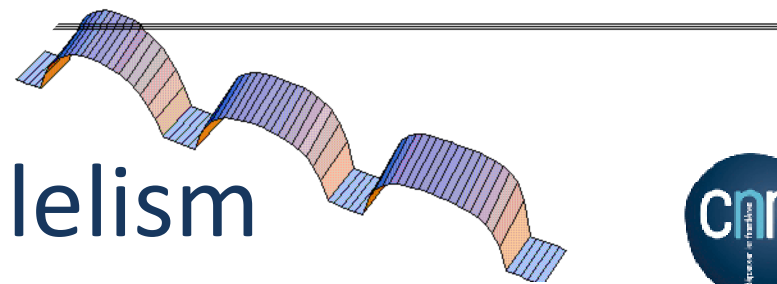
Spatial Computing
as
Intensional Data Parallelism



Antoine Spicher
Olivier Michel
Jean-Louis Giavitto
http://mgs.spatial-computing.org

IBISC
Computer Science, Integrative Biology & Complex Systems
CNRS - University of Evry - Genopole

TOC

- Motivations : data-parallelism and spatial computing
- Intensionnal Spatial Operations
- Dataflow
- Examples
- Compilation
- Conclusions

Data Parallelism

- Parallelism and Spatial Computing:

  if two computations occur simultaneously,
  they must take place at different location

  ⇒ **taking space into account**

- Parallelism as an operational *vs* a semantic property

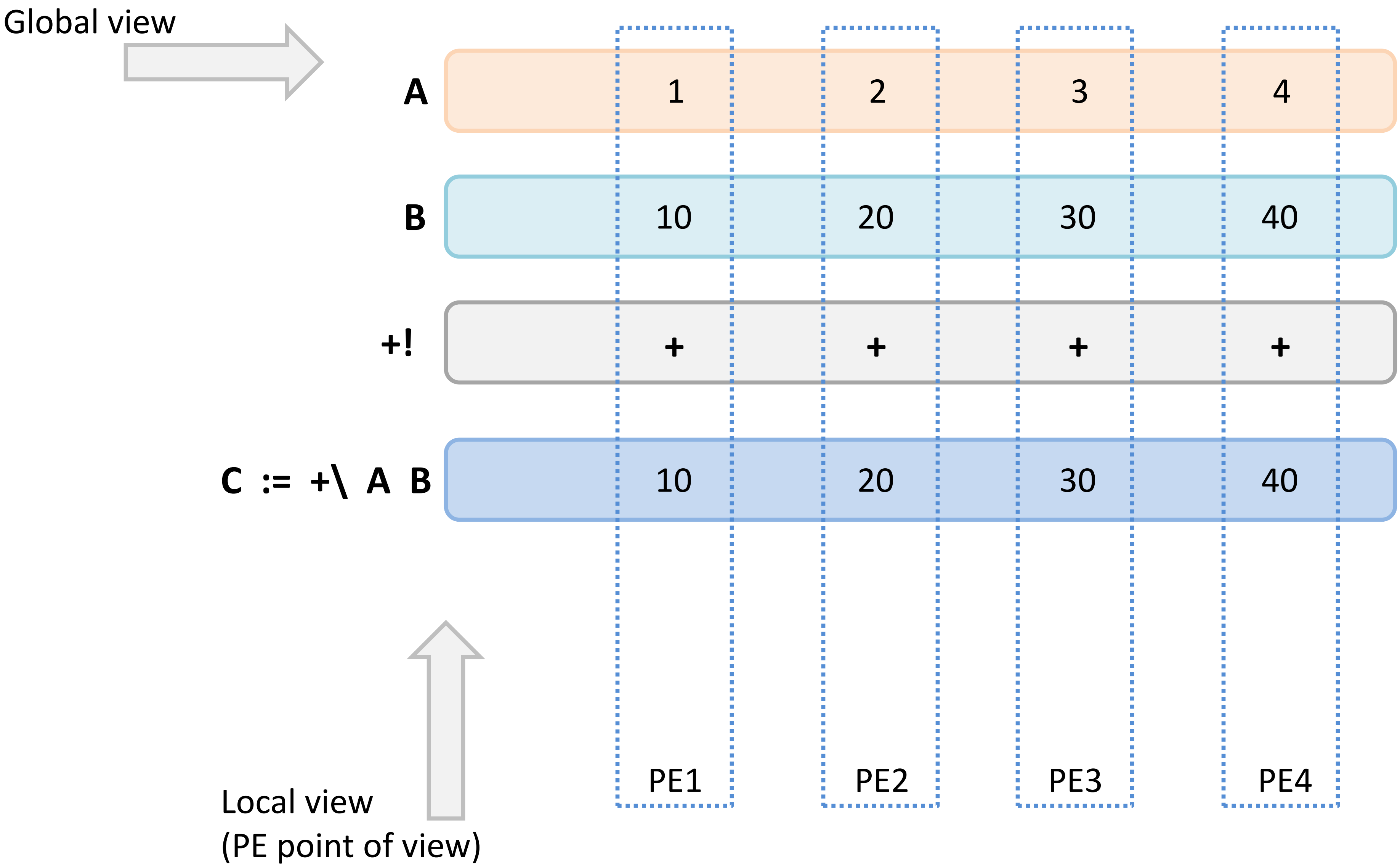
- Three ways to express parallelism :
- - parallelism is expressed through the data: data parallelism
- - parallelism is expressed through the control: control parallelism
- - parallelism is expressed through a mix of data and control: pipe-line

- An alternative classification:

	0 INSTRUCTION COUNTER **Declarative languages**	1 INSTRUCTION COUNTER **Sequential languages**	*n* INSTRUCTIONS COUNTER **Concurrent languages**
SCALAR	SISAL, ID, LAU, Actors	Fortran, Pascal, C	Adda, Occam
COLLECTION	Gamma, 81/2, MGS, PROTO	APL *Lisp, HPF, CMFortran	CMFortran + multi-threadings

The global (spatial, intensional) *vs.* the local (PE) view

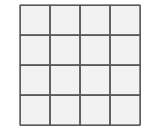
(intensional point of view on spatially distributed objects and processes)



From Arrays, Data Fields and GBF to Chain

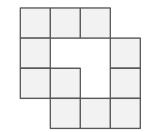
ARRAY
(Total Function)

