10. AuD Tafelübung T-C3

Simon Ruderich

12. Januar 2011

$\mathcal{O}$-Kalkül

„$n \to \infty$“

- n (und Co.) wird als sehr groß betrachtet („$n \to \infty$“)
- keine Fallunterscheidung nötig, z. B. für $n \leq 5$, $n > 5$
- Ausnahme: z. B. $\mathcal{O}(n^a + n^b) = \mathcal{O}(n^{\max(a,b)})$

Vollständige Induktion

z. Z. $\sum_{i=1}^{n} 4i = 2n(n+1)$

$$\text{IA: } n = 1 : \sum_{i=1}^{1} 4i = 4,\ 2 \cdot 1(1+1) = 4$$

$$\text{IV: } \sum_{i=1}^{n} 4i = 2n(n+1)$$

$$\text{IS: } n \to n+1 : \sum_{i=1}^{n+1} 4i = 4 \cdot \sum_{i=1}^{n} i + 4(n+1) =$$

$$4 \cdot \frac{n(n+1)}{2} + 4n + 4 = 2(n+1)(n+1+1)$$

Keine korrekte Induktion, da IV nie verwendet wurde!

Vollständige Induktion

Bitte nicht als Gleichungen!

$$\sum_{i=1}^{n+1} 4i = 2(n+1)(n+1+1)$$

$$\sum_{i=1}^{n} 4i + 4(n+1) = 2(n+1)(n+1+1)$$

$$2n(n+1) + 4(n+1) = 2(n+1)(n+1+1)$$

„beide Seiten sind gleich, Aussage gilt“

Sondern eine lange Gleichung.

$$\sum_{i=1}^{n+1} 4i = \sum_{i=1}^{n} 4i + 4(n+1) = \ldots = 2(n+1)(n+1+1)$$

Exceptions

Welche Exception wirft dieser Code?

```
public static float f(float x) {
    // ...
    return x/0;
}
```

Exceptions

Welche Exception wirft dieser Code?

```
public static float f(float x) {
    // ...
    return x/0;
}
```

Keine, da Float-Division. Ergebnis ist $\pm\infty$.

`ArithmeticException` nur bei Integer-Division!

Abstrakter Datentyp (ADT)

Allgemein

- beschreibt das Verhalten einer Schnittstelle
- keine konkrete Implementierung
- unabhängig von der Programmiersprache
- ermöglicht es Funktionen zu nutzen ohne deren Implementierung zu kennen

Eigenschaften

- jeder ADT besteht aus Signatur und Axiomen
- Signatur beschreibt die Namen und Argumente der „Methoden“ des ADTs
- Axiome geben das Verhalten des ADTs (der „Methoden“) an

ADT Liste

Signatur

- besteht aus ({Primär-, Hilfs-})Konstruktoren, Projektionen
- Konstruktoren erzeugen neue Objekte des ADTs
- Projektionen liefern Informationen über den ADT

Primärkonstruktoren:

- create: $\rightarrow$ Liste
- append: Object $\times$ Liste $\rightarrow$ Liste

(Hilfs-)Konstruktor:

- tail: Liste $\rightarrow$ Liste

Projektionen:

- head: Liste $\rightarrow$ Object
- length: Liste $\rightarrow$ Integer

ADT Liste

Konstruktoren

Primärkonstruktoren	erzeugen alle möglichen Objekte
(Hilfs-)Konstruktoren	erzeugen ebenfalls Objekte, aber Primärkonstruktoren sind einfacher

Projektionen

- bilden auf ein anderes ADT ab (und z. B. Zahlen, etc.) nicht auf Objekte des aktuellen ADT

ADT Liste

Axiome

- geben Verhalten des ADT an
- bestehen pro „Methode“ meist aus Basisfall und Rekursionsfall, ggf. mit Unterscheidung („if-Abfrage“)

Axiome des ADT Liste

head(create) fehlt, ist also nicht spezifiziert

A1 head(append(x, L)) = x

tail(create) analog

A2 tail(append(x, L)) = L

A3 length(create) = 0

A4 length(append(x, L)) = 1 + length(L)

ADT Liste (erweitert)

Signatur

- contains: Object $\times$ Liste $\rightarrow$ Boolean
- copy: Liste $\rightarrow$ Liste
- delete: Object $\times$ Liste $\rightarrow$ Liste
- deleteAll: Object $\times$ Liste $\rightarrow$ Liste
- prepend: Object $\times$ Liste $\rightarrow$ Liste
- addAll: Liste $\times$ Liste $\rightarrow$ Liste

Axiome

...

