

Scaps: Type-Directed API Search for Scala

Lukas Wegmann, 1plusX; Farhad Mehta, HSR;
Peter Sommerlad, HSR; Mirko Stocker, HSR

"A Hooogle for Scala"

Scaps: Type-Directed API Search for Scala

Lukas Wegmann, 1plusX; Farhad Mehta, HSR;
Peter Sommerlad, HSR; Mirko Stocker, HSR

Outline

- **Why types for API search?**
- Fingerprint Evaluation Model
 - Type Fingerprints
 - Query Expression Trees
- Conclusion

Claim: It's hard to discover functionality
in Scala libraries

Claim: It's hard to discover functionality in Scala libraries

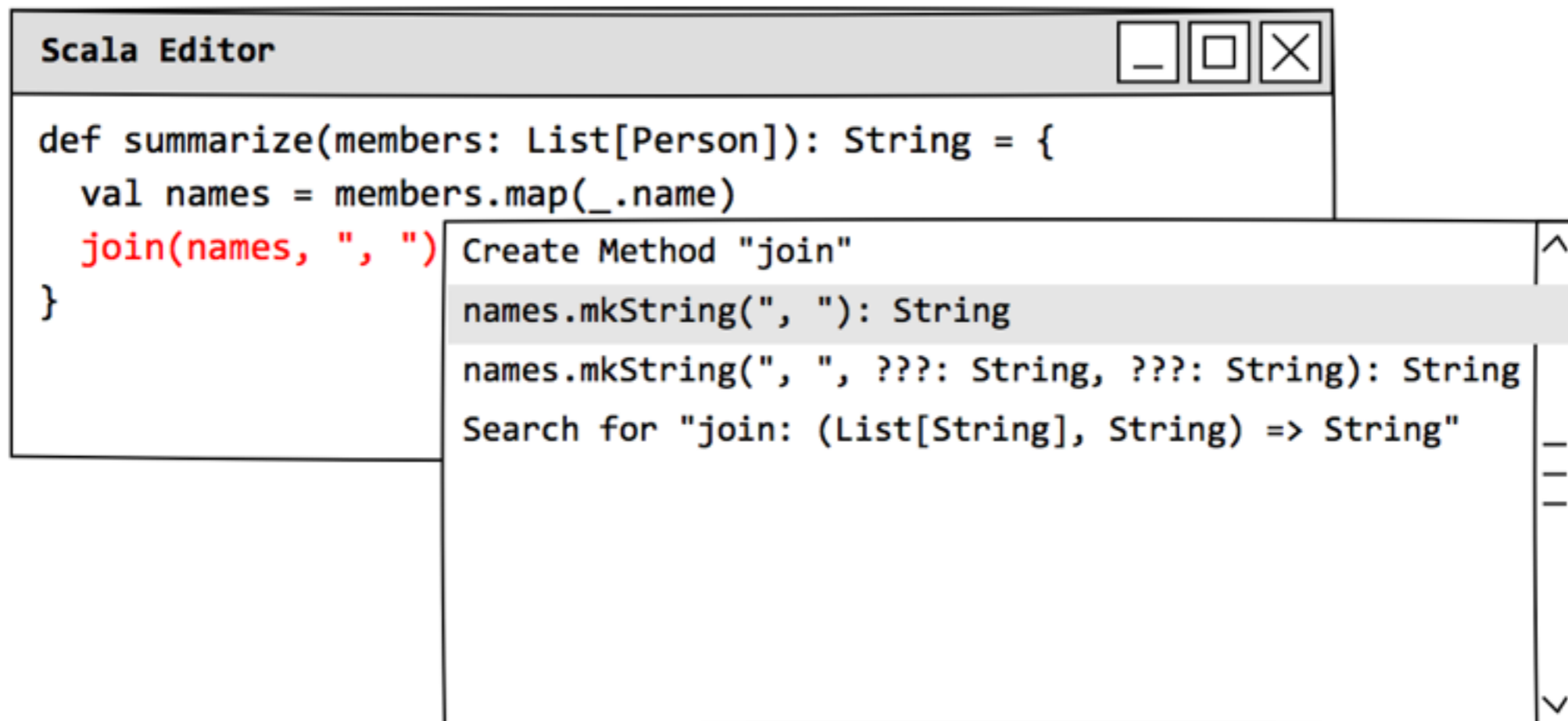
- Functional programming allows library designers to provide a large number of useful abstractions
 - `java.util.List`: ~30 members
 - `scala.collection.immutable.List`: ~150 members

