Avatar Logic to Set Theory

by Sven Nilsen, 2021

In this paper I introduce a method of translating Avatar Logic to Zermelo-Fraenkel Set Theory.

The axioms of Avatar Logic[1] might be translated to Zermelo-Fraenkel Set Theory[2]:

(a, b) ∧ b : p ∧ uniq(b)
(p, (a, b)) ∧ ∃! z { (p, (a, z)) } ∧ ∃! r { (r, (a, b)) }

(a, q'(b)) ∧ q'(b) : p
(p, (a, (q, b))) ∧ ∀ x { ∃! z { (p, (a, (z, x))) } } ∧ ∃! r { (r, (a, (q, b))) }

Instead of predicates e.g. `p(a, b)`, a pair used instead `(p, (a, b))` to avoid Second-Order Logic[3].

Translation must happen for every relation, otherwise it would require extending Second-Order Logic[3] with tuples, roles and 1-avatars. Per relation requires only First-Order Logic[4].

The translation uses Kuratowski's definiton[5] of an ordered pair `{{x}, {x, y}}` for `x'(y)`.
This representation is chosen because ordered pairs are not used as arguments in Avatar Logic.

Ordered pairs might also be used without `b : p`, but only to mean `(a, b)` as a binary relation.

The `uniq` predicate returns `true` for all atomic symbols, plus those 1-avatars that are optionally chosen to be behaving uniquely. Both axioms must be applied when the 1-avatar is unique.

An expanded version is provided on the next page.

In expanded form limited to quantifiers `∀, ∃`, connectives `=>, =, ∈, ∨, ∧`, negation `¬`:

(a, b) ∧ b : p ∧ uniq(b)

```
∃ x1 { x1 ∈ _k1 => ∀ x2 { x2 ∈ _k1 => x2 = a ∨ x2 = x1 } ∧ ∀ x3 { x3 ∈ x1 => x3 = a ∨ x3 = b } } ∧
∃ x4 { x4 ∈ _k2 => ∀ x5 { x5 ∈ _k2 => x5 = p ∨ x5 = x4 } ∧ ∀ x6 { x6 ∈ x4 => x6 = p ∨ x6 = _k1 } } ∧
_k2 ∧
∃ x9 {
 ∃ x10 { x10 ∈ _k3 => ∀ x11 { x11 ∈ _k3 => x11 = a ∨ x11 = x10 } ∧ ∀ x12 { x12 ∈ x10 => x12 = a ∨ x12 = x9 } } ∧
 ∃ x13 { x13 ∈ _k4 => ∀ x14 { x14 ∈ _k4 => x14 = p ∨ x14 = x13 } ∧ ∀ x15 { x15 ∈ x13 => x15 = p ∨ x15 = _k3 } } ∧
 _k4 ∧ ¬∃ x16 {
  ∃ x17 { x17 ∈ _k5 => ∀ x18 { x18 ∈ _k5 => x18 = a ∨ x18 = x17 } ∧ ∀ x19 { x19 ∈ x17 => x19 = a ∨ x19 = x16 } } ∧
  ∃ x20 { x20 ∈ _k6 => ∀ x21 { x21 ∈ _k6 => x21 = p ∨ x21 = x20 } ∧ ∀ x22 { x22 ∈ x20 => x22 = p ∨ x22 = _k5 } } ∧
  _k6 ∧ ¬x9 = x16
 }
} ∧
∃ x25 {
 ∃ x26 { x26 ∈ _k7 => ∀ x27 { x27 ∈ _k7 => x27 = a ∨ x27 = x26 } ∧ ∀ x28 { x28 ∈ x26 => x28 = a ∨ x28 = b } } ∧
 ∃ x29 { x29 ∈ _k8 => ∀ x30 { x30 ∈ _k8 => x30 = x25 ∨ x30 = x29 } ∧ ∀ x31 { x31 ∈ x29 => x31 = x25 ∨ x31 = _k7 } } ∧
 _k8 ∧ ¬∃ x32 {
  ∃ x33 { x33 ∈ _k9 => ∀ x34 { x34 ∈ _k9 => x34 = a ∨ x34 = x33 } ∧ ∀ x35 { x35 ∈ x33 => x35 = a ∨ x35 = b } } ∧
  ∃ x36 { x36 ∈ _k10 => ∀ x37 { x37 ∈ _k10 => x37 = x32 ∨ x37 = x36 } ∧ ∀ x38 { x38 ∈ x36 => x38 = x32 ∨ x38 = _k9 } } ∧
  _k10 ∧ ¬x25 = x32
 }
}
```

