Programmierung und Modellierung

Fixpunktsatz und Semantik rekursiver Funktionen

Martin Wirsing

in Zusammenarbeit mit

Moritz Hammer

Inhalt

12.1 Fixpunkte

- **Definition**

  Sei M eine Menge mit einer (partiellen) Ordnung $\blacktriangleleft \subseteq M \times M$,

  $f: M \to M$ eine Funktion.

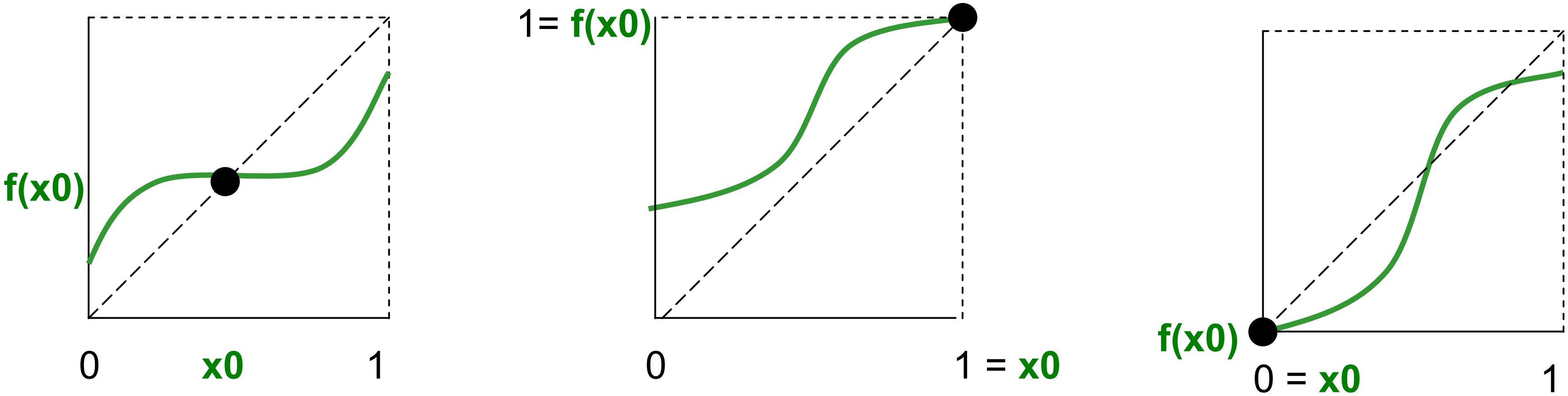
  - Ein Wert $x_0 \in M$ heißt **Fixpunkt** von f, wenn gilt

    $x_0 = f(x_0)$.

  - Wir bezeichnen mit **fix f** den **kleinsten Fixpunkt** von f (sofern dieser existiert), d.h.
    - (fix f) $\blacktriangleleft$ y für jeden (anderen) Fixpunkt y von f und
    - (fix f) ist eindeutig bestimmt.

Fixpunkte: Beispiele

- **Fixpunkt einer stetigen, monoton wachsenden Funktion über dem Intervall [0,1]**



- Jede stetige, monoton wachsende Funktion f: [0,1] → [0,1] besitzt einen kleinsten Fixpunkt

Fixpunkte: Beispiele

- **Fixpunkt einer BNF-Grammatik**

Beispiel

- `sq ::= “0” | “1” sq`

kann als Gleichung zwischen Mengen aufgefasst werden:

- $X = \{0\} \cup \{\text{“1x”} \mid x \in X\}$,

d.h. wir haben eine Gleichung der Form

- $X = E(X)$

wobei X eine Variable ist, die eine Menge bezeichnet und

E(X) ein Ausdruck, der X enthält.

- Lösung

$M_0 = \{ 1\ldots10 \mid n >= 0\} = \{ 1^n0 \mid n >= 0\}$

M_0 ist kleinster Fixpunkt bzgl. der Teilmengenordnung $\subseteq$ zwischen Mengen endlicher Wörter über dem Alphabet {0,1}.

