Project Report: Logic Programming in Constructive Provability Logic

15-816 Modal Logic

Robert J. Simmons, Bernardo Toninho

May 9, 2010

Abstract

We present a novel formulation of **CPL**, a *constructive logic of provability* that is closely connected to the Gödel-Löb logic of provability. Our logical formulation allows modal operators to talk about both *provability* and *non-provability* of propositions at reachable worlds. We use this logic as a basis for a discussion of negation in logic programming.

1 Introduction

Consider the following propositions:

$$\begin{aligned} \mathsf{edge}(X,Y) &\supset \mathsf{edge}(Y,X) \\ \mathsf{edge}(X,Y) &\supset \mathsf{path}(X,Y) \\ \mathsf{edge}(X,Y) \supset \mathsf{path}(Y,Z) &\supset \mathsf{path}(X,Z) \end{aligned}$$

By treating these propositions as a *bottom-up logic program*, we can take a description of the edge relation and compute its symmetric, transitive closure as the path relation by making repeated forward inferences. Once the only facts we can derive are facts we already know — for instance, once we have $\mathsf{edge}(\mathsf{a},\mathsf{b})$, $\mathsf{edge}(\mathsf{b},\mathsf{a})$, $\mathsf{path}(\mathsf{a},\mathsf{b})$, $\mathsf{path}(\mathsf{b},\mathsf{a})$, $\mathsf{path}(\mathsf{a},\mathsf{a})$, and $\mathsf{path}(\mathsf{b},\mathsf{b})$ — we say we have reached *saturation*.

Next, consider the following proposition:

$$\mathsf{path}(X,Y) \supset \neg\mathsf{edge}(X,Y) \supset \mathsf{noedge}(X,Y)$$

Intuition says that this is meaningful and that, given our example, we should be able to prove, for instance, $\mathsf{noedge}(\mathsf{a}, \mathsf{a})$ and $\mathsf{noedge}(\mathsf{b}, \mathsf{b})$. A bottom-up logic programming semantics based on *stratified negation* verifies this intuition. In a stratified logic program such as this one, we derive all the consequences of the first three rules until saturation is reached. At this point, we know that our database of facts contains everything there is to know about the edge and path relations. When considering the negated premise $\neg\mathsf{edge}(X, Y)$ in the fourth rule, we simply check the saturated database and conclude that the premise holds if the fact does not appear in the database.

Stratified negation would, however, disallow the addition of the following rule as paradoxical or contradictory:

$$\mathsf{path}(X, Y) \supset \neg\mathsf{edge}(X, Y) \supset \mathsf{edge}(X, Y)$$

In this report, we consider a constructive modal logic that can be used to give meaning to stratified negation. We call this logic **CPL**, for *constructive provability logic*, for two reasons. The first is its connection to the Gödel-Löb modal logic of provability, and the second is the fact that our modal operators give us the ability to reflect on logical provability at accessible worlds. We will discuss both of these points in turn.

Relationship to the provability logic of Gödel-Löb One of the critical consequences of the design of our logic is that the accessibility relation must be *converse well-founded* (defined in Section 2.1). This is a bit of an odd requirement, and we are aware of no axiom scheme that gives a characterization of this class of Kripke structures. However, when combined with the fairly pedestrian requirement that the accessibility relation be *transitive*, the resulting logic validates the axioms of the Gödel-Löb logic of provability, which is known variously as **GL**, **Pr**, **KW**, and **K4W** and is characterized by the following three axioms:

- K: $\Box(A \supset B) \supset \Box A \supset \Box B$,
- 4: $\Box A \supset \Box\Box A$,
- and GW: $\Box(\Box A \supset A) \supset \Box A$.

KW is an important logic for a number of reasons, and has been previously used for investigations of negation in logic programming by Gabbay in [Gab91]. Gabbay argued that a modal interpretation of logic programs must be based on either **KW** or a temporal modal logic; he then chose the former because it intuitively corresponds with the idea that logic programming is about a search strategy for proofs.

While we will focus in this report on the use of **CPL** in the study of logic programming, the connection between **CPL** and **KW** is also interesting because necessitation in **KW** is essentially the same as the *approximation modality* that can be used to give semantic models to programming languages with recursive types and other traditionally difficult-to-model language features [Ric10, AMRV07].

Logical reflection on provability The logic **CPL** that we present combines a familiar way of reasoning about truth at any particular world with a powerful ability to reason about *provability* at accessible worlds.

Existing sequent calculi for modal logic, such as Pfenning-Davies S4 [PD01] or Simpson's intuitionistic Kripke semantics [Sim94], treat an assumption "$\Box A$ is true at world w" as license to assume that A is true at all worlds w' accessible from w. In our logic, however, the assumption "$\Box A$ is true at world w" does not act as an assumption that A is true at all accessible worlds so much as it acts as an *assertion that A is provable* at all accessible worlds. In particular, if we have an assumption "$\Box A$ is true at world w" and there is some w' accessible from w where A is *not* provable using the currently assumed facts about w', then the assumption $\Box A$ is *contradictory* and can be used (like any other form of contradiction) to conclude any fact at w. Given the ability to reflect over provability at accessible worlds, it is then natural to consider a modality for *non-provability* at accessible worlds — it is this modality that can be used as an explanation for stratified negation in logic programming.

1.1 Outline of this report

This report has two parts. Our primary contribution, the presentation and machine-checked formalization of **CPL** and its metatheory, can be found in Section 2. In Section 3 we discuss the use of **CPL** as a logical justification for constructive negation in logic programming, and in Section 4 we conclude with a substantial discussion of future work suggested by our investigations.

2 The constructive provability logic CPL

In this section, we propose a constructive provability logic which we call **CPL**. The distinguishing feature of **CPL** is that it allows for a proposition at any particular world to reflect over not truth, but *provability* at accessible worlds.

In the following two sections we motivate and present **CPL** as an odd sort of intuitionistic modal logic that, when given any transitive Kripke frame, validates the axioms of the Gödel-Löb logic of provability. Then in Section 2.3 we will present the full propositional sequent calculus for **CPL**, including a modal operator for constructive negation and connectives capturing provability and non-provability at specific worlds. In Section 2.4 we discuss the metatheory of **CPL**, and in Section 2.5 we discuss the formalization of this metatheory in the dependently-typed programming language Agda [Nor07].

2.1 Basic CPL

The genesis of **CPL** was an exploration of constructive negation in modal logic, and we considered and took ideas from the study of Judgmental S4 [PD01], adjoint logic [Ree09], intuitionistic Kripke semantics [Sim94], and tethered modal logics [Pfe10]. The result is a system that is quite different from any of these existing systems.

