Compiler for P3: A Language to Specify Protocol-Independent Packet Parsers
$(Draft)$

Compiler Group, System Software and Software Engineering Laboratory
Department of Computer Science and Technology, Tsinghua University

July 25, 2020

1 Introduction

This document presents a domain-specific language P3 for reconfigurable Protocol-independent Packet Parsers.

For the requirement to facilitate the implementation of a high-security network, we design the language from the perspective of high trustworthiness, including the formal definition of type system and operational semantics of the language and its trusted compiler architecture. Based on the full understanding of the basic requirements of the reconfigurable hardware, from the view of hardware-software co-design, we finally defined the core characteristics of P3 language and its trusted compiler architecture called P3C. As the reconfigurable packet parser is an important part of SDN and programmable data plane, implementing the trusted compiler architecture of P3C will be of great significance to the security of SDN.

To build trustworthy compilers, one approach is to specify the source, target, intermediate languages, and the compilation algorithms formally in an interactive proof assistant such as Coq [1, 2], and then mechanically prove a semantic preserving relation between the source and target. CompCert [7, 6, 4], a formally verified compiler from a large subset of C language Clight [3] to assembly code for several machines, is one of the most successful efforts in this way. The proof can be mechanically checked, yielding the highest level of assurance we can hope to achieve[9].

Translation validation [13] is another approach to certify compilers. There have been many efforts in using the translation validation approach for synchronous languages such as Signal [13, 12, 11, 10]. This approach is also used to verify the translation from Simulink to C [14].

Fig.1 shows the architecture of the P3C compiler. In the project 2017ZX01030-301-003, we are required to implement the P3C compiler. However, our research include the verification of the compiler. Referring to the dotted lines in Fig.1, the design of the compiler concerned the verification on the *parsing*, the *type*

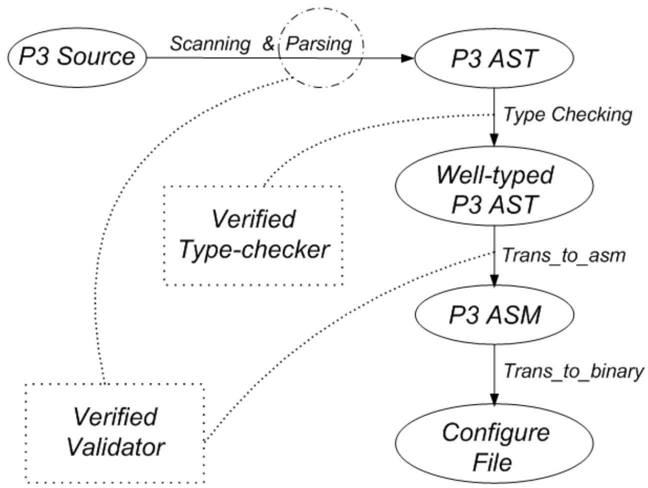

Figure 1: The compiler architecture of P3C

checking and the translation form the P3 abstract syntax tree (AST) to the P3 assembly (ASM).

The rest of the document is organized as follows. In Section 2, we give the syntax of P3, and the informal interpretation of a P3 specification by examples. In Section 3, we present the syntax of a special P3 assembly language, and the format of a target configuration file. In Section 4, the definition of the P3 AST, and the implementation and verification of the parsing, in the P3C compiler, are described. The definition of type system for P3 AST and the type checking in the P3C compiler are presented in Section 5. In Section 6, we specify the abstract syntax of the P3 assembly, and describe the translation from the P3 AST to the P3 assembly and the translation from the P3 assembly to the configuration file. In Section 7, we define the operational semantics of the P3 AST, the operational semantics of the P3 assembly, and then discuss the verification of the translation from the P3 AST to the P3 assembly. The document is concluded in Section 8.

2 The source language : P3

2.1 Syntax of P3

⟨*parser_spec*⟩ ::= ⟨*parameters*⟩ { ⟨*decl*⟩ }

⟨*parameters*⟩ ::= ⟨*layer_reg_len*⟩ ⟨*cell_reg_len*⟩ ⟨*protocol_set*⟩ ⟨*layer_set*⟩

⟨*layer_reg_len*⟩ ::= **lreglen** '=' *Integer* **bytes** ';'

⟨*cell_reg_len*⟩ ::= **creglen** '=' *Integer* **bytes** ';'

⟨protocol_set⟩ ::= **pset** '=' '{' ⟨id_list⟩ '}' ';'

⟨layer_set⟩ ::= **lset** '=' '{' ⟨id_list⟩ '}' ';'

⟨id_list⟩ ::= IDENT { ',' IDENT }

⟨decl⟩ ::= ⟨const_decl⟩
| ⟨reg_acc_set⟩
| ⟨protocol_decl⟩
| ⟨layer_action⟩

⟨const_decl⟩ ::= **const** IDENT '=' ⟨expr⟩ ';' // ⟨expr⟩ must be a constant expression

⟨const⟩ ::= IDENT // constant identifiers
| Integer //integer constants, signed 32 bits
| Hexadecimal //hex constants, such as 0x88a8, 0xFFFFFF, 0x89,0x103
| Bits //binary constants, such as 001001, 100, 0, 1, 1100, 00, 11111

⟨protocol_decl⟩ ::= **protocol** ⟨protocol_id⟩ '{' ⟨protocol⟩ '}'

⟨protocol_id⟩ ::= IDENT

⟨protocol⟩ ::= ⟨fields⟩ ⟨p_stmts⟩

⟨fields⟩ ::= **fields** '=' '{' ⟨field⟩ { ⟨field⟩ } [⟨option_field⟩] '}'

⟨field⟩ ::= IDENT ':' ⟨const⟩ ';'

⟨option_field⟩ ::= **options** ':' '*' ';'

⟨p_stmts⟩ ::= { ⟨p_stmt⟩ }

⟨p_stmt⟩ ::= ⟨if_else_p_stmt⟩
| **next_header** '=' ⟨protocol_id⟩ ';'
| **length** '=' ⟨const⟩ ';'
| **bypass** '=' ⟨const⟩ ';'
| ⟨action_stmt⟩

⟨if_else_p_stmt⟩ ::= **if** '(' ⟨expr⟩ ')' ⟨p_stmts⟩ { **elseif** '(' ⟨expr⟩ ')' ⟨p_stmts⟩ }
[**else** ⟨p_stmts⟩] **endif**

⟨layer_action⟩ ::= ⟨layer_id⟩ '{' ⟨local_reg_decl⟩ ⟨l_decls⟩ ⟨l_actions⟩ '}'

⟨layer_id⟩ ::= IDENT

⟨l_decls⟩ ::= { ⟨l_decl⟩ }

⟨l_decl⟩ ::= ⟨protocol_id⟩ ⟨id_list⟩ ';'

⟨*local_reg_decl*⟩ ::= [⟨*cella_regs*⟩] [⟨*cellb0_regs*⟩] [⟨*cellb1_regs*⟩]

⟨*cella_regs*⟩ ::= **ARegisters** '{' { ⟨*reg_acc_set*⟩ } '}'

⟨*cellb0_regs*⟩ ::= **B0Registers** '{' { ⟨*reg_acc_set*⟩ } '}'

⟨*cellb1_regs*⟩ ::= **B1Registers** '{' { ⟨*reg_acc_set*⟩ } '}'

⟨*l_actions*⟩ ::= [⟨*cella_actions*⟩] [⟨*cellb0_actions*⟩] [⟨*cellb1_actions*⟩]

⟨*cella_actions*⟩ ::= **cellA** '{' { ⟨*l_stmt*⟩ } '}'

⟨*cellb0_actions*⟩ ::= **cellB0** '{' { ⟨*l_stmt*⟩ } '}'

⟨*cellb1_actions*⟩ ::= **cellB1** '{' { ⟨*l_stmt*⟩ } '}'

⟨*l_stmt*⟩ ::= ⟨*if_else_l_stmt*⟩
| **next_header** '=' ⟨*protocol_id*⟩ ';'
| **length** '=' ⟨*expr*⟩ ';'
| **bypass** '=' ⟨*const*⟩ ';'
| ⟨*action_stmt*⟩

⟨*l_stmts*⟩ ::= { ⟨*l_stmt*⟩ }

⟨*if_else_l_stmt*⟩ ::= **if** '(' ⟨*expr*⟩ ')' ⟨*l_stmts*⟩ { **elseif** '(' ⟨*expr*⟩ ')' ⟨*l_stmts*⟩ }
[**else** ⟨*l_stmts*⟩] **endif**

⟨*expr*⟩ ::= ⟨*atom*⟩ //atom expressions
| ⟨*unop*⟩ ⟨*expr*⟩ //unary expressions
| ⟨*expr*⟩ ⟨*binop*⟩ ⟨*expr*⟩ //binary expressions
| ⟨*expr*⟩ '.' *IDENT* //access to a field in a protocol
| ⟨*expr*⟩ '[' ⟨*expr*⟩ ']' //access to a bit of a field or register
| ⟨*expr*⟩ '[' ⟨*expr*⟩ ':' ⟨*expr*⟩ ']' //access to a section of a field or register
| '(' ⟨*expr*⟩ ')'
| *IDENT* '.' **length**

⟨*atom*⟩ ::= ⟨*const*⟩ //const expressions
| *IDENT* //all kinds of access name , ex., field or register access name

⟨*unop*⟩ ::= **int** //convert hexadecimal or binary numbers to integers(signed 32 bits)
| **not** //logical negation
| '~' //bit-wise negation

⟨*binop*⟩ ::= '+' //addition
| '-' //subtraction
| '*' //multiplication
| '/' //division integer
| '%' //remainder
| '&&' //logical and
| '||' //logical or
| '&' //bit-wise and
| '|' //bit-wise or

```
| '^'        //bit-wise exclusive or
| '=='       //equality between any type of values
| '<>'       //inequality between any type of values
| '<'        //lower on numerics
| '>'        //greater on numerics
| '<='       //lower or equal on numerics
| '>='       //greater or equal on numerics
| '<<'       //shift left
| '>>'       //shift right
| '++'       //concatenation of 2 binary bits' or 2 hexadecimal digits'
| hexes   //convert a binary number or an integer to a hexadecimal number
| bits    //convert an integer or a hexadecimal number to a binary number

⟨action_stmt⟩ ::= action '=' '{' ⟨instructions⟩ '}'
               | ⟨instruction⟩

⟨instructions⟩ ::= { ⟨instruction⟩ }

⟨instruction⟩ ::= ⟨set⟩
               | ⟨mov⟩
               | ⟨lg⟩
               | ⟨eq⟩

⟨set⟩ ::= set ⟨tgt_reg_acc_name⟩ ',' ⟨expr⟩ ';'

⟨mov⟩ ::= mov ⟨mov_reg_acc_name⟩ ',' ⟨expr⟩ ';'

⟨lg⟩ ::= lg ⟨tgt_reg_acc_name⟩ ',' ⟨expr⟩ ',' ⟨expr⟩ ';'

⟨eq⟩ ::= eq ⟨tgt_reg_acc_name⟩ ',' ⟨expr⟩ ',' ⟨expr⟩ ';'

⟨reg_acc_set⟩ ::= ⟨reg_acc_name⟩ '=' IRF '[' ⟨expr⟩ ':' ⟨expr⟩ ']' ';'
               | ⟨reg_acc_name⟩ '=' IRF '[' ⟨expr⟩ ']' ';'

⟨tgt_reg_acc_name⟩ ::= ⟨reg_acc_name⟩
                    | ⟨tgt_reg_acc_name⟩ '[' ⟨expr⟩ ':' ⟨expr⟩ ']'
                    | ⟨tgt_reg_acc_name⟩ '[' ⟨expr⟩ ']'

⟨mov_reg_acc_name⟩ ::= ⟨tgt_reg_acc_name⟩
                    | ⟨mov_reg_acc_name⟩ '++' ⟨tgt_reg_acc_name⟩

⟨reg_acc_name⟩ ::= IDENT
```

2.2 Example: A P3 specification

An example of a P3 specification is shown in Fig.2, Fig.3, Fig.4, Fig.5 and Fig.6.

In Fig.2, some global declarations are given, including the length of the global IRF working register set by *lreglen*, the length of the cell IRF register set by *creglen*, a set of protocol identifiers (ethernet, ieee802-1qTag, ipv4, and etc.), a set of layer action identifiers (l2, l2s, l3, l3s, l4, listed in order-dependence), a

```
lreglen = 72 bytes;

creglen = 24 bytes;

pset = {
    ethernet,
    ieee802-1qTag,
    ipv4,
    mpls,
    ieee802-1OuterTag,
    lldp,
    trill,
    qcn,
    igmp,
    ospf,
    pim,
    tcp,
    udp
};

lset = {
    l2,
    l2s,
    l3,
    l3s,
    l4
};

const global_IRF_len = 64;

IRF_gp_reg0_2b = IRF[global_IRF_len+1:global_IRF_len];
IRF_gp_reg1_2b = IRF[global_IRF_len+3:global_IRF_len+2];
}
```

Figure 2: Example of global declarations in P3

global constant declaration identified by *global_IRF_len* and the global register declarations identified by *IRF_gp_reg0_2b* and *IRF_gp_reg1_2b*.

In Fig.3, the layer-action identified *l2* is specified, including the declarations of protocol instances (an *ethernet* instance *eth*, an *ieee802-1qTag* instance *vlan* and an *ieee802-1OuterTag* instance *qinq*), the declarations of registers and guarded actions in the cell A and cell B0.

In Fig.4 are other two layer-action specifications identified by *l2s* and *l3*. Here, the *l2s* layer-action specification is empty. In the *l3* layer-action specification, an *ipv4* protocol instance *v4* as declared, and the declarations of registers and guarded actions respectively in the cell A, cell B0 and cell B1.

Fig.5 and Fig.6 present several examples of protocol specification, which are identified by *ethernet*, *ieee802-1qTag*, *ieee802-1OuterTag*, *mpls* and *ipv4* respectively. They all include the specification for fields, the offset length set by *len* and the guarded actions.

The components of the example above will be used in the following sections.

3 The target language

The target of the P3C compiler is a hardware configuration file consisting of several special tables, whose syntax is referred to the sub-section 3.2.

```
l2 {
    ARegisters {
        IRF_l2_send_to_cpu_8b = IRF[15:8];
        IRF_tag_type_2b = IRF[23:16];
        IRF_pkt_type_3b = IRF[31:24];
        IRF_l2_protocol_flag_type_8b = IRF[39:32];

        IRF_outer_vlan_high = IRF[199:192];
        IRF_outer_vlan_low = IRF[207:200];
        IRF_inner_vlan_high = IRF[215:208];
        IRF_inner_vlan_low = IRF[223:216];
    }

    B0Registers {
        IRF_l2_type = IRF[7:0];
    }

    ethernet eth;
    ieee802-1qTag vlan;
    ieee802-1OuterTag qinq;

   cellA {
        if ((eth.etherType == 0x8100) && (vlan.etherType == 0x0800))
            length = length(eth) + length(vlan);
            next_header = ipv4;
            bypass = 1;
            action = {
                  mov IRF_outer_vlan_high ++ IRF_outer_vlan_low, vlan.pcp
        ++ vlan.cfi ++ vlan.vid;
                  set IRF_tag_type_2b, 1;
                set IRF_pkt_type_3b, 0;
            }
        elseif ((eth.etherType == 0x88a8 || eth.etherType == 0x9200 || eth
        .etherType == 0x9300) && (qinq.ethertype_o == 0x8100) && (qinq.
        etherType_i == 0x0800))
            length = length(eth) + length(qinq);
            next_header = ipv4;
            bypass = 1;
            action = {
                mov IRF_outer_vlan_high ++ IRF_outer_vlan_low, qinq.pcp_o
        ++ qinq.cfi_o ++ qinq.vid_o;
                mov IRF_inner_vlan_high ++ IRF_inner_vlan_low, qinq.pcp_i
        ++ qinq.cfi_i ++ qinq.vid_i;
                set IRF_tag_type_2b, 2;
                set IRF_pkt_type_3b, 0;
              }
        elseif (eth.etherType == 0x0800)
            length = length(eth);
            next_header = ipv4;
            bypass = 1;
            action = {
                set IRF_tag_type_2b, 0;
                set IRF_pkt_type_3b, 0;
            }
        elseif (eth.etherType == 0x88CC)
            length = length(eth);
            next_header = lldp;
            bypass = 2;
            action = {
                 set IRF_l2_protocol_flag_type_8b, 66;
            }
        else
            bypass = 2;
            action = {
                set IRF_l2_send_to_cpu_8b, 1;
            }
        endif
    }

    cellB0 {
        if (eth.dmac == 0xFFFFFFFFFFFF)
            set IRF_l2_type,3;
        elseif (eth.dmac[40] == 1)
            set IRF_l2_type, 2;
        else
            set IRF_l2_type, 1;
        endif
    }
}
}
```

Figure 3: Example of a layer action specification

```
l2s {
}

l3{
    ARegisters {
        IRF_l3_send_to_cpu_8b = IRF[7:0];
        IRF_l3_encode = IRF[15:8];
        IRF_l3_type = IRF[23:16];
        IRF_l3_protocol_flag_type_8b = IRF[31:24];

        IRF_TOS_8b = IRF[199:192];
        IRF_ttl_8b = IRF[207:200];

        IRF_TTL_EXP = IRF[391:384];
    }

    B0Registers {
        IRF_DIP_LB_MUL = IRF[7:0];
        IRF_IPV4_IP_SPECIAL = IRF[15:8];
        IRF_IPV4_SIP_LB = IRF[23:16];
    }

    B1Registers {
        IRF_IP_FRAG_STATUS = IRF[7:0];
    }

    ipv4 v4;

    cellA {
        mov IRF_TOS_8b, v4.diffserv;
        mov IRF_ttl_8b, v4.ttl;
        lg IRF_TTL_EXP, 2, v4.ttl;
    }

    cellB0 {
        if (v4.srcAddr == 0x00000000)
            set IRF_IPV4_IP_SPECIAL, 1;
        elseif (v4.srcAddr[31:24] == 0x7f)
            set IRF_IPV4_SIP_LB, 1;
        endif

        if (v4.dstAddr == 0xffffffff)
            set IRF_IPV4_IP_SPECIAL, 1;
        elseif (v4.dstAddr[31:24] == 0x7f)
            set IRF_DIP_LB_MUL, 1;
        elseif (v4.dstAddr[31:8] == 0xe000000)
            set IRF_DIP_LB_MUL, 2;
        elseif (v4.dstAddr[31:28] == 0xe)
            set IRF_DIP_LB_MUL, 4;
        endif
    }

    cellB1 {
        if (v4.flagOffset == 0)
            if (v4.flags == 0)
                set IRF_IP_FRAG_STATUS, 3;
            else
                set IRF_IP_FRAG_STATUS, 1;
            endif
        endif
    }
}
}
```

Figure 4: Examples of other layer action specifications

```
protocol ethernet {
    fields = {
        dmac : 48;
        smac : 48;
        etherType : 16;
    }

    length = 14;
}

protocol ieee802-1qTag {
    fields = {
        pcp : 3;
        cfi : 1;
        vid : 12;
        etherType : 16;
    }

    length = 4;
}

protocol ieee802-1OuterTag {
    fields = {
        pcp_o : 3;
        cfi_o : 1;
        vid_o : 12;
        etherType_o : 16;
        pcp_i : 3;
        cfi_i : 1;
        vid_i : 12;
        etherType_i : 16;
    }

    length = 8;
}

protocol mpls {
    fields = {
        lable : 20;
        tc : 3;
        s : 1;
        ttl : 8;
    }

    length = 4;

    if (s == 0)
        next_header = mpls;
    endif

    action = {
    }
}
}
```

Figure 5: Some protocol specifications in P3

```
protocol ipv4 {
    fields = {
        version : 4;
        ihl : 4;
        diffserv : 8;
        totalLen : 16;
        identificaiton : 16;
        flags : 3;
        fragOffset : 13;
        ttl : 8;
        theProtocol : 8;
        hdrChecksum : 16;
        srcAddr : 32;
        dstAddr : 32;
        options : *;
    }

    if (ihl == 5)
        length = 20;
        action = {
            set IRF_l3_type[3], 0;
        }
    elseif (ihl == 6)
        length = 24;
        action = {
            set IRF_l3_type[3], 1;
        }
    elseif (ihl == 7)
        length = 28;
        action = {
            set IRF_l3_type[3], 1;
        }
    else
        action = {
            set IRF_l3_cpu_code_8b, 2;
        }
    endif

    if (theProtocol == 2)
        next_header = igmp;
        bypass = 2;
        action = {
            set IRF_l3_encode, 3;
            set IRF_l3_type[1:0], 0;
            set IRF_l3_protocol_flag_type_8b, 33;
        }
    elseif (theProtocol == 4)
        next_header = ipv4;
        bypass = 0;
        action = {
            set IRF_l3_encode, 7;
              set IRF_l3_type[1:0], 1;
        }
    elseif (theProtocol == 6)
        next_header = tcp;
        bypass = 1;
        action = {
            set IRF_l3_encode, 1;
              set IRF_l3_type[1:0], 2;
        }
    elseif (theProtocol == 0x11)
        next_header = udp;
        bypass = 1;
        action = {
              set IRF_l3_encode, 0;
              set IRF_l3_type[1:0], 2;
        }
    else
        set IRF_l3_send_to_cpu_1b, 1;
    endif
}
}
```


Figure 6: Another protocol specification in P3

To facilitate the co-design of software and hardware, a P3 assembly (ASM) intermediate representation is designed to be generated before the configuration file is produced, referring to the sub-section 3.1 for its syntax and the sub-section 6.1.1 for the associate abstract syntax. In the subsection The P3 assembly is very close to the configuration file in format. In the P3C compiler, we only verify the translations other than the translation from the P3 assembly to the configuration file. The sub-section 6.1.2 includes some examples of the P3 assembly.

3.1 Syntax of P3 assembly

⟨parser_asm⟩ ::= ⟨const_decl⟩ ⟨register_decl⟩ { ⟨layer_block⟩ }

⟨const_decl⟩ ::= **const** *IDENT* '=' *Integer* ';'   //integer constants, signed 32 bits

⟨layer_block⟩ ::= ⟨layer_id⟩ ':'
{ ⟨Pins⟩ }
⟨cella_pb⟩
⟨cella_pc_cur⟩
⟨cella_pc_nxt⟩
⟨cellb0_pb⟩
⟨cellb0_pc_cur⟩
⟨cellb1_pb⟩
⟨cellb1_pc_cur⟩

⟨layer_id⟩ ::= *IDENT*

⟨Pins⟩ ::= ‘`Pins`’ ‘`(`’ ⟨ins_name⟩ ‘`,`’ ⟨ins_size⟩ ‘`)`’

⟨cella_pb⟩ ::= ‘`Abegin`’ { ⟨cella_pb_item⟩ } ‘`Aend`’

⟨cella_pc_cur⟩ ::= ‘`ACbegin`’ { ⟨cella_pc_cur_item⟩ } ‘`ACend`’

⟨cella_pc_nxt⟩ ::= ‘`ANbegin`’ { ⟨cella_pc_nxt_item⟩ } ‘`ANend`’

⟨cellb0_pb⟩ ::= ‘`B0begin`’ { ⟨cellb0_pb_item⟩ } ‘`B0end`’

⟨cellb0_pc_cur⟩ ::= ‘`B0Cbegin`’ { ⟨cellb0_pc_cur_item⟩ } ‘`B0Cend`’

⟨cellb1_pb⟩ ::= ‘`B1begin`’ { ⟨cellb1_pb_item⟩ } ‘`B1end`’

⟨cellb1_pc_cur⟩ ::= ‘`B1Cbegin`’ { ⟨cellb1_pc_cur_item⟩ } ‘`B1Cend`’

⟨cella_pb_item⟩ ::= ⟨hdr_id⟩ ‘`,`’ ‘{’ ⟨cond⟩ { ‘`,`’ ⟨cond⟩ } ‘}’ ‘`,`’ ⟨sub_id⟩ ‘`,`’ ⟨nxt_id⟩ ‘`,`’ ⟨bypas⟩

⟨cella_pc_cur_item⟩ ::= ⟨sub_id⟩ ‘`,`’ ‘{’ ⟨cmd⟩ { ‘`,`’ ⟨cmd⟩ } ‘}’ ‘`,`’ ⟨lyr_offset⟩

⟨cella_pc_nxt_item⟩ ::= ⟨nxt_id⟩ ‘`,`’ ‘{’ ⟨cella_nxt⟩ ‘}’ ‘`,`’ ‘{’ ⟨cellb0_nxt⟩ ‘}’ ‘`,`’ ‘{’ ⟨cellb1_nxt⟩ ‘}’

⟨*cellb0_pb_item*⟩ ::= ⟨*hdr_id*⟩ ‘,’ ‘{’ ⟨*cond*⟩ { ‘,’ ⟨*cond*⟩ } ‘}’ ‘,’ ⟨*sub_id*⟩

⟨*cellb0_pc_cur_item*⟩ ::= ⟨*sub_id*⟩ ‘,’ ‘{’ ⟨*cmd*⟩ { ‘,’ ⟨*cmd*⟩ } ‘}’

⟨*cellb1_pb_item*⟩ ::= ⟨*hdr_id*⟩ ‘,’ ‘{’ ⟨*cond*⟩ { ‘,’ ⟨*cond*⟩ } ‘}’ ‘,’ ⟨*sub_id*⟩

⟨*cellb1_pc_cur_item*⟩ ::= ⟨*sub_id*⟩ ‘,’ ‘{’ ⟨*cmd*⟩ { ‘,’ ⟨*cmd*⟩ } ‘}’

⟨*hdr_id*⟩ ::= ⟨*num*⟩

⟨*sub_id*⟩ ::= ⟨*num*⟩

⟨*nxt_id*⟩ ::= ⟨*num*⟩

⟨*bypas*⟩ ::= ⟨*num*⟩

⟨*lyr_offset*⟩ ::= ⟨*num*⟩

⟨*cella_nxt*⟩ ::= ‘(’ { ⟨*irf_offset*⟩ } ‘)’ ‘+’ ‘(’ { ⟨*prot_offset*⟩ } ‘)’

⟨*cellb0_nxt*⟩ ::= ‘(’ { ⟨*irf_offset*⟩ } ‘)’ ‘+’ ‘(’ { ⟨*prot_offset*⟩ } ‘)’

⟨*cellb1_nxt*⟩ ::= ‘(’ { ⟨*irf_offset*⟩ } ‘)’ ‘+’ ‘(’ { ⟨*prot_offset*⟩ } ‘)’

⟨*irf_offset*⟩ ::= ⟨*num*⟩

⟨*prot_offset*⟩ ::= ⟨*num*⟩

⟨*cond*⟩ ::= ⟨*reg_seg*⟩ ‘==’ ⟨*num*⟩
| ⟨*ins_seg*⟩ ‘==’ ⟨*num*⟩

⟨*cmd*⟩ ::= ⟨*set_cmd*⟩
| ⟨*mov_cmd*⟩
| ⟨*lg_cmd*⟩
| ⟨*eq_cmd*⟩

⟨*set_cmd*⟩ ::= ‘(’ **set** ⟨*reg_seg*⟩ ‘,’ ⟨*num*⟩ ‘)’

⟨*mov_cmd*⟩ ::= ‘(’ **mov** ⟨*reg_seg*⟩ ‘,’ ⟨*src_reg*⟩ ‘)’

⟨*lg_cmd*⟩ ::= ‘(’ **lg** ⟨*reg_seg*⟩ ‘,’ ⟨*src_reg*⟩ ‘,’ ⟨*src_reg*⟩ ‘)’

⟨*eq_cmd*⟩ ::= ‘(’ **eq** ⟨*reg_seg*⟩ ‘,’ ⟨*src_reg*⟩ ‘,’ ⟨*src_reg*⟩ ‘)’

⟨*src_reg*⟩ ::= ‘(’ **IRF** ‘,’ ⟨*reg_offset*⟩ ‘,’ ⟨*reg_size*⟩ ‘)’
| ⟨*num*⟩

⟨*reg_seg*⟩ ::= ‘(’ **IRF** ‘,’ ⟨*reg_offset*⟩ ‘,’ ⟨*seg_size*⟩ ‘)’

⟨*ins_seg*⟩ ::= ‘(’ ⟨*ins_name*⟩ ‘,’ ⟨*ins_offset*⟩ ‘,’ ⟨*seg_size*⟩ ‘)’

$\langle \mathit{reg\_offset} \rangle$ ::= $\langle \mathit{num} \rangle$

$\langle \mathit{reg\_size} \rangle$ ::= $\langle \mathit{num} \rangle$

$\langle \mathit{seg\_size} \rangle$ ::= $\langle \mathit{num} \rangle$

$\langle \mathit{ins\_size} \rangle$ ::= $\langle \mathit{num} \rangle$

$\langle \mathit{num} \rangle$ ::= *Integer* //integer constants, signed 32 bits
| *Hexadecimal* //hex constants, such as 0x88a8, 0xFFFFFF, 0x89,0x103

3.2 The configuration file format

$\langle \mathit{configuration} \rangle$ ::= { $\langle \mathit{layer\_config} \rangle$ }

$\langle \mathit{layer\_con} \rangle$ ::= $\langle \mathit{layer\_id} \rangle$ ':' $\langle \mathit{pb\_lut} \rangle$ $\langle \mathit{pc\_cur\_lut} \rangle$ $\langle \mathit{pc\_nxt\_lut} \rangle$

$\langle \mathit{layer\_con} \rangle$ ::= $\langle \mathit{layer\_id} \rangle$ ':'
$\langle \mathit{cella\_pb\_con} \rangle$
$\langle \mathit{cella\_pc\_cur\_con} \rangle$
$\langle \mathit{cella\_pc\_nxt\_con} \rangle$
$\langle \mathit{cellb0\_pb\_con} \rangle$
$\langle \mathit{cellb0\_pc\_cur\_con} \rangle$
$\langle \mathit{cellb1\_pb\_con} \rangle$
$\langle \mathit{cellb1\_pc\_cur\_con} \rangle$

$\langle \mathit{layer\_id} \rangle$ ::= *IDENT*

$\langle \mathit{cella\_pb\_con} \rangle$ ::= `CellA PB` { $\langle \mathit{cella\_pb\_con\_item} \rangle$ }

$\langle \mathit{cella\_pc\_cur\_con} \rangle$ ::= `CellA PC CUR` { $\langle \mathit{cella\_pc\_cur\_con\_item} \rangle$ }

$\langle \mathit{cella\_pc\_nxt\_con} \rangle$ ::= `CellA PC NXT` { $\langle \mathit{cella\_pc\_nxt\_con\_item} \rangle$ }

$\langle \mathit{cellb0\_pb\_con} \rangle$ ::= `CellB0 PB` { $\langle \mathit{cellb0\_pb\_con\_item} \rangle$ }

$\langle \mathit{cellb0\_pc\_cur\_con} \rangle$ ::= `CellB0 PC CUR` { $\langle \mathit{cellb0\_pc\_cur\_con\_item} \rangle$ }

$\langle \mathit{cellb1\_pb\_con} \rangle$ ::= `CellB1 PB` { $\langle \mathit{cellb1\_pb\_con\_item} \rangle$ }

$\langle \mathit{cellb1\_pc\_cur\_con} \rangle$ ::= `CellB1 PC CUR` { $\langle \mathit{cellb1\_pc\_cur\_con\_item} \rangle$ }

$\langle \mathit{cella\_pb\_con\_item} \rangle$ ::= $\cdots$

$\langle \mathit{cella\_pc\_cur\_con\_item} \rangle$ ::= $\cdots$

$\langle \mathit{cella\_pc\_nxt\_con\_item} \rangle$ ::= $\cdots$

$\langle \mathit{cellb0\_pb\_con\_item} \rangle$ ::= $\cdots$

$\langle \mathit{cellb0\_pc\_cur\_con\_item} \rangle$ ::= $\cdots$

$\langle cellb1\_pb\_con\_item \rangle$ ::= $\cdots$

$\langle cellb1\_pc\_cur\_con\_item \rangle$ ::= $\cdots$

3.3 Examples of the P3 assembly

In this subsection, we show shows some examples of the P3 assembly corresponding to the example of P3 specification in Section 2.2.

