Affine Control Logic

Chuck Liang
Hofstra University
Hempstead, NY
December 27, 2014

Abstract

This paper extends our previous work in combining classical logic with intuitionistic logic [LM13a, LM13b] to also include affine linear logic (linear logic with weakening), resulting in a unified system that we call Affine Control Logic. The goal is to allow affine-linear extensions of the Curry-Howard isomorphism to be combined with the well known capabilities of classical λµ calculus. To this end linear logic is adjusted so that contraction is not enabled by the exponential ? but by a restricted form of Peirce’s formula. This formula also has a dual, which replaces the role of ! in cut elimination. Classical fragments of proofs are better isolated from non-classical fragments, thus allowing different logics to coexist without a collapse. We define a phase space semantics for this logic and a sequent calculus that is proved to be sound and complete. A natural deduction system with a Curry-Howard interpretation is then defined. We demonstrate this system by showing how it can improve the formulation of exception handling in programming languages.

Contents

1 Introduction 2
2 Syntax and Polarization 3
3 Semantics 3
4 A Single Conclusion Sequent Calculus 7
  4.1 Fragments of ACL ................................................................. 10
  4.2 Adding Intuitionistic Disjunction ............................................ 10
5 Cut Elimination 11
  5.1 Consequences of Cut Elimination ............................................ 14
6 Soundness and Completeness 15
7 A Curry-Howard Interpretation 18
  7.1 Capturing Delimited Continuations ........................................... 22
  7.2 Improving the Handling of Exceptions ...................................... 23
  7.3 Concurrent Programming with fork and yield ................................ 24
8 Multiple Conclusion Alternative 26
  8.1 More Aggressive Curry-Howard .............................................. 27
9 Conclusion 27
1 Introduction

Linear logic offers many enticing connections to computation. Since its inception, various systems have been designed to give computational interpretations of linear logic. Some of these systems extend the Curry-Howard isomorphism to linear lambda calculi (e.g., [BBdPH93, Bar96]). These systems are based on fragments, and sometimes variations of linear logic, including affine linear logic (e.g., [CLR05, MA11]). They have been applied to, for example, the study of side effects and concurrency. However, linear logic is not the only path to extending typed lambda calculi. Since the work of Griffin [Gri90], Parigot [Par92] and many others, we have known that the Curry-Howard isomorphism also extends to classical logic. Contraction on the right-hand side is interpreted as saving a copy of the current continuation, while weakening can be seen as discarding the current continuation and jumping to a saved one. Such interpretations give logical meaning to control operators such as call/cc, as well as provide formalizations of programming features such as exception handling. Unfortunately there has been little effort to combine the two approaches, even though such a combination could lead to valuable benefits (see [MZ10]). There is good reason for this lack of intersection. Linear logic embeds both classical and intuitionistic logics, but it is limited in its ability to mix them. For example, the interpretation of intuitionistic implication as $\forall A \rightarrow B$ is a crucial component of linear logic. However, this interpretation is not compatible with the fragment that interprets classical logic. Consider $?(\forall A \rightarrow B) \oplus C$ (equivalently $?(\forall A \rightarrow B)\neg\neg C$): here we are attempting to write an intuitionistic implication as a subformula of a classical disjunction. The strength of intuitionistic implication is compromised: it may be possible to use the assumption $A$ to prove $C$: the intuitionistic meaning of $\forall A \rightarrow B$ would not survive such a mixture. Other unified systems based on linear logic, including LU [Gir93] and our own LKU [LM11], contain similar problems.

The computational power of $\lambda \mu$ calculus comes from allowing contractions on the right-hand side, which are enabled in linear logic only in formulas that are equivalent to some $\forall A$. Once we place a $?$ before a formula we lose the ability to impose invariants on proofs: contractions on that formula may appear anywhere in a proof. In other words, the placement of a $?$ represents a static approach to controlling contractions. A more dynamic approach would allow contractions only when certain conditions are encountered in a proof. Contractions should be controlled by the proof, not just the formula. For example, consider Peirce’s formula and its proof in $\lambda \mu$ calculus (i.e., the call/cc operator):

$$\lambda x.\mu d.[d](x \lambda y.\mu e.[d]y) : ((P \rightarrow Q) \rightarrow P) \rightarrow P$$

Here, $\mu d.[d] \ldots$ indicates contraction on $P$ while $\mu e.[d] \ldots$ indicates weakening on $Q$ and jumping to the copy of $P$. To emulate this proof in linear logic we would need a $?$ before $Q$ as well as one before at least the head occurrence of $P$. We cannot guarantee that the linear logic proof will only contract in the right place. Why emulate a proof in linear logic if there is little left that’s still linear?

Our goal is to isolate classical reasoning to fragments of proofs, thereby allowing them to coexist with non-classical fragments without collapsing one into the other. In our previous work [LM13b] we introduced Polarized Control Logic (PCL), which unified classical and intuitionistic logics. With this logic we were able to see that, in order to prove Peirce’s formula, it is necessary to interpret the innermost implication $P \rightarrow Q$ as a classical one (equivalent to $\neg P \lor Q$). However, the other instances of $\rightarrow$ can stay intuitionistic. Retaining intuitionistic implication, when it is not necessary to be classical, certainly has desirable consequences, including the ability to enforce scope. Using PCL we were able to demonstrate, for example, how to enforce where in a program an exception can be thrown. We would like to be able refine such abilities further: a procedure should throw an exception at most once.

In this paper we modify PCL to obtain a new propositional logic that unifies intuitionistic logic, classical logic, and the essential elements of affine linear logic. We refer to this system as Affine Control Logic (ACL). The intent here is to be able to extend the Curry-Howard isomorphism to include $\lambda \mu$-style controls while also allowing non-classical constructs to retain their meaning and expected proof-theoretic properties. Many uses of linear logic are also valid in affine logic, and some are enhanced by it. Semantically, the models of affine logic are closer to those of intuitionistic logic, and this will allow us to adopt our previous efforts in combining classical logic with intuitionistic logic [LM13a, LM13b].
Creating an alternative to linear logic is no easy task. Linear logic generalizes the principles of Gentzen in allowing cut elimination in a setting where *some but not all* formulas are subject to contraction. This is a central role of the ?/! duality, and we will need to find a new approach.

2 Syntax and Polarization

We focus on propositional logic in this presentation. Formulas of ACL are freely composed from connectives ∧, ∨, →, _ and ⊗, and constants 1, 0 and ⊥. The symbol → represents affine implication while → will have the meaning of intuitionistic implication. The constant ⊤ is equivalent to 1 in affine logic. The exponentials ? and ! will be replaced by the ability to mix ⊗ and → with intuitionistic and classical formulas: this will result in stronger invariants than the unrestricted use of ! and ?. Despite using the same symbol, the constant ⊥ in ACL is entirely different from its counterpart in linear logic. The new role of ⊥ is the most significant distinction between ACL and affine linear logic.

We use the term “polarization,” but to avoid confusion with other common uses of the term we call our polarities “red” and “green” as opposed to positive and negative.

Atomic formulas are arbitrarily colored red or green. Polarization extends to all formulas as follows:

- ⊥ is green; 0 and 1 are red.
- A ∧ B is green if both A and B are green, otherwise, it is red.
- A ∨ B is green if A is green or B is green, otherwise, it is red.
- A → B is green if B is green, otherwise, it is red.
- A _ B is green if B is green, otherwise, it is red.
- A ⊗ B is always red.

Since there are two implications and two constants for false, a green ⊥ and a red 0, there are four forms of negation in ACL. We define abbreviations for them as follows:

\[-A = A → ⊥ \quad \neg A = A → ⊥ \quad \sim A = A → 0 \quad \dashv A = A → 0\]

This paper is primarily concerned with −A, which is in fact logically equivalent to ¬A.

We use the letter E for an arbitrary green formula and e for either a green atom or ⊥. Unlike the positive/negative polarization, red and green are not “duals” of each other. For example, if E is green then −E is still green: “−” is not an involutive negation. It is possible for red and green formulas to be logically equivalent. Conceptually, green means classical and red means arbitrary: classical or non-classical. The polarization of ∧ and ∨ is similar to the positive/negative polarization of LC [Gir91], but these are the only cases that intersect. The polarities of LC are consistent with those of focused proofs, which are also the subjects of some of our own work. Although the proof systems of ACL use a small element of focusing, that is not the purpose of the red/green polarization. It is possible to explain positive/negative polarization purely syntactically, in terms of the invertibility of inference rules and the role that they play in controlling cut elimination. In contrast, the red and green polarities represent two levels of soundness. The role they play in unifying logics is difficult to explain and justify without a semantics of validity.

3 Semantics

The semantic tradition of linear logic emphasize the understanding of proofs, not formulas. Traditional model theory is sometimes disparaged because it is concerned primarily with truth and consistency, but not with proofs. This type of semantics, however, is exactly what we need in order to explain the difference between the red and green polarities, for they define two levels of provability. The green ⊥ defines a stronger
The interpretation (valuation) of all formulas is then defined as follows: to facts, with the following conditions:

Define a **Phased Frame** to be a structure \( \langle W, \preceq, r, \cdot \rangle \) where \( \preceq \) is a partial ordering relation on the set of possible worlds (or phases) \( W \). This structure also forms a commutative monoid with operation \( \cdot \) and unit \( r \in W \). We write \( ab \) for \( a \cdot b \). Given two sets of worlds \( A \) and \( B \), \( AB = \{ xy : x \in A, y \in B \} \). It always holds that \( ab = ba \) and \( (ab)c = a(bc) \). We further require the following property:

- **a \preceq b if and only if ac = b for some c.**

By inference, it always holds that \( a \preceq ab \). Also by inference, the unit \( r \) is the least element of \( W \) since \( r \preceq ru = u \) for all \( u \in W \). We shall refer to \( r \) as the root. The following properties are also easily inferred:

- If \( a \preceq b \) then \( ac \preceq bc \).
- If \( a \preceq b \) and \( bb = b \), then \( ab = b \).

It is also worth noting that finite phased structures must contain a top world \( t \) with the property that \( tt = t \).

Our phased models are closer to those of Okada [Oka02] than to Girard’s original [Gir87]. The principal difference between our models and those of linear logic is twofold. First, the facts of the space (subsets of \( W \) that can interpret formulas) are upwardly closed sets. A set \( S \) is upwardly closed if \( x \in S \) and \( x \preceq y \) implies \( y \in S \). This corresponds to the monotonicity property of intuitionistic Kripke models. However, unlike intuitionistic logic, not all upwardly closed sets are necessarily facts. The second, and most important difference is that, in phase semantics for linear logic \( \bot \) is represented by any arbitrary set, whereas here it is fixed to be \( W \setminus \{r\} \), the upwardly closed set that consists of all worlds above the root. The two sets \( W \) (1) and \( W \setminus \{r\} \) (\( \bot \)) form an embedded, two-element boolean algebra with nothing in between them.

Formally, let an **Ordered Phase Space** be a structure of the form \( (W, \cdot, r, \preceq, D) \), where \( W, \preceq, r \) and \( \cdot \) satisfy the requirements of a phased frame. \( D \) is a set of upwardly closed subsets of \( W \) called facts that is furthermore required to satisfy the following properties:

1. \( D \) contains \( W \) and \( W \setminus \{r\} \), both of which are upwardly closed (\( \preceq \) is a partial order, not just a preorder).
2. For any subsets \( A \) and \( B \) of \( W \), if \( B \in D \), the set \( \{ x \in W : \text{ for all } y \in A, xy \in B \} \) is also in \( D \).

   This set is upwardly closed because if \( x \preceq x' \) then \( x' = xx \) and \( xzy \in B \) hence \( xzy \in B \) since \( B \) is upwardly closed.

   We can call this set the pseudo-pseudocomplement of \( A \) relative to \( B \).
3. \( D \) must be closed under the following closure operator on subsets of \( W \) for any subset \( S \): \( cl(S) = \bigcap \{ V \in D : S \subseteq V \} \).

   Upward closure is preserved by arbitrary intersections.

   It holds that \( S \subseteq cl(S) \) and if \( S \) is already a fact in \( D \) then \( S = cl(S) \). It also holds that \( cl(cl(S)) = S \), \( cl(S)V \subseteq cl(SV) \) and if \( S \subseteq V \) then \( cl(S) \subseteq cl(V) \).

   We will not distinguish between formulas \( A \) and their interpretation in phase space \( A^p \) except when there is possibility for confusion.

   Given \( S \subseteq W \), let \( I(S) = \{ u \in S : uu = u \} \). These are the worlds that admit contraction. \( I(W) \) is never empty since \( rr = r \). A phase model on an ordered phase space is defined by a mapping from atomic formulas to facts, with the following conditions:

   - Red atoms are mapped to arbitrary facts (elements of \( D \)).
   - Green atoms are mapped to either \( W \setminus \{r\} \) or to \( W \), i.e., to either \( \bot^p \) or \( 1^p \).

