Formale Modellierung
Vorlesung 11 vom 17.06.13: Die Z-Notation

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2013

Fahrplan

- Teil I: Formale Logik
- Teil II: Spezifikation und Verifikation
  - Modellierung von Programmen
  - Die Z-Notation
  - Formale Modellierung mit der UML und OCL
- Teil III: Schluß

Das Tagesmenü

- Die Z-Notation
  - Grundlagen
  - Der Schemakalkül
  - Die Bücherei
- Das Beispiel
  - Der sichere autonome Roboter

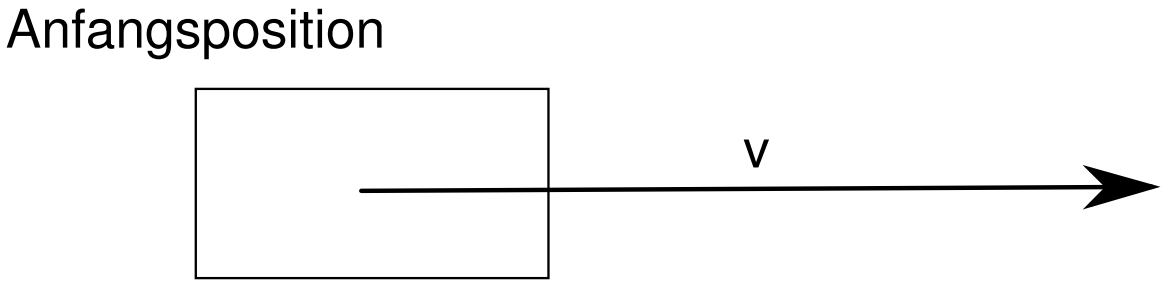
Was ist Z?

- 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`

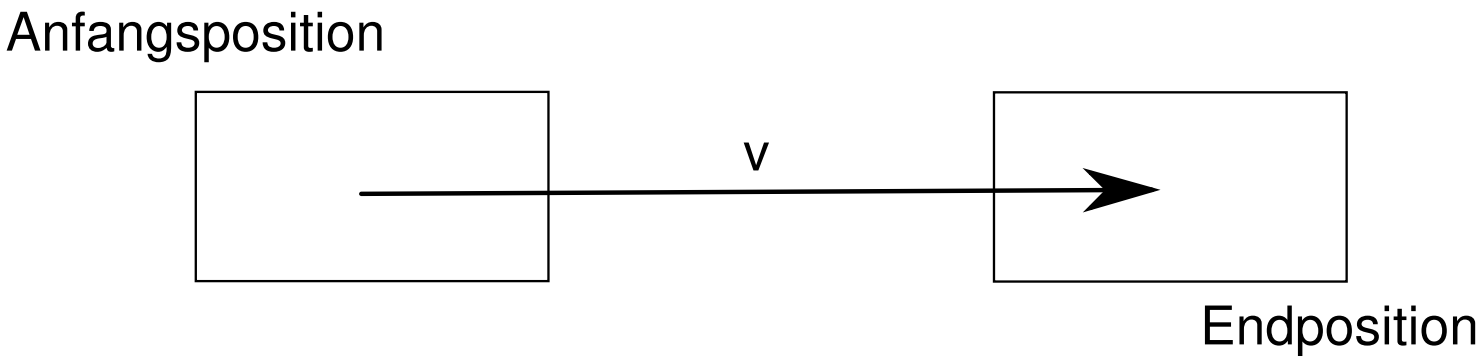
Beispiel: Fahrzeug in der Ebene



- Bremszeit und Bremsstrecke:

$$v(t) = v_0 - a_{brk}\,t \qquad s(t) = v_0\,t - \frac{a_{brk}}{2}\,t^2 \qquad T = \frac{v_0}{a_{brk}} \qquad S = \frac{v_0^2}{2a_{brk}}$$

