The Bulletin of Symbolic Logic
Volume 32, Number 3, September 2026

A FORMALISATION OF CONSTRUCTIVE EVIDENCE-BASED REASONING: CONSTRUCTING JUSTIFICATIONS

JUAN C. AGUDELO-AGUDELO AND WALTER CARNIELLI

Abstract. A *Constructive Logic of Evidence and Truth* (LET_C) is introduced. This logic is both paraconsistent and paracomplete, providing connectives for consistency and determinedness that enable the independent recovery of explosiveness and the law of excluded middle for specific propositions. Dual connectives for inconsistency and undeterminedness are also defined in LET_C. Evidence is explicitly formalised by integrating lambda calculus terms into LET_C, resulting in the type system $\mathsf{LET}^{\lambda}_\mathsf{C}$. In this system, lambda calculus terms represent procedures for constructing evidence for compound formulas based on the evidence of their constituent parts. A realisability interpretation is provided for LET_C, establishing a strong connection between deductions in this system and recursive functions.

§1. Introduction. The aim of the project behind the Logics of Evidence and Truth (LET's), initiated by the publication of Carnielli and Rodrigues [10], is to offer an interpretation of paraconsistent logics in which contradictions are epistemically viewed as conflicting evidence, where evidence for a proposition A is understood as reasons supporting the belief that A is true. Under this view, the project is antagonistic (or orthogonal) to the dialetheist approach, that is, the view that true contradictions exist, cf. Priest [19], since the LETs cannot support a real contradiction, at the risk of trivialisation. The project actually has a dual purpose: to formalise the notion of evidence, a concept that has received limited attention from both philosophers and logicians.

Some interesting developments include Kripke-style semantics for these logics, which allows the concept of evidence to be viewed as a modality, as described in Antunes et al. [2], and the proposal of Bayesian paraconsistent epistemology, as in Carnielli and Bueno-Soler [8]. What remains to be done is a proper algebraic interpretation of these logics.

Received November 14, 2024.
2020 Mathematics Subject Classification. 03B53, 03B38, 68T27.
Key words and phrases. paraconsistent logic, paracomplete logic, constructive logic, evidence-based logic.

© The Author(s), 2025. Published by Cambridge University Press on behalf of The Association for Symbolic Logic.
1079-8986/26/3203-0001
DOI:10.1017/bsl.2025.10

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

The *Basic Logic of Evidence* (BLE), proposed in Carnielli and Rodrigues [10] is extended to the *Logic of Evidence and Truth* ($\mathsf{LET_J}$). BLE was conceived to formalise deductions based on evidence, in the above sense. In BLE the negation connective, here denoted by $\sim$, formalises falsity; thus evidence for $\sim A$ can be understood as reasons for believing in the falsity of A, and reasons for believing in the falsity of the falsity of A are taking as reasons for believing in A (consequently $\sim\sim A$ and A are equivalent in BLE). BLE is a paraconsistent and paracomplete logic. Contradictions in BLE reflect the existence of conflicting evidence, and the paraconsistency of BLE corresponds to the possible existence of conflicting evidence without leading to a deductive collapse. On the other hand, the paracompleteness of BLE corresponds to the possible lack of evidence for some propositions and their negations. Although conceived with different motivations, BLE turns out to be equivalent to the propositional fragment of *Nelson's Paraconsistent Logic*, usually denoted by N4 (see, for instance, Kamide [13]). $\mathsf{LET_J}$ is obtained by extending BLE with a primitive *classicality operator* $\circ$, which allows to recover simultaneously the explosiveness of contradictions and the excluded middle for some propositions. Evidence for a proposition A can be *non-conclusive*, in the sense that it may not absolutely determine the truth of A. The formula $\circ A$ represents the existence of conclusive evidence for either the truth or the falsity of the proposition A, and reasoning under conclusive evidence follows the rules of classical logic (this is what the *Derivability Adjustment Theorem* shows), hence the name given to the operator $\circ$.

Another logic of evidence and truth ($\mathsf{LET_F}$), and a probabilistic semantics for it, is proposed in Rodrigues et al. [20]. $\mathsf{LET_F}$ is an extension of the Logic of First-Degree Entailment (FDE), also known as Belnap–Dunn four-valued logic, and which corresponds to the fragment of N4 without implication. $\mathsf{LET_F}$ is a slightly modified version of $\mathsf{LET_J}$, which additionally to the classicality operator $\circ$ also contains a *non-classicality operator* $\bullet$ (i.e., $\mathsf{LET_F}$ is $\mathsf{LET_J}$ dropping implication and adding the non-classicality operator). The operator $\bullet$ is dual to $\circ$ in the sense that, while $\circ A$ implies the classical behaviour of A, a non-classical behaviour of A implies $\bullet A$. The elimination of implication in $\mathsf{LET_F}$ is due to some difficulties in the probabilistic interpretation of this connective.

A summary of recent work on *Logics of Evidence and Truth (LETs)* is presented in Rodrigues et al. [21], where a new LET named $\mathsf{LET_K}$ and its first-order extension $\mathsf{QLET_K}$ are also introduced, and an application of these logics to the problem of abduction is shown. $\mathsf{LET_K}$ is an extension of $\mathsf{LET_F}$ with a classical implication connective.

Here, a *Constructive Logic of Evidence and Truth*, denoted by $\mathsf{LET_C}$, is introduced. $\mathsf{LET_C}$ is a definitional extension of N4*, which in turn is an extension of N4 with a co-implication operator and bottom and top particles. $\mathsf{LET_C}$ extends N4* by defining connectives for consistency,

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

inconsistency, determinedness, and undeterminedness. In $\mathsf{LET_C}$, differently of the other LETs, explosiveness of contradictions and excluded middle cannot be simultaneously recovered by a single classicality operator, but they can be independently recovered by the operators of consistency and determinedness, respectively.

Truth and falsity are treated asymmetrically in *Intuitionistic Logic* (Int): the truth of a proposition signifies the existence of a proof for it and is established directly, whereas the falsity of a proposition is proven indirectly by demonstrating the impossibility of constructing a proof for it. This is reflected in the definition of intuitionistic negation as $\neg A \equiv A \to \bot$. In Int, it is not possible to directly prove the falsity of a proposition. Moreover, the very nature of intuitionistic falsity is debatable (cf. Shramko [23]).

In contrast, $\mathsf{LET_C}$ maintains full symmetry between truth and falsity, allowing for direct proofs of both the truth and falsity of a proposition. Additionally, $\mathsf{LET_C}$ enables an interesting distinction between truth and the impossibility of falsity, as well as between falsity and the impossibility of truth—concepts that are inseparable in classical logic and whose distinction appears muddled in Int.

As discussed below, these differences allow us to reinterpret the notion of 'constructive proof' as 'direct proof'. This leads us to understand the proof of $\neg A$ in Int as a direct proof of the impossibility of A's truth, rather than as an indirect (and thus non-constructive) proof of A's falsity. Against this backdrop, $\mathsf{LET_C}$ proves to be constructively richer than Int, as it also allows direct proofs of the falsity of propositions and the impossibility of their truth.

In order to explicitly formalise evidence, lambda calculus terms are added to $\mathsf{LET_C}$, obtaining a type system denoted by $\mathsf{LET^{\lambda}_C}$. Some usual properties of type systems are proven for $\mathsf{LET^{\lambda}_C}$: lemmas of generation, free variables, uniqueness of types (under a strong equivalence defined on formulas), substitution and subject reduction, and a normalisation theorem. Our proposal of representing evidence by lambda calculus terms, is a different approach for the explicit formalisation of evidence in the justification logics proposed in Artemov [5], which are obtained by understanding the necessity operator of some modal logics as implicitly representing the existence of evidence, and by transforming the necessity operator into justification polynomials (or justification terms) for explicitly represent evidence. Artemov's approach is used in Fitting [12] for providing an explicit formalisation of evidence for BLE, by translating BLE into the modal logic KX4 and constructing the justification logic JX4 from KX4.

A multimodal system that combines S4 and KX4 into a single formal system, denoted as $\mathsf{SKX4^{+}}$, is introduced in Carnielli et al. [9]. The motivation behind this integration is to develop a logic that, given the embeddings of Int and BLE into S4 and KX4, can effectively represent the deductive behavior

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

of factive and non-factive evidence. To clarify this, it is useful to recall that non-factive evidence refers to evidence that neither implies nor guarantees the truth of the sentence it is supposed to support, while factive evidence is evidence that implies or guarantees the truth of the sentence it supports.

While Artemov's justification logics aim to provide a new evidence-based foundation for epistemic logics, our goal here is to offer a functional (and constructive) interpretation of evidence-based reasoning. To this end, we use lambda terms to explicitly represent evidence, drawing on the methods and philosophy of type theory. This approach contrasts with Artemov's use of justification polynomials. In our framework, the variables of lambda terms represent arbitrary evidence, while more complex lambda terms describe procedures that enable evidence to be derived from other evidence.

In order to emphasise the constructive aspects of LET_C, a Kleene-style realisability interpretation is provided for this logic. This establishes a strong connection between derivations in LET_C and recursive functions.

The *Dynamic Logics of Evidence-Based Beliefs*, introduced in van Benthem and Pacuit [7], are a proposal to formalise evidence-based reasoning by taking into account its dynamic properties. These logics are defined by extending the neighbourhood semantics for modal logics, where evidence is formalised as families of sets of worlds. This formalisation turns out to be very technical and difficult to understand intuitively. Contrarily, the LETs (particularly LET_C) formalise a instantaneous view of evidence-based reasoning, in the sense of considering only evidence accessible at a given time, and the rules of deduction are based on a rather intuitive conception of the acceptance of propositions according to existing evidence. This instantaneous approach to evidence, however, creates a dynamic of movement, much like in cinema, with the evidence being continuously changed and updated.

In the area of Artificial Intelligence, the logics of evidence and truth can be useful in the formalisation and automation of deductions from information sources (for instance, some databases or the Web) where there may be a lack of information or conflicting information, which may lead to neither accepting a proposition nor its negation, or to having reasons to accept both a proposition and its negation. Classical logic is clearly not apt for such contexts because, when applied to information, the excluded third states that it is complete (i.e., there is always enough information to determine, for any given proposition, that it or its negation is true), and the explosion principle states that any proposition can be derived from contradictory information, leading to a deductive collapse.