The interpretation (valuation) of all formulas is then defined as follows:

- \( 1 \) (\( 1^p \)) is represented by \( W = cl(\{r\}) \). There is no need for both 1 and \( \top \) in affine logic.
- \( \bot \) is represented by \( W \setminus \{r\} \)
- 0 is \( cl(\emptyset) = \bigcap D \), the smallest possible fact (\( \emptyset \) is the empty set).
ACL is restricted to those connectives that do not require the closure operator \( cl \) right introduction rules in our sequent calculus. If all upwardly closed sets are facts, or if non-invertible stoup a fact.

Consistency can only be guaranteed at the root, which has the property completeness would be lost. The cases that require the closed sets remains upwardly closed. It would be simpler to allow all upwardly closed sets to be facts, but just on the green formula but on all formulas at that point in the proof.

Lemma 1 For every green formula \( E \), \( E^p \neq 1^p \) if and only if \( E^p = \bot^p \).

It should now be clear why \( A \otimes B \) is always red: we cannot guarantee that it will always valuate to 1 or \( \bot \) even if \( A \) “and” \( B \) are green. The constant 1 can be designated red or green: it makes little difference in this case (1 is red in PCL as well, so we wish to be consistent).

An important consequence of interpreting \( \bot \) as \( W \setminus \{ r \} \) is that \( A \vee \neg A \) is valid (without a \( ? \)). Note that showing \( r \in A \rightarrow B \) is equivalent to showing that \( A \subseteq B \). Thus if \( r \notin A^p \) then \( A^p \subseteq \bot^p \) and therefore \( r \in \neg A^p \). Thus \( r \in A^p \cup \neg A^p \subseteq cl(A^p \cup \neg A^p) \).

An even more important consequence is that Peirce’s formula in the form \(( (P \rightarrow E) \rightarrow P ) \rightarrow P \) is valid as long as \( E \) is green. \( P \) can be arbitrary (the occurrence of \( \rightarrow \) is stronger than \( \rightarrow \) in that position). If \( r \in P \) \( (r \in P^p) \) the result is obvious since then \(( P \rightarrow E ) \rightarrow P \subseteq P = W \) because \( P \) is upwardly closed. If \( r \notin P \), then \( r \in (P \rightarrow E) \) since \( \bot \subseteq E \) (all green formulas valuate to \( \bot \) or 1). Then, since \( rr = r \), we also have \((P \rightarrow E) \rightarrow P \subseteq P \).

The syntactic consequences of the validity of this version of Peirce’s formula are profound. It means that, upon encountering a green formula as the current or stoup formula in a proof, contraction is enabled, not just on the green formula but on all formulas at that point in the proof.

The closure operator is not needed in all the cases of \( A^p \). In the cases of \( \otimes, \vee \) and 0, the sets defined are already upwardly closed even without applying the closure operator \( cl \). For example, \( AB \) is upwardly closed if either \( A \) or \( B \) is a fact: if \( xy \in AB \) with \( x \in A \) and \( y \in B \), and \( xy \leq z \), then \( xyc = z \) for some \( c \) with \( x \in A \) and \( yc \in B \) because \( y \leq yc \) and \( B \) is upwardly closed. Similarly, the union of two upwardly closed sets remains upwardly closed. It would be simpler to allow all upwardly closed sets to be facts, but completeness would be lost. The cases that require the \( cl \) operator above correspond to the connectives with non-invertible right introduction rules in our sequent calculus. If all upwardly closed sets are facts, or if ACL is restricted to those connectives that do not require the closure operator \( cl \), then these phase models are perhaps better seen as Kripke models: \( u \in A^p \) can be read as “\( u \models A \).”

The constant 0 is not necessarily interpreted by the empty set, which is to be expected in phase semantics. The completeness proofs of such semantics typically define a set of multisets of formulas and multiset union as the monoid operation. This means that we cannot guarantee that these multisets will be consistent (does not derive 0), because the union of two consistent multisets may become inconsistent. In the phase semantics of linear logic, 0 is interpreted by \( W^\bot \). However, our “\( \bot \)” has an entirely different meaning than \( \bot \) in linear logic. Consistency can only be guaranteed at the root, which has the property \( rr = r \), (i.e., it can be a set as opposed to a multiset). In models with a non-empty \( 0^p \), the empty set, which is upwardly closed, is not a fact.

It easily holds that \( 0^p \subseteq A^p \) for all formulas \( A \). The largest possible \( 0^p \) is \( \bot^p \) and the smallest possible \( \bot^p \) is the empty set, but these cases only occur if \( D \) is just a two-element boolean algebra. In terms of Kripke style semantics, the fact that \( 0^p \) may not be empty means that there will be possible worlds that “force” 0.

---

1Under this interpretation, \( u \models A \rightarrow B \) holds if and only if \( uv = v \). The argument uses the properties noted above: if \( u \models A \rightarrow B \), and if \( v \models A \) for \( v \geq u \), then with the assumption that \( uv = u \) it follows that \( uv \leq uv = v \leq vv \), and so \( v \models B \).
Such kinds of Kripke models are not unknown [Vel76, ILH10]. Furthermore, because the root world cannot be in \(0^p\), there is still no model for 0 (or for \(\bot\)).

For another example of reasoning with this semantics, one can check that \((A \otimes B) \rightarrow A\) is valid, confirming the admissibility of weakening as follows. We need to show that \(r \in (A \otimes B) \rightarrow A\). This means showing that \((A \otimes B)^p \subseteq A^p\). By \(A^p\) is upwardly closed and \(x \preceq xy\), so \(A^p B^p \subseteq A^p\). Thus \((A \otimes B)^p = cl(A^p B^p) \subseteq cl(A^p)\). But \(cl(A^p) = A^p\) since \(A^p\) is a fact. The semantics also determine the validity of the following examples

- \(A \rightarrow (A \otimes A)\) is not valid, but \(A \rightarrow (A \otimes A)\) is valid (because of the \(vv = v\) assumption). This shows that linearity is present and distinguishable.

- \((A \rightarrow B) \rightarrow (A \rightarrow B)\) is valid. This is dereliction. The converse is not valid unless \(B\) is green. \(A \otimes B \rightarrow A \land B\) is also valid (because of weakening).

- \([A \otimes (A \rightarrow B)] \rightarrow B\) is not valid, but \([A \otimes (A \rightarrow B)] \rightarrow B\) is valid. This is consistent with the linear logic translation that \(A \rightarrow B\) is \(!A \rightarrow B\). However, \(!((A \otimes B) \rightarrow A)\) holds if weakening is always available.

- \((A \land B) \rightarrow (A \otimes B)\) is not valid but \((A \land B) \rightarrow (A \otimes B)\) is valid.

- \((A \rightarrow B) \rightarrow (\neg A \lor B)\) is valid, but the converse \((\neg A \lor B) \rightarrow (A \rightarrow B)\) is only valid if \(B\) is green. \(A \rightarrow E\) and \(A \rightarrow E\) are equivalent when \(E\) is green.

- In addition to Peirce formula in the form \((\neg A \rightarrow A) \rightarrow A\), the following counterpart or “dual” is also valid and forms an important component of ACL: \((A \rightarrow \neg A) \rightarrow \neg A\). The generalized form, \((A \rightarrow A \rightarrow E) \rightarrow A \rightarrow E\), is also valid. When observed in a purely classical or intuitionistic context, the provability of this formula appears meaningless. However, we should note that it is (intuitionistically) provable precisely because of contraction on the left. When linear/affine connectives are added to the mix, this formula implies the admissibility of the following inference:

\[
\frac{A \otimes A \vdash \bot}{A \vdash \bot}
\]

Here, \(\bot\) can also be replaced by any green formula. The validity of a green formula \(E\) is entirely determined by the root world \(r\), which has the property \(rr = r\). Just as Peirce’s formula enables contraction on the right-hand side, this counterpart enables contraction on the left, which otherwise would not be subject to contraction. Lessons taken from the proof theory of Gentzen and Girard allow us to anticipate this kind of duality when cut elimination is mixed with contraction. The duality between Peirce’s formula and its counterpart replaces the duality between ? and !.

- Several other important properties, mostly inherited from PCL [LM13b], should also be noted. These include the fact that none of the negations \(\neg, \sim, \neg, \text{ and } \sim\) are involutive. In particular, \(!A \rightarrow A\) is not valid: our \(\bot\) is not the same as the \(\bot\) of linear logic. We do have that \(\neg \neg E \rightarrow E\) and \(\neg \neg E \rightarrow E\) are valid if \(E\) is green. It also holds as an admissible rule that if \(!A\) is valid, then \(A\) is valid. Additionally, the De Morgan law \(\neg(A \land B) \rightarrow \neg A \lor \neg B\) is valid: the others cases do not require \(\bot\) to be green.

The following model, with three distinct worlds \(r, q\) and \(qq\), verifies several of the examples above:

\[
\begin{array}{c}
qq \in a, b, c \\
q \in a, c \\
r
\end{array}
\]

Here it is assumed that \(qq = qqq = qqqq\). All upwardly closed sets in this model are facts. This means that \(cl(S) = S\) for all upwardly closed \(S\): this is an intuitionistic Kripke frame, but with \(q \neq qq\). If \(u \in Q^p\) we will say that “\(u\) forces \(Q\).” The interpretation of the atoms \(a, b, c\) are that \(a^p = c^p = \{q, qq\}\) and \(b^p = \{qq\}\).
In other words, $q$ forces $a$, $c$, and $qq$ forces $a$, $b$, $c$. For example, $r$ forces $a \rightarrow b$ since the only world above $r$ that has the property $uu = u$ is $qq$. But $r \notin a \rightarrow b$ because $q \in a$ but $rq = q \notin b$. Another example: $q \in a \land c$ but $q \notin a \otimes c$ because $qq \neq q$. The same model also shows that $\neg$ and $\land$ are not involutive negations in ACL: let $d$ be a red atom that is not forced at any of the worlds. Then all worlds above $r$ forces $\neg\neg d$ and $\neg d$ because they force $\bot$ ($\sqcup^p = \{q, qq\}$), but they do not force $d$, and thus $r \notin \neg\neg d \rightarrow d$ and $r \notin \neg d \rightarrow d$.

The same model also plays the part of an intuitionistic Kripke model and shows that $b \lor \neg\neg b$ is not valid, and that $\neg\neg b \rightarrow b$ is not valid (regardless of the polarity of $b$), since $r \in \neg\neg b$ but $r \notin b$.

Our semantics preserve the advantage of Kripke semantics in the existence of small but effective counter-models. However, it should be noted that the monoid's closure property also diverges from what is typically expected in Kripke semantics. The countermodel for $\neg a \lor \neg\neg a$ requires a top world that "forces" 0:

$$
\begin{array}{c}
\downarrow \\
v \in a \\
ung \in 0, a \\
r
\end{array}
$$

In this model $0^p = \{uv\}$ and $a^p = \{v, uv\}$. It can be assumed that $I(W) = W$. The empty set is not a fact since $0^p$ must be the smallest fact. In the intuitionistic Kripke countermodel the top world is not needed, but it is rather unavoidable if the frame is a monoid.

It is important to recognize that the meaning of affine and intuitionistic (red) formulas do not necessarily collapse when mixed with classical (green) formulas. As an example of this property, the green formula $E \lor \neg E$ is still not valid because the subformula $\neg E = E \rightarrow 0$ is an intuitionistic implication. The root world is the only classically consistent world (i.e., consistent with respect to $\bot$). The validity of red formulas and subformulas are thus determined by more than just the root. Another example is that $E \rightarrow E \otimes E$ is not valid even with the green $E$.

### 4 A Single Conclusion Sequent Calculus

Various choices can be made in designing a proof system for ACL. The Kripke-like semantics suggests a system similar to the Beth-Fitting intuitionistic tableaux [Fit69], which is often written as a multiple-conclusion version of intuitionistic sequent calculus. This is the approach we used in PCL because, when converted to a natural deduction form, it offered more opportunities to assign computational meaning to proofs. We in fact showed how the proof system of PCL naturally suggested a form of dynamic scoping for continuation variables, and devised an abstract machine to realize this interpretation. While these options are also available for ACL, we choose here to begin with a simpler, single-conclusion sequent calculus. While multiple conclusions offer more flexibility, a single conclusion version offers a few more invariants that will be useful in proving cut elimination.

The sequent calculus $LAC$ is found in Figure 1. Here, $\Delta$ is a multiset but $A$, $\Gamma$ does not preclude the possibility that $A \in \Gamma$. Weakening and contraction in $\Gamma$, and weakening in $\Delta$, hold as admissible rules. The semantic interpretation of a sequent $\Gamma, \Delta \vdash A$ is as for the formula $\Gamma^\land \rightarrow (\Delta^\otimes \rightarrow A)$, where $\Gamma^\land$ is the $\land$-conjunction over formulas in $\Gamma$ and $\Delta^\otimes$ is the $\otimes$-conjunction over formulas in $\Delta$. An empty $\Gamma$ or $\Delta$ means 1. Elements $[\Theta : A]$ are treated as any other formula in $\Gamma$, and has the same meaning as $\neg (\Theta^\otimes \rightarrow A)$. The special notation is used so that this formula can only be principal as part of $Unlock$. $Unlock$ is a focused version of the $\rightarrow L$ rule combined with the $\bot L$ and $\otimes L$ rules. (see Corollary 5 of Section 6 for further explanation). Intuitively, from a computational perspective, one can regard $[\Theta : A]$ as a form of closure: $Lock$ saves not only a copy of the current continuation but also a part of its operating environment. The continuation is no longer stateless.

