Part II:

Symbolic reachability for prefix rewriting

Case study: Drawing skylines

```
static Random r = new Random();

static void m() {
  if (r.nextBoolean()) {
    s(); right(); if (r.nextBoolean()) m();
  }
  else { up(); m(); down(); }
}

static void s() {
    if (r.nextBoolean()) return;
    up(); m(); down();
}

public static void main() { s(); }
```

Model

```
static void s() {
  s0: if (r.nextBoolean())
  s1: return;
  s2: up();
  s3: m();
  s4: down();
  s5:
}
```

var `st`:**stack** `of` $\{s_0, \ldots, s_5, \ldots\}$

$s_0 \longrightarrow s_1 \qquad s_0 \longrightarrow s_2$

$s_1 \longrightarrow \epsilon$

$s_2 \longrightarrow up_0\, s_3$

$s_3 \longrightarrow m_0\, s_4$

$s_4 \longrightarrow down_0\, s_5$

$s_5 \longrightarrow \epsilon$

Symbolic reachability in prefix rewriting

Recall: program state $(\ g, \ell, n, (\ell_1, n_1) \ldots (\ell_k, n_k)\ )$ modelled as a word $g\,\langle \ell, n\rangle\,\langle \ell_1, n_1\rangle\ \ldots\ \langle \ell_k, n_k\rangle$.

Denote by G the alphabet of valuations of globals.

Denote by L the alphabet of pairs $\langle \ell, n\rangle$.

The set of possible programs states is given by $G\,L^*$

A subset of GL^* words is regular if it can be recognized by a finite automaton.

Typically, the sets I and D of initial and dangerous program states are regular sets. (Even very simple ones, like $g\,I\,L^*$.)

Challenge: show that if $S \subseteq GL^*$ is (effectively) regular, then so are $pre^*(S)$ and $post^*(S)$.

This gives a procedure to check if $I \cap pre^*(D) = \emptyset$ or $post^*(I) \cap D = \emptyset$.

Symbolic search

Forward symbolic search

Initialize $S := I$

Iterate $S := S \cup post(S)$ until fixpoint.

Backward search: replace I by D, replace *post* by *pre*.

Questions:

- Are $S \cup post(S)$ and $S \cup pre(S)$ regular for regular S ?
- Does the search terminate ?

We answer these questions for backward search, the forward case is similar.

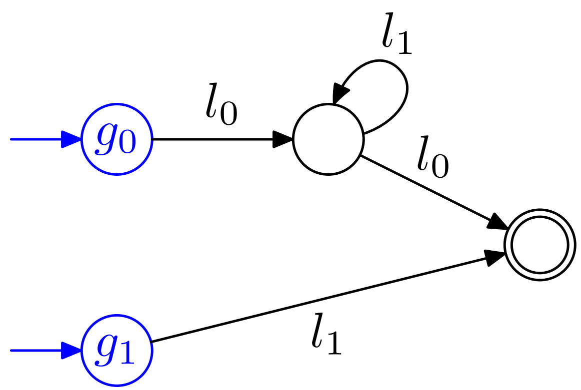
If S regular, then $S \cup pre(S)$ regular

We represent a regular set $S \subseteq G\,L^*$ by an NFA.

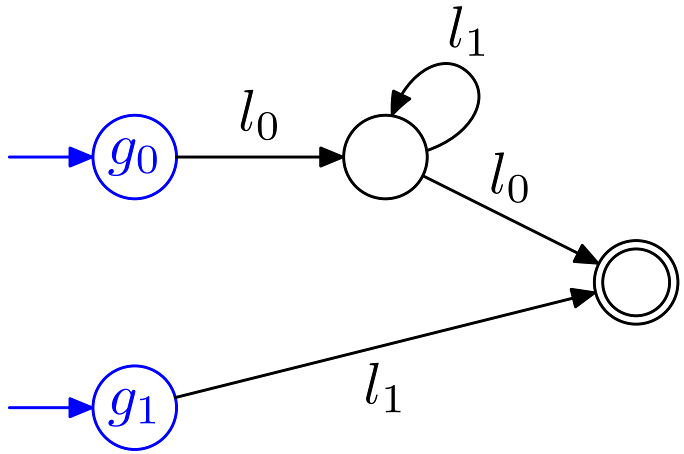
- G as set of initial states, L as alphabet.
- gw recognized if $g \xrightarrow{w} q$ for some final state q.

Example: $G = \{g_0, g_1\}$ and $L = \{l_0, l_1\}$

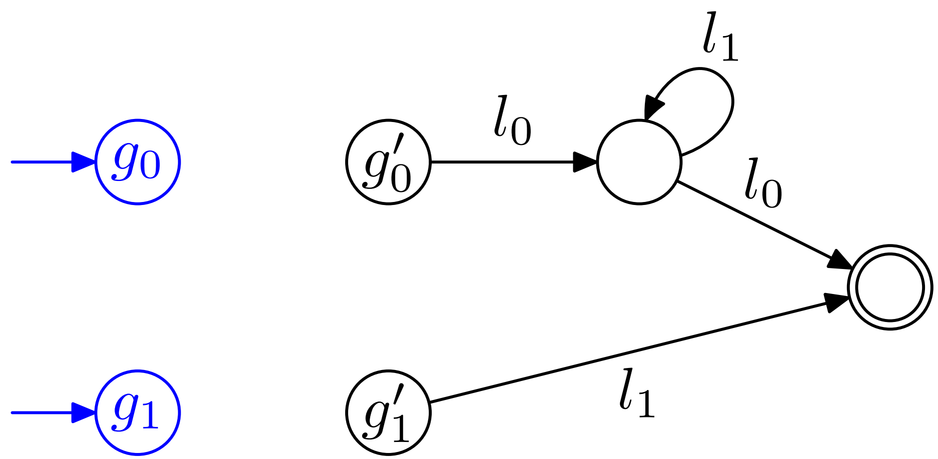
Automaton coding the set $g_0\, l_1^*\, l_0 \;+\; l_1\, l_1$:



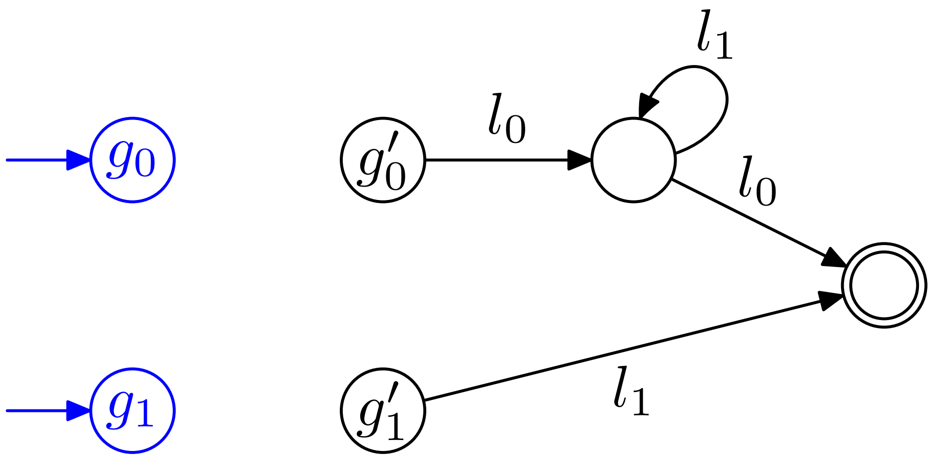
$$R = \{ \quad g_0\, l_0 \rightarrow g_0 \quad , \quad g_1\, l_1 \rightarrow g_0 \quad , \quad g_1\, l_1 \rightarrow g_1\, l_1\, l_0 \quad \}$$



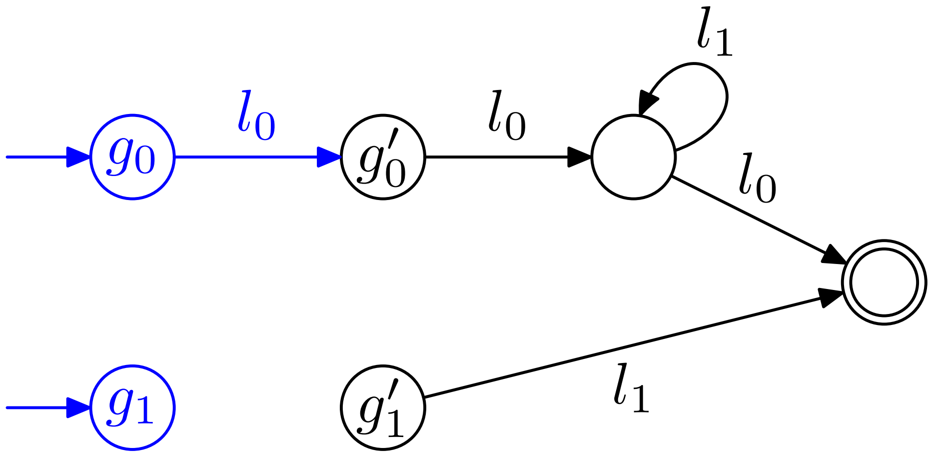
$$R = \{ \quad g_0\, l_0 \rightarrow g_0 \quad , \quad g_1\, l_1 \rightarrow g_0 \quad , \quad g_1\, l_1 \rightarrow g_1\, l_1\, l_0 \quad \}$$



