# Two-Handed Emulation

## How to build non-blocking implementations of complex data-structures using DCAS

written by Michael Greenwald

presented by
Schönegger Andreas
University of Salzburg

2009-06-03

# Übersicht

# Übersicht

# Definition von Non-Blocking

An implementation of a concurrent data structure is non-blocking if we guarantee that at least one process will make progress after a finite number of steps.
By “make progress” we mean that it will complete a high-level operation;
by “finite number of steps” we count the number of primitive operations in the underlying implementation.

# Warum Non-Blocking Synchronization

- Deadlock free
- Fehlertolerant
- Keine Einmischung von Synchronisation und Scheduling

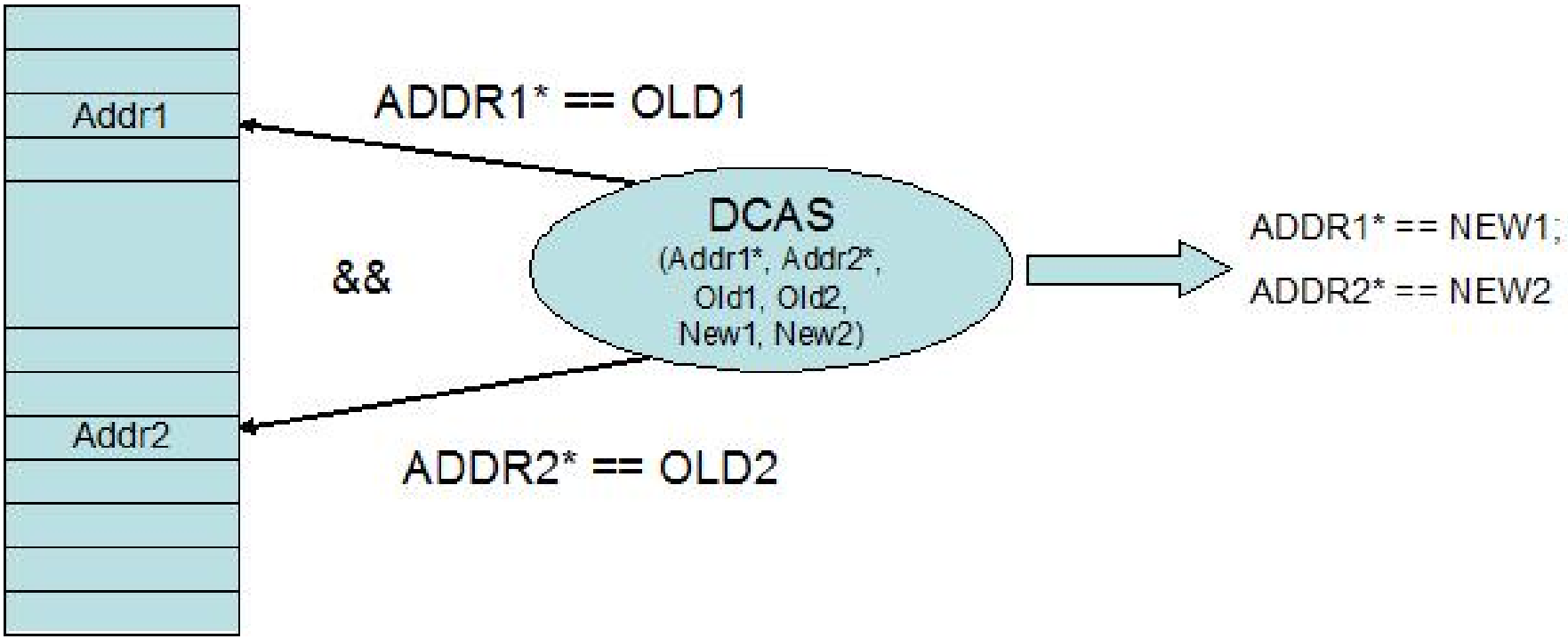
# Probleme bestehender Lösungen

- Benötigen komplexe Algorithmen
- Versteckte Fehler
- Schwer zu Debuggen

# Idee

- Verwendung zusätzlicher Hardware Routinen (DCAS)
  (DCAS nicht neu -> früherer Focus auf spezialfälle)

# DCAS (Double Compare and Swap)



# Operationen

4 Grundoperationen

- Read
- Write
- CAS
- DCAS

# Übersicht

# Was ist Two Handed Emulation

- Transformation
- Protokoll
- non-bolocking Algorithmus

# Two-Handed Emulation Idee

- Führe den Befehl nicht selbst aus sondern registriere den Befehl bei der Datenstruktur
  (Es kann immer nur ein Befehl registriert sein)
- Jeder Prozess kann den Befehl abarbeiten

