Type Class and Logic Programming with Term Matching

Peng Fu

November 17, 2014

1 Introduction

1.1 Logic Programming and Intuitionistic Sequent Calculus

In Section 13.4, Girard's famous [1], he discuss the connection between LP and the intuitionistic sequent calculus. Here I am just rephrasing what he mention in his work:

In the case of LP, we know that Prolog program can be viewed as atomic intuitionistic sequents(also called *Horn clauses*) $\underline{A} \vdash B$, this in LP shape would be something like $B \leftarrow \underline{A}$. In this note, we use $\underline{A} \Rightarrow B$ to better connect with type class notation. The aim of LP is to prove goal, namely atomic sequents of the form $\Rightarrow B$.

If we confine ourself to *atomic* sequents as proper axioms, then there is no need for the logical rules.

So we have the following:

- instances of proper axiom: $\underline{A} \Rightarrow B$.
- identity axioms $A \Rightarrow A$ with A atomic.
- cut rule.
- the structure rules.

Lemma 1. *Weakening and contraction rules are redunant, identity axioms are useless.*

Proposition 1. *To prove a goal, we only need to use cut rule together with the proper axioms.*

So we see LP is really a true intuitionistic logic with cut rule at its heart as inference engine.

1.2 Term matching v.s. Unification

Now that we know that the shape LP is intuitionistic logic with cut, let us focus now on two different strategies to search for possible cut. Let us remind ourself the cut rule:

$$\frac{\underline{A} \Rightarrow E \quad \underline{B}, E \Rightarrow C}{\underline{A}, \underline{B} \Rightarrow C}\ \mathit{cut}$$

Note that the E in the two premises are idenitical.

Proof Search by unification. This is LP traditionally being formulated. In this case a query $P(x)$ is actually searching for derivation of $\exists x.P(x)$. We said unification has an existential commitment. The procedure of proof search roughly speaking is: Given a goal G, try to unify G with the head formula in the axioms, if there is an axiom's head that is unifiable with G then we apply the substitution to the body of the axiom, and invoke the same proof search procedure to each formula in the body.

Proof Search by term matching. This is the strategy LP rarely considered. In this case a query $P(x)$ is actually searching for derivation of $\forall x.P(x)$. We said term matching has an universal commitment. The procedure of proof search roughly speaking is: Given a goal G, try to match the head formula in the axioms against G, if there is an axiom's head that is matchable to G then we apply the substitution to the body of the axiom, and invoke the same proof search procedure to each formula in the body.

Note that the above description is rather informal, several subtleties will be explained later for example, parallelization and the treatment of existential variable at term-matching. Let us now see a simple example:
Given a query $P(x)$ and axioms $\Rightarrow Q(a)$ and $Q(x) \Rightarrow P(x)$, where a is a constant and x is first order unification variable. The query $P(x)$ in LP with unification would first try to unify $P(x)$ with $P(x)$, which produce identity substitution, and then replace goal $P(x)$ with $Q(x)$, which is unified with $Q(a)$, thus produce substitution $[a/x]$, and current goal $Q(x)$ get replaced to empty. So we found a derivation for $\exists x.P(x)$:

$$\frac{\Rightarrow Q(a) \quad [a/x]Q(x) \Rightarrow [a/x]P(x)}{\Rightarrow P(a)}\ cut$$

Now let use using the term-matching to do proof search, for the same example, with query $P(x)$, we find out that $Q(x) \Rightarrow P(x)$'s head $P(x)$ can be matched to $P(x)$, with an identity substitution, thus we produce a new goal $Q(x)$, now we can't find any axioms head that is matchable to $Q(x)$, note that $Q(a)$ can not match to $Q(x)$, so we fail to find a derivation for $\forall x.P(x)$. We look at the axioms again it would make sense, since there is simply not enough information for us to prove the universal $\forall x.P(x)$.

1.3 Type Class and LP with Term Matching

In this section we want to argue that LP with term matching is more suitable for type inference with type class compare to LP with unification. Consider the instance delaration of list:

```
instance Eq x => Eq (List x) where ...
```

This instance delaration will generate a single piece of logic program: Eq $x \Rightarrow$ Eq (List x). With this single piece of logic program, if we use unification, a query Eq(List x) will result in an infinite loop, since x is a unification variable. But LP with term matching will simply return an answer no. Since, query on Eq(List x) will generate a new query Eq x, which no any rule can match to, notice here x is a term-matching variable. This is the behaviour one would expect in Haskell for example. In another word, in type class inference, query has universal commitment.

