Abschließendes Beispiel

- Axiome: Verhalten von `lookup`:

```
axioms
 new_key db = (db', k) ==> lookup db' k = None
 lookup init k = Nothing
 lookup (upd (db k d)) k = Just d
 k ~= l ==> lookup (upd (db k d) l)= lookup db l
 lookup (rem db k) k = Nothing
 k ~= l ==> lookup (rem db k) l= lookup db l
```


Abschließendes Beispiel

- Verhalten von `rem` und `update` (benötigt?)::

```
rem init = init
rem (upd db k d) k = rem db k
k ~= l ==>  rem (upd db k d) l = upd (rem db l) k d
upd (upd db k d) k d'= upd db k d'
k ~= l ==> upd (upd db k d) l d= upd (upd db l d) k d
```

- `new_key`: *Seiteneffekt* bei Generierung eines neuen Schlüssels
- Modellierung des Zustands *explizit* — kann umständlich werden
- Deshalb: *imperative* Modellierung $\rightsquigarrow$ Z (nächste Woche)

Zusammenfassung

- *Algebraische* Spezifikationen:
  - ○ *Signatur* deklariert *Typen* und *Operationen*
  - ○ *Axiome* spezifieren *Verhalten*
- *Signatur-* und *Theoriemorphismen*
  - ○ *Implementation* als *Theoriemorphismus* $\sigma : Spec \to Imp$
- Der *Entwicklungszyklus*:
  - ○ Anforderungsspezifikation (axiomatisch)
  - ○ Entwurfsspezifikation (konsistent)
  - ○ Ausführbare Spezifikation (Prototyp)
- Nächste Woche: *modellbasierte* Spezifikation *imperativer* Programme

Vorlesung vom 16.01.06: Z für Anfänger

Wo sind wir?

- Teil I: Grundlagen formaler Logik
- Teil II: Modellierung
- Teil III: Spezifikationsformalismen
  - ○ Axiomatische Spezifikation
  - ○ *Spezifikation mit Z: Grundlagen & Einführung*
  - ○ Z: Datentypen & Schemakalkül
  - ○ Verfeinerung, Datenverfeinerung und Verhaltensgleichheit

Was ist Z?

- Spezifikationssprache, basierend auf *getypter Mengenlehre*
  - ○ *Alles* ist eine *Menge* (Mengen sind Typen)
  - ○ Viel syntaktische Konvention
- Spezifikation von *imperativen* Programmen
  - ○ *Zustand* und *Zustandsänderung* integraler Bestandteil
- *Entwickelt* Ende 80er, Oxford UCL und IBM UK

Mengenlehre

- Begründet *1874–1884* durch Georg *Cantor* (*Naive Mengenlehre*)
  - ○ Leider inkonsistent (Burali-Forte Paradox, Russelsches Paradox)
- *Axiomatisierungen* durch *Zermelo* (inkonsistent), später *Fränkel* (ZF), *von Neumann, Gödel, Bernays*
- *Axiome*: *Extensionalität*, *Separation*, *Paarbildung*, *Vereinigung*, *Potenzmenge*, *Wohlfundiertheit*, *Ersetzung*, *Leere Menge*, *Unendlichkeit*, *Auswahl*
  - ○ *Auswahlaxiom* unabhängig vom Rest
- *Getypte* Mengenlehre: HOL
  - ○ *Mengen* sind *Prädikate* sind *Funktionen nach* `bool`

Meine erste Z-Spezifikation

Eine *axiomatische Definition*:

$$iroot : \mathbb{N} \to \mathbb{N}$$

$$\forall\, a : \mathbb{N} \bullet iroot(a) * iroot(a) \leq a < (iroot(a) + 1) * (iroot(a) + 1)$$

- *Boxen* kennzeichnen *Spezifikationen*
- *Deklaration*, dann *Spezifikation* (*Paragraphen*)

Elemente von Z

- Sprache *im kleinen:*
  - Deklarationen
  - Ausdrücke
  - Prädikate
- Basiselemente: das *mathematical toolkit*
- Sprache *im großen:*
  - Schema
  - Schemakalkül

Der Texteditor

- *Beispiel*: Editor zum Bearbeiten von *Texten*
- *Typdeklaration* und *Typsynomym*:

$$[CHAR]$$
$$TEXT == seq\ CHAR$$

  - seq: Listen in Z
  - Beliebige Abkürzungen mit $==$
- *Maximale* Anzahl Zeichen im Editor:

$maxsize : \mathbb{N}$

$maxsize \leq 65535$

Zustandsschemata

- *Repräsentation* von *Zuständen* in Z: *Schema*
- Editor hat internen *Zustand* (edierter Text):

Editor

$left, right : TEXT$

$\#(left \frown right) \leq maxsize$

- Schema haben *Namen*
- Schema sind *geschlossen*
- *Prädikate* sind *Invarianten*

Anfangszustand

- Spezieller *Name* $Init$

Init

$Editor$

$left = right = \langle\rangle$

- *Benutzt* Schema $Editor$
  - Schemas als *Makros*

Zustandsübergang: Zeichen einfügen

- Axiomatische Definition: druckende Zeichen

$printing : \mathbb{P}\ CHAR$

- *Zustandsübergang* des Editors: Zeichen einfügen

Insert

$\Delta Editor$
$ch? : CHAR$

$ch? \in printing$
$left' = left \frown \langle ch? \rangle$
$right' = right$

  - Δ deklariert *Zustandsübergang*
  - Vorher: $left$, nachher: $left'$
  - $ch?$ Eingabevariable

