Synchronous Modeling of Data-Intensive Applications

Huafeng. Yu, A. Gamatié, É. Rutten,
P. Boulet and J.-L. Dekeyser

{Yu, Gamatie, Boulet, Dekeyser}@lifl.fr
Eric.Rutten@inrialpes.fr
DART project, INRIA Futurs / WEST group, LIFL

This talk :

- Detailed presentation of ARRAY-OL/GASPARD
- First results of study

Plan

Introduction

Context : data-intensive applications (DIA) in embedded systems

- regular multidimensional data processing
- parallel processing in System-on-Chip (SoC)

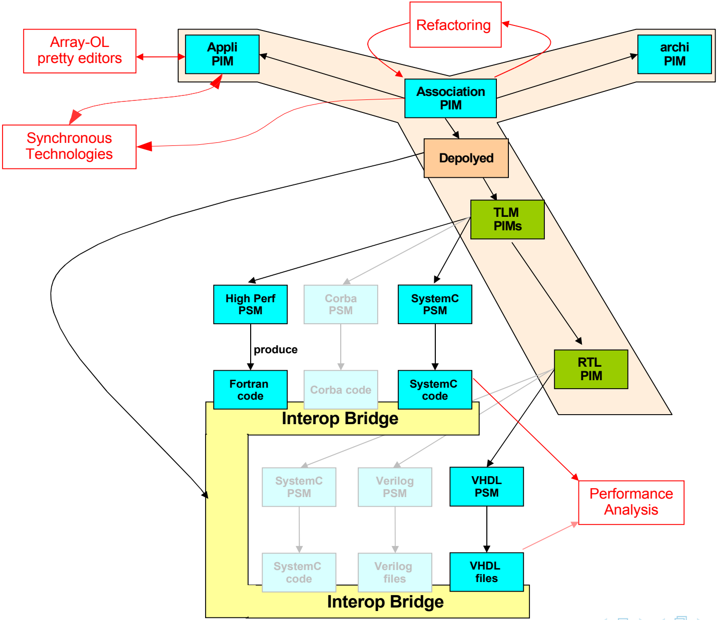
Motivations : adequate techniques for

- efficient data manipulation
- analysis of implementation properties

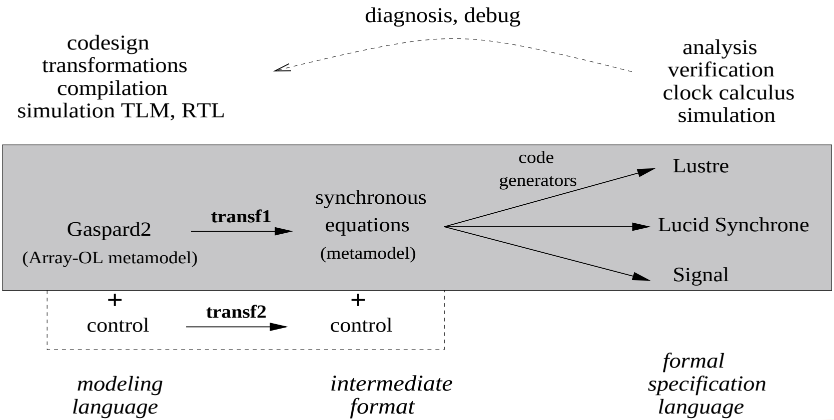
Approach : combination of

- a formalism dedicated to DIA (ARRAY-OL)
- data-flow synchronous equation models

GASPARD methodology



General scheme



Introduction

GASPARD methodology

General scheme

Data-intensive processing

ARRAY-OL language

Existing works

Simple synchronous modeling of GASPARD models

Parallel model

Serialized model

Validation issues

Conclusions

ARRAY-OL

ARRAY-OL (Array-Oriented Language) : initially proposed by Thomson Marconi Sonar [DD98]

- Specification language for full parallelism
- Data manipulation through arrays
- Deadlock free and deterministic by construction

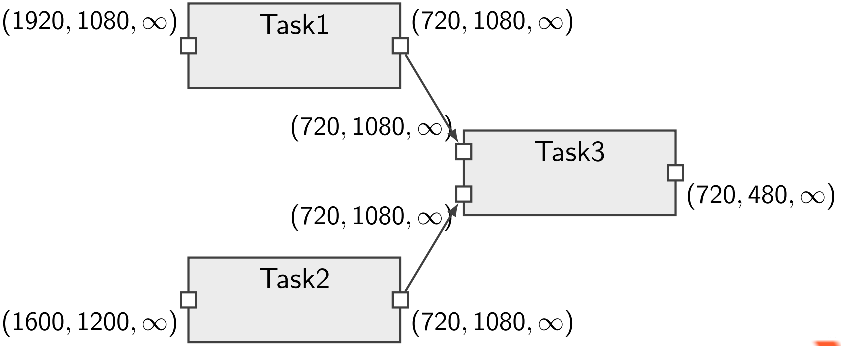
ARRAY-OL

ARRAY-OL (Array-Oriented Language) : initially proposed by Thomson Marconi Sonar [DD98]

- Specification language for full parallelism
- Data manipulation through arrays
- Deadlock free and deterministic by construction
- Descriptions independent from implementation platforms
- Two types of parallelism in application specifications : Task parallelism and Data parallelism

ARRAY-OL (cont'd)

Task parallelism and data dependencies :



ARRAY-OL (cont'd)

Different task models :

- Elementary task : atomic computation block (instantaneous function)

ARRAY-OL (cont'd)

Different task models :

- Elementary task : atomic computation block (instantaneous function)
- Hierarchical task : task represented by hierarchical acyclic graphs in which
  - each node consists of a task, and
  - edges are labeled by the arrays

Array-OL **(cont'd)**

Different task models :

- Elementary task : atomic computation block (instantaneous function)
- Hierarchical task : task represented by hierarchical acyclic graphs in which
  - each node consists of a task, and
  - edges are labeled by the arrays
- Repetition task : expression of data parallelism

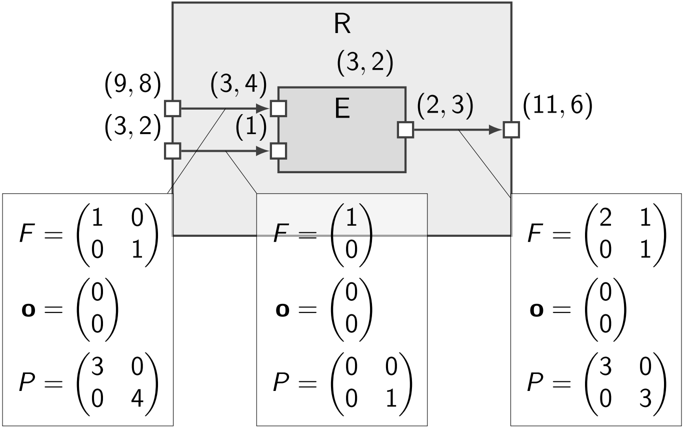
ARRAY-OL (cont'd)

