A-030

ML と C 間の多相型外部関数インターフェース

Polymorphic Foreign Function Interface between ML and C

上野 雄大† 大堀 淳†
Katsuhiro Ueno Atsushi Ohori

1. 序論

プログラミング言語 ML が有する多相型システムは，安全かつ柔軟なプログラミング環境を実現する基礎をなす機構のひとつである．一方，C 言語はオペレーティングシステムや多くのライブラリの記述言語として広く用いられており，その膨大なソフトウェア資産の活用は高機能なアプリケーションを実現する上で必要不可欠である．ML と C を相互に結合し，ML の多相型と C の豊富かつ高度なライブラリ群を同時に利用可能にするプログラミング言語が実現されるならば，ソフトウェア生産の信頼性と生産性を飛躍的に高めることが可能になると考えられる．なかでも，並列プログラミングのためのライブラリである MPI などのような，様々なデータを汎用的に取り扱うことのできる C ライブラリの関数は多相的に書かれており，これら多相的な C 関数を多相性を保ったまま ML の多相型システムの下で型安全に利用可能とすることは，より実践的かつ高度なアプリケーションを記述可能な高信頼プログラミング環境の実現に向けての重要な課題のひとつである．

本稿では，多相的な振る舞いをする C 関数を ML の多相関数としてインポートするための外部関数インターフェースについて報告する．著者らはこれまで，ML と C とのシームレスな相互運用性を確立し，ML プログラムから C の既存のソフトウェア資産を直接利用するための理論の構築を行ってきた．著者らは，その成果を東北大学電気通信研究所が算譜工房の協力を得て開発を進めている次世代 ML 言語 SML#[3] に実装し，この機構によって種々のライブラリを SML# プログラムにインポートし，様々な実用アプリケーションを容易かつ型安全に記述できることを確認している．この機構は，整数や浮動小数点データなどの実行時の表現を C と共通な自然なものにする型主導コンパイル技術 [1] と，ML と C との関数表現の違いを克服しコールバックを含む言語間の関数呼び出しを可能にする外部関数とのインターフェイス技術 [2] によって実現されている．この機構は，一部の多相的な C 関数を多相関数として ML にインポートする機能を含む．しかし，その多相性はポインタ型そのものに限定されており，qsort などの型に関する情報を追加で受け取るような C 関数には対応していない．本稿では，これら既存の成果を拡張し，C の持つ多相性に関する分析を通じて，より多くの C 関数を多相的に ML にインポートする機構の構築を目指す．

2. 型情報を伴うポインタ多相性

文献 [2] では，ポインタ多相の概念を導入し，多相的な振る舞いをする C 関数を多相関数として ML にインポートする方式を示した．本稿では，より広範囲な C の多相関数を多相性を保ったまま ML にインポート可能となるように，ポインタ多相性の概念を型情報を伴うより洗練された多相性に拡張する．以下，文献 [2] で導入したポインタ多相を振り返りながら，型情報を伴うポインタ多相性を導入する．

C 言語は ML のような多相型を含む型システムを有していないにもかかわらず，いくつかの汎用性を持つ C 関数は多相的な振る舞いを持つように定義されている．C において多相的な振る舞いをする関数を扱う一つの方法は，指しているオブジェクトの型が不明なポインタ型（void*型）として引数を宣言し，プログラマはその関数の使用時に具体的な型と void*型との変換を行うというものである．その典型的な例は，C の標準ライブラリ関数である qsort 関数に見て取れる．qsort 関数は C において以下のような型で宣言されている．

```
void qsort(void *base, size_t nmemb,
           size_t size,
           int (*compar)(void*, void*));
```

この関数は，任意の型の配列をソートする．引数 base はソートする対象の配列へのポインタ，nmemb は配列中の要素の数，size は配列中の各要素がメモリを占めるバイト数，compar は 2 つの要素を比較する関数へのポインタである．プログラマは，base にある型の配列へのポインタを void*型にキャストした値を，size および compar に base が指す実際の配列の型に合わせた適切な値を与えて，qsort 関数を呼び出す．qsort 関数の実装は，void*型のポインタ値に引数 size に与えられた要素のバイト数を加えて配列内の各要素のアドレスを計算し，compar に与えられた関数を呼び出して 2 つの要素を比較し，要素のバイト数分のメモリコピーによって破壊的な配列要素の入れ替えを行う．このようなポインタ型のキャストとポインタ演算の組み合わせによって任意の型の値を操作することができる関数を定義する手法は，qsort だけでなく，多くの汎用性のある C ライブラリにおいて多相的な振る舞いをする関数を定義するために用いられている．

型情報を必要とする多相的な C 関数の動作を整理すると，その多相性は以下の性質に起因すると考えることができる．

1. 型の抽象は，void*型として宣言された任意の型のポインタを取る引数によって導入される．
2. 型抽象の例の生成は，ある具体的な型のポインタから void*型へのキャストによって行われる．
3. オブジェクトの大きさなど，void*型のポインタが実際に指しているオブジェクトの型に関する情報は追加の引数として取る．
4. C 関数の多相的な振る舞いは，ある特定の型に依存しないアドレス演算とメモリ領域間の単純なコピーに起因する．

