Software Verification
Exercise class:
Model Checking

Carlo A. Furia

Recap of definitions and results

Finite State Automata: Syntax

Def. Nondeterministic Finite State Automaton (FSA):

a tuple $[\Sigma, S, I, \rho, F]$:

- Σ: finite nonempty (input) alphabet
- S: finite nonempty set of states
- $I \subseteq S$: set of initial states
- $F \subseteq S$: set of accepting states
- $\rho: S \times \Sigma \rightarrow 2^S$: transition function

Finite State Automata: Semantics

Def. An accepting run of an FSA $A=[\Sigma, S, I, \rho, F]$
over input word $w = w(1)\, w(2) \ldots w(n) \in \Sigma^*$
is a sequence $r = r(0)\, r(1)\, r(2) \ldots r(n) \in S^*$
of states such that:

- it starts from an initial state: $r(0) \in I$
- it ends in an accepting state: $r(n) \in F$
- it respects the transition function:
  $r(i+1) \in \rho(r(i), w(i))$ for all $0 \leq i < n$

Finite State Automata: Semantics

Def. Any FSA $A=[\Sigma, S, I, \rho, F]$ defines
a set of input words $\langle A \rangle$:
$\langle A \rangle \triangleq \{ w \in \Sigma^* \mid$ there is an accepting run of A over w $\}$

$\langle A \rangle$ is called the language of A

Linear Temporal Logic: Syntax

Def. Propositional Linear Temporal Logic (LTL) formulae are defined by the grammar:

$$F ::= p \mid \neg F \mid F \wedge G \mid X F \mid F \, U \, G$$

with $p \in P$ any atomic proposition from a fixed set P.

Temporal (modal) operators:

- next: $X F$
- until: $F \, U \, G$
- release: $F \, R \, G \triangleq \neg(\neg F \, U \, \neg G)$
- eventually: $\Diamond F \triangleq True \, U \, F$
- always: $\Box F \triangleq \neg \Diamond \neg F$

Propositional connectives:

- not: $\neg F$
- and: $F \wedge G$
- or: $F \vee G \triangleq \neg(\neg F \wedge \neg G)$
- implies: $F \Rightarrow G \triangleq \neg F \vee G$
- iff: $F \Leftrightarrow G \triangleq (F \Rightarrow G) \wedge (G \Rightarrow F)$

Linear Temporal Logic: Semantics

Def. A word $w = w(1)\ w(2) \ldots w(n) \in P^*$

satisfies an LTL formula F at position $1 \leq i \leq n$,
denoted $w, i \models F$, under the following conditions:

- $w, i \models p$ iff $p = w(i)$
- $w, i \models \neg F$ iff $w, i \models F$ does not hold
- $w, i \models F \wedge G$ iff both $w, i \models F$ and $w, i \models G$ hold
- $w, i \models X\ F$ iff $i < n$ and $w, i+1 \models F$
  - i.e., F holds in the next step
- $w, i \models F\ U\ G$ iff for some $i \leq j \leq n$ it is: $w, j \models G$ and for all $i \leq k < j$ it is $w, k \models F$
  - i.e., F holds until G will hold

Linear Temporal Logic: Semantics

For derived operators:

- $w, i \models \Diamond F$ iff for some $i \le j \le n$ it is: $w, j \models F$
  - i.e., F holds eventually (in the future)
- $w, i \models \Box F$ iff for all $i \le j \le n$ it is: $w, j \models F$
  - i.e., F holds always (in the future)

Linear Temporal Logic: Semantics

Def. Satisfaction:

$$w \models F \quad \triangleq \quad w, 1 \models F$$

i.e., word w satisfies formula F initially

Def. Any LTL formula F defines a set of words ⟨F⟩:

$$\langle F \rangle \triangleq \{ w \in P^* \mid \quad w \models F \}$$

⟨F⟩ is called the language of F

Automata-theoretic Model Checking

An semantic view of the Model Checking problem:

– Given: a finite-state automaton *A*
and a temporal-logic formula F

– if ⟨*A*⟩ ∩ ⟨¬ F⟩ is empty then any run of *A* satisfies F

– if ⟨*A*⟩ ∩ ⟨¬ F⟩ is not empty then some run of *A* does not satisfy F

- any member of the nonempty intersection ⟨*A*⟩ ∩ ⟨¬ F⟩ is a counterexample

Automata-theoretic Model Checking

How to check $\langle A \rangle \cap \langle \neg F \rangle = \emptyset$ algorithmically (given A, F)?

