Ways to Compute in Euclidean Frameworks

Jérôme Durand-Lose

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

June 2017 – UCNC 2017 – Fayetteville, AR, USA

Euclidean Geometry

Key ingredient of the dynamics

Dependence from initial positions

Localization is meaningful

Results are geometrical

Outline

- **Compass and Rule** (analog)
- **Mondrian Automata** (hybrid)
- **Piece-wise Constant Derivative** (hybrid)
- **Signal Machines** (hybrid)
- **Intrinsic Universality among Signal Machines**

Not addressed (too well known)

- **Tile Assemble Systems** (discrete)
- **Cellular Automata** (discrete)

1. Compass and Rule (Huckenbeck, 1989, 1991)
2. Mondrian Automata (Jacopini and Sontacchi, 1990)
3. Piece-wise Constant Derivative (Asarin et al., 1995; Bournez, 1999)
4. Signal Machines
5. Intrinsically Universal Family of Signal Machines

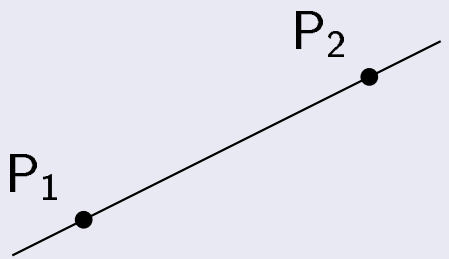
Geometrical Construction Machine

Space: $\mathbb{R}^2$

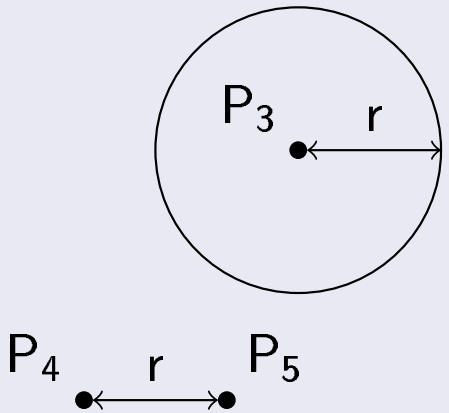
- 2-dimensional Euclidean space
- figure constructions

Point

Straight Line



Circle



- Variables and automaton

Dynamics: Program / Automaton (1/2)

Creation Operators

- circle out of 3 points

Dynamics: Program / Automaton (1/2)

Creation Operators

- circle out of 3 points
- line out of 2 points E

E Possible error

Dynamics: Program / Automaton (1/2)

Creation Operators

- circle out of 3 points
- line out of 2 points E
- point as intersection of 2 lines E

E Possible error

Dynamics: Program / Automaton (1/2)

Creation Operators

- circle out of 3 points
- line out of 2 points E
- point as intersection of 2 lines E
- point as intersection of 2 circles E N

E Possible error

N Non-deterministic

Dynamics: Program / Automaton (1/2)

Creation Operators

- circle out of 3 points
- line out of 2 points E
- point as intersection of 2 lines E
- point as intersection of 2 circles E N
  different from a point E (N)

E Possible error

N Non-deterministic

Dynamics: Program / Automaton (1/2)

Creation Operators

- circle out of 3 points
- line out of 2 points E
- point as intersection of 2 lines E
- point as intersection of 2 circles E N
  different from a point E (N)
- point as intersection of a circle and a line E N
  different from point E (N)

E Possible error

N Non-deterministic

Dynamics: Program / Automaton (2/2)

Other Operations

branch	If a point is in a set $\mathcal{A}$ (oracle)
output	Write a point, a ligne or a circle
termination	End

Computation Success

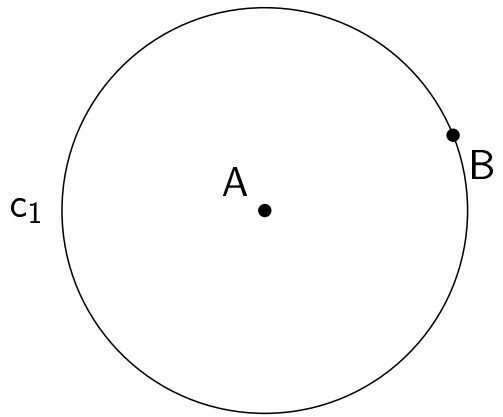
- no branch with Error
- all branches
  - end with End
  - have the same output ($\rightsquigarrow$ result)

Example: Computing the Middle (of Distinct Points)

1: $c_1 \leftarrow$ Circle (center A, radius d(A,B))
2: $c_2 \leftarrow$ Circle (center B, radius d(A,B))
3: $p_1 \leftarrow$ Intersection (c_1, c_2)
4: $p_2 \leftarrow$ Intersection (c_1, c_2) different from p_1
5: $d_1 \leftarrow$ Line (p_1, p_2)
6: $d_2 \leftarrow$ Line (A, B)
7: $p_3 \leftarrow$ Intersection (d_1, d_2)
8: Write p_3
9: End

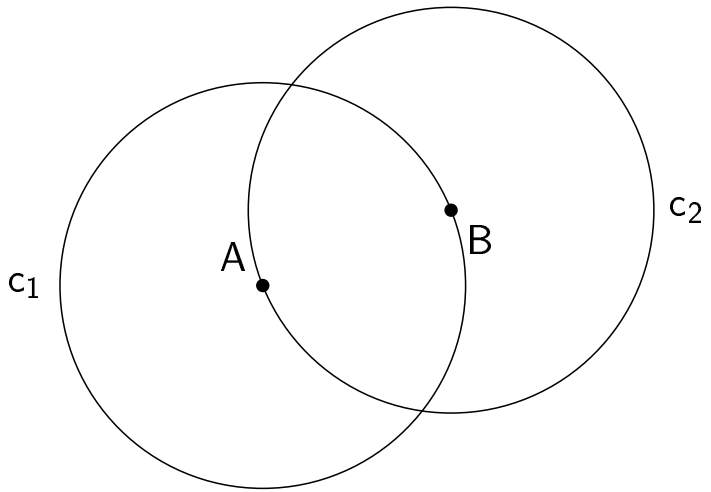
Example: Computing the Middle (of Distinct Points)

1: $c_1 \leftarrow$ Circle (center A, radius d(A,B))
2: $c_2 \leftarrow$ Circle (center B, radius d(A,B))
3: $p_1 \leftarrow$ Intersection (c_1, c_2)
4: $p_2 \leftarrow$ Intersection (c_1, c_2) different from p_1
5: $d_1 \leftarrow$ Line (p_1, p_2)
6: $d_2 \leftarrow$ Line (A, B)
7: $p_3 \leftarrow$ Intersection (d_1, d_2)
8: Write p_3
9: End



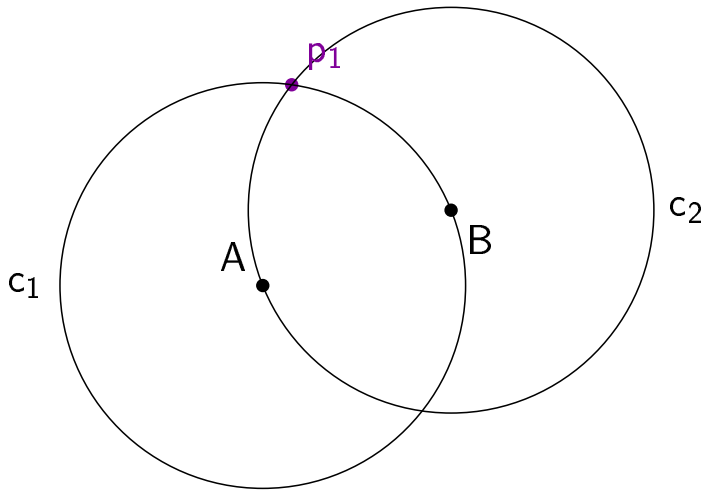
Example: Computing the Middle (of Distinct Points)

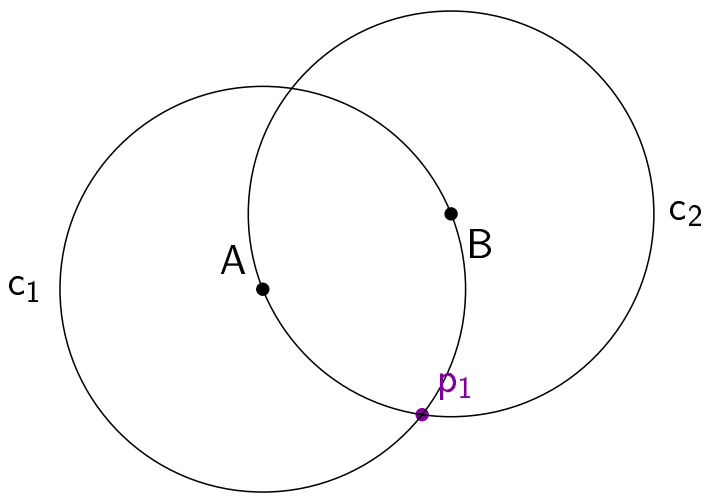
1: $c_1 \leftarrow$ Circle (center A, radius d(A,B))
2: $c_2 \leftarrow$ Circle (center B, radius d(A,B))
3: $p_1 \leftarrow$ Intersection (c_1, c_2)
4: $p_2 \leftarrow$ Intersection (c_1, c_2) different from p_1
5: $d_1 \leftarrow$ Line (p_1, p_2)
6: $d_2 \leftarrow$ Line (A, B)
7: $p_3 \leftarrow$ Intersection (d_1, d_2)
8: Write p_3
9: End



Example: Computing the Middle (of Distinct Points)

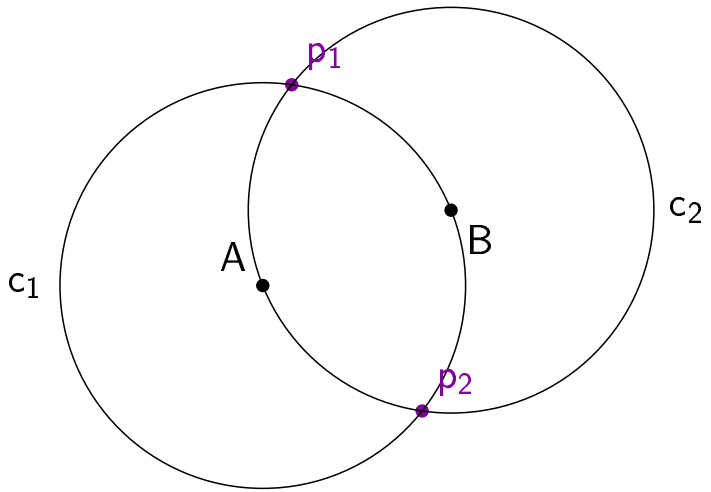
1: $c_1 \leftarrow$ Circle (center A, radius d(A,B))
2: $c_2 \leftarrow$ Circle (center B, radius d(A,B))
3: $p_1 \leftarrow$ Intersection (c_1, c_2)
4: $p_2 \leftarrow$ Intersection (c_1, c_2) different from p_1
5: $d_1 \leftarrow$ Line (p_1, p_2)
6: $d_2 \leftarrow$ Line (A, B)
7: $p_3 \leftarrow$ Intersection (d_1, d_2)
8: Write p_3
9: End

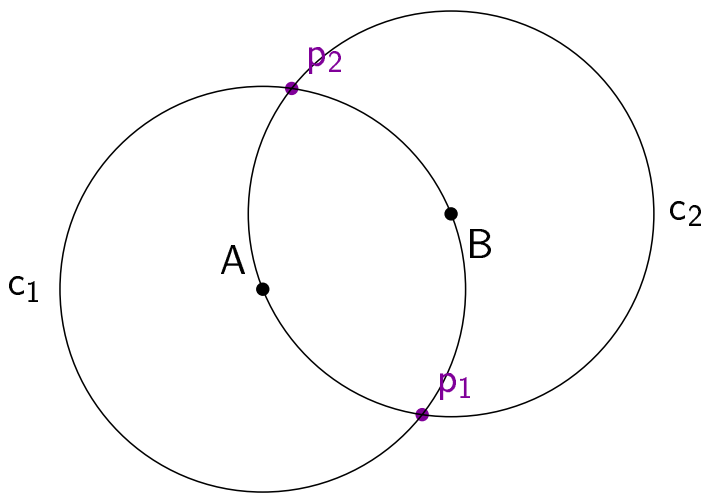




Example: Computing the Middle (of Distinct Points)

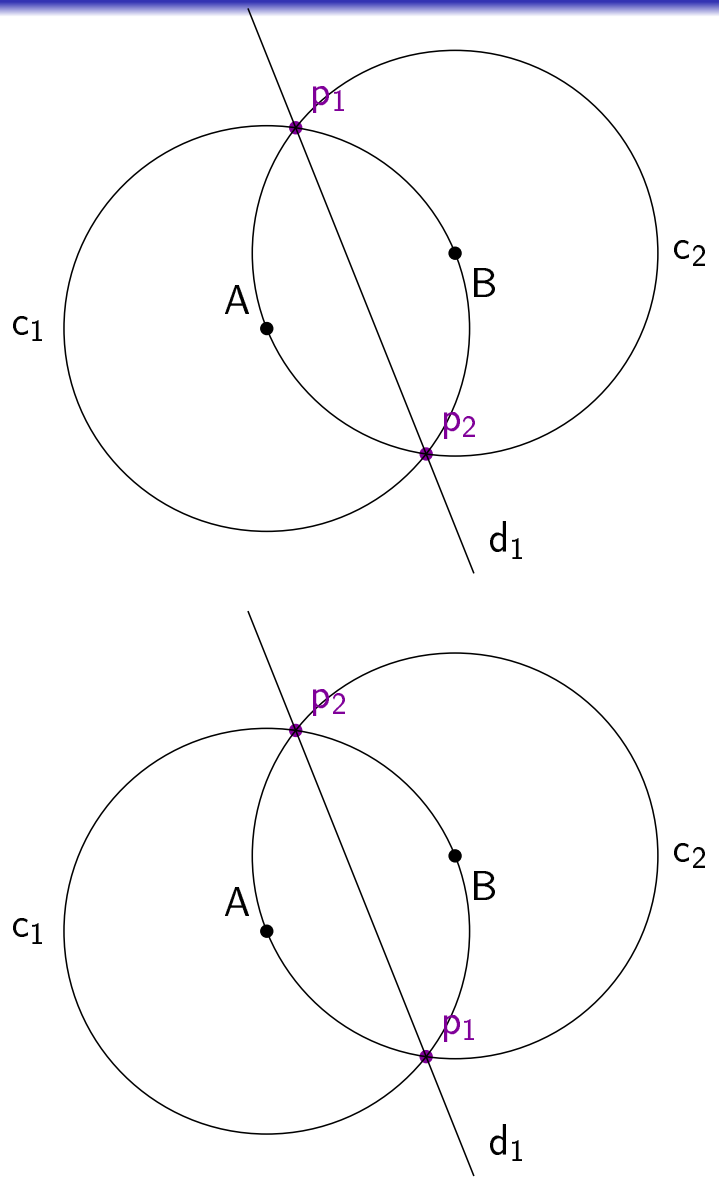
1: $c_1 \leftarrow$ Circle (center A, radius d(A,B))
2: $c_2 \leftarrow$ Circle (center B, radius d(A,B))
3: $p_1 \leftarrow$ Intersection (c_1, c_2)
4: $p_2 \leftarrow$ Intersection (c_1, c_2) different from p_1
5: $d_1 \leftarrow$ Line (p_1, p_2)
6: $d_2 \leftarrow$ Line (A, B)
7: $p_3 \leftarrow$ Intersection (d_1, d_2)
8: Write p_3
9: End





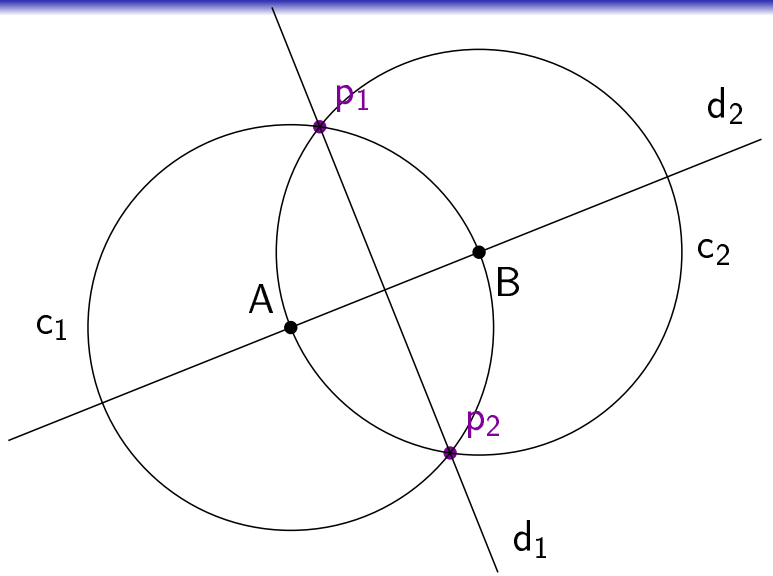
Example: Computing the Middle (of Distinct Points)

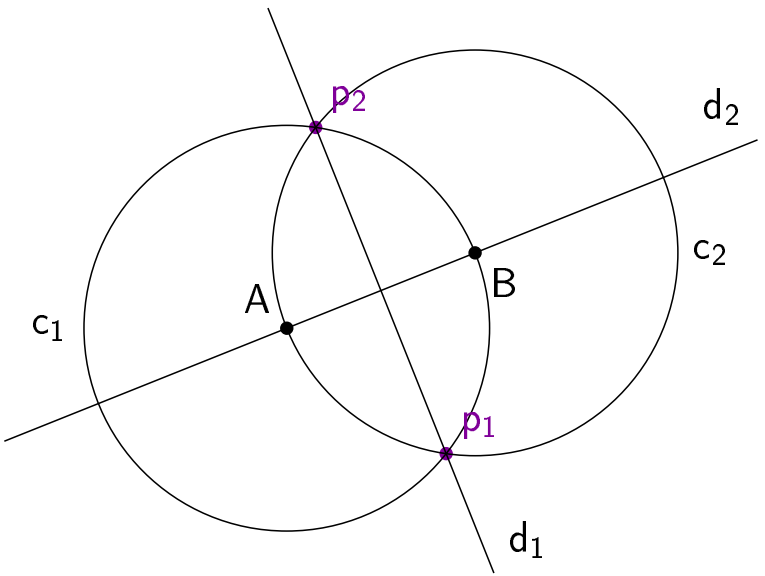
1: $c_1 \leftarrow$ Circle (center A, radius d(A,B))
2: $c_2 \leftarrow$ Circle (center B, radius d(A,B))
3: $p_1 \leftarrow$ Intersection (c_1, c_2)
4: $p_2 \leftarrow$ Intersection (c_1, c_2) different from p_1
5: $d_1 \leftarrow$ Line (p_1, p_2)
6: $d_2 \leftarrow$ Line (A, B)
7: $p_3 \leftarrow$ Intersection (d_1, d_2)
8: Write p_3
9: End



Example: Computing the Middle (of Distinct Points)

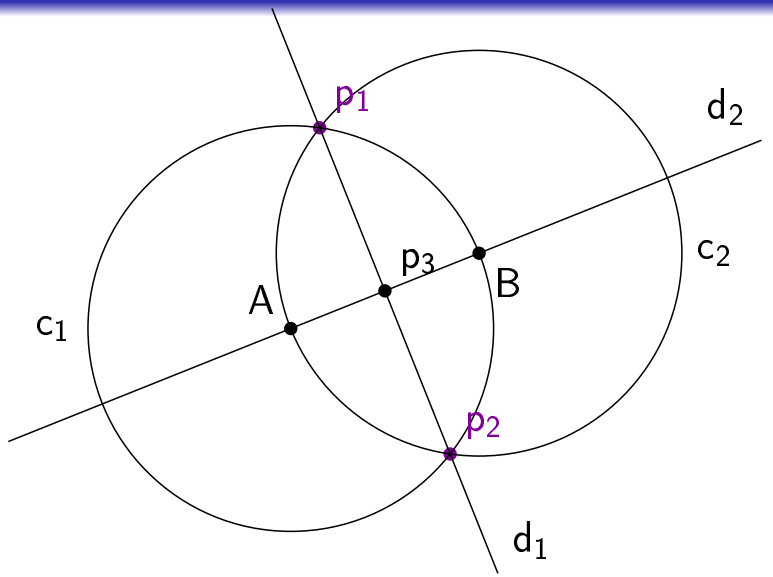
1: $c_1 \leftarrow$ Circle (center A, radius d(A,B))
2: $c_2 \leftarrow$ Circle (center B, radius d(A,B))
3: $p_1 \leftarrow$ Intersection (c_1, c_2)
4: $p_2 \leftarrow$ Intersection (c_1, c_2) different from p_1
5: $d_1 \leftarrow$ Line (p_1, p_2)
6: $d_2 \leftarrow$ Line (A, B)
7: $p_3 \leftarrow$ Intersection (d_1, d_2)
8: Write p_3
9: End

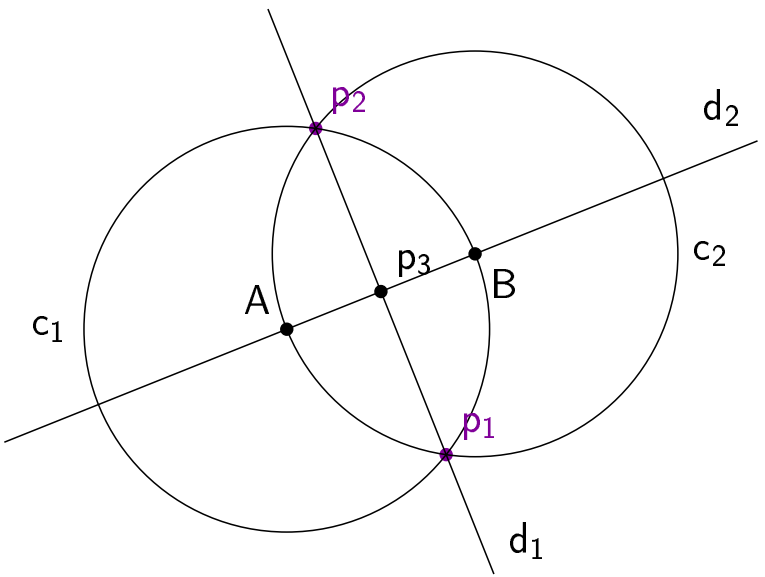




Example: Computing the Middle (of Distinct Points)

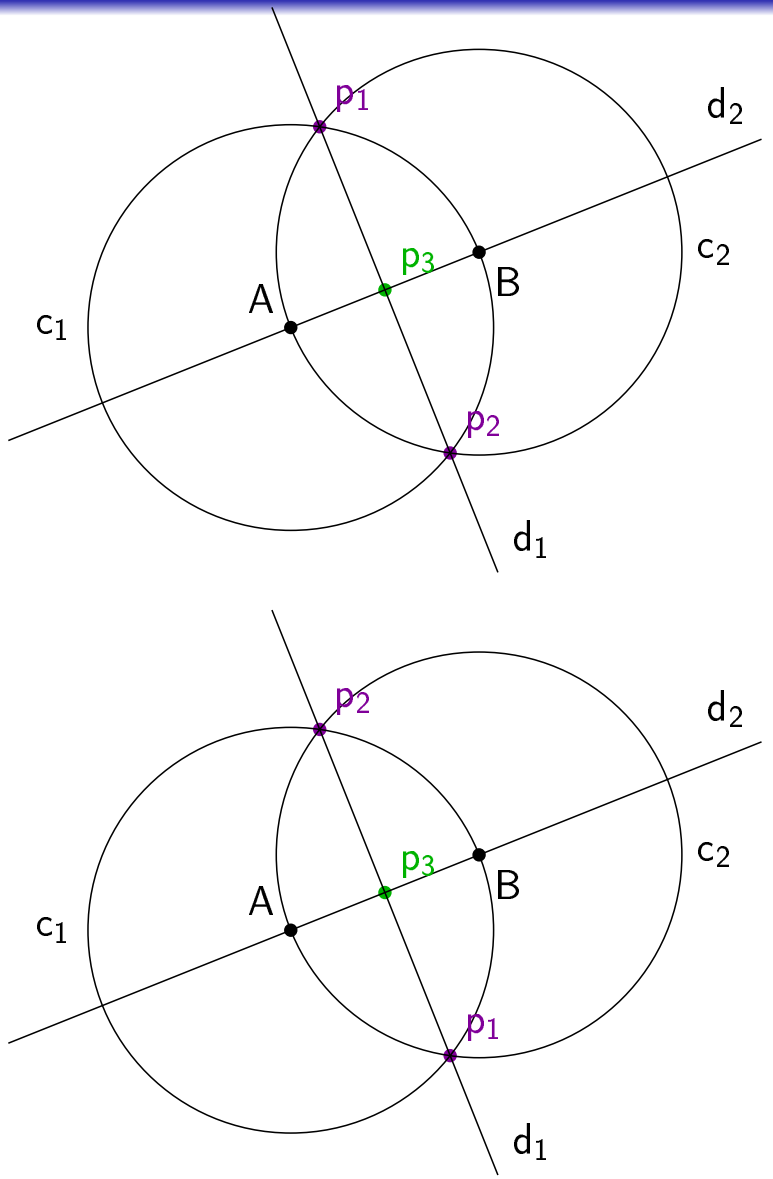
1: $c_1 \leftarrow$ Circle (center A, radius d(A,B))
2: $c_2 \leftarrow$ Circle (center B, radius d(A,B))
3: $p_1 \leftarrow$ Intersection (c_1, c_2)
4: $p_2 \leftarrow$ Intersection (c_1, c_2) different from p_1
5: $d_1 \leftarrow$ Line (p_1, p_2)
6: $d_2 \leftarrow$ Line (A, B)
7: $p_3 \leftarrow$ Intersection (d_1, d_2)
8: Write p_3
9: End





Example: Computing the Middle (of Distinct Points)

1: $c_1 \leftarrow$ Circle (center A, radius d(A,B))
2: $c_2 \leftarrow$ Circle (center B, radius d(A,B))
3: $p_1 \leftarrow$ Intersection (c_1, c_2)
4: $p_2 \leftarrow$ Intersection (c_1, c_2) different from p_1
5: $d_1 \leftarrow$ Line (p_1, p_2)
6: $d_2 \leftarrow$ Line (A, B)
7: $p_3 \leftarrow$ Intersection (d_1, d_2)
8: Write p_3
9: End

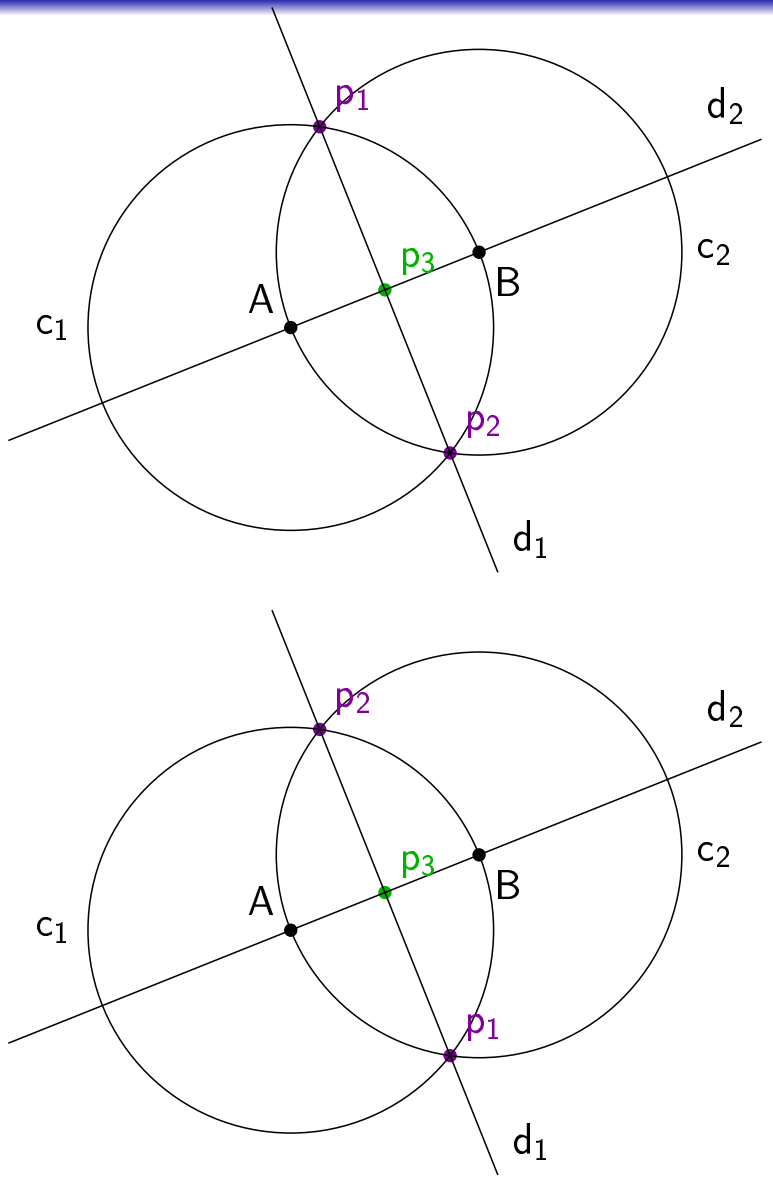


Example: Computing the Middle (of Distinct Points)

1: $c_1 \leftarrow$ Circle (center A, radius d(A,B))
2: $c_2 \leftarrow$ Circle (center B, radius d(A,B))
3: $p_1 \leftarrow$ Intersection (c_1, c_2)
4: $p_2 \leftarrow$ Intersection (c_1, c_2) different from p_1
5: $d_1 \leftarrow$ Line (p_1, p_2)
6: $d_2 \leftarrow$ Line (A, B)
7: $p_3 \leftarrow$ Intersection (d_1, d_2)
8: Write p_3
9: End

Two Branches

- success
- same output

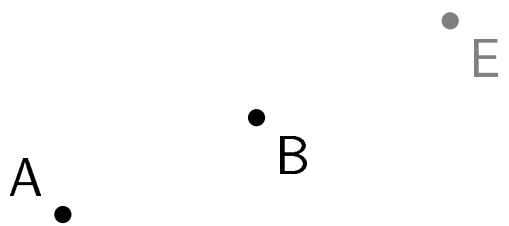


Exercise: Double a (non null) Vector...

Problem

Input A and B: 2 distinct points

Result E such that $\vec{AE} = 2\vec{AB}$



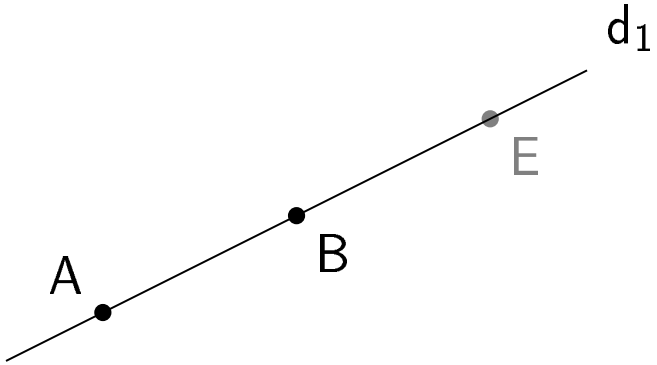
Exercise: Double a (non null) Vector...

Problem

Input A and B: 2 distinct points

Result E such that $\vec{AE} = 2\vec{AB}$

2: d ← Line (A, B)
3: c ← Circle (center B, radius d(A,B))
4: E ← Intersection (c, d) different from A
5: Write E
6: End



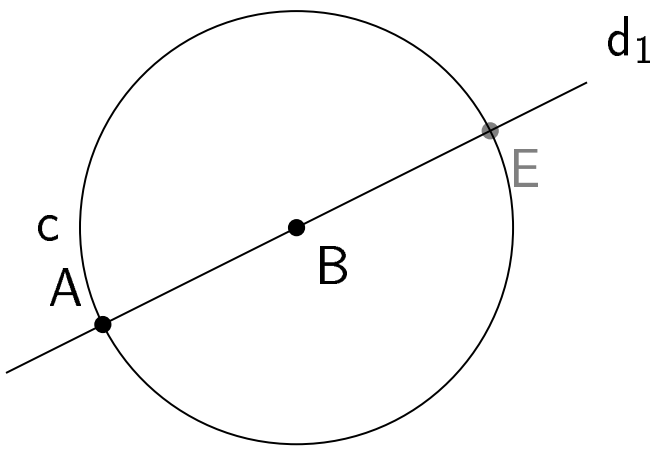
Exercise: Double a (non null) Vector...

Problem

Input A and B: 2 distinct points

Result E such that $\vec{AE} = 2\vec{AB}$

2: d ← Line (A, B)
3: c ← Circle (center B, radius d(A,B))
4: E ← Intersection (c, d) different from A
5: Write E
6: End



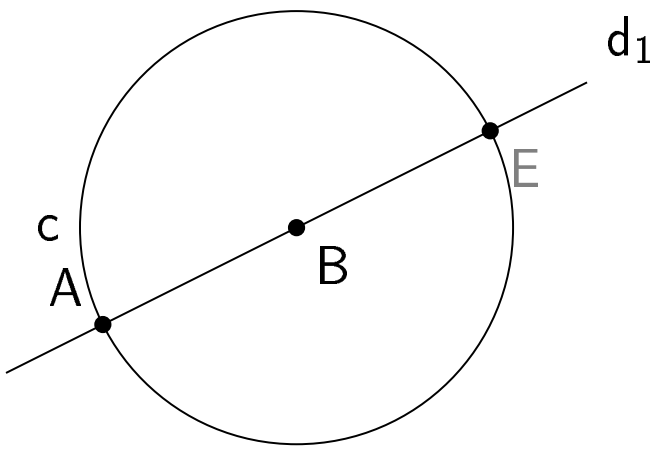
Exercise: Double a (non null) Vector...

Problem

Input A and B: 2 distinct points

Result E such that $\vec{AE} = 2\vec{AB}$

2: d ← Line (A, B)
3: c ← Circle (center B, radius d(A,B))
4: E ← Intersection (c, d) different from A
5: Write E
6: End



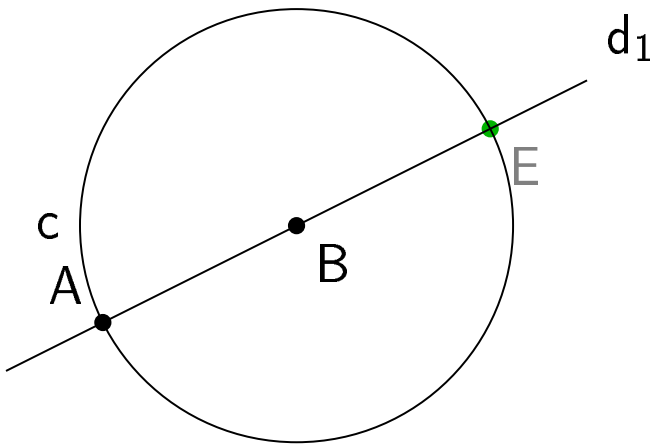
Exercise: Double a (non null) Vector...

