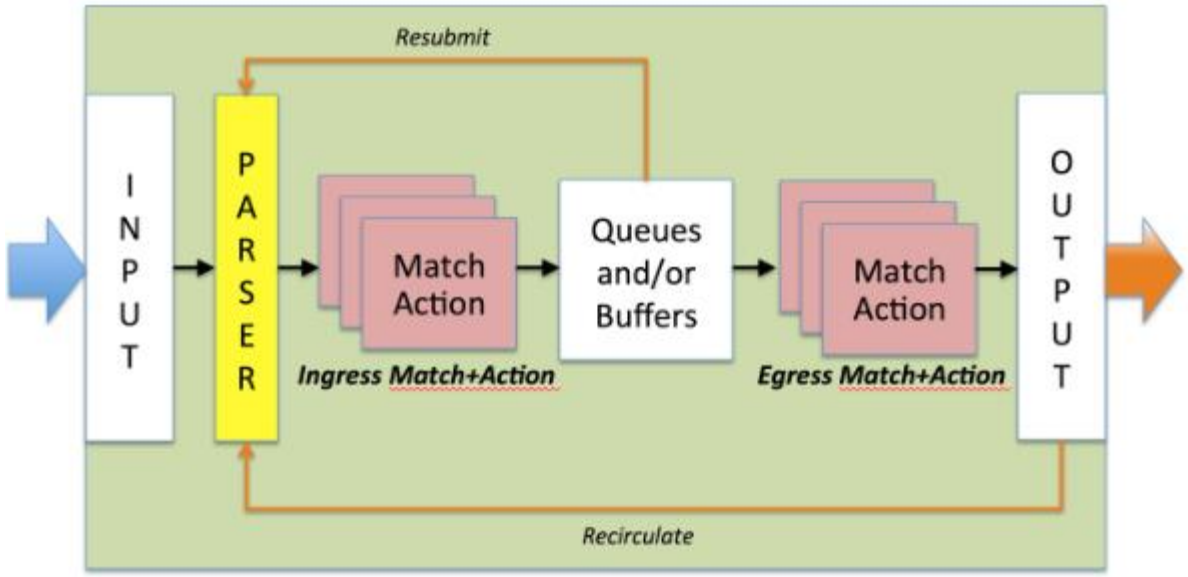
Recirculate in P4-16

using v1model.p4

A way to preserve metadata and packet data

- recirculate
- resubmit
- clone



These are very similar, so we only discuss recirculate.

What is the problem?

- The way v1model.p4 is designed it ***cannot*** implement these operations
- The compiler used to "work", but it was by accident
  - For P4-14 programs
  - For P4-16 programs written using v1model
- The v1model API is: recirculate<T>(in T data);
- This cannot work because for the compiler the following two are equivalent:

```
recirculate(a);
        and
b = a;
recirculate(b);
```

How can we do it?

- We have (at least) three choices
  1. Using explicit input/outputs
  2. Using annotations on fields of the user metadata structure
  3. Using pairs of functions
     - recirculate, recirculate_receive

- All solutions will require people to rewrite P4-16 programs that use these functions
- But we don’t need changes for P4-14 programs or P4-16 programs that do not need these features

1. Using an explicit input/output

```
parser Parser<H, M, RECIRC>(
    packet_in buffer,
    out H parsed_hdr,
    inout M user_meta,
    in RECIRC recirc_meta);

control Egress<H, M, RECIRC>(
    inout H hdr,
    inout M user_meta,
    in    standard_metadata_t std,
    out   RECIRC recirc_meta)
```

- recirculate() needs no arguments
  - (it could also be a Boolean standard metadata output field)
- Semantics is similar to P4-14: recirculate happens at the end of the pipeline
- We would add a new V1Switch package

```
Package V1Switch<H, M, RECIRC>(Parser<H, M, RECIRC> p, …)
```

2. Using annotations on the user metadata

```
struct metadata {
    bit<32> field0;
    @recirculate(1,2)
    bit<32> field1;
}
```

- recirculate(1) now has no arguments
- Implemented in https://github.com/p4lang/p4c/pull/1698
- No other changes required to model
- Semantics similar to P4-14: recirculation happens at the end of the pipeline
- Cannot reuse metadata structures from libraries
- Cannot recirculate standard metadata
  - Should we even allow recirculating standard metadata?
  - BMv2/P4-14 allows you to ask for it, but doesn’t handle it for many fields

3. Using explicit functions for input & output

```
// Model
extern RecirculateChannel<T> {
   bool valid();
   T recirculate_receive();
   void recirculate(in T data);
}
// user program
RecirculateChannel<S> rc;
parser p(…) {
   state start {
      S s = rc.recirculate_receive();
       … }}
control ingress(…) {
     apply {
         rc.recirculate({ … }); }}
```

- Semantics different from P4-14: recirculation values are captured at the recirculate function call execution
- Unfortunately today list-expressions cannot be left-values, to the user will have to copy the data from s
  - Or we change the language to allow left-values from list expressions. Could be useful.

How does PSA do it?

- Uses solution 1 above
- However, this metadata is not visible in ingress, so the user has to copy it to the user_meta

```
parser IngressParser<H, M, RESUBM, RECIRCM>(
    packet_in buffer,
    out H parsed_hdr,
    inout M user_meta,
    in psa_ingress_parser_input_metadata_t istd,
    in RESUBM resubmit_meta,
    in RECIRCM recirculate_meta);

control Ingress<H, M>(
    inout H hdr, inout M user_meta,
    in    psa_ingress_input_metadata_t  istd,
    inout psa_ingress_output_metadata_t ostd);

control IngressDeparser<H, M, CI2EM, RESUBM, NM>(
    packet_out buffer,
    out CI2EM clone_i2e_meta,
    out RESUBM resubmit_meta,
    out NM normal_meta,
    inout H hdr,
    in M meta,
    in psa_ingress_output_metadata_t istd);
```

P4-14: Field lists

- P4-14 construct to specify a list of fields
- Recirculation is implemented using a function
  recirculate(field_list)
- Assumes all values are global: same fields visible in ingress, egress, etc.
  - Not true in v1model or PSA
- A form of “call-by-name”
- Supported natively by BMv2

- No equivalent in P4-16
- Not a practical language construct for many architectures