Efficient Techniques for Advanced Data Dependence Analysis

PACT 2005

Konstantinos Kyriakopoulos and Kleanthis Psarris
Department of Computer Science
The University of Texas at San Antonio
San Antonio, TX 78249

UTSA COMPUTER SCIENCE

Problems of Dependence Analysis

- Traditional analysis techniques oversimplify the dependence problem ignoring:
  - IF-Statements
  - Coupled Subscripts
  - Complex Loop Bounds etc.
- The vast majority of these techniques apply only to linear expressions
- Non-Linear expressions are found very frequently in actual source code
- Large amount of potential parallelism remain unexploited

Improvements in Data Dependence Analysis

- The I-Test
  - Conclusively proves integer solutions
  - Has near linear time complexity
- The Omega Test
  - Considers if-statement constraints
  - Can provide simultaneous solution across coupled subscripts
  - Can handle complex loop regions (triangular/trapezoidal/symbolic)
  - Can prove dependences
- The Range Test
  - Can handle non-linear expressions
  - Can handle complex loop regions

Limitations of Data Dependence Analysis Techniques

- **The I-Test**
  - Cannot handle non-linear expressions
  - Does not consider if-statements
  - Cannot handle coupled subscripts
  - Cannot handle complex loops regions
- **The Omega Test**
  - Cannot handle non-linear expressions
  - Has exponential time complexity
- **The Range Test**
  - Does not produce complete direction vector information
  - Cannot handle coupled subscripts
  - Cannot prove dependences

The NLVI-Test

- Can handle non-linear expressions
- Can handle simple if-statement constraints
- Can minimize or eliminate coupling across equations
- Can handle complex loop regions such as trapezoidal/symbolic
- Can produce accurate and complete direction vector information
- Can conclusively prove or disprove dependences
- Has polynomial time complexity

Variable Interval

- **Integer Interval**

  $[L, U] = \{L, L + 1, \ldots, U\}$

  e.g. $[1, 4] = \{1, 2, 3, 4\}$

- **Single Variable Integer Interval**

  $[L(X), U(X)]$ where $P \le X \le Q$ =

  $[L(P), U(P)] \cup [L(P + 1), U(P + 1)] \cup \ldots \cup [L(Q), U(Q)]$

  e.g.

  $[X, 3X]$, $1 \le X \le 3$ =

  $[1, 3] \cup [2, 6] \cup [3, 9]$ =

  $[1, 9] = \{1, 2, \ldots, 9\}$.

- **General Variable Integer Interval**

  $[L(\boldsymbol{x}), U(\boldsymbol{x})]$ where $\boldsymbol{x}$ in $\Re$ =

  $$\bigcup_{\boldsymbol{x}_i \in \Re} [L(\boldsymbol{x}_i), U(\boldsymbol{x}_i)]$$

  e.g.

  $[2X_1X_2 - 1, 3X_1X_2 + X_1]$,
  where $1 \le X_1 \le 5$, $1 \le X_2 \le X_1^2$

Variable Interval Theory

- Applies to linear and non-linear expressions
- It relies on the monotonicity of multivariable functions
- Provides the conditions under which a variable integer interval can be contiguous

Fundamental Theorem

Given a single variable integer interval $[L(X), U(X)]$, where $P \leq X \leq Q$:

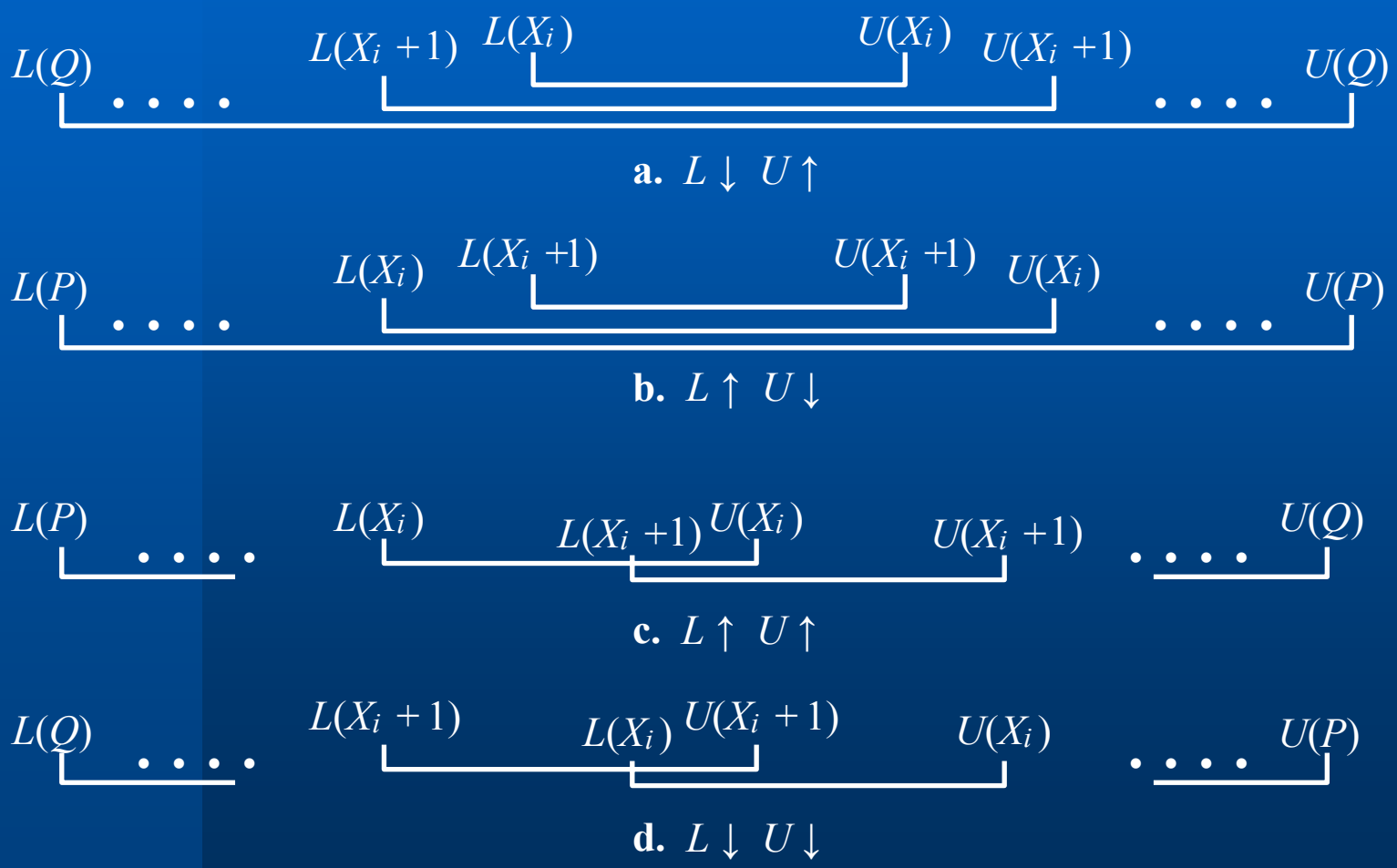
a. If L is decreasing and U is increasing, then $[L(X), U(X)]$ is equal to the integer interval $[L(Q), U(Q)]$.

b. If L is increasing and U is decreasing, then $[L(X), U(X)]$ is equal to the integer interval $[L(P), U(P)]$.

c. If L and U are increasing and $U(X_i) - L(X_i + 1) + 1 \geq 0$ for all X_i, $P \leq X_i \leq Q - 1$, then $[L(X), U(X)]$ is equal to the integer interval $[L(P), U(Q)]$.

d. If L and U are decreasing and $U(X_i + 1) - L(X_i) + 1 \geq 0$ for all X_i, $P \leq X_i \leq Q - 1$, then $[L(X), U(X)]$ is equal to the integer interval $[L(Q), U(P)]$.

