Light Dialectica Revisited

Mircea-Dan Hernest[1] and Trifon Trifonov[2,⋆]

[1] Informatics Institute, University of Innsbruck, Austria
dan.hernest@uibk.ac.at
[2] Mathematics Institute, University of Munich, Germany
trifonov@math.lmu.de

Abstract. We upgrade the light Dialectica interpretation [6] by adding two more light universal quantifiers, which are both semi-computational and semi-uniform and complement each other. An illustrative example is presented for the new light quantifiers and a new application is given for the older uniform quantifier. The realizability of new light negative formulations for the Axiom of Choice and for the Independence of Premises is explored in the new setting.

1 Introduction

We extend the light Dialectica (LD) interpretation [7] formulated for arithmetic without strong existential quantifiers. The reason for disregarding the intuitionistic existentials is that we are only interested in program extraction from classical proofs such that the translated proofs are classical as well. Thus in both the input and the verifying systems the existential quantifiers are defined in terms of the corresponding universal quantifiers. From here on *quantifier* will mean "universal quantifier" and *existential quantifier* will refer to the corresponding weak/classical existential quantifier.

Besides the non-computational quantifier $\forall_\emptyset$ (which is fully *uniform*, in the sense of Berger [2], and was denoted $\overline{\forall}$ in [7]) we consider two more mutually complementary light quantifiers defined as follows:

$[+]$ the positively computational but negatively uniform quantifier $\forall_+$;

$[-]$ the negatively computational but positively uniform quantifier $\forall_-$.

Thus $\forall_\emptyset$ is the "lightest" quantifier, since it has no computational content at all, be it positive or negative. For expository reasons we will name "light" also the semi-uniform / semi-computational quantifiers $\forall_+$ and $\forall_-$. The usual universal quantifier will be denoted $\forall_\pm$ in the input system NA_l and simply $\forall$ in the verifying system NA. The reason for this distinction is twofold: on one hand we want to make a clear separation between the input and the verifying system and on the other hand we want to stress that the regular universal quantifier is fully computational (both positively and negatively) when interpreted by the light Dialectica. We denote the existential counterparts of all five quantifiers as follows[3]: $\widetilde{\exists}_\diamond x :\equiv \neg\forall_\diamond x\neg$ for $\diamond \in \{\sqcup, \emptyset, +, -, \pm\}$.

⋆ The second author gratefully acknowledges financial support by MATHLOGAPS (MEST-CT-2004-504029), a Marie Curie Early Stage Training Site.

[3] Throughout the whole paper we will use "$\sqcup$" as a placeholder for the empty space.

In [7], $\widetilde{\exists}_\emptyset$ was denoted $\overline{\exists^{\mathtt{cl}}}$ and $\widetilde{\exists}, \widetilde{\exists}_\pm$ were both denoted $\exists^{\mathtt{cl}}$. As we will see later, $\widetilde{\exists}_\emptyset$ does have a certain computational contribution when placed in front of a computationally meaningful formula, manifested by the increase in type level of its light Dialectica translation. The effect is identical to the one of a double negation. Although generally not void of computational content, $\widetilde{\exists}_\emptyset$ is nevertheless fully uniform, in the sense that its quantified variable has no contribution to its computational content. Similarly, for $\widetilde{\exists}_+$ and $\widetilde{\exists}_-$ the quantified variable has only a partial impact on their computational content, reason why we can call them *semi-uniform* and thus "light".

2 Arithmetical systems for light Dialectica extraction

The verifying arithmetical system NA, into which input proofs will be translated by the light Dialectica algorithm, is the standard Natural Deduction (abbreviated "ND") formulation of the negative fragment of Heyting Arithmetic in all finite types HA^ω from [18]. The input arithmetical system NA_l is basically an annotated refinement of NA with the light quantifiers, where $\forall_\pm$ replaces $\forall$. System NA_l will also include a number of peculiar "light" principles, some of them just annotated variants of certain NA theorems (which may not be NA_l theorems), which are straightforwardly LD-realizable in NA.

2.1 The system NA

Below we define finite types T, terms $\mathcal{T}$, formulas $\mathcal{F}$ and light formulas $\mathcal{F}_l$:

$$
\begin{array}{llll}
T & \rho,\sigma & ::= & \iota \mid o \mid (\rho\sigma) \\
\mathcal{T} & s,t & ::= & x^\rho \mid \mathtt{T}^o \mid \mathtt{F}^o \mid \mathtt{0}^\iota \mid \mathtt{S}^{\iota\iota} \mid \mathtt{If}^{o\rho\rho\rho} \mid \mathtt{R}^{\iota\rho(\iota\rho\rho)\rho} \mid (\lambda x^\rho.\, t^\sigma)^{\rho\sigma} \mid (t^{\rho\sigma} s^\rho)^\sigma \\
\mathcal{F} & A,B & ::= & \mathtt{at}(t^o) \mid A \to B \mid A \wedge B \mid \forall x^\rho A \\
\mathcal{F}_l & A,B & ::= & \mathtt{at}(t^o) \mid A \to B \mid A \wedge B \mid \forall_\diamond x^\rho A \quad \text{for } \diamond \in \{\emptyset,+,-,\pm\}
\end{array}
$$

For simplicity we employ just two basic types: integers ι and booleans o, and use $\rho\sigma\tau$ for $(\rho(\sigma\tau))$. Apart from the usual constructors for booleans ($\mathtt{T},\mathtt{F}$) and integers ($\mathtt{0},\mathtt{S}$), our terms include case distinction $\mathtt{If}$ and Gödel recursion $\mathtt{R}$.

The operator $\mathsf{FV}(\cdot)$ returning the set of free variables of its argument $t \in \mathcal{T}$ or $A \in \mathcal{F}/\mathcal{F}_l$ is defined as usual. Atomic formulas are identified with boolean terms and thus are decidable by definition. In particular, we will use decidable falsity $\bot :\equiv \mathtt{at}(\mathtt{F})$ and truth $\top :\equiv \mathtt{at}(\mathtt{T})$. As usual, we abbreviate $A \to \bot$ by $\neg A$. The language of NA (with $\forall$) is denoted $\mathcal{L}$ and the language of NA_l (with $\forall_\emptyset, \forall_+, \forall_-, \forall_\pm$) is denoted $\mathcal{L}_l$.

We use a special ND presentation of our systems, where proofs are represented as sequents $\Gamma \vdash B$, meaning that formula B is the root of the ND tree whose leaves Γ are typed assumption variables (abbreviated "avars") $a:A$. Here the formula A is the type of the avar a. Since there may be more leaves labelled with the same $a:A$, Γ is a multiset. The logical rules of system NA are presented in Table 1, with the usual restriction on $\forall^i$ that $z \notin \mathsf{FV}(\Gamma) \equiv \bigcup_{a:A\in\Gamma} \mathsf{FV}(A)$. At $\to^i$, $[a:A]$ denotes the multisubset of all occurrences of $a:A$ in the multiset of assumptions of the premise sequent of $\to^i$. Thus $a:A \notin \Gamma$, hence $a:A$ is no longer an assumption in the conclusion sequent of $\to^i$. In

the ND tree, this means that all the leaves labelled $a:A$ are inactivated (or "discharged" as one usually says in Natural Deduction terminology).

$$a:A \vdash A \quad (\mathrm{id}) \qquad \frac{\Gamma, [a:A] \vdash B}{\Gamma \vdash A \to B}\ {\to}^i \qquad \frac{\Gamma \vdash A \quad \Delta \vdash A \to B}{\Gamma, \Delta \vdash B}\ {\to}^e \qquad \frac{\Gamma \vdash A}{\Gamma \vdash \forall z A}\ \forall^i$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge_l^e \qquad \frac{\Delta \vdash A \wedge B}{\Delta \vdash B}\ \wedge_r^e \qquad \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \wedge B}\ \wedge^i \qquad \frac{\Gamma \vdash \forall z A}{\Gamma \vdash A[t/z]}\ \forall^e$$

Table 1. Logical rules

We find it convenient to introduce induction for booleans and naturals as the rules presented in Table 2. Here we assume that the induction variables b^o and respectively n^ι do not occur freely in Γ, nor Δ, and that they do occur in the formula A.

$$\frac{\Gamma \vdash A(\mathtt{T}) \quad \Delta \vdash A(\mathtt{F})}{\Gamma, \Delta \vdash A(b)}\ \mathtt{Ind}_o \qquad \frac{\Gamma \vdash A(\mathtt{0}) \quad \Delta \vdash A(n) \to A(\mathtt{S}n)}{\Gamma, \Delta \vdash A(n)}\ \mathtt{Ind}_\iota$$

Table 2. Induction rules

Computation in NA is expressed via the usual β-reduction rule $(\lambda x.t)s \hookrightarrow t[x \mapsto s]$, plus rewrite rules defining the computational meaning of `If` and `R` :

$$\begin{array}{ll} \mathtt{If}\,\mathtt{T}\,s\,t \hookrightarrow s & \qquad \mathtt{R}\,\mathtt{0}\,s\,t \hookrightarrow s \\ \mathtt{If}\,\mathtt{F}\,s\,t \hookrightarrow t & \qquad \mathtt{R}\,(\mathtt{S}n)\,s\,t \hookrightarrow t\,n\,(\mathtt{R}\,n\,s\,t) \end{array}$$

Since this typed term system is confluent and strongly normalizing (cf. [15]), we are free not to fix a particular evaluation strategy. For simplicity, we assume that all terms occurring in proofs are automatically in normal form[4]. When building proofs, some computation is thus carried out implicitly, behind the scene.

$$\mathtt{TAx}: \ \vdash \mathtt{at}(\mathtt{T}) \qquad \mathtt{CmpAx}: \ \vdash x =_\rho y \to A(x) \to A(y)$$

Table 3. Basic axioms

Using recursion at higher types we can define any provably total function of ground arithmetic, including such decidable predicates. For instance, decidable equality $\mathtt{Eq}_o$

[4] Normalization is necessary only when matching terms in formulas. We only avoid introducing equality axioms `AxEQL` as in [7] and skip the corresponding easy applications of Compatibility.

for booleans and $\mathtt{Eq}_\iota$ for natural numbers is defined as follows:

$$\begin{aligned}\mathtt{Eq}_o^{ooo} &:\equiv \lambda x.\mathtt{If}\, x\, (\lambda y.y)\, (\lambda y.\mathtt{If}\, y\, \mathtt{F}\, \mathtt{T})\\ \mathtt{Eq}_\iota^{\iota\iota o} &:\equiv \lambda x.\mathtt{R}\, x\, \big(\lambda y.\mathtt{R}\, y\, \mathtt{T}\, (\lambda n, q^o.\mathtt{F})\big)\, \big(\lambda m, p^{\iota o}, y.\mathtt{R}\, y\, \mathtt{F}\, (\lambda n, q^o.p\, n)\big)\end{aligned}$$

The $\mathtt{at}(\cdot)$ construction allows us to view boolean programs as decidable predicates. Given $\mathtt{Ind}_o$, its logical meaning is settled by the *truth axiom* $\mathtt{TAx}$, see Table 3. In this way we can define *predicate equality* at base types as $s =_\sigma t :\equiv \mathtt{at}(\mathtt{Eq}\, s\, t)$ for $\sigma \in \{o, \iota\}$ and further at higher types extensionally as usual $s =_{\rho\tau} t :\equiv \forall x^\rho (sx =_\tau tx)$. It is straightforward to prove by induction on ρ that $=_\rho$ is reflexive, symmetric and transitive at any type ρ.

