Negation-as-Failure in the Base-extension Semantics for Intuitionistic Propositional Logic

Alexander V. Gheorghiu    David J. Pym

Abstract

Proof-theoretic semantics (P-tS) is the paradigm of semantics in which meaning in logic is based on proof (as opposed to truth). A particular instance of P-tS for intuitionistic propositional logic (IPL) is its base-extension semantics (B-eS). In this paper, we prove completeness of IPL with respect to the B-eS through logic programming (LP). This reveals that inherent to P-tS is proof-search and the negation-as-failure protocol. Traditionally, the denial of a proposition is understood as the assertion of its negation, which is indeed the case in the model-theoretic semantics for IPL, but the connection to LP reveals that in P-tS the denial of a proposition refers to the failure of finding a proof of it. In this way, assertion and denial are both prime concepts in P-tS.

Keywords: Logic programming, proof-theoretic semantics, bilateralism, negation-as-failure.

1. Introduction

The definition of a system of logic may be given proof-theoretically as a collection of rules of inference that, when composed, determine proofs; that is, formal constructions of arguments that establish that a conclusion is a consequence of some assumptions:

\[
\begin{array}{c}
\text{Premiss}_1 \\
\vdots \\
\text{Premiss}_k \\
\hline
\text{Conclusion}
\end{array}
\]

The systematic use of symbolic and mathematical techniques to determine the forms of valid deductive argument defines deductive logic: conclusions are inferred from assumptions.
This is all very well as a way of defining what proofs are, but it relatively rarely reflects either the way in which logic is used in practical reasoning problems or the method by which proofs are actually found. Rather, proofs are more often constructed by starting with a desired, or putative, conclusion and applying the rules of inference ‘backwards’. In this usage, the rules are sometimes called *reduction operators*, read from conclusion to premisses, and denoted

\[
\begin{array}{c}
\text{Sufficient Premiss}_1 & \ldots & \text{Sufficient Premiss}_k \\
\hline
\text{Putative Conclusion}
\end{array}
\]

Constructions in a system of reduction operators are called *reductions*. This paradigm is known as *reductive logic*. The space of reductions of a putative conclusion is larger than its space of proofs, including also failed searches — Pym and Ritter [22] have studied the reductive logic for intuitionistic and classical logic in which such objects are meaningful entities.

Logic programming (LP) is a particular use of reductive logic to capture reasoning. While the reductive logic perspective on LP is, perhaps, somewhat obscured by the usual presentation of Horn-clause LP with SLD-resolution — see, for example, Kowalski [14] and Lloyd [17] — it is explicit in work by Miller et al. [19, 20]. The basic idea is that a configuration is a pair \( \mathcal{P} \vdash G \), in which \( \mathcal{P} \) is a *program* (i.e., a set of *definite* formulas) and \( G \) is a *goal* formula, the execution of which is defined by the search for a *uniform* proof.

Deductive logic is suitable for considering the validity of propositions relative to sets of axioms. In contrast, reductive logic is suitable for considering the meaning of propositions relative to *systems of inference*. That the semantics of a statement is determined by its inferential behaviour is known as *inferentialism* (see Brandom [2]), which has a mathematical realization as *proof-theoretic semantics* (P-tS).

In P-tS, the meaning of the logical connectives is usually derived from the rules of a natural deduction system for the logic — for example, typically, one uses Gentzen’s [30] NJ for intuitionistic logic. Meanwhile, the meanings of atomic propositions are supplied by *atomic systems* — sets of rules over atomic propositions. For example, taken from Sandqvist [25], the meaning of the proposition ‘Tammy is a vixen’ arises from the following rule:

\[
\begin{array}{c}
\text{Tammy is a fox} & \text{Tammy is female} \\
\hline
\text{Tammy is a vixen}
\end{array}
\]
Sandqvist [28] gave a P-tS for intuitionistic propositional logic (IPL) called base-extension semantics (B-eS) using an inductively defined judgement, called support, whose base case is given by provability in an atomic system.

There is an intuitive relationship between P-tS and LP: the way in which sets of rules are definitional in P-tS is precisely the way in which a program in LP is definitional. Schroeder-Heister and Hallnäs [9, 10] have used this relationship to address questions of harmony and inversion in P-tS. In this paper, we show that the completeness of IPL with respect to the B-eS can be understood in terms of LP by considering the execution of a special program $N$ that precisely encodes the inferential behaviour of the connectives.

The P-tS of negation is a subtle issue — see, for example, Kürbis [16]. Meanwhile, in LP, the relationship between provability, refutation, and negation is a well-established protocol known as negation-as-failure (NAF) — a statement $\neg \varphi$ is established precisely when the system fails to find a proof for $\varphi$. The completeness argument for IPL in this paper shows that this NAF is essential for the B-eS of IPL. In particular, negation in B-eS is essentially failure to find a proof. Hence, from the perspective of B-eS, it is not the case, as advanced by Frege [5] and endorsed by Dummett [4], that denying a statement $\varphi$ is equal to asserting the negation of $\varphi$. Instead, denial in P-tS is conceptually prior to negation and refers to the failure of finding a proof. In this way, through the lens of reductive logic, P-tS practices a form of bilateralism — the philosophical practice of giving equal consideration to dual concepts such as assertion and denial, and truth and falsity, and so on.

The paper has two parts. In the first part (i.e., Section 2), we give the relevant background on IPL and its B-eS: Section 2.1 contains the syntax and terminology that we adopt for IPL; Section 2.2 defines its B-eS; and, Section 2.3 presents the completeness argument as given by Sandqvist [28]. In the second part (i.e., Section 3), we relate the B-eS to LP: Section 3.1 gives the background to LP for IPL; Section 3.2 formally relates atomic systems in B-eS and programs in LP; Section 3.3 presents a completeness argument for IPL with respect to the B-eS using LP; and, Section 3.4 illustrates how negation-as-failure manifests in the B-eS. The paper concludes in Section 4 with a summary of our results and discussion of future work.
2. Intuitionistic Logic

2.1. Syntax and Proof Theory

There are various presentations of intuitionistic propositional logic (IPL) in the literature. We begin by fixing the relevant concepts and terminology as used in this paper.

**Definition 2.1 (Formulas).** Fix a set of atomic propositions $\mathcal{A}$. The set of formulas $\mathcal{F}$ (over $\mathcal{A}$) is constructed by the following grammar:

$\varphi ::= p \in \mathcal{A} \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi \rightarrow \varphi \mid \bot$

We use the following abbreviations:

$\hat{\varphi} := \bigwedge_{p \in \varphi} p$  
$\neg \varphi := \varphi \rightarrow \bot$

**Definition 2.2 (Sequent).** A sequent is a pair $\Gamma : \varphi$ in which $\Gamma$ is a set of formulas and $\varphi$ is a formula.

We use $\vdash$ as the IPL-consequence judgement on sequents; that is, $\Gamma \vdash \varphi$ denotes that the sequent $\Gamma : \varphi$ is a consequence of IPL. We may write $\vdash \varphi$ to abbreviate $\emptyset \vdash \varphi$.

Throughout, we assume familiarity with the standard natural deduction system $NJ$ and sequent calculus $LJ$ for IPL as introduced by Gentzen [30].

2.2. Base-extension Semantics

In this section, we give a brief, but complete, synopsis of the base-extension semantics (B-eS) for IPL introduced by Sandqvist [28]. Unless stated otherwise, we are working in the setting of natural deduction.

