Finding Code-Clone Snippets in Large Source-Code Collection by **`ccgrep`**

Katsuro Inoue[1], Yuya Miyamoto[1],
Daniel M. German[2], and Takashi Ishio[3]

1: Osaka University
2: University of Victoria
3: Nara Institute of Science and Technology

Ternary Operator in Linux USB Driver File

linux/drivers/usb/gadget/function/u_ether.c

```
        ...
{
        int ret;

        rtnl_lock();
        ret = snprintf(name, len, "%s¥n",
                       netdev_name(net));
        rtnl_unlock();
        return ret < len ? ret : len;
}
        ...
```

Returns the minimum of two.
Should be changed to
`min(ret, len)`

Using `grep`

```
ret < len ? ret : len;
```

```
$ grep '<' -r .
./misc/chaoskey.c: * Copyright © 2015 Keith Packard <keithp@keithp.com>
./misc/chaoskey.c:#include <linux/module.h>
./misc/chaoskey.c:#include <linux/slab.h>
...
```

16,335 matches in linux/drivers/usb/*

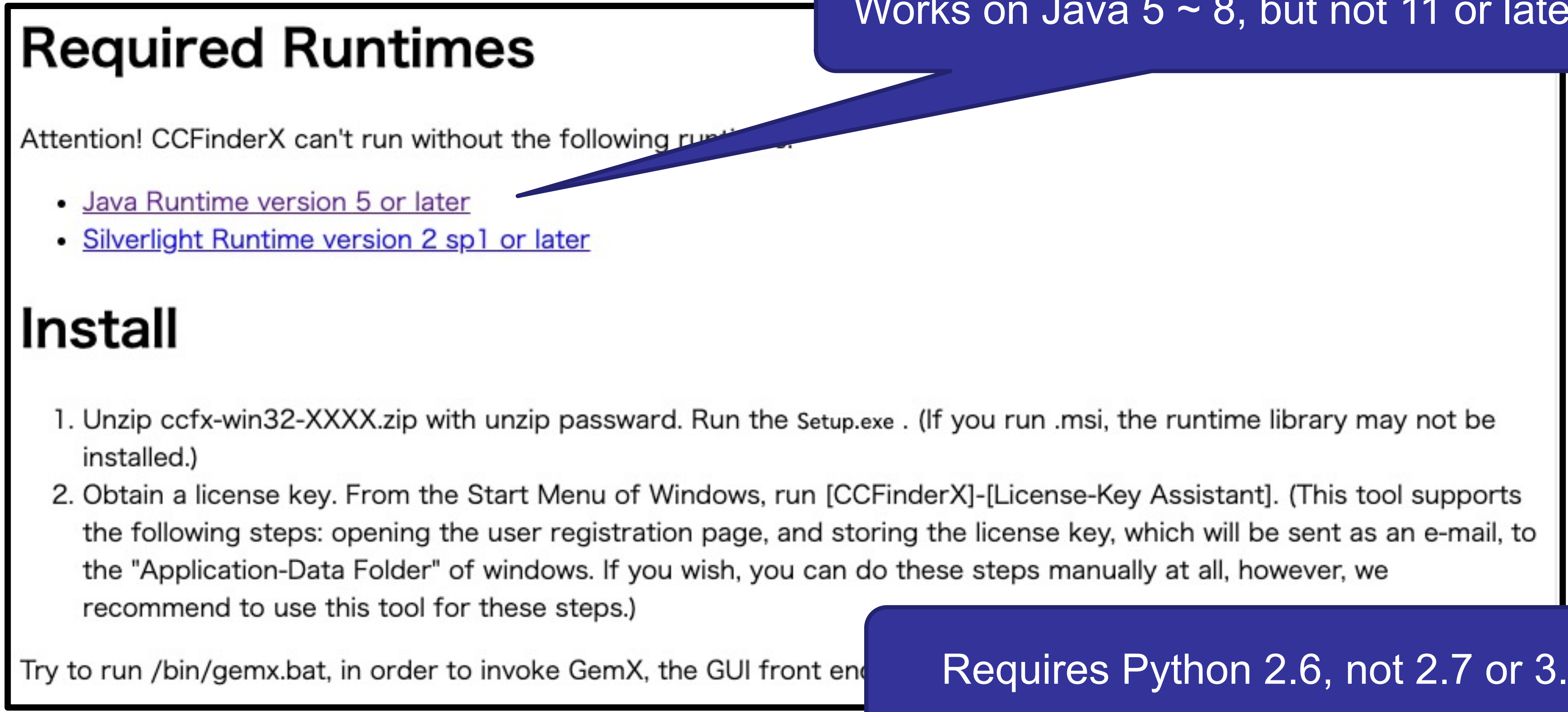
```
$ grep '<' -r . | grep '?'| grep ':'
./host/oxu210hp.h:#define QTD_STS_PING    (1 << 0)  /* issue PING? */
./misc/usbtest.c:      ? (INTERRUPT_RATE << 3)
./misc/usbtest.c:      return (retval < 0) ? retval : -EDOM;
...
```

