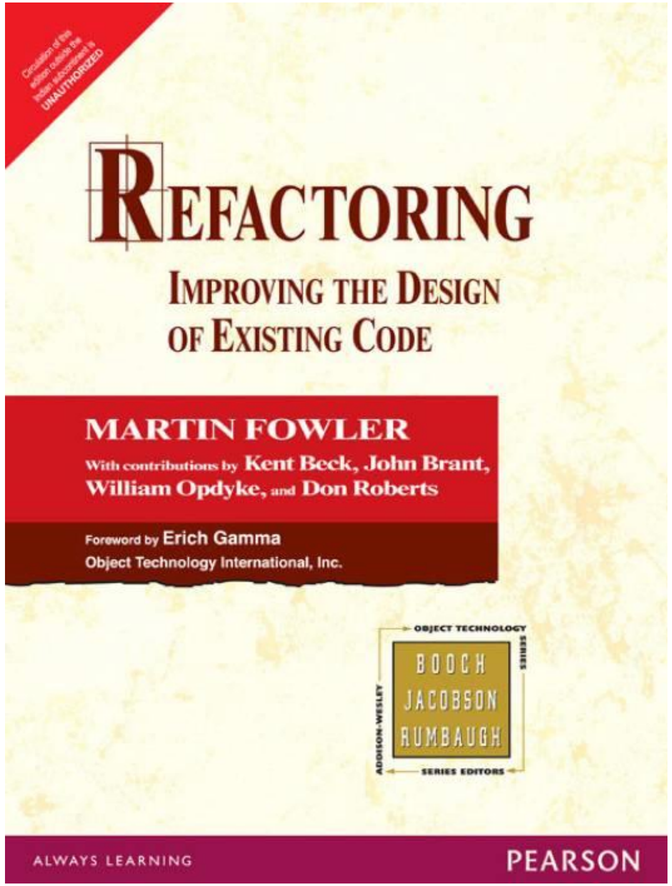
Implementing Refactorings in Spoofax

Phil Misteli - 4932129

IN4333 “Language Engineering Project”

28.06.2019



Circulation of this edition outside the Indian subcontinent is UNAUTHORIZED

REFACTORING

IMPROVING THE DESIGN OF EXISTING CODE

MARTIN FOWLER

With contributions by Kent Beck, John Brant, William Opdyke, and Don Roberts

Foreword by Erich Gamma
Object Technology International, Inc.

OBJECT TECHNOLOGY SERIES

BOOCH JACOBSON RUMBAUGH

ADDISON-WESLEY SERIES EDITORS

ALWAYS LEARNING PEARSON

Refactoring Definition

Structural Change

Unchanged observable behaviour

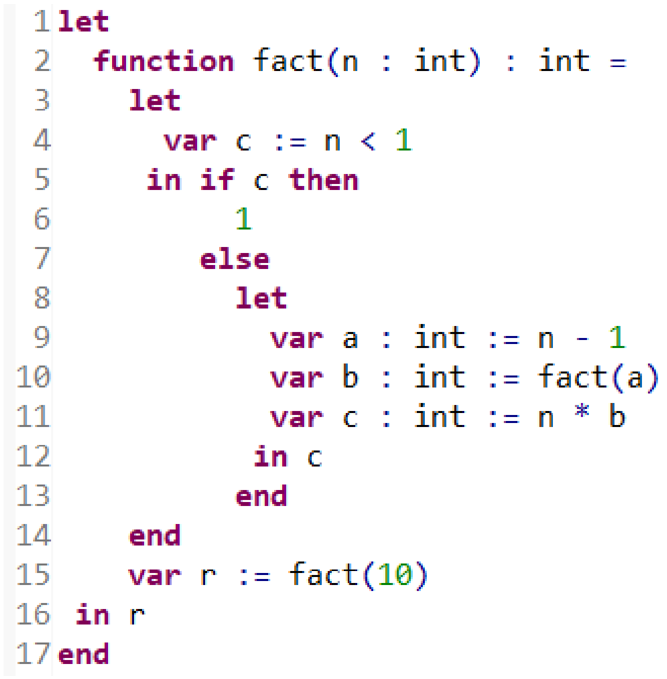
Refactoring Goal

Improve Code Quality

Lower Maintenance Costs

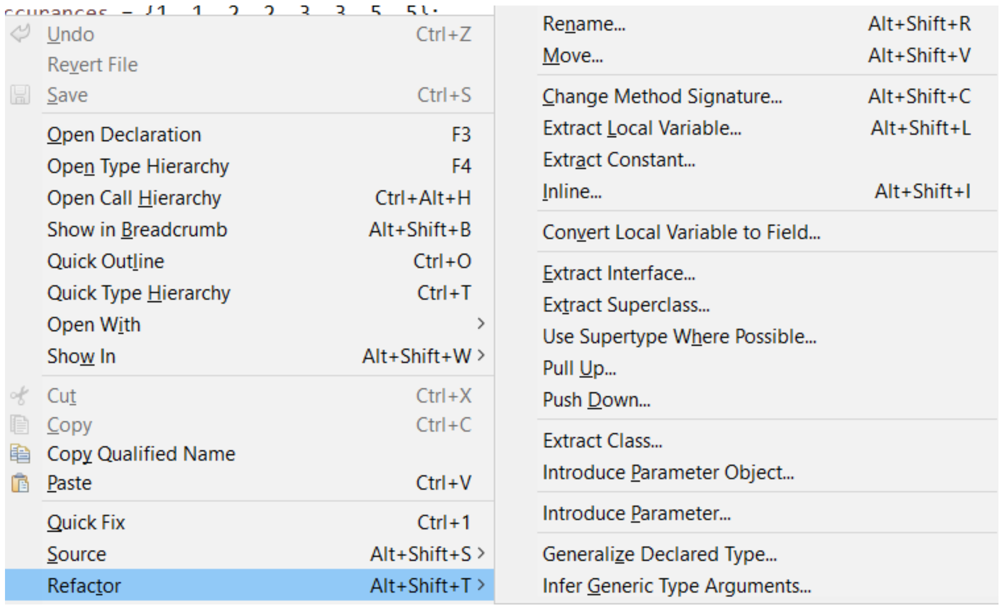
TUDelft

Tiger Programming Language



```
let
 function fact(n : int) : int =
  let
   var c := n < 1
   in if c then
     1
    else
     let
      var a : int := n - 1
      var b : int := fact(a)
      var c : int := n * b
      in c
     end
  end
  var r := fact(10)
 in r
end
```

Refactoring as an Editor Service

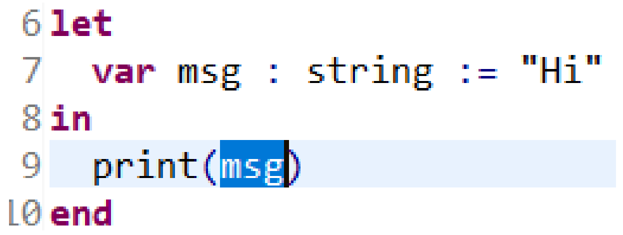


```
ccurances = {1, 1, 2, 2, 3, 3, 5, 5};
```

