Constant Propagation w/ SSA- and Predicated SSA Form

Jens Knoop

Vienna University of Technology, Austria

This is joint work with Oliver Rüthing

TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

Outline of the Talk

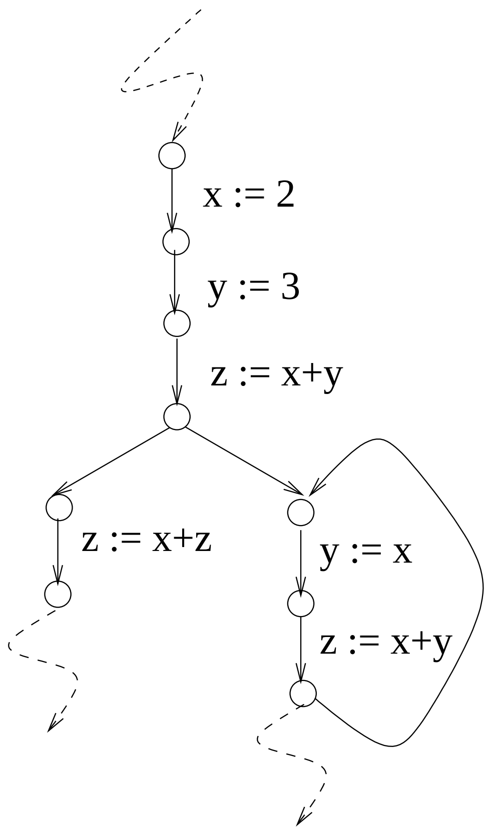
- Part I: Constant Propagation
- Part II: Constant Propagation w/ SSA Form
- Part III: Constant Propagation w/ Predicated SSA Form

Part I: Constant Propagation

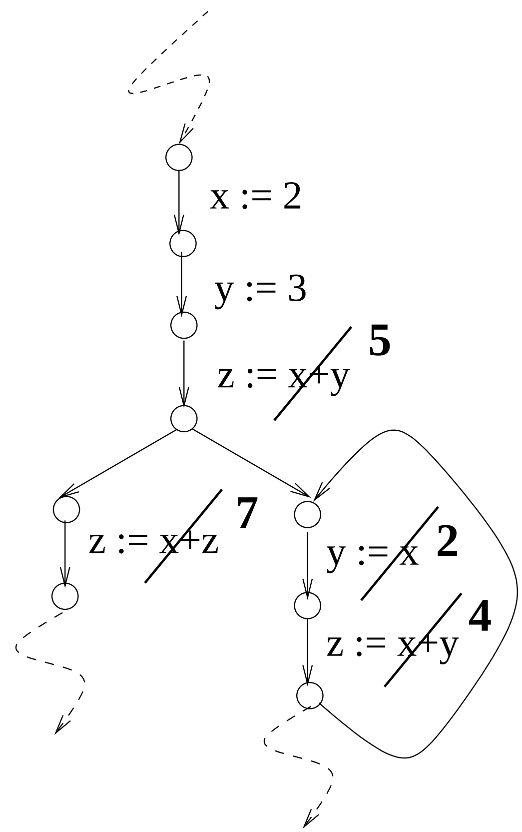
Constant Propagation

The very idea...

a)



b)



Original program

After simple constant propagation

Constant Propagation Reconsidered

Remember

- Kildall's algorithm for simple constants (SC) (POPL'73)

and Kenneth's talk on Monday morning on further attacks...

- Wegbreit (1st attack)
- Lewis, Tarjan, and Reif (2nd attack)
- Wegman and Zadeck (3rd attack)
- ...

Constant Propagation Reconsidered (Cont'd)

Advancements of Kildall's work on SC aimed at...

- Scope
  - Interprocedurally

    Callahan, Cooper, Kennedy, Torczon (SCC'86)

    Grove, Torczon (PLDI'93)

    Metzer, Stroud (LOPLAS, 1993)

    Sagiv, Reps, Horwitz (TAPSOFT'95)

    Duesterwald, Gupta, Soffa (TOPLAS, 1997)
  - Explicitly parallel

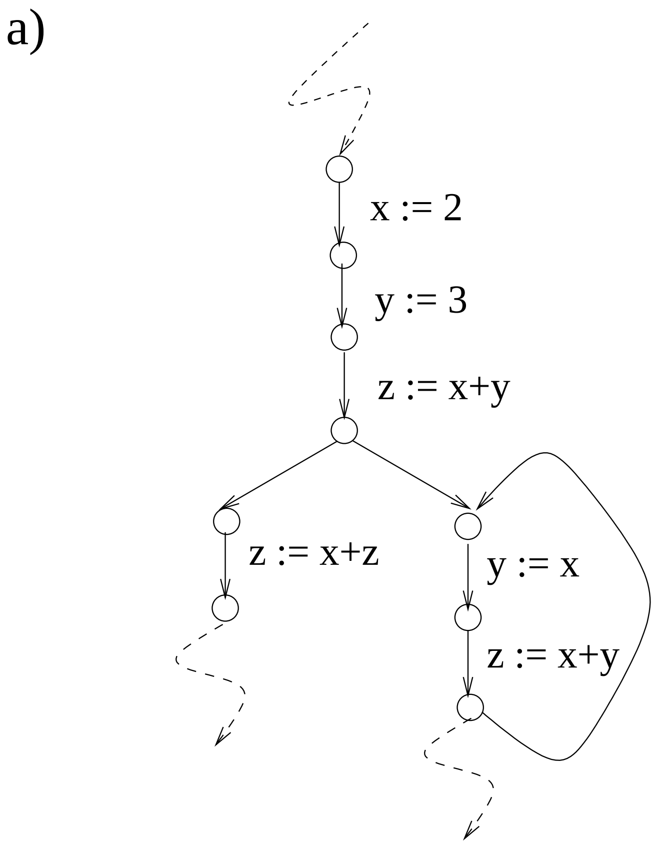
    Lee, Midkiff, Padua (J. of Parallel Prog., 1998)

    Knoop (Euro-Par'98)

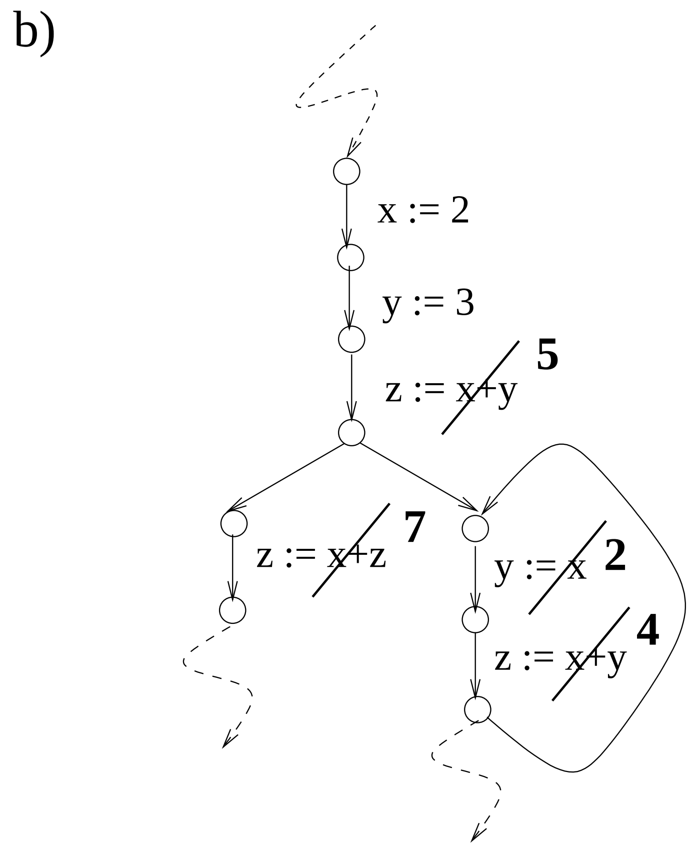
Constant Propagation Reconsidered (Cont'd)

- Performance
  - **SSA**: Wegman, Zadeck (POPL'85)
- Expressivity
  - "SC+": Kam, Ullman (Acta Inf., 1977)
  - Conditional Constants: Wegman, Zadeck (POPL'85)
  - Finite Constants: Steffen, Knoop (MFCS'89)

Why Striving for Greater Expressivity?



Original program



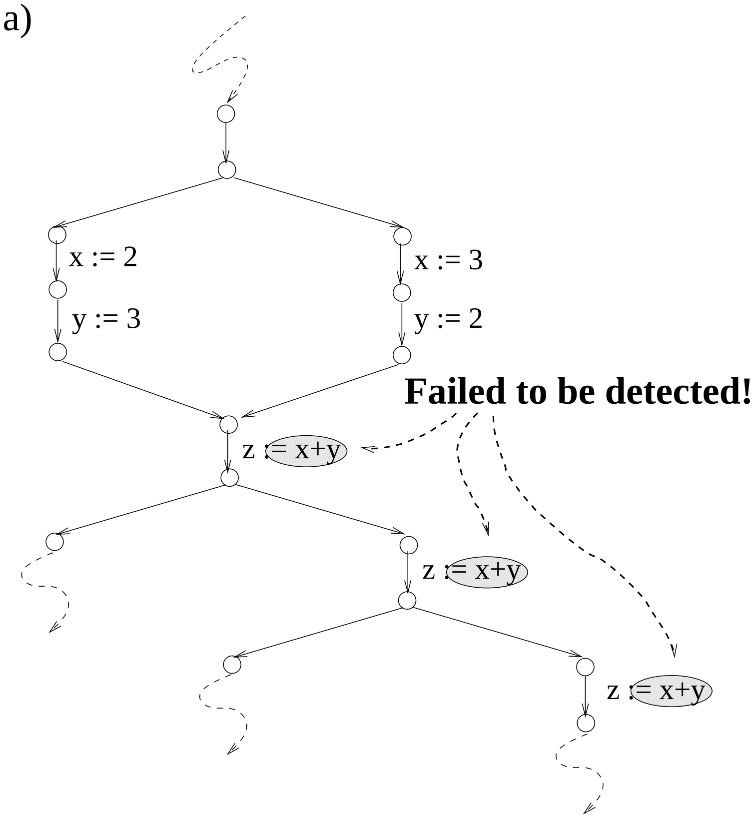
After simple constant propagation

It's ok, isn't it?

Actually, it is not

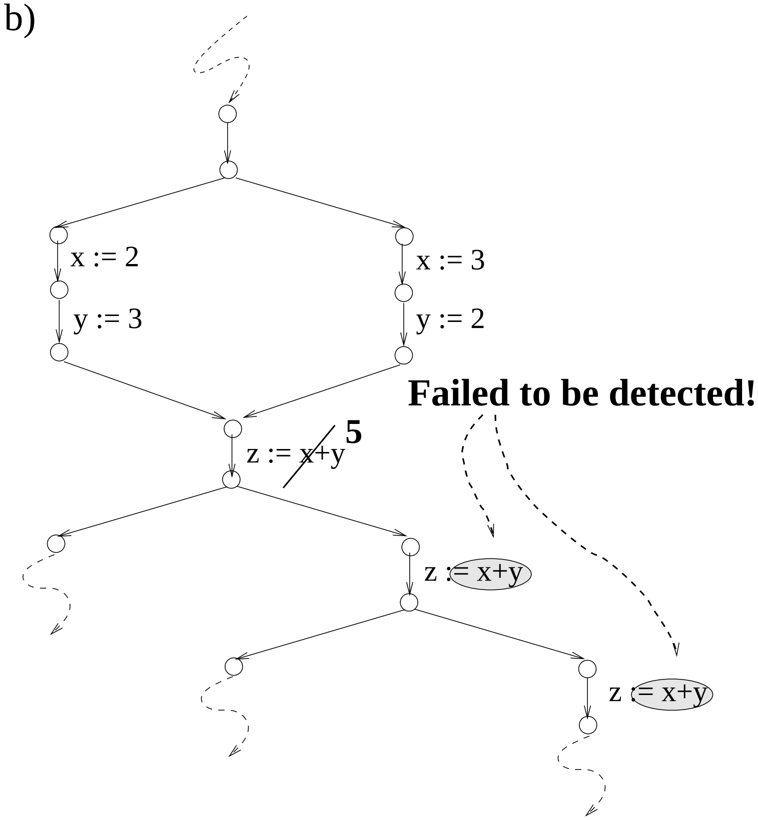
Simple constants are weak...

a)



After simple constant propagation
(Note: No effect at all!)

b)



After simple constant propagation
enhanced by the "look−ahead−of−one" heuristics
of Kam and Ullman

Decidability Issues of Constant Propagation

As a matter of fact...

