```
⟦A⟧: V* → S → (S, $V_⊥$)$_⊥$
⟦I⟧: S → $S_⊥$
⟦C⟧: S → $B_⊥$
⟦E⟧: S → $V_⊥$
```

Algorithm

```
⟦a⟧(v*)(s) = pop_al_context(⟦i⟧(s’)) where
      a  = (e*, i)
      s  = (w*,        c’*, h)
      s’ = (w*,  c :: c’*, h)
      c  = (ρ, ⊥, 0)
      ρ  = bind(ℰ, e*, v*)
where
      pop_al_context(
             w*, (_, $v_⊥$, _) :: c’*, h
      ) = (w*, c’*, h), $v_⊥$
```

Instruction

```
⟦i1; i2⟧(s) = if $v_⊥$ = ⊥ then ⟦i2⟧(s’) else s’ where
      s’ = ⟦i1⟧(s)
         = (w*, c :: c’*, h)
      c  = (_, $v_⊥$, _)

⟦if c then i1 else i2⟧(s) = if ⟦c⟧(s) then ⟦i1⟧(s) else ⟦i2⟧(s)
⟦either i1 or i2⟧(s) = ⟦i1⟧(s)
⟦enter e1: e2 after i⟧(s) = cleanup(⟦i⟧(s’)) where
      v  = ⟦e1⟧(s)
      v* = ⟦e2⟧(s)
      s  = (      w*, c  :: c*, h)
      s’ = (w’ :: w*, c’ :: c*, h)
      c  = (ρ, $v_⊥$, n)
      c’ = (ρ, $v_⊥$, n + 1)
      w’ = (v, ℰ, v* ++ [ConstructorV(“end”, [])])
where
cleanup(s) = if n > 0 then cleanup($s_{res}$) else $s_{res}$ where
      s  = (w  :: w*, c :: c*, h)
      s’ = (w’ :: w*, c :: c*, h)
      w  = ($v_{ctx}$, $v_{val}$*, v :: $v_{instr}$*)
      w’ = ($v_{ctx}$, $v_{val}$*,      $v_{instr}$*)
      v  = ConstructV (str, v’*)
      $s_{res}$= ⟦lookup(p, str)⟧(v’*)(s’).1
         = ($w_{res}$*,     c :: c*, h)
      c  = (ρ, $v_⊥$, n)
⟦assert c⟧(s) = if ⟦c⟧(s) then s else ⊥
⟦push e⟧(s) = s’ where
      v  = ⟦e⟧(s)
      s  = (w  :: w*, c*, h)
      s’ = (w’ :: w*, c*, h)
      w  = (v1,      v2*, v3*)
      w’ = (v1, v :: v2*, v3*)
⟦pop e⟧(s) = s’ where
      s  = (w  :: w*, c  :: c*, h)
      s’ = (w’ :: w*, c’ :: c*, h)
      w  = (v1, v :: v2*, v3*)
      w’ = (v1,      v2*, v3*)
      c  = (ρ , $v_⊥$, n)
      c’ = (ρ’, $v_⊥$, n)
      ρ’ = bind(ρ, e, v)
⟦pop all e⟧(s) = s’ where
      s  = (w  :: w*, c  :: c*, h)
      s’ = (w’ :: w*, c’ :: c*, h)
      w  = (v1, v2*, v3*)
      w’ = (v1,   ℰ, v3*)
      c  = (ρ , $v_⊥$, n)
      c’ = (ρ’, $v_⊥$, n)
      ρ’ = bind(ρ, e, v2*)
⟦let e1 e2⟧(s) = s’ where
      v = ⟦e2⟧(s)
      s  = (w*, c  :: c*, h)
      s’ = (w*, c’ :: c*, h)
      c  = (ρ, $v_⊥$, n)
      c’ = (ρ’,$v_⊥$, n)
      ρ’ = bind(ρ, e1, v)
⟦trap⟧(s) = ⊥
⟦nop⟧(s) = s
⟦return e⟧(s) = s’ where
      v  = ⟦e⟧(s)
      s  = (w*, c  :: c*, h)
      s’ = (w*, c’ :: c*, h)
      c  = (ρ,  _, n)
      c’ = (ρ,  v, n)
⟦execute e⟧(s) = ⟦lookup(p,str)⟧(v*)(s).1 where
      v = ⟦e⟧(s)
        = ConstructV (str, v*)
⟦execute all e⟧(s) = $s_n$ where
      v = ⟦e⟧(s)
        = $v_0$ … $v_{n-1}$
      $v_i$ = ConstructV (str$_i$, $v_i$*)
      $s_0$ = s
      $s_{i+1}$ = ⟦lookup(p, str$_i$)⟧($v_i$*)($s_i$).1
⟦exit⟧(s) = s’ where
      s  = (w :: w’*, c* ,  h)
      s’ = (     w’*, c’*,  h)
      c’*= decrease_first_al_context_whose_number_is_nonzero(c*)
⟦ref x e⟧(s) = s’ where
      v = ⟦e2⟧(s)
      s  = (w*, c  :: c*, h)
      s’ = (w*, c’ :: c*, h’)
      c  = (ρ, $v_⊥$, n)
      c’ = (ρ’,$v_⊥$, n)
      h’, a = alloc(h, v)
      ρ’ = ρ + (x -> a)
⟦replace e1 [p*] with e2⟧(s) = s’ where
      s  = (w*, c  :: c*, h )
      s’ = (w*, c  :: c*, h’)
      c  = (ρ, _, _)
      a  = ⟦e1⟧(s)
      v  = ⟦e2⟧(s)
      v’ = replace(h(a), p*, v)
      h’ = h + (a -> v’)
where
replace(v1, [], v2) = v2
replace(v1, p :: p’*, v2) = v1 + (p -> replace(v1(p), p’*, v2))
```

