# Truthful Monadic Abstractions*

Taus Brock-Nannestad and Carsten Schürmann

IT University of Copenhagen
`tbro|carsten@itu.dk`

**Abstract.** In intuitionistic sequent calculi, detecting that a sequent is unprovable is often used to direct proof search. This is for instance seen in backward chaining, where an unprovable subgoal means that the proof search must backtrack. In undecidable logics, however, proof search may continue indefinitely, finding neither a proof nor a disproof of a given subgoal.
In this paper we characterize a family of truth-preserving abstractions from intuitionistic first-order logic to the monadic fragment of classical first-order logic. Because they are truthful, these abstractions can be used to disprove sequents in intuitionistic first-order logic.

## 1 Introduction

Two common methods of proof search in intuitionistic sequent calculi are forward and backward chaining. In forward chaining, the hypotheses are used to derive new facts. The proof succeeds if the goal is derived, and fails if this process saturates (i.e. no new facts can be derived) without deriving the goal.

Backward chaining proceeds bottom-up from the goal, continually splitting subgoals into other subgoals or closing subgoals if these are an immediate consequence of what is in the context. Backtracking is employed if a subgoal can be neither split nor immediately satisfied. In the case of backward chaining, the proof search succeeds if all subgoals are satisfied, and it fails if backtracking returns to the original goal without finding a proof.

In undecidable logics such as intuitionistic first-order logic, both of these approaches may fail to terminate. Forward chaining may fail to saturate, continually adding new facts to the context, and backward chaining may have an unbounded number of points at which backtracking may occur. Consider, for instance, an attempt to prove that 5 is an even number. Let the predicate $D(x, y)$ — signifying that $y$ is twice the value of $x$ — be given by the following context

$$\Gamma = D(z, z), \forall x.\forall y.D(x, y) \supset D(s(x), s(s(y)))$$

The statement that 5 is even is equivalent to the following sequent

$$\Gamma \Longrightarrow \exists x.D(x, s(s(s(s(s(z))))))$$

* This work supported in part by DemTech grant 10-092309 from the Danish Council for Strategic Research, Programme Commission on Strategic Growth Technologies.

Clearly, attempting to prove the above sequent cannot succeed, but a naive forward chaining proof attempt may fail to terminate, as any number of distinct hypotheses $D(s(z), s(s(z))), D(s(s(z)), s(s(s(s(z))))), \ldots$ may be generated by the left implication rule. Thus, the proof search never saturates, and we cannot conclude that the above sequent is not derivable.

Of course, the above sequent fails to be derivable even in classical logic, hence one might try to feed it to a classical first-order theorem prover, such as Spass [13] or Vampire [11], or a countermodel generator, such as for example Paradox [2] hoping to generate a counterexample. This technique will work for the above simple example, but in general there is no guarantee that the tools will terminate either, as we have reduced one undecidable problem to another.

If we insist on having decidability, we must let go of either *soundness*, in which case whatever proofs we find are not necessarily correct, or *completeness*, in which case we may not be able to find a proof even if a proof exists. Thus, if we primarily care about disproving sequents, having an unsound but complete and decidable procedure is sufficient.

In this paper, we define a family of *truthful* (or TI [3]) abstractions i.e. functions that preserve provability. Thus, each abstraction is a function $\alpha$ that satisfies the truthfulness condition:

$$\Gamma \Longrightarrow A \quad \text{entails} \quad \alpha(\Gamma) \longrightarrow \alpha(A)$$

where the sequent on the left is in the *base* logic, and the sequent on the right is in the *abstraction* logic. By choosing a logic in which derivability is decidable, we get the aforementioned decision procedure.

In our case, our abstractions map into the monadic fragment of classical first-order logic which has the finite model property, and is thus decidable [4]. In the monadic fragment, only unary predicates and functions may appear, but all other connectives are allowed. Thus, the defining feature of the abstraction is how it acts on atoms. This mapping can be intuitively understood as mapping arbitrary terms — which may be seen as being tree-shaped — onto paths through these trees. Thus, the following formula

$$P(f(g(a), b), c) \supset P(f(c, b), g(a))$$

will have the following abstractions:

$$P_1(f_1(g_1(a))) \supset P_1(f_1(c))$$

$$P_1(f_2(b)) \supset P_1(f_2(b))$$

$$P_2(c) \supset P_2(g_1(a))$$

Here, the subscripts indicate which argument became part of the path.

