Interprocedural analyses and compilers

François IRIGOIN
Corinne ANCOURT, Béatrice CREUSILLET, Nga NGUYEN
Centre de Recherche en Informatique
École des mines de Paris
Fontainebleau

Is this a parallel optimal loop?

```
   subroutine saxpy(n,da,dx,incx,dy,incy)
   real dx(1),dy(1),da

   if(n.le.0)return
   if (da .eq. 0.0e0) return
   ...
   do 30 i = 1,n
     dy(i) = dy(i) + da*dx(i)
30 continue
   end
```

Why do we need interprocedural analyses?

1. Automatic parallelization
   - Dependence test
   - Array privatization
2. Code optimization:
   - Lower and upper bounds for loop bounds
   - Use-def code elimination
   - Expression simplification: constant folding
   - Memory allocation: escape analysis

3. Correctness/Safety/Debugging:
   - Array bound checking (statically, dynamically)
   - Alias/Shape analysis (statically, dynamically)
   - Array initialization (?)
4. Maintenance/Reuse/Specialization/Reverse-engineering:
   - *Partial evaluation*, constant folding, slicing,...
   - Dead code and dead module detection
   - Array resizing
   - Summary information

5. Checking/Certification/Proofs:
   - Convex properties: mutual exclusion, producer-consumer synchronization,...
   - Other properties: liveliness, memory allocation, determinism, floating point precision,...
   - Summary information for non-automatic proof

Automatic Parallelization

```
DOALL J = 1, NKJ
 PRIVATE J,JI,JE,I,O(1:NFAC),PHW(1:NFAC)
 CALL PHWAK(J)
 CALL GRAD1(PHW, O)
 DOALL I = 1, NKJ
  PRIVATE I,JI,JE
  JE = JEX(I)
  JI = JIX(I)
  IF (1.LE.JE.AND.JE.LE.NFAC.AND.1.LE.JI.AND.JI.LE.NFAC) THEN
   CC(J,I) = O(JE)-O(JI)
  ENDIF
 ENDDO
ENDDO
```

1997 Report by Béatrice Creusillet

Code specialization: CURMOD

```
C  P(I,II,INC,ISIG,J,K,NFREE,NTERMS)
C    {IC==10, ILIM==0, INCMAX==10,
C    IPRIN==0, MODE==0, NFREE+NLS==NPTS,
C    NTERMS==NLS, NSEARC==0}

31    DO 34 J = 1, NSEARC

C  P() {0==-1}
        BETA(J) = 0.0

C  P() {0==-1}
        DO 34 K = 1, J

C  P() {0==-1}
34          ALPHA(J,K) = 0.0
```

Array bound checking: excerpt from TOMCATV

```
      PROGRAM TOMCATV
      PARAMETER (NMAX = 513, ITMAX = 1000)
      REAL*8 RX(NMAX,NMAX), RXM(ITMAX)

      READ (5, *) N, ITACT
      IF (ITACT.GT.ITMAX.OR.N.GT.NMAX) THEN
         PRINT *, 'Please recompile: This version is lacking Storage'
         STOP
      ENDIF
C  P(ITACT,N) {ITACT<=1000, N<=513}
      DO 140 ITER = 1, ITACT
         DO 80 J = 2, N-1
            DO 80 I = 2, N-1
C  P(I,ITACT,ITER,J,N) {2<=I, I+1<=N, ITACT<=1000, ITER<=ITACT,
C   1<=ITER, 2<=J, J+1<=N, N<=513}
               RXM(ITER) = MAX(RXM(ITER), ABS(RX(I,J)))
80              CONTINUE
140        CONTINUE
      END
```

Property proof

```
C  T(A,B,C,D,E,F,G) {A+C#init+E+G==A#init+C+E#init+G#init,
C    B+C+E#init+G#init==B#init+C#init+E+G,
C    C+D+E==C#init+D#init+E#init, F+G==F#init+G#init}
     DO WHILE (.TRUE.)
C     Transition sc
       IF (A.GE.1.AND.D.GE.1) THEN
C           T(A,B,D,E) {A==A#init-1, B==B#init+1, D==D#init-1, E==E#init+1}
          A = A-1, D = D-1, B = B+1, E = E+1
C     Transition rc
       ELSEIF (E.GE.1.AND.F.GE.1) THEN
C           T(D,E,F,G) {D==D#init+1, E==E#init-1, F==F#init-1, G==G#init+1}
          E = E-1, F = F-1, D = D+1, G = G+1
C     Transition sd
       ELSEIF (B.GE.1.AND.D.GE.1) THEN
C           T(A,B,C,D) {A==A#init+1, B==B#init-1, C==C#init+1, D==D#init-1}
          B = B-1, D = D-1, A = A+1, C = C+1
C     Transition rd
       ELSEIF (C.GE.1.AND.G.GE.1) THEN
C           T(C,D,F,G) {C==C#init-1, D==D#init+1, F==F#init+1, G==G#init-1}
          C = C-1, G = G-1, D = D+1, F = F+1
       ELSE
          PRINT *, A, B, C, D, E, F
       ENDIF
```

Property proof (cont.)

```
C  P(A,B,C,D,E,F,G) {A==1, B==0, C==0, D==1, E==0, F==1, G==0}
      DO WHILE (.TRUE.)
C       Transition sc
         IF (A.GE.1.AND.D.GE.1) THEN
            A = A-1, D = D-1, B = B+1, E = E+1
C       Transition rc
         ELSEIF (E.GE.1.AND.F.GE.1) THEN
            E = E-1, F = F-1, D = D+1, G = G+1
C       Transition sd
         ELSEIF (B.GE.1.AND.D.GE.1) THEN
            B = B-1, D = D-1, A = A+1, C = C+1
C       Transition rd
         ELSEIF (C.GE.1.AND.G.GE.1) THEN
            C = C-1, G = G-1, D = D+1, F = F+1
         ELSE
C  P(A,B,C,D,E,F,G) {A+B==1, B+2C+D==G+1, C+D+E==1, F+G==1}
            PRINT *, A, B, C, D, E, F
         ENDIF
      ENDDO
```

