

Advanced Kenya

Errors & Debugging

Tristan Allwood - tora@doc.ic.ac.uk

Warning: Errors



The following slides contain Kenya programs with errors that some students may find disturbing. Students of a sensitive nature should pay attention now.

Warning: Errors



These errors have all been created under controlled conditions. Please try not to recreate them in coursework submissions or during exams

Whats wrong?

```
void main() {  
    println("Hello")  
}
```

Whats wrong?

```
void main() {  
    println(b());  
}
```

```
boolean b() {  
    return "Hello";  
}
```

Whats wrong?

```
void main() {  
    for(int i = 0; true; i++) {  
        println(".");  
    }  
  
    println("hi!");  
}
```

Whats wrong?

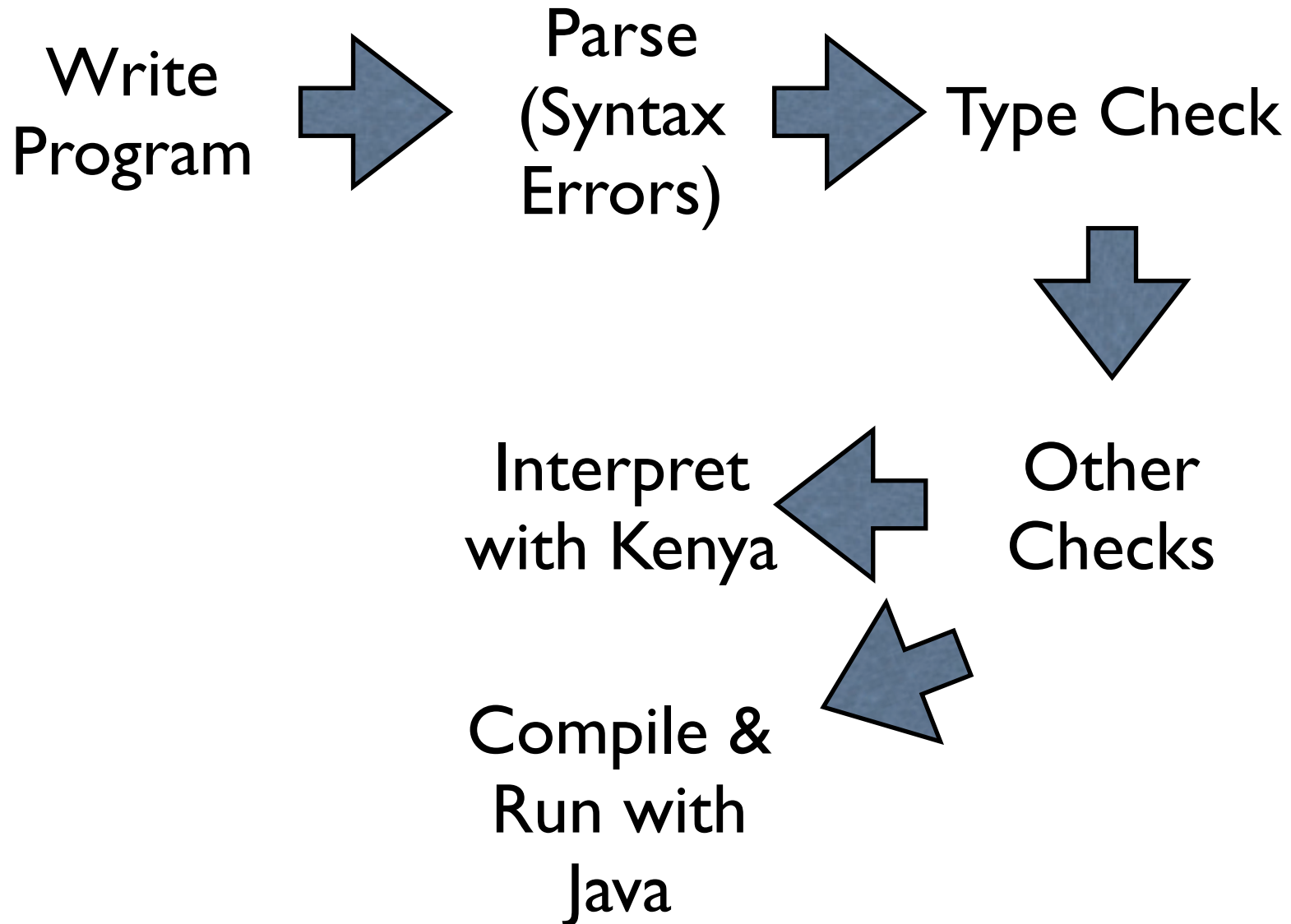
```
void main() {  
    for(int i = 0 ; i < 10; i++ ) {  
        println ( 10 / i );  
    }  
}
```

