A Unified Framework for Nonlinear Dependence Testing and Symbolic Analysis

Kyle A. Gallivan

Collaboration with: R. van Engelen, J. Birch, B. Walsh, X. Shou

School of Computational Science and Information Technology

Florida State University

Research Interests

- high-performance algorithm technology for multiple applications
- algorithm/architecture interaction
- software support for design/implementation/tuning of high-performance algorithms

Algorithms

- large-scale numerical linear algebra – sparse linear systems, eigenproblems
- real-time numerical linear algebra — adaptive filtering
- model reduction of dynamical systems – novel algorithms and unification theory
- differential equations – highly oscillatory ODEs, hierarchical methods for circuits,PDEs
- inverse problems – autofocus for synthetic aperture radar, blind deconvolution

Algorithms

- encoding and retrieval – text retrieval, image retrieval
- large-scale manifold computations – geodesics, means, optimization
- Current funding
  - NSF ITR ACI0324944 with Purdue and Rice
  - NSF CCR9912415
- Previous – NSF, DARPA, ARO
- Pending – NSF on SAR autofocus with D. Munson U. Michigan

Algorithm/Architecture

- Systolic array design for adaptive filtering
- Cedar – algorithms, applications, performance analysis
- block-based algorithms – BLAS3
- Load/Store performance prediction
- cache models and iterative compilation with Knijnenburg and O'Boyle
- timing estimation and voltage scaling with van Engelen
- Currrent funding
  - NSF EIA 0072043 with D. Whalley FSU

Software support

- Cache and memory restructuring for uniformly-generated accesses with Gannon and Jalby
- FALCON – MATLAB restructuring compiler with Padua and Gallopoulos
- Chains of recurrenes-based restructuring infrastructure with Van Engelen
- Current funding – NSF CCR0105422 with Van Engelen
- Pending funding – NSF Unified Compiler Framework with Van Engelen FSU

Introduction

- Efficient code $\longrightarrow$ restructuring compilers
- efficacy of restructuring depends on power of analysis system
- efficiency of restructuring depends on efficiency of analysis implementation
- current systems are powerful but
  - tend to be collection of independent subsystems with heuristics
  - powerful symbolic processing tends to be slow
  - heuristics are faster but limited
- limitations are still significant

Limitations

- nonlinear symbolic processing
- pointer arithmetic analysis
- conditional control flow
- efficient parameterization of decisions/questions
- Recent commentary
  - Psarris (PC 03, TPDS 04)
  - Franke and O'Boyle (ETAPS CC 01)
  - Wu, Cohen, Hoeflinger Padua (ICS 01)

Unified Framework

Goal: Unified and efficient symbolic analysis system that subsumes most of current technology and extends restructuring capabilities significantly.

- an efficient technique for the detection and substitution of generalized induction variables
- improved array recovery methods for both linear and nonlinear pointer updates
- enhanced value range analysis and array region analysis for conditionally updated and coupled induction variables, and pointers

Unified Framework

- increased accuracy for data dependence analysis for affine and nonlinear array indices, and index expressions involving conditionally updated variables and pointers
- implementation of an efficient nonlinear dependence test that does not require closed forms or code adjustments, e.g. induction variable substitution and array recovery.
- application of technology to compiler support for related embedded systems problems of timing estimation and voltage scaling

Outline

- CR algebra
- Induction variables
- Pointer arithmetic and Array Recovery
- Monotonicity
- Value range analysis
- Array region analysis
- Nonlinear dependence analysis
- Conditional analysis
- Parametric time and power estimation

CR Algebra

- Chains of Recurrences (CR) algebra developed by E. Zima and O. Bachmann
- Extended with new algebraic rules by Van Engelen [ETAPS CC'01]
- Composition of CRs is closed under $+$ and $\times$
- CRs subsume functions of form $\chi(n) = p(n) + \alpha r^n$, i.e., generalized induction variables (GIVs)
- Normal form for CR expression comparison Van Engelen [FSU CS TR '00]

CR Algebra

- Set of complete rewriting rules for CR $\leftrightarrow$ closed form
- tools to create CRs and their closed forms for high level code and RTL available and efficient

  Birch [MS'03] and Walsh '04
- unifies the production of compile-time symbolic decisions that drive restructuring and run-time parameterized decisions for multiversion code.

Basic Function Form

A function f can be rewritten as an system of recurrence relations in terms of $f_0, f_1, f_2, \ldots, f_k$. Evaluating the function represented by f_j for $j = 0, 1, \ldots, k-1$ within a loop $i \geq 0$,

$$f_j(i) = \begin{cases} \iota_j, & \text{if } i = 0 \\ f_j(i-1) \odot_{j+1} f_{j+1}(i-1), & \text{if } i > 0 \end{cases}$$

with $\odot_{j+1} \in \{+, *\}$, and ι_j representing a loop invariant expression. f_k is either loop invariant or another CR function.

Linear Function

Loop counter $i \geq 0$ and induction variable j with addition of constant h

```
for (i = 0; i < N; i++) {
        j = j + h;
}
```

Variable	CR	function
i	$\{0, +, 1\}_i$	i
j	$\{j_0, +, h\}_i$	$j_0 + i * h$

Geometric Series

Loop counter $i \geq 0$ and induction variable j with multiplication by constant h

for (i = 0; $i < N$; i++) {
    j = j * h;
}

Variable	CR	function
i	$\{0, +, 1\}_i$	i
j	$\{j_0, *, h\}_i$	$j_0 * h^i$

Multivariate Affine

Loop counters i_1 and i_2 and induction variable j with addition of constant h.

$$
\begin{array}{l}
\text{for } (i_1 = 0; i_1 <= N; i_1\text{++}) \ \{ \\
\quad \text{for } (i_2 = 0; i_2 < M; i_2\text{++}) \ \{ \\
\quad\quad j = j + h; \\
\quad \} \\
\}
\end{array}
$$

Variable	CR	function
i_1	$\{0, +, 1\}_{i_1}$	i_1
i_2	$\{0, +, 1\}_{i_2}$	i_2
j	$\{\{j_0, +, M * h\}_{i_1}, +, h\}_{i_2}$	$j + (M * h * i_1) + h * i_2$

Multivariate Exponential

Loop counters i_1 and i_2 and induction variable j with multiplication of constant h.

```
for (i1 = 0; i1 <= N; i1++) {
    for (i2 = 0; i2 < M; i2++) {
        j = j * h;
}   }
```

Variable	CR	function
i_1	$\{0, +, 1\}_{i_1}$	i_1
i_2	$\{0, +, 1\}_{i_2}$	i_2
j	$\{\{j_0, *, h^M\}_{i_1}, *, h\}_{i_2}$	$j * h^{i_2} * h^{i_1 * M}$

CR Algebra Rules

- Basic CR is a linear function over $i \in \mathbb{N}$:

$$f(i) = x_0 + i\,h \equiv \{x_0, +, h\}_i$$

- …or a geometric series over $i \in \mathbb{N}$:

$$g(i) = x_0 * h^i \equiv \{x_0, *, h\}_i$$

- Some example algebraic identities:

$$\begin{aligned}
c + \{x_0, +, h\}_i &\Leftrightarrow \{c + x_0, +, h\}_i \\
c * \{x_0, +, h\}_i &\Leftrightarrow \{c\,x_0, *, c\,h\}_i \\
\{x_0, +, h\}_i + \{y_0, +, g\}_i &\Leftrightarrow \{x_0 + y_0, +, g + h\}_i \\
\{x_0, +, h\}_i * \{y_0, +, g\}_i &\Leftrightarrow \{x_0\,y_0, +, g\,h + g\,x + h\,y, +, 2\,g\,h\}_i
\end{aligned}$$

Stride/Start Nesting

$$\begin{aligned}\Phi_i &= \{\phi_0, \odot_1, \{\phi_1, \odot_2, \cdots, \{\phi_{k-1}, \odot_k, f_k\}_i \cdots\}_i\}_i \\ \Phi_i &= \{\{\cdots\{\phi_0, \odot_1, f_1\}_i, \odot_2, f_2\}_i, \cdots\}_i, \odot_k, f_k\}_i\end{aligned}$$

Multiply nested loops generate typically one of these two nested CRs

- recursively specify initial condition of loop by CR for next outer loop
- recursively specify stride of loop by CR for next inner loop

Flattened CR and Loop Forms

All forms can be flattened into a single flattened tuple

$$\Phi_i = \{\phi_0, \odot_1, \phi_1, \odot_2, \cdots, \odot_k, f_k\}_i$$

with $k = L(\Phi_i)$ the length of the CR.

