Sinteza popravki programa na osnovu ispravnih primera

Ivan Ristović, Milana Kovačević, Strahinja Stanojević

septembar 2018.

Zadatak

- Ulaz: Dva isečka koda, jedan koji služi kao specifikacija i drugi sa greškama
- Zadatak: ispraviti kod da postane semantički ekvivalentan ispravom kodu
- Ispravljen kod treba da zadrži originalnu strukturu sto je više moguće

Primeri ulaza

```
int foo1()
{
    int x = 1, a = 2;
    return x + a;
}
```

```
int foo2()
{
    int x = 0, a = 2;
    return x + a;
}
```

Primeri ulaza

- Ekvivalentno?

```
int fooEq1()
{
    int x = 1;
    int a = 2;
    return x + a + 1;
}
```

```
int fooEq2()
{
    return 4;
}
```

Primeri ulaza

- Sporedni efekti

```
int fooSideEff()
{
    print("Hello!");
    return 1;
}
```

```
int foo()
{
    return 1;
}
```

Primeri ulaza

- Moguće evaluirati u nekim slučajevima

```
int anotherFooEq1()
{
    int x = 0;
    x += 3;
    return x;
}
```

```
int anotherFooEq2()
{
    int a = 0;
    a++;
    a++;
    ++a;
    return a;
}
```

Primeri ulaza

```
int ambFoo1(int x)
{
    if (x >= 0)
        return 1;
    else
        return 0;
}
```

```
int ambFoo2(int y)
{
    return y % 2;
}
```

```
void wrapper1()
{
    int x = ambFoo1(1);
}
```

```
void wrapper2()
{
    int x = ambFoo2(1);
}
```

Pretpostavke

- Sve promenljive moraju biti inicijalizovane pre njihovog korišćenja
- Funkcije nemaju sporedne efekte
- Uslovi grananja i petlji moraju biti deterministički

Ograničenja

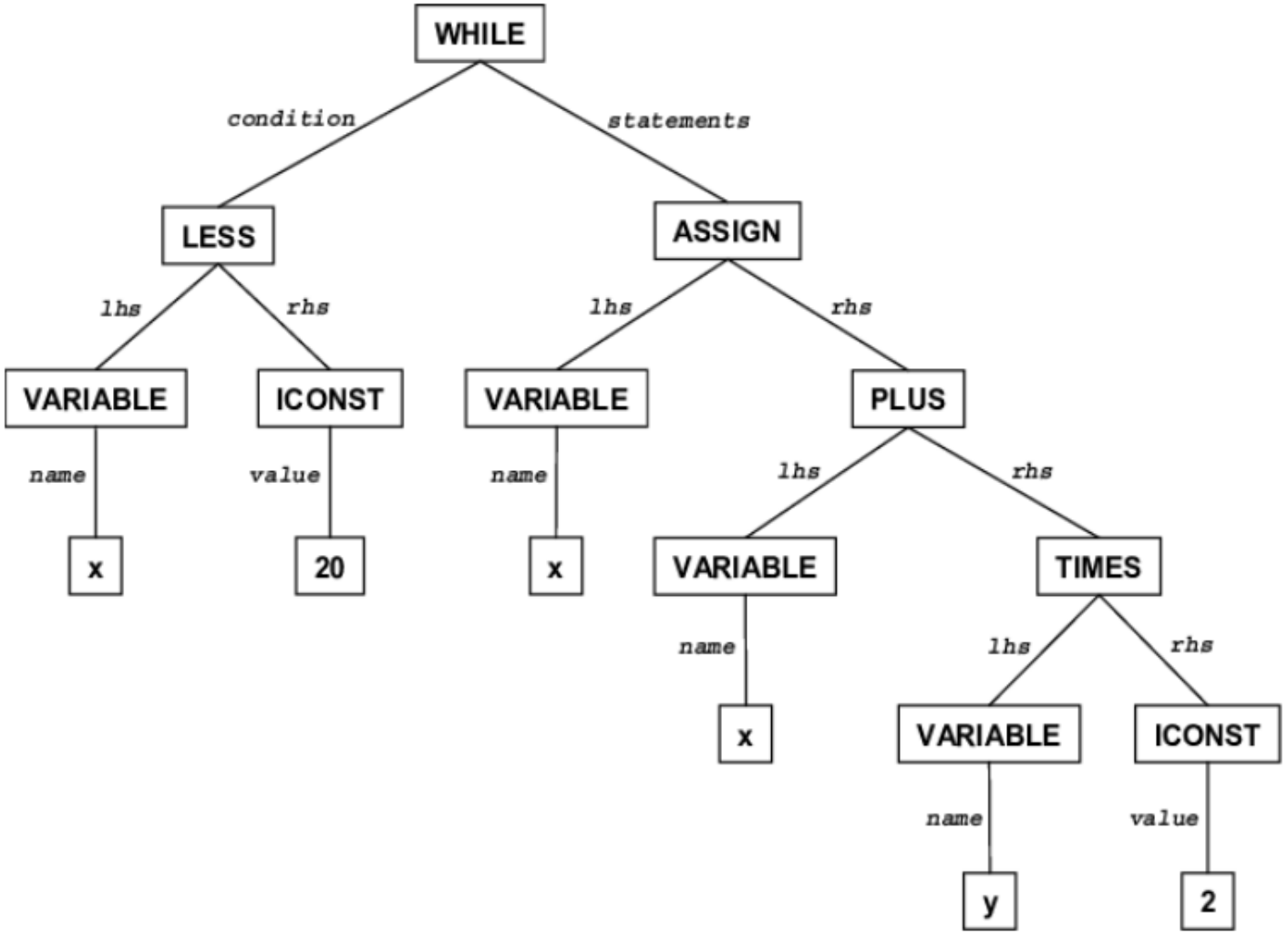
- Trenutno promenljive mogu biti samo int tipa
- Implementirana je analiza jednostavnih konstrukata kao što su:
  - deklaracije (bilo promenljivih ili funkcija)
  - naredbe dodele
  - binarni aritmetički i logički izrazi
  - deterministička grananja
  - povratne vrednosti funkcija

