Agnostic Language Bias

by Sven Nilsen, 2024

In this paper I show that Agnosticism of any kind has a language bias that can not use excluded middle and the Sesh axiom at the same time. It is fine to use either, but not both.

In the debate about Theism vs Atheism vs Agnosticism, I came across a discussion where a person used sets to clarify the positions why Atheism is distinct from Agnosticism[1]:

- Theist: Anyone who believes it is more probable than not that a deity exists.
- Atheist: Anyone who is not in the set “theist”.
- Gnostic: Anyone who makes the positive claim that a deity does or does not exist.
- Agnostic: Anyone who is not in the set “gnostic”.

The problem is that Agnostics might agree with the proposition that a deity either exists or does not exists, given that the deity is well defined. The argument is clear for a debate, but formally I think it is ambiguous and upon exploring what this meant in logic, I came up with the following proof:

∵	!~(Theism^true) & !~(false^Theism)	
∴	~!(Theism^true) & !~(false^Theism)	using Sesh
∴	~(false^Theism) & !~(false^Theism)	using excluded middle
∴	false	

The result is that Agnosticism has a language bias built into it that can not allow use of excluded middle and the Sesh axiom at the same time.

The excluded middle is an axiom, which in HOOO EP[2] is written as following:

    (a | !a)^true   for all `a`

The Sesh axiom in HOOO EP (for more information, see “Overview of Path Semantical Logics”[3]):

    (!~a == ~!a)^true   for all `a`

Most logicians know that when excluded middle is added to constructive logic, one gets classical logic. However, the Sesh axiom is much more mysterious. Here, the easiest way to think of it is that if you have some proposition `a`, there is a proposition `~a` that hold all information about some set of people who believe `a`. We do not know in detail what the proposition `~a` implies, but neither do we care as long we can prove the theorems we need to make progress.

The following is a tautology in HOOO EP: (a | !a)^true → (!(a^true) == false^a)^true   for all `a`
For proof, see Lemma 1 in Appendix A. It means, under excluded middle one can get tautological equality, which is needed for substituting the argument of the path semantical qubit operator `~`. Therefore, `~!(a^true)` becomes `~false^a`.

The Sesh axiom tells us that when we negate the proposition `~a` to `!~a`, which might be thought of as inverting the set of people who believe `a`, this set is equal to a second set of people. This second set of people are those who believe `!a`, described as `~!a`. In standard Path Semantics, the Sesh axiom is assumed, so this means that Agnosticism in standard Path Semantics is constructive.

References:

[1] “Comment on clarified definitions of Theism vs Atheism vs Agnosticism”
https://old.reddit.com/r/DebateAnAtheist/comments/1fdnsd4/new_atheist_epistemology/lmhkb3l/

[2] “Hooo”
AdvancedResearch – Propositional logic with exponentials
https://github.com/advancedresearch/hooo

[3] “Overview of Path Semantical Logics”
Sven Nilsen, 2024
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip2/overview-of-path-semantical-logics.pdf

