Introducing fractal computation

Jérôme Durand-Lose
Joint work with Denys Duchier *and* Maxime Senot

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

COPCOM '11 — Cluj Napoca

Brute force search

- easy check
- (too) many to check

Physical limitations

- classical parallelism
- unconventional computation

Fractal parallelism

- continuous space and time idealization
- using a fractal to broadcast

Hard to find but easy to check

- Very important in, *e.g.*, asymmetric cryptography

Example: find 3 integers a, b, and c such that...

$$10 < a < b < c < 100$$

and

$$a^2 + b^2 = c^2$$

Hard to find but easy to check

- Very important in, *e.g.*, asymmetric cryptography

Example: find 3 integers a, b, and c such that...

$$10 < a < b < c < 100$$

and

$$a^2 + b^2 = c^2$$

Easy

$$30^2 + 40^2 = 50^2$$

Hard to find but easy to check

- Very important in, *e.g.*, asymmetric cryptography

Example: find 3 integers a, b, and c such that...

$$10 < a < b < c < 100$$

and

$$a^2 + b^2 = c^2$$

Easy

$$30^2 + 40^2 = 50^2$$

Indeed

$$900 + 1\,600 = 2\,500$$

Link with NP problems

NP: class of *decision* problems

- `YES` : easily proved (polynomial time) with the right *certificate*
- `NO` : there is no certificate

Link with NP problems

NP: class of *decision* problems

- `YES` : easily proved (polynomial time) with the right *certificate*
- `NO` : there is no certificate

Trivial algorithm

- try all certificates

Link with NP problems

NP: class of *decision* problems

- `YES` : easily proved (polynomial time) with the right *certificate*
- `NO` : there is no certificate

Trivial algorithm

- try all certificates
- BUT there are too many to try

SAT

Instance

ϕ : Boolean formula with free variables $x_1, x_2, \ldots, x_n$.

Question

Is there a way to set the free variables such that ϕ is true?

Example

$$\phi = (x_1 \vee \neg x_2) \wedge x_3$$

SAT

Instance

ϕ : Boolean formula with free variables $x_1, x_2, \ldots, x_n$.

Question

Is there a way to set the free variables such that ϕ is true?

Example

$$\phi = (x_1 \vee \neg x_2) \wedge x_3$$

x_1 $\overline{x_2}$ x_3

SAT

Instance

ϕ : Boolean formula with free variables $x_1, x_2, \ldots, x_n$.

Question

Is there a way to set the free variables such that ϕ is true?

Example

$$\phi = (x_1 \vee \neg x_2) \wedge x_3$$

x_1 $\overline{x_2}$ x_3

Complexity according to n, the number of variables

- Test a valuation: linear in the length of the formula *easy*
- Number of valuations: 2^n *exponential growth*

Brute force parallelism

- *Try them all!*

- A few... *easy*
- Polynomially many... *might take a while*
- Exponentially many... *not feasible*

...unless

- exponentially many computing units

Classical computation

Parallelism / grid / cloud

- 1 microprocessor $\Rightarrow$ 1 unit of space
- $\approx d^3$ processor at distance d
- $\rightsquigarrow$ exponential diameter for exponentially many processors
- $\rightsquigarrow$ exponential communication time

Unconventional Computation and Natural Computation (UC NC)

DNA computation

- very small
- totally uncentralized computation
- still at some (faraway) point the *soup* will be too big

Unconventional Computation and Natural Computation (UC NC)

DNA computation

- very small
- totally uncentralized computation
- still at some (faraway) point the *soup* will be too big

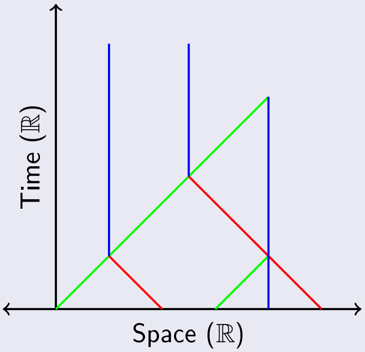
Quantum computation

- superposition of exponentially many states
- big decoherence/stability problem
- limited set of operations (unitary operators and projections)

Idealized model

Wanted properties

- no space nor time granularity
- can work at any scale
- can compute



- continuous space
- continuous time

Vocabulary

- Signal (meta-signal)
  - dimensionless
- Collision (rule)
  - deterministic
  - uniform

