SUPPORT IS SEARCH

ALEXANDER V. GHEORGHIU

Abstract. For intuitionistic propositional logic, Sandqvist's base-extension semantics defines a support relation parametrised by atomic bases, with validity identified as support in every base. Sandqvist's completeness theorem answers the *global* question: which formulae are valid? This paper addresses the *local* question: given a fixed base, what does support in that base correspond to? We show that support in a fixed base coincides with proof-search in a hereditary Harrop logic program with a quantifier over atoms, via an encoding of formulae as logic programming goals. The encoding is a variant of the Russell–Prawitz translation of the connectives, with quantification restricted to atoms, and it is obtained by reading the semantic clauses in continuation-passing style. So read, the universal quantifiers over base extensions and atoms in those clauses need not range over a completed totality: they are eigenvariables under a standard freshness discipline. Base-extension semantics thereby acquires an explicit computational content: support is proof-search. The result complements Sandqvist's global theorem with a local correspondence and makes the constructive character of the framework explicit.

1. Introduction

The semantics of the logical constants admits, broadly, two styles of explanation. On the first, model-theoretic semantics, meaning is given by the conditions under which a sentence is "true" in a structure. On the second, proof-theoretic semantics, meaning is given by the inferential behaviour that constitutes a demonstration of it; the tradition is that of Prawitz [32, 33], Dummett [3, 1, 2], and Schroeder-Heister [40, 41, 42]. The latter has received less attention, and its computational character in particular remains largely unexamined. This paper examines that character in a central case.

The case is the *base-extension semantics* for intuitionistic propositional logic due to Sandqvist [39], a particularly successful realisation of proof-theoretic semantics. A *base* $\mathcal{B}$ is a collection of rules governing atomic sentences from which one defines a derivability relation on atoms $\vdash_{\mathcal{B}}$. From this one has an inductively defined *support* relation $\Vdash_{\mathcal{B}}$ on formulae, the clauses of which define each connective. The clause for falsum, for example, is

$$\Vdash_{\mathcal{B}} \bot \quad \text{iff} \quad \Vdash_{\mathcal{B}} A \text{ for every atom } A.$$

The clause is rather demanding computationally. The atoms may be infinitely many, and a verification of $\Vdash_{\mathcal{B}} \bot$ accordingly comprises infinitely many verifications — and this within a programme whose founding commitment is that the meaning of a sentence is exhausted by what one can do with it. The constructivist has a

2020 *Mathematics Subject Classification.* 03B20, 68N17, 03F50.

Key words and phrases. proof-theoretic semantics, base-extension semantics, intuitionistic propositional logic, logic programming, hereditary Harrop formulae, uniform proofs.

standard reply: a universal claim is a claim to a method, and to say that every atom is supported is to say that one has, for any given atom, a way of showing that it is. What is that method? The clause does not say. That is the question of this paper.

The clause for disjunction is more remarkable still. It is *not* the expected

$$\Vdash_{\mathcal{B}} \varphi \vee \psi \quad \text{iff} \quad \Vdash_{\mathcal{B}} \varphi \text{ or } \Vdash_{\mathcal{B}} \psi$$

— which would yield not intuitionistic but inquisitive logic, which is strictly stronger (see Piecha et al. [27, 26, 30]; Stafford [43]) — but rather the 'second-order condition'

$$\begin{aligned} \Vdash_{\mathcal{B}} \varphi \vee \psi \quad \text{iff} \quad & \text{for every } \mathcal{C} \sqsupseteq \mathcal{B} \text{ and every atom } A, \\ & \text{if } \varphi \Vdash_{\mathcal{C}} A \text{ and } \psi \Vdash_{\mathcal{C}} A, \text{ then } \Vdash_{\mathcal{C}} A, \end{aligned}$$

where $\mathcal{C} \sqsupseteq \mathcal{B}$ means that $\mathcal{C}$ extends $\mathcal{B}$ by further rules, and $\varphi \Vdash_{\mathcal{C}} A$ (resp. $\psi \Vdash_{\mathcal{C}} A$) means that A is supported in every extension of $\mathcal{C}$ in which φ (resp. ψ) is. Whether $A_1 \vee A_2$ is supported in $\mathcal{B}$ — where A_1 and A_2 are atoms — thus depends upon every atom of the language, including those bearing no relation whatever to A_1 or A_2, and upon every extension of $\mathcal{B}$. Whether that demand is met seems, *prima facie*, rather difficult to verify.

The two clauses raise a single question. Sandqvist's theorem is that $\Vdash_{\mathcal{B}} \varphi$ for every base $\mathcal{B}$ if and only if φ is intuitionistically provable. The result is striking, but it is also *global*: it characterises the formulae supported in every base. It leaves untouched what is, from a foundational point of view, the prior question — *what does $\Vdash_{\mathcal{B}} \varphi$ amount to in a fixed base $\mathcal{B}$?*

The naive answer has much to recommend it. At atoms, support simply is derivability: $\Vdash_{\mathcal{B}} A$ holds precisely when A is derivable by the rules of $\mathcal{B}$. Support is monotone in the base, as derivability is in its hypotheses; the clause for implication, $\Vdash_{\mathcal{B}} \varphi \to \psi$ iff $\varphi \Vdash_{\mathcal{B}} \psi$, has the form of a deduction theorem; and a base, being a set of rules, may be read as a set of hypotheses. One expects, accordingly, that $\Vdash_{\mathcal{B}} \varphi$ should coincide with derivability of φ from some natural translation of $\mathcal{B}$ into a set of formulae, and that the universal quantifiers of the clauses are merely a circuitous way of saying as much.

The expectation is refuted by an example that Sandqvist gives [39]. Fix atoms A_1 and A_2, and let $\mathcal{R}$ consist of the rules

$$\frac{\begin{array}{cc}[A_1] & [A_2] \\ A & A\end{array}}{A}$$

one for each atom A: from A-if-A_1 and A-if-A_2, infer A. The disjunction $A_1 \vee A_2$ is supported in $\mathcal{R}$: in any extension in which A follows from A_1 and from A_2, the rule for A concludes it outright. Yet neither $\Vdash_{\mathcal{R}} A_1$ nor $\Vdash_{\mathcal{R}} A_2$ holds: a derivation of A_1 in $\mathcal{R}$ could end only with the rule for A_1, one of whose premisses is a derivation of A_1 from A_2, which could in turn end only with that same rule, and the same premiss remains outstanding; the case of A_2 is symmetric.

Derivability cannot behave in this way. The rules of $\mathcal{R}$, read as formulae, are $\vee$-free — Harrop formulae — and from Harrop hypotheses a disjunction is derivable only if one of its disjuncts is [17]. What fails is the principle Piecha and Schroeder-Heister call *export* [30]: that the rules of a base may be exported into hypotheses, so that $\Vdash_{\mathcal{B}\cup\Sigma} \varphi$ just in case $\Sigma^* \Vdash_{\mathcal{B}} \varphi$, where Σ^* is the set of rules of Σ read as formulae, and hence, taking $\mathcal{B}$ empty, that $\Vdash_{\Sigma} \varphi$ just in case $\Sigma^* \vdash \varphi$. The lesson is that support in a base is not simply derivability in disguise.

We propose an alternative reading. The clause for disjunction does not say when a disjunction is supported; it says what one may *do* with one — namely, conclude any atom that each disjunct would yield on its own. It has, in this respect, a recognisable form in functional programming as a *continuation-passing* definition: the atom A is the answer sought, and the hypotheses $\varphi \Vdash_{\mathcal{C}} A$ and $\psi \Vdash_{\mathcal{C}} A$ are the continuations — the branches of a case analysis, one of which will deliver it. A type theorist will recognise in the clause the type

$$\forall \rho \, ((\alpha \to \rho) \to (\beta \to \rho) \to \rho)$$

that a sum acquires under continuation-passing conversion. A logician will recognise it as Russell's definition of disjunction [36, §19] and Prawitz's second-order encoding of the connectives [34]. The recognition matters for what it suggests: that the clauses may be read computationally after all, and that the place to read them is a paradigm at once logical and computational — *logic programming*.

Logic programming, in the tradition of Kowalski [19] and, in the form relevant here, of Miller et al. [24, 23], is the view of computation as proof-search. A *program* is a set of formulae of restricted form — *program clauses* — and a *goal* is a formula whose provability from the program is in question; to *execute* the goal is to search for a proof of it, and the search is driven by the logical structure of the goal, each connective being an instruction as to how to proceed. A conjunction instructs one to establish both conjuncts; a disjunction, to establish either; an implication $D \supset G$, to add D to the program and establish G; an atom, to consult the program. The program is thus consulted only at atoms, and the connectives of the goal are operations of search.

The bearing on the present problem is immediate: a base is a set of rules for atoms — that is, a program — and the support clauses, read as above, are themselves goals: instructions as to how a formula is to be verified against it. The instructions are not always those of the like-named connectives. Support for a disjunction does not require support for either disjunct, and the clause for disjunction is accordingly not a disjunctive goal; it is a universally quantified implication, which instructs one to take an arbitrary atom, to assume that it follows from each disjunct, and to establish it.

In logic programming, moreover, a universal quantifier in a goal is not a claim about a totality at all but an instruction: introduce a fresh name and proceed [24]. If the quantifiers of Sandqvist's clauses admit this reading, the infinities disappear, and each support clause says precisely what a verification consists in — which is what was asked for.

To each formula φ we associate a goal $[\![\varphi]\!]$ of the propositional hereditary Harrop language of Miller et al. [24], extended by a quantifier over atoms, and we establish that, for every base $\mathcal{B}$,

$$\Vdash_{\mathcal{B}} \varphi \quad \text{iff} \quad \mathcal{B} \vdash_{O} [\![\varphi]\!],$$

where $\vdash_O$ is the operational semantics of the language — the relation that obtains when $[\![\varphi]\!]$, executed as a goal against the program $\mathcal{B}$, succeeds (Theorem 4.2). The encoding is a variant of the Russell–Prawitz translation, and the proof pairs each universal quantifier of the semantic clauses with a step of proof-search: the quantifiers over base extensions become Augment, which adds a hypothesis to the program, and the quantifiers over atoms become Generic, which introduces an

atom under the usual freshness condition and proceeds. The constructive reading promised a method; the theorem exhibits it. Support is search.

The correspondence is not to be had for nothing, and three points require care. The encoding of an implication, $[\![\varphi]\!] \supset [\![\psi]\!]$, must serve both as a goal and as a clause of a program, and its consequent is in general compound; the clausal language must accordingly admit program clauses with compound heads, as the hereditary Harrop formulae of Miller et al. [24] do, such clauses being resolved into clauses with atomic heads when the program is consulted (Definition 2.6).

The quantifier of the language is instantiated at atoms only. It is the quantifier of Ferreira and Ferreira's atomic polymorphism [4], not that of second-order logic. So the substitution lemma one would naturally reach for is false: a universal goal may be derivable while a compound instance of it is not (Remark 4.5).

The proof therefore proceeds by renaming of atoms, together with identity and cut for the class of formulae that are at once program clauses and goals; it follows the method of Miller's argument that his Kripke models of hereditary Harrop programs agree with uniform provability [21, 22]. What it adds is the treatment of Sandqvist's clauses for $\vee$ and $\bot$, which are elimination conditions quantifying over atoms and not the satisfaction clauses of a Kripke model, and of the atomic quantifier that answers to them.