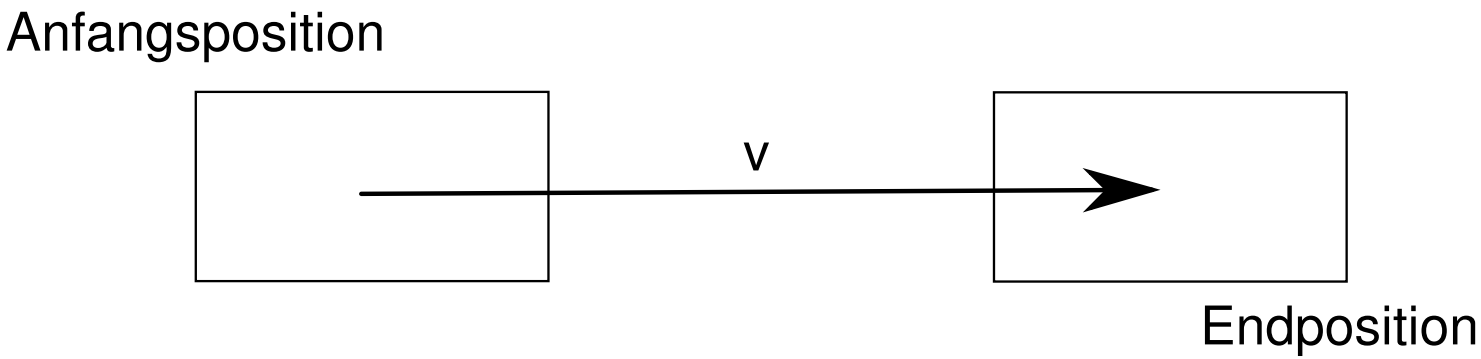
Beispiel: Fahrzeug in der Ebene



- Bremszeit und Bremsstrecke:

$$v(t) = v_0 - a_{brk}\,t \qquad s(t) = v_0\,t - \frac{a_{brk}}{2}\,t^2 \qquad T = \frac{v_0}{a_{brk}} \qquad S = \frac{v_0^2}{2a_{brk}}$$

Beispiel: Fahrzeug in der Ebene



- Bremszeit und Bremsstrecke:

$$v(t) = v_0 - a_{brk}\,t \qquad s(t) = v_0\,t - \frac{a_{brk}}{2}\,t^2 \qquad T = \frac{v_0}{a_{brk}} \qquad S = \frac{v_0^2}{2a_{brk}}$$

- Modellierung in Z: Berechnung des Bremsweges

$$brk : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$$

$$\forall\, v, a : \mathbb{N} \bullet brk\,(v, a) = (v * v)\ \mathrm{div}\ (2 * a)$$

Modellierung: Punkte und Polygone

Schematyp für Punkte (Vektoren):

$$
\begin{array}{|l}
\textit{VEC} \\
\hline
x : \mathbb{Z} \\
y : \mathbb{Z} \\
\hline
\end{array}
$$

Typ für Polygone und Strecken:

$$\textit{POLY} == \{s : \operatorname{seq} \textit{VEC} \mid \#s > 3 \land \textit{head}\, s = \textit{last}\, s\}$$
$$\textit{SEG} == \textit{VEC} \times \textit{VEC}$$

Addition und Skalarmultiplikation von Vektoren

$$
\begin{array}{|l}
\textit{add} : \textit{VEC} \times \textit{VEC} \to \textit{VEC} \\
\textit{smult} : R \times \textit{VEC} \to \textit{VEC} \\
\hline
\forall\, p, q : \textit{VEC} \bullet \textit{add}\,(p, q) = (p.x + q.x, p.y + q.y) \\
\forall\, n : R;\ p : \textit{VEC} \bullet \textit{smult}\,(n, p) = (n * p.x, n * p.y)
\end{array}
$$

