Static Type System for Information-Flow Control of JavaScript - Appendix

Yonathan Volpin

September 2020

1 Appendix:

1.1 Operational examples:

Each JS code is followed by JSL code and the Celf result.

```
var person = {};
person = {};
person.name = 1;
```

```
basic : cmd
  = newproperty λperson.
    newproperty λname.
    (; (var' !person !(#' nat low) !emptyobj)
    (; (# (assign' !person !emptyobj))
    (; (var (dot (ref global) person) name
                    !(#' nat low) (λthis. s z))
    skip))).

 #query 1000 ⊗ ⊗ 1 (prototype global nil -@
                    run basic empty ⊸ {result V}).
```

```
 Solution: \@X1. \X2. {
    let {[!f, X3]} = run/newproperty X2 in
    let {[!f_1, X4]} = run/newproperty X3 in
    let {X5} = run/; X4 in
    let {[!d1, [X6, X7]]} = run/var X5 in
    let {X8} = eval/ref X6 in
    let {[!d2, [X9, X10]]} = X7 !global X8 !nil @X1 in
    let {[!o, [@X11, X12]]} = eval/empty X9 in
    let {[@X13, [@X14, X15]]} = X10 !(ref !o) X12 in
    let {X16} = run/skip1 X15 in
    let {X17} = run/; X16 in
    let {[!d, [X18, X19]]} = run/# X17 in
    let {[!d1_1, [X20, X21]]} = eval/assign X18 in
    let {X22} = eval/ref X20 in
    let {[!d2_1, [X23, X24]]} = X21 !global X22 in
    let {[!o_1, [@X25, X26]]} = eval/empty X23 in
    let {[@X27, X28]} = X24 !(ref !o_1) X26 !(ref !o) @X13 in
    let {X29} = X19 !unit X28 in
    let {X30} = run/; X29 in
    let {[!d1_2, [X31, X32]]} = run/var X30 in
    let {[!d_1, [X33, X34]]} = eval/dot X31 in
    let {X35} = eval/ref X33 in
    let {[@X36, X37]} = X34 !global X35 !(ref !o_1) @X27 in
    let {[!d2_2, [X38, X39]]} = X32 !o_1 X37 !nil @X25 in
    let {[!d_2, [X40, X41]]} = eval/s X38 in
    let {X42} = eval/z X40 in
    let {X43} = X41 !z X42 in
    let {[@X44, [@X45, X46]]} = X39 !(s !z) X43 in
    let {X47} = run/skip1 X46 in
    let {X48} = run/skip2 X47 in X48}
 #V = unit
```

Example: Object update

```
function Person(x) {
   this.name = x;
   this.age = 3;
}
var alice = new Person(1);
var bob = new Person(5);
alice.name = 2;
alice.age = 4;
return(alice.name);

