

# Complete Removal of Redundant Expressions

Rastislav Bodík

Rajiv Gupta

Mary Lou Soffa

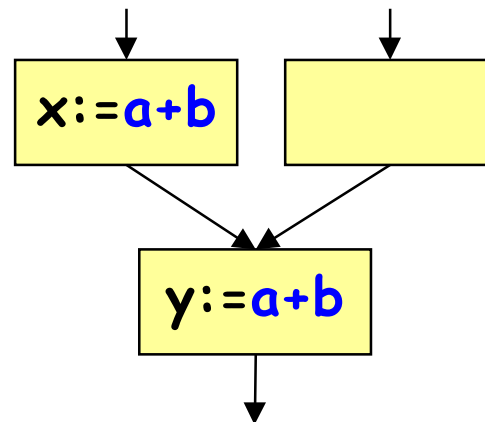
University of Pittsburgh

# Partially redundant expressions

---

2

**Partially redundant:** computed on some incoming path.

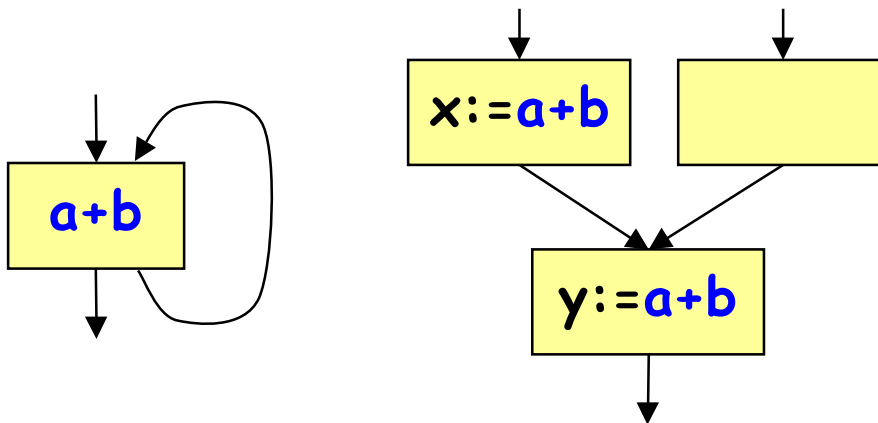


# Partially redundant expressions

---

2

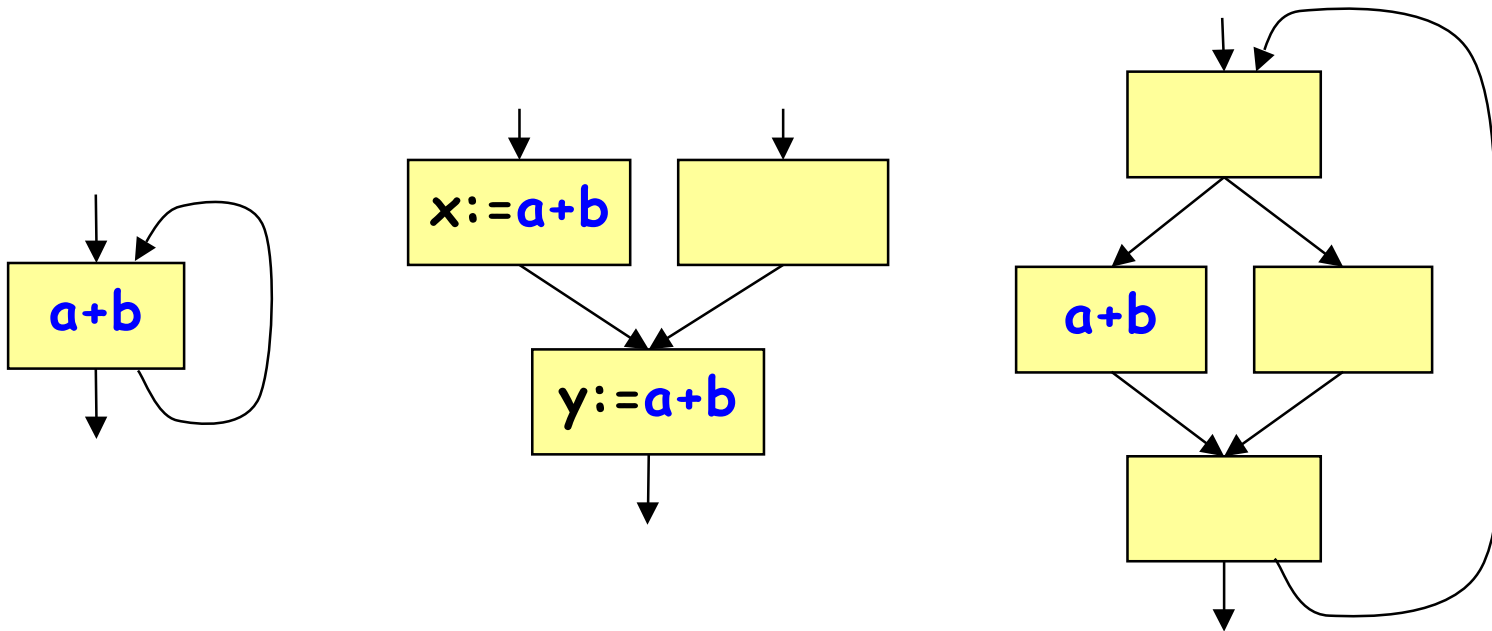
**Partially redundant:** computed on some incoming path.



# Partially redundant expressions

2

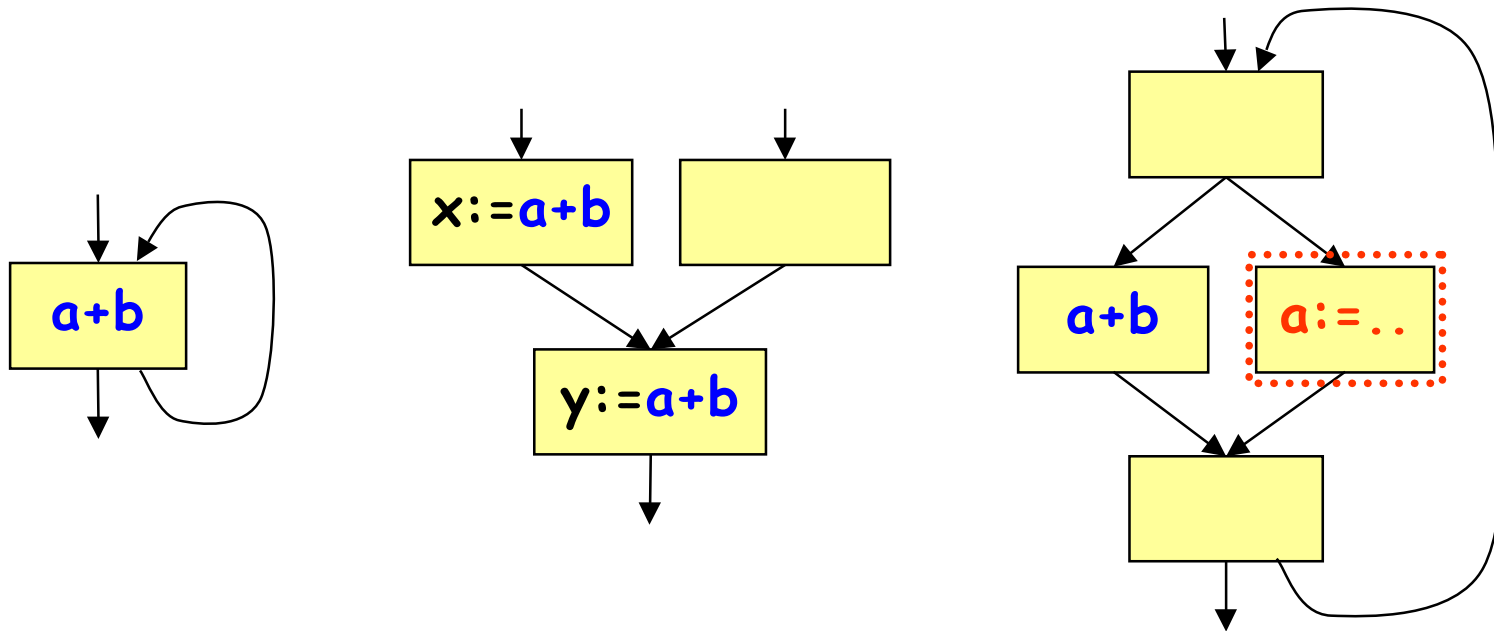
**Partially redundant:** computed on some incoming path.



