Semantic Array Dataflow Analysis

Paul Iannetta
UCBL 1, CNRS, ENS de Lyon, Inria, LIP, F-69342, LYON Cedex 07, France

Laure Gonnord
UCBL 1, CNRS, ENS de Lyon, Inria, LIP, F-69342, LYON Cedex 07, France

Lionel Morel
Univ Grenoble Alpes, CEA, List F-38000 Grenoble, France

Tomofumi Yuki
Inria, Univ Rennes, CNRS, IRISA F-35000 Rennes, France

January 23, 2019

If you think I missed a reference please tell me!

paul.iannetta@ens-lyon.fr
1. Inspiration & Motivations

2. Approach

3. Direct Dependencies
1. Inspiration & Motivations

2. Approach

3. Direct Dependencies
Thesis Context (ANR CoDaS: [Gonnord 2017])

Inspiration [Alias et al. 2010]:

- Termination: generates affine schedules (ranking functions) with classical Polyhedral Model computations.
- Program semantics: approximated with (polyhedral) Abstract Interpretation.

Thesis’ subject:

- A Polyhedral Model Extension which supports:
  - Trees [Cohen 1999]
  - Maps = allow to index arrays by array cells
- No closed form to access elements
- Need to make approximations
- First step here: general control flow.
A Semantic Ground For Abstract Interpretation

- Not rely on syntax
- Set as few as possible restrictions
A Semantic Ground For Abstract Interpretation

- Not rely on syntax
- Set as few as possible restrictions

Too constrained syntax (iteration variable is apparent)

- for i from 0 to N [Feautrier 1991]
- for i from 0 while cond(i) [Griebel 1997]
A Semantic Ground For Abstract Interpretation

- Not rely on syntax
- Set as few as possible restrictions

Too constrained syntax (iteration variable is apparant)

- for i from 0 to N [Feautrier 1991]
- for i from 0 while cond(i) [Griebl 1997]

Our target (general while loops)

```plaintext
while cond(i,j,k,l) {
    ...
}
```
A Semantic Ground For Abstract Interpretation

- Not rely on syntax
- Set as few as possible restrictions

Too constrained syntax (iteration variable is apparent)

- for i from 0 to N [Feautrier 1991]
- for i from 0 while cond(i) [Griebl 1997]

Our target (general while loops)

```
while cond(i,j,k,l) {
    ...
}
```

- Iteration variable is not visible anymore
- Leads to non polyhedral programs
- Polyhedral approximation
Benefits of a Semantic - of Abstract Interpretation

- Dissociate definitions from computations:
  - Computations are expressed within the model
  - Can characterize dependences within the model
  - Allows verification.
  - Allows precise characterisations of where abstractions/approximations are made.
A Semantic Ground for Earlier Projects

- Be a model for compiler IR, LLVM [Grosser et al. 2012] or GCC [Trifunović et al. 2010]
  - Integration within real compiler
  - Composition with other optimizations

- Would *a posteriori* justify the implementation on top of a compiler IR.
1. Inspiration & Motivations

2. Approach

3. Direct Dependencies
Steps of the Approach

1. Define a barebone language
   - Allow general programs on arrays
   - Can be computed from a CFG
Steps of the Approach

1. Define a barebone language
   - Allow general programs on arrays
   - Can be computed from a CFG

2. Equip it with a dependence-enabled semantic
Steps of the Approach

1. Define a barebone language
   ▶ Allow general programs on arrays
   ▶ Can be computed from a CFG

2. Equip it with a dependence-enabled semantic

3. Show that dependences can be statically computed (equivalence with previous work).
A Barebone Language

\[
\langle Aexp \rangle ::= \langle Num \rangle \mid \langle Aexp \rangle \langle Aop \rangle \langle Aexp \rangle \mid \langle Vexp \rangle
\]

\[
\langle Aop \rangle ::= '+' \mid '*' \mid '-' \mid '/' \mid 'mod'
\]

\[
\langle Bexp \rangle ::= 'true' \mid 'false' \mid !\langle Bexp \rangle \mid \langle Bexp \rangle \langle Bop \rangle \langle Bexp \rangle \mid \langle Aexp \rangle \langle Cop \rangle \langle Aexp \rangle
\]

\[
\langle Bop \rangle ::= 'or' \mid 'and'
\]

\[
\langle Cop \rangle ::= '<' \mid '=='
\]

\[
\langle Vexp \rangle ::= X \mid X '[' \langle Aexp \rangle ']'\]

\[
\langle Sexp \rangle ::= \kappa_n ':begin' \mid 'skip' \mid \langle Sexp \rangle ';' \langle Sexp \rangle \mid \kappa_n ':if' \langle Bexp \rangle 'then' \langle Sexp \rangle 'else' \langle Sexp \rangle 'fi' \mid \kappa_n ':while' \langle Bexp \rangle 'do' \langle Sexp \rangle 'done' \mid \langle Vexp \rangle ' ':=' \langle Aexp \rangle
\]
A Barebone Language

\[ \langle Aexp \rangle ::= \langle Num \rangle | \langle Aexp \rangle \langle Aop \rangle \langle Aexp \rangle | \langle Vexp \rangle \]

