

Dualized Type Theory

Harley Eades III

University of Iowa
14 Maclean Hall
Iowa City, IA 52242-1419

harley-eades@uiowa.edu

Aaron Stump

University of Iowa
14 Maclean Hall
Iowa City, IA 52242-1419

aaron-stump@uiowa.edu

Ryan McCleary

University of Iowa
14 Maclean Hall
Iowa City, IA 52242-1419

ryan-mccleary@uiowa.edu

We propose a new bi-intuitionistic type theory called Dualized Type Theory (DTT). It is a type theory with perfect intuitionistic duality, and corresponds to a single-sided polarized sequent calculus. We prove DTT strongly normalizing, and prove type preservation. DTT is based on a new propositional bi-intuitionistic logic called Dualized Intuitionistic Logic (DIL) that builds on Pinto and Uustalu's logic L. DIL is a simplification of L which removes several admissible inference rules from L while maintaining consistency and completeness. Furthermore, DIL is defined using a dualized syntax by labeling formulas and logical connectives with polarities thus reducing the number of inference rules needed to define the logic. We give a direct proof of consistency, but prove completeness by reduction to L.

1 Introduction

Classical logic is rich with duality. Using the De Morgan dualities it is straightforward to prove that conjunction is dual to disjunction and negation is self dual. In addition, it is also possible to prove that $\neg A \wedge B$ is dual to implication. In intuitionistic logic these dualities are no longer provable, but in [21] Rauszer gives a conservative extension of the Kripke semantics for intuitionistic logic that not only models conjunction, disjunction, negation, and implication, but also the dual to implication, by introducing a new logical connective. The usual interpretation of implication in a Kripke model is as follows:

$$\llbracket A \rightarrow B \rrbracket_w = \forall w'. w \leq w' \rightarrow \llbracket A \rrbracket_{w'} \rightarrow \llbracket B \rrbracket_{w'}$$

Now Rauszer took the dual of the previous interpretation to obtain the following:

$$\llbracket A - B \rrbracket_w = \exists w'. w' \leq w \wedge \neg \llbracket A \rrbracket_{w'} \wedge \llbracket B \rrbracket_{w'}$$

This is called subtraction or exclusion. Propositional bi-intuitionistic logic is a conservative extension of propositional intuitionistic logic with perfect duality. That is, it contains the logical connectives for disjunction, conjunction, implication, and subtraction, and it is sound and complete with respect to the Rauszer's extended Kripke semantics.

Propositional bi-intuitionistic (BINT) logic is fairly unknown in computer science. Filinski studied a fragment of BINT logic in his investigation into first class continuations in [12]. Crolard introduced a logic and corresponding type theory called subtractive logic, and showed it can be used to study constructive coroutines in [7, 8]. He initially defined subtractive logic in sequent style with the Dragalin restriction, and then defined the corresponding type theory in natural deduction style by imposing a restriction on Parigot's $\lambda\mu$ -calculus in the form of complex dependency tracking. Just as linear logicians have found – for example in [22] – Pinto and Uustalu were able to show that imposing the Dragalin restriction in subtractive logic results in a failure of cut elimination [20]. They recover cut elimination by

proposing a new BINT logic called L that lifts the Dragalin restriction by labeling formulas and sequents with nodes and graphs respectively; this labeling corresponds to placing constraints on the sequents where the graphs can be seen as abstract Kripke models. Goré et. al. also proposed a new BINT logic that enjoys cut elimination using nested sequents; however it is currently unclear how to define a type theory with nested sequents [14]. Bilinear logic in its intuitionistic form is a linear version of BINT and has been studied by Lambek in [17, 18]. Biasi and Aschieri propose a term assignment to polarized bi-intuitionistic logic in [6]. One can view the polarities of their logic as an internalization of the polarities of the logic we propose in this article. Bellin has studied BINT similar to that of Biasi and Aschieri from a philosophical perspective in [2, 3, 4], and he defined a linear version of Crolard’s subtractive logic for which he was able to construct a categorical model using linear categories in [5].

Contributions. The contributions of this paper are a new formulation of Pinto and Uustalu’s BINT labeled sequent calculus L called Dualized Intuitionistic Logic (DIL) and a corresponding type theory called Dualized Type Theory (DTT). DIL is a single-sided polarized formulation of Pinto and Uustalu’s L, and builds on L by removing the following rules (see Section 2 for a complete definition of L):

$$\begin{array}{c}
 \frac{\Gamma \vdash_{G \cup \{(n,n)\}} \Delta}{\Gamma \vdash_G \Delta} \quad \text{REFL} \qquad \frac{\begin{array}{c} n_1 G n_2 \\ n_2 G n_3 \\ \Gamma \vdash_{G \cup \{(n_1, n_3)\}} \Delta \end{array}}{\Gamma \vdash_G \Delta} \quad \text{TRANS} \\
 \\
 \frac{n G n'}{\Gamma, n : T, n' : T \vdash_G \Delta} \quad \text{MONL} \qquad \frac{n' G n}{\Gamma \vdash_G n' : T, n : T, \Delta} \quad \text{MONR} \\
 \frac{\Gamma, n : T \vdash_G \Delta}{\Gamma, n : T \vdash_G \Delta} \quad \text{MONL} \qquad \frac{\Gamma \vdash_G n : T, \Delta}{\Gamma \vdash_G n : T, \Delta} \quad \text{MONR}
 \end{array}$$

We show that in the absence of the previous rules DIL still maintains consistency and completeness. Furthermore, DIL is defined using a dualized syntax which reduces the number of inference rules needed to define the logic. Again, DIL is a single-sided sequent calculus with multiple conclusions and thus must provide a means of moving conclusions from left to right. This is done in DIL using cuts on hypotheses. We call these types of cuts “axiom cuts.”

Now we consider BINT logic to be the closest extension of intuitionistic logic to classical logic while maintaining constructivity. BINT has two forms of negation, one defined as usual, $\neg A \stackrel{\text{def}}{=} A \rightarrow \perp$, and a second defined in terms of subtraction, $\sim A \stackrel{\text{def}}{=} \top - A$. The latter we call “non- A ”. Now in BINT it is possible to prove $A \vee \sim A$ for any A [7]. Furthermore, when the latter is treated as a type in DTT, the inhabitant is a continuation without a canonical form, because the inhabitant contains as a subexpression an axiom cut. Thus, the presence of these continuations prevents the canonicity result for a type theory – like DTT – from holding. Thus, if general cut elimination was a theorem of DIL, then $A \vee \sim A$ would not be provable. So DIL must contain cuts that cannot be eliminated. This implies that DIL does not enjoy general cut elimination, but all cuts other than axiom cuts can be eliminated. Throughout the sequel we define “cut elimination” as the elimination of all cuts other than axiom cuts, and we call DIL “cut free” with respect to this definition of cut elimination. The latter point is similar to Wadler’s dual calculus [25].

Finally, we present a computer-checked proof – in Agda – of consistency for DIL with respect to Rauszer’s Kripke semantics for BINT logic, prove completeness of DIL by reduction to Pinto and Uustalu’s L, and show type preservation and strong normalization for DTT. We show the latter using a version of Krivine’s classical realizability by translating DIL into a classical logic.