In addition to LET_C being a logic suitable for formalising inferences in contexts with missing or conflicting information, the addition of terms to this logic can allow the construction of explanations based on the evidence for the inferences made.

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

This article is structured as follows: Section 2 introduces $\mathsf{LET_C}$ through a natural deduction system. Section 3 extends $\mathsf{LET_C}$ with the addition of lambda terms. Section 4 outlines several relevant properties of $\mathsf{LET}^{\lambda}_{\mathsf{C}}$. A realisability interpretation for $\mathsf{LET_C}$ is presented in Section 5, followed by a discussion of the constructive nature of this logic in Section 6. Finally, Section 7 provides concluding remarks and suggest directions for future research.

§2. A natural deduction system for $\mathsf{LET_C}$. We begin by describing the system $\mathsf{N4}^{\star}$, an extension of Nelson's paraconsistent logic $\mathsf{N4}$ with a co-implication operator ($\prec$) and bottom ($\bot$) and top ($\top$) particles. The signature of $\mathsf{N4}^{\star}$ is $\mathcal{S}_{\mathsf{N4}^{\star}} = \{\wedge, \vee, \rightarrow, \prec, \sim, \bot, \top\}$, and the set of formulas is defined in the usual way. The unary operator $\sim$ is called *constructive negation* (or *strong negation*), and a formula $B \prec A$ can be read as *A co-implies B* or as *B excludes A* (cf. [25, Footnote 2]).

The natural deduction system for $\mathsf{N4}^{\star}$ is presented in Table 1. Analogously as the rules of Int can be interpreted by means of the existence of proofs, each one of the rules in $\mathsf{N4}^{\star}$ admits an implicit interpretation in terms of evidence: if there is evidence for accepting the formulas in the premises, there is evidence for accepting the formula in the conclusion. Assumptions correspond to the presupposition of existence of evidence (details in Carnielli and Rodrigues [10]).

The fragment of $\mathsf{N4}^{\star}$ without $\top$ and $\prec$ is equivalent to $\mathsf{N4}^{\bot}$, which is an Odintsov's extenstion of $\mathsf{N4}$ with a bottom particle introduced in Odintsov [18]. As $\top$ and $\prec$ can be defined in $\mathsf{N4}^{\bot}$ by $\top \stackrel{\text{def}}{=} \sim \bot$ and $B \prec A \stackrel{\text{def}}{=} \sim(\sim A \rightarrow \sim B)$, the semantics for $\mathsf{N4}^{\bot}$ presented in Odintsov [18] is also adequate for $\mathsf{N4}^{\star}$.

REMARK 1. Before going further, we outline here some reasons for working with natural deduction in this paper. In contrast to Hilbertian axiomatisation, natural deduction systems are more aligned with intuitive reasoning. The introduction and elimination rules for logical connectives in natural deduction help to clarify the connectives themselves. This approach makes proofs more goal-oriented, as they begin with assumptions and apply inference rules to simplify or construct conclusions. Furthermore, in natural deduction, the normalisation property ensures that every proof has a normal form, meaning that redundant steps (such as unnecessary assumptions or intermediate steps) can be eliminated. This property is rarely found in Hilbertian systems. From a computational complexity perspective, Hilbert systems are typically NP-complete or even PSPACE-complete, whereas natural deduction systems are not inherently NP-complete. This distinction helps explain the dedication that proof theorists like Gerhard Gentzen,

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

Table 1. Natural deduction system for N4*.

$\dfrac{}{\top}(I_\top)$	$\dfrac{\bot}{A}(E_\bot)$
$\dfrac{A \quad B}{A\wedge B}(I_\wedge)$	$\dfrac{A\wedge B}{A}(E1_\wedge) \quad \dfrac{A\wedge B}{B}(E2_\wedge)$
$\dfrac{A}{A\vee B}(I1_\vee)$	$[A] \quad [B]$
$\dfrac{B}{A\vee B}(I2_\vee)$	$\vdots \quad \vdots$ $\dfrac{A\vee B \quad C \quad C}{C}(E_\vee)$
$[A]$ $\vdots$ $\dfrac{B}{A\to B}(I_\to)$	$\dfrac{A\to B \quad A}{B}(E_\to)$
$\dfrac{\sim A \quad B}{B\prec A}(I_\prec)$	$\dfrac{B\prec A}{\sim A}(E1_\prec) \quad \dfrac{B\prec A}{B}(E2_\prec)$
$\dfrac{}{\sim\bot}(I_{\sim\bot})$	$\dfrac{\sim\top}{A}(E_{\sim\top})$
$\dfrac{\sim A}{\sim(A\wedge B)}(I1_{\sim\wedge})$	$[\sim A] \quad [\sim B]$
$\dfrac{\sim B}{\sim(A\wedge B)}(I2_{\sim\wedge})$	$\vdots \quad \vdots$ $\dfrac{\sim(A\wedge B) \quad C \quad C}{C}(E_{\sim\wedge})$
$\dfrac{\sim A \quad \sim B}{\sim(A\vee B)}(I_{\sim\vee})$	$\dfrac{\sim(A\vee B)}{\sim A}(E1_{\sim\vee}) \quad \dfrac{\sim(A\vee B)}{\sim B}(E2_{\sim\vee})$
$\dfrac{A \quad \sim B}{\sim(A\to B)}(I_{\sim\to})$	$\dfrac{\sim(A\to B)}{A}(E1_{\sim\to}) \quad \dfrac{\sim(A\to B)}{\sim B}(E2_{\sim\to})$
$[\sim A]$ $\vdots$ $\dfrac{\sim B}{\sim(B\prec A)}(I_{\sim\prec})$	$\dfrac{\sim A \quad \sim(B\prec A)}{\sim B}(E_{\sim\prec})$
$\dfrac{A}{\sim\sim A}(I_{\sim\sim})$	$\dfrac{\sim\sim A}{A}(E_{\sim\sim})$

Stanisław Jaśkowski, Dag Prawitz, and Michael Dummett have devoted to natural deduction systems.

An equivalence operator $\leftrightarrow$ can be defined in N4* as usual $A \leftrightarrow B \stackrel{\text{def}}{=} (A \to B)\wedge(B \to A)$. As in N4, substitution by equivalent formulas in N4* is not valid. For instance, in N4* the equivalence $(B \prec A) \leftrightarrow (B\wedge \sim A)$ is provable, whereas the equivalence $\sim(B \prec A) \leftrightarrow \sim(B\wedge \sim A)$ is not provable.

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

However, a *strong equivalence* operator $\Leftrightarrow$ can be defined by $A \Leftrightarrow B \stackrel{\text{def}}{=} (A \leftrightarrow B) \wedge (\sim A \leftrightarrow \sim B)$, and substitution by strongly equivalent formulas is valid.

In $\mathsf{N4}^{\star}$, the connectives $\wedge$ and $\vee$ are dual, because $A \vee B$ is strongly equivalent to $\sim(\sim A \vee \sim B)$ (these formulas are practically the same in this system). Moreover, the connectives $\rightarrow$ and $\prec$ can also be seen as dual, because $B \prec A$ is strongly equivalent to $\sim(\sim A \rightarrow \sim B)$ (which can be easily proven using the rules in Table 1). It is therefore possible to define the connectives $\vee$ and $\prec$ in terms of the connectives $\wedge$, $\rightarrow$ and $\sim$, rather than regarding them as primitive. Nevertheless, we consider $\vee$ and $\prec$ to be primitive connectives. This is with the aim of making the duality between the connectives and the symmetry between truth and falsity more evident.

In $\mathsf{N4}^{\star}$, an *intuitionistic negation* $\neg$ is defined by $\neg A \stackrel{\text{def}}{=} A \rightarrow \bot$ and a *co-negation* $\sim$ is defined by $\sim A \stackrel{\text{def}}{=} \top \prec A$. While the intention of Nelson's constructive negation is to express falsity (i.e., $\sim A$ must be interpreted as 'A is false', cf. Nelson [17]), intuitionistic negation must be interpreted as expressing impossibility of truth (i.e., $\neg A$ must be interpreted as 'it is impossible that A be true' or as 'it is impossible to construct a proof of A', cf. Shramko [23]). Accordingly, the formula $\sim\sim A$, which is strongly equivalent to $\neg\sim A$, must be understood as the *impossibility of the falsity of* A. Under such interpretations, the following theorem shows that, in $\mathsf{N4}^{\star}$, the falsity of a formula A is not equivalent to the impossibility of the truth of A (item 1), and the truth of a formula A is not equivalent to the impossibility of the falsity of A (item 2). However, the falsity of a formula A is equivalent to the falsity of the impossibility of the falsity of A (item 3), and the truth of a formula A is equivalent to the falsity of the impossibility of the truth of A (item 4). The other items show additional properties of the interaction between negations in $\mathsf{N4}^{\star}$.

THEOREM 2.1.

1. $\nvdash \sim A \leftrightarrow \neg A$ (*neither* $\vdash \sim A \rightarrow \neg A$ *nor* $\vdash \neg A \rightarrow \sim A$).
2. $\nvdash A \leftrightarrow \sim\sim A$ (*neither* $\vdash A \rightarrow \sim\sim A$ *nor* $\vdash \sim\sim A \rightarrow A$).
3. $\vdash \sim A \leftrightarrow \sim\sim\sim A$ (*and also* $\vdash \sim A \leftrightarrow \sim A$).
4. $\vdash A \leftrightarrow \sim\neg A$.
5. $\vdash \sim\neg A \Leftrightarrow \sim\sim A$.
6. $\vdash \sim\sim A \Leftrightarrow \neg\sim A$.

PROOF. Using the natural deduction system for provable formulas and any of the semantics for $\mathsf{N4}^{\perp}$ presented in Odintsov [18] for unprovable formulas. ⊣

It is important to highlight that the distinctions between truth and impossibility of falsity, and between falsity and impossibility of truth, cannot be made in other logics, such as classical or intuitionistic logic. These are

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

remarkable features of $\mathsf{N4}^{\star}$ that are important for the constructive properties of this logic, as explained in Section 6.

The following theorem shows some properties of formulas related with the principles of *explosiveness* (or *ex contradictione [sequitur] quodlibet*) and *excluded middle* (or *tertium non datur*) in $\mathsf{N4}^{\star}$.

THEOREM 2.2.

