Using QuickCheck for testing a domain specific language

Adam Lindberg
Erlang Training and Consulting

What is QuickCheck?

- QuickCheck test software based on properties
- Write one property and let QuickCheck do the tedious work of creating a lot of test cases

```
?FORALL(S, string(),
        S == lists:reverse(
                 lists:reverse(S))).
```

What is QuickCheck?

- Test cases are generated randomly from properties
- Test data is provided by generators

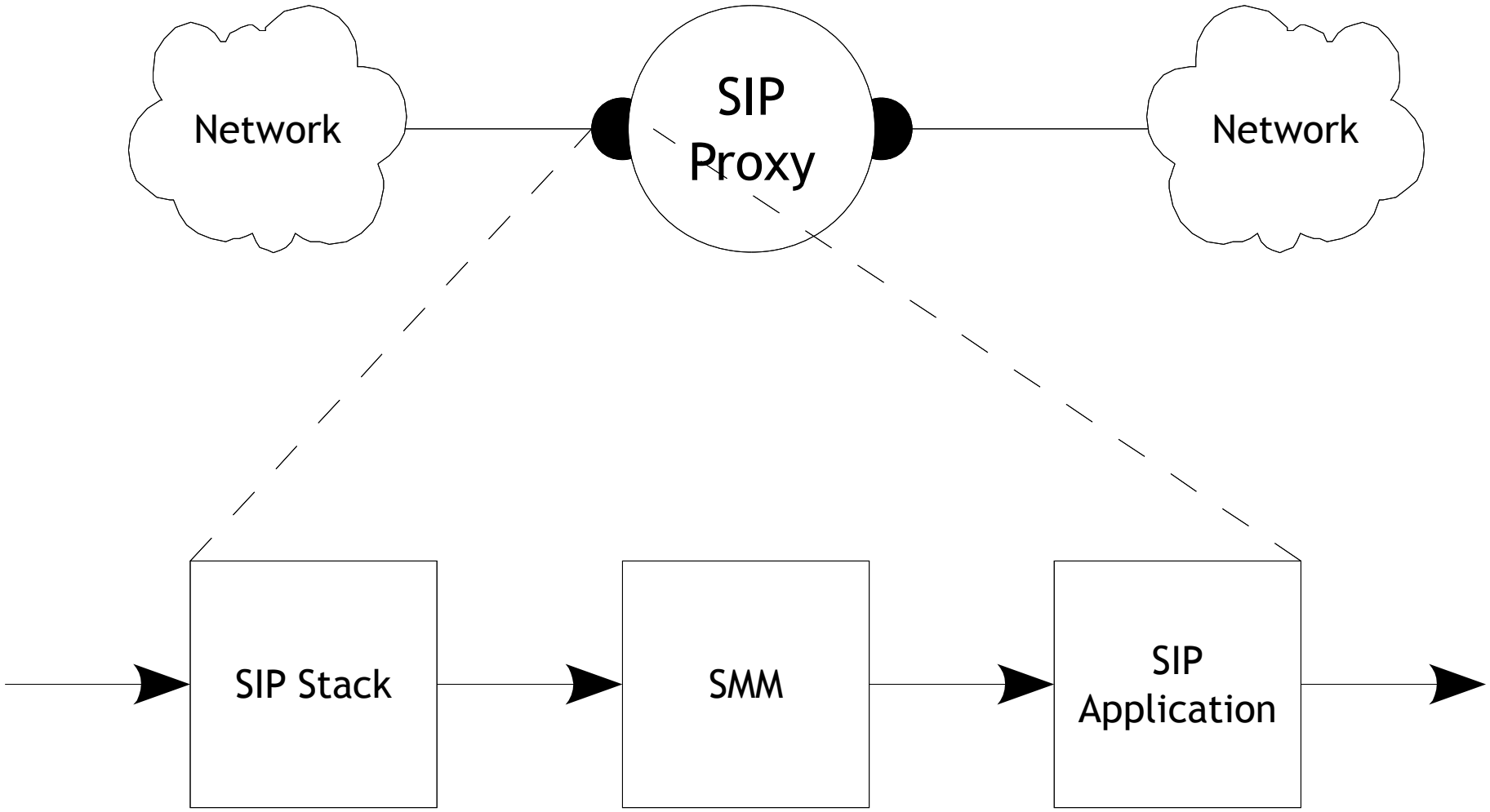
```
string() ->
    list(char()).

sip_response_code() →
    choose(100, 699).
```

What is a Domain Specific Language?

