Modern statikus analízis

Norbert Pataki

Programozási Nyelvek és
Fordítóprogramok Tanszék

Bolyai Kollégium, 2023

Miről lesz szó?

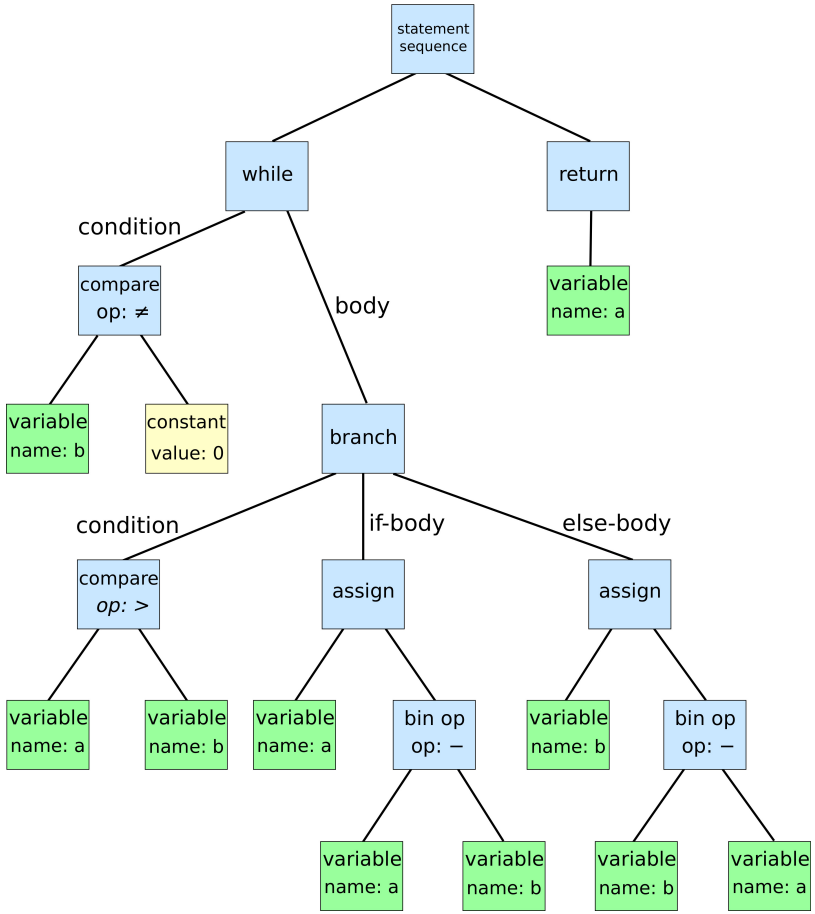
A forráskód feldolgozása

- Fordítóprogramok, interpreterek
- Syntax-highlight
- Linterek, indent
- Klasszikus statikus analízis
- Modern statikus analízis

A forráskód feldolgozása

- Lexikális egységek, lexikális elemző (lexer)
- Szintaktikus elemzés, szintaktikus elemző (parser)
- Szintaxisfa, absztrakt szintaxisfa (AST)

AST, példa


statement sequence
while
return
condition
body
compare op: ≠
variable name: a
variable name: b
constant value: 0
branch
condition
if-body
else-body
compare op: >
assign
assign
variable name: a
variable name: b
variable name: a
bin op op: −
variable name: b
bin op op: −
variable name: a
variable name: b
variable name: b
variable name: a

Fordítóprogramok

- AST felépítése
- Szemantikus ellenőrzések (fordítási figyelmeztetések és hibák)
- Optimalizációk
- Kódgenerálás (pl. Assembly)

Optimalizációk

```
// af.cpp:
void array_fill( int* p )
{
  for( int i = 0; i < 6; ++i )
  {
    p[ i ] = 0;
  }
}
```

