CISPA
HELMHOLTZ CENTER FOR
INFORMATION SECURITY

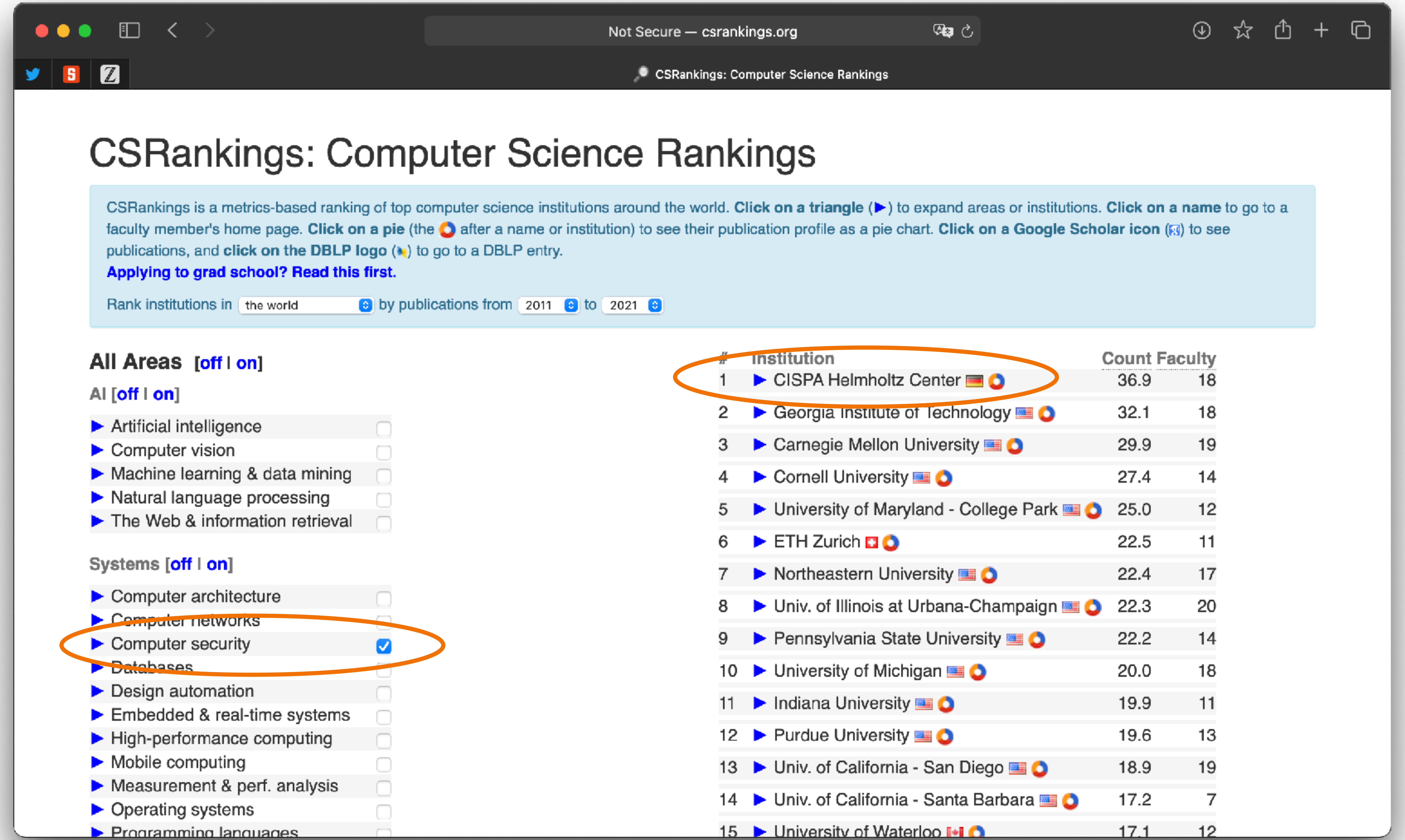
Language-Based Fuzzing

Journées Nationales 2021 du GDR Sécurité Informatique • 2 Juillet 2021

Andreas Zeller

with Rahul Gopinath, Rafael Dutra, and Zeller's team at CISPA

CISPA
HELMHOLTZ CENTER FOR
INFORMATION SECURITY


Not Secure — csrankings.org
CSRankings: Computer Science Rankings

CSRankings: Computer Science Rankings

CSRankings is a metrics-based ranking of top computer science institutions around the world. **Click on a triangle** (▶) to expand areas or institutions. **Click on a name** to go to a faculty member's home page. **Click on a pie** (the ● after a name or institution) to see their publication profile as a pie chart. **Click on a Google Scholar icon** to see publications, and **click on the DBLP logo** to go to a DBLP entry.

Applying to grad school? Read this first.

Rank institutions in [the world] by publications from [2011] to [2021]

All Areas [off | on]

AI [off | on]

▶ Artificial intelligence	☐
▶ Computer vision	☐
▶ Machine learning & data mining	☐
▶ Natural language processing	☐
▶ The Web & information retrieval	☐

Systems [off | on]

▶ Computer architecture	☐
▶ Computer networks	☐
▶ Computer security	✓
▶ Databases	☐
▶ Design automation	☐
▶ Embedded & real-time systems	☐
▶ High-performance computing	☐
▶ Mobile computing	☐
▶ Measurement & perf. analysis	☐
▶ Operating systems	☐
▶ Programming languages	☐

#	Institution	Count	Faculty
1	▶ CISPA Helmholtz Center	36.9	18
2	▶ Georgia Institute of Technology	32.1	18
3	▶ Carnegie Mellon University	29.9	19
4	▶ Cornell University	27.4	14
5	▶ University of Maryland - College Park	25.0	12
6	▶ ETH Zurich	22.5	11
7	▶ Northeastern University	22.4	17
8	▶ Univ. of Illinois at Urbana-Champaign	22.3	20
9	▶ Pennsylvania State University	22.2	14
10	▶ University of Michigan	20.0	18
11	▶ Indiana University	19.9	11
12	▶ Purdue University	19.6	13
13	▶ Univ. of California - San Diego	18.9	19
14	▶ Univ. of California - Santa Barbara	17.2	7
15	▶ University of Waterloo	17.1	12

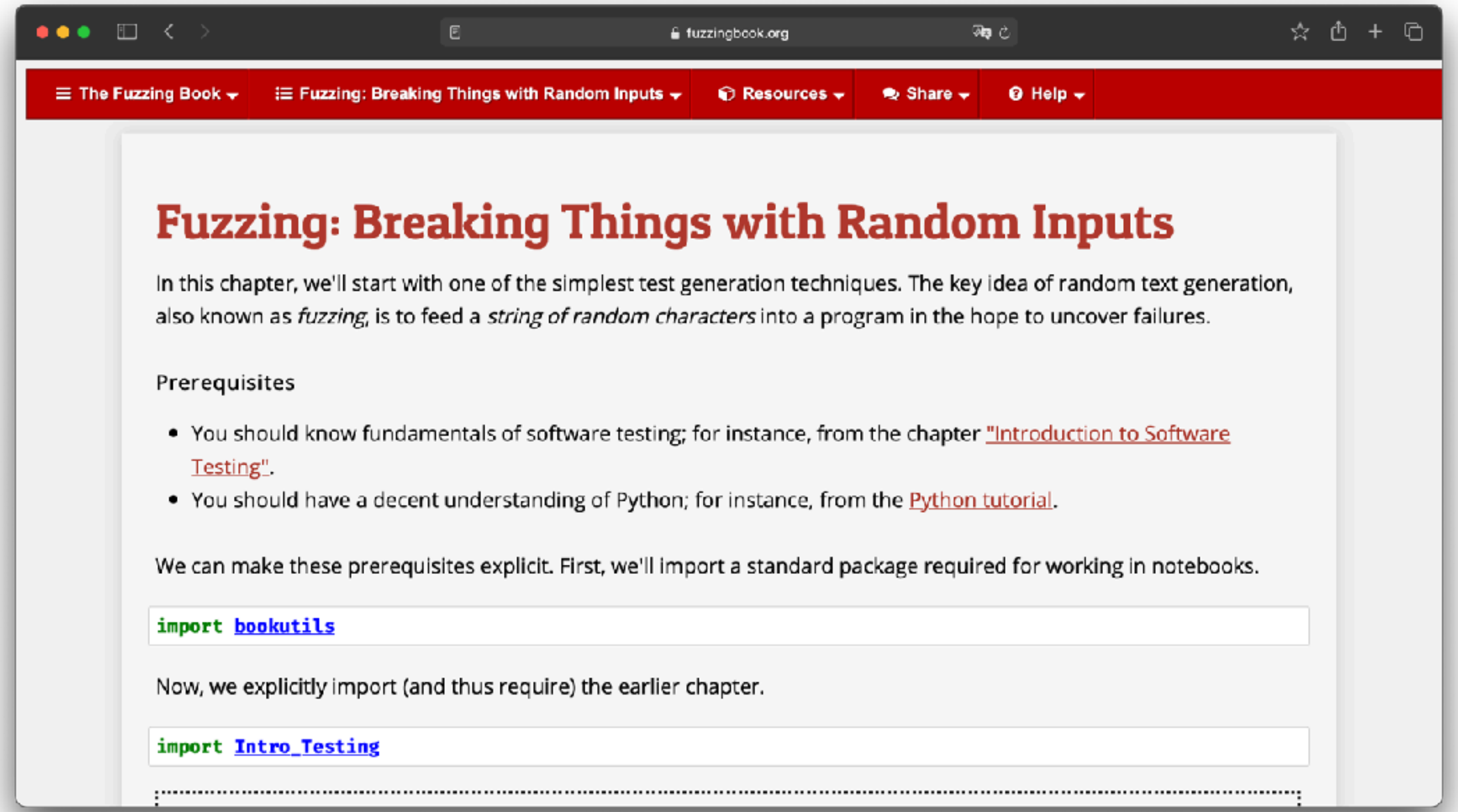
Fuzzing

```
'+),9+3!>5=;-"736 ..=;"/,/,8:\'+.=8!?
>03"$""823-(>1(.=-$00)$-7>-*-.*"$9=.40\'/7=5
:\')42: /2:))<0/"7<1$2;-7/,;9$69%63%26?6?
3+3(3$,$+-;(62387+ 3$0:1:653,.-(#4*6\'2$?
&6$<1 <7.0/8*>4<471$4%/\'!7=8*?41 <=--+*?
671&:#&,754(=\'>;=,*-2440\'29/;?%!
\',9(7252;,5 6)9*4\'!.",$+(<5;3,.$,*-7064,7!
90(<<,9;%3$;:&;#&$80<2<($).&&4."
.7<(8<8:/6<88315<,!9718573#?1-4-:#>>4!
77+/)4+2*6$* %8$/=%9/)<0/%8#$20,;/87%73?1 *?
6&,10;54::-"9)--;3>$=<)\'8+%><%%4\'"</+/
```

Modern Fuzzers

fuzzingbook.org

- Mutate given **input seeds**
- Are guided by **coverage**
- Solve **path constraints**
- Can be applied **out of the box**
- Find **bugs** and **vulnerabilities**



Fuzzing: Breaking Things with Random Inputs

In this chapter, we'll start with one of the simplest test generation techniques. The key idea of random text generation, also known as *fuzzing*, is to feed a *string of random characters* into a program in the hope to uncover failures.

Prerequisites

- You should know fundamentals of software testing; for instance, from the chapter "Introduction to Software Testing".
- You should have a decent understanding of Python; for instance, from the Python tutorial.

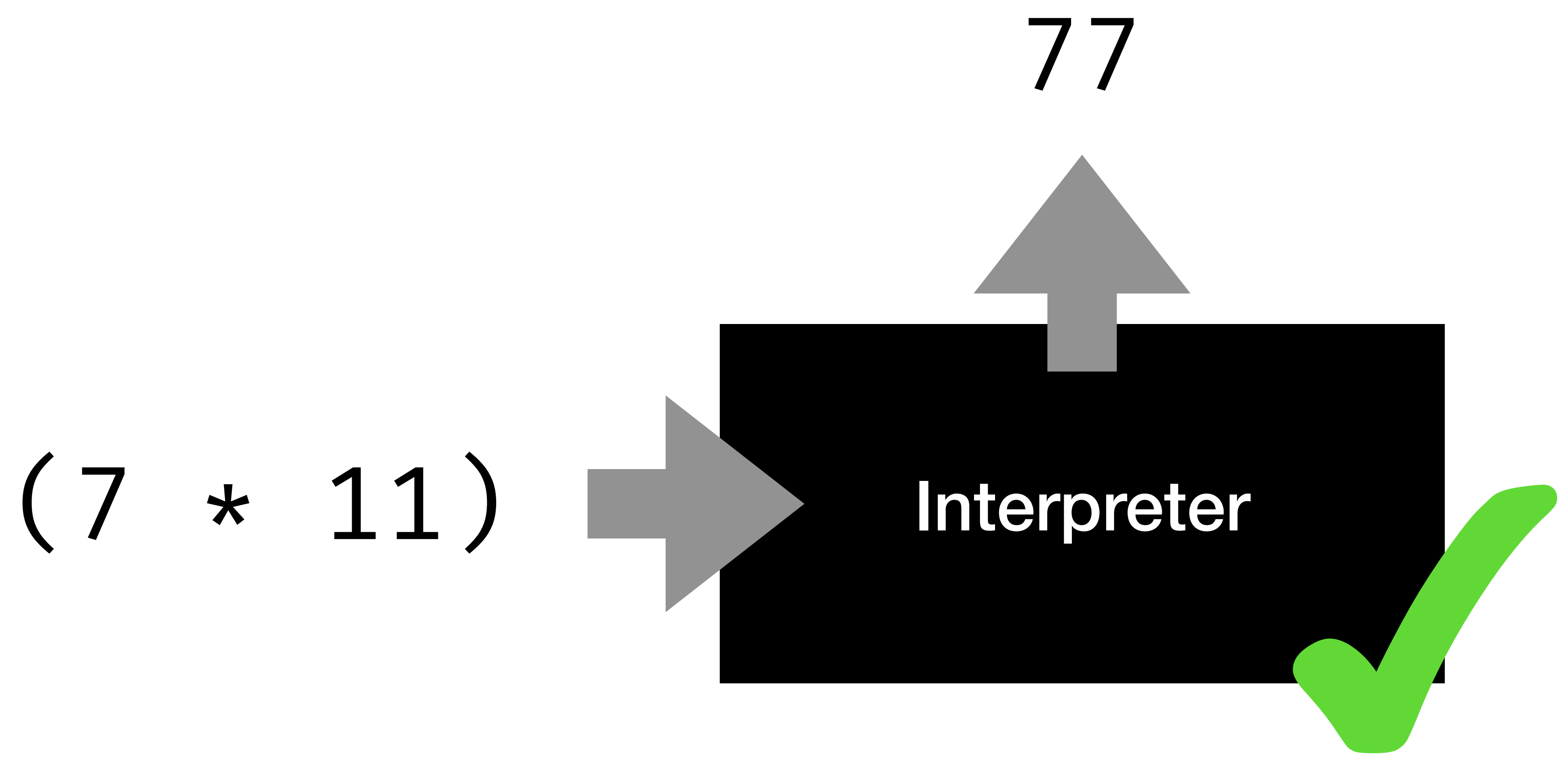
We can make these prerequisites explicit. First, we'll import a standard package required for working in notebooks.

```
import bookutils
```

Now, we explicitly import (and thus require) the earlier chapter.

```
import Intro_Testing
```

A Language Processor



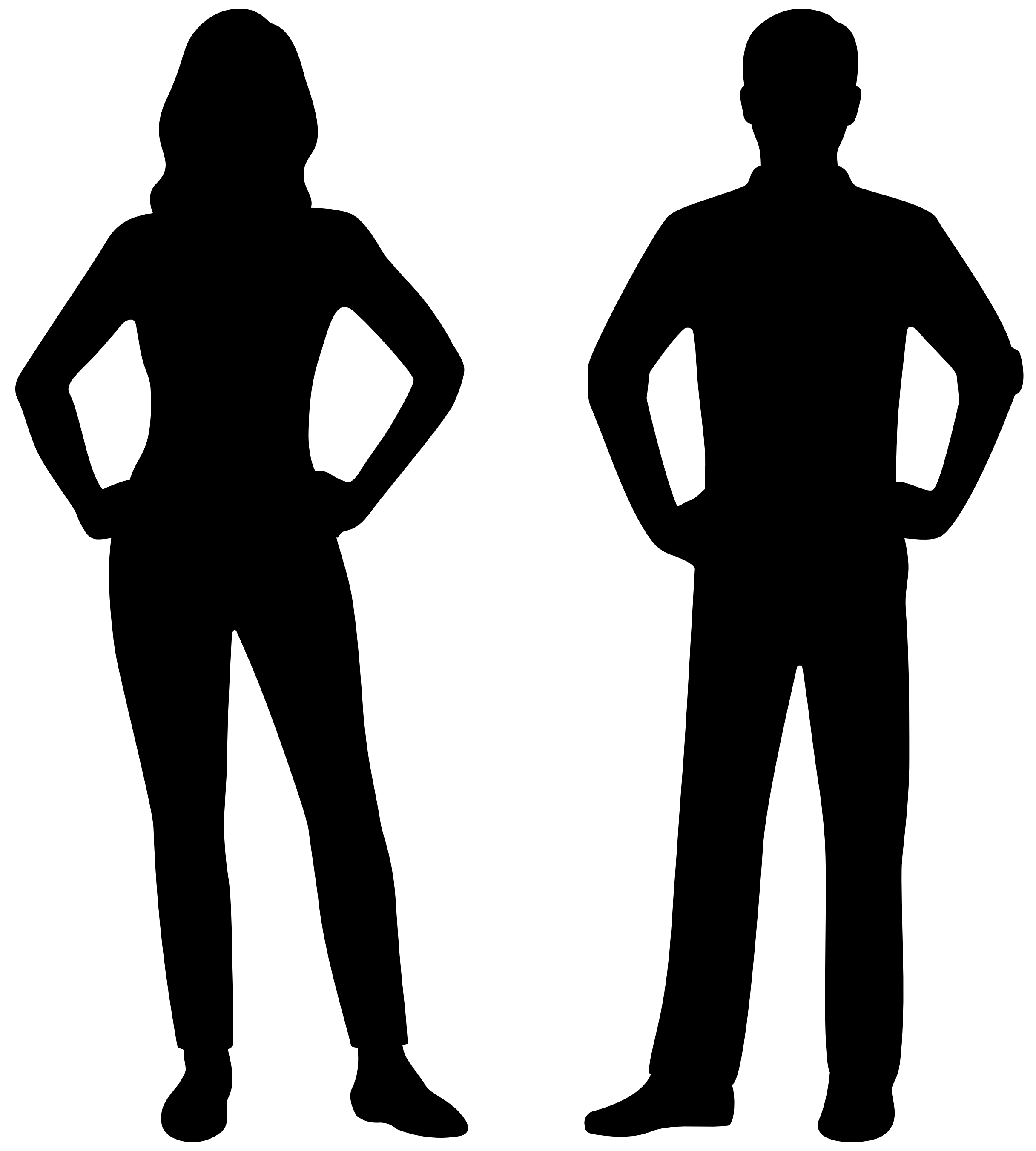
Most Inputs are Invalid

```
(144 60 )5(5-(05*/(   * *)910)25/509505)3)/
09211762 /(7**+22)76-+/29+/4**2+

8(  )04/844)

4)632/3/7 *0525+)7*
```

How can we teach the fuzzer what an expression looks like?