Fixpunkte: Beispiele

- **Fixpunkte bei rekursiven Funktionen**
  - `fun fakt = fn x => if x=0 then 1 else x*fakt(x-1)`

    hat die Gestalt (mit `f = fakt`)

    ```
    f = F(f)
    ```

    wobei

    ```
    F(f) = fn x => if x=0 then 1 else x*f(x-1)
    ```

    bzw..

    ```
    fun F f = fn x => if x=0 then 1 else x*f(x-1)
    ```

    ist eine Funktion höherer Ordnung des Typs (ein **Funktional**)

    ```
    (int -> int) -> int -> int
    ```
  - Die Fakultätsfunktion `x!` ist (kleinster) Fixpunkt der Gleichung `f = F(f)`
  - Allgemein läßt sich die Funktion fix zur Berechnung des kleinsten Fixpunkts eines Funktionals F in SML definieren als:

    `fun fix F x = F (fix F) x` vom Typ

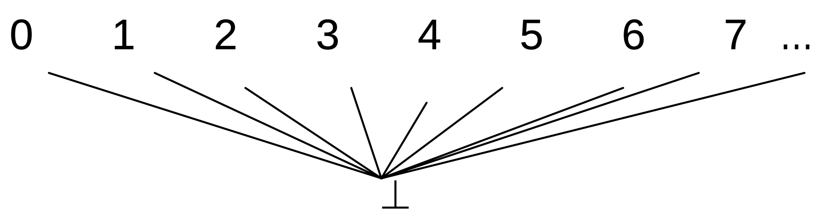
    ```
    fix : (('a -> 'b) -> 'a -> 'b) -> 'a -> 'b
    ```

Partielle Ordnung

- Eine binäre Relation ◀ auf einer Menge M heißt **partielle Ordnung**, wenn für alle x, y, z ∈ M Folgendes gilt:
  - x ◀ x (Reflexivität)
  - x ◀ y, y ◀ x => x = y (Antisymmetrie)
  - x ◀ y, y ◀ z => x ◀ z (Transitivität)
- x ∈ M ist **kleinstes Element**, wenn x ◀ y für alle y ∈ M.

Partielle Ordnung: Beispiele

- **Natürliche Ordnung**: M = N, x ◂ y => $x \leq y$; kleinstes Element: 0
- **Diskrete Ordnung:**
  M beliebig, x ◂ y => x = y; für |M| >1 gibt es kein kleinstes Element
- **Teilmengenordnung:** M = $\wp(X)$, x ◂ y => $x \subseteq y$; kleinstes Element: $\emptyset$
- **Flache Ordnung** :
  $M_\perp = M \cup \{\perp\}$, x ◂ y => $x = \perp$ oder x = y; kleinstes Element: $\perp$
  Beispiel: $N_\perp$



- Es ist üblich, die partielle Ordnung durch die **Trägermenge** zu bezeichnen und immer das Symbol ◂ für die Relation zu verwenden.

Ketten und Suprema

- Sei $U \subseteq X$. Ein Element $x \in X$ heißt **kleinste obere Schranke** oder **Supremum** von U (geschrieben **sup U**), wenn gilt
  - $u \blacktriangleleft x$ für alle $u \in U$,
  - Falls $u \blacktriangleleft y$ für alle $u \in U$, so gilt $x \blacktriangleleft y$
- Eigenschaften
  - Sind x, y beides kleinste Elemente, so folgt x = y. Man bezeichnet das kleinste Element, wenn es existiert, mit dem Symbol $\perp$.
  - Sind x, y beides kleinste obere Schranken von $U \subseteq X$, so folgt x = y.

Vollständigkeit

Sei X eine partielle Ordnung.

