Seshatic Queenity

by Sven Nilsen, 2022

In this paper I present a Seshatic relation that "points directly".

In the paper "Path Semantical Quality"[1], I introduced a partial equivalence relation that lifts biconditions with symbolic distinction. Since Intuitionistic Propositional Logic (IPL)[2] does not have symbolic distinction, I created a model for the "Prop" library[3] (written in Rust[4]). This means that instead of using symbolic distinction, one adds manual assumptions of the kind:

(a == b) => (a ~~ b)

This model works very well and might help to explore proofs using quality in more detail. One particular area of interest is Seshatism[5]. Seshatism are logical structures that have the assumption:

¬(a ~~ a)

During a conversation with Sirisys[6] on Discord, we discussed a Seshatic relation such that each person `X` has that relation to themselves, given that some person `Y` has that relation to `X`. This property is similar to how the Loop Witness `X ~~ X` relates to the Product Witness `X ~~ Y`[7]. One might think that this would make it a Platonic relation instead of Seshatic. However, the Seshatic relation we discussed, had an additional property: Other people `Y` can not have the same relation to themselves as to the person `X`. Formalized in Avalog[8]:

```
(X, Sq'(X)) :- Sq : seshatic_queenity, (Y, Sq'(X)).
uniq Sq :- Sq : seshatic_queenity.
```

The formalization in Avalog looks simple, but it is also subtle. Let us take a look at an example:

```
sq : seshatic_queenity
sq'(X) : p :- (Y, sq'(X)).
(b, sq'(a))
```

This proves `p(b) = sq'(a)`. When adding `(b, sq'(b))`, one can prove `amb` (ambiguity).

However, it is possible to assign different roles:

```
sq : seshatic_queenity
sq'(a) : p1
sq'(b) : p2
(b, sq'(a))
(b, sq'(b))
```

Now, `p1(b) = sq'(a)`, which does not collide with `p2(b) = sq'(b)`.

The formalization in Avalog is helpful, but it would also be nice to have a model in IPL, to be used in the Prop library to explore proofs using quality.

The first step to developing a model in IPL, is to get rid of roles and avatars. This is needed since IPL only contains propositions. So, the Seshatic relation will be a binary operator.

The binary operator is written in one of the following ways:

a ¬> b a !> b seshatic_queenity(a, b) sq(a, b)

I call it "Seshatic Queenity", using the intuition that of a queen (ant or human):

- Any queen is queen to herself (`(a ¬> b) => (b ¬> b)`)
- There is no middle-queen (`(a ¬> b) => ¬(a ¬> a)` when `(a == b) => (a ~~ b)`)

The `¬>` relation is transitive, which might seem counter-intuitive. When there are no proofs of symbolic distinction `(a == b) => (a ~~ b)`, this behaves as a normal transitivity property. However, with symbolic distinction, the transitivity property actually proves the opposite. There can be no indirect pointing, because it proves `false`. Hence, Seshatic Queenity is said to "point directly".

Just like quality projects down onto equality, I chose to make Seshatic Queenity project down into implication. Therefore, one has the following:

(a ¬> b) => (a => b)

The idea is that Seshatic Queenity can be used as a stronger assumption than implication. This is useful when modeling stuff like simple types, which has no type hierarchy.

Two other properties inherited from implication, that allows manipulation of the arguments:

(a ¬> b) ∧ (a = c) => (c ¬> b)

(a ¬> b) ∧ (b = c) => (a ¬> c)

Last, one needs to imply Seshatism to make the relation Seshatic:

(a ¬> b) => ¬(a ~~ b)

This completes the model of Seshatic Queenity in IPL.

An analysis[9] of Seshatic Queenity gives the following language in Joker Calculus[10]:

Seshatic (Platonism, Joker Seshatism)

The depth language is Seshatic Platonism due to the terminality condition of "pointing directly". Terminality conditions are related to Platonism. However, this relation is directional, so it is also Seshatic. The surface language is Seshatic Joker Seshatism to distance itself from Platonism[11]. This way, there are some aspects that are related to Platonism, but overall the relation is Seshatic.

By replacing `Seshatism` with `Platonism` and vice versa, one gets the following language:

Platonic (Seshatism, Joker Platonism)

This might be thought of as an assumption that denies Seshatic Queenity in general:

∀ a { ¬(a ¬> a) }

Vaguely, this can be translated into non-terminality condition, an infinite Platonic construction.

References:

[1] “Path Semantical Quality”
Sven Nilsen, 2021
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip2/path-semantical-quality.pdf

[2] “Intuitionistic logic”
Wikipedia
https://en.wikipedia.org/wiki/Intuitionistic_logic

[3] “Prop”
AdvancedResearch – Propositional logic with types in Rust
https://github.com/advancedresearch/prop

[4] “Rust”
Rust programming language
https://www.rust-lang.org/

[5] “Seshatism”
Sven Nilsen, 2021-2022
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip2/seshatism.pdf

[6] “SIRISYS”
Twitter
https://twitter.com/SIRISYSPrime

[7] “Seshatism vs Platonism”
AdvancedResearch – Summary page on Avatar Extensions
https://advancedresearch.github.io/avatar-extensions/summary.html#seshatism-vs-platonism

[8] “Avalog”
AdvancedResearch – An implementation of Avatar Logic with a Prolog-like syntax
https://github.com/advancedresearch/avalog

[9] “Joker Calculus Analysis: Seshatic Queenity”
AdvancedResearch – An implementation of Joker Calculus in Rust
https://github.com/advancedresearch/joker_calculus/issues/26

[10] “Joker Calculus”
Daniel Fischer, William Alexander Morris, Sven Nilsen, 2021
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip2/joker-calculus.pdf

[11] “What does `Seshatic Joker Seshatism` mean?”
AdvancedResearch – An implementation of Joker Calculus in Rust
https://github.com/advancedresearch/joker_calculus/issues/19