Multithreaded-Cartesian Abstract Interpretation of Multithreaded Recursive Programs is Polynomial

Alexander Malkis

malkis@in.tum.de

Technische Universität München

9th International Workshop on Reachability Problems

Acknowledgements:

- Neil D. Jones,
- IMDEA Software Institute, Manfred Broy's research group at Technische Universität München,
- H. Seidl, J. Esparza, C. Broadbent, L. Mauborgne, A. Podelski, ...

Multithreaded programs

Program: $(\mathsf{Glob}, \mathsf{Frame}, \mathsf{init}, (\boxplus_t, \boxminus_t, \boxplus_t)_{t<n})$

- n: number of threads (ordinal).
- Interleaving semantics.
- Local state = stack contents;
  $\mathsf{Loc} = \mathsf{Frame}^+$.
- Thread state: $(g, \mathsf{stack\_word}) \in \mathsf{Glob} \times \mathsf{Loc}$
- Program state:
  $(\mathsf{shared}, (\mathsf{stack\_word}_0, \mathsf{stack\_word}_1, \mathsf{stack\_word}_2, \ldots))$;
  $\mathsf{State} = \mathsf{Glob} \times \mathsf{Loc}^n$.
- Initially, each stack contains exactly one letter.

For each $t < n$:

- $\boxplus_t \subseteq (\mathsf{Glob}\times\mathsf{Frame}) \times (\mathsf{Glob}\times\mathsf{Frame}\times\mathsf{Frame})$,
- $\boxminus_t \subseteq (\mathsf{Glob}\times\mathsf{Frame}) \times (\mathsf{Glob}\times\mathsf{Frame})$,
- $\boxplus_t \subseteq (\mathsf{Glob}\times\mathsf{Frame}\times\mathsf{Frame}) \times (\mathsf{Glob}\times\mathsf{Frame})$.

Multithreaded shared-memory recursive programs

Multithreading + recursion is rare, but exists, both in the models and in real code.

- Model of the Bluetooth driver from Windows NT
- Model of the old synchronized `java.util.Vector`
- "Concurrent manipulation of binary search trees", H. T. Kung and Philip L. Lehman, 1980
- Parallel Merge Sort: merging sorted pairs of subsequences may happen in parallel for independent pairs of subsequences
- Cilkchess
- "A new multithreaded and recursive direct algorithm for parallel solution of the sparse linear systems", Ercan Selçuk Bölükbaşi, 2013
- ...

Escape undecidability through overapproximation

- Membership in the strongest inductive invariant:
  undecidable.
- Membership in a special-form, not necessarily strongest inductive invariant:
  perhaps decidable.

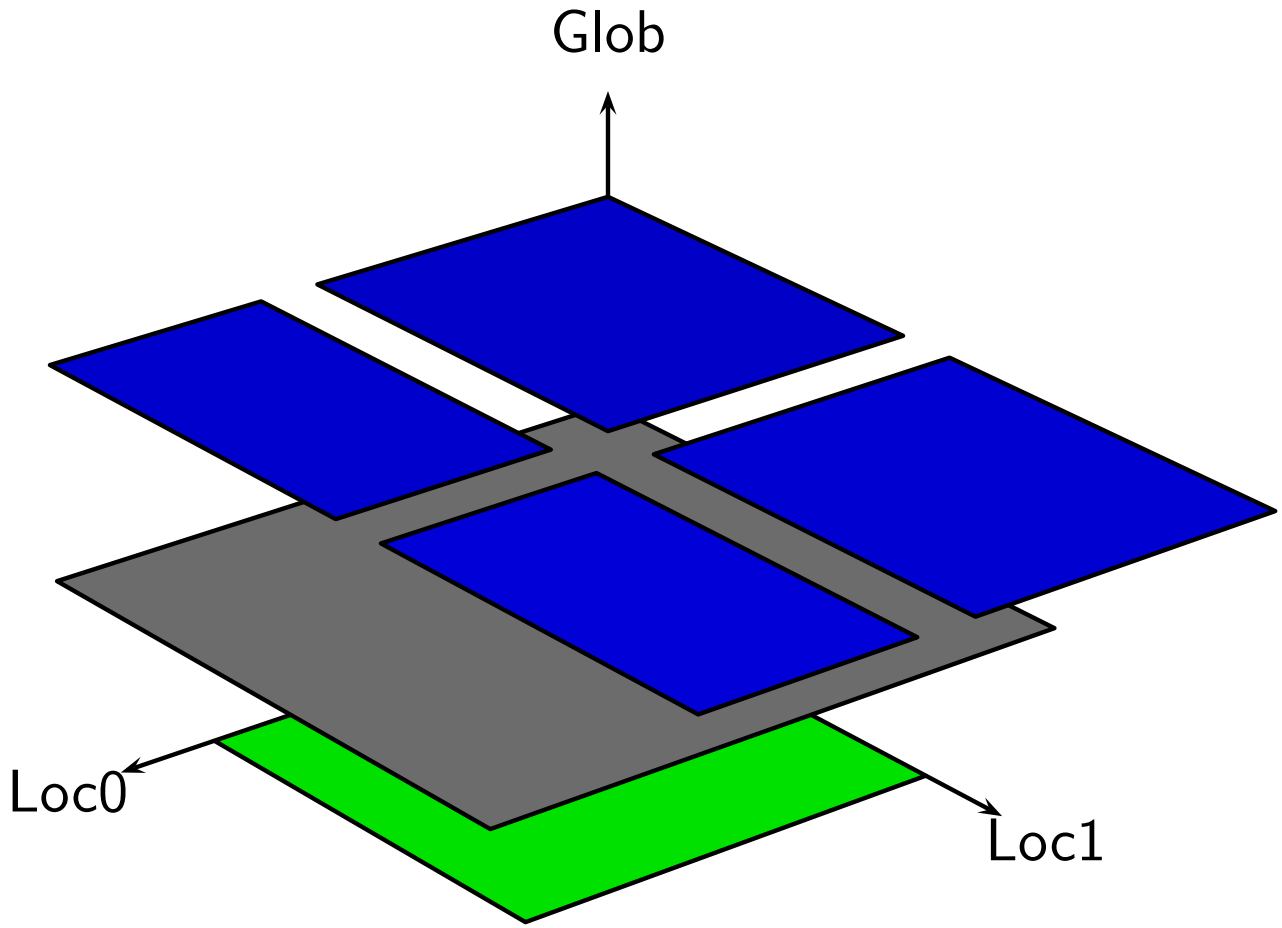
Multithreaded-Cartesian set of program states

Assume: finite $\mathsf{Glob} = \{0, 1, \ldots |\mathsf{Glob}| - 1\}$, finite n.
A set $S \subseteq \mathsf{State}$ is in multithreaded-Cartesian form iff
there are $L_{g,t} \subseteq \mathsf{Loc}$ ($g \in \mathsf{Glob}$, $t < n$) s. t.

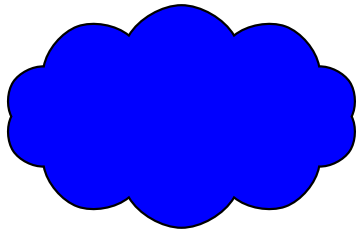
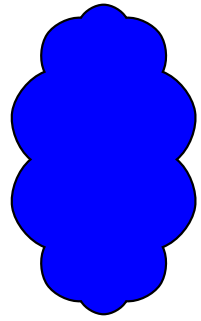
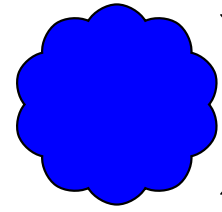
$$
\begin{array}{rcl}
S & = & \{0\} \times L_{0,0} \times \ldots \times L_{0,n-1} \\
 & & \cup \{1\} \times L_{1,0} \times \ldots \times L_{1,n-1} \\
 & & \vdots \\
 & & \cup \{|\mathsf{Glob}| - 1\} \times L_{|\mathsf{Glob}|-1,0} \times \ldots \times L_{|\mathsf{Glob}|-1,n-1} \cdot
\end{array}
$$

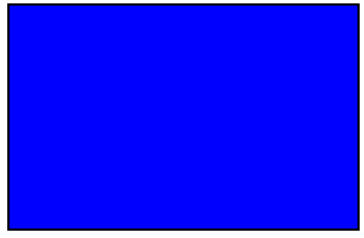
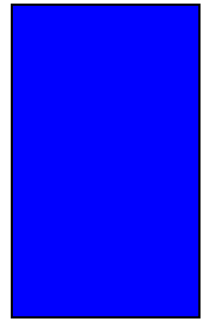
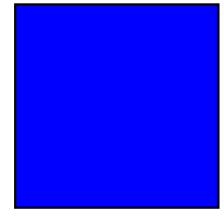
Multithreaded-Cartesian set of program states

Assume: arbitrary Glob, arbitrary n.
States $(g, \vec{\ell})$, $(g', \vec{\ell'})$ are equivalent iff $g = g'$.
A set $S \subseteq$ State is in multithreaded-Cartesian form iff the intersection of each equivalence class with S is a Cartesian product.



Multithreaded-Cartesian overapproximation

$\rho_{\mathsf{mc}}\Big(\{0\}\times$  $\cup\,\{1\}\times$  $\cup\,\{2\}\times$  $\Big)$

$=\quad \{0\}\times$  $\cup\,\{1\}\times$  $\cup\,\{2\}\times$  .

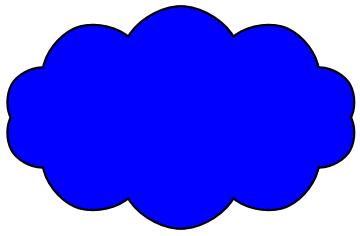
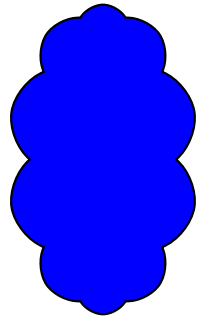
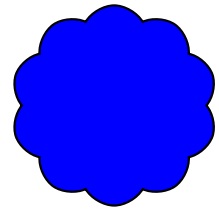
For $S \subseteq \mathsf{State}$,

$$\rho_{\mathsf{mc}}(S) \;=\; \{(g,\ell) \mid \forall\, t{<}n\ \exists\, \tilde{\ell} \in \mathsf{Loc}^n\colon\ \tilde{\ell}_t = \ell_t \wedge (g,\tilde{\ell}) \in S\}\,.$$

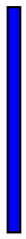
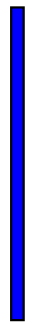
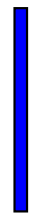
ρ_{mc} is an upper closure operator.

Multithreaded-Cartesian Galois-connection: abstraction

Concrete domain: $D = \mathfrak{P}(\text{State}) = \mathfrak{P}(\text{Glob} \times \text{Loc}^n)$.
Abstract domain: $D^\# = (\mathfrak{P}(\text{Glob} \times \text{Loc}))^n$.

$\alpha_{\text{mc}}\Big(\{0\} \times$  $\cup\ \{1\} \times$  $\cup\ \{2\} \times$  $\Big)$

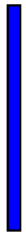
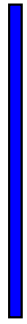
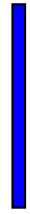
$= \big(\{0\} \times$  $\cup\ \{1\} \times$  $\cup\ \{2\} \times$  $,$

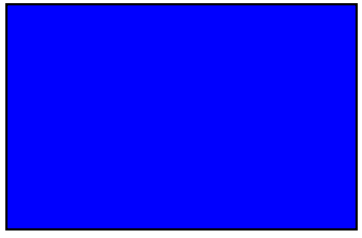
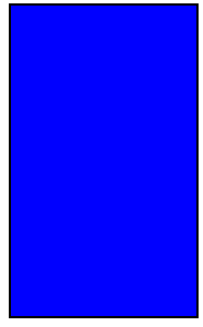
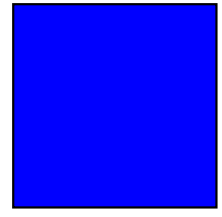
$\{0\} \times$  $\cup\ \{1\} \times$  $\cup\ \{2\} \times$  $\big).$

Abstraction $\alpha_{\text{mc}}(S) = \big(\{(g, \ell_t) \mid (g, \ell) \in S\}\big)_{t<n}$.

Multithreaded-Cartesian abstraction: concretization

$\gamma_{\mathsf{mc}}\big(\{0\}\times$  $\cup\,\{1\}\times$  $\cup\,\{2\}\times$  ,

$\{0\}\times$  $\cup\,\{1\}\times$  $\cup\,\{2\}\times$  $\big)$

$= \{0\}\times$  $\cup\,\{1\}\times$  $\cup\,\{2\}\times$  .

Concretization $\gamma_{\mathsf{mc}}\left((A_t)_{t<n}\right) = \{(g,\ell) \mid \forall\, t{<}n\colon (g,\ell_t) \in A_t\}$.

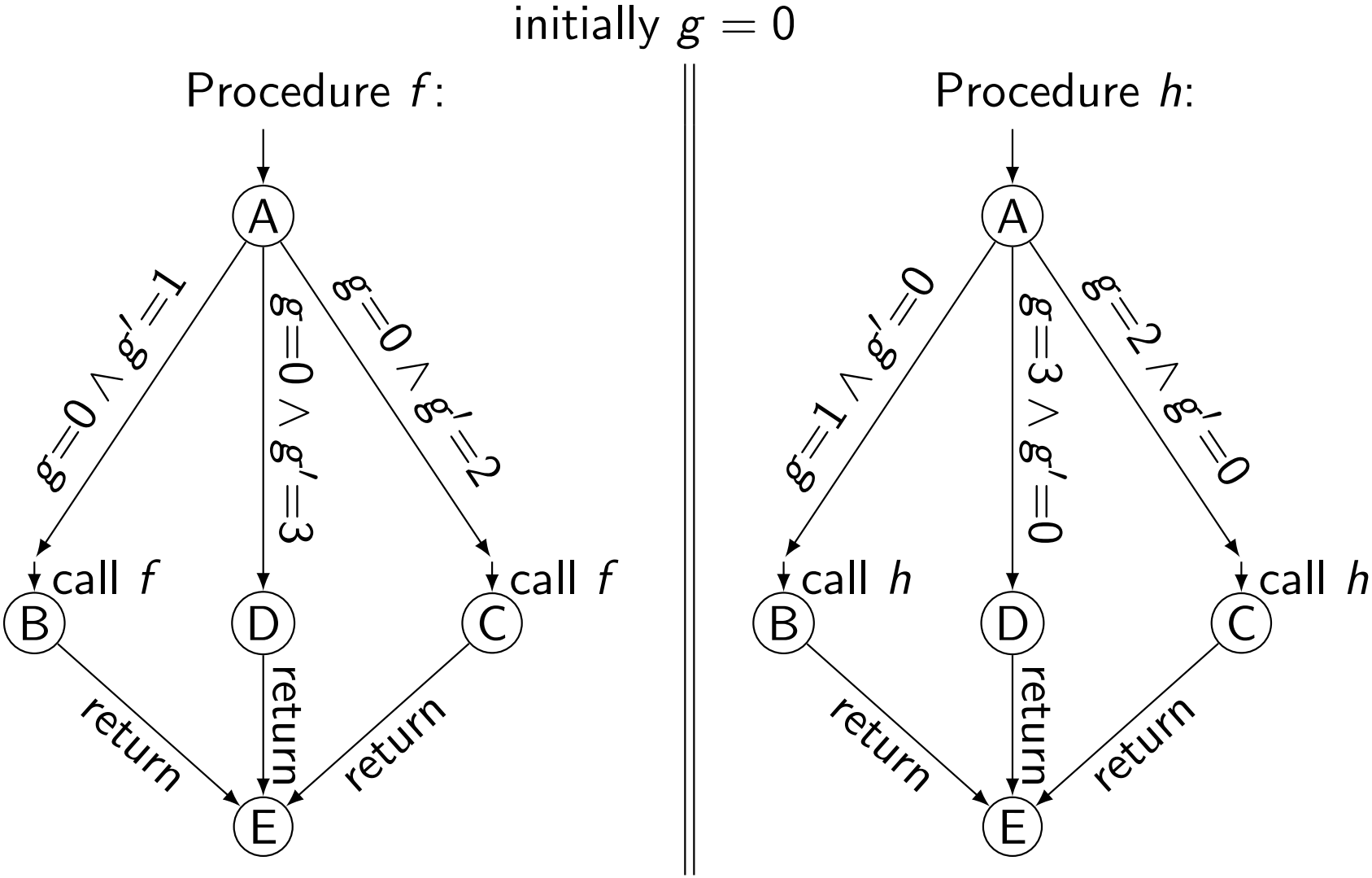
$\rho_{\mathsf{mc}} = \gamma_{\mathsf{mc}} \circ \alpha_{\mathsf{mc}}$

Multithreaded-Cartesian abstract interpretation

$$\mathsf{lfp}(\lambda\ S \in D.\ \rho_{\mathsf{mc}}(\mathsf{init} \cup \mathsf{post}(S)))$$

$$\mathsf{lfp}(\lambda\ A \in D^{\#}.\ \alpha_{\mathsf{mc}}(\mathsf{init} \cup \mathsf{post}(\gamma_{\mathsf{mc}}(A))))$$

Intricate example



The strongest inductive invariant is not context-free.

Strongest multithreaded-Cartesian inductive invariant for the example

$$S = \begin{pmatrix} & \{0\} \times (\{\mathrm{A}x, \mathrm{D}x \mid x \in \{\mathrm{B}, \mathrm{C}\}^*\} \cup \{\mathrm{B}, \mathrm{C}\}^+) \\ \cup & \{1\} \times \{\mathrm{AB}x \mid x \in \{\mathrm{B}, \mathrm{C}\}^*\} \\ \cup & \{2\} \times \{\mathrm{AC}x \mid x \in \{\mathrm{B}, \mathrm{C}\}^*\} \\ \cup & \{3\} \times (\{\mathrm{D}x \mid x \in \{\mathrm{B}, \mathrm{C}\}^*\} \cup \{\mathrm{B}, \mathrm{C}\}^+) \end{pmatrix} \times \left(\{\mathrm{A}x, \mathrm{D}x \mid x \in \{\mathrm{B}, \mathrm{C}\}^*\} \cup \{\mathrm{B}, \mathrm{C}\}^+\right) .$$

Notice:

$$(|w_1| = |w_2| = 1 \wedge (g, (w_1, w_2)) \in S) \quad \Rightarrow \quad (g = 0 \vee g = 3) .$$

The property would be proven by a multithreaded-Cartesian analysis generating (a finite representation of) S.
Viewed as a set of words, S is regular.
It can be generated!

