FAKULTA
INFORMATIKY
Masarykova univerzita

Angelic Verification

Precise Verification Modulo Unknowns

Jan Tušil

22. března 2018

Příklad

```
var m:[int]int;
```

Příklad

```
var m:[int]int;

// entry point
procedure Foo(z:int) {
  call Baz(NULL);
}
```

Příklad

```
var m:[int]int;

// entry point
procedure Foo(z:int) {
  call Baz(NULL);
}

procedure Baz(y:int) {
  assert y != NULL;
  m[y] := 4;
}
```

Příklad

```
var m:[int]int;

// entry point
procedure Foo(z:int) {
  call Baz(NULL);
}

procedure Baz(y:int) {
  assert y != NULL; // 100% bug
  m[y] := 4;
}
```

Jiný příklad

```
var gs:int, m:[int]int;
```

Jiný příklad

```
var gs:int, m:[int]int;

// entry point
procedure Foo(z:int) {
  call Bar(z);
}
```

Jiný příklad

```
var gs:int, m:[int]int;

// entry point
procedure Foo(z:int) {
  call Bar(z);
}

procedure Bar(x:int) {
  if (x != NULL) { gs := 1; }
  else { gs := 2; }
  assert x != NULL;
  m[x] := 5;
}
```

Jiný příklad

```
var gs:int, m:[int]int;

// entry point
procedure Foo(z:int) {
  call Bar(z);
}

procedure Bar(x:int) {
  if (x != NULL) { gs := 1; }
  else { gs := 2; }
  assert x != NULL; // bug
  m[x] := 5;
}
```

Jiný příklad

```
var gs:int, m:[int]int;
// precondition: z != NULL
// entry point
procedure Foo(z:int) {
  call Bar(z);
}

procedure Bar(x:int) {
  if (x != NULL) { gs := 1; }
  else { gs := 2; }
  assert x != NULL; // bug
  m[x] := 5;
}
```

Jiný příklad

```
var gs:int, m:[int]int;
// precondition: z != NULL
// entry point
procedure Foo(z:int) {
  call Bar(z);
}

procedure Bar(x:int) {
  if (x != NULL) { gs := 1; }
  else { gs := 2; }
  assert x != NULL; // ok due to precondition
  m[x] := 5;
}
```

Jiný příklad

```
var gs:int, m:[int]int;
// precondition: z != NULL
// entry point
procedure Foo(z:int) {
  call Bar(z);
}

procedure Bar(x:int) {
  if (x != NULL) { gs := 1; // unreachable }
  else { gs := 2; }
  assert x != NULL; // ok due to precondition
  m[x] := 5;
}
```

Jiný příklad

```
var gs:int, m:[int]int;
// precondition: z != NULL
// entry point
procedure Foo(z:int) {
  call Bar(z);
}
// inconsistent
procedure Bar(x:int) {
  if (x != NULL) { gs := 1; // unreachable }
  else { gs := 2; }
  assert x != NULL; // ok due to precondition
  m[x] := 5;
}
```

Do třetice ...

```
var m:[int]int;
```

Do třetice ...

```
var m:[int]int;

// library
procedure Lib1() {
  returns (r:int);

procedure Lib2()
  returns (r:int);
```

Do třetice ...

```
var m:[int]int;

// library
procedure Lib1() {
  returns (r:int);

procedure Lib2()
  returns (r:int);


// entry point
procedure Foo(z:int) {
  call FooBar();
}
```

Do třetice ...

```
var m:[int]int;

// library
procedure Lib1() {
  returns (r:int);

procedure Lib2()
  returns (r:int);


// entry point
procedure Foo(z:int) {
  call FooBar();
}
```

```
procedure FooBar() {
  var x, w, z:int

}
```

Do třetice ...

```
var m:[int]int;

// library
procedure Lib1() {
  returns (r:int);

procedure Lib2()
  returns (r:int);


// entry point
procedure Foo(z:int) {
  call FooBar();
}
```

```
procedure FooBar() {
  var x, w, z:int
  call z := Lib1();


}
```

Do třetice ...

```
var m:[int]int;

// library
procedure Lib1() {
  returns (r:int);

procedure Lib2()
  returns (r:int);


// entry point
procedure Foo(z:int) {
  call FooBar();
}
```

```
procedure FooBar() {
  var x, w, z:int
  call z := Lib1();
  assert z != NULL;



}
```

