Optional matters

Steve Powell

April 26, 2009

We define an optional generic type (analogous to say, seq) for use in specifications where optional values are to be supplied.

Contents

1 Optionals

We wanted to define a data type that was generic, something that Zed proper doesn't allow (and fuzz doesn't type-check for us):

$$Opt[X] ::= Nil \mid Just\langle\langle X \rangle\rangle$$

This fails because data type definitions aren't allowed to be generic.

Nor are they allowed to be parametric, as in Haskell:

$$\texttt{data} Opt\ a = Nil \mid Just\ a$$

so instead we define the following generic construction:

$$Opt[X] == (\lambda\, s : \mathbb{P}\, X \bullet \{0\} \pfun s)$$

where we declare an optional $Opt\ S$ as a *partial* function from a singleton set (the zero isn't significant, but we had to choose something, so what do you do?) to the set S —this essentially picks out a single element from S, or it doesn't. Exactly what we want.

[The reason we chose to use functions and not simply subsets is because that makes it simple to define the function *defaultsTo* (see below).]

1.1 Type injections *Nil* and *Just*

In order to exploit this definition in the normal way, we introduce two mechanisms, corresponding to the injections *Nil* and *Just* of the Haskell mechanism.

$$Nil[X] == \varnothing[\{0\} \times X]$$

which denotes the *Nil* ('not supplied') object in $Opt\ X$ (yes, the empty set—but the generic types here are significant for type inference), and:

$$Just[X] == (\lambda\, x : X \bullet \{0 \mapsto x\})$$

which puts a value x in the set S into the $Opt\ S$ type (where S has elements of type X).

1.2 Type deconstructor *Take*

We also define *Take*, which, given an optional which is $Just\ x$, extracts the value behind it.

$$Take[X] == (\lambda\, o : Opt\ X \bullet o\, 0)$$

Of course, *Take* is not defined on *Nil*.

1.3 Using optionals with a default mechanism

Given an optional part of the state, it would be nice to define a value that was this optional value, but defaults to something else if the optional value is not supplied (is *Nil*). This can be conveniently captured by an infix operator defaultsTo:

$$
\begin{array}{|l}
\hline
[X] \\
\_ \,\mathsf{defaultsTo}\, \_ : \mathit{Opt}\, X \times X \to X \\
\hline
\forall\, o : \mathit{Opt}\, X;\ x : X \bullet \\
\quad o \,\mathsf{defaultsTo}\, x = \mathit{Take}(\mathit{Just}\, x \oplus o) \\
\hline
\end{array}
$$

which can be used as in this small example:

$$
\begin{array}{|l}
\hline
\mathit{State} \\
\mathit{value} : \mathit{Opt}\mathbb{N} \\
\hline
\end{array}
$$

Query the state and ask for the value – if it is not set in the state (it is optional after all), we want to get a default value zero:

$$
\begin{array}{|l}
\hline
\mathit{QueryValue} \\
\Xi \mathit{State} \\
\mathit{val}! : \mathbb{N} \\
\hline
\mathit{val}! = \mathit{value} \,\mathsf{defaultsTo}\, 0 \\
\hline
\end{array}
$$

This operator can be chained, so that, if $o\mathit{Val}_1$ and $o\mathit{Val}_2$ are optional values (of the same type $\mathbb{Z}$, say), the expression:

$$o\mathit{Val}_1 \,\mathsf{defaultsTo}\, o\mathit{Val}_2 \,\mathsf{defaultsTo}\, 42$$

denotes the value in $o\mathit{Val}_1$, unless this is not specified, when it is that in $o\mathit{Val}_2$, unless *that* is not specified, in which case it is the value 42.

2 How to use this in specifications

In order to use this in a specification there are two options: one, to include the definitions of *Opt* and the like each time we wish to use them, and two, to ensure that these terms (and types) are available for *f*UZZ when it checks the document that uses them.

There are two ways to get *f*UZZ to have these definitions around beforehand, one is to extend the standard prelude (a file called `fuzzlib` contains the standard prelude and this can be manually extended) and the other is simply to supply this file to *f*UZZ before the file that makes use of it.

One drawback with these solutions is that the LaTeX formatting is still needed for the symbol defaultsTo for example. This is obtained by the definition:

```
\newcommand{\defaultsTo}{\mathbin{\sf defaultsTo}}
```

which ought to be put near the top of the file that uses this symbol.

Example The file `TestOptional.tex` is checked with the technique that supplies both files to *f*UZZ. We execute the command:

```
fuzz Optional.tex TestOptional.tex
```

and this test...

...works!