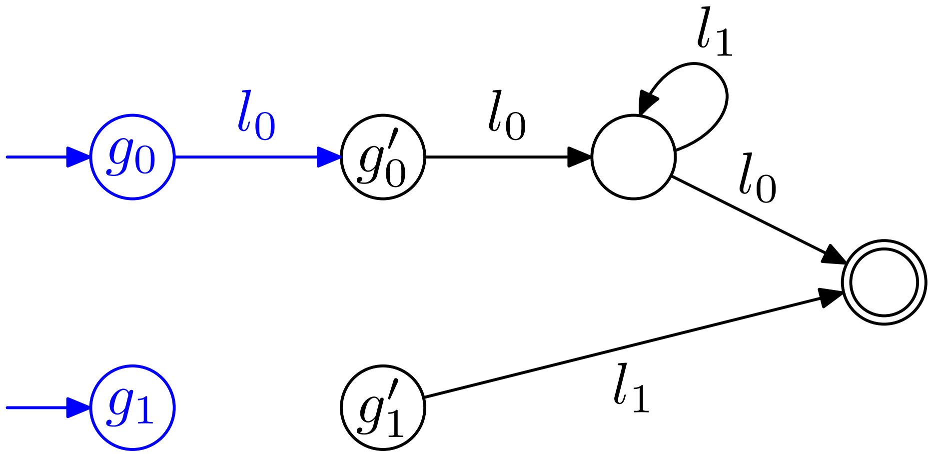
$g_0 \, l_0 \rightarrow g_0$



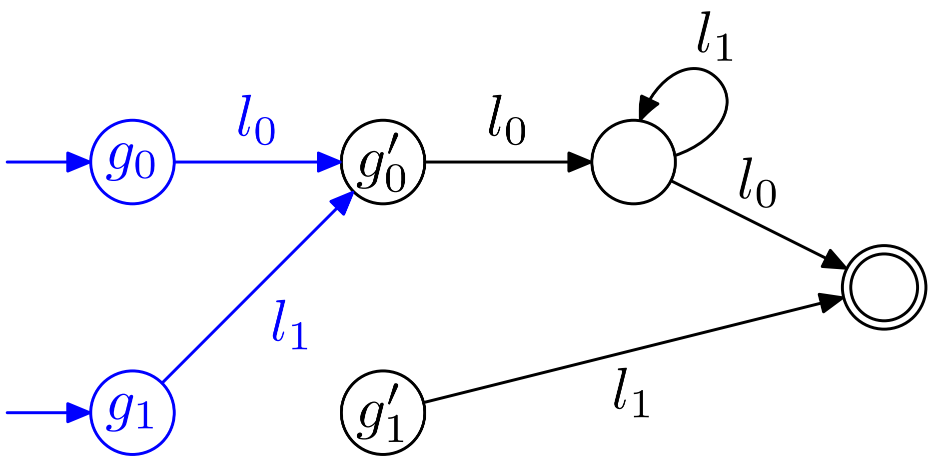
$g_0\, l_0 \rightarrow g_0$



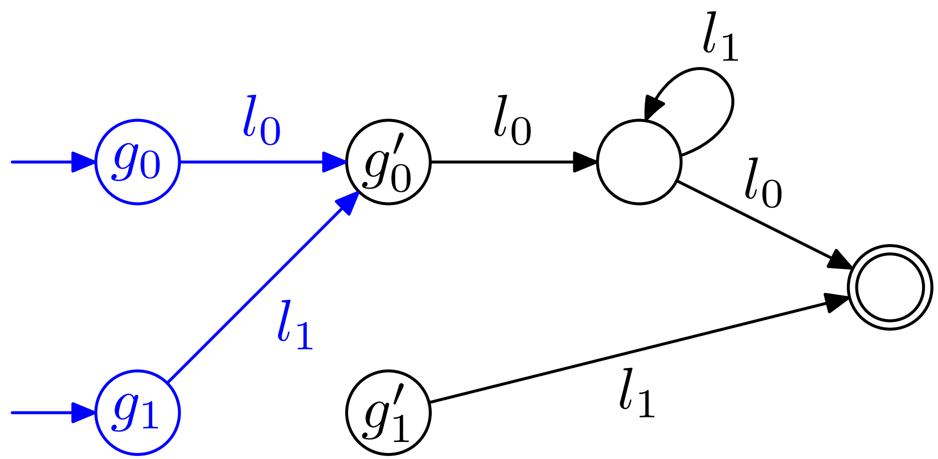
$g_1\, l_1 \rightarrow g_0$



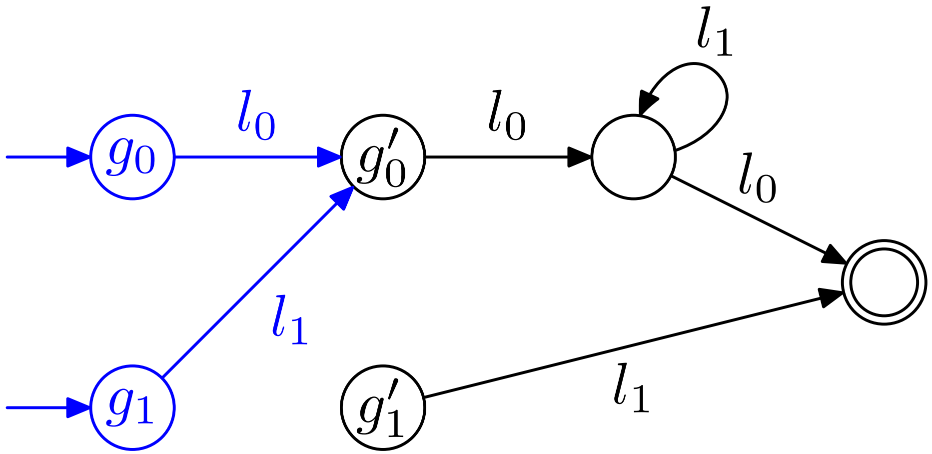
$g_1 \, l_1 \rightarrow g_0$



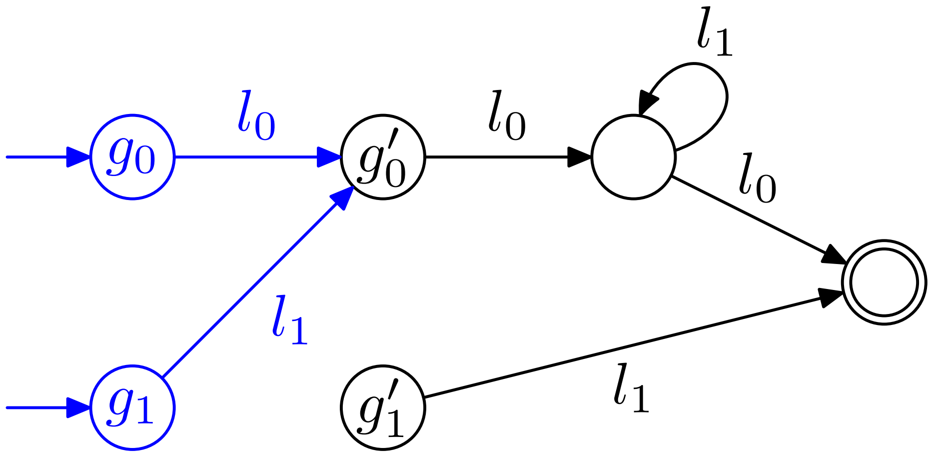
$g_1\ l_1 \rightarrow g_1\ l_1\ l_0$



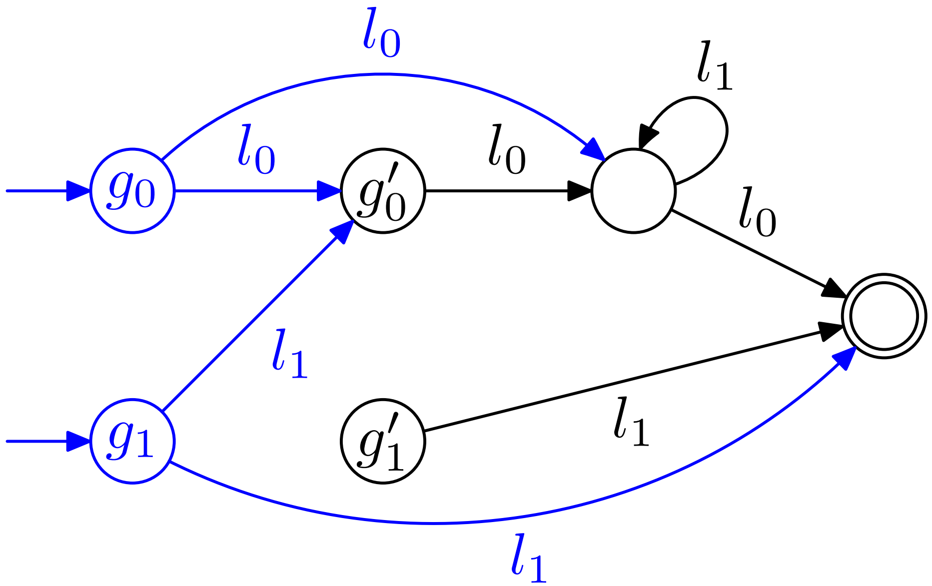
$g_1\ l_1 \rightarrow g_1\ l_1\ l_0$



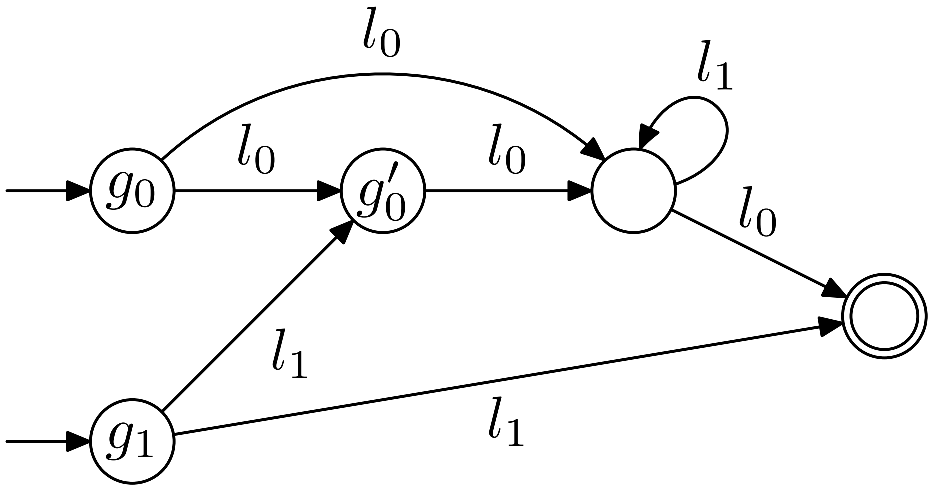
$$R = \{ \quad g_0\, l_0 \to g_0 \quad , \quad g_1\, l_1 \to g_0 \quad , \quad g_1\, l_1 \to g_1\, l_1\, l_0 \quad \}$$



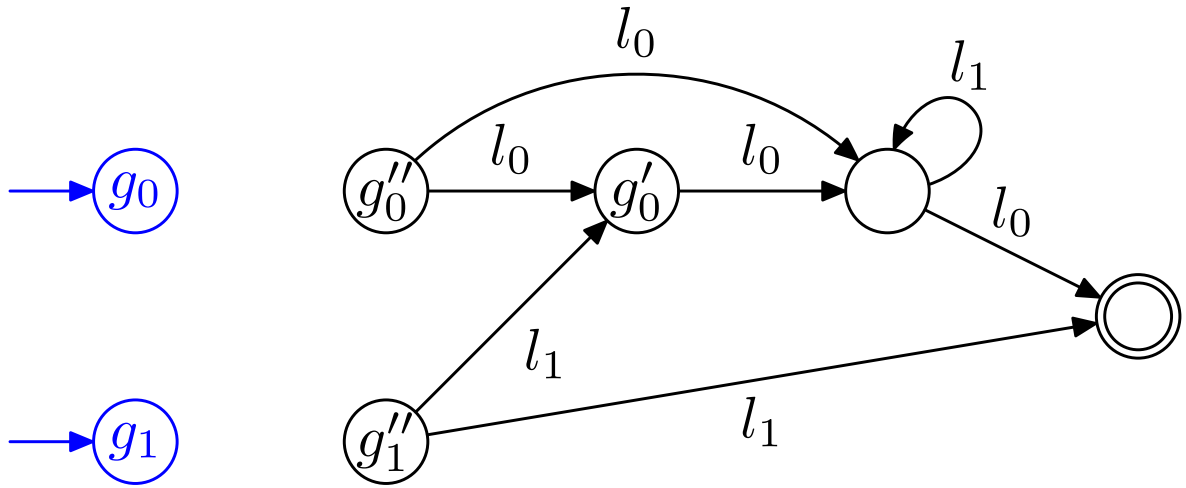
$$R = \{ \quad g_0\, l_0 \rightarrow g_0 \quad , \quad g_1\, l_1 \rightarrow g_0 \quad , \quad g_1\, l_1 \rightarrow g_1\, l_1\, l_0 \quad \}$$



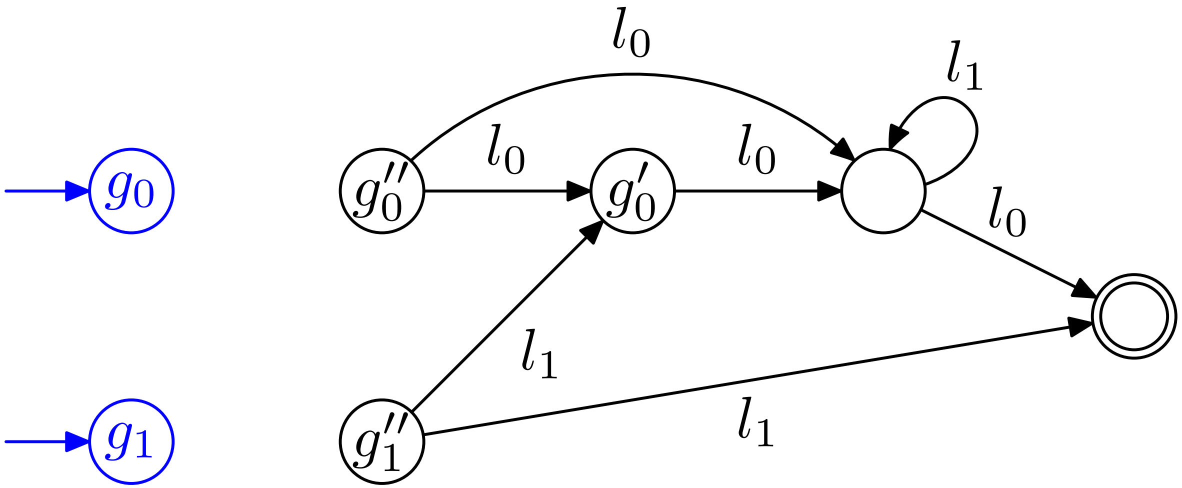
$$R = \{ \quad g_0\, l_0 \to g_0 \quad , \quad g_1\, l_1 \to g_0 \quad , \quad g_1\, l_1 \to g_1\, l_1\, l_0 \quad \}$$