This paper is organized as follows. We reiterate the definition of intuitionistic first-order logic, our choice of base logic, in Section 2. In Section 3 we define the

$$\dfrac{P \in \Gamma}{\Gamma \Longrightarrow P}\,\text{init} \qquad \dfrac{}{\Gamma \Longrightarrow \top}\,\top\text{R} \qquad \dfrac{}{\Gamma, \bot \Longrightarrow C}\,\bot\text{L}$$

$$\dfrac{\Gamma, A \wedge B, A, B \Longrightarrow C}{\Gamma, A \wedge B \Longrightarrow C}\,\wedge\text{L} \qquad \dfrac{\Gamma \Longrightarrow A \quad \Gamma \Longrightarrow B}{\Gamma \Longrightarrow A \wedge B}\,\wedge\text{R}$$

$$\dfrac{\Gamma \Longrightarrow A_i}{\Gamma \Longrightarrow A_1 \vee A_2}\,\vee\text{R}_i \qquad \dfrac{\Gamma, A \vee B, A \Longrightarrow C \quad \Gamma, A \vee B, B \Longrightarrow C}{\Gamma, A \vee B \Longrightarrow C}\,\vee\text{L}$$

$$\dfrac{\Gamma, A \supset B \Longrightarrow A \quad \Gamma, A \supset B, B \Longrightarrow C}{\Gamma, A \supset B \Longrightarrow C}\supset\text{L} \qquad \dfrac{\Gamma, A \Longrightarrow B}{\Gamma \Longrightarrow A \supset B}\supset\text{R}$$

$$\dfrac{\Gamma, \exists x.A, [a/x]A \Longrightarrow C}{\Gamma, \exists x.A \Longrightarrow C}\,\exists\text{L}^a \qquad \dfrac{\Gamma \Longrightarrow [t/x]A}{\Gamma \Longrightarrow \exists x.A}\,\exists\text{R}$$

$$\dfrac{\Gamma, \forall x.A, [t/x]A \Longrightarrow C}{\Gamma, \forall x.A \Longrightarrow C}\,\forall\text{L} \qquad \dfrac{\Gamma \Longrightarrow [a/x]A}{\Gamma \Longrightarrow \forall x.A}\,\forall\text{R}^a$$

**Fig. 1.** Base Logic: First-order intuitionistic logic

abstraction formally. Section 4 describes a few examples, illustrating the applicability of the decision procedure. In Section 5 we sketch briefly an implementation of truthful abstractions and report on our experimental results on the intuitionistic fragment ILTP[10] of TPTP[12], using Spass as a decision procedure for the monadic fragment of classical logic. We assess results, describe related work and conclude in Section 6.

## 2 Base Logic

In this paper we focus on intuitionistic first-order logic as a base logic, which is of particular interest to us because the result seems to scale to proof search problems in type theories, which we plan to study in future work. In general, the idea of truthful monadic abstractions is applicable in other settings as well.

In particular, we work with a sequent calculus formulation of first-order intuitionistic logic (FOL). The syntactic categories of terms and formulas are defined as follows:

$$\begin{aligned} t &::= x \mid c \mid f(t_1, \ldots, t_n) \\ A, B &::= P(t_1, \ldots, t_n) \mid A \wedge B \mid A \vee B \mid A \supset B \mid \top \mid \bot \mid \forall x.A \mid \exists x.A \end{aligned}$$

Here, $x$ is a variable and $c$ a constant. We let $F$ denote the set of function symbols $f$ and assume there is a function $\Sigma : F \to \mathbb{N}$ that records the arity of each function symbol. The connectives and the inference rules depicted in Figure 1 are standard. The superscripted variables on the $\forall$R and $\exists$L rules indicate that these rules are subject to the eigenvariable condition.

## 3 Truthfulness

There are many different decidable fragments of classical first-order logic. Most of these fragments impose restrictions on the shape of the formulas that may occur e.g. by restricting the quantifier prefixes, number of variables or guardedness of the formulas. For our purposes, we require that there is no restriction on the shape of the formulas, hence these fragments are *a priori* not suitable as targets for our abstraction.

The truthful abstraction that we present in this section maps sequents from the base logic into the *abstraction logic*, the monadic fragment of classical first-order logic, which we will not define in any more detail than to say that all predicate and function symbol are required to be unary. Base and abstraction logic formulas both share the same connectives, and as we will see in this section, we will define the truthful abstraction as a composition of three conceptually simpler abstractions through two intermediate logics.

1. Base logic: Intuitionistic first-order logic
2. Intermediate logic 1: Intuitionistic first-order logic with monadic predicates
3. Intermediate logic 2: Monadic intuitionistic first-order logic
4. Abstraction logic: Monadic classical first-order logic

Since the composition of truthful abstractions is again a truthful abstraction, we are able to compose the three and obtain the desired truthful abstraction from base to abstraction logic. The compositionality of truthful abstractions bears another interesting opportunity: Any logic that can be mapped truthfully into the base logic gives rise to a truthful abstraction into monadic first-order logic.

In the remainder of this section, we discuss each of the three abstractions in turn, establish the desired completeness result.

