Current Object Constrain Notation

by Sven Nilsen, 2017

In this paper I develop a path semantical notation for constraining current objects.

Using syntax from the Dyon programming language, one can write the following:

```
move_x_to_y(x : nat) ~ y : nat = if x < y { x + 1 } else { x }
```

This function uses something called a “current object”, written using a `~` symbol after the arguments. A current object is kind of like a global that can be initialized before calling the function. Current objects run out of scope with the block they are declared inside.

```
fn main() {
	~ y := 20	// initialize `y`
	println(move_x_to_y(10))	// prints 11.
}
```

When the program calling the function `move_x_to_y` initializes current objects deterministically, the program itself will be deterministic. Yet, when looking at `move_x_to_y` alone, it is not possible to determine what output it gives. On its own, it is a non-deterministic function.

Current objects can be constrained by the caller, so one can use `~` in the domain constraint notation:

```
move_x_to_y{~ y: (= 20)} <=> \(x : nat) = if x < 20 { x + 1 } else { x }
```

This function returns no zeroes, because zero is less than `20`. The number `20` is returned twice, since `19 => 20` and `20 => 20`.

The existential path, which tells what the function returns, is “all natural numbers greater than 0”:

```
∃move_x_to_y{~ y: (= 20)} <=> (> 0)
```

The probabilistic existential path, which tells the probability of some output value, is undefined because there are infinite natural numbers. Constraining input to less than `100`:

```
∃ₚmove_x_to_y{(< 100) ~ y: (= 20)} <=> \(x : nat) =
	if x == 20 { 2/99 }
	else if (x < 100) && (x > 0) { 1/99 }
	else { 0 }
```

So far, this worked because the current object constraint `(= 20)` determines the function. It would also work in cases where the existential path is unique, for example the constraint is narrow enough that no input can give different outputs. However, if the existential path is not unique, then there is no deterministic interpretation.