

Reducing partial applications of

$$f = \lambda u \lambda v (u (f v)) = \Lambda \Lambda (\#1 (f \#0))$$

$$(f v)$$



$$\lambda v^1 ((\Lambda \Lambda (\#1 (f \#0))) v) v^1$$

Reducing partial applications of

$$f = \lambda u \lambda v (u (f v)) = \Lambda \Lambda (\#1 (f \#0))$$

$$\begin{array}{c} (f v) \\ \downarrow \\ \lambda v^1 ((\Lambda \Lambda (\#1 (f \#0)) v) v^1) \\ \downarrow \\ \lambda v^1 (v (f v^1)) \\ \downarrow \\ \lambda v^1 (v \lambda v^2 ((\Lambda \Lambda (\#1 (f \#0)) v^1) v^2)) \end{array}$$

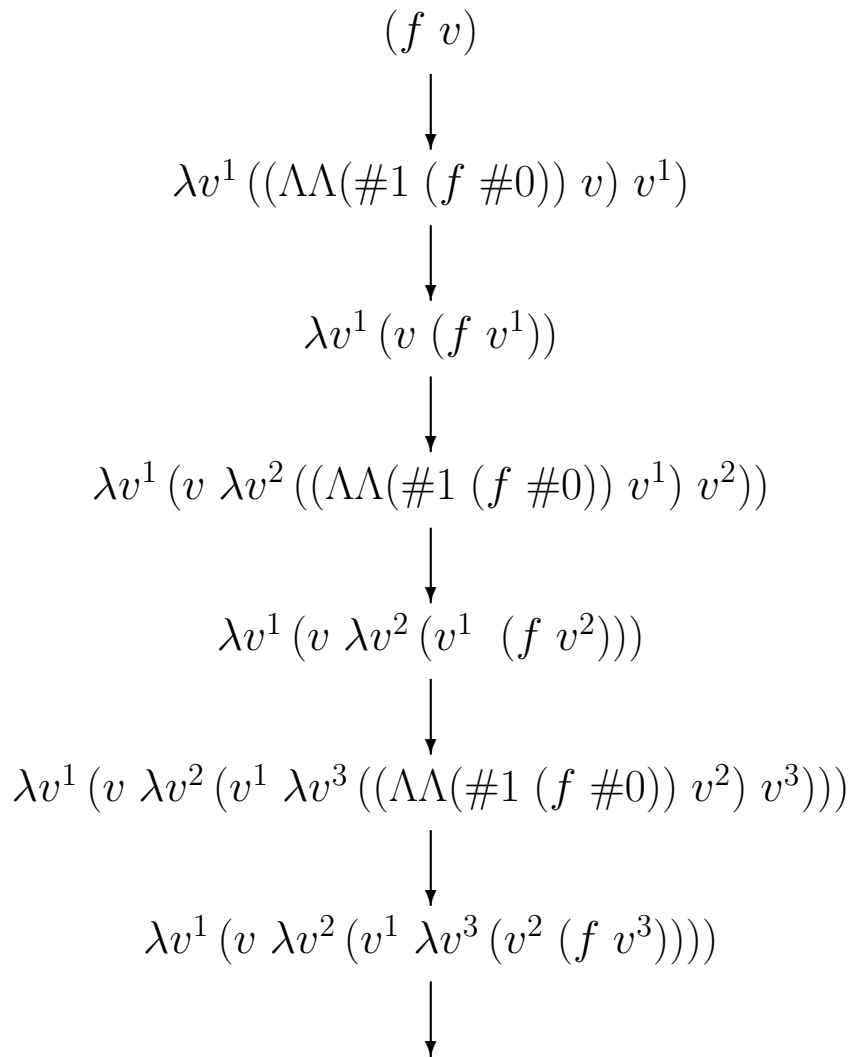
Reducing partial applications of

$$f = \lambda u \lambda v (u (f v)) = \Lambda \Lambda (\#1 (f \#0))$$

$$\begin{array}{c} (f v) \\ \downarrow \\ \lambda v^1 ((\Lambda \Lambda (\#1 (f \#0)) v) v^1) \\ \downarrow \\ \lambda v^1 (v (f v^1)) \\ \downarrow \\ \lambda v^1 (v \lambda v^2 ((\Lambda \Lambda (\#1 (f \#0)) v^1) v^2)) \\ \downarrow \\ \lambda v^1 (v \lambda v^2 (v^1 (f v^2))) \\ \downarrow \\ \lambda v^1 (v \lambda v^2 (v^1 \lambda v^3 ((\Lambda \Lambda (\#1 (f \#0)) v^2) v^3))) \end{array}$$