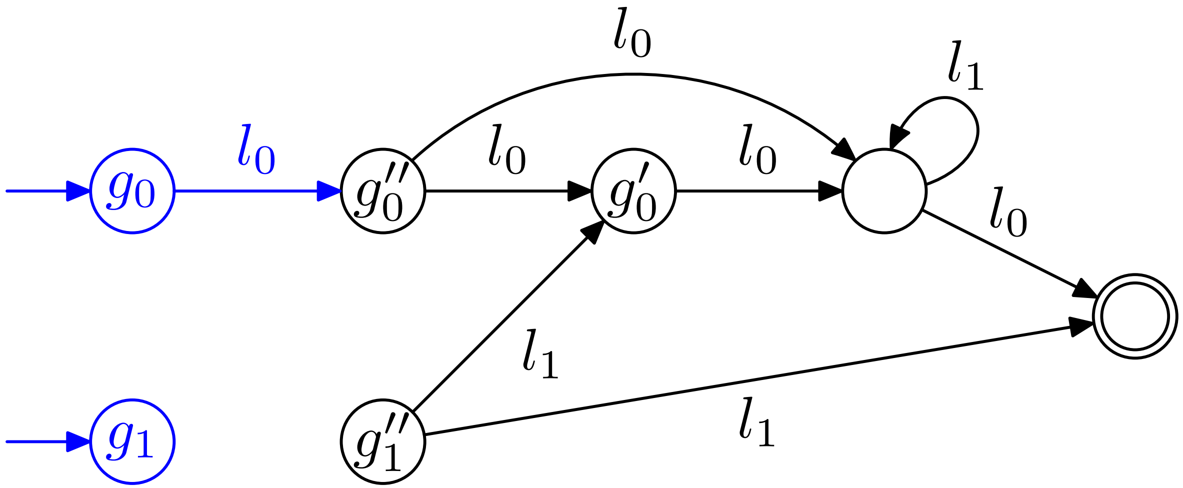
$$R = \{ \quad g_0\, l_0 \rightarrow g_0 \quad , \quad g_1\, l_1 \rightarrow g_0 \quad , \quad g_1\, l_1 \rightarrow g_1\, l_1\, l_0 \quad \}$$