1. $\vdash \neg(A \wedge \sim A) \Leftrightarrow \sim \urcorner(A \vee \sim A)$.
2. $\vdash \urcorner(A \wedge \sim A) \leftrightarrow (A \vee \sim A)$, *but* $\nvdash \urcorner(A \wedge \sim A) \Leftrightarrow (A \vee \sim A)$.
3. $\vdash \urcorner(A \wedge \sim A) \Leftrightarrow \sim \neg(A \vee \sim A)$.
4. $\vdash \sim \neg(A \wedge \sim A) \leftrightarrow (A \wedge \sim A)$, *but* $\nvdash \sim \neg(A \wedge \sim A) \Leftrightarrow (A \wedge \sim A)$.
5. $\vdash \sim \neg(A \wedge \sim A) \Leftrightarrow \urcorner(A \vee \sim A)$.
6. $\vdash \sim \urcorner(A \wedge \sim A) \Leftrightarrow \neg(A \vee \sim A)$.

PROOF. Using the natural deduction system for provable formulas and any of the semantics for $\mathsf{N4}^{\perp}$ presented in Odintsov [18] for unprovable formulas. ⊣

The previous theorem motivates the following definitions of connectives for *consistency* ($\circ$), *inconsistency* ($\bullet$), *determinedness* ($\bigstar$), and *undetermined-ness* ($\bigstar$). The logic $\mathsf{LET}_{\mathsf{C}}$ is the result of adding to $\mathsf{N4}^{\star}$ such definitions.

$$\circ A \stackrel{\text{def}}{=} \neg(A \wedge \sim A),$$
$$\bigstar A \stackrel{\text{def}}{=} \urcorner(A \wedge \sim A) \text{ (equivalently, } \bigstar A \stackrel{\text{def}}{=} \sim \neg(A \vee \sim A)),$$
$$\bullet A \stackrel{\text{def}}{=} \urcorner(A \vee \sim A) \text{ (equivalently, } \bullet A \stackrel{\text{def}}{=} \sim \circ A),$$
$$\bigstar A \stackrel{\text{def}}{=} \neg(A \vee \sim A) \text{ (equivalently, } \bigstar A \stackrel{\text{def}}{=} \sim \bigstar A).$$

Note that although the definition of the consistency connective is similar to the one proposed by da Costa in his well-known hierarchy of *Inconsistent Formal Systems* C_{n}, our definition differs in a significant way. We employ two negations in place of one, with each negation representing a distinct concept (namely the impossibility of truth and falsity).

The following properties are immediate from the previous definition of connectives and theorems.

THEOREM 2.3.

1. $A, \sim A, \circ A \vdash B$.
2. $\bullet A \vdash A \wedge \sim A$ *and* $A \wedge \sim A \vdash \bullet A$.
3. $\bigstar A \vdash A \vee \sim A$, $A \vdash \bigstar A$ *and* $\sim A \vdash \bigstar A$.
4. $A, \bigstar A \vdash B$ *and* $\sim A, \bigstar A \vdash B$.

§3. Adding lambda-terms to $\mathsf{LET}_{\mathsf{C}}$: making evidence explicit. A BHK-style interpretation of $\mathsf{LET}_{\mathsf{C}}$ in terms of evidence for accepting or rejecting

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

propositions is presented below. This informal interpretation gives us the intuitions for adding lambda terms to $\mathsf{LET_C}$. The emptyset symbol ($\emptyset$) is used to represent evidence for accepting propositions which are self-evident (i.e., propositions accepted without need of proofs or explanations):

- Evidence for accepting $A \wedge B$ is a pair (e_1, e_2) where e_1 is evidence for accepting A and e_2 is evidence for accepting B.
- Evidence for accepting $A \vee B$ is a pair (e_1, e_2) where $e_1 = 0$ and e_2 is evidence for accepting A, or $e_1 = 1$ and e_2 is evidence for accepting B.
- Evidence for accepting $A \rightarrow B$ is a method that converts evidence for accepting A into evidence for accepting B.
- Evidence for accepting $B \prec A$ is a pair (e_1, e_2) where e_1 is evidence for rejecting A and e_2 is evidence for accepting B.
- Evidence for accepting $\sim A$ is evidence for rejecting A.
- There is no evidence for accepting $\bot$.
- $\emptyset$ is evidence for accepting $\top$.
- Evidence for rejecting $A \wedge B$ is a pair (e_1, e_2) where $e_1 = 0$ and e_2 is evidence for rejecting A, or $e_1 = 1$ and e_2 is evidence for rejecting B.
- Evidence for rejecting $A \vee B$ is a pair (e_1, e_2) where e_1 is evidence for rejecting A and e_2 is evidence for rejecting B.
- Evidence for rejecting $A \rightarrow B$ is a pair (e_1, e_2) where e_1 is evidence for accepting A and e_2 is evidence for rejecting B.
- Evidence for rejecting $B \prec A$ is a method that converts evidence for rejecting A into evidence for rejecting B.
- Evidence for rejecting $\sim A$ is evidence for accepting A.
- $\emptyset$ is evidence for rejecting $\bot$.
- There is no evidence for rejecting $\top$.

Note that the previous clauses only state what constitutes evidence for accepting or rejecting compound formulas, based on evidence for accepting or rejecting their constituent parts. There is no clauses stating what constitute evidence for accepting or rejecting propositional variables because, as argued in Carnielli and Rodrigues [10], it depends on the area of study and the subject matter, and this is not a problem of logic.

In order to explicitly express evidence, based on the previous informal interpretation of $\mathsf{LET_C}$, lambda-terms from an extended lambda calculus (λ_C-calculus) are added to $\mathsf{LET_C}$, obtaining $\mathsf{LET^{\lambda}_C}$. The λ_C-calculus is an extension of the pure lambda-calculus with (two kinds of) pairs and projections, injections, case distinction, empty term, impossible (or exception) terms, and (two kinds of) identities.[1] $\mathsf{LET^{\lambda}_C}$ is really a type system whose types are the formulas of $\mathsf{LET_C}$ and whose terms are the elements of

[1] For a good introduction to lambda calculus and type theory, see Sørensen and Urzyczyn [24].

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

the λ_C-calculus. Two kinds of pairs and projections are used to ensure the uniqueness of types (module strong equivalence). The first kind will be used for conjunctions and negations of disjunctions, while the second kind will be used for co-implications and negation of implications. Pairs of the second kind will be called *combined pairs*, since they always have a term of a formula and a term of a negated formula. The two kinds of identities are used to differentiate terms of formulas and of their double strong negations.

To define the set of terms of the λ_C-calculus, we consider a denumerable set of variables V, whose elements represent the evidence that is assumed to exist. We shall use the last letters (possibly with subscripts) of the Latin alphabet $x, x_1, \ldots, y, y_1, \ldots$ to denote elements of V, and letters r, s, t to denote arbitrary λ_C-terms.

DEFINITION 3.1. The *set of terms of the λ_C-calculus*, denoted by Λ, is defined by the following grammar:

$$\begin{aligned}
\Lambda = {} & V \mid && \text{(variables)}\\
& \lambda V.\Lambda \mid \mathrm{ap}(\Lambda,\Lambda) \mid && \text{(abstractions and applications)}\\
& (\Lambda,\Lambda) \mid \pi_1(\Lambda) \mid \pi_2(\Lambda) \mid && \text{(pairs and projections)}\\
& \langle\Lambda,\Lambda\rangle \mid \pi_1^*(\Lambda) \mid \pi_2^*(\Lambda) \mid && \text{(combined pairs and projections)}\\
& \mathrm{in}_1(\Lambda) \mid \mathrm{in}_2(\Lambda) \mid && \text{(injections)}\\
& \text{case } \Lambda \text{ of } [V]\Lambda \text{ or } [V]\Lambda \mid && \text{(case distintion)}\\
& \emptyset \mid \mathcal{E}(\Lambda) \mid && \text{(empty and exception terms)}\\
& \mathrm{id}(\Lambda) \mid \mathrm{id}^{-1}(\Lambda) && \text{(identities).}
\end{aligned}$$

We shall use $r[x := s]$ to denote the result of substituting s for the free occurrences of x in r (changing bound variables to fresh variables if necessary to avoid creating new occurrences of bound variables). The β-reduction relation on Λ is defined as follows.

DEFINITION 3.2. The *one step β-reduction* relation on Λ, denoted by $\to_\beta$, is the least compatible relation on Λ with base rules:[2]

$$\begin{aligned}
\mathrm{ap}(\lambda x.r, s) &\to_\beta r[x := s]\\
\pi_1((r,s)) &\to_\beta r\\
\pi_2((r,s)) &\to_\beta s\\
\pi_1^*(\langle r,s\rangle) &\to_\beta r\\
\pi_2^*(\langle r,s\rangle) &\to_\beta s\\
\text{case } \mathrm{in}_1(t) \text{ of } [x]r \text{ or } [y]s &\to_\beta r[x := t]
\end{aligned}$$

[2]The compatibility of $\to_\beta$ allows to apply the base β-reduction rules to sub-terms; i.e., if $r \to_\beta s$, and if t' is the result of substituting s for an occurrence of r in t, then $t \to_\beta t'$. For a formal definition see Sørensen and Urzyczyn [24].

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

$$\text{case in}_2(t) \text{ of } [x]r \text{ or } [y]s \to_\beta s[y := t]$$
$$\text{id}(\text{id}^{-1}(t)) \to_\beta t$$
$$\text{id}^{-1}(\text{id}(t)) \to_\beta t.$$

The *multi-step β-reduction* relation on Λ, denoted by $\twoheadrightarrow_\beta$, is the reflexive and transitive closure of $\to_\beta$.

We shall use $\mathcal{F}(\mathsf{LET_C})$ to denote the set of formulas of $\mathsf{LET_C}$. The notions of statement, declaration, context and judgement are defined as usual in type systems: A *statement* is an expression of the form $t : A$, where $t \in \Lambda$ and $A \in \mathcal{F}(\mathsf{LET_C})$ (in such a statement t is called the *subject*); a *declaration* is a statement with a variable as subject; a *context* is a set of declarations with different subjects; and a *judgement* has the form $\Gamma \vdash t : A$, with Γ a context and $t : A$ a statement. A judgement $\Gamma \vdash t : A$ in $\mathsf{LET}^\lambda_\mathsf{C}$ means that there is a derivation of statement $t : A$, with open declarations (or assumptions) in Γ, using the rules in Table 2. Intuitively, $x_1 : B_1, \ldots, x_n : B_n \vdash t : A$ means that if we consider that x_i represents evidence for accepting B_i, for $1 \leq i \leq n$, then t represents evidence for accepting A (evidence for accepting $\sim B$ is the same that evidence for rejecting B).