```

```
objectupdate: cmd
 = newproperty λperson.
   newproperty λname.
   newproperty λage.
   newproperty λalice.
   newproperty λbob.
   (; (function !(ref global) !person  !(#' rec low)
      !(λthis. fn λobj. return (fn λx.
            (; (var obj name !(#' nat low) (λthis. x))
            (; (var obj age !(#' nat low) (λthis.
                                      (s (s (s z)))))
      (return obj))))))
  (; (var' !alice !(#' nat low) !(new (dot' !person) (s z)))
  (; (var' !bob !(#' nat low) !(new (dot' !person)
                             (s (s (s (s (s z)))))))
  (; (# (assign (dot' !alice) name (s (s z))))
  (; (# (assign (dot' !alice) age (s (s (s (s z))))))
  (return (dot (dot' !alice) name))))))).

#query 1000 ⊗ ⊗ 1 (prototype global nil -@ run
                    objectupdate empty ⊸ {result V}).
```

```
Solution: \@X1. \X2. {
let {[!f, X3]} = run/newproperty X2 in
let {[!f_1, X4]} = run/newproperty X3 in
let {[!f_2, X5]} = run/newproperty X4 in
let {[!f_3, X6]} = run/newproperty X5 in
let {[!f_4, X7]} = run/newproperty X6 in
let {X8} = run/; X7 in
let {X9} = run/; X8 in
let {[!d1, [X10, X11]]} = run/var X9 in
let {X12} = eval/ref X10 in
let {[!d2, [X13, X14]]} = X11 !global X12 !nil @X1 in
let {[!o, [@X15, X16]]} = eval/empty X13 in
let {[@X17, [@X18, X19]]} = X14 !(ref !o) X16 in
let {X20} = run/skip1 X19 in
let {[!d1_1, [X21, X22]]} = run/var X20 in
let {[!d, [X23, X24]]} = eval/dot X21 in
let {X25} = eval/ref X23 in
let {[@X26, X27]} = X24 !global X25 !(ref !o) @X17 in
let {[!d2_1, [X28, X29]]} = X22 !o X27 !nil @X15 in
let {X30} = eval/fn X28 in
let {[@X31, [@X32, X33]]} = X29 !(fn !(\!X31. return
!(fn !(\!x. ; !(var !X31 !f_1 !(#’ !nat !low) !(\!this.
x)) !(; !(var !X31 !f_2 !(#’ !nat !low) !(\!this. s
!(s !(s !z)))) !(return !X31)))))) X30 in
let {X34} = run/skip1 X33 in
let {X35} = run/; X34 in
let {[!d1_2, [X36, X37]]} = run/var X35 in
let {X38} = eval/ref X36 in
let {[!d2_2, [X39, X40]]} = X37 !global X38
```

```
!(field !f !nil) @X18 in
let {[!d_1, [X41, X42]]} = eval/new X39 in
let {[!d_2, [X43, X44]]} = eval/dot X41 in
let {X45} = eval/ref X43 in
let {[@X46, X47]} = X44 !global X45 !(ref !o) @X26 in
let {[@X48, [!o_1, [@X49, X50]]]} = X42 !o X47
!(field !code !nil) @X32 in
let {[@X51, [@X52, X53]]} = copy/field X50 !(fn
!(\!X51. return !(fn !(\!x. ; !(var !X51 !f_1
!(#' !nat !low) !(\!this. x)) !(; !(var !X51 !f_2
!(#' !nat !low) !(\!this. s !(s !(s !z))))
!(return !X51)))))) @X31 in
let {X54} = copy/nil X53 in
let {[!d1_3, [X55, [!d2_3, [X56, X57]]]]} = eval/app X54 in
let {[!d1_4, [X58, [!d2_4, [X59, X60]]]]} = eval/app X55 in
let {X61} = eval/ref X59 in
let {[!d_3, [X62, X63]]} = eval/s X56 in
let {[!d_4, [X64, X65]]} = eval/dot X58 in
let {X66} = eval/z X62 in
let {X67} = eval/ref X64 in
let {X68} = X63 !z X66 in
let {[@X69, X70]} = X65 !o_1 X67 !(fn !(\!X69.
return !(fn !(\!x. ; !(var !X69 !f_1 !(#' !nat !low)
!(\!this. x)) !(; !(var !X69 !f_2 !(#' !nat !low)
!(\!this. s !(s !(s !z)))) !(return !X69))))))
@X52 in
let {[X71, X72]} = X60 !(\!X71. return !(fn !(\!x.
; !(var !X71 !f_1 !(#' !nat !low) !(\!this. x))
!(; !(var !X71 !f_2 !(#' !nat !low) !(\!this. s
!(s !(s !z)))) !(return !X71))))) X70 !(ref !o_1)
X61 in
let {[!d_5, [X73, X74]]} = run/return X71 in
let {X75} = eval/fn X73 in
let {X76} = X74 !(fn !(\!X76. ; !(var !(ref !o_1)
!f_1 !(#' !nat !low) !(\!this. X76)) !(; !(var
!(ref !o_1) !f_2 !(#' !nat !low) !(\!this. s
!(s !(s !z)))) !(return !(ref !o_1))))) X75 in
let {X77} = X72 !(fn !(\!X77. ; !(var !(ref !o_1)
!f_1 !(#' !nat !low) !(\!this. X77)) !(; !(var
!(ref !o_1) !f_2 !(#' !nat !low) !(\!this. s
!(s !(s !z)))) !(return !(ref !o_1))))) X76 in
let {[X78, X79]} = X57 !(\!X78. ; !(var !(ref
!o_1) !f_1 !(#' !nat !low) !(\!this. X78)) !(;
!(var !(ref !o_1) !f_2 !(#' !nat !low) !(\!this.
s !(s !(s !z)))) !(return !(ref !o_1)))) X77 !(s !z) X68 in
let {X80} = run/; X78 in
let {[!d1_5, [X81, X82]]} = run/var X80 in
let {X83} = eval/ref X81 in
let {[!d2_5, [X84, X85]]} = X82 !o_1 X83 !(field
!code !nil) @X49 in
let {[!d_6, [X86, X87]]} = eval/s X84 in
let {X88} = eval/z X86 in
let {X89} = X87 !z X88 in
let {[@X90, [@X91, X92]]} = X85 !(s !z) X89 in
let {X93} = run/skip1 X92 in
let {X94} = run/; X93 in
let {[!d1_6, [X95, X96]]} = run/var X94 in
let {X97} = eval/ref X95 in
let {[!d2_6, [X98, X99]]} = X96 !o_1 X97 !(field
!f_1 !(field !code !nil)) @X91 in
let {[!d_7, [X100, X101]]} = eval/s X98 in
let {[!d_8, [X102, X103]]} = eval/s X100 in
let {[!d_9, [X104, X105]]} = eval/s X102 in
let {X106} = eval/z X104 in
let {X107} = X105 !z X106 in
let {X108} = X103 !(s !z) X107 in
let {X109} = X101 !(s !(s !z)) X108 in
let {[@X110, [@X111, X112]]} = X99 !(s !(s
!(s !z))) X109 in
let {X113} = run/skip1 X112 in
let {[!d_10, [X114, X115]]} = run/return X113 in
let {X116} = eval/ref X114 in
let {X117} = X115 !(ref !o_1) X116 in
let {X118} = X79 !(ref !o_1) X117 in
let {[@X119, [@X120, X121]]} = X40 !(ref !o_1) X118 in
let {X122} = run/skip1 X121 in
let {X123} = run/; X122 in
let {[!d1_7, [X124, X125]]} = run/var X123 in
let {X126} = eval/ref X124 in
let {[!d2_7, [X127, X128]]} = X125 !global X126
!(field !f_3 !(field !f !nil)) @X120 in
let {[!d_11, [X129, X130]]} = eval/new X127 in
let {[!d_12, [X131, X132]]} = eval/dot X129 in
let {X133} = eval/ref X131 in
let {[@X134, X135]} = X132 !global X133 !(ref !o) @X46 in
let {[@X136, [!o_2, [@X137, X138]]]} = X130 !o X135
!(field !code !nil) @X48 in
let {[@X139, [@X140, X141]]} = copy/field X138 !(fn
!(\!X139. return !(fn !(\!x. ; !(var !X139 !f_1
!(#' !nat !low) !(\!this. x)) !(; !(var !X139
!f_2 !(#' !nat !low) !(\!this. s !(s !(s !z))))
!(return !X139)))))) @X51 in
```

```
let {X142} = copy/nil X141 in
let {[!d1_8, [X143, [!d2_8, [X144, X145]]]]} = eval/app X142 in
let {[!d1_9, [X146, [!d2_9, [X147, X148]]]]} = eval/app X143 in
let {X149} = eval/ref X147 in
let {[!d_13, [X150, X151]]} = eval/s X144 in
let {[!d_14, [X152, X153]]} = eval/s X150 in
let {[!d_15, [X154, X155]]} = eval/dot X146 in
let {[!d_16, [X156, X157]]} = eval/s X152 in
let {[!d_17, [X158, X159]]} = eval/s X156 in
let {X160} = eval/ref X154 in
let {[!d_18, [X161, X162]]} = eval/s X158 in
let {X163} = eval/z X161 in
let {X164} = X162 !z X163 in
let {[@X165, X166]} = X155 !o_2 X160 !(fn !(\!X165.
return !(fn !(\!x. ; !(var !X165 !f_1 !(#' !nat !low)
!(\!this. x)) !(; !(var !X165 !f_2 !(#' !nat !low)
!(\!this. s !(s !(s !z)))) !(return !X165)))))) @X140 in
let {[X167, X168]} = X148 !(\!X167. return !(fn !(\!x.
; !(var !X167 !f_1 !(#' !nat !low) !(\!this. x)) !(;
!(var !X167 !f_2 !(#' !nat !low) !(\!this. s !(s !(s
!z)))) !(return !X167))))) X166 !(ref !o_2) X149 in
let {X169} = X159 !(s !z) X164 in
let {X170} = X157 !(s !(s !z)) X169 in
let {X171} = X153 !(s !(s !(s !z))) X170 in
let {X172} = X151 !(s !(s !(s !(s !z)))) X171 in
let {[!d_19, [X173, X174]]} = run/return X167 in
let {X175} = eval/fn X173 in
let {X176} = X174 !(fn !(\!X176. ; !(var !(ref !o_2)
!f_1 !(#' !nat !low) !(\!this. X176)) !(; !(var !(ref
!o_2) !f_2 !(#' !nat !low) !(\!this. s !(s !(s !z))))
!(return !(ref !o_2))))) X175 in
let {X177} = X168 !(fn !(\!X177. ; !(var !(ref !o_2)
!f_1 !(#' !nat !low) !(\!this. X177)) !(; !(var !(ref
!o_2) !f_2 !(#' !nat !low) !(\!this. s !(s !(s !z))))
!(return !(ref !o_2))))) X176 in
let {[X178, X179]} = X145 !(\!X178. ; !(var !(ref !o_2)
!f_1 !(#' !nat !low) !(\!this. X178)) !(; !(var !(ref
!o_2) !f_2 !(#' !nat !low) !(\!this. s !(s !(s !z))))
!(return !(ref !o_2)))) X177 !(s !(s !(s !(s !(s !z))))) X172 in
let {X180} = run/; X178 in
let {[!d1_10, [X181, X182]]} = run/var X180 in
let {X183} = eval/ref X181 in
let {[!d2_10, [X184, X185]]} = X182 !o_2 X183 !(field
!code !nil) @X137 in
let {[!d_20, [X186, X187]]} = eval/s X184 in
let {[!d_21, [X188, X189]]} = eval/s X186 in
let {[!d_22, [X190, X191]]} = eval/s X188 in
let {[!d_23, [X192, X193]]} = eval/s X190 in
let {[!d_24, [X194, X195]]} = eval/s X192 in
let {X196} = eval/z X194 in
let {X197} = X195 !z X196 in
let {X198} = X193 !(s !z) X197 in
let {X199} = X191 !(s !(s !z)) X198 in
let {X200} = X189 !(s !(s !(s !z))) X199 in
let {X201} = X187 !(s !(s !(s !(s !z)))) X200 in
let {[@X202, [@X203, X204]]} = X185 !(s !(s !(s !(s
!(s !z))))) X201 in
let {X205} = run/skip1 X204 in
let {X206} = run/; X205 in
let {[!d1_11, [X207, X208]]} = run/var X206 in
let {X209} = eval/ref X207 in
let {[!d2_11, [X210, X211]]} = X208 !o_2 X209 !(field
!f_1 !(field !code !nil)) @X203 in
let {[!d_25, [X212, X213]]} = eval/s X210 in
let {[!d_26, [X214, X215]]} = eval/s X212 in
let {[!d_27, [X216, X217]]} = eval/s X214 in
let {X218} = eval/z X216 in
let {X219} = X217 !z X218 in
let {X220} = X215 !(s !z) X219 in
let {X221} = X213 !(s !(s !z)) X220 in
let {[@X222, [@X223, X224]]} = X211 !(s !(s !(s !z))) X221 in
let {X225} = run/skip1 X224 in
let {[!d_28, [X226, X227]]} = run/return X225 in
let {X228} = eval/ref X226 in
let {X229} = X227 !(ref !o_2) X228 in
let {X230} = X179 !(ref !o_2) X229 in
let {[@X231, [@X232, X233]]} = X128 !(ref !o_2) X230 in
let {X234} = run/skip1 X233 in
let {X235} = run/; X234 in
let {[!d_29, [X236, X237]]} = run/# X235 in
let {[!d1_12, [X238, X239]]} = eval/assign X236 in
let {[!d_30, [X240, X241]]} = eval/dot X238 in
let {X242} = eval/ref X240 in
let {[@X243, X244]} = X241 !global X242 !(ref !o_1) @X119 in
let {[!d2_12, [X245, X246]]} = X239 !o_1 X244 in
let {[!d_31, [X247, X248]]} = eval/s X245 in
let {[!d_32, [X249, X250]]} = eval/s X247 in
let {X251} = eval/z X249 in
let {X252} = X250 !z X251 in
let {X253} = X248 !(s !z) X252 in
let {[@X254, X255]} = X246 !(s !(s !z)) X253 !(s !z) @X90 in
let {X256} = X237 !unit X255 in
```

```
    let {X257} = run/; X256 in
    let {[!d_33, [X258, X259]]} = run/# X257 in
    let {[!d1_13, [X260, X261]]} = eval/assign X258 in
    let {[!d_34, [X262, X263]]} = eval/dot X260 in
    let {X264} = eval/ref X262 in
    let {[@X265, X266]} = X263 !global X264 !(ref !o_1) @X243 in
    let {[!d2_13, [X267, X268]]} = X261 !o_1 X266 in
    let {[!d_35, [X269, X270]]} = eval/s X267 in
    let {[!d_36, [X271, X272]]} = eval/s X269 in
    let {[!d_37, [X273, X274]]} = eval/s X271 in
    let {[!d_38, [X275, X276]]} = eval/s X273 in
    let {X277} = eval/z X275 in
    let {X278} = X276 !z X277 in
    let {X279} = X274 !(s !z) X278 in
    let {X280} = X272 !(s !(s !z)) X279 in
    let {X281} = X270 !(s !(s !(s !z))) X280 in
    let {[@X282, X283]} = X268 !(s !(s !(s !(s !z)))) X281
    !(s !(s !(s !z))) @X110 in
    let {X284} = X259 !unit X283 in
    let {[!d_39, [X285, X286]]} = run/return X284 in
    let {[!d_40, [X287, X288]]} = eval/dot X285 in
    let {[!d_41, [X289, X290]]} = eval/dot X287 in
    let {X291} = eval/ref X289 in
    let {[@X292, X293]} = X290 !global X291 !(ref !o_1) @X265 in
    let {[@X294, X295]} = X288 !o_1 X293 !(s !(s !z)) @X254 in
    let {X296} = X286 !(s !(s !z)) X295 in X296}
 #V = s !(s !z)
```

Example: Aliasing

```
function Person() {
  this.name = 2;
  this.age = 0;
}
var alice = new Person;
var bob = new Person;
alice.next = bob;
x = alice.next;
x.age = 5;
return(bob.age);

