

**Motivation: Von Verifikation über Analyse zur  
Transformation – Optimierung am Beispiel  
“code motion”-basierter Transformationen (Teil 2)**

**Analyse und Verifikation (WS 2006/2007) / 8. Teil (12.12.2006)**

**Jens Knoop**

**Technische Universität Wien**



TECHNISCHE  
UNIVERSITÄT  
WIEN

VIENNA  
UNIVERSITY OF  
TECHNOLOGY

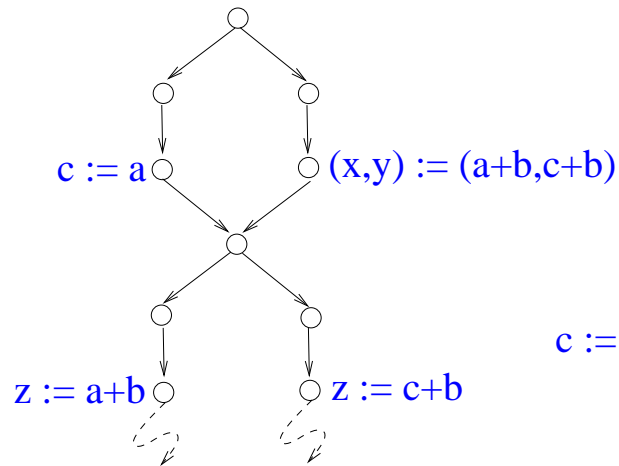
## Optimality Results are quite sensitive!

Three examples to give evidence...

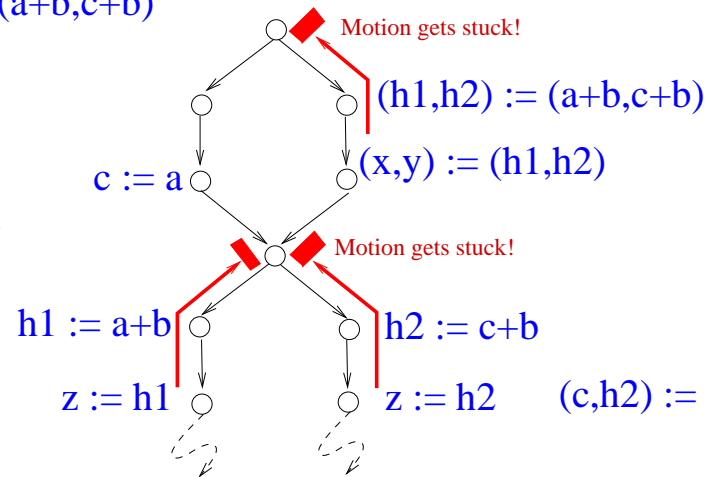
- (I) Code motion vs. code placement
- (II) Interdependencies of (elementary) transformations
- (III) Paradigm dependencies

# (I) Code Motion vs. Code Placement

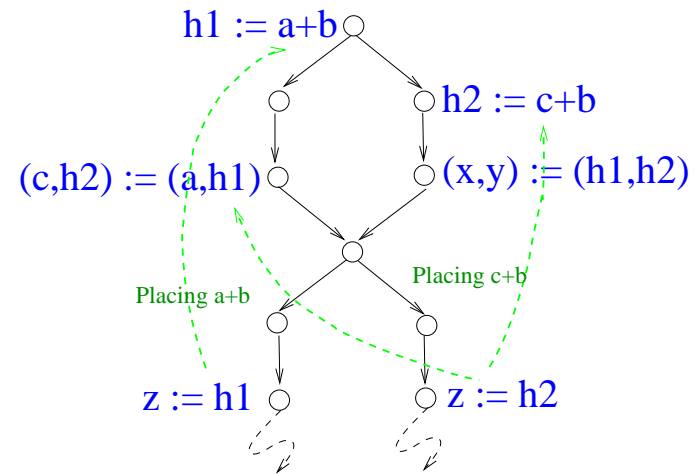
...not just synonyms!



