Language for Code Search

Luyao Ren
Wentao Wang

北京大学
PEKING UNIVERSITY

Midterm

Motivation

Find all the pieces of code contains method call named “close” after “open”.

```
...  open(...);   ...  close(...);  ...
```

A Simple Solution

Use regular expression:

```
{[\s\S]*open[\s\S]*close[\s\S]*}
```

Is it good enough?

Motivation

Find all the pieces of code contains method call named “close” after “open”, and both “open” and “close” contains the same **variable**.

```
...  open($var);  ...  close($var); …
```

Motivation

Syntax-based Matching for Code Search!

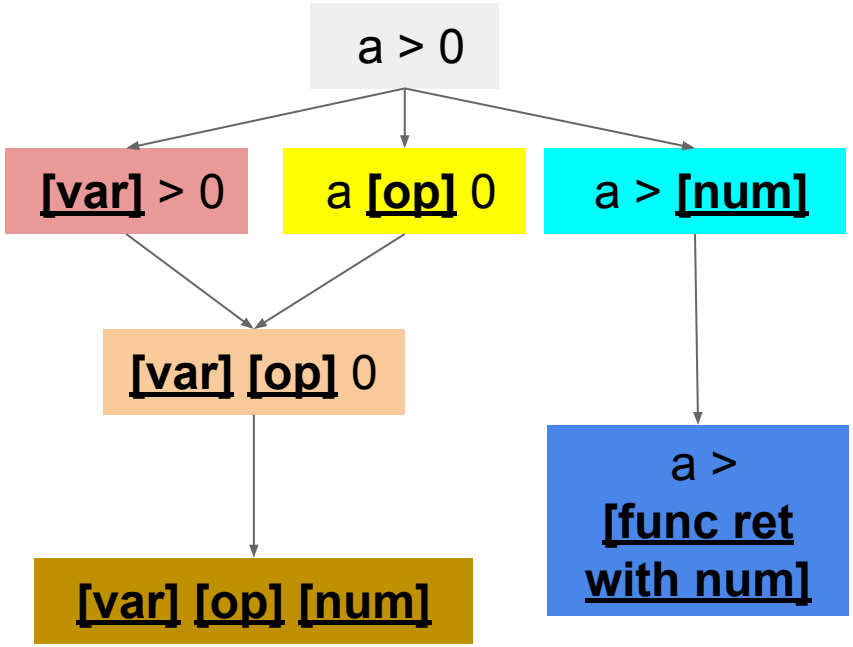
To do this, we need first have a **language** to let user define what they want.

Abstraction - Quick Example

Consider following code

(or expression):

```
a > 0
```



Goal

Flexibility: support different levels of **abstraction**

Easy use: pattern could be extracted from existing piece of code, support **iterative** query

Scalability: **multi-language** support

Final Presentation

Language for Code Search (L4CS)

Luyao Ren
Wentao Wang

北京大学
PEKING UNIVERSITY

Goal

Design a language to define **patterns** for

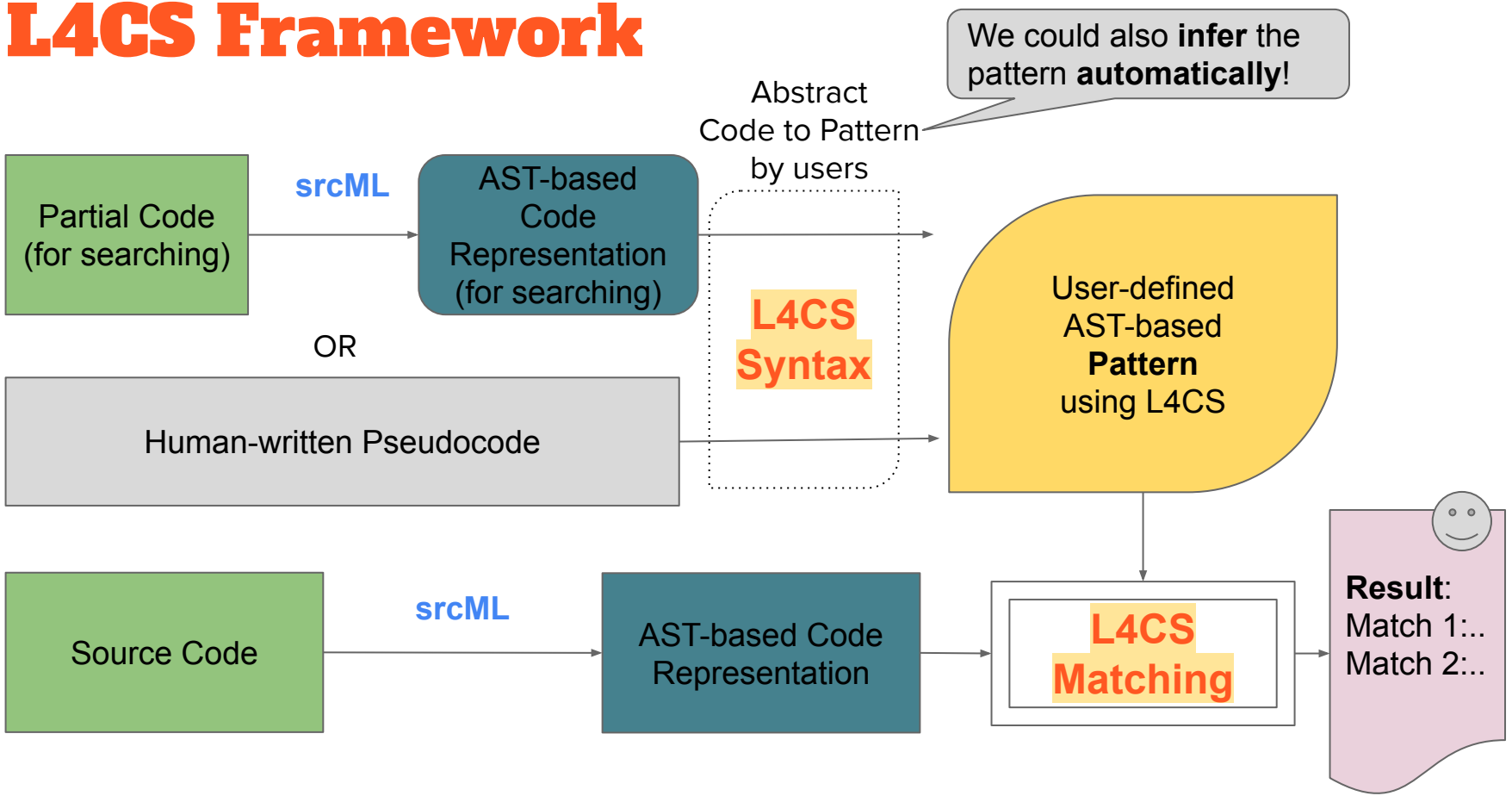
syntax-based code search with **abstraction**