Interpreter
?

Taming Fuzzers: Adapt Fuzzing to *Your* Needs



You know

- about the **domain**
- about the **program**
- about its **input**
- about **what needs to be tested**

How do you get this into a fuzzer?

Grammars

Specify a language (= a set of inputs)

Expansion rule

Nonterminal symbol

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩ + ⟨expr⟩ | ⟨term⟩ - ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩ * ⟨factor⟩ | ⟨term⟩ / ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= + ⟨factor⟩ | - ⟨factor⟩ | ( ⟨expr⟩ ) | ⟨int⟩ | ⟨int⟩ . ⟨int⟩
⟨int⟩     ::= ⟨digit⟩ ⟨int⟩ | ⟨digit⟩
⟨digit⟩   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Terminal symbol

Grammars as Producers

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩  +  ⟨expr⟩ | ⟨term⟩  -  ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩  *  ⟨factor⟩ | ⟨term⟩  /  ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= + ⟨factor⟩ | - ⟨factor⟩ | ( ⟨expr⟩ ) | ⟨int⟩ | ⟨int⟩ . ⟨int⟩
⟨int⟩     ::= ⟨digit⟩  ⟨int⟩ | ⟨digit⟩
⟨digit⟩   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Grammars as Producers

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩  +  ⟨expr⟩ | ⟨term⟩  -  ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩  *  ⟨factor⟩ | ⟨term⟩  /  ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= + ⟨factor⟩ | - ⟨factor⟩ | ( ⟨expr⟩ ) | ⟨int⟩ | ⟨int⟩ . ⟨int⟩
⟨int⟩     ::= ⟨digit⟩  ⟨int⟩ | ⟨digit⟩
⟨digit⟩   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

⟨start⟩

Grammars as Producers

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩  +  ⟨expr⟩ | ⟨term⟩  -  ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩  *  ⟨factor⟩ | ⟨term⟩  /  ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= + ⟨factor⟩ | - ⟨factor⟩ | ( ⟨expr⟩ ) | ⟨int⟩ | ⟨int⟩ . ⟨int⟩
⟨int⟩     ::= ⟨digit⟩  ⟨int⟩ | ⟨digit⟩
⟨digit⟩   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

⟨expr⟩

Grammars as Producers

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩ + ⟨expr⟩ | ⟨term⟩ - ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩ * ⟨factor⟩ | ⟨term⟩ / ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= + ⟨factor⟩ | - ⟨factor⟩ | ( ⟨expr⟩ ) | ⟨int⟩ | ⟨int⟩ . ⟨int⟩
⟨int⟩     ::= ⟨digit⟩ ⟨int⟩ | ⟨digit⟩
⟨digit⟩   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

```
⟨term⟩ - ⟨expr⟩
```

Grammars as Producers

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩ + ⟨expr⟩ | ⟨term⟩ - ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩ * ⟨factor⟩ | ⟨term⟩ / ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= + ⟨factor⟩ | - ⟨factor⟩ | ( ⟨expr⟩ ) | ⟨int⟩ | ⟨int⟩ . ⟨int⟩
⟨int⟩     ::= ⟨digit⟩ ⟨int⟩ | ⟨digit⟩
⟨digit⟩   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

```
⟨term⟩ - ⟨expr⟩
```

Grammars as Producers

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩ + ⟨expr⟩ | ⟨term⟩ - ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩ * ⟨factor⟩ | ⟨term⟩ / ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= + ⟨factor⟩ | - ⟨factor⟩ | ( ⟨expr⟩ ) | ⟨int⟩ | ⟨int⟩ . ⟨int⟩
⟨int⟩     ::= ⟨digit⟩ ⟨int⟩ | ⟨digit⟩
⟨digit⟩   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

```
⟨factor⟩ - ⟨expr⟩
```

Grammars as Producers

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩  +  ⟨expr⟩ | ⟨term⟩  -  ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩  *  ⟨factor⟩ | ⟨term⟩  /  ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= + ⟨factor⟩ | - ⟨factor⟩ | ( ⟨expr⟩ ) | ⟨int⟩ | ⟨int⟩ . ⟨int⟩
⟨int⟩     ::= ⟨digit⟩  ⟨int⟩ | ⟨digit⟩
⟨digit⟩   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

```
⟨int⟩ . ⟨int⟩ - ⟨expr⟩
```

Grammars as Producers

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩  +  ⟨expr⟩ | ⟨term⟩  -  ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩  *  ⟨factor⟩ | ⟨term⟩  /  ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= + ⟨factor⟩ | - ⟨factor⟩ | ( ⟨expr⟩ ) | ⟨int⟩ | ⟨int⟩ . ⟨int⟩
⟨int⟩     ::= ⟨digit⟩  ⟨int⟩ | ⟨digit⟩
⟨digit⟩   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

```
⟨digit⟩ . ⟨int⟩ - ⟨expr⟩
```

Grammars as Producers

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩  +  ⟨expr⟩ | ⟨term⟩  -  ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩  *  ⟨factor⟩ | ⟨term⟩  /  ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= + ⟨factor⟩ | - ⟨factor⟩ | ( ⟨expr⟩ ) | ⟨int⟩ | ⟨int⟩ . ⟨int⟩
⟨int⟩     ::= ⟨digit⟩  ⟨int⟩ | ⟨digit⟩
⟨digit⟩   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

```
⟨digit⟩ . ⟨digit⟩ - ⟨expr⟩
```

Grammars as Producers

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩  +  ⟨expr⟩ | ⟨term⟩  -  ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩  *  ⟨factor⟩ | ⟨term⟩  /  ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= + ⟨factor⟩ | - ⟨factor⟩ | ( ⟨expr⟩ ) | ⟨int⟩ | ⟨int⟩ . ⟨int⟩
⟨int⟩     ::= ⟨digit⟩  ⟨int⟩ | ⟨digit⟩
⟨digit⟩   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

```
8. ⟨digit⟩ - ⟨expr⟩
```

Grammars as Producers

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩  +  ⟨expr⟩ | ⟨term⟩  -  ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩  *  ⟨factor⟩ | ⟨term⟩  /  ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= + ⟨factor⟩ | - ⟨factor⟩ | ( ⟨expr⟩ ) | ⟨int⟩ | ⟨int⟩ . ⟨int⟩
⟨int⟩     ::= ⟨digit⟩  ⟨int⟩ | ⟨digit⟩
⟨digit⟩   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

```
8.2 - ⟨expr⟩
```

Grammars as Producers

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩  +  ⟨expr⟩ | ⟨term⟩  -  ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩  *  ⟨factor⟩ | ⟨term⟩  /  ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= + ⟨factor⟩ | - ⟨factor⟩ | ( ⟨expr⟩ ) | ⟨int⟩ | ⟨int⟩ . ⟨int⟩
⟨int⟩     ::= ⟨digit⟩  ⟨int⟩ | ⟨digit⟩
⟨digit⟩   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

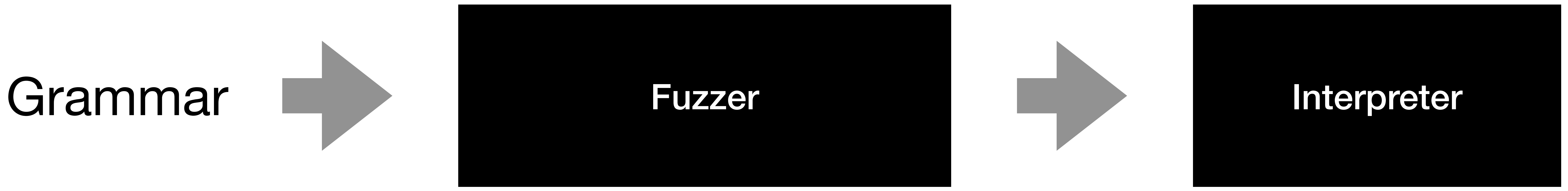
```
8.2 - 27 - -9 / +((+9 * --2 + --+-+-((-1 *
+(8 - 5 - 6)) * (-((-+(((+(4)))))) - ++4) / +
(-+---((5.6 - --(3 * -1.8 * +(6 * +-(((-(-6)
* ---+6)) / +--(+-+-7 * (-0 * (+(((((2)) + 8
- 3 - ++9.0 + ---(--+7 / (1 / +++6.37) + (1)
/ 482) / +++-+0)))) * -+5 + 7.513)))) -
(+1 / ++((-84)))))))) * ++5 / +-(--2 - -+
+-9.0)))) / 5 * --++090
```

Fuzzing with Grammars

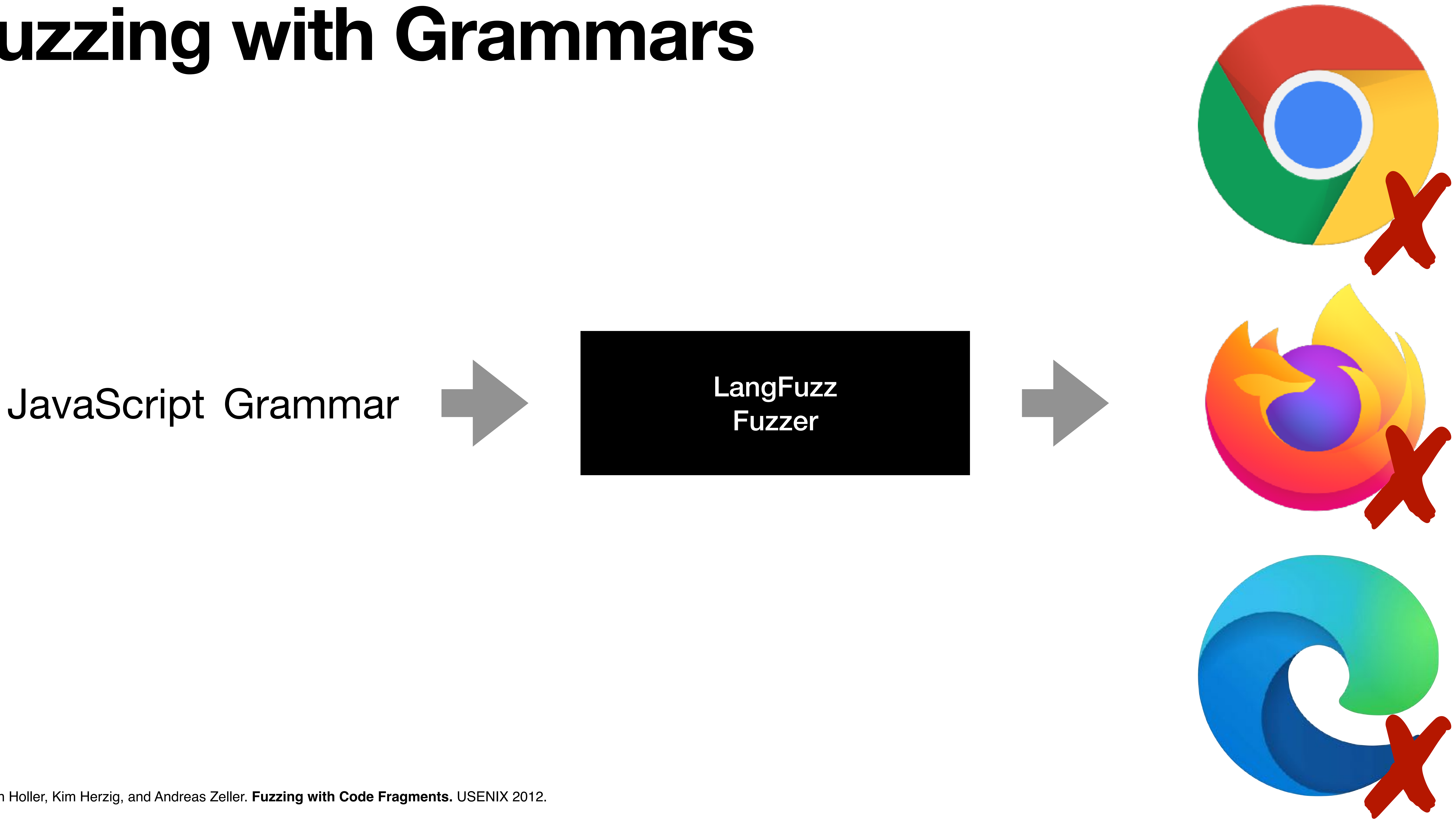
```
8.2 - 27 - -9 / +((+9 * --2 + --+-+-((-1 *
+(8 - 5 - 6)) * (-((-+(((+(4)))))) - ++4) / +
(-+---((5.6 - --(3 * -1.8 * +(6 * +-(((-(-6)
* ---+6)) / +--(+-+-7 * (-0 * (+(((((2)) + 8
- 3 - ++9.0 + ---(--+7 / (1 / +++6.37) + (1)
/ 482) / +++-+0)))) * -+5 + 7.513)))) -
(+1 / ++((-84)))))))) * ++5 / +-(--2 - -+
+-9.0)))) / 5 * --++090
```

Interpreter

Fuzzing with Grammars

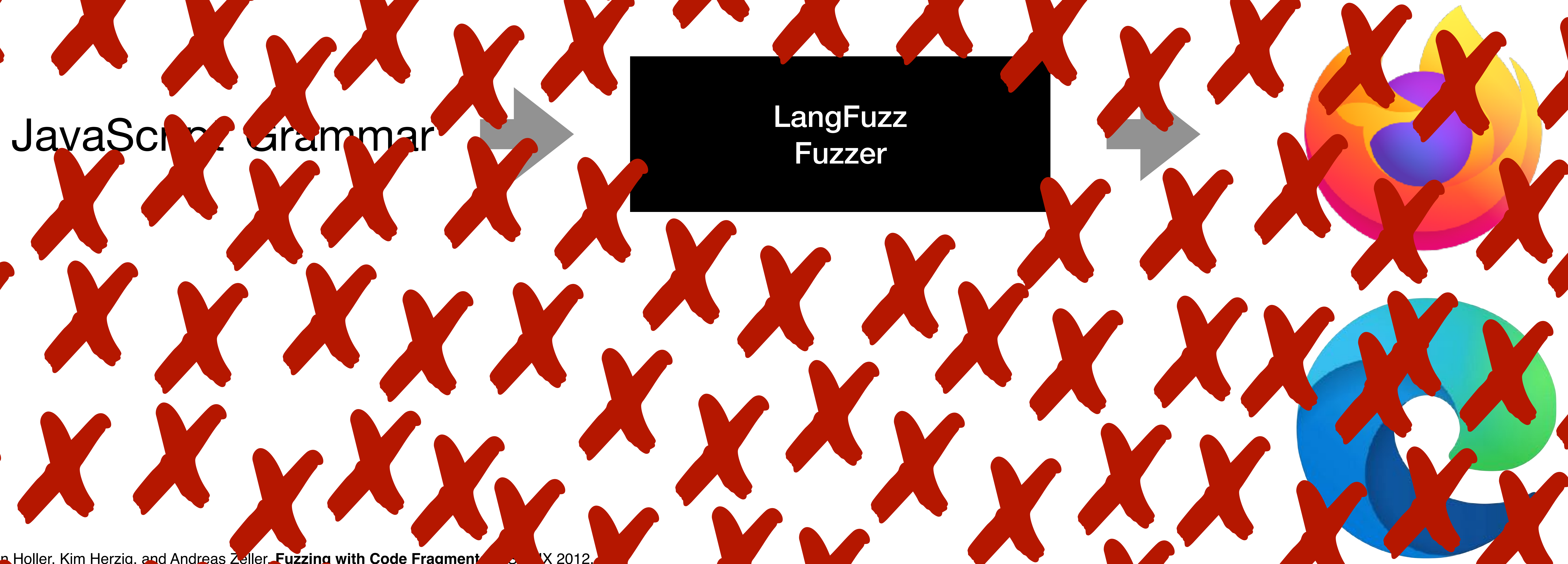


Fuzzing with Grammars

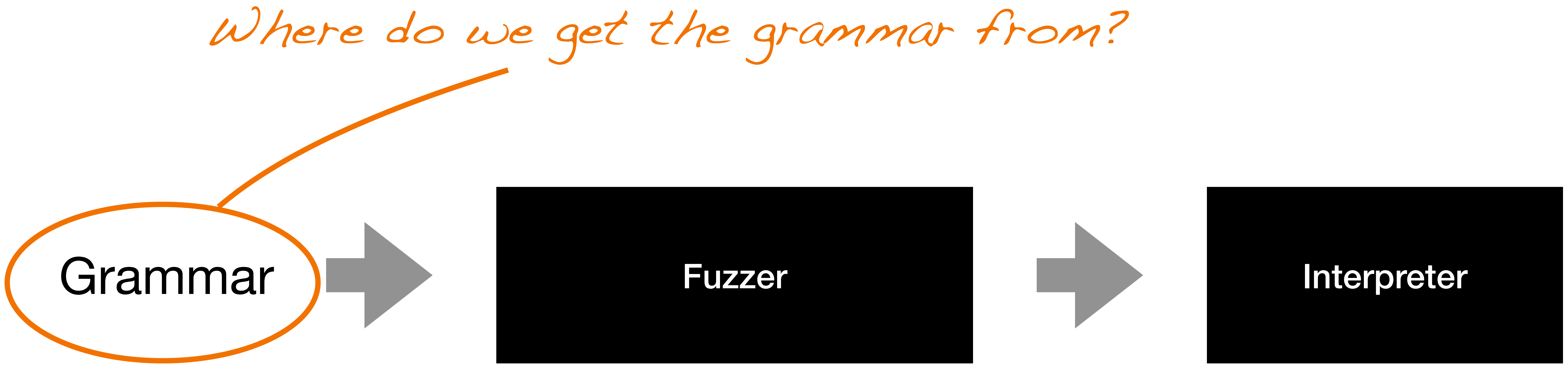


Christian Holler, Kim Herzig, and Andreas Zeller. **Fuzzing with Code Fragments.** USENIX 2012.