Towards a logic of provability Consider the introduction rule defining modal $\Diamond$ in a Simpson-style intuitionistic Kripke semantics (we write "$w_1 \prec w_2$" to indicate that the world w_2 is accessible from the world w_1):

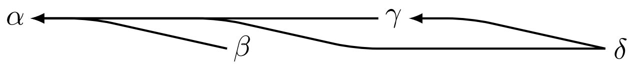
$$\frac{w \prec w' \quad \Gamma \vdash A[w']}{\Gamma \vdash \Diamond A[w]}\ \Diamond_R$$

Following Simpson, we can give a left rule for $\Diamond$ that posits a new future world where A is true:

$$\frac{\Gamma, \Diamond A[w], \alpha\ world, w \prec \alpha, A[\alpha] \vdash J}{\Gamma, \Diamond A[w] \vdash J}\ \Diamond_{LI}$$

The introduction rule is in harmony with the elimination rule in the sense that the logic is sound (given a proof of $\Diamond A[w]$ and another proof that uses the assumption $\Diamond A[w]$ to prove something else, there's a proof of that something else that doesn't use the assumption) and complete (an assumption of $\Diamond A[w]$ can be used to prove $\Diamond A[w]$). However, these standard metatheoretic results, while absolutely necessary, are not always sufficient to ensure that we are satisfied with our logical definition.

As an example of where our dissatisfaction might arise, consider the following accessibility relation (where the arrows point to accessible worlds, so $\delta \prec \gamma$ and $\beta \prec \alpha$):



Given this setting, a Simpson-style modal logic provides no way of proving the following sequent:

$$\Diamond A[\delta], A \supset B[\alpha], A \supset C[\gamma] \vdash \Diamond(B \wedge C)[\delta]$$

The sequent is unprovable because the assumption $\Diamond A[\delta]$ can only be used to posit some new world δ' and generate the additional hypothesis that $\delta \prec \delta'$ and that A is true at δ'. But this hardly seems fair! The assumption $\Diamond A[\delta]$ means that A is true at some world accessible from δ, and we know what worlds are accessible from δ: there are two of them, α and γ. If A is true at α then we can prove the sequent...

$$\dfrac{\dfrac{}{\ldots \vdash A[\alpha]}\;\mathit{hyp} \qquad \dfrac{\delta \prec \alpha \quad \dfrac{\dfrac{}{\ldots, B[\alpha] \vdash B[\alpha]}\;\mathit{hyp}}{\ldots, B[\alpha] \vdash B \vee C[\alpha]}\;\vee_{R1}}{\ldots, B[\alpha] \vdash \Diamond(B \vee C)[\delta]}\;\Diamond_R}{A[\alpha], A \supset B[\alpha], A \supset C[\gamma] \vdash \Diamond(B \vee C)[\delta]}\;\supset_L$$

...and if A is true at γ the we can also prove the sequent...

$$\dfrac{\dfrac{}{\ldots \vdash A[\gamma]}\;\mathit{hyp} \qquad \dfrac{\delta \prec \gamma \quad \dfrac{\dfrac{}{\ldots, C[\gamma] \vdash C[\gamma]}\;\mathit{hyp}}{\ldots, C[\alpha] \vdash B \vee C[\gamma]}\;\vee_{R2}}{\ldots, C[\alpha] \vdash \Diamond(B \vee C)[\delta]}\;\Diamond_R}{A[\gamma], A \supset B[\alpha], A \supset C[\gamma] \vdash \Diamond(B \vee C)[\delta]}\;\supset_L$$

So why can we not combine these two facts to prove $\Diamond(B \vee C)[\delta]$ from the assumption $\Diamond A[\delta]$? Simpson's intuitionistic Kripke semantics demand not just provability but *uniform* provability, and the reasoning above is *non-uniform*: essentially, we perform a case analysis on the accessibility relation and return an entirely different proof in each branch. If we want to express a non-uniform rule, we can write the rule as follows:

$$\frac{\forall w'.(w \prec w') \Rightarrow \Gamma, \Diamond A[w], A[w'] \vdash J}{\Gamma, \Diamond A[w] \vdash J}\ \Diamond_{LE}$$

The premise of $\Diamond_{LE}$ demands not a derivation but a *function* from proofs that there is some w' such that $w \prec w'$ to proofs that $\Gamma, \Diamond A[w], A[w'] \vdash J$. We generally think of such functions as being defined by case analysis with a separate branch for each w' accessible from w. However, this is only a rough analogy: there is no reason that w could not have an infinite number of successors, which would mean the "case analysis" would have an infinite number of branches (and would realistically have to be defined in some other way, perhaps parametrically as in $\Diamond_{LI}$ or perhaps by induction, if worlds are represented by something like natural numbers).

We moved from an uniform modal logic to an non-uniform modal logic when we elevated the process of reasoning about the accessibility relation from an hypothesis *within* the logic to an assumption *about* the accessibility relation. We move from an intuitionistic modal logic to a constructive logic of provability when we do the same for provability. Instead of adding the hypothesis $A[w']$ as in rule $\Diamond_{LE}$, we can consider the following rule:

$$\frac{\forall w'.(w \prec w') \Rightarrow (\Gamma \vdash A[w']) \Rightarrow \Gamma, \Diamond A[w] \vdash J}{\Gamma, \Diamond A[w] \vdash J}\ \Diamond_{LX}$$

As it turns out, this rule is unreasonable: $\Gamma \vdash A[w']$ is a *negative occurrence* of the judgment $\Gamma \vdash J$, which is disallowed when we are defining what $\Gamma \vdash J$ means in the first place. We can solve this problem if the accessibility relation is *converse well-founded* (sometimes called *upwards well-founded*), meaning that there is no infinite ascending chain $w_1 \prec w_2$, $w_2 \prec w_3$, $w_3 \prec w_4$, …in the accessibility relation. In this case, we can consider the meaning of $\Gamma \vdash A[w']$ to be entirely established *before* we consider the meaning of $\Gamma \vdash A[w]$. Using this observation, we can correct the broken rule $\Diamond_{LX}$ by tethering the premise to the conclusion:

$$\frac{\forall w'.(w \prec w') \Rightarrow (\Gamma \vdash A[w']) \Rightarrow \Gamma, \Diamond A[w] \vdash C[w]}{\Gamma, \Diamond A[w] \vdash C[w]}\ \Diamond_{L}$$

We can now give a sequent calculus presentation of **CPL** (Fig. 1). The right and left rules for $\Diamond$ are those that follow from the previous discussion.

$$\frac{}{\Gamma, Q[w] \vdash Q[w]}\ hyp \quad (Q \text{ is an atomic proposition})$$

$$\frac{\Gamma, A[w] \vdash B[w]}{\Gamma \vdash A \supset B[w]} \supset_R \quad \frac{\Gamma, A \supset B[w] \vdash A[w] \quad \Gamma, A \supset B[w], B[w] \vdash C[w]}{\Gamma, A \supset B[w] \vdash C[w]} \supset_L$$

$$\frac{\forall w'.(w \prec w') \Rightarrow (\Gamma \vdash A[w'])}{\Gamma \vdash \Box A[w]} \Box_R$$

$$\frac{(\forall w'.(w \prec w') \Rightarrow (\Gamma \vdash A[w'])) \Rightarrow \Gamma, \Box A[w] \vdash C[w]}{\Gamma, \Box A[w] \vdash C[w]} \Box_L$$

$$\frac{w \prec w' \quad \Gamma \vdash A[w']}{\Gamma \vdash \Diamond A[w]} \Diamond_R \quad \frac{\forall w'.(w \prec w') \Rightarrow (\Gamma \vdash A[w']) \Rightarrow (\Gamma, \Diamond A[w] \vdash C[w])}{\Gamma, \Diamond A[w] \vdash C[w]} \Diamond_L$$

Figure 1: Sequent Calculus for **CPL**

Similarly, the right rule for $\Box$ has as its premise a function that, for any successor of w', produces a proof of A at that world. The left rule for $\Box$ may be surprising: it states that we are justified in using an assumption $\Box A[w]$ to conclude proposition C at world w if we can produce a higher-order function that, given a function that for any successor of w provides a proof of A at that world, produces a proof of C.