To complete our system, we must include in NA also the *compatibility* axiom $\mathtt{CmpAx}$, see Table 3. Note that *ex falso quodlibet* ($\mathtt{EFQ}$) $\bot \to A$ and *stability* ($\mathtt{Stab}$) $\neg\neg A \to A$ are fully provable in NA (cf. [15], by induction on A, using $\mathtt{TAx}$ and $\mathtt{Ind}_o$).

2.2 The system NA_l

System NA_l refines the clone of NA (just $\forall$ renamed to $\forall_\pm$) with introduction and elimination rules for the light quantifiers (see Table 4). These are copies of the clone rules $\forall^e_\pm$ and $\forall^i_\pm$, but with the usual restriction ($\pm$) on $\forall^i_\pm$ that $z \notin \mathsf{FV}(\Gamma)$ refined with the following conditions referring to the LD-interpretation (later defined in Sect. 3):

($+$) at $\forall^i_+$, z may be used computationally only positively, i.e., z must not be free in the *challengers* of the LD-translation of Γ.

($-$) at $\forall^i_-$, z may be used computationally only negatively, i.e., z must not be free in the *witnesses* of the LD-translation of A.

($\emptyset$) at $\forall^i_\emptyset$, z may not be used computationally at all, i.e., both ($+$) and ($-$).

$$\frac{\Gamma \vdash_l A}{\Gamma \vdash_l \forall_\pm z A}\ \forall^i_\pm \qquad \frac{\Gamma \vdash_l \forall_\pm z A}{\Gamma \vdash_l A[t/z]}\ \forall^e_\pm$$

$$\frac{\Gamma \vdash_l A}{\Gamma \vdash_l \forall_+ z A}\ \forall^i_+ \qquad \frac{\Gamma \vdash_l \forall_+ z A}{\Gamma \vdash_l A[t/z]}\ \forall^e_+$$

$$\frac{\Gamma \vdash_l A}{\Gamma \vdash_l \forall_- z A}\ \forall^i_- \qquad \frac{\Gamma \vdash_l \forall_- z A}{\Gamma \vdash_l A[t/z]}\ \forall^e_-$$

$$\frac{\Gamma \vdash_l A}{\Gamma \vdash_l \forall_\emptyset z A}\ \forall^i_\emptyset \qquad \frac{\Gamma \vdash_l \forall_\emptyset z A}{\Gamma \vdash_l A[t/z]}\ \forall^e_\emptyset$$

Table 4. Additional rules for NA_l

Notice that the restrictions ($+$), ($-$) and ($\emptyset$) assume knowledge of the LD-interpretation of whole proofs, in their full depth, thus forcing the definition of NA_l proofs to go inductively in parallel with the LD-extraction of part of their computational content (namely free variables of the extracted terms).

Definition 1 (computational relevance of formulas). We simultaneously define the classes of *realization irrelevant* $A_\oplus$ and *refutation irrelevant* $A_\ominus$ formulas as follows:

$$\begin{aligned} A_\oplus, B_\oplus &::= \mathtt{at}(t) \mid A_\oplus \wedge B_\oplus \mid A_\ominus \to B_\oplus \mid \forall_\diamond x\, A_\oplus \quad \text{for } \diamond \in \{\emptyset, +, -, \pm\}\\ A_\ominus, B_\ominus &::= \mathtt{at}(t) \mid A_\ominus \wedge B_\ominus \mid A_\oplus \to B_\ominus \mid \forall_\diamond x\, A_\ominus \quad \text{for } \diamond \in \{\emptyset, +\}\end{aligned}$$

A formula is *realization (refutation) relevant* if it is not realization (refutation) irrelevant. An assumption formula in a sequent is *computationally relevant* if it is refutation relevant and the conclusion formula in a sequent is *computationally relevant* if it is realization relevant. An explanation for this terminology is given by Remark 1 in Sec. 3.

One necessary change when adopting principles from NA is to replace $\mathtt{CmpAx}$ with a weak compatibility rule. This is because Dialectica is unable to interpret full extensionality (cf. [18]). We here employ an upgraded variant of the $\mathtt{CMP}$ rule[5] from [7]:

$$\frac{\Gamma_{\ominus} \vdash_l x =_\rho y}{\Gamma_{\ominus} \vdash_l B(x) \to B(y)}\ \mathtt{CMP}_\rho$$

where all formulas in $\Gamma_{\ominus}$ are refutation irrelevant.

Whereas in NA alone we could have safely let all contractions be handled implicitly at $\to^i$, for expository purposes it is convenient to explicitate the computationally relevant contractions of NA_l. We achieve this by including in NA_l the *contraction anti-rule*[6] $\mathcal{C}_l$ (see Table 5) for all formulas A that are refutation relevant and $\bigstar$: do not contain any $\forall_+$, nor $\forall_\emptyset$. This triggers the addition to NA of an explicit (unrestricted) contraction anti-rule C. The restriction $\bigstar$ ensures that all contraction formulas that require at least

$$\frac{\Delta, a:A, a:A \vdash B}{\Delta, a:A \vdash B}\ \mathsf{C} \qquad \frac{\Delta, a:A, a:A \vdash_l B}{\Delta, a:A \vdash_l B}\ \mathcal{C}_l$$

Table 5. Contraction anti-rules C for NA and (restricted) $\mathcal{C}_l$ for NA_l

one challenger term for their LD-interpretation will have quantifier-free (hence decidable) LD-translations, which is necessary for attaining soundness. Note that, being a purely syntactical criterion, $\bigstar$ does not admit formulas whose LD-translations contain quantifiers, but could nevertheless be decidable in certain models. Moreover, in order to avoid having any computationally relevant contractions implicit in $\to^i$, we constrain the deduction rules of NA_l to disallow multiple occurrences of refutation relevant assumptions in any of the premise sequents. Thus, whenever a double occurrence of a refutation relevant assumption is created in a conclusion sequent by one of the binary rules of NA_l, such sequent cannot be directly a premise for the application of an(other) NA_l rule: the anti-rule $\mathcal{C}_l$ must be applied first, in order to eliminate the critical double. If $\bigstar$ is not satisfied and yet $a:A$ is a refutation relevant assumption occurring at least twice in some conclusion sequent, this is a dead end: such sequent can only be the root of the NA_l proof-tree.

Since the $\mathtt{Ind}_\iota$ rule corresponds to a virtually unbounded number of contractions of each assumption in Δ (cf. [7]), its clone in the system NA_l is subject to a restriction like the one of $\mathcal{C}_l$. Namely, we need to require that all refutation relevant avars in Δ satisfy

[5] The weak extensionality (compatibility) rule is originally due to Spector [16], cf. [9].

[6] We refer to contraction as "anti-rule", rather than "rule" because, despite the sequent-like representation of our calculi, in fact our formalisms are Natural Deduction (ND) and in the ND *directed* tree the representation of contraction is by convergent arrows that go in the direction which is reverse to the "normal" direction of all the other rules.

$\bigstar$. Moreover, since the contractions for $\mathtt{Ind}_\iota$ will be handled differently than for simple binary rules like $\rightarrow^e$ or $\wedge^i$, it is more convenient to require that $\mathtt{Ind}_\iota$ in NA_l implicitly contracts all its refutation relevant assumptions. We will use the notation $\Gamma \uplus \Delta$ for a special multiset union in which refutation relevant assumptions appear only once, even if they appear in both Γ and Δ. Thus $\mathtt{Ind}_\iota$ for NA_l is obtained by replacing "Γ, Δ" with "$\Gamma \uplus \Delta$ in Table 2."

Notation for tuples. We use bold face variables $\boldsymbol{f}, \boldsymbol{g}, \ldots, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}, \ldots$ for tuples of variables, and bold face terms $\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{t}, \ldots, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\zeta} \ldots$ for tuples of terms. Given the sequences of terms $\boldsymbol{t}$ and $\boldsymbol{s}$, by $\boldsymbol{ts}$ we mean the sequence of terms $t_0\boldsymbol{s}, \ldots, t_n\boldsymbol{s}$. Similarly for the multiple simultaneous substitution $\boldsymbol{t}[\boldsymbol{s}/\boldsymbol{x}]$.

Canonical zero terms. For each higher-order type ρ we define a corresponding zero term $\mathtt{0}^\rho :\equiv \lambda \boldsymbol{x}.\, \mathtt{0}^\sigma$ where $\sigma \in \{o, \iota\}$ is the corresponding ground type and $\mathtt{0}^o :\equiv \mathtt{F}$.

