2.1.2 Monotonic and Continuous Functions (Part 1)

Dienstag, 10. Mai 2016 15:00

Semantics of a function symbol of Haskell is a certain function between domains. But we don't need all of these functions for the semantics.

For the semantics, we only use "total" functions, whose argument or result can also be $\perp$. This allows us to express <u>non-strict</u> functions that return a result $\neq \perp$ even if their argument is $\perp$.

(Slide 30)

<u>Def 2.1.6</u> (Extension and Strictness of Functions)

Let $f : A \to B$ be a function. Every function $f' : A_\perp \to B$ is called an <u>extension</u> of f iff $A_\perp = A \cup \{\perp_{A_\perp}\}$ and $f(d) = f'(d)$ for all $d \in A$.

A function $g : D_1 \times \ldots \times D_n \to D$ for domains $D_1, \ldots, D_n$ is <u>strict</u> iff $g(d_1, \ldots, d_n) = \perp_D$ whenever $d_i = \perp_{D_i}$ for some i.

Ex:

```
one :: Int -> Int
one x = 1
```

This corresponds to a function f that maps numbers from $\mathbb{Z}$ to 1. But the semantics of "one" should be a function f' that is an <u>extension</u> of f.

There are several possible extensions:

(a) $f_1'(x) = 1$ for all $x \in \mathbb{Z}$, $f_1'(\perp) = 1$ ← This should be the semantics of "one" in Haskell

(a) $f_1'(x) = 1$ for all $x \in \mathbb{Z}$, $f_1'(\bot) = 1$ ← This should be the semantics of "one" in Haskell.

(b) $f_2'(x) = 1$ — " —, $f_2'(\bot) = \bot$ ← This semantics would be used in languages with eager (innermost) evaluation.

(c) $f_3'(x) = 1$ — " —, $f_3'(\bot) = 0$

Depending on whether its argument is defined or not, this fct. returns 1 or 0. → This function solves the halting problem! So f_3' is not computable and it should not be contained in our function domain.

Functions g where $g(\bot) \neq \bot$ and

$$g(d) \neq g(\bot) \text{ for some } d \neq \bot$$

are not computable. ⇒ Such functions should be excluded from the function domain.

⇒ We restrict ourselves to <u>monotonic</u> functions.

<u>Def. 2.1.7</u> (Monotonic Functions)

Let $\sqsubseteq_{D_1}$, $\sqsubseteq_{D_2}$ be partial orders on D_1 and D_2, resp.

A function $f: D_1 \to D_2$ is <u>monotonic</u> iff

$$d \sqsubseteq_{D_1} d' \text{ implies } f(d) \sqsubseteq_{D_2} f(d').$$

So: if the argument gets more defined, then the result of f also gets more defined (or remains equal).

In our example, f_1' and f_2' are monotonic.
But f_3' is not monotonic.
All computable functions are monotonic!
More examples:

```
id :: Int -> Int
id x = x
```

The semantics of id should be the function $f: \mathbb{Z}_\perp \to \mathbb{Z}_\perp$ with $f(x) = x$ for $x \in \mathbb{Z}$ and $f(\perp) = \perp$.
(Otherwise, f would not be monotonic).
Here, f is strict.
In fact, strictness implies monotonicity.

Lemma 2.1.8 (Strictness implies Monotonicity)
Let $D_1, \ldots, D_n, D$ be domains, where $D_1, \ldots, D_n$ are flat.
If $f: D_1 \times .. \times D_n \to D$ is strict, then f is also monotonic.

Proof: Let $(d_1, \ldots, d_n) \sqsubseteq (d_1', \ldots, d_n')$.
To show: $f(d_1, \ldots, d_n) \sqsubseteq f(d_1', \ldots, d_n')$.
If $(d_1, .., d_n) = (d_1', \ldots, d_n')$, then this is trivial by reflexivity of $\sqsubseteq$.
Otherwise, there must be some $1 \leq i \leq n$ where $d_i = \perp$
(Since $D_1, \ldots, D_n$ are flat).
Thus: $f(d_1, \ldots, \underbrace{d_i}_{\perp}, \ldots, d_n) \underset{\text{str}}{=} \perp \sqsubseteq f(d_1', \ldots, d_n')$. $\boxtimes$

$\perp$ since f is strict

But since Haskell uses lazy evaluation, we also need non-strict functions for the semantics.

$$\text{cond} :: (\text{Bool}, \text{Int}, \text{Int}) \to \text{Int}$$
$$\text{cond}\ (\text{True}, x, y) = x$$
$$\text{cond}\ (\text{False}, x, y) = y$$

The semantics of cond should be a function $f : \mathbb{B}_\perp \times \mathbb{Z}_\perp \times \mathbb{Z}_\perp \to \mathbb{Z}_\perp$

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

This function is monotonic, but not strict:

$$f(\text{True}, 2, \perp) = 2.$$

How does the step from a Haskell declaration of a function to the semantics of that function work in general? To this end: regard chains of elements that become more and more defined.

