From Mozart and Freud to .Net Technologies

Jens Knoop

Vienna University of Technology, Austria

TECHNISCHE
UNIVERSITÄT
WIEN

VIENNA
UNIVERSITY OF
TECHNOLOGY

The Mystery Keynote !?

Disclosing the secret...

Wolfgang Amadeus Mozart

2006

Celebrating the
250th Anniversary of the Birth of Mozart!

1756 – 2006

Sigmund Freud

2006

Celebrating the
150th Anniversary of the Birth of Freud!

1856 – 2006

.Net Technologies

2006

Celebrating the
50th Anniversary of the First Release of .Net!

1956 – 2006

.Net Technologies

2006

Celebrating the
50th Anniversary of the First Release of .Net!

1956 – 2006

Really? After some investigation, unfortunately, not.

Started Thinking about the Question

Is there a topic in computer science researchers

- started to investigate as early as 1956, and
- still do, and
- which is of interest for the .Net-community?

Travelling Back in Time

...as far as possible.

Coming up with:

- The first issue of the

  Communications of the ACM

  did not appear until 1958!

Having a Closer Look

...at this very first issue of the CACM, I got quite excited:

- Ershov, A. P. *On Programming of Arithmetic Operations*. CACM 1 (8), 3 - 6, 1958.

Findings

- In this article, Ershov proposes the use of "convention numbers" for denoting the results of computations and avoid having to recompute them.
- Ershov's work can be considered the initial work on *value numbering schemes*.
- Rephrased in more modern terms:

  The origin of research on *Code Motion* (*CM*) can be traced back to Ershov's CACM article of

1958!

Conclusions drawn from these Findings

Research on CM-based Program Optimizations...

Its 50th Anniversary
is approaching!

1956/58 – 2006/08

2006

The Year of Anniversaries...

- 1756-2006: 250th Anniversary of the Birth of Mozart
- 1856-2006: 150th Anniversary of the Birth of Freud
- 1956/58-2006: 50th Anniversary of the Origin of Research on Code Motion

Does CM Meet the Criteria?

Indeed, it is an active area of research...

- Relevant ...widely used in practice
- General ...a family of optimizations rather than a single one
- Challenging ...conceptually simple, but exhibits lots of thought-provoking phenomena

Last but not least...

- The underlying theory is a nice piece of mathematics!

Technology for the Impatient .Net Programmer and User

The Plan for this Presentation

- Providing a Tour through 50 Years of Research on CM
- Focusing on
  - Achievements
  - Phenomena
  - Open Problems and Challenges

Tour Stops Included

- Part I: Code motion – Exploring the Design Space
- Part II: Code motion – Classically, but Advanced
- Part III: Code motion – Phenomena of its Derivatives
- Part IV: Code Motion – Recent Strands of Research
- Conclusions and Perspectives

Part I: CM – Exploring the Design Space

CM in the early days essentially meant...

- Common subexpression elimination
- Loop invariant code motion

CM – The Seminal Work

Even if CM can be traced back to...

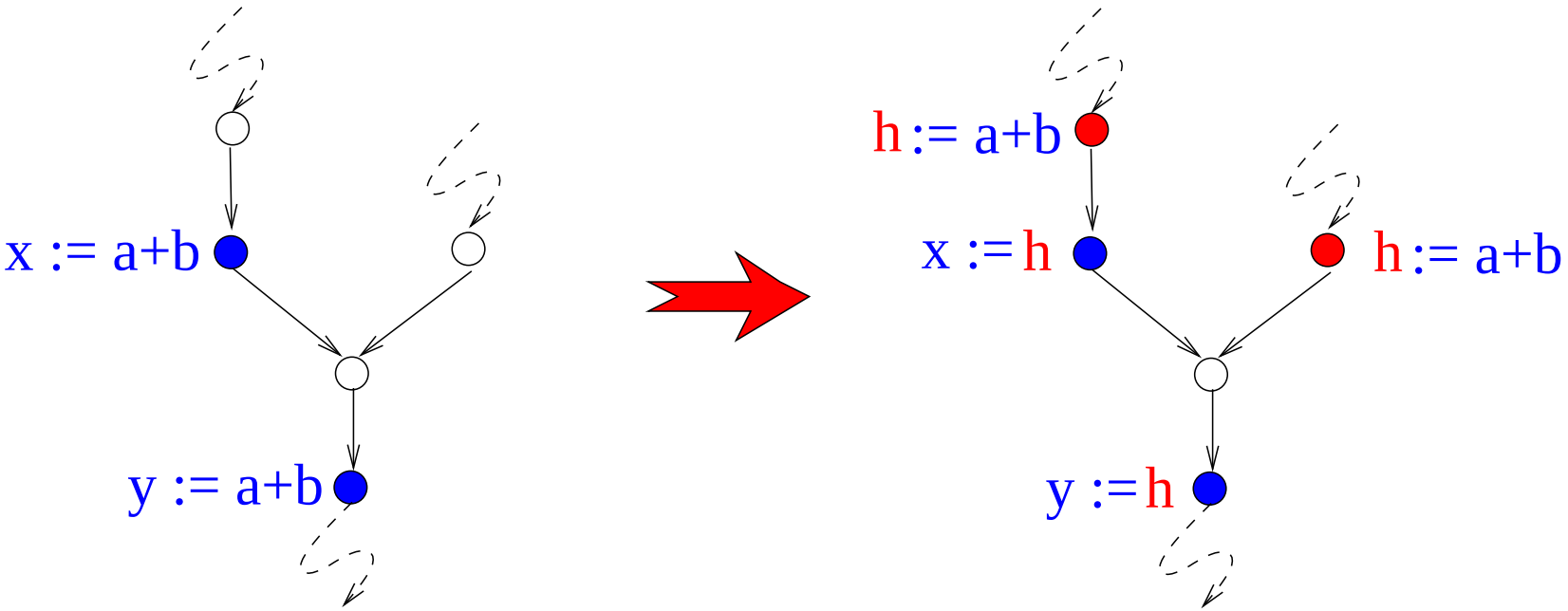
- Ershov, A. P. *On Programming of Arithmetic Operations*. CACM 1 (8), 3 - 6, 1958.

...it is fair to say that contemporary CM starts with the seminal work of

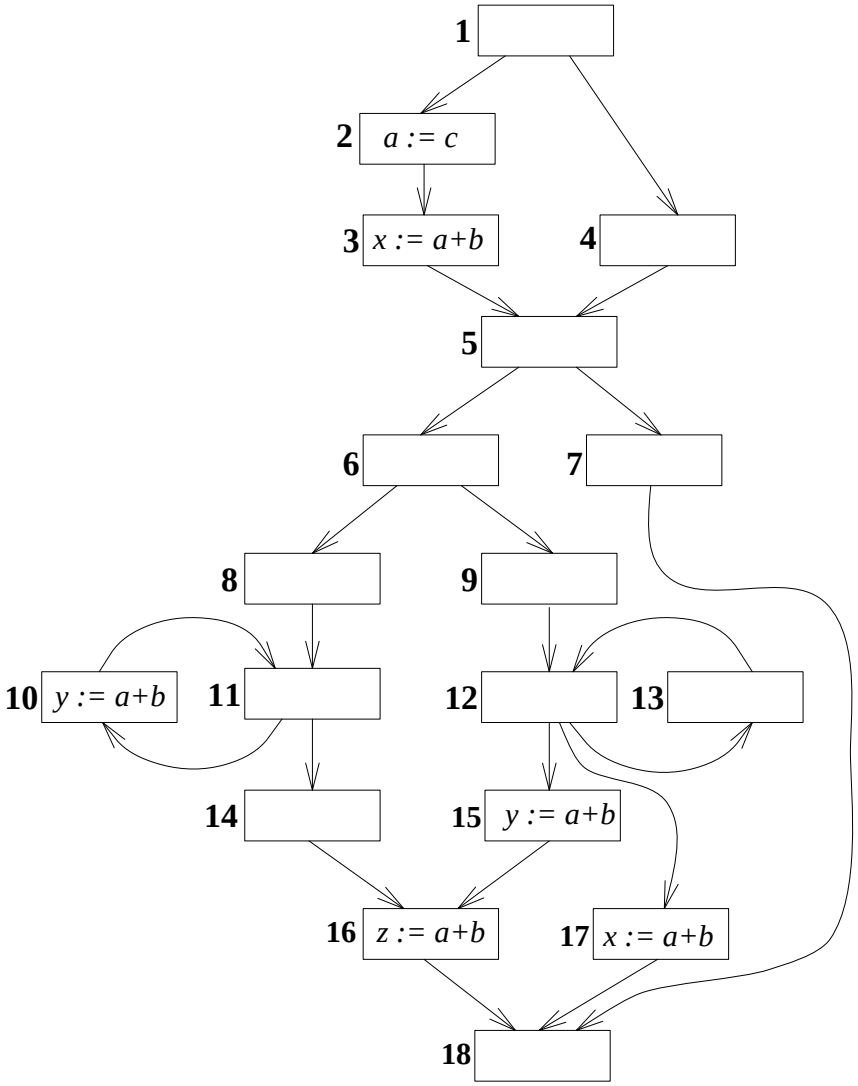
- Morel, E. and Renvoise, C. *Global Optimization by Suppression of Partial Redundancies*. CACM 22 (2), 96 - 103, 1979.

CM – What's it all about?

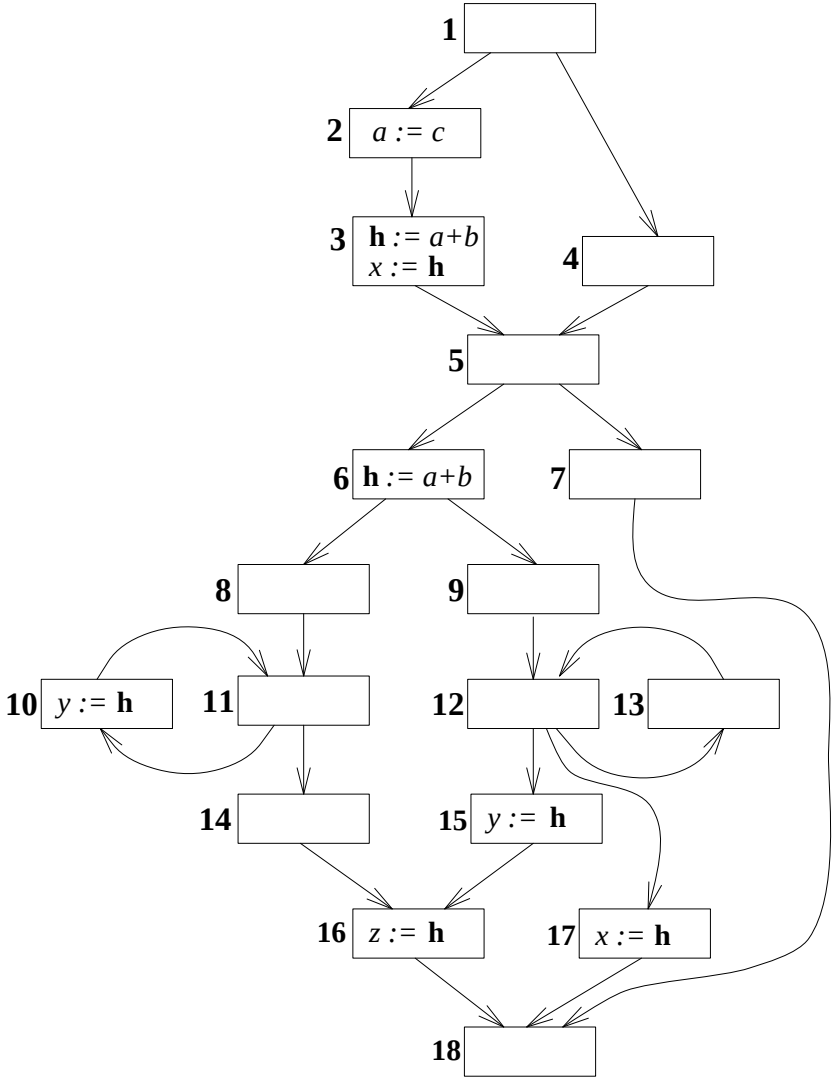
- Essentially, CM aims at avoiding recomputing values



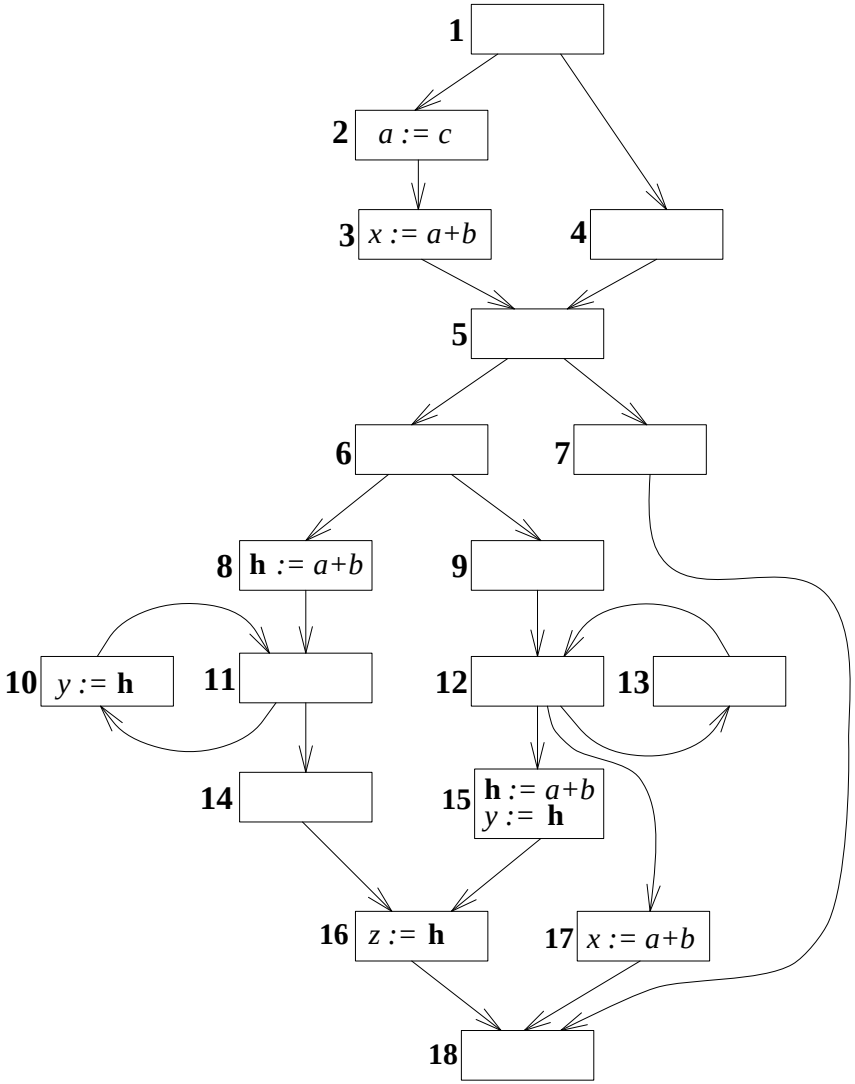
In practice, it is slightly more complex!



A Transformation

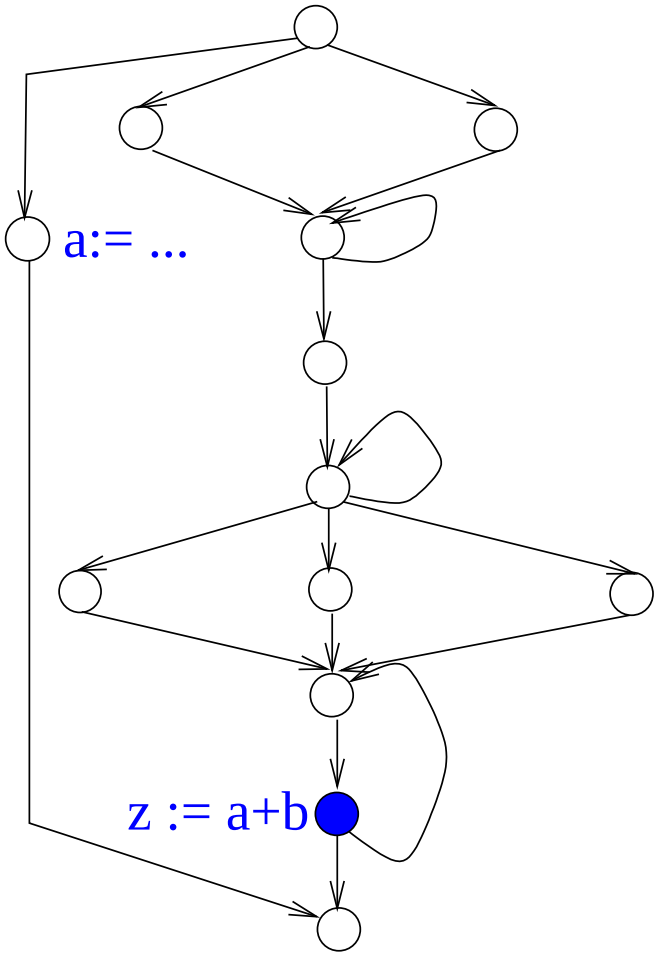


Another Transformation



In more detail

CM and its two traditional optimization goals...



Conceptually

...CM can be considered a two-stage process

1. Expression hoisting

   ...hoisting expressions to "earlier" safe computation points

2. Total redundancy elimination

   ...eliminating computations which became totally redundant

Extreme Strategy – Earliestness Principle

Placing computations as early as possible...

- Theorem [Computational Optimality]

  ...hoisting expressions to their earliest safe computation points yields computationally optimal programs

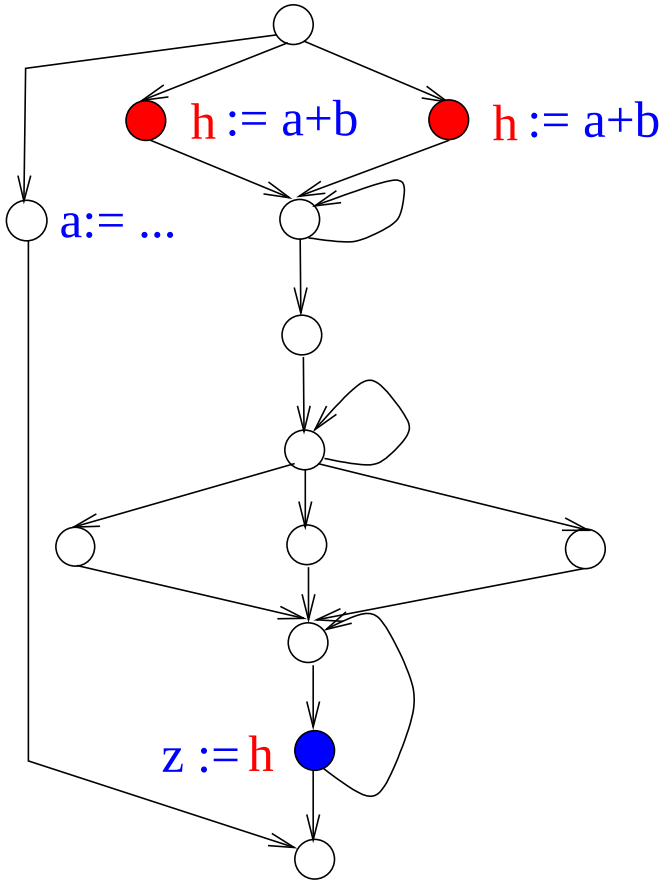
  $\rightsquigarrow$ ...known as Busy Code Motion (PLDI'92, Knoop et al.)