The result is a local correspondence theorem complementing Sandqvist's global one. It has three consequences.

First, the semantic clauses need not be regarded as ranging over completed infinite collections of inferential contexts. Their content is exhausted by a finitary search procedure — a successful search being a finite derivation. Proof-theoretic semantics, anti-realist in motivation, calls for a constructive reading. That reading is supplied not by external argument but by the structure of the semantics itself.

Second, support acquires computational content of a definite kind. The relation $\Vdash_{\mathcal{B}} \varphi$ is not merely constructive in some abstract sense; it is the input–output specification of an explicit procedure — execute $[\![\varphi]\!]$ as a goal against $\mathcal{B}$ — whose successful runs are uniform derivations witnessing support. The semantics is, in this sense, executable; and since, by Sandqvist's theorem, φ is intuitionistically provable just in case $[\![\varphi]\!]$ succeeds against the empty program (Corollary 4.9), so too is intuitionistic provability.

Third, and most broadly, the result places proof-theoretic semantics and the proof theory of logic programming in unexpected proximity. Sandqvist's construction was developed independently of, and later than, Mints' resolution-based proof methods for intuitionistic logic [25]; we have shown that the two constructions coincide up to renaming [6]. The present paper extends this convergence: it is not only Sandqvist's particular base that is computationally intelligible, but the semantic framework in which it is constructed. What was offered as a philosophical foundation turns out to be, in mathematical content, a chapter of computational logic.

Organisation. Section 2 develops the framework of atomic systems and hereditary Harrop programs with a quantifier over atoms, and recalls the correspondence between them established by Gheorghiu and Pym [11]. Section 3 recalls Sandqvist's base-extension semantics, the relevant background results, and the local question. Section 4 introduces the encoding of intuitionistic formulae as logic programming goals and proves the main theorem. Section 5 discusses the implications of the result — in particular for the question of what the universal quantifiers in base-extension

semantics mean, for the role of higher-level rules, and for the prospects of a uniformly computational reading of proof-theoretic semantics more broadly.

2. Atomic Systems and Clausal Programs

Base-extension semantics rests on a primitive notion of atomic inference: before any account is given of the logical constants, one must first say what it is for an atomic sentence to be derivable. This is the role of an *atomic system*, or *base*. The purpose of this section is to fix the framework of atomic systems we shall use, and to develop a computational presentation of it. Two moves are involved. First, following Prawitz [32, 33] and Piecha and Schroeder-Heister [28, 29], we admit atomic rules of arbitrary finite level — rules whose premisses may themselves be rules. Second, following Hallnäs and Schroeder-Heister [15, 16] and Miller et al. [20, 24], we re-present these rules as clauses of a logic programming language, with derivability in the atomic system corresponding to operational proof-search in the clausal program.

The first move lets the semantics characterise intuitionistic rather than classical logic: Sandqvist has shown that first-level rules alone yield classical logic [38], while rules of level two suffice for the intuitionistic case [39]. We admit rules of arbitrary finite level for uniformity with the clausal language. The second move makes the main theorem of the paper possible. Both are well established in the literature, but they are not customarily presented together, and the combination repays explicit treatment.

Fix an infinite set At of *atoms*, or atomic sentences, ranged over by $A, B, C, \ldots$. The infinity at issue is *potential*, not actual: at any stage of a derivation only finitely many atoms have been introduced, but no bound is placed on how many may be introduced later. This is the only notion of infinity admissible from an anti-realist standpoint, and the only one we shall use — it presupposes no completed totality, only the open-ended availability of fresh names as derivation proceeds.

Higher-level atomic systems. The simplest atomic systems consist of *production rules*: rules that license the inference of an atom from a finite collection of atoms.

Definition 2.1 (First-level atomic system)**.** A *first-level atomic system* $\mathcal{S}$ is a set of rules of the form

$$\frac{A_1 \quad \cdots \quad A_n}{C}$$

in which $A_1, \ldots, A_n$ and C are atoms. Derivability in $\mathcal{S}$ — written $\vdash_{\mathcal{S}}$ — is the least relation closed under the rules.

This format is, however, too restrictive. Rules in natural deduction for the logical constants — implication introduction and disjunction elimination, in particular — are not of this shape: their premisses are not assertions but *derivations from hypotheses*. If the atomic level is to support a semantics flexible enough to deliver intuitionistic logic, it must admit rules of the same kind. The generalisation, due to Piecha and Schroeder-Heister [28, 29], allows the premisses of an atomic rule to be themselves finite sets of atomic rules; iterating yields rules of arbitrary finite level.

Definition 2.2 (Higher-level atomic system)**.** An *atomic rule of level* 0 is an axiom

$$\frac{}{B},$$

where B is an atom. An *atomic rule of level* 1 is a production rule

$$\frac{A_1 \quad \cdots \quad A_n}{B}$$

on atoms. For $n \geq 1$, an *atomic rule of level* $n+1$ has the form

$$\frac{\begin{array}{ccc}[\Sigma_1] & & [\Sigma_k] \\ A_1 & \ldots & A_k\end{array}}{B}$$

where B and each A_i are atoms and each Σ_i is a finite set of atomic rules of level at most $n-1$. Thus a rule of level 2 discharges atoms, a rule of level 3 discharges production rules, and so on. The rule is read as licensing the conclusion B whenever each A_i is derivable from the rules in Σ_i, with those rules thereby discharged. A *higher-level atomic system* is a set of atomic rules of level at most n for some fixed n, and $\vdash_{\mathcal{S}}$ is again the least relation closed under the rules.

Example 2.3 (A third-level rule)**.** Let $\mathcal{S}$ contain the level-3 rule

$$\frac{\left[\dfrac{A}{B}\right]}{\dfrac{C}{D}}$$

together with the level-2 rule

$$\frac{\begin{array}{c}[A] \\ B\end{array}}{C}$$

The first rule licenses D whenever C is derivable using the production rule from A to B, that rule being discharged. The second rule licenses C whenever B is derivable from the assumption A. With the production rule from A to B in hand, it is; so C is derivable using that rule, and the first rule yields D in $\mathcal{S}$.

Rules as clauses. The presentation of atomic systems we have just given is faithful to their natural deduction origins, but for the purposes of this paper a different presentation is more convenient. We re-present atomic rules as clauses of a logic programming language, and atomic derivability as operational proof-search. The translation, which we use throughout, is due in essence to Hallnäs and Schroeder-Heister [15, 16].

The clausal language is the propositional fragment of the *hereditary Harrop* formulae of Miller et al. [20, 24].

Definition 2.4 (Definite formulae, goals, programs)**.** The language is given by mutual recursion. *Definite formulae* D, which appear in programs, and *goal formulae* G, which are queried against them, are generated by the grammar

$$D ::= A \mid G \supset D \mid D \cap D \qquad\qquad G ::= A \mid D \supset G \mid G \cap G \mid G \cup G,$$

where A ranges over atoms. A *program* P is a finite set of definite formulae; a *query* is a pair of a program P and a goal G. We adopt the convention that $\cap$ and $\cup$ bind more tightly than $\supset$, so that $G_1 \cap G_2 \supset D$ abbreviates $(G_1 \cap G_2) \supset D$. In a definite formula $G \supset D$, the formula D is the *head*.

FACT	$P \vdash_O A$	if $A \in [P]$
BACKCHAIN	$P \vdash_O A$	if $G \supset A \in [P]$ and $P \vdash_O G$, for some G
AND	$P \vdash_O G_1 \cap G_2$	if $P \vdash_O G_1$ and $P \vdash_O G_2$
OR	$P \vdash_O G_1 \cup G_2$	if $P \vdash_O G_1$ or $P \vdash_O G_2$
AUGMENT	$P \vdash_O D \supset G$	if $P \cup \{D\} \vdash_O G$
GENERIC	$P \vdash_O \forall X\, G$	if $P \vdash_O G[A/X]$, for some A fresh for P and G

FIGURE 1. Rules of search. The rule GENERIC belongs to the extension by the atomic quantifier (Definition 2.10).

Remark 2.5. We use the signs $\supset$, $\cap$, $\cup$ for the connectives of the clausal language in order to reserve the more standard $\to$, $\wedge$, $\vee$ for the object logic. They are to be read intuitively as *operational* implication, conjunction, and disjunction, governed by the proof-search rules of Definition 2.6. The emphasis on 'operational' as opposed to 'logical' is crucial: it is logic itself that is here under scrutiny, and were the clausal connectives understood logically, the explanation would move in a circle. The reader who finds this distinction elusive is referred to Dummett [1] for its philosophical articulation. In this sense the clausal language is *pre-logical* — a calculus of inference about atoms, on which a semantics for the logical constants will subsequently be erected.

The asymmetry between D and G is the source of the language's good proof-theoretic behaviour: definite formulae allow conjunction but not disjunction; goal formulae allow both, but their implications are *hypothetical* — the antecedent is a definite formula, added to the program for the duration of the goal.

The intended reading of a query is procedural: to resolve G against P, decompose G according to its logical structure, consulting P only when an atomic goal is reached. This is captured by the following operational semantics.

Definition 2.6 (Operational semantics)**.** The relation $\vdash_O$ between programs and goals is defined by the rules FACT, BACKCHAIN, AND, OR, and AUGMENT of Figure 1. Here $[P]$, the *elaboration* of P, is the least set of definite formulae that contains P and satisfies the following conditions:

- if $D_1 \cap D_2 \in [P]$, then $D_1, D_2 \in [P]$;
- if $G \supset (D_1 \cap D_2) \in [P]$, then $G \supset D_1$, $G \supset D_2 \in [P]$;
- if $G \supset (G' \supset D) \in [P]$, then $G \cap G' \supset D \in [P]$.

The formulae of $[P]$ of the forms A and $G \supset A$ are the *resolved clauses* of P.

Remark 2.7. Though Definition 2.6 is stated deductively, the relation is best thought of as *reductive* in the sense of Pym and Ritter [35]. A query is a putative conclusion $P \vdash_O G$. The rules of the operational semantics mean that we can resolve the query by case analysis on the structure of the goal G. Thus the goal drives a proof-search, and the program is consulted only at atoms, where one either appeals to a fact or backchains through an implication.

The elaboration $[P]$ discharges the bookkeeping of definite conjunctions and of compound heads: every clause of the program is resolved into clauses with atomic

heads, $G \supset (G' \supset A)$ doing the work of $G \cap G' \supset A$, and FACT and BACKCHAIN consult only these. Each condition on $[P]$ has a single premiss, so elaboration distributes over union: $[P \cup Q] = [P] \cup [Q]$. Derivations in $\vdash_O$ are, in the terminology of Miller et al. [24], *uniform*: the goal is decomposed before the program is consulted.

The connection between atomic systems and clausal programs is now direct: each atomic rule translates to a program clause, and the level of the rule corresponds to the nesting depth of implications in that clause.

