Appendix B: Implementation Showcase

of the doctoral thesis: Unfolding semantics of the untyped λ-calculus with letrec

Jan Rochel <jan@rochel.info>

June 1, 2016

To demonstrate the realisability of our method, and for further illustration, we include the output of our implementation for some examples from the thesis. The implementation is called maxsharing and is available on Hackage. It is written in Haskell and therefore requires the Haskell Platform to be installed. Then, maxsharing can be installed via cabal-install using the commands `cabal update` and `cabal install maxsharing` from the terminal. Invoke the executable `maxsharing` in your cabal-directory with a file as an argument that contains a $\lambda_{\mathsf{letrec}}$-term. Run `maxsharing -h` for help on run-time flags.

Consider the following explanations for understanding the tool's output.

λ-letrec-term: The original term as recognised by the parser

translation used: The user can specify which term graph semantics shall be used in processing the term. All further output is with respect to that translation. The two options are:

- $[\![\cdot]\!]_{\mathcal{T}}$ is called: `maximal prefix lengths but eager scope`
- $[\![\cdot]\!]_{\mathcal{T}}^{\min}$ is called: `minimal prefix lengths`.

scoped (adbmals): The term enriched by abdmals. The adbmal ($\curlywedge$) is to be read as a scope delimiter that explicitly includes the name of the λ-variable whose scope it delimits. The adbmals are placed in accordance to the translation used.

scoped (nameless): A nameless scoped representation, where the names of variables are omitted for abstractions as well as for abstraction variable occurrences, shown as a 0-symbol. The scoping is expressed by scope-delimiters in the shape of an S-symbol. Note that these terms can be obtained by a simple syntactical transformation of the adbmal-terms.

derivation: derivation according to the proof system in fig. 3.4 that shows the translation as a stepwise process and includes all subterms with their abstraction prefixes. Note that in the notation of the prefixed terms the binding annotation of a variable is omitted if it is empty. The leftmost variable $\star$ is omitted entirely if it has no binding annotations. Even though the correspondence between the derivations in the proof system in fig. 3.4 and the translations $[\![\cdot]\!]_{\mathcal{T}}$ and $[\![\cdot]\!]_{\mathcal{T}}^{\min}$ is not provided here, we think that the derivation can help as an illustration of the translation process and its result.

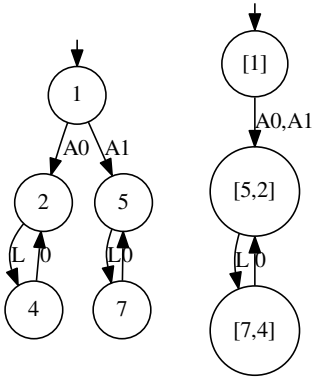
DFA: The implementation produces a graphical depiction for the term's λ-term-graph in DFA form, as well as for the minimised form of the DFA; the pictures are included next to the textual output.

spanning tree: A textual representation of the minimised DFA's spanning tree, used for readback. It is displayed in first-order term rewriting notation, i.e. with a unary function symbols `L`, `S` for abstraction and scope delimiters, a binary function symbol `A` for application and a nullary symbol `0` for abstraction variable occurrences. Furthermore there is a class of function symbols written as a vertical bar followed by an upper-case variable name `|F`, `|G`, etc. which signify a vertex with multiple incoming non-backlink edges, and therefore a vertex that will be the root of a shared subterm. There is also a class of corresponding nullary function symbols `F`, `G`, etc. which represent non-backlink, non-spanning-tree edges to these shared vertices.

readback: readback of the minimised DFA.

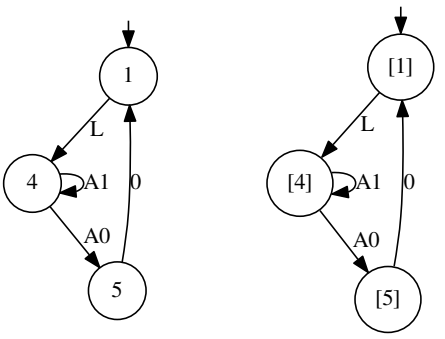
§ B.1 (example 3.3.2)**.** For this example only $[\![\cdot]\!]_{\mathcal{T}}^{\min}$ is shown since both semantics yield the exact same output.

```
λ-letrec-term:
(λx. x) (λx. x)

translation used: minimal prefix lengths

scoped (adbmals):
(λx. x) (λx. x)

scoped (nameless):
(λ. 0) (λ. 0)

derivation:
----- 0     ----- 0
(x) x       (x) x
-------- λ  -------- λ
() λx. x    () λx. x
---------------------- @
() (λx. x) (λx. x)

DFA: writing to file

spanning tree:
@(L(0), L(0))

readback:
(λx. x) (λy. y)

minimised DFA: writing to file

spanning tree:
@(|F(L(0)), F)

readback:
let F = λx. x
in F F
```

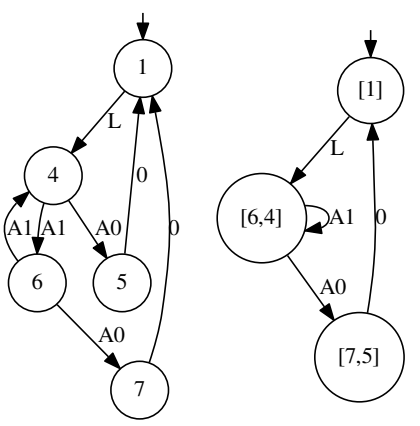


§ B.2 (example 3.3.3)**.** For both L and P only $[\![\cdot]\!]^{\min}_{\mathcal{T}}$ is shown because $[\![\cdot]\!]^{\min}_{\mathcal{T}}$ yields the same output.

```
λ-letrec-term:
λf. let r = f r
    in r

translation used: minimal prefix lengths

scoped (adbmals):
λf. let r = f r
    in r

scoped (nameless):
λ. let r = 0 r
   in r

derivation:
-------- 0
(f{r}) f    (f{r}) r
-------------------- @
(f{r}) f r              (f{r}) r
-------------------------------- let
(f) let r = f r
    in r
---------------------------------- λ
() λf. let r = f r
       in r

DFA: writing to file

spanning tree:
L(|F(@(0, F)))

readback:
λx. let F = x F
    in F

minimised DFA: writing to file

spanning tree:
L(|F(@(0, F)))

readback:
λx. let F = x F
    in F
```

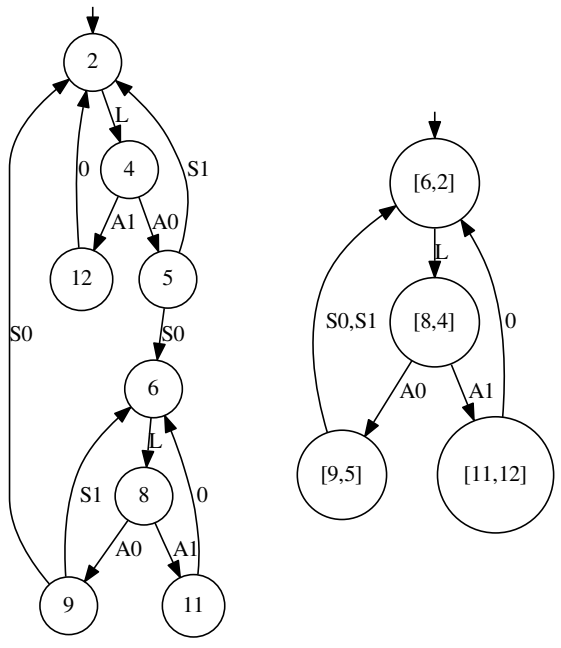



```
λ-letrec-term:
λf. let r = f (f r)
    in r

translation used: minimal prefix lengths

scoped (adbmals):
λf. let r = f (f r)
    in r

scoped (nameless):
λ. let r = 0 (0 r)
   in r

derivation:
            -------- 0
            (f{r}) f    (f{r}) r
-------- 0  -------------------- @
(f{r}) f    (f{r}) f r
---------------------------------- @
(f{r}) f (f r)                          (f{r}) r
------------------------------------------------ let
(f) let r = f (f r)
    in r
-------------------------------------------------- λ
() λf. let r = f (f r)
       in r

DFA: writing to file

spanning tree:
L(|F(@(0, @(0, F))))

readback:
λx. let F = x (x F)
    in F

minimised DFA: writing to file

spanning tree:
L(|F(@(0, F)))

readback:
λx. let F = x F
    in F
```

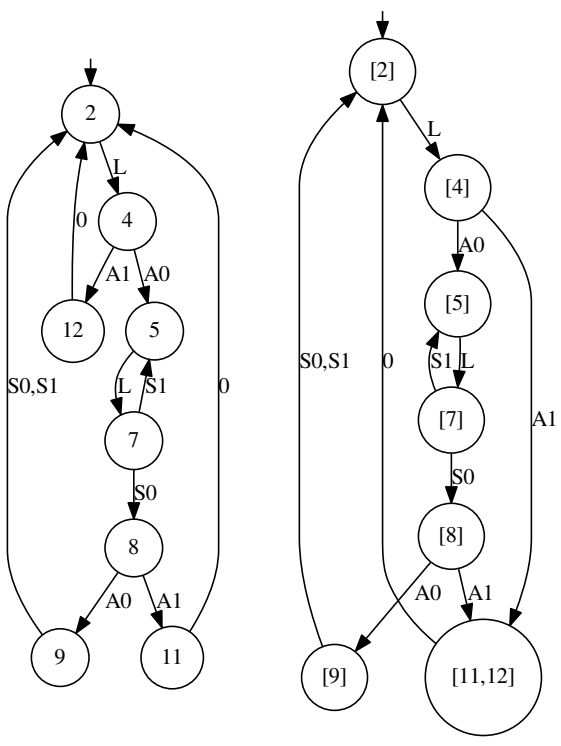


§ B.3 (example 3.5.2)**.** Again, for the terms L_1 and L_2 of example 3.5.2 the translations are identical for $[\![\cdot]\!]_{\mathcal{T}}$ and $[\![\cdot]\!]_{\mathcal{T}}^{\min}$.

```
λ-letrec-term:
let f = λx. (λy. f y) x
in f

translation used: minimal prefix lengths

scoped (adbmals):
let f = λx. /x. λy. /y. f y x
in f

scoped (nameless):
let f = λ. S((λ. S(f) 0)) 0
in f

derivation:
(*{f}) f
---------- S  ---------- 0
(*{f} y) f    (*{f} y) y
------------------------- @
(*{f} y) f y
--------------------------- λ
(*{f}) λy. f y
----------------------------- S  ---------- 0
(*{f} x) λy. f y                (*{f} x) x
------------------------------------------- @
(*{f} x) (λy. f y) x
-------------------------------------------- λ
(*{f}) λx. (λy. f y) x                          (*{f}) f
------------------------------------------------------- let
() let f = λx. (λy. f y) x
   in f

DFA: writing to file

spanning tree:
|F(L(@(S(L(@(S(F), 0))), 0)))

readback:
let F = λx. /x. λy. /y. F y x
in F

minimised DFA: writing to file

spanning tree:
|F(L(@(S(F), 0)))

readback:
let F = λx. F x
in F
```