```

```
aliasing: cmd
 = newproperty λperson.
   newproperty λname.
```

```
  newproperty λage.
  newproperty λalice.
  newproperty λbob.
  newproperty λnext.
  newproperty λx.
  (; (function !(ref global) !person  !(#' rec low)
       !(λthis. fn λobj. return (fn λx.
             (; (var obj name !(#' nat low) (λthis. (s (s z))))
             (; (var obj age !(#' nat low) (λthis. z))
       (return obj))))))
  (; (var' !alice !(#' nat low) !(new (dot' !person) unit))
  (; (var' !bob !(#' nat low) !(new (dot' !person) unit))
  (; (var (dot' !alice) next !(#' nat low) (λthis. dot' !bob))
  (; (var' !x !(#' nat low) !(dot (dot' !alice) next))
  (; (# (assign (dot' !x) age (s (s (s (s (s z)))))))
  (return (dot (dot' !bob) age)))))))).

#query 1000 ⊗ ⊗ 1 (prototype global nil -@ run
                        aliasing empty ⊸ {result V}).
```

```
Solution: \@X1. \X2. {
    let {[!f, X3]} = run/newproperty X2 in
    let {[!f_1, X4]} = run/newproperty X3 in
    let {[!f_2, X5]} = run/newproperty X4 in
    let {[!f_3, X6]} = run/newproperty X5 in
    let {[!f_4, X7]} = run/newproperty X6 in
    let {[!f_5, X8]} = run/newproperty X7 in
    let {[!f_6, X9]} = run/newproperty X8 in
    let {X10} = run/; X9 in
    let {X11} = run/; X10 in
    let {[!d1, [X12, X13]]} = run/var X11 in
    let {X14} = eval/ref X12 in
    let {[!d2, [X15, X16]]} = X13 !global X14 !nil @X1 in
    let {[!o, [@X17, X18]]} = eval/empty X15 in
    let {[@X19, [@X20, X21]]} = X16 !(ref !o) X18 in
    let {X22} = run/skip1 X21 in
    let {[!d1_1, [X23, X24]]} = run/var X22 in
    let {[!d, [X25, X26]]} = eval/dot X23 in
    let {X27} = eval/ref X25 in
    let {[@X28, X29]} = X26 !global X27 !(ref !o) @X19 in
    let {[!d2_1, [X30, X31]]} = X24 !o X29 !nil @X17 in
    let {X32} = eval/fn X30 in
    let {[@X33, [@X34, X35]]} = X31 !(fn !(\!X33.
    return !(fn !(\!x. ; !(var !X33 !f_1 !(#’ !nat
    !low) !(\!this. s !(s !z))) !(; !(var !X33
    !f_2 !(#' !nat !low) !(\!this. z)) !(return !X33)))))) X32 in
    let {X36} = run/skip1 X35 in
    let {X37} = run/; X36 in
    let {[!d1_2, [X38, X39]]} = run/var X37 in
    let {X40} = eval/ref X38 in
    let {[!d2_2, [X41, X42]]} = X39 !global X40
    !(field !f !nil) @X20 in
    let {[!d_1, [X43, X44]]} = eval/new X41 in
    let {[!d_2, [X45, X46]]} = eval/dot X43 in
    let {X47} = eval/ref X45 in
    let {[@X48, X49]} = X46 !global X47 !(ref !o) @X28 in
    let {[@X50, [!o_1, [@X51, X52]]]} = X44 !o
    X49 !(field !code !nil) @X34 in
    let {[@X53, [@X54, X55]]} = copy/field X52
    !(fn !(\!X53. return !(fn !(\!x. ; !(var !X53
    !f_1 !(#' !nat !low) !(\!this. s !(s !z)))
    !(; !(var !X53 !f_2 !(#' !nat !low) !(\!this. z)) !(return
    !X53)))))) @X33 in
    let {X56} = copy/nil X55 in
    let {[!d1_3, [X57, [!d2_3, [X58, X59]]]]} = eval/app X56 in
    let {X60} = eval/unit X58 in
    let {[!d1_4, [X61, [!d2_4, [X62, X63]]]]} = eval/app X57 in
    let {X64} = eval/ref X62 in
    let {[!d_3, [X65, X66]]} = eval/dot X61 in
    let {X67} = eval/ref X65 in
    let {[@X68, X69]} = X66 !o_1 X67 !(fn !(\!X68.
    return !(fn !(\!x. ; !(var !X68 !f_1 !(#' !nat !low)
    !(\!this. s !(s !z))) !(; !(var !X68 !f_2
    !(#' !nat !low) !(\!this. z)) !(return !X68)))))) @X54 in
    let {[X70, X71]} = X63 !(\!X70. return !(fn
    !(\!x. ; !(var !X70 !f_1 !(#' !nat !low)
    !(\!this. s !(s !z))) !(; !(var !X70 !f_2
    !(#' !nat !low) !(\!this. z)) !(return !X70)))))
    X69 !(ref !o_1) X64 in
    let {[!d_4, [X72, X73]]} = run/return X70 in
    let {X74} = eval/fn X72 in
    let {X75} = X73 !(fn !(\!X75. ; !(var
    !(ref !o_1) !f_1 !(#' !nat !low) !(\!this.
    s !(s !z))) !(; !(var !(ref !o_1) !f_2
    !(#' !nat !low) !(\!this. z)) !(return
    !(ref !o_1))))) X74 in
    let {X76} = X71 !(fn !(\!X76. ; !(var !(ref
    !o_1) !f_1 !(#' !nat !low) !(\!this. s !(s !z)))
    !(; !(var !(ref !o_1) !f_2 !(#' !nat !low)
    !(\!this. z)) !(return !(ref !o_1))))) X75 in
```

```
let {[X77, X78]} = X59 !(\!X77. ; !(var !(ref
!o_1) !f_1 !(#' !nat !low) !(\!this. s !(s !z)))
!(; !(var !(ref !o_1) !f_2 !(#' !nat !low)
!(\!this. z)) !(return !(ref !o_1)))) X76 !unit X60 in
let {X79} = run/; X77 in
let {[!d1_5, [X80, X81]]} = run/var X79 in
let {X82} = eval/ref X80 in
let {[!d2_5, [X83, X84]]} = X81 !o_1 X82
!(field !code !nil) @X51 in
let {[!d_5, [X85, X86]]} = eval/s X83 in
let {[!d_6, [X87, X88]]} = eval/s X85 in
let {X89} = eval/z X87 in
let {X90} = X88 !z X89 in
let {X91} = X86 !(s !z) X90 in
let {[@X92, [@X93, X94]]} = X84 !(s !(s !z)) X91 in
let {X95} = run/skip1 X94 in
let {X96} = run/; X95 in
let {[!d1_6, [X97, X98]]} = run/var X96 in
let {X99} = eval/ref X97 in
let {[!d2_6, [X100, X101]]} = X98 !o_1 X99
!(field !f_1 !(field !code !nil)) @X93 in
let {X102} = eval/z X100 in
let {[@X103, [@X104, X105]]} = X101 !z X102 in
let {X106} = run/skip1 X105 in
let {[!d_7, [X107, X108]]} = run/return X106 in
let {X109} = eval/ref X107 in
let {X110} = X108 !(ref !o_1) X109 in
let {X111} = X78 !(ref !o_1) X110 in
let {[@X112, [@X113, X114]]} = X42 !(ref !o_1) X111 in
let {X115} = run/skip1 X114 in
let {X116} = run/; X115 in
let {[!d1_7, [X117, X118]]} = run/var X116 in
let {X119} = eval/ref X117 in
let {[!d2_7, [X120, X121]]} = X118 !global X119
!(field !f_3 !(field !f !nil)) @X113 in
let {[!d_8, [X122, X123]]} = eval/new X120 in
let {[!d_9, [X124, X125]]} = eval/dot X122 in
let {X126} = eval/ref X124 in
let {[@X127, X128]} = X125 !global X126
!(ref !o) @X48 in
let {[@X129, [!o_2, [@X130, X131]]]} = X123 !o
X128 !(field !code !nil) @X50 in
let {[@X132, [@X133, X134]]} = copy/field X131
!(fn !(\!X132. return !(fn !(\!x. ; !(var !X132
!f_1 !(#' !nat !low) !(\!this. s !(s !z))) !(;
!(var !X132 !f_2 !(#' !nat !low) !(\!this. z))
!(return !X132)))))) @X53 in
let {X135} = copy/nil X134 in
let {[!d1_8, [X136, [!d2_8, [X137, X138]]]]}
= eval/app X135 in
let {X139} = eval/unit X137 in
let {[!d1_9, [X140, [!d2_9, [X141, X142]]]]}
= eval/app X136 in
let {[!d_10, [X143, X144]]} = eval/dot X140 in
let {X145} = eval/ref X141 in
let {X146} = eval/ref X143 in
let {[@X147, X148]} = X144 !o_2 X146 !(fn
!(\!X147. return !(fn !(\!x. ; !(var !X147
!f_1 !(#' !nat !low) !(\!this. s !(s !z)))
!(; !(var !X147 !f_2 !(#' !nat !low)
!(\!this. z)) !(return !X147)))))) @X133 in
let {[X149, X150]} = X142 !(\!X149. return
!(fn !(\!x. ; !(var !X149 !f_1 !(#' !nat !low)
!(\!this. s !(s !z))) !(; !(var !X149 !f_2
!(#' !nat !low) !(\!this. z)) !(return
!X149))))) X148 !(ref !o_2) X145 in
let {[!d_11, [X151, X152]]} = run/return X149 in
let {X153} = eval/fn X151 in
let {X154} = X152 !(fn !(\!X154. ; !(var
!(ref !o_2) !f_1 !(#' !nat !low) !(\!this.
s !(s !z))) !(; !(var !(ref !o_2) !f_2
!(#' !nat !low) !(\!this. z)) !(return
!(ref !o_2))))) X153 in
let {X155} = X150 !(fn !(\!X155. ; !(var
!(ref !o_2) !f_1 !(#' !nat !low) !(\!this.
s !(s !z))) !(; !(var !(ref !o_2) !f_2
!(#' !nat !low) !(\!this. z)) !(return
!(ref !o_2))))) X154 in
let {[X156, X157]} = X138 !(\!X156. ;
!(var !(ref !o_2) !f_1 !(#' !nat !low)
!(\!this. s !(s !z))) !(; !(var !(ref !o_2)
!f_2 !(#' !nat !low) !(\!this. z)) !(return
!(ref !o_2)))) X155 !unit X139 in
let {X158} = run/; X156 in
let {[!d1_10, [X159, X160]]} = run/var X158 in
let {X161} = eval/ref X159 in
let {[!d2_10, [X162, X163]]} = X160 !o_2 X161
!(field !code !nil) @X130 in
let {[!d_12, [X164, X165]]} = eval/s X162 in
let {[!d_13, [X166, X167]]} = eval/s X164 in
let {X168} = eval/z X166 in
let {X169} = X167 !z X168 in
let {X170} = X165 !(s !z) X169 in
let {[@X171, [@X172, X173]]} = X163 !(s !(s !z)) X170 in
let {X174} = run/skip1 X173 in
let {X175} = run/; X174 in
let {[!d1_11, [X176, X177]]} = run/var X175 in
let {X178} = eval/ref X176 in
let {[!d2_11, [X179, X180]]} = X177 !o_2 X178
!(field !f_1 !(field !code !nil)) @X172 in
let {X181} = eval/z X179 in
let {[@X182, [@X183, X184]]} = X180 !z X181 in
let {X185} = run/skip1 X184 in
let {[!d_14, [X186, X187]]} = run/return X185 in
let {X188} = eval/ref X186 in
let {X189} = X187 !(ref !o_2) X188 in
let {X190} = X157 !(ref !o_2) X189 in
let {[@X191, [@X192, X193]]} = X121 !(ref !o_2) X190 in
let {X194} = run/skip1 X193 in
let {X195} = run/; X194 in
let {[!d1_12, [X196, X197]]} = run/var X195 in
let {[!d_15, [X198, X199]]} = eval/dot X196 in
let {X200} = eval/ref X198 in
let {[@X201, X202]} = X199 !global X200 !(ref !o_1) @X112 in
let {[!d2_12, [X203, X204]]} = X197 !o_1 X202
!(field !f_2 !(field !f_1 !(field !code !nil))) @X104 in
let {[!d_16, [X205, X206]]} = eval/dot X203 in
let {X207} = eval/ref X205 in
let {[@X208, X209]} = X206 !global X207 !(ref !o_2) @X191 in
let {[@X210, [@X211, X212]]} = X204 !(ref !o_2) X209 in
let {X213} = run/skip1 X212 in
let {X214} = run/; X213 in
let {[!d1_13, [X215, X216]]} = run/var X214 in
let {X217} = eval/ref X215 in
let {[!d2_13, [X218, X219]]} = X216 !global
X217 !(field !f_4 !(field !f_3 !(field !f
!nil))) @X192 in
let {[!d_17, [X220, X221]]} = eval/dot X218 in
let {[!d_18, [X222, X223]]} = eval/dot X220 in
let {X224} = eval/ref X222 in
let {[@X225, X226]} = X223 !global X224 !(ref !o_1) @X201 in
let {[@X227, X228]} = X221 !o_1 X226 !(ref !o_2) @X210 in
let {[@X229, [@X230, X231]]} = X219 !(ref !o_2) X228 in
let {X232} = run/skip1 X231 in
let {X233} = run/; X232 in
let {[!d_19, [X234, X235]]} = run/# X233 in
let {[!d1_14, [X236, X237]]} = eval/assign X234 in
let {[!d_20, [X238, X239]]} = eval/dot X236 in
```

```
    let {X240} = eval/ref X238 in
    let {[@X241, X242]} = X239 !global X240
    !(ref !o_2) @X229 in
    let {[!d2_14, [X243, X244]]} = X237 !o_2 X242 in
    let {[!d_21, [X245, X246]]} = eval/s X243 in
    let {[!d_22, [X247, X248]]} = eval/s X245 in
    let {[!d_23, [X249, X250]]} = eval/s X247 in
    let {[!d_24, [X251, X252]]} = eval/s X249 in
    let {[!d_25, [X253, X254]]} = eval/s X251 in
    let {X255} = eval/z X253 in
    let {X256} = X254 !z X255 in
    let {X257} = X252 !(s !z) X256 in
    let {X258} = X250 !(s !(s !z)) X257 in
    let {X259} = X248 !(s !(s !(s !z))) X258 in
    let {X260} = X246 !(s !(s !(s !(s !z)))) X259 in
    let {[@X261, X262]} = X244 !(s !(s !(s
    !(s !(s !z))))) X260 !z @X182 in
    let {X263} = X235 !unit X262 in
    let {[!d_26, [X264, X265]]} = run/return X263 in
    let {[!d_27, [X266, X267]]} = eval/dot X264 in
    let {[!d_28, [X268, X269]]} = eval/dot X266 in
    let {X270} = eval/ref X268 in
    let {[@X271, X272]} = X269 !global X270
    !(ref !o_2) @X208 in
    let {[@X273, X274]} = X267 !o_2 X272 !(s !(s
    !(s !(s !(s !z))))) @X261 in
    let {X275} = X265 !(s !(s !(s !(s !(s !z))))) X274 in X275}
 #V = s !(s !(s !(s !(s !z))))
```

