023: lib/Prelude.hs の Ord クラス定義を完成させる

↑ up

- issued: 2020-04-13
- 分類： B 機能追加
- status: Closed (2020-04-15)

概要

lib/Prelude.hs の Ord クラス定義を完成させる。

調査結果

2020-04-13 (Mon)

現状では、以下のように一部をコメントアウトしないとエラーする。（以下、min の型宣言を忘れているような…）

```
class (Eq a) => Ord a where
```

そこで、未実装機能だけを確認する mycompare.hs をつくって、まずは、そちらで調査を継続する。

mycompare.hs

mycompare.hs では、以下の GuardedRhs をリネームする部分が未実装なのでコケてる：

```
GuardedRhs [([ExpStmt (InfixExp (VarExp (Name {origName = "x", namePos = (1,17), isConName = False})
```

mycompare.hs シリーズのうち、まず、mycompare4.hs を 009 で扱うことにする。

⇒ 009 の対処で nodup0.hs は通るようになったのだけど、mycompare4.hs はランタイムにて abend! するようになってしまったので、このテストケースはこちらに戻します。

2020-04-14 (Tue)

辞書の多重継承化

jout/mycompare4 を jout/mycompare4e にコピーし、まずは、手で書き直して実験してみる。Eq, Ord, Num などは interface にし、各辞書の実態は、たとえば Int の Ord 辞書は、Eq と Ord の 2 つのインターフェイスを implement する。

一部の書き換えを以下に例示する。

Dict_36_Prelude_46_Eq.java：

```
import jp.ne.sakura.uhideyuki.brt.brtsyn.*;
import jp.ne.sakura.uhideyuki.brt.runtime.*;

interface Dict_36_Prelude_46_Eq {
    abstract public Expr mk_61__61_();
    abstract public Expr mk_47__61_();
}
```

Dict_36_Prelude_46_Ord.java :

```
import jp.ne.sakura.uhideyuki.brt.brtsyn.*;
import jp.ne.sakura.uhideyuki.brt.runtime.*;

interface Dict_36_Prelude_46_Ord {
    abstract public Expr mk_60_();
    abstract public Expr mk_60__61_();
    abstract public Expr mk_62__61_();
    abstract public Expr mk_62_();
}
```

Dict_36_Prelude_46_Integer_64_Prelude_46_Eq.java:

```
import jp.ne.sakura.uhideyuki.brt.brtsyn.*;
import jp.ne.sakura.uhideyuki.brt.runtime.*;

public class Dict_36_Prelude_46_Integer_64_Prelude_46_Eq
    extends Dictionary
    implements Dict_36_Prelude_46_Eq {
    public Expr mk_61__61_(){
      return Prelude.mkI_37_Integer_46__61__61_();
    }
    public Expr mk_47__61_(){
      return Prelude.mkI_37_Integer_46__47__61_();
    }
}
```

Dict_36_Prelude_46_Integer_64_Prelude_46_Ord.java:

```
import jp.ne.sakura.uhideyuki.brt.brtsyn.*;
import jp.ne.sakura.uhideyuki.brt.runtime.*;

public class Dict_36_Prelude_46_Integer_64_Prelude_46_Ord
    extends Dictionary
    implements Dict_36_Prelude_46_Eq, Dict_36_Prelude_46_Ord {

    private Dict_36_Prelude_46_Integer_64_Prelude_46_Eq dictEq;

    public Dict_36_Prelude_46_Integer_64_Prelude_46_Ord(){
        this.dictEq = new Dict_36_Prelude_46_Integer_64_Prelude_46_Eq();
    }

    /* methods of Eq */
    public Expr mk_61__61_(){
        return this.dictEq.mk_61__61_();
    }
    public Expr mk_47__61_(){
        return this.dictEq.mk_61__61_();
    }

    /* methods of Ord */
    public Expr mk_60_(){
      return Prelude.mkI_37_Integer_46__60_();
    }
    public Expr mk_60__61_(){
      return Prelude.mkI_37_Integer_46__60__61_();
    }
    public Expr mk_62__61_(){
      return Prelude.mkI_37_Integer_46__62__61_();
    }
    public Expr mk_62_(){
      return Prelude.mkI_37_Integer_46__62_();
    }
}
```