Implementation in Spoofax

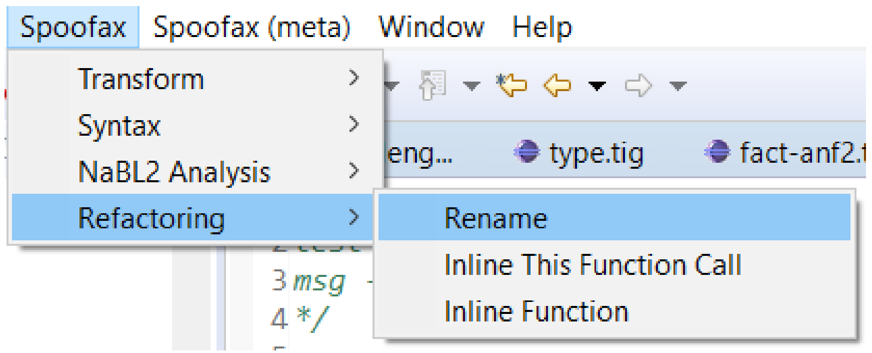
Program Database	→	Scope Graph
Parse Tree	→	AST
Accuracy	→	SPT
Transformation	→	Stratego

User Interface



```
let
  var msg : string := "Hi"
in
  print(msg)
end
```

.rfac
Config
File



Renaming

Named Language Constructs

Variables

Functions

Function Arguments

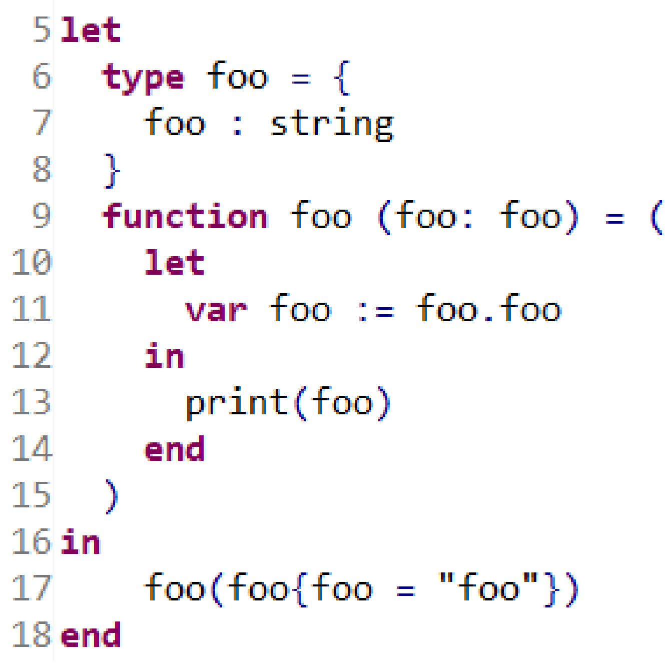
Types

Fields

TU Delft

Renaming

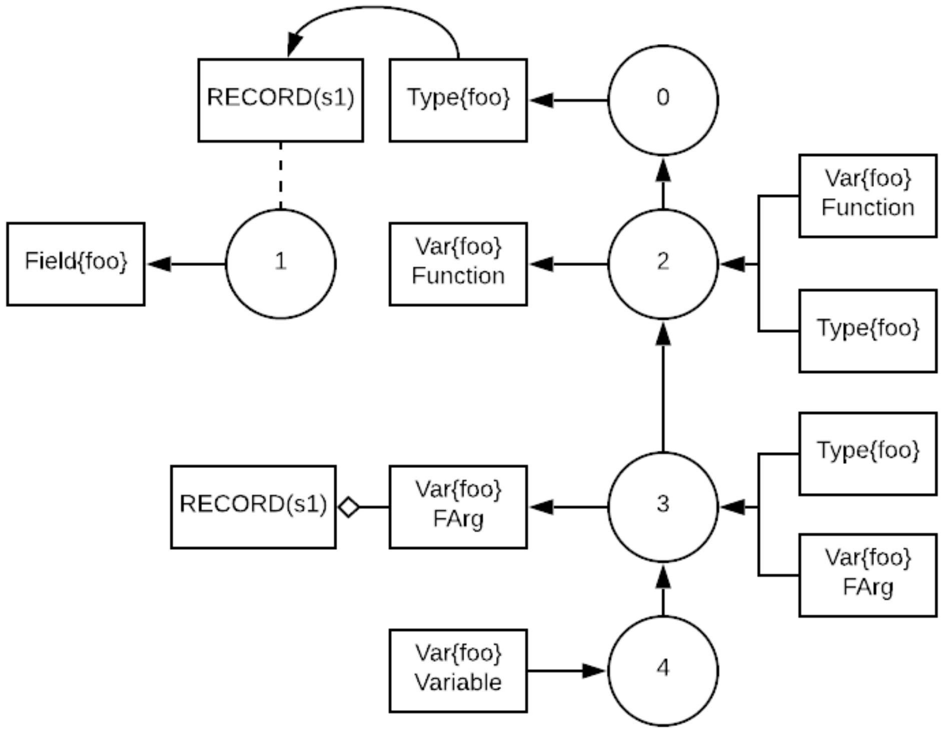
The Foo Challenge



```
let
  type foo = {
    foo : string
  }
  function foo (foo: foo) = (
    let
      var foo := foo.foo
    in
      print(foo)
    end
  )
in
    foo(foo{foo = "foo"})
end
```