149 matches in linux/drivers/usb/*

Using Clone Detector

```
ret < len ? ret : len;
```

Installing **`CCFinderX`**



Required Runtimes

Attention! CCFinderX can't run without the following ru

- Java Runtime version 5 or later
- Silverlight Runtime version 2 sp1 or later

Install

1. Unzip ccfx-win32-XXXX.zip with unzip passward. Run the `Setup.exe` . (If you run .msi, the runtime library may not be installed.)
2. Obtain a license key. From the Start Menu of Windows, run [CCFinderX]-[License-Key Assistant]. (This tool supports the following steps: opening the user registration page, and storing the license key, which will be sent as an e-mail, to the "Application-Data Folder" of windows. If you wish, you can do these steps manually at all, however, we recommend to use this tool for these steps.)

Try to run /bin/gemx.bat, in order to invoke GemX, the GUI front en

Works on Java 5 ~ 8, but not 11 or later

Requires Python 2.6, not 2.7 or 3.X

Simple Tool

```
$ tool -r .
   ret < len ? ret : len;
```

Copy and Paste Snippet

```
./misc/adutux.c:
    int amount =
         bytes_to_read < data_in_secondary ?
              bytes_to_read :
              data_in_secondary;
./storage/realtek_cr.c:
    residue=residue < buf_len ? residue:buf_len;
./gadget/function/u_ether.c:
    return ret < len ? ret : len;
```

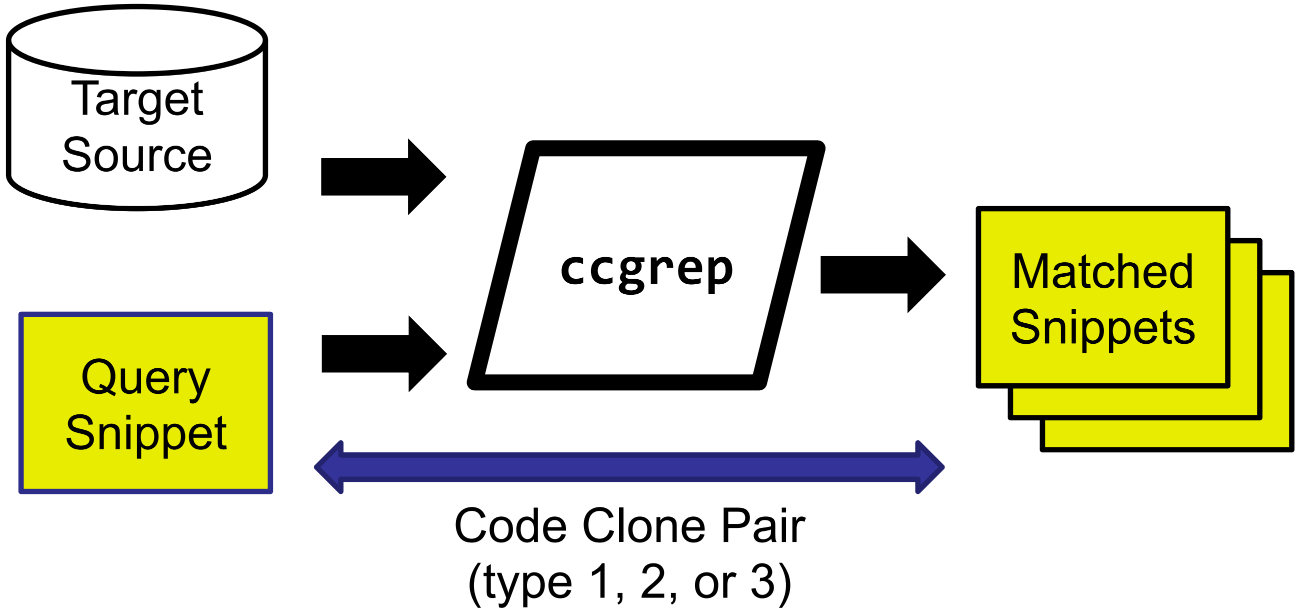
Query Seed Snippet

Motivation

- Find similar code snippets
    - Similar bug finding, code refactoring, ...
- `grep` is too primitive
- Code clone detector generally
    - Requires complex installation and environment setting
    - Generates excessive output
- Simple tool to find similar code snippets