3 The light Dialectica interpretation

With each formula A of NA_l we associate its LD-translation: a not necessarily quantifier-free formula $|A|^{\boldsymbol{x}}_{\boldsymbol{y}}$ of NA where $\boldsymbol{x}, \boldsymbol{y}$ are tuples of fresh variables, not appearing in A. The variables $\boldsymbol{x}$ in the superscript are called the *witness variables*, while the subscript variables $\boldsymbol{y}$ are called the *challenge variables*. Terms $\boldsymbol{t}$ substituting witness variables (like $|A|^{\boldsymbol{t}}_{\boldsymbol{y}}$) are called *realizing terms* or "witnesses" and terms $\boldsymbol{s}$ substituting challenge variables (like $|A|^{\boldsymbol{x}}_{\boldsymbol{s}}$) are called *refuting terms* or "challengers". The new and more compact notation $|A|^{\boldsymbol{x}}_{\boldsymbol{y}}$ is originally due to Oliva [11].

Intuitively, the LD-interpretation of A can be viewed as a game in which first Eloise ($\exists$) and then Abelard ($\forall$) make one move each by playing type-corresponding objects $\boldsymbol{t}$ and $\boldsymbol{s}$ for the tuples $\boldsymbol{x}$ and respectively $\boldsymbol{y}$. Formula $|A|^{\boldsymbol{x}}_{\boldsymbol{y}}$ specifies (cf. [10]) the "adjudication relation", here not necessarily decidable: Eloise wins iff $\mathsf{NA} \vdash |A|^{\boldsymbol{t}}_{\boldsymbol{s}}$. In our light context as well, Eloise has a *winning move* whenever A is provable in NA_l : the LD-interpretation will explicitly provide it from the input NA_l proof of A as a tuple of witnesses $\boldsymbol{t}$ (s.t. $\mathsf{FV}(\boldsymbol{t}) \subseteq \mathsf{FV}(A)$) together with the *verifying proof* in NA of $\forall \boldsymbol{y} |A|^{\boldsymbol{t}}_{\boldsymbol{y}}$ (Eloise wins by $\boldsymbol{t}$ regardless of the instances $\boldsymbol{s}$ for Abelard's $\boldsymbol{y}$).

Definition 2 (LD-translation of formulas). *The interpretation does not change atomic formulas, i.e.,* $|\mathtt{at}(t^o)| :\equiv \mathtt{at}(t^o)$. *Assuming* $|A|^{\boldsymbol{x}}_{\boldsymbol{y}}$ *and* $|B|^{\boldsymbol{u}}_{\boldsymbol{v}}$ *are already defined,*

$$\begin{aligned} |A \wedge B|^{\boldsymbol{x},\boldsymbol{u}}_{\boldsymbol{y},\boldsymbol{v}} &:\equiv |A|^{\boldsymbol{x}}_{\boldsymbol{y}} \wedge |B|^{\boldsymbol{u}}_{\boldsymbol{v}} \\ |A \to B|^{\boldsymbol{f},\boldsymbol{g}}_{\boldsymbol{x},\boldsymbol{v}} &:\equiv |A|^{\boldsymbol{x}}_{\boldsymbol{f}\boldsymbol{x}\boldsymbol{v}} \to |B|^{\boldsymbol{g}\boldsymbol{x}}_{\boldsymbol{v}} \quad . \end{aligned}$$

The interpretation of the four universal quantifiers is (upon renaming, we assume that quantified variables occur uniquely in a formula):

$$\begin{aligned} |\forall_\pm z\, A(z)|^{\boldsymbol{f}}_{z,\boldsymbol{y}} &:\equiv |A(z)|^{\boldsymbol{f}z}_{\boldsymbol{y}} & |\forall_+ z\, A(z)|^{\boldsymbol{f}}_{\boldsymbol{y}} &:\equiv \forall z |A(z)|^{\boldsymbol{f}z}_{\boldsymbol{y}} \\ |\forall_- z\, A(z)|^{\boldsymbol{x}}_{z,\boldsymbol{y}} &:\equiv |A(z)|^{\boldsymbol{x}}_{\boldsymbol{y}} & |\forall_\emptyset z\, A(z)|^{\boldsymbol{x}}_{\boldsymbol{y}} &:\equiv \forall z |A(z)|^{\boldsymbol{x}}_{\boldsymbol{y}} \end{aligned}$$

Since $|\bot| \equiv \bot$, *we get*

$$|\neg A|^{\boldsymbol{f}}_{\boldsymbol{x}} \equiv \neg |A|^{\boldsymbol{x}}_{\boldsymbol{f}\boldsymbol{x}} \qquad |\neg\neg A|^{\boldsymbol{f}}_{\boldsymbol{g}} \equiv \neg\neg |A|^{\boldsymbol{f}\boldsymbol{g}}_{\boldsymbol{g}(\boldsymbol{f}\boldsymbol{g})}$$

It is straightforward to compute that

$$|\widetilde{\exists}_{\pm} z\, A(z)|^{Z,\boldsymbol{f}}_{\boldsymbol{g}} :\equiv \neg\neg |A(Z\boldsymbol{g})|^{\boldsymbol{fg}}_{\boldsymbol{g}(Z\boldsymbol{g})(\boldsymbol{fg})} \qquad |\widetilde{\exists}_{+} z\, A(z)|^{\boldsymbol{f}}_{\boldsymbol{g}} :\equiv \widetilde{\exists} z\, |A(z)|^{\boldsymbol{fg}}_{\boldsymbol{gz}(\boldsymbol{fg})}$$
$$|\widetilde{\exists}_{-} z\, A(z)|^{Z,\boldsymbol{f}}_{\boldsymbol{g}} :\equiv \neg\neg |A(Z\boldsymbol{g})|^{\boldsymbol{fg}}_{\boldsymbol{g}(\boldsymbol{fg})} \qquad |\widetilde{\exists}_{\emptyset} z\, A(z)|^{\boldsymbol{f}}_{\boldsymbol{g}} :\equiv \widetilde{\exists} z\, |A(z)|^{\boldsymbol{fg}}_{\boldsymbol{g}(\boldsymbol{fg})}$$

The length and types of the witnessing and challenging tuples are uniquely determined.

Remark 1 It is easy to see that a formula is realization relevant exactly when its tuple of witness variables is not empty and, similarly, a formula is refutation relevant exactly when its tuple of challenge variables is not empty.

We prove the soundness of our interpretation, i.e., we show how Eloise's winning move in the game $|A|^{\boldsymbol{x}}_{\boldsymbol{y}}$ can be algorithmically extracted from a proof of A in NA_l.

Theorem 1 (Soundness of light Dialectica interpretation).
Let $A_0, A_1, \ldots, A_n$ be a sequence of formulas in $\mathcal{F}_l$ with $\boldsymbol{w}$ all their free variables. If the sequent $a_1 : A_1, \ldots, a_n : A_n \vdash_l A_0$ is provable in NA_l, then terms $\boldsymbol{t}_0, \ldots, \boldsymbol{t}_n$ can be automatically synthesised from its formal proof, such that the translated sequent $a_1 : |A_1|^{\boldsymbol{x}_1}_{\boldsymbol{t}_1}, \ldots, a_n : |A_n|^{\boldsymbol{x}_n}_{\boldsymbol{t}_n} \vdash |A_0|^{\boldsymbol{t}_0}_{\boldsymbol{x}_0}$ is provable in NA, where the following free variable condition (c) *holds: $\mathsf{FV}(\boldsymbol{t}_i) \subseteq \{\boldsymbol{w}, \boldsymbol{x}_0, \ldots, \boldsymbol{x}_n\}$ and $\boldsymbol{x}_0 \notin \mathsf{FV}(\boldsymbol{t}_0)$. Here $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_n$ are tuples of fresh variables, s.t. equal avars share a common such tuple.*

Proof: The extraction meta-algorithm proceeds recursively on the structure of the input proof. It thus suffices to present witnesses for each realization relevant axiom and for each rule to produce terms for the conclusion sequent out of terms assumed for the premise sequent(s). Due to its importance, we first present the treatment of contraction.

$$\frac{\Delta, a : A, a : A \vdash_l B}{\Delta, a : A \vdash_l B}\ \mathcal{C}_l$$

We are given $|\Delta|^{\boldsymbol{u}}_{\boldsymbol{\delta}}$, $a : |A|^{\boldsymbol{x}}_{\boldsymbol{t'}}$, $a : |A|^{\boldsymbol{x}}_{\boldsymbol{t''}} \vdash |B|^{\boldsymbol{s}}_{\boldsymbol{v}}$, where the LD-variables $\boldsymbol{x}$ may occur freely in all terms $\boldsymbol{\delta}, \boldsymbol{t'}, \boldsymbol{t''}, \boldsymbol{s}$. We need to *equalize* the possibly distinct $\boldsymbol{t'}$ and $\boldsymbol{t''}$. For this we use the decidability of $|A|$ which is ensured by $\bigstar$. We can thus define for each pair of corresponding $t' \in \boldsymbol{t'}$ and $t'' \in \boldsymbol{t''}$ the term $t :\equiv \mathtt{If}(|A|^{\boldsymbol{x}}_{\boldsymbol{t'}})\, t''\, t'$. We then have in NA that $|A|^{\boldsymbol{x}}_{\boldsymbol{t'}} \to \boldsymbol{t} = \boldsymbol{t''}$ and $\neg|A|^{\boldsymbol{x}}_{\boldsymbol{t'}} \to \boldsymbol{t} = \boldsymbol{t'}$. Thus by $\mathtt{CmpAx}$, $|A|^{\boldsymbol{x}}_{\boldsymbol{t'}} \to (|A|^{\boldsymbol{x}}_{\boldsymbol{t}} \to |A|^{\boldsymbol{x}}_{\boldsymbol{t''}})$ and $\neg|A|^{\boldsymbol{x}}_{\boldsymbol{t'}} \to (|A|^{\boldsymbol{x}}_{\boldsymbol{t}} \to |A|^{\boldsymbol{x}}_{\boldsymbol{t'}})$, hence by prop. logic $|A|^{\boldsymbol{x}}_{\boldsymbol{t'}} \to (|A|^{\boldsymbol{x}}_{\boldsymbol{t}} \to |A|^{\boldsymbol{x}}_{\boldsymbol{t'}} \wedge |A|^{\boldsymbol{x}}_{\boldsymbol{t''}})$ and $\neg|A|^{\boldsymbol{x}}_{\boldsymbol{t'}} \to (|A|^{\boldsymbol{x}}_{\boldsymbol{t}} \to |A|^{\boldsymbol{x}}_{\boldsymbol{t}} \wedge |A|^{\boldsymbol{x}}_{\boldsymbol{t''}})$, where for the latter we used $\mathtt{EFQ}$. By *case distinction* we get $\vdash |A|^{\boldsymbol{x}}_{\boldsymbol{t}} \to |A|^{\boldsymbol{x}}_{\boldsymbol{t'}} \wedge |A|^{\boldsymbol{x}}_{\boldsymbol{t''}}$, hence both $|A|^{\boldsymbol{x}}_{\boldsymbol{t}} \vdash |A|^{\boldsymbol{x}}_{\boldsymbol{t'}}$ and $|A|^{\boldsymbol{x}}_{\boldsymbol{t}} \vdash |A|^{\boldsymbol{x}}_{\boldsymbol{t''}}$. From these we get[7] $|\Delta|^{\boldsymbol{u}}_{\boldsymbol{\delta}}$, $a : |A|^{\boldsymbol{x}}_{\boldsymbol{t}}$, $a : |A|^{\boldsymbol{x}}_{\boldsymbol{t}} \vdash |B|^{\boldsymbol{s}}_{\boldsymbol{v}}$, to which a C is finally applied.