...already known to Morel and Renvoise (though no theorem or proof).

Earliestness Principle

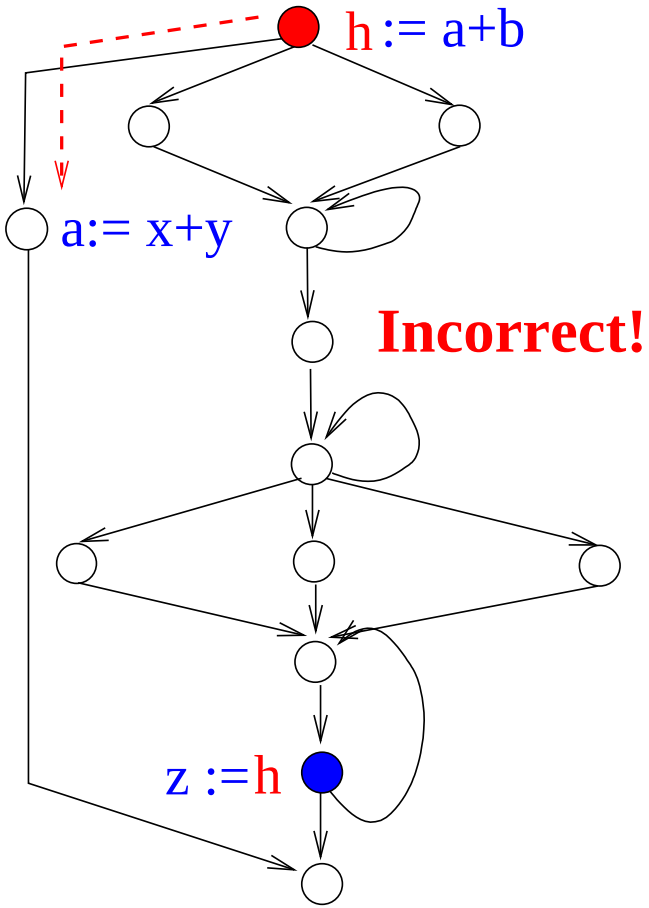
Placing computations as early as possible...

...yields computationally optimal programs.



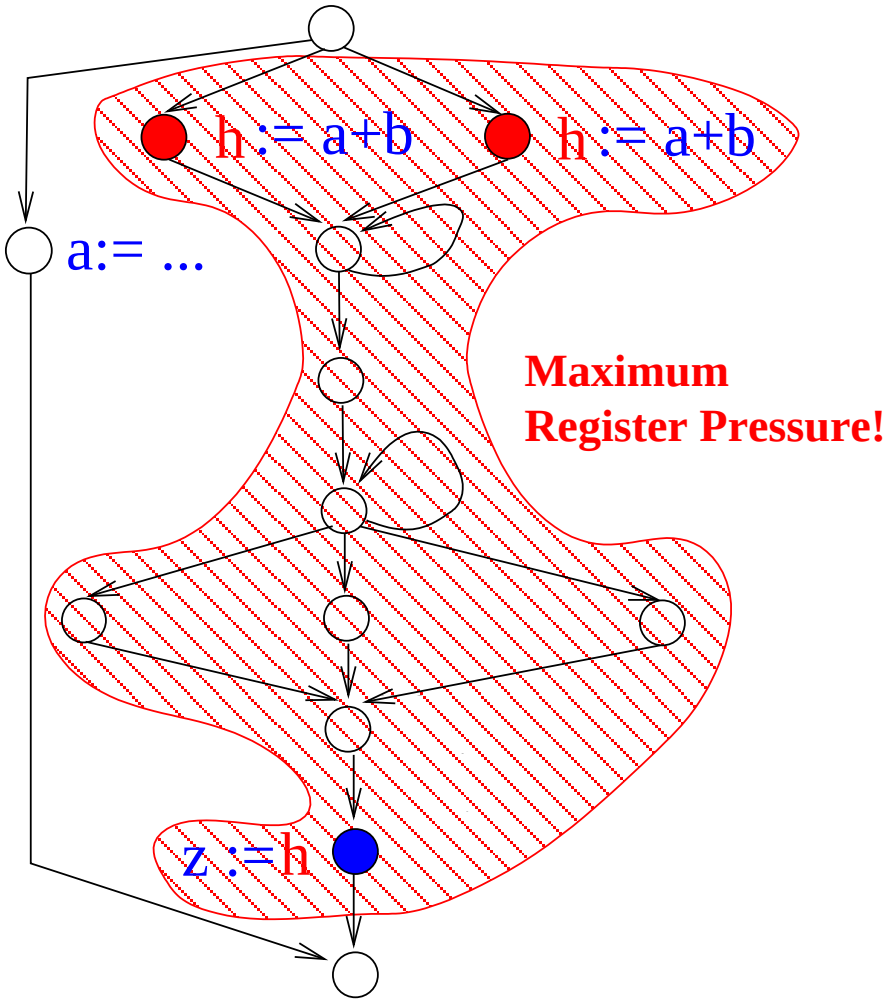
Note: Earliestness means in fact...

...as early as possible, but not earlier!



Earliestness Principle: Important Drawback

...computationally optimal, but maximum register pressure



Dual Extreme Strategy – Latestness Principle

Placing computations as late as possible...

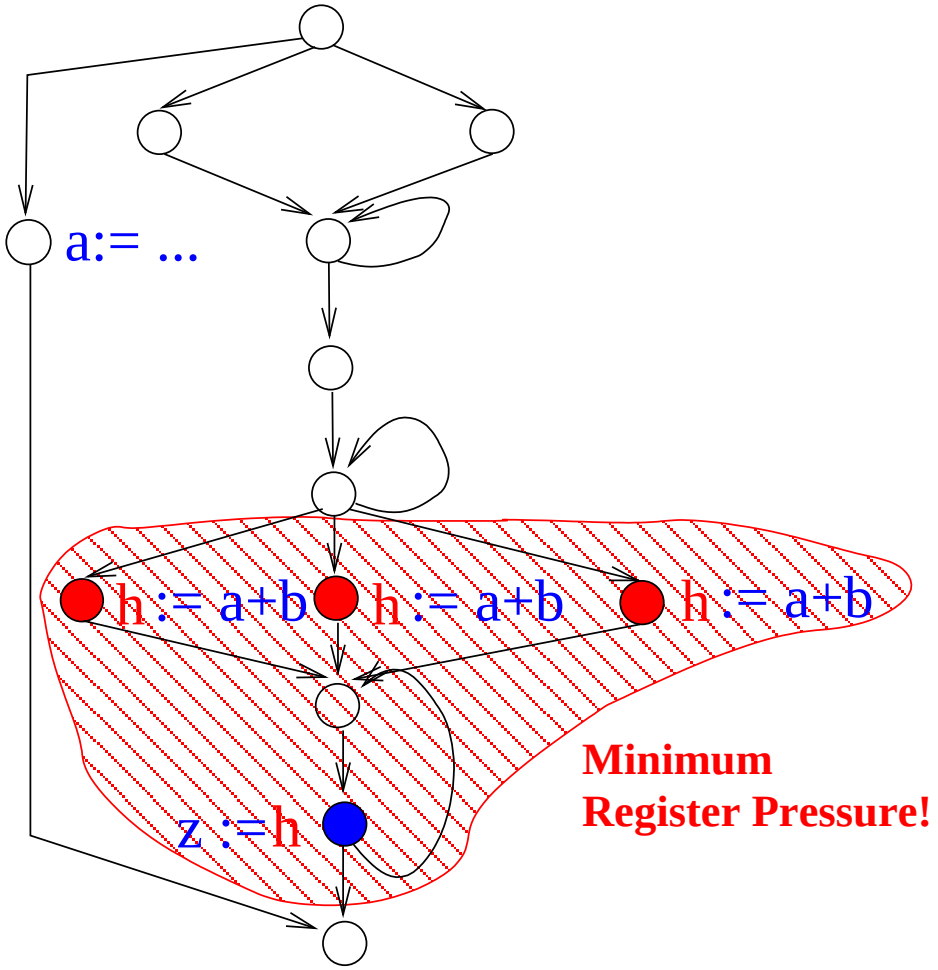
- Theorem [Optimality]

  ...hoisting expressions to their latest safe computation points yields computationally optimal programs with minimum register pressure

  $\rightsquigarrow$ ...known as Lazy Code Motion (PLDI'92, Knoop et al.)

Latestness Principle

...computationally optimal, too, with mininum register pressure!



These days...

Lazy Code Motion is...

- ...the de-facto standard algorithm for CM used in contemporary state-of-the-art compilers
  - Gnu compiler family
  - Sun Sparc compiler family
  - ...

Towards Exploring the Design Space

Traditionally,

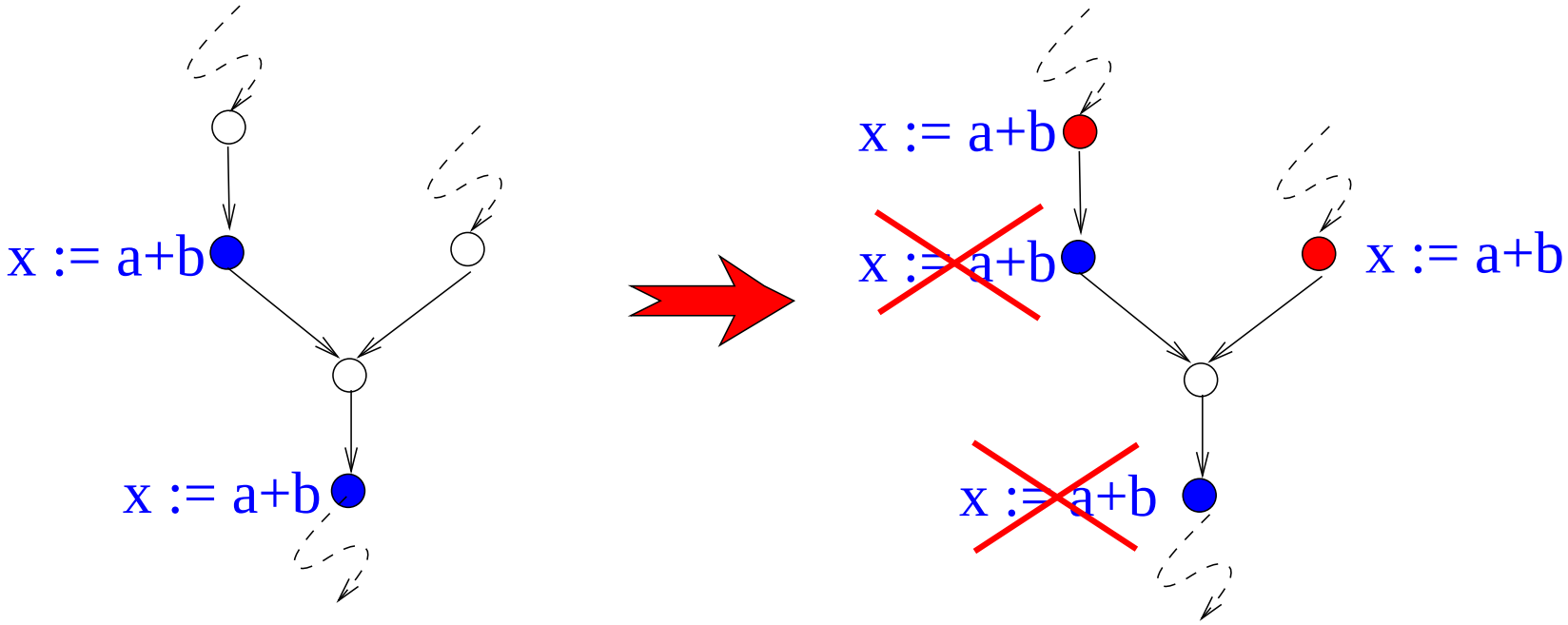
- Code (C) means expressions
- Motion (M) means hoisting

But...

- CM is more than hoisting of expressions and PR(E)E!

Obviously, code...

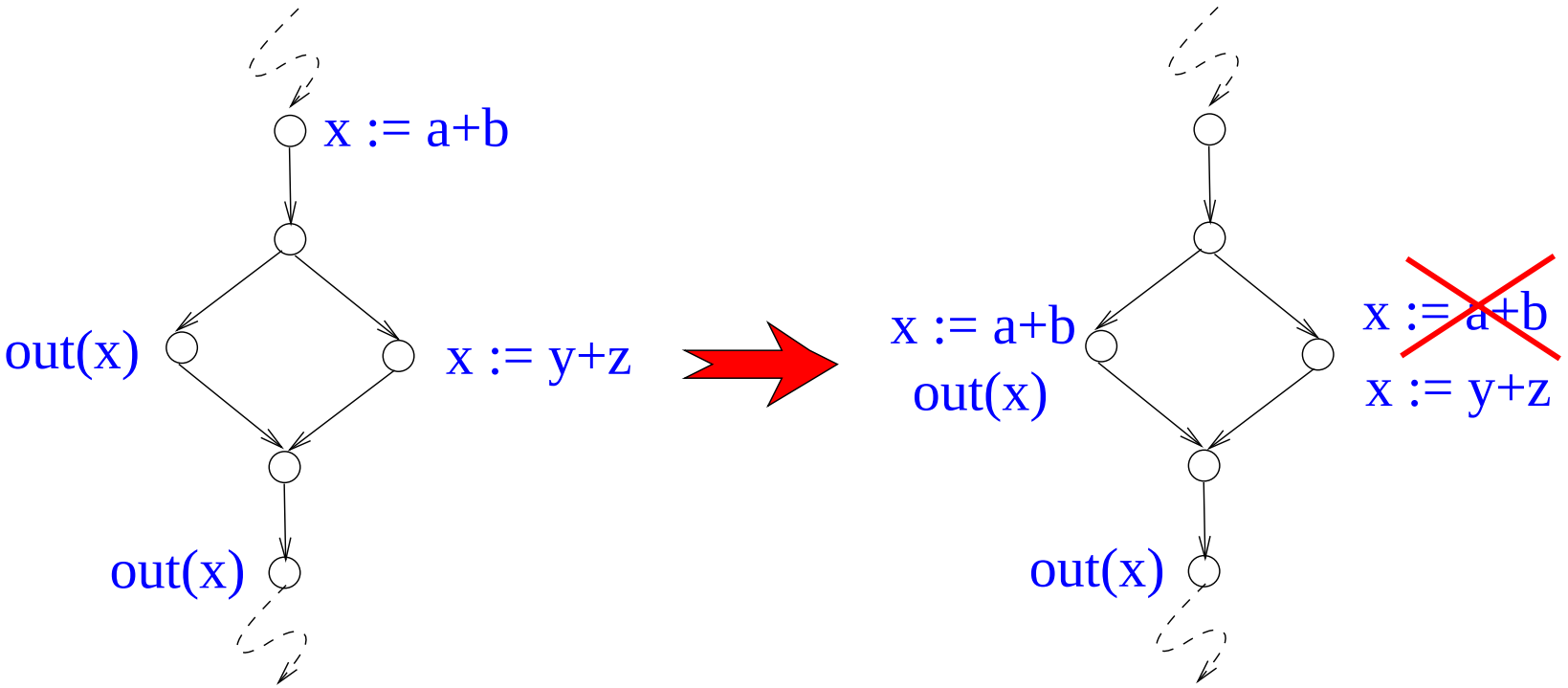
...can be assignments, too.



- Here, CM means partially redundant assignment elimination (PRAE)

In contrast to expressions, assignments...

...might also be sunk.



- Now, CM means partially dead code elimination (PDCE)

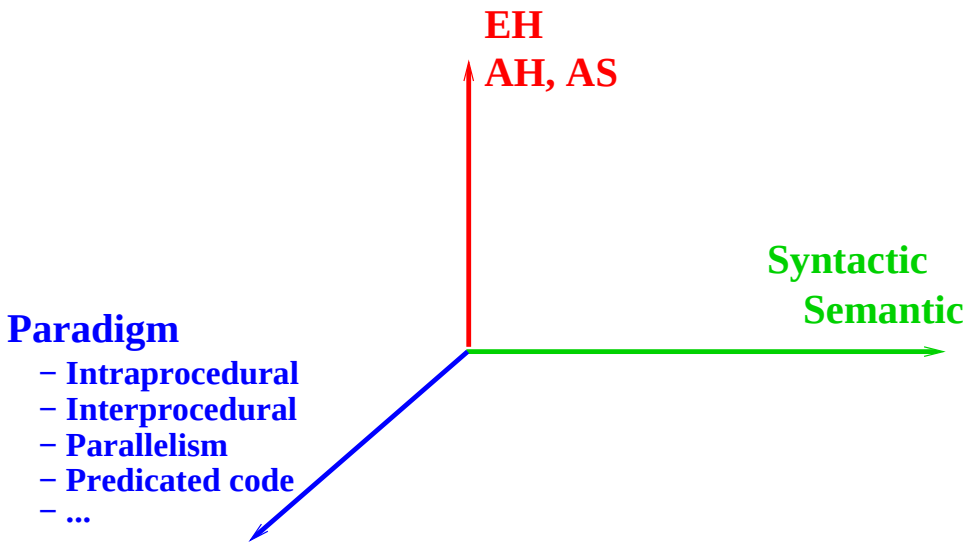
Towards the Design Space of CM-Algorithms...

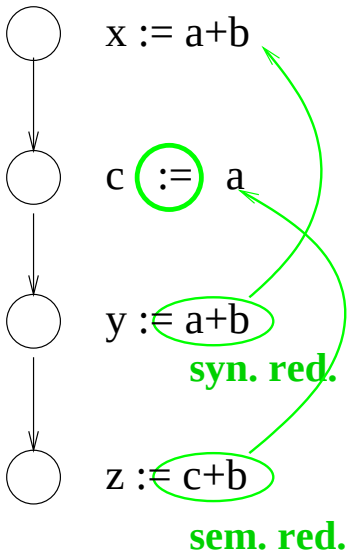
More generally...