### 3.1 From Predicates to Monadic Predicates

We describe the mapping from the base logic into the intermediate logic 1, i.e. from intuitionistic first-order formulas to intuitionistic first-order logic with monadic predicates. An obvious candidate for such a mapping is a homomorphism that replaces predicates with monadic predicates, dropping all but one argument. Notice, that the rules init, $\exists$R, and $\forall$L from Figure 1 pose a particular challenge in the proof of truthfulness, because abstraction and substitution must commute in a suitable fashion for the proof to go through.

Let $\pi$ be a function from the set of predicates to $\mathbb{N}$ such that $\pi(P)$ points to a valid index, i.e. it is at most equal to the arity of $P$. We then define a map, $\alpha$, as follows:

$$\begin{aligned}
\alpha(\top) &= \top\\
\alpha(\bot) &= \bot\\
\alpha(A \odot B) &= \alpha(A) \odot \alpha(B)\\
\alpha(Qx.A) &= Qx.\alpha(A)\\
\alpha(P(t_1,\ldots,t_n)) &= P(t_i) \text{ where } i = \pi(P)
\end{aligned}$$

where $\odot$ is $\wedge$,$\vee$ or $\supset$ and $Q$ is either $\forall$ or $\exists$.

Since $\alpha$ is only defined on formulas and not terms, this definition behaves nicely with regard to substitution, as the following lemma shows.

**Lemma 1 (Substitutivity).** *For any term $t$, formula $A$, and $\alpha$ defined as above, the equality $\alpha([t/x]A) = [t/x]\alpha(A)$ holds.*

*Proof.* By straightforward structural induction. □

Proving that $\alpha$ is a truthful abstraction is now easy.

**Theorem 1 (Truthfulness).** *If $\Gamma \Longrightarrow A$ then $\alpha(\Gamma) \Longrightarrow \alpha(A)$.*

*Proof.* By structural induction on the derivation of $\Gamma \Longrightarrow A$. Most cases are immediate. We show here a few of the more interesting cases.

**Case:** $\mathcal{D} :: \dfrac{}{\Gamma, P(t_1,\ldots,t_n) \Longrightarrow P(t_1,\ldots,t_n)}\,\text{init}$

$\alpha(P(t_1,\ldots,t_n)) = P(t_i)$ for some $i$ — by definition of $\alpha$.

$\alpha(\Gamma), P(t_i) \Longrightarrow P(t_i)$ — by init.

**Case:**

$$\dfrac{\begin{matrix}\mathcal{D}\\ \Gamma \Longrightarrow [t/x]A\end{matrix}}{\Gamma \Longrightarrow \exists x.A}\,\exists\text{R}$$

$\alpha(\Gamma) \Longrightarrow \alpha([t/x]A)$ — by i.h. on $\mathcal{D}$.

$\alpha(\Gamma) \Longrightarrow [t/x]\alpha(A)$ — by Lemma 1

$\alpha(\Gamma) \Longrightarrow \exists x.\alpha(A)$ — by $\exists$R.

$\alpha(\Gamma) \Longrightarrow \alpha(\exists x.A)$ — by definition of $\alpha$.

**Case:**

$$\dfrac{\begin{matrix}\mathcal{D}\\ \Gamma, \forall x.A, [t/x]A \Longrightarrow C\end{matrix}}{\Gamma, \forall x.A \Longrightarrow C}\,\forall\text{L}$$

$\alpha(\Gamma), \alpha(\forall x.A), \alpha([t/x]A) \Longrightarrow \alpha(C)$ — by i.h. on $\mathcal{D}$.

$\alpha(\Gamma), \forall x.\alpha(A), [t/x]\alpha(A) \Longrightarrow \alpha(C)$ — by definition of $\alpha$ and Lemma 1.

$\alpha(\Gamma), \forall x.\alpha(A) \Longrightarrow \alpha(C)$ — by $\forall$L.

$\alpha(\Gamma), \alpha(\forall x.A) \Longrightarrow \alpha(C)$ — by definition of $\alpha$.

□

With this abstraction in place, we may now assume (without loss of generality) that all predicates are unary until the end of this section.

### 3.2 From Terms to Monadic Terms

We now tend to the most challenging part of the abstraction, a mapping form intermediate logic 1 to intermediate logic 2 that maps intuitionistic first-order logic with monadic predicates into monadic intuitionistic first-order logic, replacing $n$-ary function symbols of the term language into unary function symbols. Here again, the abstraction and substitution applications from rules init, ∃R, and ∀L must commute. This is more complex than before, because the abstraction defined here will abstract terms as well.