Original Program



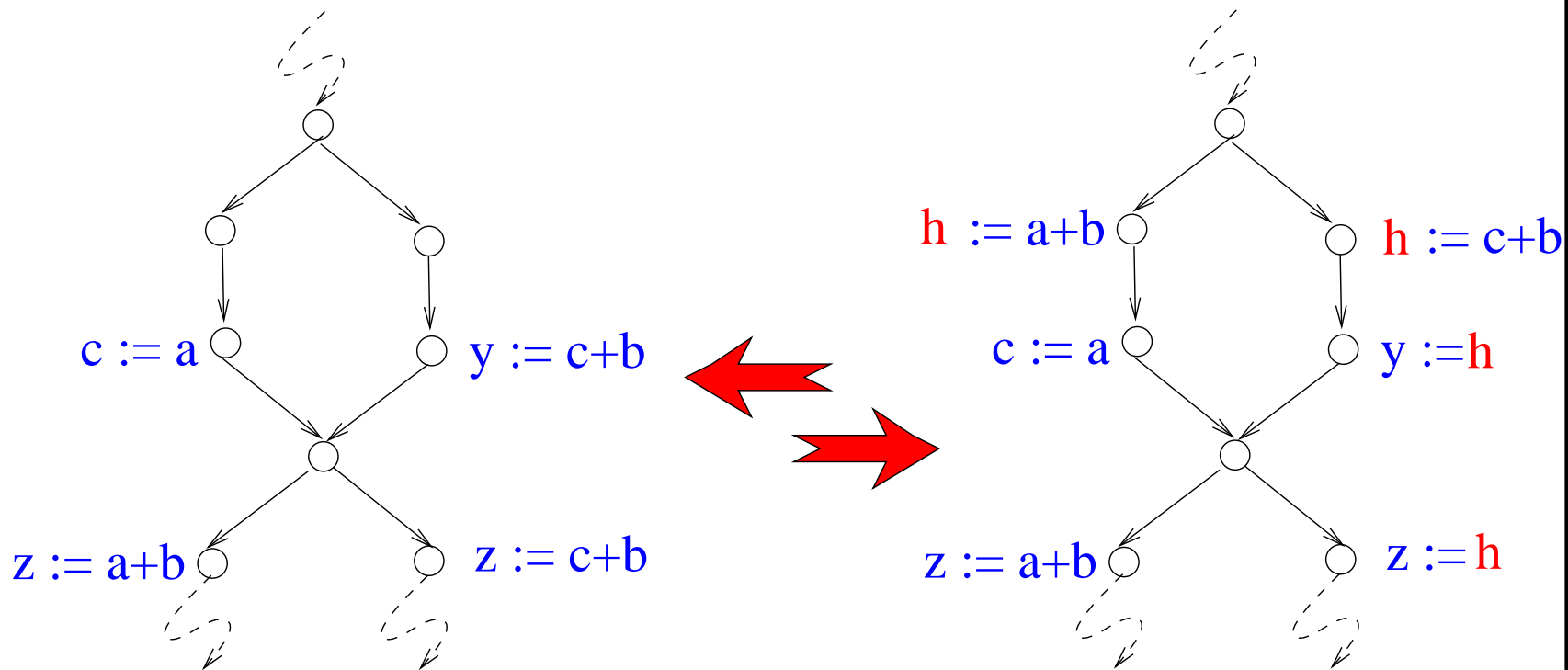
After Sem. Code Motion



After Sem. Code Placement

## Even worse...

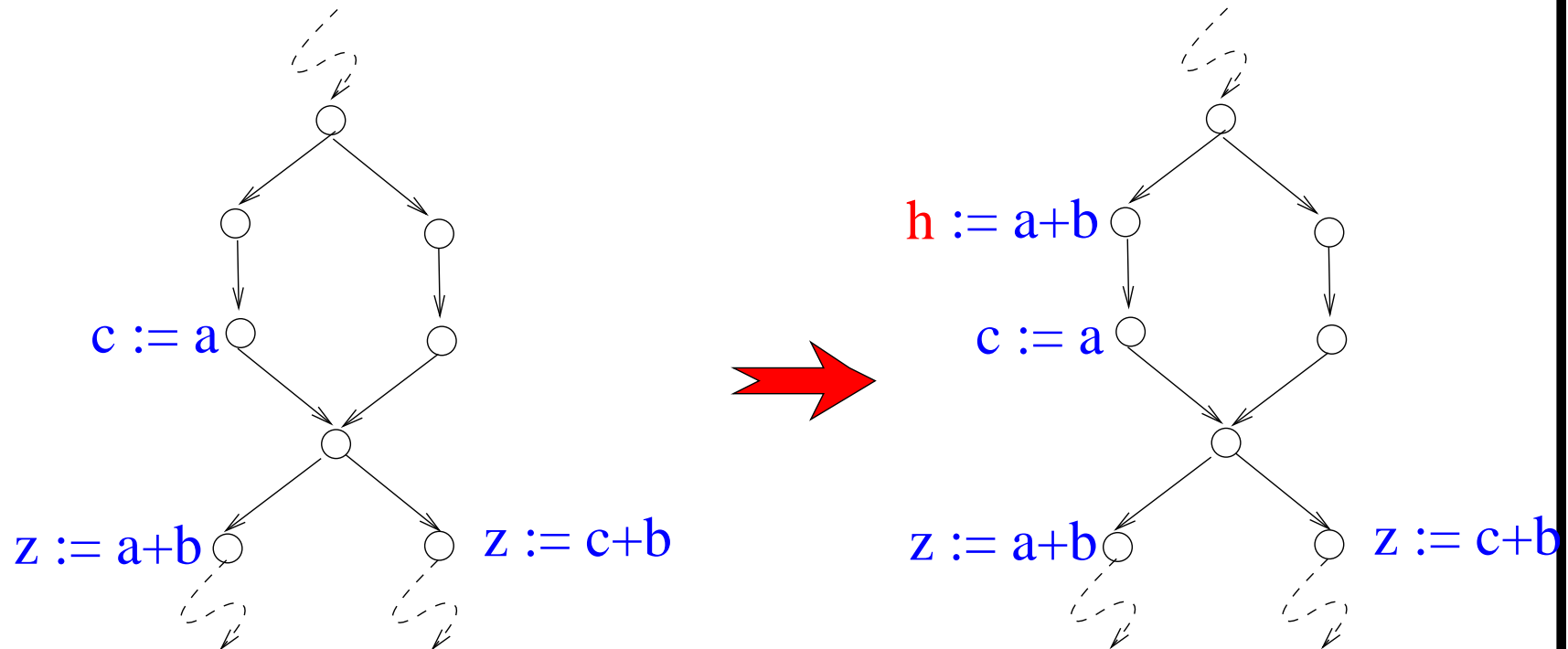
Optimality is lost!