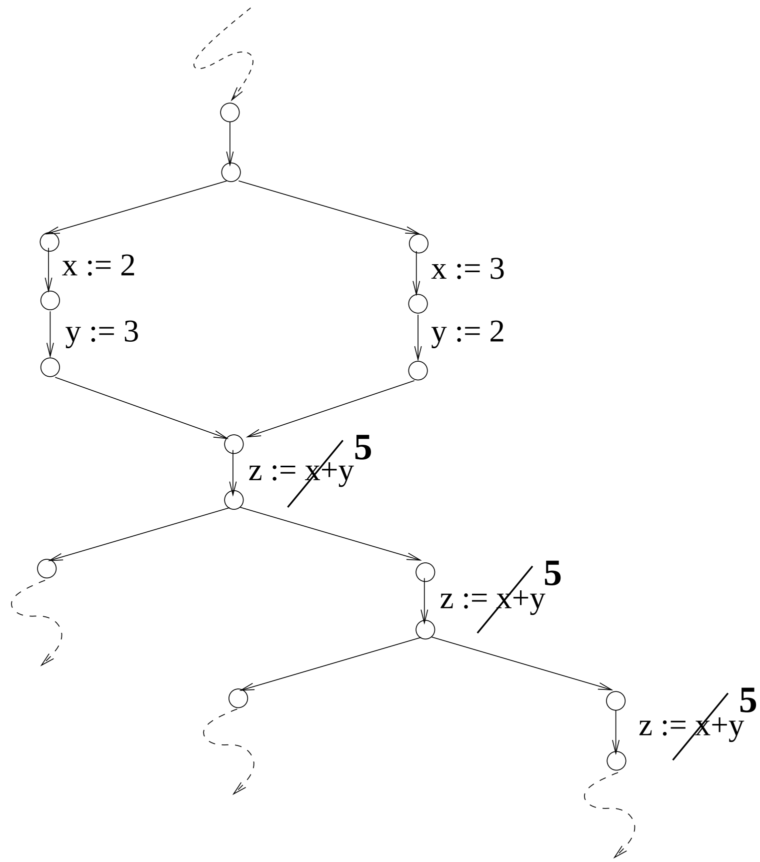
- Constant propagation is undecidable

On the other hand...

- Constant propagation is decidable on DAGs

Finite Constants (FC)

...are optimal on DAGs!



Finite Constants (Cont'd)

Intuitively

- FC are a systematic, exhaustive, and finitely computable extension of Kam&Ullman's "look-ahead of one" heuristics

Key Facts on Finite Constants

- Proper extension of SC for unrestricted control flow
- Optimal on DAGs
- Exponential worst-case time complexity (even on DAGs)

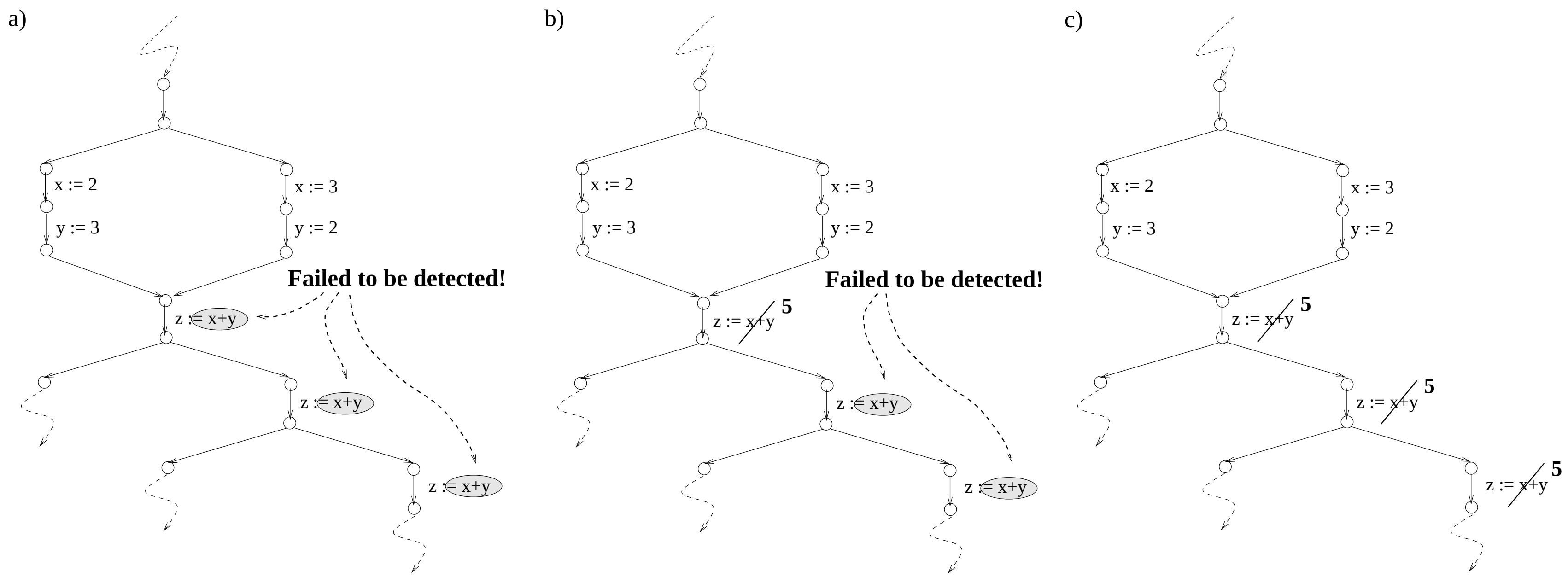
Note

- Constant Propagation on DAGs is Co-NP-Complete

Knoop, Rüthing (CC'00)

Müller-Olm, Rüthing (ESOP'01)

Reconsidering the Running Example



After simple constant propagation
(Note: No effect at all!)

After simple constant propagation
enriched by the "look-ahead-of-one" heuristics
of Kam and Ullman

The effect of the new algorithm

A New CP Algorithm

...carefully balancing

- Expressivity and Performance

This new algorithm is...

- based on the Value Graph of Alpern, Wegman, and Zadeck (POPL'88)
- which itself is based on SSA

Hence: CP w/ SSA form (instead of on SSA form)

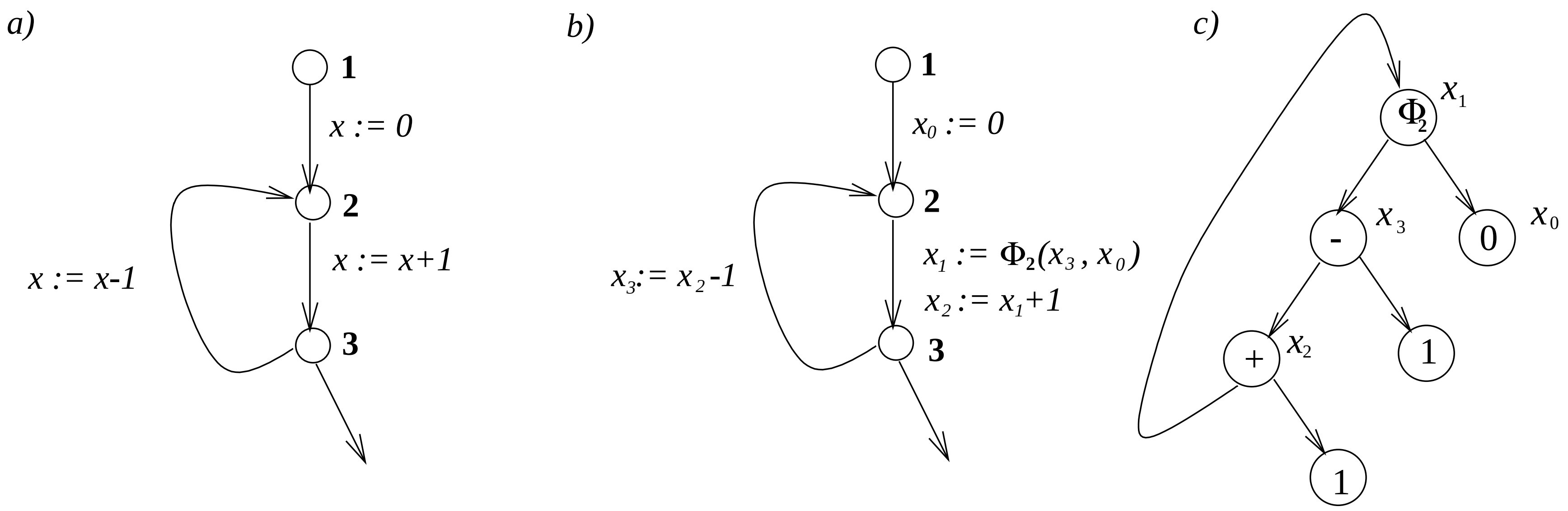
Part II: Constant Propagation w/ SSA Form

Own Work Related to Part II of the Talk

Joint work with Oliver Rüthing...

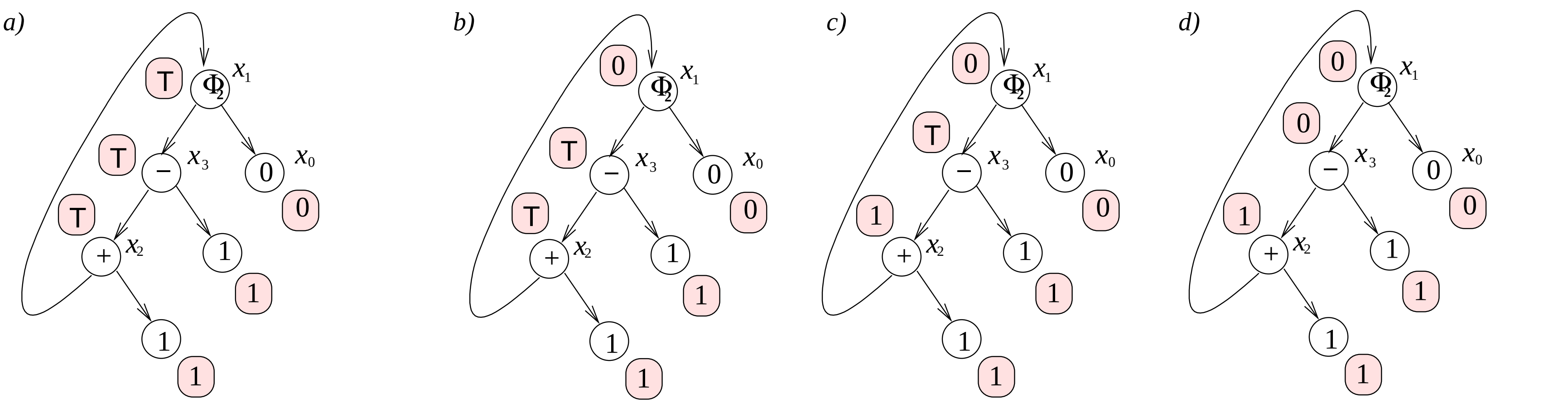
- Constant Propagation w/ SSA Form
  - *Constant Propagation on the Value Graph: Simple Constants and Beyond*. In Proc. 9th Int. Conf. on Compiler Construction (CC 2000), LNCS 1781 (2000), 94 - 109.

The Value Graph of Alpern, Wegman, and Zadeck



Original Program SSA Form Value Graph

Constant Propagation on the Value Graph



After the initialization step

After the 1st iteration step

After the 2nd iteration step

After the 3rd iteration step: Stable

Hence: x_2 and x_3 have constant values!