Although $Lock$ can be applied at any point in a proof, the effect of contraction is only available when $Unlock$ can be applied. Thus if a formula has no green subformulas, it can only have a non-classical proof.

We often refer to the singleton formula on the right-hand side as the stoup. The stoup is never empty. A formula $A$ is provable if $; \vdash A$ is provable.
### The Unified Sequent Calculus LAC

Figure 1: The Unified Sequent Calculus LAC. 

Many linear logic proof systems use dual contexts on the left (and sometimes right) hand side in sequents. If one examined the part of LAC that only unifies intuitionistic logic with affine-linear logic, then this sequent calculus is similar to that of Lolli [HM94], Forum [Mil96], as well as several other systems including LU [Gir93], DILL [Bar96], and Andreoli’s focusing sequent calculus [And92], where the practice likely originated.

The following are sample proofs, of a version of the excluded middle, \( A \lor \neg A \) (\( A \lor \neg A \rightarrow \bot \)), and of a version of the double-negation axiom, \( \neg \neg A \rightarrow A \) (\( \neg \neg A \rightarrow (\neg A \rightarrow \bot) \)).

<table>
<thead>
<tr>
<th>Rule</th>
<th>Left</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \vdash A )</td>
<td>( \vdash A, \bot \rightarrow A )</td>
<td>( \vdash A )</td>
</tr>
<tr>
<td>( \vdash A \lor \neg A )</td>
<td>( \vdash \neg A \lor A )</td>
<td>( \vdash A \lor \neg A )</td>
</tr>
<tr>
<td>( \vdash \neg A \lor A )</td>
<td>( \vdash A \lor \neg A )</td>
<td>( \vdash A \lor \neg A )</td>
</tr>
<tr>
<td>( \vdash A )</td>
<td>( \vdash A \lor \neg A )</td>
<td>( \vdash A \lor \neg A )</td>
</tr>
<tr>
<td>( \vdash A \lor \neg A )</td>
<td>( \vdash A \lor \neg A )</td>
<td>( \vdash A \lor \neg A )</td>
</tr>
<tr>
<td>( \vdash A \lor \neg A )</td>
<td>( \vdash A \lor \neg A )</td>
<td>( \vdash A \lor \neg A )</td>
</tr>
<tr>
<td>( \vdash A \lor \neg A )</td>
<td>( \vdash A \lor \neg A )</td>
<td>( \vdash A \lor \neg A )</td>
</tr>
<tr>
<td>( \vdash A \lor \neg A )</td>
<td>( \vdash A \lor \neg A )</td>
<td>( \vdash A \lor \neg A )</td>
</tr>
</tbody>
</table>

The proofs will fail if \( \bot \) was relaced with a red formula, such as 0 (\( A \lor \neg A \) remains unprovable). On the other hand, if 0 was replaced with \( \bot \) in the proof of \( \neg \neg A \rightarrow A \), then that proof will also fail, unless \( A \) is green (\( \bot \rightarrow A \) holds only for green \( A \)). None of the negations of ACL are involutive without conditions, but the negations can be mixed to give the desired computational effect (i.e., the \( C \) control operator). A slight adjustment to the proof of \( \neg \neg A \rightarrow A \) also proves a version of Peirce’s formula, \( \neg A \rightarrow A \rightarrow A \): replace 0\( L \) with an \( Id \) rule.

Given the semantic validity of the Peirce-like formula \( (\neg A \rightarrow A) \rightarrow A \), it would be valid to design structural rules of the following forms:

\[
\begin{align*}
\frac{\vdash A}{\vdash A \lor \neg A} & \quad \text{Id} \\
\frac{\vdash A \lor \neg A; A \vdash \bot}{\vdash \bot} & \quad \text{Unlock} \\
\frac{\vdash A \lor \neg A; \neg A \vdash A}{\vdash \bot} & \quad \text{Unlock} \\
\frac{\vdash A \lor \neg A; \neg A \vdash A}{\vdash \bot} & \quad \text{Unlock}
\end{align*}
\]

Here, the Unlock rule can be seen as just a special case of \( \rightarrow L \), since \( \bot \rightarrow e \) is valid for any green \( e \). Indeed these simplified rules are enough for the examples above. However, they are not enough for cut-elimination.
The most crucial case of cut-reduction is permutation of cut above a contraction. In particular, consider:

\[
\begin{align*}
&\Gamma; \Delta_1 \vdash A \\
&\frac{-A, \Gamma_1; \Delta_1 \vdash e}{-A, \Gamma_1; \Delta_1 \vdash e} \quad \text{Unlock} \\
&\vdots \\
&\frac{-A, \Gamma; \Delta \vdash A}{\Gamma; \Delta \vdash A} \quad \text{Lock} \\
&\frac{\Gamma; \Delta, \Delta' \vdash A}{\Gamma; \Delta \Delta' \vdash B} \quad \text{cut}_1
\end{align*}
\]

Multiple, structural cuts are needed to cut the extra copies of \(A\) that appear as \(-A\) on the left-hand side, which may be unlocked multiple times when green formulas are encountered on the right-hand side. These multiple cuts will entail not only the need to contract copies of \(B\) at the end, but also copies of the multiset \(\Delta'\), which is not normally subject to contraction. For example, the proof of \(-A, \Gamma; \Delta \vdash A\) may require a \(\otimes R\) rule that splits the context \(\Delta\), but which copies \(-A\) to each premise: these copies will spawn multiple copies of \(\Delta'\) in the resulting proof after cuts are applied to the subproofs. Thus \text{Lock} must be generalized to contract more than just the right-hand side (stoup) formula. The given rules for \text{Lock} and \text{Unlock} in Figure 1 subsume the simpler cases since we can choose \(\Theta\) to be empty (thus representing the formula 1). Clearly the generalized \text{Lock} still represents instances of \((\neg P \to P) \to P\), with \(P\) replaced by \(\Theta \otimes \neg A\): the rule is semantically sound. The \text{Unlock} optionally retains \([\Theta : A]\) inside \(\Gamma\) in the premise.

An example of a formula that requires not just the stoup formula to be copied in its cut-free proof is

\[(b \to a) \to (((b \to c) \to e) \to (e \to a) \to a)\]

where \(e\) is a green atom and \(a\) is a red one. This mixture of intuitionistic and affine implication requires that the premise \(b \to a\) be copied because of the requirements of \(\to L\).

The \text{Pr} rule corresponds to the counterpart or dual of the Peirce-like formula: \((A \to \neg A) \to \neg A\). It is in fact possible to derive a rule similar to \text{Pr} using the generalized \text{Lock} and \text{Unlock} rules:

\[
\begin{align*}
&\Gamma; \Delta, A, A \vdash e \\
&\frac{[A : e], \Gamma; \Delta, A \vdash e}{\Gamma; \Delta, A \vdash e} \quad \text{Unlock} \\
&\frac{\Gamma; \Delta, A \vdash e}{\Gamma; \Delta, A \vdash e} \quad \text{Lock}
\end{align*}
\]

That is, contraction inside the multiset context also becomes valid when a green \(e\) is found in the stoup. The generalized \text{Lock} rule captures not only the Peirce-like formula for right-side contraction, but also its dual for left-side contraction. This duality is crucial for cut elimination to succeed. The \text{Pr} rule is not technically equivalent to this derived rule because of our dyadic representation of sequents (using both sets and multisets). Thus \text{Pr} is kept as a separate inference rule. It is needed to prove formulas such as \((A \to \neg A) \to \neg A\).

The key contrast between the role of green formulas and that of the ? operator in linear logic can be described as dynamic versus static approaches to allowing contraction. Once we place a ? before a formula, it can be contracted anywhere. However, ?\(A\) only enables contraction on itself. In contrast, the presence of a green formula in the stoup effectively switches the proof into a “classical mode;” contractions become unlocked on all formulas, left and right. Conceptually, this means that we do not have to keep ? on all the formulas that may at some point require contraction. Subformula occurrences of green formulas mean that it is possible but not necessary for a proof to include classical fragments. They determine where in the proof, as opposed to on which formulas, are contractions allowed. Classical reasoning is thus localized inside segments of proofs. In proving a formula such as Peirce’s: \(((P \to E) \to P) \to P\), only \(E\) needs to be green whereas in linear logic, clearly more than one ? would be needed. In ACL, there is no restriction on the formula \(P\): no ? is required for it to be contracted. Only the inner \(P \to E\) becomes a classical implication: the others keep their strengths in the sense that the proof segment below \text{Unlock} stays non-classical, and must stay as such.
It should also not be assumed that the presence of a green \(e\) in the stoup cancels the meaning of all non-classical connectives and constants. For example, while \(\neg\neg A \rightarrow E\) is provable, \(\neg\neg E \rightarrow E\) is still not provable. The constant 0, being red, cannot enable a contraction on \(E\). Also, it does not hold that \((E_1 \land E_2) \rightarrow (E_1 \otimes E_2)\) even when \(E_1\) and \(E_2\) are both green (\(A \otimes B\) is always red). It would be entirely incorrect to suppose that the entire subproof above a sequent with \(e\) in the stoup becomes classical. Once the green \(e\) vacates the stoup, by an Unlock for example, the classical mode is canceled. The contracted formulas do not lose their non-classical strength. Only in the “purely classical” fragment, where all atoms are green and 0 and \(\otimes\) are not used, does LAC become classical logic.

4.1 Fragments of ACL

Polarity information is used in LAC to enforce soundness: classical versus non-classical soundness. Without the common restriction on Unlock, \(Pr\) and \(\perp L\), it is easy to see that LAC degenerates into another proof system for classical logic, with some redundant symbols and rules.

We enumerate some important fragments of ACL, all determined by the subformula property. However, ACL is more than the sum of these fragments because connectives can mix without restriction.

- **Purely Negative ACL.** Restrict to \(\land, \rightarrow, \neg\rightarrow, \perp\) and 1. The semantic interpretations of these connectives do not require the closure operator \(cl\). This fragment is the core of ACL. It can be given a simpler, Kripke style semantics. It is already possible to have control operators in this fragment, without becoming entirely classical.

- **Affine-Linear and Classical Logic.** Do not use \(\rightarrow\).

- **Classical Logic.** Color all atoms green and restrict to \(\land, \lor, \rightarrow, \perp\) and 1. In this fragment, \(\neg\) becomes an involutive negation. The polarity restriction in the Unlock, \(Pr\) and \(\perp L\) rules becomes meaningless.

- **Negative Fragment of Intuitionistic Logic.** Color all atoms red and restrict to \(\land, \rightarrow, 0\) and 1. All formulas are red. All proofs, even partial proofs, are intuitionistic once useless Locks are discarded. ACL as defined above is not complete with respect to full propositional intuitionistic logic without using green formulas. The connective \(\lor\) cannot be considered the equivalent of intuitionistic disjunction. The axiom \((A \rightarrow C) \rightarrow (B \rightarrow C) \rightarrow (A \lor B) \rightarrow C\) is not provable in ACL if \(C\) is red. All other propositional intuitionistic axioms are provable. The negative fragment of intuitionistic logic is often the only one considered for computational purposes.

4.2 Adding Intuitionistic Disjunction

It is not difficult to extend ACL as given to include full propositional intuitionistic logic as a fragment. Add a second disjunction \(\lor^i\) with the following semantic interpretation and sequent calculus inference rules:

\[
(A \lor^i B)^p = cl( I(A^p) \cup I(B^p) )
\]

\[
\frac{\Gamma; \mid A_i}{\Gamma; \mid A_1 \lor^i A_2} \quad \lor^i R 
\quad \frac{A, \Gamma; \Delta \vdash C \quad B, \Gamma; \Delta \vdash C}{\Gamma; \Delta, A \lor^i B \vdash C} \quad \lor^i L
\]

The polarity of \(A \lor^i B\) must be designated as *always red* since there is no guarantee that \((A \lor^i B)^p\) valuates to \(\perp\) or 1 even if both \(A\) and \(B\) are green. The soundness of the introduction rules is readily checked. Our cut elimination and completeness proofs are extended with the necessary cases without difficulty. However, we currently have no better reason to include this connective than just for the sake of intuitionistic completeness. Adding \(\lor^i\) has the consequence that the intuitionistic implication \(\rightarrow\) becomes redundant:

\[
A \rightarrow B \equiv (A \lor^i 0) \rightarrow B
\]

The formula \(A \lor^i 0\) mimics \(!A\) in linear logic. It holds that \((A \lor^i B) \rightarrow (A \lor B)\). \(A \lor^i \neg A\) is also provable. Our semantic framework allows other connectives to be considered as well. However the goal of ACL is not
to contain every possible connective but to achieve a combination with good proof theoretic properties that are also computationally useful. It is not currently clear what the practical benefits of adding $\lor^i$ would be in addition to $\lor$: both are additive connectives in a single conclusion sequent calculus. It is available if the need for it arises (perhaps in a multiple-conclusion intuitionistic setting). Other fragments of ACL become possible with this new connective, including our previous effort, polarized control logic (PCL), which does contain $\lor^i$ but not $\oplus$, $\rightarrow$ and $\otimes$. However, PCL models are equivalent to purely intuitionistic models where $I(W) = W$, thus the difference between $\lor$ and $\lor^i$ disappears.

