



p4-constraints

Restrict what entries are allowed in P4 tables

Nate Foster <jnfoster@cs.cornell.edu>

Stefan Heule <heule@google.com>

Steffen Smolka <smolkaj@google.com>

Konstantin Weitz <konne@google.com>

P4LD WG

Aug 17, 2020

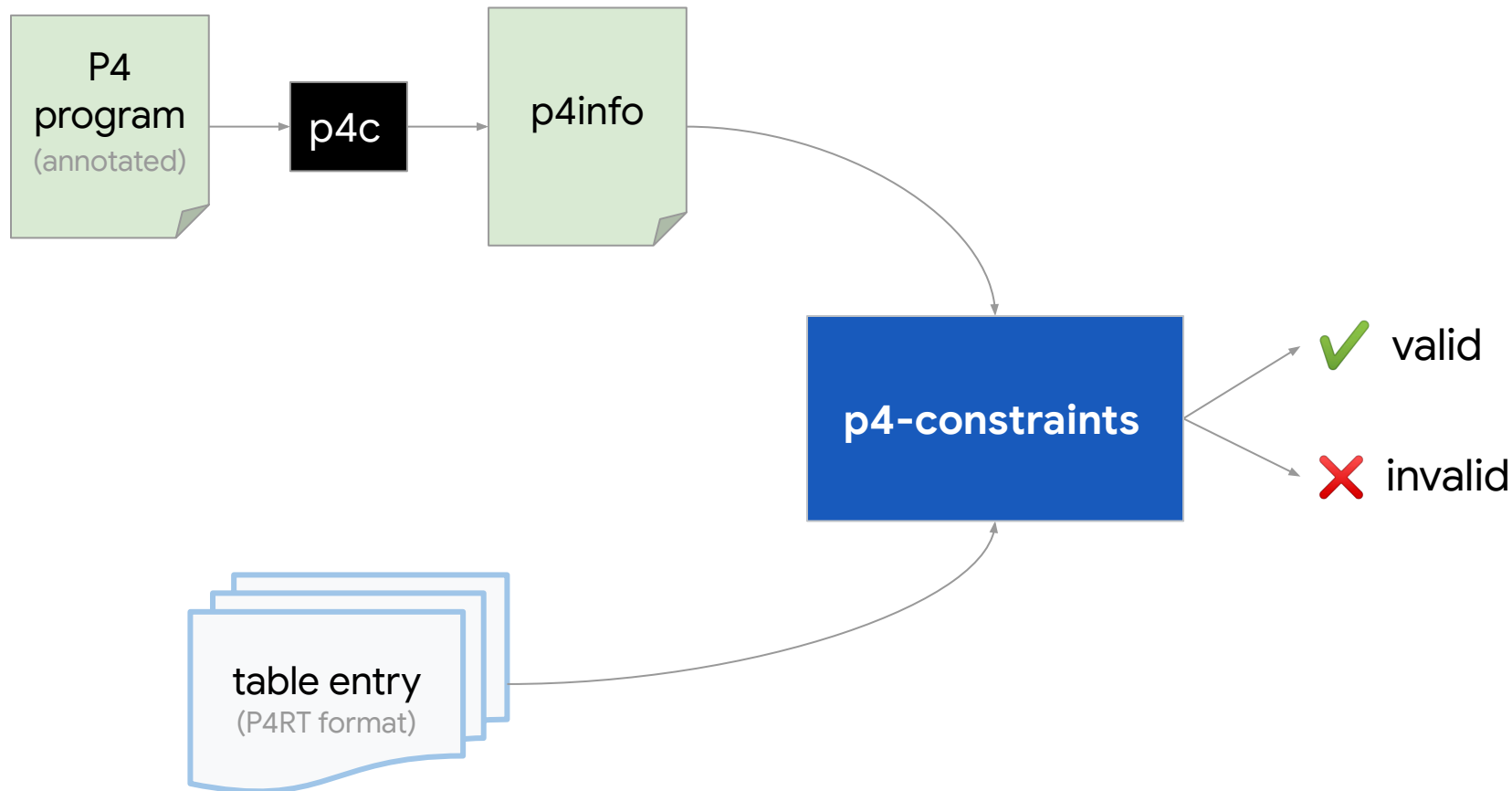
Restricting What Entries are Valid

Sometimes, not all table entries are valid.

