Linear Contextual Modal Type Theory

Anders Schack-Nielsen
Carsten Schürmann

IT University Technical Report Series **TR-2011-TR-2011-151**

ISSN 1600–6100 **December 2011**

ISSN 1600–6100

ISBN 978-87-7949-250-9

Copies may be obtained by contacting:

IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S
Denmark

Telephone: +45 72 18 50 00
Telefax: +45 72 18 50 01
Web `www.itu.dk`

Linear Contextual Modal Type Theory*

Anders Schack-Nielsen and Carsten Schürmann

IT University of Copenhagen
Copenhagen, Denmark
anderssn|carsten@itu.dk

Abstract. When one implements a logical framework based on linear type theory, for example the Celf system [SNS08], one is immediately confronted with questions about their equational theory and how to deal with logic variables. In this paper, we propose linear contextual modal type theory that gives a mathematical account of the nature of logic variables. Our type theory is conservative over intuitionistic contextual modal type theory proposed by Nanevski, Pfenning, and Pientka. Our main contributions include a mechanically checked proof of soundness and a working implementation.

1 Introduction

Over recent years, linear logic has become increasingly popular as a logic for concurrency, stateful computation, and even security. So far, the idea of resource awareness has had far reaching consequences for the design and implementation of logics and logical frameworks; LLF [CP96], CLF [CPWW02] and even separation logic [Rey02] use ideas borrowed from linear logic at their cores. There are implementations of theorem provers, logic programming languages, and proof assistants that do implement linear logic, as for example, Lolli [HM94], and Celf [SNS08].

All of these implementations depend crucially on the choice of fragment of linear logic and the choice of logic variable. Logic variables stand, for example, for open subgoals in a derivation tree or holes in a term to be instantiated via unification. As a motivating example consider the unification problem

$$c \mathbin{\widehat{}} (F \mathbin{\widehat{}} x \mathbin{\widehat{}} y) = c \mathbin{\widehat{}} (F \mathbin{\widehat{}} y \mathbin{\widehat{}} x)$$

where we write F for the logic variable, x, y for the two resources that need to be consumed exactly once, c for constant symbols, and $\widehat{}$ for linear application. If $\widehat{}$ were intuitionistic application then any instantiation of F with a constant function is a solution. In the multiplicative fragment of linear logic, the problem has no solution because any instantiation of F will need to mention x and y exactly once on two different rigid paths. Thus the left hand side and the right hand side of the equation above will differ in these two places. If we were to work in linear logic with $\top$, the problem is also solvable by choosing the constant function $F = \widehat{\lambda} x.\, \widehat{\lambda} y.\, F' \mathbin{\widehat{}} \langle\rangle$ where we write $\widehat{\lambda}$ for linear functional abstraction and $\langle\rangle$ for the proof term of $\top$.

This little example illustrates the complex nature of logic variables in linear logic and their role in higher-order linear unification. In the $\top$-free case, every linear resource needs to be consumed by the same term on both sides of the equation. In the presence of $\top$, however, this is no longer the case. Therefore, without a clear understanding of the nature of logic variables from a mathematical point of view, it seems hopeless to try to devise and design algorithms for equality in linear logic.

In this paper we provide such an understanding by the means of linear contextual modal type theory that gives a precise mathematical meaning to logic variables for linear logic, building on ideas from contextual modal type theory by Nanevski, Pfenning, and Pientka [NPP08]. In their paper they work out a modal explanation of contextual validity, which accounts for the contexts that are usually associated with logic variables in the intuitionistic setting [DHKP96]. In this paper we define a contextual modal type theory for linear logic, which accounts for the definition of logic variables used for Cervesato's and Pfenning's linear pre-unification algorithm [CP97].

The underlying philosophical basis for this work is provided by Martin-Löf's view of logical truth in form of judgments and evidence in form of axioms and inference rules. Using his technique we construct the meaning of

* This work was in part supported by NABITT grant 2106-07-0019 of the Danish Strategic Research Council.

availability, which corresponds to the multiplicative fragment of linear logic with $\multimap$ as the main connective, *truth*, which corresponds to truth in intuitionistic logic with $\to$ as the main connective, and *contextual validity*, which corresponds to the logic of *logic variables* with $[\Gamma; \Delta]\!\!\to$ as the main connective (pronounced box arrow Γ, Δ). If we know that A $[\Gamma; \Delta]\!\!\to$ B is true, then a proof of B may mention a logic variable of type $[\Gamma; \Delta]A$, which may refer arbitrarily many times to assumptions in Γ and exactly once to each assumption in Δ. The justification of this construction can be found in Section 2.

Next we show the soundness of linear contextual modal logic in Section 3. To this end we give a sound and complete proof theoretical account of availability, truth, and contextual validity in form of a sequent calculus. Next we prove the admissibility of cut, which guarantees the existence of canonical proofs in linear contextual modal logic. The cut-elimination result of this section is formalized and machine-checked in Twelf [PS99]. The proofs can be downloaded at `http://www.twelf.org/~celf/download/lcml.tgz`.

In Section 4 we introduce proof terms [Bie94] via a Curry-Howard correspondence. In this section we define all basic operations on logic variables including abstraction, instantiation, and substitution application. The choice of canonical proofs induces the equational theory based on β reduction and η expansion. We show that every term is equivalent to a β-normal η-long form.

Linear contextual modal type theory presents the theoretical foundation of our implementation of the Celf proof assistant [SNS08].

2 Linear Contextual Modal Logic

The central idea in linear logic [Gir87] is that of a resource. Linear assumptions play the role of a fixed set of *available* resources that must be consumed (exactly once) in a derivation. Therefore, available resources form the philosophical foundation of linear contextual modal logic. The idea of linear logic as a resource oriented logic has rendered it attractive to many application areas. In Petri nets, tokens can be modeled as resources, in programming language theory it is state, and in security simply messages that are being created and consumed.

Traditionally one recovers intuitionistic logic from linear logic by singling out those resources that can be constructed without resource consumption. They can be used as often as desired, and thus, constructively speaking, they are *true*.

Finally, we introduce the judgment of *contextual validity*, which will ultimately serve as the logical justification of the existence of logic variables. We say that a judgment is valid if it is true in all contexts. But here we refine this idea one step further and refer to the validity of a proposition in a context $\Gamma; \Delta$, where Γ is a collection of true propositions, and Δ is a collection of available resources.

These three judgments can be defined by a set of inference rules and axioms following the ideas of the judgmental reconstruction of modal logic that goes back to Davies and Pfenning [DP01].