Intuition Behind Theorem



Monotonicity of Multi-variable Functions

- Increasing for variable function $F(\boldsymbol{x}, X)$ for variable X
  - $F(\boldsymbol{x}_i, X_j) \leq F(\boldsymbol{x}_i, X_j + 1)$ for any $\boldsymbol{x}_i \in \mathrm{Z}^n$ and for any $X_j \in \mathrm{Z}$
- Decreasing for variable function $F(\boldsymbol{x}, X)$ for variable X
  - $F(\boldsymbol{x}_i, X_j) \geq F(\boldsymbol{x}_i, X_j + 1)$ for any $\boldsymbol{x}_i \in \mathrm{Z}^n$ and for any $X_j \in \mathrm{Z}$
- If F is defined in a region subset of Z^{n+1} then the conditions need only apply for values of $(\boldsymbol{x}, X)$ in that region
- F can be proven increasing if $\min(F(\boldsymbol{x}, X + 1) - F(\boldsymbol{x}, X)) \geq 0$
- F can be proven decreasing if $\max(F(\boldsymbol{x}, X + 1) - F(\boldsymbol{x}, X)) \leq 0$
- For more efficiency we can use the first partial derivative
  - $\min(\mathrm{d}F(\boldsymbol{x}, X)/\mathrm{dX}) \geq 0$: Increasing
  - $\max(\mathrm{d}F(\boldsymbol{x}, X)/\mathrm{dX}) \leq 0$: Decreasing

Mutli-variable Interval Theorem

Given a variable integer interval $[L(\boldsymbol{x}, X), U(\boldsymbol{x}, X)]$ subject to a set of constraints on $\boldsymbol{x}$ in $\Re$ and the constraint $P(\boldsymbol{x}) \leq X \leq Q(\boldsymbol{x})$, where X does not appear in any constraints in $\Re$:

a. If L is decreasing and U is increasing for variable X, then $[L(\boldsymbol{x}, X), U(\boldsymbol{x}, X)]$ is equal to the variable integer interval $[L(\boldsymbol{x}, Q(\boldsymbol{x})), U(\boldsymbol{x}, Q(\boldsymbol{x}))]$, where $\boldsymbol{x}$ in $\Re$ and $P(\boldsymbol{x}) \leq Q(\boldsymbol{x})$.

b. If L is increasing and U is decreasing for variable X, then $[L(\boldsymbol{x}, X), U(\boldsymbol{x}, X)]$ is equal to the variable integer interval $[L(\boldsymbol{x}, P(\boldsymbol{x})), U(\boldsymbol{x}, P(\boldsymbol{x}))]$, where $\boldsymbol{x}$ in $\Re$ and $P(\boldsymbol{x}) \leq Q(\boldsymbol{x})$.

c. If L and U are increasing for variable X and $\min(U(\boldsymbol{x}, X) - L(\boldsymbol{x}, X + 1) + 1) \geq 0$, then $[L(\boldsymbol{x}, X), U(\boldsymbol{x}, X)]$ is equal to the variable integer interval $[L(\boldsymbol{x}, P(\boldsymbol{x})), U(\boldsymbol{x}, Q(\boldsymbol{x}))]$, where $\boldsymbol{x}$ in $\Re$ and $P(\boldsymbol{x}) \leq Q(\boldsymbol{x})$.

d. If L and U are decreasing for variable X and $\min(U(\boldsymbol{x}, X + 1) - L(\boldsymbol{x}, X) + 1) \geq 0$, then $[L(\boldsymbol{x}, X), U(\boldsymbol{x}, X)]$ is equal to the variable integer interval $[L(\boldsymbol{x}, Q(\boldsymbol{x})), U(\boldsymbol{x}, P(\boldsymbol{x}))]$, where $\boldsymbol{x}$ in $\Re$ and $P(\boldsymbol{x}) \leq Q(\boldsymbol{x})$.

General Loop Structure

```
for I1 = p1 to q1 do
  for I2 = p2(I1) to q2(I1) do
    for I3 = p3(I1, I2) to q3(I1, I2) do
      •
      •
      for In = pn(I1, I2, …,In-1) to qn(I1, I2, …,In-1) do
s1:                 A[f(I1, I2, …,In)] = …
s2:                                    … = A[g(I1, I2, …,In)]
      endfor
      •
      •
    endfor
  endfor
endfor
```

Handling General Loop Regions

- **General loop nest dependence constraints**

$$f(I_1, I_2, \ldots I_n) = g(I'_1, I'_2, \ldots, I'_n)$$

$$p_k(I_1, I_2, \ldots I_{k-1}) \le I_k \le q_k(I_1, I_2, \ldots I_{k-1})$$

$$p_k(I'_1, I'_2, \ldots I'_{k-1}) \le I'_k \le q_k(I'_1, I'_2, \ldots I'_{k-1}), \quad 1 \le k \le n.$$

- **The can also be represented as**

$$F(\boldsymbol{x}) = 0$$

$$P_{2k-1}(\boldsymbol{x}) \le X_{2k-1} \le Q_{2k-1}(\boldsymbol{x}) \qquad 1 \le k \le n$$

$$P_{2k}(\boldsymbol{x}) \le X_{2k} \le Q_{2k}(\boldsymbol{x}) \qquad \boldsymbol{x} = (X_1, X_2, \ldots, X_{2n}),$$

Handling General Loop Regions

- **Start from variable interval equation**
- $F(\boldsymbol{x}) = [0, 0]$
- **Eliminate one variable at a time from**
  $F(\boldsymbol{x}) = [L(\boldsymbol{x}, X), U(\boldsymbol{x}, X)], P(\boldsymbol{x}) \le X \le Q(\boldsymbol{x})$
- **Use a DAG to determine the order of variable elimination**
- **Check for Accuracy Condition 1**
  L, U should have opposite monotonicity for X or else
  - if they are both increasing, then $\min(L(\boldsymbol{x}, X) - U(\boldsymbol{x}, X + 1) + 1) \ge 0$
  - if they are both decreasing, then $\min(L(\boldsymbol{x}, X + 1) - U(\boldsymbol{x}, X) + 1) \ge 0$
- **Check for Accuracy Condition 2**
  $\min(Q(\boldsymbol{x}) - P(\boldsymbol{x})) \ge 0$
- **Compare the final integer interval with zero**

Handling Direction Vector Constraints

- **Additional direction vector constraint**

$$P_{2k-1}(\boldsymbol{x}) \le X_{2k-1} \le Q_{2k-1}(\boldsymbol{x})$$
$$P_{2k}(\boldsymbol{x}) \le X_{2k} \le Q_{2k}(\boldsymbol{x})$$
$$X_{2k-1} < X_{2k} \qquad 1 \le k \le n$$

- **Two orders of elimination**

$$P_{2k-1}(\boldsymbol{x}) \le X_{2k-1} \le \min(Q_{2k-1}(\boldsymbol{x}),\ Q_{2k}(\boldsymbol{x}) - 1)$$
$$\max(P_{2k}(\boldsymbol{x}),\ X_{2k-1} + 1) \le X_{2k} \le Q_{2k}(\boldsymbol{x})$$

OR

$$P_{2k-1}(\boldsymbol{x}) \le X_{2k-1} \le \min(Q_{2k-1}(\boldsymbol{x}),\ X_{2k} - 1)$$
$$\max(P_{2k}(\boldsymbol{x}),\ P_{2k-1}(\boldsymbol{x}) + 1) \le X_{2k} \le Q_{2k}(\boldsymbol{x})$$