Let $\mu$ be any endofunction on the set of terms satisfying $\mu(x) = x$ for all variables $x$. Its homomorphic extension to formulas and contexts is given by the following definition

$$\begin{array}{ccc}
\mu(P(t)) = P(\mu(t)) & \mu(\top) = \top & \mu(\bot) = \bot \\
\mu(A \wedge B) = \mu(A) \wedge \mu(B) & \mu(A \vee B) = \mu(A) \vee \mu(B) & \mu(A \supset B) = \mu(A) \supset \mu(B) \\
\mu(\forall x.A) = \forall x.\mu(A) & \mu(\exists x.A) = \exists x.\mu(A) & \\
\mu(\cdot) = \cdot & \mu(\Gamma, A) = \mu(\Gamma), \mu(A) &
\end{array}$$

where $\mu$ must map arbitrary terms $t$ to monadic terms $\mu(t)$, which refer only to function symbols of arity 1. The intermediate logic 2 under the image of $\mu$ must admit the following three rules of inference that we obtain from Figure 1. The other inference rules of the intermediate logic 1 remain unchanged.

$$\frac{}{\Gamma, P(\mu(t)) \longrightarrow P(\mu(t))}\ \mathrm{init}^{\mu}$$

$$\frac{\Gamma \longrightarrow [\mu(t)/x]A}{\Gamma \longrightarrow \exists x.A}\ \exists\mathrm{R}^{\mu} \qquad \frac{\Gamma, \forall x.A, [\mu(t)/x]A \longrightarrow C}{\Gamma, \forall x.A \longrightarrow C}\ \forall\mathrm{L}^{\mu}$$

**Theorem 2.** *Let $\mu$ be a function defined as above. The following properties are then equivalent:*

1. *(Truthfulness) If $\Gamma \Longrightarrow A$ is provable, then so is $\mu(\Gamma) \longrightarrow \mu(A)$.*
2. *(Substitutivity on terms) For all terms $t, t'$:*

$$\mu([t/x]t') = [\mu(t)/x]\mu(t').$$

3. *(Substitutivity on formulas) For all terms $t$ and formulas $A$:*

$$\mu([t/x]A) = [\mu(t)/x]\mu(A)$$

*Proof.* (2)⇒(3) follows from a straightforward structural induction on $A$.

(1)⇒(2) Let $t$ and $t'$ be given. The sequent

$$P(t), \forall x.P(x) \supset Q(t') \Longrightarrow Q([t/x]t')$$

is easily seen to be derivable, as the following derivation shows:

$$\dfrac{\dfrac{}{P(t) \Longrightarrow P(t)}\,\text{init} \quad \dfrac{}{Q([t/x]t') \Longrightarrow Q([t/x]t')}\,\text{init}}{\dfrac{P(t), P(t) \supset Q([t/x]t') \Longrightarrow Q([t/x]t')}{P(t), \forall x.P(x) \supset Q(t') \Longrightarrow Q([t/x]t')}\,\forall\text{L}}\,\supset\text{L}$$

hence by assumption the sequent

$$\mu(P(t), \forall x.P(x) \supset Q(t')) \longrightarrow \mu(Q([t/x]t'))$$

which is equal to

$$P(\mu(t)), \forall x.P(x) \supset Q(\mu(t')) \longrightarrow Q(\mu([t/x]t'))$$

is likewise derivable. It may be shown (e.g. by means of *focusing*[1,6]) that the following must then be a proof (with occurrences of contraction elided)

$$\dfrac{\dfrac{}{P(\mu(t)) \longrightarrow P(\mu(s))}\,\text{init}^\mu \quad \dfrac{}{Q([\mu(s)/x]\mu(t')) \longrightarrow Q(\mu([t/x]t'))}\,\text{init}^\mu}{\dfrac{P(\mu(t)), P(\mu(s)) \supset Q([\mu(s)/x]\mu(t')) \longrightarrow Q(\mu([t/x]t'))}{P(\mu(t)), \forall x.P(x) \supset Q(\mu(t')) \longrightarrow Q(\mu([t/x]t'))}\,\forall\text{L}^\mu}\,\supset\text{L}^\mu$$

for some term $\mu(s)$. Substituting the equality $\mu(t) = \mu(s)$ from the left init rule into the equality $[\mu(s)/x]\mu(t') = \mu([t/x]t')$ from the right init rule yields the desired equality $\mu([t/x]t') = [\mu(t)/x]\mu(t')$.

