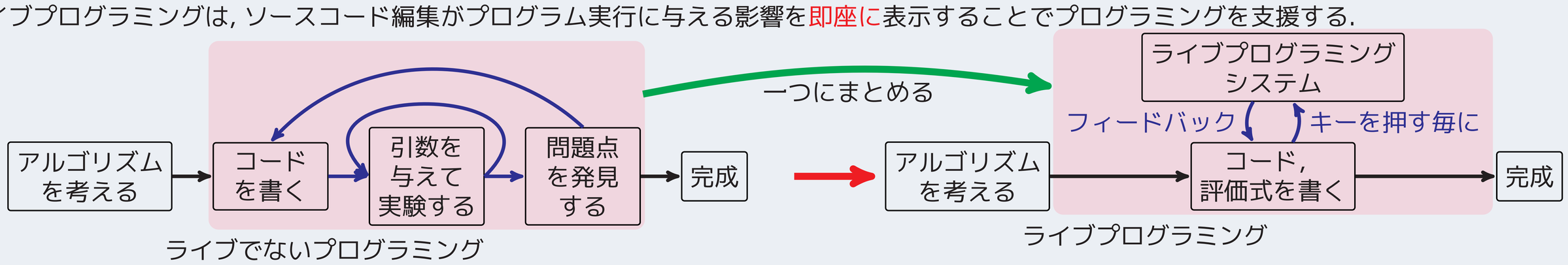
Shiranui: テストフレンドリーなライブプログラミング言語環境

今井 朝貴 [1]　増原英彦 [1]　青谷 知幸 [1]

[1]東京工業大学

ライブプログラミングとは

ライブプログラミングは, ソースコード編集がプログラム実行に与える影響を即座に表示することでプログラミングを支援する.



- 「実行する」プロセスはシステムに任せる
- 即座に受けるフィードバックにより，プログラムが実際に「どう動くのか」をすぐに理解できる

目標:ライブプログラミングと従来の開発手法との連携

- 既存のライブプログラミング環境は，おもちゃである[McDirmid,2013]
  - 既存の他の開発手法との連携があまり取られていない為
- 本研究では，単体テスト，契約による設計をライブプログラミングに取り入れた
  - 実行時情報を使用することでプログラミングを支援するため，ライブプログラミングと相性が良い

チャレンジ

- コードの近くにテストケースを置きたい
- 評価式から簡単にテストケースを作りたい
- 複雑なテストパラメータを書きたくない
- 実用的な実行速度とテストのしやすさを両立したい

Shiranuiのアプローチ

Shiranuiは簡単な言語を使用したプロトタイプである.

- 注釈記法:コードと混在/(評価式+結果)とテストケースを統合
- 実行時情報を契約にフィードバック (未達成)
- インタプリタとコンパイラ
- 再実行の必要のあるテストのみを実行 (未達成)

Shiranuiにおけるテスト駆動開発(例:階乗を計算するプログラム)

1. いくつかわかっているテスト，評価式を書く
   - “#-”からはじまる行はテストケースである.
     - “->”の左辺の評価値と右辺の評価値が一致するかを即座に表示する
   - “#+”からはじまる行は評価式である.
     - “->”の左辺の評価値を右辺に即座に表示する

自動的に挿入

```
#- fact(2) -> 2 || "No such variable: fact" ;
#- fact(3) -> 6 || "No such variable: fact" ;
#+ fact(4) -> "No such variable: fact" ;
```

2. とりあえず関数の形だけ書く

```
#- fact(2) -> 2 || 1;
#- fact(3) -> 6 || 1;
#+ fact(4) -> 1;
#+ fact(-1) -> 1;
let fact = \(n){
    return 1;
};
```

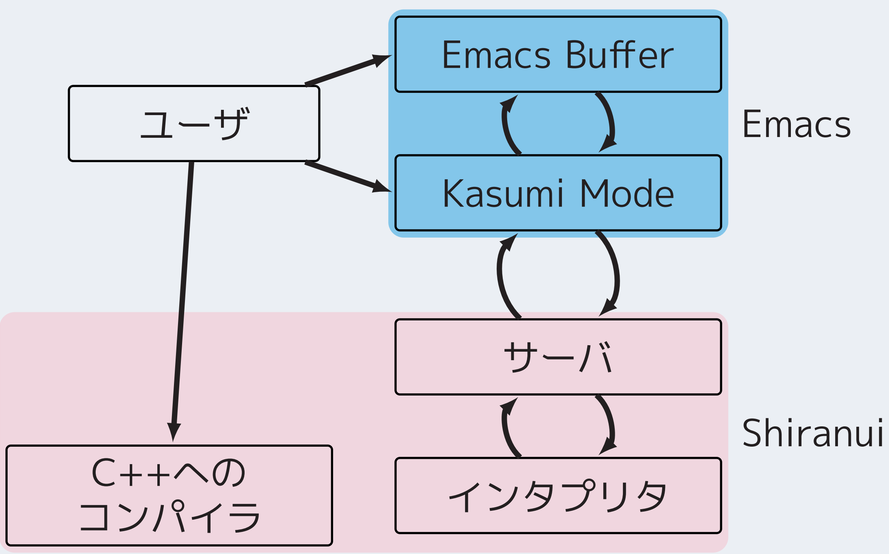
3. 関数の本体を書く.

```
#- fact(2) -> 2;
#- fact(3) -> 6;
#+ fact(4) -> 24;
#+ fact(-1) -> "Assert violated";
let fact = \(n){
    pre { assert n >= 0; }
    if n = 0 {
        return 1;
    } else {
        return n * fact(n-1);
    }
};
```

- 評価式からテストケースへは容易に変換可能である
  - 利点: 回帰テスト，テストの追加が容易
- 実際に動作するサンプルとして機能する

Shiranuiの実装 ([Tomoki Shiranui]で検索)

- Shiranui本体は，C++で実装
- エディタはEmacsを使用
- Shiranuiを非同期プロセスとして起動
- 通信はプッシュ式，Unixパイプを使用
- 独立かつ並行に実行される
  - 対応: グローバル変数によるTest fixture，停止しない場合



まとめ

ライブプログラミング言語環境Shiranuiを作成した.

- (評価式+結果)とテストケースを統合した記法を提案
  - → ライブプログラミングと単体テストを統合

これから

- probe機能[McDirmid,2013]を実装
  - printfデバッグをライブプログラミングにしたもの
- データ構造をサポート
  - “OCaml等のレコード型や代数的データ型”と“C++やJavaのclass”どっちが良い？
- 途中の関数呼び出しをテストにする機能を実装
  - 環境を含むテストパラメータの生成
  - 複雑なデータ構造を手書きする必要を無くす
- テスト失敗の表示方法を改良
- 静的型検査(Gradual Typing?)を実装
- 開発効率に関する利用者実験