Informal Theorem Proving

by Sven Nilsen, 2019

In this paper I show that informal theorem proving is more powerful than formal theorem proving.

When doing informal theorem proving, one is permitted to use any formal language[1], or a combination of formal languages to write proofs. We say that informal theorem proving "uses" formal languages. This relation, between two languages, where one uses another, is a binary relation `uses`:

uses : language × language → bool

Every language uses itself:

∀ a { uses(a, a) }

This relation is transitive[2]:

∀ a, b, c { uses(a, b) ∧ uses(b, c) => uses(a, c) }

The statement about how informal theorem proving can be written down formally:

uses("informal theorem proving", _ : formal) : true

formal : language → bool

In first-order logic[3]:

∀ x : formal { uses("informal theorem proving", x) }

Intuition of why every language uses itself

The fact that every language uses itself is justified by the following intuition:

∀ a, { ∃ b { (a ¬= b) ∧ (a ≃ b) } }

For every language, there exists some language which is not equal, but isomorphic.
One can say about the isomorphism between languages that they use each other:

a ≃ b => uses(a, b) ∧ uses(b, a)

By using the transitive property setting `c = a`, one can then show that every language uses itself:

∀ a, b { uses(a, b) ∧ uses(b, a) => uses(a, a) }

More powerful languages

A language `a` is more powerful for theorem proving than `b` if `a` uses `b`, but not vice versa:

more_powerful(a, b) = uses(a, b) ∧¬uses(b, a)

Since every language uses itself, it is not more powerful than itself:

∀ x { ¬more_powerful(x, x) }

No formal language uses all formal languages:

¬∃ x : formal { ∀ y : formal { uses(x, y) } }

Nor do any formal language use an informal language:

¬∃ x : formal, y : ¬formal { uses(x, y) }

If a formal language used an informal language, it would not be formal.

Informal theorem proving is more powerful than formal theorem proving

Since informal theorem proving uses all formal languages, informal theorem proving is informal:

“informal theorem proving” : ¬formal

Informal theorem proving uses all formal languages, but no formal language uses informal theorem proving. Therefore, informal theorem proving is more powerful than any formal language:

∀ x : formal { more_powerful(“informal theorem proving”, x) }

When a formal language is used for theorem proving, it can be formally written as a function:

theorem_proving : language → bool

Theorem proving is a function of some context.
This requires higher order reasoning using higher order functions.
One can use Higher Order Operator Overloading[4] (HOOO) to simplify higher order reasoning.

When some language is used for theorem proving, the languages it uses,
under HOOO, are also used for theorem proving:

∀ x, y { theorem_proving(x) ∧ uses(x, y) => theorem_proving(y) }

This means, in the context of informal theorem proving, formal languages are used for theorem proving, which makes informal theorem proving more powerful than formal languages used for theorem proving. Therefore, informal theorem proving is more powerful than formal theorem proving.

References:

[1] “Formal language”
Wikipedia
https://en.wikipedia.org/wiki/Formal_language

[2] “Transitive relation”
Wikipedia
https://en.wikipedia.org/wiki/Transitive_relation

[3] “First-order logic”
Wikipedia
https://en.wikipedia.org/wiki/First-order_logic

[4] “Higher Order Operator Overloading”
Sven Nilsen, 2018
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/higher-order-operator-overloading.pdf