$F[0] := \phi_0; cr_1 := \phi_1; \cdots ; cr_{k-1} := \phi_{k-1}; cr_k := f_k$

for $i = 1$ **to** n **do**

$$\begin{aligned} F[i] &:= F[i-1] \odot_1 cr_1 \\ cr_1 &:= cr_1 \odot_2 cr_2 \\ &\vdots \\ cr_{k-1} &:= cr_{k-1} \odot_k cr_k \end{aligned}$$

od

Normal Forms

$$\Phi_i = \{\phi_0, \odot_1, \phi_1, \odot_2, \cdots, \odot_k, f_k\}_i$$

- Φ_i is *polynomial* or *pure-sum* CR if $\odot_j = +$, $j = 1, \ldots, k$.
- Φ_i is *exponential* or *pure-product* CR if $\odot_j = *$, for all $j = 1, \ldots, k$.
- $\Phi_i = \{\phi_0, *, f_1\}_i$ is *geometric* if f_1 is i-loop invariant.
- Φ_i is a *GIV* if $\odot_j = +$ for $j = 1, \ldots, k-1$ and $\odot_k = *$.
- $\Phi_i = \{\phi_0, *, \phi_1, +, f_2\}_i$ is a *factorial* CR if $\phi_1 \geq 1$ and $f_2 = 1$, or $\phi_1 \leq -1$ and $f_2 = -1$.

Example

Expression $n * j + i + 2 * k + 1$, unit-stride index variables $i \geq 0$ and $j \geq 0$ and GIV $k(i) = (i^2 - i)/2$ and $\Phi(k) = \{0, +, 0, +, 1\}_i$.

Replace i, j and k then normalize.

$$
\begin{aligned}
&\mathcal{CR}(\mathcal{CR}(\mathcal{CR}(n * j + i + 2 * k + 1))) \\
&\quad = \mathcal{CR}(\mathcal{CR}(n * j + \{0, +, 1\}_i + 2 * k + 1)) \\
&\quad = \mathcal{CR}(n * \{0, +, 1\}_j + \{0, +, 1\}_i + 2 * k + 1) \\
&\quad = n * \{0, +, 1\}_j + \{0, +, 1\}_i + 2 * \{0, +, 0, +, 1\}_i + 1 \\
&\quad = \{\{1, +, n\}_j, +, 1, +, 2\}_i
\end{aligned}
$$

Closed Forms

- Polynomial, factorial, GIV, and exponential CRs can always be converted to a closed-form. However, some CRs do not have equivalent closed-forms.
- Rewriting rules terminate with a closed form of the CR expression or showing that one does not exist.
- Polynomial closed form is easily constructed and evaluated in $O(k^2)$ time

$$\chi(i) = \sum_{j=0}^{k} \phi_j \binom{i}{j}$$

Induction Variable Analysis

- Detect (coupled) GIV recurrences RVE [ETAPS CC'01]:

for i = 1 to n do
  j = j+k
  k = k+1

$\Downarrow$ CR

for i = 1 to n do
  $j \equiv \{j_0, +, k_0, +, 1\}_i$
  $k \equiv \{k_0, +, 1\}_i$

$\Downarrow$ Closed form

for i = 1 to n do
  $j \equiv j_0 + i\, k_0 + \frac{i^2 - i}{2}$
  $k \equiv k_0 + i$

Induction Variable Substitution

```
int i, j = 0, k = 0, m = 1;
for (i = a; i <= b; i++) {
  ...
  A[f(i,j,k,m)] = A[g(i,j,k,m)];
  ...
  j = j + k;
  k = k + 1;
  m = m << 1;
  ...
}
```

(a) *Original*

```
int i;
for (i = 0; i <= b-a; i++) {
  ...
  A[f(i+a, (i^2-i)/2, i, 1<<i)]
  = A[g(i+a, (i^2-i)/2, i, 1<<i)]
  ...
}
```

(b) *After IVS*

Note nonlinear indices result.

TRFD Loop

```
ijkl=0
ij=0
DO i=1,m
  DO j=1,i
    ij=ij+1
    ijkl=ijkl+i-j+1
    DO k=i+1,m
      DO l=1,k
        ijkl=ijkl+1
        xijkl[ijkl]=xkl[l]
      ENDDO
    ENDDO
    ijkl=ijkl+ij+left
  ENDDO
ENDDO
```

(a) *Original*

```
DO i=0,m-1
  DO j=0,i-1
    DO k=0,m-i-2
      DO l=0,k
        xijkl[Φ(ijkl)]=xkl[Φ(l)]
      ENDDO
    ENDDO
  ENDDO
ENDDO
```

(b) *GIV Analyzed*

TRFD CRs and Closed Forms

$$\begin{aligned}\Phi(\mathsf{ijkl}) = \; & \{\{\{\{2,+,\mathsf{left}{+}\mathsf{m}(\mathsf{m}{+}1)/2{+}2,+,\mathsf{left}{+}\mathsf{m}(\mathsf{m}{+}1)/2{+}1\}_i \\ & ,+,\mathsf{left}{+}\mathsf{m}(\mathsf{m}{+}1)/2\}_j \\ & ,+,\{2,+,1\}_i,+,1\}_k \\ & ,+,1\}_l\end{aligned}$$

and

$$\Phi(l) = \{1,+,1\}_l$$

$$\begin{aligned}\mathcal{CR}^{-1}(\Phi(\mathsf{ijkl})) = \; & l + i * (k + (\mathsf{m} + \mathsf{m}^2 + 2 * \mathsf{left} + 6)/4) \\ & + j * (\mathsf{left} + (\mathsf{m} + \mathsf{m}^2)/2) \\ & + ((i * \mathsf{m})^2 + \mathsf{m} * i^2)/4 \\ & + (k^2 + 3 * k + i^2 * (\mathsf{left} + 1))/2 + 2\end{aligned}$$

Pointers and Array Recovery

- CR-GIV for pointers RVE/KAG [IWIA'01]
- Array recovery for dependence analysis or data layout
- Facilitates dependence analysis without array recovery

	Access	*PAD*
1	`a[i]`	$\{\mathtt{a}, +, 1\}_i$
2	`a[2*i+1]`	$\{\mathtt{a}{+}1, +, 2\}_i$
3	`a[(i*i-i)/2]`	$\{\mathtt{a}, +, 0, +, 1\}_i$
4	`a[1<<i]`	$\{\mathtt{a}{+}1, +, 1, *, 2\}_i$
5	`p++`	$\{p_0, +, 1\}_i$
6	`q-=2`	$\{q_0, +, -2\}_i$
7	`r+=i`	$\{r_0, +, 0, +, 1\}_i$
8	`s+=2*i`	$\{s_0, +, 0, +, 2\}_i$