Data parallelism

- Repetition element : the subtask to be repeated
- Repetition space : limitation of repetition number and link between inputs and outputs
- Interface : input and output arrays
- Tiler : defines how to obtain sub-arrays from a input array and how to store sub-arrays in a output array

ARRAY-OL (cont'd)

Example of a repetition task



$$F = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \mathbf{o} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad P = \begin{pmatrix} 3 & 0 \\ 0 & 4 \end{pmatrix}$$

$$F = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \mathbf{o} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad P = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

$$F = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix} \quad \mathbf{o} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad P = \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix}$$

ARRAY-OL (cont'd)

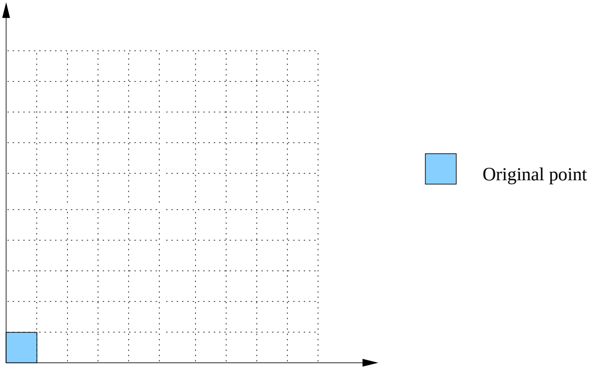
Tiler specification :

- o : original point of the array or reference pattern
- P : paving matrix (how the array is tiled by patterns)
- F : fitting matrix (how patterns are filled by array elements)

ARRAY-OL (cont'd)

Paving : how the array is tiled by patterns.
These patterns are calculated in any order.

Paving example : $o = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $P = \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}$

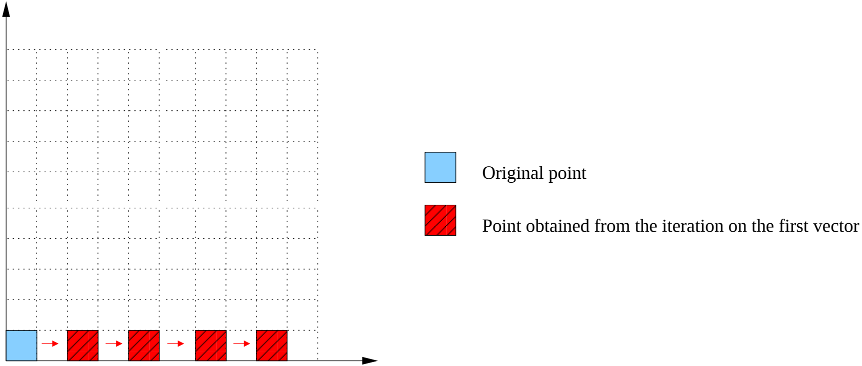


Repetition space : [5,4], limitation of pattern repetitions.

ARRAY-OL (cont'd)

Paving : how the array is tiled by patterns.
These patterns are calculated in any order.

Paving example : $o = \begin{pmatrix}0\\0\end{pmatrix}$, $P = \begin{pmatrix}2 & 0\\0 & 3\end{pmatrix}$

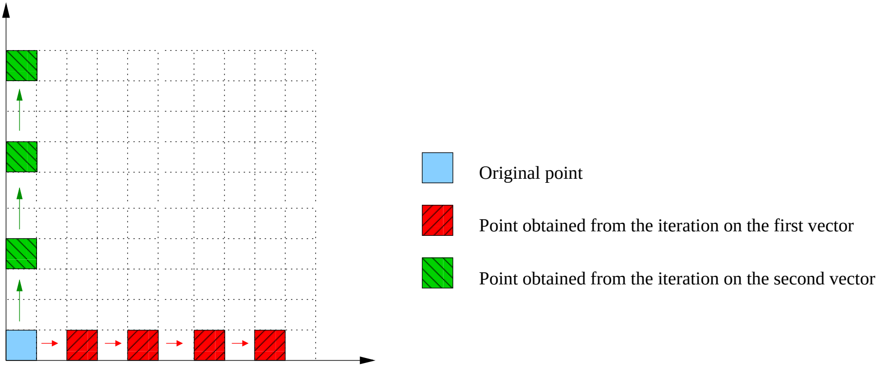


Repetition space : [5,4], limitation of pattern repetitions.

ARRAY-OL (cont'd)

Paving : how the array is tiled by patterns.
These patterns are calculated in any order.

Paving example : $o = \begin{pmatrix}0\\0\end{pmatrix}$, $P = \begin{pmatrix}2 & 0\\0 & 3\end{pmatrix}$

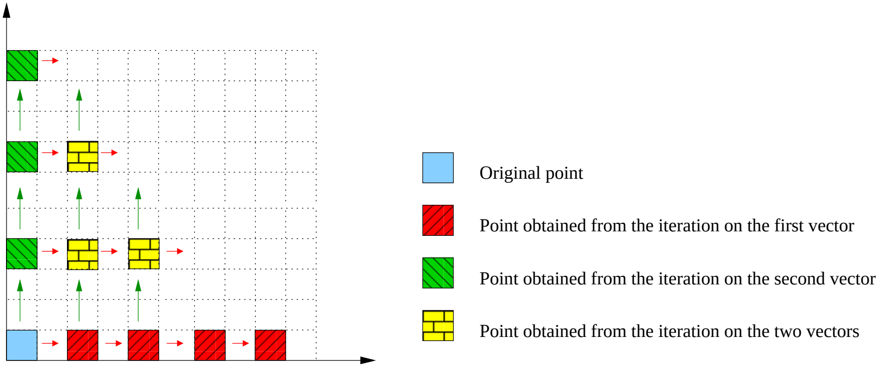


Repetition space : [5,4], limitation of pattern repetitions.

ARRAY-OL (cont'd)

Paving : how the array is tiled by patterns.
These patterns are calculated in any order.

Paving example : $o = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $P = \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}$



🟦 Original point

🟥 Point obtained from the iteration on the first vector

🟩 Point obtained from the iteration on the second vector

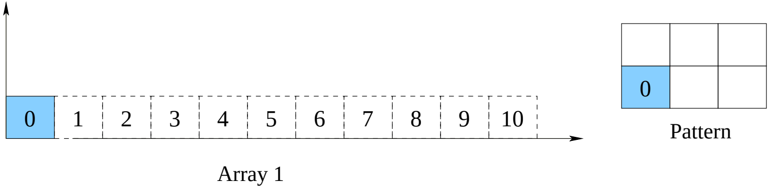
🟨 Point obtained from the iteration on the two vectors

Repetition space : [5,4], limitation of pattern repetitions.