# Two-Handed Emulation

3 Stufen:

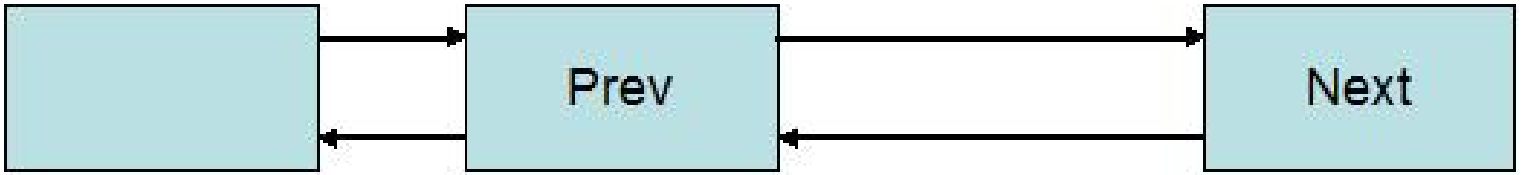
- Neue Operation Registrieren
- Ausführung der sequenziellen Implementierung der Operationen
- Zum Schluss muss der Prozess die Datenstruktur in einen Status bringen, sodass neue Prozesse neue Operationen registrieren können

# Ablauf des Befehlausführens

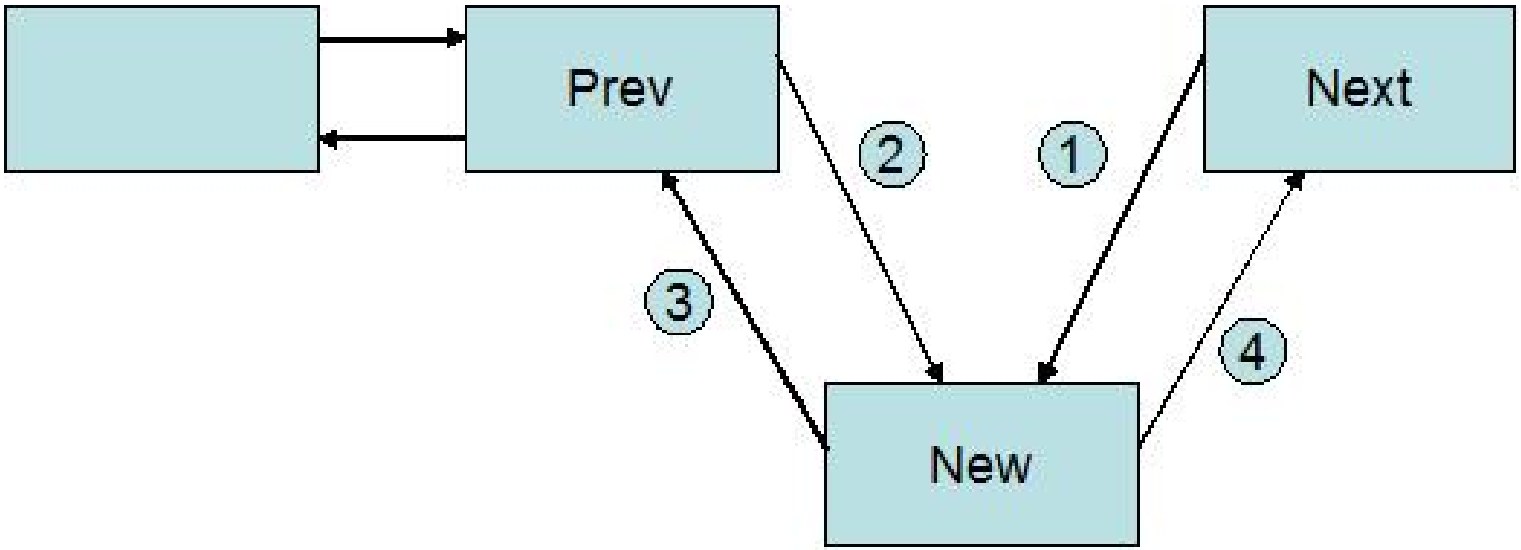
- Speichere Werte mit DCAS Commands
  - setze den Wert nur wenn man noch im selben “Step” ist
  - Setze den Wert nur wenn der exp. OldValue noch drin steht