The following theorem, which has a simple proof, shows that $\mathsf{LET}^\lambda_\mathsf{C}$ actually achieves the objective for which it was designed: to formalise evidence in an explicit way. Moreover, the lambda terms and their reduction rules give us procedures to calculate evidence of compound formulas from their constituent parts.

THEOREM 3.3. *$B_1, \ldots, B_n \vdash A$ in $\mathsf{LET_C}$ iff there is $t \in \Lambda$ such that $x_1 : B_1, \ldots, x_n : B_n \vdash t : A$ in $\mathsf{LET}^\lambda_\mathsf{C}$.*

PROOF. For the left to right direction, take the proof of $B_1, \ldots, B_n \vdash A$ in $\mathsf{LET_C}$ and add lambda terms according to the rules of $\mathsf{LET}^\lambda_\mathsf{C}$. For the right to left direction, take the proof of $x_1 : B_1, \ldots, x_n : B_n \vdash t : A$ and drop the lambda terms. ⊣

§4. Some properties of $\mathsf{LET}^\lambda_\mathsf{C}$. In this section, some usual properties of type systems are proven for $\mathsf{LET}^\lambda_\mathsf{C}$. Before that, some definitions are necessary.

DEFINITION 4.1. Let Γ be a context:

1. The *domain* of Γ, denoted by $\text{Dom}(\Gamma)$, is the set $\{x \mid x : A \in \Gamma, \text{ for some } A \in \mathcal{F}(\mathsf{LET_C})\}$.
2. If $X \subseteq V$, the *restriction* of Γ to X, denoted by $\Gamma{\restriction}X$, is the set $\{x : A \mid x : A \in \Gamma \text{ and } x \in X\}$.

The set of *free variables* of terms $t \in \Lambda$, denoted by $\text{FV}(t)$, is defined as usual (the only binding operators are λ in abstractions and $[\cdot]$ in case distinctions).

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

Table 2. Natural deduction system for $\mathsf{LET}^{\lambda}_{\mathsf{C}}$

$\dfrac{}{\emptyset : \top}(I^{\lambda}_{\top})$	$\dfrac{t : \bot}{\mathcal{E}(t) : A}(E^{\lambda}_{\bot})$
$\dfrac{r : A \quad s : B}{(r,s) : A \wedge B}(I^{\lambda}_{\wedge})$	$\dfrac{t : A \wedge B}{\pi_1(t) : A}(E1^{\lambda}_{\wedge}) \quad \dfrac{t : A \wedge B}{\pi_2(t) : B}(E2^{\lambda}_{\wedge})$
$\dfrac{t : A}{\mathrm{in}_1(t) : A \vee B}(I1^{\lambda}_{\vee})$	$[x : A] \quad [y : B]$
$\dfrac{t : B}{\mathrm{in}_2(t) : A \vee B}(I2^{\lambda}_{\vee})$	$\vdots \qquad \vdots$
$[x : A]$	$\dfrac{t : A \vee B \quad r : C \quad s : C}{\text{case } t \text{ of } [x]r \text{ or } [y]s : C}(E^{\lambda}_{\vee})$
$\vdots$	$\dfrac{r : A \to B \quad s : A}{\mathrm{ap}(r,s) : B}(E^{\lambda}_{\to})$
$\dfrac{t : B}{\lambda x.t : A \to B}(I^{\lambda}_{\to})$	
$\dfrac{r : \sim A \quad s : B}{\langle r,s \rangle : B \prec A}(I^{\lambda}_{\prec})$	$\dfrac{t : B \prec A}{\pi^*_1(t) : \sim A}(E1^{\lambda}_{\prec}) \quad \dfrac{t : B \prec A}{\pi^*_2(t) : B}(E2^{\lambda}_{\prec})$
$\dfrac{}{\emptyset : \sim \bot}(I^{\lambda}_{\sim \bot})$	$\dfrac{t : \sim \top}{\mathcal{E}(t) : A}(E^{\lambda}_{\sim \top})$
$\dfrac{t : \sim A}{\mathrm{in}_1(t) : \sim(A \wedge B)}(I1^{\lambda}_{\sim \wedge})$	$[x : \sim A] \quad [y : \sim B]$
$\dfrac{t : \sim B}{\mathrm{in}_2(t) : \sim(A \wedge B)}(I2^{\lambda}_{\sim \wedge})$	$\vdots \qquad \vdots$
	$\dfrac{t : \sim(A \wedge B) \quad r : C \quad s : C}{\text{case } t \text{ of } [x]r \text{ or } [y]s : C}(E^{\lambda}_{\sim \wedge})$
$\dfrac{r : \sim A \quad s : \sim B}{(r,s) : \sim(A \vee B)}(I^{\lambda}_{\sim \vee})$	$\dfrac{t : \sim(A \vee B)}{\pi_1(t) : \sim A}(E1^{\lambda}_{\sim \vee}) \quad \dfrac{t : \sim(A \vee B)}{\pi_2(t) : \sim B}(E2^{\lambda}_{\sim \vee})$
$\dfrac{r : A \quad s : \sim B}{\langle r,s \rangle : \sim(A \to B)}(I^{\lambda}_{\sim \to})$	$\dfrac{t : \sim(A \to B)}{\pi^*_1(t) : A}(E1^{\lambda}_{\sim \to}) \quad \dfrac{t : \sim(A \to B)}{\pi^*_2(t) : \sim B}(E2^{\lambda}_{\sim \to})$
$[x : \sim A]$	
$\vdots$	$\dfrac{r : \sim(B \prec A) \quad s : \sim A}{\mathrm{ap}(r,s) : \sim B}(E^{\lambda}_{\sim \prec})$
$\dfrac{t : \sim B}{\lambda x.t : \sim(B \prec A)}(I^{\lambda}_{\sim \prec})$	
$\dfrac{t : A}{\mathrm{id}(t) : \sim \sim A}(I^{\lambda}_{\sim \sim})$	$\dfrac{t : \sim \sim A}{\mathrm{id}^{-1}(t) : A}(E^{\lambda}_{\sim \sim})$

LEMMA 4.2 (Generation Lemma).

1. *If* $\Gamma \vdash x : A$, *then* $x : A \in \Gamma$.
2. *If* $\Gamma \vdash ap(r,s) : A$, *then there is* B *such that* $\Gamma \vdash r : B \to A$ *and* $\Gamma \vdash s : B$, *or there are* B *and* C *such that* $A = \sim C$, $\Gamma \vdash r : \sim(C \prec B)$ *and* $\Gamma \vdash s : \sim B$.

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

3. *If* $\Gamma \vdash \lambda x.t : A$, *then there are* B *and* C *such that* $A = B \to C$ *and* $\Gamma, x : B \vdash t : C$, *or such that* $A = {\sim}(C \prec B)$ *and* $\Gamma, x : {\sim} B \vdash t : {\sim} C$.
4. *If* $\Gamma \vdash (r, s) : A$, *then there are* B *and* C *such that* $A = B \wedge C$, $\Gamma \vdash r : B$ *and* $\Gamma \vdash s : C$, *or such that* $A = {\sim}(B \vee C)$, $\Gamma \vdash r : {\sim} B$ *and* $\Gamma \vdash s : {\sim} C$.
5. *If* $\Gamma \vdash \langle r, s \rangle : A$, *then there are* B *and* C *such that* $A = {\sim}(B \to C)$, $\Gamma \vdash r : B$ *and* $\Gamma \vdash s : {\sim} C$, *or such that* $A = C \prec B$, $\Gamma \vdash r : C$ *and* $\Gamma \vdash s : {\sim} B$.
6. *If* $\Gamma \vdash \pi_1(t) : A$ (*respectively* $\Gamma \vdash \pi_2(t) : A$), *then there is* B *such that* $\Gamma \vdash t : A \wedge B$ (*respectively* $\Gamma \vdash t : B \wedge A$), *or there are* B *and* C *such that* $A = {\sim} C$ *and* $\Gamma \vdash t : {\sim}(C \vee B)$ (*respectively* $\Gamma \vdash t : {\sim}(B \vee C)$).
7. *If* $\Gamma \vdash \pi_1^*(t) : A$ (*respectively* $\Gamma \vdash \pi_2^*(t) : A$), *then there are* B *and* C *such that* $A = {\sim} C$ *and* $\Gamma \vdash t : B \prec C$ (*respectively* $\Gamma \vdash t : {\sim}(B \to C)$), *or there is* B *such that* $\Gamma \vdash t : {\sim}(A \to B)$ (*respectively* $\Gamma \vdash t : A \prec B$).
8. *If* $\Gamma \vdash in_1(t) : A$ (*respectively* $\Gamma \vdash in_2(t) : A$), *then there are* B *and* C *such that* $A = (B \vee C)$ *and* $\Gamma \vdash t : B$ (*respectively* $\Gamma \vdash t : C$), *or such that* $A = {\sim}(B \wedge C)$ *and* $\Gamma \vdash t : {\sim} B$ (*respectively* $\Gamma \vdash t : {\sim} C$).
9. *If* $\Gamma \vdash$ *case* t *of* $[x]r$ *or* $[y]s : A$, *then there are* B *and* C *such that* $\Gamma \vdash t : B \vee C$, $\Gamma, x : B \vdash r : A$ *and* $\Gamma, y : C \vdash s : A$, *or such that* $\Gamma \vdash t : {\sim}(B \wedge C)$, $\Gamma, x : {\sim} B \vdash r : A$ *and* $\Gamma, y : {\sim} C \vdash s : A$.
10. *If* $\Gamma \vdash \mathcal{E}(t) : A$, *then* $\Gamma \vdash t : \bot$ *or* $\Gamma \vdash t : {\sim} \top$.
11. *If* $\Gamma \vdash id(t) : A$, *then there is* B *such that* $A = {\sim} {\sim} B$ *and* $\Gamma \vdash t : B$.
12. *If* $\Gamma \vdash id^{-1}(t) : A$, *then* $\Gamma \vdash t : {\sim} {\sim} A$.

PROOF. By inspection of the rules of $\mathsf{LET}^{\lambda}_{\mathsf{C}}$. ⊣

LEMMA 4.3 (Free Variables Lemma).

1. *If* $\Gamma \vdash t : A$, *then* $FV(t) \subseteq Dom(\Gamma)$.
2. *If* $\Gamma \vdash t : A$, *then* $\Gamma{\restriction}FV(t) \vdash t : A$.