The meaning of atomic propositions is given by sets of atomic rules governing their inferential behaviour. Piecha and Schroeder-Heister [29, 21] have given a useful inductive hierarchy of them.

**Definition 2.3 (Atomic Rule).** An $n$th-level atomic rule is defined as follows:

- A zeroth-level atomic rule is a rule of the following form in which $c \in \mathcal{A}$:

$\overline{c}$
- A first-level atomic rule is a rule of the following form in which
  \( p_1, ..., p_n, c \in \mathcal{A} \),

\[
\frac{p_1 \ldots p_n}{c}
\]

- An \((n + 1)\)th-level atomic rule is a rule of the following form in which
  \( p_1, ..., p_n, c \in \mathcal{A} \) and \( \Sigma_1, ..., \Sigma_n \) are (possibly empty) sets of \( n \)th-level atomic rules:

\[
\frac{[\Sigma_1]}{p_1} \ldots \frac{[\Sigma_n]}{p_n}
\]

Having sets of atomic rule as hypotheses is more general than have sets of atomic propositions as hypotheses; the latter is captured by the former by taking zeroth-order atomic rules. Nonetheless, the generalization is, perhaps, unexpected. We discuss it further in Section 3.3.

**Definition 2.4 (Atomic System).** An atomic system is a set of atomic rules. An atomic system is an \( n \)th-level atomic system if it contains \( n \)th-level atomic rules.

Atomic systems may have infinitely many rules, but they are at most countably infinite. They are used to base validity in P-tS on proof. In this paper, we will only consider second-level atomic systems, for which the definition of a derivation is as usual (i.e., as in Gentzen [30]); for the more general case of \( n \)th-level atomic systems, see Schroeder-Heister and Piecha [29, 21].

Typically, we do not consider all atomic systems, but restrict attention to some particular class.

**Definition 2.5 (Basis).** A basis is a set of atomic systems.

Having fixed a basis \( \mathcal{B} \), an atomic system \( \mathcal{B} \in \mathcal{B} \) is called a base. A base-extension semantics is formulated relative to a basis via a support relation.

**Definition 2.6 (Support in a Base).** Fix a basis \( \mathcal{B} \). That a sequent \( \Gamma : \varphi \) is supported in base \( \mathcal{B} \in \mathcal{B} \) — denoted \( \Gamma \vdash_{\mathcal{B}} \varphi \) — is defined by the clause of Figure 1. The validity judgement \( \Gamma \vdash \varphi \) obtains iff \( \Gamma \vdash_{\mathcal{B}} \varphi \) obtains for any \( \mathcal{B} \in \mathcal{B} \).
We may write $\Vdash \varphi$ and $\vdash \varphi$ to abbreviate $\emptyset \Vdash \varphi$ and $\emptyset \vdash \varphi$, respectively.

Observe that the clauses of Figure 1 are inductive — see Sandqvist [28]. We call the induction ordering provided by them proof-theoretic induction. It is a kind of structural induction in which disjunction ($\lor$) is considered later than any of the other connectives; in particular, $\varphi \lor \psi$ is later than $(\varphi \rightarrow p) \land (\psi \rightarrow p) \rightarrow p$ for any $p \in A$.

In this paper, we will restrict attention to the basis of atomic systems containing at most second-level rules, which we call the Sandqvist basis $\mathcal{S}$.

**Theorem 2.7 (Soundness & Completeness).** $\Gamma \vdash \varphi$ iff $\Gamma \vDash \varphi$ over $\mathcal{S}$.

**Proof:** Proved by Sandqvist [28] — see Section 2.3.

There are related base-extension semantics for classical logic — see Sandqvist [26, 27] and Makinson [18].

This summarizes the B-eS for IPL. In the next section we present the completeness proof as provided by Sandqvist [28] as it will be useful to understand the connections to reductive logic later on.

### 2.3. Completeness of IPL via a Natural Base

Sandqvist [28] proved the soundness of IPL with respect to the B-eS by showing that validity admits all the rules of NJ. The proof of completeness is more subtle. In essence, Sandqvist [28] proved completeness of IPL for the B-eS by simulating an NJ-derivation by an atomic system $\mathcal{N}$ that is bespoke to the given validity judgement.
We want to show that if $\Gamma \vdash \gamma$ obtains, then there is a $\text{NJ}$-proof witnessing $\Gamma \vdash \gamma$. To this end, we associate to each formula $\rho$ in the sequent $\Gamma : \gamma$ a unique atom $r$ and construct a base $\mathcal{N}$ emulating $\text{NJ}$ such that $r$ behaves in $\mathcal{N}$ as $\rho$ behaves in $\text{NJ}$. For example, let $\Gamma : \gamma$ contain $\rho := p \land q$. The rules governing $\rho$ are the conjunction introduction and elimination rules of $\text{NJ}$, so we require $\mathcal{N}$ to contain the following rules in which $r$ is alien to $\Gamma$:

$\begin{array}{c}
& p & q \\
\hline 
 p & r & r
\end{array}$

These rules are designed such that $r$ behaves in $\mathcal{N}$ precisely as $\rho$ does in $\text{NJ}$; that is, they emulate the conjunction rules. The shorthand for $r$ is $(p \land q)^\flat$, so that the above rules may be expressed more clearly as follows:

$\begin{array}{c}
& p & q \\
\hline 
 (p \land q)^\flat & p & (p \land q)^\flat
\end{array}$

If $\Gamma : \gamma$ also contains $\sigma := p \rightarrow q$, then $\mathcal{N}$ contains the following rules too, which emulate the implication introduction and elimination rules of $\text{NJ}$ for $\sigma$, in which $(p \rightarrow q)^\flat$ is alien to $\Gamma$ and $\gamma$:

$\begin{array}{c}
 (p \rightarrow q)^\flat & p & (p \rightarrow q)^\flat \\
\hline 
 q & p & q
\end{array}$

The details of how $\mathcal{N}$ is constructed and how it delivers completeness are below.

Given $\Gamma \vdash \gamma$, to every formula $\varphi$ occurring in $\Gamma : \gamma$ associate a unique atomic proposition $\varphi^\flat$ as follows:

- if $\varphi \notin \mathbb{A}$, then $\varphi^\flat$ does not occur in $\Gamma : \gamma$;

- if $\varphi \in \mathbb{A}$, then $\varphi^\flat = \varphi$.

The right-inverse of $-^\flat$ is $-^\natural$ and both functions act on sets point-wise,

$\Sigma^\flat := \{ \varphi^\flat \mid \varphi \in \Sigma \} \quad \Sigma^\natural := \{ \varphi^\natural \mid \varphi \in \Sigma \}$

Let $\mathcal{N}$ be the atomic system containing precisely the rules of Figure 2 for any $\varphi$, $\psi$, and $\chi$ occurring in $\Gamma : \gamma$. These rules are precisely such that $\varphi^\flat$ behaves in $\mathcal{N}$ as $\varphi$ does in $\text{NJ}$. Note that, for any validity judgement,
the atomic system $\mathcal{N}$ thus generated is indeed a Sandqvist base; moreover, it is a finite set.