Fig.7 shows the P3 ASM corresponding to the layer action L2 in Fig.3, where three protocol instances *eth*, *vlan* and *qinq* are declared.

The P3 ASM for a layer action should have 7 tables, three tables for the Cell A, and two tables for the Cell B0 and Cell B1 respectively. In Fig.7, we omit the tables for Cell B1 since no table-driven actions specified in this cell in Fig.3.

The three tables of Cell A are *cella_pb*, *cella_pc_cur* and *cella_pc_nxt*. In the table *cella_pb*, the attributes for the next protocol identifier (*nxt_id*) and the bypass value are set. Besides, a table looking-up operation is specified by a combination of *guard-condition* and *jump-to*(*sub_id*), such as the *guard-condition* (eth, 96, 16) == 0x88a8, (qinq, 16, 16) == 0x8100, (qinq, 48, 16) == 0x0800, the conjunction of the atomic conditions (propositions), and *jump-to* the 0x2 labeled actions' set in *cella_pc_cur* table.

In the *cella_pc_cur* table, 0x2 labeled actions' set includes the following commands:

- mov IRF_outer_vlan_high, IRF_outer_vlan_low, qinq.pcp_o,qinq.cfi_o,qinq.vid_o;
- mov IRF_inner_vlan_high, IRF_inner_vlan_low, qinq.pcp_i,qinq.cfi_i,qinq.vid_i;
- set IRF_tag_type_2b, 2;set IRF_pkt_type_3b, 0;

Besides, in the *cella_pc_cur* table, the attribute for the next-layer offset (*lyr_offset*) is set.

In the table *cella_pc_nxt*, some fields of a protocol are masked for the usage in the later layers, partitioned by the different cells. For example in the first line of the *cella_pc_nxt* table in Fig.7, the masked fields of the protocol *ipv4* are (0 9) for the cell A, (0xc 0xd 0xe 0xf 0x10 0x11 0x12 0x13) for the cell B0, and (6 7) for the cell B1. From the Fig.4, we know that an *ipv4* instance *v4* is declared in the layer action *l3s*. The fields *ihl* and *theProtocol* are used in the cell A, the fields *srcAddr* and *dstAddr* are used in the cell B0, and the fields *flagOffset* and *flags* are used in the cell B1. Referring to the offsets of these fields in the *ipv4* protocol in Fig.6, the fields *ihl* and *theProtocol* are masked by (0 9) correponding to the 1st byte and the 10th byte in *ipv4*. Similarly, the fields *srcAddr* and *dstAddr* are masked by (0xc 0xd 0xe 0xf 0x10 0x11 0x12 0x13), and the fields *flagOffset* and *flags* are masked by (6 7).

In Fig.7, the tables *cellb0_pb* and *cellb0_pc_cur* are the same in structure with *cella_pb* and *cella_pc_cur* except for without the information to set the next protocol identifier, the bypass value and the next-layer offset.

Fig.8 shows the P3 ASM extracts corresponding to the layer action L3 in Fig.4, where the protocol instances *p4* is declared.

```
//L2header
//Ethernet II:DMAC(6B)+SMAC(6B)+Type(2B)
//VLAN(18B)  :DMAC(6B)+SMAC(6B)+Type(2B)+TAG(2B:PRI/3b+CFI/1b+VID/12b)+
        Length/Type(2B)
//QinQ(22B)  :DMAC(6B)+SMAC(6B)+EType(2B)+TAG(2B:Pri/3b+CFI/1b+VID/12b)+
        EType(2B)+TAG(2B:Pri/3b+CFI/1b+VID/12b)+LEN/ETYPE(2B)

l2:

Pins (eth, 112)   // size: 8*14
Pins (vlan, 32)   // size: 8*4
Pins (qinq, 64)   // size: 8*8

Abegin

//cella_pb(407bit*32)
//hdr_id(7)+mask(24*8b)+value(24*8b)+sub_id(7)+nxt_id(7)+bypass(2:
        mainbypass(1)+subbypass(1))

0x1,{(eth, 96, 16) == 0x8100, (vlan, 16, 16) == 0x0800}, 0x1, 0x3, 1//eth+
        vlan+ipv4
0x1, {(eth, 96, 16) == 0x88a8, (qinq, 16, 16) == 0x8100, (qinq, 48, 16) ==
          0x0800},0x2,0x3,1//eth+qinq+ipv4
0x1, {(eth, 96, 16) == 0x9200, (qinq, 16, 16) == 0x8100, (qinq, 48, 16) ==
          0x0800},0x2,0x3,1//eth+qinq+ipv4
0x1, {(eth, 96, 16) == 0x9300, (qinq, 16, 16) == 0x8100, (qinq, 48, 16) ==
          0x0800},0x2,0x3,1//eth+qinq+ipv4
0x1, {(eth, 96, 16) == 0x0800}, 0x3, 0x3, 1//eth+ipv4
0x1, {(eth, 96, 16) == 0x88cc}, 0x4, 0x6, 2//eth+lldp

Aend

ACbegin

//cella_pc_cur(328bit*32)
//vliw(320:alu(8*24b)+mov(8*8b)+set(8*8b))+lyr_offset(8)
0x1,{(mov (IRF,192,16), (vlan,0,16)),(set (IRF,16,8), 1), (set (IRF,24,8),
         0)}, 0x12
0x2,{(mov (IRF,192,16), (vlan,0,16)),(mov (IRF,208,16), (vlan,0,16)),(set
        (IRF,16,8), 2), (set (IRF,24,8), 0)}, 0x16
0x3,{(set (IRF,16,8), 0), (set (IRF,24,8), 0)}, 0xc
0x4,{(set (IRF,32,8), 66)}, 0xc

ACend

ANbegin

//cella_pc_nxt(583bit*32)
//nxt_id(7)+pa_offset(3*24*8b:cellA(irf/2+fra/22)+cellB0(irf/2+fra/22)+
        cellB1(irf/2+fra/22))

0x3, {( ) + (0 9)}, {( ) +(0xc 0xd 0xe 0xf 0x10 0x11 0x12 0x13}, {( ) + (6
         7)}//ipv4
0x6, {( ) + ( )}, {( ) + ( )}, {( ) + ( )}//lldp

ANend

B0begin

//cellb0_pb(398bit*32)
//hdr_id(7)+mask(24*8b)+value(24*8b)+sub_id(7)

0x1, {(eth, 0, 48) == 0xFFFFFFFFFFFF}, 0x1//eth.dmac == 0xFFFFFFFFFFFF
0x1, {(eth, 40, 1) == 1}, 0x2//eth.dmac[40] == 1

B0end

B0Cbegin

//cellb0_pc_cur(320bit*32)
//vliw(320:alu(8*24b)+mov(8*8b)+set(8*8b))
0x2, {(set (IRF,0,8), 3)}//sub_id:01,set IRF_l2_type = 3;
0x2, {(set (IRF,0,8), 2)}//sub_id:02,set IRF_l2_type = 2;

B0Cend
```


Figure 7: P3 ASM for the layer action L2 in Fig.3

```
//L3header(IPheader)
//IPv4(20B)  :Ver(4bit)+IHL(4bit)+TyoS(8bit)+TtlLen(16bit)+Iden(16bit)+Flg
        (3bit)+FraOffset(13bit)+TimeTOLive(8bit)+Protocol(8bit)+HdrCheSUM
        (16bit)+SAddr(32bit)+DAddr(32bit)
l3:

Pins (v4, 224)   // size: 8*28

Abegin
//cella_pb(407bit*32)
//hdr_id(7)+mask(24*8b)+value(24*8b)+sub_id(7)+nxt_id(7)+bypass(2:
        mainbypass(1)+subbypass(1))

0x3, {(v4, 4, 4) == 5, (v4, 72, 8) == 2}, 0x1, 0x9, 2
......
0x3, {(v4, 4, 4) == 7, (v4, 72, 8) == 0x11}, 0xc, 0xd, 1
Aend

ACbegin
//cella_pc_cur(328bit*32)
//vliw(320:alu(8*24b)+mov(8*8b)+set(8*8b))+lyr_offset(8)
0x1,{(mov (IRF,192,8), (v4,8,8)),(mov (IRF,200,8), (v4,64,8)), (lg (IRF
        ,384,8), 2, (v4,64,8)), (set (IRF,19,1), 0), (set (IRF,8,8), 3),(
        set (IRF,16,2),0), (set (IRF,24,8), 33))}, 0x14
......
0xc, {(mov (IRF,192,8), (v4,8,8)),(mov (IRF,200,8), (v4,64,8)), (lg (IRF
        ,384,8), 2, (v4,64,8)),  (set (IRF,19,1), 1), (set (IRF,8,8), 0),
        (set (IRF,16,2), 2)}, 0x1c
ACend

ANbegin
//cella_pc_nxt(583bit*32)
//nxt_id(7)+pa_offset(3*24*8b)
0x9, {( ) + ( )}, {( ) + ( )}, {( ) + ( )}//igmp
0x3, {( ) + ( )}, {( ) + ( )}, {( ) + ( )}//ipv4
0xc, {( ) + ( )}, {( ) + ( )}, {( ) + ( )}//tcp
0xd, {( ) + ( )}, {( ) + ( )}, {( ) + ( )}//udp
ANend

B0begin
//cellb0_pb(398bit*32)
//hdr_id(7)+mask(24*8b)+value(24*8b)+sub_id(7)
0x3, {(v4, 96, 32) == 0x00000000, (v4,128,32) == 0xffffffff}, 0x1
......
0x3, {v4.srcAddr?,(v4,156,4) == 0xe}, 0xe
B0end

B0Cbegin
//cellb0_pc_cur(320bit*32)
//vliw(320:alu(8*24b)+mov(8*8b)+set(8*8b))
0x1, {(set (IRF,8,8),1),?}//sub_id:01,set IRF_IPV4_IP_SPECIAL, 1;set
        IRF_IPV4_IP_SPECIAL, 1;
......
0xe, {(set (IRF,0,8),4)}//sub_id:0e,set IRF_DIP_LB_MUL, 4;
B0Cend

B1begin

//cellb1_pb(398bit*32)
//hdr_id(7)+mask(24*8b)+value(24*8b)+sub_id(7)
0x3, {(v4, 51, 13) == 0, (v4,48,3) == 0}, 0x1//v4.flagOffset == 0,v4.flags
         == 0
0x3, {(v4, 51, 13) == 0, (v4,48,3) =\= 0 ?}, 0x2//v4.flagOffset == 0,v4.
        flags != 0
B1end

B1Cbegin
//cellb1_pc_cur(320bit*32)
//vliw(320:alu(8*24b)+mov(8*8b)+set(8*8b))
0x1, {(set (IRF,0,7),3)} //sub_id:01,set IRF_IP_FRAG_STATUS, 3;
0x2, {(set (IRF,0,7),1)} //sub_id:02,set IRF_IP_FRAG_STATUS, 1;
B1Cend//
```

Figure 8: P3 ASM for another layer action L3 in Fig.4 (extracts)

It is worth to noting that the table looking-up operations of *cella_pb* in Fig.8 is corresponding to the guarded actions specified for the protocol *ipv4* in Fig.6.

4 Parsing

By scanning and parsing, a P3 source specification such as the example in Section 2.2 is translated into its corresponding P3 AST, referring to Fig.1. The syntax of a P3 AST is presented in the subsection 4.1.

4.1 The P3 Abstract Syntax Tree

⟨*parser_spec*⟩ ::= *Parser* (⟨*layer_reg_len*⟩, ⟨*cell_reg_len*⟩, ⟨*protocol_set*⟩, ⟨*layer_set*⟩,
{ ⟨*decl*⟩ })

⟨*layer_reg_len*⟩ ::= *Lreglen* (*IntConst*(*Integer*))

⟨*cell_reg_len*⟩ ::= *Creglen* (*IntConst*(*Integer*))

⟨*protocol_set*⟩ ::= *Pset* (⟨*id_list*⟩)

⟨*layer_set*⟩ ::= *Lset* (⟨*id_list*⟩)

⟨*id_list*⟩ ::= { *IDENT* }

⟨*decl*⟩ ::= *ConstDecl* (⟨*const_decl*⟩)
| *RegAccSet* (⟨*reg_acc_set*⟩)
| ⟨*protocol_decl*⟩
| ⟨*layer_action*⟩

⟨*const_decl*⟩ ::= *ConstDcl*(*IDENT*, ⟨*const*⟩)

⟨*const*⟩ ::= *IDENT* // constant identifiers
| *IntConst*(*Integer*) //integer constants, signed 32 bits
| *HexConst*(*Hexadecimal*) //hex constants, such as 0x88a8, 0xFFFFFF
| *BitSConst*(*BITS*) //binary constants, such as 001001, 100, 0, 1

⟨*protocol_decl*⟩ ::= *ProtocolDecl* (*IDENT* , ⟨*protocol*⟩)

⟨*protocol*⟩ ::= *Protocol* (⟨*fields*⟩ , ⟨*p_stmts*⟩)

⟨*fields*⟩ ::= (*Fields* (⟨*field*⟩ { ⟨*field*⟩ }, *OptionFields* ([⟨*option_field*⟩]))

⟨*field*⟩ ::= (*IDENT* , ⟨*const*⟩)

⟨*option_field*⟩ ::= (*IDENT* , 0)

⟨*p_stmts*⟩ ::= { ⟨*p_stmt*⟩ }

$\langle p\_stmt \rangle$ ::= $\langle if\_else\_p\_stmt \rangle$
| *NextHeader* (*IDENT*)
| *Length* ($\langle const \rangle$)
| *Bypass* ($\langle const \rangle$)
| $\langle action\_stmt \rangle$

$\langle if\_else\_p\_stmt \rangle$ ::= *IfElseP*({ $\langle if\_branch\_p \rangle$ } , $\langle default\_branch\_p \rangle$)

$\langle if\_branch\_p \rangle$ ::= ($\langle expr \rangle$, $\langle p\_stmts \rangle$)

$\langle default\_branch\_p \rangle$ ::= [$\langle p\_stmts \rangle$]

$\langle layer\_action \rangle$ ::= *LayerAction* (*IDENT*, $\langle local\_reg\_decl \rangle$, $\langle l\_decls \rangle$, $\langle l\_actions \rangle$)

$\langle l\_decls \rangle$ ::= $\langle local\_reg\_decl \rangle$ { $\langle l\_decl \rangle$ }

$\langle l\_decl \rangle$ ::= *ProtocolDef* (*IDENT* , $\langle id\_list \rangle$)

$\langle local\_reg\_decl \rangle$ ::= *LocalRegs* ($\langle cella\_regs \rangle$, $\langle cellb0\_regs \rangle$, $\langle cellb1\_regs \rangle$)

$\langle cella\_regs \rangle$ ::= *CellARegs* ({ $\langle reg\_acc\_set \rangle$ })

$\langle cellb0\_regs \rangle$ ::= *CellB0Regs* ({ $\langle reg\_acc\_set \rangle$ })

$\langle cellb1\_regs \rangle$ ::= *CellB1Regs* ({ $\langle reg\_acc\_set \rangle$ })

$\langle l\_actions \rangle$ ::= *LocalActions* ($\langle cella\_actions \rangle$, $\langle cellb0\_actions \rangle$, $\langle cellb1\_actions \rangle$)

$\langle cella\_actions \rangle$ ::= *CellA* ({ $\langle l\_stmt \rangle$ })

$\langle cellb0\_actions \rangle$ ::= *CellB0* ({ $\langle l\_stmt \rangle$ })

$\langle cellb1\_actions \rangle$ ::= *CellB1* ({ $\langle l\_stmt \rangle$ })

$\langle l\_stmt \rangle$ ::= $\langle if\_else\_l\_stmt \rangle$
| *NextHeader* (*IDENT*)
| *Length* ($\langle expr \rangle$)
| *Bypass* ($\langle const \rangle$)
| $\langle action\_stmt \rangle$

$\langle l\_stmts \rangle$ ::= { $\langle l\_stmt \rangle$ }

$\langle if\_else\_l\_stmt \rangle$ ::= *IfElseL* ({ $\langle if\_branch\_l \rangle$ } , $\langle default\_branch\_l \rangle$)

$\langle if\_branch\_l \rangle$ ::= ($\langle expr \rangle$, $\langle l\_stmts \rangle$)

$\langle default\_branch\_l \rangle$ ::= [$\langle l\_stmts \rangle$]

```
⟨expr⟩ ::= Eatom(⟨atom⟩)
     | Eunop(⟨unop⟩, ⟨expr⟩)   (* unary operation *)
     | Ebinop(⟨binop⟩, ⟨expr⟩, ⟨expr⟩)   (* binary operation *)
     | Efield(⟨expr⟩, IDENT)   (* access to a field in a protocol *)
     | EFieldBit(⟨expr⟩, ⟨expr⟩)   (*access to a bit of a field or a register access*)
     | EFieldSection(⟨expr⟩, ⟨expr⟩, ⟨expr⟩)
                 (* access to a section of a field or a register access *)
     | ProtLen(IDENT)

⟨atom⟩ ::= Econst(⟨const⟩)   //const expressions
     | IDENT   //all kinds of access name , ex., field or register access name

⟨unop⟩ ::= Oint   //convert hexadecimal or binary numbers to integers
     | Onot   //logical negation
     | Oneg   //bit-wise negation

⟨binop⟩ ::= Oadd   // addition '+'
     | Osub   // subtraction '-'
     | Omul   // multiplication '*'
     | Odivint   // division integer '/'
     | Omod   // remainder '%'
     | Oand   //logical and '&&'
     | Oor   //logical or '||'
     | Oband   //bit-wise and '&'
     | Obor   //bit-wise or '|'
     | Obeor   //bit-wise exclusive or '^'
     | Oeq   // comparison ([=])
     | One   // comparison ([<>])
     | Olt   // comparison ([<])
     | Ogt   // comparison ([>])
     | Ole   // comparison ([<=])
     | Oge   // comparison ([>=])
     | Osl   //shift left '<<'
     | Osr   //shift right '>>'
     | Obc   //bits' concatenation '++'
     | Ohexes //convert a binary number or an integer to a hexadecimal number
     | Obits   //convert an integer or a hexadecimal number to a binary number

⟨action_stmt⟩ ::= Action( ⟨instructions⟩ )

⟨instructions⟩ ::= { ⟨instruction⟩ }

⟨instruction⟩ ::= Set (⟨tgt_reg_acc_name⟩, ⟨expr⟩)
     | Mov (⟨mov_reg_acc_name⟩, ⟨expr⟩)
     | Lg (⟨tgt_reg_acc_name⟩, ⟨expr⟩, ⟨expr⟩)
     | Eq (⟨tgt_reg_acc_name⟩, ⟨expr⟩, ⟨expr⟩)

⟨reg_acc_set⟩ ::= IRF( IDENT, ⟨expr⟩ , ⟨expr⟩ )
     | IRF( IDENT, ⟨expr⟩ )

⟨tgt_reg_acc_name⟩ ::= TargetRegAccName( IDENT )
     | TargetRegAccName ( ⟨tgt_reg_acc_name⟩, ⟨expr⟩ , ⟨expr⟩ )
     | TargetRegAccName ( ⟨tgt_reg_acc_name⟩, ⟨expr⟩ )
```

$$
\begin{aligned}
\langle \mathit{mov\_reg\_acc\_name} \rangle ::= \ & \mathit{MovRegAccName}(\ \langle \mathit{tgt\_reg\_acc\_name} \rangle\ ) \\
\mid\ & \mathit{MovRegAccName}(\ \langle \mathit{mov\_reg\_acc\_name} \rangle, \langle \mathit{tgt\_reg\_acc\_name} \rangle\ )
\end{aligned}
$$

4.2 Implementation and Verification

In the current version of the compiler, the OCaml code of parsing is generated in Coq by Menhir *Menhir* [8], which has been formally validated and produces proofs of correctness and completeness by the J.-H. Jourdan approach [5] used in CompCert [7, 6, 4].