$[0, d_1] \times \ldots \times [0, d_n] \longmapsto Val$



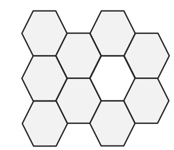
Data Field
(Partial Function)

$Z^n \longrightarrow Val$



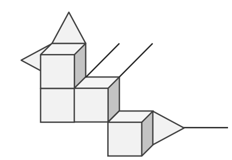
GBF (Group based Fields)
(Partial Function)

$Group \longrightarrow Val$



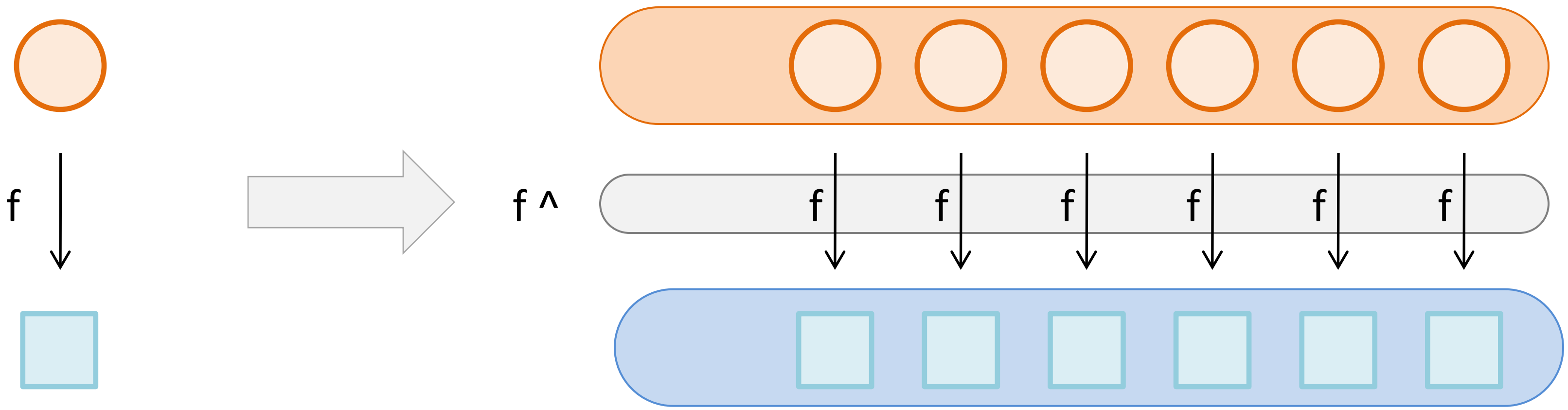
Chain:
(Partial Function)