Claim: It's hard to discover functionality in Scala libraries

- Functional programming allows library designers to provide a large number of useful abstractions
 - `java.util.List`: ~30 members
 - `scala.collection.immutable.List`: ~150 members
- Types are often open for extensions for third parties
 - Implicit conversions
 - Utility objects

Vision

Use type information from the editor's context to discover library functionality



The image shows a screenshot of a Scala Editor window. The window title is "Scala Editor". The code in the editor is:

```
def summarize(members: List[Person]): String = {  
  val names = members.map(_.name)  
  join(names, ", ")  
}
```

A code completion popup is visible over the `join` call. The popup contains the following text:

- Create Method "join"
- `names.mkString(", "): String` (highlighted)
- `names.mkString(", ", ???: String, ???: String): String`
- Search for "join: (List[String], String) => String"

The popup also has a search bar and a list of results, with a scroll bar on the right side.

Objective

Find a model that retrieves values from Scala libraries by types and keywords.

Objective

Find a model that **retrieves values** from Scala libraries by types and keywords.

Public
Values

Public
Methods

Public
Constructors

Private
Members

Types

Objective

Find a model that retrieves values from **Scala** libraries by types and keywords.

Existing retrieval models do..

- not support subtyping
- neglect parametric polymorphism
- or limit search to current scope

Objective

Find a model that retrieves values from Scala libraries **by types** and keywords.

Query:

$A \Rightarrow B$

Potentially Useful Implementations:

$A \Rightarrow X \Rightarrow B$

$A1 \Rightarrow B$ if $A <: A1$

$A \Rightarrow B1$ if $B1 <: B$

$A \Rightarrow X[B]$

$X[A] \Rightarrow B$

$(X \Rightarrow A) \Rightarrow B$

$\text{Promise}[B1] \Rightarrow \text{Promise}[A1]$ if $A <: A1, B1 <: B$

Objective

Find a model that retrieves values from Scala libraries **by** types and **keywords**.

```
pi: Double
```

```
max: Double
```

```
print: String => Unit
```

```
log: String => Unit
```

Test Collection

- > 60 queries mined from StackOverflow and personal experience



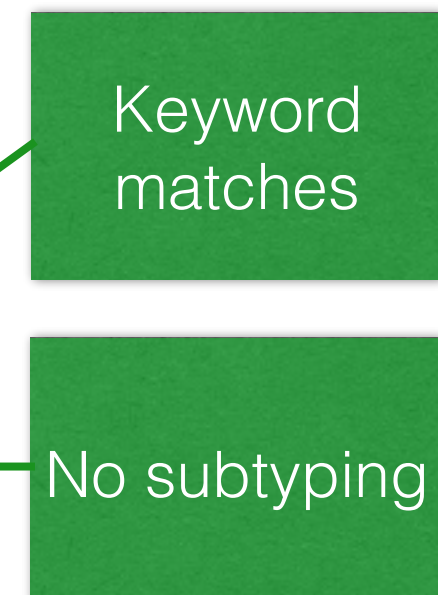
- Covering
 - Scala Standard Library
 - Scala-Refactoring

Baseline: TF-IDF

- Index types by terms:
 - A value of type `List[A] => Option[A]`
 - consists of the terms `List`, `?`, `Option`, `?`
- Use type signatures as returned by scalac
 - Also index inherited members
 - Roughly what you see in Scaladoc
- Apache Lucene to index and retrieve values and doc comments