Example: find the middle

Meta-signals, speed

M, $S(\mathrm{M}) = 0$

Collision rules

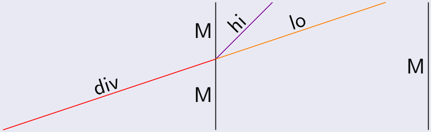
Example: find the middle

Meta-signals, speed

M, $S(\mathrm{M}) = 0$
div, $S(\mathrm{div}) = 3$

Collision rules

Example: find the middle



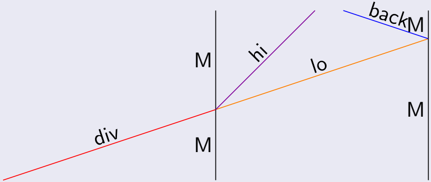
Meta-signals, speed

M, $S(\mathrm{M}) = 0$
div, $S(\mathrm{div}) = 3$
hi, $S(\mathrm{hi}) = 1$
lo, $S(\mathrm{lo}) = 3$

Collision rules

$\{ \text{div, M} \} \rightarrow \{ \text{M, hi, lo} \}$

Example: find the middle



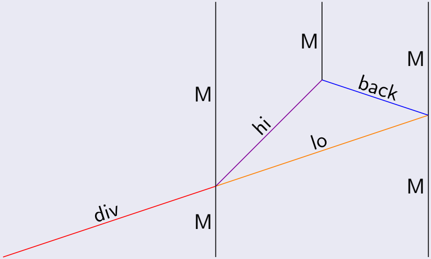
Meta-signals, speed

M, $S(\mathrm{M}) = 0$
div, $S(\mathrm{div}) = 3$
hi, $S(\mathrm{hi}) = 1$
lo, $S(\mathrm{lo}) = 3$
back, $S(\mathrm{back}) = -3$

Collision rules

$\{$ div, M $\}$ $\rightarrow$ $\{$ M, hi, lo $\}$
$\{$ lo, M $\}$ $\rightarrow$ $\{$ back, M $\}$

Example: find the middle



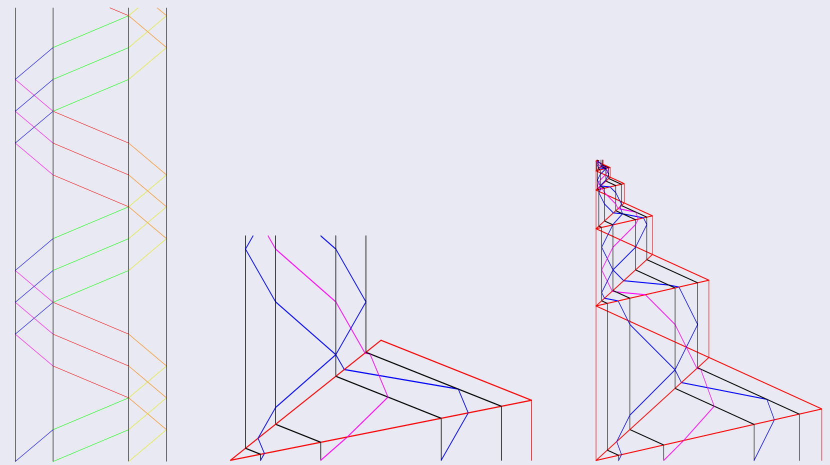
Meta-signals, speed

M, $S(\mathrm{M}) = 0$
div, $S(\mathrm{div}) = 3$
hi, $S(\mathrm{hi}) = 1$
lo, $S(\mathrm{lo}) = 3$
back, $S(\mathrm{back}) = -3$

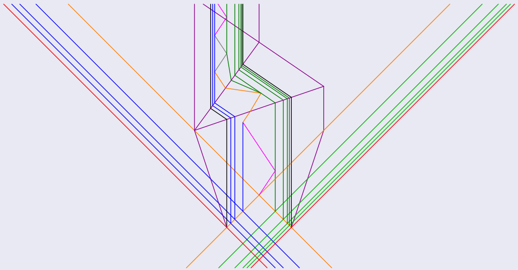
Collision rules

$\{$ div, M $\} \rightarrow \{$ M, hi, lo $\}$
$\{$ lo, M $\} \rightarrow \{$ back, M $\}$
$\{$ hi, back $\} \rightarrow \{$ M $\}$

