Generic GluonJ に向けて

小島 健介, 五十嵐 淳 (京都大学・JST CREST)
{kozima,igarashi}@kuis.kyoto-u.ac.jp

GluonJ: Java + reviser [Chiba, Igarashi, Zakirov '10]

reviser

- 既存のクラスに施したい変更の差分の記述
- 継承ではなく直接変更

```
class A {
  int answer() { return 0; }
}
```

```
reviser AnswerToEeverything extends A {
  int answer() { return 42; } // 上書き
}
```

一緒にコンパイルすると挙動が変化:
print(new A().ansewr());
=> 42

継承ではなく上書きなので A のままでよい
⇒既存のコードに手を入れずに機能拡張ができる

参照

requires

優先順位

- reviser 間の依存関係を示す
  - requires 節から決定
- reviser を適用する順番を与える
- total order になることが要求されている

```
reviser Triple requires AnswerToEverything
extends A {
  // 新しいメソッドを追加
  int ansTimesThree() {
    return 3 * super.answer(); // => 126
  }
}
```

generic reviser の使用例: derivative の記述

状況設定 (MobileMedia SPL)

- 各種メディアファイル管理アプリ
- SPL (Software Product Line) による開発
  - 機能ごとに別々のモジュールに記述
    - 扱えるメディア: Photo, Music, Video, ...
    - 可能な処理: Copy, Send, Delete, Rename, ...
  - いくつかの機能を選択してコンパイル
    ⇒選んだ機能だけをもつバージョンができる

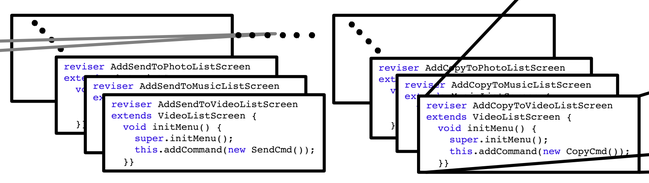
こういうの

Related work
MobileMedia SPL [Young, Murphy '05]
FeatureGluonJ [Takeyama, Chiba '12]

問題: コードの重複

- 選択された操作のための項目をメニューに追加したい
  - reviser を用いて initMenu メソッドを変更
- メディアと操作の種類の組合せの数だけ同様の処理

同じような記述がたくさん……



```
reviser AddCopyToVideoListScreen
extends VideoListScreen {
  void initMenu() {
    super.initMenu();
    // 「コピー」の項目をメニューに追加
    this.addCommand(new CopyCmd());
  }}
```

解決案

generic reviser
として一箇所にまとめる

```
reviser AddCmdToScreen<S ext MediaListScr, C ext OpCmd> ext S {
  void initMenu() {
    super.initMenu();
    this.addCommand(new C());
  }}
```

Copy, Send, Delete, Rename, ...

Photo, Music, Video, ...

意味論を与えるにあたっての問題

generic reviser の位置づけ

- 二つの方針が考えられるが, どちらが適切か?

テンプレート　vs.　それ自身が効果をもつ

型引数は明示的に与える (誰がどう与えるかが問題)
意味論は GluonJ と同様にすればよい

型引数は明示的には与えない (適用できるところに処理系が適宜適用)
メソッド呼び出しのたびにマッチングの処理

R<T1> と R<T2> の間の優先順位?

問題: GluonJ では, R requires S なら R での記述が優先としているが, この方法では R<T1> と R<T2> の順位は決まらない

- 必要に応じて処理系が勝手に決めてもよい?
  - 順位に依存しないようにプログラムを書けるならそれでよさそう
- 指定する方法を提供するとしたら, どういうものが考えられるか?
  - 必要なものだけ具体的に指定, 推論規則を書く, など?
  - 部分型関係から導かれるのが自然な場合もあるかも

具体例が足りない

メソッド呼び出しの評価手続き

- e.m(e') を評価したいとする (型は e:T, e':T')
- reviser R<X> extends V {U m(U' x){ ... } ...} を考える
- T=[S/X]V, T'<:[S/X]U' を満たす S があれば, R<S> が候補
  - 本当は X の上限などの条件も考慮
- そういう R<S> たちの中で最も優先順位の高いものを採用
- N.B. 一般に T を revise する reviser は無限個ありうる
- (ので, これをそのまま実装するのは困難?)

その他, 非自明かもしれないこと

- 型引数に対する new, extends が書きたい
- 名前の衝突の可能性
- JVM 上では型パラメータの情報がない (実装時に問題?)