- Code means expressions/assignments
- Motion means hoisting/sinking

Code / Motion	Hoisting	Sinking
Expressions	EH	·/·
Assignments	AH	AS

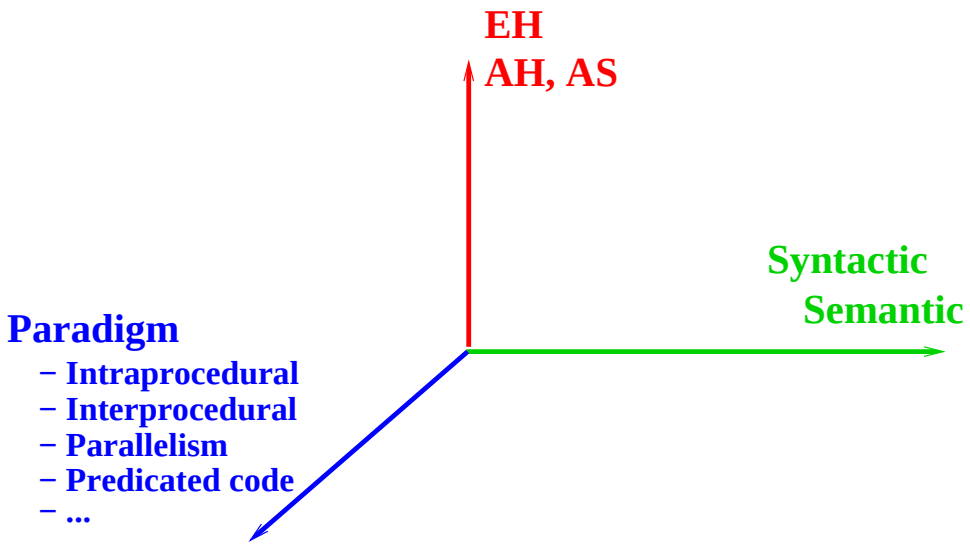
Refining the Design Space of CM-Algorithms...

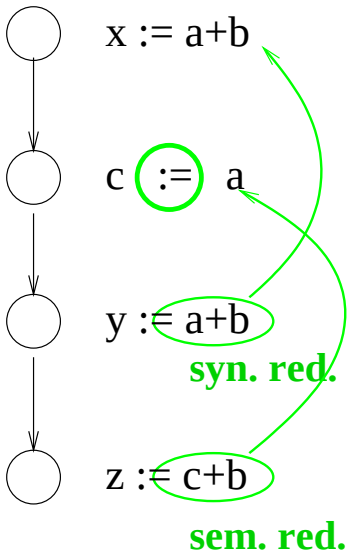




Introducing semantics... !

Refining the Design Space of CM-Algorithms...

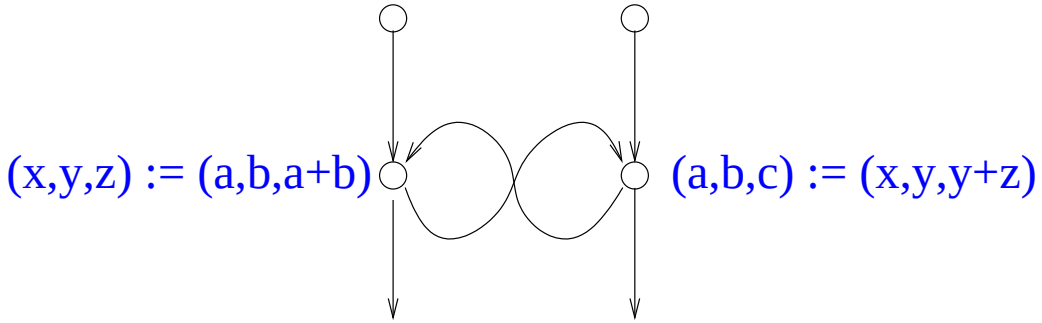


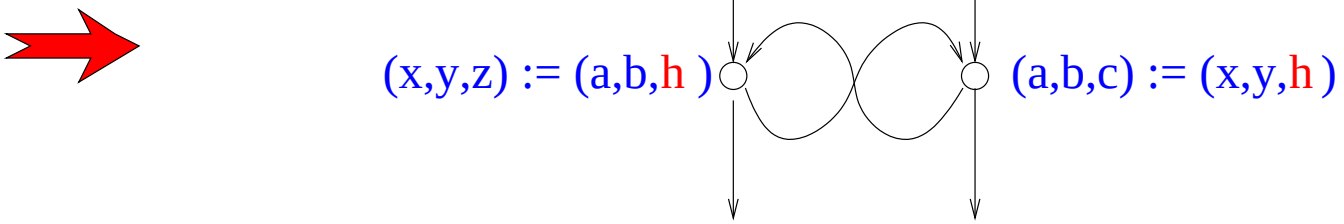


Introducing semantics... !

Semantic Code Motion...

allows more powerful optimizations!





(example by B. Steffen, TAPSOFT'87)

Remember,...

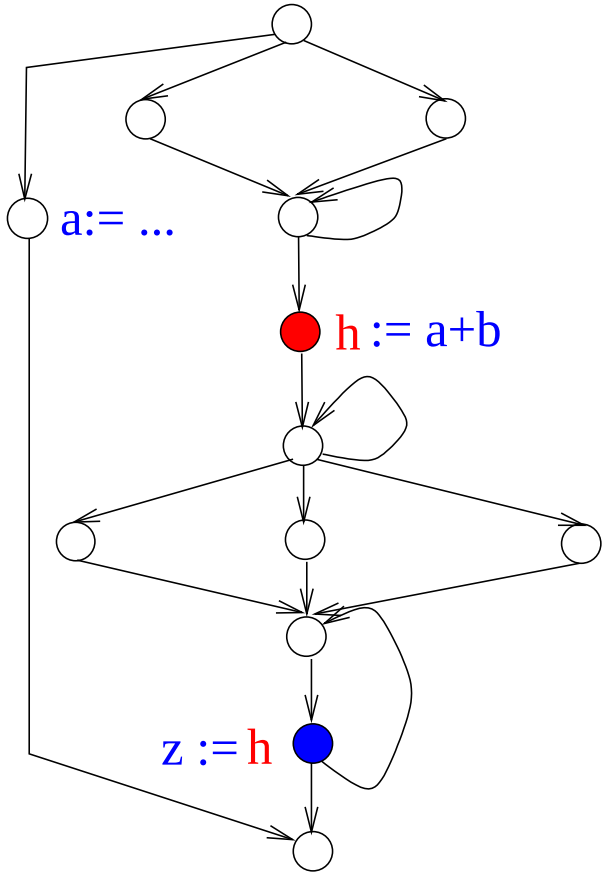
CM (PREE) and its optimization goals!

- Speed
- Register Pressure

There might be a third one:

- Code Size

A Computationally and Code-Size Optimal Program



$\rightsquigarrow$ Code size

1999 World Market for Microprocessors

Going for size makes sense...

Chip Category	Number Sold
Embedded 4-bit	2000 million
Embedded 8-bit	4700 million
Embedded 16-bit	700 million
Embedded 32-bit	400 million
DSP	600 million
Desktop 32/64-bit	150 million

$\sim$ **2%**

... David Tennenhouse (Intel Director of Research). Keynote Speech at the *20th IEEE Real-Time Systems Symposium* (*RTSS'99*), Phoenix AZ, 1999.

Think of...

... domain-specific processors as used in embedded systems

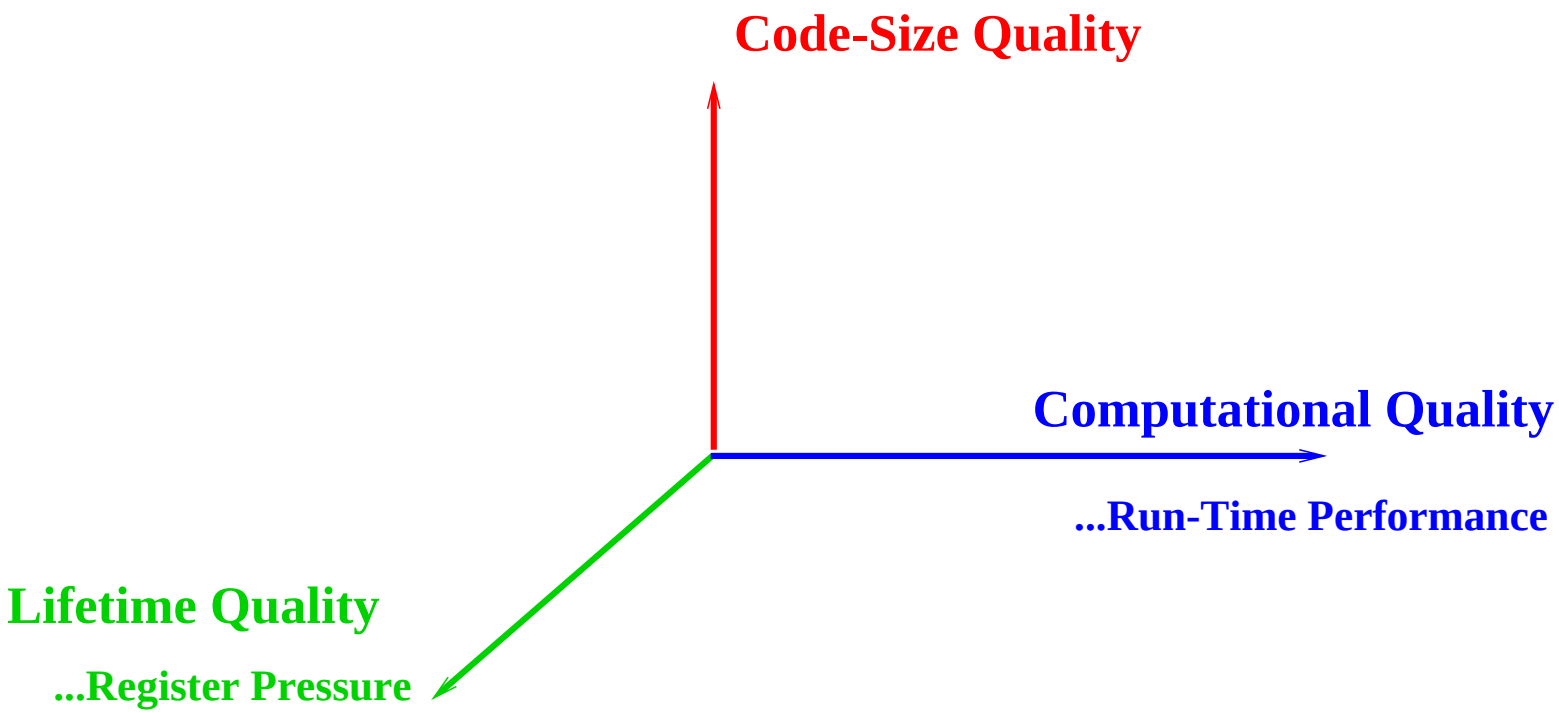
- **Telecom**
  - Cell phones, pagers, ...
- **Consumer Electronics**
  - MP3 player, cameras, pocket games, ...
- **Automative**
  - GPS navigation, airbags, ...
- ...

For such applications...

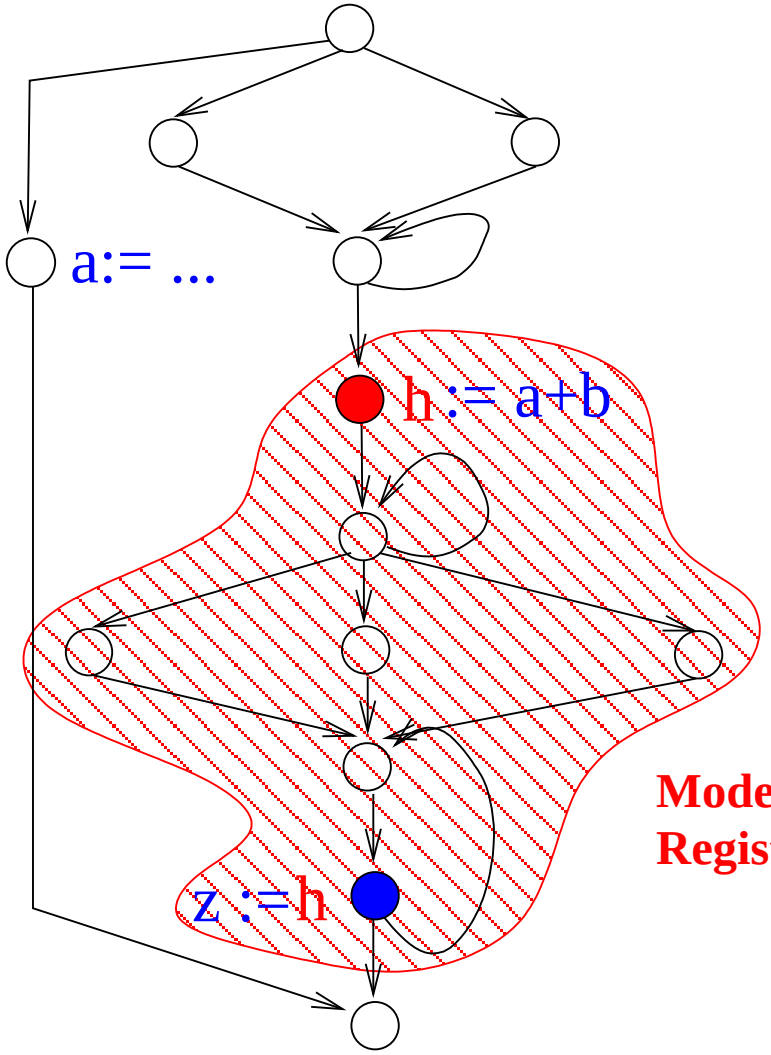
...code size often more critical than speed!

Part II: CM – Classically, but Advanced

...enhancing (L)CM to take a user's priorities into account!



...rendering possible this transformation, too:

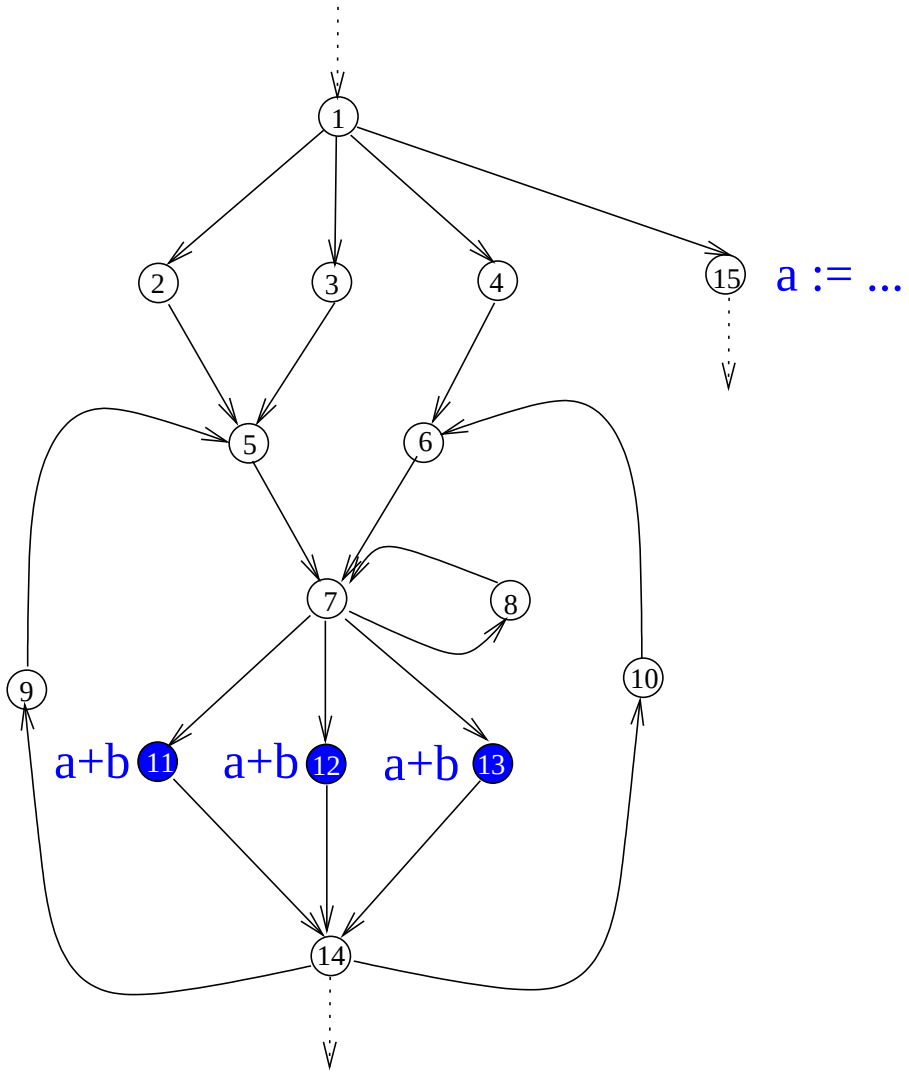


Moderate Register Pressure!

