

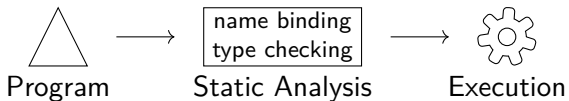
A Constraint-based Approach to Name Binding and Type Checking using Scope Graphs

Hendrik van Antwerpen

Delft University of Technology, The Netherlands

January 7, 2016

Introduction



```
1 let x = 1
2 in "two" * y
```

Outline

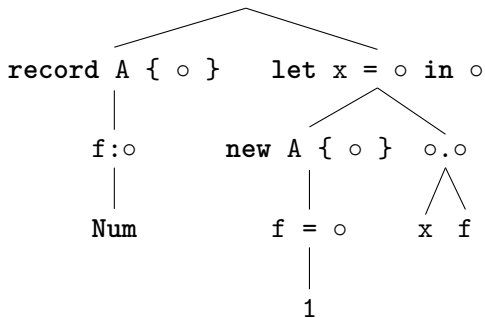
- ① Traditional Approach
- ② A Theoretical Model for Name Binding
- ③ Improved Approach
- ④ Evaluation

Motivating Example

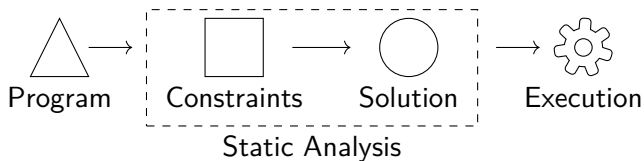
Program

```
1 record A {  
2   f:Num  
3 }  
4  
5 let  
6   x = new A {  
7     f = 1  
8   }  
9 in x.f
```

Abstract Syntax Tree



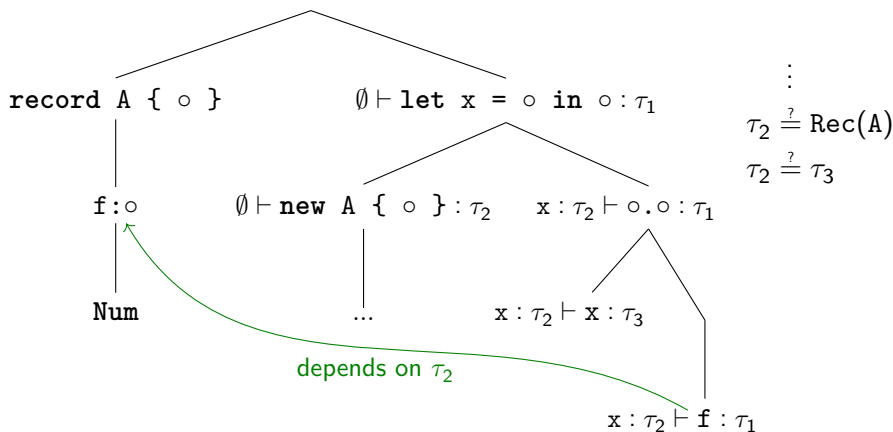
Traditional Constraint-based Approach: Overview



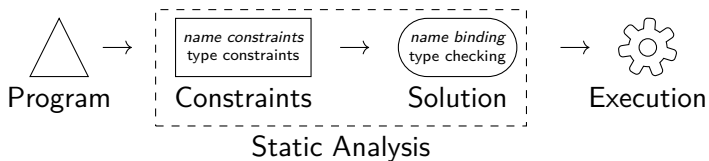
Traditional Constraint-based Approach: Example

Top-down Traversal

Constraints



Improved Approach

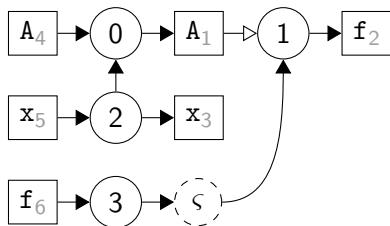


Intermezzo: Scope Graphs

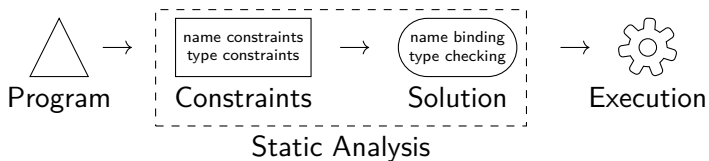
Program

```
1 record A1 {  
2   f2:Num  
3 }  
4  
5 let  
6   x3 = new A4 {  
7     ...  
8   }  
9 in x5.f6
```

Scope Graph



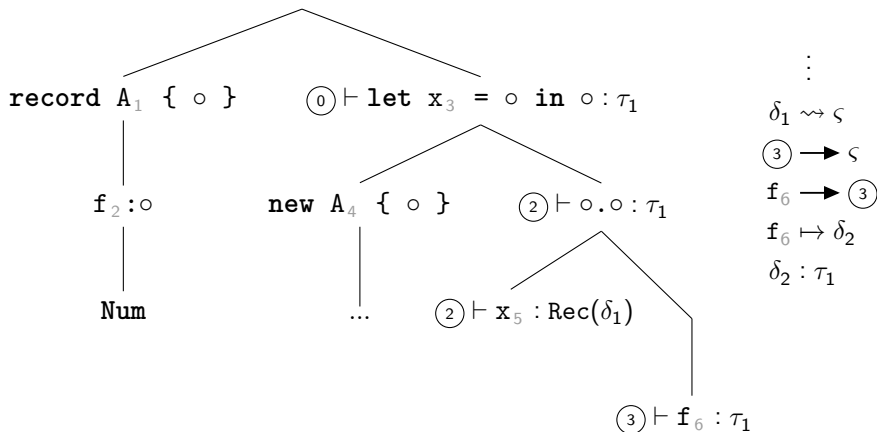
Improved Approach: Overview



Improved Approach: Constraints

Top-down Traversal

Constraints



Improved Approach: Solver

Program

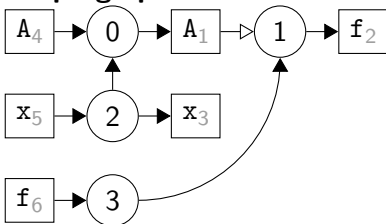
```
1 record A1 {  
2   f2:Num  
3 }  
4  
5 let  
6   x3 = new A4 {  
7     ...  
8   }  
9 in x5.f6
```

Constraints

⋮
A₁ \rightsquigarrow ς
③ \rightarrow ς
f₆ \rightarrow ③
f₆ \mapsto δ_2
f₂ : τ_1

Solution

Scope graph



Var Types

f₂ : Num
x₃ : Rec(A₁)

Substitution

$\delta_1 \mapsto A_1$ $\delta_2 \mapsto f_2$
 $\varsigma \mapsto$ ① $\tau_1 \mapsto$ Num

Improved Approach: Evaluation

- Expressiveness of Constraints
 - Functional language: PCF
 - Object-oriented language: Featherweight Java
- Effectiveness of Solver
 - Sound and terminating
 - Incomplete
 - Prototype implementation

Conclusion

- Constraint-based name binding
- Constraint solver
- Support variety of languages
- Future work:
 - Support advanced types, e.g. polymorphism
 - Efficient constraint solving