Abstract Geometrical Computation and the Blum, Shub and Smale Model (CiE 2007 + 2008)

Jérôme Durand-Lose

Laboratoire d'Informatique Fondamentale d'Orléans,
Université d'Orléans, Orléans, France

27 mars 2008 – MC2, LIP, ÉNS Lyon

Context

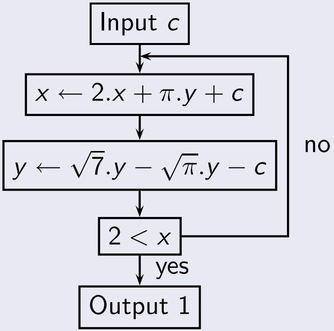
- Computation on the continuum
- Analog/continuous models

- No consensus on an *analog Turing thesis*
- Relating various models

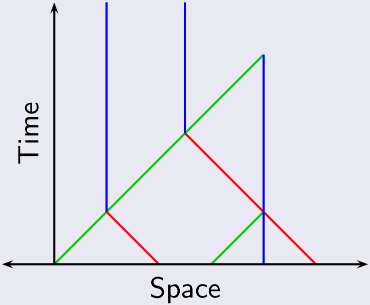
- Blum, Shub and Smale model on $\mathbb{R}$ ($\mathbb{R}$-BSS)
  [Blum, Shub, and Smale, 1989]
  [Blum, Cucker, Shub, and Smale, 1998]
- Abstract geometrical computation (AGC)
  [JDL: MCU 04, CiE 05]

Goal: to relate these

Lin-$\mathbb{R}$-BSS



Abstract Geometrical Computation



Full-BSS (last part only)

Polynomial instead of *linear* (*i.e.* unconstrained multiplication)

Definition: Linear-$\mathbb{R}$-BSS

- Variables hold real numbers
- Computing linear functions over the variables
- Branch with $0 \leq$ test

Definition: Linear-$\mathbb{R}$-BSS on unbounded sequences

- Variables hold real numbers
- Computing linear functions over the variables
- Branch with $0 \leq$ test

To handle unbounded number of variables

- Variables ordered in an infinite array
- `shift` operator

Definition: Linear-$\mathbb{R}$-BSS on unbounded sequences

- Variables hold real numbers
- Computing linear functions over the variables
- Branch with $0 \leq$ test

To handle unbounded number of variables

- Variables ordered in an infinite array
- `shift` operator

Not the easiest to handle

- Switch for an equivalent model [Novak, 1995]
- $\rightsquigarrow$ *linear-$\mathbb{R}$-Unlimited Register Machines*

Definition: Linear-$\mathbb{R}$-Unlimited Register Machines

- An accumulator
- Infinite array of registers (holding real numbers)

Basic operations

- Done on the accumulator
- Store into / load from a register
- Addition of the value of a register
- Multiplication by a constant

Definition: Linear-$\mathbb{R}$-Unlimited Register Machines

- An accumulator
- Infinite array of registers (holding real numbers)
- Finitely many addresses (special registers)

Basic operations

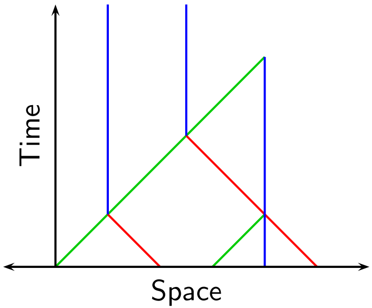
- Done on the accumulator
- Store into / load from a register
- Addition of the value of a register
- Multiplication by a constant

Indirect addressing

- Through address registers
- `Dec` and `Inc`

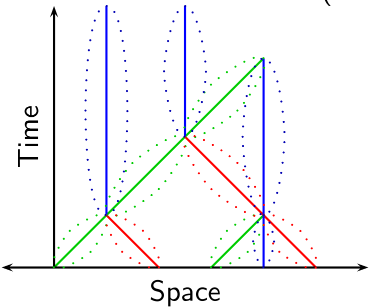
Definition: Abstract Geometrical Computation and Signal Machines

$$\mathbb{R} \times \mathbb{R}^+$$



Definition: Abstract Geometrical Computation and Signal Machines

$\mathbb{R} \times \mathbb{R}^{+}$



Signal

(Meta-signal, position)

