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

October 29, 2019

1 Introduction

......

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 Semantics of P3

Informal interpretation of P3 semantics. ⋯ (Based on some simple example)

⋯⋯

3 The target language

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⟩ ‘**)**’

⟨reg_offset⟩ ::= ⟨num⟩

⟨reg_size⟩ ::= ⟨num⟩

⟨seg_size⟩ ::= ⟨num⟩

⟨ins_size⟩ ::= ⟨num⟩

⟨num⟩ ::= *Integer* //integer constants, signed 32 bits
| *Hexadecimal* //hex constants, such as 0x88a8, 0xFFFFFF, 0x89,0x103

3.2 The configuration file format

⟨configuration⟩ ::= { ⟨layer_config⟩ }

⟨layer_con⟩ ::= ⟨layer_id⟩ ’:’ ⟨pb_lut⟩ ⟨pc_cur_lut⟩ ⟨pc_nxt_lut⟩

⟨layer_con⟩ ::= ⟨layer_id⟩ ’:’
⟨cella_pb_con⟩
⟨cella_pc_cur_con⟩
⟨cella_pc_nxt_con⟩
⟨cellb0_pb_con⟩
⟨cellb0_pc_cur_con⟩
⟨cellb1_pb_con⟩
⟨cellb1_pc_cur_con⟩

$\langle layer\_id \rangle$ ::= *IDENT*

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

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

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

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

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

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

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

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

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

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

$\langle cellb0\_pb\_con\_item \rangle$ ::= $\cdots$

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

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

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

3.3 Semantics

Informal interpretation of the semantics of the P3 assembly. $\cdots$ (Based on some simple example)

$\cdots\cdots$

4 Parsing

4.1 The P3 Abstract Syntax Tree

$\langle parser\_spec \rangle$ ::= *Parser* ($\langle layer\_reg\_len \rangle$, $\langle cell\_reg\_len \rangle$, $\langle protocol\_set \rangle$, $\langle layer\_set \rangle$, { $\langle decl \rangle$ })

$\langle layer\_reg\_len \rangle$ ::= *Lreglen* (*IntConst*(*Integer*))

$\langle cell\_reg\_len \rangle$ ::= *Creglen* (*IntConst*(*Integer*))

$\langle protocol\_set \rangle$ ::= *Pset* ($\langle id\_list \rangle$)

⟨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⟩ }

⟨p_stmt⟩ ::= ⟨if_else_p_stmt⟩
| NextHeader (IDENT)
| Length (⟨const⟩)
| Bypass (⟨const⟩)
| ⟨action_stmt⟩

⟨if_else_p_stmt⟩ ::= IfElseP({ ⟨if_branch_p⟩ } , ⟨default_branch_p⟩)

⟨if_branch_p⟩ ::= (⟨expr⟩, ⟨p_stmts⟩)

⟨default_branch_p⟩ ::= [⟨p_stmts⟩]

⟨layer_action⟩ ::= LayerAction (IDENT, ⟨local_reg_decl⟩, ⟨l_decls⟩ , ⟨l_actions⟩)

⟨l_decls⟩ ::= ⟨local_reg_decl⟩ { ⟨l_decl⟩ }

⟨l_decl⟩ ::= ProtocolDef (IDENT , ⟨id_list⟩)

⟨local_reg_decl⟩ ::= LocalRegs (⟨cella_regs⟩, ⟨cellb0_regs⟩, ⟨cellb1_regs⟩)

⟨cella_regs⟩ ::= CellARegs ({ ⟨reg_acc_set⟩ })

$\langle cellb0\_regs \rangle$ ::= *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* ({ $\langle if\_branch\_l \rangle$ } , $\langle default\_branch\_l \rangle$)

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

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

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

$\langle atom \rangle$ ::= *Econst*($\langle const \rangle$) //const expressions
| *IDENT* //all kinds of access name , ex., field or register access name

$\langle unop \rangle$ ::= *Oint* //convert hexadecimal or binary numbers to integers
| *Onot* //logical negation
| *Oneg* //bit-wise negation

$\langle binop \rangle$ ::= *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*⟩)

⟨*mov_reg_acc_name*⟩ ::= *MovRegAccName*(⟨*tgt_reg_acc_name*⟩)
| *MovRegAccName*(⟨*mov_reg_acc_name*⟩, ⟨*tgt_reg_acc_name*⟩)