The contributions of this article are subgoals of a larger one. Due to the rich duality in BINT logic we believe it shows promise of being a logical foundation for induction and coinduction, because induc-

$$\begin{array}{ll}
\text{(formulas)} & A, B, C ::= \top \mid \perp \mid A \supset B \mid A \prec B \mid A \wedge B \mid A \vee B \\
\text{(graphs)} & G ::= \cdot \mid (n, n') \mid G, G' \\
\text{(contexts)} & \Gamma ::= \cdot \mid n : A \mid \Gamma, \Gamma'
\end{array}$$

Figure 1: Syntax of L.

tion is dual to coinduction. Our working hypothesis is that a logical foundation based on intuitionistic duality will allow the semantic duality between induction and coinduction to be expressed in type theory, yielding a solution to the problems with these important features in existing systems. For example, Agda restricts how inductive and coinductive types can be nested (see the discussion in [1]), while Coq supports general mixed inductive and coinductive data, but in doing so, sacrifices type preservation.

The rest of this paper is organized as follows. We first introduce Pinto and Uustalu’s L calculus in Section 2, and then DIL in Section 3. Then we prove DIL consistent and complete (with only axiom cuts) in Section 3.1 and Section 3.2 respectively. Following DIL we introduce DTT in Section 4, and its metatheory in Section 5. All proofs can be found in a companion report available at [11]. All of the mathematical content of this paper was typeset with the help of Ott [23].

2 Pinto and Uustalu’s L

In this section we briefly introduce Pinto and Uustalu’s L from [20]. The syntax for formulas, graphs, and contexts of L are defined in Figure 1, while the inference rules are defined in Figure 2. The formulas include true and false denoted $\top$ and $\perp$ respectively, implication and subtraction denoted $A \supset B$ and $A \prec B$ respectively, and finally, conjunction and disjunction denoted $A \wedge B$ and $A \vee B$ respectively. So we can see that for every logical connective its dual is a logical connective of the logic. This is what we meant by BINT containing perfect intuitionistic duality in the introduction. Sequents have the form $\Gamma \vdash_G n : A, \Delta$, where Γ and Δ are multisets of formulas labeled by a node, G is the abstract Kripke model or sometimes referred to as simply the graph of the sequent, and n is a node in G .

Graphs are treated as sets of edges and we denote $(n_1, n_2) \in G$ by $n_1 G n_2$. Furthermore, we denote the union of two graphs G and G' as $G \cup G'$. Now each formula present in a sequent is labelled with a node in the graph. This labeling is denoted $n : A$ and should be read as the formula A is true at the node n . We denote the operation of constructing the list of nodes in a graph or context by $|G|$ and $|\Gamma|$ respectively. The reader should note that it is possible for some nodes in the sequent to not appear in the graph. For example, the sequent $n : A \vdash n : A, \cdot$ is a derivable sequent. Now the complete graph can always be recovered if needed by using the graph structural rules REFL, TRANS, MONL, and MONR.

The labeling on formulas essentially adds constraints to the set of Kripke models. This is evident in the proof of consistency for DIL in Section 3.1; see the definition of validity. Consistency of L is stated in [20] without a detailed proof, but is proven complete with respect to Rauszer’s Kripke semantics using a counter model construction. In Section 3 we give a translation of the formulas of L into the formulas of DIL, and in Section 3.2 we will give a translation of the rest of L into DIL which will be used to conclude completeness of DIL.

3 Dualized Intuitionistic Logic (DIL)

The syntax for polarities, formulas, and graphs of DIL is defined in Figure 3, where **a** ranges over atomic formulas. The following definition shows that DIL’s formulas are simply polarized versions of

$$\begin{array}{c}
\frac{\Gamma \vdash_{G \cup \{(n,n)\}} \Delta}{\Gamma \vdash_G \Delta} \text{ REFL} \qquad \frac{n_1 Gn_2 \quad n_2 Gn_3}{\Gamma \vdash_{G \cup \{(n_1,n_3)\}} \Delta} \text{ TRANS} \qquad \frac{}{\Gamma, n : T \vdash_G n : T, \Delta} \text{ HYP} \\
\\
\frac{nGn'}{\Gamma, n : T, n' : T \vdash_G \Delta} \text{ MONL} \qquad \frac{n'Gn}{\Gamma \vdash_G n' : T, n : T, \Delta} \text{ MONR} \qquad \frac{\Gamma \vdash_G \Delta}{\Gamma, n : \top \vdash_G \Delta} \text{ TRUEL} \\
\\
\frac{}{\Gamma \vdash_G n : \top, \Delta} \text{ TRUER} \qquad \frac{}{\Gamma, n : \perp \vdash_G \Delta} \text{ FALSEL} \qquad \frac{\Gamma \vdash_G \Delta}{\Gamma \vdash_G n : \perp, \Delta} \text{ FALSER} \\
\\
\frac{\Gamma, n : T_1, n : T_2 \vdash_G \Delta}{\Gamma, n : T_1 \wedge T_2 \vdash_G \Delta} \text{ ANDL} \qquad \frac{\Gamma \vdash_G n : T_1, \Delta \quad \Gamma \vdash_G n : T_2, \Delta}{\Gamma \vdash_G n : T_1 \wedge T_2, \Delta} \text{ ANDR} \qquad \frac{\Gamma, n : T_1 \vdash_G \Delta \quad \Gamma, n : T_2 \vdash_G \Delta}{\Gamma, n : T_1 \vee T_2 \vdash_G \Delta} \text{ DISJL} \\
\\
\frac{\Gamma \vdash_G n : T_1, n : T_2, \Delta}{\Gamma \vdash_G n : T_1 \vee T_2, \Delta} \text{ DISJR} \qquad \frac{nGn' \quad \Gamma \vdash_G n' : T_1, \Delta \quad \Gamma, n' : T_2 \vdash_G \Delta}{\Gamma, n : T_1 \supset T_2 \vdash_G \Delta} \text{ IMPL} \\
\\
\frac{n' \notin |G|, |\Gamma|, |\Delta| \quad \Gamma, n' : T_1 \vdash_{G \cup \{(n,n')\}} n' : T_2, \Delta}{\Gamma \vdash_G n : T_1 \supset T_2, \Delta} \text{ IMPR} \qquad \frac{n' \notin |G|, |\Gamma|, |\Delta| \quad \Gamma, n' : T_1 \vdash_{G \cup \{(n,n')\}} n' : T_2, \Delta}{\Gamma, n' : T_1 \prec T_2 \vdash_G \Delta} \text{ SUBL} \\
\\
\frac{n'Gn \quad \Gamma \vdash_G n' : T_1, \Delta \quad \Gamma, n' : T_2 \vdash_G \Delta}{\Gamma \vdash_G n : T_1 \prec T_2, \Delta} \text{ SUBR}
\end{array}$$

Figure 2: Inference Rules for L.

(polarities)	$p ::= + -$
(formulas)	$A, B, C ::= \mathbf{a} \langle p \rangle A \rightarrow_p B A \wedge_p B$
(graphs)	$G ::= \cdot n \preceq_p n' G, G'$
(contexts)	$\Gamma ::= \cdot pA @ n \Gamma, \Gamma'$

Figure 3: Syntax for DIL.

$$\begin{array}{c}
\frac{G \vdash n \preceq_p^* n'}{G; \Gamma, pA @ n, \Gamma' \vdash pA @ n'} \text{ AX} \qquad \frac{}{G; \Gamma \vdash p \langle p \rangle @ n} \text{ UNIT} \\
\\
\frac{G; \Gamma \vdash pA @ n \quad G; \Gamma \vdash pB @ n}{G; \Gamma \vdash p(A \wedge_p B) @ n} \text{ AND} \qquad \frac{G; \Gamma \vdash pA_d @ n}{G; \Gamma \vdash p(A_1 \wedge_{\bar{p}} A_2) @ n} \text{ ANDBAR} \\
\\
\frac{n' \notin |G|, |\Gamma| \quad (G, n \preceq_p n'); \Gamma, pA @ n' \vdash pB @ n'}{G; \Gamma \vdash p(A \rightarrow_p B) @ n} \text{ IMP} \qquad \frac{G \vdash n \preceq_{\bar{p}}^* n' \quad G; \Gamma \vdash \bar{p}A @ n' \quad G; \Gamma \vdash pB @ n'}{G; \Gamma \vdash p(A \rightarrow_{\bar{p}} B) @ n} \text{ IMPBAR} \\
\\
\frac{G; \Gamma, \bar{p}A @ n \vdash +B @ n' \quad G; \Gamma, \bar{p}A @ n \vdash -B @ n'}{G; \Gamma \vdash pA @ n} \text{ CUT}
\end{array}$$

Figure 4: Inference Rules for DIL.