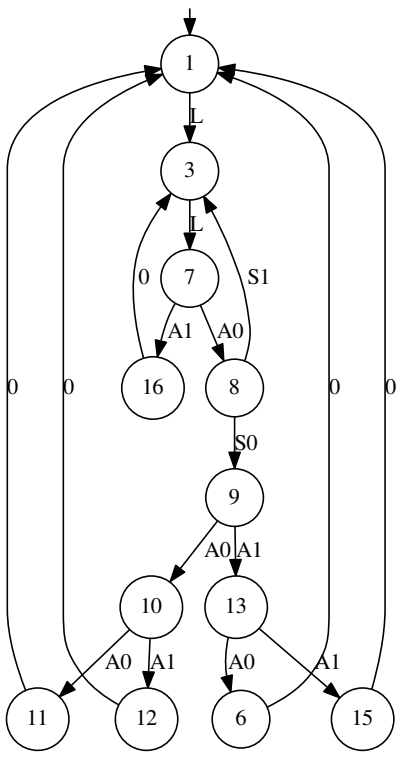


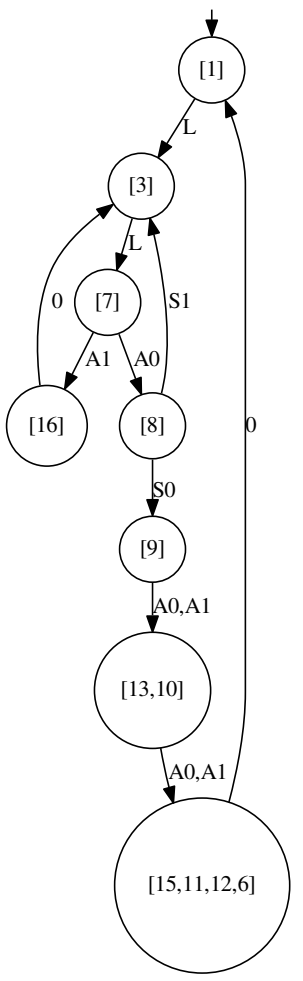

```
λ-letrec-term:
let f = λx. (λy. f x) x
in f

translation used: minimal prefix lengths

scoped (adbmals):
let f = λx. (λy. /y. /x. f x) x
in f

scoped (nameless):
let f = λ. (λ. S((S(f) 0))) 0
in f

derivation:
(*{f}) f
---------- S  ---------- 0
(*{f} x) f    (*{f} x) x
------------------------- @
(*{f} x) f x
--------------------------- S
(*{f} x y) f x
----------------------------- λ  ---------- 0
(*{f} x) λy. f x                (*{f} x) x
------------------------------------------- @
(*{f} x) (λy. f x) x
-------------------------------------------- λ
(*{f}) λx. (λy. f x) x                          (*{f}) f
------------------------------------------------------- let
() let f = λx. (λy. f x) x
   in f

DFA: writing to file

spanning tree:
|F(L(@(L(S(@(S(F), 0))), 0)))

readback:
let F = λx. (λy. /y. /x. F x) x
in F

minimised DFA: writing to file

spanning tree:
|F(L(@(S(F), |G(0)))), G)))

readback:
let F = λx. let G = x
            in (λy. F G) G
in F
```



§ B.4 (fig. 3.6)**.** Also for the term $\lambda a.\, \lambda b.\, \mathsf{let}\ f = a\ \mathsf{in}\ a\ a\ (f\ a)\ b$ from fig. 3.6 the translations are identical for $[\![\cdot]\!]_{\mathcal{T}}$ and $[\![\cdot]\!]^{\min}_{\mathcal{T}}$.

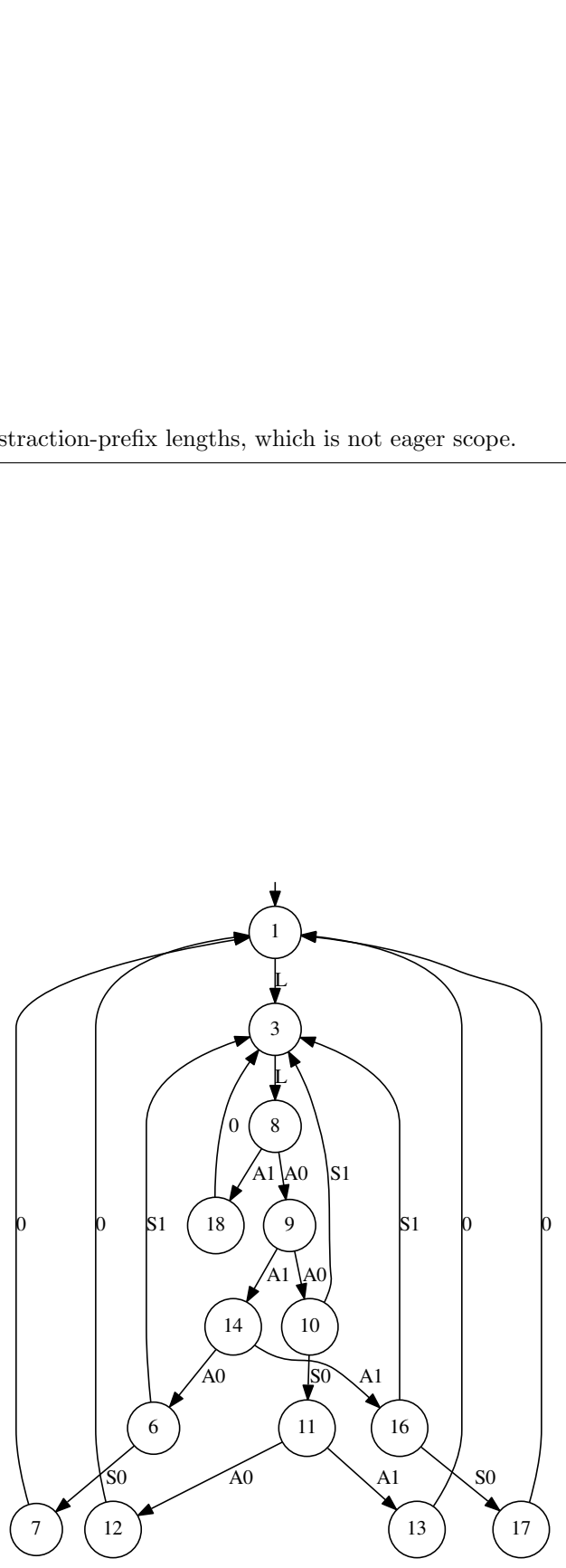
```
λ-letrec-term:
λa. λb. let f = a
        in a a (f a) b

translation used: minimal prefix lengths

scoped (adbmals):
λa. λb. let f = a
        in /b. a a (f a) b

scoped (nameless):
λ. λ. let f = 0
      in S((0 0 (f 0))) 0

derivation:
            -------- 0  -------- 0              -------- 0
            (a{f}) a    (a{f}) a      (a{f}) f  (a{f}) a
            ---------------------- @  -------------------- @
            (a{f}) a a                (a{f}) f a
            ------------------------------------------------ @
            (a{f}) a a (f a)
            ------------------------------------------------ S  ---------- 0
            (a{f} b) a a (f a)                                   (a{f} b) b
-------- 0  ------------------------------------------------------------- @
(a{f}) a    (a{f} b) a a (f a) b
------------------------------------------------------------------------- let
(a b) let f = a
      in a a (f a) b
------------------------------------------------------------------------- λ
(a) λb. let f = a
        in a a (f a) b
------------------------------------------------------------------------- λ
() λa. λb. let f = a
           in a a (f a) b

DFA: writing to file

spanning tree:
L(L(@(S(@(@(0, 0), @(0, 0))), 0)))

readback:
λx. λy. /y. x x (x x) y

minimised DFA: writing to file

spanning tree:
L(L(@(S(@(|F(@(|G(0), G)), F)), 0)))

readback:
λx. let F = G G
        G = x
    in λy. F F y
```

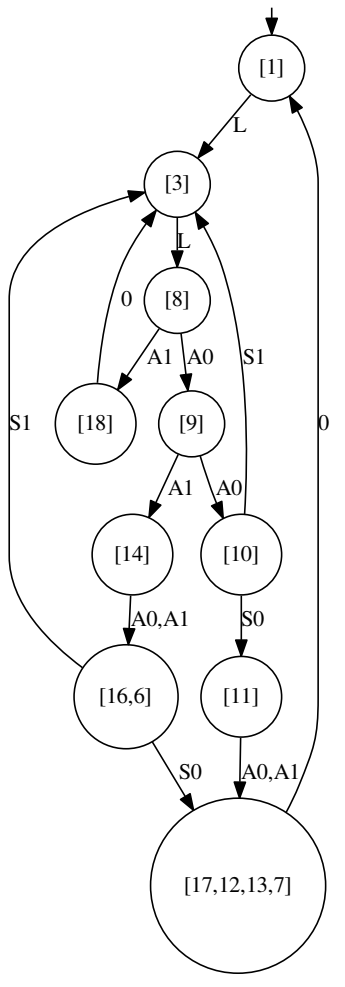




To illustrate the necessity for different prefix lengths (§ 3.5.17) we also provide the translation with equal abstraction-prefix lengths, which is not eager scope.