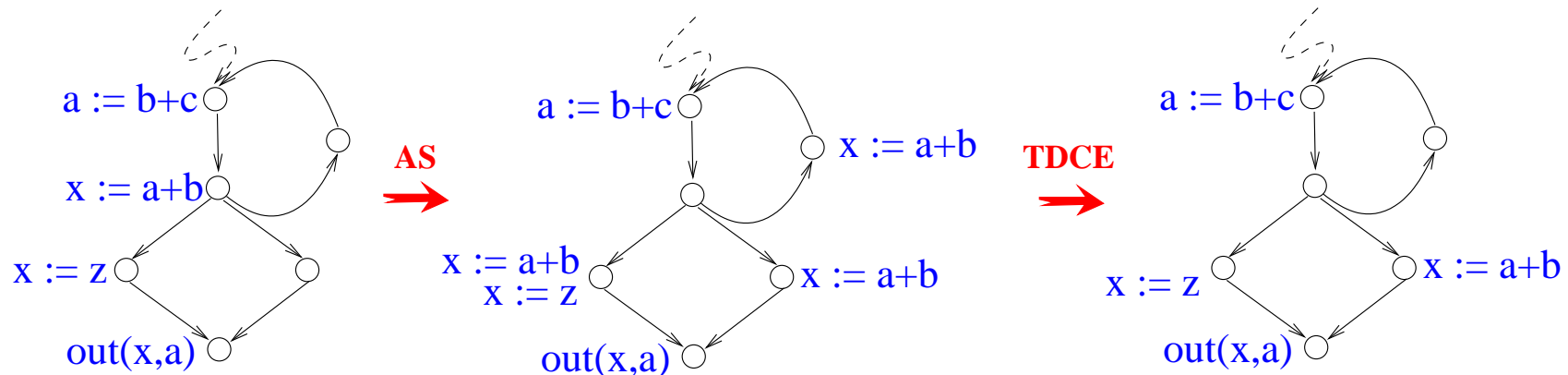
Incomparable!

## Even more worse...

Performance may be lost, when naively applied!



