Executable Formal Semantic of P4 and Applications

Ali Kheradmand, Grigore Rosu
University of Illinois at Urbana Champaign
A need: Automated Verification

Complexity (of networks and hardware) + Flexibility and Agility (of SDNs and P4) → Subtle Bugs

(increased chance of)


Current approach

Language designers → Language specification (In English) → Tool developer

Tool developer → Compiler/Interpreter

Verification tool
Formal semantics matter

Example from C language:

```c
* int main(void) {
    int x = 0;
    return (x = 1) + (x = 2);
}
```

P4 Language Specification Version 1.0.3 (November 2, 2016):

“P4 assumes parallel semantics for the application of all the primitive actions executing as a result of a match in a given table. The execution of actions across different tables assumes sequential semantics where the sequence is determined by the control flow, described in Section 12.”

modify_field(hdr.fldA, 1);
modify_field(hdr.fldB, hdr.fldA);
modify_field(hdr.fldA, 1);
modify_field(hdr.fldA, 2);

* Rosu et al., http://www.kframework.org/index.php/K_Overview

GCC: 4
Clang: 3
Frama-C [Filliâtre et al]: 4
HAVOC [Lahiri et al]: 4
ISO C11: undefined
Our vision

Formal Language Definition (Syntax and Semantics)

- Interpreter/Debugger
- Test Generator
- Deductive Program Verifier
- Compiler
- Model Checker
- Equivalence Checker
- Symbolic Execution Engine
- ...
K Framework [Rosu et al, 2010]

- Rewrite-based programming language semantics engineering framework
  - Successfully used to give complete semantics to C, Java, JavaScript, ...

- Semantics:
  - Configuration (state): nested cells
  - Rewrite rules (transitions): $C[L_1 \Rightarrow R_1, \ldots, L_n \Rightarrow R_n]$
P4K: Semantics of P4\textsubscript{14} (V1.0.3) in K

• Not all features are currently supported
  • Enough rules to run simple P4 programs
  • e.g: \texttt{basic\_routing} from p4factory

• Challenge: ambiguities and undefined behavior
  • Parallel semantics
  • Deparsing
  • Operands with different widths
  • ...

• More: https://github.com/kframework/p4-semantics/blob/master/issues.txt
• Most addressed in P4\textsubscript{16}
Tools (all for free!)

- Interpreter/Debugger
- Test Generator
- Deductive Program Verifier
- Compiler
- Model Checker
- Equivalence Checker
- Symbolic Execution Engine
- ...
Potential App 1: Finding bugs using Symbolic Execution

• Property: Does the program either drop the packet or set the value of `egress_spec`? *

• Start with a **symbolic packet** $\langle ? P \rangle_{\text{packet}}$

• Search for a pattern in which neither the packet is dropped nor the `egress_spec` is set

* Nate Foster, personal communication
Potential App 1: Finding bugs using Symbolic Execution (cont.)

• Tested on *basic_routing*

• Found 2 type of inputs that lead to violation:
  • *P.ether*
  • *P.ipv4.dstAddr not in ipv4_fib and ipv4_fib_lpm*

```cpp
parser parse_ethernet {
    extract(ethernet);
    return select(latest.etherType) {
        0x0800 : parse_ipv4;
        default: ingress;
    }
}
```
Potential App 2: Data plane verification

• Check network-wide reachability properties in data plane snapshot (for all packet headers)
  • E.g: Does all packets from A reach B?
  • HSA\textsuperscript{[Kazemian et al, NSDI’12]}, Veriflow\textsuperscript{[Khurshid et al, NSDI’13]}, Delta-net\textsuperscript{[NSDI’17]}, ...

• Can be checked by inserting \texttt{symbolic packets} and using symbolic execution

• Need semantics of network
  • Easy to add
Tools (all for free!)

Interpreter/Debugger

Test Generator

Deductive Program Verifier

Compiler

Model Checker

Equivalence Checker

Symbolic Execution Engine

P4K
Potential App 3: Semantic coverage measurement

• “How much” of the language semantics is covered by the compiler tests suits?

• Similar technique for JavaScript ([Park et al, PLDI’15]) revealed:
  • Inconsistencies in JavaScript standard
  • Bugs in Web browsers
More Potential Apps

• Automatic conformance test generation
• Model checking
• Comprehensive network verification
  • by plugging controller programs written in C/Java/… without modification
• Equivalence check / translation validation
• Better language specification
  • Formalization itself might reveal problems in the specification
  • Use K rules in the language specification
    • or formalize the pseudo-code language
• [insert ideas here]
Conclusion

• Formal semantics matters
• P4K: Towards complete executable formal semantics of P4 in K
• Tools for P4 developers and designers based on the semantics
• Suggestion: Consider the framework for future versions of P4 language

• Check it out: https://github.com/kframework/p4-semantics/
• Learn more: http://www.kframework.org/
• Looking for ideas/collaborators
• Let’s get in touch: kheradm2@illinois.edu