- Eine Folge $(x_i)_{i \in N}$ von Elementen $x_i \in X$ heißt **ω-Kette**, kurz **Kette**, wenn $x_i \blacktriangleleft x_{i+1}$ für alle i.
- Eine partielle Ordnung X heißt **vollständig** (bzgl. ω-Ketten), wenn für jede ω-Kette (x_i) das Supremum von $\{x_i \mid i \in N\}$ existiert. Man schreibt sup (x_i) für dieses Supremum.
- Eine vollständige partielle Ordnung bezeichnet man auf Englisch als **ω-complete partial order**, kurz **ω–cpo** oder noch kürzer **cpo**.
- **Bemerkung**:

  Man kann den Begriff der Vollständigkeit auf **gerichtete Mengen** verallgemeinern. Wir brauchen das hier nicht.

Beispiele

- Alle auf Folie 9 gegebenen Beispiele ausser N sind cpos; insbesondere sind
  - die Potenzmenge ($\wp$(X), $\subseteq$) mit Teilmengenordnung und
  - die flache Ordnung $X_\perp$

  cpos mit kleinstem Element.
- Seien X, Y cpos mit kleinstem Element $\perp$; Z ein cpo. Die folgenden Konstruktionen liefern cpos mit kleinstem Element:
  - $[Z \to Y] = \{f : Z \to Y \mid f \text{ stetig}\}$ mit
    - f $\blacktriangleleft$ g, wenn f(x) $\blacktriangleleft$ g(x) für alle x $\in$ X
    - Kleinstes Element $\Omega(x) = \perp$
  - X ^{x} Y mit
    - (x, y) $\blacktriangleleft$ (x0, y0), wenn x $\blacktriangleleft$ x0 und y $\blacktriangleleft$ y0;
    - kleinstes Element $(\perp_X, \perp_Y)$
  - X* mit
    - [x1, ... , xm] $\blacktriangleleft$ [y1, ... , yn], wenn m = n und xi $\blacktriangleleft$ yi für i = 1, ... , m.

Stetige Funktionen

Seien X, Y partielle Ordnungen. Alle betrachteten Funktionen seien total.

- $f : X \to Y$ heißt **monoton**, wenn für alle $x,y \in X$ gilt:

  $$x \blacktriangleleft y \text{ impliziert } f(x) \blacktriangleleft f(y).$$

- Eine monotone Funktion $f : X \to Y$ heißt **stetig** (bzgl. ω-Ketten), wenn für jede Kette $(x_i)_{i \in N}$ gilt:

  $$f(\ \sup\ (x_i)_{i \in N}\ ) = \sup\ (f(x_i))_{i \in N}\ )$$

- Man beachte, dass die Folge $(f(x_i))_{i \in N}$ wegen der Monotonie von f eine Kette bildet.

Stetige Funktionen

- Beispiele
  - Die Projektionen $p_X : X \times Y \to X$, $p_Y : X \times Y \to Y$ sind stetig.
  - Sind $f : Z \to X$ und $g : Z \to Y$ stetig, so auch $h(z) = (f(z), g(z))$.
  - Die Applikation . (.) : $[X \to Y] * X \to Y$ ist stetig.
  - Ist $f : X * Y \to Z$ stetig, so ist die “gecurryte” Funktion $h : X \to Y \to Z$ definiert durch `h(x) = fn y => f(x, y)` wohldefiniert (liefert also stetige Funktionen) und außerdem selbst stetig.

Fixpunktsätze

- **Fixpunktsatz von Knaster-Tarski***

  Sei X ein ω-cpo mit kleinstem Element.

  Dann besitzt jede **monotone** Funktion $f : X \to X$ einen kleinsten Fixpunkt.

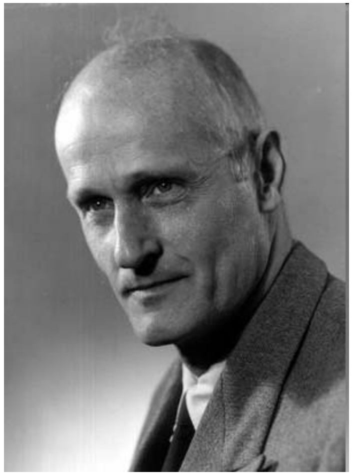
Bronisław Knaster
1893–1990
Poln. Mathematiker
Topologie, Mengenlehre

Alfred Tarski
1901–1983
Pole, Prof. Berkeley
Bedeutender Logiker

*Alfred Tarski: A lattice-theoretical fixpoint theorem and its applications". *Pacific J. of Math.***5:2**: 1955, 285–309.

