Kapitel 10: Laufzeitsystem

Aufgabe

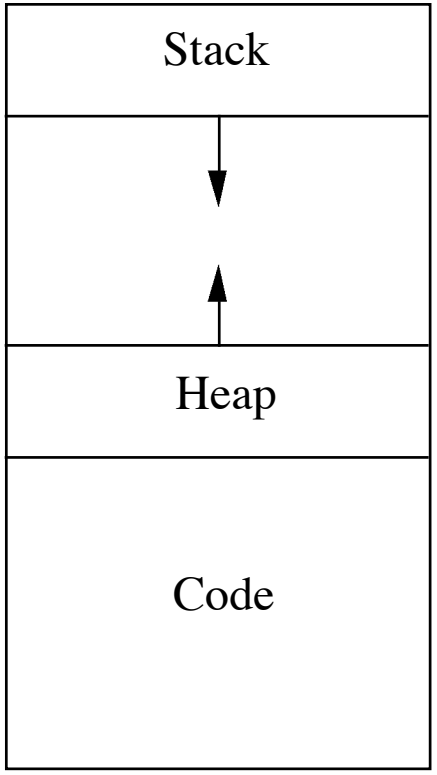
Speicherverwaltung

Themen

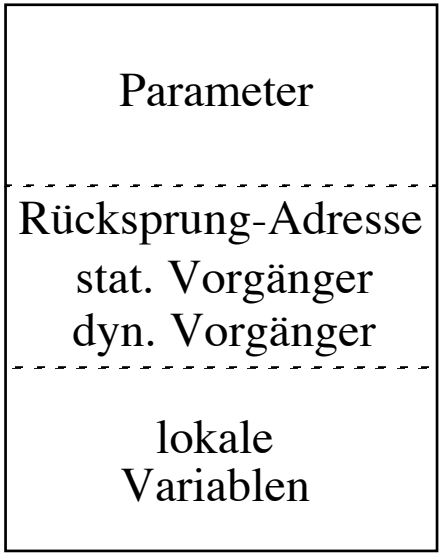
- Stack-verwaltung
- Dynamische Objekte
- Heap-Verwaltung

