日本ソフトウェア科学会第24回プログラミングおよびプログラミング言語ワークショップ(PPL 2022), Mar., 2022

サイズ情報を伴った再帰データ型を扱う小規模組込みシステム向けFRP言語へのパラメータ多相の導入

白井瑞貴・横山陽彦・森口草介・渡部卓雄 (東京工業大学)

概要

- サイズ情報を伴った再帰データ型を扱う言語Emfrp^{BCT}にパラメータ多相を導入
- 式の型付けと使用メモリ量に関する健全性が多相の導入後も成り立つことを証明
- 多相の導入によって例題を効率的に記述できることを確認

Emfrp [1]

- 関数リアクティブプログラミング (FRP) 言語
  - 時変値 (時間変化する値) の関係を宣言的に記述
- 小規模組込みシステム向け
  - 実行時の使用メモリ量の上限を静的に決定可能
  - 再帰データ型・再帰関数を禁止
  - 関数・時変値が第一級オブジェクトでない

[1] Sawada et al, Emfrp: A Functional Reactive Programming Language for Small-Scale Embedded Systems. MODULARITY Companion (CROW 2016), pp. 36-44, 2016.

Emfrp^{BCT} [2]

- 使用メモリ量の上限を静的に決定可能
- 再帰データ型・再帰関数が利用可能

- 型にサイズパラメータを付与
  - 値に含まれるコンストラクタ数の上限を表す
  - 定数・サイズ変数・加減算で構成
  - 例: `List[3]` `List[n]` `List[n+1]`

```
type List = Nil | Cons(Int, List)

func length(l: List[n]): Int where {n>0} [n] =
  case l return Int of
  | Nil -> 0
  | Cons(hd, tl) -> 1 + length(tl)
```

サイズパラメータ　事前条件　再帰の尺度

課題

型と関数が型についての多相性を持たないため、汎用データ構造のライブラリ化が困難である

[2] Yokoyama et al, A Functional Reactive Programming Language for Small-Scale Embedded Systems with Recursive Data Types. Journal of Information Processing, Vol. 29, pp. 685-706, 2021.

提案言語: Emfrp^{PBCT}

- 型引数を用いて型や関数を総称的に定義
- 汎用データ構造 (リスト等) の円滑なライブラリ化が可能

```
type List<T> = Nil | Cons(T, List)

func insert<T>(x: T, l: List<T>[n]): List<T>[n+1]
    where {n>0} [n] =
  case l return List<T>[n+1] of
  | Nil -> Cons(x, Nil adj[n])
  | Cons(hd, tl) ->
      if compare(x, hd) < 0 then Cons(x, l)
      else Cons(hd, insert(x, tl))
func sort<T>(l: List<T>[n]): List<T>[n]
    where {n>0} [n] =
  case l return List<T>[n] of
  | Nil -> Nil adj[n]
  | Cons(hd, tl) -> insert(hd, sort(tl))
```

リストの挿入ソート

実型引数のサイズパラメータに関する制限

サイズパラメータにサイズ変数を含む型は型引数に束縛できない
- `List< List<Int>[3] >[n]` ← 定数なので可能
- `List< List<Int>[n] >[n]` ← サイズ変数を含むので不可

compare式

- 任意の型に自動で定義される構造的な大小比較 (cf. OCaml)
  - コンストラクタを定義順に従って大小比較
  - コンストラクタが一致するとき辞書順でパラメータを再帰的に比較
- リストのソート、ヒープ木などを記述可能
- 静的ディスパッチ
  - 関数がcompareを含む場合、実型引数によって異なるコードを生成

例題

要素の型が異なるリストを2つ用いる例題

- 入力時変値 `v`: センサの値　`t`: 時刻
- 出力時変値 `mv`: `v` の過去7件の中央値
  (`mmv`, `mmt`): 組(`mv`, `t`) の過去15件の中央値

- 中央値を求めるための履歴をリストで保持
- Emfrp^{BCT} では型と中央値を得る関数の再定義が必要
- Emfrp^{PBCT} ではコードの再利用が可能

評価実験
- 例題をEmfrp^{BCT}とEmfrp^{PBCT}でそれぞれ実装
- コンパイル結果のバイナリサイズを比較
- 入力を10^7回与えて実行時間を比較

	実装の行数	text [byte]	data [byte]	bss [byte]	実行時間 [sec]
Emfrp^{BCT}	129	8820	40	6568	38.613
Emfrp^{PBCT}	82	7092	40	6552	29.617

- バイナリサイズは縮小 (多相の関数がコードを共有するため)
- 実行時間は増加なし

→多相の導入によるオーバーヘッドがないことを確認

形式化

先行研究 Emfrp^{BCT} で成り立つ性質
「適切に型がつく式は静的に決定したメモリ量で評価できる」
が Emfrp^{PBCT} でも成り立つことを証明

$$[s]E \mid [t]H \vdash^{\mathcal{T};\mathcal{F}}_{\mathcal{L}} e \Downarrow_u l; [t']H'$$

使用メモリ量を明示する操作的意味論

$$\Gamma \vdash^{\mathcal{T};\mathcal{F}}_{f} e : \tau \mid C$$

式の型付け規則

$$H \vdash_{\mathcal{T}} l : \sigma$$

値の型付け規則

$$\mathcal{M}^{\mathcal{T}}_{\mathcal{F}}(\Delta; \eta; \Gamma)[\![e]\!] = (\tau, s, t, u)$$

使用メモリ量決定アルゴリズム

E スタック　s スタックの空き容量
H ヒープ　t ヒープの空き容量
l ヒープ上の位置
u 関数呼び出しの深さ限界
τ 型　σ サイズパラメータが定数の型
C サイズに関する制約式
Δ サイズ変数と定数の対応
η 仮型引数と実型引数の対応

今後の課題

- 実型引数のサイズパラメータに関する制限の緩和
- 関数のパラメータとして関数を渡す機能 (compareの代替)