Example 2.8 (Rules as clauses)**.** The level-2 rule

$$\frac{\begin{array}{c}[A]\\ B\end{array}}{C}$$

becomes the program clause $(A \supset B) \supset C$. The reading is the same in both notations: to derive C, it suffices to derive B from the assumption A. The operational semantics confirms this. With $P = \{(A \supset B) \supset C\}$, BACKCHAIN reduces $P \vdash_O C$ to $P \vdash_O A \supset B$, and AUGMENT reduces this to $P, A \vdash_O B$:

$$\frac{P, A \vdash_O B}{P \vdash_O C}$$

Gheorghiu and Pym [11] have shown that the hierarchy of higher-level atomic systems corresponds exactly to the implication-nesting depth of definite formulae: an atomic rule of level n becomes a definite formula of nesting depth n, and conversely. For any atomic system $\mathcal{S}$ with clausal representation P, and any atom A,

$$P \vdash_O A \qquad \text{iff} \qquad \vdash_{\mathcal{S}} A.$$

Clauses with compound heads add convenience, not strength: the clausal representation of an atomic rule has an atomic head, and a clause with a compound head is, by the elaboration, a notation for clauses with atomic heads. Compound heads are admitted because Section 4 requires implications to be at once program clauses and goals whatever their consequents. We shall henceforth identify an atomic system with its clausal representation, writing $\mathcal{S} \vdash_O A$ in place of $\vdash_{\mathcal{S}} A$.

Schematic infinity and the atomic quantifier. One complication remains. The atomic systems used in base-extension semantics are routinely *infinite*; indeed, the completeness of intuitionistic propositional logic with respect to support requires this. But not every infinite system is to be admitted. This is not a technical convenience but a substantive constraint: the anti-realist foundations of the semantics forbid appeal to arbitrary completed infinities. The infinities of base-extension semantics must be *schematic*: each instance of a rule must be specifiable by a finite description.

A representative case illustrates the point. Sandqvist's completeness proof [39] uses an atomic system containing every instance of the schema

$$(A \supset X) \cap (B \supset X) \supset X$$

for some fixed atoms A and B as X ranges over atoms. The system is infinite, but its content is exhausted by a finite specification. We need only formalise that finite specification. To this end, we extend the logic programming language with a quantifier ranging over atoms. Propositional variables, ranged over by X, are drawn from a set Var disjoint from the atoms.

Definition 2.9 (Atomic quantifier)**.** The grammar of definite formulae and goals is extended by the productions

$$G \;::=\; \ldots \mid \forall X\, G \qquad\qquad D \;::=\; \ldots \mid \forall X\, D,$$

where X is a propositional variable.

The schema above becomes the single program clause

$$\forall X\big((A \supset X) \cap (B \supset X) \supset X\big),$$

and the move from infinite to finite specification is complete.

The *atomic formulae* of the extended language are the atoms together with the variables. We write $G[A/X]$ for the result of replacing the free occurrences of X in G by the atom A, and we say that an atom is *fresh* for a formula or program if it does not occur in it. Programs, and the goals queried against them, are *closed*: every variable occurring in them is bound. In a closed formula, therefore, an atomic subformula standing alone as a goal or as a program clause is an atom.

The quantifier is not the quantifier of second-order propositional logic, nor the higher-order quantifier of Miller et al. [24], either of which may be instantiated at compound formulae of the appropriate type. It is instantiated at atoms only, and is in this respect the quantifier of the system $\mathbf{F}_{\mathrm{at}}$ of atomic polymorphism studied by Ferreira and Ferreira [4, 5]; the point is taken up in Remark 4.5. Its operational behaviour is governed by the eigenvariable discipline.

Definition 2.10 (Operational rules for $\forall$)**.** The relation $\vdash_O$ is extended by the rule Generic of Figure 1. The elaboration of programs is extended correspondingly: if $\forall X\, D \in [P]$, then $D[A/X] \in [P]$ for every atom A; and if $G \supset \forall X\, D \in [P]$, then $G \supset D[A/X] \in [P]$ for every atom A.

Remark 2.11. Were one to require that the body of a $\forall$-goal hold for *every* atomic instantiation, the infinitary problem would simply recur, displaced from the program to the proof. The freshness condition replaces this universal quantification with a single schematic instance — a witness that, because nothing in the program or goal mentions it, stands in for any atom whatever. This is finitary and mechanical, and it is the device that makes the schematic infinities of the semantics tractable. We shall see in Section 4 that the universal quantifiers in the support clauses are governed by this same discipline.

We close the section by recording three properties of $\vdash_O$ which will be used freely in what follows. Each is proved by induction on the derivation, and each preserves the height of the derivation, a fact on which the inductions of Section 4 depend. A function σ from atoms to atoms is extended to formulae homomorphically, leaving variables unchanged, and to programs pointwise; we write $F\sigma$ and $P\sigma$ for the results. Observe that $[P]\sigma \subseteq [P\sigma]$: the map σ commutes with the conditions that define the elaboration, and in particular it carries the instance $D[A/X]$ of $\forall X\, D$ to the instance $(D\sigma)[A\sigma/X]$ of $\forall X\, (D\sigma)$.

Lemma 2.12 (Renaming)**.** *Let σ be any function from atoms to atoms. If $P \vdash_O G$, then $P\sigma \vdash_O G\sigma$.*

Proof. By induction on the height of the derivation of $P \vdash_O G$, the statement being taken for all σ at once. For Fact, $A \in [P]$ gives $A\sigma \in [P]\sigma \subseteq [P\sigma]$. For Backchain, $G' \supset A \in [P]$ and $P \vdash_O G'$; the induction hypothesis gives $P\sigma \vdash_O G'\sigma$,

and $G'\sigma \supset A\sigma \in [P\sigma]$. The cases And, Or, and Augment are immediate from the induction hypothesis, the last because $(P \cup \{D\})\sigma = P\sigma \cup \{D\sigma\}$. For Generic, $G = \forall X\, G'$ and $P \vdash_O G'[A/X]$ for some atom A fresh for P and G'. Choose an atom B fresh for $P\sigma$ and $G'\sigma$, and let σ' agree with σ except that $\sigma'(A) = B$. The induction hypothesis applied with σ' gives $P\sigma' \vdash_O G'[A/X]\sigma'$. Since A occurs in neither P nor G', we have $P\sigma' = P\sigma$ and $G'[A/X]\sigma' = (G'\sigma)[B/X]$. As B is fresh for $P\sigma$ and $G'\sigma$, Generic gives $P\sigma \vdash_O \forall X\, (G'\sigma) = G\sigma$. □

The case of interest is that in which σ moves a single atom. If $P \vdash_O G$ and the atom A does not occur in P, then $P \vdash_O G[B/A]$ for every atom B, where $G[B/A]$ denotes the result of replacing A by B throughout G. An eigenvariable introduced by Generic may thus be moved to any atom whatever, provided the program does not mention it. This is the only form of substitution the paper requires; see Remark 4.5.

We write $Q \sqsupseteq P$ to mean that $Q \supseteq P$, that is, that Q extends P as a program.

Lemma 2.13 (Monotonicity)**.** *If $P \vdash_O G$ and every resolved clause of P is a resolved clause of Q, then $Q \vdash_O G$. In particular, if $P \vdash_O G$ and $Q \sqsupseteq P$, then $Q \vdash_O G$.*

Proof. The proof is by induction on the height of the derivation. The cases Fact and Backchain hold by the hypothesis on resolved clauses, and the cases And and Or are immediate. For Augment, elaboration distributes over union, so the resolved clauses of $P \cup \{D\}$ are those of P together with those of $\{D\}$, and all are resolved clauses of $Q \cup \{D\}$; the induction hypothesis applies. For Generic, $P \vdash_O G'[A/X]$ with A fresh for P and G', but A may occur in Q. Choose B fresh for Q and G'. By Lemma 2.12, $P \vdash_O G'[B/X]$ by a derivation of the same height, so the induction hypothesis gives $Q \vdash_O G'[B/X]$, and Generic concludes. □

A conjunction $D_1 \cap D_2$ is not itself a resolved clause, and $[D_1 \cap D_2] = \{D_1 \cap D_2\} \cup [D_1] \cup [D_2]$. The programs $P \cup \{D_1 \cap D_2\}$ and $P \cup \{D_1, D_2\}$ therefore have the same resolved clauses, and the lemma gives

$$P, D_1 \cap D_2 \vdash_O G \qquad \text{iff} \qquad P, D_1, D_2 \vdash_O G.$$

Lemma 2.14 (Inversion)**.** *The following statements hold:*

(i) If $P \vdash_O D \supset G$, then $P, D \vdash_O G$.
(ii) If $P \vdash_O G_1 \cap G_2$, then $P \vdash_O G_i$ for $i = 1, 2$.
(iii) If $P \vdash_O \forall X\, G$, then $P \vdash_O G[A/X]$ for every atom A.

Proof. The rules Fact and Backchain conclude only atomic goals, so the last rule of a derivation of a compound goal is determined by the principal connective of the goal, and (i) and (ii) follow by reading off its premisses. For (iii), the last rule is Generic, with premiss $P \vdash_O G[A'/X]$ for some A' fresh for P and G. Given any atom A, Lemma 2.12 applied to the renaming $A' \mapsto A$ gives $P \vdash_O G[A'/X][A/A'] = G[A/X]$, since A' occurs in neither P nor G. □

Finally, $\vdash_O$ is closed under cut on atoms: if $P \vdash_O A$ and $P, A \vdash_O G$, then $P \vdash_O G$. The proof is by induction on the height of the derivation of $P, A \vdash_O G$. The atom A contributes only itself to $[P, A]$, so it can be used only by Fact, to conclude A, and there the derivation of $P \vdash_O A$ serves instead; at Augment, that derivation is first weakened by Lemma 2.13. Lemma 4.8 extends the property to a wider class of formulae.

$\Vdash_{\mathcal{B}} A$	iff	$\mathcal{B} \vdash_O A$	(At)
$\Vdash_{\mathcal{B}} \varphi \to \psi$	iff	$\varphi \Vdash_{\mathcal{B}} \psi$	$(\to)$
$\Vdash_{\mathcal{B}} \varphi \wedge \psi$	iff	$\Vdash_{\mathcal{B}} \varphi$ and $\Vdash_{\mathcal{B}} \psi$	$(\wedge)$
$\Vdash_{\mathcal{B}} \varphi \vee \psi$	iff	for every $\mathcal{C} \sqsupseteq \mathcal{B}$ and every $A \in \mathrm{At}$, if $\varphi \Vdash_{\mathcal{C}} A$ and $\psi \Vdash_{\mathcal{C}} A$, then $\Vdash_{\mathcal{C}} A$	$(\vee)$
$\Vdash_{\mathcal{B}} \bot$	iff	$\Vdash_{\mathcal{B}} A$ for every $A \in \mathrm{At}$	$(\bot)$
$\Gamma \Vdash_{\mathcal{B}} \varphi$	iff	for every $\mathcal{C} \sqsupseteq \mathcal{B}$, if $\Vdash_{\mathcal{C}} \psi$ for every $\psi \in \Gamma$, then $\Vdash_{\mathcal{C}} \varphi$	(Inf)

FIGURE 2. Support in a base.

3. Base-extension Semantics

We turn now to the semantics itself. The atomic systems of Section 2 are the ground on which the logical constants are given meaning. The semantics is due, in the form we present, to Sandqvist [39], and we record his global completeness theorem before stating the local question that motivates the present paper.

The language of intuitionistic propositional logic is given by

$$\varphi, \psi, \chi \ ::= \ A \in \mathrm{At} \ \mid \ \bot \ \mid \ \varphi \wedge \psi \ \mid \ \varphi \vee \psi \ \mid \ \varphi \to \psi,$$

with $\neg\varphi \coloneqq \varphi \to \bot$. We write $\Gamma \vdash \varphi$ for intuitionistic derivability of φ from Γ.