GSM Speech CODEC

```
int *f = ..., *lsp = ...;
...
f += 2; lsp += 2;
for (i = 2; i <= 5; i++) {
  *f = f[-2];
  for (j = 1; j < i; j++, f--)
    *f += f[-2]-2*(*lsp)*f[-1];
  *f -= 2*(*lsp);
  f += i; lsp += 2;
}
```

(a) *Original*

```
int *f = ..., *lsp = ...;
...
for (i = 0; i <= 3; i++) {
  *({f+2, +, 1}_i) = *({f, +, 1}_i);
  for (j = 0; j <= {0, +, 1}_i; j++)
    *({{f+2, +, 1}_i, +, −1}_j) += *({{f, +, 1}_i, +, −1}_j)
      - 2 *(*{lsp+2, +, 2}_i)*(*{{f, +, 1}_i, +, −1}_j);
  *(f+1) -= 2*{lsp+2, +, 2}_i;
}
```

(b) *Analyzed*

ETSI `Lsp_Az` Code Segment

GSM Speech CODEC

```
int *f = ..., *lsp = ...;
...
for (i = 0; i <= 3; i++) {
   f[i+2] = f[i];
   for (j = 0; j <= i; j++)
      f[i-j+2] += f[i-j] - 2*lsp[2*i+2]*f[i-j+1];
   f[1] -= 2*lsp[2*i+2];
}
```

(c) *Transformed*

Step Functions

Definition: The *initial value* $V\Phi_i$ of a CR form Φ_i is

$$V\Phi_i = V\{\phi_0, \odot_1, \ldots, \odot_k, \phi_k\}_i = \phi_0$$

Definition: The *step* function $\Delta\Phi_i$ of a CR form Φ_i is

$$\Delta\Phi_i = \Delta\{\phi_0, \odot_1, \phi_1, \odot_2, \ldots, \odot_k, \phi_k\}_i = \{\phi_1, \odot_2, \ldots, \odot_k, \phi_k\}_i$$

Definition: The *direction-wise step* function $\Delta_j\Phi_i$ of a multi-variate CR form Φ_i is the step function with respect to an index variable j

$$\Delta_j\Phi_i = \begin{cases} \Delta\Phi_i & \text{if } i = j \\ \Delta_j V\Phi_i & \text{otherwise} \end{cases}$$

Monotonicity

$$
\begin{aligned}
\Delta_i \Phi(\mathsf{ijkl}) &= \{\mathsf{left}+\mathsf{m}(\mathsf{m}+1)/2+2, +, \mathsf{left}+\mathsf{m}(\mathsf{m}+1)/2+1\}_i \\
\Delta_j \Phi(\mathsf{ijkl}) &= \mathsf{left}+\mathsf{m}(\mathsf{m}+1)/2 \\
\Delta_k \Phi(\mathsf{ijkl}) &= \{\{2, +, 1\}_i, +, 1\}_k \\
\Delta_l \Phi(\mathsf{ijkl}) &= 1
\end{aligned}
$$

- $i \geq 0$ and $k \geq 0 \rightarrow \Delta_k > 0$ and $\Delta_l > 0$
- In the k, l direction the addressing of the xijkl[ijkl] is monotonically increasing
- The inner k, l loop nest can be parallelized.
- ijkl over i, j, k, l index space is monotonically increasing if $\mathsf{left} > \mathsf{m}(\mathsf{m}+1)/2$.
- Closed-forms steps give runtime tests for multiversion code

Value Range Analysis

The accuracy of a range analyzer depends on how the analyzed expression is formed and if monotonicity can be exploited

Blume/Eigenman [IPPS'95], Haghighat [Kluwer'95]

Example without Monotonicity:

$i(i-1)$ and $2^i - i$ are monotonic and non-negative for $0 \le i \le n$.
Without using monotonic properties, bounds can be very loose,

$$
\begin{aligned}
[0,n]([0,n]-1) &= [0,n][-1,n-1] \\
&= [-n,n^2-n] \\
2^{[0,n]} - [0,n] &= [1-n,2^n]
\end{aligned}
$$

Value Range Analysis

- Techniques such as symbolic forward differencing can be used to detect monotonicity but very costly and not always safe.
- CR unified approach generates normal forms and is straightforward to integrate in a compiler framework.
- calculate CR normal forms on index space for GIVs; use the step functions to test monotonicity of expressions for value range analysis.

CR Bounds

Lower and upper CR bounds $L\Phi_i$ and $U\Phi_i$

$$L\Phi_i = \begin{cases} L\,V\Phi_i & \text{if } L\,M\Phi_i \geq 0 \\ L\,\mathcal{CR}_i^{-1}(\Phi_i)[i \leftarrow n] & \text{if } U\,M\Phi_i \leq 0 \\ L\,\mathcal{CR}_i^{-1}(\Phi_i) & \text{otherwise} \end{cases}$$

$$U\Phi_i = \begin{cases} U\,V\Phi_i & \text{if } U\,M\Phi_i \leq 0 \\ U\,\mathcal{CR}_i^{-1}(\Phi_i)[i \leftarrow n] & \text{if } L\,M\Phi_i \geq 0 \\ U\,\mathcal{CR}_i^{-1}(\Phi_i) & \text{otherwise} \end{cases}$$

$$M\Phi_i = \begin{cases} \Delta\Phi_i & \text{if } \odot_1 = + \\ \Delta\Phi_i - 1 & \text{if } \odot_1 = * \wedge L\,V\Phi_1 \geq 0 \wedge L\,\Delta\Phi_i > 0 \\ 1 - \Delta\Phi_i & \text{if } \odot_1 = * \wedge U\,V\Phi_1 < 0 \wedge L\,\Delta\Phi_i > 0 \\ \mathit{undefined} & \text{otherwise} \end{cases}$$

Handles GIVs, multivariate polynomial expressions, and shifts

Value Range Analysis Example

Assume $0 \le i \le n$, $0 \le j \le m$, $k = (i^2 - i)/2$.

$$\begin{aligned} f(i,j,k) &= n * j + i + 2 * k + 1 \\ \mathcal{CR}(f) &= \{\{1,+,n\}_j,+,1,+,2\}_i \end{aligned}$$

The bounds can be derived by straightforward application of the rules:

$$[\, L\{\{1,+,n\}_j,+,1,+,2\}_i \;\;,\;\; U\{\{1,+,n\}_j,+,1,+,2\}_i \,]$$
$$= [\, L\{1,+,n\}_j \;\;,\;\; U(\{1,+,n\}_j + n^2) \,]$$
$$= [\, 1 \;\;,\;\; 1 + m\,n + n^2 \,]$$

Array Region Analysis

Bounds of accesses to an array region facilitate many code optimizations

- eliminating dynamic array bounds checking
- locality optimizations
- prefetching
- blocking

Array region analysis is also used by methods for dependence testing.