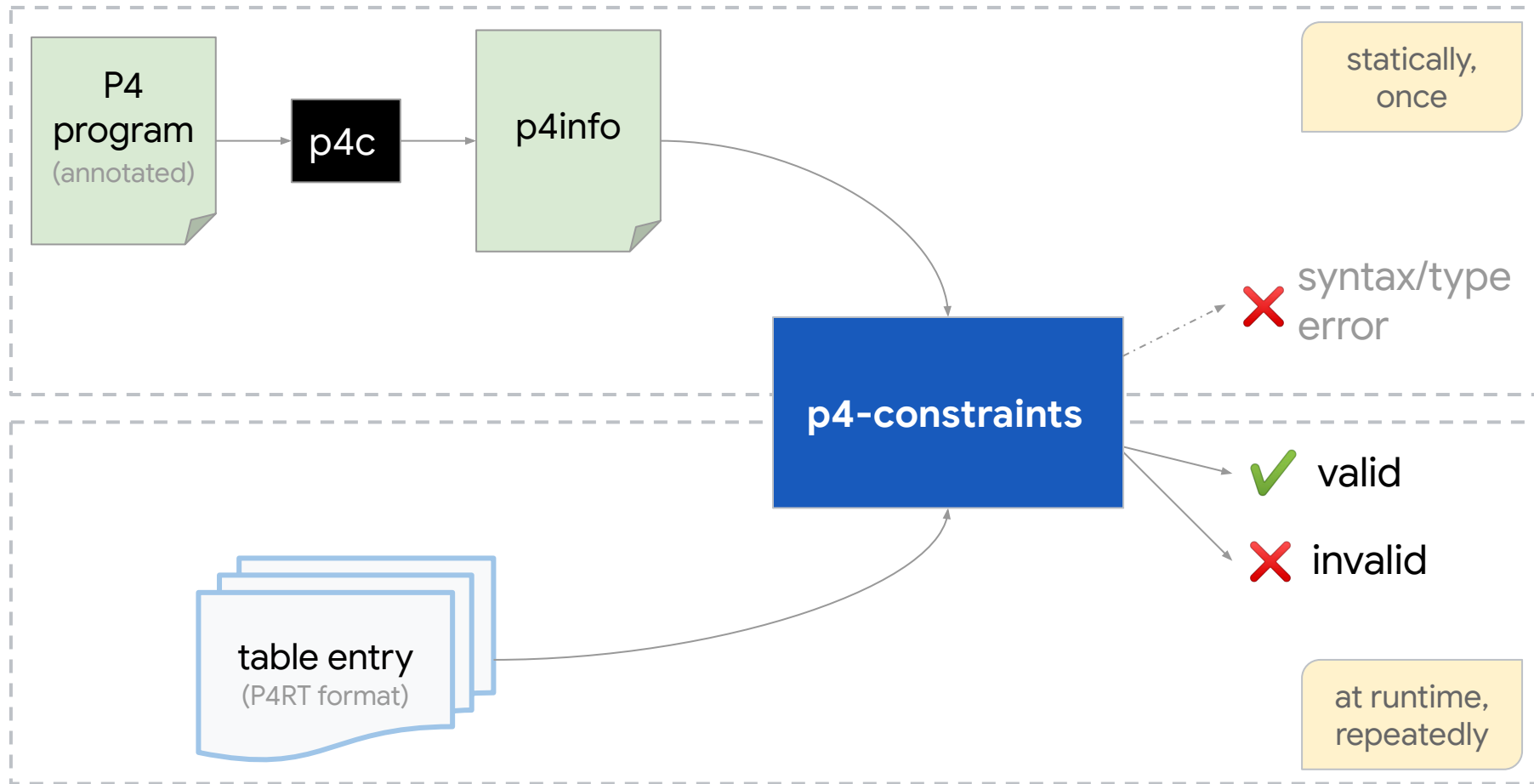
p4-constraints is a library that provides an annotation `@entry_restriction` that can express constraints on valid table entries.

```
@entry_restriction("  
    // Only match on IPv4 addresses of IPv4 packets.  
    hdr.ipv4.dst_addr::mask != 0 ->  
        hdr.ethernet.ether_type == IPv4_ETHER_TYPE;  
")  
table acl_table {  
    key = {  
        hdr.ethernet.ether_type : ternary;  
        hdr.ethernet.src_addr : ternary;  
        hdr.ipv4.dst_addr : ternary;  
        hdr.ipv6.dst_addr : ternary;  
    }  
    actions = { ... }  
}
```