The following shows that $\varphi^b$ acts in $\mathcal{N}$ as intended:

**Lemma 2.8.** For any $\varphi$, $\models_{\mathcal{N} \cup \{\varphi^b\}} \varphi$.

**Proof:** The clauses of support (Figure 1) all emulate elimination rules of $NJ$ when base-extension is trivial (i.e., taking $\mathcal{C} = B$ in the condition $\mathcal{C} \supseteq B$). The unfolding of $\models_{\mathcal{N} \cup \{\varphi^b\}} \varphi$ with trivial base-extensions terminates in judgements of the form $\models_{\mathcal{N} \cup \{\varphi^b\}} p$ in which $p \in A$, but since these elimination rules may be emulated by $\mathcal{N}$, it follows that that $\vdash_{\mathcal{N} \cup \{\varphi^b\}} p$. Hence, $\models_{\mathcal{N} \cup \{\varphi^b\}} \varphi$, as required. $\square$

In this set-up, Sandqvist [28] establishes three properties that collectively deliver completeness.

**Lemma 2.9.** Let $\Sigma \subseteq A$ and $p \in A$ and let $B \in S$,

$$\Sigma \models_{B} p \iff \Sigma \vdash_{B} p$$

This claim is a basic completeness result in which the context $\Sigma$ is restricted to a set of atomic propositions and the extract p is an atomic proposition.

**Lemma 2.10.** For every $\varphi$ occurring in $\Gamma : \gamma$ and any $\mathcal{N}' \supseteq \mathcal{N}$,

$$\emptyset \models_{\mathcal{N}'} \varphi^b \iff \emptyset \vdash_{\mathcal{N}'} \varphi$$

In other words, $\varphi^b$ and $\varphi$ are equivalent in $\mathcal{N}$ — that is, $\varphi^b \models_{\mathcal{N}} \varphi$ and $\varphi \vdash_{\mathcal{N}} \varphi^b$. The property allows us to move between the basic case (i.e., the

---

**Figure 2. Atomic System $\mathcal{N}$**

| $\varphi^b$ | $\psi^b$ | $([\varphi^b])$ | $([\psi^b])$ | $([\varphi^b])$ | $([\varphi^b])$ | $([\varphi^b])$ | $([\psi^b])$ |
|---------------|---------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
| $(\varphi \land \psi)^b$ | $\varphi^b$ | $(\varphi \land \psi)^b$ | $\psi^b$ | $(\varphi \rightarrow \psi)^b$ | $\varphi^b$ | $(\varphi \land \psi)^b$ | $\psi^b$ | $(\varphi \land \psi)^b$ |
| $\varphi^b$ | $\psi^b$ | $(\varphi \land \psi)^b$ | $\chi^b$ | $\chi^b$ | $\varphi^b$ | $(\varphi \land \psi)^b$ | $\psi^b$ | $(\varphi \land \psi)^b$ |

---

The atomic system $\mathcal{N}$ thus generated is indeed a Sandqvist base; moreover, it is a finite set.

The following shows that $\varphi^b$ acts in $\mathcal{N}$ as intended:

**Lemma 2.8.** For any $\varphi$, $\models_{\mathcal{N} \cup \{\varphi^b\}} \varphi$.

**Proof:** The clauses of support (Figure 1) all emulate elimination rules of $NJ$ when base-extension is trivial (i.e., taking $\mathcal{C} = B$ in the condition $\mathcal{C} \supseteq B$). The unfolding of $\models_{\mathcal{N} \cup \{\varphi^b\}} \varphi$ with trivial base-extensions terminates in judgements of the form $\models_{\mathcal{N} \cup \{\varphi^b\}} p$ in which $p \in A$, but since these elimination rules may be emulated by $\mathcal{N}$, it follows that that $\vdash_{\mathcal{N} \cup \{\varphi^b\}} p$. Hence, $\models_{\mathcal{N} \cup \{\varphi^b\}} \varphi$, as required. $\square$

In this set-up, Sandqvist [28] establishes three properties that collectively deliver completeness.

**Lemma 2.9.** Let $\Sigma \subseteq A$ and $p \in A$ and let $B \in S$,

$$\Sigma \models_{B} p \iff \Sigma \vdash_{B} p$$

This claim is a basic completeness result in which the context $\Sigma$ is restricted to a set of atomic propositions and the extract p is an atomic proposition.

**Lemma 2.10.** For every $\varphi$ occurring in $\Gamma : \gamma$ and any $\mathcal{N}' \supseteq \mathcal{N}$,

$$\emptyset \models_{\mathcal{N}'} \varphi^b \iff \emptyset \vdash_{\mathcal{N}'} \varphi$$

In other words, $\varphi^b$ and $\varphi$ are equivalent in $\mathcal{N}$ — that is, $\varphi^b \models_{\mathcal{N}} \varphi$ and $\varphi \vdash_{\mathcal{N}} \varphi^b$. The property allows us to move between the basic case (i.e., the
set-up of Lemma 2.9) and the general case (i.e., completeness — Theorem 2.7).

**Lemma 2.11.** Let $\Sigma \subseteq \mathbb{A}$ and $p \in \mathbb{A}$,

$$\Sigma \vdash_{\mathcal{N}} p \text{ implies } \Sigma^\natural \vdash p^\natural$$

This property is the simulation statement. It allows us to make the final move from derivability in $\mathcal{N}$ to derivability in $\mathcal{NJ}$.

These lemmas collectively suffice for completeness:

**Proof:** Theorem 2.7 — Completeness. If $\Gamma \vdash \chi$, then $\Gamma^\flat \vdash_{\mathcal{N}} \chi^\natural$ because if $\mathcal{N}' \supseteq \mathcal{N}$ and $\emptyset \vdash_{\mathcal{N}'} \phi^\natural$ for $\phi^\natural \in \Gamma^\flat$, then (by Lemma 2.10) $\emptyset \vdash_{\mathcal{N}'} \phi$ for every $\phi \in \Gamma$. Hence, $\emptyset \vdash_{\mathcal{N}'} \chi$ (since $\Gamma \vdash \chi$); whence (by Lemma 2.10) $\emptyset \vdash_{\mathcal{N}'} \chi^\natural$; whence (by Lemma 2.9) it follows that $\Gamma^\flat \vdash_{\mathcal{N}} \chi^\flat$. Thus (by Lemma 2.11) it follows that $\Gamma \vdash \chi$.

In the next section, we show that the completeness follows intuitively from regarding $\mathcal{N}$ as a program capturing the inferential content of $\mathcal{NJ}$. In general, a base may be regarded as a program, so that the application of a rule in the base corresponds to the use of a clause in the program. We demonstrate that the validity of a formula $\varphi$ in the base $\mathcal{N}$ emulates the execution of a goal $\varphi^\natural$ relative to the program $\mathcal{N}$. By construction of $\mathcal{N}$, such executions simulate the construction of an $\mathcal{NJ}$ proof of $\varphi$. Hence, IPL is complete with respect to the B-eS.

### 3. Logic Programming

#### 3.1. The Hereditary Harrop Logic Programming Language

To begin, we introduce the relevant logic programming (LP) language for this paper. This section closely follows work by Miller [19] (see also Harland [11]).

The propositional hereditary Harrop formulas are generated by the following grammar in which $A \in \mathbb{A} \cup \{ \bot \}$ is an atomic proposition, $D$ is a **definite formula**, and $G$ is a **goal formula**:

$$
D : = A \mid G \rightarrow A \mid D \wedge D \\
G : = A \mid D \rightarrow G \mid G \wedge G \mid G \vee G
$$
A program \( \mathcal{P} \) is a finite set of definite formulas; the set of all programs is \( \mathbb{P} \). We call a sequent \( \mathcal{P} : G \), in which \( \mathcal{P} \) is a program and \( G \) is a goal, a query. For purely technical reasons, we require a decomposition function \([-]\) : \( \mathbb{P} \rightarrow \mathbb{P} \) that will unpack conjunctions. Let \([\mathcal{P}]\) be the least set satisfying the following:

- \( \mathcal{P} \subseteq [\mathcal{P}] \)
- If \( D_1 \land D_2 \in [\mathcal{P}] \), then \( D_1 \in [\mathcal{P}] \) and \( D_2 \in [\mathcal{P}] \).

