神奈川大学理学部　情報科学科　馬谷研究室

論理型埋め込み DSL に対する条件付き論理変数の追加

豊島 民木 201703412

1　はじめに

本研究で扱う論理型埋め込み DSL とは miniKanren[1] のことを指す．miniKanren は Racket[2] プログラム中に論理コードを埋め込むことが可能なプログラミング言語である．論理型プログラムとは, どうやるのかではなく何が欲しいかを記述する関係定義を用いたプログラムである．例えば 2 乗の関係を表すとき，squareo (3,x) は 3 を底，x を解とすると x を 9 に結びつけて成功する．一方 squareo(y,9) は y が-3 や 3 と結びついたときに成功する．squareo(w,v) とすると w の 2 乗が v であるような全ての w，v の組み合わせを表す．このように複数の値を関連付け組として表現することができる．

miniKanren には問題点がある．それは元の Racket 関数を関係定義に書き換えることで書き換え後のプログラムの意味が変わってしまう場合があることである．これは既存の Scheme プログラムを基にして宣言的な論理コードを埋め込みたいユーザーにとって良くないことである．

本研究で, は論理変数に新たに付帯条件を付け, その条件を満たす場合にのみ解となるよう miniKanren を改良することで元の Racket 関数と同じ意味の関係定義を記述可能にする．

2　背景

membero 関係を例に，miniKanren における関係定義について説明する．membero は第 2 引数にリスト l を受け取り，その中に第 1 引数 x として受け取った値が存在する関係を定義しており次のように定義されている．

```
(defrel (membero x l)
  (conde
    ((caro l x))
    ((fresh (d)
      (cdro l d)
      (member x d)))))
```

プログラム中の conde は引数の第 1 要素を順に評価し成功した場合その引数全体を評価する関係を表す．fresh は新たに未確定な値を導入する．fresh によって導入された値を以降フレッシュな変数と呼ぶ．cdro は第 1 引数のリストの cdr 部が第 2 引数となる関係を表す．この関係を次の式で呼び出す場合を考える．

```
>(run 3 l (membero 'tofu l))
'((tofu . _0)
  (_0 tofu . _1)
  (_0 _1 tofu . _2))
```

この式は tofu を含むリストを 3 つ生成する式である．リストの生成は tofu を発見した時点で動作を終了しドット以降の値は無視されている．_0，_1，... はフレッシュな変数である．ここに問題が生じる．これらはすべての値に成り代わることができる．つまり，tofu になることも可能である．2 つ目の解の_0 に tofu を当てはめた場合，1 つ目の解 (tofu . _0) と重複してしまうことになる．これは membero の表現しようとしている本来の意味に反している．

この問題点に着目し membero の定義を拡張し Scheme 関数定義 member?本来の意味に近づける．

3 設計

2 節の問題を解決するために式 (run 3 l (membero 'tofu l)) の解において，2 つ目の解は 1 つ目の解を持たない，3 つ目の解は 1 つ目と 2 つ目両方の解を持たない，つまり解を生成する時それ以前に生成した解を持ってはならないという付帯条件を付けることで解決を図る．条件の追加は membero の conde を改良して行った．conde を改良したものを conds と名付けることとする．

conds では解 l が生成される時にそれまでの解以外という付帯条件を変数 l に追加する．これは membero 定義の conde の 1 行目にある (caro l x) が成功した時に付帯条件を付けくわえ，conde の 1 行目が失敗した時は 2 行目を評価するようにすればよい．

membero 拡張後の実行イメージは次のようになる．

```
>(run 2 l (membero 'tofu l))
'((tofu . _0)
  ((_0 . (not (==this tofu)))
   tofu . _1)
  ((_0 . (not (== this tofu)))
   (_1 . (not (== this tofu)))
   tofu . _2))
```

(_0 . (not (== this tofu))) tofu . _1) でフレッシュな変数_0 に付帯条件が追加されている．this には条件が追加されるフレッシュな変数を格納する．この場合では_0 が格納されている．付帯条件である (not (== this tofu)) は this に tofu が格納されてた場合失敗し，値が違えば成功を返す式である．実際の表記では付帯条件部分は表示せず内部で行う．変数の格納データを元々の変数名のみだったものに付帯条件データを追加で持たせることで解の差別化が可能にすることができた．

4 他のプログラム例

conds は membero 以外の関係定義にも用いることができる．

シングルトンからなるリストの関係を定義した loso に着目する．この定義を次に示す．

```
(defrel (loso l)
  (conde
   ((nullo l))
   ((fresh (a)
           (caro l a)
           (singletono a))
    (fresh (d)
           (cdro l d)
           (loso d)))))
```

定義を見ると conde の 1 行目で解が確定し，そうでない場合 2 行目を実行するプログラムである．よって loso でも conds による拡張が可能である．これにより loso を用いた式生成でも membero 同様解に付帯条件を付けることができる．

5 まとめ

本研究では関係定義 membero の変数に付帯条件を追加することにより機能拡張を行い，miniKanren で表現できていなかった変数に条件を持たせることができた．これにより miniKanren を Scheme 本来の動作に近づけることができた．

参考文献

[1] D. P. Friedman, W. E. Byrd, and O. Kiselyov, J Hemman, “The Reasoned Schemer, 2nd ed.,” MIT Press, 2018.

[2] The Racket Reference. https://docs.racket-lang.org/reference/index.html