ARRAY-OL (cont'd)

Fitting : how each pattern is filled by array elements.

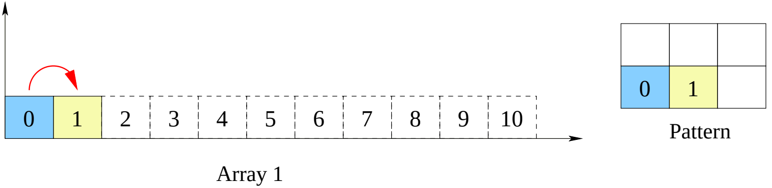
Fitting example 1 : $o = \left(\, 0 \,\right)$, $F = \left(\begin{smallmatrix}1\\3\end{smallmatrix}\right)$



ARRAY-OL (cont'd)

Fitting : how each pattern is filled by array elements.

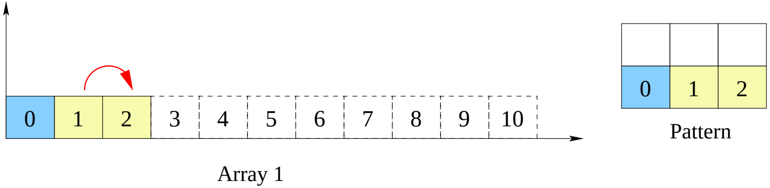
Fitting example 1 : $o = \begin{pmatrix} 0 \end{pmatrix}$, $F = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$



ARRAY-OL (cont'd)

Fitting : how each pattern is filled by array elements.

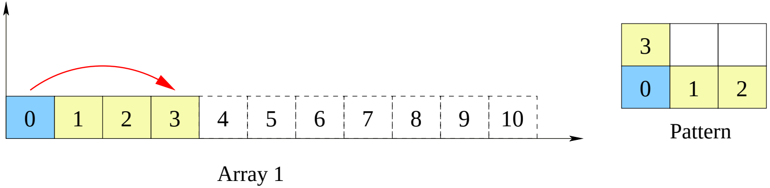
Fitting example 1 : $o = \left(\,0\,\right)$, $F = \left(\,\frac{1}{3}\,\right)$



ARRAY-OL (cont'd)

Fitting : how each pattern is filled by array elements.

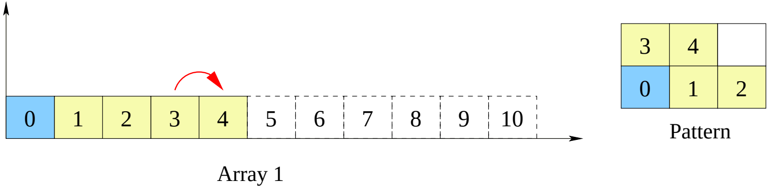
Fitting example 1 : $o = \begin{pmatrix} 0 \end{pmatrix}$, $F = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$



ARRAY-OL (cont'd)

Fitting : how each pattern is filled by array elements.

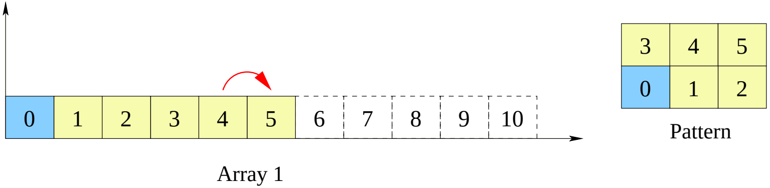
Fitting example 1 : $o = (\,0\,)$, $F = \left(\frac{1}{3}\right)$



ARRAY-OL (cont'd)

Fitting : how each pattern is filled by array elements.

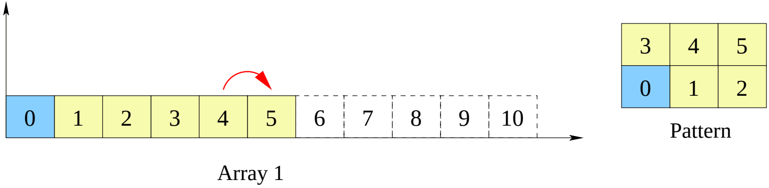
Fitting example 1 : $o = \begin{pmatrix} 0 \end{pmatrix}$, $F = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$



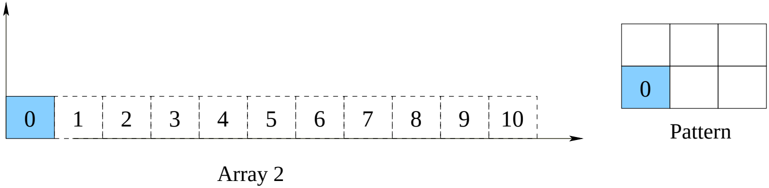
ARRAY-OL (cont'd)

Fitting : how each pattern is filled by array elements.

Fitting example 1 : $o = \begin{pmatrix} 0 \end{pmatrix}$, $F = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$



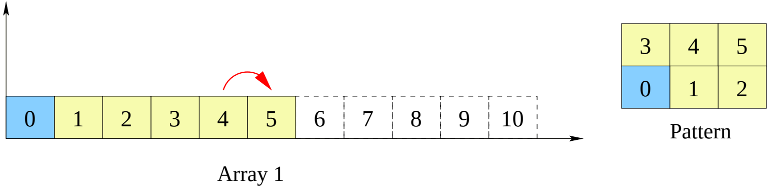
Fitting example 2 : $o = \begin{pmatrix} 0 \end{pmatrix}$, $F = \begin{pmatrix} 2 \\ 6 \end{pmatrix}$



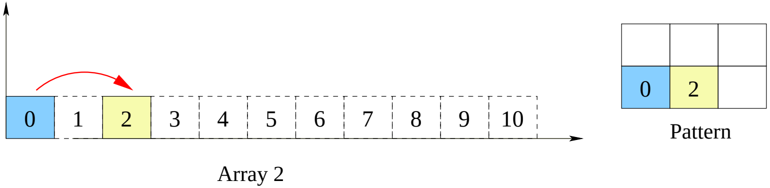
ARRAY-OL (cont'd)

Fitting : how each pattern is filled by array elements.

Fitting example 1 : $o = (0)$, $F = \begin{pmatrix}1\\3\end{pmatrix}$



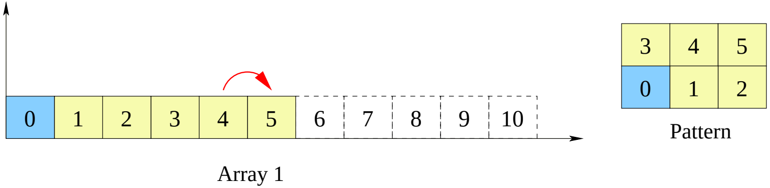
Fitting example 2 : $o = (0)$, $F = \begin{pmatrix}2\\6\end{pmatrix}$



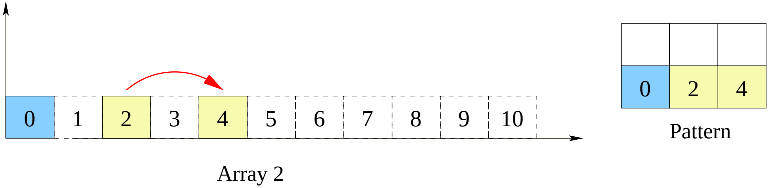
ARRAY-OL (cont'd)

Fitting : how each pattern is filled by array elements.