Combination of three different algorithms:

- LTL2FSA: given LTL formula F build automaton $a(F)$ such that $\langle F \rangle = \langle a(F) \rangle$
- FSA-Intersection: given automata A, B build automaton C such that $\langle A \rangle \cap \langle B \rangle = \langle C \rangle$
- FSA-Emptiness: given automaton A check whether $\langle A \rangle = \emptyset$ is the case

Exercises:
Semantics of derived operators

LTL derived operators: eventually

Prove that the satisfaction relation

$$w, i \models \Diamond F$$

for eventually, defined as:

$$\Diamond F \triangleq \text{True } U \ F$$

is equivalent to:

for some $i \le j \le n$ it is: $w, j \models F$

LTL derived operators: eventually

$w, i \models \Diamond F$

iff

$w, i \models \text{True } U \; F$ (definition of eventually)

iff

for some $i \leq j \leq n$ it is: $w, j \models F$

and for all $i \leq k < j$ it is $w, k \models \text{True}$

(definition of until)

iff

for some $i \leq j \leq n$ it is: $w, j \models F$

(simplification of *A* and True)

LTL derived operators: always

Prove that the satisfaction relation

$$w, i \models \square F$$

for always, defined as:

$$\square F \triangleq \neg \lozenge \neg F$$

is equivalent to:

for all $i \leq j \leq n$ it is: $w, j \models F$

LTL derived operators: always

$w, i \models \square F$

iff

$w, i \models \neg \Diamond \neg F$ (definition of always)

iff

$w, i \models \Diamond \neg F$ is not the case (definition of not)

iff

it is not the case that: for some $i \leq j \leq n$ it is: $w, j \models \neg F$

(semantics of eventually)

iff

for all $i \leq j \leq n$ it is not the case that $w, j \models \neg F$

(semantics of quantifiers: pushing negation inward)

iff

for all $i \leq j \leq n$: it is not the case that it is not the case that $w, j \models F$

(semantics of negation)

