

For any rule R of form (4), τR stands for the conjunction of the elements in the set of formulas that consists of $\tau\rho$ for all instances ρ of R. By definition, each of these instances (and thus the expressions occurring within them) are closed. For any program Π, $\tau\Pi$ is the set of the formulas τR for each R in Π.

Example 2. For instance, $\tau\,(not\ asg(v,C)\ :\ col(C))$ is

$$\{(col(c) \rightarrow \neg asg(v,c)) \mid c \in |I|^{s_p}\}^{\wedge}.$$

Thus, τ applied to rule (1) is

$$\{\neg\,(\tau\,(not\ asg(v,C)\ :\ col(C)) \wedge vtx(v)) \mid v \in |I|^{s_p}\}^{\wedge}.$$

Consider the syntax of the language presented in Section 2, which is a subset of the Abstract Gringo language, whose semantics are defined by the translation τ. The following definition is from Gebser et al. [15].

Definition 4. *We say that a set S of ground atoms is a* gringo answer set *of a program Π if S is a Truszczyński stable model of $\tau\Pi$.*

5 Connecting Semantics

Ultimately, this section shows that the answer sets as introduced in Definition 2 by means of the SM operator coincide with the gringo answer sets as provided by Definition 4. Before that, we review the process of converting first order sentences (typically denoted by variants of F and G) into infinitary formulas originally proposed by

Truszczyński [28]. In addition, we review the concept of strong equivalence in the settings of IPL. Equipped with these notions, we show how, given a program Π, the application of the transformation by Truszczyński on $\tau^*\Pi$ results in an IPL formula that is strongly equivalent to the IPL formula obtained by $\tau\Pi$. This is the key step that helps us to establish our main result: answer sets (Definition 2) and gringo answer sets (Definition 4) coincide.

5.1 Preliminaries: From First Order Sentences to Infinitary Formulas

Truszczyński (2012) provides a definition of stable models for first-order sentences via a grounding procedure, which converts them into infinitary formulas [28, Section 3]. Later, this grounding process was generalized to many-sorted first-order formulas and extensional predicate symbols [7,8]. We review this generalization below. Prior to the review we introduce some necessary notation.

If $\mathscr{I}$ is an interpretation of a signature σ then by $\sigma^{\mathscr{I}}$ we denote the signature obtained from σ by adding, for every element d of a domain $|I|^s$, its *name* d^* as an object constant of sort s. The interpretation $\mathscr{I}$ is extended to $\sigma^{\mathscr{I}}$ by defining $(d^*)^{\mathscr{I}} = d$. The value $t^{\mathscr{I}}$ assigned by an interpretation $\mathscr{I}$ of $\sigma^{\mathscr{I}}$ to a ground term t over $\sigma^{\mathscr{I}}$ and the satisfaction relation between an interpretation of $\sigma^{\mathscr{I}}$ and a sentence over $\sigma^{\mathscr{I}}$ are defined recursively, in the usual way. If $\mathbf{d}$ is a tuple $d_1,\ldots,d_n$ of elements of domains of $\mathscr{I}$ then $\mathbf{d}^*$ stands for the tuple $d_1^*,\ldots,d_n^*$ of their names. If $\mathbf{t}$ is a tuple $t_1,\ldots,t_n$ of ground terms then $\mathbf{t}^{\mathscr{I}}$ stands for the tuple $t_1^{\mathscr{I}},\ldots,t_n^{\mathscr{I}}$ of values assigned to them by $\mathscr{I}$.

For an interpretation $\mathscr{I}$ and a list $\mathbf{p}$ of predicate symbols, by $\mathscr{I}^{\mathbf{p}}$ we denote the set of precomputed atoms $p(t_1,\ldots,t_k)$ satisfied by $\mathscr{I}$ where $p \in \mathbf{p}$. Let $\mathbf{p},\mathbf{q}$ be a partition of the predicate symbols in the signature. Then, the *grounding of a first-order sentence F with respect to an interpretation $\mathscr{I}$ and a set of intensional predicate symbols* $\mathbf{p}$ (and extensional predicate symbols $\mathbf{q}$) is defined as follows:

1. $gr^{\mathbf{p}}_{\mathscr{I}}(\bot) = \bot$;
2. for $p \in \mathbf{p}$, $gr^{\mathbf{p}}_{\mathscr{I}}(p(t_1,\ldots,t_k)) = p((t_1^{\mathscr{I}})^*,\ldots,(t_k^{\mathscr{I}})^*)$;
3. for $p \in \mathbf{q}$, $gr^{\mathbf{p}}_{\mathscr{I}}(p(t_1,\ldots,t_k)) = \top$ if $p((t_1^{\mathscr{I}})^*,\ldots,(t_k^{\mathscr{I}})^*) \in I^{\mathbf{q}}$ and $gr^{\mathbf{p}}_{\mathscr{I}}(p(t_1,\ldots,t_k)) = \bot$ otherwise;
4. $gr^{\mathbf{p}}_{\mathscr{I}}(t_1 = t_2) = \top$ if $t_1^{\mathscr{I}} = t_2^{\mathscr{I}}$ and $\bot$ otherwise;
5. $gr^{\mathbf{p}}_{\mathscr{I}}(F \otimes G) = gr^{\mathbf{p}}_{\mathscr{I}}(F) \otimes gr^{\mathbf{p}}_{\mathscr{I}}(G)$ if $\otimes$ is $\wedge$, $\vee$, or $\to$;
6. $gr^{\mathbf{p}}_{\mathscr{I}}(\exists X\, F(X)) = \{gr^{\mathbf{p}}_{\mathscr{I}}(F(u^*)) \mid u \in |\mathscr{I}|^s\}^{\vee}$ if X is a variable of sort s;
7. $gr^{\mathbf{p}}_{\mathscr{I}}(\forall X\, F(X)) = \{gr^{\mathbf{p}}_{\mathscr{I}}(F(u^*)) \mid u \in |\mathscr{I}|^s\}^{\wedge}$ if X is a variable of sort s.

Recall that $\neg F$ is an abbreviation for $F \to \bot$ so that $gr^{\mathbf{p}}_{\mathscr{I}}(\neg F) = \neg gr^{\mathbf{p}}_{\mathscr{I}}(F)$. For a first order theory Γ, we define $gr^{\mathbf{p}}_{\mathscr{I}}(\Gamma) = \{gr^{\mathbf{p}}_{\mathscr{I}}(F) \mid F \in \Gamma\}^{\wedge}$.

In the sequel, most of the references to the stated definition of grounding are in the context of standard interpretations. Given the conditions 3-5 of the definition of standard interpretations, we can simplify the definition of grounding by restating conditions 2, 3, 6, and 7, as follows

2. for $p \in \mathbf{p}$, $gr^{\mathbf{p}}_{\mathscr{I}}(p(t_1,\ldots,t_k)) = p(t_1^{\mathscr{I}},\ldots,t_k^{\mathscr{I}})$;
3. for $p \in \mathbf{q}$, $gr^{\mathbf{p}}_{\mathscr{I}}(p(t_1,\ldots,t_k)) = \top$ if $p(t_1^{\mathscr{I}},\ldots,t_k^{\mathscr{I}}) \in I^{\mathbf{q}}$ and $gr^{\mathbf{p}}_{\mathscr{I}}(p(t_1,\ldots,t_k)) = \bot$ otherwise;
6. $gr^{\mathbf{p}}_{\mathscr{I}}(\exists X\, F(X)) = \{gr^{\mathbf{p}}_{\mathscr{I}}(F(u)) \mid u \in |\mathscr{I}|^s\}^{\vee}$ if X is a variable of sort s;
7. $gr^{\mathbf{p}}_{\mathscr{I}}(\forall X\, F(X)) = \{gr^{\mathbf{p}}_{\mathscr{I}}(F(u)) \mid u \in |\mathscr{I}|^s\}^{\wedge}$ if X is a variable of sort s.

5.2 Preliminaries: Strong Equivalence in the Infinitary Setting

The Truszczyński stable models of a set of infinitary logic formulas (Definition 3 in Section 4) can be characterized by an extension of equilibrium logic to the infinitary setting [18, Theorem 2]. This allows strong equivalence of infinitary (propositional) formulas to be defined as follows: About sets $\mathscr{P}_1$, $\mathscr{P}_2$ of infinitary formulas we say that they are *strongly equivalent* (denoted as $\mathscr{P}_1 \equiv_s \mathscr{P}_2$) to each other if, for every set $\mathscr{P}$ of infinitary formulas, the sets $\mathscr{P}_1 \cup \mathscr{P}$ and $\mathscr{P}_2 \cup \mathscr{P}$ have the same Truszczyński stable models [18]. About infinitary formulas P and P' we say that they are strongly equivalent if the singleton sets $\{P\}$ and $\{P'\}$ are strongly equivalent. Theorem 3 from that work shows that two sets of infinitary formulas are strongly equivalent if and only if they are equivalent in the infinitary logic of here-and-there. Sometimes, we will abuse the term *strong equivalence* or notation $\equiv_s$ by stating that a set of infinitary formulas is strongly equivalent to an infinitary formula, understanding that in such a case these two entities share the same HT-models.

5.3 Connecting Semantics

In this section, we present the main results of this paper, which relate our proposed semantics – the answer sets of Definition 2 – to the established semantics – the gringo answer sets of Definition 4. Propositions 1-3 are counterparts of Propositions 1-3 from earlier work by Lifschitz et al. [21] within the context of a different language of logic programs. In particular, conditional literals are part of the language considered here. We also allow the absolute value function symbol, and our definition of the values of terms of the form t_1/t_2 and $t_1 \setminus t_2$ differ. It is due to note that the most recent dialect of mini-GRINGO [12] uses a different definition of integer division than the one presented in earlier publications.

Let us start by reviewing some notation. For a program Π, we call a partition $\mathbf{p}, \mathbf{q}$ of predicate symbols from the signature σ_Π (defined in Section 3.1) the *standard partition* if $\mathbf{p}$ contains all predicate symbols from Π and $\mathbf{q}$ contains every comparison symbol. It is easy to see that the standard partition is unique to program Π, and thus can be identified with its first element $\mathbf{p}$ (all predicate symbols but comparisons occurring in Π). For a formula F, variable Z, and term t, by F^Z_t we denote the result of substituting term t for variable Z. For instance, $val_a(Z)^Z_b$ results in formula $b = a$, where b and a are symbolic constants. In the case when substitution is applied to the formula of the form $val_t(Z)$, we often write $val_t(r)$ to denote $val_t(Z)^Z_r$. Intuitively, the formula $val_t(r)$ expresses that r is one of the values of t. This claim is formalized in Proposition 1 below. Note that in the statements of the propositions below, some program Π is assumed implicitly with its corresponding signature σ_Π.

Proposition 1. *Let $\mathscr{I}$ be a standard interpretation, and let $\mathbf{p}$ be the standard partition. Then, for any program variable Z, ground term t, and precomputed term r, the formula $gr^{\mathbf{p}}_{\mathscr{I}}(val_t(Z)^Z_r)$ is strongly equivalent to $\top$ if $r \in [t]$ and to $\bot$ otherwise.*

Recall from Section 2 that in our proposed language, a basic literal or comparison is a special type of conditional literal with an empty list of conditions. Thus, within our extension of the mini-GRINGO language, conditional literals form the main syntactic element. Proposition 1 helps us establish the following result:

Proposition 2. *Let $H : \mathbf{L}$ be a closed conditional literal with local variables $\mathbf{X}$. Then, for a standard interpretation $\mathscr{I}$, the standard partition $\mathbf{p}$, and a tuple of program variables $\mathbf{Z}$,*

$$gr^{\mathbf{p}}_{\mathscr{I}}\left(\tau^{B}_{\mathbf{Z}}(H : \mathbf{L})\right) \equiv_s \tau(H : \mathbf{L}).$$

Using the preceding proposition, we can conclude the following result:

Proposition 3. *For a rule R, standard interpretation $\mathscr{I}$, and the standard partition $\mathbf{p}$,*

$$gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*(R)) \equiv_s \tau R.$$

We can extend Proposition 3 to programs:

Proposition 4. *For a program Π, standard interpretation $\mathscr{I}$, and the standard partition $\mathbf{p}$,*

$$gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*(\Pi)) \equiv_s \tau\Pi.$$

The last proposition is an important result supporting one of the key theorems of this work, namely, Theorem 1. For a (many-sorted) first-order interpretation $\mathscr{I}$ and a set of predicates $\mathbf{p}$, we use $\mathscr{I}^{\mathbf{p}}$ to denote $A(\mathscr{I})$ restricted to atoms whose predicate symbols occur in $\mathbf{p}$.

Theorem 1. *Let Π be a program, and let $\mathbf{p}$ be the standard partition. Then, a set $\mathscr{I}^{\mathbf{p}}$ of precomputed atoms is a Truszczyński stable model of $gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*\Pi)$ iff $\mathscr{I}^{\mathbf{p}}$ is a Truszczyński stable model of $\tau\Pi$.*

Proof. An immediate consequence of Proposition 4 is that, for a program Π and interpretation $\mathscr{I}$, $gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*(\Pi)) \equiv_s \tau\Pi$. Thus, $gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*(\Pi))$ and $\tau\Pi$ have the same HT-models. This implies that they have the same infinitary equilibrium models. Consequently, they have the same Truszczyński stable models due to Theorem 3 by Truszczyński [28].

Now that we have uncovered the connection between the infinitary formulas produced by $gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*\Pi)$ and those produced by $\tau\Pi$, we use the result by Fandinno et al. (2020) reformulated below to connect the Truszczyński stable models of these formulas to the $\mathbf{p}$-stable models of $\tau^*\Pi$.

Proposition 5. *[8, Proposition 2] For any finite two-sorted (target language) theory Γ, an interpretation $\mathscr{I}$, and list of predicate symbols $\mathbf{p}$, $\mathscr{I}$ is $\mathbf{p}$-stable model of Γ if and only if $\mathscr{I}^{\mathbf{p}}$ is a Truszczyński stable model of $gr^{\mathbf{p}}_{\mathscr{I}}(\Gamma)$.*

We are ready to state the main result of this section, connecting our SM-based semantics for programs with the established semantics reviewed in Section 4:

Theorem 2. *Let Π be a program and let $\mathbf{p}$ be the standard partition. Then, $A(\mathscr{I})$ is an answer set of Π iff $A(\mathscr{I})$ is a gringo answer set of Π.*

Proof. $A(\mathscr{I})$ is an answer set of Π
iff $\mathscr{I}$ is a $\mathbf{p}$-stable model of $\tau^*\Pi$ (Definition 2; $\mathbf{p}$ is the standard partition)
iff $\mathscr{I}^{\mathbf{p}}$ is a Truszczyński stable model of $gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*\Pi)$ (Proposition 5)
iff $\mathscr{I}^{\mathbf{p}}$ is a Truszczyński stable model of $\tau\Pi$ (Theorem 1)
iff $\mathscr{I}^{\mathbf{p}}$ is a gringo answer set of Π (Definition 4)
iff $A(\mathscr{I})$ is a gringo answer set of Π ($\mathbf{p}$ is the standard partition).

6 Conclusions and Future Work

In this paper we introduced semantics based on the SM operator for logic programs containing both conditional literals and arithmetic. The key result of this work – Theorem 2 – demonstrates that the definition of answer sets using our extension of the τ^* translation correctly characterizes the behavior of the answer set solver CLINGO.