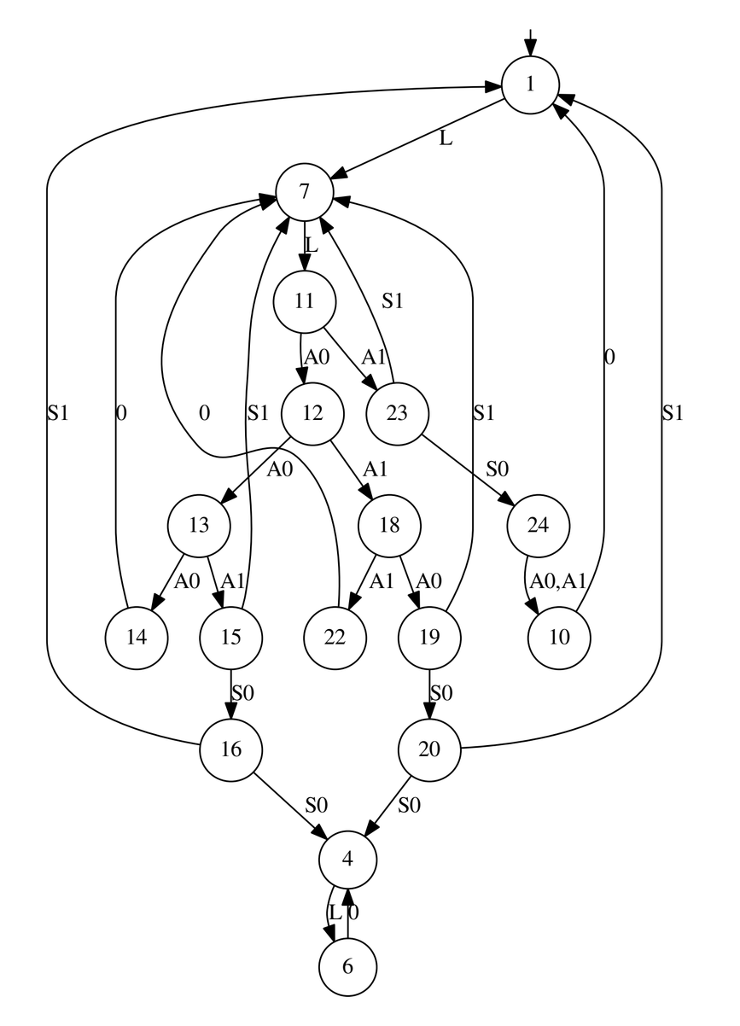
```
λ-letrec-term:
λa. λb. let f = a
        in a a (f a) b

translation used: maximal prefix lengths

scoped (adbmals):
λa. λb. let f = /b. a
        in /b. a a (f /b. a) b

scoped (nameless):
λ. λ. let f = S(0)
      in S((0 0)) (f S(0)) 0

derivation:
              ----- 0  ----- 0                  ----- 0
              (a) a    (a) a                    (a) a
              ---------------- @                ---------- S
              (a) a a                 (a b{f}) f  (a b{f}) a
              ----------------- S  ------------------------ @
              (a b{f}) a a            (a b{f}) f a
----- 0       ------------------------------------------------ @  ---------- 0
(a) a         (a b{f}) a a (f a)                                   (a b{f}) b
---------- S  ------------------------------------------------------------- @
(a b{f}) a    (a b{f}) a a (f a) b
------------------------------------------------------------------------- let
(a b) let f = a
      in a a (f a) b
------------------------------------------------------------------------- λ
(a) λb. let f = a
        in a a (f a) b
------------------------------------------------------------------------- λ
() λa. λb. let f = a
           in a a (f a) b

DFA: writing to file

spanning tree:
L(L(@(@(S(@(0, 0)), @(S(0), S(0))), 0)))

readback:
λx. λy. /y. x x (/y. x /y. x) y

minimised DFA: writing to file

spanning tree:
L(L(@(@(S(@(|G(0), G)), @(|F(S(G)), F)), 0)))

readback:
λx. let G = x
    in λy. let F = G
           in G G (F F) y
```

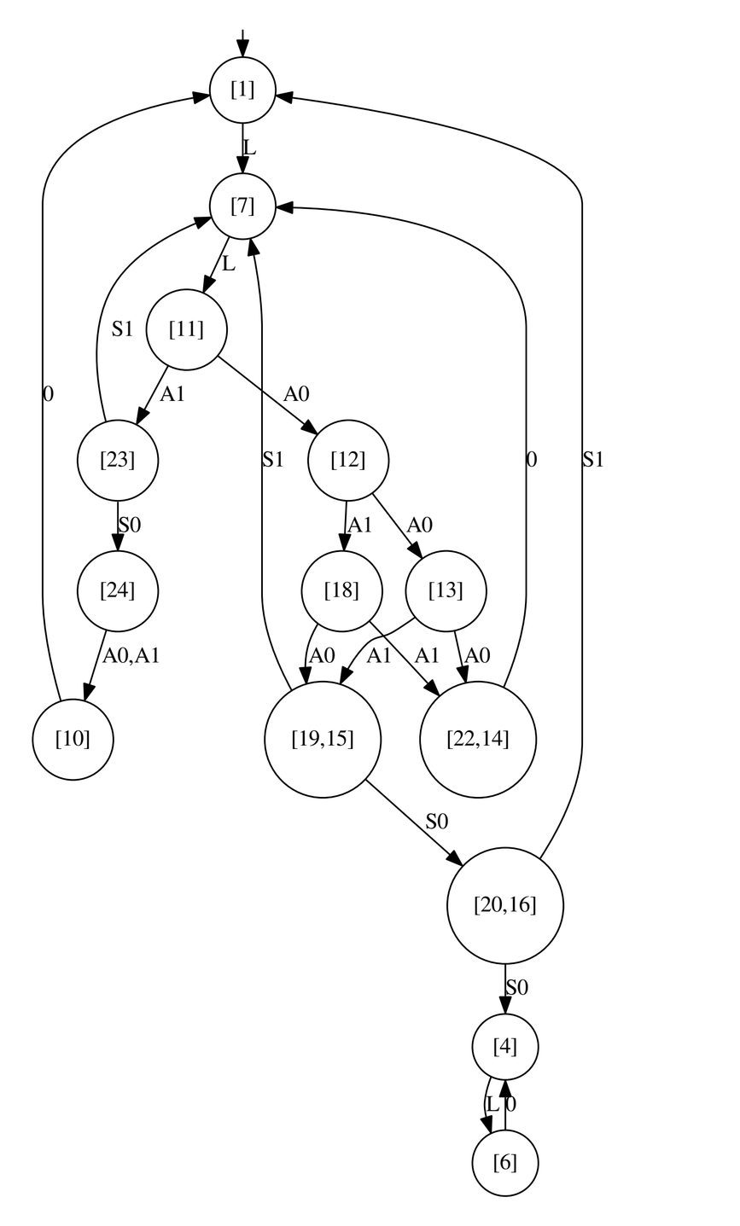




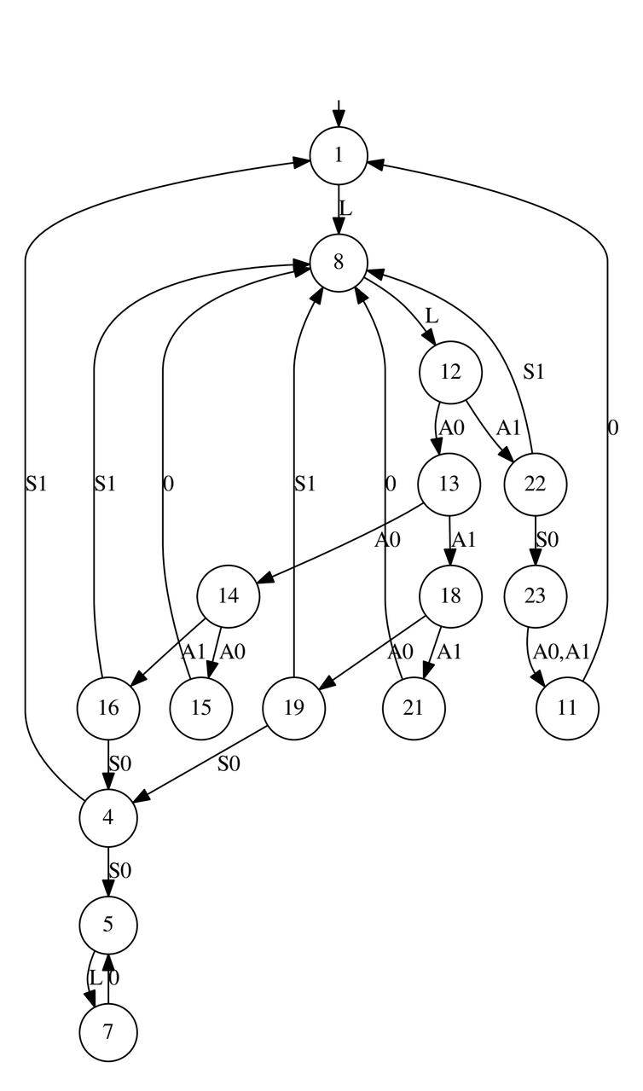
§ B.5 (example 3.6.18). For the terms L_1, L_2, and L_3 from example 3.6.18 the translations differ for $[\![\cdot]\!]_{\mathcal{T}}$ and $[\![\cdot]\!]_{\mathcal{T}}^{\min}$. Thus we provide the output for both translations for each of the terms.