PROOF. By structural induction on t. ⊣

LEMMA 4.4 (Uniqueness of Types). *If* $\Gamma \vdash t : A$ *and* $\Gamma \vdash t : B$, *then* $\vdash A \Leftrightarrow B$.

PROOF. By structural induction on t. ⊣

LEMMA 4.5 (Substitution Lemma). *If* $\Gamma, x : A \vdash r : B$ *and* $\Gamma \vdash s : A$, *then* $\Gamma \vdash r[x := s] : B$.

PROOF. By structural induction on r. ⊣

LEMMA 4.6 (Subject Reduction). *If* $\Gamma \vdash t : A$ *and* $t \twoheadrightarrow_\beta s$, *then* $\Gamma \vdash s : A$.

PROOF. First proving the case of one step β-reductions (i.e., for $t \to_\beta s$), the general case follows for induction on the number of β-reductions. As the relation $\to_\beta$ is a compatible relation on Λ, generated by some base rules

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

(see 3.2), the proof proceeds by induction on the generation of $\rightarrow_\beta$. For the base rules, we only show the case when $t = \text{case in}_1(r)$ of $[x]p$ or $[y]q$ and $s = p[x := r]$ (the other cases are similar or easier): From $\Gamma \vdash t : A$, by the Generation Lemma (item 9), we have two cases:

1. There is B and C such that $\Gamma \vdash \text{in}_1(r) : B \vee C$, $\Gamma, x : B \vdash p : A$ and $\Gamma, y : C \vdash q : A$: From $\Gamma \vdash \text{in}_1(r) : B \vee C$, by Generation Lemma (item 8), we have that $\Gamma \vdash r : B$. Then, from $\Gamma, x : B \vdash p : A$ and $\Gamma \vdash r : B$, by the Substitution Lemma, we have that $\Gamma \vdash p[x := r] : A$.
2. There is B and C such that $\Gamma \vdash \text{in}_1(r) : \sim(B \wedge C)$, $\Gamma, x : \sim B \vdash p : A$ and $\Gamma, y : \sim C \vdash q : A$: Similar to the previous case.

The proofs for the compatibility rules are routine (but lengthy) inductive steps. ⊣

DEFINITION 4.7. A term $t \in \Lambda$ is *legal* if there is a context Γ and a formula A such that $\Gamma \vdash t : A$.

THEOREM 4.8 (Normalisation Theorem). *Every legal term is strongly normalising.*

PROOF. Let us consider the simple typed λ-calculus extended with products, sums, empty type and unity type (denoted by $\wedge$, $\vee$, $\bot$, and $\top$, respectively). This calculus, which we will denote by E^λ, is a fragment of $\mathsf{LET}^\lambda_\mathsf{C}$ (E^λ is $\mathsf{LET}^\lambda_\mathsf{C}$ dropping the rules for co-implication and constructive negation), and can also be seen as a fragment of Intuitionistic Type Theory (ITT) (in the version presented in Martin-Löf [15], for instance). Therefore, the strong normalisation of ITT implies the strong normalisation of E^λ. A term translation function φ, from terms of the λ_C-calculus to terms of the extended λ-calculus used in E^λ, can be defined by:

$$\begin{aligned}
\varphi(x) &= x, & \varphi(\lambda x.t) &= \lambda x.\varphi(t),\\
\varphi(\text{ap}(r,s)) &= \text{ap}(\varphi(r),\varphi(s)), & \varphi((r,s)) &= (\varphi(r),\varphi(s)),\\
\varphi(\pi_i(t)) &= \pi_i(\varphi(t))\ (i \in \{1,2\}), & \varphi(\langle r,s\rangle) &= (\varphi(r),\varphi(s)),\\
\varphi(\pi_i^*(t)) &= \pi_i(\varphi(t))\ (i \in \{1,2\}), & \varphi(\text{in}_i(t)) &= \text{in}_i(\varphi(t))\ (i \in \{1,2\}),\\
\varphi(\text{case } t \text{ of } [x]r \text{ or } [y]s) &= \text{case } \varphi(t) \text{ of } [x]\varphi(r) \text{ or } [y]\varphi(s),\\
\varphi(\emptyset) &= \emptyset, & \varphi(\mathcal{E}(t)) &= \mathcal{E}(\varphi(t)),\\
\varphi(\text{id}(t)) &= \varphi(t), & \varphi(\text{id}^{-1}(t)) &= \varphi(t).
\end{aligned}$$

Considering that the types of E^λ are generated by the set of propositional variables $V' = V \cup \{\overline{p} \mid p \in V\}$, a type translation function ψ, from types of $\mathsf{LET}^\lambda_\mathsf{C}$ to types of E^λ, can be defined by:

$$\begin{aligned}
\psi(p) &= p, & \psi(\sim p) &= \overline{p},\\
\psi(\bot) &= \bot, & \psi(\top) &= \top,
\end{aligned}$$

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

$$\begin{aligned}
\psi(A\wedge B) &= \psi(A)\wedge\psi(B), & \psi(A\vee B) &= \psi(A)\vee\psi(B),\\
\psi(A\to B) &= \psi(A)\to\psi(B), & \psi(B\prec A) &= \psi(B)\wedge\psi(\sim A),\\
\psi(\sim\bot) &= \top, & \psi(\sim\top) &= \bot,\\
\psi(\sim(A\wedge B)) &= \psi(\sim A)\vee\psi(\sim B), & \psi(\sim(A\vee B)) &= \psi(\sim A)\wedge\psi(\sim B),\\
\psi(\sim(A\to B)) &= \psi(A)\wedge\psi(\sim B), & \psi(\sim(B\prec A)) &= \psi(\sim A)\to\psi(\sim B),\\
\psi(\sim\sim A) &= A.
\end{aligned}$$

By induction on t, it can be proven that if $\Gamma\vdash t:A$ in $\mathsf{LET_C}$, then $\{\varphi(s):\psi(B)\mid s:B\in\Gamma\}\vdash\varphi(t):\psi(A)$ in E^{λ}. Moreover, if t has an infinite reduction chain in λ_C-calculus, then $\varphi(t)$ has an infinite reduction chain in the extended λ-calculus used in E^{λ}. Consequently, as E^{λ} satisfies strong normalisation, then $\mathsf{LET}^{\lambda}_{\mathsf{C}}$ satisfies strong normalisation. ⊣

§5. Realisability interpretation for $\mathsf{LET_C}$. A *realisability interpretation* of *Heyting Arithmetic* (HA) was proposed in Kleene [14]. Under such interpretation, codes of recursive functions are assigned to provable formulas of HA, establishing in this way a connection between provability in HA and recursive functions, which in a certain way justifies that Int is indeed constructive (at least in the formalisation of arithmetic). Nelson [17] extends Kleene's notion of realisability interpreting arithmetic formulas by two recursively and simultaneously defined notions of *positive realisability (P-realisability)* and *negative realisability (N-realisability)*, and presents a formal system of number theory, to which we shall refer as *Nelson Arithmetic* (NA), that satisfies such realisability interpretations. NA is basically the result of substituting the axioms for (intuitionistic) negation in HA by axioms of a *constructive negation*, which formalises constructive falsity in a dual way of constructive truth. The logic axioms of NA are what comprise what is currently known as *Nelson's Logic*, and a variation of this logic (presented in Almukdad and Nelson [1]), in which the explosiveness of the constructive negation is avoided, is currently known as *Nelson's Paraconsistent Logic*. As previously mentioned, the propositional fragment of Nelson's paraconsistent logic is usually denoted by N4. Based on Nelson's notions of positive and negative realisabilty, dropping the clauses for elementary arithmetical formulas and for quantifiers, and adding some clauses for $\prec$, $\bot$ and $\top$, the realisability interpretation of formulas of $\mathsf{N4}^{\star}$ (and also of $\mathsf{LET_C}$) can be defined as follows, where $\mathrm{p}(\cdot,\cdot)$ denote a fixed primitive recursive bijection from $\mathbb{N}^2$ to $\mathbb{N}$ (a pair coding function), and φ_l is the partial recursive function with index l (under some fixed enumeration of the partial recursive functions).

DEFINITION 5.1. The following clauses define whether a natural number l *P-realises* or *N-realises* a formula of $\mathsf{N4}^{\star}$.

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

(1P) No l P-realises $\bot$.
(1N) No l N-realises $\top$.
(2P) l P-realises $\top$ (for every $l \in \mathbb{N}$).
(2N) l N-realises $\bot$ (for every $l \in \mathbb{N}$).
(3P) l P-realises $A \wedge B$ iff $l = \mathrm{p}(m, n)$, m P-realises A and n P-realises B.
(3N) l N-realises $A \wedge B$ iff $l = \mathrm{p}(m, n)$, $m = 0$ and n N-realises A, or $m > 0$ and n N-realises B.
(4P) l P-realises $A \vee B$ iff $l = \mathrm{p}(m, n)$, $m = 0$ and n P-realises A, or $m > 0$ and n P-realises B.
(4N) l N-realises $A \vee B$ iff $l = \mathrm{p}(m, n)$, m N-realises A and n N-realises B.
(5P) l P-realises $A \to B$ iff, for every m that P-realises A, $\varphi_l(m)$ P-realises B.
(5N) l N-realises $A \to B$ iff $l = \mathrm{p}(m, n)$, m P-realises A and n N-realises B.
(6P) l P-realises $B \prec A$ iff $l = \mathrm{p}(m, n)$, m N-realises A and n P-realises B.
(6N) l N-realises $B \prec A$ iff, for every m that N-realises A, $\varphi_l(m)$ N-realises B.
(7P) l P-realises $\sim A$ iff l N-realises A.
(7N) l N-realises $\sim A$ iff l P-realises A.

In order to make more intuitive the relationship between P-realisability and provability in $\mathsf{LET}_{\mathsf{C}}$ (Theorem 5.2 below), and also to establish a connection with our explicit representation of evidence by means of λ_C-terms, we shall consider that evidence is codified by natural numbers, thus every assignation of natural numbers to free variables of λ_C-terms, in a deduction in $\mathsf{LET}^{\lambda}_{\mathsf{C}}$, will give as a particular justification of such deduction based on evidence.