Reducing partial applications of

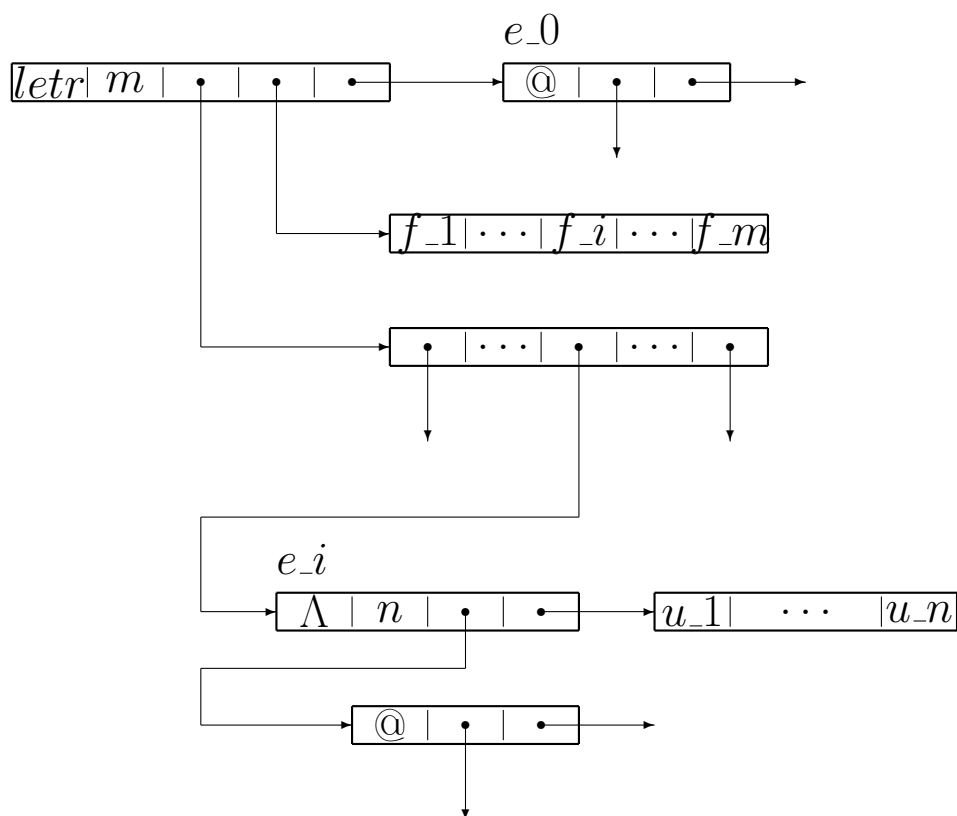
$$f = \lambda u \lambda v (u (f v)) = \Lambda \Lambda (\#1 (f \#0))$$



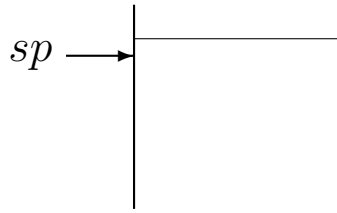
## Mutually recursive function definitions

```
( letrec ( ( f_1 e_1 ) ... ( f_i e_i ) ... ( f_m e_m )
```

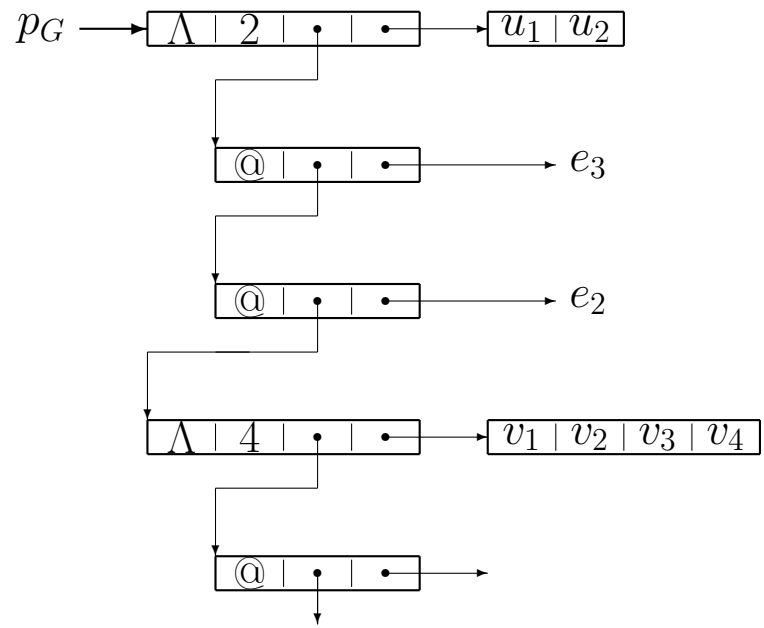
```
  e_i = ( lambda ( u_1 ... u_n ) ( ( f_j ... ) ... ) )
```