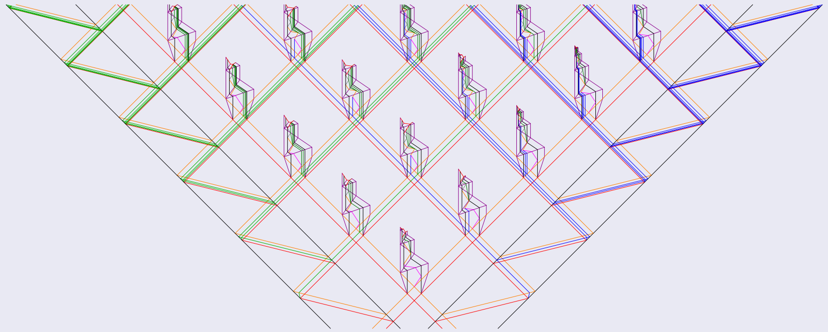
Complex dynamics



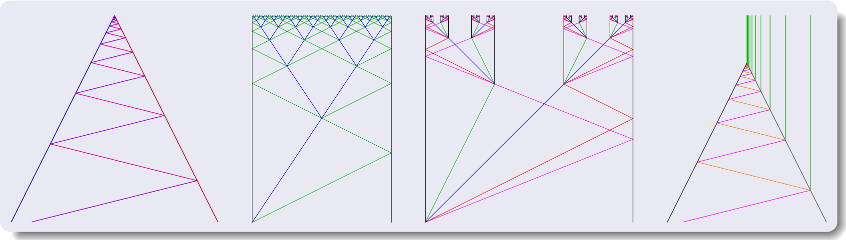
Complex dynamics



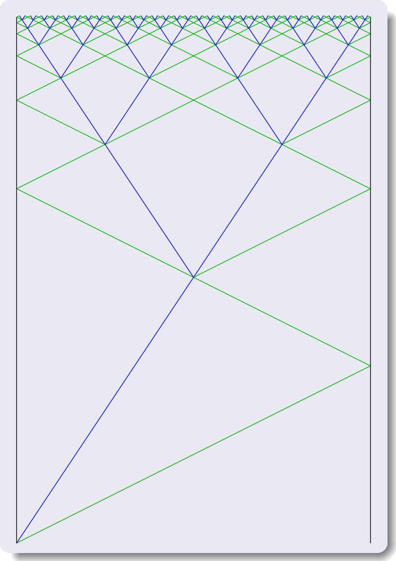
Complex dynamics



Fractal space-time diagrams



Using a fractal to compute



Scheme

Use the structure
to dispatch the computation

QSat: quantified satisfaction problem

- Quantified boolean formula (without free variable)
- Find its logical value
- PSPACE-complete problem

Example

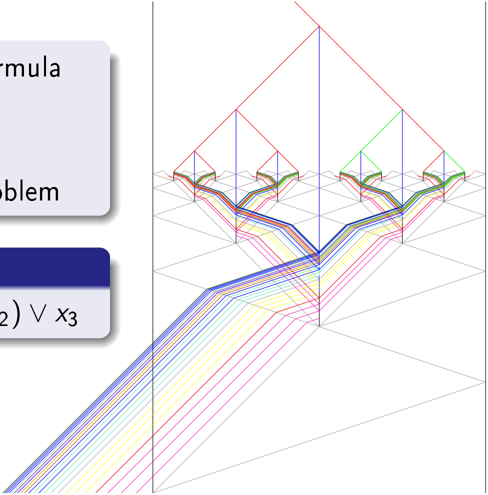
$$\phi = \exists x_1 \forall x_2 \forall x_3 \quad (x_1 \wedge \neg x_2) \vee x_3$$

QSat: quantified satisfaction problem

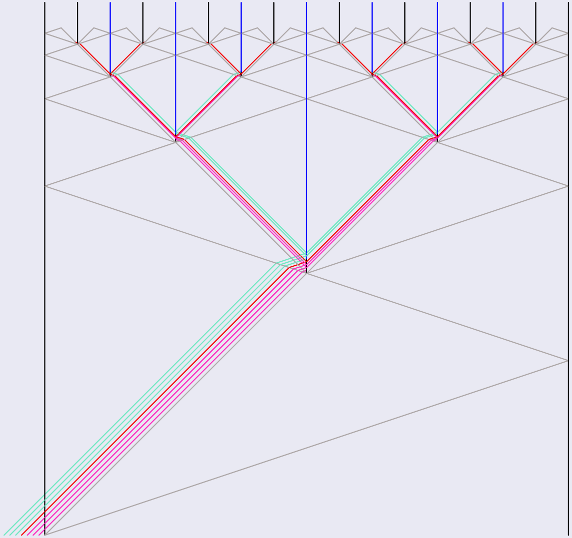
- Quantified boolean formula (without free variable)
- Find its logical value
- PSPACE-complete problem