Appendix A:

```
fn lemma_1 : (a | !a)^true -> (!(a^true) == false^a)^true {
    use std::pow_transitivity;
    use std::tauto_hooo_or;

    x : (a | !a)^true;

    fn f1 : a^true | (!a)^true -> !(a^true) == false^a {
        use std::imply_lift;
        use std::refl;
        use std::tauto_not_to_para;
        use std::triv;

        x : a^true | (!a)^true;

        lam f : !(a^true) => false^a {

            y : !(a^true);

            lam f : a^true => false^a {
                z : a^true;
                let z2 = y(z) : false;
                let r = match z2 : false^a;
                return r;
            }
            lam g : (!a)^true => false^a {
                z : (!a)^true;
                let r = tauto_not_to_para(z) : false^a;
                return r;
            }
            let r = match x (f, g) : false^a;
            return r;
        }
        lam g : false^a => !(a^true) {
            y : false^a;

            lam f : a^true => !(a^true) {
                z : a^true;
                let z2 = triv(z) : a;
                let z3 = y(z2) : false;
                let r = match z3 : !(a^true);
                return r;
            }
            lam g : !(a^true) {
                z : a^true;
                let z2 = triv(z) : a;
                let r = y(z2) : false;
                return r;
            }
            let g2 = imply_lift(g) : (!a)^true => !(a^true);
            let r = match x (f, g2) : !(a^true);
            return r;
        }
        let r = refl(f, g) : !(a^true) == false^a;
        return r;
    }

    let x2 = tauto_hooo_or(x) : (a^true | (!a)^true)^true;
    let r = pow_transitivity(x2, f1) : (!(a^true) == false^a)^true;
    return r;
}
```

```
fn lemma_2 : (a | !a)^true -> (a^true == a)^true {
    use std::pow_transitivity;
    use std::tauto_hooo_or;

    x : (a | !a)^true;

    fn f : (a^true | (!a)^true) -> (a^true == a) {
        use std::imply_lift;
        use std::refl;
        use std::triv;

        y : a^true | (!a)^true;

        lam f : a^true => (a^true == a) {
            z : a^true;
            let z2 = triv(z) : a;
            let f = imply_lift(z2) : a^true => a;
            let g = imply_lift(z) : a => a^true;
            let r = refl(f, g) : a^true == a;
            return r;
        }
        lam g : (!a)^true => (a^true == a) {
            z : (!a)^true;
            lam f : a^true => a {
                z2 : a^true;
                let r = triv(z2) : a;
                return r;
            }
            lam g : a => a^true {
                z2 : a;
                let z3 = triv(z) : !a;
                let z4 = z3(z2) : false;
                let r = match z4 : a^true;
                return r;
            }
            let r = refl(f, g) : a^true == a;
            return r;
        }
        let r = match y (f, g) : a^true == a;
        return r;
    }

    let x2 = tauto_hooo_or(x) : (a^true | (!a)^true)^true;
    let r = pow_transitivity(x2, f) : (a^true == a)^true;
    return r;
}
```

Appendix B

Under the excluded middle, there are only Theists and Atheists.
The Gnostic or Agnostic bias goes away.
Gnostic Theist becomes equal to Agnostic Theist.
Gnostic Atheist becomes equal to Agnostic Atheist.
However, this does not imply that Agnosticism is absurd.

One can prove this using the following theorem (for proof, see Lemma 2 in Appendix A):

    (a | !a)^true -> (a^true == a)^true   for all `a`

Gnosticism   ~(Theism^true) | ~(false^Theism)

Agnosticism   !~(Theism^true) & !~(false^Theism)

    This is the same as `!(~(Theism^true) | ~(false^Theism))`.

    Under excluded middle and the Sesh axiom, Agnosticism becomes absurd.

Gnostic Theist   ~(Theism^true)

Gnostic Atheist   ~(false^Theism)

    This is the same as `~((!Theism)^true)`.

    One can prove that under excluded middle and the Sesh axiom that
    all people who are not theists are atheists. If either axiom holds but not both, or neither,
    then one can no longer prove that people who are not theists are atheists.
    Since the Sesh axiom is assumed by default in standard Path Semantics,
    this means that Atheism is constructive in standard Path Semantics.

Agnostic Theist   ~Theism

Agnostic Atheist   ~!Theism

Theist   ~Theism | ~(Theism^true)

    The set of theists is the union of the set of agnostic theists and gnostic theists.

Atheist   ~!Theism | ~(false^Theism)

    The set of atheists is the union of the set of agnostic atheists and gnostic atheists.

In principle, a person can be both agnostic theist and gnostic theist.
These two sets can overlap and the intersection can be assigned some additional meaning,
such as a person holding some agnostic theist beliefs and some gnostic theist beliefs.
The same principle holds for atheists.