A Quick Example

Find all the pieces of code contains method call named “close” after “open”.

```
...   open($var);  ...   close($var);    ...
```

L4CS Framework



L4CS Syntax

XML Format

Code element is based on srcML Grammar

Context-free Grammar

code ::= code1 code2

<AnyElement> code' </AnyElement>

text

<abs> </abs>

<fold>** code' **</fold>

variable

OR operator: ***v1 | v2 | v3 ...***

variable ::= *<udv> v1 </udv> <udv> v2 </udv> <udv> v3 </udv> ...*

<v-$varName>** code **</v-$varName>

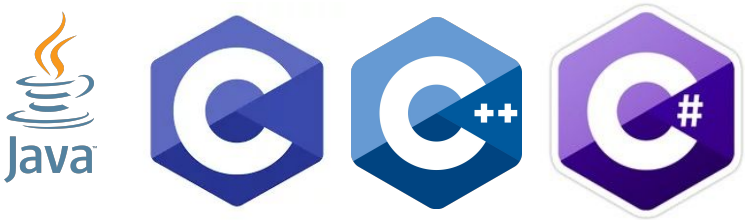
L4CS Matching

Top-Down matching

Greedy algorithm for <abs>

Supporting with OR operator, <folding>, ...

Feature: Multi-language Support



↓

srcML

↓

Unified Code Representation

↓

L4CS

Feature: Folding

```
<v-hasf1>
    <call>
        <name>f</name>
        <argument_list>()</argument_list>
    </call>
</v-hasf1>
```

f()

```
<v-hasf2>
    <call>
        <name>
            <abs></abs>
            <name>f</name>
        </name>
        <argument_list>()</argument_list>
    </call>
</v-hasf2>
```

a.f()

...

a.b.f(), a.b.c.f(), ...

```
<call>
    <fold>
        <name>f</name>
    </fold>
    <argument_list>()</argument_list>
</call>
```

{Anything}.f()

Folding!

Feature: Binding

Motivation:

Find all the pieces of code contains method call named “close” after “open”, and both “open” and “close” contains the same **variable**.

```
...  open($var);  ...  close($var); …
```

Feature: Binding

User A:
This two $var be the **same**!
User B:
This two $var needn't be the **same**!

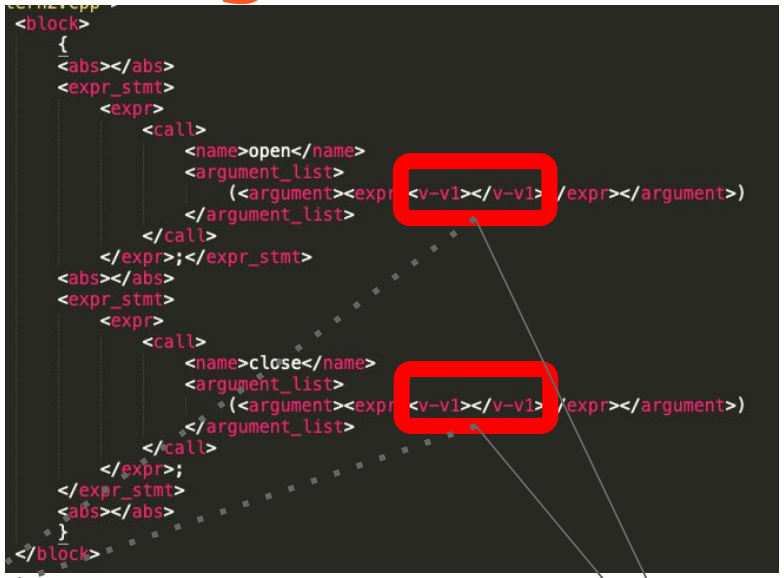
Pattern:

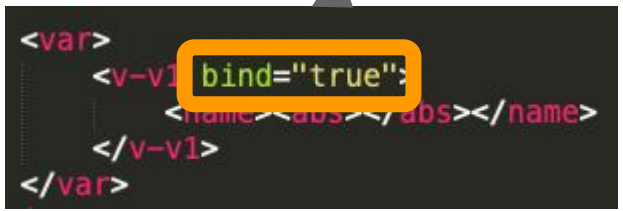
```
... open($var); ... close($var); …
```

Code:

	User A	User B
`... open(tmp); ... close(tmp); …`		
`... open(tmp); ... close(tmp2); …`		

Feature: Binding





User A

User B

Feature: Interactive Query

```
-----Language For Code Search-----
setpath /Users/luyaoren/LanguageForCodeSearch/resources/test/4
Current path:/Users/luyaoren/LanguageForCodeSearch/resources/test/4
add p5.xml
Add pattern successfully!
search src.xml
----Pattern----
 [ABS] ( [ABS] f [ABS] | [ABS] [ABS] f [ABS] [ABS] ) [ABS]
----Match Code----
Match #1
a.f() + a.g() + a.q()
Match #2
b.f() + b.g() + b.w()
Match #3
b.getA().f()
----Finish----
```

Expression contains “f”

```
clear
Clear all patterns successfully!
add p6.xml
Add pattern successfully!
search src.xml
----Pattern----
 [ABS] [ABS] b [ABS] [ABS] [ABS]
----Match Code----
Match #1
b.f() + b.g() + b.w()
Match #2
b.getA().f()
Match #3
b.getA().g() + b.getA().num
----Finish----
```

Expression contains “b”

```
clear
Clear all patterns successfully!
add p5.xml
Add pattern successfully!
add p6.xml
Add pattern successfully!
search src.xml
----Pattern----
 [ABS] ( [ABS] f [ABS] | [ABS] [ABS] f [ABS] [ABS] ) [ABS]
 [ABS] [ABS] b [ABS] [ABS] [ABS]
----Match Code----
Match #1
b.f() + b.g() + b.w()
Match #2
b.getA().f()
----Finish----
```

Expression contains both “b” and “f”

Demo

Division of Labour

Idea: Luyao Ren

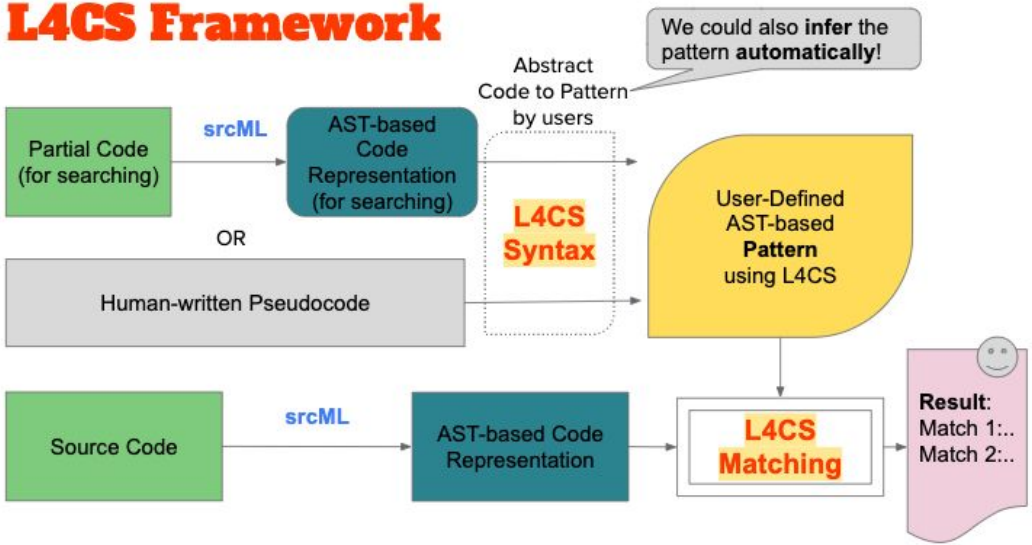
Design: Luyao Ren & Wentao Wang

Implement: Luyao Ren

Report: Wentao Wang

L4CS

L4CS Framework



L4CS Syntax

XML Format

Code element is based on srcML Grammar

Context-free grammar

code = code1 code2

<AnyElement> code' </AnyElement>

<fold>** code' **<fold>

<abs> </abs>

variable

variable = ***variable1 | variable2 | variable3 ...***

<v-$varName>** code (without variable) **</v-$varName>

Key Features:

Multi-language Support

Folding

Binding

Interactive Query