Problem

Input A and B: 2 distinct points

Result E such that $\vec{AE} = 2\vec{AB}$

2: d ← Line (A, B)
3: c ← Circle (center B, radius d(A,B))
4: E ← Intersection (c, d) different from A
5: Write E
6: End



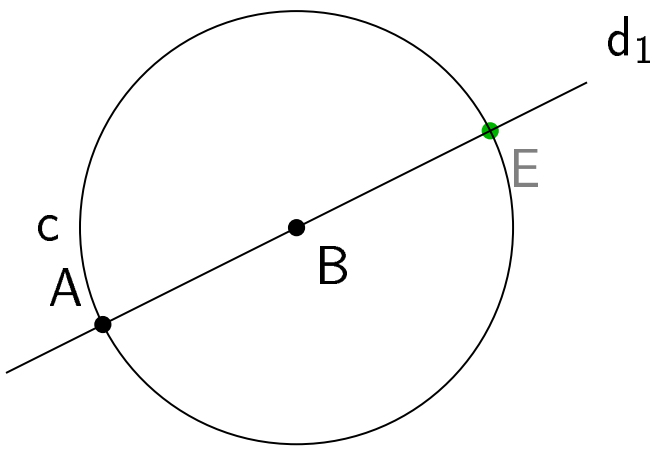
Exercise: Double a (non null) Vector...

Problem

Input A and B: 2 distinct points

Result E such that $\vec{AE} = 2\vec{AB}$

2: d $\leftarrow$ Line (A, B)
3: c $\leftarrow$ Circle (center B, radius d(A,B))
4: E $\leftarrow$ Intersection (c, d) different from A
5: Write E
6: End

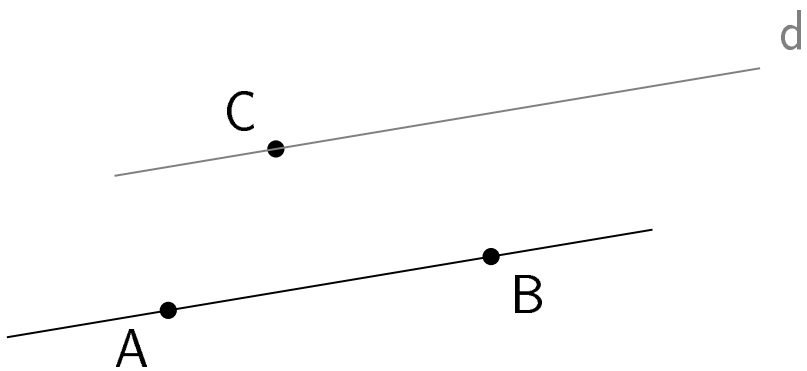


Exercise: Parallel Line Passing through a Point

Problem

Input A, B and C: 3 distinct points

Result d such that AB is parallel to d and C belongs to d

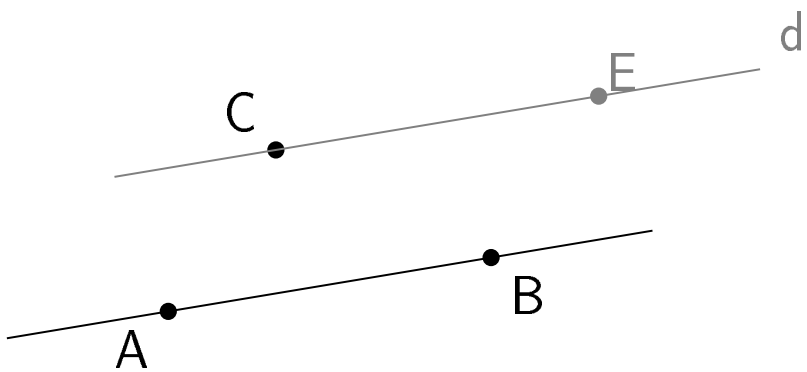


Exercise: Parallel Line Passing through a Point

Problem

Input A, B and C: 3 distinct points

Result d such that AB is parallel to d and C belongs to d

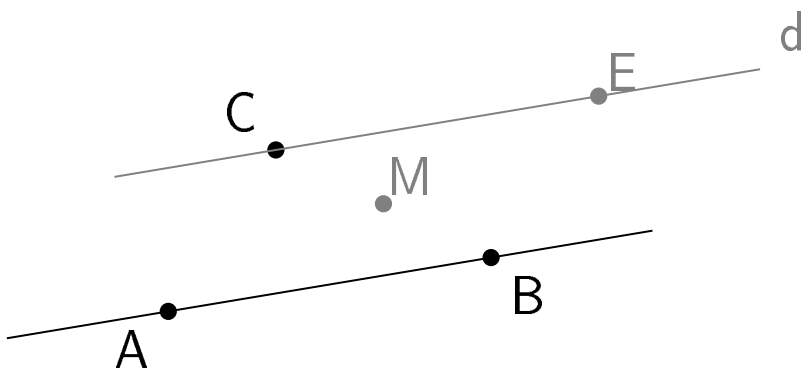


Exercise: Parallel Line Passing through a Point

Problem

Input A, B and C: 3 distinct points

Result d such that AB is parallel to d and C belongs to d



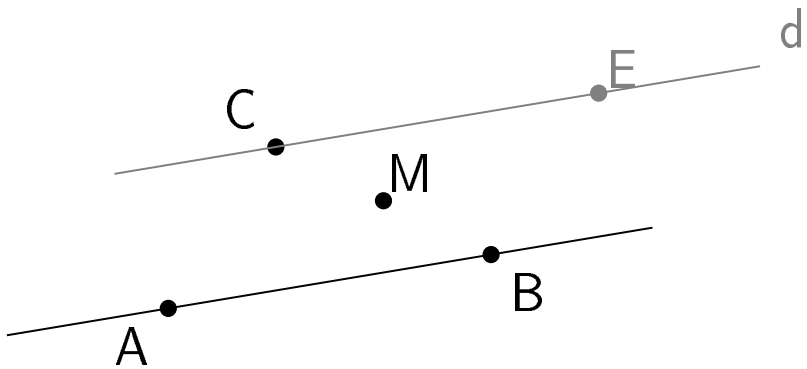
Exercise: Parallel Line Passing through a Point

Problem

Input A, B and C: 3 distinct points

Result d such that AB is parallel to d and C belongs to d

4: M$\leftarrow$ *MIDDLE* (B, C)
5: E$\leftarrow$ *DOUBLE_VECTOR* (A, M)
6: d$\leftarrow$ Line (C, E)
7: Write d
8: End



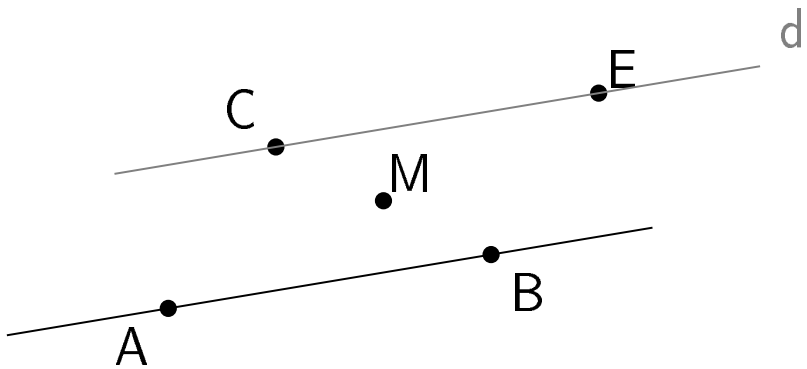
Exercise: Parallel Line Passing through a Point

Problem

Input A, B and C: 3 distinct points

Result d such that AB is parallel to d and C belongs to d

4: M$\leftarrow$ *MIDDLE* (B, C)
5: E$\leftarrow$ *DOUBLE_VECTOR* (A, M)
6: d$\leftarrow$ Line (C, E)
7: Write d
8: End



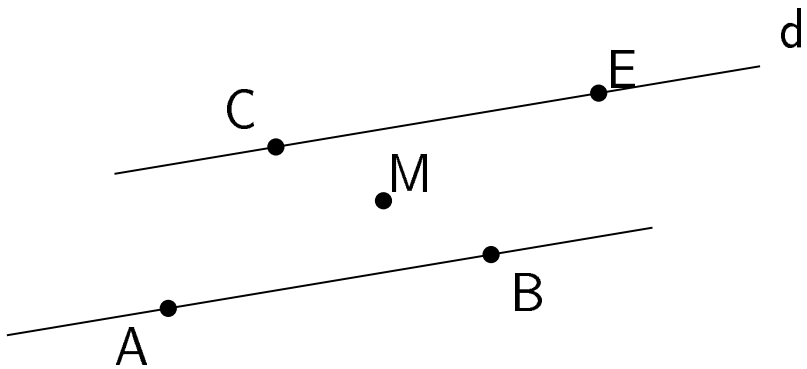
Exercise: Parallel Line Passing through a Point

Problem

Input A, B and C: 3 distinct points

Result d such that AB is parallel to d and C belongs to d

4: M$\leftarrow$ *MIDDLE* (B, C)
5: E$\leftarrow$ *DOUBLE_VECTOR* (A, M)
6: d$\leftarrow$ Line (C, E)
7: Write d
8: End



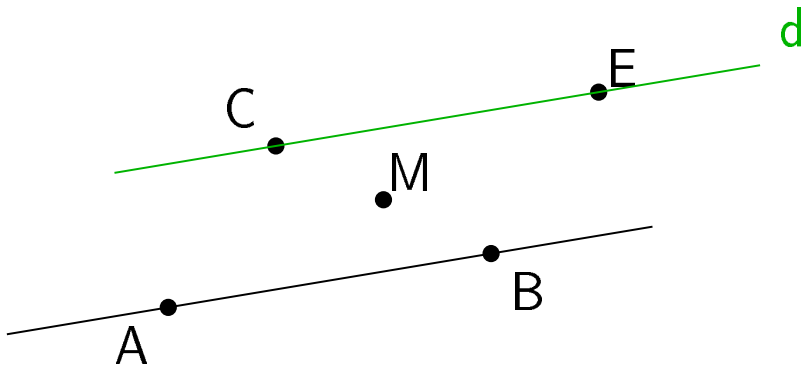
Exercise: Parallel Line Passing through a Point

Problem

Input A, B and C: 3 distinct points

Result d such that AB is parallel to d and C belongs to d

4: M$\leftarrow$ *MIDDLE* (B, C)
5: E$\leftarrow$ *DOUBLE_VECTOR* (A, M)
6: d$\leftarrow$ Line (C, E)
7: Write d
8: End



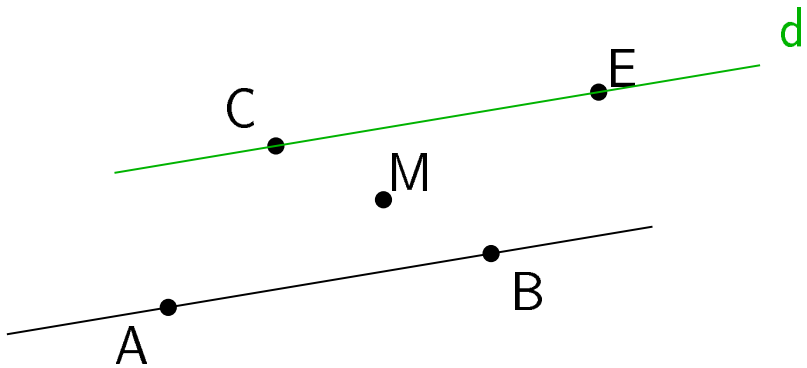
Exercise: Parallel Line Passing through a Point

Problem

Input A, B and C: 3 distinct points

Result d such that AB is parallel to d and C belongs to d

4: M$\leftarrow$ *MIDDLE* (B, C)
5: E$\leftarrow$ *DOUBLE_VECTOR* (A, M)
6: d$\leftarrow$ Line (C, E)
7: Write d
8: End



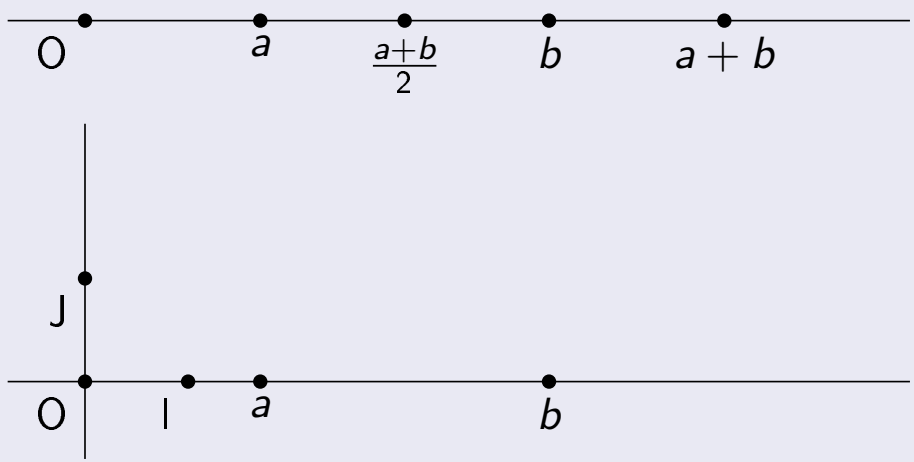
Compute with O, I, J Without Branching

Everything with coordinates computable from:

- initial coordinates and integers
- addition, subtraction, multiplication and division

Coordinates

Addition, multiplication, division

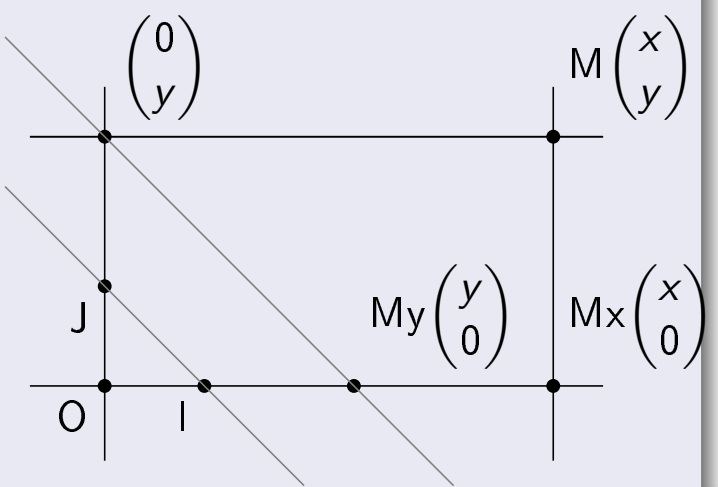


Compute with O, I, J Without Branching

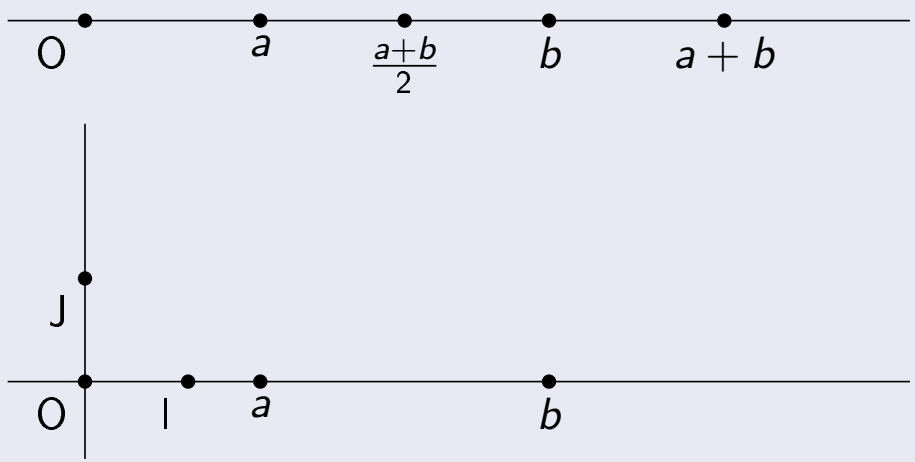
Everything with coordinates computable from:

- initial coordinates and integers
- addition, subtraction, multiplication and division

Coordinates



Addition, multiplication, division

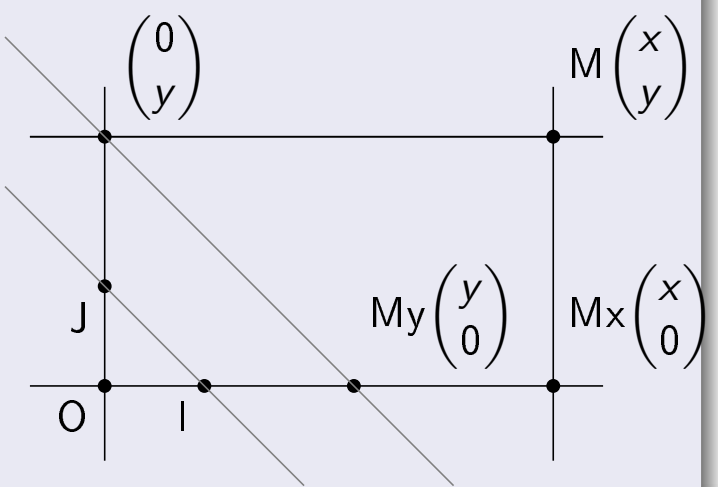


Compute with O, I, J Without Branching

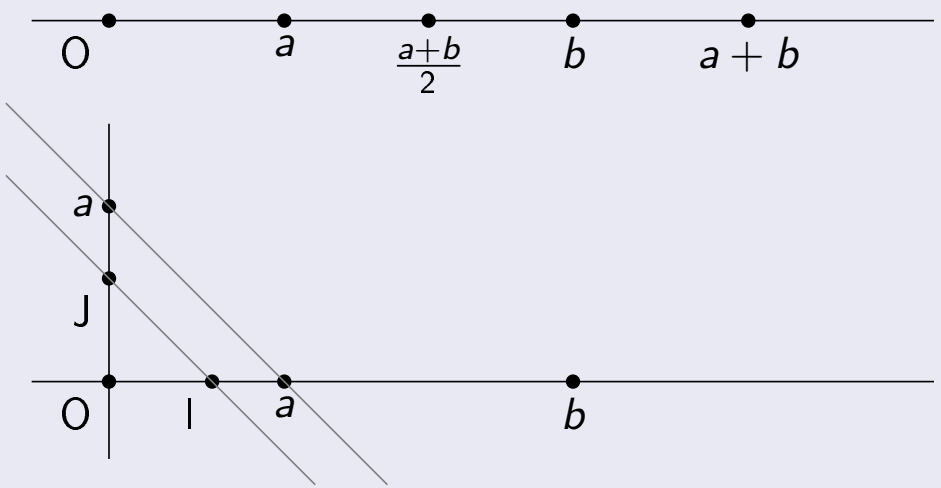
Everything with coordinates computable from:

- initial coordinates and integers
- addition, subtraction, multiplication and division

Coordinates



Addition, multiplication, division

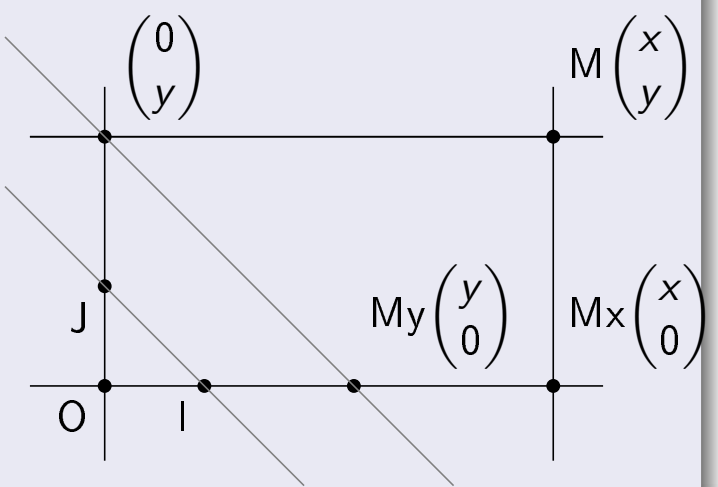


Compute with O, I, J Without Branching

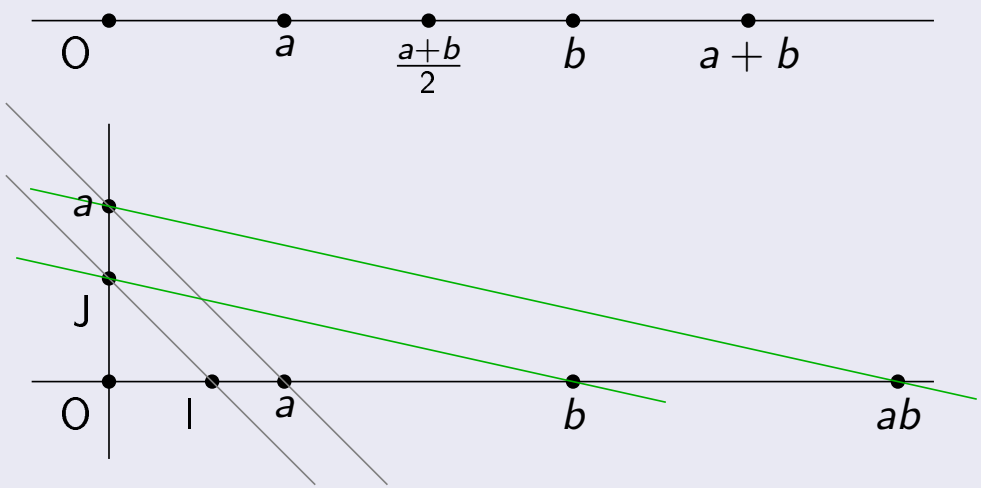
Everything with coordinates computable from:

- initial coordinates and integers
- addition, subtraction, multiplication and division

Coordinates



Addition, multiplication, division

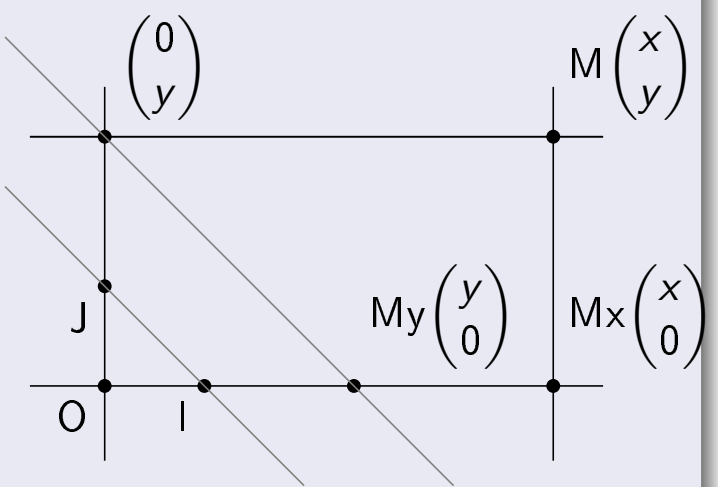


Compute with O, I, J Without Branching

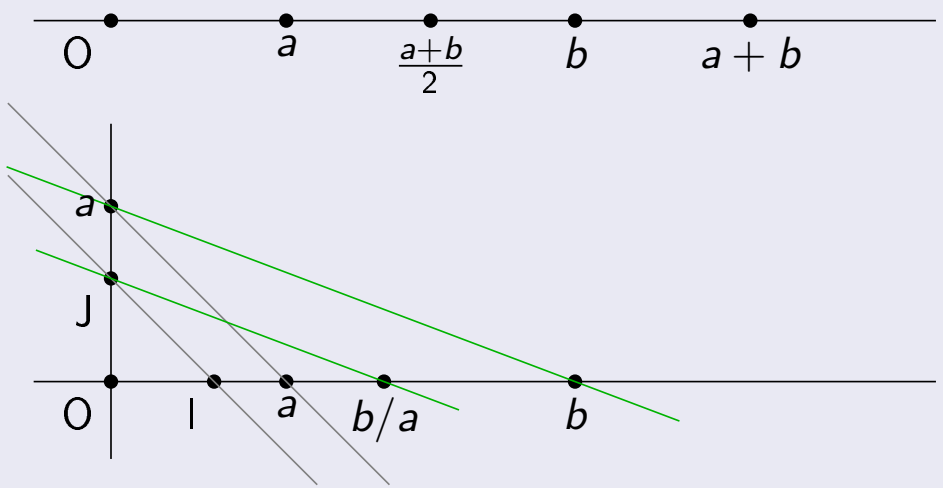
Everything with coordinates computable from:

- initial coordinates and integers
- addition, subtraction, multiplication and division

Coordinates



Addition, multiplication, division



Compute with O, I, J With Branching and $\mathcal{A} = \{O\}$

Turing Universal

2-counter automata simulation

More Generally

- As before, but piece-wise
- Frontiers are algebraic curves/surfaces with integer coefficients

$\sqrt{2}$ cannot be generated

Spatial / Static aspect

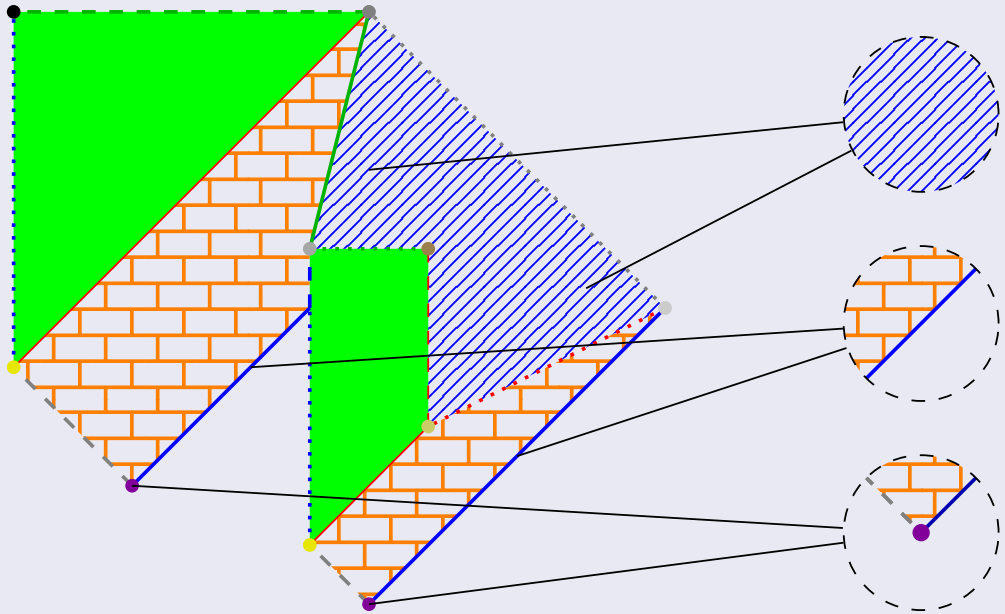
Each Position (in $\mathbb{R}^d$)

- a state
  $\rightsquigarrow$ a color

- Finite number of colors

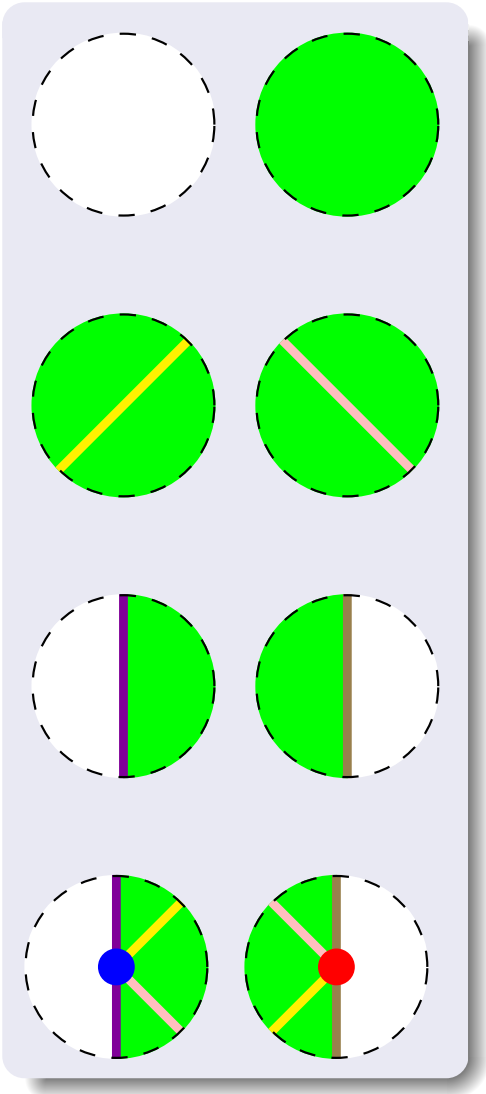
Uniformity

- same *rules* everywhere
- same color
  $\Rightarrow$ same neighborhood



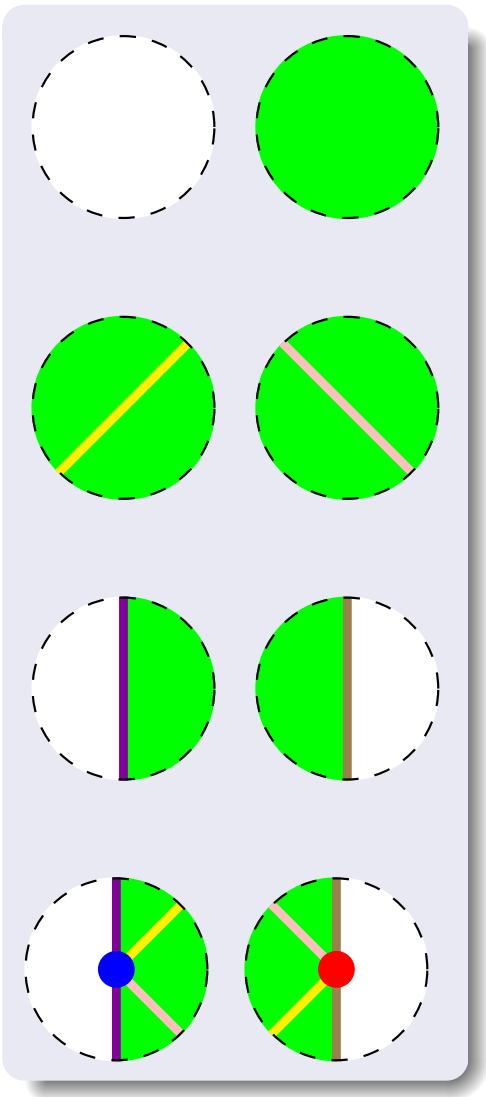
(a) Colored space (b) Local constraints

Exercice



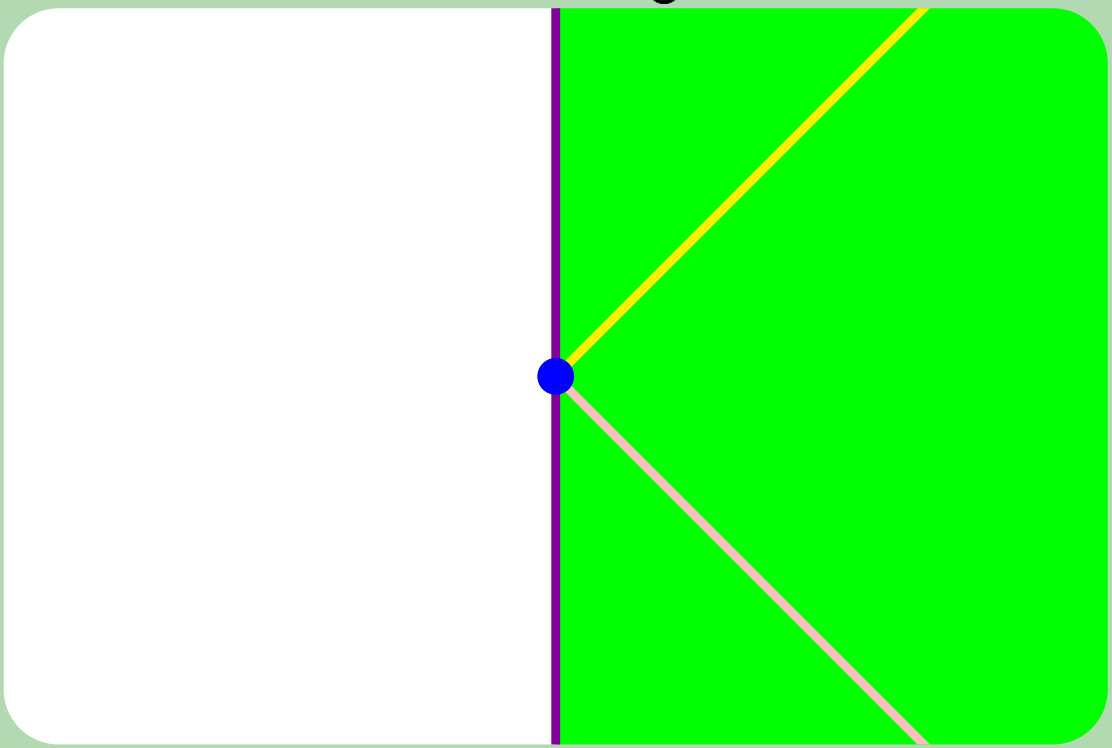
What Can be Generated?

Exercice

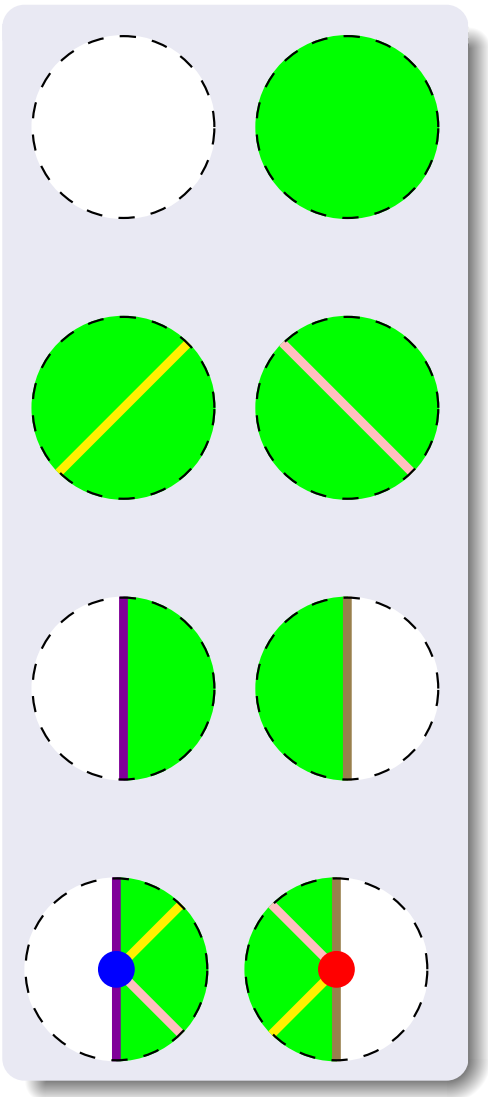


What Can be Generated?