Example: Function objects

```
function myfirstfunction(x) {
   return (x+1);
}
var result = 0;
result = myfirstfunction(1)
alert (result);

```

```
funcobject : cmd
 = newproperty λmyfirstfunction.
   newproperty λresult.
   (; (function !(ref global)
                !myfirstfunction  !(#' rec low)
        !(λthis. fn λx. return (s x)))
```

```
  (; (var' !result !(#' nat low) !z)
  (; (# (assign' !result !(fapp !(ref global)
                            !myfirstfunction !(s z))))
  (return (dot' !result))))).

#query 1000 ⊗ ⊗ 1 (prototype global nil -@ run
                   funcobject empty ⊸ {result V}).


```

```
Solution: \@X1. \X2. {
    let {[!f, X3]} = run/newproperty X2 in
    let {[!f_1, X4]} = run/newproperty X3 in
    let {X5} = run/; X4 in
    let {X6} = run/; X5 in
    let {[!d1, [X7, X8]]} = run/var X6 in
    let {X9} = eval/ref X7 in
    let {[!d2, [X10, X11]]} = X8 !global X9 !nil @X1 in
    let {[!o, [@X12, X13]]} = eval/empty X10 in
    let {[@X14, [@X15, X16]]} = X11 !(ref !o) X13 in
    let {X17} = run/skip1 X16 in
    let {[!d1_1, [X18, X19]]} = run/var X17 in
    let {[!d, [X20, X21]]} = eval/dot X18 in
    let {X22} = eval/ref X20 in
    let {[@X23, X24]} = X21 !global X22 !(ref !o) @X14 in
    let {[!d2_1, [X25, X26]]} = X19 !o X24 !nil @X12 in
    let {X27} = eval/fn X25 in
    let {[@X28, [@X29, X30]]} = X26 !(fn !(\!X28.
    return !(s !X28))) X27 in
    let {X31} = run/skip1 X30 in
    let {X32} = run/; X31 in
    let {[!d1_2, [X33, X34]]} = run/var X32 in
    let {X35} = eval/ref X33 in
    let {[!d2_2, [X36, X37]]} = X34 !global X35
    !(field !f !nil) @X15 in
    let {X38} = eval/z X36 in
    let {[@X39, [@X40, X41]]} = X37 !z X38 in
    let {X42} = run/skip1 X41 in
    let {X43} = run/; X42 in
    let {[!d_1, [X44, X45]]} = run/# X43 in
    let {[!d1_3, [X46, X47]]} = eval/assign X44 in
    let {X48} = eval/ref X46 in
    let {[!d2_3, [X49, X50]]} = X47 !global X48 in
    let {[!d1_4, [X51, [!d2_4, [X52, X53]]]]} =
    eval/app X49 in
    let {[!d_2, [X54, X55]]} = eval/dot X51 in
    let {[!d_3, [X56, X57]]} = eval/dot X54 in
    let {[!d_4, [X58, X59]]} = eval/s X52 in
    let {X60} = eval/ref X56 in
    let {X61} = eval/z X58 in
    let {[@X62, X63]} = X57 !global X60 !(ref !o) @X23 in
    let {X64} = X59 !z X61 in
    let {[@X65, X66]} = X55 !o X63 !(fn !(\!X65. return
    !(s !X65))) @X28 in
    let {[X67, X68]} = X53 !(\!X67. return !(s !X67))
    X66 !(s !z) X64 in
    let {[!d_5, [X69, X70]]} = run/return X67 in
    let {[!d_6, [X71, X72]]} = eval/s X69 in
    let {[!d_7, [X73, X74]]} = eval/s X71 in
    let {X75} = eval/z X73 in
    let {X76} = X74 !z X75 in
    let {X77} = X72 !(s !z) X76 in
    let {X78} = X70 !(s !(s !z)) X77 in
    let {X79} = X68 !(s !(s !z)) X78 in
    let {[@X80, X81]} = X50 !(s !(s !z)) X79 !z @X39 in
    let {X82} = X45 !unit X81 in
    let {[!d_8, [X83, X84]]} = run/return X82 in
    let {[!d_9, [X85, X86]]} = eval/dot X83 in
    let {X87} = eval/ref X85 in
    let {[@X88, X89]} = X86 !global X87 !(s !(s !z)) @X80 in
    let {X90} = X84 !(s !(s !z)) X89 in X90}
 #V = s !(s !z)
```

```
function Person(x) {
    this.next = true;
    this.name = x;
}
var alice = new Person(0);
var bob = new Person(1);
alice.next = bob;
bob.next = alice;
alert (alice.next.next.next.next.next.name);