Fuzzing with Grammars



Fuzzing with Grammars



Alternative #1: Mining Grammars

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩  +  ⟨expr⟩ | ⟨term⟩  -  ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩  *  ⟨factor⟩ | ⟨term⟩  /  ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= + ⟨factor⟩ | - ⟨factor⟩ | ( ⟨expr⟩ ) | ⟨int⟩ | ⟨int⟩ . ⟨int⟩
⟨int⟩     ::= ⟨digit⟩  ⟨int⟩ | ⟨digit⟩
⟨digit⟩   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

```
void parse_expr() {
    parse_term();
    if (lookahead() == '+') { consume(); parse_expr(); }
    if (lookahead() == '-') { consume(); parse_expr(); }
}
void parse_term() { ... }
void parse_factor() { ... }
void parse_int() { ... }
void parse_digit() { ... }
```

Rules and Locations

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩  +  ⟨expr⟩ | ⟨term⟩  -  ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩  *  ⟨factor⟩ | ⟨term⟩  /  ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= + ⟨factor⟩ | - ⟨factor⟩ | ( ⟨expr⟩ ) | ⟨int⟩ | ⟨int⟩ . ⟨int⟩
⟨int⟩     ::= ⟨digit⟩  ⟨int⟩ | ⟨digit⟩
⟨digit⟩   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

```
void parse_expr() {
    parse_term();
    if (lookahead() == '+') { consume(); parse_expr(); }
    if (lookahead() == '-') { consume(); parse_expr(); }
}
```

Consumption

The character is last accessed (consumed) in this method

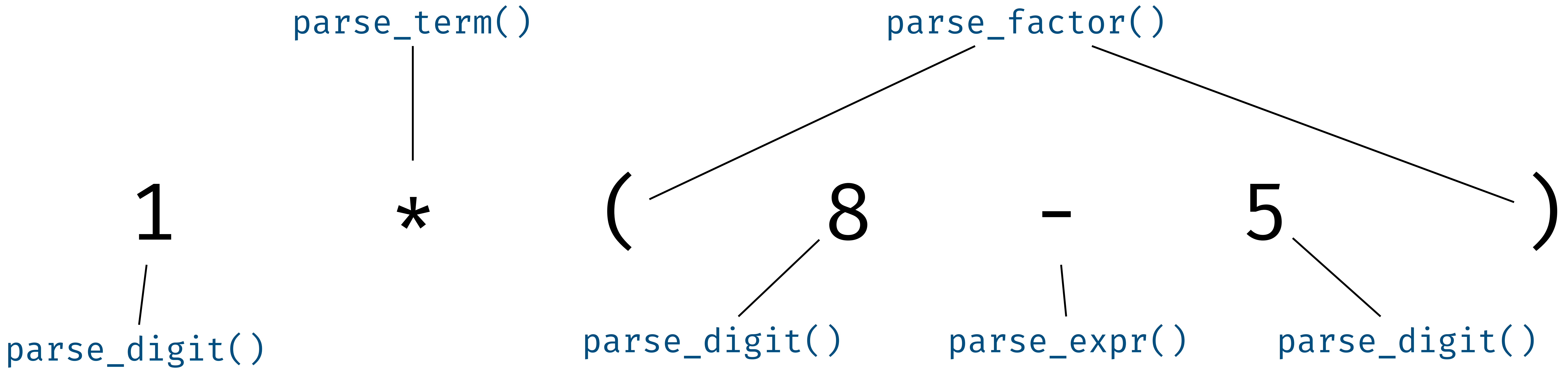
```
void parse_expr() {
    parse_term();
    if (lookahead() == '+') { consume(); parse_expr(); }
    if (lookahead() == '-') { consume(); parse_expr(); }
}
```

Consumption

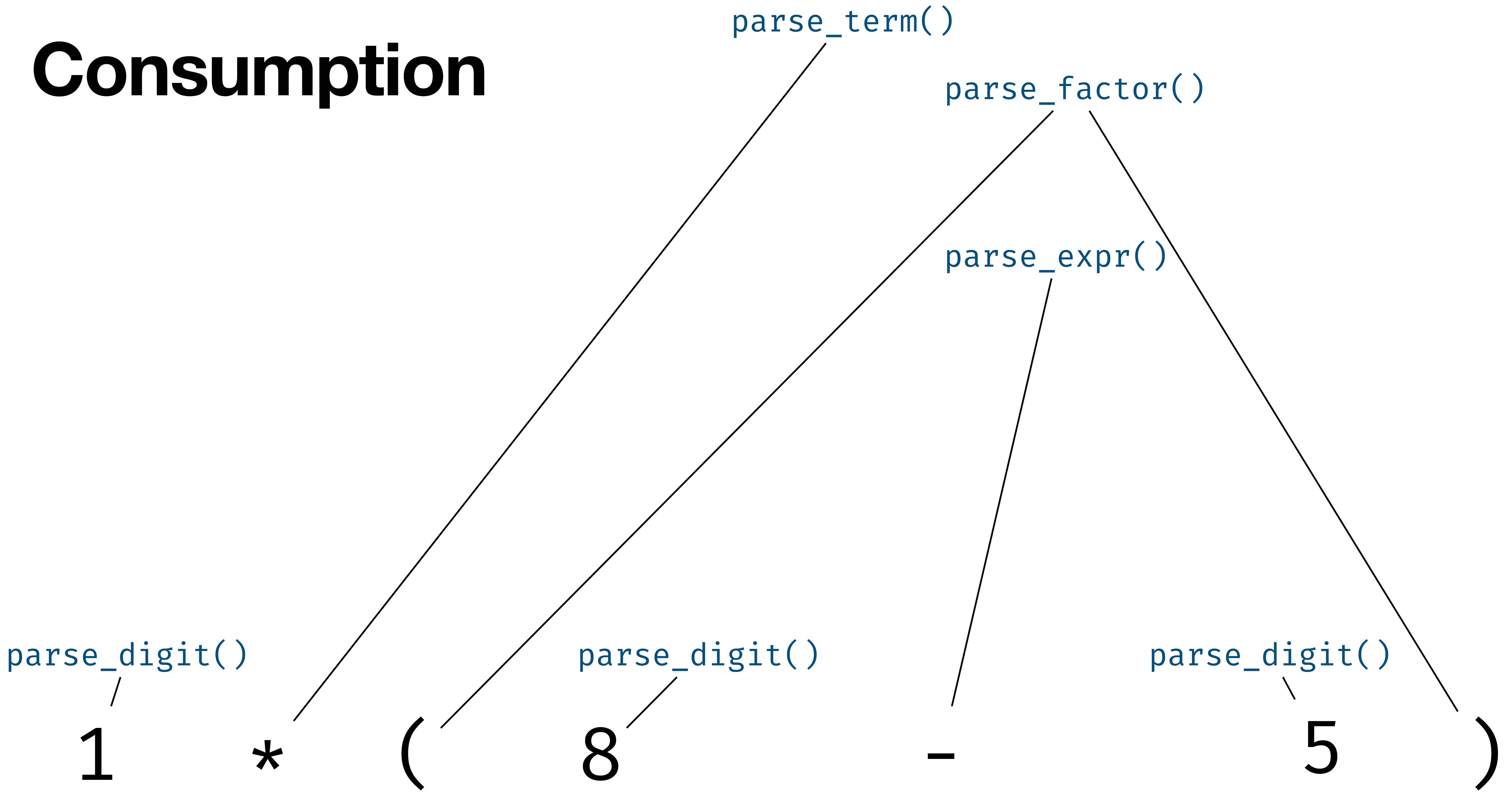
For each input character, we dynamically track where it is consumed

```
1 * ( 8 - 5 )
```