As examples, we list as follows the printed P3 AST corresponding to Fig.2 and Fig.3, part of the example in Section 2.2.

```
<parser>
    <layer_reg_len>
        <const>
            <int>(72)
    <cell_reg_len>
        <const>
            <int>(24)
    <protocol_set>
        <id>(ethernet)
        <id>(ieee802−1qTag)
        <id>(ipv4)
        <id>(mpls)
        <id>(ieee802−1OuterTag)
        <id>(lldp)
        <id>(trill)
        <id>(qcn)
        <id>(igmp)
        <id>(ospf)
        <id>(pim)
        <id>(tcp)
        <id>(udp)
    <layer_set>
        <id>(l2)
        <id>(l2s)
        <id>(l3)
        <id>(l3s)
        <id>(l4)
    <decl>
        <const_decl>
            <id>(global_IRF_len)
            <const_expr>
                <const>
                    <int>(64)
    <decl>
        <reg_acc_set>
            <reg_acc_name>
                <id>(IRF_gp_reg0_2b)
            <binop_expr>
                <binop>(+)
                <const_expr>
                    <const>
                        <id>(global_IRF_len)
                <const_expr>
```

```
                <const>
                    <int>(1)
        <const_expr>
            <const>
                <id>(global_IRF_len)
<decl>
    <reg_acc_set>
        <reg_acc_name>
            <id>(IRF_gp_reg1_2b)
        <binop_expr>
            <binop>(+)
            <const_expr>
                <const>
                    <id>(global_IRF_len)
            <const_expr>
                <const>
                    <int>(3)
        <binop_expr>
            <binop>(+)
            <const_expr>
                <const>
                    <id>(global_IRF_len)
            <const_expr>
                <const>
                    <int>(2)
<decl>
    <layer_action>
        <id>(l2)
        <local_reg_decl>
            <cella_regs>
                <reg_acc_set>
                    <reg_acc_name>
                        <id>(IRF_l2_send_to_cpu_8b)
                    <const_expr>
                        <const>
                            <int>(15)
                    <const_expr>
                        <const>
                            <int>(8)
                <reg_acc_set>
                    <reg_acc_name>
                        <id>(IRF_tag_type_2b)
                    <const_expr>
                        <const>
                            <int>(23)
                    <const_expr>
                        <const>
                            <int>(16)
                <reg_acc_set>
                    <reg_acc_name>
                        <id>(IRF_pkt_type_3b)
                    <const_expr>
                        <const>
                            <int>(31)
                    <const_expr>
                        <const>
                            <int>(24)
```

```
    <reg_acc_set>
        <reg_acc_name>
            <id>(IRF_l2_protocol_flag_type_8b)
        <const_expr>
            <const>
                <int>(39)
        <const_expr>
            <const>
                <int>(32)
    <reg_acc_set>
        <reg_acc_name>
            <id>(IRF_outer_vlan_high)
        <const_expr>
            <const>
                <int>(199)
        <const_expr>
            <const>
                <int>(192)
    <reg_acc_set>
        <reg_acc_name>
            <id>(IRF_outer_vlan_low)
        <const_expr>
            <const>
                <int>(207)
        <const_expr>
            <const>
                <int>(200)
    <reg_acc_set>
        <reg_acc_name>
            <id>(IRF_inner_vlan_high)
        <const_expr>
            <const>
                <int>(215)
        <const_expr>
            <const>
                <int>(208)
    <reg_acc_set>
        <reg_acc_name>
            <id>(IRF_inner_vlan_low)
        <const_expr>
            <const>
                <int>(223)
        <const_expr>
            <const>
                <int>(216)
<cellb0_regs>
    <reg_acc_set>
        <reg_acc_name>
            <id>(IRF_l2_type)
        <const_expr>
            <const>
                <int>(7)
        <const_expr>
            <const>
                <int>(0)
<cellb1_regs>(None)
<l_decl>
```

```
        <id>(ethernet)
        <id>(eth)
    <l_decl>
        <id>(ieee802−1qTag)
        <id>(vlan)
    <l_decl>
        <id>(ieee802−1OuterTag)
        <id>(qinq)
    <l_actions>
        <cella_actions>
            <l_stmt>
                <if_else_l_stmt>
                    <if_branch_l>
                        <binop_expr>
                            binop(&&)
                            <paren_expr>
                                <binop_expr>
                                    binop(==)
                                    <field_expr>
                                        <const_expr>
                                            <const>
                                                <id>(eth)
                                        <id>(etherType)
                                    <const_expr>
                                        <const>
                                            <hex>(0x8100)
                            <paren_expr>
                                <binop_expr>
                                    binop(==)
                                    <field_expr>
                                        <const_expr>
                                            <const>
                                                <id>(vlan)
                                        <id>(etherType)
                                    <const_expr>
                                        <const>
                                            <hex>(0x0800)
                        <l_stmt>
                            <binop_expr>
                                <binop>(+)
                                <length _expr>
                                    <id>(eth)
                                <length _expr>
                                    <id>(vlan)
                        <l_stmt>
                            <id>(ipv4)
                        <l_stmt>
                            <const>
                                <int>(1)
                        <l_stmt>
                            <action_stmt>
                                <mov_instruction>
                                    <mov_reg_acc_name>
                                        <mov_reg_acc_name>
                                            <tgt_reg_acc_name>
                                                <id>(
IRF_outer_vlan_high)
```

```
                    <tgt_reg_acc_name>
                        <id>(IRF_outer_vlan_low)
                <binop_expr>
                    binop(++)
                    <binop_expr>
                        binop(++)
                        <field_expr>
                            <const_expr>
                                <const>
                                    <id>(vlan)
                            <id>(pcp)
                        <field_expr>
                            <const_expr>
                                <const>
                                    <id>(vlan)
                            <id>(cfi)
                    <field_expr>
                        <const_expr>
                            <const>
                                <id>(vlan)
                        <id>(vid)
            <seq_instruction>
                <tgt_reg_acc_name>
                    <id>(IRF_tag_type_2b)
                <const_expr>
                    <const>
                        <int>(1)
            <seq_instruction>
                <tgt_reg_acc_name>
                    <id>(IRF_pkt_type_3b)
                <const_expr>
                    <const>
                        <int>(0)
<if_branch_l>
    <binop_expr>
        binop(&&)
        <binop_expr>
            binop(&&)
            <paren_expr>
                <binop_expr>
                    binop (||)
                    <binop_expr>
                        binop (||)
                        <binop_expr>
                            binop(==)
                            <field_expr>
                                <const_expr>
                                    <const>
                                        <id>(eth)
                                <id>(etherType)
                            <const_expr>
                                <const>
                                    <hex>(0x88a8)
                        <binop_expr>
                            binop(==)
                            <field_expr>
                                <const_expr>
```

```
                                <const>
                                    <id>(eth)
                            <id>(etherType)
                        <const_expr>
                            <const>
                                <hex>(0x9200)
                <binop_expr>
                    binop(==)
                    <field_expr>
                        <const_expr>
                            <const>
                                <id>(eth)
                        <id>(etherType)
                    <const_expr>
                        <const>
                            <hex>(0x9300)
        <paren_expr>
            <binop_expr>
                binop(==)
                <field_expr>
                    <const_expr>
                        <const>
                            <id>(qinq)
                    <id>(ethertype_o)
                <const_expr>
                    <const>
                        <hex>(0x8100)
    <paren_expr>
        <binop_expr>
            binop(==)
            <field_expr>
                <const_expr>
                    <const>
                        <id>(qinq)
                <id>(etherType_i)
            <const_expr>
                <const>
                    <hex>(0x0800)
<l_stmt>
    <binop_expr>
        <binop>(+)
        <length _expr>
            <id>(eth)
        <length _expr>
            <id>(qinq)
<l_stmt>
    <id>(ipv4)
<l_stmt>
    <const>
        <int>(1)
<l_stmt>
    <action_stmt>
        <mov_instruction>
            <mov_reg_acc_name>
                <mov_reg_acc_name>
                    <tgt_reg_acc_name>
                        <id>(
```

```
IRF_outer_vlan_high)
            <tgt_reg_acc_name>
                <id>(IRF_outer_vlan_low)
        <binop_expr>
            binop(++)
            <binop_expr>
                binop(++)
                <field_expr>
                    <const_expr>
                        <const>
                            <id>(qinq)
                    <id>(pcp_o)
                <field_expr>
                    <const_expr>
                        <const>
                            <id>(qinq)
                    <id>(cfi_o)
            <field_expr>
                <const_expr>
                    <const>
                        <id>(qinq)
                <id>(vid_o)
    <mov_instruction>
        <mov_reg_acc_name>
            <mov_reg_acc_name>
                <tgt_reg_acc_name>
                    <id>(
IRF_inner_vlan_high)
            <tgt_reg_acc_name>
                <id>(IRF_inner_vlan_low)
        <binop_expr>
            binop(++)
            <binop_expr>
                binop(++)
                <field_expr>
                    <const_expr>
                        <const>
                            <id>(qinq)
                    <id>(pcp_i)
                <field_expr>
                    <const_expr>
                        <const>
                            <id>(qinq)
                    <id>(cfi_i)
            <field_expr>
                <const_expr>
                    <const>
                        <id>(qinq)
                <id>(vid_i)
    <seq_instruction>
        <tgt_reg_acc_name>
            <id>(IRF_tag_type_2b)
        <const_expr>
            <const>
                <int>(2)
    <seq_instruction>
        <tgt_reg_acc_name>
```

```
                    <id>(IRF_pkt_type_3b)
                <const_expr>
                    <const>
                        <int>(0)
<if_branch_l>
    <binop_expr>
        binop(==)
        <field_expr>
            <const_expr>
                <const>
                    <id>(eth)
            <id>(etherType)
        <const_expr>
            <const>
                <hex>(0x0800)
    <l_stmt>
        <length _expr>
            <id>(eth)
    <l_stmt>
        <id>(ipv4)
    <l_stmt>
        <const>
            <int>(1)
    <l_stmt>
        <action_stmt>
            <seq_instruction>
                <tgt_reg_acc_name>
                    <id>(IRF_tag_type_2b)
                <const_expr>
                    <const>
                        <int>(0)
            <seq_instruction>
                <tgt_reg_acc_name>
                    <id>(IRF_pkt_type_3b)
                <const_expr>
                    <const>
                        <int>(0)
<if_branch_l>
    <binop_expr>
        binop(==)
        <field_expr>
            <const_expr>
                <const>
                    <id>(eth)
            <id>(etherType)
        <const_expr>
            <const>
                <hex>(0x88CC)
    <l_stmt>
        <length _expr>
            <id>(eth)
    <l_stmt>
        <id>(lldp)
    <l_stmt>
        <const>
            <int>(2)
    <l_stmt>
```

```
          <action_stmt>
            <seq_instruction>
              <tgt_reg_acc_name>
                <id>(
IRF_l2_protocol_flag_type_8b)
              <const_expr>
                <const>
                  <int>(66)
      <default_branch_l>
        <l_stmt>
          <const>
            <int>(2)
        <l_stmt>
          <action_stmt>
            <seq_instruction>
              <tgt_reg_acc_name>
                <id>(IRF_l2_send_to_cpu_8b)
              <const_expr>
                <const>
                  <int>(1)
<cellb0_actions>
  <l_stmt>
    <if_else_l_stmt>
      <if_branch_l>
        <binop_expr>
          binop(==)
          <field_expr>
            <const_expr>
              <const>
                <id>(eth)
            <id>(dmac)
          <const_expr>
            <const>
              <hex>(0xFFFFFFFFFFFF)
        <l_stmt>
          <action_stmt>
            <seq_instruction>
              <tgt_reg_acc_name>
                <id>(IRF_l2_type)
              <const_expr>
                <const>
                  <int>(3)
      <if_branch_l>
        <binop_expr>
          binop(==)
          <bit_expr>
            <field_expr>
              <const_expr>
                <const>
                  <id>(eth)
              <id>(dmac)
            <const_expr>
              <const>
                <int>(40)
          <const_expr>
            <const>
              <int>(1)
```

```
                <l_stmt>
                    <action_stmt>
                        <seq_instruction>
                            <tgt_reg_acc_name>
                                <id>(IRF_l2_type)
                            <const_expr>
                                <const>
                                    <int>(2)
            <default_branch_l>
                <l_stmt>
                    <action_stmt>
                        <seq_instruction>
                            <tgt_reg_acc_name>
                                <id>(IRF_l2_type)
                            <const_expr>
                                <const>
                                    <int>(1)
    <cellb1_actions>(None)
```

5 Type Checking

By type checking on the P3 AST, we can guarantee a P3 source specification to be *well-typed*. The type checking is based on the type rules in the type system for P3 AST defined in the subsection 5.1. If the P3 AST can not pass the type checking, the P3 source specification will be refused by the compiler.

5.1 Type system for P3

5.1.1 Type expressions

A basic type expression can be defined by the syntax shown as follows.

<type>	::=	*Int*	integer type, signed integer up to 32 bits
	|	*Bool*	boolean type
	|	*Hexes*(n)	hexadecimal type, with n hexadecimal digits
	|	*Bits*(n)	binary type, with n binary digits
	|	*RegAcc*(k, i, j)	register segment access type, $0 \leq j \leq i < k$, and k is the size of the register *IRF* in the current context
	|	*FieldAcc*(id, k, i, j)	protocol field access type in a cell context, k is the protocol instance length, with $0 \leq i \leq j < k \vee (i = k \wedge j \textit{ is undefined})$
	|	*FieldAcc*(k, i, j)	protocol field access type in a protocol context, k is the protocol instance length, with $0 \leq i \leq j < k \vee (i = k \wedge j \textit{ is undefined})$
	|	X	type to specify that any instance of the protocol named X has a type X

For a constant expression, we need to compute its value for the validity checking in many places. Hence, we add an associate value to form an additional basic type, shown as follows.

<type>	::=	(τ, i)	a integer constant type, with the type τ and the integer value i, a signed integer up to 32 bits

5.1.2 Typing environment

A typing environment associates type expressions to variables and has the form

$$\mathcal{E} ::= [\ x_1 : A_1,\ x_2 : A_2,\ ...,\ x_n : A_n\ ]$$

where $x_i \neq x_j$ for all i and j , satisfying $i \neq j$ and $(1 \leq i, j \leq n)$.

We use $\mathcal{C}$, $\mathcal{R}$, $\mathcal{L}$ and $\mathcal{P}$ to denote a global const identifiers' typing environment, a special typing environment (see below), a local typing environment for a layer, and a local typing environment for a protocol respectively. We use $\mathcal{L}_A$, $\mathcal{L}_{B0}$ and $\mathcal{L}_{B1}$ to denote a particular local typing environment specific to the Cell A, Cell B0, and Cell B1 contexts in the current layer environment $\mathcal{L}$. In some cases, we use $\mathcal{L}_{id}$ or $\mathcal{P}_{id}$ to denote a particular local typing environment specific to the context of a layer or a protocol identified by *id*.

We introduce a special typing environment $\mathcal{R}$, which records the read-only register accesses to the last layer and is dynamically changed between the layers. At the beginning, $\mathcal{R}$ is initialized by the global register declarations, which is available to be read at the first layer declared. The it is changed when a new layer is just entered, and become the combination of $\mathcal{L}_A$, $\mathcal{L}_{B0}$ and $\mathcal{L}_{B1}$ in the last layer environment $\mathcal{L}$.

Finally, to provide more confident consistency, we define some parameters syntactically, including the size of a layer register, the size of a cell register, a protocol set and a layer set syntactically. Accordingly, we introduce special global environments $\mathcal{L}reglen$, $\mathcal{C}reglen$, $\mathcal{P}set$ and $\mathcal{L}set$. For convenience, we use $\mathcal{G}$ to denote the combination of them, that is, $\mathcal{G} = (\mathcal{L}reglen, \mathcal{C}reglen, \mathcal{P}set, \mathcal{L}set)$.

5.1.3 Judgements

- $\mathcal{E} \vdash e : A$, implies that,
  under the the well-formed typing environment $\mathcal{E}$, the expression e is well-typed and has the type A. Here, $\mathcal{E}$ can be $\emptyset$, $\mathcal{G}$, or $\mathcal{C}$.

- $\mathcal{E} \vdash \diamond$, means that $\mathcal{E}$ is a well-formed typing environment. Here, $\mathcal{E}$ can be $\emptyset$, $\mathcal{G}$, $\mathcal{C}$, $\mathcal{L}$, $\mathcal{P}$, $\mathcal{L}_A$, $\mathcal{L}_{B0}$ or $\mathcal{L}_{B1}$.

- $\mathcal{G}, \mathcal{C} \vdash e : A$, implies that,
  under the well-formed typing environments $\mathcal{G}$ and $\mathcal{C}$, the expression e is well-typed and has the type A.

- $\mathcal{G}, \mathcal{C}, \mathcal{R} \vdash e : A$, implies that,
  under the well-formed typing environments $\mathcal{G}$, $\mathcal{C}$ and $\mathcal{R}$, the expression e is well-typed and has the type A.

- $\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L} \vdash e : A$, implies that,
  under the well-formed typing environments $\mathcal{G}$, $\mathcal{C}$, $\mathcal{R}$ and $\mathcal{L}$, the expression e is well-typed and has the type A.

- $\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_{\mathcal{C}} \vdash e : A$, implies that,
  under the well-formed typing environments $\mathcal{G}$, $\mathcal{C}$, $\mathcal{R}$, $\mathcal{L}$ and $\mathcal{L}_{\mathcal{C}}$ ($\mathcal{L}_A$, $\mathcal{L}_{B0}$ or $\mathcal{L}_{B1}$), the expression e is well-typed and has the type A.

- $\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash e : A$, implies that,
  under the well-formed typing environments $\mathcal{G}$, $\mathcal{C}$, $\mathcal{R}$, $\mathcal{L}$, $\mathcal{L}_A$ and $\mathcal{P}$, the expression e is well-typed and has the type A.

- $\mathcal{S} \vdash D$, implies that,
  under the the well-formed typing environment $\mathcal{S}$, the parser component D is well-typed. Here, $\mathcal{S}$ can be $\emptyset$, $\mathcal{G}$, or $\mathcal{C}$.

- $\mathcal{G}, \mathcal{C} \vdash D$, implies that,
  under the well-formed typing environments $\mathcal{G}$ and $\mathcal{C}$, the parser component D is well-typed.

- $\mathcal{G}, \mathcal{C}, \mathcal{R} \vdash D$, implies that,
  under the well-formed typing environments $\mathcal{G}$, $\mathcal{C}$ and $\mathcal{R}$, the parser component D is well-typed.

- $\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L} \vdash D$, implies that
  under the well-formed typing environments $\mathcal{G}$, $\mathcal{C}$, $\mathcal{R}$ and $\mathcal{L}$, the parser component D is well-typed.

- $\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_{\mathcal{C}} \vdash D$, implies that
  under the well-formed typing environments $\mathcal{G}$, $\mathcal{C}$, $\mathcal{R}$, $\mathcal{L}$ and $\mathcal{L}_{\mathcal{C}}$ ($\mathcal{L}_A$, $\mathcal{L}_{B0}$ or $\mathcal{L}_{B1}$), the parser component D is well-typed.

- $\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash D$, implies that
  under the well-formed typing environments $\mathcal{G}$, $\mathcal{C}$, $\mathcal{R}$, $\mathcal{L}$, $\mathcal{L}_A$ and $\mathcal{P}$, the parser component D is well-typed.

5.1.4 Typing rules

- Common

$$\frac{}{\emptyset \vdash \diamond} \text{ (C-1)} \qquad \frac{\mathcal{E} \vdash \diamond \qquad x : A \in \mathcal{E}}{\mathcal{E} \vdash x : A} \text{ (C-2)}$$

$$\frac{\mathcal{E}' \vdash \diamond \qquad x \notin dom(\mathcal{E}') \qquad \mathcal{E} = \mathcal{E}' \cup \{x : A\}}{\mathcal{E} \vdash \diamond} \text{ (C-3)}$$

$$\frac{\mathcal{E}' \vdash e : A \qquad y \notin dom(\mathcal{E}') \qquad \mathcal{E} = \mathcal{E}' \cup \{y : A'\}}{\mathcal{E} \vdash e : A} \text{ (C-4)}$$

$$\frac{\mathcal{G} \vdash \diamond \qquad \mathcal{C} \vdash e : A}{\mathcal{G}, \mathcal{C} \vdash e : A} \text{ (C-5)} \qquad \frac{\mathcal{G} \vdash \diamond \qquad \mathcal{C} \vdash \diamond \qquad \mathcal{R} \vdash e : A}{\mathcal{G}, \mathcal{C}, \mathcal{R} \vdash e : A} \text{ (C-6)}$$

- Initialization of $\mathcal{G}$, opened at the beginning of the specification and not to be closed

$$\frac{\mathcal{G} = (\mathcal{L}reglen, \mathcal{C}reglen, \mathcal{P}set, \mathcal{L}set) \qquad \mathcal{L}reglen = k \qquad k > 0}{\mathcal{G} \vdash Lreglen(k)} \text{ (IG-1)}$$

$$\frac{\mathcal{G} = (\mathcal{L}reglen, \mathcal{C}reglen, \mathcal{P}set, \mathcal{L}set) \qquad \mathcal{C}reglen = k \qquad k > 0}{\mathcal{G} \vdash Creglen(k)} \text{ (IG-2)}$$

$$\frac{\begin{array}{c}\mathcal{G} = (\mathcal{L}reglen, \mathcal{C}reglen, \mathcal{P}set, \mathcal{L}set) \\ \mathcal{P}set = \{id_1, \cdots, id_k\} \qquad \forall i, j (1 \le i, j \le k \rightarrow id_i \ne id_j)\end{array}}{\mathcal{G} \vdash Pset(id_1, \cdots, id_k)} \text{ (IG-3)}$$

$$\frac{\begin{array}{c}\mathcal{G} = (\mathcal{L}reglen, \mathcal{C}reglen, \mathcal{P}set, \mathcal{L}set) \\ \mathcal{L}set = \{id_1, \cdots, id_k\} \qquad \forall i, j (1 \le i, j \le k \rightarrow id_i \ne id_j)\end{array}}{\mathcal{G} \vdash Lset(id_1, \cdots, id_k)} \text{ (IG-4)}$$

$$\frac{\begin{array}{c}\mathcal{G} = (\mathcal{L}reglen, \mathcal{C}reglen, \mathcal{P}set, \mathcal{L}set) \\ \mathcal{L}reglen = k \qquad \mathcal{G} \vdash Lreglen(k) \qquad \mathcal{C}reglen = k' \\ \mathcal{G} \vdash Creglen(k') \qquad \mathcal{P}set = \{pid_1, \cdots, pid_p\} \qquad \mathcal{G} \vdash Pset(pid_1, \cdots, pid_p) \\ \mathcal{L}set = \{lid_1, \cdots, lid_l\} \qquad \mathcal{G} \vdash Lset(lid_1, \cdots, lid_l)\end{array}}{\mathcal{G} \vdash \diamond} \text{ (IG-5)}$$

- Initialization of $\mathcal{C}$, opened at the beginning of the specification and not to be closed

$$\frac{\mathcal{C}' \vdash c : (\tau, n) \qquad id \notin dom(\mathcal{C}') \qquad \mathcal{C} = \mathcal{C}' \cup \{id : (\tau, n)\}}{\mathcal{C} \vdash ConstDcl(id, c)} \text{ (IC-1)}$$

$$\frac{val(i)\ is\ a\ signed\ integer\ up\ to\ 32\ bits}{\emptyset \vdash IntConst(i) : (Int, val(i))}\ \text{(IC-2)}$$

$$\frac{val(i)\ is\ the\ decimal\ result\ from\ a\ hexadecimal\ number\ i\ (with\ n\ hexadecimal\ digits)}{\emptyset \vdash HexConst(i) : (Hexes(n), val(i))}\ \text{(IC-3)}$$

$$\frac{val(bs)\ is\ the\ non\ negtive\ integer\ from\ a\ binary\ bit\ string\ bs\ with\ the\ length\ n}{\emptyset \vdash BitSConst(bs) : (Bits(n), val(bs))}\ \text{(IC-4)}$$

- Initialization of $\mathcal{R}$, initialized at the beginning of the specification (Rules IR-1 and IR-2) and each time at the leaving of a layer context (Rule IR-3) , and opened at the beginning of a layer context.

$$\frac{\begin{array}{c} \mathcal{G} \vdash Lreglen(n) \qquad \mathcal{G}, \mathcal{C} \vdash e_1 : (Int, n_1) \\ \mathcal{G}, \mathcal{C} \vdash e_2 : (Int, n_2) \qquad 0 \leq n_2 \leq n_1 < n \qquad id \notin dom(\mathcal{R}') \\ \forall id' \in dom(\mathcal{R}').(\mathcal{G}, \mathcal{C}, \mathcal{R}' \vdash id' : RegAcc(n, n_1', n_2') \rightarrow n_1' < n_2 \vee n_1 < n_2') \\ \mathcal{R} = \mathcal{R}' \cup \{id : RegAcc(n, n_1, n_2)\} \end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R} \vdash IRF(id, e_1, e_2)}\ \text{(IR-1)}$$

$$\frac{\begin{array}{c} \mathcal{G} \vdash Lreglen(n) \qquad \mathcal{G}, \mathcal{C} \vdash e : (Int, k) \qquad 0 \leq k < n \qquad id \notin dom(\mathcal{R}') \\ \forall id' \in dom(\mathcal{R}').(\mathcal{G}, \mathcal{C}, \mathcal{R}' \vdash id' : RegAcc(n, n_1', n_2') \rightarrow n_1' < k \vee k < n_2') \\ \mathcal{R} = \mathcal{R}' \cup \{id : RegAcc(n, k, k)\} \end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R} \vdash IRF(id, e)}\ \text{(IR-2)}$$

$$\frac{\begin{array}{c} \mathcal{G} \vdash Lreglen(n) \qquad \mathcal{G} \vdash Creglen(k) \qquad n = 3 * k \\ \mathcal{R} = \{id : RegAcc(n, 2*k + n_1, 2*k + n_2) \mid id : RegAcc(k, n_1, n_2) \in \mathcal{L}_{\mathcal{A}}\} \\ \cup \{id : RegAcc(n, k + n_1, k + n_2) \mid id : RegAcc(k, n_1, n_2) \in \mathcal{L}_{B0}\} \\ \cup \{id : RegAcc(n, n_1, n_2) \mid id : RegAcc(k, n_1, n_2) \in \mathcal{L}_{B1}\} \end{array}}{\mathcal{R} \vdash \diamond}\ \text{(IR-3)}$$

- Initialization of $\mathcal{L}$, opened at the beginning and closed at the end of a LayerAction specification

$$\frac{\begin{array}{c} \mathcal{G}, \mathcal{C}, \mathcal{R} \vdash ProtocolDecl(pid, protocol) \\ \mathcal{L}' \vdash \diamond \qquad id_i \notin dom(\mathcal{L}'), 1 \leq i \leq k \qquad \mathcal{L} = \mathcal{L}' \cup \{id_i : pid \mid 1 \leq i \leq k\} \end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L} \vdash ProtocolDef(pid, (id_1, \cdots, id_k))}\ \text{(IL)}$$

- Initialization of $\mathcal{L}_{\mathcal{A}}$ at the CellA Registers specification, opened at the beginning and closed at the end of a Cell A specification

$$\frac{\begin{array}{c}\mathcal{G} \vdash \mathit{Creglen}(n) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_{\mathcal{A}}{}' \vdash e_1 : (\mathit{Int}, n_1) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_{\mathcal{A}}{}' \vdash e_2 : (\mathit{Int}, n_2) \\ 0 \leq n_2 \leq n_1 < n \qquad id \notin \mathit{dom}(\mathcal{L}_{\mathcal{A}}{}') \cup \mathit{dom}(\mathcal{R}) \\ \forall id' \in \mathit{dom}(\mathcal{L}_{\mathcal{A}}{}').(\mathcal{L}_{\mathcal{A}}{}' \vdash id' : \mathit{RegAcc}(n, n_1', n_2') \rightarrow n_1' < n_2 \vee n_1 < n_2') \\ \forall id' \in \mathit{dom}(\mathcal{R}).(\mathcal{R} \vdash id' : \mathit{RegAcc}(n, n_1', n_2') \rightarrow n_1' < 2*n+n_2 \vee 2*n+n_1 < n_2' \\ \mathcal{L}_{\mathcal{A}} = \mathcal{L}_{\mathcal{A}}{}' \cup \{id : \mathit{RegAcc}(n, n_1, n_2)\}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_{\mathcal{A}} \vdash \mathit{IRF}(id, e_1, e_2)} \text{ (ILA-1)}$$

$$\frac{\begin{array}{c}\mathcal{G} \vdash \mathit{Creglen}(n) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_{\mathcal{A}}{}' \vdash e : (\mathit{Int}, k) \qquad 0 \leq k < n \qquad id \notin \mathit{dom}(\mathcal{L}_{\mathcal{A}}{}') \cup \mathit{dom}(\mathcal{R}) \\ \forall id' \in \mathit{dom}(\mathcal{L}_{\mathcal{A}}{}').(\mathcal{L}_{\mathcal{A}}{}' \vdash id' : \mathit{RegAcc}(n, n_1', n_2') \rightarrow n_1' < k \vee k < n_2') \\ \forall id' \in \mathit{dom}(\mathcal{R}).(\mathcal{R} \vdash id' : \mathit{RegAcc}(n, n_1', n_2') \rightarrow n_1' < 2*n+k \vee 2*n+k < n_2') \\ \mathcal{L}_{\mathcal{A}} = \mathcal{L}_{\mathcal{A}}{}' \cup \{id : \mathit{RegAcc}(n, k, k)\}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_{\mathcal{A}} \vdash \mathit{IRF}(id, e)} \text{ (ILA-2)}$$

- Initialization of $\mathcal{L}_{B0}$ at the CellB0 Registers specification, opened at the beginning and closed at the end of a Cell B0 specification

$$\frac{\begin{array}{c}\mathcal{G} \vdash \mathit{Creglen}(n) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_{B0}' \vdash e_1 : (\mathit{Int}, n_1) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_{B0}' \vdash e_2 : (\mathit{Int}, n_2) \\ 0 \leq n_2 \leq n_1 < n \qquad id \notin \mathit{dom}(\mathcal{L}_{B0}') \cup \mathit{dom}(\mathcal{R}) \\ \forall id' \in \mathit{dom}(\mathcal{L}_{B0}').(\mathcal{L}_{B0}' \vdash id' : \mathit{RegAcc}(n, n_1', n_2') \rightarrow n_1' < n_2 \vee n_1 < n_2') \\ \forall id' \in \mathit{dom}(\mathcal{R}).(\mathcal{R} \vdash id' : \mathit{RegAcc}(n, n_1', n_2') \rightarrow n_1' < n+n_2 \vee n+n_1 < n_2' \\ \mathcal{L}_{B0} = \mathcal{L}_{B0}' \cup \{id : \mathit{RegAcc}(n, n_1, n_2)\}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_{B0} \vdash \mathit{IRF}(id, e_1, e_2)} \text{ (ILB0-1)}$$

$$\frac{\begin{array}{c}\mathcal{G} \vdash \mathit{Creglen}(n) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_{B0}' \vdash e : (\mathit{Int}, k) \qquad 0 \leq k < n \qquad id \notin \mathit{dom}(\mathcal{L}_{B0}') \cup \mathit{dom}(\mathcal{R}) \\ \forall id' \in \mathit{dom}(\mathcal{L}_{B0}').(\mathcal{L}_{B0}' \vdash id' : \mathit{RegAcc}(n, n_1', n_2') \rightarrow n_1' < k \vee k < n_2') \\ \forall id' \in \mathit{dom}(\mathcal{R}).(\mathcal{R} \vdash id' : \mathit{RegAcc}(n, n_1', n_2') \rightarrow n_1' < n+k \vee n+k < n_2') \\ \mathcal{L}_{B0} = \mathcal{L}_{B0}' \cup \{id : \mathit{RegAcc}(n, k, k)\}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_{B0} \vdash \mathit{IRF}(id, e)} \text{ (ILB0-2)}$$