Linear Judgments In linear logic, resources are constructed from other resources, all of which are necessarily consumed during the process. We call judgments of this form *linear judgments*. If A is constructed using each linear resource among $A_1 \ldots A_n$ exactly once, we write

$$x_1 : A_1 \ \mathsf{avail}, \ldots, x_n : A_n \ \mathsf{avail} \vdash A \ \mathsf{avail}$$

The list of linear resources to the left of the $\vdash$ symbol enjoys among the three structural properties only exchange (and neither weakening nor contraction) and will be abbreviated in the remainder of this paper by the aforementioned Δ. In the remainder of the paper we refer to $x_i : A_i$ avail as a *linear assumption* and to A_i as a *resource*.

In our formulation of the rules that define introduction and elimination forms for the linear implication connective $\multimap$, we use the $\vdash$ symbol as a notational convenience for accounting all resources consumed by the derivation A avail.

$$\frac{}{x : A \ \mathsf{avail} \vdash A \ \mathsf{avail}} \ \mathsf{lhyp}_x$$

$$\frac{(\Delta, x : A \ \mathsf{avail}) \vdash B \ \mathsf{avail}}{\Delta \vdash A \multimap B \ \mathsf{avail}} \multimap \mathsf{I}^x \qquad \frac{\Delta_1 \vdash A \multimap B \ \mathsf{avail} \quad \Delta_2 \vdash A \ \mathsf{avail}}{(\Delta_1, \Delta_2) \vdash B \ \mathsf{avail}} \multimap \mathsf{E}$$

Theorem 1 (Principle of substitution). *If* $\Delta_1 \vdash A$ avail *and* $(\Delta_2, x : A \text{ avail}) \vdash B$ avail *then* $(\Delta_1, \Delta_2) \vdash B$ avail.

Proof. By induction on the second assumption.

Hypothetical Judgments We define the judgment A true to mean that A is available irrespective of which resources are available. This is only possible in the case that the derivation of A avail does not consume any resources. In compliance with the literature, we call the truth judgment a *hypothetical judgment* because it may rely on the assumptions that $x_1 : A_1$ true, $\ldots, x_n : A_n$ true.

$$x_1 : A_1 \text{ true}, \ldots, x_n : A_n \text{ true} \vdash A \text{ true}$$

In contrast to above, this list enjoys all structural properties, i.e. weakening, exchange, and contraction. In the following we will abbreviate the list of assumptions by Γ and refer to $x_i : A_i$ as an *intuitionistic assumption*.

Using $\Gamma; \Delta \vdash A$ avail as an appropriate notational device for linear judgments, we introduce truth in the following way.

$$\frac{\Gamma; \cdot \vdash A \text{ avail}}{\Gamma \vdash A \text{ true}}$$

Furthermore, we define the rules regarding the introduction and elimination of the intuitionistic implication connective $\to$ as follows.

$$\frac{}{(\Gamma, x : A \text{ true}); \cdot \vdash A \text{ avail}}\ \text{hyp}_x$$

$$\frac{(\Gamma, x : A \text{ true}); \Delta \vdash B \text{ avail}}{\Gamma; \Delta \vdash A \to B \text{ avail}} \to \text{I}^x \qquad \frac{\Gamma; \Delta \vdash A \to B \text{ avail} \quad \Gamma \vdash A \text{ true}}{\Gamma; \Delta \vdash B \text{ avail}} \to \text{E}$$

Theorem 2 (Principle of substitution). *If* $\Gamma; \cdot \vdash A$ avail *and* $(\Gamma, x : A \text{ true}); \Delta \vdash B$ avail *then* $\Gamma; \Delta \vdash B$ avail.

Proof. By induction on the second assumption.

Categorical Judgments We write A valid$[\Gamma; \Delta]$ if A is valid relative to a list of hypothetical judgments Γ and linear judgments Δ. This means that A is available whenever all hypotheses in Γ are true and all resources in Δ are available. Furthermore, we write

$$u_1 :: A_1 \text{ valid}[\Gamma_1; \Delta_1] \ldots u_n :: A_n \text{ valid}[\Gamma_n; \Delta_n] \vdash A \text{ valid}[\Gamma; \Delta]$$

for the judgments of relative validity. As above, the list of assumptions to the left of the $\vdash$ symbol enjoys the full set of structural properties, including exchange, weakening, and contraction. In the interest of clarity, we abbreviate this list by Ψ, and refer to a declaration A_i valid$[\Gamma_i; \Delta_i]$ as a *contextual modal hypothesis*. We introduce it in the following way.

$$\frac{\Psi; \Gamma; \Delta \vdash A \text{ avail}}{\Psi \vdash A \text{ valid}[\Gamma; \Delta]}$$

When using a contextual modal hypothesis in $\Psi; \Gamma; \Delta$ of type A valid$[\Gamma'; \Delta']$, we need to make sure that all hypotheses Γ' are true and all resources in Δ' can be provided. More formally, we express this fact by $\Psi; \Gamma; \Delta \vdash \Gamma'; \Delta'$, which is defined as follows: all hypotheses A in Γ' must satisfy $\Psi; \Gamma \vdash A$ true, and for some partition of $\Delta = \Delta_1, \ldots, \Delta_n$ the following holds: $\Psi; \Gamma; \Delta_i \vdash B_i$ avail for all resources B_i in Δ'. The presence of this new kind of contextual assumption gives rise to a new arrow, which is defined by the following introduction and elimination rule.

$$\frac{(\Psi, u :: A \text{ valid}[\Gamma'; \Delta']); \Gamma; \Delta \vdash \Gamma'; \Delta'}{(\Psi, u :: A \text{ valid}[\Gamma'; \Delta']); \Gamma; \Delta \vdash A \text{ avail}}\ \text{mhyp}_u$$

$$\frac{(\Psi, u :: A \text{ valid}[\Gamma'; \Delta']); \Gamma; \Delta \vdash B \text{ avail}}{\Psi; \Gamma; \Delta \vdash A\ [\Gamma'; \Delta'] \!\!\to B \text{ avail}} \Box\!\!\to \text{I}^u$$

$$\frac{\Psi; \Gamma; \Delta \vdash A\ [\Gamma'; \Delta'] \!\!\to B \text{ avail} \quad \Psi \vdash A \text{ valid}[\Gamma'; \Delta']}{\Psi; \Gamma; \Delta \vdash B \text{ avail}} \Box\!\!\to \text{E}$$

$$\frac{}{\Psi; (\Gamma, x\!:\!A\ \mathsf{true}); \cdot \vdash A\ \mathsf{avail}}\ \mathsf{hyp}_x \qquad \frac{}{\Psi; \Gamma; x\!:\!A\ \mathsf{avail} \vdash A\ \mathsf{avail}}\ \mathsf{lhyp}_x$$

$$\frac{(\Psi, u\!::\!A\ \mathsf{valid}[\Gamma'; \Delta']); \Gamma; \Delta \vdash \Gamma'; \Delta'}{(\Psi, u\!::\!A\ \mathsf{valid}[\Gamma'; \Delta']); \Gamma; \Delta \vdash A\ \mathsf{avail}}\ \mathsf{mhyp}_u$$