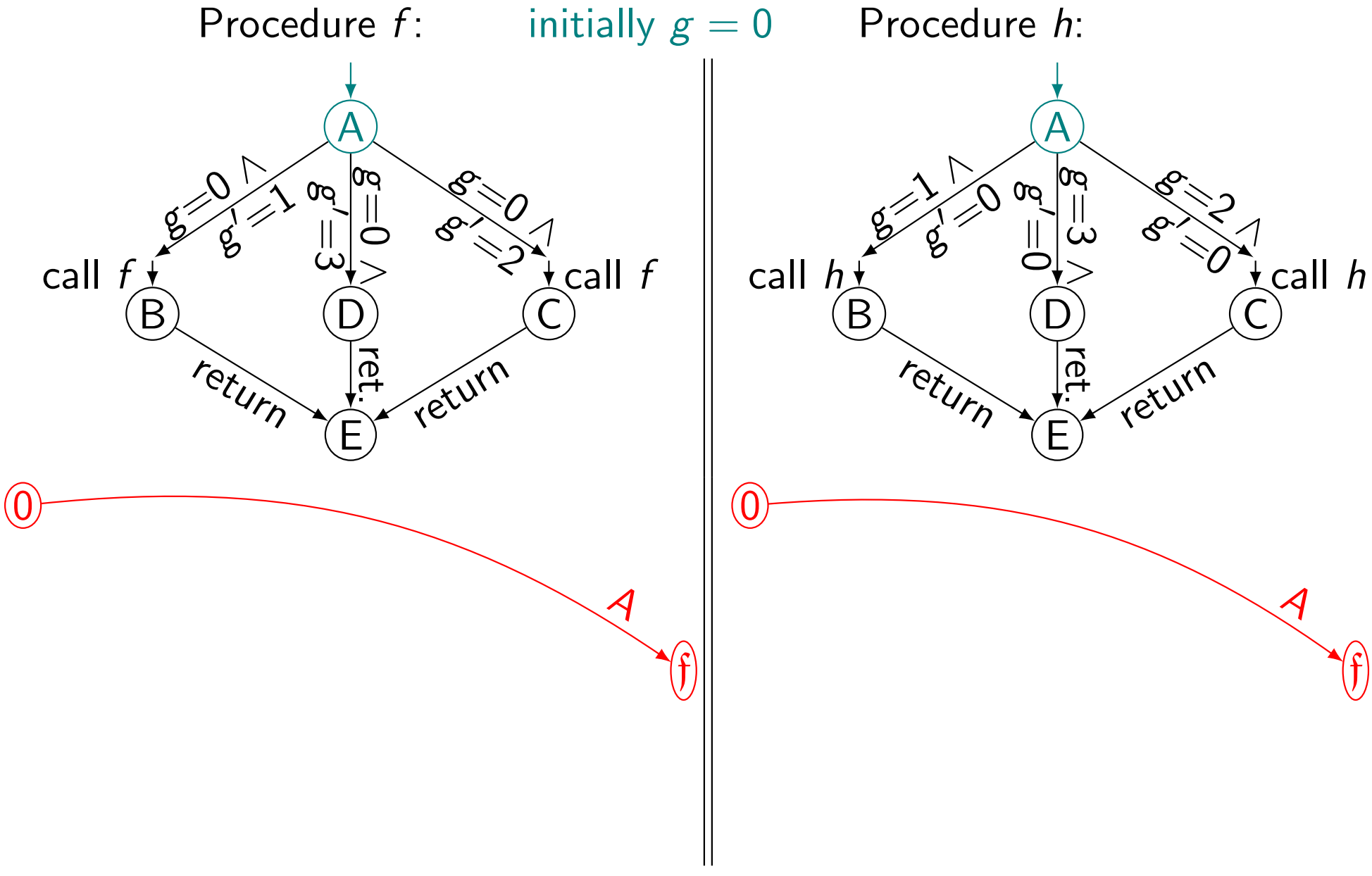
TMR algorithm

Generating a regular representation of

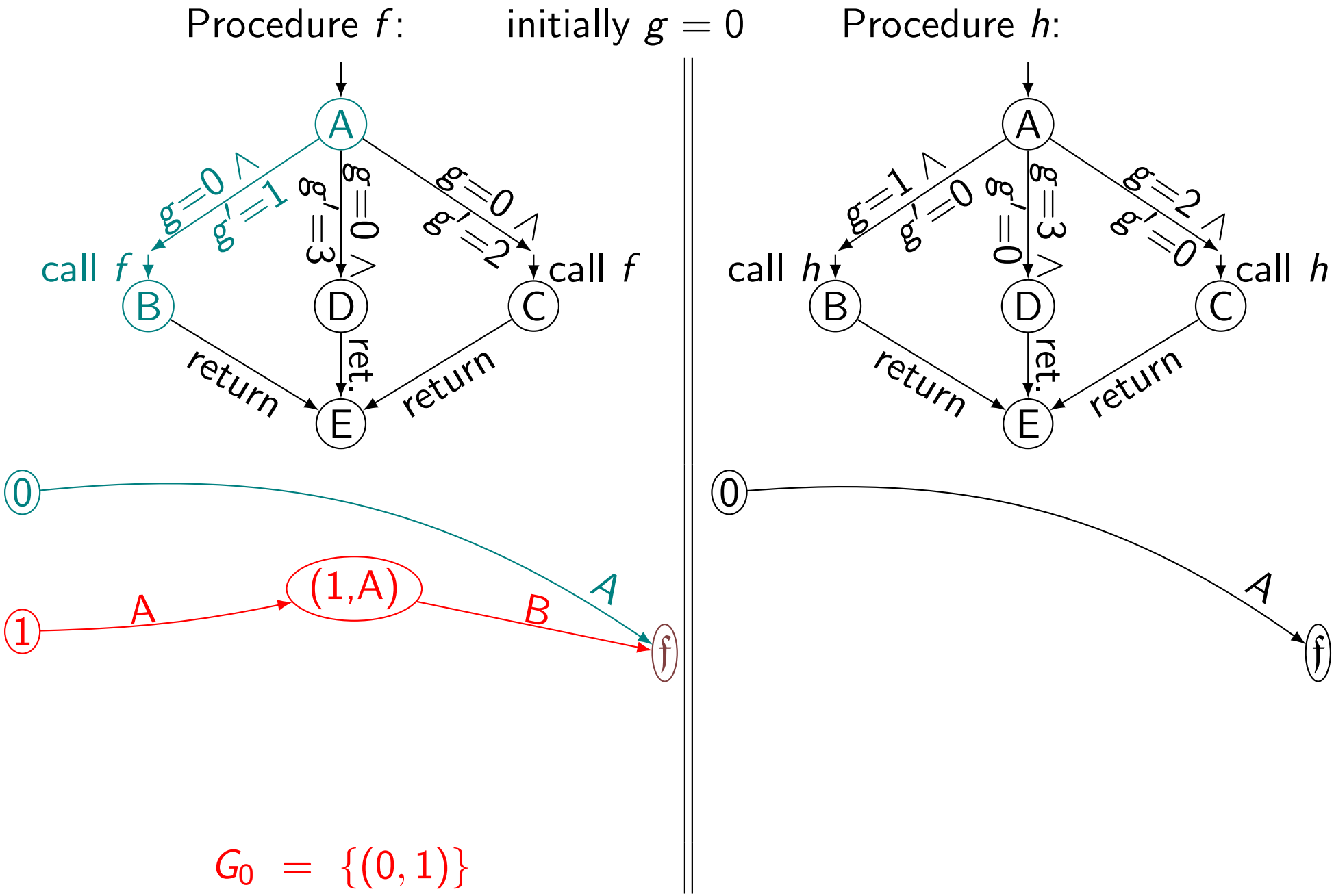
$$\mathsf{lfp}(\lambda\ S \in D.\ \rho_{\mathsf{mc}}(\mathsf{init} \cup \mathsf{post}(S)))\,.$$

- Construct n NFAs simultaneously, one per thread.
- Each NFA describes a set of thread states ($\in \mathsf{Glob} \times \mathsf{Loc}$).
- Sequentially chain the NFAs.

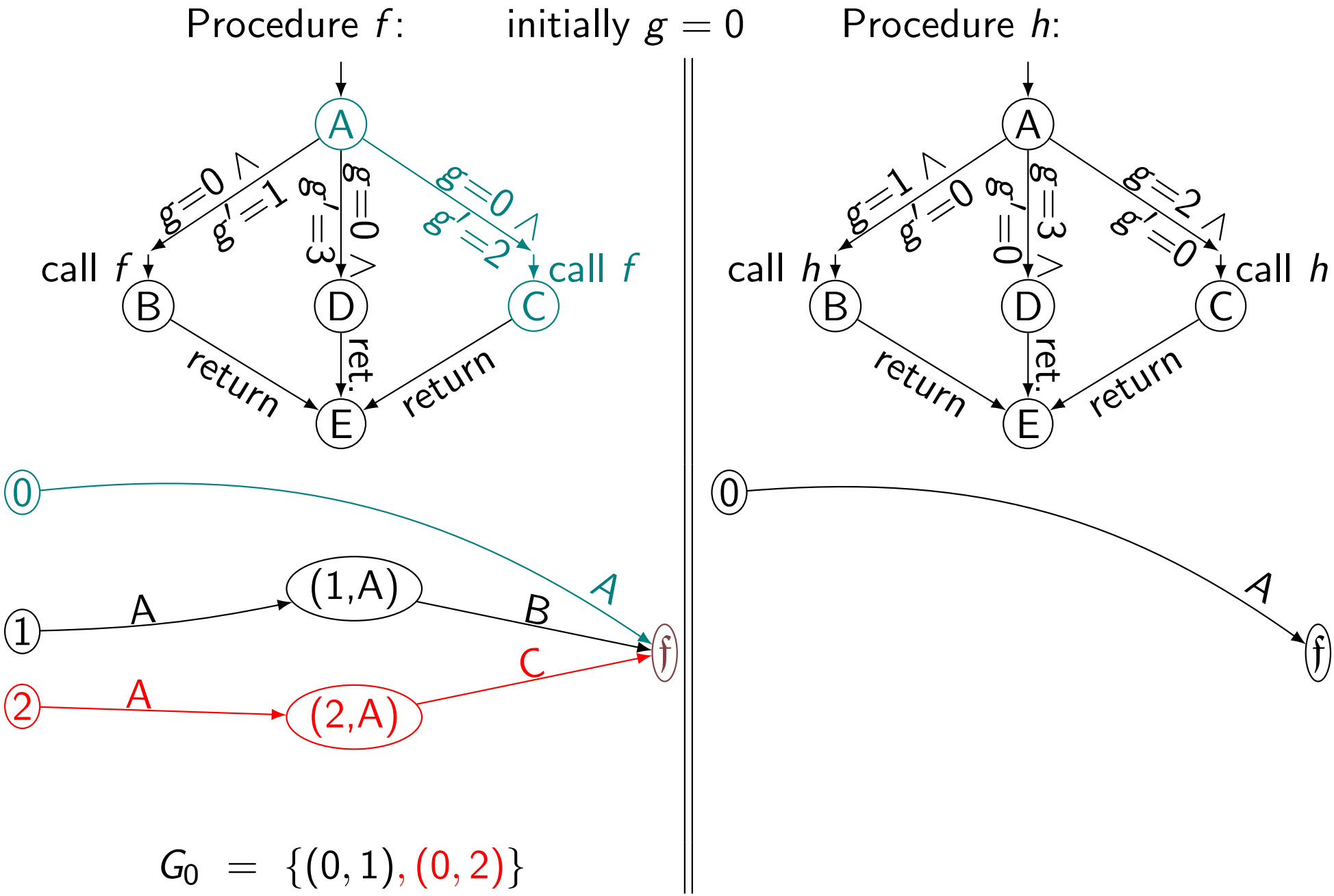
Generating automata for the left and right threads (1)



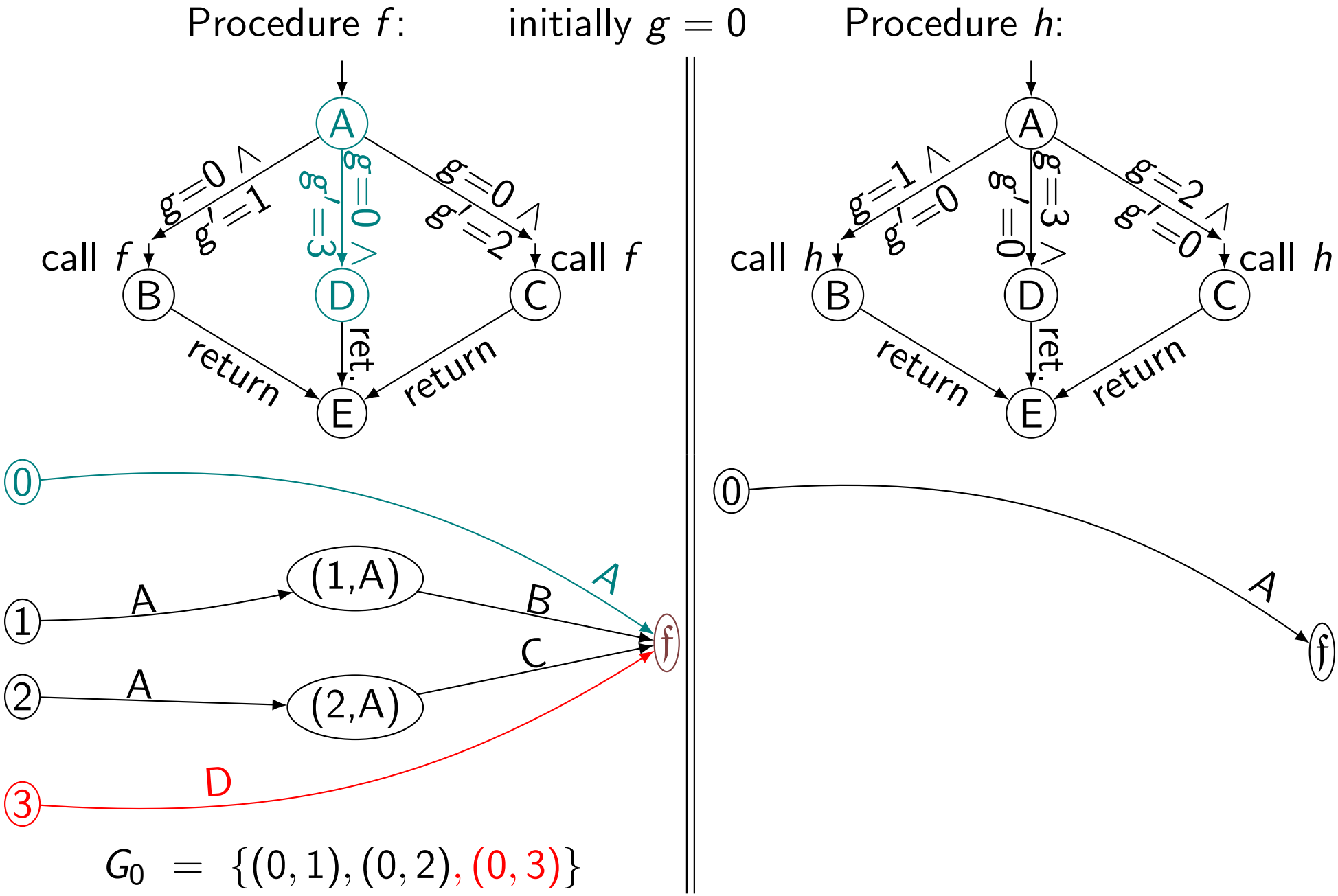
Generating automata for the left and right threads (2)



Generating automata for the left and right threads (3)

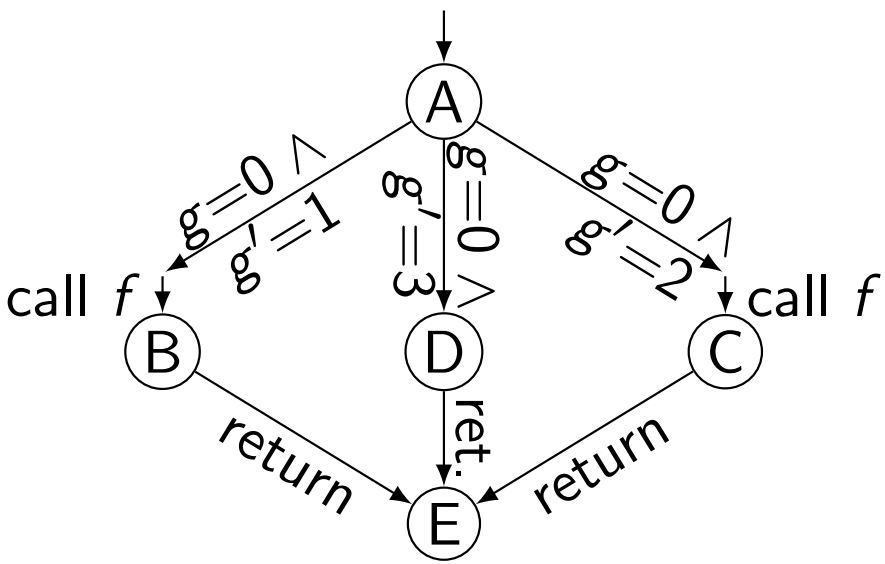


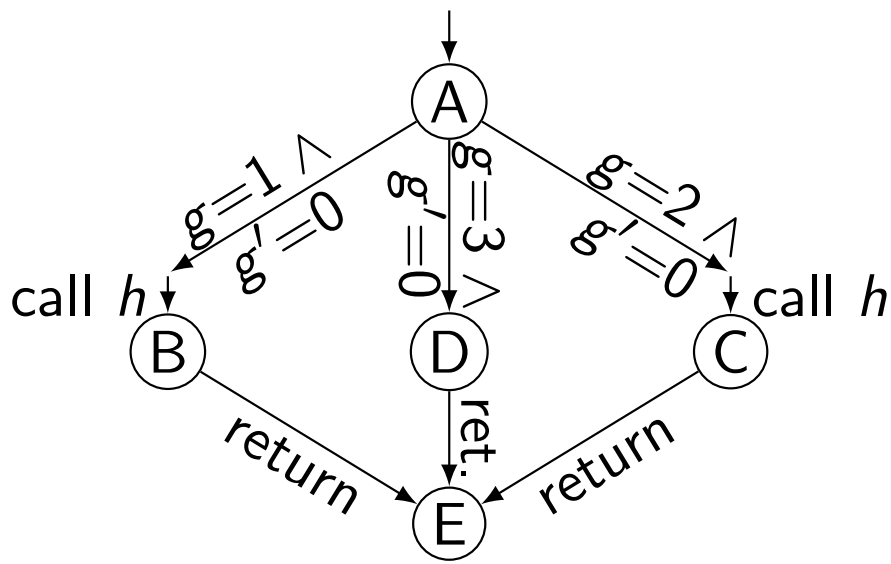
Generating automata for the left and right threads (4)

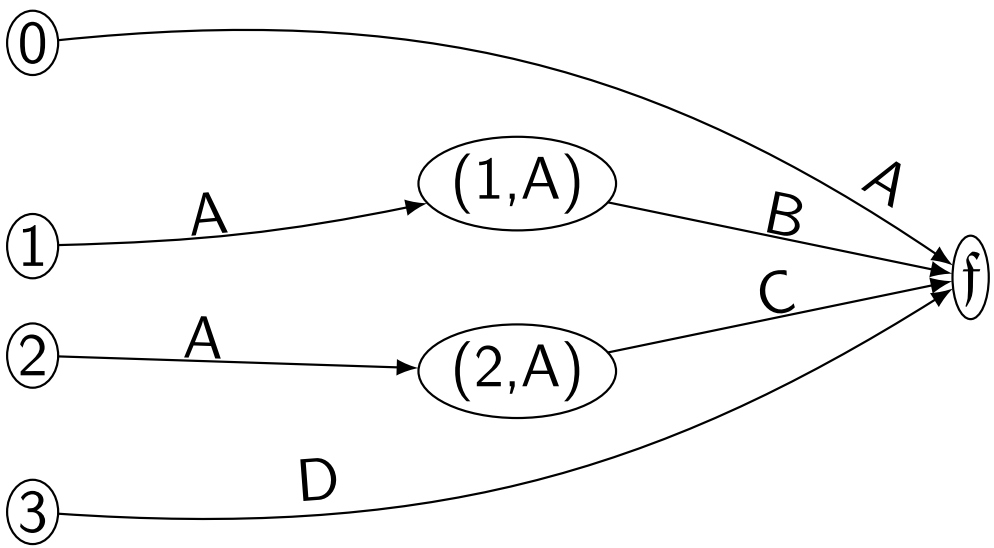


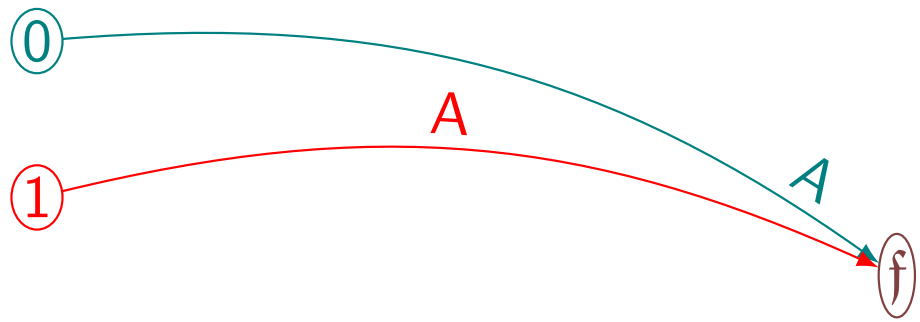
Generating automata for the left and right threads (5)

Procedure f: initially $g = 0$ Procedure h:





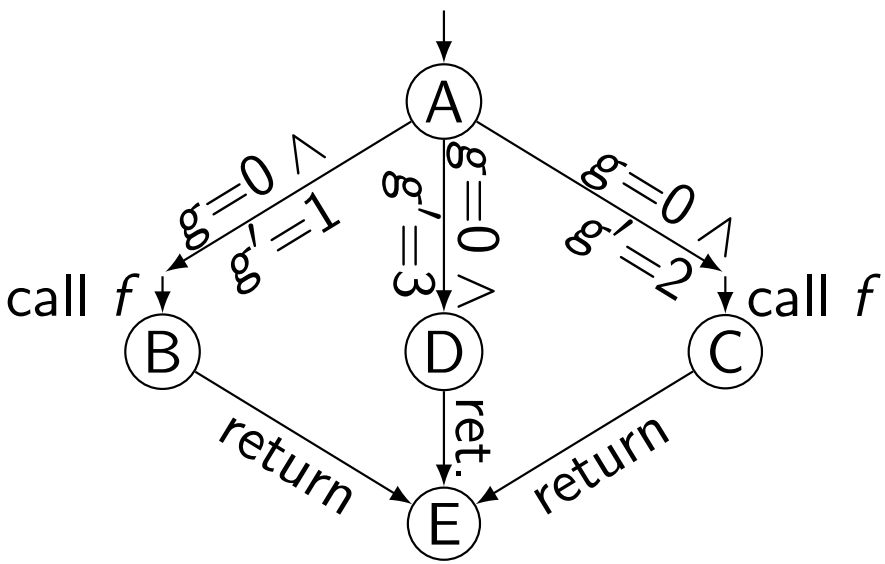


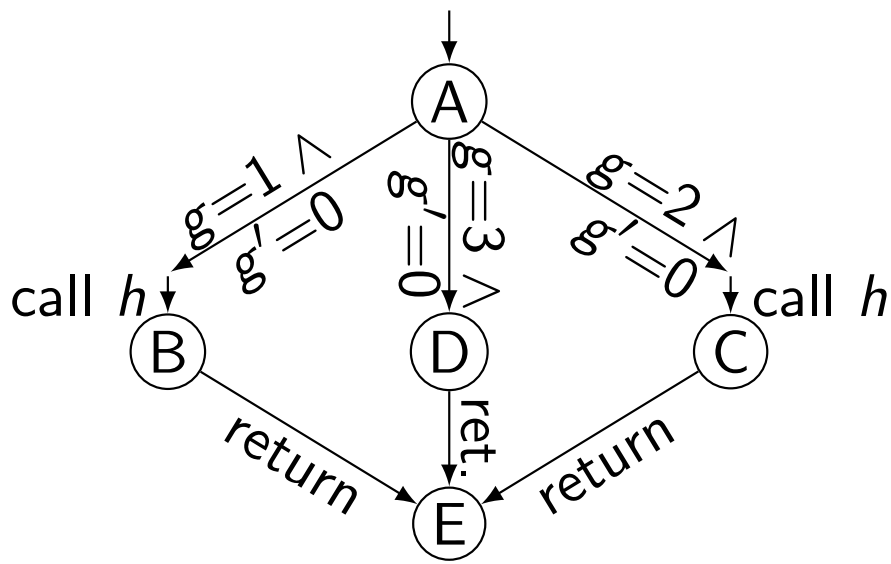


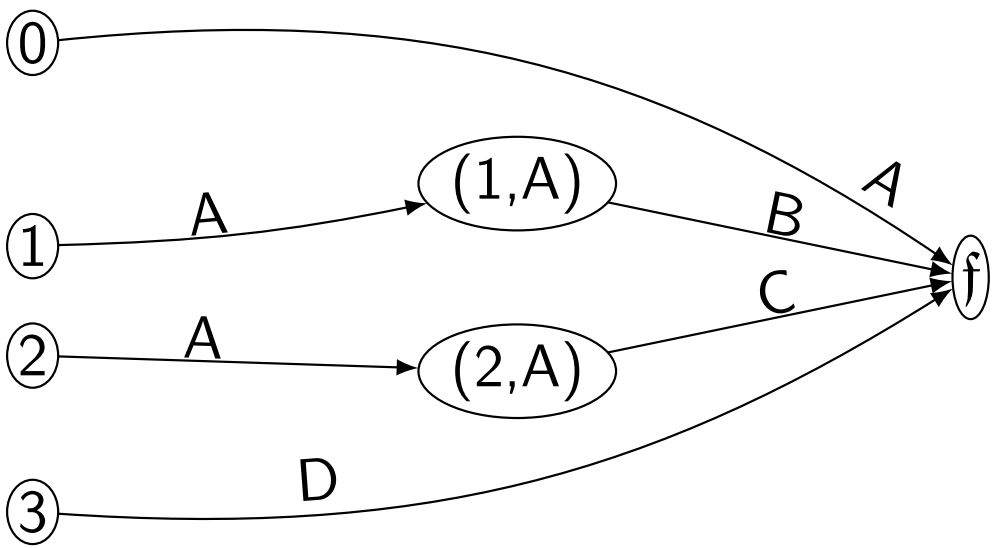
$G_0 = \{(0,1),(0,2),(0,3)\}$

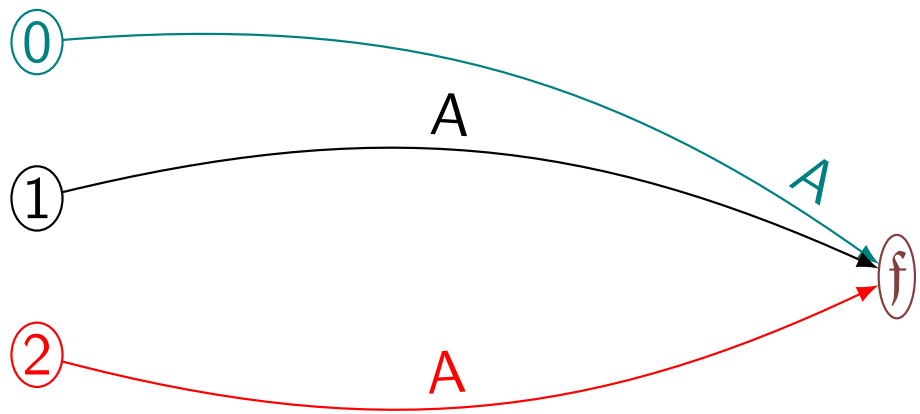
Generating automata for the left and right threads (6)

Procedure f: initially $g = 0$ Procedure h:



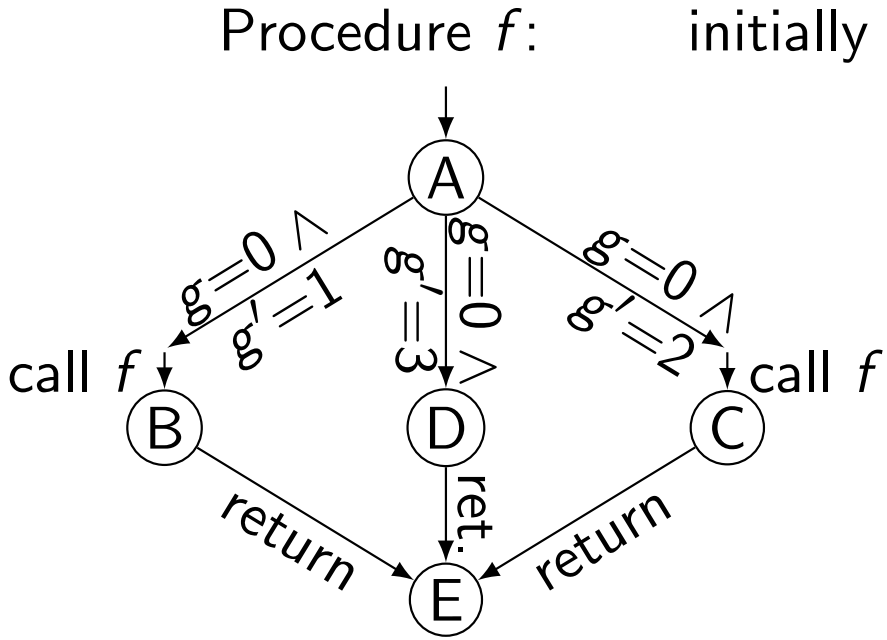


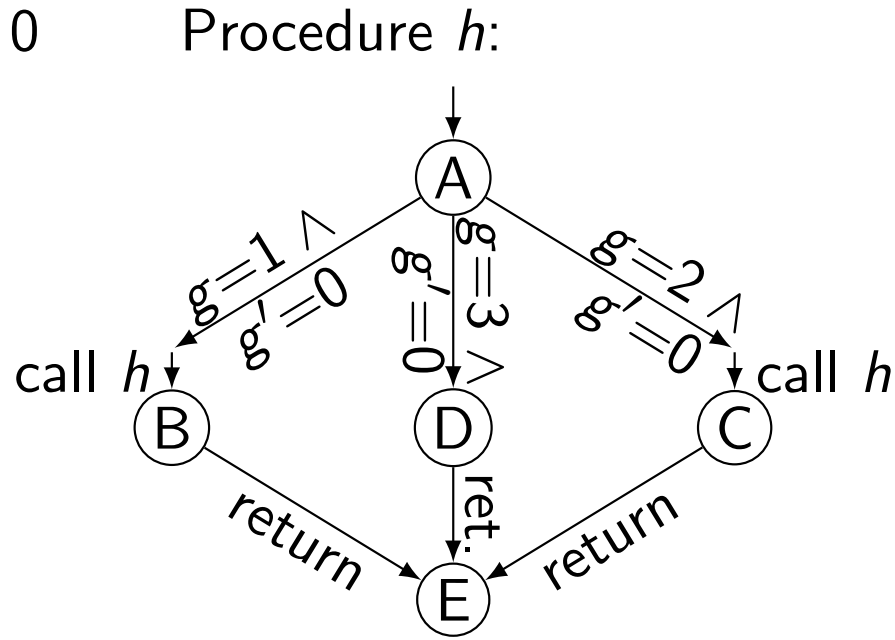


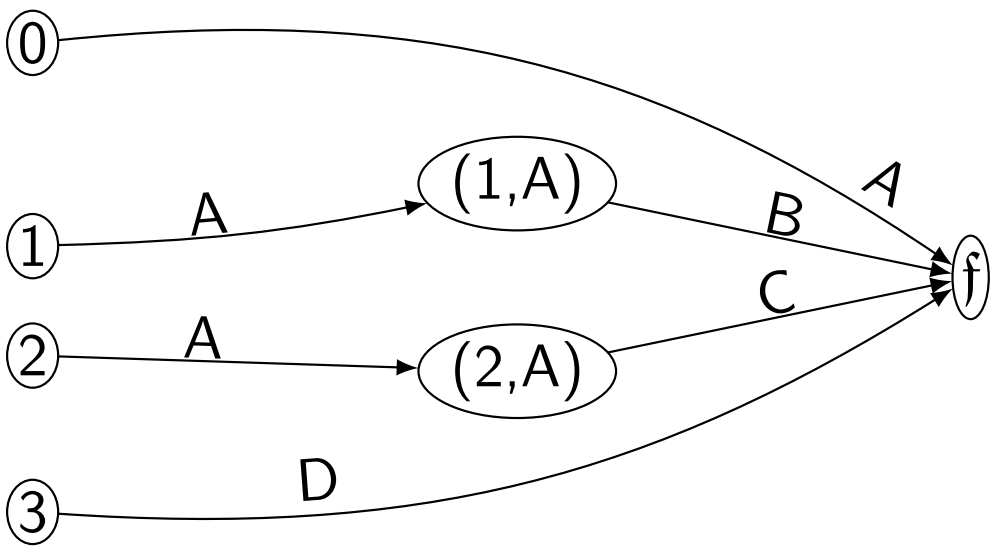


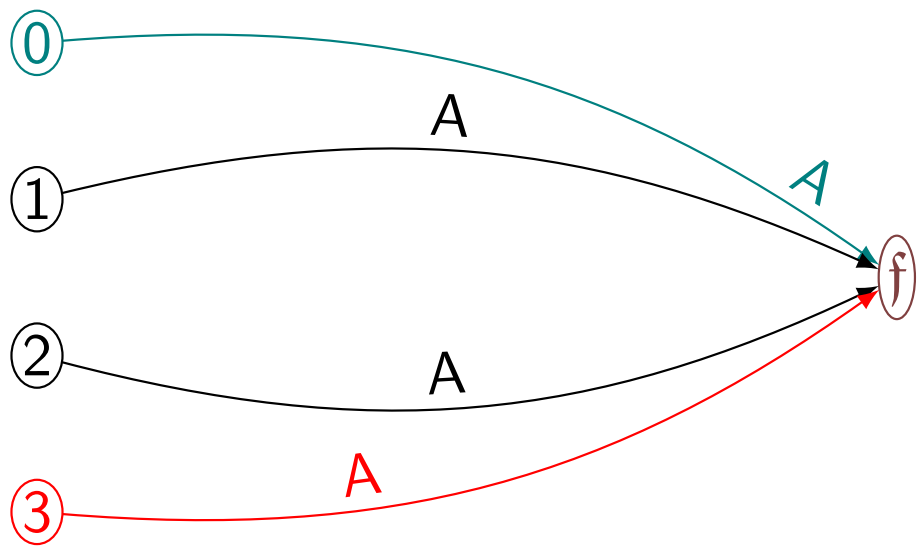
$G_0 = \{(0,1),(0,2),(0,3)\}$

Generating automata for the left and right threads (7)



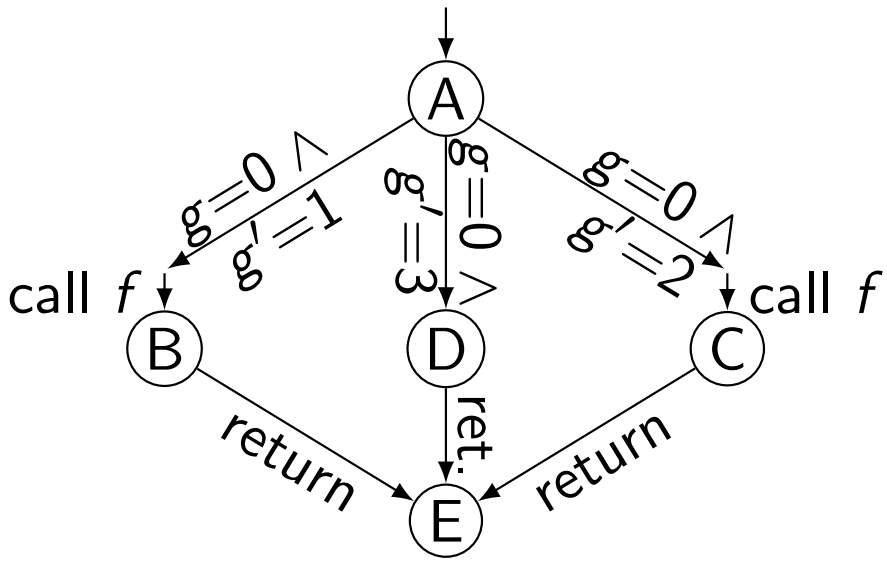


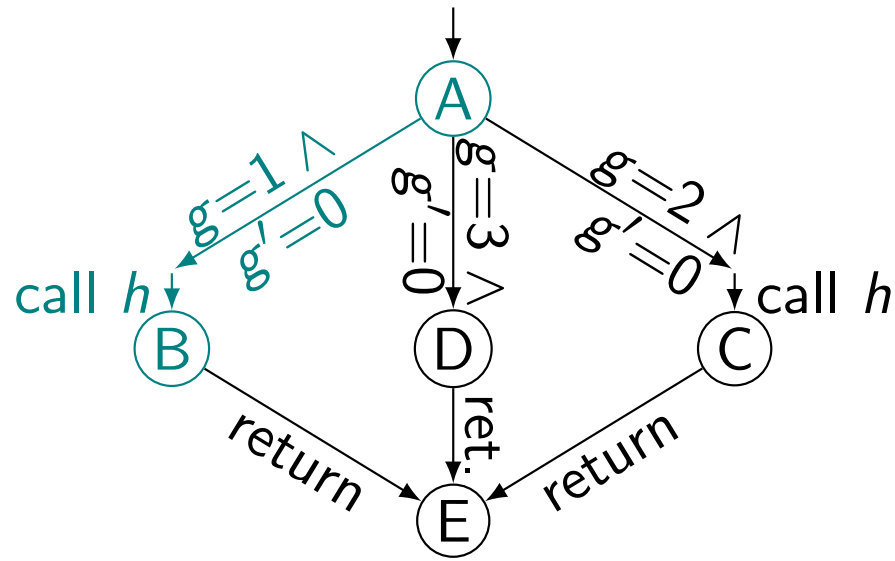


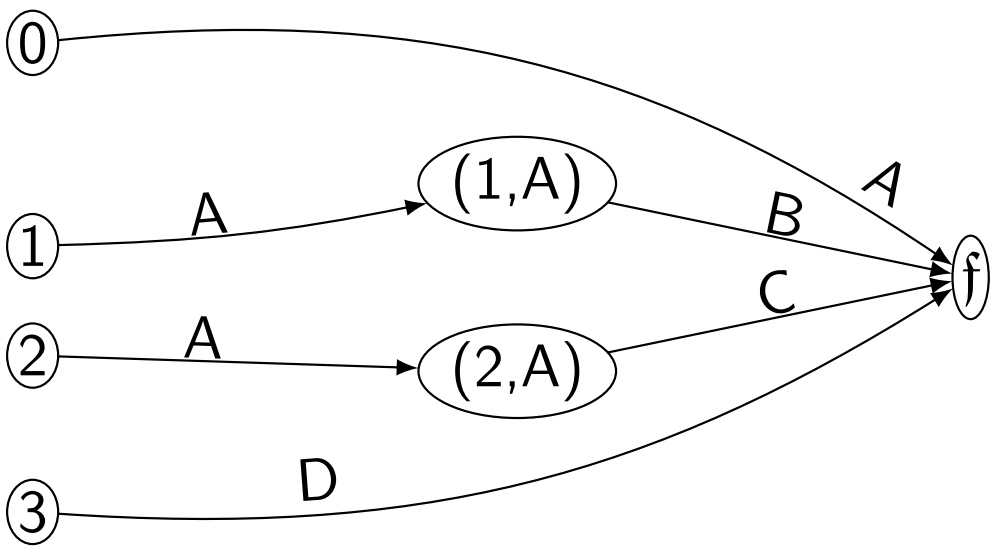


$G_0 = \{(0,1),(0,2),(0,3)\}$

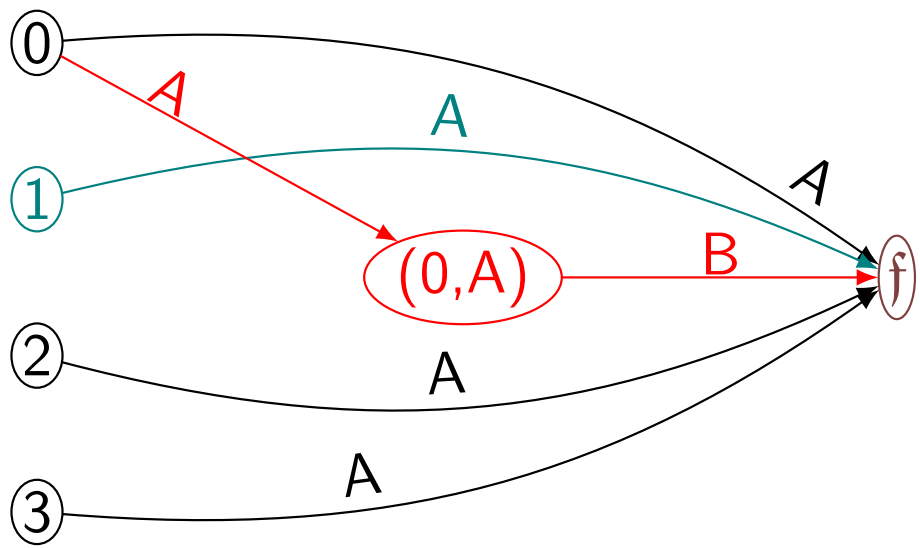
Generating automata for the left and right threads (8)







