Joining Functions in Exact Real Computation

Seokbin Lee, Sewon Park, Martin Ziegler

Korea Advanced Institute of Science and Technology

26th July 2019

Exact Real Computation (1/4)

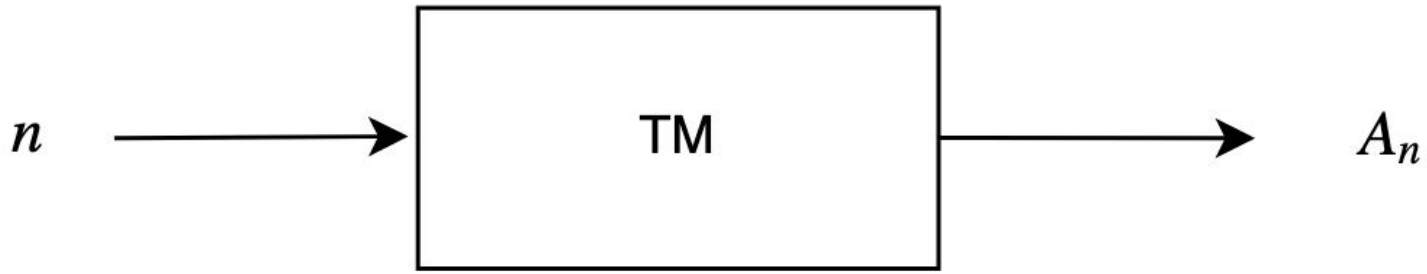
Definition (A. Turing, 1936).

A number $x \in \mathbb{R}$ is *computable* if there is a Turing machine that can, for any $n \in \mathbb{N}$, print the n-th digit of x.

Exact Real Computation (2/4)

Definition.

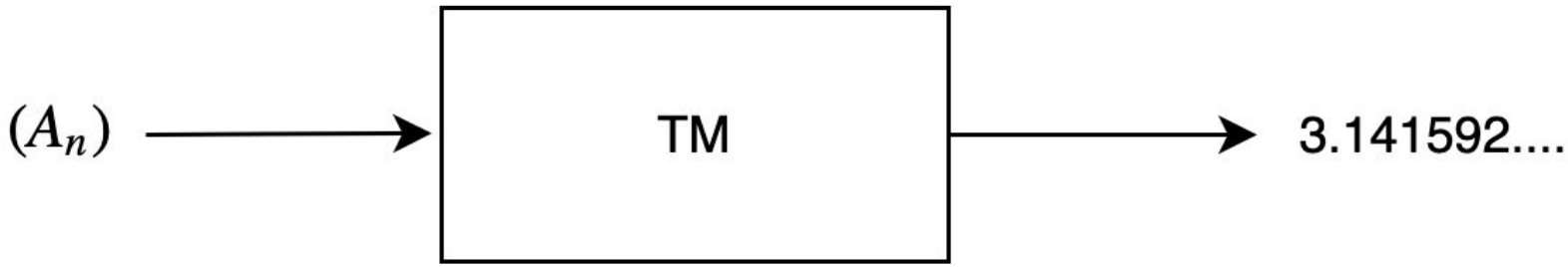
A number $x \in \mathbb{R}$ is computable if there is a computable sequence of rational numbers (A_n) such that $|x - A_n| < 2^{-n}$ for all $n \in \mathbb{N}$.



Exact Real Computation (3/4)

Definition.

A real function f is computable if there is a Turing machine that, for all r in the domain, computes a sequence (A_n) approximating the sequence, and outputs $f(r)$.



From now on, we say "numbers" to mean *computable* numbers, and similarly for functions.

Exact Real Computation (4/4)

Exact Real Computation aims to emphasize the reals as encodings of sequences.

Definition.

In Exact Real Computation, the *representation* of $r \in \mathbb{R}$ is an infinite encoding of a rational sequence (A_n) where $|r - A_n| < 2^{-n}$.

This construction has a caveat:

Fact.

Testing inequality "$x \neq y$" for two numbers x and y is equivalent to the Halting problem, so is undecidable. In particular, a test "$x > 0$" cannot return in the case $x = 0$.

However, this construction also has a merit...