Mehr zu Punkten und Polygonen

Durch eine Strecke definierte linke Halbebene

$$left : SEG \to \mathbb{P}\, VEC$$

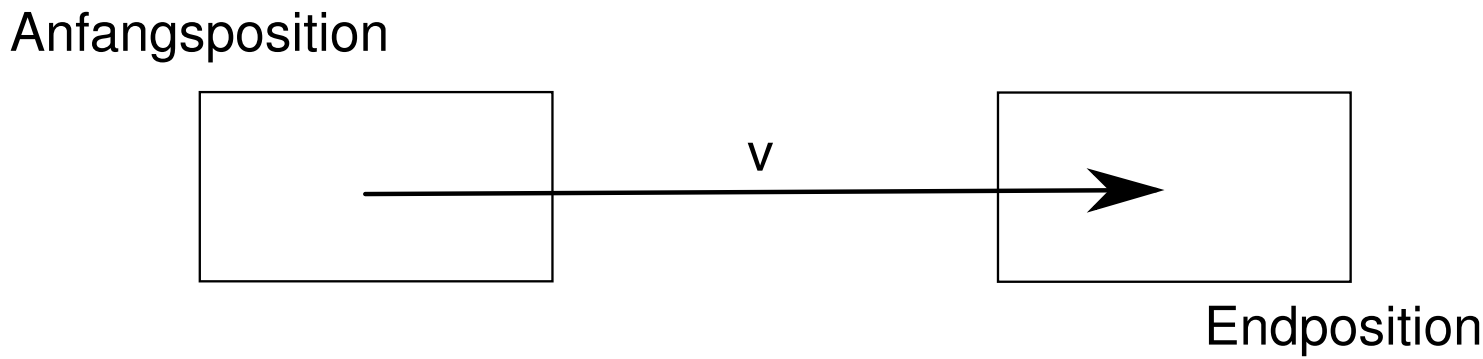
$$\forall\, a, b : VEC \bullet left\,(a, b) = \{p : VEC \mid (b.y - a.y) * (p.x - b.x) - (p.y - b.y) * (b.x - a.x) < 0\}$$

Fläche eines Polygons:

$$sides : POLY \to \mathbb{P}\, SEG$$
$$area : POLY \to \mathbb{P}\, VEC$$

$$\forall\, p : POLY \bullet sides\, p = \{s : SEG \mid \langle s.1, s.2 \rangle \text{ in } p\}$$
$$\forall\, p : POLY \bullet area\, p = \bigcap \{s : SEG \mid s \in sides\, p \bullet left\, s\}$$

Bewegung von Polygonen



Bewegung eines Polygons um einen Vektor

$$move : POLY \times VEC \to POLY$$

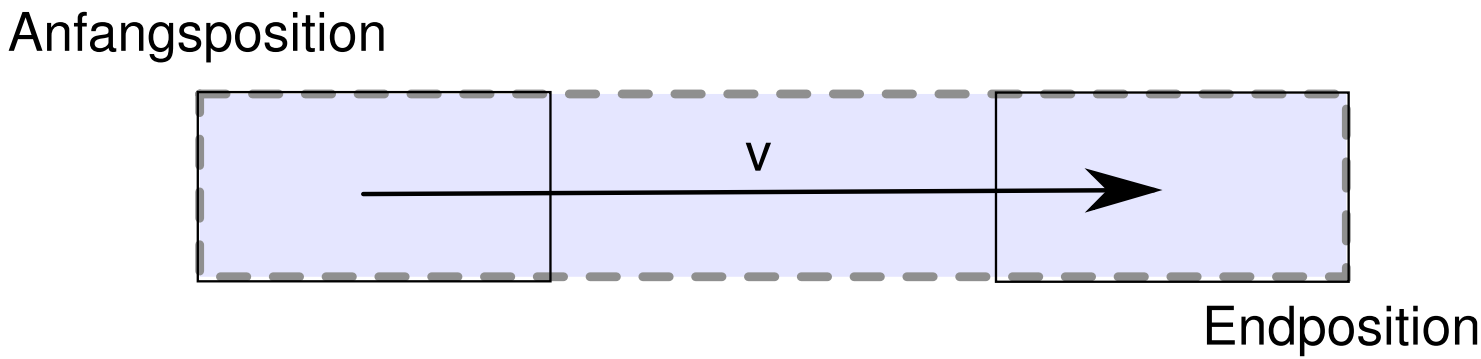
$$\forall\, p : POLY;\ v : VEC \bullet move\,(p, v) = (\lambda\, x : VEC \bullet add\,(x, v)) \circ p$$