L's formulas.

Definition 1. *The following defines a translation of formulas of L to formulas of DIL:*

$$\begin{array}{lll}
\lceil \top \rceil = \langle + \rangle & \lceil A \wedge B \rceil = \lceil A \rceil \wedge_+ \lceil B \rceil & \lceil A \supset B \rceil = \lceil A \rceil \rightarrow_+ \lceil B \rceil \\
\lceil \perp \rceil = \langle - \rangle & \lceil A \vee B \rceil = \lceil A \rceil \wedge_- \lceil B \rceil & \lceil B \prec A \rceil = \lceil A \rceil \rightarrow_- \lceil B \rceil
\end{array}$$

We represent graphs as lists of edges denoted $n_1 \preceq_p n_2$, where we consider the edge $n_1 \preceq_+ n_2$ to mean that there is a path from n_1 to n_2 , and the edge $n_1 \preceq_- n_2$ to mean that there is a path from n_2 to n_1 . Lastly, contexts denoted Γ are represented as lists of formulas. Throughout the sequel we denote the opposite of a polarity p by $\bar{p}$. This is defined by $\bar{+} = -$ and $\bar{-} = +$. The inference rules for DIL are in Figure 4.

The sequent has the form $G; \Gamma \vdash pA @ n$ which when p is positive (resp. negative) can be read as the formula A is true (resp. false) at node n in the context Γ with respect to the graph G . Note that to the metavariable d in the premise of the ANDBAR rule ranges over the set $\{1, 2\}$ and prevents the need for two rules. The inference rules depend on a reachability judgment that provides a means of proving when a node is reachable from another within some graph G . This judgment is defined in Figure 5. In addition, the IMP rule depends on the operations $|G|$ and $|\Gamma|$ which simply compute the list of all the nodes in G and Γ respectively. The condition $n' \notin |G|, |\Gamma|$ in the IMP rule is required for consistency.

The most interesting inference rules of DIL are the rules for implication and coimplication from Figure 4. Let us consider these two rules in detail. These rules mimic the definitions of the interpretation of implication and coimplication in a Kripke model. The IMP rule states that the formula $p(A \rightarrow_p B)$ holds at node n if assuming $pA @ n'$ for an arbitrary node n' reachable from n , then $pB @ n'$ holds. Notice that when p is positive n' will be a future node, but when p is negative n' will be a past node. Thus,

$$\begin{array}{c}
\frac{}{G, n \preceq_p n', G' \vdash n \preceq_p^* n'} \text{REL_AX} \qquad \frac{}{G \vdash n \preceq_p^* n} \text{REL_REFL} \\
\\
\frac{G \vdash n \preceq_p^* n' \quad G \vdash n' \preceq_p^* n''}{G \vdash n \preceq_p^* n''} \text{REL_TRANS} \qquad \frac{G \vdash n' \preceq_p^* n}{G \vdash n \preceq_p^* n'} \text{REL_FLIP}
\end{array}$$

Figure 5: Reachability Judgment for DIL.

universally quantifying over past and future worlds is modeled here by adding edges to the graph. Now the IMPBAR rule states the formula $p(A \rightarrow_{\bar{p}} B)$ is derivable if there exists a node n' that is provably reachable from n , $\bar{p}A$ is derivable at node n' , and $pB @ n'$ is derivable at node n' . When p is positive n' will be a past node, but when p is negative n' will be a future node. This is exactly dual to implication. Thus, existence of past and future worlds is modeled by the reachability judgment.

Before moving on to proving consistency and completeness of DIL we first show that the formula $A \wedge_{-} \sim A$ has a proof in DIL that contains a cut that cannot be eliminated. This also serves as an example of a derivation in DIL. Consider the following where we leave off the reachability derivations for clarity and $\Gamma' \equiv -(A \wedge_{-} \sim A) @ n, -A @ n$:

$$\begin{array}{c}
\frac{}{G; \Gamma, \Gamma' \vdash -A @ n} \text{AX} \quad \frac{}{G; \Gamma, \Gamma' \vdash \langle + \rangle @ n} \text{UNIT} \\
\hline
\frac{}{G; \Gamma, \Gamma' \vdash + \sim A @ n} \text{IMPBAR} \\
\hline
\frac{}{G; \Gamma, \Gamma' \vdash (A \wedge_{-} \sim A) @ n} \text{ANDBAR} \quad \frac{}{G; \Gamma, \Gamma' \vdash -(A \wedge_{-} \sim A) @ n} \text{AX} \\
\hline
\frac{}{G; \Gamma, -(A \wedge_{-} \sim A) @ n \vdash +A @ n} \text{CUT} \\
\hline
\frac{}{G; \Gamma, -(A \wedge_{-} \sim A) @ n \vdash (A \wedge_{-} \sim A) @ n} \text{ANDBAR}
\end{array}$$

Now using only an axiom cut we may conclude the following derivation:

$$\frac{G; \Gamma, -(A \wedge_{-} \sim A) @ n \vdash (A \wedge_{-} \sim A) @ n \quad \frac{}{G; \Gamma, -(A \wedge_{-} \sim A) @ n \vdash -(A \wedge_{-} \sim A) @ n} \text{AX}}{G; \Gamma \vdash (A \wedge_{-} \sim A) @ n} \text{CUT}$$

The reader should take notice to the fact that all cuts within the previous two derivations are axiom cuts, where the inner most cut uses the hypothesis of the outer cut. Therefore, neither can be eliminated.

3.1 Consistency of DIL

In this section we prove consistency of DIL with respect to Rauszer's Kripke semantics for BINT logic. All of the results in this section have been formalized in the Agda proof assistant¹. We begin by first defining a Kripke frame.

Definition 2. A *Kripke frame* is a pair (W, R) of a set of worlds W , and a preorder R on W .

Then we extend the notion of a Kripke frame to include an evaluation for atomic formulas resulting in a Kripke model.

Definition 3. A *Kripke model* is a tuple (W, R, V) , such that, (W, R) is a Kripke frame, and V is a binary monotone relation on W and the set of atomic formulas of DIL.

Now we can interpret formulas in a Kripke model as follows:

¹Agda source code is available at <https://github.com/heades/DIL-consistency>

Definition 4. The interpretation of the formulas of DIL in a Kripke model (W, R, V) is defined by recursion on the structure of the formula as follows:

$$\begin{array}{ll} \llbracket \langle + \rangle \rrbracket_w = \top & \llbracket A \wedge_+ B \rrbracket_w = \llbracket A \rrbracket_w \wedge \llbracket B \rrbracket_w \\ \llbracket \langle - \rangle \rrbracket_w = \perp & \llbracket A \wedge_- B \rrbracket_w = \llbracket A \rrbracket_w \vee \llbracket B \rrbracket_w \\ \llbracket \mathbf{a} \rrbracket_w = V w \mathbf{a} & \llbracket A \rightarrow_+ B \rrbracket_w = \forall w' \in W. R w w' \rightarrow \llbracket A \rrbracket_{w'} \rightarrow \llbracket B \rrbracket_{w'} \\ & \llbracket A \rightarrow_- B \rrbracket_w = \exists w' \in W. R w' w \wedge \neg \llbracket A \rrbracket_{w'} \wedge \llbracket B \rrbracket_{w'} \end{array}$$

The interpretation of formulas really highlights the fact that implication is dual to coimplication. Monotonicity holds for this interpretation.