# Partially redundant expressions

2

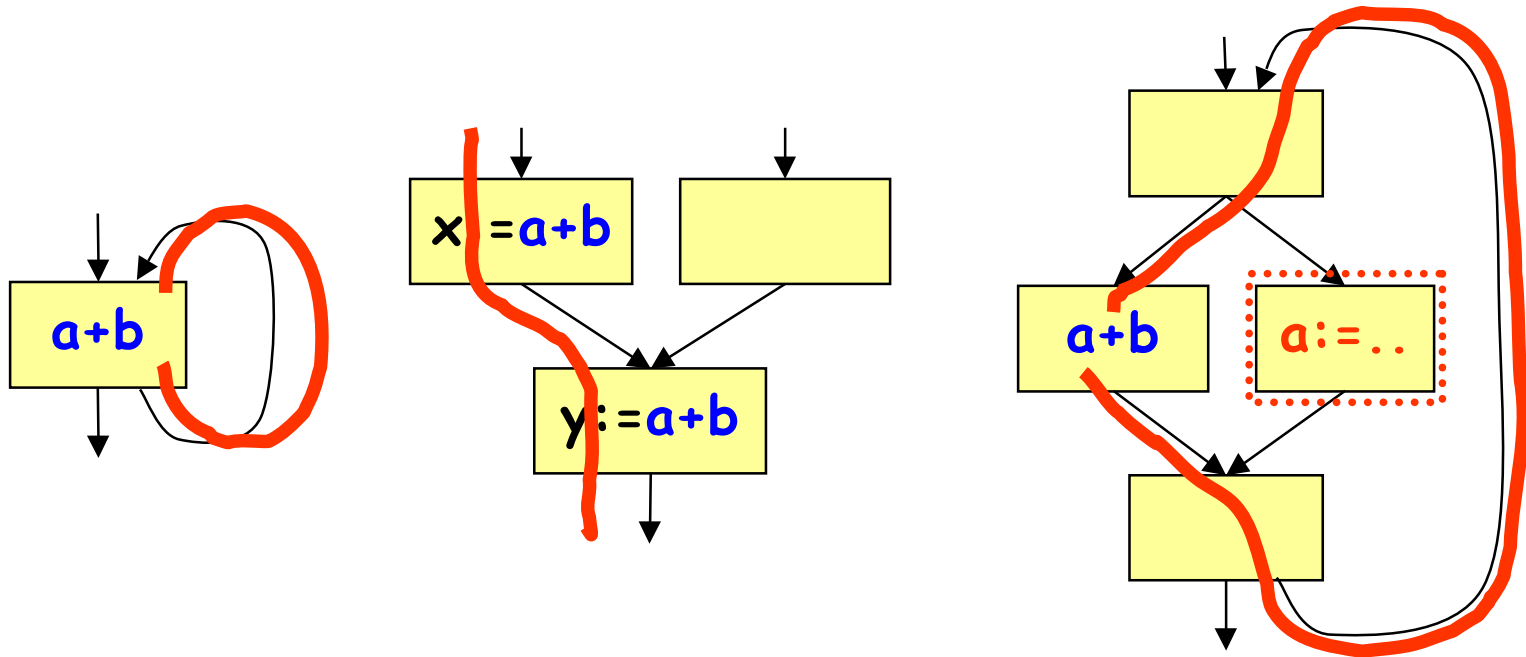
**Partially redundant:** computed on some incoming path.



# Partially redundant expressions

2

**Partially redundant:** computed on some incoming path.



**Goal:** remove redundancy from "reuse" paths.

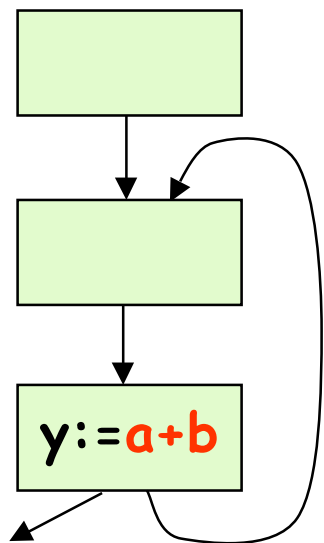
# PRE via code motion

[Morel, Renviose '79]

3

Hoist partially redundant expression until it becomes:

- **fully** redundant  $\Rightarrow$  **remove it**
- **not** redundant  $\Rightarrow$  **insert it**



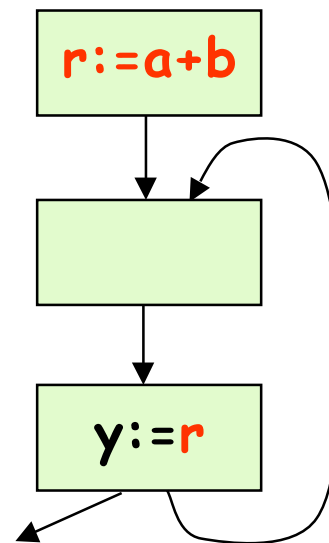
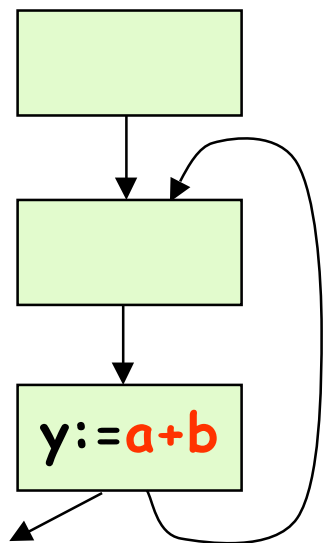
# PRE via code motion

[Morel, Renviose '79]

3

Hoist partially redundant expression until it becomes:

- **fully** redundant  $\Rightarrow$  **remove it**
- **not** redundant  $\Rightarrow$  **insert it**



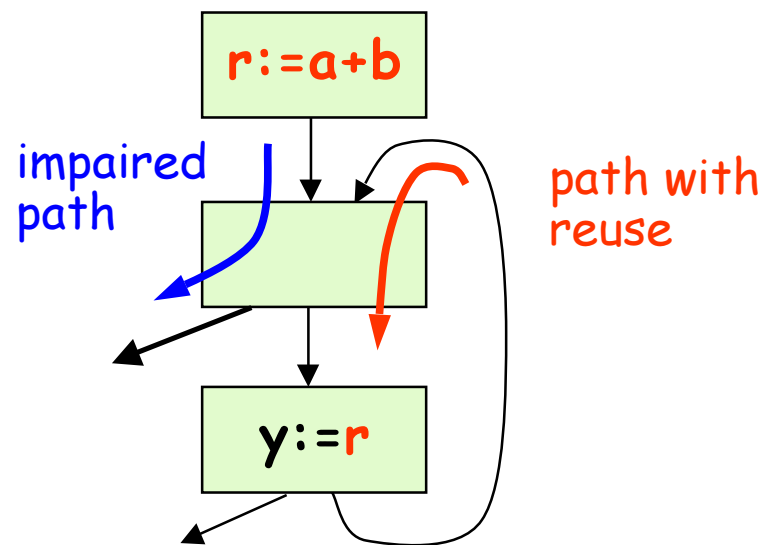
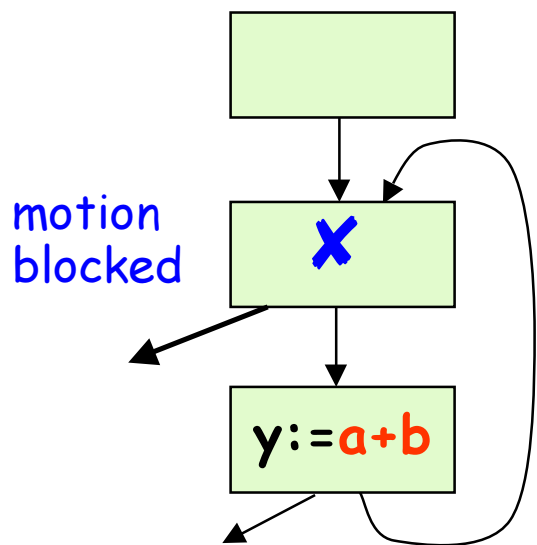
# PRE via code motion

[Morel, Renviose '79]

3

Hoist partially redundant expression until it becomes:

- **fully** redundant  $\Rightarrow$  **remove it**
- **not** redundant  $\Rightarrow$  **insert it**