5 Cut Elimination

In order to consider cut-elimination carefully, let us regard *modus ponens* in the following forms.

\[
\frac{\ldots \vdash A \otimes (A \rightarrow B) \quad \ldots \vdash B}{\ldots \vdash A \land (A \rightarrow B) \quad \ldots \vdash B}
\]

Other forms, that use other combinations of $\rightarrow$, $\rightarrow$, $\land$ and $\otimes$ are not generally valid, at least not without restrictions. This analysis implies that the cut rules relative to LAC should be in two forms:

\[
\frac{\Gamma, \Delta_1 \vdash A \quad \Gamma, \Delta_2, A \vdash B}{\Gamma, \Delta_1, \Delta_2 \vdash B} \quad \text{cut}_1 \quad \frac{\Gamma \vdash A \quad A, \Gamma \vdash B}{\Gamma \vdash B} \quad \text{cut}_2
\]

Lessons from linear logic may suggest that $\text{cut}_1$ (and $\text{cut}_2$) cannot be admissible, for a contraction in the form of *Lock* is possible on the cut formula $A$ in the left subproof. We do not have the $!$ operator and the $Pr$ rule is not an exact analogue. This suggests that the context $\Delta_2$ in $\text{cut}_1$ (and $\Delta$ in $\text{cut}_2$) should be empty. But the semantics clearly show that the cuts are sound as written: this is no guarantee that cut elimination would work but it gives us considerable confidence to proceed.

By the admissibility of both weakening and contraction in the $\Gamma$ context, we could have written the rules multiplicatively with respect to $\Gamma$ as well, but that would only confuse the issue here. However, we will not hesitate to split $\Gamma$ into $\Gamma_1 \Gamma_2$ whenever we find it to be more convenient.

Formulas $[\Theta : A]$ cannot be cut formulas because they cannot appear on the right-hand side. Formulas inside the $[\ ]$ are not subject to cut rules. See Corollary 5 for further clarification.

With a minor exception in the case of the $\bot L$ rule, the reduction of cut with respect to the introduction rules is rather mundane: our introduction rules are no different from those found elsewhere, such as in the logic programming language Lolli [HM94]. The structural rules dominate the cut-elimination proof. The proof is by simultaneous induction on both $\text{cut}_1$ and $\text{cut}_2$. The inductive measure is the lexicographical ordering consisting of the size of the cut formula, followed by the number of *Lock* rules on the cut formula above the cut, followed by the number of $Pr$ rules on the cut formula above the cut, followed by the number of $Dr$ rules on the cut formula above the cut, followed by the heights of subproofs.

We detail the permutation of cut above *Lock*, *Unlock*, *Pr* and *Dr* below.

The case of Lock

This case of the cut-elimination procedure demonstrates the presence of *structural* as opposed to *logical* cuts as described by Parigot for the $\lambda \mu$-calculus. We show the case for $\text{cut}_2$ as the case for $\text{cut}_1$ is simpler. The scenario for permuting $\text{cut}_2$ above *Lock* is the following

\[
\frac{\Gamma_1 ; \Delta_1 \vdash A}{[\ : A], \Gamma_1 ; \Delta_1 \vdash \epsilon_1} \quad \text{Unlock} \quad \frac{\Gamma_1 ; A \vdash A}{[\ : A], \Gamma ; \vdash A} \quad \text{Lock} \quad \frac{A, \Gamma ; \Delta \vdash B}{\Gamma \Gamma_1 ; \Delta \vdash B} \quad \text{cut}_2
\]
The figure means to convey that there could be multiple Unlocks above the left subproof, possibly stacked. Note that although the affine-linear context is initially empty in the left subproof (regarding the proof from the bottom), it may become non-empty when Unlock is applied. This cut is reduced as follows:

\[
\frac{\Gamma_1; \Delta_1 \vdash A \quad A, \Gamma' \vdash B}{\Delta; \Delta' \vdash B} \text{ cut}_2
\]

\[
\frac{\Gamma \vdash A, \Gamma' \vdash B}{\Delta; \Delta' \vdash B} \text{ Unlock}
\]

\[
\frac{\Delta: B, \Gamma \vdash A}{\Delta: B, \Gamma' \vdash A} \text{ Pr*}
\]

\[
\vdots
\]

\[
\frac{\Delta: B, \Gamma' \vdash A}{\Delta: B, \Gamma' \vdash B} \text{ Lock}
\]

The illustration generalizes to cuts on multiple branches and to stacked cuts, in which case \(\Delta: B\) is not removed from \(\Gamma\) until the topmost occurrence of Unlock. Naturally we need not be concerned with duplication inside the intuitionistic (\(\Gamma\)) context: we chose a multiplicative presentation of these contexts here for clarity.

The case for cut_1 is in fact a simpler version of the cut_2 case because it does not require the Pr rule. For both cut_1 and cut_2, there is another case where the cut formula is inside the affine-linear multiset being locked. For example:

\[
\frac{\Gamma_1': \Delta_1' \Theta, A \vdash B}{\Theta, A : B, \Gamma_1'; \Delta_1' \vdash e_1} \text{ Unlock}
\]

\[
\vdots
\]

\[
\frac{\Theta, A : B, \Gamma_1'; \Delta_1' \vdash e_1}{\Gamma_1'; \Delta_1' \vdash B} \text{ Lock}
\]

\[
\frac{\Gamma_1; \Delta \vdash A}{\Gamma_1'; \Delta' \Theta, A \vdash B} \text{ cut}_1
\]

Such a case reduces to

\[
\frac{\Gamma; \Delta \vdash A}{\Gamma_1'; \Delta_1' \Theta, A \vdash B} \text{ cut}_1
\]

\[
\frac{\Gamma_1'; \Delta_1' \Theta \vdash B}{\Delta_1' : B, \Gamma_1'; \Delta_1' \vdash e_1} \text{ Unlock}
\]

\[
\vdots
\]

\[
\frac{\Delta_1' : B, \Gamma_1'; \Delta_1' \vdash e_1}{\Gamma_1'; \Delta_1' \vdash B} \text{ Lock}
\]

So we have merely replaced locking \(A\) with locking \(\Delta\).

**The case of Dr**

The potential problem here is the effect of Dr on the applicability of cut_2, which requires an empty affine-linear context:

\[
\frac{\Gamma_1, \Delta_1 \vdash C_1}{A, \Gamma_1; \Delta_1 \vdash C_1} \text{ Dr}
\]

\[
\frac{\Gamma; B \vdash A}{\Gamma; \Delta \vdash C} \text{ cut}_2
\]
The notation \( \vdash \) represents multiple instances of \( Dr \) on the cut formula \( A \) in the right subproof. \( A \) can be ignored above the topmost \( Dr \). This cut is reduced as follows:

\[
\frac{\Gamma; B \vdash A \quad \Gamma_1; \Delta_1, A \vdash C_1}{\Delta_1; \Gamma_1; B \vdash C_1} \quad \text{cut}_1
\]

\[
\frac{B, \Gamma_1; \Delta_1 \vdash C_1}{\Gamma, B, \Gamma_1; \Delta_1 \vdash C} \quad \text{Dr}
\]

The permutation of \( Dr \) above \( \text{cut}_1 \) is relatively trivial.

The case of Unlock

This case was also appears in PCL, but was argued for a multiple-conclusion proof system. It concerns permuting cut above Unlock (and to a lesser extent \( \perp E \)) We have restricted \( e \) in Unlock and \( \perp E \) to be a green atom or \( \perp \). In fact this restriction can be relaxed to allow any formula \( E \): one can check that this relaxation preserves semantic soundness since green formulas are characterized by exactly the same semantic properties as green atoms. The restriction was used for two reasons. First, technically speaking, “sequent calculus” should not look beyond the top-level form of a formula to determine which inference rule applies. The second reason is that it allows us to write a simpler cut elimination proof.

Let us temporarily refer to the relaxed version of Unlock as Unlock\(^'\). A crucial case of cut-elimination for LAC is the following:

\[
\frac{\Gamma; \Delta_1 \Theta \vdash A \quad \Theta : A \quad \Gamma; \Delta_1 \vdash B}{\Gamma; \Delta_2 \vdash B} \quad \text{Unlock}\(^'\), \text{cut}
\]

A similar case occurs with the \( \perp L \) rule and with \( \text{cut}_2 \).

Cut-elimination in sequent calculus usually calls for the cut to be permuted parametrically above each inference rule to reach a “key case” where the cut formula in both sequents are the principal formulas of introduction rules. That strategy clearly would not work here. However, if \( E \) was a green atom, then the cut can only be permuted parametrically above the right-side subproof (the “attractive” subproof) until an \( Id \) rule is reached, at which point the cut is eliminated by substitution as in natural deduction. If the cut formula \( e \) is \( \perp \), then we similarly permute the cut upwards until the right-proof branch reaches \( \perp L \), at which point the right-side formula in the conclusion of \( \perp L \) must be green, which means the cut can then be replaced by:

\[
\frac{\Gamma; \Delta_1 \Theta \vdash A \quad \Theta : A \quad \Gamma; \Delta_1 \vdash B}{\Gamma; \Delta_1 \Delta_2 \vdash e} \quad \text{Unlock}\(^'\), (weakening)
\]

We can also generalize cut-elimination to allow the unrestricted versions of Unlock (and \( \perp E \) and Pr) by showing that their uses can be permuted to atomic cases:

- **In a cut-free proof, Unlock\(^'\) can be replaced by Unlock.**

This is proved by showing that an Unlock\(^'\) rule on a green formula \( E \) can always be permuted to Unlock\(^'\) on its subformulas. We show the most interest case of the transformation

\[
\frac{\Gamma; \Delta \Theta \vdash A}{\Theta : A, \Gamma; \Delta \vdash B \rightarrow E} \quad \text{Unlock}\(^'\), (weakening)
\]

\[
\frac{\Gamma; \Delta \Theta, B \vdash A}{\Theta : A, \Gamma; \Delta, B \vdash E} \quad \text{Unlock}\(^'\), (\rightarrow r)
\]

The admissibility of weakening in \( \Delta \) is thus crucial for the cut-elimination argument.

Results analogous to the equivalence between Unlock and Unlock\(^'\) also hold for \( \perp L \) and for the Pr rule (Lemma 2 below).
The case of Pr

The methods used in the previous cases are combined in the case of permuting cut above the Pr rule. There are two principal scenarios to consider:

\[
\begin{align*}
\Gamma; \Delta; A \vdash e & \quad \text{Pr} \\
\Gamma; \Delta, A \vdash e & \quad \text{cut}_1
\end{align*}
\]

The first (left) case is solved by restricting \( e \) to be a green atom or \( \bot \), just as in the case for Unlock, then showing that the restriction can be relaxed to any green formula using a separate set of permutations. One example should suffice to convince:

\[
\begin{align*}
A, \Gamma; \Delta; A \vdash B \lor E & \quad \text{Pr} \\
\Gamma; \Delta, A \vdash B \lor E & \quad \text{Unlock} \quad \vdash B \lor E, \Gamma; \Delta, A \vdash B \lor E \quad \text{Pr} \\
\vdash B \lor E, \Gamma; \Delta, A \vdash B \lor E & \quad \lor R \\
\Gamma; \Delta, A \vdash B \lor E & \quad \text{Lock}
\end{align*}
\]

A green (classical) disjunction is only “additive” in a superficial sense. The argument is similar in the cases of \( \land \), \( \rightarrow \) and \( \_ \) (weakening is required in the cases of \( \rightarrow \) and \( \_ \)). However, had we naively designated \( B \otimes C \) to be green if \( B \) “and” \( C \) are green, then such a permutation cannot be made (the context is split below Pr). Semantically, \( B \otimes C \) is always red because we cannot guarantee that it will be valid above the root even when \( B \) and \( C \) are both green. This rather abstract semantic explanation is represented syntactically in the non-permutability of \( \otimes \) above Pr (and Unlock).

The fact that the restriction on Unlock, Pr and \( \bot \) can be all be relaxed is an important property of ACL, since the relaxed forms will be used in the completeness proof and in the natural deduction system.

**Lemma 2** The restriction to \( e \) being a green atom or \( \bot \) in the Unlock, Pr and \( \bot \) rules can be relaxed to allow any green formula \( E \).