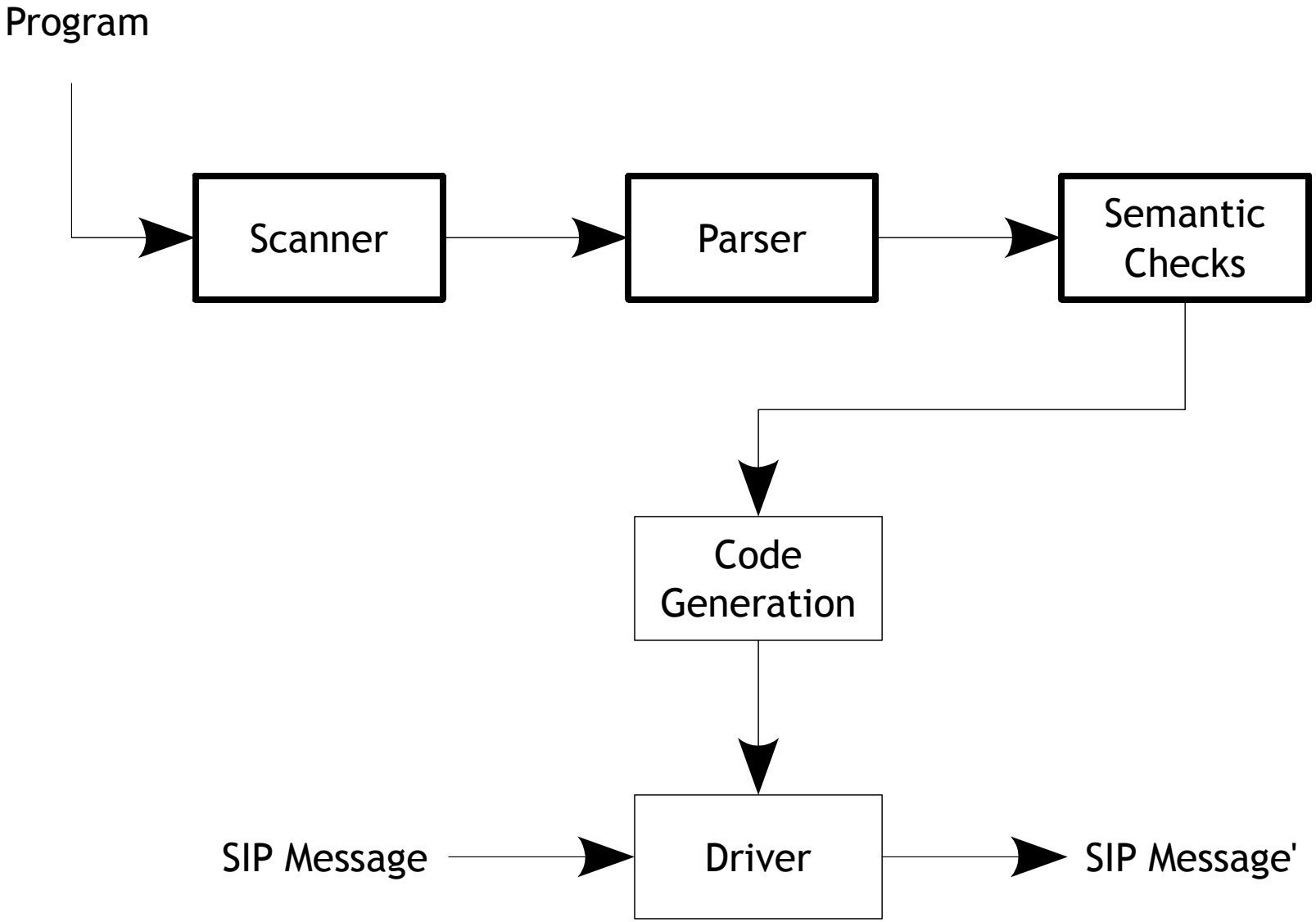
”In software development, a domain-specific language (DSL) is a programming language or specification language dedicated to a particular problem domain, a particular problem representation technique, and/or a particular solution technique”

- Wikipedia

What is SIP Message Manipulation?