The semantics is parametrised by atomic systems. In view of the clausal presentation of Section 2, we identify atomic systems with logic programs throughout.

Definition 3.1 (Base)**.** A *base* $\mathcal{B}$ is a program in the clausal language of Section 2.

This is not quite Sandqvist's class of bases. His are arbitrary sets of atomic rules of level at most two [39]; ours are finite programs, whose rules may be of any finite level and which admit infinite families of rules only through the quantifier. The results quoted from [39] below — Propositions 3.4 and 3.5 and Theorem 3.8 — are stated there for his class. Their proofs transfer unchanged, because they use only two facts about the class of bases: that a base extended by atoms or rules is again a base, and, for completeness, that the flattening base $\mathcal{N}$ constructed below belongs to the class. The base $\mathcal{N}$, in particular, consists of finitely many rule schemas, a few for each subformula of the given formula, and each schema is a single clause of that language; Gheorghiu and Pym [11] carry out the construction for clausal programs.

The meaning of the logical constants is given by an inductively defined relation between bases and formulae — the *support* relation — whose clauses we now state.

Definition 3.2 (Support)**.** The relation $\Gamma \Vdash_{\mathcal{B}} \varphi$ is defined by the clauses of Figure 2, in which A ranges over At and Γ is a finite set of formulae; we write $\varphi \Vdash_{\mathcal{B}} \psi$ for $\{\varphi\} \Vdash_{\mathcal{B}} \psi$.

Remark 3.3. The clauses of Figure 2 contain phrases of the form 'for every $A \in \mathrm{At}$' — in particular, the clause for falsum reads

$$\Vdash_{\mathcal{B}} \bot \qquad \text{iff} \qquad \Vdash_{\mathcal{B}} A \text{ for every } A \in \mathrm{At}.$$

What does this mean when At is infinite only potentially, not actually? Read set-theoretically, the phrase ranges over a completed totality of atoms. Read constructively, it claims a method which, given any atom, establishes that the atom is supported; and this reading is available, since nothing in the clauses requires the set-theoretic one. But the clause does not say what the method is.

Read in the spirit of Section 2, it does: 'for every A' is understood as 'for an arbitrary A', an arbitrary atom being represented by an eigenvariable, and the clause for falsum becomes

$$\Vdash_{\mathcal{B}} \bot \qquad \text{iff} \qquad \mathcal{B} \vdash_O A \quad \text{for some atom } A \text{ fresh for } \mathcal{B},$$

where freshness means that A does not occur in $\mathcal{B}$. The two readings agree on which formulae are supported; they differ in what a verification consists in. On the first, it is whatever establishes the universal claim; on the second, it is a single derivation together with a renaming argument (Lemma 2.12). Section 4 shows that the second reading is available for every clause, not only for falsum.

Two structural properties of support are used throughout. The first is immediate from the definition. The second is nearly so. From left to right, it is clause (Inf) at the base $\mathcal{B} \cup S$, together with clause (At). From right to left, it requires in addition that $\vdash_O$ be closed under cut on atoms, as recorded at the end of Section 2: if $P \vdash_O B$ for every $B \in S$ and $P, S \vdash_O A$, then $P \vdash_O A$.

Proposition 3.4 (Monotonicity [39])**.** *If $\Vdash_{\mathcal{B}} \varphi$ and $\mathcal{C} \sqsupseteq \mathcal{B}$, then $\Vdash_{\mathcal{C}} \varphi$.*

Proposition 3.5 (Atomic adequacy [39])**.** *For every atom A and every finite set of atoms S,*

$$S \Vdash_{\mathcal{B}} A \qquad \textit{iff} \qquad \mathcal{B} \cup S \vdash_O A.$$

A third property is not needed for the proof of the main theorem, but bears on its interpretation in Section 4. It says that every formula is characterised by what it licenses one to conclude, and it generalises a reformulation of the clause for conjunction observed by Gheorghiu et al. [9]. Two consequences of clause (Inf) are used in its proof and thereafter without comment: if $\Gamma \Vdash_{\mathcal{C}} \varphi$ and $\mathcal{D} \sqsupseteq \mathcal{C}$, then $\Gamma \Vdash_{\mathcal{D}} \varphi$; and if $\Gamma \Vdash_{\mathcal{C}} \varphi$ and $\Vdash_{\mathcal{C}} \psi$ for every $\psi \in \Gamma$, then $\Vdash_{\mathcal{C}} \varphi$, by taking the extension in (Inf) to be $\mathcal{C}$ itself.

Lemma 3.6 (Elimination form)**.** *For every base $\mathcal{B}$ and formula ϑ,*

$$\Vdash_{\mathcal{B}} \vartheta \qquad \textit{iff} \qquad \textit{for every } \mathcal{C} \sqsupseteq \mathcal{B} \textit{ and every atom } A, \textit{ if } \vartheta \Vdash_{\mathcal{C}} A \textit{ then } \Vdash_{\mathcal{C}} A.$$

Proof. For the left-to-right direction, let $\mathcal{C} \sqsupseteq \mathcal{B}$ and A be given with $\vartheta \Vdash_{\mathcal{C}} A$. Proposition 3.4 gives $\Vdash_{\mathcal{C}} \vartheta$, whence $\Vdash_{\mathcal{C}} A$.

The right-to-left direction is by induction on ϑ, the statement being taken for all bases at once. Suppose that, for every $\mathcal{C} \sqsupseteq \mathcal{B}$ and atom A, $\vartheta \Vdash_{\mathcal{C}} A$ implies $\Vdash_{\mathcal{C}} A$.

Case $\vartheta = A'$ (atom). Since $A' \Vdash_{\mathcal{B}} A'$ holds by clause (Inf), the supposition with $\mathcal{C} = \mathcal{B}$ and $A = A'$ gives $\Vdash_{\mathcal{B}} A'$.

Case $\vartheta = \bot$. For any atom A, $\bot \Vdash_{\mathcal{B}} A$ holds by clause ($\bot$), so the supposition with $\mathcal{C} = \mathcal{B}$ gives $\Vdash_{\mathcal{B}} A$. Since A was arbitrary, clause ($\bot$) gives $\Vdash_{\mathcal{B}} \bot$.

Case $\vartheta = \varphi \wedge \psi$. We show $\Vdash_{\mathcal{B}} \varphi$ by the induction hypothesis for φ. Let $\mathcal{C} \sqsupseteq \mathcal{B}$ and A be given with $\varphi \Vdash_{\mathcal{C}} A$. Then $\varphi \wedge \psi \Vdash_{\mathcal{C}} A$: if $\mathcal{D} \sqsupseteq \mathcal{C}$ and $\Vdash_{\mathcal{D}} \varphi \wedge \psi$, clause ($\wedge$) gives $\Vdash_{\mathcal{D}} \varphi$, and $\varphi \Vdash_{\mathcal{D}} A$ gives $\Vdash_{\mathcal{D}} A$. The supposition now gives $\Vdash_{\mathcal{C}} A$. Hence $\Vdash_{\mathcal{B}} \varphi$; likewise $\Vdash_{\mathcal{B}} \psi$; and clause ($\wedge$) concludes.

Case $\vartheta = \varphi \vee \psi$**.** Let $\mathcal{C} \sqsupseteq \mathcal{B}$ and A be given with $\varphi \Vdash_{\mathcal{C}} A$ and $\psi \Vdash_{\mathcal{C}} A$; by clause $(\vee)$ it suffices to show $\Vdash_{\mathcal{C}} A$. Now $\varphi \vee \psi \Vdash_{\mathcal{C}} A$: if $\mathcal{D} \sqsupseteq \mathcal{C}$ and $\Vdash_{\mathcal{D}} \varphi \vee \psi$, then $\varphi \Vdash_{\mathcal{D}} A$ and $\psi \Vdash_{\mathcal{D}} A$, and clause $(\vee)$ applied at $\mathcal{D}$ gives $\Vdash_{\mathcal{D}} A$. The supposition gives $\Vdash_{\mathcal{C}} A$.

Case $\vartheta = \varphi \to \psi$**.** By clause $(\to)$ it suffices to show $\varphi \Vdash_{\mathcal{B}} \psi$. Let $\mathcal{C} \sqsupseteq \mathcal{B}$ with $\Vdash_{\mathcal{C}} \varphi$; we show $\Vdash_{\mathcal{C}} \psi$ by the induction hypothesis for ψ at the base $\mathcal{C}$. Let $\mathcal{D} \sqsupseteq \mathcal{C}$ and A be given with $\psi \Vdash_{\mathcal{D}} A$. Then $\varphi \to \psi \Vdash_{\mathcal{D}} A$: if $\mathcal{E} \sqsupseteq \mathcal{D}$ and $\Vdash_{\mathcal{E}} \varphi \to \psi$, Proposition 3.4 gives $\Vdash_{\mathcal{E}} \varphi$, clause $(\to)$ gives $\Vdash_{\mathcal{E}} \psi$, and $\psi \Vdash_{\mathcal{E}} A$ gives $\Vdash_{\mathcal{E}} A$. Since $\mathcal{D} \sqsupseteq \mathcal{B}$, the supposition gives $\Vdash_{\mathcal{D}} A$. □

Corollary 3.7. *For every base $\mathcal{B}$ and formulae φ, ψ:*

(i) $\Vdash_{\mathcal{B}} \varphi \wedge \psi$ *iff for every* $\mathcal{C} \sqsupseteq \mathcal{B}$ *and atom* A*, if* $\varphi, \psi \Vdash_{\mathcal{C}} A$ *then* $\Vdash_{\mathcal{C}} A$*;*
(ii) $\Vdash_{\mathcal{B}} \varphi \to \psi$ *iff for every* $\mathcal{C} \sqsupseteq \mathcal{B}$ *and atom* A*, if* $\Vdash_{\mathcal{C}} \varphi$ *and* $\psi \Vdash_{\mathcal{C}} A$ *then* $\Vdash_{\mathcal{C}} A$*.*

Proof. (i) Left to right: given $\mathcal{C}$ and A with $\varphi, \psi \Vdash_{\mathcal{C}} A$, Proposition 3.4 and clause $(\wedge)$ give $\Vdash_{\mathcal{C}} \varphi$ and $\Vdash_{\mathcal{C}} \psi$, whence $\Vdash_{\mathcal{C}} A$. Right to left: by Lemma 3.6 it suffices to show that $\varphi \wedge \psi \Vdash_{\mathcal{C}} A$ implies $\Vdash_{\mathcal{C}} A$ for every $\mathcal{C} \sqsupseteq \mathcal{B}$ and atom A. If $\varphi \wedge \psi \Vdash_{\mathcal{C}} A$, then $\varphi, \psi \Vdash_{\mathcal{C}} A$, since $\Vdash_{\mathcal{D}} \varphi$ and $\Vdash_{\mathcal{D}} \psi$ give $\Vdash_{\mathcal{D}} \varphi \wedge \psi$ by clause $(\wedge)$; the hypothesis then gives $\Vdash_{\mathcal{C}} A$.