Speicherbedarf zur Laufzeit

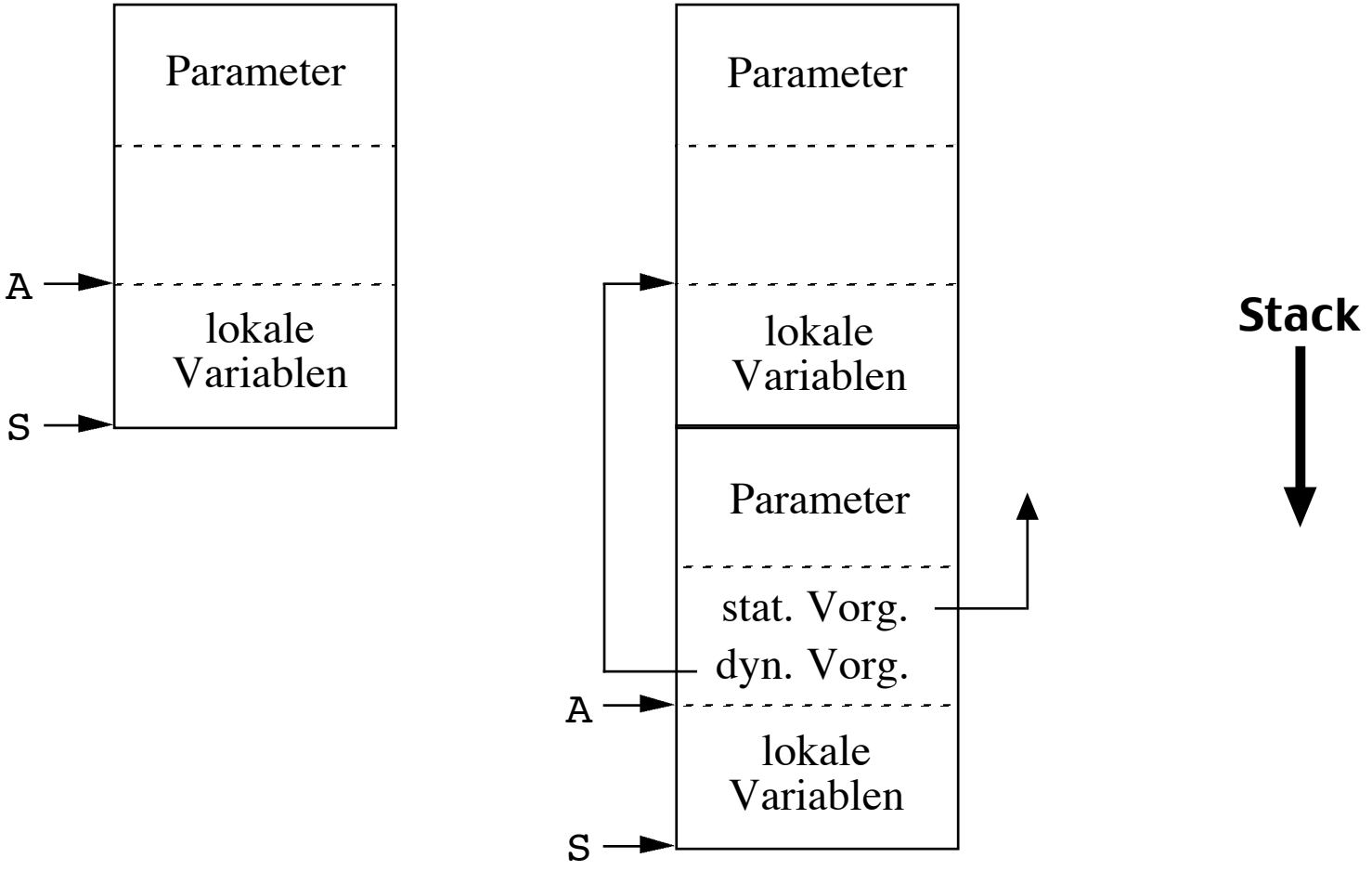
Programm



Activation Record (Frame) eines Prozedur–Aufrufs

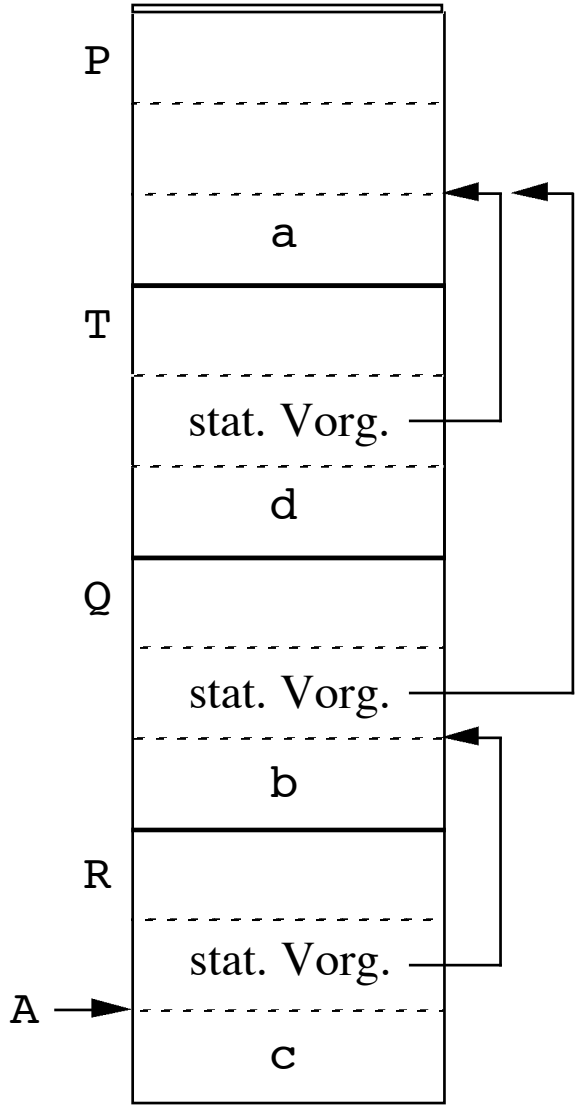


Prozedur–Aufruf



Geschachtelte Prozeduren (Pascal/Modula)

```
PROGRAM P
    a:REAL
    PROCEDURE Q
        b:REAL
        PROCEDURE R
            c:REAL
            (*)
        END
        ->R
    END
    PROCEDURE T
        d:REAL
        ->Q
    END
    ->T
END
```