- **Select bounds using the table**
- **Proceed with the NLVI-Test algorithm**

Conditions for Selecting Direction Vectors

v_k = "<"					
Conditions		**Bounds**	**Conditions**		**Bounds**
$P_{2k}(\boldsymbol{x}) \le P_{2k-1}(\boldsymbol{x}) + 1$			$Q_{2k-1}(\boldsymbol{x}) \le Q_{2k}(\boldsymbol{x}) - 1$		
	$Q_{2k-1}(\boldsymbol{x}) \le Q_{2k}(\boldsymbol{x}) - 1$	$P_{2k-1}(\boldsymbol{x}) \le X_{2k-1} \le Q_{2k}(\boldsymbol{x})-1$ $X_{2k-1} + 1 \le X_{2k} \le Q_{2k}(\boldsymbol{x})$		$P_{2k}(\boldsymbol{x}) \le P_{2k-1}(\boldsymbol{x}) + 1$	$P_{2k-1}(\boldsymbol{x}) \le X_{2k-1} \le X_{2k} - 1$ $P_{2k-1}(\boldsymbol{x}) + 1 \le X_{2k} \le Q_{2k}(\boldsymbol{x})$
	$Q_{2k-1}(\boldsymbol{x}) \le Q_{2k}(\boldsymbol{x}) - 1$	$P_{2k-1}(\boldsymbol{x}) \le X_{2k-1} \le Q_{2k-1}(\boldsymbol{x})$ $X_{2k-1} + 1 \le X_{2k} \le Q_{2k}(\boldsymbol{x})$		$P_{2k}(\boldsymbol{x}) \le P_{2k-1}(\boldsymbol{x}) + 1$	$P_{2k-1}(\boldsymbol{x}) \le X_{2k-1} \le X_{2k} - 1$ $P_{2k}(\boldsymbol{x}) \le X_{2k} \le Q_{2k}(\boldsymbol{x})$
v_k = ">"					
Conditions		**Bounds**	**Conditions**		**Bounds**
$P_{2k-1}(\boldsymbol{x}) \le P_{2k}(\boldsymbol{x}) + 1$			$Q_{2k}(\boldsymbol{x}) \le Q_{2k-1}(\boldsymbol{x}) - 1$		
	$Q_{2k}(\boldsymbol{x}) \le Q_{2k-1}(\boldsymbol{x}) - 1$	$X_{2k} + 1 \le X_{2k-1} \le Q_{2k-1}(\boldsymbol{x})$ $P_{2k}(\boldsymbol{x}) \le X_{2k} \le Q_{2k-1}(\boldsymbol{x}) - 1$		$P_{2k-1}(\boldsymbol{x}) \le P_{2k}(\boldsymbol{x}) + 1$	$P_{2k}(\boldsymbol{x}) + 1 \le X_{2k-1} \le Q_{2k-1}(\boldsymbol{x})$ $P_{2k}(\boldsymbol{x}) \le X_{2k} \le X_{2k-1} - 1$
	$Q_{2k}(\boldsymbol{x}) \le Q_{2k-1}(\boldsymbol{x}) - 1$	$X_{2k} + 1 \le X_{2k-1} \le Q_{2k-1}(\boldsymbol{x})$ $P_{2k}(\boldsymbol{x}) \le X_{2k} \le Q_{2k}(\boldsymbol{x})$		$P_{2k-1}(\boldsymbol{x}) \le P_{2k}(\boldsymbol{x}) + 1$	$P_{2k-1}(\boldsymbol{x}) \le X_{2k-1} \le Q_{2k-1}(\boldsymbol{x})$ $P_{2k}(\boldsymbol{x}) \le X_{2k} \le X_{2k-1} - 1$
v_k = "="					
Conditions		**Bounds**	**Co nditions**		**Bounds**
$P_{2k}(\boldsymbol{x}) \le P_{2k-1}(\boldsymbol{x})$			$P_{2k}(\boldsymbol{x}) \le P_{2k-1}(\boldsymbol{x})$		
	$Q_{2k}(\boldsymbol{x}) \le Q_{2k-1}(\boldsymbol{x})$	$P_{2k-1}(\boldsymbol{x}) \le X_{2k-1} \le Q_{2k-1}(\boldsymbol{x})$		$Q_{2k}(\boldsymbol{x}) \le Q_{2k-1}(\boldsymbol{x})$	$P_{2k}(\boldsymbol{x}) \le X_{2k-1} \le Q_{2k-1}(\boldsymbol{x})$
	$Q_{2k}(\boldsymbol{x}) \le Q_{2k-1}(\boldsymbol{x})$	$P_{2k-1}(\boldsymbol{x}) \le X_{2k-1} \le Q_{2k}(\boldsymbol{x})$		$Q_{2k}(\boldsymbol{x}) \le Q_{2k-1}(\boldsymbol{x})$	$P_{2k}(\boldsymbol{x}) \le X_{2k-1} \le Q_{2k}(\boldsymbol{x})$

Computation of Minimum and Maximum

- **Utilized throughout the NLVI-Test algorithm**
- **Used to compare expressions.**

  $\min(Q(\boldsymbol{x}) - P(\boldsymbol{x})) \geq 0 \Leftrightarrow Q(\boldsymbol{x}) \geq P(\boldsymbol{x})$ for all $\boldsymbol{x}$ **in** $\Re$

  $\max(Q(\boldsymbol{x}) - P(\boldsymbol{x})) \leq 0 \Leftrightarrow Q(\boldsymbol{x}) \leq P(\boldsymbol{x})$ for all $\boldsymbol{x}$ **in** $\Re$
- **Utilize a Variable Substitution Algorithm**
- **Use the monotonicity of the expression**
  - if $F(\boldsymbol{x}, X)$ and is $\uparrow$ for X where $P(\boldsymbol{x}) \leq X \leq Q(\boldsymbol{x})$
    then $\max(F(\boldsymbol{x}, X)) = \max(F(\boldsymbol{x}, Q(\boldsymbol{x}))$
    and $\min(F(\boldsymbol{x}, X)) = \min(F(\boldsymbol{x}, P(\boldsymbol{x}))$
  - if $F(\boldsymbol{x}, X)$ and is $\downarrow$ for X where $P(\boldsymbol{x}) \leq X \leq Q(\boldsymbol{x})$
    then $\max(F(\boldsymbol{x}, X)) = \max(F(\boldsymbol{x}, P(\boldsymbol{x}))$
    and $\min(F(\boldsymbol{x}, X)) = \min(F(\boldsymbol{x}, Q(\boldsymbol{x}))$

Computation of Minimum and Maximum

- **The computation of monotonicity relies on the computation of min and max**
- **Each procedure calls the other recursively**
  - $F(\boldsymbol{x}, X)$ and is $\uparrow$ for X iff $\min(\mathrm{d}F(\boldsymbol{x}, X)) \geq 0$
  - $F(\boldsymbol{x}, X)$ and is $\downarrow$ for X iff $\max(\mathrm{d}F(\boldsymbol{x}, X)) \leq 0$
- **The halting condition depends on the type of functions used**
  - **For polynomials the level of recursion is the degree of the polynomial**
- **A DAG is used to determine the order of substitution**
  - **Any topological short is a valid sequence**

Coupled Subscripts and If-Statements

- **Apply the equation propagation technique while considering if-statement constraints:**

```
if(N > 1) then
  for I = 1 to N do
    for J = 1 to I*I do
     A[2*N*I - J + N, N*I - J + 1] = A[2*J, J]
    endfor
  endfor
endif
```

- **The dependence problem constraints:**

$$2NX_1 - X_3 - 2X_4 + N = 0 \qquad 2 \le N \le +\infty$$

$$NX_1 - X_3 - X_4 = -1 \qquad 1 \le X_1 \le N, \qquad 1 \le X_2 \le N$$

$$1 \le X_3 \le X_1^2, \qquad 1 \le X_4 \le X_2^2$$

- **Equation propagation produces**

$$N + X_3 = 2$$

- **Dependence is disproved**

Real Code Examples

```
DO n0 = 1, n, 1
 t(n0) = t(n0) + (-c(n0))*c0
 t(n+n0) = t(n+n0) + (-c(n0))*c1
 t(n0+2*n) = t(n0+2*n) + (-c(n0))*c2
 t(n0+3*n) = t(n0+3*n) + (-c(n0))*c3
 t(n0+4*n) = t(n0+4*n) + (-c(n0))*c4
 t(n0+5*n) = t(n0+5*n) + (-c(n0))*c5
ENDDO
```

- **BDNA of Perfect**
  - **Symbolic Loop Bounds**
  - **I-Test cannot parallelize**
  - **NLVI-Test, Omega and Range can parallelize loop and increase speedup**

Real Code Examples

```
DO mrs = 1, (num*(num+1))/2, 1
 DO mi = 1, num, 1
  DO mj = 1, mi, 1
   xrsij(mj+(mi**2-mi-num-num**2+mrs*num+mrs*num**2)/2)
   = xij(mj)
  ENDDO
 ENDDO
ENDDO
```