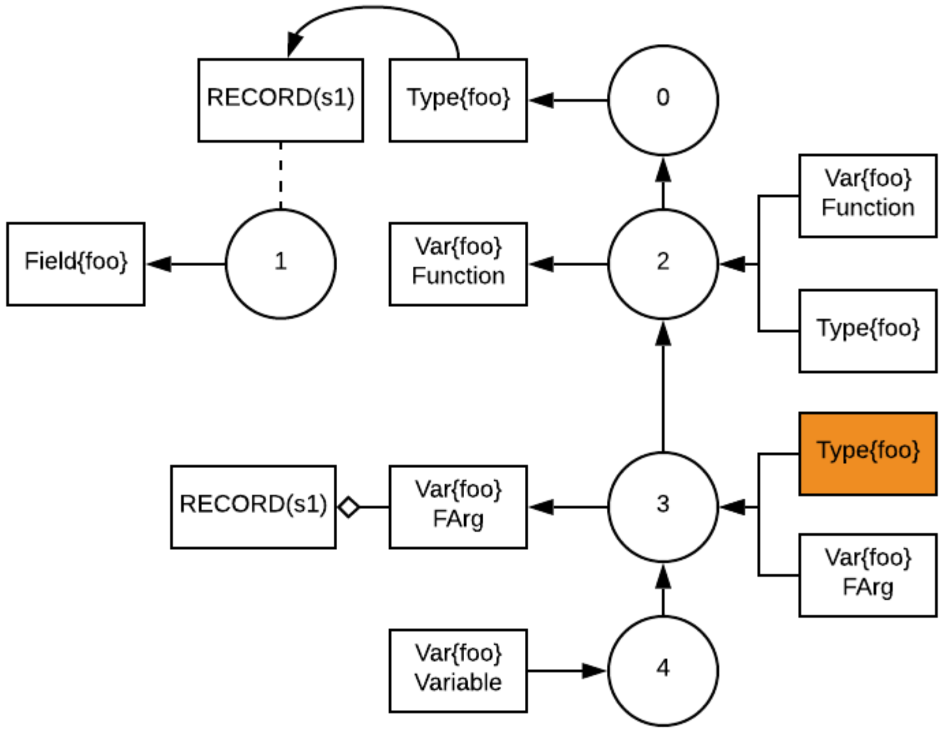
Renaming

The Foo Challenge



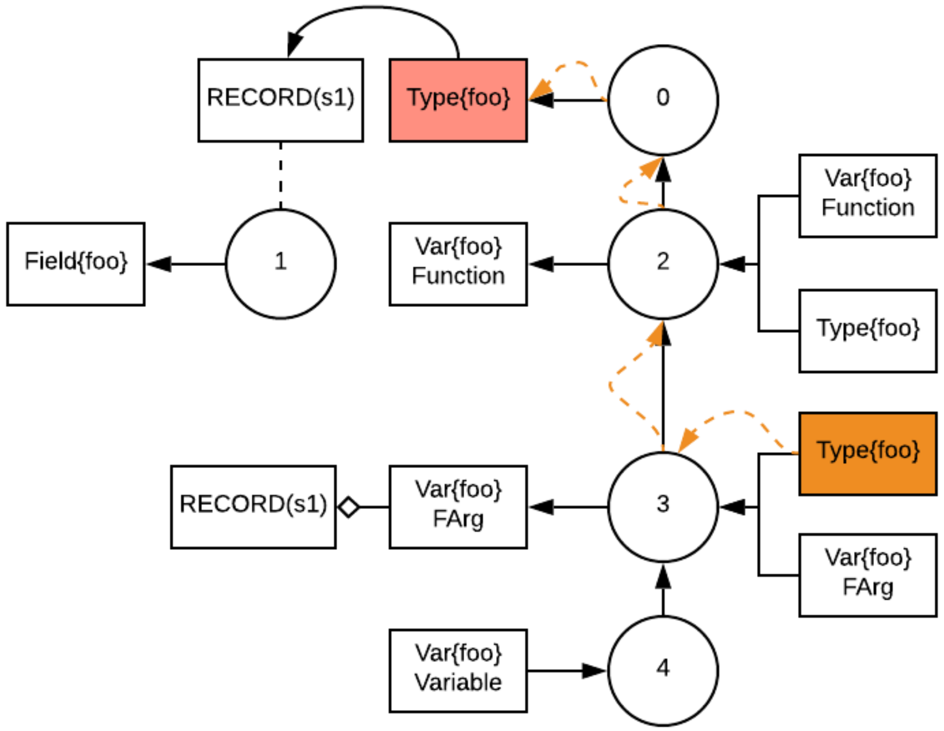
Renaming

Step 1: Select Occurrence



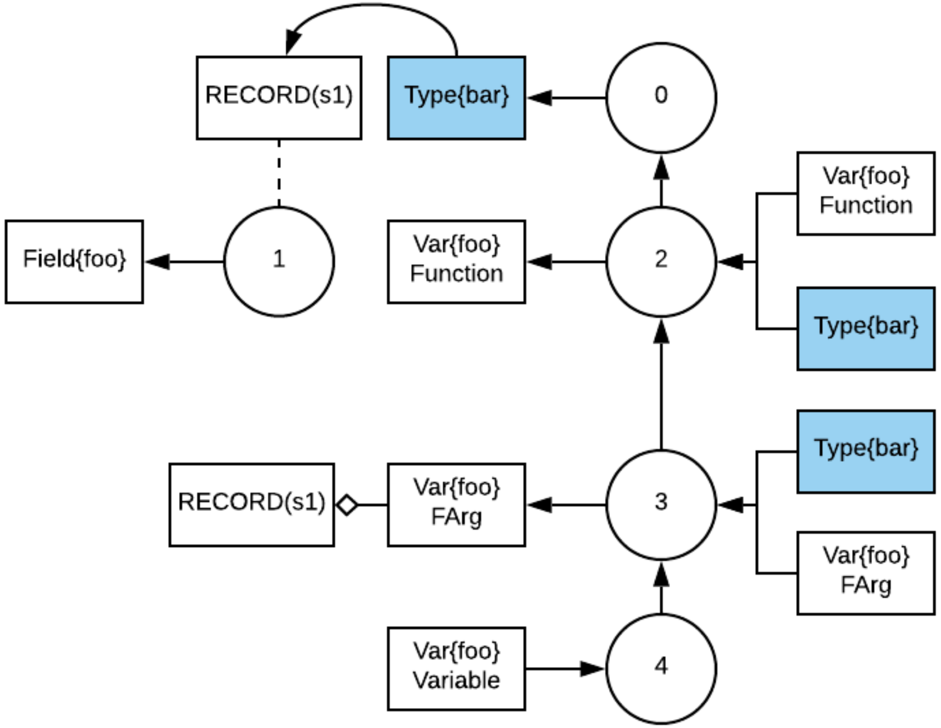
Renaming

Step 2: Find Declaration



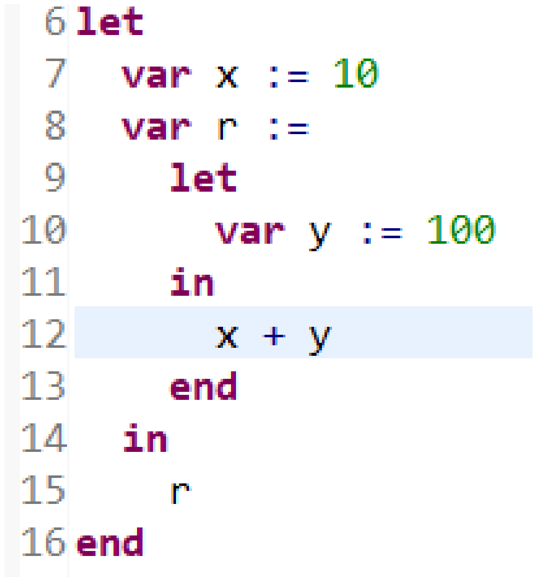
Renaming

Step 3: Rename Occurrences

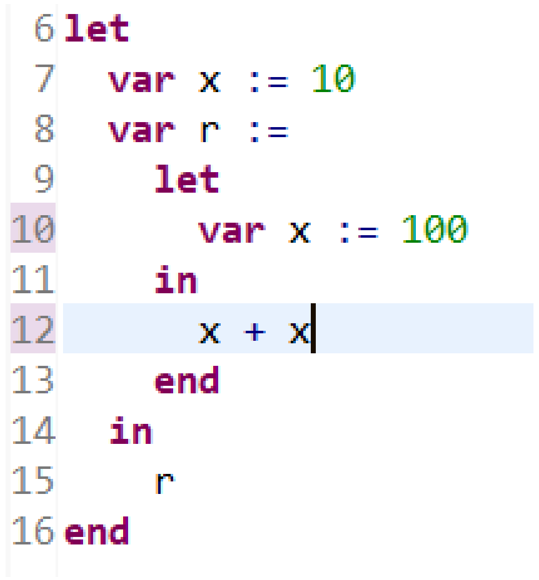


Renaming

Problem: Capture



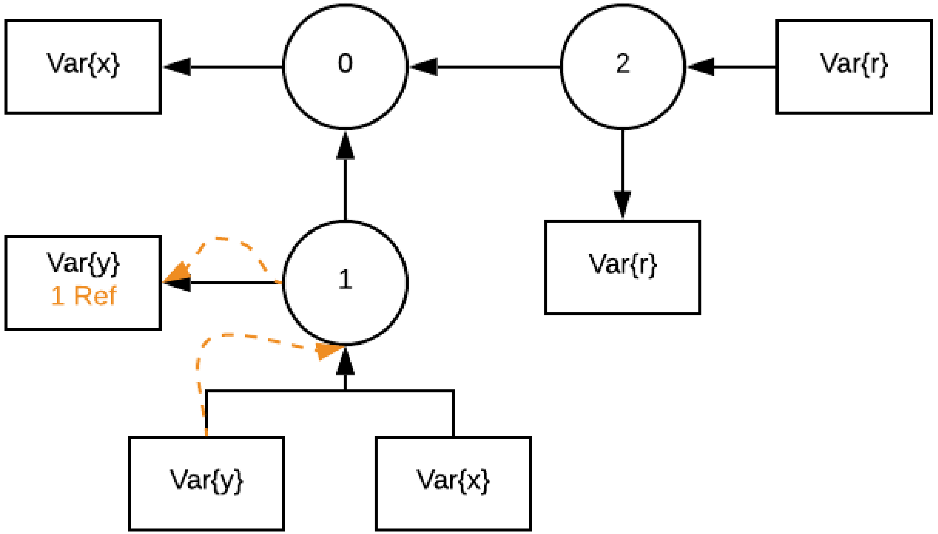
```
let
  var x := 10
  var r :=
    let
      var y := 100
    in
      x + y
    end
  in
    r
end
```



```
let
  var x := 10
  var r :=
    let
      var x := 100
    in
      x + x
    end
  in
    r
end
```