Statische Kette

... ist die Zeigerkette der statischen Vorgänger
Die Elemente zeigen auf die ARs der statisch umschließenden Prozeduren

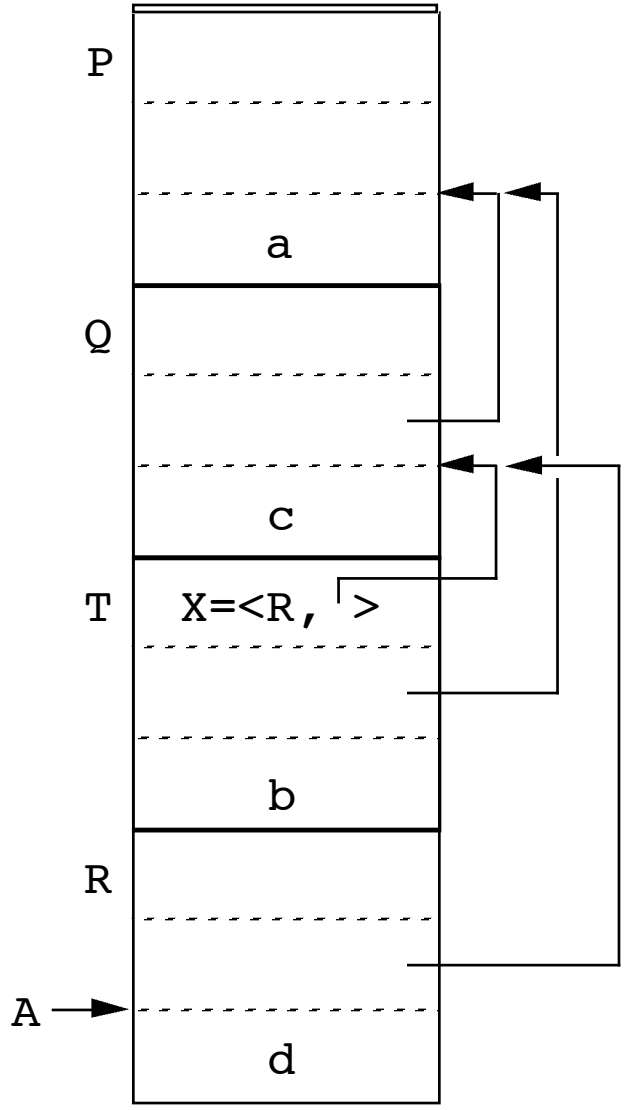
Die Anzahl der Elemente ist zu jedem Zeitpunkt der Ausführung gleich der statischen Schachtelungstiefe des aktuellen Programmpunkts

Nichtlokale Variablen werden durch n Schritte entlang der stat. Kette gefunden
n = Differenz der Schachtelungstiefen von Benutzung und Deklaration

Der statische Vorgänger einer aufgerufenen Prozedur wird ebenso gefunden
n = Differenz der Schachtelungstiefen von Aufruf und Deklaration

Prozedur als Parameter (Pascal)

```
PROGRAM P
    a:REAL
    PROCEDURE T(X)
        b:REAL
        ->X
    END
    PROCEDURE Q
        c:REAL
        PROCEDURE R
            d:REAL
            (*)
        END
        ->T(R)
    END
    ->Q
END
```

