SuperP4: Preprocessor-Aware Syntax and Semantic Analysis for P4 Programs

Presenter:
Kaarthik A. Alagappan

Collaborators:
Robert Soule
Mario Baldi
Paul Gazzillo
Antonio Carzaniga
Ali Fattaholmanan

P4 Programs

- Developers use the preprocessor to make a single program configurable

```
    state parse_ethernet {
        packet.extract(hdr.ethernet);
        ...
        transition select(hdr.ethernet.eth_type){
            ...
#ifdef WITH_IPV6
            ETHERTYPE_IPV6: pre_parse_ipv6;
#endif // WITH_IPV6
            default: accept;
        }
    }
```

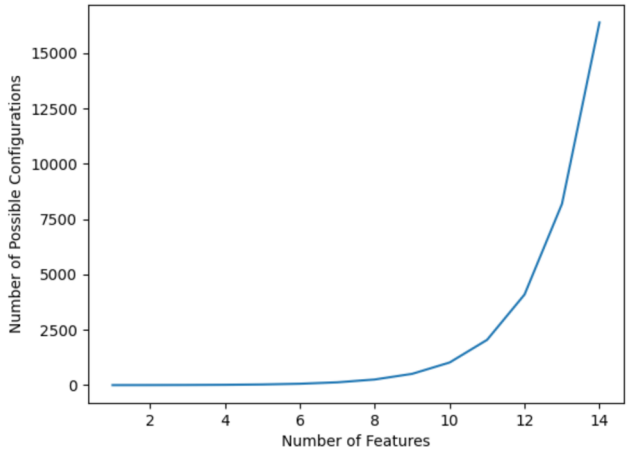
```
#ifdef WITH_IPV6
    // Intermediate state to set is_ipv6
    state pre_parse_ipv6 {
        fabric_metadata.is_ipv6 = _TRUE;
        transition parse_ipv6;
    }
    state parse_ipv6 {
        packet.extract(hdr.ipv6);
        ...
        transition select(hdr.ipv6.next_hdr) {
            ...
            default: accept;
        }
    }
#endif // WITH_IPV6
```

```
struct parsed_headers_t {
    ...
    ipv4_t ipv4;
#ifdef WITH_IPV6
    ipv6_t ipv6;
#endif // WITH_IPV6
    tcp_t tcp;
    ...
}
```

- Onos-satellite's fabric uses 11 preprocessor directives
  - 2,047 possible combination of configurations

Challenges of Many Configurations

- Existing verifications tools like P4v, Vera, and Assert-P4 work only on a single-configuration of a P4 program
- Expensive to verify all configurations (>2k for fabric program)
  - Thus, developers only check a small set of configurations, potentially missing cross-configuration bugs



Potential Configuration Related Bug

```
control Ingress (...) {
    apply {
#ifdef FEATURE_1
        ControlFeat1.apply(...);
#endif // FEATURE_1

#ifdef FEATURE_2
        PortCounterFeat.apply(...);
#endif // FEATURE_2
    }
}
```

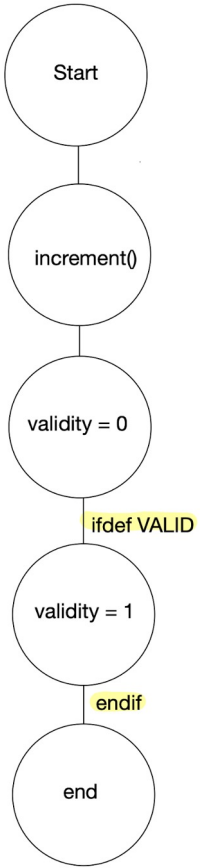
```
control ControlFeat1(...) {
    apply {
        if (inner_udp.isValid()) {
            udp.setInvalid();
        }
    }
}

control PortCounterFeat(...) {
    apply {
        if (udp.isInValid()) {
            udp.setValid();
        }
    }
}
```

Conflicting statements: PortCounterFeat will override ControlFeat1's changes if both are enabled (udp will be valid at the end)

SuperP4

- A preprocessor-aware parser for P4 programs
  - Perform analysis on P4 program with that preprocessor information
- SuperP4 produces a preprocessor-aware AST

```
void increment() {
    bit<16> validity = 0;
    #ifdef VALID
    validity = 1;
    #endif
}
```



SuperP4

- Goal is to build an analyzer that will perform cross-configuration analysis and warn developer of potential conflicts
  - Checking if the same value is modified under multiple configurations
  - Type errors within and across configurations
  - Incompatible configurations

SuperP4: Current Status

- Able to parse P4 program while retaining preprocessor information
- Currently developing the type checker and call graph analysis with the preprocessor context
  - Where each node will be tagged with the preprocessor configuration it is present under
  - Analyze a P4 program with all configurations at once

Thank You

Contact: kaarthik@knights.ucf.edu