A precondition in module D2ESP (FPPP)

```
C  P(I,IACC,IAJ,IAJ1,IAK,IAK1,IBETA,ICHECK,ICRIT,IDM,IDN,IDUMP,IEND,
C    IF1,IFHAM,IFIRST,IFLAGS,IFRC,IFROZE,IFXYZ,IHDIAG,IMO,INCONF,
C    INDDO,INEQ,INITF,IPRINT,IREST,ISCF,ISCFP,ISCVEE,ISCVEN,ISCVNN,
C    ISHLBF,ISYMGR,IVEE,JFXYZ,LENB,LENHAM,LENMO,LENNDO,LRWSY1,NAT3,
C    NDEN,NHAM,NONC1,NOP1,NORB,NOSYM,NTT,USESYM) {IAJ==IFHAM,
C    IAJ==INCONF, IAJ1==IEND+2, IAJ1==IF1+1, IFIRST+ISCVNN==2IBETA-1,
C    IBETA+NBASIS==IFIRST, IDM==6NATOMS+2, IDN==2NAT3+2,
C    IDN==6NATOMS+2, ISCVNN+LENNDO==IF1+3, IFLAGS==IHDIAG-4,
C    IFLAGS==ISCVEN-1, IFRC==0, IFXYZ==2, IHDIAG==IMO+4,
C    IHDIAG==ISCVEE+2, INCONF+LENHAM==ISCVNN-3, 3NSHELL+NTT==INEQ-2,
C    IOUT==6, IREST==0, ISCF==0, ISCFP==1, ISCVEN==ISCVNN-2, IVEE==1,
C    JFXYZ==NAT3+2, 2JFXYZ==3NSHELL+4, LENB==7401, LENMO==0,
C    LRWSY1==13, MAXSHL==400, NDEN==1, NHAM==0, NONC1==1, NOP1==1,
C    NORB==0, NOSYM==0, 1<=I, NTT+1<=I, IACC<=0,
C    3INCONF+2ISHLBF+3LENHAM+11<=5IBETA,
C    5IBETA<=3INCONF+2ISHLBF+3LENHAM+12,
C    3INCONF+3LENHAM+2LENNDO+3<=IBETA+2ISHLBF,
C    IBETA+2ISHLBF<=3INCONF+3LENHAM+2LENNDO+4, 0<=ICRIT, ICRIT<=9,
C    4INCONF<=4IFROZE+NSHELL+2, 4IFROZE+NSHELL<=4INCONF}

      IF (.NOT.USESYM) THEN
```

Issues and approaches

1. Issues:
   (a) Speed/Time complexity
   (b) Size/Space complexity
   (c) Accuracy/Utility vs Speed/Size (and Robustness)
   (d) Implementation cost/complexity
2. Approaches:
   (a) Inlining (Whole Program Analysis?)
   (b) Summarization (Modularity)
   (c) Cloning

Modularity: transformers and preconditions

- abstract command, $Cmd = Store \rightarrow Store$, not abstract store σ (avec $Store = Loc \rightarrow Value$)
- semantics: function from store to store
- abstraction: polyedric relation between two sets of stores
- data structure: sparse representation and minimisation of explicit equations:

```
C  T(I,J) {I+J==I#init+J#init}
      DO WHILE (I.LT.N)
```

```
C  P(IXX,J,NN) {J==1, 2N==NN}
      DO 11 I = 1, NN, 2
```

- impact on accuracy: the best convex relation depends on the set of input stores

Interprocedural: transformer for caller D2ESP

Each module is analyzed only once:

```
C  T(LENB) {37MAXSHL+2<=2LENB, 2LENB<=37MAXSHL+3}

C
C     INITIALIZE THIS LINK.
C
C***        CALL DRUM
C***        CALL TRAKIO(6HD2ESP ,1)
      LENB = 15*MAXSHL+INTOWP(2+7*MAXSHL)
```

Transformer for callee INTWOP

```
C  T(INTOWP) {2INTOWP<=I+1, I<=2INTOWP}

      INTEGER FUNCTION INTOWP(I)
      INTOWP = (I+1)/2
      END
```

Inaccurate Precondition (example TME)

Best postcondition: $0 \leq \sigma(\rho(A)) < \sigma(\rho(T))$

```
      PROGRAM NON_CONVEX
      INTEGER I, A ,T
      A = 0
      T = 0

C  P(A,T) {A==0, T==0}
      DO I = 1, N

C  P(A,I,T) {1<=I, I<=N, T+1<=I, 0<=T}
         READ *, X
         IF (X.GT.X0) THEN
            A = T
C  P(A,I,T) {A==T, I<=N, T+1<=I, 0<=T}
            T = T+1
         ENDIF
      ENDDO

C  P(A,I,T) {N+1<=I, T+1<=I, 0<=T}
      PRINT *, A, T
      END
```

Approximate Convex Transformer

$$(A = A' \wedge T = T') \vee (A = T' \wedge T = T' + 1)$$

```
      PROGRAM NON_CONVEX
      INTEGER I, A ,T
      A = 0
      T = 0

C  T(A,I,T) {T+1<=I+T#init, T#init<=T}
      DO I = 1, N
         READ *, X

C  T(A,T) {T#init<=T, T<=T#init+1}
         IF (X.GT.X0) THEN
            A = T
            T = T+1
         ENDIF
      ENDDO

      PRINT *, A, T
      END
```

