LLVMバックエンド用テストプログラムのテンプレート記述

Parameterized Template for Test Programs Targeting LLVM Back-End

藤原大輔 Daisuke Fujiwara
石浦菜岐佐 Nagisa Ishiura

関西学院大学 理工学部
School of Science and Technology, Kwansei Gakuin University

1 はじめに

コンパイラには, 非常に高い信頼性が求められるため, テストスイートやランダムテスト等による徹底的なテストが行われるが, そのテストプログラムは通常そのコンパイラが処理対象とする言語で書かれる. これに対し, コンパイラ基盤 LLVM [1] を用いたコンパイラの開発では, フロントエンドの clang [2] が生成する LLVM 中間表現に対するバックエンドのポーティングを行うため, 初期のテストや特定の最適化パスを対象としたテストでは, LLVM アセンブリを入力とするテストが有用となる. しかし, 様々な型や値に対応した LLVM アセンブリを用意することは手間を要する.

そこで本稿では, パラメータ化されたテンプレートから複数のテストを生成する手法を提案する.

2 LLVM アセンブリによるテストプログラム

LLVM アセンブリによるテストプログラムの例を図 1 に示す. この例では i32 型 (整数型 32 ビット) の変数だけが使われているが, これ以外の型についても同様のテストが必要になる. しかし, 型の変更に伴って, 型変換命令の挿入が必要になることがある. さらに, 場合によっては期待値の再計算が必要になることもあり, テストの作成に大きなコストがかかる点が課題となる.

```
 1: @i = global i32 3, align 4
 2: @j = global i32 5, align 4
 3:
 4: define i32 @main {
 5: entry:
 6:     %0 = load i32* @i, align 4
 7:     %1 = load i32* @j, align 4
 8:     %add = add nsw i32 %0, %1
 9:     %cmp = icmp ne i32 %add, 8
10:     ret i32 %cmp
11: }
```

図 1 LLVM アセンブリによるテストプログラム

3 LLVM アセンブリのテンプレート記述

本稿では, 前節の問題を解決するために, パラメータ化されたテンプレートから複数のテストを生成する手法を提案する. 一旦生成されたテストプログラムに対し, 静的解析に基づいて型変換命令を挿入し, 動的解析によって期待値の自動計算を行う. これによって, 前述した点を意識せずにテストを作成することが可能になる.

テンプレート記述の例を図 2 に示す. この例では 1〜17 行が引数付きマクロであり, 19〜21 行目で引数の値を指定してプログラムを生成する. 1 行目の引数$TY1, $TY2 に対して, 20 行目で値の集合を{}で指定すると, その引数に各要素を順に代入し, それに対するテストファイルを生成する. この例では, $TY1 には i8, i16, i32, i64··· が, $TY2 には i8, i16··· が順に代入されるため, 合計 $4 \times 5 = 20$ 本のテストプログラムが生成される.

図 3 は型変換命令の自動挿入の例である. 静的解析により %1 と %add の型の相違を検出し, 必要な型変換命令を挿入する. 5, 6, 10, 11 行目の$_ALIGN は型を引数とするマクロ関数であり, target datalayout の情報に応じて, 指定した型の align 値に置換される. 13 行目の $_EXVAL はプログラムの動的解析により取得した変数の期待値に置換される.

```
 1: <define name="$T000" args="$TY1,$TY2">
 2: <file name="t000_*.ll">
 3: target datalayout = "e-p:32:32:32-i1:8:8-i8:8:8-i16:
    16:16-i32:32:32-i64:32:32-f32:32:32-f64:32:32-v64:
    64:64-v128:128:128-a0:0:1-f80:32:32"
 4:
 5: @i = global $TY1 3, align $_ALIGN($TY1)
 6: @j = global $TY2 5, align $_ALIGN($TY2)
 7:
 8: define i32 @main {
 9: entry:
10:     %0 = load $TY1* @i, align $_ALIGN($TY1)
11:     %1 = load $TY2* @j, align $_ALIGN($TY2)
12:     %add = add nsw i32 %0, %1
13:     %cmp = icmp ne $TY1 %add, $_EXVAL(%add)
14:     ret i32 %cmp
15: }
16: </file>
17: </define>
18:
19: <generate>
20: $T000({i8,i16,i32,i64},{i8,i16,i32,i64,double})
21: </generate>
```

図 2 テンプレート記述例

```
%0 = load $TY1* @i
%1 = load $TY2* @j

%add = add nsw i32 %0, %1
```

⇒

```
%0 = load $TY1* @i($TY1=i32)
%1 = load $TY2* @j($TY2=i16)
%tmp1 = sext i16 %1 to i32
%add = add nsw i32 %0, %tmp1
```

図 3 型変換命令の自動挿入

4 実装

提案手法に基づくツールを Perl5 を用いて実装した. 対応する clang のバージョンは 3.2 であり, このバージョンについて, 構造体や配列を除いた場合に正しいテストプログラムが生成されることを確認した.

5 むすび

本稿では, テンプレートから LLVM アセンブリを生成する手法を提案した. 構造体や配列への対応が, 今後の課題である.

謝辞 本研究の遂行にあたり, ご指導・ご助言を頂きました株式会社ルネサスソリューションズの福井昭也様, 下村靖弘様, 臼杵恵介様に感謝致します.

参考文献

[1] http://llvm.org/.
[2] http://clang.llvm.org/.