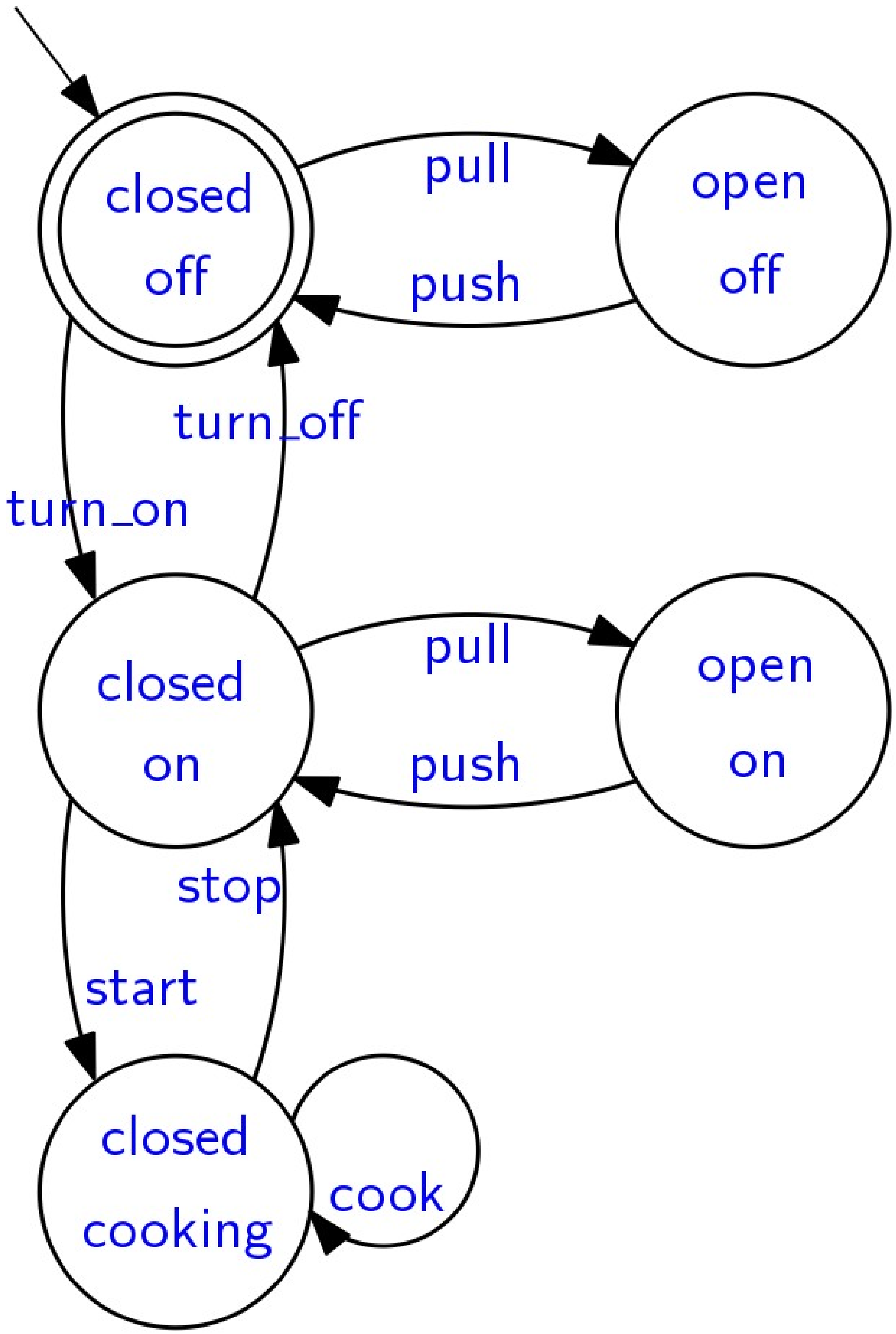
iff

for all $i \leq j \leq n$ it is: $w, j \models F$

(simplification of double negation)

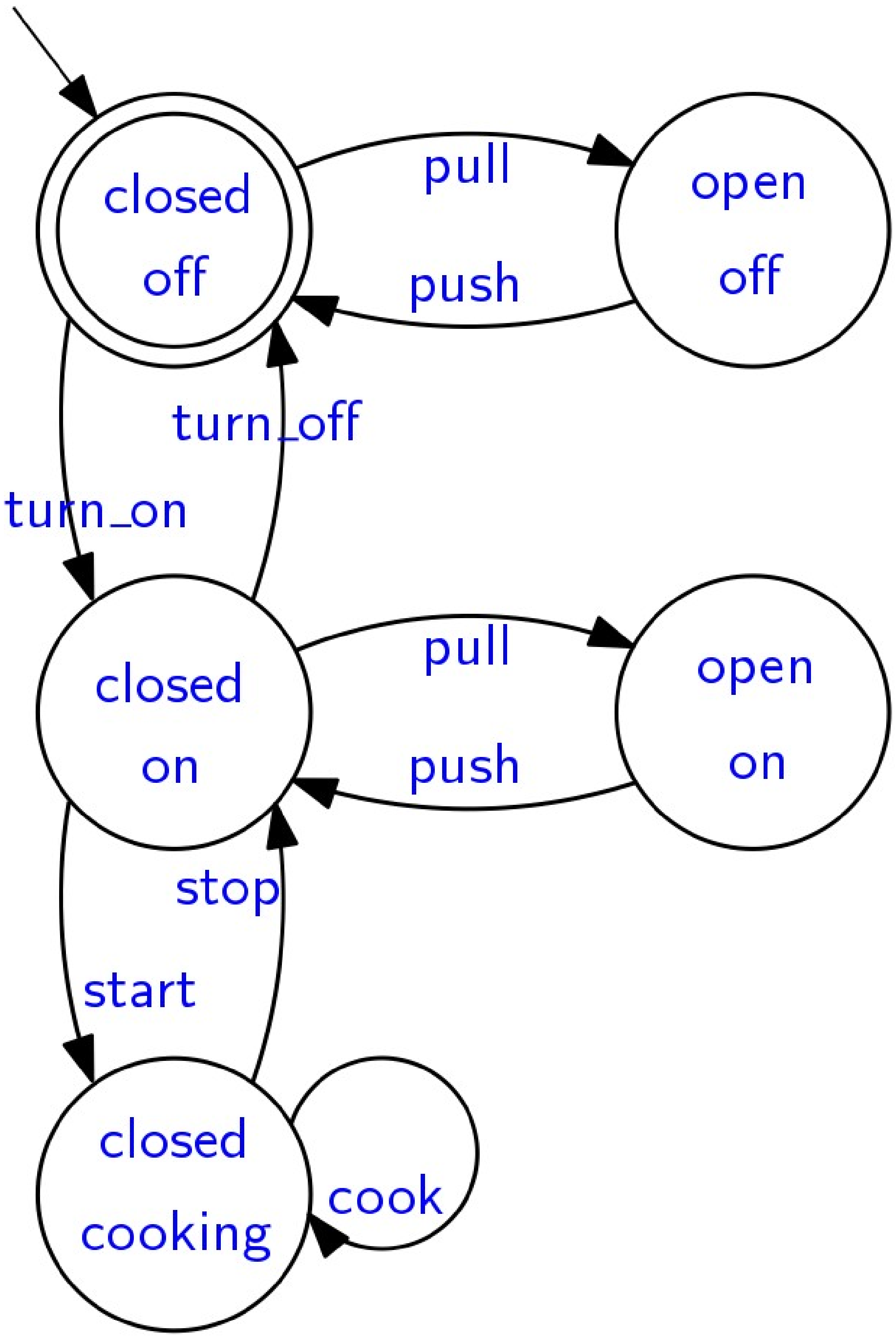
Exercises:
Evaluate LTL formulas on automata