Whats wrong?

```
void main() {  
    assert power(2,3) == 8;  
    assert power(2,0) == 1;  
}
```

```
int power(int x,int y) {  
    int val = x;  
    for(int i = 1 ; i < y ; i++) {  
        val = val * x;  
    }  
    return val;  
}
```

Kenya Outline



Write Program

Parse - Syntax Errors

- `parse :: String -> KenyaProgramADT`

```
type Identifier = String
```

```
data KenyaProgramADT =  
  ConstDecl KType Identifier |  
  MethodDecl KType Identifier [ArgDec] |  
  ...
```

```
data KType = Int | String | Void | ...
```

Parse - Syntax Errors

- Parsing is done by recognising patterns in the input String

```
type name ( arguments ) {  
    code  
}
```

Parse - Syntax Errors

- Parsing is done by recognising patterns in the input String

```
type name ( arguments ) {  
    code  
}
```

Parse - Syntax Errors

`type ::= void | boolean | int`
`| ...`

`name ::= an alphanumeric`
`word starting with a letter`

`type name (arguments) {`
`code`
`}`

`arguments ::= argument | arguments , argument`
`argument ::= type name`

Parse - Syntax Errors

- Manually writing the code to check for these patterns is annoying
- Parser generators to the rescue!
- Tools you can use to do this:
 - lex & yacc - c / c++
 - antlr - java
 - sablecc - java - what kenya uses

Kenya Syntax

```
declaration =  
  {func_dec}  type_param? type bracket_pair*  
              identifier l_parenthese  
              formal_param_list? r_parenthese block |  
  ... other declaration types
```

```
type = {basic_type} basic_type |  
       {reference_type} reference_type;
```

```
basic_type =  
  {char} char | {int} int | {double} double |  
  {string} String | {boolean} boolean | {void} void;
```

```
reference_type = {class_type} identifier type_param?
```

Syntax - Errors

- Parsing is done to produce a representation of the program
- If parsing fails (code does not match the patterns) we can only say where roughly we stopped matching the patterns and what pattern(s) we expected
- Error recovery for parsing is hard

Should these parse?

```
void main()) {  
}
```

```
void main() {  
    b();  
}
```

```
boolean b() {  
    return "Hello";  
}
```

Type Check

```
void main() {  
    int x = 3;  
    double y = x;  
    takesADouble(x);  
}
```

```
void takesADouble(double d) {  
    println(d);  
}
```

Type Check (oops!)

```
void main() {  
}
```

```
void blah( void blah ) {  
}
```

Other Checks

```
void main() {  
    b(false);  
}
```

```
boolean b(boolean b) {  
    if(b) {  
        return true;  
    } else {  
        println("hello");  
    }  
}
```

Other Checks

```
void main() {  
    for(int i = 0 ; true ; i++ ) {  
    }  
    println("Hello");  
}
```

Other Checks

```
void main() {  
    if(true) {  
        println("tt");  
    } else {  
        println("ff");  
    }  
}
```

Other Checks

```
void main() {  
    println(b());  
}
```

```
boolean b() {  
    for(int i = 0 ; true ; i++ ) {  
    }  
}
```