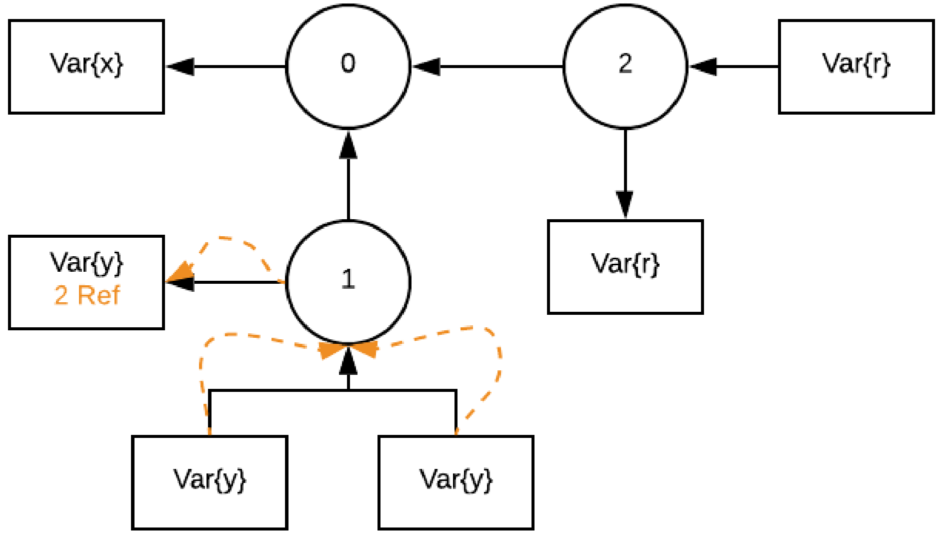
Renaming

Solution: Capture



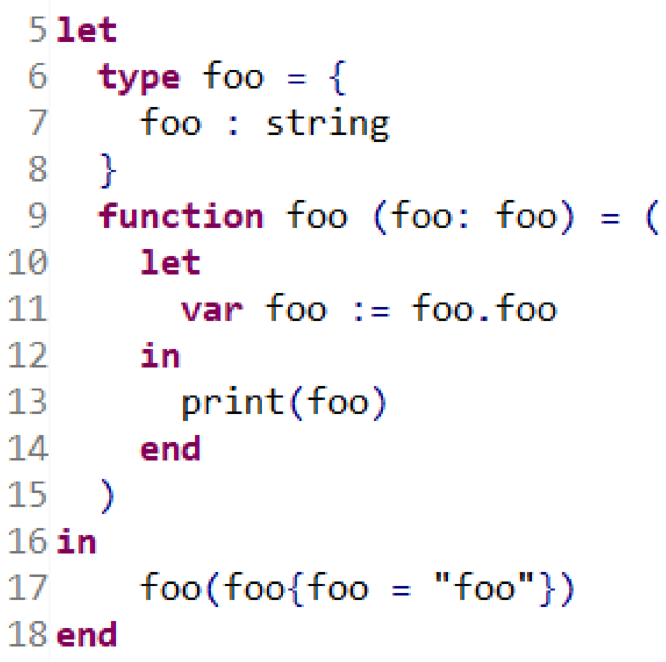
Renaming

Solution: Capture



Renaming

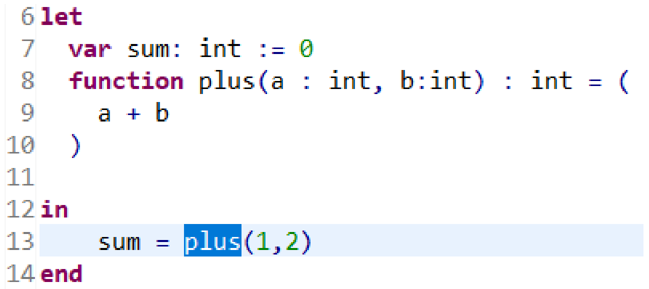
DEMO



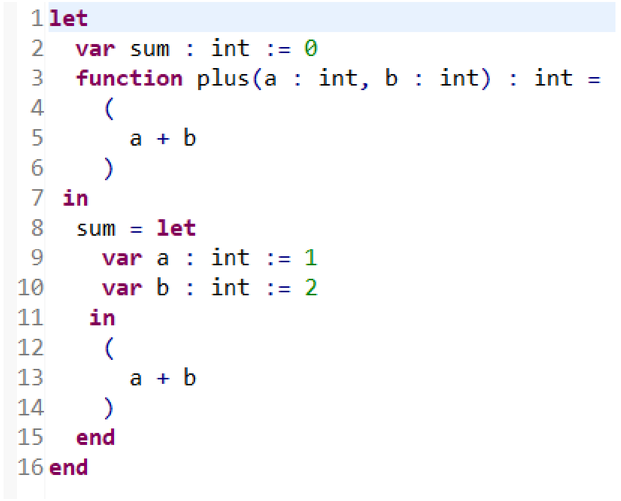
```
let
  type foo = {
    foo : string
  }
  function foo (foo: foo) = (
    let
      var foo := foo.foo
    in
      print(foo)
    end
  )
in
    foo(foo{foo = "foo"})
end
```

Inline Function

Inline Specific Call



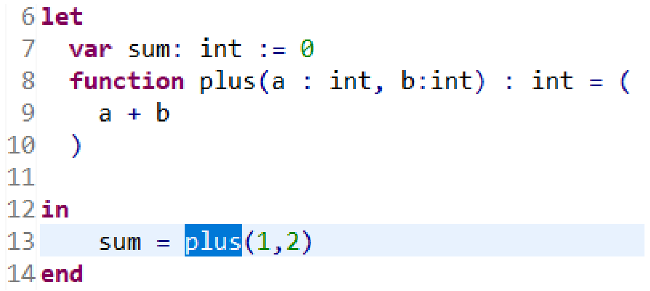
```
let
  var sum: int := 0
  function plus(a : int, b:int) : int = (
    a + b
  )

in
    sum = plus(1,2)
end
```



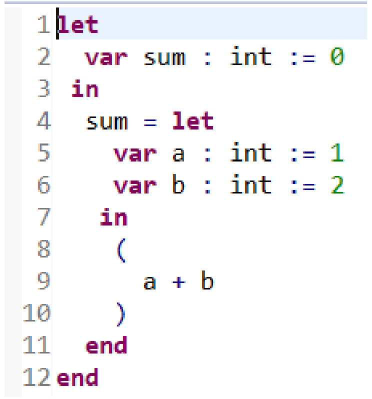
```
let
  var sum : int := 0
  function plus(a : int, b : int) : int =
    (
      a + b
    )
 in
  sum = let
    var a : int := 1
    var b : int := 2
   in
    (
      a + b
    )
  end
end
```