The main intuition of the ϕ translation [17], which provided SM-based semantics for programs with conditional literals but *without* arithmetic, was that conditional literals in rule bodies behave like nested implications. Our proposed translation preserves this intuition. However, allowing for arithmetic in the language considered here made the argument of the correspondence between our definition of answer sets and gringo answer sets substantially more complex in comparison to the similar argument made for the case of translation ϕ. Specifically, the correspondence now relies on strong equivalence as opposed to syntactic identity.

The newly introduced SM-based semantics enables ASP practitioners to verify programs with conditional literals and arithmetic in the style of past work on modular verification [5,9]. The definition of **p**-answer sets offers greater flexibility (due to the possibility to distinguish intensional and extensional predicate symbols) than the traditional notion of gringo answer sets and supports this verification style. Furthermore, conditional literals can make programs more concise and easier to verify. For example, as illustrated in the Introduction, we can refactor Listing 1.1 to use a smaller set of predicate symbols by employing a conditional literal in constraint (1).

In future work we intend to investigate how the process of using conditional literals to eliminate auxiliary predicates can be generalized. Automated verification is another important direction for future work. Recent progress in this direction is manifested in the ANTHEM[1] system, which employs an automated theorem prover to establish strong [23] or external [11] equivalence of mini-GRINGO programs. We plan to extend the theory and implementation supporting both types of verification to the language presented in this paper. Similar verification tools include CCT [26] and LPEQ [4,19]. Future work will include detailed comparisons of such systems against ANTHEM.

Finally, it is worth investigating ways to simplify the formulas produced by our extension of τ^*. For instance, it is easy to see that the formulas in Example 1 contain several unnecessary existential quantifiers. By applying ht-equivalent simplifications, we obtain a considerably more readable translation:

$$\forall V\,(\forall C(col(C) \to \neg asg(V,C)) \wedge vtx(V) \to \bot)\,.$$

Past work in this vein has been devoted to developing a “natural" translation ν for a broad fragment of the mini-GRINGO language [22]. We plan to extend the ν translation with conditional literals to make (automated) verification of programs easier.

Acknowledgments. We are grateful to Jorge Fandinno and Vladimir Lifschitz for their valuable comments, and to our anonymous reviewers for their feedback.

Disclosure of Interests. The authors have no competing interests.

[1] https://github.com/potassco/anthem

References

1. Abels, D., Jordi, J., Ostrowski, M., Schaub, T., Toletti, A., Wanko, P.: Train scheduling with hybrid answer set programming. Theory and Practice of Logic Programming **21**(3), 317–347 (2021). https://doi.org/10.1017/S1471068420000046
2. Balduccini, M., Gelfond, M.: Model-based reasoning for complex flight systems. In: Proceedings of Infotech@Aerospace (American Institute of Aeronautics and Astronautics) (2005)
3. Balduccini, M., Gelfond, M., Nogueira, M., Watson, R., Barry, M.: An A-Prolog decision support system for the Space Shuttle. In: Working Notes of the AAAI Spring Symposium on Answer Set Programming (2001)
4. Bomanson, J., Janhunen, T., Niemelä, I.: Applying visible strong equivalence in answer-set program transformations. ACM Trans. Comput. Logic **21**(4) (Oct 2020). https://doi.org/10.1145/3412854, https://doi.org/10.1145/3412854
5. Cabalar, P., Fandinno, J., Lierler, Y.: Modular answer set programming as a formal specification language. Theory and Practice of Logic Programming **20**(5), 767–782 (2020)
6. Cabalar, P., Muñiz, B., Pérez, G., Suárez, F.: Explainable machine learning for liver transplantation (2021), https://arxiv.org/abs/2109.13893
7. Fandinno, J., Hansen, Z., Lierler, Y.: Axiomatization of aggregates in answer set programming. In: Proceedings of the Thirty-six National Conference on Artificial Intelligence (AAAI'22). AAAI Press (2022)
8. Fandinno, J., Lifschitz, V., Lühne, P., Schaub, T.: Verifying tight logic programs with anthem and vampire. Theory and Practice of Logic Programming **20**(5), 735–750 (2020)
9. Fandinno, J., Hansen, Z., Lierler, Y.: Arguing correctness of asp programs with aggregates. In: Gottlob, G., Inclezan, D., Maratea, M. (eds.) Logic Programming and Nonmonotonic Reasoning. p. 190–202. Lecture Notes in Computer Science, Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-15707-3_15
10. Fandinno, J., Hansen, Z., Lierler, Y.: Axiomatization of non-recursive aggregates in first-order answer set programming. Journal of Artificial Intelligence Research **80**, 977–1031 (Jul 2024). https://doi.org/10.1613/jair.1.15786
11. Fandinno, J., Hansen, Z., Lierler, Y., Lifschitz, V., Temple, N.: External behavior of a logic program and verification of refactoring. Theory and Practice of Logic Programming **23**(4), 933–947 (2023). https://doi.org/10.1017/S1471068423000200
12. Fandinno, J., Lifschitz, V., Temple, N.: Locally tight programs. Theory and Practice of Logic Programming p. 1–31 (2024). https://doi.org/10.1017/S147106842300039X
13. Ferraris, P., Lee, J., Lifschitz, V.: Stable models and circumscription. Artificial Intelligence **175**(1), 236–263 (2011)
14. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Thiele, S.: A user's guide to `gringo`, `clasp`, `clingo`, and `iclingo`, http://potassco.org
15. Gebser, M., Harrison, A., Kaminski, R., Lifschitz, V., Schaub, T.: Abstract gringo. Theory and Practice of Logic Programming **15**(4–5), 449–463 (Jul 2015). https://doi.org/10.1017/S1471068415000150
16. Gebser, M., Obermeier, P., Otto, T., Schaub, T., Sabuncu, O., Nguyen, V., Son, T.C.: Experimenting with robotic intra-logistics domains. Theory and Practice of Logic Programming **18**(3-4), 502–519 (2018). https://doi.org/10.1017/S1471068418000200
17. Hansen, Z., Lierler, Y.: Semantics for conditional literals via the SM operator. In: Gottlob, G., Inclezan, D., Maratea, M. (eds.) Logic Programming and Nonmonotonic Reasoning. p. 259–272. Lecture Notes in Computer Science, Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-15707-3_20

18. Harrison, A., Lifschitz, V., Pearce, D., Valverde, A.: Infinitary equilibrium logic and strongly equivalent logic programs. Artificial Intelligence **246**, 22–33 (May 2017). https://doi.org/10.1016/j.artint.2017.02.002
19. Janhunen, T., Oikarinen, E.: Lpeq and dlpeq — translators for automated equivalence testing of logic programs. In: Lifschitz, V., Niemelä, I. (eds.) Logic Programming and Nonmonotonic Reasoning. p. 336–340. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24609-1_30
20. Kaminski, R., Romero, J., Schaub, T., Wanko, P.: How to build your own ASP-based system?! Theory and Practice of Logic Programming p. 1–63 (2021). https://doi.org/10.1017/S1471068421000508
21. Lifschitz, V., Lühne, P., Schaub, T.: Verifying strong equivalence of programs in the input language of gringo. In: Proceedings of the 15th International Conference on Logic Programming and Non-monotonic Reasoning (2019), http://www.cs.utexas.edu/users/ai-lab?verification
22. Lifschitz, V.: Transforming gringo rules into formulas in a natural way. In: Logics in Artificial Intelligence: 17th European Conference, JELIA 2021, Virtual Event, May 17–20, 2021, Proceedings. p. 421–434. Springer-Verlag, Berlin, Heidelberg (May 2021). https://doi.org/10.1007/978-3-030-75775-5_28, https://doi.org/10.1007/978-3-030-75775-5_28
23. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM Transactions on Computational Logic **2**(4), 526–541 (Oct 2001). https://doi.org/10.1145/383779.383783
24. Marek, V.W., Truszczyński, M.: Stable Models and an Alternative Logic Programming Paradigm, pp. 375–398. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
25. Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. Annals of Mathematics and Artificial Intelligence **25**(3), 241–273 (Nov 1999). https://doi.org/10.1023/A:1018930122475
26. Oetsch, J., Seidl, M., Tompits, H., Woltran, S.: Testing relativised uniform equivalence under answer-set projection in the system cct. In: Seipel, D., Hanus, M., Wolf, A. (eds.) Applications of Declarative Programming and Knowledge Management. p. 241–246. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00675-3_16
27. Syrjänen, T.: Cardinality constraint programs. In: Alferes, J.J., Leite, J. (eds.) Logics in Artificial Intelligence. p. 187–199. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30227-8_18
28. Truszczyński, M.: Connecting first-order ASP and the logic FO(ID) through reducts. In: Erdem, E., Lee, J., Lierler, Y., Pearce, D. (eds.) Correct Reasoning: Essays on Logic-Based AI in Honour of Vladimir Lifschitz, Lecture Notes in Computer Science, vol. 7265, pp. 543–559. Springer-Verlag (2012)
29. Wotawa, F., Kaufmann, D.: Model-based reasoning using answer set programming. Applied Intelligence **52**(15), 16993–17011 (Dec 2022). https://doi.org/10.1007/s10489-022-03272-2

A Proofs of Theoretical Claims of the Paper (Propositions 1-4)

A.1 Preliminaries: Basic Facts

Fact 1 *As customary, let P denote an infinitary formula, and $\mathscr{P}$ denote a set of infinitary formulas.*

1. $P \wedge \top \equiv_s P$; $\left\{\mathscr{P}, \top\right\}^{\wedge} \equiv_s \mathscr{P}^{\wedge}$.
2. $P \wedge \bot \equiv_s \bot$; $\left\{\mathscr{P}, \bot\right\}^{\wedge} \equiv_s \bot$.

3. $P \vee \top \equiv_s \top$*;* $\left\{\mathscr{P}, \top\right\}^{\vee} \equiv_s \top$.
4. $P \vee \bot \equiv_s P$*;* $\left\{\mathscr{P}, \bot\right\}^{\vee} \equiv_s \mathscr{P}^{\vee}$.
5. $\bot \rightarrow P \equiv_s \top$.
6. $\mathscr{P} \equiv_s \top$ *iff every HT-interpretation* $\langle S, S'\rangle$ *satisfies* $\mathscr{P}$.
7. $\mathscr{P} \equiv_s \bot$ *iff every HT-interpretation* $\langle S, S'\rangle$ *fails to satisfy* $\mathscr{P}$.

Proof. The first four points follow directly.

To see point 5, take an arbitrary HT-interpretation $\langle S, S'\rangle$ and observe that it satisfies $\bot \rightarrow P$. By definition of ht-satisfaction, $\langle S, S'\rangle \models \bot \rightarrow P$ iff $I \models \bot \rightarrow P$ and $\langle S, S'\rangle \not\models \bot$ or $\langle S, S'\rangle \models P$. $I \models \bot \rightarrow P$ follows from the rules of classical satisfaction, and $\langle S, S'\rangle \not\models \bot$ follows from the rules of ht-satisfaction.

To see point 6, assume $\mathscr{P} \equiv_s \top$ and take an arbitrary HT-interpretation $\langle S, S'\rangle$. $\langle S, S'\rangle \models \top$ since $\top$ is an abbreviation for $\emptyset^{\wedge}$. $\mathscr{P}$ and $\top$ have the same HT-models by the definition of strong equivalence, thus $\langle S, S'\rangle \models \mathscr{P}$. Now assume every HT-interpretation $\langle S, S'\rangle$ satisfies $\mathscr{P}$. We know that every $\langle S, S'\rangle \models \top$. Thus, $\mathscr{P}$ and $\top$ have the same HT-models and $\mathscr{P} \equiv_s \top$.

To see point 7, assume $\mathscr{P} \equiv_s \bot$ and take an arbitrary HT-interpretation $\langle S, S'\rangle$. $\langle S, S'\rangle \not\models \bot$ since $\bot$ is an abbreviation for $\emptyset^{\vee}$. $\mathscr{P}$ and $\bot$ have the same HT-models by the definition of strong equivalence, thus $\langle S, S'\rangle \not\models \mathscr{P}$. Now assume every HT-interpretation $\langle S, S'\rangle$ fails to satisfy $\mathscr{P}$. We know that every $\langle S, S'\rangle \not\models \bot$. Thus, $\mathscr{P}$ and $\bot$ have the same HT-models and $\mathscr{P} \equiv_s \bot$.

Fact 2 *Let $P(u,v)$ be an infinitary formula containing precomputed terms u and v, and let S_1 and S_2 denote sets of precomputed terms of the same sort as u and v, respectively. Then, the infinitary formulas*

$$\left\{\{P(u,v) \mid u \in S_1\}^{\vee} \mid v \in S_2\right\}^{\vee} \tag{8}$$

and

$$\left\{P(u,v) \mid \langle u,v\rangle \in S_1 \times S_2\right\}^{\vee} \tag{9}$$

are strongly equivalent. This fact can be generalized in a straightforward way to tuples of more than two terms.

Proof. Recall that the condition of strong equivalence follows if we can establish that (8) is satisfied by an HT-interpretation iff (9) is satisfied by that same HT-interpretation.
Left-to-right: Take any HT-interpretation $\langle S, S'\rangle$ and assume that $\langle S, S'\rangle$ satisfies (8). By the definition of infinitary ht-satisfaction (bullet 3), $\langle S, S'\rangle \models \{P(u,y) \mid u \in S_1\}^{\vee}$ for some y belonging to S_2. Since $\langle S, S'\rangle \models \{P(u,y) \mid u \in S_1\}^{\vee}$, it follows from the definition of infinitary ht-satisfaction (bullet 3) that $\langle S, S'\rangle \models P(x,y)$ for some x belonging to S_1. Since $\langle S, S'\rangle \models P(x,y)$ and the pair $\langle x,y\rangle$ belongs to $S_1 \times S_2$, it follows that $\langle S, S'\rangle$ satisfies (9) (again, by the definition of infinitary ht-satisfaction).

Right-to-left: Take any HT-interpretation $\langle S,S'\rangle$ and assume that $\langle S,S'\rangle$ satisfies (9). It follows that $\langle S,S'\rangle \models P(x,y)$ for some pair $\langle x,y\rangle \in S_1 \times S_2$. Thus, $\langle S,S'\rangle$ satisfies the formula $P(u,y)$ for some u (specifically, $u=x$, $u \in S_1$). It follows that $\langle S,S'\rangle$ also satisfies the formula $\{P(u,y) \mid u \in S_1\}^{\vee}$. Similarly, $\langle S,S'\rangle$ satisfies the formula $\{P(u,v) \mid u \in S_1\}^{\vee}$ for some v (specifically, $v=y$, $v \in S_2$). From this, it follows that $\langle S,S'\rangle$ also satisfies (8).

Fact 3 *Let $\mathscr{I}$ be a standard interpretation, and let* $\mathbf{p},\mathbf{q}$ *be a partition of predicate symbols in our signature (so that every comparison $\prec$ is in* $\mathbf{q}$*), and t_1 and t_2 are ground terms. Then*

- $gr^{\mathbf{p}}_{\mathscr{I}}(t_1 \prec t_2) = \top$ *when relation $\prec$ holds for the pair $(t_1^{\mathscr{I}}, t_2^{\mathscr{I}})$; and*
- $gr^{\mathbf{p}}_{\mathscr{I}}(t_1 \prec t_2) = \bot$*, otherwise.*

Proof. This fact follows from the definitions of grounding (condition 3) and standard interpretations (condition 6).