$$\frac{\Psi; (\Gamma, x\!:\!A\ \mathsf{true}); \Delta \vdash B\ \mathsf{avail}}{\Psi; \Gamma; \Delta \vdash A \rightarrow B\ \mathsf{avail}} \rightarrow \mathsf{I}^x \qquad \frac{\Psi; \Gamma; \Delta \vdash A \rightarrow B\ \mathsf{avail} \quad \Psi; \Gamma; \cdot \vdash A\ \mathsf{avail}}{\Psi; \Gamma; \Delta \vdash B\ \mathsf{avail}} \rightarrow \mathsf{E}$$

$$\frac{\Psi; \Gamma; (\Delta, x\!:\!A\ \mathsf{avail}) \vdash B\ \mathsf{avail}}{\Psi; \Gamma; \Delta \vdash A \multimap B\ \mathsf{avail}} \multimap \mathsf{I}^x \qquad \frac{\Psi; \Gamma; \Delta_1 \vdash A \multimap B\ \mathsf{avail} \quad \Psi; \Gamma; \Delta_2 \vdash A\ \mathsf{avail}}{\Psi; \Gamma; (\Delta_1, \Delta_2) \vdash B\ \mathsf{avail}} \multimap \mathsf{E}$$

$$\frac{(\Psi, u\!::\!A\ \mathsf{valid}[\Gamma'; \Delta']); \Gamma; \Delta \vdash B\ \mathsf{avail}}{\Psi; \Gamma; \Delta \vdash A\ [\Gamma'; \Delta']\!\!\mapsto B\ \mathsf{avail}}\ \Box\!\!\mapsto \mathsf{I}^u$$

$$\frac{\Psi; \Gamma; \Delta \vdash A\ [\Gamma'; \Delta']\!\!\mapsto B\ \mathsf{avail} \quad \Psi; \Gamma'; \Delta' \vdash A\ \mathsf{avail}}{\Psi; \Gamma; \Delta \vdash B\ \mathsf{avail}}\ \Box\!\!\mapsto \mathsf{E}$$

. . . .

$$\frac{}{\Psi; \Gamma; \cdot \vdash \cdot; \cdot} \qquad \frac{\Psi; \Gamma; \Delta \vdash \Gamma'; \cdot \quad \Psi; \Gamma; \cdot \vdash A\ \mathsf{avail}}{\Psi; \Gamma; \Delta \vdash (\Gamma', x\!:\!A); \cdot} \qquad \frac{\Psi; \Gamma; \Delta_1 \vdash \Gamma'; \Delta' \quad \Psi; \Gamma; \Delta_2 \vdash A\ \mathsf{avail}}{\Psi; \Gamma; (\Delta_1, \Delta_2) \vdash \Gamma'; (\Delta', x\!:\!A)}$$

Fig. 1. Natural Deduction Calculus for LCML

Theorem 3 (Principle of substitution). *If* $\Psi; \Gamma'; \Delta' \vdash A\ \mathsf{avail}$ *and* $(\Psi, u :: A\ \mathsf{valid}[\Gamma'; \Delta']); \Gamma; \Delta \vdash B\ \mathsf{avail}$ *then* $\Psi; \Gamma; \Delta \vdash B\ \mathsf{avail}$.

Proof. By induction on the second assumption.

Figure 1 summarizes the rules that define the meaning of the connectives purely in terms of availability. This can always be achieved because truth and contextual validity is defined by one single rule, which is always invertible. We call the resulting logic linear contextual modal logic (LCML).

Returning to our motivating example, contextual modal assumptions are logic variables.

Example 1 (Logic Variables). $u :: A\ \mathsf{valid}[\cdot; \cdot]$ is a logic variable that can only be instantiated by closed terms (that may neither refer to intuitionistic or linear assumptions). $v :: A\ \mathsf{valid}[\cdot; (u\!:\!B, v\!:\!C)]$ is a logic variable that must consume the two respective resources u and v exactly once. $w :: A\ \mathsf{valid}[u\!:\!B; v\!:\!C]$ is a logic variable that must consume the resource v, but may mention u arbitrarily many times.

3 Proof Theory

The rules defining the meaning of the connectives of linear contextual modal logic are sound in the sense that we understand the meaning of a proposition by examining only its constituents, in analogy to how global soundness is defined for contextual modal type theory [NPP08].

In pursuit of establishing soundness, we proceed by defining a sequent calculus for our logic for which we then show cut-elimination. Furthermore, we can argue that the sequent calculus is a sound and complete characterization of the rules introduced above.

The defining judgment of the sequent calculus for linear contextual modal logic is written as

$$\Psi; \Gamma; \Delta \Longrightarrow A.$$

$$\frac{}{\Psi; \Gamma; P \Longrightarrow P}\ \mathsf{init} \qquad \frac{\Psi; (\Gamma, A); (\Delta, A) \Longrightarrow C}{\Psi; (\Gamma, A); \Delta \Longrightarrow C}\ \mathsf{copy}$$

$$\frac{(\Psi, A[\Gamma'; \Delta']); \Gamma; \Delta_1 \Longrightarrow \Gamma'; \Delta' \quad (\Psi, A[\Gamma'; \Delta']); \Gamma; (\Delta_2, A) \Longrightarrow C}{(\Psi, A[\Gamma'; \Delta']); \Gamma; (\Delta_1, \Delta_2) \Longrightarrow C}\ \mathsf{reflect}$$

$$\frac{\Psi; (\Gamma, A); \Delta \Longrightarrow B}{\Psi; \Gamma; \Delta \Longrightarrow A \to B} \to \mathsf{R} \qquad \frac{\Psi; \Gamma; \cdot \Longrightarrow A \quad \Psi; \Gamma; (\Delta, B) \Longrightarrow C}{\Psi; \Gamma; (\Delta, A \to B) \Longrightarrow C} \to \mathsf{L}$$

$$\frac{\Psi; \Gamma; (\Delta, A) \Longrightarrow B}{\Psi; \Gamma; \Delta \Longrightarrow A \multimap B} \multimap \mathsf{R} \qquad \frac{\Psi; \Gamma; \Delta_1 \Longrightarrow A \quad \Psi; \Gamma; (\Delta_2, B) \Longrightarrow C}{\Psi; \Gamma; (\Delta_1, \Delta_2, A \multimap B) \Longrightarrow C} \multimap \mathsf{L}$$

$$\frac{(\Psi, A[\Gamma'; \Delta']); \Gamma; \Delta \Longrightarrow B}{\Psi; \Gamma; \Delta \Longrightarrow A\ [\Gamma'; \Delta'] \!\mapsto B} \square\!\!\to \mathsf{R} \qquad \frac{\Psi; \Gamma'; \Delta' \Longrightarrow A \quad \Psi; \Gamma; (\Delta, B) \Longrightarrow C}{\Psi; \Gamma; (\Delta, A\ [\Gamma'; \Delta'] \!\mapsto B) \Longrightarrow C} \square\!\!\to \mathsf{L}$$