Semantics equations

Basic constructs

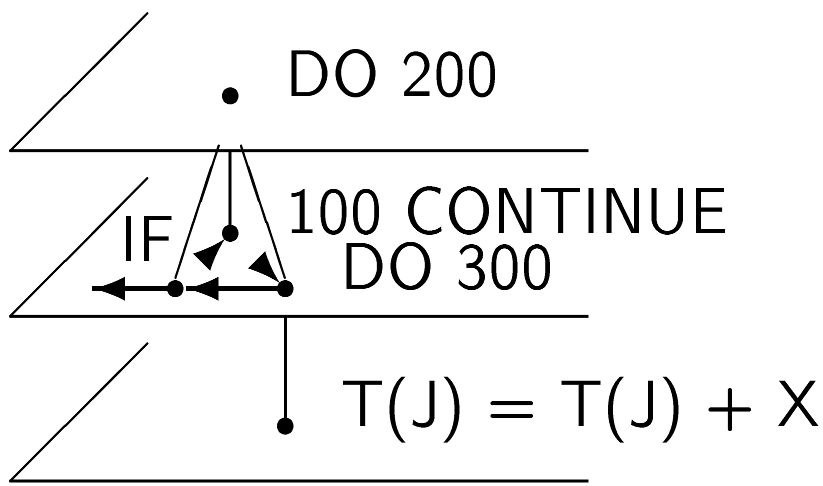
- call:
  - assignment
  - intrinsic
  - user call
- sequence: opérateur *rond*
- test: enveloppe convexe
- loop, whileloop: point fixe
- unstructured (CFG): point fixe

Hierarchical Control Flow Graph and AST

- benefit from structured code:
  - simpler semantics equations
- minimize CFG size when CFG are necessary
  - complexity depends on vertex and arc numbers
- recursive representation of structured and non-structured code

HCFG: An Example

```
      DO 200 I = 1, N
100      CONTINUE
         DO 300 J = 1, M
            T(J) = T(J) + X
300      CONTINUE
         IF(X.GT.T(I)) GO TO 100
200   CONTINUE
```



Abstract Domain: Linear Algebra

- satisfiability: Fourier-Motzkin and Simplex (with integer extensions)
- redundancy elimination: dual or satisfiability
- convex hull: Chernikova (Halbwachs)
- heuristics

Problems: magnitude (64 bits), space and time explosions

Impact of repetitive overflows

```
C  P(A1,A2,I,N,PC1,PC2,S,T) {I<=N, 1<=2I+PC1, PC1+1<=2I,
C    7PC1+6PC2+21S+42<=29I+14T, 9PC1+6PC2+18S+27<=21I+11T,
C    4PC1+9PC2+8S+15<=14I+8T, 2PC1+3PC2+6S+9<=7I+4T,
C    PC1+2PC2+4S+6<=5I+2T, 2PC1+PC2+4S+6<=5I+2T, PC1+PC2+2S+2<=2I+T,
C    PC1+2S+3<=4I, PC1+3S+6<=5I+2T, PC1+2S+3<=3I+T, T<=I+PC1+2S+1,
C    T+1<=3I+PC1, PC1+1<=I+T, 1<=2I+PC2, PC2+1<=2I, PC2+2S+3<=4I,
C    PC2+3S+6<=5I+2T, PC2+2S+3<=3I+T, T<=I+PC2+2S+1, T+1<=3I+PC2,
C    PC2+1<=I+T, S+2<=2I, T<=2I, T<=PC1+PC2+2S+2, 0<=PC1+2S+1,
C    0<=PC2+2S+1, 0<=S, 0<=T}

      IF (PC1.EQ.1) THEN
c        end of critical section for process 1
         PC1 = -1
         S = S+1
      ENDIF

C  P(A1,A2,I,N,PC1,PC2,S,T) {I<=N, 1<=2I+PC2, PC2+1<=2I, T+1<=3I+PC2,
C    PC2+1<=I+T, T<=2I, 0<=T}
```

Fixpoint operators for loops

They are computed for transformers and may be recomputed for preconditions:

- kernel of transition matrix: invariant equations
- pattern-matching: FORMA
- derivative: look for constraints on differences
- Halbwachs/Cousot: two heuristics (exponential wrt loop depth)
- Talpin & al.: not implemented

Problem: involutive functions, $f^k = I$

Fixpoint example: precondition

```
C  P(IJ,IX,IY) {1<=IX, NX2+1<=IX}
      DO IX = 1, NX2

C  P(IJ,IX,IY) {1<=IX, IX<=NX2}
         IJ = IX

C  P(IJ,IX,IY) {IJ==IX, 1<=IX, IX<=NX2}
         DO IY = 1, NY2

C  P(IJ,IX,IY) {IX+IY<=IJ+1, 1<=IX, IX<=NX2, 1<=IY, IY<=NY2}
            A(IJ) = 0.

C  P(IJ,IX,IY) {IX+IY<=IJ+1, 1<=IX, IX<=NX2, 1<=IY, IY<=NY2}
            IJ = IJ+NX2
         ENDDO
      ENDDO
```

Fixpoint example: transformer

```
C  T(IJ,IX,IY) {1<=IX}
     DO IX = 1, NX2
        IJ = IX
C  T(IJ,IY) {1<=IY}
        DO IY = 1, NY2
           A(IJ) = 0.
C  T(IJ) {IJ#init+NX2==IJ}
           IJ = IJ+NX2
        ENDDO
     ENDDO
```

The transformer fixpoint must be recomputed wrt to precondition to be accurate: IJ is increasing

Fixpoints for CFGs

- recursive SCC decomposition
- delete closing arcs
- if DAG, compute transfer functions along each path
- move up and compute fixpoint of the union of the transfer functions, update the transfer function of the SCC entry node
- one fixpoint per circuit
- same algorithm for transformers and preconditions

See Bourdoncle, 1993.