1.4 Evidence Construction and LP with Term Matching

So the next question would be: how exactly the evidence construction (in type class) connect to LP with term matching? We want the answer to be, they are really the same thing. Let us look at a simple example:

```
class Eq x where
  eq :: Eq x => x -> x -> Bool

instance Eq Nat where
  eq = eq_Nat

instance Eq x => Eq [x] where
  eq (x:xs) (y:ys) = (eq x y) && eq xs ys
  eq [] [] = True
  eq _ _ = False
test = eq [1] [2]
```

Ideally, the class `Eq` declaration will generate the following data type and its selector:

```
data Eq x = Eq (x -> x -> Bool)
eq :: Eq x -> (x -> x -> Bool)
eq (Eq f) = f
```

What would the instance declaration of `Eq Nat` generated then? Actually, it will generate an evidence for `Eq Nat`, namely:

```
e :: Eq Nat
e = Eq (eq_Nat)
```

For the instance declaration of `Eq [x]`, here is an possible code that it will generate:

```
f :: Eq x -> Eq [x]
f e = Eq (g e)
 where g e (x:xs) (y:ys) = (eq e x y) && g e xs ys
       g e [] [] = True
       g e _ _ = False
```

So after this process of code generation, we have the following LP:

```
e ::  => Eq Nat
f :: Eq x => Eq [x]
```

Note that here we use `e` and `f` to as label annotate the two axioms. And since we are in intuitionistic logic anyway, `=>` can be read as functional arrow `->`. Now we can type check `test`, it will invoke a query `Eq [Nat]`, also, it needs an evidence for `Eq [Nat]`. So how are we going to construct this evidence? Easy, by using LP with term matching, we will have following derivation:

$$\frac{e :: \Rightarrow \text{Eq Nat} \quad f :: \text{Eq Nat} \Rightarrow \text{Eq[Nat]}}{f\ e :: \Rightarrow \text{Eq[Nat]}}$$

So by LP with term-matching, we successfully construct the evidence for `Eq [Nat]`, which is `f e`.

2 LP with Term Matching

2.1 A Proof System for LP

Definition 1.
Term t ::= $x \mid \hat{x} \mid f(t_1, ..., t_n)$
Atomic Formula A, B, C, D ::= $P(t_1, ..., t_n)$
PreFormula F ::= $A_1, ..., A_n \Rightarrow A$
Quantified Formula Q ::= $\forall \underline{x}.F$
Proof Term/Evidence p, e ::= $a \mid \lambda a.e \mid e\ e'$

Note that we call x (universal) variable, $\hat{x}$ is called (existential variable) *covariable*, and $x \not\equiv \hat{x}$. We use covariable and existential variable interchangely in this note. Note that all the terms are first order term, for function of arity zero, we call it *constant.* and all the formulas are first order formula, i.e. no quantifying over predicate. The P in $P(t_1, ..., t_n)$ is called *predicate.* Predicate of arity zero is called *proposition.* We use $\underline{A}$ to denote $A_1, ..., A_n$, when we do not care about the number n, if n is zero, then we write $\Rightarrow B$. Note that B is an atomic formula, but $\Rightarrow B$ is a preformula, they are **different**. We use $\forall \underline{x}.F$ to mean quantifying over all the free variable and covariable in F.

Note that proof term is intended to use to denote the proof of a formula, please do not confuse them with term. Also, proof term is higher order in the sense that it allows lambda bind variable. We use a, b, c to denote proof term variable.

Definition 2 (Well-formed Formula). *A preformula $A_1, ..., A_n \Rightarrow B$ is well-formed formula if* $\mathrm{coFV}(B) = \emptyset$.

Note that the above definition includes the case when n is zero. In this note when we write a preformula, we mean it is well-formed. coFV return the set of covariables in a atomic formula and $\mathrm{coFV}(A_1, ..., A_n)$ is a shorthand for $\mathrm{coFV}(A_1) \cup ... \cup \mathrm{coFV}(A_n)$.