The usual interpretation for a left rule for $\Box$ is that it should capture the non-uniform interpretation of a universal quantifier:

$$\frac{w \prec w' \quad (\Gamma \vdash A[w']) \Rightarrow (\Gamma, \Box A[w] \vdash C[w])}{\Gamma, \Box A[w] \vdash C[w]} \Box_{L0}$$

However, it turns out that such a rule would be very nearly too weak in terms of the metatheory of our logic, further requiring our accessibility relation to be finitely branching. We thus adopted this version of $\Box_L$ since not only does it cleanly handle infinitely branching accessibility relations, but also because it is equivalent to the less surprising $\Box_{L0}$ (assuming a finite branching accessibility relation).

2.2 Axiomatic CPL and KW

In this section, we attempt to clarify what kind of modal logic **CPL** actually is. An upwards well-founded accessibility relation cannot be reflexive, so the axiom T, which characterizes reflexivity in the accessibility relation, will not (and cannot) hold. However, under the condition that the accessibility relation is transitive (that is, if axiom *4* holds), the previously mentioned axiom GL characterizing **KW** holds.

A key concept for the proof of Theorem 1 (and for some of our metatheoretical results) is that since we know the accessibility relation to be predetermined and upwards well-founded, we can prove properties of our logic by using a induction principle over the accessibility relation itself; in fact, the way we require the accessibility relation to be upwards well-founded is by forcing it to admit just such an induction principle.

Definition 1 (Upwards well-founded accessibility relation)**.** *The accessibility relation $w \prec w'$ is upwards well founded if, for any property $P(w)$, we can conclude that $P(w)$ is true at every world if we know that, at every world w, $P(w)$ is true under the assumption that $P(w')$ is true at every w' such that $w \prec w'$.*

Theorem 1. *For all Γ, A, B, C, and w, the following hold:*

- MP — $\Gamma \vdash A[w]$ *and* $\Gamma \vdash A \supset B[w]$ *implies* $\Gamma \vdash B[w]$.
- $\supset S$ — $\Gamma \vdash (A \supset B \supset C) \supset (A \supset B) \supset A \supset C[w]$.
- $\supset K$ — $\Gamma \vdash A \supset B \supset A[w]$.
- $\supset I$ — $\Gamma \vdash A \supset A[w]$.
- $\Box K$ — $\Gamma \vdash \Box(A \supset B) \supset \Box A \supset \Box B[w]$.
- *4* — *If the accessibility relation is transitive,* $\Gamma \vdash \Box\Box A \supset \Box A[w]$.
- GL — *If the accessibility relation is transitive,* $\Gamma \vdash \Box(\Box A \supset A) \supset \Box A[w]$.

Proof. By induction on the accessibility relation. As an example, *modus ponens* is derivable without recourse to the induction hypothesis at any world by using cut, identity, and weakening (which we discuss in Section 2.4). We can then prove $\Box K$ by applying $\supset_R$ twice, $\Box_L$ twice, $\Box_R$ and *modus ponens* at the accessible world introduced by the $\Box_R$ rule.

Not unexpectedly, the only essential use of the induction hypothesis is in the proof of axiom GL, which is itself an induction principle! □

Theorem 1 also serves to establish the necessitation rule: if A is provable from modus ponens and the axioms of either **K** or **GL** (and if, in the latter case, the accessibility relation is transitive), then for all Γ and w, $\Gamma \vdash A[w]$. But this also means that $\Gamma \vdash A[w']$ at every world w' accessible from w, so $\Gamma \vdash \Box A[w]$.

$$\frac{}{\Gamma \vdash \top[w]}\ \top_R \qquad \frac{\Gamma \vdash A[w] \quad \Gamma \vdash B[w]}{\Gamma \vdash A \wedge B[w]}\ \wedge_R$$

$$\frac{\Gamma, A \wedge B[w], A[w] \vdash C[w]}{\Gamma, A \wedge B[w] \vdash C[w]}\ \wedge_{L1} \qquad \frac{\Gamma, A \wedge B[w], B[w] \vdash C[w]}{\Gamma, A \wedge B[w] \vdash C[w]}\ \wedge_{L2}$$

$$\frac{}{\Gamma, \bot[w] \vdash C[w]}\ \bot_L \qquad \frac{\Gamma \vdash A[w]}{\Gamma \vdash A \vee B[w]}\ \vee_{R1} \qquad \frac{\Gamma \vdash B[w]}{\Gamma \vdash A \vee B[w]}\ \vee_{R2}$$

$$\frac{\Gamma, A \vee B[w], A[w] \vdash C[w] \quad \Gamma, A \vee B[w], B[w] \vdash C[w]}{\Gamma, A \wedge B[w] \vdash C[w]}\ \vee_L$$

Figure 2: Sums and products in **CPL**

2.3 Full CPL

Having presented the core of the modal logic **CPL**, we can extend it straightforwardly with all the other standard connectives of a propositional logic: conjunction, disjunction, and their units. This is done in Figure 2, and it offers no surprises. More interestingly, we can define new connectives using **CPL**'s ability to reason about provability at accessible worlds. The first such connective is a modality of constructive negation similar to $\Box$.

Consider our justification of $\Box A$ at a world w. The proposition $\Box A$ is true in **CPL** at world w if A is provable in all worlds accessible from w. We will define constructive negation in a similar manner: $\neg A$ is true at a world w if, for every world accessible from w, a proof of A entails a metalevel contradiction (written as 0):

$$\frac{\forall w'.(w \prec w') \Rightarrow (\Gamma \vdash A[w']) \Rightarrow 0}{\Gamma \vdash \neg A[w]}\ \neg_R$$

Similarly to $\Box_L$, the left rule for negation uses a higher-order function in its premise.

$$\frac{(\forall w'.(w \prec w') \Rightarrow (\Gamma \vdash A[w']) \Rightarrow 0) \Rightarrow \Gamma, \neg A[w] \vdash C[w]}{\Gamma, \neg A[w] \vdash C[w]}\ \neg_L$$

We understand $\neg_L$ as follows: we can use $\neg A[w]$ to conclude $C[w]$ if we can produce a function that, given a function that derives a contradiction from A being true at any world accessible from w, can produce a proof of C.

As we will see, the metatheory that ultimately shows that our logic is reasonable is compatible with this interpretation of negation. We believe that this leads to a very general and satisfyingly proof-theoretic treatment of negation from a logic programming point of view, as discussed in Section 3.