(3)⇒(1) By structural induction on the derivation of $\Gamma \Longrightarrow A$. All cases are straightforward. The substitutivity property is used in the ∃R and ∀L cases.

$$\dfrac{\begin{array}{c}\mathcal{D}\\ \Gamma \Longrightarrow [t/x]A\end{array}}{\Gamma \Longrightarrow \exists x.A}\,\exists\text{R}$$

Applying the induction hypothesis to $\mathcal{D}$, we get a derivation of $\mu(\Gamma) \longrightarrow \mu([t/x]A)$. By assumption, this is equal to a derivation $\mu(\Gamma) \longrightarrow [\mu(t)/x]\mu(A)$ to which we may apply the $\exists\text{R}^\mu$ rule to get a derivation of $\mu(\Gamma) \longrightarrow \mu(\exists x.A)$. The case for ∀L is similar. □

**Definition 1 (Monadic Terms).** *Monadic terms are built using the following syntax:*

$$m ::= x \mid c \mid f(m)$$

*where, again, $x$ represents a variable and $c$ a constant.*

A crucial part of the definition of our abstraction is the concept of *monadic subterm.*

**Definition 2 (Monadic Subterm).** *The judgment $m \prec t$ (read: $m$ is a monadic subterm of $t$) is defined by the following inference rules:*

$$\frac{}{x \prec x} \qquad \frac{}{c \prec c} \qquad \frac{m \prec t_i}{f_i(m) \prec f(t_1, \ldots, t_n)}$$

*where $x$ is a variable, $c$ is a constant and $n \geq 1$.*

Loosely speaking, a monadic term $m$ is a monadic subterm of a term $t$, written $m \prec t$ if $m$ encodes a maximal path within the syntax tree of $t$, i.e. a directed path from the root to a leaf. For example, the term $f(g(a, b), h(c))$ has three monadic subterms: $f_1(g_1(a))$, $f_1(g_2(b))$ and $f_2(h_1(c))$.

We are particularly interested in a specific class of monadic subterms, those given by the following definition:

**Definition 3.** *A function $\mu$ is* regular *if*

1. *there exists a function $\sigma : F \to \mathbb{N}$ such that $\sigma(f) \leq \Sigma(f)$ for all functions $f$.*
2. *For all functions $f$ of arity $n$ and terms $t_1, \ldots, t_n$ the following equation holds*

$$\mu(f(t_1, \ldots, t_n)) = f_i(\mu(t_i))$$

*where $i = \sigma(f)$.*

The following property is straightforward to prove:

**Lemma 2.** *For any regular function $\mu$ and term $t$, we have $\mu(t) \prec t$.*

Note that the monadic subterms under a regular function $\mu$ are much fewer than compared to monadic subterms in general. For instance, the term $f(f(a, b), c)$ has only two regular monadic subterms: $f_1(f_1(a))$ and $f_2(c)$. The monadic subterm $f_1(f_2(b))$ cannot be generated using a regular function. This might seem overly restrictive, but as we shall see below, regular functions are the only ones that are of interest when proving truthfulness.

The above lemma has a partial converse, as the following theorem shows:

**Theorem 3.** *Let $\mu$ be an endofunction on the set of terms. The following properties are then equivalent:*

1. *For all terms $t$ and $t'$*

$$\mu(t) \prec t \qquad \textit{and} \qquad \mu([t/x]t') = [\mu(t)/x]\mu(t')$$

2. *The function $\mu$ is regular.*

*Proof.* (1)⇒(2) We will show that $\mu$ satisfies the defining equations of a regular function. In particular, we show that for any function $f$ of arity $n$ and any terms $t_1, \ldots, t_n$, the following equality holds:

$$\mu(f(t_1, \ldots, t_n)) = f_i(\mu(t_i))$$

for some value $i$ that only depends on $f$.

Assume that $f$ has arity $n$. Let $x_1, \dots, x_n$ be freshly chosen variables that do not appear in the terms $t_1, \dots, t_n$. Then

$$\mu(f(x_1, \dots, x_n)) \prec f(x_1, \dots, x_n)$$

hence $\mu(f(x_1, \dots, x_n))$ must be of the form $f_i(x_i)$ for some $i$. To see that $i$ cannot depend on the choice of variables $x_1, \dots, x_n$, we note that if we had chosen instead the variables $y_1, \dots, y_n$, we would have the following string of equations

$$\begin{aligned}
\mu(f(y_1, \dots, y_n)) &= \mu([y_1/x_1] \cdots [y_n/x_n] f(x_1, \dots, x_n)) \\
&= [\mu(y_1)/x_1] \cdots [\mu(y_n)/x_n] \mu(f(x_1, \dots, x_n)) \\
&= [y_1/x_1] \cdots [y_n/x_n] f_i(x_i) \\
&= f_i(y_i)
\end{aligned}$$

where $i$ was the index we had determined previously. Thus, the value of this $i$ can only depend on the function $f$, hence we may define a function $\sigma : F \to \mathbb{N}$ by $\sigma(f) = i$. Now, given the terms $t_1, \dots, t_n$, we have the following equalities

$$\begin{aligned}
\mu(f(t_1, \dots, t_n)) &= \mu([t_1/x_1] \cdots [t_n/x_n] f(x_1, \dots, x_n)) \\
&= [\mu(t_1)/x_1] \cdots [\mu(t_n)/x_n] \mu(f(x_1, \dots, x_n)) \\
&= [\mu(t_1)/x_1] \cdots [\mu(t_n)/x_n] f_i(x_i) \\
&= f_i(\mu(t_i))
\end{aligned}$$

where $i = \sigma(f)$, hence $\mu$ satisfies the definition of a regular function.