(ii) Left to right: given $\mathcal{C}$ and A with $\Vdash_{\mathcal{C}} \varphi$ and $\psi \Vdash_{\mathcal{C}} A$, Proposition 3.4 gives $\Vdash_{\mathcal{C}} \varphi \to \psi$, clause $(\to)$ gives $\Vdash_{\mathcal{C}} \psi$, and $\psi \Vdash_{\mathcal{C}} A$ gives $\Vdash_{\mathcal{C}} A$. Right to left: let $\mathcal{C} \sqsupseteq \mathcal{B}$ with $\Vdash_{\mathcal{C}} \varphi$; by clause $(\to)$ it suffices to show $\Vdash_{\mathcal{C}} \psi$, and by Lemma 3.6 it suffices that $\psi \Vdash_{\mathcal{D}} A$ implies $\Vdash_{\mathcal{D}} A$ for every $\mathcal{D} \sqsupseteq \mathcal{C}$ and atom A. Proposition 3.4 gives $\Vdash_{\mathcal{D}} \varphi$; given $\psi \Vdash_{\mathcal{D}} A$, the hypothesis at $\mathcal{D} \sqsupseteq \mathcal{B}$ then gives $\Vdash_{\mathcal{D}} A$. □

Part (i) is the reformulation of Gheorghiu et al. [9]; part (ii) is the corresponding fact for implication. Together they show that every connective, and not only $\bot$ and $\vee$, admits a clause in elimination form.

The principal result of the framework is Sandqvist's theorem.

Theorem 3.8 (Sandqvist [39])**.** *For every formula φ,*

$\Vdash_{\mathcal{B}} \varphi$ *for every base* $\mathcal{B}$ *iff* φ *is provable in intuitionistic propositional logic.*

Soundness — the direction from right to left — is routine: each rule of natural deduction is shown to be respected by support, and the result follows by induction on derivations. Completeness — from left to right — is the substantive direction, and its proof is illuminating for what follows.

Sandqvist's argument proceeds by *flattening*. Fix a formula φ and let Ξ be its set of subformulae. Construct an injection $(-)^\flat \colon \Xi \to \mathrm{At}$ that is the identity on atoms ($A^\flat = A$ for $A \in \Xi \cap \mathrm{At}$) and that associates a fresh atom to each complex subformula. From this, one constructs a base $\mathcal{N}$ with two key properties:

(†) $\chi^\flat$ and χ are semantically interchangeable in any extension $\mathcal{N}' \sqsupseteq \mathcal{N}$:

$$\Vdash_{\mathcal{N}'} \chi \qquad \text{iff} \qquad \mathcal{N}' \vdash_O \chi^\flat;$$

(††) derivability in $\mathcal{N}$ simulates provability in intuitionistic propositional logic, written $\vdash$:

$$\mathcal{N} \vdash_O \varphi^\flat \qquad \text{iff} \qquad \vdash \varphi.$$

Global completeness follows immediately:

$$\begin{aligned} \Vdash_{\mathcal{B}} \varphi \text{ for all } \mathcal{B} \quad &\Longrightarrow \quad \Vdash_{\mathcal{N}} \varphi \\ &\Longrightarrow \quad \mathcal{N} \vdash_{O} \varphi^{\flat} \qquad (\dagger) \\ &\Longrightarrow \quad \vdash \varphi. \qquad (\dagger\dagger) \end{aligned}$$

The argument is elementary, constructive, and versatile, having since been deployed to establish completeness for a range of logics [7, 12, 9, 10].

Sandqvist's theorem is global. It leaves untouched the local question of the introduction: what $\Vdash_{\mathcal{B}} \varphi$ amounts to in a fixed base.

The question has been partly addressed. Gheorghiu and Pym [11] have generalised the flattening construction by relativising the operator $(-)^{\flat}$ to a given base $\mathcal{B}$: the fresh atom $\chi^{\flat}$ assigned to each complex subformula χ of φ is required not to occur in $\mathcal{B}$. The resulting simulation lemma is exactly local:

$$\Vdash_{\mathcal{B}} \varphi \qquad \text{iff} \qquad \mathcal{N} \cup \mathcal{B} \vdash_{O} \varphi^{\flat}.$$

The equivalence is a genuine local correspondence theorem, but it is purchased at a cost. The flattening operator absorbs the entire logical structure of φ into the base $\mathcal{N}$, replacing the formula with a single fresh atom $\varphi^{\flat}$. Whatever relation the connectives of φ bear to the proof-search procedure is invisible in the theorem itself: the procedure sees only an atom, and the structure of φ has been compiled away.

What is wanted, instead, is a correspondence in which the connectives of φ remain visible — in which the universal quantifiers of the support clauses correspond to identifiable steps of proof-search, and the procedure is driven by the logical structure of the formula rather than by a precomputed translation of it. The next section supplies it. The encoding is direct: each connective of φ is matched with a fixed pattern of the clausal language, and the universal quantifiers in the support clauses turn out to be the eigenvariables of proof-search in disguise.

4. The Encoding and the Main Theorem

The clauses of the support relation, read as they stand, define a procedure: given a formula and a base, decompose the formula according to its outermost connective, generate the clauses for each connective, and consult the base when an atomic case is reached. The procedure is, in everything but name, the operational semantics of a logic programming language. This section makes the identification exact. It begins with the shape of the support clauses.

The clause for disjunction is the following:

$$\begin{aligned} \Vdash_{\mathcal{B}} \varphi \vee \psi \quad \text{iff} \quad &\text{for every } \mathcal{C} \sqsupseteq \mathcal{B} \text{ and every atom } A, \\ &\text{if } \varphi \Vdash_{\mathcal{C}} A \text{ and } \psi \Vdash_{\mathcal{C}} A, \text{ then } \Vdash_{\mathcal{C}} A. \end{aligned}$$

The clause says what one may *do* with a disjunction: conclude any atom A that can be concluded from each disjunct separately. The atom A is an arbitrary conclusion which the disjunction is to deliver, and the disjuncts function as *continuations* — branches of a case analysis to which control is passed when A is sought. In this sense the clause is a continuation-passing definition. The reading has independent support: Gu et al. [14] interpret the minor premisses of disjunction elimination as continuations, the conclusion being the result of the computation, and analyse Sandqvist's completeness proof as an implicit CPS transformation.

The description calls for a word of justification, since continuation-passing style (CPS) is in the first instance a transformation on programs [31], and the semantics transforms nothing. What is intended is the type discipline that the transformation induces. Under CPS conversion a value of type α becomes a function that accepts a continuation of type $\alpha \to \rho$ and returns an answer of type ρ, so that the type α itself becomes $(\alpha \to \rho) \to \rho$; abstracting the answer type gives $\forall \rho\, ((\alpha \to \rho) \to \rho)$, and the image of a sum type $\alpha + \beta$ is $\forall \rho\, ((\alpha \to \rho) \to (\beta \to \rho) \to \rho)$. Griffin [13] observed that, read at the level of types, the conversion is the double-negation translation.

The clause $(\vee)$ is this type with the answer type restricted to atoms and the arrows read as consequence in a base. It is in this sense, and not in the sense of a transformation applied to source programs, that we call the clauses continuation-passing. The resulting encoding of the connectives by implication and a propositional quantifier is older than either reading: it is the Russell–Prawitz translation [36, §19], [34], and we use that name when the encoding is in question as a syntactic object rather than as a reading of the semantics.

The other connectives admit the same reading. Falsum is the trivial case: it passes *any* continuation, since $\Vdash_{\mathcal{B}} \bot$ licenses the conclusion of every atom. Sandqvist's clause for conjunction is direct, but Corollary 3.7(i) shows it to be equivalent to an elimination form of the same shape as $(\vee)$, an observation first made by Gheorghiu et al. [9]; and Corollary 3.7(ii) shows that implication, too, is characterised by what it licenses one to conclude: from $\varphi \to \psi$ together with φ, whatever follows from ψ. Every connective is thus given by its elimination behaviour. The encoding transcribes that behaviour where Sandqvist's clauses state it, for $\bot$ and $\vee$; for $\wedge$ and $\to$, whose clauses are direct, the encoding is direct also.

CPS-shaped definitions translate directly into clauses of a hereditary Harrop language. The case analysis on continuations becomes a conjunction of implications; the universal quantification over atoms becomes the quantifier over atoms; the procedure of trying the continuation against each disjunct becomes backchaining through the implications. We make this translation explicit.

Definition 4.1 (Encoding)**.** To each formula φ of intuitionistic propositional logic, assign a formula $[\![\varphi]\!]$ of the clausal language by induction on the structure of φ:

$$\begin{aligned}
[\![A]\!] &= A \\
[\![\bot]\!] &= \forall X\, X \\
[\![\varphi \to \psi]\!] &= [\![\varphi]\!] \supset [\![\psi]\!] \\
[\![\varphi \wedge \psi]\!] &= [\![\varphi]\!] \cap [\![\psi]\!] \\
[\![\varphi \vee \psi]\!] &= \forall X\, \big(([\![\varphi]\!] \supset X) \cap ([\![\psi]\!] \supset X) \supset X\big)
\end{aligned}$$

where X is a variable not occurring in $[\![\varphi]\!]$ or $[\![\psi]\!]$.

The encoding follows Sandqvist's clauses one by one. Conjunction and implication, whose clauses are direct, are encoded by the corresponding connectives of the clausal language; falsum and disjunction, whose clauses are elimination conditions quantifying over atoms, are encoded by quantified formulae of the same shape. For $\bot$, $\to$, and $\vee$ the result is the Russell–Prawitz translation. Conjunction, which that translation encodes in elimination form, is here taken directly, the clausal language having it as a primitive; for this reason we speak of a variant of that translation.

Nothing turns on the choice of the direct forms. Corollary 3.7 shows that Sandqvist's clauses for conjunction and implication are equivalent to their elimination forms, and the theorem holds equally for the corresponding encodings, $\forall X\,((\llbracket\varphi\rrbracket \cap \llbracket\psi\rrbracket \supset X) \supset X)$ and $\forall X\,(\llbracket\varphi\rrbracket \cap (\llbracket\psi\rrbracket \supset X) \supset X)$, by arguments modelled on the case of disjunction below. We do not need these alternatives, and omit the verification. The direct forms are chosen because they are the natural ones, and because they make the corresponding cases of the proof short.

The direct encoding of implication is the reason for which the clausal language admits compound clause heads. Whenever an implication occurs to the left of another, its encoding $\llbracket\varphi\rrbracket \supset \llbracket\psi\rrbracket$ must be installed in a base by Augment, and its head $\llbracket\psi\rrbracket$ is in general compound. It is the encoding adopted in the short version of this paper [8]. No substitution of compound formulae for atoms is involved in its treatment — a substitution which, as Remark 4.5 shows, is in any case unavailable.

The universal quantifier $\forall X$ in the clauses for $\bot$ and $\vee$ reflects, at the syntactic level, the universal quantification over atoms in the corresponding semantic clauses. Under the operational semantics of Section 2, X is instantiated only with atoms, and the instantiation is governed by the eigenvariable discipline of Generic: a single fresh witness suffices. The infinity that the set-theoretic reading of the support clauses appeared to require has become a single name-introduction step, standard in the proof theory of quantification.

The main theorem of the paper is that the encoding is faithful: to be supported in a base is for the encoded goal to succeed against that base as a program.

Theorem 4.2 (Support is Search)**.** *For every base $\mathcal{B}$ and every formula φ,*

$$\Vdash_{\mathcal{B}} \varphi \qquad \textit{iff} \qquad \mathcal{B} \vdash_{O} \llbracket\varphi\rrbracket.$$

The left-hand side is defined by quantification over every extension of the base and every atom; the right-hand side, by six rules of search. The theorem says that they are one relation.

