Snapshottable stores

Clément Allain (INRIA Paris)
Basile Clément (OCamlPro)
Alexandre Moine (INRIA Paris)
Gabriel Scherer (INRIA Paris)

September 25, 2024

From OCaml to Zoo

ocaml2zoo

Zoo

Specification: a simple store...

$$\frac{\{\,\mathrm{True}\,\}}{\texttt{create}\ ()}\ \{\,t.\,\mathrm{store}\ t\ \emptyset\,\}$$

$$\frac{\{\,\mathrm{store}\ t\ \sigma\,\}}{\texttt{ref}\ t\ v}\ \{\,r.\,r \notin \mathrm{dom}(\sigma) * \mathrm{store}\ t\ \sigma[r \mapsto v]\,\}$$

$$\frac{\{\,\mathrm{store}\ t\ \sigma * \sigma(r) = v\,\}}{\texttt{get}\ t\ r}\ \{\,v.\,\mathrm{store}\ t\ \sigma\,\}$$

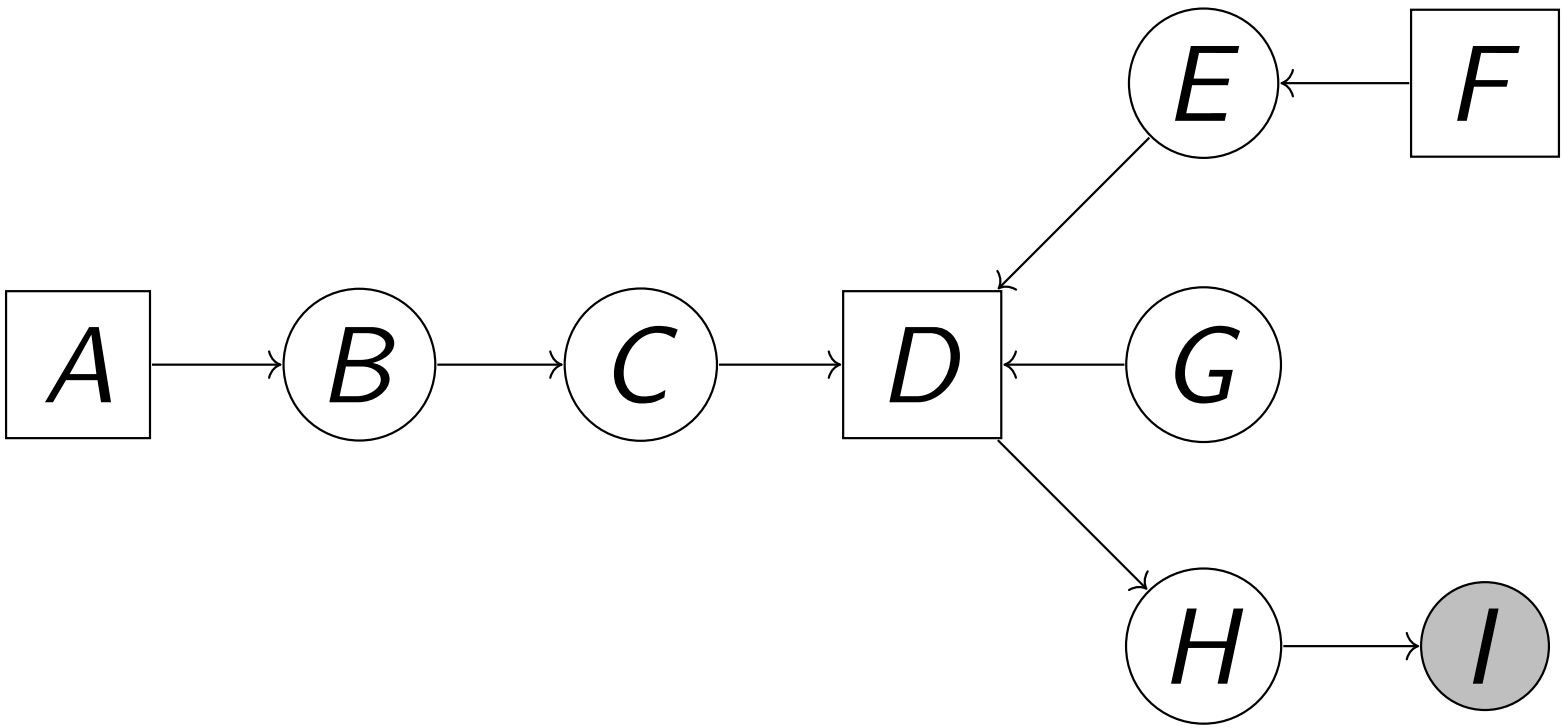
$$\frac{\{\,\mathrm{store}\ t\ \sigma * r \in \mathrm{dom}(\sigma)\,\}}{\texttt{set}\ t\ r\ v}\ \{\,v.\,\mathrm{store}\ t\ \sigma[r \mapsto v]\,\}$$