Cursorbewegung

$right\_arrow : CHAR$

$right\_arrow \notin printing$

- $right\_arrow$ nicht einfügen

Forward

$\Delta Editor$
$ch? : CHAR$

$ch? = right\_arrow$
$left' = left \frown \langle head(right) \rangle$
$right' = tail(right)$
$right \neq \langle\rangle$

- *Undefiniert* wenn $right = \langle\rangle$
- *Implizite* Vorbedingung *explizit* machen
- Operation *partiell*

Cursorbewegung an Dateiende

Cursor an *Dateiende*:

EOF

$Editor$

$right = \langle\rangle$

Eingabe ist Pfeil nach rechts:

RightArrow

$ch? : CHAR$

$ch? = right\_arrow$

- Alles *zusammen*:

$$T\_Forward \mathrel{\widehat{=}} Forward \lor (EOF \land RightArrow \land \Xi Editor)$$

  - ΞS: Zustand *unverändert*

$$\Xi Editor == \Delta Editor \land left' = left \land right' = right$$

Mengen und Typen

- *Mengen*: Aufzählungen $\{red, green, blue\}$ oder $l \mathrel{..} k$; vordefiniert: $\mathbb{N}, \mathbb{N}_1, \mathbb{Z}, \mathbb{R}$
- Mengen sind getypt: $\{1, 2, red\}$ *Typfehler!*
- Definition neuer Typen:
  - Freie Typen:

$$COLOUR ::= red \mid green \mid blue$$
$$NAT ::= null \mid cons\langle\langle NAT \rangle\rangle$$

  - Deklarationen: $[NAME]$
- Konstruktionen auf Mengen: $\cup, \cap, \setminus, \mathbb{P}, \mathbb{F}$

Strukturierte Typen

- *Kartesisches Produkt*:

$$DATE == DAY \times MONTH \times YEAR \qquad heute = (16, 1, 2006)$$

- Binäre *Relationen*: $A \leftrightarrow B == \mathbb{P}(A \times B)$
  mit $\mathsf{dom}(A \leftrightarrow B) = \{a : A \mid \exists\, b \in B \bullet (a \mapsto b) \in A \leftrightarrow B)\}$
  $\mathsf{ran}(A \leftrightarrow B) = \{b : B \mid \exists\, a \in A \bullet (a \mapsto b) \in A \leftrightarrow B)\}$
- Partielle und totale *Funktionen*:

$$A \rightarrowtail\!\!\!\!\!\!+ B == \{f : A \leftrightarrow B \mid \forall\, a : A, b_1, b_2 : B \bullet (a \mapsto b_1) \in f \wedge (a \mapsto b_2) \in f \Rightarrow b_1 = b_2\}$$
$$A \rightarrow B == \{f : A \rightarrowtail\!\!\!\!\!\!+ B \mid \mathsf{dom}\, f = A\}$$

Ausdrücke

- Ausdrücke:
  - Mengenlehre ($\cup$, $\cap$, Komprehension . . .)
  - Arithmetische und logische Operatoren
  - *Kein* vordefinierter Typ Bool
- Sequenzen:

$$\mathsf{seq}\, X == \{f : \mathbb{N} \twoheadrightarrow X \mid \mathsf{dom}\, f = 1 \,..\, \#f\}$$

  - Vordefinierte Operatoren: *head*, *tail*, *front*, *last*, $\frown$, *in*

Der Geburtstagskalender

- Ziel: *Kalender*, der an *Geburtstage* erinnert

$$[NAME, DATE]$$

- *Datenstrukturen*

BirthdayBook
$known : \mathbb{P}\, NAME$
$birthday : NAME \rightarrowtail\!\!\!\!\!\!+ DATE$

$known = \mathsf{dom}\, birthday$

- *Operationen*: hinzufügen, finden, erinnern

Geburtstag hinzufügen

AddBirthday
$\Delta BirthdayBook$
$name? : NAME$
$date? : DATE$

$name? \notin known$
$birthday' = birthday \cup \{name? \mapsto date?\}$

Geburtstag finden

FindBirtday
$\Xi BirthdayBook$
$name? : NAME$
$date! : DATE$

$name? \in known$
$date! = birthday(name?)$

An Geburtstage erinnern

Remind
$\Xi BirthdayBook$
$today? : DATE$
$cards! : \mathbb{P}\, NAME$

$cards! = \{n \in known \mid birthday(n) = today\}$

Zusammenfassung

- Z: formale *Spezifikationssprache*
- Basiert auf *getypter Mengenlehre*
- Typen = Mengen = Prädikate
- Elemente von Z:
  - Deklaratione
  - Ausdrücke
  - Prädikate
- Vordefinierte Deklarationen, Operatoren im *mathematical toolkit*
- *Schema* beschreiben *Zustand* und *Zustandsübergänge*
  - Rechnen mit Schemata: Schemakalkül $\leadsto$ nächste Woche