What is important about that syntax is that:

- Allow arrays (scalars = 1-length array)
- Allow conditional tests to reference array cells
- Allow array cells to be referenced by other array cells
- Allow while loops with no restrictions on conditions

\[ \kappa_n \mid \text{while} \langle Bexp \rangle \text{do} \langle Sexp \rangle \text{done} \]
\[ \langle Vexp \rangle ::= \langle Aexp \rangle \]

paul.iannetta@ens-lyon.fr

Semantic Array Dataflow Analysis

January 23, 2019
An Example Program

01 i = 0
02 while i < N
03 j = 0
04 while j < 2
06 j = j + 1
07 k = 0
08 while k < 2
09 A[k+3+i] = A[k] + i
10 k = k + 1
11 i = i + 1

- Iteration variables are not visible
- Add annotation to keep track of operations
00 \kappa_0: \text{begin}
01 \quad i = 0
02 \kappa_1: \text{while } i < N
03 \quad j = 0
04 \kappa_2: \text{while } j < 2
05 \quad A[i+j+1] = A[j] + j
06 \quad j = j + 1
07 \quad k = 0
08 \kappa_3: \text{while } k < 2
09 \quad A[k+3+i] = A[k] + i
10 \quad k = k + 1
11 \quad i = i + 1

- Add variables which counts operations on a hierarchical level
Iteration variables $\kappa_i$ in the semantics

What the semantic is about?
Describe the evolution of an augmented state:
- Standard state: snapshot of the memory at time $t$
- Augmented Memory: value and last modification time
- The current timestamp: a vector of $\kappa$s
Unrolling of an Execution

00 \texttt{\kappa_0\!:} \texttt{begin}
01 i = 0
02 \texttt{\kappa_1\!:} \texttt{while} i < N
03 \texttt{j = 0}
04 \texttt{\kappa_2\!:} \texttt{while} j < 2
05 \texttt{A[i+j+1] = A[j] + j}
06 \texttt{j = j + 1}
07 \texttt{k = 0}
08 \texttt{\kappa_3\!:} \texttt{while} k < 2
09 \texttt{A[k+3+i] = A[k] + i}
10 \texttt{k = k + 1}
11 \texttt{i = i + 1}

Table: Timestamp

\begin{tabular}{c}
\kappa_0 \\
0
\end{tabular}

Table: Memory State

<table>
<thead>
<tr>
<th>Cell</th>
<th>Value</th>
<th>Last access</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>0</td>
<td>[\langle \kappa_0 = 0 \rangle]</td>
</tr>
<tr>
<td>j</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A[1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A[2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>k</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A[3]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Unrolling of an Execution

00 $\kappa_0$: begin 
01 $\kappa_0$: begin 
02 $\kappa_1$: while $i < N$ 
03 $\kappa_1$: while $i < N$ 
04 $\kappa_2$: while $j < 2$ 
05 $\kappa_2$: while $j < 2$ 
06 $\kappa_3$: while $k < 2$ 
07 $\kappa_3$: while $k < 2$ 
08 $\kappa_4$: while $i < N$ 
09 $\kappa_4$: while $i < N$ 
10 $\kappa_4$: while $i < N$ 
11 $\kappa_4$: while $i < N$ 

Table: Timestamp

| $\kappa_0$ | 1 |

Table: Memory State

<table>
<thead>
<tr>
<th>Cell</th>
<th>Value</th>
<th>Last access</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>0</td>
<td>$[\langle \kappa_0 = 0 \rangle]$</td>
</tr>
<tr>
<td>$j$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A[1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A[2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A[3]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Unrolling of an Execution

00 $\kappa_0$: begin
01 i = 0
02 $\kappa_1$: while i < N
03 j = 0
04 $\kappa_2$: while j < 2
05 $\quad$ A[i+j+1] = A[j] + j
06 j = j + 1
07 k = 0
08 $\kappa_3$: while k < 2
09 $\quad$ A[k+3+i] = A[k] + i
10 k = k + 1
11 i = i + 1

Table: Timestamp

<table>
<thead>
<tr>
<th>$\kappa_0$</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\kappa_1$</td>
<td>0</td>
</tr>
</tbody>
</table>

Table: Memory State

<table>
<thead>
<tr>
<th>Cell</th>
<th>Value</th>
<th>Last access</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>0</td>
<td>$[\langle \kappa_0 = 0 \rangle]$</td>
</tr>
<tr>
<td>j</td>
<td>0</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 0 \rangle]$</td>
</tr>
<tr>
<td>A[1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A[2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>k</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A[3]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Unrolling of an Execution

00 $\kappa_0$: \textit{begin}
01 i = 0
02 $\kappa_1$: \textit{while} i < N
03 j = 0
04 $\kappa_2$: \textit{while} j < 2
06 j = j + 1
07 k = 0
08 $\kappa_3$: \textit{while} k < 2
09 A[k+3+i] = A[k] + i
10 k = k + 1
11 i = i + 1