. . . with persistent snapshots

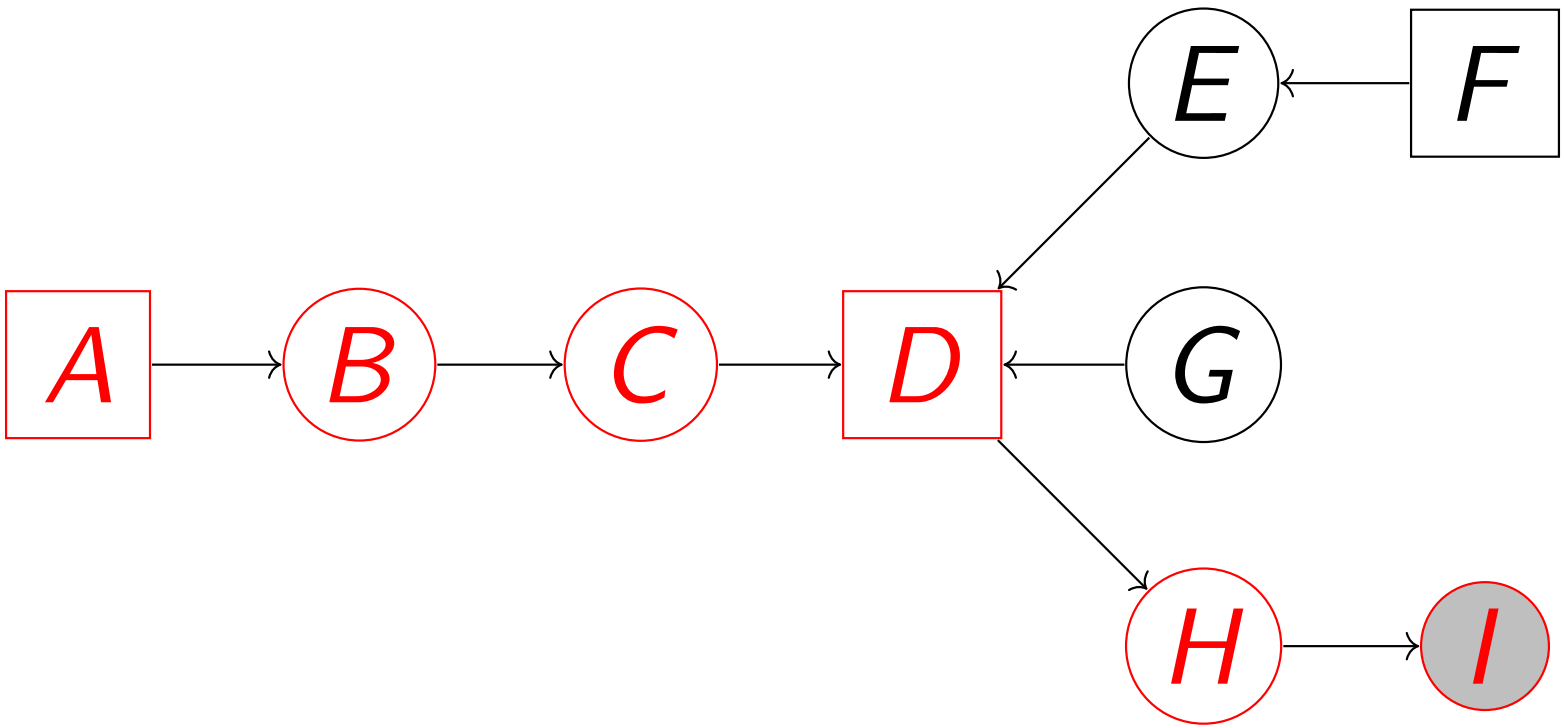
$$\frac{\{\,\mathrm{store}\ t\ \sigma\,\}}{\dfrac{\texttt{capture}\ t}{\{\,s.\,\mathrm{store}\ t\ \sigma * \mathrm{snapshot}\ t\ s\ \sigma\,\}}}$$

$$\frac{\{\,\mathrm{store}\ t\ \sigma * \mathrm{snapshot}\ t\ s\ \sigma'\,\}}{\dfrac{\texttt{restore}\ t\ s}{\{\,().\,\mathrm{store}\ t\ \sigma'\,\}}}$$