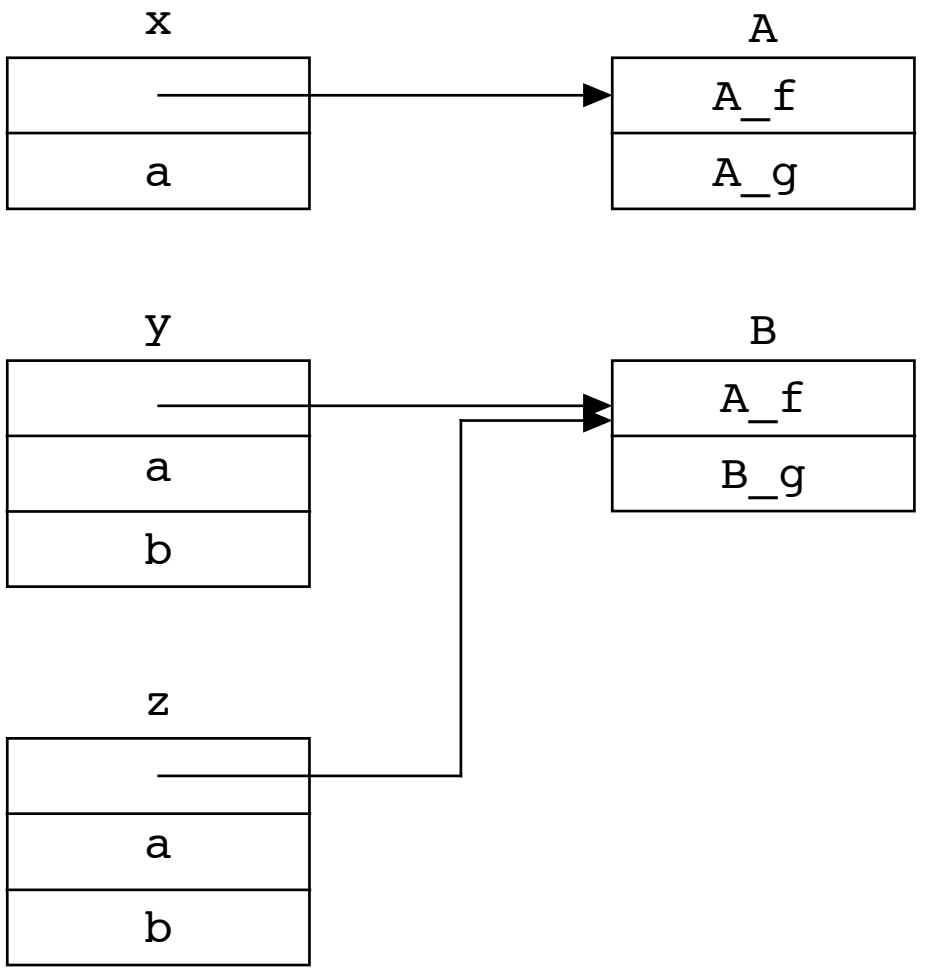


Dynamische Objekte und Methoden

Programm

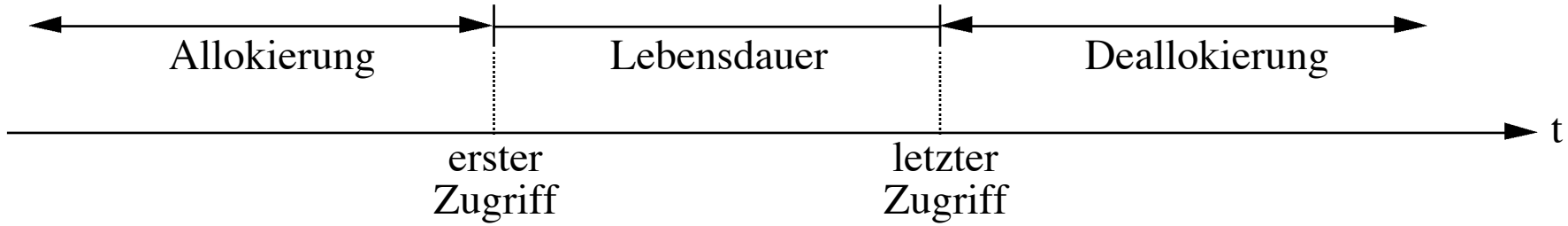
```
class A
   var a
   method f()
   method g()

class B extends A
   var b
   method g()

var x = new A
var y = new B
var z = new B

var v: A

v = ...

v.g()
```

