Informelle Spezifikation

- “Entscheide, ob eine gegeben Sequenz von Elementen ein Palindrom ist.”
- “Berechne die Primfaktorzerlegung einer ganzen Zahl $n \in \mathbb{N}$”
- “Generiere in aufsteigender Reihenfolge die ersten n ($n \in \mathbb{N}$) Elemente der Sequenz aller natürlichen Zahlen, die durch keine Primzahlen außer 2, 3 und 5 teilbar sind.” (Hammings Problem)

Von der informellen zur formalen Spezifikation

- Präzisieren:

  Palindrom *das*; Wortfolge od. Satz, die vorwärts wie rückwärts gelesen [den gleichen] Sinn ergeben, z.B. “Otto”, “Relief-pfeiler”. (Duden)

- Formalisieren und Reduktion: Sei `rev` eine Funktion, die Listen umdreht, dann ist

```
palindrom w <==> rev w = w
```

Spezifikation von `rev`

- Axiomatisch:

  Erster Versuch:

  ```
  rev (rev x) = x
  x ~= [] ==> rev x ~= x
  ```

  Zweiter Versuch:

  ```
  i < length w ==> nth i w = nth (length w- i- 1) (rev w)
  ```

- Operational:

  ```
  rev []     = []
  rev (x#xs) = (rev xs) @ [x]
  ```

Spezifikation: Was ist zu beachten?

- Notation (oben Isabelle)
- Semantik (oben Isabelle/HOL)
- Basisdatentypen (z.B. Listen)
- Programmierstil — funktional vs. imperativ
- Spezifikationsstil — algebraisch (Isabelle) vs. modellorientiert (Z)

Kurzwiederholung Logik I

- Syntaktische *Ableitung* $\vdash \phi$ durch Ableitungsregeln.
- Semantische *Folgerung* $\models \phi$: ϕ gilt in allen *Modellen*.
- Konsistenz: $\vdash \phi \Rightarrow \models \phi$
- Vollständigkeit: $\models \phi \Rightarrow \vdash \phi$

- Gödel's Unvollständigkeitssätze:

  1. Logik $\mathcal{L}$ konsistent und beinhaltet Peano-Arithmetik (nat. Zahlen plus Induktionsschema) dann ist $\mathcal{L}$ unvollständig.

  2. Logik $\mathcal{L}$ konsistent und beinhaltet Peano-Arithmetik dann gibt es keinen Satz ϕ, so daß $\vdash \phi$ gdw $\mathcal{L}$ konsistent.

  $\Rightarrow$ Nichttriviale Spezifikationssprachen sind unvollständig

- Bekannte Logiken: Gleichungslogik, Horn-Logik, Aussagenlogik und Prädikatenlogik erster Stufe (konsistent, vollständig, langweilig)
  Prädikatenlogik höherer Stufe (intuitionistisch, klassisch)

Kurzwiederholung Algebra

- Eine *Signatur* $\Sigma = (S, \Omega)$ besteht aus *Sorten* S und *Operationen* $\omega : s_1 \times \ldots \times s_n \to s$.

- *Terme* $t \in T_\Sigma(X)$ sind entweder Variablen $x \in X$, oder eine Operation $\omega : s_1 \times \ldots \times s_n \to s$, angewandt auf n Terme $t_i \in T_\Sigma(X)$

- Eine Spezifikation $SP = (\Sigma, R)$ besteht aus einer Signatur und Regeln R (Gleichungen, Ableitungsregeln, ...)

- Beispiel: Monoide
$$\begin{aligned} M &= (\Sigma_M, R_M) \\ \Sigma_M &= (\{s\}, \{0 :\to s,\ \cdot : s \times s \to s\} \\ R_M &= \{0 \cdot x = x, x \cdot 0 = x, (x \cdot y) \cdot z = x \cdot (y \cdot z)\} \end{aligned}$$

Beispiel: Palindrome in Isabelle

```
Palindrome = List +
consts  palindrome :: 'a list => bool
defs    palindrome_def  "palindrome xs == (rev xs) = xs"
end

List = ...
consts  rev          :: 'a list => 'a list
primrec "rev([]) = []"
        "rev(x#xs) = rev(xs) @ [x]"
```

Beispiel: Palindrome in Isabelle

```
> use_thy "Palindrome";
Reading "Palindrome.thy"
val it = () : unit;
> goalw thy [palindrome_def] "palindrome [1,2,1,1]";
Level 0
palindrome [1, 2, 1, 1]
 1. rev [1, 2, 1, 1] = [1, 2, 1, 1]
val it = [] : thm list
>  by (Asm_simp_tac 1);
Level 1
palindrome [1, 2, 1, 1]
 1. False
val it = () : unit
>
```

Beispiel: Palindrome in Isabelle

```
> goalw thy [palindrome_def] "palindrome [2,1,2,1,2]";
Level 0
palindrome [2, 1, 2, 1, 2]
 1. rev [2, 1, 2, 1, 2] = [2, 1, 2, 1, 2]
val it = [] : thm list
> by (Asm_simp_tac 1);
Level 1
palindrome [2, 1, 2, 1, 2]
No subgoals!
val it = () : unit
>
```

Literature zu Isabelle: Tobias Nipkow: Isabelle — The Tutorial.

Beispiel: Palindrome in Z

$$
\begin{array}{|l}
\hline
[X] \\
\mathit{palindrome} : \mathbb{P}\,\mathit{seq}\,X \\
\mathit{rev} : \mathit{seq}\,X \to \mathit{seq}\,X \\
\hline
\mathit{palindrome} = \{\, w \in \mathbb{P}\,\mathit{seq}\,X \mid \mathit{rev}(w) = w \,\} \\
\forall\, s : \mathit{seq}\,X \bullet \mathit{rev}\; s = (\lambda\, i : 1 \ldots \#s \bullet (\#s + 1) - i) \fatsemi s \\
\hline
\end{array}
$$