4.2 Implementation and Verification

Construct a formally verified parser based on J.-H. Jourdan's method. $\cdots$

$\ldots\ldots$

5 Type Checking

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- 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

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

- 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

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

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

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

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

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

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

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

$$\frac{\begin{array}{c}\mathcal{G} \vdash \mathit{Creglen}(n) \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L_{B\infty}}' \vdash e : (\mathit{Int}, k) \qquad 0 \leq k < n \qquad id \notin dom(\mathcal{L_{B\infty}}') \\ \forall id' \in dom(\mathcal{L_{B\infty}}').(\mathcal{L_{B\infty}}' \vdash id' : \mathit{RegAcc}(n, n_1', n_2') \to n_1' < k \vee k < n_2') \\ \mathcal{L_{B\infty}} = \mathcal{L_{B\infty}}' \cup \{id : \mathit{RegAcc}(n, k, k)\}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L_{B\infty}} \vdash \mathit{IRF}(id, e)} \text{(ILB1-2)}$$

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

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

- Expressions

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_1 : \mathit{Bool} \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash e_2 : \mathit{Bool} \qquad \mathit{binop} \in \{\mathit{Oand}, \mathit{Oor}\}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_A,\mathcal{P} \vdash \mathit{Ebinop}(\mathit{binop}, e_1, e_2) : \mathit{Bool}} \text{ BOPL-2}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- Instructions

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

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

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

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

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

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

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

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

- 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

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

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

- Bypass statement

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

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

- 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

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

$$\frac{\begin{array}{c}\mathit{if\_l\_list} = ((e_1, \mathit{l\_stmts}_1), \cdots, (e_k, \mathit{l\_stmts}_k)) \\ \mathit{d\_l} = \mathit{l\_stmts} \qquad \forall i : 1 \leq i \leq k.\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash e_k : \mathit{Bool} \\ \forall i : 1 \leq i \leq k.\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{l\_stmts}_i \\ \mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{d\_l} \qquad \mathcal{L}_C\ \mathit{is}\ \mathcal{L}_A, \mathcal{L}_{B0}\ \mathit{or}\ \mathcal{L}_{B1}\end{array}}{\mathcal{G},\mathcal{C},\mathcal{R},\mathcal{L},\mathcal{L}_C \vdash \mathit{IfElseL}(\mathit{if\_l\_list}, \mathit{d\_l})}\ \text{IFEL}$$

- Layer local actions

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

- Layer local register declarations

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

- 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

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

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

$$\frac{\begin{array}{c}\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash (Fields(flds), OptionFields(oflds)) \\ flds = ((fld_1 : c_1), \cdots, (fld_k : c_k)) \\ \phi \vdash c_1 : (Int, n_1), \cdots, \phi \vdash c_k : (Int, n_k) \\ \phi \vdash e : (Int, n) \qquad n * 8 \geq n_1 + \cdots + n_k\end{array}}{\mathcal{G}, \mathcal{C}, \mathcal{R}, \mathcal{L}, \mathcal{L}_A, \mathcal{P} \vdash Length(e)} \text{ LENGTH-P}$$

- Protocol declaration

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

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

- Global declarations

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

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

- Parser Specification

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

5.2 Implementation and Verification

......

6 Translation

6.1 Translation of AST to the P3 Assembly

6.1.1 Abstract Syntax of the P3 assembly

$\langle parser\_asm \rangle$ ::= { $\langle const\_decl \rangle$ } { $\langle register\_decl \rangle$ } { $\langle layer\_block \rangle$ }

$\langle const\_decl \rangle$ ::= $ConstDcl(IDENT, \langle num \rangle)$

$\langle register\_decl \rangle$ ::= $Register(\ IDENT\ ,\ \langle num \rangle\ )$
| $PRegister(\ IDENT\ ,\ \langle num \rangle\ )$

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

$\langle layer\_id \rangle ::=$ *IDENT*

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

$\langle cella \rangle ::=$ *CellA*($\langle cella\_pb \rangle$, $\langle cella\_pc\_cur \rangle$, $\langle cella\_pc\_nxt \rangle$)

$\langle cellb0 \rangle ::=$ *CellB0*($\langle cellb0\_pb \rangle$, $\langle cellb0\_pc\_cur \rangle$)