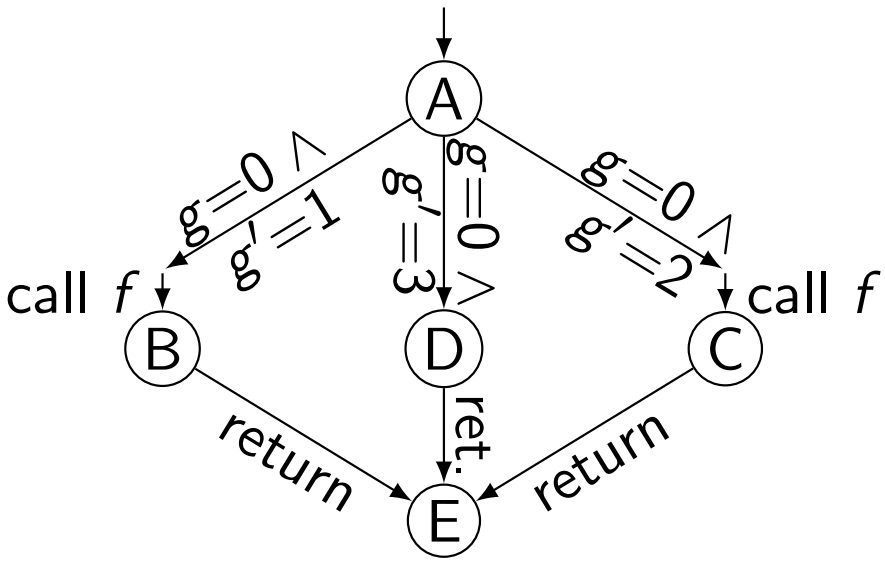
$G_0 = \{(0,1),(0,2),(0,3)\}$

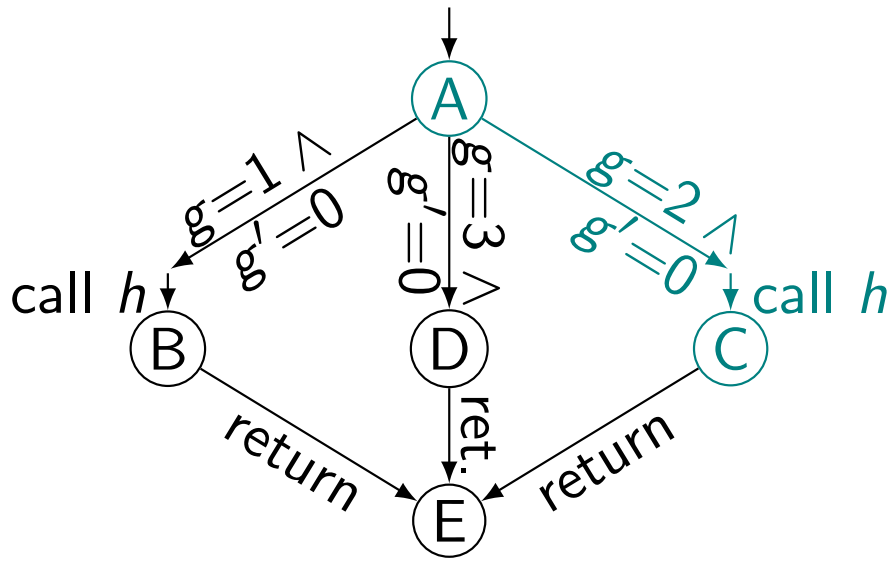


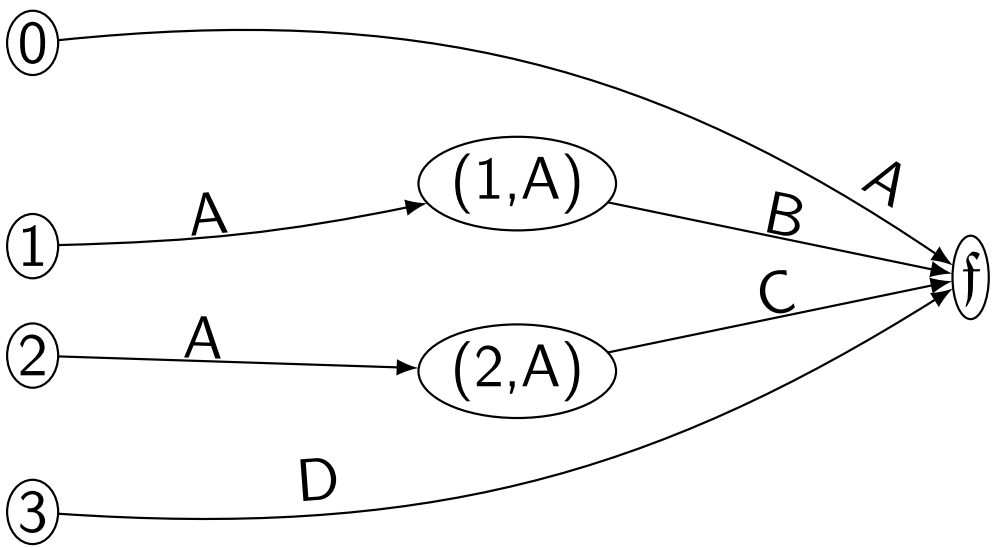
$G_1 = \{(1,0)\}$

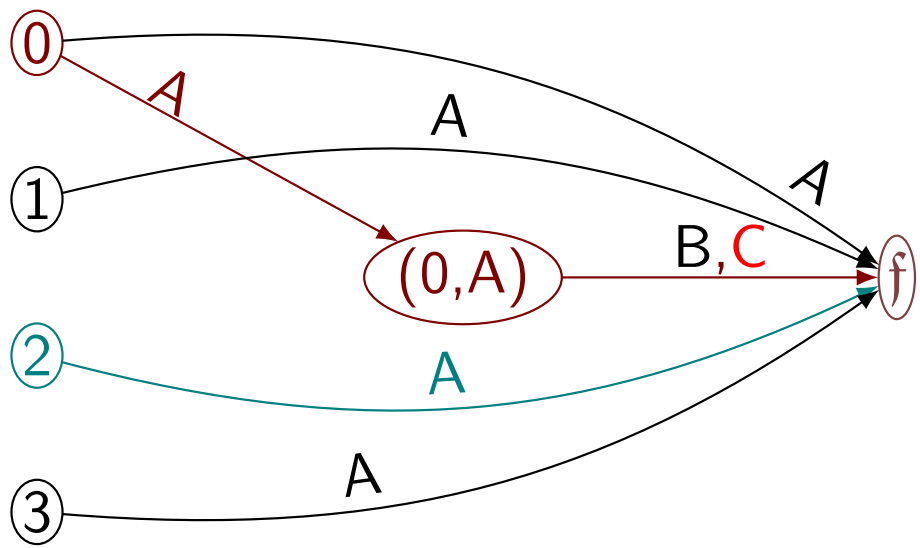
Generating automata for the left and right threads (9)

Procedure f: initially $g = 0$ Procedure h:





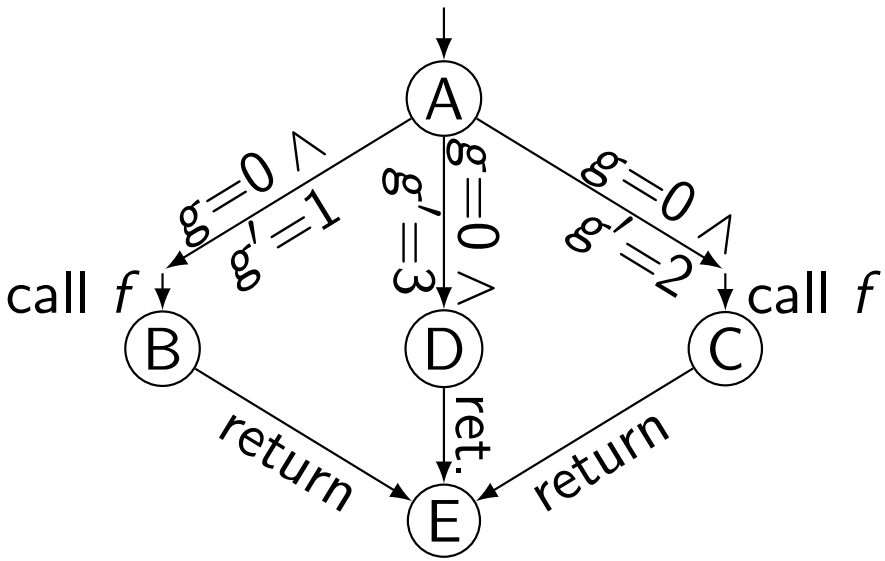


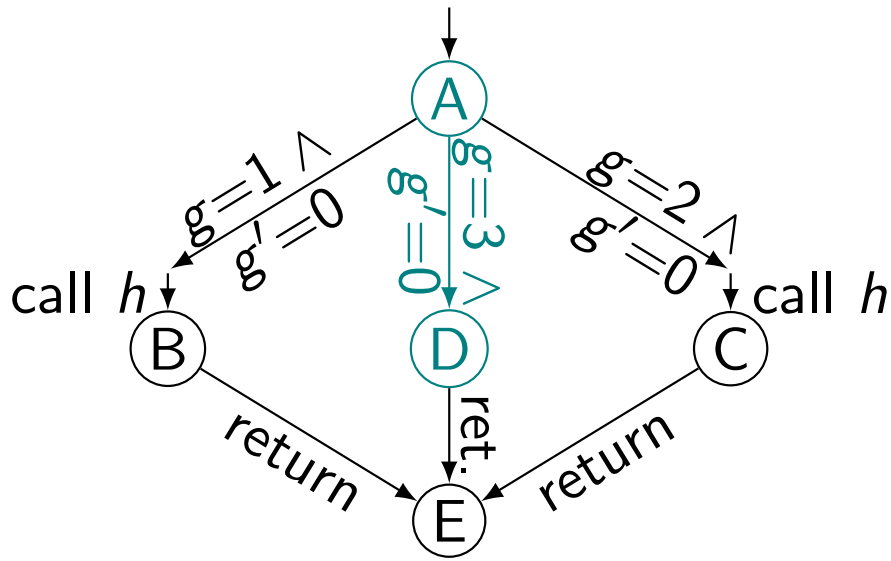


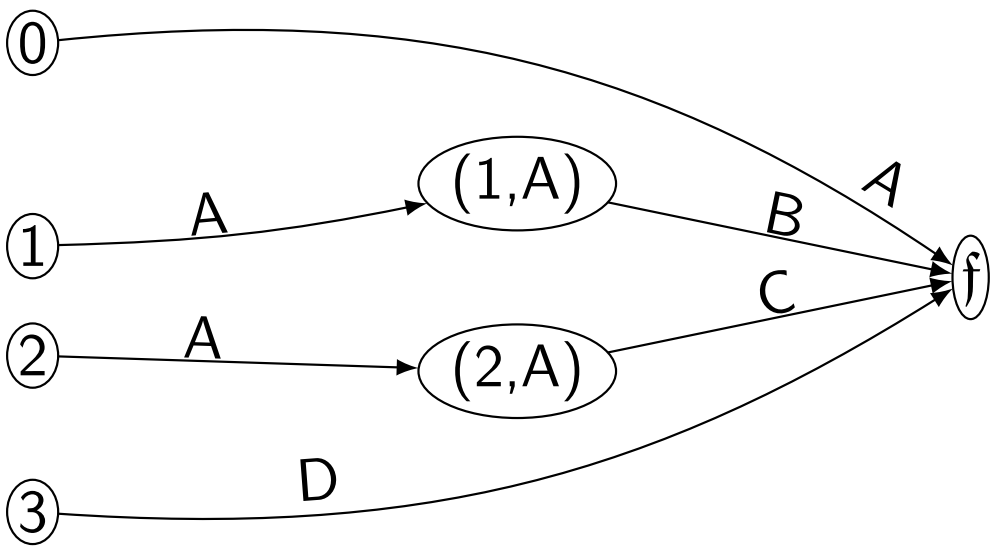
$G_0 = \{(0,1),(0,2),(0,3)\}$ $G_1 = \{(1,0),(2,0)\}$

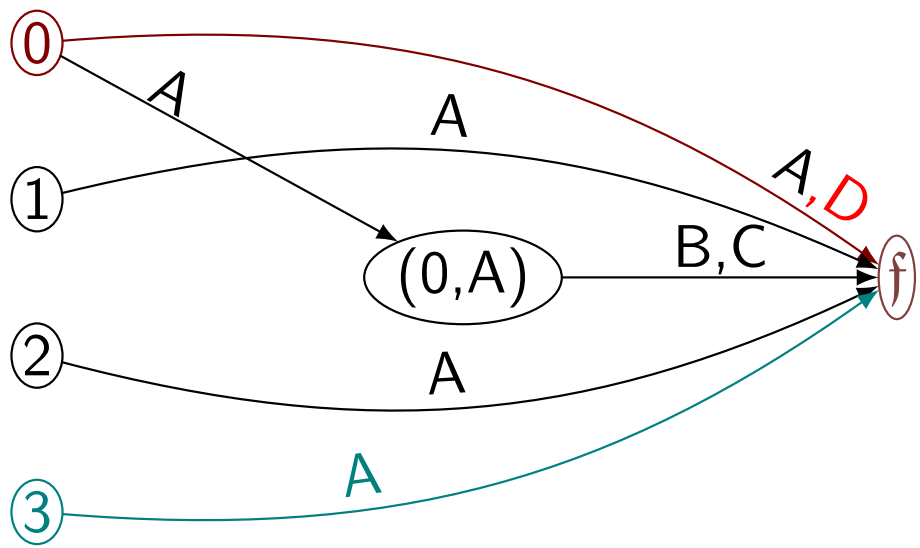
Generating automata for the left and right threads (10)

Procedure f: initially $g = 0$ Procedure h:





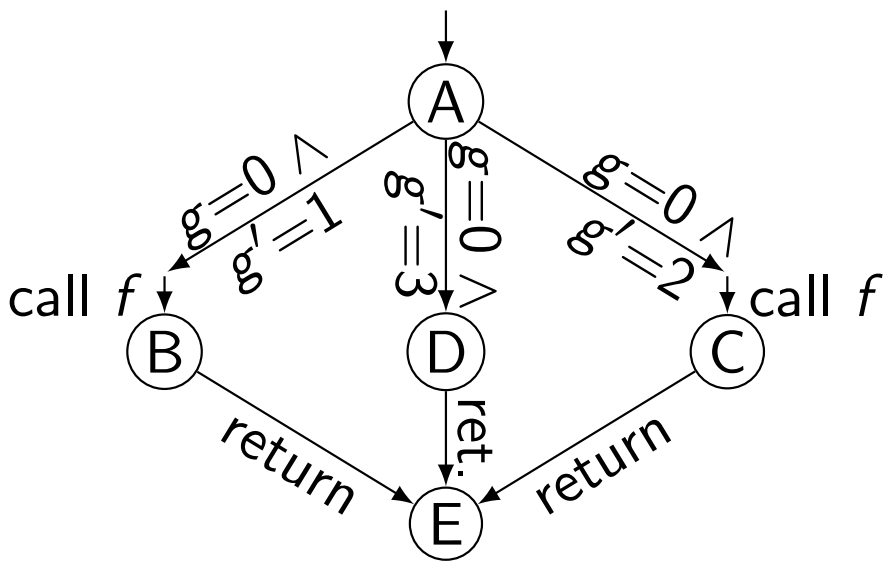


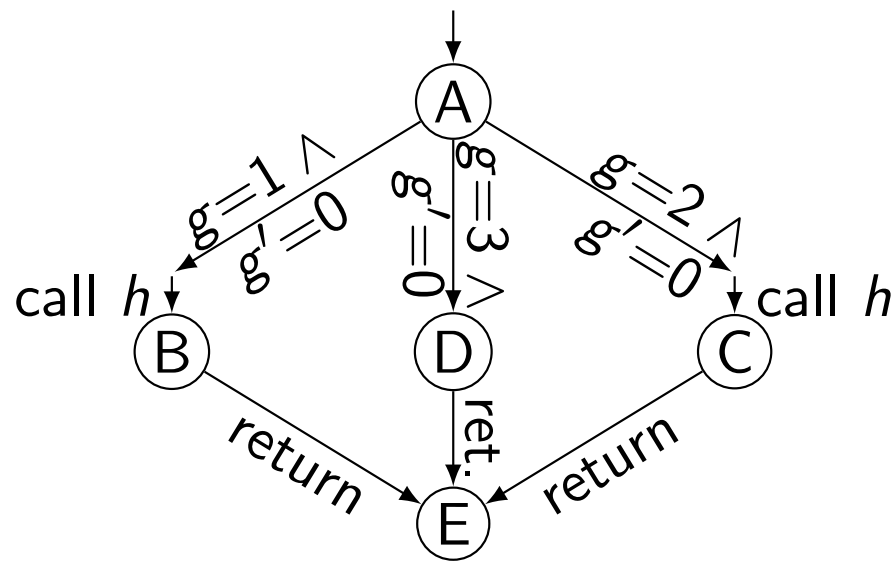


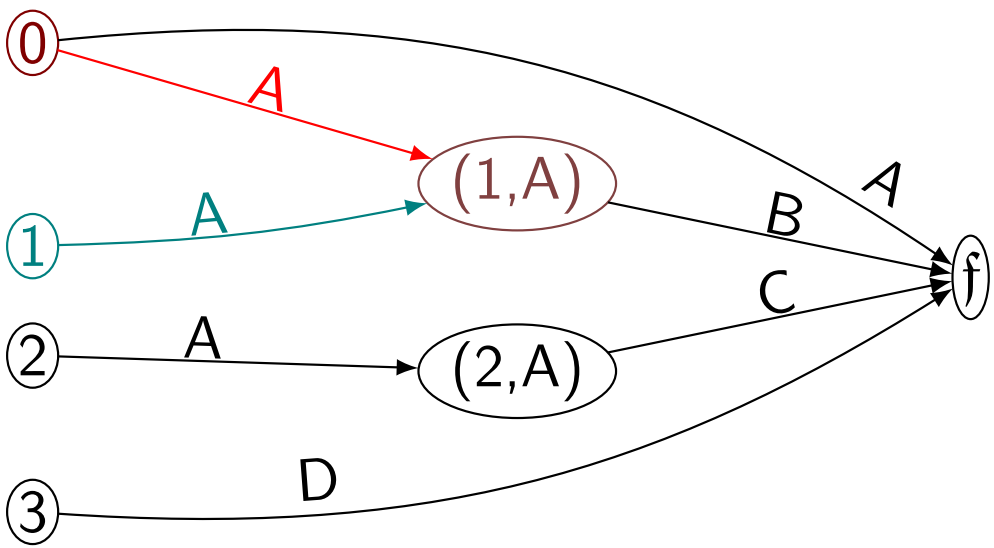
$G_0 = \{(0,1),(0,2),(0,3)\}$

$G_1 = \{(1,0),(2,0),(3,0)\}$

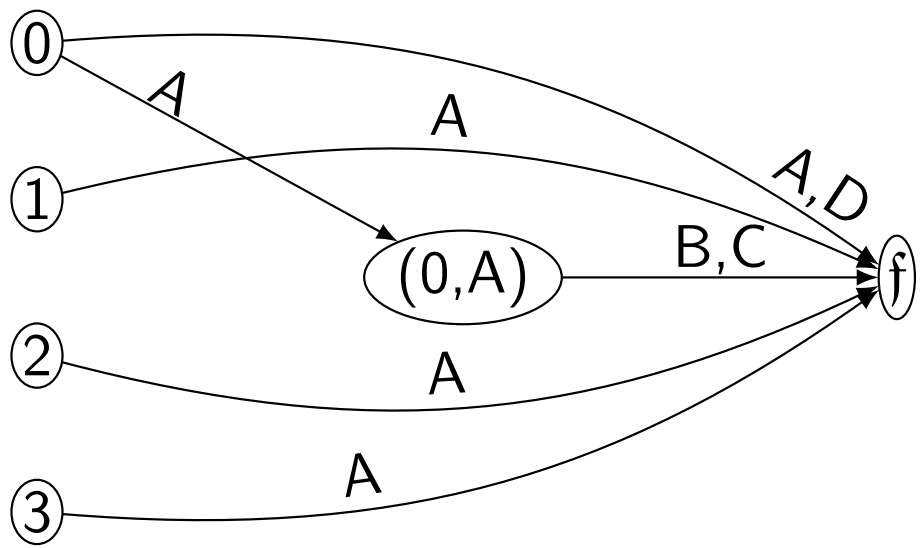
Generating automata for the left and right threads (11)