AST

- *Apstraktno sintaksno stablo* je stablo reprezentacije apstraktne sintaksne strukture koda pisanog u nekom programskom jeziku
- Svaki čvor drveta odgovara nekom konstruktu koda

```
while (x < 20)
    x = x + y * 2;
```

AST


WHILE
condition
statements
LESS
ASSIGN
lhs
rhs
lhs
rhs
VARIABLE
ICONST
VARIABLE
PLUS
name
value
name
lhs
rhs
x
20
x
VARIABLE
TIMES
name
lhs
rhs
x
VARIABLE
ICONST
name
value
y
2

Korišćeni alati

- GumTree
- Eclipse JDT Core API

Algoritam poredjenja

- Pretpostavka: Semantički ekvivalentni kodovi imaju iste vrednosti promenljivih na izlazu svakog bloka

```
int x = 1;
int foo()
{
    int a = 5;

    if (a > 3)
        x = 2;

    int c = 1;

    if (a > 3)
        c = 2;
}
```

```
int y = 1;
int foo2()
{
    int a = 2;

    if (a < 3)
        y = 2;

    int c = 1;

    if (a < 3)
        c = 2;
}
```

Algoritam poredjenja

```
// Block depth 0, ordinal 1
int x = 1;

int foo()
{
    // Block depth 1, ordinal 1
    // Vars passed: x = 1

    int a = 5;

    if (a > 3) {
        // Block depth 2, ordinal
            1
        x = 2;
        // End of block: Update x
            in parent
    }

    // x = 2 here because of the
        update

    int c = 1;
    if (a > 3) {
        // Block depth 2, ordinal
            2
        c = 2;
        // End of block: Update c
            in parent
    }

    // Vars: x = 2, a = 5, c = 2
    // End of block: Update x in
        parent
}

// End of root block: x = 2
```

Algoritam poredjenja

Block depth	Block ordinal	Variables
0	1	x
1	1	x, a, c
2	1	x, a
2	2	x, a, c

Algoritam poredjenja

```
// Block depth 0, ordinal 1
int y = 1;  // Matches to x

int foo()
{
    // Block depth 1, ordinal 1
    // Vars passed: y = 1

    int a = 2;

    if (a < 3) {
        // Block depth 2, ordinal
              1
        y = 2;
        // End of block: Update y
              in parent
    }

    // y = 2 here because of the
         update

    int c = 1;
    if (a < 3) {
        // Block depth 2, ordinal
              2
        c = 2;
        // End of block: Update c
              in parent
    }

    // Vars: y = 2, a = 2, c = 2
    // Conflict found: a = 5 in
         source, but found a = 2
    // End of block: Update y in
         parent
}

// End of root block: y = 2
```

Algoritam poredjenja

```
Analyze(tree1: AST, tree2: AST)
begin
    if (CreateMatcher(tree1, tree2).OnlyUpdateActionsFound()):
        /* We have found only rename actions */
        print("Given snippets are equivallent")
        return

    /* In general case, traverse the first tree */
    vmap = tree1.TraverseAndRecordVars()

    /* Traverse the second tree and compare vars per block */
    conflicts = tree2.TraverseAndCompareVars(vmap)

    foreach (conflict in conflicts):
        print(conflict.Details)
end
```

Primeri

:)

Poboljšanja

- Proširivanje tipova na primitivne i referencne
- Simboličke promenljive
- Petlje sa poznatim brojem iteracija; razmotavanje
- Složeni konstrukti jezika - anonimne funkcije npr.

Pitanja

???

Hvala na pažnji!