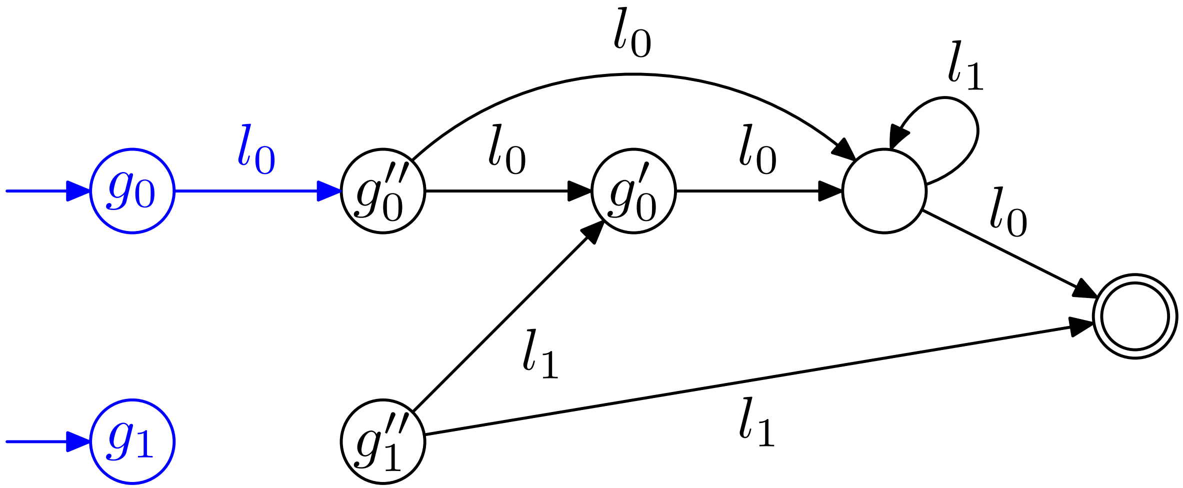
$g_0 \, l_0 \rightarrow g_0$



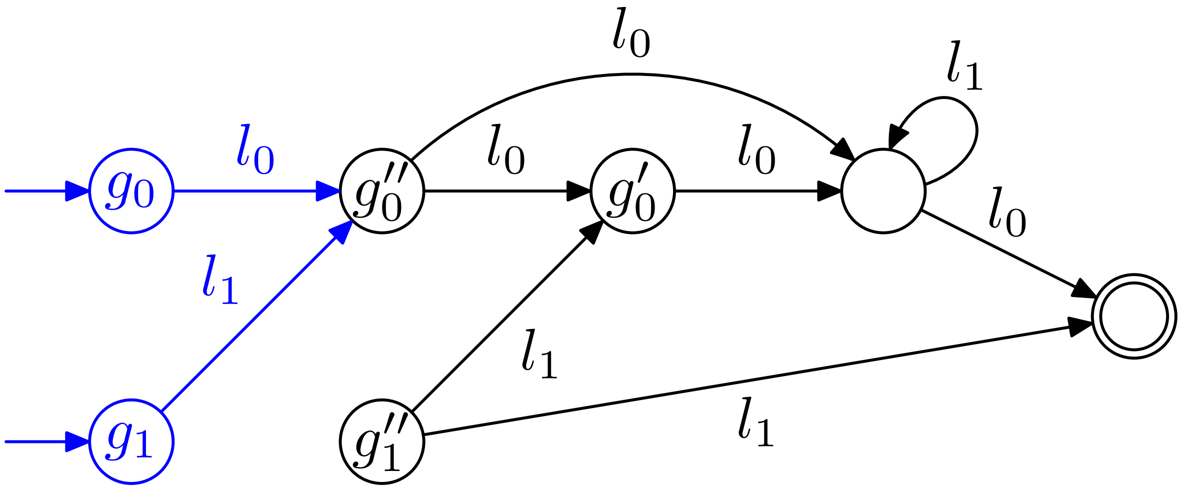
$g_0\ l_0 \rightarrow g_0$



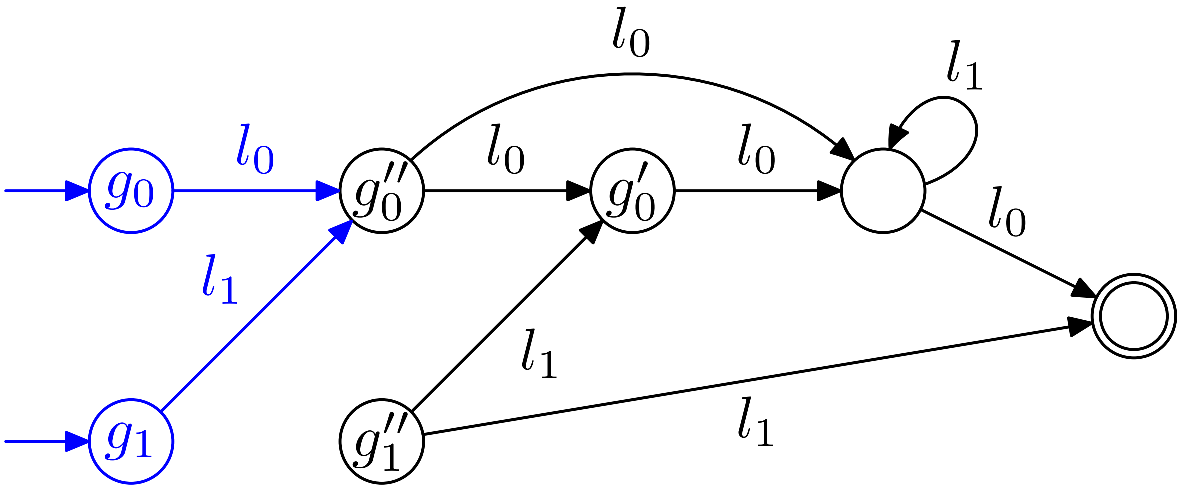
$g_1\ l_1 \rightarrow g_0$



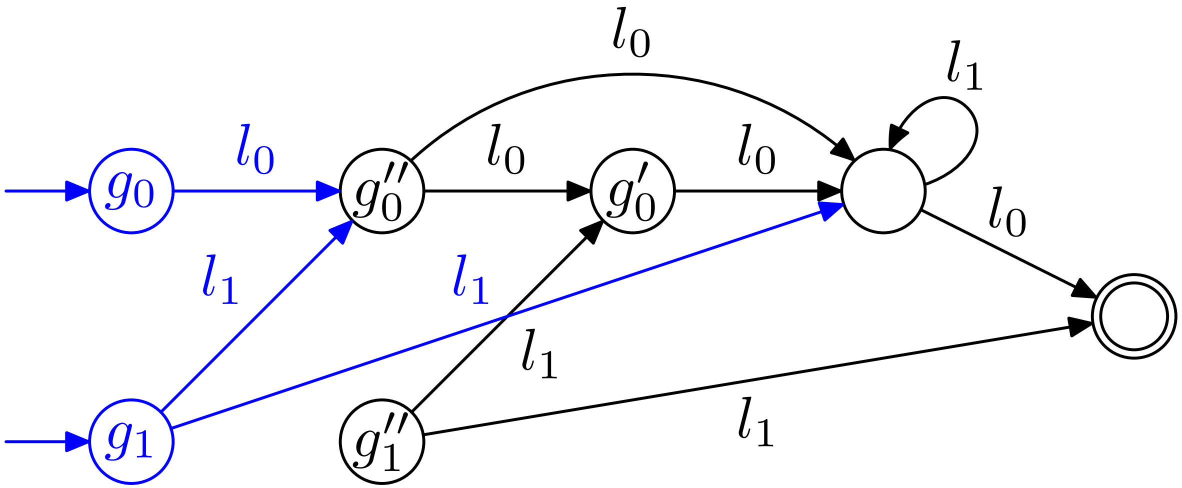
$g_1\ l_1 \rightarrow g_0$



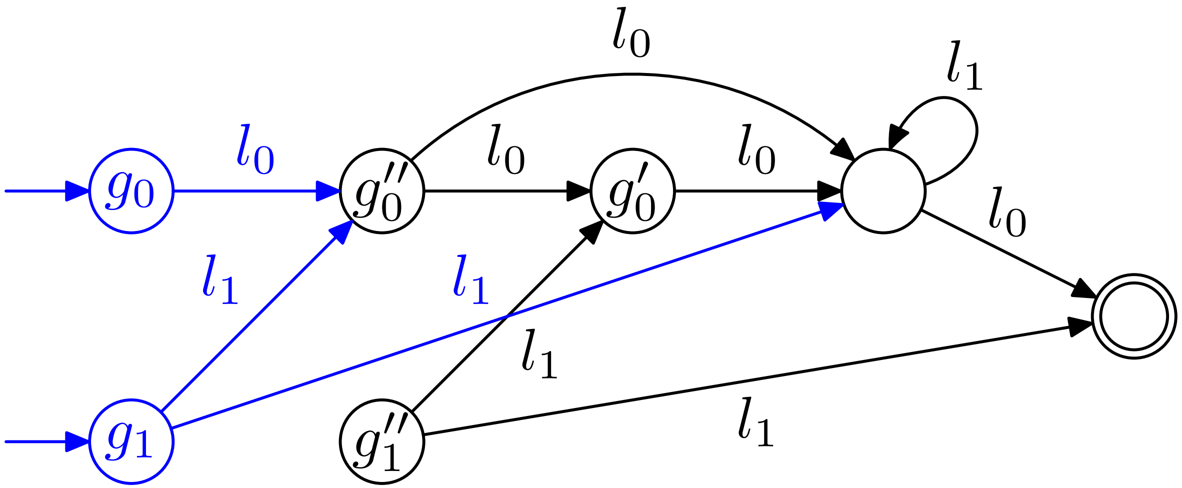
$g_1\ l_1 \rightarrow g_1\ l_1\ l_0$



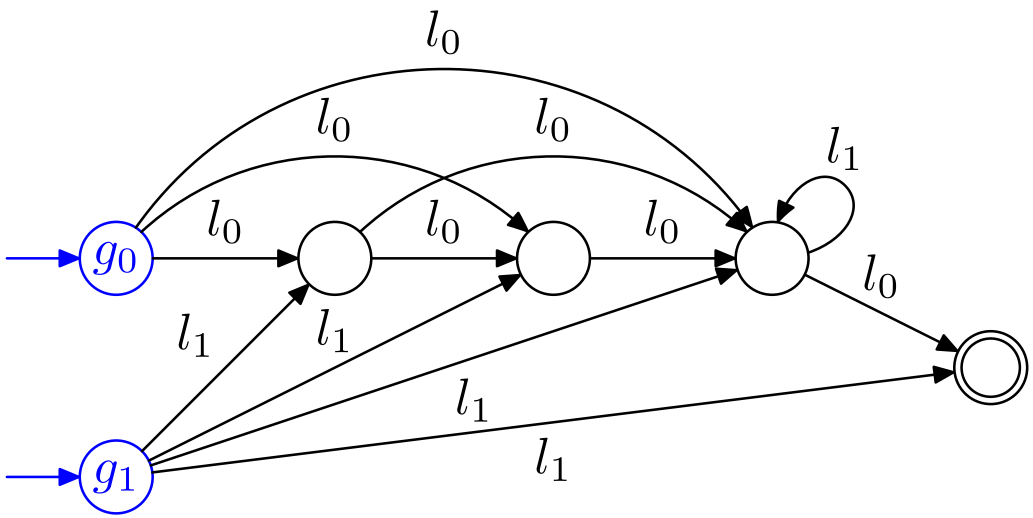
$g_1\ l_1 \to g_1\ l_1\ l_0$



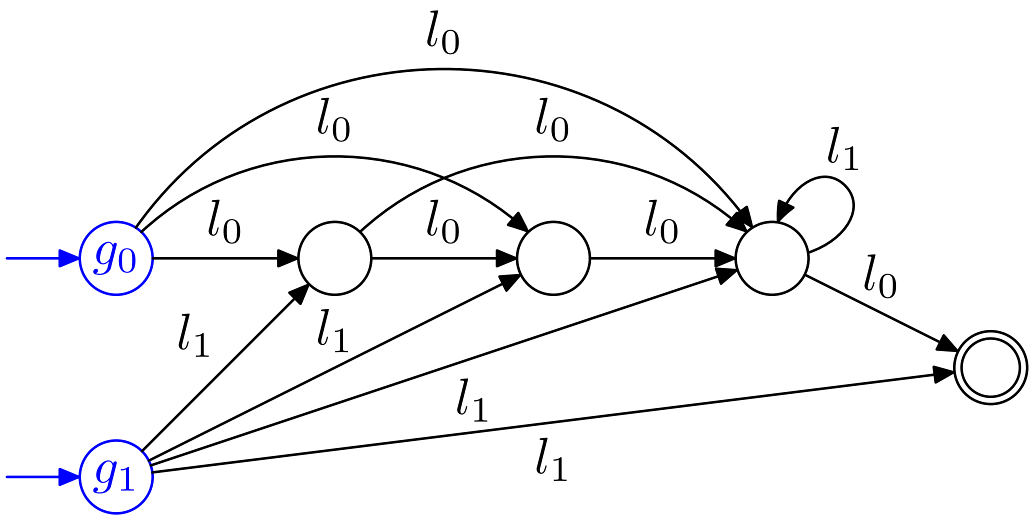
$$R = \{ \quad g_0\, l_0 \rightarrow g_0 \quad , \quad g_1\, l_1 \rightarrow g_0 \quad , \quad g_1\, l_1 \rightarrow g_1\, l_1\, l_0 \quad \}$$



$$R = \{ \quad g_0\, l_0 \rightarrow g_0 \quad , \quad g_1\, l_1 \rightarrow g_0 \quad , \quad g_1\, l_1 \rightarrow g_1\, l_1\, l_0 \quad \}$$



$$R = \{ \quad g_0\, l_0 \to g_0 \quad , \quad g_1\, l_1 \to g_0 \quad , \quad g_1\, l_1 \to g_1\, l_1\, l_0 \quad \}$$



Termination fails

$G = \{g_0, g_1\}$, $L = \{l_0, l_1\}$

$R = \{ \, g_0 \, l_0 \rightarrow g_0 \, , \, g_1 \, l_1 \rightarrow g_0 \, , \, g_1 \, l_1 \rightarrow g_1 \, l_1 \, l_0 \, \}$

Termination fails

$G = \{g_0, g_1\}, L = \{l_0, l_1\}$

$R = \{ g_0\, l_0 \rightarrow g_0 \, , \, g_1\, l_1 \rightarrow g_0 \, , \, g_1\, l_1 \rightarrow g_1\, l_1\, l_0 \}$

$$S_0 \;=\; D \qquad\qquad = \; g_0\, l_0\, l_1^*\, l_0 \; + \; g_1\, l_1$$

Termination fails

$G = \{g_0, g_1\}$, $L = \{l_0, l_1\}$

$R = \{ g_0\, l_0 \rightarrow g_0 \,,\; g_1\, l_1 \rightarrow g_0 \,,\; g_1\, l_1 \rightarrow g_1\, l_1\, l_0 \}$

$$
\begin{aligned}
S_0 &= D && = g_0\, l_0\, l_1^*\, l_0 \;+\; g_1\, l_1 \\
S_1 &= S_0 \cup pre(S_0) && = g_0\, (l_0 + l_0^2)\, l_1^*\, l_0 \;+ \\
& && \quad\; g_1\, l_1\, (\epsilon + l_0)\, l_1^*\, (\epsilon + l_0)
\end{aligned}
$$

Termination fails

$G = \{g_0, g_1\}$, $L = \{l_0, l_1\}$

$R = \{ \ g_0 \, l_0 \rightarrow g_0 \ , \ g_1 \, l_1 \rightarrow g_0 \ , \ g_1 \, l_1 \rightarrow g_1 \, l_1 \, l_0 \ \}$