Fact 4 *1. For any real numbers n and m such that $n<0$ and $m=|n|$, $-n=m$.*
2. For any real numbers i and j, $\frac{|i|}{|j|} = |\frac{i}{j}|$.
3. For any real number n, $\lceil n \rceil = -\lfloor -n \rfloor$.

Proof. The first two points follow directly from the definition of absolute value. For the last point, $\lceil n \rceil = \lceil -(-n) \rceil$, and $\lceil -(-n) \rceil$ is the smallest integer k such that $k \geq -(-n)$. Thus, $k \geq n$ and $-k \leq -n$. Therefore, $-k$ is the largest integer less than or equal to $-n$. By the definition of floor, $\lfloor -n \rfloor = -k$. Since $\lceil -(-n) \rceil$ is k and $-k$ is $\lfloor -n \rfloor$, it follows that $\lceil n \rceil = -\lfloor -n \rfloor$.

Fact 5 *[18, Corollary 1] Within an infinitary propositional formula P, (infinitely many) parts of P can be simultaneously replaced with strongly equivalent formulas without changing the set of stable models of P.*

A.2 Proof of Proposition 1

Lemma 1. *Let $\overline{i},\overline{j},\overline{k}$ be numerals such that k is the integer $\lfloor |i|/|j| \rfloor$. Then,*

- $\overline{k}$ *is* $\overline{round(i/j)}$ *if* $i \times j \geq 0$*;*
- $\overline{-k}$ *is* $\overline{round(i/j)}$ *if* $i \times j < 0$.

Proof. First note that the lemma condition implies that $j \neq 0$.
Assume $i \times j \geq 0$. It follows that $round(i/j) = \lfloor i/j \rfloor$ and $i/j = |i|/|j|$, thus $\lfloor i/j \rfloor = \lfloor |i|/|j| \rfloor$. By the lemma conditions, $k = \lfloor |i|/|j| \rfloor$ and consequently, $k = \lfloor i/j \rfloor = round(i/j)$. It follows that $\overline{k} = \overline{round(i/j)}$.

Assume $i \times j < 0$. It follows that $round(i/j) = \lceil i/j \rceil$ and $\lceil i/j \rceil \leq 0$.
Case 1: Assume $\lceil i/j \rceil < 0$. Thus, $|j| \leq |i|$. Denote $\frac{i}{j}$ by n, and denote $\frac{|i|}{|j|}$ by m. It follows from points 1 and 2 of Fact 4 that $-n = m$. Furthermore, it follows from point 3 of Fact 4 that $\lceil n \rceil = -\lfloor -n \rfloor$, and therefore $\lceil n \rceil = -\lfloor m \rfloor$. Thus, $\lceil i/j \rceil = -\lfloor |i|/|j| \rfloor = -k$

(recall that k is $\lfloor |i|/|j| \rfloor$), which entails that $\overline{-k}$ is $\overline{\lceil i/j \rceil}$.
Case 2: Assume $\lceil i/j \rceil = 0$. Thus, $|j| > |i|$, and $0 \le |i|/|j| < 1$. Consequently, $\lfloor |i|/|j| \rfloor = 0$, recall that k is $\lfloor |i|/|j| \rfloor$. It follows that $k = 0$, and thus $-k = 0$. Consequently, $\overline{-k}$ is $\overline{\lceil i/j \rceil}$.

Lemma 2. *Let $\mathscr{I}$ be a standard interpretation and let $F_1(IJK)$ be the formula*

$$K \times |J| \le |I| < (K + \overline{1}) \times |J|$$

Then $gr^{\mathbf{p}}_{\mathscr{I}}(F_1(\overline{i}\overline{j}\overline{k})) \equiv_s \top$ if $\overline{i}, \overline{j}, \overline{k}$ are numerals such that integer $k = \lfloor |i|/|j| \rfloor$ and $gr^{\mathbf{p}}_{\mathscr{I}}(F_1(\overline{i}\overline{j}\overline{k})) \equiv_s \bot$ otherwise.

Proof. Recall that $F_1(IJK)$ is an abbreviation for the formula

$$(K \times |J| \le |I|) \wedge (|I| < (K + \overline{1}) \times |J|)$$

thus,

$$gr^{\mathbf{p}}_{\mathscr{I}}(F_1(\overline{i}\overline{j}\overline{k})) = gr^{\mathbf{p}}_{\mathscr{I}}(\overline{k} \times |\overline{j}| \le |\overline{i}|)) \wedge gr^{\mathbf{p}}_{\mathscr{I}}(|\overline{i}| < (\overline{k} + \overline{1}) \times |\overline{j}|)$$

Let $\overline{n_1}$ denote the domain element $(\overline{k} \times |\overline{j}|)^{\mathscr{I}}$, and let $\overline{n_2}$ denote the domain element $(|\overline{i}|)^{\mathscr{I}}$. By the definition of grounding, $gr^{\mathbf{p}}_{\mathscr{I}}(\overline{k} \times |\overline{j}| \le |\overline{i}|)) \equiv_s \top$ if the relation $\le$ holds between $\overline{n_1}$ and $\overline{n_2}$, and $gr^{\mathbf{p}}_{\mathscr{I}}(\overline{k} \times |\overline{j}| \le |\overline{i}|)) \equiv_s \bot$ otherwise. Since $\mathscr{I}$ is a standard interpretation, $\overline{n_1}$ and $\overline{n_2}$ are the numerals $\overline{k \times |j|}$ and $\overline{|i|}$, respectively.

Similarly, let $\overline{n_3}$ denote the domain element $((\overline{k} + \overline{1}) \times |\overline{j}|)^{\mathscr{I}}$. By the definition of grounding, $gr^{\mathbf{p}}_{\mathscr{I}}(|\overline{i}| < (\overline{k} + \overline{1}) \times |\overline{j}|) \equiv_s \top$ if the relation $<$ holds between $\overline{n_2}$ and $\overline{n_3}$, and $gr^{\mathbf{p}}_{\mathscr{I}}(|i| < (k + \overline{1}) \times |j|) \equiv_s \bot$ otherwise. Since $\mathscr{I}$ is a standard interpretation, $\overline{n_2}$ and $\overline{n_3}$ are the numerals $\overline{|i|}$ and $\overline{(k+1) \times |j|}$, respectively.

Since the total order on numerals mirrors the total order on integers, this means that we need to establish that $n_1 \le n_2 < n_3$ for integers n_1, n_2, and n_3. We proceed by contradiction and assume that either $n_1 > n_2$ or $n_2 \ge n_3$.

If $n_1 > n_2$, then $k \times |j| > |i|$. Thus, $k > |i|/|j| \ge \lfloor |i|/|j| \rfloor$. But this contradicts the lemma's condition that k is $\lfloor |i|/|j| \rfloor$.

If $n_2 > n_3$, then $|i| > (k+1) \times |j|$. Thus, $k + 1 < |i|/|j|$. Let $l = |i|/|j| - \lfloor |i|/|j| \rfloor$. Note how $l < 1$ and we can represent $|i|/|j| = \lfloor |i|/|j| \rfloor + l$. It is easy to see that $k + 1 < \lfloor |i|/|j| \rfloor + l$ and consequently, $k < \lfloor |i|/|j| \rfloor + (l - 1)$. Since $l - 1 < 0$, $k < \lfloor |i|/|j| \rfloor$. This again contradicts the lemma's condition that $k = \lfloor |i|/|j| \rfloor$.

Finally, if $n_2 = n_3$, then $|i| = (k+1) \times |j|$ and consequently, $|i|/|j| = (k+1)$. Thus, $|i|/|j| - 1 = k$. Recall that $|i|/|j| = \lfloor |i|/|j| \rfloor + l$, where l is strictly less than 1. Consequently, $k = \lfloor |i|/|j| \rfloor + (l - 1)$, which contradicts that $k = \lfloor |i|/|j| \rfloor$.

Lemma 3. *Let $\mathscr{I}$ be a standard interpretation and let $F_1(IJK)$ be the formula*

$$K \times |J| \le |I| < (K + \overline{1}) \times |J|$$

If $\overline{i}$ and $\overline{k}$ are any numerals and $\overline{j}$ is $\overline{0}$, then $gr^{\mathbf{p}}_{\mathscr{I}}(F_1(\overline{i}\overline{j}\overline{k})) \equiv_s \bot$.

Proof. Recall that

$$gr^{\mathbf{p}}_{\mathscr{I}}(F_1(\overline{i}\overline{j}\overline{k})) = gr^{\mathbf{p}}_{\mathscr{I}}(\overline{k} \times |\overline{j}| \leq |\overline{i}|)) \wedge gr^{\mathbf{p}}_{\mathscr{I}}(|\overline{i}| < (\overline{k}+\overline{1}) \times |\overline{j}|)$$

By the definition of grounding, $gr^{\mathbf{p}}_{\mathscr{I}}(|\overline{i}| < (\overline{k}+\overline{1}) \times |\overline{j}|) \equiv_s \bot$ if the relation $<$ does not hold between $(|\overline{i}|)^{\mathscr{I}}$ and $((\overline{k}+\overline{1}) \times |\overline{j}|)^{\mathscr{I}}$ which is the case when $|i| \geq (k+1) \times |j|$. Given that $\overline{j} = \overline{0}$, $(k+1) \times |j| = 0$. For any integer i, $|i| \geq 0$ due to the definition of an absolute value.

Lemma 4. *Let $\mathscr{I}$ be a standard interpretation, let r be a precomputed term, and let $F_2(IJKZ)$ be the formula*

$$(I \times J \geq \overline{0} \wedge Z = K) \vee (I \times J < \overline{0} \wedge Z = -K)$$

Further, let $\overline{i}, \overline{j}, \overline{k}$ be numerals such that k is $\lfloor |i|/|j| \rfloor$. Then $gr^{\mathbf{p}}_{\mathscr{I}}(F_2(\overline{i}\overline{j}\overline{k}r)) \equiv_s \top$ if r is the numeral $\overline{round(i/j)}$ and $gr^{\mathbf{p}}_{\mathscr{I}}(F_2(\overline{i}\overline{j}\overline{k}r)) \equiv_s \bot$ otherwise.

Proof.

$$\begin{aligned} gr^{\mathbf{p}}_{\mathscr{I}}(F_2(\overline{i}\overline{j}\overline{k}r)) &= gr^{\mathbf{p}}_{\mathscr{I}}\big((\overline{i} \times \overline{j} \geq \overline{0} \wedge r = \overline{k}) \vee (\overline{i} \times \overline{j} < \overline{0} \wedge r = \overline{-k})\big) \\ &= gr^{\mathbf{p}}_{\mathscr{I}}(\overline{i} \times \overline{j} \geq \overline{0} \wedge r = \overline{k}) \vee gr(\overline{i} \times \overline{j} < \overline{0} \wedge r = \overline{-k}) \\ &= \big(gr^{\mathbf{p}}_{\mathscr{I}}(\overline{i} \times \overline{j} \geq \overline{0}) \wedge gr^{\mathbf{p}}_{\mathscr{I}}(r = \overline{k})\big) \vee \big(gr^{\mathbf{p}}_{\mathscr{I}}(\overline{i} \times \overline{j} < \overline{0}) \wedge gr^{\mathbf{p}}_{\mathscr{I}}(r = \overline{-k})\big) \end{aligned}$$

Case 1. Assume r is $\overline{round(i/j)}$. We need to show that $gr^{\mathbf{p}}_{\mathscr{I}}(F_2(\overline{i}\overline{j}\overline{k}r)) \equiv_s \top$.
Case 1.1 Assume $i \times j \geq 0$. By the definition of grounding,

$$gr^{\mathbf{p}}_{\mathscr{I}}(F_2(\overline{i}\overline{j}\overline{k}r)) = \big(\top \wedge gr^{\mathbf{p}}_{\mathscr{I}}(r = \overline{k})\big) \vee \big(\bot \vee gr^{\mathbf{p}}_{\mathscr{I}}(r = \overline{-k})\big)$$

which entails (by Fact 1) that $gr^{\mathbf{p}}_{\mathscr{I}}(F_2(\overline{i}\overline{j}\overline{k}r)) \equiv_s gr^{\mathbf{p}}_{\mathscr{I}}(r = \overline{k})$. $gr^{\mathbf{p}}_{\mathscr{I}}(r = \overline{k})$ is $\top$ iff $r^{\mathscr{I}} = \overline{k}^{\mathscr{I}}$ iff $\overline{round(i/j)}^{\mathscr{I}} = \overline{k}^{\mathscr{I}}$. This last condition follows immediately from Lemma 1.
Case 1.2 Assume $i \times j < 0$. By the definition of grounding,

$$gr^{\mathbf{p}}_{\mathscr{I}}(F_2(\overline{i}\overline{j}\overline{k}r)) = \big(\bot \wedge gr^{\mathbf{p}}_{\mathscr{I}}(r = \overline{k})\big) \vee \big(\top \vee gr^{\mathbf{p}}_{\mathscr{I}}(r = \overline{-k})\big)$$

which entails (by Fact 1) that $gr^{\mathbf{p}}_{\mathscr{I}}(F_2(\overline{i}\overline{j}\overline{k}r)) \equiv_s gr^{\mathbf{p}}_{\mathscr{I}}(r = \overline{-k})$. $gr^{\mathbf{p}}_{\mathscr{I}}(r = \overline{-k})$ is $\top$ iff $r^{\mathscr{I}} = \overline{-k}^{\mathscr{I}}$ iff $\overline{round(i/j)}^{\mathscr{I}} = \overline{-k}^{\mathscr{I}}$. This last condition follows immediately from Lemma 1.

Case 2. Assume r is not $\overline{round(i/j)}$. We need to show that $gr(F_2(\overline{i}\overline{j}\overline{k}r)) \equiv_s \bot$.
Case 2.1 Assume $i \times j \geq 0$. Lemma 1 implies that $\overline{k}$ is $\overline{round(i/j)}$. As in Case 1.1, we derive that $gr^{\mathbf{p}}_{\mathscr{I}}(F_2(\overline{i}\overline{j}\overline{k}r)) \equiv_s gr^{\mathbf{p}}_{\mathscr{I}}(r = \overline{k})$. Thus, $gr^{\mathbf{p}}_{\mathscr{I}}(r = \overline{k})$ is $\bot$ iff $r^{\mathscr{I}} \neq \overline{k}^{\mathscr{I}}$. As r is not $\overline{round(i/j)}$, $r^{\mathscr{I}} \neq \overline{round(i/j)}$. It follows that $r^{\mathscr{I}} \neq \overline{k}^{\mathscr{I}}$.
Case 2.2 Assume $i \times j < 0$. Lemma 1 implies that $\overline{-k}$ is $\overline{round(i/j)}$. As in Case 1.2, we derive that $gr^{\mathbf{p}}_{\mathscr{I}}(F_2(\overline{i}\overline{j}\overline{k}r)) \equiv_s gr^{\mathbf{p}}_{\mathscr{I}}(r = \overline{-k})$. Thus, $gr^{\mathbf{p}}_{\mathscr{I}}(r = \overline{-k})$ is $\bot$ iff $r^{\mathscr{I}} \neq \overline{-k}^{\mathscr{I}}$ or, in other words, $r^{\mathscr{I}} \neq \overline{-k}$. Just as in Case 1.2, r is not $\overline{round(i/j)}$, but $\overline{-k}$ is.