*B. Knaster: Un théorème sur les fonctions d'ensembles. *Ann. Soc. Polon. Math.* **6, 1928,** 133–134

Fixpunktsätze



Stephen Kleene
1909–1994
Amerik. Logiker

- **Fixpunktsatz von Kleene**:** Sei X ein ω-cpo mit kleinstem Element $\perp$. Dann besitzt jede **stetige** Funktion $f : X \to X$ einen kleinsten Fixpunkt fix(f), nämlich $\mathrm{fix}(f) =_{def} \sup (f^i(\perp))$, wobei
  - $f^0(x) = \perp$ für alle $x \in X$
  - $f^{i+1}(x) = f(f^i(x))$ für alle $x \in X$
- **Satz**:

  Die Funktion $\mathrm{fix} : (X \to X) \to X$ ist selbst stetig.

Stephen Kleene: On notation for ordinal numbers, The Journal of Symbolic Logic, **3 (1938), 150-155.

Operationelle und denotationelle Semantik

Gegeben sei eine formale Sprache L

- Die **operationelle Semantik** beschreibt die Semantik eines Ausdrucks von L durch Rechenregeln, die sukzessive angewandt werden. Wert eines Ausdrucks ist dann das Endergebnis der Rechnung.
  - Beispiel: Substitutionssemantik von SML
  - Vorteile: Konkret, einfach verständlich
- Die **denotationelle Semantik** weist jedem Ausdruck von L einen abstrakten mathematischen Wert zu.
  - Beispiel: Fixpunktsemantik für rekursive Funktionen
  - Vorteile: Implementierungsdetails werden versteckt, mathematische Beweismethoden (Gleichungsschließen, Abstiegsfunktion, Induktion) werden verfügbar gemacht.

12.2 Fixpunktsemantik von BNF-Grammatiken

- Eine einfache Form der BNF-Grammatik hat die Form:
  - Endliche Menge A von Terminalsymbolen
  - Ein Nichtterminalsymbol S
  - Eine Regel der Form

    S ::= E

    wobei E ein Ausdruck ist, der aufgebaut ist aus
    - den Terminalsymbolen aus A
    - dem Nichtterminalsymbol S
    - Ausdrücken der Form

      E1 E2 (Konkatenation), E1 | E2 (Alternative), ε („leeres Wort“)
- Beispiele
  - **„1en gefolgt von einer 0“:** `S ::= „0“ | „1“ S`
  - **Palindrome**: `S ::= ε | „0“ | „1“ | „0“ S „0“| „1“ S „1“`
- Bemerkung
  - Allgemeine BNF-Grammatiken erhält man, wenn man mehrere Nichtterminalsymbole erlaubt.

CPO für BNF-Grammatiken

- Gegeben eine möglicherweise unendliche Menge A von Terminalsymbolen. Dann sei ($\wp(A^*)$, $\subseteq$) der cpo für BNF-Grammatiken.
- Die folgenden Operationen über $\wp(A^*)$ sind monoton und stetig.
  - Konkatenation $M1 \circ M2 = \{w_1 w_2 \mid w_1 \in M1, w_2 \in M2\}$
  - Vereinigung $M1 \cup M2$

Semantikfunktion für BNF-Grammatiken

- Wir definieren eine Semantik-Funktion $[\![E]\!]_{env}$, die abhängig von der Umgebung env jedem BNF-Ausdruck E einen Wert aus $\wp(A^*)$ zuweist.
  - $[\![\varepsilon]\!]_{env} = \{\varepsilon\}$
  - $[\![S]\!]_{env} = env(S)$
  - $[\![a]\!]_{env} = \{a\}$ für $a \in A$
  - $[\![E1\ E2]\!]_{env} = [\![E1]\!]_{env} \circ [\![E2]\!]_{env}$
  - $[\![E1 \mid E2]\!]_{env} = [\![E1]\!]_{env} \cup [\![E2]\!]_{env}$
- Die Semantik weist jedem Ausdruck E eine stetige Funktion zu, die wir ebenfalls mit $[\![E]\!]$ bezeichnen:
  - $[\![E]\!] : \wp(A^*) \to \wp(A^*)$,
  - Für alle $M \in \wp(A^*)$: $[\![E]\!](M) =_{def} [\![E]\!]_{envM}$ mit $envM(S) =_{def} M$