Turing-completeness of ERC (1/2)

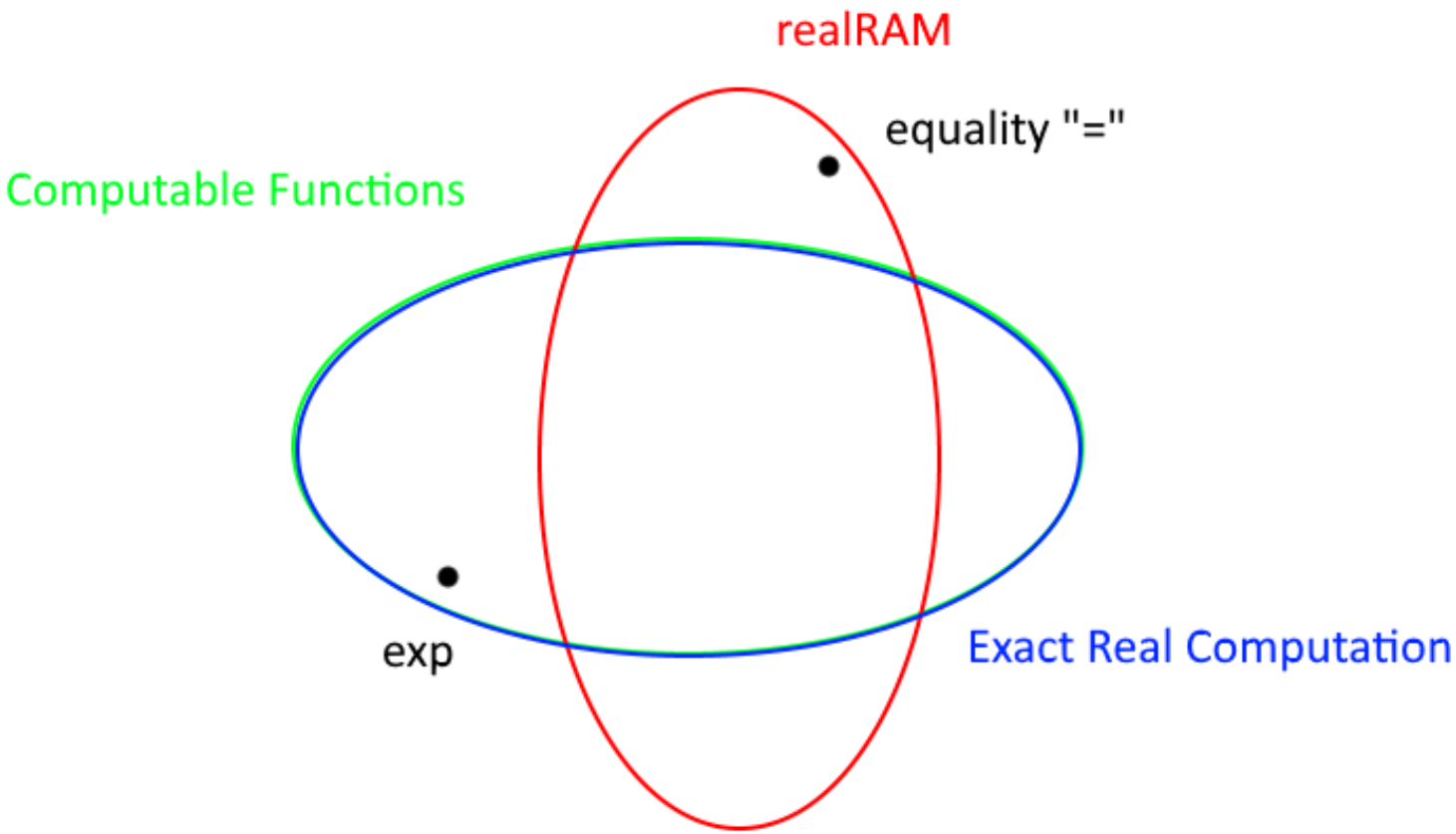
Lemma (F. Braußе, P. Collins, J. Kanig, S. Kim, M. Konečný, G. Lee, N, Müller, E. Neumann, S. Park, N. Preining, M. Ziegler, 2016).

The language of computable numbers is Turing-complete; a function is computable iff it can be expressed in Exact Real Computation. Similarly, the language of functionals is also Turing complete.

This differs from other paradigms such as realRAM (BSS machine), which does not have Turing-completeness.