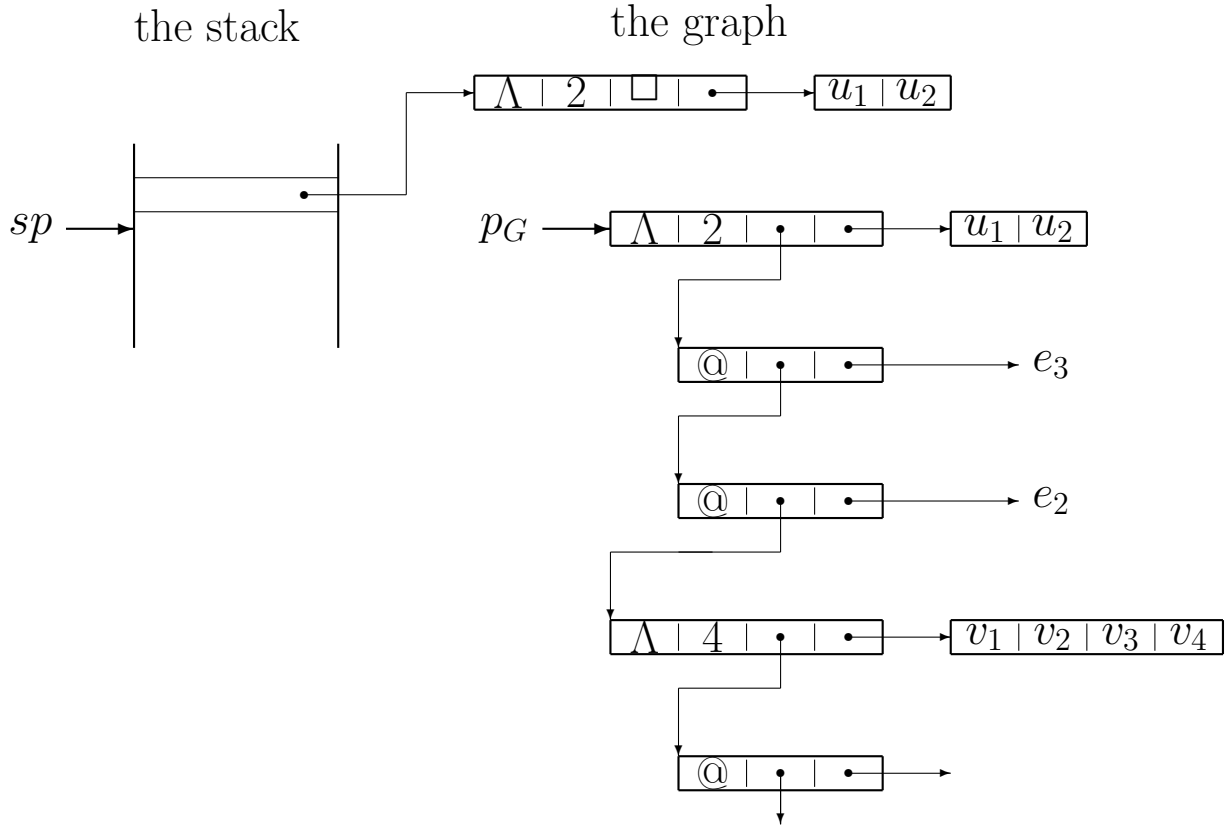


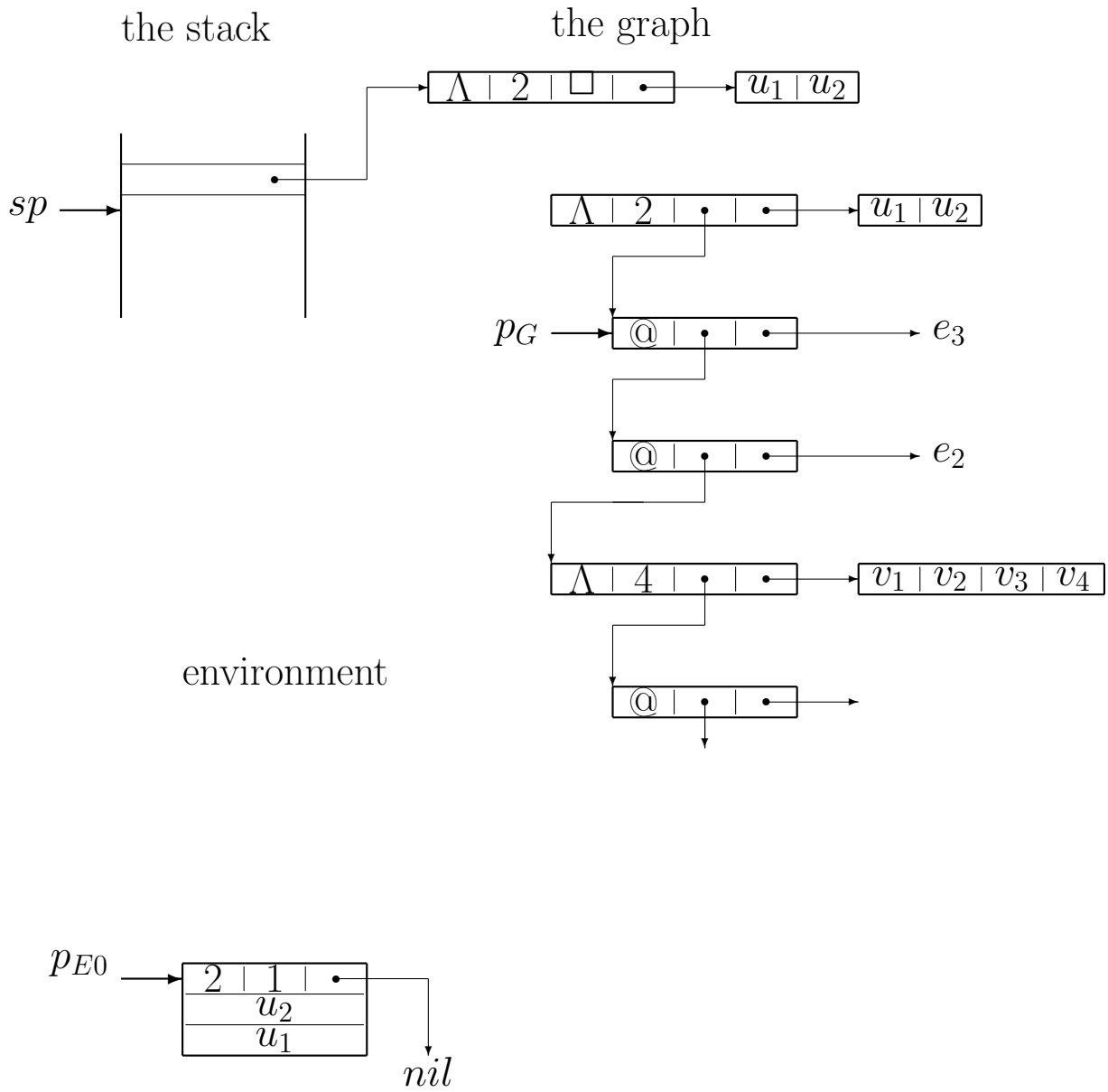
the stack

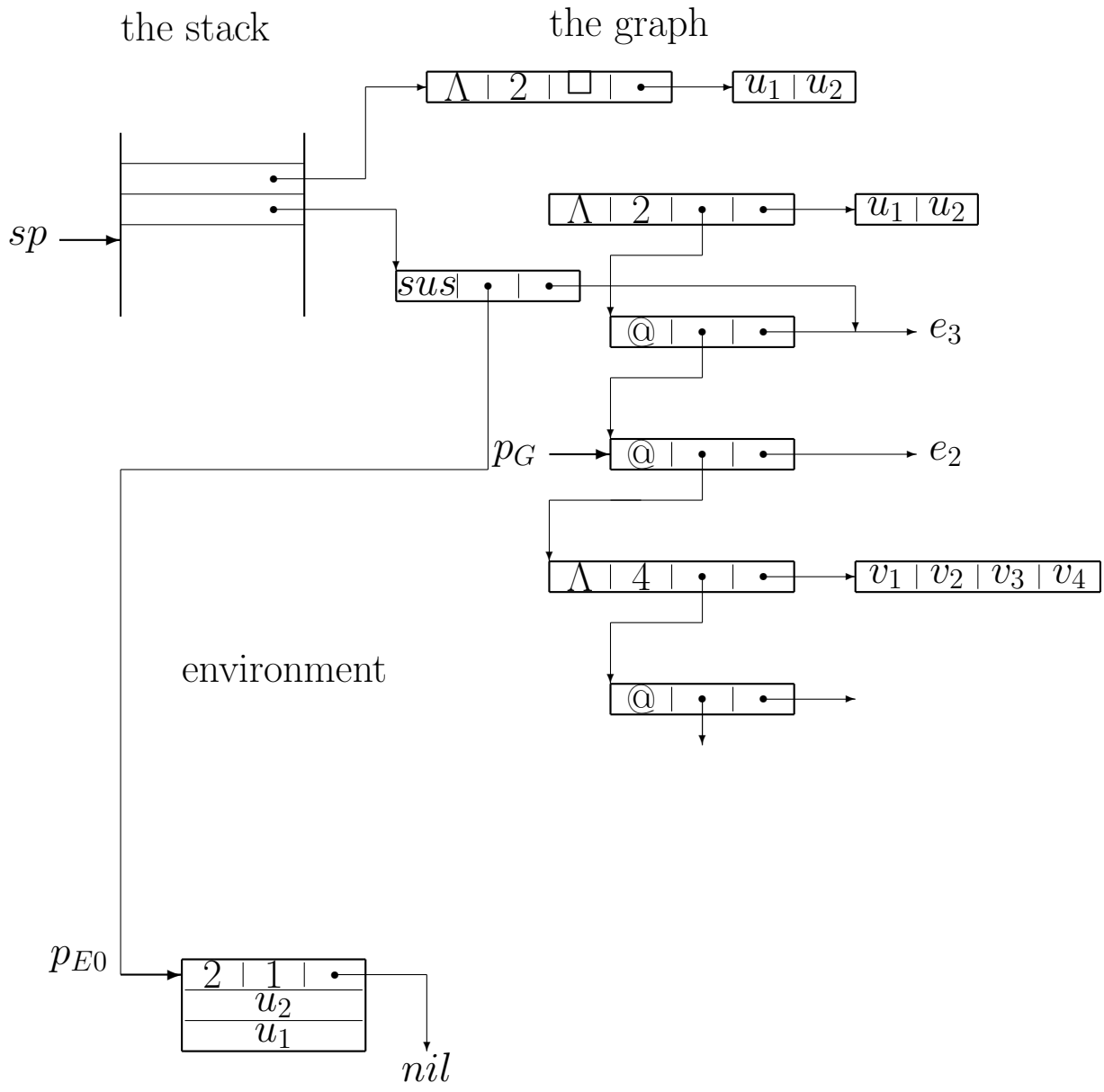