Denotationelle Semantik für BNF-Grammatiken

- Die denotationelle Semantik einer BNF-Grammatik der Form

  S ::= E

  ist definiert als kleinster Fixpunkt

  $\mathrm{fix}\ [\![\mathtt{E}]\!]$

  der zu E gehörigen Semantikfunktion.

- Gemäß dem Satz von Kleene gilt:

  $\mathrm{fix}\ [\![\mathtt{E}]\!] = \sup\ ([\![\mathtt{E}]\!]^{\mathtt{i}})_{i \in N}$

  wobei

  $[\![\mathtt{E}]\!]^{\mathtt{0}} = \varnothing$,

  $[\![\mathtt{E}]\!]^{\mathtt{i+1}} = [\![\mathtt{E}]\!]_{envi}$ mit $envi(S) = [\![\mathtt{E}]\!]^{\mathtt{i}}$

Denotationelle Semantik für BNF-Grammatiken

- Beispiel
  - Die denotationelle Semantik der BNF-Grammatik

    `S ::= „0“ | „1“ S`

    ist definiert als kleinster Fixpunkt

    fix $[\![$„0“ | „1“ S$]\!]$
- Gemäß dem Satz von Kleene gilt:

  fix $[\![$„0“ | „1“ S$]\!]$ = sup $([\![$„0“ | „1“ S$]\!]^i)_{i \in N}$

  wobei

  $[\![$„0“|„1“S$]\!]^0 = \emptyset$,

  $[\![$„0“|„1“S$]\!]^1$ = $[\![$„0“|„1“S$]\!]_{envi}$ (mit envi(S) = $\emptyset$) = {0}

  $[\![$„0“|„1“S$]\!]^2$ = $[\![$„0“|„1“S$]\!]_{envi}$ (mit envi(S) = {0}) = {0, 10}

  $[\![$„0“|„1“S$]\!]^3$ = $[\![$„0“|„1“S$]\!]_{envi}$ (mit envi(S) = {0, 10}) = {0, 10, 110}

  . . .
- Also fix $[\![$„0“|„1“S$]\!]$ = sup $([\![$„0“|„1“S$]\!]^i)_{i \in N}$ = $\{ 1^n 0 \mid n >= 0 \}$

Denotationelle Semantik für BNF-Grammatiken

- Beispiel
  - Die denotationelle Semantik der BNF-Grammatik
    `S ::= Pal mit Pal = ε | „0“ | „1“ | „0“S„0“ | „1“S„1“`
    wird folgendermaßen berechnet:
  - Gemäß dem Satz von Kleene gilt:
    $\text{fix } [\![\texttt{Pal}]\!] = \sup ([\![\texttt{Pal}]\!]^{i})_{i \in N}$

wobei

$[\![\texttt{Pal}]\!]^{0} = \emptyset$,

$[\![\texttt{Pal}]\!]^{1} = [\![\texttt{Pal}]\!]_{envi}$ (mit envi(S) = $\emptyset$) = $\{\varepsilon,0,1\}$

$[\![\texttt{Pal}]\!]^{2} = [\![\texttt{Pal}]\!]_{envi}$ (mit envi(S) = $\{\varepsilon,0,1\}$) =

$\{\varepsilon,0,1, \quad 00, 11, \quad 000, 101, 010, 111\}$

$[\![\texttt{Pal}]\!]^{3} = [\![\texttt{Pal}]\!]_{envi}$ (mit envi(S) = $[\![\texttt{Pal}]\!]^{2}$) =

$\{\varepsilon,0,1, \quad 00, 11, \quad 000, 101, 010, 111, 0000, 1001, 0110, 1111,$
$00000, 10001, 01010, 11011, 00100, 10101, 01110, 11111\}$

. . .