Do třetice ...

```
var m:[int]int;

// library
procedure Lib1() {
  returns (r:int)
  ensures (r != NULL);
procedure Lib2()
  returns (r:int);


// entry point
procedure Foo(z:int) {
  call FooBar();
}
```

```
procedure FooBar() {
  var x, w, z:int
  call z := Lib1();
  assert z != NULL;



}
```

Do třetice ...

```
var m:[int]int;

// library
procedure Lib1() {
  returns (r:int)
  ensures (r != NULL);
procedure Lib2()
  returns (r:int);


// entry point
procedure Foo(z:int) {
  call FooBar();
}
```

```
procedure FooBar() {
  var x, w, z:int
  call z := Lib1();
  assert z != NULL;
  m[z] := NULL;


}
```

Do třetice ...

```
var m:[int]int;

// library
procedure Lib1() {
  returns (r:int)
  ensures (r != NULL);
procedure Lib2()
  returns (r:int);


// entry point
procedure Foo(z:int) {
  call FooBar();
}
```

```
procedure FooBar() {
  var x, w, z:int
  call z := Lib1();
  assert z != NULL;
  m[z] := NULL;
  call x := Lib2();



}
```

Do třetice ...

```
var m:[int]int;

// library
procedure Lib1() {
  returns (r:int)
  ensures (r != NULL);
procedure Lib2()
  returns (r:int);


// entry point
procedure Foo(z:int) {
  call FooBar();
}
```

```
procedure FooBar() {
  var x, w, z:int
  call z := Lib1();
  assert z != NULL;
  m[z] := NULL;
  call x := Lib2();
  assert x != NULL;



}
```

Do třetice ...

```
var m:[int]int;

// library
procedure Lib1() {
  returns (r:int)
  ensures (r != NULL);
procedure Lib2()
  returns (r:int)
  ensures (r != NULL);

// entry point
procedure Foo(z:int) {
  call FooBar();
}
```

```
procedure FooBar() {
  var x, w, z:int
  call z := Lib1();
  assert z != NULL;
  m[z] := NULL;
  call x := Lib2();
  assert x != NULL;



}
```

Do třetice ...

```
var m:[int]int;

// library
procedure Lib1() {
  returns (r:int)
  ensures (r != NULL);
procedure Lib2()
  returns (r:int)
  ensures (r != NULL);

// entry point
procedure Foo(z:int) {
  call FooBar();
}
```

```
procedure FooBar() {
  var x, w, z:int
  call z := Lib1();
  assert z != NULL;
  m[z] := NULL;
  call x := Lib2();
  assert x != NULL;
  w := m[x];


}
```

Do třetice ...

```
var m:[int]int;

// library
procedure Lib1() {
  returns (r:int)
  ensures (r != NULL);
procedure Lib2()
  returns (r:int)
  ensures (r != NULL);

// entry point
procedure Foo(z:int) {
  call FooBar();
}
```

```
procedure FooBar() {
  var x, w, z:int
  call z := Lib1();
  assert z != NULL;
  m[z] := NULL;
  call x := Lib2();
  assert x != NULL;
  w := m[x];
  assert w != NULL;

}
```

Do třetice ...

```
var m:[int]int;

// library
procedure Lib1() {
  returns (r:int)
  ensures (r != NULL);
procedure Lib2()
  returns (r:int)
  ensures (r != NULL);

// entry point
procedure Foo(z:int) {
  call FooBar();
}
```

```
allways Lib1() != Lib2();

procedure FooBar() {
  var x, w, z:int
  call z := Lib1();
  assert z != NULL;
  m[z] := NULL;
  call x := Lib2();
  assert x != NULL;
  w := m[x];
  assert w != NULL;

}
```

Do třetice ...

```
var m:[int]int;

// library
procedure Lib1() {
  returns (r:int)
  ensures (r != NULL);
procedure Lib2()
  returns (r:int)
  ensures (r != NULL);

// entry point
procedure Foo(z:int) {
  call FooBar();
}
```

```
allways Lib1() != Lib2();

procedure FooBar() {
  var x, w, z:int
  call z := Lib1();
  assert z != NULL;
  m[z] := NULL;
  call x := Lib2();
  assert x != NULL;
  w := m[x];
  assert w != NULL;
  m[w] := 4;
}
```

Shrnutí