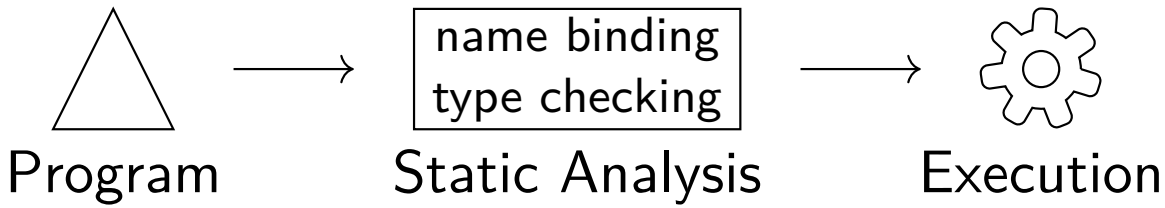
A Constraint-based Approach to Name Binding and Type Checking using Scope Graphs

Hendrik van Antwerpen

Delft University of Technology, The Netherlands

January 7, 2016

Introduction




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

Outline

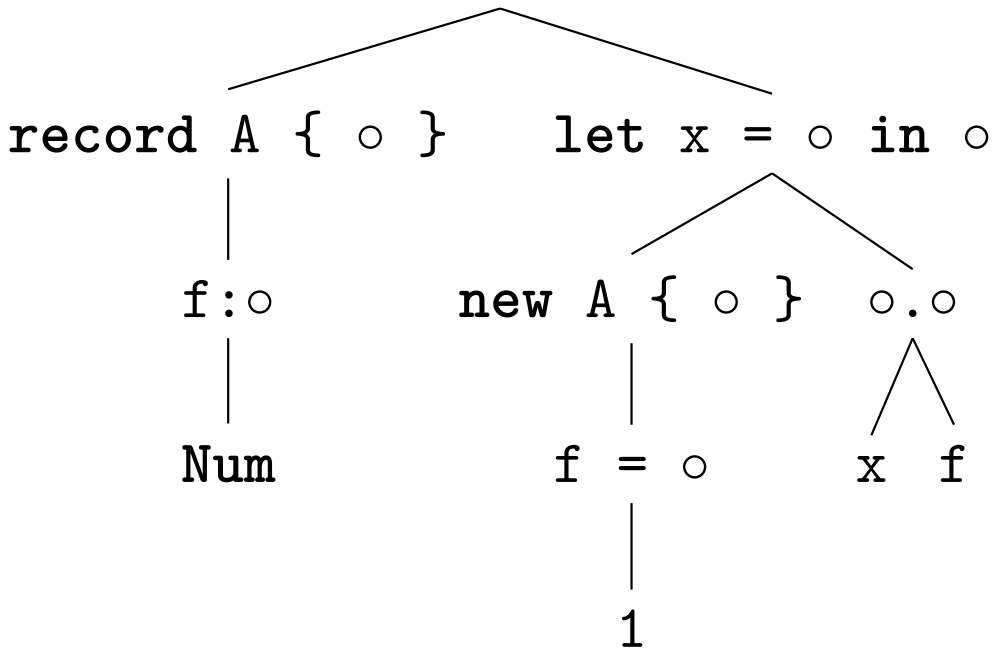
1. Traditional Approach
2. A Theoretical Model for Name Binding
3. Improved Approach
4. Evaluation