Position

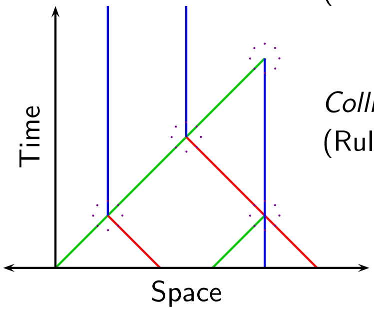
(x, t)

Meta-signal

$\mu = (\iota, \nu)$

Definition: Abstract Geometrical Computation and Signal Machines

$\mathbb{R} \times \mathbb{R}^+$



Signal

(Meta-signal, position)

Position

(x, t)

Meta-signal

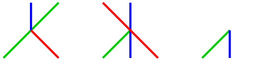
$\mu = (\iota, \nu)$

Collision

(Rule, position)

Rule

$\rho = \{\mu_i^-\}_i \rightarrow \{\mu_j^+\}_j$



Computation

Lin-$\mathbb{R}$-URM

- Initial values of the registers
- Runs until stops
- Final values of the registers

Abstract Geometrical Computation

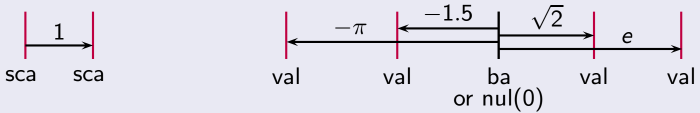
- Signals in the initial configuration
- Runs until no more collision is possible
- Signals in the final configuration

Out of scope

- Infinite computations
- Accumulations (AGC is subject to Zeno's paradox) except in the last part

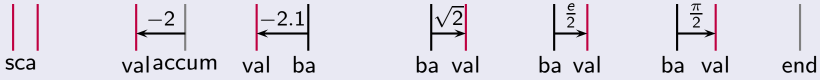
Encoding of Lin-ℝ-URM configuration - 1

Real values



Scale and positions of val for values $-\pi$, -1.5, 0, $\sqrt{2}$ and e

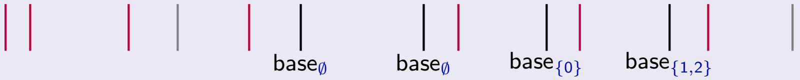
Accumulator and registers



Accumulator (-2) and registers (-2.1, $\sqrt{2}$, $\frac{e}{2}$, $\frac{\pi}{2}$)