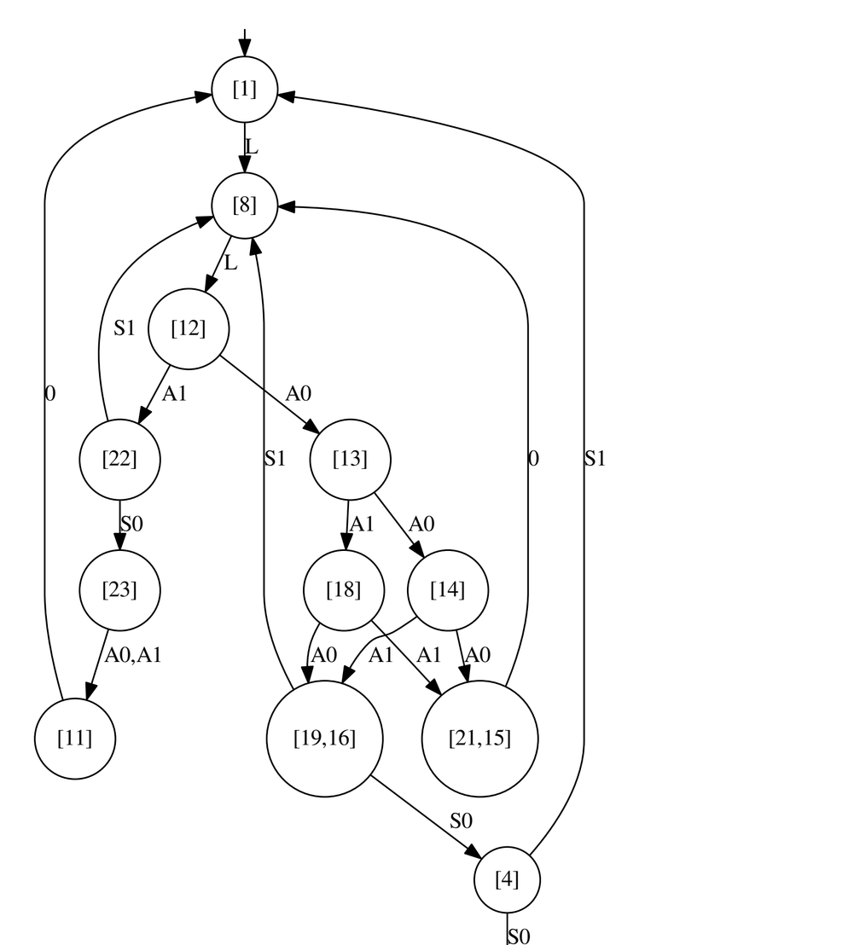
```
λ-letrec-term:
λx. let I = λz. z
    in λy. let f = x
           in y I (I y) (f f)

translation used: minimal prefix lengths

scoped (adbmals):
λx. let I = λz. z
    in λy. let f = x
           in y /y. /x. I (/y. /x. I y) /y. f f

scoped (nameless):
λ. let I = λ. 0
   in λ. let f = 0
         in 0 S(S(I)) (S(S(I)) 0) S((f f))

derivation:
                                                     (*{I}) I             (*{I}) I
                                                     ------------- S      ------------- S
                                                     (*{I} x{f}) I        (*{I} x{f}) I
                                  --------------- 0  --------------- S    --------------- S  --------------- 0
                                  (*{I} x{f} y) y    (*{I} x{f} y) I      (*{I} x{f} y) I    (*{I} x{f} y) y       (*{I} x{f}) f  (*{I} x{f}) f
                                  ------------------------------------ @  ------------------------------------ @   ----------------------------- @
                                  (*{I} x{f} y) y I                       (*{I} x{f} y) I y                        (*{I} x{f}) f f
                                  --------------------------------------------------------------------------- @    ------------------------------- S
                                  (*{I} x{f} y) y I (I y)                                                          (*{I} x{f} y) f f
                 ------------- 0  ------------------------------------------------------------------------------------------------------------- @
                 (*{I} x{f}) x    (*{I} x{f} y) y I (I y) (f f)
                 --------------------------------------------------------------------------------------------------------------------------------- let
                 (*{I} x y) let f = x
----------- 0               in y I (I y) (f f)
(*{I} z) z       ------------------------------------------------------------------------------------------------------------------------------------ λ
------------ λ   (*{I} x) λy. let f = x
(*{I}) λz. z                  in y I (I y) (f f)
---------------------------------------------------------------------------------------------------------------------------------------------------- let
(x) let I = λz. z
    in λy. let f = x
           in y I (I y) (f f)
-------------------------------------------------------------------------------------------------------------------------------------------------------- λ
() λx. let I = λz. z
       in λy. let f = x
              in y I (I y) (f f)

DFA: writing to file

spanning tree:
L(L(@(@(@(0, S(S(|F(L(0))))), @(S(S(F)), 0)), S(@(|G(0), G)))))

readback:
let F = λy. y
in λx. let G = x
       in λz. z /z. /x. F (/z. /x. F z) /z. G G

minimised DFA: writing to file

spanning tree:
L(L(@(@(@(|H(0), |G(S(S(L(0))))), @(G, H)), S(@(|F(0), F)))))

readback:
λx. let F = x
    in λy. let H = y
               G = λz. z
           in H G (G H) (F F)
```



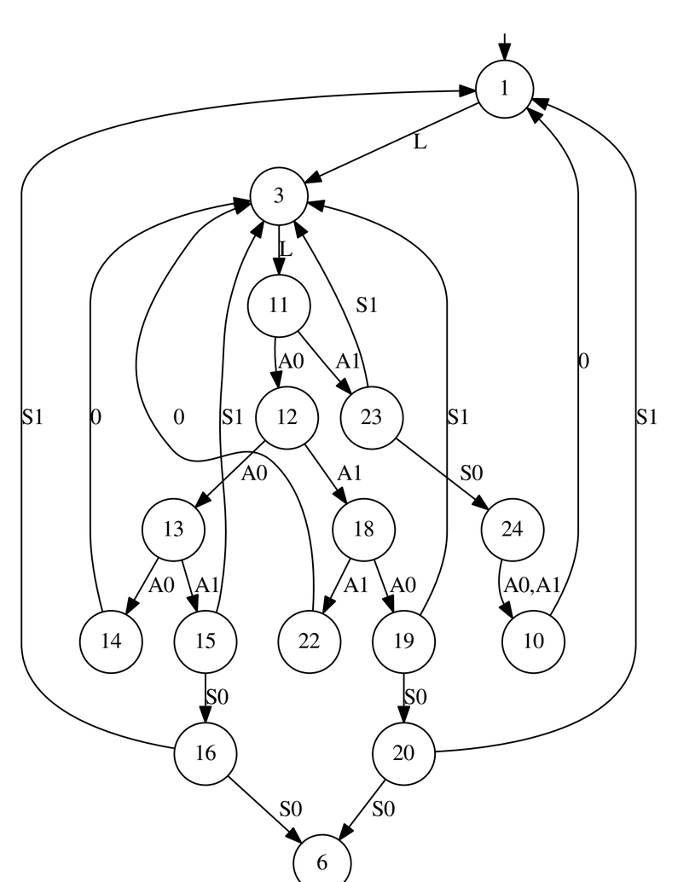


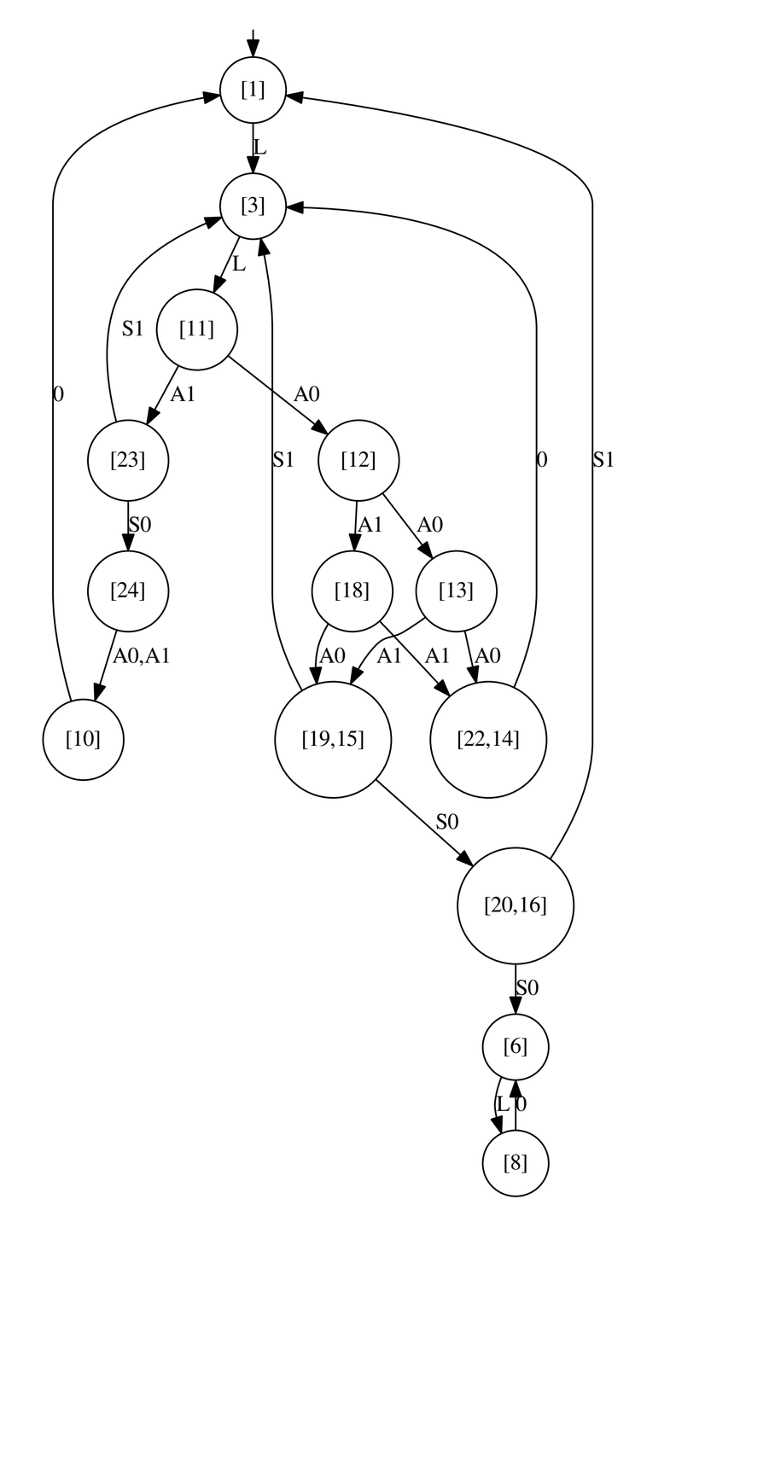
```
λ-letrec-term:
λx. let I = λz. z
    in λy. let f = x
           in y I (I y) (f f)

translation used: maximal prefix lengths but eager scope

scoped (adbmals):
λx. let I = /x. λz. z
    in λy. let f = x
           in y /y. I (/y. I y) /y. f f

scoped (nameless):
λ. let I = S((λ. 0))
   in λ. let f = 0
         in 0 S(I) (S(I) 0) S((f f))

derivation:
                                             (x{I f}) I        (x{I f}) I
                              ------------ 0  ------------ S    ------------ S  ------------ 0
                              (x{I f} y) y    (x{I f} y) I      (x{I f} y) I    (x{I f} y) y       (x{I f}) f  (x{I f}) f
                              ------------------------------ @  ------------------------------ @   ---------------------- @
                              (x{I f} y) y I                    (x{I f} y) I y                     (x{I f}) f f
                              ------------------------------------------------------------------ @ ------------------------ S
                              (x{I f} y) y I (I y)                                                 (x{I f} y) f f
                ---------- 0  ------------------------------------------------------------------------------------------ @
                (x{I f}) x    (x{I f} y) y I (I y) (f f)
----- 0         ------------------------------------------------------------------------------------------------------ let
(z) z           (x{I} y) let f = x
-------- λ               in y I (I y) (f f)
() λz. z        ----------------------------------------------------------------------------------------------------------- λ
------------ S  (x{I}) λy. let f = x
(x{I}) λz. z               in y I (I y) (f f)
----------------------------------------------------------------------------------------------------------------------------- let
(x) let I = λz. z
    in λy. let f = x
           in y I (I y) (f f)
---------------------------------------------------------------------------------------------------------------------------------- λ
() λx. let I = λz. z
       in λy. let f = x
              in y I (I y) (f f)

DFA: writing to file

spanning tree:
L(L(@(@(@(0, S(|F(S(L(0))))), @(S(F), 0)), S(@(|G(0), G)))))

readback:
λx. let F = /x. λy. y
        G = x
    in λz. z /z. F (/z. F z) /z. G G

minimised DFA: writing to file

spanning tree:
L(L(@(@(@(|H(0), |G(S(S(L(0))))), @(G, H)), S(@(|F(0), F)))))

readback:
λx. let F = x
    in λy. let H = y
               G = λz. z
           in H G (G H) (F F)
```