- Trivial extension of neighborhood

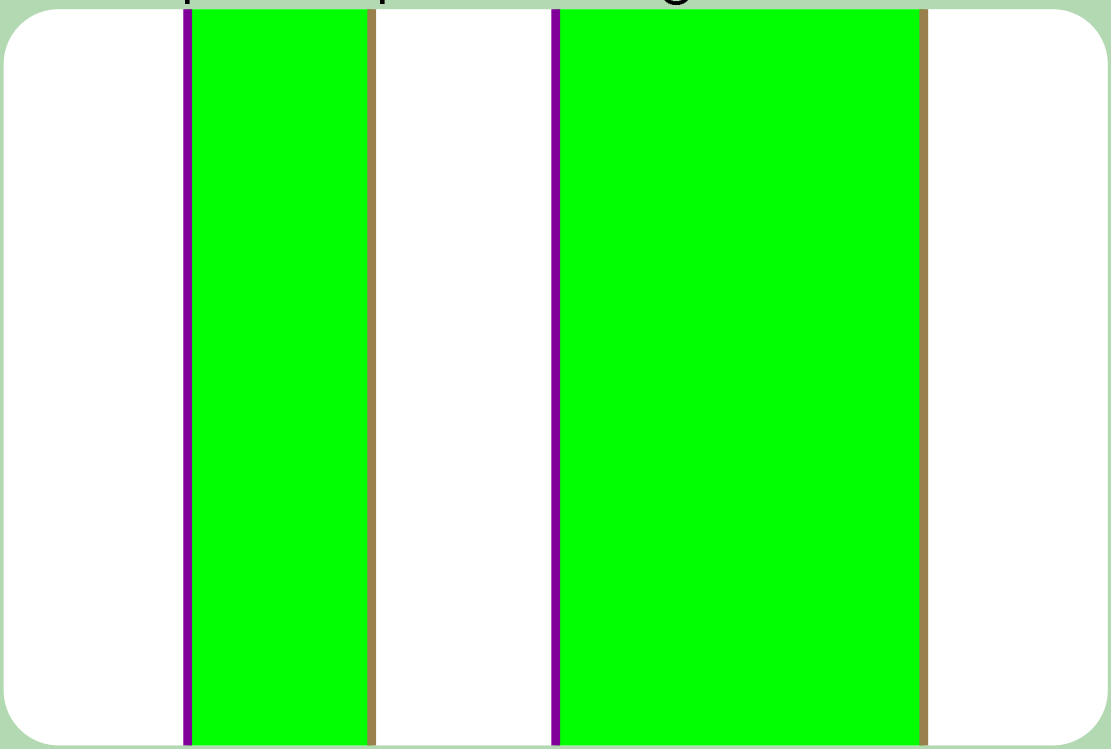


Exercice

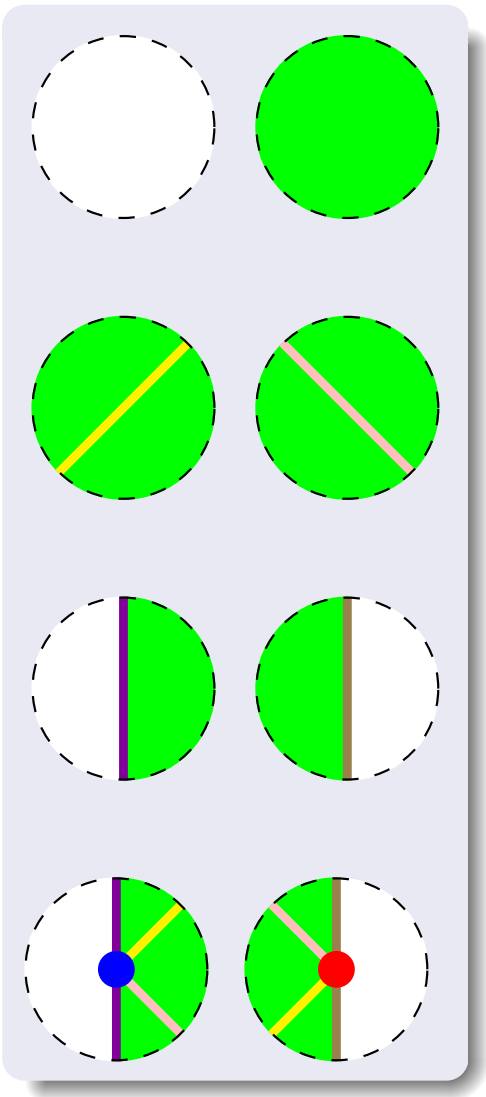


What Can be Generated?

- Trivial extension of neighborhood
- Simple composition neighborhood

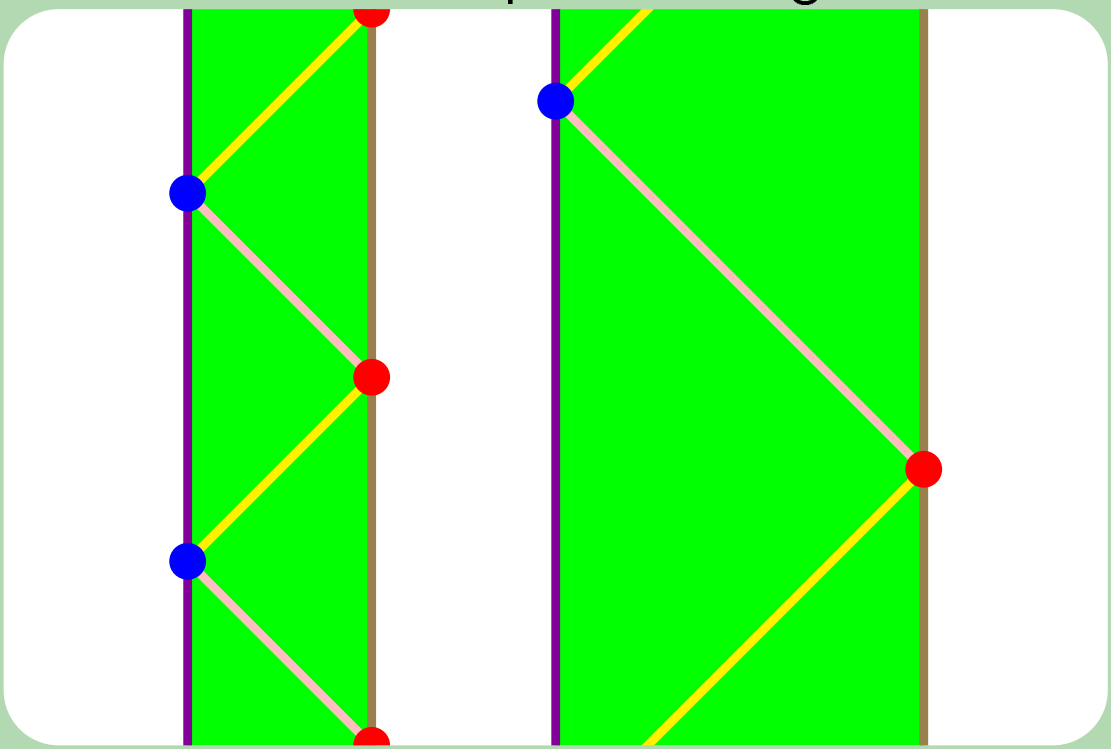


Exercice

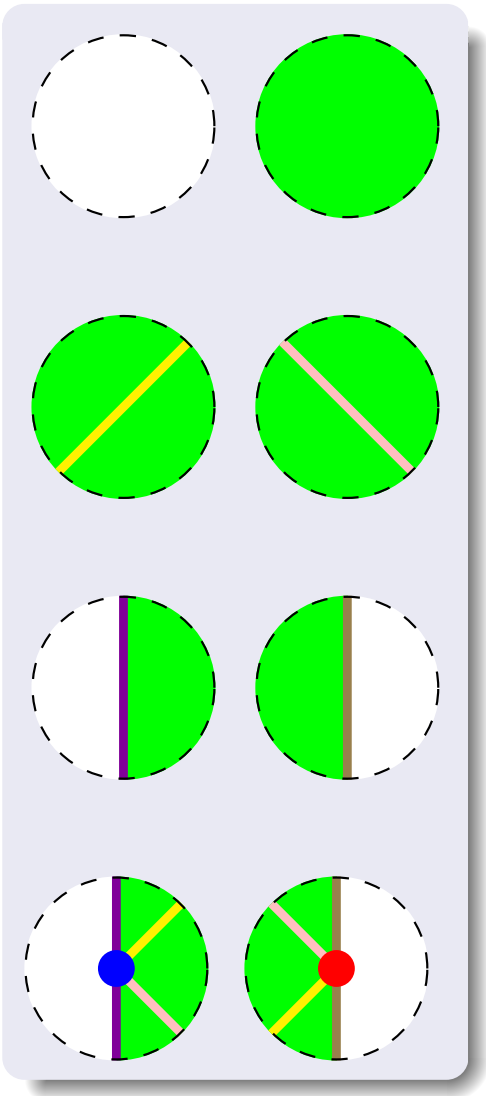


What Can be Generated?

- Trivial extension of neighborhood
- Simple composition neighborhood
- Constrained composition neighborhood

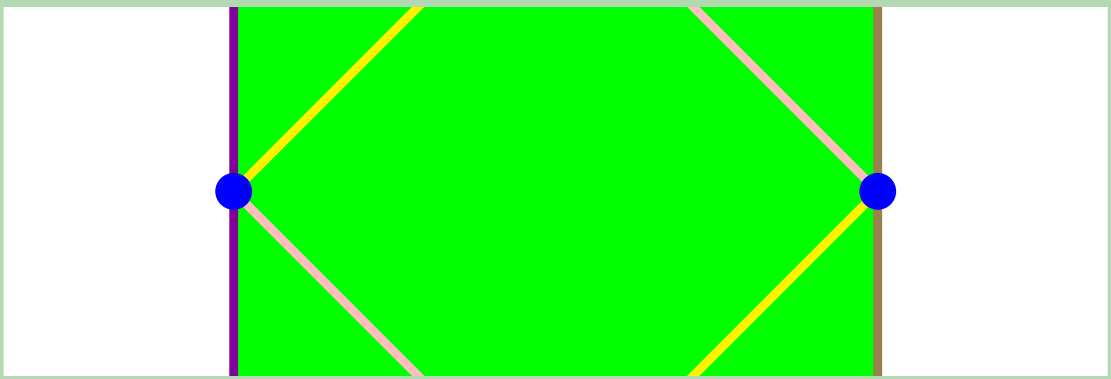


Exercice



What Can be Generated?

- Trivial extension of neighborhood
- Simple composition neighborhood
- Constrained composition neighborhood
- Globally impossible



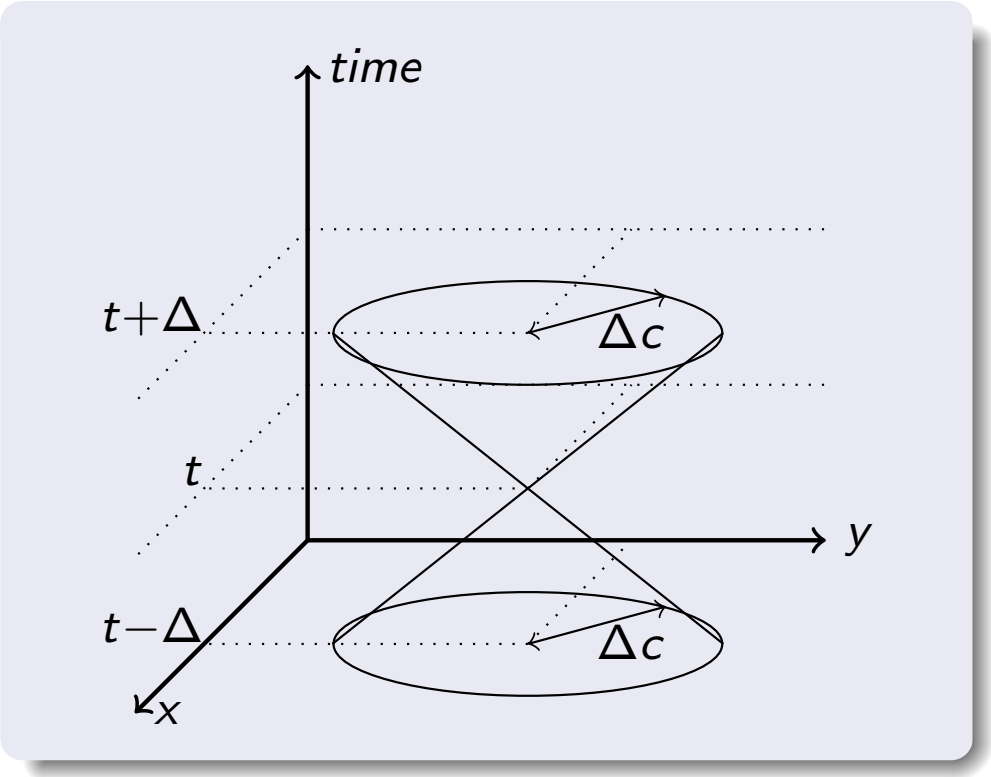
Temporal / Dynamic Aspects

Speed Limit (c)

- information propagation

Space-time Cone

- dependence
- influence



Temporal / Dynamic Aspects

Speed Limit (c)

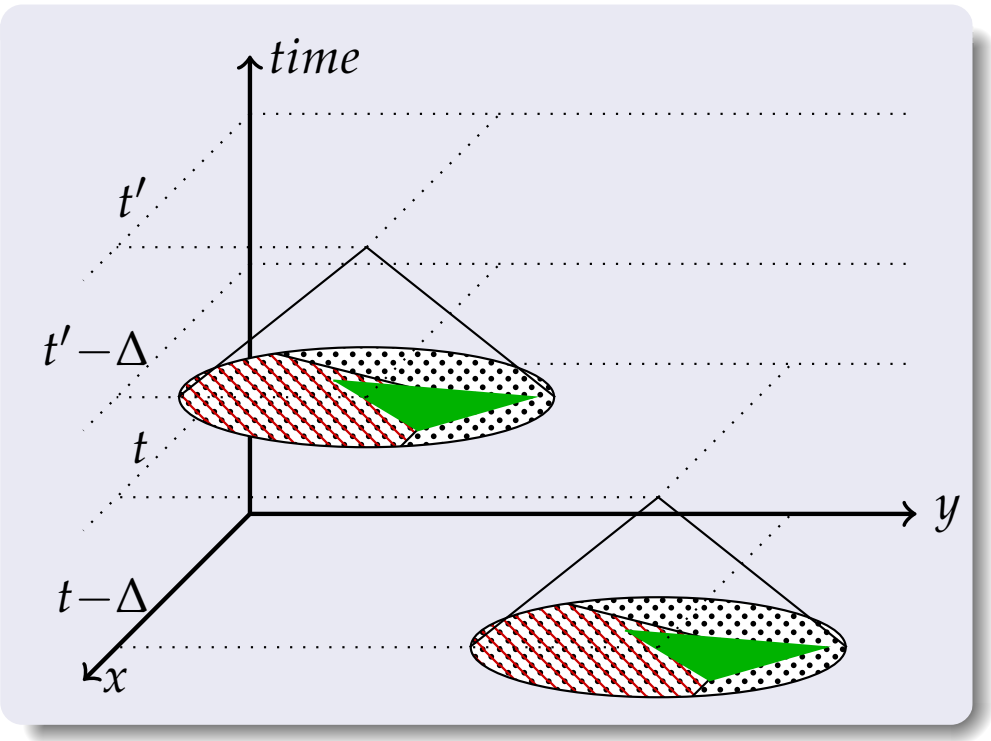
- information propagation

Space-time Cone

- dependence
- influence

Uniformity (past cones)

- Same bases
  $\Rightarrow$ Same above in cone



Temporal / Dynamic Aspects

Speed Limit (c)

- information propagation

Space-time Cone

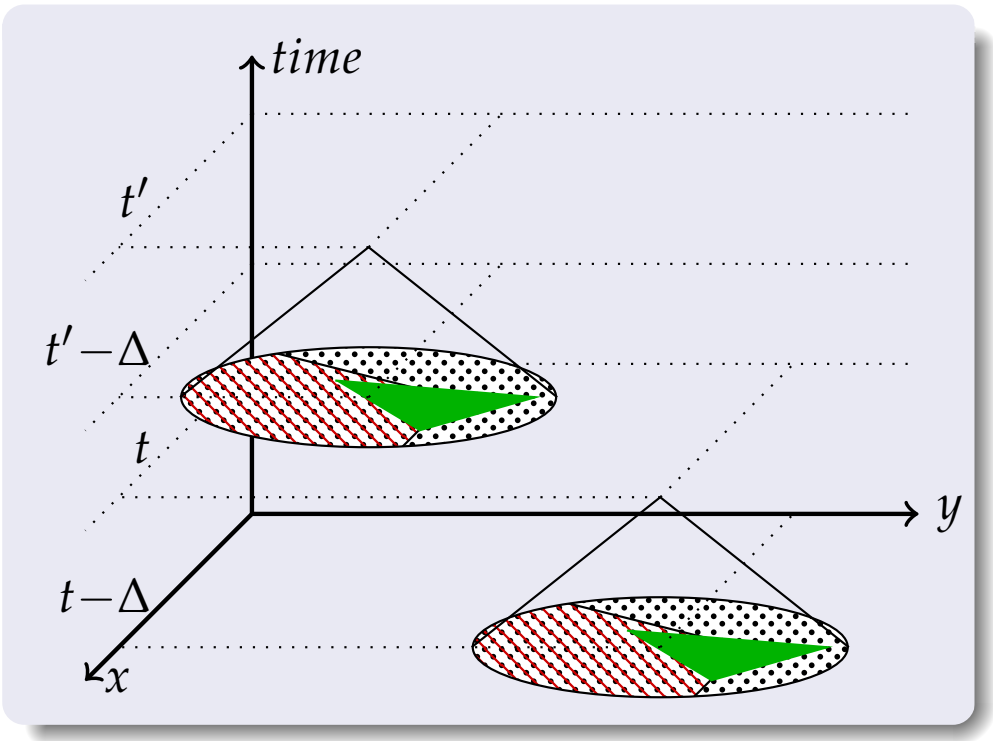
- dependence
- influence

Uniformity (past cones)

- Same bases
  $\Rightarrow$ Same above in cone

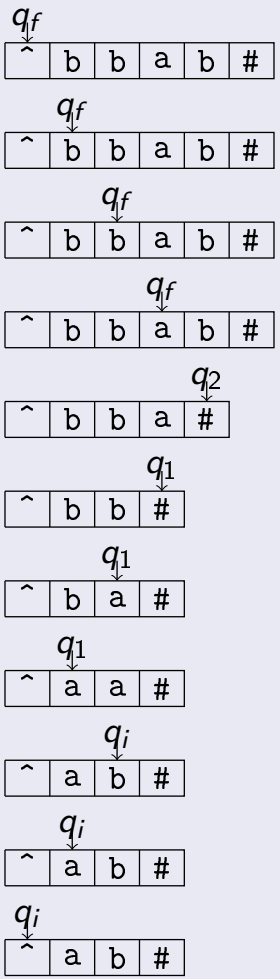
Reversibility

- Same backwards

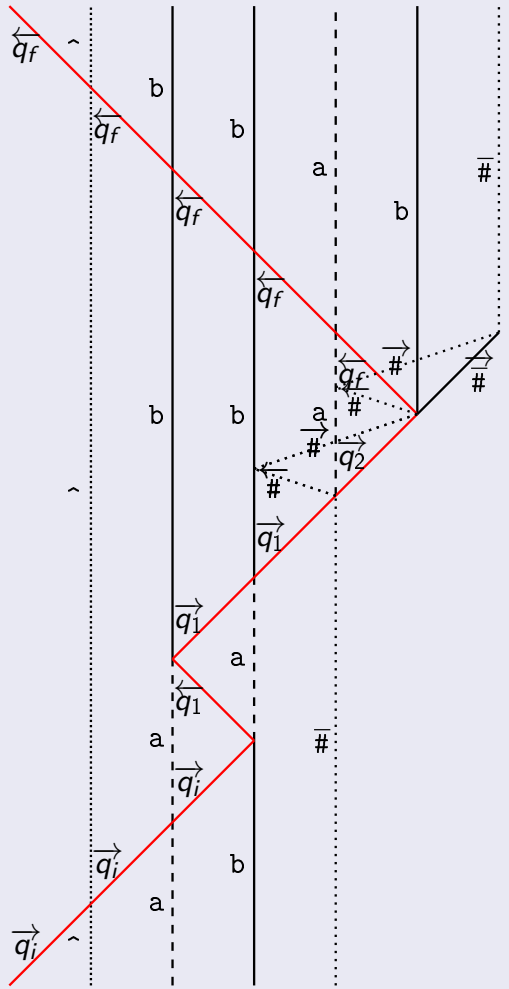


(Turing) Computing

(Reversible) Turing Machine

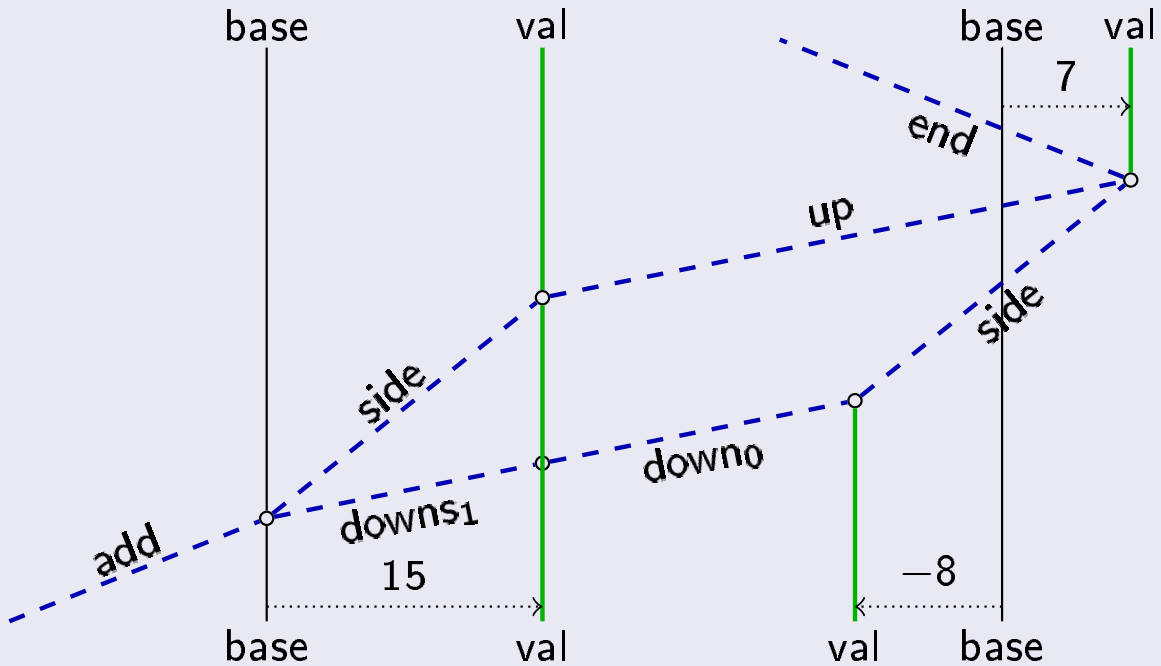


Simulation



Computing with Real Numbers

Addition (value ≈ distance)



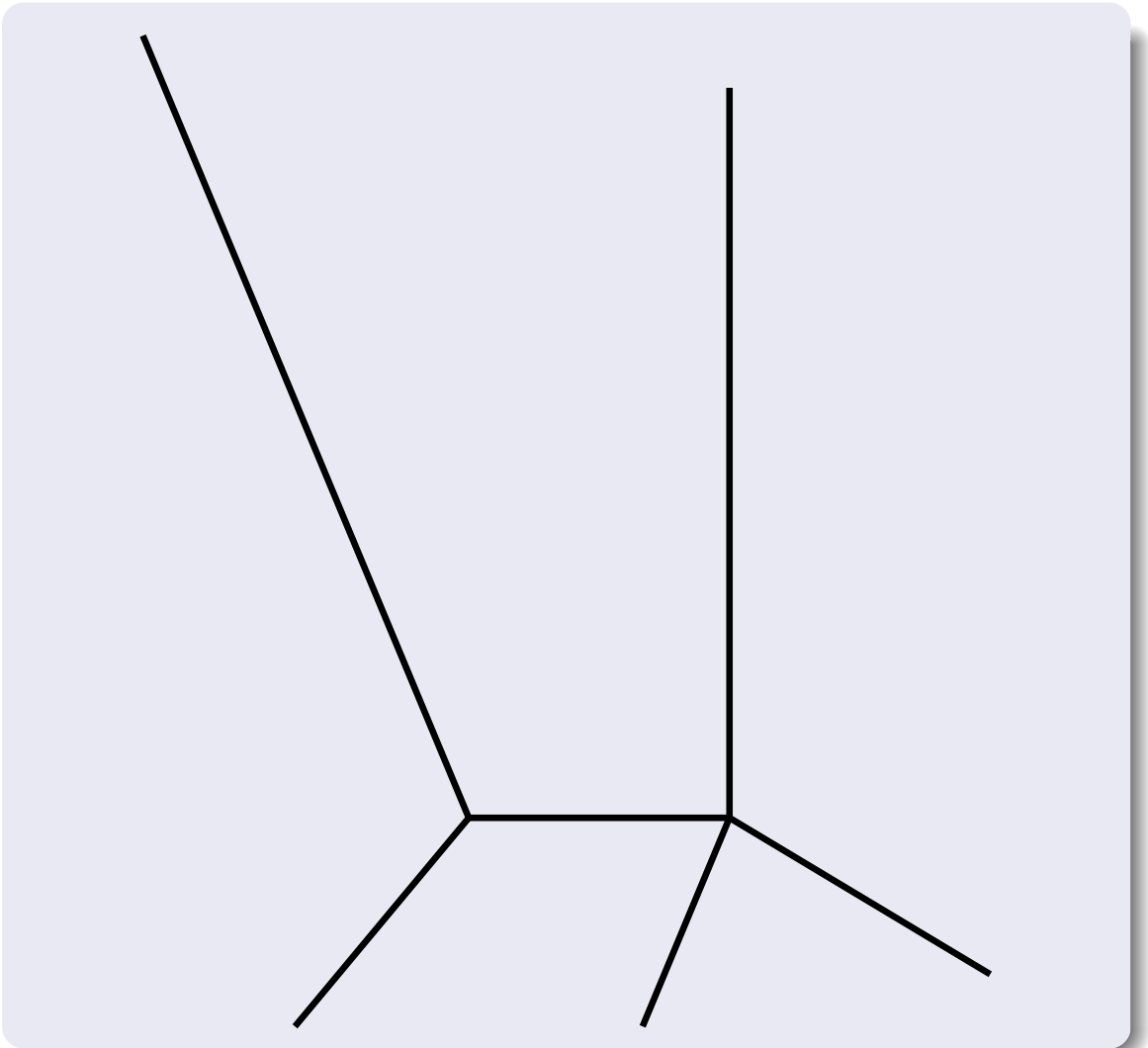
(Reversible linear-BSS)