The other case of permuting cut with respect to Pr is relatively simple since the Pr rule can be duplicated beneath to remove the additional copies of \( \Delta' \). That is, we permute the cut above instances of Dr on \( A \): if there are no such instances then the result follows from weakening. Otherwise we have:

\[
\begin{align*}
\Gamma; \Delta \vdash A & \quad \Gamma_1; \Delta_1; A \vdash D \quad \text{cut}_1 \\
\Delta \vdash T \Gamma_1; \Delta_1; \Delta' \vdash D & \quad \text{Dr}^* \\
\vdash \Delta \vdash e & \quad \text{Pr}^* \\
\Gamma; \Delta \vdash e & \quad \text{Pr}^*
\end{align*}
\]

The arguments for \( \text{cut}_2 \) are the same as for \( \text{cut}_1 \).

**Theorem 3** \( \text{cut}_1 \) and \( \text{cut}_2 \) are admissible in LAC.

5.1 Consequences of Cut Elimination

Another result, relatively easy to prove but which is crucial for completeness, is initial elimination.
Theorem 4: \( A \vdash A \) is provable for any formula \( A \)

The Unlock rule is a special case of \( \rightarrow L \), but we wish to keep the effect integral (a focused effect):

Corollary 5: \( [\Theta : A], \Gamma; \Delta \vdash B \) is provable if and only if \( -(\Theta^0 \rightarrow A), \Gamma; \Delta \vdash B \) is provable

The forward direction (soundness of focusing) follows because Unlock can be emulated with \( \rightarrow L, \perp L, \rightarrow R \) and \( \otimes L \). The reverse direction (completeness of focusing) follows from cut elimination and initial elimination because we can show that

\[ [\Theta : A] \vdash -(\Theta^0 \rightarrow A) \]

is provable. This corollary is also critical for completeness.

The most important use of focusing in the Unlock rule is that the right premise of the implicit \( \rightarrow L \) must be the conclusion of an initial rule (\( \perp L \)).

Another relatively obvious but important result is the following:

Proposition 6: If a formula is provable with an atom \( b \) colored red, then the same formula is provable with \( b \) colored green.

This holds because a green atom can only lead to more proofs. The consequences of this lemma are significant. Combined with cut elimination and initial elimination, it allows us to show that LAC has the substitution property.

Theorem 7: The substitution property for LAC holds as follows:

1. If a formula \( A \) is provable with an atom \( b \) colored red, then \( A[C/b] \) is also provable for any formula \( C \).
2. If a formula \( A \) is provable with an atom \( e \) colored green, then \( A[E/e] \) is also provable for any green formula \( E \).

Part 2 of this theorem follows from Lemma 2.

The other important consequence of Theorem 7 is that, were we to extend ACL to include second order propositional quantifiers, then the polarities of bound variables are not in question: universally quantified propositional variables are red, while existentially quantified ones are green.

We observe that, since ACL is intended as a unified logic, in the worst case, cut-elimination could become as uncontrolled as in classical LK; in particular when all atoms are green. However, such uncontrolled segments are localized in proofs. Cut elimination involving the Unlock and the Pr rules are all by substitution as in natural deduction, without duplicating structure in the subproofs that end in these rules.

Polarity information in ACL affects cut elimination, just as in other polarized systems. However, polarization in ACL (and PCL) is principally concerned with provability, not with the structure of proofs.

6 Soundness and Completeness

The soundness of LAC inference rules is argued by induction on the structure of proofs. In particular, Lock is sound by the validity of the version of Peirce’s formula \( \neg P \rightarrow P \rightarrow P \), and Pr is sound because of its counterpart \( (P \rightarrow \neg P) \rightarrow \neg P \). The Unlock rule is just a synchronized instance of \( \rightarrow L \). The other rules can be checked to be sound case by case. For example, the soundness of the \( \forall L \) rule holds since if \( A^p \subseteq C^p \) and \( B^p \subseteq C^p \) then \( A^p \cup B^p \subseteq C^p \) and thus \( cl(A^p \cup B^p) \subseteq cl(C^p) = C^p \).

The completeness proof of ACL differs from other phase semantic completeness proofs principally in the following ways. Because of the meaning and central role of \( \perp \), the unit/root of the monoid that we build is not the empty set or multiset. Instead, it is a maximally consistent set with the characteristics of Hintikka sets. Also, instead of constructing a canonical model of all proofs, we build a countermodel for a formula that’s assumed to be unprovable. In addition, our completeness proof differs from others in that it requires cut-elimination, for otherwise there is no mention of the Lock rule in the proof.
Assuming that a formula $A$ is not provable, we show the existence of a countermodel as follows. A set or multiset $\Theta$ is said to be consistent with respect to a formula $P$ if $P$ is not derivable from it. The root world of the model will be a set that is maximally consistent with respect to $A$ and to $\bot$. In the following we write $\Gamma; \Delta \not\vdash A$ to mean $\Gamma; \Delta \not\vdash A$ is not provable.

**Lemma 8** If $;\not\vdash A$ then $;\not\vdash A \lor \bot$.

This is a non-trivial lemma. First, it is shown that $\bot$ has no cut-free proof. Then we show the contrapositive of the lemma. The essential argument is that, first, we show if $A \lor \bot$ is provable then it should follow from $[\vdash A \lor \bot]; \vdash A$. But when $A \lor \bot$ is unlocked there must be a green subformula $e$ of $A$ in the stoup, which means that if $\bot$ becomes derivable from the left-hand side at this point, then so is $e$. We can therefore continue to emulate the proof of $A \lor \bot$ to construct a proof of $A$.

**Lemma 9** If $B \lor C, \Gamma; \not\vdash A \lor \bot$ then either $B, \Gamma; \not\vdash A \lor \bot$ or $C, \Gamma; \not\vdash A \lor \bot$.

This lemma holds because $A \lor \bot$ is green. By Lemma 2 we can assume that the relaxed form of $PR$ can be used in proofs. If $B \lor C, \Gamma; \not\vdash A \lor \bot$ then $B \lor C \not\vdash A \lor \bot$ by the $DR$ rule (arguing the contrapositive). But then by the $\lor L$ rule either $B, \not\vdash A \lor \bot$ or $C, \Gamma; \not\vdash A \lor \bot$. Thus by the (relaxed) $PR$ rule either $B, \Gamma; \not\vdash A \lor \bot$ or $C, \Gamma; \not\vdash A \lor \bot$.

Define a proxy subformula $B$ of a formula $P$ to be either a subformula of $P$ or a formula in the form $\Delta \rightarrow B$ where $B$ and every $D \in \Delta$ are subformulas of $P$. The $Lock$ rule is implicitly applied to proxy subformulas.

For the purpose of the completeness proof, we extend the notion of the provability of $\Gamma; \Delta \not\vdash B$ to allow $\Gamma$ to be an infinite set. Such an infinite sequent is provable if $\Gamma; \not\vdash B$ is provable for some finite subset $\Gamma'$ of $\Gamma$.

Now we construct a countermodel $CA$ as follows:

1. A possible world in $W$ consists of a set $\Gamma$ and a multiset $\Delta$ of formulas that we simply write as $\Gamma \Delta$. Let $\Gamma^\infty$ represent a multiset such that, for each distinct formula $A$ in $\Gamma$, there are countably infinite many occurrences of $A$ in $\Gamma^\infty$. This device type casts a set into a multiset and simplifies some arguments. $\Delta$ will always be a finite multiset so if $\bot$ becomes derivable from the left-hand side at this point, then so is $e$. We can therefore continue to emulate the proof of $A \lor \bot$ to construct a proof of $A$.

2. Construct the root world $r = \Gamma_r$ as follows. Enumerate all proxy subformulas $B$ of $A$ and their negations $\neg B$. Then construct $\Gamma_r$ to be a maximally consistent set with respect to $A \lor \bot$ by inserting each $B$ or $\neg B$ into $\Gamma_r$ as long as $\Gamma_r$ remains $A \lor \bot$-consistent (by “inserting” we of course mean a hypothetical construction to show that such a saturation exists). By Corollary 5, inserting $\neg ((\Delta \rightarrow \bot)$ is equivalent to inserting $[\Delta : C]$. Two other properties are assured:

   a. It cannot be the case that $B$ and $\neg B$ are both in $\Gamma_r$ as that would mean that $\bot$ and thus $A \lor \bot$ are derivable from $\Gamma_r$. Since $\Gamma_r$ is $\bot$-consistent, it must also be $0$-consistent.

   b. If $\Gamma_r; \not\vdash A \lor \bot$, then $B \lor \neg B, \Gamma; \not\vdash A \lor \bot$ because $;\vdash B \lor \neg B$ is provable and cut is admissible. By Lemma 9, this means that in a maximally consistent saturation exactly one of either $B$ or $\neg B$ will be inserted into $\Gamma_r$. With $\Gamma_r$ thus saturated, it follows that any proper addition to $\Gamma_r$ (limited to the proxy subformulas of $A$ and their negations) will render it $\bot$-inconsistent. In other words, either $\Gamma_r C = \Gamma_r$ or $\Gamma_r; C \not\vdash \bot$ becomes provable. This is the most critical use of cut elimination in the completeness proof. It confirms that the $Lock$ rule, which is otherwise not directly referred to in this proof, is required for completeness.

3. The worlds $W$ consists of all pairs $\Gamma \Delta$ of proxy subformulas and their negations such that $\Gamma_r \subseteq \Gamma$. Furthermore, we can assume that the number of formulas in $\Gamma \Delta$ is finite. This assumption is important.
It is easily verified that $\Gamma_r$ satisfy the requirements of being the root. $I(W)$ corresponds to those worlds where the proper multiset $\Delta$ is empty.

The rest of the proof mostly emulates Okada.

4. For any formula $A$, let $Pr(A) = \{\Gamma \Delta : \Gamma \vdash \Delta \text{ is provable}\}$. By the admissibility of weakening, $Pr(A)$ is upwardly closed. The set of facts $D$ of the model are restricted to be those subsets of $W$ that are equivalent to $\bigcap Pr(A_i)$ where $A_i$ ranges over an arbitrary collection of formulas $A_0, \ldots, A_i, \ldots$. Clearly we have $1^p \in D$ since $Pr(1) = W$ and $\bot^p \in D$ since $Pr(\bot) = W\setminus \{r\}$. $D$ is certainly closed under the $\bigcap$ operator as defined. To see that, if $B \in D$ then $\{x : \text{for all } y \in A, xy \in B\} \in D$. Assume that $B = \bigcap Pr(C_i)$ and $\Gamma \Delta$ is in this set. Then for any $\Gamma, \Gamma' \Delta' \in B$, we have that $\Gamma' \Delta' \vdash C_i$ is provable for all $C_i$. Since we can assume that $\Gamma'$ and $\Delta'$ are finite sets and multisets, this means that $\Gamma ; \Delta \vdash \Gamma' \Delta' \rightarrow \Delta'^{\otimes} \rightarrow C_i$ is provable. Thus $\Gamma \Delta \subseteq \bigcap Pr(\Gamma' \Delta' \rightarrow \Delta'^{\otimes} \rightarrow C_i)$ for all $C_i$ and $\Gamma ; \Gamma' \Delta' \in B$, thus qualifying as a fact.

5. The valuation of atomic formulas is defined to be

$$a^p = Pr(a) = \{\Gamma \Delta : \Gamma \vdash a \text{ is provable}\}$$

Naturally, green atoms are mapped to $\bot^p$ or $1^p$ since all $\Gamma \Delta$ above $\Gamma_r$ derives $\bot$ and therefore all green formulas (by cut). The fact $0^p = \bigcap D$ is provable. Clearly this is the smallest fact since (by cut) $Pr(0) \subseteq Pr(B)$ for all formulas $B$. $0^p$ is not empty if 0 is a subformula of the formula that’s assumed to be unprovable.

6. We can show that $B^p = Pr(B)$ for all formulas $B$. However, for completeness it is only necessary to show that $\Gamma_r ; B \in B^p$ and $B^p \subseteq Pr(B)$. This is proved by mutual induction on the structure of $B$. The cases for atoms and constants are trivial. We show a selection of representative cases for the connectives.

For $\Gamma_r ; A \otimes B \in (A \otimes B)^p$ we need to show that if $\Gamma_r ; A \in A^p$ and $\Gamma_r ; B \in B^p$ then $\Gamma_r ; A \otimes B \in (A \otimes B)^p$. So $A^p \otimes B^p \subseteq Pr(A) \otimes Pr(B)$. By the $\otimes R$ rule, $Pr(A) \otimes Pr(B) \subseteq Pr(A \otimes B)$. Since $(A \otimes B)^p$ is the intersection of all facts that contain $A^p \otimes B^p$ and $Pr(A \otimes B)$ is a fact, it holds that $(A \otimes B)^p \subseteq Pr(A \otimes B)$.

Notice here we can apply cut elimination to show that in fact $Pr(A \otimes B) \subseteq (A \otimes B)^p$, but this is not necessary.