Array Region for Lsp_Az

$$
\begin{aligned}
&L\{\{\mathsf{f}{+}2,+,1\}_i,+,-1\}_j\\
&\quad = L\,\mathcal{CR}_j^{-1}(\{\{\mathsf{f}{+}2,+,1\}_i,+,-1\}_j)[j \leftarrow \{0,+,1\}_i]\\
&\quad = L(\{\mathsf{f}{+}2,+,1\}_i - \{0,+,1\}_i)\\
&\quad = \mathsf{f}{+}2\\
&U\{\{\mathsf{f}+2,+,1\}_i,+,-1\}_j\\
&\quad = U\{\mathsf{f}{+}2,+,1\}_i\\
&\quad = U\,\mathcal{CR}_i^{-1}\{\mathsf{f}{+}2,+,1\}_i[i \leftarrow 3]\\
&\quad = \mathsf{f}{+}5
\end{aligned}
$$

- bounds $\mathsf{f}[2..5]$
- iteration space is triangular and the bound on the inner loop variable $j \le i$ is given by the CR form $\{0,+,1\}_i$

Nonlinear dependence analysis

```
p = q = A;
for (i = 0; i < 10; i++) {
   for (j = 0; j <= i; j++)
      *q += *++p;
   q++;
}
```

(a) *Loop Nest*

$$\{0,+,1\}_{i^d} = \{\{1,+,1,+,1\}_{i^u},+,1\}_{j^u}$$

$$0 \le i^d \le 9$$

$$0 \le i^u \le 9$$

$$0 \le j^d \le i^d$$

$$0 \le j^u \le i^u$$

(b) *Dependence Equations*

CR-based Banerjee Bounds Test

- CR-based extreme value test
- direction vector hierarchy by performing subscript-by-subscript testing in multidimensional loops

$$\{\{\{-1,+,1\}_{i^d},+,-1,+,-1\}_{i^u},+,-1\}_{j^u} = 0$$

fbw dependence $\rightarrow i^{\,d} < i^u$ and $j^d < j^u$ and

$$\left.\begin{array}{r}1\\ \{1,+,1\}_{i^d}\end{array}\right\} \le i^u \le 9 \qquad\qquad 0 \le i^d \le \left\{\begin{array}{l}8\\ \{-1,+,1\}_{i^u}\end{array}\right.$$

$$\left.\begin{array}{r}1\\ \{1,+,1\}_{j^d}\end{array}\right\} \le j^u \le \{0,+,1\}_{i^u} \qquad\qquad 0 \le j^d \le \left\{\begin{array}{l}\{-1,+,1\}_{i^d}\\ \{-1,+,1\}_{j^u}\end{array}\right.$$

Lower Bound

$$L\{\{\{-1,+,1\}_{i^d},+,-1,+,-1\}_{i^u},+,-1\}_{j^u}$$

$$= L((\{\{-1,+,1\}_{i^d},+,-1,+,-1\}_{i^u} - j^u)[j^u \leftarrow \{0,+,1\}_{i^u}])$$

$$= L(\{\{-1,+,1\}_{i^d},+,-1,+,-1\}_{i^u} - \{0,+,1\}_{i^u} \qquad \text{(subst.)}$$

$$= L(\{\{-1,+,1\}_{i^d},+,-2,+,-1\}_{i^u} \qquad \text{(simplify)}$$

$$= L((\{-1,+,1\}_{i^d} - (3i^u - (i^u)^2)/2)[i^u \leftarrow 9])$$

$$= L\{-55,+,1\}_{i^d} \qquad \text{(subst.)}$$

$$= -55$$

Upper Bound

$$
\begin{aligned}
& U\{\{\{-1,+,1\}_{i^d},+,-1,+,-1\}_{i^u},+,-1\}_{j^u} \\
&= U\{\{-1,+,1\}_{i^d},+,-1,+,-1\}_{i^u} \\
&= U\{\{-1,+,-1,+,-1\}_{i^u},+,1\}_{i^d} && \text{(swap)} \\
&= U((\{-1,+,-1,+,-1\}_{i^u}+i^d)[i^d \leftarrow \{-1,+,1\}_{i^u}]) \\
&= U(\{-1,+,-1,+,-1\}_{i^u}+\{-1,+,1\}_{i^u}) && \text{(subst.)} \\
&= U\{-2,+,0,+,-1\}_{i^u} && \text{(simplify)} \\
&= -2
\end{aligned}
$$

no flow dependence ($i^d < i^u$ and $j^d < j^u$) since

$$0 \notin [-55 \cdots -2]$$

Conditional Analysis

- extended algorithms to handle conditionally updated variables that may or may not have closed forms
- a *set of* CR forms or PADs for a variable is determined (one for each path)
- combine the set of CR forms to bound the possible range of values of the conditionally updated variables
- use *min* and *max* bounding functions for a set of CR forms over an index space.

Min/Max

Definition: Let $\{\Phi_i^1, \ldots, \Phi_i^n\}$ be a set of n multi-variate polynomial CR forms over i.

The *minimum* CR form is defined by

$$min(\Phi_i^1, \ldots, \Phi_i^n) =$$
$$\{min(V\Phi_i^1, \ldots, V\Phi_i^n), +, min(\Delta\Phi_i^1, \ldots, \Delta\Phi_i^n)\}_i$$

The *maximum* CR form is defined by

$$max(\Phi_i^1, \ldots, \Phi_i^n) =$$
$$\{max(V\Phi_i^1, \ldots, V\Phi_i^n), +, max(\Delta\Phi_i^1, \ldots, \Delta\Phi_i^n)\}_i$$

Conditional Analysis Example

```
p = A; q = B; k = 0; m = 0;
for (i = 0; i <= n; i++)
  if (C[k+2]) {      // Path 1
    for (j = 0; j < i; j++)
      *p++ = *q++;
    k += 2;
  } else {           // Path 2
    *p++ = 0;
    q += i;
    k += m;
    m += 2;
  }
```

A_{p_1} (*Path 1*) CRs

$p = \{A, +, 0, +, 1\}_i$

$q = \{B, +, 0, +, 1\}_i$

$k = \{0, +, 2\}_i$

$m = 0$

A_{p_2} (*Path 2*) CRs

$p = \{A, +, 1\}_i$

$q = \{B, +, 0, +, 1\}_i$

$k = 0$

$k = \{0, +, 0, +, 2\}_i$

$m = \{0, +, 2\}_i$

***Minimum* CRs**

$p \geq A$

$q \geq \{B, +, 0, +, 1\}_i$

$k \geq 0$

$m \geq 0$

***Maximum* CRs**

$p \leq \{A, +, 1, +, 1\}_i$

$q \leq \{B, +, 0, +, 1\}_i$

$k \leq \{0, +, 2, +, 2\}_i$

$m \leq \{0, +, 2\}_i$

Conditional Analysis Example

A_{p_1} (*Path 1*) CRs	*Minimum* CRs	*Minimum* Closed
$\mathsf{p} = \{\mathsf{A}, +, 0, +, 1\}_i$	$\mathsf{p} \geq \mathsf{A}$	$\mathsf{p} \geq \mathsf{A}$
$\mathsf{q} = \{\mathsf{B}, +, 0, +, 1\}_i$	$\mathsf{q} \geq \{\mathsf{B}, +, 0, +, 1\}_i$	$\mathsf{q} \geq \&\mathsf{B}[(i^2 - i)/2]$
$\mathsf{k} = \{0, +, 2\}_i$	$\mathsf{k} \geq 0$	$\mathsf{k} \geq 0$
$\mathsf{m} = 0$	$\mathsf{m} \geq 0$	$\mathsf{m} \geq 0$

