Transitive Existential Paths

by Sven Nilsen, 2017

The concept of a transitive existential path `$\exists(f \to g)$` formalizes what it means to have "information" and "symmetry" stored in some space and performing information-preserving transformations. It does so without referring to the specific data structure used to model the information. In addition, a modular transitive existential path `$\exists(f \% g)$` gives solutions of infinite spaces `$|\exists(f \% g)| == 1$` and groups `$|\exists(f \% g)| == \infty$`.

A transitive existential path of `f` by `g` with inverse `$g^{-1}$` is defined as the following:

$$\exists(f \to g) := \backslash(x : [\exists g] \text{ true}) = (\exists f)(g^{-1}(x))$$

$$g : A \to A$$
$$g^{-1} \Leftrightarrow id[id \to g]$$

The transitive existential path of `f` by `id` is equal to the existential path:

$$\exists(f \to id) \Leftrightarrow \exists f$$

Repeated transformations is written:

$$\exists(f \to g^n) := \backslash(x : [\exists g^n] \text{ true}) = (\exists f)(g^{-n}(x))$$

$$n : nat$$

It is easy to see that the transitive existential path can be defined in terms of any "previously defined" transitive existential path:

$$\exists(f \to g^{n+m}) := \backslash(x : [\exists g^{n+m}] \text{ true}) = (\exists(f \to g^n))(g^{-m}(x))$$

$$n : nat$$
$$m : nat$$
$$g^0 \Leftrightarrow id$$

Some examples:

$$\exists(f \to g^1) := \backslash(x : [\exists g^1] \text{ true}) = (\exists f)(g^{-1}(x))$$
$$\exists(f \to g^2) := \backslash(x) = (\exists(f \to g^1))(g^{-1}(x))$$
$$\exists(f \to g^3) := \backslash(x) = (\exists(f \to g^2))(g^{-1}(x))$$
$$\exists(f \to g^3) := \backslash(x) = (\exists(f \to g^1))(g^{-2}(x))$$

As a closely related concept, a modular transitive existential path is a function returning `true` for `n` reducing to the existential path.

$$\exists(f \% g) : nat \rightarrow bool$$

The function captures a characteristic variable `n` such that the following condition is satisfied:

$$\forall\ x : nat \{ \exists(f \rightarrow g^{x \cdot n}) <=> \exists f \}$$

Notice that any natural number `m` multiplied with `n` is also a characteristic variable, so this forms a subset relationship between the two series of modular transitive existential paths:

$$\exists(f \% g^{n \cdot m}) \subseteq \exists(f \% g^{n})$$

For infinite spaces, a modular transitive existential path is defined as:

$$\exists(f \% g) := \backslash(x : nat) = x == 0$$
$$\exists(f \% g) <=> (= 0)$$

$$|\exists(f \% g)| == 1$$

For groups, a modular transitive existential path is defined as:

$$\exists(f \% g) := \backslash(x : nat) = (x \% n) == 0$$

$$|\exists(f \% g)| == \infty$$

A group is naturally unbounded, so you can change `nat` to `int`, if you want to.

This can also be done for infinite spaces, if there exists an inverse for all input. Remember that a transitive existential path got a domain constraint on the argument:

$$\exists(f \rightarrow g) := \backslash(x : [\exists g]\ true) = (\exists f)(g^{-1}(x))$$

The domain constraint `[∃g] true` prevents `x` from taking on values that `$g^{-1}$` does not map to. If you wonder whether `$g^{-1}$` is a partial function, you are both right and wrong, because `$g^{-1}$ <=> id[id → g]` defines a logical equivalence between the path sets and supports partial functions of `$g^{-1}$` when this is the case.

For example, for real numbers a function `g` might return all values:

$$\exists g := (\_ : real) = true$$

This means you can drop the constraint:

$$\exists(f \rightarrow g) := \backslash(x : real) = (\exists f)(g^{-1}(x))$$

A simple example of an infinite space is the `add(k)` function:

add(k) : nat → nat
add := \(k : nat ∧ (> 0)) = \(x : nat) = x + k

Let us say `f` is `even_sequence` and `g` is `add(k)`, it is just a matter of substitute and solve:

∃(f → g) := \(x : [∃g] true) = (∃f)(g^{-1}(x))
∃(even_sequence → add(k)) := \(x : [∃add(k)] true) = (∃even_sequence)($add(k)^{-1}$(x))
  f <=> even_sequence
  g <=> add(k)
∃(even_sequence → add(k)) := \(x : [∃add(k)] true) = (∃even_sequence)(x – k)
  $add(k)^{-1}$(x) = x – k
∃(even_sequence → add(k)) := \(x : [∃add(k)] true) = even(x – k)
  ∃even_sequence <=> even

The modular transitive existential path has only one solution, because if you move even numbers by adding with `k > 0`, you lose `0` from the transitive existential path.

(∃(even_sequence → add(k : (> 0))))(0) = false
(∃even_sequence)(0) <=> even(0) <=> true

∃(even_sequence % add(k)) := \(x : nat) = x == 0

|∃(even_sequence % add(k))| == 1

A simple example of a group is one that increments numbers and maps back using modulus:

g := \(x : nat) = (x + 1) % 8

Think of this as a rotating pattern where `f := [1, 3, 4, 7]`:

	■		■	■			■
■		■		■	■		
	■		■		■	■	
		■		■		■	■
■			■		■		■
■	■			■		■	
	■	■			■		■
■		■	■			■	
	■		■	■			■

After 8 rounds, we end up with the same pattern as we started with. This is because `$g^8$ <=> g`. So, why not use `g`? Why do we have to worry about `f`? The reason is that some patterns repeats themselves more frequently, so to express this precisely we need to write `∃(f % g)`.