On Interactive Synthesis of Code Snippets

Tihomir Gvero, Ruzica Piskac, Viktor Kuncak

Introduction

- Search for method composition in large and complex libraries
  - Substantial effort
  - Obstructing
- Auto Complete in IDE
  - Predicting a symbol that the user wants to type
  - No need to read source code and doc.
- Code Synthesis – Synthesizes code from specification

Introduction

- Search for method composition in large and complex libraries
  - Substantial effort
  - Obstructing
- Auto Complete in IDE
  - Predicting a symbol that the user wants to type
  - No need to read source code and doc.
- Code Synthesis – Synthesizes code from specification
- Can we combine Auto Complete with Code Synthesis?

Introduction

- Search for method composition in large and complex libraries
  - Substantial effort
  - Obstructing
- Auto Complete in IDE
  - Predicting a symbol that the user wants to type
  - No need to read source code and doc.
- Code Synthesis – Synthesizes code from specification
- Can we combine Auto Complete with Code Synthesis? YES

Introduction

- Search for method composition in large and complex libraries
  - Substantial effort
  - Obstructing
- Auto Complete in IDE
  - Predicting a symbol that the user wants to type
  - No need to read source code and doc
- Code Synthesis – Synthesizes code from specification
- **iSynth** - a tool that suggests multiple meaningful expressions at a given program point

iSynth

- Implemented for Scala language
- Input:
  - Partial Scala program – visible declarations
  - Program point – desired type

  } **Type Information**
- Output:
  - Code snippets – expression with desired type
- Runs synthesis algorithm based on resolution to find candidate snippets
- Handles simple and generic (parametric) types

Example

```
def fopen(name:String):File = { ... }
def fread(f:File, p:Int):Data = { ... }
var currentPos : Int = 0
var fname : String= null

...
def getData():Data = ?
```

Example

```
def fopen(name:String):File = { ... }
def fread(f:File, p:Int):Data = { ... }
var currentPos : Int = 0
var fname : String= null

...
def getData():Data = ?
```

DEMO

Example – Parametric types

```
def map[A,B](f: A=>B, l:List[A]):List[B] = { ... }
def stringConcat(lst : List[String]) :String = { ... }
...
def printInts(intList: List[Int], fun:Int=>String):String = ?
```

Example – Parametric types

```
def map[A,B](f: A=>B, l:List[A]):List[B] = { ... }
def stringConcat(lst : List[String]) :String = { ... }
...
def printInts(intList: List[Int], fun:Int=>String):String = ?
```

DEMO

Example – User Preferences

```
object Calendar {
  private val events:List[Event] = { ... }

  def reserve(user:User, date:Date):Event = { ... }
  def getEvent(user:User, date:Date):Event = { ... }
  def remove(user:User, date:Date):Event = ?
}
```

Example – User Preferences

```
object Calendar {
  private val events:List[Event] = { ... }

  def reserve(user:User, date:Date):Event = { ... }
  def getEvent(user:User, date:Date):Event = { ... }
  def remove(user:User, date:Date):Event = ?
}
```

DEMO

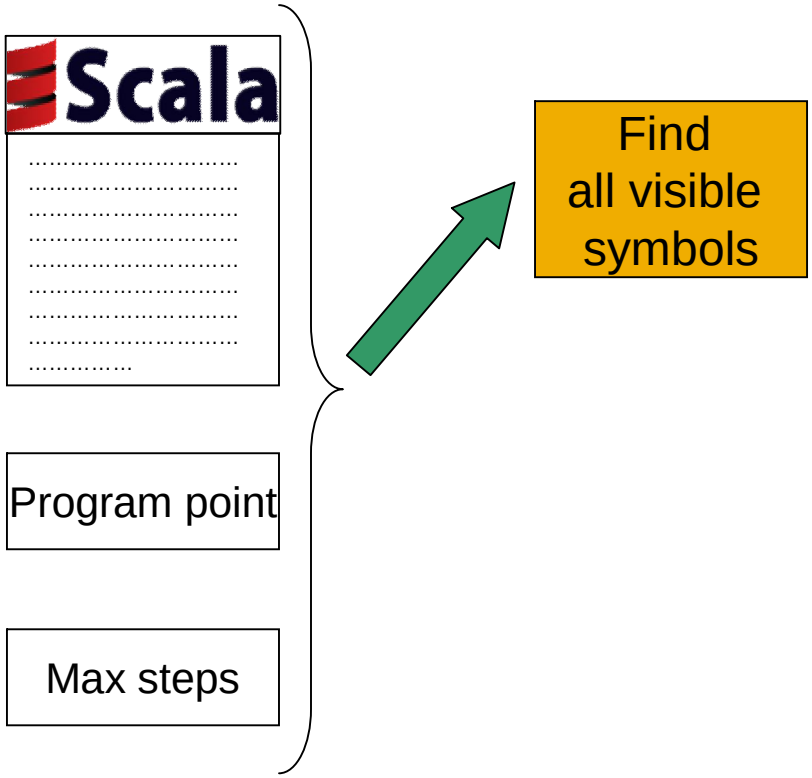
Snippet Synthesis Algorithm

.....
.....
.....
.....
.....
.....
.....
.....
.....