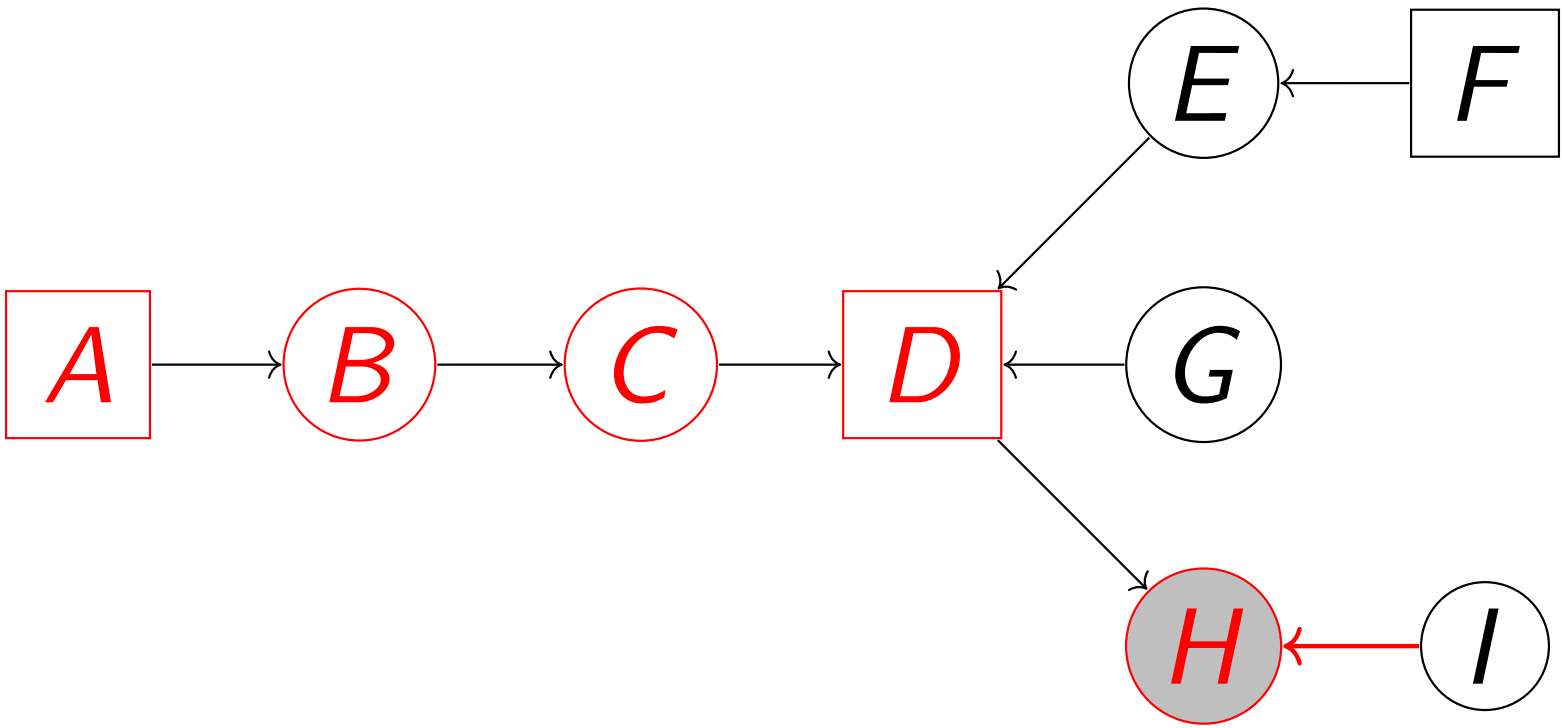
Implementation *without* elision



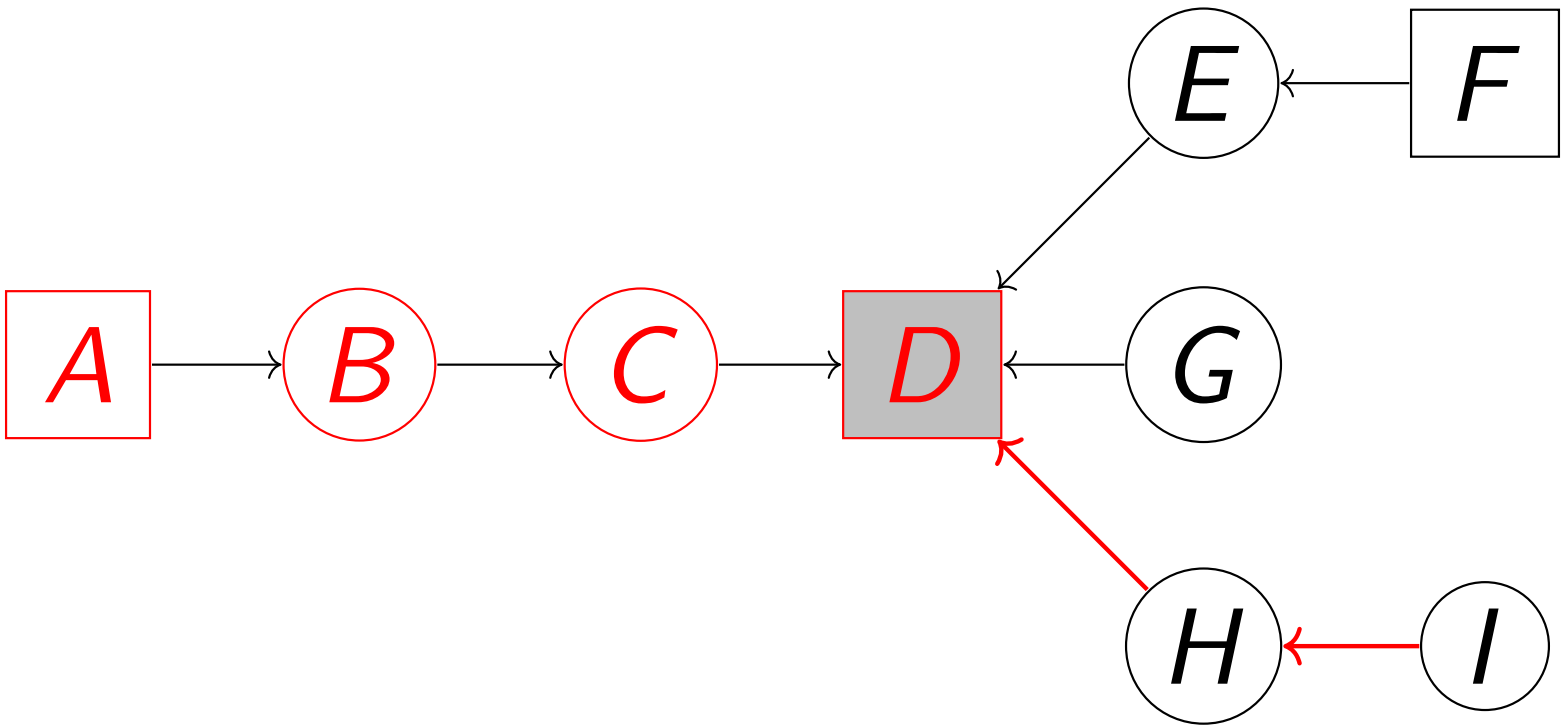