Definition 3 (Proof System).

$$\frac{e_1 : \underline{A} \Rightarrow D \quad e_2 : \underline{B}, D \Rightarrow C}{\lambda \underline{a}.\lambda \underline{b}.(e_2\ \underline{b})\ (e_1\ \underline{a}) : \underline{A}, \underline{B} \Rightarrow C}\ cut \qquad \frac{e : \forall \underline{x}.F \quad \mathrm{coFV}(\underline{t}) = \emptyset}{e : [\underline{t}/\underline{x}]F}\ inst \qquad \frac{e : F}{e : \forall \underline{x}.F}\ gen \qquad \frac{}{a : Q}\ axiom$$

The inst and the gen rule apply to both variable and covariable. We use $\lambda \underline{a}.e$ to denote $\lambda a_1 ... \lambda a_n.e$, when n is zero, we it denotes just e. Similarly, $e\ \underline{b}$ denote $e\ b_1\ ...\ b_n$, when n is zero, it denotes just e. In the cut rule, we require $\underline{a}, \underline{b}$ do not appear free in e_1, e_2. We use Φ to denote a set of axioms.

2.2 LP by Term Matching

Definition 4 (Term Matching). *We simultaneously define $A \mapsto_\sigma B$, where* $\mathrm{coFV}(A, B) = \emptyset$*, A is matchable to B with a substitution σ and $t \mapsto_\sigma t'$, t is matchable to t' with a substitution σ.*

$$\frac{\{t_i \mapsto_{\sigma_i} t'_i\}_{i \in \{1,...,n\}}}{P(t_1, ..., t_n) \mapsto_{\sigma_1 \cup ... \cup \sigma_n} P(t'_1, ..., t'_n)} \qquad \frac{\{t_i \mapsto_{\sigma_i} t'_i\}_{i \in \{1,...,n\}}}{f(t_1, ..., t_n) \mapsto_{\sigma_1 \cup ... \cup \sigma_n} f(t'_1, ..., t'_n)} \qquad \frac{}{x \mapsto_{[t/x]} t}$$

Note that here we adopt some syntactic conventions to ease the reasoning about term matching. We treat substitution as set and the union of them will invoke a syntactic comparison of two term. For example $[t_1/x] \cup [t_2/x] = [t_1/x]$ if $t_1 \equiv t_2$, else, $[t_1/x] \cup [t_2/x]$ fails; and $[t_1/x] \cup [t_2/y] = [t_1/x, t_2/y]$.

Definition 5 (Co-Term Matching). *We simultaneously define $A \mapsto_{\hat{\sigma}} B$, where* $\mathrm{coFV}(A) \neq \emptyset, \mathrm{coFV}(B) = \emptyset$*, A is comatchable to B with a cosubstitution $\hat{\sigma}$ and $t \mapsto_{\hat{\sigma}} t'$, t is comatchable to t' with a cosubstitution $\hat{\sigma}$.*

$$\frac{}{\hat{x} \mapsto_{[t/\hat{x}]} t} \qquad \frac{}{x \mapsto_{\emptyset} t}$$

$$\frac{\{t_i \mapsto_{\hat{\sigma}_i} t'_i\}_{i \in \{1,...,n\}}}{P(t_1, ..., t_n) \mapsto_{\hat{\sigma}_1 \cup ... \cup \hat{\sigma}_n} P(t'_1, ..., t'_n)} \qquad \frac{\{t_i \mapsto_{\hat{\sigma}_i} t'_i\}_{i \in \{1,...,n\}}}{f(t_1, ..., t_n) \mapsto_{\hat{\sigma}_1 \cup ... \cup \hat{\sigma}_n} f(t'_1, ..., t'_n)}$$

$$\frac{f \not\equiv g}{f(t_1, ..., t_n) \mapsto_{\emptyset} g(t'_1, ..., t'_n)} \qquad \frac{P \not\equiv Q}{P(t_1, ..., t_n) \mapsto_{\emptyset} Q(t'_1, ..., t'_n)}$$

Definition 6 (Failure of Co-term Matching). *We say A fails to comatch with B if $A \mapsto_\emptyset B$. So when we write $A \mapsto_{\hat{\sigma}} B$, we always mean $\hat{\sigma} \neq \emptyset$.*

We will model LP with term matching by a reduction relation between sets of atomic formula. We define $\hat{\sigma}$ to be the substitution obtained from σ that substitute only the existential variables.