Turing-completeness of ERC (2/2)

The following diagram illustrates this point.



Primitive Operations (1/2)

Recall the Kleene and Priest ternary logic $\{0, 1, \perp\}$, where $\perp$ means "undefined".
We introduce two primitive operations under this paradigm. The first is a `select` function that takes two values in $\{0, 1, \perp\}$.

Definition (Multi-valued `select`).

The multi-valued `select` function takes two inputs b and c from $\{0, 1, \perp\}$, and computes as follows:

$$\texttt{select}(b, c) = \begin{cases} 0 & \text{if } b \text{ is defined} \\ 1 & \text{if } c \text{ is defined} \\ 0/1 & \text{if both are defined} \\ \perp & \text{if neither are defined} \end{cases}$$

We note that this is computable!

Primitive Operations (2/2)

The second operation is a modification of the well-known "inline-if" ternary operation `b?x:y`.

Definition (Variant of inline-if).

The following variant of the ternary conditional is defined as follows: given $b \in \{0, 1, \perp\}$ and $x, y \in \mathbb{R}$,

$$b \,?\, x \,:\, y = \begin{cases} x & \text{if } b = 1 \\ y & \text{if } b = 0 \\ x/y & \text{if } b = \perp \wedge x = y \\ \perp & \text{if } b = \perp \wedge x \neq y \end{cases}$$

We note that this is also computable!
Using these operations, we can construct a workaround for testing equality of two numbers.

Joining Functions (1/2)

We are interested in *joining* two functions.

Definition.

The *join* of two functions $f : A \to \mathbb{R}$ and $g : B \to \mathbb{R}$, where $A, B \subseteq \mathbb{R}$ and $f = g$ on $A \cap B$, is a function $h : A \cup B \to \mathbb{R}$ which equals f on A and equals g on B.

We remark that this makes sense because the language is Turing-complete.
Our goal is to express joins of functions explicitly, in terms of the primitive operations `select` and `b?x:y`.

Example.

Given functions $f : [0;1] \to \mathbb{R}$ and $g : [-1;0] \to \mathbb{R}$ such that $f(0) = g(0)$, their join is the function from $[-1;1]$ to $\mathbb{R}$ given by

$$t \;\mapsto\; \big(\, (t > 0)\;?\; f(t) \;:\; g(t) \,\big)$$

Joining Functions (2/2)

We can generalize such joins to more than two functions taking domains in $\mathbb{R}^d$.

Definition.

Given functions $f_1 : A_1 \subseteq \mathbb{R}^d \to \mathbb{R}$, $f_2 : A_2 \subseteq \mathbb{R}^d \to \mathbb{R}$, ..., $f_n : A_n \subseteq \mathbb{R}^d \to \mathbb{R}$, pairwise agreeing on values on the intersections of their respective domains, the join of the function is a function on $A_1 \cup A_2 \cup ... \cup A_n$ that equals f_i on A_i, for all i.

How do we express these joins explicitly?

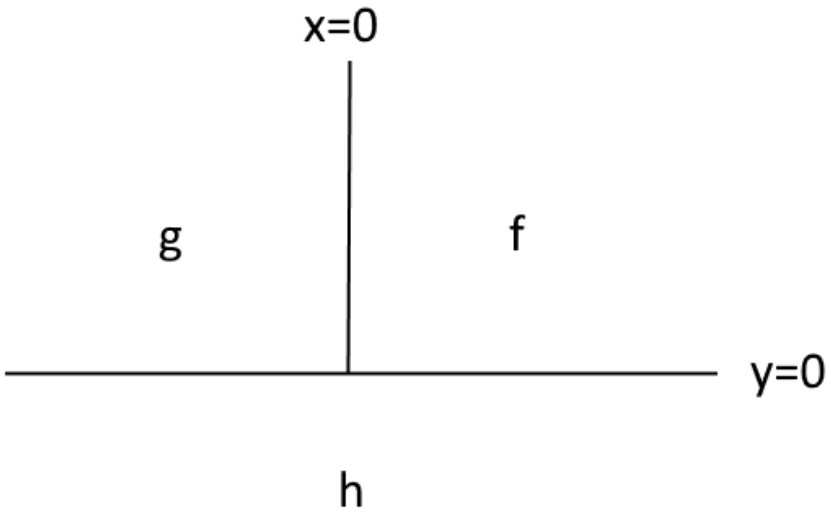
Results (1/4)

We consider three functions f, g, and h defined on subsets of $\mathbb{R}^2$ with pairwise boundaries as lines.
The simplest result is as follows:

Theorem.