Rerooting *without* elision



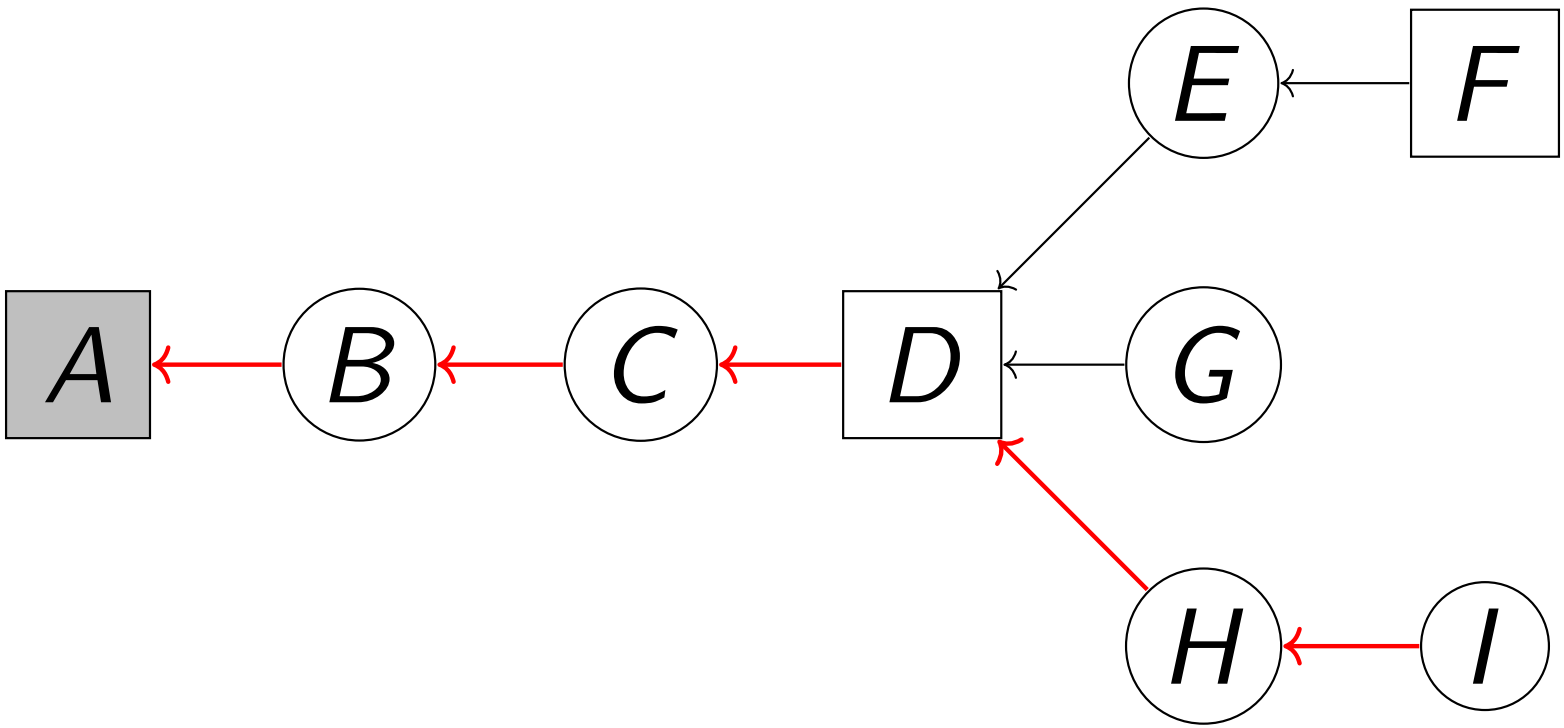
Rerooting *without* elision



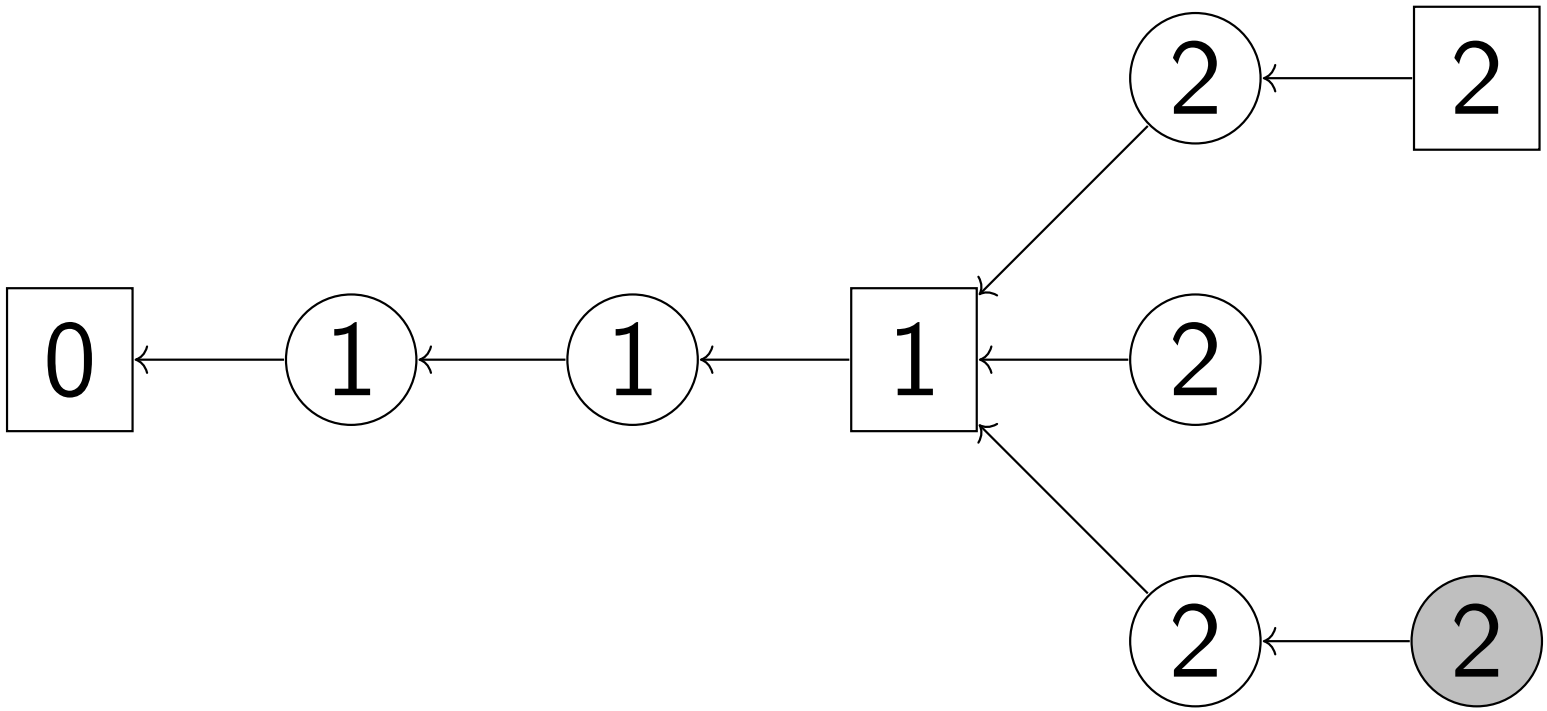