Evaluation: TF-IDF

	AP* I ₂
All Queries (Mean)	0.66
<code>remove: List[A] => A => List[A]</code>	1
<code>List[A] => (List[A], List[A])</code>	1
<code>List[A] => Option[A]</code>	0.58
<code>List[Future[A]] => Future[List[A]]</code>	0
<code>(List[Int], String) => String</code>	0.14



Keyword matches

No subtyping

* Average Precision

Outline

- Why types for API search?
- Fingerprint Evaluation Model
 - **Type Fingerprints**
 - Query Expression Trees
- Conclusion

Type Fingerprints

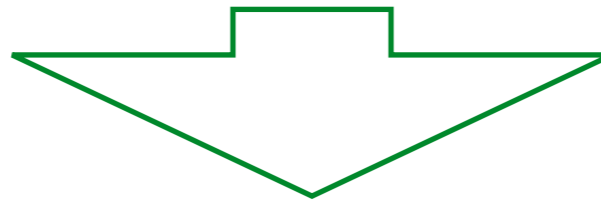
- Decompose types into **atomic, independent terms**
 - Ordering of terms and structure are not relevant

Type Fingerprints

- Decompose types into **atomic, independent terms**
 - Ordering of terms and structure are not relevant
- Allows the use of common text retrieval techniques
 - Term vectors
 - Inverted indexes
 - Relevance statistics similar to TF-IDF

Type Fingerprints

`Array[A].mapFirst(A => B) => Option[B]`



{ -Array, ◦?, -=>, +Nothing, -Any, +Option, +Nothing }

- Member, method and constructor types are normalized
- Variance annotations +, - and ◦ (Co-, contra- and invariant)
- Type parameters substituted by upper/lower bound or ◦?

Type Fingerprints

- Distinguish between types that can be read, written or both
 - $FP(A \Rightarrow \text{Array}[A] \Rightarrow A) = \{ -A, -\text{Array}, \circ A, +A \}$

Type Fingerprints

- Distinguish between types that can be read, written or both
 - $FP(A \Rightarrow \text{Array}[A] \Rightarrow A) = \{ -A, -\text{Array}, \circ A, +A \}$
- Capture relaxed equivalence relations and similarities
 - Isomorphisms: $FP(A \Rightarrow B \Rightarrow C) \approx FP((A, B) \Rightarrow C)$
 - Boxing: $FP(A \Rightarrow B) \subset FP(C[A] \Rightarrow D[B])$
 - Ordering: $FP(A \Rightarrow B \Rightarrow C) = FP(B \Rightarrow A \Rightarrow C)$
 - Type Param Names: $FP(f[A, B]: A \Rightarrow B) = FP(f[X, Y]: X \Rightarrow Y)$

Evaluation: Fingerprints

	AP I ₂	AP I ₃
All Queries (Mean)	0.66	0.67
<code>remove: List[A] => A => List[A]</code>	1	1
<code>List[A] => (List[A], List[A])</code>	1	1
<code>List[A] => Option[A]</code>	0.58	1
<code>List[Future[A]] => Future[List[A]]</code>	0	0
<code>(List[Int], String) => String</code>	0.14	0.2

Type params and common types are more specific

Subtyping & Implicit Conversions

Outline

- Why types for API search?
- Fingerprint Evaluation Model
 - Type Fingerprints
 - **Query Expression Trees**
- Conclusion

