2.1.3 Fixpoints

Dienstag, 24. Mai 2016 15:00

Denotational semantics: map every Haskell expression to some mathematical object.

Now: How do we map Haskell-functions to continuous functions on corresponding domains?

Example without recursion:

```
conv :: Bool -> Int
conv = \b -> if b==True then 1 else 0
```

If function declarations have no recursion, we can assume that we already know the semantics of the right-hand sides of fct. declarations. Then conv should get the same semantics as the rhs of its declaration:

$$f : \mathbb{B}_\perp \to \mathbb{Z}_\perp$$

$$f(b) = \begin{cases} 1, & \text{if } b = True \\ 0, & \text{if } b = False \\ \perp_{\mathbb{Z}_\perp}, & \text{if } b = \perp_{\mathbb{B}_\perp} \end{cases}$$

Example with recursion:

```
fact :: Int -> Int
fact = \x -> if x <= 0 then 1 else fact(x-1) * x
```

Semantics of fact should be a fct. from $\mathbb{Z}_\perp \to \mathbb{Z}_\perp$.

We would like to compute the semantics of the rhs and then assign this semantics to fact.

Problem: rhs contains fact!

We will now present 2 solutions to this problem, i.e., 2 ways how one could define the semantics of such recursive functions. It will turn out that these 2 alternatives lead to the same result.

(Slide 33)

<u>Solution 1</u>: Replace the recursive function definition by a sequence of non-recursive function definitions:

We use an auxiliary symbol bot that is always undefined (i.e., bot has the semantics $\perp_{\mathbb{Z}_\perp}$).

$$fact_0 = \backslash x \to bot$$
$$fact_1 = \backslash x \to \text{if } x <= 0 \text{ then } 1 \text{ else } fact_0\,(x-1) * x$$
$$fact_2 = \backslash x \to \text{if } x <= 0 \text{ then } 1 \text{ else } fact_1\,(x-1) * x$$
$$\vdots$$

Now we can compute the semantics fact_0 of $fact_0$, the semantics fact_1 of $fact_1$, etc.

$$\mathit{fact}_0(x) = \perp \quad \text{for all } x \in \mathbb{Z}_\perp$$

$$\mathit{fact}_1(x) = \begin{cases} x!, & \text{for } 0 \le x < 1 \\ 1, & \text{for } x < 0 \\ \perp, & \text{for } x = \perp \text{ or } 1 \le x \end{cases}$$

$$\mathit{fact}_2(x) = \begin{cases} x!, & \text{for } 0 \le x < 2 \\ 1, & \text{for } x < 0 \\ \perp, & \text{for } x = \perp \text{ or } 2 \le x \end{cases}$$

$$\vdots$$

$fact_n$ is like fact, but the n-th recursive call is replaced by $\perp$.

To obtain the semantics $fact$ for fact, we compute the semantics for its non-recursive approximations $fact_0, fact_1, \ldots$ and then take their lub, i.e.:

$$fact = \bigsqcup \{ fact_0, fact_1, \ldots \}$$

The step from one approximation $fact_n$ to $fact_{n+1}$ is done by the following function $ff : \langle \mathbb{Z}_\perp \to \mathbb{Z}_\perp \rangle \to \langle \mathbb{Z}_\perp \to \mathbb{Z}_\perp \rangle$

$$(ff(g))(x) = \begin{cases} 1, & \text{if } x \leq 0 \\ g(x-1) \cdot x, & \text{otherwise} \end{cases}$$

ff can also be implemented as a Haskell function.

$$fact_0 = ff^0(\perp)$$
$$fact_1 = ff^1(\perp)$$
$$fact_2 = ff^2(\perp)$$
$$\vdots$$

$$fact = \bigsqcup \{ ff^n(\perp) \mid n \in \mathbb{N} \}.$$

(Slide 34)

Solution 2: alternative defin. of the semantics of recursively defined functions

Idea: regard the defining equations as constraints.
The semantics should be a function that satisfies these constraints

these constraints.

fact should be a function that satisfies

$$fact(x) = \underbrace{\text{if } x<=0 \text{ then } 1 \text{ else } fact(x-1)*x}_{ff(fact)}$$

In other words: fact should be a fixpoint of ff

$$ff\ (fact) = fact$$

In general: x is a fixpoint of a function f iff
$f(x) = x$.