A_{p_2} (*Path 2*) CRs	*Maximum* CRs	*Maximum* Closed
$\mathsf{p} = \{\mathsf{A}, +, 1\}_i$	$\mathsf{p} \leq \{\mathsf{A}, +, 1, +, 1\}_i$	$\mathsf{p} \leq \&\mathsf{A}[(i^2 + i)/2]$
$\mathsf{q} = \{\mathsf{B}, +, 0, +, 1\}_i$	$\mathsf{q} \leq \{\mathsf{B}, +, 0, +, 1\}_i$	$\mathsf{q} \leq \&\mathsf{B}[(i^2 - i)/2]$
$\mathsf{k} = 0$	$\mathsf{k} \leq \{0, +, 2, +, 2\}_i$	$\mathsf{k} \leq i^2 + i$
$\mathsf{k} = \{0, +, 0, +, 2\}_i$	$\mathsf{m} \leq \{0, +, 2\}_i$	$\mathsf{m} \leq 2i$
$\mathsf{m} = \{0, +, 2\}_i$		

Conditional Analysis Example

```
p = A; q = B; k = 0; m = 0;
for (i = 0; i <= n; i++)
    if (C[k+2]) {      // Path 1
        for (j = 0; j < i; j++)
            *p++ = *q++;
        k += 2;
    } else {           // Path 2
        *p++ = 0;
        q += i;
        k += m;
        m += 2;
    }
```

$\Delta_i \Phi_i$ *Strides*	$L\Phi_i..U\Phi_i$ *Bounds*
A$[] = 0, \{0, +, 1\}_i$	A$[0..(n^2+n/2)]$
B$[] = \{0, +, 1\}_i$	B$[0..(n^2-n/2)]$
C$[] = 0, \{0, +, 2\}_i$	C$[2..n^2+n+2]$

Dynamic Voltage Scaling

$$T_{dead} = \frac{1}{f} * RWEC$$

$$E_{\mathrm{N}} \approx V^2 (N\, c_L + c_R)$$

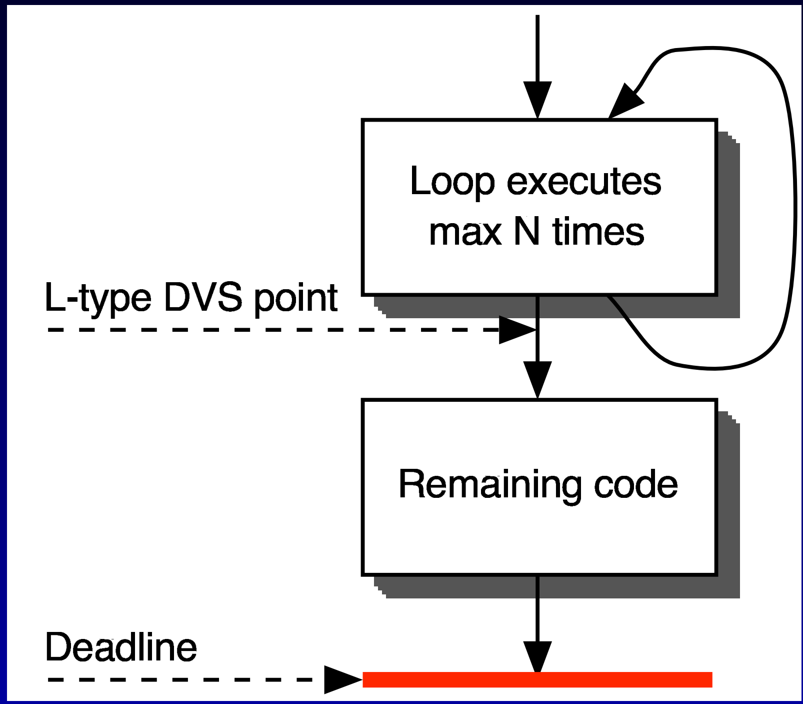
$$E_{\mathrm{act}} \approx V^2 (n\, c_L + c_R)$$

$$E_{\mathrm{Ltype}} \approx V^2 (n\, c_L + \gamma^2 c_R)$$

$$\gamma = \left(\frac{c_R}{(N-n)c_L + c_R}\right)$$

$$E_{\mathrm{Ptype}} \approx \left(V \frac{n\, c_L + c_R}{N\, c_L + c_R}\right)^2 (n\, c_L + c_R)$$

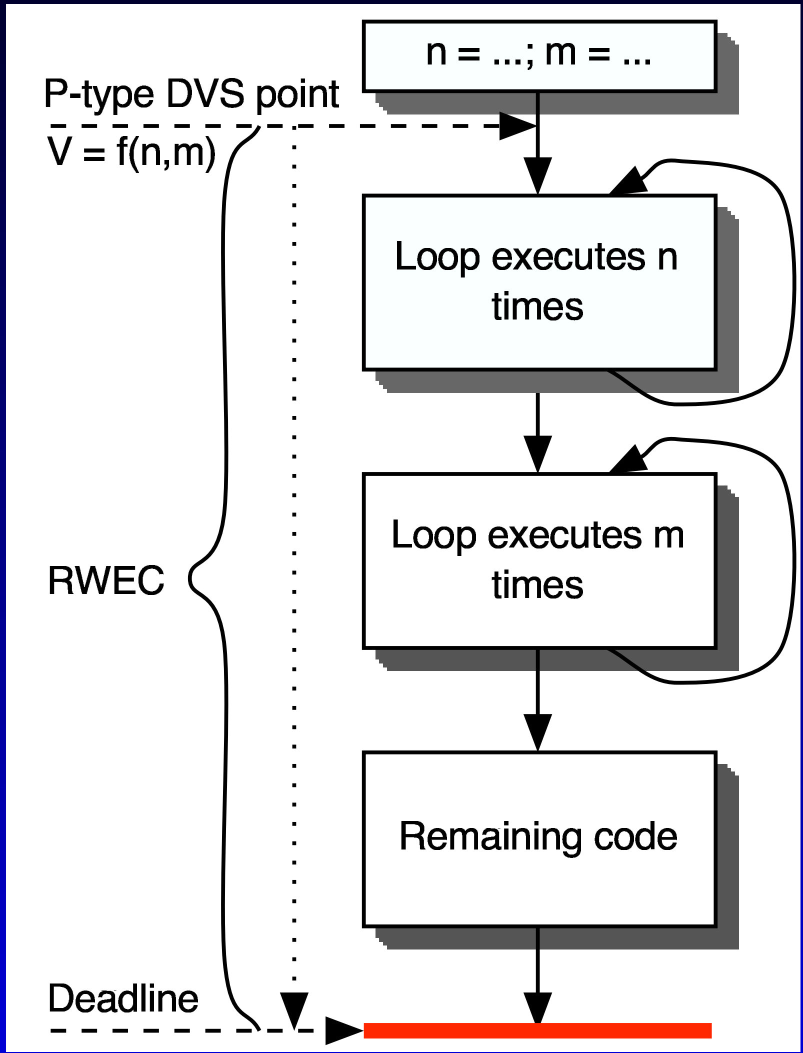
$$E_{\mathrm{Ptype}} \leq E_{\mathrm{Ltype}} \leq E_{\mathrm{act}}$$