$\langle cellb1 \rangle ::=$ *CellB0*($\langle cellb1\_pb \rangle$, $\langle cellb1\_pc\_cur \rangle$)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$\langle nxt\_id \rangle$ ::= $NxtID(\langle num \rangle)$

$\langle bypas \rangle$ ::= $Bypas(\langle num \rangle)$

$\langle lyr\_offset \rangle$ ::= $LyrOffset(\langle num \rangle)$

$\langle cella\_nxt \rangle$ ::= $CellANxt(\langle irf\_offsets \rangle, \langle prot\_offsets \rangle)$

$\langle cellb0\_nxt \rangle$ ::= $CellB0Nxt(\langle irf\_offsets \rangle, \langle prot\_offsets \rangle)$

$\langle cellb1\_nxt \rangle$ ::= $CellB1Nxt(\langle irf\_offsets \rangle, \langle prot\_offsets \rangle)$

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

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

$\langle cond \rangle$ ::= $(\langle reg\_seg \rangle, \langle num \rangle)$ $\quad | \ (\langle ins\_seg \rangle, \langle num \rangle)$

$\langle cmd \rangle$ ::= $\langle set\_cmd \rangle$
$\quad | \ \langle mov\_cmd \rangle$
$\quad | \ \langle lg\_cmd \rangle$
$\quad | \ \langle eq\_cmd \rangle$

$\langle set\_cmd \rangle$ ::= $Set(\langle reg\_seg \rangle, \langle num \rangle)$

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

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

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

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

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

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

$\langle reg\_offset \rangle$ ::= $\langle num \rangle$

$\langle reg\_size \rangle$ ::= $\langle num \rangle$

$\langle seg\_size \rangle$ ::= $\langle num \rangle$

$\langle ins\_size \rangle$ ::= $\langle num \rangle$

$\langle num \rangle$ ::= $Integer$ //integer constants, signed 32 bits

6.1.2 Translation to the AST of P3 Assembly

......

6.2 Translation of P3 Assembly to the Configuration File

$\cdots\cdots$

7 Verification

Refer to Section 4.2 and Section 5.2 for the verification of parser and type checker respectively.

7.1 Verification of the translation from AST to Assembly

$\cdots\cdots$

7.2 Semantics of the P3 AST

$\cdots\cdots$

7.2.1 Values and Memory model for Registers and Fields

$\cdots\cdots$

7.2.2 Semantic environment

The semantic environment associates to variables 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 2 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

$\cdots\cdots$

7.2.4 Semantic rules

- Common rules $\cdots\cdots$
- 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 :

Global	ge	$::=$	$(\gamma,\ \sigma,\ \delta)$	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	$::=$	$(\xi_\iota,\ nh,\ len,\ bp)$	divide layer local environment into five parts
	ξ_ι	$::=$	$id \rightarrow (len,\ (fid \rightarrow (n{,}bv)))$	map a protocol instance identifier to a protocol identifier, then a field identifier and then 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	$::=$	$(\delta_A,\ \delta_{B0},\ \delta_{B1})$	divide cell local environment into three parts
	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 1: Semantic Environments

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

SCR–Initialization of cr :

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

SPS–Initialization of ps :

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

SLS–Initialization of ls :

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

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 (Rules SIR-1 and SIR-2) and each time at the leaving of a layer context (Rule SIR-3) , and opened at the beginning of a layer context.

$$\frac{\begin{array}{c} ge = (\gamma, \sigma, \delta) \qquad ge \vdash e_1 \Rightarrow n_1 \qquad ge \vdash e_2 \Rightarrow n_2 \\ \gamma = (\mathit{lreglen}(n), cr, ps, ls, \iota, \rho) \qquad 0 \leq n_2 \leq n_1 < n \qquad id \notin dom(\delta) \\ \forall id' \in dom(\delta).(\delta \vdash id' \Rightarrow regacc(n, n'_1, n'_2, base\_layer\_bv_{id'}) \rightarrow n'_1 < n_2 \vee n_1 < n'_2) \\ \delta' = \delta \cup \{id : regacc(n, n_1, n_2, base\_layer\_bv_{id})\} \qquad ge' = (\gamma, \sigma, \delta') \end{array}}{\vdash (ge, IRF(id, e_1, e_2)) \Rightarrow ge'} \text{(SIR-1)}$$