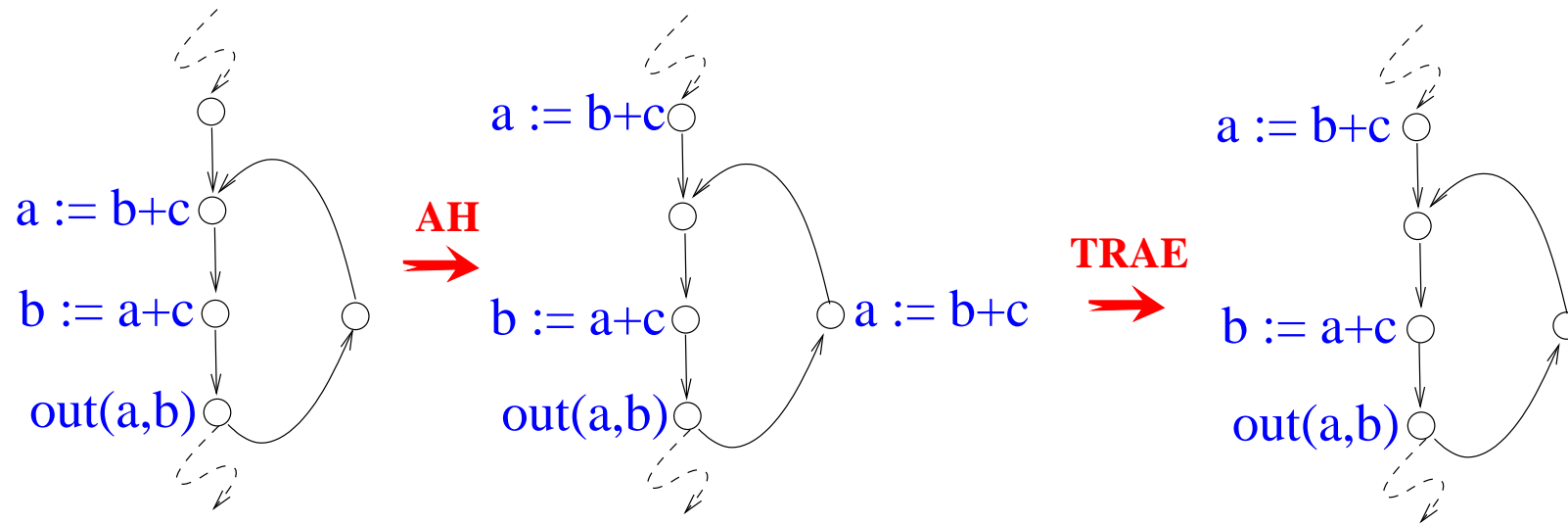
## (II) Interdependencies of Transformations



**...2nd Order Effects!**

~> ...Partial Dead-Code Elimination (PDCE)

## Interdependencies of Transformations



**...2nd Order Effects!**

~> ...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

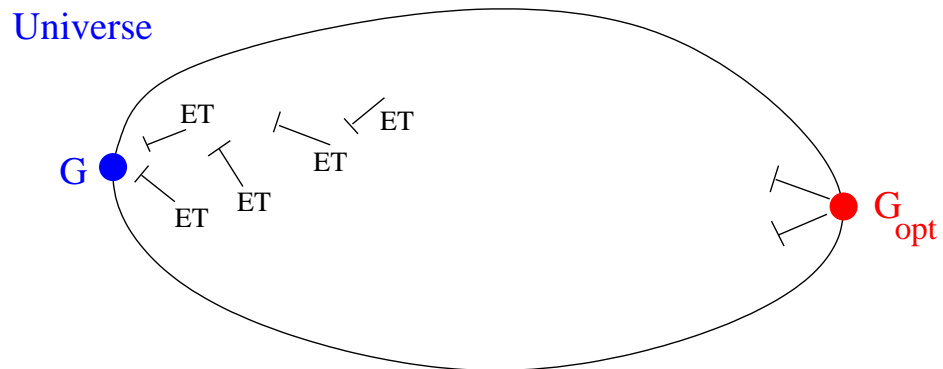
Derivation relation  $\vdash \dots$

- 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



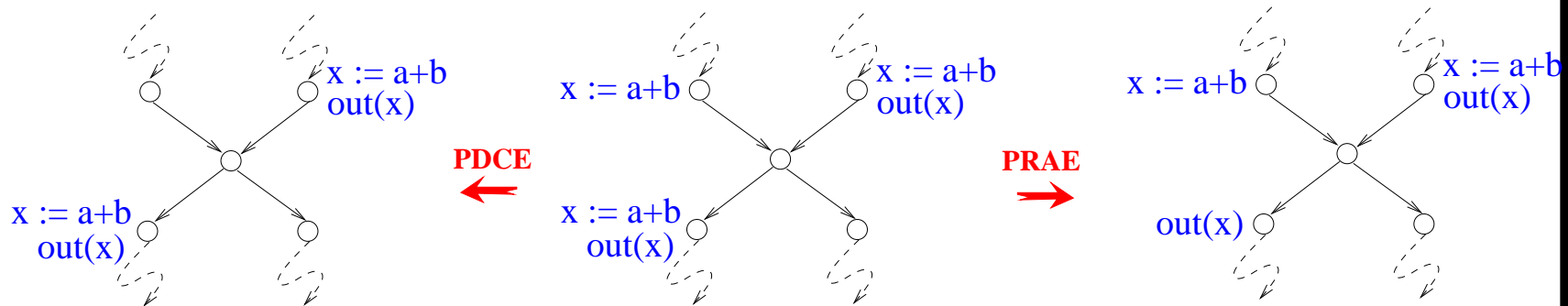
## Consider now...