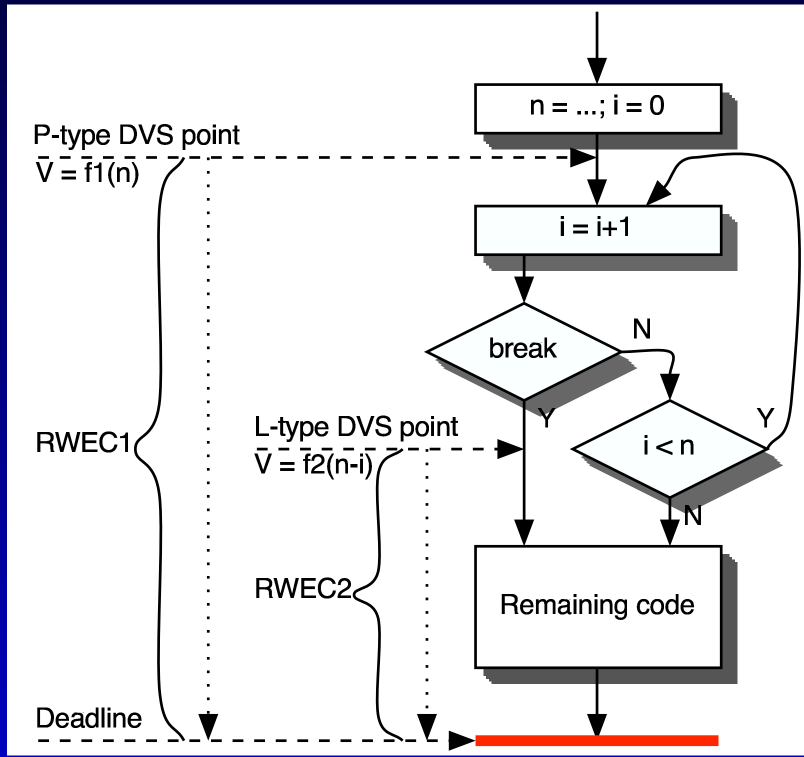
L-Type DVS

Shin/Kim [DAC'01]

P-Type DVS

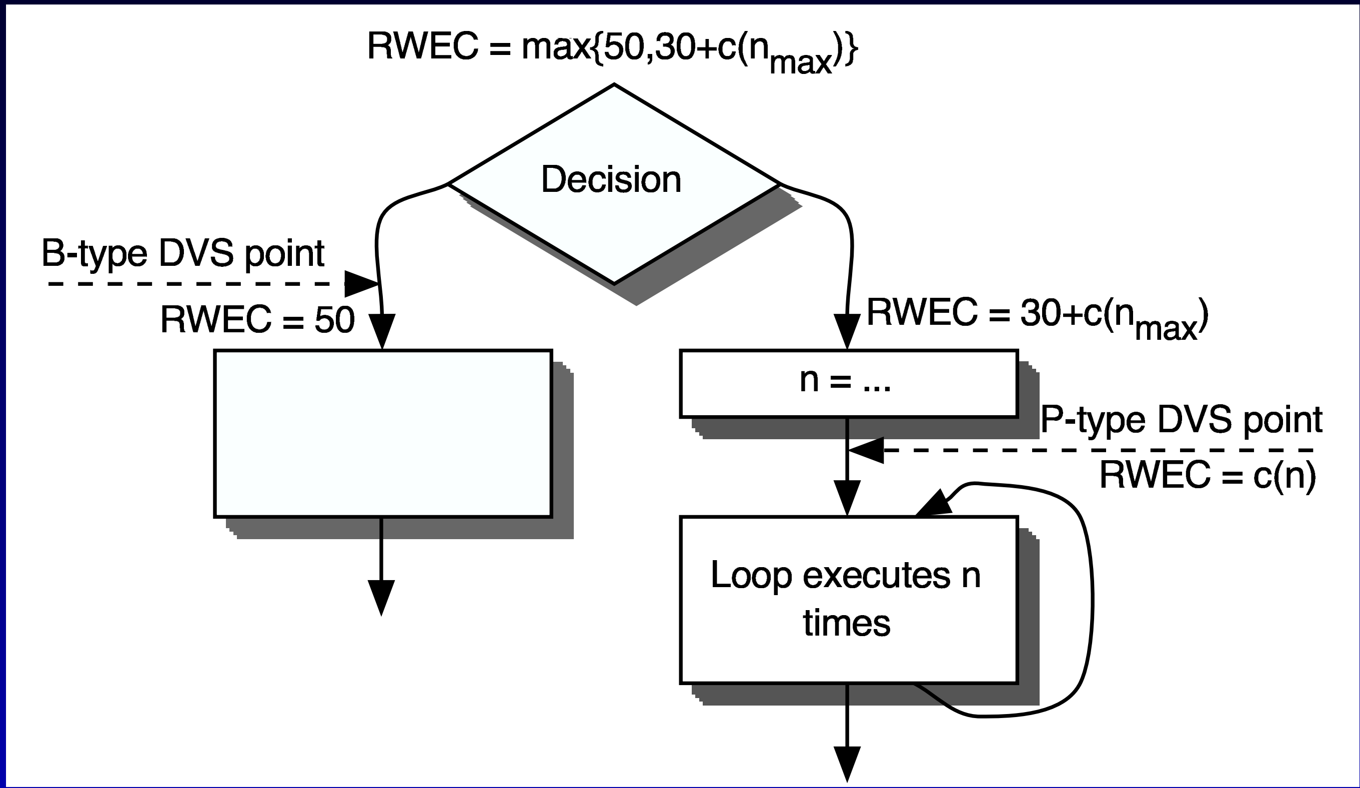


P-Type DVS
RVE/KAG/BW [COLP'03]



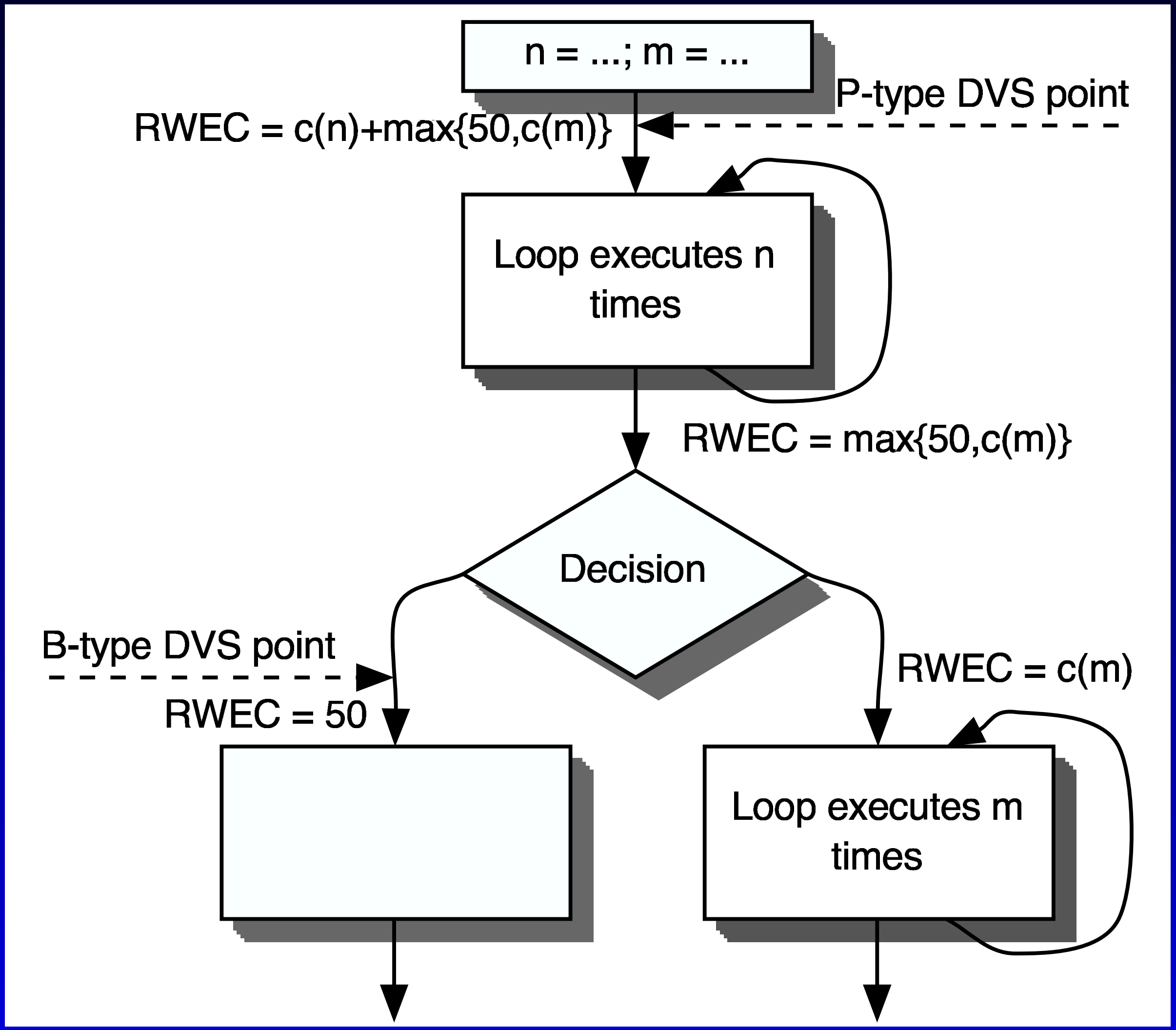
L/P-Type DVS
RVE/KAG/BW[COLP'03]