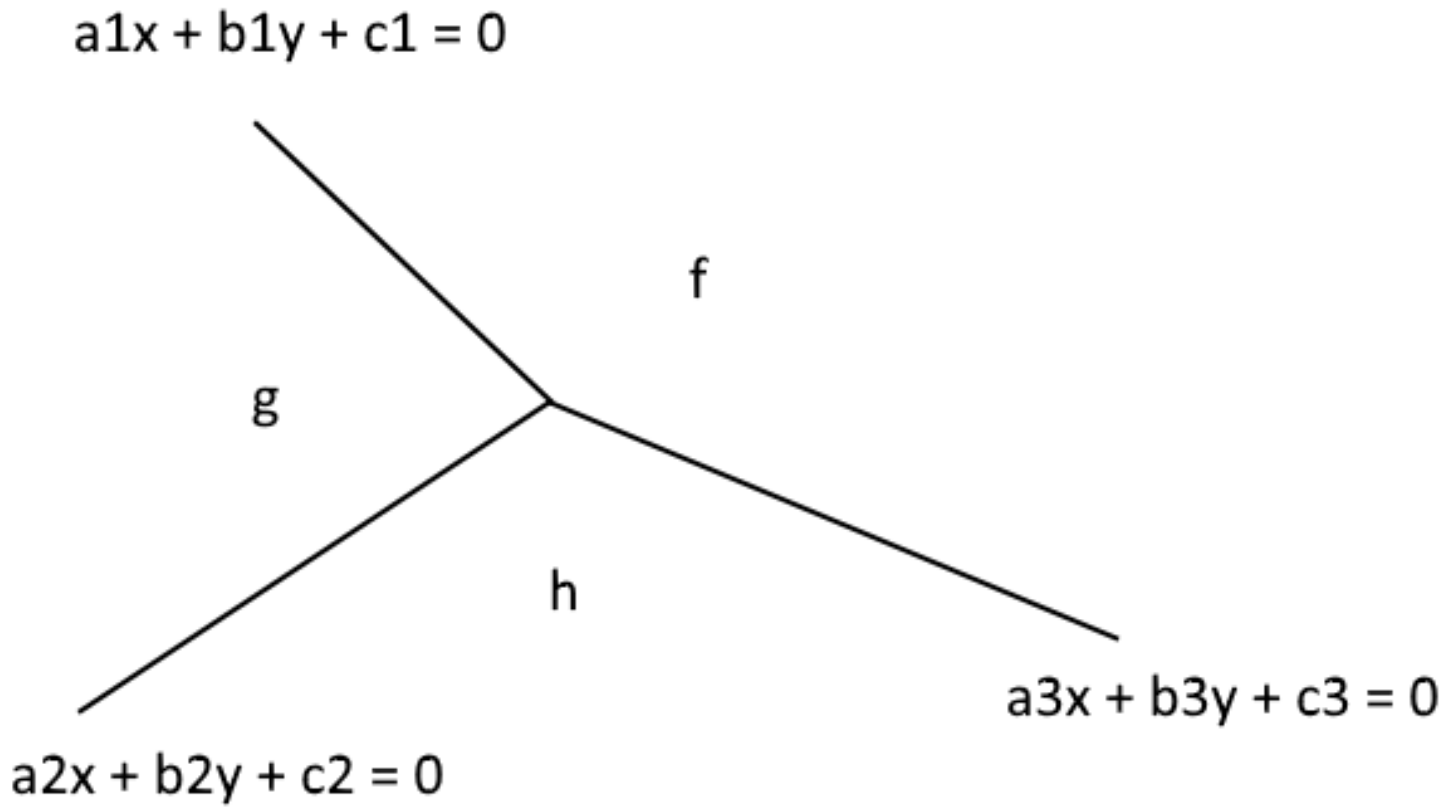
The join of three functions $f : [0;1] \times [0;1] \to \mathbb{R}$, $g : [-1;0] \times [0;1] \to \mathbb{R}$, *and* $h : [-1;1] \times [-1;0] \to \mathbb{R}$ *is given as*

$$(s,t) \;\mapsto\; \big(\,(t>0)\;?\;\big(\,(s>0)\;?\;f(s,t) : g(s,t)\,\big) : h(s,t)\,\big)$$



Results (2/4)

The above can be generalized. Consider any three lines meeting at a single point, as in the diagram.