Compile & Run with Java

Interpret & Debug With Kenya

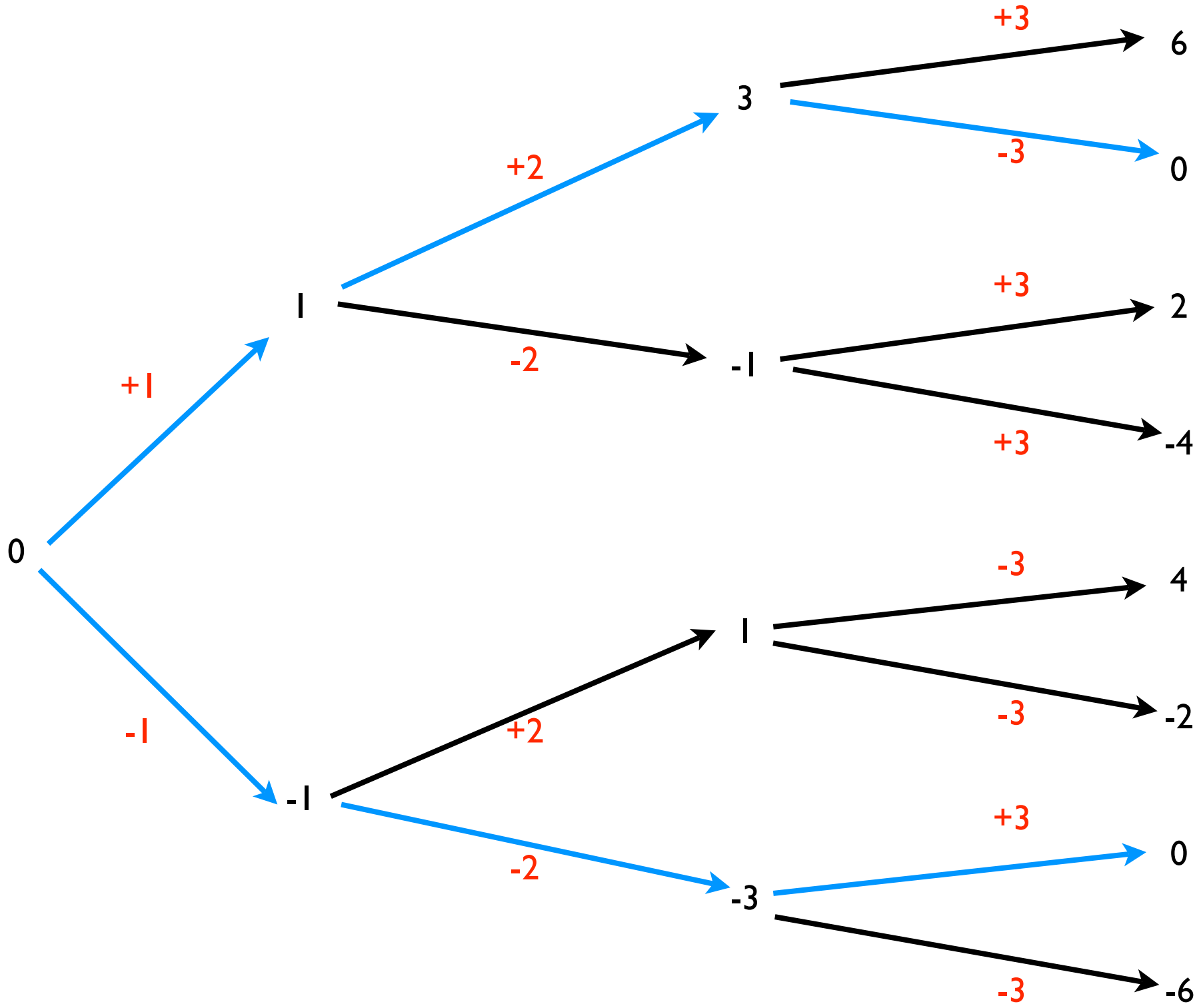
**This slide is intentionally
left blank**

All Is Nothing

- Consider the sequence of digits from 0 through N ($N \geq 1$) in increasing order.
- Insert either a '+' (addition) or a '-' (subtraction) between each pair
- Print all combinations that sum to 0

All Is Nothing

- $N = 3$
 - $0 + 1 + 2 - 3 = 0$
 - $0 - 1 - 2 + 3 = 0$
- $N = 7$
 - $0 + 1 + 2 - 3 + 4 - 5 - 6 + 7 = 0$
 - $0 + 1 + 2 - 3 - 4 + 5 + 6 - 7 = 0$
 - $0 + 1 - 2 + 3 + 4 - 5 + 6 - 7 = 0$
 - $0 + 1 - 2 - 3 - 4 - 5 + 6 + 7 = 0$
 - $0 - 1 + 2 + 3 + 4 + 5 - 6 - 7 = 0$
 - $0 - 1 + 2 - 3 - 4 + 5 - 6 + 7 = 0$
 - $0 - 1 - 2 + 3 + 4 - 5 - 6 + 7 = 0$
 - $0 - 1 - 2 + 3 - 4 + 5 + 6 - 7 = 0$



All Is Nothing

```
void main() {
    assert doSum(1) == 0;
    assert doSum(3) == 2;
    assert doSum(7) == 8;
}
int doSum(int n) {
    assert n >= 1 : "Can only do sums for n >= 1";
    return doSum(1, n, "0", 0);
}
int doSum(int cval, int n, String path, int total) {
    if(cval > n) {
        if(total == 0) {
            println(path + " = 0");
            return 1;
        } else {
            return 0;
        }
    }
    int count = doSum(cval+1, n, path + "+" + cval, total + cval);
    count = count + doSum(cval+1, n, path + "-" + cval, total - cval);
    return count;
}
```

Next Time: Ministry of Funny Walks

Arrays, and interesting ways of walking them