- Also $\text{fix } [\![\texttt{Pal}]\!] = \sup ([\![\texttt{Pal}]\!]^{i})_{i \in N} = \{ w(\text{rev } w) \mid w \in \{1,0\}^* \}$

12.3 Denotationelle Semantik rekursiver Funktionen

- Wir betrachten eine einfache Subsprache von SML.
  - Typausdrücke
    - `bool, int`
    - `typ1 * typ2, typ1 -> typ2`
  - Wertausdrücke
    - Konstanten und Standardfunktionen für `bool` und `int`
    - Identifikatoren
    - `if E0 then E1 else E2`
    - `fn Id => E`
    - `E1 (E2)`
    - `let val Id = E0 in E1`
    - `fun Id1 Id2 = E`
- Zur Vereinfachung interpretieren wir nur wohltypisierte Typ- und Wertausdrücke.

CPOs für SML

- Jedem Typausdruck `typ` ordnen wir eine cpo W(`typ`) folgendermaßen zu:
- W(`int`) = {Integer Konstanten} mit diskreter Ordnung
- W(andere Basistypen) = analog
- W(`typ1 -> typ2`) = $[$W(`typ1`) $\rightarrow$ W(`typ2`)$_\perp]$
- W(`typ1 * typ2`) = W(`typ1`) $\times$ W(`typ2`)

CPO der Umgebungen

- Sei Γ = {`x1: typ1, … , xn: typn`} eine Typzuweisung (endliche Funktion von Bezeichnern auf Typen).
- Eine **Umgebung** U für Γ weist jedem Bezeichner in dom(Γ) ein Element U(x) $\in$ W(Γ(x)) zu.
- Die Umgebungen für Γ bilden eine cpo Env(Γ) mit
  - U $\blacktriangleleft$ U0, wenn U(x) $\blacktriangleleft$ U0(x) für alle x $\in$ dom(Γ)

Denotationelle Semantik von SML-Ausdrücken

- Zu jedem Wertausdruck `E : typ` definieren wir eine stetige Funktion

  $$\mathrm{W}(\mathtt{E}) : \mathrm{Env}(\Gamma) \to \mathrm{W}(\mathtt{typ})_\perp$$

  durch Rekursion über den Aufbau von E wie folgt.
- Wir schreiben wie üblich $\mathrm{W}_\mathrm{U}(\mathtt{E})$ statt W(E)(U).

Auswertungsregeln

- Ist `c` eine Konstante, so ist W_U(`c`) = `c`,
- Ist `x` ein Identifikator, so ist W_U(`x`) = w, falls U(`x`)= w.
  - Beachte: Es muss `x` in dom(U) sein.
- Ist `op` ein Infixoperator (aber nicht `andalso, orelse`), welcher eine Basisfunktion ⊗ bezeichnet, so ist
  - W_U(`E1 op E2`) = W_U(`E1`) ⊗ W_U(`E2`) , falls W_U(`E1`), W_U(`E2`) beide ungleich ⊥.
  - Ansonsten ist das Ergebnis ⊥.
- Einstellige Basisfunktionen wie `not` und ~ sind analog.

Semantik von Funktionstermen und `let`

- Sei `E` eine Funktionsanwendung der Form `E1 E2`.
  1. W_U(`E1(E2)`) = W_U(`E1`) (W_U(`E2`)) , falls W_U(`E1`), W_U(`E2`) beide ungleich $\perp$.
  2. Ansonsten ist das Ergebnis $\perp$.
  - **Bemerkung**: Auch im Fall 1. kann das Ergebnis = $\perp$ sein, da W(`typ1->typ2`) = [W(`typ1`) $\rightarrow$ W(`typ2`)$_\perp$]

- Sei `E = fn x => E0` und Γ ▸ `E : typ1 -> typ2`.
  Dann ist W_U(E) diejenige Funktion, die w $\in$ W(`typ1`) auf $W_{U+\{<x,w>\}}$(E0) $\in$ W(`typ2`)$_\perp$ abbildet.