Definition 7 (Reduction by Term Matching). *Let Φ denote the set of axioms, we define the following relation.*

- *cut reduction*

  $\Phi \vdash \{A_1, ..., A_i, ..., A_n\} \rightsquigarrow_a \{A_1, ..., \sigma B_1, ..., \sigma B_m, ..., A_n\}$*, if* $\mathrm{coFV}(A_1, ..., A_i, ..., A_n) = \emptyset$*, and there exists $a : \forall \underline{x}.B_1, ..., B_n \Rightarrow C \in \Phi$ such that $C \mapsto_\sigma A_i$.*

- *existential reduction*

  $\Phi \vdash \{A_1, ..., A_i, ..., A_n\} \rightsquigarrow_a \{\hat{\sigma} A_1, ..., \hat{\sigma} A_i, ..., \hat{\sigma} A_n\}$*, if there exists an A_i such that* $\mathrm{coFV}(\{A_i\}) \neq \emptyset$*, and there is an $a : \forall \underline{x}.B_1, ..., B_n \Rightarrow C \in \Phi$ and $A_i \mapsto_{\hat{\sigma}} C$.*

2.3 Soundness

Definition 8. *We say $\Phi \vdash \{A\}$ is reducible to empty set if there exists a reduction path from $\{A\}$ to $\emptyset$.*

Lemma 2 (Orthogonal)**.** *The cut reduction and the existential reduction are orthogonal, i.e. given a set of atomic formulas and a set of axioms, one can only choose one reduction to reduce the set of formulas at a time, not both.*

Lemma 3 (Finiteness of Existential Reduction)**.** *Existential reduction is finite, i.e. there can not be an infinite reduction path consisted of only exitential reductions.*

Proof. We know that the head of an axiom $a : \forall \underline{x}. B_1, ..., B_n \Rightarrow C$, namely, C, can not contain any existential variables, by well-formness; so existential reduction strictly decrease the number of existential variables. □

Lemma 4. *If $\Phi \vdash \{A_1, ..., A_n\}$ is reducible to emptyset, where $\mathrm{coFV}(A_1, ..., A_i, ..., A_n) = \emptyset$, then there exists proofs $e_1 : \forall \underline{x}. \Rightarrow A_1, ..., e_n : \forall \underline{x}. \Rightarrow A_n$ given axioms Φ.*

Proof. By induction on the length of the reduction.

- Base Case. Suppose the length is one, namely, $\Phi \vdash \{A\} \rightsquigarrow_a \emptyset$. It must be the cut reduction, thus there exists $(a : \forall \underline{x}. \Rightarrow C) \in \Phi$, such that $C \mapsto_\sigma A$. So we have $a :\Rightarrow \sigma C$ by the inst rule, thus $a :\Rightarrow A$, hence $a : \forall \underline{x}. \Rightarrow A$ by the gen rule.
- Step Case. Suppose $\Phi \vdash \{A_1, ..., A_i, ..., A_n\} \rightsquigarrow_a \{A_1, ..., \sigma B_1, ..., \sigma B_m, ..., A_n\} \rightsquigarrow ... \rightsquigarrow \emptyset$, where $a : \forall \underline{x}. B_1, ..., B_n \Rightarrow C$ and $C \mapsto_\sigma A_i$.
  - If $\mathrm{coFV}(B_1, ..., B_m) = \emptyset$, then by IH, we know that there exists proofs $e_1 : \forall \underline{x}. \Rightarrow A_1, ..., p_1 : \forall \underline{x}. \Rightarrow \sigma B_1, ..., p_m : \forall \underline{x}. \Rightarrow \sigma B_m, ..., e_n : \forall \underline{x}. \Rightarrow A_n$. We can construct a proof $e_i = a\ p_1\ ...p_m$ with $e_i : \forall \underline{x}. \Rightarrow A_i$, by first use inst to instantiate the quantifiers of a, then applying the cut rule m times. So we have a proof for each of A_i in $\{A_1, ..., A_n\}$.
  - If $\mathrm{coFV}(B_1, ..., B_m) \neq \emptyset$, by lemma 2, 3, we know the shape of the reduction must be of the form $\Phi \vdash \{A_1, ..., A_i, ..., A_n\} \rightsquigarrow_a \{A_1, ..., \sigma B_1, ..., \sigma B_m, ..., A_n\} \rightsquigarrow^* \{A_1, ..., \hat{\delta}\sigma B_1, ..., \hat{\delta}\sigma B_m, ..., A_n\} \rightsquigarrow ... \rightsquigarrow \emptyset$, where the domain of $\hat{\delta}$ are exitential variables, $\rightsquigarrow^*$ is finite existential reductions and $\mathrm{coFV}(\hat{\delta}\sigma B_1, ..., \hat{\delta}\sigma B_m) = \emptyset$. Since $a : \forall \underline{x}. B_1, ..., B_n \Rightarrow C$, by inst rule we have $a : \hat{\delta} B_1, ..., \hat{\delta} B_n \Rightarrow C$, we have this because C does not contain existential variables. We apply inst rule again with the σ in $C \mapsto_\sigma A_i$, so we have $a : \sigma\hat{\delta} B_1, ..., \sigma\hat{\delta} B_n \Rightarrow \sigma C$, thus $a : \sigma\hat{\delta} B_1, ..., \sigma\hat{\delta} B_n \Rightarrow A_i$. We know that $\hat{\delta}\sigma = \sigma\hat{\delta}$, so $a : \hat{\delta}\sigma B_1, ..., \hat{\delta}\sigma B_n \Rightarrow A_i$. By IH, we know there exists proofs $e_1 : \forall \underline{x}. \Rightarrow A_1, ..., p_1 : \forall \underline{x}. \Rightarrow \hat{\delta}\sigma B_1, ..., p_m : \forall \underline{x}. \Rightarrow \hat{\delta}\sigma B_m, ..., e_n : \forall \underline{x}. \Rightarrow A_n$. So applying the cut rule m times give us $e_i = a\ p_1 ... p_m$ and $e_i : \forall \underline{x}. \Rightarrow A_i$.

