Introduction to classic & constructive

Path **S**emantical **Q**uantum Logic

PSQ

By Sven Nilsen, 2022

Quick Summary of Logic 1 of 2

- 16 binary operators (AND, OR, IMPLY etc.)
- Propositional Logic is the 0-th order logic
- 1-th order logic adds predicates and quantifiers
- Classical Logic has excluded middle `a | !a`
- Constructive Logic has no excluded middle
- Exponential Complexity `O(2^n)` for `n` arguments

Quick Summary of Logic 2 of 2

- A classical proof `f` is
  `f : bool^n → bool` where `f <=> \true`

  This means `f` returns `true` for all inputs
- A constructive proof `f` is a program
  `f : A` where `A` is a type
  This means `f` is a point in the space `A`

Extending Logic for PSQ

- Three extensions necessary:
  - 1^{st} step: Exponential `^` operator
  - 2^{nd} step: Qubit `~` operator
  - 3^{rd} step: Path semantical levels using core axiom
- Related families of logics (excm = excluded middle):
  - **Constructive PSQ: 1, 2, 3** PSI: 2, 3
  - **Classical PSQ: 2 + excm** PSL: 3 + excm

Exponential `^` Operator 1 of 3

- `a^b` means `a` is provable from `b`
- Similar to `b => a`, but can not capture from the environment like lambda/closure expressions
- Can be thought of as like a function pointer
- `uniform(a) = (a^true | false^a)`
- `theory(a) = !uniform(a)`

Exponential `^` Operator 2 of 3

- `a^b => (a^b)^c`
- `(a => b^a) => b^a`
- `(a^b)^c => a^(b & c)`
- `(a^b)^a`
- `uniform(a) | theory(a)`

Exponential `^` Operator 3 of 3

- `(false^(a == b) | false^(b == c)) => (a == c)^true`
- `(a □ b)^c == (a^c □ b^c)`
- `c^(a □ b) == (c^a □[¬] c^b)`
- `((!a)^b)^(!(a^b))`
- The `□` symbol stands for any binary operator
- The expression `□[¬]` means dual operator

The Qubit `~` Operator 1 of 2

- In classic logic, `~a` uses `a` as a pseudo-random seed such that `~a` can in principle be equal to any other proposition – by some infinitesimal chance
- This makes classical proofs probabilistic
- The number of nested applications of the qubit operator +1 defines homotopy levels, e.g. `~~a` has homotopy level 3

The Qubit `~` Operator 2 of 2

- In constructive logic, `~a` is a 1-avatar (new-type) which only allows substitution `~b` under tautological equality `(a == b)^true`
- It means, one can not turn `~a` into `~b` except under special circumstances where is it provable that `a == b` under none assumptions
- Requires Exponential Propositions `^`

The Quality `~~` Operator

- `(a ~~ b) == ((a == b) & ~a & ~b)`

  This proves `(a ~~ a) == ~a`
- `(a == b) & theory(a == b) => (a ~~ b)`
  This lifts equality into quality,
  which does not hold for reflexivity `a == a`
- Quality `~~` is used in the core axiom

The Aquality `~!~` Operator

- `(a ~!~ b) == ((a == b) & !~a & !~b)`

  This proves `(a ~!~ a) == !~a`
- Usually accompanied with axiom `!~a == ~!a`, but does not hold in all models, e.g. Dit Calculus
- In principle the same as quality, since one can prove same theorems by swapping quality with aquality and vice versa – yet usually interpreted differently

Path Semantical Levels 1 of 2

- Each proposition has an associated natural number which is used to prove an order `a1 < a2`
- Core axiom:

```
((a1 => a2) & (b1 => b2) &
 (a1 < a2) & (b1 < b2) &
 (a1 ~~ b1)) => (b1 ~~ b2)
```

Path Semantical Levels 2 of 2

- Levels can be interpreted as moments in time, or higher universes of types
- Choice of core axioms forces bias toward quality, aquality, or even restoring symmetry of the two, which has deep philosophical implications
- Each level is like a complete language of logic, where quality is used to lift relations from one level to the next – using the core axiom (this structures levels)

Homotopy Levels 1 of 2

- Homotopy Levels are not Path Semantical Levels, but more like “unstructured relativity of time”
- `|bool^n x bool^n → bool^n|` counts number of binary functions as measure of complexity
- Rapidly grows `1, 16, 2^32, ...` larger than the number of atoms in the universe in level 4
- Humanly incomprehensibly complex and rich semantics

Homotopy Levels 2 of 2

- `hom_eq(2, a, b) == ((a == b) & (~a == ~b))`
- `(a ~~ b) => hom_eq(2, a, b)`
- `(a ~!~ b) => hom_eq(2, a, b)`
- `hom_eq(2, a, b) == (a ~~ b) | (a ~!~ b)` (excm)
- `hom_eq(n, a, b)` aligns qubits in range `[0, n)`, such that `∀ i [0, n) { ~^i(a) == ~^i(b) }`

Hypertorus Homotopy 1 of 2

- \`~a\` might be thought of as creating a “circle” around the point \`a\`
- \`~~a\` might be thought of as creating a “torus” around the point \`a\`, or “circle” around “circle” around \`a\`
- \`~^n(a)\` might be thought of as creating a “hypertorus” around the point \`a\`

Hypertorus Homotopy 2 of 2

- When `~a == ~b`, the two circles around `a` and `b` are propositionally equal, which forms a homotopy path
- This notion of homotopy path is restricted to at most one homotopy path between two points
- The points can be inequal, e.g. `a != b`
  while being connected by homotopy `~a == ~b`

Bonus: Modal Logic 1 of 2

- Uses Exponential `^` operator
- Based on theory of Avatar Extensions
- `◇p == false^(false^p))` is “taboo knowledge”
- Safe to “write”, unsafe to “read”
- Safe invariants using unsafe proofs hide the taboo knowledge from the outside world

Bonus: Modal Logic 2 of 2

- The \`!\` operator using \`^\` inverts every bit
- The \`!\` operator using \`◇\` normalises to the invertion in the sense of possible worlds
- Allows avoiding using two different \`!\` operators
- Might be thought of as a local space of a Möbius strip (or a space with Listing-Möbius shifts)

Summary

- Extend logic with `^, ~, <, ◇`
- Follows from making core axiom “well behaved”

One Axiom to rule them all,
One Axiom to find them,
One Axiom to bring them all
and in the End, a new beginning
there and back again,
so comes snow after fire,
and even dragons have their endings