Lemma 5. *Let $\mathscr{I}$ be a standard interpretation, let r be a precomputed term, and let $F_3(IJKZ)$ be the formula*

$$(I \times J \geq \overline{0} \wedge Z = I - K \times J) \vee (I \times J \leq \overline{0} \wedge Z = I + K \times J)$$

Further, let $\overline{i}, \overline{j}, \overline{k}$ be numerals such that k is $\lfloor |i|/|j| \rfloor$. Then $gr^{\mathbf{p}}_{\mathscr{I}}(F_3(\overline{i}\overline{j}\overline{k}r)) \equiv_s \top$ if r is the numeral $\overline{i - j \times round(i/j)}$ and $gr^{\mathbf{p}}_{\mathscr{I}}(F_3(\overline{i}\overline{j}\overline{k}r)) \equiv_s \bot$ otherwise.

Proof.

$$\begin{aligned} gr^{\mathbf{p}}_{\mathscr{I}}(F_3(\overline{i}\overline{j}\overline{k}r)) &= gr^{\mathbf{p}}_{\mathscr{I}}\big((\overline{i} \times \overline{j} \geq \overline{0} \wedge r = \overline{i} - \overline{k} \times \overline{j}) \vee (\overline{i} \times \overline{j} < \overline{0} \wedge r = \overline{i} + \overline{k} \times \overline{j})\big) \\ &= \big(gr^{\mathbf{p}}_{\mathscr{I}}(\overline{i} \times \overline{j} \geq \overline{0}) \wedge gr^{\mathbf{p}}_{\mathscr{I}}(r = \overline{i} - \overline{k} \times \overline{j})\big) \vee \big(gr^{\mathbf{p}}_{\mathscr{I}}(\overline{i} \times \overline{j} < \overline{0}) \wedge gr^{\mathbf{p}}_{\mathscr{I}}(r = \overline{i} + \overline{k} \times \overline{j})\big) \end{aligned}$$

Case 1. Assume $i \times j \geq 0$. It follows that $gr^{\mathbf{p}}_{\mathscr{I}}(F_3(\overline{i}\overline{j}\overline{k}r)) \equiv_s gr^{\mathbf{p}}_{\mathscr{I}}(r = \overline{i} - \overline{k} \times \overline{j})$. From Lemma 1 it follows that $\overline{k}$ is $\overline{round(i/j)}$. Thus, by the definition of grounding it follows that $gr^{\mathbf{p}}_{\mathscr{I}}(F_3(\overline{i}\overline{j}\overline{k}r)) \equiv_s \top$ if $r^{\mathscr{I}} = (\overline{i} - \overline{round(i/j)} \times \overline{j})^{\mathscr{I}}$, and $gr^{\mathbf{p}}_{\mathscr{I}}(F_3(\overline{i}\overline{j}\overline{k}r)) \equiv_s \bot$ otherwise. Consequently, $gr^{\mathbf{p}}_{\mathscr{I}}(F_3(\overline{i}\overline{j}\overline{k}r)) \equiv_s \top$ if r is the numeral $\overline{i - j \times round(i/j)}$ and $gr^{\mathbf{p}}_{\mathscr{I}}(F_3(\overline{i}\overline{j}\overline{k}r)) \equiv_s \bot$ otherwise.

Case 2. Assume $i \times j < 0$. It follows that $gr^{\mathbf{p}}_{\mathscr{I}}(F_3(\overline{i}\overline{j}\overline{k}r)) \equiv_s gr^{\mathbf{p}}_{\mathscr{I}}(r = \overline{i} + \overline{k} \times \overline{j})$. From Lemma 1 it follows that $\overline{-k}$ is $\overline{round(i/j)}$. Thus, $\overline{-k}^{\mathscr{I}} = \overline{round(i/j)}^{\mathscr{I}}$. By properties over numerals, $\overline{k} = \overline{-round(i/j)}$. Therefore, it follows from the definition of grounding that $gr^{\mathbf{p}}_{\mathscr{I}}(F_3(\overline{i}\overline{j}\overline{k}r)) \equiv_s \top$ if $r^{\mathscr{I}} = (\overline{i} + \overline{-round(i/j)} \times \overline{j})^{\mathscr{I}}$ and $gr^{\mathbf{p}}_{\mathscr{I}}(F_3(\overline{i}\overline{j}\overline{k}r)) \equiv_s \bot$ otherwise. Consequently, $gr^{\mathbf{p}}_{\mathscr{I}}(F_3(\overline{i}\overline{j}\overline{k}r)) \equiv_s \top$ if r is the numeral $\overline{i - j \times round(i/j)}$ and $gr^{\mathbf{p}}_{\mathscr{I}}(F_3(\overline{i}\overline{j}\overline{k}r)) \equiv_s \bot$ otherwise.

Proof of Proposition 1

Proof. We proceed by structural induction across the distinct cases of forms of terms. For the duration of this proof, we write gr to denote $gr^{\mathbf{p}}_{\mathscr{I}}$. Furthermore, we do not distinguish numerals from integers in this proof using these terms interchangeably, due to the fact that numerals and integers are in one to one correspondence. Thus, we will abuse the notation and drop the overline symbol denoting a numeral.

Case 1: Term t is a numeral, symbolic constant, *inf*, or *sup*. Then, $val_t(r)$ is $r = t$ and $[t] = \{t\}$. By definition, $gr(r = t) = \top$ if $r^{\mathscr{I}} = t^{\mathscr{I}}$ and $\bot$ otherwise. Given that I is a standard interpretation, $r^I = t^I$ if and only if $r \in [t]$.

In the remainder of the proof, we say that a term t has the induction hypothesis property when the formula $gr(val_t(r))$ is strongly equivalent to $\top$ if $r \in [t]$ and to $\bot$ otherwise.
Case 2: Term t is $|t_1|$, where t_1 has the induction hypothesis property. By definition,

$val_t(r)$ is $\exists I(val_{t_1}(I) \wedge r = |I|))$. Then,

$$\begin{aligned} gr(val_t(r)) &= gr(\exists I(val_{t_1}(I) \wedge r = |I|)) \\ &= \{gr(val_{t_1}(i)) \wedge gr(r = |i|) \mid i \in |\mathscr{I}|^{s_i}\}^{\vee} \\ &= \big\{(gr(val_{t_1}(j)) \wedge gr(r = |j|)), \\ &\quad (gr(val_{t_1}(k)) \wedge gr(r = |k|)) \mid j,k \in |\mathscr{I}|^{s_i}, j \in [t_1], k \notin [t_1]\big\}^{\vee} \end{aligned}$$

By the fact that t_1 has the inductive property, it follows that

$$\begin{aligned} gr(val_t(r)) \equiv_s \big\{&(\top \wedge gr(r = |j|)), \\ &(\bot \wedge gr(r = |k|)) \mid j,k \in |\mathscr{I}|^{s_i}, j \in [t_1], k \notin [t_1]\big\}^{\vee} \end{aligned}$$

In turn, this formula is strongly equivalent the following formula, by Fact 1 (conditions 1, 2) and Fact 5).

$$\big\{gr(r = |j|) \mid j \in |\mathscr{I}|^{s_i}, j \in [t_1]\big\}^{\vee}. \tag{10}$$

Case 2.1: $r \in [t]$, where $t = |t_1|$.

Our goal is to show that (10) is strongly equivalent to $\top$, which holds
iff, for an arbitrary HT-interpretation$\langle S,S'\rangle$, $\langle S,S'\rangle \models$ (10) (Fact 1, point 3)
iff, for an arbitrary HT-interpretation$\langle S,S'\rangle$,
$\langle S,S'\rangle \models gr(r = |j|)$ for some $j \in |\mathscr{I}|^{s_i}, j \in [t_1]$
iff, $gr(r = |j|) = \top$ for some $j \in |\mathscr{I}|^{s_i}, j \in [t_1]$
iff, for some numeral $j \in [t_1]$, $r^{\mathscr{I}} = |j|^{\mathscr{I}}$ (Definition of grounding)
iff, for some numeral $j \in [t_1]$, $r = |j|$ (Conditions 3 and 5 of standard interpretations)

Now, by our assumption that $r \in [t]$ it follows that $r \in \{|j| \mid j \in [t_1]\}$. Hence, there exists a $j \in [t_1]$ such that $r = |j|$, and our goal is satisfied.

Case 2.2: $r \notin [t]$, where $t = |t_1|$.

Our goal is to show that (10) is strongly equivalent to $\bot$, which holds
iff, for an arbitrary HT-interpretation$\langle S,S'\rangle$, $\langle S,S'\rangle \not\models$ (10) (Fact 1, point 4)
iff, for an arbitrary HT-interpretation$\langle S,S'\rangle$, $\langle S,S'\rangle \not\models H$ for any H in (10) (Definition)
iff, for an arbitrary HT-interpretation$\langle S,S'\rangle$, $\langle S,S'\rangle \not\models gr(r = |j|)$ for any $j \in |\mathscr{I}|^{s_i}, j \in [t_1]$
iff, $gr(r = |j|) = \bot$ for any $j \in |\mathscr{I}|^{s_i}, j \in [t_1]$
iff, for any numeral $j \in [t_1]$, $r^{\mathscr{I}} \neq |j|^{\mathscr{I}}$ (Definition of grounding)
iff, for any numeral $j \in [t_1]$, $r \neq |j|$ (Conditions 3 and 5 of standard interpretations)

Now, by our assumption that $r \notin [t]$ it follows that $r \notin \{|j| \mid j \in [t_1]\}$. Consequently, for any j in $[t_1]$, $r \neq |j|$.

Case 3: Term t is $t_1 \circ t_2$, where $\circ$ is one of the operation names $+$, $-$, $\times$, and t_1 and t_2 have the induction hypothesis property. Then, $val_t(r)$ is $\exists IJ(r = I \circ J \wedge val_{t_1}(I) \wedge$

$val_{t_2}(J))$, and

$$\begin{aligned}
gr(val_t(r)) &= gr(\exists IJ(r = I \circ J \wedge val_{t_1}(I) \wedge val_{t_2}(J))) \\
&= \{gr(\exists J(r = i \circ J \wedge val_{t_1}(i) \wedge val_{t_2}(J))) \mid i \in |\mathscr{I}|^{s_i}\}^{\vee} \\
&= \big\{\{gr(r = i \circ j) \wedge gr(val_{t_1}(i)) \wedge gr(val_{t_2}(j)) \mid i \in |\mathscr{I}|^{s_i}\}^{\vee} \mid j \in |\mathscr{I}|^{s_i}\big\}^{\vee} \\
&= \big\{\{(gr(r = i_1 \circ j) \wedge gr(val_{t_1}(i_1)) \wedge gr(val_{t_2}(j))), \\
&\qquad (gr(r = i_2 \circ j) \wedge gr(val_{t_1}(i_2)) \wedge gr(val_{t_2}(j))) \mid i_1, i_2 \in |\mathscr{I}|^{s_i}, i_1 \in [t_1], i_2 \notin [t_1]\}^{\vee} \mid j \in |\mathscr{I}|^{s_i}\big\}^{\vee}
\end{aligned}$$

By the fact that t_1 has the inductive property, Fact 1 (conditions 1, 2), and Fact 5, it follows that

$$gr(val_t(r)) \equiv_s \big\{\{gr(r = i_1 \circ j) \wedge gr(val_{t_2}(j)) \mid i_1 \in |\mathscr{I}|^{s_i}, i_1 \in [t_1]\}^{\vee} \mid j \in |\mathscr{I}|^{s_i}\big\}^{\vee}$$

In turn, this formula can be rewritten as

$$\Big\{\big\{\big(gr(r = i_1 \circ j_1) \wedge gr(val_{t_2}(j_1))\big), \big(gr(r = i_1 \circ j_2) \wedge gr(val_{t_2}(j_2))\big) \mid i_1 \in |\mathscr{I}|^{s_i}, i_1 \in [t_1]\big\}^{\vee} \\ \Big| \; j_1, j_2 \in |\mathscr{I}|^{s_i}, j_1 \in [t_2], j_2 \notin [t_2]\Big\}^{\vee}$$

By the fact that t_2 has the inductive property, Fact 1 (conditions 1, 2), and Fact 5, it follows that

$$gr(val_t(r)) \equiv_s \big\{\{gr(r = i_1 \circ j_1) \mid i_1 \in |\mathscr{I}|^{s_i}, i_1 \in [t_1]\}^{\vee} \mid j_1 \in |\mathscr{I}|^{s_i}, j_1 \in [t_2]\big\}^{\vee}$$

By Fact 2, $gr(val_t(r))$ is strongly equivalent to (where we rename i_1 by i and j_1 by j for readability)

$$\{gr(r = i \circ j) \mid i, j \in |\mathscr{I}|^{s_i}, i \in [t_1], j \in [t_2]\}^{\vee} \tag{11}$$

Case 3.1: $r \in [t]$, where $t = t_1 \circ t_2$.

Our goal is to show that (11) is strongly equivalent to $\top$, which holds
iff, for an arbitrary HT-interpretation$\langle S, S' \rangle$, $\langle S, S' \rangle \models gr(r = i \circ j)$ for some $i, j \in |\mathscr{I}|^{s_i}$, $i \in [t_1], j \in [t_2]$
iff, for some pair of numerals i, j such that $i \in [t_1], j \in [t_2]$, $gr(r = i \circ j) = \top$
iff, for some pair of numerals i, j such that $i \in [t_1], j \in [t_2]$, $r^{\mathscr{I}} = (i \circ j)^{\mathscr{I}}$ (Definition of grounding and satisfaction)
iff, for some pair of numerals i, j such that $i \in [t_1], j \in [t_2]$, $r = i \circ j$ (Conditions 3 and 4 of standard interpretations)
Now, by our assumption that $r \in [t]$ it follows that $r \in \{i \circ j \mid i \in [t_1], j \in [t_2]\}$. Hence, there exists $i \in [t_1]$ and $j \in [t_2]$ such that $r = i \circ j$, and the goal immediately follows.
Case 3.2: $r \notin [t]$, where $t = t_1 \circ t_2$. Our goal is to show that (11) is strongly equivalent to $\bot$, which holds
iff, for an arbitrary HT-interpretation$\langle S, S' \rangle$, $\langle S, S' \rangle \not\models gr(r = i \circ j)$ for any $i, j \in |\mathscr{I}|^{s_i}$, $i \in [t_1], j \in [t_2]$
iff, for any pair of numerals i, j such that $i \in [t_1], j \in [t_2]$, $gr(r = i \circ j) = \bot$

iff, for any pair of numerals i, j such that $i \in [t_1], j \in [t_2]$, $r^{\mathscr{I}} \neq (i \circ j)^{\mathscr{I}}$ (Definition of grounding and satisfaction)
iff, for any pair of numerals i, j such that $i \in [t_1], j \in [t_2]$, $r \neq i \circ j$ (Conditions 3 and 4 of standard interpretations)
Now, by our assumption that $r \notin [t]$ it follows that $r \notin \{i \circ j \mid i \in [t_1], j \in [t_2]\}$. Consequently, for any $i \in [t_1]$ and $j \in [t_2]$ it follows that $r \neq i \circ j$.