$$
\begin{array}{rcl l}
S_0 & = & D & = \ g_0 \, l_0 \, l_1^* \, l_0 \ + \ g_1 \, l_1 \\
S_1 & = & S_0 \cup pre(S_0) & = \ g_0 \, (l_0 + l_0^2) \, l_1^* \, l_0 \, + \\
 & & & \quad g_1 \, l_1 \, (\epsilon + l_0) \, l_1^* \, (\epsilon + l_0) \\
 & \ldots & & \\
S_i & = & S_{i-1} \cup pre(S_{i-1}) & = \ g_0 \, (l_0 + \ldots + l_0^{i+1}) \, l_1^* \, l_0 \, + \\
 & & & \quad g_1 \, l_1 \, (\epsilon + l_0 + \ldots + l_0^i) \, l_1^* (\epsilon + l_0) \\
 & \ldots & &
\end{array}
$$

However, the fixpoint

$$
\begin{aligned}
\mathit{pre}^*(D) \;&=\; g_0\, l_0^+\, l_1^*\, l_0 \,+ \\
&\quad\; g_1\, l_1\, l_0^*\, l_1^*\, (\epsilon + l_0)
\end{aligned}
$$

is regular.

How can we compute it?

Accelerations

By definition, $pre(D) = \bigcup_{i \geq 0} S_i$
where $S_0 = D$ and $S_{i+1} = S_i \cup pre(S_i)$ for every $i \geq 0$

If convergence fails, try to compute an acceleration :
a sequence $T_0 \subseteq T_1 \subseteq T_2 \ldots$ such that

(a) $\forall i \geq 0 : S_i \subseteq T_i$

(b) $\forall i \geq 0 : T_i \subseteq \bigcup_{j \geq 0} S_j = pre(D)$

Property (a) ensures capture of (at least) the whole set $pre(D)$

Property (b) ensures that only elements of $pre(D)$ are captured

The acceleration guarantees termination if

(c) $\exists i \geq 0 : T_{i+1} = T_i$

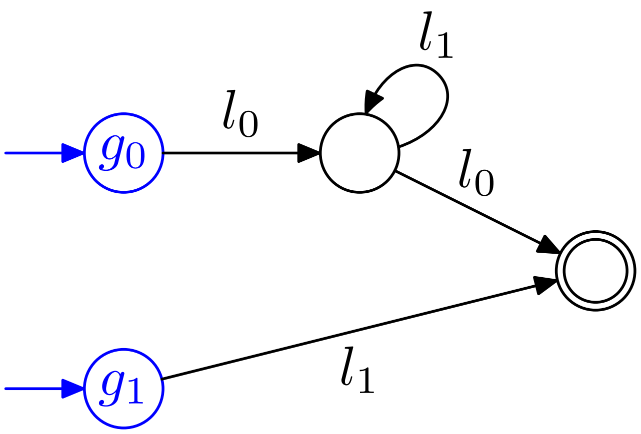
An acceleration for prefix rewriting

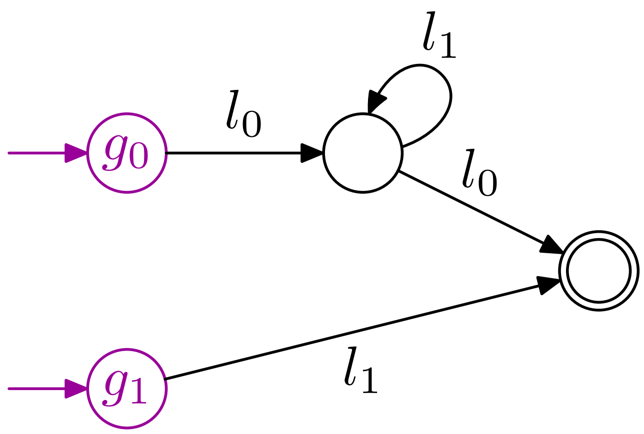
Idea: reuse the same states

An acceleration for prefix rewriting

Idea: reuse the same states

$$R = \{\ g_0\, l_0 \rightarrow g_0\ ,\ g_1\, l_1 \rightarrow g_0\ ,\ g_1\, l_1 \rightarrow g_1\, l_1\, l_0\ \}$$

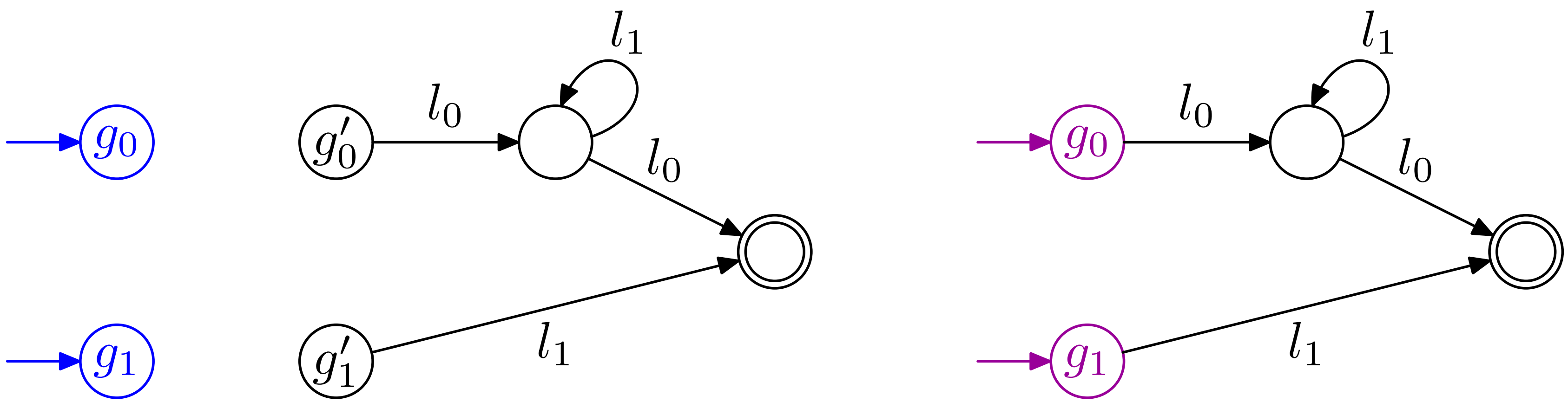




An acceleration for prefix rewriting

Idea: reuse the same states

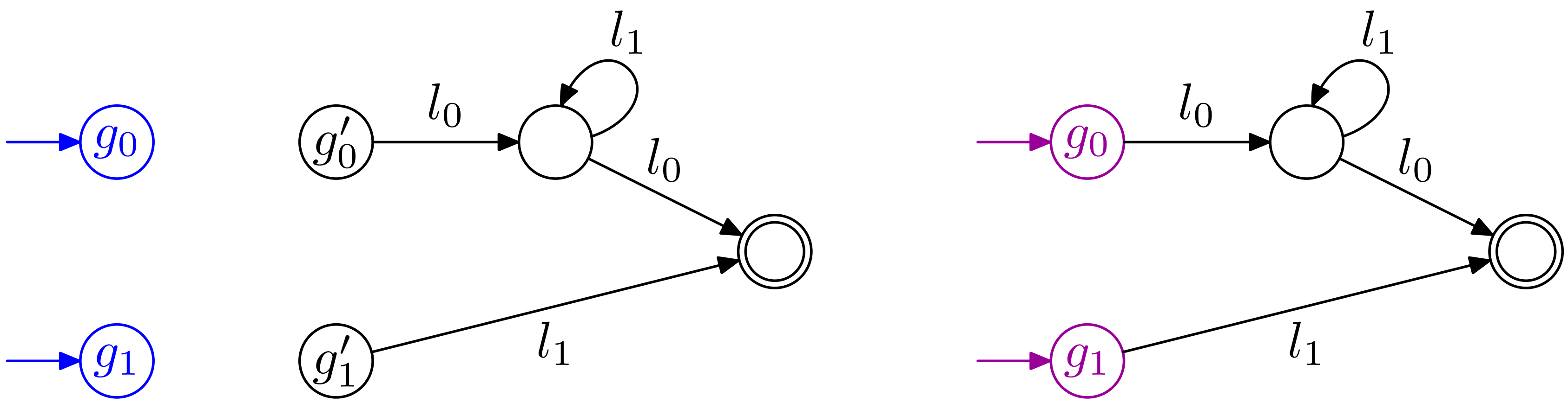
$$R = \{ g_0\, l_0 \to g_0 \, , \; g_1\, l_1 \to g_0 \, , \; g_1\, l_1 \to g_1\, l_1\, l_0 \}$$