B and P-Type DVS



B/P-Type DVS
RVE/KAG/BW [COLP'03]

B and P-Type DVS



B/P-Type DVS
RVE/KAG/BW[COLP'03]

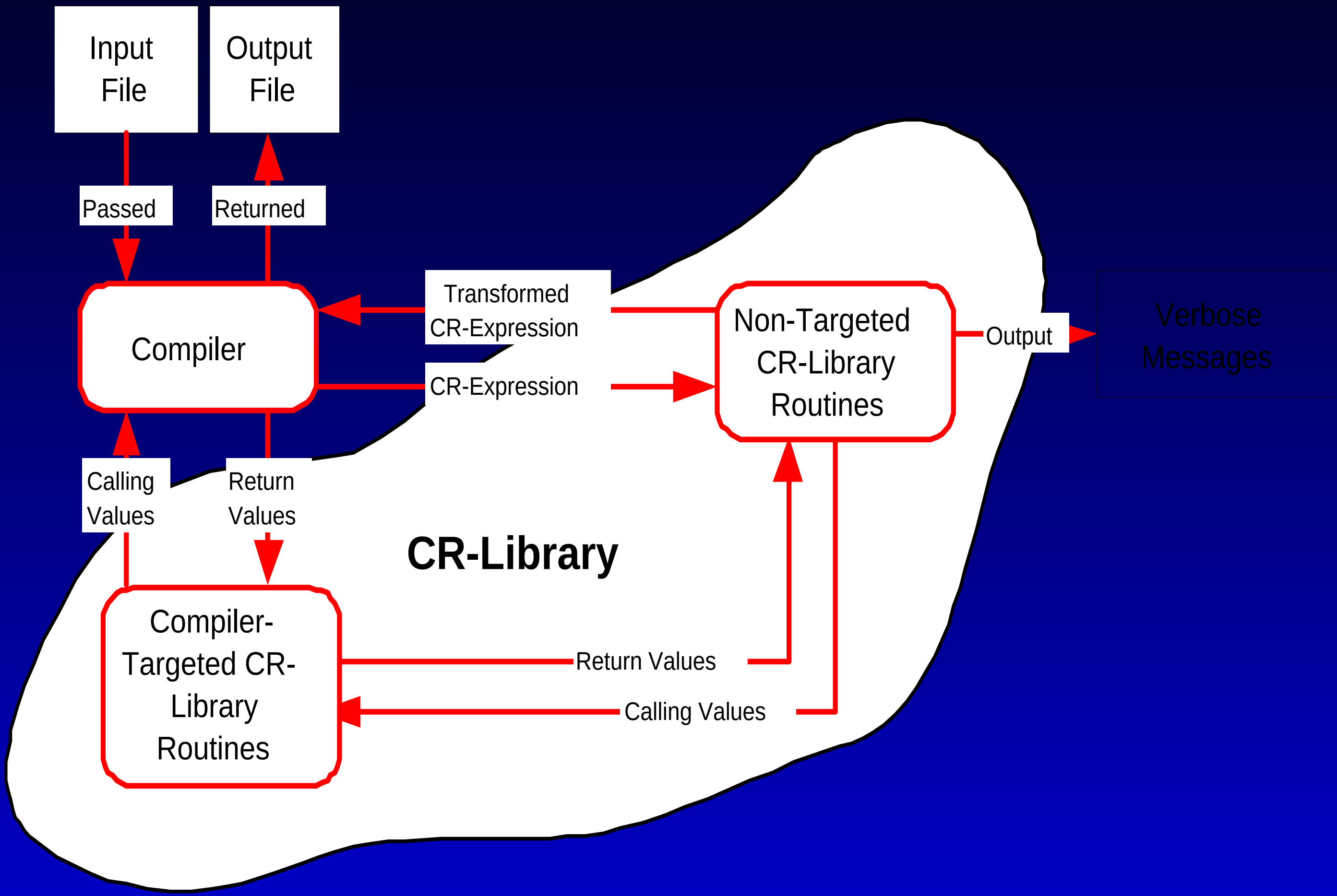
Parametric Time Estimation

- Initial work has been done to apply CR-based weighted summation to timing estimation
- RVE/KAG [IWIA’02] and RVE/KAG/Walsh [CPC’03]
- Based on Newton-Gregory polynomial form and CRs
- currently being evaluated for DSP architectures

Weighted Summation

NEED ONE OF THESE

Unified Framework



Summary

Unified framework for:

- Generalized induction variable analysis
- Pointer arithmetic and array conversion
- Value range and array range analysis
- Conditional GIVs and value range analysis
- Affine and non-linear dependence analysis
- `http://www.csit.fsu.edu/~birch/research/crDemo3.php`
- Parametric execution time estimation of loops
- Energy and power analysis of loop nests

Future Work

- nonlinear GCD for polynomial CRs
- Fourier-Motzkin-like Elimination via CR analysis
- detailed capabaility and efficiency comparison with extant compilers
- high-level language and RTL infrastructure for CR-based framework
- improved accuracy for time and power estimation
- locality analysis and transformation – data layout, blocking
- CR-based scheduling

Separator

THE END

GCD

Goal: Determines the existence of an integer solution to the dependence equation. If the GCD of the left hand side equations evenly divides the right side, dependence is possible.

Positives:

- Efficient, simple to implement
- Helps form general solutions

Negatives:

- The GCD is often 1
- Does not handle coupled subscripts
- Does not consider bounding constraints
- Restricted to affine equations

GCD

Consider the example:

$$S_1 \longrightarrow \quad \text{for } (i = 0; i < 10; i = i + 2)\ \{$$
$$S_2 \longrightarrow \quad A[i] = \ldots$$
$$S_3 \longrightarrow \quad \ldots = A[i+1]$$
$$S_4 \longrightarrow \quad \}$$

Here the dependence equation for S_2 and S_3 is $2i^d - 2i^u = 1$. The GCD of the coefficients of i^d and i^u is 2, which does not evenly divide 1, hence no dependence is possible.

