A reduction is an associative and commutative operator applied to collections of values to produce a collection of results.
**Reductions**

- A reduction is an associative and commutative operator applied to collections of values to produce a collection of results.
- Our collections are polyhedral sets.

![Diagram of polyhedral sets](image)

**Highlights**

- Simplify Reductions
  - Automatically
  - Optimally
- Non-trivial Examples
  - RNA Secondary Structure Prediction: (*mfold* package)
  - Finite Element Methods
  - Max filter
  - Line-of-sight computation in GIS
Outline
- Introduction and Problem Definition
- Sharing
- Simplification
  - Recursive Simplification
- Reduction Decomposition
- Motivating Example
- Distributivity
- Applications
- Previous Work and Conclusions

Representation
- Three equivalent forms of representation
  - Geometric
  - Loops (bounds define the polyhedron)
    for $i = 1$ to $n$ {
      $Y[i] = 0$;
      for $j = 1$ to $i-1$
        for $k = 1$ to $i-j$
          $Y[i] += F[i,j,k]$; }
  - Equations
    $Y_i = \sum_{j=1}^{i-1} \sum_{k=1}^{i-j} F_{i,j,k}$, $i \in 2, \ldots, n$
If $F_{i,j,k} = X_k$

- All index points on planes parallel to the \{i,j\} plane have the same value
- \{i,j\} is called the share space
- Denoted by green

Aim to replace this polyhedron by one of lesser dimensions
Simplification
Reduction Decomposition

Reduction Decomposition
Reduction Decomposition

Reduction Decomposition
Reduction Decomposition

Y

Reduction Decomposition

Y

Y
Reduction Decomposition

\[ i \quad j \quad k \]

\[ F \]

\[ Y \quad Y \]
Reduction Decomposition

Motivating Example

Equation

\[ Y_i = \max_{j=i}^{2i} \max_{k=i}^{3i-j} X_{j,k} \]

Loop

for i = 0 to n {
    Y[i] = -infinity;
    for j = i to 2i
        for k = i to 3i-j
            Y[i] = max(Y[i], X[j,k]);
    }
Motivating Example

Equation

\[ Y_i = \max_{j=i}^{2i} \max_{k=i}^{3i-j} X_{j,k} \]

Loop

for \( i = 0 \) to \( n \) {
  \( Y[i] = -\text{infinity}; \)
  for \( j = i \) to \( 2i \)
    for \( k = i \) to \( 3i-j \)
      \( Y[i] = \max(Y[i], X[j,k]); \)
}
Motivating Example

Diagram of a geometric model showing a three-dimensional structure with labeled axes and shaded regions.
Motivating Example
Motivating Example

[Graphical representation of a 3D diagram with labeled axes and figures labeled F and Y.]
Motivating Ex. Equations

Let us now see the same transformations applied to equations and the resultant code

\[ Y_i = \max_{j=i}^{3i-j} \max_{k=i}^{2i} X_{j,k} \]

We saw that the inner reduction should be along the \( j + k = \text{constant} \) direction.

Let's perform a change of indices to ensure this.

Introduce \( m = j + k \)

\[ Y_i = \max_{m=2i}^{3i} \max_{k=i}^{m-i} X_{m-k,k} \]

Motivating Ex. Equations (2)

\[ Y_i = \max_{m=2i}^{3i} \max_{k=i}^{m-i} X_{m-k,k} \]

Introduce temporary variable \( Z_{i,m} \) to hold the result of the inner summation \( \max_{k=i}^{m-i} X_{m-k,k} \)

\[ Y_i = \max_{m=2i}^{3i} Z_{i,m} \]

\[ Z_{i,m} = \max_{k=i}^{m-i} X_{m-k,k} \]

When \( m \geq 2(i+1) \), we get the following:

\[ Z_{i,m} = \max_{k=i+1}^{m-(i+1)} \{ X_{m-k,k}, X_{m-i,i}, X_{i,m-i} \} \]
Motivating Ex. Equations (3)

\[ Z_{i,m} = \max\left( \max_{k=i+1}^{m-(i+1)} X_{m-k,k}, X_{m-i,i}, X_{i,m-i} \right) \]