Here: we search for a fixpoint of the function

$$ff : \langle \mathbb{Z}_\perp \to \mathbb{Z}_\perp \rangle \to \langle \mathbb{Z}_\perp \to \mathbb{Z}_\perp \rangle$$

$$(ff\ (g))\ (x) = \begin{cases} 1, & \text{if } x \leq 0 \\ g(x-1)\cdot x, & \text{otherwise} \end{cases}$$

The only fixpoint of the function ff is:

$$f(x) = \begin{cases} 1, & \text{if } x \leq 0 \\ x!, & \text{if } x > 0 \\ \perp, & \text{if } x = \perp \end{cases}$$

$$(ff\ (f))\ (x) = \begin{cases} 1, & \text{if } x \leq 0 \\ \underbrace{f(x-1)\cdot x}_{x!}, & \text{if } x > 0 \\ \perp, & x = \perp \end{cases}$$

In general, a function can have several fixpoints:

non_term :: Int -> Int

non_term = \x -> non_term (x+1)

The semantics of non_term should now be a fixpoint of the following higher-order function nn :

nn :: (Int -> Int) -> (Int -> Int)

nn g = \x -> g (x+1)

$(nn(g))(x) = g(x+1)$

Which functions are fixpoints of nn?
For which functions g do we have $nn(g) = g$?

All constant functions!

For the semantics, we should take the smallest fixpoint (w.r.t. $\sqsubseteq$), i.e., the fixpoint that is "as undefined as possible". So the semantics of non_term is $g: \mathbb{Z}_\perp \to \mathbb{Z}_\perp$ with $g(x) = \perp$ for all $x \in \mathbb{Z}_\perp$.

We usually call this the least fixpoint (lfp).

We now have 2 alternative definitions for the semantics of recursively defined functions:

(Slide 35)

$$lfp\ ff = \bigsqcup \{ ff^i(\perp) \mid i \in \mathbb{N} \}$$

These 2 definitions are equivalent!

Thm 2.1.17 (Fixpoint Theorem, Tarski + Kleene)

Let $\sqsubseteq$ be a cpo on D and let $f: D \to D$ be continuous.
Then f has a least fixpoint and we have
$\mathrm{lfp}\ f = \bigsqcup \{ f^i(\bot) \mid i \in \mathbb{N} \}$.

Proof: 1) Show that $\{ f^i(\bot) \mid i \in \mathbb{N} \}$ is a chain ($\rightsquigarrow \underbrace{\bigsqcup}_{\text{exists}} \{ f^i(\bot) \mid i \in \mathbb{N} \}$)

To this end, we prove $f^i(\bot) \sqsubseteq f^{i+1}(\bot)$ by induction on i.

Ind. Base ($i=0$): $\underbrace{f^0(\bot)}_{\bot} \sqsubseteq f(\bot)$ ✓

Ind. Step ($i>0$): Ind. Hyp $f^{i-1}(\bot) \sqsubseteq f^i(\bot)$.

Since f is continuous, it is also monotonic (Thm 2.1.15 (a)).

Hence: $\underbrace{f(f^{i-1}(\bot))}_{f^i(\bot)} \sqsubseteq \underbrace{f(f^i(\bot))}_{f^{i+1}(\bot)}$

2) Show that $\bigsqcup \{ f^i(\bot) \mid i \in \mathbb{N} \}$ is a fixpoint of f.

$f(\bigsqcup \{ f^i(\bot) \mid i \in \mathbb{N} \}) =$ since f is continuous

$\bigsqcup f(\{ f^i(\bot) \mid i \in \mathbb{N} \}) =$

$\bigsqcup \{ f^{i+1}(\bot) \mid i \in \mathbb{N} \} =$ since $\bot$ is the smallest element

$\bigsqcup (\{ f^{i+1}(\bot) \mid i \in \mathbb{N} \} \cup \{ \bot \}) =$

$\bigsqcup \{ f^i(\bot) \mid i \in \mathbb{N} \}$

3) Show that $\bigsqcup \{ f^i(\bot) \mid i \in \mathbb{N} \}$ is smaller or equal to any fixpoint of f.

Let d be a fixpoint of f.

To show: $\bigsqcup \{ f^i(\bot) \mid i \in \mathbb{N} \} \sqsubseteq d$

It suffices to show that d is an upper bound of
$\{f^i(\perp) \mid i \in \mathbb{N}\}$.

We show $f^i(\perp) \sqsubseteq d$ by induction on i.

Ind. Base ($i=0$): $\underbrace{f^0(\perp)}_{\perp} \sqsubseteq d$ ✓

Ind. Step ($i>0$): Ind. Hyp $f^{i-1}(\perp) \sqsubseteq d$

Since f is continuous and hence, monotonic, this implies

$$\underbrace{f(f^{i-1}(\perp))}_{f^i(\perp)} \sqsubseteq \underbrace{f(d)}_{d}$$

because d is a fixpoint of f

Functions like ff that are obtained from programs are computable and therefore always continuous.