Example

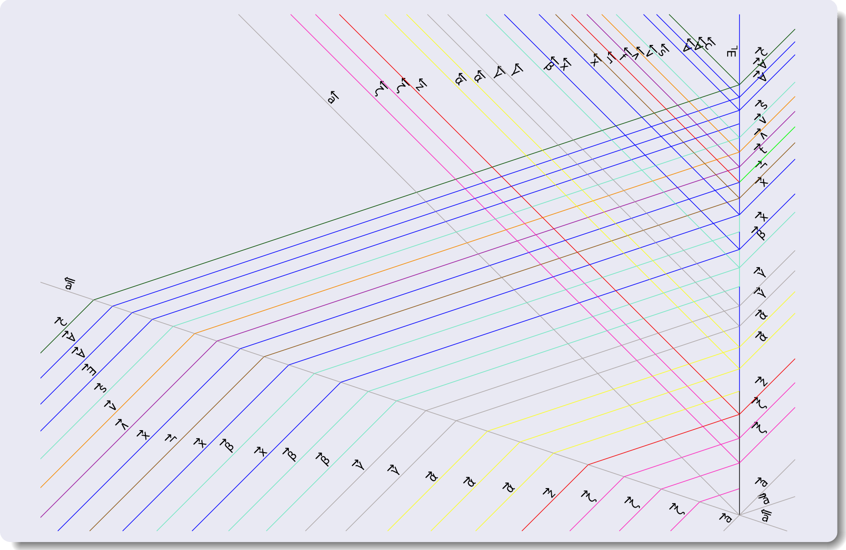
$\phi = \exists x_1 \forall x_2 \forall x_3 \quad (x_1 \wedge \neg x_2) \vee x_3$



Building the tree / combinatorial comb



Lens and variables assignment



Formula evaluation

$$\phi = \exists x_1 \forall x_2 \forall x_3 \quad (x_1 \wedge \neg x_2) \vee x_3$$

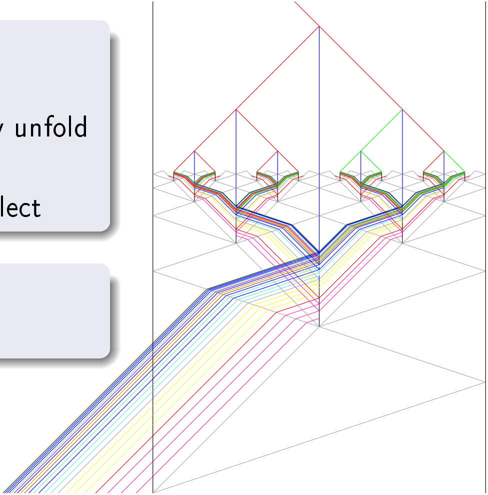
Case here

$$(\text{true} \wedge \neg\text{true}) \vee \text{true}$$

Fractal computation

- continuous media
  (time and space)
- structure automaticaly unfold
- structure used
  to dispatch and to collect

- generic machine
- modular programming



Complexities

Time

- constant (as a duration)
- cubic (as max length of collision chain)

Space

- constant (as a width)
- exponential (as max number of independant signals, antichain)

Complexities

Time

- constant (as a duration)
- cubic (as max length of collision chain)

Space

- constant (as a width)
- exponential (as max number of independant signals, antichain)

NB: Super-Turing Model with accumulations

- decide Halt in finite duration and width...

Future work

Fractal computation

- non deterministic processes
- higher complexity classes

Automatic discretization

- into a cellular automata
- nice properties are lost
- easy way to define a CA

Future work

Fractal computation

- non deterministic processes
- higher complexity classes

Automatic discretization

- into a cellular automata
- nice properties are lost
- easy way to define a CA

Thank you for your attention