- addition, subtraction
- multiplication by a constant
- test de sign, branch

Hybride Dynamical System

Partition of Space

- Polyhedral regions



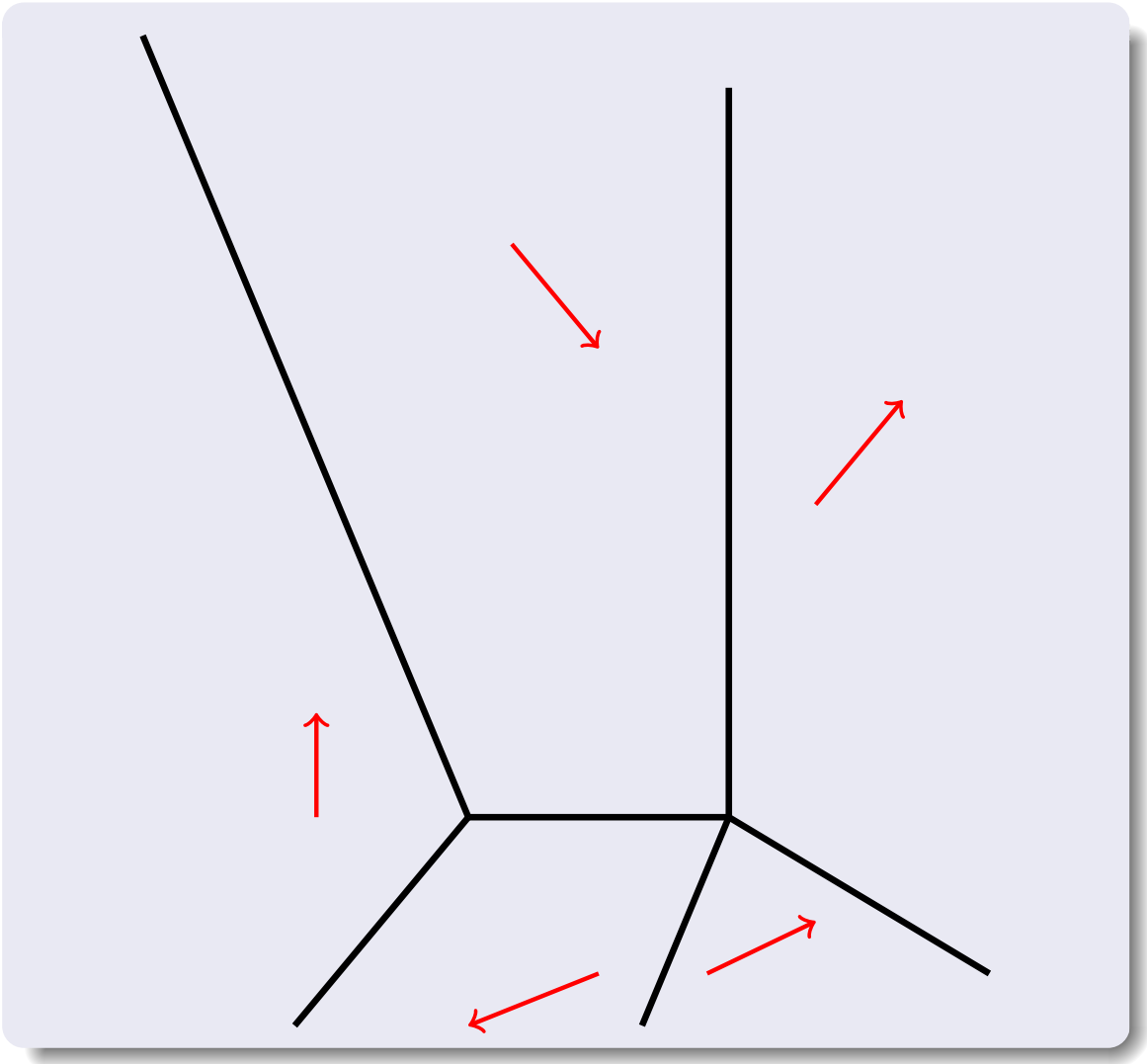
Hybride Dynamical System

Partition of Space

- Polyhedral regions

Each Region

- Constant derivative



Hybride Dynamical System

Partition of Space

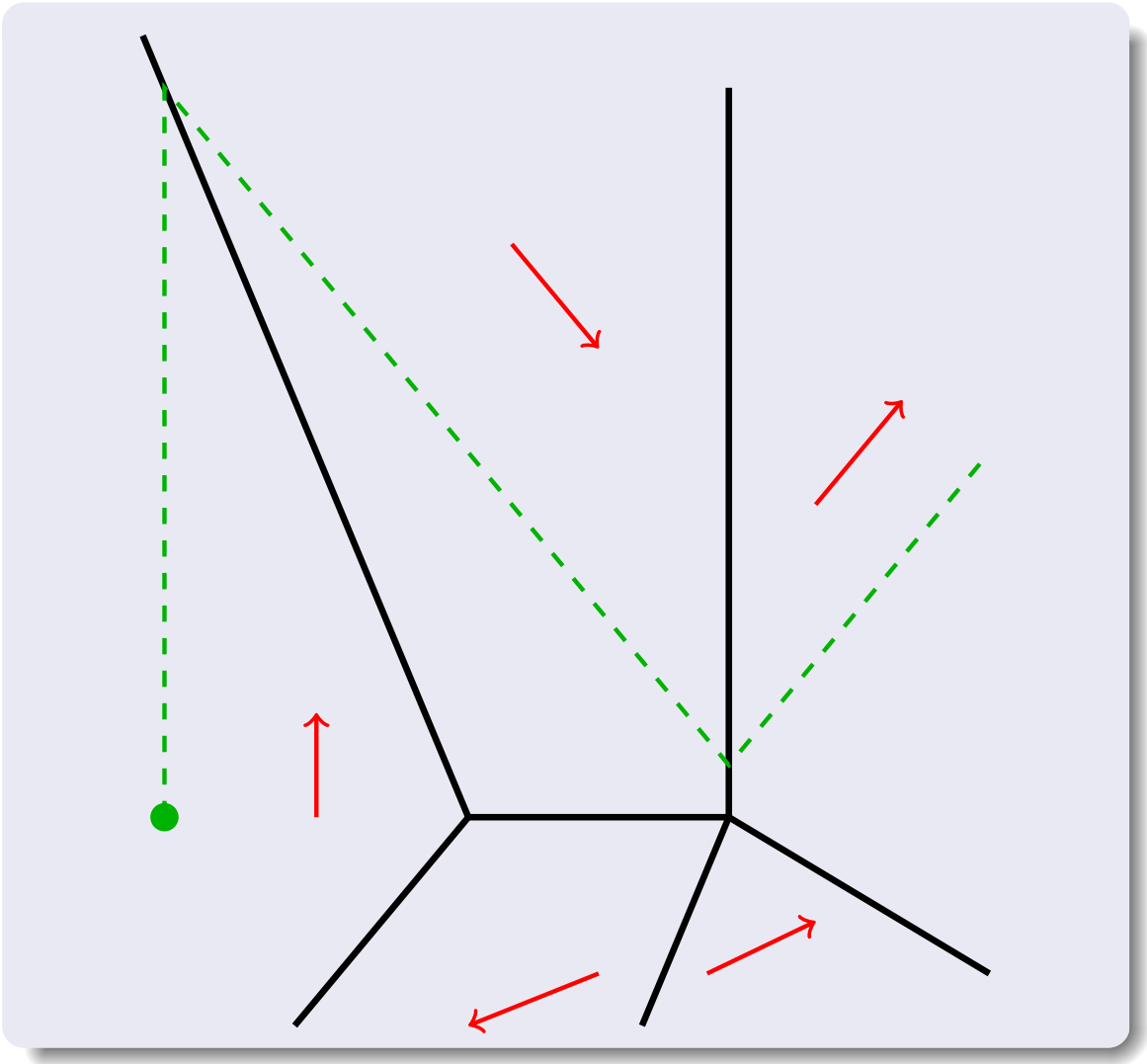
- Polyhedral regions

Each Region

- Constant derivative

Dynamics

- Follow the trajectory from a point



Hybride Dynamical System

Partition of Space

- Polyhedral regions

Each Region

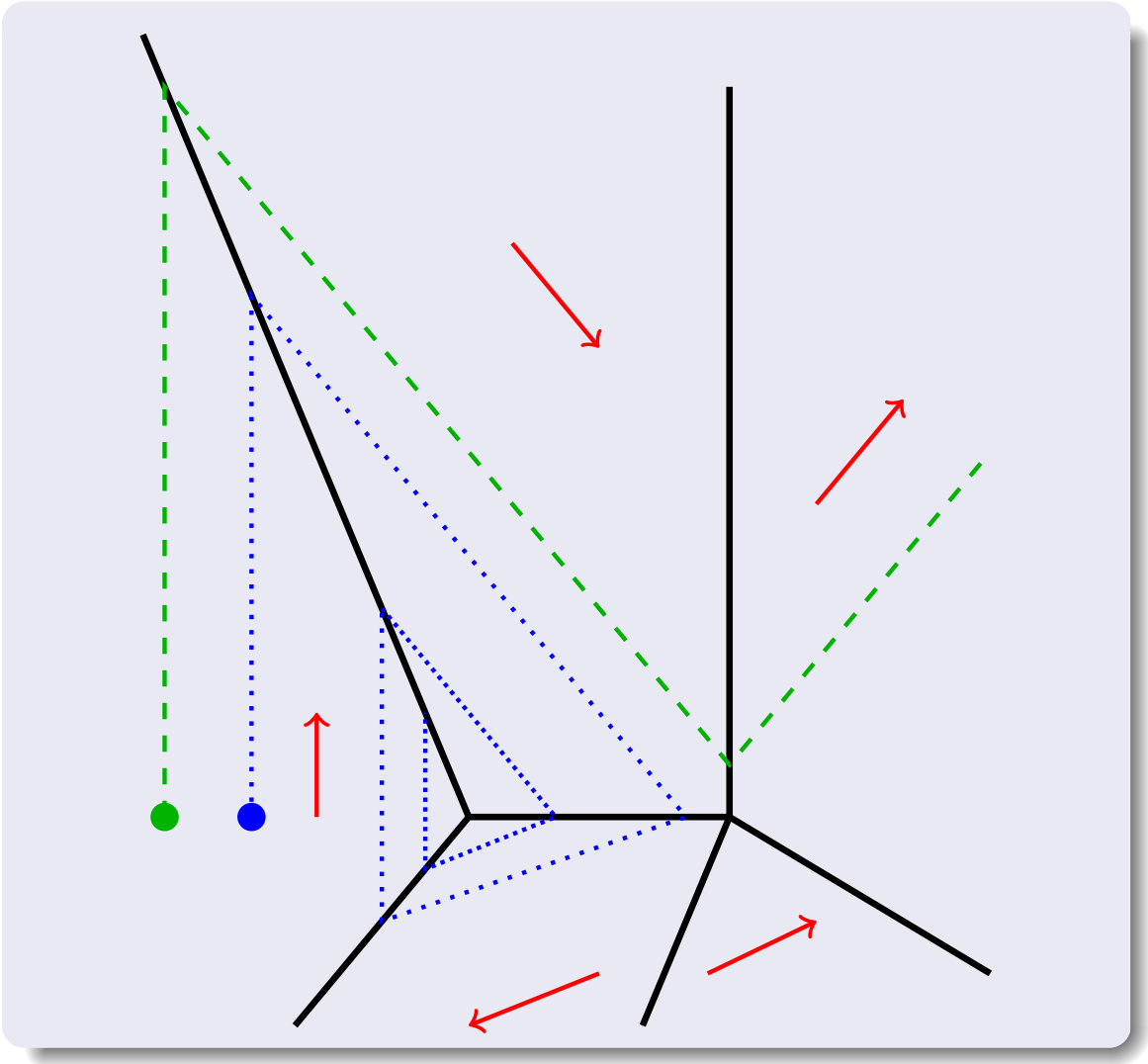
- Constant derivative

Dynamics

- Follow the trajectory from a point

Accumulation

- *Zeno effect*

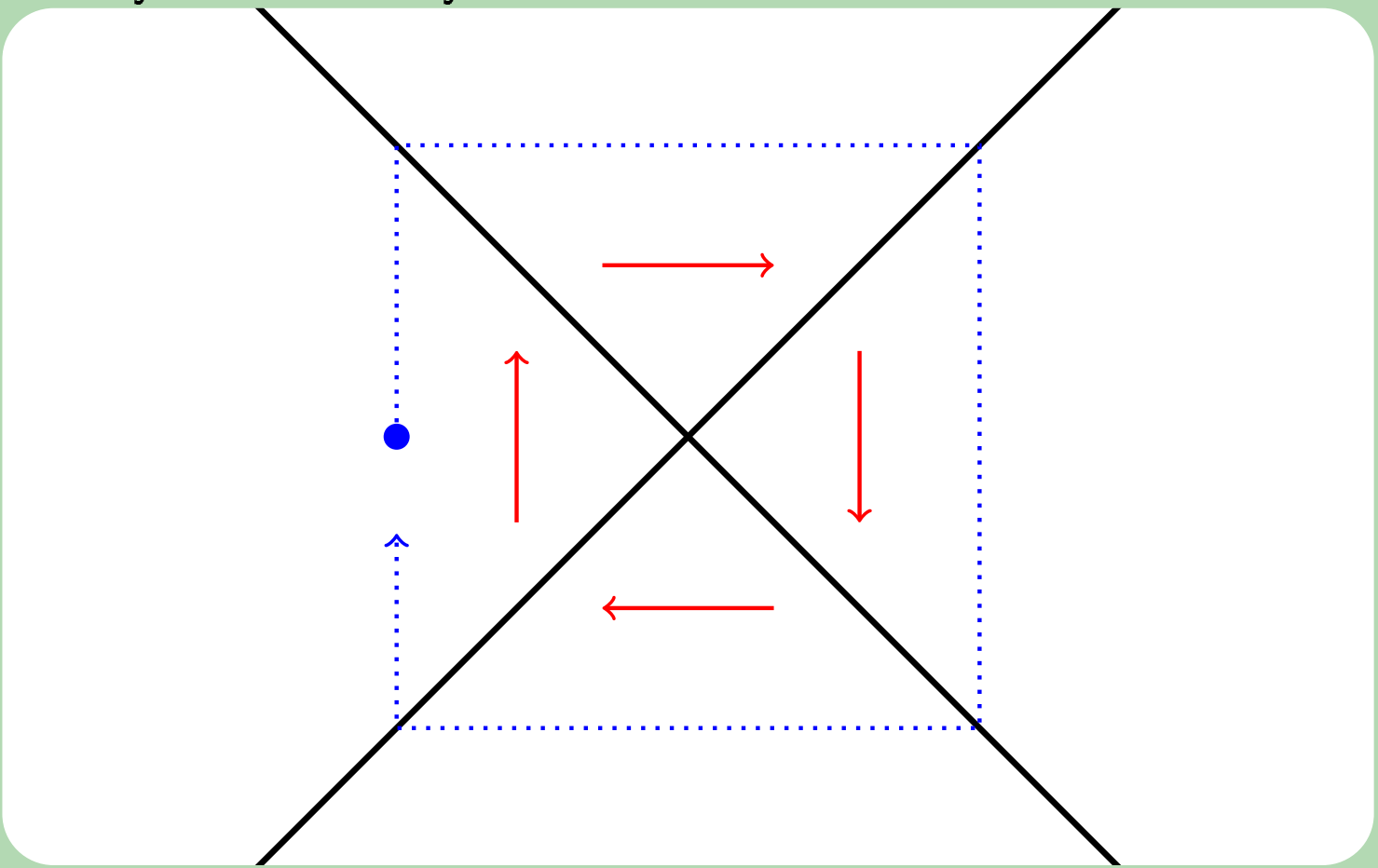


Exercice

1. Can you make a cycle?

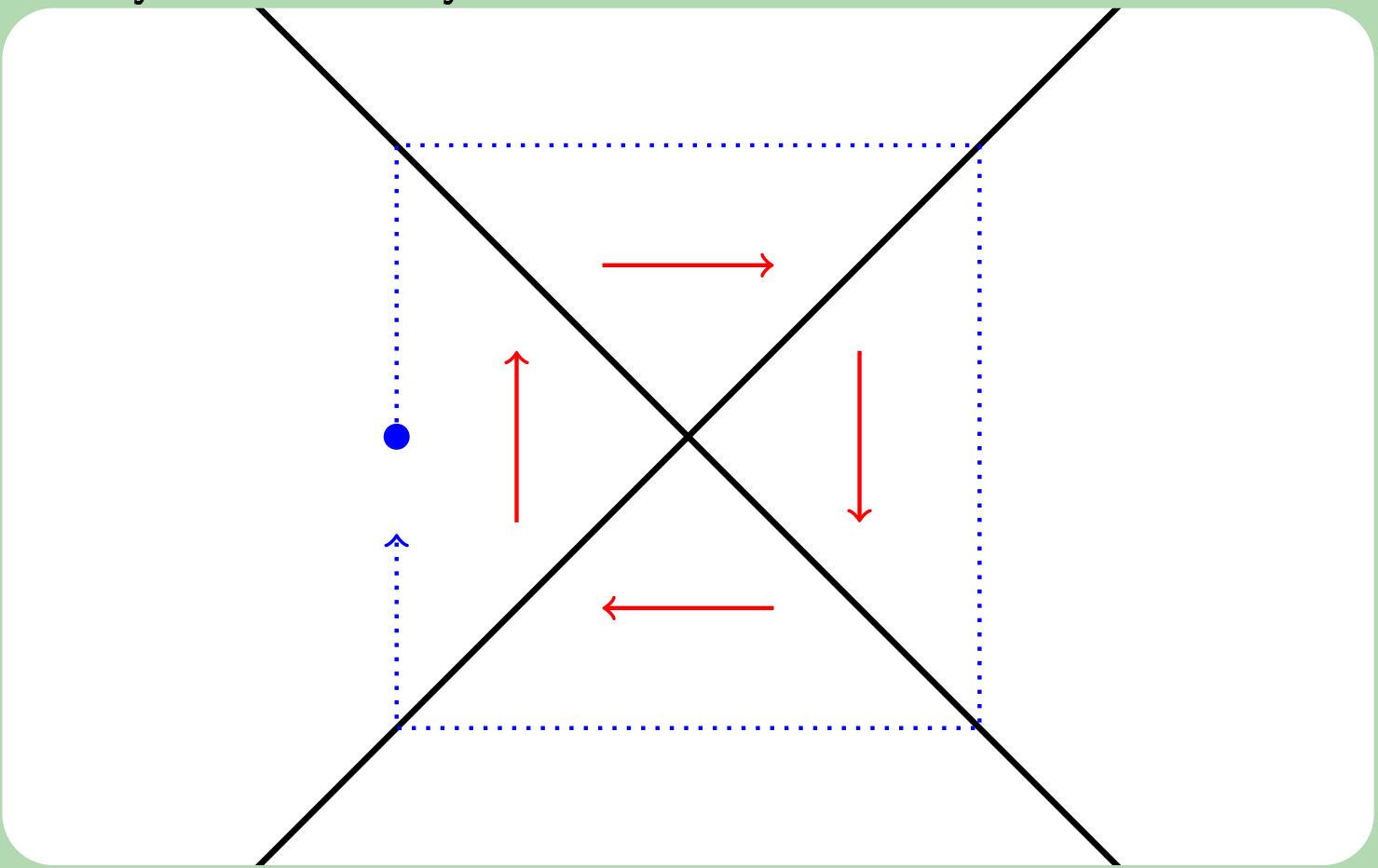
Exercice

1. Can you make a cycle?



Exercice

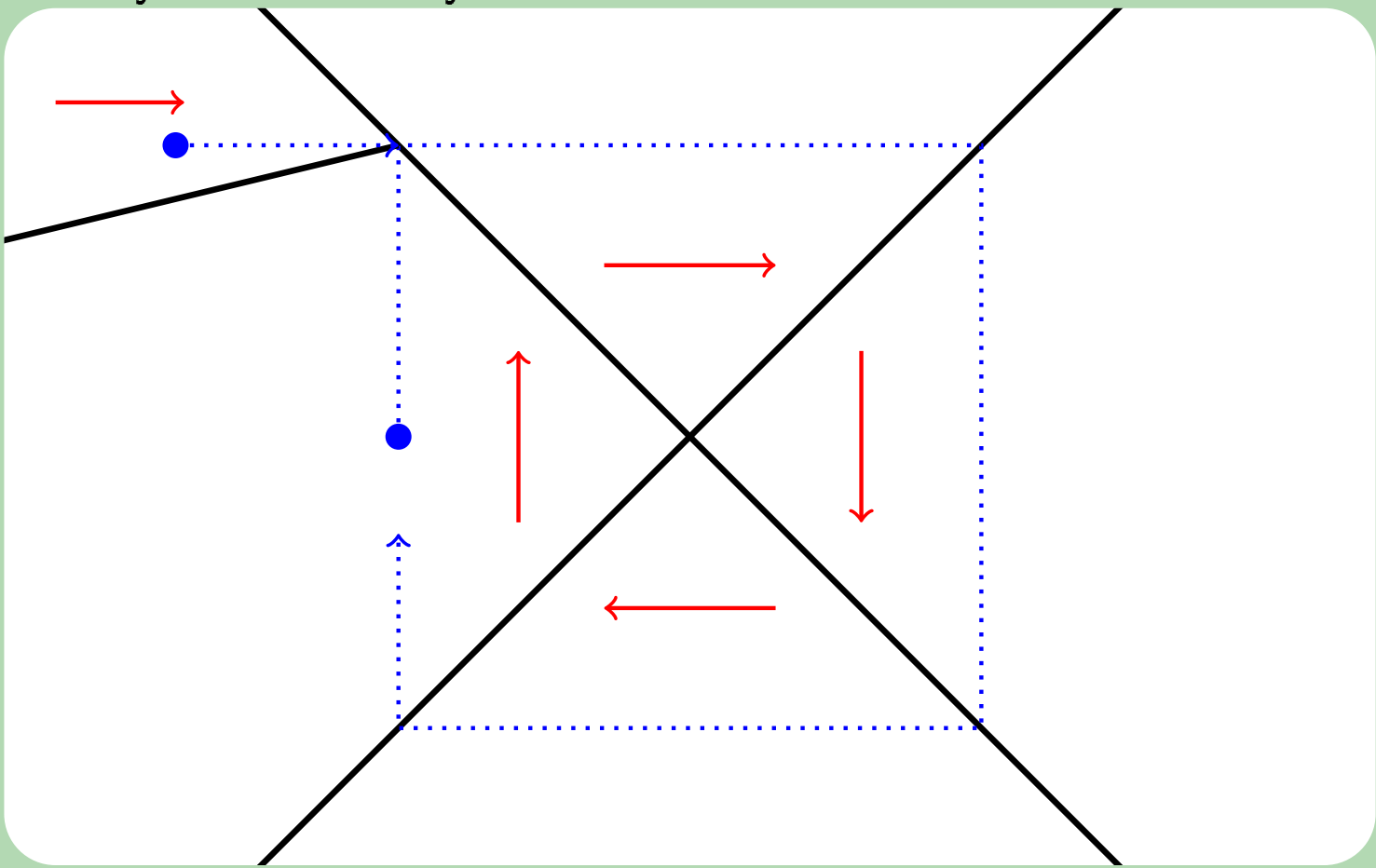
1. Can you make a cycle?



2. Can you make a point enter a cycle?

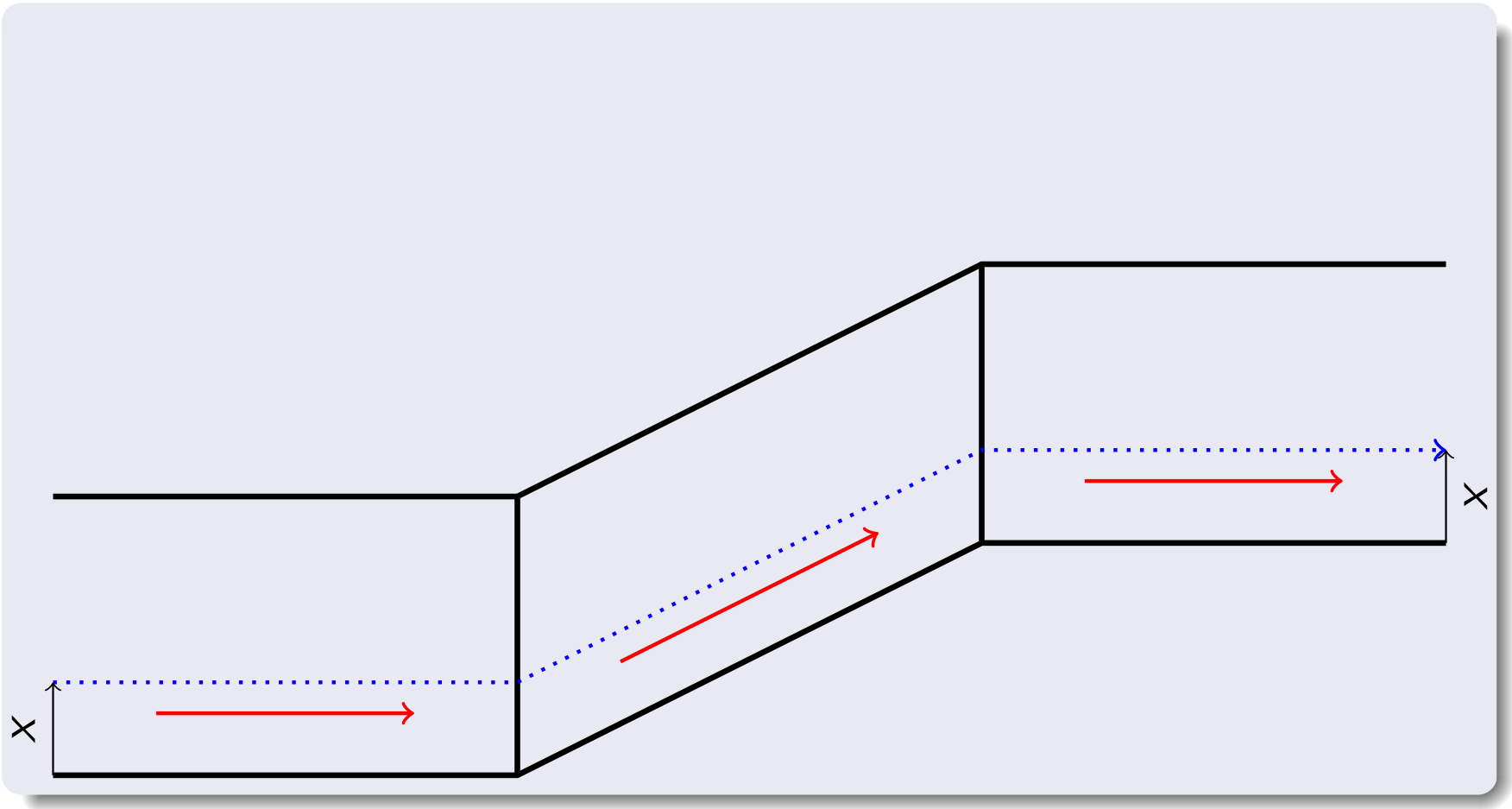
Exercice

1. Can you make a cycle?

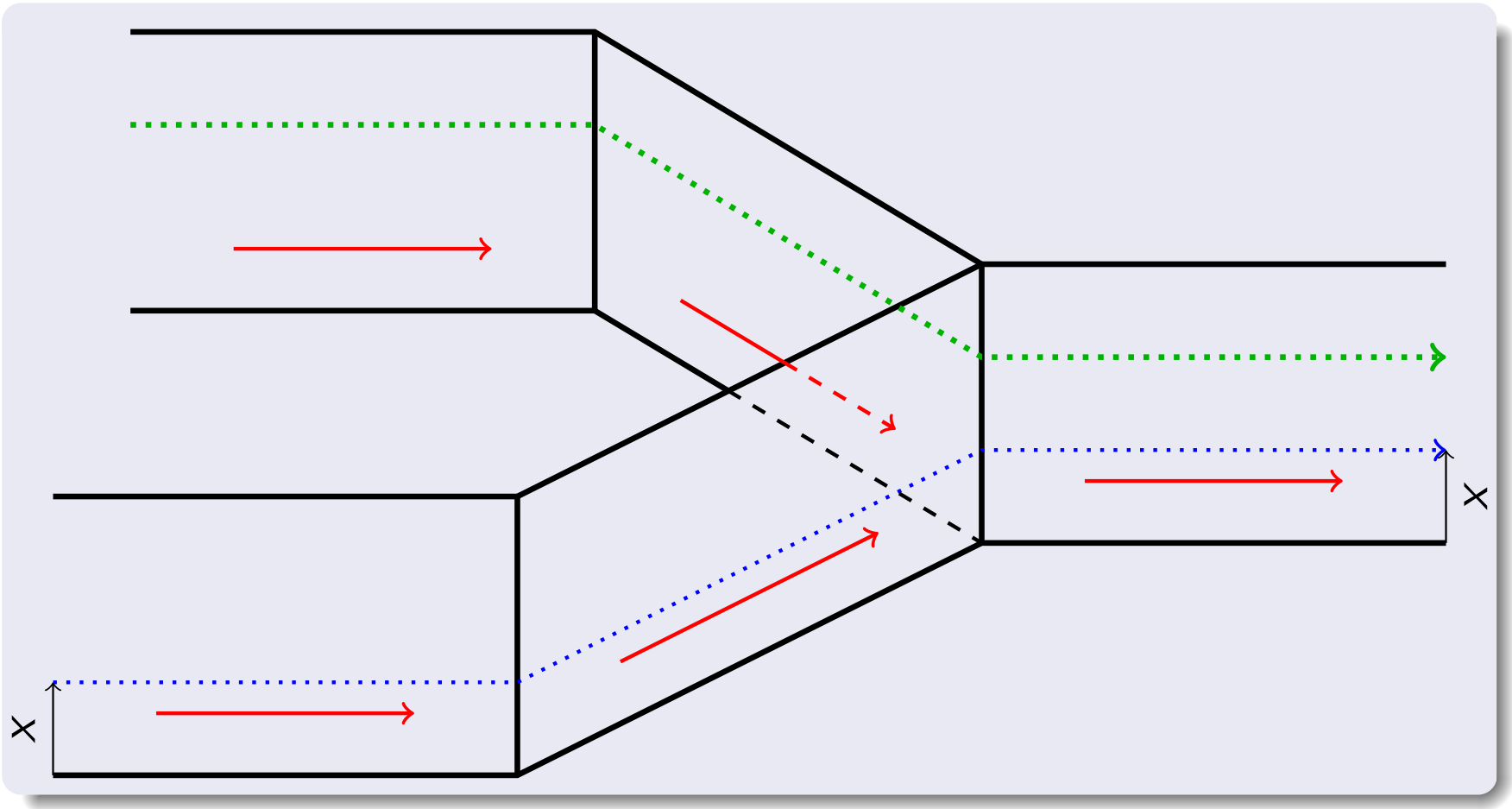


2. Can you make a point enter a cycle?

Wires for Analog Information



Wires for Analog Information



In 3D wires can be merged

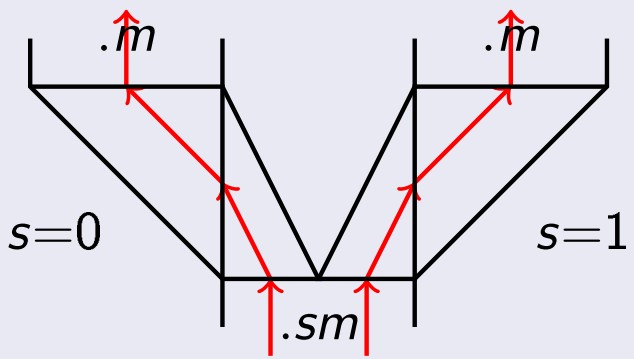
(Turing)-Computation in Dimension 4

Encoding of a TM Configuration

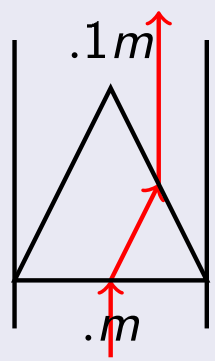
- tape (m_l, s, m_r): $l = (0.m_l)_{|\Sigma|}$ and $r = (0.sm_r)_{|\Sigma|}$
- state: height / level

Primitives for $\Sigma = \{0, 1\}$

Branching on letter



Stack 1



Dimension 3

- *managing* the other side of the tape (m_l)

Dimension 4

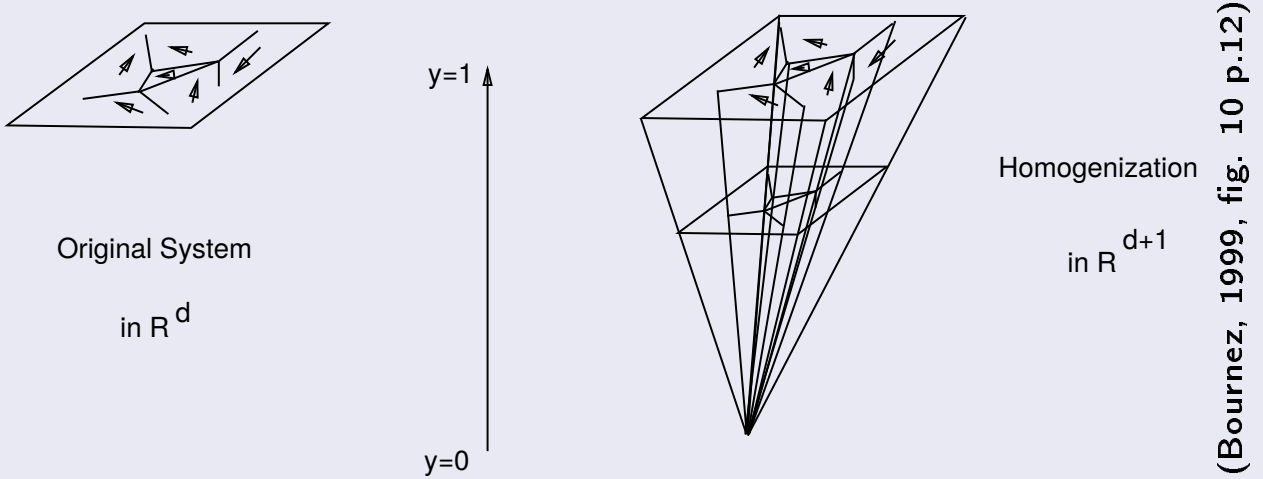
- *connection* between levels
- *fusion* of dimension 2 paths

Climbing up the Arithmetical Hierarchy

Hierarchy

- Σ_0: problems expressible as $\phi(x_1, x_2, ..., x_n)$ where ϕ is a (total) recursive predicate
- Σ_d: problems expressible as $\exists x_1 \forall x_2 \exists x_3 \ldots \phi(x_1, x_2, ..., x_n)$ with d alternating quantifiers

Zeno Effect Use



$d + 2$ dimensions to decide Σ_d

Cellular Automata: signal use

Firing Squad Synchronization (Goto, 1966)

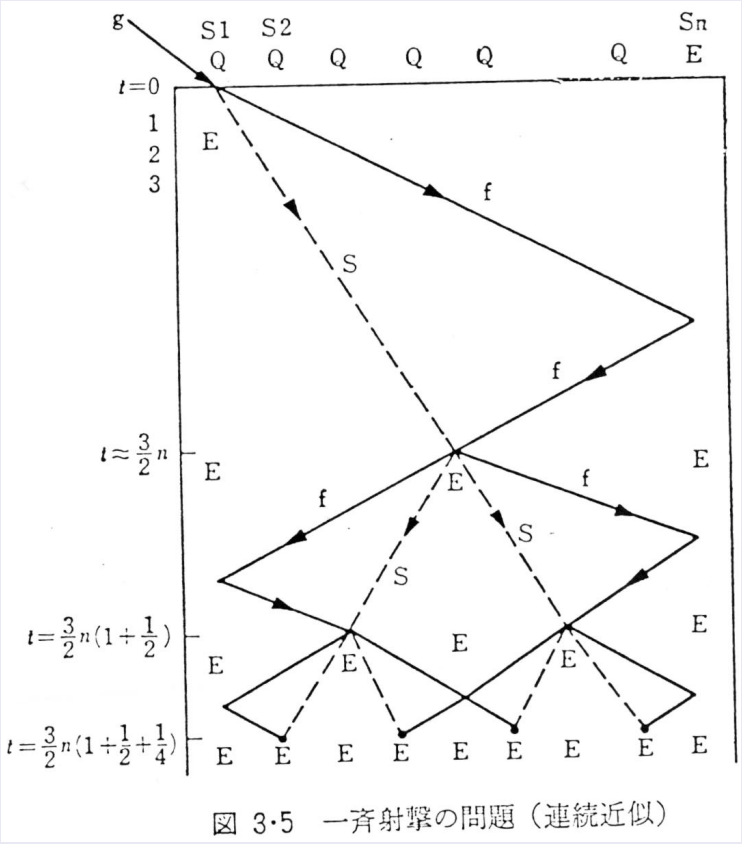


図 3·5 一斉射撃の問題（連続近似）

G	s_1	s_2	s_3	s_4	s_5	s_6
g	Q	Q	Q	Q	Q	E
t=0	f's'Efs	Q	Q	Q	Q	E
1	E	Q2f	Q	Q	Q	E
2	E	Q1	Qf	Q	Q	E
3	E	Q&	Q	Qf	Q	E
4	E	Q	Q2	Q	Qf	E
5	E	Q	Q1	Q	Q	f'Ef
6	E	Q	QS	Q	f'Q	E
7	E	Q	Q	a'Q	Q	E
8	E	Q	f'S'ESf	f's'Esf	Q	E
9	E	f'2Q	E	E	Q2f	E
10	f'Ef	1Q	E	E	Q1	f'Ef
11	E	f'S'ESf	E	E	f's'Esf	E
12	a'Ea	E	a'Ea	a'Ea	E	a'Ea
13	F	F	F	F	F	F

図 3·6 一斉射撃解（n=6）

- Cellular Automata: UC[NC] 2011 Tutorial by N. Ollinger

CA: Conception with signals

Fischer (1965)

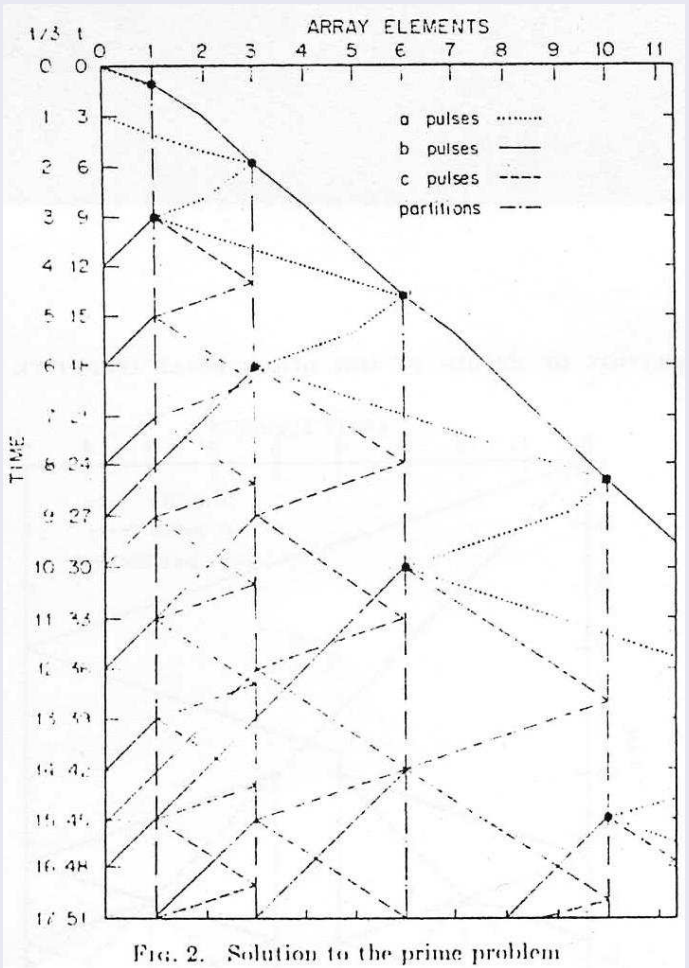
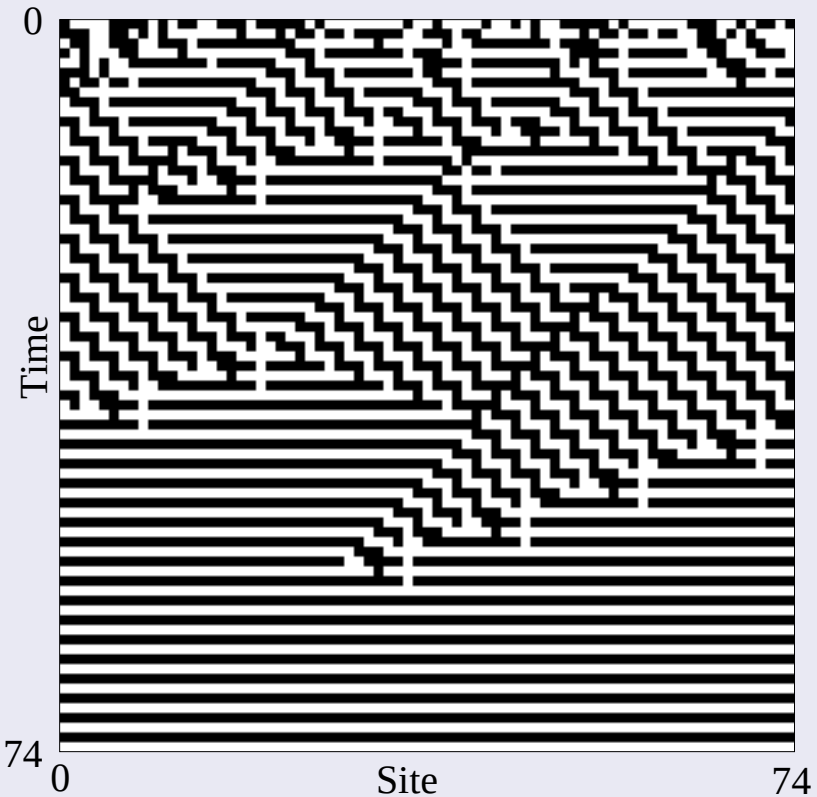