Constant Propagation on the Value Graph

...comes in two flavours

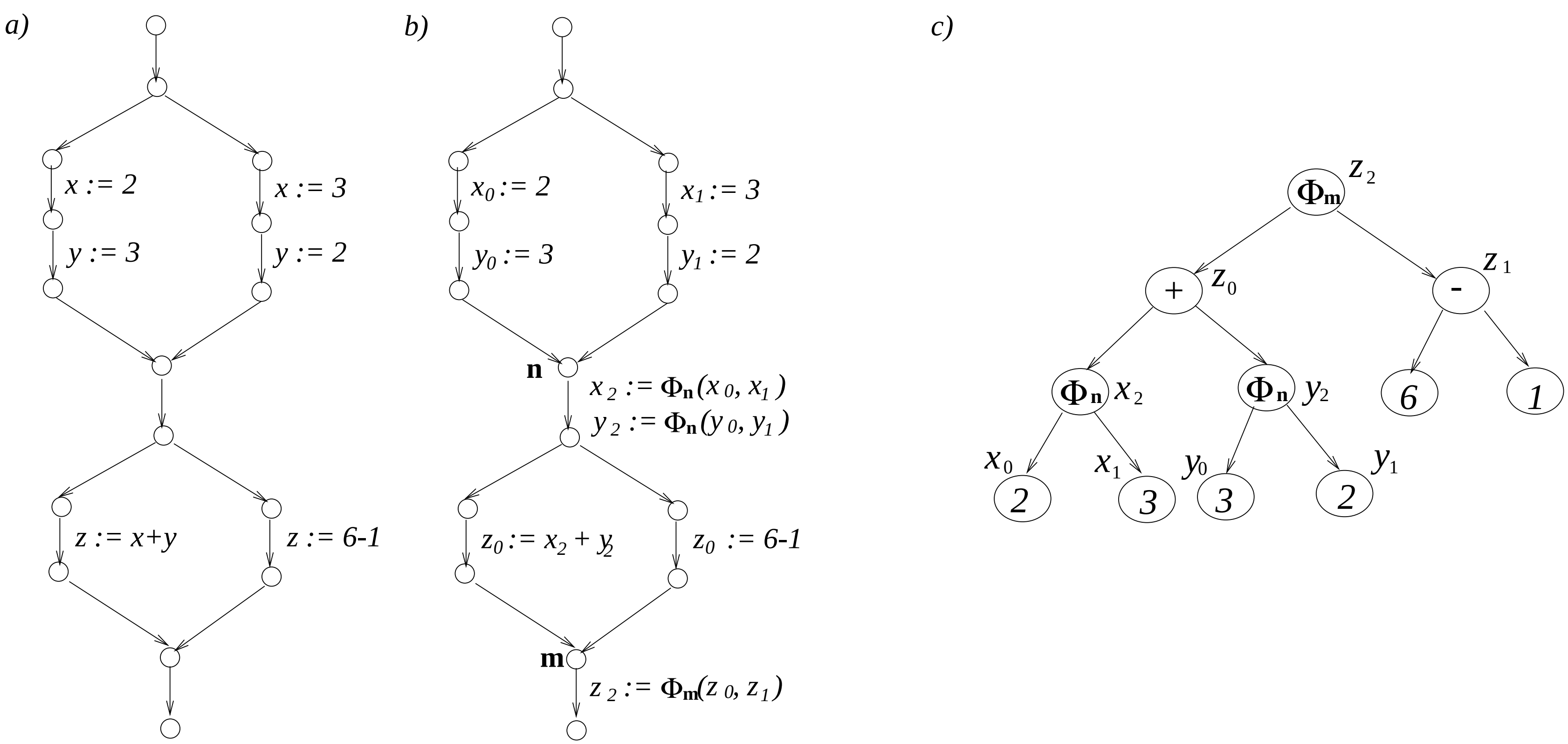
- The Basic Algorithm

  ...computes SC

- The Full Algorithm

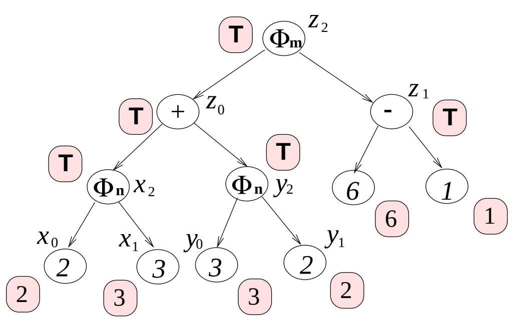
  ...goes beyond and integrates the look-ahead heuristics

A New Example for Illustrating the Full Algorithm

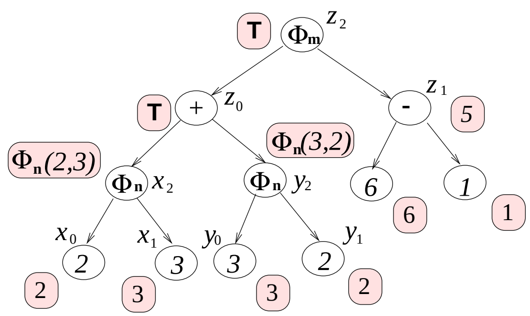


Original Program   SSA Form   Value Graph

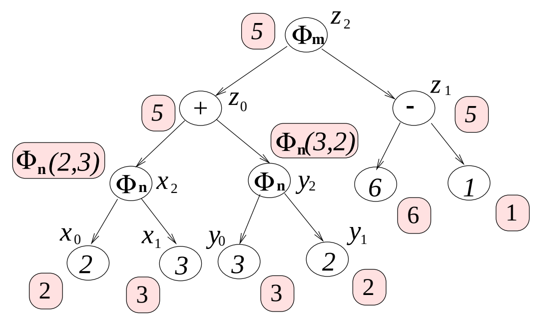
The Full Algorithm on the Value Graph



The start annotation



After the first iteration



After the second iteration. Stable!

Clou: Introducing Φ-Constants and

Adapting the Evaluation Function on Value Graphs!

Main Results

Unrestricted Control-Flow...

- The full algorithm detects a superset of SC (even constants, which are no finite constants!)

Acylic Control-Flow...

- The full algorithm detects every constant, which is only composed of operators, which are injective in their relevant arguments

Overall...

- Nicely balances expressivity and performance
- SSA and the Value Graph are key

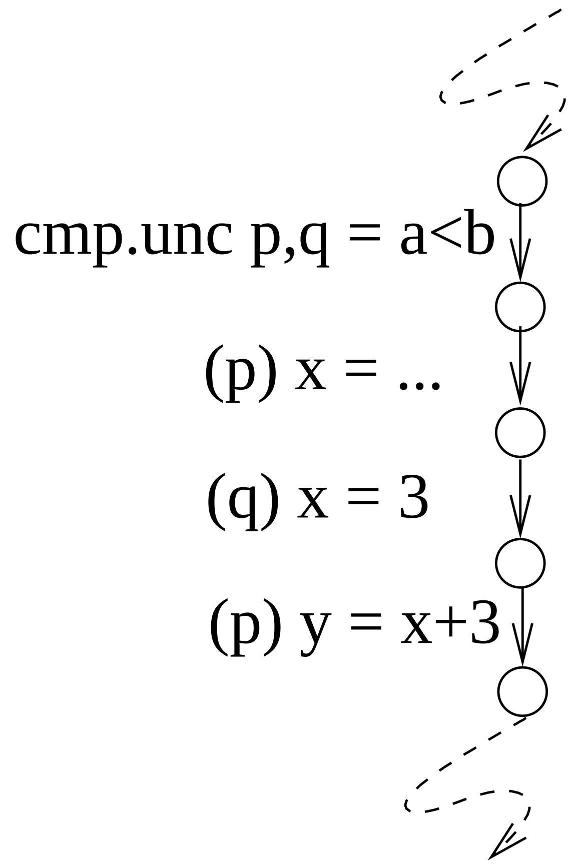
Part III: Constant Propagation w/ Predicated SSA Form

Own Work Related to Part III of the Talk

Joint work with Oliver Rüthing...

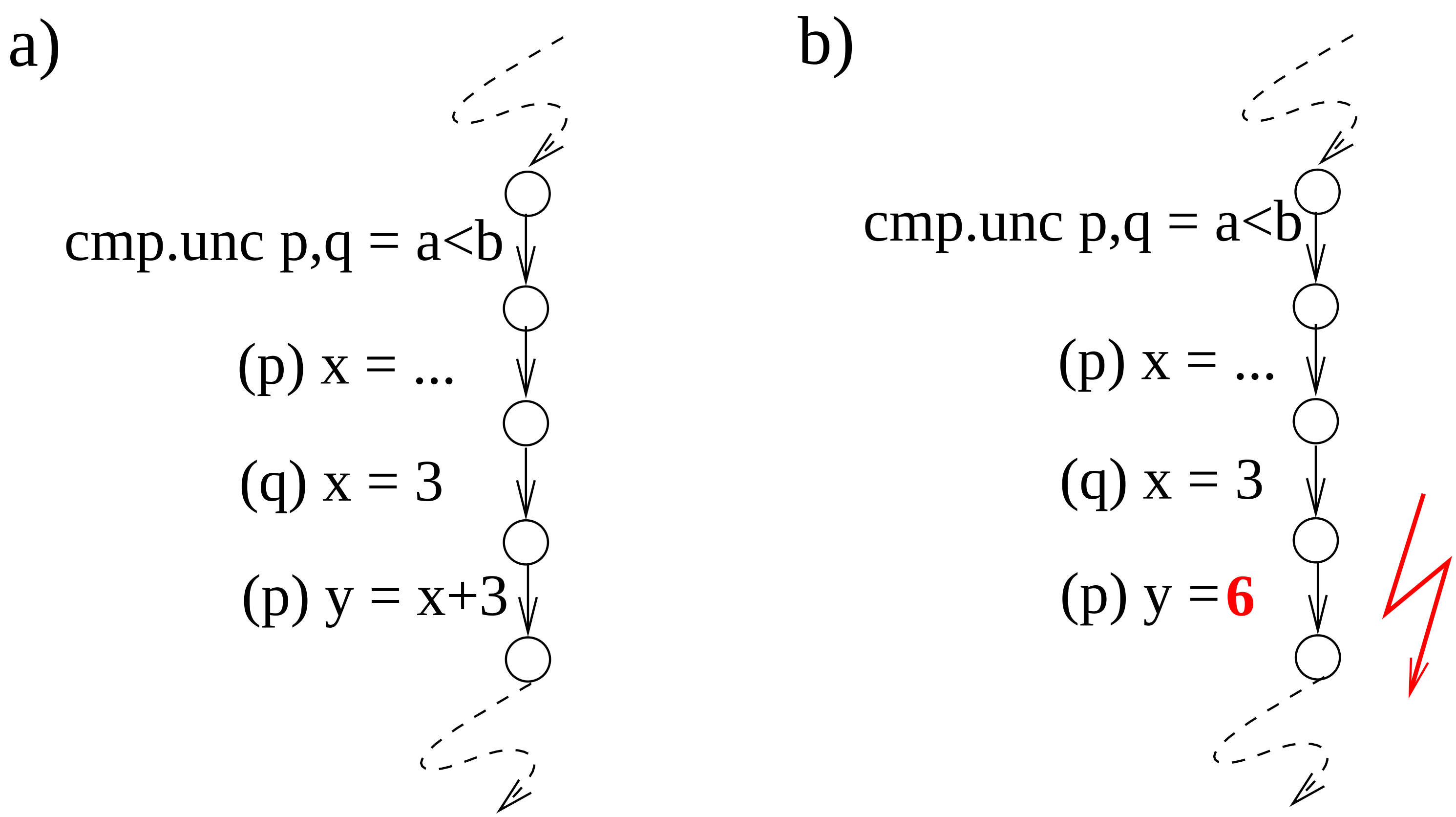
- Constant Propagation w/ Predicated SSA Form
  - *Constant Propagation on Predicated Code*. J. of Universal Computer Science 9, 8 (2003), 829 - 850. (special issue devoted to SBLP'03).
  - *Constant Propagation on Predicated Code*. In Proc. 7th Brazilian Symp. on Programming Languages (SBLP 2003), 135 - 148.

Predicated Code

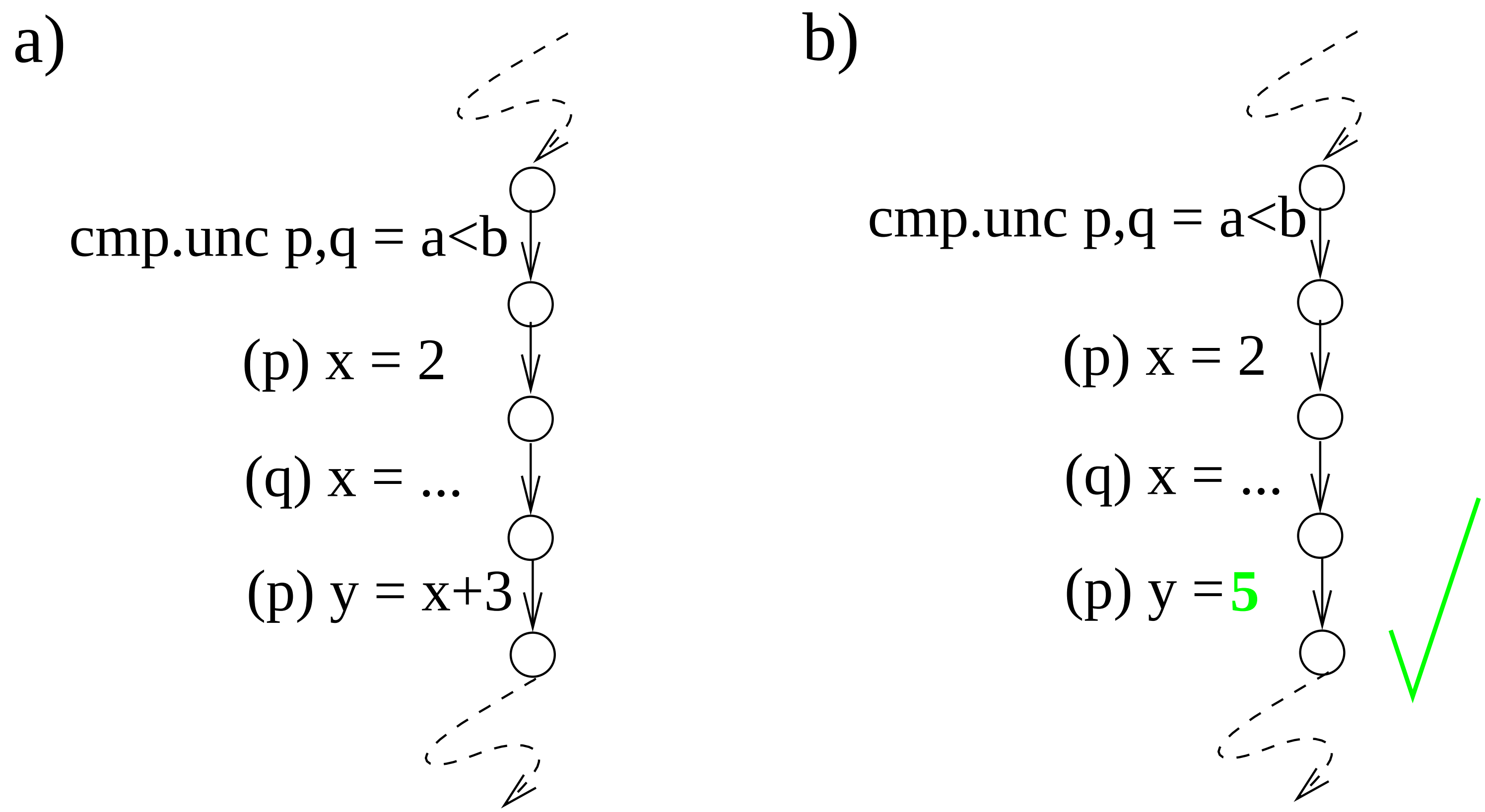


...resulting from if-conversion.

