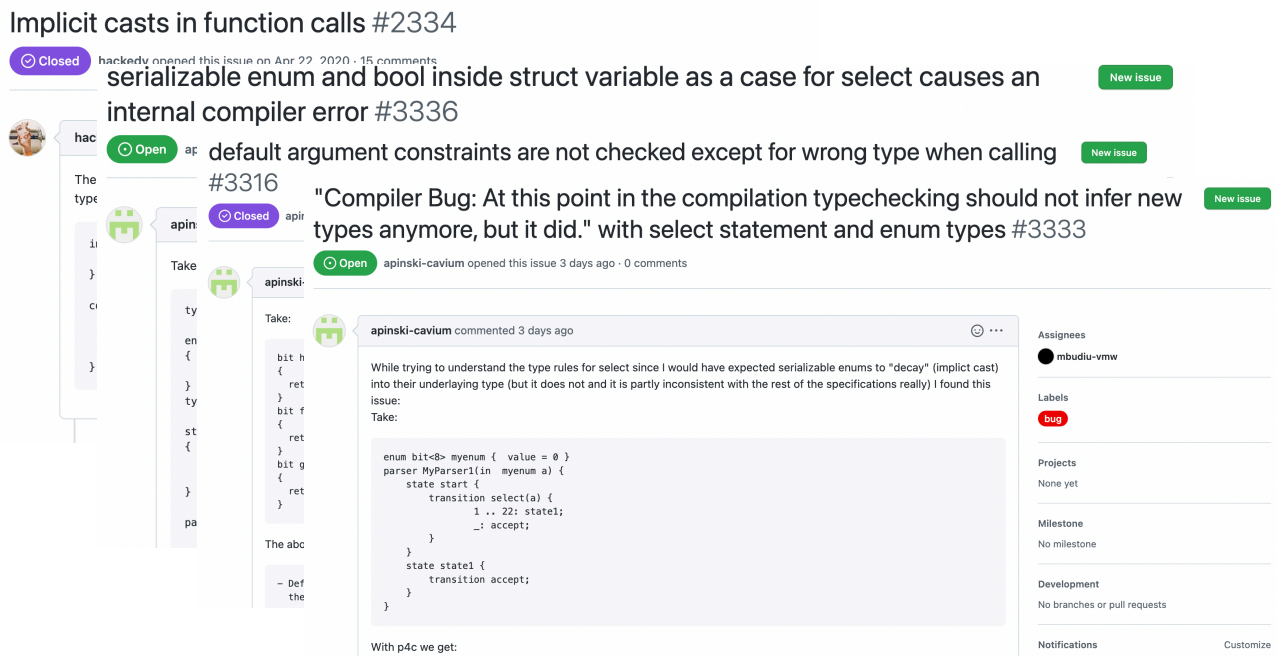
Differential Testing of P4 Implementations Using CI/CD

Parisa Ataei
Ryan Doenges
Chris Sommers
Nate Foster

Correctness of p4c

- How do we know the P4 reference implementation is correct? Not always!



- Idea: use differential testing to relate the behavior of p4c to other implementations of P4.

Petr4*

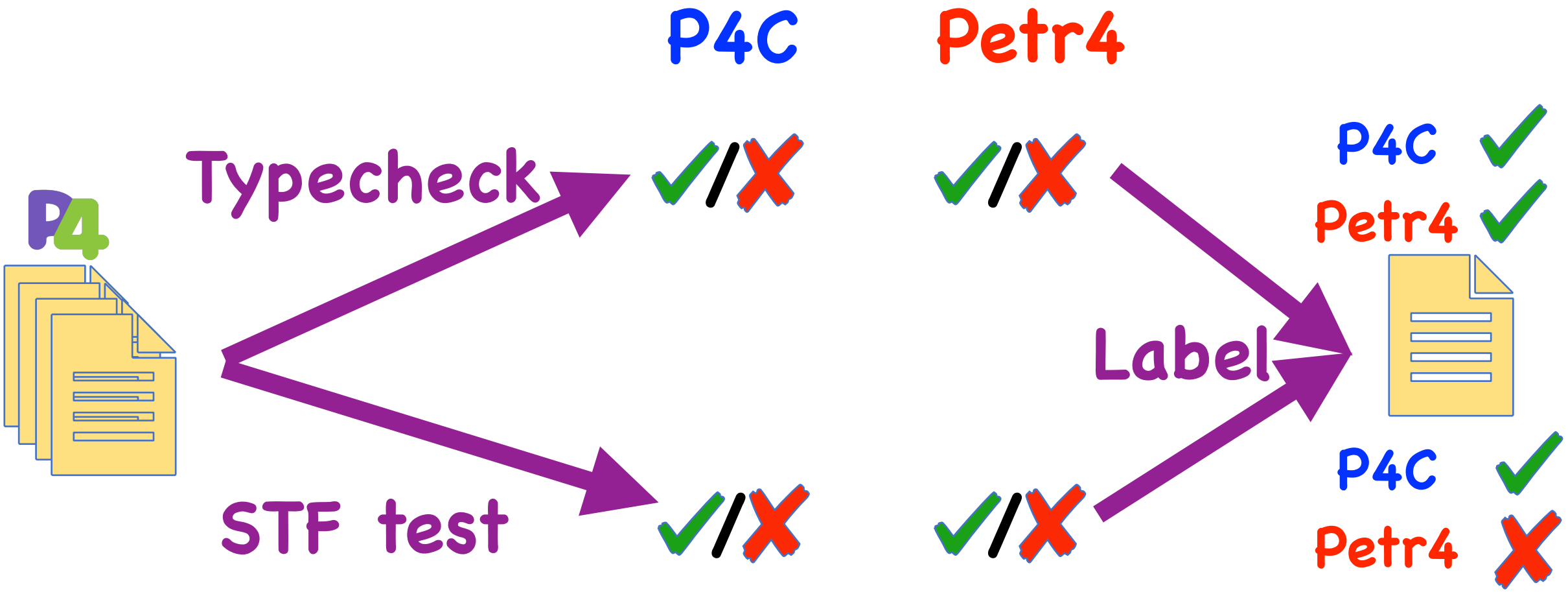
- A clean-slate implementation of P4
- Formal model of the type system and operational semantics
- Implemented in 58 KLoC of Coq and 12 KLoC of OCaml
- Provides several modes of operation: type checking and evaluation of STF test (given the input-output packet, evaluate the P4 program with the input packet and compare the result with the provided output packet)