Case 4: Term t is $(t_1..t_2)$ where t_1 and t_2 have the induction hypothesis property. Then, $val_t(r)$ is $\exists IJK(r = K \wedge I \leq K \wedge K \leq J \wedge val_{t_1}(I) \wedge val_{t_2}(J))$, and

$$\begin{aligned}
&gr(val_t(r)) = \\
&\quad \big\{\big\{\{gr(r=k) \wedge gr(i \leq k) \wedge gr(k \leq j) \wedge gr(val_{t_1}(i)) \wedge gr(val_{t_2}(j)) \mid k \in |\mathscr{I}|^{s_i}\}^{\vee} \\
&\qquad \mid j \in |\mathscr{I}|^{s_i}\big\}^{\vee} \\
&\qquad\quad \mid i \in |\mathscr{I}|^{s_i}\big\}^{\vee}
\end{aligned}$$

Using the same style of transformations and line of reasoning as in the prior cases and the fact that t_1 and t_2 have the inductive property, we are able to show that $gr(val_t(r))$ is strongly equivalent to

$$\big\{gr(r=k) \wedge gr(i \leq k) \wedge gr(k \leq j) \;\big|\; i,j,k \in |\mathscr{I}|^{s_i}, i \in [t_1], j \in [t_2]\big\}^{\vee}$$

Every element in this infinitary formula belongs to one of the following sets

$$\{gr(r=k) \wedge gr(i \leq k) \wedge gr(k \leq j) \mid i,j,k \in |\mathscr{I}|^{s_i}, i \in [t_1], j \in [t_2], i \leq k \leq j\} \tag{12}$$

and

$$\{gr(r=k) \wedge gr(i \leq k) \wedge gr(k \leq j) \mid i,j,k \in |\mathscr{I}|^{s_i}, i \in [t_1], j \in [t_2], \neg(i \leq k \leq j)\} \tag{13}$$

By Fact 3 and the definition of HT-satisfaction,

$$gr(i \leq k) \wedge gr(k \leq j) \equiv_s \top$$

iff $i \leq k \leq j$. Similarly,

$$gr(i \leq k) \wedge gr(k \leq j) \equiv_s \bot$$

iff $\neg(i \leq k \leq j)$.

Thus, every formula in (12) is strongly equivalent to $gr(r=k)$, and every formula in (13) is strongly equivalent to $\bot$. It follows that

$$gr(val_t(r)) \equiv_s \{gr(r=k) \mid i,j,k \in |\mathscr{I}|^{s_i}, i \in [t_1], j \in [t_2], i \leq k \leq j\}^{\vee} \tag{14}$$

The following chain of reasoning mirrors the proof by cases ($r \in [t]$, $r \notin [t]$) detailed earlier. First note that by the definition of the values of t (t is $t_1..t_2$), $r \in [t]$ if r belongs to the set of numerals

$$\{m \mid n_1 \in [t_1], n_2 \in [t_2], n_1 \leq m \leq n_2\} \tag{15}$$

and $r \notin [t]$ if r does not belong to (15).

It follows from (14) that $gr(val_t(r)) \equiv_s \top$ when $gr(r = k) = \top$ for some triple of numerals i, j, k such that $i \in [t_1], j \in [t_2]$, $i \leq k \leq j$, and $gr(val_t(r)) \equiv_s \bot$ otherwise. The condition $gr(r = k) = \top$ for some triple of numerals i, j, k such that $i \in [t_1], j \in [t_2]$, $i \leq k \leq j$ holds when $r^{\mathscr{I}} = k^{\mathscr{I}}$. Therefore, $gr(val_t(r)) \equiv_s \top$ when r belongs to (15) (take k to be r) and $gr(val_t(r)) \equiv_s \bot$, otherwise. Consequently, $gr(val_t(r)) \equiv_s \top$ if $r \in [t]$, and $gr(val_t(r)) \equiv_s \bot$ otherwise.

Case 5: Term t is t_1/t_2, where t_1 and t_2 have the induction hypothesis property. Then, $val_t(r)$ is $\exists IJK(val_{t_1}(I) \wedge val_{t_2}(J) \wedge F_1(IJK) \wedge F_2(IJKr)$, and

$$\begin{aligned}
&gr(val_t(r)) = \\
&\quad \Big\{\Big\{\{gr(val_{t_1}(i)) \wedge gr(val_{t_2}(j)) \wedge gr(F_1(ijk)) \wedge gr(F_2(ijkr)) \bigm| k \in |\mathscr{I}|^{s_i}\}^{\vee} \\
&\qquad \bigm| j \in |\mathscr{I}|^{s_i}\Big\}^{\vee} \\
&\qquad\quad \bigm| i \in |\mathscr{I}|^{s_i}\Big\}^{\vee}
\end{aligned}$$

By Fact 2,

$$gr(val_t(r)) \equiv_s \big\{gr(val_{t_1}(i)) \wedge gr(val_{t_2}(j)) \wedge gr(F_1(ijk)) \wedge gr(F_2(ijkr)) \bigm| i, j, k \in |\mathscr{I}|^{s_i}\big\}^{\vee}$$

Since t_1 and t_2 have the inductive property, we can use analogous reasoning to previous cases to conclude that

$$gr(val_t(r)) \equiv_s \big\{gr(F_1(ijk)) \wedge gr(F_2(ijkr)) \bigm| i, j, k \in |\mathscr{I}|^{s_i}, i \in [t_1], j \in [t_2]\big\}^{\vee} \tag{16}$$

We now illustrate that $gr(val_t(r)) \equiv_s \top$, when $r \in [t]$.

1. It follows from (16) that $gr(val_t(r)) \equiv_s \top$ iff $gr(F_1(ijk)) \equiv_s \top$ and $gr(F_2(ijkr)) \equiv_s \top$ for some triple i, j, k of numerals such that $i \in [t_1]$ and $j \in [t_2]$.

2. From Lemma 2 it follows that $gr(F_1(ijk)) \equiv_s \top$ when $k = \lfloor |i|/|j| \rfloor$.

3. By 1 and 2, $gr(val_t(r)) \equiv_s \top$ if $gr(F_2(ijkr)) \equiv_s \top$ for some i, j, k triple of numerals such that $i \in [t_1]$, $j \in [t_2]$, and $k = \lfloor |i|/|j| \rfloor$.

4. From Lemma 4 it follows that $gr(F_2(ijkr)) \equiv_s \top$ when r is the numeral $round(i/j)$ for some triple i, j, k of numerals such that $k = \lfloor |i|/|j| \rfloor$.

5. By 3 and 4, $gr(val_t(r)) \equiv_s \top$ when there is some triple i, j, k of numerals such that $i \in [t_1]$, $j \in [t_2]$, and r is the numeral $round(i/j)$.

6. By the definitions of the values of t, when $r \in [t]$, then there exist numerals i, j such that $i \in [t_1]$, $j \in [t_2]$, $j \neq 0$, and r is the numeral $round(i/j)$.

7. By 5 and 6, $gr(val_t(r)) \equiv_s \top$ when $r \in [t]$.

We now illustrate that $gr(val_t(r)) \equiv_s \bot$, when $r \notin [t]$.

1. It follows from (16) that $gr(val_t(r)) \equiv_s \bot$ when for all triples i, j, k of numerals such that $i \in [t_1]$ and $j \in [t_2]$, either $gr(F_1(ijk)) \equiv_s \bot$ or $gr(F_2(ijkr)) \equiv_s \bot$.

Consider an arbitrary triple i, j, k of numerals such that $i \in [t_1]$ and $j \in [t_2]$.

2. From Lemma 2 it follows that $gr(F_1(ijk)) \equiv_s \bot$ when $k \neq \lfloor |i|/|j| \rfloor$.

3. From Lemma 3 it follows that $gr(F_1(ijk)) \equiv_s \bot$ when $j = 0$.

From now on, we consider the case when $j \neq 0$ and $k = \lfloor |i|/|j| \rfloor$.

4. Under these assumptions it is sufficient to show that $gr(F_2(ijkr)) \equiv_s \bot$ to conclude that $gr(val_t(r)) \equiv_s \bot$ (see 1,2,3).

5. By the definition of the value of t, when $r \notin [t]$, then there do not exist numerals i, j such that $i \in [t_1]$, $j \in [t_2]$, $j \neq 0$, and r is the numeral $round(i/j)$.

6. From Lemma 4, we derive that $gr(F_2(ijkr)) \equiv_s \bot$ in case r is not the numeral $round(i/j)$.

7. By 4, 5 and 6, we conclude that $gr(val_t(r)) \equiv_s \bot$ when $r \notin [t]$.

Case 6:

Term t is $t_1 \setminus t_2$, where t_1 and t_2 have the induction hypothesis property. Then, $val_t(r)$ is $\exists IJK\big(val_{t_1}(I) \wedge val_{t_2}(J) \wedge F_1(IJK) \wedge F_3(IJKr)\big)$. By similar reasoning to that employed in Case 5, we conclude that

$$gr(val_t(r)) \equiv_s \big\{gr\big(F_1(ijk)\big) \wedge gr\big(F_3(ijkr)\big) \;\big|\; i, j, k \in |\mathscr{I}|^{s_i}, i \in [t_1], j \in [t_2]\big\}^{\vee} \quad (17)$$

We now illustrate that $gr(val_t(r)) \equiv_s \top$, when $r \in [t]$.

1. It follows from (17) that $gr(val_t(r)) \equiv_s \top$ if $gr(F_1(ijk)) \equiv_s \top$ and $gr(F_3(ijkr)) \equiv_s \top$ for some triple of numerals i, j, k such that $i \in [t_1]$ and $j \in [t_2]$.

2. From Lemma 2 it follows that $gr(F_1(ijk)) \equiv_s \top$ when $k = \lfloor |i|/|j| \rfloor$. Consequently, $gr(val_t(r)) \equiv_s \top$ if $gr(F_3(ijkr)) \equiv_s \top$ for some triple of numerals i, j, k such that $i \in [t_1]$, $j \in [t_2]$, and $k = \lfloor |i|/|j| \rfloor$.

3. From Lemma 5 it follows that $gr(F_3(ijkr)) \equiv_s \top$ if r is the numeral $i - j \times round(i/j)$ for i, j, k.

4. By 2 and 3, $gr(val_t(r)) \equiv_s \top$ if there exists a triple of numerals i, j, k such that $i \in [t_1]$, $j \in [t_2]$, $k = \lfloor |i|/|j| \rfloor$, and r is $i - j \times round(i/j)$.

5. By the definition of the value of t, when $r \in [t]$, then there exist numerals i, j such that $i \in [t_1]$, $j \in [t_2]$, $j \neq 0$, and r is the numeral $i - j \cdot round(i/j)$.

6. By 4 and 5, $gr(val_t(r)) \equiv_s \top$ when $r \in [t]$.

We now illustrate that $gr(val_t(r)) \equiv_s \bot$, when $r \notin [t]$.

1. It follows from (17) that $gr(val_t(r)) \equiv_s \bot$ when for all triples of numerals i, j, k such that $i \in [t_1]$ and $j \in [t_2]$, either $gr(F_1(ijk)) \equiv_s \bot$ or $gr(F_3(ijkr)) \equiv_s \bot$.

Consider an arbitrary triple i, j, k of numerals such that $i \in [t_1]$ and $j \in [t_2]$.

2. Recall from Case 5 that $gr(F_1(ijk)) \equiv_s \bot$ when $k \neq \lfloor |i|/|j| \rfloor$ or $j = 0$.

3. By 1 and 2, $gr(val_t(r)) \equiv_s \bot$ when $gr(F_3(ijkr)) \equiv_s \bot$ in case $k = \lfloor |i|/|j| \rfloor$ and $j \neq 0$.

4. By the definition of the value of t, when $r \notin [t]$, then there do not exist numerals i, j such that $i \in [t_1]$, $j \in [t_2]$, $j \neq 0$, and r is the numeral $i - j \cdot round(i/j)$.

5. By 3, 4, and Lemma 5, we conclude that $gr(val_t(r)) \equiv_s \bot$ when $r \notin [t]$.

Proof of Corollary 1 The following corollary is a consequence of Proposition 1 and is essentially a restatement of its claim for the case of a tuple of variables in place of a single variable. Recall that for a tuple of terms $t_1, \ldots, t_k$, abbreviated as $\mathbf{t}$, and a tuple of variables $V_1, \ldots, V_k$, abbreviated as $\mathbf{V}$, we use $val_{\mathbf{t}}(\mathbf{V})$ to denote the formula

$$val_{t_1}(V_1) \wedge \cdots \wedge val_{t_k}(V_{t_k}).$$

Corollary 1. *Let $\mathscr{I}$ be a standard interpretation, and let* $\mathbf{p}$ *be the standard partition. Then, for any k-tuple of program variables* $\mathbf{Z}$*, k-tuple of ground program terms* $\mathbf{t}$*, and k-tuple of precomputed terms* $\mathbf{r}$*, the formula* $gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{Z})^{\mathbf{Z}}_{\mathbf{r}})$ *is strongly equivalent to* $\top$ *if* $\langle r_1, \dots r_k\rangle \in [t_1, \dots, t_k]$ *and to* $\bot$ *otherwise.*

Proof. As before, we write $val_{\mathbf{t}}(\mathbf{r})$ instead of $val_{\mathbf{t}}(\mathbf{Z})^{\mathbf{Z}}_{\mathbf{r}}$ to denote the formula $val_{\mathbf{t}}(\mathbf{Z})$ where r_1 replaces Z_1, and so forth. Thus,

$$gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{r})) = gr^{\mathbf{p}}_{\mathscr{I}}(val_{t_1}(r_1)) \wedge \cdots \wedge gr^{\mathbf{p}}_{\mathscr{I}}(val_{t_k}(r_k))$$

Case 1: $\langle r_1, \dots r_k\rangle \in [t_1, \dots, t_k]$. By definition, $r_i \in [t_i]$ $(1 \le i \le k)$, thus by Proposition 1 each formula $gr^{\mathbf{p}}_{\mathscr{I}}(val_{t_i}(r_i)) \equiv_s \top$. Since every conjunctive term in $gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{r}))$ is strongly equivalent to $\top$, it follows that $gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{r}))$ is strongly equivalent to $\top$.
Case 2: $\langle r_1, \dots r_k\rangle \notin [t_1, \dots, t_k]$. By definition, there exists an r_i $(1 \le i \le k)$ such that $r_i \notin [t_i]$. By Proposition 1 it follows that the corresponding formula $gr^{\mathbf{p}}_{\mathscr{I}}(val_{t_i}(r_i)) \equiv_s \bot$. Thus, $gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{r}))$ is strongly equivalent to $\bot$.