$Cellular\ complex \longrightarrow Val$



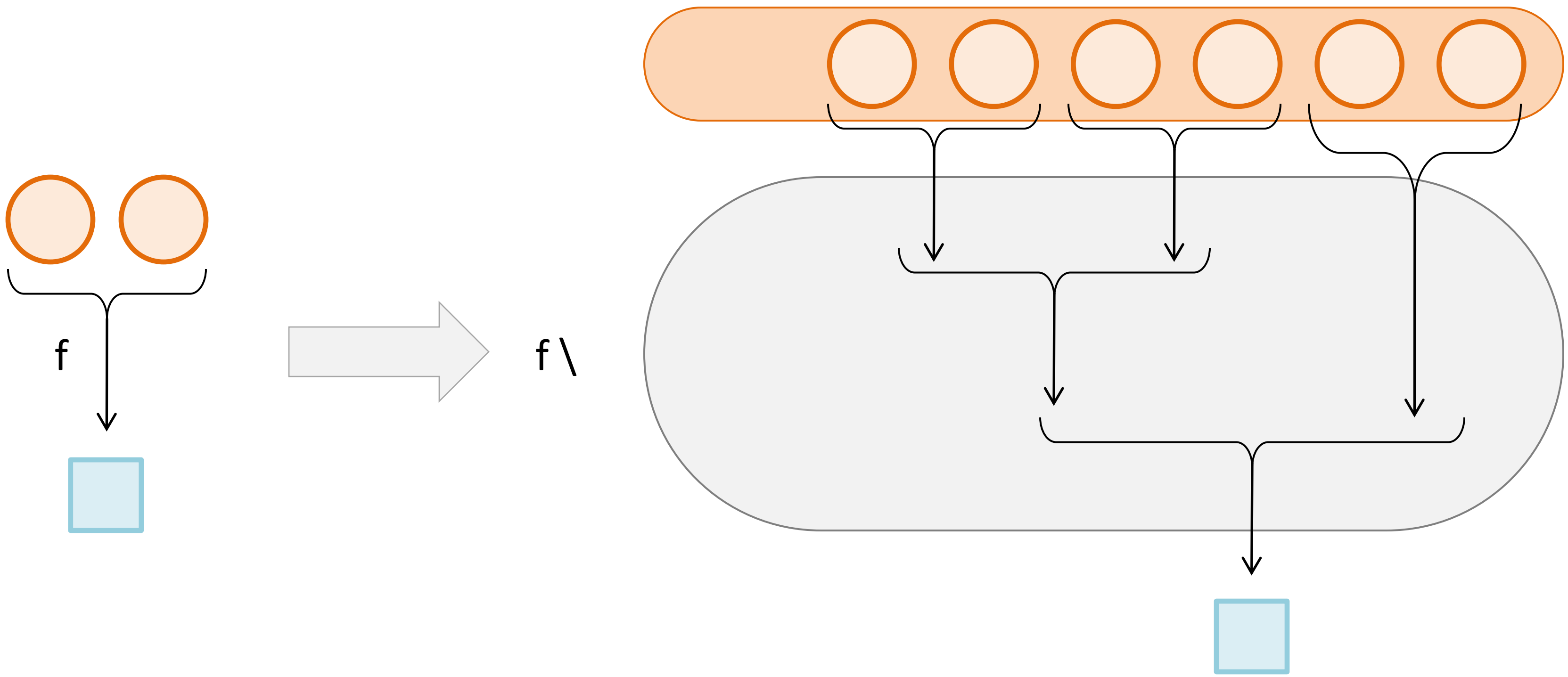
Intentional operations

- Alpha extension



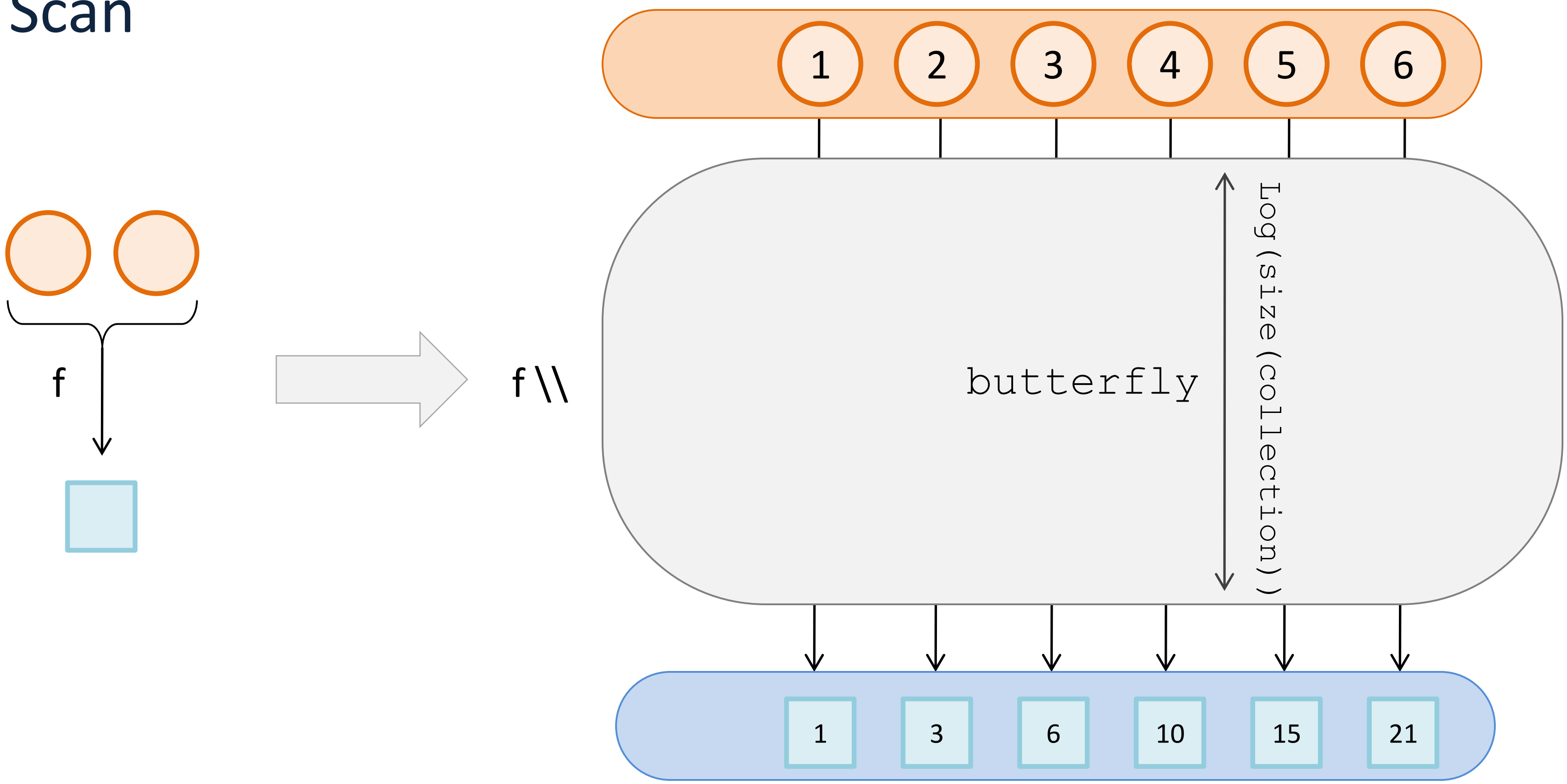
Intentional operations

- Alpha extension
- Beta reduction



Intentional operations

- Alpha extension
- Beta reduction
- Scan



Declarative definition

- A 8,5 program is a set of definitions:

$$\left\{\begin{array}{l} A = B + C \\ C = (\max \backslash B) * (+\backslash\backslash\ B) \\ B[4] = +\backslash\backslash\ (!1) \end{array}\right. \Rightarrow \left\{\begin{array}{l} A = [4, 14, 27, 44] \\ C = 4 * [1, 3, 6, 10] = [4, 12, 24, 40] \\ B = [1, 2, 3, 4] \end{array}\right.$$