<u>Def 2.1.9</u> (Chain)

Let $\sqsubseteq$ be a partial ordering on a set D. A non-empty subset $\{d_1, d_2, \ldots\}$ of D is called a <u>chain</u>

iff $\quad d_1 \sqsubseteq d_2 \sqsubseteq d_3 \sqsubseteq \dots$.

Chain on $\mathbb{Z}_\perp$: $\{\perp, 5\}$, because $\perp \sqsubseteq_{\mathbb{Z}_\perp} 5$.

Chain on $\mathbb{Z}_\perp \times \mathbb{Z}_\perp$: $\{\perp_{\mathbb{Z}_\perp \times \mathbb{Z}_\perp}, (5,\perp), (5,8)\}$, because

$$\underbrace{(\perp,\perp)}_{\perp_{\mathbb{Z}_\perp \times \mathbb{Z}_\perp}} \sqsubseteq (5,\perp) \sqsubseteq (5,8)$$

Chain on $\mathbb{Z}_\perp \to \mathbb{Z}_\perp$: $\{fact_0, fact_1, fact_2, \dots\}$ (Slide 31)

$fact_1(x)$ computes the factorial function for $x < 1$

$fact_2(x)$ — " — $x < 2$

⋮

$$fact_0 \sqsubseteq fact_1 \sqsubseteq fact_2 \sqsubseteq fact_3 \sqsubseteq \dots$$

Chains can be used to define the semantics of functions. The "limes" of the chain $fact_0 \sqsubseteq fact_1 \sqsubseteq \dots$ is the factorial function.

More precisely, we look for least upper bound of such chains.

Def 2.1.10 (Least Upper Bound)

Let $\sqsubseteq$ be a partial ordering on a set D, let $S \subseteq D$.

Then $d \in D$ is an upper bound of S iff $d' \sqsubseteq d$ for all $d' \in S$

The element d is the <u>least upper bound</u> of S (lub, "supremum") iff moreover $d \sqsubseteq e$ holds for all other upper bounds e of S. We denote the lub of S by $\bigsqcup S$.

lub of $\{\perp, 5\}$ is 5.

lub of $\{\perp, (5,\perp), (5,8)\}$ is $(5,8)$.

Here, $(5,8)$ is the only upper bound of the chain.

lub of $\{fact_0, fact_1, ..\}$ is $fact$.

This chain has infinitely many upper bounds: they are like $fact$ for arguments from $\mathbb{Z}$, but can return arbitrary results for argument $\perp$. The smallest of these upper bounds is $fact$, because it returns the result $\perp$ if the argument is $\perp$.

For base domains, it is trivial to compute lub's of chains. The following lemma shows how to compute lub's for product and function domains.

<u>Lemma 2.1.11</u> (lub's for Product and Function Domains)

Let $D_1, ..., D_n, D, D'$ be domains.

(a) Let $S \subseteq D_1 \times ... \times D_n$. For all $1 \leq i \leq n$, let

$$S_i = \{d_i \mid \text{there exists } (d_1, ..., d_i, ..., d_n) \in S\}$$

If $S = \{(\perp,\perp), (5,\perp), (5,8)\}$ then $S_1 = \{\perp, 5\}$

- $\bigsqcup S$ exists iff $\bigsqcup S_i$ exists for all $1 \leq i \leq n$

- $\bigsqcup S$ exists iff $\bigsqcup S_i$ exists for all $1 \leq i \leq n$
- If $\bigsqcup S$ exists, then $\bigsqcup S = (\bigsqcup S_1, \dots, \bigsqcup S_n)$.

$S_1 = \{\bot, 5\}$
$S_2 = \{\bot, 8\}$

(b) Let S be a set of functions from $D \to D'$. For all $i \in D$, let $S_i = \{ f(i) \mid f \in S \}$ $(\subseteq D')$.

- $\bigsqcup S$ exists iff $\bigsqcup S_i$ exists for all $i \in D$
- If $\bigsqcup S$ exists, then $(\bigsqcup S)(i) = \bigsqcup S_i$ for all $i \in D$.

<u>Proof of (a)</u>: (Proof of (b) is analogous.)

"$\Leftarrow$": To show: If all $\bigsqcup S_i$ exist, then $(\bigsqcup S_1, \dots, \bigsqcup S_n)$ is the lub of S.

We first show that $(\bigsqcup S_1, \dots, \bigsqcup S_n)$ is an upper bound of S.

Let $(d_1, \dots, d_n) \in S$. Then $d_1 \in S_1, \dots, d_n \in S_n$. Hence: $d_1 \sqsubseteq \bigsqcup S_1, \dots, d_n \sqsubseteq \bigsqcup S_n$.