....

$$\frac{}{\Psi; \Gamma; \cdot \Longrightarrow \cdot; \cdot}\ \mathsf{id} \qquad \frac{\Psi; \Gamma; \Delta \Longrightarrow \Gamma'; \cdot \quad \Psi; \Gamma; \cdot \Longrightarrow A}{\Psi; \Gamma; \Delta \Longrightarrow (\Gamma', A); \cdot}\ \mathsf{dot} \qquad \frac{\Psi; \Gamma; \Delta_1 \Longrightarrow \Gamma'; \Delta' \quad \Psi; \Gamma; \Delta_2 \Longrightarrow A}{\Psi; \Gamma; (\Delta_1, \Delta_2) \Longrightarrow \Gamma'; (\Delta', A)}\ \mathsf{ldot}$$

Fig. 2. Sequent Calculus of LCML

The judgment holds if A can be proved from contextual modal hypotheses Ψ, true hypotheses Γ, and available hypotheses Δ.

In analogy to the previous section, we generalize this judgment to lists of true and available hypotheses $\Gamma; \Delta$, for which we write formally

$$\Psi; \Gamma; \Delta \Longrightarrow \Gamma'; \Delta'.$$

The rules defining the sequent calculus are summarized in Figure 2. The init rule is defined for atomic propositions P only

$$\frac{}{\Psi; \Gamma; P \Longrightarrow P}\ \mathsf{init}.$$

The *identity principle* holds for the sequent calculus.

Theorem 4 (Identity principle). *For all propositions A, and contexts, Ψ, Γ, Γ', and Δ:*

1. $\Psi; \Gamma; A \Longrightarrow A$
2. $\Psi; \Gamma; \Delta \Longrightarrow \Gamma; \Delta$
3. $\Psi; (\Gamma, \Gamma'); \cdot \Longrightarrow \Gamma; \cdot$

Proof. By an easy mutual induction over the structure of A for 1., Δ for 2., and Γ for 3.

Lemma 1 (Weakening).

1. *If* $\Psi; \Gamma; \Delta \Longrightarrow C$ *then* $\Psi; \Gamma, A; \Delta \Longrightarrow C$.
2. *If* $\Psi; \Gamma; \Delta \Longrightarrow C$ *then* $\Psi, A[\Gamma'', \Delta'']; \Gamma; \Delta \Longrightarrow C$.
3. *If* $\Psi; \Gamma; \Delta \Longrightarrow \Gamma'; \Delta'$ *then* $\Psi; \Gamma, A; \Delta \Longrightarrow \Gamma'; \Delta'$.
4. *If* $\Psi; \Gamma; \Delta \Longrightarrow \Gamma'; \Delta'$ *then* $\Psi, A[\Gamma'', \Delta'']; \Gamma; \Delta \Longrightarrow \Gamma'; \Delta'$.

Proof. By a mutual structural induction on the given sequent derivations.

The *cut principle* also holds.

Theorem 5 (Cut Principle). *We must consider three different cuts, one for each of the three fragments.*

Linear cut: *If* $\Psi; \Gamma; \Delta_1 \Longrightarrow A$ *and* $\Psi; \Gamma; (\Delta_2, A) \Longrightarrow C$ *then* $\Psi; \Gamma; (\Delta_1, \Delta_2) \Longrightarrow C$.
Hypothetical cut: *If* $\Psi; \Gamma; \cdot \Longrightarrow A$ *and* $\Psi; (\Gamma, A); \Delta \Longrightarrow C$ *then* $\Psi; \Gamma; \Delta \Longrightarrow C$.
Categorical cut: *If* $\Psi; \Gamma'; \Delta' \Longrightarrow A$ *and* $(\Psi, A[\Gamma'; \Delta']); \Gamma; \Delta \Longrightarrow C$ *then* $\Psi; \Gamma; \Delta \Longrightarrow C$.

Proof. By structural induction lexicographically on the cut-formula A, the left, and the right derivation. The proof has been formalized in Twelf, and is accessible on the web at `http://www.twelf.org/~celf/download/lcml.tgz`.

Using the cut principle we get the usual correspondence between the natural deduction and the sequent calculus.

Theorem 6 (Soundness and Completeness).

1. $\Psi; \Gamma; \Delta \vdash A$ avail *if and only if* $\Psi; \Gamma; \Delta \Longrightarrow A$.
2. $\Psi; \Gamma; \Delta \vdash \Gamma'; \Delta'$ *if and only if* $\Psi; \Gamma; \Delta \Longrightarrow \Gamma'; \Delta'$.

Proof. The proofs are by mutual structural induction on the respective derivations. They have been formalized in Twelf, and are accessible on the web at `http://www.twelf.org/~celf/download/lcml.tgz`.

4 Linear Contextual Modal Type Theory

Constructive logics can be seen as type theories via the well-known Curry-Howard correspondence. Perhaps not very surprisingly, linear contextual modal logic gives rise to linear contextual modal type theory once augmented by proof terms.

Our version of linear contextual modal type theory (based on the rules introduced in Section 2) is perhaps less interesting for programming but it is the foundation for logical frameworks concerned with the representation of resource based deductive systems, such as LLF [CP96] and CLF [CPWW02], and permits us to study the very nature of logic variables.

The proof term assignment for linear contextual modal logic is based on the rules from Figure 1. For convenience, we drop the judgments of avail, true, and valid, and express the rules in a more traditional form as specified in Figure 3.

Proof Terms	t	$::= x \mid u[\sigma] \mid \lambda x{:}A.\, t \mid t_1\ t_2 \mid \widehat{\lambda} x{:}A.\, t \mid t_1 \mathbin{\widehat{}} t_2$ $\mid \lambda^{[\Gamma;\Delta]} u : A\,.\, t \mid t_1 \overset{\overline{\Gamma};\overline{\Delta}}{\Box} t_2$
Substitutions	σ	$::= \cdot \mid \sigma, t/x \mid \sigma, t\widehat{/}x$
Contexts	Γ, Δ	$::= \cdot \mid \Gamma, x : A$
Modal Contexts	Ψ	$::= \cdot \mid \Psi, u :: A[\Gamma; \Delta]$

We write $\overline{\Gamma}$ for the list of variable names in the context Γ.