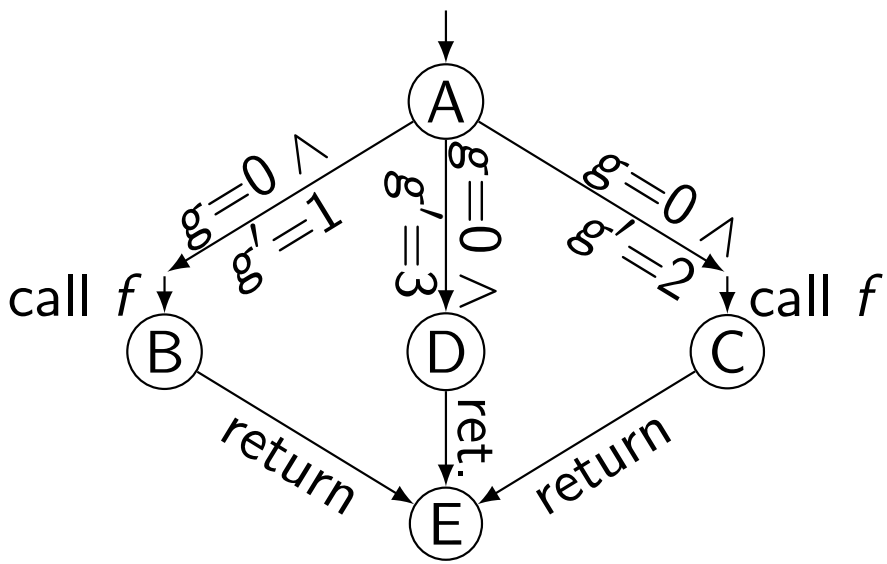


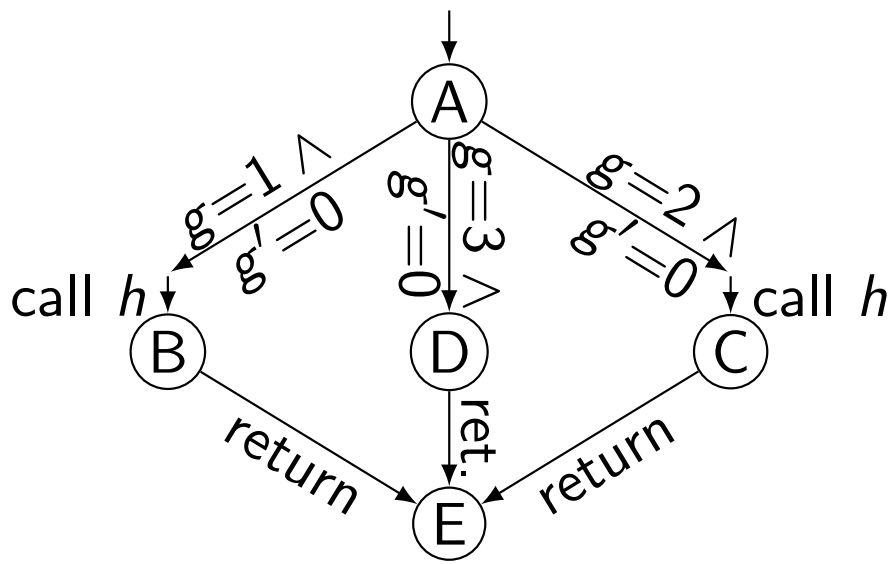
$G_0 = \{(0,1),(0,2),(0,3)\}$

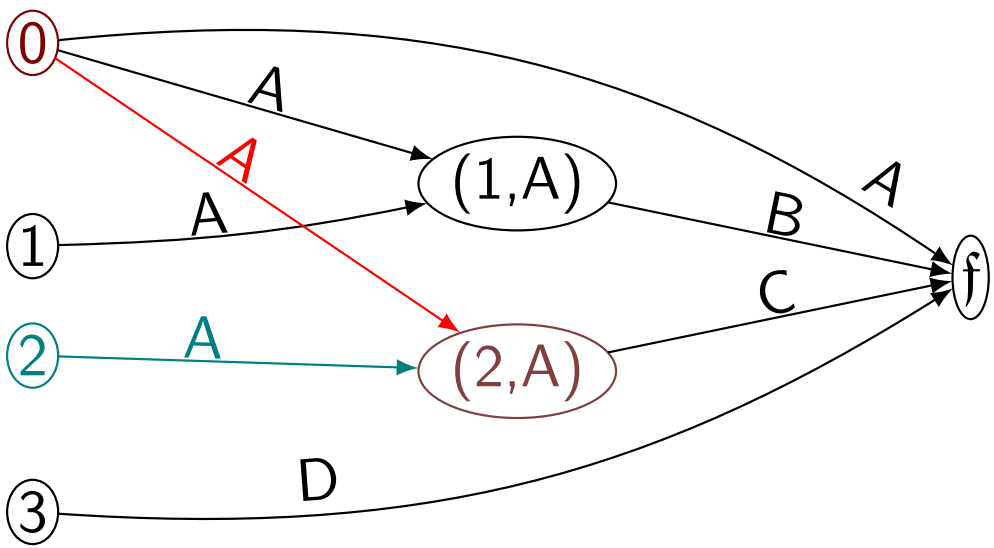


$G_1 = \{(1,0),(2,0),(3,0)\}$

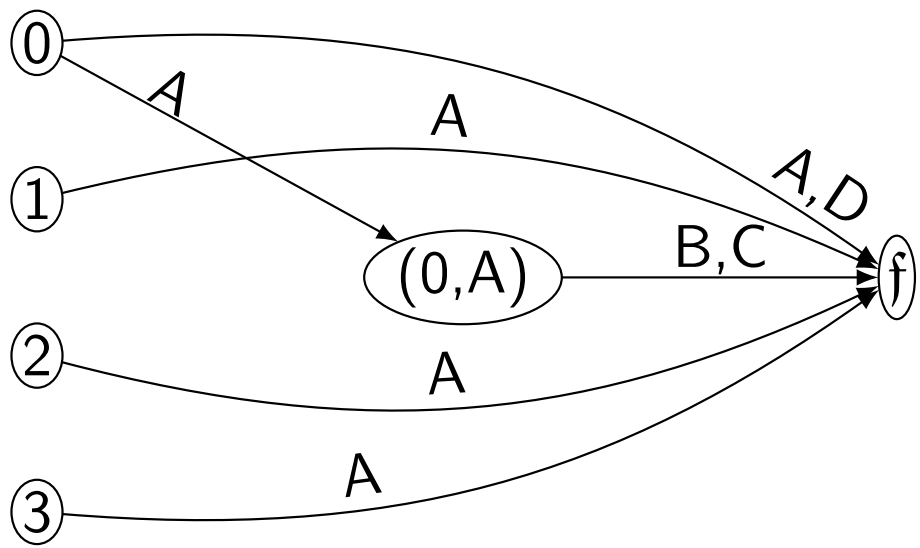
Generating automata for the left and right threads (12)





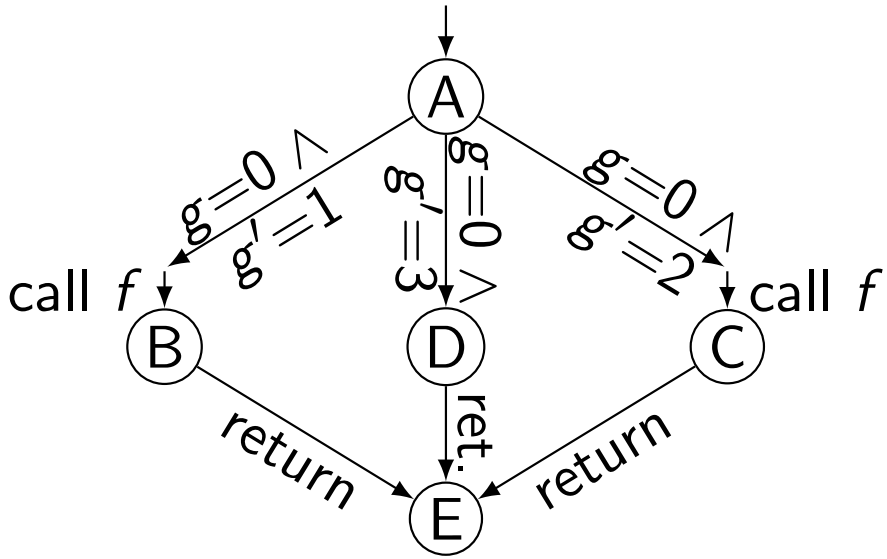


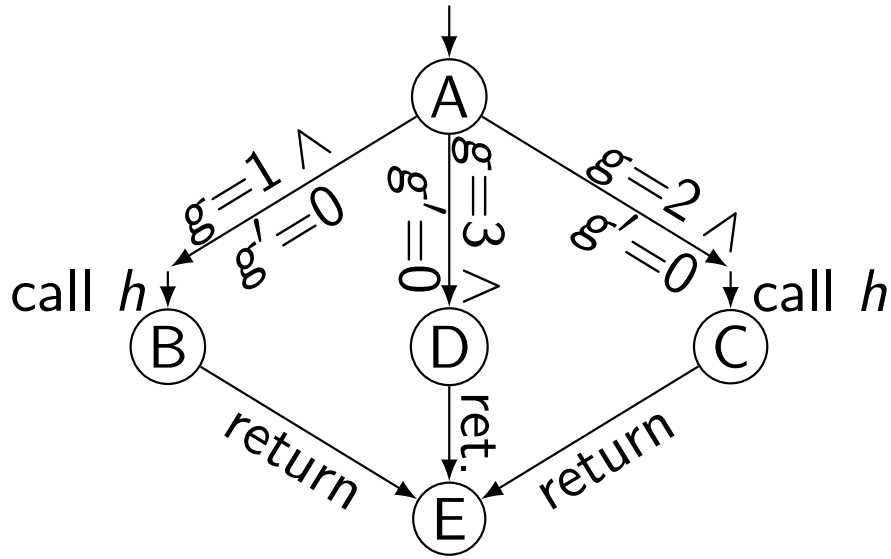
$G_0 = \{(0,1),(0,2),(0,3)\}$

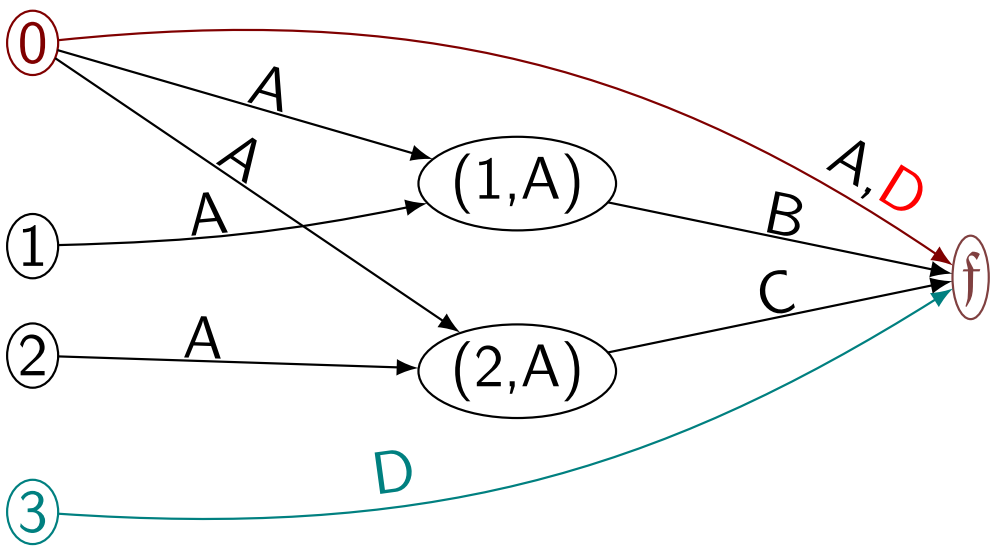


$G_1 = \{(1,0),(2,0),(3,0)\}$

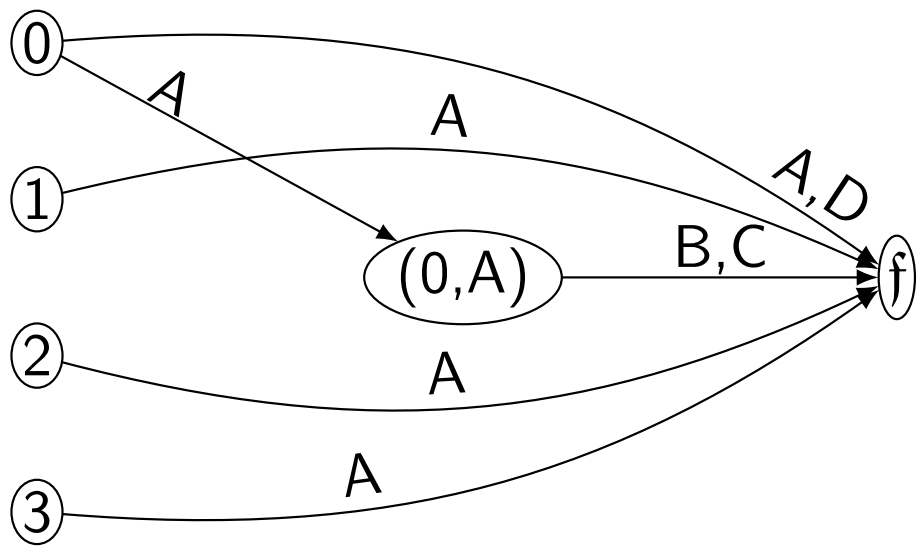
Generating automata for the left and right threads (13)





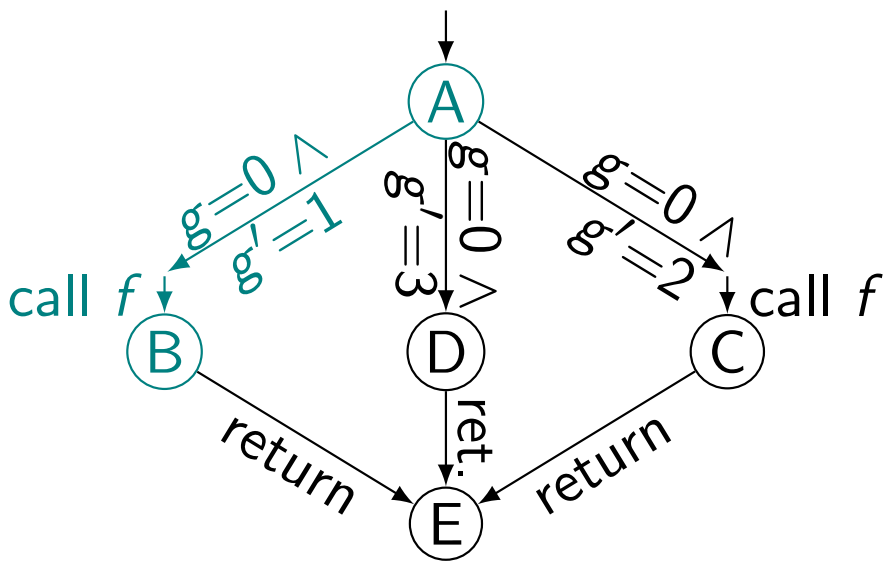


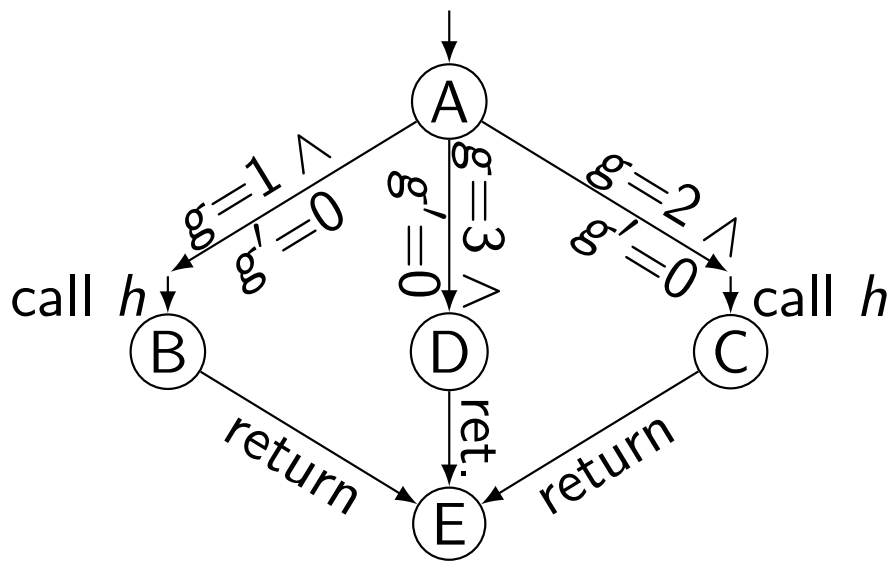
$G_0 = \{(0,1),(0,2),(0,3)\}$

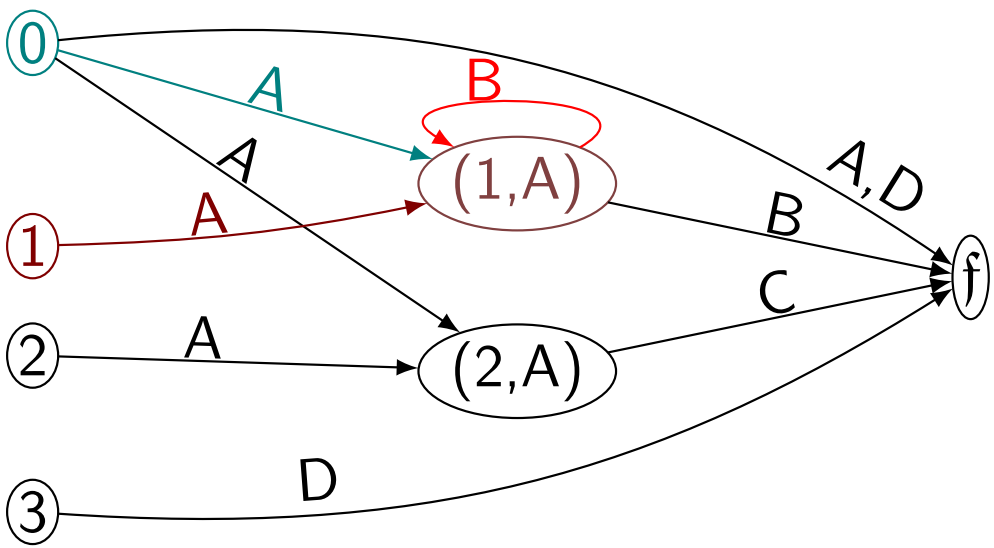


$G_1 = \{(1,0),(2,0),(3,0)\}$

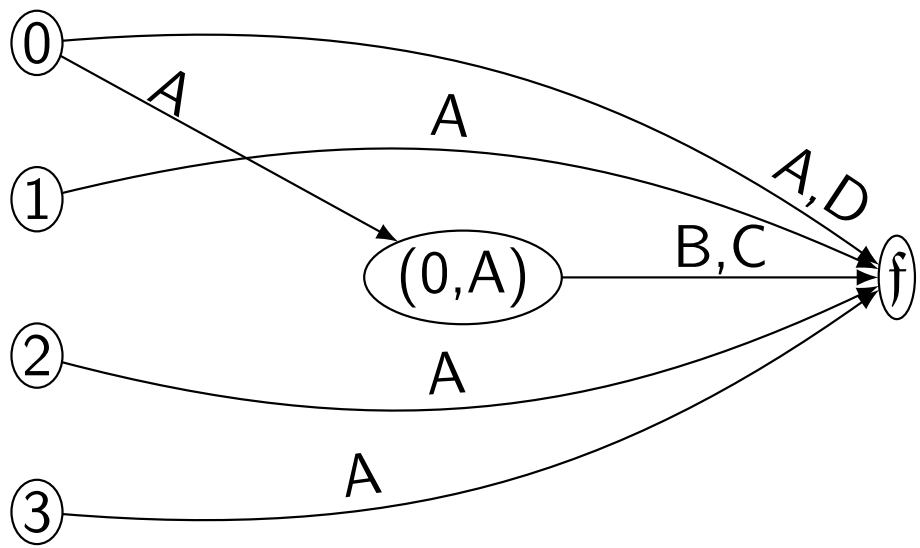
Generating automata for the left and right threads (14)







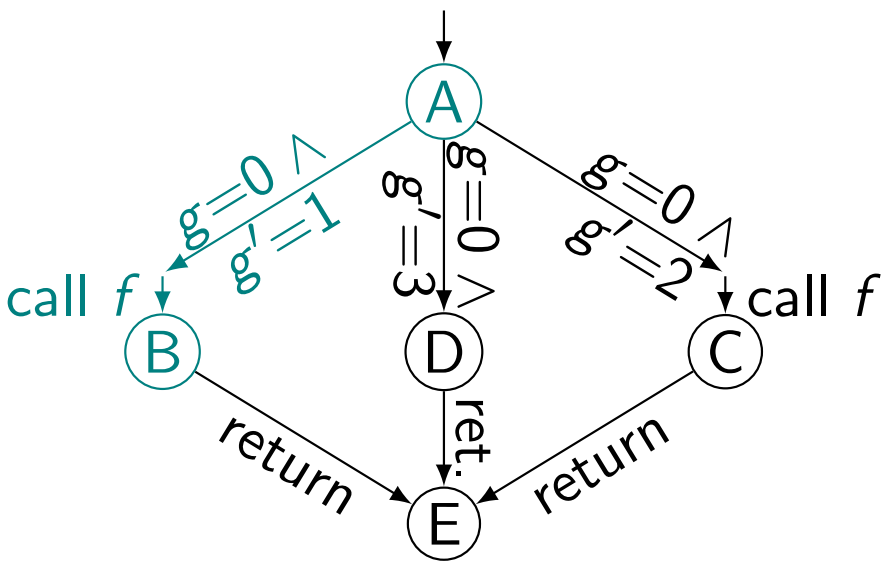
$G_0 = \{(0,1),(0,2),(0,3)\}$

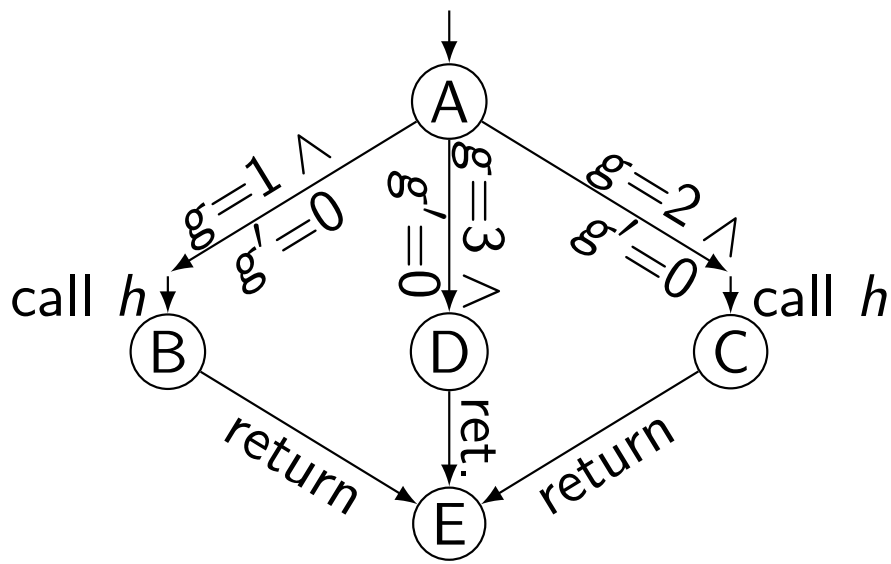


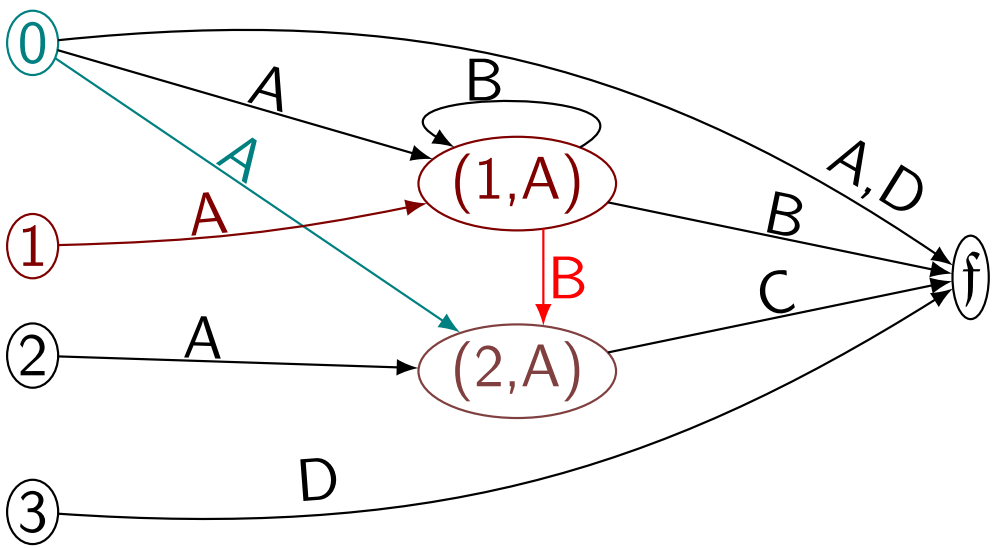
$G_1 = \{(1,0),(2,0),(3,0)\}$

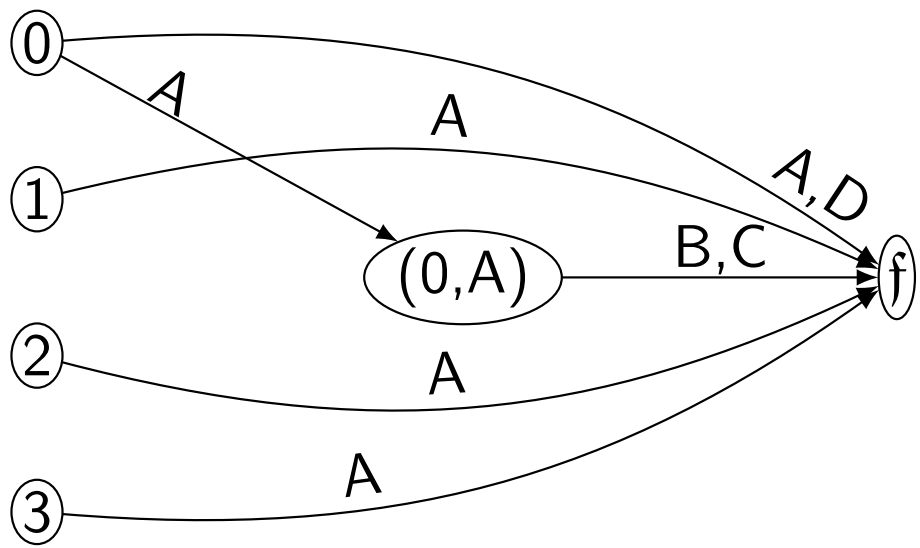
Generating automata for the left and right threads (15)