$$\frac{\begin{array}{c} ge = (\gamma, \sigma, \delta) \qquad ge \vdash e \Rightarrow k \\ \gamma = (\mathit{lreglen}(n), cr, ps, ls, \iota, \rho) \qquad 0 \leq k < n \qquad id \notin dom(\delta) \\ \forall id' \in dom(\delta).(\delta \vdash id' \Rightarrow regacc(n, n'_1, n'_2, base\_layer\_bv_{id'}) \rightarrow n'_1 < k \vee k < n'_2) \\ \delta' = \delta \cup \{id : regacc(n, k, k, base\_layer\_bv_{id})\} \qquad ge' = (\gamma, \sigma, \delta') \end{array}}{\vdash (ge, IRF(id, e)) \Rightarrow ge'} \text{(SIR-2)}$$

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

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

$$\frac{\begin{array}{c} ge = (\gamma, \sigma, \delta) \qquad \gamma = (lr, cr, ps, ls, \iota, \rho) \\ le = (\xi_\iota, nh, len, bp) \qquad \rho \vdash pid \Rightarrow ((fid_1 : n_1, \cdots, fid_m : n_m), pstmts) \\ \forall i : 1 \leq i \leq k.\ id_i \notin dom(\xi_\iota) \\ \xi'_\iota = \xi_\iota \cup \{id_i : (length(null), pins_i) \mid 1 \leq i \leq k \ \wedge \ pins_i = ((fid_1, (n_1, bv^i_1)), \cdots, (fid_m, (n_m, bv^i_m)))\}, \\ \textit{where all bv's are the input from the hardware} \\ le' = (\xi'_\iota, null, null, null) \end{array}}{ge \vdash (le, ProtocolDef(pid, (id_1, \cdots, id_k))) \Rightarrow le'} \text{(SIL)}$$

- 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

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

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

- 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

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

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

- 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

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

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

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

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

- Expressions

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- Instructions

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

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

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

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

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

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

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

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

- 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

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

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

- Layer local actions

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

- Layer action

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

- Protocol statement

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

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

- Protocol declaration

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

- Global declarations

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

7.3 Semantics of the Assembly

7.3.1 Semantic environment

The semantic environment associates to variables 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 2 show all the semantic environments we use to define the semantics.

7.3.2 Judgements

$\cdots\cdots$

7.3.3 Semantic rules

- Common rules $\cdots\cdots$

7.4 Preserving the Semantics from AST to Assembly

$\cdots\cdots$

8 Conclusion

$\cdots\cdots$

Global	ge	$::=$	$(\sigma,\ \gamma,\ \delta,\ \iota)$	divide global environment into four parts
	σ	$::=$	$id \rightarrow int$	map a constant identifier to an integer
	γ	$::=$	$id \rightarrow regv(k,\ bv)$	map a register identifier to a local register memory for cell with the size k and the bits' value bv
	δ	$::=$	$id \rightarrow regv(k,\ bv)$	map a register identifier to a global register memory for the previous layer with the size k and the bits' value bv
	ι	$::=$	$id \rightarrow ldef$	map a layer identifier to a layer definition
Layer	le	$::=$	$(\rho,\ nh,\ len,\ bp)$	divide layer local environment into four parts
	ρ	$::=$	$id \rightarrow fdv(k,\ bv)$	map a protocol instance identifier to a memory for fields with the size k and the bits' value bv
	nh	$::=$	$nextheader(id)$	the *NextHeader* set to the protocol identified by id
	len	$::=$	$length(int)$	the *Length* bound to an integer
	bp	$::=$	$bypass(int)$	the *Bypass* bound to an integer
Cell	ce	$::=$	$(\delta_A,\ \delta_{B0},\ \delta_{B1})$	divide cell local environment into three parts
	δ_A	$::=$	$id \rightarrow regv(k,\ bv)$	map a register identifier to a register memory with the size k and the bits' value bv
	δ_{B0}	$::=$	$id \rightarrow regv(k,\ bv)$	map a register identifier to a register memory with the size k and the bits' value bv
	δ_{B1}	$::=$	$id \rightarrow regv(k,\ bv)$	map a register identifier to a register memory with the size k and the bits' value bv

Figure 2: Semantic Environments for the Assembly