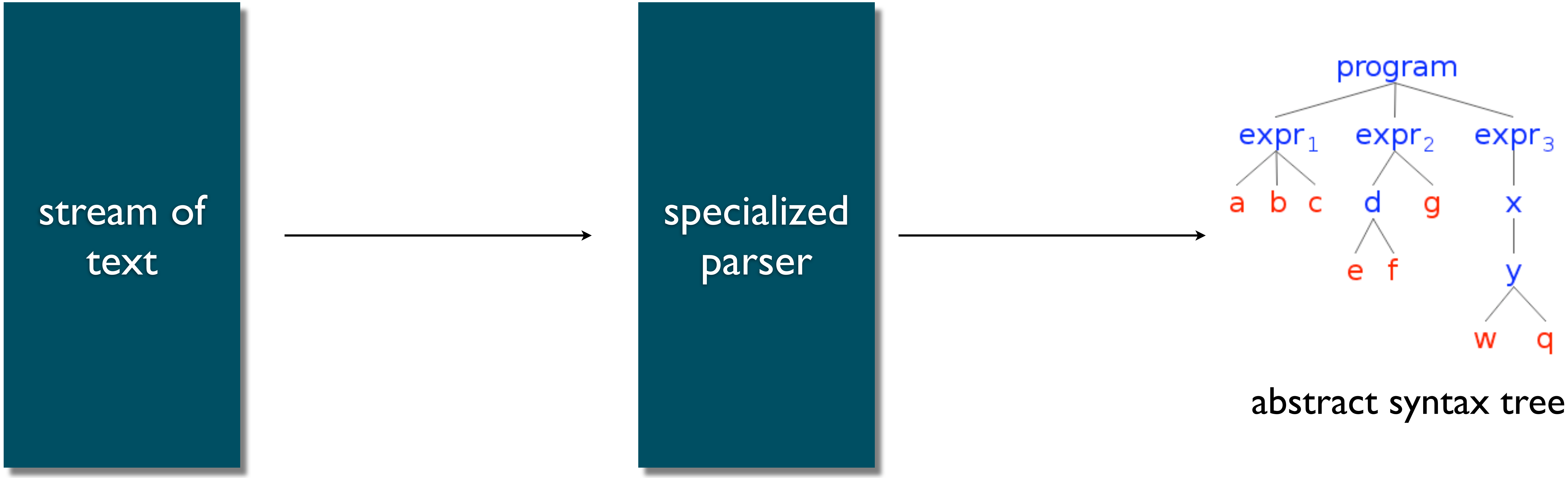
Gel: A Generic Extensible Language

Jose Falcon, William R. Cook

Language Development



stream of text

specialized parser

abstract syntax tree

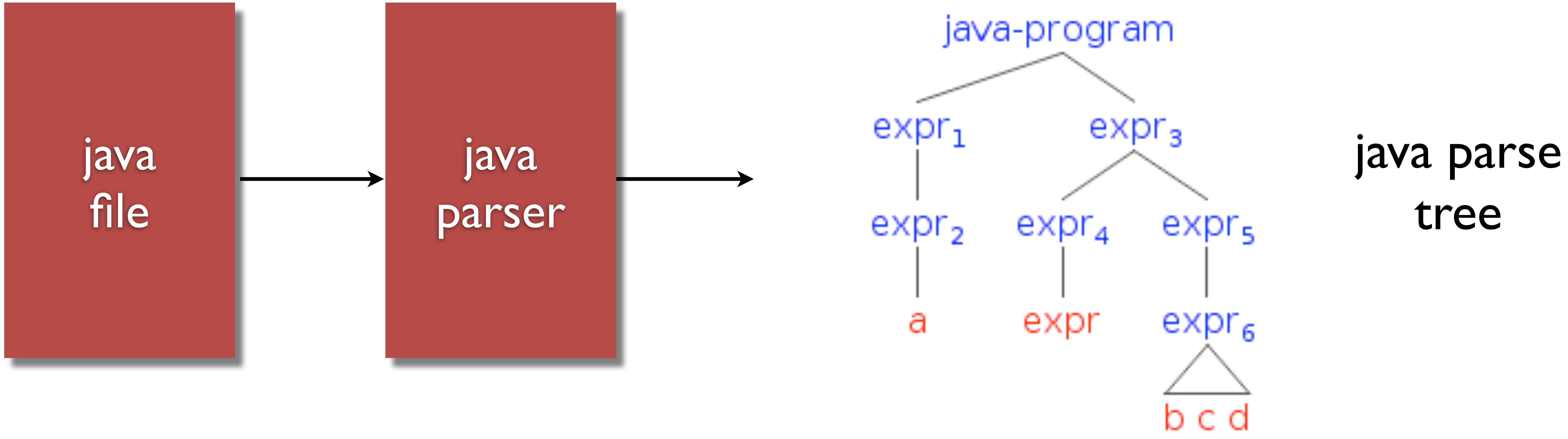
written by the language developer

Java

```
int x = 5;
int product = 1;
for (int i = 1; i<=x; i++) {
   product *= i;
}
```

CSS

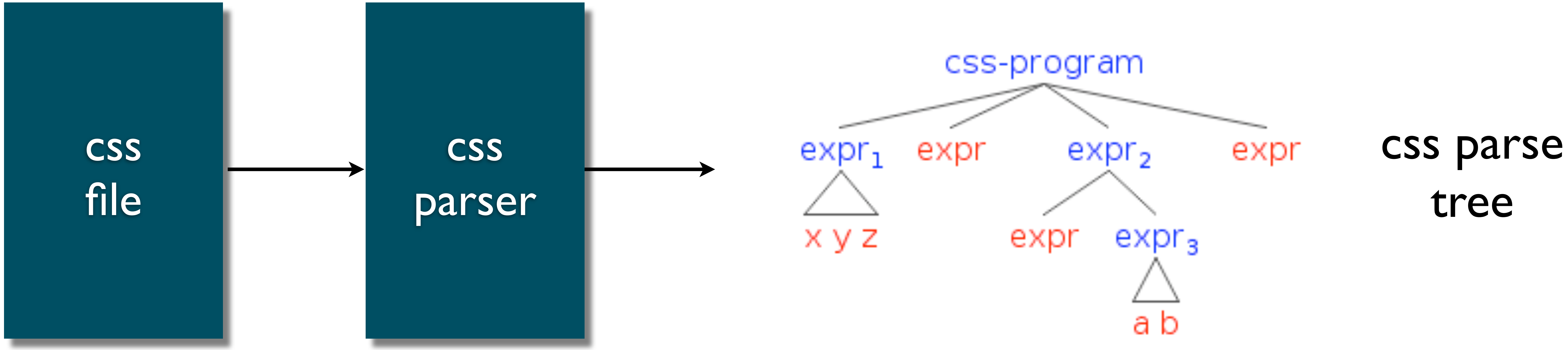
```
body {
 background-color: #ffffff;
 color: #000000;
 font-family: georgia;
}
```