- Definitions can be recursive

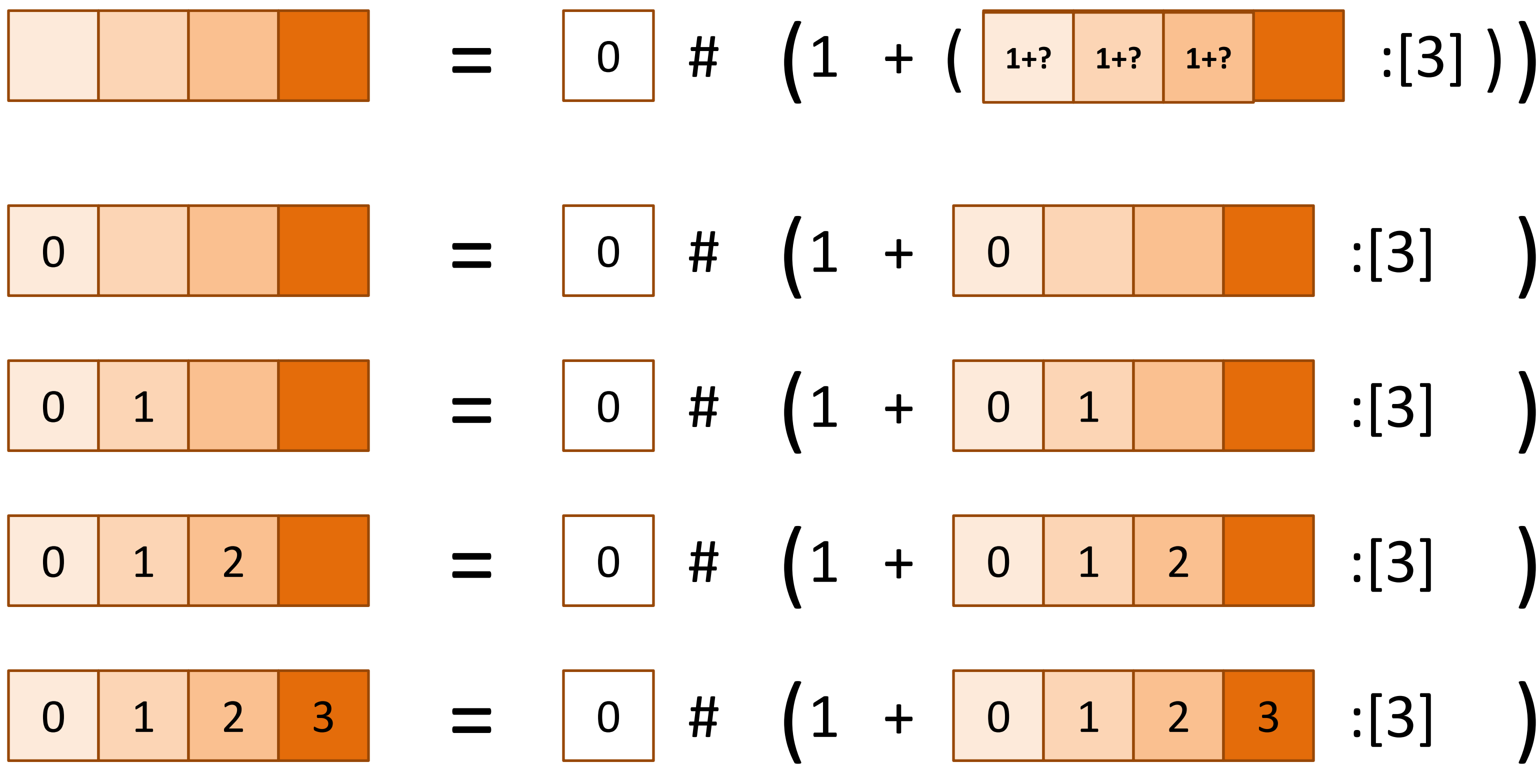
  X = 0 # (1 + x:[3])

  where

  - constant are polymorphic
  - # is the cutconcatenation
  - :[] is the cut operation

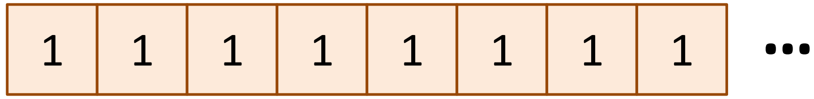
Solving a recursive definition

- Infer the geometry
- Check that the solution is *a priori* maximal
- Compute the solution by (a smart) fixed point iteration

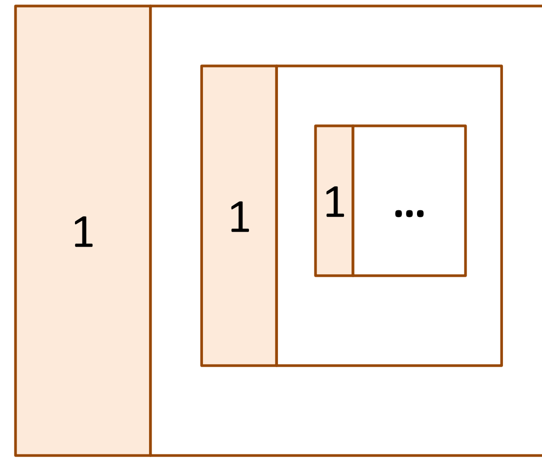


Inferring the geometry

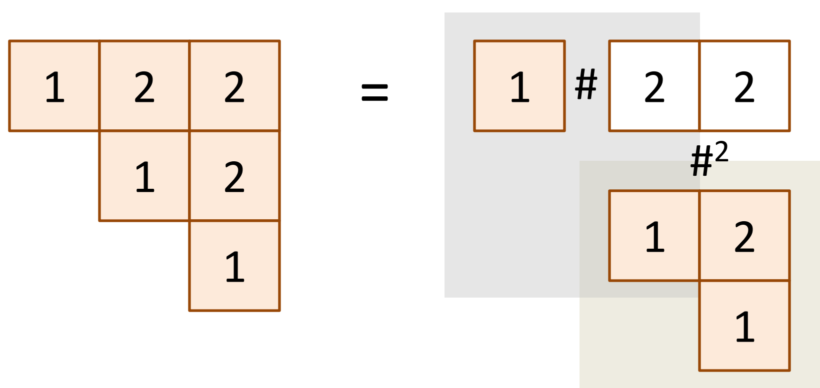
A = 1 # A



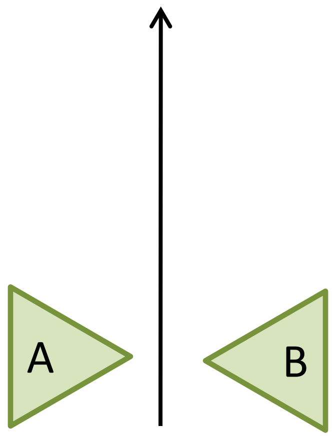
B = [1, B]



C = 1 # (2 $\#^2$ C:[2])



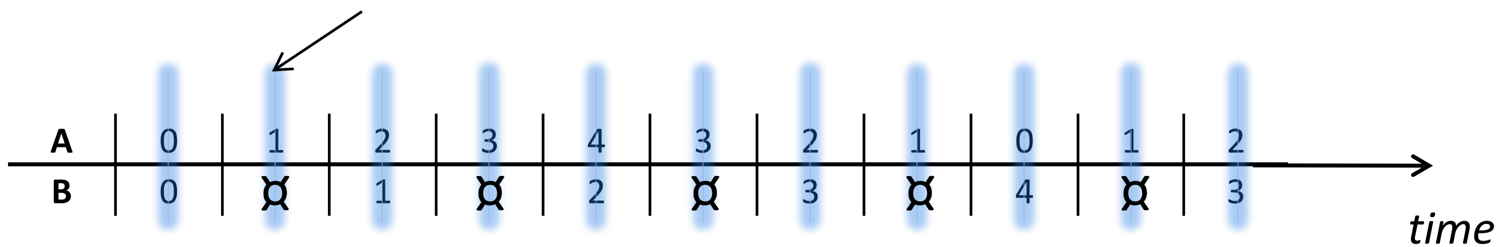
Declarative control : stream



A = 0, 1, 2, 3, 4, 3, 2, 1, 0, 1, 2, 3 …

B = 0, 1, 2, 3, 4, 3, 2, 1, 0, 1, 2, 3 …

Introduction of *hiaton*: from data flow to synchronous data flow

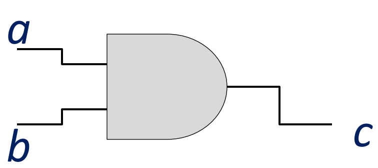


clock(B) = true, false, true, false, true, false, true, false, true, false, true, false …