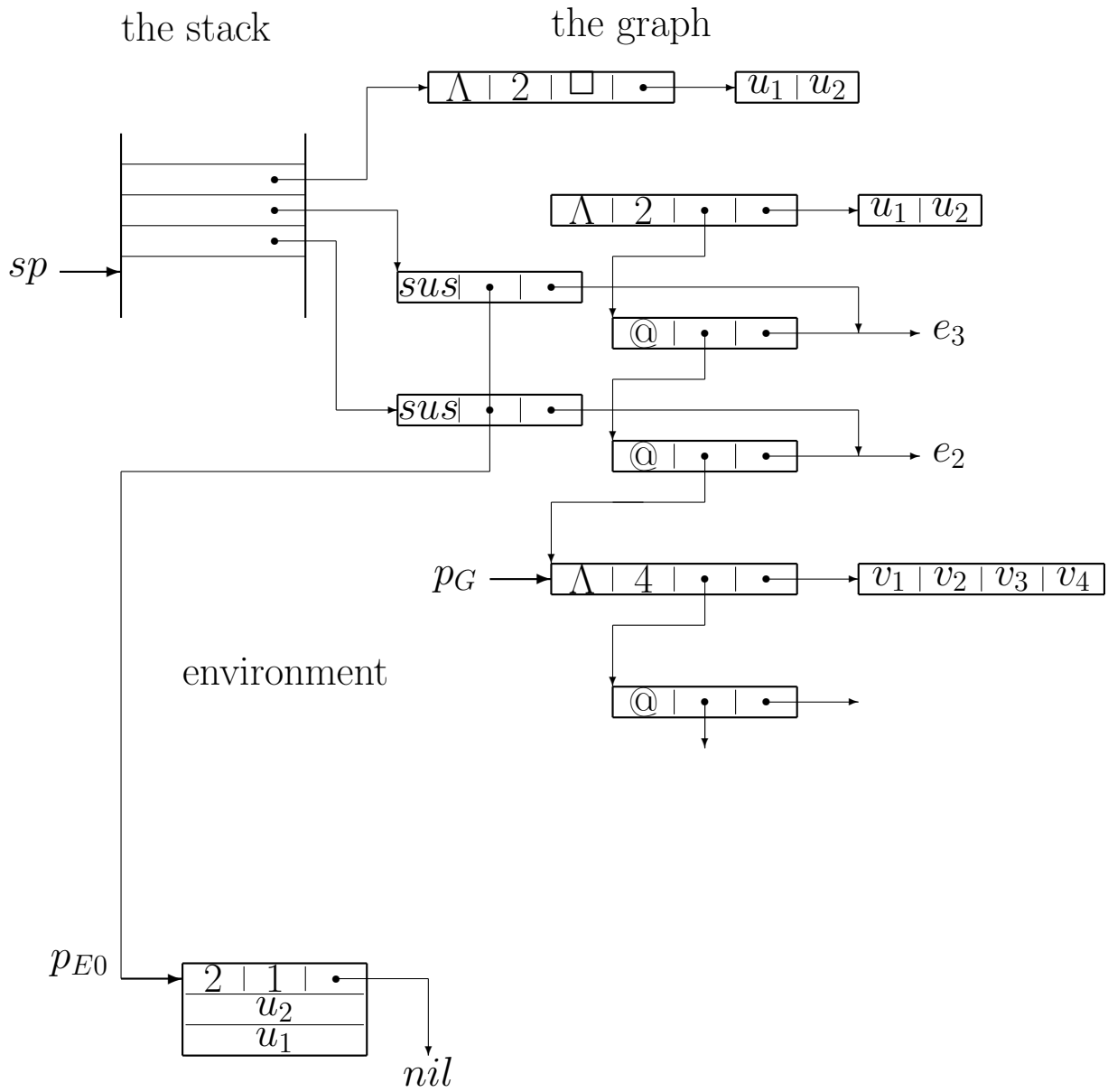
the graph

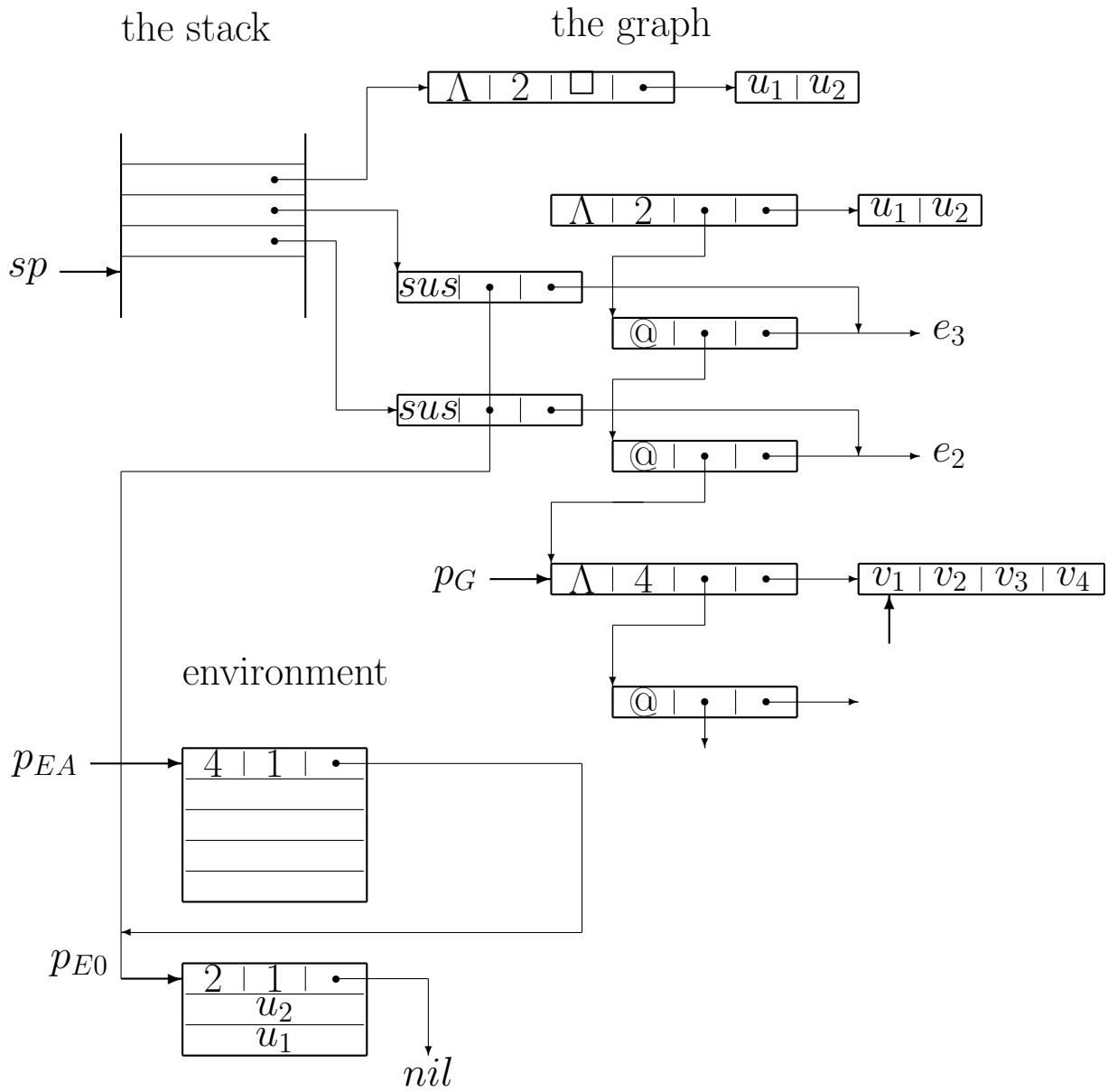


