interactive code checking with Cobra

a Tutorial

Gerard Holzmann
Nimble Research
gholzmann@acm.org

this tutorial

Code Browser and Analyzer

- how does it work, and
- how can you use it?

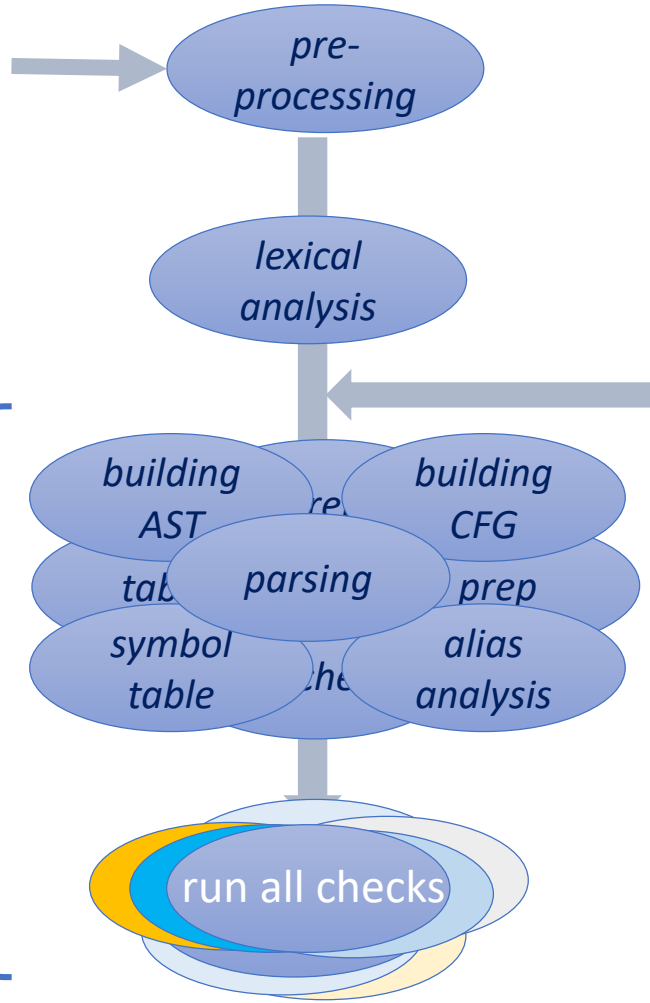
topics covered

1. *background* and principle of operation
   - installation and configuration
   - guide to online documentation
2. *pattern* queries and regular expressions
   - exercises
3. *interactive* queries
   - token attributes
   - sets and ranges
   - functions
   - reading files, libraries
   - exercises
4. *scripted* queries
   - recursive functions
   - associative arrays
   - the query libraries
   - using concurrency
   - exercises
5. *standalone* checkers
   - using concurrency: multi-threaded checkers
6. use of Cobra for *runtime verification*
   - using live data or event-logs

why does traditional static analysis take so long?