Fixpoint for CFG: Example

```
      PROGRAM UNSTRUC11

      K = 0
      I = 0
      J = 0
      L = 0

100   CONTINUE
      IF(X.GT.0.) GO TO 300
      L = L + 4
200   CONTINUE
      K = K + 3
      I = I + 1
      IF(Y.GT.0.) GO TO 100
      J = J + 2
      GO TO 200

300   CONTINUE
      PRINT *, I, J, K, L
      END
```

Fixpoint for CFG: Example (cont.)

```
C  T(I,J,K,L) {4I+2J#init+L#init==4I#init+2J+L, 3I+K#init==3I#init+K,
C    0.<=X, 2I#init+J<=2I+J#init, J#init<=J}
C     BEGIN UNSTRUCTURED
99998 CONTINUE
      IF (X.GT.0.) GOTO 300
C  T(L) {L==L#init+4}
      L = L+4

C  T(I,K) {I==I#init+1, K==K#init+3}
200   CONTINUE
      K = K+3
      I = I+1

      IF (Y.GT.0.) GOTO 99998
C  T(J) {J==J#init+2}
      J = J+2
      GOTO 200
300   CONTINUE
C     END UNSTRUCTURED
```

Beyond Fortran

- environment, $\rho = Id \rightarrow Loc$:
  - aliasing
  - data structures
  - pointers
- exceptions vs limited control effects:
  - STOP only
- recursivity
- functional formal parameters and pointers to functions

Recent and On-Going Extensions for Convex Store Abstraction

- transformer for expressions (expression linearization)
- boolean: encoded as 0 and 1
- floating point: avoid integer constants!
  x=i, y=j, x+y!=i+j
  Fourier-Motzkin compatible, excessive numerical precision
- character strings: no more than two variables per constraint
- illegal program: instrumentation with tests
- improved fixpoint for CFGs

Beyond Integer Variables

Transformer:

```
C  T(I,L1,S1,X) {S1==S2, U+V==X, 1<=I, I<=2, 0<=L1, 0<=L1, L1<=1,
C    L1<=L2, L2+L3<=L1+1, L1<=L3}
     CALL COPY_FOUR(I, 1+MOD(J, 2), X, U+V, S1, S2, L1, L2.AND.L3)
     PRINT *, I, X, S1, L1
```

Postconditions:

```
     CALL COPY_FOUR(I, 1+MOD(J, 2), X, U+V, S1, S2, L1, L2.AND.L3)
C  P(I,L1,L2,S1,S2,U,V,X) {"Hi!"==S2, 2.==U, 3.==V, L2==1, S1==S2,
C    U+V==X, 1<=I, I<=2, 0<=L1, L1<=1, L3<=L1, L1<=L3}

     PRINT *, I, X, S1, L1

     READ *, J, U, V, S2, L2
C  P(I,J,L1,L2,S1,S2,U,V,X) {"Hi!"==S1, 2.+3.==X, 1<=I, I<=2, 0<=L1,
C    L1<=1, L3<=L1, L1<=L3}
```

Boolean Variables in TWLDRV (FPPP)

```
C  P(A1,A12,A1234,A1234I,A12I,A2,A3,A34,A34I,A4,ABVE00,ABX,ABY,ABZ,AX,
C    AX1,AY,AZ,BX,BY,BZ,C1110,C1120,C1130,C1210,C1220,C1230,C1310,
C    C1320,C1330,C2110,C2120,C2130,C2210,C2220,C2230,C2310,C2320,
C    C2330,C3110,C3120,C3130,C3210,C3220,C3230,C3310,C3320,C3330,
C    CDVE00,CDX,CDY,CDZ,CPA,CPB,CPC,CPD,CPP,CPPP,CPPPP,CPPPS,CPPS,
C    CPPSP,CPPSS,CPS,CPSP,CPSPP,CPSPS,CPSS,CPSSP,CPSSS,CSA,CSB,CSC,
C    ...
C    V1200,V1210,V1220,V1230,V1300,V1310,V1320,V1330,V2000,V2010,
C    V2020,V2030,V2100,V2110,V2120,V2130,V2200,V2210,V2220,V2230,
C    V2300,V2310,V2320,V2330,V3000,V3010,V3020,V3030,V3100,V3110,
C    V3120,V3130,V3200,V3210,V3220,V3230,V3300,V3310,V3320,V3330,VE00,
C    VE00S,VE11,VE11S,VE12,VE12S,VE13,VE14,VE21,VE21S,VE22,VE22S,VE23,
C    VE24,VE31,VE31S,VE32,VE32S,VE33,VE34,VEE,VTEST,VTOL1,VTOL2,VTOLS)
C    {FMTGEN:FOUR==2FMTGEN:TWO, 2FMTGEN:ONE==FMTGEN:TWO,
C    FMTGEN:TEN==5FMTGEN:TWO, ICRIT==ICRITP-1, IOUT==6, ISCF==0,
C    2NATOMS==NSHELL, NONC1==1, NSYMOP==1, FI<=1, FIJ<=FI, FIJK<=FIJ,
C    0<=FIJK, 1<=ICRITP, ICRITP<=10, 0<=IFMT, JSHELL<=ISHELL,
C    KSHELL<=ISHELL, ISHELL<=2NATOMS, 1<=JSHELL, LSHELL<=KSHELL,
C    1<=LSHELL, 0<=USESYM, USESYM<=1}
```

