## Warum Denotationelle Semantik?

- Abstraktere Betrachtung:
  Wenn $c_1 \sim c_2$, dann “gleiche Bedeutung”
- Programme in verschiedenen Programmiersprachen vergleichen
- Kompositionalität und referentielle Transparenz:

$$\mathcal{D}[\![\phi(e_1, \ldots, e_n)]\!] = \Phi(\mathcal{D}[\![e_1]\!], \ldots, \mathcal{D}[\![e_n]\!])$$

## Denotationelle Semantik

- Programme werden durch *Funktionen* denotiert.

- *Semantische Funktionen* ordnen Kommandos und Ausdrücken Bedeutungen zu:

| | |
|---|---|
| Arith. Ausdrücke | $\mathcal{E} : \textbf{Aexp} \to (\Sigma \to \mathbb{N})$ |
| Boolsche Ausdrücke | $\mathcal{B} : \textbf{Bexp} \to (\Sigma \to \mathbb{B})$ |
| Kommandos | $\mathcal{D} : \textbf{Com} \to (\Sigma \rightharpoonup \Sigma)$ |

- Programme = Partielle Funktion auf Zuständen

## Denotationelle Semantik von **IMP**: Arithmetische Ausdrücke

$$\mathcal{E}[\![n]\!] \stackrel{def}{=} \lambda\sigma \in \Sigma.n$$

$$\mathcal{E}[\![X]\!] \stackrel{def}{=} \lambda\sigma \in \Sigma.\sigma(X)$$

$$\mathcal{E}[\![a_0 + a_1]\!] \stackrel{def}{=} \lambda\sigma \in \Sigma.(\mathcal{E}[\![a_0]\!]\sigma + \mathcal{E}[\![a_1]\!]\sigma)$$

$$\mathcal{E}[\![a_0 - a_1]\!] \stackrel{def}{=} \lambda\sigma \in \Sigma.(\mathcal{E}[\![a_0]\!]\sigma - \mathcal{E}[\![a_1]\!]\sigma)$$

$$\mathcal{E}[\![a_0 * a_1]\!] \stackrel{def}{=} \lambda\sigma \in \Sigma.(\mathcal{E}[\![a_0]\!]\sigma \cdot \mathcal{E}[\![a_1]\!]\sigma)$$

## Denotationelle Semantik von **IMP**: Boolsche Ausdrücke

$$
\begin{aligned}
\mathcal{B}[\![\texttt{True}]\!] &\stackrel{def}{=} \lambda\sigma \in \Sigma.\top \\
\mathcal{B}[\![\texttt{False}]\!] &\stackrel{def}{=} \lambda\sigma \in \Sigma.\bot \\
\mathcal{B}[\![\texttt{not}\ b]\!] &\stackrel{def}{=} \lambda\sigma \in \Sigma.\neg\mathcal{B}[\![b]\!]\sigma \\
\mathcal{B}[\![a_0\ \texttt{=}\ a_1]\!] &\stackrel{def}{=} \lambda\sigma \in \Sigma.\begin{cases} \top & \text{falls } \mathcal{E}[\![a_0]\!]\sigma = \mathcal{E}[\![a_1]\!]\sigma \\ \bot & \text{falls } \mathcal{E}[\![a_0]\!]\sigma \neq \mathcal{E}[\![a_1]\!]\sigma \end{cases} \\
\mathcal{B}[\![a_0\ \texttt{<=}\ a_1]\!] &\stackrel{def}{=} \lambda\sigma \in \Sigma.\begin{cases} \top & \text{falls } \mathcal{E}[\![a_0]\!]\sigma \leq \mathcal{E}[\![a_1]\!]\sigma \\ \bot & \text{falls } \mathcal{E}[\![a_0]\!]\sigma > \mathcal{E}[\![a_1]\!]\sigma \end{cases} \\
\mathcal{B}[\![b_0\ \texttt{and}\ b_1]\!] &\stackrel{def}{=} \lambda\sigma \in \Sigma.\mathcal{B}[\![b_0]\!]\sigma \wedge \mathcal{B}[\![b_1]\!]\sigma \\
\mathcal{B}[\![b_0\ \texttt{or}\ b_1]\!] &\stackrel{def}{=} \lambda\sigma \in \Sigma.\mathcal{B}[\![b_0]\!]\sigma \vee \mathcal{B}[\![b_1]\!]\sigma
\end{aligned}
$$

## Complete Partial Orders I

- **Defn:** Eine *partielle Ordnung* $\sqsubseteq$ auf einer Menge $X$ ist
  - transitiv: $x \sqsubseteq y, y \sqsubseteq z \Rightarrow x \sqsubseteq z$
  - reflexiv: $x \sqsubseteq x$
  - antisymmetrisch: $x \sqsubseteq y, y \sqsubseteq x \Rightarrow x = y$
- **Defn:** Für $Y \subseteq X$, die kleinste oberste Schranke $\bigsqcup Y \in X$ ist:
  - $\forall y \in Y . y \sqsubseteq \bigsqcup Y$
  - $\forall z \in X . (\forall y \in Y . y \sqsubseteq z) \Rightarrow \bigsqcup Y \sqsubseteq Z$

## Complete Partial Orders II

- **Defn:** Eine partielle Ordnung $\sqsubseteq$ ist *vollständig* (complete, *cpo*) wenn für jede $\omega$-Kette $x_1 \sqsubseteq x_2 \sqsubseteq x_3 \sqsubseteq x_4 \ldots = \{x_i \mid i \in \omega\}$ eine kleinste oberste Schranke $\bigsqcup_{i\in\omega} x_i$ existiert.