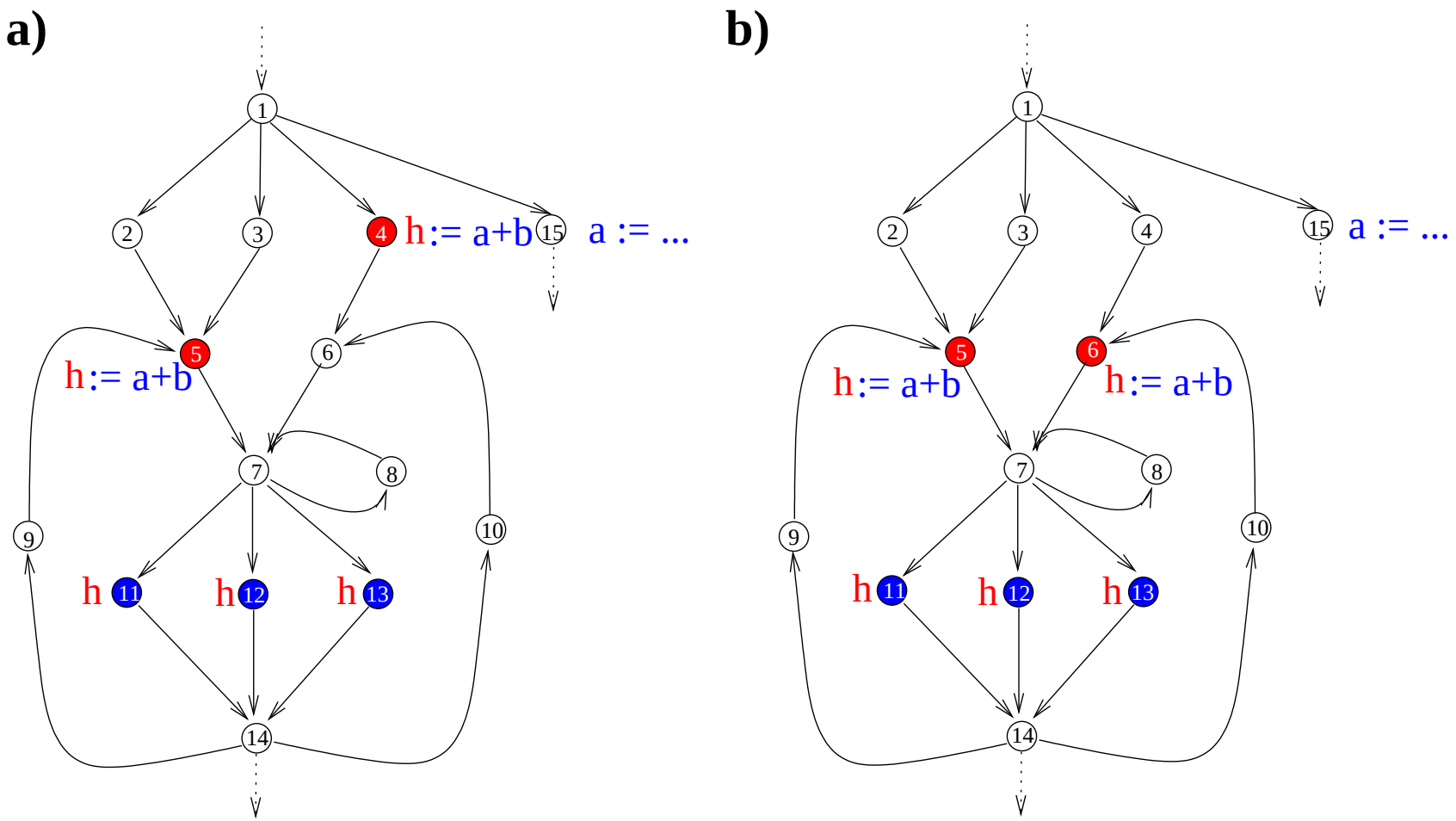
Towards Code-Size Sensitive CM...

- **Background: Classical CM**
  - $\rightsquigarrow$ Busy CM (BCM) / Lazy CM (LCM) (Knoop et al., PLDI'92)
    - Received the *ACM SIGPLAN Most Influential PLDI Paper Award 2002 (for 1992)*
    - Selected for *"20 Years of the ACM SIGPLAN PLDI: A Selection"* (60 papers out of ca. 600 papers)
- **Code-Size Sensitive CM** (Knoop et al., POPL'00)
  - $\rightsquigarrow$ ...modular extension of BCM/LCM
    - Modelling and Solving the Problem
      ...based on graph-theoretical means
    - Main Results
      ...**correctness, optimality**

The Running Example



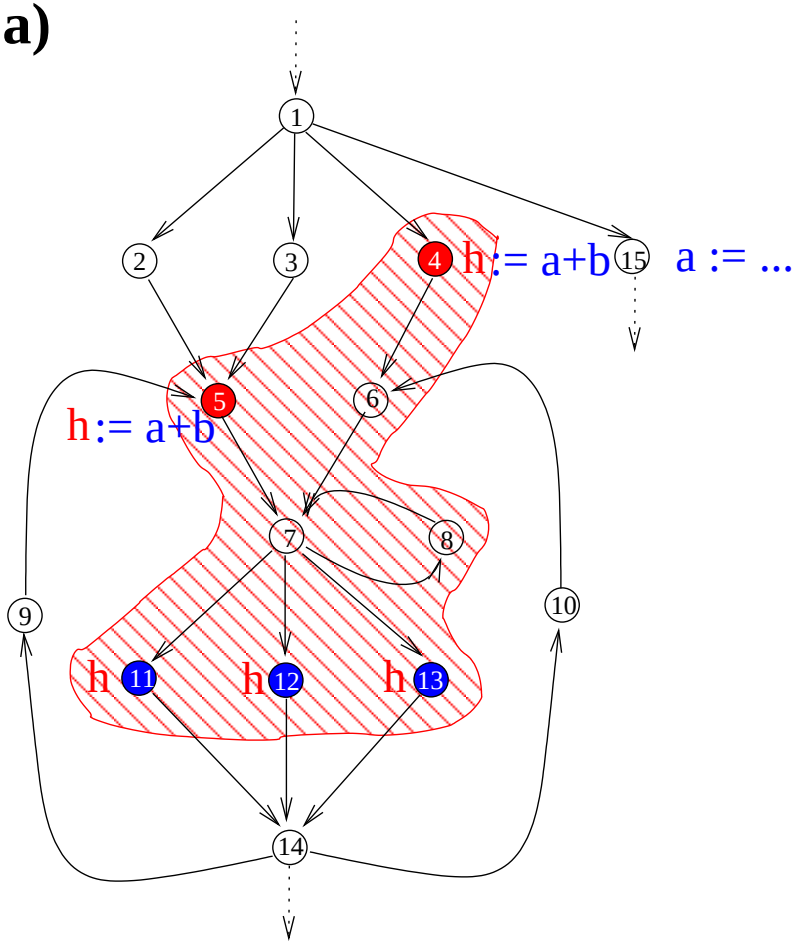
The Running Example (Cont'd)



Two Code–size Optimal Programs

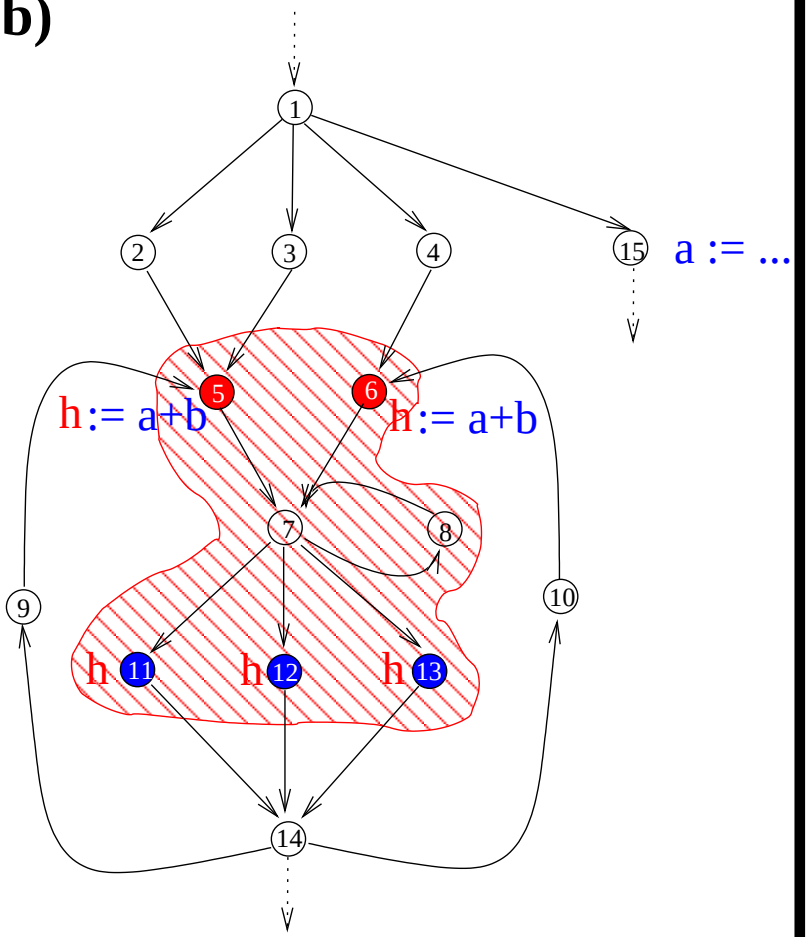
The Running Example (Cont'd)

a)



SQ > CQ > LQ

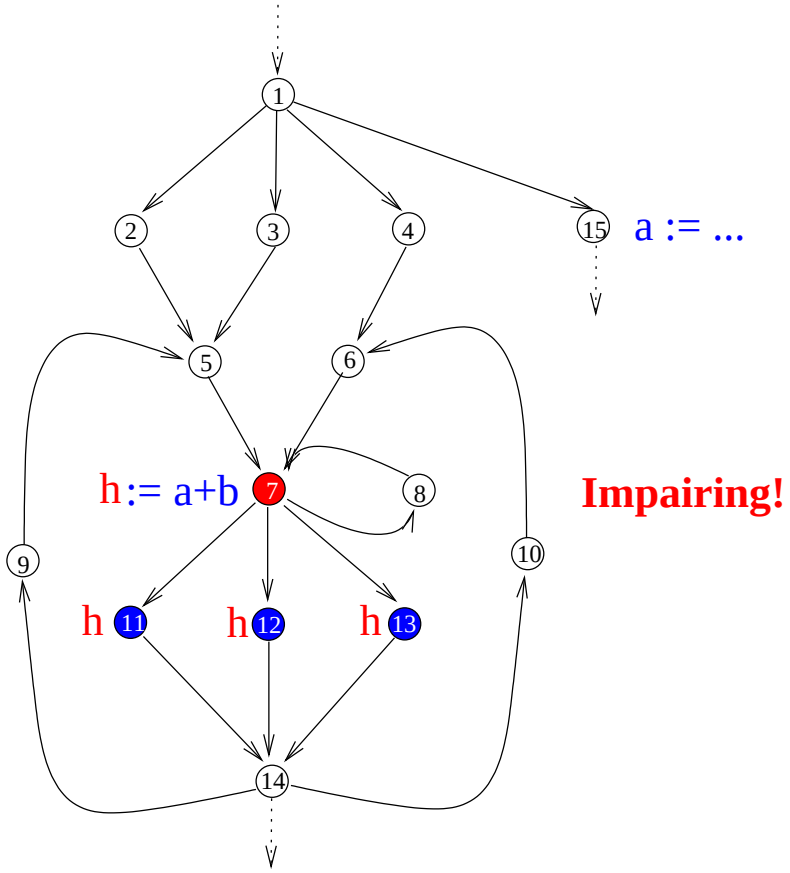
b)



SQ > LQ > CQ

The Running Example (Cont'd)

Note, we do not want the following transformation: It's no option!



Code-Size Sensitive PRE

⇝ The Problem

...how to get a code-size minimal placement of computations, i.e., a placement which is

- admissible (semantics & performance preserving)
- code-size minimal

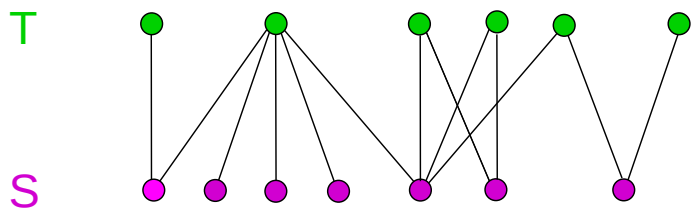
⇝ Solution: A Fresh Look at PRE

...considering PRE a trade-off problem: trading the original computations against newly inserted ones!

⇝ The Clou: Use Graph Theory!

...reducing the trade-off problem to the computation of tight sets in bipartite graphs based on maximum matchings!

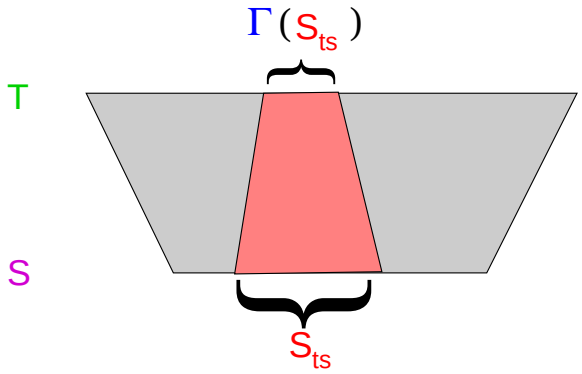
Bipartite Graph



Tight Set

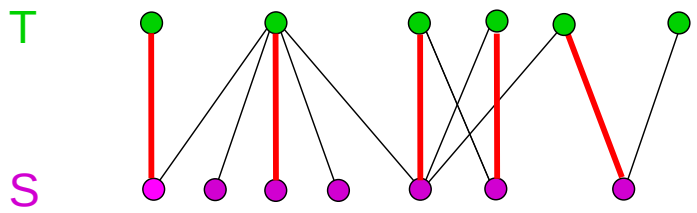
...of a bipartite graph $(S \cup T, E)$ is a subset $S_{ts} \subseteq S$ such that

$$\forall\, S' \subseteq S.\ |S_{ts}| - |\Gamma(S_{ts})| \ \geq\ |S'| - |\Gamma(S')|$$



2 Variants: (1) **Largest** Tight Sets (2) **Smallest** Tight Sets

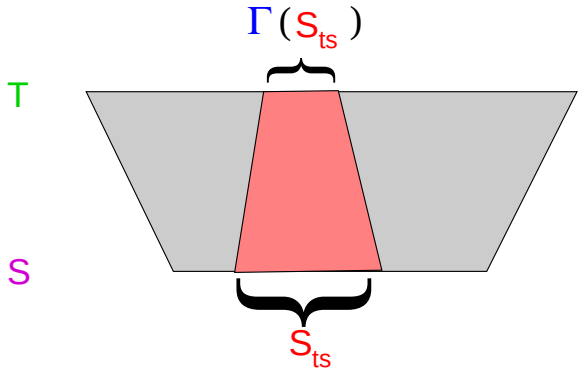
Bipartite Graph



Tight Set

...of a bipartite graph $(S \cup T, E)$ is a subset $S_{ts} \subseteq S$ such that

$$\forall\, S' \subseteq S.\ |S_{ts}| - |\Gamma(S_{ts})| \ \geq\ |S'| - |\Gamma(S')|$$



2 Variants: (1) **Largest** Tight Sets (2) **Smallest** Tight Sets

Apparently

Off-the-shelf algorithms of graph theory can be used to compute...

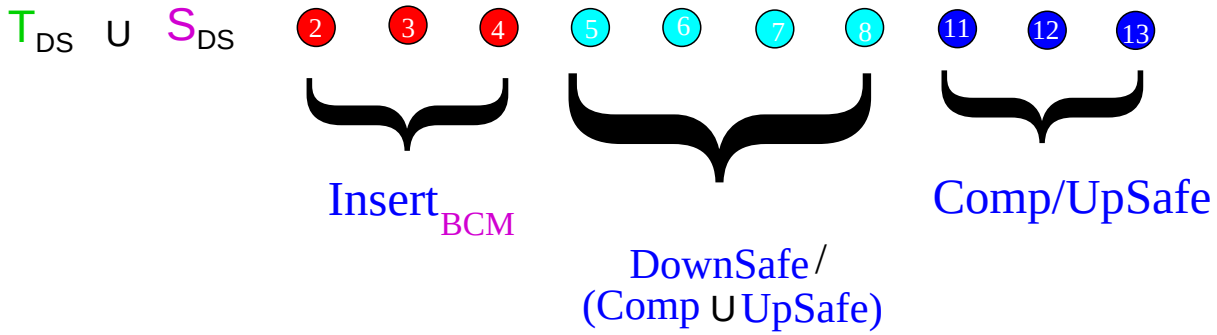
- **Maximum matchings** and
- **Tight sets**

Hence, our PRE problem boils down to...

...constructing the bipartite graph modelling the problem!

Modelling the Trade-Off Problem

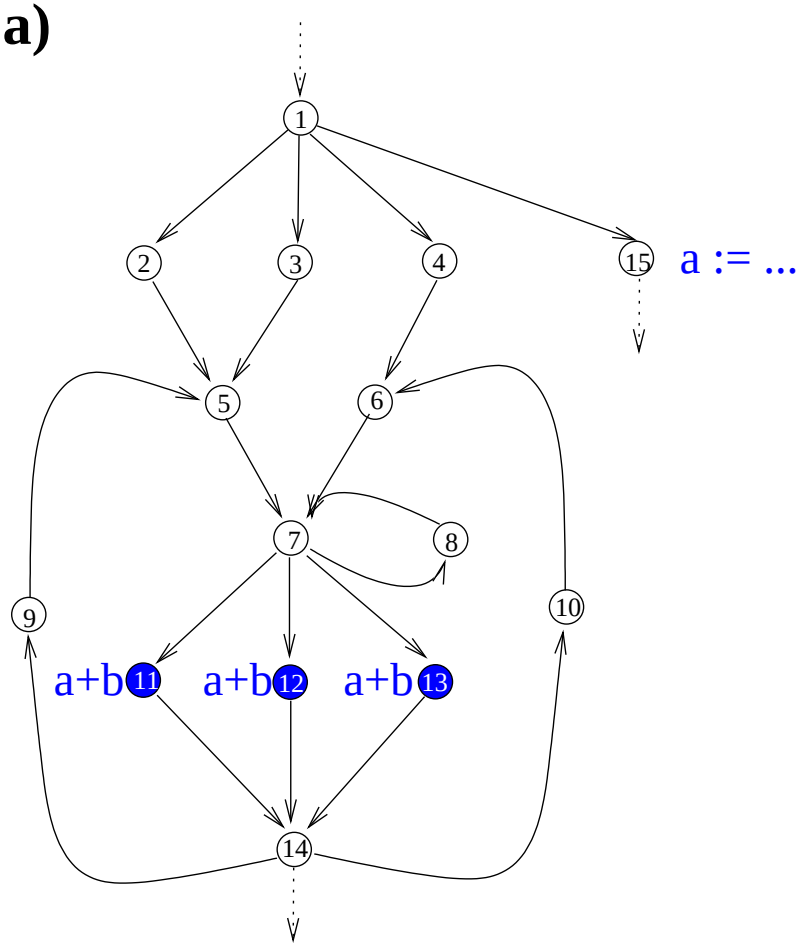
The Set of Nodes



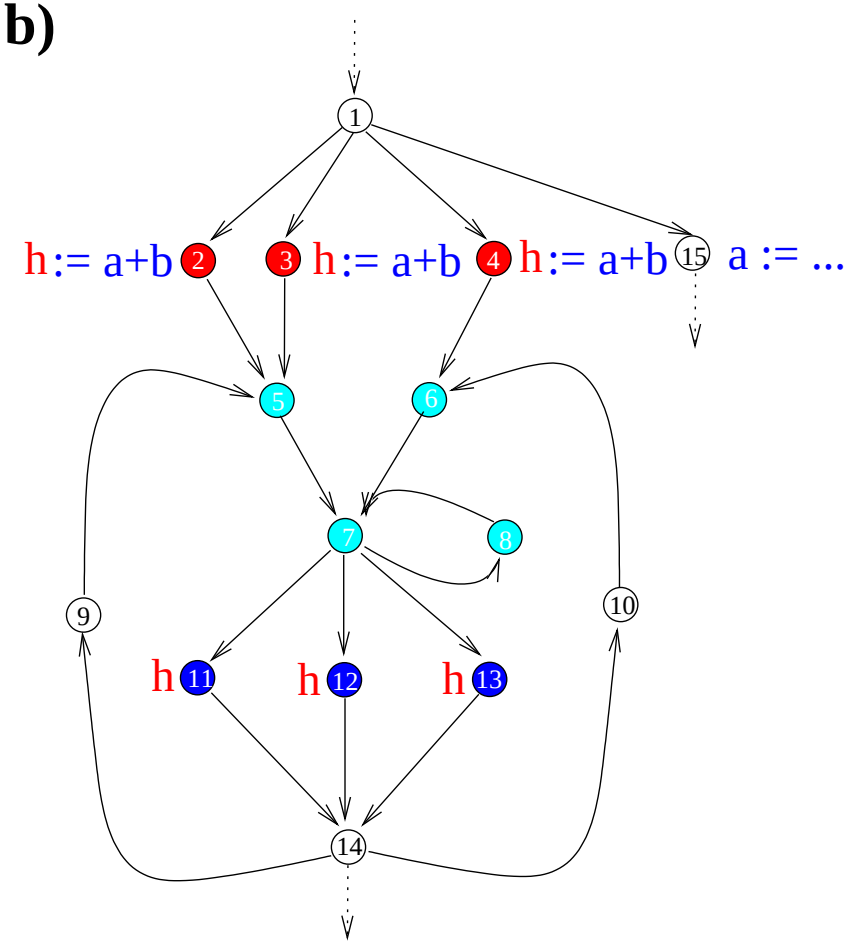
The Set of Edges...