- Initialization of $\mathcal{L}_{B1}$ at the CellB1 Registers specification, opened at the beginning and closed at the end of a Cell B1 specification

$$\frac{\begin{array}{c}\mathcal{G} \vdash \mathit{Creglen}(n) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}'_{B1} \vdash e_1 : (\mathit{Int}, n_1) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}'_{B1} \vdash e_2 : (\mathit{Int}, n_2) \\ 0 \leq n_2 \leq n_1 < n \qquad id \notin dom(\mathcal{L}'_{B1}) \cup dom(\mathcal{R}) \\ \forall id' \in dom(\mathcal{L}'_{B1}).(\mathcal{L}'_{B1} \vdash id' : \mathit{RegAcc}(n, n'_1, n'_2) \rightarrow n'_1 < n_2 \vee n_1 < n'_2) \\ \forall id' \in dom(\mathcal{R}).(\mathcal{R} \vdash id' : \mathit{RegAcc}(n, n'_1, n'_2) \rightarrow n'_1 < n_2 \vee n_1 < n'_2) \\ \mathcal{L}_{B1} = \mathcal{L}'_{B1} \cup \{id : \mathit{RegAcc}(n, n_1, n_2)\}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_{B1} \vdash \mathit{IRF}(id, e_1, e_2)} \text{(ILB1-1)}$$

$$\frac{\begin{array}{c}\mathcal{G} \vdash \mathit{Creglen}(n) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}'_{B1} \vdash e : (\mathit{Int}, k) \qquad 0 \leq k < n \qquad id \notin dom(\mathcal{L}'_{B1}) \cup dom(\mathcal{R}) \\ \forall id' \in dom(\mathcal{L}'_{B1}).(\mathcal{L}'_{B1} \vdash id' : \mathit{RegAcc}(n, n'_1, n'_2) \rightarrow n'_1 < k \vee k < n'_2) \\ \forall id' \in dom(\mathcal{R}.(\mathcal{R} \vdash id' : \mathit{RegAcc}(n, n'_1, n'_2) \rightarrow n'_1 < k \vee k < n'_2) \\ \mathcal{L}_{B1} = \mathcal{L}'_{B1} \cup \{id : \mathit{RegAcc}(n, k, k)\}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_{B1} \vdash \mathit{IRF}(id, e)} \text{(ILB1-2)}$$

- Initialization of $\mathcal{P}$, opened at each time of the instantialization of a Protocol specification and closed at the end of that instantialization.

$$\frac{\begin{array}{c}\mathit{flds} = ((\mathit{fid}_1 : c_1), \cdots, (\mathit{fid}_k : c_k)) \\ \mathit{ofld} = (\mathit{ofid} : 0) \qquad \forall i : 1 \leq i \leq k.\ (\emptyset \vdash c_i : (\mathit{Int}, n_i)) \\ n = n_1 + n_2 + \cdots + n_k \qquad \forall i (1 \leq i \leq k \rightarrow n_i > 0) \\ \forall i, j (1 \leq i < j \leq k \rightarrow \mathit{fid}_i \neq \mathit{fid}_j) \qquad \forall i.\ (1 \leq i \leq k \rightarrow \mathit{fid}_i \neq \mathit{ofid}) \\ \mathcal{G} \vdash \diamond \qquad \mathcal{C} \vdash \diamond \qquad \mathcal{R} \vdash \diamond \qquad \mathcal{L} \vdash \diamond \qquad \mathcal{L}_{\mathcal{A}} \vdash \diamond \\ \mathcal{P}' \vdash \diamond \qquad \forall i (1 \leq i \leq k \rightarrow \mathit{fid}_i \notin dom(\mathcal{P}')) \qquad \mathit{ofid} \notin dom(\mathcal{P}') \\ \mathcal{P} = \mathcal{P}' \cup \{\mathit{fid}_i : \mathit{FieldAcc}(n, n_1 + \cdots + n_{i-1}, n_1 + \cdots + n_i - 1) \mid 1 \leq i \leq k\} \\ \cup \{\mathit{ofid} : \mathit{FieldAcc}(n, n, \mathit{null})\}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_{\mathcal{A}},\mathcal{P} \vdash (\mathit{Fields}(\mathit{flds}), \mathit{OptionFields}(\mathit{ofld}))} \text{(IP-1)}$$

- Expressions

$$\frac{\begin{array}{c}\mathcal{C} \vdash c : (\tau, n) \qquad \mathcal{G} \vdash \diamond \\ \mathcal{C} \vdash \diamond \qquad \mathcal{R} \vdash \diamond \qquad \mathcal{L} \vdash \diamond \qquad \mathcal{L}_C \vdash \diamond \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_C \vdash \mathit{Econst}(c) : (\tau, n)} \text{ CE-1}$$

$$\frac{\begin{array}{c}\mathcal{C} \vdash c : (\tau, n) \\ \mathcal{G} \vdash \diamond \qquad \mathcal{C} \vdash \diamond \qquad \mathcal{R} \vdash \diamond \qquad \mathcal{L} \vdash \diamond \qquad \mathcal{L}_A \vdash \diamond \qquad \mathcal{P} \vdash \diamond\end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash \mathit{Econst}(c) : (\tau, n)} \text{ CE-2}$$

$$\frac{\mathcal{C} \vdash c : (\tau, n) \qquad \mathcal{G} \vdash \diamond}{\mathcal{G}, \mathcal{C} \vdash \mathit{Econst}(c) : (\tau, n)} \text{ CE-3}$$

$$\frac{\begin{array}{c}\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_C \vdash e : (\tau, m) \\ n = \mathit{trans\_to\_int}(\tau, m) \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_C \vdash \mathit{Eunop}(\mathit{Oint}, e) : (\mathit{Int}, n)} \text{ OINT-1}$$

$$\frac{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash e : (\tau, m) \qquad n = \mathit{trans\_to\_int}(\tau, m)}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash \mathit{Eunop}(\mathit{Oint}, e) : (\mathit{Int}, n)} \text{ OINT-2}$$

$$\frac{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_C \vdash e : \mathit{Bool} \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_C \vdash \mathit{Eunop}(\mathit{Onot}, e) : \mathit{Bool}} \text{ ONOT-1}$$

$$\frac{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash e : \mathit{Bool}}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash \mathit{Eunop}(\mathit{Onot}, e) : \mathit{Bool}} \text{ ONOT-2}$$

$$\frac{\begin{array}{c}\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_C \vdash e : (\mathit{Bits}(n), \mathit{bs}) \\ \mathit{bs}' = \mathit{bit\_wise\_negation}(\mathit{bs}) \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_C \vdash \mathit{Eunop}(\mathit{Oneg}, e) : (\mathit{Bits}(n), \mathit{bs}')} \text{ ONEG-1}$$

$$\frac{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash e : (\mathit{Bits}(n), \mathit{bs}) \qquad \mathit{bs}' = \mathit{bit\_wise\_negation}(\mathit{bs})}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash \mathit{Eunop}(\mathit{Oneg}, e) : (\mathit{Bits}(n), \mathit{bs}')} \text{ ONEG-2}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_1 : (\tau_1, m_1) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_2 : (\tau_2, m_2)\\ \mathit{binop} \in \{\mathit{Oadd}, \mathit{Osub}, \mathit{Omul}, \mathit{Odivint}, \mathit{Omod}\}\\ n = \mathit{do\_binop}(\mathit{binop}, \mathit{trans\_to\_int}(\tau_1, m_1), \mathit{trans\_to\_int}(\tau_2, m_2))\\ \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{Ebinop}(\mathit{binop}, e_1, e_2) : (\mathit{Int}, n)}\ \textsc{BopA-1}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_1 : (\tau_1, m_1) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_2 : (\tau_2, m_2)\\ \mathit{binop} \in \{\mathit{Oadd}, \mathit{Osub}, \mathit{Omul}, \mathit{Odivint}, \mathit{Omod}\}\\ n = \mathit{do\_binop}(\mathit{binop}, \mathit{trans\_to\_int}(\tau_1, m_1), \mathit{trans\_to\_int}(\tau_2, m_2))\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Ebinop}(\mathit{binop}, e_1, e_2) : (\mathit{Int}, n)}\ \textsc{BopA-2}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C} \vdash e_1 : (\tau_1, m_1)\\ \mathcal{G},\mathcal{C} \vdash e_2 : (\tau_2, m_2) \qquad \mathit{binop} \in \{\mathit{Oadd}, \mathit{Osub}, \mathit{Omul}, \mathit{Odivint}, \mathit{Omod}\}\\ n = \mathit{do\_binop}(\mathit{binop}, \mathit{trans\_to\_int}(\tau_1, m_1), \mathit{trans\_to\_int}(\tau_2, m_2))\end{array}}{\mathcal{G},\mathcal{C} \vdash \mathit{Ebinop}(\mathit{binop}, e_1, e_2) : (\mathit{Int}, n)}\ \textsc{BopA-3}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_1 : \mathit{Bool} \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_2 : \mathit{Bool}\\ \mathit{binop} \in \{\mathit{Oand}, \mathit{Oor}\} \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{Ebinop}(\mathit{binop}, e_1, e_2) : \mathit{Bool}}\ \textsc{BopL-1}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_1 : \mathit{Bool}\\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_2 : \mathit{Bool} \qquad \mathit{binop} \in \{\mathit{Oand}, \mathit{Oor}\}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Ebinop}(\mathit{binop}, e_1, e_2) : \mathit{Bool}}\ \textsc{BopL-2}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_1 : (\mathit{Bits}(n), \mathit{bs}_1)\\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_2 : (\mathit{Bits}(n), \mathit{bs}_2) \qquad \mathit{binop} \in \{\mathit{Oband}, \mathit{Obor}, \mathit{Obeor}\}\\ \mathit{bs} = \mathit{bit\_wise\_operation}(\mathit{binop}, \mathit{bs}_1, \mathit{bs}_2) \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{Ebinop}(\mathit{binop}, e_1, e_2) : (\mathit{Bits}(n), \mathit{bs})}\ \textsc{BopB-1}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_1 : (\mathit{Bits}(n), \mathit{bs}_1)\\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_2 : (\mathit{Bits}(n), \mathit{bs}_2) \qquad \mathit{binop} \in \{\mathit{Oband}, \mathit{Obor}, \mathit{Obeor}\}\\ \mathit{bs} = \mathit{bit\_wise\_operation}(\mathit{binop}, \mathit{bs}_1, \mathit{bs}_2)\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Ebinop}(\mathit{binop}, e_1, e_2) : (\mathit{Bits}(n), \mathit{bs})}\ \textsc{BopB-2}$$

$$\frac{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_1 : \tau \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_2 : \tau \\ \mathit{binop} \in \{\mathit{Oeq}, \mathit{One}, \mathit{Olt}, \mathit{Ogt}, \mathit{Ole}, \mathit{Oge}\} \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{Ebinop}(\mathit{binop}, e_1, e_2) : \mathit{Bool}}\ \textsc{BopR-1}$$

$$\frac{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_1 : \tau \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_2 : \tau \qquad \mathit{binop} \in \{\mathit{Oeq}, \mathit{One}, \mathit{Olt}, \mathit{Ogt}, \mathit{Ole}, \mathit{Oge}\}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Ebinop}(\mathit{binop}, e_1, e_2) : \mathit{Bool}}\ \textsc{BopR-2}$$

$$\frac{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_1 : \tau \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_2 : \mathit{Int} \\ \mathit{binop} \in \{\mathit{Osl}, \mathit{Osr}\} \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{Ebinop}(\mathit{binop}, e_1, e_2) : \tau}\ \textsc{BopS-1}$$

$$\frac{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_1 : \tau \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_2 : \mathit{Int} \qquad \mathit{binop} \in \{\mathit{Osl}, \mathit{Osr}\}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Ebinop}(\mathit{binop}, e_1, e_2) : \tau}\ \textsc{BopS-2}$$

$$\frac{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_1 : \mathit{Bits}(n_1) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_2 : \mathit{Bits}(n_2) \qquad n = n_1 + n_2 \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{Ebinop}(\mathit{Obc}, e_1, e_2) : \mathit{Bits}(n)}\ \textsc{BopC-1}$$

$$\frac{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_1 : \mathit{Bits}(n_1) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_2 : \mathit{Bits}(n_2) \qquad n = n_1 + n_2}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Ebinop}(\mathit{Obc}, e_1, e_2) : \mathit{Bits}(n)}\ \textsc{BopC-1'}$$

$$\frac{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_1 : \mathit{Hexes}(n_1) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_2 : \mathit{Hexes}(n_2) \\ n = n_1 + n_2 \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{Ebinop}(\mathit{Obc}, e_1, e_2) : \mathit{Hexes}(n)}\ \textsc{BopC-2}$$

$$\frac{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_1 : \mathit{Hexes}(n_1) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_2 : \mathit{Hexes}(n_2) \qquad n = n_1 + n_2}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Ebinop}(\mathit{Obc}, e_1, e_2) : \mathit{Hexes}(n)}\ \textsc{BopC-2'}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_1 : \mathit{RegAcc}(k,n_1,n_2) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_2 : \mathit{RegAcc}(k,m_1,m_2) \qquad n_2 = m_1+1 \\ 0 \le m_2 \le m_1 < n_2 \le n_1 < k \qquad \mathcal{L}_C \textit{ is } \mathcal{L}_A, \mathcal{L}_{B0} \textit{ or } \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{Ebinop}(\mathit{Obc},e_1,e_2) : \mathit{RegAcc}(k,n_1,m_2)} \textsc{BopC-3}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_1 : \mathit{RegAcc}(k,n_1,n_2) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_2 : \mathit{RegAcc}(k,m_1,m_2) \\ n_2 = m_1+1 \qquad 0 \le m_2 \le m_1 < n_2 \le n_1 < k\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Ebinop}(\mathit{Obc},e_1,e_2) : \mathit{RegAcc}(k,n_1,m_2)} \textsc{BopC-3'}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_1 : \mathit{FieldAcc}(id,k,n_1,n_2) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_2 : \mathit{FieldAcc}(id,k,m_1,m_2) \qquad m_1 = n_2+1 \\ 0 \le n_1 \le n_2 < m_1 \le m_2 < k \qquad \mathcal{L}_C \textit{ is } \mathcal{L}_A, \mathcal{L}_{B0} \textit{ or } \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{Ebinop}(\mathit{Obc},e_1,e_2) : \mathit{FieldAcc}(id,k,n_1,m_2)} \textsc{BopC-4}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_1 : \mathit{FieldAcc}(k,n_1,n_2) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_2 : \mathit{FieldAcc}(k,m_1,m_2) \\ m_1 = n_2+1 \qquad 0 \le n_1 \le n_2 < m_1 \le m_2 < k\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Ebinop}(\mathit{Obc},e_1,e_2) : \mathit{FieldAcc}(k,n_1,m_2)} \textsc{BopC-4'}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{L},\mathcal{L}_C \vdash e_1 : (\tau,m) \qquad \mathcal{G},\mathcal{C},\mathcal{L},\mathcal{L}_C \vdash e_2 : (\mathit{Int},n) \\ n \ge \mathit{num\_of\_digits}(\mathit{trans\_to\_hex}(\tau,m)) \qquad \mathcal{L}_C \textit{ is } \mathcal{L}_A, \mathcal{L}_{B0} \textit{ or } \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{L},\mathcal{L}_C \vdash \mathit{Ebinop}(\mathit{Ohexes},e_1,e_2) : \mathit{Hexes}(n)} \textsc{BopH-1}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_1 : (\tau,m) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_2 : (\mathit{Int},n) \\ n \ge \mathit{num\_of\_digits}(\mathit{trans\_to\_hex}(\tau,m))\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Ebinop}(\mathit{Ohexes},e_1,e_2) : \mathit{Hexes}(n)} \textsc{BopH-2}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_1 : (\tau,m) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_2 : (\mathit{Int},n) \\ n \ge \mathit{num\_of\_bits}(\mathit{trans\_to\_binary\_number}(\tau,m)) \\ \mathcal{L}_C \textit{ is } \mathcal{L}_A, \mathcal{L}_{B0} \textit{ or } \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{Ebinop}(\mathit{Obits},e_1,e_2) : \mathit{Bits}(n)} \textsc{BopBT-1}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_1 : (\tau,m) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_2 : (\mathit{Int},n) \\ n \ge \mathit{num\_of\_bits}(\mathit{trans\_to\_binary\_number}(\tau,m))\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Ebinop}(\mathit{Obits},e_1,e_2) : \mathit{Bits}(n)} \textsc{BopBT-2}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L}\vdash id : pid \\ \mathcal{G},\mathcal{C},\mathcal{R}\vdash \mathit{ProtocolDecl}(pid, \mathit{Protocol}(\mathit{Fields}(\mathit{flds}), \mathit{OptionFields}(\mathit{oflds})), \cdots)) \\ \mathit{flds} = ((\mathit{fid}_1 : c_1), \cdots, (\mathit{fid}_k : c_k)) \\ \mathit{ofld} = (\mathit{ofid} : \mathit{null}) \qquad \forall i : 1 \leq i \leq k.\ (\emptyset \vdash c_i : (\mathit{Int}, n_i)) \\ n = n_1 + n_2 + \cdots + n_k \qquad \exists i.\ \mathit{fid} = \mathit{fid}_i \qquad \mathcal{L}_C\ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{Efield}(id, \mathit{fid}) : \mathit{FieldAcc}(id, n, n_1 + \cdots + n_{i-1}, n_1 + \cdots + n_i - 1)}\ \textsc{Efield}$$

$$\frac{\begin{array}{c}\mathcal{G}\vdash \mathit{Creglen}(n) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_1 : \mathit{RegAcc}(n, n_1, n_2) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_2 : (\mathit{Int}, n') \\ 0 \leq n_2 \leq n_1 < n \qquad 0 \leq n' \leq n_1 - n_2 \qquad \mathcal{L}_C\ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{EFieldBit}(e_1, e_2) : \mathit{RegAcc}(n, n_2 + n', n_2 + n')}\ \text{FB-1}$$

$$\frac{\begin{array}{c}\mathcal{G}\vdash \mathit{Creglen}(n) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_1 : \mathit{RegAcc}(n, n_1, n_2) \\ \mathcal{G},\mathcal{C},\mathcal{G},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_2 : (\mathit{Int}, n') \\ 0 \leq n_2 \leq n_1 < n \qquad 0 \leq n' \leq n_1 - n_2\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{EFieldBit}(e_1, e_2) : \mathit{RegAcc}(n, n_2 + n', n_2 + n')}\ \text{FB-1'}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_1 : \mathit{FieldAcc}(id, n, n_1, n_2) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_2 : (\mathit{Int}, n') \\ 0 \leq n_1 \leq n_2 < n \qquad 0 \leq n' \leq n_2 - n_1 \qquad \mathcal{L}_C\ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{EFieldBit}(e_1, e_2) : \mathit{FieldAcc}(id, n, n_1 + n', n_1 + n')}\ \text{FB-2}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_1 : \mathit{FieldAcc}(n, n_1, n_2) \\ \mathcal{G},\mathcal{C},\mathcal{G},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_2 : (\mathit{Int}, n') \\ 0 \leq n_1 \leq n_2 < n \qquad 0 \leq n' \leq n_2 - n_1\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{EFieldBit}(e_1, e_2) : \mathit{FieldAcc}(n, n_1 + n', n_1 + n')}\ \text{FB-2'}$$

$$\frac{\begin{array}{c}\mathcal{G} \vdash \mathit{Creglen}(n) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_1 : \mathit{RegAcc}(n, n_1, n_2) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_2 : (Int, n') \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_3 : (Int, n'') \\ 0 \le n_2 \le n_1 < n \qquad 0 \le n'' \le n' \le n_1 - n_2 \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{EFieldSection}(e_1, e_2, e_3) : \mathit{RegAcc}(n, n_2 + n'', n_2 + n')} \text{ FS-1}$$

$$\frac{\begin{array}{c}\mathcal{G} \vdash \mathit{Creglen}(n) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_1 : \mathit{RegAcc}(n, n_1, n_2) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_2 : (Int, n') \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_3 : (Int, n'') \\ 0 \le n_2 \le n_1 < n \qquad 0 \le n'' \le n' \le n_1 - n_2\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{EFieldSection}(e_1, e_2, e_3) : \mathit{RegAcc}(n, n_2 + n'', n_2 + n')} \text{ FS-1'}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_1 : \mathit{FieldAcc}(id, n, n_1, n_2) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_2 : (Int, n') \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_3 : (Int, n'') \\ 0 \le n_1 \le n_2 < n \qquad 0 \le n'' \le n' \le n_2 - n_1 \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{EFieldSection}(e_1, e_2, e_3) : \mathit{FieldAcc}(id, n, n_1 + n'', n_1 + n')} \text{ FS-2}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_1 : \mathit{FieldAcc}(n, n_1, n_2) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_2 : (Int, n') \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_3 : (Int, n'') \\ 0 \le n_1 \le n_2 < n \qquad 0 \le n'' \le n' \le n_2 - n_1\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{EFieldSection}(e_1, e_2, e_3) : \mathit{FieldAcc}(n, n_1 + n'', n_1 + n')} \text{ FS-2'}$$

$$\frac{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L} \vdash id : pid \qquad \mathcal{G},\mathcal{C},\mathcal{R} \vdash \mathit{ProtocolDecl}(pid, protocol)}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L} \vdash \mathit{ProtLen}(id) : \mathit{Int}} \text{ (PLen)}$$

- Instructions

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash ra : \mathit{RegAcc}(n', n_1, n_2) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e : (\tau, m) \qquad \mathit{trans\_to\_bits\_type}(\tau, m) = (\mathit{Bits}(n), m) \\ n = n_1 - n_2 + 1 \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{Set}(ra, e)} \ \textsc{Set-1}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash ra : \mathit{RegAcc}(n', n_1, n_2) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e : (\tau, m) \\ \mathit{trans\_to\_bits}(\tau, m) = (\mathit{Bits}(n), m) \qquad n = n_1 - n_2 + 1\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Set}(ra, e)} \ \textsc{Set-2}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash mra : \mathit{RegAcc}(n', n_1, n_2) \\ m = n_1 - n_2 + 1 \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e : \tau \\ \tau = \mathit{Bits}(m) \vee \tau = \mathit{RegAcc}(n_r, r', r'') \vee \tau = \mathit{FieldAcc}(id, n_f, f', f'') \\ m = r' - r'' + 1 = f'' - f' + 1 \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{Mov}(mra, e)} \ \textsc{Mov-1}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash mra : \mathit{RegAcc}(n', n_1, n_2) \\ m = n_1 - n_2 + 1 \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e : \tau \\ \tau = \mathit{Bits}(m) \vee \tau = \mathit{RegAcc}(n_r, r', r'') \vee \tau = \mathit{FieldAcc}(id, n_f, f', f'') \\ m = r' - r'' + 1 = f'' - f' + 1\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Mov}(mra, e)} \ \textsc{Mov-2}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash ra : \mathit{RegAcc}(n', n_1, n_2) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e : (\tau, m) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e' : (\tau', m') \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{Eq}(ra, e, e')} \ \textsc{Eq-1}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash ra : \mathit{RegAcc}(n', n_1, n_2) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e : (\tau, m) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e' : (\tau', m')\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Eq}(ra, e, e')} \ \textsc{Eq-2}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash ra : \mathit{RegAcc}(n', n_1, n_2) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e : (\tau, m) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e' : (\tau', m') \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{Lg}(ra, e, e')} \ \textsc{Lg-1}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash ra : \mathit{RegAcc}(n', n_1, n_2) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e : (\tau, m) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e' : (\tau', m')\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Lg}(ra, e, e')} \ \textsc{Lg-2}$$

- Access of registers in instructions

$$\frac{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{id} : \mathit{RegAcc}(n,n_1,n_2) \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A,\mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{TargetRegAccName}(\mathit{id}) : \mathit{RegAcc}(n,n_1,n_2)}\ \textsc{TRegAcc-1}$$

$$\frac{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{id} : \mathit{RegAcc}(n,n_1,n_2)}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{TargetRegAccName}(\mathit{id}) : \mathit{RegAcc}(n,n_1,n_2)}\ \textsc{TRegAcc-1'}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{tran} : \mathit{RegAcc}(n,m_1,m_2)\\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_1 : (\mathit{Int},k_1)\\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_2 : (\mathit{Int},k_2) \qquad 0 \le k_2 \le k_1 \le m_1 - m_2\\ n_1 = m_2 + k_1 \qquad n_2 = m_2 + k_2 \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A,\mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{TargetRegAccName}(\mathit{tran},e_1,e_2) : \mathit{RegAcc}(n,n_1,n_2)}\ \textsc{TRegAcc-2}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{tran} : \mathit{RegAcc}(n,m_1,m_2)\\ \mathcal{G},\mathcal{C},\mathcal{L},\mathcal{L}_C \vdash e_1 : (\mathit{Int},k_1) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_2 : (\mathit{Int},k_2)\\ 0 \le k_2 \le k_1 \le m_1 - m_2 \qquad n_1 = m_2 + k_1 \qquad n_2 = m_2 + k_2\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{TargetRegAccName}(\mathit{tran},e_1,e_2) : \mathit{RegAcc}(n,n_1,n_2)}\ \textsc{TRegAcc-2'}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{tran} : \mathit{RegAcc}(n,m_1,m_2) \qquad \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e : (\mathit{Int},k)\\ 0 \le k \le m_1 - m_2 \qquad m = m_2 + k \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A,\mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{TargetRegAccName}(\mathit{tran},e) : \mathit{RegAcc}(n,m,m)}\ \textsc{TRegAcc-3}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{tran} : \mathit{RegAcc}(n,m_1,m_2)\\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e : (\mathit{Int},k) \qquad 0 \le k \le m_1 - m_2 \qquad m = m_2 + k\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{TargetRegAccName}(\mathit{tran},e) : \mathit{RegAcc}(n,m,m)}\ \textsc{TRegAcc-3'}$$

$$\frac{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{tra} : \mathit{RegAcc}(n,m_1,m_2) \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A,\mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{MovRegAccName}(\mathit{tra}) : \mathit{RegAcc}(n,m_1,m_2)}\ \textsc{MRegAcc-1}$$

$$\frac{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{tra} : \mathit{RegAcc}(n,m_1,m_2)}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{MovRegAccName}(\mathit{tra}) : \mathit{RegAcc}(n,m_1,m_2)}\ \textsc{MRegAcc-1'}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{mra} : \mathit{RegAcc}(n, m_1, m_2) \\ \mathcal{G},\mathcal{C},\mathcal{L},\mathcal{L},\mathcal{L}_C \vdash \mathit{tra} : \mathit{RegAcc}(n, n_1, n_2) \\ m_2 = n_1 + 1 \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{MovRegAccName}(\mathit{mra}, \mathit{tra}) : \mathit{RegAcc}(n, m_1, n_2)} \ \textsc{MRegAcc-2}$$

$$\frac{\begin{array}{c}\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{mra} : \mathit{RegAcc}(n, m_1, m_2) \\ \mathcal{G},\mathcal{C},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{tra} : \mathit{RegAcc}(n, n_1, n_2) \qquad m_2 = n_1 + 1\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{MovRegAccName}(\mathit{mra}, \mathit{tra}) : \mathit{RegAcc}(n, m_1, n_2)} \ \textsc{MRegAcc-2'}$$

- Action statement

$$\frac{\forall i : 1 \leq i \leq k.\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{ins}_i \qquad \mathcal{L}_C \ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{Action}(\mathit{ins}_1, \cdots, \mathit{ins}_k)} \ \text{AS-1}$$

$$\frac{\forall i : 1 \leq i \leq k.\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{ins}_i}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Action}(\mathit{ins}_1, \cdots, \mathit{ins}_k)} \ \text{AS-2}$$

- Bypass statement

$$\frac{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A \vdash c : (\mathit{Int}, n) \qquad n = 0 \lor n = 1 \lor n = 2}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A \vdash \mathit{Bypass}(c)} \ \textsc{BypS-1}$$

$$\frac{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash c : (\mathit{Int}, n) \qquad n = 0 \lor n = 1 \lor n = 2}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Bypass}(c)} \ \textsc{BypS-2}$$

- NextHeader statement

$$\frac{\begin{array}{c}\mathcal{G} \vdash \mathit{Pset}(\mathit{id}_1, \cdots, \mathit{id}_k) \qquad \mathit{id} \in \{\mathit{id}_1, \cdots, \mathit{id}_k\} \\ \mathcal{G} \vdash \diamond \qquad \mathcal{C} \vdash \diamond \qquad \mathcal{R} \vdash \diamond \qquad \mathcal{L} \vdash \diamond \qquad \mathcal{L}_\mathcal{A} \vdash \diamond\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A \vdash \mathit{NextHeader}(\mathit{id})} \ \textsc{NextHeader-1}$$

$$\frac{\begin{array}{c}\mathcal{G} \vdash \mathit{Pset}(\mathit{id}_1, \cdots, \mathit{id}_k) \qquad \mathit{id} \in \{\mathit{id}_1, \cdots, \mathit{id}_k\} \\ \mathcal{G} \vdash \diamond \qquad \mathcal{C} \vdash \diamond \qquad \mathcal{R} \vdash \diamond \qquad \mathcal{L} \vdash \diamond \qquad \mathcal{L}_\mathcal{A} \vdash \diamond \qquad \mathcal{P} \vdash \diamond\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{NextHeader}(\mathit{id})} \ \textsc{NextHeader-2}$$

