型安全性の証明付きインタプリタのための汎用ライブラリの実装へ向けて

津山 勝輝　　叢 悠悠　　増原 英彦　(東京工業大学)

1. 背景

[1] 型付き言語の設計

関数型言語で型検査器・インタプリタ実装

手作業での型安全性の証明

[2] 問題

・手作業の証明は手間がかかり、ミスしやすい
・実装と証明の一貫性を損ないやすい
・例外処理コードによる可読性の低下と実装の手間
　(理由): インタプリタは型付け不可能な項
　　(例:1 + true)の処理も定義する必要あり

[3] 証明付きインタプリタ:
依存型言語でASTとインタプリタを実装[Altenkirch CSL'99]
=> 意味論の実装がそのまま型安全性の証明になる

(利点)
・実装以外の証明作業が不要
・意味論仕様(実装)の変更と同時に証明が更新
・例外処理が不要、評価方法の記述のみで十分
　(理由): インタプリタは型付け可能な項
　　に対してのみ定義される

2. 先行研究

・依存型アプローチを実用的な言語機能に拡張
　(例)
　- 参照・可変状態[Poulsen POPL'18]
　- Middleweight Java[Poulsen POPL'18]
　- 線形型システム [Rouvoet CPP'20]
・実装者が評価方法の記述に集中できるように、
　モナディックな抽象化で複雑な証明項をカプセル化

3. 提案

証明付きインタプリタための
Agdaライブラリの実装

1. 具体的な言語機能のAgda実装
2. インタプリタ実装者が扱いやすい抽象化を開発
　既存の実装に組込可能なモナディック抽象化

4. 例: STLC+再帰関数

```
-- 型付け可能な式のみを表現する型
data Expr (Γ : List Ty) : Ty -> Set where
  . . .
  lam : Expr (A :: Γ) B -> Expr Γ (A ⇒ B)
  -- recursive function: fix (λf.λx.e)
  fix : Expr (A :: (A ⇒ B) :: Γ) B -> Expr Γ (A ⇒ B)
  app : Expr Γ (A ⇒ B) -> Expr Γ A -> Expr Γ B
```

e : Expr Γ T
⇔ Γ ⊢ e : T

$$\frac{\Gamma \vdash f : A \Rightarrow B \quad \Gamma \vdash e : A}{\Gamma \vdash \mathrm{app}\ f\ e : B}$$

型規則に対応した
コンストラクタ定義

```
-- 型付け可能な値
data Val : Ty -> Set where
  [_,_] : Env Γ -> Expr (A :: Γ) B -> Val (A ⇒ B)
  rec[_,_] : Env Γ -> Expr (A :: (A ⇒ B) :: Γ) B ->
         Val (A ⇒ B) -- closure for recursive func
```

```
-- インタプリタ & 型安全性の証明
-- Env Γ は型環境Γに適合する実行時環境
eval : Expr Γ T -> Env Γ -> Val T
. . .
eval (lam e) env = [env , e]
eval (fix e) = rec[env , e]
eval (app f e) env =
  case eval f env of λ {
    rec[env , e'] ->
      case eval e env of λ { v ->
      eval body (v :: rec[env , e'] :: env') }
    [env' , e'] ->
      case eval e env of λ { v ->
      eval e' (v :: env') }}
```

型安全性の言明
Γ ⊢ e : T ∧
⊧ env : Γ
⇒ ∃v.
env ⊢ e⇓v ∧ v:T

$$\frac{env \vdash f \Downarrow [env', e'] \quad env \vdash e \Downarrow v \quad (env', x{=}v) \vdash e' \Downarrow v'}{env \vdash \mathrm{app}\ f\ e \Downarrow v'}$$

5. サポートしたい機能

(実装済)　バリアント + 再帰関数 + 再帰型

(今後)　Quantitative Types[Atkey LICS'18]
多段階計算