The Set of Nodes

a)

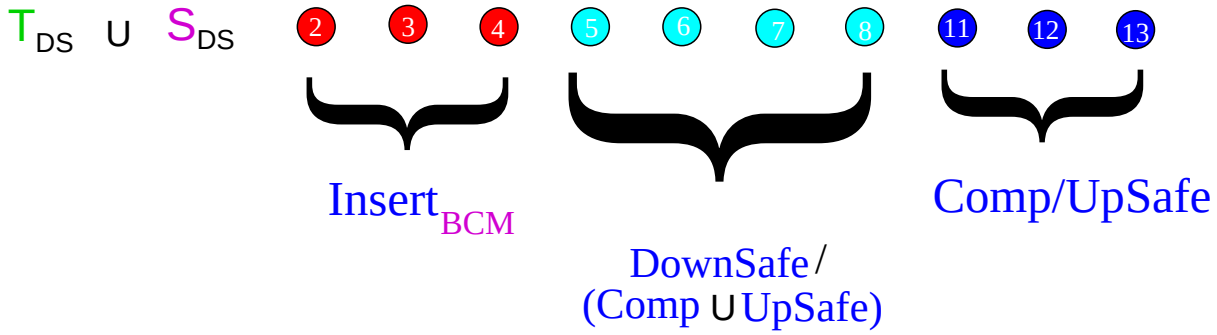


b)

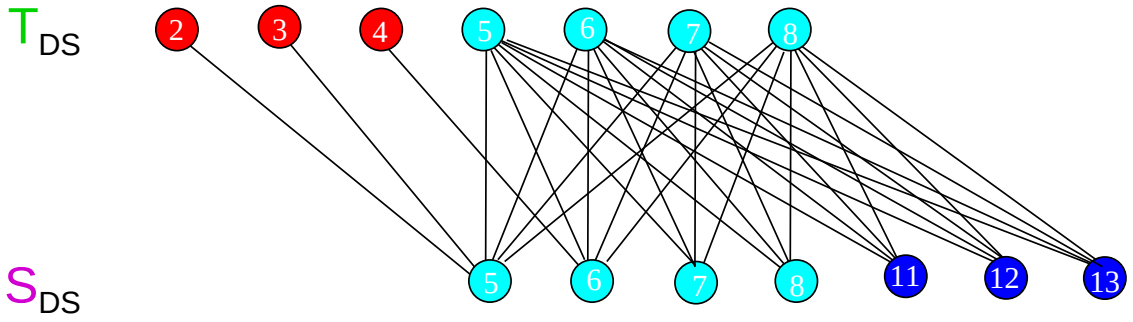


Modelling the Trade-Off Problem

The Set of Nodes



The Bipartite Graph



The Set of Edges ... $\forall n \in S_{DS} \, \forall m \in T_{DS}.$

$$\{n, m\} \in E_{DS} \iff_{df} m \in \textbf{Closure}(pred(n))$$

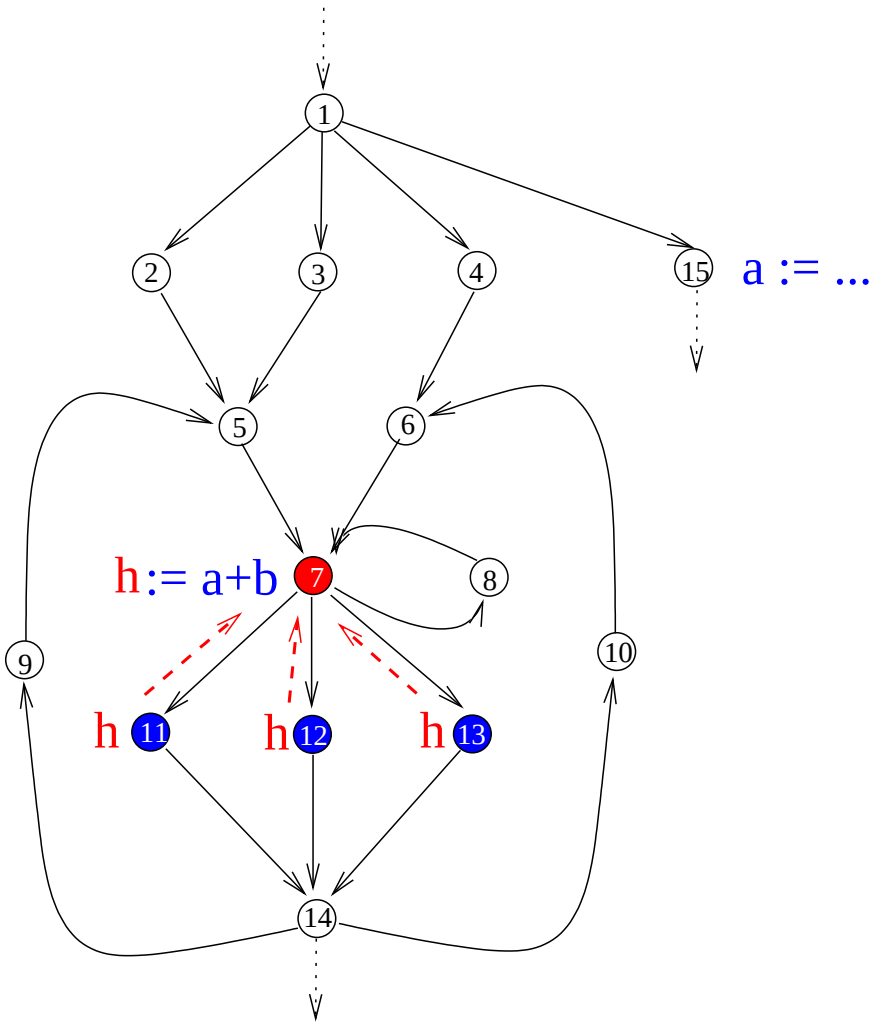
DownSafety Closures

DownSafety Closure

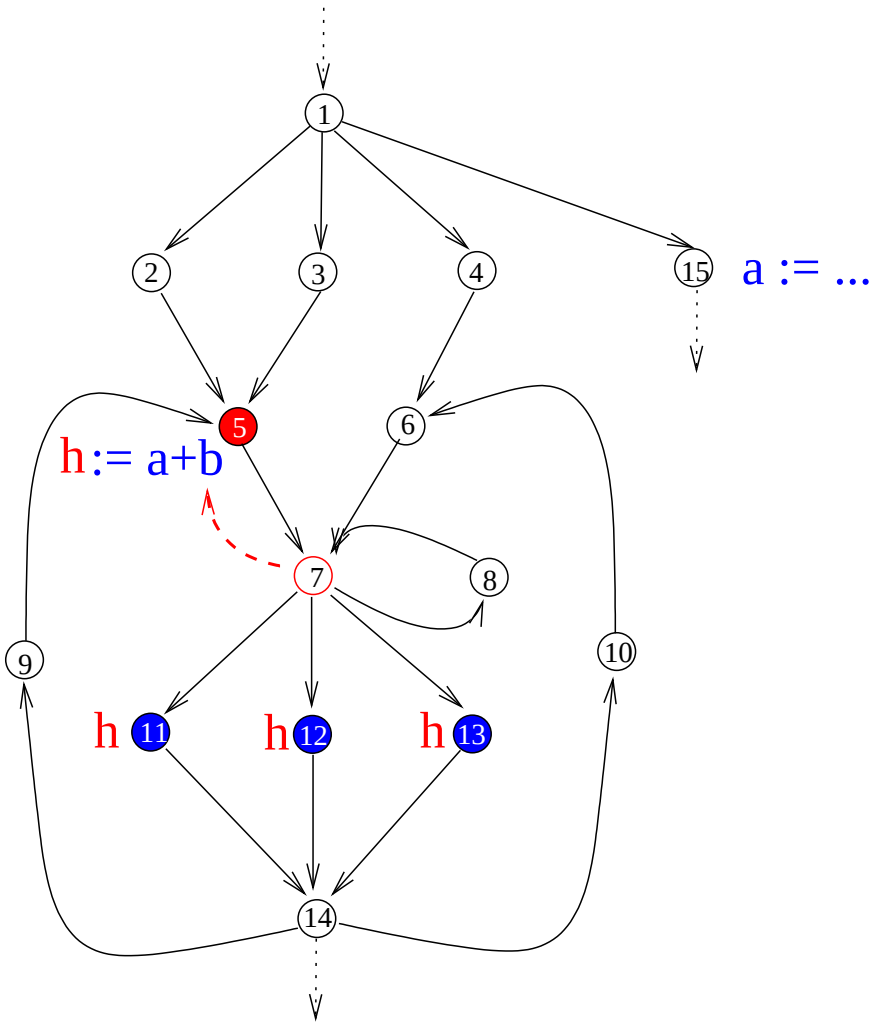
For $n \in$ *DownSafe/Upsafe* the DownSafety Closure $Closure(n)$ is the smallest set of nodes satisfying

1. $n \in Closure(n)$
2. $\forall\, m \in Closure(n) \setminus \mathit{Comp}.\ succ(m) \subseteq Closure(n)$
3. $\forall\, m \in Closure(n).\ pred(m) \cap Closure(n) \neq \emptyset \ \Rightarrow$
   $pred(m) \setminus \mathit{UpSafe} \subseteq Closure(n)$

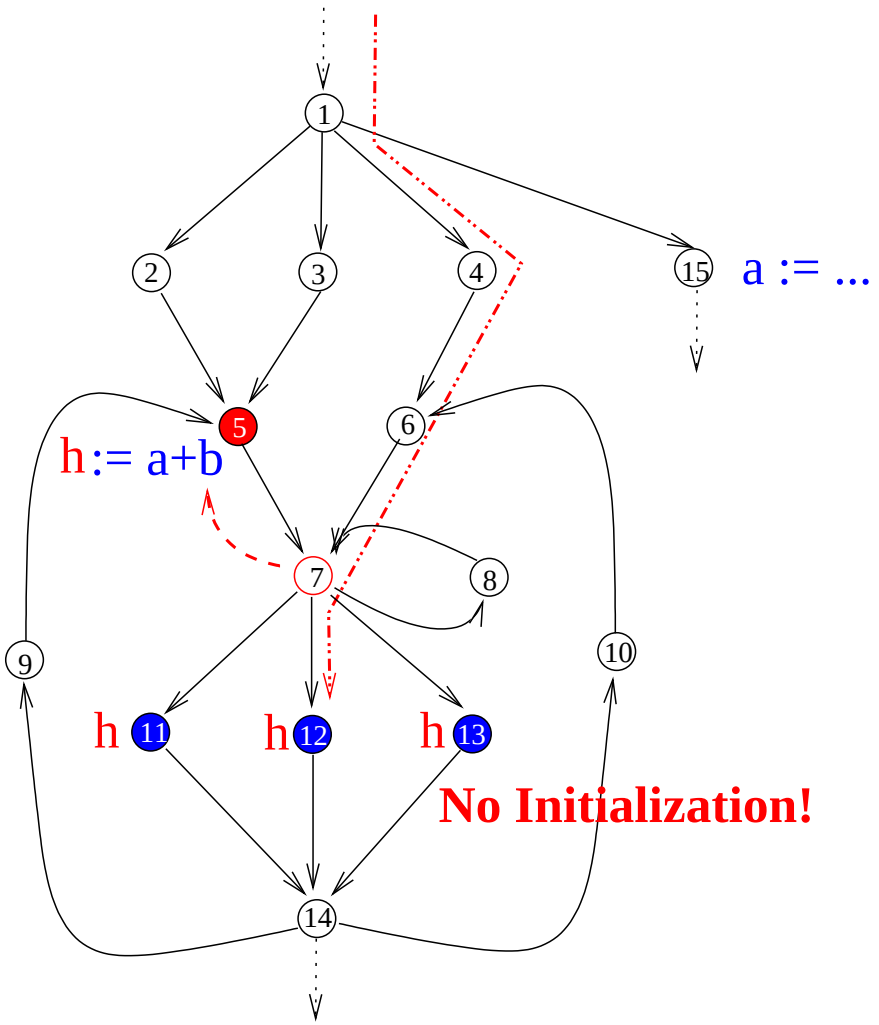
DownSafety Closures – The Very Idea 1(4)



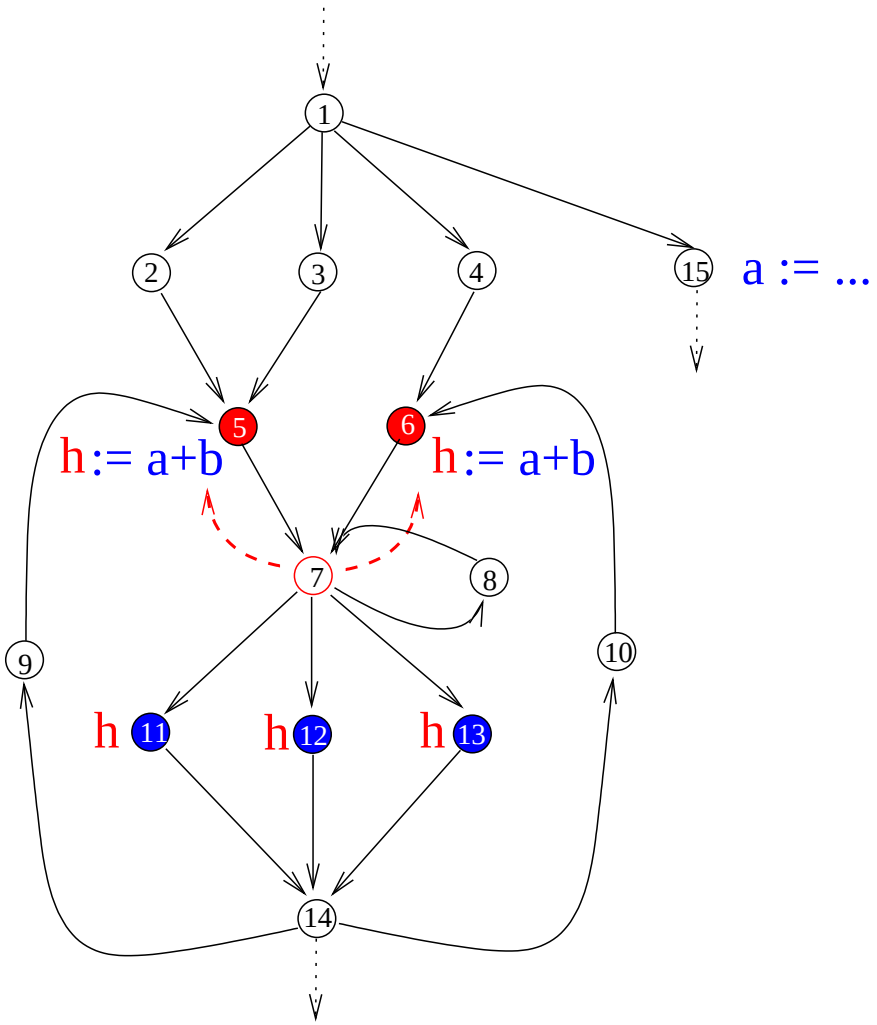
DownSafety Closures – The Very Idea 2(4)



DownSafety Closures – The Very Idea 3(4)



DownSafety Closures – The Very Idea 4(4)



DownSafety Closures

DownSafety Closure

For $n \in$ *DownSafe/Upsafe* the DownSafety Closure $Closure(n)$ is the smallest set of nodes satisfying

1. $n \in Closure(n)$
2. $\forall\, m \in Closure(n) \setminus \mathit{Comp}.\ succ(m) \subseteq Closure(n)$
3. $\forall\, m \in Closure(n).\ pred(m) \cap Closure(n) \neq \emptyset \ \Rightarrow$
   $pred(m) \setminus \mathit{UpSafe} \subseteq Closure(n)$

DownSafety Regions

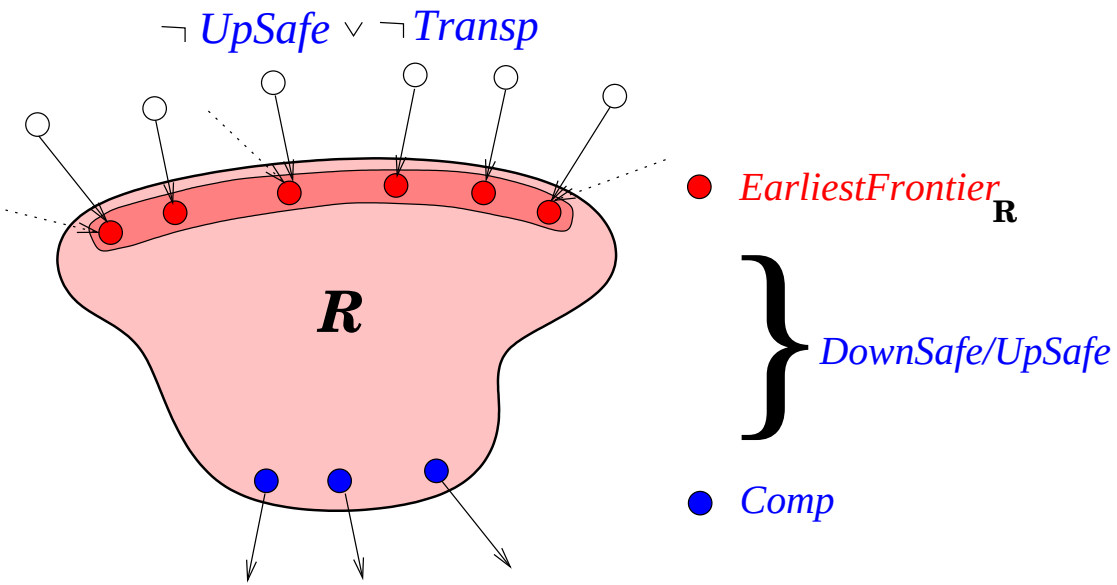
Some subsets of nodes are distinguished. We call each of these sets a DownSafety Region...