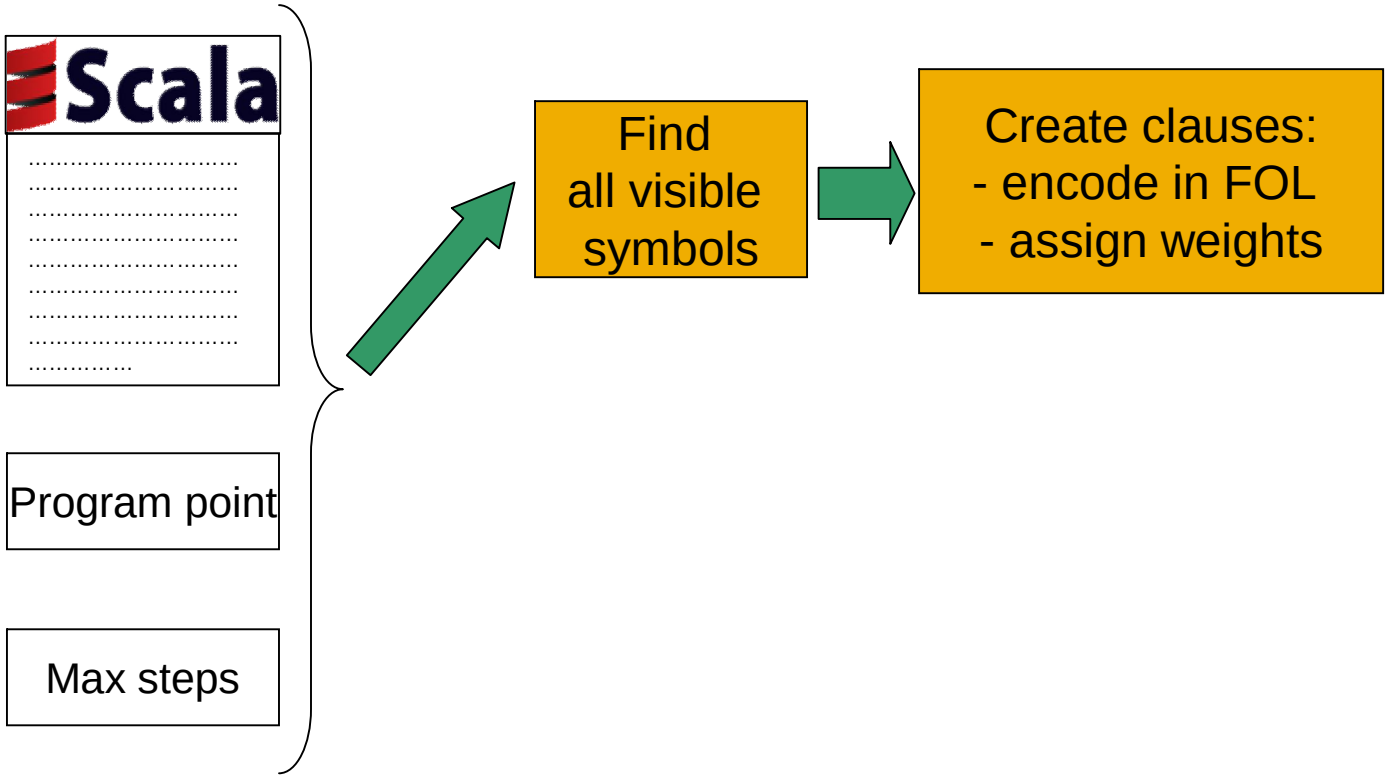
Program point

Max steps

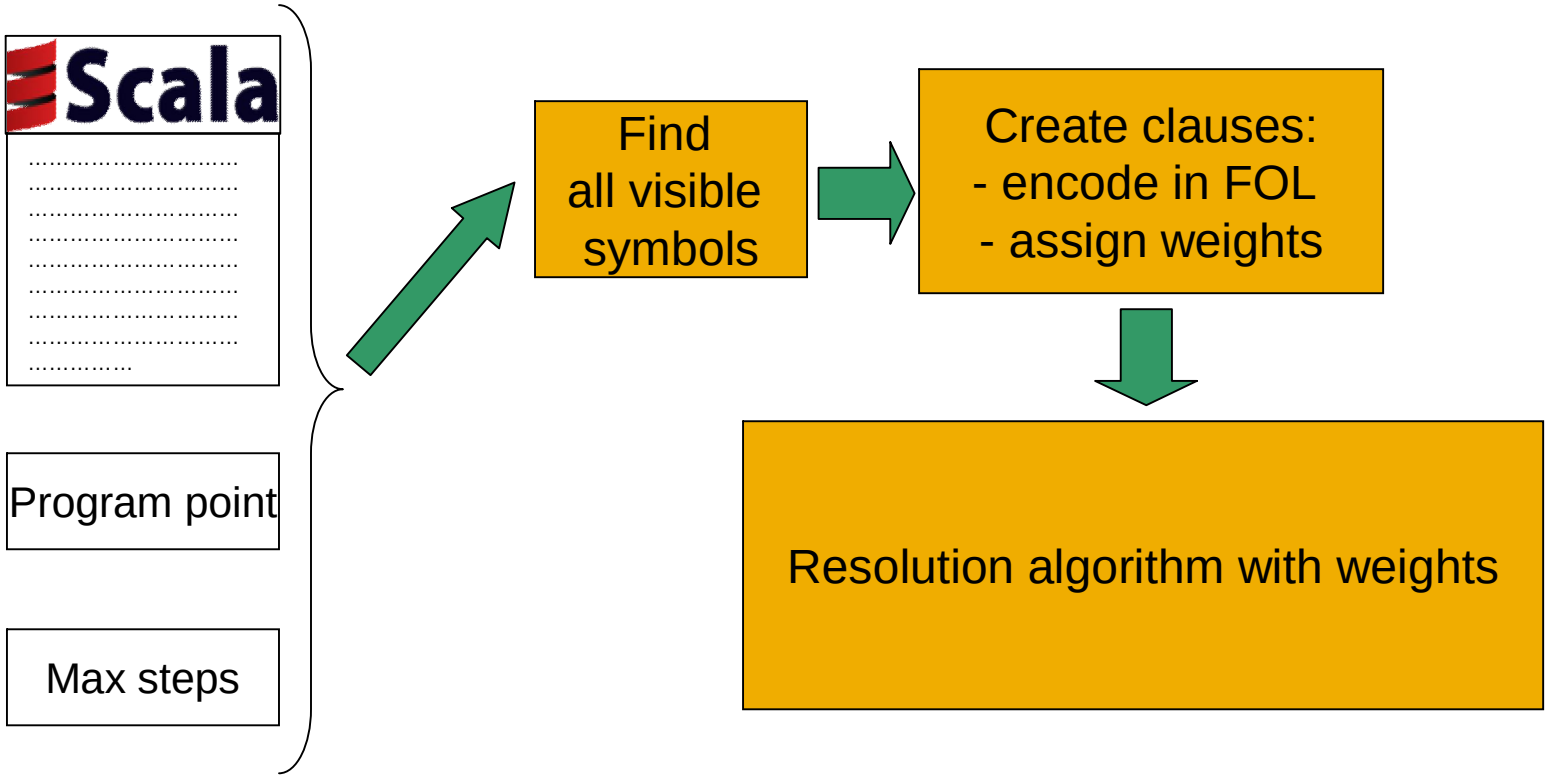
Snippet Synthesis Algorithm