building data structures

thousands or millions of lines of source code



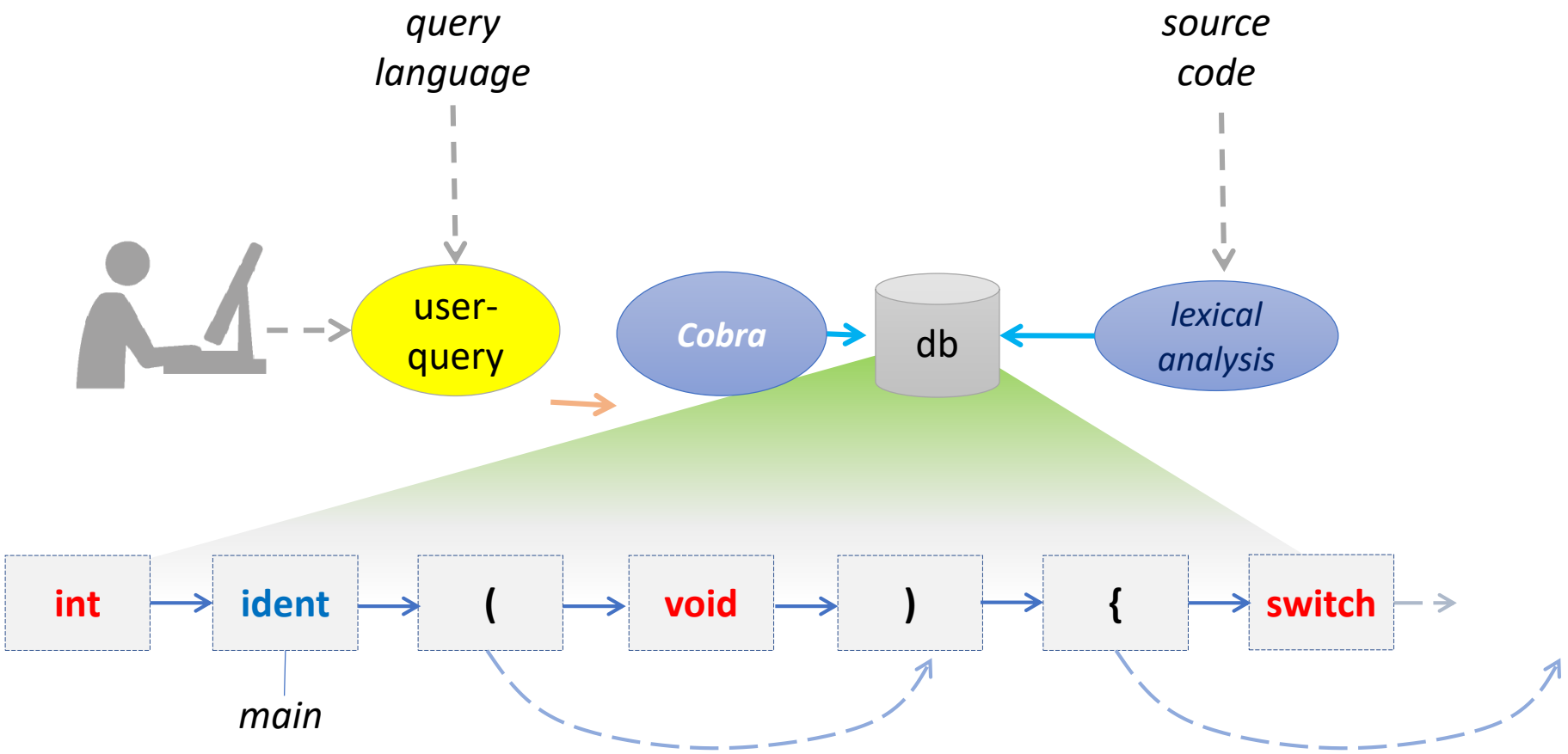
most of the time is spent here for a large code-base this can take *hours*

but, a large number of checks require no more information than is already available here

more detailed information may be derived as needed, *if* needed

cobra's design

minimize prep-time *and* query time



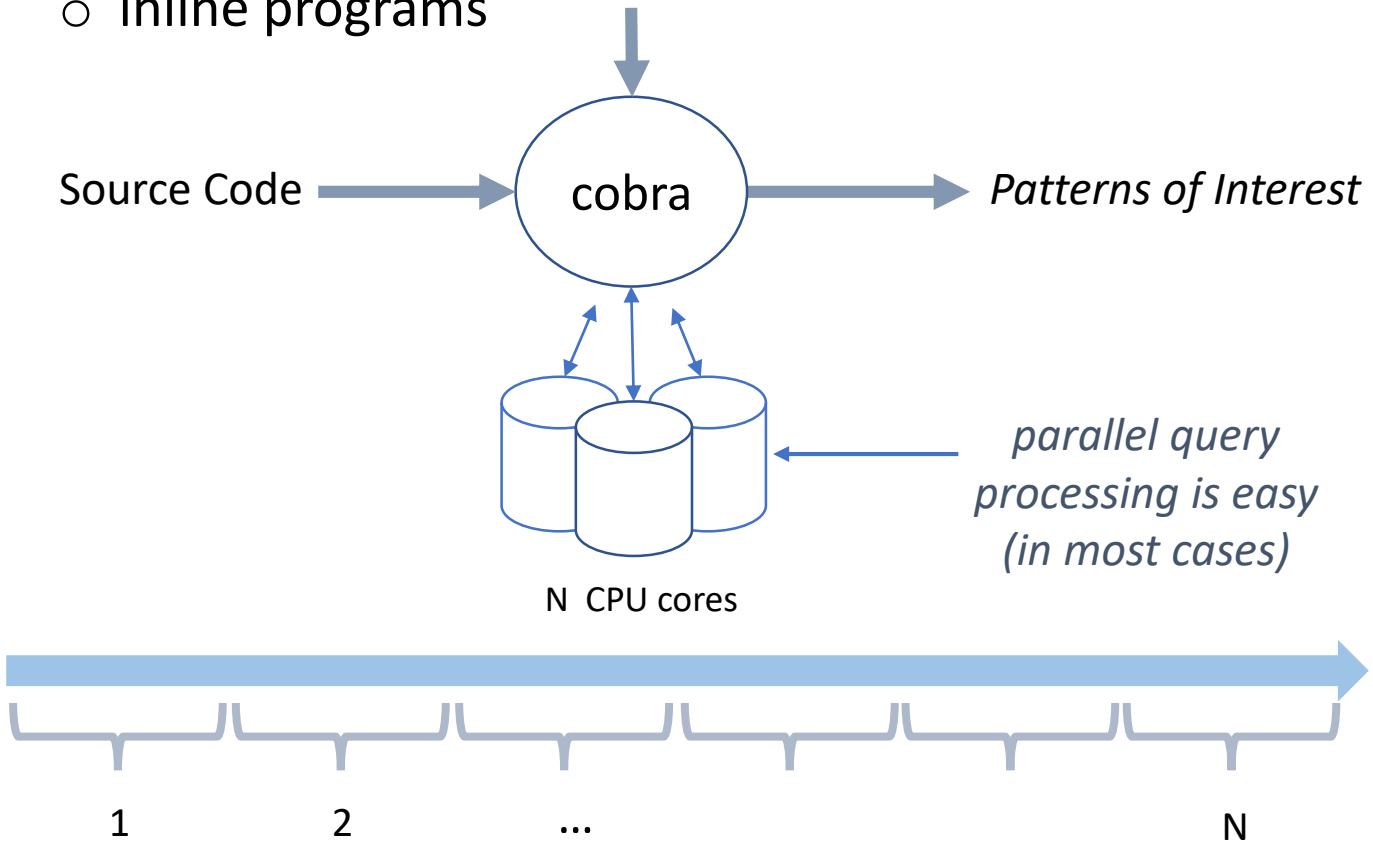
a linked list of lexical tokens with annotations