Procedure f: initially $g = 0$ Procedure h:



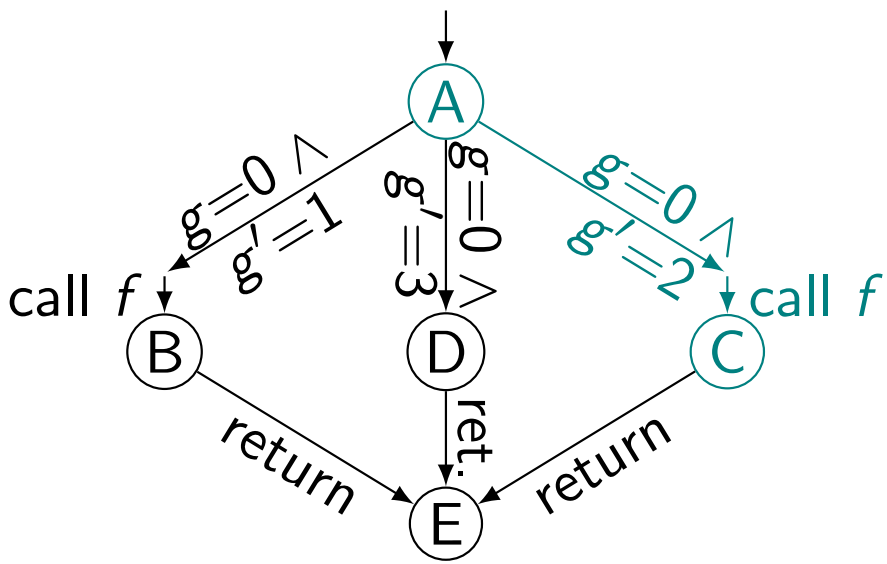


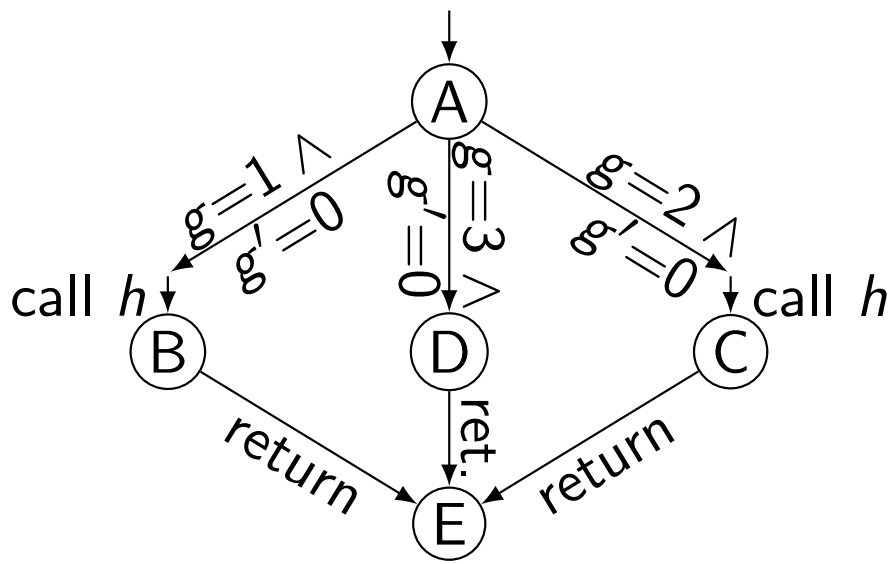


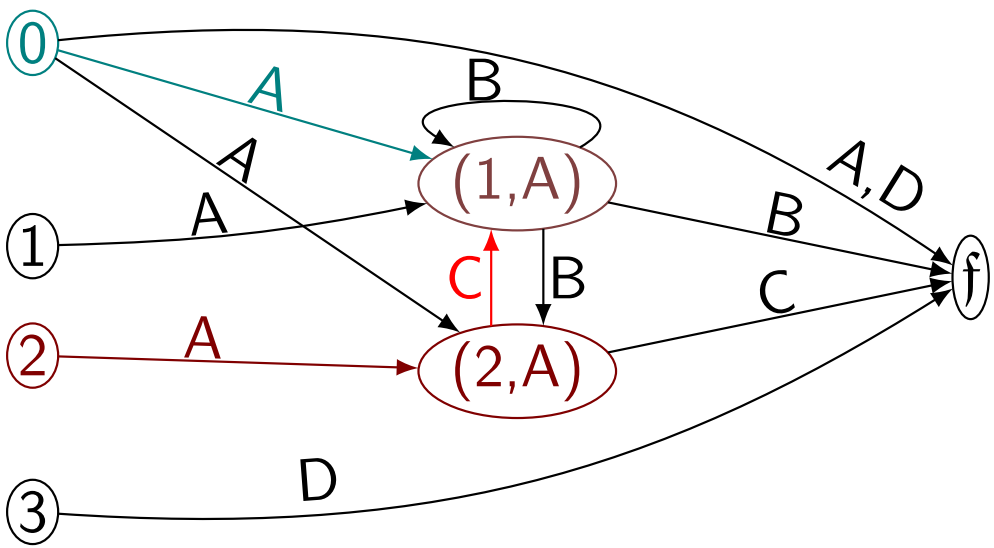


$G_0 = \{(0,1),(0,2),(0,3)\}$ $G_1 = \{(1,0),(2,0),(3,0)\}$

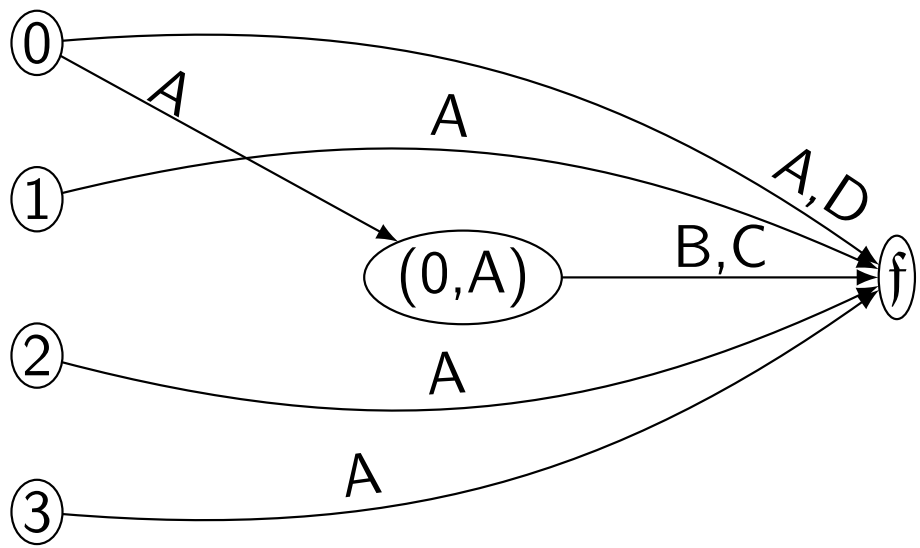
Generating automata for the left and right threads (16)





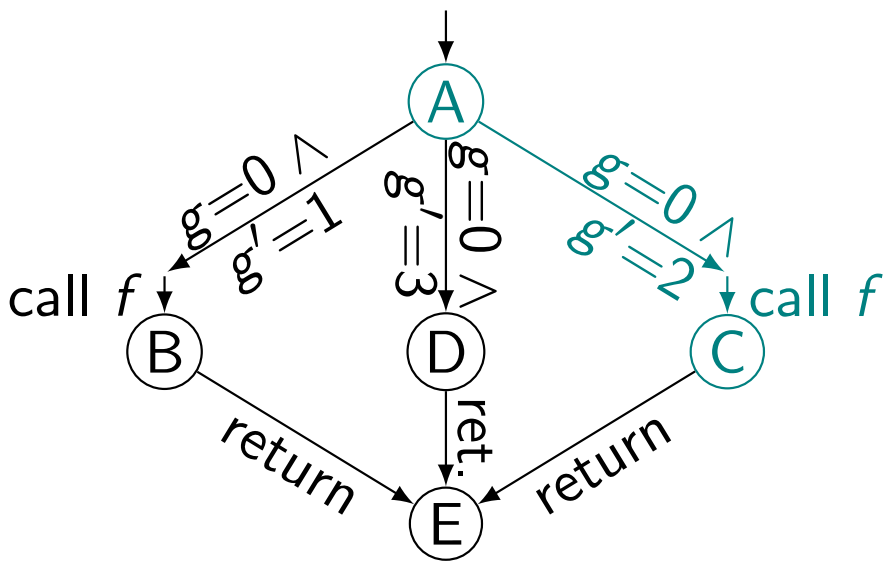


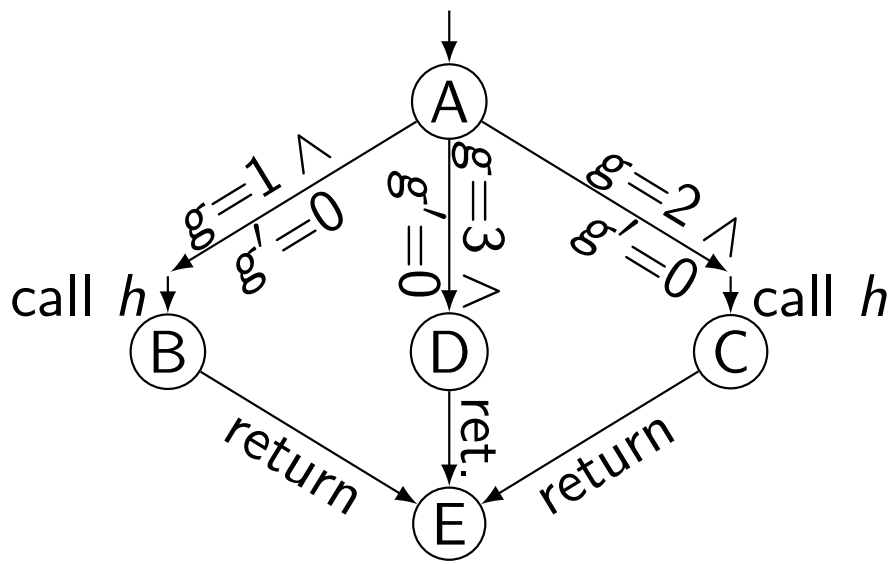
$G_0 = \{(0,1),(0,2),(0,3)\}$

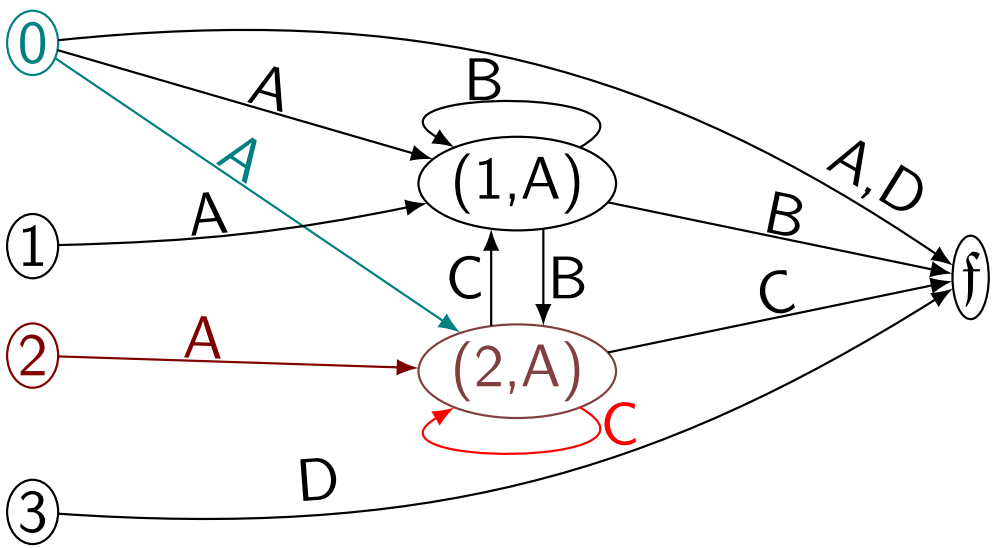


$G_1 = \{(1,0),(2,0),(3,0)\}$

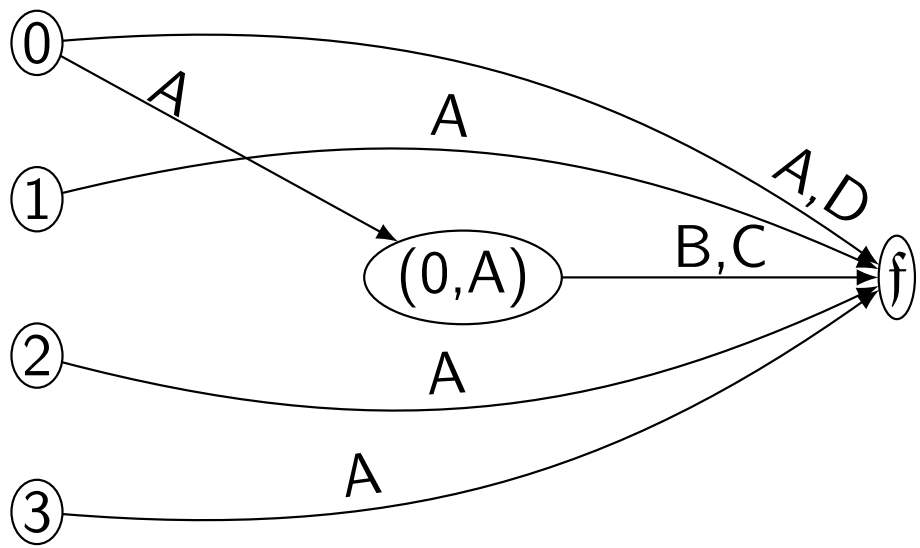
Generating automata for the left and right threads (17)







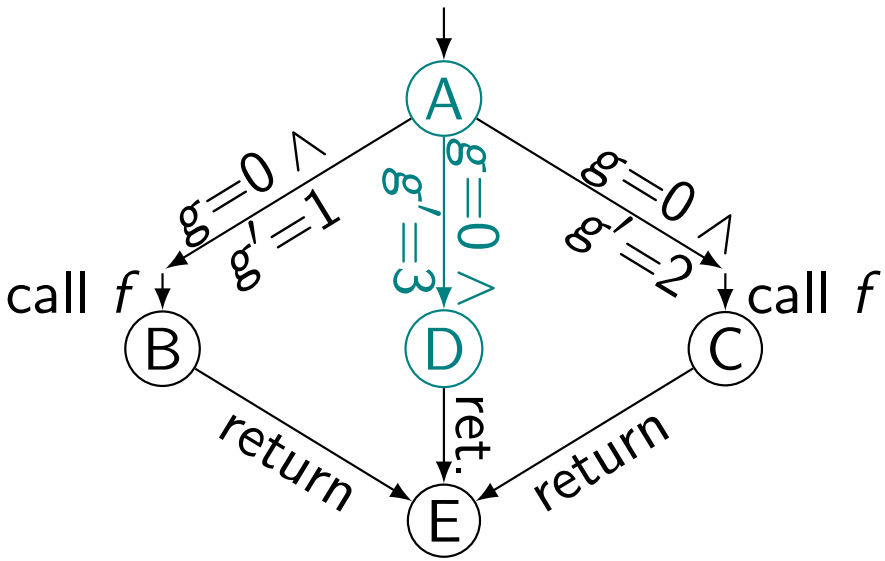
$G_0 = \{(0,1),(0,2),(0,3)\}$

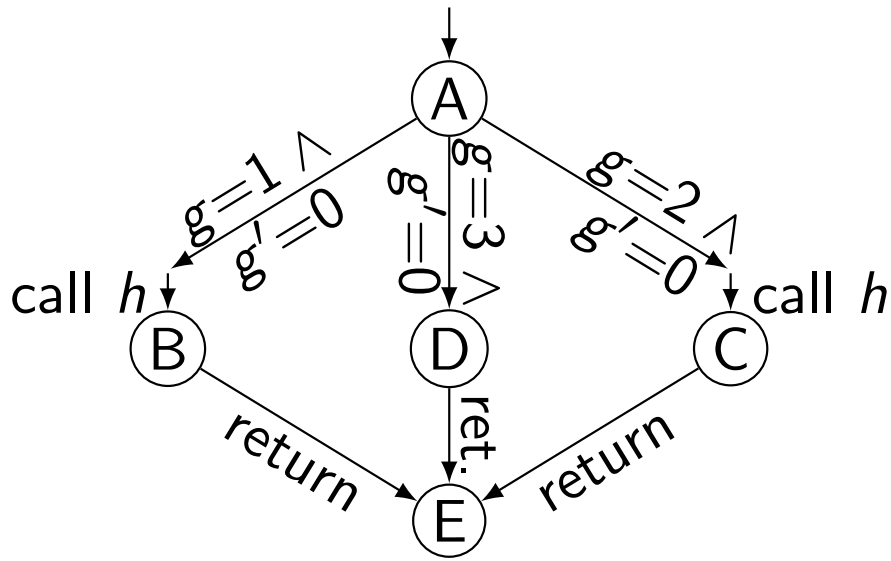


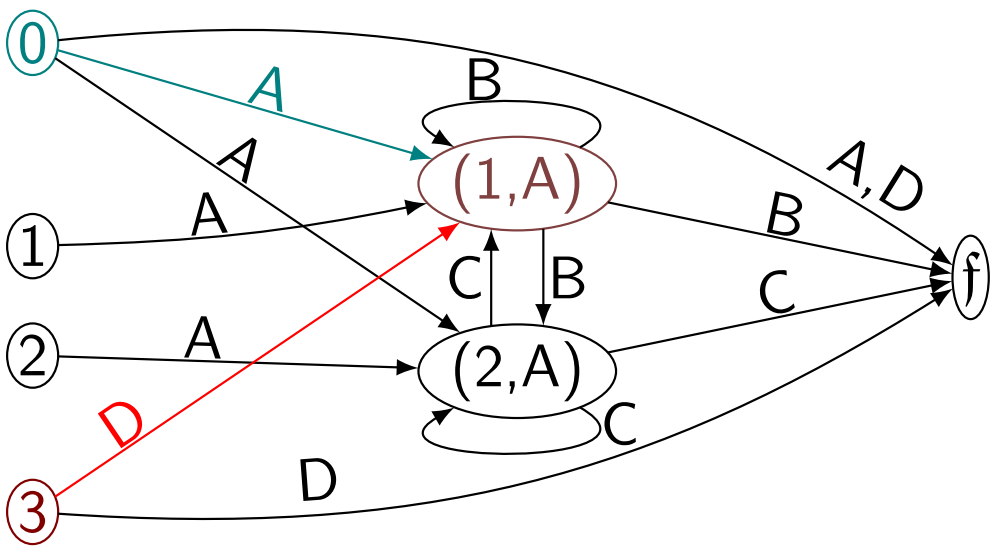
$G_1 = \{(1,0),(2,0),(3,0)\}$

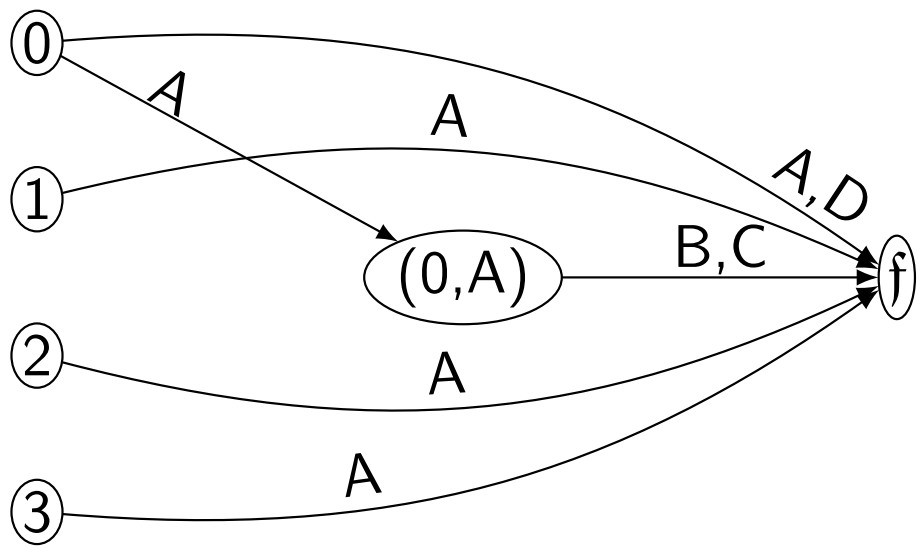
Generating automata for the left and right threads (18)