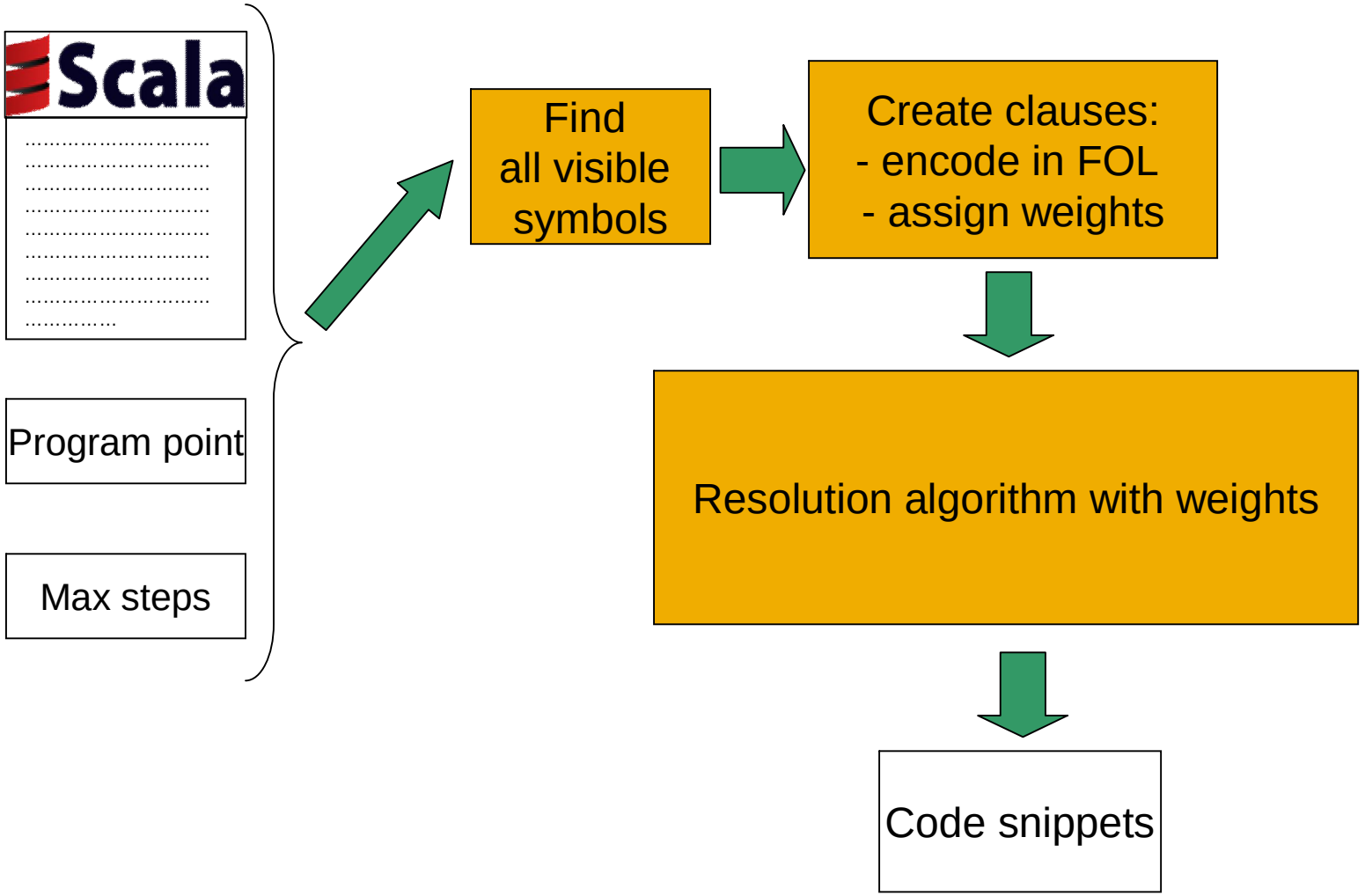
Snippet Synthesis Algorithm



Snippet Synthesis Algorithm



Snippet Synthesis Algorithm



Encoding in FOL