`ccgrep` (Code Clone grep)

- Use notion of code clone for matching
- Query and matched snippet forms a code clone pair of type 1, 2, or 3



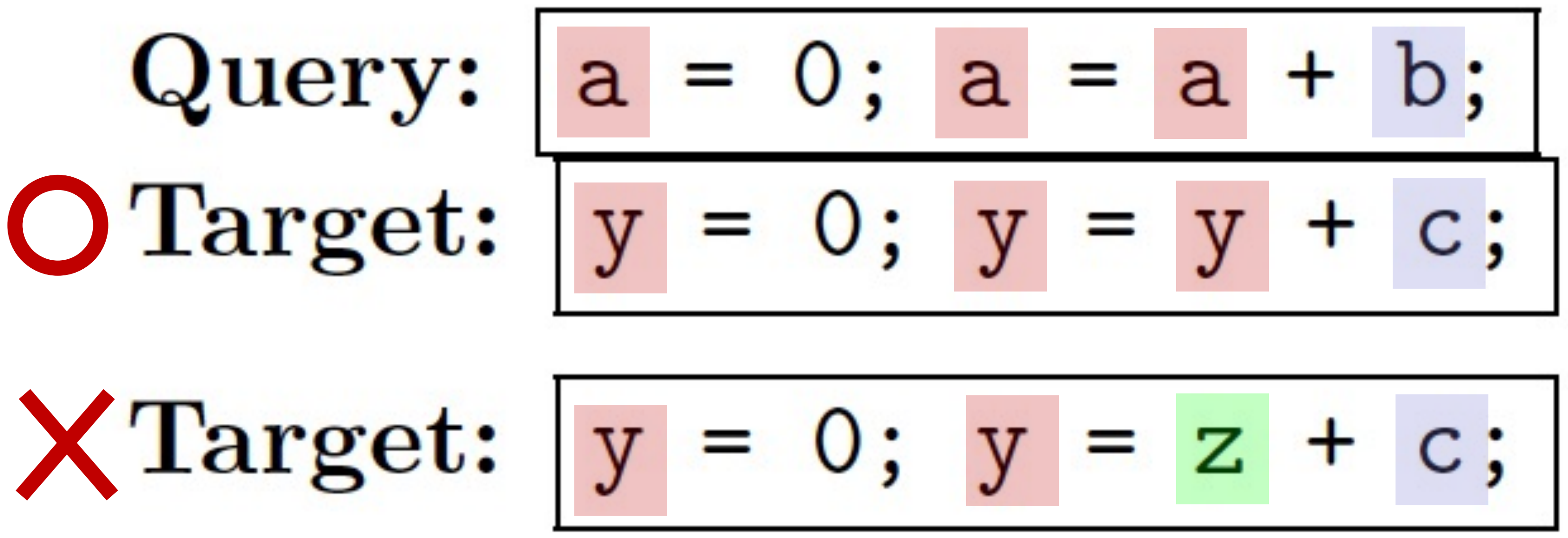
Type Classification of Code Clone Pairs

Two code snippets with differences of

- Type 1: spaces, line breaks, or comments
- Type 2: replacement of identifiers or literals (+ type 1)
- Type 3: some statement additions, deletions, or changes (+ type 2)
- Type 4: syntax but semantics are equivalent