Optimalizációk

```
g++ -W -Wall -pedantic -ansi -save-temps -O0 af.cpp
```

```
        .file   "af.cpp"
        .text
        .globl  _Z10array_fillPi
        .type   _Z10array_fillPi, @function
_Z10array_fillPi:
.LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movq    %rdi, -24(%rbp)
        movl    $0, -4(%rbp)
.L3:
        cmpl    $5, -4(%rbp)
        jg      .L4
        movl    -4(%rbp), %eax
        cltq
        leaq    0(,%rax,4), %rdx
        movq    -24(%rbp), %rax
        addq    %rdx, %rax
        movl    $0, (%rax)
        addl    $1, -4(%rbp)
        jmp     .L3
.L4:
        nop
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   _Z10array_fillPi, .-_Z10array_fillPi
        .ident  "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 20160609"
        .section        .note.GNU-stack,"",@progbits
```

Optimalizációk

```
g++ -W -Wall -pedantic -ansi -save-temps -O3 af.cpp
```

```
        .file   "af.cpp"
        .section        .text.unlikely,"ax",@progbits
.LCOLDB0:
        .text
.LHOTB0:
        .p2align 4,,15
        .globl  _Z10array_fillPi
        .type   _Z10array_fillPi, @function
_Z10array_fillPi:
.LFB0:
        .cfi_startproc
        movq    $0, (%rdi)
        movq    $0, 8(%rdi)
        movq    $0, 16(%rdi)
        ret
        .cfi_endproc
.LFE0:
        .size   _Z10array_fillPi, .-_Z10array_fillPi
        .section        .text.unlikely
.LCOLDE0:
        .text
.LHOTE0:
        .ident  "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 20160609"
        .section        .note.GNU-stack,"",@progbits
```

Klasszikus statikus elemzési feladatok

- Hibakeresés, több ellenőrzés, figyelmeztetés
  - False positive-ok csökkentése, pontosság javítása
- Refaktorálás
- Forráskód bonyolultság mérése
- Kódmegértés támogatása

Clang példa

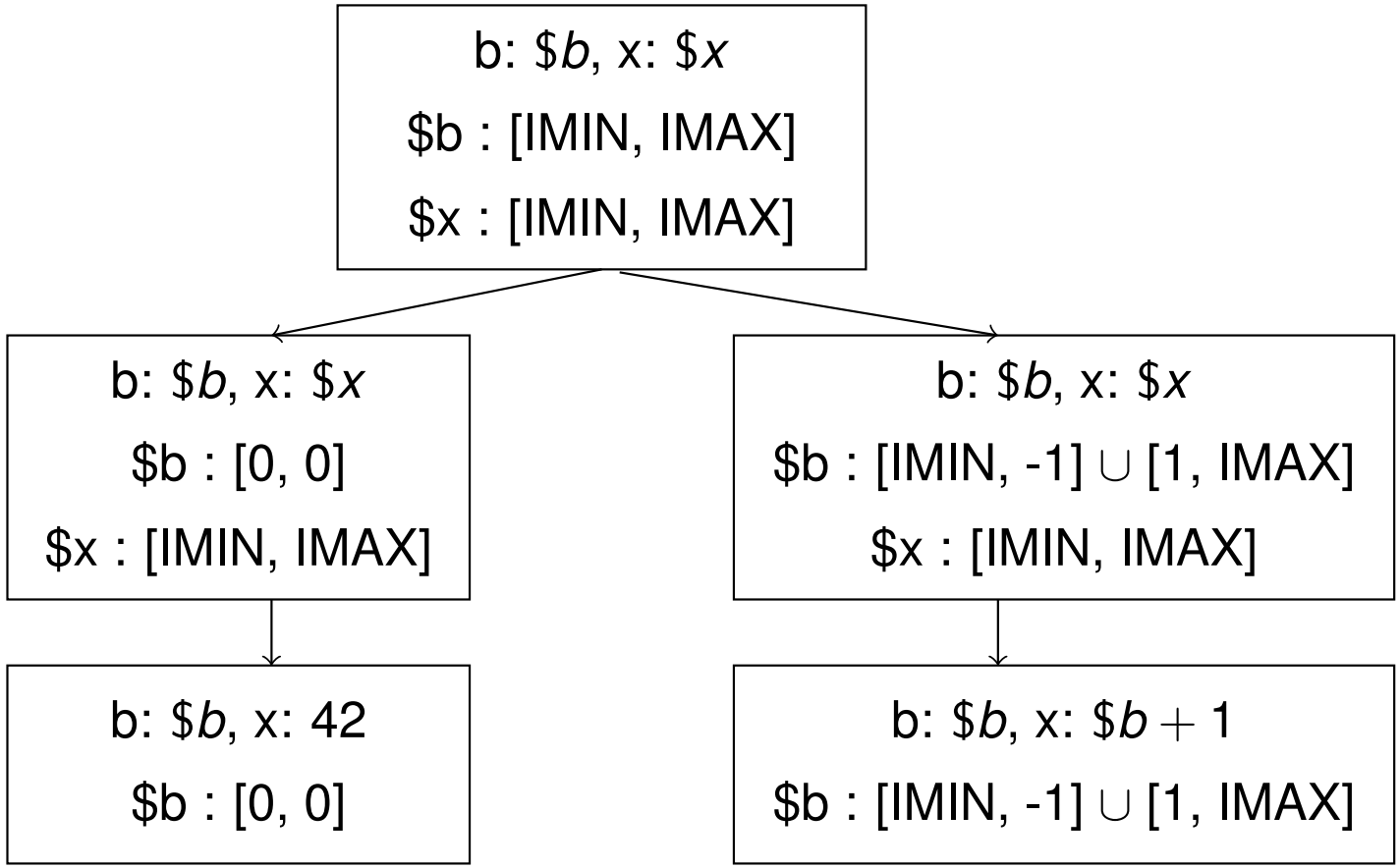
```
memberCallExpr(
  on( hasType( container ) ),
  callee( methodDecl( hasName( "size" ) ) ),
  unless( anyOf(
      hasAncestor( binaryOperator(
        unless( has( integerLiteral(
          equals( 0 ) ) ) ),
        unless( anyOf(
          hasOperatorName( "&&" ),
          hasOperatorName( "||" ) ) ) ) ),
      hasAncestor( varDecl() ) ) ) ).bind( "id" );
```