- A set $\mathcal{R} \subseteq N$ of nodes is a DownSafety Region if and only if
  1. $\mathit{Comp} \backslash \mathit{UpSafe} \subseteq \mathcal{R} \subseteq \mathit{DownSafe} \backslash \mathit{UpSafe}$
  2. $\mathit{Closure}(\mathcal{R}) = \mathcal{R}$

Fundamental...

Insertion Theorem

Insertions of admissible PRE-Transformations are always at **"earliest-frontiers"** of **DownSafety regions**.



...characterizes for the first time all semantics preserving CM-transf.

The Key Questions

...concerning correctness and optimality:

1. Where to insert computations, why is it correct?
2. What is the impact on the code size?
3. Why is it optimal, i.e., code-size minimal?

...three theorems answering one of these questions each.

Main Results / First Question

1. Where to insert computations, why is it correct?

Intuitively, at the earliestness frontier of the DS-region induced by the tight set...

Theorem 1 [Tight Sets: Insertion Points]

Let $TS \subseteq S_{DS}$ be a tight set.

Then $\mathcal{R}_{TS} =_{df} \Gamma(TS) \cup (Comp \backslash UpSafe)$

is a DownSafety Region with $Body_{\mathcal{R}_{TS}} = TS$

Correctness

...immediate corollary of Theorem 1 and Insertion Theorem

Main Results / Second Question

2. What is the impact on the code size?

Intuitively, the difference between computations inserted and replaced...

Theorem 2 [DownSafety Regions: Space Gain]

Let $\mathcal{R}$ be a DownSafety Region

with $Body_{\mathcal{R}} =_{df} \mathcal{R} \setminus EarliestFrontier_{\mathcal{R}}$

Then

- **Space Gain of Inserting at EarliestFrontier$_{\mathcal{R}}$:**
  $|Comp \setminus UpSafe| - |EarliestFrontier_{\mathcal{R}}| =$
  $|Body_{\mathcal{R}}| - |\Gamma(Body_{\mathcal{R}})| \quad_{df} = defic(Body_{\mathcal{R}})$

Main Results / Third Question

3. Why is it optimal, i.e., code-size minimal?

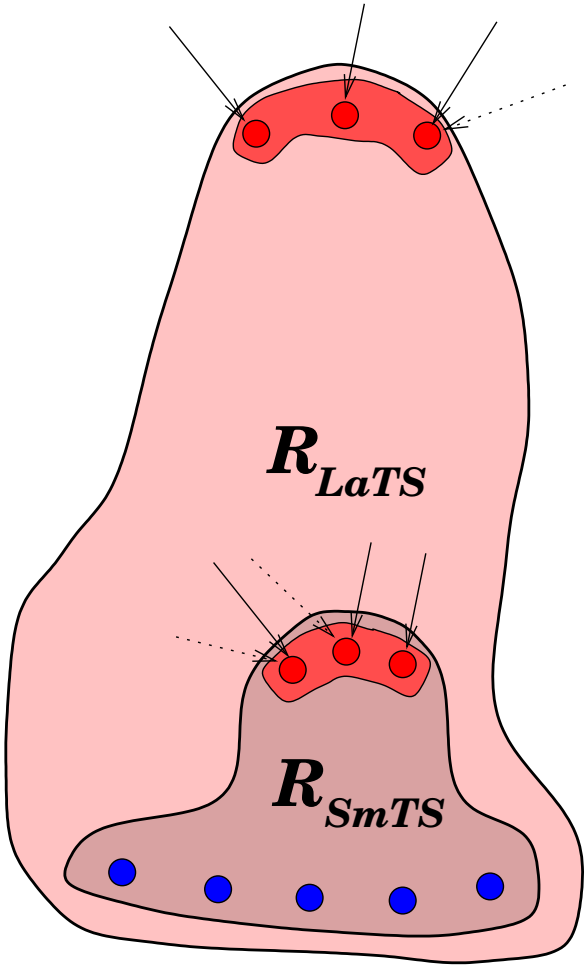
Due to an inherent property of tight sets (*non-negative deficiency!*)...

Optimality Theorem [The Transformation]

Let $TS \subseteq S_{DS}$ be a tight set.

- **Insertion Points:**
  $\mathit{Insert}_{SpCM} =_{df} \mathit{EarliestFrontier}_{\mathcal{R}_{TS}} = \mathcal{R}_{TS} \setminus TS$
- **Space Gain:**
  $\mathit{defic}(TS) =_{df} |TS| - |\Gamma(TS)| \geq 0$ max.

Largest vs. Smallest Tight Sets: The Impact



- *EarliestFrontier*$_{R_{LaTS}}$

Largest tight sets favor

Computational Quality

⇒ **Earliestness Principle**

- *EarliestFrontier*$_{R_{SmTS}}$

Smallest tight sets favor

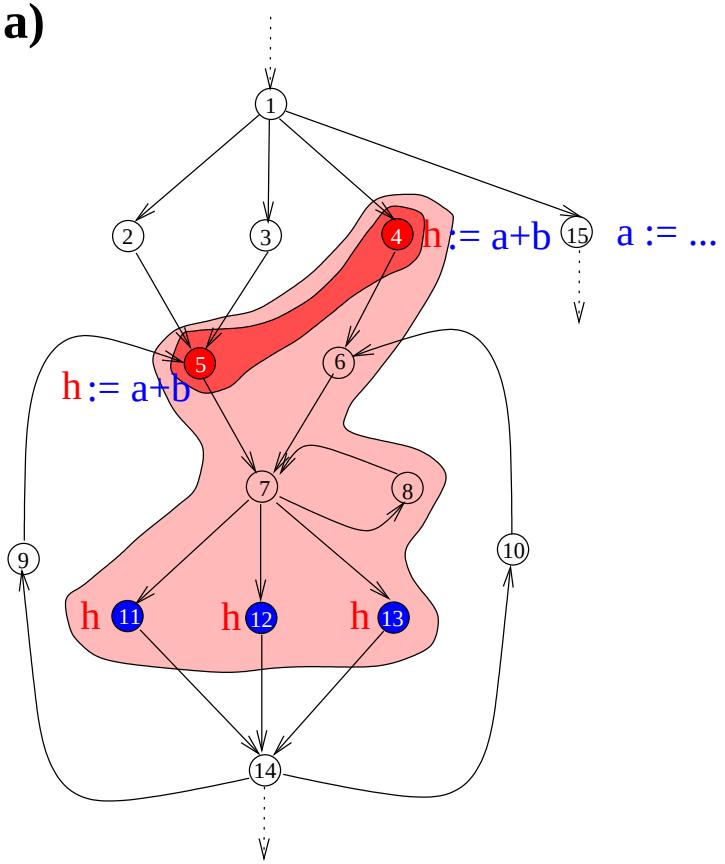
Lifetime Quality

⇒ **Latestness Principle**

- *Comp*

Recall the Running Example

a)

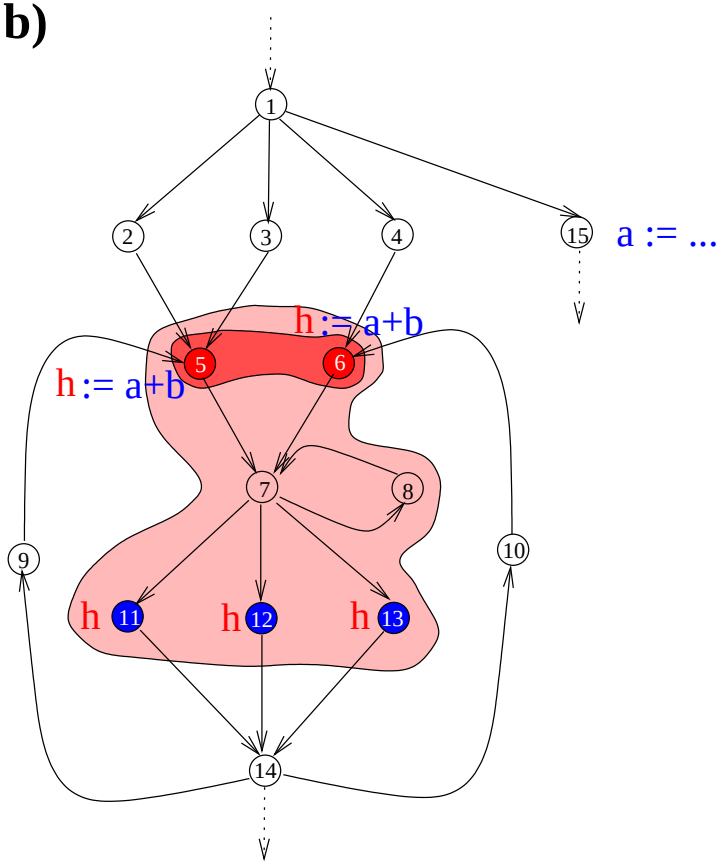


Largest Tight Set

(SQ > CQ)

Earliestness Principle

b)

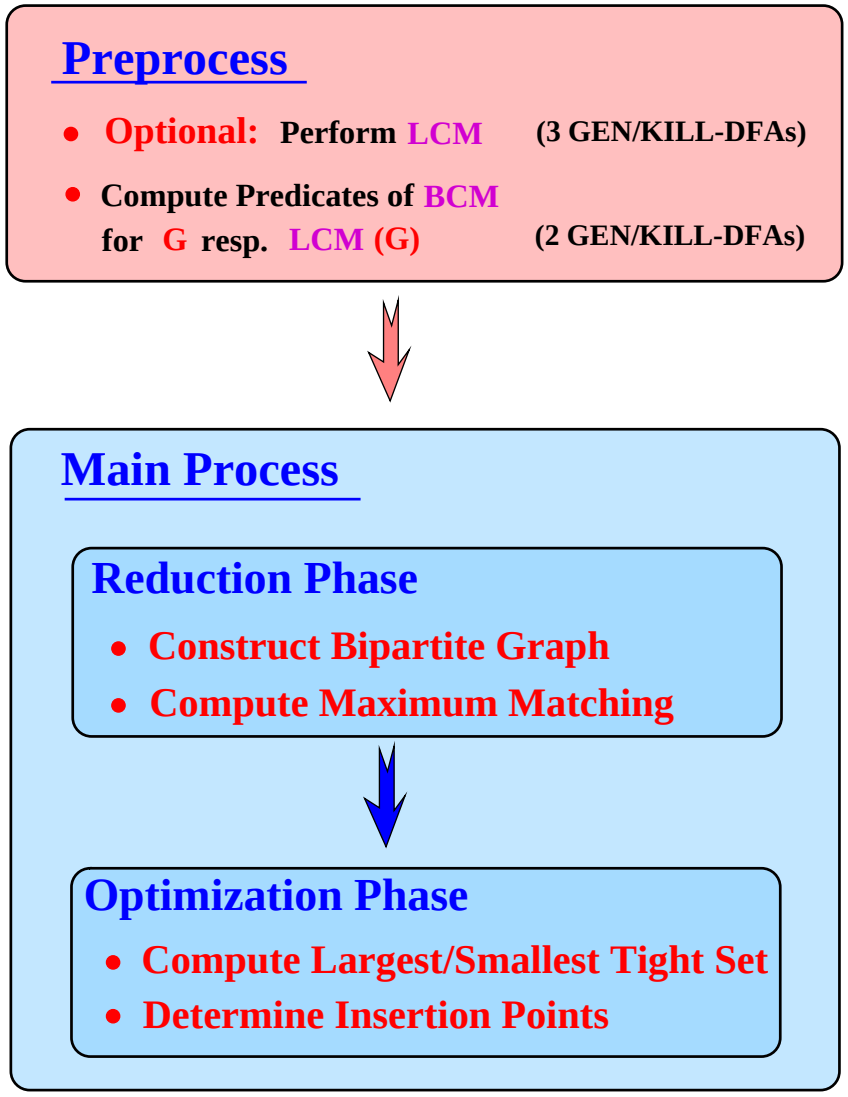


Smallest Tight Set

(SQ > LQ)

Latestness Principle

Code-Size Sensitive CM at a Glance



A brief overview on the history of CM...

- 1958: ...*first glimpse of PRE*
  - ⇝ Ershov's work on *On Programming of Arithmetic Operations*.
- **1979:** ...*origin of contemporary PRE*
  - ⇝ Morel/Renvoise's seminal work on PRE
- **1992:** ...*LCM* [Knoop et al., PLDI'92]
  - ⇝ ...first to achieve comp. optimality with minimum register pressure
  - ⇝ ...first to rigorously be proven correct and optimal
- **2000:** ...*origin of code-size sensitive PRE* [Knoop et al., POPL 2000]
  - ⇝ ...first to allow prioritization of goals
  - ⇝ ...rigorously be proven correct and optimal
  - ⇝ ...first to bridge the gap between traditional compilation and compilation for embedded systems

Overview (Cont'd)

- ca. since 1997: ...*a new strand of research on PRE*

  $\rightsquigarrow$ Speculative PRE: Gupta, Horspool, Soffa, Xue, Scholz, Knoop,...

- **2005**: ...*another fresh look at PRE* (*as maximum flow problem*)

  $\rightsquigarrow$ Unifying PRE and Speculative PRE [Jingling Xue and J. Knoop]

Part III: CM – Phenomena of its Derivatives

Optimality results are quite sensitive!

Three examples to provide evidence...

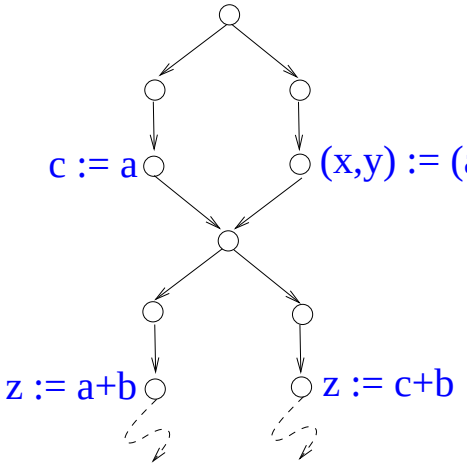
(A) Code motion vs. code placement

(B) Interdependencies of (elementary) transformations

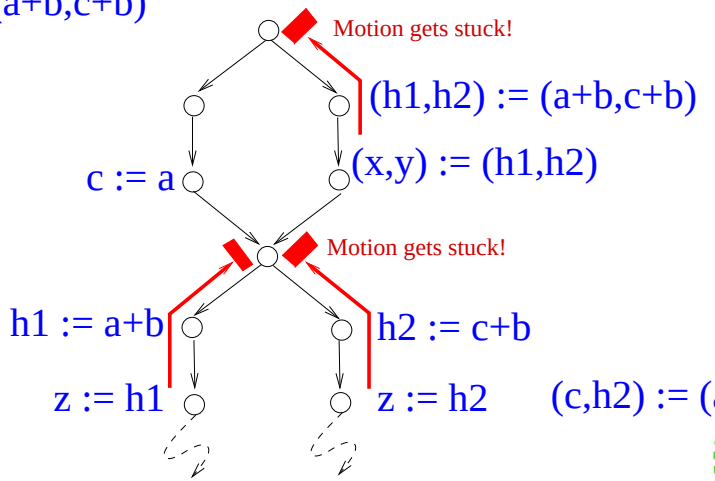
(C) Paradigm dependencies

(A) Code Motion vs. Code Placement

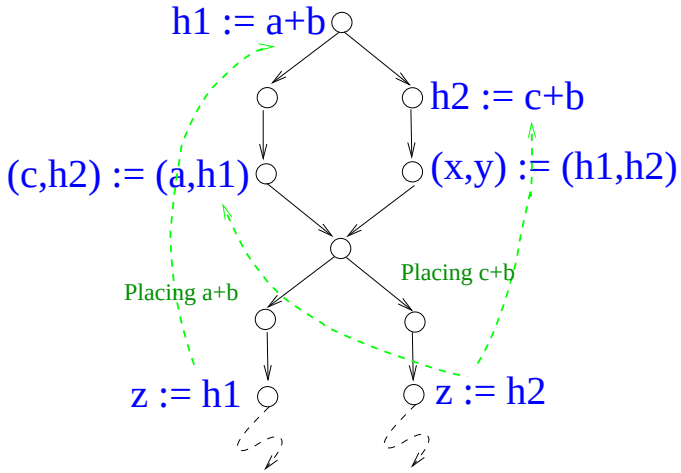
...not just synonyms!