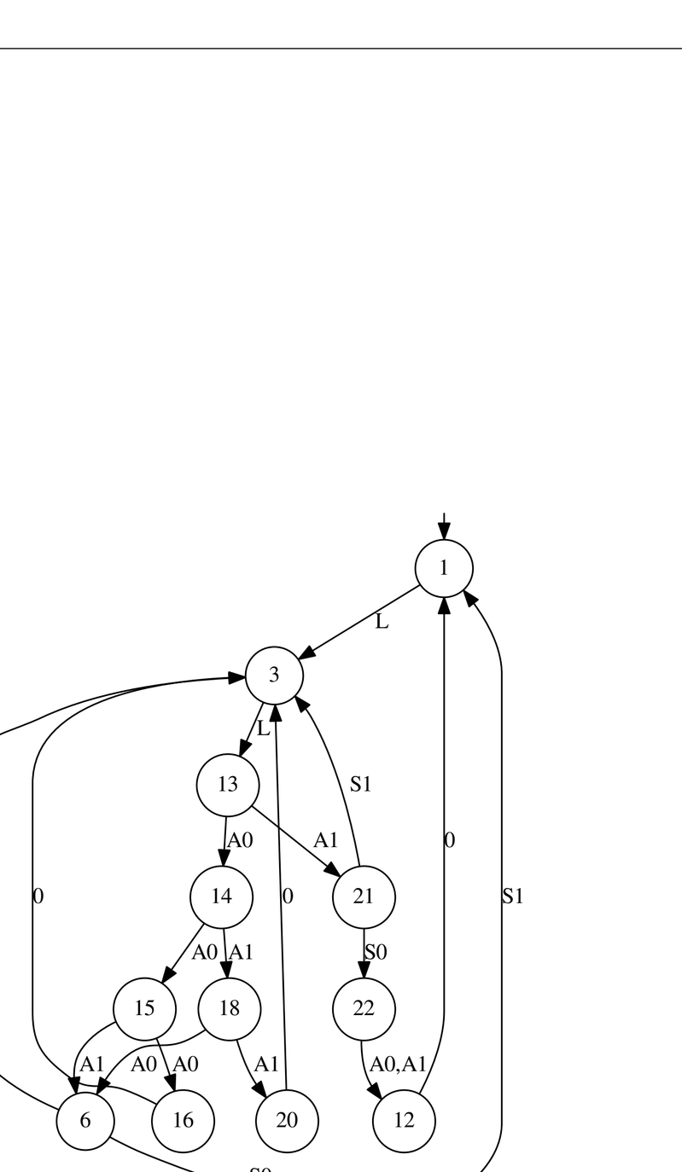


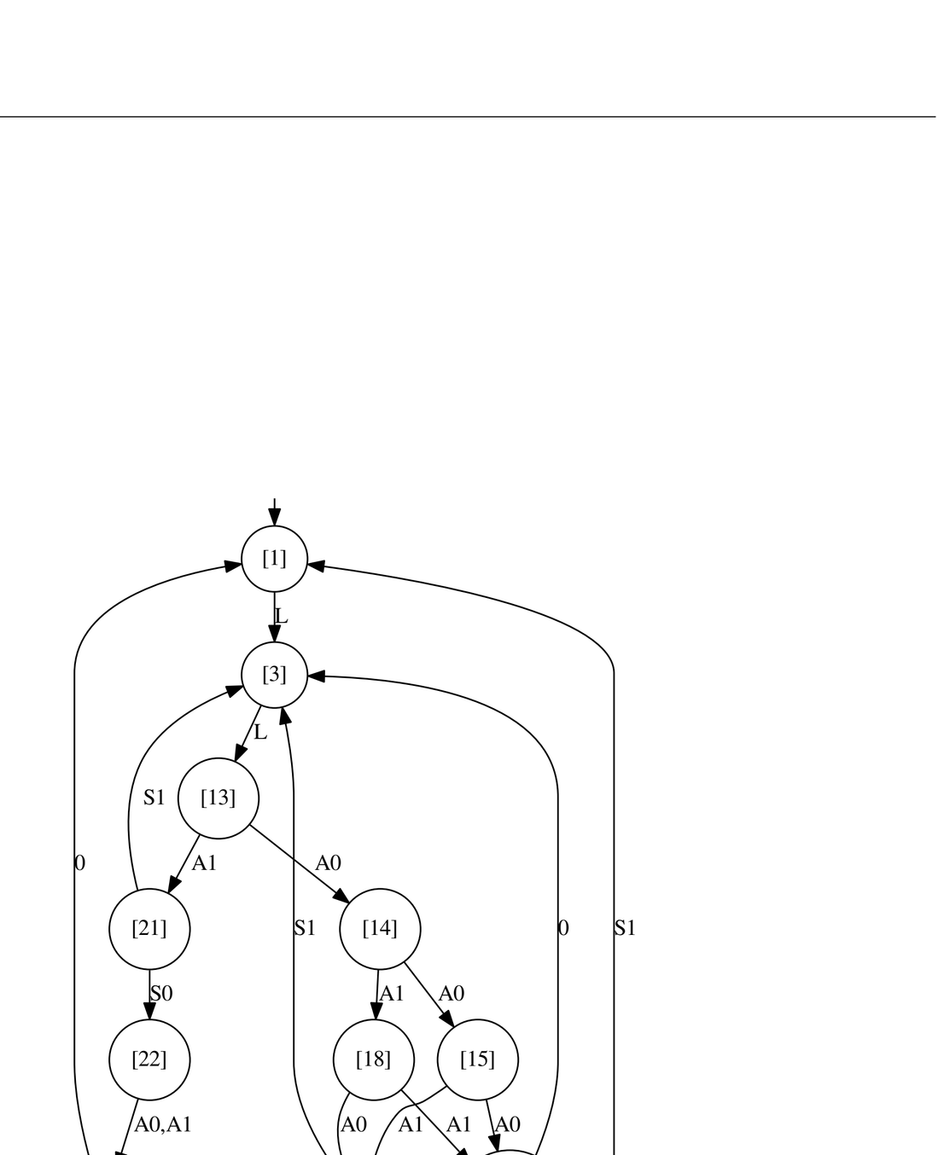
```
λ-letrec-term:
λx. λy. let I = λz. z
            f = x
        in y I (I y) (f f)

translation used: minimal prefix lengths

scoped (adbmals):
λx. λy. let I = λz. z
            f = x
        in y /y. /x. I (/y. /x. I y) /y. f f

scoped (nameless):
λ. λ. let I = λ. 0
          f = 0
      in 0 S(S(I)) (S(S(I)) 0) S((f f))

derivation:
                                                   (*{I}) I             (*{I}) I
                                                   ------------- S      ------------- S
                                                   (*{I} x{f}) I        (*{I} x{f}) I
                                --------------- 0  --------------- S    --------------- S  --------------- 0
                                (*{I} x{f} y) y    (*{I} x{f} y) I      (*{I} x{f} y) I    (*{I} x{f} y) y       (*{I} x{f}) f  (*{I} x{f}) f
                                ------------------------------------ @  ------------------------------------ @   ----------------------------- @
                                (*{I} x{f} y) y I                       (*{I} x{f} y) I y                        (*{I} x{f}) f f
---------- 0                    --------------------------------------------------------------------------- @    ------------------------------- S
(*{I} z) z                      (*{I} x{f} y) y I (I y)                                                          (*{I} x{f} y) f f
------------ λ  ------------- 0 ----------------------------------------------------------------------------------------------------------- @
(*{I}) λz. z    (*{I} x{f}) x   (*{I} x{f} y) y I (I y) (f f)
---------------------------------------------------------------------------------------------------------------------------------------- let
(x y) let I = λz. z
          f = x
      in y I (I y) (f f)
-------------------------------------------------------------------------------------------------------------------------------------------- λ
(x) λy. let I = λz. z
            f = x
        in y I (I y) (f f)
---------------------------------------------------------------------------------------------------------------------------------------------- λ
() λx. λy. let I = λz. z
               f = x
           in y I (I y) (f f)

DFA: writing to file

spanning tree:
L(L(@(@(@(0, S(S(|F(L(0))))), @(S(S(F)), 0)), S(@(|G(0), G)))))

readback:
let F = λz. z
in λx. let G = x
       in λy. y /y. /x. F (/y. /x. F y) /y. G G

minimised DFA: writing to file

spanning tree:
L(L(@(@(@(|H(0), |G(S(S(L(0))))), @(G, H)), S(@(|F(0), F)))))

readback:
λx. let F = x
    in λy. let H = y
               G = λz. z
           in H G (G H) (F F)
```