Modern megközelítések

- Szimbolikus végrehajtás
- Forráskód megfiatalítás (source code rejuvenation)
- Modern kódmegértéssel kapcsolatos feladatok

Szimbolikus végrehajtás

```
void g(int b,
       int &x)
{
  if (b)
    x = b+1;
  else
    x = 42;
}
```



Rejuvenation

- Nyelvi változások
- Backward kompatibilitás?
- Újabb, jobb nyelvi eszközök
- Áttérés (automatikusan)
- Nem refaktorálás
- Példa: `virtual`, `override`
- Elavulttá váló konstrukciók (`std::iterator`)

Kérdés

```
void f();
```

- Dobhat-e `f` kivételt?
- Biztos-e, hogy `f` kivétel kiváltása nélkül lefuthat? Tudjuk-e, hogy `f` soha nem dob kivételt?
- Csak a deklarációk elég információt adnak-e?

C

- C: nincsenek kivétel-kezeléssel kapcsolatos nyelvi elemek, workaround-ok
- C: nincs kivétel-specifikáció
- Könnyen hihetjük, hogy egy C függvény sosem dob kivételt

C

```
// f.c:
void g( void );

void f()
{
  g();
}
```

```
// g.cpp:
extern "C"
{
  void g()
  {
    throw "Leave me alone";
  }
}
```

Klasszikus C++

```
void f()
{
  throw "Leave me alone";
}

void g() throw(double)
{
  throw "Leave me alone";
}
```

Mindkettő lefordul.

Klasszikus C++

```
int id( int i )
{
  return i;
}

int id_excspec( int i ) throw(std::exception)
{
  return i;
}
```

Mindkettő lefordul.

Modern C++

```
void f() noexcept
{
  throw "Leave me alone";
}

int id( int i )
{
  return i;
}
```

Mindkettő lefordul.

This is fine

Összefoglalás

- Forráskóddal kapcsolatos feladatok és eszközök
- Számos feladat
- Új jellegű megközelítések

Köszönöm a figyelmet!