Motivating Example

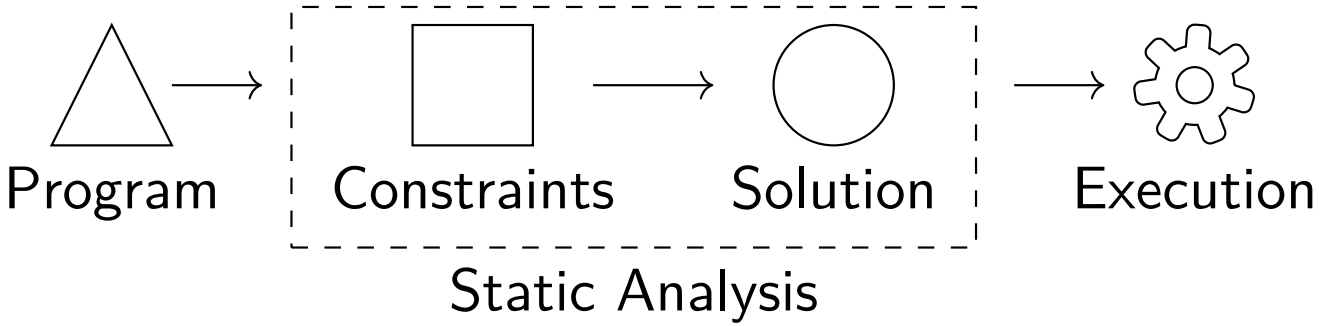
Program

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