java file

java parser

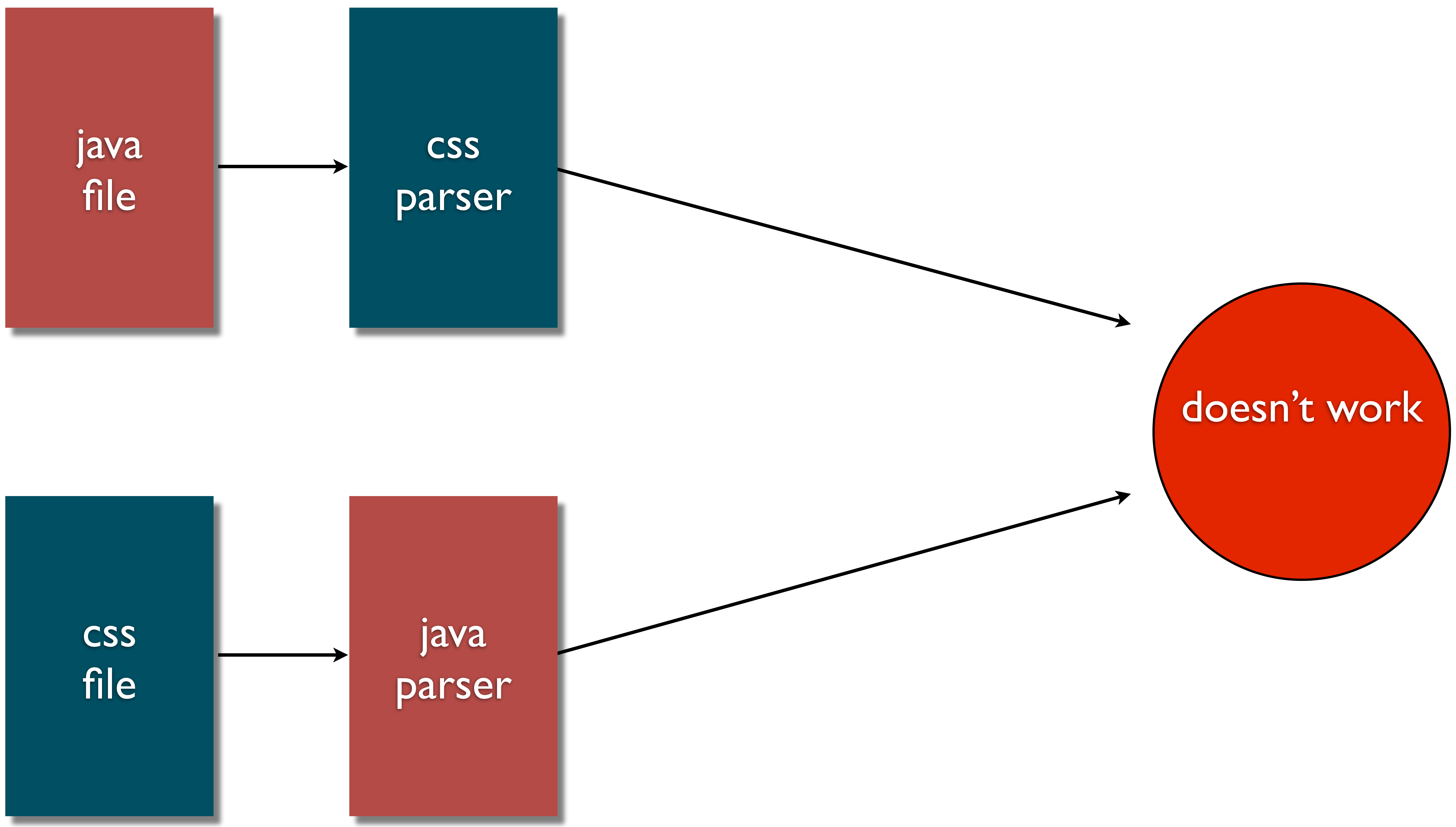
java parse tree



css file

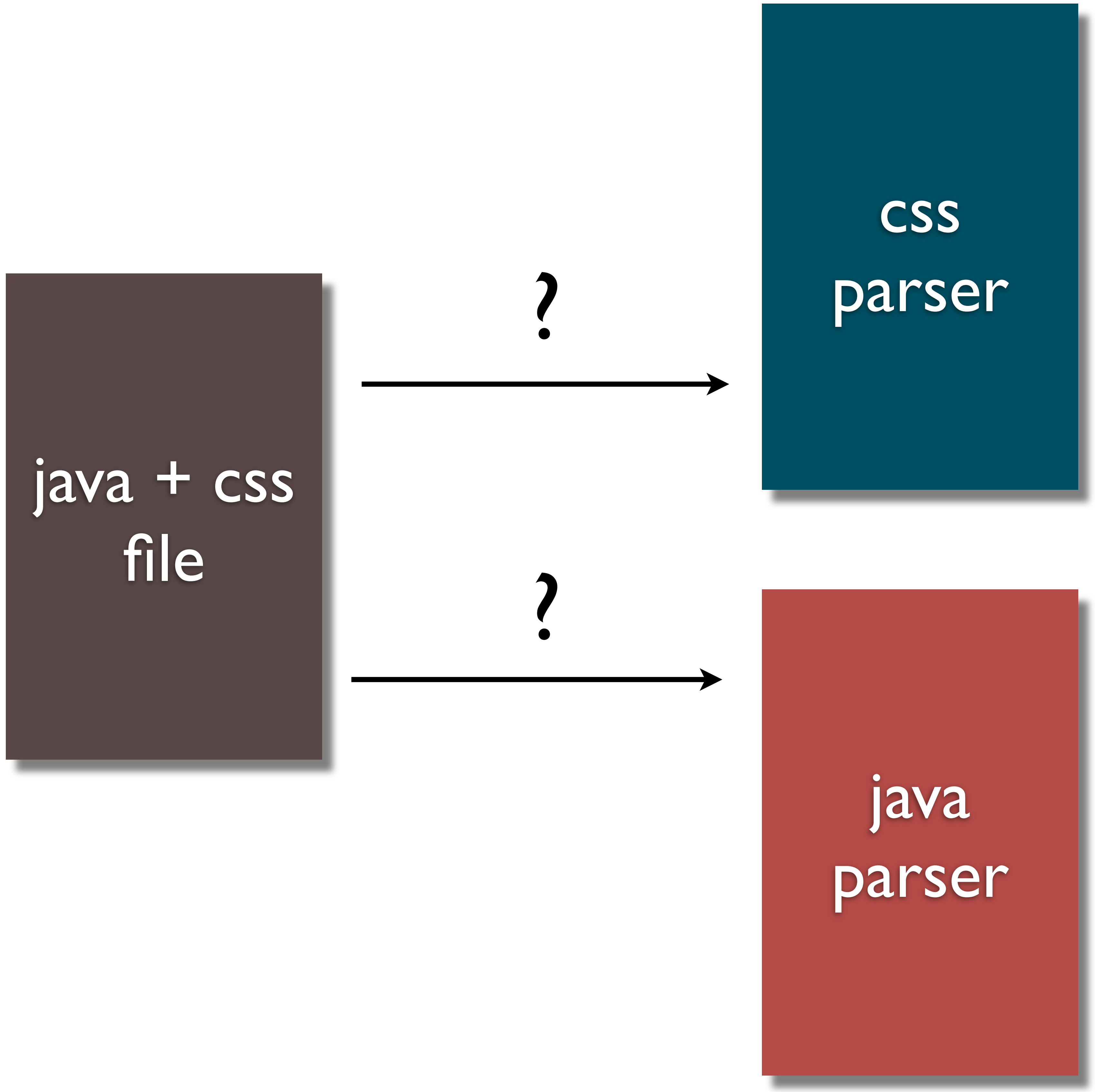
css parser

css parse tree


java
file
css
parser
doesn’t work
css
file
java
parser

Java + CSS

```
int x = 5;
int product = 1;
for (int i = 1; i<=x; i++) {
 product = (body {
  background-color: i;
  color: #000000;
  font-family: georgia;
 });
}
```

