定理証明支援系におけるプロパティベーステストのための網羅的関数生成

野木知優[1]、中野圭介[2]、浅田和之[2]、菊池健太郎[2]、佐藤龍之介[1]（東北大学大学院情報科学研究科[1], 東北大学電気通信研究所[2]）

概要

QuickCheck の Rocq 版である QuickChick 上で、関数に関する命題の反例を発見するために網羅的な関数生成を実現した

背景

定理証明支援系Rocq … プログラムの正しさを証明するツール

Rocqの証明の流れ

関数を定義する → 性質を表す言明の入力によって証明のゴールが定まる → ユーザが証明の記述を始める → 段階的に証明を進める → 最終的に証明すべき部分がなくなる（QED）

```
Fixpoint remove (x : nat) (l : list nat) : list nat :=
  match l with
  | [ ] ⇒ [ ]
  | h :: t ⇒ if h =? x then t else h :: remove x t
  end.
(* 自然数x とリストl を引数に取り
   リストl から自然数x を削除する関数 *)

Lemma removeP : ∀ x l, ¬ (In x (remove x l)).
(* remove x l の結果に x が含まれない *)

Proof.
  intros x l. induction l as [l h t lht].
...
  apply lHt.
...
Qed.
```

証明を進めていく中で定義が間違っていることに気づく

QuickChick … Rocqの命題に対して反例があるかを探索するコマンド

QuickChickの流れ

1. x : nat, l ; list の値をランダムに生成（**generate**）
   - x = 0,1,2 ...
   - l = [0], [0,2,1],
2. 反例が見つかったら、反例を縮小（**shrink**）
   - x = 0, l = [0,1,0]
   - x = 0, l = [0,0]
3. 反例をユーザに表示（**show**）

Gen 型クラス
・型Aの値を生成する

Shrink 型クラス
・テストで見つかった反例を単純なものにする

Show 型クラス
・型Aの値を受け取りそれを文字列として出力

```
QuickChecking removeP
0 (* 自然数 x *)
[0,0]（* リスト l *）
*** Failed after 25 tests and 1 shrinks.
```

```
Conjecture removeP :
    ∀ x l, ¬ (In x (remove x l)).
(* remove x l の結果に x が含まれない *)
QuickChick removeP.
```

関数に関する命題に対する反例の探索

```
Conjecture MapFilterP :
  ∀ (f : nat →nat) (p : nat → bool) (xs : list nat),
    filter p (map f xs) = map f (filter p xs).
```

MapFilterP :
リストに対して map と filter を適用したとき順序が結果に影響しない

Rocq の QuickChick では関数についての反例生成に対応できていない

Haskell の QuickCheck（改良版） では以下の反例が出力される

f x = 1,　p x = (x == 2),　xs = [2]

Haskell の QuickCheck における関数生成	Rocq への移植
初期実装（2000年）　**網羅的であるが表示できない** ・CoArbitrary 型クラスを用いる ・Shrink と Show ができない	CoArbitrary 型クラスは実装されているが未完成
改良版（2012年）　**表示はできるが網羅的でない** ・Table+Default 形式 関数の異なる入力に対する値を明示的に列挙したTableとそれ以外の入力に対してDefaultを設定する方法を組み合わせたもの （例. {1→0, 2→1, _→5}）	ネストされたデータ型や遅延評価による型クラスの実装が必要

目的

Rocq の QuickChick 上で関数を生成して表示すること

- Shrink, Show も可能
- 高階関数が生成可能
- 網羅的な関数生成を実現

方針

- 関数として型付きλ式を生成（de Brujin Index 形式を採用）
- 関数に現れる型（基底型）をユーザが指定
- 使用できるライブラリ関数をユーザが指定
- 得られた反例のλ式をRocq（Gallina）プログラムとして表示

今回行った実装

λ式として関数を de Brujin Index の形式で表現

```
e ::= c          定数
    | i          変数
    | f          ライブラリ関数
    | e e        関数適用
    | IfEq e e e e   条件分岐
    | λ^τ. e     関数抽象, τは引数の型
τ ::= b          基底型
    | τ → τ      関数型
```

ユーザが指定

```
IfEq a1 a2 e1 e2
```

2つの式 a_1 a_2 が等しい場合 e_1 を評価し等しくなければ e_2 を評価する

```
e ::= a                  a ::= c
    | IfEq a^b a^b e e       | i
    | λ^τ. e                 | f
                             | a e
```