Fitting example 1 : $o = \begin{pmatrix} 0 \end{pmatrix}$, $F = \begin{pmatrix} 1 & 3 \end{pmatrix}$



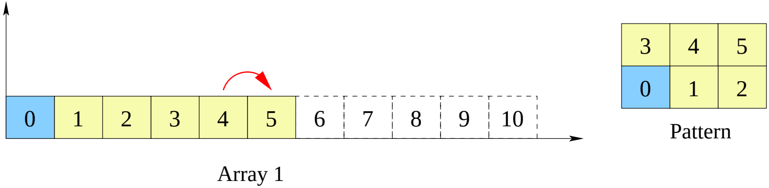
Fitting example 2 : $o = \begin{pmatrix} 0 \end{pmatrix}$, $F = \begin{pmatrix} 2 & 6 \end{pmatrix}$



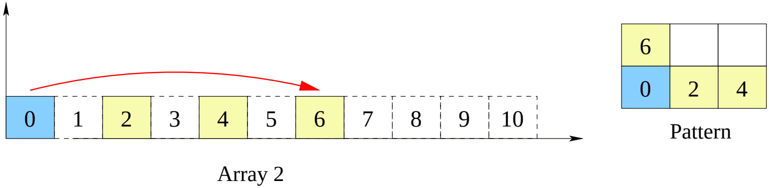
ARRAY-OL (cont'd)

Fitting : how each pattern is filled by array elements.

Fitting example 1 : $o = \begin{pmatrix} 0 \end{pmatrix}$, $F = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$



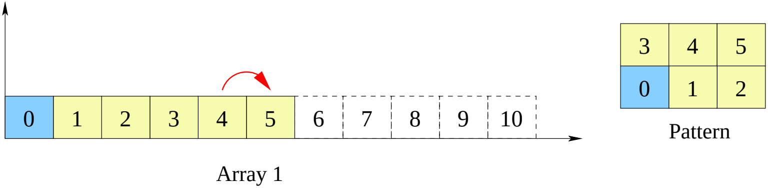
Fitting example 2 : $o = \begin{pmatrix} 0 \end{pmatrix}$, $F = \begin{pmatrix} 2 \\ 6 \end{pmatrix}$



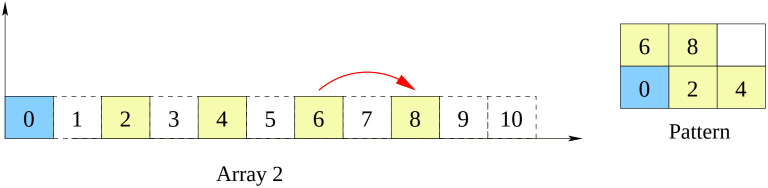
ARRAY-OL (cont'd)

Fitting : how each pattern is filled by array elements.

Fitting example 1 : $o = \begin{pmatrix} 0 \end{pmatrix}$, $F = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$



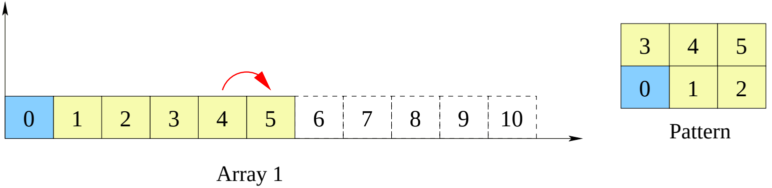
Fitting example 2 : $o = \begin{pmatrix} 0 \end{pmatrix}$, $F = \begin{pmatrix} 2 \\ 6 \end{pmatrix}$



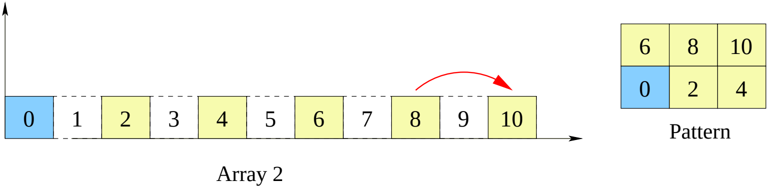
Array-OL (cont'd)

Fitting : how each pattern is filled by array elements.

Fitting example 1 : $o = \begin{pmatrix} 0 \end{pmatrix}$, $F = \begin{pmatrix} 1 & 3 \end{pmatrix}$

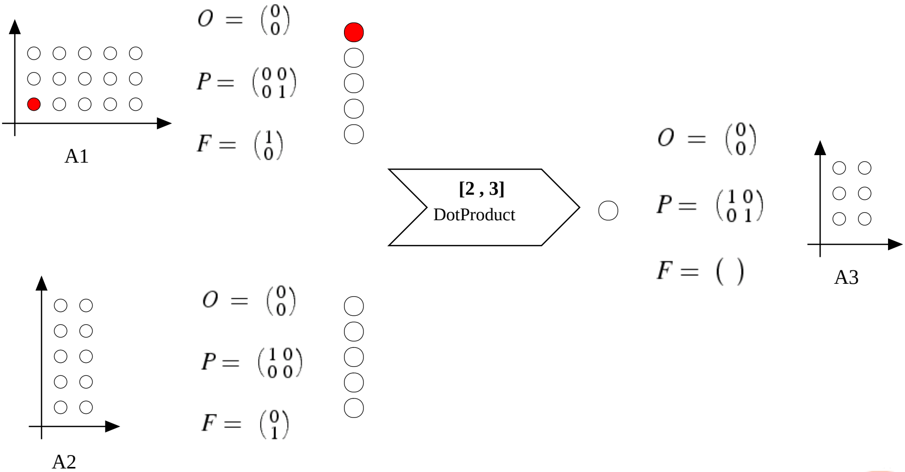


Fitting example 2 : $o = \begin{pmatrix} 0 \end{pmatrix}$, $F = \begin{pmatrix} 2 & 6 \end{pmatrix}$



ARRAY-OL (cont'd)