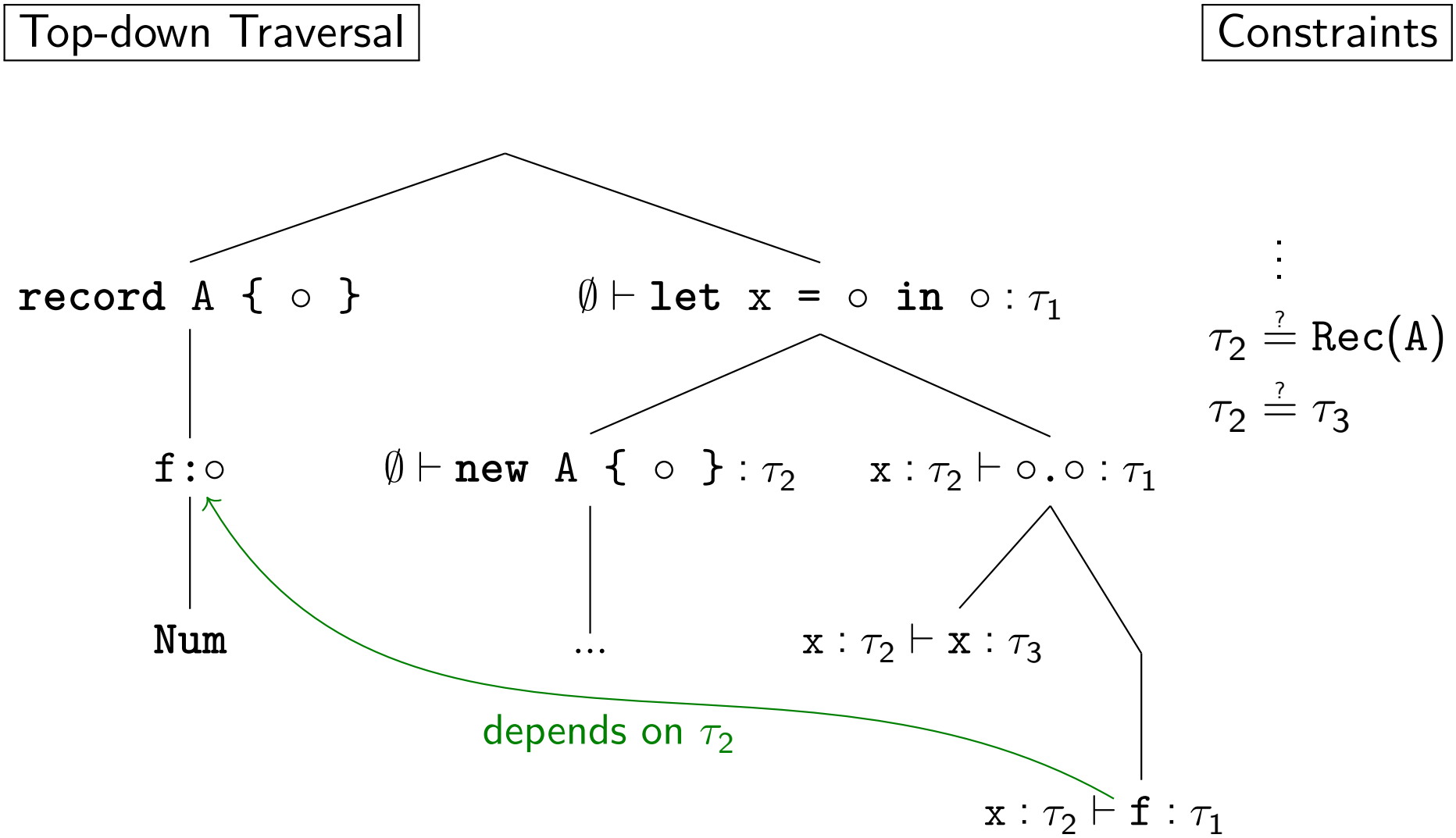
Abstract Syntax Tree



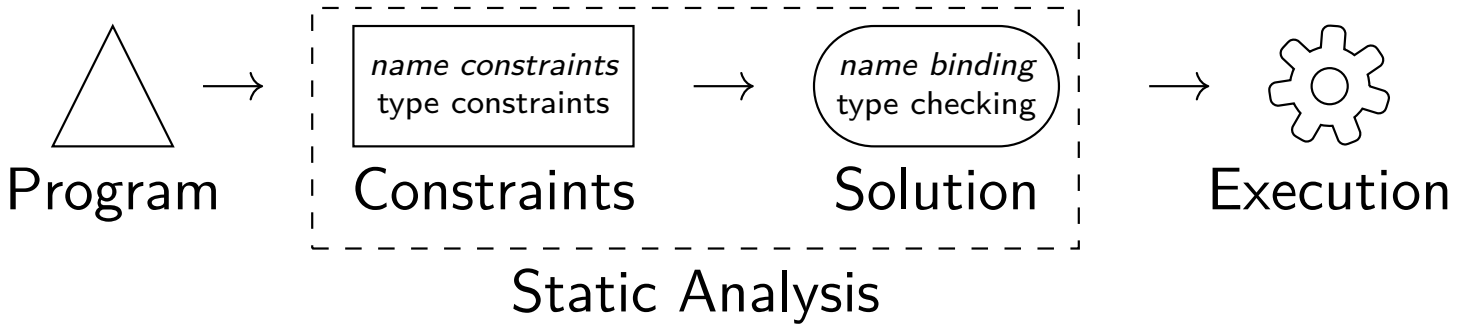
Traditional Constraint-based Approach: Overview