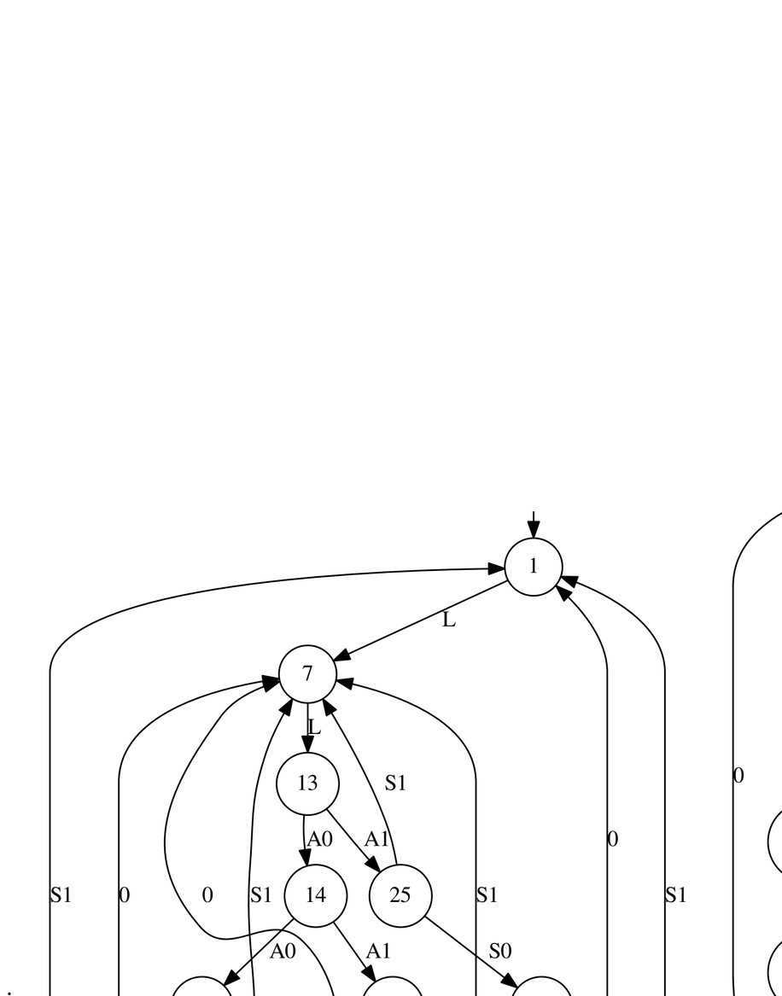


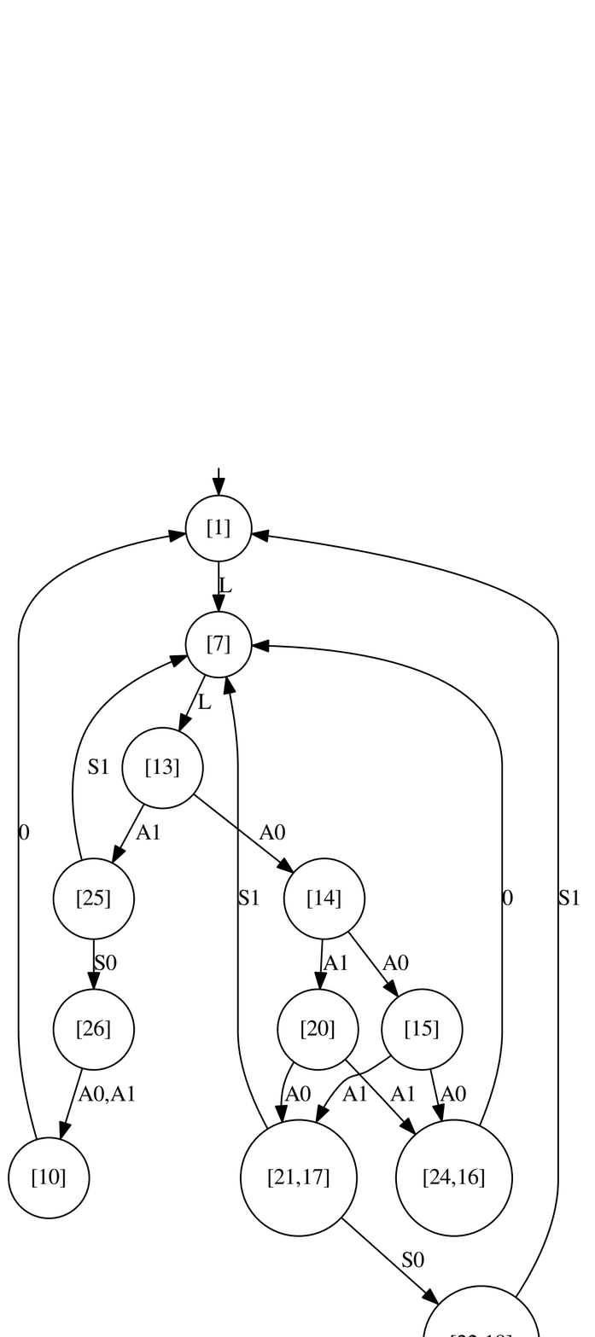
```
λ-letrec-term:
λx. λy. let I = λz. z
            f = x
        in y I (I y) (f f)

translation used: maximal prefix lengths but eager scope

scoped (adbmals):
λx. λy. let I = /y. /x. λz. z
            f = x
        in y I (I y) /y. f f

scoped (nameless):
λ. λ. let I = S(S((λ. 0)))
          f = 0
      in 0 I (I 0) S((f f))

derivation:
----- 0                          ------------- 0                                   ------------- 0
(z) z                            (x{f} y{I}) y    (x{f} y{I}) I    (x{f} y{I}) I  (x{f} y{I}) y        (x{f}) f  (x{f}) f
-------- λ                       ------------------------------ @  ------------------------------ @    ------------------ @
() λz. z                         (x{f} y{I}) y I                   (x{f} y{I}) I y                     (x{f}) f f
------------ S                   ------------------------------------------------------------------ @  -------------------- S
(x{f}) λz. z                     (x{f} y{I}) y I (I y)                                                 (x{f} y{I}) f f
----------------- S  -------- 0  ----------------------------------------------------------------------------------------- @
(x{f} y{I}) λz. z    (x{f}) x    (x{f} y{I}) y I (I y) (f f)
----------------------------------------------------------------------------------------------------------------------- let
(x y) let I = λz. z
          f = x
      in y I (I y) (f f)
-------------------------------------------------------------------------------------------------------------------------- λ
(x) λy. let I = λz. z
            f = x
        in y I (I y) (f f)
---------------------------------------------------------------------------------------------------------------------------- λ
() λx. λy. let I = λz. z
               f = x
           in y I (I y) (f f)

DFA: writing to file

spanning tree:
L(L(@(@(@(0, |F(S(S(L(0))))), @(F, 0)), S(@(|G(0), G)))))

readback:
λx. let G = x
    in λy. let F = /y. /x. λz. z
           in y F (F y) /y. G G

minimised DFA: writing to file

spanning tree:
L(L(@(@(@(|H(0), |F(S(S(L(0))))), @(F, H)), S(@(|G(0), G)))))

readback:
λx. let G = x
    in λy. let H = y
               F = λz. z
           in H F (F H) (G G)
```