Table: Timestamp

<table>
<thead>
<tr>
<th>$\kappa$</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\kappa_0$</td>
<td>1</td>
</tr>
<tr>
<td>$\kappa_1$</td>
<td>1</td>
</tr>
</tbody>
</table>

Table: Memory State

<table>
<thead>
<tr>
<th>Cell</th>
<th>Value</th>
<th>Last access</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>0</td>
<td>$[\langle \kappa_0 = 0 \rangle]$</td>
</tr>
<tr>
<td>j</td>
<td>0</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 0 \rangle]$</td>
</tr>
<tr>
<td>A[1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A[2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>k</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A[3]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Unrolling of an Execution

00 $\kappa_0$: begin
01 i = 0
02 $\kappa_1$: while $i < N$
03 j = 0
04 $\kappa_2$: while $j < 2$
06 j = j + 1
07 k = 0
08 $\kappa_3$: while $k < 2$
09 $A[k+3+i] = A[k] + i$
10 k = k + 1
11 i = i + 1

Table: Timestamp

<table>
<thead>
<tr>
<th>$\kappa$</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\kappa_0$</td>
<td>1</td>
</tr>
<tr>
<td>$\kappa_1$</td>
<td>1</td>
</tr>
<tr>
<td>$\kappa_2$</td>
<td>0</td>
</tr>
</tbody>
</table>

Table: Memory State

<table>
<thead>
<tr>
<th>Cell</th>
<th>Value</th>
<th>Last access</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>0</td>
<td>$[\langle \kappa_0 = 0 \rangle]$</td>
</tr>
<tr>
<td>j</td>
<td>0</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 0 \rangle]$</td>
</tr>
<tr>
<td>A[1]</td>
<td>$A[0]$</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 0 \rangle]$</td>
</tr>
<tr>
<td>A[2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>k</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A[3]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Unrolling of an Execution

00 \( \kappa_0 \): begin
01 \( i = 0 \)
02 \( \kappa_1 \): while \( i < N \)
03 \( j = 0 \)
04 \( \kappa_2 \): while \( j < 2 \)
05 \( A[i+j+1] = A[j] + j \)
06 \( j = j + 1 \)
07 \( k = 0 \)
08 \( \kappa_3 \): while \( k < 2 \)
09 \( A[k+3+i] = A[k] + i \)
10 \( k = k + 1 \)
11 \( i = i + 1 \)

Table: Timestamp

\begin{tabular}{c|c}
\( \kappa_0 \) & 1 \\
\( \kappa_1 \) & 1 \\
\( \kappa_2 \) & 1 \\
\end{tabular}

Table: Cell Value Last access

<table>
<thead>
<tr>
<th>Cell</th>
<th>Value</th>
<th>Last access</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i )</td>
<td>0</td>
<td>( \langle \kappa_0 = 0 \rangle )</td>
</tr>
<tr>
<td>( j )</td>
<td>1</td>
<td>( \langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 1 \rangle )</td>
</tr>
<tr>
<td>( A[1] )</td>
<td>( A[0] )</td>
<td>( \langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 0 \rangle )</td>
</tr>
</tbody>
</table>

Table: Memory State

paul.iannetta@ens-lyon.fr  Semantic Array Dataflow Analysis  January 23, 2019  15 / 22
Unrolling of an Execution

\[ \kappa_0: \text{begin} \]
\[ i = 0 \]
\[ \kappa_1: \text{while } i < N \]
\[ j = 0 \]
\[ \kappa_2: \text{while } j < 2 \]
\[ j = j + 1 \]
\[ k = 0 \]
\[ \kappa_3: \text{while } k < 2 \]
\[ A[k+3+i] = A[k] + i \]
\[ k = k + 1 \]
\[ i = i + 1 \]

<table>
<thead>
<tr>
<th>Cell</th>
<th>Value</th>
<th>Last access</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>0</td>
<td>(\langle \kappa_0 = 0 \rangle)</td>
</tr>
<tr>
<td>j</td>
<td>1</td>
<td>(\langle \kappa_0 = 1, \kappa_1 = 1, \kappa_2 = 1 \rangle)</td>
</tr>
<tr>
<td>A[1]</td>
<td>A[0]</td>
<td>(\langle \kappa_0 = 1, \kappa_1 = 1, \kappa_2 = 0 \rangle)</td>
</tr>
<tr>
<td>A[2]</td>
<td>A[1] + 1</td>
<td>(\langle \kappa_0 = 1, \kappa_1 = 1, \kappa_2 = 2 \rangle)</td>
</tr>
</tbody>
</table>

Table: Memory State

<table>
<thead>
<tr>
<th>Timestamp</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(\kappa_0)</td>
<td>1</td>
</tr>
<tr>
<td>(\kappa_1)</td>
<td>1</td>
</tr>
<tr>
<td>(\kappa_2)</td>
<td>2</td>
</tr>
</tbody>
</table>
Unrolling of an Execution