- Assignment Placement AP

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

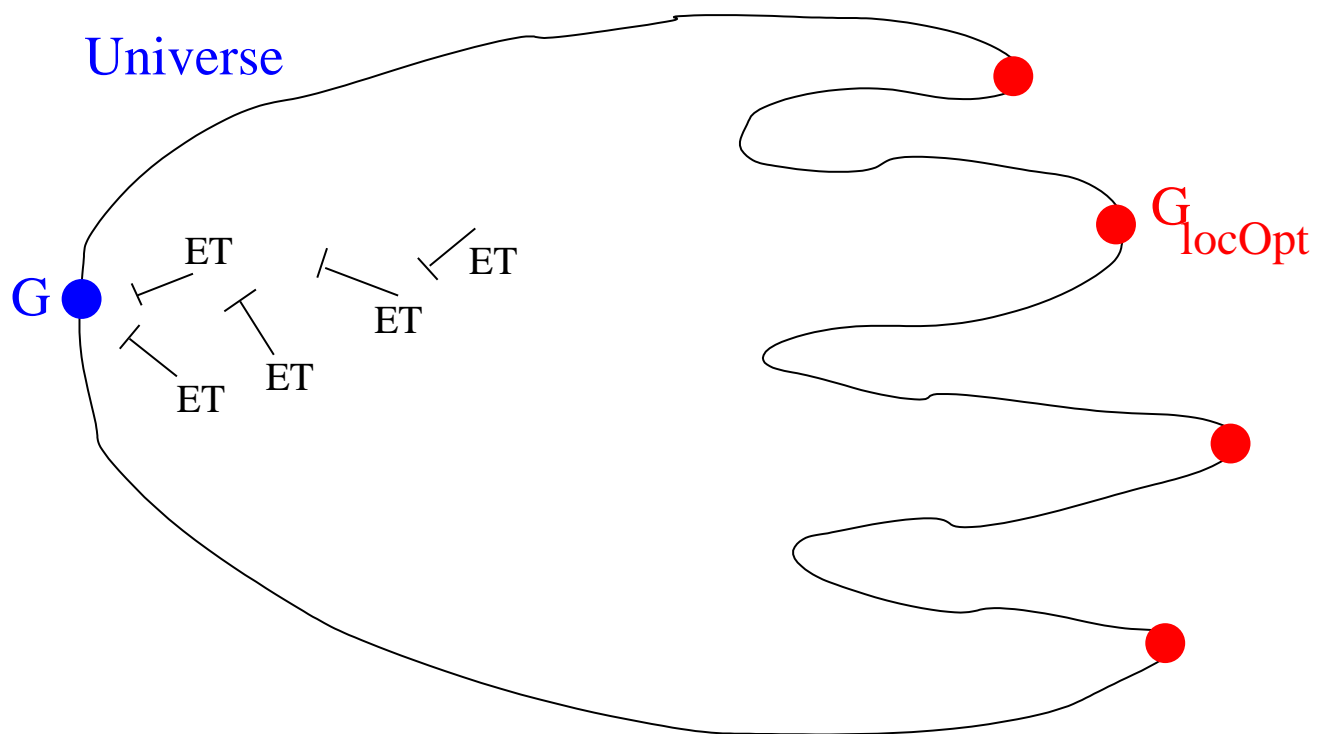
...should be even more powerful!

Indeed, but...



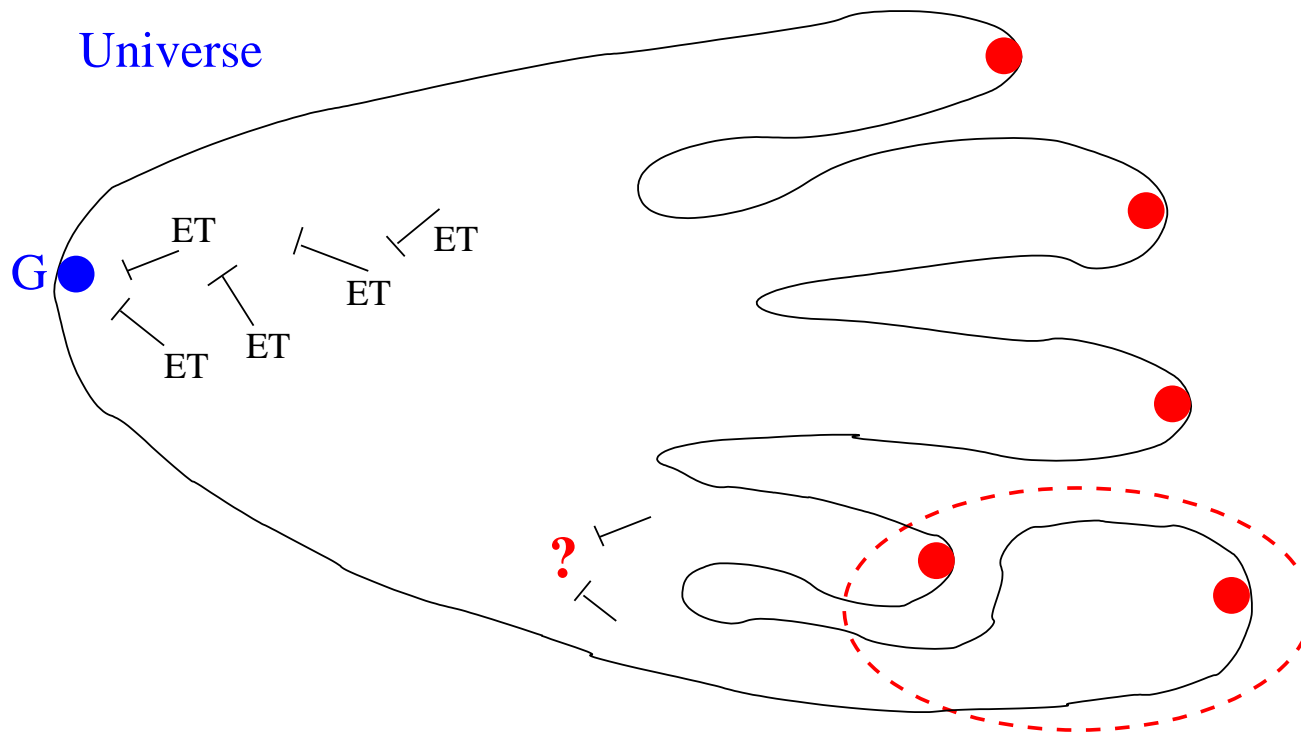
## Confluence...