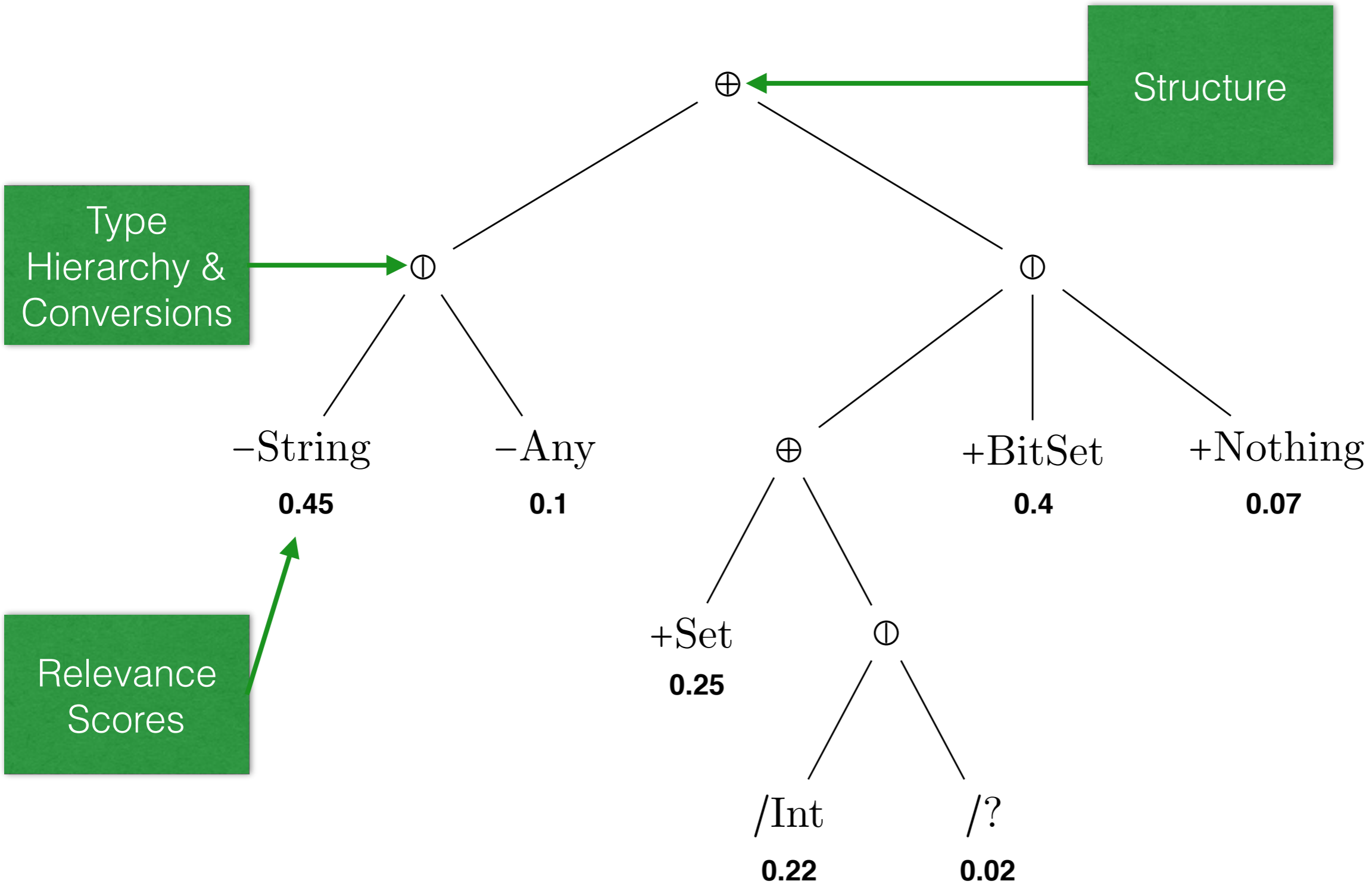
Query Expression Trees

- What terms are relevant to a query?
 - Also terms derived through subtyping/implicit conversions
 - E.g., if A extends B, fingerprints with -B should also be considered for a query A => _