Provability at particular worlds Our $\Box$ and $\Diamond$ modalities deal with provability at all accessible worlds and at one arbitrary accessible world, respectively. Similarly, our interpretation of negation deals with a lack of provability at all accessible worlds. Describing logic programming with the modal operators defined so far is not especially natural, since we generally think of an atomic proposition as being provable (or disprovable) in *one particular* way. We could map atomic propositions onto a linear accessibility relation and use the repeated application of $\Diamond$ to point back to the correct world whenever we need to refer to truth or falsehood of a fact that lives at the (eventually) accessible world. This, however, is not a particularly satisfying explanation.

A better solution is obtained by introducing two new propositions: $A \mathbin{@} w$ and $A \mathbin{\not@} w$. The former refers to provability of A at a particular accessible world w, and the latter refers to non-provability of A at the accessible world w. The rules giving meaning to these propositions are straightforward (we use $\Gamma \nvdash A[w]$ as shorthand notation for $(\Gamma \vdash A[w]) \Rightarrow 0$):

$$\frac{w \prec w' \quad \Gamma \vdash A[w']}{\Gamma \vdash A \mathbin{@} w'[w]}\ @_R \qquad \frac{w \prec w' \quad (\Gamma \vdash A[w']) \Rightarrow (\Gamma, A \mathbin{@} w'[w] \vdash C[w])}{\Gamma, A \mathbin{@} w'[w] \vdash C[w]}\ @_L$$

$$\frac{w \prec w' \quad \Gamma \nvdash A[w']}{\Gamma \vdash A \mathbin{\not@} w'[w]}\ \not@_R \qquad \frac{w \prec w' \quad (\Gamma \nvdash A[w']) \Rightarrow (\Gamma, A \mathbin{\not@} w'[w] \vdash C[w])}{\Gamma, A \mathbin{\not@} w'[w] \vdash C[w]}\ \not@_L$$

Using these connectives, programs can then be annotated with references to appropriate worlds. We postpone a further discussion of these connectives to Section 3.1, and proceed with a presentation of the metatheory of **CPL**.

2.4 Metatheory of CPL

So far, we have motivated and presented all the rules of our logic. Before proceeding to the application of **CPL** in logic programming, we must first show that the logic is indeed a reasonable logic. The usual way to do this is by exploring its metatheoretic properties, ultimately showing that we have developed a logic that is indeed globally sound and complete, which follows from the metatheoretical properties of admissibility of Identity and Cut.

All the results in this section were implemented in Agda with contexts represented as lists of assumptions. This fact accounts for several of the slightly non-standard features of our development such as the definition of $\Gamma \sim \Gamma'$ *at* w.

2.4.1 Irrelevance of hypotheses at unreachable worlds

The logic we defined allows the context to contain propositions at arbitrary worlds, and the existence of constructive negation as a modal operator means that weakening cannot hold at accessible worlds: if we have $\Gamma \vdash \neg A[w]$ and $w \prec w'$, then $\Gamma, A[w'] \vdash \neg A[w]$ is *not* true! But what about hypotheses at *inaccessible* worlds? When we introduced our system of reflecting over provability at other worlds, we said that in order for provability at w to be defined, we must first fix the definition of provability at all worlds accessible from w. Therefore, if $w \prec w'$, assumptions at world w should not be able to have any affect whatsoever on provability at world w'. The irrelevance theorem verifies the intuition that if $w \prec w'$ then there is a proof of $\Gamma \vdash A[w']$ if and only if there is a proof of $\Gamma, A[w] \vdash A[w']$.

We develop the proof of irrelevant weakening and strengthening by first defining a notion of equivalence of contexts at a given world.

Definition 2 (Context Equivalence)**.** *Context equivalence at a world w, written $\Gamma \sim \Gamma'$ at w, is inductively defined by the following rules, where $w \not\prec w'$ denotes that there does not exist any chain of accessible worlds $w \prec w_1 \ldots w_n \prec w'$.*

- *$\cdot \sim \cdot$ at w, for any w.*
- *If $\Gamma \sim \Gamma'$ at w and $w \not\prec w'$ then $\Gamma \sim (\Gamma', A[w'])$ at w, for any A and w'.*
- *If $\Gamma \sim \Gamma'$ at w and $w \not\prec w'$ then $(\Gamma, A[w']) \sim \Gamma'$ at w, for any A and w'.*
- *If $\Gamma \sim \Gamma'$ at w then $(\Gamma, A[w']) \sim (\Gamma', A[w'])$ at w, for any A and w'.*

We can now properly state a general property about weakening and strengthening of hypotheses that are irrelevant from the perspective of the world we are considering.

Theorem 2 (Irrelevance of hypotheses at unreachable worlds)**.**

- *If $\Gamma \sim \Gamma'$ at w and $\Gamma \vdash A[w]$ then $\Gamma' \vdash A[w]$.*

Proof. The proof follows by structural induction on the proof of $\Gamma \vdash A[w]$ and by induction on the accessibility relation. □

2.4.2 Weakening and Exchange

As expected, the usual structural properties of weakening and exchange hold in our system. These properties follow from a more general theorem that is easier to establish within the Agda proof assistant. The informal idea of this theorem is we can extend, contract, and rearrange elements in a context freely so long as those changes pertain only to the particular world we are referring to. We denote by $\Gamma{\downharpoonleft}w$ the judgment that context Γ only holds assumptions of the form $A[w]$.

Theorem 3 (Generalized Tethered Weakening)**.**

- *For all* Γ, Γ' *and* Ψ*, if* $\Gamma \subseteq \Gamma'$*,* $\Gamma'{\downharpoonleft}w$ *and* $\Psi, \Gamma \vdash A[w]$ *then* $\Psi, \Gamma' \vdash A[w]$*.*

Proof. The proof follows by structural induction over the derivation of $\Gamma, \Psi \vdash A[w]$. □

The following are all immediate corollaries of Theorem 3:

- Tethered weakening: $\Gamma \vdash C[w]$ implies $\Gamma, A[w] \vdash C[w]$,
- Tethered exchange: $\Gamma, A[w], B[w] \vdash C[w]$ implies $\Gamma, B[w], A[w] \vdash C[w]$, and
- Tethered contraction: $\Gamma, A[w], A[w] \vdash C[w]$ implies $\Gamma, A[w] \vdash C[w]$.

2.4.3 Identity

A logic can only be called such if it is *globally complete*. Global completeness states that the left rules of the system are strong enough to decompose a complex formula down to its atomic propositions (which can then be concluded by the hypothesis rule) and is embodied by the admissibility of Identity.