For $(A \otimes B)^p = cl(A^p \otimes B^p) \subseteq Pr(A \otimes B)$: by inductive hypothesis $A', A \in A^p$. By the $\otimes R$ rule, $Pr(A) \otimes Pr(B) \subseteq Pr(A \otimes B)$. Since $(A \otimes B)^p$ is the intersection of all facts that contain $A^p \otimes B^p$ and $Pr(A \otimes B)$ is a fact, it holds that $(A \otimes B)^p \subseteq Pr(A \otimes B)$.

The mutual induction is required in the case of implication. For $\Gamma_r ; A \rightarrow B \in (A \rightarrow B)^p$ we need to show that if $\Gamma_r ; A \in A^p$ then $\Gamma_r ; A \rightarrow B \in (A \rightarrow B)^p$. By inductive hypothesis $A^p \subseteq Pr(A)$ and $\Gamma_r ; B \in B^p = \bigcap Pr(C_i)$ where $C_i$ ranges over a collection of formulas. Thus $\Gamma_r ; A \rightarrow B \in (A \rightarrow B)^p$.

For $(A \rightarrow B)^p \subseteq Pr(A \rightarrow B)$, by inductive hypothesis $\Gamma_r ; A \in A^p$ and $B^p \subseteq Pr(B)$. Thus if $\Gamma_r \in (A \rightarrow B)^p$ then $\Gamma_r ; A \rightarrow B \in B^p \subseteq Pr(B)$. So by the $\rightarrow R$ rule we have that $\Gamma_r ; A \rightarrow B \in Pr(A \rightarrow B)$.

For the implication $\rightarrow$, if $A \in A^p$ then $A^m \in A^p \cap I(W)$ (by the $DR$ rule), and if $\Gamma_r \notin A^p \cap I(W)$ then $\Delta$ must be empty. With these observations similar arguments as for $\rightarrow$ can then be applied.

7. Completeness then follows since $\Gamma_r ; \neg A$ and thus $\Gamma_r \notin Pr(A)$, so $\Gamma_r \notin A^p$. That is, the unit/root of the monoid is not found inside $A^p$ (in terms of Kripke models, $\Gamma_r \notin A$).

**Theorem 10** A formula is provable in LAC if and only if it is valid in ACL.
of ACL are consistent with those of affine linear logic (facts are ideals). Although the model CA
tional affine linear logic is decidable [Kop95], and in [Laf97] Lafont gave a phase model proof. The models
This section defines a natural deduction proof system
Also a model of affine linear logic. For the time being, we leave the question of propositional decidability to
is finite. It would be somewhat of a surprise, however, if ACL is not decidable since every model of ACL is
Figure 2: The Natural Deduction System NAC. E must be green in Unlock, Pr and ⊥E

One might expect the completeness proof to shed light on the question of decidability for ACL. Propositional
itional affine linear logic is decidable [Kop95], and in [Laf97] Lafont gave a phase model proof. The models
of ACL are consistent with those of affine linear logic (facts are ideals). Although the model CA of the
completeness proof is infinite, it is possible to construct a quotient model by defining a congruence relation.
However, we are not able to duplicate Lafont’s arguments further because the quotient model is not finitely
generated. This is because there can be infinitely many proxy subformulas of a formula. To prove decidability
along these lines we would need to show that the number of possible formulas subject to Lock in a proof
is finite. It would be somewhat of a surprise, however, if ACL is not decidable since every model of ACL is
also a model of affine linear logic. For the time being, we leave the question of propositional decidability to
future work.

7 A Curry-Howard Interpretation

This section defines a natural deduction proof system NAC (Figure 2) with a computational interpretation of
proofs. We restrict to the connectives \( \rightarrow, \leftarrow, \) and \( \otimes \) as \( \land \) and \( \lor \) are uninteresting for the intent of this section. We prefer to associate a proof term with an entire subproof, and not just the formula on the right-side of \( \vdash \). However, when we speak of the “type” of a term, we are naturally referring to this “stoup” formula. The
intent of this section is to demonstrate one possible interpretation of proofs as opposed to exploring the full
range of computational possibilities that ACL may offer.

The Lock/Unlock rules are modified in two ways. First, \( E \) can be any green formula, not just green
atoms and \( \bot \). Lemma 2 justifies this change. Secondly, the Lock/Unlock rules are now parameterized. The
notation \( [\Theta : A] E \) has the same meaning as the formula \( (\Theta \rightarrow A) \rightarrow E \). These rules are sound because
Peirce’s formula \( ((P \rightarrow E) \rightarrow P) \rightarrow P \) holds for any green formula \( E \), not just \( \bot \). The original Lock rule is
now seen as Lock\( \bot \). The Lock\( E \) rule only superficially violates the subformula property. It is useless without
Unlock\( E \), which can only be applied if \( E \) is a subformula of what is being proved. Instances of Lock\( E \) where
\( E \) is not such a subformula can be discarded from proofs.

A formula locked using Lock\( \bot \) can still be unlocked by any green formula \( E \) using the \( \bot \)-elimination rule;
thus the new rules are also complete with respect to the LAC rules. The modified rules are not required
for completeness but are more useful in that they allow us to use the green formulas more meaningfully as types.

The following admissible rule is not stated as part of the system, but may be used implicitly:

\[
\frac{t : B^x, B^y, \Gamma; \Delta \vdash A}{t[x/y] : B^x, \Gamma; \Delta \vdash A}
\]

It is easily shown that this system is complete in that LAC can be recaptured. Soundness can be shown semantically, or by using cut elimination.

All formulas except the stoup are indexed. The notation \([\Theta : A]_E^d\) indicates that the index variable \(d\) is associated with the entire intended formula \((\Theta \to \to A) \to E\). Such boxed formulas are indexed by \(\mu\)-variables while others are indexed by \(\lambda\) variables. We assume that these variables are always distinguishable and are renamed to avoid clash when necessary.

Terms \(yd.s\) are equivalent in their computational behavior to \(\mu d|[s]\) in classical \(\lambda\mu\) calculus. However, we have chosen not to use \(\mu d|[s]\) to represent the \(\Lock\) rule as it would incorrectly suggest that \(\Lock\) can be decomposed into smaller steps. The \(\lambda\mu\) notation \([d]t\) in this context is equivalent to \((d \cdot t)\).

There are two types of lambda abstraction: \(\lambda\) and \(\chi\), that correspond to \(\to\) and \(\to\) respectively. There are also two types of application: \((st)\) and \((st)\): these correspond to the implicit conjunctions \(A \land (A \to B)\) and \(A \to (A \to B)\) respectively. One potential problem with linear lambda terms is how to type terms such as \(\lambda x.((\lambda f.\lambda y.f \ (f \ y)) \ x)\): \(x\) appears once before reduction but twice afterwards. In (intuitionistic) linear logic there is only one \(\to\) and a \(\to\) operator that can be placed anywhere, offering few invariants. The solution to this problem in our unified logic is rather obvious: the term \(\chi x.((\lambda f.\lambda y.f \ (f \ y)) \ x)\) cannot be assigned a \(\red\) type, because of the context restriction on \(\to\) elimination. The term \(\chi x.((\lambda f.\lambda y.f \ (f \ y)) \ x)\) is not typable at all.

We have chosen to represent the \(Dr\) and \(Pr\) rules with only variable renaming in proof terms: it is difficult to assign additional computational content to \(Dr\) given that there are no left-introduction rules. It is also difficult to determine how many applications of \(Pr\) would be needed. The \(\Lock\) and \(Unlock\) rules already capture the essence of the logic: \(Dr\) and \(Pr\) play a rather managerial role. As an example, the term \(\chi x.\chi y.(x \ y)\) proves \((A \to E) \to (A \to E)\) for green \(E\). Uses of \(Pr\) and \(Dr\) are implicit in the term. Note that should we have rules for eta-reduction, then they should only apply to \(\lambda y.(x \ y)\) and \(\chi y.(x \ y)\). Thus \(\chi x.\chi y.(x \ y)\) cannot be confused with identity. A term such as \(\chi x.\chi y.(x \ y) \cdot y\), which proves the counterpart to Peirce’s formula, \((P \to \neg P) \to \neg P\), is only green typable, not red typable.

As an alternative, one can eliminate the \(Pr\) rule by folding it into \(Unlock\):

\[
\frac{t : \Gamma; \Theta \vdash A}{[d]t : [\Theta : A]_E^d, \Gamma; \Delta \vdash E} Unlock_E
\]

However, such a choice will mean that proofs for formulas such as \((P \to \neg P) \to \neg P\), which is essentially intuitionistic, would still require \(\Lock\) and \(Unlock\) and thus will look very different from a \(\lambda\) term. The term representation given here is an example of how proofs can be interpreted: it is not intended to be canonical. Certainly other interpretations can be given. For example, we can, as we have done in [LM13b] use a multiple conclusion system with two distinct contexts on the right-hand side: in such a case \(\Lock\) can be decomposed into smaller steps and \(\mu\) is restored. The same comment applies to the \(Pr\) and \(Dr\) rules.

The following sample proof represents a version of the \(call/cc\) operator.

\[
\frac{y : [P]_E^d; ((P \to E) \to P) \vdash (P \to E) \to P}{y : [P]_E^d; ((P \to E) \to P) \vdash P} Unlock_E
\]

\[
\frac{\gamma d.(x \ y,[d]y) : [\vdash ((P \to E) \to P) \vdash P]}{\chi x.\gamma d.(x \ y,[d]y) : [\vdash ((P \to E) \to P) \vdash P]} Unlock_E
\]

\[
\frac{x : P[y]_E^d; ((P \to E) \to P) \vdash (P \to E) \to P}{\gamma d.(x \ y,[d]y) : [\vdash ((P \to E) \to P) \vdash P]} Unlock_E
\]

\[
\frac{x : [P]_E^d; ((P \to E) \to P) \vdash (P \to E) \to P}{x : [P]_E^d; ((P \to E) \to P) \vdash P} Unlock_E
\]

\[
\frac{x : [P]_E^d; ((P \to E) \to P) \vdash (P \to E) \to P}{x : [P]_E^d; ((P \to E) \to P) \vdash P} Unlock_E
\]

\[
\frac{x : [P]_E^d; ((P \to E) \to P) \vdash (P \to E) \to P}{x : [P]_E^d; ((P \to E) \to P) \vdash P} Unlock_E
\]

\[
\frac{x : [P]_E^d; ((P \to E) \to P) \vdash (P \to E) \to P}{x : [P]_E^d; ((P \to E) \to P) \vdash P} Unlock_E
\]

\[
\frac{x : [P]_E^d; ((P \to E) \to P) \vdash (P \to E) \to P}{x : [P]_E^d; ((P \to E) \to P) \vdash P} Unlock_E
\]
Call this proof term \( \mathcal{K} \), then \((\mathcal{K} \cdot M) k_1 k_2 \) reduces to \( \gamma d.((M \lambda y.[d](y k_1 k_2))) k_1 k_2 \). For example, given the term context \( E[z] = (z k_1 k_2) \), \( E[\mathcal{K} \cdot M] \) reduces to \( \gamma d.E[M(\lambda y.[d]E[y])] \): this emulates the behavior of call/cc (see [dG94] for further analysis of \( \lambda \mu \)-based systems and control operators).

What is different between this call/cc and the one found in classical logic, or in a linear embedding of classical logic, is that the type of \( P \) need not be a classical. The proof beneath Unlock must be entirely affine/intuitionistic. Moreover, a subproof of \( \Gamma; \Delta \vdash E \) need not be entirely classical: once a green formula vacates the stoup, the classical Unlock and Pr rules will no longer be applicable. Such a call/cc operator is usually applied with \( P \) instantiated with an arrow type (i.e. \( y \) will capture the external continuation context). In classical logic, or in the linear logic embedding of classical logic, \( P \) can only be a classical implication, whereas in ACL is can be \( A \to B \) or \( A \to B \). The strengths of those connectives do not collapse because the use of classical reasoning is localized above Unlock.

The fact that \( \gamma \) locks in the sequence \( \mu d.[d] \ldots \) doesn’t mean that we cannot derive the more general \( C \) control operator [FFKD87]. We can prove the purely classical \( \neg \neg \to E \to E \), or the hybrid \( ((A \to \bot) \to 0) \to A \). Given that there are two implications, two constants for false, and two polarities, there are 64 versions of this formula that can be considered in the unified logic.

**Term Reduction Rules**

We list below a relatively conservative set of reduction rules for our proof terms.

- \((\lambda x.s) t \longrightarrow s[t/x] \), \((\lambda \cdot x.s) t \longrightarrow s[t/x] \)
- \((\gamma d.s) t \longrightarrow \gamma d.s[[d]t/d]\) \( t \), \((\gamma d.s) t \longrightarrow \gamma d.s[[d]t/d]\) \( t \)
- \(A(s) t/A(s) t \longrightarrow A(s) \), \(B(s) t/B(s) t \longrightarrow B(s) \)
- \(let (x;y) = (u;v) in t \longrightarrow t[u/x, v/y] \)
- \(\gamma a \gamma b.s \longrightarrow \gamma a.s[a/b] \)
- \([d] \gamma a.s \longrightarrow [d]s[d/a] \)
- \(\gamma d.s \longrightarrow s \) when \( d \) is not free in \( s \)

The first four sets of rules are reduction rules while the last three are renaming rules, which eliminate redundant Locks (redundant contractions).