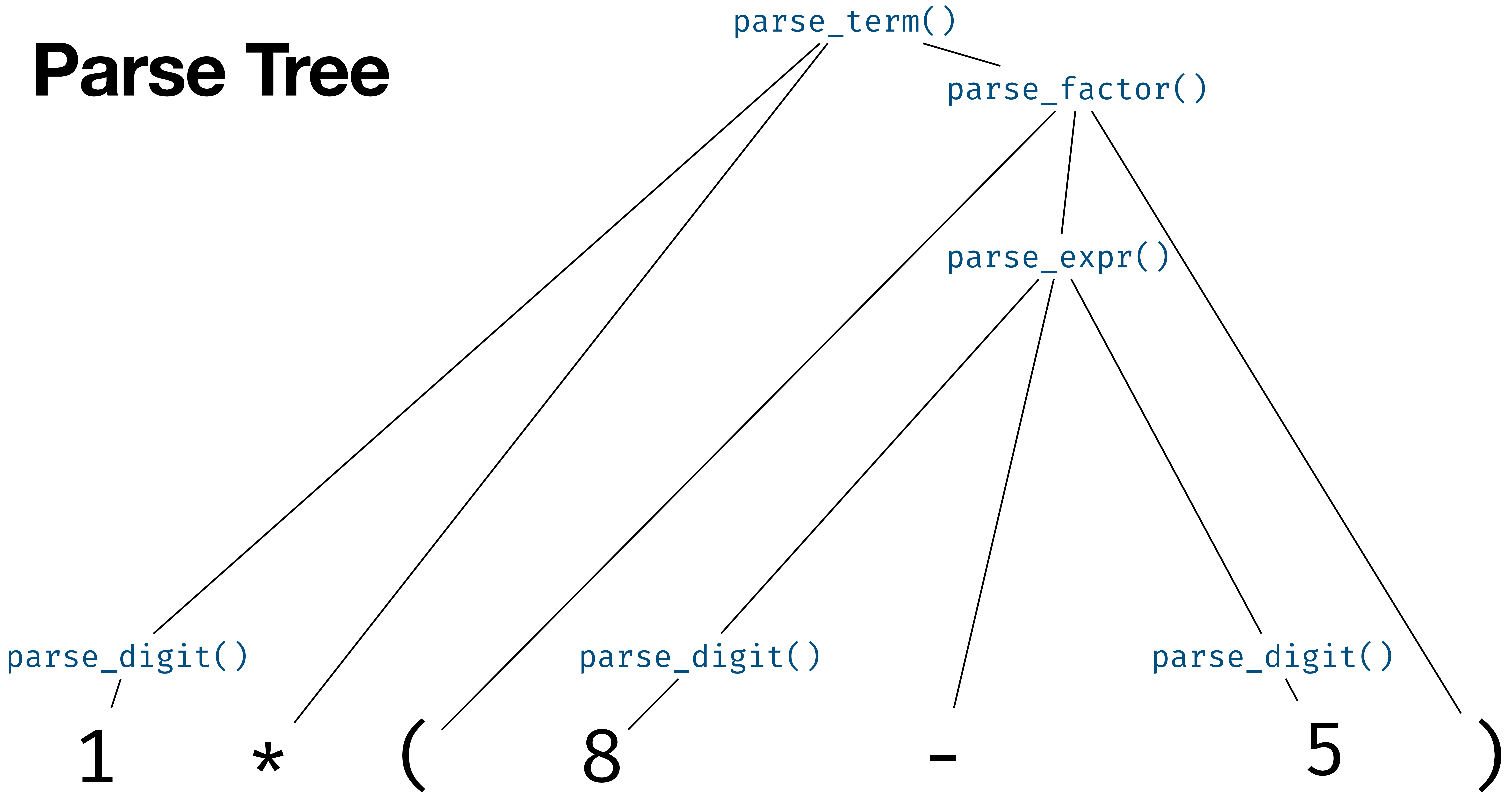
Consumption



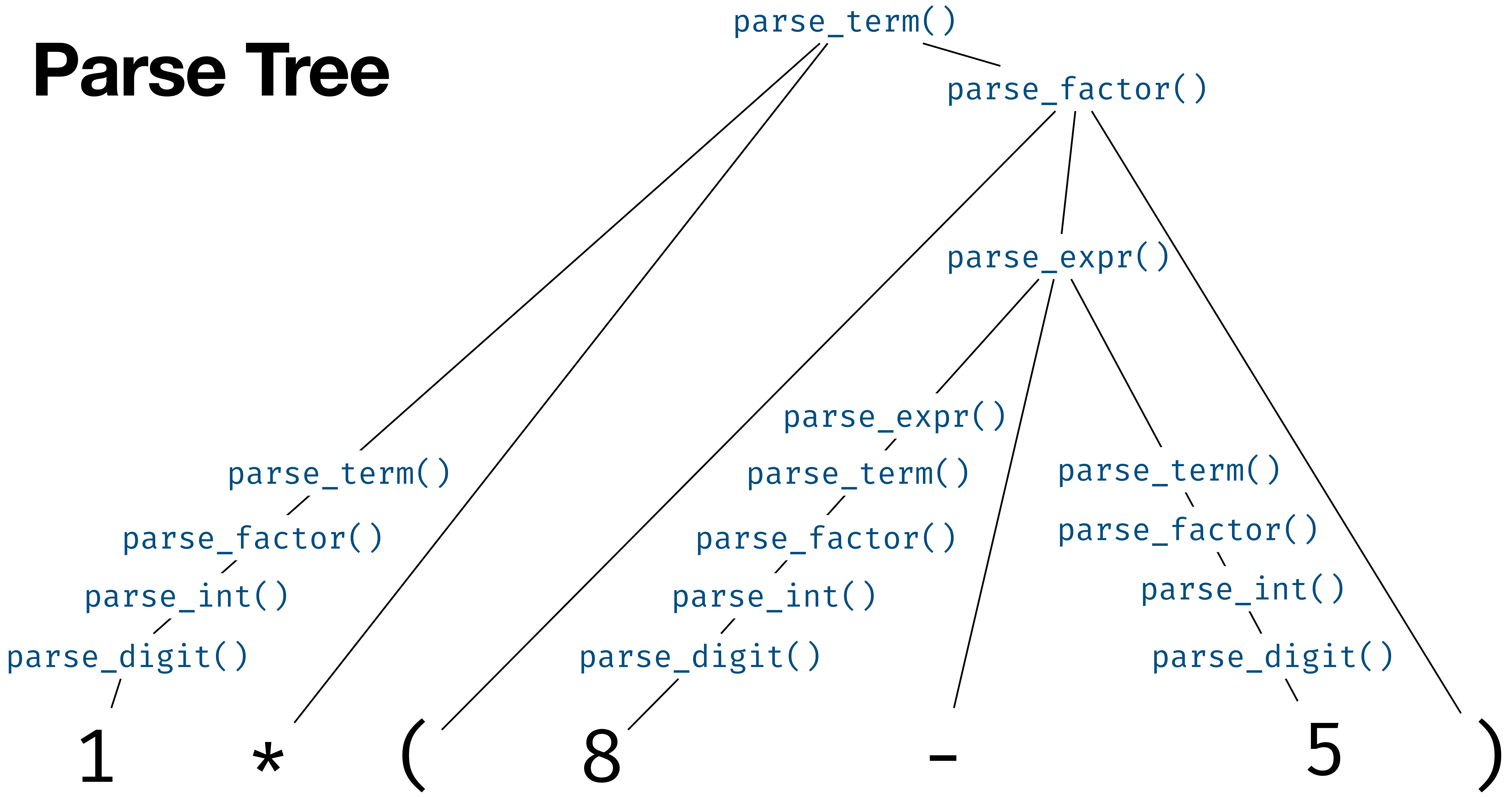
Consumption



Parse Tree

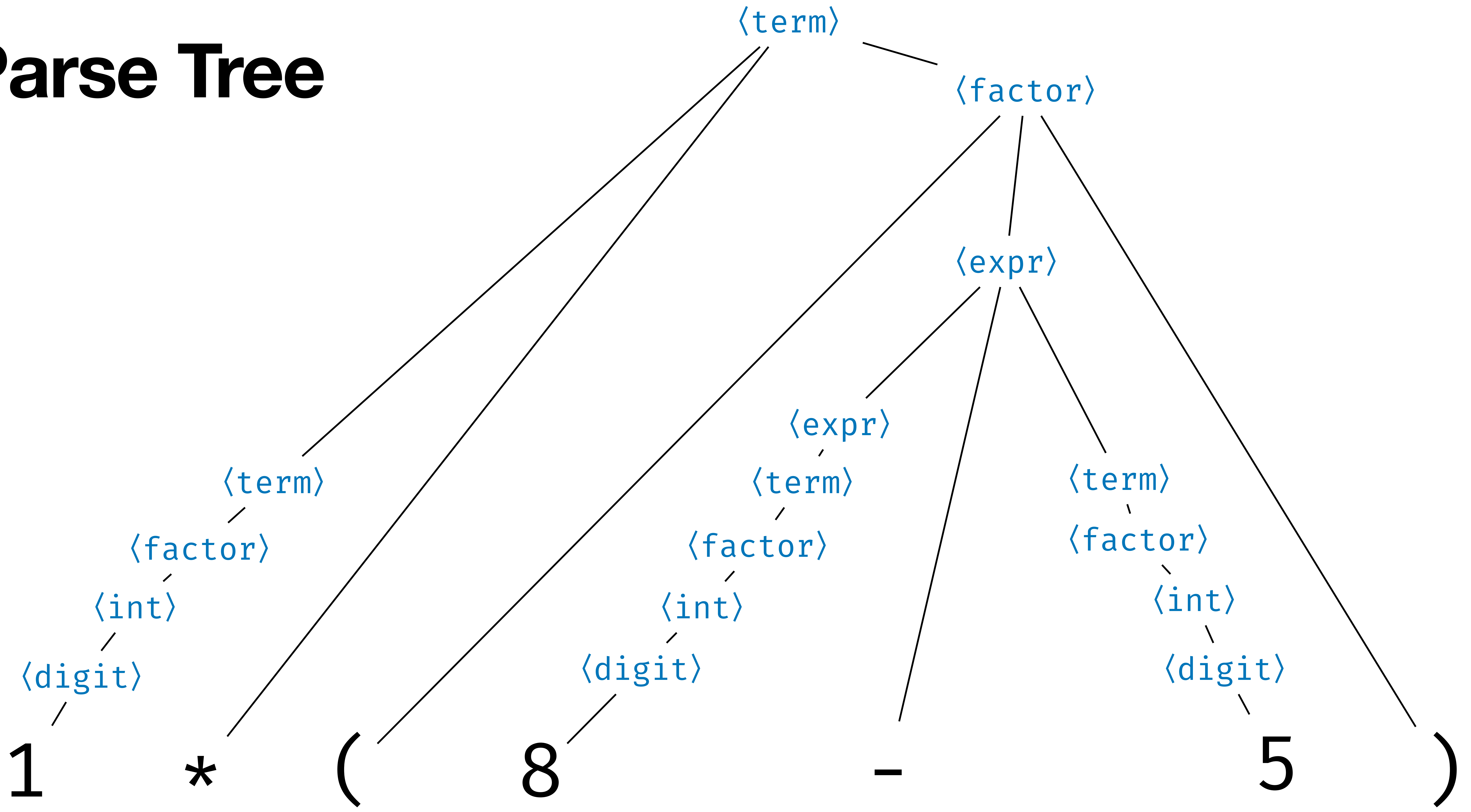

parse_term()
parse_factor()
parse_expr()
parse_digit()
parse_digit()
parse_digit()
1
*
(
8
-
5
)

Parse Tree

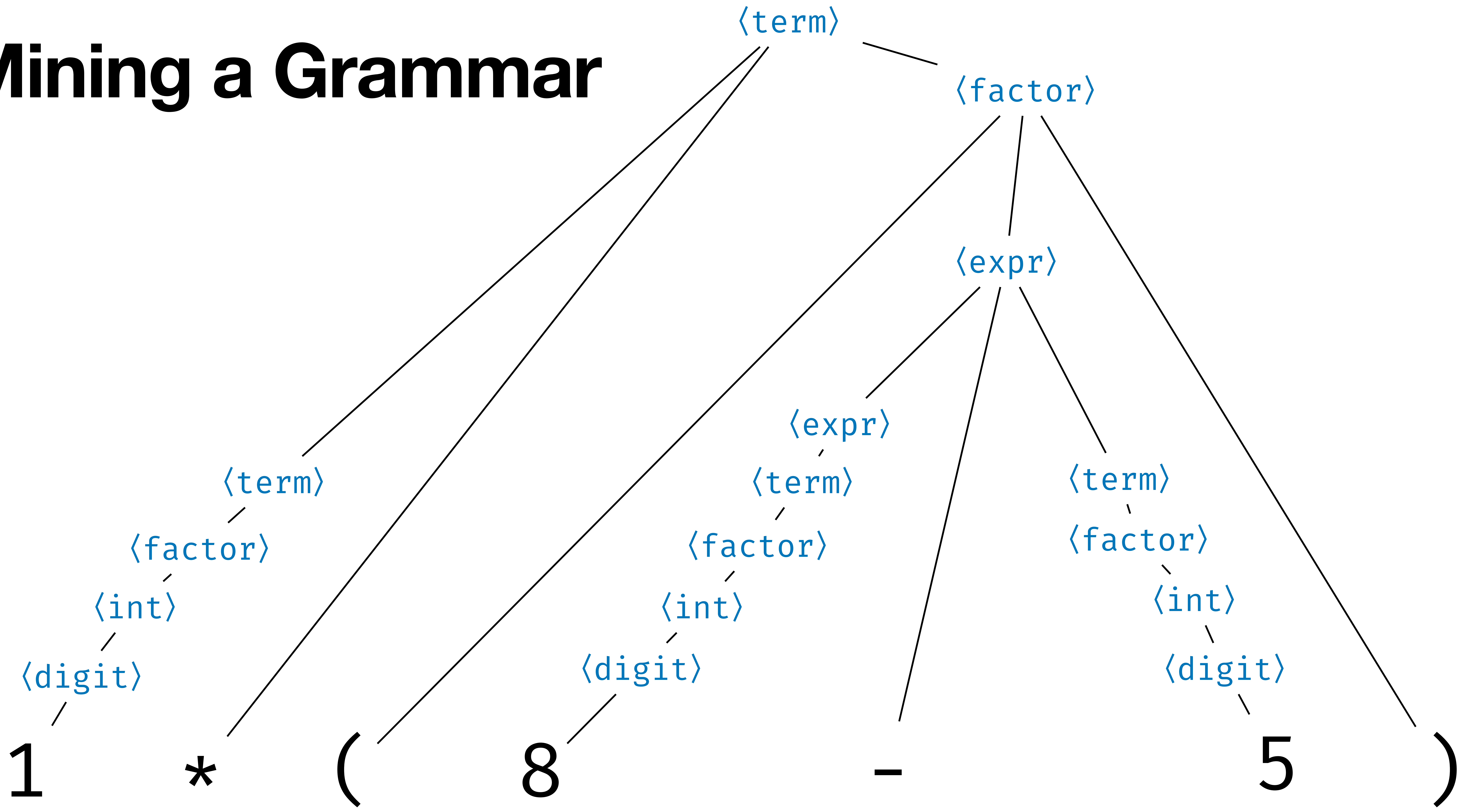


parse_term()
parse_factor()
parse_expr()
parse_expr()
parse_term()
parse_term()
parse_term()
parse_factor()
parse_factor()
parse_factor()
parse_int()
parse_int()
parse_int()
parse_digit()
parse_digit()
parse_digit()
1
*
(
8
-
5
)

Parse Tree


⟨term⟩
⟨factor⟩
⟨expr⟩
⟨expr⟩
⟨term⟩
⟨term⟩
⟨term⟩
⟨factor⟩
⟨factor⟩
⟨factor⟩
⟨int⟩
⟨int⟩
⟨int⟩
⟨digit⟩
⟨digit⟩
⟨digit⟩
1
*
(
8
-
5
)

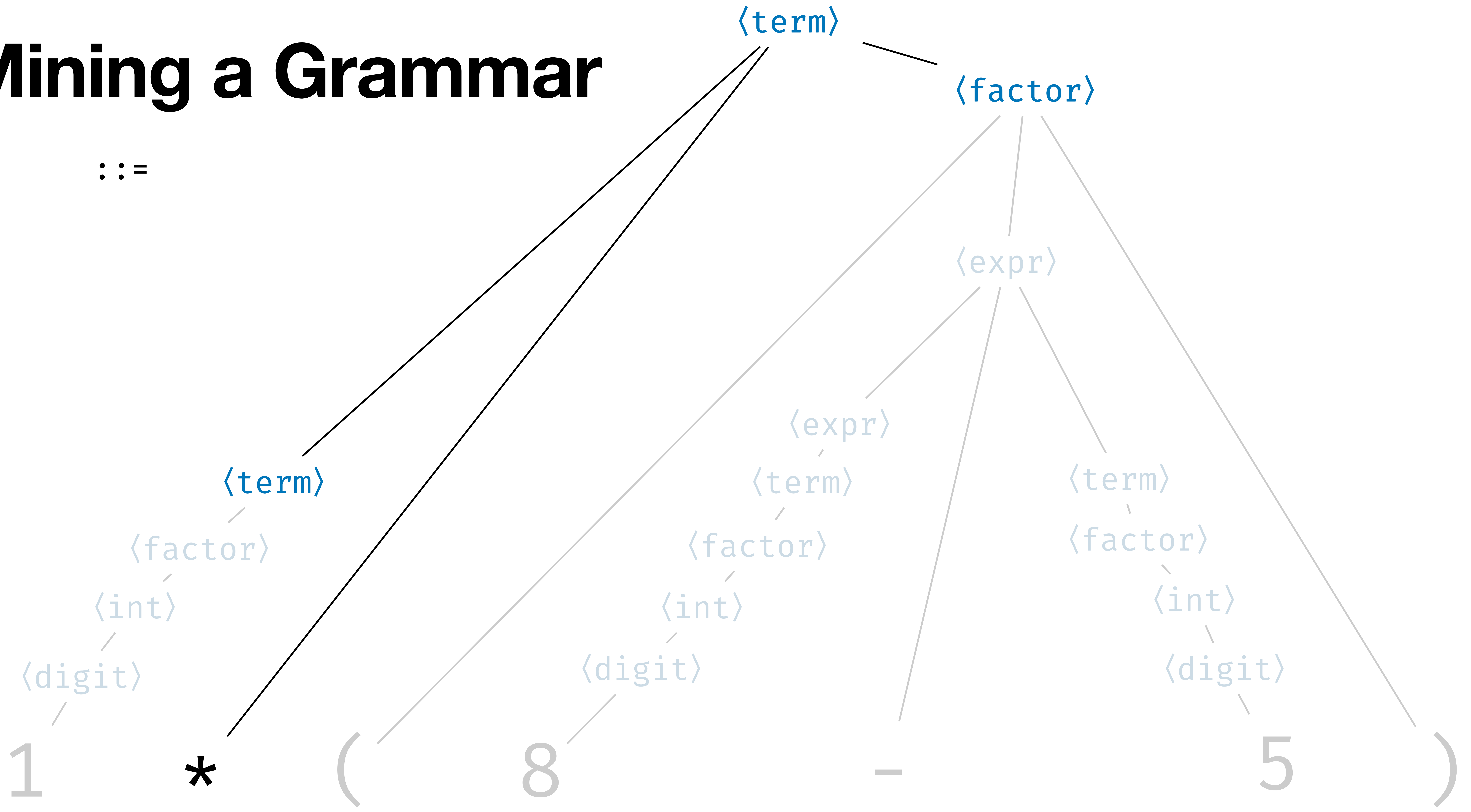
Mining a Grammar



Rahul Gopinath, Björn Mathis, and Andreas Zeller. **Mining Input Grammars from Dynamic Control Flow.** ESEC/FSE 2020.