We distinguish between variables x (intuitionistic or linear) and $u[\sigma]$ (contextual modal), where the contextual modal variables should be thought of as logic variables and σ as a delayed substitution. Every implication (linear, intuitionistic, and contextual modal) provides an abstraction and application term. Intuitionistic and linear application are standard. Therefore we comment only on the contextual modal application $t_1 \overset{\overline{\Gamma};\overline{\Delta}}{\Box} t_2$: all names in $\overline{\Gamma}$ and $\overline{\Delta}$ are binding occurrences of those intuitionistic and linear declarations that define the *world* the argument term t_2 lives in. The $\overline{\Gamma}$ and $\overline{\Delta}$ should be viewed as iterated abstractions of t_2 subject to α-conversion. Their order is therefore important.

4.1 Substitutions

Linear contextual type theory provides two notions of substitution application. One is for the simultaneous substitutions that witness derivations of the judgment : $\Psi; \Gamma; \Delta \vdash \sigma : \Gamma'; \Delta'$, and the other is for single point substitutions that instantiate contextual modal variables. Recall that contextual modal variables model logic variables.

$$\frac{}{\Psi;(\Gamma,x{:}A);\cdot\vdash x:A}\ \mathsf{hyp}_x \qquad \frac{}{\Psi;\Gamma;x{:}A\vdash x:A}\ \mathsf{lhyp}_x$$

$$\frac{(\Psi,u{::}A[\Gamma';\Delta']);\Gamma;\Delta\vdash\sigma:\Gamma';\Delta'}{(\Psi,u{::}A[\Gamma';\Delta']);\Gamma;\Delta\vdash u[\sigma]:A}\ \mathsf{mhyp}_u$$

$$\frac{\Psi;(\Gamma,x{:}A);\Delta\vdash t:B}{\Psi;\Gamma;\Delta\vdash\lambda x{:}A.t:A\to B}\to\mathsf{I}^x \qquad \frac{\Psi;\Gamma;\Delta\vdash t_1:A\to B\quad\Psi;\Gamma;\cdot\vdash t_2:A}{\Psi;\Gamma;\Delta\vdash t_1\ t_2:B}\to\mathsf{E}$$

$$\frac{\Psi;\Gamma;(\Delta,x{:}A)\vdash t:B}{\Psi;\Gamma;\Delta\vdash\widehat{\lambda}x:A.t:A\multimap B}\multimap\mathsf{I}^x \qquad \frac{\Psi;\Gamma;\Delta_1\vdash t_1:A\multimap B\quad\Psi;\Gamma;\Delta_2\vdash t_2:A}{\Psi;\Gamma;(\Delta_1,\Delta_2)\vdash t_1\,\hat{}\,t_2:B}\multimap\mathsf{E}$$

$$\frac{(\Psi,u{::}A[\Gamma';\Delta']);\Gamma;\Delta\vdash t:B}{\Psi;\Gamma;\Delta\vdash\lambda^{[\Gamma';\Delta']}u:A\,.\,t:A\,[\Gamma';\Delta']\!\mapsto B}\ \Box\!\!\mapsto\mathsf{I}^u$$

$$\frac{\Psi;\Gamma;\Delta\vdash t_1:A\,[\Gamma';\Delta']\!\mapsto B\quad\Psi;\Gamma';\Delta'\vdash t_2:A}{\Psi;\Gamma;\Delta\vdash t_1\ \overset{\overline{\Gamma'};\overline{\Delta'}}{\Box}\ t_2:B}\ \Box\!\!\mapsto\mathsf{E}$$

. . . .

$$\frac{}{\Psi;\Gamma;\cdot\vdash\cdot:\cdot;\cdot} \qquad \frac{\Psi;\Gamma;\Delta\vdash\sigma:\Gamma';\Delta'\quad\Psi;\Gamma;\cdot\vdash t:A}{\Psi;\Gamma;\Delta\vdash\sigma,t/x:(\Gamma',x{:}A);\Delta'} \qquad \frac{\Psi;\Gamma;\Delta_1\vdash\sigma:\Gamma';\Delta'\quad\Psi;\Gamma;\Delta_2\vdash t:A}{\Psi;\Gamma;(\Delta_1,\Delta_2)\vdash\sigma,\widehat{t/}x:\Gamma';(\Delta',x{:}A)}$$

Fig. 3. Type Theory LCMTT

The definition of simultaneous substitution is given in Figure 4. The definition is largely standard, and we comment only on the two marked equations. In (1) modal contextual variables never occur in the domain of σ. Therefore σ does not need to be extended by x/x as in the other two abstraction cases. In (2), the argument to a contextual modal term will always live in a world different from the domain of σ, and thus σ must not be applied to t_2.

Theorem 7 (Substitution).

1. *If* $\Psi;\Gamma;\Delta\vdash\sigma:\Gamma';\Delta'$ *and* $\Psi;\Gamma';\Delta'\vdash t:A$ *then* $\Psi;\Gamma;\Delta\vdash[\sigma](t):A$.
2. *If* $\Psi;\Gamma;\Delta\vdash\sigma:\Gamma';\Delta'$ *and* $\Psi;\Gamma';\Delta'\vdash\sigma':\Gamma'';\Delta''$ *then* $\Psi;\Gamma;\Delta\vdash[\sigma](\sigma'):\Gamma'';\Delta''$.

Proof. To account for the multiplicative splits of the linear context we must generalize the induction hypothesis. Let σ_s be a sub-substitution of σ in the sense that only some $\Delta'_s\subseteq\Delta'$ are substituted for and $\Psi;\Gamma;\Delta_s\vdash\sigma_s:\Gamma';\Delta'_s$ for the corresponding $\Delta_s\subseteq\Delta$. We prove $\Psi;\Gamma';\Delta'_s\vdash t:A$ implies $\Psi;\Gamma;\Delta_s\vdash[\sigma](t):A$ and $\Psi;\Gamma';\Delta'_s\vdash\sigma':\Gamma'';\Delta''$ implies $\Psi;\Gamma;\Delta_s\vdash[\sigma](\sigma'):\Gamma'';\Delta''$ by mutual structural induction on the derivations of t and σ'. The statement of the theorem can now be obtained by taking $\sigma_s=\sigma$ and correspondingly $\Delta_s=\Delta$ and $\Delta'_s=\Delta'$.

More interesting is the definition of substitution for contextual modal variables (or logic variables). Here, it is sufficient to define only a single point substitution, which corresponds to the instantiation of several instances of one particular logic variable. The presence of linearity does not lead to any surprises, and the definition of substitution is reminiscent to that described in [NPP08].