00 \( \kappa_0: \) begin
01 \( i = 0 \)
02 \( \kappa_1: \) while \( i < N \)
03 \( j = 0 \)
04 \( \kappa_2: \) while \( j < 2 \)
05 \( A[i+j+1] = A[j] + j \)
06 \( j = j + 1 \)
07 \( k = 0 \)
08 \( \kappa_3: \) while \( k < 2 \)
09 \( A[k+3+i] = A[k] + i \)
10 \( k = k + 1 \)
11 \( i = i + 1 \)

Table: Timestamp

<table>
<thead>
<tr>
<th>( \kappa )</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \kappa_0 )</td>
<td>1</td>
</tr>
<tr>
<td>( \kappa_1 )</td>
<td>1</td>
</tr>
<tr>
<td>( \kappa_2 )</td>
<td>3</td>
</tr>
</tbody>
</table>

Table: Memory State

<table>
<thead>
<tr>
<th>Cell</th>
<th>Value</th>
<th>Last access</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i )</td>
<td>0</td>
<td>( \langle \kappa_0 = 0 \rangle )</td>
</tr>
<tr>
<td>( j )</td>
<td>2</td>
<td>( \langle \kappa_0 = 1, \kappa_1 = 1, \kappa_2 = 3 \rangle )</td>
</tr>
<tr>
<td>( A[1] )</td>
<td>( A[0] )</td>
<td>( \langle \kappa_0 = 1, \kappa_1 = 1, \kappa_2 = 0 \rangle )</td>
</tr>
<tr>
<td>( A[2] )</td>
<td>( A[1] + 1 )</td>
<td>( \langle \kappa_0 = 1, \kappa_1 = 1, \kappa_2 = 2 \rangle )</td>
</tr>
<tr>
<td>( k )</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>( A[3] )</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
Unrolling of an Execution

00 \(\kappa_0\): begin
01 \(i = 0\)
02 \(\kappa_1\): while \(i < N\)
03 \(j = 0\)
04 \(\kappa_2\): while \(j < 2\)
05 \(A[i+j+1] = A[j] + j\)
06 \(j = j + 1\)
07 \(k = 0\)
08 \(\kappa_3\): while \(k < 2\)
09 \(A[k+3+i] = A[k] + i\)
10 \(k = k + 1\)
11 \(i = i + 1\)

Table: Timestamp

| \(\kappa_0\) | 1 |
| \(\kappa_1\) | 2 |

Table: Memory State

<table>
<thead>
<tr>
<th>Cell</th>
<th>Value</th>
<th>Last access</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
<td>0</td>
<td>[\langle \kappa_0 = 0 \rangle]</td>
</tr>
<tr>
<td>(j)</td>
<td>2</td>
<td>[\langle \kappa_0 = 1\rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 3 \rangle]</td>
</tr>
<tr>
<td>(A[1])</td>
<td>(A[0])</td>
<td>[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 0 \rangle]</td>
</tr>
<tr>
<td>(A[2])</td>
<td>(A[1] + 1)</td>
<td>[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 2 \rangle]</td>
</tr>
<tr>
<td>(k)</td>
<td>0</td>
<td>[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 2 \rangle]</td>
</tr>
<tr>
<td>(A[3])</td>
<td>(A[4])</td>
<td></td>
</tr>
</tbody>
</table>
Unrolling of an Execution

00 $\kappa_0$: begin
01 $i = 0$
02 $\kappa_1$: while $i < N$
03 $j = 0$
04 $\kappa_2$: while $j < 2$
06 $j = j + 1$
07 $k = 0$
08 $\kappa_3$: while $k < 2$
09 $A[k+3+i] = A[k] + i$
10 $k = k + 1$
11 $i = i + 1$

Table: Timestamp

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\kappa_0$</td>
<td>1</td>
</tr>
<tr>
<td>$\kappa_1$</td>
<td>3</td>
</tr>
</tbody>
</table>

Table: Memory State

<table>
<thead>
<tr>
<th>Cell</th>
<th>Value</th>
<th>Last access</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>0</td>
<td>$[\langle \kappa_0 = 0 \rangle]$</td>
</tr>
<tr>
<td>$j$</td>
<td>2</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 3 \rangle]$</td>
</tr>
<tr>
<td>A[1]</td>
<td>A[0]</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 0 \rangle]$</td>
</tr>
<tr>
<td>A[2]</td>
<td>A[1] + 1</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 2 \rangle]$</td>
</tr>
<tr>
<td>$k$</td>
<td>0</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 2 \rangle]$</td>
</tr>
<tr>
<td>A[3]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Unrolling of an Execution

00 \( \kappa_0 : \text{begin} \)
01 \( i = 0 \)
02 \( \kappa_1 : \text{while } i < N \)
03 \( j = 0 \)
04 \( \kappa_2 : \text{while } j < 2 \)
05 \( A[i+j+1] = A[j] + j \)
06 \( j = j + 1 \)
07 \( k = 0 \)
08 \( \kappa_3 : \text{while } k < 2 \)
09 \( A[k+3+i] = A[k] + i \)
10 \( k = k + 1 \)
11 \( i = i + 1 \)