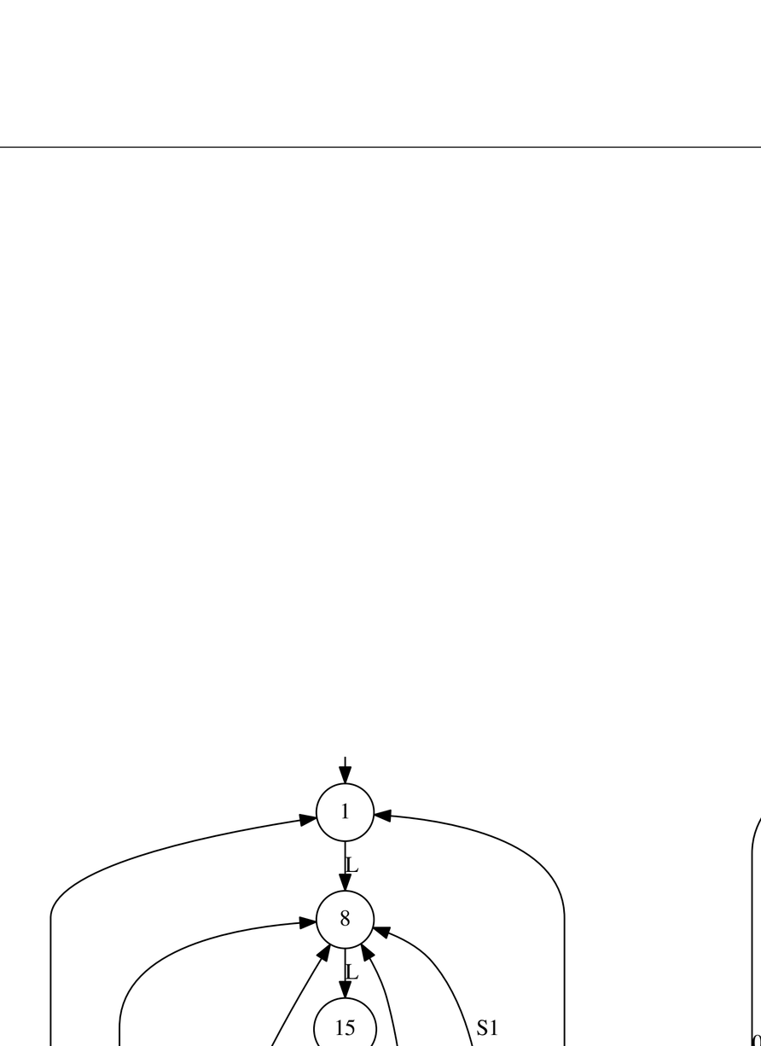


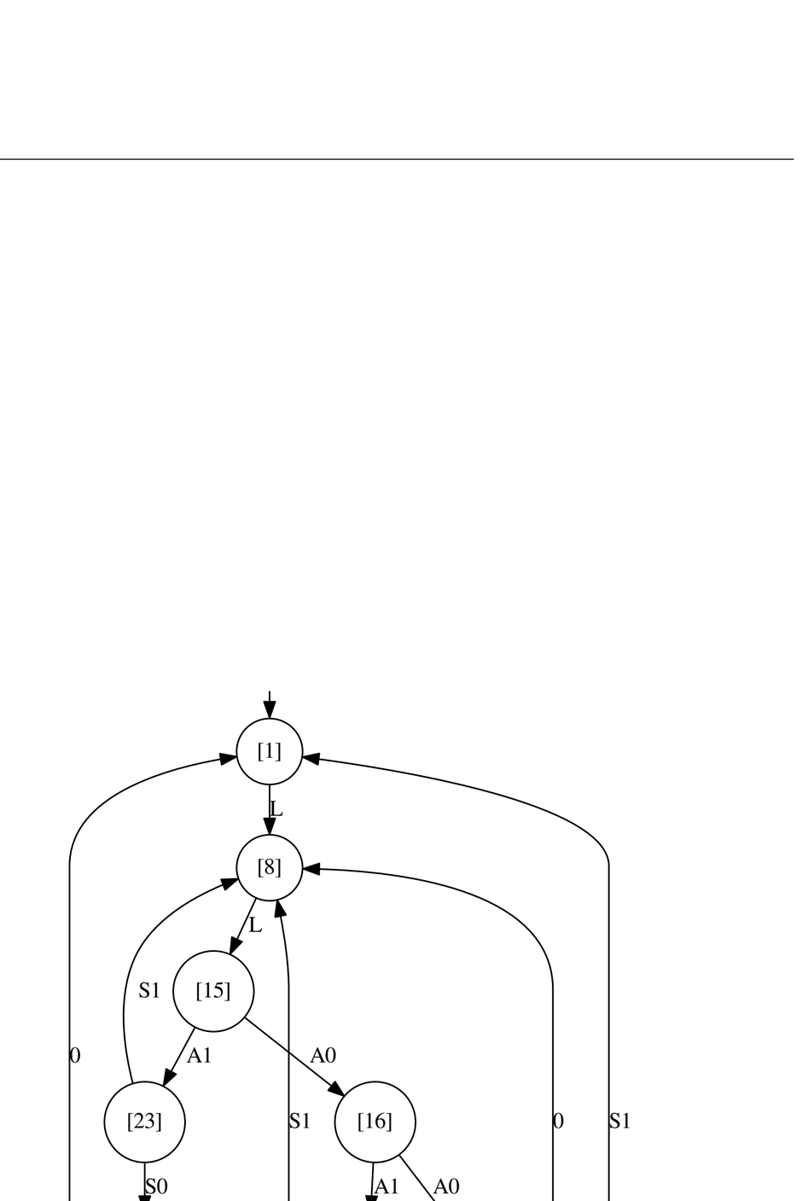
```
λ-letrec-term:
λx. let I = λz. z
    in λy. let f = x
               g = I
           in y g (g y) (f f)

translation used: minimal prefix lengths

scoped (adbmals):
λx. let I = λz. z
    in λy. let f = x
               g = I
           in y /y. /x. g (/y. /x. g y) /y. f f

scoped (nameless):
λ. let I = λ. 0
   in λ. let f = 0
             g = I
         in 0 S(S(g)) (S(S(g)) 0) S((f f))

derivation:
                                                                (*{I g}) g             (*{I g}) g
                                                                --------------- S      --------------- S
                                                                (*{I g} x{f}) g        (*{I g} x{f}) g
                                            ----------------- 0 ----------------- S    ----------------- S  ----------------- 0
                                            (*{I g} x{f} y) y   (*{I g} x{f} y) g      (*{I g} x{f} y) g    (*{I g} x{f} y) y       (*{I g} x{f}) f  (*{I g} x{f}) f
                                            ---------------------------------------- @ ----------------------------------------- @  -------------------------------- @
                                            (*{I g} x{f} y) y g                        (*{I g} x{f} y) g y                          (*{I g} x{f}) f f
                                            ------------------------------------------------------------------------------------ @  ---------------------------------- S
                                            (*{I g} x{f} y) y g (g y)                                                               (*{I g} x{f} y) f f
               --------------- 0            --------------------------------------------------------------------------------------------------------------------------- @
               (*{I g} x{f}) x  (*{I g}) I  (*{I g} x{f} y) y g (g y) (f f)
               ------------------------------------------------------------------------------------------------------------------------------------------------ let
               (*{I} x y) let f = x
                              g = I
                          in y g (g y) (f f)
---------- 0   -------------------------------------------------------------------------------------------------------------------------------------------------- λ
(*{I} z) z     (*{I} x) λy. let f = x
------------ λ                  g = I
(*{I}) λz. z                in y g (g y) (f f)
------------------------------------------------------------------------------------------------------------------------------------------------------------------ let
(x) let I = λz. z
    in λy. let f = x
               g = I
           in y g (g y) (f f)
----------------------------------------------------------------------------------------------------------------------------------------------------------------------- λ
() λx. let I = λz. z
       in λy. let f = x
                  g = I
              in y g (g y) (f f)

DFA: writing to file

spanning tree:
L(L(@(@(@(0, S(S(|F(L(0))))), @(S(S(F)), 0)), S(@(|G(0), G)))))

readback:
let F = λy. y
in λx. let G = x
       in λz. z /z. /x. F (/z. /x. F z) /z. G G

minimised DFA: writing to file

spanning tree:
L(L(@(@(@(|H(0), |G(S(S(L(0))))), @(G, H)), S(@(|F(0), F)))))

readback:
λx. let F = x
    in λy. let H = y
               G = λz. z
           in H G (G H) (F F)
```





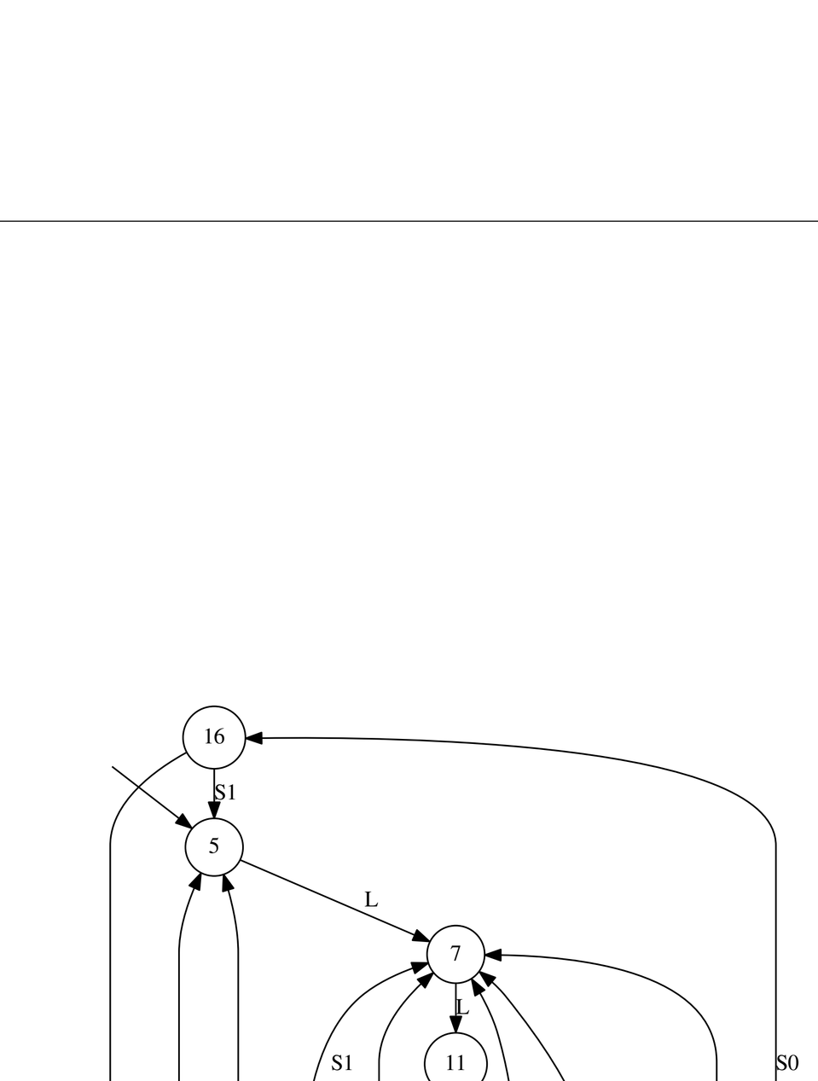
```
λ-letrec-term:
λx. let I = λz. z
    in λy. let f = x
               g = I
           in y g (g y) (f f)

translation used: maximal prefix lengths but eager scope

scoped (adbmals):
λx. let I = /x. λz. z
    in λy. let f = x
               g = /y. I
           in y g (g y) /y. f f

scoped (nameless):
λ. let I = S((λ. 0))
   in λ. let f = 0
             g = S(I)
         in 0 g (g 0) S((f f))

derivation:
                                              --------------- 0                                   --------------- 0
                                              (x{I f} y{g}) y    (x{I f} y{g}) g    (x{I f} y{g}) g  (x{I f} y{g}) y        (x{I f}) f  (x{I f}) f
                                              ---------------------------------- @  ---------------------------------- @    ---------------------- @
                                              (x{I f} y{g}) y g                     (x{I f} y{g}) g y                       (x{I f}) f f
                          (x{I f}) I          ---------------------------------------------------------------------- @      ------------------------ S
                                              (x{I f} y{g}) y g (g y)                                                       (x{I f} y{g}) f f
              ---------- 0  --------------- S ------------------------------------------------------------------------------------------------ @
              (x{I f}) x    (x{I f} y{g}) I   (x{I f} y{g}) y g (g y) (f f)
              ------------------------------------------------------------------------------------------------------------------------------ let
              (x{I} y) let f = x
----- 0                      g = I
(z) z                    in y g (g y) (f f)
-------- λ    -------------------------------------------------------------------------------------------------------------------------------- λ
() λz. z      (x{I}) λy. let f = x
------------ S                   g = I
(x{I}) λz. z                 in y g (g y) (f f)
---------------------------------------------------------------------------------------------------------------------------------------------- let
(x) let I = λz. z
    in λy. let f = x
               g = I
           in y g (g y) (f f)
-------------------------------------------------------------------------------------------------------------------------------------------------- λ
() λx. let I = λz. z
       in λy. let f = x
                  g = I
              in y g (g y) (f f)

DFA: writing to file

spanning tree:
L(L(@(@(@(0, |G(S(S(L(0))))), @(G, 0)), S(@(|F(0), F)))))

readback:
λx. let F = x
    in λz. let G = /z. /x. λy. y
           in z G (G z) /z. F F

minimised DFA: writing to file

spanning tree:
L(L(@(@(@(|H(0), |G(S(S(L(0))))), @(G, H)), S(@(|F(0), F)))))

readback:
λx. let F = x
    in λy. let H = y
               G = λz. z
           in H G (G H) (F F)
```

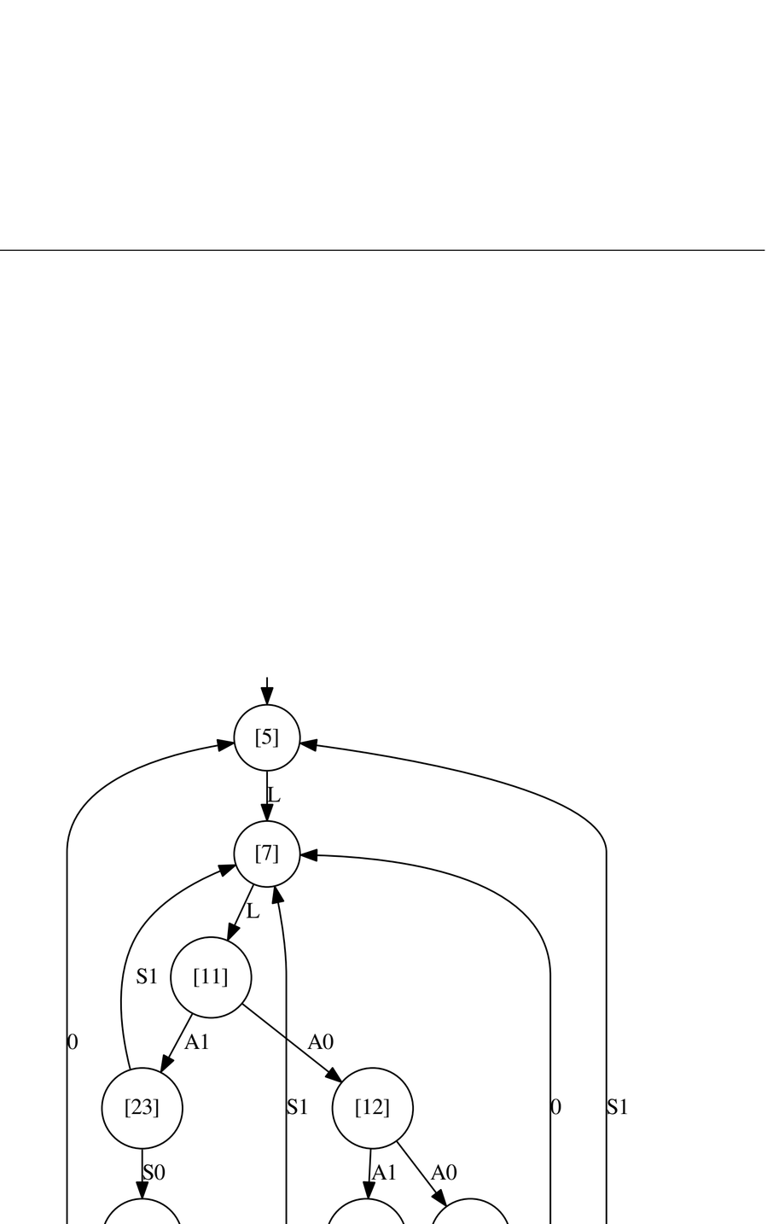




For the term L' from the same example (example 3.6.18, the translations for $[\![\cdot]\!]_{\mathcal{T}}$ and $[\![\cdot]\!]_{\mathcal{T}}^{\min}$ are identical again.