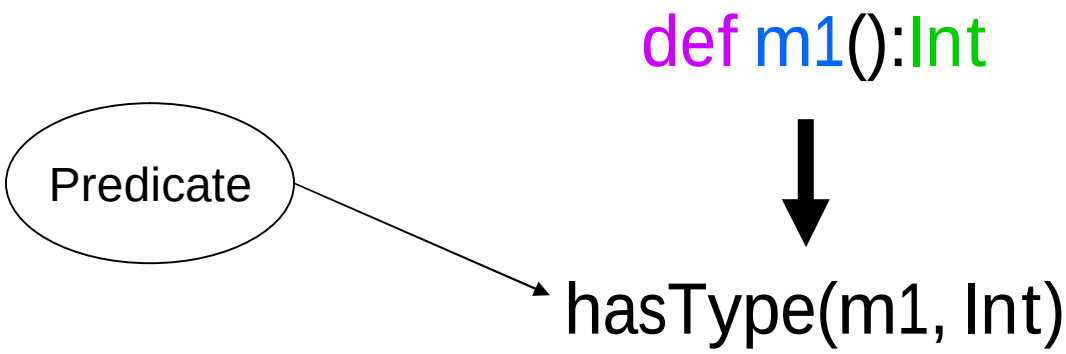
```
def m1():Int
```

Encoding in FOL

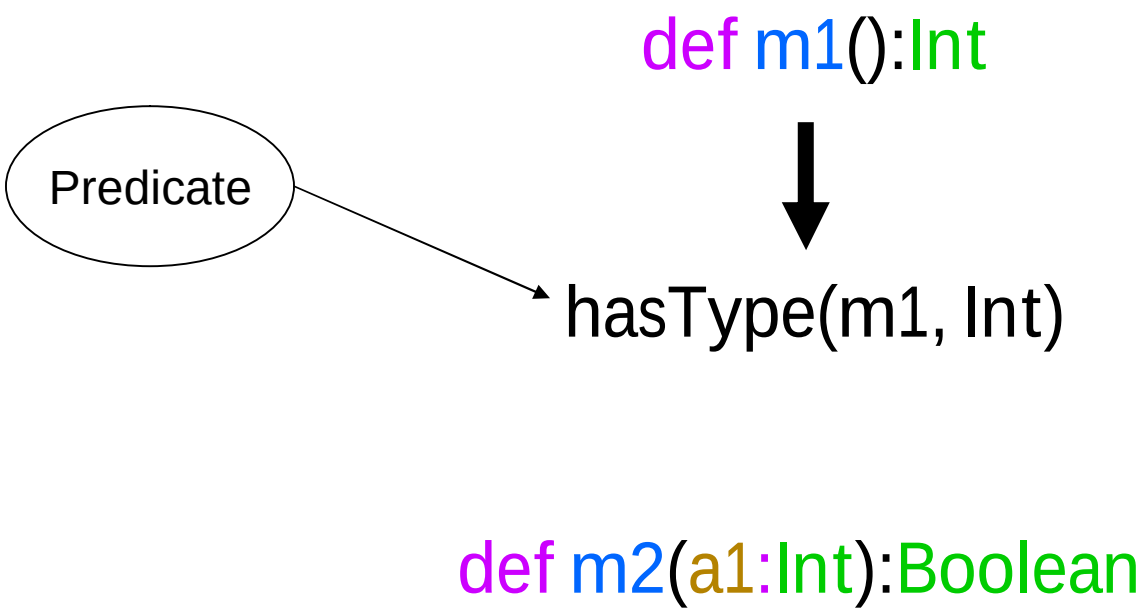
```
def m1():Int
```

hasType(m1, Int)