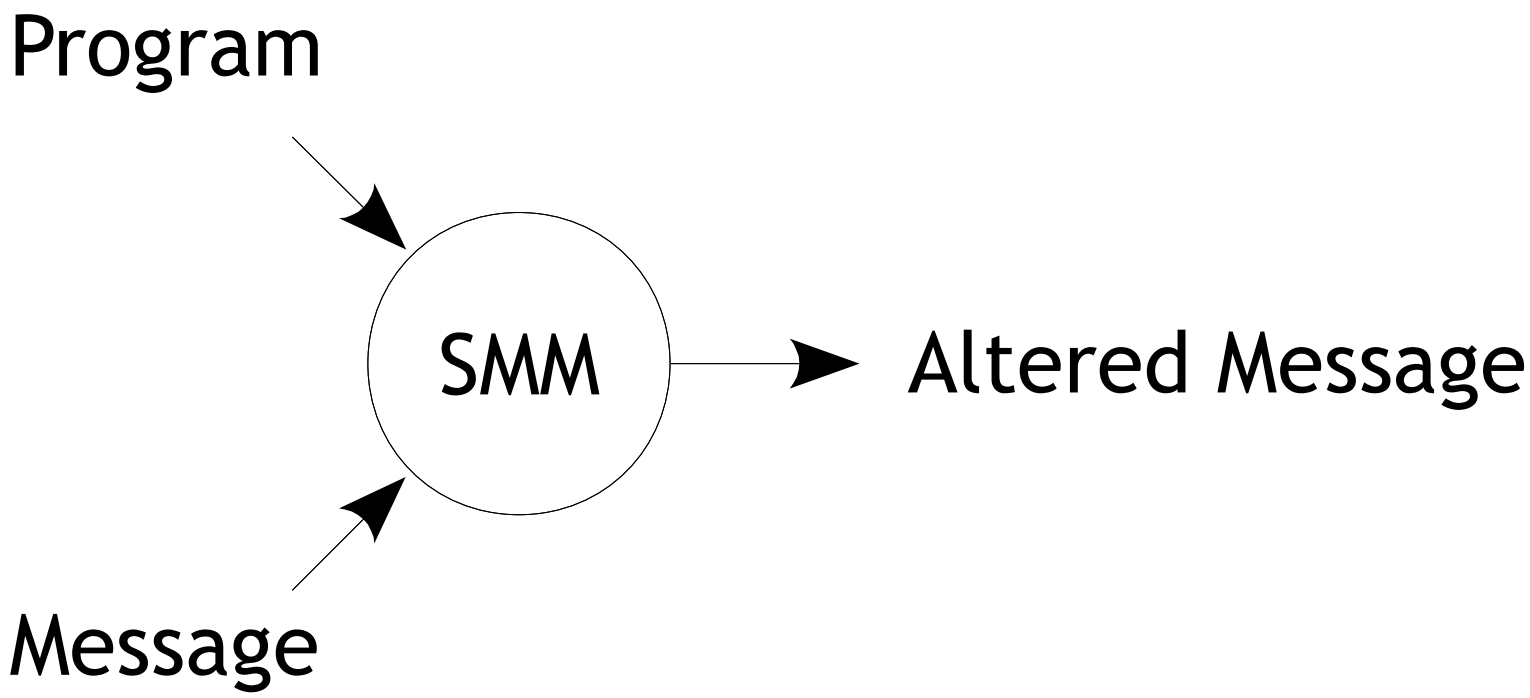


What is SIP Message Manipulation?



Scope

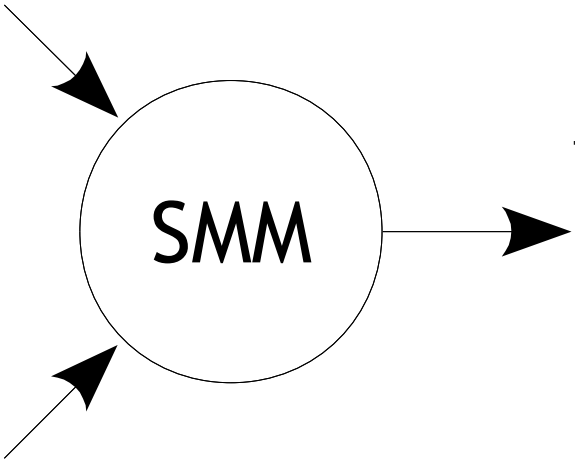
- Our system has two inputs and one output:



Scope

- Our system has two inputs and one output:

```
Ruleset “my ruleset”
    If
        SIP:To.sip_uri.host == “erlang.org”
    Do
        SIP:Subject := “Erlang rocks!”
    End
End
```




```
INVITE sip:joe@erlang.org SIP/2.0
To: Joe <sip:joe@erlang.org>
From: Robert <sip:robert@erlang.org>
Call-ID: a84b4c76e66710@erlang.org
Subject: Hello Joe!
Contact: <sip:robert@erlang.org>
```

```
INVITE sip:joe@erlang.org SIP/2.0
To: Joe <sip:joe@erlang.org>
From: Robert <sip:robert@erlang.org>
Call-ID: a84b4c76e66710@erlang.org
Subject: Erlang rocks!
Contact: <sip:robert@erlang.org>
```

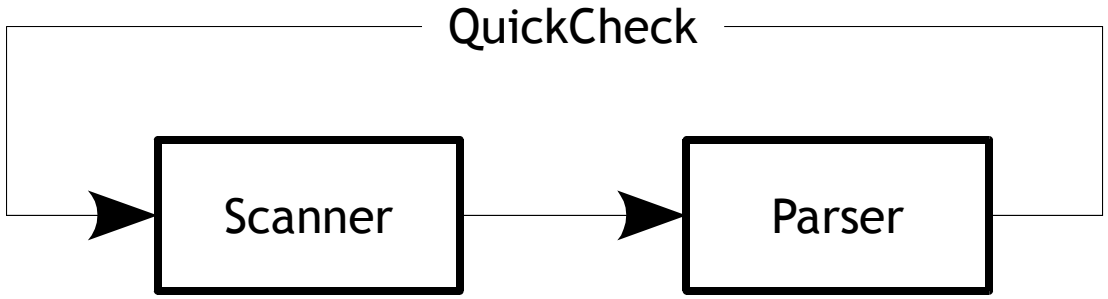
Scope

- We have narrowed down the scope to two types of testing
  - Symmetry
  - Functionality assurance (“crash testing”)
- Theoretically it is possible to test all functionality
  - Probably very hard!
- We are testing code that does not exist until delivered to the customer

How the Tests Are Set Up

- Symmetry tests send test data on a round trip in the system

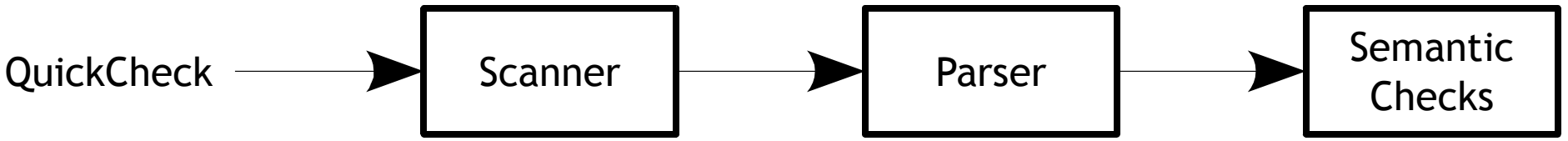
```
?FORALL(Program, program(),
        begin
            String = to_string(Program),
            Program == parse(
                         scan(String))
        end)
```



How the Tests Are Set Up

- Functionality assurance tests excersizes the code with random test data and makes sure it doesn't crash

```
?FORALL(Program, program(),
        begin
            String = to_string(Program),
            check(parse(scan(String))),
            true
        end)
```



Our Initial Approach

- Home made generators
- Introduced code duplication between the implementation and test code
  - Takes time, hard to maintain

Using the Grammar Generator

- Generates test data from the domain of all possible permutations of programs valid in the language
- Makes sure other parts of the system accepts valid input according to the grammar
- Tests the grammar too, each failed test case can indicate a fault in the grammar

Faults Found

- Inconsistencies in the output produced by the grammar
  - The internal format had different representations of the same data
- Crashes in the semantic checks
  - Valid programs produced crashes

Lessons Learned

- Don't try to keep up with your code, write generators that do it for you
- Implement symmetric input / output functions, even if you don't need it
  - Great for testing symmetry
  - Data round trips can be tested at various points in your program, the deeper the better

Thank you for listening!

Questions please!