Last Order Logic

by Sven Nilsen, 2021

In this paper I present a logical language which is kind of like First Order Logic, but funny.

Last Order Logic (LOL) is a language given by the following syntax:

1	term for truth
0	term for falsehood
I	unit interval type
(a, b)	tuple of \`a\` and \`b\`
¬a	logical NOT
a ∧ b	logical AND
a ∨ b	logical OR
a ⊻ b	logical XOR
a == b	logical EQ
a => b	logical IMPLY
a ~= b	a path from \`a\` to \`b\`
(a ~= b) ~ 0 : a	path start point
(a ~= b) ~ 1 : b	path end point
((a ~= b) ~= (c ~= d)) ~ (1, 0) : c	surface point
f(a)	lambda application
\(i : I) = f(i)	lambda abstraction
∀ i : I { f(i) } : un(T)	where \`f\` has true type \`T\` for all \`i\`
∀ i : I { f(i) } : nu(U)	where \`f\` has false type \`U\` for at least one \`i\`
∃ i : I { f(i) } : nu(T)	where \`f\` has true type \`T\` for at least one \`i\`
∃ i : I { f(i) } : un(U)	where \`f\` has false type \`U\` for all \`i\`
f(a ~= b) == (f(a) ~= f(b))	lambda application for paths
lift(a) : a	\`a\` is lifted to type level
(0 ~= 1) : (I ~= I)	the path \`0 ~= 1\` has path type \`I ~= I\`

Notice that path points and quantifiers \`∀\` and \`∃\` do not evaluate, they produce a type instead. The quantifiers \`∀\` and \`∃\` are homogenous, which means when it has e.g. a true type, it has the same type for every true case. The reason for this design choice is to introduce multiple senses of truth values. In First Order Logic[1], there is only one sense of truth which is \`false\` and \`true\`. In Last Order Logic, one can have e.g. \`un(1)\` which means "uniformly true". This can be nested, e.g. \`nu(un(1))\` which means "non-uniformly uniformly true". Furthermore, a path can be a truth value, e.g. \`un(a ~= a)\` means "uniformly \`a\` in one dimension".

un	**un**iform	alternatives: objective, "clothes on"
nu	**n**on-**u**niform	alternatives: personal, "nude"

The terminology "uniform" and "non-uniform" comes from Avatar Extensions[2].
These senses of truth are "truthful" when every symbol is concrete, such as \`un(1)\` or \`nu(1 ~= 0)\`.
When the symbols are not concrete, the uniformity or non-uniformity is not to be taken seriously.
The intuition is that a uniform truth can turn into non-uniform and vice versa.
This idea borrows from the mathematical universe called "The Joker"[3].

This paper only presents the language of Last Order Logic and does not include inference rules.
Rest of this paper are examples.

Example 1:

∵ f(i : I) = (1 ~= 0) ~ i
∴ f(0) : 1 f(1) : 0 ∃ i { f(i) } : nu(1)

Example 2:

∵ f(i : I) = ((1 ~= 0) ~= (1 ~= 0)) ~ i
∴ f(0) : (1 ~= 0) f(1) : (1 ~= 0) ∀ i : I { f(i) } : un(1 ~= 0)

Example 3:

∵ p ~ 0 : 1 p ~ 1 : 0
∴ p == (1 ~= 0)

Example 4:

∵ ∃ i : I { p ~ i } : nu(1) p ~ 0 : 0
∴ p == (1 ~= 0)

Example 5:

∵ ∃ i : I { p ~ i } : nu(1 ~= 0) p ~ 0 : (0 ~= 0)
∴ p == ((0 ~= 0) ~= (1 ~= 0))

Example 6:

∵ ∀ i : I, j : I { p ~ (i, j) } : un(un(1))
∴ p = ((1 ~= 1) ~= (1 ~= 1))

Example 7:

∵ ∀ i : I { p ~ i } : un(1)
∴ ∀ i : I { ¬p ~ i } : un(0)

Example 8:

∵ ∀ i : I { p ~ i } : un(1)
∴ ¬∃ i : I { ¬p ~ i } : un(1)

Example 9:

∵ ∀ i : I { p ~ i } : un(1)
∴ ∃ i : I { ¬p ~ i } : un(0)

Example 10:

∵ ∃ i : I { p ~ i } : nu(1)
∴ ¬∃ i : I { p ~ i } : nu(0)

Example 11:

∵ x : un(0)
∴ ¬x : un(1)

Example 12:

∵ x : nu(0)
∴ ¬x : nu(1)

Example 13:

∵ ¬∀ i : I { p ~ i } : nu(1)
∴ ∃ i : I { ¬p ~ i } : nu(1)

Example 14:

∵ ∃ i : I, j : I { p ~ (i, j) } : un(false)
∴ ∃ i : I { ¬p ~ i } : nu(true)

References:

[1] "First Order Logic"
Wikipedia
https://en.wikipedia.org/wiki/First-order_logic

[2] "Avatar Extensions"
AdvancedResearch – Summary page on Avatar Extensions
https://advancedresearch.github.io/avatar-extensions/summary.html

[3] "The Joker"
Sven Nilsen, 2021
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip2/the-joker.pdf