Generalized GCD

Goal: Looks for simultaneous solution to a system of dependence equation, to determine if dependence is possible. The test drives A, in $Ax = b$, to an echelon matrix that can be solved. If an integer solutions exist, dependence is possible.

Positives:

- Finds simultaneous solution to subscripts
- Often produces under-determined parameters which may be used to consider other constraints

Negatives:

- Does not consider bounding constraints
- Restricted to affine equations

Generalized GCD

Consider the example:

$$
\begin{array}{ll}
S_1 \rightarrow & \text{for } (i = 0; i < 10; i = i + 1)\ \{ \\
S_2 \rightarrow & \quad \text{for } (j = 0; j < 10; j = j + 1)\ \{ \\
S_3 \rightarrow & \quad\quad A[i + j + 1][j + 1] = \ldots \\
S_4 \rightarrow & \quad\quad \ldots = A[i + j][j] \\
S_5 \rightarrow & \quad \} \\
S_6 \rightarrow & \}
\end{array}
$$

Where the dependence equations:

$$
\begin{aligned}
i_d - i_u + j_d - j_u &= -1 \\
j_d - j_u &= -1
\end{aligned}
$$

is written in matrix form $\mathbf{Ax} = \mathbf{b}$.

Generalized GCD

Continuing:

Applying only elementary row operations we drive the augmented matrix

$$[\mathbf{U}|\mathbf{A^T}] = \left[\begin{array}{cccc|cc} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & -1 & -1 \end{array}\right]$$

to upper echelon form:

$$[\mathbf{U}|\mathbf{D^T}] = \left[\begin{array}{cccc|cc} 0 & 0 & 0 & -1 & -1 & -1 \\ 0 & 1 & 0 & -1 & 0 & -1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{array}\right].$$

Generalized GCD

Continuing:

Now that we have our echelon $\mathbf{D^T}$ the algorithm tries to solve for integer vector $\mathbf{t}$ in $\mathbf{Dt} = \mathbf{b}$, written out:

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$

, to produce $t_1 = 1$ and $t_2 = 0$, with t_3 and t_4 free variables. Since t is composed of only integers the generalized GCD test demonstrates that there is a simultaneous integer solution to the system, the test returns there is dependence possible.

Banerjee Bounds

Goal: Determines the existence of a real solution to the dependence equation. If the right side constant of the equation lies between the upper bound and lower bound of the left side, dependence is possible.

Positives:

- Efficient, simple to implement
- Determines direction vector hierarchy
- Considers bounding constraints

Negatives:

- Determines real solutions exist, not integer solutions
- Restricted to affine equations

Banerjee Bounds

Consider the example:

$$
\begin{array}{ll}
\mathrm{S}_1 \rightarrow & \text{for } (i = 0; i < 10; i = i + 1) \; \{ \\
\mathrm{S}_2 \rightarrow & \qquad A[2i] = B[i + 6] \\
\mathrm{S}_3 \rightarrow & \qquad D[i] = A[3i - 1] \\
\mathrm{S}_4 \rightarrow & \}
\end{array}
$$

With dependence equation $2i^d - 3i^u = -1$ we compute the upper and lower bounds

$$
\begin{array}{lclclcl}
\text{Lower Bound} & = & 2i^d - 3i^u & \qquad & \text{Upper Bound} & = & 2i^d - 3i^u \\
 & = & 2(0) - 3(10) & & & = & 2(10) - 3(0) \\
 & = & -30 & & & = & 20
\end{array}
$$

Since -1 lies in the range of [-30,20], dependence is possible.

I-Test

Goal: Extends the Banerjee test. Interval equation are produced, and terms are eliminated, updating the interval. Each step, a modified GCD test is performed. If all terms are eliminated or the Banerjee test on remaining terms does not fail, dependence is possible.

Positives:

- Efficient, simple to implement
- Can provide benefits of both GCD and Banerjee

Negatives:

- Determines real solutions exist, not integer
- Does not handle coupled subscripts
- Restricted to affine equations

I-Test

Consider the example:

$S_1 \rightarrow$ for $(i = 0; i < 10; i = i + 1)$ {

$S_2 \rightarrow$ for $(j = 0; j < 10; j = j + 1)$ {

$S_3 \rightarrow$ $A[i] = \ldots$

$S_4 \rightarrow$ $\ldots = A[4j + 8]$

$S_5 \rightarrow$ }

$S_6 \rightarrow$ }

Dependence equation $i^d - 4j^u - 8 = 0$ is converted to interval form $i^d - 4j^u - 8 = [0, 0]$. At each step of elimination, determine if $\lceil L/d \rceil \leq \lfloor U/d \rfloor$ where d is the GCD of the coeffi cients $a_1, a_2, \ldots, a_n$, and $[L, U]$ is the interval. Terms are incorporated into the interval if $|a_n| \leq U - L + 1$.

I-Test

Continuing:

$$
\begin{aligned}
\text{Interval} &= i^d - 4j^u - 8 = [0,0] & \\
&= i^d - 4j^u = [8,8] & \text{rid constant} \\
&= -4j^u = [-1,8] & \text{rid } i^d \text{ since } |1| \le 0 - 0 + 1 \\
&= 0 = [-1,44] & \text{rid } j^u \text{ since } |-4| \le 8 - (-1) + 1
\end{aligned}
$$

Since 0 lies in the range of [-1,44], an integer solution is possible, and hence, dependence is possible.

Fourier Motzkin

Goal: A linear programming method applied to dependence analysis. Test systematically eliminates variables from a system of inequalites until all but a single variable has been eliminated. If no contradictions occur, dependence is possible.

Positives:

- Determines direction vector hierarchy
- Considers bounding constraints

Negatives:

- Determines real solutions exist, not integer
- Can be expensive, not practical.
- Restricted to affine equations

Omega Test

Goal: An integer programming method based on Fourier Motzkin. Variables projected produce shadows based on integer solutions.

Positives:

- Produces fewest inexact returns.
- Determines integer solution.

Negatives:

- Can be expensive, not practical
- Restricted to affine equations

Fourier Motzkin vs. Omega Test

Step 1: Rewrite constraints

$c_1 \rightarrow$	$3x + 4y \geq 16$	$x \geq 16/3 - 47/3$
$c_2 \rightarrow$	$4x + 7y \leq 56$	$x \leq 14 - 7y/4$
$c_3 \rightarrow$	$4x + 7y \leq 20$	$x \leq 5 + 7y/4$
$c_4 \rightarrow$	$2x = 3y \geq -9$	$x \geq -9/2 + 3y/2$

Step 2: Eliminate x

	Fourier Motzkin	Omega
$c_{1,3} \rightarrow$	$5 + 7y/4 \geq 16/3 - 4y/3$	$5 + 7y/4 \geq 16/3 - 4y/3 + 1$
$c_{3,4} \rightarrow$	$5 + 7y/4 \geq 16/3 - 9/2 + 3y/2$	$5 + 7y/4 \geq -9/2 + 3y/2 + 1$
$c_{1,2} \rightarrow$	$14 - 7y/4 \geq 16/3 - 4y/3$	$14 - 7y/4 \geq 16/3 - 4y/3 + 1$
$c_{2,4} \rightarrow$	$14 - 7y/4 \geq -9/2 + 3y/2$	$14 - 7y/4 \geq -9/2 + 3y/2 + 1$

Step 3: Test for contradictions

$4/37 \leq y \leq 74/13 \qquad 16/37 \leq y \leq 66/13$

Both test show dependence is possible.