Encoding in FOL

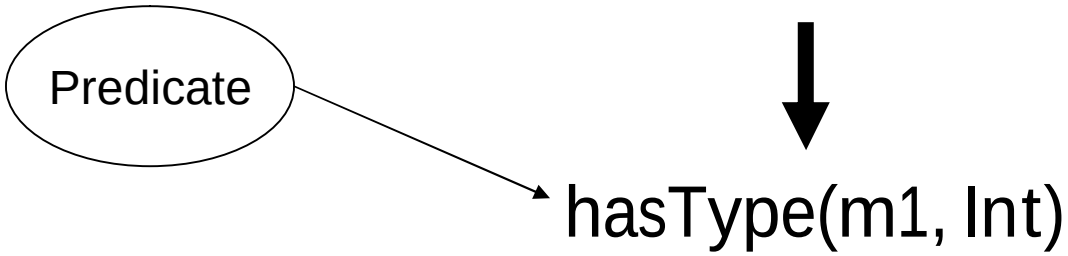


Encoding in FOL



Encoding in FOL

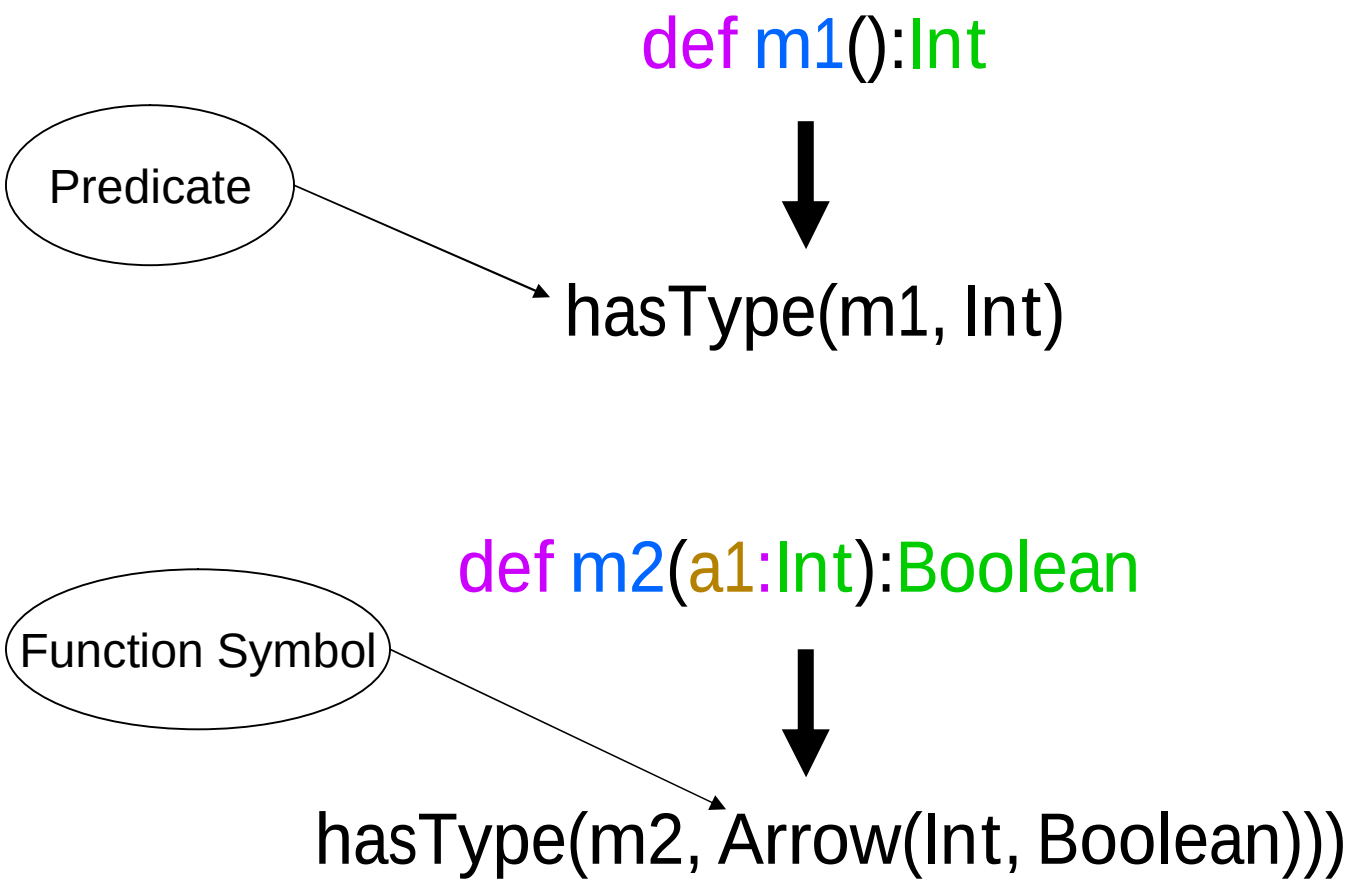
```
def m1():Int
```




```
def m2(a1:Int):Boolean
```

hasType(m2, Arrow(Int, Boolean)))

Encoding in FOL



Encoding in FOL

```
def m3(a1: Int, a2:Boolean):String
```

Encoding in FOL

```
def m3(a1: Int, a2:Boolean):String
```

hasType(m_3, Arrow(Int, Arrow(Boolean, String)))