Performing CP Naively on Predicated Code Fails...



On the Other Hand...



...naive sound CP is likely to be too conservative and to miss many optimization opportunities!

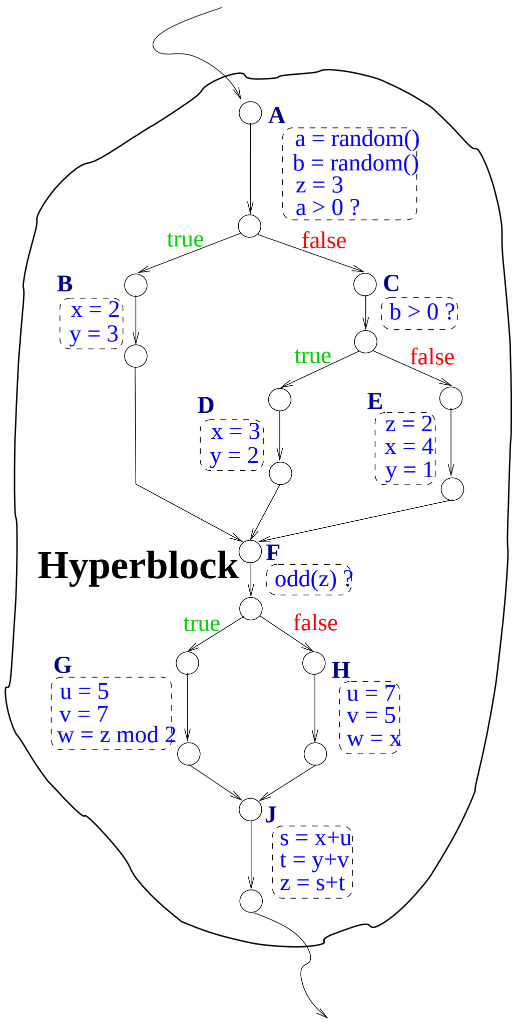
Workplan: Handling Predicated Code more Smartly

Hyperblocks

...important building blocks in predicated code.

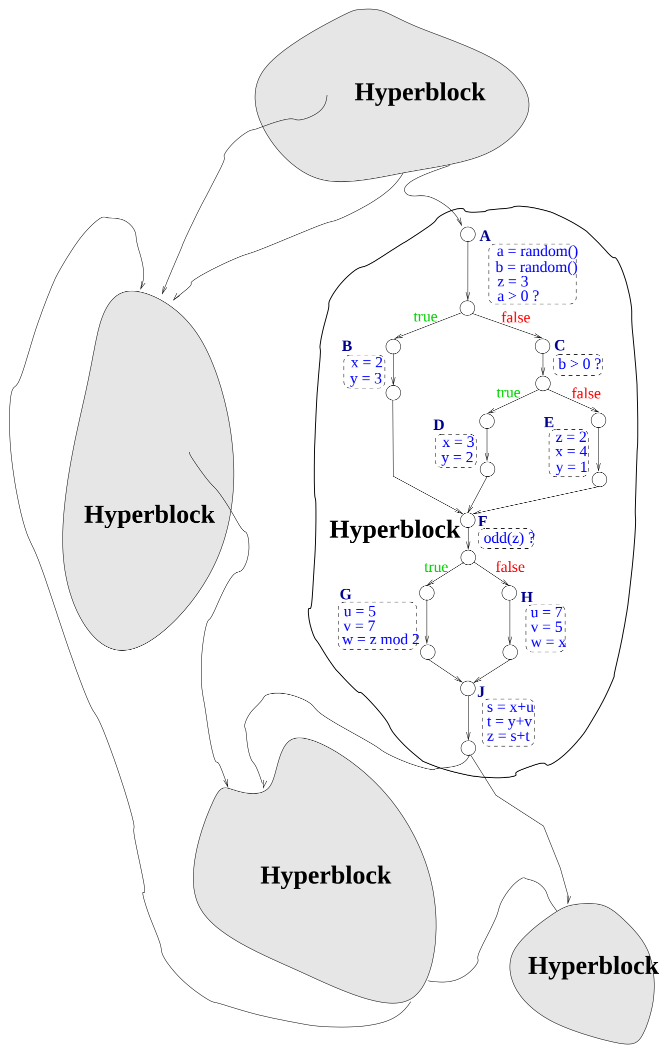
A Hyperblock

Single entry, multiple exits...



Embedded into a Program

The running example...



The New CP Algorithm on Predicated Code

...comes in two/plus flavours

- The Basic Algorithm
- The Full Algorithm

plus

- Performance-tuned Variants

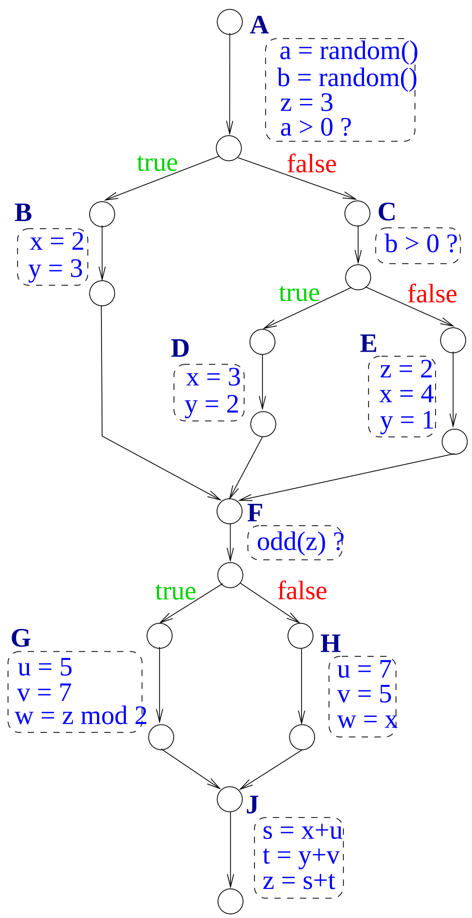
Each consisting of a

- global
- local

stage.

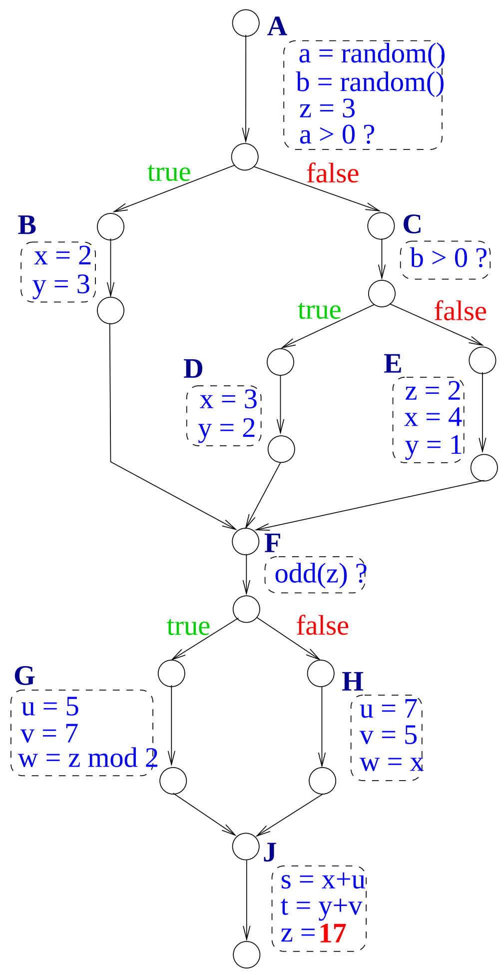
Discussing the Local Stage

The hyperblock we will focus on...



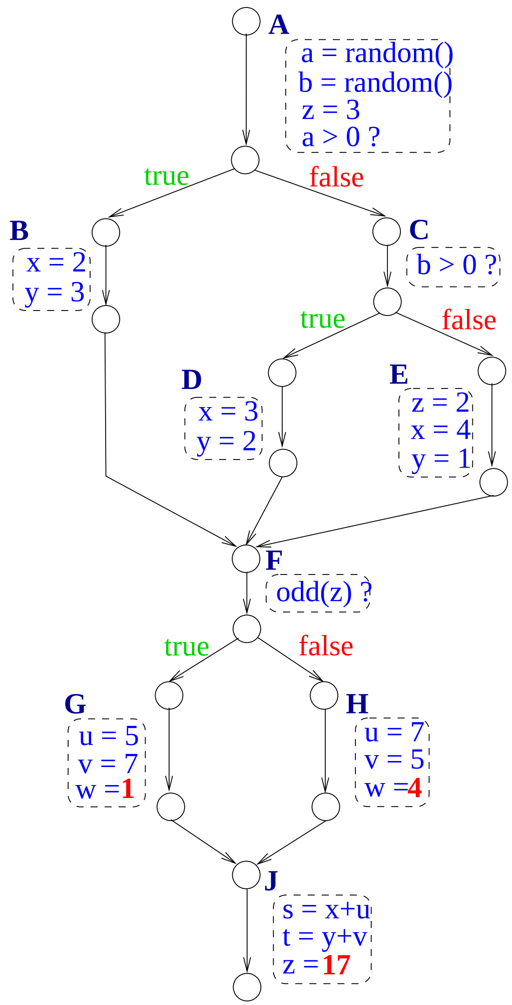
Original Hyperblock

Optimization of the Basic Algorithm



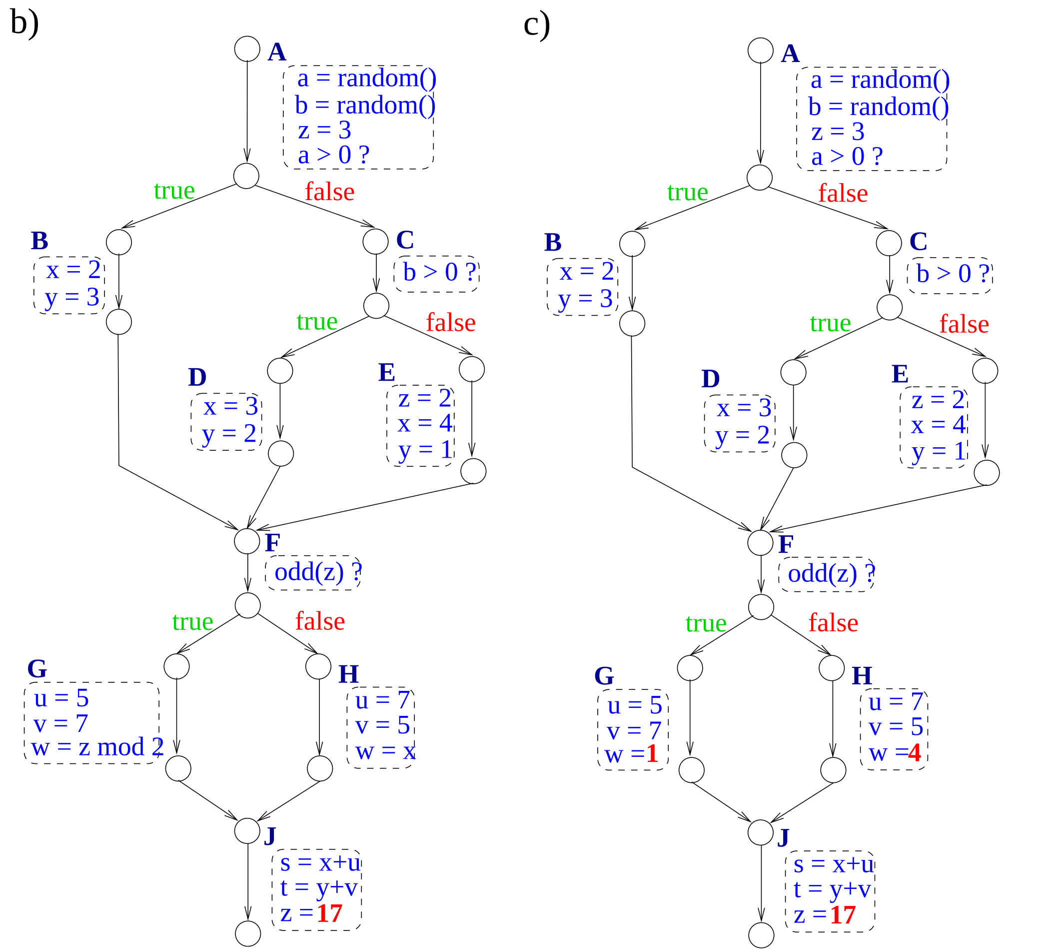
The Non−Deterministic Path−Precise
Basic Optimization