Definition 1 (Substitution for Contextual Modal Variables). *A substitution for a contextual modal variable is a single point substitution that substitutes a term* t'*, declared in world* $\Gamma;\Delta$ *for the contextual modal variable* u *in* t*. Application written in short as* $[\![(\overline{\Gamma};\overline{\Delta}).t'/u]\!](t)$*. The rules defining this judgment are depicted in Figure 5.*

Contextual modal substitution application traverses the entire structure of the term and substitutes t' into each occurrence of u. The only two interesting cases are equations (3) and (4). The former describes the actual substitution step: If we substitute t' for u in $u[\sigma]$, we follow the following steps. First, we replace all occurrences of the contextual

$$
\begin{array}{lll}
[\sigma_1, t/x, \sigma_2](x) = t & [\sigma](t_1 \mathbin{\widehat{}} t_2) & = [\sigma](t_1) \mathbin{\widehat{}} [\sigma](t_2) \\
[\sigma_1, t\widehat{/}x, \sigma_2](x) = t & [\sigma](\lambda^{[\Gamma;\Delta]} u{:}A\,.\,t) & = \lambda^{[\Gamma;\Delta]} u{:}A\,.\,[\sigma](t) \qquad (1) \\
[\sigma](u[\tau]) = u[[\sigma](\tau)] & [\sigma](t_1 \overset{\overline{\Gamma};\overline{\Delta}}{\square} t_2) & = [\sigma](t_1) \overset{\overline{\Gamma};\overline{\Delta}}{\square} t_2 \qquad (2) \\
[\sigma](\lambda x{:}A.\,t) = \lambda x{:}A.\,[\sigma, x/x](t) & [\sigma](\cdot) & = \cdot \\
[\sigma](t_1\ t_2) = [\sigma](t_1)\ [\sigma](t_2) & [\sigma](\sigma', t/x) & = [\sigma](\sigma'), [\sigma](t)/x \\
[\sigma](\widehat{\lambda} x{:}A.\,t) = \widehat{\lambda} x{:}A.\,[\sigma, x\widehat{/}x](t) & [\sigma](\sigma', t\widehat{/}x) & = [\sigma](\sigma'), [\sigma](t)\widehat{/}x
\end{array}
$$

Fig. 4. Simultaneous Substitution

$$
\begin{array}{lll}
[\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](x) & = x & \\
[\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](u[\sigma]) & = [[\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](\sigma)]t' \quad \text{where } \mathrm{dom}(\sigma) = (\overline{\Gamma'};\overline{\Delta'}) & (3) \\
[\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](v[\sigma]) & = v[[\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](\sigma)] \quad \text{where } u \neq v & (4) \\
[\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](\lambda x{:}A.\,t) & = \lambda x{:}A.\,[\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](t) & \\
[\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](t_1\ t_2) & = [\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](t_1)\ [\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](t_2) & \\
[\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](\widehat{\lambda} x{:}A.\,t) & = \widehat{\lambda} x{:}A.\,[\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](t) & \\
[\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](t_1 \mathbin{\widehat{}} t_2) & = [\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](t_1) \mathbin{\widehat{}} [\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](t_2) & \\
[\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](\lambda^{[\Gamma;\Delta]} v{:}A\,.\,t) & = \lambda^{[\Gamma;\Delta]} v{:}A\,.\,[\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](t) \quad \text{where } v \text{ does not occur in } t' & \\
[\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](t_1 \overset{\overline{\Gamma};\overline{\Delta}}{\square} t_2) & = [\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](t_1) \overset{\overline{\Gamma};\overline{\Delta}}{\square} [\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](t_2) & \\
[\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](\cdot) & = \cdot & \\
[\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](\sigma, t/x) & = [\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](\sigma), [\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](t)/x & \\
[\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](\sigma, t\widehat{/}x) & = [\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](\sigma), [\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](t)\widehat{/}x &
\end{array}
$$

Fig. 5. Substitution for Contextual Modal Variables

modal variable u in the delayed simultaneous substitution σ, written as $\sigma' = [\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](\sigma)$. However, before σ' can be applied to t', we must α-rename to make sure its domain equals the variable names in $\Gamma';\Delta'$ for which we write the side condition $\mathrm{dom}(\sigma') = (\overline{\Gamma'};\overline{\Delta'})$ or equivalently $\mathrm{dom}(\sigma) = (\overline{\Gamma'};\overline{\Delta'})$.

Theorem 8 (Contextual Modal Substitution).

1. *If* $\Psi;\Gamma';\Delta' \vdash t' : C$ *and* $(\Psi, u :: C[\Gamma';\Delta']);\Gamma;\Delta \vdash t : A$ *then* $\Psi;\Gamma;\Delta \vdash [\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](t) : A$.
2. *If* $\Psi;\Gamma';\Delta' \vdash t' : C$ *and* $(\Psi, u :: C[\Gamma';\Delta']);\Gamma;\Delta \vdash \sigma : \Gamma'';\Delta''$ *then* $\Psi;\Gamma;\Delta \vdash [\![(\overline{\Gamma'};\overline{\Delta'}).t'/u]\!](\sigma) : \Gamma'';\Delta''$.

Proof. By mutual structural induction on t in 1. and σ in 2.

4.2 Equational Theory

Based on the definition of these two notions of substitution application, every connective in linear contextual modal type theory gives rise to a β- and an η-rule, justifying our choice of definitional equality denoted by $\equiv$.

As a preliminary definition, we define the identity substitution on arbitrary contexts.

$$
\begin{array}{ll}
\mathsf{id}_{(\cdot;\cdot)} & = \cdot \\
\mathsf{id}_{((\overline{\Gamma},x);\cdot)} & = \mathsf{id}_{(\overline{\Gamma};\cdot)}, x/x \\
\mathsf{id}_{(\overline{\Gamma};(\overline{\Delta},x))} & = \mathsf{id}_{(\overline{\Gamma};\overline{\Delta})}, x\widehat{/}x
\end{array}
$$

Theorem 9 (Identity). *For all contexts Ψ, Γ, and Δ,*

$$\Psi; \Gamma; \Delta \vdash \mathsf{id}_{(\overline{\Gamma};\overline{\Delta})} : \Gamma; \Delta$$

Our choice of definitional equality $\equiv$ is defined as the symmetric, reflexive, and transitive closure of the following typed β-reduction and η-expansion rules defining $\Longrightarrow$. For the fragment motivated by linear and hypothetical judgments, the rules are standard.

$$\begin{aligned}
(\widehat{\lambda} x{:}A.\, t_1) \mathbin{\widehat{}} t_2 : B &\Longrightarrow [t_2/x]t_1 \\
t : A \multimap B &\Longrightarrow \widehat{\lambda} x : A.\, t \mathbin{\widehat{}} x \\
(\lambda x{:}A.\, t_1)\ t_2 : B &\Longrightarrow [t_2/x]t_1 \\
t : A \to B &\Longrightarrow \lambda x : A.\, t\ x
\end{aligned}$$

The fragment that corresponds to the contextual judgment is less standard but nevertheless intuitive.

$$\begin{aligned}
(\lambda^{[\Gamma;\Delta]} u{:}A\,.\,t_1)\ \overset{\overline{\Gamma};\overline{\Delta}}{\square}\ t_2 : B &\Longrightarrow [\![(\overline{\Gamma};\overline{\Delta}).t_2/u]\!]t_1 \\
t : A\ [\Gamma;\Delta]\!\!\rightarrow B &\Longrightarrow \lambda^{[\Gamma;\Delta]} u : A\,.\,t\ \overset{\overline{\Gamma};\overline{\Delta}}{\square}\ u[\mathsf{id}_{(\overline{\Gamma};\overline{\Delta})}]
\end{aligned}$$

Notice that the annotation $\overline{\Gamma};\overline{\Delta}$ of the contextual modal application binds variables in t_2 and can therefore always be α-renamed to match the annotation of the λ in the β-rule.