THEOREM 5.2. *For every natural number $n \geq 1$, if $B_1, \ldots, B_n \vdash A$, there is a recursive function $h : \mathbb{N}^n \to \mathbb{N}$ such that, if the natural numbers $l_1, \ldots, l_n$ P-realise $B_1, \ldots, B_n$, respectively, then $h(l_1, \ldots, l_n)$ P-realises A.*

PROOF. Let us suppose that $B_1, \ldots, B_n \vdash A$. By Theorem 3.3, there is $t \in \Lambda$ such that $x_1 : B_1, \ldots, x_n : B_n \vdash t : A$ in $\mathsf{LET}^{\lambda}_{\mathsf{C}}$. We will suppose that the variables $x_1, \ldots, x_n$ range over $\mathbb{N}$, thus any substitution of natural numbers $l_1, \ldots, l_n$ for variables $x_1, \ldots, x_n$, in the context $x_1 : B_1, \ldots, x_n : B_n$, will give as a particular assumption of realisers of the formulas $B_1, \ldots, B_n$. Now, we will show the existence of the function h in the statement of the theorem, by structural induction on t, where the context $x_1 : B_1, \ldots, x_n : B_n$ will be denoted by Γ, and the sequence of natural numbers $l_1, \ldots, l_n$ will be denoted by $\overrightarrow{l}$. The similarity in the proofs of subcases is by taking into account that a natural number l P-realises the (constructive) negation of a formula iff it N-realises the formula.

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

1. If $t \in V$: By the Generation Lemma (item 1), there is $i \in \{1, \ldots, n\}$ such that $t = x_i$ and $A = B_i$, then h is the projection function defined by $h(\overrightarrow{l}) = l_i$.
2. If $t = \mathrm{ap}(r, s)$: By the Generation Lemma (item 2), we have the following two cases:
   (a) There is B such that $\Gamma \vdash r : B \to A$ and $\Gamma \vdash s : B$: By inductive hypothesis there are recursive functions $h_1 : \mathbb{N}^n \to \mathbb{N}$ and $h_2 : \mathbb{N}^n \to \mathbb{N}$ such that, if $\overrightarrow{l}$ P-realise $B_1, \ldots, B_n$, respectively, then $h_1(\overrightarrow{l})$ P-realises $B \to A$ and $h_2(\overrightarrow{l})$ P-realises B. Then $\varphi_{h_1(\overrightarrow{l})}(h_2(\overrightarrow{l}))$ P-realises A. The function h can then be defined by $h(\overrightarrow{l}) = U^1(h_1(\overrightarrow{l}), h_2(\overrightarrow{l}))$, where U^1 is the universal partial recursive function with one parameter given by the Kleene's Normal Form Theorem (see, for instance, [11, Section 16.E.]). Then: $U^1(h_1(\overrightarrow{l}), h_2(\overrightarrow{l})) = \varphi_{h_1(\overrightarrow{l})}(h_2(\overrightarrow{l}))$.
   (b) There are B and C such that $A = \sim C$, $\Gamma \vdash r : \sim(C \prec B)$ and $\Gamma \vdash s : \sim B$: Similar to the previous case.
3. If $t = \lambda x.r$: By the Generation Lemma (item 3), we have the following two cases:
   (a) There are B and C such that $A = B \to C$ and $\Gamma, x : B \vdash r : C$: By inductive hypothesis there is a recursive function $h_1 : \mathbb{N}^{n+1} \to \mathbb{N}$ such that, if $\overrightarrow{l}, l$ P-realise $B_1, \ldots, B_n, B$, respectively, then $h_1(\overrightarrow{l}, l)$ P-realises C. Let $h_1 = \varphi_z^{n+1}$, by Kleene's s-m-n Theorem (see, for instance, [11, Section 16.F.]), there is a recursive function s_1^n such that $\varphi_z^{n+1}(\overrightarrow{l}, l) = \varphi^1_{s_1^n(z, \overrightarrow{l})}(l)$. The function h can then be defined by $h(\overrightarrow{l}) = s_1^n(z, \overrightarrow{l})$.
   (b) There are B and C such that $A = \sim(C \prec B)$ and $\Gamma, x : \sim B \vdash t : \sim C$: Similar to the previous case.
4. If $t = (r, s)$: By the Generation Lemma (item 4), we have the following two cases:
   (a) There are B and C such that $A = B \wedge C$, $\Gamma \vdash r : B$ and $\Gamma \vdash s : C$: By inductive hypothesis there are recursive functions $h_1 : \mathbb{N}^n \to \mathbb{N}$ and $h_2 : \mathbb{N}^n \to \mathbb{N}$ such that, if $\overrightarrow{l}$ P-realise $B_1, \ldots, B_n$, respectively, then $h_1(\overrightarrow{l})$ P-realises B and $h_2(\overrightarrow{l})$ P-realises C. The function h can then be defined by $h(\overrightarrow{l}) = \mathrm{p}(h_1(\overrightarrow{l}), h_2(\overrightarrow{l}))$.
   (b) There are B and C such that $A = \sim(B \vee C)$, $\Gamma \vdash r : \sim B$ and $\Gamma \vdash s : \sim C$: Similar to the previous case.
5. If $t = \langle r, s \rangle$: Similar to the previous case.
6. If $t = \pi_1(r)$ (respectively $t = \pi_2(r)$): By the Generation Lemma (item 6), we have the following two cases:

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

(a) There is B such that $\Gamma \vdash r : A \wedge B$ (respectively $\Gamma \vdash r : B \wedge A$): By inductive hypothesis there is a recursive function $h_1 : \mathbb{N}^n \to \mathbb{N}$ such that, if $\overrightarrow{l}$ P-realise $B_1, \ldots, B_n$, then $h_1(\overrightarrow{l})$ P-realises $A \wedge B$ (respectively $B \wedge A$). The function h can then be defined by $h(\overrightarrow{l}) = \mathrm{u}_1(h_1(\overrightarrow{l}))$ (respectively $h(\overrightarrow{l}) = \mathrm{u}_2(h_1(\overrightarrow{l}))$), where $\mathrm{u}_1(\cdot)$ (respectively $\mathrm{u}_2(\cdot)$) is a fixed recursive unpairing function of the first coordinate (respectively the second coordinate).

(b) There are B and C such that $A = \sim C$ and $\Gamma \vdash r : \sim(C \vee B)$ (respectively $\Gamma \vdash r : \sim(B \vee C)$): Similar to the previous case.

7. If $t = \pi_1^*(r)$ (respectively $t = \pi_2^*(r)$): Similar to the previous case.
8. If $t = \mathrm{in}_1(r)$ (respectively $t = \mathrm{in}_2(r)$): By the Generation Lemma (item 8), we have the following two cases:
   (a) There are B and C such that $A = (B \vee C)$ and $\Gamma \vdash r : B$ (respectively $\Gamma \vdash r : C$): By inductive hypothesis there is a recursive function $h_1 : \mathbb{N}^n \to \mathbb{N}$ such that, if $\overrightarrow{l}$ P-realise $B_1, \ldots, B_n$, then $h_1(\overrightarrow{l})$ P-realises B (respectively C). The function h can then be defined by $h(\overrightarrow{l}) = \mathrm{p}(0, h_1(\overrightarrow{l}))$ (respectively $h(\overrightarrow{l}) = \mathrm{p}(1, h_1(\overrightarrow{l}))$).
   (b) There are B and C such that $A = \sim(B \wedge C)$ and $\Gamma \vdash r : \sim B$ (respectively $\Gamma \vdash r : \sim C$): Similar to the previous case.
9. If $t = \mathrm{case}\ q\ \mathrm{of}\ [x]r\ \mathrm{or}\ [y]s$: By the Generation Lemma (item 9), we have the following two cases:
   (a) There are B and C such that $\Gamma \vdash q : B \vee C$, $\Gamma, x : B \vdash r : A$ and $\Gamma, y : C \vdash s : A$: By inductive hypothesis there are recursive functions $h_1 : \mathbb{N}^n \to \mathbb{N}$, $h_2 : \mathbb{N}^{n+1} \to \mathbb{N}$ and $h_3 : \mathbb{N}^{n+1} \to \mathbb{N}$ such that, if $\overrightarrow{l}, l_{n+1}, l_{n+2}$ P-realise $B_1, \ldots, B_n, B, C$, respectively, then $h_1(\overrightarrow{l})$ P-realises $B \vee C$, $h_2(\overrightarrow{l}, l_{n+1})$ P-realises A and $h_3(\overrightarrow{l}, l_{n+2})$ P-realises A. The function h can then be defined by

$$h(\overrightarrow{l}) = \begin{cases} h_2(\overrightarrow{l}, \mathrm{u}_2(h_1(\overrightarrow{l}))) & \text{if } \mathrm{u}_1(h_1(\overrightarrow{l})) = 0 \\ h_3(\overrightarrow{l}, \mathrm{u}_2(h_1(\overrightarrow{l}))) & \text{if } \mathrm{u}_1(h_1(\overrightarrow{l})) \neq 0. \end{cases}$$

   (b) There are B and C such that $\Gamma \vdash q : \sim(B \wedge C)$, $\Gamma, x : \sim B \vdash r : A$ and $\Gamma, y : \sim C \vdash s : A$: Similar to the previous case.
10. If $t = \mathcal{E}(r)$: By the Generation Lemma (item 10), we have the following two cases:
    (a) $\Gamma \vdash r : \bot$: By inductive hypothesis there is a recursive function $h_1 : \mathbb{N}^n \to \mathbb{N}$ such that, if $\overrightarrow{l}$ P-realise $B_1, \ldots, B_n$, then $h_1(\overrightarrow{l})$ P-realises $\bot$. As $\bot$ have no P-realisers, then there are no P-realisers for all formulas $B_1, \ldots, B_n$. Consequently, h can be h_1 (or any n-ary recursive function).
    (b) $\Gamma \vdash r : \sim \top$: Similar to the previous case.