Original Program



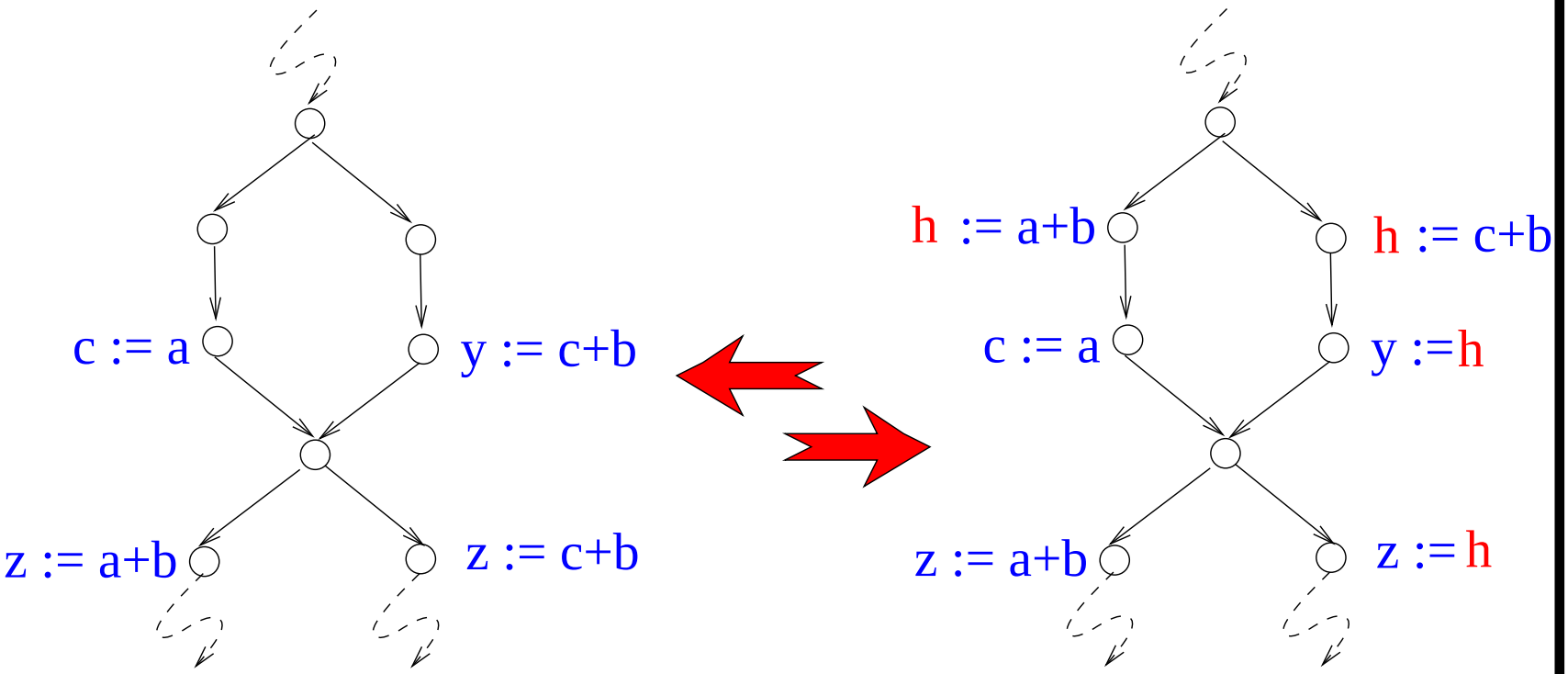
After Sem. Code Motion



After Sem. Code Placement

Even worse...

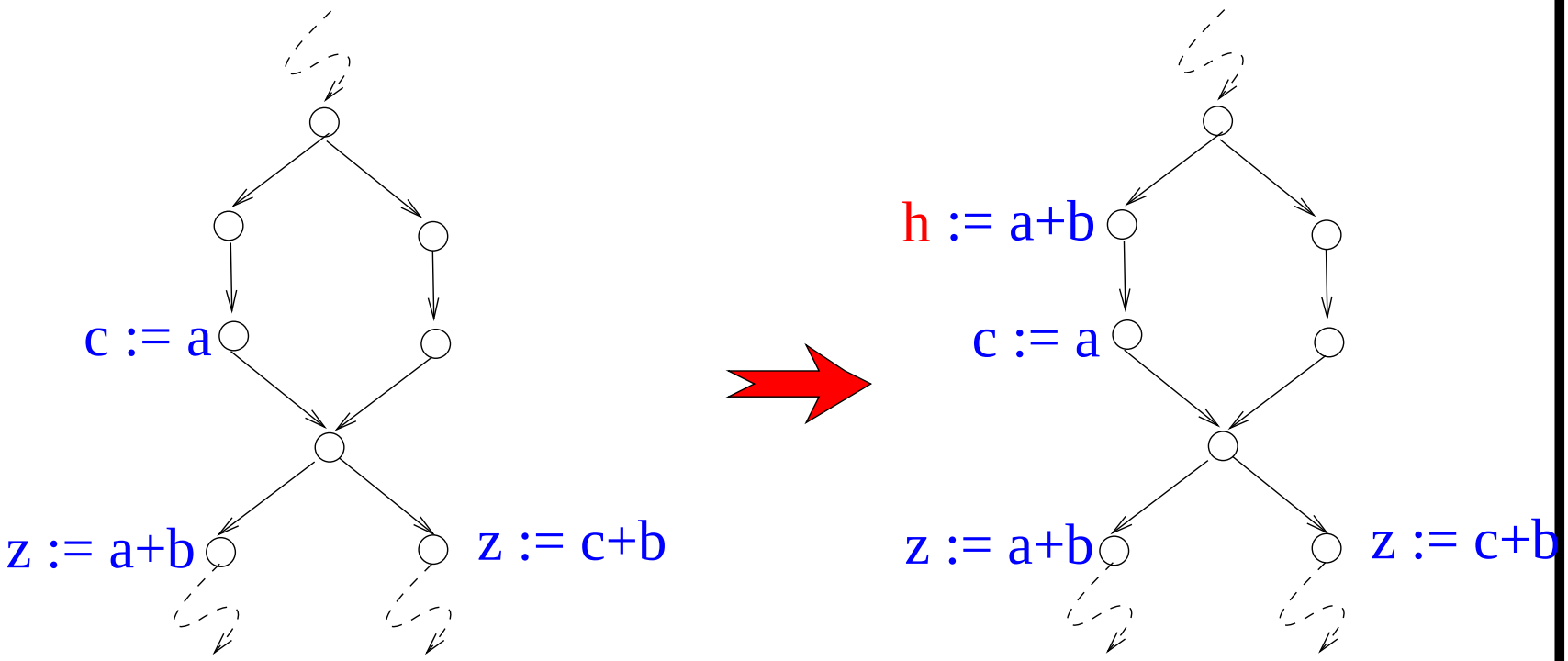
Optimality is lost!



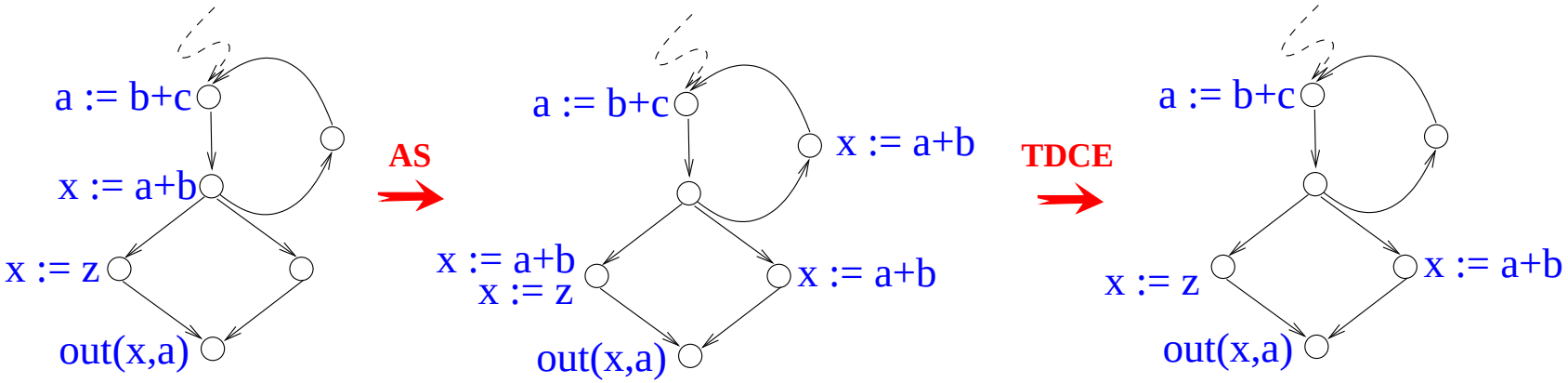
Incomparable!

Even more worse...

Peformance may be lost, when naively applied!



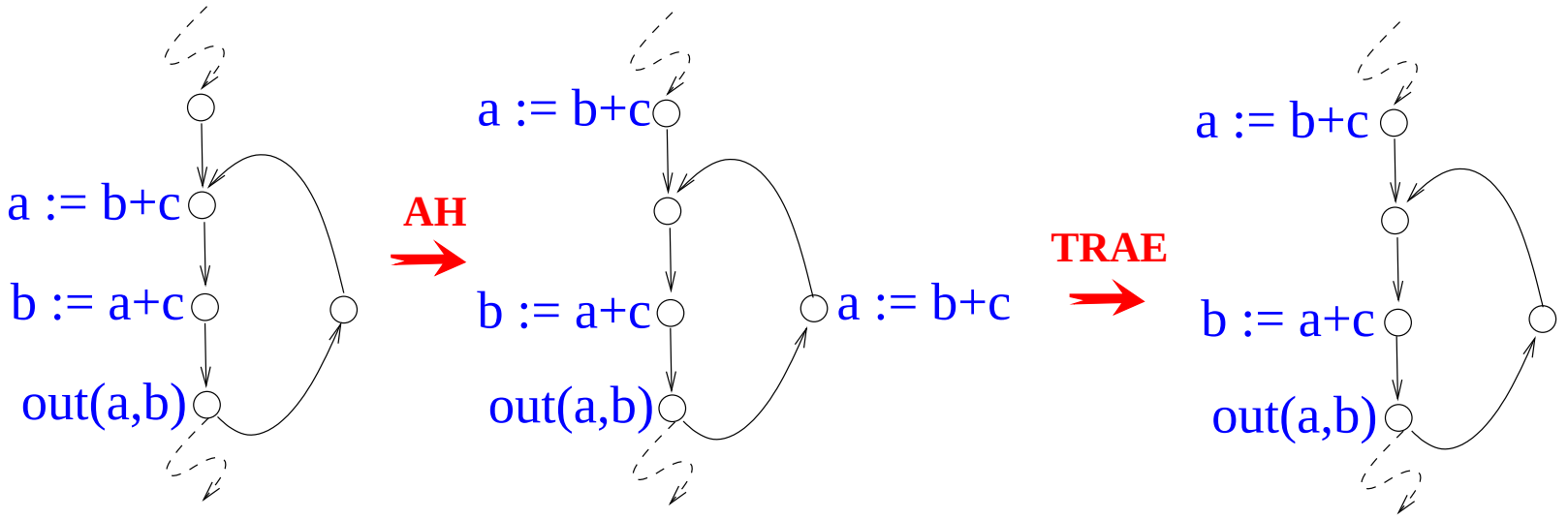
(B) Interdependencies of Transformations



...2nd Order Effects!

$\leadsto$ ...Partial Dead-Code Elimination (PDCE)

Interdependencies of Transformations



...2nd Order Effects!

$\leadsto$ ...Partially Redundant Assignment Elimination (PRAE)

Conceptually

...we can think of PREE, PRAE and PDCE in terms of

- PREE = AH ; TREE
- PRAE = (AH + TRAE)*
- PDCE = (AS + TDCE)*

PRAE/PDCE – Optimality Results

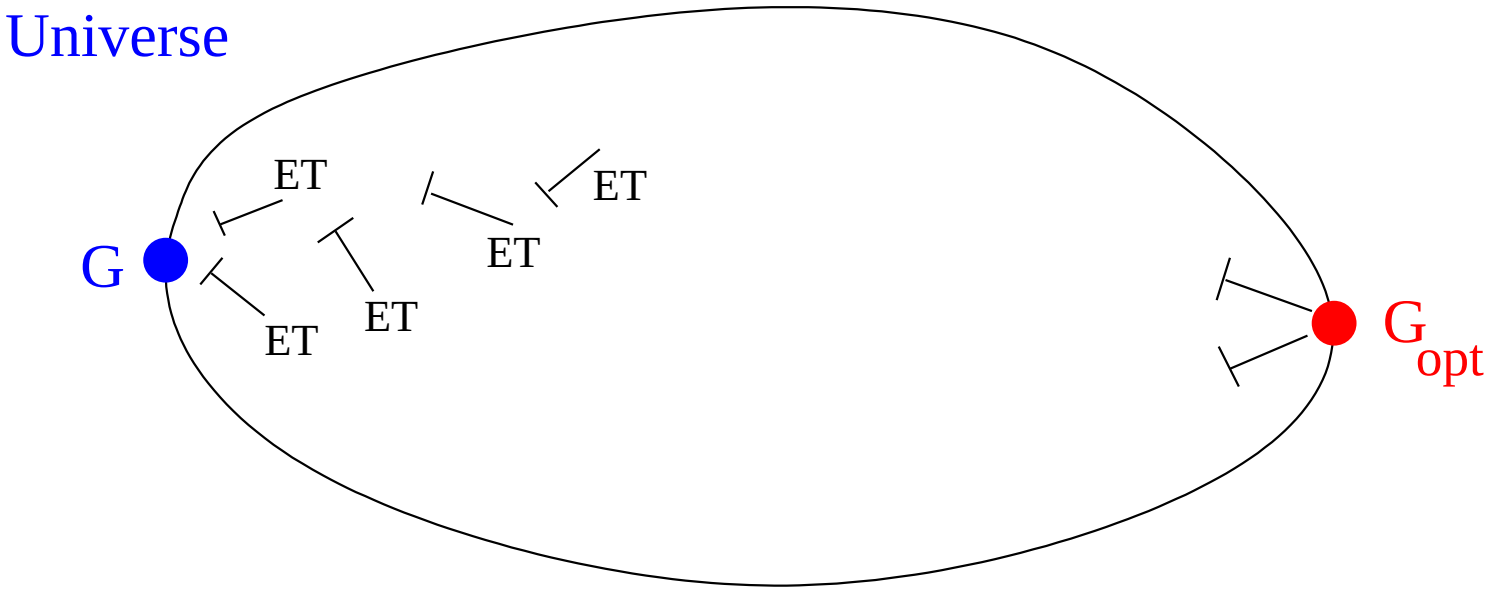
Derivation relation $\vdash$...

- PRAE... $G \vdash_{AH,TRAE} G'$
  (ET=$\{$AH,TRAE$\}$)
- PDCE... $G \vdash_{AS,TDCE} G'$
  (ET=$\{$AS,TDCE$\}$)

We can prove...

Optimality Theorem

For both PRAE and PDCE, $\vdash_{ET}$ is confluent and terminating


Universe
G
ET
ET
ET
ET
ET
G_{opt}

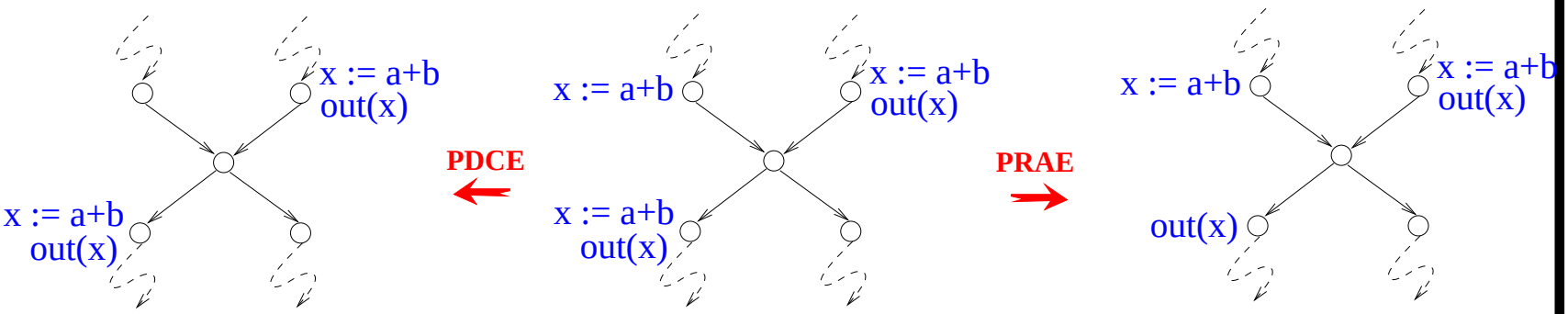
Consider now...

- Assignment Placement AP

  AP = (AH + TRAE + AS + TDCE)*

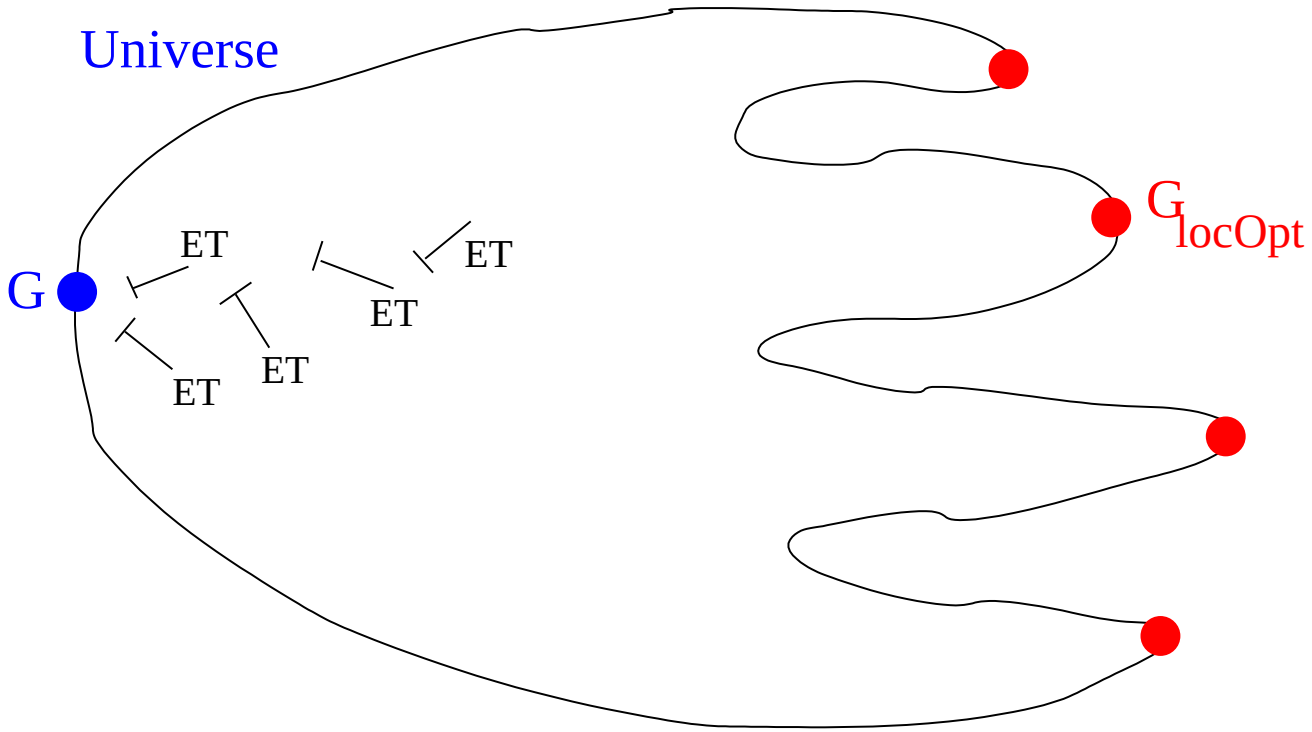
...should be even more powerful!

Indeed, but...



