ScalaCheck

A Tool for Property-Based Test

Sheng-Ying Hsiao

What is ScalaCheck?

- ScalaCheck is a library written in Scala and used for automated property-based testing of Scala or Java programs.
- ScalaCheck was originally inspired by the Haskell library QuickCheck.
- The basic idea is that you define a property that specifies the behaviour of a method or some unit of code, and ScalaCheck checks that the property holds.
- All test data are generated automatically in a random fashion, so you don't have to worry about any missed cases.

https://www.scalacheck.org/
https://github.com/typelevel/scalacheck/blob/master/doc/UserGuide.md

API for Property

- Import the *forAll* method, which creates universally quantified properties.
- *forAll* takes a function as parameter, and creates a property out of it that can be tested with the check method.
  - The function should return Boolean or another property
- We will dig into the different property methods later on, but *forAll* is probably the one you will use the most.

```
import org.scalacheck.Prop.forAll

def trivialProperty() =
      forAll {
            (i: Int) => true
      }
```

https://github.com/typelevel/scalacheck/blob/master/doc/UserGuide.md

Property for List :::

- List(1,2,3) ::: List(4,5) === List(1, 2, 3, 4, 5)
- Practice defining a property for :::
- Connect inputs (of ::: and lambda function)
- Make sure your property always true

```
import org.scalacheck.Prop.forAll

def propConcatLists() =
    forAll {
        (l1: List[Int], l2: List[Int]) =>
            l1.size + l2.size == (l1 ::: l2).size
    }
```

Incorrect Property for sqrt

- sqrt(100) == 10
- The following property fails

```
import org.scalacheck.Prop.forAll

def propSqrt() =
    forAll {
        (n: Int) =>
            scala.math.sqrt(n*n) == n
    }
```

Testing in SBT

- Example of String Concate
  - "Hello" + "World" === “HelloWorld”
- The Properties class contains a main method that can be used for simple execution of the property tests.

https://github.com/typelevel/scalacheck/blob/master/doc/UserGuide.md

Test a Simple Implementation

- 比較 String Concat. 之前與之後的長度關係
- Compare two different testing strategies
    - Example-Based Test
    - Property-Based Test

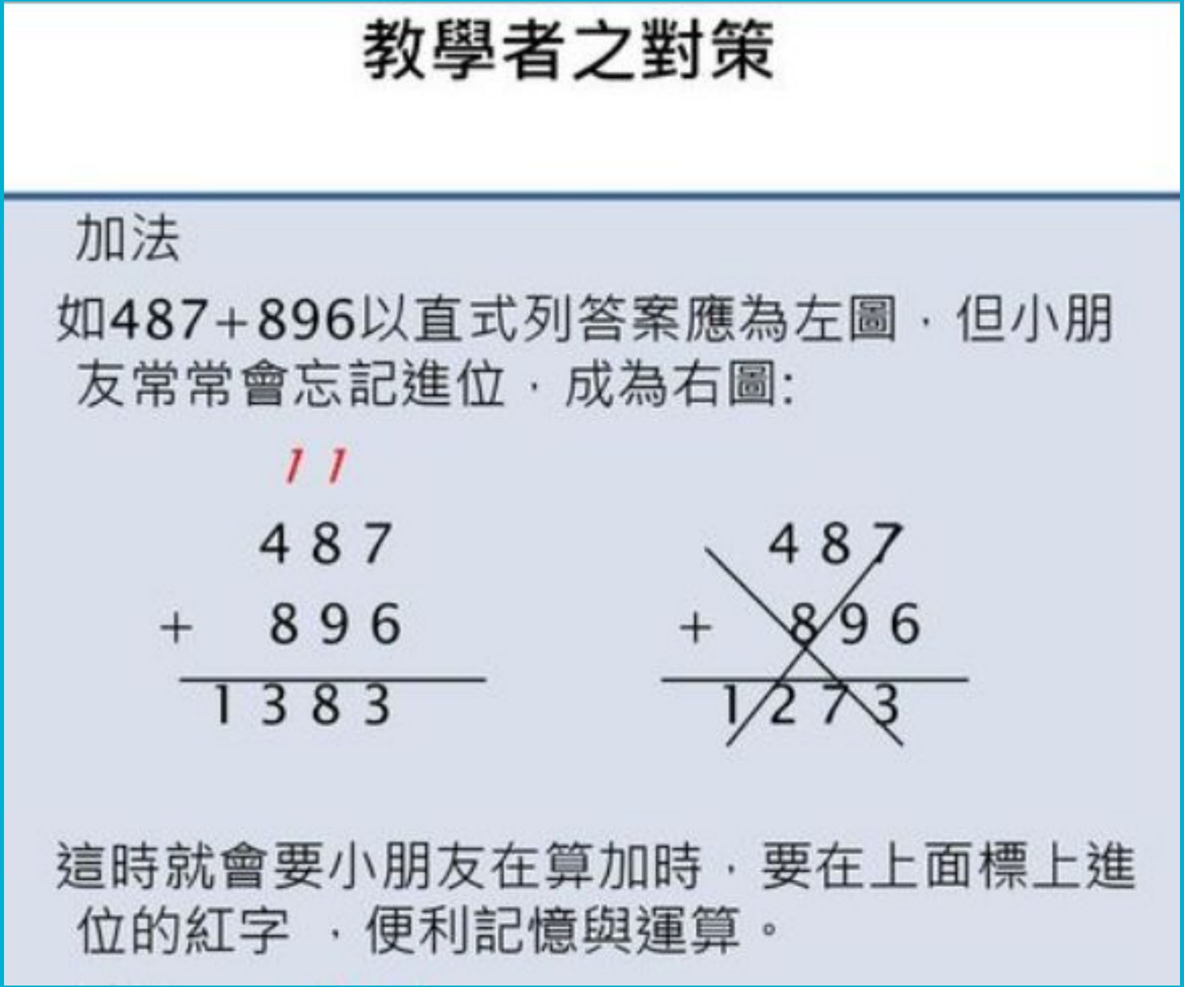
```
def checkConcatenateLength(a: String, b: String): Boolean = {
      // Relation between a.length, b.length, and (a+b).length
}
```