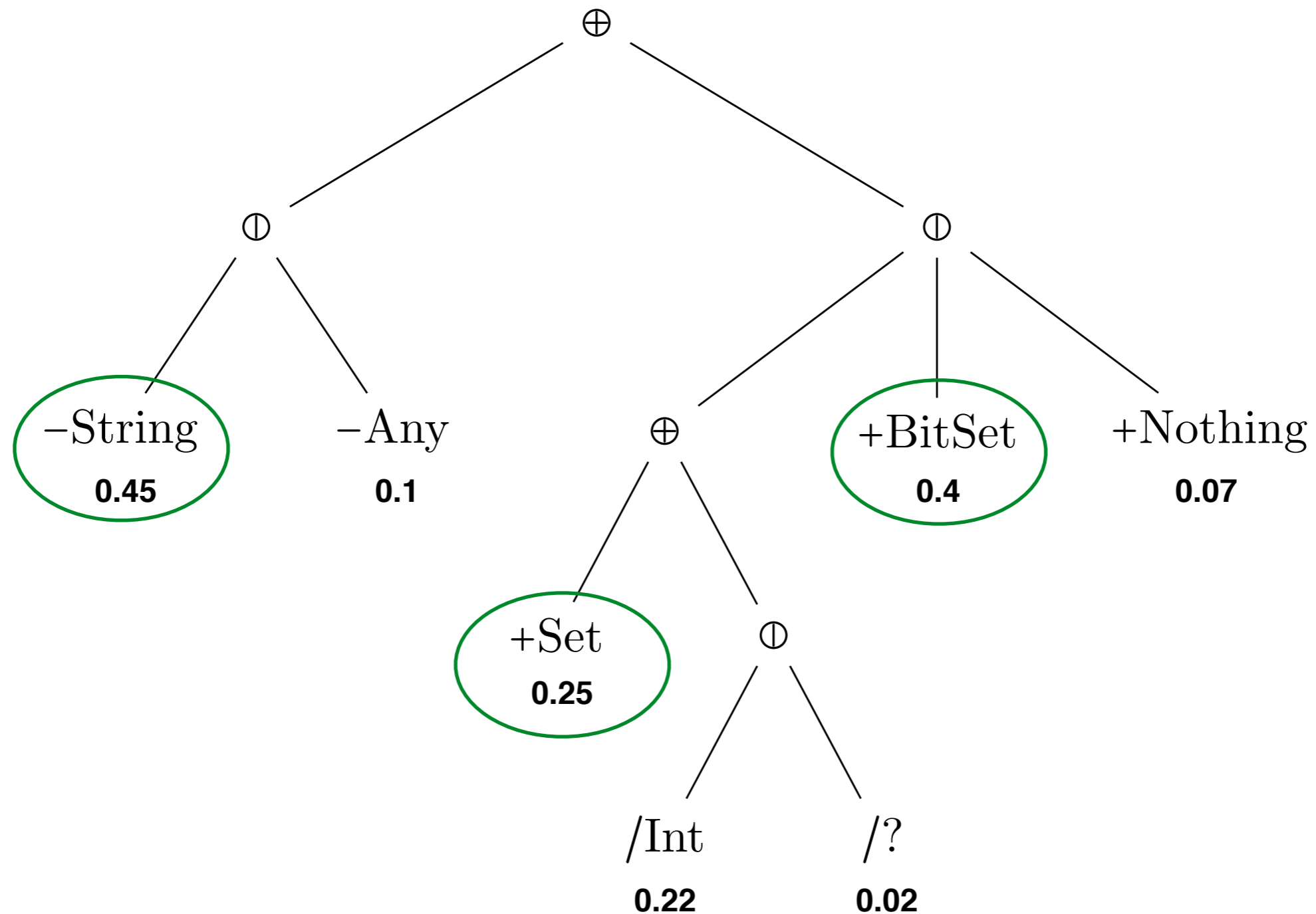
Query Expression Trees

- What terms are relevant to a query?
 - Also terms derived through subtyping/implicit conversions
 - E.g., if A extends B, fingerprints with -B should also be considered for a query A => _
- Similarity function for types
 - Just comparing term vectors does not capture structure of types
 - Evaluate QET derived from query with retrieved fingerprints