tick… **tock …**

A logic of signal vs. a logic of state

A logic of signal



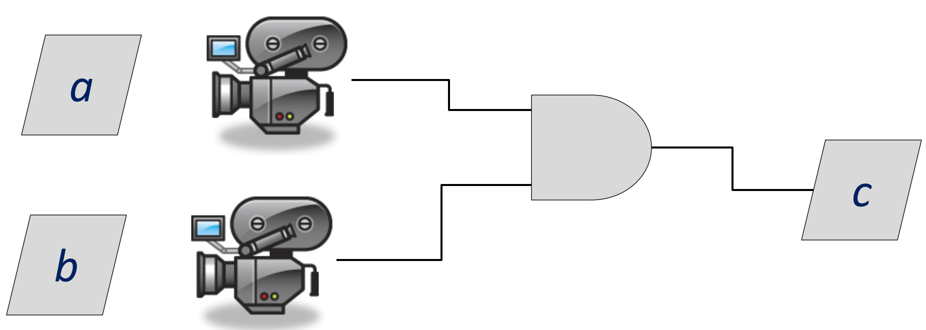
A	1		1	1		1	1			1	1
B	2		2		2	2			2		2
A = B + C	3		3			3					3

ERROR

An input is missing

A logic of state

A	1		1	1		1	1			1	1
B	2		2		2	2			2		2
A = B + C	3		3	3	3	3	3		3	3	3



OK

The result is the combination of the last seen values.
A result is computed if there is a change in one input.

Synchronous stream algebra

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	...
1	1									...
1+2	3									...
Clock 2	$true$		$true$		$true$		$true$		$true$	...
assuming A	1		2	3		4	5	6		...
assuming B		1		2			1		1	...
C = A+B		2	3	5		6	6	7	7	...
$ C			2	3		5	6	6	7	...

A	1	2	3	4	5	6	7	8	9	...
B	$false$	$false$	$false$	$true$	$false$	$true$	$true$	$false$	$true$	...
A when B				4		6	7		9	...

Recursive stream definitions

- $X = \$X + 1$ ⇒ ∅(the empty stream)
  Hint : what is the initial value of the stream ?

- $X@0 = 1$
  $X = \$X + 1$ ⇒

1

Hint : at which pace the counter increase ?

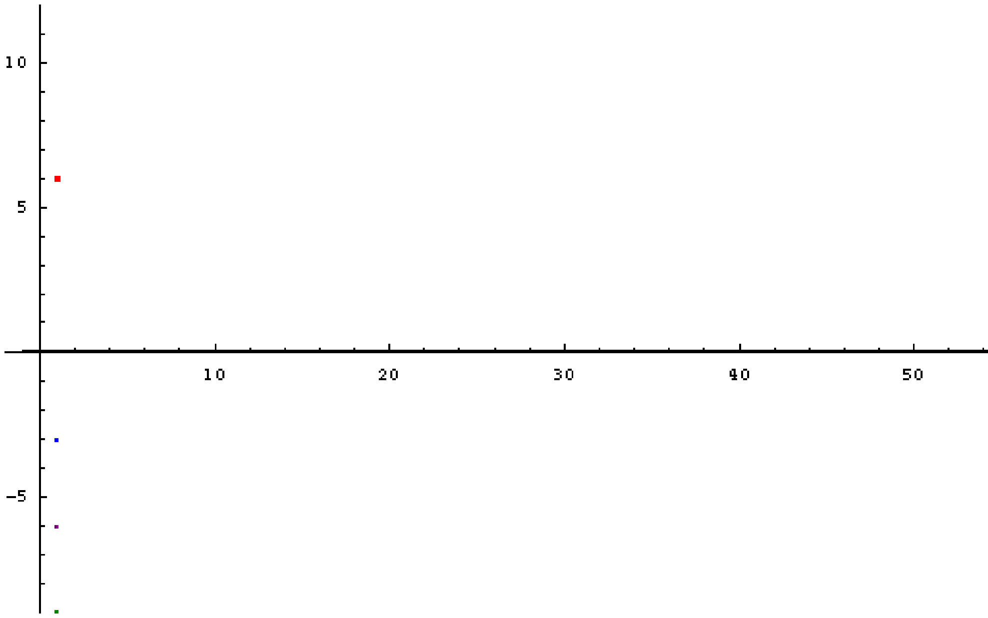
- $X@0 = 0$
  $X = \$X + 1$ when Clock1 ⇒

1	2	¤	4	¤	…
	t	f	t	f	…

clock(Clock1)