- Sei `E` von der Form `let val x = E0 in E1 end`.
  1. W_U(E) = $W_{U+\{<x,w>\}}$(`E1`), falls W_U(`E0`) = `w`.
  2. Ansonsten ist das Ergebnis $\perp$.

Semantik von `if`, Boole'schen Operatoren und Tupeln

- Sei E ein bedingter Term der Gestalt `if E0 then E1 else E2`.
  1. W_U(E) = W_U(`E1`), falls W_U(`E0`) = `true`.
  2. W_U(E) = W_U(`E2`), falls W_U(`E0`) = `false`.
  3. Ansonsten ist das Ergebnis $\perp$.

- W_U(`E1 orelse E2`) = W_U(`if E1 then true else E2`)
- W_U(`E1 andalso E2`) = W_U(`if E1 then E2 else false`)

- Sei E = (E1, E2).
  1. W_U(E) = (W_U(`E1`), W_U(`E2`)), falls falls W_U(`E1`), W_U(`E2`) beide ungleich $\perp$.
  2. Ansonsten ist das Ergebnis $\perp$.

Semantik rekursiver Funktionen

- Sei E eine rekursive Funktionsdeklaration der Gestalt

  `fun  f x = E1`

  $W_U(E)$ ist definiert als kleinster Fixpunkt

  `fix G mit G = (fn F => fn w` $\Rightarrow W_{U+\{<f,F>, <x,w>\}}(E)$),

- Gemäß dem Satz von Kleene gilt:

  $\text{fix } G = \sup (G^i)_{i \in N}$

  wobei

  $G^0 = \Omega$, (wobei $\Omega(x) = \perp$ für alle x)

  $G^{i+1} = G(G^{i+1})$.

Beispiel: Fakultät

- Sei `E` gleich

  `fun fakt = fn x => if x=0 then 1 else x*fakt(x-1)`

- $W_{\emptyset}(E) = \text{fix}\ \Phi$ mit
- $\Phi(f) =$ `fn w =>`

  $W_{U+\{<fakt,f>,\ <x,w>\}}$`(if x=0 then 1 else x*fakt(x-1))`

- Es gilt

  $\Phi(f)\ (w) =$ `if w=0 then 1 else w*f(w-1)`

Bestimmung des Fixpunkts

- Wir berechnen $\Phi^i(\Omega)$:

  Es ergibt sich folgende Wertetabelle:

	< 0	0	1	2	3	4	5	< 5
Ω	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
$\Phi(\Omega)$	$\perp$	1	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
$\Phi^2(\Omega)$	$\perp$	1	1	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
$\Phi^3(\Omega)$	$\perp$	1	1	2	$\perp$	$\perp$	$\perp$	$\perp$
$\Phi^4(\Omega)$	$\perp$	1	1	2	6	$\perp$	$\perp$	$\perp$
$\Phi^4(\Omega)$	$\perp$	1	1	2	6	24	$\perp$	$\perp$

- Wir sehen, dass sup $(\Phi^i(\Omega))$ = `fn w => w!` .

Zusammenfassung

- Berechnung von Fixpunkten wird in der Informatik in unterschiedlichen Situationen benötigt, wie etwa zur Beschreibung der Semantik rekursiver Funktionen.
- Der **Satz von Knaster und Tarski** besagt, dass jede **monotone** Funktion $f : X \rightarrow X$ einen kleinsten Fixpunkt besitz, falls X ein ω-cpo mit kleinstem Element ist.
- Ist f sogar **stetig**, so läßt sich der kleinste Fixpunkt nach dem **Fixpunktsatz von Kleene** als Supremum einer Kette von Approximationen berechnen.
- Der Satz von Kleene ist auf **BNF-Grammatiken** anwendbar und liefert für diese eine denotationelle Semantik über dem teilmengengeordneten cpo der Mengen von Wörtern.
- Ebenso dient der Satz von Kleene als Basis für eine denotationelle Semantik rekursiver Funktionen und damit von SML.