An acceleration for prefix rewriting

Idea: reuse the same states

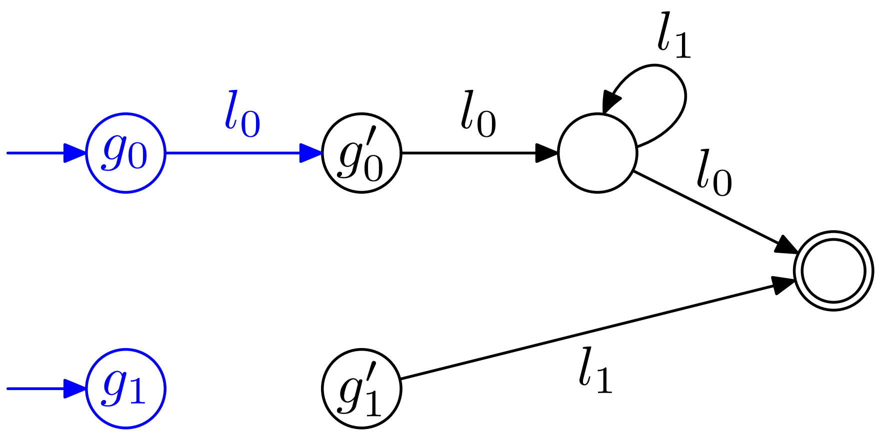
$$g_0\, l_0 \to g_0$$

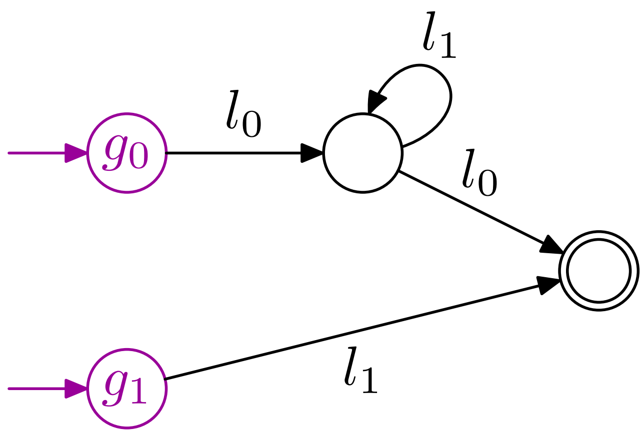


An acceleration for prefix rewriting

Idea: reuse the same states

$$g_0\, l_0 \rightarrow g_0$$

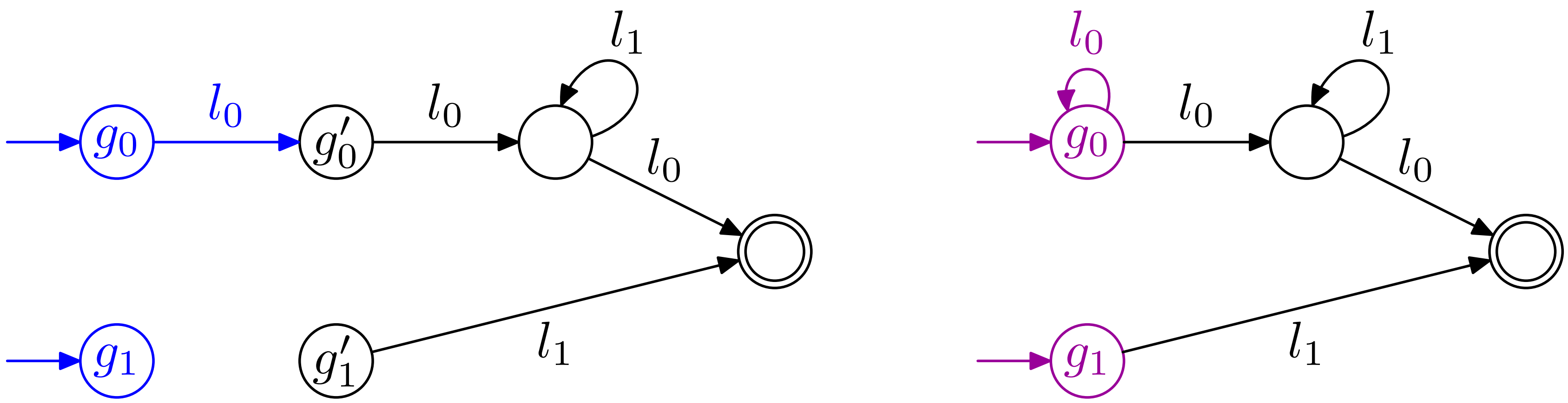




An acceleration for prefix rewriting

Idea: reuse the same states

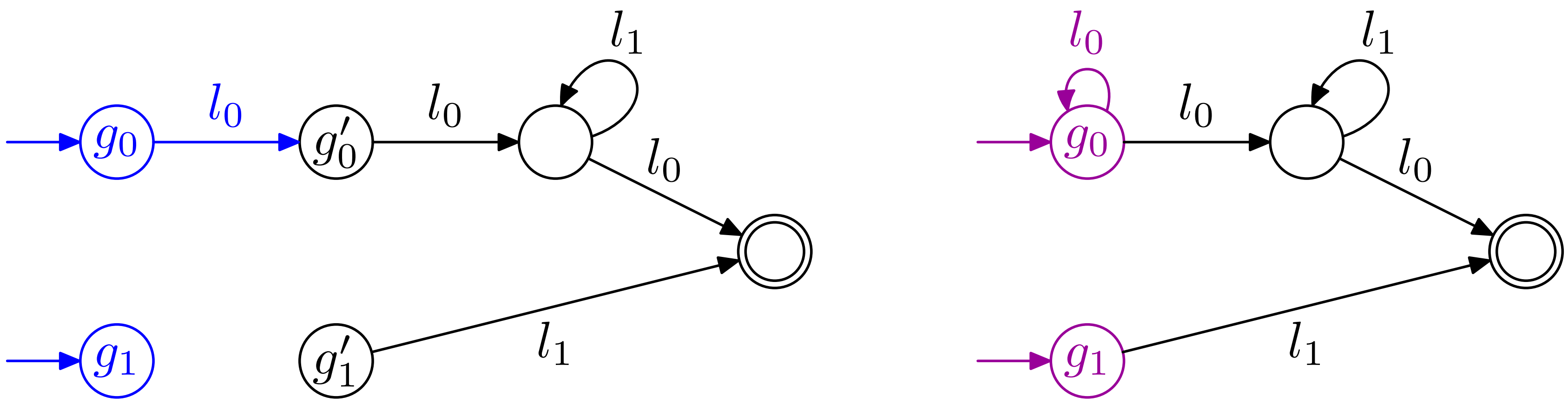
$$g_0\, l_0 \rightarrow g_0$$



An acceleration for prefix rewriting

Idea: reuse the same states

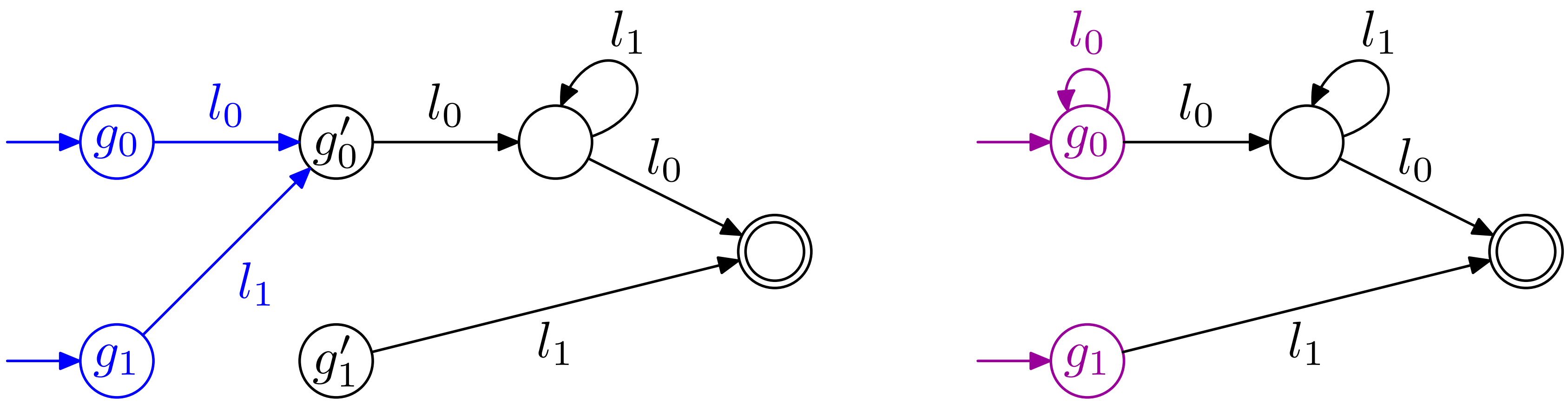
$$g_1\ l_1 \longrightarrow g_0$$



An acceleration for prefix rewriting

Idea: reuse the same states

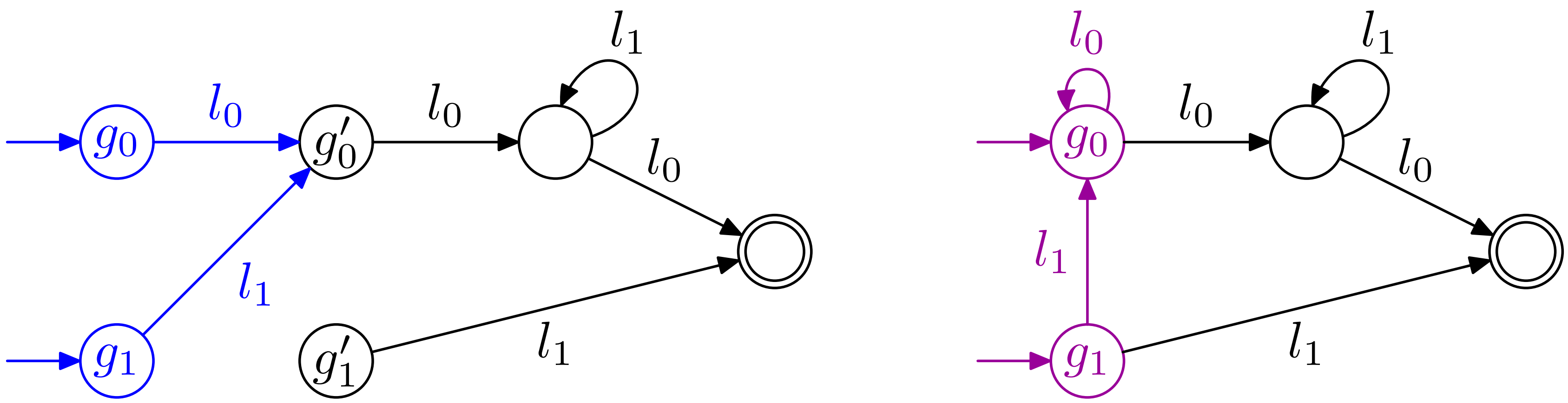
$$g_1\, l_1 \rightarrow g_0$$



An acceleration for prefix rewriting

Idea: reuse the same states

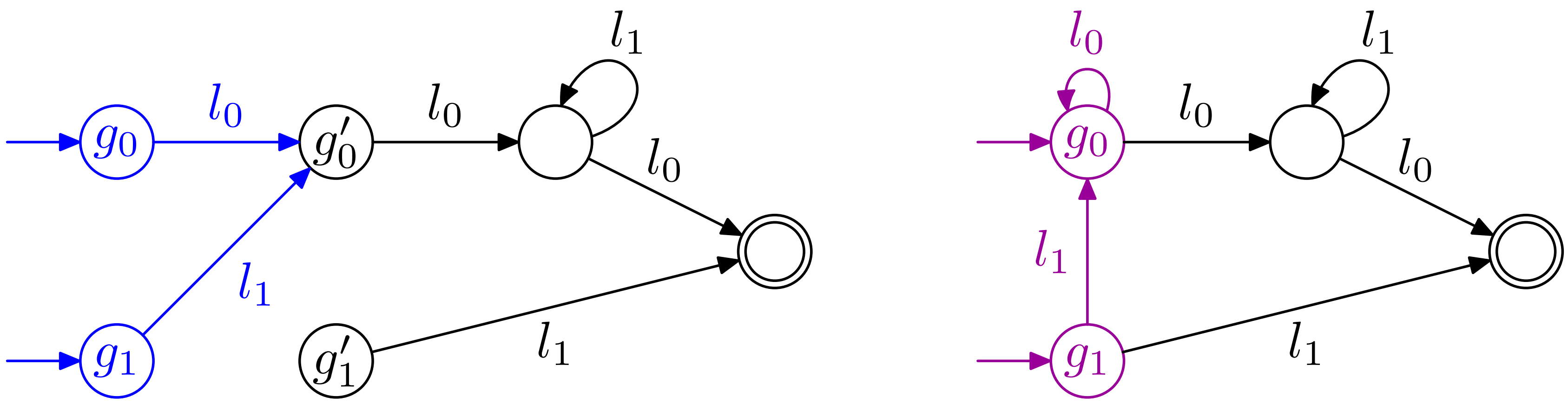
$$g_1\, l_1 \rightarrow g_0$$



An acceleration for prefix rewriting

Idea: reuse the same states

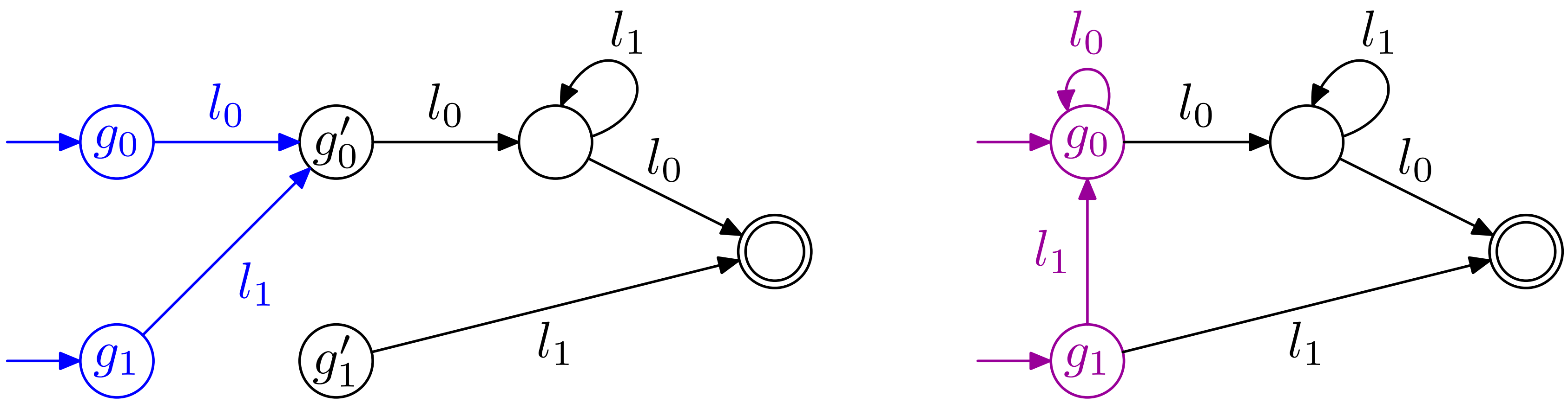
$g_1\, l_1 \longrightarrow g_1\, l_1\, l_0$



An acceleration for prefix rewriting

Idea: reuse the same states

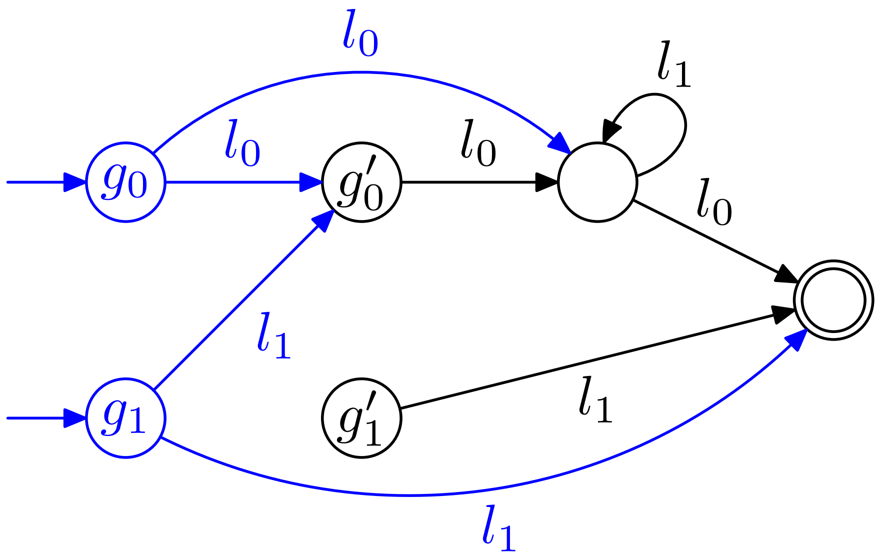
$$R = \{ g_0\, l_0 \rightarrow g_0 \;,\; g_1\, l_1 \rightarrow g_0 \;,\; g_1\, l_1 \rightarrow g_1\, l_1\, l_0 \}$$

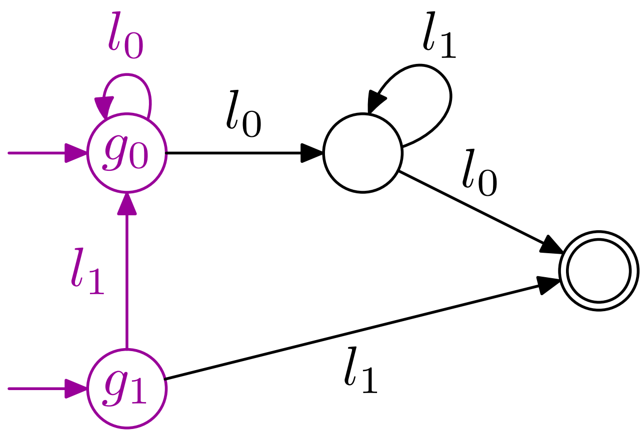


An acceleration for prefix rewriting

Idea: reuse the same states

$$R = \{ g_0\, l_0 \to g_0 \,,\; g_1\, l_1 \to g_0 \,,\; g_1\, l_1 \to g_1\, l_1\, l_0 \}$$

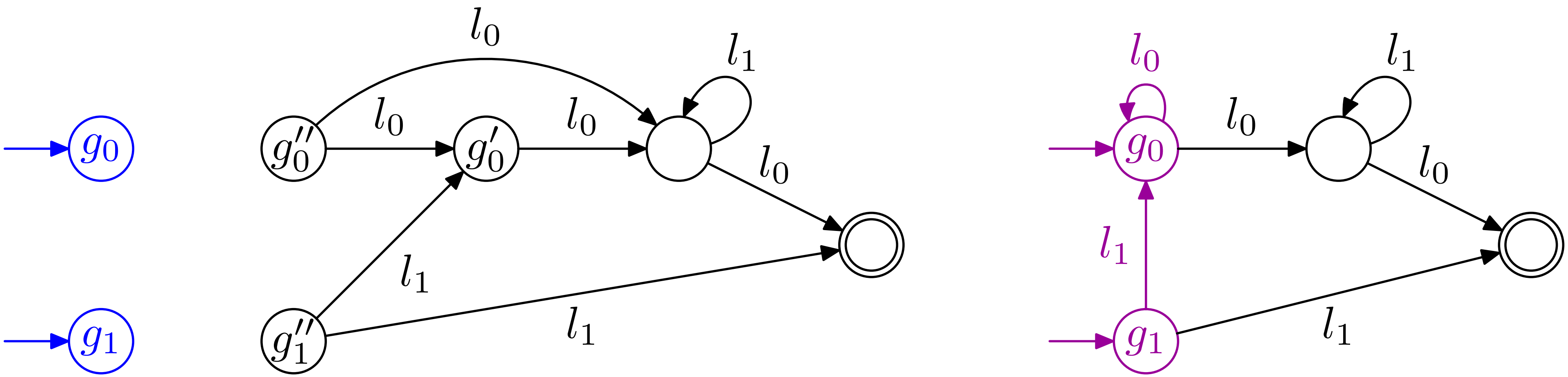




An acceleration for prefix rewriting

Idea: reuse the same states

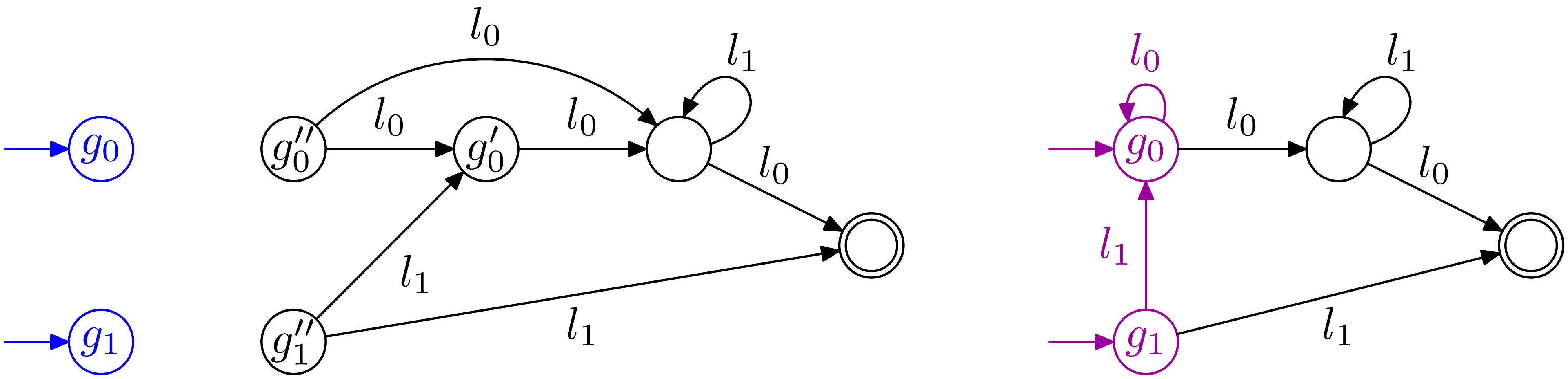
$$R = \{\ g_0\, l_0 \rightarrow g_0\ ,\ g_1\, l_1 \rightarrow g_0\ ,\ g_1\, l_1 \rightarrow g_1\, l_1\, l_0\ \}$$