- Length statement

$$\frac{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A \vdash e : (\mathit{Int}, n)}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A \vdash \mathit{Length}(e)} \ \textsc{Length-1} \quad \frac{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e : (\mathit{Int}, n)}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Length}(e)} \ \textsc{Length-2}$$

- Layer statement

$$\frac{\begin{array}{c}\forall i: 1 \leq i \leq n.\ (ls_i = Action(ins_1, \cdots, ins_k) \rightarrow \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_C \vdash Action(ins_1, \cdots, ins_k)) \\ \forall i: 1 \leq i \leq n.\ (ls_i = Bypass(c) \rightarrow \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A \vdash Bypass(c)) \\ \forall i: 1 \leq i \leq n.\ (ls_i = NextHeader(id) \rightarrow \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A \vdash NextHeader(id) \\ \forall i: 1 \leq i \leq n.\ (ls_i = Length(e) \rightarrow \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A \vdash Length(e) \\ \forall i: 1 \leq i \leq n.\ (ls_i = IfElseL(if\_l\_list, d\_l) \rightarrow \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_C \vdash IfElseL(if\_l\_list, d\_l)) \\ \mathcal{L}_C \textit{ is } \mathcal{L}_A, \mathcal{L}_{B0} \textit{ or } \mathcal{L}_{B1}\end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_C \vdash (ls_1, \cdots, ls_n)} \text{ LSL}$$

$$\frac{\begin{array}{c}if\_l\_list = ((e_1, l\_stmts_1), \cdots, (e_k, l\_stmts_k)) \\ d\_l = l\_stmts \qquad \forall i: 1 \leq i \leq k.\ \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_C \vdash e_k : Bool \\ \forall i: 1 \leq i \leq k.\ \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_C \vdash l\_stmts_i \\ \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_C \vdash d\_l \qquad \mathcal{L}_C \textit{ is } \mathcal{L}_A, \mathcal{L}_{B0} \textit{ or } \mathcal{L}_{B1}\end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_C \vdash IfElseL(if\_l\_list, d\_l)} \text{ IFEL}$$

- Layer local actions

$$\frac{\begin{array}{c}caas = CellA(ca\_l\_s\_list) \\ cb0as = CellB0(cb0\_l\_s\_list) \qquad cb1as = CellB1(cb1\_l\_s\_list) \\ \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_C \vdash ca\_l\_s\_list \qquad \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_C \vdash cb0\_l\_s\_list \\ \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_C \vdash cb1\_l\_s\_list \qquad \mathcal{L}_C \textit{ is } \mathcal{L}_A, \mathcal{L}_{B0} \textit{ or } \mathcal{L}_{B1}\end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_C \vdash LocalActions(caas, cb0as, cb1as)} \text{ LLA}$$

- Layer local register declarations

$$\frac{\begin{array}{c}cars = CellARegs(ca\_ra\_ss\_list) \\ cb0rs = CellB0Regs(cb0\_ra\_ss\_list) \\ cb1rs = CellB1Regs(cb1\_ra\_ss\_list) \\ \forall ras \in ca\_ra\_ss\_list.\ \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L} \vdash ras \\ \forall ras \in cb0\_ra\_ss\_list.\ \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L} \vdash ras \\ \forall ras \in cb1\_ra\_ss\_list.\ \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L} \vdash ras\end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L} \vdash LocalRegs(cars, cb0rs, cb1rs)} \text{ LLRD}$$

- Layer action

$$\frac{\begin{array}{c}\mathcal{G} \vdash Lset(id_1, \cdots, id_k) \qquad id \in \{id_1, \cdots, id_k)\} \qquad \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}_{id} \vdash lvs \\ \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}_{id} \vdash lrd \qquad \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}_{id} \vdash ld \qquad \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}_{id} \vdash las\end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R} \vdash LayerAction(id, lvs, lrd, ld, las)} \text{ LA}$$

- Protocol statement

$$\frac{\begin{array}{c}\forall i : 1 \leq i \leq n.\ (ps_i = \mathit{Action}(ins_1, \cdots, ins_k) \rightarrow \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash \mathit{Action}(ins_1, \cdots, ins_k) \\ \forall i : 1 \leq i \leq n.\ (ps_i = \mathit{IfElseP}(\mathit{if\_p\_list}, d\_p \rightarrow \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash \mathit{IfElseP}(\mathit{if\_p\_list}, d\_p) \\ \forall i : 1 \leq i \leq n.\ (ps_i = \mathit{NextHeader}(id) \rightarrow \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash \mathit{NextHeader}(id) \\ \forall i : 1 \leq i \leq n.\ (ps_i = \mathit{Bypass}(c) \rightarrow \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash \mathit{Bypass}(c) \\ \forall i : 1 \leq i \leq n.\ (ps_i = \mathit{Length}(e) \rightarrow \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash \mathit{Length}(e)\end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash (ps_1, \cdots, ps_n)}\ \text{PSL}$$

$$\frac{\begin{array}{c}\mathit{if\_p\_list} = ((e_1, p\_stmts_1), \cdots, (e_k, p\_stmts_k)) \\ d\_p = p\_stmts \qquad \forall i : 1 \leq i \leq k.\ \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash e_k : \mathit{Bool} \\ \forall i : 1 \leq i \leq k.\ \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash p\_stmts_i \\ \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash p\_stmts\end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash \mathit{IfElseL}(\mathit{if\_p\_list}, d\_p)}\ \text{IFEP}$$

$$\frac{\begin{array}{c}\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash (\mathit{Fields}(\mathit{flds}), \mathit{OptionFields}(\mathit{oflds})) \\ \mathit{flds} = ((\mathit{fld}_1 : c_1), \cdots, (\mathit{fld}_k : c_k)) \\ \emptyset \vdash c_1 : (\mathit{Int}, n_1), \cdots, \emptyset \vdash c_k : (\mathit{Int}, n_k) \\ \emptyset \vdash e : (\mathit{Int}, n) \qquad n * 8 \geq n_1 + \cdots + n_k\end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash \mathit{Length}(e)}\ \textsc{Length-P}$$

- Protocol declaration

$$\frac{\begin{array}{c}\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash \mathit{fields} \\ p\_stmts = (ps_1, \cdots, ps_m) \qquad \forall i : 1 \leq i \leq m.\ \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash ps_i\end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash \mathit{Protocol}(\mathit{fields}, p\_stmts)}\ \textsc{Protocol}$$

$$\frac{\begin{array}{c}\mathcal{G} \vdash \mathit{Pset}(id_1, \cdots, id_k) \\ id \in \{id_1, \cdots, id_k)\} \qquad \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P}_{id} \vdash p\end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A \vdash \mathit{ProtocolDecl}(id, p)}\ \text{PD}$$

- Global declarations

$$\frac{\begin{array}{c}\mathcal{G} \vdash Lset(id_1, \cdots, id_k) \\ \forall lid \in \{id_1, \cdots, id_k)\}.\ (ProtocolDef(id, \cdots) \textit{ is declared at the layer lid} \rightarrow \\ \mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}_{lid}, \mathcal{L}_A \vdash ProtocolDecl(id, p))\end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R} \vdash ProtocolDecl(id, p)} \text{ PDG}$$

$$\frac{\begin{array}{c}\forall i : 1 \leq i \leq n.\ (decl_i = ConstDecl(consdcl) \rightarrow \mathcal{C} \vdash consdcl) \\ \forall i : 1 \leq i \leq n.\ (decl_i = RegAccSet(regacc) \rightarrow \mathcal{G}, \mathcal{C}, \mathcal{R} \vdash regacc) \\ \forall i : 1 \leq i \leq n.\ (decl_i = ProtocolDecl(pdcl) \rightarrow \mathcal{G}, \mathcal{C}, \mathcal{R} \vdash pdcl) \\ \forall i : 1 \leq i \leq n.\ (decl_i = LayerAction(lact) \rightarrow \mathcal{G}, \mathcal{C}, \mathcal{R} \vdash lact) \\ \mathcal{G} \vdash Lset(id_1, \cdots, id_k) \\ \forall lid \in \{id_1, \cdots, id_k)\}.\ LayerAction(id, lvs, lrd, ld, las) \textit{ is declared in the same order}\end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R} \vdash (decl_1, \cdots, decl_n)} \text{ GDECL}$$

- Parser Specification

$$\frac{\begin{array}{c}\mathcal{G} \vdash l\_reg\_len \\ \mathcal{G} \vdash c\_reg\_len \qquad \mathcal{G} \vdash p\_set \qquad \mathcal{G} \vdash l\_set \qquad \mathcal{G}, \mathcal{C}, \mathcal{R} \vdash decls\end{array}}{\emptyset \vdash Parser(l\_reg\_len, c\_reg\_len, p\_set, l\_set, decls)} \text{ PSPEC}$$

5.2 Implementation and Verification

The implementation of the type checking is in Coq first and then extracted into the OCaml code. Based on the typing rules in the subsection 5.1, we define:

- $wt(p)$: p is well-typed P3 AST

To verify the correctness of the type checking program *type_checker*, we will prove two properties as follows:

- *Soundness.* $\forall p, \exists p'.\ type\_checker(p) = OK(p') \rightarrow wt(p')$
- *Completeness.* $\forall p, \exists p'.\ wt(p) \rightarrow type\_checker(p) = OK(p')$

6 Translations

Referring to Fig.1, we have two steps of the translations in the compiler: from a P3 AST to its P3 assembly, and from the P3 assembly to the configuration File.

6.1 Translation of AST to the P3 Assembly

The subsection 6.1.1 gives the abstract syntax of the P3 assembly generated from a P3 AST.

6.1.1 Abstract Syntax of the P3 assembly

$\langle \textit{parser\_asm} \rangle ::= \langle \textit{layer\_reg\_len} \rangle\ \langle \textit{cell\_reg\_len} \rangle\ \{\ \langle \textit{layer\_block} \rangle\ \}$

$\langle \textit{layer\_reg\_len} \rangle ::= \textit{Lreglen}\ (\ \langle \textit{num} \rangle\ )$

$\langle \textit{cell\_reg\_len} \rangle ::= \textit{Creglen}\ (\ \langle \textit{num} \rangle\ )$

$\langle \textit{layer\_block} \rangle ::= \textit{LayerBlock}\ (\ \langle \textit{layer\_id} \rangle, \langle \textit{pins} \rangle, \langle \textit{cella} \rangle, \langle \textit{cellb0} \rangle, \langle \textit{cellb1} \rangle\ )$

$\langle \textit{layer\_id} \rangle ::= \textit{IDENT}$

$\langle \textit{pins} \rangle ::= \{\ \textit{Pins}(\ \langle \textit{ins\_name} \rangle, \langle \textit{ins\_size} \rangle\ )\ \}$

$\langle \textit{cella} \rangle ::= \textit{CellA}(\ \langle \textit{cella\_pb} \rangle, \langle \textit{cella\_pc\_cur} \rangle, \langle \textit{cella\_pc\_nxt} \rangle\ )$

$\langle \textit{cellb0} \rangle ::= \textit{CellB0}(\ \langle \textit{cellb0\_pb} \rangle, \langle \textit{cellb0\_pc\_cur} \rangle\ )$

$\langle \textit{cellb1} \rangle ::= \textit{CellB0}(\ \langle \textit{cellb1\_pb} \rangle, \langle \textit{cellb1\_pc\_cur} \rangle\ )$

$\langle \textit{cella\_pb} \rangle ::= \textit{Apb}(\ \{\ \langle \textit{cella\_pb\_item} \rangle\ \}\ )$

$\langle \textit{cella\_pc\_cur} \rangle ::= \textit{ApcCur}(\ \{\ \langle \textit{cella\_pc\_cur\_item} \rangle\ \}\ )$

$\langle \textit{cella\_pc\_nxt} \rangle ::= \textit{ApcNxt}(\ \{\ \langle \textit{cella\_pc\_nxt\_item} \rangle\ \}\ )$

$\langle \textit{cellb0\_pb} \rangle ::= \textit{B0pb}(\ \{\ \langle \textit{cellb0\_pb\_item} \rangle\ \}\ )$

$\langle \textit{cellb0\_pc\_cur} \rangle ::= \textit{B0pcCur}(\ \{\ \langle \textit{cellb0\_pc\_cur\_item} \rangle\ \}\ )$

$\langle \textit{cellb1\_pb} \rangle ::= \textit{B1pb}(\ \{\ \langle \textit{cellb1\_pb\_item} \rangle\ \}\ )$

$\langle \textit{cellb1\_pc\_cur} \rangle ::= \textit{B1pcCur}(\ \{\ \langle \textit{cellb1\_pc\_cur\_item} \rangle\ \}\ )$

$\langle \textit{cella\_pb\_item} \rangle ::= (\ \langle \textit{hdr\_id} \rangle, \langle \textit{cond\_list} \rangle, \langle \textit{sub\_id} \rangle, \langle \textit{nxt\_id} \rangle, \langle \textit{bypas} \rangle\ )$

$\langle \textit{cella\_pc\_cur\_item} \rangle ::= (\ \langle \textit{sub\_id} \rangle, \langle \textit{cmd\_list} \rangle, \langle \textit{lyr\_offset} \rangle\ )$

$\langle \textit{cella\_pc\_nxt\_item} \rangle ::= (\ \langle \textit{nxt\_id} \rangle, \langle \textit{cella\_nxt} \rangle, \langle \textit{cellb0\_nxt} \rangle, \langle \textit{cellb1\_nxt} \rangle\ )$

$\langle \textit{cellb0\_pb\_item} \rangle ::= (\ \langle \textit{hdr\_id} \rangle, \langle \textit{cond\_list} \rangle, \langle \textit{sub\_id} \rangle\ )$

$\langle \textit{cellb0\_pc\_cur\_item} \rangle ::= (\ \langle \textit{sub\_id} \rangle, \langle \textit{cmd\_list} \rangle\ )$

$\langle \mathit{cellb1\_pb\_item} \rangle$ ::= ($\langle \mathit{hdr\_id} \rangle$, $\langle \mathit{cond\_list} \rangle$, $\langle \mathit{sub\_id} \rangle$)

$\langle \mathit{cellb1\_pc\_cur\_item} \rangle$ ::= ($\langle \mathit{sub\_id} \rangle$, $\langle \mathit{cmd\_list} \rangle$)

$\langle \mathit{cond\_list} \rangle$ ::= *Conds*($\langle \mathit{cond} \rangle$ {, $\langle \mathit{cond} \rangle$ })

$\langle \mathit{cmd\_list} \rangle$ ::= *Cmds*($\langle \mathit{cmd} \rangle$ {, $\langle \mathit{cmd} \rangle$ })

$\langle \mathit{hdr\_id} \rangle$ ::= *HdrID*($\langle \mathit{num} \rangle$)

$\langle \mathit{sub\_id} \rangle$ ::= *SubID*($\langle \mathit{num} \rangle$)

$\langle \mathit{nxt\_id} \rangle$ ::= *NxtID*($\langle \mathit{num} \rangle$)

$\langle \mathit{bypas} \rangle$ ::= *Bypas*($\langle \mathit{num} \rangle$)

$\langle \mathit{lyr\_offset} \rangle$ ::= *LyrOffset*($\langle \mathit{num} \rangle$)

$\langle \mathit{cella\_nxt} \rangle$ ::= *CellANxt*($\langle \mathit{irf\_offsets} \rangle$, $\langle \mathit{prot\_offsets} \rangle$)

$\langle \mathit{cellb0\_nxt} \rangle$ ::= *CellB0Nxt*($\langle \mathit{irf\_offsets} \rangle$, $\langle \mathit{prot\_offsets} \rangle$)

$\langle \mathit{cellb1\_nxt} \rangle$ ::= *CellB1Nxt*($\langle \mathit{irf\_offsets} \rangle$, $\langle \mathit{prot\_offsets} \rangle$)

$\langle \mathit{irf\_offsets} \rangle$ ::= *IRFOffset*($\langle \mathit{num} \rangle$ {, $\langle \mathit{num} \rangle$ })

$\langle \mathit{prot\_offsets} \rangle$ ::= *ProtOffset*($\langle \mathit{num} \rangle$ {, $\langle \mathit{num} \rangle$ })

$\langle \mathit{cond} \rangle$ ::= ($\langle \mathit{reg\_seg} \rangle$, $\langle \mathit{num} \rangle$) | ($\langle \mathit{ins\_seg} \rangle$, $\langle \mathit{num} \rangle$)

$\langle \mathit{cmd} \rangle$ ::= $\langle \mathit{set\_cmd} \rangle$
| $\langle \mathit{mov\_cmd} \rangle$
| $\langle \mathit{lg\_cmd} \rangle$
| $\langle \mathit{eq\_cmd} \rangle$

$\langle \mathit{set\_cmd} \rangle$ ::= *Set*($\langle \mathit{reg\_seg} \rangle$, $\langle \mathit{num} \rangle$)

$\langle \mathit{mov\_cmd} \rangle$ ::= *Mov*($\langle \mathit{reg\_seg} \rangle$, $\langle \mathit{src\_reg} \rangle$)

$\langle \mathit{lg\_cmd} \rangle$ ::= *Lg*($\langle \mathit{reg\_seg} \rangle$, $\langle \mathit{src\_reg} \rangle$, $\langle \mathit{src\_reg} \rangle$)

$\langle \mathit{eq\_cmd} \rangle$ ::= *Eq*($\langle \mathit{reg\_seg} \rangle$, $\langle \mathit{src\_reg} \rangle$, $\langle \mathit{src\_reg} \rangle$)

$\langle \mathit{src\_reg} \rangle$ ::= (**IRF**, $\langle \mathit{reg\_offset} \rangle$, $\langle \mathit{reg\_size} \rangle$)
| $\langle \mathit{num} \rangle$

$\langle \mathit{reg\_seg} \rangle$ ::= (**IRF**, $\langle \mathit{reg\_offset} \rangle$, $\langle \mathit{seg\_size} \rangle$)

$\langle \mathit{ins\_seg} \rangle$ ::= ($\langle \mathit{ins\_name} \rangle$, $\langle \mathit{ins\_offset} \rangle$, $\langle \mathit{seg\_size} \rangle$)

⟨*reg_offset*⟩ ::= ⟨*num*⟩

⟨*reg_size*⟩ ::= ⟨*num*⟩

⟨*seg_size*⟩ ::= ⟨*num*⟩

⟨*ins_size*⟩ ::= ⟨*num*⟩

⟨*num*⟩ ::= *Integer* //integer constants, signed 32 bits

6.1.2 Translation to the AST of P3 Assembly

After the translation, the P3 AST example in the subsection 4.2 is translated into the P3 assembly looks like those in but in the format that can be derived from the syntax in the subsection 6.1.1.

6.2 Translation of P3 Assembly to the Configuration File

The information of a hardware configuration File is the binary format strictly equivalent to the P3 Assembly. As example, the binary format of the 1st and 5th items in the *cella_pb* table in Fig.7 is shown in Fig.9.

```
//l2
//cella_pb(407bit*32)
//hdr_id(7)+mask(24*8b)+value(24*8b)+sub_id(7)+nxt_id(7)+bypass(2:mainbypass(1)+subbypass(1))
//    1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
01_00 00 ff ff ff ff 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00_00 00 81 00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00_01_03_1  //eth+vlan+ipv4
01_00 00 ff ff 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00_00 00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00_03_03_1  //eth+ipv4
```

Figure 9: Part of information of the binary hardware configuration in the *cella_pb* table in Fig.7

7 Verification

We focus on the verification of the translation from a P3 AST to the P3 assembly in this section. Refer to Section 4.2 and Section 5.2 for the verification of the parser and the type checker respectively.

7.1 Verification of the translation from AST to Assembly

To verify the translation from a P3 AST to the P3 assembly, we define the formal semantics of the P3 AST language and the P3 assembly language in the subsections 7.2 and 7.3 respectively.

7.2 Semantics of the P3 AST

. . . . . .

7.2.1 Values and Memory model for Registers and Fields

The semantic values and the memory model used in the semantics definition can be inferred from the semantic environment in the subsection 7.2.2. To save the space, we omit to present them separately.

7.2.2 Semantic environment

Global	ge	$::=$	(γ, σ, δ)	divide global environment into three parts
	γ	$::=$	$(lr, cr, ps, ls, \iota, \rho)$	several basic settings of a *P3* specification
	σ	$::=$	$id \rightarrow val$	map a constant identifier to *val*
	δ	$::=$	$raid \rightarrow regacc(n,i,j,bv)$	map a register-access identifier to a segment $(i..j)$ of a register *IRF* sized n, with the binary value bv
	lr	$::=$	$lreglen(k)$	the *Lreglen* value set to k
	cr	$::=$	$creglen(k)$	the *Creglen* value set to k
	ps	$::=$	$pset(id, \cdots, id)$	the set of protocol identifiers
	ls	$::=$	$lset(id, \cdots, id)$	the set of layer identifiers
	ι	$::=$	$lid \rightarrow ldef$	map a layer identifier to a layer definition
	ρ	$::=$	$pid \rightarrow pdef$	map a protocol identifier to a protocol definition
Layer	le	$::=$	(ξ_ι, nh, len, bp)	divide layer local environment into four parts
	ξ_ι	$::=$	$id \rightarrow (k, (fid \rightarrow (n,bv)))$	map a protocol instance identifier to the length of the protocol (k) and a function that maps a field identifier to a binary value bv sized n
	nh	$::=$	$nextheader(pid)$	the *NextHeader* set to the protocol identified by *pid*
	len	$::=$	$length(k)$	the *Length* bound to an integer
	bp	$::=$	$bypass(k)$	the *Bypass* bound to an integer
Cell	ce	$::=$	$\delta_A \mid \delta_{B0} \mid \delta_{B1}$	divide cell local environment into three branches
	δ_A	$::=$	$raid \rightarrow regacc(n,i,j,bv)$	map a register-access identifier to a segment $(i..j)$ of a register *IRF* sized n, with the binary value bv
	δ_{B0}	$::=$	$raid \rightarrow regacc(n,i,j,bv)$	map a register-access identifier to a segment $(i..j)$ of a register *IRF* sized n, with the binary value bv
	δ_{B1}	$::=$	$raid \rightarrow regacc(n,i,j,bv)$	map a register-access identifier to a segment $(i..j)$ of a register *IRF* sized n, with the binary value bv
Protocol	ξ_ρ	$::=$	$fid \rightarrow fdacc(id, n,i,j,bv)$	map a field identifier to a segment $(i..j)$ of a protocol instance identified *id* sized no less than n, with the binary value bv
Identifier	*raid, lid, pid, fid*	$::=$	id	

Figure 10: Semantic Environments

The semantic environment maps variables to the values and memory for registers and fields, and has the form

$$\mathcal{E} ::= [\ x_1 : v_1,\ x_2 : v_2,\ ...,\ x_n : v_n\ ]$$

where $x_i \neq x_j$ for all i and j , satisfying $i \neq j$ and $(1 \leq i, j \leq n)$.

Figure 10 show all the semantic environments we use to define the semantics. In some cases, we use the subscript *id* to denote a particular local semantic environment specific to the context of a protocol or a layer identified by *id*.

7.2.3 Judgements

The judgements used in the semantics definition can be inferred from the semantic rules in the subsection 7.2.4. To save the space, we omit to present them separately.

7.2.4 Semantic rules

- Initialization of γ, opened at the beginning of the specification and not to be closed, where $\gamma = (lr,\ cr,\ ps,\ ls,\ \iota,\ \rho)$

  SLR–Initialization of lr :

$$\frac{\begin{array}{c}\gamma = (lr, cr, ps, ls, \iota, \rho) \qquad \vdash IntConst(k) \Rightarrow val(k) \\ lr = null \qquad lr' = lreglen(val(k)) \qquad \gamma' = (lr', cr, ps, ls, \iota, \rho)\end{array}}{\vdash (\gamma, Lreglen(IntConst(k))) \Rightarrow \gamma'}\ \mathrm{SLR}$$

  SCR–Initialization of cr :

$$\frac{\begin{array}{c}\gamma = (lr, cr, ps, ls, \iota, \rho) \qquad \vdash IntConst(k) \Rightarrow val(k) \\ cr = null \qquad cr' = lreglen(val(k)) \qquad \gamma' = (lr, cr', ps, ls, \iota, \rho)\end{array}}{\vdash (\gamma, Creglen(IntConst(k))) \Rightarrow \gamma'}\ \mathrm{SCR}$$

  SPS–Initialization of ps :

$$\frac{\begin{array}{c}\gamma = (lr, cr, ps, ls, \iota, \rho) \\ ps = null \qquad ps' = pset(id_1, \cdots, id_k) \qquad \gamma' = (lr, cr, ps', ls, \iota, \rho)\end{array}}{\vdash (\gamma, Pset(id_1, \cdots, id_k)) \Rightarrow \gamma'}\ \mathrm{SPS}$$

  SLS–Initialization of ls :

$$\frac{\begin{array}{c}\gamma = (lr, cr, ps, ls, \iota, \rho) \\ ls = null \qquad ls' = pset(id_1, \cdots, id_k) \qquad \gamma' = (lr, cr, ps, ls', \iota, \rho)\end{array}}{\vdash (\gamma, Lset(id_1, \cdots, id_k)) \Rightarrow \gamma'}\ \mathrm{SLS}$$

  SLA–Initialization of ι :

$$\frac{\begin{array}{c}\gamma = (lr, cr, ps, ls, \iota, \rho) \qquad ldef = get\_layer\_def(lvs, lrd, ld, las) \\ id \notin dom(\iota) \qquad \iota' = \iota \cup \{id : ldef\} \qquad \gamma' = (lr, cr, ps, ls, \iota', \rho)\end{array}}{\vdash (\gamma, LayerAction(id, lvs, lrd, ld, las)) \Rightarrow \gamma'} \text{SLA}$$

SPD–Initialization of ρ :

$$\frac{\begin{array}{c}\gamma = (lr, cr, ps, ls, \iota, \rho) \qquad pdef = get\_protocol\_def(p) \\ id \notin dom(\rho) \qquad \rho' = \rho \cup \{id : pdef\} \qquad \gamma' = (lr, cr, ps, ls, \iota, \rho')\end{array}}{\vdash (\gamma, ProtocolDecl(id, p)) \Rightarrow \gamma'} \text{SPD}$$

- Initialization of σ, opened at the beginning of the specification and not to be closed

$$\frac{\begin{array}{c}ge = (\gamma, \sigma, \delta) \\ \vdash c \Rightarrow v \qquad id \notin dom(\sigma) \qquad \sigma' = \sigma \cup \{id : v\} \qquad ge' = (\gamma, \sigma', \delta)\end{array}}{\vdash (ge, ConstDcl(id, c)) \Rightarrow ge'} \text{(SIC-1)}$$

$$\frac{val(i)\ is\ a\ signed\ integer\ up\ to\ 32\ bits}{\vdash IntConst(i) \Rightarrow val(i)} \text{(SIC-2)}$$

$$\frac{val(i)\ is\ a\ number\ i\ with\ \ hexadecimal\ digits}{\vdash HexConst(i) \Rightarrow val(i)} \text{(SIC-3)}$$

$$\frac{val(bs)\ is\ the\ binary\ bit\ string\ of\ bs}{\vdash BitSConst(bs) \Rightarrow val(bs)} \text{(IC-4)}$$

- Initialization of δ, initialized at the beginning of the specification and changed each time at the leaving of a layer context (Rule SIR-3).