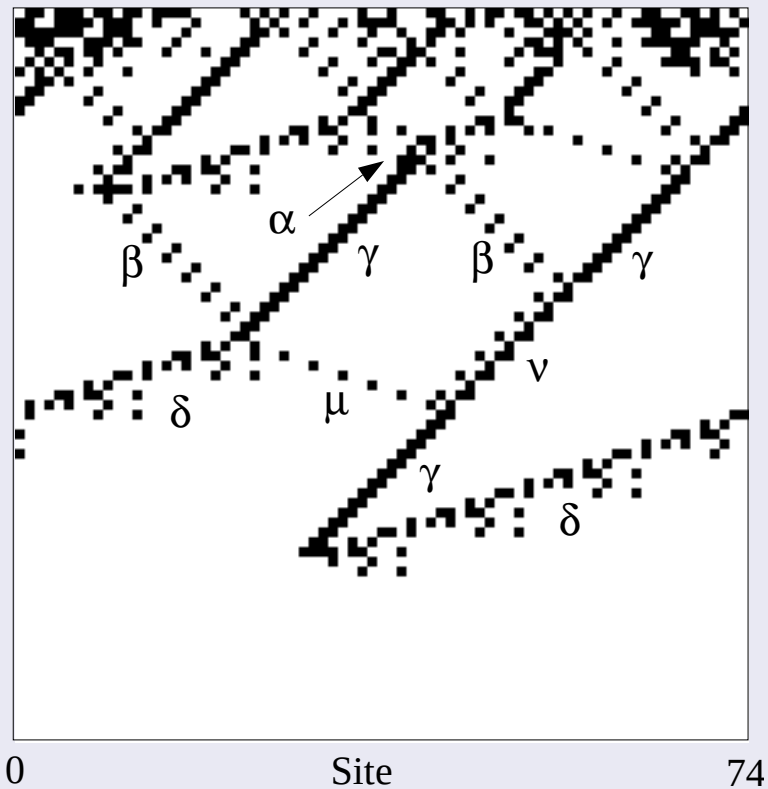
Fig. 2. Solution to the prime problem

CA: Analyzing with Signals

Das et al. (1995)

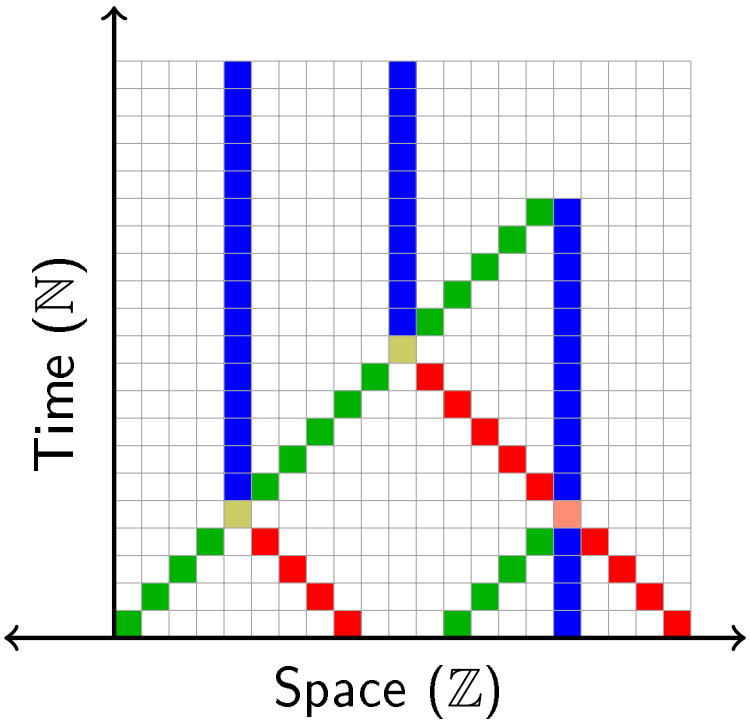


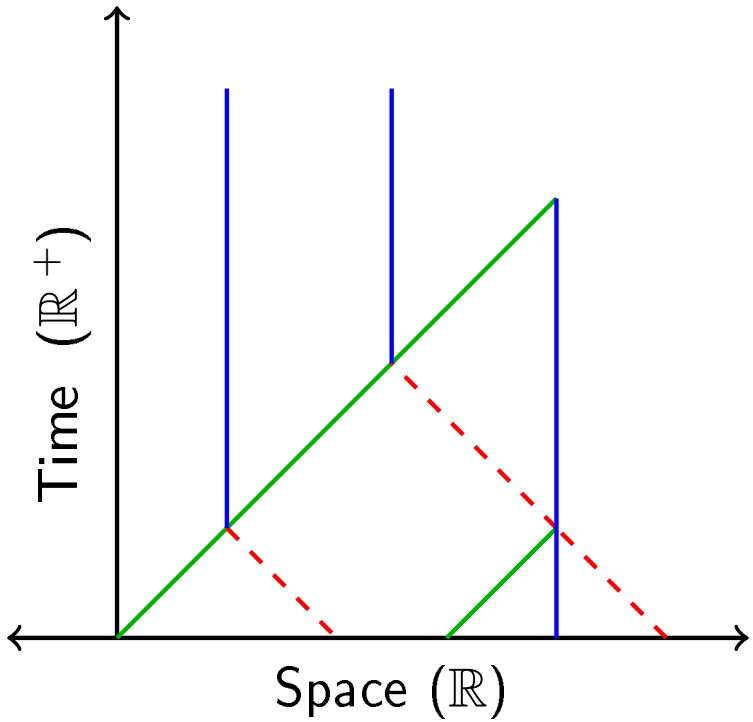
(a) Space-time diagram.



(b) Filtered space-time diagram.

Signals





- Signal (meta-signal)
- Collision (rule)

Vocabulary and Example: Find the Middle

Meta-signals (speed)

M (0)

Collision rules

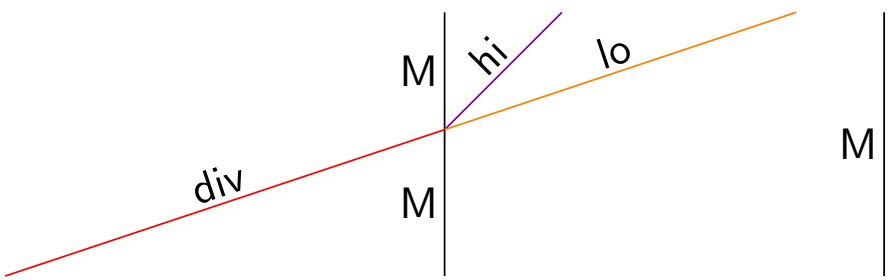
Vocabulary and Example: Find the Middle

Meta-signals (speed)

M	(0)
div	(3)

Collision rules

Vocabulary and Example: Find the Middle



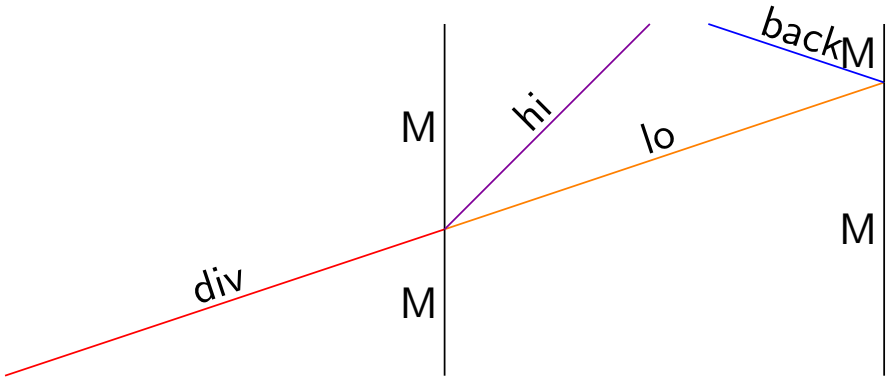
Meta-signals (speed)

M	(0)
div	(3)
hi	(1)
lo	(3)

Collision rules

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

Vocabulary and Example: Find the Middle



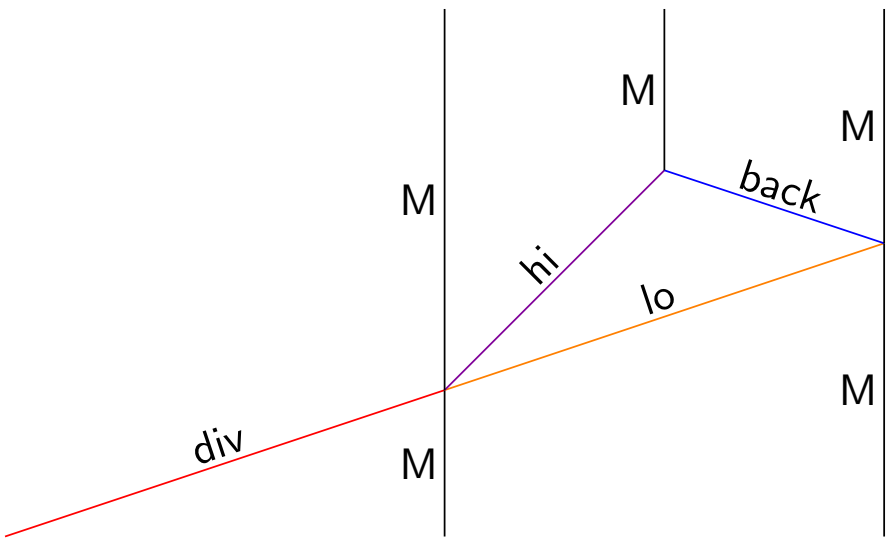
Meta-signals (speed)

M	(0)
div	(3)
hi	(1)
lo	(3)
back	(-3)

Collision rules

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

Vocabulary and Example: Find the Middle



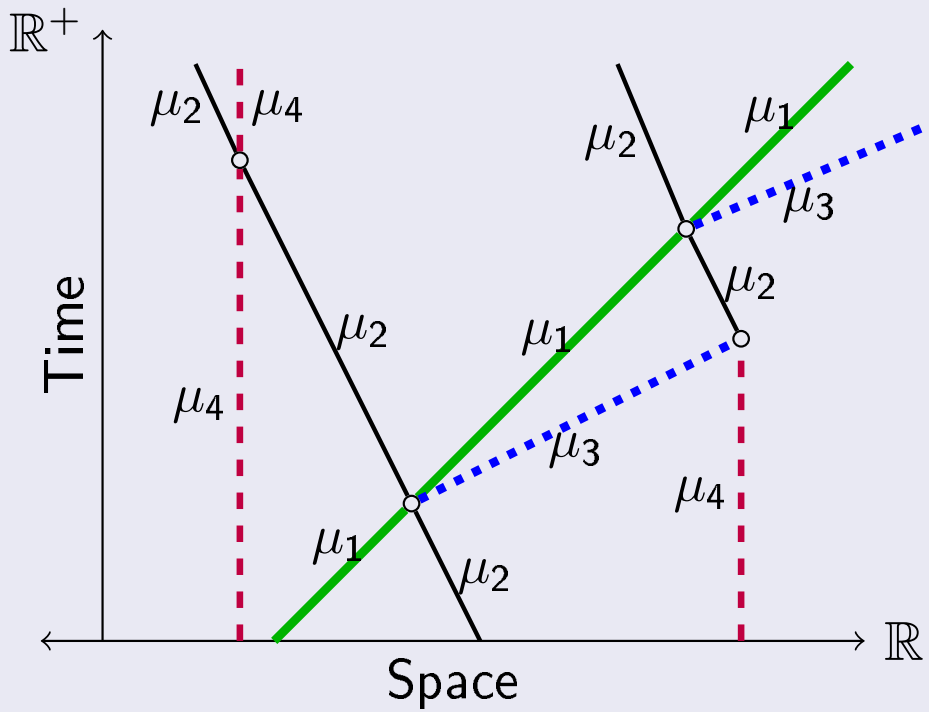
Meta-signals (speed)

M	(0)
div	(3)
hi	(1)
lo	(3)
back	(-3)

Collision rules

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

Another Example



Name	Speed
μ_1	1
μ_2	$-1/2$
μ_3	3
μ_4	0

Collision rules

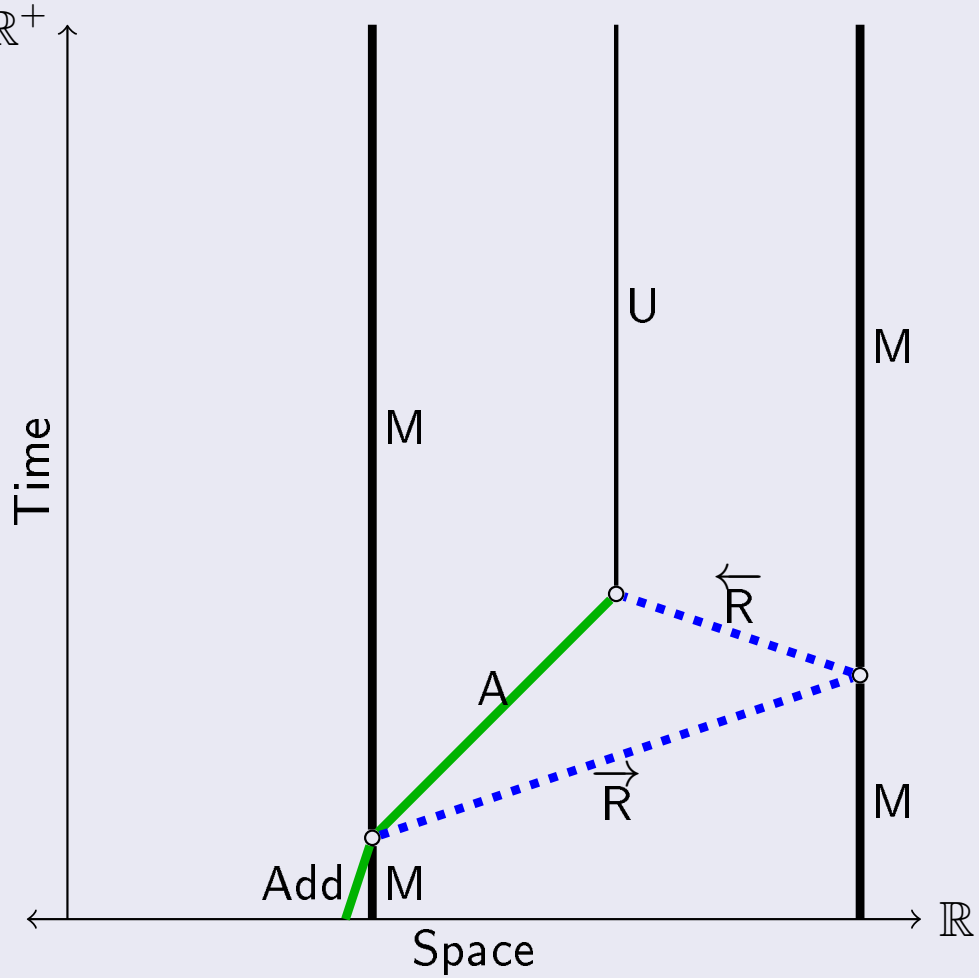
$$\{\mu_1, \mu_2\} \to \{\mu_2, \mu_1, \mu_3\}$$
$$\{\mu_3, \mu_4\} \to \{\mu_2\}$$
$$\{\mu_4, \mu_2\} \to \{\mu_2, \mu_4\}$$

Stack Implantation

Name	Speed
Add, Rem	1/3
A, E	1
U, M	0
$\overrightarrow{R}$	3
$\overleftarrow{R}$	−3

Collision rules

$$\{\text{Add}, \text{M}\} \to \{\text{M}, \text{A}, \overrightarrow{\text{R}}\}$$
$$\{\overrightarrow{\text{R}}, \text{M}\} \to \{\overleftarrow{\text{R}}, \text{M}\}$$
$$\{\text{A}, \overleftarrow{\text{R}}\} \to \{\text{U}\}$$
$$\{\overrightarrow{\text{R}}, \text{U}\} \to \{\overleftarrow{\text{R}}, \text{U}\}$$
$$\{\text{Rem}, \text{M}\} \to \{\text{M}, \text{E}\}$$
$$\{\text{E}, \text{U}\} \to \{\}$$



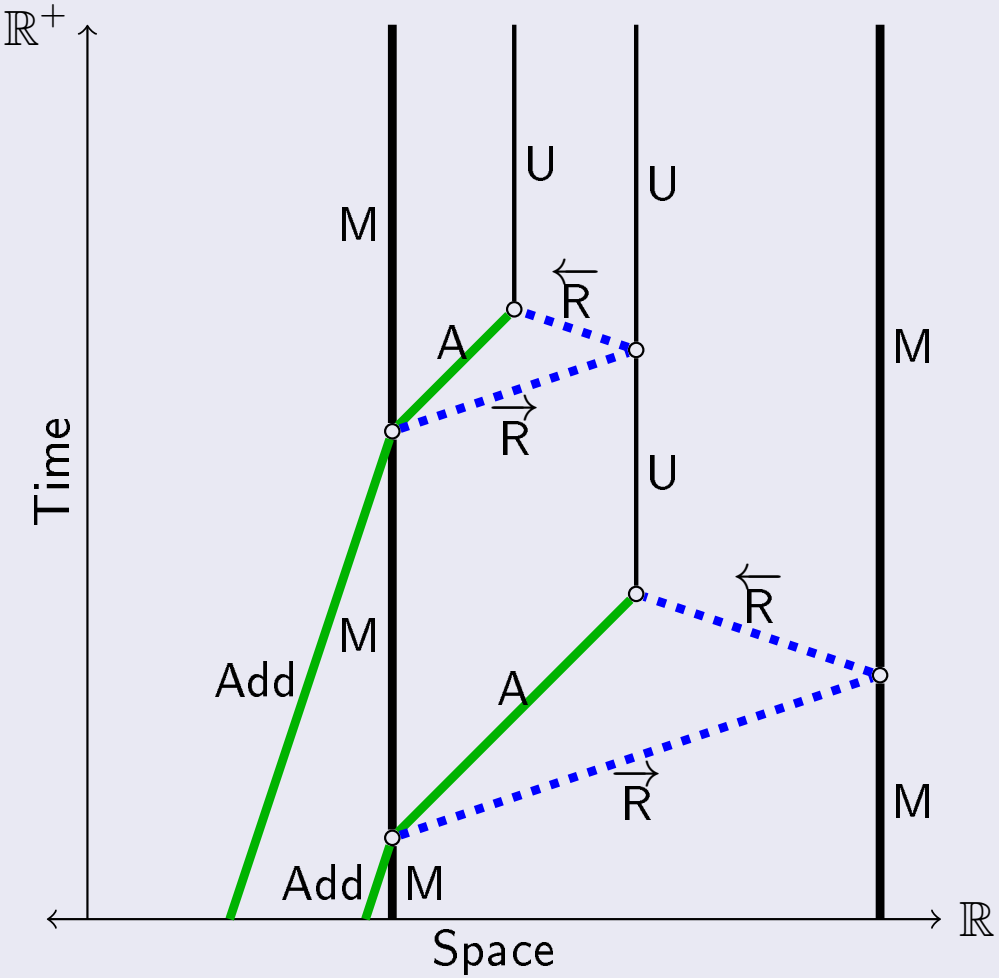
Modify the initial configuration to add another signal U

Stack Implantation

Name	Speed
Add, Rem	1/3
A, E	1
U, M	0
$\overrightarrow{R}$	3
$\overleftarrow{R}$	−3

Collision rules

$$\{\text{Add}, \text{M}\} \to \{\text{M}, \text{A}, \overrightarrow{\text{R}}\}$$
$$\{\overrightarrow{\text{R}}, \text{M}\} \to \{\overleftarrow{\text{R}}, \text{M}\}$$
$$\{\text{A}, \overleftarrow{\text{R}}\} \to \{\text{U}\}$$
$$\{\overrightarrow{\text{R}}, \text{U}\} \to \{\overleftarrow{\text{R}}, \text{U}\}$$
$$\{\text{Rem}, \text{M}\} \to \{\text{M}, \text{E}\}$$
$$\{\text{E}, \text{U}\} \to \{\}$$

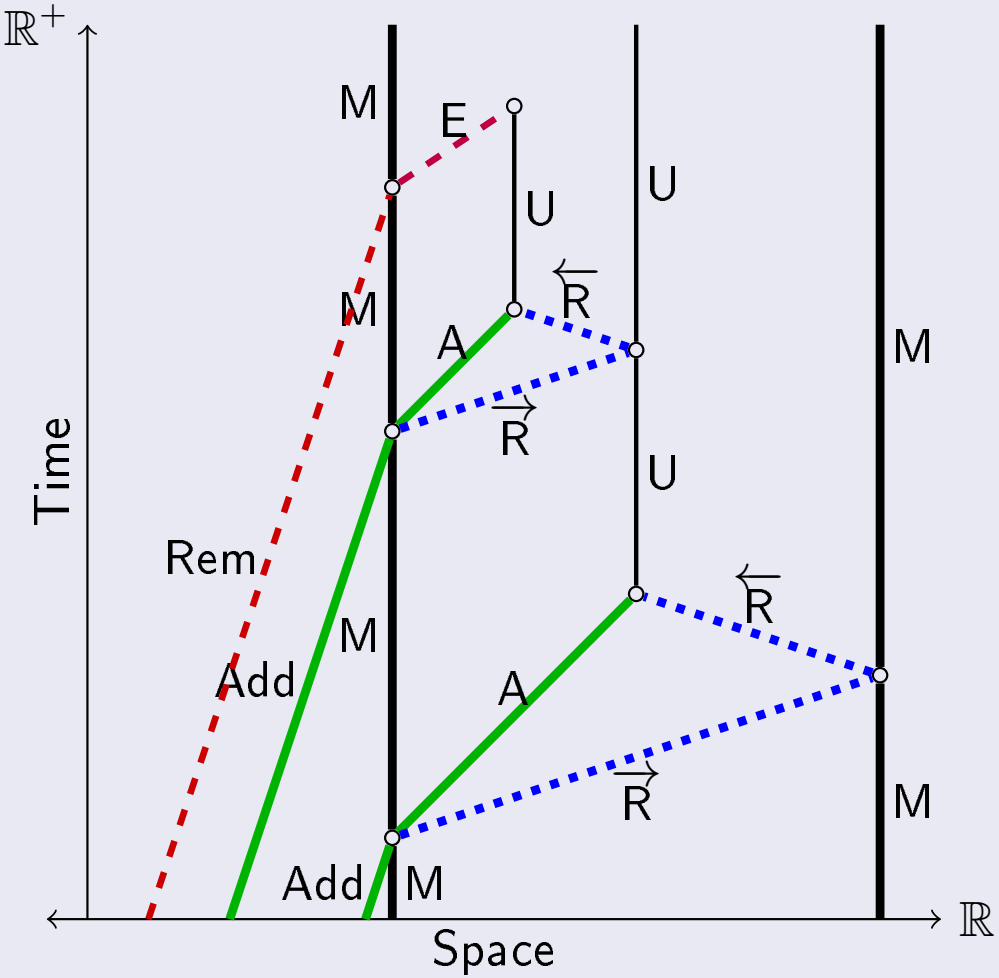


Stack Implantation

Name	Speed
Add, Rem	1/3
A, E	1
U, M	0
$\overrightarrow{R}$	3
$\overleftarrow{R}$	−3

Collision rules

$$\{\text{Add}, \text{M}\} \to \{\text{M}, \text{A}, \overrightarrow{\text{R}}\}$$
$$\{\overrightarrow{\text{R}}, \text{M}\} \to \{\overleftarrow{\text{R}}, \text{M}\}$$
$$\{\text{A}, \overleftarrow{\text{R}}\} \to \{\text{U}\}$$
$$\{\overrightarrow{\text{R}}, \text{U}\} \to \{\overleftarrow{\text{R}}, \text{U}\}$$
$$\{\text{Rem}, \text{M}\} \to \{\text{M}, \text{E}\}$$
$$\{\text{E}, \text{U}\} \to \{\}$$

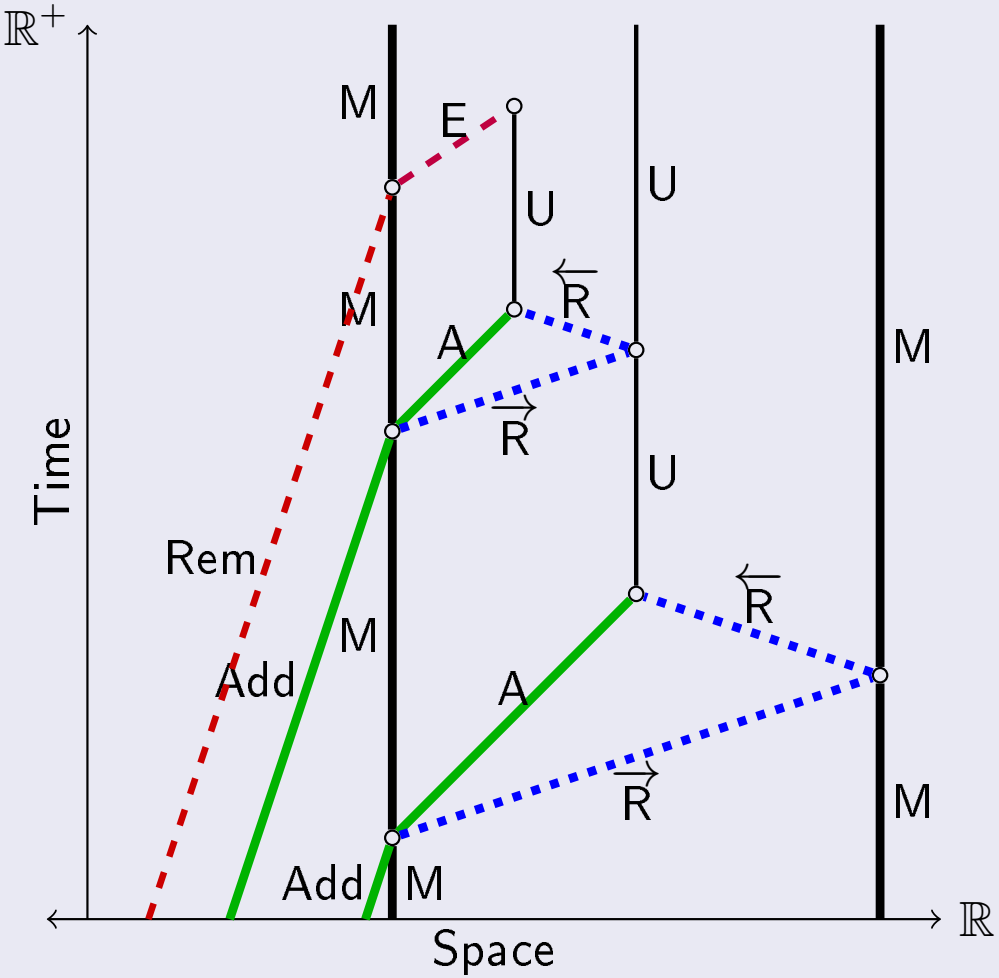


Stack Implantation

Name	Speed
Add, Rem	1/3
A, E	1
U, M	0
$\overrightarrow{R}$	3
$\overleftarrow{R}$	−3

Collision rules

$$\{\text{Add}, \text{M}\} \to \{\text{M}, \text{A}, \overrightarrow{\text{R}}\}$$
$$\{\overrightarrow{\text{R}}, \text{M}\} \to \{\overleftarrow{\text{R}}, \text{M}\}$$
$$\{\text{A}, \overleftarrow{\text{R}}\} \to \{\text{U}\}$$
$$\{\overrightarrow{\text{R}}, \text{U}\} \to \{\overleftarrow{\text{R}}, \text{U}\}$$
$$\{\text{Rem}, \text{M}\} \to \{\text{M}, \text{E}\}$$
$$\{\text{E}, \text{U}\} \to \{\}$$



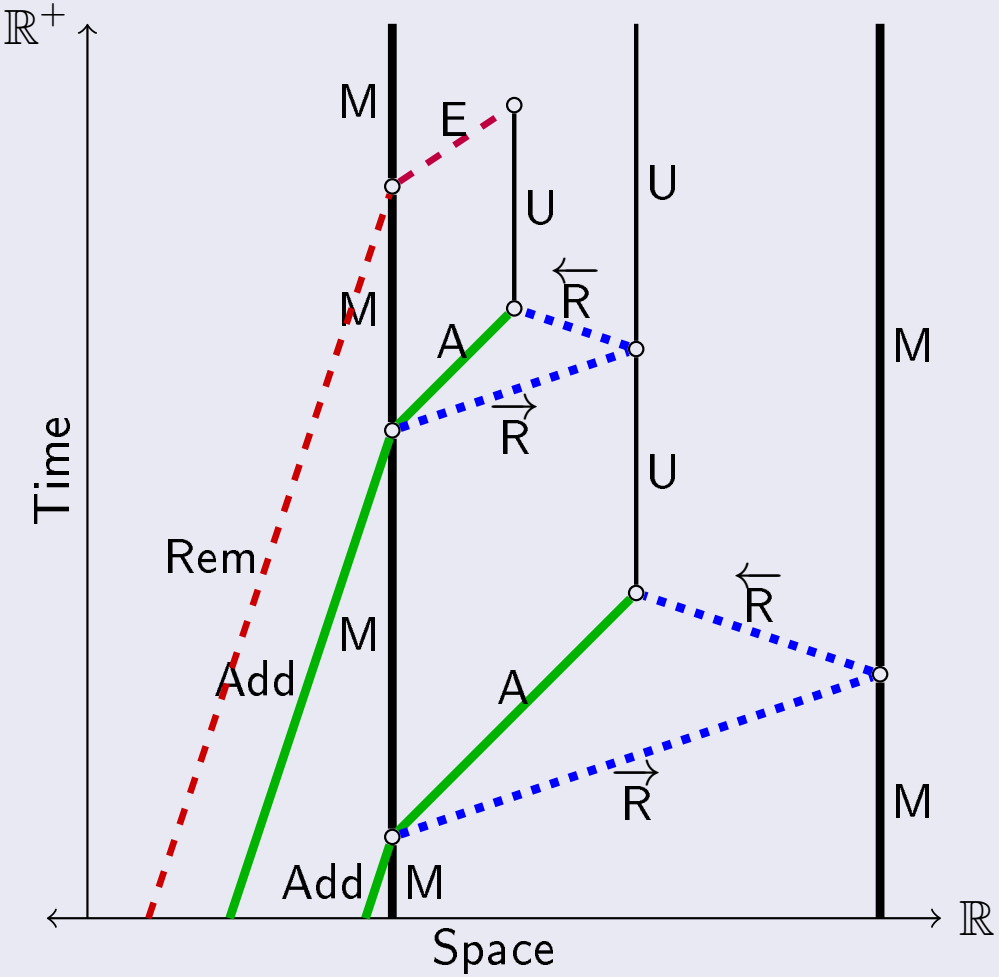
Modify the initial configuration to remove the other signal U

Stack Implantation

Name	Speed
Add, Rem	1/3
A, E	1
U, M	0
$\overrightarrow{R}$	3
$\overleftarrow{R}$	−3

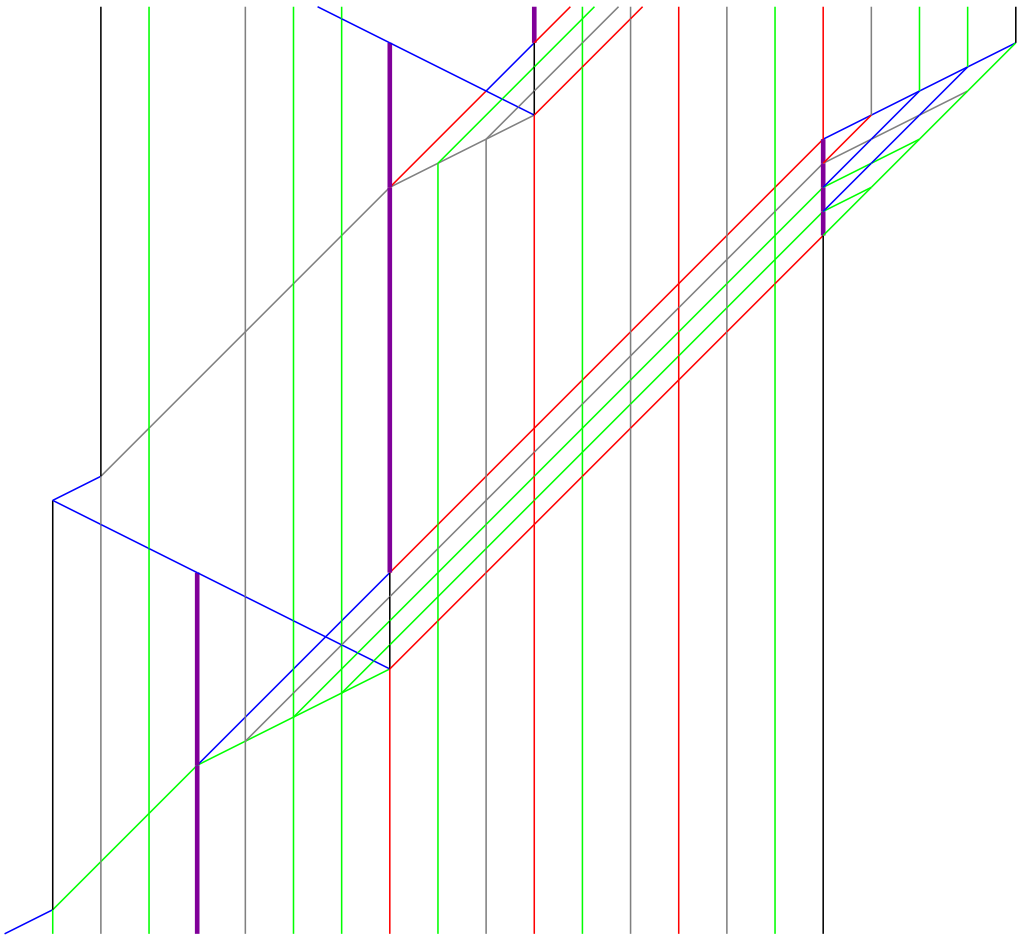
Collision rules

$$\{\text{Add}, \text{M}\} \to \{\text{M}, \text{A}, \overrightarrow{\text{R}}\}$$
$$\{\overrightarrow{\text{R}}, \text{M}\} \to \{\overleftarrow{\text{R}}, \text{M}\}$$
$$\{\text{A}, \overleftarrow{\text{R}}\} \to \{\text{U}\}$$
$$\{\overrightarrow{\text{R}}, \text{U}\} \to \{\overleftarrow{\text{R}}, \text{U}\}$$
$$\{\text{Rem}, \text{M}\} \to \{\text{M}, \text{E}\}$$
$$\{\text{E}, \text{U}\} \to \{\}$$