Example: VERY HARD COMPUTATION

- Implementation target: ADDITION
- Example-Based Test
- Property-Based Test
  - motivation
  - define properties step by step

https://elixirschool.com/en/lessons/libraries/stream-data/

你認為的「加法」長什麼樣子？



教學者之對策

加法

如487+896以直式列答案應為左圖·但小朋友常常會忘記進位·成為右圖:

$$\begin{array}{r} 1\,1 \\ 4\,8\,7 \\ +\quad 8\,9\,6 \\ \hline 1\,3\,8\,3 \end{array} \qquad \xcancel{\begin{array}{r} 4\,8\,7 \\ +\quad 8\,9\,6 \\ \hline 1\,2\,7\,3 \end{array}}$$

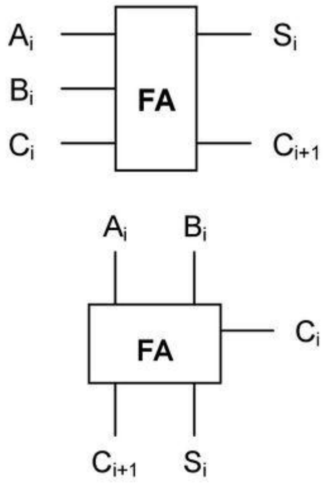
這時就會要小朋友在算加時·要在上面標上進位的紅字 ·便利記憶與運算。

https://slidesplayer.com/slide/11603365/

加 法 器

□全加器：(Full-Adder； FA)

功能模組



真值表

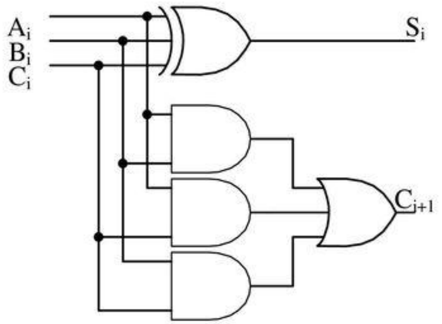
A_i	B_i	C_i	C_{i+1}	Si
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

函數表示式

$$S_i = A_i \oplus B_i \oplus C_i$$

$$C_{i+1} = A_i \cdot B_i + A_i \cdot C_i + B_i \cdot C_i$$

邏輯電路



https://slidesplayer.com/slide/11111276/

Example: VERY HARD COMPUTATION

- Example-Based Tests have some drawbacks:
  - It can be easy to miss edge cases, since you’re only testing a few inputs.
  - You can write these tests without thinking through your requirements thoroughly.
  - These tests can be very verbose when you use several examples for one function.

https://elixirschool.com/en/lessons/libraries/stream-data/

Example: VERY HARD COMPUTATION

- How to use Property-Based Test
    - 不要寫 test case
        - 不生成 test case 的話要怎麼 test?
    - 使用 Scalacheck
        - 無法預先知道隨機生成的 test case 內容，也就無法知道運算結果
        - 你不能把運算實作再拿去 testing 裡使用

Properties for VERY HARD COMPUTATION

- The parameter order doesn't matter
  - Commutativity property
- Doing "add 1" twice is the same as doing "add 2" once
  - Associativity property
    - (a + b) + c = a + (b + c) implies (a + 1) + 1 = a + (1 + 1)
- Adding zero does nothing These properties apply to ALL inputs So we have a very high confidence that the implementation is correct
  - Identity property

https://www.slideshare.net/ScottWlaschin/an-introduction-to-property-based-testing

Properties for VERY HARD COMPUTATION

- These properties define addition
- 少了任何一個 property 就不是一個合格的 testing

https://www.slideshare.net/ScottWlaschin/an-introduction-to-property-based-testing

Example: VERY HARD COMPUTATION

- 你認為的「加法」長什麼樣子？
- Another Properties？

Property-Based Testing

- You do not supply specific example inputs with expected outputs as with unit tests.
- Instead, you
  - define properties about the code and
  - use a generative-testing engine

  to create <u>randomized inputs</u> to ensure the defined <u>properties</u> are correct.

- 精神: 了解待測函數的核心性質，並且把性質寫成 code 去測試該函數

https://dev.to/jdsteinhauser/intro-to-property-based-testing-2cj8

Library for Property-Based Testing

- Scalacheck 是一個讓你可以輕易完成 Property-Based Testing 的工具
  - Scalacheck can define properties about the code and
  - it has a generative-testing engine.

https://dev.to/jdsteinhauser/intro-to-property-based-testing-2cj8

References

- https://www.scalacheck.org/
- https://github.com/rickynils/scalacheck/blob/master/doc/UserGuide.md
- https://www.slideshare.net/ScottWlaschin/an-introduction-to-property-based-testing
- https://dev.to/jdsteinhauser/intro-to-property-based-testing-2cj8
- https://elixirschool.com/en/lessons/libraries/stream-data/

The End