Lemma 5 (Monotonicity). Suppose (W, R, V) is a Kripke model, A is some DIL formula, and $w, w' \in W$. Then $R w w'$ and $\llbracket A \rrbracket_w$ imply $\llbracket A \rrbracket_{w'}$.

At this point we must set up the mathematical machinery which allows for the interpretation of sequents in a Kripke model. This will require the interpretation of graphs, and hence, nodes. We interpret nodes as worlds in the model using a function we call a node interpreter.

Definition 6. Suppose (W, R, V) is a Kripke model and S is a set of nodes of an abstract Kripke model G . Then a **node interpreter** on S is a function from S to W .

Now using the node interpreter we can interpret edges as statements about the reachability relation in the model. Thus, the interpretation of a graph is just the conjunction of the interpretation of its edges.

Definition 7. Suppose (W, R, V) is a Kripke model, G is an abstract Kripke model, and N is a node interpreter on the set of nodes of G . Then the interpretation of G in the Kripke model is defined by recursion on the structure of the graph as follows:

$$\begin{array}{ll} \llbracket \cdot \rrbracket_N & = \top \\ \llbracket n_1 \preceq_+ n_2, G \rrbracket_N & = R(N n_1)(N n_2) \wedge \llbracket G \rrbracket_N \\ \llbracket n_1 \preceq_- n_2, G \rrbracket_N & = R(N n_2)(N n_1) \wedge \llbracket G \rrbracket_N \end{array}$$

Now we can prove that if a particular reachability judgment holds, then the interpretation of the nodes are reachable in the model.

Lemma 8 (Reachability Interpretation). Suppose (W, R, V) is a Kripke model, and $\llbracket G \rrbracket_N$ for some abstract Kripke graph G . Then

- i. if $G \vdash n_1 \preceq_+ n_2$, then $R(N n_1)(N n_2)$, and
- ii. if $G \vdash n_1 \preceq_- n_2$, then $R(N n_2)(N n_1)$.

We now have everything we need to interpret abstract Kripke models. The final ingredient to the interpretation of sequents is the interpretation of contexts.

Definition 9. If F is some meta-logical formula, we define pF as follows:

$$+F = F \quad \text{and} \quad -F = \neg F.$$

Definition 10. Suppose (W, R, V) is a Kripke model, Γ is a context, and N is a node interpreter on the set of nodes in Γ . The interpretation of Γ in the Kripke model is defined by recursion on the structure of the context as follows:

$$\begin{array}{ll} \llbracket \cdot \rrbracket_N & = \top \\ \llbracket pA @ n, \Gamma \rrbracket_N & = p \llbracket A \rrbracket_{(N n)} \wedge \llbracket \Gamma \rrbracket_N \end{array}$$

Combining these interpretations results in the following definition of validity.

Definition 11. Suppose (W, R, V) is a Kripke model, Γ is a context, and N is a node interpreter on the set of nodes in Γ . The interpretation of sequents is defined as follows:

$$\llbracket G; \Gamma \vdash pA @ n \rrbracket_N = \text{if } \llbracket G \rrbracket_N \text{ and } \llbracket \Gamma \rrbracket_N, \text{ then } p\llbracket A \rrbracket_{(Nn)}.$$

Notice that in the definition of validity the graph G is interpreted as a set of constraints imposed on the set of Kripke models, thus reinforcing the fact that the graphs on sequents really are abstract Kripke models. Finally, using the previous definition of validity we can prove consistency.

Theorem 12 (Consistency). Suppose $G; \Gamma \vdash pA @ n$. Then for any Kripke model (W, R, V) and node interpreter N on $|G|$, $\llbracket G; \Gamma \vdash pA @ n \rrbracket_N$.

3.2 Completeness of DIL

In this section we prove that every derivable sequent in L can be translated to a derivable sequent of DIL. We will call a sequent in L a L-sequent and a sequent in DIL a DIL-sequent. Throughout this section we assume without loss of generality that all L-sequents have non-empty right-hand sides. That is, for every L-sequent, $\Gamma \vdash_G \Delta$, we assume that $\Delta \neq \cdot$. We do not lose generality because it is possible to prove that $\Gamma \vdash_G \cdot$ holds if and only if $\Gamma \vdash_G n : \perp$ for any node n (proof omitted).

Along the way, we will see admissibility of the analogues of the rules we mentioned in Section 3. The proof of consistency was with respect to DIL including the cut rule, but we prove completeness with respect to DIL where the general cut rule has been replaced with the following two inference rules, which can be seen as restricted instances of the cut rule:

$$\frac{pB @ n' \in (\Gamma, \bar{p}A @ n) \quad G; \Gamma, \bar{p}A @ n \vdash \bar{p}B @ n'}{G; \Gamma \vdash pA @ n} \text{ AXCUT}$$

$$\frac{\bar{p}B @ n' \in (\Gamma, \bar{p}A @ n) \quad G; \Gamma, \bar{p}A @ n \vdash pB @ n'}{G; \Gamma \vdash pA @ n} \text{ AXCUTBAR}$$

Note that we will use admissible rules as if they are inference rules of the logic throughout the sequel. These two rules are required for the following crucial lemma.

Lemma 13 (Left-to-Right). If $G; \Gamma_1, \bar{p}A @ n, \Gamma_2 \vdash \bar{p}'B @ n'$ is derivable, then so is $G; \Gamma_1, \Gamma_2, p'B @ n' \vdash pA @ n$.

We mentioned above that DIL avoids analogs of a number of rules from L. To be able to translate every derivable sequent of L to DIL, we must show admissibility of those rules in DIL.

Lemma 14 (Reflexivity). If $G, m \preceq_p m; \Gamma \vdash pA @ n$ is derivable, then so is $G; \Gamma \vdash pA @ n$.

Lemma 15 (Transitivity). If $G, n_1 \preceq_{p'} n_3; \Gamma \vdash pA @ n$ is derivable, $n_1 \preceq_{p'} n_2 \in G$ and $n_2 \preceq_{p'} n_3 \in G$, then $G; \Gamma \vdash pA @ n$ is derivable.

Lemma 16 (AndL). If $G; \Gamma, \bar{p}A @ n \vdash pB @ n$ is derivable, then $G; \Gamma \vdash p(A \wedge \bar{p}B) @ n$ is derivable.

Lemma 17 (MonoL). If $G; \Gamma, pA @ n_1, pA @ n_2, \Gamma' \vdash p'B @ n'$ is derivable and $n_1 \preceq_p n_2 \in G$, then $G; \Gamma, pA @ n_1, \Gamma' \vdash p'B @ n'$ is derivable.

Lemma 18 (MonoR). If $G; \Gamma, \bar{p}A @ n_1, \Gamma' \vdash pA @ n_2$ and $n_1 \preceq_p n_2 \in G$, then $G; \Gamma, \Gamma' \vdash pA @ n_2$ is derivable.