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

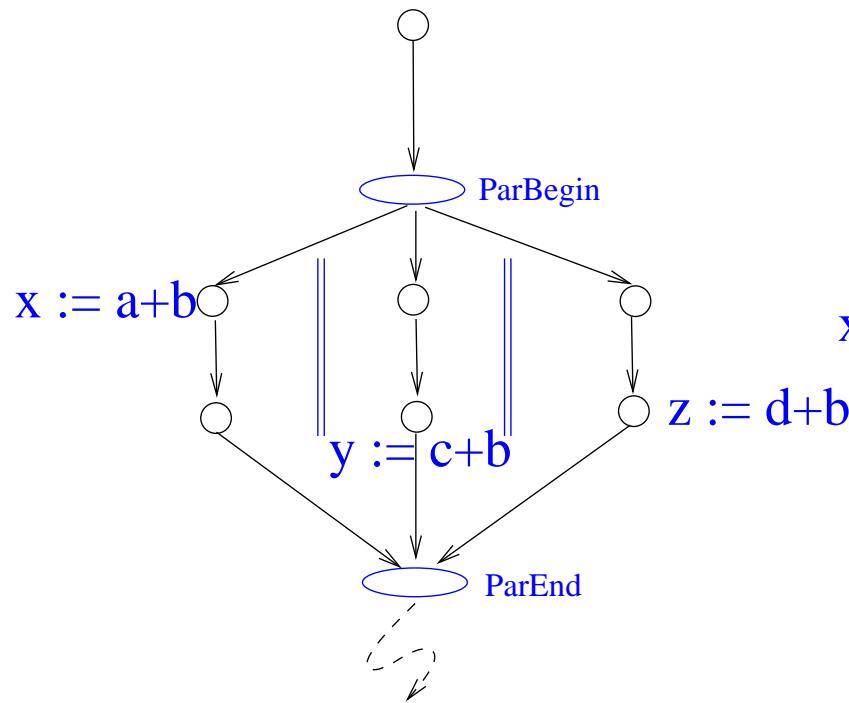


## Even worse...

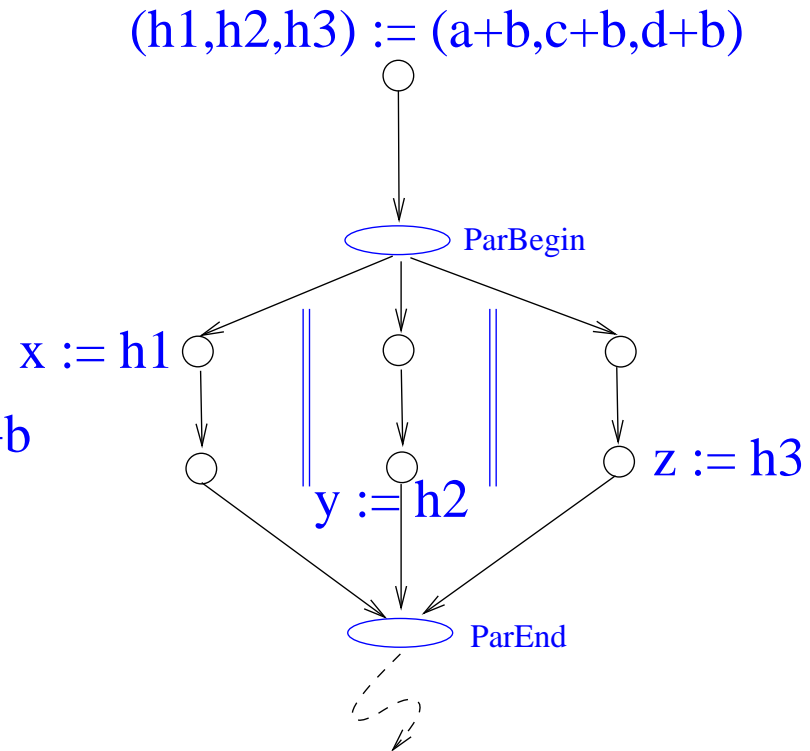
...there are scenarios, where we can end up with universes like