The hereditary Harrop fragment of IPL produces a logic programming language hHLP whose operational semantics is given by proof-search for \( \mathcal{P} : G \) in LJ. The proofs generated are uniform proofs — see Miller et al. [20].

**Definition 3.1 (Operational Semantics for hHLP).** The operational semantics for hHLP is given by the clauses in Figure 3.

**Definition 3.2 (Interpretation).** An interpretation is a mapping \( I : \mathbb{P} \rightarrow \mathcal{P}(\mathbb{A}) \) such that \( B \subseteq C \) implies \( I(B) \subseteq I(C) \).

**Definition 3.3 (Satisfaction).** The satisfaction judgement is given by the clauses of Figure 4.
\[ I, \mathcal{P} \models A \iff A \in I(\mathcal{P}) \]
\[ I, \mathcal{P} \models G_1 \lor G_2 \iff I, \mathcal{P} \models G_1 \text{ or } I, \mathcal{P} \models G_2 \]
\[ I, \mathcal{P} \models G_1 \land G_2 \iff I, \mathcal{P} \models G_1 \text{ and } I, \mathcal{P} \models G_2 \]
\[ I, \mathcal{P} \models D \rightarrow G \iff I, \mathcal{P} \cup \{D\} \models G \]

**Figure 4.** Denotational Semantics for hHLP

We desire a particular interpretation \( J \) such that the following holds:

\[ J, \mathcal{P} \models G \iff \mathcal{P} \vdash G \]

To this end, we consider a function \( T \) from interpretations to interpretations that corresponds to unfolding derivability in a base:

\[
T(I)(\mathcal{P}) := \{ A \mid A \in [\mathcal{P}] \} \cup \\
\{ A \mid (G \rightarrow A) \in [\mathcal{P}] \text{ and } I, \mathcal{P} \not\models G \} \cup \\
\{ A \mid I, \mathcal{P} \not\models \bot \}
\]

Interpretations form a lattice under point-wise union (\( \sqcup \)), point-wise intersection (\( \sqcap \)), and point-wise subset (\( \subseteq \)); the bottom of the lattice is given by \( I_{\bot} : \mathcal{P} \rightarrow \emptyset \). It is easy to see that \( T \) is monotonic and continuous on this lattice, and, by the Knaster-Tarski Theorem \([1]\), its least fixed-point is given as follows:

\[ T^\omega I_{\bot} := I_{\bot} \sqcup T(I_{\bot}) \sqcup T^2(I_{\bot}) \sqcup \ldots \]

Intuitively, each application of \( T \) concerns the application of a clause so that \( T^\omega I_{\bot} \) corresponds to arbitrarily many applications.

**Lemma 3.4.** For any program \( \mathcal{P} \) and goal \( G \),

\[ T^\omega I_{\bot}, \mathcal{P} \models G \iff \mathcal{P} \vdash G \]

The result follows from the proof in Miller \([19]\) with modification to handle absurdity (\( \bot \)); see also Harland \([11]\).

**Proof:** First, we prove \( \mathcal{P} \vdash G \) implies \( T^\omega I_{\bot}, \mathcal{P} \models G \).

The proof is by induction on the height of executions for \( \mathcal{P} \vdash G \). For \( G \neq \bot \), the proof proceed as usual, except that it may always be that the final rule was EFQ. If \( \mathcal{P} \vdash G \) follows from EFQ, then \( \mathcal{P} \vdash \bot \). By the
induction hypothesis, $T^\omega I_\perp, \mathcal{P} \models \perp$. Hence, by the definition of $T$, we have $T^\omega I_\perp, \mathcal{P} \models A$ for any $A \in \mathcal{A}$. It follows that $T^\omega I_\perp, \mathcal{P} \models G$, as required.

There are two possibilities for $G = \perp$:

- If the last inference was $\text{IN}$, then $\perp \in \Gamma$. Hence, by definition of $T$, we have $\perp \in T I_\perp(\mathcal{P})$. Whence, by monotonicity, $\perp \in T^\omega I_\perp(\mathcal{P})$. Thus, by definition, $T^\omega I_\perp, \mathcal{P} \models \perp$.

- If the last inference was $\text{CLAUSE}$, then $(G \rightarrow \perp) \in [\mathcal{P}]$ such that $\mathcal{P} \models G$. By the induction hypothesis, $T^\omega I_\perp, \mathcal{P} \models G$. By the definition of $T$, we have $T(T^\omega I_\perp, \mathcal{P} \models \perp)$. Hence, since $T(T^\omega I_\perp) = T^\omega I_\perp$, we have $T^\omega I_\perp, \mathcal{P} \models \perp$, as required.

Second, we prove that $T^\omega I_\perp, \mathcal{P} \models G$ implies $\mathcal{P} \models G$.

Let $k \geq 1$ be the least integer such that $T^k I_\perp, \mathcal{P} \models G$ and let $n \geq 0$ be the number of logical connectives in $G$, excluding $\perp$. The proof is by ordinal induction on $\omega \cdot (k - 1) + n$. For $G \neq \perp$, the proof proceeds as usual.

There are two possibilities for $G = \perp$:

- Let $k = 1$. Then we have $TI_\perp, \mathcal{P} \models \perp$. By the definition of $T$, it must be that $\perp \in [\mathcal{P}]$. By $\text{IN}$, we have $\mathcal{P} \models \perp$, as required.

- Let $k > 1$. Then we have $T^k I_\perp, \mathcal{P} \models \perp$. By the definition of $T$, we have $\perp \in T^{k+1} I_\perp(\mathcal{P})$. There are two possibilities: $\perp \in [\mathcal{P}]$ or there is $G' \rightarrow \perp \in [\mathcal{P}]$ such that $T^k I_\perp, \mathcal{P} \models G'$. In the first case, $\mathcal{P} \models \perp$ is immediate by $\text{IN}$. In the second case, the ordinal $\omega \cdot (k - 2) + \beta'$, in which $\beta'$ is the number of logical connectives in $G'$, is smaller than $\omega \cdot (k - 1)$. Hence, by the induction hypothesis, $P \models G'$. By $\text{CLAUSE}$, we have $P \models \perp$, as required.

In the next section we relate atomic systems in B-eS to programs in LP, formally.

3.2. Atomic Systems vs. Programs

Intuitively, atomic systems in B-eS are definitional in precisely the same way as programs in hHLP are definitional. To illustrate this, we must systematically move between them, which we do by encoding atomic systems as programs.
Let $[-]$ be as follows:

- The encoding of zeroth-level rule is as follows:
  \[
  \left[\overline{c}\right] := c
  \]

- The encoding of a first-level rule is as follows:
  \[
  \left[\overline{p_1 \ldots p_n c}\right] := (p_1 \land \ldots \land p_n) \rightarrow c
  \]

- The encoding of an $n$th-level rule is as follows:
  \[
  \left[\Sigma_1 \ldots \Sigma_n \overline{p_1 \ldots p_n c}\right] := \big((\Sigma_1 \rightarrow p_1) \land \ldots \land (\Sigma_n \rightarrow p_n)\big) \rightarrow c
  \]

We suppress the encoding function for readability when it is clear from context that we are reading the atomic system as a program, whence the same notation for atomic systems and programs.