There are a number of other admissible rules that the results in this section depend on, all of which, can be found in the companion report [11].

We now have everything we need to prove that every derivable sequent of L can be translated to a derivable sequent in DIL. The proof technique we use here does not provide an algorithm taking a

sequent of L and yielding a derivable sequent in DIL. Such an algorithm would have to choose a particular conclusion in the L-sequent to be the active formula of the DIL-sequent, but this is quite difficult. Instead we show that all possible conclusions in the L-sequent can be chosen to be active and yield a derivable DIL-sequent.

Using the translation of formulas given in Definition 1 we can easily translate contexts. Right contexts Γ in L are translated to positive hypotheses, while left contexts, not including the formula chosen as the active formula, are translated into negative hypotheses. The following definition defines the translation of both types of contexts.

Definition 19. *We extend the translation of formulas to contexts Γ and Δ with respect to a polarity p as follows:*

$$\ulcorner \cdot \urcorner^p = \cdot.$$

$$\ulcorner n : A, \Gamma \urcorner^p = p \ulcorner A \urcorner @ n, \ulcorner \Gamma \urcorner^p$$

Abstract Kripke models are straightforward to translate.

Definition 20. *We define the translation of graphs G in L to graphs in DIL as follows:*

$$\ulcorner \cdot \urcorner = \cdot.$$

$$\ulcorner (n_1, n_2), G \urcorner = n_1 \preceq_+ n_2, \ulcorner G \urcorner$$

The previous definition implies the following result:

Lemma 21 (Reachability). *If $n_1 G n_2$, then $\ulcorner G \urcorner \vdash n_1 \preceq_+^* n_2$.*

The translation of a derivable L-sequent is a DIL-sequent which requires a particular formula as the active formula. We define such a translation in the following definition.

Definition 22. *An activation of a derivable L-sequent $\Gamma \vdash_G \Delta$ is a DIL-sequent $\ulcorner G \urcorner; \ulcorner \Gamma \urcorner^+, \ulcorner \Delta_1, \Delta_2 \urcorner^- \vdash + \ulcorner A \urcorner @ n$, where $\Delta = \Delta_1, n : A, \Delta_2$.*

Finally, the following theorem is the main result showing that any activation of a derivable L-sequent is derivable in DIL.

Theorem 23 (Containment of L in DIL). *If $\ulcorner G \urcorner; \Gamma' \vdash + A @ n$ is an activation of the derivable L-sequent $\Gamma \vdash_G \Delta$, then $\ulcorner G \urcorner; \Gamma' \vdash + A @ n$ is derivable.*

Corollary 24 (Completeness). *DIL is complete.*

Proof. Completeness of L is proved in [20], and by Theorem 23 we know that every derivable sequent of L is derivable in DIL. $\square$

4 Dualized Type Theory (DTT)

In this section we give DIL a term assignment yielding Dualized Type Theory (DTT). First, we introduce DTT, and give several examples illustrating how to program in DTT. Then we present the metatheory of DTT.

The syntax for DTT is defined in Figure 6. Polarities, types, and graphs are all the same as they were in DIL. Contexts differ only by the addition of labeling each hypothesis with a variable. Terms, denoted t , consist of introduction forms, together with cut terms $\vee x.t \bullet t'^2$. We denote variables as $x, y, z, \dots$. The

²In classical type theories the symbol μ usually denotes cut, but we have reserved that symbol – indexed by a polarity – to be used with inductive (positive polarity) and coinductive (negative polarity) types in future work.

(indices)	$d ::= 1 \mid 2$
(polarities)	$p ::= + \mid -$
(types)	$A, B, C ::= \langle p \rangle \mid A \rightarrow_p B \mid A \wedge_p B$
(terms)	$t ::= x \mid \mathbf{triv} \mid (t, t') \mid \mathbf{in}_d t \mid \lambda x. t \mid \langle t, t' \rangle \mid \nu x. t \bullet t'$
(canonical terms)	$c ::= x \mid \mathbf{triv} \mid (t, t') \mid \mathbf{in}_d t \mid \lambda x. t \mid \langle t, t' \rangle$
(graphs)	$G ::= \cdot \mid n \preceq_p n' \mid G, G'$
(contexts)	$\Gamma ::= \cdot \mid x : pA @ n \mid \Gamma, \Gamma'$

Figure 6: Syntax for DTT.

$$\begin{array}{c}
\frac{G \vdash n \preceq_p^* n'}{G; \Gamma, x : pA @ n, \Gamma' \vdash x : pA @ n'} \quad \text{AX} \qquad \frac{}{G; \Gamma \vdash \mathbf{triv} : p \langle p \rangle @ n} \quad \text{UNIT} \\
\\
\frac{G; \Gamma \vdash t_1 : pA @ n \quad G; \Gamma \vdash t_2 : pB @ n}{G; \Gamma \vdash (t_1, t_2) : p(A \wedge_p B) @ n} \quad \text{AND} \qquad \frac{G; \Gamma \vdash t : pA_d @ n}{G; \Gamma \vdash \mathbf{in}_d t : p(A_1 \wedge_p A_2) @ n} \quad \text{ANDBAR} \\
\\
\frac{n' \notin |G|, |\Gamma| \quad (G, n \preceq_p n'); \Gamma, x : pA @ n' \vdash t : pB @ n'}{G; \Gamma \vdash \lambda x. t : p(A \rightarrow_p B) @ n} \quad \text{IMP} \\
\\
\frac{G \vdash n \preceq_p^* n' \quad G; \Gamma \vdash t_1 : \bar{p}A @ n' \quad G; \Gamma \vdash t_2 : pB @ n'}{G; \Gamma \vdash \langle t_1, t_2 \rangle : p(A \rightarrow_{\bar{p}} B) @ n} \quad \text{IMPBAR} \qquad \frac{G; \Gamma, x : \bar{p}A @ n \vdash t_1 : +B @ n' \quad G; \Gamma, x : \bar{p}A @ n \vdash t_2 : -B @ n'}{G; \Gamma \vdash \nu x. t_1 \bullet t_2 : pA @ n} \quad \text{CUT}
\end{array}$$

Figure 7: Type-Assignment Rules for DTT.

term $\mathbf{triv}$ is the introduction form for units, (t, t') is the introduction form for pairs, similarly the terms $\mathbf{in}_1 t$ and $\mathbf{in}_2 t$ introduce disjunctions, $\lambda x. t$ introduces implication, and $\langle t, t' \rangle$ introduces coimplication. The type-assignment rules are defined in Figure 7, and result from a simple term assignment to the rules for DIL. Finally, the reduction rules for DTT are defined in Figure 8. The reduction rules should be considered rewrite rules that can be applied anywhere within a term. (The congruence rules for this are omitted.)