Inline Function

Inline Call and Delete Declaration



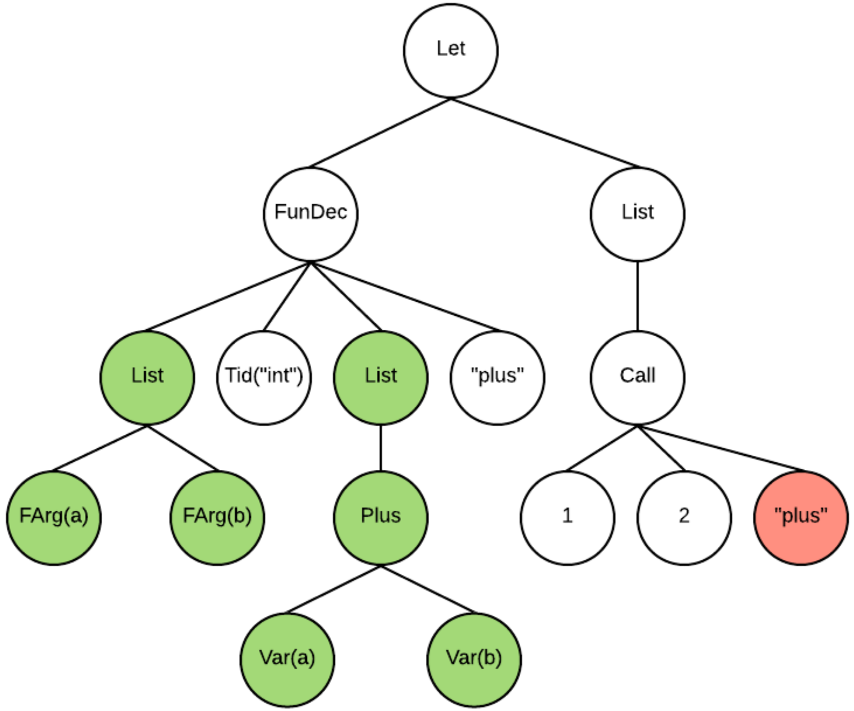
```
let
  var sum: int := 0
  function plus(a : int, b:int) : int = (
    a + b
  )

in
    sum = plus(1,2)
end
```



```
let
  var sum : int := 0
 in
  sum = let
    var a : int := 1
    var b : int := 2
   in
    (
      a + b
    )
  end
end
```