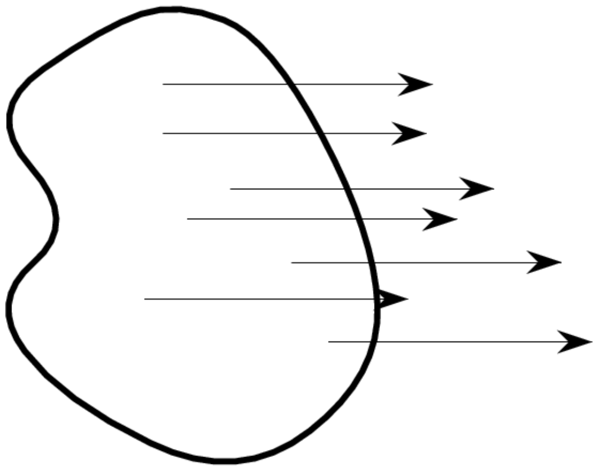
The wlumf : a reactive animat

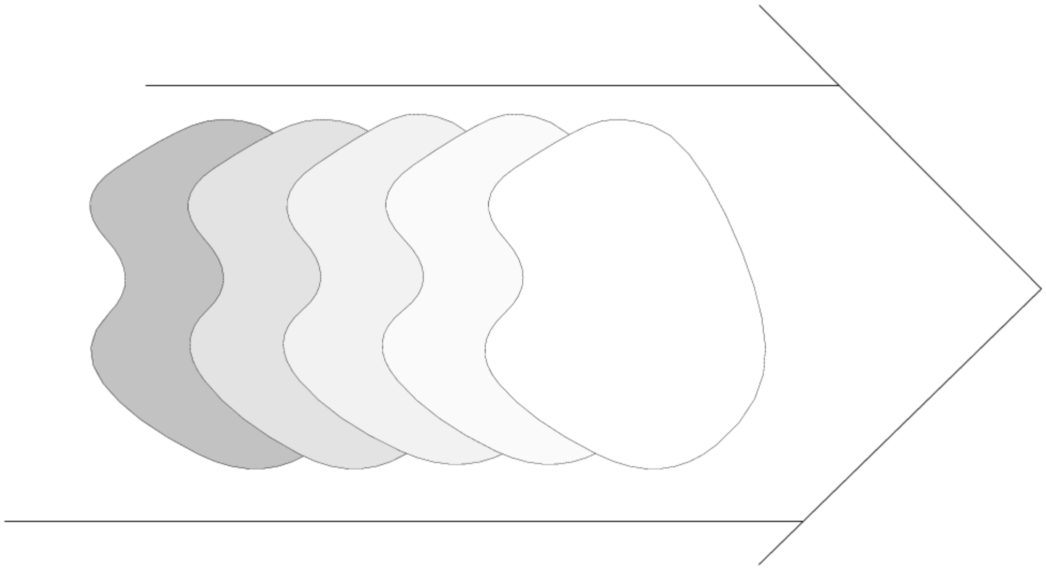
```
System wlumf = {

    glycemia@0 = 6;
    glycemia = if eating
               then 12
               else max(0, $glycemia -1)when Clock

    eating@0 = false;
    eating = $hungry && environment.food;

    hungry@0 = false;
    hungry = (glycemia < 6);
}
```




```
System Environment = {
    food = ((t%2) == 0);
    t@0 = 0;
    t = $t+1 when Clock(-2);
}
```

Fabric = stream(collection) = collection(stream)