Floating Point and Integer Values

Floating point constant are handled as variables:

```
      PROGRAM INTFLOAT
      A = 0.
      J = 0

C  T(A,I,J) {2I+J#init==J+2, 1<=I}
      DO I = 1, N
C  T(A,J) {2.+A#init==A, J==J#init+2}
         J = J+2
         A = A+2.
      ENDDO
      PRINT *, A, J
      END
```

```
      PROGRAM INTFLOAT
      A = 0.
      J = 0
      DO I = 1, N
         J = J+2
         A = A+2.
      ENDDO

C  P(A,I,J) {2I==J+2, 1<=I, N+1<=I}
      PRINT *, A, J
      END
```

No fixpoint for mathematical reasons

Future improvements

- evaluation or re-evaluation of transformers wrt preconditions
- linear appoximation of non-linear expressions
- backward analysis
- improved/specific detection of bounded variables
- array analysis: $b > n \;\equiv\; \forall i \; b[i] > n$
- convexification of non-convex information

Expression Linearization

```
    DO 210 I=2,NX1
      Q1=0.D0
      DO 200 J=2,NY1
        XX=(I-1.5D0)*HX
        YY=(J-1.5D0)*HY
        L=I+(J-1)*NX2
        DO 200 KSP=1,NSP
          TM=DENS(XX,YY)*DSIGN(1.0D0,QSPEC(KSP))
          IF (NPX(KSP)*NPY(KSP).EQ.0) TM=0.D0
          Q(L) = Q(L) - TM
          Q1=Q1+TM
200   CONTINUE
      QAV=QAV-Q1
210 CONTINUE
```

Non-Convex Union

```
      PARAMETER (MAXVALUE=10)
      READ *, I, J, K
      IF (I.LT.-MAXVALUE.OR.I.GT.MAXVALUE.OR.J.LT.-MAXVALUE.OR.J.GT.
     &MAXVALUE.OR.M.LT.-MAXVALUE.OR.M.GT.MAXVALUE) STOP
C  P(I,J,K) {0<=I+10, I<=10, 0<=J+10, J<=10, 0<=M+10, M<=10}
      IF (X.GT.0.) THEN
         I = M, K = 1
      ELSE
         J = M, K = 0
      ENDIF
C  P(I,J,K) {0<=I+10, I<=10, M<=I+J+10, I+J<=M+10, 20K+M<=I+20,
C   I+20K<=M+20, 0<=J+10, J<=10, M<=J+20K, J<=20K+M, 0<=M+10, M<=10}
      IF (K.EQ.1) THEN
C      Can we guess the relation between i and n?
         IF (I.EQ.M) THEN
C  P(I,J,K) {I==M, K==1, 0<=J+10, J<=10, M<=J+20, J<=M+20, 0<=M+10,
C   M<=10}
            PRINT *, I, J, K
         ELSE
C  P(I,J,K) {0==-1}
            PRINT *, I, J, K
         ENDIF
      ENDIF
```

Related Work for Store Abstractions

- constant propagation/constant folding, PRE,...
- intervals
- equalities and inequalities: Cousot/Halbwachs
- equalities: Karr
- Rice:
  - Callahan, Cooper, Torczon, Kennedy
  - Havlak: GSA
- IBM Research:
  - Wegman, Zadeck
  - SSA

- Bull: guarded abstractions (Dehbonei)
- UIUC CSRD: polynomials (Haghighat)
- Polyspace Technologie: intervals?!? (Alain Deutsch)

Linear constraint interprocedural analysis: Implementation

- sparse linear library (44.5 K, including dense part):
  - vector (3 K), constraint (2K)
  - constraint system (12 K)
  - generating system (0.6 K)
  - polyedral (6 K)
- dense linear modules: Simplex, Chernikova (Polylib)
- transformer library + environnement: 4.5 K

- semantics library: C semantics equation implementation (12 KLoc)
  - dbm_interface.c: 790
  - ri_to_transformers.c: 1511
  - ri_to_preconditions.c: 693
  - loop.c: 906
  - expression.c: 2017
  - interprocedural.c: 1322
  - unstructured.c: 2519 (CFG)