**Theorem 11** The term reduction rules satisfy subject reduction and are strongly normalizing.

Subject reduction is shown by checking case by case that the rules represent valid proof transformations. All of these rules have equivalents in classical \( \lambda \mu \) calculus. All typable ACL terms are typable in classical logic: there are just fewer valid reductions because of the stronger type system. For example, the case for \( let (x;y) = (u;v) in t \) is just a special case of \( (\lambda x \lambda y.t) u v \). Every reduction path here corresponds to a reduction path in \( \lambda \mu \) calculus. The presence of Dr, Pr and the treatment of indices in the Lock and Unlock rules represent nothing more than \( \alpha \)-conversion, which does not affect normalization. Thus there is no question that this system is strongly normalizing given that the result is known for classically typed \( \lambda \mu \) terms.

The rules that we enumerated are also confluent, but we do not attach much value to such kinds of results. Confluence is only attainable after discarding certain valuable alternatives. For example, the rule

\[ (\lambda x.s) \gamma d.t \longrightarrow \gamma d.t[[d]((\lambda x.s)u)/[d]w] \]

represents a valid proof transformation that can subsume the given rules for \( \gamma \) reduction. However, it clashes with \( \beta \) reduction and must be discarded in order to achieve confluence. An effective computational interpretation of proof terms requires that certain choices must be made. It has been observed (e.g. [OS97]) that the above rule need not destroy confluence as long as one is more careful as to when it can be applied:
typically by identifying terms that represent values in a call-by-value framework. For an effective computational strategy, we found that confluent rewrite systems are still not enough. Instead we prefer the approach of deterministic abstract machine. Such a machine was defined in our preceding paper on PCL. The same techniques can be applied to ACL, but it is outside of the scope of the present paper.

**Delimited Abort**

The manner in which polarity information determines how cut is reduced with respect to the Unlock rule can be taken advantage of to formulate a delimitation effect\(^2\). Consider

\[
\frac{s : \Gamma; \Delta \vdash e_1 \rightarrow e_2 \quad t : \Gamma; \Delta' \vdash A}{[d]s : [\Delta':A], \Gamma; \Delta \vdash e_2 \quad ? : [\Delta':A], \Gamma; \Delta \vdash e_2} \text{Unlock}
\]

Here, [ ] means [ ]\(_\perp\). With \(e_1\) and \(e_2\) both green, there are two ways to reduce this cut. The first is by usual \(\beta\)-reduction, once \(s\) has been reduced to a lambda-term. A second possibility is to reduce to the following:

\[
\frac{t : \Gamma; \Delta \vdash e_1 \rightarrow e_2}{[d]t : [\Delta':A], \Gamma; \Delta \vdash e_2} \text{Unlock}
\]

With weakening, the same \(t\) still proves the premise. This choice throws away the context \(s\). However, if \(e_2\) was not green, then the only choice is \(\beta\)-reduction. A similar situation exists if the last rule of \(s\) is Unlock (assuming the relaxed version of Unlock):

\[
\frac{s : \Gamma; \Delta \vdash A \quad [d]s : [\Delta':A], \Gamma; \Delta \vdash a \rightarrow e_2 \quad ? : [\Delta':A], \Gamma; \Delta \vdash e_2}{t : \Gamma \vdash a \rightarrow e_2 \quad \text{Unlock}} \rightarrow E
\]

Since \(e_2\) is green (else the implication is not green), the proof remains \([d]s\), this time discarding \(t\). In fact, this time that is the only reduction possible if \(a\) is red. If \(a\) is green and the right subproof also ends in Unlock, then nondeterminism results and an appropriate evaluation strategy/abstract machine will be needed if determinism is required.

Thus within a purely green context, \([d]t\) has the same effect as abort (0-elimination).

We can distinguish a green context such as \(E[y] = (s,y)\) (i.e., \(E[t]\) always has a green type) from red contexts \(R[y]\), which are of red type. Then the following rule would be valid:

\[
\frac{\gamma d. R[E[y]] \rightarrow \gamma d. R[\lfloor d \rfloor t]}{}
\]

In contrast to a term \(A(t)\), which uses 0-elimination, the “break” generated by \([d]t\) cannot escape the entire program but is thrown upwards towards the nearest red context, i.e., the red continuation skips ahead to where \([d]t\) occurs. Let us write a special case of \(\rightarrow\)-elimination (analogously also for \(\leftarrow\)-elimination) marking a switch between green and red contexts:

\[
\frac{u : \Gamma; \Delta \vdash E \rightarrow R \quad v : \Gamma; \Delta' \vdash E}{(u \# v) : \Gamma; \Delta \Delta' \vdash R} \rightarrow \text{Reset}
\]

This rule is intended to supersede all such instances of \(\rightarrow\) introduction. Then we can also write the rule

\[
F[zE[\lfloor d \rfloor t]] \rightarrow F[z[d]t]
\]

Reduction can then be disambiguated by the following definition:

\(^2\)This example also applies to PCL, but was not described in [LM13b]. The details of this section is also somewhat different from the presentation in the conference version of this paper.
\[(d\ u) \mapsto [d\ u]\]

\[(\lambda x. s) \mapsto \text{match } t \text{ with }\]

\[\begin{align*}
| \ [d\ u] & \mapsto [d\ u] \\
| \sharp u & \mapsto s[u/x] \\
| \ v & \mapsto s[v/x]
\end{align*}\]

Note that the delimiter is dropped after substitution, which gives this operator a rather dynamic behavior. For example, \(\lambda x. s\) could be \(\lambda x. \sharp (f\ x)\) where \(f\) is of type \(E \rightarrow E'\) and \(u\) of type \(E' \rightarrow R\). A green abort can escape the context \(f\) as well. We can prove that the usual congruent closure of this reduction relation preserves types (subject reduction) regardless of evaluation order. This is because instances of subterms \((s\ t)\) where \(s\) is of type \(E' \rightarrow R\) are not well-typed: they must be in the form \((s\ \sharp t)\). It is dynamic because one cannot determine which \(\sharp\) will stop the abort without reducing the term.

This example is consistent with the work of Herbelin and Ilik [Her10, Ili12] who showed that delimited control behavior is a result of the transition between an intuitionistic and a classical mode of proof. This example demonstrates the value of mixing classical (green) and non-classical (red) types, although it does not require the affine component of the logic. The next example of delimitation does.

### 7.1 Capturing Delimited Continuations

Besides Unlock, the other structural rule that is sensitive to polarity information is Pr, which cancels the non-contractable context when the stoup is green. Consider the following scenario:

\[
gam\Delta \vdash e_1 \quad \rightarrow \quad \vdash \mathrm{UnLock} \\
\vdash \mathrm{Lock} \\
\vdash \mathrm{Pr^*} \\
\vdash \mathrm{Reset} \\
\vdash \mathrm{Shift-Lock}
\]

If \(A\) is red, then the only way to reduce this cut is to substitute into the left subproof (β-reduction). However, if \(A\) is green, then the cut can be permuted above the \(Pr\), and then above the \(Unlock\). Here is another opportunity for delimited control. We introduce the following rules. This time we shall annotate the \(Pr\) rule more meaningfully, but only in the context of a green to red transition (\(R\) is red):

\[
\begin{align*}
u : \Gamma; \Delta \vdash E & \rightarrow R \\
\vdash \mathrm{Reset} \\
\vdash \mathrm{Shift-Lock}
\end{align*}
\]

The \(Shift-Lock\) rule is characterized by locking a green stoup formula. The context/continuation captured by \(Dk\) is delimited by the nearest \(!x\). Alternatively we can combine \(Shift-Lock\) with \(Pr\), and use the same \(\rightarrow \mathrm{reset}\) rule introduced previously. It should be noted that the continuation captured is in the form \([k](ft)\) and not \(kt\) as expected in the \(Shift\) operator, but the two forms are resolved if a single green type is used. The formulation of \(Shift\) in [Ili12] assumes a similar condition. Compared to that work, green formulas are more general than \(\Sigma\) formulas that may not contain implication and negation. To be fair, however, we have not yet explored all the issues that the wider range of types imply.

To facilitate comparison with existing literature on delimited control operators (e.g. [?]), we can define a call-by-value framework by defining terms, values and evaluation contexts as follows:

- Values \(v = x \mid \lambda x. t\)
- Terms \(t = v \mid (t_1 \ t_2) \mid (t_1 \ \sharp t_2) \mid Dk. t\)
• Evaluation Context \( F = [ | (F \ t) | (v F) | (F \ v) | (v \ vF) \)

• Pure Evaluation Context \( P = [ | (P \ t) | (v P) \)

The reductions rules are:

• \( (\lambda x.t) \ v \rightarrow t[v/x] \)
• \( (\lambda x.t) \ v \rightarrow t[v/x] \)
• \( v \ vP[dk.t] \rightarrow v \ vP[t\{\lambda x.P[x]/k\}] \)

Compared to the formulation of the dynamic Control/Prompt operators, the (pure) context \( P \) is repeated because Lock represents a form of Peirce's formula, not double-negation elimination. The external \( P \) can be dropped using a \( \bot \)-elimination (break) or 0-elimination (abort). In particular, from the validity of the formula \( (E \rightarrow E') \rightarrow \bot \rightarrow E \) we can fashion a variant of Lock that does not require an extra cut. The static Shift/Reset operations, which require the substitution to be in the form \( t\{\lambda x.vP[x]/k\} \), can be simulated by relaxing the mandatory use of \( \bot E \) in place of \( E \). In practice, a green formula \( E \) can always be cast to a red equivalent \( (E \otimes 1) \), although the inverse cast is not generally possible.

This phenomenon cannot be duplicated in linear logic without additional assumptions on how cut elimination should proceed. A classical implication in linear logic is typically \( !A \rightarrow \bot B \), which is equivalent to \( ?(A \otimes B) \). Although \( ? \) formulas are subject to contraction, they have no effect on any other formula. Their meaning is entirely static. The delimited control behavior exhibited above is only possible because of the much stronger \( Pr \) rule, which is enabled by the new meaning of \( \bot \) and the separation between red and green.

### 7.2 Improving the Handling of Exceptions

We shall not enumerate all possible uses of ACL but will give an example to demonstrate the value of combining logics. One application of \( \lambda \mu \) calculus has been to formalize exception handling in programming languages (e.g., see [Cro99]). Essentially, catch is represented by contraction, which saves a copy of the current continuation while throw (raise) is a form of weakening, which discards the current continuation and escapes to the original one\(^3\). Already with PCL, we demonstrated how this kind of formalization can be improved by combining intuitionistic logic with classical logic: in particular, we can specify the type of the exception as well as where in a program an exception can be thrown using the fact that intuitionistic implication enforces scope. In Java, for example, one must specify \( int p(...) \) throws IOException if it is possible for a procedure to throw such an exception. These kinds of constraints are not possible in classical logic. With affine implication, we can further enhance the constraints of exception handling (although enforcing scope is difficult in the continuation passing style; further techniques were used in PCL then what is illustrated below).

In this scenario, green types will represent error conditions that lead to exceptions. In this context, \( \bot \) is analogous to the “superclass” Exception in Java that represents all possible exceptions. The generalized Lock/Unlock rules allow us to specify the type of the exception being handled. All exception types (IOException, ArithmeticException, etc) will be green. The type of the result of any computation that may raise an exception of type \( E \) is specified in the form \( \sim E \rightarrow R \), where \( R \) is the type of the computation without an exception being thrown (and \( \sim E \) is \( E \rightarrow 0 \)). catch and throw can be represented by the following derived rules.

\[
\begin{align*}
s : [\Delta : R^d_E, \Gamma, \Delta \sim E^x \vdash R] & \quad \gamma ds : [\Gamma, \Delta \sim E^x \vdash R] \\
\frac{\text{Lock}_{E} (\text{catch}_{E})}{\text{Unlock}_{E} (\text{throw}_{E})}
\end{align*}
\]

\(^3\)In our context, this weakening will take the form of a 0-elimination (abort).
This example illustrates the working of a unified logic. First, classical contraction is used to save the current continuation and operating context. Next, affine implication allows the presence of ∼E to represent a permission “token” to throw an exception of type E at most once. Finally, the choice of intuitionistic negation in ∼E means that the affine-linear context must be empty when the saved continuation is unlocked. In other words, when an exception is thrown, the current mutable context of operation (Θ) is discarded and the saved one (∆) is restored along with the saved continuation (R).

In other words, when an exception is thrown, the current mutable context of operation (Θ) is discarded and the saved one (∆) is restored along with the saved continuation (R). Note that ∆ may contain permission tokens to throw other kinds of exceptions. For example, there may be [∆₁, ∼e₁ : A] and [∆₂, ∼e₂ : B] saved in the set context, and one exception can be thrown after the other. We can also select a different behavior by using ∼E (E → 0) in place of ∼E. This will merge the saved context with the current one.