Type 2 Clone Match

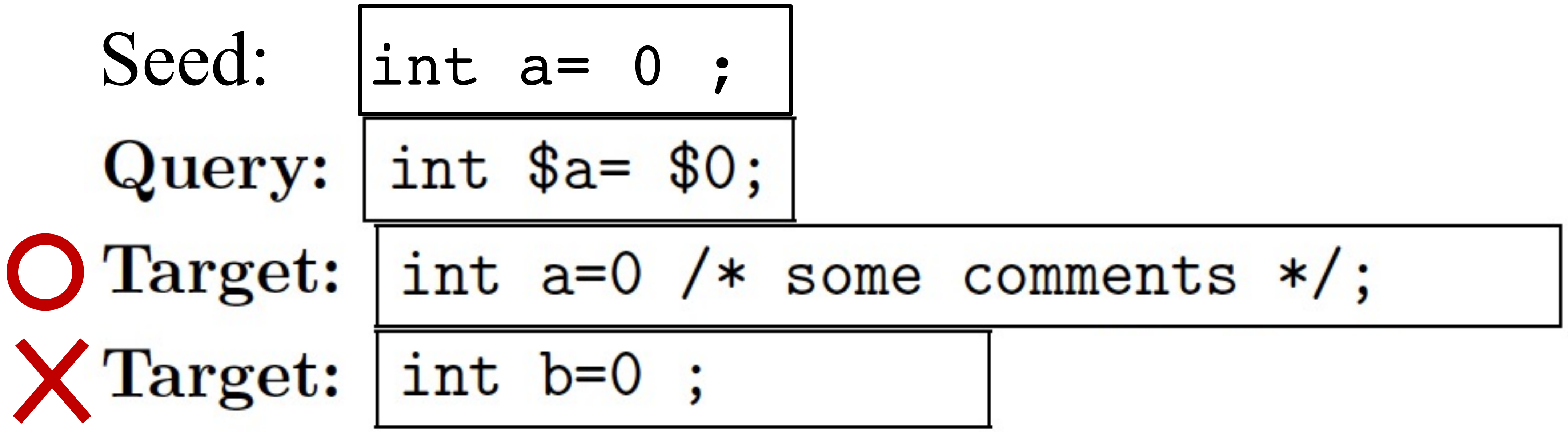
- Basis and default of **`ccgrep`**
- Allows differences of identifiers or literals
- P-(parameterized)match (consistent mapping)



Query:
```
a = 0; a = a + b;
```
○ Target:
```
y = 0; y = y + c;
```
✕ Target:
```
y = 0; y = z + c;
```

Type 1 Matching

- Same snippets with different spaces, line-breaks, comments
- Give an option (-b none) or fix identifier/literal in the query with meta symbol $



Seed:
```
int a= 0 ;
```
Query:
```
int $a= $0;
```
○ Target:
```
int a=0 /* some comments */;
```
✗ Target:
```
int b=0 ;
```

Type 3 Matching

- Allows statement additions, deletions, or changes
- Employ meta tokens $. and $$