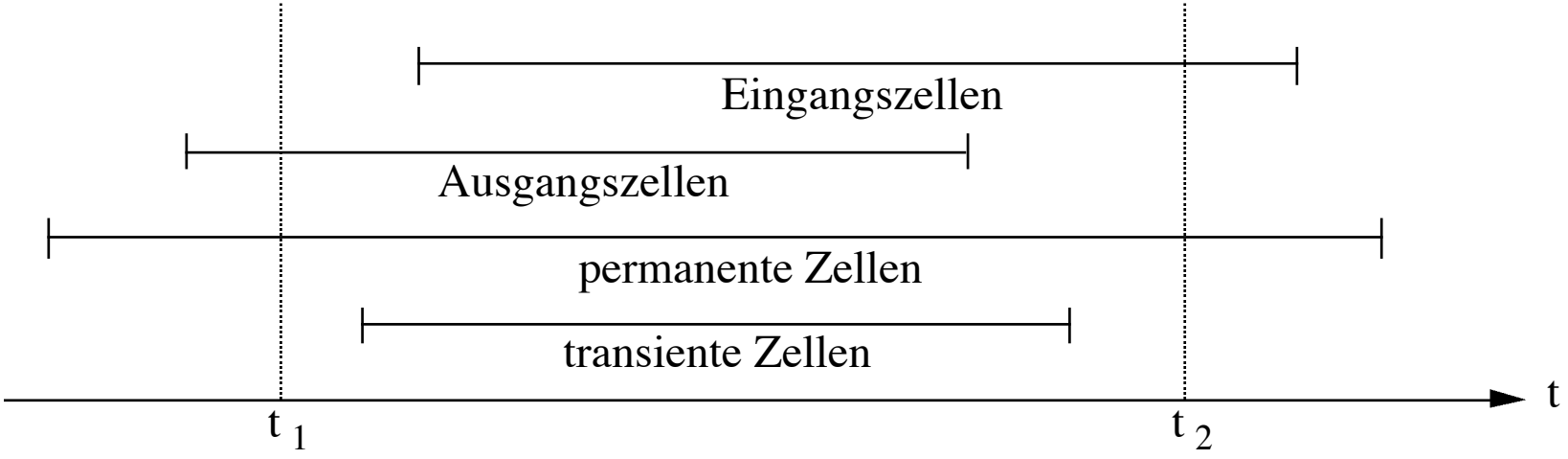
Objekte **Deskriptoren**



Heap–Verwaltung

Lebensdauer von Heap–Zellen





Teilaufgaben der Heapverwaltung

Ziel: Rückgewinnung von freiem Speicher
Voraussetzung: Zellen verschiebbar (d.h. keine Zeigerarithmetik wie in C)

a) Feststellung der benötigten Zellen (Referenzierbarkeit)

b) Reorganisation der benötigten Zellen

a) Feststellung der benötigten Zellen

1. Markierungsverfahren

Referenzierte Zellen in Zeitabständen rekursiv markieren
(pro Zelle ein Markierungsbit)

Vorteil: Guter Durchsatz
Nachteil: Schlechte maximale Antwortzeit (diskontinuierliche Verzögerung)
Aufwand: abhängig von Nperm

2. Referenzzähler

Referenzen auf eine Zelle laufend mitzählen
(pro Zelle ein Referenzzähler)

Vorteil: Gute maximale Antwortzeiten (kontinuierliche Verzögerung)
Problem: Zyklische Strukturen werden nicht erkannt
Aufwand: unabhängig von Nperm

b) Reorganisation der benötigten Zellen

1. Ordnungsbewahrende Kompaktierer

Anwendung: Zellen verschiebbar, aber Ordnung muß erhalten bleiben
Methode: Benötigte Zellen in freie Bereiche verschieben, Zeiger anpassen

2. Ordnungszerstörende Kompaktierer (Kopierer)

Anwendung: Zellen verschiebbar, Ordnung muß nicht erhalten bleiben
Methode: 2 Speicherbereiche (Arbeitsspeicher/Freispeicher)

- Benötigte Zellen werden im Arbeitsspeicher sequentiell allokiert
- Bei Überlauf alle benötigten Zellen in den Freispeicher kopieren
- Freispeicher wird zum Arbeitsspeicher und umgekehrt

Vorteil: Effizienz (Referenzierbarkeit und Reorganisation in einem Durchlauf)
Nachteil: Doppelter Speicherbedarf
Aufwand: abhängig von Nperm –> Optimierung durch "Generationen"