Sandqvist's example from the introduction shows the theorem at work. As a program, the base $\mathcal{R}$ is the single program clause

$$\forall X\,((A_1 \supset X) \cap (A_2 \supset X) \supset X),$$

and this is the goal $\llbracket A_1 \vee A_2\rrbracket$ itself. The search for it succeeds at once: Generic introduces a fresh atom A, Augment adds $(A_1 \supset A) \cap (A_2 \supset A)$ to the program, and Backchain on the instance of the clause at A leaves, by And, the goals $A_1 \supset A$ and $A_2 \supset A$, each of which is closed by Augment, Backchain, and Fact.

The search for $\llbracket A_1\rrbracket = A_1$ does not succeed. The only resolved clause with head A_1 is the instance of the clause at A_1, and backchaining on it leaves, among others, the goal $A_2 \supset A_1$: that is, A_1 again, from a program in which that instance is still the only clause with head A_1. Every attempt regenerates its own goal, and there is no derivation. The disjunction is supported and neither disjunct is, and the search shows why.

The proof of the theorem requires four lemmas, to which we now turn.

Neutral formulae. The encodings produced by Definition 4.1 have a distinctive syntactic feature: the goal-only connective $\cup$ does not occur in them. They are therefore simultaneously definite formulae *and* goal formulae — a property we name and exploit.

Definition 4.3 (Neutral formulae)**.** A formula of the clausal language is *neutral* if it is generated by the grammar

$$N \;::=\; A \;\mid\; N \supset N \;\mid\; N \cap N \;\mid\; \forall X\, N,$$

where A ranges over atomic formulae.

Lemma 4.4 (Neutrality)**.** *The following statements hold:*

(i) Every neutral formula is both a definite formula and a goal formula.
(ii) For every formula φ, the encoding $[\![\varphi]\!]$ is neutral.

Proof. (i) By induction on the grammar. An atomic formula is both a definite formula and a goal. If N and N' are both, then $N \supset N'$ is a definite formula, being of the form $G \supset D$ with $G = N$ and $D = N'$, and a goal, being of the form $D \supset G$ with $D = N$ and $G = N'$. Conjunction and universal quantification are available to both classes.

(ii) By induction on φ. The cases of atoms and $\bot$ are immediate. If $[\![\varphi]\!]$ and $[\![\psi]\!]$ are neutral, then so are $[\![\varphi]\!] \supset [\![\psi]\!]$, $[\![\varphi]\!] \cap [\![\psi]\!]$, $[\![\varphi]\!] \supset X$, and $[\![\psi]\!] \supset X$; hence so is $([\![\varphi]\!] \supset X) \cap ([\![\psi]\!] \supset X) \supset X$; and universal quantification preserves neutrality. □

In particular, a base extended by the encoding of a formula is again a base, a fact used repeatedly below. The neutral formulae are exactly the formulae of the clausal language in which $\cup$ does not occur. The elaboration of a neutral formula consists of neutral formulae, each condition on $[P]$ preserving neutrality.

Before proving the encoding faithful, we ask how neutral formulae sit within the language. Atoms are the paradigm of formulae that are at once goals and program clauses, and neutral formulae extend that class; one might expect them to behave like atoms in every respect, and in particular to be substitutable for them. They are not, and the reason is instructive.

Remark 4.5 (Failure of compound instantiation). Consider the principle: if $P \vdash_O G$ and the atom A does not occur in P, then $P \vdash_O G[F/A]$ for every neutral formula F. Lemma 2.12 establishes this for atomic F; for compound F it fails. Let $P = \{\forall Y\, (Y \supset q)\}$ and $G = p \supset q$, where p, q, and a are distinct atoms. The query $P \vdash_O p \supset q$ succeeds: Augment reduces it to $P, p \vdash_O q$; the instance $p \supset q$ lies in $[P]$, and Backchain on it leaves $P, p \vdash_O p$, which Fact closes.

The instance at $F = a \supset a$ fails: Augment reduces $P \vdash_O (a \supset a) \supset q$ to $P, a \supset a \vdash_O q$, and the clauses of $[P \cup \{a \supset a\}]$ with head q are exactly the formulae $B \supset q$ with B an atom, so some atom must be derivable; the only heads available are q, which requires an atom in turn, and a, which requires a. Hence $P \nvdash_O (a \supset a) \supset q$, although, by Generic, $P \vdash_O \forall X\, (X \supset q)$.

The universal goal is thus derivable while one of its compound instances is not: the quantifier is genuinely atomic. The clausal language is, in this respect, the operational counterpart of the system $\mathbf{F}_{\mathrm{at}}$ of Ferreira and Ferreira [4], in which second-order quantification is likewise restricted to atomic instantiation, and the encoding of Definition 4.1 is a variant of the Russell–Prawitz translation by which intuitionistic propositional logic is embedded in that system. Corollary 4.9 below is, for the operational fragment, an analogue of the faithfulness of that embedding [5]. The methodological consequence for the present paper is that Theorem 4.2 cannot be proved by a substitution lemma. The proof below uses instead the renaming of atoms by atoms (Lemma 2.12), together with identity and cut for neutral formulae, to which we now turn.

Some of the behaviour of atoms does survive, and it is exactly the behaviour that proof-search relies on. The rule FACT, which closes an atomic goal found in the elaborated program, generalises to neutral formulae; so does the admissibility of cut. Throughout, the *size* of a formula is the number of occurrences of atomic formulae, connectives, and quantifiers in it; note that $F[A/X]$ has the same size as F, and that $G \cap G' \supset D$ has the same size as $G \supset (G' \supset D)$.

Lemma 4.6 (Identity)**.** *Let F be neutral. If $F \in [P]$, then $P \vdash_O F$. In particular, $P, F \vdash_O F$ for every program P.*

Proof. By induction on the size of F and, among implications $N \supset M$ of the same size, on the size of the head M.

Case $F = A$**.** Since F is closed, A is an atom, and FACT applies.

Case $F = N \supset A$**.** By AUGMENT, it suffices to show $P, N \vdash_O A$. Since $N \supset A \in [P] \subseteq [P, N]$, BACKCHAIN reduces this to $P, N \vdash_O N$. As $N \in [P, N]$ and N is neutral of smaller size, the induction hypothesis applies.

Case $F = N \supset (M_1 \cap M_2)$**.** Each $N \supset M_i$ lies in $[P]$ and is neutral of smaller size, so the induction hypothesis gives $P \vdash_O N \supset M_i$, and Lemma 2.14(i) gives $P, N \vdash_O M_i$. AND followed by AUGMENT concludes.

Case $F = N \supset (M' \supset M'')$**.** The formula $N \cap M' \supset M''$ lies in $[P]$; it is neutral, of the same size as F, and its head M'' is smaller than that of F. The induction hypothesis gives $P \vdash_O N \cap M' \supset M''$, and Lemma 2.14(i) gives $P, N \cap M' \vdash_O M''$, whence $P, N, M' \vdash_O M''$ by the observation following Lemma 2.13. Two applications of AUGMENT conclude.

Case $F = N \supset \forall X\, M$**.** Take an atom A fresh for P, N, and M. Then $N \supset M[A/X]$ lies in $[P]$ and is neutral of smaller size, so the induction hypothesis gives $P \vdash_O N \supset M[A/X]$, and Lemma 2.14(i) gives $P, N \vdash_O M[A/X]$. As A is fresh for P, N and M, GENERIC gives $P, N \vdash_O \forall X\, M$, and AUGMENT concludes.

Case $F = N_1 \cap N_2$**.** Each N_i lies in $[P]$ and is neutral of smaller size, so the induction hypothesis gives $P \vdash_O N_i$, and AND concludes.

Case $F = \forall X\, N$**.** Take an atom A fresh for P and N. Then $N[A/X] \in [P]$, and $N[A/X]$ is neutral of smaller size than F, so the induction hypothesis gives $P \vdash_O N[A/X]$, and GENERIC concludes. □

Lemma 4.7 (Elaboration)**.** *Let F be neutral with $P \vdash_O F$. If $F' \in [F]$, then F' is neutral, of size at most that of F, and $P \vdash_O F'$.*

Proof. By induction on the generation of F' in $[F]$, that is, in $[\{F\}]$. If $F' = F$ there is nothing to prove. If $F' = N_i$ arises by projection from $N_1 \cap N_2 \in [F]$, the induction hypothesis gives $P \vdash_O N_1 \cap N_2$, and Lemma 2.14(ii) gives $P \vdash_O N_i$. If $F' = N[A/X]$ arises by instantiation from $\forall X\, N \in [F]$, the induction hypothesis gives $P \vdash_O \forall X\, N$, and Lemma 2.14(iii) gives $P \vdash_O N[A/X]$.

The remaining three conditions concern compound heads. If $F' = N \supset M_i$ arises from $N \supset (M_1 \cap M_2) \in [F]$, the induction hypothesis and Lemma 2.14(i) and (ii) give $P, N \vdash_O M_i$, and AUGMENT gives $P \vdash_O N \supset M_i$. If $F' = N \cap M' \supset M''$ arises from $N \supset (M' \supset M'') \in [F]$, the induction hypothesis and two applications of Lemma 2.14(i) give $P, N, M' \vdash_O M''$, whence $P, N \cap M' \vdash_O M''$ by the observation following Lemma 2.13, and AUGMENT gives $P \vdash_O N \cap M' \supset M''$.

If $F' = N \supset M[A/X]$ arises from $N \supset \forall X\, M \in [F]$, the induction hypothesis and Lemma 2.14(i) and (iii) give $P, N \vdash_O M[A/X]$, and AUGMENT gives $P \vdash_O N \supset M[A/X]$. Neutrality and the bound on size are preserved at each step. □

Lemma 4.8 (Cut)**.** *Let F be neutral. If $P \vdash_O F$ and $Q, F \vdash_O G$, then $P, Q \vdash_O G$.*

Proof. By induction on the size of F and, within it, on the height of the derivation of $Q, F \vdash_O G$; the statement is taken for all P, Q, and G at once. Elaboration distributes over union, so $[Q, F] = [Q] \cup [F]$. We proceed by cases on the last rule of the derivation of $Q, F \vdash_O G$.

Case Fact: $G = A \in [Q] \cup [F]$. If $A \in [Q]$, then $A \in [P, Q]$ and FACT applies. If $A \in [F]$, Lemma 4.7 gives $P \vdash_O A$, and Lemma 2.13 gives $P, Q \vdash_O A$.

Case Backchain: $G = A$, with $G' \supset A \in [Q] \cup [F]$ and $Q, F \vdash_O G'$ by a shorter derivation. The secondary induction hypothesis gives $P, Q \vdash_O G'$. If $G' \supset A \in [Q]$, then $G' \supset A \in [P, Q]$ and BACKCHAIN applies. If $G' \supset A \in [F]$, then by Lemma 4.7 $G' \supset A$ is neutral of size at most that of F, so G' is neutral of smaller size than F, and $P \vdash_O G' \supset A$, whence Lemma 2.14(i) gives $P, G' \vdash_O A$. The primary induction hypothesis, applied to the cut formula G' with $P, Q \vdash_O G'$ and $P, G' \vdash_O A$, gives $P, Q, P \vdash_O A$, that is, $P, Q \vdash_O A$.

Cases And and Or. Immediate from the secondary induction hypothesis.

Case Augment: $G = D \supset G'$, with $Q, F, D \vdash_O G'$ by a shorter derivation. The secondary induction hypothesis, with $Q \cup \{D\}$ in place of Q, gives $P, Q, D \vdash_O G'$, and AUGMENT concludes.