- Note that the max over \( k \) is simply \( Z_{i+1,m} \)
- We can replace the equation for \( Z_{i,m}, m \geq 2(i+1) \) by \( Z_{i,m} = \max(Z_{i+1,m}, X_{m-i,i}, X_{i,m-i}) \)
- The final set of equations

\[
Y_i = \max_{m=2i}^{3i} Z_{i,m}
\]

\[
Z_{i,m} = \begin{cases} 
2i \leq m < 2(i+1) & \max_{k=i}^{m-i} X_{m-k,k} \\
2(i+1) \leq m \leq 3i & \max(Z_{i+1,m}, X_{m-i,i}, X_{i,m-i}) 
\end{cases}
\]

Motivating Ex. Equations (4)

- The code for the transformed set of equations

```plaintext
for i = n downto 0 {
    for m = 2i to 2i+1 {
        Z[i,m] = -infinity;
        for k = i to m-i
            Z[i,m] = max(Z[i,m], X[m-k,k]);
    }
    for m = 2i+2 to 3i
        Z[i,m] = max(Z[i+1,m], X[m-i,i], X[i,m-i]);
    Y[i] = -infinity;
    for m = 2i to 3i
        Y[i] = max(Y[i], Z[i,m]);
    Y[i] = max(Y[i], Z[i,m]);
}

Y_i = \max_{m=2i}^{3i} Z_{i,m}
```
Compound Expressions

- Consider the expression $A_{i,k} \otimes B_{k,j}$
  - The subexpression $A_{i,k}$ is shared along $j$
  - The subexpression $B_{k,j}$ is shared along $i$
- Combined expression has no sharing

Distributivity

- Consider a reduction of the form $\bigoplus_{j=1}^{i} A_{i,k} \otimes B_{k,j}$
- The combined expression has no sharing
- If $\otimes$ distributes over $\bigoplus$, we can do something
  - $A_{i,k}$ is constant within the summation
- We can replace the reduction with $A_{i,k} \otimes \bigoplus_{j=1}^{i} B_{k,j}$
- The reduction is over $B_{k,j}$ now
  - Has a non-trivial share space
  - Can decrease the complexity by one dimension
Final Example

- Consider the following: \( Y_i = \sum_{j=1}^{i} \sum_{k=1}^{i} A_{i,j+k} \times B_{k,j} \)
- \( A_{i,j+k} \) is shared along the vector \((0,1,-1)\)
- \( B_{k,j} \) is shared along the vector \((1,0,0)\)
- Combined expression has no sharing
- Can we exploit distributivity: Not yet!

Final Example

- Consider the following: \( Y_i = \sum_{j=1}^{i} \sum_{k=1}^{i} A_{i,j+k} \times B_{k,j} \)
- Reduction Decomposition may be applied
Final Example

Consider the following: \[ Y_i = \sum_{j=1}^{i} \sum_{k=1}^{i} A_{i,j+k} \times B_{k,j} \]

- Reduction Decomposition may be applied
  - The inner reduction has to be along the vector \((0,1,-1)\)
  - Allows \(A_{i,j+k}\) to be distributed out
  - \(\sum B_{k,j}\) is the inner reduction. Has sharing
  - Enables a one dimensional decrease in complexity

Applications

- RNA Secondary Structure Prediction
- Finite Element Methods
Message to YOU

- The Polyhedral Model is the granddaddy of many loop optimizations
- Known for over 20 years
- Polyhedra are simple
- Can be used to solve reasonably important problems (ex. Decreasing complexity, others)

Previous Work

- Cocke and Keneddy (1977): “Strength Reduction”
- Liu et. al. (2005): “Incrementalization”
Conclusions

- Characterization of complexity of reductions
- Mathematical foundations for simplification
- An algorithm that automatically and optimally applies different simplification transformations