Encoding of Lin-ℝ-URM configuration - 2

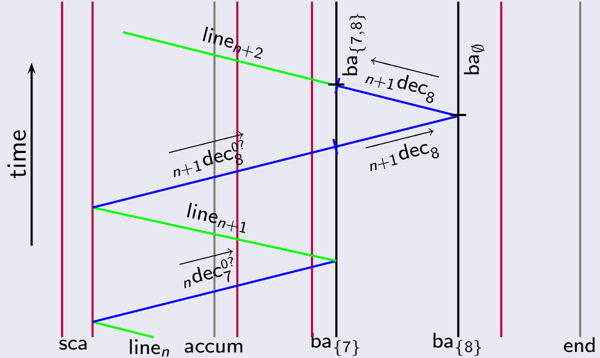
Addresses



$A_0 = 2$, $A_1 = 3$ and $A_2 = 3$

State / line number

Updating the addresses – dec A_i

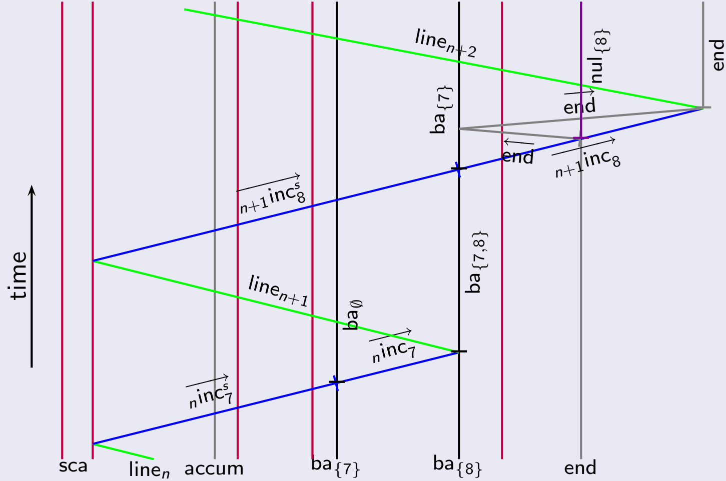


n: dec A_7

$n+1$: dec A_8

(check for zero)

Updating the addresses – inc A_i

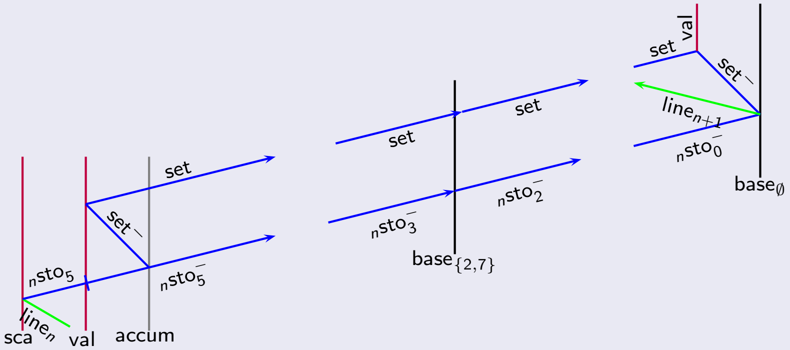


n: inc A_7

$n{+}1$: inc A_8

Load and store

Example of `store` R_5 (direct addressing)



Load, store and add

(not shown)

- Previous value must be disposed of

Indirect addressing

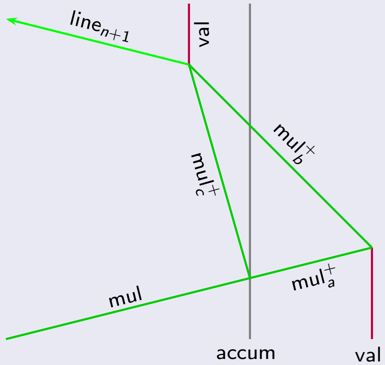
- Move until marker found

Addition

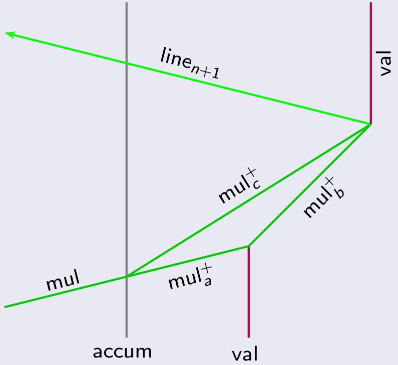
- With the same construction but considering val instead of ba

Multiplication

$\alpha < 1$ (here $\alpha = -\frac{1}{2}$)



$1 < \alpha$ (here $\alpha = 2$)



Normalisation

Addressing a new counter

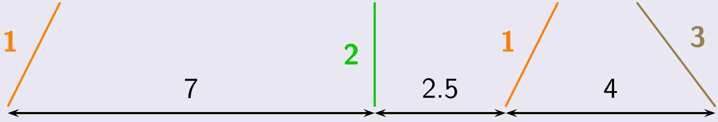
$\rightsquigarrow$ Configuration is enlarged with a register at 0

Values too large

$\rightsquigarrow$ Everything is scaled down

Encoding of Signal machine configurations

Alternatively signal id and distance to the next



1	7	2	2.5	1	4	3	-1	0	0	⋯

Updating

Main loop

- Find date of next collision
  - Go through the configuration
  - Find minimal time to zero a distance
- Update the distances (Go through the configuration again)
- Treat the collision(s) (Go through the configuration again)
  - Find maximal sequences of zero distances
  - Replace the signals and shift the rest if necessary

Linear URM because. . .