A procedure that would normally be of type A → B, but which may throw an exception of type E, will instead have type ∼E ↠ A → B (equivalent to A ↠ ∼E ↠ B). We note that ↠-introduction is an invertible rule, and thus the presence of ∼E need not interfere with the normal flow of computation:

\[
\frac{s : \Gamma; \Delta \vdash E \rightarrow A \rightarrow B \quad x : \vdash E \rightarrow E \rightarrow E}{(s \cdot x)t : \Gamma; \Delta, \vdash A \rightarrow B}
\]

Without restricting contraction, the execution of a procedure above the throw can again throw the exception: we wish to control whether this is allowed. Without the affine-linear →, we will need to require that x does not appear free in t to enforce the condition. In other words, we would have to require externally that the “classical proof” is actually a linear one. The availability of weakening in affine logic is also crucially important in this example: we certainly do not wish to say that a procedure must throw an exception.

In contrast, with linear logic (proper or affine), the different features of this example are difficult to combine. Contraction requires ? to be placed before formulas. Structural rules cannot be dynamically restricted to fragments of proofs. We would not be able to distinguish the exception handling, classical phase of computation from the normal, non-classical phase. The linear logic proof would not have much more constraints than a classical proof. In ACL proofs, the necessary contractions are only allowed when the green subformula E is encountered in the stoup. There is no need to place a ? before every formula that may be contracted anywhere in the proof. The strengths of the non-classical connectives are not compromised when the stoup is red. There is no known translation of ACL into linear logic, and we have reason to believe that there cannot be one, as there is no way to capture the enriched meaning of the constant ⊥ from which all green formulas derive their properties.

7.3 Concurrent Programming with fork and yield

This section is experimental ...

We can generalize the idea used in the exception handling example to formulate a new control operator, a guarded call/cc:

\[K_g = X x X y z . gamma x . (x y . A (z \cdot [d] y)) \quad \vdash ((P \rightarrow Q) \rightarrow P) \rightarrow (\sim E \rightarrow P)\]

Both P and Q can be of arbitrary polarity, but ∼E is not superfluous as in classical logic. Although items [Θ : A] are stored in the contractable set context, we can control Unlock\(_E\) using a non-reusable token ∼E or ∼E. Since ⊥ ⇐ E always holds, ∼E represents a one-time assumption that ⊥ and 0 are equivalent, causing a local collapse to classical logic. Similarly, a guarded version of the C operator is

\[C_g = X x X y z . gamma d . A (x \cdot X y . z \cdot [d] y) \quad \vdash \sim A \rightarrow (\sim E \rightarrow A)\]

Another version uses intuitionistic implication, which forces an emptying of the multiset context before Unlock:

\[lambda x X z . gamma d . A (x (lambda y . z ([d] y))) \quad \vdash \sim A \rightarrow (\sim E \rightarrow A)\]

\footnote{we can translate proofs, but cannot form synthetic (localized) connectives to emulate ACL formulas}

24
These operators can in fact form the basis of a concurrent programming interpretation of proofs. Some of the ideas here are derived from [MZ10], which uses classical linear logic and a pseudo-control operator. Their formulation uses some fortuitous interpretations of linear logic, assigning extra meaning to $P^\perp \perp \leadsto P$. With ACL we have the benefit of real control operators, based on non-involutive negations, which can be combined with affine restrictions.

The fork operation such as found in Unix splits a process into parent and child copies. The yield operation, called from the parent, suspends the current computation (after possibly saving it first), then waits for a child to complete and receives a value computed by the child process. Conceptually, fork can be represented by a form of contraction, but to control the contraction we use $\sim e$ or $\sim e$. The affine token effectively enforces that only one child is created by each fork. Thus $K_g$ is the basis of fork. What about yield? Let us think of the parent process that waits for the child as an “incomplete” proof. Yet it still needs to be a well typed term. The form of double-negation elimination represented by the $C_g$ operator is the basis of yield: when the parent is waiting for a value from the child, it can use a fictitious proof of $\sim \sim A$ in the meantime. The subterm $x y z \cdot [d] y$ is telling. The bound variable $y$ has type $A$: conceptually speaking, this term is waiting for a real proof of $A$ from the child process. With this intuition, we now write fork and yield as the following synthetic rules.

$$
\begin{align*}
\text{fork} & \quad s_e : \Gamma; \Delta \vdash A \\
& \quad s_p : [: A]_e^d, \Gamma; \sim e^\perp, \sim \sim A^\perp, \Delta \vdash A \\
& \quad \nu d, x, (s_p^d | s_p) : \Gamma; \Delta, \sim e^\perp \vdash A \\
& \quad (d, x, z)(y) : [: A]_e^d, \Gamma; \sim e^\perp, \sim \sim A^\perp, \Theta \vdash B \\
& \quad \text{yield}
\end{align*}
$$

In the fork rule, $s_e$ represents the child process while $s_p$ is the parent. $s_e$ is indexed by $d$ so as to correspond to its parent. $\nu$ binds both $d$ and $x$. It is assumed that $e$ does not appear as a subformula elsewhere in the sequent. The fork rule is logically sound since the conclusion follows from the left premise; the right premise is always provable. The yield rule is initial; its proof term is “complete.” The term $(d, x, z)(y)$ replaces $A(x \cdot x y z \cdot [d] y)$. $y$ is bound in $(d, x, z)(y)$.

Of course, in order for the program to succeed, fork must be called at the right point in the program. Otherwise, enough resources may not have accumulated in $\Delta$ and $\Gamma$ for the child process to complete. Not all programs are parallelizable.

For evaluation rules, we can distinguish values $v$ (some weak normal form), terms $t$ and contexts $F[\ ]$ in the usual way. Then we can write the following rewrite rules:

$$
\begin{align*}
& F[v d, x, (t_1 | t_2)] \rightarrow v d, x, (F[t_1] | F[t_2]) \\
& v d \mid F[(d, x, z)(y)] \rightarrow \gamma d.F[A(z \cdot [d] v)]
\end{align*}
$$

The first rule allows for parallel execution of the two processes, which share the same evaluation context (cuts are made in parallel as opposed to sequentially). The second rule allows for communication once the child has evaluated to a value, at which point the child process is terminated and the parent is rewritten in the sequential form using $\gamma$ instead of $\nu$. The initial yield is replaced by a more interesting proof term.

To see that the evaluation rules are correct, let us write a more succinct instance of the first rule, this time annotating some of terms with their types. Assume that context $F[y]$ is just $(y u)$:

$$
(v d : (A \rightarrow B), x t_1 t_2) u : A \rightarrow v d : B, x, (t_1 u)(t_2 u)
$$

During the course of normalization the variable $d$ is reused after the proof is transformed. However, the term representation stays the same. Thus the value communicated back to the parent process will be of the correct type.

Finally, we would like to make a comparison between the semantically motivated approach of ACL and the mostly syntactic approach of focusing (focalization). The notion of polarization as used in focused proofs is also found in LC and its generalization LLP [Lau02]. It is based on dualities found in the context of linear logic. This form of positive/negative polarization is also a means for controlling the behavior of classical
A proving Peirce’s formula, or with formulas of the form ∼ based on focusing). In particular, implication is a decisively negative connective. Focusing is of no help in quite enough to form a unified logic with the desired properties (see our previous attempt, LKU, which is where Θ the affine-linear context and Γ and Θ are sets representing the intuitionistic context.

As we have shown with PCL, such a proof system can also offer a greater range of computational possibilities. In this section we adopt a multiple conclusion sequent calculus, which reveals further properties about ACL.

To consider other forms. Furthermore, focusing is only defined for a particular form of sequent calculus. We may wish principally to control cut elimination in proofs that can be represented in linear logic (focusing is a

Unlock focusing that forces one premise of _ as a synthetic rule (starting with _

principally serve to control cut elimination in proofs that can be represented in linear logic (focusing is a consequence). Furthermore, focusing is only defined for a particular form of sequent calculus. We may wish to consider other forms.

8 Multiple Conclusion Alternative

In this section we adopt a multiple conclusion sequent calculus, which reveals further properties about ACL. As we have shown with PCL, such a proof system can also offer a greater range of computational possibilities.

Sequents of the multiple-conclusion MAC have the form Γ; Δ ⊩ Θ, where Δ is a multiset representing the intuitionistic context and Γ and Θ are sets representing the intuitionistic context. A, Γ and A, Θ does not preclude the possibility that A ∈ Γ (or Θ). Sequents have the meaning of the formula Γ^ ^ → Δ° → Θ^ where Θ^ is the ∨-disjunction of formulas in Θ, with an empty Θ representing 0. LMU is found in Figure 3.

The validity of Lock in this system is based on the fact that

((-A → B) → (A → (B ∨ C))) → (A → (B ∨ C))

is another valid generalization of Peirce’s formula. To see this semantically, construct a “truth table” with choices r ∈ (A → B)^ and r /∈ (A → B)^ . The first case means A^ ⊆ B^ so of course A^ ⊆ B^ ∪ C^ . The

5The D rule in LC that moves a formula into the stoup is focused in the same way if we look at its translation to intuitionistic logic.
second case implies that \( r \in -(A \rightarrow B)^* \) so \( -(A \rightarrow B)^* = 1^* \) and thus \( -(A \rightarrow B) \rightarrow (A \rightarrow (B \lor C)) = (A \rightarrow (B \lor C))^* \). The validity of Unlock depends on the fact that, when any green formula is in the right-hand side \( \Theta \), then \( \Theta^\forall \) is green. Given a green formula \( B \lor E \), \( A \rightarrow (B \lor E) \) collapses into a classical implication \( -A \lor B \lor E \). This means that \( A \rightarrow (B \lor E) \) is equivalent to \( (A \rightarrow B) \lor (A \rightarrow E) \).

There are aspects of this sequent calculus that someone versed in the conventions of linear logic might find strange. For example, in \( \otimes R \) we split the left-side linear context \( \Delta \), but not \( \Theta \). In a tableau interpretation of these proofs, formulas on the right-hand side of a sequent are those that are signed \( F \), i.e., not forced at the current world (which is roughly represented by the left-side of the sequent). Thus by the contrapositive of monotonicity, they’re also not forced in worlds beneath the current world. One might also observe that the \( \land L \) rule is still additive while the \( \lor R \) rule becomes multiplicative: a mismatch based on what linear logic usually teaches us. But this is due to the fact that we wish to write only one copy of the inference rules: we can always move formulas from \( \Gamma \) to \( \Delta \). If the \( \land L \) rule was written so that the principal formula was in \( \Gamma \), then it can be multiplicative.

The rule for \( \rightarrow R \) is also revealing. Since \( u \in A \rightarrow B \) if whenever \( v \in A \), \( uv \in B \). But \( u \preceq uv \), thus we’re always moving to a possibly future world: i.e., to show \( u \notin A \rightarrow B \), we need to show \( uv \notin B \). Thus our affine implication preserves scope just as intuitionistic implication does. In proving \( (A \rightarrow B) \lor C \), one cannot use the assumption \( A \) to prove \( C \). However, without monotonicity (upward closure) and the property that \( u \preceq uv \), then the scoping restriction cannot be justified.

The soundness of MAC is easily checked semantically: interpret the right side context \( \Theta \) as a \( \lor \)-disjunction of formulas, with an empty \( \Theta \) representing 0. Completeness is obvious since multiple conclusions subsume single conclusions.

8.1 More Aggressive Curry-Howard

The sequent calculus LAC and its natural deduction counterpart are rather conservative since they target the technical results of cut elimination and completeness. In this section we preliminarily introduce a more aggressive representation of proofs, which can be found in Figure 4.

9 Conclusion

Let us summarize the important components of ACL as follows.

1. A phase semantics that approximate the Kripke semantics of intuitionistic logic. Facts are upwardly closed sets. The constant \( \perp \) is the second-largest fact, with more attributes than its counterpart in linear logic.

2. The polarization of formulas into green (classical) and red (possibly non-classical). The polarity of arbitrary formulas reduce to the polarity of atoms and constants, in particular to the green \( \perp \).

3. The validity of the Peirce-like formula \( -(P \rightarrow P) \rightarrow P \) and its dual, \( (P \rightarrow -P) \rightarrow -P \). They enable contractions on arbitrary formulas when \( \perp \), or any green formula, is encountered as the current, or stoup formula in a proof. This duality replaces \? and \! in allowing restrictions on contraction to coexist with cut elimination.

4. A sound and complete sequent calculus that enables contractions dynamically. The classical effect of green formulas is localized in proof segments. The cut-elimination procedure for this proof system includes \( \lambda \mu \)-style structural reductions.

5. From these elements we derive a Curry-Howard interpretation of natural deduction proofs that allows intuitionistic and affine-linear lambda terms to coexist with control operators such as \textit{call/cc}.

The logical interpretation of the computational content of proofs that use contractions on the right-hand side thus does not require a collapse into classical logic.
Figure 4: Natural Deduction System NACM
References


