Aufgabe 4: Interprozedurale Konstantenpropagation

Optimierende Übersetzer WS 2015/16

Abgabetermin: 27. Jänner 2016, 16:00 Uhr

1 Interprozedurale Konstantenpropagation

Spezifizieren Sie mit PAG eine interprozedurale Konstantenpropagationsanalyse für die Sprache SL_2; diese ist die Erweiterung von SL_1 um Funktionsdefinitionen und -Aufrufe. Funktionsaufrufe dürfen in komplexeren Ausdrücken vorkommen, Sie können allerdings davon ausgehen, dass jeder Ausdruck höchstens einen Aufruf enthält.

Jede in einem SL_2-Programm aufgerufene Funktion muss vor dem Aufruf mit einer konkreten Anzahl von Parametern deklariert und irgendwo im Programm definiert werden. Funktionen dürfen mehrere Parameter vom Typ `int` haben; der Rückgabetyp jeder Funktion ist ebenfalls `int`. Jede Ausführung einer Funktion in SL_2 muss mit einer `return`-Anweisung enden. Funktionen kommunizieren ausschließlich über Argumente und Rückgabewerte, globale Variablen existieren in SL_2 nicht.

Im Folgenden ein Beispiel für ein SL_2-Programm:

```
int add(int a, int b) {
    return a + b;
}
int main() {
    int x, y;
    x = 42;
    y = 23;
    return add(x, y);
}
```

Der interprozedurale Kontrollflussgraph von SATIrE verwendet spezielle globale Variablen (ähnlich Argument- und Ergebnisregistern auf Maschinenebene), um die Übergabe von Argumentwerten und die Rückgabe von Ergebniswerten abzubilden. Dies geschieht über die folgenden Knoten:

`ArgumentAssignment` modelliert die Zuweisung eines Argumentausdrucks an eine Argumentvariable.

`FunctionCall` ist der Punkt eines Funktionsaufrufs. Von diesem Knoten gehen zwei Kanten aus: `call_edge` verbindet den Aufruf mit dem Einstiegspunkt der aufgerufenen Funktion, `local_edge` fließt innerhalb der aufrufenden Funktion zum entsprechenden Rückkehrpunkt.

`ParamAssignment` ist am Anfang jeder Funktion für die Zuweisung der Argumentvariablen an die Funktionsparameter zuständig.

`FunctionReturn` markiert den Punkt, an dem die Kontrolle von einem Aufruf zurückkehrt.

`ReturnAssignment` ist eine Zuweisung der Rückgabevariable an eine ebenfalls spezielle Variable, die den Funktionsaufruf in seinem ursprünglichen Kontext ersetzt.

Die Analyseinformation soll an jedem Programmpunkt nur Programmvariablen enthalten, die in der jeweiligen Funktion tatsächlich sichtbar sind. Lokale Variablen müssen somit spätestens am `FunctionExit`-Knoten aus der Analyseinformation entfernt werden. Von einem Funktionsaufruf sollen nur die Argumentwerte an die aufgerufene Funktion fließen, die restliche Information muss lokal weitergereicht werden. Argument- und Rückgabevariablen, identifiziert durch die Funktion `is_tmpvarid`, soll Ihre Analyse sofort nach jeder Verwendung entfernen. (Diese Variablen werden in SATIrE zwischen zwei Verwendungen immer überschrieben.)

Ihre Analyse soll zumindest dieselben Ausdrücke wie in Aufgabe 3 verarbeiten können, weiters alles, was für die Behandlung von Funktionsaufrufen notwendig ist. Abweichungen von SL_2 muss die Analyse nicht diagnostizieren, das Verhalten in solchen Fällen ist Ihnen freigestellt.

2 Auswirkungen von Call-String-Längen

Wie in der Vorlesung besprochen, kann die Genauigkeit von interprozeduralen Analysen mittels Kontextinformation erhöht werden. Eine solche Art von Kontextinformation stellen Call Strings dar; deren Auswirkungen auf die Genauigkeit der Analyse sollen hier untersucht werden. Zur Beeinflussung der Länge der von PAG automatisch verwalteten Call Strings stellt Ihr mit SATIrE generiertes Analyseprogramm das Kommandozeilenargument `--callstringlength=`n zur Verfügung, ohne Argument gilt $n = 0$.

Mit der Kombination aus interprozeduraler Analyse und Behandlung von Bedingungen kann die Analyse die Werte einfacher rekursiver Funktionsaufrufe ausrechnen. Insbesondere soll Ihre Analyse das Programm in `/usr/local/optub/test_files/fac_rec.c` auf der Übungsmaschine korrekt analysieren. Mit einer entsprechenden Call-String-Länge soll die in der Funktion `main` gestartete Rekursion vollständig ausgewertet werden; andern-

falls kann Ihre Analyse keinen konstanten Rückgabewert für `main` vorhersagen.

Die Verbesserung der Analyse gilt natürlich nicht nur für rekursive Programme: Schreiben Sie ein nichtrekursives SL_2-Programm, für welches Ihre Analyse bei Call-String-Längen von 0, 1 und 2 jeweils unterschiedliche (immer genauere) Ergebnisse liefert. Markieren Sie durch einen Kommentar im Programm einen Punkt in der Funktion `main`, an dem sich die drei Ergebnisse unterscheiden.

3 Abgabe

Senden Sie Ihre Lösungen bis **27. Jänner 2016, 16:00 Uhr** per E-Mail an `hans@complang.tuwien.ac.at`. Die Betreffzeile der E-Mail soll wie gewohnt '`OU: Aufgabe 4,` *Nachname(n)*' lauten. Hängen Sie die Analysespezifikation für Teilaufgabe 1 als `.optla`-Datei, das Eingabeprogramm für Teilaufgabe 2 als C-Programm an die E-Mail an.