Table: Timestamp

<table>
<thead>
<tr>
<th>( \kappa )</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \kappa_0 )</td>
<td>1</td>
</tr>
<tr>
<td>( \kappa_1 )</td>
<td>3</td>
</tr>
<tr>
<td>( \kappa_3 )</td>
<td>0</td>
</tr>
</tbody>
</table>

Table: Memory State

<table>
<thead>
<tr>
<th>Cell</th>
<th>Value</th>
<th>Last access</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>0</td>
<td>([\langle \kappa_0 = 0 \rangle])</td>
</tr>
<tr>
<td>j</td>
<td>1</td>
<td>([\langle \kappa_0 = 1, \langle \kappa_1 = 1, \langle \kappa_2 = 3 \rangle])</td>
</tr>
<tr>
<td>A[1]</td>
<td>A[0]</td>
<td>([\langle \kappa_0 = 1, \langle \kappa_1 = 1, \langle \kappa_2 = 0 \rangle])</td>
</tr>
<tr>
<td>A[2]</td>
<td>A[1] + 1</td>
<td>([\langle \kappa_0 = 1, \langle \kappa_1 = 1, \langle \kappa_2 = 2 \rangle])</td>
</tr>
<tr>
<td>k</td>
<td>0</td>
<td>([\langle \kappa_0 = 1, \langle \kappa_1 = 2 \rangle])</td>
</tr>
<tr>
<td>A[3]</td>
<td>A[0]</td>
<td>([\langle \kappa_0 = 1, \langle \kappa_1 = 3, \langle \kappa_3 = 0 \rangle])</td>
</tr>
<tr>
<td>A[4]</td>
<td>A[0]</td>
<td>([\langle \kappa_0 = 1, \langle \kappa_1 = 3, \langle \kappa_3 = 0 \rangle])</td>
</tr>
</tbody>
</table>
Unrolling of an Execution

00 $\kappa_0$:begin
01 $i = 0$
02 $\kappa_1$:while $i < N$
03 $j = 0$
04 $\kappa_2$:while $j < 2$
06 $j = j + 1$
07 $k = 0$
08 $\kappa_3$:while $k < 2$
09 $A[k+3+i] = A[k] + i$
10 $k = k + 1$
11 $i = i + 1$

<table>
<thead>
<tr>
<th>Cell</th>
<th>Value</th>
<th>Last access</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>0</td>
<td>$[\langle \kappa_0 = 0 \rangle]$</td>
</tr>
<tr>
<td>$j$</td>
<td>2</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 3 \rangle]$</td>
</tr>
<tr>
<td>$A[1]$</td>
<td>$A[0]$</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 0 \rangle]$</td>
</tr>
<tr>
<td>$A[2]$</td>
<td>$A[1] + 1$</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 2 \rangle]$</td>
</tr>
<tr>
<td>$k$</td>
<td>1</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 3 \rangle, \langle \kappa_3 = 1 \rangle]$</td>
</tr>
<tr>
<td>$A[3]$</td>
<td>$A[0]$</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 3 \rangle, \langle \kappa_3 = 0 \rangle]$</td>
</tr>
</tbody>
</table>

Table: Timestamp

| $\kappa_0$ | 1 |
| $\kappa_1$ | 3 |
| $\kappa_3$ | 1 |

Table: Memory State

paul.iannetta@ens-lyon.fr
Semantic Array Dataflow Analysis
January 23, 2019
Unrolling of an Execution

00 $\kappa_0$: begin
01 i = 0
02 $\kappa_1$: while i < N
03 j = 0
04 $\kappa_2$: while j < 2
06 j = j + 1
07 k = 0
08 $\kappa_3$: while k < 2
09 $A[k+3+i] = A[k] + i$
10 k = k + 1
11 i = i + 1

Table: Timestamp

<table>
<thead>
<tr>
<th>$\kappa$</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\kappa_0$</td>
<td>1</td>
</tr>
<tr>
<td>$\kappa_1$</td>
<td>3</td>
</tr>
<tr>
<td>$\kappa_3$</td>
<td>2</td>
</tr>
</tbody>
</table>

Table: Memory State

<table>
<thead>
<tr>
<th>Cell</th>
<th>Value</th>
<th>Last access</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>0</td>
<td>[$\langle \kappa_0 = 0 \rangle$]</td>
</tr>
<tr>
<td>j</td>
<td>2</td>
<td>[$\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 3 \rangle$]</td>
</tr>
<tr>
<td>A[1]</td>
<td>A[0]</td>
<td>[$\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 0 \rangle$]</td>
</tr>
<tr>
<td>A[2]</td>
<td>A[1] + 1</td>
<td>[$\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 2 \rangle$]</td>
</tr>
<tr>
<td>k</td>
<td>1</td>
<td>[$\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 3 \rangle, \langle \kappa_3 = 1 \rangle$]</td>
</tr>
<tr>
<td>A[3]</td>
<td>A[0]</td>
<td>[$\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 3 \rangle, \langle \kappa_3 = 0 \rangle$]</td>
</tr>
<tr>
<td>A[4]</td>
<td>A[1]</td>
<td>[$\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 3 \rangle, \langle \kappa_3 = 2 \rangle$]</td>
</tr>
</tbody>
</table>
Unrolling of an Execution