(a, q'(b)) ∧ q'(b) : p

```
_k3 ∧
∀ x10 {
 ∃ x13 {
  ∃ x14 { x14 ∈ _k4 => ∀ x15 { x15 ∈ _k4 => x15 = x13 ∨ x15 = x14 } ∧ ∀ x16 { x16 ∈ x14 => x16 = x13 ∨ x16 = x10 } } ∧
  ∃ x17 { x17 ∈ _k5 => ∀ x18 { x18 ∈ _k5 => x18 = a ∨ x18 = x17 } ∧ ∀ x19 { x19 ∈ x17 => x19 = a ∨ x19 = _k4 } } ∧
  ∃ x20 { x20 ∈ _k6 => ∀ x21 { x21 ∈ _k6 => x21 = p ∨ x21 = x20 } ∧ ∀ x22 { x22 ∈ x20 => x22 = p ∨ x22 = _k5 } } ∧
  _k6 ∧ ¬∃ x23 {
   ∃ x24 { x24 ∈ _k7 => ∀ x25 { x25 ∈ _k7 => x25 = x23 ∨ x25 = x24 } ∧ ∀ x26 { x26 ∈ x24 => x26 = x23 ∨ x26 = x10 } } ∧
   ∃ x27 { x27 ∈ _k8 => ∀ x28 { x28 ∈ _k8 => x28 = a ∨ x28 = x27 } ∧ ∀ x29 { x29 ∈ x27 => x29 = a ∨ x29 = _k7 } } ∧
   ∃ x30 { x30 ∈ _k9 => ∀ x31 { x31 ∈ _k9 => x31 = p ∨ x31 = x30 } ∧ ∀ x32 { x32 ∈ x30 => x32 = p ∨ x32 = _k8 } } ∧
   _k9 ∧ ¬x13 = x23
  }
 }
} ∧
∃ x35 {
 ∃ x36 { x36 ∈ _k10 => ∀ x37 { x37 ∈ _k10 => x37 = q ∨ x37 = x36 } ∧ ∀ x38 { x38 ∈ x36 => x38 = q ∨ x38 = b } } ∧
 ∃ x39 { x39 ∈ _k11 => ∀ x40 { x40 ∈ _k11 => x40 = a ∨ x40 = x39 } ∧ ∀ x41 { x41 ∈ x39 => x41 = a ∨ x41 = _k10 } } ∧
 ∃ x42 { x42 ∈ _k12 => ∀ x43 { x43 ∈ _k12 => x43 = x35 ∨ x43 = x42 } ∧ ∀ x44 { x44 ∈ x42 => x44 = x35 ∨ x44 = _k11 } } ∧
 _k12 ∧ ¬∃ x45 {
  ∃ x46 { x46 ∈ _k13 => ∀ x47 { x47 ∈ _k13 => x47 = q ∨ x47 = x46 } ∧ ∀ x48 { x48 ∈ x46 => x48 = q ∨ x48 = b } } ∧
  ∃ x49 { x49 ∈ _k14 => ∀ x50 { x50 ∈ _k14 => x50 = a ∨ x50 = x49 } ∧ ∀ x51 { x51 ∈ x49 => x51 = a ∨ x51 = _k13 } } ∧
  ∃ x52 { x52 ∈ _k15 => ∀ x53 { x53 ∈ _k15 => x53 = x45 ∨ x53 = x52 } ∧ ∀ x54 { x54 ∈ x52 => x54 = x45 ∨ x54 = _k14 } } ∧
  _k15 ∧ ¬x35 = x45
 }
} ∧
∃ x1 { x1 ∈ _k1 => ∀ x2 { x2 ∈ _k1 => x2 = q ∨ x2 = x1 } ∧ ∀ x3 { x3 ∈ x1 => x3 = q ∨ x3 = b } } ∧
∃ x4 { x4 ∈ _k2 => ∀ x5 { x5 ∈ _k2 => x5 = a ∨ x5 = x4 } ∧ ∀ x6 { x6 ∈ x4 => x6 = a ∨ x6 = _k1 } } ∧
∃ x7 { x7 ∈ _k3 => ∀ x8 { x8 ∈ _k3 => x8 = p ∨ x8 = x7 } ∧ ∀ x9 { x9 ∈ x7 => x9 = p ∨ x9 = _k2 } }
```

Notice that `p` might be written with uppercase letter in standard First-Order Logic notation.
The variables starting with underscore e.g. `_n`, are introduced to bind the sub-expressions together.

References:

[1] “Avatar Logic”
AdvancedResearch – Summary Page on Avatar Extensions
https://advancedresearch.github.io/avatar-extensions/summary.html#avatar-logic

[2] “Zermelo-Fraenkel set theory”
Wikipedia
https://en.wikipedia.org/wiki/Zermelo%E2%80%93Fraenkel_set_theory

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

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

[5] “Kuratowski’s definition – Ordered pair”
Wikipedia
https://en.wikipedia.org/wiki/Ordered_pair#Kuratowski's_definition