(2)$\Rightarrow$(1) Assuming $\mu$ is regular, we now need to show that it satisfies the substitution property

$$\mu([t/x]t') = [\mu(t)/x]\mu(t')$$

for all terms $t, t'$. We show this by induction on the structure of $t'$. In the case where $x$ does not occur in $t'$, the result is immediate, as both sides become equal to $\mu(t')$. If $t' = x$, then both sides equal $\mu(t)$. Now, assume $t' = f(t_1, \dots, t_n)$ and that $\sigma(f) = i$. Then

$$\begin{aligned}
\mu([t/x]t') &= \mu([t/x] f(t_1, \dots, t_n)) \\
&= \mu(f([t/x]t_1, \dots, [t/x]t_n)) \\
&= f_i(\mu([t/x]t_i)) \\
&= f_i([\mu(t)/x]\mu(t_i)) && \text{by the induction hypothesis} \\
&= [\mu(t)/x] f_i(\mu(t_i)) \\
&= [\mu(t)/x] \mu(f(t_1, \dots, t_n)) \\
&= [\mu(t)/x] \mu(t').
\end{aligned}$$

This completes the proof. $\square$

With the above theorems and lemmas, we may now prove the main theorem:

**Theorem 4.** *Let $\mu$ be any function satisfying $\mu(t) \prec t$ for all terms $t$ and let $\alpha$ be defined as in Section 3.1. The following properties are then equivalent:*

1. $\alpha(\Gamma) \Longrightarrow \alpha(A)$ *implies* $\mu(\alpha(\Gamma)) \longrightarrow \mu(\alpha(A))$ *in the monadic fragment.*
2. $\mu$ *is regular.*

*Proof.* If $\mu$ is regular, then the substitutivity property follows from Theorem 3, and hence truthfulness by Theorem 2. That the derivation is in the monadic fragment follows from the fact that the image of a regular function is a subset of the set of monadic terms.

Conversely, if $\mu$ is truthful, then $\mu$ satisfies the substitutivity property by Theorem 2, and is thus regular by Theorem 3. □

As a consequence of this theorem we get that, assuming the reasonable requirement that $\mu(t) \prec t$ for all terms $t$, it is exactly the regular functions that induce truthful abstractions.

Note that because we only need to consider regular functions, there is no need to keep track of the indices on the abstracted predicates and functions, as these can always be recovered using the $\sigma$ function defined above.

### 3.3 From Intuitionistic to Classical Logic

The third abstraction mapping the intermediate logic 2 into the abstraction logic is trivial. In fact, if monadic intuitionistic first-order logic were decidable, we would already be done, but it is not [8]. Monadic classical logic, on the other hand is. We chose therefore the obvious complete embedding of intuitionistic logic into classical logic as the third and final abstraction. This abstraction is obviously truthful.

## 4 Examples

First, we will consider the example given in the introduction. Recall that the predicate $D(x, y)$ was given by the following axioms

$$\Gamma = D(z, z), \forall x.\forall y.D(x, y) \supset D(s(x), s(s(y)))$$

There are two abstractions of the sequent

$$\Gamma \Longrightarrow \exists x.D(x, s(s(s(s(z)))))$$

corresponding to the choice of either $\pi(D) = 1$ or $\pi(D) = 2$. In the first case, we get the following abstracted sequent

$$D_1(z), \forall x.\forall y.D_1(x) \supset D_1(s(x)) \longrightarrow \exists x.D_1(x)$$

which is immediately provable by instantiating $x$ by $z$. The second abstraction is

$$D_2(z), \forall x.\forall y.D_2(y) \supset D_2(s(s(y))) \longrightarrow \exists x.D_2(s(s(s(s(z))))))$$

and in this case, a counterexample is found immediately by Spass. It therefore follows that the original sequent is not provable.

As a second example, we consider the following: Let $\Gamma$ be the following context:

$$\begin{aligned}&N(z),\\ &\forall x.N(x) \supset N(s(x)),\\ &\forall x.N(x) \supset S(z,x,x),\\ &\forall x_1,x_2,x_3.N(x_1) \wedge N(x_2) \wedge N(x_3) \wedge S(x_1,x_2,x_3) \supset S(s(x_1),x_2,s(x_3)).\end{aligned}$$

The two predicates $N$ and $S$ specify the Peano numerals and triples satisfying $x_1 + x_2 = x_3$ respectively.