Theorem 4 (Admissibility of identity)**.** *For all* A*,* $\Gamma, A[w] \vdash A[w]$*.*

Proof. By structural induction on A. The cases for implication, conjunction, disjunction and atomic propositions are standard. We present here some of the more interesting cases:

Case: ($A = \neg A$)

$\mathcal{F} :: (\forall w'.(w \prec w') \Rightarrow (\Gamma \vdash A[w']) \Rightarrow 0) \Rightarrow \Gamma, \neg A[w] \vdash A[w]$
by the following hypothetical reasoning:

Assume $\mathcal{D}_0 :: (\Gamma \vdash A[w']) \Rightarrow 0$ for an arbitrary w' such that $w \prec w'$

$\mathcal{D}_1 :: (\Gamma \vdash \neg A[w])$ by $\neg_R$

$\mathcal{D}_{1w} :: (\Gamma, \neg A[w] \vdash \neg A[w])$ by weakening on $\mathcal{D}_1$

$\mathcal{F}' :: (\Gamma, \neg A[w] \vdash \neg A[w])$ by $\neg_L$ on $\mathcal{F}$

Case: $(A = \Box A)$

$\mathcal{F} :: (\forall w'.(w \prec w') \Rightarrow (\Gamma \vdash A[w']) \Rightarrow \Gamma, \Box A[w] \vdash \Box A[w])$
by the following hypothetical reasoning:

Assume $\mathcal{D}_0 :: (\forall w'.(w \prec w') \Rightarrow \Gamma \vdash A[w'])$

$\mathcal{D}_1 :: (\Gamma \vdash \Box A[w])$ by $\Box_R$ on $\mathcal{D}_0$

$\mathcal{D}_{1w} :: (\Gamma, \Box A[w] \vdash \Box A[w])$ by weakening on $\mathcal{D}_1$

$\mathcal{F}_1 :: (\Gamma, \Box A[w] \vdash \Box A[w])$ by $\Box_L$ on $\mathcal{F}$

Case: $(A = \Diamond A)$

$\mathcal{F} :: (\forall w'.(w \prec w') \Rightarrow (\Gamma \vdash A[w']) \Rightarrow \Gamma, \Diamond A[w] \vdash \Diamond A[w])$
by the following hypothetical reasoning:

Assume $\mathcal{D}_0 :: (\Gamma \vdash A[w'])$ for an arbitrary w' such that $w \prec w'$

$\mathcal{D}_1 :: (\Gamma \vdash \Diamond A[w])$ by $\Diamond R$ on $\mathcal{D}_0$

$\mathcal{D}_{1w} :: (\Gamma, \Diamond A[w] \vdash \Diamond A[w])$ by weakening on $\mathcal{D}_1$

$\mathcal{F}_1 :: \Gamma, \Diamond A[w] \vdash \Diamond A[w]$ by $\Diamond_L$ on $\mathcal{F}$

□

2.4.4 Cut elimination

Finally, we insist that a reasonable logic be not only globally complete, but also *globally sound*. Global soundness in this setting follows from the proof of admissibility of cut. It is particularly interesting that Theorem 5 can be proved *independently* of cut admissibility at any other world.

Theorem 5 (Admissibility of tethered cut)**.**

- *If* $\Gamma \vdash A[w]$ *and* $\Gamma, A[w] \vdash C[w]$ *then* $\Gamma \vdash C[w]$

Proof. Lexicographic induction over A, the first derivation, and the second derivation. We present the most interesting (principal) cases:

Case: $(A = A \supset B)$

$$\mathcal{D} = \cfrac{\overset{\mathcal{D}_1}{\Gamma, A[w] \vdash B[w]}}{\Gamma \vdash A \supset B[w]} \supset_R$$

$$\mathcal{E} = \cfrac{\overset{\mathcal{E}_1}{\Gamma, A \supset B[w] \vdash A[w]} \quad \overset{\mathcal{E}_2}{\Gamma, A \supset B[w], B[w] \vdash C[w]}}{\Gamma, A \supset B[w] \vdash C[w]} \supset_L$$

$\mathcal{D}' :: \Gamma \vdash A[w]$	i.h. on $A \supset B$, $\mathcal{D}$ and $\mathcal{E}_1$
$\mathcal{D}_1' :: \Gamma \vdash B[w]$	i.h. on A, $\mathcal{D}'$ and $\mathcal{E}_1$
$\mathcal{D}_w :: \Gamma, B[w] \vdash A \supset B[w]$	by weakening on $\mathcal{D}$
$\mathcal{E}_2' :: \Gamma, B[w] \vdash C[w]$	i.h. on $A \supset B$, $\mathcal{D}_w$ and $\mathcal{E}_2$
$\mathcal{F} :: \Gamma \vdash C[w]$	i.h. on B, $\mathcal{D}_1'$ and $\mathcal{E}_2'$

Case: $(A = \Diamond A)$

$$\mathcal{D} = \cfrac{\overset{\mathcal{D}_1}{w \prec w'} \quad \overset{\mathcal{D}_2}{\Gamma \vdash A[w']}}{\Gamma \vdash \Diamond A[w]} \Diamond_R$$

$$\mathcal{E} = \cfrac{\overset{\mathcal{E}_1}{\forall w'.(w \prec w') \Rightarrow (\Gamma \vdash A[w']) \Rightarrow \Gamma, \Diamond A[w] \vdash C[w]}}{\Gamma, \Diamond A[w] \vdash C[w]} \Diamond_L$$

$\mathcal{F}_1 :: ((\Gamma \vdash A[w']) \Rightarrow \Gamma, \Diamond A[w] \vdash C[w])$	by $\mathcal{E}_1(\mathcal{D}_1)$
$\mathcal{F}_2 :: (\Gamma, \Diamond A[w] \vdash C[w])$	by $\mathcal{F}_1(\mathcal{D}_2)$
$\mathcal{F}_3 :: (\Gamma \vdash C[w])$	i.h. on $\Diamond A$, $\mathcal{D}$ and $\mathcal{F}_2$

Case: $(A = \neg A)$

$$\mathcal{D} = \cfrac{\overset{\mathcal{D}_1}{\forall w'.(w \prec w') \Rightarrow (\Gamma \vdash A[w'] \Rightarrow 0)}}{\Gamma \vdash \neg A[w]} \neg_R$$

$$\mathcal{E} = \cfrac{\overset{\mathcal{E}_1}{w \prec w'} \quad \overset{\mathcal{E}_2}{(\Gamma \vdash A[w'] \Rightarrow 0) \Rightarrow (\Gamma, \neg A[w] \vdash C[w])}}{\Gamma, \neg A[w] \vdash C[w]} \neg_L$$

$\mathcal{F}_1 :: (\Gamma \vdash A[w'] \Rightarrow 0)$ by $\mathcal{D}_1(\mathcal{E}_1)$
$\mathcal{F}_2 :: (\Gamma, \neg A[w] \vdash C[w])$ by $\mathcal{E}_2(\mathcal{F}_1)$
$\mathcal{F}_3 :: (\Gamma \vdash C[w])$ i.h. on $\neg A$, $\mathcal{D}$ and $\mathcal{F}_2$

Case: $(A = \Box A)$

$$\mathcal{D} = \frac{\overset{\mathcal{D}_1}{\forall w'.(w \prec w') \Rightarrow \Gamma \vdash A[w']}}{\Gamma \vdash \Box A[w]} \Box_R$$

$$\mathcal{E} = \frac{\overset{\mathcal{E}_1}{(\forall w'.(w \prec w') \Rightarrow (\Gamma \vdash A[w'])) \Rightarrow \Gamma, \Box A[w] \vdash C[w]}}{\Gamma, \Box A[w] \vdash C[w]} \Box_{L2}$$

$\mathcal{F}_1 :: (\Gamma, \Box A[w] \vdash C[w])$ by $\mathcal{E}_1(\mathcal{D}_1)$
$\mathcal{F}_2 :: (\Gamma \vdash C[w])$ i.h. on $\Box A$, $\mathcal{D}$ and $\mathcal{F}_1$

□

2.5 Formalization of CPL and its metatheory

All of the results discussed up to this point have been formalized in the Agda proof assistant. In this section, we will discuss several aspects of this formalization.