Procedure f: initially $g = 0$ Procedure h:



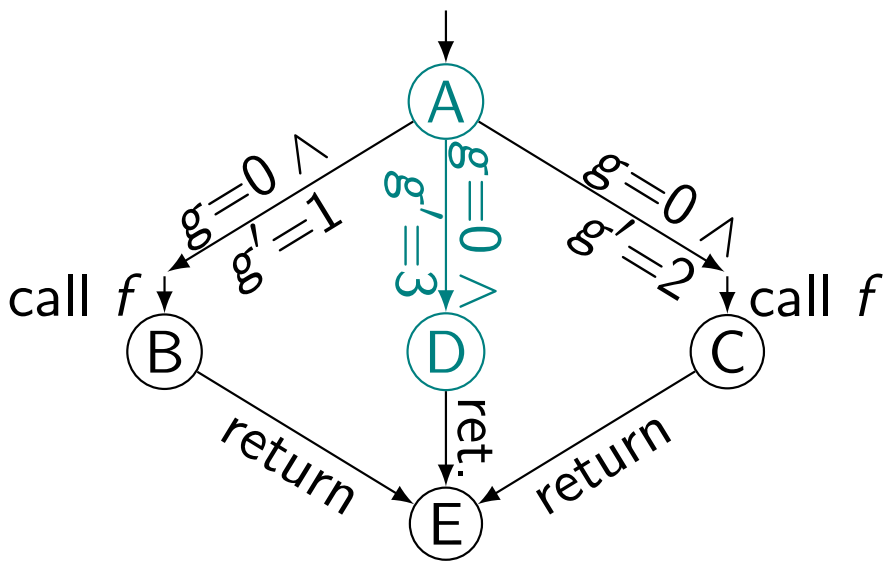


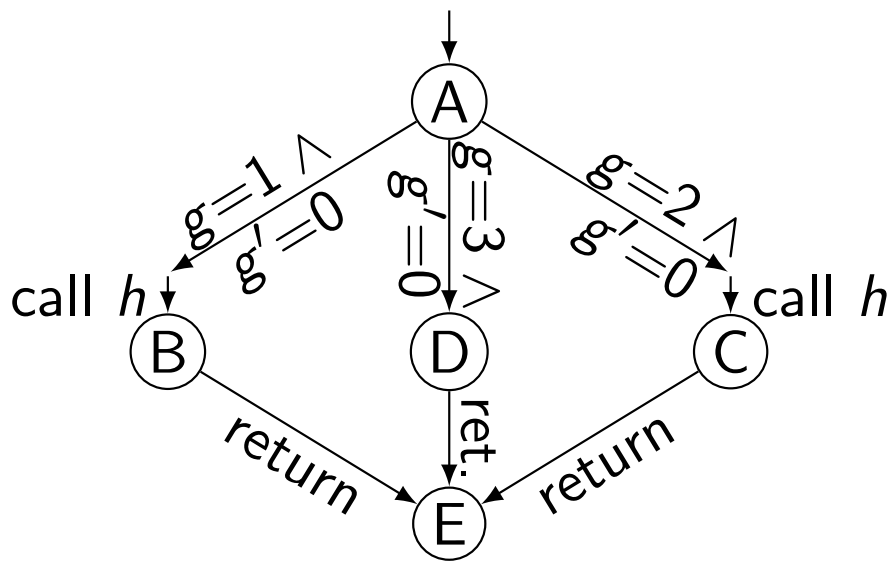


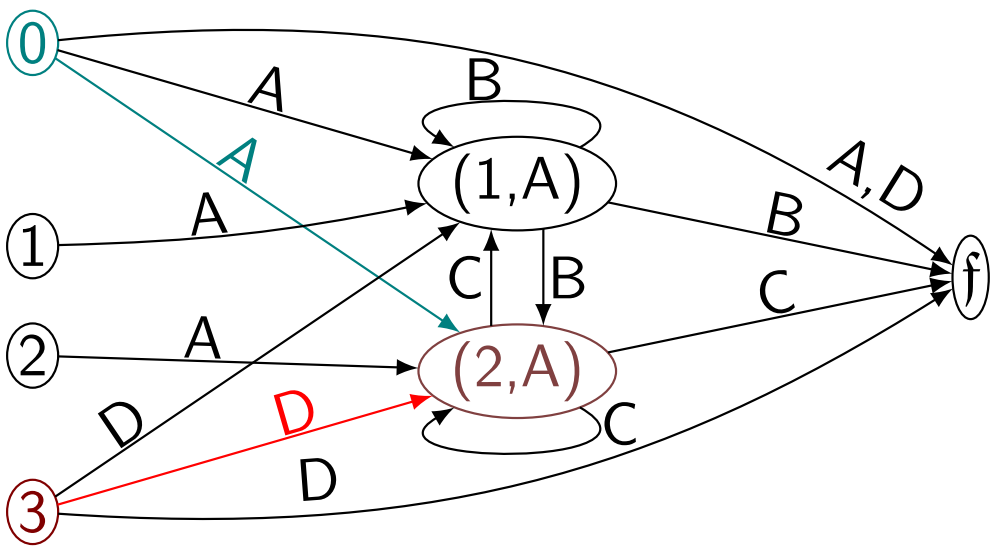


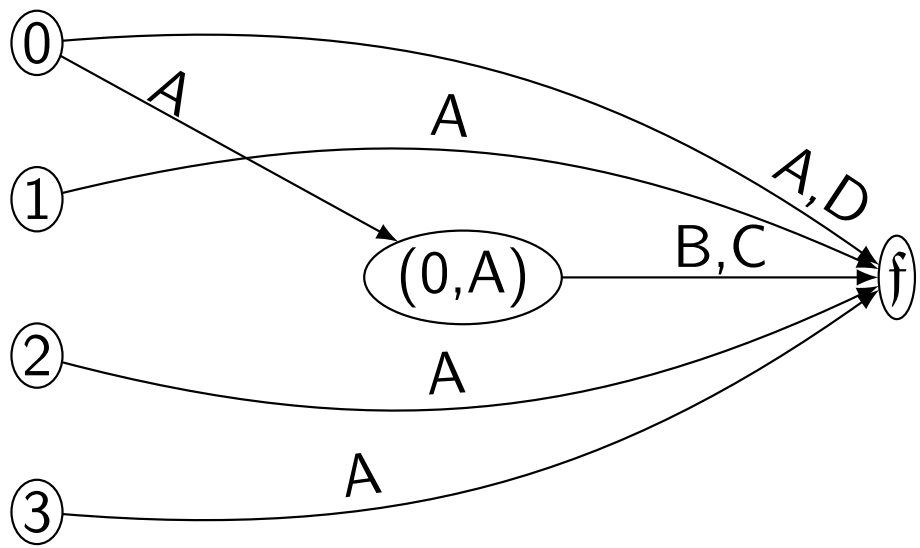
$G_0 = \{(0,1),(0,2),(0,3)\}$ $G_1 = \{(1,0),(2,0),(3,0)\}$

Generating automata for the left and right threads (19)





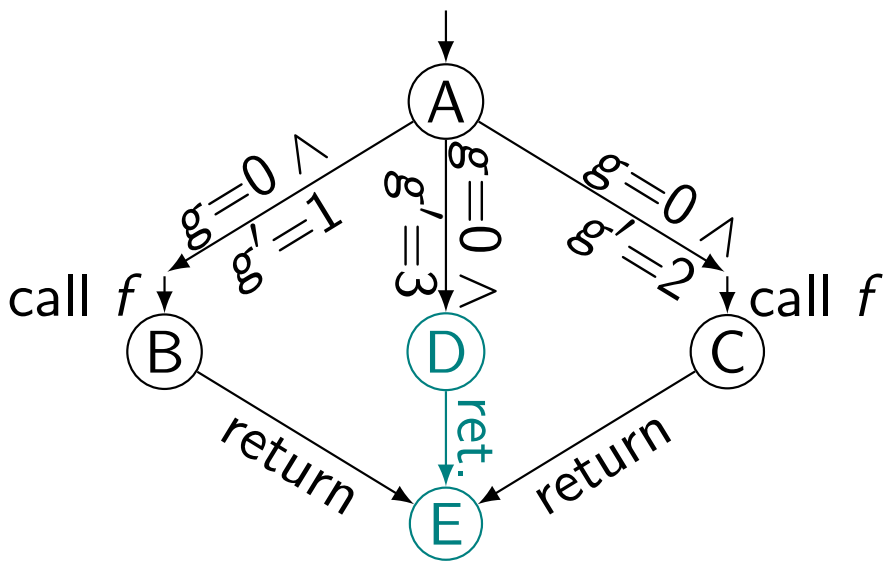


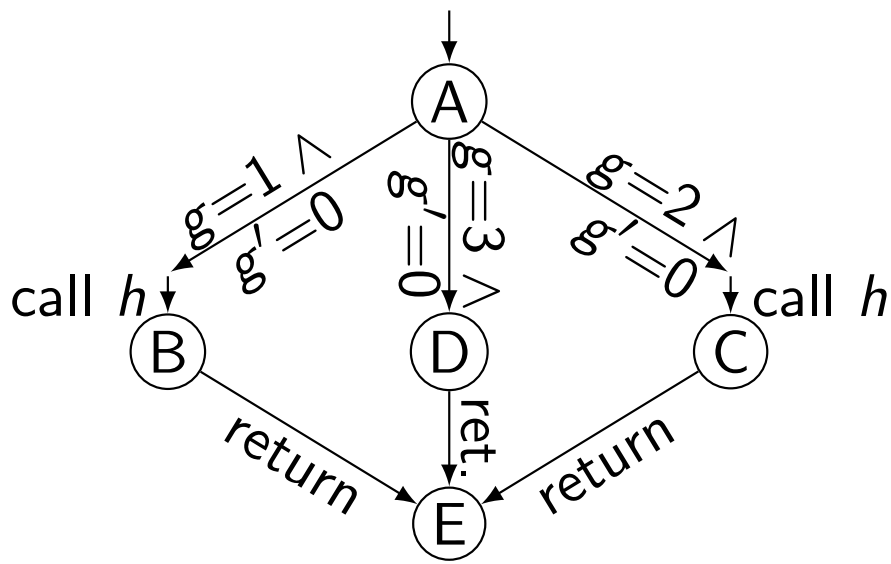


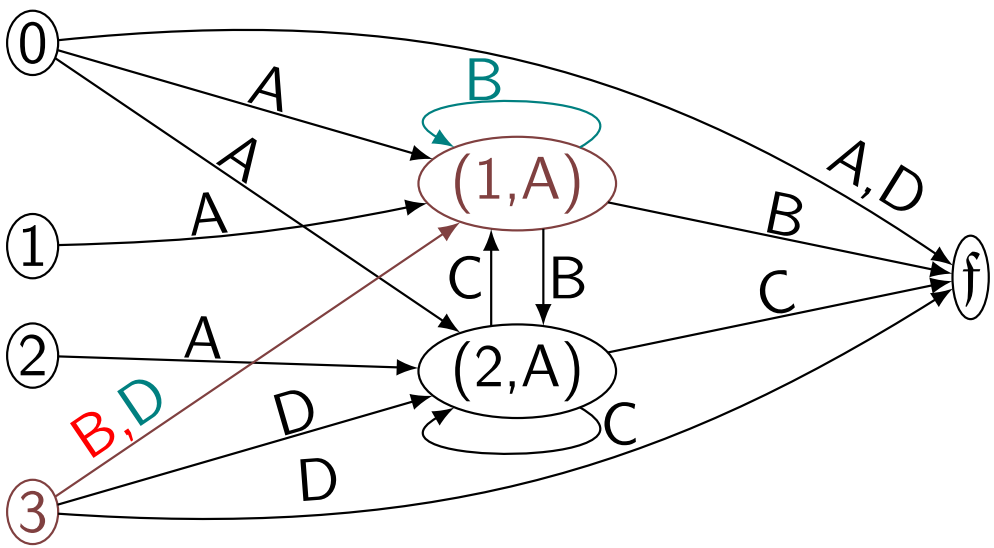
$G_0 = \{(0,1),(0,2),(0,3)\}$

$G_1 = \{(1,0),(2,0),(3,0)\}$

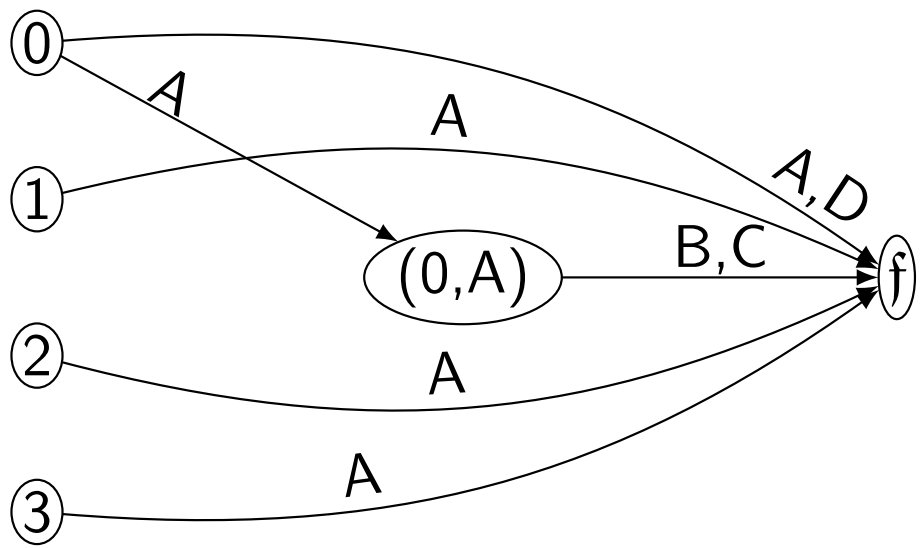
Generating automata for the left and right threads (20)





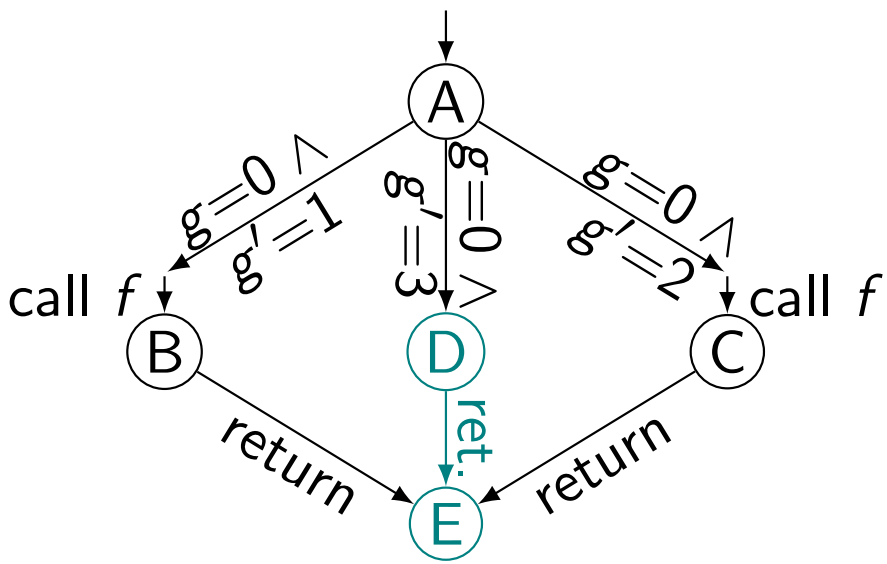


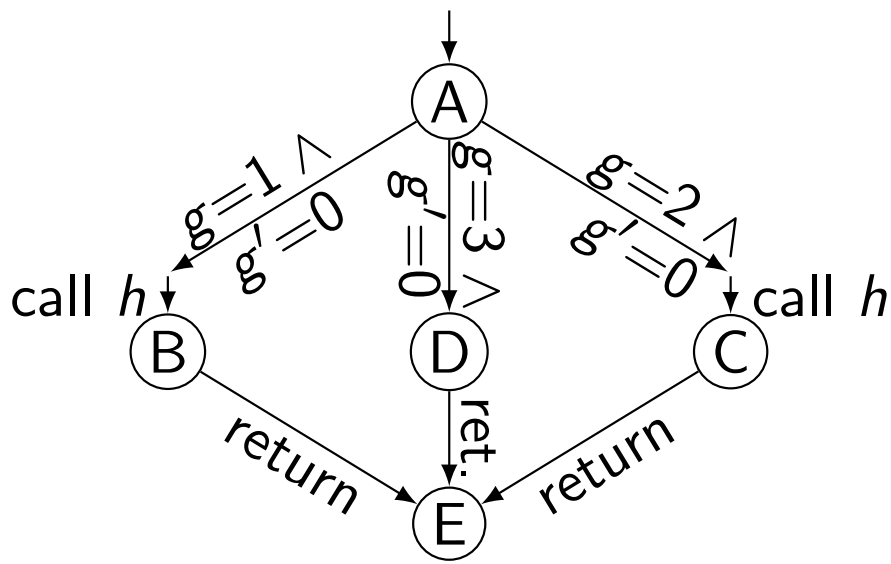
$G_0 = \{(0,1),(0,2),(0,3)\}$

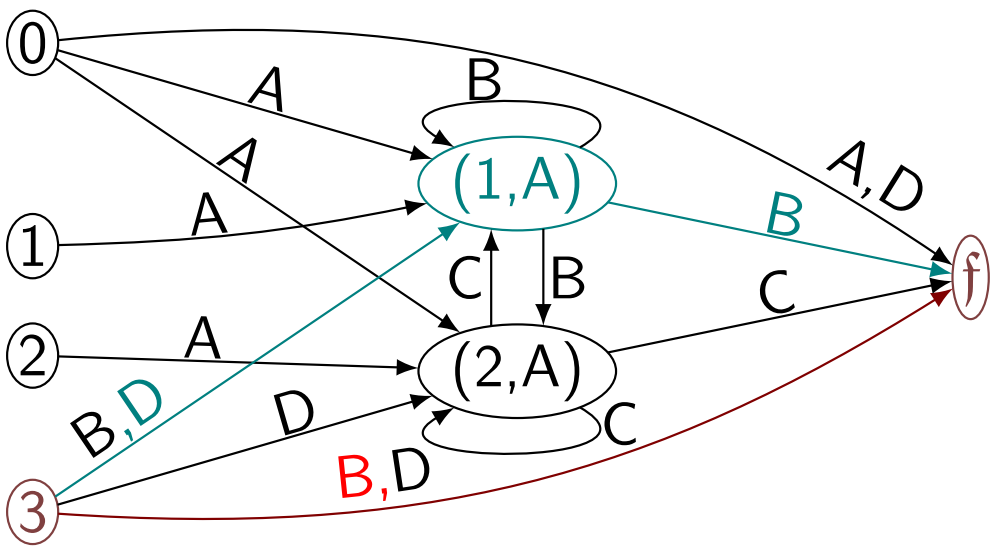


$G_1 = \{(1,0),(2,0),(3,0)\}$

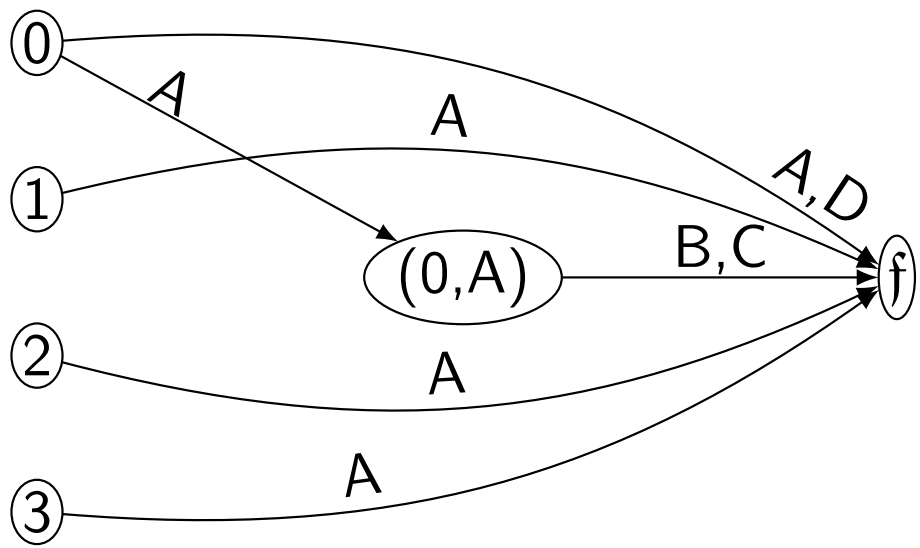
Generating automata for the left and right threads (21)