We omit the straightforward definition of the congruence rules, which also extends to substitutions. Every equivalence class is represented by a β-normal η-long form, which we also call canonical forms. The idea of equivalence and canonical form generalizes directly to simultaneous substitutions.

4.3 Meta Theory

We can reassure the reader that linear contextual modal type theory behaves as expected. First we show that reduction preserves types.

Theorem 10 (Subject Reduction). *If $\Psi; \Gamma; \Delta \vdash t : A$ and $t : A \Longrightarrow t'$ then $\Psi; \Gamma; \Delta \vdash t' : A$.*

Proof. To see that β and η rules preserve types, one returns to the interpretation of the typing derivation as a logic derivation. Each β rule is justified by the the argument of local soundness: Introducing a connective followed immediately by elimination is justified by the respective substitution principle. Each η rule is justified by the argument of local completeness. Any derivation of a non-atomic formula followed by an elimination, and reintroduction of the main connective ends in exactly the same derivation of the non-atomic formula we started with.

Subject reduction ensures that the equivalence classes induced by $\equiv$ are indeed well-defined sets of typed terms.

Theorem 11 (Canonical forms).

1. *If $\Psi; \Gamma; \Delta \vdash t : A$ then there exists a unique term t' in canonical form, such that $\Psi; \Gamma; \Delta \vdash t' : A$ and $t \equiv t'$.*
2. *If $\Psi; \Gamma; \Delta \vdash \sigma : \Gamma'; \Delta'$ then there exists a unique substitution σ' in canonical form, such that $\Psi; \Gamma; \Delta \vdash \sigma' : \Gamma'; \Delta'$ and $\sigma \equiv \sigma'$.*

Proof. This theorem follows from the fact that canonical forms correspond to cut-free derivations in the sequent calculus.

With canonical forms we can give a nice concise proof of raising and lowering.

Theorem 12 (Raising and Lowering).

1. *There is a bijection between proof terms t_1 with $(\Psi, u :: B[\Gamma'; (\Delta', x{:}A)]); \Gamma; \Delta \vdash t_1 : C$ and proof terms t_2 with $(\Psi, v :: (A \multimap B)[\Gamma'; \Delta']); \Gamma; \Delta \vdash t_2 : C$*
2. *There is a bijection between proof terms t_1 with $(\Psi, u :: B[(\Gamma', x{:}A); \Delta']); \Gamma; \Delta \vdash t_1 : C$ and proof terms t_2 with $(\Psi, v :: (A \to B)[\Gamma'; \Delta']); \Gamma; \Delta \vdash t_2 : C$*

Proof. In canonical forms the term $u[\sigma, t/x]$ is interchangeable with $v[\sigma] \mathbin{\widehat{}} t$ in 1. and $v[\sigma]\ t$ in 2.

5 Applications

The development of linear contextual modal type theory was driven by the necessity to understand higher-order unification in the concurrent logical framework CLF. In the interest of clarity we glossed over the issues with dependencies, which can be added without much trouble. The construction follows the steps of [NPP08].

Celf, as many other modern proof assistants, relies on unification, for example, for type reconstruction, logic programming, and proof search. Linear contextual modal type theory provides us with precise type invariants for unification problems, which we discuss in this section using a representation of the asynchronous π-calculus in Celf as a running example.

The π-calculus is defined in Celf as an algebra of expressions `expr` and channel names `chan`: we write `zero` for the empty process, $\texttt{par}\ P\ Q$ for parallel composition, $\texttt{new}\ (\lambda u : \texttt{chan}.\ P)$ for the process that creates a new channel name, $\texttt{out}\ U\ V$ for the process that writes a message V on channel U, and finally $\texttt{cin}\ U\ (\lambda m : \texttt{chan}.\ P)$ for the process that reads message m from channel U. As running example, we will use

$$\texttt{new}\ (\lambda u : \texttt{chan}.\, \texttt{par}\ (\texttt{new}\ (\lambda m.\, \texttt{out}\ u\ m))\ (\texttt{cin}\ u\ (\lambda x : \texttt{chan}.\, \texttt{zero}))), \tag{5}$$

a process that creates and transmits a channel name m from the left process to the right.

The operational semantics builds on the following idea. We use the linear context to contain thunks of the processes still to be executed (as linear hypotheses of type $\texttt{proc}\ P$) and messages that are currently available "on the network" (as linear hypotheses of type $\texttt{msg}\ U\ V$). As they are represented by linear resources, they must be consumed before the computation can terminate. Simulating a reduction sequence in the π-calculus corresponds therefore to constructing an object of the desired type in linear contextual modal type theory, which we call `run`.

In Celf, one may specify the reduction semantics of the π-calculus as follows.

$$\begin{aligned}
&\texttt{p_done} : \texttt{run}.\\
&\texttt{p_zero} : \texttt{run} \multimap \texttt{proc}\ \texttt{zero} \multimap \texttt{run}.\\
&\texttt{p_par} \ : \Pi P : \texttt{expr}.\, \Pi Q : \texttt{expr}.\\
&\qquad (\texttt{proc}\ P \multimap \texttt{proc}\ Q \multimap \texttt{run}) \multimap \texttt{proc}\ (\texttt{par}\ P\ Q) \multimap \texttt{run}.\\
&\texttt{p_new} \ : \Pi P : \texttt{chan} \to \texttt{expr}.\\
&\qquad (\Pi u : \texttt{chan}.\, \texttt{proc}\ (P\ u) \multimap \texttt{run}) \multimap \texttt{proc}\ (\texttt{new}\ P) \multimap \texttt{run}.\\
&\texttt{p_out} \ : \Pi U : \texttt{chan}.\, \Pi V : \texttt{chan}.\\
&\qquad (\texttt{msg}\ U\ V \multimap \texttt{run}) \multimap \texttt{proc}\ (\texttt{out}\ U\ V) \multimap \texttt{run}.\\
&\texttt{p_cin} \ : \Pi P : \texttt{chan} \to \texttt{expr}.\, \Pi U : \texttt{chan}.\, \Pi V : \texttt{chan}.\\
&\qquad (\texttt{proc}\ (P\ V) \multimap \texttt{run}) \multimap \texttt{msg}\ U\ V \multimap \texttt{proc}\ (\texttt{cin}\ U\ P) \multimap \texttt{run}.
\end{aligned}$$