Optimization of the Full Algorithm



The Deterministic Path−Precise
Full Optimization

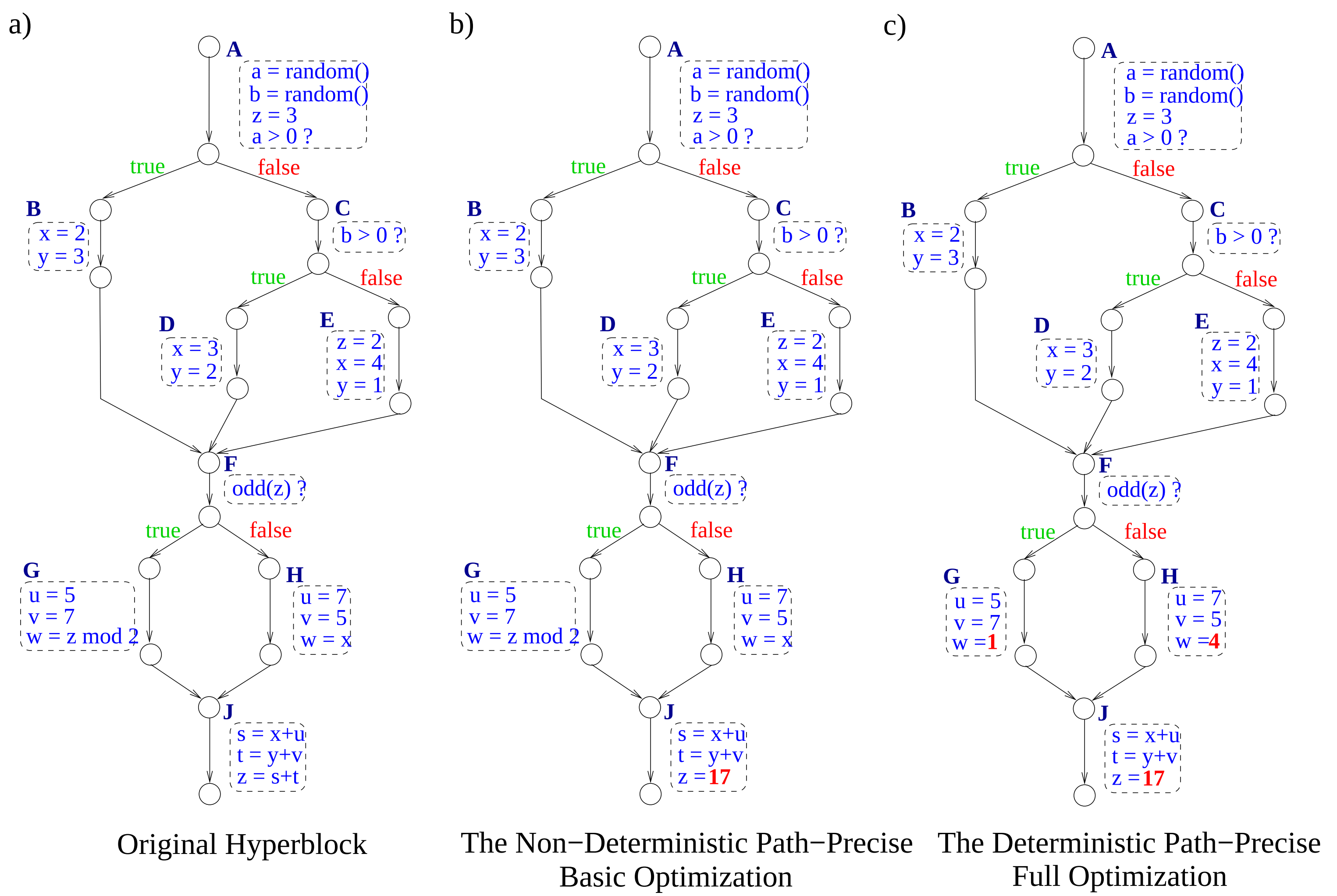
Optimizations of Basic and Full Alg. at a Glance



The Non−Deterministic Path−Precise Basic Optimization

The Deterministic Path−Precise Full Optimization

Optimizations of Basic and Full Alg. at a Glance



Original and Predicated Code

```
begin \\ Original Hyperblock   | begin \\ After if-Conversion
 (a,b) = (random(),random());  |  (p0) (a,b) = (random(),random());
 z = 3;                        |  (p0) z = 3;
 if a>0 then                   |  (p0) cmp.unc B,C (a>0);
   x = 2;                      |  (B)  x = 2;
   y = 3                       |  (B)  y = 3;
 elsif b>0 then                |  (C)  cmp.unc D,E (b>0);
   x = 3;                      |  (D)  x = 3;
   y = 2                       |  (D)  y = 2;
 else                          |
   z = 2;                      |  (E)  z = 2;
   x = 4;                      |  (E)  x = 4;
   y = 1 fi;                   |  (E)  y = 1;
 if odd(z) then                |  (p0) cmp.unc G,H (odd(z));
   u = 5;                      |  (G)  u = 5;
   v = 7;                      |  (G)  v = 7;
   w = z mod 2                 |  (G)  w = z mod 2;
 else                          |
   u = 7;                      |  (H)  u = 7;
   v = 5;                      |  (H)  v = 5;
   w = x fi;                   |  (H)  w = x;
 s = x+u;                      |  (p0) s = x+u;
 t = y+v;                      |  (p0) t = y+v;
 z = s+t end.                  |  (p0) z = s+t end.
```

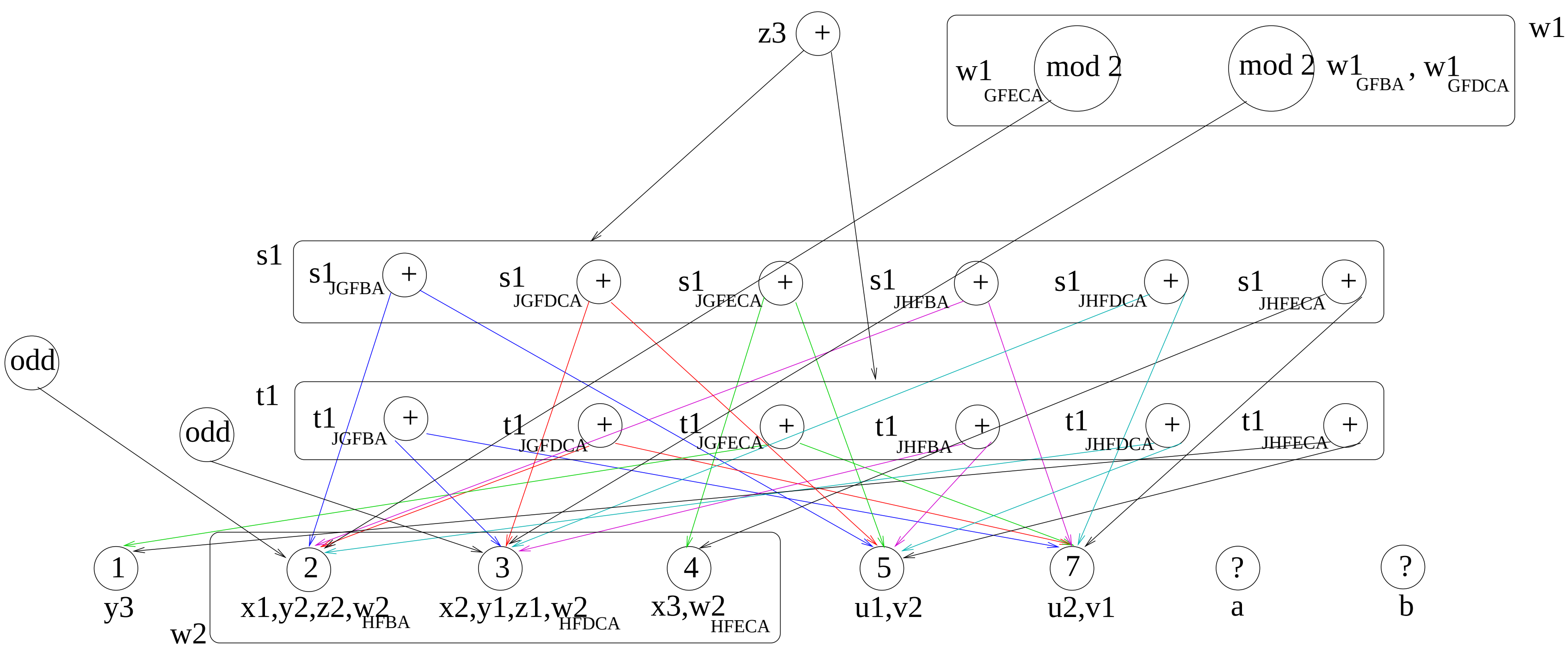
Predicated SSA

...by Carter, Simon, Calder, Ferrante (PACT’99)