Programming in DTT is not functional programming as usual, so we now give several illustrative examples. The reader familiar with type theories based on sequent calculi will find the following very familiar. The encodings are very similar to that of Curien and Herbelin's $\tilde{\lambda}\mu\tilde{\mu}$ -calculus [9]. The locus of computation is the cut term, so naturally, function application is modeled using cuts. Suppose

$$\begin{array}{ll}
D_1 & \stackrel{\text{def}}{=} G; \Gamma \vdash \lambda x. t : +(A \rightarrow_+ B) @ n \\
D_2 & \stackrel{\text{def}}{=} G; \Gamma \vdash t' : +A @ n \\
\Gamma' & \stackrel{\text{def}}{=} \Gamma, y : -B @ n
\end{array}$$

Then we can construct the following typing derivation:

$$\begin{array}{c}
\frac{}{vz.\lambda x.t \bullet \langle t_1, t_2 \rangle \rightsquigarrow vz.[t_1/x]t \bullet t_2} \text{RIMP} \qquad \frac{}{vz.\langle t_1, t_2 \rangle \bullet \lambda x.t \rightsquigarrow vz.t_2 \bullet [t_1/x]t} \text{RIMBAR} \\
\\
\frac{}{vz.(t_1, t_2) \bullet \mathbf{in}_1 t \rightsquigarrow vz.t_1 \bullet t} \text{RAND1} \qquad \frac{}{vz.(t_1, t_2) \bullet \mathbf{in}_2 t \rightsquigarrow vz.t_2 \bullet t} \text{RAND2} \\
\\
\frac{}{vz.\mathbf{in}_1 t \bullet (t_1, t_2) \rightsquigarrow vz.t \bullet t_1} \text{RANDBAR1} \qquad \frac{}{vz.\mathbf{in}_2 t \bullet (t_1, t_2) \rightsquigarrow vz.t \bullet t_2} \text{RANDBAR2} \\
\\
\frac{x \notin \text{FV}(t)}{vx.t \bullet x \rightsquigarrow t} \text{RRET} \qquad \frac{}{vz.(vx.t_1 \bullet t_2) \bullet t \rightsquigarrow vz.[t/x]t_1 \bullet [t/x]t_2} \text{RBETAL} \\
\\
\frac{}{vz.c \bullet (vx.t_1 \bullet t_2) \rightsquigarrow vz.[c/x]t_1 \bullet [c/x]t_2} \text{RBETAR}
\end{array}$$

Figure 8: Reduction Rules for DTT.

$$\frac{D_2 \quad \frac{}{G; \Gamma' \vdash y : -B @ n} \text{AX} \quad \frac{}{G; \Gamma' \vdash \langle t', y \rangle : -(A \rightarrow_+ B) @ n} \text{IMBAR}}{G; \Gamma \vdash vz.\lambda x.t \bullet \langle t', y \rangle : +B @ n} \text{CUT}$$

Implication was indeed eliminated, yielding the conclusion.

There is some intuition one can use while thinking of this style of programming. In [15] Kimura and Tatsuta explain how we can think of positive variables as input ports, and negative variables as output ports. Clearly, these notions are dual. Then a cut of the form $vz.t \bullet t'$ can be intuitively understood as a device capable of routing information. We think of this term as first running the term t , and then plugging its value into the continuation t' . Thus, negative terms are continuations. Now consider the instance of the previous term $vz.t \bullet y$ where t is a positive term and y is a negative variable (an output port). This can be intuitively understood as after running t , route its value through the output port y . Now consider the instance $vz.t \bullet z$. This term can be understood as after running the term t , route its value through the output port z , but then capture this value as the return value. Thus, the cut term reroutes output ports into the actual return value of the cut.

There is one additional bit of intuition we can use when thinking about programming in DTT. We can think of cuts of the form $vz.(\lambda x_1 \cdots \lambda x_i.t) \bullet \langle t_1, \langle t_2, \dots \langle t_i, z \rangle \dots \rangle \rangle$ as an abstract machine, where $\lambda x_1 \cdots \lambda x_i.t$ is the functional part of the machine, and $\langle t_1, \langle t_2, \dots \langle t_i, z \rangle \dots \rangle \rangle$ is the stack of inputs the abstract machine will apply the function to ultimately routing the final result of the application through z , but rerouting this into the return value. This intuition is not new, but was first observed by Curien and Herbelin in [9]; see also [10].

Similarly to the eliminator for implication we can define the eliminator for disjunction in the form of the usual case analysis. Suppose $G; \Gamma \vdash t : +(A \wedge B) @ n$, $G; \Gamma, x : +A @ n \vdash t_1 : +C @ n$, and $G; \Gamma, x : +B @ n \vdash t_2 : +C @ n$ are all admissible. Then we can derive the usual eliminator for disjunction. Define $\text{case } t \text{ of } x.t_1, x.t_2 \stackrel{\text{def}}{=} vz_0.(vz_1.(vz_2.t \bullet (z_1, z_2)) \bullet (vx.t_2 \bullet z_0)) \bullet (vx.t_1 \bullet z_0)$. Then we have the following result.

Lemma 25. *The following rule is admissible:*

$$\begin{array}{c}
\frac{}{\Gamma, x : pA, \Gamma' \vdash_c x : pA} \text{CLASSAX} \qquad \frac{}{\Gamma \vdash_c \mathbf{triv} : p\langle p \rangle} \text{CLASSUNIT} \\
\\
\frac{\Gamma \vdash_c t_1 : pA \quad \Gamma \vdash_c t_2 : pB}{\Gamma \vdash_c (t_1, t_2) : p(A \wedge_p B)} \text{CLASSAND} \qquad \frac{\Gamma \vdash_c t : pA_d}{\Gamma \vdash_c \mathbf{in}_d t : p(A_1 \wedge_{\bar{p}} A_2)} \text{CLASSANDBAR} \\
\\
\frac{\Gamma, x : pA \vdash_c t : pB}{\Gamma \vdash_c \lambda x. t : p(A \rightarrow_p B)} \text{CLASSIMP} \qquad \frac{\Gamma \vdash_c t_1 : \bar{p}A \quad \Gamma \vdash_c t_2 : pB}{\Gamma \vdash_c \langle t_1, t_2 \rangle : p(A \rightarrow_{\bar{p}} B)} \text{CLASSIMPBAR} \\
\\
\frac{\Gamma, x : \bar{p}A \vdash_c t_1 : +B \quad \Gamma, x : \bar{p}A \vdash_c t_2 : -B}{\Gamma \vdash_c \mathbf{v} x. t_1 \bullet t_2 : pA} \text{CLASSCUT}
\end{array}$$

Figure 9: Classical typing of DTT terms

$$\frac{G; \Gamma, x : pA @ n \vdash t_1 : pC @ n \quad G; \Gamma, x : pB @ n \vdash t_2 : pC @ n \quad G; \Gamma \vdash t : p(A \wedge_{\bar{p}} B) @ n}{G; \Gamma \vdash \mathbf{case of } x. t_1, x. t_2 : pC @ n} \text{CASE}$$

Now consider the term $\mathbf{v} x. \mathbf{in}_1 (\mathbf{v} y. \mathbf{in}_2 \langle y, \mathbf{triv} \rangle \bullet x) \bullet x$. This term is the inhabitant of the type $A \wedge_{-} \sim A$, and its typing derivation follows from the derivation given in Section 3. We can see by looking at the syntax that the cuts involved are indeed on the axiom x , thus this term has no canonical form. In [8] Crolard shows that inhabitants such as these amount to a constructive coroutine. That is, it is a restricted form of a continuation.

We now consider several example reductions in DTT. In the following examples we underline non-top-level redexes. The first example simply α -converts the function $\lambda x. x$ into $\lambda z. z$ as follows:

$$\begin{array}{ccc}
\lambda z. \underline{\mathbf{v} y. \lambda x. x \bullet \langle z, y \rangle} & \xrightarrow{\text{(RIMP)}} & \lambda z. \underline{\mathbf{v} y. z \bullet y} \\
& \xrightarrow{\text{(RRET)}} & \lambda z. z
\end{array}$$

A more involved example is the application of the function $\lambda x. (\lambda y. y)$ to the arguments $\mathbf{triv}$ and $\mathbf{triv}$.

$$\begin{array}{ccc}
\mathbf{v} z. \lambda x. (\lambda y. y) \bullet \langle \mathbf{triv}, \langle \mathbf{triv}, z \rangle \rangle & \xrightarrow{\text{(RIMP)}} & \mathbf{v} z. \lambda y. y \bullet \langle \mathbf{triv}, z \rangle \\
& \xrightarrow{\text{(RIMP)}} & \mathbf{v} z. \mathbf{triv} \bullet z \\
& \xrightarrow{\text{(RRET)}} & \mathbf{triv}
\end{array}$$

5 Metatheory of DTT

We now present the basic metatheory of DTT, starting with type preservation (proof omitted).