□

Theorem 1 (Soundness)**.** *If $\Phi \vdash \{A\}$ is reducible to emptyset, where $\mathrm{coFV}(A) = \emptyset$, then there exists proofs $e : \forall \underline{x}. \Rightarrow A$ given axioms Φ.*

Proof. By lemma 4. Here we want to emphasis that since we have a constructive proof, we know how to actually construct such proof e. □

2.4 Completeness

Definition 9. *For substitution σ, δ whose domain are variable, we define $\delta \cdot \sigma$ by induction on length of σ:*

- $\delta \cdot [] = []$
- $\delta \cdot ([t/x] \cup \sigma) = [\delta t/x] \cup \delta \cdot \sigma$

Lemma 5. *$\delta \cdot \sigma(A) = \delta(\sigma A)$ and $\hat{\sigma}(\delta A) = \delta(\hat{\sigma} A)$.*

Lemma 6.

- *If $A \mapsto_{\sigma} B$, then $A \mapsto_{\delta \cdot \sigma} \delta B$.*
- *If $t \mapsto_{\sigma} t'$, then $t \mapsto_{\delta \cdot \sigma} \delta t'$.*

Lemma 7. *Let δ be a substitution whose domain are variables.*

- *If $A \mapsto_{\hat{\sigma}} B$, then $\delta A \mapsto_{\hat{\sigma}} B$.*
- *If $t \mapsto_{\hat{\sigma}} t'$, then $\delta t \mapsto_{\hat{\sigma}} t'$.*

Lemma 8. *If $\Phi \vdash \{A_1, .., A_i, ..., A_n\} \rightsquigarrow_a \{A_1, ..., \sigma B_1, ..., \sigma B_m, ..., A_n\}$ with $a : \forall \underline{x}.B_1, ..., B_m \Rightarrow C$ and $C \mapsto_{\sigma} A_i$. Then for a substitution δ where the domain of δ are variable, then $\Phi \vdash \{\delta A_1, .., \delta A_i, ..., \delta A_n\} \rightsquigarrow_a \{\delta A_1, ..., \delta\sigma B_1, ..., \delta\sigma B_m, ..., \delta A_n\}$.*

Proof. By lemma 6, we know that $C \mapsto_{\sigma} A_i$ implies $C \mapsto_{\delta \cdot \sigma} \delta A_i$, thus by cut reduction we have $\Phi \vdash \{\delta A_1, .., \delta A_i, ..., \delta A_n\} \rightsquigarrow_a \{\delta A_1, ..., \delta\sigma B_1, ..., \delta\sigma B_m, ..., \delta A_n\}$. ☐

Lemma 9. *Suppose $\Phi \vdash \{A_1, ..., A_i, ..., A_n\} \rightsquigarrow_a \{\hat{\sigma} A_1, ..., \hat{\sigma} A_i, ..., \hat{\sigma} A_n\}$, if there exists an A_i such that $\mathrm{coFV}(\{A_i\}) \neq \emptyset$, and there is an $a : \forall \underline{x}.B_1, ..., B_n \Rightarrow C \in \Phi$ and $A_i \mapsto_{\hat{\sigma}} C$. For a substitution δ with domain of variables, we have $\Phi \vdash \{\delta A_1, ..., \delta A_i, ..., \delta A_n\} \rightsquigarrow_a \{\delta\hat{\sigma} A_1, ..., \delta\hat{\sigma} A_i, ..., \delta\hat{\sigma} A_n\}$.*

