Referential transparency

From HaskellWiki

Referential transparency is an oft-touted property of (pure) functional languages, which makes it easier to reason about the behavior of programs. I don't think there is any formal definition, but it usually means that an expression always evaluates to the same result in any context. Side effects like (uncontrolled) imperative update break this desirable property. C and ML are languages with constructs that are not referentially transparent.

As an example, consider the following program in Standard ML:

```
print "h"; print "a"; print "h"; print "a"
```

which prints "haha". In an attempt to factor out the repetition, we write

```
let val ha = (print "h"; print "a")
in  ha; ha end
```

but now the laugh is on us, because "ha" is only printed once. The reason is that print's side effect is realized when ha gets bound, so we should have written

```
let fun ha () = (print "h"; print "a")
in  ha (); ha () end
```

Haskell's monadic I/O system distinguishes between *values* and *actions* like the print procedure above. So we do indeed have that

```haskell
putChar 'h' >> putChar 'a' >> putChar 'h' >> putChar 'a'
```

is equivalent to

```haskell
let ha = putChar 'h' >> putChar 'a'
in  ha >> ha
```

This example is taken from

- Philip Wadler (http://homepages.inf.ed.ac.uk/wadler/) . How to declare an imperative. *ACM Computing Surveys*, 29(3):240--263, September 1997. [1] (http://homepages.inf.ed.ac.uk/wadler/topics/monads.html)

Some definitions of referential transparency are summarized in a USENET post by Tom DeBoni [2] (http://www.cas.mcmaster.ca/~kahl/reftrans.html)

There is some debate about whether the imprecisely defined semantics of
`Int`
breaks referential transparency. For instance,
`even (maxBound :: Int)`
might to true in some contexts and to false in others. Another example is
`System.Info.os :: String`
.

One perspective is that Haskell is not just one language (plus Prelude), but a family of languages, parameterized by a collection of implementation-dependent parameters. Each such language is RT, even if the collection as a whole might not be. Some people are satisfied with situation and others are not.

Category:

- Glossary