There are other possibilities, some take advantage of the concurrency monad, but since we did not discuss it in this paper, we restrict ourselves to the LLF fragment of CLF to illustrate how logic variables are used. In linear contextual modal type theory, the query from above is then formulated as the following judgment

$$u :: (\texttt{proc}\ (5) \multimap \texttt{run})[\cdot;\cdot];\cdot;\cdot \vdash u[\mathsf{id};\mathsf{id}] : \texttt{proc}\ (5) \multimap \texttt{run}$$

The semantics of such a query is to search for an instantiation (categorical cut) of u. The first step of the search procedure is to η-expand and apply lowering (Theorem 12), which instantiates u by

$$\widehat{\lambda} p_0 : \texttt{proc}\ (5).\, v[\mathsf{id}; p_0/p_0]/u,$$

which (after one right focusing step) leads to

$$v :: \texttt{run}[\cdot; p_0 : \texttt{proc}\ (5)];\cdot; p_0 : \texttt{proc}\ (5) \vdash v[\mathsf{id}; p_0/p_0] : \texttt{run}.$$

Notice how u above is declared with empty contexts but through lowering implicitly specifies the contexts of v. In general Celf uses lowering in this way to give a nice clean interface for the specification of meta-variable contexts.

We show one more step. As the linear context is no longer empty, we cannot apply the rule p_done. Instead, we use p_new. After several steps, backchaining results in the modal contextual substitution

$$\texttt{p\_new}\ (\lambda u : \texttt{chan}.\ (6))\hat{}(\lambda u : \texttt{chan}.\ \widehat{\lambda} p_1 : \texttt{proc}\ (6).\ w[u/u; p_1/p_1])\hat{}\ p_0/v.$$

with the following abbreviation

$$\texttt{par}\ (\texttt{new}\ (\lambda m.\ \texttt{out}\ u\ m))\ (\texttt{cin}\ u\ (\lambda x : \texttt{chan}.\ \texttt{zero})) \tag{6}$$

and a new proof state with the meta-variable w:

$$w :: \texttt{run}[u : \texttt{chan}; \texttt{prog}\ (6)]; u : \texttt{chan}; p_1 : \texttt{prog}\ (6) \vdash w[u/u; p_1/p_1] : \texttt{run}.$$

After six more steps, we can discharge all meta-variables, and arrive at a "solved state". For increased readability, we omit the arguments that are usually called implicit (because they can be easily inferred).

$$\cdot\ \cdot; \cdot \vdash \widehat{\lambda} p_0.\ \texttt{p\_new}\hat{}(\lambda u.\ \widehat{\lambda} p_1.\ \texttt{p\_par}\hat{}(\widehat{\lambda} p_2.\ \widehat{\lambda} p_3.\ \texttt{p\_new}\hat{}(\lambda u_1.\ \widehat{\lambda} p_4.\ \texttt{p\_out}\hat{}(\widehat{\lambda} p_5.\ \texttt{p\_cin}\hat{}(\widehat{\lambda} p_6.\ \texttt{p\_zero}\hat{}\ \texttt{p\_done}\hat{}\ p_6)\hat{}\ p_5\hat{}\ p_3)\hat{}\ p_4)\hat{}\ p_2)\hat{}\ p_1)\hat{}\ p_0$$

This concludes the example.

6 Conclusion

In this paper we present linear contextual modal type theory as a mathematical explanation for logic variables in the presence of linear types. We prove the existence of canonical forms by means of cut-elimination, whose proof has been formalized and verified in the proof assistant Twelf.

Linear contextual modal type theory provides us with a deep understanding of the interaction between linear type theory and logic variables that are prevalent in many systems based on linear logic, for example, theorem provers and logic programming languages. It can easily be extended to dependent types in the style of LLF because the dependent Π only binds intuitionistic variables, thus making linearity and dependent types orthogonal extensions. The dependently typed version forms the theoretical foundation for our implementation of the concurrent logical framework CLF [CPWW02] in the Celf system [SNS08].

References

[Bie94] G. Bierman. *On Intuitionistic Linear Logic*. PhD thesis, University of Cambridge, 1994.

[CP96] Iliano Cervesato and Frank Pfenning. A linear logical framework. In E. Clarke, editor, *Proceedings of the Eleventh Annual Symposium on Logic in Computer Science*, pages 264–275, New Brunswick, New Jersey, July 1996. IEEE Computer Society Press.

[CP97] Iliano Cervesato and Frank Pfenning. Linear higher-order pre-unification. In Glynn Winskel, editor, *Proceedings of the Twelfth Annual Sumposium on Logic in Computer Science (LICS'97)*, pages 422–433, Warsaw, Poland, June 1997. IEEE Computer Society Press.

[CPWW02] Iliano Cervesato, Frank Pfenning, David Walker, and Kevin Watkins. A concurrent logical framework I: Judgments and properties. Technical Report CMU-CS-02-101, Carnegie Mellon University. Department of Computer Science, 2002.

[DHKP96] Gilles Dowek, Thérèse Hardin, Claude Kirchner, and Frank Pfenning. Unification via explicit substitutions: The case of higher-order patterns. In M. Maher, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pages 259–273, Bonn, Germany, September 1996. MIT Press.

[DP01] Rowan Davies and Frank Pfenning. A modal analysis of staged computation. *Journal of the ACM*, 48(3):555–604, 2001.

[Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[HM94] Joshua Hodas and Dale Miller. Logic programming in a fragment of intuitionistic linear logic. *Information and Computation*, 110(2):327–365, 1994.

[NPP08] Aleksandar Nanevski, Frank Pfenning, and Brigitte Pientka. Contextual modal type theory. *Transactions on Computational Logic*, 9(3):Article 23, 1–49, June 2008.

[PS99] Frank Pfenning and Carsten Schürmann. System description: Twelf — a meta-logical framework for deductive systems. In H. Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction (CADE-16)*, pages 202–206, Trento, Italy, July 1999. Springer-Verlag LNAI 1632.

[Rey02] John C. Reynolds. Separation logic: A logic for shared mutable data structures. In *Proceedings of the IEEE Symposium on Logic in Computer Science*, pages 55–74. IEEE Computer Society, 2002.

[SNS08] Anders Schack-Nielsen and Carsten Schürmann. System description: Celf – a logical framework for deductive and concurrent systems. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *International Joint Conference on Automated Reasoning (IJCAR)*, pages 320–331, Sydney, Australia, 2008.