Lemma 26 (Type Preservation). *If $G; \Gamma \vdash t : pA @ n$, and $t \rightsquigarrow t'$, then $G; \Gamma \vdash t' : pA @ n$.*

A more substantial property is strong normalization of reduction for typed terms. To prove this result, we will prove a stronger property, namely strong normalization for reduction of terms which are typable using the system of classical typing rules in Figure 9 [7]. This is justified by the following easy result (proof omitted), where $\lceil \Gamma \rceil$ just drops the world annotations from assumptions in Γ :

Theorem 27. *If $G; \Gamma \vdash t : pA @ n$, then $\lceil \Gamma \rceil \vdash_c t : pA$*

$$\begin{array}{ll}
t \in \llbracket A \rrbracket^+ & \Leftrightarrow \forall x \in \text{Var}. \forall t' \in \llbracket A \rrbracket^-. \forall x. t \cdot t' \in \mathbf{SN} \\
t \in \llbracket A \rrbracket^- & \Leftrightarrow \forall x \in \text{Var}. \forall t' \in \llbracket A \rrbracket^{+c}. \forall x. t' \cdot t \in \mathbf{SN} \\
t \in \llbracket \langle + \rangle \rrbracket^{+c} & \Leftrightarrow t \in \text{Var} \vee t \equiv \mathbf{triv} \\
t \in \llbracket \langle - \rangle \rrbracket^{+c} & \Leftrightarrow t \in \text{Var} \\
t \in \llbracket A \rightarrow_+ B \rrbracket^{+c} & \Leftrightarrow t \in \text{Var} \vee \exists x, t'. t \equiv \lambda x. t' \wedge \forall t'' \in \llbracket A \rrbracket^+. [t''/x]t' \in \llbracket B \rrbracket^+ \\
t \in \llbracket A \rightarrow_- B \rrbracket^{+c} & \Leftrightarrow t \in \text{Var} \vee \exists t_1 \in \llbracket A \rrbracket^-, t_2 \in \llbracket B \rrbracket^+. t \equiv \langle t_1, t_2 \rangle \\
t \in \llbracket A \wedge_+ B \rrbracket^{+c} & \Leftrightarrow t \in \text{Var} \vee \exists t_1 \in \llbracket A \rrbracket^+, t_2 \in \llbracket B \rrbracket^+. t \equiv (t_1, t_2) \\
t \in \llbracket A_1 \wedge_- A_2 \rrbracket^{+c} & \Leftrightarrow t \in \text{Var} \vee \exists d. \exists t' \in \llbracket A_d \rrbracket^+. t \equiv \mathbf{in}_d t'
\end{array}$$

Figure 10: Interpretations of types

Let $\mathbf{SN}$ be the set of terms which are strongly normalizing with respect to the reduction relation. Let Var be the set of term variables, and let us use x and y as metavariables for variables. We will prove strong normalization for classically typed terms using a version of Krivine's classical realizability [16]. We define three interpretations of types in Figure 10. The definition is by mutual induction, and can easily be seen to be well-founded, as the definition of $\llbracket A \rrbracket^+$ invokes the definition of $\llbracket A \rrbracket^-$ with the same type, which in turn invokes the definition of $\llbracket A \rrbracket^{+c}$ with the same type; and the definition of $\llbracket A \rrbracket^{+c}$ may invoke either of the other definitions at a strictly smaller type. The reader familiar with such proofs will also recognize the debt owed to Girard [13]. All missing proofs below may be found in the companion report [11].

Lemma 28 (Step interpretations). *If $t \in \llbracket A \rrbracket^+$ and $t \rightsquigarrow t'$, then $t' \in \llbracket A \rrbracket^+$; and similarly if $t \in \llbracket A \rrbracket^-$ or $t \in \llbracket A \rrbracket^{+c}$.*

Lemma 29 (SN interpretations). $\begin{array}{ll} 1. \llbracket A \rrbracket^+ \subseteq \mathbf{SN} & 3. \llbracket A \rrbracket^- \subseteq \mathbf{SN} \\ 2. \text{Vars} \subseteq \llbracket A \rrbracket^- & 4. \llbracket A \rrbracket^{+c} \subseteq \mathbf{SN} \end{array}$

Definition 30 (Interpretation of contexts). $\llbracket \Gamma \rrbracket$ is the set of substitutions σ such that for all $x : pA \in \Gamma$, $\sigma(x) \in \llbracket A \rrbracket^p$.

Lemma 31 (Canonical positive is positive). $\llbracket A \rrbracket^{+c} \subseteq \llbracket A \rrbracket^+$

Theorem 32 (Soundness). *If $\Gamma \vdash_c t : pA$ then for all $\sigma \in \llbracket \Gamma \rrbracket$, $\sigma t \in \llbracket A \rrbracket^p$.*

Proof. We consider one interesting case (see the companion report for all the cases). Define $\delta(t)$ to be the length of the longest reduction sequence from t to a normal form, for $t \in \mathbf{SN}$.

$$\frac{\Gamma, x : +A \vdash_c t_1 : +B \quad \Gamma, x : +A \vdash_c t_2 : -B}{\Gamma \vdash_c \forall x. t_1 \cdot t_2 : -A} \text{CLASSCUT}$$

It suffices to consider arbitrary $y \in \text{Vars}$ and $t' \in \llbracket A \rrbracket^{+c}$, and show $\forall y. t' \cdot (\forall x. \sigma t_1 \cdot \sigma t_2) \in \mathbf{SN}$. By the IH and part 2 of Lemma 29, we have $\sigma t_1 \in \llbracket B \rrbracket^+$ and $\sigma t_2 \in \llbracket B \rrbracket^-$, which implies $\sigma t_1 \in \mathbf{SN}$ and $\sigma t_2 \in \mathbf{SN}$ by Lemma 29 again. We proceed by inner induction on $\delta(t') + \delta(\sigma t_1) + \delta(\sigma t_2)$, using Lemma 28, to show that all one-step successors of $\forall y. t' \cdot (\forall x. \sigma t_1 \cdot \sigma t_2)$ are in $\mathbf{SN}$. If it is t' , σt_1 , or σt_2 which steps, then the result holds by inner IH. The only other reduction possible is by RBETAR, since t' cannot be a cut term by the definition of $\llbracket A \rrbracket^{+c}$. In this case, the IH gives us $[t'/x]\sigma t_1 \in \llbracket B \rrbracket^+$ and $[t'/x]\sigma t_2 \in \llbracket B \rrbracket^-$, and we then have $\forall y. [t'/x]\sigma t_1 \cdot [t'/x]\sigma t_2 \in \mathbf{SN}$ by the definition of $\llbracket B \rrbracket^+$. $\square$

Corollary 33 (Strong Normalization). *If $G; \Gamma \vdash t : pA @ n$, then $t \in \mathbf{SN}$.*

Proof. This follows easily by putting together Theorems 27 and 32, with Lemma 29. $\square$