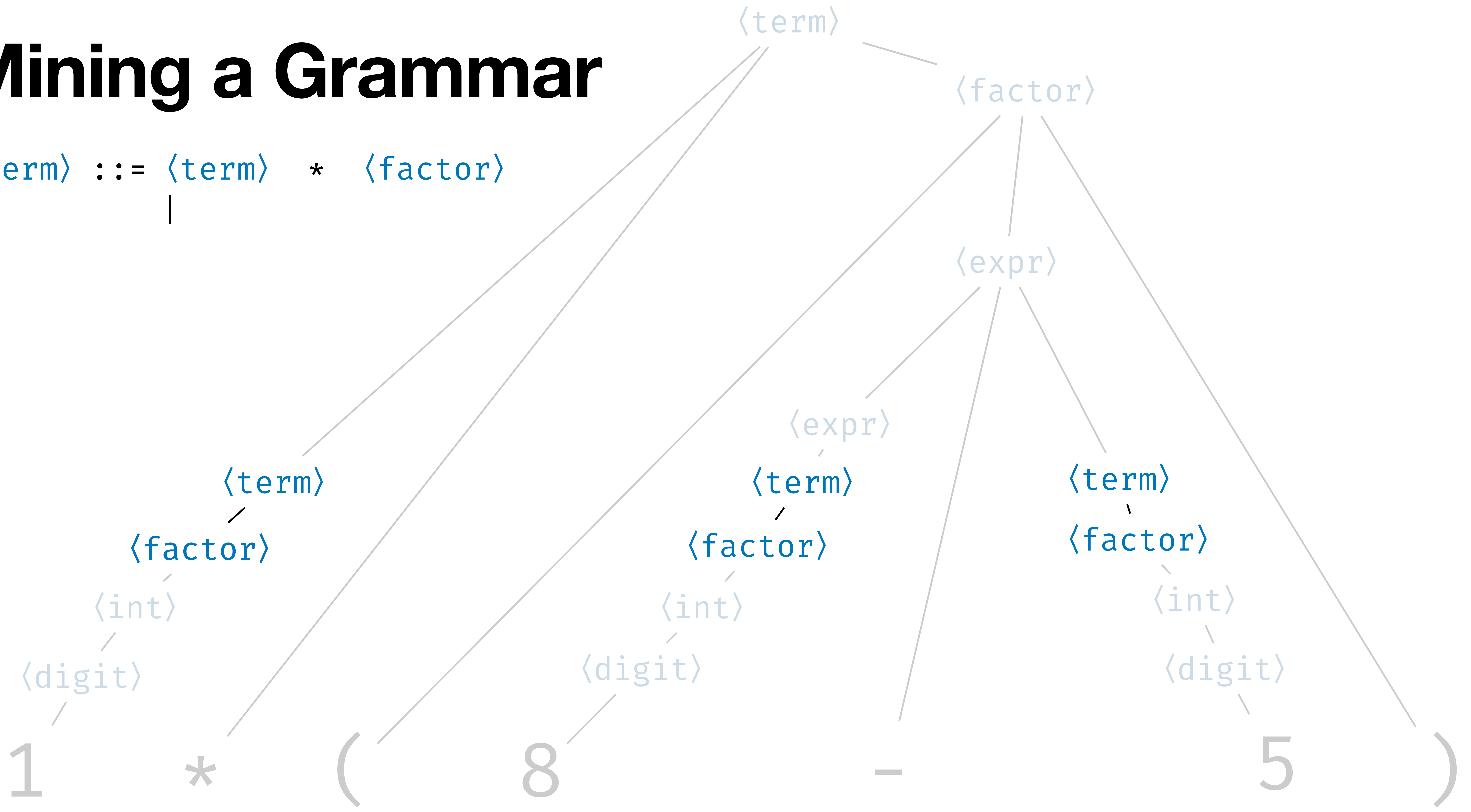
Mining a Grammar

::=



Mining a Grammar

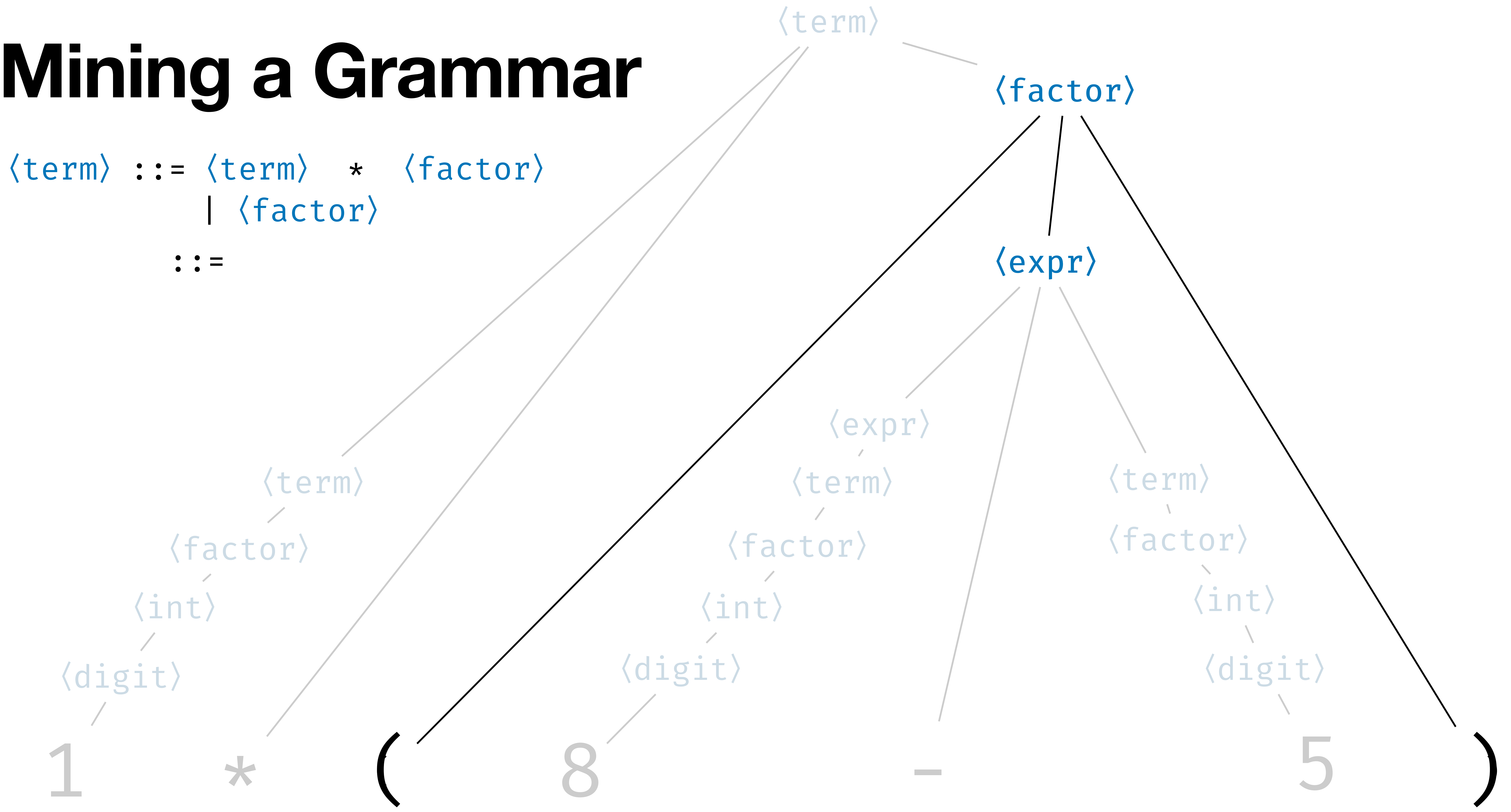
```
⟨term⟩ ::= ⟨term⟩  *  ⟨factor⟩
           |
```



Rahul Gopinath, Björn Mathis, and Andreas Zeller. **Mining Input Grammars from Dynamic Control Flow.** ESEC/FSE 2020.

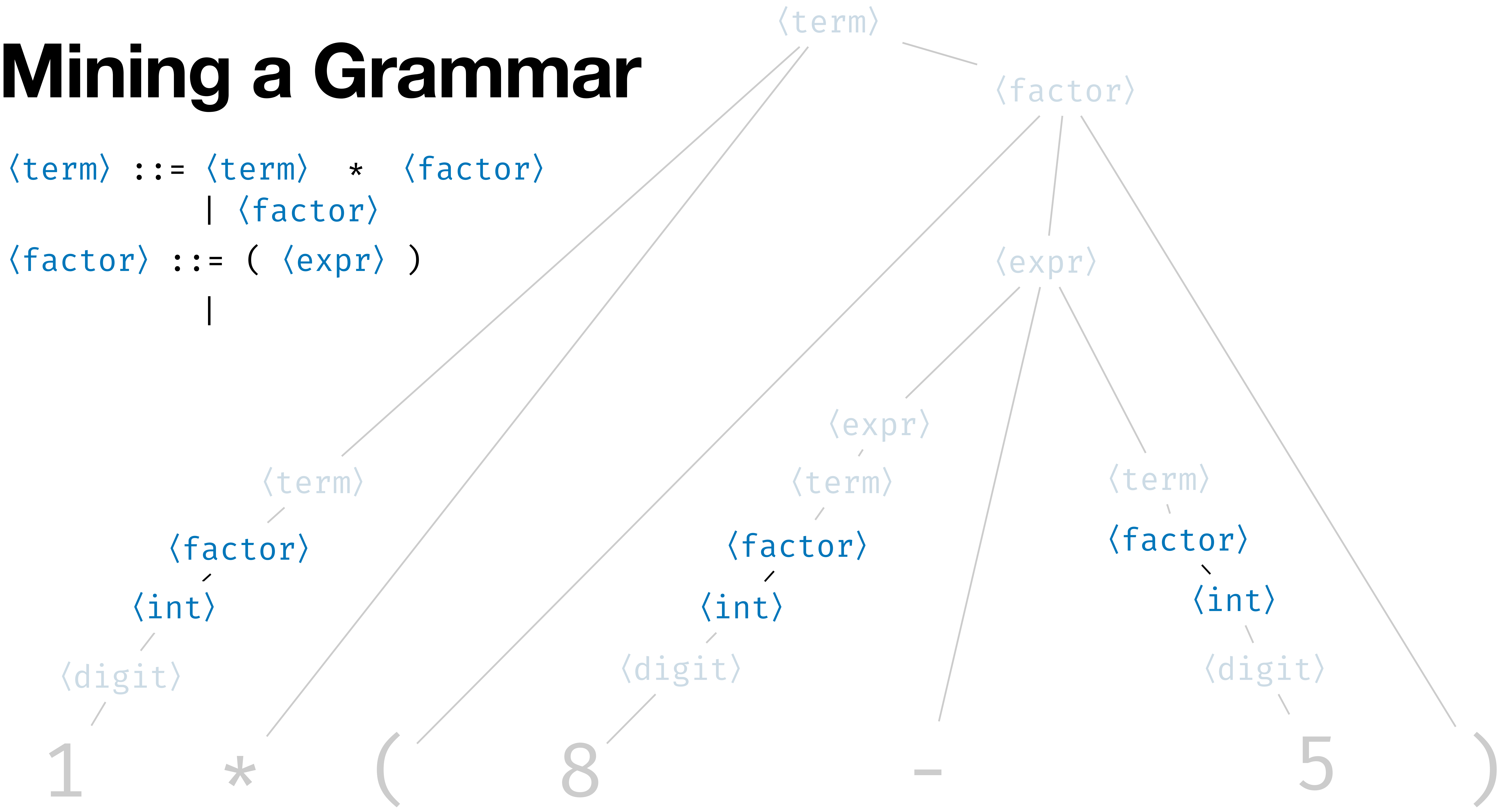
Mining a Grammar

```
⟨term⟩ ::= ⟨term⟩  *  ⟨factor⟩
         | ⟨factor⟩
       ::=
```



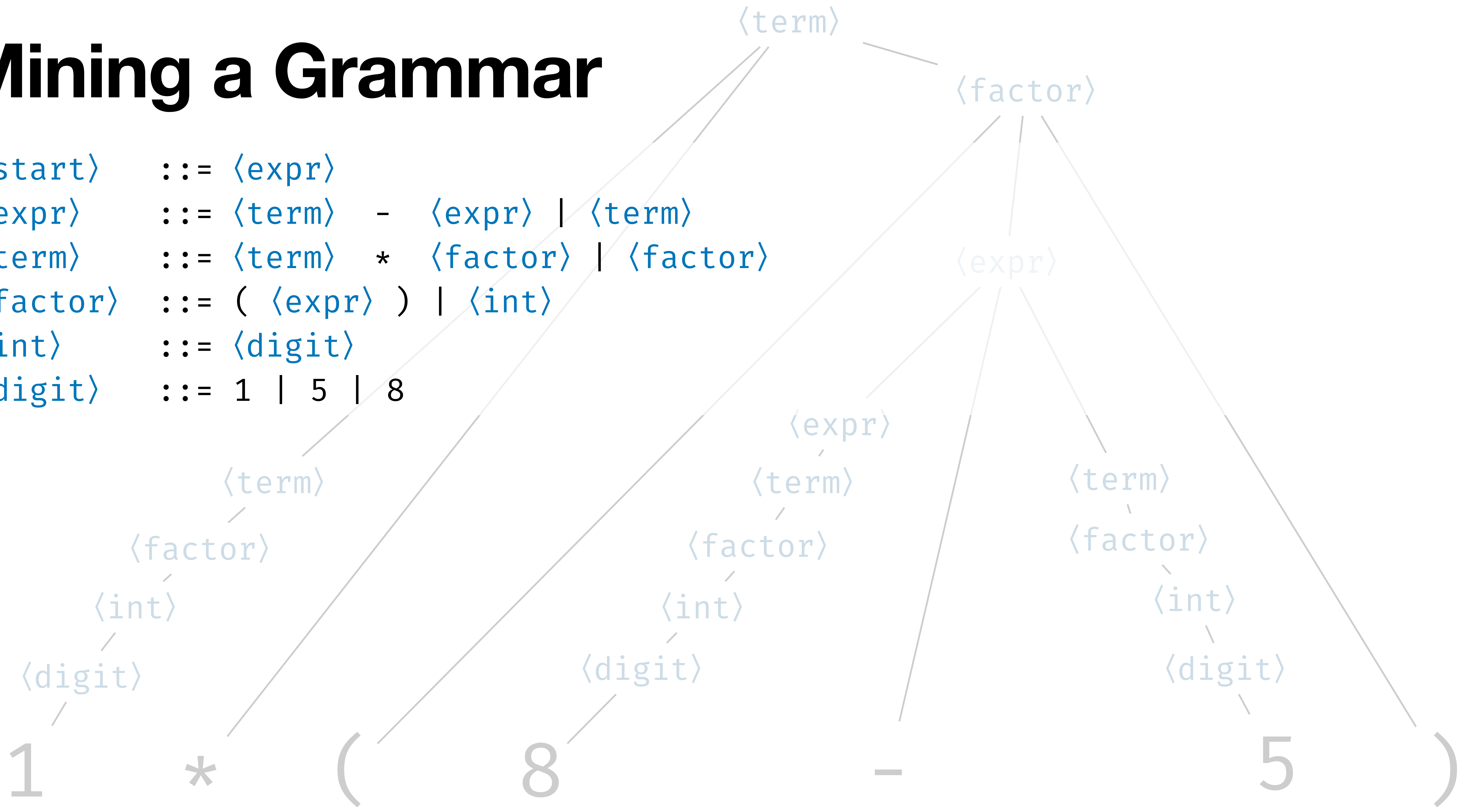
Mining a Grammar

```
⟨term⟩ ::= ⟨term⟩  *  ⟨factor⟩
          | ⟨factor⟩
⟨factor⟩ ::= ( ⟨expr⟩ )
          |
```



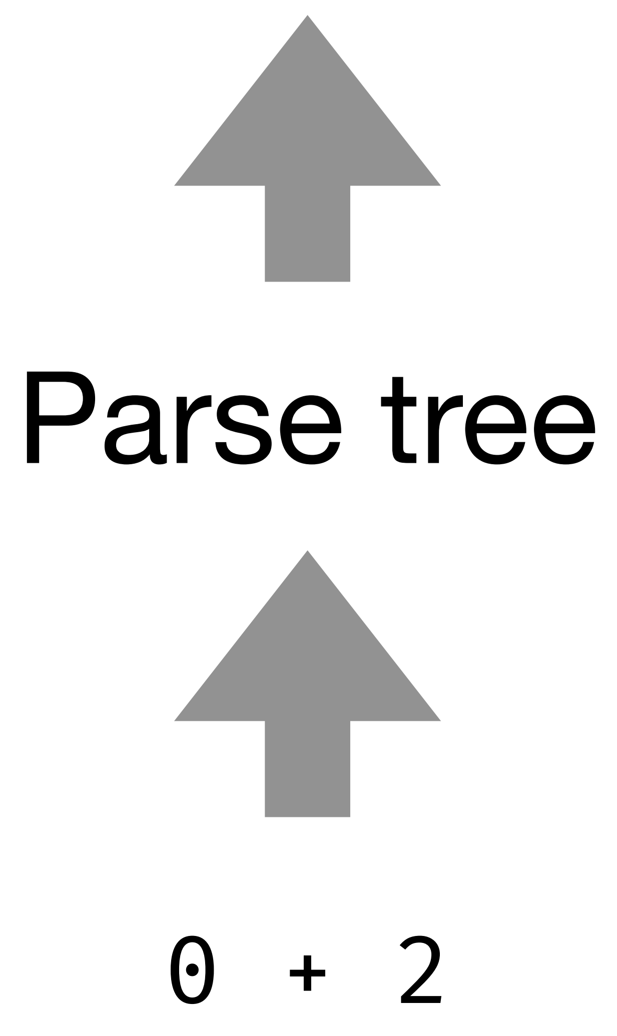
Mining a Grammar

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩  -  ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩  *  ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= ( ⟨expr⟩ ) | ⟨int⟩
⟨int⟩     ::= ⟨digit⟩
⟨digit⟩   ::= 1 | 5 | 8
```



Completing the Grammar

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩  -  ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩  *  ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= ( ⟨expr⟩ ) | ⟨int⟩
⟨int⟩     ::= ⟨digit⟩
⟨digit⟩   ::= 1 | 5 | 8
```



Completing the Grammar

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩  +  ⟨expr⟩ | ⟨term⟩  -  ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩  *  ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= ( ⟨expr⟩ ) | ⟨int⟩
⟨int⟩     ::= ⟨digit⟩
⟨digit⟩   ::=  0 | 1 | 2 | 5 | 8
```

↑

Parse tree

↑

`0 + 2`

Completing the Grammar

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩ + ⟨expr⟩ | ⟨term⟩ - ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩ * ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= ( ⟨expr⟩ ) | ⟨int⟩
⟨int⟩     ::= ⟨digit⟩
⟨digit⟩   ::= 0 | 1 | 2 | 5 | 8
```

Parse tree

```
0 + 2
+3 / -46.79
```

Rahul Gopinath, Björn Mathis, and Andreas Zeller. **Mining Input Grammars from Dynamic Control Flow.** ESEC/FSE 2020.

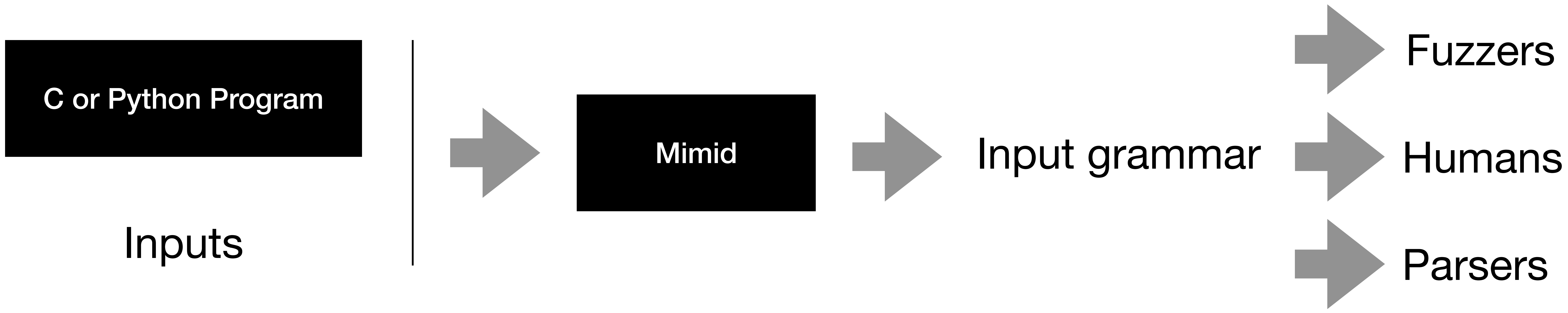
Completing the Grammar

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩  +  ⟨expr⟩ | ⟨term⟩  -  ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩  *  ⟨factor⟩ | ⟨term⟩  /  ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= + ⟨factor⟩ | - ⟨factor⟩ | ( ⟨expr⟩ ) | ⟨int⟩ | ⟨int⟩ . ⟨int⟩
⟨int⟩     ::= ⟨digit⟩  ⟨int⟩ | ⟨digit⟩
⟨digit⟩   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Parse tree