The hierarchy of atomic system provided by Piecha and Schroeder-Heister \cite{29,21} (Definition 2.3) precisely corresponds to the inductive depth of the grammar for hereditary Harrop formulas. Of course, in the Sanqvist basis, we are limited to second-level atomic systems, but the grammar of definite clauses can handle considerably more. Indeed, the work below suggests that completeness holds for $n$th-level atomic systems for $n \geq 2$. In the proof of completeness we shall make essential use of rules with hypotheses, which are second-level, so that we cannot comment on the case for $n = 0$ or $n = 1$.

Formally, to say that bases are definitional in the sense of programs, we mean the following:

\[
\models_{\mathcal{B}} \varphi \quad \text{iff} \quad \mathcal{N} \cup \mathcal{B} \vdash \varphi^\flat
\]

We assume for this equivalence that $\neg^\flat$ is sensitive to the presence of $\mathcal{B}$ so that $\varphi^\flat$ does not occur in $\mathcal{B}$. The flattening of $\varphi$ is essential; that is, it is certainly not the case that bases behave exactly as contexts; that is, we do not have the following equivalence:

\[
\models_{\mathcal{B}} \varphi \quad \text{iff} \quad \mathcal{B} \vdash \varphi \quad \text{(*)}
\]
Example 3.5. A counter-example to (⋆) is provided by the following formula:

\[ \varphi := (a \rightarrow b \lor c) \rightarrow ((a \rightarrow b) \lor (a \rightarrow c)) \]

The formula \( \varphi \) is not a consequence of IL; hence, by completeness, \( \not\vdash_{\mathcal{B}} (a \rightarrow b \lor c) \lor (a \rightarrow c) \), for some \( \mathcal{B} \). However, under (⋆), we have \( \vdash_{\mathcal{B}} (a \rightarrow b \lor c) \) implies \( \vdash_{\mathcal{B}} (a \rightarrow b) \lor (a \rightarrow c) \), for any \( \mathcal{B} \), contradicting completeness. To see this, we reasoning as follows:

\[
\begin{align*}
\vdash_{\mathcal{B}} a \rightarrow b \lor c & \quad \text{implies} \quad \mathcal{B} \vdash a \rightarrow b \lor c \quad \text{(⋆)} \\
& \quad \text{implies} \quad \mathcal{B} \cup \{a\} \vdash b \lor c \quad \text{(LOAD)} \\
& \quad \text{implies} \quad \mathcal{B} \cup \{a\} \vdash b \lor \mathcal{B} \cup \{a\} \vdash c \quad \text{(OR)} \\
& \quad \text{implies} \quad \mathcal{B} \vdash a \rightarrow b \lor \mathcal{B} \vdash a \rightarrow c \quad \text{(LOAD)} \\
& \quad \text{implies} \quad \mathcal{B} \vdash (a \rightarrow b) \lor (a \rightarrow c) \quad \text{(OR)} \\
& \quad \text{implies} \quad \vdash_{\mathcal{B}} (a \rightarrow b) \lor (a \rightarrow c) \quad \text{(⋆)}
\end{align*}
\]

In the next section, we use the relationship between atomic system \( s \) and programs to prove completeness of IPL with respect to the B-eS.

3.3. Completeness of IPL via Logic Programming

We may prove completeness of IPL with respect to the B-eS by passing through hHLP as follows:

\[
\begin{array}{c}
T^\omega I_1, \mathcal{N} \models \varphi^b \\
\vdash_{\mathcal{N}} \varphi \\
\vdash \varphi
\end{array}
\]

The diagram requires three claims, the middle one of which is Lemma 3.4. The other two are proved below.

The intuition of the completeness argument is two-fold: firstly, that \( \mathcal{N} \) is to \( \varphi^b \) as \( \mathcal{N} \) is to \( \varphi \); secondly, the use of a rule in a base corresponds to the use of a clause in the corresponding program; thirdly, execution in \( \mathcal{N} \) corresponds to proof-search in \( \mathcal{N} \). In this set-up, the \( T^\omega \) construction captures the construction of a proof: the application of a rule corresponds to a use of \( T \), the iterative application of rules corresponds to the iterative application of \( T \) — that is, to \( T^\omega \).
Example 3.6. Consider the judgement \( p \land q \vdash p \lor q \). We illustrate how the LP perspective delivers\( p \land q \vdash N \vdash p \lor q \).

By Definition 2.6, \( p \land q \vdash N \vdash p \lor q \), where \( N \) is as in Section 2.3. Therefore, for any \( N' \supseteq N \),

\[
\vdash N' \vdash p \land q \vdash p \lor q
\]

Heuristically, the rôle of \( N' \) is to capture the validity of \( p \land q \); therefore, choose \( N' = N \cup \{(p \land q)^b\} \) with the intuition that \( (p \land q)^b \) in \( N \) has the same inferential semantics as \( p \land q \) in NJ. That is, \( N \) is designed such that \( N \cup \{(p \land q)^b\} \) captures the proof-theoretic validity of \( p \land q \).

Indeed, by Lemma 2.8, we have \( p \vdash N \cup \{(p \land q)^b\} \). Hence, by \( (**) \), we have \( p \vdash p \lor q \). By Definition 2.6, for any \( N'' \supseteq N \cup \{(p \land q)^b\} \) and any \( r \in A \),

\[
p \vdash p \lor q \quad \text{and} \quad q \vdash p \lor q \quad \text{implies} \quad \vdash N'' \vdash (p \lor q)
\]

We compact this statement to the following, equivalent, judgement:

\[
\vdash N' \cup \{(p \land q)^b\} \quad (p \rightarrow r) \land (q \rightarrow r) \rightarrow r
\]

At this point, we could proceed to unfold the support judgement using the clauses of Figure 1. Yet, since we regard the base \( N \cup \{(p \land q)^b\} \) to have the same inferential content as its corresponding program, we may appeal to the LP perspective:

\[
T \omega I_L, N \cup \{(p \land q)^b\} \models \{(p \rightarrow r) \land (q \rightarrow r) \rightarrow r)^b
\]

Thus, we require the conditions for \( (p \rightarrow r) \land (q \rightarrow r) \rightarrow r)^b \) in \( T \omega I_L, N \cup \{(p \land q)^b\} \). By the definitions of \( N \) and \( T \), we require either \( T \omega I_L, N \cup \{(p \land q)^b\} \models p \) or \( T \omega I_L, N \cup \{(p \land q)^b\} \models q \). By construction of the program and of \( T \), we indeed have \( p, q \in T \omega I_L (N \cup \{(p \land q)^b\} \).

It follows that we have, \( T \omega I_L, N \cup \{(p \land q)^b\} \models (p \lor q)^b \). By Lemma 3.4, there is an execution witnessing \( N \cup \{(p \land q)^b\} \vdash (p \lor q)^b \). Tracing the execution yields the following derivations witnessing \( p \land q \vdash p \lor q \), as required:

\[
\begin{array}{ccc}
p \land q & p \land q & p \land q \\
p \lor q & p \lor q & p \lor q \\
\end{array}
\]
It remains to prove the claims and completeness:

**Lemma 3.7 (Emulation).** If $\models_{\mathcal{N}} \varphi$, then $T^\omega I_\perp, \mathcal{N} \models \varphi^\circ$.