Positivity The definition of the logic itself is the only point at which our formalization is not fully verified by the proof assistant, and for precisely the reasons we discussed in our development of the left rule for $\Diamond$. The well-foundedness of our logic definition is entirely dependent on the well-foundedness of our accessibility relation, and Agda is unable to verify that our definition is not tantamount to a negative occurrence. This is unsurprising: seemingly minor changes to our logic (such as a copy rule that allowed us to assume $A[w]$ given an assumption of $A[w']$ for $w \prec w'$) would break not only the irrelevance theorem but could interfere with the consistency of Agda itself.

For this reason, most of our results were also established in a different style that did not push past the limits of Agda's positivity checker. In this alternate formalization, provability at each world was defined independently in an Agda module that took as an argument an unspecified entailment relation at each of an unspecified number of reachable worlds.

Then, given a finite accessibility relation, the logic could be instantiated "by hand" by instantiating the logic at all worlds with no other accessible worlds and then working backwards.

In this setting, the proof of every result established by induction on the accessibility relation (such as the irrelevance of unreachable hypotheses and the validity of axiom *GL* given a transitive accessibility relation) was rather unpleasant. The truth of the theorem at all accessible worlds (the induction hypothesis) needed to be passed as an argument to the module and then verified at the present world within the module. This was an important stage in the development of **CPL**, but once we had established that the logic made sense in a fully-verified setting, we found it much more pleasant to use the uniform version of the logic that does not pass Agda's positivity checker, allowing us to use infinite (but still upwards well-founded) accessibility relations.

Justifying cut elimination Agda verifies the termination arguments for recursive proofs entirely by structural induction — in its simplest form, this requires that one of the arguments to the recursive call is be a strict subterm of one of the arguments to the function. However, if we consider the principal cut for $\Diamond A$ in the proof of cut elimination, what validates the invocation of the induction hypothesis on $\neg A[w]$, $\mathcal{D}$, and $\mathcal{F}_2$? It is the fact that $\mathcal{F}_2 = (\mathcal{E}_1(\mathcal{D}_1))(\mathcal{D}_2)$ which we take to be smaller; the function $\mathcal{E}_1$ is a subterm of the larger proof $\mathcal{E}$, and the arguments to the function are understood as merely selecting the particular smaller proof that we need; any one of these proofs is a subterm of $\mathcal{E}$.

Primarily because of the complexity of the termination arguments for the proof of cut admissibility, the Agda formalization of our metatheoretic results was critical to our confidence in the language's consistency. However, the validation of our termination arguments came at high cost. Many cases of cut elimination (such as the principal cut for implication) rely on the structural properties of weakening and exchange, and Agda does not consider a subderivation of $\mathcal{D}$ with weakening applied to it to be a subderivation of $\mathcal{D}$ for the purposes of cut elimination.

Our solution to this, which was straightforward if relatively high-effort, was to create a copy of the logic that is indexed by a structural metric that captures the shape of the derivation. A stronger version of the weakening and exchange lemmas was then proved; this stronger lemma asserts that the application of weakening and exchange does not change the metric representing the shape of the derivation. By proving cut elimination for the metric-endowed logic, Agda was able to verify that in every case the shape of the derivation (if not the derivation itself) became smaller. Finally, to establish cut for the logic as originally defined, we wrote two functions that added and stripped the metric as needed. The higher-order aspects of our derivation made the structural metric rather involved; we refer readers to our Agda formalization for details.[1]

3 Logic programming in CPL

In this section, we consider the use of **CPL** as a way of understanding negation in logic programming. In Section 3.1 we describe how Horn programs with negation can be translated into stratified **CPL** programs, and in Section 3.2 we discuss how both backward-chaining and forward-chaining proof search (as well as a mixture of the two) could be used as a basis for implementing logic programming in **CPL**.

The examples in this section consider a first-order extension of **CPL** with universal quantification. We did not formalize first-order **CPL** in Agda as this would have introduced technical complications, but the inclusion is not problematic from a logical perspective:

$$\frac{\Gamma \vdash A(\alpha)[w]}{\Gamma \vdash \forall x.A(x)[w]}\ \forall_R^\alpha \qquad \frac{\Gamma, \forall x.A(x)[w], A(t)[w] \vdash C[w]}{\Gamma, \forall x.A(x)[w] \vdash C[w]}\ \forall_L$$

In terms of the operational semantics of a logic programming language, there can be significant issues with the addition of quantification. Quantification in logic programming is generally implemented by the introduction of metavariables that are subject to instantiation, and it is often not possible to give a good logical account for the use of metavariables (especially when mixing forward-chaining and backward-chaining). In order to avoid consideration of the interaction of metavariables and the modalities, we impose an operational requirement on proof search: queries about provability or non-provability at accessible worlds must be *ground*. This means that when the logic programming interpreter encounters a subgoal about provability at an accessible world, that subgoal can no longer contain any metavariables. Existing techniques for mode checking in logic programming can be utilized to verify this requirement [RP96], and we do not consider such analyses here.

[1] The code is available from a public Subversion repository. To access the code, type the command `svn co https://svn.concert.cs.cmu.edu/lollibot/papers/modlog/cpl/` at the command line of any machine with Subversion.

3.1 Logic programming with explicit worlds

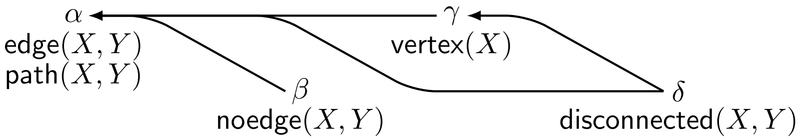
In this section, we will consider a modified version of the logic program with stratified negation from the introduction (conjunction $\wedge$ binds tighter than implication $\supset$):

$$\begin{aligned}
\mathsf{edge}(X,Y) &\supset \mathsf{edge}(Y,X)\\
\mathsf{edge}(X,Y) &\supset \mathsf{path}(X,Y)\\
\mathsf{edge}(X,Y) \wedge \mathsf{path}(Y,Z) &\supset \mathsf{path}(X,Z)\\
\mathsf{path}(X,Y) \wedge \neg\mathsf{edge}(X,Y) &\supset \mathsf{noedge}(X,Y)\\
\mathsf{path}(X,Y) &\supset \mathsf{vertex}(X)\\
\neg\mathsf{path}(X,Y) &\supset \mathsf{disconnected}(X,Y)
\end{aligned}$$