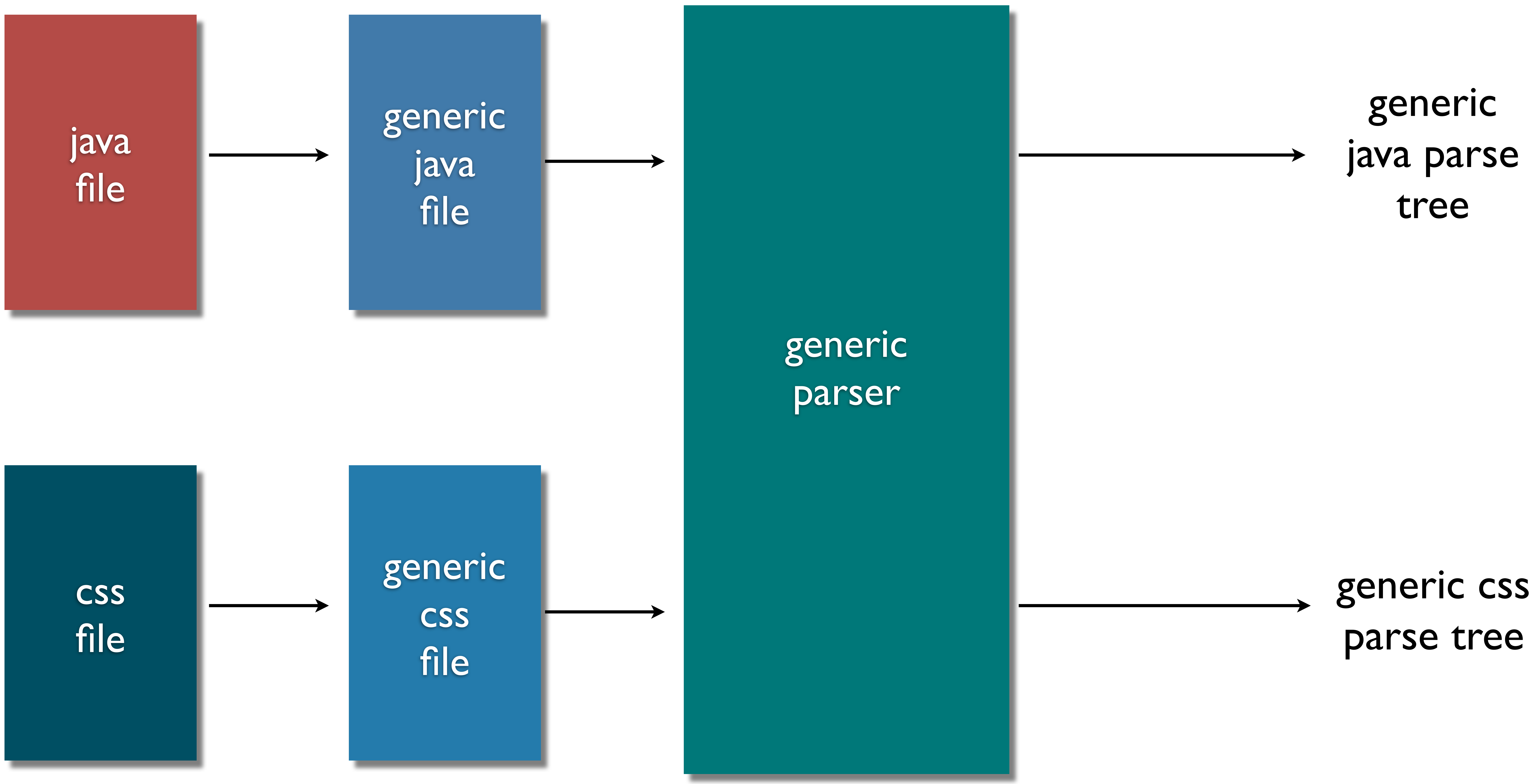


Generic Approach

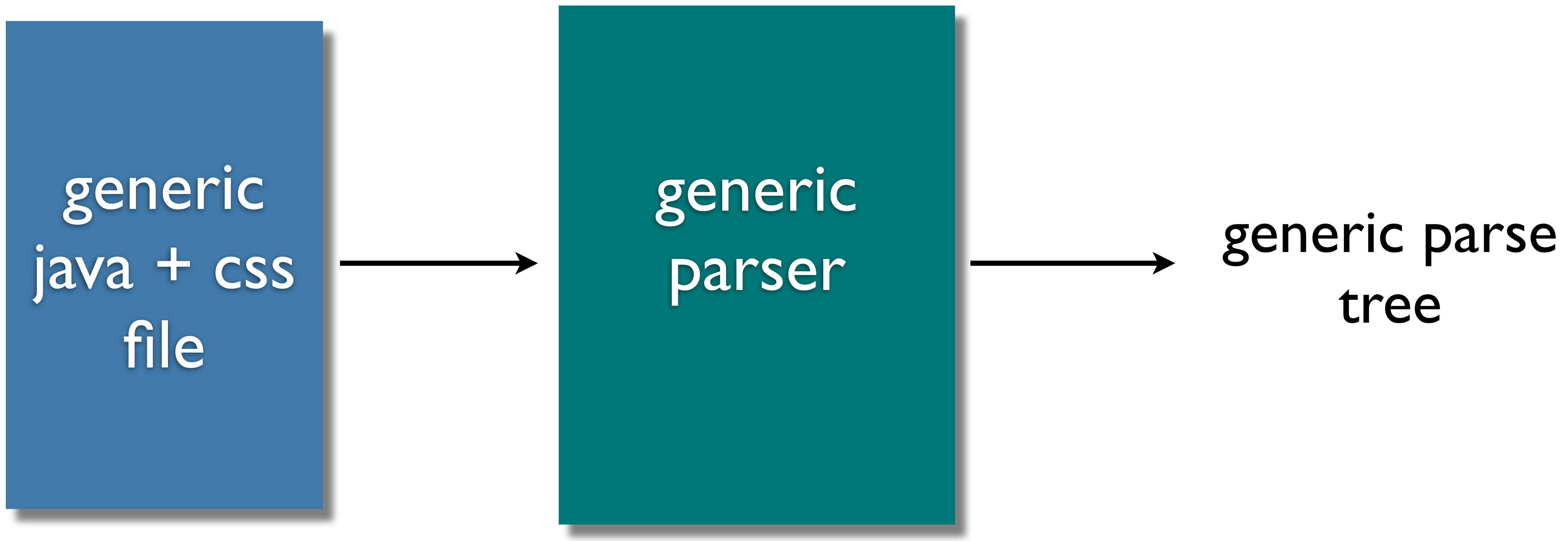
- Use a structured generic language
- Examples
  - XML
  - Lisp S-Expressions

```
<java>
  <declaration>
    <left><symbol>int</symbol>
    <symbol>x</symbol></left>
    <right><integer>5</integer></right>
  </declaration>
  <declaration>
    <left><symbol>int</symbol>
    <symbol>product</symbol></left>
    <right><integer>1</integer></right>
  </declaration>
  <loop>for...
```

```
<css>
  <rule>
    <symbol>body</symbol>
    <property><key>background-color</key>
      <value><hexcolor>ffffff</hexcolor></value>
    </property>
    <property><key>color</key>
      <value><hexcolor>000000</hexcolor></value>
    </property>
    <property><key>font-family</key>
      <value><symbol>georgia</symbol></value>
    </property> ...
```


