Existential Paths of Real Addition on Intervals

by Sven Nilsen, 2017

The existential paths of real addition, constrained by two functions `$g_0$` and `$g_1$` of some real variables, are closed under a constructive higher order family of functions:

$$\exists \mathrm{add}\{g_0(a), g_1(b)\} \Leftrightarrow g_2(a+b)$$

g_0/g_1	**false$_1$**	**(> a)**	**(= a)**	**(>= a)**	**(< a)**	**(¬= a)**	**(<= a)**	**true$_1$**
false$_1$	false$_1$	false$_1$	false$_1$	false$_1$	false$_1$	false$_1$	false$_1$	false$_1$
(> b)	false$_1$	(> a+b)	(> a+b)	(> a+b)	true$_1$	true$_1$	true$_1$	true$_1$
(= b)	false$_1$	(> a+b)	(= a+b)	(>= a+b)	(< a+b)	(¬= a+b)	(<= a+b)	true$_1$
(>= b)	false$_1$	(> a+b)	(>= a+b)	(>= a+b)	true$_1$	true$_1$	true$_1$	true$_1$
(< b)	false$_1$	true$_1$	(< a+b)	true$_1$	(< a+b)	true$_1$	(< a+b)	true$_1$
(¬= b)	false$_1$	true$_1$	(¬= a+b)	true$_1$	true$_1$	true$_1$	true$_1$	true$_1$
(<= b)	false$_1$	true$_1$	(<= a+b)	true$_1$	(< a+b)	true$_1$	(<= a+b)	true$_1$
true$_1$	false$_1$	true$_1$	true$_1$	true$_1$	true$_1$	true$_1$	true$_1$	true$_1$

Each function defines a sub-type of the real numbers. There are 8 functions in a concrete family, parameterized over a real number. The inverted sub-type is also in the same concrete family.

Using functions from concrete families, one can use Boolean algebra to define an arbitrary sub-set of the real numbers, which becomes a family of functions parameterized over multiple real numbers. The existential path is then constructed by the following law:

$$\exists \mathrm{add}\{\sum i \{ \prod n \{ g_{in} \} \}, \sum j \{ \prod m \{ g_{jm} \} \}\} \Leftrightarrow \sum i, j \{ \prod n, m \{ \exists \mathrm{add}\{g_{in}, g_{jm}\} \} \}$$

This is possible because every sub-type defined as a Boolean expression of sub-types can be written in the following form, where `∑` means union and `∏` means intersection of sub-types:

$$\sum i \{ \prod n \{ g_{in} \} \}$$

It does not matter what the original Boolean expression is, since transformation to the form above allows one to swap any inverted sub-type with another function from the same family.

¬((> 2) ∧ (<= 3))	Original expression
¬(> 2) ∨ ¬(<= 3)	Using De Morgan's law
(<= 2) ∨ (> 3)	Target form of expression

The target form makes it easier to apply the law for the existential path.

For example:

$\exists add\{(> 2) \land (< 3), (> 3) \land (<= 4)\}$
$\exists add\{(> 2), (> 3)\} \land \exists add\{(< 3), (> 3)\} \land \exists add\{(> 2), (<= 4)\} \land \exists add\{(< 3), (<= 4)\}$
$(> 5) \land true_1 \land true_1 \land (< 7)$
$(> 5) \land (< 7)$

Another example:

$\exists add\{(> 2) \land (< 3) \lor (= 8), (<= 4)\}$
$\exists add\{(> 2) \land (< 3), (<= 4)\} \lor \exists add\{(= 8), (<= 4)\}$
$\exists add\{(> 2), (<= 4)\} \land \exists add\{(< 3), (<= 4)\} \lor \exists add\{(= 8), (<= 4)\}$
$true_1 \land (< 7) \lor (<= 12)$
$(< 7) \lor (<= 12)$
$(<= 12)$

Another way to write is to use the `+` operator on the sub-types:

$(> 2) \land (< 3) \lor (= 8) + (<= 4)$
$((> 2) \land (< 3) + (<= 4)) \lor ((= 8) + (<= 4))$
$((> 2) + (<= 4)) \land ((< 3) + (<= 4)) \lor ((= 8) + (<= 4))$
$true_1 \land (< 7) \lor (<= 12)$
$(< 7) \lor (<= 12)$
$(<= 12)$

One can use the following rules to easier remember operator precedence:

$a \land b + c = (a + c) \land (b + c)$
$a \lor b + c = (a + c) \lor (b + c)$
$a \land b \lor c + d = (a \land b + c) \lor (c + d) = (a + c) \land (b + c) \lor (c + d)$