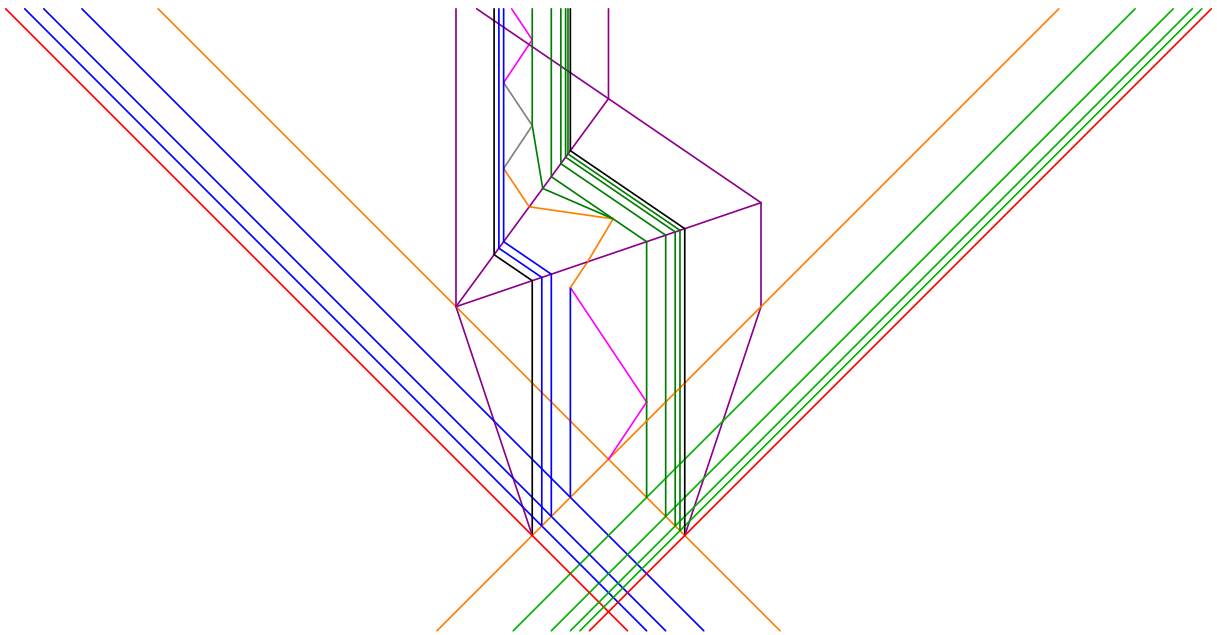


Add a rule so that no signal extends on the right (when there is no U)

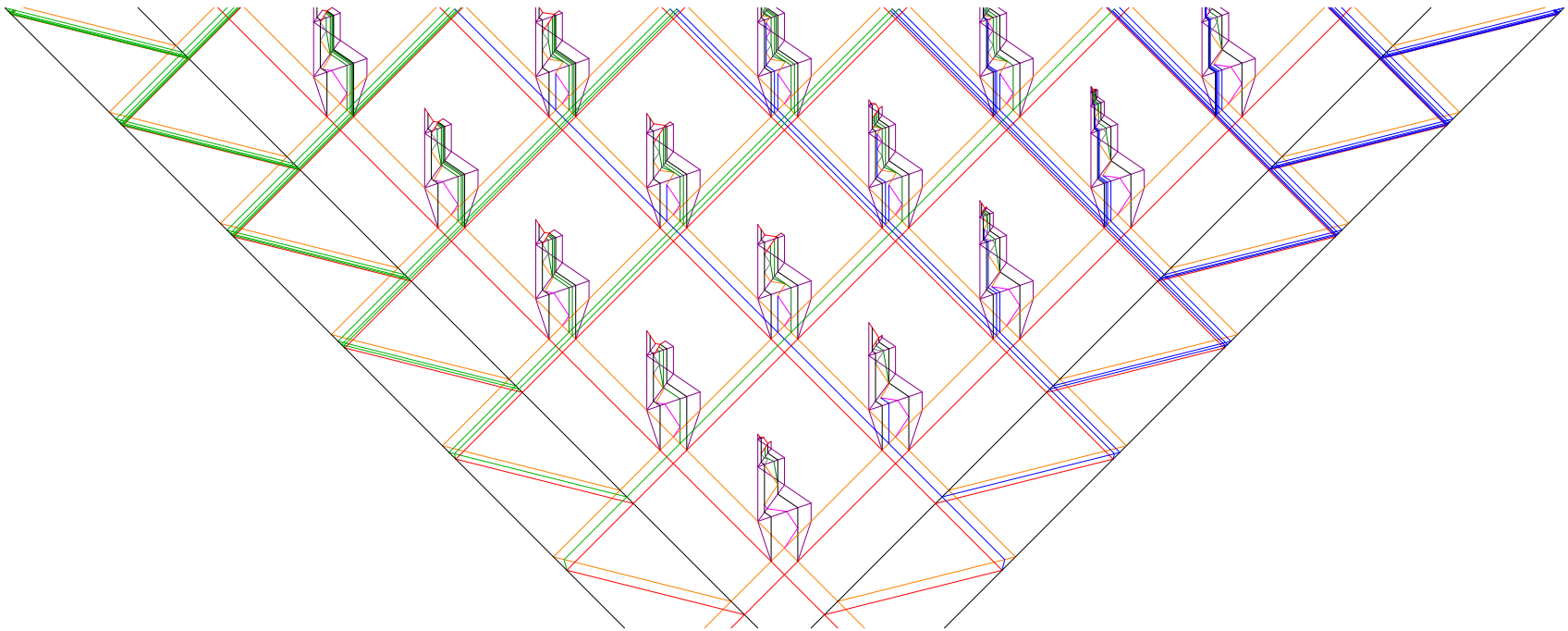
Complex Dynamics



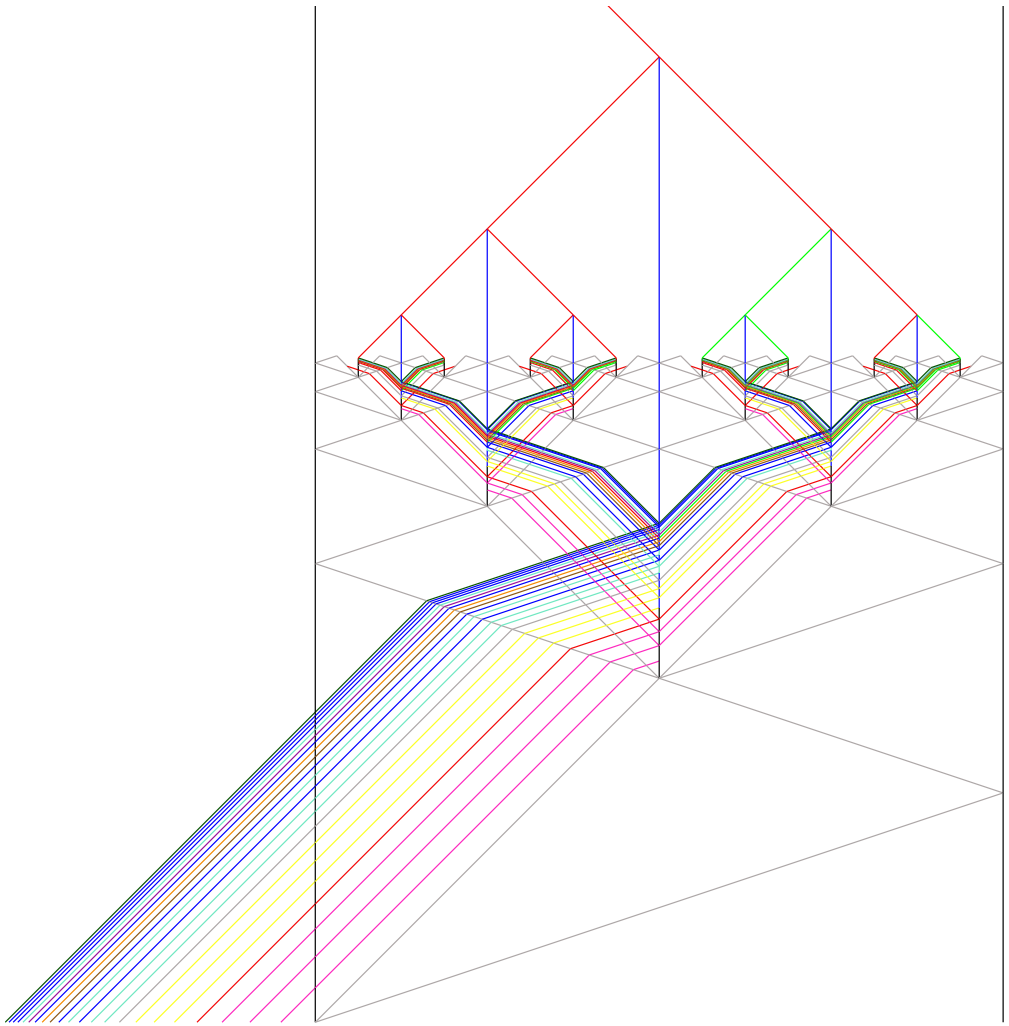
Complex Dynamics



Complex Dynamics

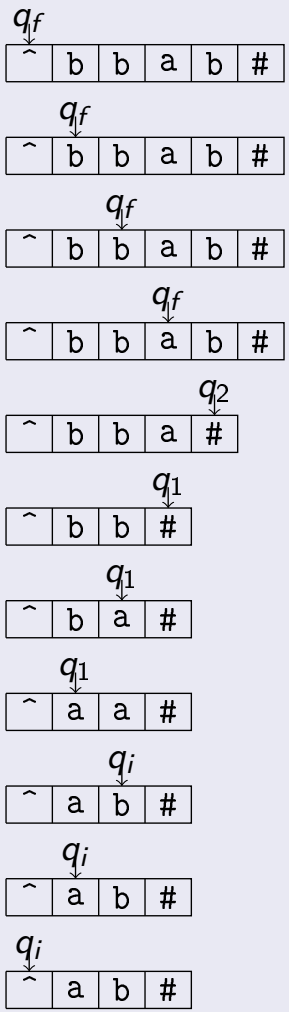


Complex Dynamics

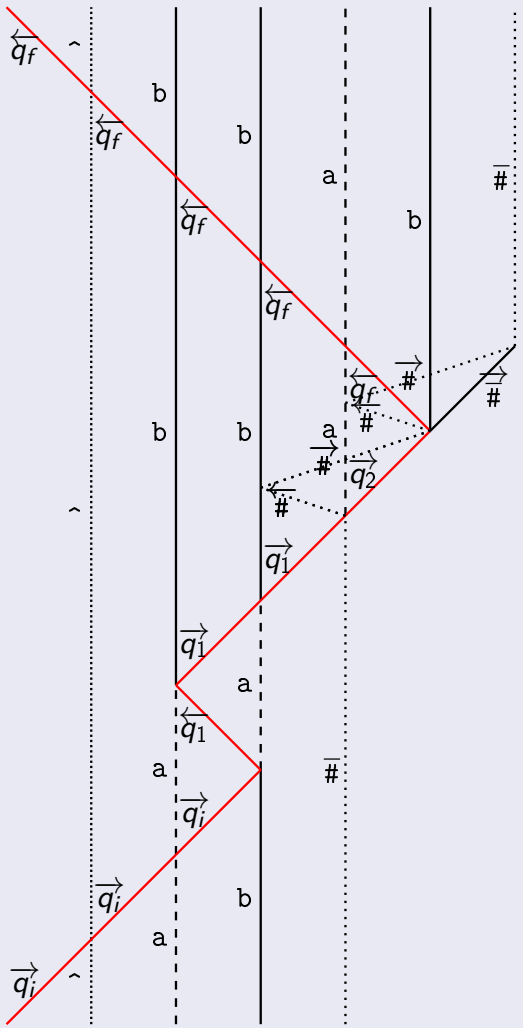


(Turing-)computation

Turing Machine

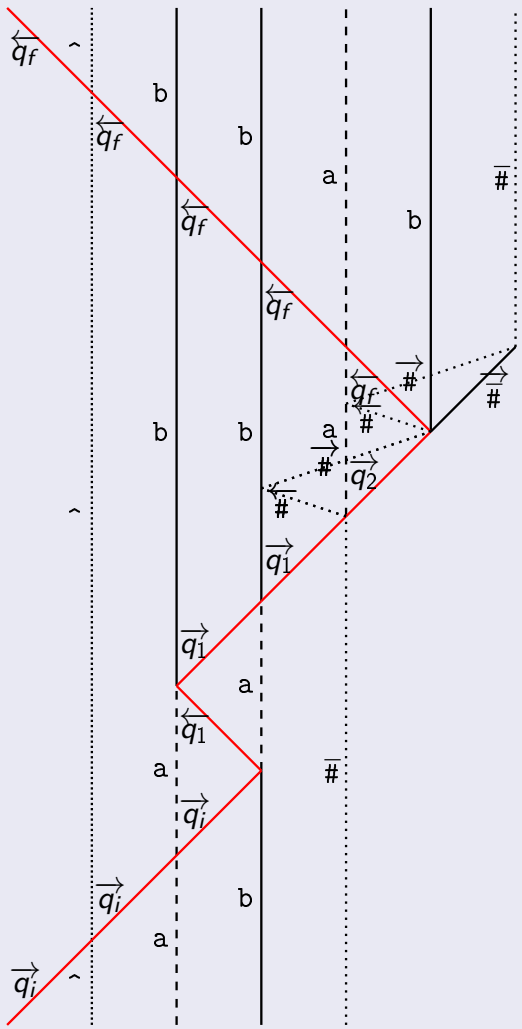


Simulation



(Turing-)computation

Simulation



Rational machines

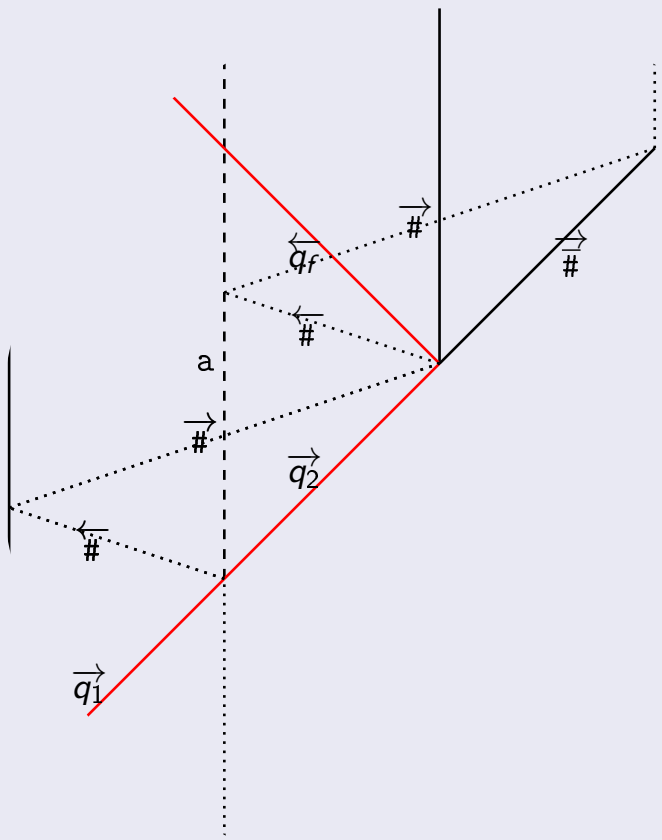
- speeds $\in \mathbb{Q}$
- initial positions $\in \mathbb{Q}$
- $\Rightarrow$ collision coordinates $\in \mathbb{Q}$
- exact simulation on computer/TM

Undecidability

- finite number de collisions
- meta-signal appereance
- use of a rule
- disappearing of all signals
- involvement of a signal in any collision
- extension on the side, etc.

(Turing-)computation

Simulation



Exercise

A new *cell* is always added at the same distance on the right.

Head signals have speed 1 and -1.

$\overrightarrow{\#}$ has speed 1.

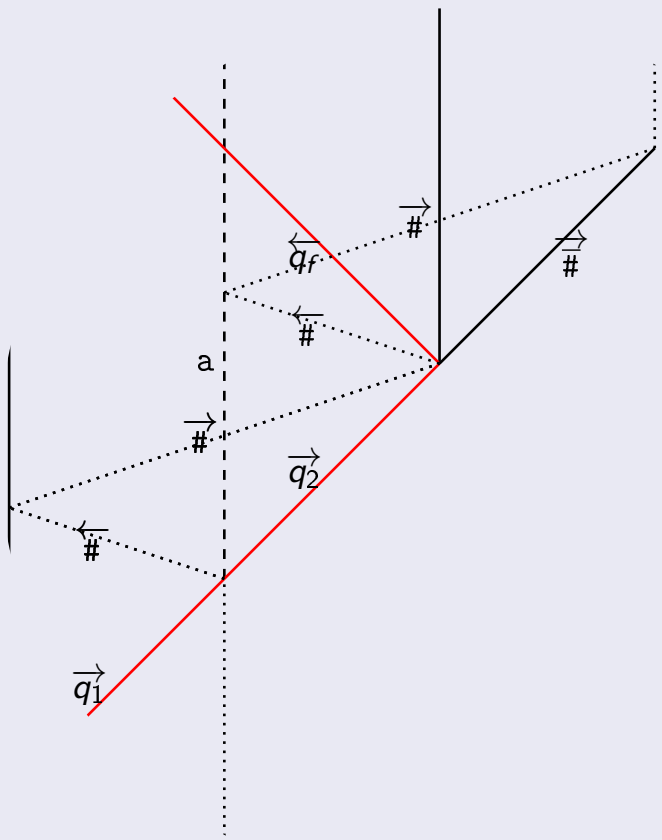
$\overleftarrow{\#}$ and $\overrightarrow{\#}$ have oposite speed.

What are the speed of $\overleftarrow{\#}$ and $\overrightarrow{\#}$?

Recognize a backward consruction to check the result.

(Turing-)computation

Simulation



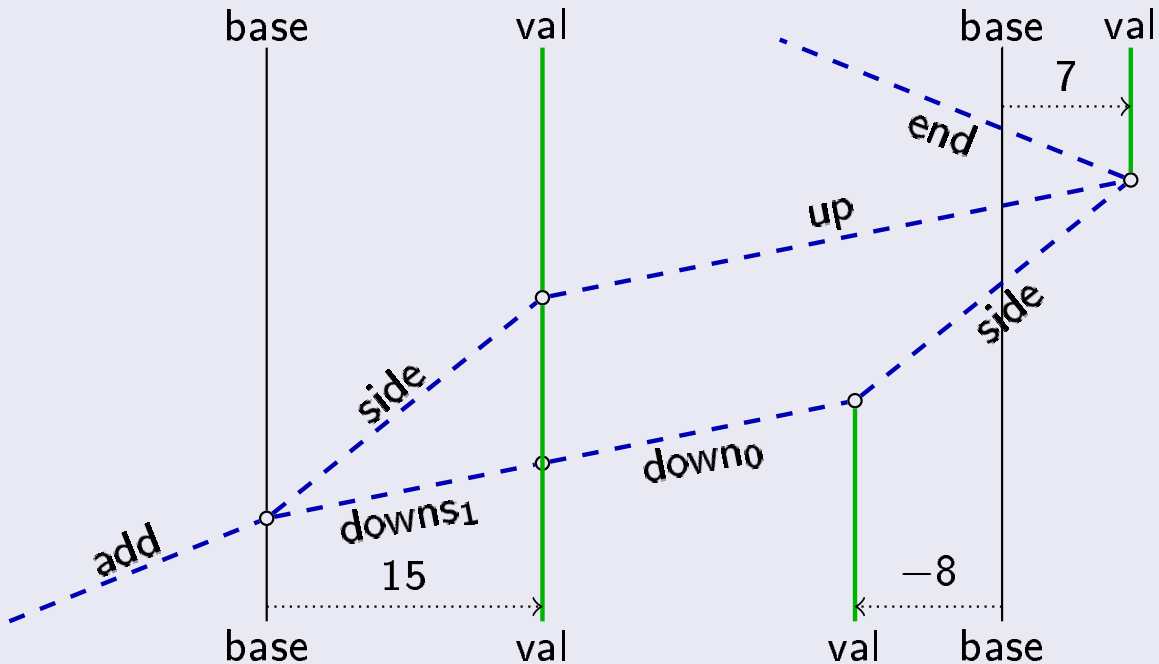
Exercise

How to change the speed so that the distance is halved each time?

This way the ribbon remains in a bounded space.

Computing with Real Numbers

Addition (value ≈ distance)

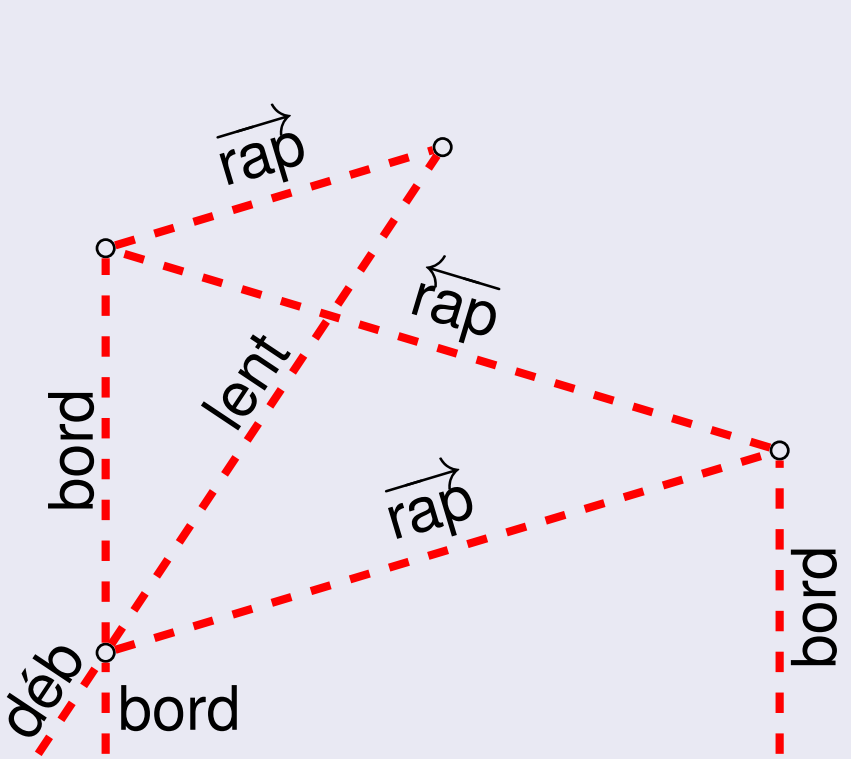


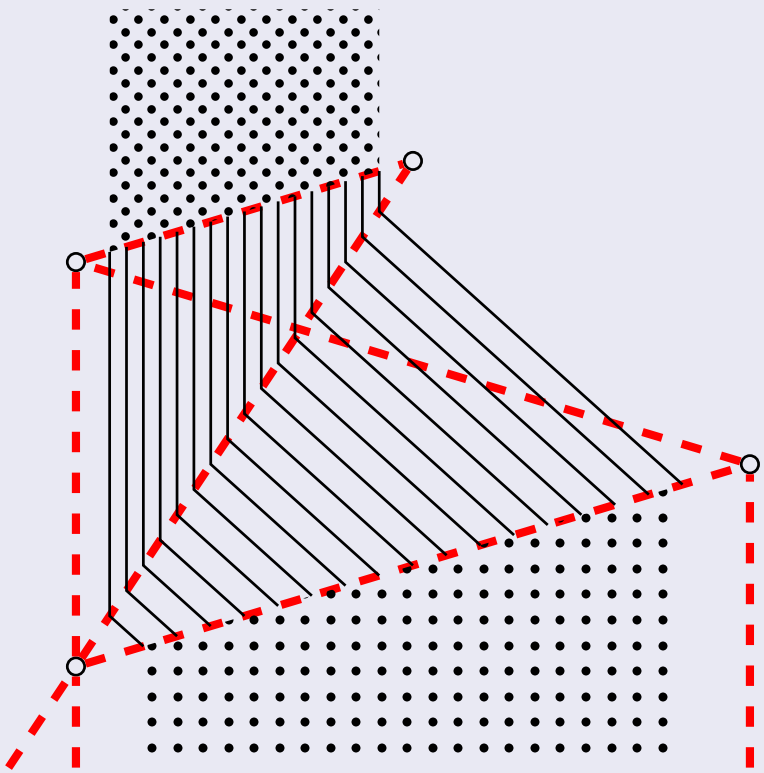
Characterization Out of Accumulation: lin-BSS

- addition, subtraction
- multiplication by a constant
- test de sign, branch

Contraction

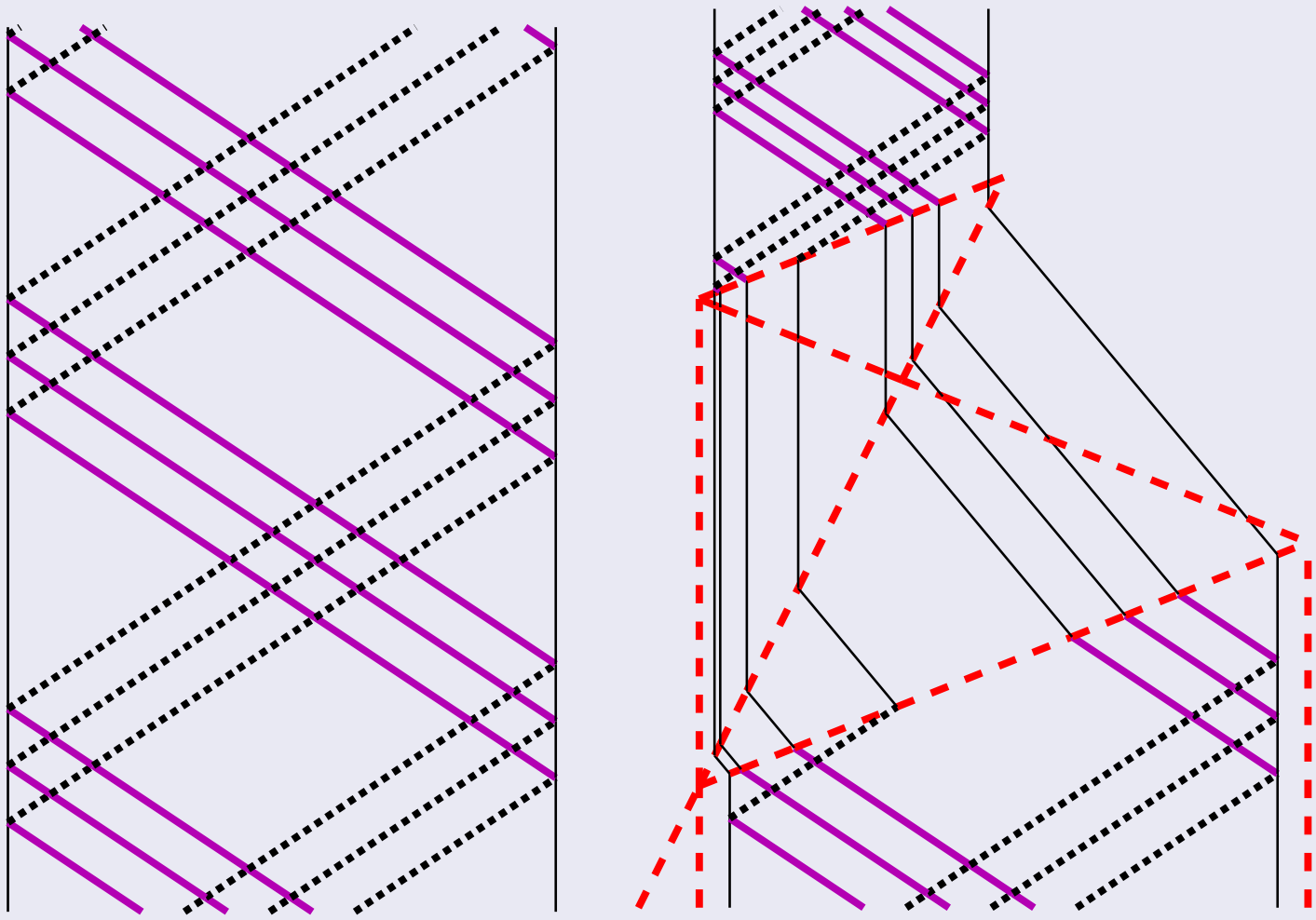
Principe





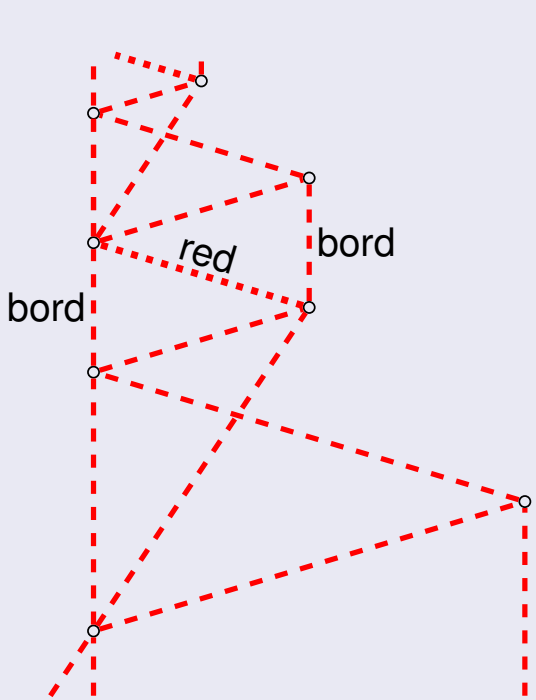
Contraction

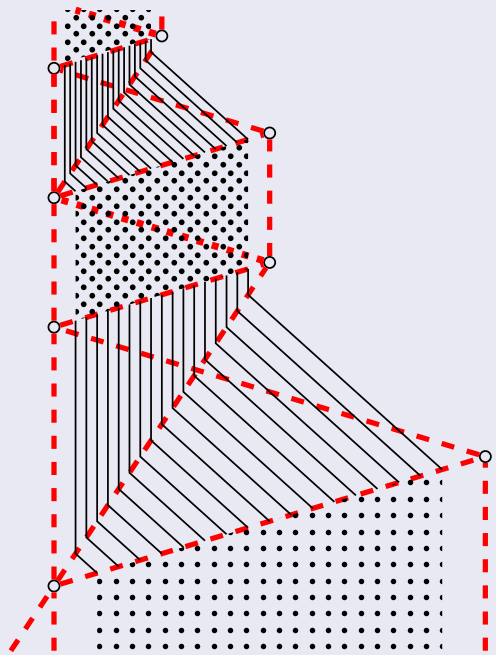
Example

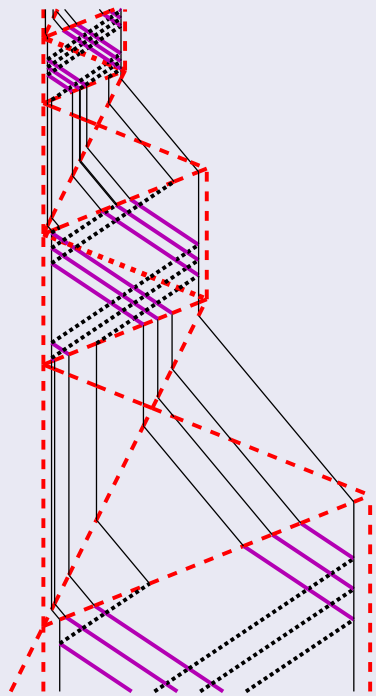


Iterated Contraction

Principe and Example







Consequences

Folding Space

- Any *computation* initiated on a finite potion of space can be folded in a finite portion of space-time
- A "bijective correspondence" between an unbounded part of $\mathbb{R}^2$ and a bounded part

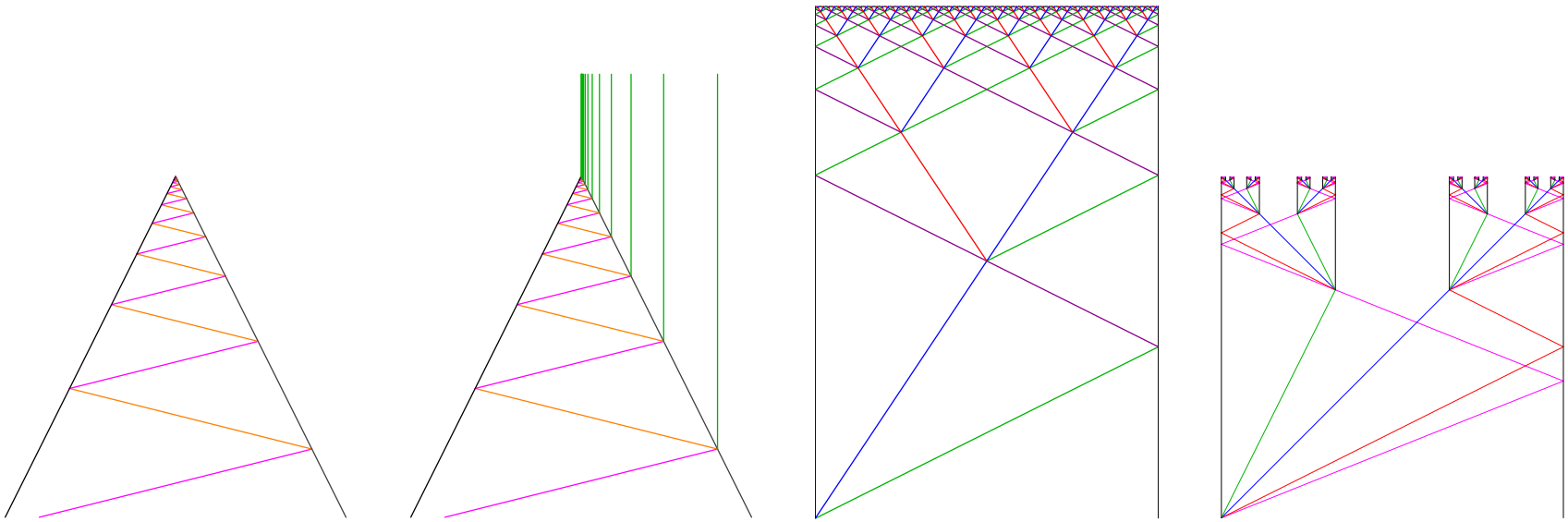
Two Time Scales

- continuous time: finite duration
- discrete time (collisions): infinity of moments

Black Hole Model: Decide the Halting Problem (and Above)

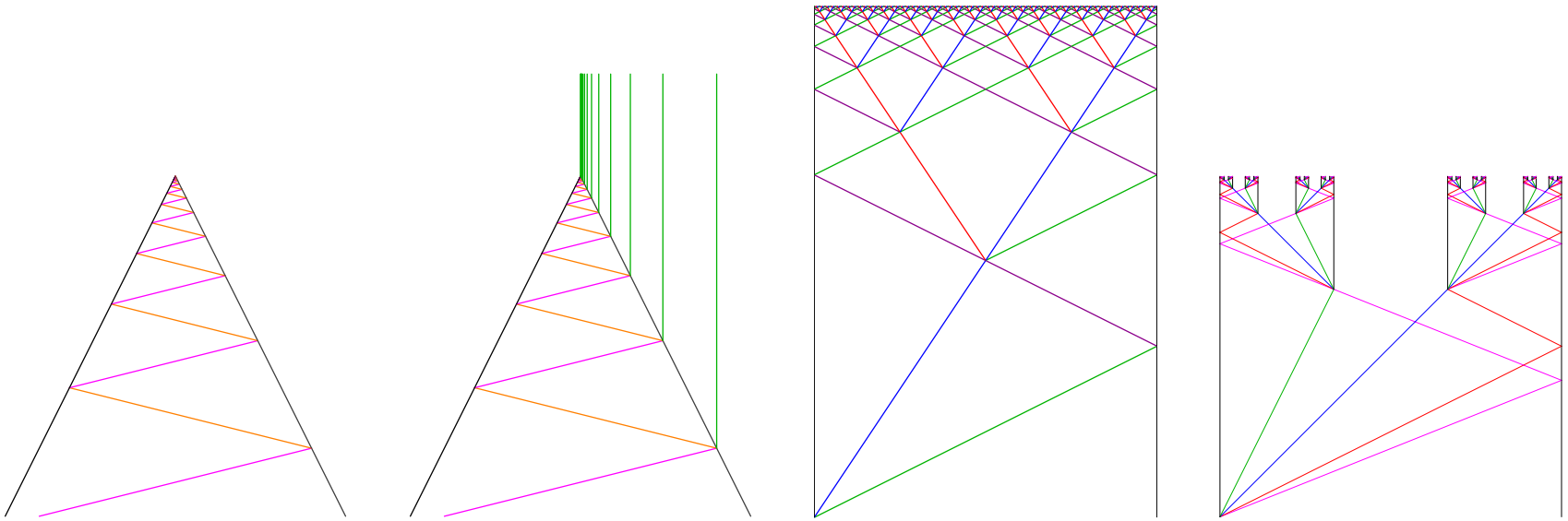
- (Turing)-computation intricated but any output leaves free
- outside, one waits for the result
- after a *known bounded* time, either an end signal was received or there were no end.

