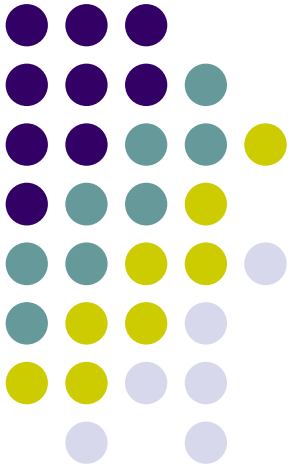
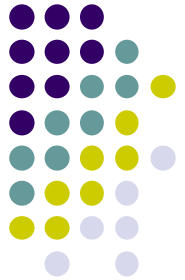
TBCppA: a Tracer Approach for Automatic Accurate Analysis of C Preprocessor's Behaviors

K. Gondow
(Tokyo Institute of Tech., Japan)



Demo: TBCppA



- Produces CPP info. in XML.

- Visualizing macro expansions (tmacro).
  - STREQ in bash-3.1/shell.c

demo

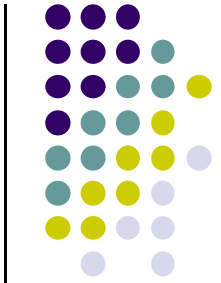
- Visualizing conditional compilation (tifdef).
  - bash-3.1/execute_cmd.c#1068

demo

- Recovering the original code (trecov-demo).
  - bash-3.1/shell.c.{E,xml}→shell.c

demo

Hard to parse C programs.