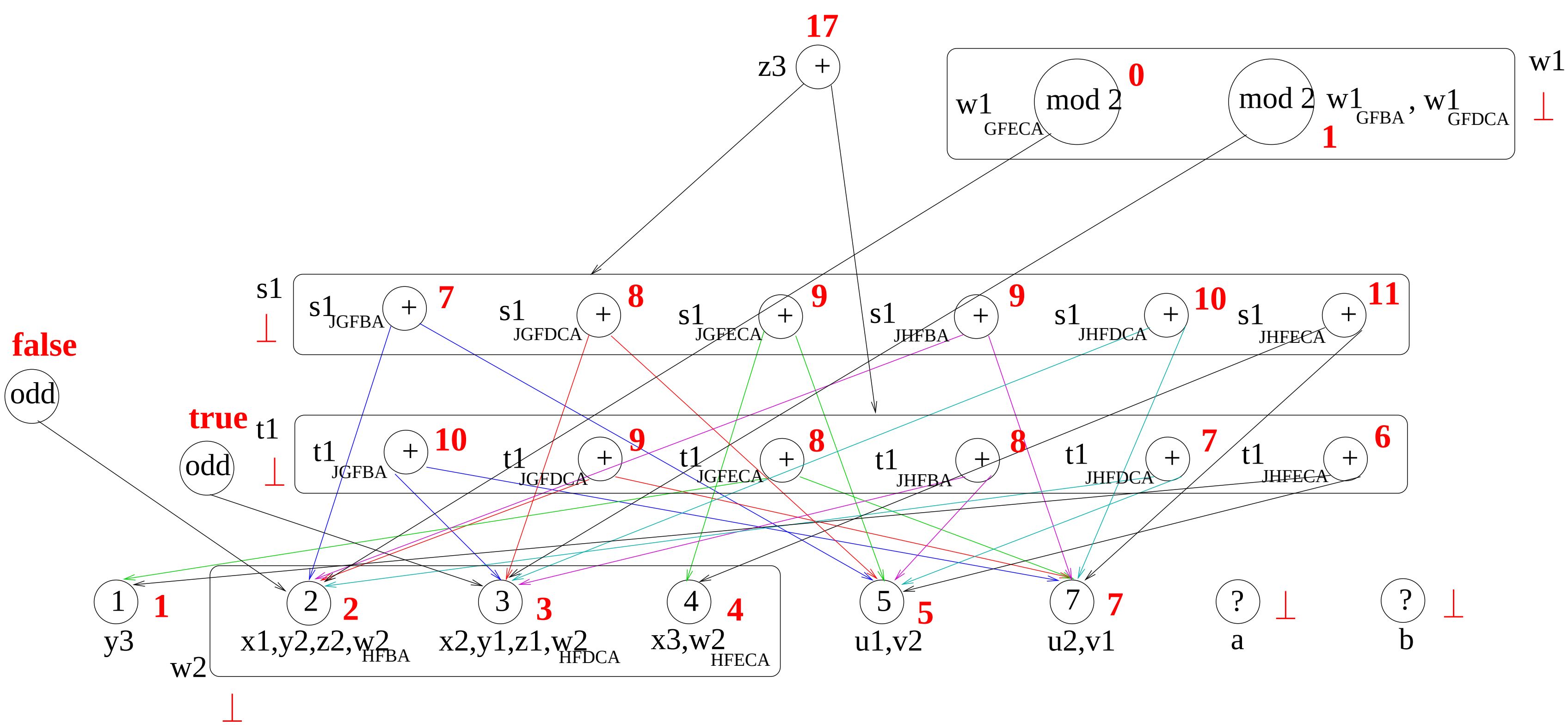
```
begin (p0)      A = OR(TRUE);
      (A)       (a1,b1) = (random(),random());
      (A)       z1 = 3;
      (A)       cmp.unc BA,CA (a1>0);
      (p0)      B = OR(BA);
      (p0)      C = OR(CA);
      (B)       x1 = 2;
      (B)       y1 = 3;
      (C)       cmp.unc DCA,ECA (b1>0);
      (p0)      D = OR(DCA);
      (p0)      E = OR(ECA);
      (D)       x2 = 3;
      (D)       y2 = 2;
      (E)       z2 = 2;
      (E)       x3 = 4;
      (E)       y3 = 1;
      (BA)      FBA = OR(TRUE);
      (DCA)     FDCA = OR(TRUE);
      (ECA)     FECA = OR(TRUE);
      (p0)      F = OR(FBA,FDCA,FECA);
      (FBA)     cmp.unc GFBA,HFBA (odd(z1));
      (FDCA)    cmp.unc GFDCA,HFDCA (odd(z1));
      (FECA)    cmp.unc GFECA,HFECA (odd(z2));
  [-] (p0)      G = OR(GFBA,GFDCA,GFECA);
  [-] (p0)      H = OR(HFBA,HFDCA,HFECA);
      (GFBA)    w1 = z1 mod 2;
      (GFDCA)   w1 = z1 mod 2;
  [*] (GFECA)   w1 = z2 mod 2;
      (G)       u1 = 5;
      (G)       v1 = 7;

[*] (HFBA)    w2 = x1;
[*] (HFDCA)   w2 = x2;
    (HFECA)   w2 = x3;
    (H)       u2 = 7;
    (H)       v2 = 5;
    (GFBA)    JGFBA = OR(TRUE);
    (GFDCA)   JGFDCA = OR(TRUE);
[*] (GFECA)   JGFECA = OR(TRUE);
[*] (HFBA)    JHFBA = OR(TRUE);
[*] (HFDCA)   JHFDCA = OR(TRUE);
    (HFECA)   JHFECA = OR(TRUE);
[-] (p0)      J = OR(JGFBA,JGFDCA,
                     JGFECA,JHFBA,
                     JHFDCA,JHFECA);
    (JGFBA)   s1 = x1+u1;
    (JGFBA)   t1 = y1+v1;
[*] (JGFDCA)  s1 = x2+u1;
[*] (JGFDCA)  t1 = y2+v1;
    (JGFECA)  s1 = x3+u1;
    (JGFECA)  t1 = y3+v1;
[*] (JHFBA)   s1 = x1+u2;
[*] (JHFBA)   t1 = y1+v2;
[*] (JHFDCA)  s1 = x2+u2;
[*] (JHFDCA)  t1 = y2+v2;
    (JHFECA)  s1 = x3+u2;
    (JHFECA)  t1 = y3+v2;
    (J)       z3 = s1+t1;
  end.
```

The Basic Predicated Value Graph based on PSSA Form

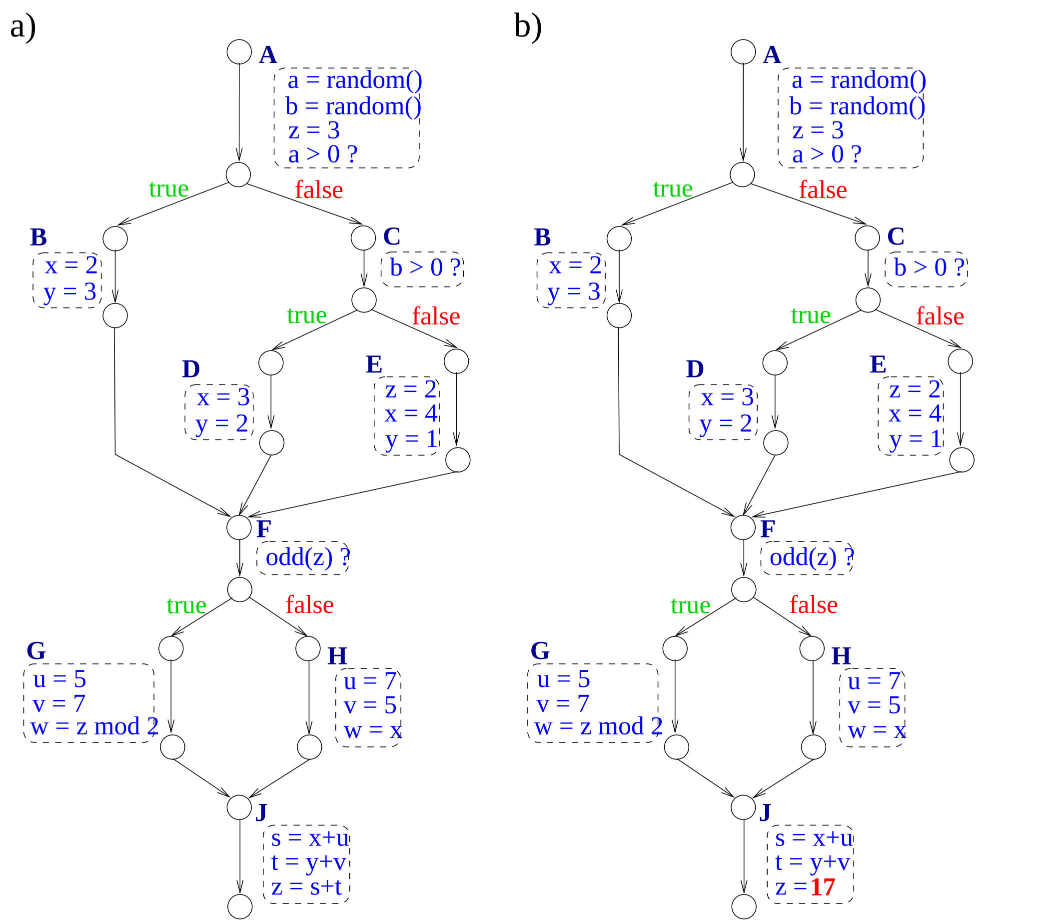
W/out taking advantage of guarding predicates...



After CP on the Basic PVG / Basic Algorithm



Optimization of the Basic Algorithm

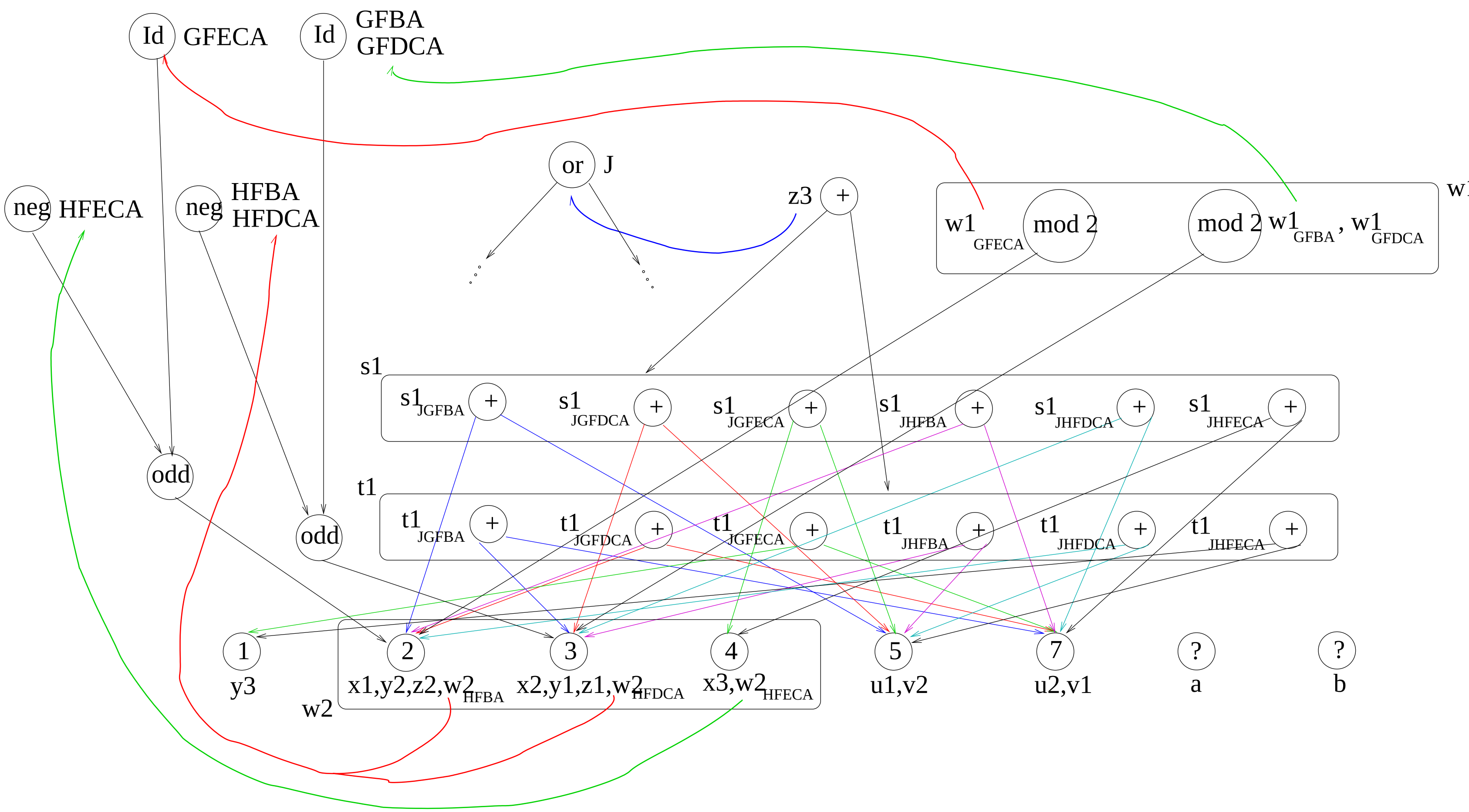


Original Hyperblock

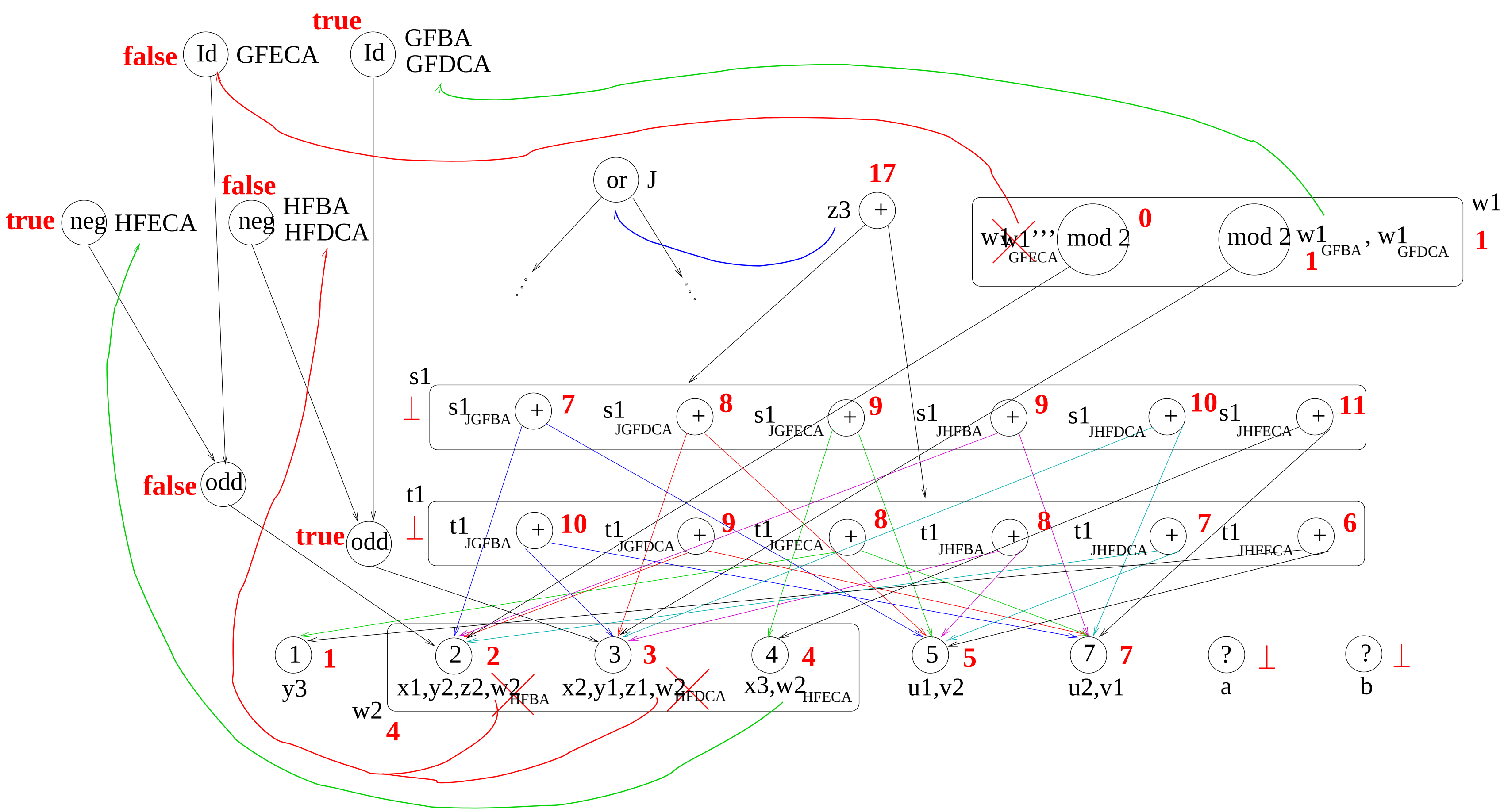
The Non−Deterministic Path−Precise Basic Optimization