- **TRFD of Perfect**
  - **Non-Linear Expressions**
  - **I-Test, Omega cannot parallelize**
  - **NLVI-Test and Range can parallelize loops and produce speedup**

Real Code Examples

```
DO k = 2, lmi-1, 1
 DO i3 = 1, mm(k), 1
  DO i2 = 1, mm(k), 1
   DO i1 = 1, mm(k), 1
    u(-1+ir(k)+i1-mm(k)-mm(k)**2+mm(k)*i2+i3*mm(k)**2)
  = 0
   ENDDO
  ENDDO
 ENDDO
ENDDO
```

- **MGRID of SPEC**
  - **Non-Affine loop bounds and Non-Linear expression**
  - **I-Test, Omega and Range cannot parallelize loop *i3***
  - **NLVI-Test can parallelize *i3* and increase speedup**

Real Code Examples

```
DO j = 3, n-1, 1
 DO i = 2, n-1, 1
  r = aa(i,j)*d(i,j-1)
  d(i,j) = 1.D0/(dd(i,j)-aa(i,j-1)*r)
  rx(i,j) = rx(i,j)-rx(i,j-1)*r
  ry(i,j) = ry(i,j)-ry(i,j-1)*r
 ENDDO
ENDDO
```

- **TOMCATV of SPEC**
  - **Dependence with (<, =) direction vector**
  - **Range test detects only (<, *)**
  - **Range test cannot interchange loops**
  - **NLVI-Test, I-Test and Omega interchange loop and increase speedup**

Implementation Details

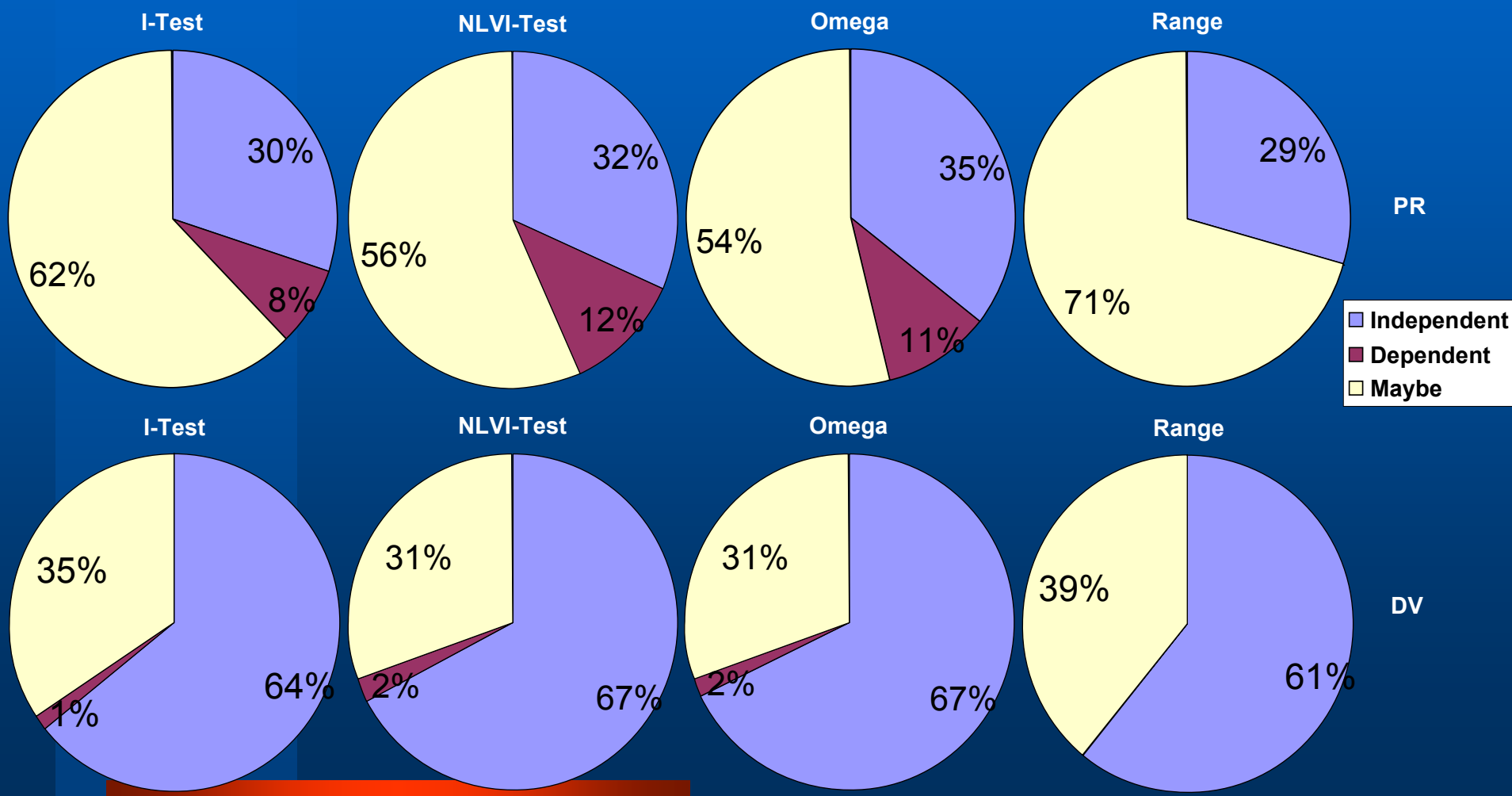
- We implemented a portable library that contains several data dependence tests. The PLATO library contains:
  - Banerjee Test
  - I-Test
  - VI-Test for linear expressions
  - NLVI-Test for polynomial and rational polynomial expressions
- We used the Polaris Compiler Infrastructure

- Programs were transformed
  - Constant Propagation
  - Induction variable recognition
  - Reduction
  - Scalar and array privatization
  - Loop Normalization (etc)
- We also implemented a simple loop interchange transformation