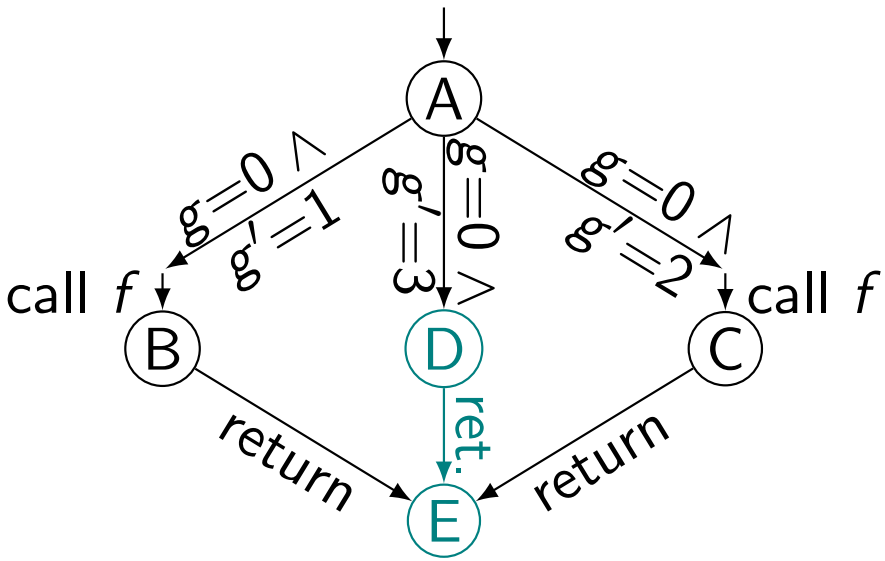
$G_0 = \{(0,1),(0,2),(0,3)\}$

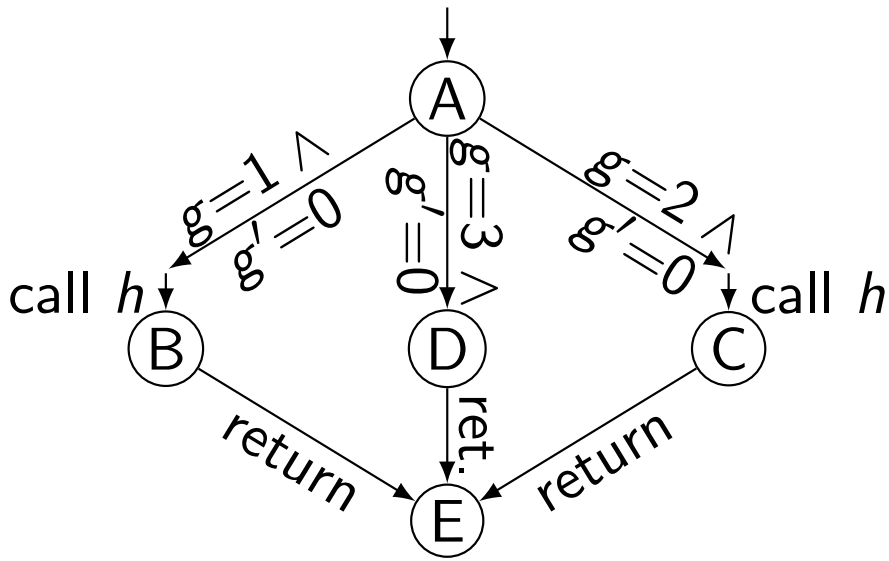


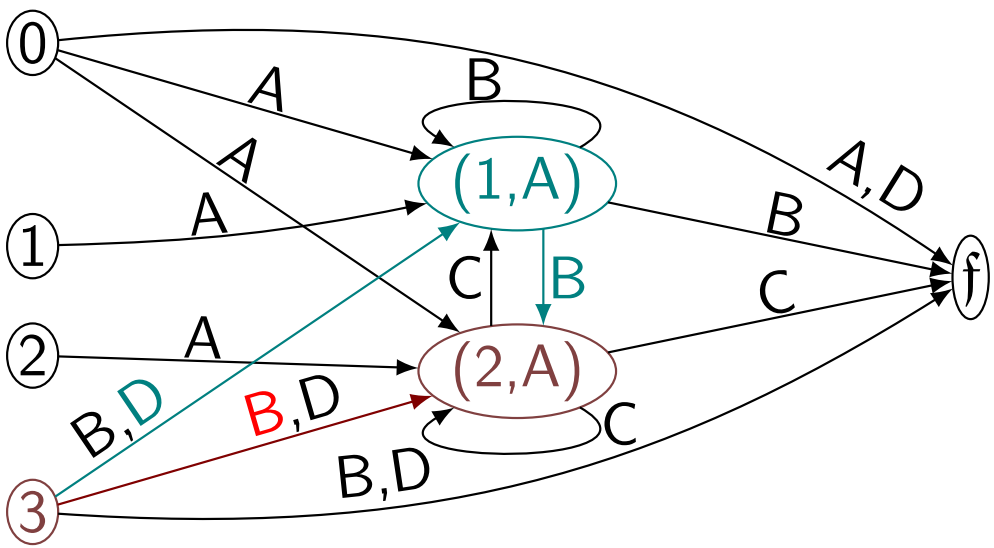
$G_1 = \{(1,0),(2,0),(3,0)\}$

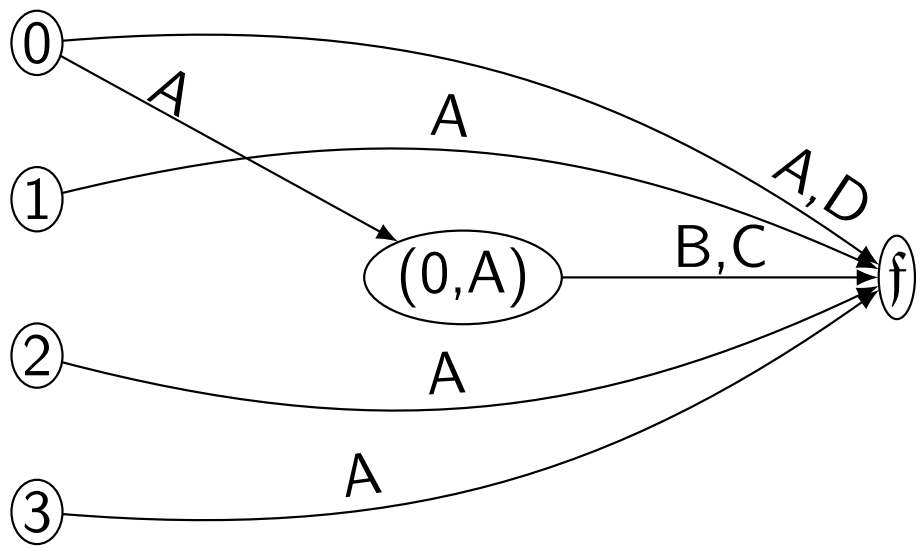
Generating automata for the left and right threads (22)

Procedure f: initially $g = 0$ Procedure h:









$G_0 = \{(0,1),(0,2),(0,3)\}$ $G_1 = \{(1,0),(2,0),(3,0)\}$

Generating automata for the left and right threads (. . .)

. . .

TMR inference system

Couple post* with thread-modular verification.

Let $\mathfrak{f}$ be a fresh symbol. Let $V = \mathsf{Glob} \cup \mathsf{Glob} \times \mathsf{Loc}$.

$$(\textsc{tmr init})\ \frac{(g,\ell) \in \mathsf{init}}{g \xrightarrow[t]{\ell_t} \mathfrak{f}}\ t{\in}n$$

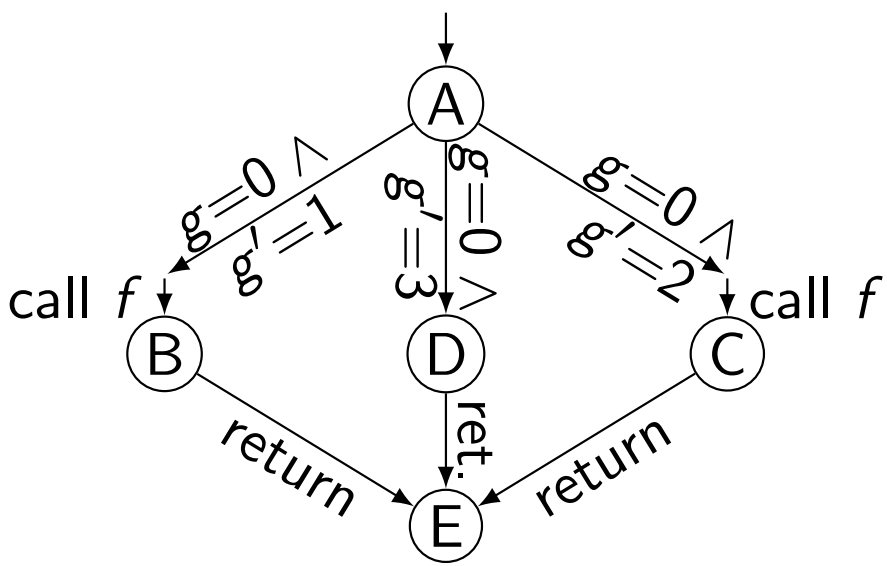
$$(\textsc{tmr step})\ \frac{((g,a),(g',b)) \in \uplus_t \qquad g \xrightarrow[t]{a} v}{g' \xrightarrow[t]{b} v \qquad (g,g') \in G_t}\ t{\in}n$$

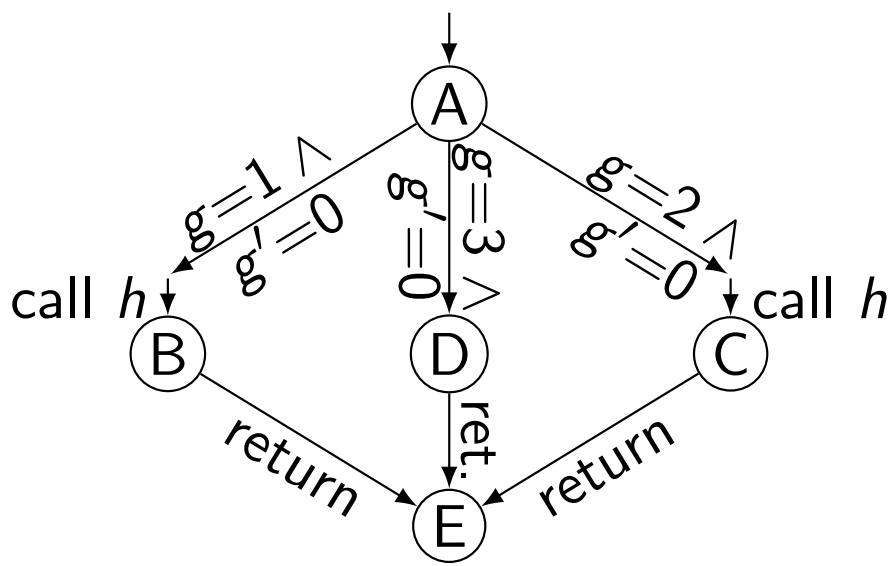
$$(\textsc{tmr push})\ \frac{g \xrightarrow[t]{a} v \qquad ((g,a),(g',b,c)) \in \downarrow_t}{g' \xrightarrow[t]{b} (g',b) \xrightarrow[t]{c} v \qquad (g,g') \in G_t}$$

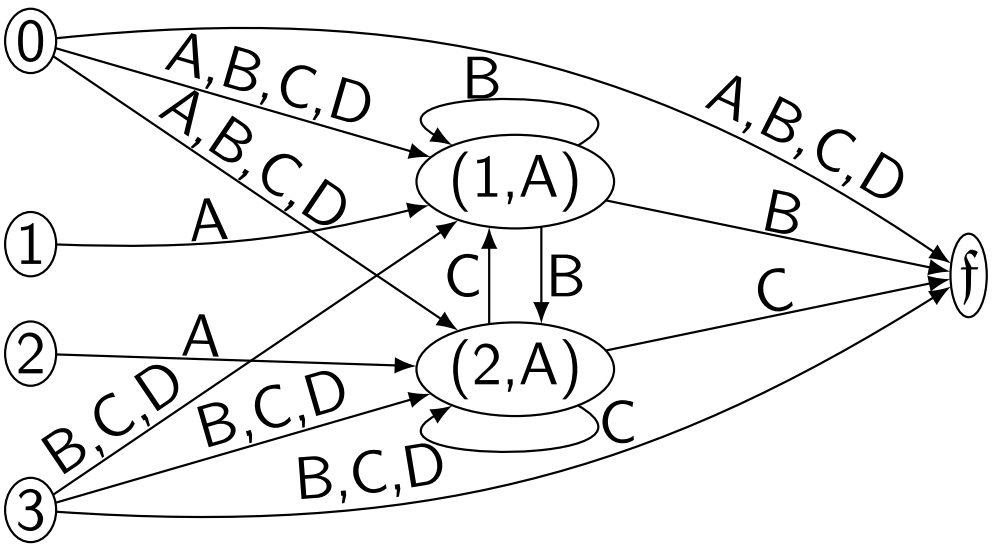
$$(\textsc{tmr pop})\ \frac{g \xrightarrow[t]{a} v \xrightarrow[t]{b} \bar{v} \qquad ((g,a,b),(g',c)) \in \uparrow_t}{g' \xrightarrow[t]{c} \bar{v} \qquad (g,g') \in G_t}$$

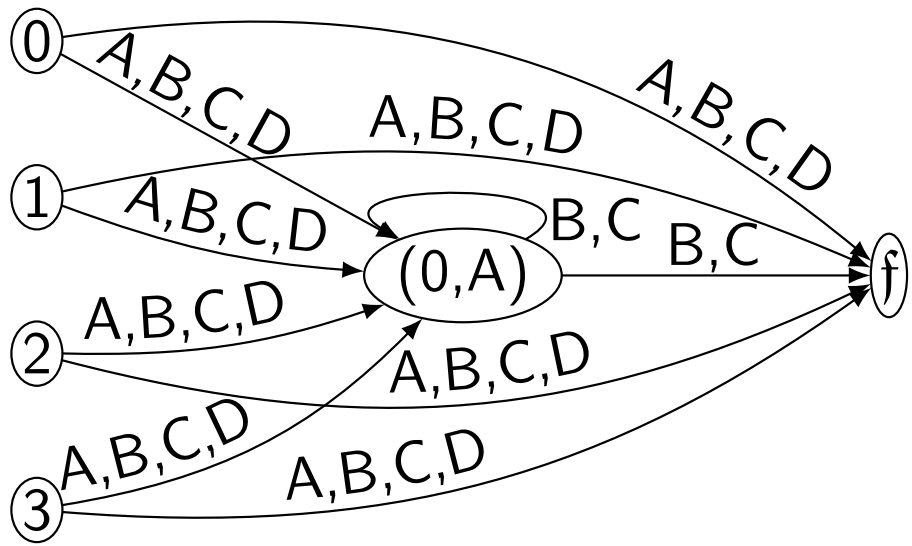
$$(\textsc{tmr env})\ \frac{(g,g') \in G_t \qquad g \xrightarrow[s]{a} v}{g' \xrightarrow[s]{a} v}\ t{\neq}s \text{ are in } n$$

Generated automata for the left and right threads





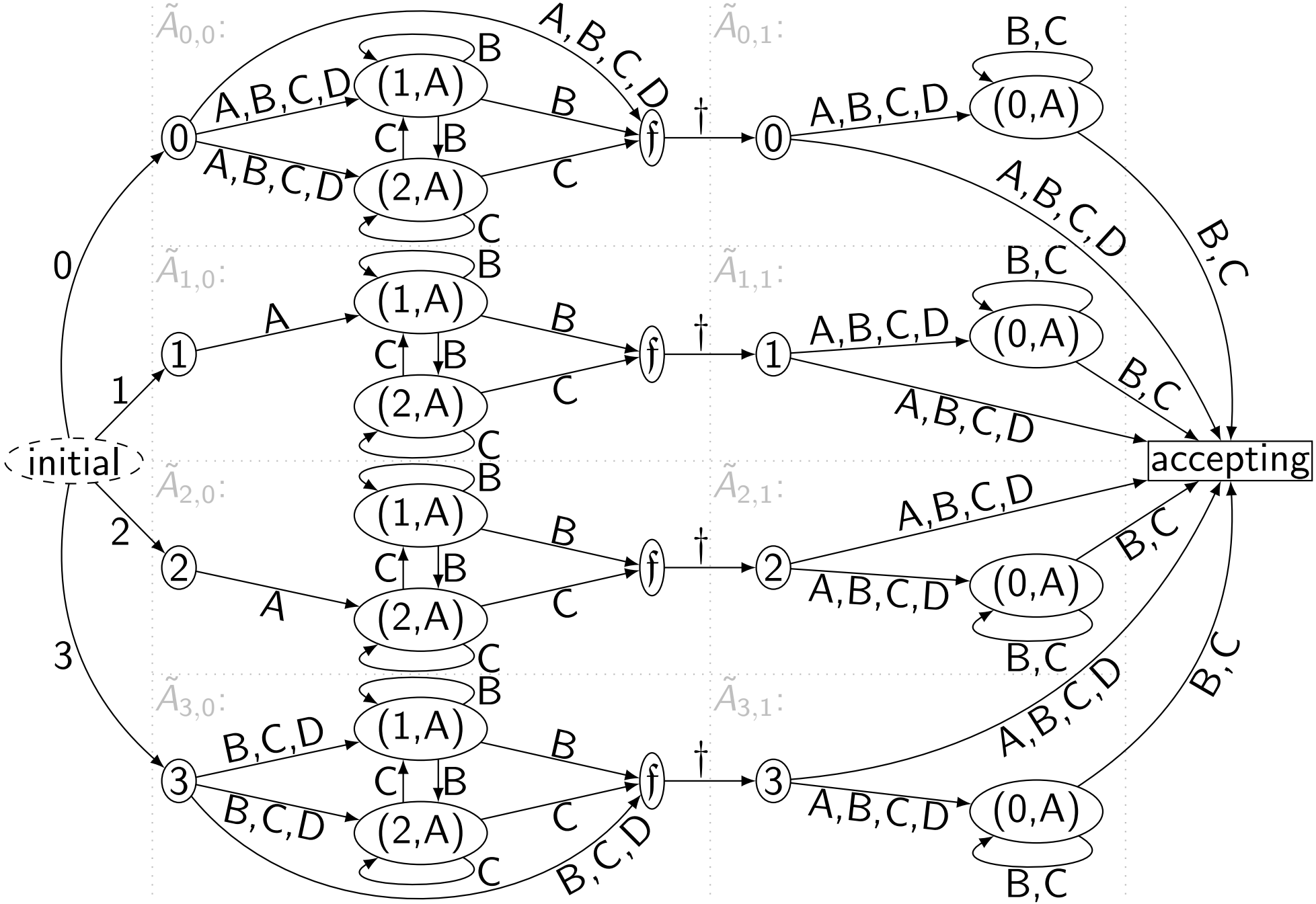




$G_0 = \{(0,1),(0,2),(0,3)\}$

$G_1 = \{(1,0),(2,0),(3,0)\}$

NFAs for the threads → NFAs for the program



Summary

The strongest multithreaded-Cartesian inductive invariant is a regular language.

Multithreaded-Cartesian abstract interpretation can be implemented in $\mathcal{O}(n \log n)$ time on a RAM under log-cost measure (and in polynomial time in other quantities).

(Proof: see paper.)

Precise running time on RAM under log-cost

- Representation of all data: lists in tables.
- Let $L(x)$ be the length of the binary representation of $x \in \mathbb{N}_0$.
- Time of a single access $= L(\text{address}) + L(\text{data}) + 1$.
- Running time of TMR:
  $\mathcal{O}\big(n(|\text{init}| + |\text{Glob}|^4|\text{Frame}|^5)(L(|\text{init}|) + L(n) + L(|\text{Glob}|) + L(|\text{Frame}|))\big)$.
- Constructing the full NFA after constructing the threads' NFAs is asymptotically negligible.
- In the input length size, it is
  $\mathcal{O}((\text{input length})^2 L(\text{input length}))$.

Questions?