## **IMP**

```
This is the symbolic semantics of IMP. It contains the normal semantics if IMP and the rules for symbolic execution, as
```

```
described in the technical report available here: https://fmse.info.uaic.ro/publications/156/.
MODULE IMP-SYNTAX
  SYNTAX AExp ::= Int
                      Id
                      AExp / AExp [strict]
                      AExp + AExp [strict]
                      (AExp) [bracket]
  SYNTAX BExp ::= Bool
                      AExp \le AExp [seqstrict]
                       ! BExp [strict]
                      BExp && BExp [strict(1)]
                      (BExp) [bracket]
  SYNTAX Block ::= \{\}
                    | \{Stmt\}
  SYNTAX Stmt ::= Block
                     Id = AExp; [strict(2)]
                      if (BExp)Block else Block [strict(1)]
                      while (BExp)Block
                     Stmt Stmt
  SYNTAX Pgm ::= int Ids ; Stmt
   SYNTAX Ids ::= List\{Id, ","\}
   Programs are \mathbb{K} configurations.
  SYNTAX Stmt ::= \#ps(Bag)
    Assertions syntax
   SYNTAX Stmt ::= assert (BExp) ; [strict]
   K compiler issues
  SYNTAX Dummy ::= symInt [dummySymInt]
  SYNTAX Int ::= #symInt (Id) [onlyLabel, klabel(#symInt)]
   SYNTAX Id ::= Token\{"a"\}
                   Token\{"b"\}
                   Token\{"c"\}
                   Token\{"sum"\}
                   Token\{"i"\}
                   Token\{"n"\}
                   Token\{"x"\}
                  | Token{"min"}
END MODULE
MODULE IMP
  SYNTAX KResult ::= Int
                       Bool
   IMP configuration is enriched with cell condition.
  CONFIGURATION:
                                                         condition
             #ps ($PGM:Bag)
                                                             true
    The concrete semantics of IMP which remains unchanged for symbolic execution
                           state
                            X\mapsto I
  RULE
  RULE I1:Int + I2:Int
            I1 +_{Int} I2
  Rule I1:Int \leq I2:Int
            I1 \leq_{Int} I2
          ! T:Bool
  RULE
           \neg_{Bool} T
  RULE
                                                                                                                                                                                                                                                                           [structural]
         \{S\}
                                                                                                                                                                                                                                                                           [structural]
                X = I:Int;
  RULE
          S1 S2
                                                                                                                                                                                                                                                                           [structural]
  RULE
          \overline{S1 \curvearrowright S2}
                      \quad \text{while } (B)S
                                                                                                                                                                                                                                                                           [structural]
  RULE
          if (B)\{S \text{ while } (B)S\} \text{ else } \{\}
   IMP symbolic semantics
                I1:Int / I2:Int
                                                Phi:Bool
                                                                           requires checkSat (Phi \wedge_{Bool} (I2 = /=_{Int} \mathbf{0})) \neq_{K} "unsat"
                                                                                                                                                                                                                                                                           [transition]
  RULE
                 I1 divInt I2
                                         Phi \wedge_{Bool} I2 = /=_{Int} 0
                                               condition
                B1:Bool && B2:BExp
                                                         Phi:Bool
                                                                                      requires checkSat (Phi \wedge_{Bool} B1) \neq_K "unsat"
                                                                                                                                                                                                                                                                           [transition]
  RULE
                                                Phi \wedge_{Bool} B1 == Bool true
                          B2
                                               condition
                B1:Bool && B2:BExp
                                                                              requires checkSat (Phi \wedge_{Bool} \neg_{Bool} B1) \neq_K "unsat"
                                                     Phi:Bool
                                                                                                                                                                                                                                                                           [transition]
  RULE
                         false
                                                Phi \wedge_{Bool} \neg_{Bool} B1
                                                 condition
                 if(B:Bool)S else —
                                                                         requires checkSat (Phi \wedge_{Bool} B) \neq_K "unsat"
  RULE
                                                   Phi:Bool
                                                                                                                                                                                                                                                                           [transition]
                                                  Phi \wedge_{Bool} B
                                                 condition
                 if (B:Bool)— else S
                                                      Phi:Bool
                                                                                                                                                                                                                                                                           [transition]
                                                                               requires checkSat (Phi \wedge_{Bool} \neg_{Bool} B) \neq_K "unsat"
  RULE
                                                 Phi \wedge_{Bool} \neg_{Bool} B
    Assert semantics: remain stuck when the assertion doesn' hold.
                                           condition
                 assert(B:Bool);
                                             Phi:Bool
  RULE
                                                                 requires checkSat (Phi \wedge_{Bool} B) =_K "sat"
   Load PGM.
                                     state
                                                       condition
                \#ps(B:Bag)
  RULE
                                                                                                                                                                                                                                                                           [structural]
```

END MODULE

 $\dot{B}$