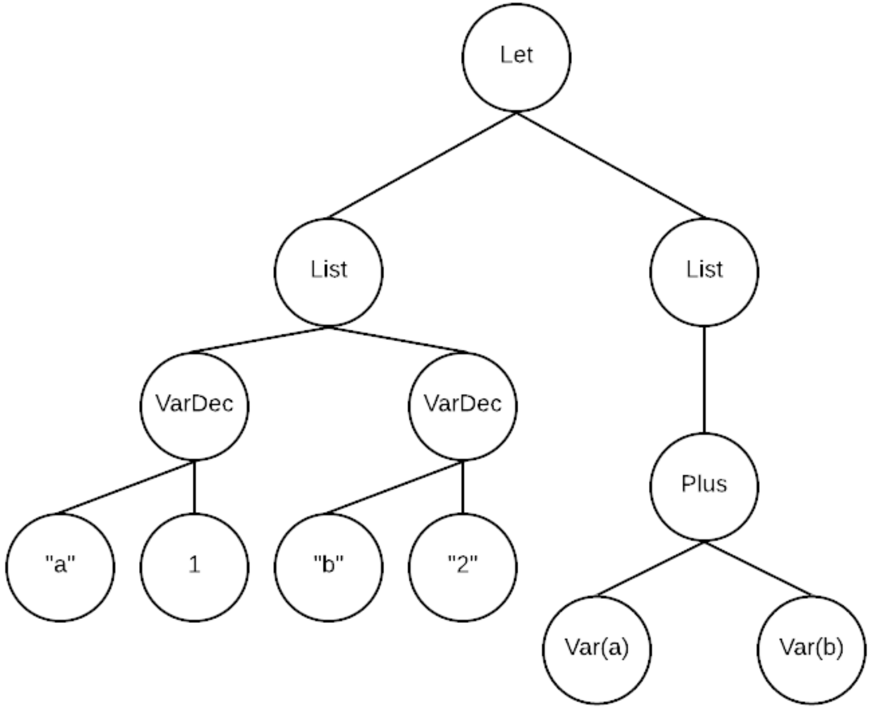
Inline Function

Step 1: Find Function Declaration



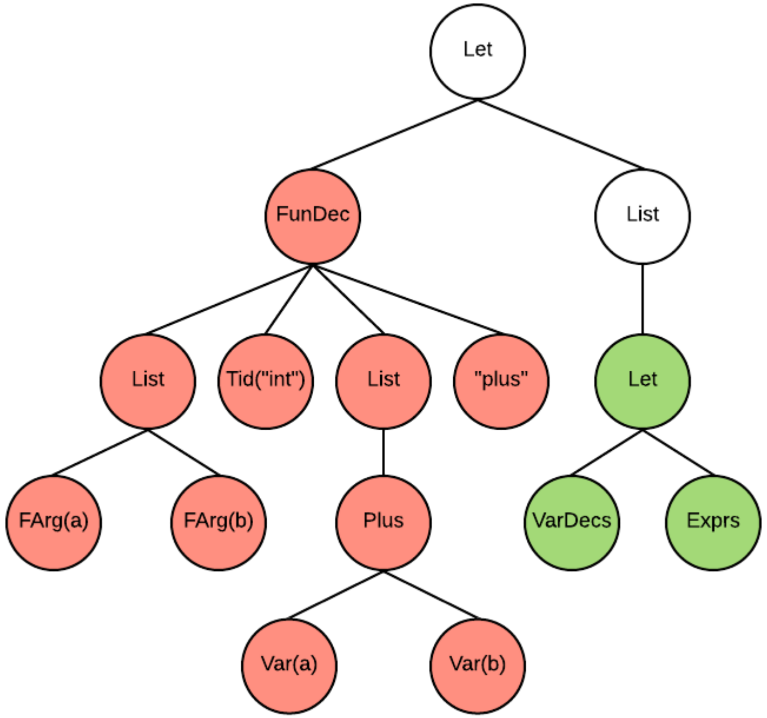
Inline Function

Step 2: Replace Call



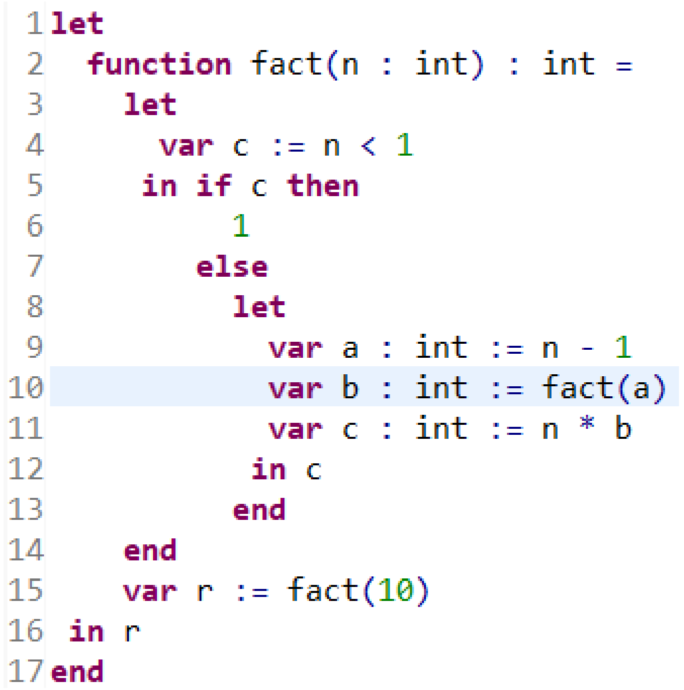
Inline Function

Step 3: Delete Declaration



Inline Function

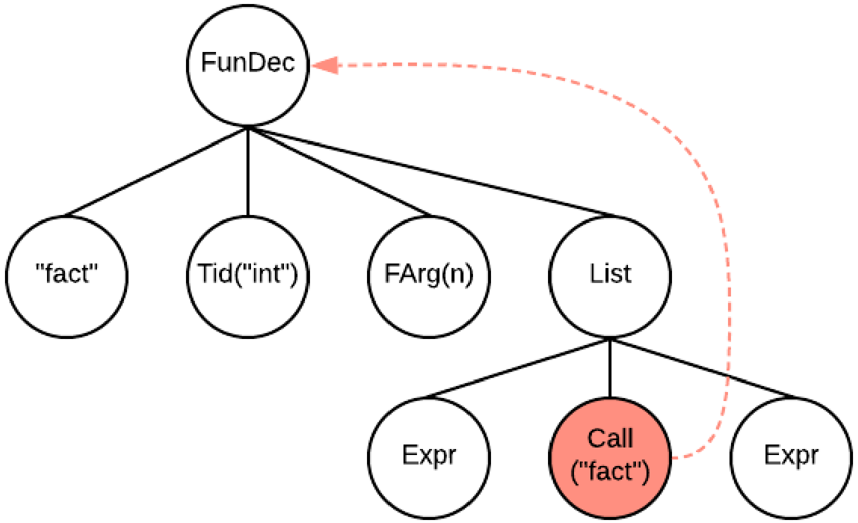
Problem: Recursive Calls



```
let
  function fact(n : int) : int =
    let
      var c := n < 1
     in if c then
          1
        else
          let
            var a : int := n - 1
            var b : int := fact(a)
            var c : int := n * b
           in c
          end
    end
    var r := fact(10)
 in r
end
```

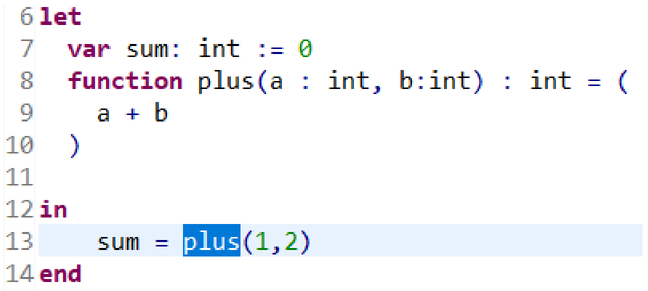
Inline Function

Solution: Recursive Calls

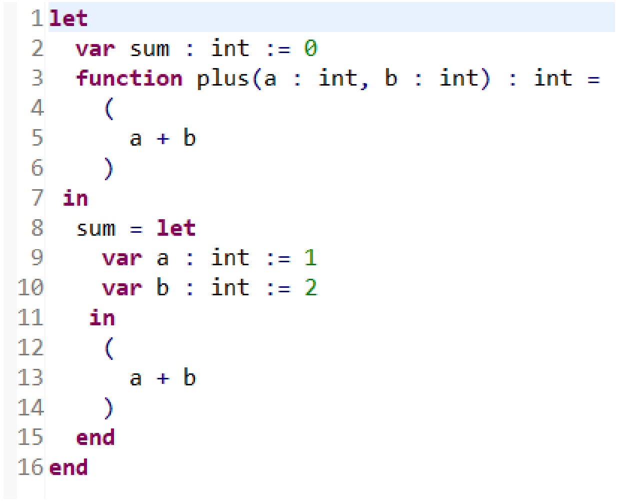


Inline Function

DEMO

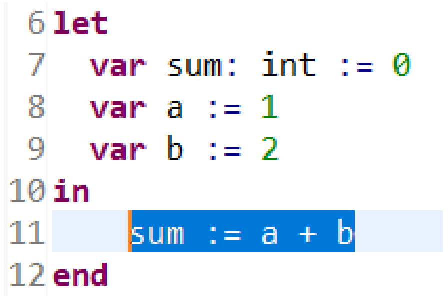


```
let
  var sum: int := 0
  function plus(a : int, b:int) : int = (
    a + b
  )

in
    sum = plus(1,2)
end
```

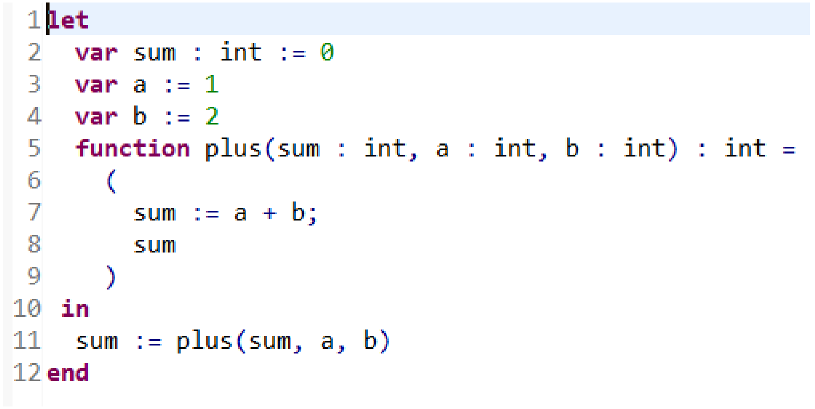


```
let
  var sum : int := 0
  function plus(a : int, b : int) : int =
    (
      a + b
    )
 in
  sum = let
    var a : int := 1
    var b : int := 2
   in
    (
      a + b
    )
  end
end
```

Extract Function



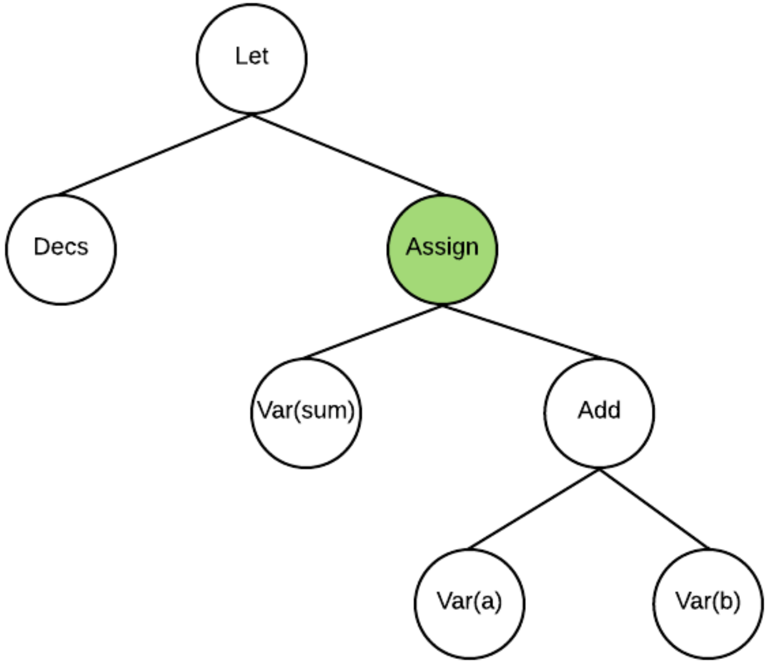
```
let
  var sum: int := 0
  var a := 1
  var b := 2
in
      sum := a + b
end
```



```
let
  var sum : int := 0
  var a := 1
  var b := 2
  function plus(sum : int, a : int, b : int) : int =
    (
      sum := a + b;
      sum
    )
 in
  sum := plus(sum, a, b)
end
```