PRE fails on 70% of loop-invariants

# PRE limitations

4

## Optimization model

**Too conservative**

**Rule:**

improve "reuse" paths  
**without**  
impairing other paths

## Program transformation

**Not aggressive**

Code motion only:

motion blocked  
⇓  
opportunities missed

## Our approach:

**Relaxed**

allow control speculation

**Aggressive**

apply restructuring

integration



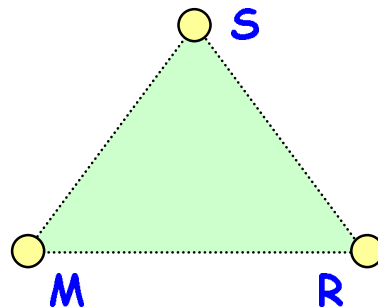
# Three PRE transformations

5

- impairs some paths
- no code growth

## control speculation

### code motion



### CFG restructuring

- misses opportunities
- no code growth

- complete
- code growth

# Talk outline

---

6

## ~ Motion + restructuring

safe model

reduce code growth

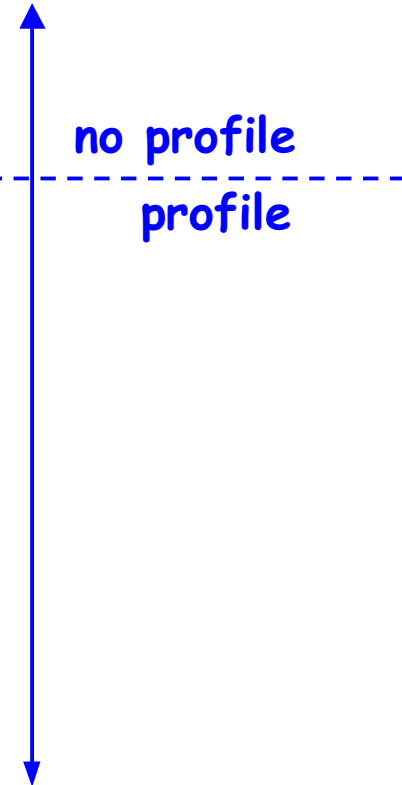
## ~ Motion + speculation

relaxed model

## ~ Dynamic benefit

large-scale computation

## ~ Related work & summary



# Talk outline

6

## ~ Motion + restructuring

safe model    complete PRE/minimal growth    no profile  
-----  
reduce code growth    near-complete PRE    profile

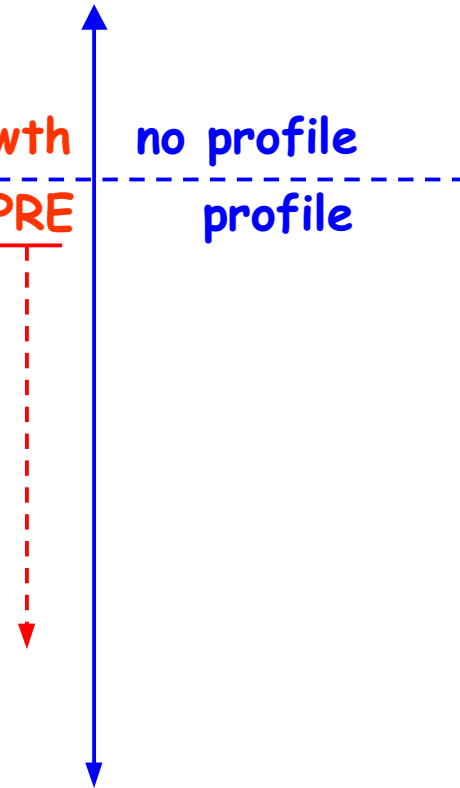
## ~ Motion + speculation

relaxed model    maximal PRE/no growth

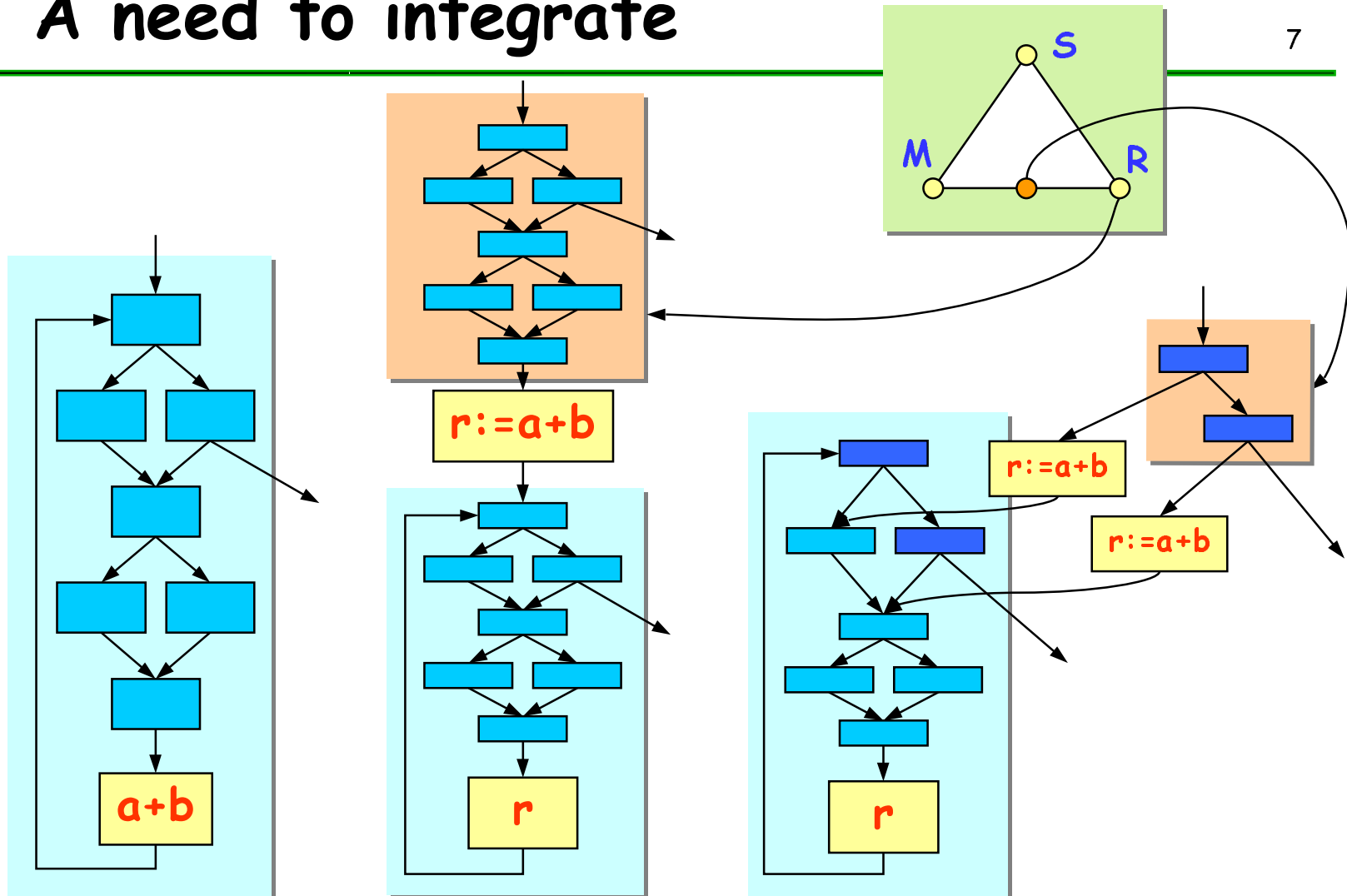
## ~ Dynamic benefit

large-scale computation

## ~ Related work & summary



# A need to integrate

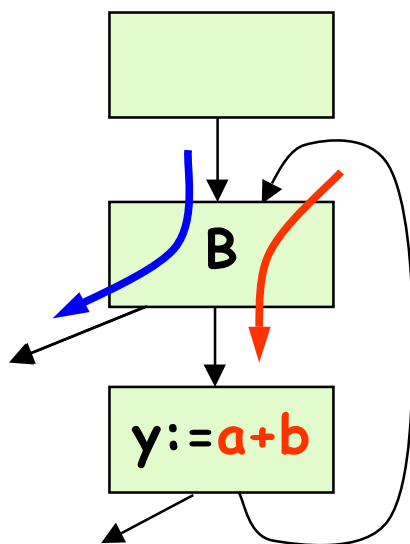


# CMP: code motion preventing region

---

8

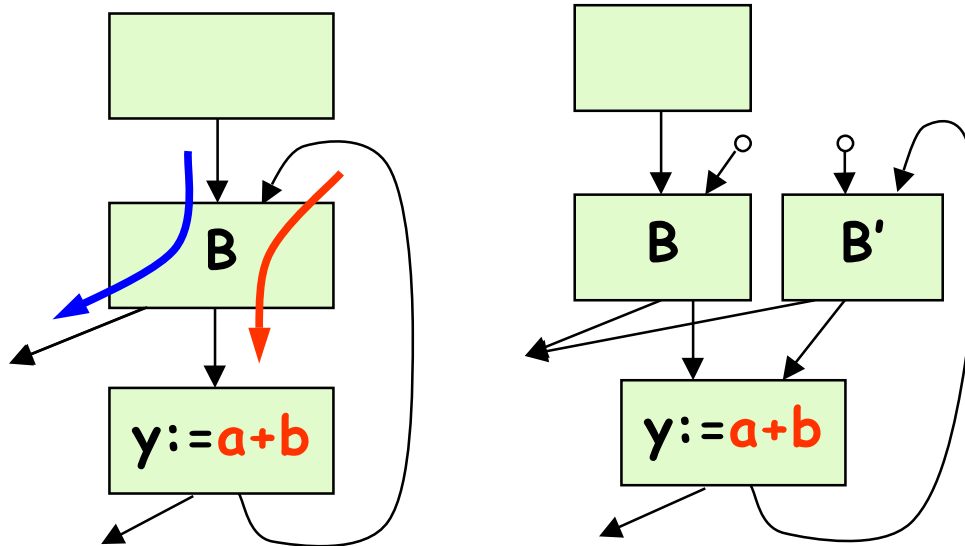
👉 duplicate only nodes blocking code motion