Encoding in FOL

```
def m3(a1: Int, a2:Boolean):String
```

hasType(m_3, Arrow(Int, Arrow(Boolean, String)))

```
def m4(fun:Int=> Boolean):String
```

Encoding in FOL

def m_3(a1: Int, a2:Boolean):String

hasType(m_3, Arrow(Int, Arrow(Boolean, String)))

def m_4(fun:Int=>Boolean):String

hasType(m_4, Arrow(Arrow(Int,Boolean),String))

Encoding in FOL

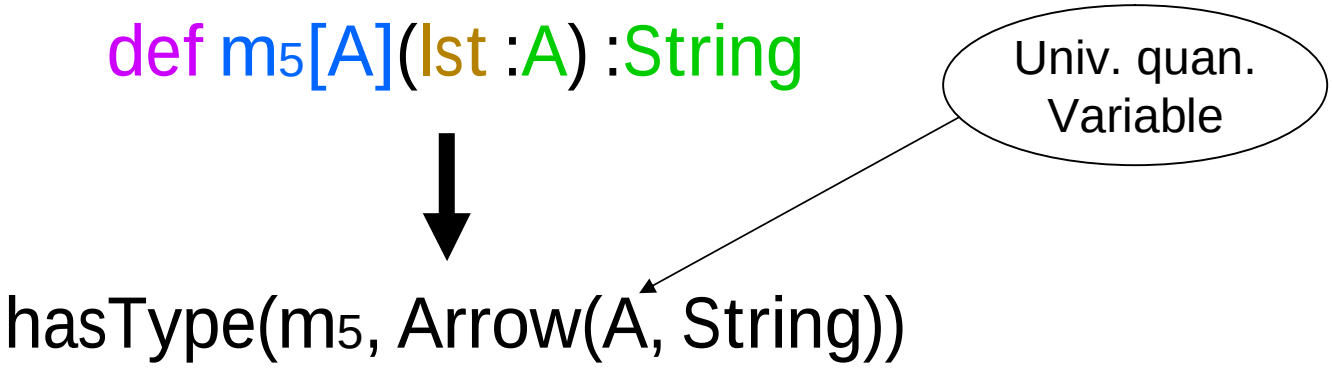
def m_5[A](lst :A) :String

Encoding in FOL

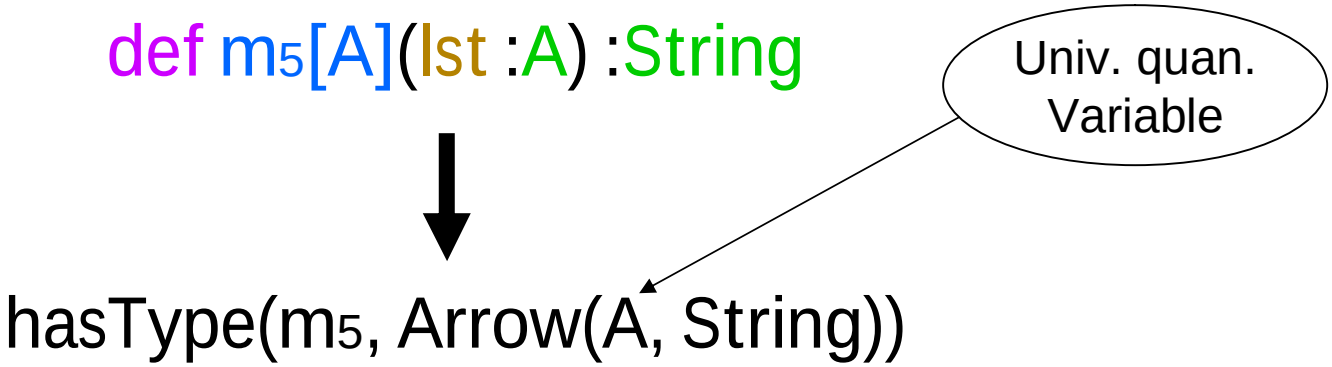
def m_5[A](lst :A) :String

hasType(m_5, Arrow(A, String))