Corollary 34 (Cut Elimination). *If $G; \Gamma \vdash t : pA @ n$, then there is normal t' with $t \rightsquigarrow^* t'$ and t' containing only cut terms of the form $\forall x. y \bullet t$ or $\forall x. t \bullet y$, for y a variable.*

Lemma 35 (Local Confluence). *The reduction relation of Figure 8 is locally confluent.*

Proof. We may view the reduction rules as higher-order pattern rewrite rules. It is easy to confirm that all critical pairs (e.g., between RBETAR and the rules RIMP, RIMPBAR, RAND1, RANDBAR1, RAND2, and RANDBAR2) are joinable. Local confluence then follows by the higher-order critical pair lemma [19]. $\square$

Theorem 36 (Confluence for Typable Terms). *The reduction relation restricted to terms typable in DTT is confluent.*

Proof. Suppose $G; \Gamma \vdash t : pA @ n$ for some G, Γ, p , and A . By Lemma 26, any reductions in the unrestricted reduction relation from t are also in the reduction relation restricted to typable terms. The result now follows from Newman's Lemma, using Lemma 35 and Theorem 33. $\square$

6 Conclusion

We have presented a new type theory for bi-intuitionistic logic. We began with a compact dualized formulation of this logic, Dualized Intuitionistic Logic (DIL), and showed soundness with respect to a standard Kripke semantics (in Agda), and completeness with respect to Pinto and Uustalu's system L. We then presented Dualized Type Theory (DTT), and showed type preservation, strong normalization, and confluence for typable terms. Future work includes further additions to DTT, for example with polymorphism and inductive types. It would also be interesting to obtain a Canonicity Theorem as in [24], identifying some set of types where closed normal forms are guaranteed to be canonical values (as canonicity fails in general in DIL/DTT, as in other bi-intuitionistic systems).

References

- [1] Andreas Abel, Brigitte Pientka, David Thibodeau & Anton Setzer (2013): *Copatterns: programming infinite structures by observations*. In: *Proceedings of the 40th annual ACM SIGPLAN-SIGACT symposium on Principles of Programming Languages (POPL)*, ACM, pp. 27–38.
- [2] G. Bellin (2004): *Natural Deduction and term assignment for co-Heyting algebras in Polarized Bi-Intuitionistic Logic*.
- [3] G. Bellin (2005): *A Term Assignment for Dual Intuitionistic Logic*. *LICS'05-IMLA'05*.
- [4] G. Bellin, M. Carrara, D. Chiffi & A. Menti (2014): *A pragmatic dialogic interpretation of bi-intuitionism*. *Submitted to Logic and Logical Philosophy*.
- [5] Gianluigi Bellin (2012): *Categorical Proof Theory of Co-Intuitionistic Linear Logic*. *Submitted to LOMECS*.
- [6] Corrado Biasi & Federico Aschieri (2008): *A Term Assignment for Polarized Bi-intuitionistic Logic and Its Strong Normalization*. *Fundam. Inf.* 84(2), pp. 185–205. Available at <http://dl.acm.org/citation.cfm?id=2365856.2365859>.
- [7] Tristan Crolard (2001): *Subtractive logic*. *Theor. Comput. Sci.* 254(1-2), pp. 151–185.
- [8] Tristan Crolard (2004): *A Formulae-as-Types Interpretation of Subtractive Logic*. *J. Log. and Comput.* 14(4), pp. 529–570.

- [9] P. Curien & H. Herbelin (2000): *The duality of computation*. In: *Proceedings of the fifth ACM SIGPLAN International Conference on Functional Programming (ICFP)*, ACM, pp. 233–243.
- [10] Pierre-Louis Curien (2002): *Abstract Machines, Control, and Sequents*. In Gilles Barthe, Peter Dybjer, Lus Pinto & Joo Saraiva, editors: *Applied Semantics, Lecture Notes in Computer Science 2395*, Springer Berlin Heidelberg, pp. 123–136.
- [11] H. Eades, A. Stump & R. McCleary (2014): *Dualized Type Theory*. Companion report, available at <http://metatheorem.org/wp-content/papers/CLC14-comp.pdf>.
- [12] Andrzej Filinski (1989): *Declarative continuations: An investigation of duality in programming language semantics*. In DavidH. Pitt, DavidE. Rydeheard, Peter Dybjer, AndrewM. Pitts & Axel Poign, editors: *Category Theory and Computer Science, Lecture Notes in Computer Science 389*, Springer Berlin Heidelberg, pp. 224–249, doi:10.1007/BFb0018355. Available at <http://dx.doi.org/10.1007/BFb0018355>.
- [13] J.-Y. Girard, Y. Lafont & P. Taylor (1990): *Proofs and Types*. Cambridge University Press.
- [14] Rajeev Goré, Linda Postniece & Alwen Tiu (2008): *Cut-elimination and proof-search for bi-intuitionistic logic using nested sequents*. In Carlos Areces & Robert Goldblatt, editors: *Advances in Modal Logic*, College Publications, pp. 43–66. Available at <http://www.aiml.net/volumes/volume7/Gore-Postniece-Tiu.pdf>.
- [15] Daisuke Kimura & Makoto Tatsuta (2009): *Dual Calculus with Inductive and Coinductive Types*. In Ralf Treinen, editor: *Rewriting Techniques and Applications (RTA)*, pp. 224–238.
- [16] Jean-Louis Krivine (2009): *Realizability in classical logic*. *Panoramas et synthèses* 27, pp. 197–229. Interactive models of computation and program behaviour. Société Mathématique de France.
- [17] Joachim Lambek (1993): *Substructural Logics*, chapter From Categorical Grammar to Bilinear Logic. 2, Oxford Science Publications.
- [18] Joachim Lambek (1995): *Cut Elimination for Classical Bilinear Logic*. *Fundam. Inform.* 22(1/2), pp. 53–67.
- [19] T. Nipkow (1991): *Higher-Order Critical Pairs*. *Proceedings of Sixth Annual IEEE Symposium on Logic in Computer Science*, pp. 342–349.
- [20] Lus Pinto & Tarmo Uustalu (2009): *Proof Search and Counter-Model Construction for Bi-intuitionistic Propositional Logic with Labelled Sequents*. In Martin Giese & Arild Waaler, editors: *Automated Reasoning with Analytic Tableaux and Related Methods, Lecture Notes in Computer Science 5607*, Springer Berlin Heidelberg, pp. 295–309.
- [21] Cecylia Rauszer (1974): *Semi-Boolean algebras and their applications to intuitionistic logic with dual operations*. *Fundamenta Mathematicae* 83, pp. 219–249.
- [22] Harold Schellinx (1991): *Some Syntactical Observations on Linear Logic*. *Journal of Logic and Computation* 1(4), pp. 537–559.
- [23] Peter Sewell, Francesco Zappa Nardelli, Scott Owens, Gilles Peskine, Thomas Ridge, Susmit Sarkar & Rok Strnisa (2010): *Ott: Effective tool support for the working semanticist*. In: *Journal of Functional Programming*, 20, pp. 71–122.
- [24] Aaron Stump (2014): *The Recursive Polarized Dual Calculus*. In: *Proceedings of the ACM SIGPLAN 2014 Workshop on Programming Languages Meets Program Verification, PLPV '14*, ACM, New York, NY, USA, pp. 3–14, doi:10.1145/2541568.2541575. Available at <http://doi.acm.org/10.1145/2541568.2541575>.
- [25] P. Wadler (2005): *Call-by-Value Is Dual to Call-by-Name – Reloaded*. In Jürgen Giesl, editor: *Rewriting Techniques and Applications (RTA)*, Lecture Notes in Computer Science, Springer, pp. 185–203.