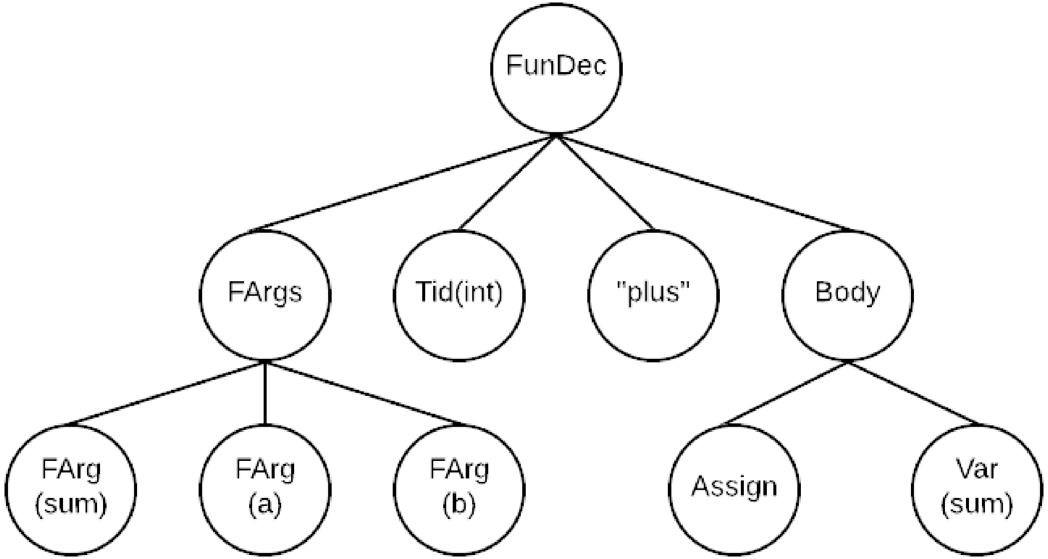
Extract Function

Step 1: Check Selected Term



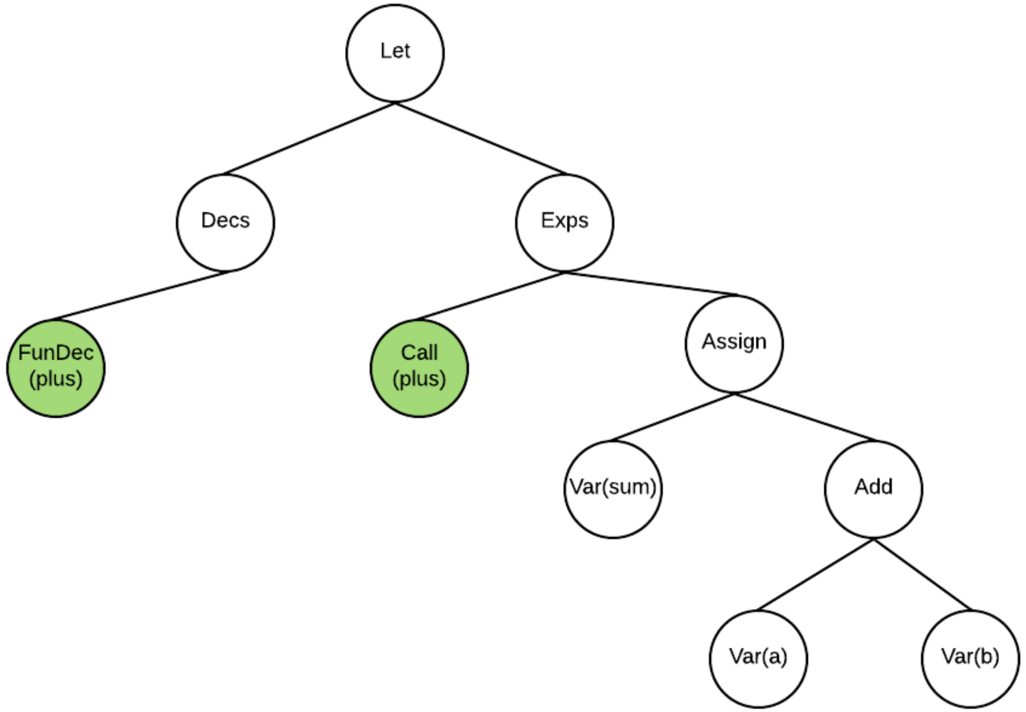
Extract Function

Step 2: Create Function Definition



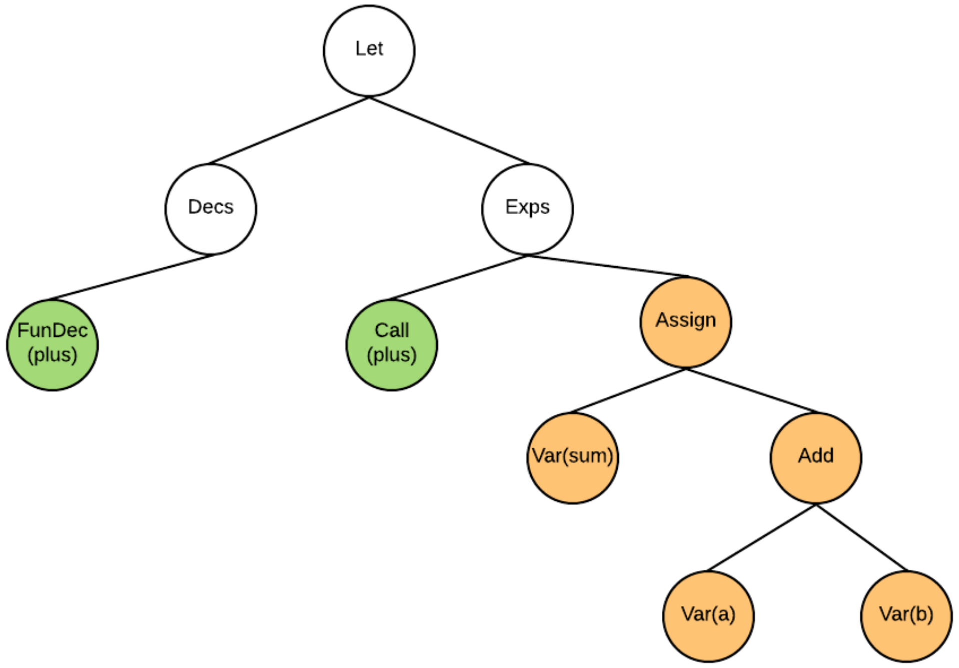
Extract Function

Step 3: Insert Call



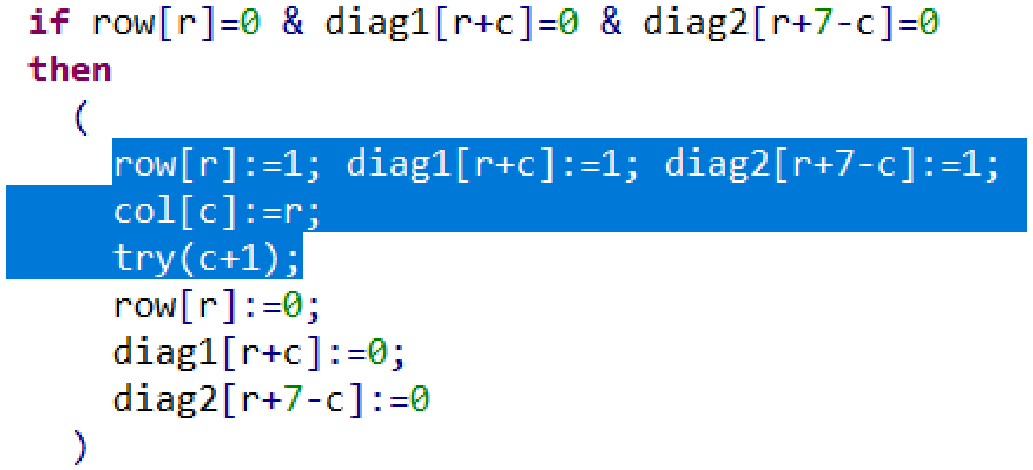
Extract Function

Step 4: Delete Extracted Expression



Extract Function

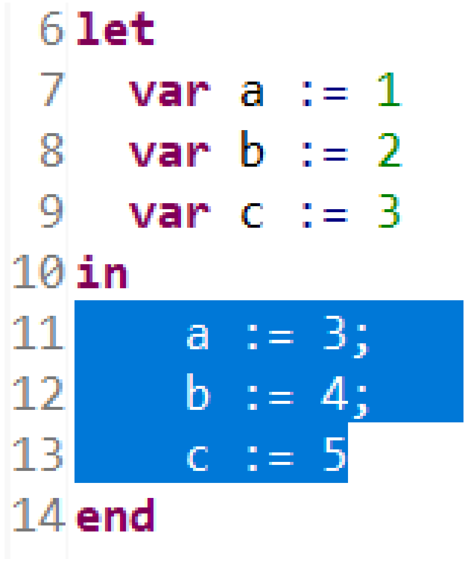
Problem: Partial List Selection



```
if row[r]=0 & diag1[r+c]=0 & diag2[r+7-c]=0
then
  (
    row[r]:=1; diag1[r+c]:=1; diag2[r+7-c]:=1;
    col[c]:=r;
    try(c+1);
    row[r]:=0;
    diag1[r+c]:=0;
    diag2[r+7-c]:=0
  )
```

Extract Function

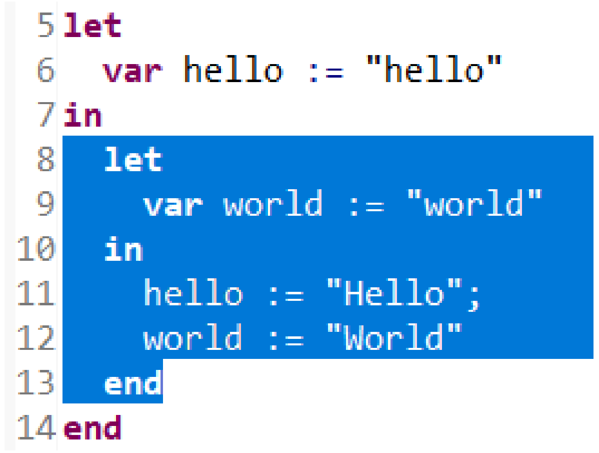
Problem: Multiple Writes



```
let
  var a := 1
  var b := 2
  var c := 3
in
      a := 3;
      b := 4;
      c := 5
end
```

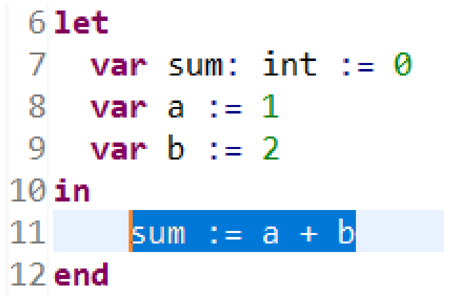
Extract Function

Problem: Internal Variables

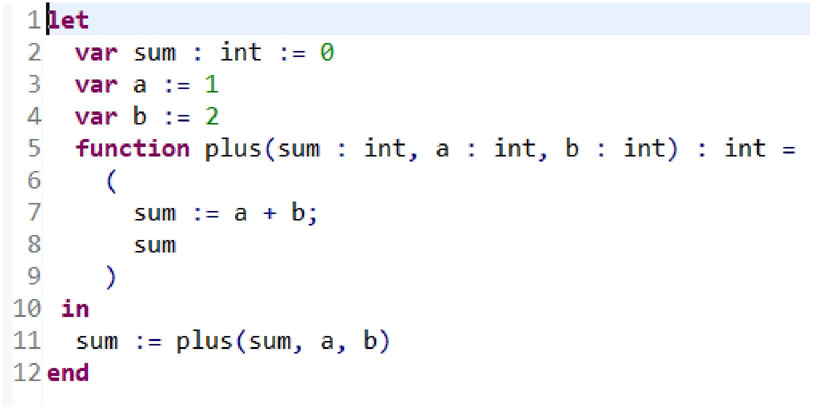

```
let
  var hello := "hello"
in
  let
    var world := "world"
  in