Architecture



Architecture



What does p4-constraint provide?

```
// Translates P4Info to ConstraintInfo.  
//  
// Parses all tables and their constraint annotations into an in-memory  
// representation suitable for constraint checking. Returns parsed  
// representation or a nonempty list of error statuses if parsing fails.  
absl::variant<ConstraintInfo, std::vector<absl::Status>> P4ToConstraintInfo(  
    const p4::config::v1::P4Info& p4info);  
  
// Checks if a given table entry satisfies the entry constraint attached to its  
// associated table.  
absl::StatusOr<bool> EntryMeetsConstraint(const p4::v1::TableEntry& entry,  
    const ConstraintInfo& context);
```

Constraint language

// Constraints are expressions of type bool.

expression ::=

'true' 'false'	// Boolean constants.
numeral	// Numeric constants.
key	// Table keys.
'!' expression	// Boolean negation.
'-' expression	// Arithmetic negation.
'(' expression ')'	// Parentheses.
expression '::' id	// Field access (projection).
expression ('&&' ' ' '->' ';') expression	// Binary boolean operators.
expression ('==' '!=' '>' '>=' '<' '<=') expression	// Comparisons.

numeral ::=

(0[dD])? [0-9]+	// Decimal numerals.
0[bB] [0-1]+	// Binary numerals.
0[oO] [0-7]+	// Octary numerals.
0[xX] [0-9a-fA-F]+	// Hexadecimal numerals.

key ::= id ('.' id)*

id ::= [_a-zA-Z][_a-zA-Z0-9]*

// Table keys, e.g. "hdr.ipv4.dst".

// Identifiers.

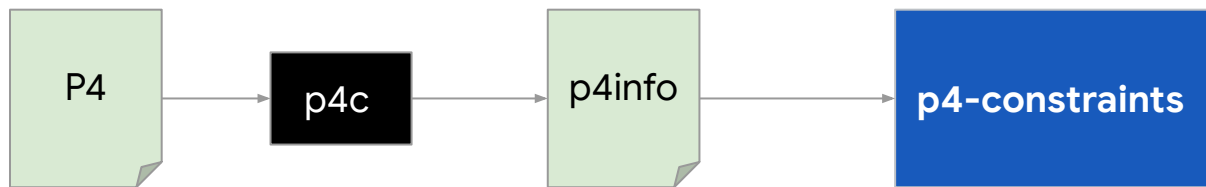
How do we envision p4-constraints to be used?

- As a **specification language** to further clarify the control plane API.
- In P4RT server implementations to **reject ill-formed table entries**.
- In the controller as a **defense-in-depth** check.
- During **testing** to check for valid vs invalid table entries.
 - To **guide a fuzzer** to valid table entries.

What's next?

- Give it a try! Give feedback, submit a PR, file an issue :-)
 - github.com/p4lang/p4-constraints
- Some ideas to make it even more expressive, e.g.:
 - Allow multi-table constraints and constraints on actions.
 - Allow constraints on strings (via `p4runtime_translation(.., string)`).
- Consider standardizing some time in the future if this is generally useful.

A case for a modular architecture



- Starting from p4info made p4-constraints easy to develop
 - standardized input format
 - easy to consume
- But p4info is not sufficient for all tools
 - e.g., it ignores control flow / action bodies
- What tools might a more general, standardized "p4info" enable?



Thank you

github.com/p4lang/p4-constraints

Google Cloud