(token types, ranges, levels of nesting for parentheses, brackets, and braces, etc.)

cobra's design

minimizing query response time

- interactive query commands over sets & ranges
- pattern matching commands
- inline programs



getting started

installation and configuration

```
$ # pick the directory where you'll install the cobra files
$ git clone https://github.com/nimble-code/Cobra
$ ls –l
drwxrwxr-x 2 gh gh 4096 May 16 12:59 bin_linux      # executables for linux
drwxrwxr-x 2 gh gh 4096 May 16 12:59 bin_cygwin     # executables for cygwin
drwxrwxr-x 2 gh gh 4096 May 16 12:59 bin_mac        # executables for macs
drwxrwxr-x 2 gh gh 4096 May 16 10:03 doc            # change history, manpage
drwxrwxr-x 2 gh gh 4096 May 16 10:03 gui            # optional small tcl/tk script
drwxrwxr-x 8 gh gh 4096 May 16 15:55 rules          # cobra checker libraries
drwxrwxr-x 1 gh gh 4096 May 16 12:43 src            # cobra source files
drwxrwxr-x 1 gh gh 4096 May 16 12:43 src_app        # standalone cobra checkers
```

```
$ cd src
$ sudo make install_mac    # or install_cygwin, install_linux
$ cd ..
```

optional,
to compile from scratch

```
$ export PATH=$PATH:`pwd`/bin_mac          # or bin_cygwin, bin_linux
$ cobra –configure `pwd`/rules
```

recommended:
install also tcl/tk
install also graphviz
on ubuntu:
sudo apt-get install graphviz
on mac:
brew install graphviz
on cygwin:
install Cygwin-X and then
add tcl/tk and graphviz

getting started

manual pages are all online

http://spinroot.com/cobra

Cobra Static Code Analyzer

about papers manpages downloads

Cobra is a structural source code analyzer, fast enough that it can be used interactively. The tool prototype (Version 1.0) was developed at NASA's Jet Propulsion Laboratory late 2015, and released for general distribution about a year later.

Versions 2 and 3 of the tool are extended versions that can handle interactive analyses of code bases with up to millions of lines of code, while supporting a significantly richer online query scripting language. It also comes with multi-core support for many types of queries, including a new set of cyber-security related checks.

Starting with Version 3, the Cobra code is distributed in open source form at github.com/nimble-code.

Cobra can analyze C, C++, Ada, and Python, and can relatively easily be retargeted for other languages. The distribution includes sample query libraries and scripts.

For bug reports and additional information:
gholzmann atsign acm dot org

getting started

all manual pages are online

http://spinroot.com/cobra/manual.html

COBRA Reference Manual
Code Browser and Analysis Tool

Principle of Operation

Cobra uses a lexical analyzer to scan in the source code in the files given as arguments on the command-line. It then builds a data structure that can be used for querying that source code, either interactively or with predefined scripts.

The internal data structure dat Cobra builds is a basic linked list of lexical tokens, annotated with some basic information and links to other tokens, for instance to identify matching pairs of parentheses, brackets and braces. The tool does not attempt to parse the code, which means that it can handle a broad range of possible inputs. Despite the simplicity of the data structure, the tool can be remarkably powerful in quickly locating complex patterns in a code base to assist in peer review, code development, or structural code analysis.