このようにしておけば、あらゆる辞書は Dictionary 型変数に格納することができ、たとえば、Integer Ord

辞書は、Dict_36_Prelude_46_Ord にも Dict_36_Prelude_46_Eq にもキャストできる。

…が、このようにしても、また、別のランタイムエラーが生じてしまった。今度は、Eq 辞書の実態を Ord にキャストしようとして abend しており、これは、確かに変。

`--ddump-core` で確認してみると、mycompare の型は `[Prelude.Eq t15,Prelude.Ord t15] :=> (t15 -> (t15 -> Pre` となっていて、辞書は Eq, Ord の順でわたされることを期待しているのに、呼び出し側で逆順にわたしてしまっている。

…が、そもそも、mycompre の型は `Eq a, Ord a => a -> a -> Ordering` ではなく、`Ord a => a -> a -> Ordering` ではないのか。

というわけで、問題は 3 つあることになる。

- Runtime における辞書の実装を変える（interface による多重継承）
- 推論された、各項の型における述語リストを最も単純な形にする (simplify が使えそう）
- 定義側と呼び出し側で辞書渡しの順序がくいちがっていた

推論された、各項の型における述語リストを最も単純な形にする (simplify が使えそう）

述語リストが単純になっていなかったのは、simplify が呼ばれてないとかではなく、ClassEnv に親子関係が正しく登録されていなかったため。Rename.renClassDecls を修正して、Class 定義から super class 情報を読み取って、ClassEnv に追加するようにした。

ただし、現状の実装はかなり特殊なので、要改善 (026)。

これにて、mycompare の型はただしく `Forall [Star] ([Prelude.Ord a] :=> (a -> (a -> Prelude.Ordering)))` となった。

次に、DictPass における lookupDictArgs をこれに対応。そのため、Sement から ClassEnv 情報を DictPass まで引き回して利用。ClassEnv には defaults も含まれるため、DictPass で defaulting しなければいけなくなった場合に使えそう。

Runtime における辞書の実装を変える（interface による多重継承）

これは、すでに手で書いてためしていたような出力になるよう、CodeGen を書き換えて完了。

mycompare4.hs は通ったので sample143 とした。

定義側と呼び出し側で辞書渡しの順序がくいちがっていた

これを試すために、↓のような関数を用いればいいかなと思ったのですが、

```
Prelude> let f x y a b = (x + y) == (a * b)
Prelude> :t f
f :: (Eq a, Num a) => a -> a -> a -> a -> Bool
```

Haskell 2010 では、Eq が Num のスーパークラスなので (参考[*1])、述語はひとつに単純化されてしまう。

[*1] https://blog.miz-ar.info/2016/06/haskell-num-class/#Eq_Show

そこで、Num と Monad が述語に現れるようなプログラムを用意した（028)。

2020-04-15 (Wed)

008 が解決したので、mycompare*.hs 系も通るようになった。通ったものは test/sample 入り。

mycompare2.hs は Prim.() の扱いに難があってエラー、これを回避した mycompare2x.hs は通った。

- mycompare.hs ok -> sample145
- mycompare2.hs NG
- mycompare2x.hs ok -> sample146
- mycompre3.hs ok -> sample147

これをうけて、lib/Prelude.hs を修正、Ord クラスの定義を完全にした。そのテストのためにつくった ordtest.hs では単項マイナスの未実装が露見。単項マイナスをつかわない ordtest2.hs は通った (sample148)。

mycompare2.hs, ordtest.hs の件はそれぞれ独立のイシューをたてる。

本件は、これでクローズする。