Does the property hold?



$\square\,(start \Rightarrow \Diamond\, stop)$

Does the property hold?

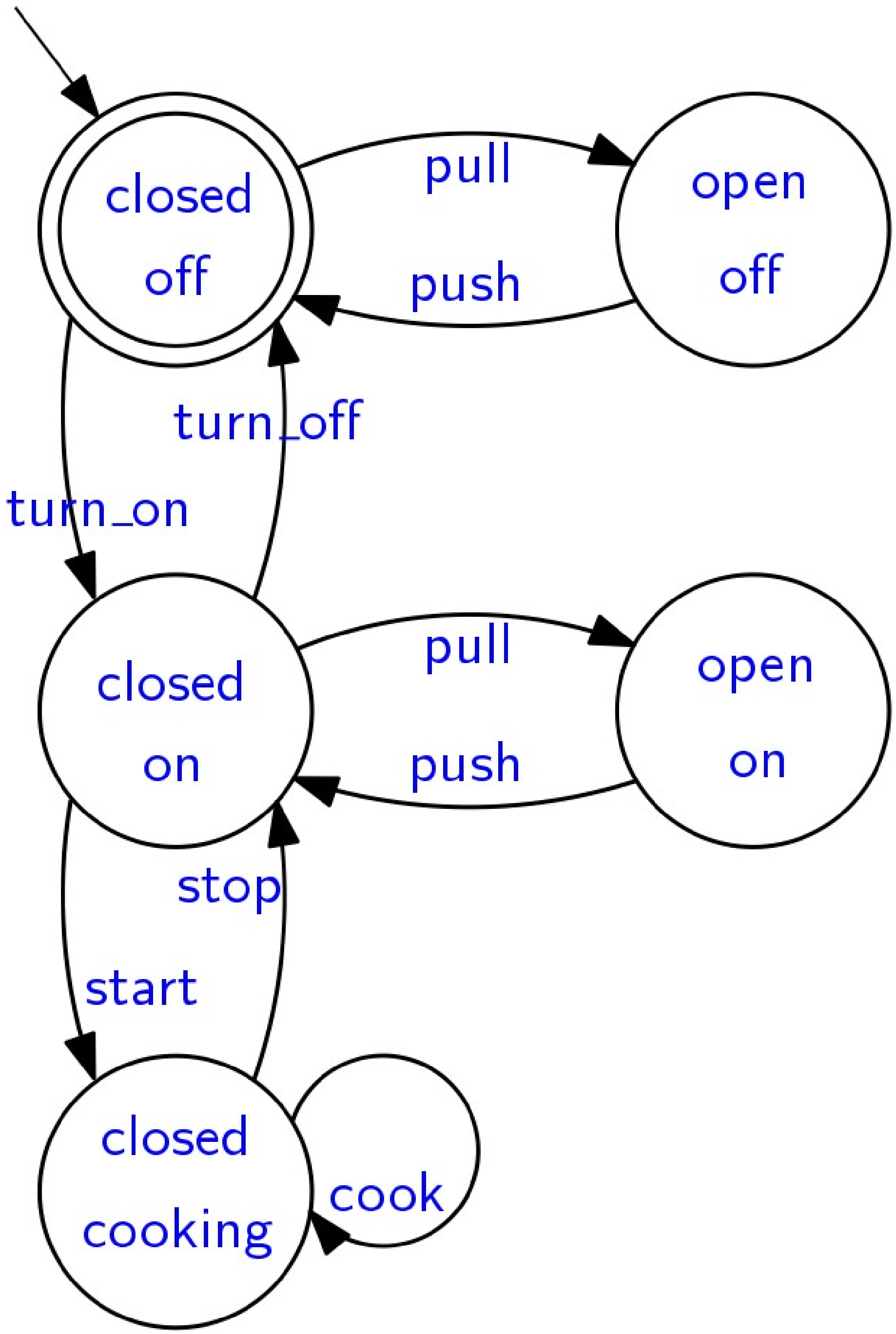


$\Box\,(\text{start} \Rightarrow \Diamond\,\text{stop})$

Yes:

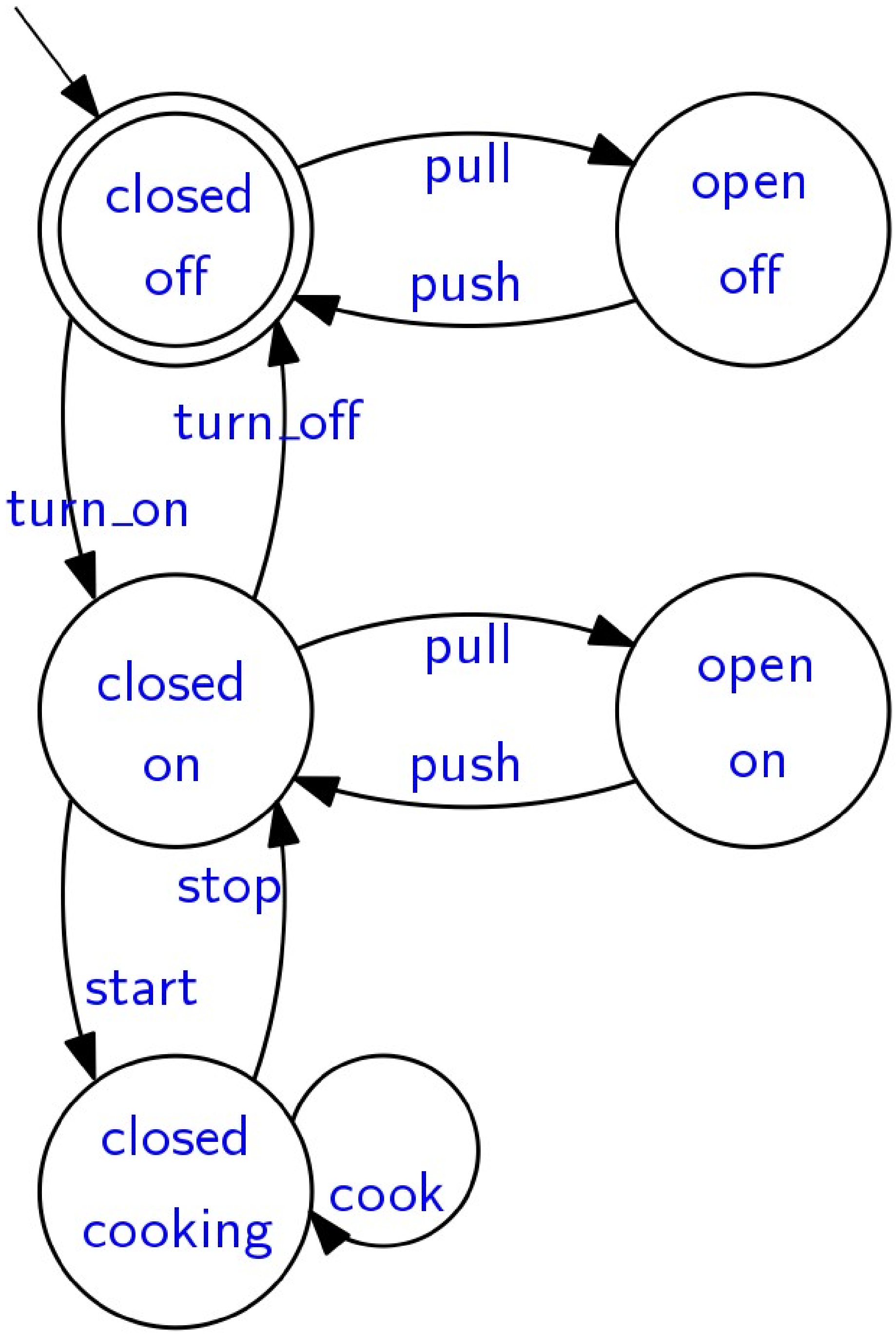
- whenever start occurs we reach state closed-cooking
- we must eventually exit state closed-cooking to reach the only accepting state closed-off
- state closed-cooking can be exited only if stop occurs

Does the property hold?