SPEC95 CFP Intraprocedural Array Bound Check

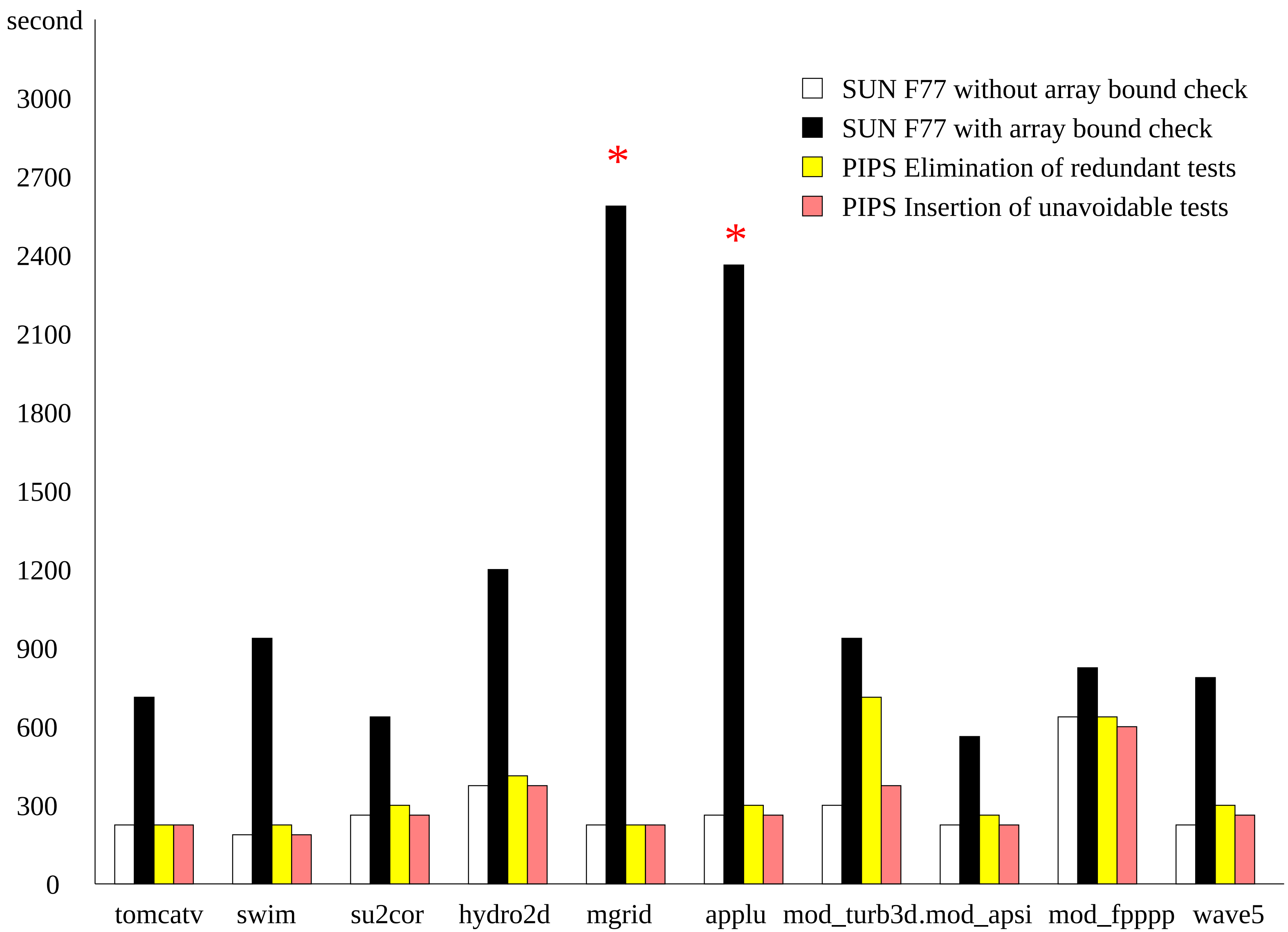
	Elimination of redundant tests				Insertion of unavoidable tests			
	Intra		Inter		Intra		Inter	
Prog.	**Comp.**	**Run**	**Comp.**	**Run**	**Comp.**	**Run**	**Comp.**	**Run**
tomcatv	99.67%	99.99%	100.00%	100.00%	99.01%	99.99%	100.00%	100.00%
swim	97.00%	97.00%	98.45%	99.99%	84.60%	99.99%	84.72%	99.99%
su2cor	94.52%	95.20%	96.59%	97.54%	92.12%	96.60%	94.48%	97.66%
hydro2d	95.13%	93.50%	96.13%	94.14%	90.02%	97.70%	93.85%	99.46%
mgrid	91.32%	99.60%	94.92%	99.60%	96.61%	99.50%	97.93%	99.66%
applu	98.54%	96.75%	99.62%	97.09%	96.38%	99.80%	96.41%	99.87%
turb3d	92.23%	51.03%	97.62%	59.85%	87.03%	76.78%	98.97%	85.57%
apsi	97.08%	99.20%	98.02%	99.90%	98,70%	99.31%	99.79%	99.99%
fpppp	94.12%	96.48%	94.61%	97.02%	92.18%	97.23%	95.82%	97.40%
wave5	94.52%	86.26%	94.66%	86.86%	91.12%	89.83%	94.01%	91.29%

