Existential Cover in Constructive Logic

by Sven Nilsen, 2025

In this paper, I show that Intuitionistic Propositional logic extended with axioms of meta-theorem proving using Higher Order Operator Overloading Exponential Propositions, implies Existential Logic. As a consequence, meta-theorem proving in general is not trusted, but for which theories theorems should be graded. I propose a grading 0-4 in meta-strength using 4 axioms.

Higher Order Operator Overloading Exponential Propositions (HOOO EP) finalizes Intuitionistic Propositional Logic (IPL) by extending it with an operator for function pointers, written `^` or `→`:

```
pow_lift : b^a → (b^a)^c
tauto_hooo_imply : (a => b)^c → (a^c => b^c)^true
tauto_hooo_or : (a | b)^c → (a^c | b^c)^true
```

These 3 axioms in HOOO EP enables merging meta- and object-language into one.

This introduces a problem, because usually one interprets truth in IPL by the property that every true statement has some proof. It holds in isolation both for IPL and IPL extended with HOOO EP, but when interpreting IPL extended with HOOO EP relative to IPL, there are true statements which are not provable. It violates the intuition that every expression of some theorem produced with meta-theory, ought to contain some extended operator for the meta-theory. Otherwise, the meta-theory seems too strong for the underlying object-theory.

The most important proposition, that is provable in IPL + HOOO EP, but not IPL, is the following:

!!a | !a   for all `a`

This property holds for all propositions in Existential Logic, which is logically equivalent to:

excm(!a)   for all `a`

Notice that these statemens do not contain `^` or `→`, which is introduced in HOOO EP.

The proof only requires `tauto_hooo_or`. There is no need to use `pow_lift` or `tauto_hooo_imply`.

One might try to address this situation by removing `tauto_hooo_or`.
Yet, on closer inspection, this strategy not sufficient.

Intuitively, this axiom says that when you can prove `a` or `b` in the context `c` only, then you can construct either a function pointer `c → a` or a function pointer `c → b` under no assumptions. It is impossible to imagine a situation where this is not the case.

A proof of the type `a | b` needs either a proof of `a` or a proof of `b`. If `a | b` can be proven under the context `c`, then there must be a way to get either a proof of `a` or a proof of `b` in context `c`.

In general, this intuition sounds valid. However, it is less clear in the case of `(a | b) → (a | b)`. It does not sound as valid that there is a pointer of type `(a | b) → a` or a pointer of type `(a | b) → b`.

To see how this works, think about the situation where one can prove `a | b`, without any additional assumptions. This requires a proof of `a` or a proof of `b`. What assumptions are possible here?

One assumption that is possible is an empty context `true`. Since one can prove `a | b`, this implies being able to construct a function pointer of type `true → a` or a function pointer of type `true → b`.

Another assumption that is possible is `a`. It is trivial to get a function pointer of type `a → a`. The same argument can be used when assuming `b`.

A third assumption that is possible is `a | b`. While one can not distinguish whether the function pointer `(a | b) → a` or the function pointer `(a | b) → b` exists, at least one of them must exists, using the intuition when one assumes either `a` or `b`.

This means, no matter how problematic `tauto_hooo_or` is for our interpretation of IPL extended with HOOO EP, there is no true justification to remove it. The axiom `tauto_hooo_or` has to be part of HOOO EP to finalize IPL. There is no way to work around this issue.

Yet, one can introduce non-logical concepts such as privacy, to argue in favor of `pow_lift` and `tauto_hooo_imply`, but against meta-theorem proving on the strength of `tauto_hooo_or`. This is because `pow_lift` and `tauto_hooo_imply` respects privacy, but `tauto_hooo_or` does not. In a module that exposes `a | b` to the rest of the world, there is no way for the outside world to construct the function pointers returning `a` or `b`. This knowledge is kept secret by the module.

From the perspective of privacy, the `tauto_hooo_or` axiom assumes that the function pointers that might be only knowledgable to specific modules, are available everywhere.

In some languages, e.g. most mainstream programming languages, privacy is possible. However, this does not mean that privacy is the correct design choice in all languages. In a language with no privacy, HOOO EP provides the correct meta-theorem proving framework to reason about theorems. So, whether one should include `tauto_hooo_or` or not, is a matter of language design.

It is not possible to decide what the correct strength of meta-theorem proving should be, as long one does not decide how the language one talks about through logic is implemented. With other words, this is an undecidable problem.

As a consequence, meta-theorem proving in general is not trusted.

This idea has many philosphical implications.

To work around this issue, I propose to grade theorems in meta-strength using 4 axioms:

```
1. pow_lift : b^a → (b^a)^c
2. tauto_hooo_imply : (a => b)^c → (a^c => b^c)^true
3. tauto_hooo_tauto_or : (a | b)^true → (a^true | b^true)^true
4. tauto_hooo_or : (a | b)^c → (a^c | b^c)^true
```

All theorems provable in IPL only, gets the grade 0.
A grade N means that might use axiom N, N – 1, N – 2 etc. down to 0.
This means when using a meta-strength grade on the scale 0-4, one assumes IPL by default.

The axiom 3 is a weaker version of 4, which is useful when reasoning about partial privacy.

Existential Logic does not imply that HOOO EP of grade 4 is used. However, it is possible to grade theorems informally, as a comment. This means, `!!a | !a` for all `a` in Existential Logic might be given the grade 4, because it is provable under HOOO EP of grade 4. A grade is given informally when it is not connected to a particular proof. When given formally, it is connected to the proof.