# CMP: code motion preventing region

8

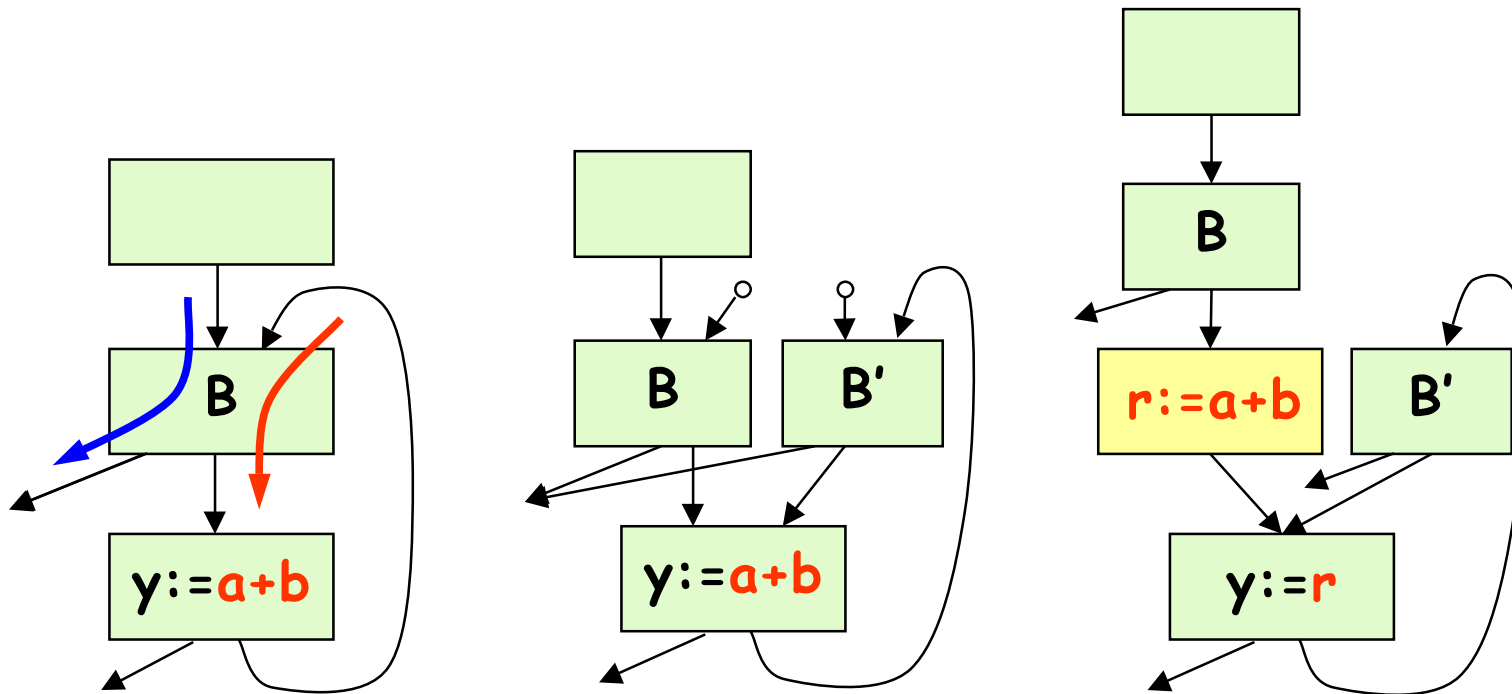
👉 duplicate only nodes blocking code motion



# CMP: code motion preventing region

8

👉 duplicate only nodes blocking code motion



# Identifying CMP regions

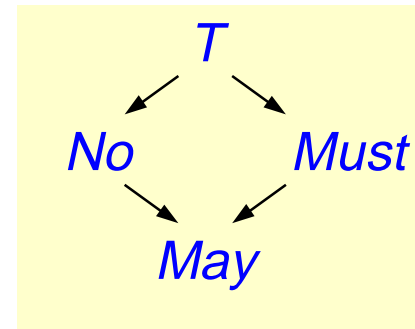
9

**AVAIL** - availability

incoming paths

**ANTIC** - anticipability

outgoing paths



$AVAIL[n, a+b] = \begin{matrix} \textit{Must} \\ \textit{No} \\ \textit{May} \end{matrix}$  available along  $\begin{matrix} \textit{all} \\ \textit{no} \\ \textit{some} \end{matrix}$  paths

$CMP[a+b] =$  set of nodes  $n$  s.t.

$AVAIL[n, a+b] = \textit{May} \wedge ANTIC[n, a+b] = \textit{May}$

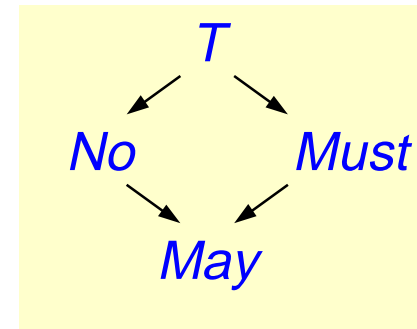
# Identifying CMP regions

**AVAIL - availability**

incoming paths

**ANTIC - anticipability**

outgoing paths

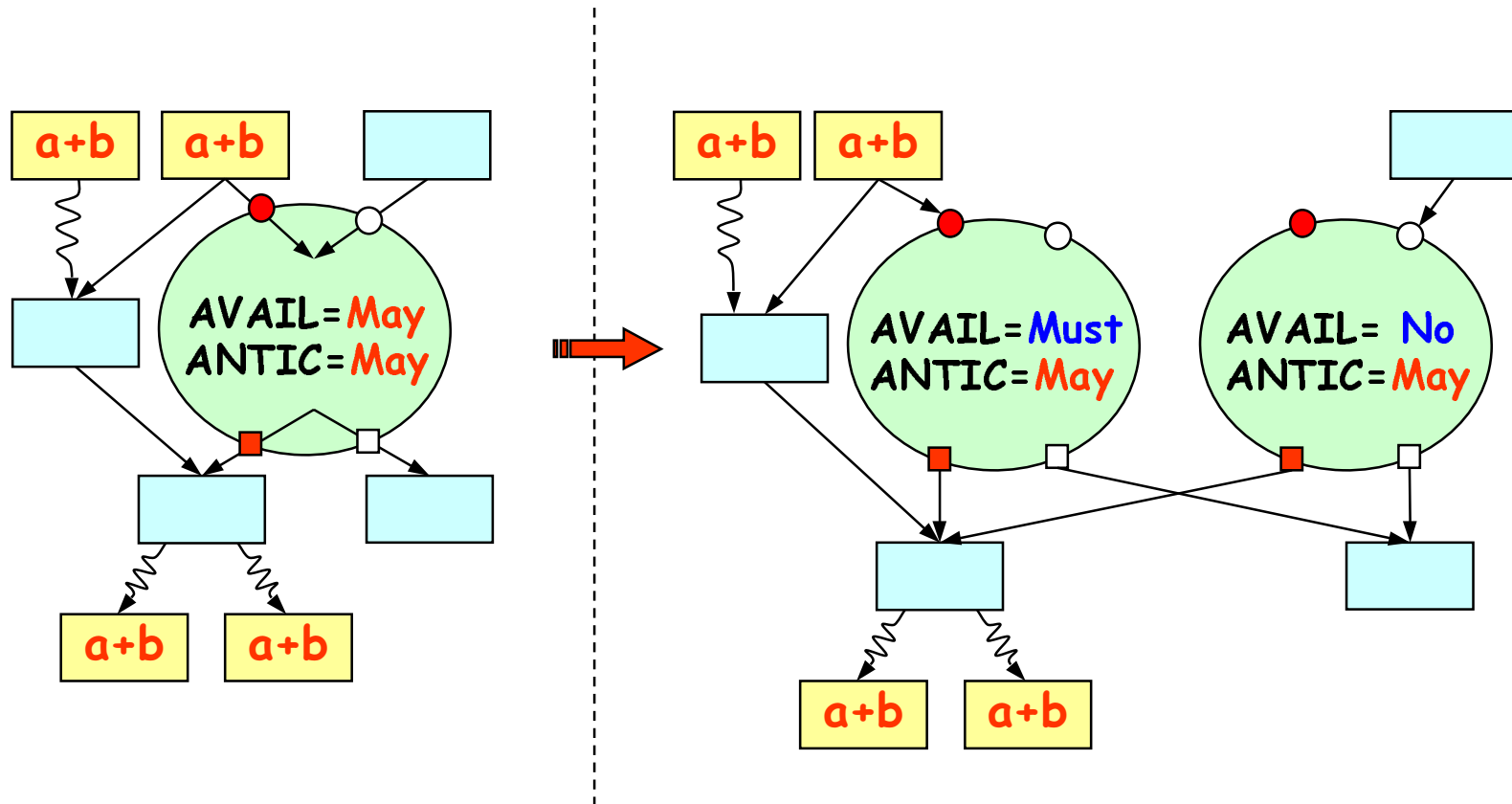


AVAIL[n, a+b] = *Must* available along *all* *no* paths  
*May* *some*

CMP[a+b] = set of nodes n s.t.