Rerooting *without* elision



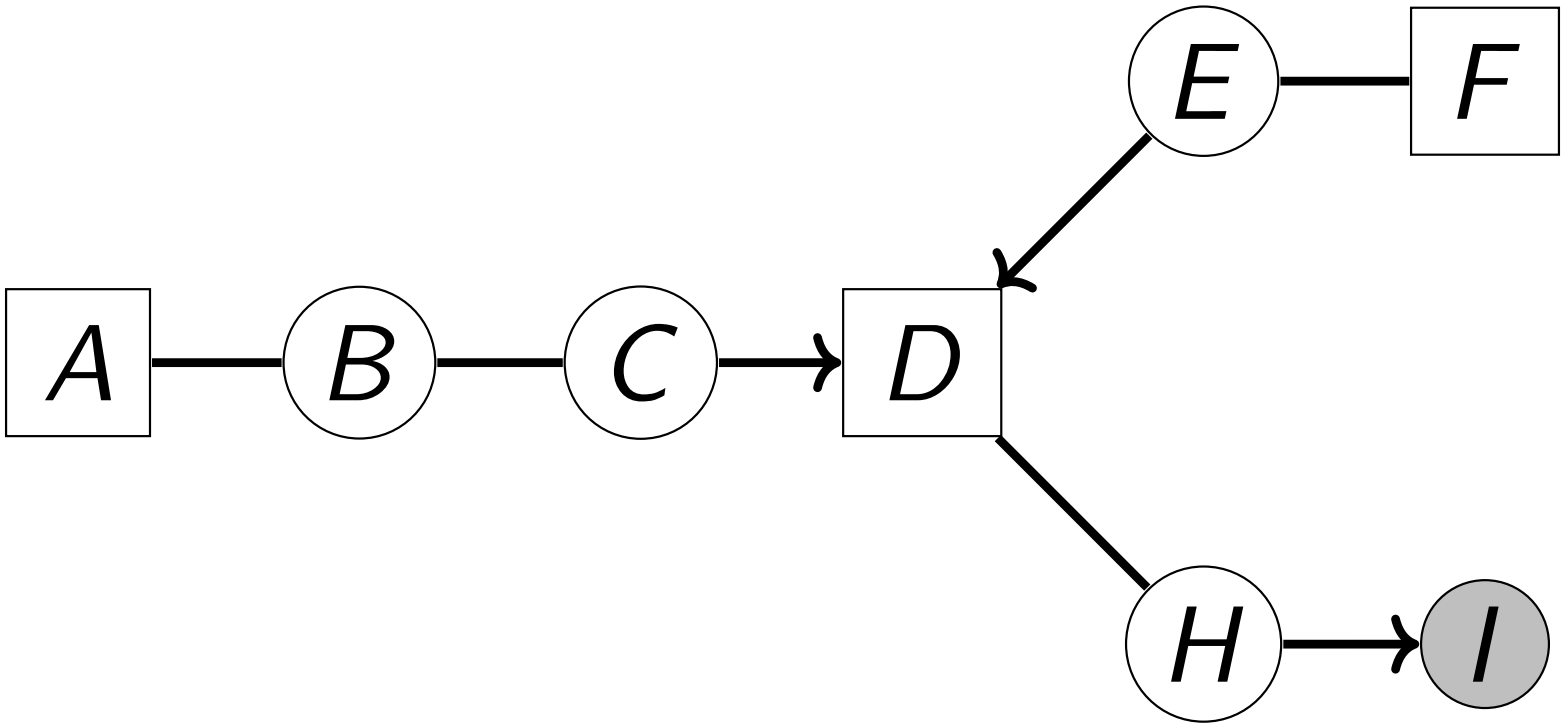
Rerooting *without* elision



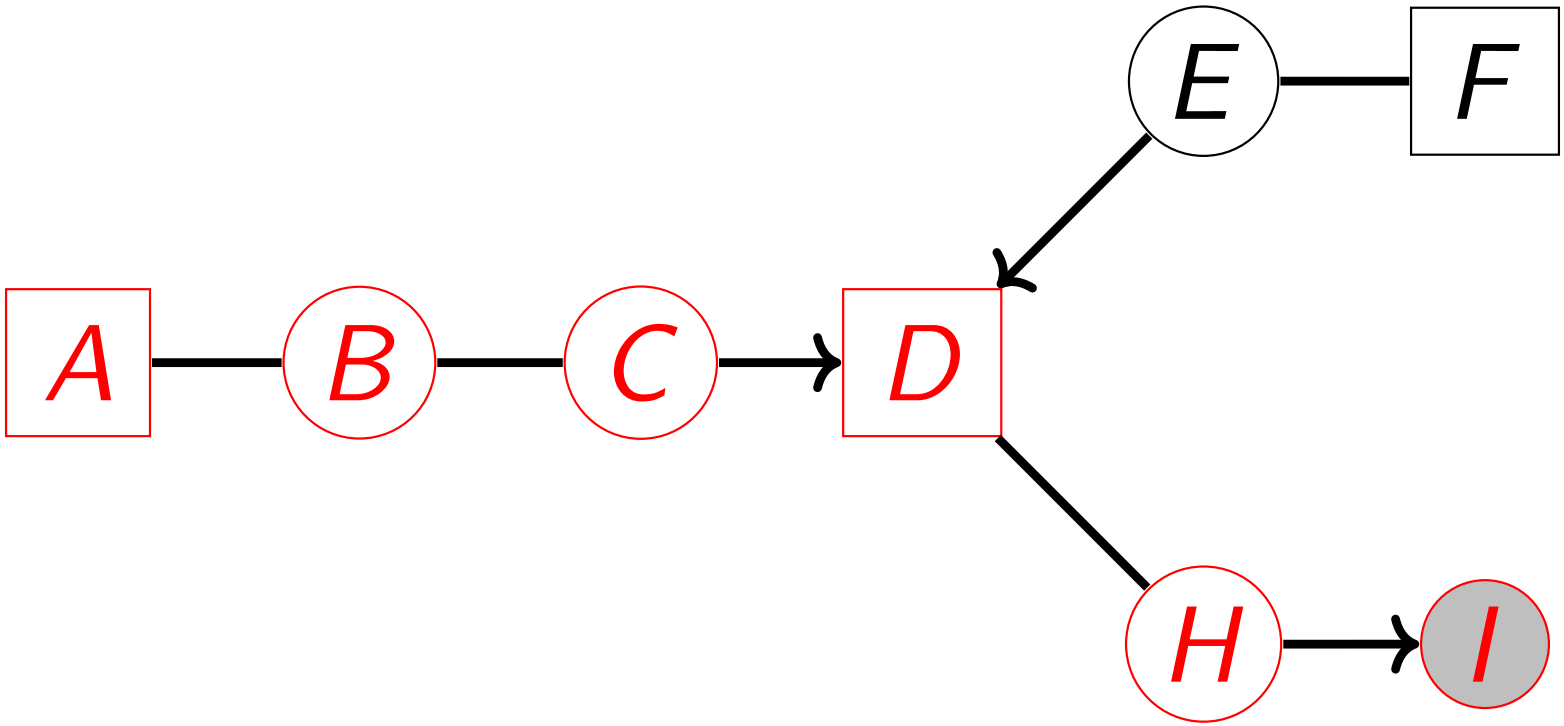