```

```
selfref: cmd
 = newproperty λperson.
   newproperty λname.
   newproperty λalice.
```

```
  newproperty λbob.
  newproperty λnext.
  newproperty λresult.
  (; (function !(ref global) !person  !(#' rec low)
       !(λthis. fn λobj. return (fn λx.
              (; (var obj next !(#' nat low) (λthis. true))
              (; (var obj name !(#' nat low) (λthis. x))
        (return obj))))))
  (; (var' !alice !(#' nat low) !(new (dot' !person) z))
  (; (var' !bob !(#' nat low) !(new (dot' !person) (s z)))
  (; (# (assign (dot' !alice) next (dot' !bob)))
  (; (# (assign (dot' !bob) next (dot' !alice)))
  (return (dot  (dot (dot (dot (dot (dot (dot'
       !alice) next) next) next) next) next) name))))))).

#query 1000 ⊗ ⊗ 1 (prototype global nil -@ run
                       selfref empty ⊸ {result V}).
```

```
Solution: \@X1. \X2. {
let {[!f, X3]} = run/newproperty X2 in
let {[!f_1, X4]} = run/newproperty X3 in
let {[!f_2, X5]} = run/newproperty X4 in
let {[!f_3, X6]} = run/newproperty X5 in
let {[!f_4, X7]} = run/newproperty X6 in
let {[!f_5, X8]} = run/newproperty X7 in
let {X9} = run/; X8 in
let {X10} = run/; X9 in
let {[!d1, [X11, X12]]} = run/var X10 in
let {X13} = eval/ref X11 in
let {[!d2, [X14, X15]]} = X12 !global X13 !nil @X1 in
let {[!o, [@X16, X17]]} = eval/empty X14 in
let {[@X18, [@X19, X20]]} = X15 !(ref !o) X17 in
let {X21} = run/skip1 X20 in
let {[!d1_1, [X22, X23]]} = run/var X21 in
let {[!d, [X24, X25]]} = eval/dot X22 in
let {X26} = eval/ref X24 in
let {[@X27, X28]} = X25 !global X26 !(ref !o) @X18 in
let {[!d2_1, [X29, X30]]} = X23 !o X28 !nil @X16 in
let {X31} = eval/fn X29 in
let {[@X32, [@X33, X34]]} = X30 !(fn !(\!X32. return !(fn !(\!x. ;
!(var !X32 !f_4 !(#’ !nat !low) !(\!this. true)) !(; !(var !X32 !f_1
!(#’ !nat !low) !(\!this. x)) !(return !X32)))))) X31 in
let {X35} = run/skip1 X34 in
let {X36} = run/; X35 in
let {[!d1_2, [X37, X38]]} = run/var X36 in
let {X39} = eval/ref X37 in
let {[!d2_2, [X40, X41]]} = X38 !global X39 !(field !f !nil) @X19 in
let {[!d_1, [X42, X43]]} = eval/new X40 in
let {[!d_2, [X44, X45]]} = eval/dot X42 in
let {X46} = eval/ref X44 in
let {[@X47, X48]} = X45 !global X46 !(ref !o) @X27 in
let {[@X49, [!o_1, [@X50, X51]]]} = X43 !o X48 !(field !code !nil) @X33 in
let {[@X52, [@X53, X54]]} = copy/field X51 !(fn !(\!X52. return !(fn
!(\!x. ; !(var !X52 !f_4 !(#' !nat !low) !(\!this. true)) !(;
!(var !X52 !f_1 !(#' !nat !low) !(\!this. x)) !(return
!X52)))))) @X32 in
let {X55} = copy/nil X54 in
let {[!d1_3, [X56, [!d2_3, [X57, X58]]]]} = eval/app X55 in
let {X59} = eval/z X57 in
let {[!d1_4, [X60, [!d2_4, [X61, X62]]]]} = eval/app X56 in
let {X63} = eval/ref X61 in
let {[!d_3, [X64, X65]]} = eval/dot X60 in
let {X66} = eval/ref X64 in
let {[@X67, X68]} = X65 !o_1 X66 !(fn !(\!X67. return
!(fn !(\!x. ; !(var !X67 !f_4 !(#' !nat !low) !(\!this.
true)) !(; !(var !X67 !f_1 !(#' !nat !low) !(\!this. x))
!(return !X67)))))) @X53 in
let {[X69, X70]} = X62 !(\!X69. return !(fn !(\!x. ; !(var
!X69 !f_4 !(#' !nat !low) !(\!this. true)) !(; !(var !X69
!f_1 !(#' !nat !low) !(\!this. x)) !(return !X69))))) X68
!(ref !o_1) X63 in
let {[!d_4, [X71, X72]]} = run/return X69 in
let {X73} = eval/fn X71 in
let {X74} = X72 !(fn !(\!X74. ; !(var !(ref !o_1) !f_4
!(#' !nat !low) !(\!this. true)) !(; !(var !(ref !o_1)
!f_1 !(#' !nat !low) !(\!this. X74)) !(return !(ref !o_1)))))
X73 in
let {X75} = X70 !(fn !(\!X75. ; !(var !(ref !o_1) !f_4
!(#' !nat !low) !(\!this. true)) !(; !(var !(ref !o_1)
!f_1 !(#' !nat !low) !(\!this. X75)) !(return !(ref !o_1))))) X74 in
let {[X76, X77]} = X58 !(\!X76. ; !(var !(ref !o_1) !f_4 !
(#' !nat !low) !(\!this. true)) !(; !(var !(ref !o_1) !f_1
!(#' !nat !low) !(\!this. X76)) !(return !(ref !o_1)))) X75
!z X59 in
let {X78} = run/; X76 in
let {[!d1_5, [X79, X80]]} = run/var X78 in
let {X81} = eval/ref X79 in
let {[!d2_5, [X82, X83]]} = X80 !o_1 X81 !(field !code !nil) @X50 in
let {X84} = eval/true X82 in
let {[@X85, [@X86, X87]]} = X83 !true X84 in
let {X88} = run/skip1 X87 in
let {X89} = run/; X88 in
let {[!d1_6, [X90, X91]]} = run/var X89 in
let {X92} = eval/ref X90 in
let {[!d2_6, [X93, X94]]} = X91 !o_1 X92 !(field !f_4 !(field
!code !nil)) @X86 in
let {X95} = eval/z X93 in
let {[@X96, [@X97, X98]]} = X94 !z X95 in
let {X99} = run/skip1 X98 in
let {[!d_5, [X100, X101]]} = run/return X99 in
let {X102} = eval/ref X100 in
let {X103} = X101 !(ref !o_1) X102 in
let {X104} = X77 !(ref !o_1) X103 in
let {[@X105, [@X106, X107]]} = X41 !(ref !o_1) X104 in
let {X108} = run/skip1 X107 in
let {X109} = run/; X108 in
let {[!d1_7, [X110, X111]]} = run/var X109 in
let {X112} = eval/ref X110 in
let {[!d2_7, [X113, X114]]} = X111 !global X112 !(field !f_2 !(field
!f !nil)) @X106 in
let {[!d_6, [X115, X116]]} = eval/new X113 in
let {[!d_7, [X117, X118]]} = eval/dot X115 in
let {X119} = eval/ref X117 in
let {[@X120, X121]} = X118 !global X119 !(ref !o) @X47 in
let {[@X122, [!o_2, [@X123, X124]]]} = X116 !o X121 !(field !code
!nil) @X49 in
let {[@X125, [@X126, X127]]} = copy/field X124 !(fn !(\!X125. return
!(fn !(\!x. ; !(var !X125 !f_4 !(#' !nat !low) !(\!this. true))
!(; !(var !X125 !f_1 !(#' !nat !low)
!(\!this. x)) !(return !X125)))))) @X52 in
let {X128} = copy/nil X127 in
let {[!d1_8, [X129, [!d2_8, [X130, X131]]]]} = eval/app X128 in
let {[!d1_9, [X132, [!d2_9, [X133, X134]]]]} = eval/app X129 in
let {[!d_8, [X135, X136]]} = eval/dot X132 in
let {X137} = eval/ref X133 in
let {[!d_9, [X138, X139]]} = eval/s X130 in
let {X140} = eval/z X138 in
let {X141} = eval/ref X135 in
let {X142} = X139 !z X140 in
let {[@X143, X144]} = X136 !o_2 X141 !(fn !(\!X143. return
!(fn !(\!x. ; !(var !X143 !f_4 !(#' !nat !low) !(\!this.
true)) !(; !(var !X143 !f_1 !(#' !nat !low) !(\!this. x))
!(return !X143)))))) @X126 in
let {[X145, X146]} = X134 !(\!X145. return !(fn !(\!x. ;
!(var !X145 !f_4 !(#' !nat !low) !(\!this. true)) !(;
!(var !X145 !f_1 !(#' !nat !low) !(\!this. x)) !(return
!X145))))) X144 !(ref !o_2) X137 in
let {[!d_10, [X147, X148]]} = run/return X145 in
let {X149} = eval/fn X147 in
let {X150} = X148 !(fn !(\!X150. ; !(var !(ref !o_2) !f_4
!(#' !nat !low) !(\!this. true)) !(; !(var !(ref !o_2) !f_1
!(#' !nat !low) !(\!this. X150)) !(return !(ref !o_2))))) X149 in
let {X151} = X146 !(fn !(\!X151. ; !(var !(ref !o_2) !f_4
!(#' !nat !low) !(\!this. true)) !(; !(var !(ref !o_2) !f_1
!(#' !nat !low) !(\!this. X151)) !(return !(ref !o_2))))) X150 in
let {[X152, X153]} = X131 !(\!X152. ; !(var !(ref !o_2)
!f_4 !(#' !nat !low) !(\!this. true)) !(; !(var !(ref
!o_2) !f_1 !(#' !nat !low) !(\!this. X152)) !(return
!(ref !o_2)))) X151 !(s !z) X142 in
let {X154} = run/; X152 in
let {[!d1_10, [X155, X156]]} = run/var X154 in
let {X157} = eval/ref X155 in
let {[!d2_10, [X158, X159]]} = X156 !o_2 X157 !(field
!code !nil) @X123 in
let {X160} = eval/true X158 in
let {[@X161, [@X162, X163]]} = X159 !true X160 in
let {X164} = run/skip1 X163 in
let {X165} = run/; X164 in
let {[!d1_11, [X166, X167]]} = run/var X165 in
let {X168} = eval/ref X166 in
let {[!d2_11, [X169, X170]]} = X167 !o_2 X168 !(field
!f_4 !(field !code !nil)) @X162 in
let {[!d_11, [X171, X172]]} = eval/s X169 in
let {X173} = eval/z X171 in
let {X174} = X172 !z X173 in
let {[@X175, [@X176, X177]]} = X170 !(s !z) X174 in
let {X178} = run/skip1 X177 in
let {[!d_12, [X179, X180]]} = run/return X178 in
let {X181} = eval/ref X179 in
let {X182} = X180 !(ref !o_2) X181 in
let {X183} = X153 !(ref !o_2) X182 in
let {[@X184, [@X185, X186]]} = X114 !(ref !o_2) X183 in
let {X187} = run/skip1 X186 in
let {X188} = run/; X187 in
let {[!d_13, [X189, X190]]} = run/# X188 in
let {[!d1_12, [X191, X192]]} = eval/assign X189 in
let {[!d_14, [X193, X194]]} = eval/dot X191 in
let {X195} = eval/ref X193 in
let {[@X196, X197]} = X194 !global X195 !(ref !o_1) @X105 in
let {[!d2_12, [X198, X199]]} = X192 !o_1 X197 in
let {[!d_15, [X200, X201]]} = eval/dot X198 in
let {X202} = eval/ref X200 in
```

```
    let {[@X203, X204]} = X201 !global X202 !(ref !o_2) @X184 in
    let {[@X205, X206]} = X199 !(ref !o_2) X204 !true @X85 in
    let {X207} = X190 !unit X206 in
    let {X208} = run/; X207 in
    let {[!d_16, [X209, X210]]} = run/# X208 in
    let {[!d1_13, [X211, X212]]} = eval/assign X209 in
    let {[!d_17, [X213, X214]]} = eval/dot X211 in
    let {X215} = eval/ref X213 in
    let {[@X216, X217]} = X214 !global X215 !(ref !o_2) @X203 in
    let {[!d2_13, [X218, X219]]} = X212 !o_2 X217 in
    let {[!d_18, [X220, X221]]} = eval/dot X218 in
    let {X222} = eval/ref X220 in
    let {[@X223, X224]} = X221 !global X222 !(ref !o_1) @X196 in
    let {[@X225, X226]} = X219 !(ref !o_1) X224 !true @X161 in
    let {X227} = X210 !unit X226 in
    let {[!d_19, [X228, X229]]} = run/return X227 in
    let {[!d_20, [X230, X231]]} = eval/dot X228 in
    let {[!d_21, [X232, X233]]} = eval/dot X230 in
    let {[!d_22, [X234, X235]]} = eval/dot X232 in
    let {[!d_23, [X236, X237]]} = eval/dot X234 in
    let {[!d_24, [X238, X239]]} = eval/dot X236 in
    let {[!d_25, [X240, X241]]} = eval/dot X238 in
    let {[!d_26, [X242, X243]]} = eval/dot X240 in
    let {X244} = eval/ref X242 in
    let {[@X245, X246]} = X243 !global X244 !(ref !o_1) @X223 in
    let {[@X247, X248]} = X241 !o_1 X246 !(ref !o_2) @X205 in
    let {[@X249, X250]} = X239 !o_2 X248 !(ref !o_1) @X225 in
    let {[@X251, X252]} = X237 !o_1 X250 !(ref !o_2) @X247 in
    let {[@X253, X254]} = X235 !o_2 X252 !(ref !o_1) @X249 in
    let {[@X255, X256]} = X233 !o_1 X254 !(ref !o_2) @X251 in
    let {[@X257, X258]} = X231 !o_2 X256 !(s !z) @X175 in
    let {X259} = X229 !(s !z) X258 in X259}
 #V = s !z
```