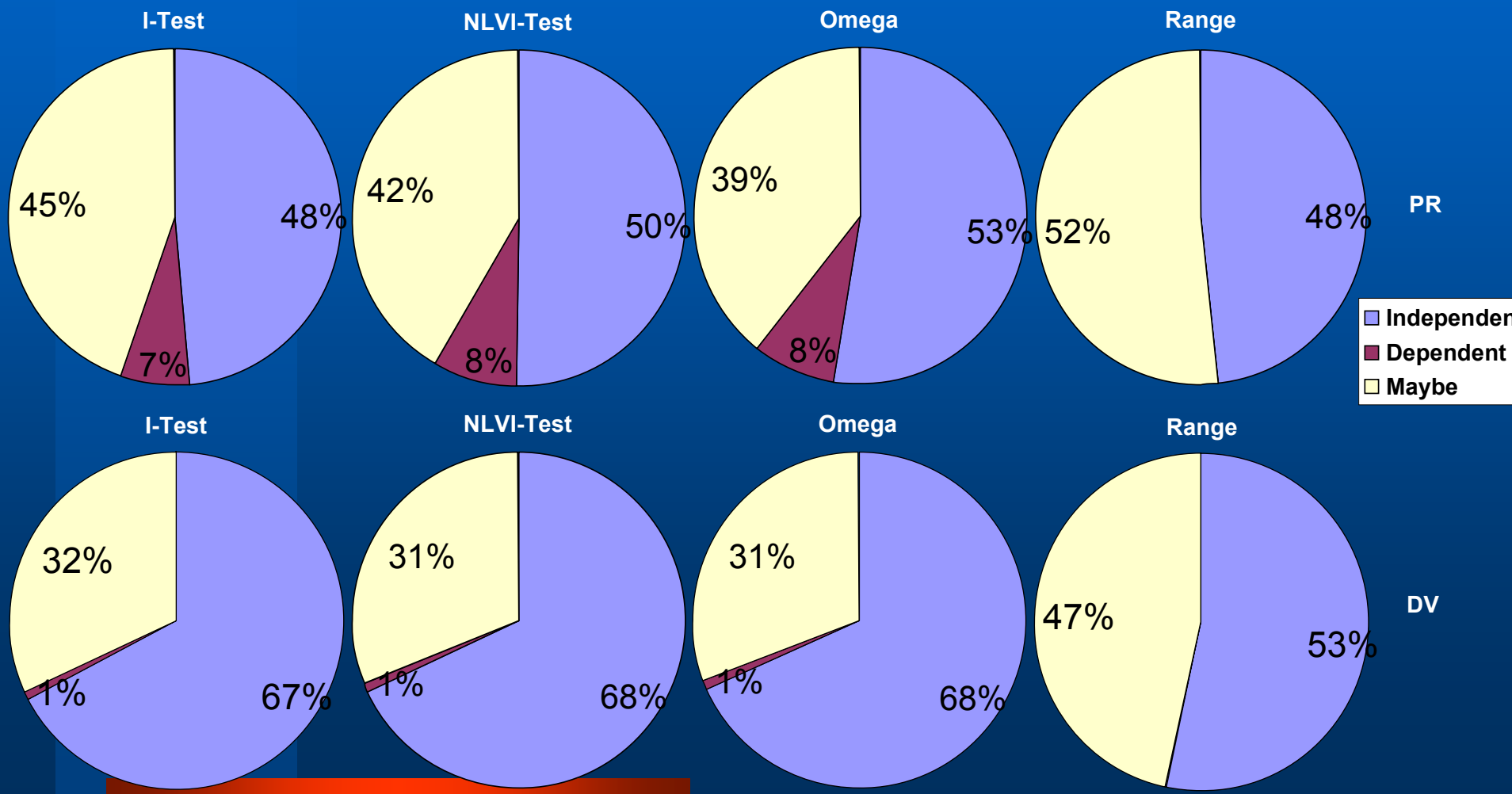
Experiments

- We compare 4 Data Dependence tests
  - I-Test
  - NLVI-Test
  - Omega
  - Range
- We compare in terms of
  - Data Dependence Accuracy
  - Compilation efficiency
  - Loop Parallelization
  - Program Execution Performance
- We use two Benchmark Suites
  - The Perfect Benchmarks
  - The SPEC Benchmarks

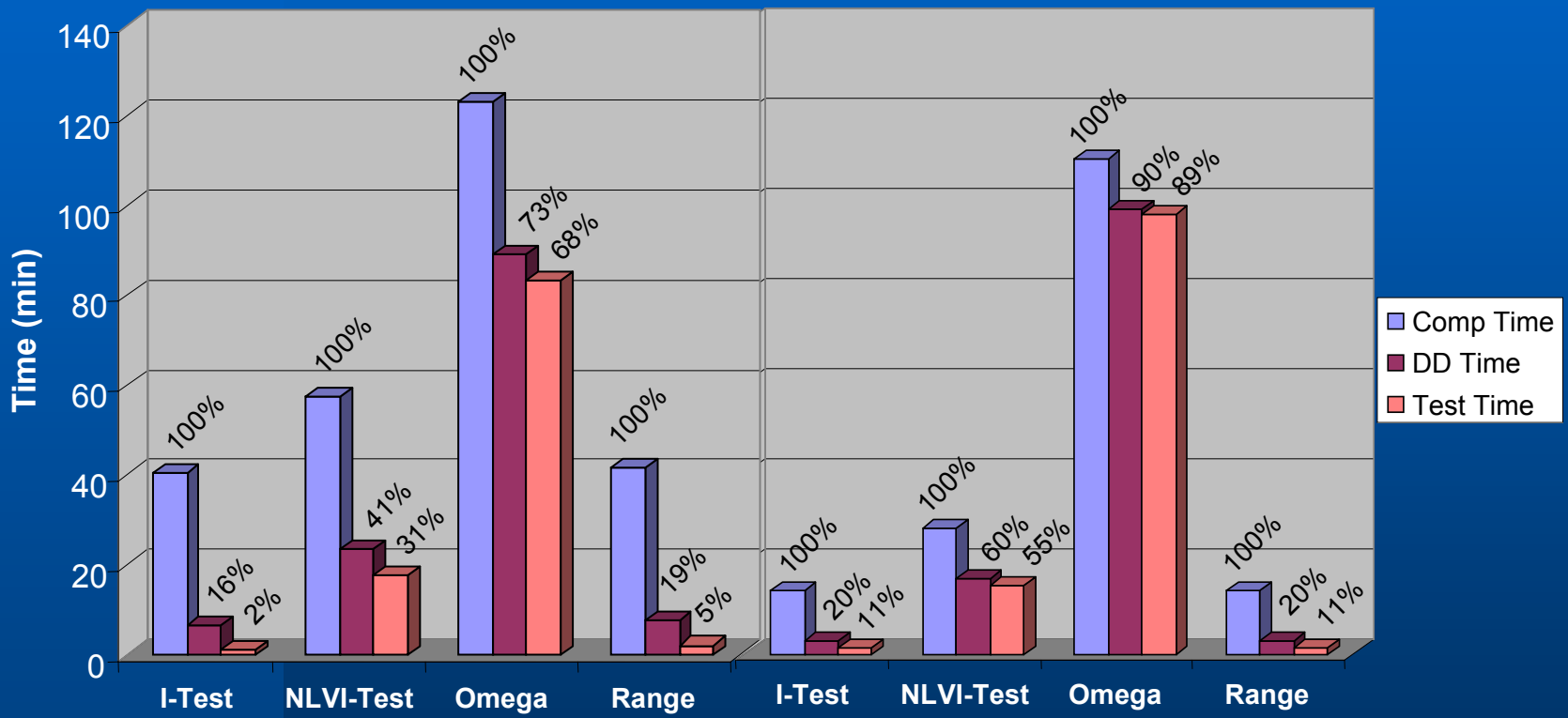
Data Dependence Accuracy in the Perfect Benchmarks



Data Dependence Accuracy in the SPEC Benchmarks



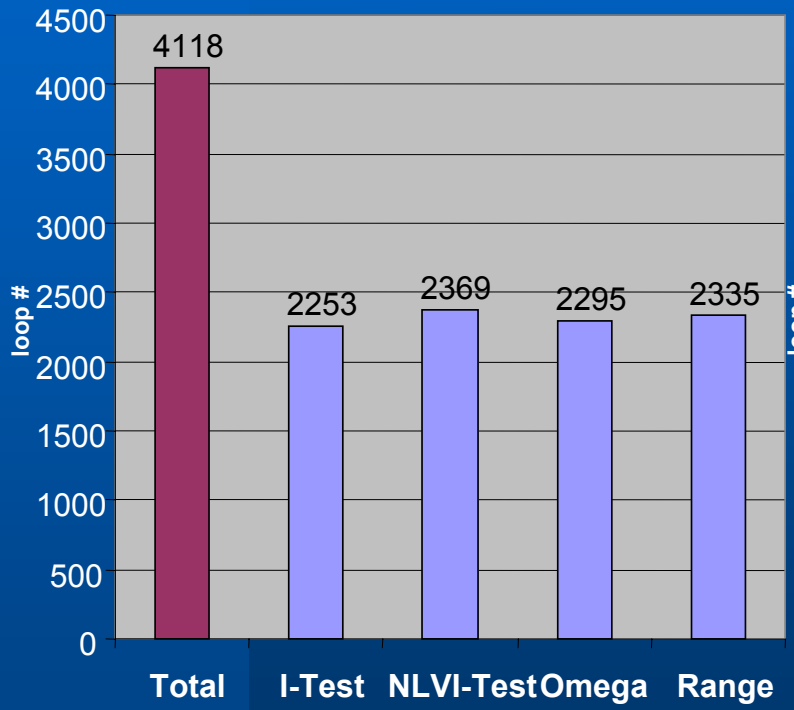
Compilation Efficiency in the Perfect and SPEC Benchmarks