$$\frac{\Gamma \vdash_l A \quad \Delta \vdash_l A \to B}{\Gamma, \Delta \vdash_l B}\ \to^e$$

We are given $|\Delta|^{\boldsymbol{z}}_{\boldsymbol{\delta}[\boldsymbol{x}]} \vdash |A|^{\boldsymbol{x}}_{\boldsymbol{txv}} \to |B|^{\boldsymbol{sx}}_{\boldsymbol{v}}$ and $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}[\boldsymbol{y}]} \vdash |A|^{\boldsymbol{r}}_{\boldsymbol{y}}$ in which we simultaneously substitute $\boldsymbol{x} \mapsto \boldsymbol{r}$ and $\boldsymbol{y} \mapsto \boldsymbol{trv}$ and by a $\to^e$ we get $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}[\boldsymbol{trv}]}, |\Delta|^{\boldsymbol{z}}_{\boldsymbol{\delta}[\boldsymbol{r}]} \vdash |B|^{\boldsymbol{sr}}_{\boldsymbol{v}}$. We used that $\boldsymbol{x} \notin \mathsf{FV}(\boldsymbol{t}, \boldsymbol{s})$ and $\boldsymbol{y} \notin \mathsf{FV}(\boldsymbol{r})$, which follow from (c). Since $\mathsf{FV}(\boldsymbol{r}) \subseteq \mathsf{FV}(\Gamma) \cup \mathsf{FV}(A) \cup \{\boldsymbol{u}\}$ (also a consequence of (c)), we have that $\boldsymbol{v} \notin \mathsf{FV}(\boldsymbol{r})$, hence (c) is preserved.

[7] Applying twice $\to^i$ followed by $\to^e$.

$\dfrac{\Gamma, [a\!:\!A] \vdash_l B}{\Gamma \vdash_l A \to B}\ {\to}^i$ If $[a : A]$ is a multiset then A is refutation irrelevant and thus $|A \to B|^{\sqcup,\boldsymbol{s}}_{\boldsymbol{x},\boldsymbol{v}} \equiv |A|^{\boldsymbol{x}}_{\sqcup} \to |B|^{\boldsymbol{sx}}_{\boldsymbol{v}}$. All we need is to λ-abstract the realizers for $|B|$ over the LD-variables $\boldsymbol{x}$ from $|A|$ and apply an ${\to}^i$. Thus $\boldsymbol{x} \notin \mathsf{FV}(\boldsymbol{s})$, hence (c) is preserved. If $[a : A]$ is just a set, then we are given that $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}}, a\!:\!|A|^{\boldsymbol{x}}_{\boldsymbol{t'}} \vdash |B|^{\boldsymbol{s'}}_{\boldsymbol{v}}$. Let $\boldsymbol{t}{:}{\equiv}\lambda\boldsymbol{x},\boldsymbol{v}.\,\boldsymbol{t'}$ and $\boldsymbol{s}{:}{\equiv}\lambda\boldsymbol{x}.\,\boldsymbol{s'}$. Then by an ${\to}^i$ we get exactly $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}} \vdash |A \to B|^{\boldsymbol{t},\boldsymbol{s}}_{\boldsymbol{x},\boldsymbol{v}}$, with (c) preserved since we knew that $\boldsymbol{v} \notin \mathsf{FV}(\boldsymbol{s'})$.

$\dfrac{\Gamma \vdash_l A(\mathtt{T}) \quad \Delta \vdash_l A(\mathtt{F})}{\Gamma, \Delta \vdash_l A(b)}\ \mathtt{Ind}_o$ We are given $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}[\boldsymbol{x}]} \vdash |A(\mathtt{T})|^{\boldsymbol{r}}_{\boldsymbol{x}}$ and $|\Delta|^{\boldsymbol{z}}_{\boldsymbol{\delta}[\boldsymbol{y}]} \vdash |A(\mathtt{F})|^{\boldsymbol{s}}_{\boldsymbol{y}}$. For each pair of corresponding $r \in \boldsymbol{r}$ and $s \in \boldsymbol{s}$ we define $t{:}{\equiv}\mathtt{If}\, b\, r\, s$ and we also substitute $\boldsymbol{y} \mapsto \boldsymbol{x}$. Then by $\mathtt{Ind}_o$ we get $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}[\boldsymbol{x}]}, |\Delta|^{\boldsymbol{z}}_{\boldsymbol{\delta}[\boldsymbol{x}]} \vdash |A(b)|^{\boldsymbol{t}}_{\boldsymbol{x}}$. Adding the variable b to $\boldsymbol{t}$ does not violate (c), because, as we formally requested, b certainly occurs in $A(b)$.

$\dfrac{\Gamma \vdash_l A(\mathtt{0}) \quad \Delta \vdash_l A(n) \to A(\mathtt{S}n)}{\Gamma \uplus \Delta \vdash_l A(n)}\ \mathtt{Ind}_\iota$ We are given $(\circ\circ)$: $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}[\boldsymbol{y}]} \vdash |A(\mathtt{0})|^{\boldsymbol{r}}_{\boldsymbol{y}}$ and $(\circ)$: $|\Delta|^{\boldsymbol{z}}_{\boldsymbol{\delta}[\boldsymbol{x};\boldsymbol{v}]} \vdash |A(n)|^{\boldsymbol{x}}_{\boldsymbol{t}\boldsymbol{x}\boldsymbol{v}} \to |A(\mathtt{S}n)|^{\boldsymbol{sx}}_{\boldsymbol{v}}$. We show $(*)$: $\forall \boldsymbol{v}\big(|\Gamma \uplus \Delta|^{\boldsymbol{u}\uplus\boldsymbol{z}}_{\boldsymbol{\zeta}[n]\boldsymbol{v}} \to |A(n)|^{\boldsymbol{t'}[n]}_{\boldsymbol{v}}\big)$, where[8] $t'[n] {:}{\equiv} \mathtt{R}\, n\, r\, (\lambda n.s)$ for every corresponding $\langle r \in \boldsymbol{r} / s \in \boldsymbol{s}\rangle$ and $\boldsymbol{\zeta}[n]$ will be constructed as functional terms depending on $\boldsymbol{v}$. Let $b\!:\!B$ be a refutation relevant avar in $\Gamma \uplus \Delta$. Let $\boldsymbol{\gamma'} \in \boldsymbol{\gamma}$ and/or $\boldsymbol{\delta'} \in \boldsymbol{\delta}$ be the challengers for b in Γ and/or Δ. If b appears *only* in Γ, we define $\boldsymbol{\zeta'}[n]{:}{\equiv}\mathtt{R}\, n\, (\lambda \boldsymbol{v}.\boldsymbol{\gamma'}[\boldsymbol{v}])\, \big(\lambda n, p, \boldsymbol{v}.p(\boldsymbol{t}\,\boldsymbol{t'}\boldsymbol{v})\big)$. If b appears in Δ, then the decidability of $|B|$ is needed at each recursive step to equalize the terms $p(\boldsymbol{t}\,\boldsymbol{t'}\boldsymbol{v})$ obtained by the recursive call with the corresponding terms $\boldsymbol{\delta'}$. Thus we provide the right stop point of the backwards recursion. In fact an implicit contraction over b happens at each inductive step and ★ guarantees that $|B|$ is decidable. We define $(\circ_1)$: $\boldsymbol{\zeta''}[n]{:}{\equiv}\mathtt{R}\, n\, (\lambda \boldsymbol{v}.\boldsymbol{\gamma'}[\boldsymbol{v}])\, \Big(\lambda n, p, \boldsymbol{v}.\mathtt{If}(|B|^{\boldsymbol{z'}}_{\boldsymbol{\delta'}[\boldsymbol{t'};\boldsymbol{v}]})\, \big(p(\boldsymbol{t}\,\boldsymbol{t'}\boldsymbol{v})\big)\, \boldsymbol{\delta'}[\boldsymbol{t'};\boldsymbol{v}]\Big)$ for $b \in \Gamma \cap \Delta$. If b appears *only* in Δ, then we define its $\boldsymbol{\zeta''}[n]$ by replacing in $(\circ_1)$ the $\boldsymbol{\gamma'}$ with canonical zeros. Let $\boldsymbol{\zeta}$ denote the tuple of all such $\boldsymbol{\zeta'}$ and $\boldsymbol{\zeta''}$. Notice that $(\circ_2)$: $t'[\mathtt{S}n] = st'[n]$ and $(\circ_3)$: $\boldsymbol{\zeta'}[\mathtt{S}n]\boldsymbol{v} = \boldsymbol{\zeta'}[n](\boldsymbol{t}\,\boldsymbol{t'}\boldsymbol{v})$. We attempt to extend the latter to the whole $\boldsymbol{\zeta}$, by proving $(\circ_4)$: $|B|^{\boldsymbol{z'}}_{\boldsymbol{\zeta''}[\mathtt{S}n]\boldsymbol{v}} \vdash \boldsymbol{\zeta''}[\mathtt{S}n]\boldsymbol{v} = \boldsymbol{\zeta''}[n](\boldsymbol{t}\,\boldsymbol{t'}\boldsymbol{v})$. With $(\circ_1)$, we obtain this as an immediate consequence of $(\circ_5)$: $|B|^{\boldsymbol{z'}}_{\boldsymbol{\zeta''}[\mathtt{S}n]\boldsymbol{v}} \vdash |B|^{\boldsymbol{z'}}_{\boldsymbol{\delta'}[\boldsymbol{t'};\boldsymbol{v}]}$. Assuming $\neg|B|^{\boldsymbol{z'}}_{\boldsymbol{\delta'}[\boldsymbol{t'};\boldsymbol{v}]}$, by $(\circ_1)$ we get $\boldsymbol{\zeta''}[\mathtt{S}n]\boldsymbol{v} = \boldsymbol{\delta'}[\boldsymbol{t'};\boldsymbol{v}]$, hence $\neg|B|^{\boldsymbol{z'}}_{\boldsymbol{\zeta''}[\mathtt{S}n]\boldsymbol{v}}$, and $(\circ_5)$ follows via $\mathtt{Stab}$.