Seed: `a = 5 ;`

Query: `a = $. ;`

○ Target: `a = b ;`

$. Any single token

Seed: `a = 5 ;`

Query: `a = $$ ;`

○ Target: `a = b+c+10 ;`

○ Target: `a = f(g,h) ;`

$$ Any token sequence

Matching Various Code Snippets

Method XYZ with no parameter

Query: `$XYZ( )`

Method XYZ with 0 or more parameters

Query: `$XYZ($$)`

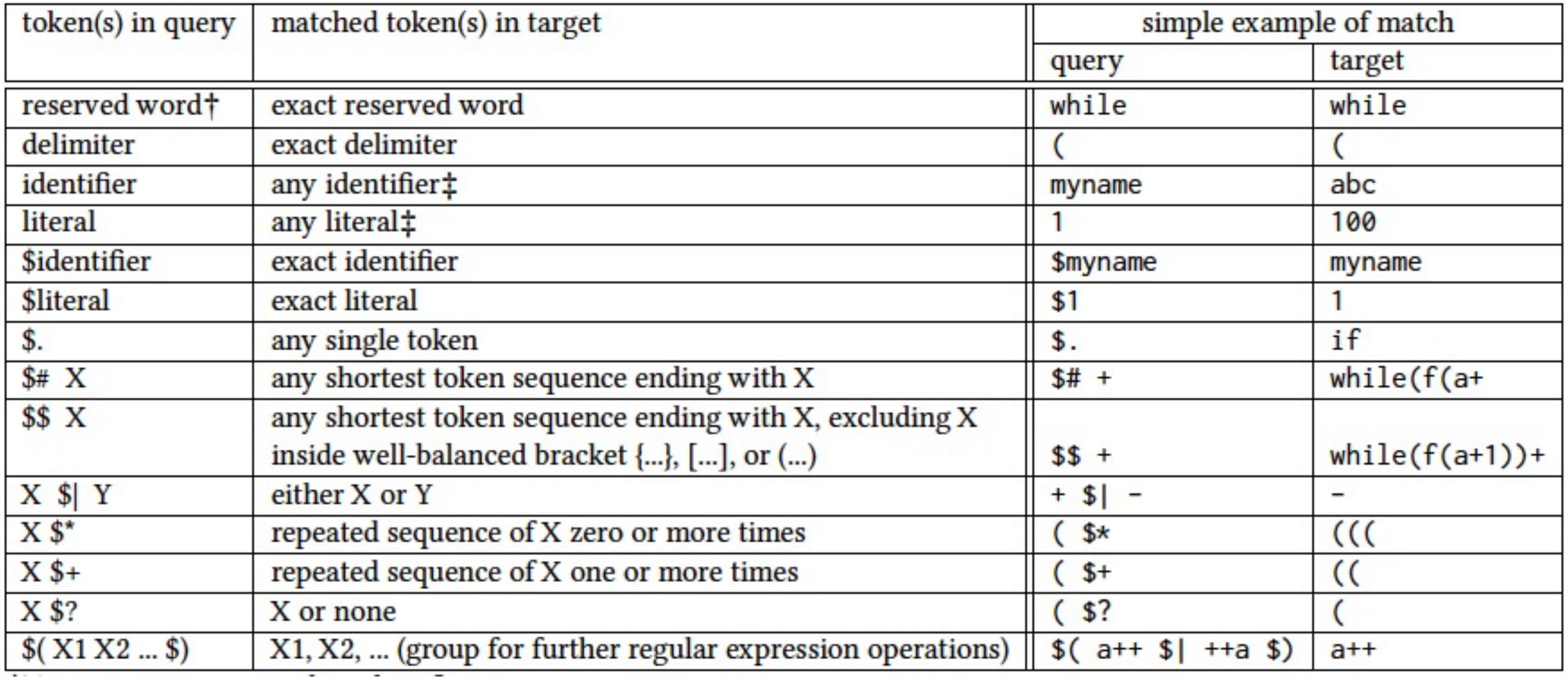
if **statement**

Query: `if ($$){$$}`

for **statement using control variable**

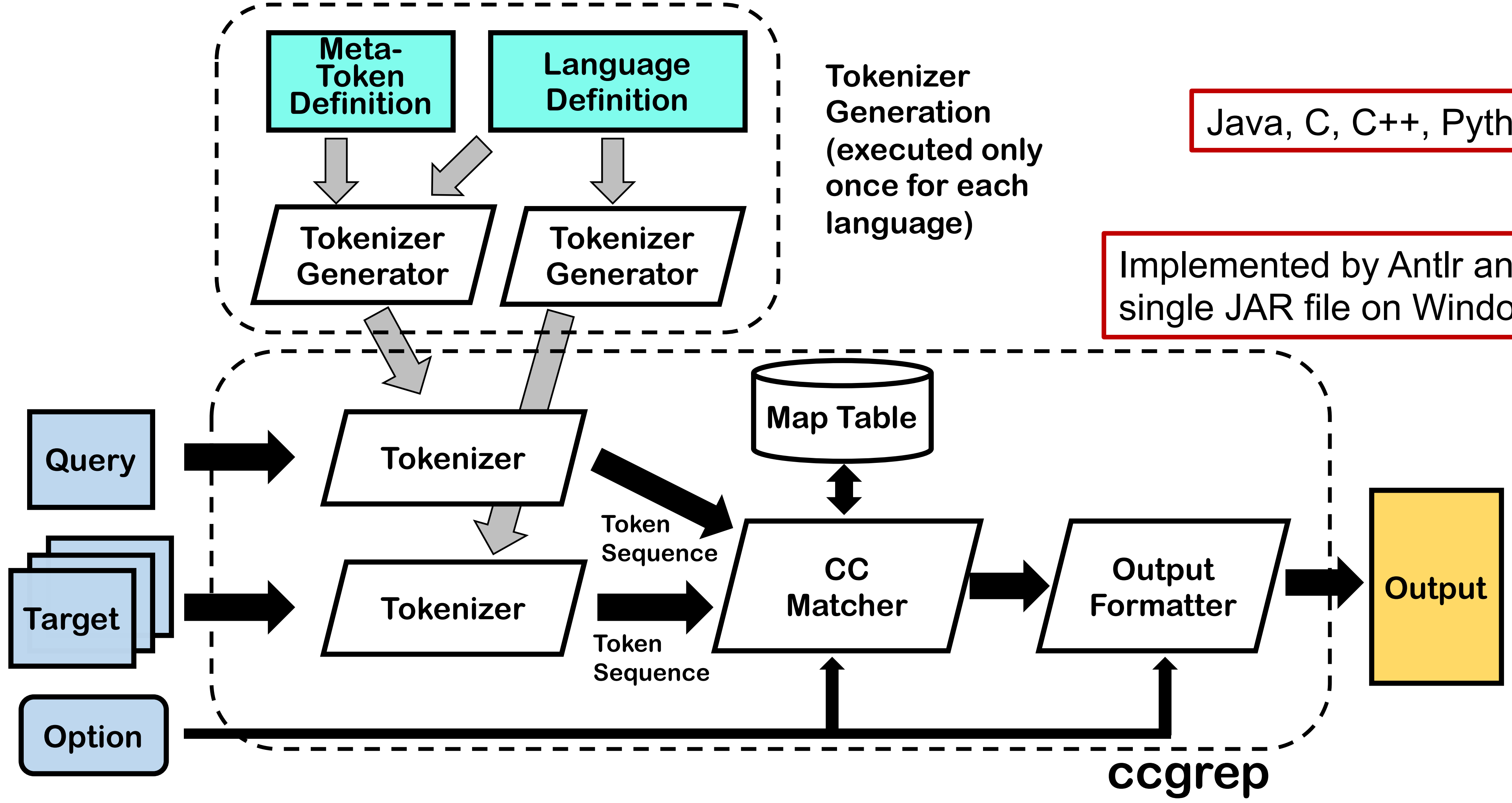
Query: `for(T i=0; i<$$; i++){$$}`

Formal Definition of Matching(Appendix Table 3)



token(s) in query	matched token(s) in target	simple example of match	
		query	target
reserved word†	exact reserved word	`while`	`while`
delimiter	exact delimiter	`(`	`(`
identifier	any identifier‡	`myname`	`abc`
literal	any literal‡	`1`	`100`
$identifier	exact identifier	`$myname`	`myname`
$literal	exact literal	`$1`	`1`
$.	any single token	`$.`	`if`
$# X	any shortest token sequence ending with X	`$# +`	`while(f(a+`
$$ X	any shortest token sequence ending with X, excluding X inside well-balanced bracket {...}, [...], or (...)	`$$ +`	`while(f(a+1))+`
X $| Y	either X or Y	`+ $| -`	`-`
X $*	repeated sequence of X zero or more times	`( $*`	`(((`