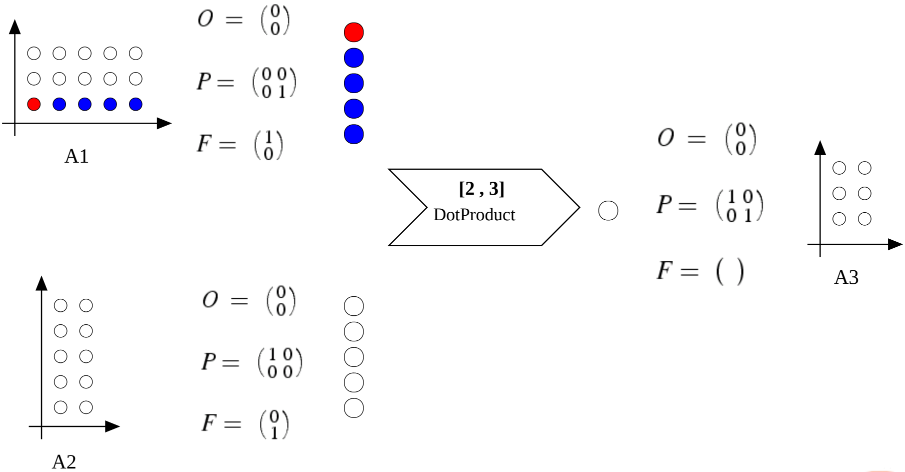
An example of repetition task : array product



Number of instances : 2*3=6 ; Repetition point : [0,0]

ARRAY-OL (cont'd)

An example of repetition task : array product



$$O = \begin{pmatrix}0\\0\end{pmatrix} \quad P = \begin{pmatrix}0 & 0\\0 & 1\end{pmatrix} \quad F = \begin{pmatrix}1\\0\end{pmatrix}$$

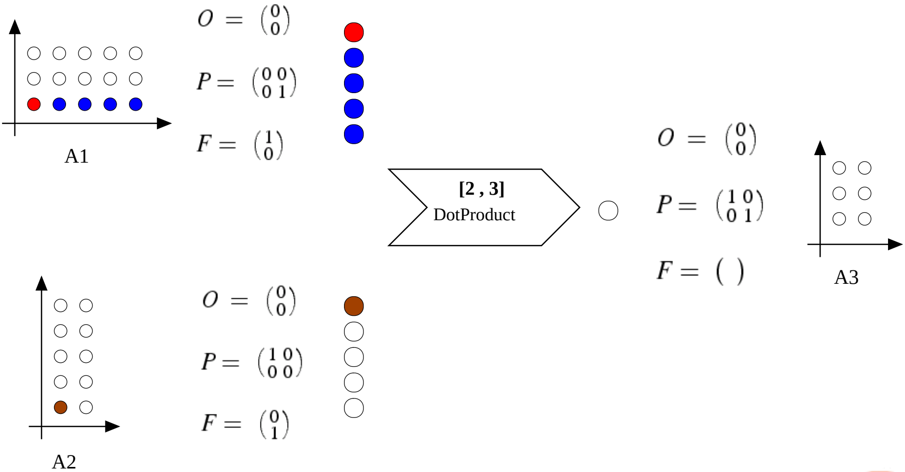
$$O = \begin{pmatrix}0\\0\end{pmatrix} \quad P = \begin{pmatrix}1 & 0\\0 & 0\end{pmatrix} \quad F = \begin{pmatrix}0\\1\end{pmatrix}$$

$$O = \begin{pmatrix}0\\0\end{pmatrix} \quad P = \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} \quad F = (\;)$$

Number of instances : 2*3=6 ; Repetition point : [0,0]

ARRAY-OL (cont'd)

An example of repetition task : array product



$O = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ $P = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$ $F = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

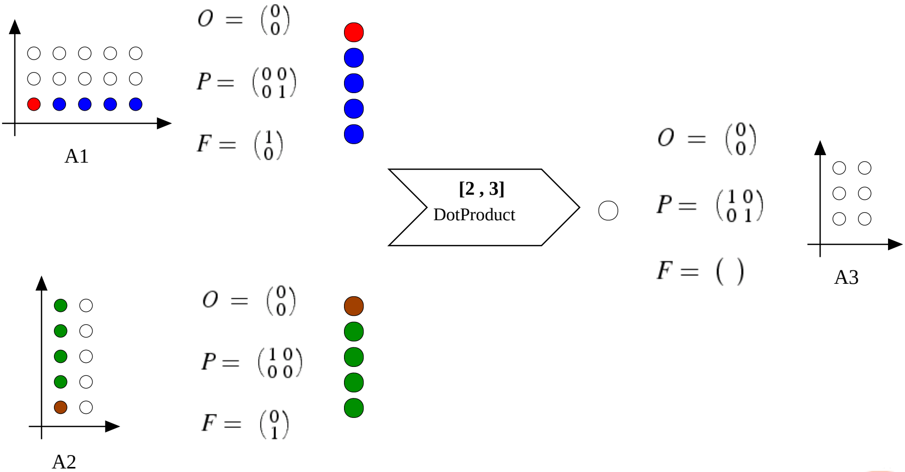
$O = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ $P = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ $F = (\ )$

$O = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ $P = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ $F = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

Number of instances : 2*3=6 ; Repetition point : [0,0]

ARRAY-OL (cont'd)

An example of repetition task : array product



$O = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ $P = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$ $F = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

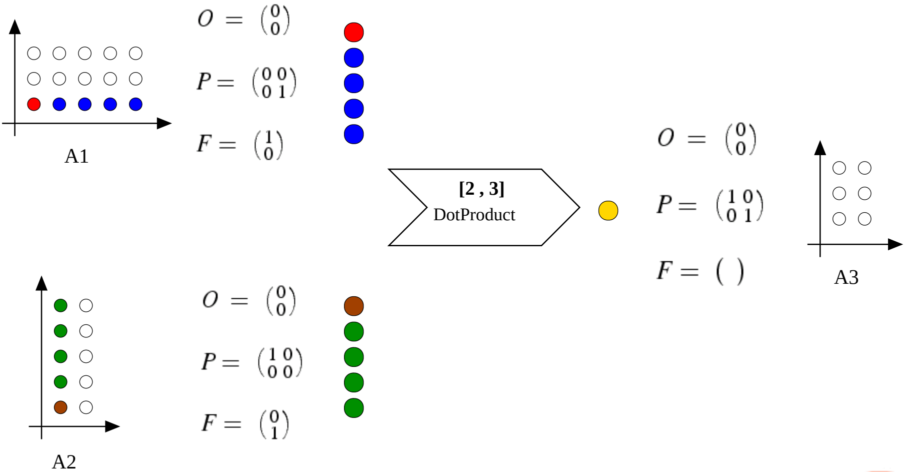
$O = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ $P = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ $F = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

$O = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ $P = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ $F = (\ )$

Number of instances : 2*3=6 ; Repetition point : [0,0]

ARRAY-OL (cont'd)