AVAIL[n, a+b] = *May*  $\wedge$  ANTIC[n, a+b] = *May*  
 impaired path ..... *No* ..... *No*  
 "reuse" path ..... *Must* ..... *Must*

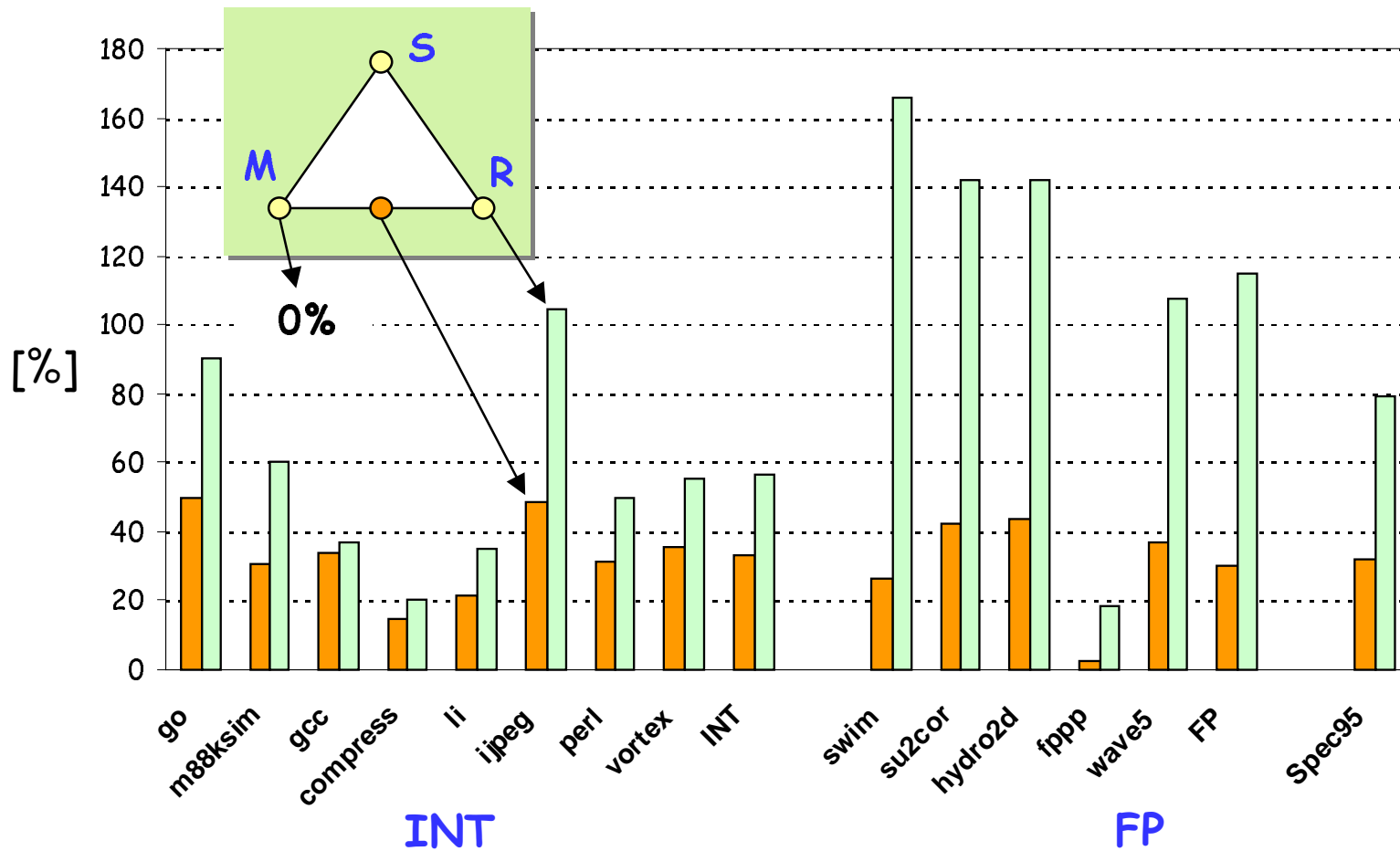
# Complete PRE



- AVAIL=Must
- AVAIL=No
- ANTIC=Must
- ANTIC=No

# Code growth

**motion+restructuring vs. restructuring-only**



# 1. Duplicate only profitable CMP's

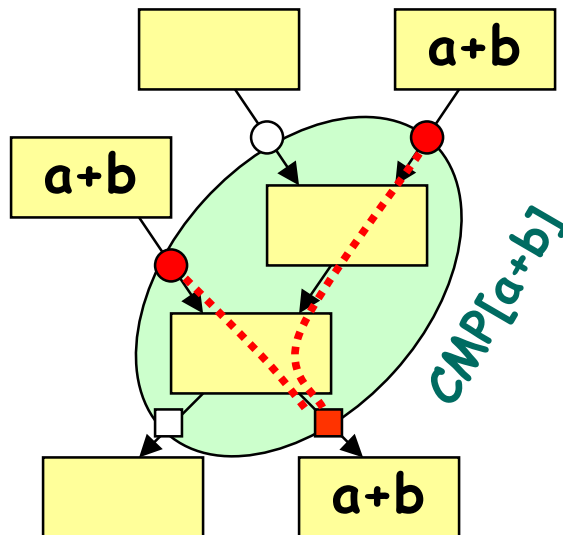
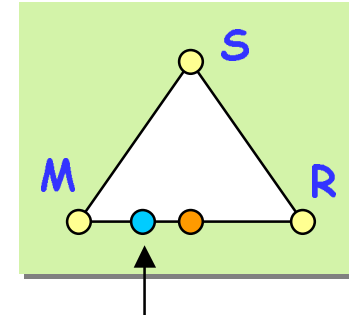
12

profitable

$\text{benefit} > c.\text{size}(\text{CMP})$

benefit

dynamic amount of eliminated expressions



Computing the benefit

edge profile

probability of  $a+b$  being available

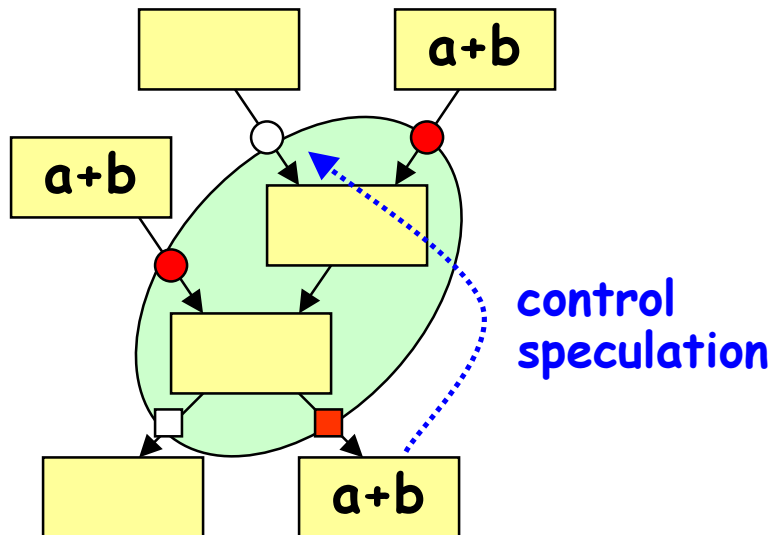
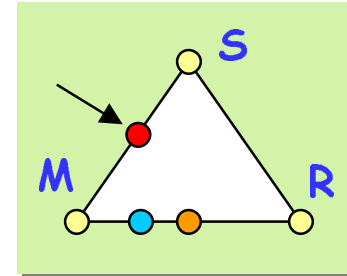
path profile

sum of **Must-Must** paths freq's

## 2. Motion + speculation

13

- Relaxed model,
- hoist into No-AVAIL entries,
- ☠ harmful without profile info.