*R. Doenges, M. Tahmasbi Arashloo, S. Bautista, A. Chang, N. Ni, S. Parkinson, R, Peterson, A, Solko-Breslin, A. Xu, and N. Foster. 2021. Petr4: formal foundations for p4 data planes. Proc. ACM Program. Lang. 5, POPL, Article 41 (January 2021), 32 pages. https://doi.org/10.1145/3434322

Methodology

- Take all the tests in the open-source test suite
- Run them on p4c and petr4 and compare the outputs
- What are the outputs?
  - Do they both parse a P4 program?
  - Do they both type check a P4 program?
  - Do they both process packets the same way on the available input-output test (STF test) of a program?
- Integrate this into P4 implementation's CI

Differential Testing Workflow



Example

```
struct intrinsic_metadata_t {
    bit<8> f0;
    bit<8> f1;
}

struct empty_t {}

control C<H, M>(
    inout H hdr,
    inout M meta,
    in intrinsic_metadata_t intr_md = {0, 0});

package P<H, M>(C<H, M> c);

struct hdr_t { }
struct meta_t { }

control MyC(inout hdr_t hdr, inout meta_t meta) {
    apply {}
}

P(MyC()) main;
```

P4C ✓

Petr4 ✗

```
Uncaught exception:

  ("could not solve type equality t1 = t2"
   (t1
    (SpecializedType
     ((base
       (TypeName
        (BareName (tags (M ""))
         (name
          ((tags
            (I
             (filename
              /petr4/ci-test/testdata/p4_16_samples/default-control-argument.p4)
             (line_start 13) (line_end ()) (col_start 16) (col_end 17)))
           (string C))))))
      (args
       ((TypeName
         (BareName (tags (M "")) (name ((tags (M "")) (string H0)))))
        (TypeName
         (BareName (tags (M "")) (name ((tags (M "")) (string M1))))))))))
   (t2
    (Control
     ((type_params ())
      (parameters
       (((annotations ()) (direction InOut) (typ (Struct ((fields ()))))
         (variable
          ((tags
            (I
             (filename
              /petr4/ci-test/testdata/p4_16_samples/default-control-argument.p4)
             (line_start 18) (line_end ()) (col_start 24) (col_end 27)))
           (string hdr)))
         (opt_value ()))
        ((annotations ()) (direction InOut) (typ (Struct ((fields ()))))
         (variable
          ((tags
            (I
             (filename
              /petr4/ci-test/testdata/p4_16_samples/default-control-argument.p4)
             (line_start 18) (line_end ()) (col_start 42) (col_end 46)))
           (string meta)))
         (opt_value ())))))))
```

Implementation

- Extend p4c's Docker image with petr4
- Build the Docker container in a GitHub action*
- Compare the output of running tests with the labels
  - If they match, approve new changes
  - If they don't match, notify the developer to look into it
    - If a new feature has been added or a bug has been fixed, change the labels accordingly

*https://github.com/verified-network-toolchain/petr4/blob/main/.github/workflows/p4-impls-diff-test.yml

Conclusion

- Implemented a differential testing tool that compares p4c and petr4 behavior (type checking and STF test) for a sample of P4 programs
- Integrated the tool into petr4's CI
- Plan to integrate it into p4c's CI
- Plan to keep a public log of the current differences between P4 language specification, p4c, and petr4
- Plan to add more tests such as PTF testing

Thank You

Questions?