X $+	repeated sequence of X one or more times	`( $+`	`((`
X $?	X or none	`( $?`	`(`
$(X1 X2 ... $)	X1, X2, ... (group for further regular expression operations)	`$( a++ $| ++a $)`	`a++`

Architecture of **ccgrep**



Java, C, C++, Python

Implemented by Antlr and Java single JAR file on Windows or Unix

Evaluation of `ccgrep`

- RQ1: Query Expressiveness
  Are various kinds of code clones expressed by the queries?
- RQ2: Accuracy
  Does it accurately find clones already detected by other approaches?
- RQ3 : Performance
  What is the execution time?

RQ1: Query Expressiveness

Query for

- Type 1 clone: seed snippet plus an option or $id or $literal
- Type 2 clone: seed snippet (P-match as default)

  (Non P-match → by option)
- Type 3 clone: seed snippet with meta tokens ($., $$, or $#) at addition, deletion, or change points

All queries for type 1, 2, and 3 clones are effectively expressed

RQ2: Accuracy

- Type 1 and 2: 53K clones in BigCloneBench
  - Made one snippet of each clone pair as query

  Successfully found another snippet in each case

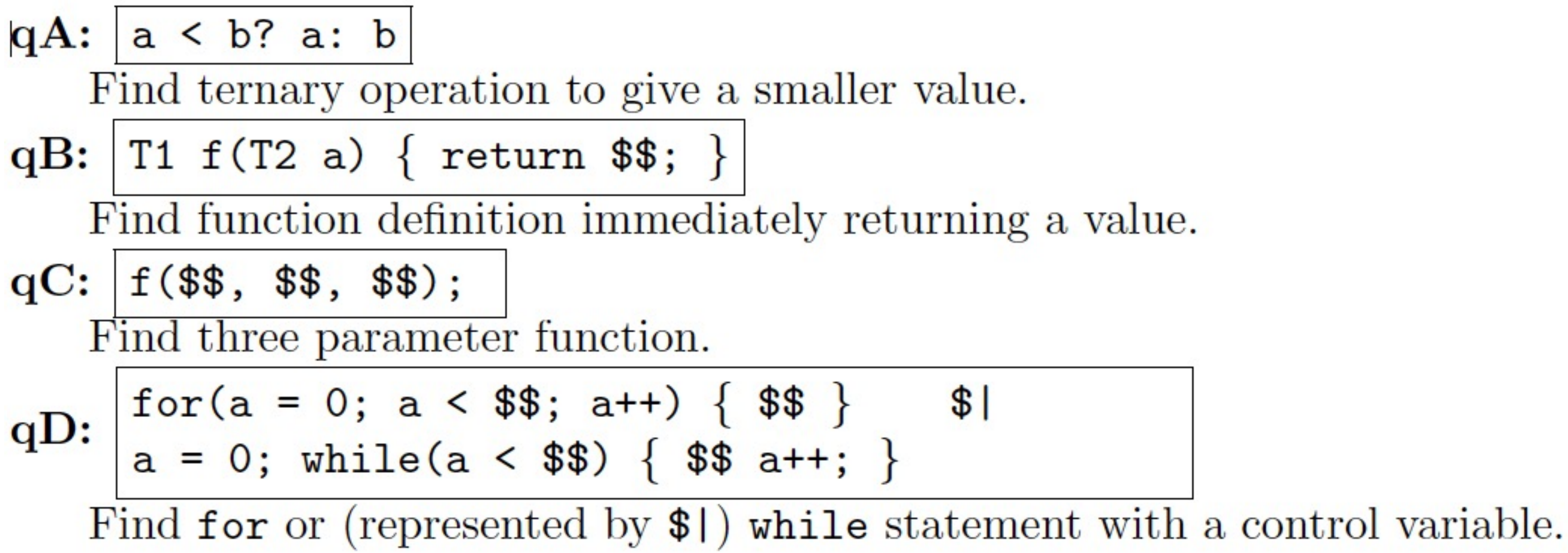
- Finding 11 type 3 clones for CBCD data[16]
  - Crafted the query from one snippet in each clone pair to match another snippet

