What’s new in P4-16

Mihai Budiu

on behalf of the

P4 Language Design Working Group

P4 workshop

April 2023, Santa Clara, CA

Open-source Language Design Process

- Monthly meetings on zoom
  - First Monday of every month
  - See the p4.org calendar
- Any ONF member can participate
- Proposals and implementations tracked on github
- https://github.com/p4lang/p4-spec
- Discussion meetings available on github as well
- Changes prototyped in the open-source compiler
- We welcome new participants

P4-16 1.2.4 to be released

- Yearly release cadence
- Previous release was in July 2022
- Many clarifications
- A few small improvements
- Fully backwards compatible with 1.2.3

Clarifications

- Driven by formal modelling of P4 syntax and semantics
  - Several academic teams (Princeton, Cornell)
- “list expressions” -> “tuple expressions”
- Semantics of ‘exact’, ‘ternary’, ‘lpm’ is now part of the spec
- Semantics of “negative” ranges such as 5..2 (empty)
- Many small other fixes

New features

- static_assert – compile-time assertions

```
const bool _check = static_assert(V1MODEL_VERSION > 20180000,
                                  "Expected a v1 model version >= 20180000");
```

- Allow comparisons for tuples
- Optional trailing commas

```
enum E {
#if SUPPORT_A
    a,
#endif
    b,
    c,
}
```

A new list<> type

- Currently we only have list literals
- Can be used in constructor parameters

```
extern E {
    E(list<pair_t> data);
    void run();
}

control c() {
    E((list<pair_t>) {{2, 3}, {4, 5}}) e;
    apply {
        e.run();
    }
}
```

More kinds of expressions

- Invalid header and invalid union literals

  `{#}` (a single token)

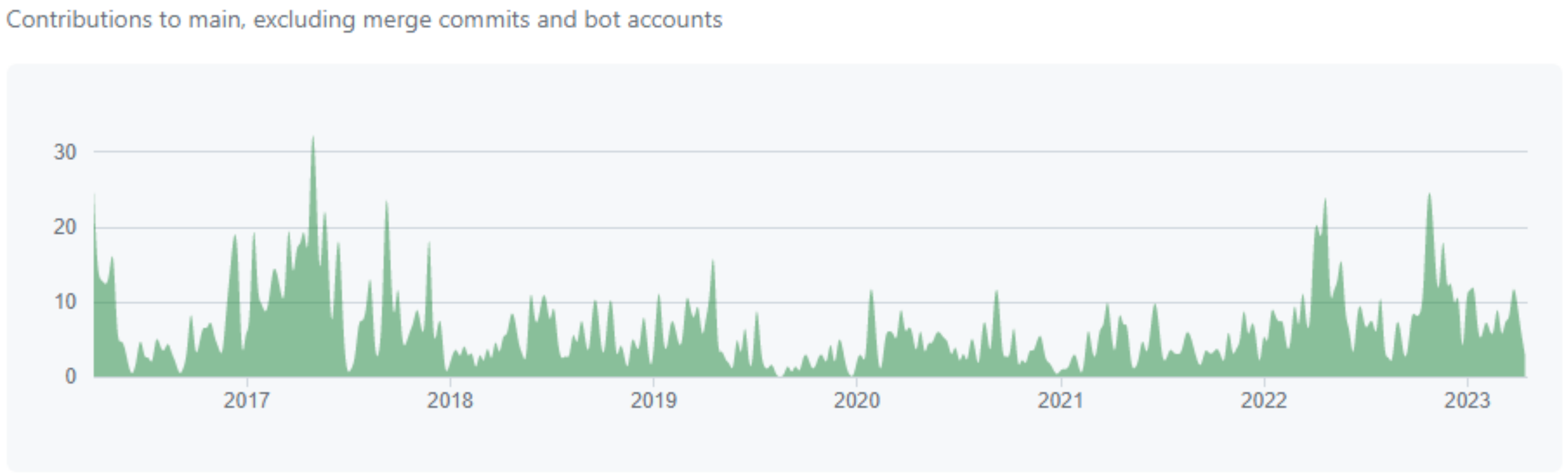
- Stack initializers

```
H<bit<32>>[3] s;
   s = (H<bit<32>>[3]){ {0, 1},
                        {2, 3},
                        (H<bit<32>>){#} };
```

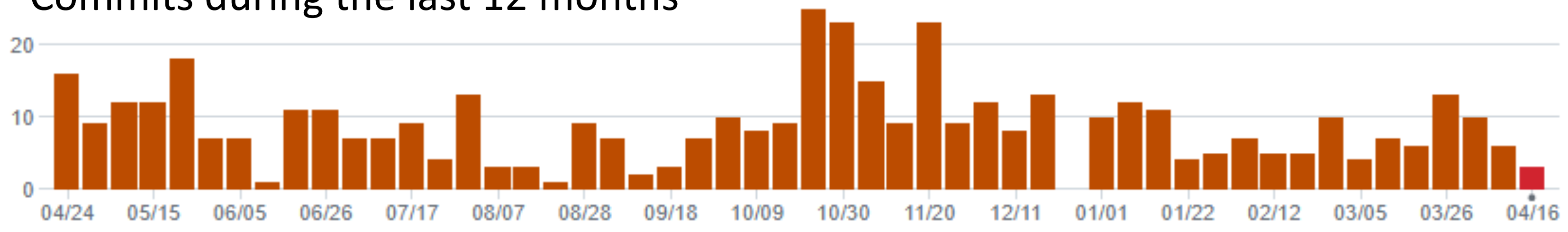
Non-const table entries

```
table t {
   …
   largest_priority_wins = false;
   priority_delta = 10;
   entries = {
     const priority=10: (0x01, 0x1111 &&& 0xF    ) : a(1);
                        (0x02, 0x1181            ) : a(2);
                        (0x03, 0x1000 &&& 0xF000) : a(3);
     const              (0x04, 0x0210 &&& 0x02F0) : a(4);
           priority=40: (0x04, 0x0010 &&& 0x02F0) : a(5);
                        (0x06, _                 ) : a(6);
}}
```

Compiler implementation



Commits during the last 12 months



Open-source compiler contributions

- Several complex O/S backends contributed
  - DPDK backend (generates DPDK assembly)
  - P4 to ebpf/PSA – production quality
  - Testing backend – P4 test generation using symbolic execution
  - PTF (Packet Test Framework) Python-based testing for many backends
- Many bug fixes and improvements
  - Default initializers fully implemented (…)
  - Code style enforced across all languages
  - Improvements in build process
  - Loop unrolling for parsers