Durch eine Bewegung überstrichene Fläche

$$cov : POLY \times VEC \to \mathbb{P}\, VEC$$

$$\forall\, p : POLY;\ v : VEC \bullet$$
$$cov\,(p, v) = \bigcup \{\tau : R \mid 0 \leq \tau \leq 1 \bullet area\,(move\,(p, \tau * v))\}$$

Bewegung von Polygonen



Bewegung eines Polygons um einen Vektor

$$move : POLY \times VEC \rightarrow POLY$$

$$\forall\, p : POLY;\ v : VEC \bullet move\,(p, v) = (\lambda\, x : VEC \bullet add\,(x, v)) \circ p$$

Durch eine Bewegung überstrichene Fläche

$$cov : POLY \times VEC \rightarrow \mathbb{P}\, VEC$$

$$\forall\, p : POLY;\ v : VEC \bullet$$
$$cov\,(p, v) = \bigcup \{\tau : R \mid 0 \leq \tau \leq 1 \bullet area\,(move\,(p, \tau * v))\}$$

Der Autonome Roboter

RobotParam

$cont : POLY$

$a_{brk} : \mathbb{Z}$

$T : \mathbb{Z}$

World

RobotParam

$obs : \mathbb{P}\, VEC$

Robot

RobotParam

$vel : \mathbb{Z}$

$\omega : \mathbb{Z}$

$v : VEC$

$o : VEC$

$a? : \mathbb{Z}$

$\delta\omega? : \mathbb{Z}$

$c : POLY$

$c = move\,(cont, o)$

$v = cart\,(vel, \omega)$

Der Autonome Roboter in Bewegung

RobotMoves

$\Delta Robot$

$\Xi World$

$vel' = vel + a? * T$

$\omega' = \omega + \delta\omega? * T$

$o' = add\,(o, v')$

- Wann werden Änderungen effektiv?
- Wann ist der Roboter sicher?

Der Autonome Roboter Bremst

- Verhalten beim Bremsen:

RobotBrakes

$\Delta Robot$

$\Xi World$

$vel' = vel - a_{brk} * T$

$\omega' = \omega$

$o' = add\,(o, v')$

- Invariante: Bremsen muß immer sicher sein.

Der Sichere Roboter: Anforderung

- Jetzt und in unmittelbarer Zukunft sicher

RobotSafeMove

RobotMoves

$$cov\,(c, v') \cap obs = \emptyset$$

- Sicheres Bremsen: Bremsweg frei

RobotSafeBrake

RobotBrakes

$$cov\,(c, brk\,(v, \omega, a_{brk})) \cap obs = \emptyset$$

- Sicher: Roboter kann sicher fahren oder bremsen

$$RobotSafe == RobotSafeMove \lor RobotSafeBrake$$

Der Sichere Roboter: Implementation

Sicheres Fahren: Weg ist frei, Bremsweg ist frei

RobotMovesSafely

$$\Delta Robot$$
$$\Xi World$$

$$(cov\,(c, v') \cup cov\,(move\,(c, v'), brk\,(v', \omega', a_{brk}))) \cap obs = \emptyset$$
$$vel' = vel + a? * T$$
$$\omega' = \omega + \delta\omega? * T$$
$$o' = add\,(o, v')$$

Implementation des sicheren Roboterverhaltens:

$$RobotDrivesSafely = RobotMovesSafely \lor RobotBrakes$$

Zu zeigen:

$$RobotDrivesSafely \Rightarrow RobotSafe$$
$$RobotDrivesSafely \Rightarrow RobotSafe'$$

Zusammenfassung

- Z basiert auf getypter Mengenlehre
- Elemente der Sprache:
  - Axiomatische Definitionen
  - Schema, Schemakalkül
  - Mathematical Toolkit (Standardbücherei)
- Modellbasierte und Zustandsbasierte Spezifikationen