We now prove $(*)$ by (assumptionless) induction on n. The base $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\zeta'}[\mathtt{0}]\boldsymbol{v}} \vdash |A(\mathtt{0})|^{\boldsymbol{t'}[\mathtt{0}]}_{\boldsymbol{v}}$ follows from $(\circ\circ)$. Given $(*)$, we want to prove $(**)$: $|\Gamma \uplus \Delta|^{\boldsymbol{u}\uplus\boldsymbol{z}}_{\boldsymbol{\zeta}[\mathtt{S}n]\boldsymbol{v}} \vdash |A(\mathtt{S}n)|^{\boldsymbol{t'}[\mathtt{S}n]}_{\boldsymbol{v}}$. To $(*)$ we apply $\forall^e_{[\boldsymbol{v}\mapsto \boldsymbol{t}\,\boldsymbol{t'}\boldsymbol{v}]}$ and get $(\circ_6)$: $|\Gamma \uplus \Delta|^{\boldsymbol{u}\uplus\boldsymbol{z}}_{\boldsymbol{\zeta}[n](\boldsymbol{t}\,\boldsymbol{t'}\boldsymbol{v})} \vdash |A(n)|^{\boldsymbol{t'}[n]}_{\boldsymbol{t}\,\boldsymbol{t'}\boldsymbol{v}}$. From $(\circ_3)$ and $(\circ_4)$ we can write $|\Gamma \uplus \Delta|^{\boldsymbol{u}\uplus\boldsymbol{z}}_{\boldsymbol{\zeta}[\mathtt{S}n]\boldsymbol{v}} \vdash \boldsymbol{\zeta}[\mathtt{S}n]\boldsymbol{v} = \boldsymbol{\zeta}[n](\boldsymbol{t}\,\boldsymbol{t'}\boldsymbol{v})$, which combined with $(\circ_6)$ yields $(\circ_7)$: $|\Gamma \uplus \Delta|^{\boldsymbol{u}\uplus\boldsymbol{z}}_{\boldsymbol{\zeta}[\mathtt{S}n]\boldsymbol{v}} \vdash |A(n)|^{\boldsymbol{t'}[n]}_{\boldsymbol{t}\,\boldsymbol{t'}\boldsymbol{v}}$. In $(\circ)$ we substitute $\boldsymbol{x} \mapsto \boldsymbol{t'}[n]$ and get

[8] We here intentionally use the same variable n that occurs freely in s and t. We often omit to explicitate the appearance of n in t', like $t'[n]$. In fact, just "t'" will implicitly denote $t'[n]$.

$|\Delta|^{\boldsymbol{z}}_{\boldsymbol{\delta}[\boldsymbol{t}';\boldsymbol{v}]} \vdash |A(n)|^{\boldsymbol{t}'[n]}_{\boldsymbol{t}\boldsymbol{t}'\boldsymbol{v}} \to |A(\mathtt{S}n)|^{\boldsymbol{s}\boldsymbol{t}'[n]}_{\boldsymbol{v}}$, which gives $(**)$ by means of $(\circ_2)$, $(\circ_5)$, $(\circ_7)$.

$$\frac{\Gamma_\ominus \vdash_l x =_\rho y}{\Gamma_\ominus \vdash_l B(x) \to B(y)}\ \mathtt{CMP}_\rho$$

We are given $|\Gamma_\ominus| \vdash |x =_\rho y|_{\boldsymbol{z}}$. Since $\boldsymbol{z}$ do not occur freely in $|\Gamma_\ominus|$ we can use $\forall^i$ to obtain $|\Gamma_\ominus| \vdash \forall \boldsymbol{z} |x =_\rho y|_{\boldsymbol{z}}$. But by definition we have $\forall \boldsymbol{z} |x =_\rho y|_{\boldsymbol{z}} \equiv x =_\rho y$. Then by $\mathtt{CmpAx}$ $|\Gamma_\ominus| \vdash |B(x)|^{\boldsymbol{u}}_{\boldsymbol{v}} \to |B(y)|^{\boldsymbol{u}}_{\boldsymbol{v}}$ and clearly a realizing tuple for $B(x) \to B(y)$ is $(\lambda \boldsymbol{u}.\boldsymbol{u}, \lambda \boldsymbol{u}, \boldsymbol{v}.\boldsymbol{v})$, with (c) obviously satisfied.

$$\frac{\Gamma \vdash_l A \wedge B}{\Gamma \vdash_l A}\ \wedge^e_l$$

Keep terms for A in which substitute variables from $\mathsf{FV}(B) \setminus \mathsf{FV}(\Gamma \vdash A)$ with type-corresponding zeros.

$$\frac{\Delta \vdash_l A \wedge B}{\Delta \vdash_l B}\ \wedge^e_r$$

Keep terms for B in which substitute variables from $\mathsf{FV}(A) \setminus \mathsf{FV}(\Delta \vdash B)$ with type-corresponding zeros.

$$\frac{\Gamma \vdash_l A \quad \Delta \vdash_l B}{\Gamma, \Delta \vdash_l A \wedge B}\ \wedge^i$$

Given $|\Gamma| \vdash |A|^{\boldsymbol{t}}_{\boldsymbol{x}}$ and $|\Delta| \vdash |B|^{\boldsymbol{s}}_{\boldsymbol{y}}$, by a $\wedge^i$ one gets $|\Gamma|, |\Delta| \vdash |A \wedge B|^{\boldsymbol{t},\boldsymbol{s}}_{\boldsymbol{x},\boldsymbol{y}}$.

$$a_1 : A_1 \vdash_l A_0\ (\mathrm{id})$$

With $\boldsymbol{t}_1 :\equiv \boldsymbol{x}_0$ and $\boldsymbol{t}_0 :\equiv \boldsymbol{x}_1$ one gets $a_1 : |A_1|^{\boldsymbol{x}_1}_{\boldsymbol{x}_0} \vdash |A_0|^{\boldsymbol{x}_1}_{\boldsymbol{x}_0}$, since $A_0 \equiv A_1$.

$$\frac{\Gamma \vdash_l \forall_\pm z A}{\Gamma \vdash_l A[r/z]}\ \forall^e_\pm$$

We are given $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}[z]} \vdash |A|^{\boldsymbol{t}z}_{\boldsymbol{y}}$, where $z, \boldsymbol{y} \notin \mathsf{FV}(\boldsymbol{t})$ but z may occur in $\boldsymbol{\gamma}$. We substitute $z \mapsto r$ in the proof and get $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}[r]} \vdash |A[r/z]|^{\boldsymbol{t}r}_{\boldsymbol{y}}$.

$$\frac{\Gamma \vdash_l \forall_+ z A}{\Gamma \vdash_l A[r/z]}\ \forall^e_+$$

We are given $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}} \vdash \forall z |A|^{\boldsymbol{t}z}_{\boldsymbol{y}}$, where $z, \boldsymbol{y} \notin \mathsf{FV}(\boldsymbol{t}, \boldsymbol{\gamma})$. By a $\forall^e_{[r]}$ we get $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}} \vdash |A[r/z]|^{\boldsymbol{t}r}_{\boldsymbol{y}}$.

$$\frac{\Gamma \vdash_l \forall_- z A}{\Gamma \vdash_l A[r/z]}\ \forall^e_-$$

We are given $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}[z]} \vdash |A|^{\boldsymbol{t}}_{\boldsymbol{y}}$, where $z, \boldsymbol{y} \notin \mathsf{FV}(\boldsymbol{t})$ but z may occur in $\boldsymbol{\gamma}$. We substitute $z \mapsto r$ in the proof and get $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}[r]} \vdash |A[r/z]|^{\boldsymbol{t}}_{\boldsymbol{y}}$.

$$\frac{\Gamma \vdash_l \forall_\emptyset z A}{\Gamma \vdash_l A[r/z]}\ \forall^e_\emptyset$$

We are given $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}} \vdash \forall z |A|^{\boldsymbol{t}}_{\boldsymbol{y}}$, where $z, \boldsymbol{y} \notin \mathsf{FV}(\boldsymbol{t}, \boldsymbol{\gamma})$. By a $\forall^e_{[r]}$ we get $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}} \vdash |A[r/z]|^{\boldsymbol{t}}_{\boldsymbol{y}}$.

We now give a comparative treatment of the introduction rules for all quantifiers. They all share the same induction hypothesis, namely that the sequent $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}[z]} \vdash |A(z)|^{\boldsymbol{t}[z]}_{\boldsymbol{v}}$ is provable in NA (by a proof denoted $\mathcal{H}$), where $\boldsymbol{\gamma}[z], \boldsymbol{t}[z]$ are terms extracted from the NA_l proof $\Gamma \vdash_l A(z)$. If z occurs free in A (at least once) then z can occur free in $\boldsymbol{\gamma}, \boldsymbol{t}$.

$(\pm)$ $\mathcal{H}$ is directly a proof of $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}[z]} \vdash |\forall_\pm z A(z)|^{\lambda z.\, \boldsymbol{t}[z]}_{z,\boldsymbol{v}}$, with (c) obviously satisfied.

