

## 10. Induktion in PVS (Forts.)

Die Form der Induktion, die sich (bei ADTs wie Listen, Bäumen usw.) an der induktiven Struktur des Datentyps orientiert, wird im allgemeinen als *strukturelle Induktion* bezeichnet.

Die allgemeinste Form der Induktion ist *fundierte Induktion* (engl. *well-founded induction*), die eine fundierte Relation als Grundlage erfordert.

Strukturelle Induktion kann als Spezialfall aus der fundierten Induktion über dem entsprechenden Datentyp (mit geeigneter fundierter Ordnungsrelation) abgeleitet werden.

# Fundierte Induktion in PVS

Zur Erinnerung: Definition eines Prädikats `well_founded?(<)` über Relationen  
(jetzt in Notation mit prädikativen Subtypen)

(Aus der `prelude`-Theorie `orders`)

```
< : VAR pred[[T,T]]
```

```
p : VAR pred[T]
```

```
x, y, z : VAR T
```

```
well_founded?(<): bool =
```

```
  (FORALL p: (EXISTS x: p(x))
```

```
    IMPLIES (EXISTS (y:(p)): (FORALL (x:(p)): (NOT x < y))))
```

## Fundierte Induktion in PVS (2)

```
% wf_induction defines induction for any type that  
% has a well-founded relation.
```

```
wf_induction [T: TYPE, <: (well_founded?[T])]: THEORY  
BEGIN
```

```
  wf_induction: LEMMA  
    (FORALL (p: pred[T]):  
      (FORALL (x: T):  
        (FORALL (y: T): y<x IMPLIES p(y)) IMPLIES p(x) )  
      IMPLIES  
        (FORALL (x:T): p(x)) )
```

```
END wf_induction
```

## Fundierte Induktion in PVS (3)

Induktion mit Maßfunktion in PVS, abgeleitet von fundierter Induktion

```
% measure_induction builds on well-founded induction.
% It allows induction over a type T for which a measure
% function m is defined.

measure_induction [T: TYPE, M: TYPE,
                  m: [T -> M],
                  <: (well_founded? [M]) ]: THEORY

BEGIN

measure_induction: LEMMA
  (FORALL (p: pred[T]):
    (FORALL (x: T):
      (FORALL (y: T): m(y) < m(x) IMPLIES p(y)) IMPLIES p(x) )
    IMPLIES
      (FORALL (x: T): p(x)))

END measure_induction
```

# Induktive Definitionen

Ein Prädikat kann induktiv definiert werden und damit die Basis für einen Induktionsbeweis abgeben.

PVS unterstützt induktive Definition durch die Generierung der entsprechenden Induktionsformeln.

Beispiel: gerade Zahlen

```
even(n:nat) INDUCTIVE bool =  
  n=0 OR (n>1 AND even(n-2))
```

Für induktive Definitionen gelten Einschränkungen bzgl. Terminierung wie für rekursive Funktionen.

## Induktive Definitionen (2)

Die Definition von `even` kann aufgefasst werden als die Zusammenfassung von 2 'Regeln':

1. `even(0)`

2. `even(n) => even(n+2)`

und der "Abschlußeigenschaft": `even(x)` gilt genau dann, wenn es aufgrund der Regeln (1) und (2) der Fall ist – die Menge  $\{n : nat \mid even(n)\}$  ist die *kleinste* Menge, die unter den Regeln abgeschlossen ist.

Diese Struktur wird ausgenutzt, um induktiv eine Eigenschaft für alle Elemente der Menge, die durch das (induktiv definierte) Prädikat charakterisiert wird, zu beweisen.

## Induktive Definitionen (3)

Zugehöriges Induktionsaxiom in PVS:

```
even_weak_induction: AXIOM
  (FORALL (P: [nat->bool]):
    (FORALL (n:nat): n=0 OR (n>1 AND P(n-2)) IMPLIES P(n))
    IMPLIES (FORALL (n:nat): even(n) IMPLIES P(n)))
```

In einer zweiten Form des Induktionsaxioms steht die Eigenschaft  $\text{even}(n)$  in der Voraussetzung des Induktionsschritts zur Verfügung:

```
even_induction: AXIOM
  (FORALL (P: [nat->bool]):
    (FORALL (n:nat): n=0 OR (n>1 AND even(n-2) AND P(n-2)) IMPLIES P(n))
    IMPLIES (FORALL (n:nat): even(n) IMPLIES P(n)))
```