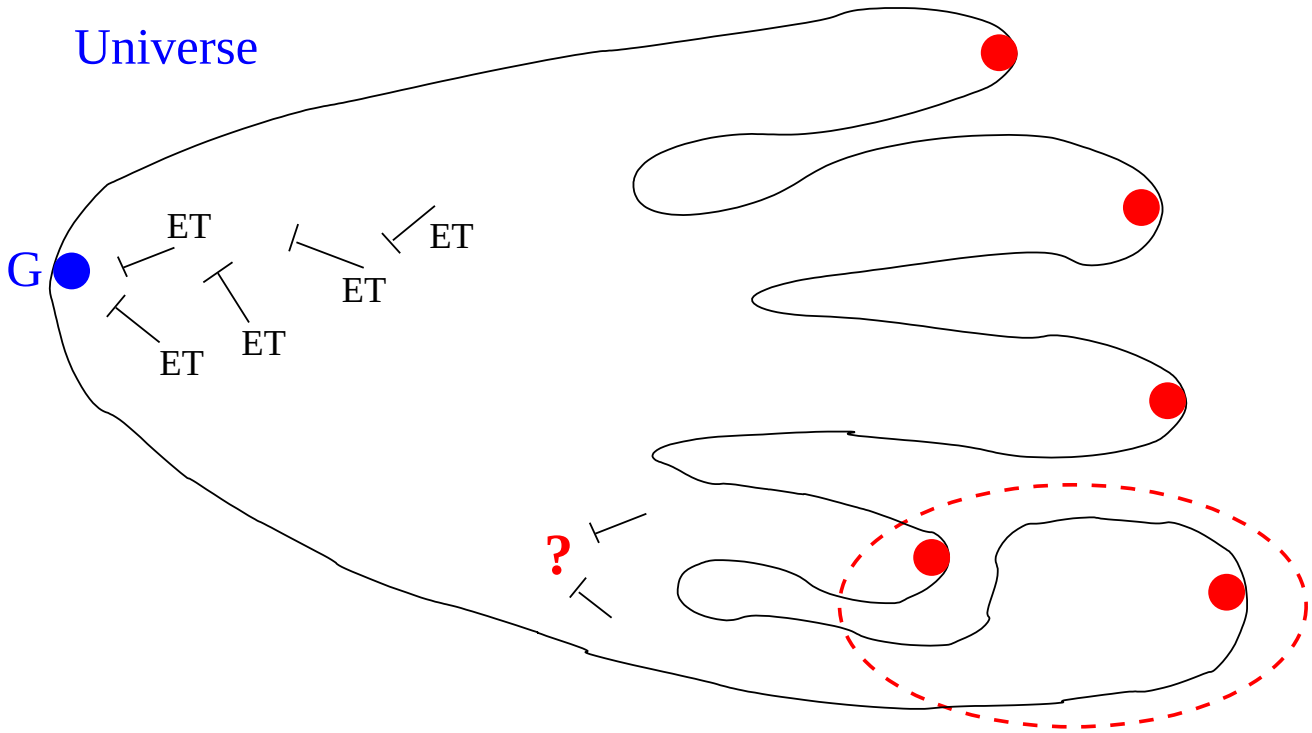
Confluence...

...and hence (global) optimality are lost!

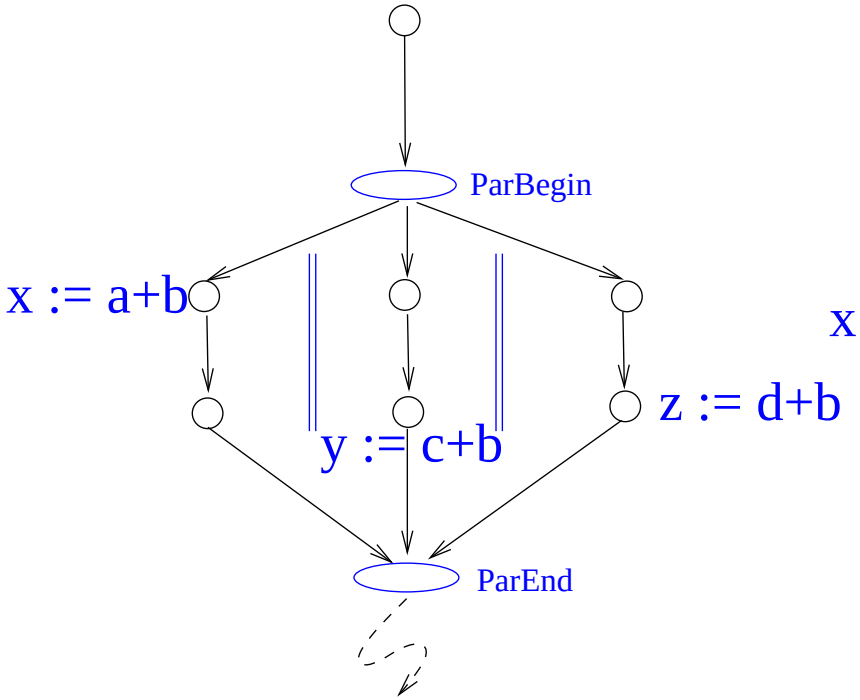


Even worse...

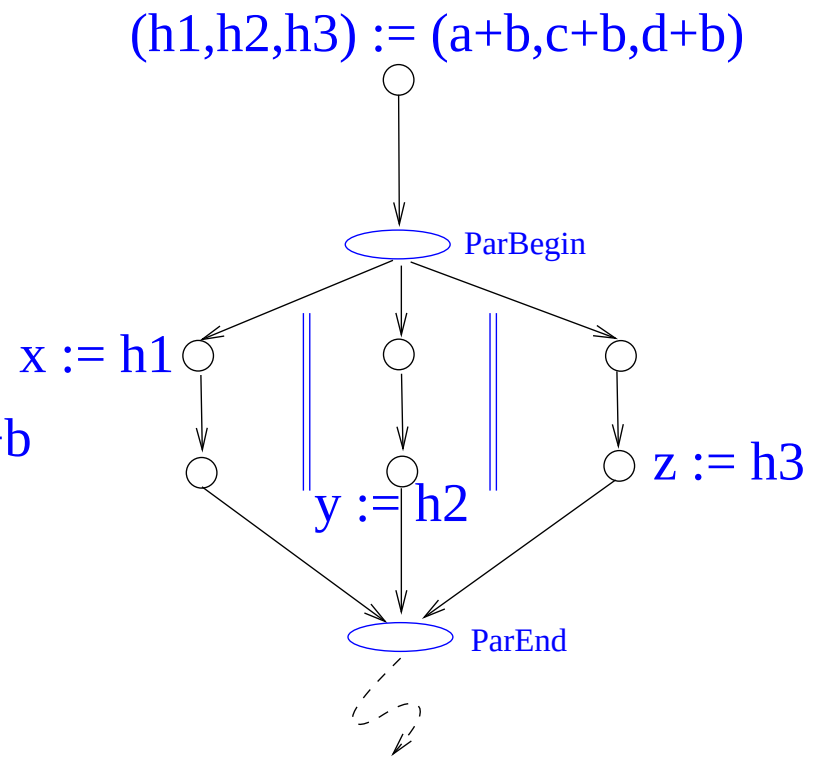
...there are scenarios, where we can end up with universes like



(C) Paradigm Dependencies



Original Program



After Earliestness Transformation

Part IV: CM – Recent Strands of Research

...another strand of research on CM is gaining more and more attention

- Speculative CM (SCM)

SCM – What's it all about?

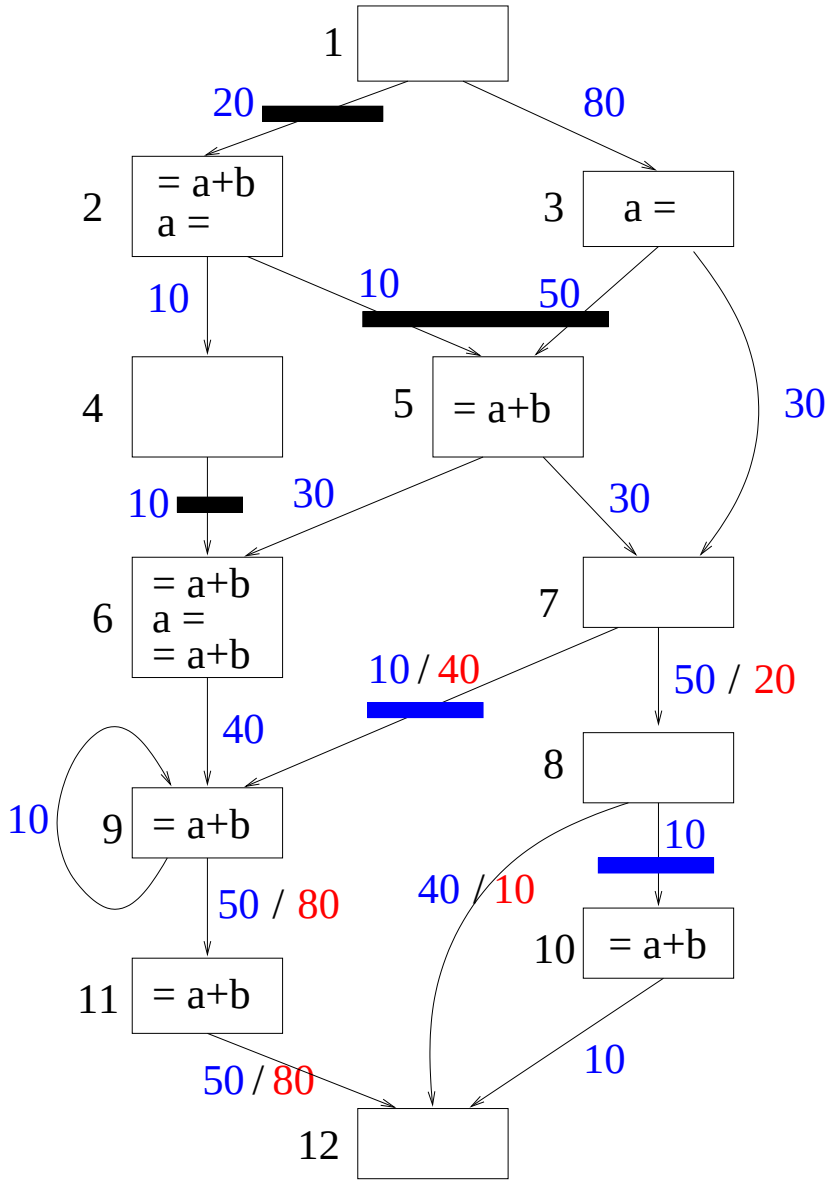
In contrast to CM

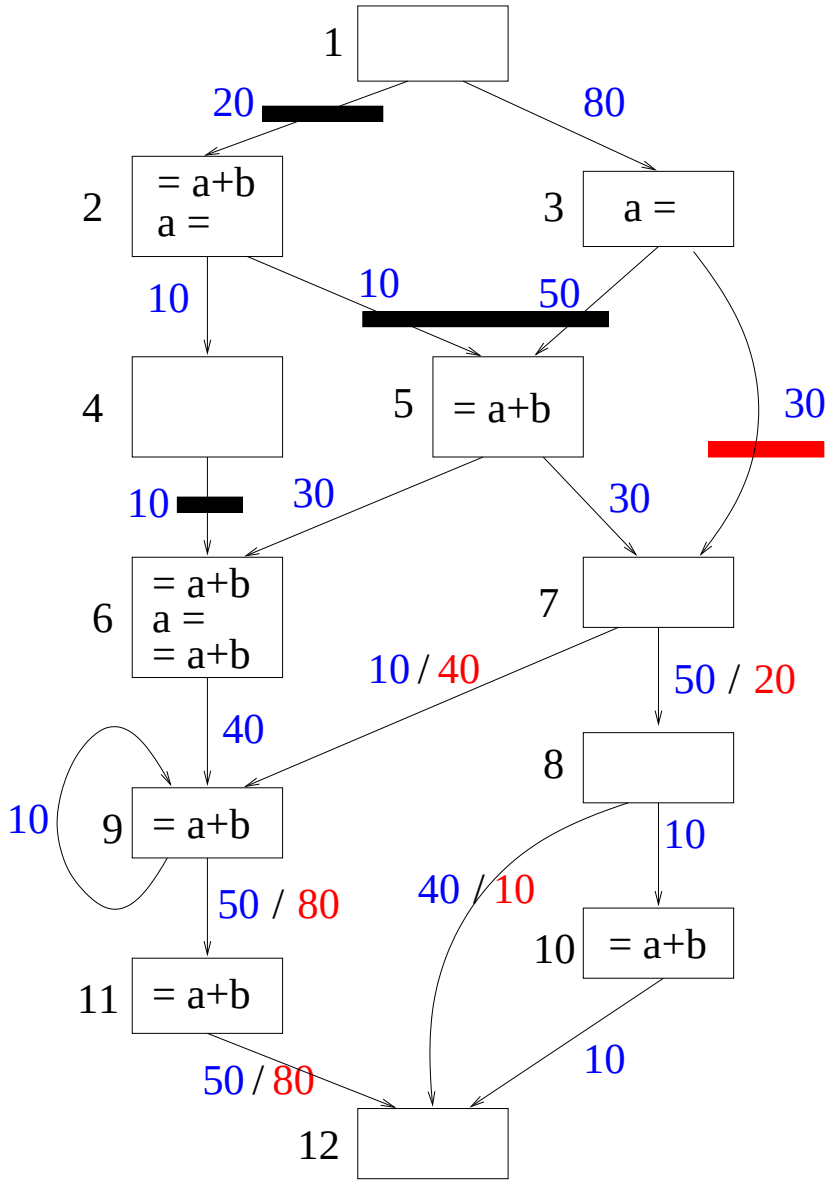
- SCM takes profile information into account

...thereby allowing to improve the performance of hot program paths at the expense of impairing cold program paths.

Anything else, especially the optimization goals,

- the same!


1
20
80
2
= a+b
a =
3
a =
10
10
50
4
5
= a+b
30
10
30
30
6
= a+b
a =
= a+b
7
10 / 40
50 / 20
40
8
10
9
= a+b
10
40 / 10
50 / 80
10
= a+b
11
= a+b
10
50 / 80
12


1
20
80
2
= a+b
a =
3
a =
10
10
50
4
5
= a+b
30
10
30
30
6
= a+b
a =
= a+b
7
10 / 40
50 / 20
40
8
10
9
= a+b
40 / 10
50 / 80
10
10
= a+b
11
= a+b
50 / 80
10
12

SCM vs. CM

Apparently

- SCM and CM are two closely related and very similar problems having much in common!

However

- SCM and CM are tackled by quite diverse algorithmic means
  - CM

    ...based on solving (typically) 4 bitvector analyses:

    Availability, Anticipability, ...
  - SCM

    ...based on solving a maximum flow problem

Recent Achievement

...the missing link between

- Classical PRE (CPRE) and Speculative PRE (SPRE)

On the theoretical side, this yields...

- a common high-level conceptual basis and understanding of CPRE and SPRE

On the practical side, we obtain...

- a new and simple algorithm for CPRE, which turns out to outperform its competitors

(joint work with Jingling Xue (CC 2006)

Major Finding

Like SCM

- CM is a maximum flow problem, too!

This means

- Each (S)CM-algorithm, if optimal, must find in one way or the other the unique minimum cut on a flow network derived from a program's CFG.

Hence, we have

- The Missing Link between CM and SCM!

On the Impact of this Finding 1(4)

Practically

- Possibly none

  ...at least not in terms of demanding replacement of implementations of optimal state-of-the-art CM algorithms by the flow-network based one.

Theoretically

- Possibly a lot

  ...a common high-level basis for understanding and reasoning about both SCM and CM.

On the Impact of this Finding 2(4)

This is in line with work on CM by other researchers striving for a simple and "motion-free" characterization of CM:

- Bronnikov, D., *A Practical Adaption of Partial Redundancy Elimination*, SIGPLAN Not., 39(8), 49-53, 2004.
- Dhamdhere, D. M., *E-Path_pre: Partial Redundancy Elimination Made Easy*, SIGPLAN Not., 37(8), 53-65, 2002.
- Paleri, V. K., Srikant, Y. N., Shankar, P., *A Simple Algorithm for Partial Redundancy Elimination*, SIGPLAN Not., 33(12), 35-43, 1998.

On the Impact of this Finding 3(4)

However, for these approaches

- either no proofs of correctness and optimality are given
- or these proofs still rely on a low-level path-based reasoning

Especially in this respect, the characterization of

- CM as a maximum flow problem

can be considered a major step forward.

On the Impact of this Finding 4(4)

A practical impact though...

Based on the new understanding, we obtained

- a new and simple CM-algorithm
  - *Like its competitors*: ...relies on 4 bitvector analyses
  - *At first sight thus*: ...yet another CM-algorithm
  - *But*: ...outperforms its competitors

Practical Measurements

...of the new algorithm show

- a reduction in the number of bitvector operations required ranging from 20% to 60% in comparison to three state-of-the-art algorithms for CM (including LCM and E-path)

Experiments were performed

- on an Intel Xeon and a Sun UltraSPARC-III platform
- with the GCC-compiler as vehicle
- using all of the 22 C/C++/Fortran SPECcpu2000 benchmarks

Conclusions and Perspectives

- Code Motion (CM)

  ...a hot topic of on-going research for almost 50 years!

- State-of-the-Art in Theory and Practice
  - Theory available and widely used in practice
    - Classic CM
  - Theory available, but not yet widely used
    - Derivatives of Classic CM (PDCE, PFCE, SR, DAP,...)
    - Speculative CM and some derivatives (SR)
    - Semantic CM
  - Theory not yet available
    - Speculative Semantic CM
    - ...

Conclusions and Perspectives

- Our obligation
  - Pushing forward the further development of CM-based optimizations
  - Demanding their application (e.g. in the Phoenix framework)

  ...in order to help the impatient (.Net) programmer and user!

Perspectives

Predicting the future...

- Niels Bohr: *"Predictions are always difficult, especially about the future.*

Perspectives: 50 Years from Now...

The future will be bright!

In particular, I predict that...

- we will celebrate the...
  - 300th Anniversary of the Birth of Mozart
  - 200th Anniversary of the Birth of Freud
  - 100th Anniversary of the begin of CM-Research
- and that...

Perspectives: 50 Years from Now...

...we will all meet again at

The 54th Annual .Net Technologies 2056 Conference in Central Europe

June 1 - 5, 2006, Plzen, Czech Republic

Perspectives: 50 Years from Now...

Browsing the programme of .Net Technologies 2056, I foresee to see...

Keynote Speech:

From .Net to .Net/XP:
The Role and the Impact of a 100 Years of CM-Research

Thank you!

Questions?

Acknowledgements: Most of the results reported are joint work with *Oliver Rüthing* (U. Dortmund), *Bernhard Steffen* (U. Dortmund), *Eduard Mehofer* (U. Wien), and more recently *Bernhard Scholz* (U. Sydney), *Nigel Horspool* (U. Victoria), and *Jingling Xue* (Univ. of New South Wales).