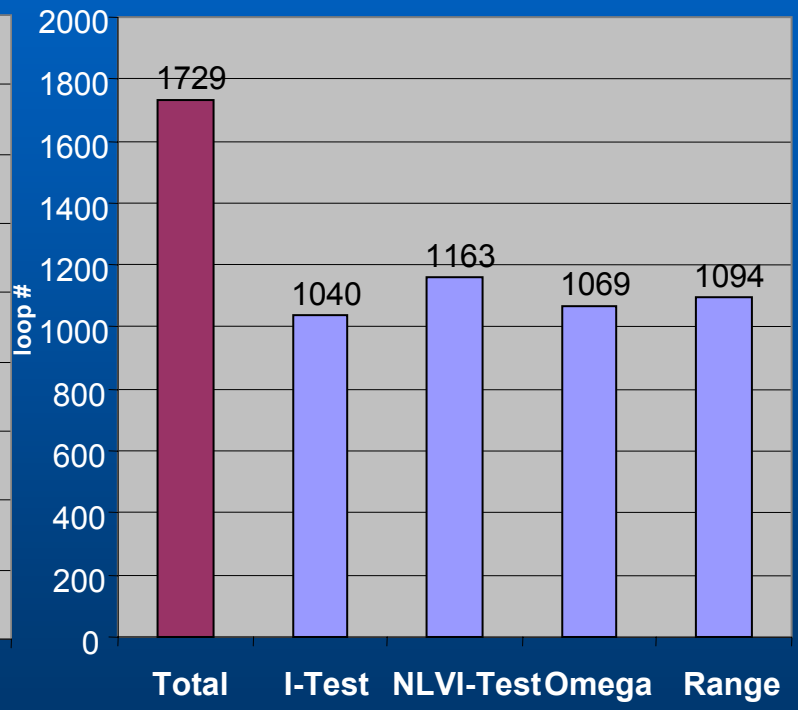


	Test	Comp TIme	DD Time	Test Time
Perfect	I-Test	40.5 (100%)	6.7 (16%)	1.0 (2%)
	NLVI-Test	57.3 (100%)	23.5 (41%)	17.8 (31%)
	Omega	122.9 (100%)	89.2 (73%)	83.4 (68%)
	Range	41.6 (100%)	7.8 (19%)	2.1 (5%)

	Test	Comp TIme	DD Time	Test Time
SPEC	I-Test	14.0 (100%)	2.8 (20%)	1.5 (11%)
	NLVI-Test	28.0 (100%)	16.7 (60%)	15.4 (55%)
	Omega	109.9 (100%)	98.6 (90%)	97.3 (89%)
	Range	14.1 (100%)	2.8 (20%)	1.5 (11%)

Loop Parallelization in the Perfect and SPEC Benchmarks





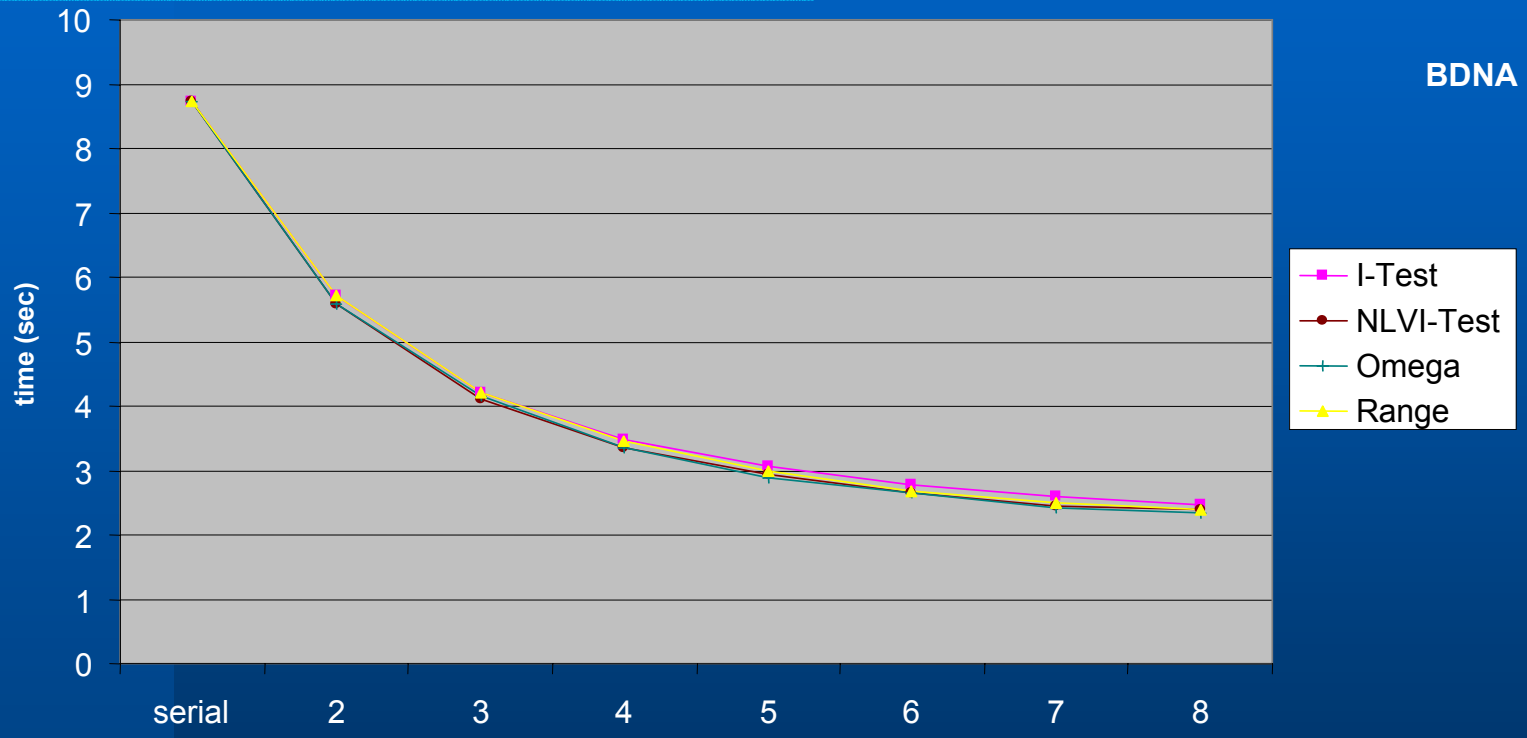
	Test	Total Loops	Parallel
Perfect	I-Test	4118 (100%)	2253 (55%)
	NLVI-Test	4118 (100%)	2369 (58%)
	Omega	4118 (100%)	2295 (56%)
	Range	4118 (100%)	2335 (57%)

	Test	Total Loops	Parallel
SPEC	I-Test	1729 (100%)	1040 (60%)
	NLVI-Test	1729 (100%)	1163 (67%)
	Omega	1729 (100%)	1069 (62%)
	Range	1729 (100%)	1094 (63%)

Execution Performance in MDG of the Perfect Benchmarks

Program	Test	serial	2-proc	sp	3-proc	sp	4-proc	sp	5-proc	sp	6-proc	sp	7-proc	sp	8-proc	sp
	I-Test	51.33	28.76	**1.8**	19.48	**2.6**	14.90	**3.4**	12.08	**4.2**	10.23	**5.0**	9.26	**5.5**	7.96	**6.4**
	NLVI-Test	51.33	28.97	**1.8**	19.63	**2.6**	15.01	**3.4**	12.16	**4.2**	10.31	**5.0**	9.35	**5.5**	8.00	**6.4**
MDG	**Omega**	51.33	28.69	**1.8**	19.43	**2.6**	14.88	**3.4**	12.09	**4.2**	10.22	**5.0**	9.27	**5.5**	7.97	**6.4**
	Range	51.33	29.03	**1.8**	19.65	**2.6**	15.00	**3.4**	12.16	**4.2**	10.31	**5.0**	9.33	**5.5**	8.00	**6.4**

Execution Performance in BDNA of the Perfect Benchmarks



Program	Test	serial	2-proc	sp	3-proc	sp	4-proc	sp	5-proc	sp	6-proc	sp	7-proc	sp	8-proc	sp
	I-Test	8.73	5.71	**1.5**	4.22	**2.1**	3.49	**2.5**	3.07	**2.8**	2.79	**3.1**	2.61	**3.3**	2.47	**3.5**
	NLVI-Test	8.73	5.58	**1.6**	4.11	**2.1**	3.35	**2.6**	2.93	**3.0**	2.64	**3.3**	2.45	**3.6**	2.39	**3.7**
BDNA	**Omega**	8.73	5.59	**1.6**	4.15	**2.1**	3.34	**2.6**	2.89	**3.0**	2.66	**3.3**	2.42	**3.6**	2.35	**3.7**
	Range	8.73	5.72	**1.5**	4.21	**2.1**	3.46	**2.5**	2.98	**2.9**	2.68	**3.3**	2.49	**3.5**	2.38	**3.7**

