レキシカル環境にメソッドを定義するオブジェクト指向言語Suzu

筑波大学情報学群情報科学類
ソフトウェアサイエンス主専攻
林　拓人

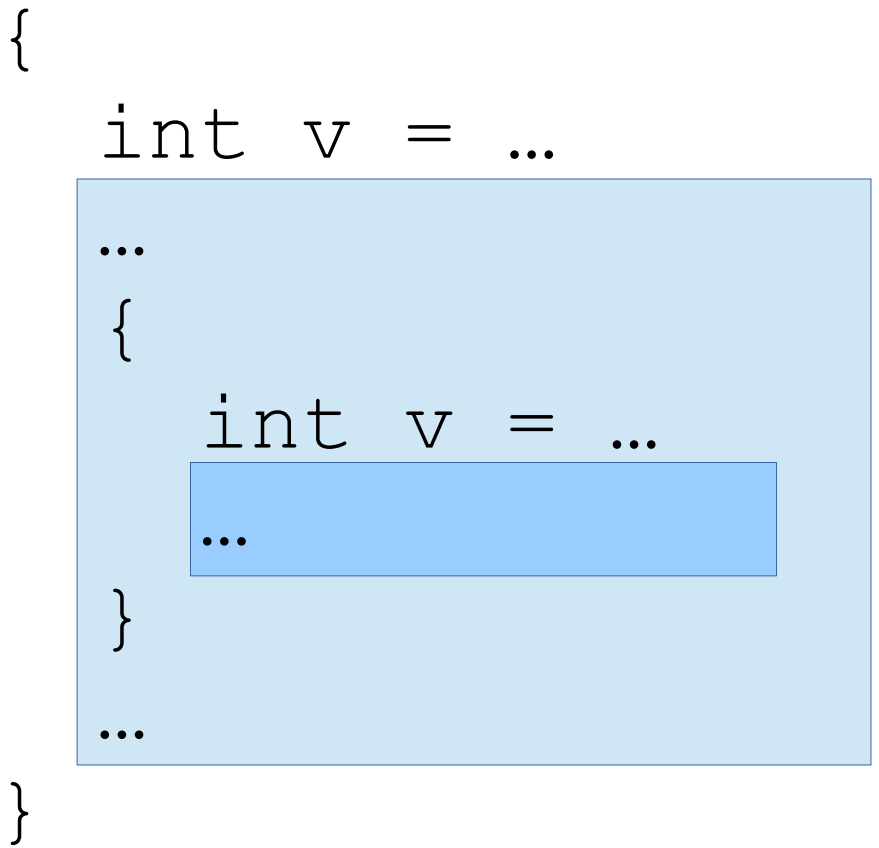
発表の流れ

- ローカル変数の有用性とローカルメソッド
- 提案：ローカルメソッドを定義可能なオブジェクト指向言語Suzu
- Suzuの活用例
- 関連研究と今後の課題

発表の流れ

- ローカル変数の有用性とローカルメソッド
- 提案：ローカルメソッドを定義可能なオブジェクト指向言語Suzu
- Suzuの活用例
- 関連研究と今後の課題

ローカル変数



- スコープはブロックや関数単位
- 局所的に追加・再定義できる
- ほとんどの言語で利用可

メソッド

```
class C {
    …
    void m(){
       …
    }
    …
}
```

- スコープはクラス単位
- ブロックや関数単位で追加・再定義はできない
- ローカルメソッドを定義できる言語は存在しない

本研究の目的

- ローカルメソッドを定義可能なオブジェクト指向言語Suzuを開発
- その有用性を実証
  - 例：内部DSL（Domain Specific Language）

発表の流れ

- ローカル変数の有用性とローカルメソッド
- 提案：ローカルメソッドを定義可能なオブジェクト指向言語Suzu
- Suzuの活用例
- 関連研究と今後の課題

ローカルメソッドへのアプローチ

- 変数定義とメソッド定義のシンタックスおよびセマンティクスの統一
  - メソッドを変数と同じように定義できるようにすればおのずとローカルメソッドを定義できる

変数とメソッドの違い

- 識別に必要な識別子の数
  - 変数：変数名1つ
  - メソッド：クラス名とメソッド名の2つ

変数

```
int v = …
```

変数名

メソッド

```
class C {
  void m(){
    …
  }
}
```

クラス名　メソッド名

変数定義とメソッド定義の統一

- クラス名とメソッド名の組を1つの識別子として扱う

変数

```
let v = …
```

変数名

メソッド

```
let C#m = …
```

クラス名とメソッド名の組

レキシカル環境

```
begin:
    let v = …
    let C#m = …
    …
    begin:
        let v = …
        let C#m = …
        …
    end
    …
end
```

- プログラムのブロック構造に対応して存在
- 変数とメソッドの内容を保持
- ブロック内のコードを実行中のみ有効
- レキシカルスコープ

発表の流れ

- ローカル変数の有用性とローカルメソッド
- 提案：ローカルメソッドを定義可能なオブジェクト指向言語Suzu
- Suzuの活用例
- 関連研究と今後の課題

組み込みオブジェクトに対するメソッドの追加

- メソッドの衝突を未然に防げる

```
begin:
  let String::C#pluralize = …
  let String::C#singularize = …
  p("person".pluralize)   //=> "people"
  p("people".singularize) //=> "person"
end
```

演算子の局所的な再定義

- 演算子の適用はメソッド呼び出し

→ローカルメソッドを定義して演算子を局所的に再定義可能

```
p(3 / 2) //=> 1
begin:
  let Int::C#(/) = …
  p(3 / 2) //=> 1.5
end
p(3 / 2) //=> 1
```

内部DSL

- グローバル環境を汚染せず可読性の高い内部DSLを作成可能

```
let regex = begin:
  open PrettyRegex
  ("foo"|"bar")+('0'-'9').one_or_more
end
p(regex) //=> "(foo|bar)[0-9]+"
```

モジュールシステム

- 変数とメソッドを統一的に扱える
- クラスによらない柔軟なグループ化
- パラメータ化されたモジュール（トレイト）による共通化・再利用
  - 多重継承に相当

発表の流れ

- ローカル変数の有用性とローカルメソッド
- 提案：ローカルメソッドを定義可能なオブジェクト指向言語Suzu
- Suzuの活用例
- 関連研究と今後の課題

関連研究

- スコープを限定してメソッドを追加・再定義
  - Classbox[Bergel et al. 2003]：ダイナミック
  - Refinements[Maeda et al. 2014]：レキシカル
  - Method Shells[竹下・千葉 2014]：ハイブリッド
- モジュール単位でメソッドを追加・再定義
- Suzuはより細かいブロック単位
  - ただしレキシカルスコープのみ

今後の課題

- 仕様
  - 継承
  - 多重ディスパッチ
  - ダイナミックスコープ
- 実装
  - ローカルメソッドの呼び出しの高速化

まとめ

- ローカル変数に対応するローカルメソッドを定義可能なオブジェクト指向言語Suzuを開発
- クラス名とメソッド名の組を用いることでメソッドを変数と同じレキシカル環境に定義
- グローバル環境を汚染しない内部DSLなどによりその有用性を示した
- 従来の研究と比べスコープをより細かく制御できる
- 機能拡張と効率的な実装が課題