Denotational Semantics for Teaching Lazy Functional Programming

Olaf Chitil

RWTH Aachen

Reduction

A function definition

```
(&&) :: Bool -> Bool -> Bool
False && _ = False
True  && x = x
```

describes not just how to reduce terms

$$
\begin{array}{ll}
 & \texttt{True \&\& (True \&\& False)} \\
\leadsto & \texttt{True \&\& False} \\
\leadsto & \texttt{False}
\end{array}
$$

but is also a description of

- a mathematical function
- a value

Tables

`(&&)`	`False`	`True`
`False`	`False`	`False`
`True`	`False`	`True`

Use table look up:

$$\underbrace{\underbrace{\texttt{even 6}}_{\texttt{True}}\ \texttt{\&\&}\ \underbrace{\underbrace{\texttt{4 + 2}}_{\texttt{6}}\ \texttt{> 7}}_{\texttt{False}}}_{\texttt{False}}$$

Determination of a Table

Combination of table look up and reduction.

```
and :: [Bool] -> Bool
and [] = True
and (x:xs) = x && (and xs)
```

$$
\begin{array}{ll}
 & \texttt{and (True:[])} \\
\leadsto & \texttt{True \&\& (and [])} \\
\leadsto & \underbrace{\texttt{True \&\& True}}_{\texttt{True}}
\end{array}
$$

Laziness / Non-Strictness

(&&)	False	True
False	False	False
True	False	True

```
False && (1 == 1/0)
```

				$\perp$
				False
				$\perp$
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

$$\underbrace{\qquad\qquad \underbrace{\qquad \underbrace{\qquad}_{\perp}}_{\perp}}_{\texttt{False}}$$

value $\perp$ represents undefinedness

$\perp$ may appear anywhere in an algebraic data structure:

	$\perp$	`[]`	`False:[]`	`True:[]`	`⊥:[ ]`	`False:⊥`	...
`head`	$\perp$	$\perp$	`False`	`True`	$\perp$	`False`	...

Summary

Teach tables and $\perp$ additionally to term reduction.

- a functional program defines mathematical functions
- a function is a value
- table look up is more natural than reduction