java file
generic java file
css file
generic css file
generic parser
generic java parse tree
generic css parse tree

```
<java>
  <declaration>
    <left><symbol>int</symbol>
    <symbol>x</symbol></left>
    <right><integer>5</integer></right>
  </declaration>
  <declaration>
    <left><symbol>int</symbol>
    <symbol>product</symbol></left>
    <right><integer>1</integer></right>
  </declaration>
  <loop>for...
  <css>
    <rule>
      <symbol>body</symbol>
      <property><key>background-color</key>
        <value><hexcolor>ffffff</hexcolor></
        value>
      </property>
      <property><key>color</key>
        <value><hexcolor>000000</hexcolor></
        value>
      </property>
      <property><key>font-family</key>
        <value><symbol>georgia</symbol></value>
      </property> ...
```



Generic Approach

- Benefits:
  - Only requires learning a single syntax
  - Easy to embed multiple documents
  - Syntactic validity is independent of any one grammar
- Problems:
  - Not human readable

Gel

- Generic Extensible Language
- Capture syntactic standards over the last 40 years
- Goal: produce human readable generic syntax

Expressions

- Primary, Unary, Binary
- Groups
- Quotes
- Sequences
- “Chunks”
- Keywords

Primary, Unary, Binary, Grouping, Quotes

- Look and act like expressions in common languages
- Quotes are used to indicate a special meaning
- Quotes used the backtick ` operator

```
s1 = ++x * 3 && c == “str”
{ b = (n - 10) ^ arr[i % 5]--; }
fib = (0, 1, 1, 2, 3, 5, 8);
(`first, `+, `last)
```

Operators

- Set of operators is not fixed
- Any combination of operator symbols is an operator
- Separators, `[;,]`, do not combine

```
{1..9} :-> [c =*= “str”]
(:-> (.. 1 9) (=*= c “str”))
```

Precedence & Associativity

- Strict precedence
- For most operators, precedence is defined by the first character
- Try to omit associativity when possible
- Right associative otherwise

```
a + b + c - d
(+ a b (- c d))
```

Sequences

- *Sequences* are expressions not separated by an operator
- Enable Gel to parse compound expressions without information about what the sequence should contain
- Must have higher precedence than binary operators

```
f a 3 + g 10                  (+ (_ f a 3) (_ g 10))
obj size + item max      (+ (_ obj size) (_ item max))
p ::= id | ‘(‘ p ‘)’        (::= p (| id (_ '(' p ')')))
public static void main   (_ public static void main)
```

Spaces

- Parsers tend to ignore whitespace
- Expressions are highly ambiguous without additional syntactic clues
- Whitespace is used to distinguish multiple interpretations

a + * b

a + *b

a + *b	a + (*b)	(+ a *[b])

a+ * b

a + *b	a + (*b)	(+ a *[b])
a+ * b	(a+) * b	(* [a]+ b)

a+ *b

a + *b	a + (*b)	(+ a *[b])
a+ * b	(a+) * b	(* [a]+ b)
a+ *b	(a+) (*b)	(_ [a]+ *[b])