Example: Control flow, while

```
var l;
var t;
var h;
l = 1;
t = 0;
while (h == 1) {
  t = 1;
}
while (t != 1) {
  t = 1;
}
return(t);

```

```
advwhile
  : cmd
  = newproperty λl.
    newproperty λt.
    newproperty λh.
    (; (var' !l !(#' nat low) !undef)
    (; (var' !t !(#' nat low) !undef)
    (; (var' !h !(#' nat low) !undef)
    (; (# (assign' !l !(s z)))
    (; (# (assign' !t !z))
    (; (while (== (dot' !h) (s z))
             (# (assign' !t !(s z))))
    (; (while (not (== (dot' !t) !(s z)))
             (# (assign' !t !(s z))))
    (return (dot' !t))))))))).

 #query 1000 ⊗ ⊗ 1 (prototype global nil -@ run
                        advwhile empty ⊸ {result V}).

```

```
Solution: \@X1. \X2. {
   let {[!f, X3]} = run/newproperty X2 in
   let {[!f_1, X4]} = run/newproperty X3 in
   let {[!f_2, X5]} = run/newproperty X4 in
   let {X6} = run/; X5 in
   let {[!d1, [X7, X8]]} = run/var X6 in
   let {X9} = eval/ref X7 in
   let {[!d2, [X10, X11]]} = X8 !global X9 !nil @X1 in
   let {X12} = eval/undef X10 in
   let {[@X13, [@X14, X15]]} = X11 !undef X12 in
   let {X16} = run/skip1 X15 in
   let {X17} = run/; X16 in
   let {[!d1_1, [X18, X19]]} = run/var X17 in
   let {X20} = eval/ref X18 in
   let {[!d2_1, [X21, X22]]} = X19 !global X20 !(field !f !nil) @X14 in
   let {X23} = eval/undef X21 in
   let {[@X24, [@X25, X26]]} = X22 !undef X23 in
   let {X27} = run/skip1 X26 in
   let {X28} = run/; X27 in
   let {[!d1_2, [X29, X30]]} = run/var X28 in
   let {X31} = eval/ref X29 in
   let {[!d2_2, [X32, X33]]} = X30 !global X31 !(field !f_1
   !(field !f !nil)) @X25 in
   let {X34} = eval/undef X32 in
   let {[@X35, [@X36, X37]]} = X33 !undef X34 in
   let {X38} = run/skip1 X37 in
   let {X39} = run/; X38 in
   let {[!d, [X40, X41]]} = run/# X39 in
   let {[!d1_3, [X42, X43]]} = eval/assign X40 in
   let {X44} = eval/ref X42 in
   let {[!d2_3, [X45, X46]]} = X43 !global X44 in
   let {[!d_1, [X47, X48]]} = eval/s X45 in
   let {X49} = eval/z X47 in
   let {X50} = X48 !z X49 in
   let {[@X51, X52]} = X46 !(s !z) X50 !undef @X13 in
   let {X53} = X41 !unit X52 in
   let {X54} = run/; X53 in
   let {[!d_2, [X55, X56]]} = run/# X54 in
   let {[!d1_4, [X57, X58]]} = eval/assign X55 in
   let {X59} = eval/ref X57 in
   let {[!d2_4, [X60, X61]]} = X58 !global X59 in
   let {X62} = eval/z X60 in
   let {[@X63, X64]} = X61 !z X62 !undef @X24 in
   let {X65} = X56 !unit X64 in
   let {X66} = run/; X65 in
   let {[!d_3, [X67, X68]]} = run/while X66 in
   let {[!d1_5, [X69, X70]]} = eval/== X67 in
   let {[!d_4, [X71, X72]]} = eval/dot X69 in
   let {X73} = eval/ref X71 in
   let {[@X74, X75]} = X72 !global X73 !undef @X35 in
   let {[!d2_5, [X76, X77]]} = X70 !undef X75 in
   let {[!d_5, [X78, X79]]} = eval/s X76 in
   let {X80} = eval/z X78 in
   let {X81} = X79 !z X80 in
   let {X82} = X77 !(s !z) X81 !false !equal/u1 in
   let {X83} = X68 !false X82 in
   let {X84} = run/exec'/false X83 in
   let {X85} = run/skip1 X84 in
   let {X86} = run/; X85 in
   let {[!d_6, [X87, X88]]} = run/while X86 in
   let {[!d_7, [X89, X90]]} = eval/not X87 in
   let {[!d1_6, [X91, X92]]} = eval/== X89 in
   let {[!d_8, [X93, X94]]} = eval/dot X91 in
   let {X95} = eval/ref X93 in
```

```
    let {[@X96, X97]} = X94 !global X95 !z @X63 in
    let {[!d2_6, [X98, X99]]} = X92 !z X97 in
    let {[!d_9, [X100, X101]]} = eval/s X98 in
    let {X102} = eval/z X100 in
    let {X103} = X101 !z X102 in
    let {X104} = X99 !(s !z) X103 !false !neq/z1 in
    let {X105} = X90 !false !true X104 !neg/false in
    let {X106} = X88 !true X105 in
    let {X107} = run/exec'/true X106 in
    let {[!d_10, [X108, X109]]} = run/# X107 in
    let {[!d1_7, [X110, X111]]} = eval/assign X108 in
    let {X112} = eval/ref X110 in
    let {[!d2_7, [X113, X114]]} = X111 !global X112 in
    let {[!d_11, [X115, X116]]} = eval/s X113 in
    let {X117} = eval/z X115 in
    let {X118} = X116 !z X117 in
    let {[@X119, X120]} = X114 !(s !z) X118 !z @X96 in
    let {X121} = X109 !unit X120 in
    let {[!d_12, [X122, X123]]} = run/while X121 in
    let {[!d_13, [X124, X125]]} = eval/not X122 in
    let {[!d1_8, [X126, X127]]} = eval/== X124 in
    let {[!d_14, [X128, X129]]} = eval/dot X126 in
    let {X130} = eval/ref X128 in
    let {[@X131, X132]} = X129 !global X130 !(s !z) @X119 in
    let {[!d2_8, [X133, X134]]} = X127 !(s !z) X132 in
    let {[!d_15, [X135, X136]]} = eval/s X133 in
    let {X137} = eval/z X135 in
    let {X138} = X136 !z X137 in
    let {X139} = X134 !(s !z) X138 !true !(equal/s !equal/z) in
    let {X140} = X125 !true !false X139 !neg/true in
    let {X141} = X123 !false X140 in
    let {X142} = run/exec'/false X141 in
    let {X143} = run/skip1 X142 in
    let {[!d_16, [X144, X145]]} = run/return X143 in
    let {[!d_17, [X146, X147]]} = eval/dot X144 in
    let {X148} = eval/ref X146 in
    let {[@X149, X150]} = X147 !global X148 !(s !z) @X131 in
    let {X151} = X145 !(s !z) X150 in X151}
 #V = s !z
```