00 \( \kappa_0 : \text{begin} \)
01 \( i = 0 \)
02 \( \kappa_1 : \text{while } i < N \)
03 \( j = 0 \)
04 \( \kappa_2 : \text{while } j < 2 \)
05 \( A[i+j+1] = A[j] + j \)
06 \( j = j + 1 \)
07 \( k = 0 \)
08 \( \kappa_3 : \text{while } k < 2 \)
09 \( A[k+3+i] = A[k] + i \)
10 \( k = k + 1 \)
11 \( i = i + 1 \)

<table>
<thead>
<tr>
<th>Cell</th>
<th>Value</th>
<th>Last access</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>0</td>
<td>( [\langle \kappa_0 = 0 \rangle] )</td>
</tr>
<tr>
<td>j</td>
<td>2</td>
<td>( [\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 3 \rangle] )</td>
</tr>
<tr>
<td>A[1]</td>
<td>A[0]</td>
<td>( [\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 0 \rangle] )</td>
</tr>
<tr>
<td>A[2]</td>
<td>A[1] + 1</td>
<td>( [\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 2 \rangle] )</td>
</tr>
<tr>
<td>k</td>
<td>2</td>
<td>( [\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 3 \rangle, \langle \kappa_3 = 3 \rangle] )</td>
</tr>
<tr>
<td>A[3]</td>
<td>A[0]</td>
<td>( [\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 3 \rangle, \langle \kappa_3 = 0 \rangle] )</td>
</tr>
<tr>
<td>A[4]</td>
<td>A[1]</td>
<td>( [\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 3 \rangle, \langle \kappa_3 = 2 \rangle] )</td>
</tr>
</tbody>
</table>

Table: Memory State

Table: Timestamp

\( \kappa_0 \quad 1 \)
\( \kappa_1 \quad 3 \)
\( \kappa_3 \quad 3 \)
Unrolling of an Execution

00 $\kappa_0$: begin
01 $i = 0$
02 $\kappa_1$: while $i < N$
03 $j = 0$
04 $\kappa_2$: while $j < 2$
06 $j = j + 1$
07 $k = 0$
08 $\kappa_3$: while $k < 2$
09 $A[k+3+i] = A[k] + i$
10 $k = k + 1$
11 $i = i + 1$

Table: Timestamp

| $\kappa_0$ | 1 |
| $\kappa_1$ | 4 |

Table: Memory State

<table>
<thead>
<tr>
<th>Cell</th>
<th>Value</th>
<th>Last access</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>1</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 4 \rangle]$</td>
</tr>
<tr>
<td>$j$</td>
<td>2</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 3 \rangle]$</td>
</tr>
<tr>
<td>$A[1]$</td>
<td>$A[0]$</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 0 \rangle]$</td>
</tr>
<tr>
<td>$A[2]$</td>
<td>$A[1] + 1$</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 2 \rangle]$</td>
</tr>
<tr>
<td>$k$</td>
<td>2</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 3 \rangle, \langle \kappa_3 = 3 \rangle]$</td>
</tr>
<tr>
<td>$A[3]$</td>
<td>$A[0]$</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 3 \rangle, \langle \kappa_3 = 0 \rangle]$</td>
</tr>
<tr>
<td>$A[4]$</td>
<td>$A[1]$</td>
<td>$[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 3 \rangle, \langle \kappa_3 = 2 \rangle]$</td>
</tr>
</tbody>
</table>
Unrolling of an Execution

00 \( \kappa_0: \text{begin} \)
01 \( i = 0 \)
02 \( \kappa_1: \text{while } i < N \)
03 \( j = 0 \)
04 \( \kappa_2: \text{while } j < 2 \)
05 \( A[i+j+1] = A[j] + j \)
06 \( j = j + 1 \)
07 \( k = 0 \)
08 \( \kappa_3: \text{while } k < 2 \)
09 \( A[k+3+i] = A[k] + i \)
10 \( k = k + 1 \)
11 \( i = i + 1 \)

<table>
<thead>
<tr>
<th>Cell</th>
<th>Value</th>
<th>Last access</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>1</td>
<td>( \langle \kappa_0 = 1, \kappa_1 = 4 \rangle )</td>
</tr>
<tr>
<td>j</td>
<td>0</td>
<td>( \langle \kappa_0 = 1, \kappa_1 = 5 \rangle )</td>
</tr>
<tr>
<td>A[1]</td>
<td>A[0]</td>
<td>( \langle \kappa_0 = 1, \kappa_1 = 1, \kappa_2 = 0 \rangle )</td>
</tr>
<tr>
<td>A[2]</td>
<td>A[1] + 1</td>
<td>( \langle \kappa_0 = 1, \kappa_1 = 1, \kappa_2 = 2 \rangle )</td>
</tr>
<tr>
<td>k</td>
<td>2</td>
<td>( \langle \kappa_0 = 1, \kappa_1 = 3, \kappa_3 = 3 \rangle )</td>
</tr>
<tr>
<td>A[3]</td>
<td>A[0]</td>
<td>( \langle \kappa_0 = 1, \kappa_1 = 3, \kappa_3 = 0 \rangle )</td>
</tr>
<tr>
<td>A[4]</td>
<td>A[1]</td>
<td>( \langle \kappa_0 = 1, \kappa_1 = 3, \kappa_3 = 2 \rangle )</td>
</tr>
</tbody>
</table>