Query: `String => Set[Int]`

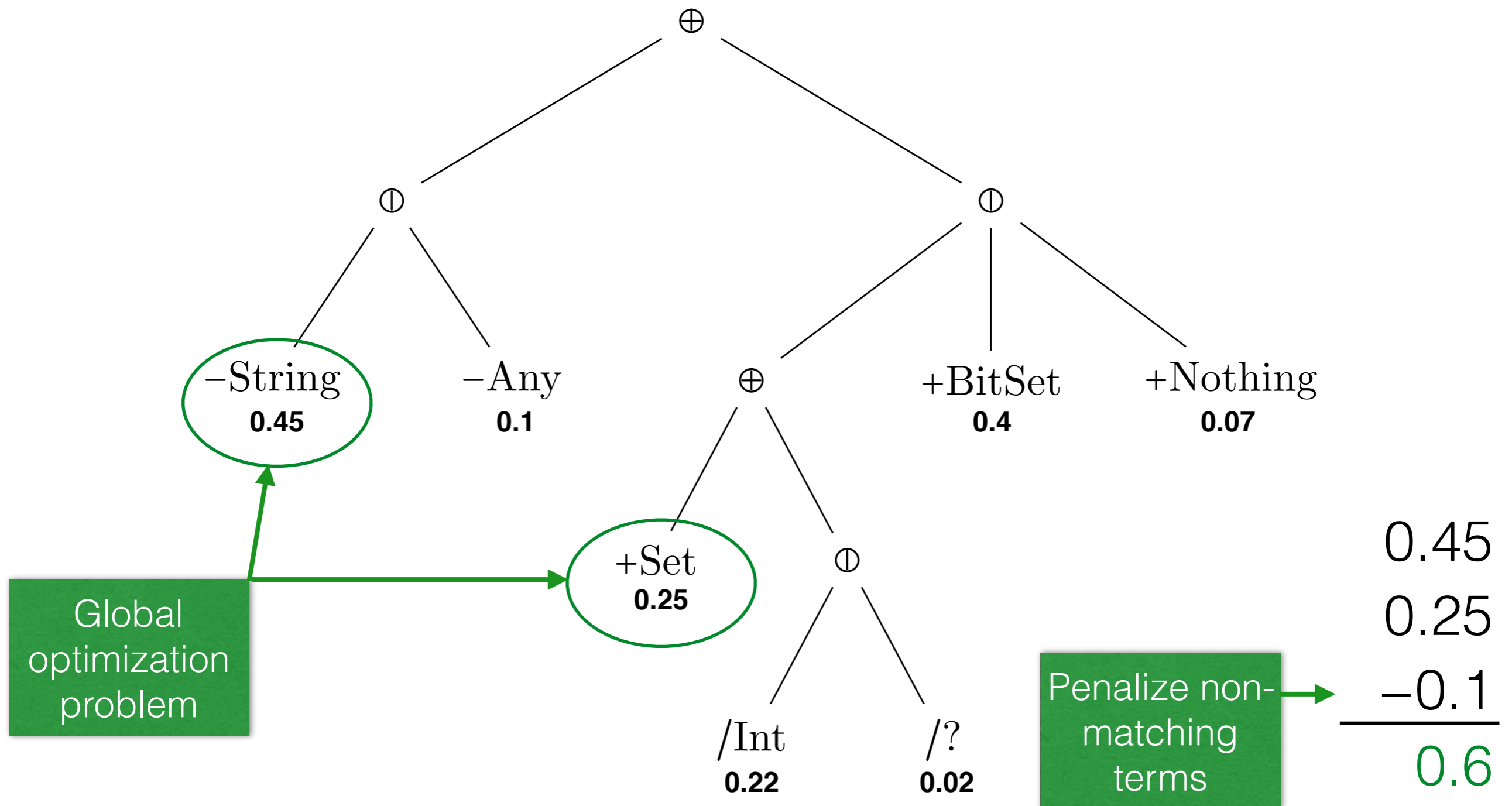


Fetch Fingerprints with **Dominant Terms**



Score Retrieved Fingerprints

E.g.: -String, -Any, +Set, /String



Evaluation: FEM

	AP I ₂	AP I ₃	AP I ₄
All Queries (Mean)	0.66	0.67	0.79
<code>remove: List[A] => A => List[A]</code>	1	1	0.33
<code>List[A] => (List[A], List[A])</code>	1	1	1
<code>List[A] => Option[A]</code>	0.58	1	1
<code>List[Future[A]] => Future[List[A]]</code>	0	0	1
<code>(List[Int], String) => String</code>	0.14	0.2	1

Tradeoff between keyword and type matching

Subtyping & Implicit conversions

Overly specific queries

Outline

- Why types for API search?
- Fingerprint Evaluation Model
 - Type Fingerprints
 - Query Expression Trees
- **Conclusion**

Conclusion

- The model is not yet ready for...
 - Type Classes (e.g. as used in Scalaz)
 - Structural Subtyping

Conclusion

- The model is not yet ready for...
 - Type Classes (e.g. as used in Scalaz)
 - Structural Subtyping
- Some performance issues when considering extremely large inheritance hierarchies
 - E.g.: `_ => TraversableOnce[TraversableOnce[_]]`

Conclusion

- The model is not yet ready for...
 - Type Classes (e.g. as used in Scalaz)
 - Structural Subtyping
- Some performance issues when considering extremely large inheritance hierarchies
 - E.g.: `_ => TraversableOnce[TraversableOnce[_]]`
- Depends on definition-side variance annotations

scala-search.org

The screenshot shows a web browser window with the title "Scaps: Scala API Search". The address bar contains the URL: `scala-search.org/?q=List[A]+%3D%3E+String+%3D%3E+String&m=org.scala-lang%3Ascala-library%3A2.11.7&m=org.scalaz%3Ascalaz-core_2.11%3A7.1`. The search bar contains the query `List[A] => String => String`. Below the search bar, there are four checkboxes for library versions: `scala-library:2.11.7` (checked), `scalajs-dom_sjs0.6_2.11:0.8.0` (unchecked), `scalajs-library_2.11:0.6.2` (unchecked), and `scalaz-core_2.11:7.1.1` (checked).