**Proof:** We prove a stronger proposition: for any $\mathcal{N}' \supseteq \mathcal{N}$, if $\models_{\mathcal{N}'} \varphi$, then $T^\omega I_\perp, \mathcal{N}' \models \varphi^\circ$. We proceed by induction on the proof-theoretic structure of $\varphi$.

- $\varphi \in \mathcal{A}$. Note $\varphi^\circ = \varphi$, by definition. Therefore, if $\models_{\mathcal{N}} \varphi$, then $\models_{\mathcal{N}'} \varphi$, but this is precisely emulated by application of $T$. Hence, $T^\omega I_\perp, \mathcal{N} \models \varphi$. 

- $\varphi = \bot$. If $\models_{\mathcal{N}} \bot$, then $\models_{\mathcal{N}'} p$, for every $p \in \mathcal{A}$. By the induction hypothesis (IH), $T^\omega I_\perp, \mathcal{N} \models p$ for every $p \in \mathcal{A}$. Hence, by the $\bot$-clause for satisfaction, $T^\omega I_\perp, \mathcal{N} \models \bot$. 

- $\varphi := \varphi_1 \land \varphi_2$. By the $\land$-clause for support, $\models_{\mathcal{N}} \varphi_1$ and $\models_{\mathcal{N}} \varphi_2$. Hence, by the IH, $T^\omega I_\perp, \mathcal{N} \models \varphi_1$ and $T^\omega I_\perp, \mathcal{N} \models \varphi_2$. The result follows by $\land$-clause for satisfaction. 

- $\varphi := \varphi_1 \lor \varphi_2$. By the $\lor$-clause for support, $\models_{\mathcal{N}} (\varphi_1 \rightarrow p) \land (\varphi_2 \rightarrow p) \rightarrow p$ for any $p \in \mathcal{A}$. Let $A := ((\varphi_1 \rightarrow p) \land (\varphi_2 \rightarrow p) \rightarrow p)^\circ \in \mathcal{N}$. By the IH, $T^\omega I_\perp, \mathcal{N} \models A$. By the $\land$-clause for satisfaction, $A \in T^\omega I_\perp(\mathcal{N})$. Hence, there is $G$ such that $(G \rightarrow A) \in [\mathcal{N}]$ and $T^\omega I_\perp, \mathcal{N} \models G$. By analysis on the construction of $\mathcal{N}$, it must be that $G = ((\varphi_1 \rightarrow p) \land (\varphi_2 \rightarrow p))^\circ \rightarrow p$. By Definition 3.3, $T^\omega I_\perp, \mathcal{N} \cup \{\varphi_1 \rightarrow p, (\varphi_2 \rightarrow p)\} \models p$, so $p \in T^\omega I_\perp(\mathcal{N} \cup \{(\varphi_1 \rightarrow p), (\varphi_2 \rightarrow p)\})$. Therefore, by the definition of $T$, either $T^\omega I_\perp, \mathcal{N} \models \varphi^\circ$ or $T^\omega I_\perp, \mathcal{N} \models \psi^\circ$. The result follows by the construction of $\mathcal{N}$ and the definition of $T$. 

- $\varphi := \varphi_1 \rightarrow \varphi_2$. By the $\rightarrow$-clause for satisfaction, $\varphi \models_{\mathcal{N}} \psi$. So, by the $\rightarrow$-clause for satisfaction, $\models_{\mathcal{N}'} \varphi_1$ implies $\models_{\mathcal{N}'} \varphi_2$ for any $\mathcal{N}' \supseteq \mathcal{N}$. In particular, let $\mathcal{N}' := \mathcal{N} \cup \{\varphi_1^\circ\}$. Since, by Lemma 2.8, we have $\models_{\mathcal{N}'} \varphi_1$, we have $\models_{\mathcal{N}'} \varphi_2$. By the IH, $T^\omega I_\perp, \mathcal{N} \cup \{\varphi_1^\circ\} \models \varphi_2^\circ$. Hence, $T^\omega I_\perp, \mathcal{N} \models \varphi_1^\circ \rightarrow \varphi_2^\circ$. By construction of $\mathcal{N}$, we have $(\varphi_1^\circ \rightarrow \varphi_2^\circ) \rightarrow (\varphi_1 \rightarrow \varphi_2)^\circ \in \mathcal{N}$. Therefore, by definition of $T$, we have $(\varphi_1 \rightarrow \varphi_2)^\circ \in T(T^\omega I_\perp(\mathcal{N}))$. Whence, $T^\omega I_\perp, \mathcal{N} \models (\varphi_1 \rightarrow \varphi_2)^\circ$, as required.

This completes the induction. \(\square\)
Lemma 3.8 (Simulation). If \( \mathcal{N} \vdash \varphi^b \), then \( \vdash \varphi \).

Proof: In the course of unfolding \( \mathcal{N} \vdash \varphi^b \), we shall require storing definite clauses in the program; that is, we shall consider queries of the form \( \mathcal{N} \cup \Gamma^b \vdash \varphi^b \). Therefore, we will use a stronger proposition as an induction invariant: if \( \mathcal{N} \cup \Gamma^b \vdash \varphi^b \), then \( \Gamma \vdash \varphi \). Intuitively, the execution of \( \mathcal{N} \cup \Gamma^b \vdash \varphi^b \) simulates the reductive construction of a proof of \( \varphi \) in NJ. We proceed by induction on the length of the execution.

Base Case: It must be that \( \varphi \in \Gamma \), so \( \Gamma \vdash \varphi \) is immediate.

Inductive Step: By construction of \( \mathcal{N} \), the execution concludes by \textsc{Clause} applied to a definite clause \( \rho \) simulating a rule \( r \in \text{NJ} \); that is, \( \mathcal{N}, \Gamma^b \vdash \psi_i^b \) for \( \psi_i \) such that \( \psi_1^b \wedge \ldots \wedge \psi_n^b \rightarrow \varphi^b \). By the induction hypothesis (IH), \( \Gamma \vdash \psi_i \) for \( 1 \leq i \leq n \). It follows that \( \Gamma \vdash \varphi \) by applying \( r \in \text{NJ} \).

For example, if the execution concludes by \textsc{Clause} applied to the clause for \( \wedge \)-introduction (i.e., \( \varphi^b \wedge \psi^b \rightarrow (\varphi \wedge \psi)^b \)), then the trace is as follows:

\[
\begin{array}{c}
\vdash \varphi^b \\
\vdash \psi^b \\
\hline
\vdash \varphi^b \wedge \psi^b \\
\hline
\vdash (\varphi \wedge \psi)^b
\end{array}
\]

By the induction hypothesis, we have proofs witnessing \( \vdash \varphi \) and \( \vdash \psi \), and by \( \wedge \)-introduction:

\[
\begin{array}{c}
\vdash \varphi \\
\vdash \psi \\
\hline
\vdash \varphi \wedge \psi
\end{array}
\]

This completes the induction.

Proof: Theorem 2.7 — Completeness. By definition, if \( \vDash \varphi \), then \( \vDash \mathcal{N} \varphi \). Hence, by Lemma 3.7, it follows that \( T^\omega I_{\perp}, \mathcal{N} \vDash \varphi^b \); whence, by Lemma 3.4 \( \mathcal{N} \vdash \varphi^b \); whence, by Lemma 3.8, \( \vdash \varphi \).