An acceleration for prefix rewriting

Idea: reuse the same states

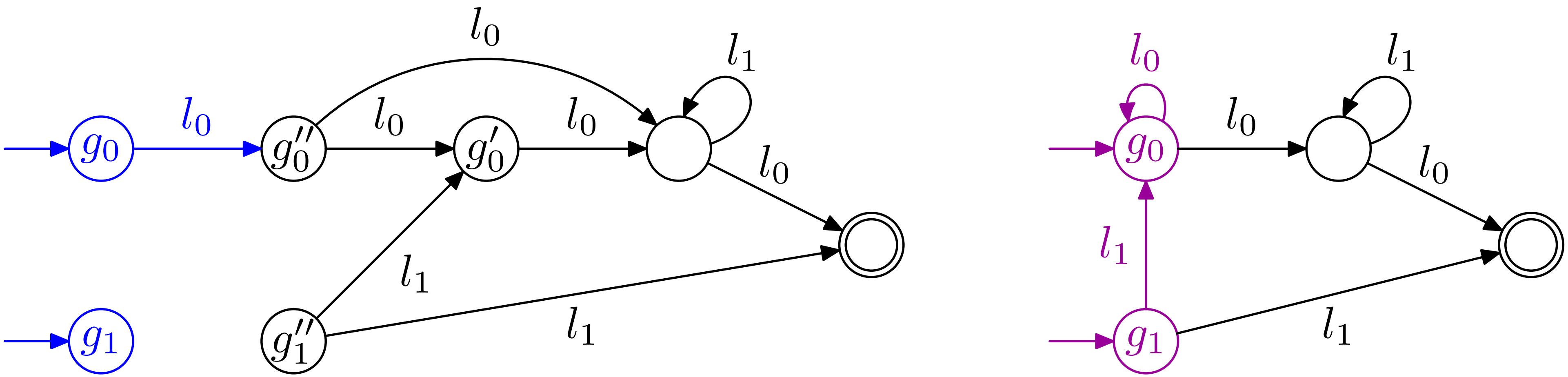
$g_0\, l_0 \rightarrow g_0$



An acceleration for prefix rewriting

Idea: reuse the same states

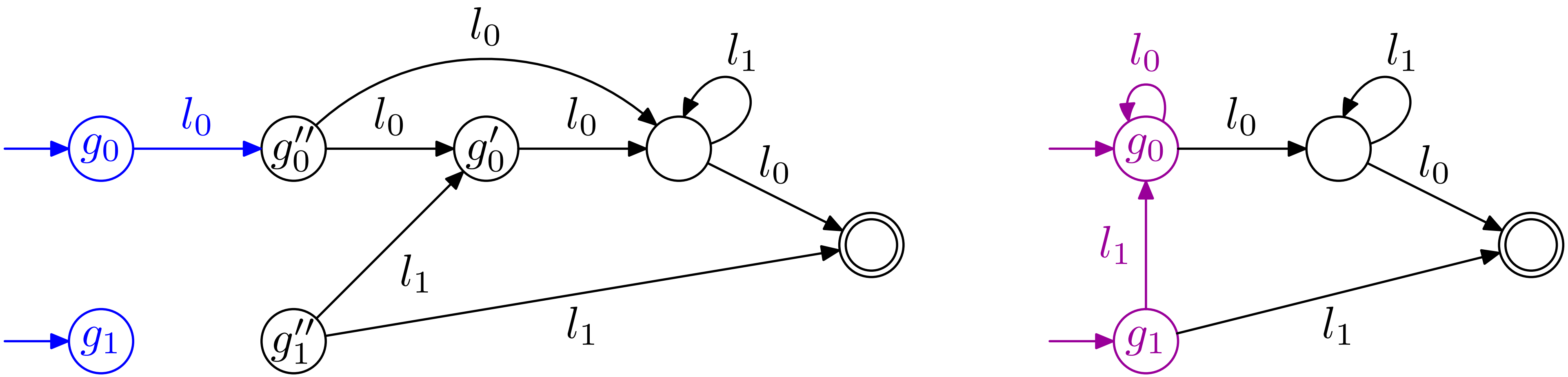
$$g_0\, l_0 \rightarrow g_0$$



An acceleration for prefix rewriting

Idea: reuse the same states

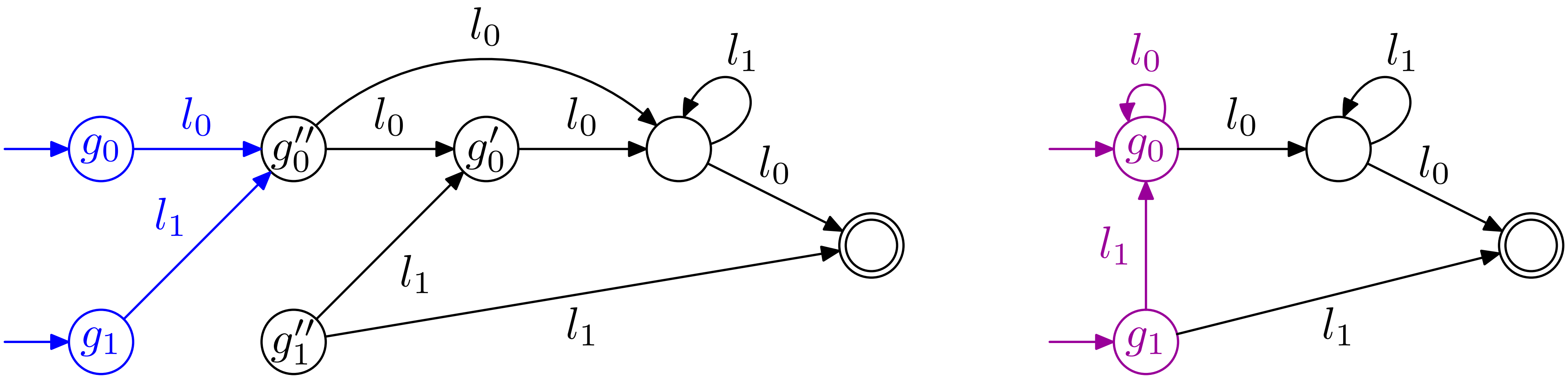
$$g_1\, l_1 \longrightarrow g_0$$



An acceleration for prefix rewriting

Idea: reuse the same states

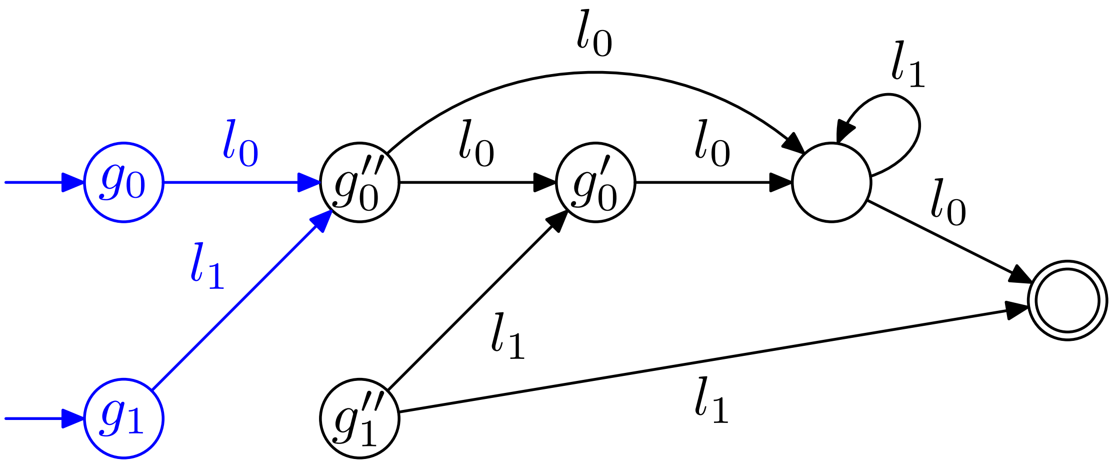
$$g_1\, l_1 \longrightarrow g_0$$

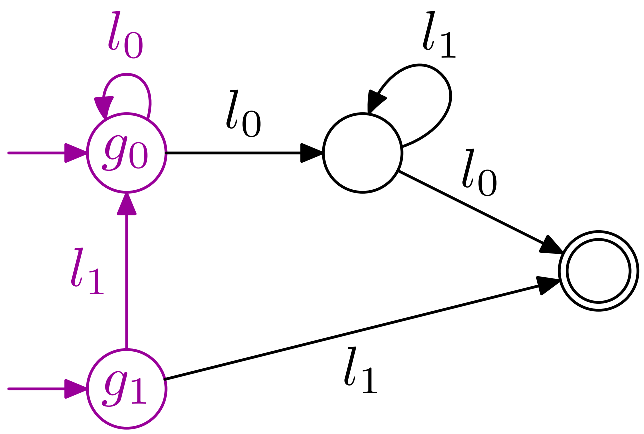


An acceleration for prefix rewriting

Idea: reuse the same states

$g_1\, l_1 \rightarrow g_1\, l_1\, l_0$

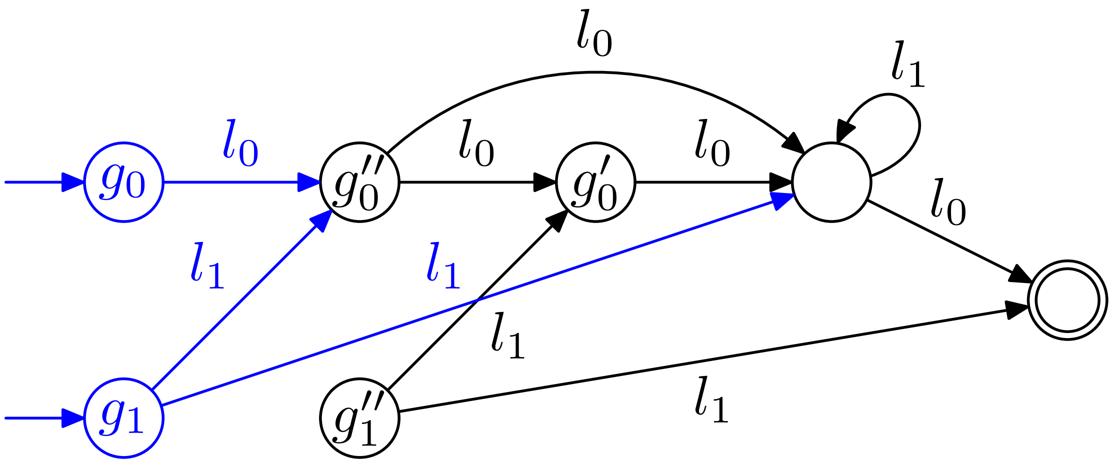


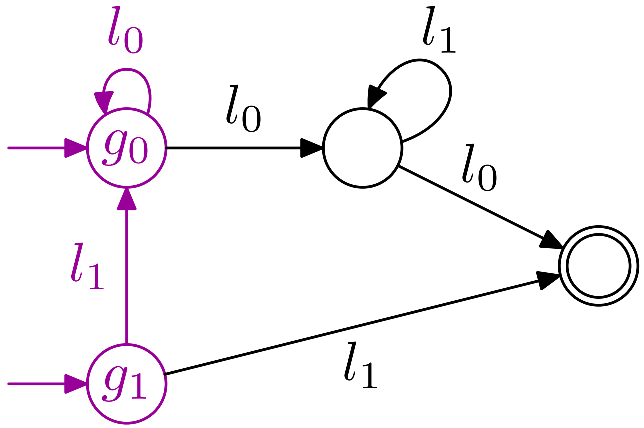


An acceleration for prefix rewriting

Idea: reuse the same states

$g_1\, l_1 \rightarrow g_1\, l_1\, l_0$

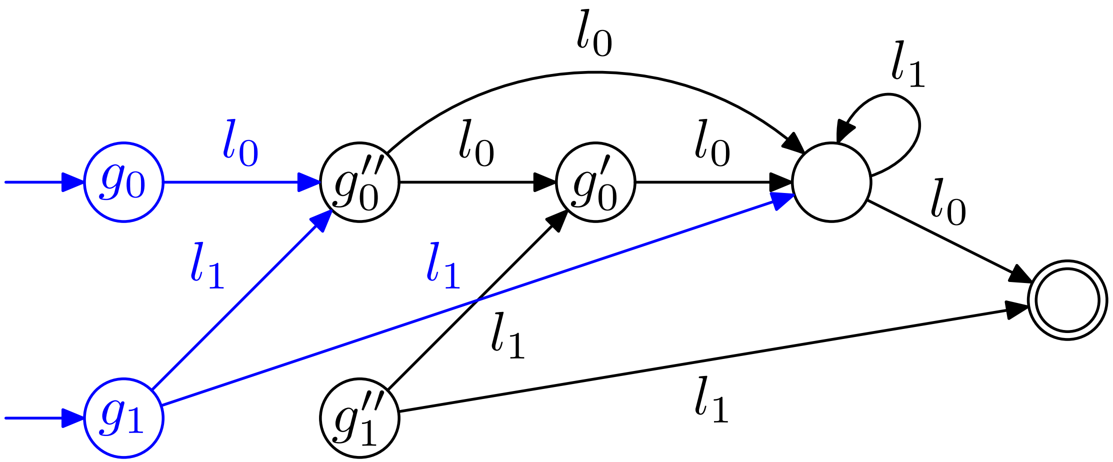


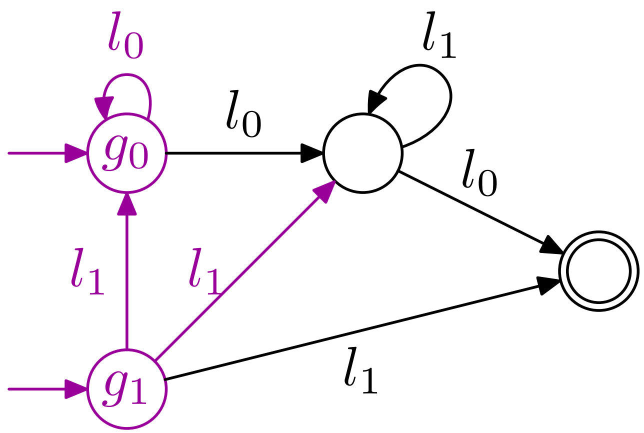


An acceleration for prefix rewriting

Idea: reuse the same states

$g_1\ l_1 \rightarrow g_1\ l_1\ l_0$

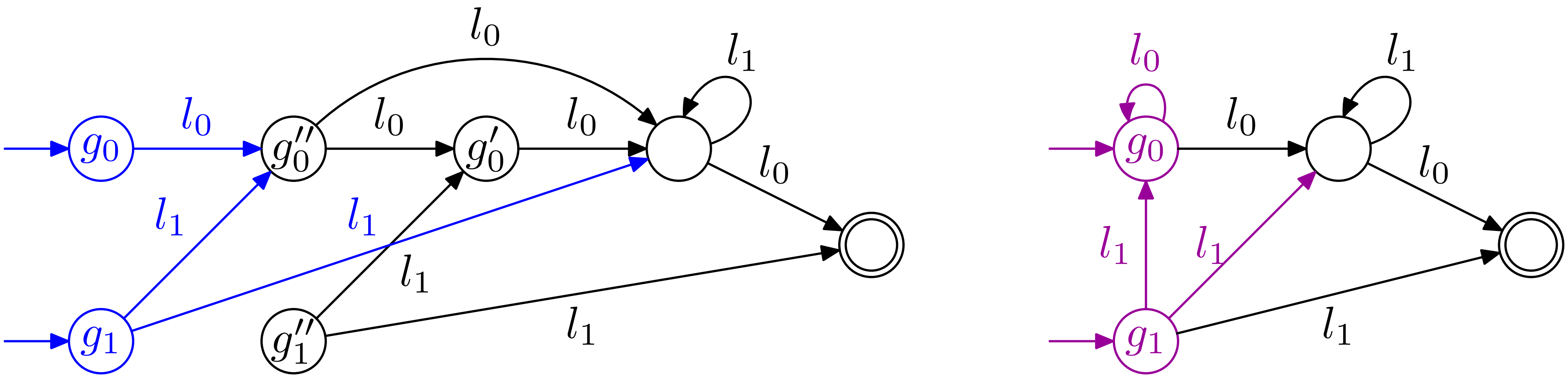




An acceleration for prefix rewriting

Idea: reuse the same states

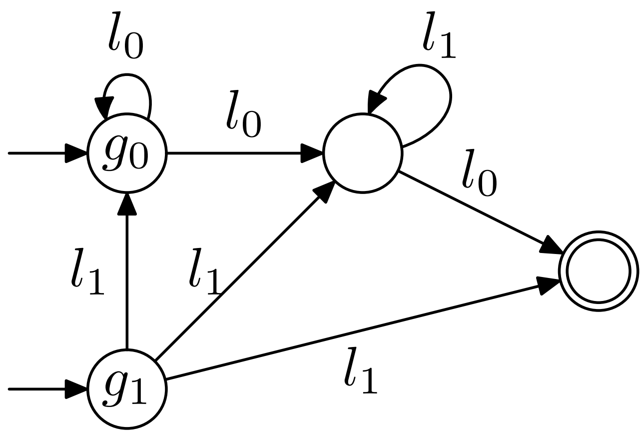
$$R = \{ g_0\, l_0 \rightarrow g_0 \ , \ g_1\, l_1 \rightarrow g_0 \ , \ g_1\, l_1 \rightarrow g_1\, l_1\, l_0 \}$$



An acceleration for prefix rewriting

Idea: reuse the same states

$$R = \{\, g_0\, l_0 \rightarrow g_0 \;,\; g_1\, l_1 \rightarrow g_0 \;,\; g_1\, l_1 \rightarrow g_1\, l_1\, l_0 \,\}$$



But does it work . . . ?

All predecessors are computed, and termination guaranteed

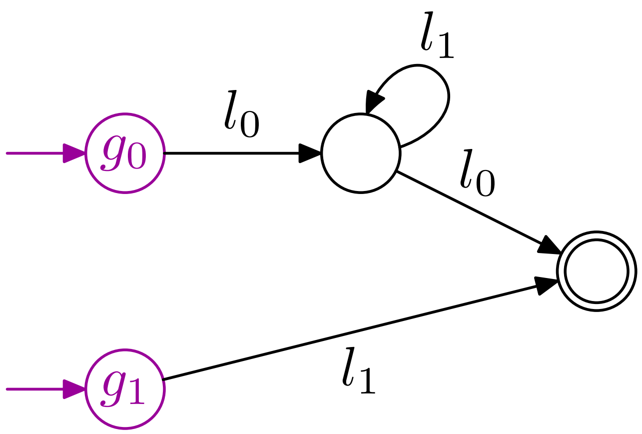
But: we might be adding non-predecessors

But does it work . . . ?

All predecessors are computed, and termination guaranteed

