型システム入門メモ

maton

第 20 章　再帰型

20.1　例

演習 **20.1.1.** [⋆⋆]　Haskell による擬似的な実装を RecursiveTypes.hs に示した。

演習 **20.1.2.** [推奨，⋆⋆]　Haskell による擬似的な実装を RecursiveTypes.hs に示した。

演習 **20.1.3.** [⋆⋆]　Haskell による擬似的な実装を RecursiveTypes.hs に示した。

演習 **20.1.4.** [⋆]　Haskell による擬似的な実装を RecursiveTypes.hs に示した。項 `if false then 1 else 0` を評価すると、項 `0` が得られ、項 `if false then 1 else false` を評価すると、項 `false` が得られる。

演習 **20.1.5.** [推奨，⋆⋆]　Haskell による擬似的な実装を RecursiveTypes.hs に示した。

20.2　形式的議論

演習 **20.2.1.** [推奨，⋆⋆]　文字化けするので $\lambda \to$ \, $\mu \to$ u と書いてます。

```
fixT = \f:T->T. fold [T] (\x:(T->T). f (x x)) fold [T] (\x:(T->T).
f (x x));
```

↑誤り。どうやら Haskell のデータコンストラクタと fold/unfold スタイルが相性が良さそうなので、Haskell による擬似的な実装を RecursiveTypes.hs に示した。

演習 **20.2.2.** [⋆⋆ ↛]　[進行] 項 `t` が `fold [T] t1` の形の場合は問題ない。部分項 `t1` が値にまで評価されても項（`t` 全体で畳み込みの値になるためである。一方、`unfold [T] t1` の形の場合は注意が必要である。もし、部分項 `t1` が値になるまで評価されて、`fold [T] v` の形にならなければ、行き詰まり状態となってしまうためである。ところが、項 `t` が正しく型付けされているなら問題はない。なぜなら、項 `t` の型付け導出の最後が `T-Unfld` であるとき、部分項 `t1` は再帰型を持っており、`E-Unfld` によって値となるまで評価されれば、部分項 `t1` は `fold [T] v` の形の値を持つことが帰納法の仮定より得られるためである。

[保存] $\mu X.T$ と $\{X \mapsto \mu X.T\}T$ が同型であるという定理が所与ならば、特に問題なさそう。