  Ein cpo heißt spitz (pointed, pcpo), wenn es ein kleinstes Element $\perp \in X$ gibt.

- **Defn:** Seien $(X, \sqsubseteq)$ und $(Y, \leq)$ cpo's. Eine Funktion $f : X \to Y$ ist
  - *monoton* wenn $x \sqsubseteq y \Rightarrow f(x) \leq f(y)$
  - *stetig* wenn monoton und $f \bigsqcup_{i\in\omega} x_i = \bigsqcup_{i\in\omega} f(x_i)$

## Fixpunkte

- **Satz:** Sei $(X, \sqsubseteq)$ ein pcpo, und $f : X \to X$ stetig, dann hat $f$ einen kleinsten lFixpunkt $\mathit{fix}(f)$, gegeben durch

$$\mathit{fix}(f) = \bigsqcup_{n \in \omega} f^n(\bot)$$

- $\Sigma$ als pcpo $\Sigma_\bot$ mit neuem Element, geordnet als $\bot \sqsubseteq \sigma$. Modelliert partielle Funktionen:

$$\Sigma \rightharpoonup \Sigma \cong \Sigma \to \Sigma_\bot$$

  $\Sigma \to \Sigma_\bot$ ist ein pcpo, geordnet als

$$f \sqsubseteq g \Longleftrightarrow \forall x. f(x) \sqsubseteq g(x)$$

# Denotationelle Semantik von **IMP**: Kommandos

$$
\begin{aligned}
\mathcal{D}[\![\texttt{skip}]\!] &\stackrel{\mathit{def}}{=} \lambda\,\sigma \in \Sigma.\sigma \\
\mathcal{D}[\![X := a]\!] &\stackrel{\mathit{def}}{=} \lambda\,\sigma \in \Sigma.\sigma[\mathcal{E}[\![a]\!]\sigma/X] \\
\mathcal{D}[\![c_0;\ c_1]\!] &\stackrel{\mathit{def}}{=} \mathcal{D}[\![c_1]\!] \circ \mathcal{D}[\![c_0]\!] \\
\mathcal{D}[\![\texttt{if}\ \ b\ \texttt{then}\ \ c_0\ \texttt{else}\ \ c_1\ \texttt{fi}]\!] &\stackrel{\mathit{def}}{=} \lambda\,\sigma \in \Sigma. \begin{cases} \mathcal{D}[\![c_0]\!]\sigma & \text{falls } \mathcal{B}[\![b]\!]\sigma = \top \\ \mathcal{D}[\![c_1]\!]\sigma & \text{falls } \mathcal{B}[\![b]\!]\sigma = \bot \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{D}[\![\texttt{while}\ \ b\ \ \texttt{do}\ \ c\ \texttt{done}]\!] &\stackrel{\mathit{def}}{=} \mathit{fix}(\Gamma) \\
\text{mit } \Gamma(\phi) &\stackrel{\mathit{def}}{=} \lambda\,\sigma \in \Sigma. \begin{cases} \phi \circ \mathcal{D}[\![c]\!]\sigma & \text{falls } \mathcal{B}[\![b]\!]\sigma = \top \\ \sigma & \text{falls } \mathcal{B}[\![b]\!]\sigma = \bot \end{cases}
\end{aligned}
$$

## Äquivalenz von operationeller und denotationeller Semantik

- **Lemma:** Für $a \in \mathbf{Aexp}$, $n \in \mathbb{N}$, $\mathcal{E}[\![a]\!]\sigma = n \iff \langle a, \sigma \rangle \to n$
  Beweis: Strukturelle Induktion über $a$

- **Lemma:** Für $b \in \mathbf{BExp}$, $t \in \mathbb{B}$, $\mathcal{B}[\![b]\!]\sigma = t \iff \langle b, \sigma \rangle \to t$
  Beweis: Strukturelle Induktion über $b$

- **Lemma:** Für $c \in \mathbf{Com}$, $\langle c, \sigma \rangle \to \sigma' \Rightarrow \mathcal{D}[\![c]\!]\sigma = \sigma'$
  Beweis: Induktion über der Herleitung von $\langle c, \sigma \rangle \to \sigma'$.

## Äquivalenz von operationeller und denotationeller Semantik

- **Satz (Äquivalenz der Semantiken):** Für $c \in \mathbf{Com}$, und $\sigma, \sigma' \in \Sigma$,

  $$\langle c, \sigma \rangle \rightarrow \sigma' \Longleftrightarrow \mathcal{D}[\![c]\!]\sigma = \sigma'$$

  Beweis: Strukturelle Induktion über $c$ mit *IFixpunktinduktion* für die Iteration.

- IFixpunktinduktion: Für pcpo $(X, \sqsubseteq)$, $f : X \rightarrow X$ stetig und $P \subseteq X$:

  - $P$ *inklusiv*: wenn $x_i \in P$ für alle $i \in \omega$, dann $\bigsqcup_{i \in \omega} x_i \in P$

  - $\bot \in P$, $x \in P \Rightarrow f(x) \in P$

  dann $\mathit{fix}(f) \in P$.