Table: Memory State
Unrolling of an Execution

```
00  \kappa_0: \text{begin}
01  i = 0
02  \kappa_1: \text{while } i < N
03      j = 0
04  \kappa_2: \text{while } j < 2
06      j = j + 1
07  k = 0
08  \kappa_3: \text{while } k < 2
09      A[k+3+i] = A[k] + i
10      k = k + 1
11  i = i + 1
```

<table>
<thead>
<tr>
<th>Cell</th>
<th>Value</th>
<th>Last access</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>0</td>
<td>[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 4 \rangle]</td>
</tr>
<tr>
<td>j</td>
<td>0</td>
<td>[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 5 \rangle]</td>
</tr>
<tr>
<td>A[1]</td>
<td>A[0]</td>
<td>[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 0 \rangle]</td>
</tr>
<tr>
<td>A[2]</td>
<td>A[1] + 1</td>
<td>[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 2 \rangle]</td>
</tr>
<tr>
<td>k</td>
<td>2</td>
<td>[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 3 \rangle, \langle \kappa_3 = 3 \rangle]</td>
</tr>
<tr>
<td>A[3]</td>
<td>A[0]</td>
<td>[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 3 \rangle, \langle \kappa_3 = 0 \rangle]</td>
</tr>
<tr>
<td>A[4]</td>
<td>A[1]</td>
<td>[\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 3 \rangle, \langle \kappa_3 = 2 \rangle]</td>
</tr>
</tbody>
</table>

Table: Timestamp

<table>
<thead>
<tr>
<th>\kappa_0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>\kappa_1</td>
<td>6</td>
</tr>
</tbody>
</table>

Table: Memory State
Unrolling of an Execution

00 \( \kappa_0 \): begin
01 i = 0
02 \( \kappa_1 \): while i < N
03 j = 0
04 \( \kappa_2 \): while j < 2
05 \( A[i+j+1] = A[j] + j \)
06 j = j + 1
07 k = 0
08 \( \kappa_3 \): while k < 2
09 \( A[k+3+i] = A[k] + i \)
10 k = k + 1
11 i = i + 1

Table: Timestamp

<table>
<thead>
<tr>
<th>( \kappa )</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \kappa_0 )</td>
<td>1</td>
</tr>
<tr>
<td>( \kappa_1 )</td>
<td>6</td>
</tr>
<tr>
<td>( \kappa_2 )</td>
<td>0</td>
</tr>
</tbody>
</table>

Table: Memory State

<table>
<thead>
<tr>
<th>Cell</th>
<th>Value</th>
<th>Last access</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>1</td>
<td>( [\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 4 \rangle] )</td>
</tr>
<tr>
<td>j</td>
<td>0</td>
<td>( [\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 5 \rangle] )</td>
</tr>
<tr>
<td>A[1]</td>
<td>A[0]</td>
<td>( [\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 6 \rangle, \langle \kappa_2 = 0 \rangle] )</td>
</tr>
<tr>
<td>A[2]</td>
<td>A[1] + 1</td>
<td>( [\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 1 \rangle, \langle \kappa_2 = 2 \rangle] )</td>
</tr>
<tr>
<td>k</td>
<td>2</td>
<td>( [\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 3 \rangle, \langle \kappa_3 = 3 \rangle] )</td>
</tr>
<tr>
<td>A[3]</td>
<td>A[0]</td>
<td>( [\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 3 \rangle, \langle \kappa_3 = 0 \rangle] )</td>
</tr>
<tr>
<td>A[4]</td>
<td>A[1]</td>
<td>( [\langle \kappa_0 = 1 \rangle, \langle \kappa_1 = 3 \rangle, \langle \kappa_3 = 2 \rangle] )</td>
</tr>
</tbody>
</table>
1. Inspiration & Motivations

2. Approach

3. Direct Dependencies
An operation is a tuple: \((s, t)\) where

- \(s\) is a statement (i.e., \(A[i] = A[i-1] + i\))
- \(t\) is a timestamp (i.e., \([\langle \kappa_0, 3 \rangle, \langle \kappa_1, 1 \rangle]\))
Trace

Operation

An operation is a tuple: \((s, t)\) where
- \(s\) is a statement (i.e., \(A[i] = A[i-1] + i\))
- \(t\) is a timestamp (i.e., \([⟨κ_0, 3⟩, ⟨κ_1, 1⟩]\))

Zoom on the state of the memory at \(o_i\)

\[
\begin{align*}
o_0 \rightarrow o_1 \rightarrow \cdots \rightarrow o_i \rightarrow o_{i+1} \rightarrow \cdots
\end{align*}
\]

\[
\begin{array}{c}
o_i = (s, t) \\
\cdots \quad \cdots \\
A[21] \quad [⟨κ_0, 3⟩, ⟨κ_1, 1⟩] \\
\cdots \quad \cdots 
\end{array}
\]
Dependence Definition

Inspired from [Feautrier 1991].