Example: Recursively embedded functions

```
function ShowMessage(arg)
{
function SayHello(x) {
     return (x+2);
}
```

```
 return SayHello(arg+1);
}
return(ShowMessage(1));
```

```
showMessage : cmd
 = newproperty λshowmessage.
   newproperty λsayhello.
   (; (function !(ref global) !showmessage !(#' rec low)
        !(λthis. fn λarg.
      (; (function !(ref this) !sayhello !(#' rec low)
               !(λthis'. fn λx. return (s (s x))))
      (return (fapp !(ref this) !sayhello !(s arg))))))
   (return (fapp !(ref global) !showmessage !(s z)))).

 #query 1000 ⊗ ⊗ 1 (prototype global nil -@ run showMessage
                                           empty ⊸ {result V}).
```

```
 Solution: \@X1. \X2. {
   let {[!f, X3]} = run/newproperty X2 in
   let {[!f_1, X4]} = run/newproperty X3 in
   let {X5} = run/; X4 in
   let {X6} = run/; X5 in
   let {[!d1, [X7, X8]]} = run/var X6 in
   let {X9} = eval/ref X7 in
   let {[!d2, [X10, X11]]} = X8 !global X9 !nil @X1 in
   let {[!o, [@X12, X13]]} = eval/empty X10 in
   let {[@X14, [@X15, X16]]} = X11 !(ref !o) X13 in
   let {X17} = run/skip1 X16 in
   let {[!d1_1, [X18, X19]]} = run/var X17 in
   let {[!d, [X20, X21]]} = eval/dot X18 in
   let {X22} = eval/ref X20 in
   let {[@X23, X24]} = X21 !global X22 !(ref !o) @X14 in
   let {[!d2_1, [X25, X26]]} = X19 !o X24 !nil @X12 in
   let {X27} = eval/fn X25 in
   let {[@X28, [@X29, X30]]} = X26 !(fn !(\!X28. ; !(; !(var !(ref !o)
   !f_1 !(#’ !nat !low) !(\!this. emptyobj)) !(var !(dot !(ref
   !o) !f_1) !code !(#’ !rec !low) !(\!X29.
   fn !(\!x. return !(s !(s !x)))))) !(return !(app !(dot !(dot
   !(ref !o) !f_1) !code) !(s !X28))))) X27 in
   let {X31} = run/skip1 X30 in
```

```
let {[!d_1, [X32, X33]]} = run/return X31 in
let {[!d1_2, [X34, [!d2_2, [X35, X36]]]]} = eval/app X32 in
let {[!d_2, [X37, X38]]} = eval/s X35 in
let {[!d_3, [X39, X40]]} = eval/dot X34 in
let {X41} = eval/z X37 in
let {X42} = X38 !z X41 in
let {[!d_4, [X43, X44]]} = eval/dot X39 in
let {X45} = eval/ref X43 in
let {[@X46, X47]} = X44 !global X45 !(ref !o) @X23 in
let {[@X48, X49]} = X40 !o X47 !(fn !(\!X48. ; !(; !(var !(ref
!o) !f_1 !(#' !nat !low) !(\!this. emptyobj)) !(var !(dot !(ref
!o) !f_1) !code !(#' !rec !low) !(\!X49.
fn !(\!x. return !(s !(s !x)))))) !(return !(app !(dot !(dot
!(ref !o) !f_1) !code) !(s !X48))))) @X28 in
let {[X50, X51]} = X36 !(\!X50. ; !(; !(var !(ref !o) !f_1
!(#' !nat !low) !(\!this. emptyobj)) !(var !(dot !(ref !o)
!f_1) !code !(#' !rec !low) !(\!X51. fn
!(\!x. return !(s !(s !x)))))) !(return !(app !(dot !(dot
!(ref !o) !f_1) !code) !(s !X50)))) X49 !(s !z)
X42 in
let {X52} = run/; X50 in
let {X53} = run/; X52 in
let {[!d1_3, [X54, X55]]} = run/var X53 in
let {X56} = eval/ref X54 in
let {[!d2_3, [X57, X58]]} = X55 !o X56 !(field !code !nil) @X29 in
let {[!o_1, [@X59, X60]]} = eval/empty X57 in
let {[@X61, [@X62, X63]]} = X58 !(ref !o_1) X60 in
let {X64} = run/skip1 X63 in
let {[!d1_4, [X65, X66]]} = run/var X64 in
let {[!d_5, [X67, X68]]} = eval/dot X65 in
let {X69} = eval/ref X67 in
let {[@X70, X71]} = X68 !o X69 !(ref !o_1) @X61 in
let {[!d2_4, [X72, X73]]} = X66 !o_1 X71 !nil @X59 in
let {X74} = eval/fn X72 in
let {[@X75, [@X76, X77]]} = X73 !(fn !(\!X75. return !(s
!(s !X75)))) X74 in
let {X78} = run/skip1 X77 in
let {[!d_6, [X79, X80]]} = run/return X78 in
let {[!d1_5, [X81, [!d2_5, [X82, X83]]]]} = eval/app X79 in
let {[!d_7, [X84, X85]]} = eval/s X82 in
let {[!d_8, [X86, X87]]} = eval/s X84 in
let {X88} = eval/z X86 in
let {[!d_9, [X89, X90]]} = eval/dot X81 in
let {[!d_10, [X91, X92]]} = eval/dot X89 in
let {X93} = X87 !z X88 in
let {X94} = eval/ref X91 in
```

```
    let {[@X95, X96]} = X92 !o X94 !(ref !o_1) @X70 in
    let {[@X97, X98]} = X90 !o_1 X96 !(fn !(\!X97. return !(s
    !(s !X97)))) @X75 in
    let {X99} = X85 !(s !z) X93 in
    let {[X100, X101]} = X83 !(\!X100. return !(s !(s !X100)))
    X98 !(s !(s !z)) X99 in
    let {[!d_11, [X102, X103]]} = run/return X100 in
    let {[!d_12, [X104, X105]]} = eval/s X102 in
    let {[!d_13, [X106, X107]]} = eval/s X104 in
    let {[!d_14, [X108, X109]]} = eval/s X106 in
    let {[!d_15, [X110, X111]]} = eval/s X108 in
    let {X112} = eval/z X110 in
    let {X113} = X111 !z X112 in
    let {X114} = X109 !(s !z) X113 in
    let {X115} = X107 !(s !(s !z)) X114 in
    let {X116} = X105 !(s !(s !(s !z))) X115 in
    let {X117} = X103 !(s !(s !(s !(s !z)))) X116 in
    let {X118} = X101 !(s !(s !(s !(s !z)))) X117 in
    let {X119} = X80 !(s !(s !(s !(s !z)))) X118 in
    let {X120} = X51 !(s !(s !(s !(s !z)))) X119 in
    let {X121} = X33 !(s !(s !(s !(s !z)))) X120 in X121}
 #V = s !(s !(s !(s !z)))
```