A basic lemma about sums is that zero is a right identity for addition, i.e. that the following sequent is provable

$$\Gamma \Longrightarrow \forall x.N(x) \supset S(x,z,x)$$

In order to prove this sequent, we would need induction that is not available to us in plain first-order logic. There are three different abstractions of the above sequent, based on which argument of the predicate $S$ is preserved. As $N$ is already a unary predicate, it is unchanged by these abstractions, hence we elide it in the following:

$$\begin{aligned}&\forall x.N(x) \supset S(z), \forall x_1,x_2,x_3.N(x_1) \wedge N(x_2) \wedge N(x_3) \wedge S(x_1) \supset S(s(x_1))\\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\longrightarrow \forall x.N(x) \supset S(x)\\ &\forall x.N(x) \supset S(x), \forall x_1,x_2,x_3.N(x_1) \wedge N(x_2) \wedge N(x_3) \wedge S(x_2) \supset S(x_2)\\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\longrightarrow \forall x.N(x) \supset S(z)\\ &\forall x.N(x) \supset S(x), \forall x_1,x_2,x_3.N(x_1) \wedge N(x_2) \wedge N(x_3) \wedge S(x_3) \supset S(s(x_3))\\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\longrightarrow \forall x.N(x) \supset S(x)\end{aligned}$$

Of these three abstracted sequents, the first sequent (and only this sequent) is unprovable, hence the unabstracted sequent is not provable, as one would expect.

## 5 Experimental Results

To test our abstraction, we applied it to a selection of problems from the ILTP[10] problem library. This is a rather coarse test, as it corresponds to only using the abstraction to disprove the original goal, and not the subgoals that are encountered during proof search. We feel, however, that even this relatively limited test shows the efficacy of this way of disproving sequents.

The abstraction was implemented as a transformation for the `tptp2X` tool, which is part of the TPTP[12] problem library. By doing this, we were able to leverage the pre-existing methods for parsing and printing problems in the TPTP format.

We applied the abstraction to a selection of ILTP problems that were flagged as Theorem, Non-Theorem or Unsolved. The resulting monadic problems were then tested with the Spass 3.5 theorem prover running on a laptop with a 2.66GHz processor and 4GB memory. A time limit of 10 seconds was set for the theorem prover. The results are summarized in Table 1.

| Status | Problems | Abstractions | Proved | Disproved | Timed out |
|---|---|---|---|---|---|
| Theorem | 167 | 7706 | 167 | 0 | 0 |
| Non-Theorem | 56 | 110 | 50 | 6 | 0 |
| Unsolved | 78 | 14855 | 5 | 59 | 14 |

**Table 1.** Experimental results.

A problem is tallied in the Disproved column if at least one abstraction was disproved, and in the Timeout column if there were attempts that timed out and no attempts that found a disproof. By testing our implementation on problems that are known to be theorems, we have an empirical verification that our implementation is indeed a truthful abstraction.

To give an accurate view of the usefulness of the abstraction, we have only included the results where we were able to test all abstractions of a given problem. This means that we have left out several problems from the NLP problem set, for which the number of abstractions in some cases exceeded 100,000. This should not be interpreted to mean that our implementation cannot be used for these problems, but rather that we thought it more important to give a complete set of results for the chosen problems. We also left out problems where there was no conjecture to prove or disprove.

It may seem surprising that it was not possible to disprove 50 of the 56 non-theorems tested. This is in part because 48 of these problems were already within the monadic fragment. Also, most of these problems are provable in classical logic, and can therefore not be disproved by our abstractions.

For each problem, finding a single abstraction without proof is enough to disprove the entire problem. To give a better view of how the three possible outcomes are distributed, we have collected a representative sample of the problems from the Unsolved problem set in Table 2. This table shows for each problem how many proof attempts resulted in the three possible outcomes.

## 6 Conclusion and Future Work

In this paper we have presented a family of truthful abstractions from intuitionistic first-order logic to monadic classical first-order logic. We have characterized

| Problem | Disproved | Proved | Timed out |
|---|---|---|---|
| KRS173 | 2 | 0 | 0 |
| SWV016 | 0 | 288 | 0 |
| SWV018 | 0 | 288 | 0 |
| SYN322 | 2 | 0 | 0 |
| SYN330 | 0 | 2 | 0 |
| SYN344 | 1 | 1 | 0 |
| SYN419 | 1022 | 0 | 2 |
| SYN420 | 973 | 0 | 51 |
| SYN421 | 997 | 0 | 27 |
| SYN422 | 999 | 0 | 25 |
| SYN423 | 379 | 0 | 645 |
| SYN424 | 201 | 0 | 823 |
| SYN425 | 963 | 0 | 61 |
| SYN426 | 164 | 0 | 860 |
| SYN427 | 244 | 0 | 780 |
| SYN428 | 614 | 0 | 410 |
| SYN429 | 753 | 0 | 271 |
| SYN513 | 0 | 32 | 0 |
| SYN514 | 32 | 0 | 0 |
| SYN515 | 32 | 0 | 0 |
| SYN516 | 32 | 0 | 0 |
| SYN517 | 32 | 0 | 0 |
| SYN518 | 3 | 24 | 5 |
| SYN519 | 0 | 26 | 6 |
| SYN520 | 0 | 22 | 10 |
| SYN521 | 32 | 0 | 0 |
| SYN540 | 8 | 24 | 0 |
| SYN541 | 32 | 0 | 0 |
| SYN544 | 181 | 32 | 43 |
| SYN545 | 207 | 0 | 49 |
| SYN546 | 178 | 0 | 78 |
| SYN547 | 256 | 0 | 0 |