This implies $(d_1, \dots, d_n) \sqsubseteq (\bigsqcup S_1, \dots, \bigsqcup S_n)$.

Now we show that $(\bigsqcup S_1, \dots, \bigsqcup S_n) \sqsubseteq (e_1, \dots, e_n)$ holds for any upper bound $(e_1, \dots, e_n)$ of S.

If $(e_1, \dots, e_n)$ is an upper bound of S, then e_1 is an upper bound of S_1, $\dots$, e_n is an upper bound of S_n. Hence: $\bigsqcup S_1 \sqsubseteq e_1, \dots, \bigsqcup S_n \sqsubseteq e_n$.

This implies $(\bigsqcup S_1, \dots, \bigsqcup S_n) \sqsubseteq (e_1, \dots, e_n)$.

"$\Rightarrow$": Now we assume that $\bigsqcup S = (u_1, \dots, u_n)$ exists.

We have to show that u_1 is lub of S_1, $\dots$, u_n is lub of S_n.

We first show that u_i is an upper bound of S_i (for any $1 \leq i \leq n$).

Let $d_i \in S_i \leadsto$ there exists $(d_1, \dots, d_i, \dots, d_n) \in S$.

As $(u_1, \dots, u_i, \dots, u_n)$ is an upper bound of S, we have $(d_1, \dots, d_i, \dots, d_n) \sqsubseteq (u_1, \dots, u_i, \dots, u_n)$. Hence: $d_i \sqsubseteq u_i$.

Now we show that $u_i \sqsubseteq e_i$ for any upper bound e_i of S_i.

If e_i is an upper bound of S_i, then $(u_1, \ldots, e_i, \ldots, u_n)$ is an upper bound of S.
Since $(u_1, \ldots, u_i, \ldots, u_n)$ is the <u>least</u> upper bound of S, we have $(u_1, \ldots, u_i, \ldots, u_n) \sqsubseteq (u_1, \ldots, e_i, \ldots, u_n)$, which implies $u_i \sqsubseteq e_i$.

$\boxed{\text{q}}$

The essential property that we need for domains D is that $\sqsubseteq_D$ must be <u>complete</u>.

<u>Def 2.1.12</u> (Complete Partial Order, cpo)
A reflexive partial order $\sqsubseteq$ on a set D is <u>complete</u> iff
(1) D has a smallest element (denoted $\bot_D$)
(2) Every chain $S \subseteq D$ has a lub $\bigsqcup S \in D$.

The domains D that we regarded up to now are indeed complete (i.e., $\sqsubseteq_D$ is a cpo).

<u>Thm 2.1.13</u> (Completeness of Domains)
Let $D_1, \ldots, D_n$ be domains.
(a) Every reflexive partial order that has a smallest element and where all chains are finite is a cpo. Thus, the flat base domains $\mathbb{Z}_\bot$, $\mathbb{B}_\bot$, $C_\bot$, $F_\bot$ are cpo's.
(b) If $\sqsubseteq_{D_1}, \ldots, \sqsubseteq_{D_n}$ are cpo's, then $\sqsubseteq_{D_1 \times \ldots \times D_n}$ is also a cpo on $D_1 \times \ldots \times D_n$.

(c) If $\sqsubseteq_{D_2}$ is a cpo, then $\sqsubseteq_{D_1 \to D_2}$ is also a cpo on $D_1 \to D_2$.

Proof of (a) and (b): ((c) is analogous to (b))

(a) If $S = \{d_1, .., d_n\}$ is a finite chain with $d_1 \sqsubseteq d_2 \sqsubseteq \ldots \sqsubseteq d_n$ then the lub of S is d_n.

($d_i \sqsubseteq d_n$ holds by transitivity $\curvearrowright$ d_n is an upper bound

For any other upper bound e of S, we also have $d_n \sqsubseteq e$.)

(b) The smallest element of $D_1 \times .. \times D_n$ is $(\perp_{D_1}, \ldots, \perp_{D_n})$.

Let $S \subseteq D_1 \times \ldots \times D_n$ be a chain, i.e.,

$S = \{ (d_1^1, .., d_n^1), (d_1^2, \ldots, d_n^2), \ldots \}$ where

$(d_1^1, .., d_n^1) \sqsubseteq (d_1^2, \ldots, d_n^2) \sqsubseteq \ldots .$

Hence: $d_i^1 \sqsubseteq d_i^2 \sqsubseteq \ldots$,

i.e., $S_i = \{ d_i \mid$ there exists $(d_1, \ldots, d_i, \ldots, d_n) \in S\}$

is also a chain for all $1 \leq i \leq n$.

Since $\sqsubseteq_{D_i}$ is a cpo for all $1 \leq i \leq n$,

the lub of S_i exists.

By Lemma 2.1.11 (a): $\sqcup S = (\sqcup S_1, \ldots, \sqcup S_n)$ exists as well. $\boxed{\text{q}}$