Fractal Generation



How to limit a fractal construction?

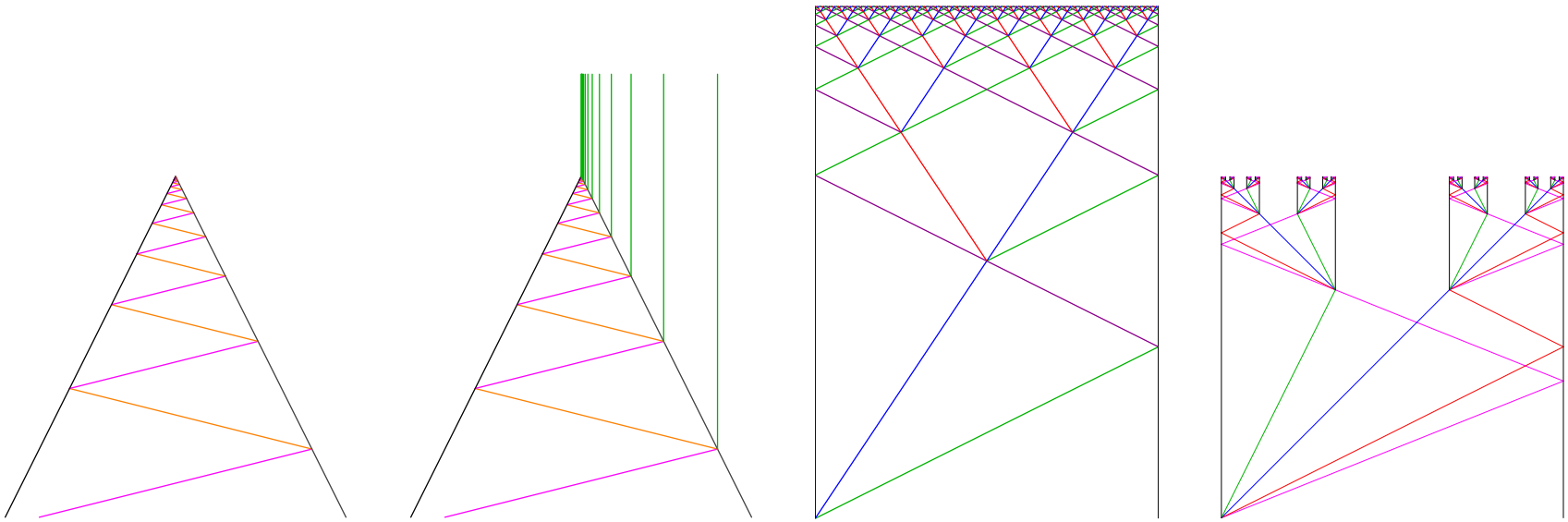
Fractal Generation



How to limit a fractal construction?

- counters embedded in meta-signals
  (*ad hoc* and not generic enough).

Fractal Generation



How to limit a fractal construction?

- counters embedded in meta-signals
  (*ad hoc* and not generic enough).
- propagation orders to produce one more level
  (for 10 levels, 10 orders)

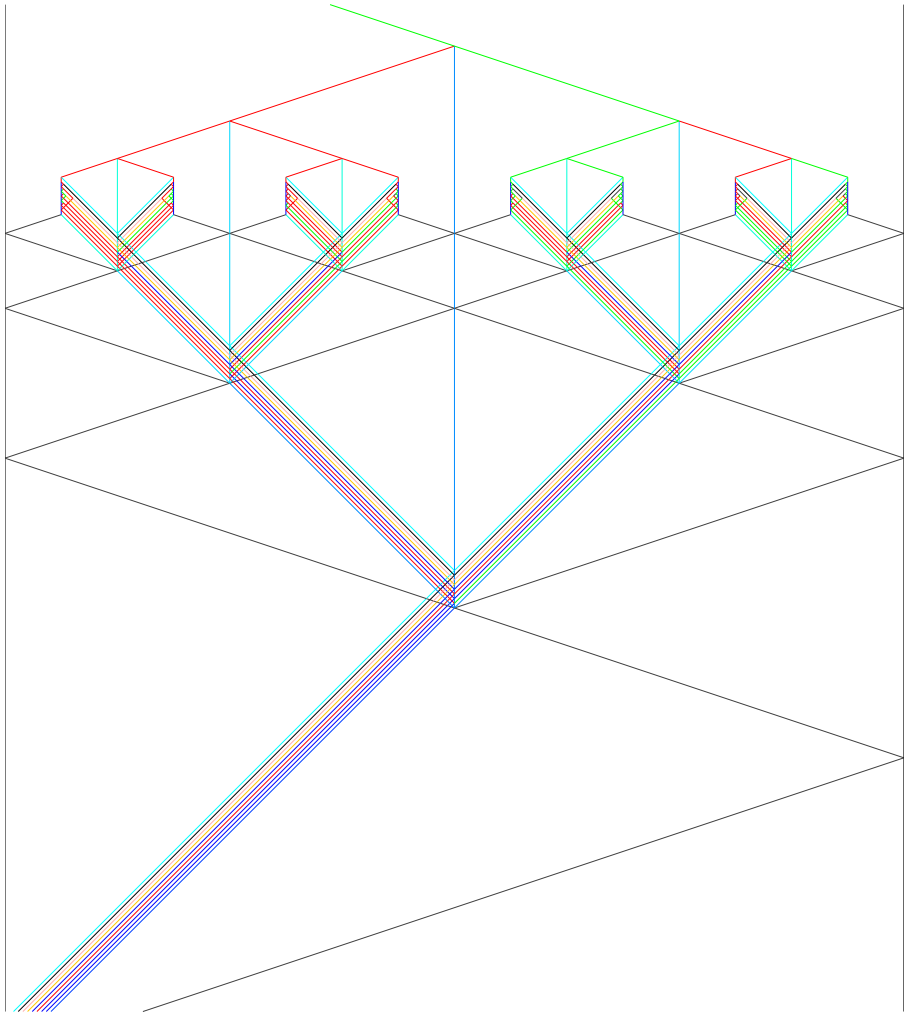
Quantified Boolean Formula Satisfaction

QSAT

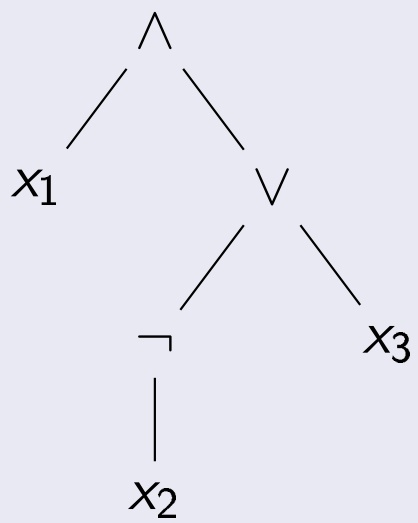
- $\exists x_1 \forall x_2 \forall x_3 x_1 \wedge (\neg x_2 \vee x_3)$

Duchier et al. (2011)

- 1 QSAT formula
  $\rightsquigarrow$ 1 signal machine

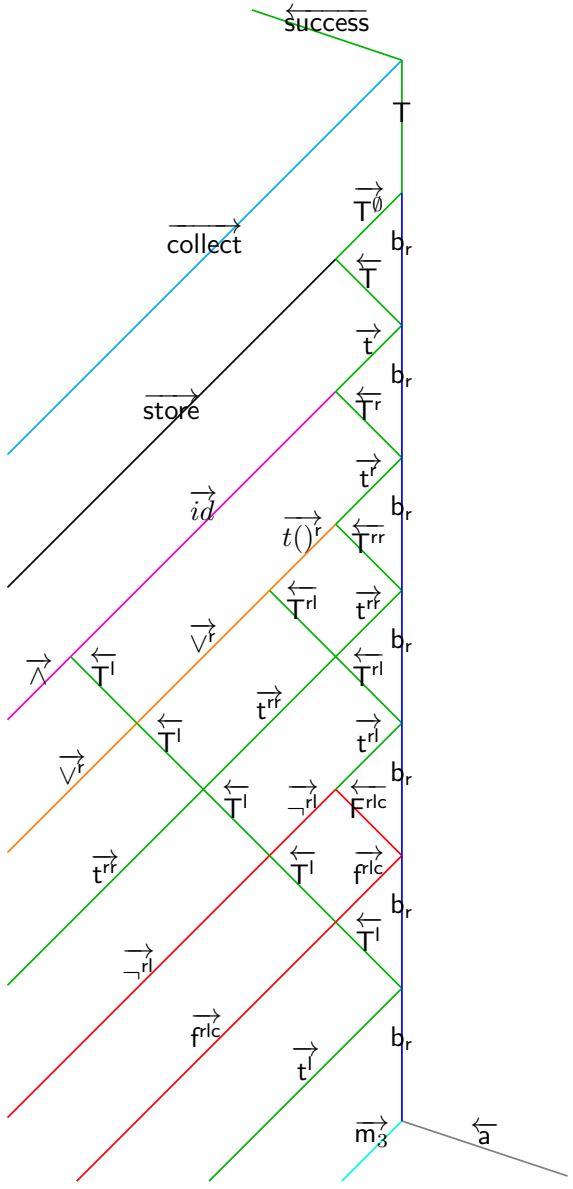


Evaluating the Formula

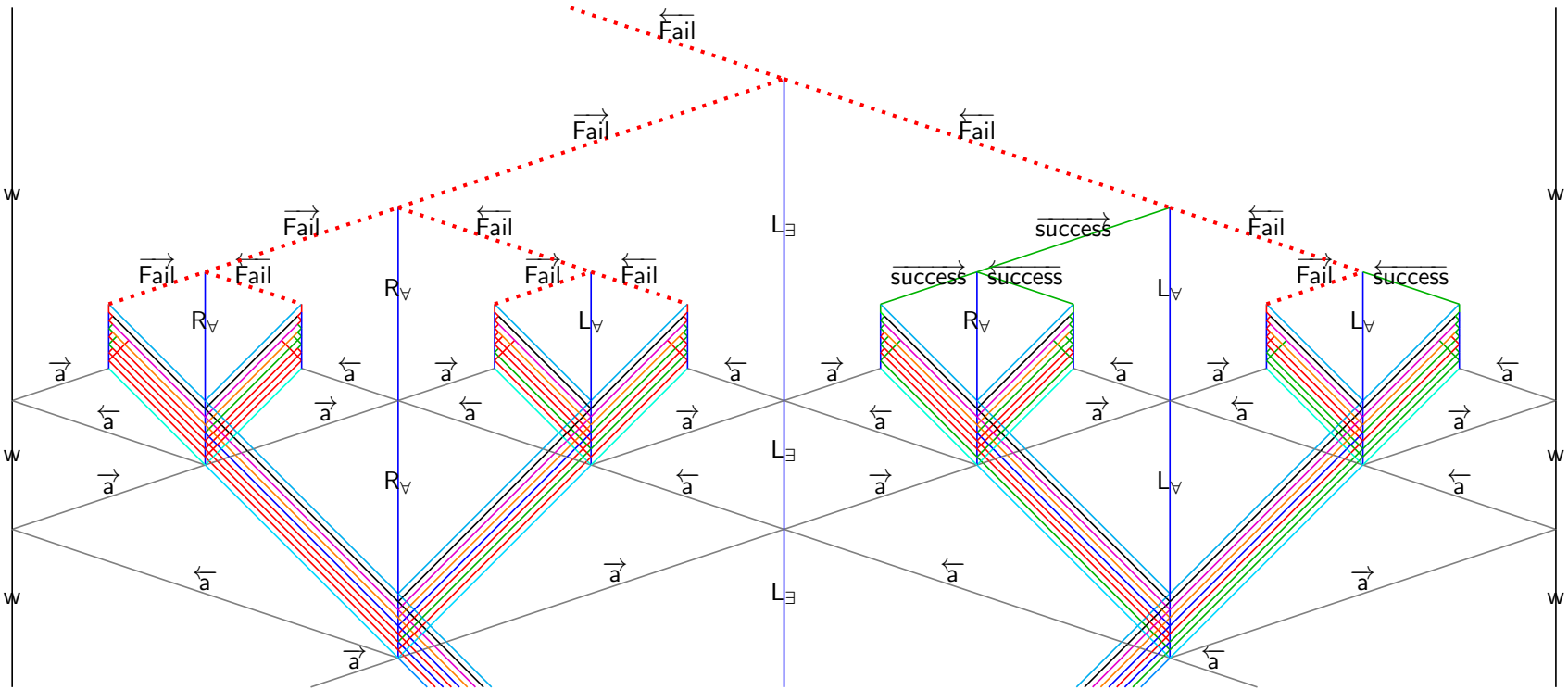


cases:

- x_1 true
- x_2 false
- x_3 true



Collecting the Results

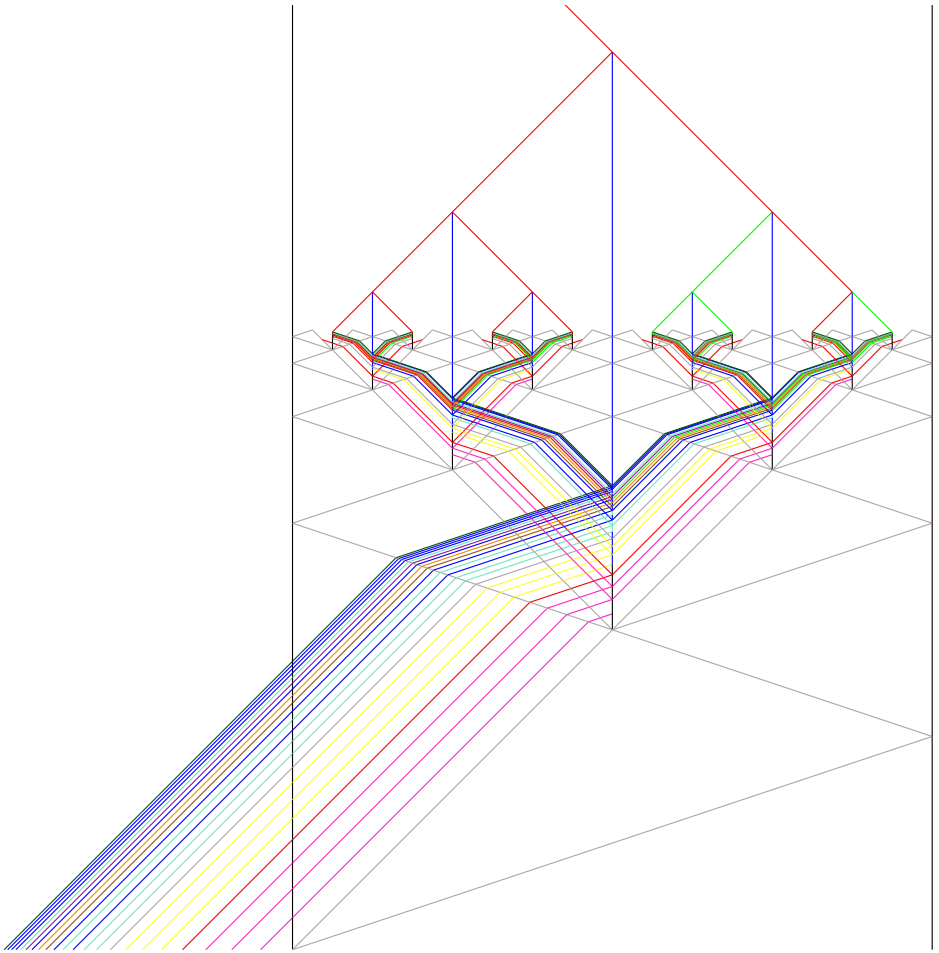


Complexity

- constant duration
- quadratic depth
- exponential width

Generic Machine for QSAT (Duchier et al., 2012)

- formula encoded in the initial configuration
- constant duration
- cubic depth
- exponential width



Intrinsic Universality

Being able to *simulate* any other dynamical system of the its *class*.

Cellular Automata

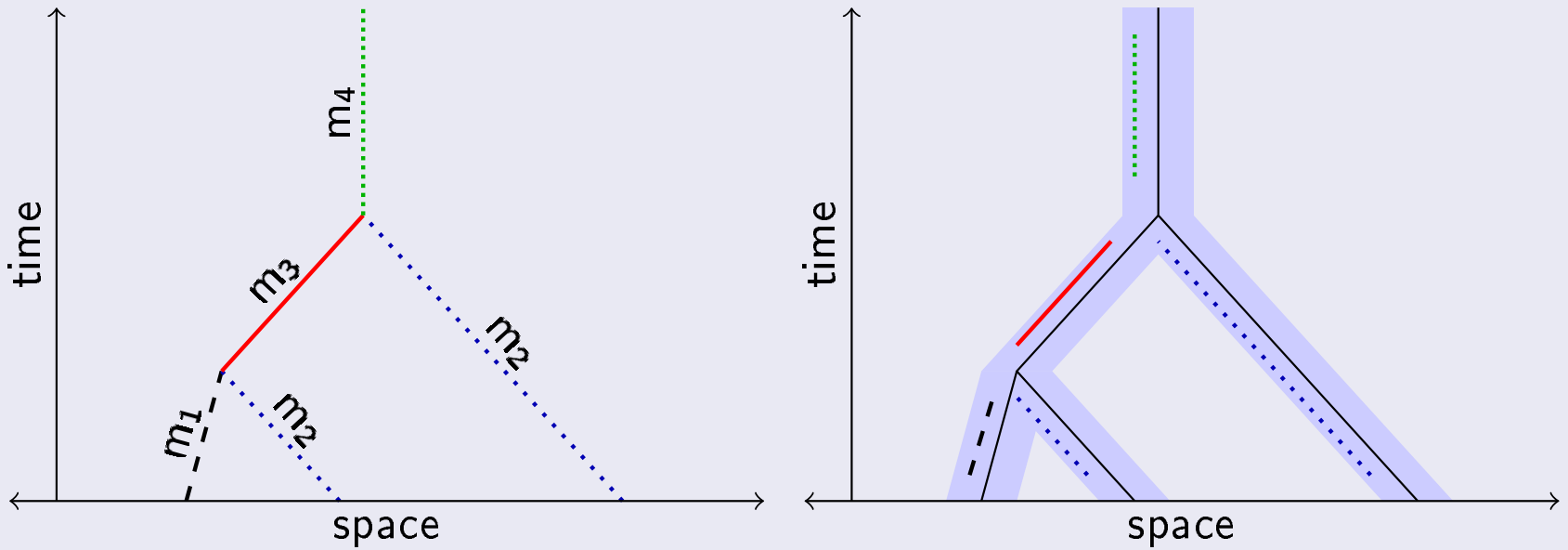
- regular (Albert and Čulik II, 1987; Mazoyer and Rapaport, 1998; Ollinger, 2001)
- reversible (Durand-Lose, 1997)

Tile Assembly Systems

- possible at T=2 and above (Woods, 2013)
- impossible at T=1 (Meunier et al., 2014)

Simulation for Signal Machines

Space-Time Diagram Mimicking



Signal Machine Simulation

$\mathcal{U}_S$ simulates $\mathcal{M}$ if there is function from the configurations of $\mathcal{M}$ to the ones of $\mathcal{U}_S$ s.t. the space-time issued from the image always mimics the original one.

Join work [Submitted]

Florent Becker, LIFO, U. Orléans
Mohammad-Hadi Foroughmand-Araabi, Sharif U. T., Tehran, Iran
Sama Goliaei, University of Tehran, Tehran, Iran

Theorem

- For any finite set of real numbers $\mathcal{S}$, there is a signal machine $\mathcal{U}_\mathcal{S}$, that can simulate any machine whose speeds belong to $\mathcal{S}$.
- The set of $\mathcal{U}_\mathcal{S}$ where $\mathcal{S}$ ranges over finite sets of real numbers is an intrinsically universal family of signal machines.

Rest of this part

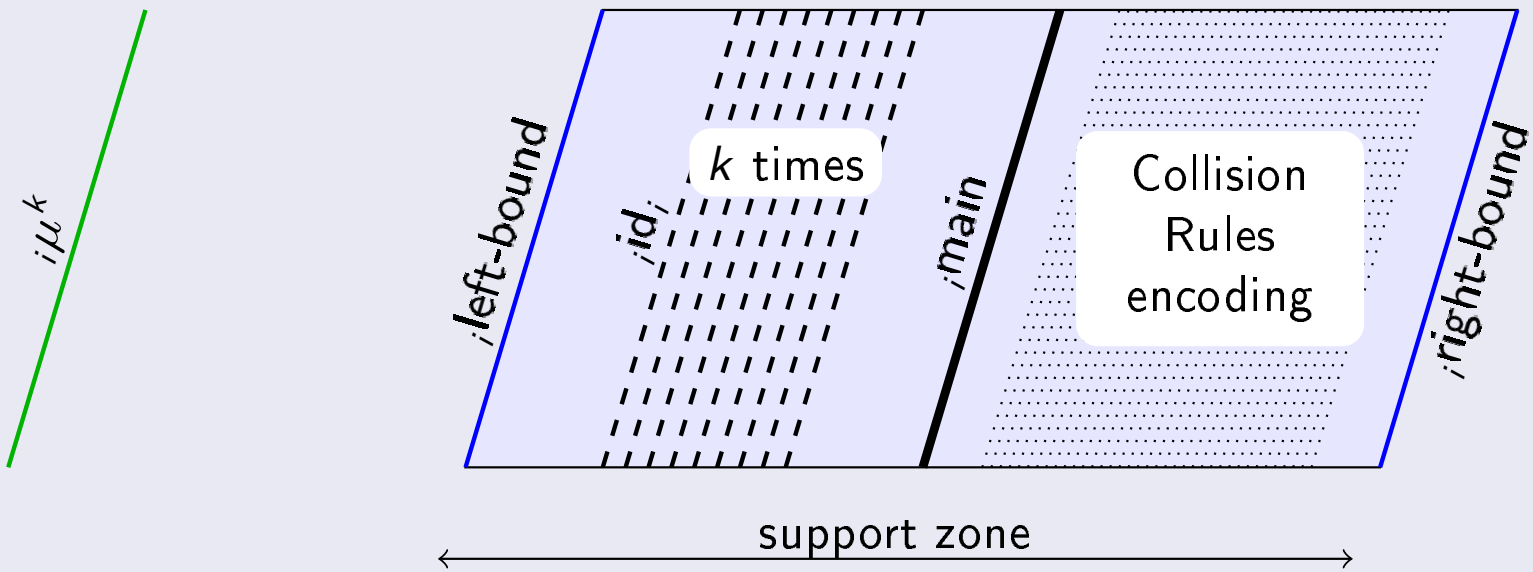
Let $\mathcal{S}$ be any finite set of real numbers,
let $\mathcal{M}$ be any signal machine whose speeds belongs to $\mathcal{S}$,
$\mathcal{U}_\mathcal{S}$ is progressively constructed as simulation is presented.

Macro-Signal

- Meta-signal of $\mathcal{M}$ identified with numbers
- Unary encoding of numbers

Macro-Signal Structure

${}_i\mu^k$: kth signal, ith speed

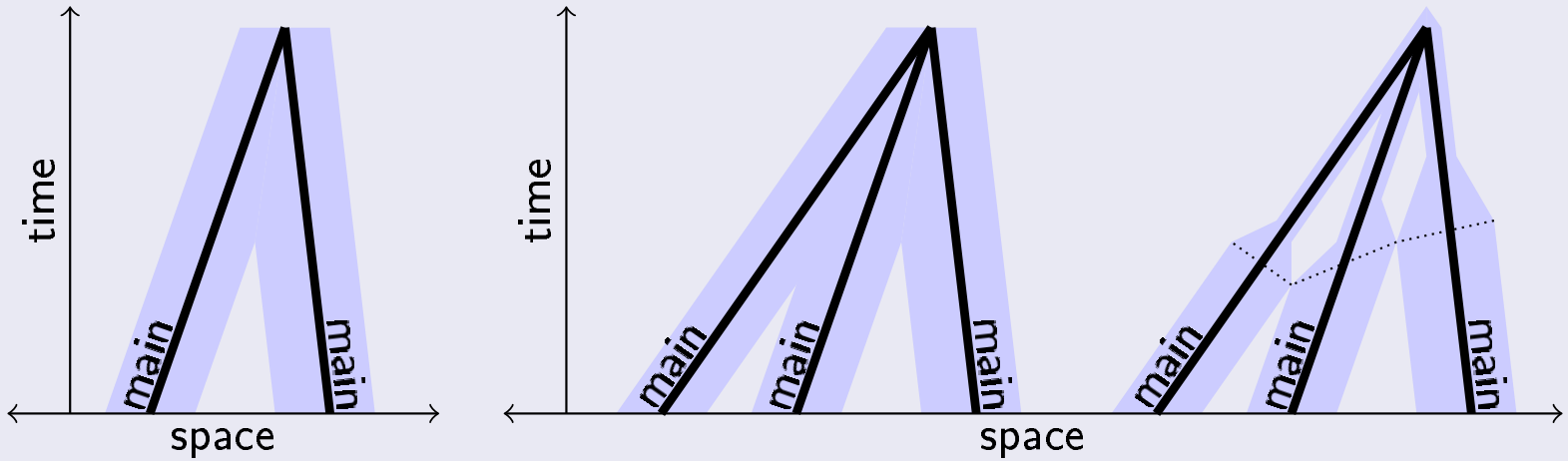


Global scheme

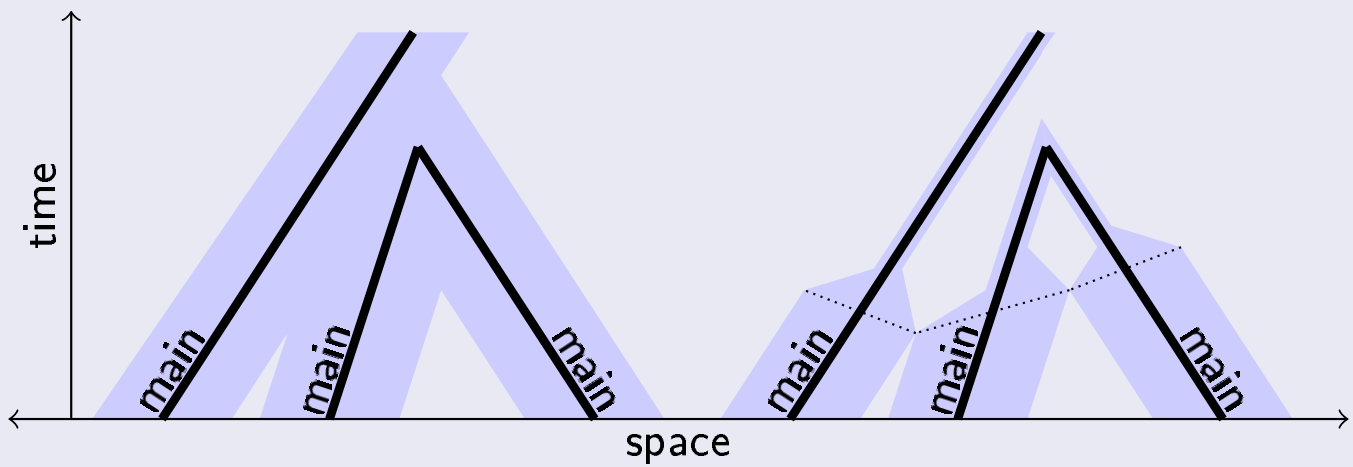
When Support Zones Meet

1. provide a delay
2. test if macro-collision is appropriate and what macro-signals are involved
3. if OK
    1. start the macro-collision