Historical tree & generations



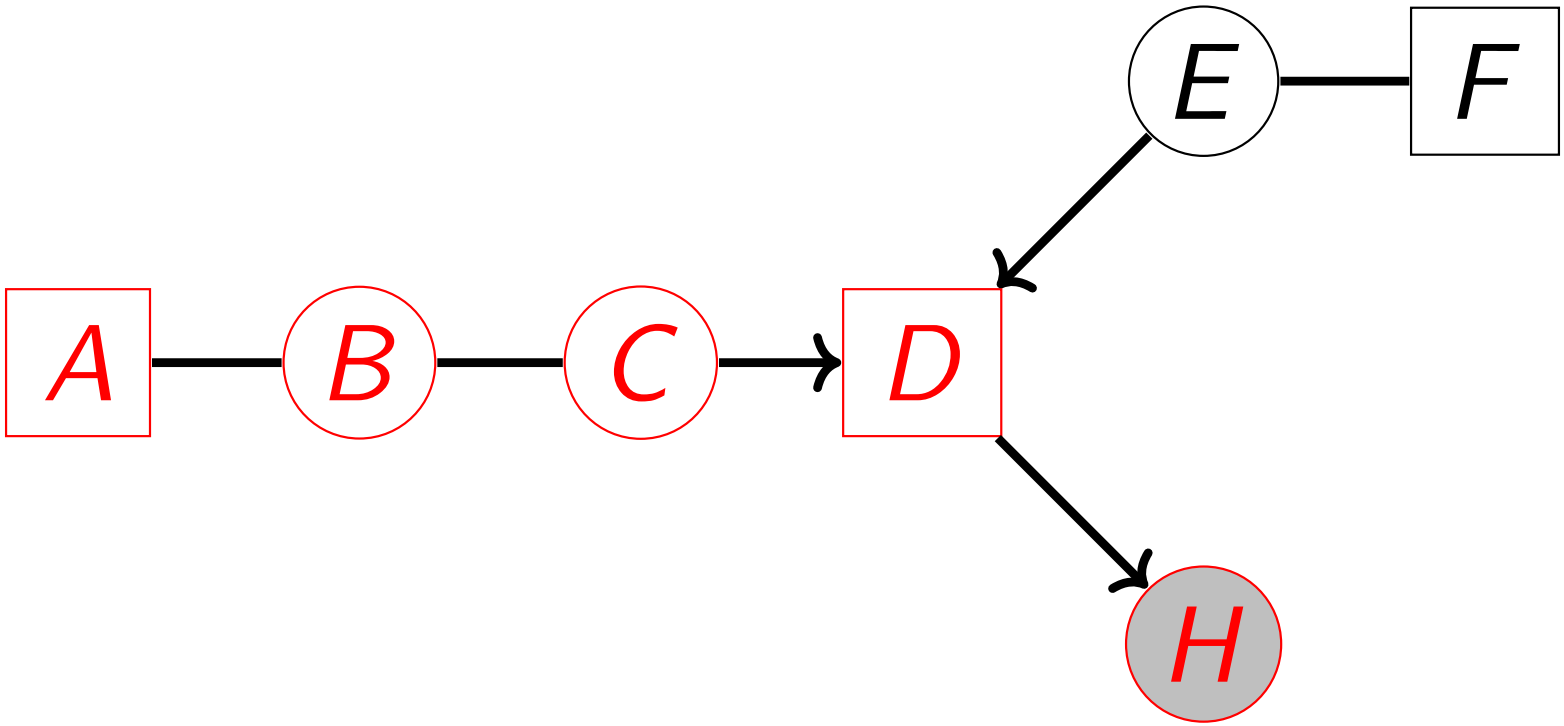
Implementation *with* elision



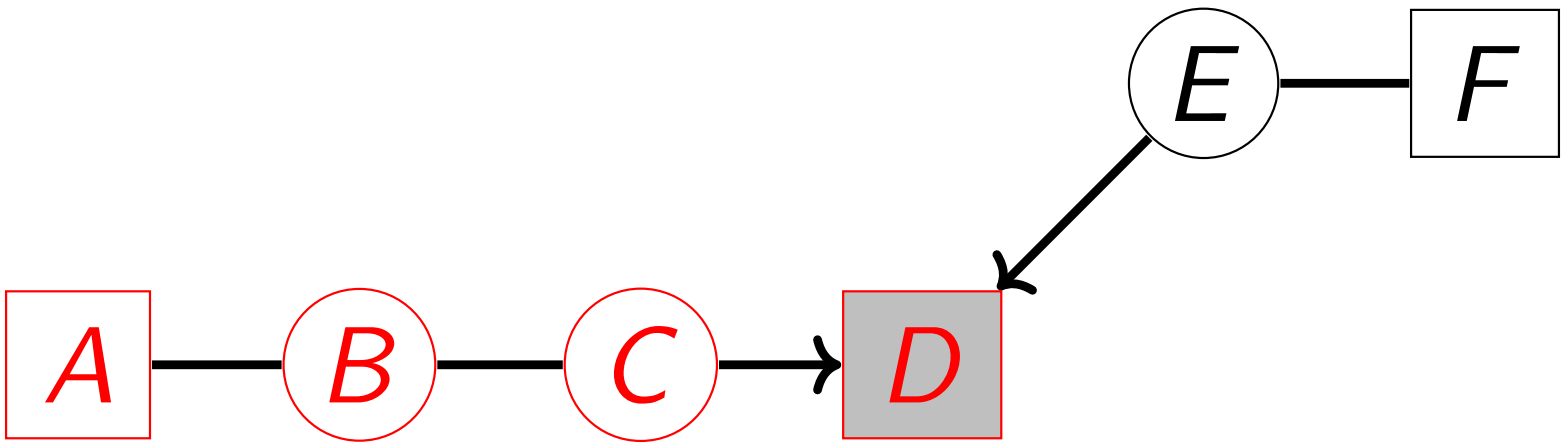