Example: Dynamic extension of objects with fields and methods

```
const safe = {
  data: 3,
  show: function() {
 return (this.data);
  }
};

safe.cipher = 0;
safe.encrypt = function() {
  this.cipher = 1+this.data;
};
safe.decrypt = function() {
  this.cipher = this.cipher-1;
  return (this.cipher);
};

safe.encrypt();

return(safe.decrypt());
```

```
advobject : cmd
 = newproperty λsafe.
   newproperty λdata.
   newproperty λshow.
   newproperty λcipher.
   newproperty λencrypt.
   newproperty λdecrypt.
   newproperty λresult.
   (; (var' !safe !(#' nat low) !emptyobj)
   (; (var (dot' !safe) data !(#' nat low) (λthis. s (s (s z))))
   (; (var (dot' !safe) show !(#' nat low)
                (λthis.fn λx. return (s (dot (ref this) data))))
   (; (var (dot' !safe) cipher !(#' nat low) (λthis. s (s z)))
   (; (var (dot' !safe) encrypt !(#' nat low)
            (λthis. fn λx. ;
         (# (assign (ref this) cipher (s (dot (ref this) data)))
                  (return unit)))
   (; (var (dot' !safe) decrypt !(#' nat low)
            (λthis. fn λx.  return (dec (dot (ref this) cipher))))
   (; (# (app (dot (dot' !safe) encrypt) unit))
   (return (app (dot (dot' !safe) decrypt) unit)))))))))).

 #query 1000 ⊗ ⊗ 1 (prototype global nil -@ run
                           advobject empty ⊸ {result V}).
```

```
Solution: \@X1. \X2. {
   let {[!f, X3]} = run/newproperty X2 in
   let {[!f_1, X4]} = run/newproperty X3 in
   let {[!f_2, X5]} = run/newproperty X4 in
   let {[!f_3, X6]} = run/newproperty X5 in
   let {[!f_4, X7]} = run/newproperty X6 in
   let {[!f_5, X8]} = run/newproperty X7 in
   let {[!f_6, X9]} = run/newproperty X8 in
   let {X10} = run/; X9 in
   let {[!d1, [X11, X12]]} = run/var X10 in
   let {X13} = eval/ref X11 in
   let {[!d2, [X14, X15]]} = X12 !global X13 !nil @X1 in
   let {[!o, [@X16, X17]]} = eval/empty X14 in
   let {[@X18, [@X19, X20]]} = X15 !(ref !o) X17 in
   let {X21} = run/skip1 X20 in
   let {X22} = run/; X21 in
   let {[!d1_1, [X23, X24]]} = run/var X22 in
   let {[!d, [X25, X26]]} = eval/dot X23 in
   let {X27} = eval/ref X25 in
   let {[@X28, X29]} = X26 !global X27 !(ref !o) @X18 in
   let {[!d2_1, [X30, X31]]} = X24 !o X29 !nil @X16 in
   let {[!d_1, [X32, X33]]} = eval/s X30 in
   let {[!d_2, [X34, X35]]} = eval/s X32 in
   let {[!d_3, [X36, X37]]} = eval/s X34 in
   let {X38} = eval/z X36 in
   let {X39} = X37 !z X38 in
   let {X40} = X35 !(s !z) X39 in
   let {X41} = X33 !(s !(s !z)) X40 in
   let {[@X42, [@X43, X44]]} = X31 !(s !(s !(s !z))) X41 in
   let {X45} = run/skip1 X44 in
   let {X46} = run/; X45 in
   let {[!d1_2, [X47, X48]]} = run/var X46 in
   let {[!d_4, [X49, X50]]} = eval/dot X47 in
   let {X51} = eval/ref X49 in
   let {[@X52, X53]} = X50 !global X51 !(ref !o) @X28 in
   let {[!d2_2, [X54, X55]]} = X48 !o X53 !(field !f_1 !nil) @X43 in
   let {X56} = eval/fn X54 in
   let {[@X57, [@X58, X59]]} = X55 !(fn !(\!X57. return !(s !(dot !(ref !o)
   !f_1)))) X56 in
   let {X60} = run/skip1 X59 in
   let {X61} = run/; X60 in
   let {[!d1_3, [X62, X63]]} = run/var X61 in
   let {[!d_5, [X64, X65]]} = eval/dot X62 in
   let {X66} = eval/ref X64 in
   let {[@X67, X68]} = X65 !global X66 !(ref !o) @X52 in
   let {[!d2_3, [X69, X70]]} = X63 !o X68 !(field !f_2 !(field !f_1 !nil)) @X58 in
   let {[!d_6, [X71, X72]]} = eval/s X69 in
   let {[!d_7, [X73, X74]]} = eval/s X71 in
   let {X75} = eval/z X73 in
   let {X76} = X74 !z X75 in
   let {X77} = X72 !(s !z) X76 in
   let {[@X78, [@X79, X80]]} = X70 !(s !(s !z)) X77 in
   let {X81} = run/skip1 X80 in
   let {X82} = run/; X81 in
   let {[!d1_4, [X83, X84]]} = run/var X82 in
   let {[!d_8, [X85, X86]]} = eval/dot X83 in
   let {X87} = eval/ref X85 in
   let {[@X88, X89]} = X86 !global X87 !(ref !o) @X67 in
   let {[!d2_4, [X90, X91]]} = X84 !o X89 !(field !f_3 !(field !f_2 !(field !f_1
   !nil))) @X79 in
   let {X92} = eval/fn X90 in
   let {[@X93, [@X94, X95]]} = X91 !(fn !(\!X93. ; !(# !(assign !(ref !o) !f_3
   !(s !(dot !(ref !o) !f_1)))) !(return !unit))) X92 in
```

```
let {X96} = run/skip1 X95 in
let {X97} = run/; X96 in
let {[!d1_5, [X98, X99]]} = run/var X97 in
let {[!d_9, [X100, X101]]} = eval/dot X98 in
let {X102} = eval/ref X100 in
let {[@X103, X104]} = X101 !global X102 !(ref !o) @X88 in
let {[!d2_5, [X105, X106]]} = X99 !o X104 !(field !f_4 !(field !f_3 !(field
!f_2 !(field !f_1 !nil)))) @X94 in
let {X107} = eval/fn X105 in
let {[@X108, [@X109, X110]]} = X106 !(fn !(\!X108. return !(dec !(dot !(ref
!o) !f_3)))) X107 in
let {X111} = run/skip1 X110 in
let {X112} = run/; X111 in
let {[!d_10, [X113, X114]]} = run/# X112 in
let {[!d1_6, [X115, [!d2_6, [X116, X117]]]]} = eval/app X113 in
let {[!d_11, [X118, X119]]} = eval/dot X115 in
let {[!d_12, [X120, X121]]} = eval/dot X118 in
let {X122} = eval/ref X120 in
let {[@X123, X124]} = X121 !global X122 !(ref !o) @X103 in
let {[@X125, X126]} = X119 !o X124 !(fn !(\!X125. ; !(# !(assign !(ref !o)
!f_3 !(s !(dot !(ref !o) !f_1)))) !(return !unit))) @X93 in
let {X127} = eval/unit X116 in
let {[X128, X129]} = X117 !(\!X128. ; !(# !(assign !(ref !o) !f_3 !(s !(dot
!(ref !o) !f_1)))) !(return !unit)) X126 !unit X127 in
let {X130} = run/; X128 in
let {[!d_13, [X131, X132]]} = run/# X130 in
let {[!d1_7, [X133, X134]]} = eval/assign X131 in
let {X135} = eval/ref X133 in
let {[!d2_7, [X136, X137]]} = X134 !o X135 in
let {[!d_14, [X138, X139]]} = eval/s X136 in
let {[!d_15, [X140, X141]]} = eval/dot X138 in
let {X142} = eval/ref X140 in
let {[@X143, X144]} = X141 !o X142 !(s !(s !(s !z))) @X42 in
let {X145} = X139 !(s !(s !(s !z))) X144 in
let {[@X146, X147]} = X137 !(s !(s !(s !(s !z)))) X145 !(s !(s !z)) @X78 in
let {X148} = X132 !unit X147 in
let {[!d_16, [X149, X150]]} = run/return X148 in
let {X151} = eval/unit X149 in
let {X152} = X150 !unit X151 in
let {X153} = X129 !unit X152 in
let {X154} = X114 !unit X153 in
let {[!d_17, [X155, X156]]} = run/return X154 in
let {[!d1_8, [X157, [!d2_8, [X158, X159]]]]} = eval/app X155 in
let {[!d_18, [X160, X161]]} = eval/dot X157 in
let {[!d_19, [X162, X163]]} = eval/dot X160 in
let {X164} = eval/ref X162 in
```

```
   let {[@X165, X166]} = X163 !global X164 !(ref !o) @X123 in
   let {X167} = eval/unit X158 in
   let {[@X168, X169]} = X161 !o X166 !(fn !(\!X168. return !(dec !(dot
   !(ref !o) !f_3)))) @X108 in
   let {[X170, X171]} = X159 !(\!X170. return !(dec !(dot !(ref !o) !f_3))) X169
   !unit X167 in
   let {[!d_20, [X172, X173]]} = run/return X170 in
   let {[!d_21, [X174, X175]]} = eval/dec X172 in
   let {[!d_22, [X176, X177]]} = eval/dot X174 in
   let {X178} = eval/ref X176 in
   let {[@X179, X180]} = X177 !o X178 !(s !(s !(s !(s !z)))) @X146 in
   let {X181} = X175 !(s !(s !(s !z))) X180 in
   let {X182} = X173 !(s !(s !(s !z))) X181 in
   let {X183} = X171 !(s !(s !(s !z))) X182 in
   let {X184} = X156 !(s !(s !(s !z))) X183 in X184}
#V = s !(s !(s !z))
```