There are several ways to write queries. You can use:

- Interactive queries (overview below, or see the index),
- Inline programs (described separately),
- Standalone checkers (described separately).

Interactive queries are written in a simple command language that can support the most frequent types of searches. When more complex queries need to be handled, requiring anything other than a sequential scan of the

getting started

all manual pages are online

http://spinroot.com/cobra/commands/index.html

Cobra Command Overview

Commands with short-hand:

a	append a source file
b	move marks back one token
B	browse a source file (cf V)
:	(colon) execute a named script
c	contains: query a range
d	display
.	(dot) read a command file
e	extend match
F	list of open files
G	grep in source files
?	help
h	command history
=	print something
i	inspect lexical tokens
j	jump
l	list
m	mark tokens

Commands without short-hand:

cfg	cfg
context	context
cpp	preprocessing
def...end	define named scripts
default	default
fcg	fcg
fcts	fcts
ff	ff
ft	ft
map	map
ncore	set nr of cores to use
nowindow	disable window popups for display commands (default)
%{...%}	inline programs
pat	pattern token expression
pe	same as pat
re	regular token expression

getting started

all manual pages are online

http://spinroot.com/cobra/commands/mark.html

Cobra **Interactive Query Language** **mark**

NAME

mark — mark tokens if they match one or two patterns

SYNTAX

```
m[ark] [qualifier]* pattern [pattern2]
pattern:   string | @string | /re | (expr)
qualifier: ir | no | &
```

DESCRIPTION

If used without qualifiers, the mark command can only add additional marks, but not remove them. The qualifiers can be used to restrict an existing set of marks to a subset.

A pattern can be one of the following:

- a string (without quotes) to match the token text precisely,
- a token type (when prefixed with a @ symbol),
- a regular expression (when preceded by a / symbol), or
- a pattern expression (when enclosed in round braces).

A qualifier is one of the three terms **ir**, **no**, or **&**. Qualifiers can be escaped as \no, \&, or \ir if a literal match is intended, as can the **/** that would otherwise identify a regular expression, or a round brace **(** that would otherwise indicate a pattern expression

getting started

predefined query libraries

try, for instance:

$ cobra –f basic *.c

or for summary output:

$ cobra –terse –f basic *.c

```
$ cd $COBRA/rules
$ ls -l
total 60
drwxr-xr-x+ 1 gh None 0 May  1 16:31 cwe
drwxr-xr-x+ 1 gh None 0 Oct 11  2018 jpl
drwxr-xr-x+ 1 gh None 0 May  6 17:16 main
drwxr-xr-x+ 1 gh None 0 Oct 11  2018 misra
drwxr-xr-x+ 1 gh None 0 Oct 11  2018 pedantic
drwxr-xr-x+ 1 gh None 0 Jun  1 14:18 play
drwxr-xr-x+ 1 gh None 0 Mar 20 15:49 stats
$ ls -l main/*.cobra
total 89
-rwxr-xr-x+ 1 USER None 1017 May 12  2017 basic.cobra
-rwxr-xr-x+ 1 USER None 3513 May 13  2017 binop.cobra
-rwxr-xr-x+ 1 USER None   21 May  6 17:16 cwe.cobra
-rwxr-xr-x+ 1 USER None  793 Apr 20  2017 extern.cobra
-rwxr-xr-x+ 1 USER None 2490 May 13  2017 iridex.cobra
-rwxr-xr-x+ 1 USER None 4004 May 15  2017 jpl.cobra
-rwxr-xr-x+ 1 USER None  589 May 12  2017 metrics.cobra
-rwxr-xr-x+ 1 USER None  714 May 12  2017 misra1997.cobra
-rwxr-xr-x+ 1 USER None  725 May 12  2017 misra2004.cobra
-rwxr-xr-x+ 1 USER None  658 May 12  2017 misra2012.cobra
-rwxr-xr-x+ 1 USER None  501 May 12  2017 p10.cobra
-rwxr-xr-x+ 1 USER None 1008 May 31 09:02 reverse_null.cobra
-rwxr-xr-x+ 1 USER None  585 May  6 17:09 stats.cobra
```

getting started

languages supported

- Cobra is language neutral, which means that:
  - it can be retargeted to a broad range of languages, by providing it with the relevant set of lexical tokens types
  - the default is C, predefined alternatives include:

    $ cobra –Ada …

    $ cobra –Java …

    $ cobra –C++ …

    $ cobra –Python …
  - other languages can be added by using the *map* command (discussed in Part 3 of this tutorial)
- to see all currently recognized cobra command-line flags:

  $ cobra --