In the following section, we discuss how reductive logic delivers the completeness proof above and the essential role played by both proofs and refutations.
3.4. Negation-as-Failure

A reduction in a proof system is constructed co-recursively by applying the rules of inference backward. Even though each step corresponds to the application of a rule, the reduction can fail to be a proof as the computation arrives at an irreducible sequent that is not an instance of an axiom in the logic. For example, in LJ, one may compute the following:

\[
\begin{align*}
&\frac{p : q}{p : p \lor q} \\
&\emptyset : p \to (p \lor q)
\end{align*}
\]

This reduction fails to be a proof, despite every step being a valid inference, since the initial sequent is not an axiom of LJ. In reductive logic, such failed attempts at constructing proofs are not meaningless. Pym and Ritter [22] have provided a semantics of the reductive logic of IPL in which such reduction are given meaning by using hypothetical rules; that is, the construction would succeed in the presence of the following rule:

\[
\frac{p}{\bot}
\]

The categorical treatment of this semantics has them as indeterminates in a polynomial category; this adumbrates current work by Pym et al. [23], who have shown that the B-eS is entirely natural from the perspective of categorical logic. The use of such additional rules to give semantics to constructions that are not proofs directly corresponds to the use of atomic systems in the B-eS for IPL. Let \( \mathcal{A} \) be the atomic system containing the rule above, we simply witness that the judgement \( p \vdash_{\mathcal{A}} q \) obtains. In summary, the analysis of Section 3.3 can be viewed as saying that IPL is autobiographic with its reductive logic; more precisely, that B-eS captures the reductive logic by using atomic systems as indeterminates. We may review the meaning of absurdity (\( \bot \)) in this context.

There is no rule constructing \( \bot \) in LJ. One may not construct a proof of absurdity without it already being, in some sense, assumed; for example, \( \varphi, \varphi \to \bot \vdash \bot \) obtains only because the context \( \{ \varphi, \varphi \to \bot \} \) is already, in some sense, absurd. We may use B-eS and LP to understand precisely what that sense is. The judgement \( \Gamma \vdash \bot \) is equivalent to \( \vdash \varphi \to \bot \) for some formula \( \varphi \). Therefore, we may restrict attention to negations of this kind to understand the meaning of absurdity. Using the work of Section 3.3, the
judgement \( \vdash \neg \varphi \) obtains iff \( T^\omega I_\bot, \mathcal{N} \vdash (\neg \varphi)^\flat \). Unfolding the semantics, this is equivalent to \( T^\omega I_\bot, \mathcal{N} \cup \{ \varphi \} \vdash \bot \). Thus, the sense in which \( \varphi \) is absurd is that its interpretation under \( T^\omega I_\bot \) contains a absurdity; that is, \( \varphi \) is absurd iff \( \bot \in T^\omega I_\bot (\varphi) \).

What does this tell us about the meaning of \( \neg \varphi \)? We are passing through the following equivalence:

\[
\vdash_{\mathcal{B}} \bot \iff \mathcal{N} \cup \mathcal{B} \vdash \bot^b
\]

Recall that \( \mathcal{B} \) is finite in this setting. Hence, according to the LP perspective, what we mean by a base supporting absurdity is that it proves \( \bot^b \). In this way, we introduce negation at the level of atomic propositions. That is, we may have have a base \( \mathcal{B} \) containing the following rules in which \( p \) and \( \bar{p} \) are both atoms:

\[
\frac{p \quad \bar{p}}{\bot^b}
\]

In this case, the inferential behaviour of \( p \) and \( \bar{p} \) is that they are contradictory propositions: together they infer absurdity. Essentially, following the construction of \( \mathcal{N} \) in Section 3.3, we have \( p = \varphi^\flat \) and \( \bar{p} = (\varphi \rightarrow \bot)^\flat \), for some \( \varphi \).

This view of negation is in contrast to the semantics, originally proposed by Dummett [4], in which the proof-theoretic meaning of absurdity is that all propositional are proved; that is, the definition in which \( \bot \) is understood by the following ‘virtually infinite’ rule:

\[
\frac{P_1 \quad ... \quad P_n}{\bot}
\]

Kürbis [16] observes that this leaves something be desired; for example, as Dummett [4] remarks, defined in this way, ‘negation lacks a feature possessed by all the other standard logical constants,’ which is that its meaning changes as the language changes!

The case in which a base proves every atomic proposition is degenerate because it corresponds to having every proof be valid. In the non-degenerate case, we may simply choose \( \bot^b \) to be an atom that does not appear in \( \mathcal{N} \cup \mathcal{B} \). Thus, the proof-theoretic meaning of \( \bot \) is the failure to find a proof of \( \bot \) while not working in a degenerate program.

It follows, by the clauses of Figure 1, that the meaning of \( \neg \varphi \) is that
there is no proof of $\varphi$ while not working in a degenerate program,

$$\vdash B \neg \varphi \iff \varphi \vdash \bot \iff \vdash \varphi \implies \bot \quad (\text{for } C \supseteq B)$$

$$\vdash N \cup C \not\vdash \varphi \quad (\text{unless } B \text{ degenerate})$$

Thus, B-eS supports negation-as-failure. In particular, since $N$ simulates NJ, the failure actually refers to failure to find a proof in the natural deduction system for IPL, even under extension by atomic rules, and not merely to the failure of hHLP to find a proof.

Piecha and Schroeder-Heister [29, 21] have argued that there are two perspectives on atomic systems: the knowledge view and the definitional view. This becomes clear according to various ways in which a program may be regarded in LP. The negation-as-failure protocol makes use of the definitional perspective; its analogue in terms of knowledge is the closed-world assumption. In this case, a knowledge base treats everything that is not known to be valid as invalid. There is a significant literature about closed-world assumption that may be useful for understanding P-tS and what it tells us about reasoning — see, for example, Clark [3], Reiter [24], and Kowalski [14, 13], and Harland [11, 12].

4. Conclusion

Proof-theoretic semantics is the paradigm of meaning based on proof (as opposed to truth). Essential to this approach is the use of atomic systems which give meaning to atomic propositions. Base-extension semantics is a particular instance of proof-theoretic semantics that proceeds by an inductively defined judgement whose base case is given by provability in an atomic system. It may be regarded as capturing the declarative content of proof-theoretic semantics in the Dummett-Prawitz tradition — see Gheorghiu and Pym [7]. Sandqvist [26] has given a base-extension semantics for intuitionistic propositional logic. Completeness follows by constructing a bespoke special base in which the validity of a complex proposition simulates a natural deduction proof of that formula.

Logic programming is a use of logic as a reasoning technology by using proof-search as the operational semantics for a programming language. In this context, a program is a set of formulas that can be interpreted as a knowledge base capturing the inferential behaviour of a proposition. As
such, atomic systems and programs are intimately related in that logic programming has been used to understand phenomena within proof-theoretic semantics — see Schroeder-Heister and Hallnäs [9, 10]. Indeed, there are quite general connections between proof-search and proof-theoretic semantics — see, for example, Gheorghiu and Pym [8].

In this paper, we have used logic programming to give a proof of the completeness of intuitionistic propositional logic with respect to its base-extension semantics. The aforementioned special base is interpreted as a program so that completeness follows immediately from the existing completeness result of the model-theoretic semantics of the logic programming language. Doing this reveals the subtle meaning of negation in proof-theoretic semantics.

Historically, the negation of a formula is understood as the denial of the formula itself. This is indeed the case in the model-theoretic semantics of IPL — see Kripke [15]. Using the connection to logic programming in this paper, we see that in base-extension semantics, negation is defined by the failure for their to be a proof. Thus, denial is conceptionally prior to negation. In short, we see that base-extension semantics consider the space of reductions, which is larger than the space of proofs, including also failed searches.