Case Generic: $G = \forall X\, G'$, with $Q, F \vdash_O G'[A/X]$ by a shorter derivation, for some A fresh for Q, F, and G'. Choose an atom B fresh for P, Q, F, and G'. Lemma 2.12 applied to the renaming $A \mapsto B$ gives $Q, F \vdash_O G'[B/X]$ by a derivation of the same height, since A occurs in none of Q, F, G'. The secondary induction hypothesis gives $P, Q \vdash_O G'[B/X]$, and as B is fresh for P, Q and G', GENERIC concludes. □

Neither Lemma 4.6 nor Lemma 4.8 would be well formed without Lemma 4.4(i): in each, F occurs once as a goal and once as a clause of a program. This is where membership in both syntactic classes earns its keep. The arguments are those of the admissibility of cut for uniform proofs [24], adapted to the atomic quantifier.

Proof of the main theorem. We are now in a position to prove Theorem 4.2.

Proof of Theorem 4.2. By simultaneous induction on the structure of φ, establishing both directions together; the induction hypothesis for a subformula is used in both directions. Monotonicity of support (Proposition 3.4) and of $\vdash_O$ (Lemma 2.13) under base extension are used freely, as is the fact (Lemma 4.4) that a base extended by neutral formulae is again a base.

Case $\varphi = A$. Immediate: clause (At) of Figure 2 gives $\Vdash_{\mathcal{B}} A$ iff $\mathcal{B} \vdash_O A$, and $[\![A]\!] = A$.

Case $\varphi = \bot$.

(⇒) Suppose $\Vdash_{\mathcal{B}} \bot$. By clause (⊥), $\Vdash_{\mathcal{B}} A$ for every atom A, hence $\mathcal{B} \vdash_O A$ for every atom A. Taking A fresh for $\mathcal{B}$, GENERIC gives $\mathcal{B} \vdash_O \forall X\, X = [\![\bot]\!]$.

($\Leftarrow$) Suppose $\mathcal{B} \vdash_O \forall X\, X$. Lemma 2.14(iii) gives $\mathcal{B} \vdash_O A$ for every atom A, hence $\Vdash_{\mathcal{B}} A$ for every A, and clause ($\bot$) gives $\Vdash_{\mathcal{B}} \bot$.

Case $\varphi = \varphi_1 \to \varphi_2$.

($\Rightarrow$) Suppose $\Vdash_{\mathcal{B}} \varphi_1 \to \varphi_2$, so that $\varphi_1 \Vdash_{\mathcal{B}} \varphi_2$ by clause ($\to$). Set $\mathcal{C} \coloneqq \mathcal{B} \cup \{[\![\varphi_1]\!]\}$. Lemma 4.6 gives $\mathcal{C} \vdash_O [\![\varphi_1]\!]$, and the induction hypothesis gives $\Vdash_{\mathcal{C}} \varphi_1$. As $\mathcal{C} \sqsupseteq \mathcal{B}$, clause (Inf) yields $\Vdash_{\mathcal{C}} \varphi_2$, and the induction hypothesis returns $\mathcal{C} \vdash_O [\![\varphi_2]\!]$. AUGMENT gives $\mathcal{B} \vdash_O [\![\varphi_1]\!] \supset [\![\varphi_2]\!]$.

($\Leftarrow$) Suppose $\mathcal{B} \vdash_O [\![\varphi_1]\!] \supset [\![\varphi_2]\!]$, so that $\mathcal{B}, [\![\varphi_1]\!] \vdash_O [\![\varphi_2]\!]$ by Lemma 2.14(i). By clauses ($\to$) and (Inf), it suffices to show, for every $\mathcal{C} \sqsupseteq \mathcal{B}$ with $\Vdash_{\mathcal{C}} \varphi_1$, that $\Vdash_{\mathcal{C}} \varphi_2$. The induction hypothesis gives $\mathcal{C} \vdash_O [\![\varphi_1]\!]$. As $[\![\varphi_1]\!]$ is neutral, cut (Lemma 4.8) applied to $\mathcal{C} \vdash_O [\![\varphi_1]\!]$ and $\mathcal{B}, [\![\varphi_1]\!] \vdash_O [\![\varphi_2]\!]$ gives $\mathcal{C}, \mathcal{B} \vdash_O [\![\varphi_2]\!]$, that is, $\mathcal{C} \vdash_O [\![\varphi_2]\!]$, and the induction hypothesis returns $\Vdash_{\mathcal{C}} \varphi_2$.

Case $\varphi = \varphi_1 \wedge \varphi_2$. By clause ($\wedge$), $\Vdash_{\mathcal{B}} \varphi_1 \wedge \varphi_2$ iff $\Vdash_{\mathcal{B}} \varphi_i$ for $i = 1, 2$; by the induction hypothesis, iff $\mathcal{B} \vdash_O [\![\varphi_i]\!]$ for $i = 1, 2$; and by AND and Lemma 2.14(ii), iff $\mathcal{B} \vdash_O [\![\varphi_1]\!] \cap [\![\varphi_2]\!] = [\![\varphi_1 \wedge \varphi_2]\!]$.

Case $\varphi = \varphi_1 \vee \varphi_2$.

($\Rightarrow$) Suppose $\Vdash_{\mathcal{B}} \varphi_1 \vee \varphi_2$. We must show

$$\mathcal{B} \vdash_O \forall X \left(([\![\varphi_1]\!] \supset X) \cap ([\![\varphi_2]\!] \supset X) \supset X \right).$$

Take an atom A fresh for $\mathcal{B}$, $[\![\varphi_1]\!]$, and $[\![\varphi_2]\!]$; set $D \coloneqq ([\![\varphi_1]\!] \supset A) \cap ([\![\varphi_2]\!] \supset A)$ and $\mathcal{C} \coloneqq \mathcal{B} \cup \{D\}$. By GENERIC followed by AUGMENT, it suffices to show $\mathcal{C} \vdash_O A$. We claim $\varphi_i \Vdash_{\mathcal{C}} A$ for each i. Indeed, take $\mathcal{C}' \sqsupseteq \mathcal{C}$ with $\Vdash_{\mathcal{C}'} \varphi_i$; the induction hypothesis gives $\mathcal{C}' \vdash_O [\![\varphi_i]\!]$; since $[\![\varphi_i]\!] \supset A \in [D] \subseteq [\mathcal{C}']$, BACKCHAIN yields $\mathcal{C}' \vdash_O A$, which is $\Vdash_{\mathcal{C}'} A$. Since $\mathcal{C}' \sqsupseteq \mathcal{C}$ was arbitrary, the claim holds. Applying the supposition $\Vdash_{\mathcal{B}} \varphi_1 \vee \varphi_2$ at the extension $\mathcal{C}$ and the atom A, clause ($\vee$) gives $\Vdash_{\mathcal{C}} A$ — that is, $\mathcal{C} \vdash_O A$, as required.

($\Leftarrow$) Suppose $\mathcal{B} \vdash_O [\![\varphi_1 \vee \varphi_2]\!]$, and let $\mathcal{C} \sqsupseteq \mathcal{B}$ and an atom A be given with $\varphi_i \Vdash_{\mathcal{C}} A$ for $i = 1, 2$; by clause ($\vee$) it suffices to show $\mathcal{C} \vdash_O A$. By monotonicity, $\mathcal{C} \vdash_O [\![\varphi_1 \vee \varphi_2]\!]$, and Lemma 2.14(iii) and (i) give

$$\mathcal{C}, D \vdash_O A, \qquad \text{where } D \coloneqq ([\![\varphi_1]\!] \supset A) \cap ([\![\varphi_2]\!] \supset A).$$

We claim $\mathcal{C} \vdash_O [\![\varphi_i]\!] \supset A$ for each i. Let $\mathcal{C}_i \coloneqq \mathcal{C} \cup \{[\![\varphi_i]\!]\}$. Lemma 4.6 gives $\mathcal{C}_i \vdash_O [\![\varphi_i]\!]$, the induction hypothesis gives $\Vdash_{\mathcal{C}_i} \varphi_i$, and since $\mathcal{C}_i \sqsupseteq \mathcal{C}$, the hypothesis $\varphi_i \Vdash_{\mathcal{C}} A$ gives $\Vdash_{\mathcal{C}_i} A$, that is, $\mathcal{C}_i \vdash_O A$; AUGMENT establishes the claim. AND then gives $\mathcal{C} \vdash_O D$, and as D is neutral, cut (Lemma 4.8) applied to $\mathcal{C} \vdash_O D$ and $\mathcal{C}, D \vdash_O A$ gives $\mathcal{C} \vdash_O A$.

This completes the induction. □

Observe that no compound instantiation of a quantifier occurs anywhere in the proof, in keeping with Remark 4.5: the fresh atom introduced by GENERIC is moved to the atom supplied by the semantic clause by Lemma 2.14(iii), which is to say by renaming, and the compound formulae required as hypotheses are supplied by identity and discharged by cut.

Since every base extends the empty base, Proposition 3.4 shows that φ is supported in every base iff it is supported in the empty base. Theorem 3.8 therefore yields the following.

Corollary 4.9. *A formula φ is provable in intuitionistic propositional logic iff $\emptyset \vdash_O [\![\varphi]\!]$.*

Remark 4.10 (Miller's Kripke model). The proof of Theorem 4.2 follows a precedent in the proof theory of logic programming. Miller [21], and again in [22, §5.6], gives a Kripke-style semantics for hereditary Harrop programs in which the worlds are the programs themselves, ordered by extension, and shows that satisfaction at a program coincides with uniform provability from it; Hodas and Miller [18] extend the construction to a fragment of intuitionistic linear logic. The skeleton of the argument is the same as ours: monotonicity of both relations under extension of the program, initial sequents and the admissibility of cut for a restricted class of formulae, and a simultaneous induction on the structure of the formula. The reader who knows Miller's argument will find the present one familiar, and the debt is gladly acknowledged.

What has to be shown before the method applies is different. Here the semantic side is Sandqvist's base-extension semantics for an independently given object logic, whose connectives do not belong to the clausal language, and whose clauses for $\vee$ and $\bot$ are stated as elimination conditions quantifying over atoms, not as the satisfaction clauses of a Kripke model. That they agree with the search behaviour of the encoded formulae is shown in the corresponding cases of the proof, and it is where the encoding of Definition 4.1 and the atomic quantifier are needed.

The machinery, then, is in essentials Miller's. What is new is the identification: Sandqvist's support relation, motivated independently and defined without reference to proof-search, is that computational structure.

The theorem identifies support in a fixed base with proof-search in a fixed program. The universal quantifiers in the support clauses — the universal over base extensions in ($\to$) and (Inf), the universals over extensions and atoms in ($\vee$), and the universal over atoms in ($\bot$) — have all become steps of search: AUGMENT adds to the program, and GENERIC takes a fresh atom. None of them ranges over a completed totality. On this reading the semantics is finitary, mechanical, and exact.

5. DISCUSSION

The result of Section 4 admits a one-line statement: support in a fixed base is proof-search in a fixed program. The remainder of this paper makes three observations about what this statement means and what it makes possible.

The first concerns the meta-theory of base-extension semantics itself. The clauses defining support quantify universally over base extensions and atoms; read set-theoretically, they range over a completed totality of inferential contexts, populated in advance and quantified over from without. That reading is not forced. A constructivist asserts universal claims freely, understanding them as claims to a method, and we do not claim that Sandqvist's meta-theory anywhere makes a constructively invalid inference. But a reading merely available is one thing and a method exhibited another. A semantics with anti-realist ambitions ought to supply the second.