Encoding in FOL

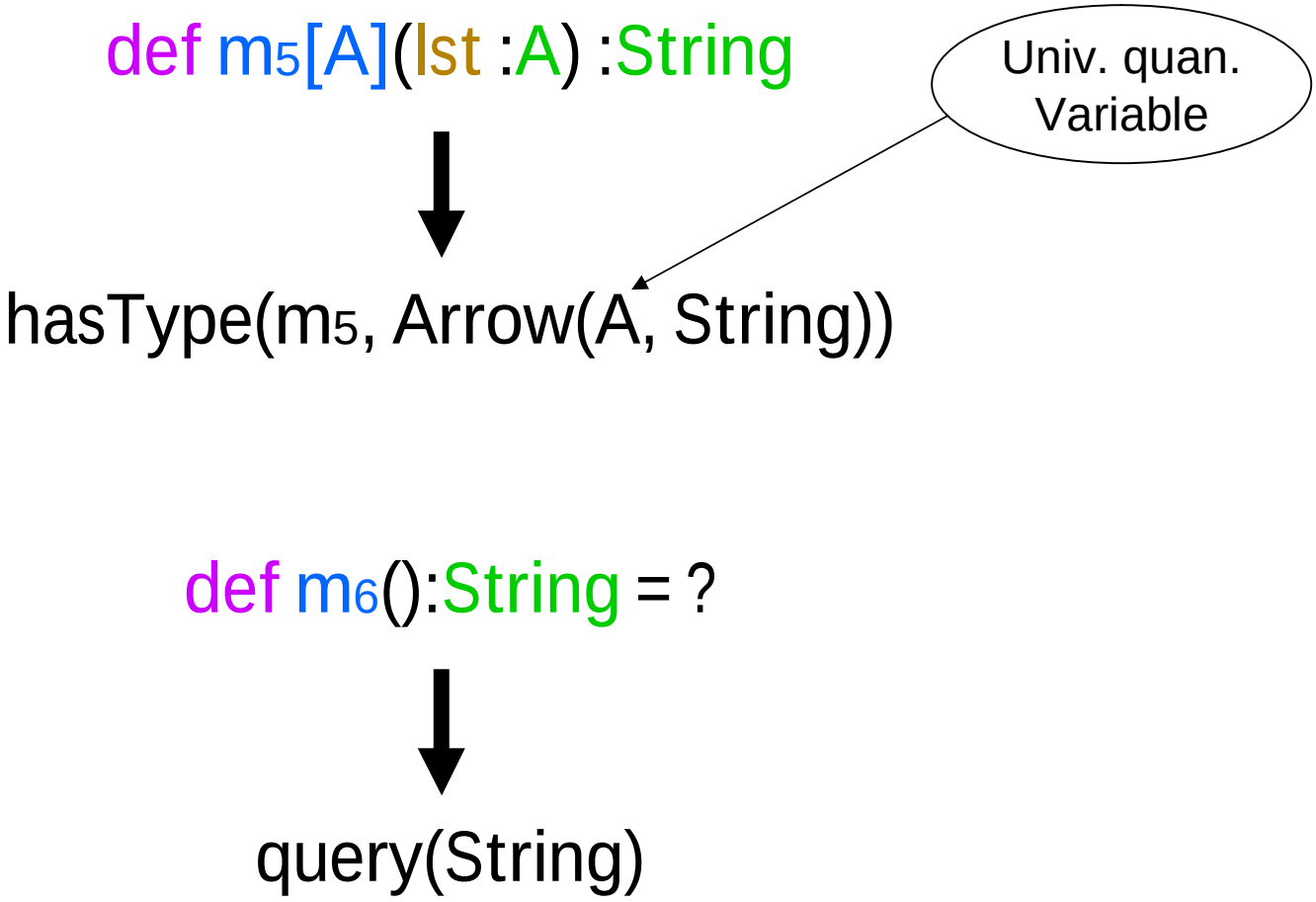


Encoding in FOL

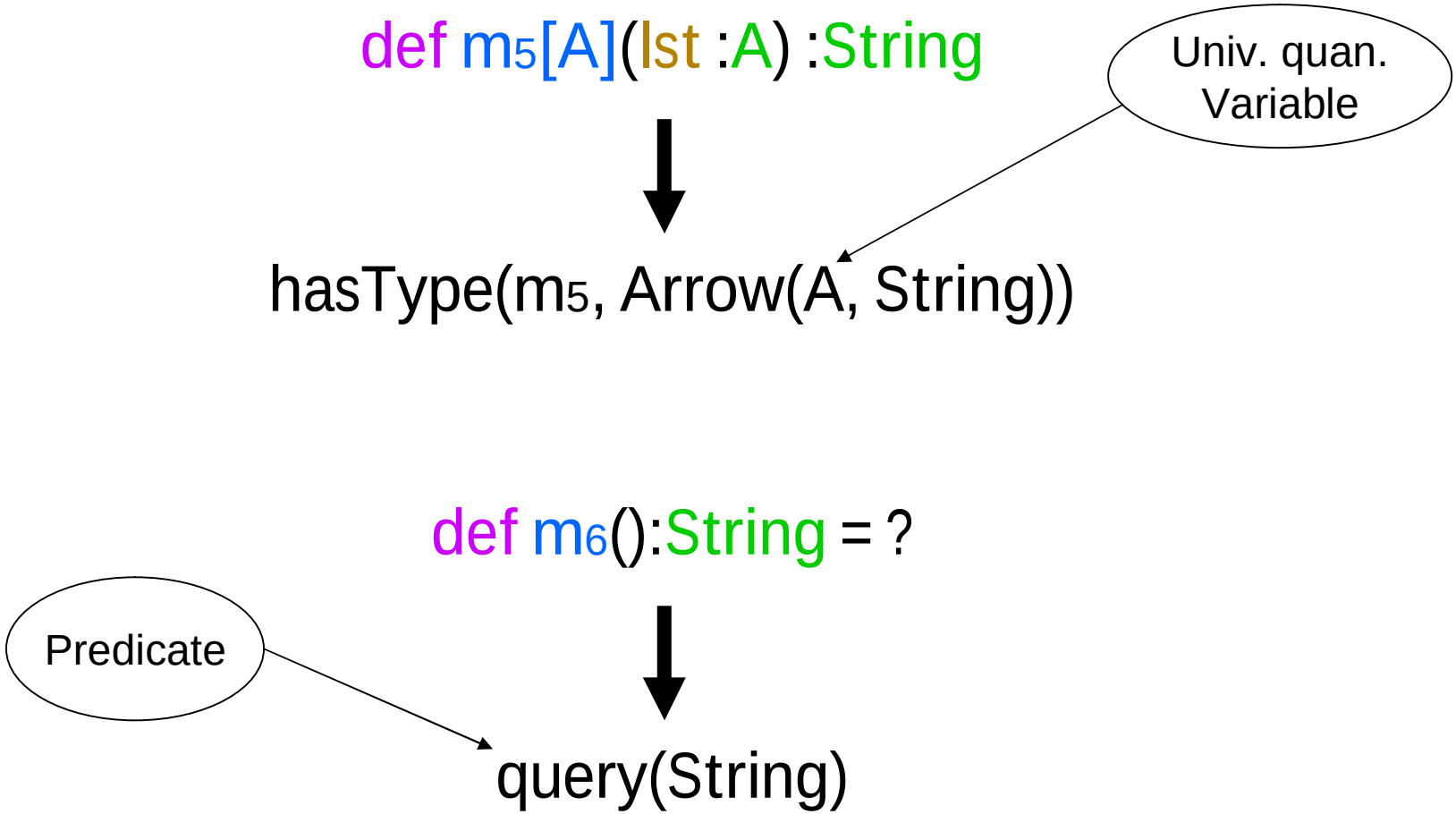


def m_6():String = ?

Encoding in FOL



Encoding in FOL



Rules

- Forward Reasoning:

$$\frac{\text{hasType}(f, \text{Arrow}(T1, T2)) \qquad \text{hasType}(a, T1)}{\text{hasType}(f(a), T2)}$$

- Example:

$$\frac{\texttt{def f(arg:T1):T2} \qquad \texttt{var a:T1}}{\texttt{f(a): T2}}$$

Rules

- Backward Reasoning:

$$\frac{\text{hasType}(f, \text{Arrow}(T1, T2)) \quad \text{query}(T2)}{\text{query}(T1)}$$

- Example:

$$\frac{\texttt{def f(arg:T1):T2} \qquad \texttt{var a:T2 = ?}}{\texttt{arg: T1 = ?}}$$

Rules

- Deriving empty clause:

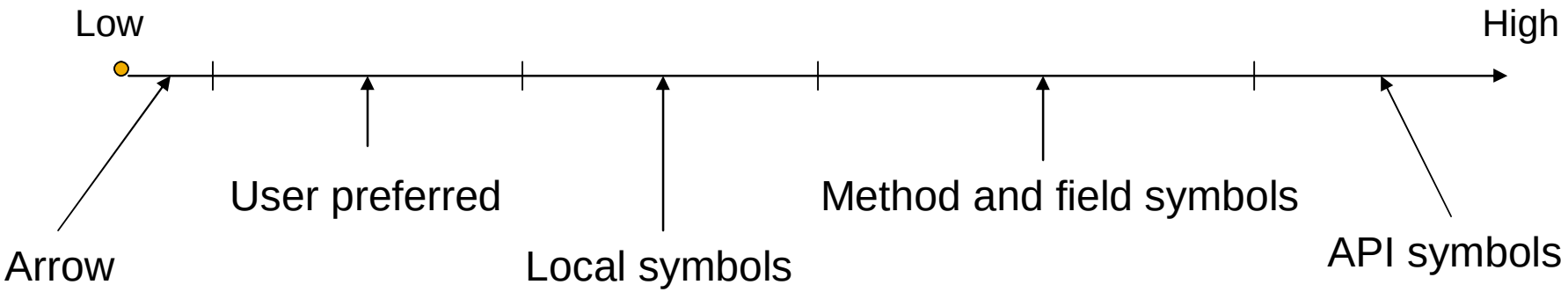
$$\frac{\text{hasType}(x, T1) \qquad \text{query}(T1)}{\blacksquare}$$

- Example:

$$\frac{\texttt{def f():T1} \qquad \texttt{var a:T1 = ?}}{\texttt{f()}}$$

Weights

- Symbol weights



- Term weights
  - Knuth-Bendix ordering
  - Recalculate weight of terms with user preferred symbols
- Clause weights
  - Sum of all terms' weight in clause

Evaluation

Program	# Loaded Declarations	# Methods in Synthesized Snippets	Time [s]
FileReader	5	4	0.001
Map	3	3	< 0.001
FileManager	7	3	0.001
Calendar	29	3	0.001
FileWriter	442	6	0.72
SwingBorder	161	2	0.02
TcpService	112	3	0.62

Conclusion

- **iSynth** - the first interactively deployed synthesis tool:
  - Parameterized types
  - Weights indicating preferences
- Based on a variation of an ordered resolution calculus
- We have found to be fast enough and helpful in synthesizing meaningful code fragments
- Future work: Weights mining

Thank you