    hello := "Hello";
    world := "World"
  end
end
```

Extract Function

DEMO

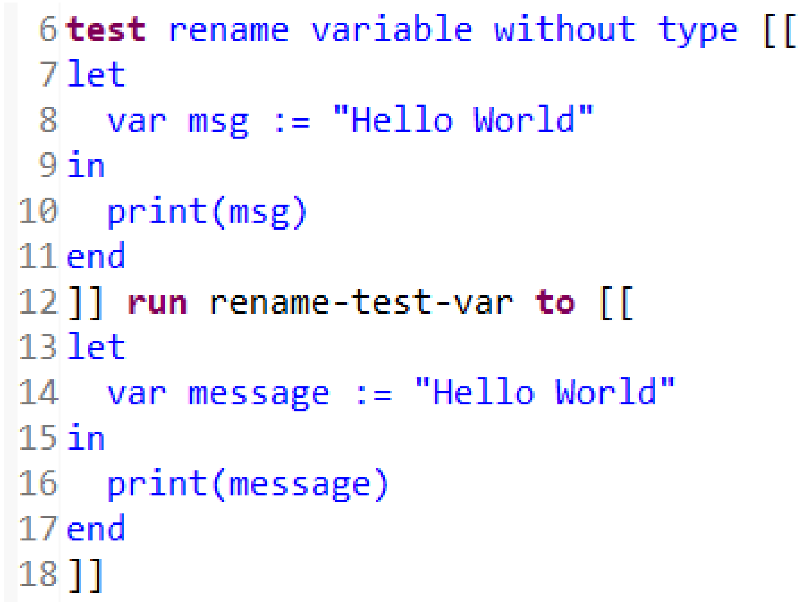


```
let
  var sum: int := 0
  var a := 1
  var b := 2
in
      sum := a + b
end
```



```
let
  var sum : int := 0
  var a := 1
  var b := 2
  function plus(sum : int, a : int, b : int) : int =
    (
      sum := a + b;
      sum
    )
 in
  sum := plus(sum, a, b)
end
```

Testing in SPT



```
test rename variable without type [[
let
  var msg := "Hello World"
in
  print(msg)
end
]] run rename-test-var to [[
let
  var message := "Hello World"
in
  print(message)
end
]]
```

Testing in SPT

```
rename-test-var : ast -> result
  where
    old-name := "msg"
    ; new-name := "message"
    ; path := "test/renaming/variables.spt"
    ; target-dec := VarDecNoType(old-name, Nil())
    ; result := <exec-rename-test(fail | old-name, new-name, target-dec, path, "Var")> ast
```

TUDelft

Conclusion

Stratego Transformations

Nabl2 API

Scope Graph

Eclipse UI Integration

Tricky

Transformations in SPT

TU Delft