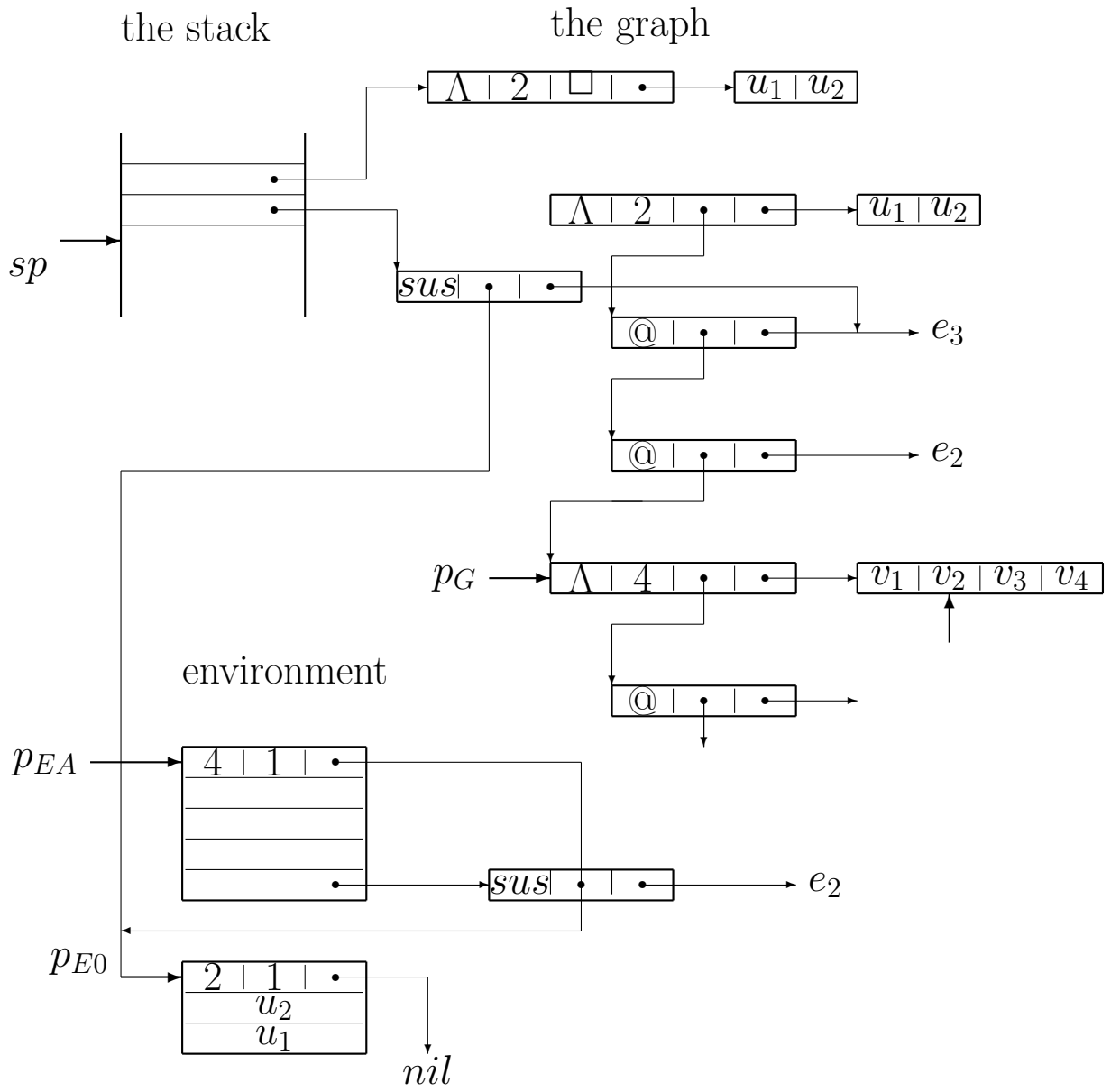


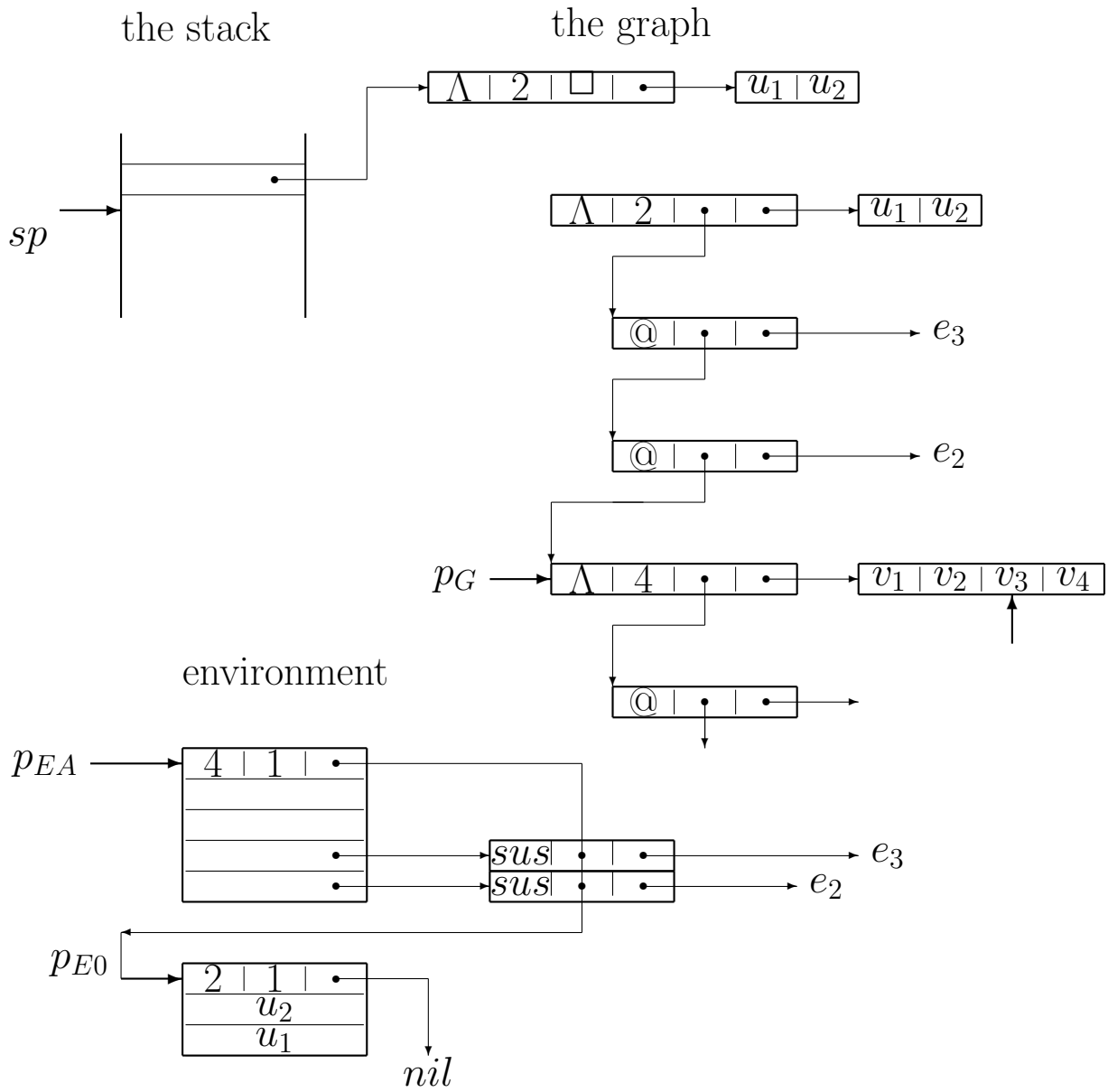


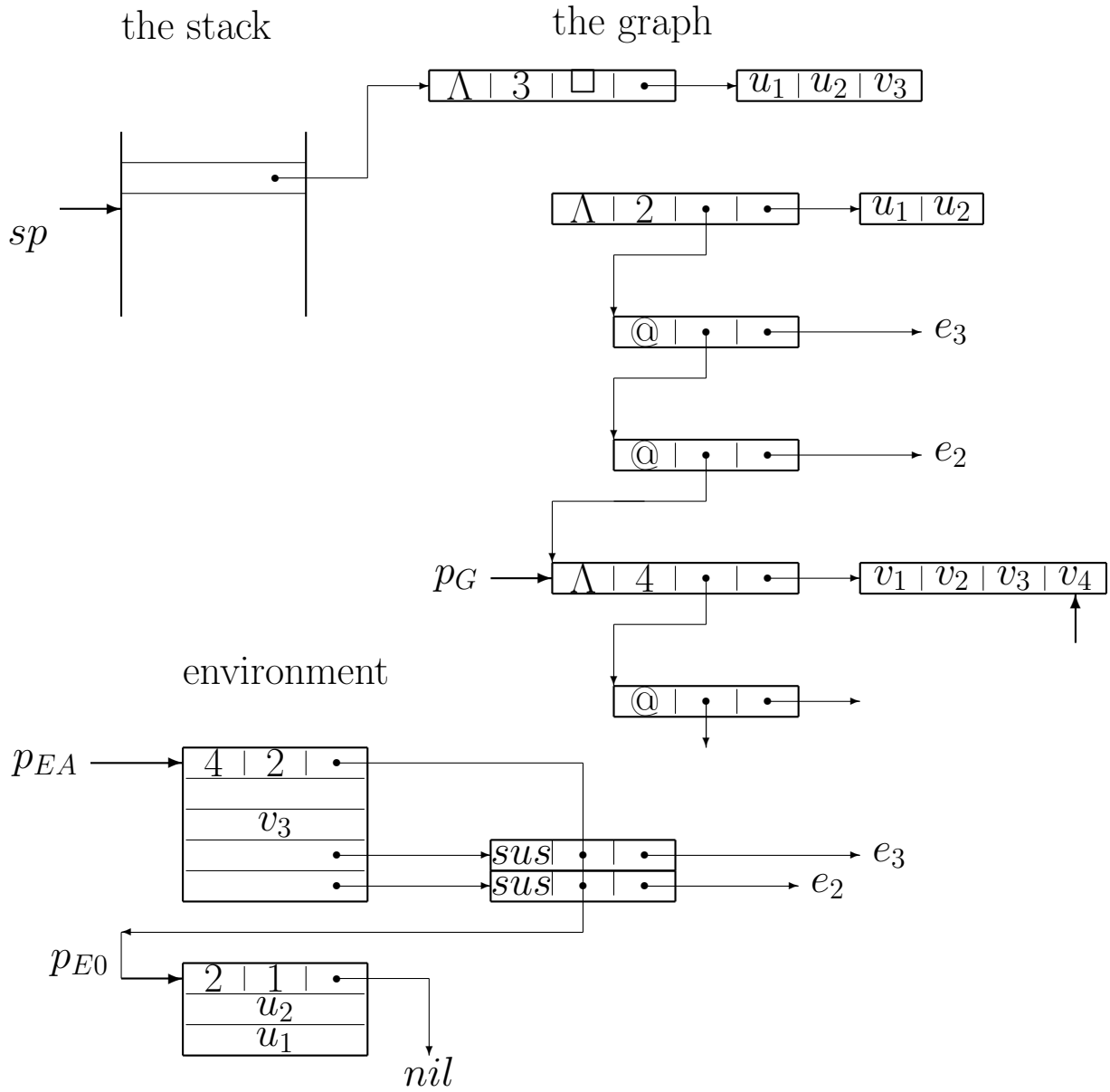


