A language no fsm accepts

$$\{\epsilon, ab, aabb, aaabbb, \ldots\}$$

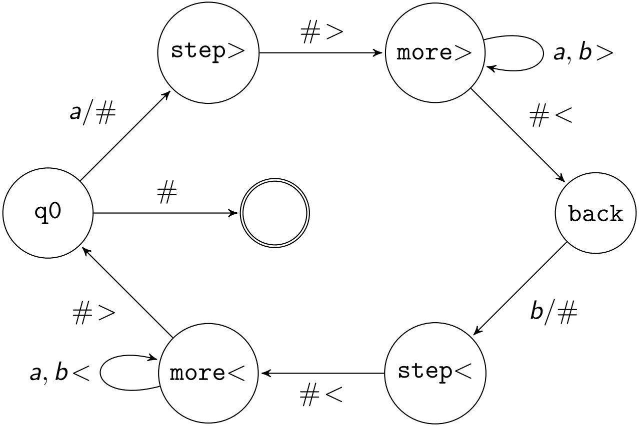
context-free grammar $\quad S \longrightarrow \epsilon \mid aSb$

A language no fsm accepts

$$\{\epsilon, ab, aabb, aaabbb, \ldots\}$$

context-free grammar $\quad S \longrightarrow \epsilon \mid aSb$

Add blank symbol # and actions write, move left

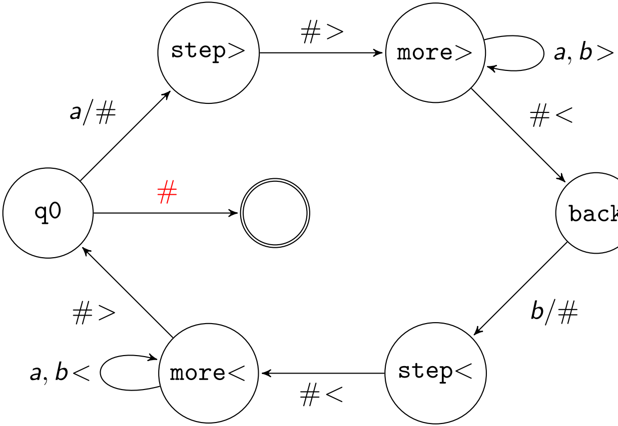


A language no fsm accepts

$$\{\epsilon, ab, aabb, aaabbb, \ldots\}$$

context-free grammar $S \longrightarrow \epsilon \mid aSb$

Add blank symbol # and actions write, move left

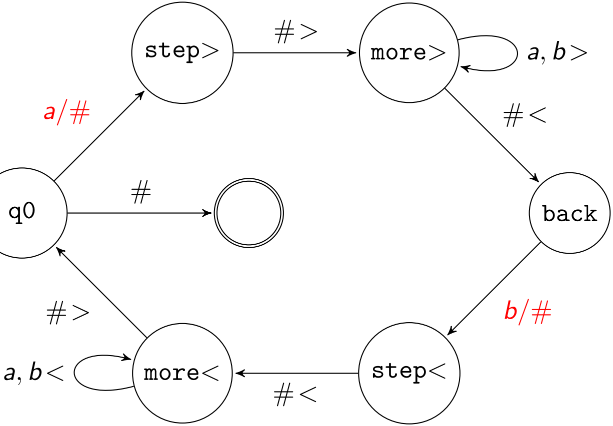


A language no fsm accepts

$$\{\epsilon, ab, aabb, aaabbb, \ldots\}$$

context-free grammar $S \longrightarrow \epsilon \mid aSb$

Add blank symbol # and actions write, move left

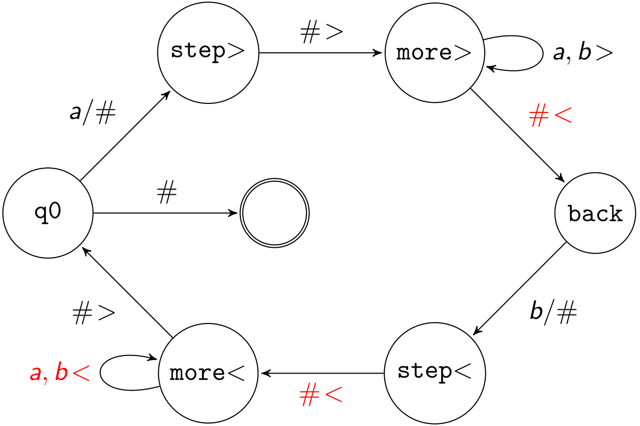


A language no fsm accepts

$$\{\epsilon, ab, aabb, aaabbb, \ldots\}$$

context-free grammar $\quad S \longrightarrow \epsilon \mid aSb$

Add blank symbol $\#$ and actions write, move left



Tm = fsm + MLeft/Write/Halt

A **Turing machine** (Tm) M is a 5-tuple

```
[MRight, MLeft, Write, Halt, Q0]
```

where

- MRight is a list of triples [Q,X,Qn] such that at state Q and seeing symbol X, M may move right, and change state to Qn
- Q0 is M's initial state

Tm = fsm + MLeft/Write/Halt

A **Turing machine** (Tm) M is a 5-tuple

```
[MRight, MLeft, Write, Halt, Q0]
```

where

- MRight is a list of triples [Q,X,Qn] such that at state Q and seeing symbol X, M may move right, and change state to Qn
- Q0 is M's initial state
- MLeft is a list of triples [Q,X,Qn] such that at Q and seeing X, M may move left, and change state to Qn

Tm = fsm + `MLeft/Write/Halt`

A **Turing machine** (Tm) `M` is a 5-tuple