### (III) Paradigm Dependencies



**Original Program**

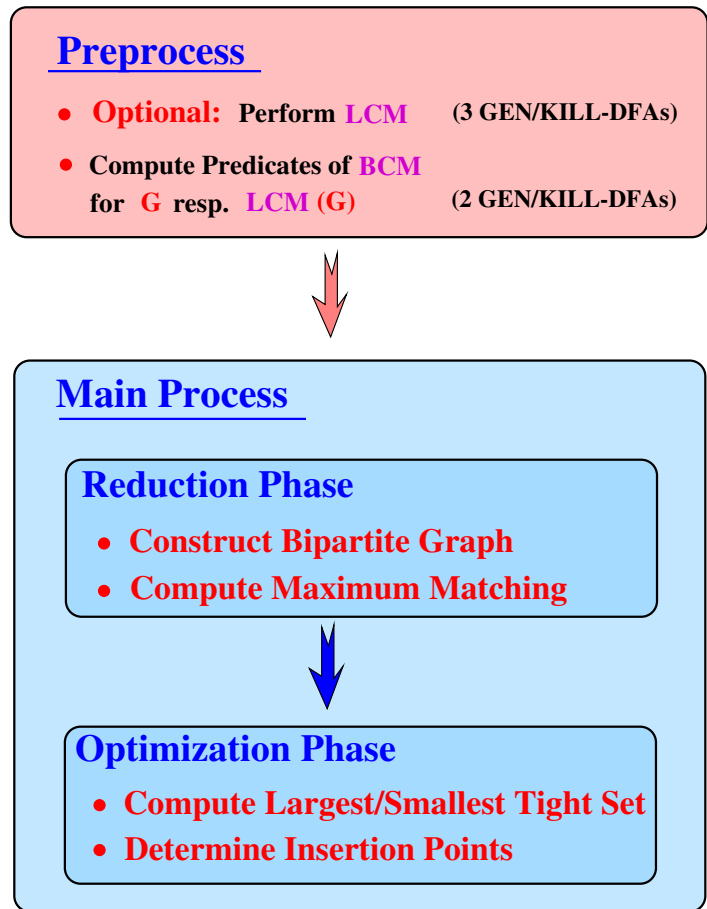


**After Earliestness Transformation**

...a naive transfer of the transformation strategy results here in essentially a sequential program!

# Back to PR(E)E

Recall: Code-size sensitive PRE at a glance...



## The Preprocess in more Detail

This means to consider the transformations of

- Busy Code Motion (BCM)
- Lazy Code Motion (LCM)

in more detail!

Fundamental are...

...the principles of

- Earliestness and
- Latestness

...for placing computations.