SPEC95 CFP Intraprocedural Array Bound Check

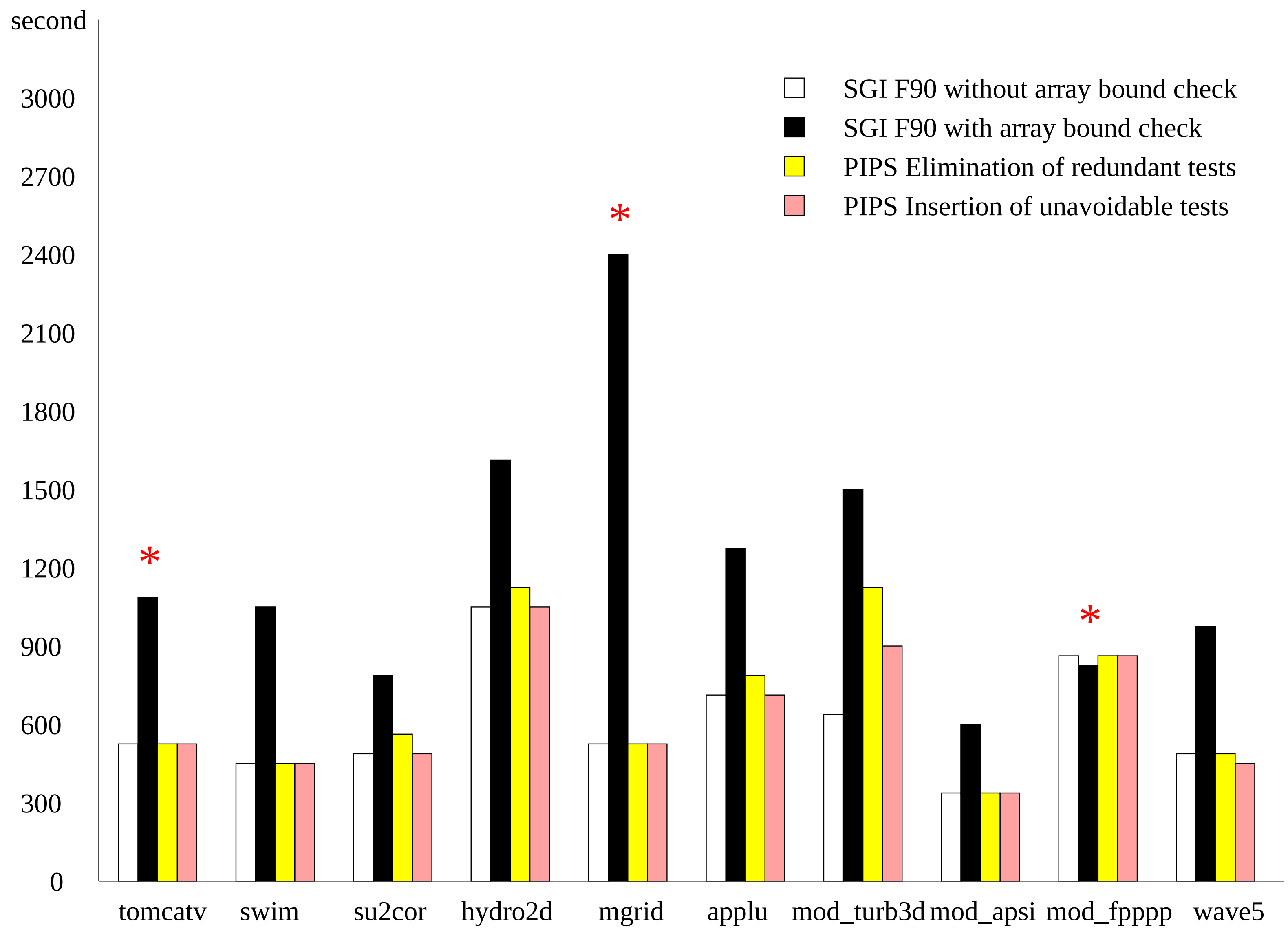
Compilation times: Ultra SPARC 360MHz, Optimized code
(SC5.0 f77 -fast -xarch=v8plusa -fsimple=2 -xprefetch)

	Elimination				**Insertion**				**SUN F77**	
Prog.	**Spd**	**PIPS**	**SUN**	**Tot.**	**Spd**	**PIPS**	**SUN**	**Tot.**	**Wo C**	**+ C**
tomcatv	55.5	0:02	0:02	0:04	22.2	0:05	0:02	0:07	0:03	0:13
swim	70.7	0:04	0:02	0:06	23.5	0:12	0:05	0:17	0:04	0:20
su2cor	39.4	0:40	0:25	1:05	10.5	2:28	0:33	3:01	0:33	2:23
hydro2d	107.0	0:16	0:19	0:35	39.8	0:43	0:32	1:15	0:32	1:03
mgrid	85.2	0:04	0:07	0:11	8.9	0:38	0:06	0:44	0:08	0:37
applu	74.9	0:33	0:25	0:58	20.2	2:02	0:54	2:56	0:57	11:12
turb3d	86.0	0:15	0:16	0:31	19.8	1:05	0:17	1:22	0:17	0:44
apsi	55.8	1:16	1:02	2:18	9.1	7:42	1:03	8:45	1:16	3:15
fpppp	50.6	0:42	0:45	1:27	8.8	4:01	0:56	4:57	0:54	1:22
wave5	35.2	5:03	1:25	6:28	8.9	12:04	1:50	13:54	2:04	6:20

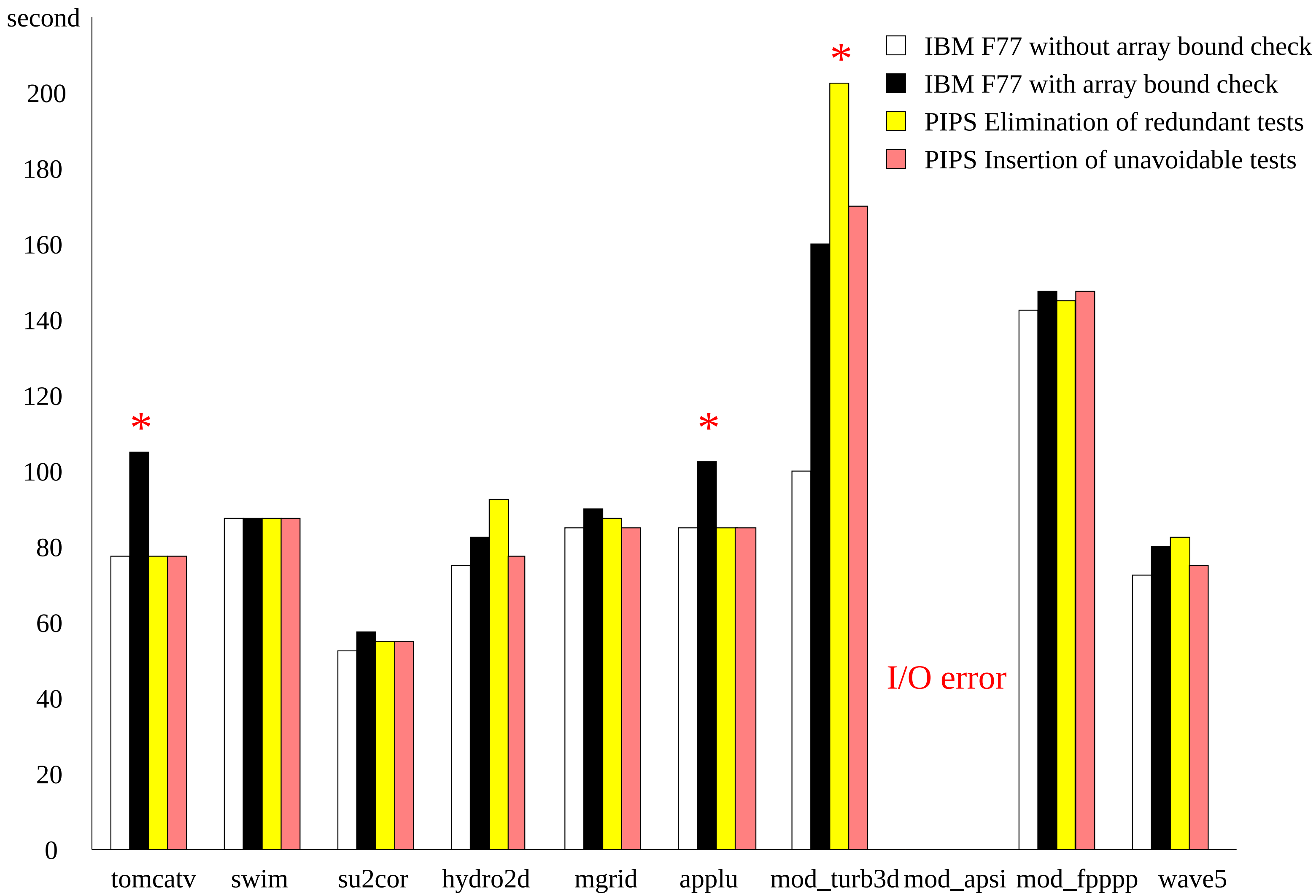
Execution time PIPS and SUN Workshop F77 5.0, Ultra SPARC 360MHz, Solaris 7