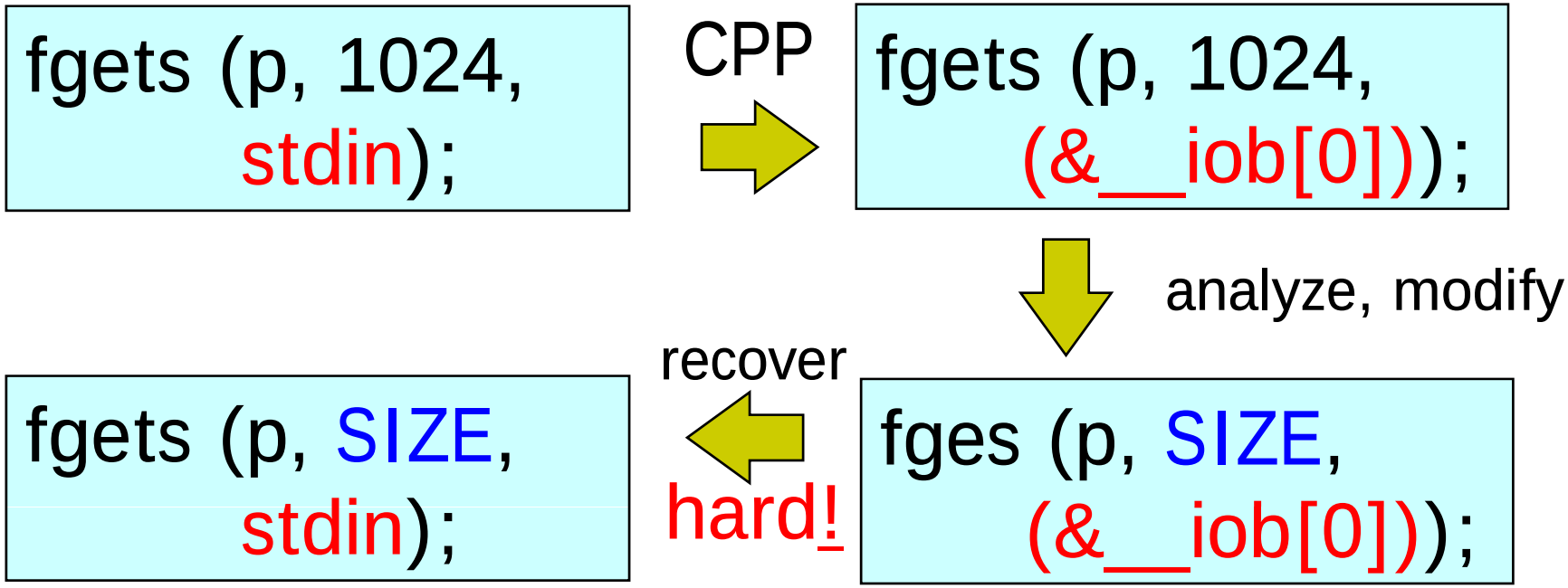


- Unpreprocessed C programs are not parseable.
- Simply because they are syntactically incorrect. (eg., unbalanced parentheses)

```
#ifdef NO_FSTAT
  if (stat(ofname, &ostat) != 0) {
#else
  if (fstat(ofd, &ostat) != 0) {
#endif
    fprintf(stderr, "%s: ", progname);
}
```

Hard to recover unpreprocessed version

- Preprocessed C programs are parseable.
- But hard to recover the unpreprocessed version.



TBCppA instrumentation (1)

```
#define  N        10
printf ("%d",      N     );
```

- Embeds XML-like tracers in C program.

```
@"define id='F4_1' dir='#define' name='N'
        macro_body='10' first_line='2' /"
#define N   @"macro_body ref='F4_1'" ¥
                          10 @"/macro_body"
printf("%d",
       @"macro_call name='N' first_line='3'"
       N
       @"/macro_call" );
```

TBCppA instrumentation (2)

- Preprocesses C program with the tracers using native CPP (eg., gcc -E).
- Converts tracers to XML and uses XML-escapes.

```
<define id="F4_1" dir="#define" name="N"
        macro_body="10" first_line="2" />
printf (&amp;quot;%d&amp;quot;,
      <macro_call name="N" first_line="3">
      <macro_body ref="F4_1"> 10 </macro_body>
      </macro_call> );
```

TBCppA: pros and cons

- Advantages:
  - Can almost perfectly process C programs.
    - Eg., gcc-4.1.1 (630KLOC) processed in 60 min.
  - Highly portable and maintainable.

- Some limitations:
  - No support for GCC-specific variadic macros.
  - Does not analyze conditionally excluded code.
  - ...

Other approaches

- Modifying real CPPs (e.g., PCp3)
  - low portability and low applicability
- Developing CPP emulators
  - inaccuracy

Summary

- Problem:
  - Very difficult to accurately obtain CPP mapping info. in a simple and portable way.
- Our solution: TBCppA
  - Makes native CPP (i.e., gcc -E) process instrumented unpreprocessed C source code.

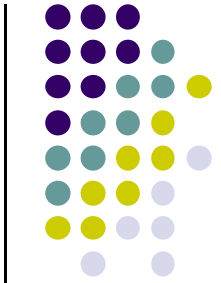
What makes C analysis imprecise

- Compiler-specific extensions
  - eg., asm, __attribute__
- Ambiguity in C standards
  - eg., padding in structures
- Signals, multithreading,
- System calls like fork and longjmp.
- Pointer alias
  - pointer arithmetic, casting, function pointers, ...
- C preprocessor
- Dynamic link, interpositioning

Low portability and low applicability in modifying CPP

- Modified CPPs are likely to run only on the specific platforms.
  - The cost of porting and maintaining them, every time a new GCC is released, is high.
- PCp3 is a well-known modified CPP, but it is based on very old CPP (GCC-2.7.2.2).

Inaccuracy in emulating CPP



- The accurate emulation is difficult:
  - Implementation-defined, unspecified behaviors of the C standard.
    - E.g., Search path for #include <foo.h>
  - Platform-specific, compiler-specific pre-defined macros.
    - E.g., stdin, unix, i386
  - Difficulty conforming to CPP spec., and mimicking the existing implementation (CPP options).
- Refactoring tools need accurate info.