  Successfully found another snippet in each case

[16] Li, J., Ernst, M.D.: Cbcd: Cloned buggy code detector. In: 2012 34th International Conference on Software Engineering (ICSE). pp. 310{320 (June 2012).

RQ3: Performance

Quires



qA: `a < b? a: b`

Find ternary operation to give a smaller value.

qB: `T1 f(T2 a) { return $$; }`

Find function definition immediately returning a value.

qC: `f($$, $$, $$);`

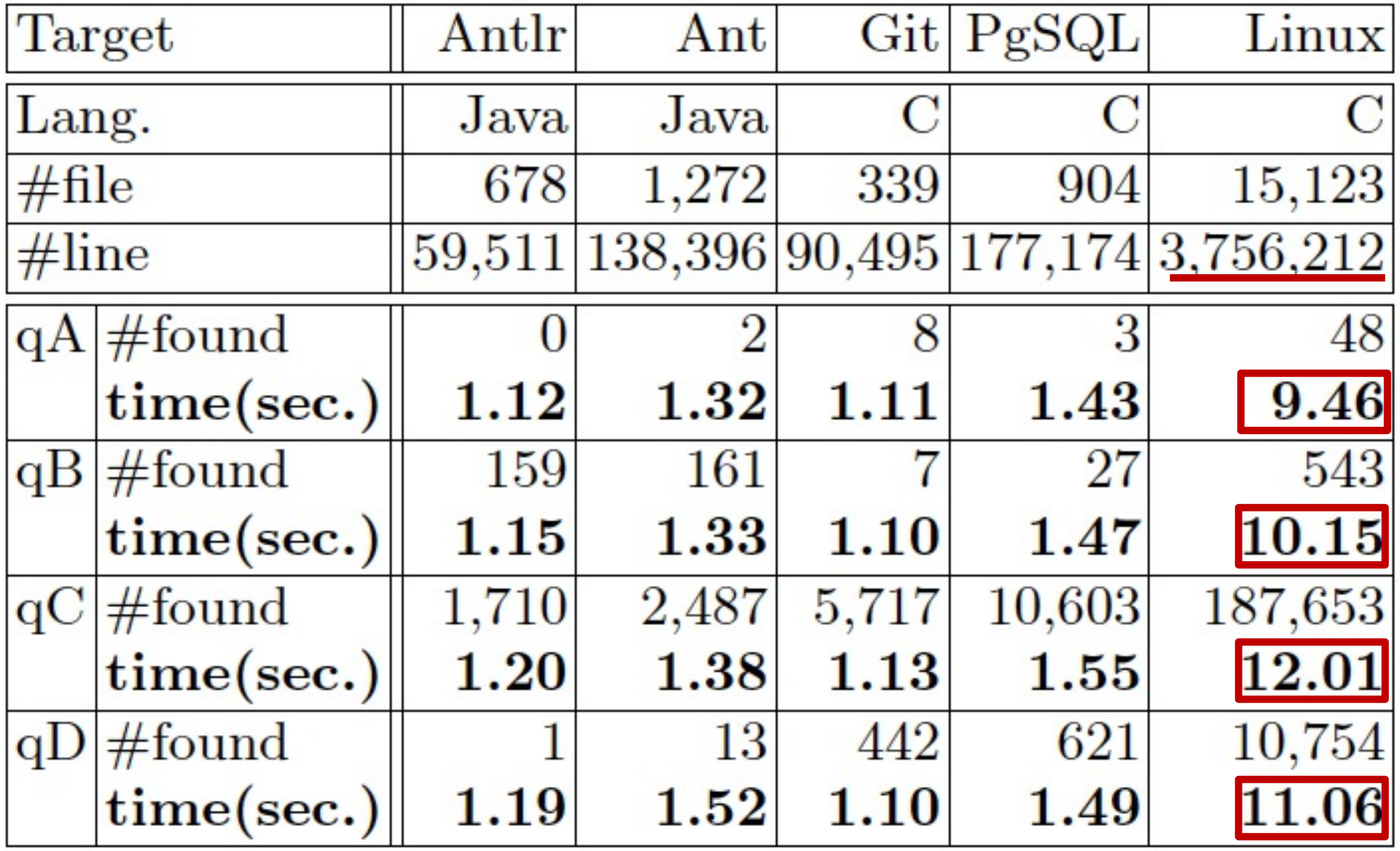
Find three parameter function.