$$\frac{\begin{array}{c} ge = (\gamma, \sigma, \delta) \qquad ge \vdash e_1 \Rightarrow n_1 \qquad ge \vdash e_2 \Rightarrow n_2 \\ \gamma = (\mathit{lreglen}(n), cr, ps, ls, \iota, \rho) \qquad 0 \leq n_2 \leq n_1 < n \qquad id \notin \mathit{dom}(\delta) \\ \forall id' \in \mathit{dom}(\delta).(\delta \vdash id' \Rightarrow \mathit{regacc}(n, n_1', n_2', \mathit{base\_layer\_bv}_{id'}) \rightarrow n_1' < n_2 \vee n_1 < n_2') \\ \delta' = \delta \cup \{id : \mathit{regacc}(n, n_1, n_2, \mathit{base\_layer\_bv}_{id})\} \qquad ge' = (\gamma, \sigma, \delta') \end{array}}{\vdash (ge, \mathit{IRF}(id, e_1, e_2)) \Rightarrow ge'} \text{(SIR-1)}$$

$$\frac{\begin{array}{c} ge = (\gamma, \sigma, \delta) \qquad ge \vdash e \Rightarrow k \\ \gamma = (\mathit{lreglen}(n), cr, ps, ls, \iota, \rho) \qquad 0 \leq k < n \qquad id \notin \mathit{dom}(\delta) \\ \forall id' \in \mathit{dom}(\delta).(\delta \vdash id' \Rightarrow \mathit{regacc}(n, n_1', n_2', \mathit{base\_layer\_bv}_{id'}) \rightarrow n_1' < k \vee k < n_2') \\ \delta' = \delta \cup \{id : \mathit{regacc}(n, k, k, \mathit{base\_layer\_bv}_{id})\} \qquad ge' = (\gamma, \sigma, \delta') \end{array}}{\vdash (ge, \mathit{IRF}(id, e)) \Rightarrow ge'} \text{(SIR-2)}$$

$$\frac{\begin{array}{c} ge = (\gamma, \sigma, \delta) \qquad \gamma = (\mathit{lreglen}(n), \mathit{creglen}(k), ps, ls, \iota, \rho) \\ n = 3 * k \qquad le = (\xi_\iota, \mathit{nextheader}(pid), \mathit{length}(i), \mathit{bypass}(j)) \\ \delta' = \{id : \mathit{regacc}(n, 2*k+n_1, 2*k+n_2, bva) \mid id : \mathit{regacc}(k, n_1, n_2, bva) \in \delta_A\} \\ \cup \{id : \mathit{regacc}(n, k+n_1, k+n_2, bvb0) \mid id : \mathit{regacc}(k, n_1, n_2, bvb0) \in \delta_{B0}\} \\ \cup \{id : \mathit{regacc}(n, n_1, n_2, bvb1) \mid id : \mathit{regacc}(k, n_1, n_2, bvb1) \in \delta_{B1}\} \\ ge' = (\gamma, \sigma, \delta') \\ le' = (\emptyset, \mathit{nextheader}(null), \mathit{length}(null), \mathit{bypass}(null)) \qquad \delta_A' = null \end{array}}{\vdash (ge, le, \delta_A, \text{``}\mathit{layer\text{-}switch}\text{''}) \Rightarrow (ge', le', \delta_A')} \text{(SIR-3)}$$

- Initialization of le, opened at the beginning and closed at the end of a LayerAction specification

$$\frac{\begin{array}{c} ge = (\gamma, \sigma, \delta) \qquad \gamma = (lr, cr, ps, ls, \iota, \rho) \\ le = (\xi_\iota, nh, len, bp) \qquad \rho \vdash pid \Rightarrow ((\mathit{fid}_1 : n_1, \cdots, \mathit{fid}_m : n_m), \mathit{pstmts}) \\ \forall i : 1 \leq i \leq k.\ id_i \notin \mathit{dom}(\xi_\iota) \\ \xi_\iota' = \xi_\iota \cup \{id_i : (n_1 + \cdots + n_m, \mathit{pins}_i) \mid 1 \leq i \leq k \wedge \mathit{pins}_i = ((\mathit{fid}_1, (n_1, bv_1^i)), \cdots, (\mathit{fid}_m, (n_m, bv_m^i)))\}, \\ \textit{where all bv's are the input from the hardware} \\ le' = (\xi_\iota', \mathit{nextheader}(null), \mathit{length}(null), \mathit{bypass}(null)) \end{array}}{ge \vdash (le, \mathit{ProtocolDef}(pid, (id_1, \cdots, id_k))) \Rightarrow le'} \text{(SIL}$$

- Initialization of δ_A at the CellA Registers specification, opened at the beginning of a Cell A specification, and closed at the leaving of the layer context

$$\frac{\begin{array}{c} ge = (\gamma, \sigma, \delta) \qquad \gamma = (\mathit{lreglen}(n), \mathit{creglen}(k), \mathit{ps}, \mathit{ls}, \iota, \rho) \\ n = 3 * k \qquad ge, le, \delta_A \vdash e_1 \Rightarrow n_1 \\ ge, le, \delta_A \vdash e_2 \Rightarrow n_2 \qquad 0 \leq n_2 \leq n_1 < k \qquad id \notin dom(\delta_A) \cup dom(\delta) \\ \forall id' \in dom(\delta_A).\ (\delta_A \vdash id' \Rightarrow regacc(k, n_1', n_2', bv) \rightarrow n_1' < n_2 \vee n_1 < n_2') \\ \forall id' \in dom(\delta).\ (\delta \vdash id' \Rightarrow regacc(n, n_1', n_2', bv) \rightarrow n_1' < 2 * k + n_2 \vee 2 * k + n_1 < n_2') \\ \delta_A' = \delta_A \cup \{id : regacc(k, n_1, n_2, null)\} \end{array}}{ge, le \vdash (\delta_A, IRF(id, e_1, e_2)) \Rightarrow \delta_A'} \text{(SILA-1)}$$

$$\frac{\begin{array}{c} ge = (\gamma, \sigma, \delta) \qquad \gamma = (\mathit{lreglen}(n), \mathit{creglen}(k), \mathit{ps}, \mathit{ls}, \iota, \rho) \qquad n = 3 * k \\ ge, le, \delta_A \vdash e \Rightarrow m \qquad 0 \leq m < k \qquad id \notin dom(\delta_A) \cup dom(\delta) \\ \forall id' \in dom(\delta_A).\ (\delta_A \vdash id' \Rightarrow regacc(k, n_1', n_2', bv) \rightarrow n_1' < m \vee m < n_2') \\ \forall id' \in dom(\delta.\ (\delta \vdash id' \Rightarrow regacc(n, n_1', n_2', bv) \rightarrow n_1' < 2 * k + m \vee 2 * k + m < n_2') \\ \delta_A' = \delta_A \cup \{id : regacc(k, m, m, null)\} \end{array}}{ge, le \vdash (\delta_A, IRF(id, e)) \Rightarrow \delta_A'} \text{(SILA-2)}$$

- Initialization of δ_{B0} at the CellB0 Registers specification, opened at the beginning of a Cell B0 specification, and closed at the leaving of the layer context

$$\frac{\begin{array}{c} ge = (\gamma, \sigma, \delta) \qquad \gamma = (\mathit{lreglen}(n), \mathit{creglen}(k), \mathit{ps}, \mathit{ls}, \iota, \rho) \\ n = 3 * k \qquad ge, le, \delta_{B0} \vdash e_1 \Rightarrow n_1 \\ ge, le, \delta_{B0} \vdash e_2 \Rightarrow n_2 \qquad 0 \leq n_2 \leq n_1 < k \qquad id \notin dom(\delta_{B0}) \cup dom(\delta) \\ \forall id' \in dom(\delta_{B0}).\ (\delta_{B0} \vdash id' \Rightarrow regacc(k, n_1', n_2', bv) \rightarrow n_1' < n_2 \vee n_1 < n_2') \\ \forall id' \in dom(\delta).\ (\delta \vdash id' \Rightarrow regacc(n, n_1', n_2', bv) \rightarrow n_1' < k + n_2 \vee k + n_1 < n_2') \\ \delta_{B0}' = \delta_{B0} \cup \{id : regacc(k, n_1, n_2, null)\} \end{array}}{ge, le \vdash (\delta_{B0}, IRF(id, e_1, e_2)) \Rightarrow \delta_{B0}'} \text{(SILB0-1)}$$

$$\frac{\begin{array}{c} ge = (\gamma, \sigma, \delta) \qquad \gamma = (\mathit{lreglen}(n), \mathit{creglen}(k), \mathit{ps}, \mathit{ls}, \iota, \rho) \qquad n = 3 * k \\ ge, le, \delta_{B0} \vdash e \Rightarrow n \qquad 0 \leq m < k \qquad id \notin dom(\delta_{B0}) \cup dom(\delta) \\ \forall id' \in dom(\delta_{B0}).\ (\delta_{B0} \vdash id' \Rightarrow regacc(k, n_1', n_2', bv) \rightarrow n_1' < m \vee m < n_2') \\ \forall id' \in dom(\delta.\ (\delta \vdash id' \Rightarrow regacc(n, n_1', n_2', bv) \rightarrow n_1' < k + m \vee k + m < n_2') \\ \delta_{B0}' = \delta_{B0} \cup \{id : regacc(k, m, m, null)\} \end{array}}{ge, le \vdash (\delta_{B0}, IRF(id, e)) \Rightarrow \delta_{B0}'} \text{(SILB0-2)}$$

- Initialization of δ_{B1} at the CellB0 Registers specification, opened at the beginning of a Cell B1 specification, and closed at the leaving of the layer context

$$\frac{\begin{array}{c} ge = (\gamma, \sigma, \delta) \qquad \gamma = (lreglen(n), creglen(k), ps, ls, \iota, \rho) \\ n = 3 * k \qquad ge, le, \delta_{B1} \vdash e_1 \Rightarrow n_1 \\ ge, le, \delta_{B1} \vdash e_2 \Rightarrow n_2 \qquad 0 \leq n_2 \leq n_1 < k \qquad id \notin dom(\delta_{B1}) \cup dom(\delta) \\ \forall id' \in dom(\delta_{B1}).(\delta_{B1} \vdash id' \Rightarrow regacc(k, n_1', n_2', bv) \rightarrow n_1' < n_2 \vee n_1 < n_2') \\ \forall id' \in dom(\delta).\ (\delta \vdash id' \Rightarrow regacc(n, n_1', n_2', bv) \rightarrow n_1' < n_2 \vee n_1 < n_2') \\ \delta_{B1}' = \delta_{B1} \cup \{id : regacc(k, n_1, n_2, null)\} \end{array}}{ge, le \vdash (\delta_{B1}, IRF(id, e_1, e_2)) \Rightarrow \delta_{B1}'} \text{(SILB1-1)}$$

$$\frac{\begin{array}{c} ge = (\gamma, \sigma, \delta) \qquad \gamma = (lreglen(n), creglen(k), ps, ls, \iota, \rho) \qquad n = 3 * k \\ ge, le, \delta_{B0} \vdash e \Rightarrow m \qquad 0 \leq m < k \qquad id \notin dom(\delta_{B1}) \cup dom(\delta) \\ \forall id' \in dom(\delta_{B1}).\ (\delta_{B1} \vdash id' \Rightarrow regacc(k, n_1', n_2', bv) \rightarrow n_1' < m \vee m < n_2') \\ \forall id' \in dom(\delta.\ (\delta \vdash id' \Rightarrow regacc(n, n_1', n_2', bv) \rightarrow n_1' < m \vee m < n_2') \\ \delta_{B1}' = \delta_{B1} \cup \{id : regacc(k, m, m, null)\} \end{array}}{ge, le \vdash (\delta_{B1}, IRF(id, e)) \Rightarrow \delta_{B1}'} \text{(SILB1-2)}$$

- Initialization of ξ_ρ, opened at each time of the instantialization of a Protocol specification and closed at the end of that instantialization.

$$\frac{\begin{array}{c} le = (\xi_\iota, nh, len, bp) \qquad \xi_\rho = \emptyset \\ flds{+}{+}ofld = ((fld_1 : c_1), \cdots, (fld_k : c_k)),\ where\ c_k\ to\ be\ a\ number\ or\ a\ (null) \\ There\ exists\ an\ unique\ protocol\ instance\ identified\ by\ id,\ such\ that\ (id : (len', pins)) \in \xi_\iota, \\ where\ pins = ((fid_1, (n_1, bv_1)), \cdots, (fid_k, (n_k, bv_k))) \\ n = n_1 + n_2 + \cdots + n_k \\ \xi_\rho' = \{fld_i : (id, n, n_1 + \cdots + n_{i-1}, n_1 + \cdots + n_i - 1), bv_i) \mid 1 \leq i \leq k\} \end{array}}{ge, le, \delta_A \vdash (\xi_\rho, ProtocolDecl(pid, Protocol((Fields(flds), OptionFields(ofld)), pstmts))) \Rightarrow \xi_\rho'} \text{(SIP-1)}$$

- Expressions

$$\frac{ge = (\gamma, \sigma, \delta) \qquad \sigma \vdash c \Rightarrow v \qquad \delta_C\ is\ \delta_A, \delta_{B0}\ or\ \delta_{B1}}{ge, le, \delta_C \vdash Econst(c) \Rightarrow v} \text{SCE-1}$$

$$\frac{ge = (\gamma, \sigma, \delta) \qquad \sigma \vdash c \Rightarrow v}{ge, le, \delta_A, \xi_\rho \vdash Econst(c) \Rightarrow v} \text{SCE-2} \qquad \frac{ge = (\gamma, \sigma, \delta) \qquad \sigma \vdash c \Rightarrow v}{ge \vdash Econst(c) \Rightarrow v} \text{SCE-3}$$

$$\frac{ge, le, \delta_C \vdash e \Rightarrow v \qquad v' = trans\_to\_int(v) \qquad \delta_C \ is \ \delta_A, \delta_{B0} \ or \ \delta_{B1}}{ge, le, \delta_C \vdash Eunop(Oint, e) \Rightarrow v'} \text{ SOINT-1}$$

$$\frac{ge, le, \delta_A, \xi_\rho \vdash e \Rightarrow v \qquad v' = trans\_to\_int(v)}{ge, le, \delta_A, \xi_\rho \vdash Eunop(Oint, e) \Rightarrow v'} \text{ SOINT-2}$$

$$\frac{ge, le, \delta_C \vdash e \Rightarrow v \qquad v' = not(v) \qquad \delta_C \ is \ \delta_A, \delta_{B0} \ or \ \delta_{B1}}{ge, le, \delta_C \vdash Eunop(Onot, e) \Rightarrow v'} \text{ SONOT-1}$$

$$\frac{ge, le, \delta_A, \xi_\rho \vdash e \Rightarrow v \qquad v' = not(v)}{ge, le, \delta_A, \xi_\rho \vdash Eunop(Onot, e) \Rightarrow v'} \text{ SONOT-2}$$

$$\frac{\begin{array}{c} ge, le, \delta_C \vdash e :\Rightarrow bs \\ bs' = bit\_wise\_negation(bs) \qquad \delta_C \ is \ \delta_A, \delta_{B0} \ or \ \delta_{B1} \end{array}}{ge, le, \delta_C \vdash Eunop(Oneg, e) \Rightarrow bs'} \text{ SONEG-1}$$

$$\frac{ge, le, \delta_A, \xi_\rho \vdash e :\Rightarrow bs \qquad bs' = bit\_wise\_negation(bs)}{ge, le, \delta_A, \xi_\rho \vdash Eunop(Oneg, e) \Rightarrow bs'} \text{ SONEG-2}$$

$$\frac{\begin{array}{c} ge, le, \delta_C \vdash e_1 \Rightarrow v_1 \\ ge, le, \delta_C \vdash e_2 \Rightarrow v_2 \qquad binop \in \{Oadd, Osub, Omul, Odivint, Omod\} \\ v = do\_binop(binop, trans\_to\_int(v_1), trans\_to\_int(v_2)) \\ \delta_C \ is \ \delta_A, \delta_{B0} \ or \ \delta_{B1} \end{array}}{ge, le, \delta_C \vdash Ebinop(binop, e_1, e_2) \Rightarrow v} \text{ SBOPA-1}$$

$$\frac{\begin{array}{c} ge, le, \delta_A, \xi_\rho \vdash e_1 \Rightarrow v_1 \\ ge, le, \delta_A, \xi_\rho \vdash e_2 \Rightarrow v_2 \qquad binop \in \{Oadd, Osub, Omul, Odivint, Omod\} \\ v = do\_binop(binop, trans\_to\_int(v_1), trans\_to\_int(v_2)) \end{array}}{ge, le, \delta_A, \xi_\rho \vdash Ebinop(binop, e_1, e_2) \Rightarrow v} \text{ SBOPA-2}$$

$$\frac{\begin{array}{c} ge \vdash e_1 \Rightarrow v_1 \\ ge \vdash e_2 \Rightarrow v_2 \qquad binop \in \{Oadd, Osub, Omul, Odivint, Omod\} \\ v = do\_binop(binop, trans\_to\_int(v_1), trans\_to\_int(v_2)) \end{array}}{ge \vdash Ebinop(binop, e_1, e_2) \Rightarrow v} \text{ SBOPA-3}$$

$$\frac{ge, le, \delta_C \vdash e_1 \Rightarrow v_1 \qquad ge, le, \delta_C \vdash e_2 \Rightarrow v_2 \qquad binop \in \{Oand, Oor\} \quad v = do\_logic\_binop(binop, v_1, v_2) \qquad \delta_C \ is \ \delta_A, \delta_{B0} \ or \ \delta_{B1}}{ge, le, \delta_C \vdash Ebinop(binop, e_1, e_2) \Rightarrow v} \ \textsc{SBopL-1}$$

$$\frac{ge, le, \delta_A, \xi_\rho \vdash e_1 \Rightarrow v_1 \qquad ge, le, \delta_A, \xi_\rho \vdash e_2 \Rightarrow v_2 \quad binop \in \{Oand, Oor\} \qquad v = do\_logic\_binop(binop, v_1, v_2)}{ge, le, \delta_A, \xi_\rho \vdash Ebinop(binop, e_1, e_2) \Rightarrow v} \ \textsc{SBopL-2}$$

$$\frac{ge, le, \delta_C \vdash e_1 \Rightarrow bs_1 \quad ge, le, \delta_C \vdash e_2 \Rightarrow bs_2 \qquad binop \in \{Oband, Obor, Obeor\} \quad bs = bit\_wise\_operation(binop, bs_1, bs_2) \qquad \delta_C \ is \ \delta_A, \delta_{B0} \ or \ \delta_{B1}}{ge, le, \delta_C \vdash Ebinop(binop, e_1, e_2) \Rightarrow bs} \ \textsc{SBopB-1}$$

$$\frac{ge, le, \delta_A, \xi_\rho \vdash e_1 \Rightarrow bs_1 \quad ge, le, \delta_A, \xi_\rho \vdash e_2 \Rightarrow bs_2 \qquad binop \in \{Oband, Obor, Obeor\} \quad bs = bit\_wise\_operation(binop, bs_1, bs_2)}{ge, le, \delta_A, \xi_\rho \vdash Ebinop(binop, e_1, e_2) \Rightarrow bs} \ \textsc{SBopB-2}$$

$$\frac{ge, le, \delta_C \vdash e_1 \Rightarrow v_1 \quad ge, le, \delta_C \vdash e_2 \Rightarrow v_2 \qquad binop \in \{Oeq, One, Olt, Ogt, Ole, Oge\} \quad v = do\_relation\_binop(binop, v_1, v_2) \qquad \delta_C \ is \ \delta_A, \delta_{B0} \ or \ \delta_{B1}}{ge, le, \delta_C \vdash Ebinop(binop, e_1, e_2) \Rightarrow v} \ \textsc{SBopR-1}$$

$$\frac{ge, le, \delta_A, \xi_\rho \vdash e_1 \Rightarrow v_1 \quad ge, le, \delta_A, \xi_\rho \vdash e_2 \Rightarrow v_2 \qquad binop \in \{Oeq, One, Olt, Ogt, Ole, Oge\} \quad v = do\_relation\_binop(binop, v_1, v_2)}{ge, le, \delta_A, \xi_\rho \vdash Ebinop(binop, e_1, e_2) \Rightarrow v} \ \textsc{SBopR-2}$$

$$\frac{ge, le, \delta_C \vdash e_1 \Rightarrow bs_1 \qquad ge, le, \delta_C \vdash e_2 \Rightarrow v_2 \qquad binop \in \{Osl, Osr\} \quad bs = do\_shift\_binop(binop, bs_1, v_2) \qquad \delta_C \ is \ \delta_A, \delta_{B0} \ or \ \delta_{B1}}{ge, le, \delta_C \vdash Ebinop(binop, e_1, e_2) \Rightarrow bs} \ \textsc{SBopS-1}$$

$$\frac{ge, le, \delta_A, \xi_\rho \vdash e_1 \Rightarrow bs_1 \qquad ge, le, \delta_A, \xi_\rho \vdash e_2 \Rightarrow v_2 \quad binop \in \{Osl, Osr\} \qquad bs = do\_shift\_binop(binop, bs_1, v_2)}{ge, le, \delta_A, \xi_\rho \vdash Ebinop(binop, e_1, e_2) \Rightarrow bs} \ \textsc{SBopS-2}$$

$$\frac{\begin{array}{c} ge, le, \delta_C \vdash e_1 \Rightarrow bs_1 \\ ge, le, \delta_C \vdash e_2 \Rightarrow bs_2 \qquad bs = cat(bs_1, bs_2) \qquad \delta_C \ is \ \delta_A, \delta_{B0} \ or \ \delta_{B1} \end{array}}{ge, le, \delta_C \vdash Ebinop(Obc, e_1, e_2) \Rightarrow bs} \text{ SBopC-1}$$

$$\frac{\begin{array}{c} ge, le, \delta_A, \xi_\rho \vdash e_1 \Rightarrow bs_1 \\ ge, le, \delta_A, \xi_\rho \vdash e_2 \Rightarrow bs_2 \qquad bs = cat(bs_1, bs_2) \end{array}}{ge, le, \delta_A, \xi_\rho \vdash Ebinop(Obc, e_1, e_2) \Rightarrow bs} \text{ SBopC-1'}$$

$$\frac{\begin{array}{c} ge, le, \delta_C \vdash e_1 \Rightarrow bs_1 \qquad ge, le, \delta_C \vdash e_2 \Rightarrow bs_2 \\ bs = hex\_cat(bs_1, bs_2) \qquad \delta_C \ is \ \delta_A, \delta_{B0} \ or \ \delta_{B1} \end{array}}{ge, le, \delta_C \vdash Ebinop(Obc, e_1, e_2) \Rightarrow bs} \text{ SBopC-2}$$

$$\frac{\begin{array}{c} ge, le, \delta_A, \xi_\rho \vdash e_1 \Rightarrow bs_1 \\ ge, le, \delta_A, \xi_\rho \vdash e_2 \Rightarrow bs_2 \qquad bs = hex\_cat(bs_1, bs_2) \end{array}}{ge, le, \delta_A, \xi_\rho \vdash Ebinop(Obc, e_1, e_2) \Rightarrow bs} \text{ SBopC-2'}$$

$$\frac{\begin{array}{c} ge, le, \delta_C \vdash e_1 : regacc(k, n_1, n_2, bs_1) \\ |bs_1| = n_1 - n_2 + 1 \qquad ge, le, \delta_C \vdash e_2 : regacc(k, m_1, m_2, bs_2) \\ |bs_2| = m_1 - m_2 + 1 \qquad n_2 = m_1 + 1 \qquad 0 \leq m_2 \leq m_1 < n_2 \leq n_1 < k \\ bs = cat(bs_1, bs_2) \qquad \delta_C \ is \ \delta_A, \delta_{B0} \ or \ \delta_{B1} \end{array}}{ge, le, \delta_C \vdash Ebinop(Obc, e_1, e_2) : regacc(k, n_1, m_2, bs)} \text{ SBopC-3}$$

$$\frac{\begin{array}{c} ge, le, \delta_A, \xi_\rho \vdash e_1 : regacc(k, n_1, n_2, bs_1) \qquad |bs_1| = n_1 - n_2 + 1 \\ ge, le, \delta_A, \xi_\rho \vdash e_2 : regacc(k, m_1, m_2, bs_2) \qquad |bs_2| = m_1 - m_2 + 1 \\ n_2 = m_1 + 1 \qquad 0 \leq m_2 \leq m_1 < n_2 \leq n_1 < k \qquad bs = cat(bs_1, bs_2) \end{array}}{ge, le, \delta_A, \xi_\rho \vdash Ebinop(Obc, e_1, e_2) : regacc(k, n_1, m_2, bs)} \text{ SBopC-3'}$$

$$\frac{\begin{array}{c} ge, le, \delta_C \vdash e_1 : fdacc(id, k, n_1, n_2, bs_1) \\ |bs_1| = n_2 - n_1 + 1 \qquad ge, le, \delta_C \vdash e_2 : fdacc(id, k, m_1, m_2, bs_2) \\ |bs_2| = m_2 - m_1 + 1 \qquad m_1 = n_2 + 1 \qquad 0 \leq n_1 \leq n_2 < m_1 \leq m_2 < k \\ bs = cat(bs_1, bs_2) \qquad \delta_C \ is \ \delta_A, \delta_{B0} \ or \ \delta_{B1} \end{array}}{ge, le, \delta_C \vdash Ebinop(Obc, e_1, e_2) : fdacc(id, k, n_1, m_2, bs)} \text{ SBopC-4}$$

$$\frac{\begin{array}{c} ge, le, \delta_A, \xi_\rho \vdash e_1 : fdacc(id, k, n_1, n_2, bs_1) \qquad |bs_1| = n_2 - n_1 + 1 \\ ge, le, \delta_A, \xi_\rho \vdash e_2 : fdacc(id, k, m_1, m_2, bs_2) \qquad |bs_2| = m_2 - m_1 + 1 \\ m_1 = n_2 + 1 \qquad 0 \leq n_1 \leq n_2 < m_1 \leq m_2 < k \qquad bs = cat(bs_1, bs_2) \end{array}}{ge, le, \delta_A, \xi_\rho \vdash Ebinop(Obc, e_1, e_2) : fdacc(id, k, n_1, m_2, bs)} \text{ SBopC-4'}$$

$$\frac{\begin{array}{c}ge, le, \delta_C \vdash e_1 \Rightarrow v \qquad ge, le, \delta_C \vdash e_2 \Rightarrow n \\ hn = trans\_to\_hex\_number(v, n) \qquad \delta_C\ is\ \delta_A, \delta_{B0}\ or\ \delta_{B1}\end{array}}{ge, le, \delta_C \vdash Ebinop(Ohexes, e_1, e_2) \Rightarrow hn}\ \textsc{SBopH-1}$$

$$\frac{\begin{array}{c}ge, le, \delta_A, \xi_\rho \vdash e_1 \Rightarrow v \\ ge, le, \delta_A, \xi_\rho \vdash e_2 \Rightarrow n \qquad hn = trans\_to\_hex\_number(v, n)\end{array}}{ge, le, \delta_A, \xi_\rho \vdash Ebinop(Ohexes, e_1, e_2) \Rightarrow hn}\ \textsc{BopH-2}$$

$$\frac{\begin{array}{c}ge, le, \delta_C \vdash e_1 \Rightarrow v \qquad ge, le, \delta_C \vdash e_2 \Rightarrow n \\ bn = trans\_to\_binary\_number(v, n) \qquad \delta_C\ is\ \delta_A, \delta_{B0}\ or\ \delta_{B1}\end{array}}{ge, le, \delta_C \vdash Ebinop(Ohexes, e_1, e_2) \Rightarrow bn}\ \textsc{SBopBT-1}$$

$$\frac{\begin{array}{c}ge, le, \delta_A, \xi_\rho \vdash e_1 \Rightarrow v \\ ge, le, \delta_A, \xi_\rho \vdash e_2 \Rightarrow n \qquad bn = trans\_to\_binary\_number(v, n)\end{array}}{ge, le, \delta_A, \xi_\rho \vdash Ebinop(Ohexes, e_1, e_2) \Rightarrow bn}\ \textsc{SBopBT-2}$$

$$\frac{\begin{array}{c}ge, le, \delta_C \vdash id \Rightarrow (pid, pins) \\ pins = ((fid_1, (n_1, bv_1)), \cdots, (fid_k, (n_k, bv_k))) \\ n = n_1 + n_2 + \cdots + n_k \qquad \exists i.\ fid = fid_i \qquad \delta_C\ is\ \delta_A, \delta_{B0}\ or\ \delta_{B1}\end{array}}{ge, le, \delta_C \vdash Efield(id, fid) \Rightarrow fdacc(id, n, n_1 + \cdots + n_{i-1}, n_1 + \cdots + n_i - 1, bv_i)}\ \textsc{SEfield}$$