Traditional Constraint-based Approach: Example



Improved Approach

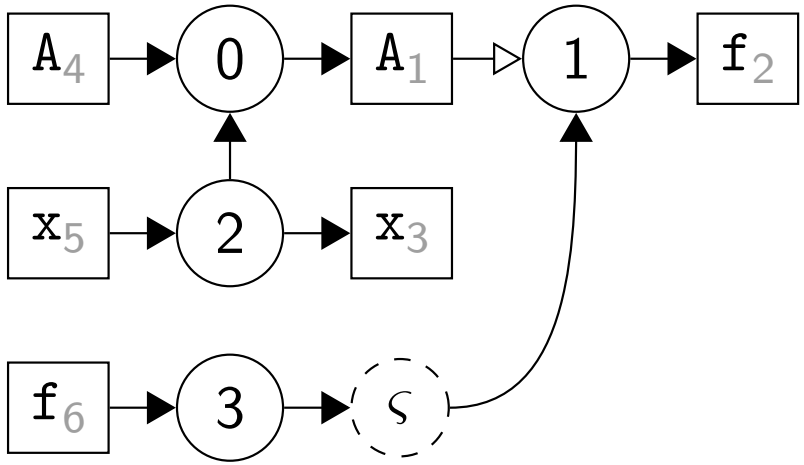


Intermezzo: Scope Graphs

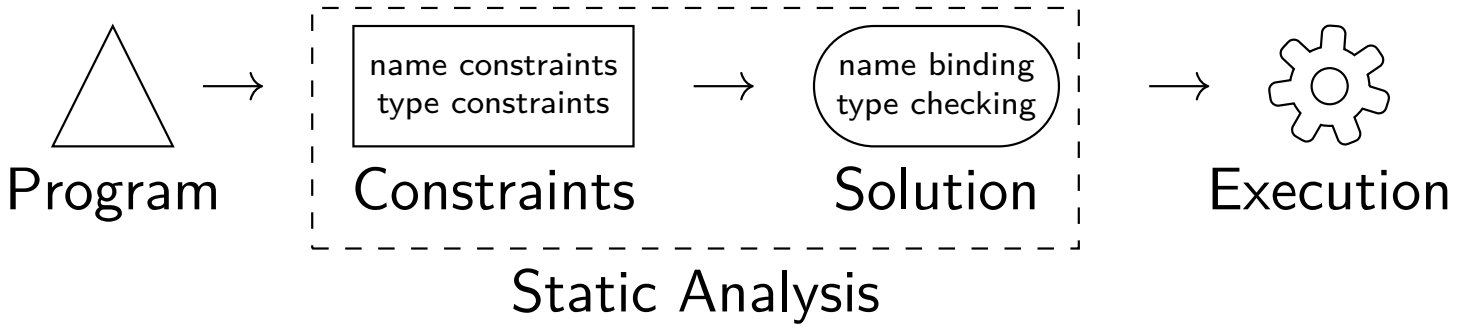
Program

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

Scope Graph

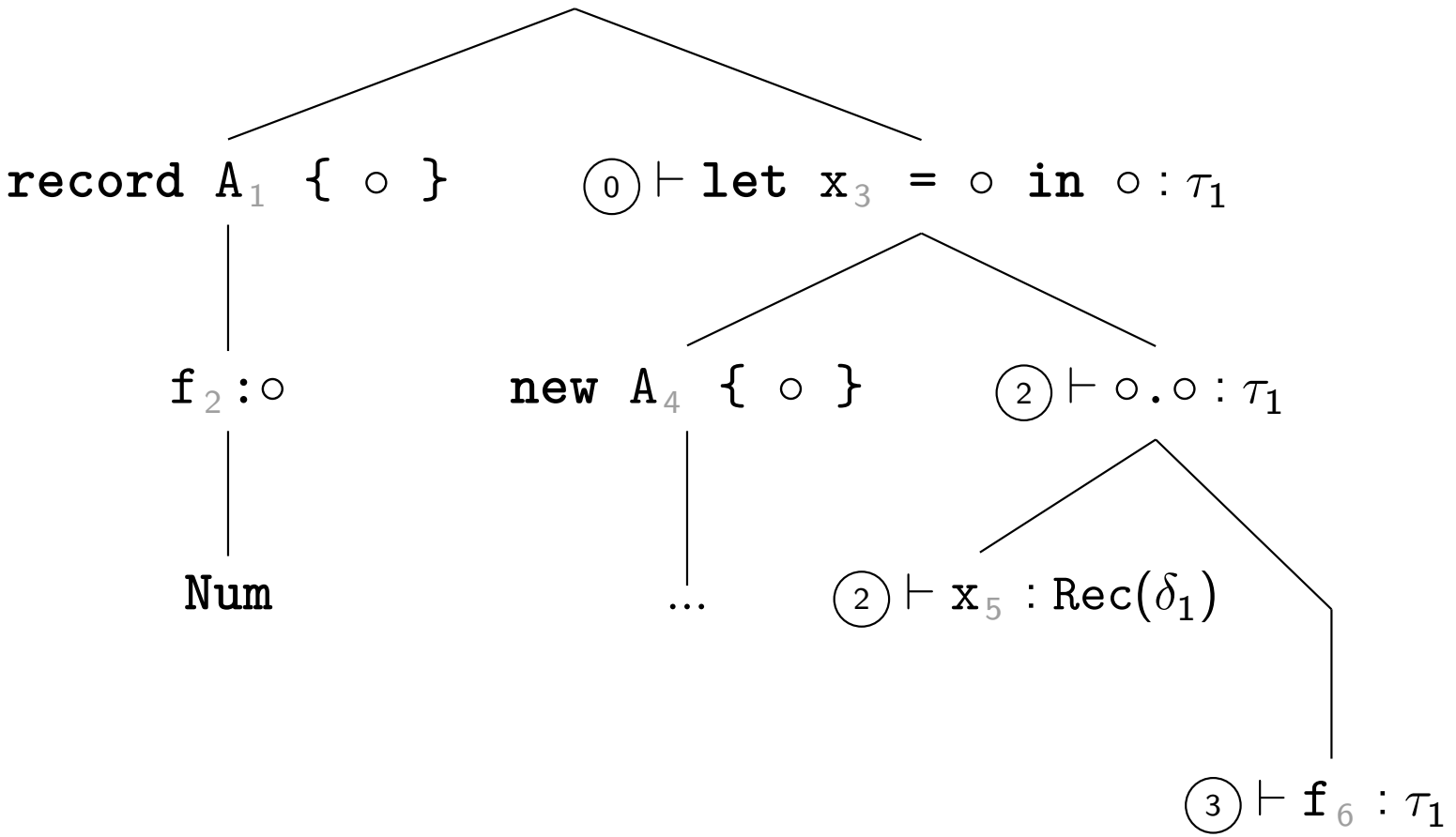


Improved Approach: Overview

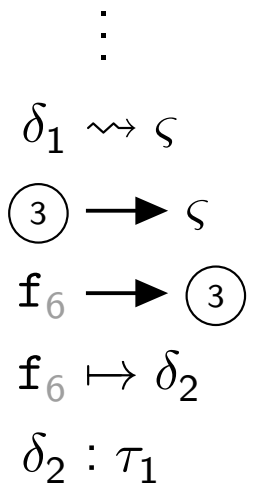


Improved Approach: Constraints

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

$$\vdots$$

$$\mathtt{A}_1 \rightsquigarrow \varsigma$$

