Proof of Exclusiveness

by Sven Nilsen, 2017

In general, the sub-types `[$g_0$] $a_0$` and `[$g_1$] $a_1$` are exclusive when:

$$f : ([g_0]\ a_0 \rightarrow [h]\ b_0) \wedge ([g_1]\ a_1 \rightarrow [h]\ b_1)$$
$$b_0 \neg= b_1 \Rightarrow |[g_0]\ a_0 \wedge [g_1]\ a_1| = 0$$

In logic, two propositions are exclusive when one being true implies the other is false.

$$(a \rightarrow \neg b) \wedge (b \rightarrow \neg a) = \neg(a \wedge b) = \text{nand}(a, b)$$

An exclusive relationship can be XOR or NAND, which itself is NAND:

$$\text{xor}(a, b) \vee \text{nand}(a, b) = \text{nand}(a, b)$$

Example: Either an operator is unary (taking one argument) or it is binary (taking two arguments). If it is neither unary or binary, it might take zero arguments or more than two. Assume that all operators and constants are described with a single data structure `expression`:

arguments : expression → [expression]
arguments : [is_unary] true → [len] 1
arguments : [is_binary] true → [len] 2

This is a proof that `is_unary` and `is_binary` are exclusive. Unary operator has list of length 1, and binary operator has list of length 2, so no expression can be both unary and binary operator.

|[is_unary] true ∧ [is_binary] true| = 0

Notice that this proof does not require checking `is_unary` and `is_binary` for all inputs. It is extremely computationally efficient because it only requires comparing two values.

Another example: In the board game Othello, one player plays with white bricks and the other player players with black bricks. Using shorthand notation:

color_of : brick → color
color_of : player(1) → (= white)
color_of : player(2) → (= black)
player : nat × brick → bool

Notice that `player(1)` is the same as `[player(1)] true` and `(= x)` is the same as `[id] x`. This proves that the sub-types `player(1)` and `player(2)` are exclusive. Since there are only 2 players, not only does player 1 have no bricks in common with player 2, but the two sets are invertible of each other:

|player(1) ∧ player(2)| = 0	Player 1 has no bricks in common with player 2
player(1) ¬= player(2)	The two sub-types are invertible sets of each other