Results (3/4)

Theorem.

The join of three functions f, g, and h with domain boundaries $a_1x + b_1y + c_1 = 0$, $a_2x + b_2y + c_2 = 0$, and $a_3x + b_3y + c_3 = 0$ as in the diagram can be expressed as follows, where $a_1, a_2, a_3 > 0$, $b_3 > 0$, and the three lines intersect at a point:

$$
\begin{aligned}
&(s,t) \mapsto \\
&\quad \Big(\big((a_1 s + b_1 t + c_1 > 0) \vee (a_2 s + b_2 t + c_2 > 0) \big) \; ? \\
&\quad \big((a_3 s + b_3 t + c_3 > 0) \; ? \; h(s,t) : f(s,t) \big) : g(s,t) \Big)
\end{aligned}
$$

We remark that the *logical or* $\vee$ respects the ternary logic; for instance, $1 \vee \bot \equiv 1$.

Results (4/4)

Theorem.

The join of three functions f, g, and h with domain boundaries $a_1x + b_1y + c_1 = 0$, $a_2x + b_2y + c_2 = 0$, and $a_3x + b_3y + c_3 = 0$ as in the diagram can be expressed as follows, where $a_1, a_2, a_3 > 0$ and $b_3 < 0$, and the three lines intersect at a point:

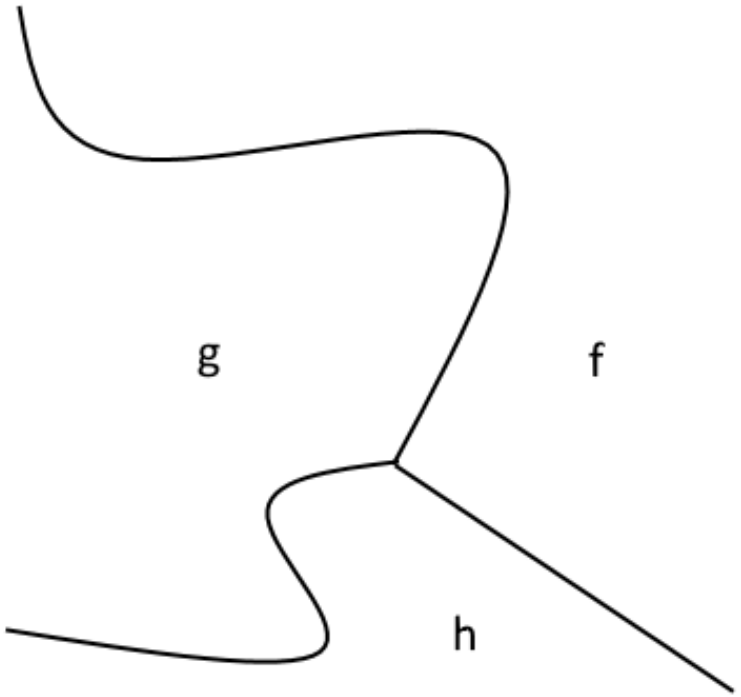
$$
\begin{aligned}
(s,t) \mapsto & \\
& \Big(\big((a_1 s + b_1 t + c_1 > 0) \vee (a_2 s + b_2 t + c_2 > 0) \big) \ ? \\
& \big((a_3 s + b_3 t + c_3 > 0) \ ? \ f(s,t) : h(s,t) \big) : g(s,t) \Big)
\end{aligned}
$$

Summary and Remaining Questions (1/2)

We have demonstrated that such explicit joins of three functions on 2 variables is indeed possible under the new primitive operations. These methods could be used to generalize even further to multiple functions in higher dimensions.
It is likely that we may be able to simplify the expressions. In particular, we may be able to remove cases such as "$b_3 > 0$".

Summary and Remaining Questions (2/2)

Separately, we had assumed lines as intersections of boundaries for the domains of functions. This could potentially be generalized; investigations into generalizations may be worthwhile.



Thank you!