**Table 2.** Distribution of abstraction results.

the shape of a possible truthful abstractions and validated the usefulness of the technique experimentally.

Although these abstractions were motivated using proof search in intuitionistic logic, they apply equally to classical first-order logic, hence there might be some benefit in applying these abstractions to proof search in classical first-order logic as well.

One drawback of using the abstractions described in this paper is that a single sequent may have a large number of possible abstractions. Thus, to disprove a sequent it may be necessary to attempt disproofs of many more abstracted sequents in the hope of finding a single disproof. Clearly this approach can be parallelized in the trivial way of simply running all of these proof searches in parallel, but there might be gains to be had from either detecting abstracted sequents that are trivially true or combining several abstractions into a single abstraction.

Related to our work is the work on the Scott system [5] that combines the tableau method as model generation with automated theorem proving. The tableau method not only detects unsatisfiability of the negated conjecture but also generates models for it. This is similar to the use of model generating systems during refutation proofs. Thus, certain classes of false conjectures can be detected by generating counter-models. However, the relationship between these classes and the class characterized by the procedure presented in this paper is unclear yet and is left for future work.

Our work can be seen as an application of *Unsound Theorem Proving* [7] to intuitionistic first-order logic. In this paper we have only explored a small subset of all the decidable fragments of first-order logic. There are many more such fragments, and the overall methodology may apply to these fragments as well.

Another point for future work would be to test this approach more thoroughly on the ILTP[10] and TPTP[12] problem libraries to see what proportion of unprovable problems may be refuted by using our decision procedure.

Finally, using the compositionality of truthful abstractions, it might be interesting to extend this approach to “larger” source logic, for instance the meta-logic of the Twelf theorem prover[9].

## References

1. J. Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297, 1992.
2. K. Claessen and N. Sorensson. New techniques that improve mace-style finite model finding. In Peter Baumgartner and Christian Fermueller, editors, *Proceedings of the CADE-19 Workshop: Model Computation - Principles, Algorithms, Applications*, Miami, USA, 2003.
3. Fausto Giunchiglia and Toby Walsh. A theory of abstraction. *Artif. Intell.*, 57(2-3):323–389, 1992.
4. Yu. Gurevich. The decision problem for the logic of predicates and of operations. *Algebra and Logic*, 8:160–174, 1969. 10.1007/BF02306690.
5. Kahlil Hodgson and John Slaney. Tptp, casc and the development of a semantically guided theorem prover. *AI Commun.*, 15:135–146, August 2002.

6. Chuck Liang and Dale Miller. Focusing and polarization in intuitionistic logic. In Jacques Duparc and Thomas Henzinger, editors, *Computer Science Logic*, volume 4646 of *Lecture Notes in Computer Science*, pages 451–465. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-74915-8.
7. Christopher Lynch. Unsound theorem proving. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *Computer Science Logic*, volume 3210 of *Lecture Notes in Computer Science*, pages 473–487. Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-30124-0_36.
8. Jekeri Okee. A semantical proof of the undecidability of the monadic intuitionistic predicate calculus of the first order. *Notre Dame J. Formal Logic*, 16:552–554, 1975.
9. Frank Pfenning and Carsten Schürmann. System description: Twelf — a meta-logical framework for deductive systems. In *Automated Deduction — CADE-16*, volume 1632 of *Lecture Notes in Computer Science*, pages 679–679. Springer Berlin / Heidelberg, 1999.
10. Thomas Raths, Jens Otten, and Christoph Kreitz. The iltp problem library for intuitionistic logic. *Journal of Automated Reasoning*, 38:261–271, 2007.
11. Alexandre Riazanov and Andrei Voronkov. The design and implementation of vampire. *AI Commun.*, 15(2-3):91–110, 2002.
12. Geoff Sutcliffe. The tptp problem library and associated infrastructure. *Journal of Automated Reasoning*, 43:337–362, 2009.
13. Christoph Weidenbach, Dilyana Dimova, Arnaud Fietzke, Rohit Kumar, Martin Suda, and Patrick Wischnewski. Spass version 3.5. In Renate A. Schmidt, editor, *CADE*, volume 5663 of *Lecture Notes in Computer Science*, pages 140–145. Springer, 2009.