Dependence: \( o_2 \) depends on an operation \( o_1 \)

1. \( o_1 \) is valid (i.e., it belongs to a trace)
2. \( o_1 = (s_1, t_1) \) occurs before \( o_2 = (s_2, t_2) \)
   - \( t_1 <_{lex} t_2 \)
3. \( o_2 = (s_2, t_2) \) is reading and/or writing a cell that \( o_1 = (s_1, t_1) \) wrote
   - \( s_1 \) is “\( A[f(i, j, k)] = \ldots \)”
   - \( s_2 \) is “\( \ldots = A[g(l, r)] \)”
   - “\( f(i, j, k) = g(l, r) \)”

In (3), the access uses real variables
Dependence Computation I

...]
08 $\kappa_3$: while $k < 2$
09 $A[k+3+i] = A[k] + i$
10 $k = k + 1$
...]

1. Express timestamps as function of real variables
   1. Express the relation between variables before and after a loop step
      ★ $k \sim k + 1$
      ★ $\kappa_3 \sim \kappa_3 + 2$
   2. Compute the transitive closure (if the loop is affine) [Verdoolaege et al. 2011]
      ★ $\kappa_3 = 3k$
[...] 
08 \( \kappa_3: \) while \( k < 2 \) 
09 \( A[k+3+i] = A[k] + i \) 
10 \( k = k + 1 \) 
[...]

Solve the parametrized integer linear programs

- Parameters: \( i, k \)
- Conditions:
  - \( 0 \leq i, k, i', k' \)
  - \( k + 3 + i = k' \)

Express back the dependences within our model
Conclusion

The ideas are not new. However,

- We got rid of the syntax
- We have a new dependence front-end to an integer linear program

Future work,

- Non polyhedral programs
  - Find reasonable approximations as polyhedral programs


References II


Annexe: Semantics 1/3

\[
\begin{align*}
\text{skip} & \quad \langle \sigma, \text{skip} \rangle \\
\text{begin} & \quad \langle \sigma, \kappa_0 : \text{begin}; s \rangle \rightarrow \langle \text{upd}(\sigma, \kappa_0, 0), s \rangle \\
\text{Assign} & \quad \langle \sigma, \nu := e ; s \rangle \rightarrow \langle \text{incr}(\sigma[\nu := e]), s \rangle
\end{align*}
\]

**incr and upd**

- **incr**: increments the timestamp
- **upd**: create a fresh $\kappa$ or does nothing
- $\sigma \setminus \kappa_n$ remove $\kappa_n$ from the state
Annexe: Semantics 2/3

\[
\begin{align*}
\text{WhT} & : \langle \sigma, b_0 \rangle \rightarrow \text{true} \quad \langle \text{upd}(\sigma, \kappa_n, 0), s_1 \rangle \rightarrow^+ \langle \sigma', \text{skip} \rangle \\
& \quad \langle \sigma, \kappa_n : \text{while } b_0 \text{ do } s_1 \text{ done ; } s \rangle \rightarrow \\
& \quad \langle \text{incr}(\sigma'), \kappa_n : \text{while } b_0 \text{ do } s_1 \text{ done ; } s \rangle
\\
\text{WhF} & : \langle \sigma, b_0 \rangle \rightarrow \text{false} \\
& \quad \langle \sigma, \kappa_n : \text{while } b_0 \text{ do } s_1 \text{ done ; } s \rangle \rightarrow \langle \text{incr}(\sigma \setminus \kappa_n), s \rangle
\\
\text{IT} & : \langle \sigma, b_0 \rangle \rightarrow \text{true} \quad \langle \text{upd}(\sigma, \kappa_n, -\text{length}(s_1)), s_1 \rangle \rightarrow^+ \langle \sigma', \text{skip} \rangle \\
& \quad \langle \sigma, \kappa_n : \text{if } b_0 \text{ then } s_1 \text{ else } s_2 \text{ fi; } s \rangle \rightarrow \langle \text{incr}(\sigma' \setminus \kappa_n); s \rangle
\\
\text{IF} & : \langle \sigma, b_0 \rangle \rightarrow \text{false} \quad \langle \text{upd}(\sigma, \kappa_n, 0), s_2 \rangle \rightarrow^+ \langle \sigma', \text{skip} \rangle \\
& \quad \langle \sigma, \kappa_n : \text{if } b_0 \text{ then } s_1 \text{ else } s_2 \text{ fi; } s \rangle \rightarrow \langle \text{incr}(\sigma' \setminus \kappa_n), s \rangle
\end{align*}
\]