We can explain the meaning of this logic program by interpreting it into a Horn-like fragment of **CPL**. In order to define the Horn-like fragment of **CPL** we first must present a function W from atomic propositions Q to worlds. A well-formed Horn clause at a world w is defined inductively in terms of W:

$$\begin{array}{rl}
Q\ \mathsf{wf\text{-}horn}\ w & \text{iff } W(Q) = w\\
A \supset Q\ \mathsf{wf\text{-}horn}\ w & \text{iff } W(Q) = w \text{ and } A\ \mathsf{wf\text{-}premise}\ w\\
Q\ \mathsf{wf\text{-}premise}\ w & \text{iff } W(Q) = w\\
Q @ w'\ \mathsf{wf\text{-}premise}\ w & \text{iff } W(Q) = w' \text{ and } w \prec w'\\
Q \not@ w'\ \mathsf{wf\text{-}premise}\ w & \text{iff } W(Q) = w' \text{ and } w \prec w'\\
A \wedge B\ \mathsf{wf\text{-}premise}\ w & \text{iff } A\ \mathsf{wf\text{-}premise}\ w \text{ and } B\ \mathsf{wf\text{-}premise}\ w
\end{array}$$

Having described well-formed Horn clauses, we need to pick some set of worlds, an accessibility relation, and a mapping from worlds to atomic propositions. The only constraints on this selection are that the result has to be consistent with the uses of negation in the original logic program. The following accessibility relation and assignment of atomic propositions to worlds works for our example, though it is not the only possibility:



Now we can appropriately annotate our program with references to the correct worlds and locate each rule at the world where its conclusion belongs. Each rule below is a valid Horn clause according to our definition.

$$
\begin{aligned}
(\mathsf{edge}(X,Y) &\supset \mathsf{edge}(Y,X)) \mathbin{@} \alpha \\
(\mathsf{edge}(X,Y) &\supset \mathsf{path}(X,Y)) \mathbin{@} \alpha \\
(\mathsf{edge}(X,Y) \wedge \mathsf{path}(Y,Z) &\supset \mathsf{path}(X,Z)) \mathbin{@} \alpha \\
(\mathsf{path}(X,Y) \mathbin{@} \alpha \wedge \mathsf{edge}(X,Y) \mathbin{\not@} \alpha &\supset \mathsf{noedge}(X,Y)) \mathbin{@} \beta \\
(\mathsf{path}(X,Y) \mathbin{@} \alpha &\supset \mathsf{vertex}(X)) \mathbin{@} \gamma \\
(\mathsf{path}(X,Y) \mathbin{\not@} \alpha &\supset \mathsf{disconnected}(X,Y)) \mathbin{@} \delta
\end{aligned}
$$

It is worthwhile to note that, because the strategy we have outlined always refers to other worlds by explicit reference, requiring the accessibility relation to be transitive comes at no cost to expressiveness. Therefore, we can equivalently consider ourselves to be doing logic programming in **CPL** or in a constructive variant of **KW**.

3.2 Proof search strategies

Logic programming is the use of a fixed proof search strategy that allows a programmer to reason (formally or informally) about the behavior of proof search. The most common proof search strategies for logic programs are SLD resolution (which is the basis of backward-chaining logic programming languages like Prolog) and hyperresolution (which is the basis of forward-chaining logic programming languages like Datalog). When dealing with Horn clauses, both of these proof search strategies are partially correct: they are *sound* (the finite success of proof search means a proof exists) and *complete* (the finite failure of proof search means that no proof exists).

As long as we restrict logic programming to the Horn-like fragment where queries $Q \mathbin{@} w$ and $Q \mathbin{\not@} w$ are only asked of ground atoms Q, both of these search strategies can be straightforwardly extended to **CPL**, and their correctness follows from the correctness of the proof search strategy and by induction over the accessibility relation.

Forward-chaining logic programming The standard way of implementing forward-chaining logic programming was sketched in the introduction: to determine the set of propositions that are provable at world w, first use forward-chaining logic programming to determine the set of propositions that are provable at all worlds accessible from w'. If and when this succeeds, then we can repeatedly derive the immediate consequences of the set of know facts and the Horn clauses at w until no new facts can be derived.

The immediate consequence operation takes a premise $A \supset Q[w]$, determines if all facts in A are currently true at w, and if so adds $Q[w]$ to the set of known facts. If A contains a premise $Q \,@\, w'$ for $w \prec w'$, then by the induction hypothesis we know that $Q[w']$ is provable if and only if $Q[w']$ is already in the set of propositions facts provable at w'. Similarly, if A contains a premise $Q \not@\, w'$, that premise is satisfiable if and only if $Q[w']$ is *not* one of the facts already determined to be true at w'.

Backward-chaining logic programming Negation as failure was a concept originally associated with backward-chaining (i.e. "Prolog-style") logic programming. Most implementations of backward-chaining logic programming with negation do not analyze for the kind of stratification condition that forward-chaining logic programs demand and that **CPL** enforces, but in the presence of this condition backward-chaining logic programming is correct with respect to the logic [Cav91].

A **CPL** logic program can therefore be implemented using backward chaining by a standard SLDNF interpreter: when proof search encounters a subgoal $Q \,@\, w'$, proof search for Q can continue at the world w'. If proof search encounters a subgoal $Q \not@\, w'$, proof search can be performed for Q at w', leading the subgoal to succeed precisely if proof search at w' fails. The correctness of the overall proof search procedure can then be established by induction over the accessibility relation: the correctness of proof search at world w is dependent on the correctness of proof search at all worlds accessible from w.

Combining forward and backward chaining Because of the strong separation between provability at different worlds in **CPL**, it is possible to use different proof search strategies at different worlds. In particular, as long as "backward-chaining worlds" can access "forward-chaining worlds" but not vice versa, proof search is entirely unproblematic; saturation can first take place at all of the forward-chaining worlds, and then queries can be run at the backward-chaining worlds. Whenever backward-chaining proof search reaches a subgoal at a forward-chaining world, the set of derivable facts at that world can be queried for the presence (or absence) of the subgoal. This setup is similar to strategies for credential chain discovery in the trust-management language RT as described by [LWM03] and implemented in Lollimon [PS06]. In our running example, for instance, it would be reasonable to set up α, β, and γ as forward-chaining worlds but to have the world δ where disconnected is defined be a backward-chaining world: rather than deriving all possible disconnected facts we would simply query the set of path facts whenever we needed to know if two points were disconnected.

It should also be possible to have "backward-chaining worlds" accessible from "forward-chaining worlds," though from a practical perspective this may be difficult or undesirable. In such an extension, the immediate consequence operation that forms the basis of forward-chaining logic programming goes from being an operation that always terminates to an operation that might fail to return a result due to divergent backward-chaining proof search at an accessible world. Such a change could, in practice, make it more difficult to reason about and predict the behavior of logic programs.

4 Conclusion

We have presented the constructive provability logic **CPL**, a modal logic that allows a strong form of reasoning about provability and non-provability at accessible worlds. We developed its metatheory, which we also implemented in Agda, in order to provide a solid basis for the logic, and gave an account of using **CPL** as a way to understand and justify constructive negation in a logic programming setting.

4.1 Future work

Accessible cut We have stated and formally proved the standard cut admissibility statement for **CPL**. However, another cut admissibility statement can be made regarding cut of hypotheses at accessible worlds.