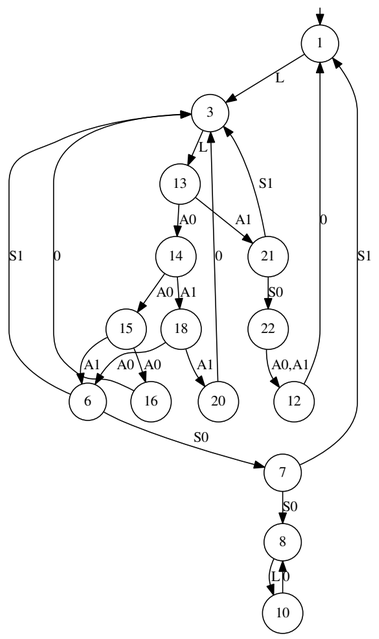
```
λ-letrec-term:
let I = λz. z
in λx. λy. let f = x
           in y I (I y) (f f)

translation used: minimal prefix lengths

scoped (adbmals):
let I = λz. z
in λx. λy. let f = x
           in y /y. /x. I (/y. /x. I y) /y. f f

scoped (nameless):
let I = λ. 0
in λ. λ. let f = 0
         in 0 S(S(I)) (S(S(I)) 0) S((f f))

derivation:
                                                   (*{I}) I             (*{I}) I
                                                   ------------- S      ------------- S
                                                   (*{I} x{f}) I        (*{I} x{f}) I
                                --------------- 0  --------------- S    --------------- S  --------------- 0
                                (*{I} x{f} y) y    (*{I} x{f} y) I      (*{I} x{f} y) I    (*{I} x{f} y) y       (*{I} x{f}) f  (*{I} x{f}) f
                                ------------------------------------ @  ------------------------------------ @   ----------------------------- @
                                (*{I} x{f} y) y I                       (*{I} x{f} y) I y                        (*{I} x{f}) f f
                                --------------------------------------------------------------------------- @    ------------------------------- S
                                (*{I} x{f} y) y I (I y)                                                          (*{I} x{f} y) f f
                ------------- 0 ---------------------------------------------------------------------------------------------------------- @
                (*{I} x{f}) x   (*{I} x{f} y) y I (I y) (f f)
                ------------------------------------------------------------------------------------------------------------------------------ let
                (*{I} x y) let f = x
                           in y I (I y) (f f)
                ---------------------------------------------------------------------------------------------------------------------------------- λ
                (*{I} x) λy. let f = x
---------- 0                 in y I (I y) (f f)
(*{I} z) z      ------------------------------------------------------------------------------------------------------------------------------------ λ
------------ λ  (*{I}) λx. λy. let f = x
(*{I}) λz. z                   in y I (I y) (f f)
-------------------------------------------------------------------------------------------------------------------------------------------------------- let
() let I = λz. z
   in λx. λy. let f = x
              in y I (I y) (f f)

DFA: writing to file

spanning tree:
L(L(@(@(@(0, S(S(|F(L(0))))), @(S(S(F)), 0)), S(@(|G(0), G)))))

readback:
let F = λx. x
in λy. let G = y
       in λz. z /z. /y. F (/z. /y. F z) /z. G G

minimised DFA: writing to file

spanning tree:
L(L(@(@(@(|H(0), |G(S(S(L(0))))), @(G, H)), S(@(|F(0), F)))))

readback:
λx. let F = x
    in λy. let H = y
               G = λz. z
           in H G (G H) (F F)
```

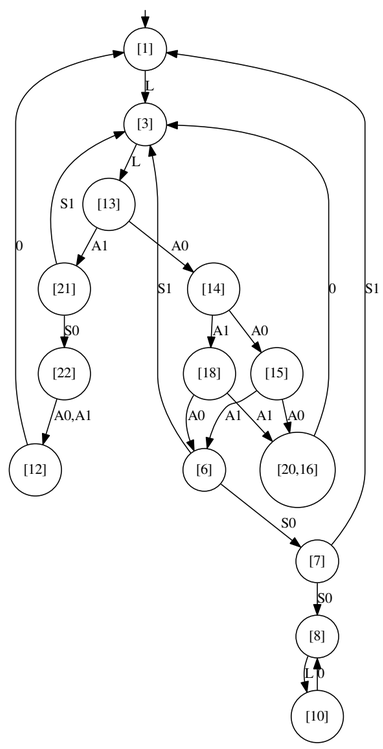




§ B.6 (L_2 from example 3.6.18)**.** This term has different translations for $[\![\cdot]\!]_{\mathcal{T}}$ and $[\![\cdot]\!]_{\mathcal{T}}^{\min}$. Thus we provide the output for both translations. See also, and compare with, the stepwise translations in appendix A.

```
λ-letrec-term:
λx. λy. let I = λz. z
            f = x
        in y I (I y) (f f)

translation used: maximal prefix lengths but eager scope

scoped (adbmals):
λx. λy. let I = /y. /x. λz. z
            f = x
        in y I (I y) /y. f f

scoped (nameless):
λ. λ. let I = S(S((λ. 0)))
          f = 0
      in 0 I (I 0) S((f f))

derivation:
----- 0                          ------------ 0                                  ------------ 0
(z) z                            (x{f} y{I}) y    (x{f} y{I}) I    (x{f} y{I}) I  (x{f} y{I}) y        (x{f}) f  (x{f}) f
-------- λ                       ------------------------------ @  ------------------------------ @    ----------------- @
() λz. z                         (x{f} y{I}) y I                  (x{f} y{I}) I y                     (x{f}) f f
------------ S                   ------------------------------------------------------------------ @  ------------------ S
(x{f}) λz. z                     (x{f} y{I}) y I (I y)                                                (x{f} y{I}) f f
---------------- S  -------- 0   ------------------------------------------------------------------------------------------ @
(x{f} y{I}) λz. z    (x{f}) x    (x{f} y{I}) y I (I y) (f f)
------------------------------------------------------------------------------------------------------------------------- let
(x y) let I = λz. z
          f = x
      in y I (I y) (f f)
------------------------------------------------------------------------------------------------------------------------- λ
(x) λy. let I = λz. z
            f = x
        in y I (I y) (f f)
------------------------------------------------------------------------------------------------------------------------- λ
() λx. λy. let I = λz. z
               f = x
           in y I (I y) (f f)

DFA: writing to file

spanning tree:
L(L(@(@(@(0, |F(S(S(L(0)))))), @(F, 0)), S(@(|G(0), G)))))

readback:
λx. let G = x
    in λy. let F = /y. /x. λz. z
           in y F (F y) /y. G G

minimised DFA: writing to file

spanning tree:
L(L(@(@(@(@(|H(0), |F(S(S(L(0)))))), @(F, H)), S(@(|G(0), G)))))

readback:
λx. let G = x
    in λy. let H = y
               F = λz. z
           in H F (F H) (G G)
```





```
λ-letrec-term:
λx. λy. let I = λz. z
            f = x
        in y I (I y) (f f)

translation used: minimal prefix lengths

scoped (adbmals):
λx. λy. let I = λz. z
            f = x
        in y /y. /x. I (/y. /x. I y) /y. f f

scoped (nameless):
λ. λ. let I = λ. 0
          f = 0
      in 0 S(S(I)) (S(S(I)) 0) S((f f))

derivation:
                                                 (*{I}) I             (*{I}) I
                                                 ------------ S       ------------ S
                                                 (*{I} x{f}) I        (*{I} x{f}) I
                                 -------------- 0 ------------- S     ------------- S  -------------- 0
                                 (*{I} x{f} y) y  (*{I} x{f} y) I     (*{I} x{f} y) I  (*{I} x{f} y) y      (*{I} x{f}) f  (*{I} x{f}) f
                                 ------------------------------- @    ------------------------------ @    ---------------------------- @
                                 (*{I} x{f} y) y I                    (*{I} x{f} y) I y                    (*{I} x{f}) f f
---------- 0                     ------------------------------------------------------------------------ @  ---------------------------- S
(*{I} z) z                       (*{I} x{f} y) y I (I y)                                                   (*{I} x{f} y) f f
---------- λ  -------------- 0   ------------------------------------------------------------------------------------------------------- @
(*{I}) λz. z    (*{I} x{f}) x    (*{I} x{f} y) y I (I y) (f f)
------------------------------------------------------------------------------------------------------------------------------------ let
(x y) let I = λz. z
          f = x
      in y I (I y) (f f)
------------------------------------------------------------------------------------------------------------------------------------ λ
(x) λy. let I = λz. z
            f = x
        in y I (I y) (f f)
------------------------------------------------------------------------------------------------------------------------------------ λ
() λx. λy. let I = λz. z
               f = x
           in y I (I y) (f f)

DFA: writing to file

spanning tree:
L(L(@(@(@(0, S(S(|F(L(0)))))), @(S(S(F)), 0)), S(@(|G(0), G)))))

readback:
let F = λz. z
in λx. let G = x
       in λy. y /y. /x. F (/y. /x. F y) /y. G G

minimised DFA: writing to file

spanning tree:
L(L(@(@(@(@(|H(0), |G(S(S(L(0)))))), @(G, H)), S(@(|F(0), F)))))

readback:
λx. let F = x
    in λy. let H = y
               G = λz. z
           in H G (G H) (F F)
```