But: we might be adding non-predecessors

$$R = \{ g_0\, l_0 \rightarrow g_0 \,,\; g_1\, l_1 \rightarrow g_0 \,,\; g_1\, l_1 \rightarrow g_1\, l_1\, l_0 \}$$

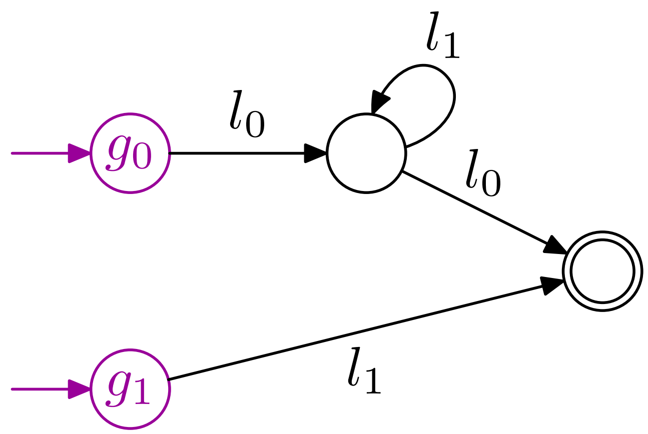


But does it work . . . ?

All predecessors are computed, and termination guaranteed

But: we might be adding non-predecessors

$$g_0\, l_0 \longrightarrow g_0$$

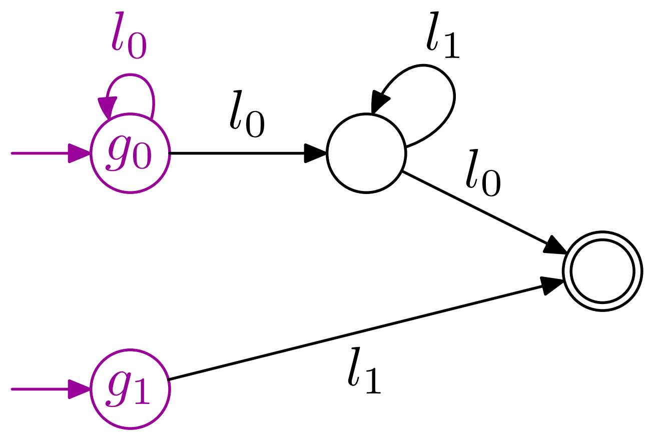


But does it work . . . ?

All predecessors are computed, and termination guaranteed

But: we might be adding non-predecessors

$$g_0\, l_0 \rightarrow g_0$$

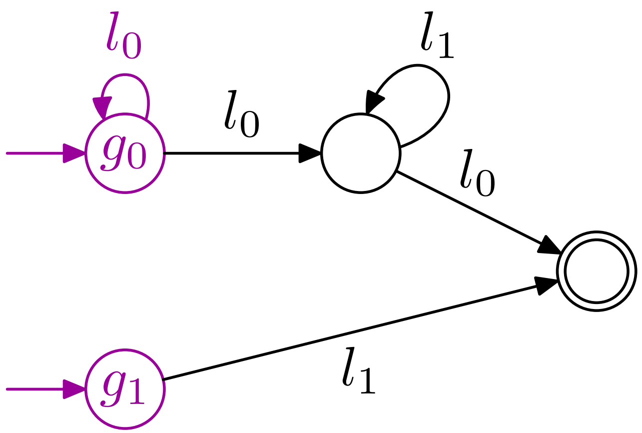


But does it work . . . ?

All predecessors are computed, and termination guaranteed

But: we might be adding non-predecessors

$$R = \{ g_0 \, l_0 \rightarrow g_0 \, , \; g_1 \, l_1 \rightarrow g_0 \, , \; g_1 \, l_1 \rightarrow g_1 \, l_1 \, l_0 \, \}$$



Fortunately: correct if initial states have no incoming arcs.

Forward search and complexity

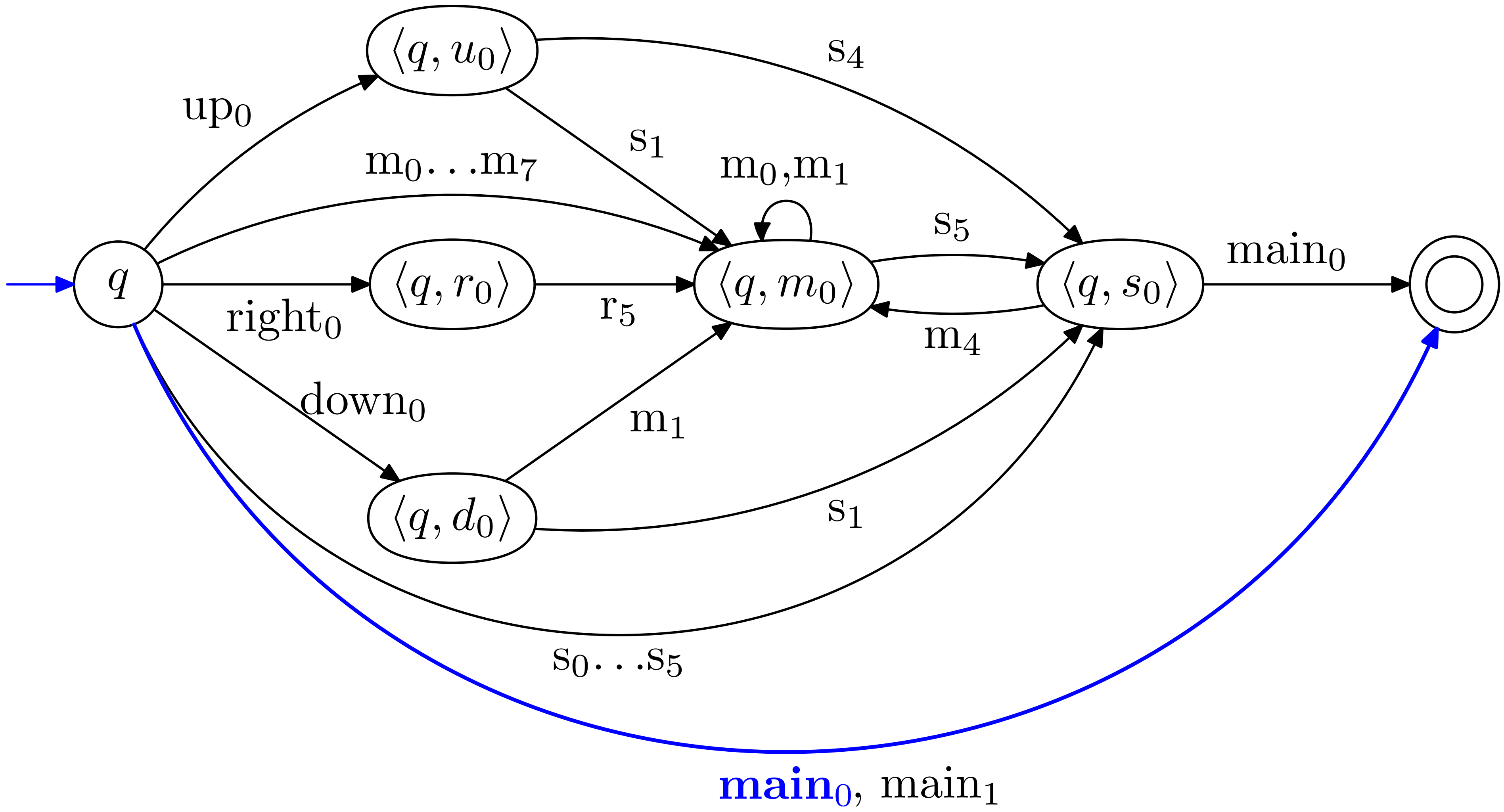
Symbolic forward search with regular sets can be accelerated in a similar way

Recall input: Alphabet $\Sigma = G \cup L$, set R of rules, NFA $\mathcal{A} = (Q, L, \rightarrow_0, G, F)$ recognizing subset of $G\,L^*$.

Complexity of backward search: $O(|Q|^2 \cdot |R|)$ time, $O(|Q| \cdot |R| + |\rightarrow_0|)$ space.

Complexity of forward search: $O(|G| \cdot |R| \cdot (|Q \setminus G| + |R|) + |G| \cdot |\rightarrow_0|)$ time and space.

Reachable configurations of the plotter program



Repeated reachability for prefix rewriting

Let $I = g_0\, l_0$ and $D = g\, L^*$.

D can be repeatedly reached from I iff

$$g_0\, l_0 \longrightarrow^* g'\, l\, w$$

and

$$g'\, l \longrightarrow^* g\, v \longrightarrow^* g'\, l\, u$$

for some g', l, w, v, u.

Repeated reachability can be reduced to computing several pre^*.