An example of repetition task : array product



A1

$O = \begin{pmatrix}0\\0\end{pmatrix}$

$P = \begin{pmatrix}0 & 0\\0 & 1\end{pmatrix}$

$F = \begin{pmatrix}1\\0\end{pmatrix}$

[2 , 3]
DotProduct

$O = \begin{pmatrix}0\\0\end{pmatrix}$

$P = \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}$

$F = (\ )$

A3

$O = \begin{pmatrix}0\\0\end{pmatrix}$

$P = \begin{pmatrix}1 & 0\\0 & 0\end{pmatrix}$

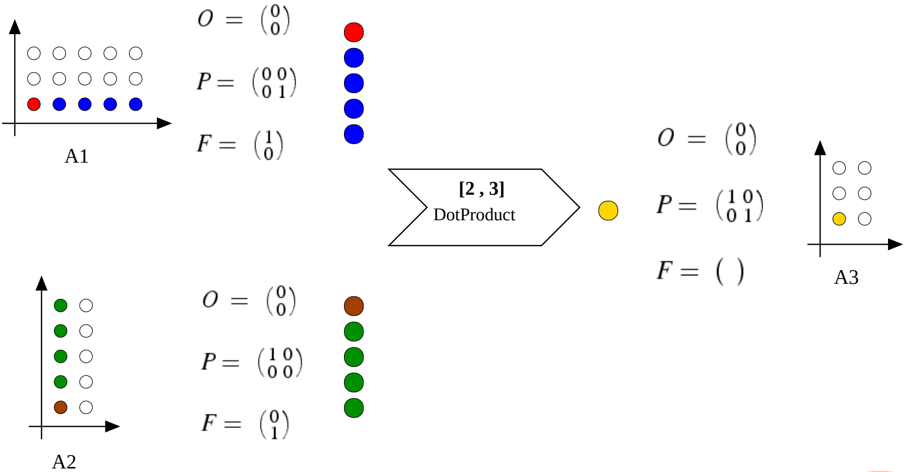
$F = \begin{pmatrix}0\\1\end{pmatrix}$

A2

Number of instances : 2*3=6 ; Repetition point : [0,0]

ARRAY-OL (cont'd)

An example of repetition task : array product



$$O = \begin{pmatrix}0\\0\end{pmatrix} \quad P = \begin{pmatrix}0 & 0\\0 & 1\end{pmatrix} \quad F = \begin{pmatrix}1\\0\end{pmatrix}$$

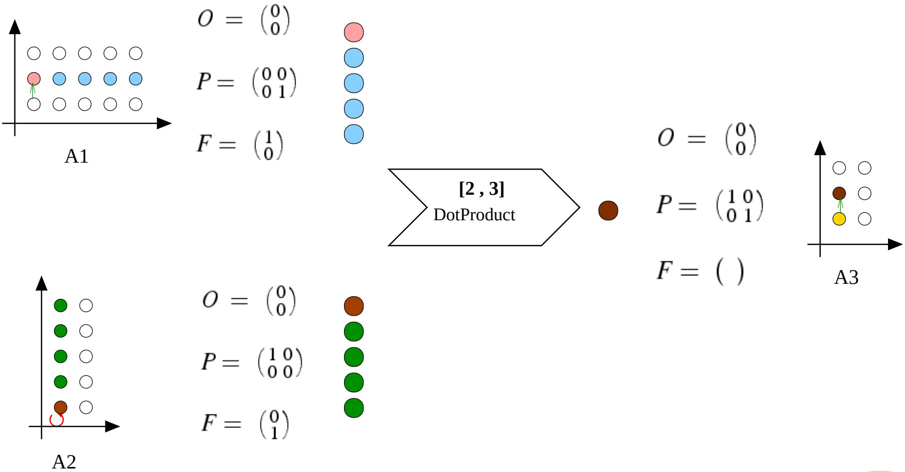
$$O = \begin{pmatrix}0\\0\end{pmatrix} \quad P = \begin{pmatrix}1 & 0\\0 & 0\end{pmatrix} \quad F = \begin{pmatrix}0\\1\end{pmatrix}$$

$$O = \begin{pmatrix}0\\0\end{pmatrix} \quad P = \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} \quad F = (\ )$$

Number of instances : 2*3=6 ; Repetition point : [0,0]

ARRAY-OL (cont'd)

An example of repetition task : array product



$$O = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad P = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \quad F = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

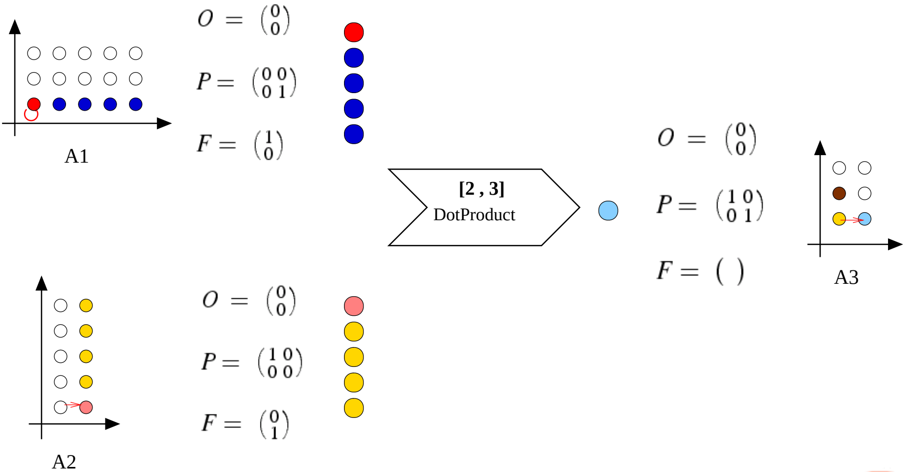
$$O = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad P = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad F = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$O = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad P = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad F = (\ )$$

Number of instances : 2*3=6 ; Repetition point : [0,1]

ARRAY-OL (cont'd)

An example of repetition task : array product



$$O = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad P = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \quad F = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$O = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad P = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad F = (\ )$$

$$O = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad P = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad F = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Number of instances : 2*3=6 ; Repetition point : [1,0]

Existing works

ALPHA language [Mauras, 1989] (vs. ARRAY-OL)

- multidimensional data structures for data-intensive applications
- union of convex polyhedra (vs. arrays)
- data access through indices calculated by affine functions (vs. hierarchical and modular pattern)
- absence of modulo (vs. presence of modulo)