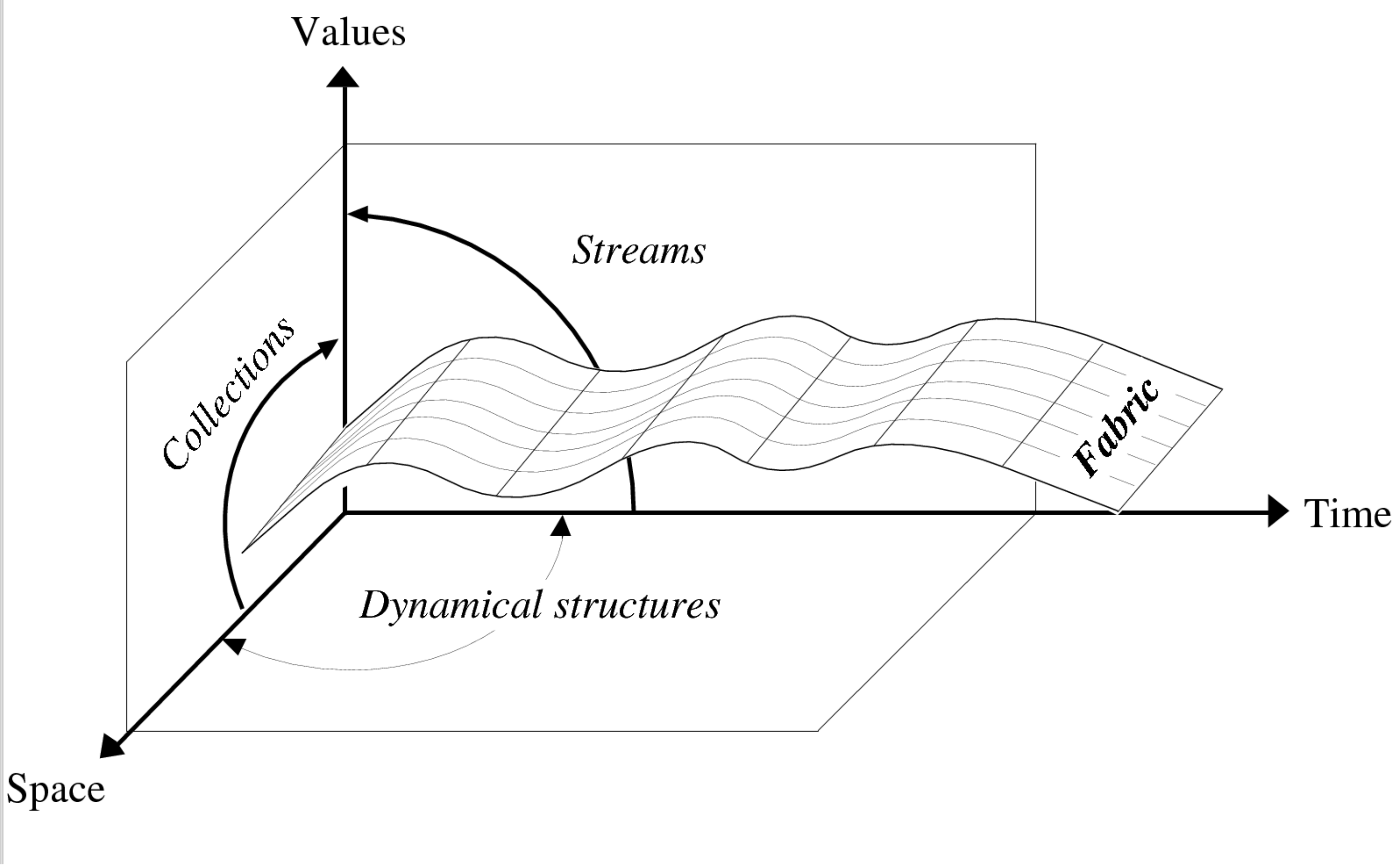




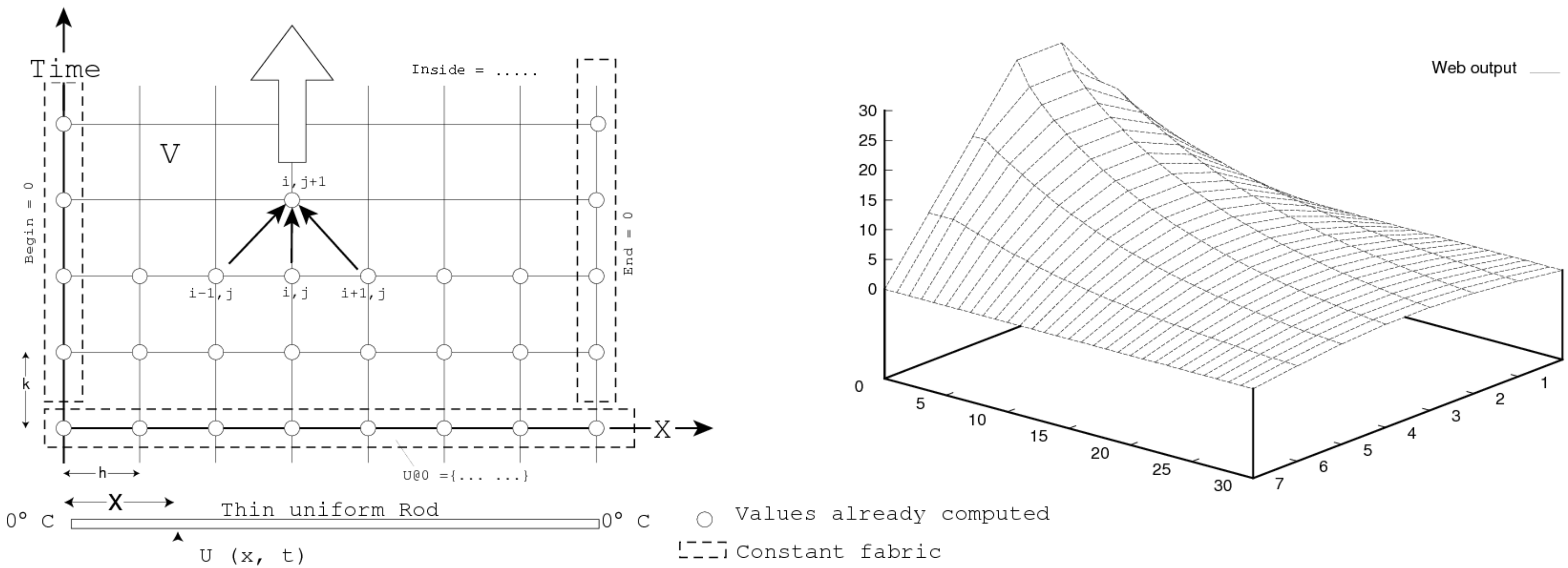
Fabric = stream of collection = collection of stream

(for static geometry)

Fabric = a “space-time” data



Heat diffusion in a thin rod



U@0 = …

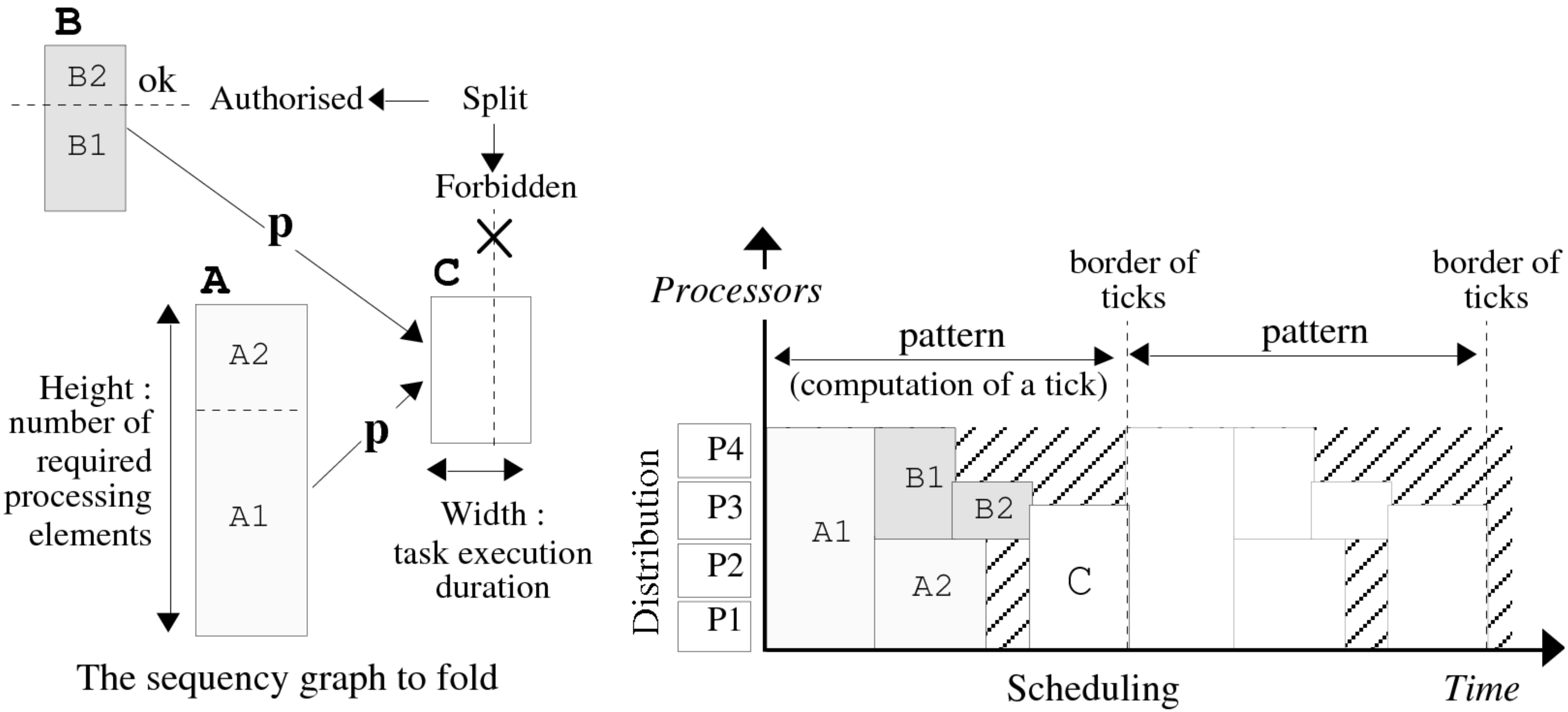
U = α(begin # inside):[n] + (1-2α)inside + α (inside # end):[-n]