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

11. If $t = \mathrm{id}(r)$: By the Generation Lemma (item 11), there is B such that $A = \sim\sim B$ and $\Gamma \vdash t : B$. By inductive hypothesis there is a recursive function $h_1 : \mathbb{N}^n \to \mathbb{N}$ such that, if $\overrightarrow{l}$ P-realise $B_1, \ldots, B_n$, then $h_1(\overrightarrow{l})$ P-realises B. As a natural number l P-realises B iff l N-realises $\sim B$, and iff l P-realises $\sim\sim B$, then h can be h_1.
12. If $t = \mathrm{id}^{-1}(r)$: By the Generation Lemma (item 12) $\Gamma \vdash r : \sim\sim A$. By inductive hypothesis there is a recursive function $h_1 : \mathbb{N}^n \to \mathbb{N}$ such that, if $\overrightarrow{l}$ P-realise $B_1, \ldots, B_n$, then $h_1(\overrightarrow{l})$ P-realises $\sim\sim A$. As a natural number l P-realises $\sim\sim A$ iff l N-realises $\sim A$, and iff l P-realises A, then h can be h_1. ⊣

COROLLARY 5.3. *If $B_1, \ldots, B_n \vdash A$ and it is supposed that there are natural numbers $l_1, \ldots, l_n$ that P-realise $B_1, \ldots, B_n$, respectively, then there is a natural number l that P-realises A (particularly, if $\vdash A$, then there is a natural number l that P-realises A).*

PROOF. If $n \geq 1$, the result immediately follows from Theorem 5.2. If $n = 0$ (i.e., if $\vdash A$), we proceed by structural induction on A (taking into account that A cannot be $\bot$, nor a propositional variable or a (constructive) negation of a propositional variable):

1. If $A = \top$, any natural number l P-realises A.
2. If $A = B \wedge C$, then $\vdash B$ and $\vdash C$, and by inductive hypothesis there are l_1 and l_2 that P-realise B and C, respectively. Consequently $\mathrm{p}(l_1, l_2)$ P-realises A.
3. If $A = B \vee C$, as $\mathsf{LET_C}$ satisfies de disjunction property, then $\vdash B$ or $\vdash C$. In the first case, by inductive hypothesis there is l that P-realises B, then $\mathrm{p}(0, l)$ P-realises A. In the second case, by inductive hypothesis there is l that P-realises C, then $\mathrm{p}(1, l)$ P-realises A.
4. If $A = B \to C$, then $B \vdash C$. By Theorem 5.2, there exists a recursive function $h : \mathbb{N} \to \mathbb{N}$ such that, if l P-realises B, then $h(l)$ P-realises C. Consequently, any index of h (under the fixed enumeration of the partial recursive functions) P-realises $A \to B$.
5. If $A = C \prec B$, then $\vdash \sim B$ and $\vdash C$, and by inductive hypothesis there are l_1 and l_2 that P-realise $\sim B$ and C, respectively. Consequently $\mathrm{p}(l_1, l_2)$ P-realises A.
6. The cases where $A = \sim B$, with B a compound formula, are similar to the previous cases. ⊣

Note that there are no clauses for the P-realisation or the N-realisation of propositional variables, consequently formulas $p \vee \sim p$ and $\sim(p \wedge \sim p)$, with $p \in V$, have no P-realisers. This justifies the paracomplete and paraconsistent character of $\mathsf{LET_C}$.

In Rose [22], it is proved that there exists an intuitionistic propositional formula for which the substitution of number-theoretic formulas for its

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

propositional variables produces realisable but not intuitionistically provable formulas. Although our definition of P-realisability and N-realisability of formulas of $\mathsf{N4}^\star$ does not depend on an arithmetic (or any other) theory, Rose's result and the fact that $\mathsf{N4}^\star$ is a conservative extension of intuitionistic propositional logic (cf. [18, Corollary 8.6.6]) are strong reasons to believe that the reciprocal of Corollary 5.3 and of Theorem 5.2 are not valid.

§6. On the constructivity of $\mathsf{LET_C}$. As stated in [25, pp. 341–342]:

> Sometimes the term "constructive logic" is used as a synonym for "intuitionistic logic". However, logics other than intuitionistic logic have also been said to be constructive, like, for instance, Johansson's minimal logic, Heyting–Brouwer logic, or David Nelsons's logics with strong negation. Whereas there exists *the* system of classical propositional and predicate logic, it is far from clear whether there exists exactly one system of constructive logic. In a situation where there are no clear, agreed-upon, individually necessary and jointly sufficient conditions for the constructiveness of a logical system, it seems quite difficult or next to pointless to designate one particular logic as *the* correct constructive logic. Nevertheless, for some reasons certain logics may still be regarded as constructive logics.

Some reasons to justify that $\mathsf{LET_C}$ is a constructive logic are the following:

- $\mathsf{LET_C}$ does not satisfy the *Principle of Excluded Middle (PEM)*. According to [16, p. 27], "Under the constructive interpretation, the law of excluded middle is not an empty "tautology," but expresses the decidability of proposition A." The failure of PEM is then related to the existence of undecidable propositions, and is usually used to justify that a logic is constructive. The decidability of a particular proposition A can be expressed in $\mathsf{LET_C}$ by $\bigstar A$.
- $\mathsf{LET_C}$ satisfies the *disjunction property* (i.e., if $\vdash A \vee B$, then $\vdash A$ or $\vdash B$).[3] This is another property usually used to justify that a logic is constructive.
- $\mathsf{LET_C}$ also satisfies the *(negation of) conjunction property* (i.e., if $\vdash \sim(A \wedge B)$, then $\vdash \sim A$ or $\vdash \sim B$),[4] which can be viewed as a constructive property dual to the disjunction property.
- As it was shown in Section 3, $\mathsf{LET_C}$ admits a BHK-style interpretation where evidence for accepting or rejecting compound propositions

[3] It is well-known that $\mathsf{N4}$ satisfies the disjunction property, and the addition of the rules for $\bot$, $\top$ and $\prec$ do not destroy this property.

[4] The conjunction property is an immediate consequence of the disjunction property and the fact that $\vdash \sim(A \wedge B) \Leftrightarrow (\sim A \vee \sim B)$.

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

are treated as constructions, in an analogous way as the BHK-interpretation for Int offers a constructive interpretation of that logic in terms of proofs.

- The system $\mathsf{LET}_{\mathsf{C}}^{\lambda}$ provides an explicit representation of evidence by means of λ_C-calculus terms, which represent algorithmic procedures to calculate evidence of compound propositions base on evidence for their constituent parts.
- The realisability interpretation provided for $\mathsf{LET}_{\mathsf{C}}$, Theorem 5.2 and Corollary 5.3 establish a strong connection between derivations in $\mathsf{LET}_{\mathsf{C}}$ and recursive functions, which also highlights the algorithmic nature of the evidence building process in $\mathsf{LET}_{\mathsf{C}}$.

Since the classical negation of a proposition is true iff the proposition is false, classical negation expresses falsity. Moreover, the classical negation of a proposition A can be defined as the intuitionistic negation (i.e., as $A \to \bot$), which can be interpreted as the impossibility of A be true. Consequently, in classical logic there is no distinction between falsity and impossibility of truth. Intuitionistically, it is less clear whether or not there is a distinction between falsity and impossibility of truth. Although Shramko [23] recognises that "There is a tradition to present intuitionistic falsity of a sentence as truth of its (intuitionistic) negation." and claims that "Brouwer and Heyting seem not to have any special conception of falsity as a philosophical (or semantic) notion. Whenever they occasionally speak of falsity, they simply mean intuitionistic negation.", he proposes a "genuine intuitionistic notion of falsity" which is not captured by the intuitionistic negation. On the other hand, as shown in Section 2, in $\mathsf{LET}_{\mathsf{C}}$ the notions of falsity and impossibility of truth are clearly separated.

From a constructive point of view, it seems clear that Int allows only direct proofs to establish the truth of propositions. However, if it is accepted that intuitionistic negation represents intuitionistic falsity, then Int allows only indirect proofs to establish the falsity of propositions. Consequently, if by a "constructive proof" we mean a "direct proof", under the interpretation of negation as falsity, Int allows only constructive proofs to establish the truth of propositions and no constructive proofs to establish the falsity of propositions. However, if we understand intuitionistic negation as impossibility of truth, Int not only allows constructive (or direct) proofs to establish the truth of propositions, but also to establish the impossibility of the truth of propositions. In our view, the correct interpretation of intuitionistic negation is the later one, and we agree with Shramko that intuitionistic negation does not represent falsity. The failure of the (negation of) conjunction property in Int can be seen as a consequence of the fact that the intuitionistic negation does not represent falsity.

In $\mathsf{LET}_{\mathsf{C}}$, the constructive negation represents falsity (i.e., $\sim A$ can be interpreted as "A is false"), the intuitionistic negation represents

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

impossibility of truth (i.e., $\neg A$ must be interpreted as "it is impossible that A be true"), and the constructive negation of co-intuitionistic negation represents impossibility of falsity (i.e., $\sim\!\vdash A$ must be interpreted as "it is impossible that A be false"). As it was shown in Section 2, these notions are separated in $\mathsf{LET_C}$. From a constructive point of view, $\mathsf{LET_C}$ allows only direct proofs to establish the truth of propositions and the impossibility of the truth of propositions, and also allows only direct proofs to establish the falsity of propositions and the impossibility of the falsity of propositions. Taking "direct proof" as a synonym for "constructive proof", it follows that $\mathsf{LET_C}$ is constructively richer than Int.

Intuitively, it seems reasonable to infer the falsity of a proposition from the impossibility of its truth, and the truth of a proposition from the impossibility of its falsity, which leads us to consider the inclusion of the following rules to $\mathsf{LET_C}$ (taking into account that $\sim\!\vdash A$ and $\neg\sim A$ are strongly equivalent in $\mathsf{LET_C}$):

$$\frac{\neg A}{\sim A}\,(E_{\neg}) \qquad\qquad \frac{\neg\sim A}{A}\,(E_{\neg\sim}).$$

However, it is easy to show that the inclusion of these rules makes admissible the following rules:

$$\begin{array}{cc} [\sim A] & [A] \\ \vdots & \vdots \\ \dfrac{\bot}{A}\,(RA_1) & \dfrac{\bot}{\sim A}\,(RA_2) \end{array}$$

which can be seen as indirect proofs (or proofs by *reductio ad absurdum*) for the truth or the falsity of A, respectively. Using the rule $(E_{\sim\sim})$, it can be proven that $(E_{\neg})$ makes admissible $(E_{\neg\sim})$, and that $(E_{\neg\sim})$ makes admissible $(E_{\neg})$. Consequently, the addition of either of these two rules to $\mathsf{LET_C}$ will cause the system to lose its constructivist properties.