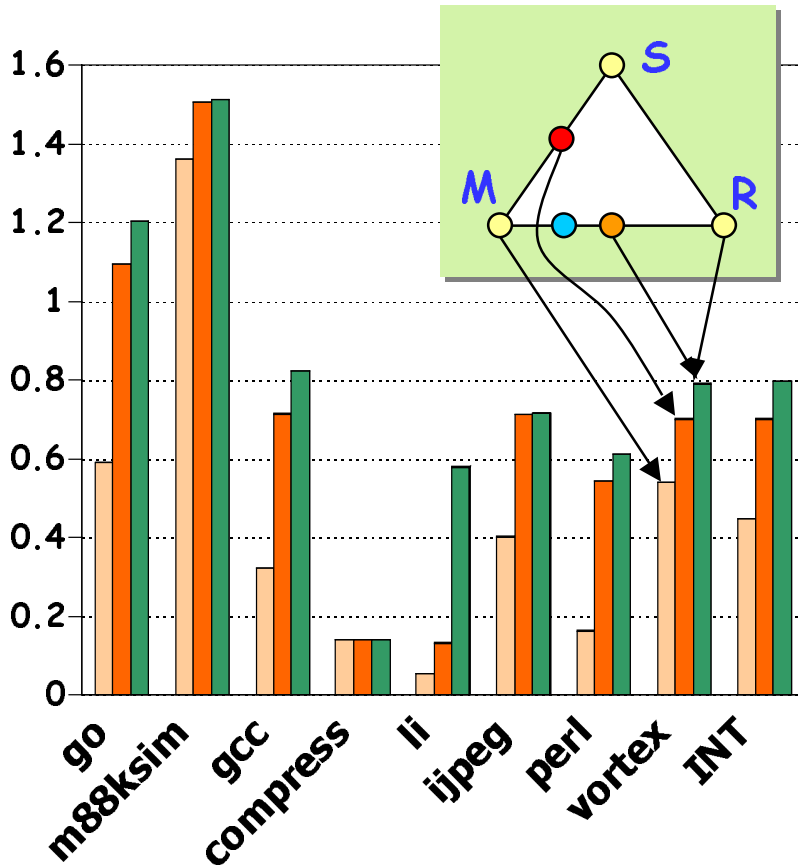
Computing the benefit  
edge profile

+freq(■) removal  
-freq(○) insertion

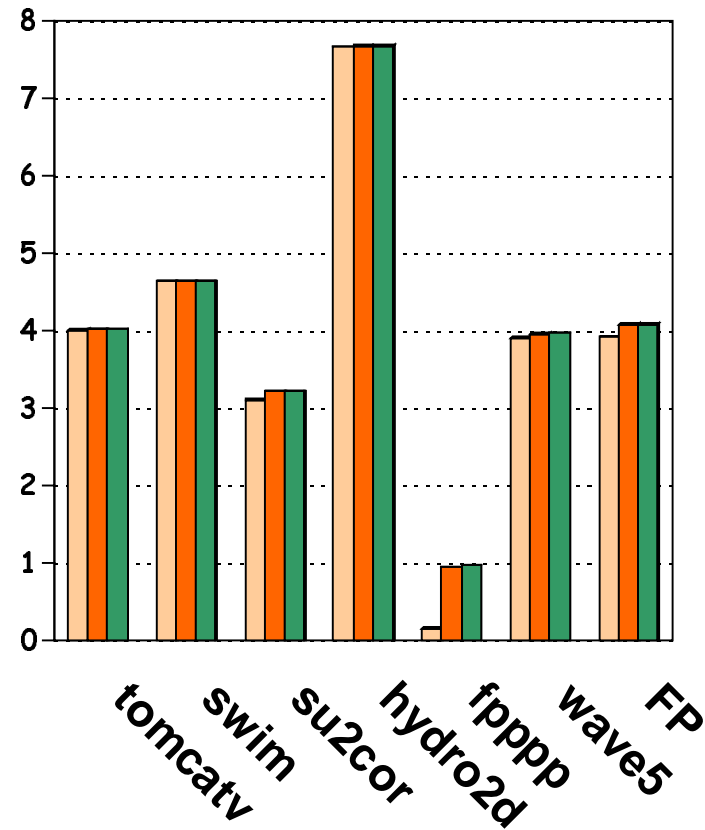
---

+freq(■) - freq(○)

# Partial redundancy removed



% of dynamic expressions



INT

FP

# Computing benefit on large scale

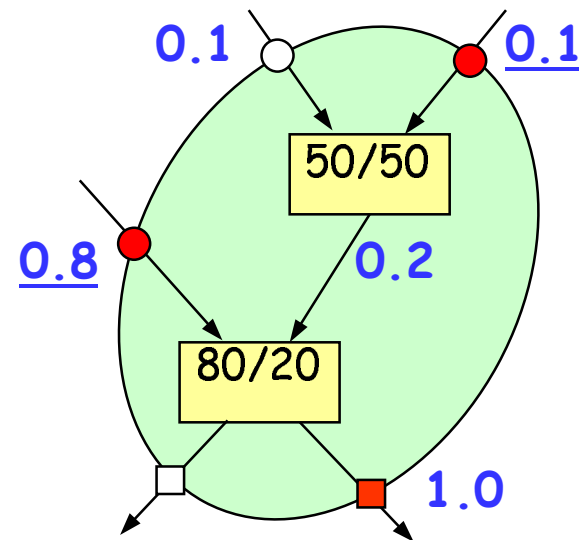
15

## Goals:

- focus on hot paths, **but**
- drop cold nodes only within allowed imprecision.

## Approximate frequency analysis

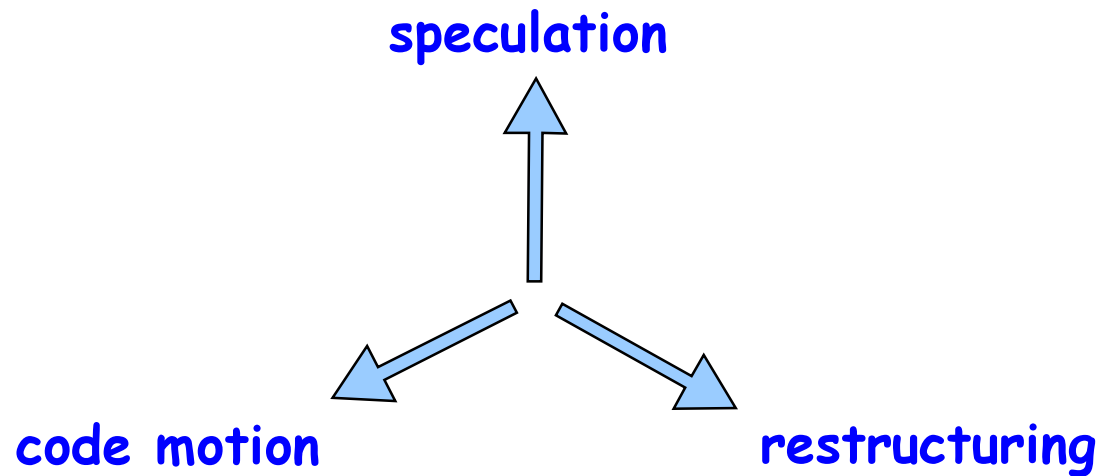
- extension of [Ramalingam '96]
- propagate contribution weight
- *demand* interval analysis