```
[MRight, MLeft, Write, Halt, Q0]
```

where

- `MRight` is a list of triples `[Q,X,Qn]` such that at state `Q` and seeing symbol `X`, `M` may move right, and change state to `Qn`
- `Q0` is `M`'s initial state
- `MLeft` is a list of triples `[Q,X,Qn]` such that at `Q` and seeing `X`, `M` may move left, and change state to `Qn`
- `Write` is a list of 4-tuples `[Q,X,Y,Qn]` such that at `Q` and `X`, `M` may write `Y`, and change state to `Qn` without moving

Tm = fsm + MLeft/Write/Halt

A **Turing machine** (Tm) M is a 5-tuple

[MRight, MLeft, Write, Halt, Q0]

where

- MRight is a list of triples [Q,X,Qn] such that at state Q and seeing symbol X, M may move right, and change state to Qn
- Q0 is M's initial state
- MLeft is a list of triples [Q,X,Qn] such that at Q and seeing X, M may move left, and change state to Qn
- Write is a list of 4-tuples [Q,X,Y,Qn] such that at Q and X, M may write Y, and change state to Qn without moving
- Halt is a list of pairs [Q,X] such that at Q and X, M may halt.

Tm = fsm + MLeft/Write/Halt

A **Turing machine** (Tm) `M` is a 5-tuple

```
[MRight, MLeft, Write, Halt, Q0]
```

where

- `MRight` is a list of triples `[Q,X,Qn]` such that at state `Q` and seeing symbol `X`, `M` may move right, and change state to `Qn`
- `Q0` is `M`'s initial state
- `MLeft` is a list of triples `[Q,X,Qn]` such that at `Q` and seeing `X`, `M` may move left, and change state to `Qn`
- `Write` is a list of 4-tuples `[Q,X,Y,Qn]` such that at `Q` and `X`, `M` may write `Y`, and change state to `Qn` without moving
- `Halt` is a list of pairs `[Q,X]` such that at `Q` and `X`, `M` may halt.

N.B. A fsm is a Tm where `MLeft = [] = Write`, and for every pair `[Q,X]` in `Halt`, the symbol `X` is # (blank).