inside = $U when Clock

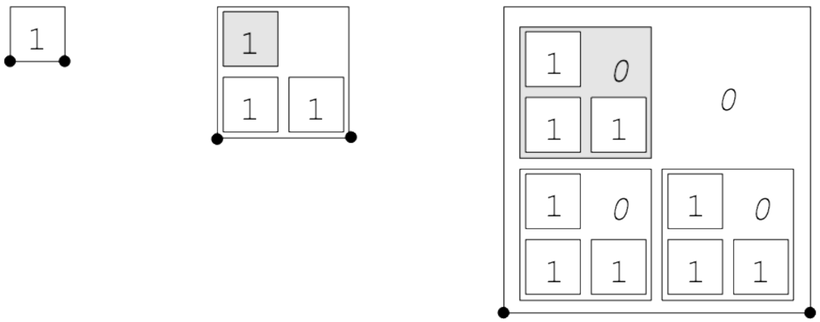
begin = 0

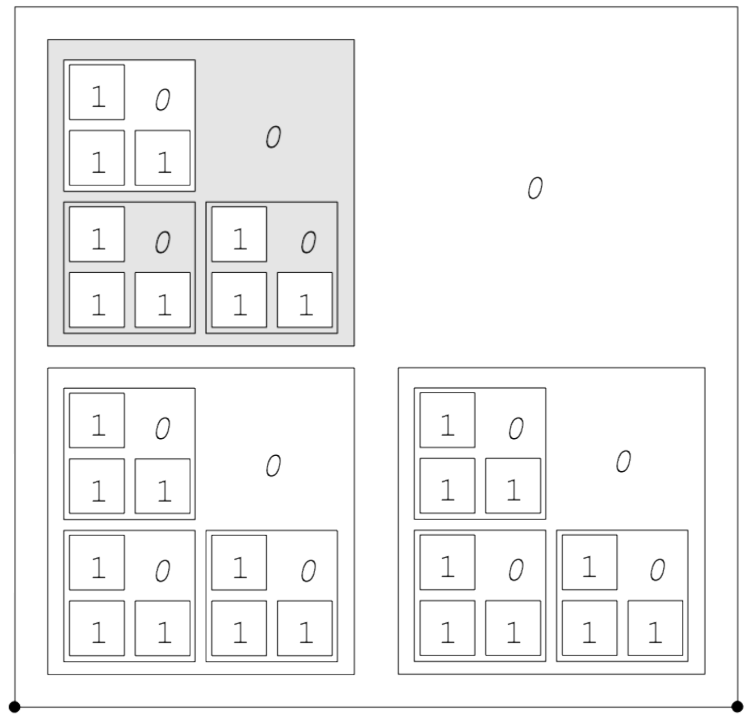
end = 0

Mapping and (cyclic) scheduling

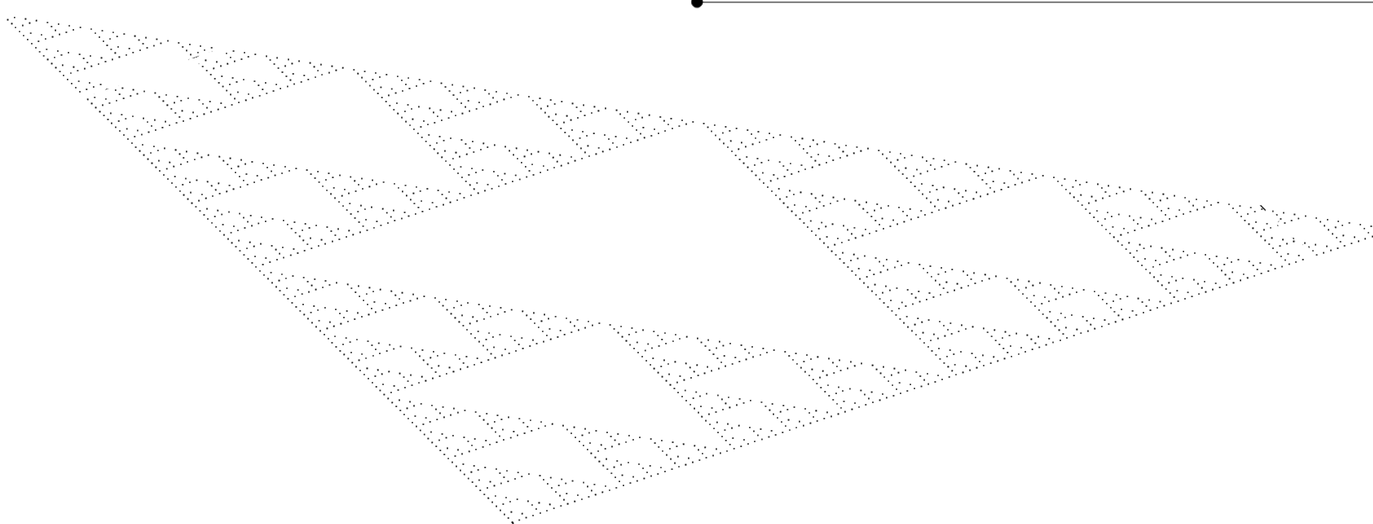


Example of a growing collection





$$
\begin{aligned}
p &= ((\$p \;\#\; 0 : |\$p|) \;\#\hat{}\; (\$p \;\#\; \$p)) \text{ when } Clock\,1 \\
p@0 &= 1 : [1, 1]
\end{aligned}
$$



Conclusions

- a C compiler to a sequential architecture
- Parallel mapping and scheduling on:
  - CM
  - MPI (paragon, network of workstation)
- efficient compilation if static

- Spatial computing: YES **but**
  - **Simple** model of underlying space (but can be extended)
  - **Synchronous** time:
    atomic, event-driven, synchronization costs
  - **Crystalline** computation
  - **Intensional** approach = working with *spatial object as a whole*
  - NO support for amorphous computing:
    - Locality can be enforced through a tailored set of operations
    - no robustness
    - Dynamic space are difficult to handle