# The three transformations in concert

---

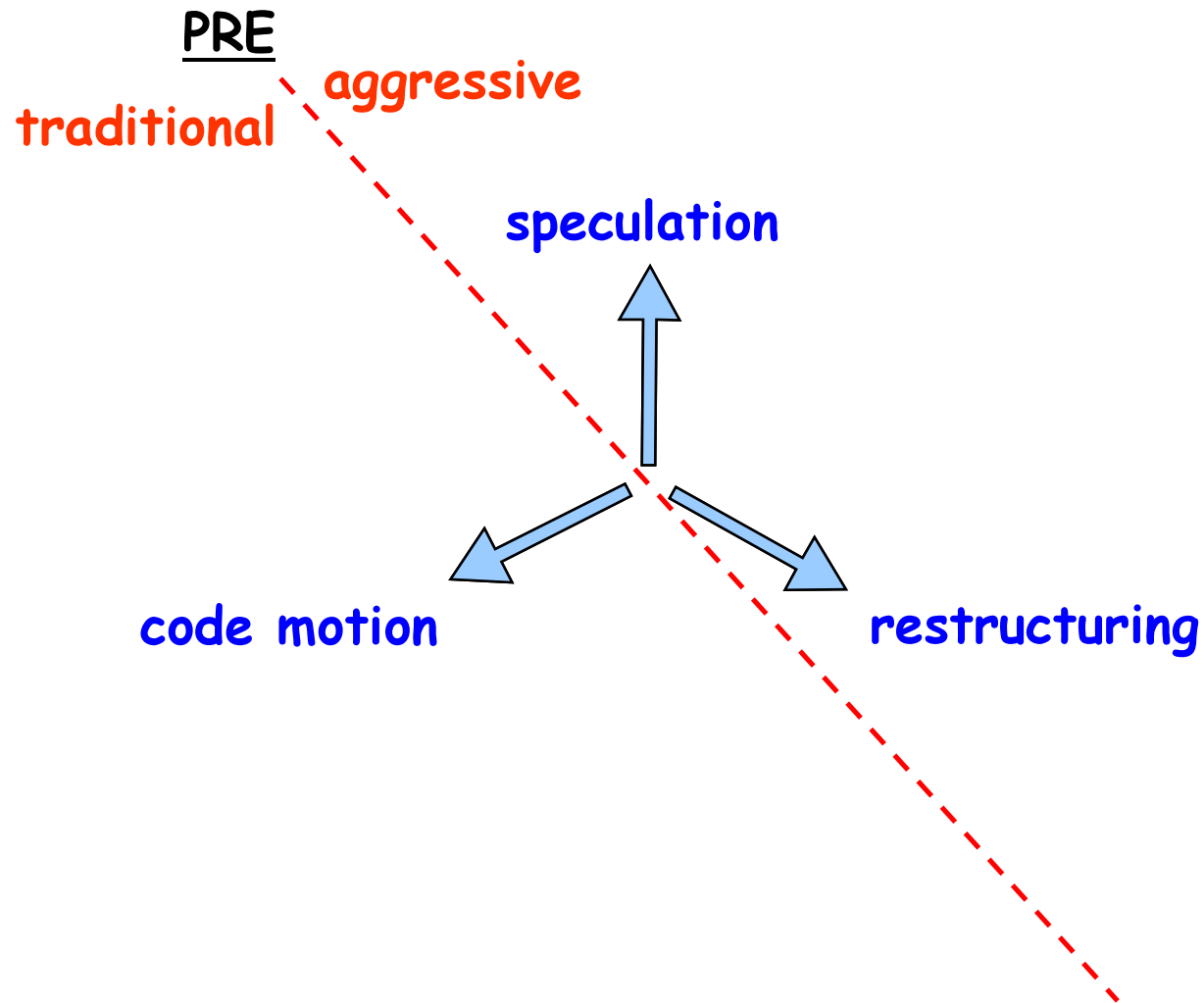
16



# The three transformations in concert

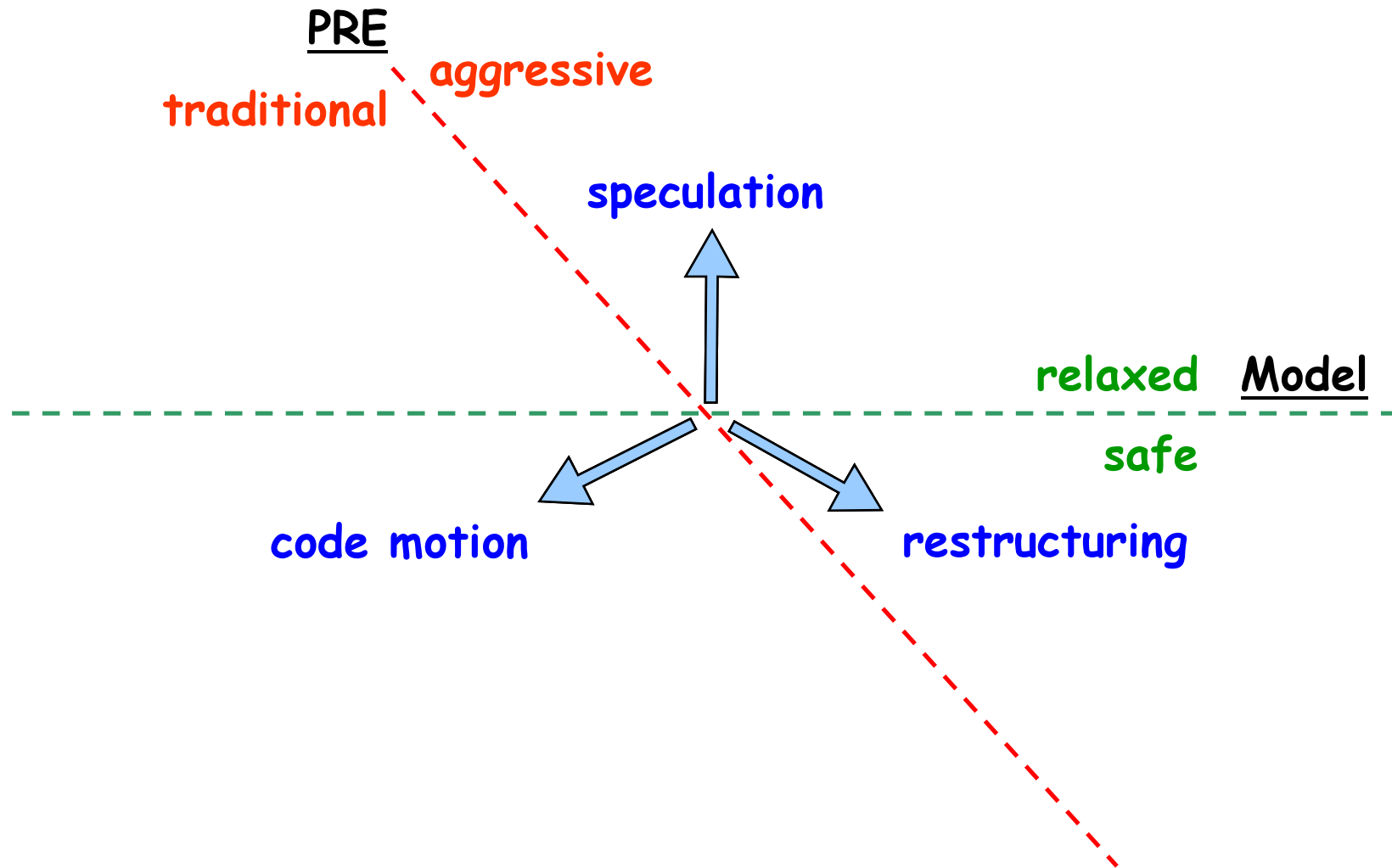
---

16



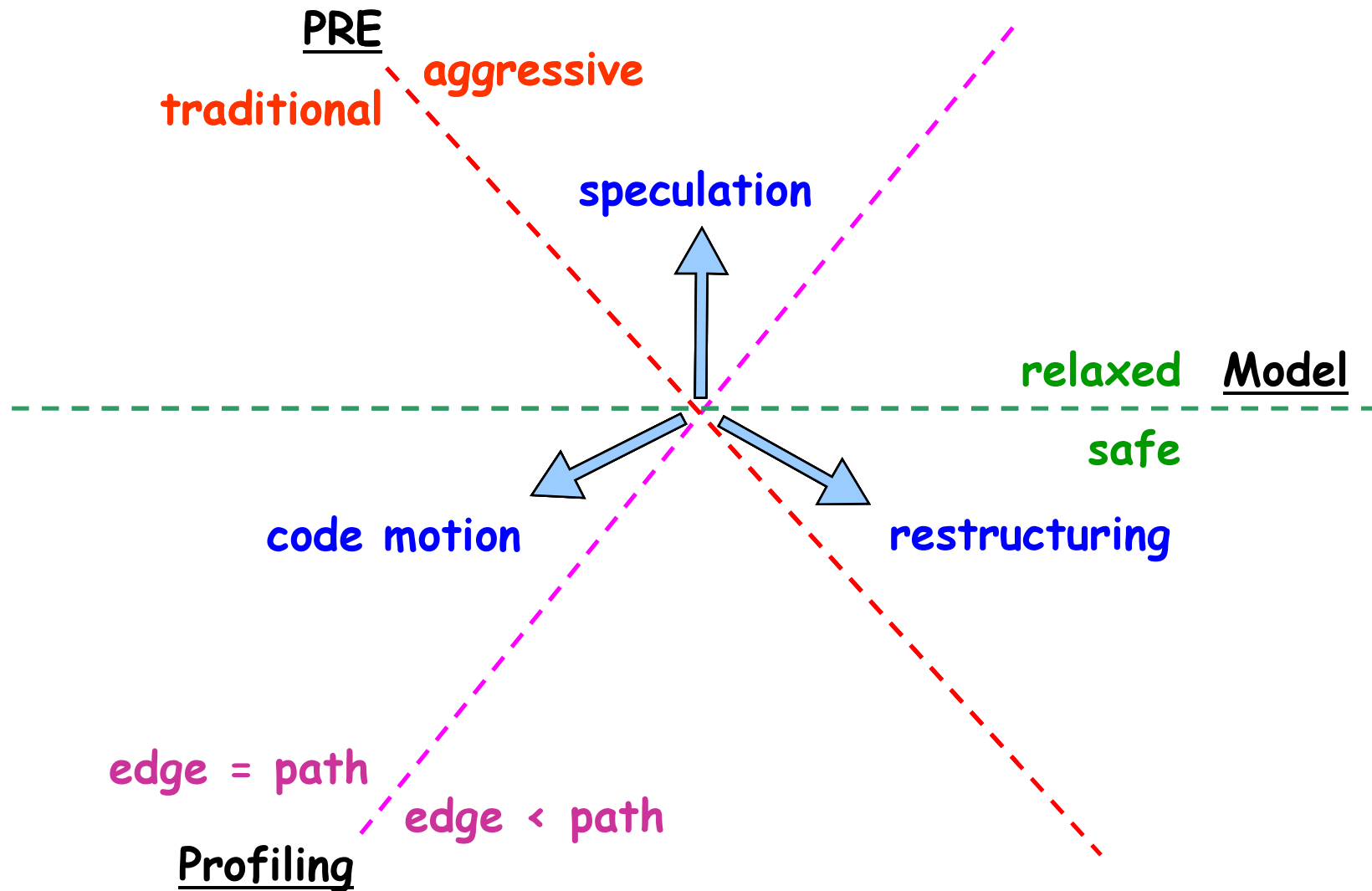
# The three transformations in concert

16



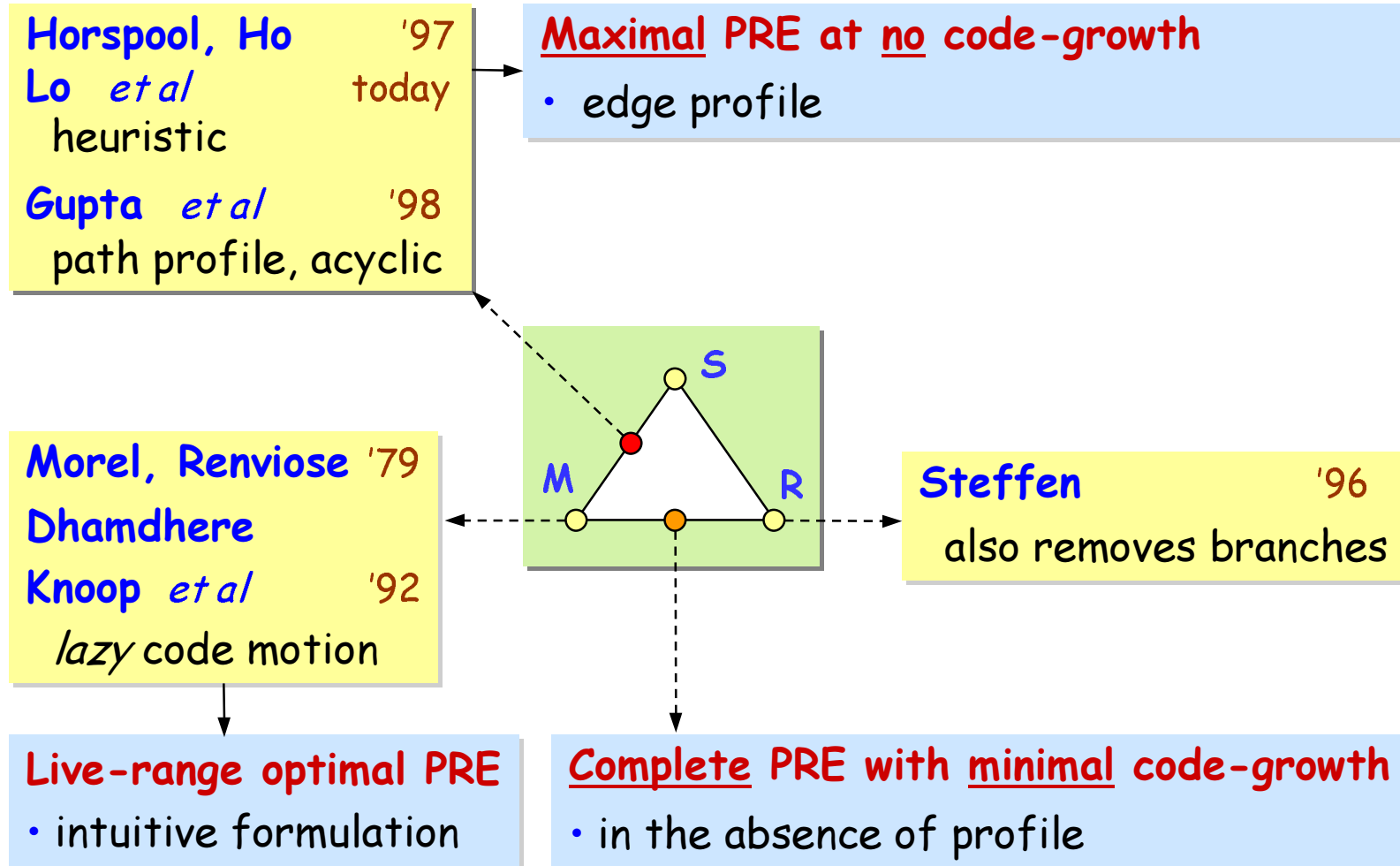
# The three transformations in concert

16



