Universal Existential Path is The Opposite Concept of Function Composition that Constructs Ordinary Conditions

by Sven Nilsen, 2017

Universal existential paths are used as oracles for reasoning about a particular class of hypothetical perfect problem solvers. Using a new concept, based on the inverse oracle, defines the sub-type of assumptions that holds for any proof criteria, not just for logic, but for functions in general used for proofs, e.g. confidence interval in probabilistic logic. The inverse oracle exists in the simple form of function composition, which is likely the simplest idea in computing and the extreme opposite in complexity to universal existential paths themselves.

The major result of this paper is that there are two definitions of inverse universal existential paths:

$\exists^{-1}f\{\} := \backslash(h : B \rightarrow \text{bool}) = h \;.\; f$
$\exists^{-1}f\{\} := \backslash(h : B \rightarrow \text{bool}) = \text{argmax } g : [\exists f\{\}]\; h\ \{\ |g|\ \}$

$f : A \rightarrow B$

If you have some proof \`f\` in an arbitrary computational logic and a proof criteria \`h\` then the solutions \`x\` are defined by the sub-type:

$x : [h \;.\; f]\ \text{true}$

Which is equivalent to:

$x : [f]\ [h]\ \text{true}$

This is pretty basic in path semantics and used everywhere, but previously it was not understood *why*. The rest of this paper shows why this true and meaningful.

In propositional logic, it is well known that a proof is a tautology, meaning the proof is true for all \`false/true\` combinations of the arguments. It is worth noticing that the human intuition when reading a proof is taking the conclusion to be true when the assumptions are true. This is because assumptions and conclusion is connected by material implication \`→\`.

$((P \rightarrow Q) \wedge P) \rightarrow Q$ proof of material implication using material implication

One can replace material implication with other functions to prove other things, such as equality.

$\neg(P \land Q) = \neg P \lor \neg Q$ proof of one of DeMorgan's law using equality

The proofs are checked simply by evaluating the expression for all values:

$\forall$ P, Q : bool { $((P \to Q) \land P) \to Q$ }
$\forall$ P, Q : bool { $\neg(P \land Q) = \neg P \lor \neg Q$ }

It is called a tautology because it returns `true` for all values.

DeMorgan's laws corresponds to two symmetric path in path semantics:

and[not] <=> or	$\neg(P \land Q) = \neg P \lor \neg Q$
or[not] <=> and	$\neg(P \lor Q) = \neg P \land \neg Q$

In path semantics, a proof in propositional logic is thought of as a boolean function of `N` arguments.

For example, the proof of material implication is a boolean function of two arguments:

\(P, Q : bool) = $((P \to Q) \land P) \to Q$

By using a function instead of for-all quantifier ($\forall$), one can use it to study partial proofs. A partial proof has some input that makes the boolean function return `false`. With other words, it is not always true, but only true for a sub-type of its input.

In path semantics, a total proof in logic has an existential path logically equivalent to `id`:

$\exists$f <=> id

f : A $\to$ bool
id := \(x : bool) = x

This means that the function returns `true` and does not return `false`.

A total proof in logic is more general than propositional logic, because propositional logic concerns itself only about boolean functions. A boolean function of `N` arguments has the type:

$\text{bool}^N \to \text{bool}$

The rule for a total proof `f` of `N` arguments in propositional logic is:

f <=> true_N

true_N := \(_ : bool^N) = true

In path semantics, one can do theorem proving in other kinds of logic by replacing `$\text{bool}^N$` with any other type. This means the rule above is no longer valid. Instead one uses the following trick:

$$\exists\text{true}_N \Leftrightarrow \text{id}$$

The existential path defines the behavior of the output while ignoring the input type. Therefore, one can use the more general rule:

$$\exists f \Leftrightarrow \text{id}$$

The only difference between a total proof and a partial proof is by adding a domain constraint:

$$\exists f\{g\} \Leftrightarrow \text{id}$$

$$g : A \to \text{bool}$$

A universal existential path is a higher order path semantical function that can take an arbitrary domain constraint and return the existential path:

$$\exists f\{\} : (A \to \text{bool}) \to (B \to \text{bool})$$

$$f : A \to B$$

Universal existential paths are used as oracles to reason about perfect/optimal problem solvers.