β簡約できる式が含まれている

β簡約できるような式を生成しない

Table+Default 形式は IfEq により表現可能

{1→0, 2→1, _→5}　⇒　λx. IfEq(x, 1, 0, IfEq(x, 2, 1, 5))

ここでは表記上の都合のため de Brujin Index 形式を用いていない

IfEqに対する Shrink の定義

```
Fixpoint shrink_expr [ts t] (e : expr' ts t) : list (expr' ts t) :=
  match e in expr _ ts t return list (expr' ts t) with
  ...
  | IfEq b1 b2 e1 e2 ⇒
      [e1; e2] (* e1 と e2 を縮小候補として追加 *) ++ ...
  end.
```

(λx. IfEq (x + 1) 3 (x + 2) 1)

shrink

λx. x + 2　　λx. 1

ユーザが指定するライブラリ関数

- ユーザが定義し、特定の操作や計算を行うために利用される
- Rocq で提供されているもの
- レコード型 **lib_entry** としてユーザが定義

```
Definition simple_libs : list (lib_entry simple_nn_bases) :=
    [LibEntry "Nat.even" Nat.even; LibEntry "S" S].
```

関数名の文字列　　実際の関数

実装と評価

```
Conjecture MapFilterP :
    ∀ (f : Expr_simple (nat → nat)) (p : Expr_simple
(nat → bool)) (xs : list nat),
      let f := eval f in
      let p := eval p in
      filter p (map f xs) = map f (filter p xs).
QuickChick MapFilterP.
```

```
QuickChecking MapFilterP
S
Nat.even
[0]
*** Failed after 2 tests and 1 shrinks.
(0 discards)
```

f の反例：自然数の後者関数
p の反例：与えられた自然数が偶数であるかを判定する関数
xs の反例：リスト [0]

反例が見つかる → MapFilterP が不成立

```
Conjecture SetTwiceP :
  ∀ (F : Expr_simple ((nat → bool) → (nat → bool)))
    (U : Expr_simple (nat → bool)),
    let F := eval F in
    let U := eval U in
    U 0 = true → F U 0 = true → F (F U) 0 = true.
QuickChick SetTwiceP.
```

F : 集合操作
自然数の集合の操作をする関数

U : 自然数の集合

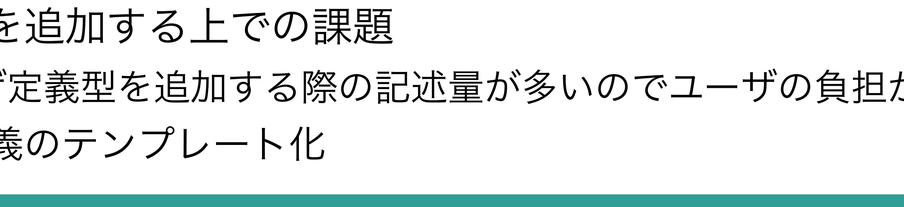
SetTwiceP :
$0 \in U \to 0 \in F\,U \to 0 \in F\,(F\,U)$

```
QuickChecking SetTwiceP
fun x0 : nat → bool ⇒ fun x1 : nat ⇒
  if Bool.eqb (x0 3) false
  then true else false
Nat.even
*** Failed after 13713 tests and 0 shrinks.
(0 discards)Time Elapsed: 1.002244s
```

F の反例：$F\,X := \begin{cases} \emptyset & (3 \in X) \\ \mathbb{N} & (3 \notin X) \end{cases}$

U の反例：偶数の集

反例が見つかる → SetTwiceP が不成立



まとめ

- RocqのQuickChick上で関数の反例生成のためのライブラリを構築
  - 型付きλ式の導入により網羅的な高階関数を生成
  - 使用できる型とライブラリ関数はユーザがカスタマイズ可能
- 実際に関数を反例としてもつ命題に対して適用できたことを確認
  - ユーザが選択したライブラリ関数により可読性の高い反例を出力
  - Haskell で表示できない高階関数の出力を実現

今後の課題

基底型を追加する上での課題

ユーザ定義型を追加する際の記述量が多いのでユーザの負担が大きい

→ 型定義のテンプレート化