$(-)$ $\mathcal{H}$ is directly a proof of $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}[z]} \vdash |\forall_- z A(z)|^{\boldsymbol{t}}_{z,\boldsymbol{v}}$, with (c) satisfied due to $z \notin \mathsf{FV}(\boldsymbol{t})$.

$(+)$ To $\mathcal{H}$ one applies a $\forall^i_{[z]}$, which is possible since $z \notin \mathsf{FV}(\Gamma)$ and also $z \notin \mathsf{FV}(\gamma)$. One gets a proof of $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}} \vdash \forall z |A(z)|^{(\lambda z.\, \boldsymbol{t}[z])z}_{\boldsymbol{v}}$, i.e., $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}} \vdash |\forall_+ z A(z)|^{\lambda z.\, \boldsymbol{t}[z]}_{\boldsymbol{v}}$.

$(\emptyset)$ The same as above, but since moreover $z \notin \mathsf{FV}(\boldsymbol{t})$ the lambda-abstraction over $\boldsymbol{t}$ is no longer needed. Applying $\forall^i_{[z]}$ to $\mathcal{H}$ gives directly a proof of $|\Gamma|^{\boldsymbol{u}}_{\boldsymbol{\gamma}} \vdash |\forall_\emptyset z A(z)|^{\boldsymbol{t}}_{\boldsymbol{v}}$.

At $(+)$ and $(\emptyset)$, since z is no longer a free variable in the conclusion sequent (not free in γ by $(\pm)$ and quantified in the conclusion formula) and also no longer appears in the list of refutation variables for $|\forall_+ z\, A(z)|$, $|\forall_\emptyset z\, A(z)|$, it is essential that z is forced not to appear in any of the realizing terms for the conclusion sequent.

3.1 Extension of NA_l with principles that are straightforwardly NA-realizable

By "NA-realizable principle" we understand a generic scheme A in $\mathcal{L}_l$ for which witnesses $\boldsymbol{t}$ exist (possibly as an empty tuple) s.t. $\mathsf{NA} \vdash |A|^{\boldsymbol{t}}_{\boldsymbol{y}}$. We are here interested in such notable A for which $\boldsymbol{t}$ can be directly presented, or at least $\vdash_l A$ is straightforward[9].

Even though $\mathtt{EFQ} : \bot \to A$ is fully provable in NA_l, we can directly give its simple realizers: any type-corresponding terms, in particular canonical $\mathtt{0}$ terms. The verification goes via $\mathtt{EFQ}$, which is provable in NA, as we had mentioned in Sec. 2.1. In contrast, $\mathtt{Stab} : \neg\neg A \to A$ is not fully provable in NA_l : as noted in [7], its usual proof in NA (constructed by induction on A) makes an unavoidable use of contractions over $\neg\neg(B \wedge C)$ for subformulas $(B \wedge C)$ of A, and these are subject to the $\bigstar$ restriction for refutation relevant $B \wedge C$. Even when such $B \wedge C$ obey $\bigstar$, they may lead to the failure of restrictions $(+)$, $(-)$ or $(\emptyset)$. It is thus safe to use NA_l lemmas $\mathtt{Stab}_l : \neg\neg A \to A$ for which $A \in \mathcal{F}$ or A is conjunction-free. Then Theorem 1 guarantees that realizers exist for $\mathtt{Stab}_l$ and produces them for concrete instances of A.

It is well known that Gödel's Dialectica interpretation [1,5] can provide straightforward realizers for certain non-constructive principles, such as $(\mathtt{IP})$ Independence of (universal) Premises, $(\mathtt{AC})$ Axiom of Choice and Markov's principle. However, all of these axioms are usually formulated with strong existence, which is not present in our negative setting. Thus it does not even make sense to consider a Markov's principle for NA_l. Nonetheless, the first author had introduced in [7] certain negative formulations of $\mathtt{IP}$ and $\mathtt{AC}$. Then the second author devised a strengthening of the negative formulation of $\mathtt{AC}$, by means of an automated realizer search, see [17]. We here upgrade and extend these older formulations to account for the new light quantifiers $\forall_+$ and $\forall_-$.

Let us consider the following variants for a negative formulation of $\mathtt{IP}$[10]:

$$\mathtt{IP}_\diamond\colon \quad (A \to \widetilde{\exists}_\diamond y\, B) \to \widetilde{\exists}_\diamond y (A \to B) \qquad y \notin \mathsf{FV}(A), \diamond \in \{\sqcup, \emptyset, +, -, \pm\}$$

$$\widehat{\mathtt{IP}}_\diamond\colon \quad (A_\oplus \to \widetilde{\exists}_\diamond y\, B_\ominus) \to \widetilde{\exists}_\diamond y (A_\oplus \to B_\ominus) \qquad y \notin \mathsf{FV}(A_\oplus), \diamond \in \{\emptyset, +, -, \pm\}$$

As noted in [7], $\mathtt{IP}_\sqcup$ is fully provable in NA (see the treatment of $\mathtt{AxIP}^{\mathtt{cl}}$ on Page 46), but modulo an unavoidable contraction. This proof can be cloned to an NA_l proof of $\mathtt{IP}_\diamond$ with contraction over the formula $C{:}\equiv\forall_\diamond y\,(\neg(A \to B))$. If $\diamond \in \{-, \pm\}$ then C is refutation relevant and the restriction $\bigstar$ is necessary. If $\diamond \in \{\emptyset, +\}$, then $\forall_\diamond y$ is negatively uniform and we need to impose $\bigstar$ only if $\neg(A \to B)$ is refutation relevant.

Lemma 1. *If A and B satisfy $\bigstar$, then $\vdash_l \mathtt{IP}_-$ and $\vdash_l \mathtt{IP}_\pm$. If whenever A is refutation relevant or B is realization relevant, both A and B satisfy $\bigstar$, then $\vdash_l \mathtt{IP}_\emptyset$ and $\vdash_l \mathtt{IP}_+$.*

[9] For all NA_l-provable principles we can automatically get realizers of their concrete instances via the algorithm of Theorem 1. However, light Dialectica is able to directly interpret certain principles formulated over $\mathcal{L}_l$, which are generally not provable in NA_l, like $\widehat{\mathtt{IP}}_\diamond$ below.

[10] Recall that $\sqcup$, which appears at $\mathtt{IP}_\sqcup$ below, is nothing but a placeholder for the empty space.

Axiom $\widehat{\mathtt{IP}}_{\pm}$ was already considered in [7] (as "$\mathtt{IP}^{\mathtt{cl}}_{\mathtt{nc}}$"), where the first author proved that it can be realized in NA by simple projection functionals. It is straightforward to check that the same holds also for the other $\widehat{\mathtt{IP}}_{\diamond}$. The proof for $\widehat{\mathtt{IP}}_{-}$ is identical and the (single) proof for both $\widehat{\mathtt{IP}}_{+}$ and $\widehat{\mathtt{IP}}_{\pm}$ uses a corresponding version of $\mathtt{IP}_{\sqcup}$, which is a NA lemma, as we mentioned.

Next, we consider adding to NA_l the following negative variants of $\mathtt{AC}$:

$$\mathtt{AC}^{\ominus}_{\lhd,\rhd}: \quad \forall_{\lhd} x\, \widetilde{\exists}_{\rhd} y\, A_{\ominus}(x,y) \to \widetilde{\exists}_{\rhd} h\, \forall_{\lhd} x\, A_{\ominus}(x,hx) \qquad \lhd,\rhd \in \{\emptyset,+,-,\pm\}$$

Let us first consider $\mathtt{AC}^{\ominus}_{\pm,\pm}$, an upgrade of "$\mathtt{AC}^{\mathtt{cl}}_{\mathtt{nc}}$" from [7], due to the second author:

$$|\forall_{\pm} x\, \widetilde{\exists}_{\pm} y\, A_{\ominus}(x,y)|^{u,\boldsymbol{v}}_{x} \equiv B(ux,\boldsymbol{v}x,x) \;\; \big[\text{where } B(a,\boldsymbol{b},c) :\equiv \neg\neg |A_{\ominus}(c,a)|^{\boldsymbol{b}}_{\sqcup}\,\big]$$
$$|\widetilde{\exists}_{\pm} h\, \forall_{\pm} x\, A_{\ominus}(x,hx)|^{z,\boldsymbol{f}}_{g} \equiv B\big(zg(g(zg)(\boldsymbol{f}g)), \boldsymbol{f}g(g(zg)(\boldsymbol{f}g)), (g(zg)(\boldsymbol{f}g))\big)$$

It is straightforward that $|\mathtt{AC}^{\ominus}_{\pm,\pm}|$ is an implication between two formulas equivalent to $B(u(gu\boldsymbol{v}), v(gu\boldsymbol{v}), gu\boldsymbol{v})$ if just $Zu\boldsymbol{v}g \equiv u$, $\boldsymbol{F}u\boldsymbol{v}g \equiv \boldsymbol{v}$ and $Xu\boldsymbol{v}g \equiv gu\boldsymbol{v}$. By tedious calculations one can prove that, except for $\mathtt{AC}^{\ominus}_{\pm,+}$ and $\mathtt{AC}^{\ominus}_{-,+}$, all the other $\mathtt{AC}^{\ominus}_{\lhd,\rhd}$ are realizable by simple terms (without constants, mostly projections) in NA. Nevertheless, for $(\lhd,\rhd) \in \{\emptyset,+\} \times \{\emptyset,+\}$, the negative $\mathtt{AC}$ must be added to the verifying system.

It should be clear that the variant $\mathtt{AC}_{\emptyset,\emptyset}$ with A unrestricted is also realizable in $\mathsf{NA} + \mathtt{AC}$ by simple projection functionals. Problems for the general $\mathtt{AC}_{\lhd,\rhd}$ appear only when one progressively adds computational content to $\lhd$ and $\rhd$. The failure of realization (without constants) can already be proved for the variant $\mathtt{AC}^{\oplus}_{\pm,\pm}$, with A realization irrelevant. Nonetheless, all the variants $\mathtt{AC}^{\oplus}_{\lhd,-}$ are easily NA-realizable (by projections, if for $\lhd \in \{\emptyset,-,\pm\}$ one replaces the conclusion with $\widetilde{\exists}_{-} h\, \forall_{\lhd} x\, A_{\oplus}(x,h)$). Even more combinations are possible, with different decorations for the corresponding quantifiers in premise and conclusion: the user can explore the various possibilities by need.