Proof. By lemma 7, we know that $\delta A_i \mapsto_{\hat{\sigma}} C$, so by existential reduction, we have $\Phi \vdash \{\delta A_1, ..., \delta A_i, ..., \delta A_n\} \rightsquigarrow_a \{\hat{\sigma}\delta A_1, ..., \hat{\sigma}\delta A_i, ..., \hat{\sigma}\delta A_n\} \equiv \{\delta\hat{\sigma} A_1, ..., \delta\hat{\sigma} A_i, ..., \delta\hat{\sigma} A_n\}$. ☐

Lemma 10. *If $\Phi \vdash \{C\} \rightsquigarrow^* \{\underline{B}, D\}$ and $\Phi \vdash \{D\} \rightsquigarrow^* \{\underline{A}\}$, then $\Phi \vdash \{C\} \rightsquigarrow^* \{\underline{B}, \underline{A}\}$. Note that $\mathrm{coFV}(D) = \emptyset$.*

We use $[\forall \underline{x}].F$ to denote F or $\forall \underline{x}.F$.

Lemma 11. *If there exists a proof $e : [\forall \underline{x}].A_1, ..., A_n \Rightarrow B$ and $\mathrm{coFV}(A_1, ..., A_n) = \emptyset$, given axioms Φ, then $\Phi \vdash \{B\} \rightsquigarrow^* \{A_1, ..., A_n\}$.*

Proof. By induction on the derivation of the proof $e : [\forall \underline{x}].A_1, ..., A_n \Rightarrow B$.

- Base Case.

  $$\frac{}{e : \forall \underline{x}.A_1, ..., A_n \Rightarrow B}$$

  By cut reduction we have the result.

- Step Case.

  $$\frac{e : A_1, ..., A_n \Rightarrow B}{e : \forall \underline{x}.A_1, ..., A_n \Rightarrow B}$$

  This case is directly by IH.

- Step Case.

  $$\frac{e : \forall \underline{x}.A_1, ..., A_n \Rightarrow B}{e : \sigma A_1, ..., \sigma A_n \Rightarrow \sigma B}$$

  By IH, we know that $\Phi \vdash \{B\} \rightsquigarrow^* \{A_1, ..., A_n\}$. By lemma 8, 9, we know that $\Phi \vdash \{\sigma B\} \rightsquigarrow^* \{\sigma A_1, ..., \sigma A_n\}$.

- Step Case.

  $$\frac{e_1 : \underline{A} \Rightarrow D \quad e_2 : \underline{B}, D \Rightarrow C}{\lambda \underline{a}.\lambda \underline{b}.(e_2\ \underline{b})\ (e_1\ \underline{a}) : \underline{A}, \underline{B} \Rightarrow C}\ \mathit{cut}$$

  We assume $\mathrm{coFV}(\underline{A}, \underline{B}) = \emptyset$, so we can apply IH to the two premises, we get $\Phi \vdash \{C\} \rightsquigarrow^* \{\underline{B}, D\}$ and $\Phi \vdash \{D\} \rightsquigarrow^* \{\underline{A}\}$. By lemma 10, we have $\Phi \vdash \{C\} \rightsquigarrow^* \{\underline{B}, \underline{A}\}$.

□

Note: the lemma above will not hold if $\text{coFV}(A_1, ..., A_n) \neq \emptyset$.

Theorem 2 (Completeness)**.** *If there exists a proof* $e : \forall \underline{x}. \Rightarrow A$ *given axioms* Φ*, then* $\Phi \vdash \{A\}$ *is reducible to* $\emptyset$*.*

Proof. By lemma 11. □

2.5 What does this mean?

- We have a foundation for LP with term matching.
- We give a true operational view on how to understand the $\Rightarrow$ in intuitionistic sequent $\underline{A} \Rightarrow B$, namely, $\Phi \vdash \{B\} \leadsto^* \{\underline{A}\}$.
- Our soundness theorem will enable us to construct proof/evidence once we know a query is reducible to empty set.
- We now understand type class better.
- Our proof system is very similar to intuitionistic sequent just with cut rule and extrange rule, but our treatment of first order quantifier is of a natural deduction style. The benefits with this formulation is that we have a term assignment system(for producing evidence) and we have an good operational semantics.

References

[1] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and Types.* Cambridge University Press, New York, NY, USA, 1989.