A.3 Proof of Proposition 2

Lemma 6. *Let $\mathscr{I}$ be a standard interpretation and let* $\mathbf{p}$ *be the standard partition. Then, for any n-tuple of program variables* $\mathbf{Z}$*, n-tuple of ground program terms* $\mathbf{t}$*, some universe* $|\mathscr{I}|$ *and formula G, it follows that*

$$\{gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{Z})^{\mathbf{Z}}_{\mathbf{r}}) \wedge gr^{\mathbf{p}}_{\mathscr{I}}(G(\mathbf{Z})^{\mathbf{Z}}_{\mathbf{r}}) \mid \mathbf{r} \in |\mathscr{I}|^n\}^{\vee} \equiv_s \{gr^{\mathbf{p}}_{\mathscr{I}}(G(\mathbf{r})) \mid \mathbf{r} \in |\mathscr{I}|^n, \mathbf{r} \in [\mathbf{t}]\}^{\vee}$$

Proof. First note that $\mathbf{r}$ is an n-tuple of precomputed terms, and that $gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{Z})^{\mathbf{Z}}_{\mathbf{r}})$ is more conveniently written as $gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{r}))$ (similarly, rewrite $gr^{\mathbf{p}}_{\mathscr{I}}(G(\mathbf{Z})^{\mathbf{Z}}_{\mathbf{r}})$ as $gr^{\mathbf{p}}_{\mathscr{I}}(G(\mathbf{r}))$). We now consider the unique partition of $|\mathscr{I}|^n$ into sets $\mathbf{J}$ and $\mathbf{K}$ so that: $\mathbf{J}$ denotes the set of tuples of the form $\langle j_1, \dots, j_n\rangle$ such that every $j_i \in [t_i]$ $(1 \le i \le n)$. As a result, every tuple $\langle k_1, \dots, k_n\rangle \in \mathbf{K}$ is such that for some i $(1 \le i \le n)$, $k_i \notin [t_i]$. By construction of $\mathbf{J}$ and $\mathbf{K}$,

$$\begin{aligned}&\{gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{Z})^{\mathbf{Z}}_{\mathbf{r}}) \wedge gr^{\mathbf{p}}_{\mathscr{I}}(G(\mathbf{Z})^{\mathbf{Z}}_{\mathbf{r}}) \mid \mathbf{r} \in |\mathscr{I}|^n\}^{\vee}\\ &= \{\left(gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{j})) \wedge gr^{\mathbf{p}}_{\mathscr{I}}(G(\mathbf{j}))\right), \left(gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{k})) \wedge gr^{\mathbf{p}}_{\mathscr{I}}(G(\mathbf{k}))\right) \mid \mathbf{j} \in \mathbf{J}, \mathbf{k} \in \mathbf{K}\}^{\vee}\end{aligned}$$

From Corollary 1, it follows that $gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{j})) \equiv_s \top$ and $gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{k})) \equiv_s \bot$ for each $\mathbf{j} \in \mathbf{J}, \mathbf{k} \in \mathbf{K}$. Thus,

$$\begin{aligned}&\{gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{Z})^{\mathbf{Z}}_{\mathbf{r}}) \wedge gr^{\mathbf{p}}_{\mathscr{I}}(G(\mathbf{Z})^{\mathbf{Z}}_{\mathbf{r}}) \mid \mathbf{r} \in |\mathscr{I}|^n\}^{\vee}\\ &\equiv_s \{\left(\top \wedge gr^{\mathbf{p}}_{\mathscr{I}}(G(\mathbf{j}))\right), \left(\bot \wedge gr^{\mathbf{p}}_{\mathscr{I}}(G(\mathbf{k}))\right) \mid \mathbf{j} \in \mathbf{J}, \mathbf{k} \in \mathbf{K}\}^{\vee} (\textit{Fact } 5)\\ &\equiv_s \{gr^{\mathbf{p}}_{\mathscr{I}}(G(\mathbf{j})) \mid \mathbf{j} \in \mathbf{J}\}^{\vee} (\text{Fact 1, construction of } \mathbf{J})\\ &\equiv_s \{gr^{\mathbf{p}}_{\mathscr{I}}(G(\mathbf{r})) \mid \mathbf{r} \in |\mathscr{I}|^n, \mathbf{r} \in [\mathbf{t}]\}^{\vee}\end{aligned}$$

Lemma 7. *Let l be a ground basic literal or ground comparison, let $\mathscr{I}$ be a standard interpretation, and let* $\mathbf{p}$ *be the standard partition. Then, for any tuple of program variables* $\mathbf{Z}$*,*

$$gr^{\mathbf{p}}_{\mathscr{I}}(\tau^B_{\mathbf{Z}}(l)) \equiv_s \tau(l).$$

Proof. We proceed by cases.

Case 1: l is an atom, $p(t_1,\dots,t_k)$ where p is an intensional symbol ($p \in \mathbf{p}$). Then,

$$\tau^B_{\mathbf{Z}}(l) = \exists V_1,\dots V_k(val_{t_1}(V_1) \wedge \cdots \wedge val_{t_k}(V_k) \wedge p(V_1,\dots,V_k))$$

and

$$\begin{aligned}
&gr^{\mathbf{p}}_{\mathscr{I}}(\tau^B_{\mathbf{Z}}(l))\\
&= \{gr^{\mathbf{p}}_{\mathscr{I}}(val_{t_1}(V_1)^{V_1}_{r_1}) \wedge \cdots \wedge gr^{\mathbf{p}}_{\mathscr{I}}(val_{t_k}(V_k)^{V_k}_{r_k}) \wedge gr^{\mathbf{p}}_{\mathscr{I}}(p(V_1,\dots,V_k)^{V_1,\dots,V_k}_{r_1,\dots,r_k}) \mid \langle r_1,\dots,r_k\rangle \in |\mathscr{I}|^{s_p} \times \cdots \times |\mathscr{I}|^{s_p}\}^{\vee}\\
&\equiv_s \{gr^{\mathbf{p}}_{\mathscr{I}}(p(r_1,\dots,r_k)) \mid \langle r_1,\dots,r_k\rangle \in |\mathscr{I}|^{s_p} \times \cdots \times |\mathscr{I}|^{s_p}, \langle r_1,\dots,r_k\rangle \in [t_1,\dots,t_k]\}^{\vee} \text{(Lemma 6)}\\
&= \{p(r_1,\dots,r_k) \mid \langle r_1,\dots,r_k\rangle \in [t_1,\dots,t_k]\}^{\vee} \text{ (Definition of } gr^{\mathbf{p}}_{\mathscr{I}})\\
&\equiv_s \tau(l) \text{ (Definition of } \tau\text{, conditions 2, 3)}
\end{aligned}$$

Cases 2,3: l is a singly or doubly negated intensional atom. The proof of these cases mimics the proof of Case 1.

Case 4: l is a comparison, $t_1 \prec t_2$. Then,

$$\tau^B_{\mathbf{Z}}(l) = \exists V_1, V_2(val_{t_1}(V_1) \wedge val_{t_2}(V_2) \wedge V_1 \prec V_2)$$

and

$$\begin{aligned}
&gr^{\mathbf{p}}_{\mathscr{I}}(\tau^B_{\mathbf{Z}}(l))\\
&= \{gr^{\mathbf{p}}_{\mathscr{I}}(val_{t_1}(V_1)^{V_1}_{r_1}) \wedge gr^{\mathbf{p}}_{\mathscr{I}}(val_{t_2}(V_2)^{V_2}_{r_2}) \wedge gr^{\mathbf{p}}_{\mathscr{I}}((V_1 \prec V_2)^{V_1,V_2}_{r_1,r_2}) \mid \langle r_1,r_2\rangle \in |\mathscr{I}|^{s_p} \times |\mathscr{I}|^{s_p}\}^{\vee}\\
&\equiv_s \{gr^{\mathbf{p}}_{\mathscr{I}}(r_1 \prec r_2) \mid \langle r_1,r_2\rangle \in |\mathscr{I}|^{s_p} \times |\mathscr{I}|^{s_p}, \langle r_1,r_2\rangle \in [t_1,t_2]\}^{\vee} \text{(Lemma 6)}\\
&\equiv_s \{\top \mid r_1 \in [t_1], r_2 \in [t_2], \prec(r_1,r_2)\}^{\vee} \vee \{\bot \mid r_1 \in [t_1], r_2 \in [t_2], \not\prec(r_1,r_2)\}^{\vee} \text{ (Definition of } gr^{\mathbf{p}}_{\mathscr{I}} \text{ conditions 3, 4)}\\
&\equiv_s \{\top \mid r_1 \in [t_1], r_2 \in [t_2], \prec(r_1,r_2)\}^{\vee} \vee \bot \text{(Fact 1, condition 4)}\\
&\equiv_s \{\top \mid r_1 \in [t_1], r_2 \in [t_2], \prec(r_1,r_2)\}^{\vee} \text{(Fact 1, condition 4)}
\end{aligned}$$

The preceding argument illustrates that $gr^{\mathbf{p}}_{\mathscr{I}}(\tau^B_{\mathbf{Z}}(l))$ is strongly equivalent to $\top$ if the set $\{\top \mid r_1 \in [t_1], r_2 \in [t_2], \prec(r_1,r_2)\}$ is not empty, and strongly equivalent to $\bot$ otherwise. Therefore, $gr^{\mathbf{p}}_{\mathscr{I}}(\tau^B_{\mathbf{Z}}(l))$ is strongly equivalent to $\top$ if some $\langle r_1,r_2\rangle \in [t_1,t_2]$ satisfies $r_1 \prec r_2$ and $\bot$ otherwise. By condition 6 of the definition of τ, $gr^{\mathbf{p}}_{\mathscr{I}}(\tau^B_{\mathbf{Z}}(l)) \equiv_s \tau(l)$.

Lemma 8. *Let $l_1,\dots,l_m$ be a sequence composed of ground basic literals and ground comparisons and let H be a ground basic literal, a ground comparison, or the symbol $\bot$. Furthermore, let $\mathscr{I}$ be a standard interpretation and let $\mathbf{p}$ be the standard partition. Then, for any tuple $\mathbf{Z}$ of program variables,*

$$gr^{\mathbf{p}}_{\mathscr{I}}\big(\tau^B_{\mathbf{z}}(l_1)\big) \wedge \cdots \wedge gr^{\mathbf{p}}_{\mathscr{I}}\big(\tau^B_{\mathbf{z}}(l_m)\big) \rightarrow gr^{\mathbf{p}}_{\mathscr{I}}\big(\tau^B_{\mathbf{z}}(H)\big) \equiv_s \tau(l_1) \wedge \cdots \wedge \tau(l_m) \rightarrow \tau(H)$$

Proof. We proceed by cases over H.

Case 1: When H is a ground basic literal or a ground comparison, the result follows from Fact 5 and Lemma 7.

Case 2: H is $\bot$. Since $gr^{\mathbf{p}}_{\mathscr{I}}\big(\tau^B_{\mathbf{Z}}(\bot)\big) = gr^{\mathbf{p}}_{\mathscr{I}}(\bot) = \bot = \tau(\bot)$, the result follows from Fact 5 and Lemma 7.

Proof of Proposition 2

Proof. First note that by assumption $H : \mathbf{L}$ is closed – thus, every global variable has been replaced with a precomputed term and only local variables $\mathbf{X}$ remain. Consequently, any expression $E^{\mathbf{X}}_{\mathbf{x}}$ where $\mathbf{x}$ is a tuple of precomputed terms of the same length as $\mathbf{X}$ is a ground expression. Thus,

$$\begin{aligned}
&gr^{\mathbf{p}}_{\mathscr{I}}\left(\tau^B_{\mathbf{Z}}(H : \mathbf{L})\right)\\
&= gr^{\mathbf{p}}_{\mathscr{I}}\left(\forall \mathbf{X}(\tau^B_{\mathbf{Z}}(\mathbf{L}) \to \tau^B_{\mathbf{Z}}(H)\right)\\
&= gr^{\mathbf{p}}_{\mathscr{I}}\left(\forall \mathbf{X}(\tau^B_{\mathbf{Z}}(l_1) \wedge \cdots \wedge \tau^B_{\mathbf{Z}}(l_m) \to \tau^B_{\mathbf{Z}}(H)\right)\\
&= \{gr^{\mathbf{p}}_{\mathscr{I}}(\tau^B_{\mathbf{Z}}([l_1]^{\mathbf{X}}_{\mathbf{x}})) \wedge \cdots \wedge gr^{\mathbf{p}}_{\mathscr{I}}(\tau^B_{\mathbf{Z}}([l_m]^{\mathbf{X}}_{\mathbf{x}})) \to gr^{\mathbf{p}}_{\mathscr{I}}(\tau^B_{\mathbf{Z}}(H^{\mathbf{X}}_{\mathbf{x}})) \mid \mathbf{x} \in |\mathscr{I}|^{s_p} \times \cdots \times |\mathscr{I}|^{s_p}\}^{\wedge}\\
&\equiv_s \{\tau([l_1]^{\mathbf{X}}_{\mathbf{x}}) \wedge \cdots \wedge \tau([l_m]^{\mathbf{X}}_{\mathbf{x}}) \to \tau(H^{\mathbf{X}}_{\mathbf{x}}) \mid \mathbf{x} \in |\mathscr{I}|^{s_p} \times \cdots \times |\mathscr{I}|^{s_p}\}^{\wedge}(\text{Lemma 8})\\
&= \tau(H : \mathbf{L})
\end{aligned}$$

A.4 Proofs of Propositions 3 and 4

In the following, it is convenient to abuse notation for a k-tuple $\mathbf{v}$ composed of terms from a universe $|\mathscr{I}|$: we write $\mathbf{v} \in |\mathscr{I}|$ to denote $\mathbf{v} \in |\mathscr{I}|^k$.

The proof of Proposition 3 relies on a rule *instantiation* process used to obtain closed rules. Let $\mathbf{Z}$ denote the global variables of rule R. By $inst_{\mathscr{G}}(R)$ we denote the set of instances of rule R w.r.t. a set $\mathscr{G}$ of precomputed terms, i.e.,

$$inst_{\mathscr{G}}(R) = \{R^{\mathbf{Z}}_{\mathbf{z}} \mid \mathbf{z} \in \mathscr{G}\} = \{[H]^{\mathbf{Z}}_{\mathbf{z}} \text{ :- } [B_1]^{\mathbf{Z}}_{\mathbf{z}}, \ldots, [B_n]^{\mathbf{Z}}_{\mathbf{z}} \mid \mathbf{z} \in \mathscr{G}\}.$$

If we take $\mathscr{G}$ to be the universe of precomputed terms, then $\tau(R) = \{\tau(\rho) \mid \rho \in inst_{\mathscr{G}}(R)\}^{\wedge}$.

Lemma 9. *Let* $\mathbf{v}$ *be a tuple of precomputed terms, let F and G be infinitary propositional formulas, and let* $\mathbf{t}$ *be a tuple of ground terms of the same length as* $\mathbf{v}$. *For a tuple of precomputed terms* $\mathbf{r}$ *of the same length as* $\mathbf{v}$,

$$\{F \to G^{\mathbf{v}}_{\mathbf{r}} \mid \mathbf{r} \in [\mathbf{t}]\}^{\wedge} \tag{18}$$

is strongly equivalent to

$$F \to \bigwedge_{\mathbf{r} \in [\mathbf{t}]} G^{\mathbf{v}}_{\mathbf{r}} \tag{19}$$

Proof. Theorem 3 by [18] states that two (sets of) infinitary formulas are strongly equivalent if and only if they are equivalent in the infinitary logic of here-and-there. This proof will illustrate that infinitary formulas (18) and (19) are equivalent in the infinitary logic of here-and-there.