$$\textcircled{3} \longrightarrow \varsigma$$

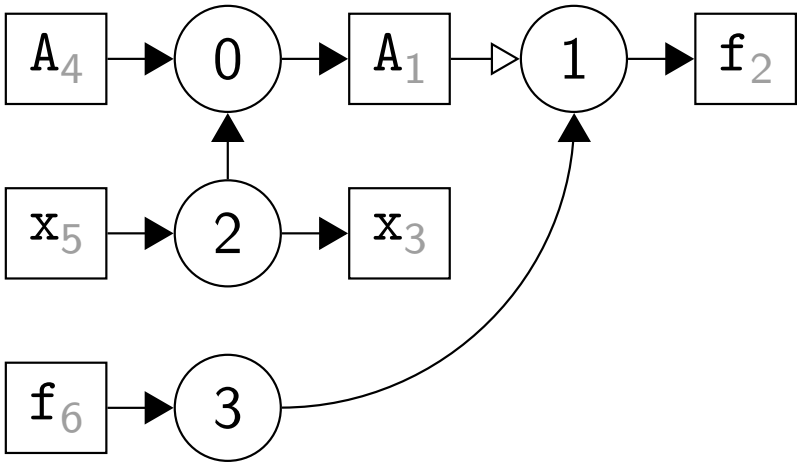
$$\mathtt{f}_6 \longrightarrow \textcircled{3}$$

$$\mathtt{f}_6 \mapsto \delta_2$$

$$\mathtt{f}_2 : \tau_1$$

Solution

Scope graph



Var Types

$\mathtt{f}_2 : \mathtt{Num}$

$\mathtt{x}_3 : \mathtt{Rec}(\mathtt{A}_1)$

Substitution

$\delta_1 \mapsto \mathtt{A}_1 \quad \delta_2 \mapsto \mathtt{f}_2$

$\varsigma \mapsto \textcircled{1} \quad \tau_1 \mapsto \mathtt{Num}$

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