†東北大学電気通信研究所, RIEC, Tohoku University

文献 [2] におけるポインタ多相性は，1 および 2 に基づくものである．本稿では 3 および 4 の性質が反映されるようにポインタ多相性の概念を拡張する．

自然なデータ表現を実現する型主導コンパイル方式 [1] では，ML の多相関数を，特定の型に依存しないコードと型に関する情報の抽象と適用に変換することで，自然なデータ表現の下で ML の多相性を実現する．特に，ポインタと互換な ML のデータ型に関しては，ポインタとそのポインタが指すデータに関する情報を受け取る同一の中間コードが生成され，このコードは中間コードの型システムにおいて，ポインタと互換な型の集合に対して多相性を持つ．従って，この型主導コンパイルの結果のコードが，void*型のポインタと追加の型情報を適切な順に並べた引数列を伴いポインタ多相 C 関数を呼び出す C コードと直接対応するコードとなるような機構を導入することによって，多相性を持つ C コードを ML の多相関数としてインポートすることが可能となる．

ポインタ多相性は，多相性を導入する void*型と，型情報を表す型に型パラメタを導入することで，パラメタ多相性として明示することができる．多相型外部関数インターフェースの基礎は，C のサブセットを型変数と型パラメタを加えて拡張した，ポインタ多相性をパラメタ多相性として明示することができる言語を通じて与えられる．この型システムの下では，qsort 関数の型は，以下のように記述することができる．

$$\texttt{qsort} : \forall t :: ptr.(\texttt{void*}(t), \texttt{size\_t}, \texttt{size}(t), (\texttt{void*}(t), \texttt{void*}(t)) \to \texttt{int}) \to \texttt{void}$$

ここで，void*(t) は，void*型と値不変で相互にキャストが可能な型 t の集合を動く型を表す．size(t) は，型 t の大きさを表す型で，t の大きさを唯一の値として持つ．ptr は t が void*型と互換な型の集合のみを動くことを表す種類である．

3. 多相型外部関数インターフェース

多相型外部関数インターフェースは，ポインタ多相型拡張を含む C のサブセット言語をターゲット言語とする型主導コンパイルアルゴリズムを通じて定義される．本稿では，qsort 関数のインポートを例に用い，その概要を述べる．

本稿で構築する外部関数インターフェースでは，以下のような構文で C 関数呼び出しを記述する．

```
fun qsort (x,y,z) =
    _ffiapply qsort
      (x:'a array, y:int, _sizeof('a),
       z:'a ptr * 'a ptr -> int) : ();
val _ = qsort (intArray:int array, 10, intCmp)
```

_ffiapply は，指定された C 関数を，指定された引数列を伴って呼び出す構文である．引数列のうち，_sizeof('a) は，型'a の大きさを与えるパラメタであることを指示する項である．qsort は qsort の多相的な呼び出しを行う ML の関数として定義され，多相型'a array * int * ('a ptr * 'a ptr) -> int を持つ．このプログラムはまず多相型推論機構によって型注釈と型適用が明示された以下の中間コードに翻訳される．

```
['a. fun qsort (x,y,z) =
    _ffiapply qsort
      (x:'a array, y:int, _sizeof('a),
       z:'a ptr * 'a ptr -> int) : ()];
val _ = [qsort int]
          (intArray:int array, 10, intCmp)
```

['a....] は型抽象を，[qsort int] は多相型に対する型適用を表す項である．多相型外部関数インターフェースを実現する型主導コンパイルアルゴリズムによって，型変数'a の束縛は型に関する情報を受け取る追加の引数に，_sizeof('a) は具体的な型の大きさを表す項に，型適用 [qsort int] は型 int の大きさを qsort への追加の引数として与える項にそれぞれコンパイルされ，以下のような中間コードとなる．

```
fun qsort (s:size('a),x,y,z) =
    qsort@CFUN
      (x:'a array, y:int, s,
       z:'a ptr * 'a ptr -> int) : ()];
val _ = qsort (4:size(int),
               intArray:int array, 10, intCmp)
```

int の大きさ 4 は，型主導コンパイルアルゴリズムによって型適用項から自動生成される．この機構によって，プログラマは，型に関する詳細に捕われることなく，qsort 関数を ML から多相的に用いることができる．配列 x および整数 y は，自然なデータ表現 [1] となるようにコンパイルされ，関数 z は関数呼び出し機構の違いを吸収するスタブコード生成 [2] によって C 関数に変換されて qsort 関数に渡される．これらの技術の組み合わせによって，C の qsort を多相関数として使用し，ML の配列を直接ソートすることが可能となる．

4. まとめ

本稿では，文献 [2] で導入したポインタ多相性の概念を拡張し，より広範囲の多相的な C 関数を ML 系言語に多相型関数としてインポートする機構について報告した．本稿で定義する方式の詳細な定義と分析を含む報告は別の機会に行う予定である．本稿で提案した外部関数インターフェースの一部は，SML# コンパイラ上に実装され公開されている．

参考文献

[1] Nguyen, H.-D. and Ohori, A.: Compiling ML polymorphism with explicit layout bitmap, *Proc. ACM Symposium on Principles and Practice of Declarative Programming*, 2006, pp. 237–248.

[2] 上野雄大, 大堀淳: SML# の外部関数インターフェース, コンピュータソフトウェアに掲載予定.

[3] SML# Compiler, available at http://www.pllab.riec.tohoku.ac.jp/smlsharp/.