Take any HT-interpretation $\langle S, S'\rangle$. We will consider two cases. In the first case, we assume that $\langle S,S'\rangle \models (18)$ and illustrate that from this assumption it follows that $\langle S,S'\rangle \models (19)$. In the second case, we assume that $\langle S,S'\rangle \models (19)$ and illustrate that from this assumption it follows that $\langle S,S'\rangle \models (18)$. These arguments illustrate that formulas (18) and (19) are equivalent in the infinitary logic of here-and-there.
Case 1: Assume $\langle S,S'\rangle \models (18)$. It follows that $\langle S,S'\rangle \models F \rightarrow G^{\mathbf{v}}_{\mathbf{r}}$ for any $\mathbf{r} \in [\mathbf{t}]$. From the definition of ht-satisfaction, we conclude that for any $\mathbf{r} \in [\mathbf{t}]$

- $S' \models F \rightarrow G^{\mathbf{v}}_{\mathbf{r}}$ and
- $\langle S,S'\rangle \not\models F$ or $\langle S,S'\rangle \models G^{\mathbf{v}}_{\mathbf{r}}$.

We now illustrate that $S' \models F \rightarrow \bigwedge_{\mathbf{r}\in[\mathbf{t}]} G^{\mathbf{v}}_{\mathbf{r}}$. For the case when $S' \not\models F$, the statement vacuously holds. Now assume that $S' \models F$. Since $S' \models F \rightarrow G^{\mathbf{v}}_{\mathbf{r}}$ for any $\mathbf{r} \in [\mathbf{t}]$, it follows that $S' \models G^{\mathbf{v}}_{\mathbf{r}}$ for any $\mathbf{r} \in [\mathbf{t}]$. Equivalently, $S' \models \bigwedge_{\mathbf{r}\in[\mathbf{t}]} G^{\mathbf{v}}_{\mathbf{r}}$. Thus, $S' \models F \rightarrow \bigwedge_{\mathbf{r}\in[\mathbf{t}]} G^{\mathbf{v}}_{\mathbf{r}}$.

Since $\langle S,S'\rangle \not\models F$ or $\langle S,S'\rangle \models G^{\mathbf{v}}_{\mathbf{r}}$ for any $\mathbf{r} \in [\mathbf{t}]$, and since the satisfaction of F is independent of $\mathbf{r}$, it follows that

- $\langle S,S'\rangle \not\models F$ or
- $\langle S,S'\rangle \models G^{\mathbf{v}}_{\mathbf{r}}$ for any $\mathbf{r} \in [\mathbf{t}]$.

Consequently, $\langle S,S'\rangle \not\models F$ or $\langle S,S'\rangle \models \bigwedge_{\mathbf{r}\in[\mathbf{t}]} G^{\mathbf{v}}_{\mathbf{r}}$.

It now follows from the definition of ht-satisfaction that $\langle S,S'\rangle \models (19)$.
Case 2: Assume $\langle S,S'\rangle \models (19)$. It follows that

- $S' \models F \rightarrow \bigwedge_{\mathbf{r}\in[\mathbf{t}]} G^{\mathbf{v}}_{\mathbf{r}}$ and
- $\langle S,S'\rangle \not\models F$ or $\langle S,S'\rangle \models \bigwedge_{\mathbf{r}\in[\mathbf{t}]} G^{\mathbf{v}}_{\mathbf{r}}$.

To illustrate that $\langle S,S'\rangle \models (18)$, it is sufficient to show that $\langle S,S'\rangle \models F \rightarrow G^{\mathbf{v}}_{\mathbf{n}}$ for an arbitrary $\mathbf{n} \in [\mathbf{t}]$.

Now assume that $S' \models F$. It follows that $S' \models \bigwedge_{\mathbf{r}\in[\mathbf{t}]} G^{\mathbf{v}}_{\mathbf{r}}$, therefore, $S' \models G^{\mathbf{v}}_{\mathbf{n}}$. Thus, $S' \models F \rightarrow G^{\mathbf{v}}_{\mathbf{n}}$.

Since $\langle S,S'\rangle \not\models F$ or $\langle S,S'\rangle \models \bigwedge_{\mathbf{r}\in[\mathbf{t}]} G^{\mathbf{v}}_{\mathbf{r}}$, it follows that $\langle S,S'\rangle \not\models F$ or $\langle S,S'\rangle \models G^{\mathbf{v}}_{\mathbf{n}} \wedge \bigwedge_{\mathbf{r}\in[\mathbf{t}]\setminus\mathbf{n}} G^{\mathbf{v}}_{\mathbf{r}}$. Therefore, $\langle S,S'\rangle \not\models F$ or $\langle S,S'\rangle \models G^{\mathbf{v}}_{\mathbf{n}}$.

Thus, $\langle S,S'\rangle \models F \rightarrow G^{\mathbf{v}}_{\mathbf{n}}$ for arbitrary $\mathbf{n} \in [\mathbf{t}]$. Consequently, $\langle S,S'\rangle \models (18)$.

The claim below follows from the definition of ht- and classical satisfaction using well known classical equivalences.

Lemma 10. *Let F and G be infinitary propositional formulas. Then,*

$$F \wedge \neg\neg G \rightarrow G \equiv_s F \rightarrow G \vee \neg G.$$

Lemma 11. *Let t be a term from a rule containing global variables* $\mathbf{Z}$*, and let* $\mathbf{z}$ *be a tuple of precomputed terms of the same length as* $\mathbf{Z}$*. Let V be a program variable which does not occur in* $\mathbf{Z}$ *or t, and let v be a precomputed term. Then,*

$$\left(val_t(V)^V_v\right)^{\mathbf{Z}}_{\mathbf{z}} = \left(val_{t^{\mathbf{Z}}_{\mathbf{z}}}(V)\right)^V_v$$

Proof. The proof uses structural induction across the forms of terms; we say that a term t has the induction hypothesis property when the following equivalence holds for t and a program variable V which does not occur in $\mathbf{Z}$:

$$val_t(V)^{\mathbf{Z}}_{\mathbf{z}} = val_{t^{\mathbf{Z}}_{\mathbf{z}}}(V)$$

We illustrate the first two cases here.

Case 1: t is a numeral, symbolic constant, variable, *inf* or *sup*.

$$\begin{aligned}
\left(val_t(V)^V_v\right)^{\mathbf{Z}}_{\mathbf{z}} &= \left((V = t)^V_v\right)^{\mathbf{Z}}_{\mathbf{z}} \\
&= (v = t)^{\mathbf{Z}}_{\mathbf{z}} \\
&= \left(v = t^{\mathbf{Z}}_{\mathbf{z}}\right) \\
&= \left(V = t^{\mathbf{Z}}_{\mathbf{z}}\right)^V_v \\
&= \left(val_{t^{\mathbf{Z}}_{\mathbf{z}}}(V)\right)^V_v
\end{aligned}$$

Case 2: t is $|t_1|$, where $|t_1|$ has the induction hypothesis property.

$$\begin{aligned}
\left(val_t(V)^V_v\right)^{\mathbf{Z}}_{\mathbf{z}} &= \left(\exists I\,(val_{t_1}(I) \land V = |I|)^V_v\right)^{\mathbf{Z}}_{\mathbf{z}} \\
&= (\exists I\,(val_{t_1}(I) \land v = |I|))^{\mathbf{Z}}_{\mathbf{z}} \\
&= \exists I\left(val_{t_1}(I)^{\mathbf{Z}}_{\mathbf{z}} \land (v = |I|)^{\mathbf{Z}}_{\mathbf{z}}\right) \\
&= \exists I\left(val_{t_1{}^{\mathbf{Z}}_{\mathbf{z}}}(I) \land v = |I|\right) \qquad \text{(induction hypothesis)} \\
&= \left(\exists I\left(val_{t_1{}^{\mathbf{Z}}_{\mathbf{z}}}(I) \land V = |I|\right)\right)^V_v \\
&= \left(val_{t^{\mathbf{Z}}_{\mathbf{z}}}(V)\right)^V_v
\end{aligned}$$

Lemma 12. *Let* $\mathbf{t}$ *be a k-tuple of ground terms,* $\mathbf{V}$ *be a k-tuple of program variables,* F, G *be infinitary propositional formulas,* $\mathscr{I}$ *be a standard interpretation, and* $\mathbf{p}$ *be the standard partition. It holds that*

$$\begin{aligned}
&\{gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{V}))^{\mathbf{V}}_{\mathbf{v}} \land F \to G \mid \mathbf{v} \in |\mathscr{I}|^{s_p}\}^{\land} \equiv_s \\
&\{F \to G \mid \mathbf{v} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in [\mathbf{t}]\}^{\land}
\end{aligned}$$

Proof.

$$
\begin{aligned}
&\{gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{V}))^{\mathbf{V}}_{\mathbf{v}} \wedge F \to G \mid \mathbf{v} \in |\mathscr{I}|^{s_p}\}^{\wedge} \\
=&\{gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{v})) \wedge F \to G \mid \mathbf{v} \in |\mathscr{I}|^{s_p}\}^{\wedge} \\
\equiv_s&\{gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{v})) \wedge F \to G \mid \mathbf{v} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in [\mathbf{t}]\}^{\wedge} \wedge \\
&\{gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{w})) \wedge F \to G \mid \mathbf{w} \in |\mathscr{I}|^{s_p}, \mathbf{w} \notin [\mathbf{t}]\}^{\wedge} \\
\equiv_s&\{\top \wedge F \to G \mid \mathbf{v} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in [\mathbf{t}]\}^{\wedge} \wedge \\
&\{\bot \wedge F \to G \mid \mathbf{w} \in |\mathscr{I}|^{s_p}, \mathbf{w} \notin [\mathbf{t}]\}^{\wedge} && \text{Corollary 1} \\
\equiv_s&\{F \to G \mid \mathbf{v} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in [\mathbf{t}]\}^{\wedge} \wedge \{\bot \to G \mid \mathbf{w} \in |\mathscr{I}|^{s_p}, \mathbf{w} \notin [\mathbf{t}]\}^{\wedge} && \text{Fact 1: condition 2} \\
\equiv_s&\{F \to G \mid \mathbf{v} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in [\mathbf{t}]\}^{\wedge} \wedge \{\top \mid \mathbf{w} \in |\mathscr{I}|^{s_p}, \mathbf{w} \notin [\mathbf{t}]\}^{\wedge} && \text{Fact 1: condition 5} \\
\equiv_s&\{F \to G \mid \mathbf{v} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in [\mathbf{t}]\}^{\wedge} && \text{Fact 1: condition 1}
\end{aligned}
$$

Proof of Proposition 3

Proof. We proceed by cases.

Case 1: R is a normal rule with atom $p(\mathbf{t})$ in the head. By definition,

$$gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*(R)) = gr^{\mathbf{p}}_{\mathscr{I}}(\forall \mathbf{V}\,(\forall \mathbf{Z}\,(val_{\mathbf{t}}(\mathbf{V}) \wedge \tau^*_{\mathbf{Z}}(B_1) \wedge \cdots \wedge \tau^*_{\mathbf{Z}}(B_n) \to p(\mathbf{V}))))$$

where $\mathbf{V}$ is a tuple of fresh, alphabetically first program variables disjoint from the global variables $\mathbf{Z}$ of the same length as $\mathbf{t}$. Let $\mathbf{z}$ denote a tuple of precomputed terms of the same length as $\mathbf{Z}$, and let $\mathbf{v}$ denote a tuple of precomputed terms of the same length as $\mathbf{V}$. Then, $gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*(R))$ is equal to

$$
\begin{aligned}
&\left\{gr^{\mathbf{p}}_{\mathscr{I}}\left(\forall \mathbf{Z}(val_{\mathbf{t}}(\mathbf{v}) \wedge \tau^*_{\mathbf{Z}}(B_1) \wedge \cdots \wedge \tau^*_{\mathbf{Z}}(B_n) \to p(\mathbf{V})^{\mathbf{V}}_{\mathbf{v}})\right) \mid \mathbf{v} \in |\mathscr{I}|^{s_p}\right\}^{\wedge} \\
&= \left\{\{gr^{\mathbf{p}}_{\mathscr{I}}\left(val_{\mathbf{t}}(\mathbf{v})^{\mathbf{Z}}_{\mathbf{z}} \wedge \tau^*_{\mathbf{Z}}(B_1)^{\mathbf{Z}}_{\mathbf{z}} \wedge \cdots \wedge \tau^*_{\mathbf{Z}}(B_n)^{\mathbf{Z}}_{\mathbf{z}} \to p(\mathbf{v})^{\mathbf{Z}}_{\mathbf{z}}\right) \mid \mathbf{z} \in |\mathscr{I}|^{s_p}\}^{\wedge} \mid \mathbf{v} \in |\mathscr{I}|^{s_p}\right\}^{\wedge} \\
&= \left\{\{gr^{\mathbf{p}}_{\mathscr{I}}\left(val_{\mathbf{t}}(\mathbf{v})^{\mathbf{Z}}_{\mathbf{z}}\right) \wedge gr^{\mathbf{p}}_{\mathscr{I}}\left(\tau^*_{\mathbf{Z}}(B_1)^{\mathbf{Z}}_{\mathbf{z}}\right) \wedge \cdots \wedge gr^{\mathbf{p}}_{\mathscr{I}}\left(\tau^*_{\mathbf{Z}}(B_n)^{\mathbf{Z}}_{\mathbf{z}}\right) \to gr^{\mathbf{p}}_{\mathscr{I}}(p(\mathbf{v})) \right. \\
&\qquad\qquad \left. \mid \mathbf{z} \in |\mathscr{I}|^{s_p}\}^{\wedge} \mid \mathbf{v} \in |\mathscr{I}|^{s_p}\right\}^{\wedge} \\
&= \left\{\{gr^{\mathbf{p}}_{\mathscr{I}}\left(val_{\mathbf{t}}(\mathbf{v})^{\mathbf{Z}}_{\mathbf{z}}\right) \wedge gr^{\mathbf{p}}_{\mathscr{I}}\left(\tau^*_{\mathbf{Z}}([B_1]^{\mathbf{Z}}_{\mathbf{z}})\right) \wedge \cdots \wedge gr^{\mathbf{p}}_{\mathscr{I}}\left(\tau^*_{\mathbf{Z}}([B_n]^{\mathbf{Z}}_{\mathbf{z}})\right) \to p(\mathbf{v}) \right. \\
&\qquad\qquad \left. \mid \mathbf{z} \in |\mathscr{I}|^{s_p}\}^{\wedge} \mid \mathbf{v} \in |\mathscr{I}|^{s_p}\right\}^{\wedge}
\end{aligned}
$$

From the preceding and the definition of ht-satisfaction, it follows that $gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*(R))$ is strongly equivalent to

$$\{gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{v})^{\mathbf{Z}}_{\mathbf{z}}) \wedge gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*_{\mathbf{Z}}([B_1]^{\mathbf{Z}}_{\mathbf{z}})) \wedge \cdots \wedge gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*_{\mathbf{Z}}([B_n]^{\mathbf{Z}}_{\mathbf{z}})) \to p(\mathbf{v}) \mid \mathbf{z} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in |\mathscr{I}|^{s_p}\}^{\wedge}$$