Easy to notice, we can add to NA_l any realization irrelevant axiom $A_{\oplus}$, *provided* that we add its LD-translation as an axiom to NA (whenever we are unable to prove it).

4 An example for the new light quantifiers

Consider the following simple theorem of Arithmetic[11]:

$$\forall x\, \widetilde{\exists} y\, (x < y \wedge P(y)) \;\to\; \forall z\, \widetilde{\exists} u, v\, (u + z < v \wedge P(u) \wedge P(v)) \tag{1}$$

where x, y, z, u and v are natural numbers $\mathbb{N}$, and $P(\cdot)$ is a predicate over $\mathbb{N}$. The proof of this goes as follows: assume $\mathtt{Hyp} :\equiv \forall x\, \widetilde{\exists} y\, (x < y \wedge P(y))$ and fix z. By $\mathtt{Hyp}$ [taking $x := \mathtt{0}$] we (weakly) get an u such that $P(u)$. Then, by $\mathtt{Hyp}$ again [taking $x := u + z$] we (weakly) get also an v, bigger than $u + z$, such that $P(v)$. Q.e.d.

Now, suppose that we want to witness u and v (as functions of z) but not the x in the premise $\mathtt{Hyp}$. Using the hybrid interpretation [8], one can see (1) as

$$!_k\, \forall x\, \exists y\, (x < y \wedge P(y)) \;\multimap\; \forall z\, \exists u, v\, (u + z < v \wedge P(u) \wedge P(v))\,.$$

[11] This example was suggested by Oliva, in the context of hybrid functional interpretations [8]. Note that "$\multimap$" denotes the linear logic [4] implication, see also [12].

The hybrid interpretation [8] of this is (in fact one just carries out a realizability):

$$\exists f, g\, \forall h, z\, [!_k \forall x\, (x < h(x) \wedge P(h(x))) \multimap \\ f(h,z) + z < g(h,z) \wedge P(f(h,z)) \wedge P(g(h,z))] \tag{2}$$

which can be witnessed by taking $f(h,z) := h(\mathsf{0})$ and $g(h,z) := h(h(\mathsf{0}) + z)$.

How would one proceed by means of the light Dialectica [7]? One cannot mark the universal quantification over x as non-computational, since x is used to produce y as $h(x)$. Is there a way to specify that one still wants to *internally* use a variable as computational, but externally we are not interested in the realizer for such a variable? And not by first producing such a witness and subsequently discarding it, but really not producing a realizer at all for that variable?

This example cannot be interpreted with pure Dialectica [1,5]; even if $P(\cdot)$ were decidable, the solution would be too complex. It can also not be directly interpreted with the light annotations proposed in [7] (see further comments on this issue at the end of this section). However, we have a direct positive answer in our upgraded light setting: we can use $\forall_+$ for that universal quantification over x. The input specification (1) can thus be annotated in $\mathcal{L}_l$ as:

$$\forall_+ x\, \widetilde{\exists}_- y\, (x < y \wedge P(y)) \;\to\; \forall_+ z\, \widetilde{\exists}_- u, v\, (u + z < v \wedge P(u) \wedge P(v)) \tag{3}$$

which LD-translates to the following verified specification of NA :

$$\forall x\, (x < hx \wedge P(hx)) \;\to\; \forall z\, (fhz + z < ghz \wedge P(fhz) \wedge P(ghz)) \tag{4}$$

with $h^{\iota\iota}$ the unique challenge variable and f, g of type $(\iota\iota)\iota\iota$ the witness variables. The LD-algorithm will produce (closed) terms $s \equiv \lambda h, z.\, h(\mathsf{0})$ and $t \equiv \lambda h, z.\, h(h(\mathsf{0}) + z)$ s.t.:

$$\forall h\, \big(\forall x\, (x < hx \wedge P(hx)) \;\to\; \forall z\, (shz + z < thz \wedge P(shz) \wedge P(thz))\big) \tag{5}$$

which is immediately seen to be the same result as the one yielded by hybrid functional interpretation, cf. (2). Note that among the annotations in (3) only $\forall_+ x$ represents an optimization, as from an input specification

$$\forall_+ x\, \widetilde{\exists}_\pm y\, (x < y \wedge P(y)) \;\to\; \forall_\pm z\, \widetilde{\exists}_\pm u, v\, (u + z < v \wedge P(u) \wedge P(v))$$

one would get by LD-interpretation the following result, equivalent to (5):

$$\forall h, z\, \big(\forall x\, (x < hx \wedge P(hx)) \;\to\; (shz + z < thz \wedge P(shz) \wedge P(thz))\big) \tag{6}$$

The reason is that $\widetilde{\exists}_\pm$ and $\widetilde{\exists}_-$ are equivalent in front of a quantifier-free formula and the fact that $\forall_\pm z$ makes z a challenge variable would have only altered realizers external to its quantification range, none in our case. Again, the *internal* action of $\forall_+$ is necessary for z, in the conclusion sentence of (3), just as it was (as we explained above) for x in the premise of (3). Both quantifications over x and z have an essential positive computational content. The difference is that, whereas for z the negative content of the quantification is inessential, for x it is an important optimization to remove the negative

content of its quantification. Otherwise the contraction over $\forall_{\pm} x \, \widetilde{\exists}_{-} y \, (x < y \wedge P(y))$ would be computationally relevant and a useless realizer for x would be produced.

Note that the older LD-interpretation of [7] is not really unusable for this example: identical results are obtained by replacing (3) with

$$\widetilde{\exists}_{\pm} h \, \forall_{\emptyset} x \, (x < hx \wedge P(hx)) \; \rightarrow \; \forall_{\pm} z \, \widetilde{\exists}_{\pm} u, v \, (u + z < v \wedge P(u) \wedge P(v)) \, .$$

The user would have to take into account the parameter $h^{\iota\iota}$ in the realizing terms anyway, as this is forced by the verified specification (4). Nonetheless we find it more convenient to be able to start with less explicit specifications and use the automated mechanism to unwind the functionals which are implicit in the input specification.

As an illustration for an effective use of the "$-$" quantifier, the conclusion of (3) can be changed to $\widetilde{\exists}_{-} u, v \, \forall_{-} z \, (u + z < v \wedge P(u) \wedge P(v))$. Even though this looks strange (if not faulty), the LD-translation *will* explicitate the dependency of u and v over the parameter z. The conclusion of (4) becomes just $fhz + z < ghz \wedge P(fhz) \wedge P(ghz)$, but the inner quantification over z in (5) was inessential anyway: as a free LD-variable, z is still a parameter, just like in (6). The final result remains unchanged!

5 List reversal - a new application for the uniform quantifier

We here treat an example for LD-extraction from a proof in classical logic that any list can be reversed. The case study was originally suggested by Berger [2] in the context of refined A-translation [3]. He showed that by using his uniform universal quantifier one can remove an unnecessary parameter from the extracted program and thus decrease its time complexity from quadratic to linear. We demonstrate that the same good program can also be obtained via LD-extraction, modulo an enhanced light annotation of Berger's proof for the weak existence of the reversed list. Moreover, in our case the uniform quantifier will not be used just to improve complexity, but even to make Dialectica extraction possible at all when list reversal is not a priori assumed to be decidable.

Let us extend our language with a type L for finite lists of natural numbers. We will use "l" for list variables and denote the constructors for L by $\mathtt{nil}$ and $n :: l$, abbreviating $x :: \mathtt{nil}$ as "x:". We also need to have a recursion constant $\mathcal{R}$ for lists, as well as an induction principle $\mathtt{Ind}_L$ and rewrite rules for $\mathcal{R}$. These are all presented in Table 6. As with $\mathtt{Ind}_\iota$, we assume that n, l do not occur freely in Γ, Δ, but l does occur in $A(l)$.

$$\frac{\Gamma \vdash A(\mathtt{nil}) \quad \Delta \vdash A(l) \to A(n :: l)}{\Gamma, \Delta \vdash A(l)} \; \mathtt{Ind}_L \qquad \begin{array}{c} \mathcal{R}\,\mathtt{nil}\,s\,t \hookrightarrow s \\ \mathcal{R}\,(n :: l)\,s\,t \hookrightarrow t\,n\,l\,(\mathcal{R}\,l\,s\,t) \end{array}$$

Table 6. List induction and recursion

We will not treat the soundness of $\mathtt{Ind}_L$ here, since it is very similar to the soundness of $\mathtt{Ind}_\iota$. The notable difference is that we use the appropriate recursion constant $\mathcal{R}$ for defining witnesses of $A(l)$ and challengers of $\Gamma \uplus \Delta$. We define the append function $(:\!+\!:)^{LLL}$ by $l_1 :\!+\!: l_2 :\equiv \mathcal{R}\,l_1\,l_2\,(\lambda n, l_1, p^L.\, n :: p)$ and the decidable equality $\mathtt{Eq}_L^{LLo}$ by:

$$\begin{array}{rcl} \mathtt{Eq}_L & :\equiv & \lambda l_1.\mathcal{R}\,l_1\,\big(\lambda l_2.\mathcal{R}\,l_2\,\mathtt{T}\,(\lambda n, l_2, q^o.\mathtt{F})\big) \\ & & \big(\lambda m, l_1, p^{Lo}, l_2.\mathcal{R}\,l_2\,\mathtt{F}\,(\lambda n, l_2, q^o.\mathtt{If}\,(\mathtt{Eq}_\iota mn)\,(p\,l_2)\,\mathtt{F})\big) \end{array}$$

As for the other ground types, we define predicate equality $l_1 =_L l_2 :\equiv \mathtt{at}(\mathtt{Eq}_L\, l_1\, l_2)$. We introduce a new binary predicate symbol "Rev" of arity (L, L) for expressing the fact that one list is the reversal of another. Formally, we will prove classically that there exists a function with a graph Rev, satisfying the assumptions

$$\mathrm{Rev}(\mathtt{nil}, \mathtt{nil}) \tag{7}$$

$$\forall_\emptyset x, l_1, l_2\, \big(\mathrm{Rev}(l_1, l_2) \to \mathrm{Rev}(l_1{:}+{:}x{:}, x :: l_2)\big) \tag{8}$$

Note that Rev can be considered as an inductively defined predicate with introduction axioms (7) and (8) (cf. [13]). They do not ensure uniqueness of Rev; to achieve this we would need to add an elimination axiom. Then we could use (7) and (8) as clauses for a decision procedure for Rev, recursively defined on its second argument. Such procedure would have quadratic complexity on the length of the list, because of appending an element to l_1 at each recursion step. Moreover, using the clauses (7) and (8) we could directly define a list reversal program and prove the theorem constructively [2,13]. Again, such program would execute in quadratic time.