Theorem 4.2 supplies it. Each universal quantifier in the support clauses is identified with a specific operational mechanism: the universal over base extensions in clauses ($\to$) and (Inf) becomes the augmentation of the program in AUGMENT; the universals over atoms in ($\vee$) and ($\bot$) become the freshness condition in GENERIC.

None of these mechanisms ranges over a completed totality. Each is a single step, standard in the proof theory of quantification. The semantics is constructive not merely in aspiration but in execution.

The set-theoretic and the constructive readings of the quantifiers agree on which formulae are supported in which bases (Remark 3.3); the theorem does not separate them extensionally. What it fixes is the intensional content of support: what it is to verify $\Vdash_{\mathcal{B}} \varphi$. On the first reading, it is the truth of a universal claim about a class of inferential contexts; on the second, it is the existence of a successful run of a finite search. The theorem shows that base-extension semantics, read through its own clauses, defines the second.

The second observation concerns the limits of the result, and is perhaps more interesting than the result itself. Theorem 4.2 depends essentially on the higher-level structure of atomic systems — that is, on the availability of hypothetical premisses in atomic rules. Without higher-level rules, the encoding of Section 4 cannot be carried out: hereditary Harrop formulae collapse to Horn clauses, and the elimination-driven clauses of the support relation lose their natural translation. The correspondence between support and proof-search dissolves with them.

This dependence is not a technical artefact; it tracks a genuine logical fact. Sandqvist [38] has shown that restricting base-extension semantics to first-level atomic systems yields a characterisation not of intuitionistic but of classical propositional logic. The same restriction that destroys our encoding also destroys the intuitionistic character of the semantics. The two phenomena are aspects of a single fact: *higher-level rules are to atomic systems what hypothetical goals are to logic programming*; and the difference between having them and lacking them is the difference between intuitionistic and classical logic. The present paper establishes the intuitionistic half of this correspondence.

The classical side remains open, and the gap is significant. Dummett's programme leaves room, in principle, for a vindication of classical logic on anti-realist grounds, without prior commitment to bivalence, though Dummett himself doubted that one was to be had and concluded that classical logic should be given up. Sandqvist [37, 38] offers his classical semantics as just such a vindication.

If support in the classical setting is genuinely a coherent anti-realist notion, then it should admit a constructive, search-based reading analogous to the one we have given for the intuitionistic case. If it does not — if the classical correspondence cannot be made computational without falling back on Horn clauses and a model-theoretic reading of the universal quantifiers — then the anti-realist credentials of classical base-extension semantics are not what they have been advertised to be. We leave this question open, but flag it as the natural sequel to the present work, and as a test case for the foundational claims of the broader programme.

The third observation looks outward. Base-extension semantics has been developed for a range of logics beyond intuitionistic propositional logic — first- and higher-order intuitionistic logic [7, 12], intuitionistic multiplicative linear logic [9], the logic of bunched implications [10], and others — and the pattern of the semantic clauses is the same in every case. The meaning of each connective is given by its elimination behaviour, expressed as a universal quantification over base extensions. The shape that drives our encoding is not peculiar to intuitionistic propositional logic: it is constitutive of the framework.

This suggests a research programme. Wherever the elimination-driven shape holds, and wherever a suitable analogue of higher-level atomic systems is available, the present result indicates that support should reduce to proof-search in a corresponding logic programming language. Carrying this out systematically would require, for each logic, identifying the right class of program clauses, the right encoding of formulae as goals, and the right notion of uniform proof. In some cases the candidate is already to hand: for first-order logic, the first-order hereditary Harrop programs of Miller et al. [24]; for intuitionistic multiplicative linear logic, the linear hereditary Harrop programs of Hodas and Miller [18]. The payoff would be substantial: a uniform computational foundation for proof-theoretic semantics, in which the philosophical content of the framework is at every point underwritten by a transparent mathematical object.

Proof-theoretic semantics has long been received — with some justice, given its expository register — as a primarily philosophical undertaking, of interest chiefly to those already persuaded of its anti-realist motivations. The present result, and the programme it sketches, suggests a different reading. Under the surface, and largely without remarking on it, proof-theoretic semantics has been writing computational specifications for the logical constants. On the reading we have given, Sandqvist's clauses for intuitionistic logic are a logic program; the theorem does not impose that structure but finds it there. Whether the same is true of the framework as a whole is a question we leave open, but one we now have the means to ask precisely.

References

[1] M. Dummett, *The Justification of Deduction*, in Truth and other Enigmas, Duckworth & Co, 1978, p. 318.

[2] ———, *The Logical Basis of Metaphysics*, Harvard University Press, 1991.

[3] ———, *What Is a Theory of Meaning?*, in The Seas of Language, Oxford University Press, 1996.

[4] F. Ferreira and G. Ferreira, *Atomic Polymorphism*, The Journal of Symbolic Logic, 78 (2013), pp. 260–274.

[5] F. Ferreira and G. Ferreira, *The faithfulness of atomic polymorphism*, Trends in Logic XIII, (2014).

[6] A. V. Gheorghiu, *On an Inferential Semantics for Intuitionistic Sentential Logic*, Journal of Logic and Computation, 35 (2025).

[7] ———, *Proof-theoretic Semantics for First-order Logic*, Logic Journal of the IGPL, 33 (2025).

[8] ———, *Support is search*. arXiv:2603.13018, 2026. Accessed August 2026.

[9] A. V. Gheorghiu, T. Gu, and D. J. Pym, *Proof-theoretic Semantics for Intuitionistic Multiplicative Linear Logic*, in Automated Reasoning with Analytic Tableaux and Related Methods — TABLEAUX, R. Ramanayake and J. Urban, eds., Springer, 2023, pp. 367–385.

[10] ———, *Proof-theoretic Semantics for the Logic of Bunched Implications*, Studia Logica, (2025).

[11] A. V. Gheorghiu and D. J. Pym, *Definite Formulae, Negation-as-Failure, and the Base-extension Semantics for Intuitionistic Propositional Logic*, Bulletin of the Section of Logic, (2023).

[12] ———, *Proof-theoretic Semantics for Second-order Logic*. arXiv preprint, August 2025.

[13] T. G. Griffin, *A formulae-as-types notion of control*, in Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '90), ACM, 1990, pp. 47–58.

[14] T. Gu, D. J. Pym, E. Ritter, and E. Robinson, *Continuations and Completeness in Proof-theoretic Semantics*, 2026. arXiv:2605.04689.

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

[16] ———, *A Proof-theoretic Approach to Logic Programming: II. Programs as Definitions*, Journal of Logic and Computation, 1 (1991), pp. 635–660.

[17] R. Harrop, *Concerning formulas of the types $A \to B \vee C$, $A \to (Ex)B(x)$ in intuitionistic formal systems* , The Journal of Symbolic Logic, 25 (1960), pp. 27–32.
[18] J. S. Hodas and D. Miller, *Logic programming in a fragment of intuitionistic linear logic*, Information and Computation, 110 (1994), pp. 327–365.
[19] R. Kowalski, *Logic for Problem Solving*, no. 7 in Artificial Intelligence Series, North-Holland, New York, 1979.
[20] D. Miller, *A Logical Analysis of Modules in Logic Programming*, The Journal of Logic Programming, 6 (1989), pp. 79–108.
[21] ———, *Abstract syntax and logic programming*, in Logic Programming: Proceedings of the First and Second Russian Conferences on Logic Programming, vol. 592 of Lecture Notes in Artificial Intelligence, Springer, 1992, pp. 322–337.
[22] ———, *Proof Theory and Logic Programming: Computation as Proof Search*, Cambridge University Press, Cambridge, 2025.
[23] D. Miller and G. Nadathur, *Programming with Higher-Order Logic*, Cambridge University Press, Cambridge, 2012.
[24] D. Miller, G. Nadathur, F. Pfenning, and A. Scedrov, *Uniform proofs as a foundation for logic programming*, Annals of Pure and Applied Logic, 51 (1991), pp. 125–157.
[25] G. Mints, *Gentzen-type systems and resolution rules Part I: Propositional logic*, in Computer Logic—COLOG-88, P. Martin-Löf and G. Mints, eds., Lecture Notes in Computer Science, Springer, 1988, pp. 198–231.
[26] T. Piecha, *Completeness in Proof-theoretic Semantics*, in Advances in Proof-theoretic Semantics, Springer, 2016, pp. 231–251.
[27] T. Piecha, W. de Campos Sanz, and P. Schroeder-Heister, *Failure of Completeness in Proof-theoretic Semantics*, Journal of Philosophical Logic, 44 (2015), pp. 321–335.
[28] T. Piecha and P. Schroeder-Heister, *Atomic Systems in Proof-Theoretic Semantics: Two Approaches*, in Epistemology, Knowledge and the Impact of Interaction, Ángel Nepomuceno Fernández, O. P. Martins, and J. Redmond, eds., Springer Verlag, 2016.
[29] ———, *The Definitional View of Atomic Systems in Proof-theoretic Semantics*, in The Logica Yearbook 2016, College Publications London, 2017, pp. 185–200.
[30] ———, *Incompleteness of Intuitionistic Propositional Logic with Respect to Proof-theoretic Semantics* , Studia Logica, 107 (2019), pp. 233–246.
[31] G. D. Plotkin, *Call-by-name, call-by-value and the λ-calculus*, Theoretical Computer Science, 1 (1975), pp. 125–159.
[32] D. Prawitz, *Ideas and Results in Proof Theory*, in Studies in Logic and the Foundations of Mathematics, vol. 63, Elsevier, 1971, pp. 235–307.
[33] ———, *Towards a Foundation of a General Proof Theory*, in Studies in Logic and the Foundations of Mathematics, vol. 74, Elsevier, 1973, pp. 225–250.
[34] ———, *Natural Deduction: A Proof-theoretical Study*, Dover Publications, 2006 [1965].
[35] D. J. Pym and E. Ritter, *Reductive Logic and Proof-search: Proof Theory, Semantics, and Control*, Oxford University Press, 2004.
[36] B. Russell, *The Principles of Mathematics*, Cambridge University Press, Cambridge, 1903.
[37] T. Sandqvist, *An Inferentialist Interpretation of Classical Logic*, PhD thesis, Uppsala University, 2005.
[38] ———, *Classical Logic without Bivalence*, Analysis, 69 (2009), pp. 211–218.
[39] ———, *Base-extension Semantics for Intuitionistic Sentential Logic*, Logic Journal of the IGPL, 23 (2015), pp. 719–731.
[40] P. Schroeder-Heister, *Validity Concepts in Proof-theoretic Semantics*, Synthese, 148 (2006), pp. 525–571.
[41] ———, *Proof-Theoretic versus Model-Theoretic Consequence*, in The Logica Yearbook 2007, M. Pelis, ed., Filosofia, 2008.
[42] ———, *Proof-Theoretic Semantics*, in The Stanford Encyclopedia of Philosophy, E. N. Zalta, ed., Metaphysics Research Lab, Stanford University, Spring 2018 ed., 2018.
[43] W. Stafford, *Proof-Theoretic Semantics and Inquisitive Logic*, Journal of Philosophical Logic, (2021).

University College London
Email address: a.v.gheorghiu@ucl.ac.uk