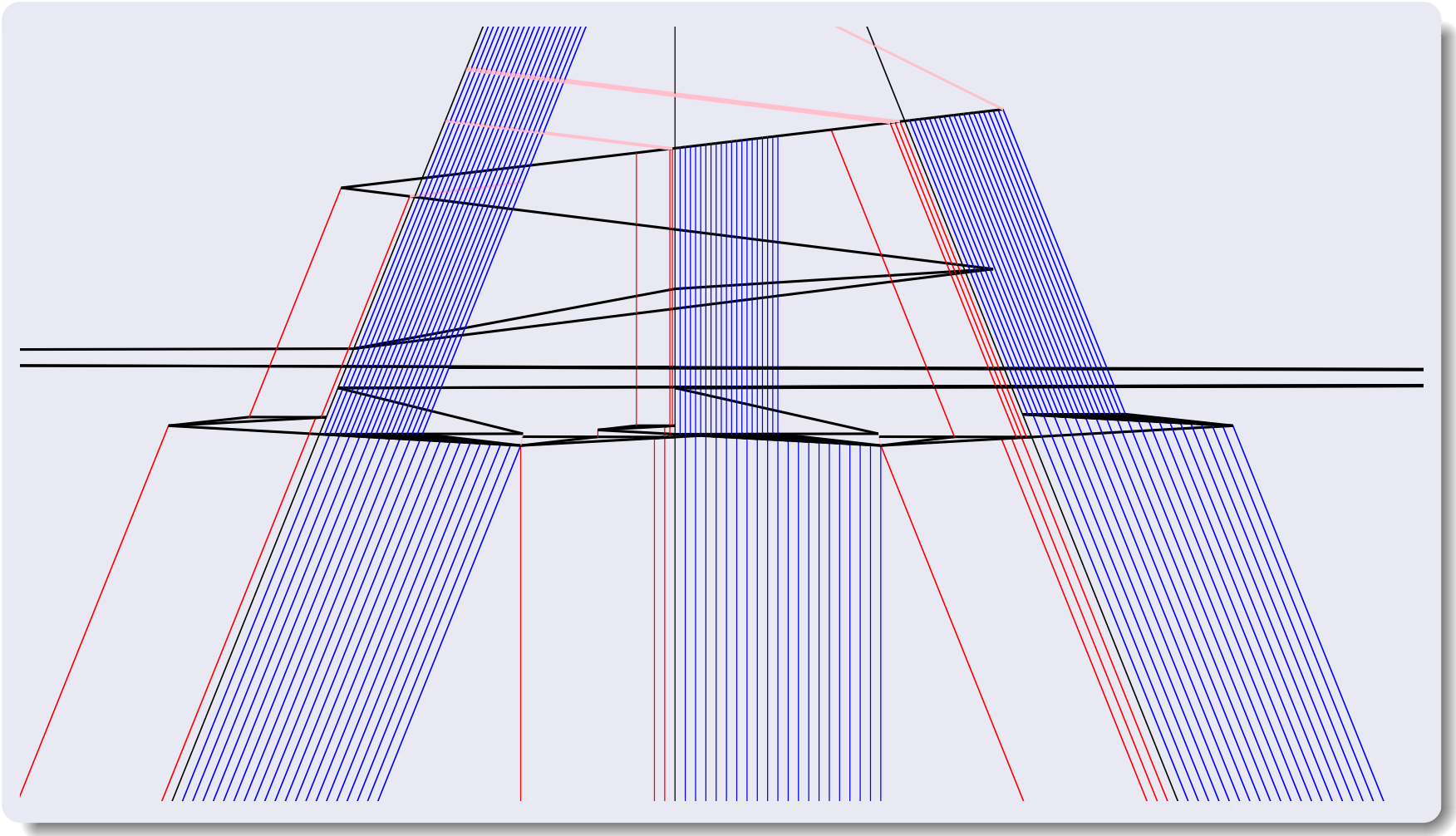
Good (?) Cases



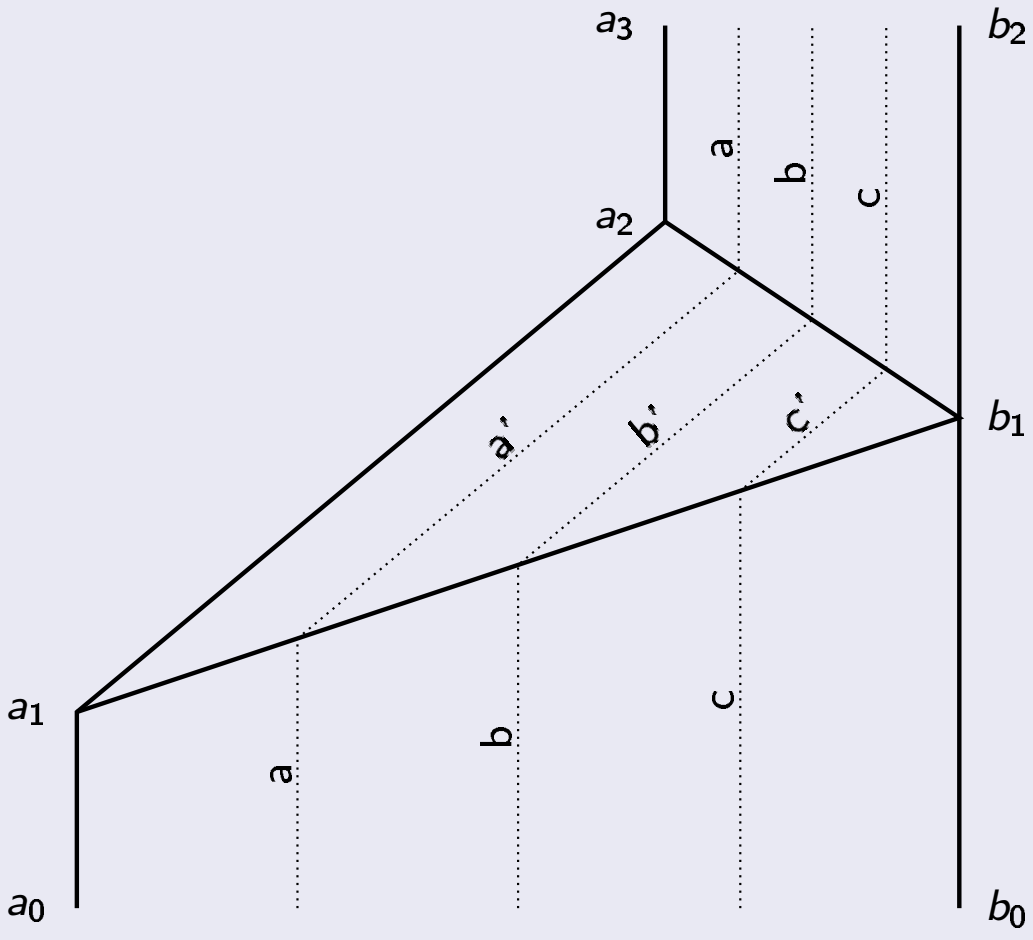
Bad Case



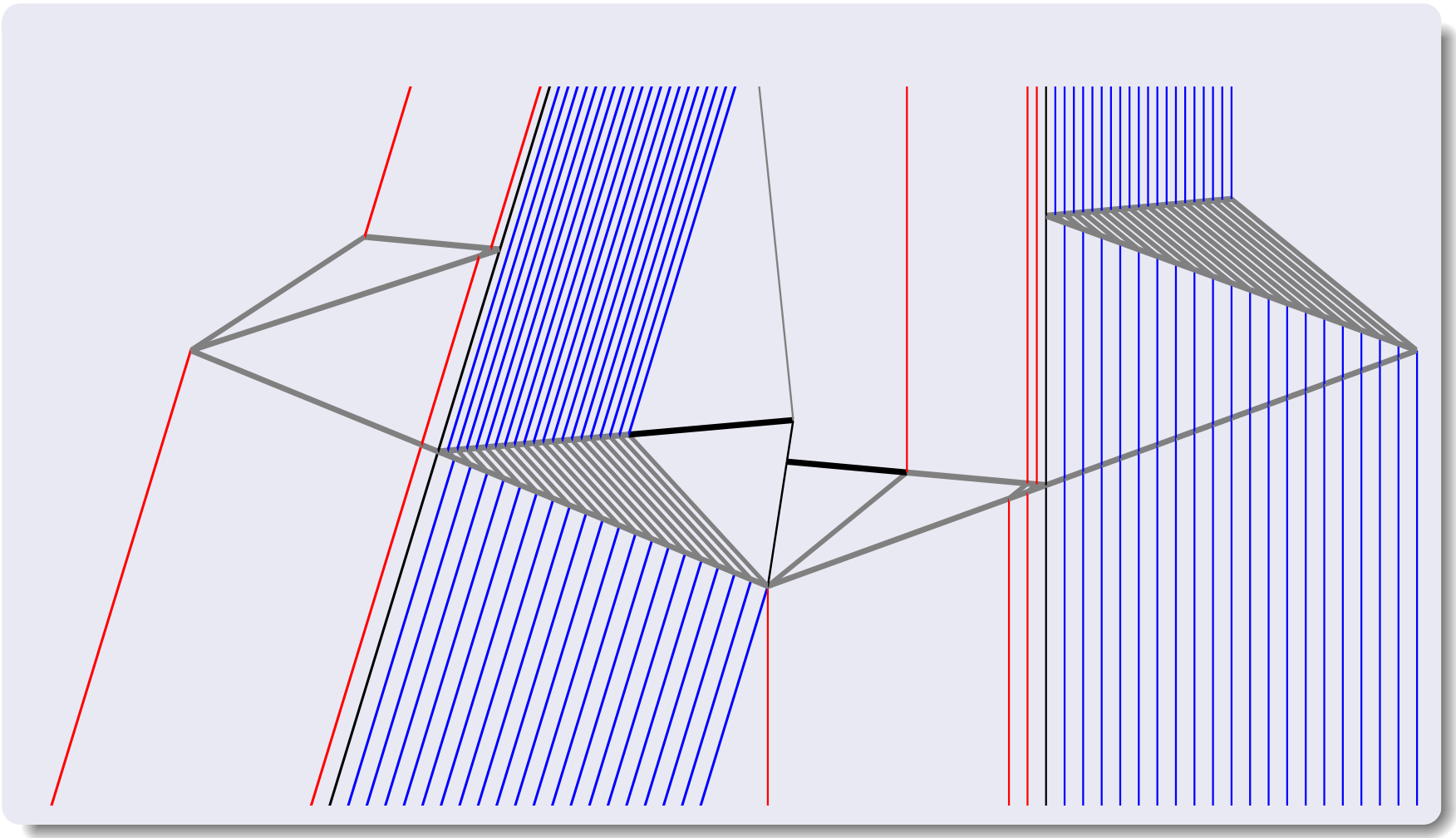
Whole Preparation (cropped on both side)



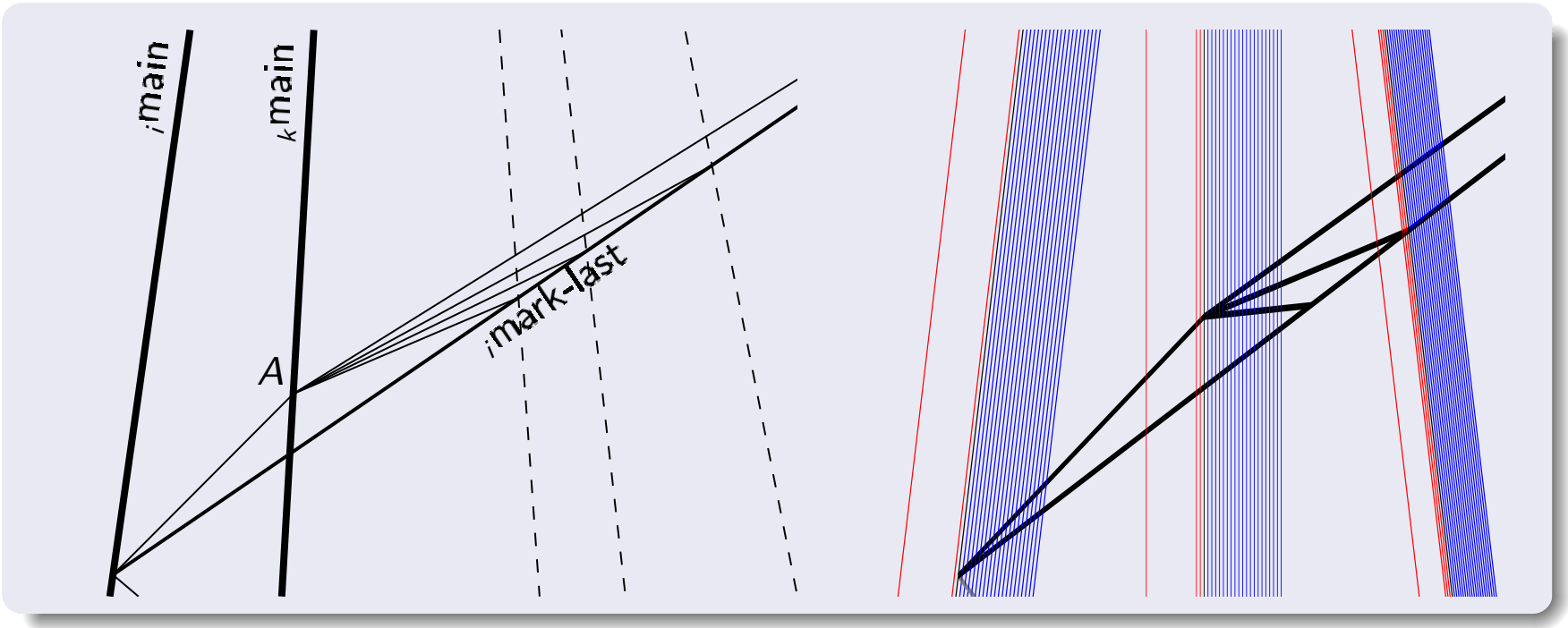
Shrinking Unit



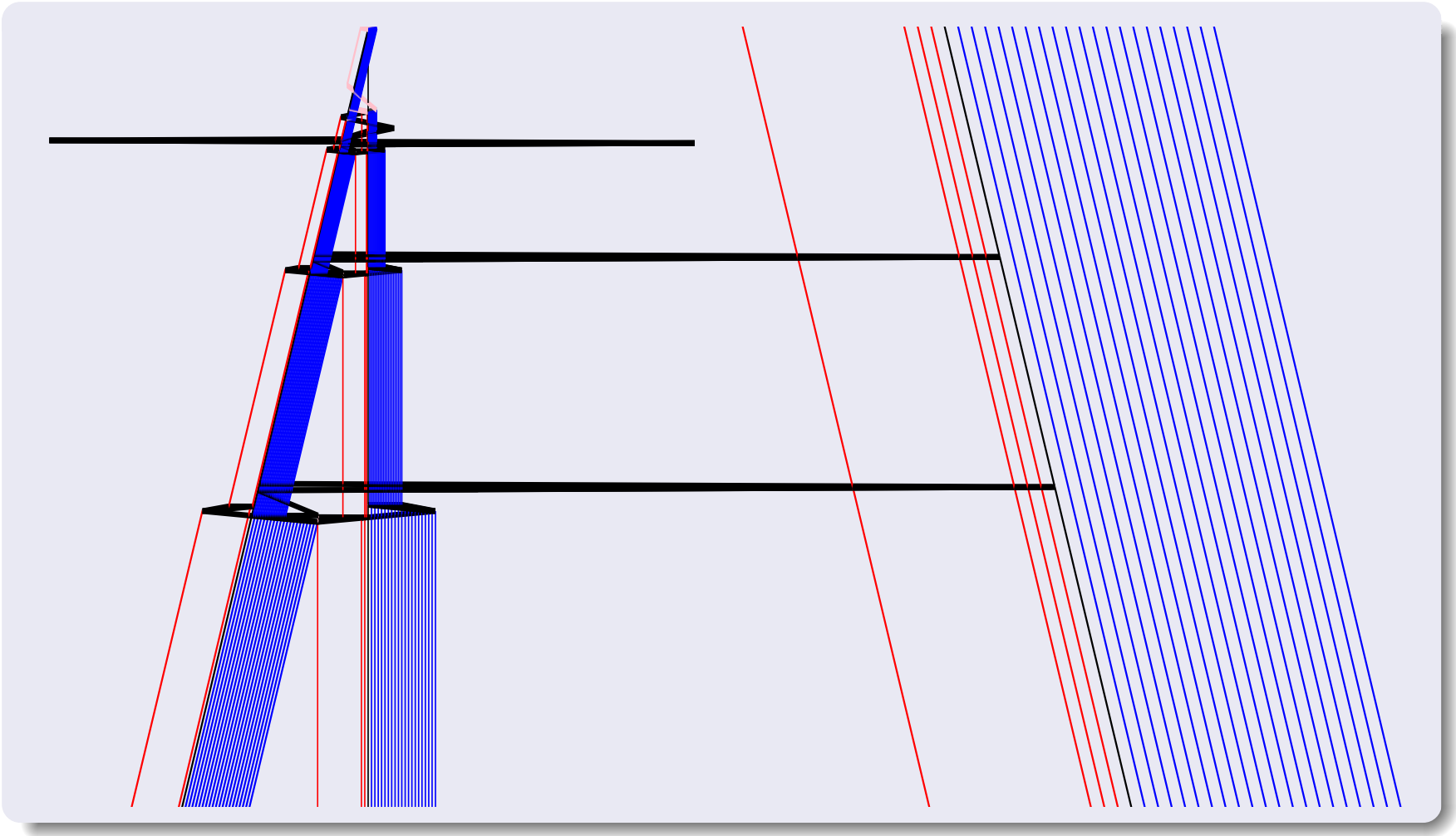
Shrink



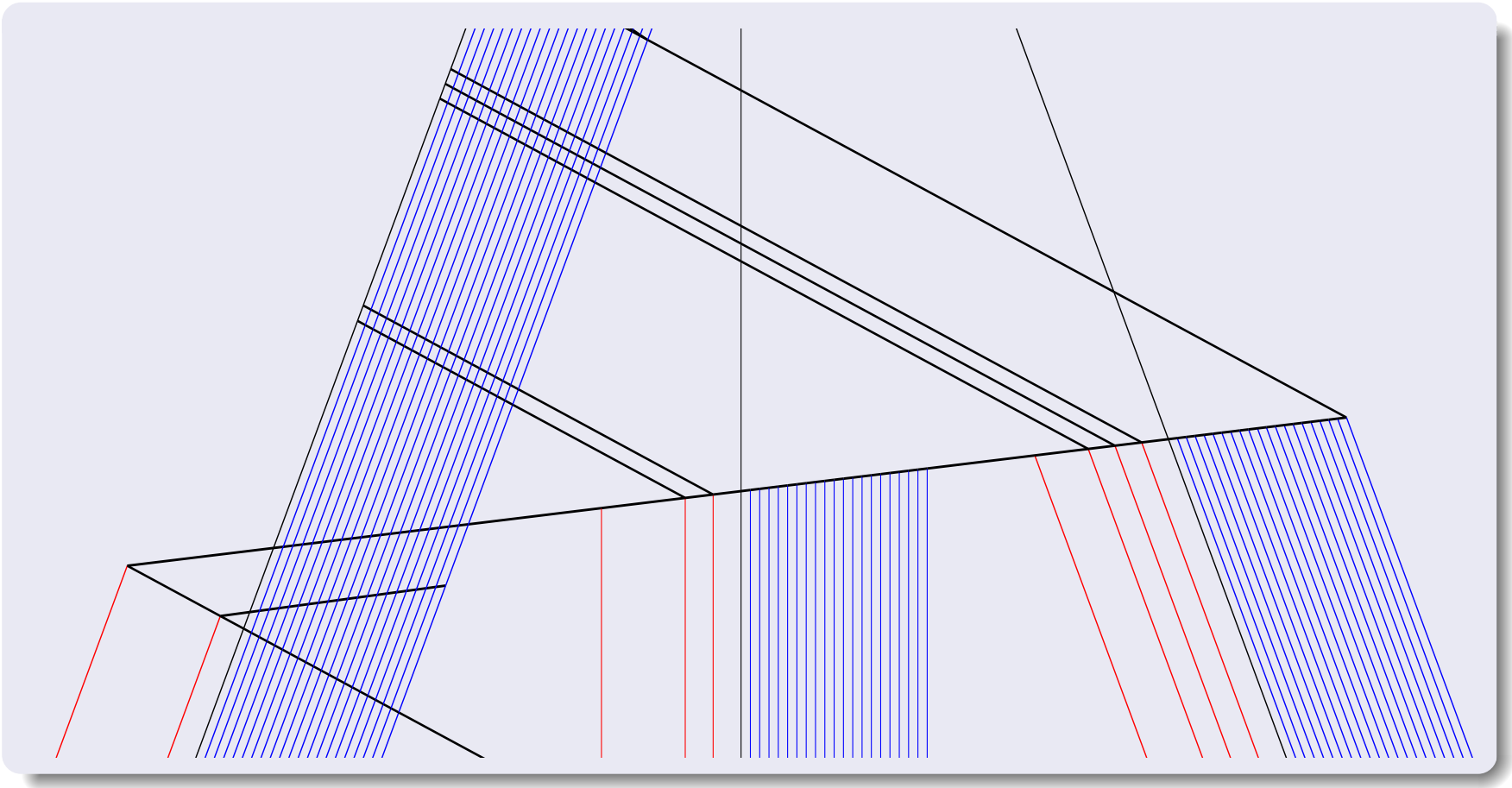
Testing for Other main Signals



Detecting Potential Overlaps



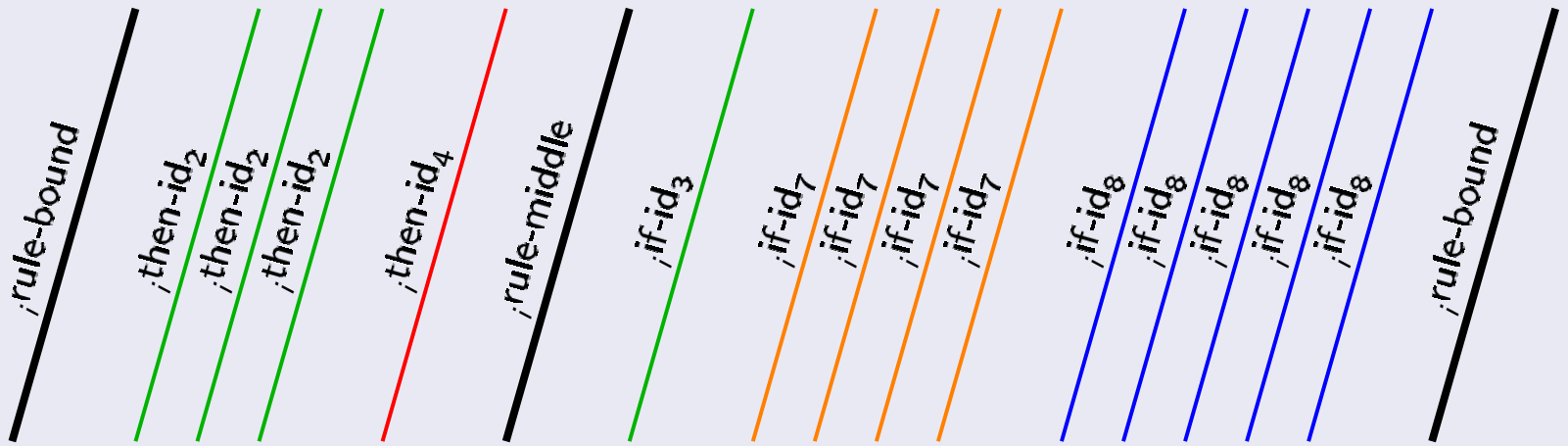
Removing Unused Tables and Sending ids to Table



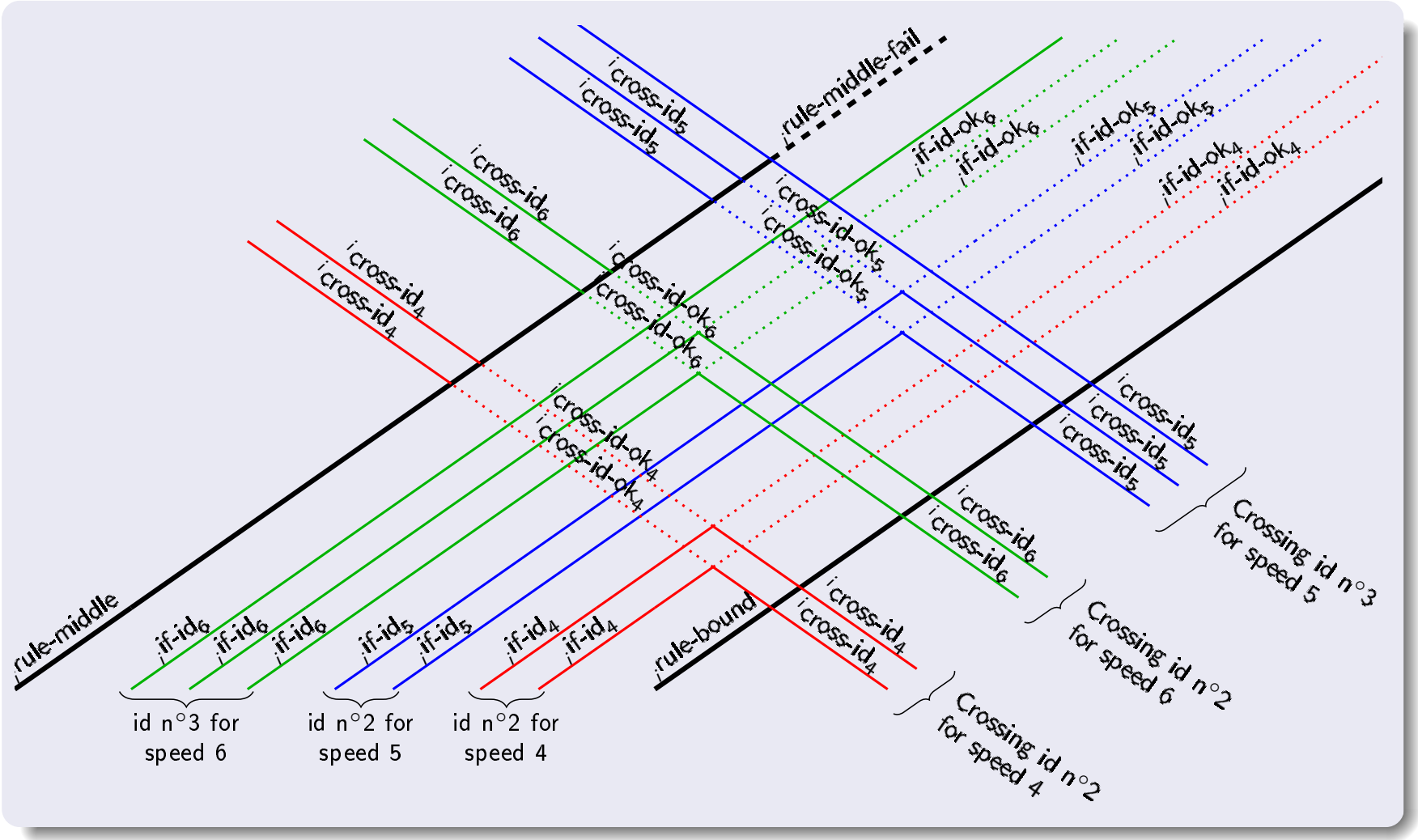
Collision Rules Encoding

One rule after the other

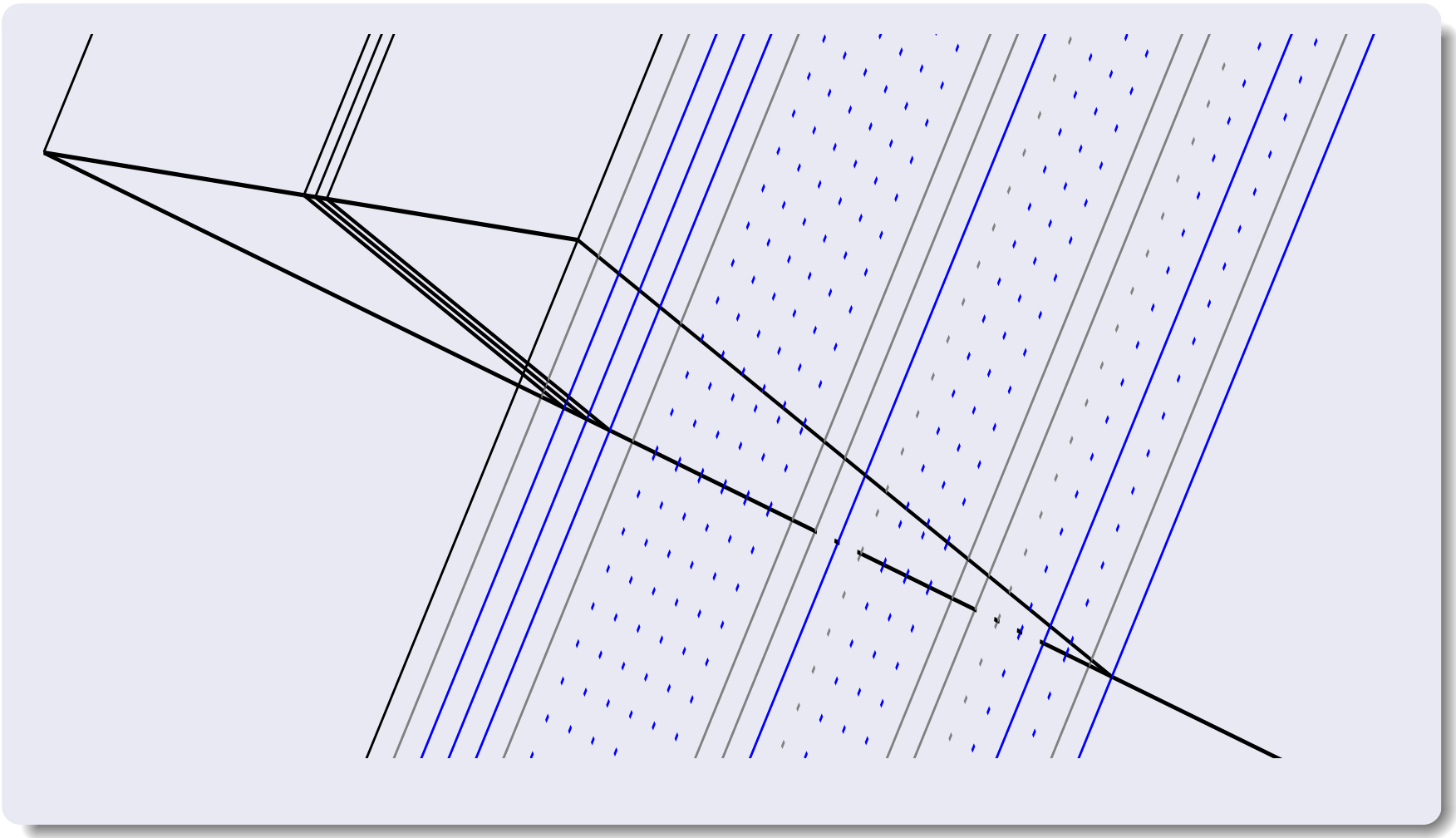
Encoding of $\{ {}_3\mu^1, {}_7\mu^4, {}_8\mu^5 \} \to \{ {}_2\mu^3, {}_4\mu^1 \}$ in the direction i.



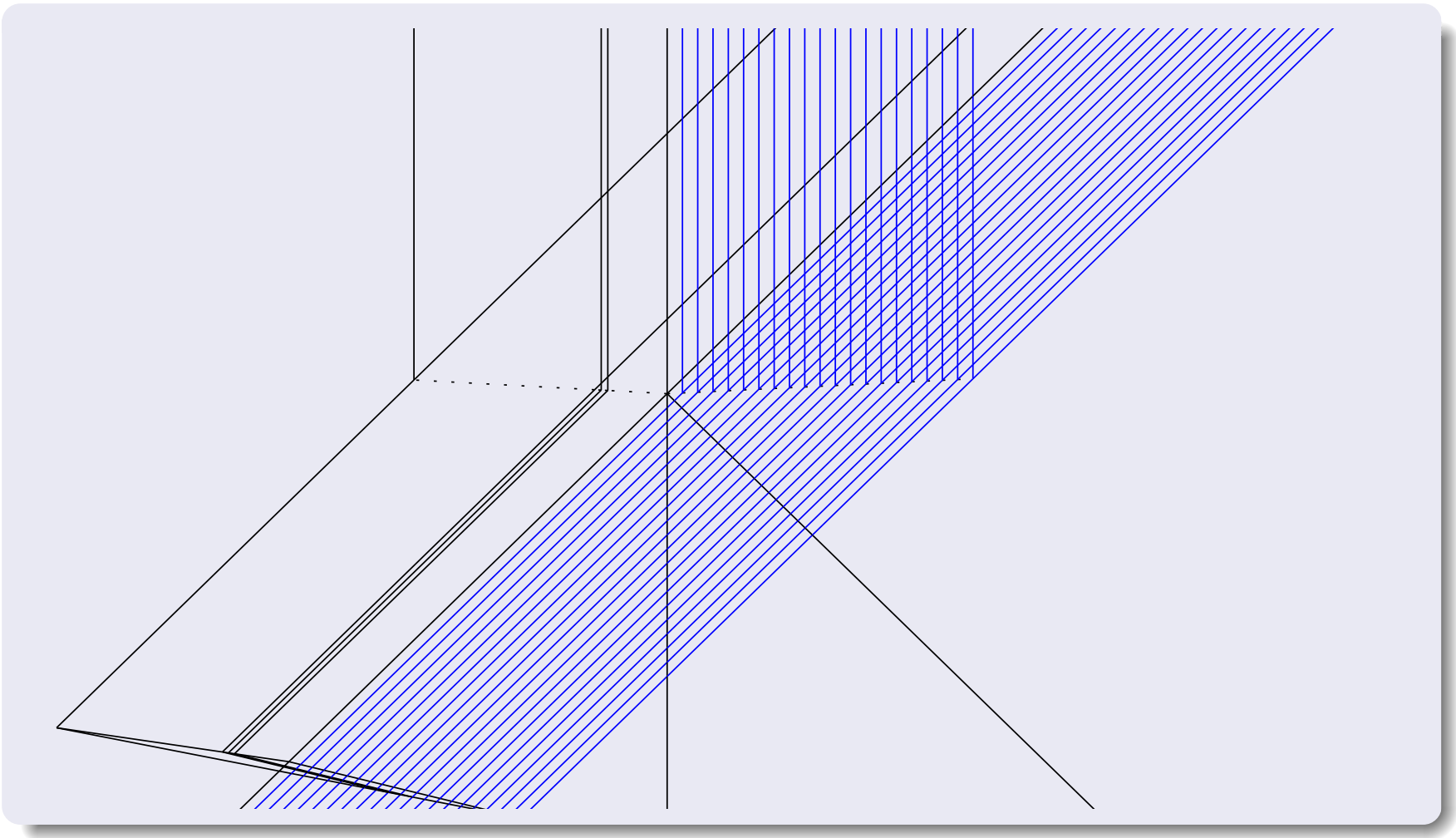
Comparison of id's in the if-part of a Rule



Rule Selection



Generating the Output



1. Compass and Rule (Huckenbeck, 1989, 1991)
2. Mondrian Automata (Jacopini and Sontacchi, 1990)
3. Piece-wise Constant Derivative (Asarin et al., 1995; Bournez, 1999)
4. Signal Machines
5. Intrinsically Universal Family of Signal Machines

Theorem

For any finite set of real numbers $\mathcal{S}$, there is a signal machine $\mathcal{U}_{\mathcal{S}}$, that can simulate any machine whose speeds belong to $\mathcal{S}$.

Theorem

The set of $\mathcal{U}_{\mathcal{S}}$ where $\mathcal{S}$ ranges over finite sets of real numbers is an intrinsically universal family of signal machines.

- Very rich setting
- Many models

Use of simple primitives

- parallel lines
- finding the middle

- Classical computation
- Analog computation

Continuity of space and time + idealization

- Hyper-computation

Thank you for your attention (and your participation)

Albert, J. and Čulik II, K. (1987). A Simple Universal Cellular Automaton and its One-Way and Totalistic Version. *Complex Systems*, 1:1–16.

Asarin, E., Maler, O., and Pnueli, A. (1995). Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoret Comp Sci*, 138(1):35–65.

Bournez, O. (1999). Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy. *Theoret Comp Sci*, 210(1):21–71.

Das, R., Crutchfield, J. P., Mitchell, M., and Hanson, J. E. (1995). Evolving globally synchronized cellular automata. In Eshelman, L. J., editor, *International Conference on Genetic Algorithms '95*, pages 336–343. Morgan Kaufmann.

Duchier, D., Durand-Lose, J., and Senot, M. (2011). Solving Q-SAT in bounded space and time by geometrical computation. In Ganchev, H., Löwe, B., Normann, D., Soskov, I., and Soskova, M., editors, *Models of computability in contecxt, 7th Int. Conf. Computability in Europe (CiE '11) (abstracts and handout booklet)*, pages 76–86. St. Kliment Ohridski University Press, Sofia University.

Duchier, D., Durand-Lose, J., and Senot, M. (2012). Computing in the fractal cloud: modular generic solvers for SAT and Q-SAT variants. In

Agrawal, M., Cooper, B. S., and Li, A., editors, *Theory and Applications of Models of Computations (TAMC '12)*, number 7287 in LNCS, pages 435–447. Springer.

Durand-Lose, J. (1997). Intrinsic Universality of a 1-Dimensional Reversible Cellular Automaton. In *STACS 1997*, number 1200 in LNCS, pages 439–450. Springer.

Fischer, P. C. (1965). Generation of primes by a one-dimensional real-time iterative array. *J ACM*, 12(3):388–394.

Goto, E. (1966). Ōtomaton ni kansuru pazuru [Puzzles on automata]. In Kitagawa, T., editor, *Jōhōkagaku eno michi [The Road to information science]*, pages 67–92. Kyoristu Shuppan Publishing Co., Tokyo.

Huckenbeck, U. (1989). Euclidian geometry in terms of automata theory. *Theoret Comp Sci*, 68(1):71–87.

Huckenbeck, U. (1991). A result about the power of geometric oracle machines. *Theoret Comp Sci*, 88(2):231–251.

Jacopini, G. and Sontacchi, G. (1990). Reversible parallel computation: an evolving space-model. *Theoret Comp Sci*, 73(1):1–46.

Mazoyer, J. and Rapaport, I. (1998). Inducing an Order on Cellular Automata by a Grouping Operation. In *15th Annual Symposium on*

Theoretical Aspects of Computer Science (STACS 1998), volume 1373 of *LNCS*, pages 116–127. Springer.

Meunier, P., Patitz, M. J., Summers, S. M., Theyssier, G., Winslow, A., and Woods, D. (2014). Intrinsic Universality in Tile Self-Assembly Requires Cooperation. In Chekuri, C., editor, *25th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA*, pages 752–771. SIAM.

Ollinger, N. (2001). Two-States Bilinear Intrinsically Universal Cellular Automata. In *FCT '01*, number 2138 in LNCS, pages 369–399. Springer.

Woods, D. (2013). Intrinsic Universality and the Computational Power of Self-Assembly. In Neary, T. and Cook, M., editors, *Proceedings Machines, Computations and Universality 2013, MCU 2013, Zürich, Switzerland*, volume 128 of *EPTCS*, pages 16–22.