qD:

```
for(a = 0; a < $$; a++) { $$ }     $|
a = 0; while(a < $$) { $$ a++; }
```

Find `for` or (represented by `$|`) `while` statement with a control variable.

Result of RQ3



Target		Antlr	Ant	Git	PgSQL	Linux
Lang.		Java	Java	C	C	C
#file		678	1,272	339	904	15,123
#line		59,511	138,396	90,495	177,174	3,756,212
qA	#found	0	2	8	3	48
	time(sec.)	**1.12**	**1.32**	**1.11**	**1.43**	**9.46**
qB	#found	159	161	7	27	543
	time(sec.)	**1.15**	**1.33**	**1.10**	**1.47**	**10.15**
qC	#found	1,710	2,487	5,717	10,603	187,653
	time(sec.)	**1.20**	**1.38**	**1.13**	**1.55**	**12.01**
qD	#found	1	13	442	621	10,754
	time(sec.)	**1.19**	**1.52**	**1.10**	**1.49**	**11.06**

Sufficiently fast and acceptable as a search tool even for large targets

grep
generally 3~9 times faster, but it misses multiple-line results

Intel Xeon E5-1603v4(@2.8GHz 4), 32GB RAM, and Windows 10 Pro for WS 64bit

Related works

- grep-like tools
  - **`agrep`**, **`cgrep`**, **`sgrep`**, ... extend grep's feature
  - **`coccigrep`** is data-structure based matching

  No idea of clone-based matching
- Pattern matching tools
  - **`CBCD`** (PDG based matching to find buggy code)
  - **`NCDSearch`** (similarity by compression distance)
  - **`Siamese`** (index of methods or files)

  No precise control of the matching

Conclusion

- Proposed clone-based pattern matching tool **`ccgrep`**
  - Effectively finds type 1, 2, and 3 clone snippets

- **`ccgrep`** is OSS on GitHub
  https://github.com/yuy-m/CCGrep

Future direction

- Improve the matching algorithm
- Add a small GUI to edit query from the seed snippet

Thank you!

Question?

- -b,--blind <LEVEL> set blind level.
  - none(Type 1) / consistent(p-match)(by default) / full(Type 2).
- -e <PATTERN> use PATTERN for matching.
- --exclude <FILE_PATTERN> skip files matching FILE_PATTERN.
- -f,--file <FILES> obtain query from file.
  - CANNOT give query as code string at once.
- --fix <ID> specify identifier to match exactly the same one.
- -h,--help show help.
- ignore-extension search all files ignoring file extensions.
- --include <FILE_PATTERN> search ONLY files that match FILE_PATTERN.
- --json print clones and execution information with JSON format.
- -l,--language <LANG> set target language.
  - c / c++ / java(by default) / python3.
  - With -f option, the language can be inferred from the file extension.
- -m,--max-count <NUM> stop after NUM clones.
- --no-messages suppress error messages.
- --no-overlap search without overlap.

- -p,--print <OPTION> set printing option c/l/n/f/e like -p fn.
    - If c is given, print the count of clones file by file.
    - If C is given, print ONLY the count of all clones.
    - If l is given, print ONLY file name per matched files.
    - If h is given, NOT print file names.
    - If n is given, print line numbers.
    - If N is given, print pairs of start and end line number.
    - If f is given, print whole code of clones.
    - If r is given, print whole code of clones in one line.
    - If o is given, print only the matched parts of a clone code.
    - If e is given, comment out the file name and line numbers.
- --parallel search each file in parallel.
- -r,--recursive traverse directories recursively.
- -s,--stdin-query obtain query from standard input. CANNOT give query as code string at once.
- -x,--file-match force QUERY to match only whole file.
- --xml print clones with XML format.