Execution time PIPS and SGI MIPSpro F90 7.3, O2 R5000 195MHz, IRIX 6.37



Execution time PIPS and IBM XL F77 7.1, RS/6000 44P-270 375MHz 4 CPU, AIX 4.3



Array Bound Checking : Conclusion

- Impact of code quality
  - array resizing (*125.turb3d, 141.apsi, 145.fpppp*)
  - control restructuring (*101.tomcatv: 98.2%* $\rightarrow$ *100.00%*)
  - procedure cloning (*104.hydro2d: 94.14%* $\rightarrow$ *99.50%*)
  - parameter checking (*110.applu: 97.09%* $\rightarrow$ *100.00%*)
- Improvements:
  - Code hoisting, non-linear expression analysis
  - Code replication, guarded regions, list of regions

Interprocedural bound violation in *wave5*, SPEC95

```
SUBROUTINE FIELD
PARAMETER (NC1 = 78885)
COMMON/EFIELD/EX(NC1),EY(NC1),EZ(NC1), BX(NC1),BY(NC1),BZ(NC1)
CALL SOLV2Y(NX2,NY2,HX,HY,0.0D0,V,BC,BDY1,BX)

SUBROUTINE SOLV2Y(NX2,NY2,HX,HY,DD,Q,BX,BY,TMP)
DIMENSION BX(4),Q(NX2,*),TMP(NX2,*),BY(4)
CALL SLV2XY(NX2-2,NY2-2,NX2,HX,HY,DUMMY,DD,Q,BX,BY,TMP,0)

SUBROUTINE SLV2XY(NX,NY,NXD,HX,HY,GX,DD,Q,BX,BY,TMP,IGXSW)
DIMENSION Q(NXD,*),TMP(NXD,NY,2),BX(4),GX(NXD),BY(4)
CALL VSLV1P(NX,NY,NXD,HX,GX,DIAG,Q(1,2),TMP,TMP(1,1,2),IGXSW,ISING)

NX = 1250, NY = 60, NXD = NX + 2, 78885 < 1252*60*2
```

SPEC95 CFP: Aliasing detection

Number of lines, routines and calls, inserted flags, tests, dynamic checks, alias violations

Program	**Lines**	**Routines**	**Calls**	**Flags**	**Tests**	**Dynamic checks**	**Violations**
tomcatv	0	0	0	0	0	0	No
swim	429	6	5	0	0	0	No
su2cor	2332	35	150	0	0	0	No
hydro2d	4292	42	85	8	12	0	Yes
mgrid	484	12	23	1	3	43	Yes
applu	3868	16	27	18	18	0	Yes
turb3d	2101	23	111	78	211	0**	Yes
apsi	7361	96	187	0	0	854**	Maybe
fpppp	2784	38	18	0	0	100**	Maybe
wave5	7764	105	168	19	114	483**	Maybe

Conclusion about PIPS convex store and location abstractions

- After twelve years... robustness and speed
- Modularity: one fixpoint per circuit, one analysis per module
- Different kinds of applications
- Program size: 1 to 160 KLOC
- Hundreds of constraints and variables
- Speed and accuracy: pragmatic improvements as required by applications

Conclusion: Implementation

- Exact semantic equations are easy to write *except for CFGs* $\Rightarrow$ *HCFG*
- Approximate equations are much harder
- Genericity wrt to the abstract domain is hard
- Combining several analysis is hard

  even without genericity
- Parallelization of parallelizers and analyzers

Conclusion: Interprocedural Analyses

- Interprocedural analyses are here to stay and to grow
- Large spectrum of applications

  Requiring diverse analyses due to necessary approximations
- Deal with incorrect programs
- Automatic parallelization and optimization could focus more on correctness:

  Users are prompt to blame the tools
- Evaluation with respect to a measurable goal