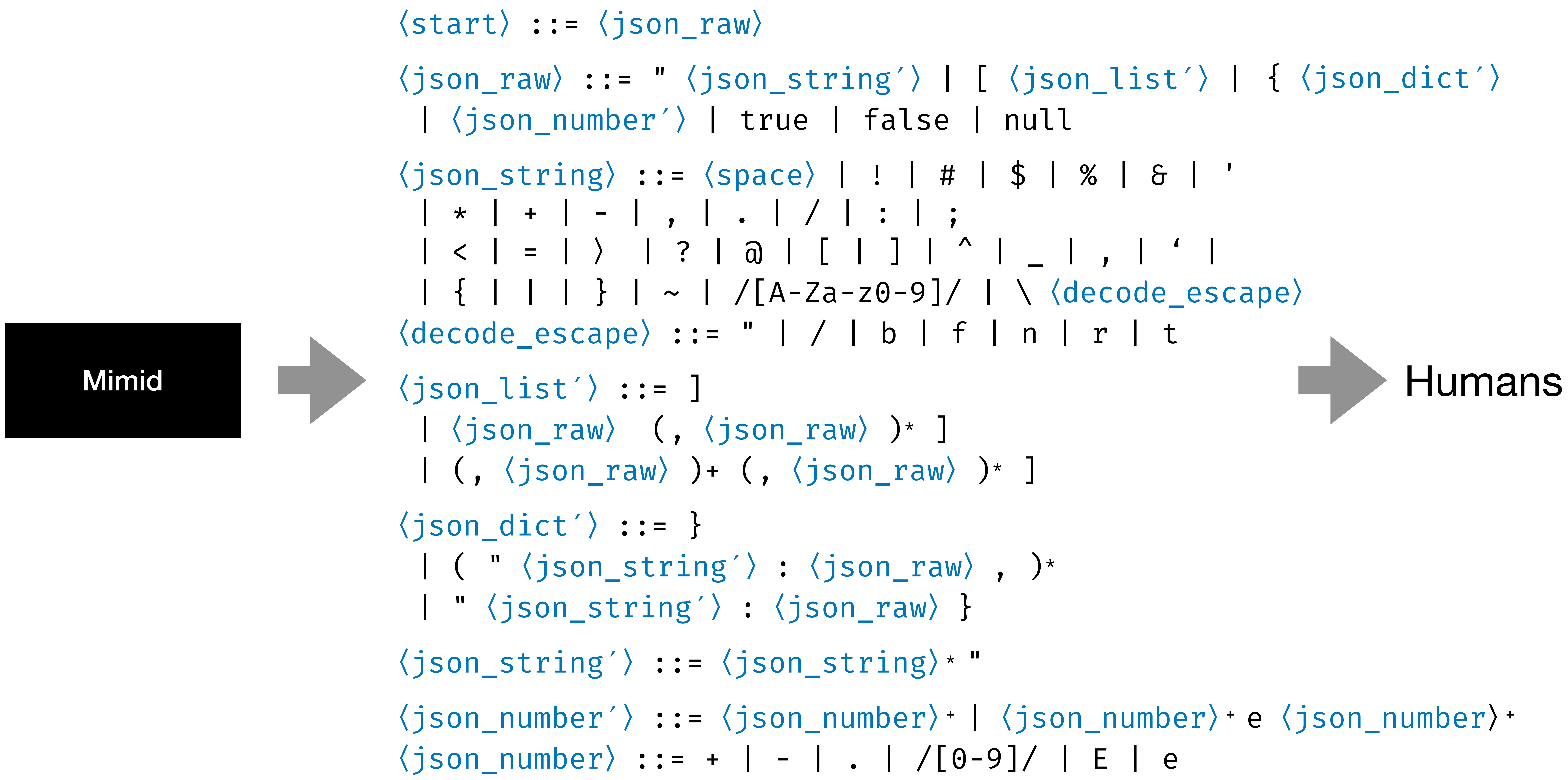
```
 0 + 2
+3 / -46.79
```

Mimid: A Grammar Miner



- **Evaluation** on CGI, URL, JSON, TinyC, JavaScript
- Mined grammars **cover 98%** of the actual language
- Mined grammars **well-structured** and **highly readable**

Rahul Gopinath, Björn Mathis, and Andreas Zeller. **Mining Input Grammars from Dynamic Control Flow.** ESEC/FSE 2020.



```
⟨start⟩ ::= ⟨json_raw⟩

⟨json_raw⟩ ::= " ⟨json_string′⟩ | [ ⟨json_list′⟩ | { ⟨json_dict′⟩
 | ⟨json_number′⟩ | true | false | null

⟨json_string⟩ ::= ⟨space⟩ | ! | # | $ | % | & | '
 | * | + | - | , | . | / | : | ;
 | < | = | ⟩  | ? | @ | [ | ] | ^ | _ | , | ‘ |
 | { | | | } | ~ | /[A-Za-z0-9]/ | \ ⟨decode_escape⟩
⟨decode_escape⟩ ::= " | / | b | f | n | r | t

⟨json_list′⟩ ::= ]
 | ⟨json_raw⟩  (, ⟨json_raw⟩ )* ]
 | (, ⟨json_raw⟩ )+ (, ⟨json_raw⟩ )* ]

⟨json_dict′⟩ ::= }
 | ( " ⟨json_string′⟩ : ⟨json_raw⟩ , )*
 | " ⟨json_string′⟩ : ⟨json_raw⟩ }

⟨json_string′⟩ ::= ⟨json_string⟩* "

⟨json_number′⟩ ::= ⟨json_number⟩+ | ⟨json_number⟩+ e ⟨json_number⟩+
⟨json_number⟩ ::= + | - | . | /[0-9]/ | E | e
```

Taming Fuzzers

```
⟨start⟩ ::= ⟨json_raw⟩

⟨json_raw⟩ ::= " ⟨json_string′⟩ | [ ⟨json_list′⟩ | { ⟨json_dict′⟩
 | ⟨json_number′⟩ | true | false | null

⟨json_string⟩ ::= ⟨space⟩ | ! | # | $ | % | & | '
 | * | + | - | , | . | / | : | ;
 | < | = | ⟩  | ? | @ | [ | ] | ^ | _ | , | ‘ |
 | { | | | } | ~ | /[A-Za-z0-9]/ | \ ⟨decode_escape⟩
⟨decode_escape⟩ ::= " | / | b | f | n | r | t

⟨json_list′⟩ ::= ]
 | ⟨json_raw⟩  (, ⟨json_raw⟩ )* ]
 | (, ⟨json_raw⟩ )+ (, ⟨json_raw⟩ )* ]

⟨json_dict′⟩ ::= }
 | ( " ⟨json_string′⟩ : ⟨json_raw⟩ , )*
 | " ⟨json_string′⟩ : ⟨json_raw⟩ }

⟨json_string′⟩ ::= ⟨json_string⟩* "

⟨json_number′⟩ ::= ⟨json_number⟩+ | ⟨json_number⟩+ e ⟨json_number⟩+
⟨json_number⟩ ::= + | - | . | /[0-9]/ | E | e
```

Fuzzers

Humans

Parsers

Taming Fuzzers

```
⟨start⟩ ::= ⟨json_raw⟩

⟨json_raw⟩ ::= " ⟨json_string′⟩ | [ ⟨json_list′⟩ | { ⟨json_dict′⟩
 | ⟨json_number′⟩ | true | false | null

⟨json_string⟩ ::= ⟨space⟩ | ! | # | $ | % | & | '
 | * | + | - | , | . | / | : | ;
 | < | = | ⟩  | ? | @ | [ | ] | ^ | _ | , | ‘ |
 | { | | | } | ~ | /[A-Za-z0-9]/ | \ ⟨decode_escape⟩
⟨decode_escape⟩ ::= " | / | b | f | n | r | t

⟨json_list′⟩ ::= ]
 | ⟨json_raw⟩  (, ⟨json_raw⟩ )* ]
 | (, ⟨json_raw⟩ )+ (, ⟨json_raw⟩ )* ]

⟨json_dict′⟩ ::= }
 | ( " ⟨json_string′⟩ : ⟨json_raw⟩ , )*
 | " ⟨json_string′⟩ : ⟨json_raw⟩ }

⟨json_string′⟩ ::= ⟨json_string⟩* "

⟨json_number′⟩ ::= ⟨json_number⟩+ | ⟨json_number⟩+ e ⟨json_number⟩+
⟨json_number⟩ ::= + | - | . | /[0-9]/ | E | e
```

Humans

Taming Fuzzers

```
⟨start⟩ ::= ⟨json_raw⟩

⟨json_raw⟩ ::= " ⟨json_string′⟩ | [ ⟨json_list′⟩ | { ⟨json_dict′⟩
 | ⟨json_number′⟩ | true | false | null

⟨json_string⟩ ::= ⟨space⟩ | ! | # | $ | % | & | '
 | * | + | - | , | . | / | : | ;
 | < | = | ⟩  | ? | @ | [ | ] | ^ | _ | , | ‘ |
 | { | | | } | ~ | /[A-Za-z0-9]/ | \ ⟨decode_escape⟩
⟨decode_escape⟩ ::= " | / | b | f | n | r | t

⟨json_list′⟩ ::= ]
 | ⟨json_raw⟩  (, ⟨json_raw⟩ )* ]
 | (, ⟨json_raw⟩ )+ (, ⟨json_raw⟩ )* ]

⟨json_dict′⟩ ::= }
 | ( " ⟨json_string′⟩ : ⟨json_raw⟩ , )*
 | " ⟨json_string′⟩ : ⟨json_raw⟩ }

⟨json_string′⟩ ::= ⟨json_string⟩* "

⟨json_number′⟩ ::= ⟨json_number⟩+ | ⟨json_number⟩+ e ⟨json_number⟩+
⟨json_number⟩ ::= + | - | . | /[0-9]/ | E | e
```

Humans

Taming Fuzzers

Fuzzer

```
⟨start⟩ ::= ⟨json_raw⟩

⟨json_raw⟩ ::= " ⟨json_string′⟩ | 10% [ ⟨json_list′⟩ | 50% { ⟨json_dict′⟩
 | ⟨json_number′⟩ | true | false | null

⟨json_string⟩ ::= ⟨space⟩ | ! | # | $ | % | & | '
 | * | + | - | , | . | / | : | ;
 | < | = | 〉 | ? | @ | [ | ] | ^ | _ | , | ‘ |
 | { | | | } | ~ | /[A-Za-z0-9]/ | \ ⟨decode_escape⟩
⟨decode_escape⟩ ::= " | / | b | f | n | r | t

⟨json_list′⟩ ::= ]
 | ⟨json_raw⟩  (, ⟨json_raw⟩ )* ]
 | (, ⟨json_raw⟩ )+ (, ⟨json_raw⟩ )* ]

⟨json_dict′⟩ ::= }
 | ( " ⟨json_string′⟩ : ⟨json_raw⟩ , )*
 | " ⟨json_string′⟩ : ⟨json_raw⟩ }

⟨json_string′⟩ ::= ⟨json_string⟩* "

⟨json_number′⟩ ::= ⟨json_number⟩+ | ⟨json_number⟩+ e ⟨json_number⟩+
⟨json_number⟩ ::= + | - | . | /[0-9]/ | E | e
```

Humans

Taming Fuzzers

Fuzzer

```
⟨start⟩ ::= ⟨json_raw⟩

⟨json_raw⟩ ::= " ⟨json_string′⟩ | [ ⟨json_list′⟩ | { ⟨json_dict′⟩
  | ⟨json_number′⟩ | true | false | null

⟨json_string⟩ ::= ⟨space⟩ | ! | # | $ | % | & | '
  | * | + | - | , | . | / | : | ;
  | < | = | 〉 | ? | @ | [ | ] | ^ | _ | , | ‘ |
  | { | | | } | ~ | /[A-Za-z0-9]/ | \ ⟨decode_escape⟩
⟨decode_escape⟩ ::= " | / | b | f | n | r | t

⟨json_list′⟩ ::= ]
  | ⟨json_raw⟩  (, ⟨json_raw⟩ )* ]
  | (, ⟨json_raw⟩ )+ (, ⟨json_raw⟩ )* ]

⟨json_dict′⟩ ::= }
  | ( " ⟨json_string′⟩ : ⟨json_raw⟩ , )*
  | " ⟨json_string′⟩ : ⟨json_raw⟩ }

⟨json_string′⟩ ::= ⟨json_string⟩* " | '; DROP TABLE students"

⟨json_number′⟩ ::= ⟨json_number⟩+ | ⟨json_number⟩+ e ⟨json_number⟩+
⟨json_number⟩ ::= + | - | . | /[0-9]/ | E | e
```

Humans

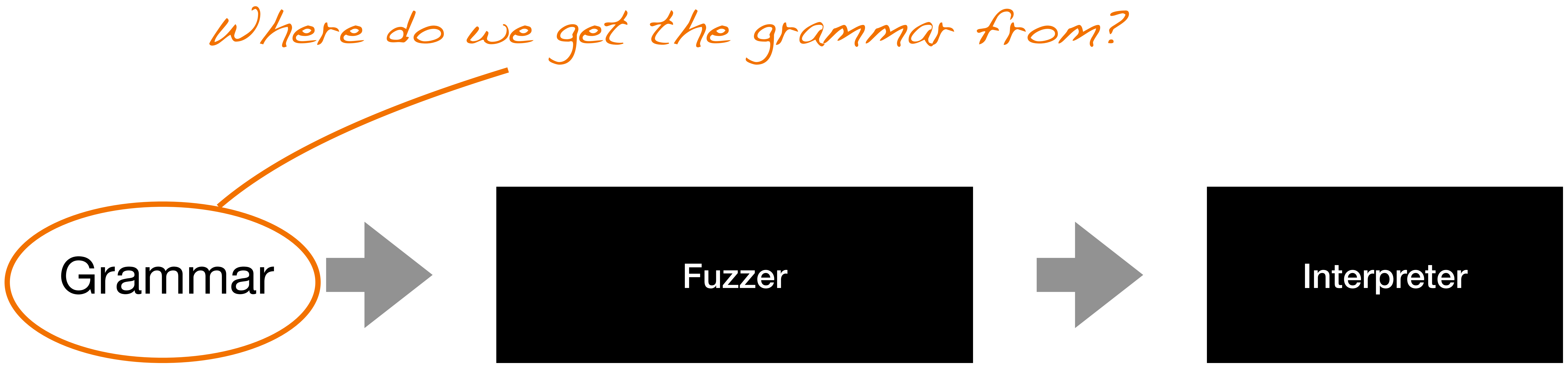
Taming Fuzzers

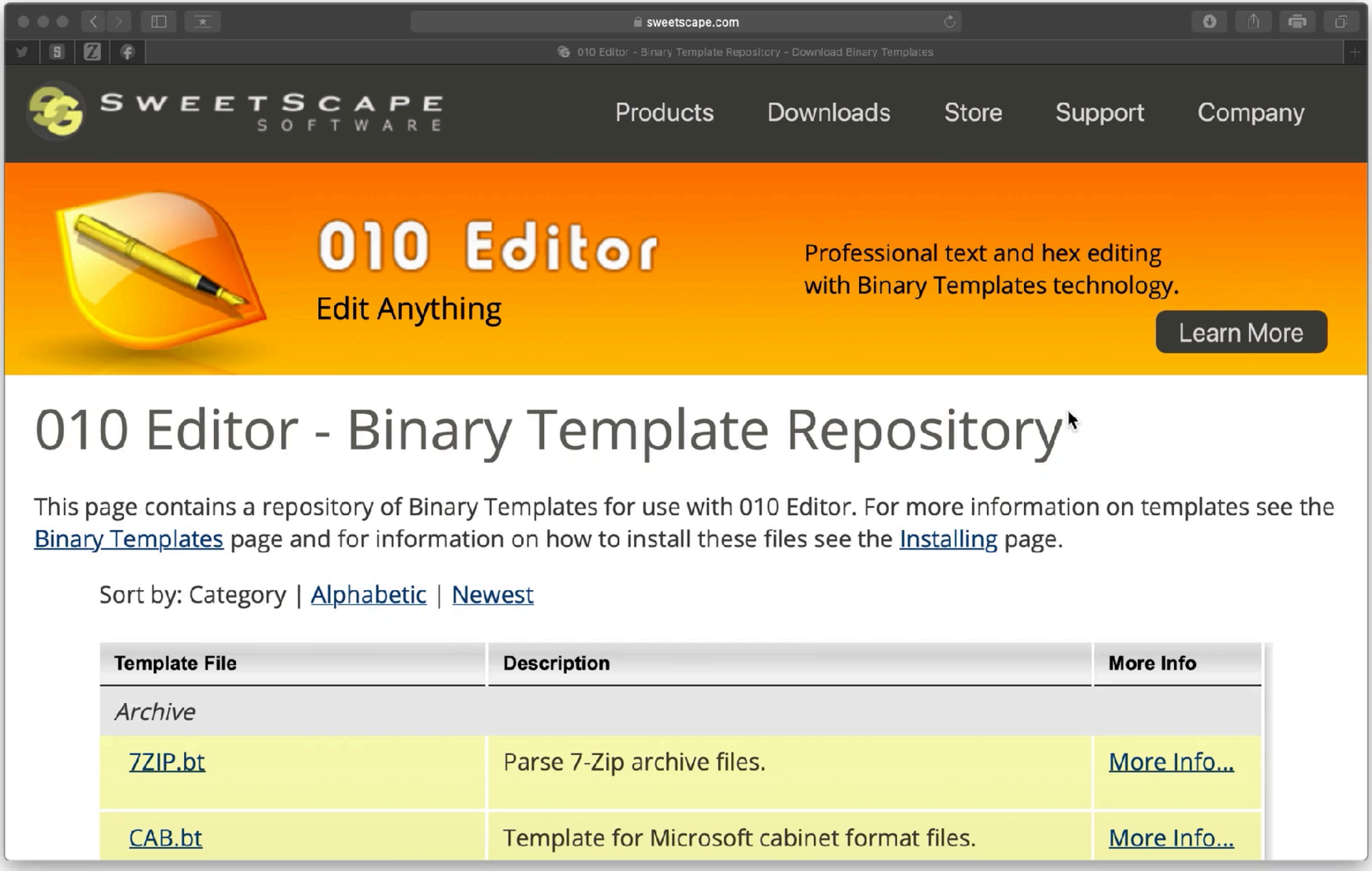
Fuzzer

```
{ "": "'; DROP TABLE STUDENTS" , "/h?O ": [ ] , "": "" , "x": false ,
"": null }
{ "": ".qF" , "": "'; DROP TABLE STUDENTS", "": 47 }
{ "7": { "y": "" }, "": false, "X": "N7|:", "": [ true ], "": [ ], "": {
} }
{ "": [ ], "9z6}l": null }
{ "#": false, "D": { "": true }, "t": 90, "g": [ "'; DROP TABLE
STUDENTS" ], "": [ false ], "=R5": [ ], " ": "'; DROP TABLE STUDENTS",
"`l": { "": "?'L", "E": null, "": [ 70.3076998940e6 ], "Ju": true } }
{ "": true, "": "%7y", "!": false, "": true, "": { "": [ ], "":
-096860E+0, "U": 0E-5 } }
{ "'ia": [ true, "'; DROP TABLE STUDENTS", null, [ false, { } ],
true ] }
{ "@meB1T]": 0.0, "": null, "": true, "7": 208.00E4, "": true, "":
70e+10, "": "", "5zJ": [ false, false ] }
{ "": "H", "d;": "'; DROP TABLE STUDENTS" }
{ "Y!Z": ".i", "h": "'; DROP TABLE STUDENTS" }
{ "": -64.0e-06, "": [ { "p[f": false, "": "'; DROP TABLE STUDENTS",
"m": [ ], "": true, "8D": -0, "@R": true } ] }
{ "": "'; DROP TABLE STUDENTS" }
{ "r": "'; DROP TABLE STUDENTS", "zJzjT": 6.59 }
{ "oh": false }
{ "c": [ false, 304e+008520, null, false, "'; DROP TABLE STUDENTS",
"m[MD" , [ false ] ] }
```

Performance by highly trained professional. **Do not try this** at home, your university or anywhere else.