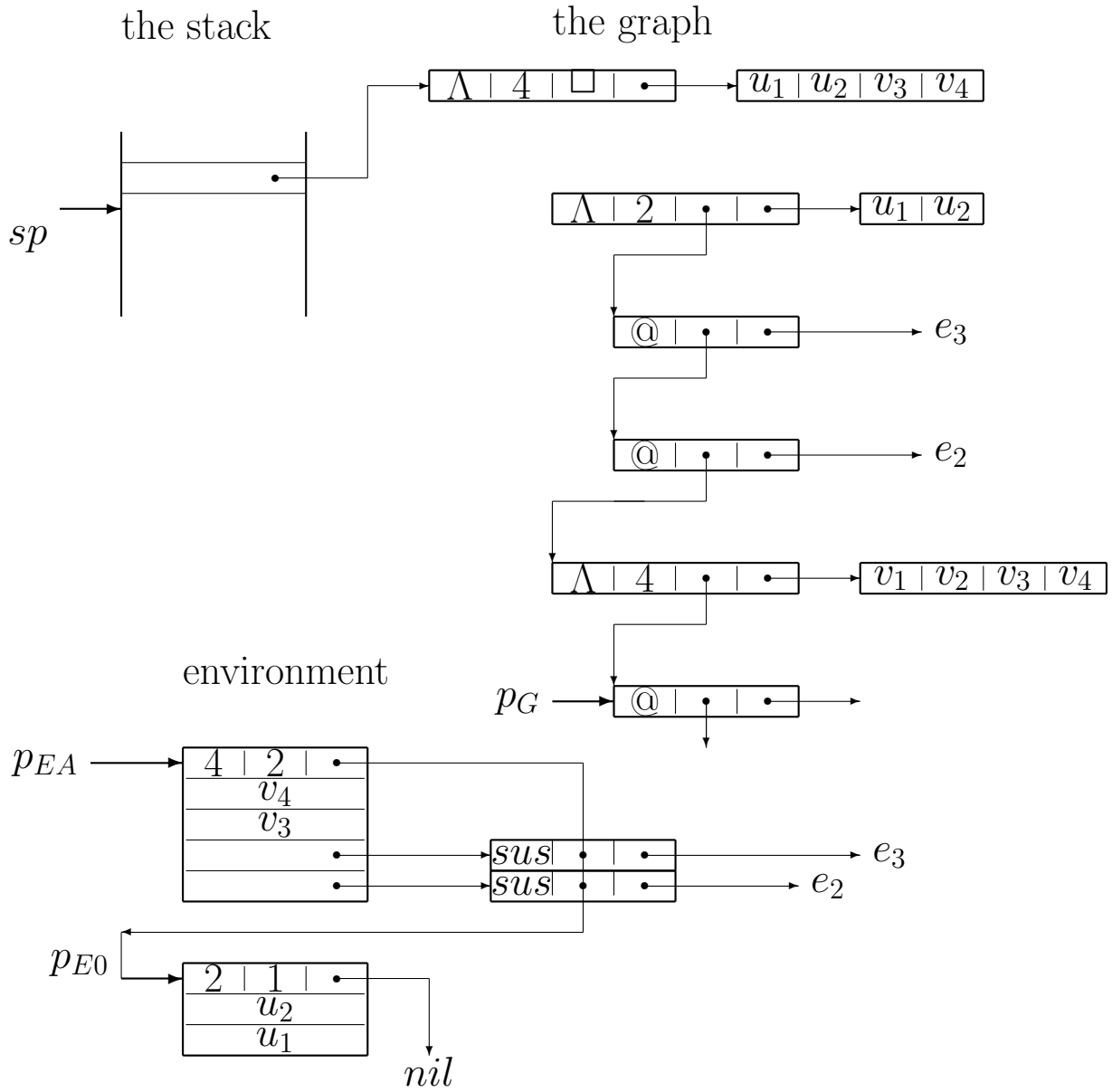












## Conclusion

... well, we have demonstrated that a fully normalizing  $\lambda$ -calculus, contrary to wide-spread dogma, can be efficiently implemented, with applications in

- > symbolic computations involving free variables,
- > high-level code optimization
- > term rewrite / proof systems
- ...
- > teaching  $\lambda$ -calculus