(a+)* b

a + *b	a + (*b)	(+ a *[b])
a+ * b	(a+) * b	(* [a]+ b)
a+ *b	(a+) (*b)	(_ [a]+ *[b])
(a+)* b	((a+)*) b	(_ [[a]+]* b)

a +(*b)

a + *b	a + (*b)	(+ a *[b])
a+ * b	(a+) * b	(* [a]+ b)
a+ *b	(a+) (*b)	(_ [a]+ *[b])
(a+)* b	((a+)*) b	(_ [[a]+]* b)
a +(*b)	a (+(*b))	(_ a +[*[b]])

a + * b

a + *b	a + (*b)	(+ a *[b])
a+ * b	(a+) * b	(* [a]+ b)
a+ *b	(a+) (*b)	(_ [a]+ *[b])
(a+)* b	((a+)*) b	(_ [[a]+]* b)
a +(*b)	a (+(*b))	(_ a +[*[b]])
a + * b	a (+) * b	(* (_ a +) b)

“Chunks”

- Many situations in which whitespace should have low precedence
- Operators without whitespace have higher precedence than those with whitespace
- Whitespace acts as an implicit grouping operator
- Chunks combined with any operator with spaces

```
exp : a=term ‘+’    (: exp (_ (X (= a term)) (‘’ +) ))
a * b+c             (* a (X (+ b c)))
```

“Chunks”

- Sequences without spaces have high precedence than all other operators
- Used for casting, function application, array access

```
f(x, y)[n]          (X (_ f (par (, x y)) ([] n)))
(Integer)x          (X (_ (par Integer) x))
str.charAt(0)       (X (. str (_ charAt (par 0))))
```

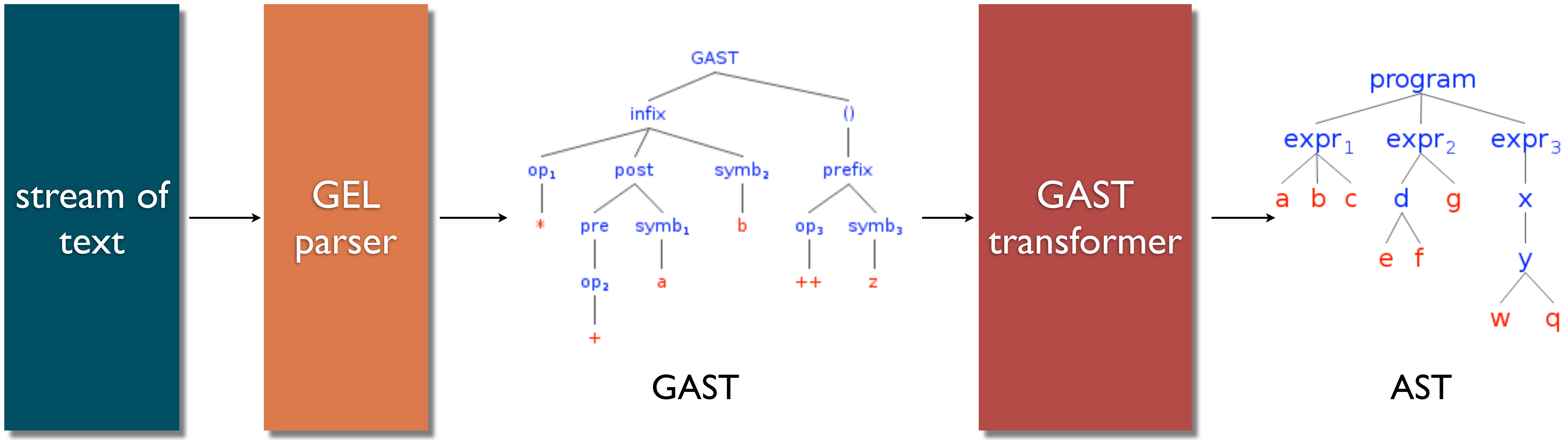
Keywords

- Many languages use keywords to denote a particular syntactic structure
- Gel can generally parse keywords correctly without specific information
- Not a general solution
- Keywords in Gel are used to form expressions into complex statements
- Created with a prefix/postfix colon (:) operator

Keywords

```
while (true) { i++ }            (_ while (par true) ({} [i]++))
return x + y                    (+ (_ return x) y)
return: x + y                   (K [return]: (+ x y))
if a = b then 1 else 2          (= (_ if a) (_ b then 1 else 2))
if: a = b then: 1 else: 2
                          (K [if]: (= a b) [then]: 1 [else]: 2)
```

Language Development



stream of text

GEL parser

GAST

GAST transformer

AST

written by the language developer

Questions?

Thank you!