ADT Liste (erweitert)

Signatur

- contains: Object $\times$ Liste $\rightarrow$ Boolean
- copy: Liste $\rightarrow$ Liste
- delete: Object $\times$ Liste $\rightarrow$ Liste
- deleteAll: Object $\times$ Liste $\rightarrow$ Liste
- prepend: Object $\times$ Liste $\rightarrow$ Liste
- addAll: Liste $\times$ Liste $\rightarrow$ Liste

Axiome

A1: $\text{contains}(x, \text{create}) = \text{false}$

A2: $$\text{contains}(x, \text{append}(y, L)) = \begin{cases} \text{true} & \text{falls } x = y \\ \text{contains}(x, L) & \text{sonst} \end{cases}$$

ADT Liste (erweitert)

Signatur

- contains: Object $\times$ Liste $\rightarrow$ Boolean
- copy: Liste $\rightarrow$ Liste
- delete: Object $\times$ Liste $\rightarrow$ Liste
- deleteAll: Object $\times$ Liste $\rightarrow$ Liste
- prepend: Object $\times$ Liste $\rightarrow$ Liste
- addAll: Liste $\times$ Liste $\rightarrow$ Liste

Axiome

$$\text{A3: } \text{copy}(L) = L$$

ADT Liste (erweitert)

Signatur

- contains: Object $\times$ Liste $\rightarrow$ Boolean
- copy: Liste $\rightarrow$ Liste
- delete: Object $\times$ Liste $\rightarrow$ Liste
- deleteAll: Object $\times$ Liste $\rightarrow$ Liste
- prepend: Object $\times$ Liste $\rightarrow$ Liste
- addAll: Liste $\times$ Liste $\rightarrow$ Liste

Axiome

A4: $\text{delete}(x, \text{create}) = \text{create}$

A5: $\text{delete}(x, \text{append}(y, L)) = \begin{cases} L & \text{falls } x = y \\ \text{append}(y, \text{delete}(x, L)) & \text{sonst} \end{cases}$

ADT Liste (erweitert)

Signatur

- contains: Object $\times$ Liste $\rightarrow$ Boolean
- copy: Liste $\rightarrow$ Liste
- delete: Object $\times$ Liste $\rightarrow$ Liste
- deleteAll: Object $\times$ Liste $\rightarrow$ Liste
- prepend: Object $\times$ Liste $\rightarrow$ Liste
- addAll: Liste $\times$ Liste $\rightarrow$ Liste

Axiome

A6: $\text{deleteAll}(x, \text{create}) = \text{create}$

A7: $$\text{deleteAll}(x, \text{append}(y, L)) = \begin{cases} \text{deleteAll}(x, L) & \text{falls } x = y \\ \text{append}(y, \text{deleteAll}(x, L)) & \text{sonst} \end{cases}$$

ADT Liste (erweitert)

Signatur

- contains: Object $\times$ Liste $\rightarrow$ Boolean
- copy: Liste $\rightarrow$ Liste
- delete: Object $\times$ Liste $\rightarrow$ Liste
- deleteAll: Object $\times$ Liste $\rightarrow$ Liste
- prepend: Object $\times$ Liste $\rightarrow$ Liste
- addAll: Liste $\times$ Liste $\rightarrow$ Liste

Axiome

schönere Version von A6/A7:

$$\text{A6}^*\text{: contains}(x, \text{deleteAll}(x, L)) = \text{false}$$

ADT Liste (erweitert)

Signatur

- contains: Object $\times$ Liste $\rightarrow$ Boolean
- copy: Liste $\rightarrow$ Liste
- delete: Object $\times$ Liste $\rightarrow$ Liste
- deleteAll: Object $\times$ Liste $\rightarrow$ Liste
- prepend: Object $\times$ Liste $\rightarrow$ Liste
- addAll: Liste $\times$ Liste $\rightarrow$ Liste

Axiome

A8: $\text{prepend}(x, \text{create}) = \text{append}(x, \text{create})$

A9: $\text{prepend}(x, \text{append}(y, L)) = \text{append}(y, \text{prepend}(x, L))$

ADT Liste (erweitert)

Signatur

- contains: Object $\times$ Liste $\rightarrow$ Boolean
- copy: Liste $\rightarrow$ Liste
- delete: Object $\times$ Liste $\rightarrow$ Liste
- deleteAll: Object $\times$ Liste $\rightarrow$ Liste
- prepend: Object $\times$ Liste $\rightarrow$ Liste
- addAll: Liste $\times$ Liste $\rightarrow$ Liste

Axiome

$$\text{A9: } \text{addAll}(L, \text{create}) = L$$

$$\text{A10: } \text{addAll}(L, \text{append}(y, H)) = \text{addAll}(\text{append}(y, L), H)$$

Quilt

Quilt

Fragen

Fragen?
Evaluation