Execution time, file sizes

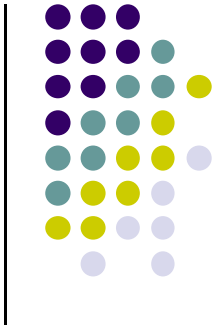


Table 1. Execution time of `TBCppA`'s embedding tracers in system header files and the resulting file size

# of files	original size of `*.h`	execution time	size of `*.h` + tracers
480	140 KLOC (5.0MB)	89 sec	73.8MB

Table 2. Execution time of `TBCppA` generating CPP mapping information and the generated file size

	size of `*.h`	size of `*.c` before CPP	size of `*.c` after CPP ¶	processing time of `*.h`	processing time of `*.c`	size of `*.c` + tracers ¶	generated file size `*.c.{xml,lex}`
`hello.c`†	—	5 lines (76B)	76B	—	0.45 sec	2.8KB	0.8KB+4.4KB
`hello2.c`†	—	5 lines (63B)	24KB	—	1.4 sec	1.7KB	520KB+3KB
gzip-1.2.4	7.4KB	7.3 KLOC (230KB)	1.5MB	0.6 sec	32 sec	3.1MB	11.9MB+7.0MB
bash-3.1‡	420KB	70 KLOC (1.8MB)	10MB	13 sec	485 sec	33MB	191MB+61MB
gcc-4.1.1§	3.7MB	630 KLOC (18MB)	102MB	72 sec	3,602 sec	332MB	1.2GB+554MB

† `hello.c` does not `#include <stdio.h>`, while `hello2.c` `#includes <stdio.h>`.

13 times larger

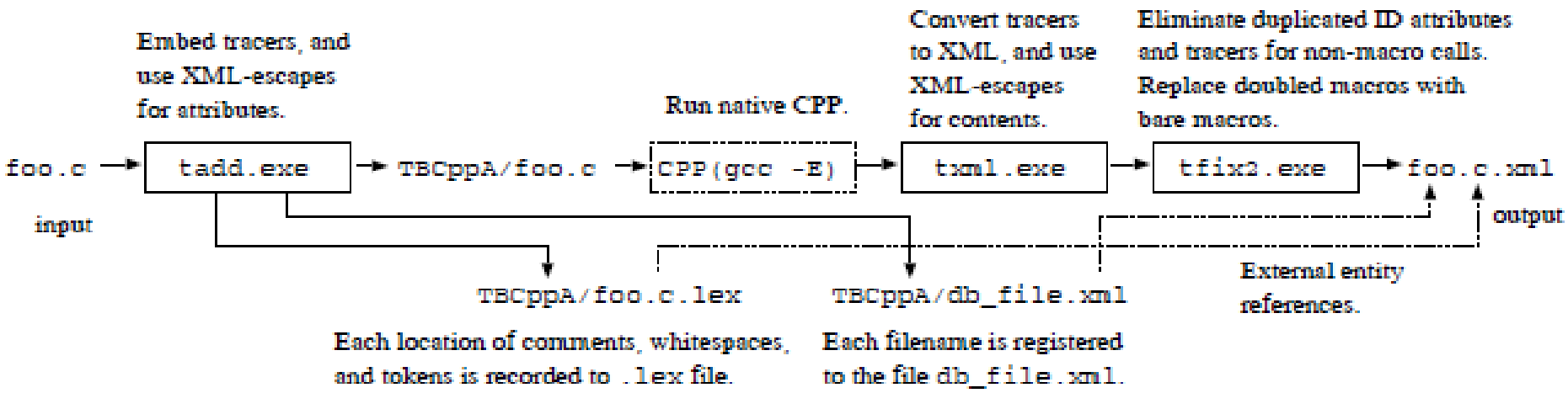


Figure 3. TBCppA**'s components and process flow chart**

tracer

- Identifiable substance (e.g., radioactive isotope in biological fields.)
- Makes it easy to observe the behavior or distribution of the observee.

tracer