The first search result is for the method `List[A].mkString(String): String` with a score of 0.78178555. The description states: "Displays all elements of this list in a string using a separator string." The parameter `sep` is described as "the separator string." The return value is "a string representation of this list. In the resulting string the string representations (w.r.t. the method `toString`) of all elements of this list are separated by the string `sep`." An example shows `List(1, 2, 3).mkString("|") = "1|2|3"`. The source is `scala-library scala.collection.immutable.List.mkString`. A link "Doc · This is what i've been looking for" is present. Below the result is a button that says "17 more results matching `mkString: _ => _ => _`".

The second search result is for the method `scala.Console.readLine(String, Any*): String` with a score of 0.74239314. The description states: "scala.Console.readLine". The source is `scala-library scala.Console.readLine`. A link "Doc · This is what i've been looking for" is present. Below the result is a button that says "2 more results matching `readLine: _ => _* => _`".

Top Search Queries

1. flatmap

2. ->

3. max: Int

4. V

5. max: (Int, Int) => Int

6. |@|

7. traverseU

8. List => Int => Option

9. map

10. Ordering[String]

11. :::

12. List[A] => Int => Option[A]

13. (Int, Int) => Int

14. /:

15. <|*|>

16. ::

17. max: Int => Int => Int

18. Int => Int

19. List[T] => T

20. max

Thank **You!**

scala-search.org

github.com/scala-search/scaps

luegg.github.io