The Predicated Value Graph

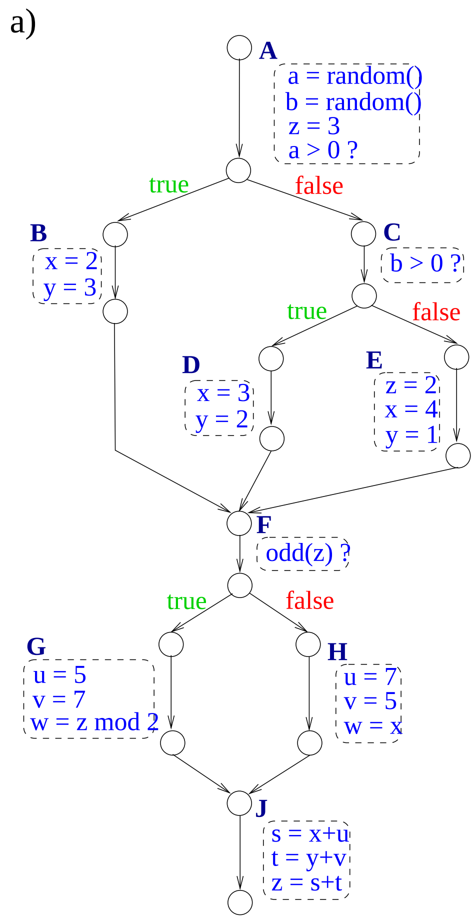
Taking advantage of guarding predicates...



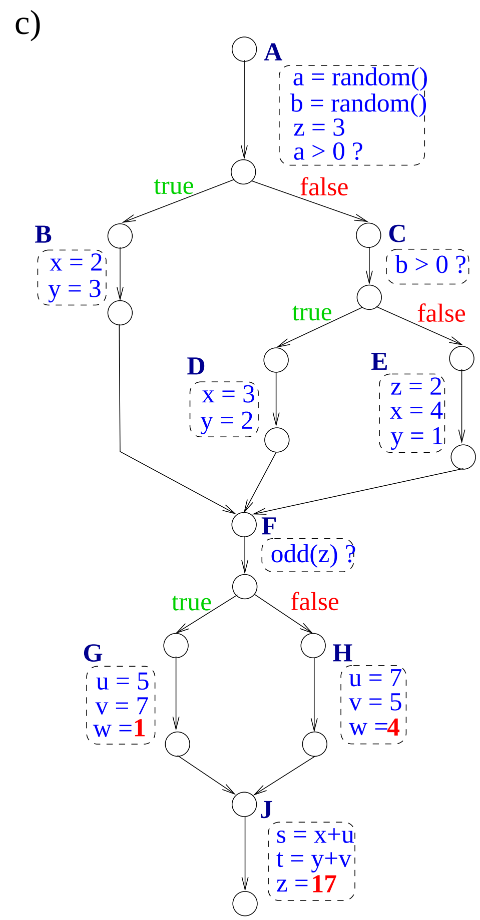
After CP on the PVG / Full Algorithm



Optimization of the Full Algorithm



Original Hyperblock



The Deterministic Path−Precise
Full Optimization

The Optimized Hyperblock in PSSA Form

```
begin (p0)      A = OR(TRUE);                  |
      (A)       a1 = random();                 | [-] (p0)      G = OR(GFBA,GFDCA);
      (A)       b1 = random();                 | [-] (p0)      H = OR(HFECA);
      (A)       z1 = 3;                        |     (G)       w1 = 1;
      (A)       cmp.unc BA,CA (a1>0);          |     (G)       u1 = 5;
      (p0)      B = OR(BA);                    |     (G)       v1 = 7;
      (p0)      C = OR(CA);                    |     (HFECA)   w2 = 4;
      (B)       x1 = 2;                        |     (H)       u2 = 7;
      (B)       y1 = 3;                        |     (H)       v2 = 5;
      (C)       cmp.unc DCA,ECA (b1>0);        |     (GFBA)    JGFBA = OR(TRUE);
      (p0)      D = OR(DCA);                   |     (GFDCA)   JGFDCA = OR(TRUE);
      (p0)      E = OR(ECA);                   |     (HFECA)   JHFECA = OR(TRUE);
      (D)       x2 = 3;                        | [-] (p0)      J = OR(JGFBA,JGFECA,
      (D)       y2 = 2;                        |                      JHFECA);
      (E)       z2 = 2;                        |     (JGFBA)   s1 = 7;
      (E)       x3 = 4;                        |     (JGFBA)   t1 = 10;
      (E)       y3 = 1;                        |     (JGFECA)  s1 = 9;
      (BA)      FBA = OR(TRUE);                |     (JGFECA)  t1 = 8;
      (DCA)     FDCA = OR(TRUE);               |     (JHFECA)  s1 = 11;
      (ECA)     FECA = OR(TRUE);               |     (JHFECA)  t1 = 6;
      (p0)      F = OR(FBA,FDCA,FECA);         |     (J)       z3 = 17;
      (FBA)     cmp.unc GFBA,HFBA (TRUE));     |    end.
      (FDCA)    cmp.unc GFDCA,HFDCA (TRUE);    |
      (FECA)    cmp.unc GFECA,HFECA (FALSE);   |
```

Main Results

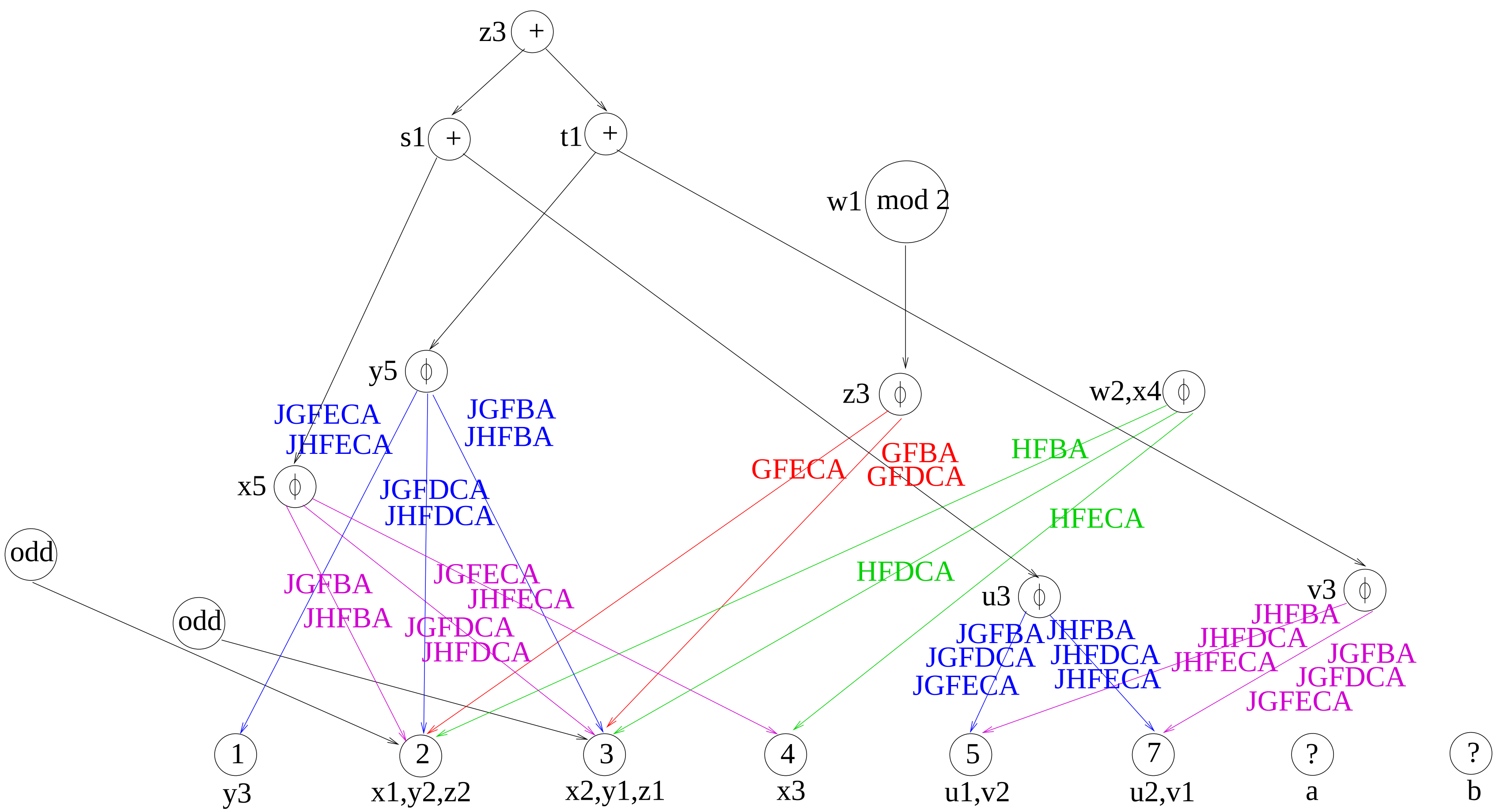
Soundness

- The global CP-Algorithm is sound (for both the basic and full algorithm of the local stage)