- Finite number of signals (and collision rules)

  $\rightsquigarrow$ Bounded number of signals involved in a collision

  $\rightsquigarrow$ Switch (nested `if`) to get to the right case
- Constant speeds $\rightsquigarrow$ Constants of the lin-ℝ-URM

What is missing

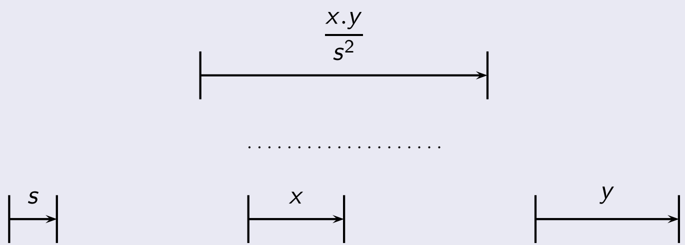
Internal multiplication

- Implementing

$$x, y \longrightarrow x.y$$

- Any polynomial function can be generated

Geometrical interpretation?



Computing with an infinite sum

Computation

- Pre-treatment to ensure $0 < y < 1$
- Binary extension of y:
$$y = \mathrm{y}_0.\mathrm{y}_1\mathrm{y}_2\mathrm{y}_3\cdots$$
- Computation
$$xy = \sum_{0\leq i} \mathrm{y}_i \left(\frac{x}{2^i}\right)$$

Potentially infinite loop

	p_{n+1}	x_{n+1}	y_{n+1}	b_{n+1}
if $y_n = 0$	stop			
else if $b_n < y_n$	$p_n + x_n$	$x_n/2$	$y_n - b_n$	$b_n/2$
else	p_n	$x_n/2$	y_n	$b_n/2$

Implementation

Infinite sum

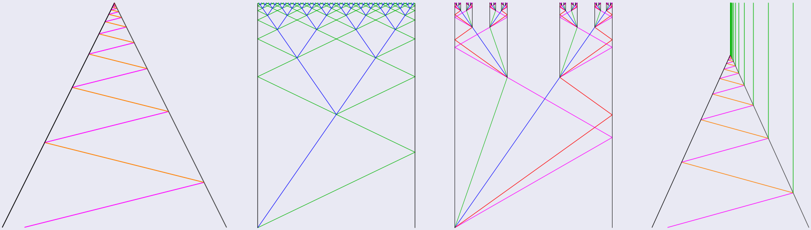
- Accumulation
- Ensure accumulation

For AGC

- Handeling accumulations
- Basic primitives
  - extract the y_i
  - generate and sometimes add $\frac{x}{2^i}$