$$\frac{\begin{array}{c}ge, le, \delta_C \vdash e_1 \Rightarrow regacc(n, n_1, n_2, bv) \\ ge, le, \delta_C \vdash e_2 \Rightarrow n' \qquad 0 \le n_2 \le n_1 < n \qquad 0 \le n' \le n_1 - n_2 \\ b = get\_binary\_bit(bv, n') \qquad \delta_C\ is\ \delta_A, \delta_{B0}\ or\ \delta_{B1}\end{array}}{ge, le, \delta_C \vdash EFieldBit(e_1, e_2) \Rightarrow regacc(n, n_2 + n', n_2 + n', b)}\ \text{SFB-1}$$

$$\frac{\begin{array}{c}ge, le, \delta_A, \xi_\rho \vdash e_1 \Rightarrow regacc(n, n_1, n_2, bv) \\ ge, le, \delta_A, \xi_\rho \vdash e_2 \Rightarrow n' \qquad 0 \le n_2 \le n_1 < n \\ 0 \le n' \le n_1 - n_2 \qquad b = get\_binary\_bit(bv, n')\end{array}}{ge, le, \delta_A, \xi_\rho \vdash EFieldBit(e_1, e_2) \Rightarrow regacc(n, n_2 + n', n_2 + n', b)}\ \text{SFB-1'}$$

$$\frac{\begin{array}{c} ge, le, \delta_C \vdash e_1 \Rightarrow \mathit{fdacc}(id, n, n_1, n_2, bv) \\ ge, le, \delta_C \vdash e_2 \Rightarrow n' \qquad 0 \leq n_1 \leq n_2 < n \qquad 0 \leq n' \leq n_2 - n_1 \\ b = \mathit{get\_binary\_bit}(bv, n') \qquad \delta_C \ \mathit{is} \ \delta_A, \delta_{B0} \ \mathit{or} \ \delta_{B1} \end{array}}{ge, le, \delta_C \vdash \mathit{EFieldBit}(e_1, e_2) : \mathit{fdacc}(id, n, n_1 + n', n_1 + n', b)} \text{ SFB-2}$$

$$\frac{\begin{array}{c} ge, le, \delta_A, \xi_\rho \vdash e_1 \Rightarrow \mathit{fdacc}(id, n, n_1, n_2, bv) \\ ge, le, \delta_A, \xi_\rho \vdash e_2 \Rightarrow n' \qquad 0 \leq n_1 \leq n_2 < n \\ 0 \leq n' \leq n_2 - n_1 \qquad b = \mathit{get\_binary\_bit}(bv, n') \end{array}}{ge, le, \delta_A, \xi_\rho \vdash \mathit{EFieldBit}(e_1, e_2) : \mathit{fdacc}(id, n, n_1 + n', n_1 + n', b)} \text{ SFB-2'}$$

$$\frac{\begin{array}{c} ge, le, \delta_C \vdash e_1 \Rightarrow \mathit{regacc}(n, n_1, n_2, bv) \qquad ge, le, \delta_C \vdash e_2 \Rightarrow n' \\ ge, le, \delta_C \vdash e_3 \Rightarrow n'' \qquad 0 \leq n_2 \leq n_1 < n \qquad 0 \leq n'' \leq n' \leq n_1 - n_2 \\ bv' = \mathit{get\_binary\_bits}(bv, n', n'') \qquad \delta_C \ \mathit{is} \ \delta_A, \delta_{B0} \ \mathit{or} \ \delta_{B1} \end{array}}{ge, le, \delta_C \vdash \mathit{EFieldSection}(e_1, e_2, e_3) \Rightarrow \mathit{regacc}(n, n_2 + n', n_2 + n'', bv')} \text{ SFS-1}$$

$$\frac{\begin{array}{c} ge, le, \delta_A, \xi_\rho \vdash e_1 \Rightarrow \mathit{regacc}(n, n_1, n_2, bv) \\ ge, le, \delta_C \vdash e_2 \Rightarrow n' \qquad ge, le, \delta_C \vdash e_3 \Rightarrow n'' \qquad 0 \leq n_2 \leq n_1 < n \\ 0 \leq n'' \leq n' \leq n_1 - n_2 bv' = \mathit{get\_binary\_bits}(bv, n', n'') \end{array}}{ge, le, \delta_A, \xi_\rho \vdash \mathit{EFieldSection}(e_1, e_2, e_3) \Rightarrow \mathit{regacc}(n, n_2 + n', n_2 + n'', bv')} \text{ SFS-1'}$$

$$\frac{\begin{array}{c} ge, le, \delta_C \vdash e_1 \Rightarrow \mathit{fdacc}(id, n, n_1, n_2, bv) \qquad ge, le, \delta_C \vdash e_2 : (Int, n') \\ ge, le, \delta_C \vdash e_3 : (Int, n'') \qquad 0 \leq n_1 \leq n_2 < n \qquad 0 \leq n' \leq n_2 - n_1 \\ bv' = \mathit{get\_binary\_bits}(bv, n', n'') \qquad \delta_C \ \mathit{is} \ \delta_A, \delta_{B0} \ \mathit{or} \ \delta_{B1} \end{array}}{ge, le, \delta_C \vdash \mathit{EFieldSection}(e_1, e_2, e_3) : \mathit{fdacc}(id, n, n_1 + n'', n_1 + n', bv')} \text{ SFS-2}$$

$$\frac{\begin{array}{c} ge, le, \delta_A, \xi_\rho \vdash e_1 \Rightarrow \mathit{fdacc}(id, n, n_1, n_2, bv) \\ ge, le, \delta_A, \xi_\rho \vdash e_2 : (Int, n') \\ ge, le, \delta_A, \xi_\rho \vdash e_3 : (Int, n'') \qquad 0 \leq n_1 \leq n_2 < n \\ 0 \leq n' \leq n_2 - n_1 \qquad bv' = \mathit{get\_binary\_bits}(bv, n', n'') \end{array}}{ge, le, \delta_A, \xi_\rho \vdash \mathit{EFieldSection}(e_1, e_2, e_3) : \mathit{fdacc}(id, n, n_1 + n'', n_1 + n', bv')} \text{ SFS-2'}$$

$$\frac{le = (\xi_\iota, nh, len, bp) \qquad \xi_\iota \vdash id \Rightarrow (length(n), pins)}{ge, le \vdash ProtLen(id) \Rightarrow n} \ (\text{SPL}\textsc{en})$$

- Instructions

$$\frac{\begin{array}{c} ge, le, \delta_C \vdash e \Rightarrow v \\ ge, le, \delta_C \vdash ra \Rightarrow regacc(k, i, j, bv) \qquad bv' = trans\_to\_bits(v, n) \\ n = i - j + 1 \qquad \delta'_C = \delta_C \mid_{ra \Rightarrow regacc(k,i,j,bv')} \qquad \delta_C \ is \ \delta_A, \delta_{B0} \ or \ \delta_{B1} \end{array}}{ge, le \vdash (\delta_C, Set(ra, e)) \Rightarrow \delta'_C} \ \textsc{SSet-1}$$

$$\frac{\begin{array}{c} ge, le, \delta_A, \xi_\rho \vdash e \Rightarrow v \\ ge, le, \delta_A, \xi_\rho \vdash ra \Rightarrow regacc(k, i, j, bv) \qquad bv' = trans\_to\_bits(v, n) \\ n = i - j + 1 \qquad \delta'_A = \delta_A \mid_{ra \Rightarrow regacc(k,i,j,bv')} \end{array}}{ge, le \vdash (\delta_A, \xi_\rho, Set(ra, e)) \Rightarrow (\delta'_A, \xi_\rho)} \ \textsc{SSet-2}$$

$$\frac{\begin{array}{c} ge, le, \delta_C \vdash e \Rightarrow v \qquad mra = ra_1 {+}{+} ra_2 {+}{+} \cdots {+}{+} ra_m \\ ge, le, \delta_C \vdash ra_1 \Rightarrow regacc(k, i_1, j_1, bv_1) \\ ge, le, \delta_C \vdash ra_2 \Rightarrow regacc(k, i_2, j_2, bv_2) \\ \cdots \qquad ge, le, \delta_C \vdash ra_m \Rightarrow regacc(k, i_m, j_m, bv_m) \\ j_1 = i_2 + 1 \qquad j_2 = i_3 + 1 \qquad \cdots \qquad j_{m-1} = i_m + 1 \\ bv' = trans\_to\_bits(v, n) \qquad n = i_1 - j_m + 1 \\ bv'_1 = bv'[i_1, j_1] \qquad bv'_2 = bv'[i_2, j_2] \qquad \cdots \qquad bv'_m = bv'[i_m, j_m] \\ \delta'_C = \delta_C \mid_{ra_1 \Rightarrow regacc(k,i_1,j_1,bv'_1), ra_2 \Rightarrow regacc(k,i_2,j_2,bv'_2), \cdots, ra_m \Rightarrow regacc(k,i_m,j_m,bv'_m)} \\ \delta_C \ is \ \delta_A, \delta_{B0} \ or \ \delta_{B1} \end{array}}{ge, le \vdash (\delta_C, Mov(mra, e)) \Rightarrow \delta'_C} \ \textsc{SMov-1}$$

$$\frac{\begin{array}{c} ge, le, \delta_A, \xi_\rho \vdash e \Rightarrow v \qquad mra = ra_1 {+}{+} ra_2 {+}{+} \cdots {+}{+} ra_m \\ ge, le, \delta_A, \xi_\rho \vdash ra_1 \Rightarrow regacc(k, i_1, j_1, bv_1) \\ ge, le, \delta_A, \xi_\rho \vdash ra_2 \Rightarrow regacc(k, i_2, j_2, bv_2) \\ \cdots \qquad ge, le, \delta_A, \xi_\rho \vdash ra_m \Rightarrow regacc(k, i_m, j_m, bv_m) \\ j_1 = i_2 + 1 \qquad j_2 = i_3 + 1 \qquad \cdots \qquad j_{m-1} = i_m + 1 \\ bv' = trans\_to\_bits(v, n) \qquad n = i_1 - j_m + 1 \\ bv'_1 = bv'[i_1, j_1] \qquad bv'_2 = bv'[i_2, j_2] \qquad \cdots \qquad bv'_m = bv'[i_m, j_m] \\ \delta'_A = \delta_A \mid_{ra_1 \Rightarrow regacc(rid,i_1,j_1,bv'_1), ra_2 \Rightarrow regacc(rid,i_2,j_2,bv'_2), \cdots, ra_m \Rightarrow regacc(rid,i_m,j_m,bv'_m)} \end{array}}{ge, le \vdash (\delta_A, \xi_\rho, Mov(mra, e)) \Rightarrow (\delta'_A, \xi_\rho)} \ \textsc{SMov-2}$$

$$\frac{\begin{array}{c} ge, le, \delta_C \vdash e_1 \Rightarrow v_1 \\ ge, le, \delta_C \vdash e_2 \Rightarrow v_2 \qquad ge, le, \delta_C \vdash ra \Rightarrow regacc(k,i,j,bv) \\ b = trans\_to\_int(v_1) == trans\_to\_int(v_2) \qquad bv' = trans\_to\_bits(b, n) \\ n = i - j + 1 \qquad \delta_C' = \delta_C \mid_{ra \Rightarrow regacc(k,i,j,bv')} \qquad \delta_C \ is \ \delta_A, \delta_{B0} \ or \ \delta_{B1} \end{array}}{ge, le \vdash (\delta_C, Eq(ra, e_1, e_2)) \Rightarrow \delta_C'} \text{ SEQ-1}$$

$$\frac{\begin{array}{c} ge, le, \delta_A, \xi_\rho \vdash e_1 \Rightarrow v_1 \\ ge, le, \delta_A, \xi_\rho \vdash e_2 \Rightarrow v_2 \qquad ge, le, \delta_A, \xi_\rho \vdash ra \Rightarrow regacc(k,i,j,bv) \\ b = trans\_to\_int(v_1) == trans\_to\_int(v_2) \qquad bv' = trans\_to\_bits(b, n) \\ n = i - j + 1 \qquad \delta_A' = \delta_A \mid_{ra = regacc(k,i,j,bv')} \end{array}}{ge, le \vdash (\delta_A, \xi_\rho, Eq(ra, e_1, e_2)) \Rightarrow (\delta_A', \xi_\rho)} \text{ SEQ-2}$$

$$\frac{\begin{array}{c} ge, le, \delta_C \vdash e_1 \Rightarrow v_1 \\ ge, le, \delta_C \vdash e_2 \Rightarrow v_2 \qquad ge, le, \delta_C \vdash ra \Rightarrow regacc(k,i,j,bv) \\ b = trans\_to\_int(v_1) > trans\_to\_int(v_2) \qquad bv' = trans\_to\_bits(b, n) \\ n = i - j + 1 \qquad \delta_C' = \delta_C \mid_{ra \Rightarrow regacc(k,i,j,bv')} \qquad \delta_C \ is \ \delta_A, \delta_{B0} \ or \ \delta_{B1} \end{array}}{ge, le \vdash (\delta_C, Lg(ra, e_1, e_2)) \Rightarrow \delta_C'} \text{ SLG-1}$$

$$\frac{\begin{array}{c} ge, le, \delta_A, \xi_\rho \vdash e_1 \Rightarrow v_1 \\ ge, le, \delta_A, \xi_\rho \vdash e_2 \Rightarrow v_2 \qquad ge, le, \delta_A, \xi_\rho \vdash ra \Rightarrow regacc(k,i,j,bv) \\ b = trans\_to\_int(v_1) > trans\_to\_int(v_2) \qquad bv' = trans\_to\_bits(b, n) \\ n = i - j + 1 \qquad \delta_A' = \delta_A \mid_{ra = regacc(k,i,j,bv')} \end{array}}{ge, le \vdash (\delta_A, \xi_\rho, Lg(ra, e_1, e_2)) \Rightarrow (\delta_A', \xi_\rho)} \text{ SLG-2}$$

- Action statement

$$\frac{\begin{array}{c} \forall i : 1 \leq i \leq k.(ge \vdash (le^i, \delta_C^i, ins_i) \Rightarrow (le^{i+1}, \delta_C^{i+1})) \\ \delta_C \ is \ \delta_A, \delta_{B0} \ or \ \delta_{B1} \end{array}}{ge \vdash (le^1, \delta_C^1, Action(ins_1, \cdots, ins_k)) \Rightarrow (le^{k+1}, \delta_C^{k+1})} \text{ SAS-1}$$

$$\frac{\forall i : 1 \leq i \leq k.(ge \vdash (le^i, \delta_A^i, \xi_\rho^i, ins_i) \Rightarrow le^{i+1}, (\delta_A^{i+1}, \xi_\rho^{i+1})}{ge, le \vdash (le^1, \delta_A^1, \xi_\rho, Action(ins_1, \cdots, ins_k)) \Rightarrow (le^{k+1}, \delta_A^{k+1}, \xi_\rho^{k+1})} \text{ SAS-2}$$

- Bypass statement

$$\frac{\begin{array}{c} ge, le, \delta_A \vdash c \Rightarrow n \\ le = (\xi_\iota, nh, len, bp) \qquad bp' \vdash bypass(n) \qquad le' = (\xi_\iota, nh, len, bp') \end{array}}{ge \vdash (le, \delta_A, Bypass(c)) \Rightarrow (le', \delta_A)} \text{SBypS-1}$$

$$\frac{\begin{array}{c} ge, le, \delta_A, \xi_\rho \vdash c \Rightarrow n \\ le = (\xi_\iota, nh, len, bp) \qquad bp' \vdash bypass(n) \qquad le' = (\xi_\iota, nh, len, bp') \end{array}}{ge \vdash (le, \delta_A, \xi_\rho, Bypass(c)) \Rightarrow (le', \delta_A, \xi_\rho)} \text{SBypS-2}$$

- NextHeader statement

$$\frac{\begin{array}{cc} ge, le, \delta_A \vdash id \Rightarrow pid & le = (\xi_\iota, nh, len, bp) \\ nh' \vdash nextheader(pid) & le' = (\xi_\iota, nh', len, bp) \end{array}}{ge \vdash (le, \delta_A, NextHeader(id)) \Rightarrow (le', \delta_A)} \text{SNextHeader-1}$$

$$\frac{\begin{array}{cc} ge, le, \delta_A, \xi_\rho \vdash id \Rightarrow pid & le = (\xi_\iota, nh, len, bp) \\ nh' \vdash nextheader(pid) & le' = (\xi_\iota, nh', len, bp) \end{array}}{ge \vdash (le, \delta_A, \xi_\rho, NextHeader(id)) \Rightarrow (le', \delta_A, \xi_\rho)} \text{SNextHeader-2}$$

- Length statement

$$\frac{\begin{array}{c} ge, le, \delta_A \vdash e \Rightarrow n \\ le = (\xi_\iota, nh, len, bp) \qquad len' \vdash length(n) \qquad le' = (\xi_\iota, nh, len', bp) \end{array}}{ge \vdash (le, \delta_A, Length(e)) \Rightarrow (le', \delta_A)} \text{SLength-1}$$

$$\frac{\begin{array}{c} ge, le, \delta_A, \xi_\rho \vdash e \Rightarrow n \qquad le = (\xi_\iota, nh, len, bp) \\ \textit{There exists an unique protocol instance identified by id, such that } (id : (len', pins)) \in \xi_\iota \\ \xi'_\iota = \xi_\iota \mid_{id \Rightarrow (length(n), pins)} \qquad le' = (\xi'_\iota, nh, len, bp) \end{array}}{ge \vdash (le, \delta_A, \xi_\rho, Length(e)) \Rightarrow (le', \delta_A, \xi_\rho)} \text{SLength-2}$$

- Layer statement

$$\frac{\begin{array}{c} ls\_list = (ls_1, ls_2, \cdots, ls_k) \\ ge \vdash (le, \delta_C, ls_1) \Rightarrow (le^1, \delta_C^1) \qquad ge \vdash (le^1, \delta_C^1, ls_1) \Rightarrow (le^2, \delta_C^2) \\ \cdots \quad ge \vdash (le^{k-1}, \delta_C^{k-1}, ls_k) \Rightarrow (le^k, \delta_C^k) \qquad \delta_C \textit{ is } \delta_A, \delta_{B0} \textit{ or } \delta_{B1} \end{array}}{ge \vdash (le, \delta_C, ls\_list) \Rightarrow (le^k, \delta_C^k)} \text{SLSL}$$

$$\dfrac{\begin{array}{c} \mathit{if\_l\_list} = ((e_1, \mathit{l\_stmts}_1), (e_2, \mathit{l\_stmts}_2), \cdots, (e_k, \mathit{l\_stmts}_k)) \qquad \mathit{d\_l} = \mathit{l\_stmts} \\ ge, le, \delta_C \vdash e_1 \Rightarrow b_1 \qquad ge, le, \delta_C \vdash e_2 \Rightarrow b_2 \qquad \cdots \qquad ge, le, \delta_C \vdash e_k \Rightarrow b_k \\ \mathit{if}\ b_1\ \mathit{then}\ ge \vdash (le, \delta_C, \mathit{l\_stmts}_1) \Rightarrow (le', \delta'_C) \\ \mathit{elseif}\ b_2\ \mathit{then}\ ge \vdash (le, \delta_C, \mathit{l\_stmts}_2) \Rightarrow (le', \delta'_C) \\ \cdots \qquad \mathit{elseif}\ b_k\ \mathit{then}\ ge \vdash (le, \delta_C, \mathit{l\_stmts}_k) \Rightarrow (le', \delta'_C) \\ \mathit{else}\ ge \vdash (le, \delta_C, \mathit{l\_stmts}) \Rightarrow (le', \delta'_C) \qquad \delta_C\ \mathit{is}\ \delta_A, \delta_{B0}\ \mathit{or}\ \delta_{B1} \end{array}}{ge \vdash (le, \delta_C, \mathit{IfElseL}(\mathit{if\_l\_list}, \mathit{d\_l})) \Rightarrow (le', \delta'_C)}\ \text{SIFEL}$$

- Layer local actions

$$\dfrac{\begin{array}{c} \mathit{caas} = \mathit{CellA}(\mathit{ca\_l\_s\_list}) \qquad \mathit{cb0as} = \mathit{CellB0}(\mathit{cb0\_l\_s\_list}) \\ \mathit{cb1as} = \mathit{CellB1}(\mathit{cb1\_l\_s\_list}) \qquad ge \vdash (le, \delta_A, \mathit{ca\_l\_s\_list}) \Rightarrow (le', \delta'_A) \\ ge \vdash (le', \delta_{B0}, \mathit{cb0\_l\_s\_list}) \Rightarrow (le', \delta'_{B0}) \\ ge \vdash (le', \delta_{B1}, \mathit{cb1\_l\_s\_list}) \Rightarrow (le', \delta'_{B1}) \end{array}}{ge \vdash (le, \mathit{LocalActions}(\mathit{caas}, \mathit{cb0as}, \mathit{cb1as})) \Rightarrow le'}\ \text{SLLA}$$

- Layer action

$$\dfrac{ge \vdash (le_{id}, las) \Rightarrow le'_{id}}{\emptyset \vdash (ge, \mathit{LayerAction}(id, lvs, lrd, ld, las)) \Rightarrow ge}\ \text{SLA}$$

- Protocol statement

$$\dfrac{\begin{array}{c} \mathit{ps\_list} = (ps_1, ps_2, \cdots, ps_k) \qquad ge \vdash (le, \delta_A, \xi_\rho, ps_1) \Rightarrow (le^1, \delta^1_A, \xi_\rho) \\ ge \vdash (le^1, \delta^1_A, \xi_\rho, ps_1) \Rightarrow (le^2, \delta^2_A, \xi_\rho) \\ \cdots \qquad ge \vdash (le^{k-1}, \delta^{k-1}_A, \xi_\rho, ps_k) \Rightarrow (le^k, \delta^k_A, \xi_\rho) \end{array}}{ge \vdash (le, \delta_A, \xi_\rho, \mathit{ps\_list}) \Rightarrow (le^k, \delta^k_A, \xi_\rho)}\ \text{SPSL}$$

$$\dfrac{\begin{array}{c} \mathit{if\_p\_list} = ((e_1, \mathit{p\_stmts}_1), (e_2, \mathit{p\_stmts}_2), \cdots, (e_k, \mathit{p\_stmts}_k)) \\ \mathit{d\_p} = \mathit{p\_stmts} \\ ge, le, \delta_A, \xi_\rho \vdash e_1 \Rightarrow b_1 \qquad ge, le, \delta_A, \xi_\rho \vdash e_2 \Rightarrow b_2 \qquad \cdots \qquad ge, le, \delta_A, \xi_\rho \vdash e_k \Rightarrow b_k \\ \mathit{if}\ b_1\ \mathit{then}\ ge \vdash (le, \delta_A, \xi_\rho, \mathit{p\_stmts}_1) \Rightarrow (le', \delta'_A, \xi_\rho) \\ \mathit{elseif}\ b_2\ \mathit{then}\ ge \vdash (le, \delta_A, \xi_\rho, \mathit{p\_stmts}_2) \Rightarrow (le', \delta'_A, \xi_\rho) \\ \cdots \qquad \mathit{elseif}\ b_k\ \mathit{then}\ ge \vdash (le, \delta_A, \xi_\rho, \mathit{p\_stmts}_k) \Rightarrow (le', \delta'_A, \xi_\rho) \\ \mathit{else}\ ge \vdash (le, \delta_A, \xi_\rho, \mathit{p\_stmts}) \Rightarrow (le', \delta'_A, \xi_\rho) \end{array}}{ge \vdash (le, \delta_A, \xi_\rho, \mathit{IfElseL}(\mathit{if\_p\_list}, \mathit{d\_p})) \Rightarrow (le', \delta'_A, \xi_\rho)}\ \text{SIFEP}$$

- Protocol declaration

$$\frac{ge \vdash (le, \delta_A, \xi_\rho, p\_stmts) \Rightarrow (le', \delta'_A, \xi_\rho)}{ge \vdash (le, \delta_A, \xi_\rho, Protocol(fields, p\_stmts)) \Rightarrow (le', \delta'_A, \xi_\rho)} \text{ SPROTOCOL}$$

- Global declarations

$$\frac{\begin{array}{c} ge = (\gamma, \sigma, \delta) \qquad \gamma = (lr, cr, ps, ls, \iota, \rho) \\ \forall lid \in dom(\iota).(le_{lid} = (\xi_\iota^{lid}, \cdots) \wedge \exists id, pins.\xi_\iota^{lid} \vdash id \Rightarrow (len, pins) \\ \rightarrow ge \vdash (le_{lid}, \delta_A^{lid}, \xi_\rho^{pid}, p) \Rightarrow (le'_{lid}, \delta_A'^{lid}, \xi_\rho^{pid})) \end{array}}{\emptyset \vdash (ge, ProtocolDecl(pid, p)) \Rightarrow ge} \text{ SPDG}$$

7.3 Semantics of the P3 Assembly

7.3.1 Semantic environment

In most of the cases, a semantic environment maps variables to the values and memory for registers and fields, and has the form

$$\mathcal{E} ::= [\ x_1 : v_1,\ x_2 : v_2,\ ...,\ x_n : v_n\ ]$$

where $x_i \neq x_j$ for all i and j , satisfying $i \neq j$ and $(1 \leq i, j \leq n)$.

Figure 11 show all the semantic environments we use to define the semantics.

7.3.2 Judgements

The judgements used in the semantics definition can be inferred from the semantic rules in the subsection 7.3.3. To save the space, we omit to present them separately.