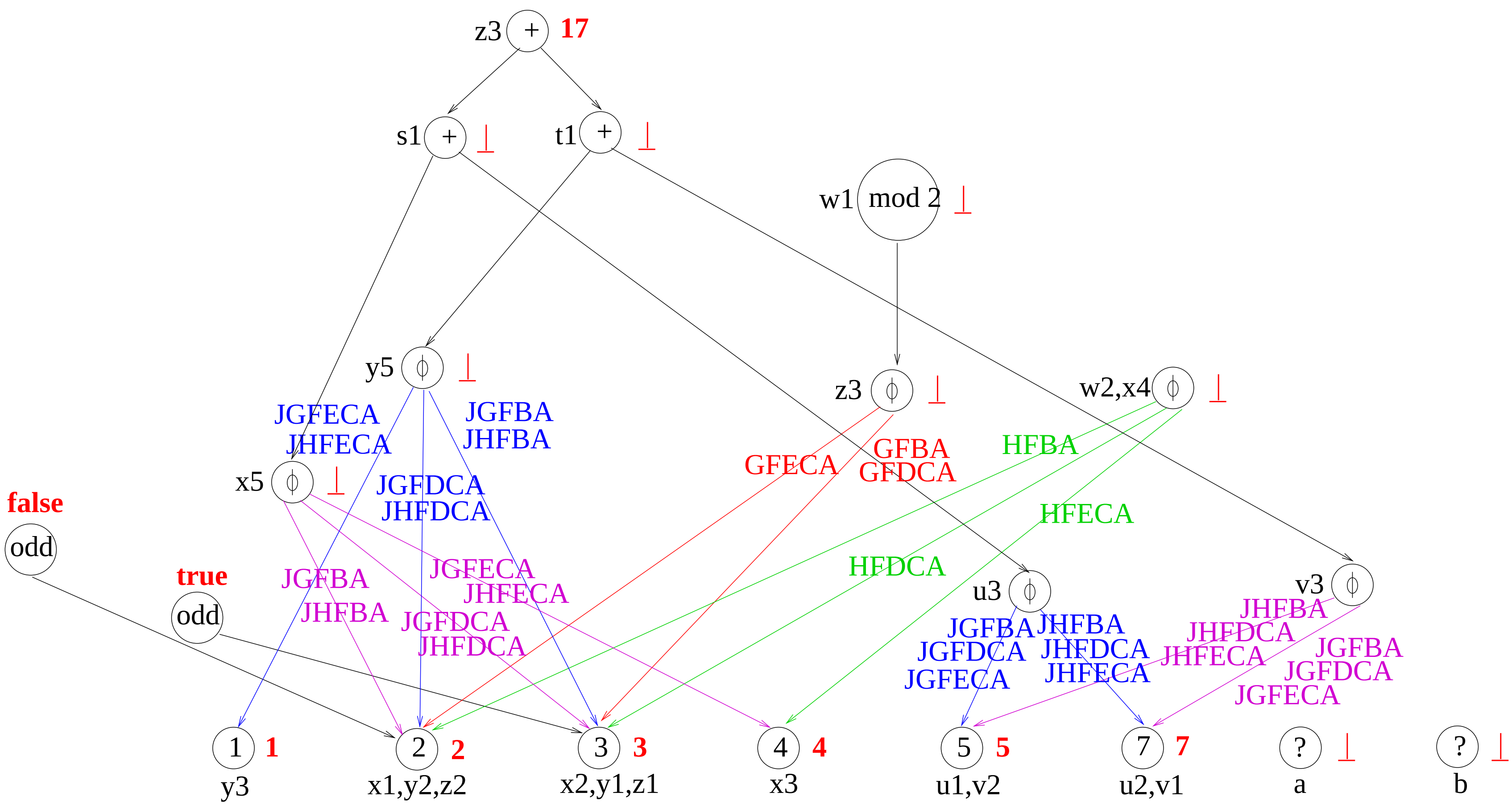
Completeness/Optimality

- The basic algorithm of the local stage is trace-precise wrt non-deterministic interpretation of branches
- The full algorithm of the local stage is predicate-sensitive trace-precise

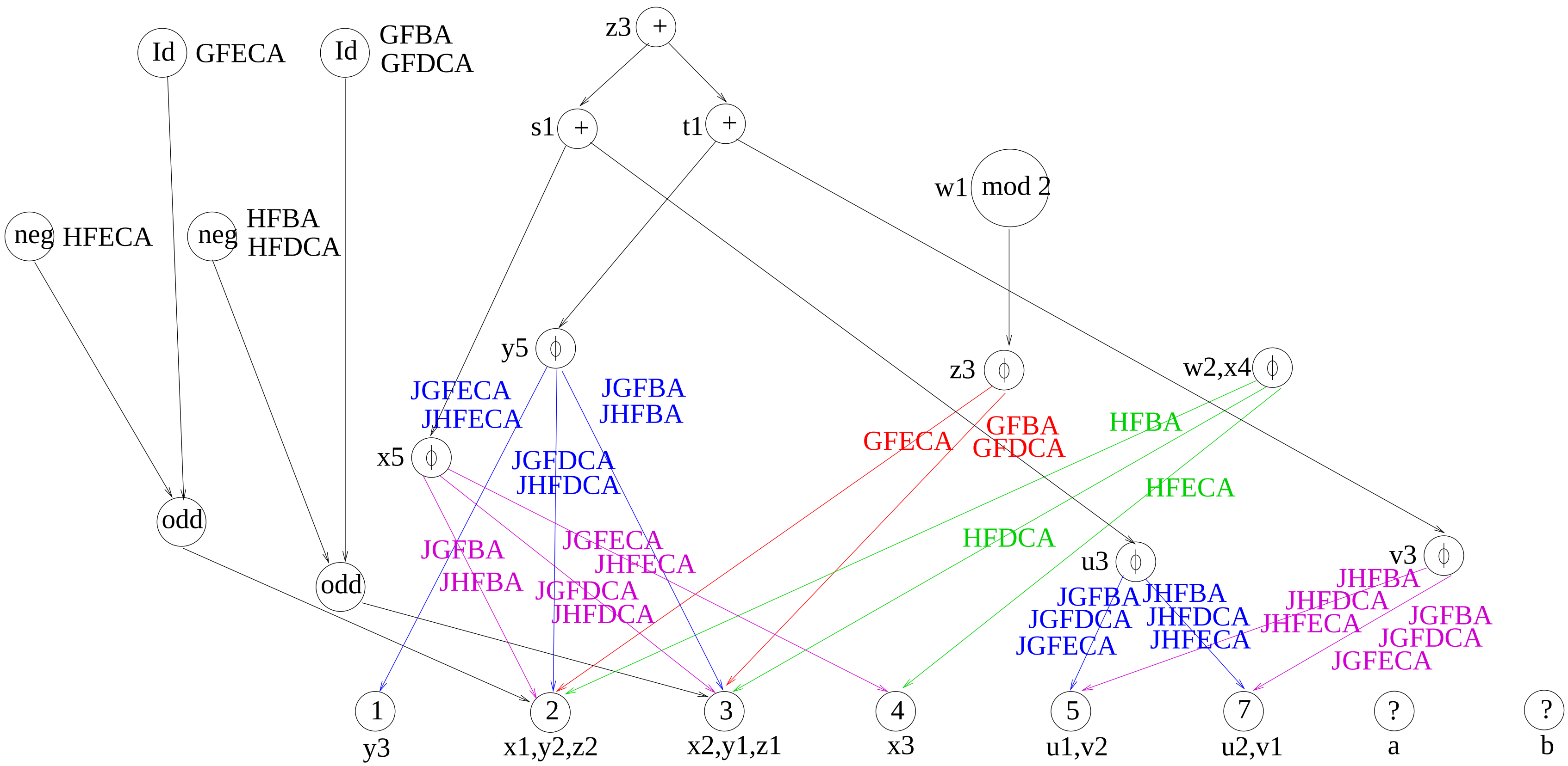
Tuning the Performance: Basic Algorithm



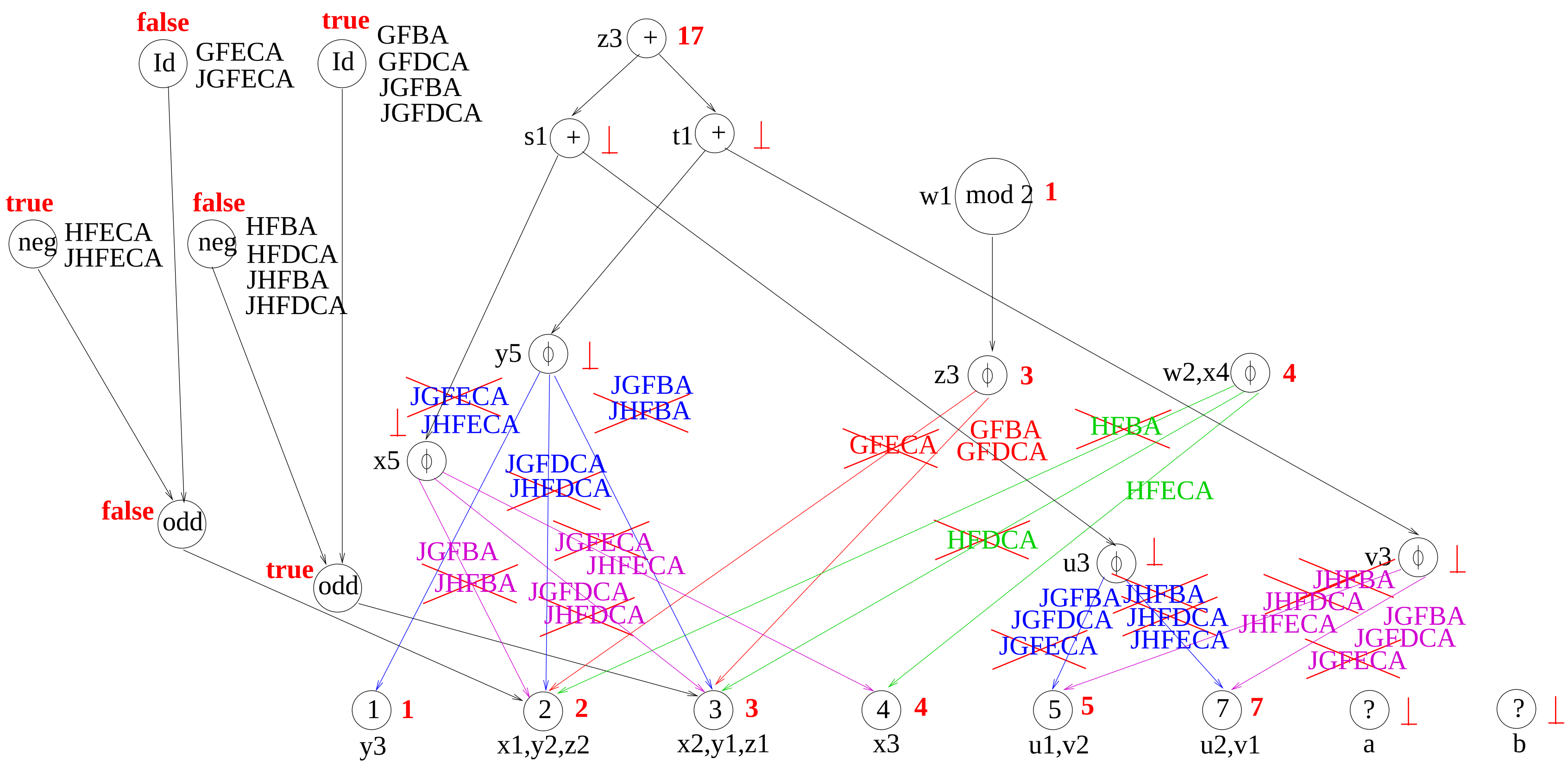
Tuning the Performance: Basic Alg. (Cont'd)



Tuning the Performance: Full Algorithm



Tuning the Performance: Full Alg. (Cont'd)



Conclusions

Constant Propagation and SSA/PSSA...

- a perfect match – SSA/PSSA really help!
- Key: Value Graph and Predicated Value Graph

Open to extensions, e.g.

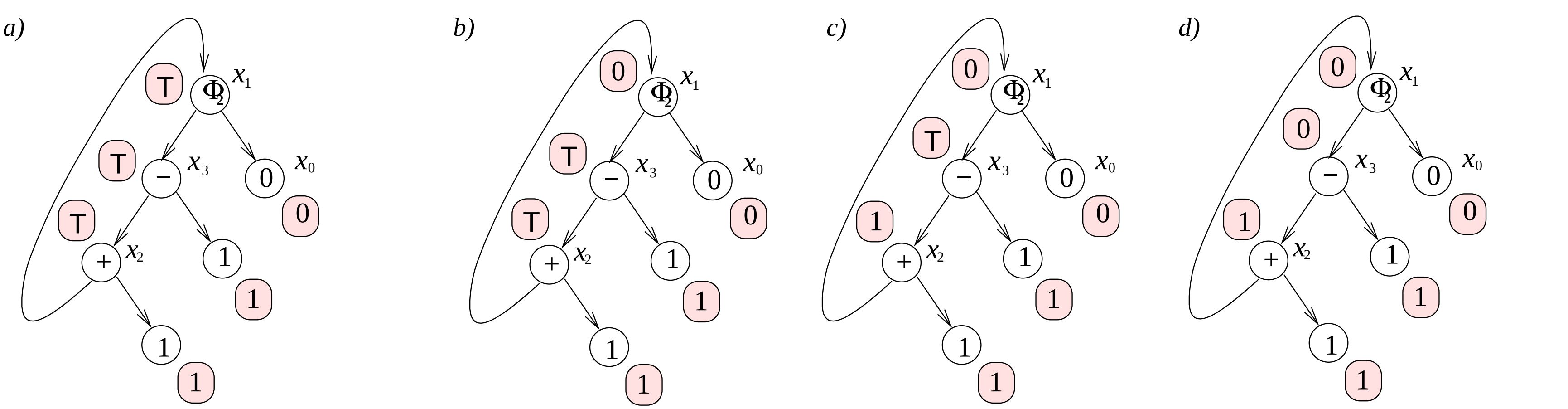
- Value Graph: Conditional Constants

Overall

- Especially neat example demonstrating the benefits of SSA

Constant Propagation w/SSA on the Value Graph

...with **Triple E** Rating: **E**xpressive, **E**fficient, **E**asy!



After the initialization step | After the 1st iteration step | After the 2nd iteration step | After the 3rd iteration step: Stable

Interests – Final Slide

In general...

- programming languages, compilers and everything related to it, especially
  - program analysis, transformation/optimization
  - WCET analysis

Interests – Final Slide (Cont'd)

Regarding SSA...

SSA & SSA extensions

- as intermediate program representation

and their usage/applications in

- program analysis, transformation/optimization