Rerooting *with* elision



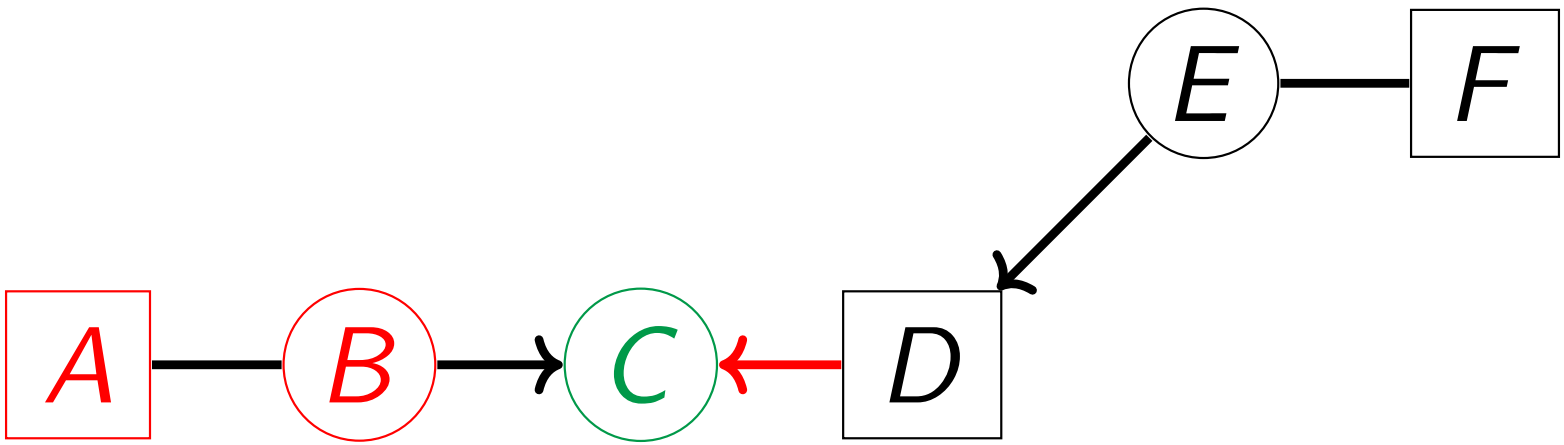
Rerooting *with* elision