Execution Performance in ARC2D of the Perfect Benchmarks

Program	Test	serial	2-proc	sp	3-proc	sp	4-proc	sp	5-proc	sp	6-proc	sp	7-proc	sp	8-proc	sp
	I-Test	29.93	23.10	**1.3**	17.97	**1.7**	13.94	**2.1**	12.19	**2.5**	11.12	**2.7**	10.35	**2.9**	9.84	**3.0**
	NLVI-Test	29.93	23.15	**1.3**	17.93	**1.7**	13.96	**2.1**	12.30	**2.4**	11.08	**2.7**	10.40	**2.9**	9.86	**3.0**
ARC2D	Omega	29.93	23.18	**1.3**	17.92	**1.7**	13.96	**2.1**	12.22	**2.4**	11.13	**2.7**	10.36	**2.9**	9.91	**3.0**
	Range	29.93	23.10	**1.3**	17.88	**1.7**	13.85	**2.2**	12.02	**2.5**	10.98	**2.7**	10.36	**2.9**	10.00	**3.0**

Execution Performance in FLO52 of the Perfect Benchmarks

Program	Test	serial	2-proc	sp	3-proc	sp	4-proc	sp	5-proc	sp	6-proc	sp	7-proc	sp	8-proc	sp
	I-Test	3.77	3.61	1.0	3.40	1.1	2.86	1.3	2.85	1.3	2.80	1.3	2.81	1.3	2.78	1.4
	RVI-Test	3.77	3.62	1.0	3.40	1.1	2.85	1.3	2.86	1.3	2.77	1.4	2.77	1.4	2.74	1.4
FLO52	Omega	3.77	3.60	1.0	3.40	1.1	2.84	1.3	2.86	1.3	2.80	1.3	2.80	1.3	2.77	1.4
	Range	3.77	3.59	1.1	3.40	1.1	2.84	1.3	2.84	1.3	2.81	1.3	2.80	1.3	2.78	1.4

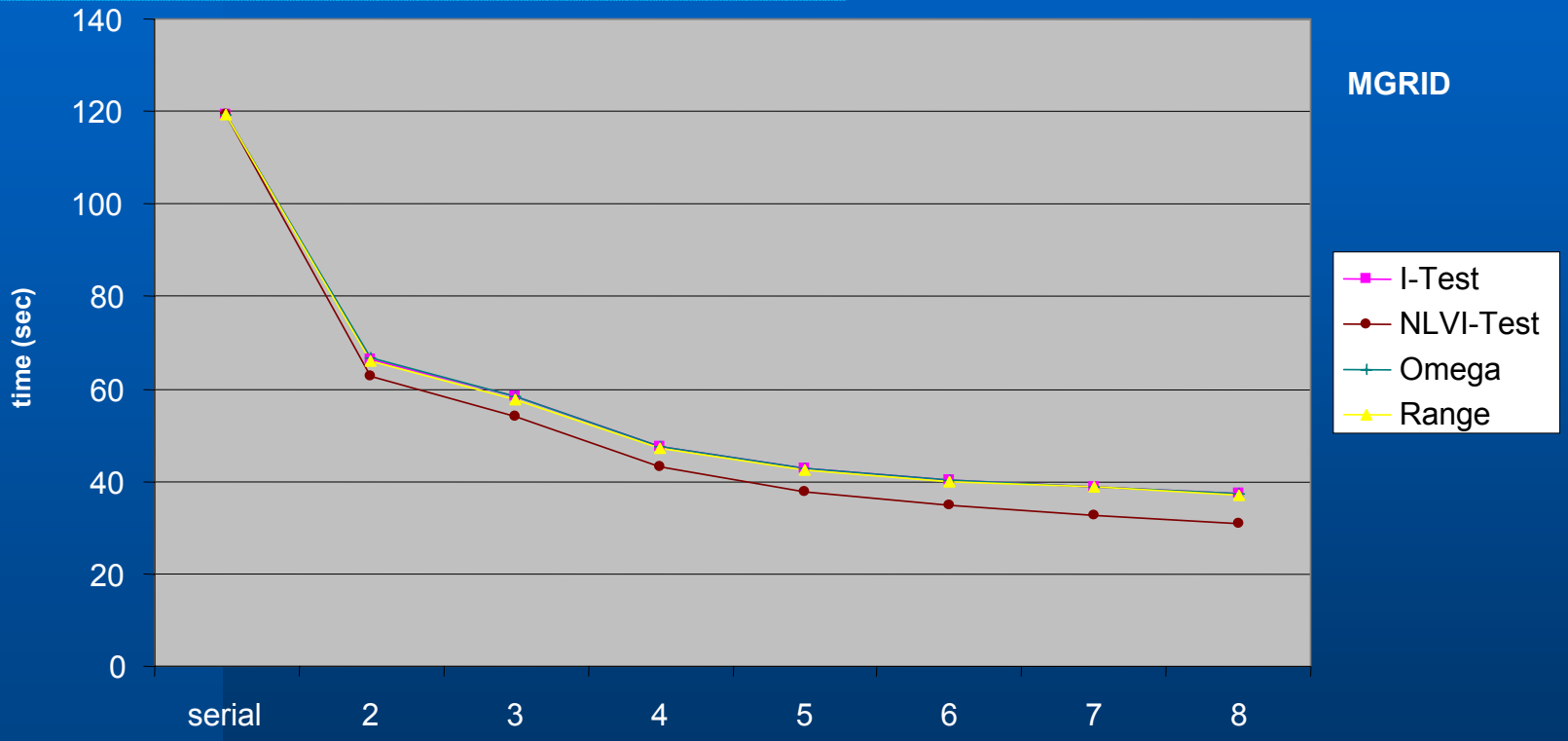
Execution Performance in TRFD of the Perfect Benchmarks

Program	Test	serial	2-proc	sp	3-proc	sp	4-proc	sp	5-proc	sp	6-proc	sp	7-proc	sp	8-proc	sp
	NLVI-Test	4.68	3.42	**1.4**	2.45	**1.9**	2.00	**2.3**	1.72	**2.7**	1.53	**3.1**	1.39	**3.4**	1.29	**3.6**
TRFD	Range	4.68	3.43	**1.4**	2.44	**1.9**	2.01	**2.3**	1.72	**2.7**	1.53	**3.1**	1.40	**3.3**	1.29	**3.6**

Execution Performance in HYDRO2D of the SPEC Benchmarks

Program	Test	serial	2-proc	sp	3-proc	sp	4-proc	sp	5-proc	sp	6-proc	sp	7-proc	sp	8-proc	sp
	I-Test	172.99	109.27	1.6	78.72	2.2	62.86	2.8	53.39	3.2	47.74	3.6	43.65	4.0	40.70	4.3
	NLVI-Test	172.99	104.11	1.7	74.11	2.3	58.30	3.0	49.01	3.5	43.21	4.0	39.31	4.4	36.29	4.8
HYDRO2D	Omega	172.99	109.67	1.6	78.58	2.2	62.70	2.8	53.26	3.2	47.86	3.6	43.69	4.0	40.70	4.3
	Range	172.99	104.34	1.7	74.15	2.3	58.54	3.0	48.93	3.5	43.37	4.0	39.28	4.4	36.18	4.8

Execution Performance in MGRID of the SPEC Benchmarks



Program	Test	serial	2-proc	sp	3-proc	sp	4-proc	sp	5-proc	sp	6-proc	sp	7-proc	sp	8-proc	sp
	I-Test	119.20	66.39	1.8	58.44	2.0	47.67	2.5	42.95	2.8	40.26	3.0	38.78	3.1	37.28	3.2
	NLVI-Test	119.20	62.84	1.9	54.07	2.2	43.05	2.8	37.83	3.2	34.95	3.4	32.63	3.7	30.82	3.9
MGRID	Omega	119.20	66.61	1.8	58.34	2.0	47.64	2.5	42.90	2.8	40.35	3.0	38.68	3.1	37.43	3.2
	Range	119.20	65.89	1.8	57.80	2.1	47.08	2.5	42.61	2.8	39.99	3.0	38.79	3.1	37.15	3.2