# Übersicht

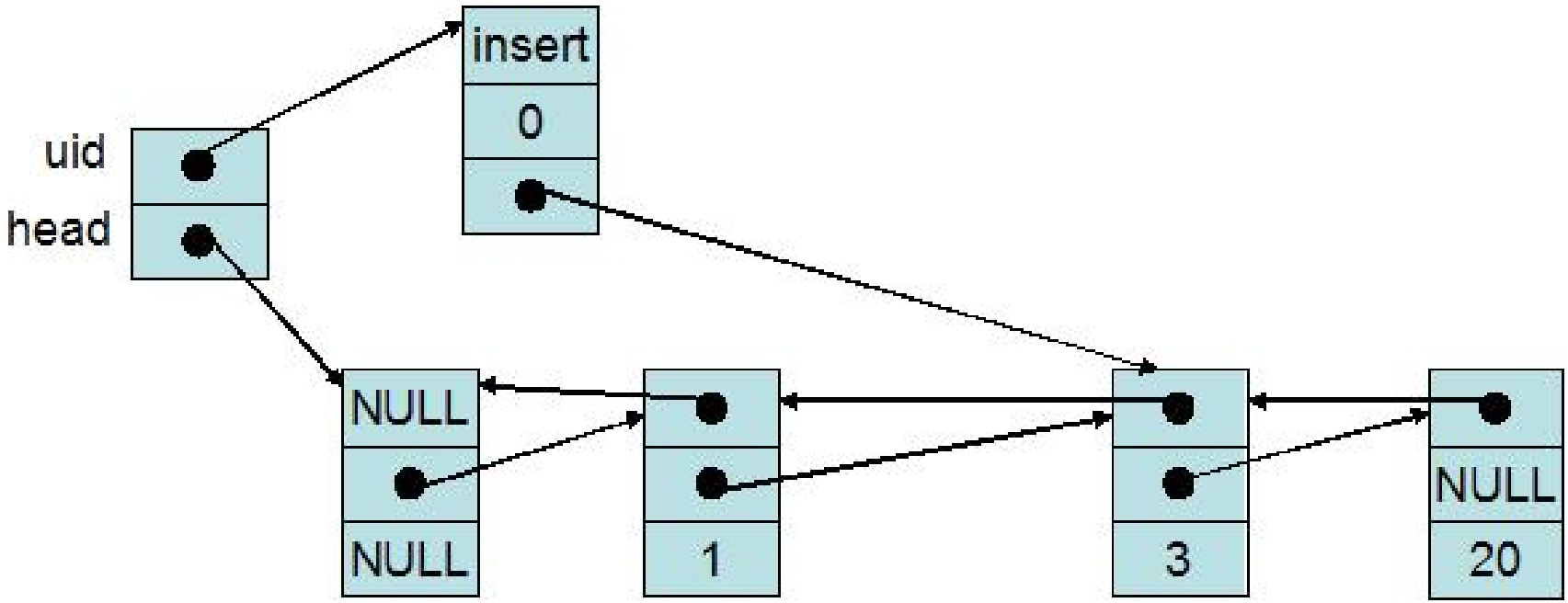
# Normale Doubly Linked List



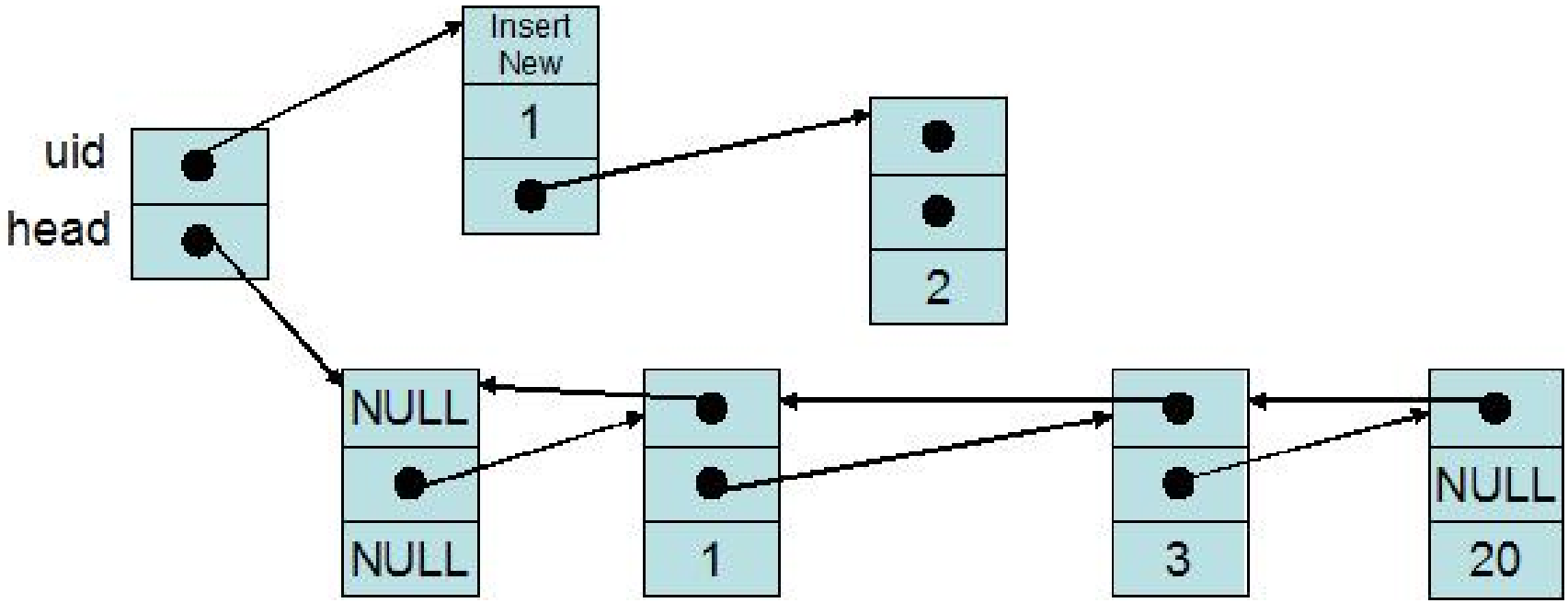

# Normale Doubly Linked List



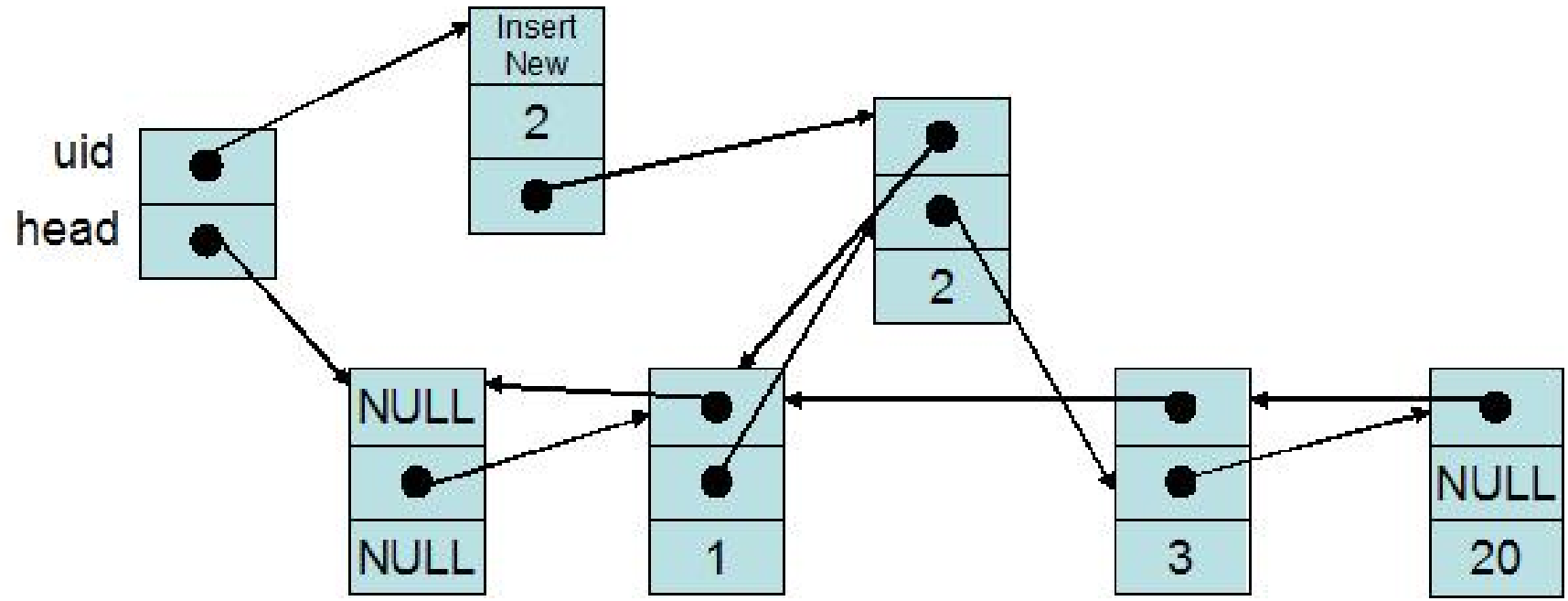
# Non-Blocking Doubly Linked List



# Non-Blocking Doubly Linked List



# Non-Blocking Doubly Linked List

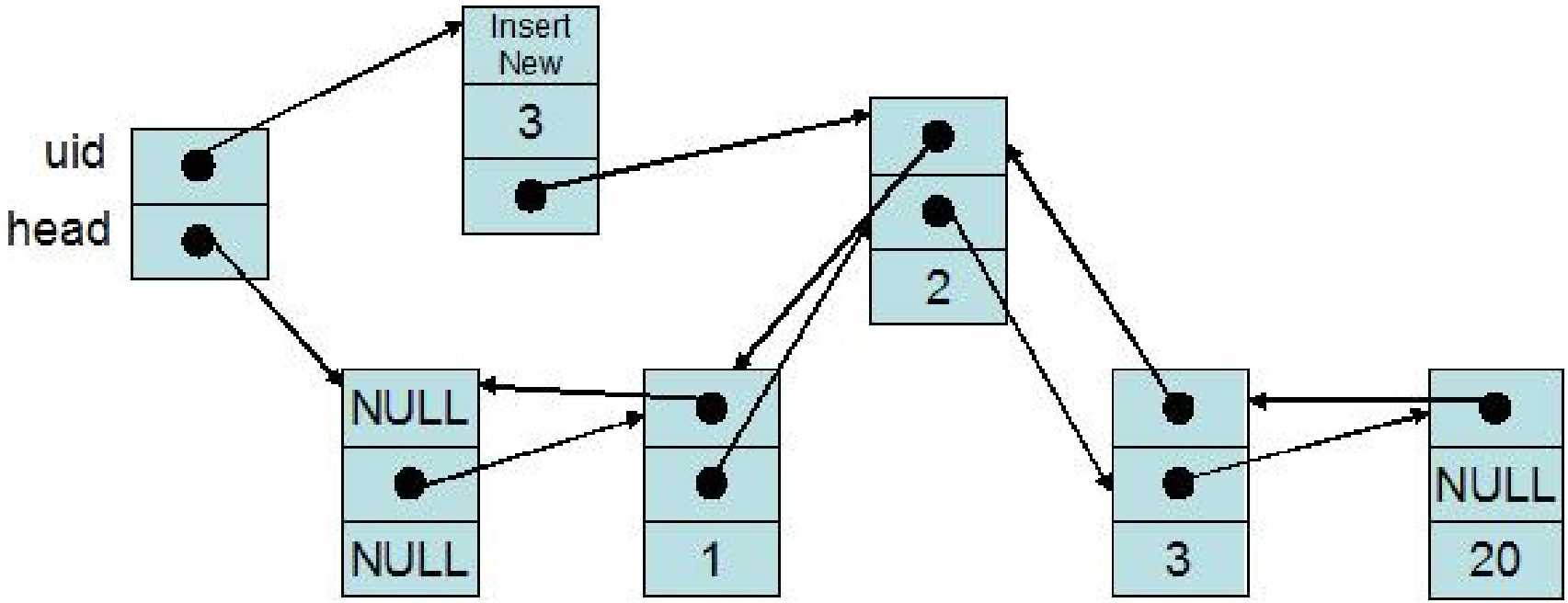



```
#define STEP0(addr, old, new)\
  (DCAS(&(uid->step), (void *)(addr),\
        step,         (void *)(old), \
        step,         (void *)(new)))
```

```
#define STEP1(addr, old, new)\
  (DCAS(&(uid->step), (void *)(addr),\
        step,         (void *)(old), \
        step+1,       (void *)(new)))
```

```
STEP0(&(entry->next), NULL, next);
STEP0(&(entry->prev), NULL, p);
STEP1(&(p->next), p->next, entry);
```

# Non-Blocking Doubly Linked List



# Schwäche von Two Handed Emulation

- Fehlende Parallelität durch die sequentielle Abarbeitung (kann in speziellen Fällen durch Tricks abgefangen werden)
- Keine lokalen Variablen erlaubt
- In einigen Fällen können die Kosten für Non-Blocking zu hoch sein
- Kosten von CAS und DCAS sind sehr hoch
- Code min. so komplex wie Ausgangssituation