Condition

```
⟦C⟧: S → $B_⊥$
⟦not c⟧(s) = ¬⟦c⟧(s)
⟦c1 ⊕ c2⟧(s) = ⟦c1⟧(s)⊕⟦c2⟧(s)
⟦e1 ⊙ e2⟧(s) = ⟦e1⟧(s)⊙⟦e2⟧(s)
⟦e is of case t⟧(s) = ⟦e⟧(s) == ConstructV(t, _)
```

Expression

```
⟦E⟧: S → $V_⊥$
⟦n⟧(s) = n
⟦t⟧(s) = t
⟦e1 ⊕ e2⟧(s) = ⟦e1⟧(s) ⊕ ⟦e2⟧(s)
⟦[e*]⟧(s) = l where
      n  = |e*|
      l(i) = ⟦e*[i]⟧(s) (for 0 <= i < n)
⟦e1^e2⟧(s) = l where
      v = ⟦e1⟧(s)
      n = ⟦e2⟧(s)
      l(i) = v (for 0 <= i < n)
⟦e1 ++ e2⟧(s) = l where
      l1, l2 = ⟦e1⟧(s), ⟦e2⟧(s)
      n1 = |l1|
      n2 = |l2|
      l[i] = if i < n1 then l1[i] else l2[i-n2](for 0 <= i < n1+n2)
⟦|e|⟧(s) = |⟦e⟧(s)|
⟦{(t -> e)*}⟧(s) = r where
      n = |t*| = |e*|
      r[t*[i]] = ⟦e*[i]⟧(s)
⟦e[p]⟧(s) = ⟦e⟧(s)[p]
⟦e1[p*] := e2⟧(s) = replace(⟦e1⟧(s), p*, ⟦e1⟧(s))
⟦e1[p*] :+ e2⟧(s) = append(⟦e1⟧(s), p*, ⟦e1⟧(s))
where
append(v1, [], v2) = v1 ++ v2
append(v1, p :: p’*, v2) = v1 + (p -> append(v1(p), p’*, v2))
⟦t(e*)⟧(s) = ConstructV(t, v*) where
      v*[i] = ⟦e*[i]⟧(s) (for 0 <= i < |e*|)
⟦(e1, e2)⟧(s) = (v1, v2) where
      v1, v2 = ⟦e1⟧(s), ⟦e2⟧(s)
⟦f(e*)⟧(s) = $v_⊥$ where
      v* = ⟦e*⟧(s)
      $v_⊥$ = ⟦lookup(p, f)⟧(v*)(s).2
⟦current context⟧(s) = v where
      s = (w :: _, _, _)
      w = (v, _, _)
⟦x⟧(s) = ρ(x) where
      s’ = (_, c :: _, _)
      c  = (ρ, _, _)
⟦e^{x*}⟧(s) = v where
      s  = (w*, c  :: c*, h)
      c  = (ρ , $v_⊥$, n)
      |ρ(x*[0])| = m
      $s_i$ = (w*, $c_i$ :: c*, h)        (for 0 <= i < m)
      $c_i$ = ($ρ_i$, $v_⊥$, n)            (for 0 <= i < m)
      $ρ_i$ = ρ + (x -> ρ(x)[i])*      (for 0 <= i < m)
      v[i] = ⟦e⟧($s_i$)              (for 0 <= i < m)
```