Introduction

GASPARD methodology

General scheme

Data-intensive processing

ARRAY-OL language

Existing works

Simple synchronous modeling of GASPARD models

Parallel model

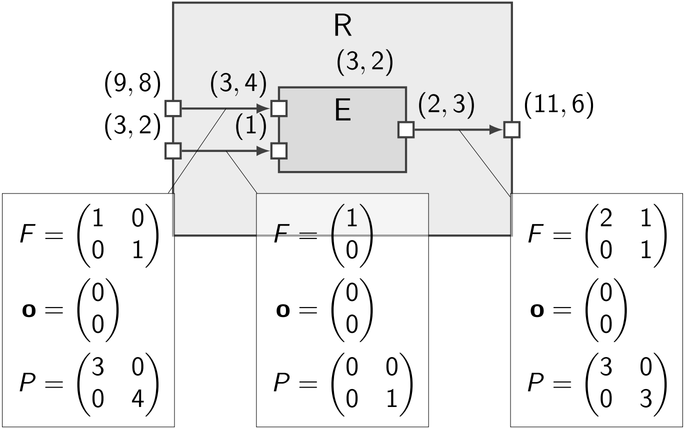
Serialized model

Validation issues

Conclusions

Synchronous modeling of Gaspard models

Illustration of the modeling by the following example :



$$F = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \mathbf{o} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad P = \begin{pmatrix} 3 & 0 \\ 0 & 4 \end{pmatrix}$$

$$F = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \mathbf{o} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad P = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

$$F = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix} \quad \mathbf{o} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad P = \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix}$$

Parallel model

Modeling of repetition task

$$\forall j \in \mathbf{r},\ A_3[< ind_3^j >] := E(A_1[< ind_1^j >], A_2[< ind_2^j >])$$

- j : a point in the repetition space r
- $< ind_i^j >$: the set of index associated with pattern j
- $A_i[< ind_i^j >]$: the pattern j associated with array A_i

Parallel model (cont'd)

Decomposition of a repetition

Input tilers : $p_1^j := A_1[< ind_1^j >]$ $p_2^j := A_2[< ind_2^j >]$

Task : $p_3^j := E(p_1^j, p_2^j)$

Output tiler : $A_3[< ind_3^j >] := p_3^j$

Introduction of local variables : p_1^j, p_2^j, p_3^j

Parallel model (cont'd)

Decomposition of a repetition

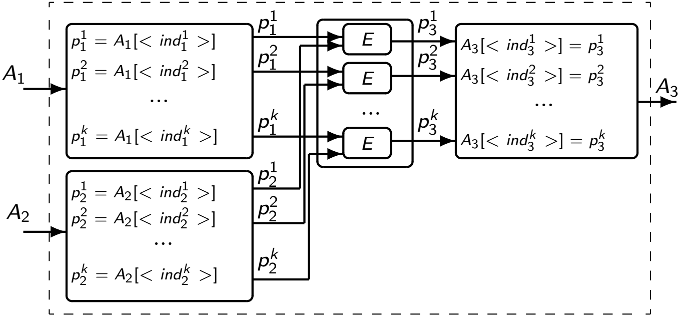
A complete system of equations :

$$\begin{array}{l} (\mid \; p_1^1 := A_1[< ind_1^1 >] \;\mid\; p_2^1 := A_2[< ind_2^1 >] \\ \mid \qquad p_3^1 := E(p_1^1, p_2^1) \;\mid\; A_3[< ind_3^1 >] := p_3^1 \\ \mid \; \ldots \\ \mid \; p_1^k := A_1[< ind_1^k >] \;\mid\; p_2^k := A_2[< ind_2^k >] \\ \mid \qquad p_3^k := E(p_1^k, p_2^k) \;\mid\; A_3[< ind_3^k >] := p_3^k \\ \mid) \end{array} \tag{1}$$

`where` $p_1^1, p_2^1, p_3^1, \ldots, p_1^k, p_2^k, p_3^k;$ `end ;`

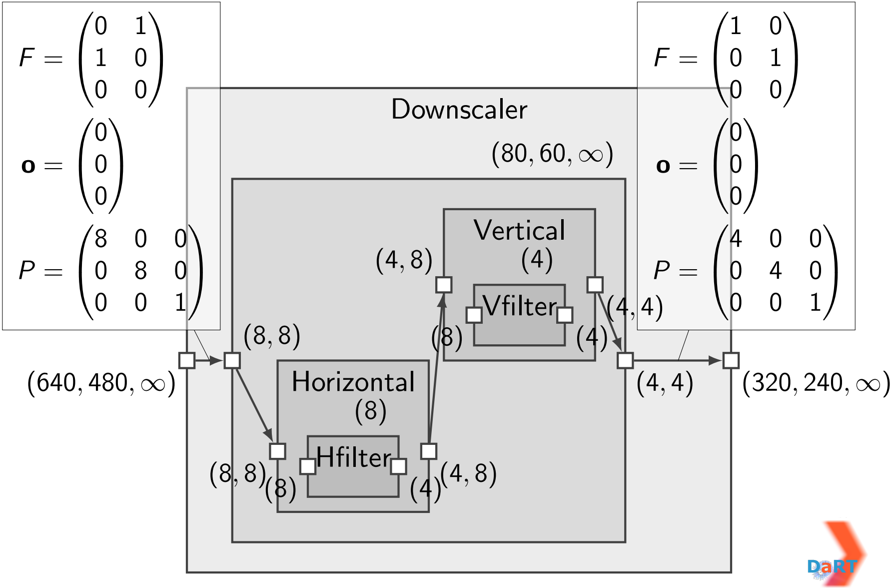
Parallel model (cont'd)

Restructuring and finalization of the model



- Commutativity and associativity of composition operator

Case study : video downscaling



$$F = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix}$$

$$\mathbf{o} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$P = \begin{pmatrix} 8 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$F = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

$$\mathbf{o} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$P = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Generated SIGNAL code

```
module Downscaler_module =
  process DOWNSCALER =
    (?type_array_i A_i;
     !type_array_o A_o;)
    (|(P_i1,...,P_iN:=
          HV_TILER_i(A_i)
     |(P_o1,...,P_oN):=
          R_HV_FILTER(P_i1,...
     |A_o:=HV_TILER_o(P_o1,... |)
     where
      type_pattern_i P_i1,...
      type_pattern_o P_o1,...
      process HV_TILER_i =
      (?type_array_i A_i;
       !type_pattern_i P_i1,...,
      (|P_i1:=HV_PATTERN_i1(A_i)
       |...|)
       where
        process HV_PATTERN_i1 =
```

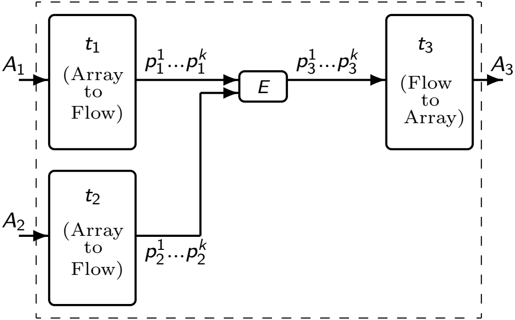
```
  ...
 end%HV_TILER_i% ;
process R_HV_FILTER =
(?type_pattern_i P_i1,...,
 !type_pattern_o P_o1,...,
(|P_o1:=HV_FILTER(P_i1)
 |...|)
 where
  process HV_FILTER =
  (? type_pattern_i P_i;
   ! type_pattern_o P_o;)
  (| p:= H_FILTER (P_i)
   | P_o := V_FILTER(p)
   |)
   where
    type_pattern_l p;
    process H_FILTER = ...
    process V_FILTER = ...
   end%HV_FILTER%;
```

