P4 Formatter

Nitish Kumar

Mentors - Bili Dong, Fabian Ruffy

GsoC 24

Nitish Kumar

- Undergrad at IIT Kharagpur
- Open source enthusiast, Notable contributions to: Rust's coreutils, Nixos, spack etc.
- Interested in Compilers, Networking and Distributed Systems.
- Worked as LFX mentee at RISC V.
- Github: github.com/snapdgn
- Hobbies: Badminton, Tabla.

Issues with the Project:

- **Missing Comments**: AST lost original comments.
- **Limited Flexibility**: `toP4` (p-printer) offered minimal formatting options and lacked customization.

Objectives

- Modify the AST/IR to attach comments from the original source code.
- Modify the pretty-printer to print the newly attached comments.
- Add common formatting options.
- Provide a minimally functional formatter with the above features.

Implementation

How `go/ast` does it ?

Comments are stored by their byte offset in the file instead of attaching it to nodes, so re-arranging nodes breaks the output.

```
func main(){
    var a int    // foo
    var b string // bar
}
```

```
func main() {
    // foo
    var b string
    var a int
    // bar
}
```

Ref: https://github.com/dave/dst

Implementation

- Inspired by how `bazel tools` preserves comments.
- Lexer saves all the comments in a global list, as a part of `` `InputSources` `` class.
- Each AST node embeds a `Comments` struct to store comments.
- Comments are of two types
  - Prefix: comments before a construct
  - Suffix: Inline comments
- Two passes: pro-order and post-order to attach comments to IR nodes.

Implementation

- Each node has a start & end position, so does each comment.
- For the independent line comment: attach to AST node right after it in preorder traversal.
- Inline trailing suffix comment: attach it to AST node right before it in a postorder traversal.

```
// line comment
struct headers {
   ethernet_t ethernet; // suffix comment
   ipv4_t     ipv4;
}
```

- Two vector(prefix & suffix) to store the comments on each node
- Extra overhead was being paid on every node.
- Could have potentially slow down the compiler / increased memory consumption.
- Shifted the attached comment to a local side map, which stores the node-id and the associated comments and attaches it while traversing on a second pass.

Results

```
typedef bit<9> egressSpec_t;
typedef bit<48> macAddr_t;
typedef bit<32> ip4Addr_t;
header ethernet_t {
   macAddr_t dstAddr;
   macAddr_t srcAddr;
   bit<16>   etherType;
}
```

Before: Comments stripped in the AST.

```
typedef bit<9> egressSpec_t; //typedef egress
typedef bit<48> macAddr_t; // typedef macaddr
// before ipv4addr
typedef bit<32> ip4Addr_t;
header ethernet_t {
   macAddr_t dstAddr;
   // TypeName: srcAddr
   macAddr_t srcAddr;
   bit<16>   etherType; // inline Type_Bits
}
```

After: Comments preserved and attached to respective Nodes.

Additional Works

Also wrote a reference checker for the `p4fmt` formatter; utilizes golden tests to check expected output.

Related PRs

- https://github.com/p4lang/p4c/pull/4710
- https://github.com/p4lang/p4c/pull/4845
- https://github.com/p4lang/p4c/pull/4778
- https://github.com/p4lang/p4c/pull/4718
- https://github.com/p4lang/p4c/pull/4862
- https://github.com/p4lang/p4c/pull/4795
- https://github.com/p4lang/p4c/pull/4887
- https://github.com/p4lang/tutorials/pull/570

Limitations

- Free-floating comments are not handled.

```
// struct comment
struct headers {
...
}

// free floating

// control block checksum
control MyVerifyChecksum(inout headers hdr, inout metadata meta) {
   ...
}
```

- Things like comments inside function definition arguments are unhandled.

```
control MyIngress(/* argument */ inout headers hdr, inout metadata meta, inout standard_metadata_t standard_metadata)

control MyIngress( inout headers hdr /* argument */, inout metadata meta, inout standard_metadata_t standard_metadata)
```

Future Works

- Come up with a standardized set of formatting rules and coding guidelines by discussing it with the P4 Community.
- More sophisticated algorithm to handle free-floating comments & code constructs with multiple attachment points.

```
const  /* (1) */ bit<16> /* (2) */  TYPE_IPV4 /* (3) */ = /* (4) */ 0x800 /* (5) */;
```

- Support for formatting options like line wrapping, indentation etc.
- Provide configuration options for customising the formatting style, through a config file & cmd line flags/params.

Thank You!!