It can also be considered reasonable that the falsity of a proposition implies the impossibility of its truth, and that the truth of a proposition implies the impossibility of its falsity, which would lead us to consider adding the following rules to $\mathsf{LET_C}$:

$$\frac{\sim A}{\neg A}\,(I_{\neg}) \qquad\qquad \frac{A}{\neg\sim A}\,(I_{\neg\sim}).$$

It can also be proven that the addition of $(I_{\neg})$ makes admissible $(I_{\neg\sim})$, and that the addition of $(I_{\neg\sim})$ makes admissible $(I_{\neg})$. Moreover, it is evident that the addition of $(I_{\neg})$ to $\mathsf{LET_C}$ will make the constructive negation explosive. Consequently, the addition of $(I_{\neg})$ or $(I_{\neg\sim})$ to $\mathsf{LET_C}$ will result in the system becoming non-paraconsistent, and no longer able to formalise deductions in contexts with conflicting information.

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

§7. Concluding remarks and future work. This article introduces the constructive logic of evidence and truth, $\mathsf{LET_C}$, a new system within the family of LETs. The logic $\mathsf{LET_C}$ extends $\mathsf{N4^\star}$ (an extension of Nelson's paraconsistent logic) by adding connectives for consistency, inconsistency, determinedness, and undeterminedness. Unlike other LETs, which feature a classicality operator that simultaneously restores the consistency and determinacy of certain propositions, $\mathsf{LET_C}$ provides distinct connectives for the independent recovery of these properties.

Lambda-calculus terms were incorporated into $\mathsf{LET_C}$, resulting in the type system $\mathsf{LET_C^\lambda}$, which provides an explicit formalisation of evidence. This approach contrasts with the one presented in Fitting [12]. In $\mathsf{LET_C^\lambda}$, lambda-calculus terms not only represent evidence but also describe algorithmic procedures for deriving evidence for compound formulas based on evidence for their constituent parts.

The Justification Logics proposed in Artemov [5] generalize the Logic of Proofs ($\mathsf{LP_0}$), introduced by the same author in Artemov [3] and further developed in Artemov [4] and Artemov and Fitting [6]. $\mathsf{LP_0}$ formalises the Brouwer–Heyting–Kolmogorov (BHK) provability interpretation of Intuitionistic Logic, with proofs explicitly represented as 'proof polynomials.' This formalisation leads to an embedding within Peano Arithmetic. However, as noted in the introduction, the goals and methods of $\mathsf{LET_C}$ differ significantly from those of justification logics, particularly from $\mathsf{LP_0}$. Nevertheless, it remains an open possibility to apply $\mathsf{LP_0}$-like techniques to formalise the BHK-style interpretation of $\mathsf{LET_C}$, where 'evidence' is understood as 'proof' within Nelson's arithmetic system.

In addition to the algorithmic interpretation offered by the lambda-calculus terms in $\mathsf{LET_C^\lambda}$, a realisability interpretation is also offered for $\mathsf{LET_C}$ which highlights the constructive properties of this logic, establishing a close relationship between deductions and recursive functions.

As previously stated in Section 6, $\mathsf{LET_C}$ is a logic that extends Int, generating new constructivist properties. $\mathsf{LET_C}$ enables not only constructive demonstrations for the truth and the impossibility of truth of propositions, but also for the falsity and the impossibility of falsity of propositions. The separation between truth and impossibility of falsity, and between falsity and impossibility of truth, is a key component for $\mathsf{LET_C}$ to achieve its constructivist properties.

Extending $\mathsf{LET_C}$ to a first-order logic is not particularly difficult. This can be achieved by incorporating the rules for the quantifiers $\forall$ and $\exists$, along with their negations, as done in Nelson's logic with quantifiers. Additionally, the rules for dependent types (Π-types and Σ-types) allow to extend $\mathsf{LET_C^\lambda}$ to a system that includes types for the quantifiers and their negations. The realisability interpretation also appears to extend naturally to a first-order version of $\mathsf{LET_C}$. However, these extensions are not presented in this article,

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

as we believe that the most interesting aspects of our proposal are already captured at the propositional level.

As mentioned in the Introduction, LET_C formalises an instantaneous view of evidence-based deductions, where reasoning is based solely on the current available information. To capture a more dynamic perspective—where an agent can revise or incorporate new evidence based on their deductions—it may be valuable to develop a meta-reasoning architecture (meta-logic) for LET_C. This meta-reasoning could allow for the modification of hypotheses (or theories) according to reasonable criteria. Such an approach would separate reasoning based on existing evidence (formalised within the logical system) from reasoning about the evidence itself (which would be formalised in the meta-logic). This presents an exciting avenue for future research.

Although much theoretical and technical work remains before LET_C can be applied to the automation of reasoning in contexts involving conflicting and insufficient information, it is worth noting that the concept of constructive justification may be particularly well-suited for exploring algorithmic explainability in the context of Artificial Intelligence.

Acknowledgments. Carnielli acknowledges the intellectual support of Chapman University during a research stay in the Fall semester of 2024. The authors also thank an anonymous and dedicated referee for providing several insightful comments that contributed to improving this work.

Funding. Juan C. Agudelo-Agudelo acknowledges support from the University of Antioquia, Medellín, Colombia (grant number 2023/60030), and from the São Paulo Research Foundation (FAPESP), Brazil (grant number 2020/16353-3) for the preparation of this paper. Walter Carnielli acknowledges support from the National Council for Scientific and Technological Development (CNPq), Brazil (grant 03780/2022-3), and from the São Paulo Research Foundation (FAPESP) Brazil [grant number 2020/16353-3] for the preparation of this paper.

REFERENCES

[1] A. Almukdad and D. Nelson, *Constructible falsity and inexact predicates*. ***The Journal of Symbolic Logic***, vol. 49 (1984), no. 1, pp. 231–233.

[2] H. Antunes, W. A. Carnielli, A. Kapsner, and A. Rodrigues, *Kripke-style models for logics of evidence and truth*. ***Axioms***, vol. 9 (2020), no. 3, p. 100.

[3] S. Artemov, *Logic of proofs*. ***Annals of Pure and Applied Logic***, vol. 67 (1994), pp. 29–59.

[4] ———, *Explicit provability and constructive semantics*. ***The Bulletin of Symbolic Logic***, vol. 7 (2001), no. 1, pp. 1–36.

[5] ———, *The logic of justification*. ***The Review of Symbolic Logic***, vol. 1 (2008), no. 4, pp. 477–513.

[6] S. Artemov and M. Fitting, ***Justification Logic: Reasoning with Reasons***, Cambridge Tracts in Mathematics, 216, Cambridge University Press, New York, 2019.

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press

[7] J. van Benthem and E. Pacuit, *Dynamic logics of evidence-based beliefs*. ***Studia Logica***, vol. 99 (2011), pp. 61–92.

[8] W. A. Carnielli and J. Bueno-Soler, *Where the truth lies: A paraconsistent approach to Bayesian epistemology*. ***Studia Logica***, vol. In print (2024).

[9] W. A. Carnielli, L. Frade, A. Rodrigues, and J. Bueno-Soler, *On factive and non-factive evidence: Combining the modal logics S4 and KX4*, unpublished manuscript, 2024.

[10] W. A. Carnielli and A. Rodrigues, *An epistemic approach to paraconsistency: A logic of evidence and truth*. ***Synthese***, vol. 196 (2019), pp. 3789–3813.

[11] R. L. Epstein and W. A . Carnielli, ***Computability: Computable Functions, Logic, and the Foundations of Mathematics***, third ed., Advanced Reasoning Forum, Socorro, 2008.

[12] M. Fitting, *Paraconsistent logic, evidence and justification*. ***Studia Logica***, vol. 105 (2017), pp. 1149–1166.

[13] N. Kamide, *Natural deduction systems for nelson's paraconsistent logic and its neighbors*. ***Journal of Applied Non-Classical Logics***, vol. 15 (2005), pp. 405–435.

[14] S. C. Kleene, *On the interpretation of intuitionistic number theory*. ***The Journal of Symbolic Logic***, vol. 10 (1945), no. 4, pp. 109–124.

[15] P. Martin-Löf, *Intuitionistic type theory*, Technical report, Bibliopolis, Napoli, 1984, Notes by Giovanni Sambin of a series of lectures given in Padua 1980.

[16] S. Negri and J. von Plato, ***Structural Proof Theory***, Cambridge University Press, New York, 2001.

[17] D. Nelson, *Constructible falsity*. ***The Journal of Symbolic Logic***, vol. 14 (1949), no. 1, pp. 16–26.

[18] S. P. Odintsov, ***Constructive Negations and Paraconsistency***, Trends in Logic, 26, Springer, Dordrecht, 2008.

[19] G. Priest, ***In Contradiction: A Study of the Transconsistent***, Martinus Nijhoff, Dordrecht, 1987.

[20] A. Rodrigues, J. Bueno-Soler, and W. A. Carnielli, *Measuring evidence: A probabilistic approach to an extension of Belnap-Dunn logic*. ***Synthese***, vol. 198 (2021), no. 22, pp. 5451–5480.

[21] A. Rodrigues, M. E. Coniglio, H. Antunes, J. Bueno-Soler, and W. A. Carnielli, *Paraconsistency, evidence, and abduction*, ***Handbook of Abductive Cognition*** (L. Magnani, editor), Springer, Cham, 2023, pp. 313–350.

[22] G. F. Rose, *Propositional calculus and realizability*. ***Transactions of the American Mathematical Society***, vol. 75 (1953), no. 1, pp. 1–19.

[23] Y. Shramko, *What is a genuine intuitionistic notion of falsity?* ***Logic and Logical Philosophy***, vol. 21 (2012), pp. 3–23.

[24] M. H. Sørensen and P. Urzyczyn, ***Lectures on the Curry-Howard isomorphism***, Studies in Logic and the Foundations of Mathematics, 149, Elsevier, Amsterdam, 2006.

[25] H. Wansing, *Constructive negation, implication and co-implication*. ***Journal of Applied Non-Classical Logics***, vol. 18 (2008), nos. 2–3, pp. 341–364.

DEPARTMENT OF MATHEMATICS
UNIVERSITY OF ANTIOQUIA
MEDELLIN, 050010, COLOMBIA
E-mail: juan.agudelo9@udea.edu.co

CENTRE FOR LOGIC
EPISTEMOLOGY AND THE HISTORY OF SCIENCES
STATE UNIVERSITY OF CAMPINAS
CAMPINAS, 13083-852, BRAZIL
E-mail: walterac@unicamp.br

https://doi.org/10.1017/bsl.2025.10 Published online by Cambridge University Press