Conjecture 1 (Admissibility of accessible cut)**.**

- *If* $w \prec w'$, $\Gamma \vdash A[w']$ *and* $\Gamma, A[w'] \vdash C[w]$ *then* $\Gamma \vdash C[w]$

We have not formalized this theorem, though we believe the difficulties are mainly due to technical issues of Agda related to the manipulation of contexts as lists and apparent need for accessibility to be decidable (that is, for any two worlds w and w', we must be able to determine whether $w \prec w'$ or $w \not\prec w'$). However, we believe that the conjecture is provable by induction on the second derivation by appropriately using tethered cut and irrelevance when necessary.

Formalization of logic programming results The discussion in Section 3 was largely informal, though we did attempt an Agda formalization of some of the results in that section. This formalization was ultimately incomplete for two reasons. The first was that we did not obtain a result of the *completeness* of proof search; such a result would follow quite simply from the completeness of a a focused [And92] (or weakly focused [PS09]) sequent calculus for **CPL**.

The primary reason that our Agda formalization of proof search ran into trouble, however, was that we were not entirely clear as to the best way to approach the problem. The following are all possibilities:

- Write a (possibly non-terminating) proof search procedure **Search** and prove it (partially) correct,
- Write a (possibly non-terminating) certificate-producing proof search procedure **SearchCert**,
- Write a terminating certificate-checking procedure **CheckCert** and prove it correct, or
- Prove that **SearchCert** always produces a witness accepted by **CheckCert**.

The power of Agda's dependent types suggests the first option: writing a function **Search** which, if it terminates, will return either a derivation of $\Gamma \vdash A[w]$ of the desired sequent or a function $(\Gamma \vdash A[w] \Rightarrow 0)$ that generates a contradiction if given such a derivation. The derivation that Agda returns may include embedded computations (and certainly this is the case if the sequent is unprovable!), so this procedure, while witness-producing, does not produce a certificate that can be easily analyzed, stored, or transmitted as data.

Therefore, we approached the problem from the perspective of writing **SearchCert** and **CheckCert**. Writing a not-necessarily-correct program like **SearchCert** is not really the intended use of Agda, however, so this process was occasionally awkward and, in any case, fraught with all the natural perils of programming without formal guarantees. A better approach would probably be to write **Search**, then **CheckCert**, and then finally to write **SearchCert** by modifying **Search** to return proof terms.

Other logic programming paradigms Because we have presented a general logic with a modality for constructive negation, it seems possible to use this logic to justify the support of negation in other logic programming formalisms. For instance, as long as negated propositions only end up as goals and never as assumptions, it would seem unproblematic to use **CPL** to justify negation in a higher-order logic programming language such as λProlog.

Additionally, we believe that the logic programming language Bedwyr's use of inductive definitions can be interpreted through **CPL**. While we have only considered the use of $Q \mathbin{@} w$ or $Q \not\mathbin{@} w$ as a *goal*, it seems like a higher-order goal formula $A \Rightarrow B$ in Bedwyr could be explained in **CPL** as $(A \mathbin{@} w_0 \supset B) \mathbin{@} w_1$, where $w_1 \prec w_0$ — in this case, $A \mathbin{@} w_0[w_1]$ ends up as a premise, a possibility we did not consider in our discussion of proof search in **CPL**. We did not have time to investigate this in detail, however.

A framework for combining and reflecting on logics We presented **CPL** as an extremely uniform system - the connectives act the same at every world. But this is not the only way in which we could have defined a logic that reflects over other logics.

Because the soundness of **CPL** is established independently at every level, we could use the modal logic we defined to reason about other, completely different logics — potentially even allowing user-defined logics to be analyzed. We could define a sound version of **CPL** logic with three "accessible" logics: intuitionistic logic, classical logic, and an unsound logic where any proposition is provable. Then, $\Diamond(A \vee (A \supset \bot))$ would be true (since the formula is provable in classical logic and the unsound logic), as would $\Diamond\bot$ (since falsehood is provable in the unsound logic). However, the logic would be *unable* to prove $\Box(A \vee (A \supset \bot))$ in general due to the fact that this formula is not, in general, true in intuitionistic logic, one of the three accessible logics.

4.2 Acknowledgments

The help of Dan Licata and Jamie Morgenstern was invaluable in our work with Agda. Christopher Richards, Vivek Nigam, Henry de Young, and David Baelde gave us a number of important insights into this logic and related work surrounding it, and Christopher Richards gave very helpful comments on an earlier draft of this work. Frank Pfenning and André Platzer must be acknowledged for teaching us everything we know about

modal logic, and André's inclusion of axiom **GL** on the course midterm, in particular, could not have been more auspicious.

References

[AMRV07] Andrew W. Appel, Paul-André Melliès, Christopher D. Richards, and Jérôme Vouillon. A very modal model of a modern, major, general type system. In *Proceedings of the 34th annual symposium on Principles of Programming Languages (POPL'07)*, pages 109–122. ACM, 2007.

[And92] Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *J. Logic Computation*, 2(3):297–347, June 1992.

[Cav91] Lawrence Cavedon. Acyclic logic programs and the completeness of sldnf-resolution. *Theoretical Computer Science*, 86(1):81–92, 1991.

[Gab91] D. M. Gabbay. Modal provability foundations for negation by failure. In *Extensions of Logic Programming*, pages 179–222. Springer LNCS 475, 1991.

[LWM03] Ninghui Li, William H. Winsborough, and John C. Mitchell. Distributed chain discovery in trust management. *Journal of Computer Security*, 11(1):35–86, February 2003.

[Nor07] Ulf Norell. *Towards a practical programming language based on dependent type theory*. PhD thesis, Chalmers University of Technology, 2007.

[PD01] Frank Pfenning and Rowan Davies. A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11:511–540, 2001. Notes to an invited talk at the *Workshop on Intuitionistic Modal Logics and Applications (IMLA'99)*, Trento, Italy, July 1999.

[Pfe10] Frank Pfenning. Lecture notes on tethered semantics. Lecture notes from 15-816: Modal Logic, March 2010.

[PS06] Jeff Polakow and Christian Skalka. Specifying distributed trust management in lollimon. In *Proceedings of the 2006 workshop on Programming Languages and Analysis for Security (PLAS'06)*, pages 37–46. ACM, 2006.

[PS09] Frank Pfenning and Robert J. Simmons. Substructural operational semantics as ordered logic programming. In *Proceedings of the 24th Annual Symposium on Logic in Computer Science (LICS'09)*, pages 101–110, Los Angeles, California, August 2009. IEEE Computer Society.

[Ree09] Jason Reed. A judgmental deconstruction of modal logic. Submitted for publication, May 2009.

[Ric10] Christopher D. Richards. *The Approximation Modality in Models of Higher-Order Types*. PhD thesis, Princeton University, 2010.

[RP96] Ekkehard Rohwedder and Frank Pfenning. Mode and termination checking for higher-order logic programs. In *Programming Languages and Systems (ESOP'96)*, pages 296–310. Springer LNCS 1058, 1996.

[Sim94] Alex K. Simpson. *The Proof Theory and Semantics of Intuitionistic Modal Logic*. PhD thesis, University of Edinburgh, 1994.