The connection between logic programming and base-extension semantics is quite intuitive and, as illustrated above, useful. More specifically, the $T$ operator delivering the semantics of logic programming corresponds to the application of a rule in a proof system; hence, the $T^\omega$ construction is fundamental to proof-theoretic semantics. Since logic programming has been studied for a variety of logics (see, for example, the treatment of BI in Gheorghiu et al. [6]), this suggests the possibility for uniform approaches to setting up base-extension semantics for logics by studying their proof-search behaviours. In particular, work by Harland [11, 12] on handling negation in logic programming may be used to address the difficulties posed by the connective — see Kürbis [16].

It remains to investigate further the connection between logic programming and base-extension semantics, in particular, and reductive logic and proof-theoretic semantics, in general.

Acknowledgements. We are grateful to Edmund Robinson for suggesting the formula in Example 3.5.
References

[1] K. R. Apt, M. H. Van Emden, *Contributions to the theory of logic programming*, Journal of the ACM (JACM), vol. 29(3) (1982), pp. 841–862.

[2] R. Brandom, *Articulating Reasons: An Introduction to Inferentialism*, Harvard University Press (2000).

[3] K. L. Clark, *Negation as Failure*, [in:] *Logic and Data Bases*, Springer (1978), pp. 293–322.

[4] M. Dummett, *The Logical Basis of Metaphysics*, Harvard University Press (1993).

[5] G. Frege, *Die Verneinung. Eine Logische Untersuchung*, Beiträge Zur Philosophie des Deutschen Idealismus, vol. 1(3/4) (1919), pp. 143–157.

[6] A. V. Gheorghiu, S. Docherty, D. J. Pym, *Reductive Logic, Coalgebra, and Proof-search: A Perspective from Resource Semantics*, [in:] A. Palmigiano, M. Sadrzadeh (eds.), *Samson Abramsky on Logic and Structure in Computer Science and Beyond*, Springer Outstanding Contributions to Logic Series, Springer (2021), to appear.

[7] A. V. Gheorghiu, D. J. Pym, *From Proof-theoretic Validity to Base-extension Semantics for Intuitionistic Propositional Logic*, http://www.cs.ucl.ac.uk/staff/D.Pym/From_Proof-theoretic_Viability_to_Base_extension_Semantics_for_IL.pdf (Accessed 15 August 2022), submitted.

[8] A. V. Gheorghiu, D. J. Pym, *Proof-theoretic Semantics and Tactical Proof*, http://www.cs.ucl.ac.uk/staff/D.Pym/Proof-theoretic_Semantics_and_Tactical_Proof.pdf (Accessed 15 August 2022), submitted.

[9] L. Hallnäs, P. Schroeder-Heister, *A Proof-theoretic Approach to Logic Programming: I. Clauses as Rules*, Journal of Logic and Computation, vol. 1(2) (1990), pp. 261–283.

[10] L. Hallnäs, P. Schroeder-Heister, *A Proof-theoretic Approach to Logic Programming: II. Programs as Definitions*, Journal of Logic and Computation, vol. 1(5) (1991), pp. 635–660.

[11] J. Harland, *On Hereditary Harrop Formulae as a Basis for Logic Programming*, Ph.D. thesis, The University of Edinburgh (1991).

[12] J. Harland, *Success and Failure for hereditary Harrop Formulae*, The Journal of Logic Programming, vol. 17(1) (1993), pp. 1–29.
[13] R. Kowalski, *Logic for Problem Solving*, https://www.doc.ic.ac.uk/~rak/papers/LFPScommentary.pdf (Accessed 15 August 2022), commentary on the book ‘Logic for Problem Solving’ by R. Kowalski.

[14] R. Kowalski, *Logic for Problem-Solving*, North-Holland Publishing Co. (1986).

[15] S. A. Kripke, *Semantical Analysis of Intuitionistic Logic I*, [in:] *Studies in Logic and the Foundations of Mathematics*, vol. 40, Elsevier (1965), pp. 92–130.

[16] N. Kürbis, *Proof and Falsity: A Logical Investigation*, Cambridge University Press (2019).

[17] J. W. Lloyd, *Foundations of Logic Programming*, Symbolic Computation, Springer-Verlag (1984).

[18] D. Makinson, *On an Inferential Semantics for Classical Logic*, *Logic Journal of IGPL*, vol. 22(1) (2014), pp. 147–154.

[19] D. Miller, *A Logical Analysis of Modules in Logic Programming*, *Journal of Logic Programming*, vol. 6(1-2) (1989), pp. 79–108.

[20] D. Miller, G. Nadathur, F. Pfenning, A. Scedrov, *Uniform Proofs as a Foundation for Logic Programming*, *Annals of Pure and Applied Logic*, vol. 51(1) (1991), pp. 125 – 157.

[21] T. Piecha, P. Schroeder-Heister, *The Definitional View of Atomic Systems in Proof-theoretic Semantics*, [in:] *The Logica Yearbook 2016*, College Publications London (2017), pp. 185–200.

[22] D. J. Pym, E. Ritter, *Reductive logic and Proof-search: Proof Theory, Semantics, and Control*, vol. 45 of Oxford Logic Guides, Oxford University Press (2004).

[23] D. J. Pym, E. Ritter, E. Robinson, *Proof-theoretic Semantics in Sheaves (Extended Abstract)*, [in:] *Proceedings of the Eleventh Scandinavian Logic Symposium — SLSS 11* (2022), pp. 36–38.

[24] R. Reiter, *On closed world data bases*, [in:] *Readings in artificial intelligence*, Elsevier (1981), pp. 119–140.

[25] T. Sandqvist, *Atomic Bases and the Validity of Peirce’s Law*, https://drive.google.com/file/d/1fX8PWhs8w2cp0kYS39zR2OGfNEhQESKkl/view (Accessed 15 August 2022), presentation at the World
Logic Day event at UCL: The Meaning of Proofs (https://sites.google.com/view/wdl-ucl2022/home).

[26] T. Sandqvist, An Inferentialist Interpretation of Classical Logic, Ph.D. thesis, Uppsala University (2005).

[27] T. Sandqvist, Classical Logic without Bivalence, Analysis, vol. 69(2) (2009), pp. 211–218.

[28] T. Sandqvist, Base-extension Semantics for Intuitionistic Sentential Logic, Logic Journal of the IGPL, vol. 23(5) (2015), pp. 719–731.

[29] P. Schroeder-Heister, T. Piecha, Atomic Systems in Proof-Theoretic Semantics: Two Approaches, [in:] Ángel Nepomuceno Fernández, O. P. Martins, J. Redmond (eds.), Epistemology, Knowledge and the Impact of Interaction, Springer Verlag (2016), pp. 47–62.

[30] M. E. Szabo (ed.), The Collected Papers of Gerhard Gentzen, North-Holland Publishing Company (1969).

Alexander V. Gheorghiu
University College London
Department of Computer Science
Gower St, London WC1E 6BT
London, United Kingdom
e-mail: alexander.gheorghiu.19@ucl.ac.uk

David J. Pym
University College London
Department of Computer Science
Gower St, London WC1E 6BT
London, United Kingdom

University College London
Department of Philosophy
Gower St, London WC1E 6BT
London, United Kingdom

University of London
Institute of Philosophy
Senate House, Malet St, London WC1E 7HU
London, United Kingdom
e-mail: d.pym@ucl.ac.uk