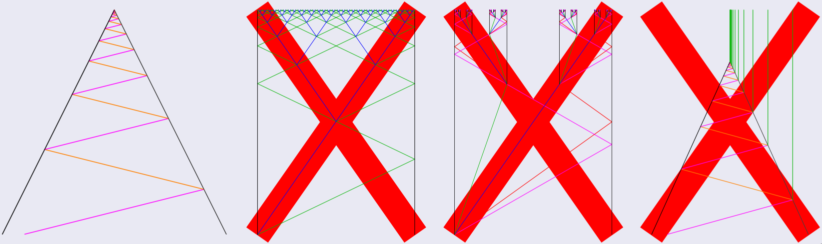
Extending AGC with simple accumulation

Various cases



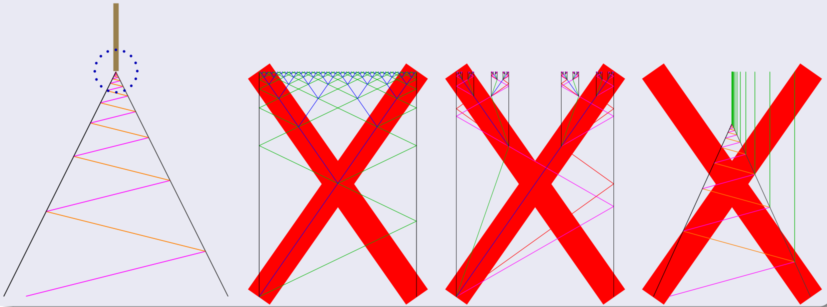
Extending AGC with simple accumulation

Various cases: only isolated accumulations are covered



Extending AGC with simple accumulation

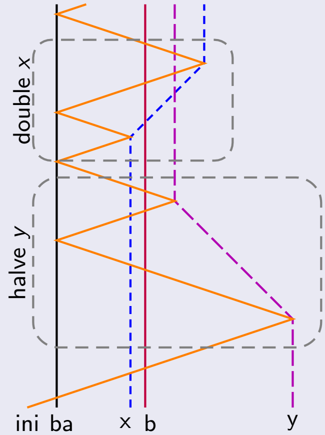
Various cases: only isolated accumulations are covered



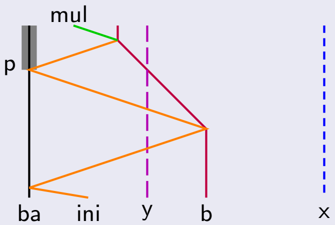
Any accumulation leaves the same signal

Preparing the data for multiplication

First iteration

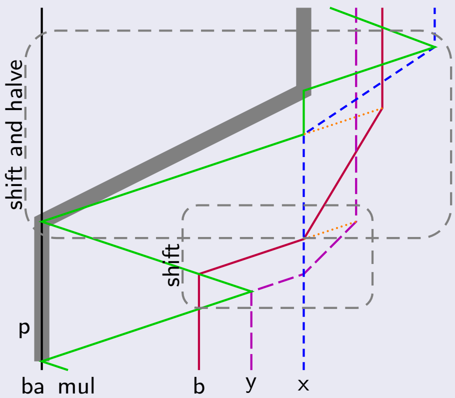


Loop end

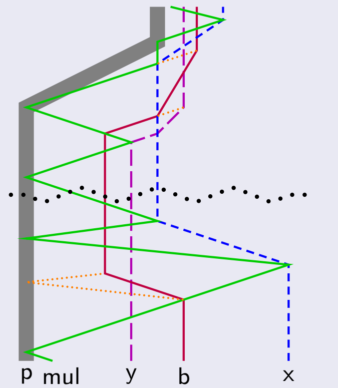


Main loop

First iteration



Two iterations



Accumulation at $x.y$

Property ensuring accumulation in bounded time

- No distant signal (everything is packed on the side)
- Everything is scaling down

Correction

- Clear from construction

Achievement

- Inner multiplication
- Full BSS!

More than BSS

Square rooting by approximation

$$b_n^2 \leq a < \left(b_n + \frac{1}{2^n}\right)^2$$

recurrence formula

$$a - \left(b_n + \frac{1}{2^{n+1}}\right)^2 = a - b_n^2 - \frac{b_n}{2^n} - \frac{1}{4^{n+1}} = d_n - e_n - f_n$$

Table for the loop

	b_{n+1}	d_{n+1}	e_{n+1}	f_{n+1}	g_{n+1}
if $d_n - e_n - f_n = 0$	stop				
else if $0 < d_n - e_n - f_n$	$b_n + g_n$	$d_n - e_n - f_n$	$e_n/2 + f_n$	$f_n/4$	$g_n/2$
else	b_n	d_n	$e_n/2$	$f_n/4$	$g_n/2$

Conclusion

Relation

- Computing capability of ACG with simple accumulation is *strictly more* than for BSS

Side result

- Square rooting of 2 can be defined with a rational SM
- *Accumulation can happen at irrational coordinate*

Conclusion

Result: Equivalence

- Lin-$\mathbb{R}$-URM / Lin-$\mathbb{R}$-BSS
- Abstract geometrical computation / Signal machines

Result: Stronger computing capability with accumulation

- AGC with accumulation is strictly stronger than full BSS ($\sqrt{\ }$)
- Accumulation at irrational coordinate

Future work

- Characterize the power of AGC with isolated accumulation
- BSS counterpart of accumulations in AGC (Zeno effect/lim)
- Link AGC with transfinite computations and computable analysis