Data Dependences

Louis-Noël Pouchet

CS & ECE
Colorado State University

February 23, 2020

PPoPP'20 Tutorial

Data Dependences

Purpose of Dependence Analysis

- Not all program transformations preserve the semantics
- Semantics is preserved if the dependence are preserved

- In standard frameworks, it usually means reordering statements/loops
  - Statements containing dependent references should not be executed in a different order
  - Granularity: usually a reference to an **array**

- In the polyhedral framework, it means reordering statement **instances**
  - Statement instances containing dependent references should not be executed in a different order
  - Granularity: a reference to an **array cell**

Detour: Compute Data Spaces

Exercise: Compute the set of cells of A accessed

Example

```
for (i = 0; i < N; ++i)
  for (j = i; j < N; ++j)
    A[2*i + 3][4*j] = i * j;
```

- $\mathcal{D}_S$: $\{i,j \mid 0 \leq i < N,\ i \leq j < N\}$
- Function: $f_A : \{2i+3, 4j \mid i,j \in \mathbb{Z}\}$
- $f_A(\mathcal{D}_S)$ is the set of cells of A accessed (a $\mathcal{Z}$-polyhedron):

ISCC Demo

- See dataspaces-and-communication-generation.pdf on website
- See dataspace.iscc on website

Data Dependence

Definition (Bernstein conditions)

Given two references, there exists a dependence between them if the three following conditions hold:

- they reference the same array (cell)
- one of this access is a write
- the two associated statements are executed

Three categories of dependences:

- RAW (Read-After-Write, aka flow): first reference writes, second reads
- WAR (Write-After-Read, aka anti): first reference reads, second writes
- WAW (Write-After-Write, aka output): both references writes

Another kind: RAR (Read-After-Read dependences), used for locality/reuse computations

Some Terminology on Dependence Relations

We categorize the dependence relation in three kinds:

- **Uniform dependences:** the distance between two dependent iterations is a constant
  - ex: $i \rightarrow i+1$
  - ex: $i,j \rightarrow i+1,j+1$
- **Non-uniform dependences:** the distance between two dependent iterations varies during the execution
  - ex: $i \rightarrow i+j$
  - ex: $i \rightarrow 2i$
- **Parametric dependences:** at least a parameter is involved in the dependence relation
  - ex: $i \rightarrow i+N$
  - ex: $i+N \rightarrow j+M$

Dependence Polyhedron [1/3]

Principle: model all **pairs** of instances in dependence

Definition (Dependence of statement instances)

A statement S depends on a statement R (written $R \to S$) if there exists an operation $S(\vec{x}_S)$ and $R(\vec{x}_R)$ and a memory location m such that:

1. $S(\vec{x}_S)$ and $R(\vec{x}_R)$ refer to the same memory location m, and at least one of them writes to that location,
2. x_S and x_R belongs to the iteration domain of R and S,
3. in the original sequential order, $S(\vec{x}_S)$ is executed before $R(\vec{x}_R)$.

Dependence Polyhedron [2/3]

1. *Same memory location*: equality of the subscript functions of a pair of references to the same array: $F_S\vec{x}_S + a_S = F_R\vec{x}_R + a_R$.

2. *Iteration domains*: both S and R iteration domains can be described using affine inequalities, respectively $A_S\vec{x}_S + c_S \geq 0$ and $A_R\vec{x}_R + c_R \geq 0$.

3. *Precedence order*: each case corresponds to a common loop depth, and is called a *dependence level*.

   For each dependence level l, the precedence constraints are the equality of the loop index variables at depth lesser to l: $x_{R,i} = x_{S,i}$ for $i < l$ and $x_{R,l} > x_{S,l}$ if l is less than the common nesting loop level. Otherwise, there is no additional constraint and the dependence only exists if S is textually before R.

   Such constraints can be written using linear inequalities: $P_{l,S}\vec{x}_S - P_{l,R}\vec{x}_R + b \geq 0$.

Dependence Polyhedron [3/3]

The dependence polyhedron for $R \to S$ at a given level l and for a given pair of references f_R, f_S is described as [Feautrier/Bastoul]:

$$\mathcal{D}_{R,S,f_R,f_S,l} : D\begin{pmatrix}\vec{x}_S\\ \vec{x}_R\end{pmatrix} + d = \left[\begin{array}{cc} F_S & -F_R \\ \hline A_S & 0 \\ 0 & A_R \\ PS & -P_R \end{array}\right]\begin{pmatrix}\vec{x}_S\\ \vec{x}_R\end{pmatrix} + \begin{pmatrix} a_S - a_R \\ c_S \\ c_R \\ b \end{pmatrix} \begin{array}{c} = 0 \\ \hline \geq \vec{0} \end{array}$$

A few properties:

- We can always build the dep polyhedron for a given pair of affine array accesses (it is convex)
- It is exact, if the iteration domain and the access functions are also exact
- it is over-approximated if the iteration domain or the access function is an approximation

ISCC Demo

Live demo: compute dependence polyhedra from access functions

Polyhedral Dependence Graph

Definition (Polyhedral Dependence Graph)

The Polyhedral Dependence Graph is a multi-graph with one vertex per syntactic program statement S_i, with edges $S_i \rightarrow S_j$ for each dependence polyhedron $\mathcal{D}_{S_i,S_j}$.