Serialized model

- Simple parallel model
  - Semantically equivalent
  - naively enumeration
- Association of application with architecture :
  from repetition to iteration, introduction of flows
- Sequentialization at different granularity degrees [Labbani 2006]

Serialized model

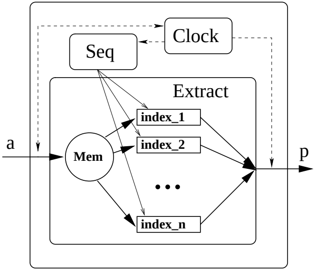
From repetition to iteration : introduction of flows



- Array to flow : produces pattern flows from arrays
- Flow to array : produces arrays from pattern flows

Serialized model

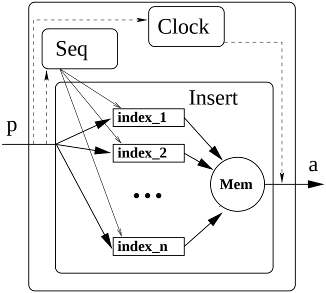
Array to flow



Main components : clock oversampling, sequencer and Extraction

Serialized model

Flow to array

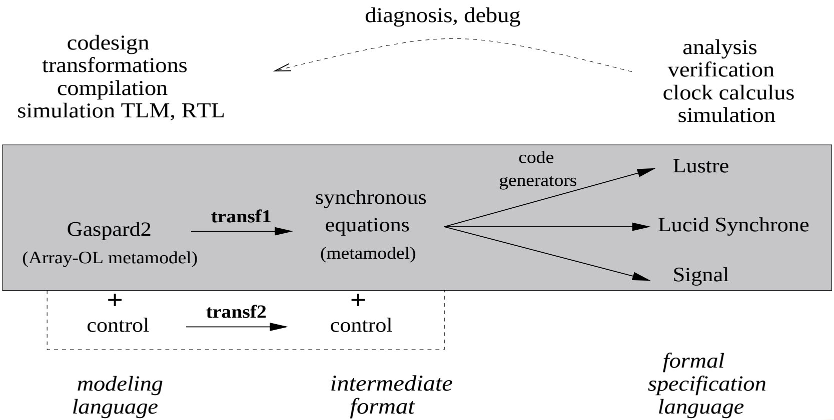


Main components : clock undersampling, sequencer and insertion

Validation issues

- We have a synchronous model with parallel and serialized version that can be combined (mixed model)
- We want to use synchronous analysis tools to address design correctness issues
  ex. N-synchronous Kahn network [Cohen et al. 2006], clock calculus, model-checking
- Example : a simple application with affine clocks synchronizability analysis [Smarandache et al. 1999]

General scheme



diagnosis, debug

codesign
transformations
compilation
simulation TLM, RTL

analysis
verification
clock calculus
simulation

Gaspard2
(Array-OL metamodel)

transf1

synchronous
equations
(metamodel)

code
generators

Lustre

Lucid Synchrone

Signal

+
control

transf2

+
control

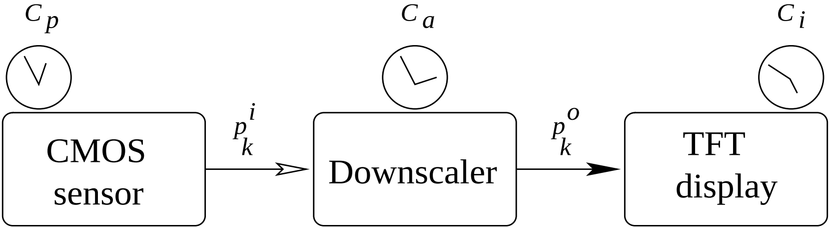
*modeling
language*

*intermediate
format*

*formal
specification
language*

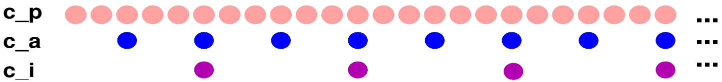
Synchronizability analysis

Camera functionality in a cell phone



Synchronizability analysis

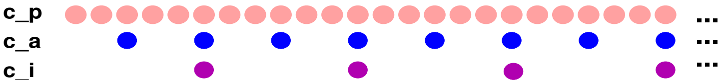
Clock constraints :



1. c_a is an affine undersampling of c_p : $c_p \overset{(1,\phi_1,d_1)}{\longrightarrow} c_a$;
2. c_i is an affine undersampling of c_a : $c_a \overset{(1,\phi_2,d_2)}{\longrightarrow} c_i$;

Synchronizability analysis

Clock constraints :

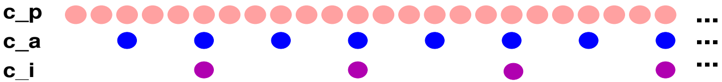


1. c_a is an affine undersampling of c_p : $c_p \xrightarrow{(1,\phi_1,d_1)} c_a$;
2. c_i is an affine undersampling of c_a : $c_a \xrightarrow{(1,\phi_2,d_2)} c_i$;

Now, let us consider a given external constraint, which imposes a particular image production rate c_i', from c_p such that : $c_p \xrightarrow{(1,\phi_3,d_3)} c_i'$. What about the synchronizability of c_i' and c_i ?

Synchronizability analysis

Clock constraints :



1. c_a is an affine undersampling of c_p : $c_p \overset{(1,\phi_1,d_1)}{\longrightarrow} c_a$;
2. c_i is an affine undersampling of c_a : $c_a \overset{(1,\phi_2,d_2)}{\longrightarrow} c_i$;

Now, let us consider a given external constraint, which imposes a particular image production rate c_i', from c_p such that : $c_p \overset{(1,\phi_3,d_3)}{\longrightarrow} c_i'$. What about the synchronizability of c_i' and c_i ?

$$c_i' \textit{ and } c_i \textit{ are synchronizable } \Leftrightarrow \begin{cases} \phi_1 + d_1\phi_2 = \phi_3 \\ d_1 d_2 = d_3 \end{cases} \tag{2}$$

Conclusions and perspectives

Current results :

- Synchronous modeling of Gaspard specifications
- Analysis of GASPARD applications with the help of synchronous techniques
- Implementation of modeling approach following MDE

In the future :

- Complete implementation and validation of current results
- Extension with control features : mode-automata
- Using mixed models (parallel/serialized) combined with task fusion technique for placements in time and space