Note that each expression $[B_i]^{\mathbf{Z}}_{\mathbf{z}}$ is a closed conditional literal. Thus, from Proposition 2 it follows that

$$gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*_{\mathbf{z}}([B_i]^{\mathbf{Z}}_{\mathbf{z}})) \equiv_s \tau([B_i]^{\mathbf{Z}}_{\mathbf{z}})$$

and so, from Fact 5, $gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*(R))$ is strongly equivalent to

$$\{gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{v})^{\mathbf{Z}}_{\mathbf{z}}) \wedge \tau([B_1]^{\mathbf{Z}}_{\mathbf{z}}) \wedge \cdots \wedge \tau([B_n]^{\mathbf{Z}}_{\mathbf{z}}) \to p(\mathbf{v}) \mid \mathbf{z} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in |\mathscr{I}|^{s_p}\}^{\wedge}.$$

From Lemma 11, which is generalized in a straightforward way to tuples of terms and variables, we conclude that $gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{v})^{\mathbf{Z}}_{\mathbf{z}}) = gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}^{\mathbf{Z}}_{\mathbf{z}}}(\mathbf{v}))$ for any $\mathbf{v} \in |\mathscr{I}|^{s_p}$. Thus, it follows that $gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*(R))$ is strongly equivalent to

$$\{gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}^{\mathbf{Z}}_{\mathbf{z}}}(\mathbf{v})) \wedge \tau([B_1]^{\mathbf{Z}}_{\mathbf{z}}) \wedge \cdots \wedge \tau([B_n]^{\mathbf{Z}}_{\mathbf{z}}) \to p(\mathbf{v}) \mid \mathbf{z} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in |\mathscr{I}|^{s_p}\}^{\wedge}.$$

From Lemma 12, it follows that $gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*(R))$ is strongly equivalent to

$$\{\tau([B_1]^{\mathbf{Z}}_{\mathbf{z}}) \wedge \cdots \wedge \tau([B_n]^{\mathbf{Z}}_{\mathbf{z}}) \to p(\mathbf{v}) \mid \mathbf{z} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in [\mathbf{t}^{\mathbf{Z}}_{\mathbf{z}}]\}^{\wedge}. \tag{20}$$

Since R is a normal rule with the atom $p(\mathbf{t})$ in the head, the rule instantiation process for R produces the following:

$$\begin{aligned}\tau(R) &= \{\tau\left(p(\mathbf{t})^{\mathbf{Z}}_{\mathbf{z}} \text{ :- } [B_1]^{\mathbf{Z}}_{\mathbf{z}}, \ldots, [B_n]^{\mathbf{Z}}_{\mathbf{z}}\right) \mid \mathbf{z} \in |\mathscr{I}|^{s_p}\}^{\wedge} \\ &= \{\tau\left(p(\mathbf{t}^{\mathbf{Z}}_{\mathbf{z}}) \text{ :- } [B_1]^{\mathbf{Z}}_{\mathbf{z}}, \ldots, [B_n]^{\mathbf{Z}}_{\mathbf{z}}\right) \mid \mathbf{z} \in |\mathscr{I}|^{s_p}\}^{\wedge} \\ &= \{\tau([B_1]^{\mathbf{Z}}_{\mathbf{z}}) \wedge \cdots \wedge \tau([B_n]^{\mathbf{Z}}_{\mathbf{z}}) \to \bigwedge_{\mathbf{r} \in [\mathbf{t}^{\mathbf{Z}}_{\mathbf{z}}]} p(\mathbf{r}) \mid \mathbf{z} \in |\mathscr{I}|^{s_p}\}^{\wedge}\end{aligned}$$

It only remains to be shown that (20) is strongly equivalent to the preceding formula. In the sequel we abbreviate expression $\tau([B_1]^{\mathbf{Z}}_{\mathbf{z}}) \wedge \cdots \wedge \tau([B_n]^{\mathbf{Z}}_{\mathbf{z}})$ by $\mathbf{B}^{\mathbf{Z}}_{\mathbf{z}}$.

Now take tuple of precomputed terms $\mathbf{z} \in |\mathscr{I}|^{s_p}$ of length $\mathbf{Z}$. It remains to show that

$$\{\mathbf{B}^{\mathbf{Z}}_{\mathbf{z}} \to p(\mathbf{v}) \mid \mathbf{v} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in [\mathbf{t}^{\mathbf{Z}}_{\mathbf{z}}]\}^{\wedge} \equiv_s \mathbf{B}^{\mathbf{Z}}_{\mathbf{z}} \to \bigwedge_{\mathbf{r} \in [\mathbf{t}^{\mathbf{Z}}_{\mathbf{z}}]} p(\mathbf{r}).$$

Since $\mathbf{B}^{\mathbf{Z}}_{\mathbf{z}}$ is an infinitary formula and since $\mathbf{t}^{\mathbf{Z}}_{\mathbf{z}}$ is a tuple of ground terms of the same length as $\mathbf{v}$, this result follows directly from Lemma 9.

Case 2: R is a choice rule with the head of the form $\{p(\mathbf{t})\}$. We take $\mathbf{V}$, $\mathbf{Z}$, $\mathbf{t}$, $\mathbf{z}$, $\mathbf{v}$ to denote the same entities as in Case 1. Now,

$$\begin{aligned}& gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*(R)) \\ &= gr^{\mathbf{p}}_{\mathscr{I}}\Big(\forall \mathbf{V}\mathbf{Z}\big(val_{\mathbf{t}}(\mathbf{V}) \wedge \tau^*_{\mathbf{Z}}(B_1) \wedge \cdots \wedge \tau^*_{\mathbf{Z}}(B_n) \wedge \neg\neg p(\mathbf{V}) \to p(\mathbf{V})\big)\Big) \\ &= \{gr^{\mathbf{p}}_{\mathscr{I}}(\forall \mathbf{Z}(val_{\mathbf{t}}(\mathbf{v}) \wedge \tau^*_{\mathbf{Z}}(B_1) \wedge \cdots \wedge \tau^*_{\mathbf{Z}}(B_n) \wedge \neg\neg p(\mathbf{v}) \to p(\mathbf{v}))) \mid \mathbf{v} \in |\mathscr{I}|^{s_p}\}^{\wedge} \\ &= \{gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{v})^{\mathbf{Z}}_{\mathbf{z}}) \wedge gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*_{\mathbf{Z}}(B_1)^{\mathbf{Z}}_{\mathbf{z}}) \wedge \cdots \wedge gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*_{\mathbf{Z}}(B_n)^{\mathbf{Z}}_{\mathbf{z}}) \wedge gr^{\mathbf{p}}_{\mathscr{I}}(\neg\neg p(\mathbf{v})) \\ &\qquad \to gr^{\mathbf{p}}_{\mathscr{I}}(p(\mathbf{v})) \mid \mathbf{z} \in |\mathscr{I}|^{s_p}\}^{\wedge}, \mathbf{v} \in |\mathscr{I}|^{s_p}\}^{\wedge} \\ &= \{gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{v})^{\mathbf{Z}}_{\mathbf{z}}) \wedge gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*_{\mathbf{Z}}([B_1]^{\mathbf{Z}}_{\mathbf{z}})) \wedge \cdots \wedge gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*_{\mathbf{Z}}([B_n]^{\mathbf{Z}}_{\mathbf{z}})) \wedge \neg\neg gr^{\mathbf{p}}_{\mathscr{I}}(p(\mathbf{v})) \\ &\qquad \to p(\mathbf{v}) \mid \mathbf{z} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in |\mathscr{I}|^{s_p}\}^{\wedge} \\ &= \{gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{v})^{\mathbf{Z}}_{\mathbf{z}}) \wedge gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*_{\mathbf{Z}}([B_1]^{\mathbf{Z}}_{\mathbf{z}})) \wedge \cdots \wedge gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*_{\mathbf{Z}}([B_n]^{\mathbf{Z}}_{\mathbf{z}})) \wedge \neg\neg p(\mathbf{v}) \\ &\qquad \to p(\mathbf{v}) \mid \mathbf{z} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in |\mathscr{I}|^{s_p}\}^{\wedge}\end{aligned}$$

From Proposition 2 and Fact 5, it follows that $gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*(R))$ is strongly equivalent to

$$\begin{aligned}&\{gr^{\mathbf{p}}_{\mathscr{I}}(val_{\mathbf{t}}(\mathbf{v})^{\mathbf{Z}}_{\mathbf{z}}) \wedge \tau\left([B_1]^{\mathbf{Z}}_{\mathbf{z}}\right) \wedge \cdots \wedge \tau\left([B_n]^{\mathbf{Z}}_{\mathbf{z}}\right) \wedge \neg\neg p(\mathbf{v}) \to p(\mathbf{v}) \mid \mathbf{z} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in |\mathscr{I}|^{s_p}\}^{\wedge} \\ &\equiv_s \{\tau\left([B_1]^{\mathbf{Z}}_{\mathbf{z}}\right) \wedge \cdots \wedge \tau\left([B_n]^{\mathbf{Z}}_{\mathbf{z}}\right) \wedge \neg\neg p(\mathbf{v}) \to p(\mathbf{v}) \mid \mathbf{z} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in [\mathbf{t}^{\mathbf{Z}}_{\mathbf{z}}]\}^{\wedge}\end{aligned}$$

(Lemmas 11, 12)

$$\equiv_s \{\tau\left([B_1]^{\mathbf{Z}}_{\mathbf{z}}\right) \wedge \cdots \wedge \tau\left([B_n]^{\mathbf{Z}}_{\mathbf{z}}\right) \to p(\mathbf{v}) \vee \neg p(\mathbf{v}) \mid \mathbf{z} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in [\mathbf{t}^{\mathbf{Z}}_{\mathbf{z}}]\}^{\wedge}.$$

(Lemma 10)

Since R is a choice rule with atom $\{p(\mathbf{t})\}$ in the head, the rule instantiation process for R produces the following:

$$\tau(R) = \{\tau([B_1]^{\mathbf{Z}}_{\mathbf{z}}) \wedge \cdots \wedge \tau([B_n]^{\mathbf{Z}}_{\mathbf{z}}) \to \bigwedge_{\mathbf{r} \in [\mathbf{t}^{\mathbf{Z}}_{\mathbf{z}}]} (p(\mathbf{r}) \vee \neg p(\mathbf{r})) \mid \mathbf{z} \in |\mathscr{I}|^{s_p}\}^{\wedge}.$$

Recall abbreviation $\mathbf{B}^{\mathbf{Z}}_{\mathbf{z}}$ from Case 1 and take any tuple of precomputed terms $\mathbf{z} \in |\mathscr{I}|^{s_p}$ of the same length as $\mathbf{Z}$. It remains to show that

$$\{\mathbf{B}^{\mathbf{Z}}_{\mathbf{z}} \to (p(\mathbf{v}) \vee \neg p(\mathbf{v})) \mid \mathbf{v} \in |\mathscr{I}|^{s_p}, \mathbf{v} \in [\mathbf{t}^{\mathbf{Z}}_{\mathbf{z}}]\}^{\wedge} \equiv_s \mathbf{B}^{\mathbf{Z}}_{\mathbf{z}} \to \bigwedge_{\mathbf{r} \in [\mathbf{t}^{\mathbf{Z}}_{\mathbf{z}}]} (p(\mathbf{r}) \vee \neg p(\mathbf{r})).$$

Since $\mathbf{B}^{\mathbf{Z}}_{\mathbf{z}}$ is an infinitary formula and since $\mathbf{t}^{\mathbf{Z}}_{\mathbf{z}}$ is a tuple of ground terms of the same length as $\mathbf{v}$, this result follows directly from Lemma 9.

Case 3: R is a constraint. By definition,

$$\begin{aligned} gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*(R)) &= gr^{\mathbf{p}}_{\mathscr{I}}\left(\forall \mathbf{Z}\left(\tau^B_{\mathbf{Z}}(B_1) \wedge \cdots \wedge \tau^B_{\mathbf{Z}}(B_n) \to \bot\right)\right) \\ &= \{gr^{\mathbf{p}}_{\mathscr{I}}\left(\tau^*_{\mathbf{Z}}(B_1)^{\mathbf{Z}}_{\mathbf{z}} \wedge \cdots \wedge \tau^*_{\mathbf{Z}}(B_n)^{\mathbf{Z}}_{\mathbf{z}} \to \bot\right) \mid \mathbf{z} \in |\mathscr{I}|^{s_p}\}^{\wedge} \\ &= \{gr^{\mathbf{p}}_{\mathscr{I}}\left(\tau^*_{\mathbf{Z}}([B_1]^{\mathbf{Z}}_{\mathbf{z}})\right) \wedge \cdots \wedge gr^{\mathbf{p}}_{\mathscr{I}}\left(\tau^*_{\mathbf{Z}}([B_n]^{\mathbf{Z}}_{\mathbf{z}})\right) \to \bot \mid \mathbf{z} \in |\mathscr{I}|^{s_p}\}^{\wedge} \end{aligned}$$

From Proposition 2 and Fact 5, it follows that

$$gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*(R)) \equiv_s \{\tau([B_1]^{\mathbf{Z}}_{\mathbf{z}}) \wedge \cdots \wedge \tau([B_n]^{\mathbf{Z}}_{\mathbf{z}}) \to \bot \mid \mathbf{z} \in |\mathscr{I}|^{s_p}\}^{\wedge}$$

Since R is a constraint, the rule instantiation process for R produces the following:

$$\begin{aligned} \tau(R) &= \{\neg\tau\left([B_1]^{\mathbf{Z}}_{\mathbf{z}} \wedge \cdots \wedge [B_n]^{\mathbf{Z}}_{\mathbf{z}}\right) \mid \mathbf{z} \in |\mathscr{I}|^{s_p}\}^{\wedge} \\ &= \{\tau\left([B_1]^{\mathbf{Z}}_{\mathbf{z}} \wedge \cdots \wedge [B_n]^{\mathbf{Z}}_{\mathbf{z}}\right) \to \bot \mid \mathbf{z} \in |\mathscr{I}|^{s_p}\}^{\wedge} \\ &= \{\tau\left([B_1]^{\mathbf{Z}}_{\mathbf{z}}\right) \wedge \cdots \wedge \tau\left([B_n]^{\mathbf{Z}}_{\mathbf{z}}\right) \to \bot \mid \mathbf{z} \in |\mathscr{I}|^{s_p}\}^{\wedge} \end{aligned}$$

Thus, $gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*(R)) \equiv_s \tau(R)$.

Proof of Proposition 4

Proof. By definition, $\tau^*\Pi$ is the first-order theory containing τ^*R for every rule R in Π. Thus,

$$gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*\Pi) = \{gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*R) \mid R \in \Pi\}^{\wedge}.$$

By definition, $\tau\Pi = \{\tau R \mid R \in \Pi\}$. Note that

$$\{\tau R \mid R \in \Pi\} \equiv_s \{\tau R \mid R \in \Pi\}^{\wedge}$$

as by the definition of ht-satisfaction these sets of infinitary formulas share the same HT-models. From Fact 5 and Proposition 3, it follows that

$$\{\tau R \mid R \in \Pi\}^{\wedge} \equiv_s \{gr^{\mathbf{p}}_{\mathscr{I}}(\tau^* R) \mid R \in \Pi\}^{\wedge}$$

Thus, $gr^{\mathbf{p}}_{\mathscr{I}}(\tau^*(\Pi)) \equiv_s \tau\Pi$.