Fuzzing with Grammars





sweetscape.com

010 Editor - Binary Template Repository - Download Binary Templates

010 Editor - Binary Template Repository

This page contains a repository of Binary Templates for use with 010 Editor. For more information on templates see the Binary Templates page and for information on how to install these files see the Installing page.

Sort by: Category | Alphabetic | Newest

Template File	Description	More Info
Archive		
7ZIP.bt	Parse 7-Zip archive files.	More Info...
CAB.bt	Template for Microsoft cabinet format files.	More Info...

Existing Input Format Specifications

```
//------------------------------------------------
//--- 010 Editor Binary Template
//
//      File: PNG.bt
//   Authors: Kevin O. Grover, RCS, Mister Wu
//   Version: 2.3
//   Purpose: Parse PNG (Portable Network Graphics) and APNG (Animated Portable Network Graphics) image files.
//  Category: Image
// File Mask: *.png,*.apng
//  ID Bytes: 89 50 4E 47 //%PNG
//   History:
//   2.3   2018-08-17 K. Grover: Uniform type names.  Formatting/comments.
//   2.2   2017-10-20 Mister Wu: Initial support of APNG chunks: acTL, fcTL, fdAT.
//   2.1   2017-08-31 K. Grover: Better colors.  Alternate chunk colors.  Cleaned up messages.
//   2.0   2016-02-10 SweetScape Software: Merged in extra chunks from PNG12Template.bt, updated header for
repository submission.
//   1.1   2009-02-23 K. Grover: Decode IHDR and tEXt chunks.
//   1.0.1 2005-06-29 K. Grover: Fixed typos in comments.
//   1.0   2005-05-11 K. Grover: Initial version.
//
// This template was written to the PNG 1.2 Specification:
//     http://www.libpng.org/pub/png/spec/1.2/
//
// It includes chunks described in the APNG 1.0 Specification:
//     https://wiki.mozilla.org/APNG_Specification
//
// Note however, that it does not check nor parse all chunk subdata, so it
// should work with all future PNG specifications.
//
// Possible caveat: PNG encourages the chunk type to be mapped to
// strings of the form "[a-zA-Z]{4}".  However, it's not a requirement.
// PNG Data types
typedef struct {
    uint16 btPngSignature[4] <format=hex>;
} PNG_SIGNATURE;

typedef enum <byte> pngColorSpaceType {
    GrayScale = 0,
    TrueColor = 2,
    Indexed = 3,
    AlphaGrayScale = 4,
    AlphaTrueColor = 6
} PNG_COLOR_SPACE_TYPE;

// Compression Methods
typedef enum <byte> pngCompressionMethod {
    Deflate = 0
} PNG_COMPR_METHOD;

// Filter Methods
typedef enum <byte> pngFilterMethod {
    AdaptiveFiltering = 0
} PNG_FILTER_METHOD;

// Interlace Methods
typedef enum <byte> pngInterlaceMethod {
    NoInterlace = 0,
    Adam7Interlace = 1
} PNG_INTERLACE_METHOD;

typedef struct {
    byte btRed <format=hex>;
    byte btGreen <format=hex>;
    byte btBlue <format=hex>;
} PNG_PALETTE_PIXEL;

typedef struct {
    uint32 x;
    uint32 y;
```

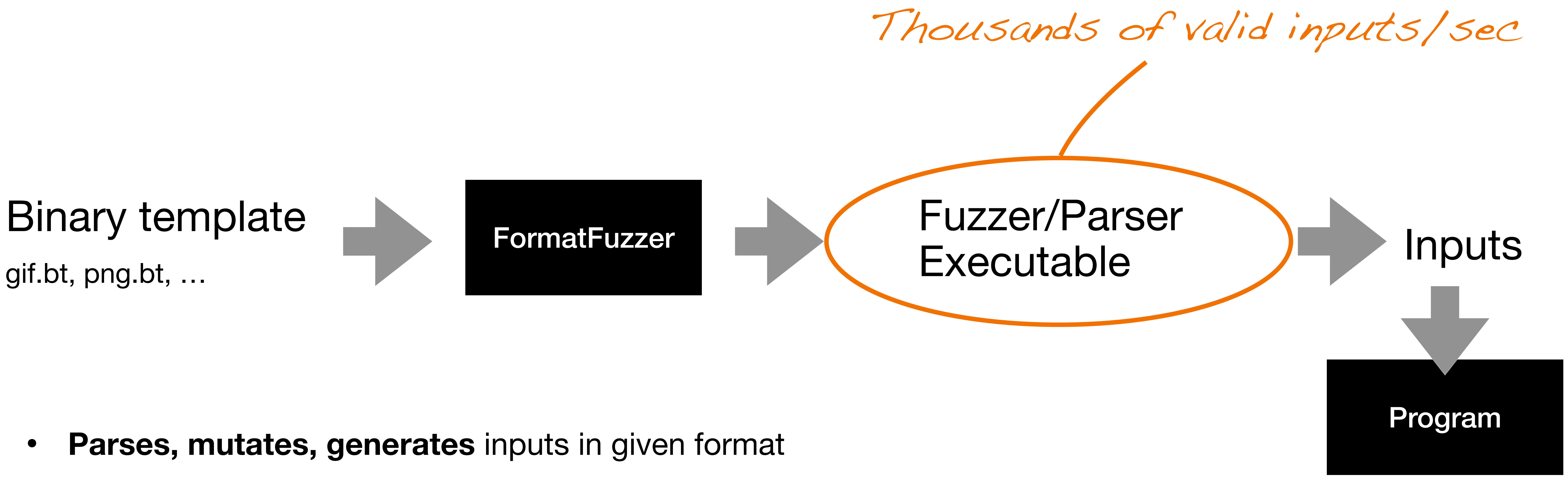
```
typedef struct {
    string label;                            // to the first NULL (including)
    char   data[length - Strlen(label) - 1]; // rest of the data
} PNG_CHUNK_TEXT <read=readtEXt>;

string readtEXt(local PNG_CHUNK_TEXT &text) {
    local string s;
    SPrintf(s, "%s = %s", text.label, text.data);
    return s;
}

struct PNG_CHUNK_PLTE (int32 chunkLen) {
    PNG_PALETTE_PIXEL plteChunkData[chunkLen/3];
};

struct PNG_CHUNK_CHRM {
    PNG_POINT white;
    PNG_POINT red;
    PNG_POINT green;
    PNG_POINT blue;
};

struct PNG_CHUNK_SRGB {
    PNG_SRGB_CHUNK_DATA srgbChunkData;
};

struct PNG_CHUNK_IEXT (int32 chunkLen) {
    string iextIdChunkData;
    byte iextCompressionFlag;
    PNG_COMPR_METHOD iextComprMethod;
    string iextLanguageTag;
    string iextTranslatedKeyword;
    char iextValChunkData[chunkLen -
                              Strlen(iextIdChunkData) -1 -
                              Strlen(iextLanguageTag) -1 -
                              Strlen(iextTranslatedKeyword) -1 -
                              2];
};
```

```
// Generic Chunks
typedef struct {
    uint32  length;                      // Number of data bytes (not including length,type, or crc)
    local int64 pos_start = FTell();
    CTYPE   type <fgcolor=cDkBlue>;        // Type of chunk
    if (type.cname == "IHDR")
        PNG_CHUNK_IHDR    ihdr;
    else if (type.cname == "tEXt")
        PNG_CHUNK_TEXT    text;
    else if (type.cname == "PLTE")
        PNG_CHUNK_PLTE    plte(length);
    else if (type.cname == "cHRM")
        PNG_CHUNK_CHRM    chrm;
    else if (type.cname == "sRGB")
        PNG_CHUNK_SRGB    srgb;
    else if (type.cname == "iEXt")
        PNG_CHUNK_IEXT    iext(length);
    else if (type.cname == "zEXt")
        PNG_CHUNK_ZEXT    zext(length);
    else if (type.cname == "tIME")
        PNG_CHUNK_TIME    time;
    else if (type.cname == "pHYs")
        PNG_CHUNK_PHYS    phys;
    else if (type.cname == "bKGD")
        PNG_CHUNK_BKGD    bkgd(chunk[0].ihdr.color_type);
    else if (type.cname == "sBIT")
        PNG_CHUNK_SBIT    sbit(chunk[0].ihdr.color_type);
    else if (type.cname == "sPLT")
        PNG_CHUNK_SPLT    splt(length);
    else if (type.cname == "acTL")
        PNG_CHUNK_ACTL    actl;
    else if (type.cname == "fcTL")
        PNG_CHUNK_FCTL    fctl;
    else if (type.cname == "fdAT")
        PNG_CHUNK_FDAT    fdat;
    else if( length > 0 )
        ubyte   data[length];       // Data (or not present)
    local int64 data_size = FTell() - pos_start;
    else if (type.cname == "PLTE")
        PNG_CHUNK_PLTE    plte(length);
    else if (type.cname == "cHRM")
        PNG_CHUNK_CHRM    chrm;
    else if (type.cname == "sRGB")
        PNG_CHUNK_SRGB    srgb;
    else if (type.cname == "iEXt")
        PNG_CHUNK_IEXT    iext(length);
    else if (type.cname == "zEXt")
        PNG_CHUNK_ZEXT    zext(length);
    else if (type.cname == "tIME")
        PNG_CHUNK_TIME    time;
    else if (type.cname == "pHYs")
        PNG_CHUNK_PHYS    phys;
    else if (type.cname == "bKGD")
        PNG_CHUNK_BKGD    bkgd(chunk[0].ihdr.color_type);
    else if (type.cname == "sBIT")
        PNG_CHUNK_SBIT    sbit(chunk[0].ihdr.color_type);
    else if (type.cname == "sPLT")
        PNG_CHUNK_SPLT    splt(length);
    else if (type.cname == "acTL")
        PNG_CHUNK_ACTL    actl;
    else if (type.cname == "fcTL")
        PNG_CHUNK_FCTL    fctl;
    else if (type.cname == "fdAT")
        PNG_CHUNK_FDAT    fdat;
    else if( length > 0 )
        ubyte   data[length];       // Data (or not present)
    local int64 data_size = FTell() - pos_start;
    uint32  crc <format=hex, fgcolor=cDkPurple>;  // CRC (not including length or crc)
    local uint32 crc_calc = Checksum(CHECKSUM_CRC32, pos_start, data_size);
    if (crc != crc_calc) {
        local string msg;
        SPrintf(msg, "*ERROR: CRC Mismatch @ chunk[%d]; in data: %08x; expected: %08x", CHUNK_CNT, crc, crc_calc);
        error_message( msg );
    }
    CHUNK_CNT++;
} PNG_CHUNK <read=readCHUNK>;
```

This is actually a grammar:

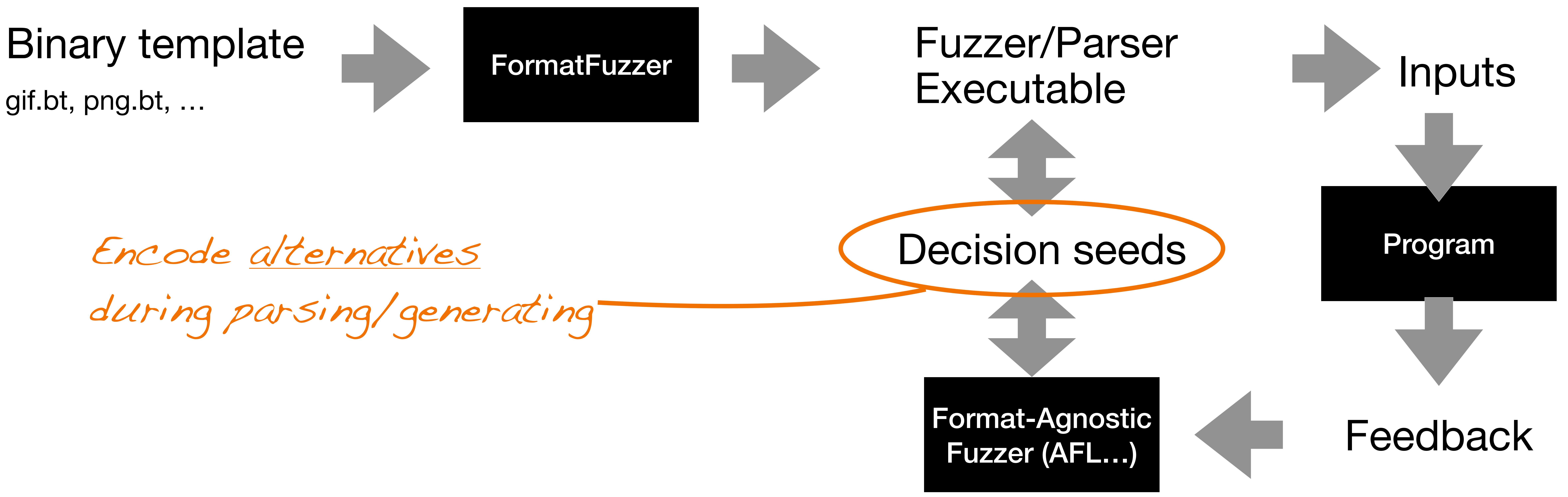
```
<PNG_CHUNK> ::= <PNG_CHUNK_IHDR>
    | <PNG_CHUNK_TEXT>
    | <PNG_CHUNK_PLTE>
    | <PNG_CHUNK_CHRM>
    | ...
```

FormatFuzzer: A Binary Fuzzer Compiler



- **Parses, mutates, generates** inputs in given format
- **Evaluation** on PNG, JPG, GIF, MIDI, MP4, ZIP, PCAP...
- Up to **50x more valid inputs** than with AFL + seeds
- **New coverage on all test subjects** – 15 new "deep" bugs in ffmpeg

Coverage Guidance



FormatFuzzer: A Binary Fuzzer Compiler

BPlist MiniDump 010Theme SHX RegistryDhcpInterfaceOptionsPAL AndroidTrace

CSharp MIDI

AndroidBoot

MTK_MCLF PowerShell ZIPIGI2_TLM EOT ELF GGPK WAVAdv RARDBFPYC Torrent Cryptfs

HFSJournal Luac SinclairMicrodrive Quake3Arena_BSP GZip MOBI InspectorDatesLNK

TGAKryoFlux ICO ROMFS DynamixelProtocol Mifare4k DDS OrCAD_SCH InspectorGUID

FTSMXF ThumbCache CDAIGI2_THM MachO 7ZIP Realflow_Bin_Particles Yara CSVSCP

MP4 LuaJIT Modo KnyttStoriesWorld SRec EatonAPR48 LUKS Batch CPP

RegistryPolicyFile Goclever **FormatFuzzer** Quake3Arena_MD3 SeqBox CLASS ISO

RegistryHive IGI2_WAV Picolog_PLW Anno2070_RDM ShpcAnim BaseMedia

FLV UTMP MongoDBWireProtocol EXE OpenType IGI2_SPR HiewCMarkers Java

WASM BSON VHDCAB ZIPAdv SHP OscarItem MifareUltralight OrochiDAT STL Bash CSS PCX