Since any referral to the decidability of Rev would imply an at least quadratic time complexity, we choose to consider Rev as undecidable. Thus we no longer need the restriction for uniqueness of Rev and hence we will prove the theorem for *any* predicate satisfying (7) and (8). Formally, we will show (following the proof in [2]) that

$$\forall_+ l'\, \widetilde{\exists}_- l''\, \mathrm{Rev}(l', l'') \tag{9}$$

We first show that if a list l_0 is not reversible, then none of its initial segments l_1 is:

$$\forall_\emptyset l_0 \Big(\forall_- l(\mathrm{Rev}(l_0, l) \to \bot) \to \forall_- l_2 \forall_\emptyset l_1 \big(l_1{:}+{:}l_2 = l_0 \to \forall_- l(\mathrm{Rev}(l_1, l) \to \bot)\big)\Big) \tag{10}$$

Fix l_0 and assume $a{:}\forall_- l(\mathrm{Rev}(l_0, l) \to \bot)$. We proceed by induction on l_2 to show

$$\forall_\emptyset l_1 \big(l_1{:}+{:}l_2 = l_0 \to \forall_- l(\mathrm{Rev}(l_1, l) \to \bot)\big) \tag{11}$$

For the base case we have $l_1 = l_0$ so we can use the assumption a. For the step case $l_2 {\equiv} n :: l_2'$ we fix l_1, l and assume $\mathrm{Rev}(l_1, l)$. By (8) we have $\mathrm{Rev}(l_1{:}+{:}n{:}, n :: l)$. We can then use the induction hypothesis (11) for l_2' with $l_1 \mapsto l_1{:}+{:}n{:}$ and $l \mapsto n :: l$ to end the proof. Now (9) follows from (10) by setting $l_0, l_2 \mapsto l'$ and $l_1 \mapsto \mathtt{nil}$ and using (7).

We proceed with the LD-extraction stepwise. The global assumptions (7) and (8) are computationally irrelevant. The induction formula (11) has empty positive content and a list variable l as negative content. The unique computationally relevant open avar is a and it appears only in the base case, so the extracted term for (10) is

$$t_L {:}\equiv \lambda l_2.\mathcal{R}\, l_2\, (\lambda l.l)\, (\lambda n, l_2, p^{LL}, l.p(n :: l))$$

The final term extracted from the proof of (9) is $t{:}\equiv \lambda l'.t_L\, l'\, \mathtt{nil}$. We thus obtain the usual linear reverse list algorithm with the use of an auxiliary function t_L. Here l plays the role of an accumulator, initialized with $\mathtt{nil}$ in t.

Let us review the role of the light quantifiers. All $\forall_-$ quantifiers in (10) can be changed to $\forall_\pm$ without impact on the extracted term, because the corresponding quantified formulas are realization irrelevant. Substituting $\forall_\emptyset l_1$ with $\forall_+ l_1$ is safe for the same reason. Changing $\forall_\emptyset l_0$ to $\forall_\pm l_0$ would result in a redundant parameter in t_L, since l_0 has no computational use in the lemma. Hence it is better that this quantifier remains fully uniform. A more interesting effect appears if $\forall_\emptyset l_1$ is replaced with $\forall_- l_1$ or $\forall_\pm l_1$.

Then the negative content of (11) is already a pair of lists l_1, l, so the extracted term t_L changes to

$$t'_L :\equiv \lambda l_2.\mathcal{R}\, l_2\, (\lambda l_1, l.l)\, (\lambda n, l_2, p, l_1, l.p(l_1{:}{+}{:}n{:})(n :: l)),$$

being invoked as $t'_L\, l'\, \mathtt{nil}\, \mathtt{nil}$. Note that in this case there is an unnecessary quadratic computation on the parameter l_1, which is being dropped in the base case. Therefore, quantifying l_1 negatively uniformly here has the same favourable effect on complexity as was noted by Berger for the case of refined A-translation [2].

Now consider the light quantifiers in the global assumption (8). Obviously, all the quantifiers can be safely replaced with $\forall_+$, since the kernel formula is quantifier-free. However, if we added negative computational meaning to any of the three quantifiers, we would introduce additional extracted terms for the corresponding variables(s). One consequence of this change is that we would be forced to add negative content to the quantifiers for l' in (9) as well as l_0 and l_1 in (10) in order to avoid violating the restriction $(+)$. However, there is an even more serious problem: since (8) is used in the step case of the induction, a boolean test over its LD-translation would be necessary. But since Rev is undecidable, such a case distinction is not possible at all! If we try to repair the situation by using the decision procedure for Rev, suggested above, the overall complexity will raise to cubic. Therefore, the use of negatively uniform quantifiers in (8) is really essential. Note that whether these quantifiers are uniform or not makes no difference when extracting by refined A-translation.

The discussion above shows that the linear list reversal algorithm can be extracted from a proof using only $\forall_\pm$ and $\forall_\emptyset$. Hence, even though we used a finer light annotation here, this example is still in the scope of Light Dialectica as defined in [7].

6 Future work - a light decorating algorithm

Having four variants of each quantifier, it becomes important to design a decorating algorithm that, starting with an NA_l proof in which all quantifiers are coloured $\emptyset$, will explore the possibilities for consistent colourings so that the input specification LD-translates to the exact verified specification desired by the user. At full power, such algorithm implies that we are able to accurately determine the set of free variables of the *normalized* extracted terms, since the normal form of a term may contain fewer free variables. Also, input proofs should rather be presented in normal form, since cut formulas may require different colourings, which would force the elimination of such cut. Overall, this gets rather complex and we may need to trade accuracy for effectivity.

Since the light decorating algorithm needs to calculate the final LD-extracted terms anyway, we can regard it as an enhancement (optimizing extension) of the already presented light Dialectica interpretation.

Acknowledgement: The idea for the new light quantifiers was born during a discussion with Oliva, mainly concerning the (counter)example from Section 4. Thanks to him, also for the indirect suggestion on upgrading `CMP`. The original direct treatment under Dialectica (with pair types) of the full Induction Rule is due to Schwichtenberg [14]. We would like to thank both of them for their comments on early drafts of this paper.

References

1. J. Avigad and S. Feferman. Gödel's functional ("Dialectica") interpretation. In S. R. Buss, editor, *Handbook of proof theory*, volume 137 of *Studies in Logic and the Foundations of Mathematics*, pages 337–405. North Holland, Amsterdam, 1998.
2. U. Berger. Uniform Heyting Arithmetic. *Annals Pure Applied Logic*, 133:125–148, 2005.
3. U. Berger, H. Schwichtenberg, and W. Buchholz. Refined program extraction from classical proofs. *Annals of Pure and Applied Logic*, 114:3–25, 2002.
4. J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.
5. K. Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, 12:280–287, 1958.
6. M.-D. Hernest. Light Functional Interpretation. *Lecture Notes in Computer Science*, 3634:477 – 492, 2005. Computer Science Logic: 19th International Workshop, CSL 2005.
7. M.-D. Hernest. *Optimized programs from (non-constructive) proofs by the light (monotone) Dialectica interpretation*. PhD Thesis, École Polytechnique and Universität München, 2006. `http://www.brics.dk/~danher/teza/thesfull.pdf`.
8. M.-D. Hernest and P. Oliva. Hybrid functional interpretations. In A. Beckmann, C. Dimitracopoulos, and B. Löwe, editors, *Logic and Theory of Algorithms: 4th "Computability in Europe" (CiE) conference, Athens, Greece*, volume 5028 of *Lecture Notes in Computer Science*, pages 251–260. Springer Verlag, 2008.
9. U. Kohlenbach. A note on Spector's quantifier-free rule of extensionality. *Archive for Mathematical Logic*, 40:89–92, 2001.
10. P. Oliva. An analysis of Gödel's Dialectica interpretation via linear logic. To appear in *Dialectica*, for preprint see `http://www.dcs.qmul.ac.uk/~pbo/`.
11. P. Oliva. Unifying functional interpretations. *Notre Dame Journal of Formal Logic*, 47(2):263–290, 2006.
12. P. Oliva. Computational interpretations of classical linear logic. In *Proceedings of WoLLIC'07, LNCS 4576*, pages 285–296. Springer, 2007.
13. H. Schwichtenberg. Content in proofs of list reversal. Marktoberdorf Summer School (2007), `http://www.math.lmu.de/~schwicht/papers/mod07/mod07.pdf`.
14. H. Schwichtenberg. Dialectica interpretation of well-founded induction. To appear in *Mathematical Logic Quarterly*, for preprint see `http://www.math.lmu.de/~schwicht/publikationen.html`.
15. H. Schwichtenberg. Minimal Logic for Computable Functionals. MinLog documentation, `http://www.math.lmu.de/~minlog/minlog/mlcf.pdf`, December 2005.
16. C. Spector. Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics. In J. C. E. Dekker, editor, *Recursive Function Theory: Proceedings of Symposia in Pure Mathematics*, volume 5, pages 1–27. American Mathematical Society, Providence, Rhode Island, U.S.A., 1962.
17. T. Trifonov. Finding Dialectica realisers for axioms. `http://www.math.lmu.de/~trifonov/talks/20070414/`, April 2007. Invited talk at the 6th *Proof, Computation, Complexity* workshop, `http://www.cs.swan.ac.uk/pcc07/`.
18. A. S. Troelstra. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, volume 344 of *Lecture Notes in Mathematics*. Springer-Verlag, 1973.