7.3.3 Semantic rules

- Initialization of ι, opened at the beginning of the specification and not to be closed

$$\frac{\begin{array}{c} ge = (\iota, lr, cr, \delta) \\ \iota' = \{layer\_id : layer\_block \mid layer\_block = LayerBlock(layer\_id, pins, cella, cellb0, cellb1)\} \\ ge' = (\iota', lr, cr, \delta) \end{array}}{\vdash (ge, \text{“}at\text{-}beginning\text{-}\iota\text{”})) \Rightarrow ge'} \text{ (AIL)}$$

$$
\begin{array}{llll}
\textit{Global} & ge & ::= & (\iota,\ lr,\ cr,\ \delta) & \text{divide global environment into four parts} \\
 & \iota & ::= & id \to ldef & \text{map a layer identifier to a layer definition} \\
 & lr & ::= & lreglen(k) & \text{the } \textit{Lreglen} \text{ value set to } k \\
 & cr & ::= & creglen(k) & \text{the } \textit{Creglen} \text{ value set to } k \\
 & \delta & ::= & \{\ IRF(ofs,\ size,\ bv)\ \} & \text{The IRF environment} \\
\textit{Layer} & le & ::= & (\xi,\ nh,\ len,\ bp,\ \lambda) & \text{divide layer local environment into five parts} \\
 & \xi & ::= & id \to fdv(k,\ bv) & \text{map a protocol instance identifier to a memory} \\
 & & & & \text{for its fields with the size } k \text{ and the bits' value } bv \\
 & nh & ::= & nextheader(k) & \text{the } \textit{NextHeader} \text{ set to the protocol identified by} \\
 & & & & \text{the index (an integer } k\text{) of its identifier} \\
 & len & ::= & length(k) & \text{the } \textit{Length} \text{ bound to an integer } k \\
 & bp & ::= & bypass(k) & \text{the } \textit{Bypass} \text{ bound to an integer } k \\
 & \lambda & ::= & id \to (nxt,\ nxt,\ nxt) & \text{map a protocol identifier to the IRF and FRA offsets} \\
 & & & & \text{of slots to be used future in Cell A, Cell B0 and Cell B1} \\
 & nxt & ::= & irf(\{num\}) \times fra(\{num\}) & num \text{ is a hexadecimal number} \\
\textit{Cell} & ce & ::= & \delta_A \mid \delta_{B0} \mid \delta_{B1} & \text{divide cell local environment into three parts} \\
 & \delta_A & ::= & (\ rs,\ idx\ ) & \\
 & \delta_{B0} & ::= & (\ rs,\ idx\ ) & \\
 & \delta_{B1} & ::= & (\ rs,\ idx\ ) & \\
 & rs & ::= & \{\ IRF(ofs,\ size,\ bv)\ \} & \text{the IRF state} \\
 & idx & ::= & comsindex(k) & \text{the commands' index } k \text{ in both } pb \text{ and } pc\_cur \text{ tables}
\end{array}
$$

Figure 11: Semantic Environments for the Assembly

- Initialization of lr and cr, opened at the beginning of the specification and not to be closed

$$
\frac{\begin{array}{c} ge = (\iota, lr, cr, \delta) \qquad LrCr = Lreglen(n)\,Creglen(k) \qquad lr' = Lreglen(n) \\ cr' = Creglen(k) \qquad n = 3 * k \qquad ge' = (\iota, lr', cr', \delta) \end{array}}{\vdash (ge, LrCr) \Rightarrow ge'} \ \textsc{ALrCr}
$$

- Initialization of δ, initialized at the beginning of the specification (Rule AIR-1) and changed each time at the leaving of a layer context (Rule AIR-2).

$$
\frac{ge = (\iota, lr, cr, \delta) \qquad \delta' = \emptyset \qquad ge' = (\iota, lr, cr, \delta')}{\vdash (ge, \text{“}\textit{at-beginning-}\delta\text{”}) \Rightarrow ge'} \ (\text{AIR-1})
$$

$$
\frac{\begin{array}{c} ge = (\iota, lr, cr, \delta) \qquad lr = lreglen(n) \qquad cr = creglen(k) \\ n = 3 * k \qquad le = (\xi, nh, len, bp, \lambda) \qquad ce = (\delta_A, \delta_{B0}, \delta_{B1}) \\ \delta' = \{IRF(2 * k + n_1, 2 * k + n_2, bva) \mid IRF(n_1, n_2, bv) \in \delta_A\} \\ \cup\{IRF(k + n_1, k + n_2, bvb0) \mid IRF(n_1, n_2, bvb0 \in \delta_{B0}\} \\ \cup\{IRF(n_1, n_2, bvb1) \mid IRF(n_1, n_2, bvb1) \in \delta_{B1}\} \qquad ge' = (\iota, lr, cr, \delta') \\ le' = (\emptyset, null, null, null, \emptyset) \qquad \delta'_C = null \qquad \delta_C\ is\ \delta_A, \delta_{B0}\ or\ \delta_{B1} \end{array}}{\vdash (ge, le, \delta_C, \text{“}\textit{layer-switch}\text{”}) \Rightarrow (ge', le', \delta'_C)} \ (\text{AIR-2})
$$

- Initialization of le, opened at the beginning and closed at the end of a Layer specification

$$\frac{\begin{array}{c} ge = (\iota, lr, cr, \delta) \qquad le = (\xi, nh, len, bp, \lambda) \qquad ins\_name \notin dom(\xi) \\ \xi' = \xi \cup \{id : fdv(ins\_size, bv)\},\ \textit{where all bv's are the input from the hardware} \\ le' = (\xi', null, null, null, \emptyset) \end{array}}{ge \vdash (le, Pins(ins\_name, ins\_size)) \Rightarrow le'} \text{(AIPINS1)}$$

$$\frac{\begin{array}{c} ge = (\iota, lr, cr, \delta) \\ le = (\xi, nh, len, bp, \lambda) \qquad pins = Pins(ins\_name, ins\_size) :: pins' \\ ge \vdash (le, Pins(ins\_name, ins\_size)) \Rightarrow le' \qquad ge \vdash (le', pins') \Rightarrow le'' \end{array}}{ge \vdash (le, pins) \Rightarrow le''} \text{(AIPINS)}$$

- Initialization of δ_A at the CellA Registers specification, opened at the beginning of a Cell A specification, and closed at the leaving of the layer context

$$\frac{\begin{array}{c} ge = (\iota, lr, cr, \delta) \\ lr = (lreglen(n) \qquad cr = creglen(k) \qquad n = 3 * k \qquad \delta_A = (rs, idx) \\ rs' = \{IRF(ofs, size, bva) \mid IRF(2 * k + ofs, size, bva) \in \delta \wedge ofs + size < k\} \\ \delta'_A = (rs', idx) \end{array}}{ge, le \vdash (\delta_A, \text{“}\textit{at the beginning of Cell A}\text{”})) \Rightarrow \delta'_A} \text{(AICA)}$$

- Initialization of δ_{B0} at the CellB0 Registers specification, opened at the beginning of a Cell B0 specification, and closed at the leaving of the layer context

$$\frac{\begin{array}{c} ge = (\iota, lr, cr, \delta) \\ lr = (lreglen(n) \qquad cr = creglen(k) \qquad n = 3 * k \qquad \delta_{B0} = (rs, idx) \\ rs' = \{IRF(ofs, size, bvb0) \mid IRF(k + ofs, size, bvb0) \in \delta \wedge ofs + size < k\} \\ \delta'_{B0} = (rs', idx) \end{array}}{ge, le \vdash (\delta_{B0}, \text{“}\textit{at the beginning of Cell B0}\text{”})) \Rightarrow \delta'_{B0}} \text{(AICB0)}$$

- Initialization of δ_{B1} at the CellB0 Registers specification, opened at the beginning of a Cell B1 specification, and closed at the leaving of the layer context

$$\frac{\begin{array}{c} ge = (\iota, lr, cr, \delta) \\ lr = (lreglen(n) \qquad cr = creglen(k) \qquad n = 3 * k \qquad \delta_{B1} = (rs, idx) \\ rs' = \{IRF(ofs, size, bvb1) \mid IRF(ofs, size, bvb1) \in \delta \wedge ofs + size < k\} \\ \delta'_{B1} = (rs', idx) \end{array}}{ge, le \vdash (\delta_{B1}, \text{“}at\ the\ beginning\ of\ Cell\ B1\text{”})) \Rightarrow \delta'_{B1}} \text{ (AICB1)}$$

- Expressions

$$\frac{ge = (\iota, lr, cr, \delta) \qquad \sigma \vdash num \Rightarrow v \qquad \delta_C\ is\ \delta_A, \delta_{B0}\ or\ \delta_{B1}}{ge, le, \delta_C \vdash num \Rightarrow v} \text{ ACE-1}$$

$$\frac{\delta_C = (rs, idx) \qquad IRF(ofs, size, bv) \in rs \qquad \delta_C\ is\ \delta_A, \delta_{B0}\ or\ \delta_{B1}}{ge, le, \delta_C \vdash (IRF, ofs, size) \Rightarrow bv} \text{ ACE-2}$$

$$\frac{\begin{array}{c} le = (\xi, nh, len, bp, \lambda) \\ \xi(ins\_id) = fdv(ins\_size, bv') \qquad ofs + size < ins\_size \\ bv = substring(bv', ofs, size) \qquad \delta_C\ is\ \delta_A, \delta_{B0}\ or\ \delta_{B1} \end{array}}{ge, le, \delta_C \vdash (ins\_id, ofs, size) \Rightarrow bv} \text{ ACE-3}$$

$$\frac{\begin{array}{c} ge, le, \delta_C \vdash num \Rightarrow v \qquad ge, le, \delta_C \vdash (IRF, ofs, size) \Rightarrow bv \\ r = if\ (value(bv) = v)\ then\ \underline{true}\ else\ \underline{false} \qquad \delta_C\ is\ \delta_A, \delta_{B0}\ or\ \delta_{B1} \end{array}}{ge, le, \delta_C \vdash ((IRF, ofs, size), num) \Rightarrow r} \text{ ACond-1}$$

$$\frac{\begin{array}{c} ge, le, \delta_C \vdash num \Rightarrow v \qquad ge, le, \delta_C \vdash (ins\_id, ofs, size) \Rightarrow bv \\ r = if\ (value(bv) = v)\ then\ \underline{true}\ else\ \underline{false} \qquad \delta_C\ is\ \delta_A, \delta_{B0}\ or\ \delta_{B1} \end{array}}{ge, le, \delta_C \vdash ((ins\_id, ofs, size), num) \Rightarrow r} \text{ ACond-2}$$

$$\frac{\begin{array}{c} conds = cond :: conds' \\ ge, le, \delta_C \vdash cond \Rightarrow r_1 \qquad ge, le, \delta_C \vdash conds' \Rightarrow r_2 \\ r = if\ (r_1 = r_2 = \underline{true})\ then\ \underline{true}\ else\ \underline{false} \qquad \delta_C\ is\ \delta_A, \delta_{B0}\ or\ \delta_{B1} \end{array}}{ge, le, \delta_C \vdash conds \Rightarrow r} \text{ AConds}$$

- Instructions

$$\frac{\begin{array}{c} ge = (\iota, lr, cr, \delta) \qquad \sigma \vdash num \Rightarrow v \\ \delta_C = (rs, idx) \qquad ra = IRF(ofs, size) \qquad bv' = trans\_to\_bits(v, size) \\ rs' = (rs - \{IRF(ofs, size, bv)\}) \cup \{IRF(ofs, size, bv')\} \\ \delta'_C = (rs', idx) \qquad \delta_C\ is\ \delta_A, \delta_{B0}\ or\ \delta_{B1} \end{array}}{ge, le \vdash (\delta_C, Set(ra, num)) \Rightarrow \delta'_C} \textsc{ASet}$$

$$\frac{\begin{array}{c} ge = (\iota, lr, cr, \delta) \qquad \delta_C = (rs, idx) \qquad ra = IRF(ofs, size) \\ ge, le, \delta_C \vdash sra \Rightarrow v \qquad bv' = trans\_to\_bits(v, size) \\ rs' = (rs - \{IRF(ofs, size, bv)\}) \cup \{IRF(ofs, size, bv')\} \\ \delta'_C = (rs', idx) \qquad \delta_C\ is\ \delta_A, \delta_{B0}\ or\ \delta_{B1} \end{array}}{ge, le \vdash (\delta_C, Mov(ra, sra)) \Rightarrow \delta'_C} \textsc{AMov}$$

$$\frac{\begin{array}{c} ge = (\iota, lr, cr, \delta) \\ \delta_C = (rs, idx) \qquad ra = IRF(ofs, size) \qquad ge, le, \delta_C \vdash sra_1 \Rightarrow v_1 \\ ge, le, \delta_C \vdash sra_2 \Rightarrow v_2 \qquad b = (trans\_to\_int(v_1) == trans\_to\_int(v_2)) \\ bv' = trans\_to\_bits(b, size) \\ rs' = (rs - \{IRF(ofs, size, bv)\}) \cup \{IRF(ofs, size, bv')\} \\ \delta'_C = (rs', idx) \qquad \delta_C\ is\ \delta_A, \delta_{B0}\ or\ \delta_{B1} \end{array}}{ge, le \vdash (\delta_C, Eq(ra, sra_1, sra_2)) \Rightarrow \delta'_C} \textsc{AEq}$$

$$\frac{\begin{array}{c} ge = (\iota, lr, cr, \delta) \qquad \delta_C = (rs, idx) \\ ra = IRF(ofs, size) \qquad ge, le, \delta_C \vdash sra_1 \Rightarrow v_1 \qquad ge, le, \delta_C \vdash sra_2 \Rightarrow v_2 \\ b = trans\_to\_int(v_1) > trans\_to\_int(v_2) \qquad bv' = trans\_to\_bits(b, size) \\ rs' = (rs - \{IRF(ofs, size, bv)\}) \cup \{IRF(ofs, size, bv')\} \\ \delta'_C = (rs', idx) \qquad \delta_C\ is\ \delta_A, \delta_{B0}\ or\ \delta_{B1} \end{array}}{ge, le \vdash (\delta_C, Lg(ra, sra_1, sra_2)) \Rightarrow \delta'_C} \textsc{ALg}$$

$$\frac{\begin{array}{c} cmds = cmd :: cmds' \qquad ge, le \vdash (\delta_C, cmd) \Rightarrow \delta'_C \\ ge, le \vdash (\delta'_C, cmds') \Rightarrow \delta''_C \delta_C\ is\ \delta_A, \delta_{B0}\ or\ \delta_{B1} \end{array}}{ge, le \vdash (\delta_C, cmds) \Rightarrow \delta''_C} \textsc{ACmds}$$

- CellA

$$\frac{\begin{array}{c} le = (\xi, nh, len, bp, \lambda) \qquad \delta_A = (rs, idx) \\ ge, le, \delta_A \vdash conds \Rightarrow b \qquad nh' = if\ (b == \underline{true})\ then\ nxt\_id\ else\ nh \\ bp' = if\ (b == \underline{true})\ then\ bypas\ else\ bp \\ idx' = if\ (b == \underline{true})\ then\ sub\_id\ else\ idx \\ \delta'_A = (rs, idx') \qquad le' = (\xi, nh', len, bp', \lambda) \end{array}}{ge \vdash (le, \delta_A, (hdr\_id, conds, sub\_id, nxt\_id, bypas)) \Rightarrow (le', \delta'_A)}\ \mathrm{A}\mathit{cella\_pb\_item}$$

$$\frac{\begin{array}{c} ca\_pb\_items = ca\_pb\_item :: ca\_pb\_items' \\ ge \vdash (le, \delta_A, ca\_pb\_item) \Rightarrow (le', \delta'_A) \\ ge \vdash (le', \delta'_A, ca\_pb\_items') \Rightarrow (le'', \delta''_A) \end{array}}{ge \vdash (le, \delta_A, Apb(ca\_pb\_items)) \Rightarrow (le'', \delta''_A)}\ \mathrm{A}\mathit{cella\_pb}$$

$$\frac{\begin{array}{c} le = (\xi, nh, len, bp, \lambda) \qquad \delta_A = (rs, idx) \qquad ge, le \vdash (\delta_A, cmds) \Rightarrow \delta''_A \\ len' = if\ (sub\_id == idx)\ then\ lyr\_offset\ else\ len \\ le' = (\xi, nh, len', bp, \lambda) \qquad \delta'_A = if\ (sub\_id == idx)\ then\ \delta''_A\ else\ \delta_A \end{array}}{ge \vdash (le, \delta_A, (sub\_id, cmds, lyr\_offset)) \Rightarrow (le', \delta'_A)}\ \mathrm{A}\mathit{cella\_pc\_cur\_item}$$

$$\frac{\begin{array}{c} ca\_pc\_cur\_items = ca\_pc\_cur\_item :: ca\_pc\_cur\_items' \\ ge \vdash (le, \delta_A, ca\_pc\_cur\_item) \Rightarrow (le', \delta'_A) \\ ge \vdash (le', \delta'_A, ca\_pc\_cur\_items') \Rightarrow (le'', \delta''_A) \end{array}}{ge \vdash (le, \delta_A, ApcCur(ca\_pc\_cur\_items)) \Rightarrow (le'', \delta''_A)}\ \mathrm{A}\mathit{cella\_pc\_cur}$$

$$\frac{\begin{array}{c} le = (\xi, nh, len, bp, \lambda) \\ ca\_nxt = CellANxt(IRFOffset(n_1, \cdots, n_r), ProtOffset(m_1, \cdots, m_p)) \\ anxt = (irf(n_1, \cdots, n_r), fra(m_1, \cdots, m_p)) \\ cb0\_nxt = CellB0Nxt(IRFOffset(n'_1, \cdots, n'_{r'}), ProtOffset(m'_1, \cdots, m'_{p'})) \\ b0nxt = (irf(n'_1, \cdots, n'_{r'}), fra(m'_1, \cdots, m'_{p'})) \\ cb1\_nxt = CellB1Nxt(IRFOffset(n''_1, \cdots, n''_{r''}), ProtOffset(m''_1, \cdots, m''_{p''})) \\ b1nxt = (irf(n''_1, \cdots, n''_{r''}), fra(m''_1, \cdots, m''_{p''})) \\ \lambda' = \lambda \cup \{(nxt\_id, (anxt, b0nxt, b1nxt))\} le' = (\xi, nh, len, bp, \lambda') \end{array}}{ge \vdash (le, (nxt\_id, ca\_nxt, cb0\_nxt, cb1\_nxt)) \Rightarrow le'}\ \mathrm{A}\mathit{cella\_pc\_nxt\_item}$$

$$\frac{\begin{array}{c} ca\_pc\_nxt\_items = ca\_pc\_nxt\_item :: ca\_pc\_nxt\_items' \\ ge \vdash (le, ca\_pc\_nxt\_item) \Rightarrow le' \\ ge \vdash (le', ca\_pc\_nxt\_items') \Rightarrow le'' \end{array}}{ge \vdash (le, ApcNxt(ca\_pc\_nxt\_items)) \Rightarrow le''}\ \mathrm{A}\mathit{cella\_pc\_nxt}$$

$$\frac{\begin{array}{c} cella = CellA(cella\_pb, cella\_pc\_cur, cella\_pc\_nxt) \\ ge \vdash (le, \delta_A, cella\_pb) \Rightarrow (le', \delta'_A) \\ ge \vdash (le', \delta'_A, cella\_pc\_cur) \Rightarrow (le'', \delta''_A) \\ ge \vdash (le'', cella\_pc\_nxt) \Rightarrow le''' \end{array}}{ge \vdash (le, \delta_A, cella) \Rightarrow (le''', \delta''_A)} \text{A}cella$$

- CellB0

$$\frac{\begin{array}{c} \delta_{B0} = (rs, idx) \qquad ge, le, \delta_{B0} \vdash conds \Rightarrow b \\ idx' = if\ (b == \underline{true})\ then\ sub\_id\ else\ idx \qquad \delta'_{B0} = (rs, idx') \end{array}}{ge, le \vdash (\delta_{B0}, (hdr\_id, conds, sub\_id)) \Rightarrow \delta'_{B0}} \text{A}cellb0\_pb\_item$$

$$\frac{\begin{array}{c} cb0\_pb\_items = cb0\_pb\_item :: cb0\_pb\_items' \\ ge, le \vdash (\delta_{B0}, cb0\_pb\_item) \Rightarrow \delta'_{B0} \\ ge, le \vdash (\delta'_{B0}, cb0\_pb\_items') \Rightarrow \delta''_{B0} \end{array}}{ge, le \vdash (\delta_{B0}, Apb(cb0\_pb\_items)) \Rightarrow \delta''_{B0}} \text{A}cellb0\_pb$$

$$\frac{\begin{array}{c} \delta_{B0} = (rs, idx) \qquad ge, le \vdash (\delta_{B0}, cmds) \Rightarrow \delta''_{B0} \\ \delta'_{B0} = if\ (sub\_id == idx)\ then\ \delta''_{B0}\ else\ \delta_{B0} \end{array}}{ge, le \vdash (\delta_{B0}, (sub\_id, cmds)) \Rightarrow \delta'_{B0}} \text{A}cellb0\_pc\_cur\_item$$

$$\frac{\begin{array}{c} cb0\_pc\_cur\_items = cb0\_pc\_cur\_item :: cb0\_pc\_cur\_items' \\ ge, le \vdash (\delta_{B0}, cb0\_pc\_cur\_item) \Rightarrow \delta'_{B0} \\ ge, le \vdash (\delta'_{B0}, cb0\_pc\_cur\_items') \Rightarrow \delta''_{B0} \end{array}}{ge, le \vdash (\delta_{B0}, B0pcCur(cb0\_pc\_cur\_items)) \Rightarrow \delta''_{B0}} \text{A}cellb0\_pc\_cur$$

$$\frac{\begin{array}{c} cellb0 = CellB0(cellb0\_pb, cellb0\_pc\_cur) \\ ge, le \vdash (\delta_{B0}, cellb0\_pb) \Rightarrow \delta'_{B0} \qquad ge, le \vdash (\delta'_{B0}, cellb0\_pc\_cur) \Rightarrow \delta''_{B0} \end{array}}{ge, le \vdash (\delta_{B0}, cellb0) \Rightarrow \delta''_{B0}} \text{A}cellb0$$

- CellB1

$$\frac{\begin{array}{c} ge = (\iota, lr, cr, \delta) \qquad \delta_{B1} = (rs, idx) \qquad ge, le, \delta_{B1} \vdash conds \Rightarrow b \\ idx' = if\ (b == \underline{true})\ then\ sub\_id\ else\ idx \qquad \delta'_{B1} = (rs, idx') \end{array}}{ge, le \vdash (\delta_{B1}, (hdr\_id, conds, sub\_id)) \Rightarrow \delta'_{B1}}\ \mathrm{A}\mathit{cellb1\_pb\_item}$$

$$\frac{\begin{array}{c} cb1\_pb\_items = cb1\_pb\_item :: cb1\_pb\_items' \\ ge, le \vdash (\delta_{B1}, cb1\_pb\_item) \Rightarrow \delta'_{B1} \\ ge, le \vdash (\delta'_{B1}, cb1\_pb\_items') \Rightarrow \delta''_{B1} \end{array}}{ge, le \vdash (\delta_{B1}, Apb(cb1\_pb\_items)) \Rightarrow \delta''_{B1}}\ \mathrm{A}\mathit{cellb1\_pb}$$

$$\frac{\begin{array}{c} \delta_{B1} = (rs, idx) \qquad ge, le \vdash (\delta_{B1}, cmds) \Rightarrow \delta''_{B1} \\ \delta'_{B1} = if\ (sub\_id == idx)\ then\ \delta''_{B1}\ else\ \delta_{B1} \end{array}}{ge, le \vdash (\delta_{B1}, (sub\_id, cmds)) \Rightarrow \delta'_{B1}}\ \mathrm{A}\mathit{cellb1\_pc\_cur\_item}$$

$$\frac{\begin{array}{c} cb1\_pc\_cur\_items = cb1\_pc\_cur\_item :: cb1\_pc\_cur\_items' \\ ge, le \vdash (\delta_{B1}, cb1\_pc\_cur\_item) \Rightarrow \delta'_{B1} \\ ge, le \vdash (\delta'_{B1}, cb1\_pc\_cur\_items') \Rightarrow \delta''_{B1} \end{array}}{ge, le \vdash (\delta_{B1}, B1pcCur(cb1\_pc\_cur\_items)) \Rightarrow \delta''_{B1}}\ \mathrm{A}\mathit{cellb1\_pc\_cur}$$

$$\frac{\begin{array}{c} cellb1 = CellB1(cellb1\_pb, cellb1\_pc\_cur) \\ ge, le \vdash (\delta_{B1}, cellb1\_pb) \Rightarrow \delta'_{B1} \qquad ge, le \vdash (\delta'_{B1}, cellb1\_pc\_cur) \Rightarrow \delta''_{B1} \end{array}}{ge, le \vdash (\delta_{B1}, cellb1) \Rightarrow \delta''_{B1}}\ \mathrm{A}\mathit{cellb1}$$

- Layer Block

$$\frac{\begin{array}{c} ge \vdash (le, pins) \Rightarrow le' \qquad ge \vdash (le', \delta_A, cella) \Rightarrow (le'', \delta'_A) \\ ge, le'' \vdash (\delta_{B0}, cellb0) \Rightarrow \delta'_{B0} \qquad ge, le'' \vdash (\delta_{B1}, cellb1) \Rightarrow \delta'_{B1} \end{array}}{ge \vdash (le, (\delta_A, \delta_{B0}, \delta_{B1}), LayerBlock(id, pins, cella, cellb0, cellb1)) \Rightarrow (le'', (\delta'_A, \delta'_{B0}, \delta'_{B1}))}\ (\mathrm{A}\mathit{layer\_block})$$

$$\frac{\begin{array}{c} ge = (\iota, lr, cr, \delta) \qquad layer\_blocks = layer\_block :: layer\_blocks' \\ ge \vdash (le, (\delta_A, \delta_{B0}, \delta_{B1}), layer\_block) \Rightarrow (le', (\delta'_A, \delta'_{B0}, \delta'_{B1})) \\ \vdash (ge, le', (\delta'_A, \delta'_{B0}, \delta'_{B1}), \text{“}\mathit{layer\text{-}switch}\text{”}) \Rightarrow (ge', (\emptyset, null, null, null, \emptyset), (null, null, null)) \\ \vdash (ge', layer\_blocks') \Rightarrow ge'' \end{array}}{\vdash (ge, layer\_blocks) \Rightarrow ge''}\ (\mathrm{A}\mathit{layer\_blocks})$$

- Parser

$$\frac{\begin{array}{c} parser\_asm = LrCrlayer\_blocks \\ \vdash (ge, \text{“}at\text{-}beginning\text{-}\iota\text{”})) \Rightarrow ge_1 \qquad \vdash (ge_1, LrCr) \Rightarrow ge_2 \\ \vdash (ge_2, \text{“}at\text{-}beginning\text{-}\iota\text{”})) \Rightarrow ge_3 \qquad \vdash (ge_3, layer\_blocks) \Rightarrow ge' \end{array}}{\vdash (ge, parser\_asm) \Rightarrow ge'} \ (\mathrm{A}parser)$$

7.4 Preserving the Semantics from AST to Assembly

We choose the translation validation approach [13] to certify the semantics preserving property in the translation from a P3 AST to the corresponding P3 assembly.

To verify the correctness of a validation function *Validate*, we should prove that $\forall S, C. Validate(S, C) = true \Rightarrow S \approx C$ [7], where S and C are a P3 AST and a P3 ASM respectively. *Validate*(S,C) = *true* indicates the success of the validation. $S \approx C$ stands for the semantic equivalence.

The validator is a component of the compiler, which will be added after the end of the translation. A transformation function with a validator can be described as [7]:

Comp'(S) = *match Comp*(S) *with*
| *Error* → *Error*
| *OK*(C) → *if Validate*(S, C) *then OK*(C) *else Error*

8 Conclusion

This document is used as the manual in the design and the implementation of the P3C compiler.

Currently, we have implemented the compilation of the P3C as the requirement of the project 2017ZX01030-301-003, including the *scanning*, *parsing*, *type checking* and *translation to the ASM* as is shown in Fig.1. The translation to the configure file is implemented partially because we need the feedback from the hardware design. Unfortunately, the hardware design had been delayed for some unexpected reasons.

The verification of the P3C compiler is not the mandatory requirement of the project 2017ZX01030-301-003. However, we will continue to complete it as soon as possible for the great significance to do so.

References

[1] The coq development team, the coq proof assistant reference manual. *http://coq.inria.fr/*.

[2] Yves Bertot and Pierre Casteran. *Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

[3] Sandrine Blazy and Xavier Leroy. Mechanized semantics for the clight subset of the c language. *Journal of Automated Reasoning*, 43(3):263–288, 2009.

[4] CompCert. http://compcert.inria.fr.

[5] J.-H. Jourdan, F. Pottier, and X. Leroy. Validating lr(1) parsers. *Proceedings of 21st European Symposium on Programming, ESOP 2012, Lecture Notes in Computer Science, vol. 7211*, pages 397–416, 2012.

[6] Xavier Leroy. Formal certification of a compiler back-end or: programming a compiler with a proof assistant. *Proceedings of the 33rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, 41(1):42–54, January 2006.

[7] Xavier Leroy. Formal verification of a realistic compiler. *Communications of the ACM*, 52(7):107–115, 2009.

[8] Menhir-reference. Url. `http://soft.cs.tsinghua.edu.cn:8000`, 2018.

[9] Greg Morrisett. Technical perspective: A compiler's story. *Communications of the ACM*, 52(7):106–106, 2009.

[10] V. C. Ngo, J.-P. Talpin, and T. Gautier. Translation validation for synchronous data-flow specification in the signal compiler. *Proceedings of the 35th IFIP WG 6.1 International Conference on Formal Techniques for Distributed Objects, Components, and Systems (FORTE 2015), Lecture Notes in Computer Science*, 9039:66–80, June 2015.

[11] V. C. Ngo, J.-P. Talpin, T. Gautier, L. Besnard, , and P. Le Guernic. Verification of compiler transformations on polychronous equations. *Proceedings of IFM 2012, D. Latella and H. Treharne (Eds.)*, pages 113–127, 2012.

[12] A. Pnueli, O. Shtrichman, and M. Siegel. Translation validation for synchronous languages. *In Proceedings of ICALP'1998. Lecture Notes in Computer Science, Volume 1443*, pages 235–246, 1998.

[13] A. Pnueli, M. Siegel, and E. Singerman. Translation validation. *In Proceedings of TACAS'98, Lecture Notes in Computer Science*, 1384:151–166, 1998.

[14] M. Ryabtsev and O. Strichman. Translation validation: From simulink to c. *In Proceedings of the 21st International Conference on Computer Aided Verification (CAV 2009), volume 5643 of Lecture Notes in Computer Science*, pages 696–701, June 2009.