TIFRDBSQLJSCJPG SytosPlus BMP Mifare1k UnityMetadata OrCAD_LIB JavaScript PCAP

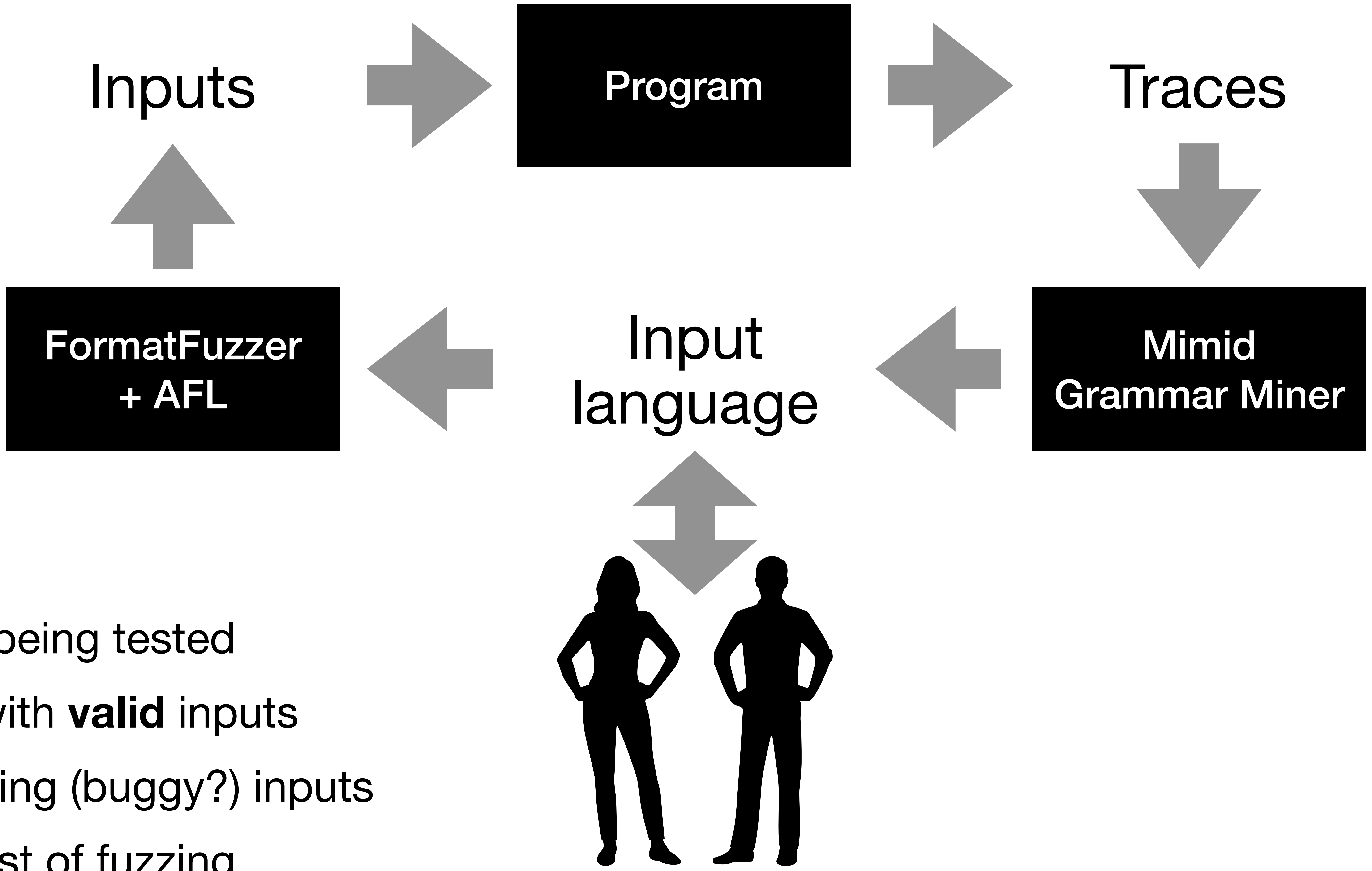
FUTX RIFF CLASSAdv SquashFS EZTap_EZVIEW2 IGI2_TMM LZ4 SQLite TOCAVI OpenWRT-BIN

DEXDMP InspectorWithMP4DateTime GIF NTAG215 IGI2_TEX OGG APFS NetflowVersion5 TNEF

PCAPNG WMFPNGPHPXMLCRXADFElTorito EMFFNT EDID Nus3Audio PDF Python AndroidManifest

AndroidVBMeta UMSE ULP WinhexPos MP3 NDS TTF iNes Drive RESCAP Tacx AndroidResource

Our Vision



- **Control** what is being tested
- Fuzz efficiently with **valid** inputs
- **Learn** from existing (buggy?) inputs
- **Integrate** the best of fuzzing

@AndreasZeller

Grammars

Specify a language (= a set of inputs)

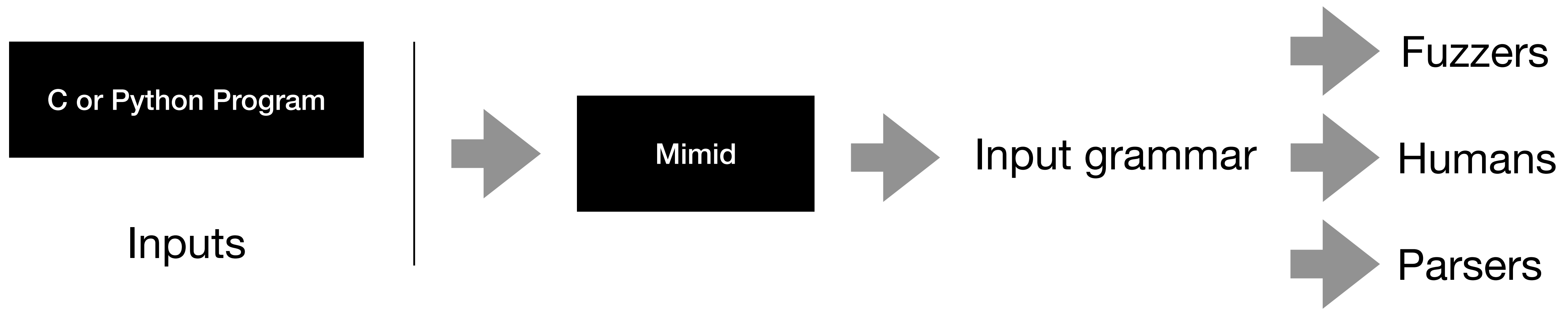
Expansion rule

Nonterminal symbol

```
⟨start⟩   ::= ⟨expr⟩
⟨expr⟩    ::= ⟨term⟩  +  ⟨expr⟩ | ⟨term⟩  -  ⟨expr⟩ | ⟨term⟩
⟨term⟩    ::= ⟨term⟩  *  ⟨factor⟩ | ⟨term⟩  /  ⟨factor⟩ | ⟨factor⟩
⟨factor⟩  ::= + ⟨factor⟩ | - ⟨factor⟩ | ( ⟨expr⟩ ) | ⟨int⟩ | ⟨int⟩ . ⟨int⟩
⟨int⟩     ::= ⟨digit⟩ ⟨int⟩ | ⟨digit⟩
⟨digit⟩   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Terminal symbol

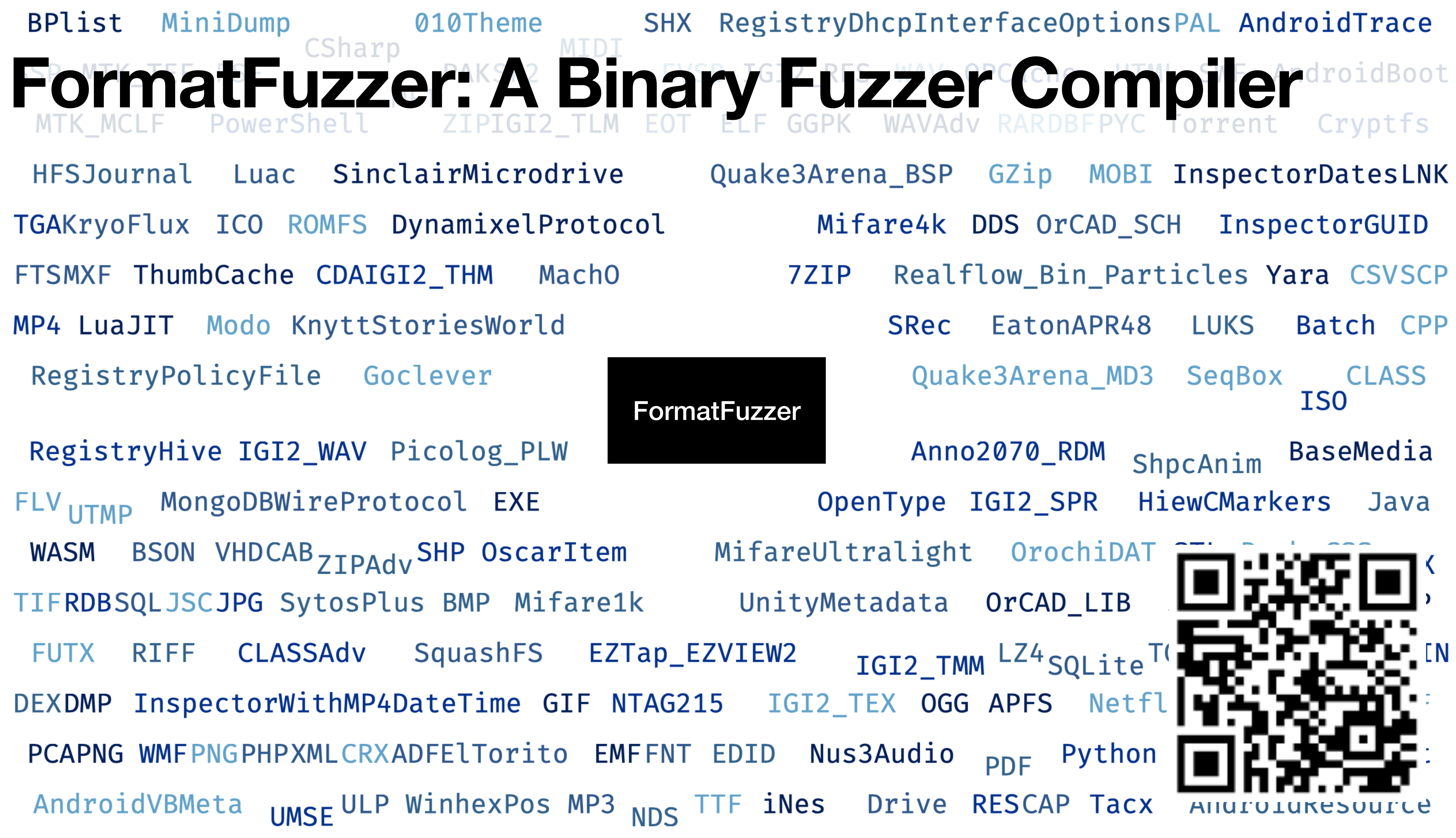
Mimid: A Grammar Miner



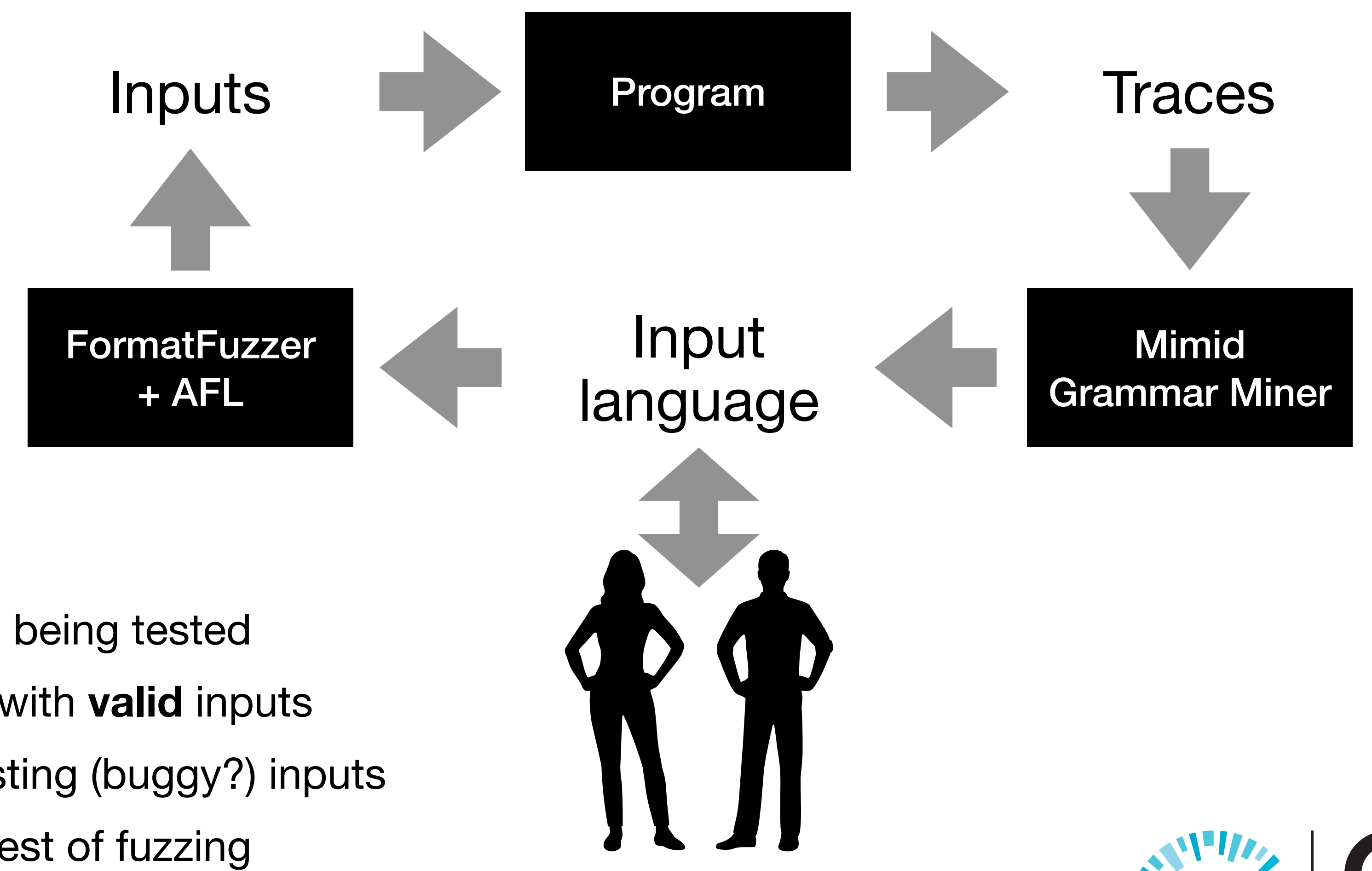
- **Evaluation** on CGI, URL, JSON, TinyC, JavaScript
- Mined grammars **cover 98%** of the actual language
- Mined grammars **well-structured** and **highly readable**

Rahul Gopinath, Björn Mathis, and Andreas Zeller. **Mining Input Grammars from Dynamic Control Flow.** ESEC/FSE 2020.

FormatFuzzer: A Binary Fuzzer Compiler


BPlist MiniDump 010Theme SHX RegistryDhcpInterfaceOptionsPAL AndroidTrace
CSharp MIDI
AndroidBoot
MTK_MCLF PowerShell ZIPIGI2_TLM EOT ELF GGPK WAVAdv RARDBFPYC Torrent Cryptfs
HFSJournal Luac SinclairMicrodrive Quake3Arena_BSP GZip MOBI InspectorDatesLNK
TGAKryoFlux ICO ROMFS DynamixelProtocol Mifare4k DDS OrCAD_SCH InspectorGUID
FTSMXF ThumbCache CDAIGI2_THM MachO 7ZIP Realflow_Bin_Particles Yara CSVSCP
MP4 LuaJIT Modo KnyttStoriesWorld SRec EatonAPR48 LUKS Batch CPP
RegistryPolicyFile Goclever Quake3Arena_MD3 SeqBox CLASS ISO
FormatFuzzer
RegistryHive IGI2_WAV Picolog_PLW Anno2070_RDM ShpcAnim BaseMedia
FLV UTMP MongoDBWireProtocol EXE OpenType IGI2_SPR HiewCMarkers Java
WASM BSON VHDCAB ZIPAdv SHP OscarItem MifareUltralight OrochiDAT
TIFRDBSQLJSCJPG SytosPlus BMP Mifare1k UnityMetadata OrCAD_LIB
FUTX RIFF CLASSAdv SquashFS EZTap_EZVIEW2 IGI2_TMM LZ4 SQLite
DEXDMP InspectorWithMP4DateTime GIF NTAG215 IGI2_TEX OGG APFS
PCAPNG WMFPNGPHPXMLCRXADFElTorito EMFFNT EDID Nus3Audio PDF Python
AndroidVBMeta UMSE ULP WinhexPos MP3 NDS TTF iNes Drive RESCAP Tacx

Our Vision



- **Control** what is being tested
- Fuzz efficiently with **valid** inputs
- **Learn** from existing (buggy?) inputs
- **Integrate** the best of fuzzing

@AndreasZeller

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Useful Links

- **Andreas Zeller** – https://andreas-zeller.info
- **The Fuzzing Book (book + software)** – https://www.fuzzingbook.org
- **FormatFuzzer (software)** – https://uds-se.github.io/FormatFuzzer/
- **Mining Grammars (paper)** – https://publications.cispa.saarland/3101/