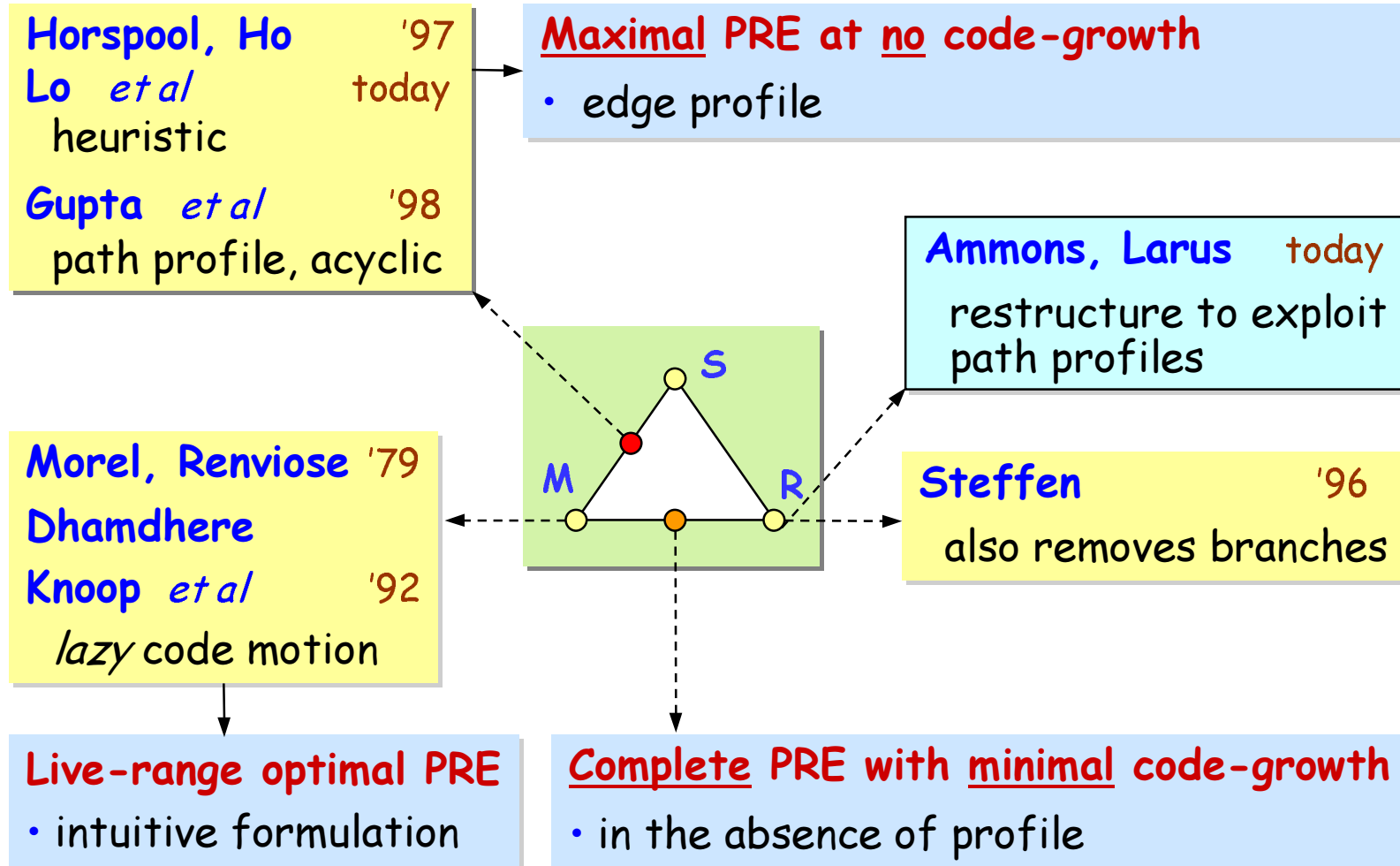
# Related work & summary

17



# Related work & summary

17



# CMP: a versatile PRE abstraction

---

18



## localizes effects of control flow

only examine:

- CMP paths (restructuring)
- CMP entries/exits (speculation)



## small effective size

- small number of entries, executed paths
- combinatorial algorithms become practical