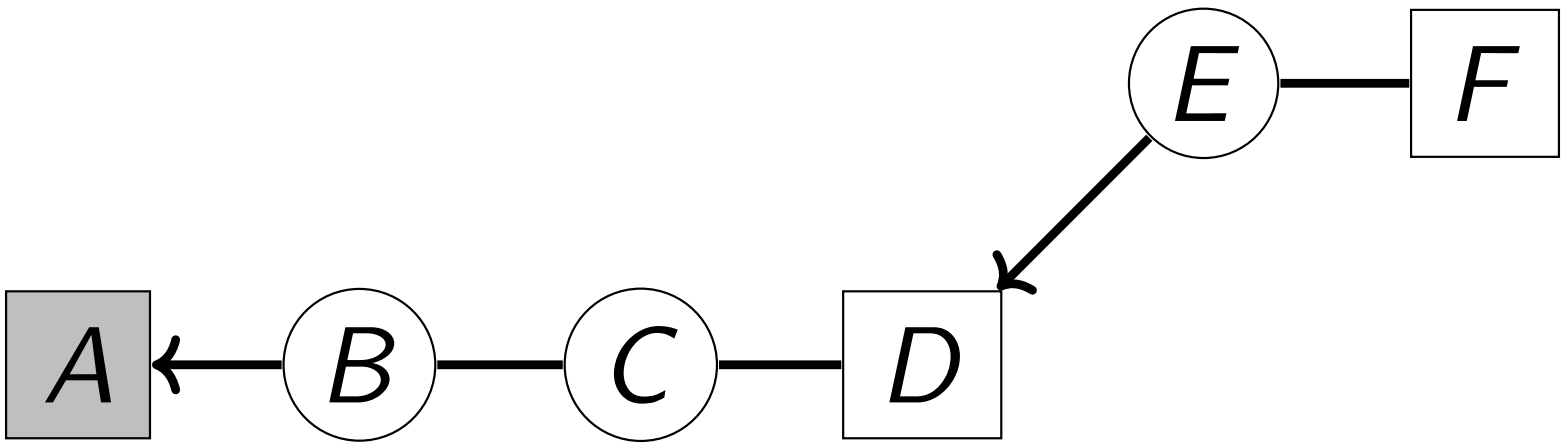
Rerooting *with* elision



Rerooting *with* elision



Rerooting *with* elision



Merci de votre attention!