## Busy Code Motion (PLDI'92)

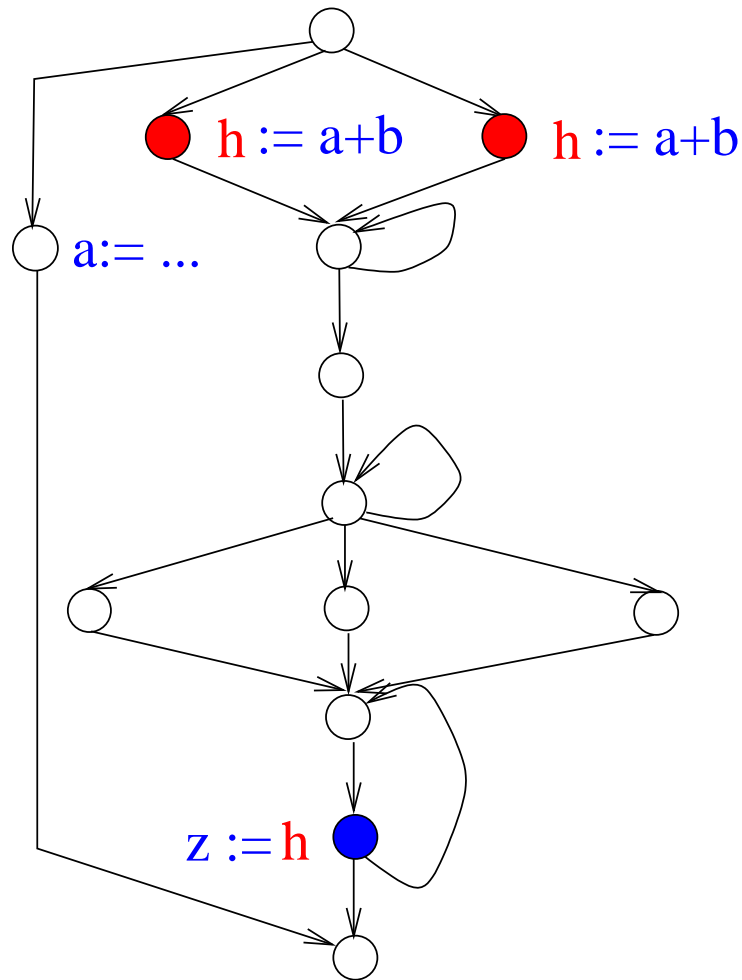
- **Computationally optimal** results can be obtained by
  - ... placing computations **as early as possible**, while maintaining **admissibility** (i.e., semantics & performance)

**Earliestness Principle**



# Earliestness Principle

...yields **computationally optimal** programs.



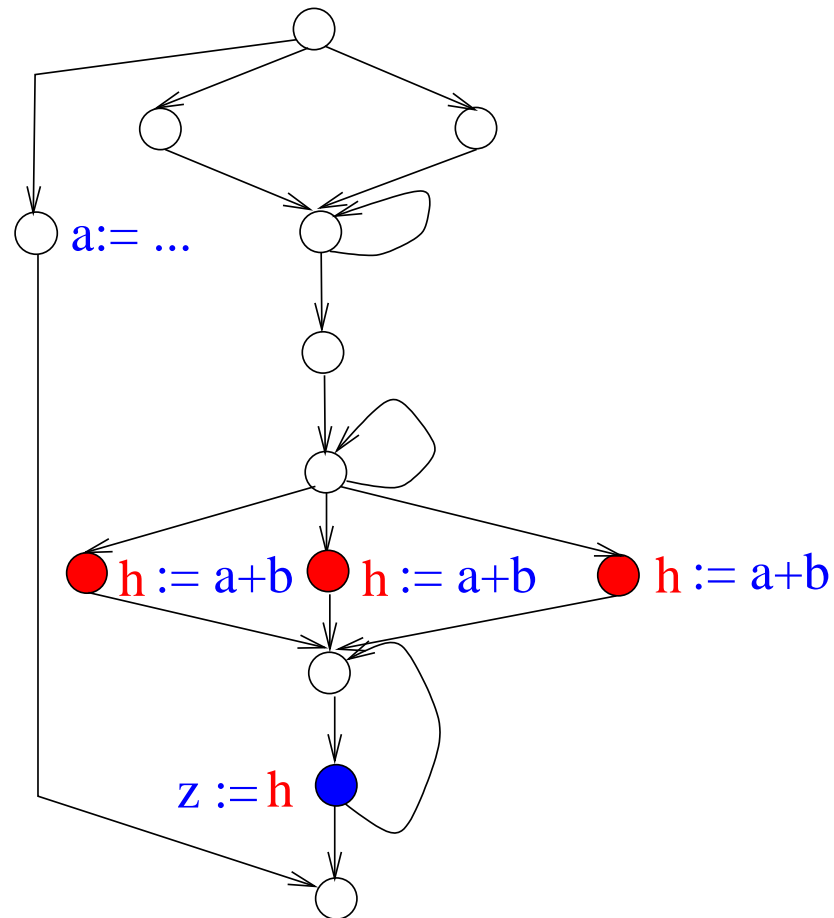
## Lazy Code Motion (PLDI'92)

- **Computationally & lifetime optimal** results can be obtained by
  - ... placing computations **as late as possible**, while maintaining **computational optimality**

**Latestness Principle**

# Latestness Principle

...yields computationally & lifetime optimal programs



## BCM and LCM Algorithmically

- **BCM** relies on only two GEN/KILL-Analyses
  - UpSafety (also known as Availability)
  - DownSafety (also known as Very Busyness)
- **LCM** relies on the two analyses of **BCM** plus one additional GEN/KILL-Analysis
  - Delayability

## How to automatically compute...

...properties such as

- UpSafety, DownSafety, and Delayability?

This will be (one of) the topics of the forthcoming lecture(s).