For theorem proving, one would like to do find the largest domain constraint such that the existential path is logically equivalent to `id`:

$$g_{max} = \text{argmax } g : [\exists f\{\}] \text{ id } \{ |g| \}$$

This finds a function `$g_{max}$` that tells whether some input configuration to the proof makes it return `true`. The proof is total when all input configurations hold, which is equivalent to:

$$g_{max} \Leftrightarrow \text{true}_A$$

$$\text{true}_A := \backslash(\_ : A) = \text{true}$$

Yet, this is not fully satisfactory. So far, only proofs in logic have been discussed. One would like to go beyond classical logic to other kinds of logic.

For example, in proofs of probabilistic logic it is not very interesting to know something to be 100% true, since this often can be proven in classical logic. Instead one would like to know when something is e.g. 95-100% true.

This is where inverse universal existential paths come in. An inverse universal existential path takes an arbitrary function `h` that defines the criteria for the proof, and returns a function `$g_{max}$` that returns `true` for all input configurations where the criteria holds, and `false` otherwise:

$$\exists^{-1}f\{\} := \backslash(h : B \to \text{bool}) = \text{argmax } g : [\exists f\{\}]\ h\ \{\ |g|\ \}$$

For example, in probabilistic logic, one can get the function `$g_{max}$` that defines the configuration of assumptions such that the proof `f` holds in the confidence interval 95-100%:

$$g_{max} := (\exists^{-1}f\{\})((>= 0.95))$$

$x : [g_{max}]$ true	`x` is some sub-type such that `f` is 95-100% true
$f : X \to$ prob	`f` is some probabilistic proof, a probability distribution over `X`

Notice that the word "assumption" is used about the configuration even the proof itself might not contain material implication. This is because the relationship between input configuration and output criteria gives a natural interpretation that corresponds to assumptions and conclusion.

In logic, if one uses false assumptions, any conclusion follows. Here, for any impossible proof criteria there is no input configuration that satisfies the proof, such that `$g_{max}$` returns `false` for all inputs:

$(\exists^{-1}f\{\})(h) \iff g_{max} \iff \text{false}_A$ The proof criteria is impossible

This means one can remove assumptions from inside the proof to make the inverse universal existential path tell what assumptions are missing. When moving the missing assumption inside the proof, one gets a tautology in logic. False assumptions are avoided by under-constraining the proof, using fewer assumptions when possible.

For example, in the proof of material implication, remove `P` from the assumptions inside the proof:

$$f := \backslash(P, Q : \text{bool}) = (P \to Q) \to Q$$

Next, create a truth table:

P	Q	$P \to Q$	$(P \to Q) \to Q$	
0	0	1	0	
0	1	1	1	$\neg P \land Q$ is true
1	0	0	1	P is true
1	1	1	1	P is true

The proof is true when:

$$(\exists^{-1}f\{\})((= \text{true})) \iff \backslash(P, Q : \text{bool}) = P \lor \neg P \land Q$$

Another example, trying to prove something impossible:

$f := \backslash(P : bool) = P \wedge \neg P$

P	¬P	P ∧ ¬P
0	1	0
1	0	0

$(\exists^{-1}f\{\})((= true)) <=> false_1$

A simple example in probabilistic logic:

$f := \backslash(x : prob) = x$

$(\exists^{-1}f\{\})((>= 0.95)) <=> (>= 0.95)$

Another example:

$f := \backslash(x, y : prob) = (x \rightarrow y) \rightarrow y$

$(x \rightarrow y) \rightarrow y$

$1 - (x \rightarrow y)(1 - y)$

$1 - (1 - x(1 - y))(1 - y)$

$1 - (1 - x + xy)(1 - y)$

$1 - (1 - x - y + 2xy - xy^2)$

$x + y - 2xy + xy^2$

$(\exists^{-1}f\{\})((>= 0.95)) <=> \backslash(x, y : prob) = x + y - 2xy + xy^2 >= 0.95$

Notice how straightforward it is to use probabilistic logic in this case.

Actually, it is just function composition that constructs an ordinary condition:

$(\exists^{-1}f\{\})(h) <=> h . f$

This holds for all proofs, both in classical logic, probabilistic logic and other kinds of logic which truth is grounded in computation.

$\exists^{-1}f\{\} := \backslash(h) = h . f$