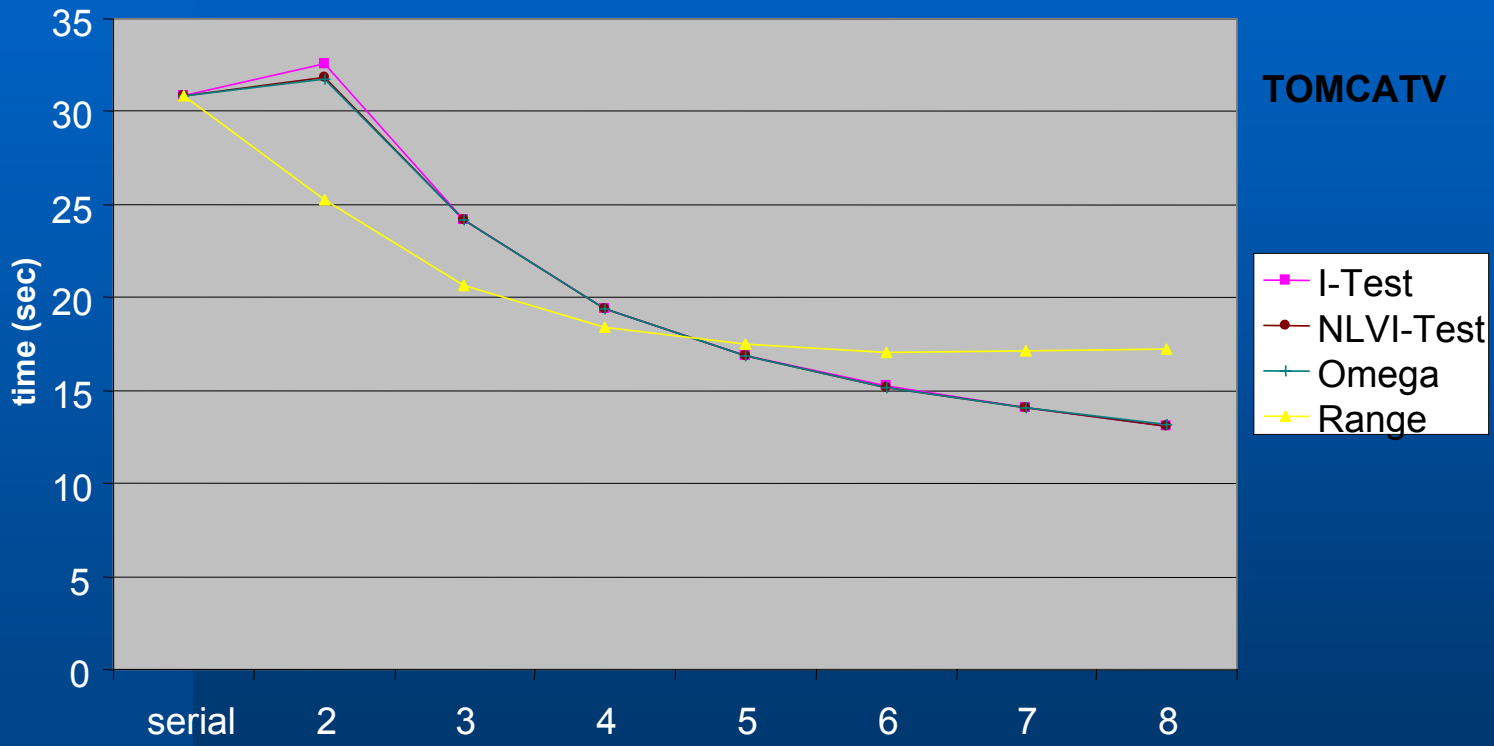
Execution Performance in SU2COR of the SPEC Benchmarks

Program	Test	serial	2-proc	sp	3-proc	sp	4-proc	sp	5-proc	sp	6-proc	sp	7-proc	sp	8-proc	sp
	I-Test	119.33	69.51	**1.7**	51.54	**2.3**	41.85	**2.9**	37.82	**3.2**	35.07	**3.4**	34.34	**3.5**	32.86	**3.6**
	NLVI-Test	119.33	69.79	**1.7**	51.35	**2.3**	41.80	**2.9**	37.87	**3.2**	35.14	**3.4**	34.19	**3.5**	33.05	**3.6**
SU2COR	**Omega**	119.33	69.72	**1.7**	51.62	**2.3**	41.83	**2.9**	37.76	**3.2**	35.07	**3.4**	34.17	**3.5**	32.94	**3.6**
	Range	119.33	70.15	**1.7**	51.60	**2.3**	42.21	**2.8**	38.14	**3.1**	35.43	**3.4**	34.27	**3.5**	33.35	**3.6**

Execution Performance in SWIM of the SPEC Benchmarks

Program	Test	serial	2-proc	sp	3-proc	sp	4-proc	sp	5-proc	sp	6-proc	sp	7-proc	sp	8-proc	sp
	I-Test	86.69	44.12	**2.0**	28.98	**3.0**	21.72	**4.0**	17.11	**5.1**	14.18	**6.1**	11.86	**7.3**	9.84	**8.8**
	NLVI-Test	86.69	44.44	**2.0**	29.13	**3.0**	21.74	**4.0**	17.16	**5.1**	14.27	**6.1**	11.88	**7.3**	9.88	**8.8**
SWIM	Omega	86.69	44.24	**2.0**	28.84	**3.0**	21.88	**4.0**	17.15	**5.1**	14.22	**6.1**	11.92	**7.3**	9.81	**8.8**
	Range	86.69	44.39	**2.0**	29.17	**3.0**	21.78	**4.0**	17.16	**5.1**	14.21	**6.1**	11.82	**7.3**	9.89	**8.8**

Execution Performance in TOMCATV of the SPEC Benchmarks



Program	Test	serial	2-proc	sp	3-proc	sp	4-proc	sp	5-proc	sp	6-proc	sp	7-proc	sp	8-proc	sp
	I-Test	30.86	32.59	**0.9**	24.22	**1.3**	19.43	**1.6**	16.89	**1.8**	15.22	**2.0**	14.06	**2.2**	13.10	**2.4**
	NLVI-Test	30.86	31.85	**1.0**	24.14	**1.3**	19.40	**1.6**	16.87	**1.8**	15.19	**2.0**	14.06	**2.2**	13.11	**2.4**
TOMCATV	Omega	30.86	31.77	**1.0**	24.19	**1.3**	19.42	**1.6**	16.87	**1.8**	15.18	**2.0**	14.04	**2.2**	13.13	**2.4**
	Range	30.86	25.29	**1.2**	20.69	**1.5**	18.36	**1.7**	17.49	**1.8**	17.06	**1.8**	17.17	**1.8**	17.25	**1.8**

Conclusions

- Traditional data dependence analysis techniques assume simple source code when in fact real source is complex
- Large amount of potential parallelism remain unexploited
- Other techniques have been proposed to address most of the issues
- Each technique has its own advantages and limitations
- We created a new data dependence test that manages to combine the advantages and overcome the limitations

Conclusions

- Defined conditions that guarantee integer solution to a data dependence problem even in presence of non-linear constraints
- It can address the issues of
  - Non-Linear expressions
  - If-Statement conditions
  - Coupled array subscripts
  - Complex loop bounds
  - Prove Definite Dependence
  - Produce complete direction vector information
  - Has polynomial time complexity

Conclusions

- **NLVI-Test**
  - **Is more accurate than all polynomial time tests**
  - **Is much faster than the Omega test**
  - **Produced the highest degree of parallelization**
  - **Increased the execution performance in several benchmarks**
- **In order to achieve better parallelization non-linear expressions need to be taken into account**
- **Compilation efficiency is essential**