$\Box \Diamond$ turn_off

Does the property hold?

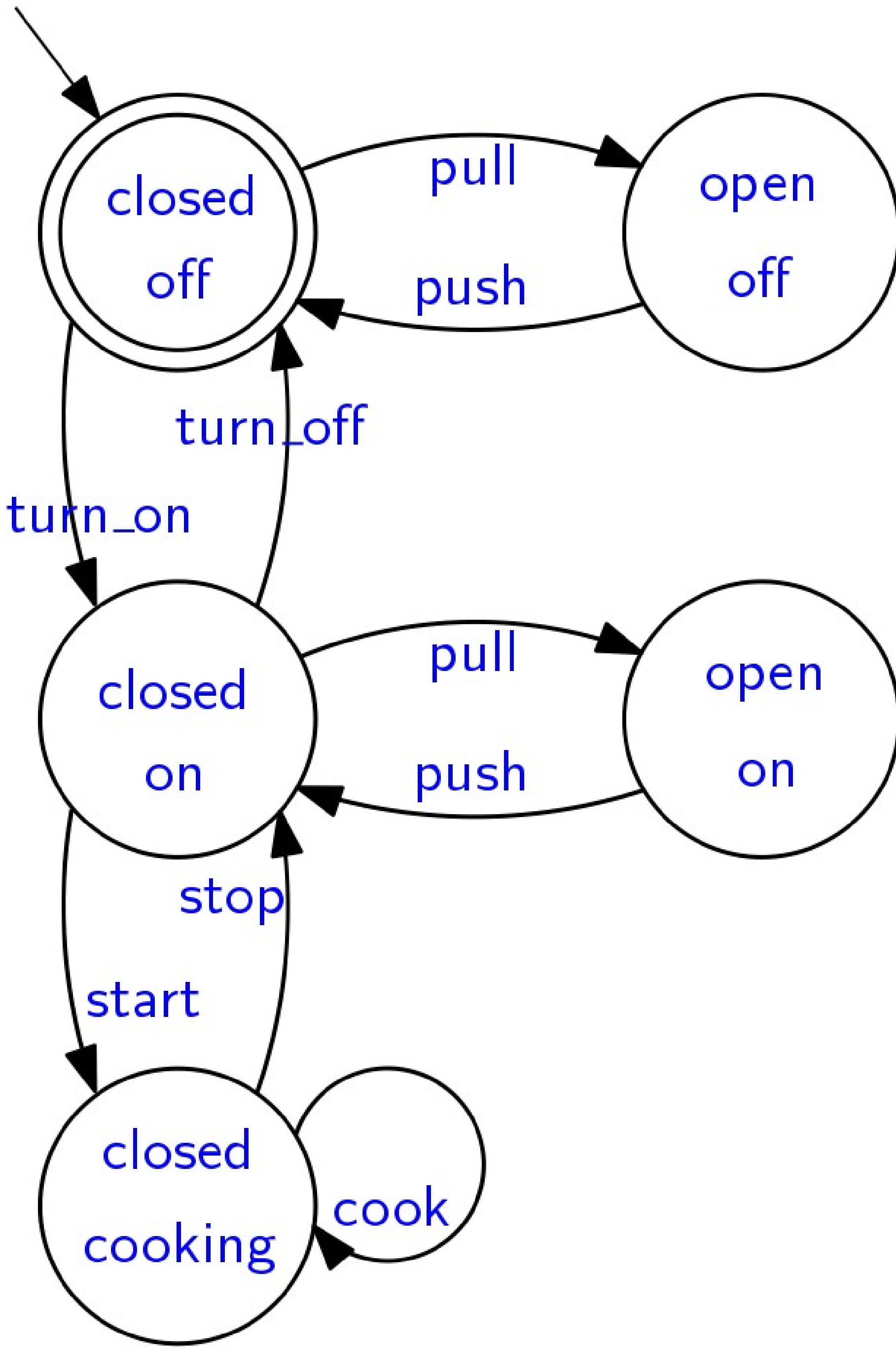


$\Box \Diamond$ turn_off

No:

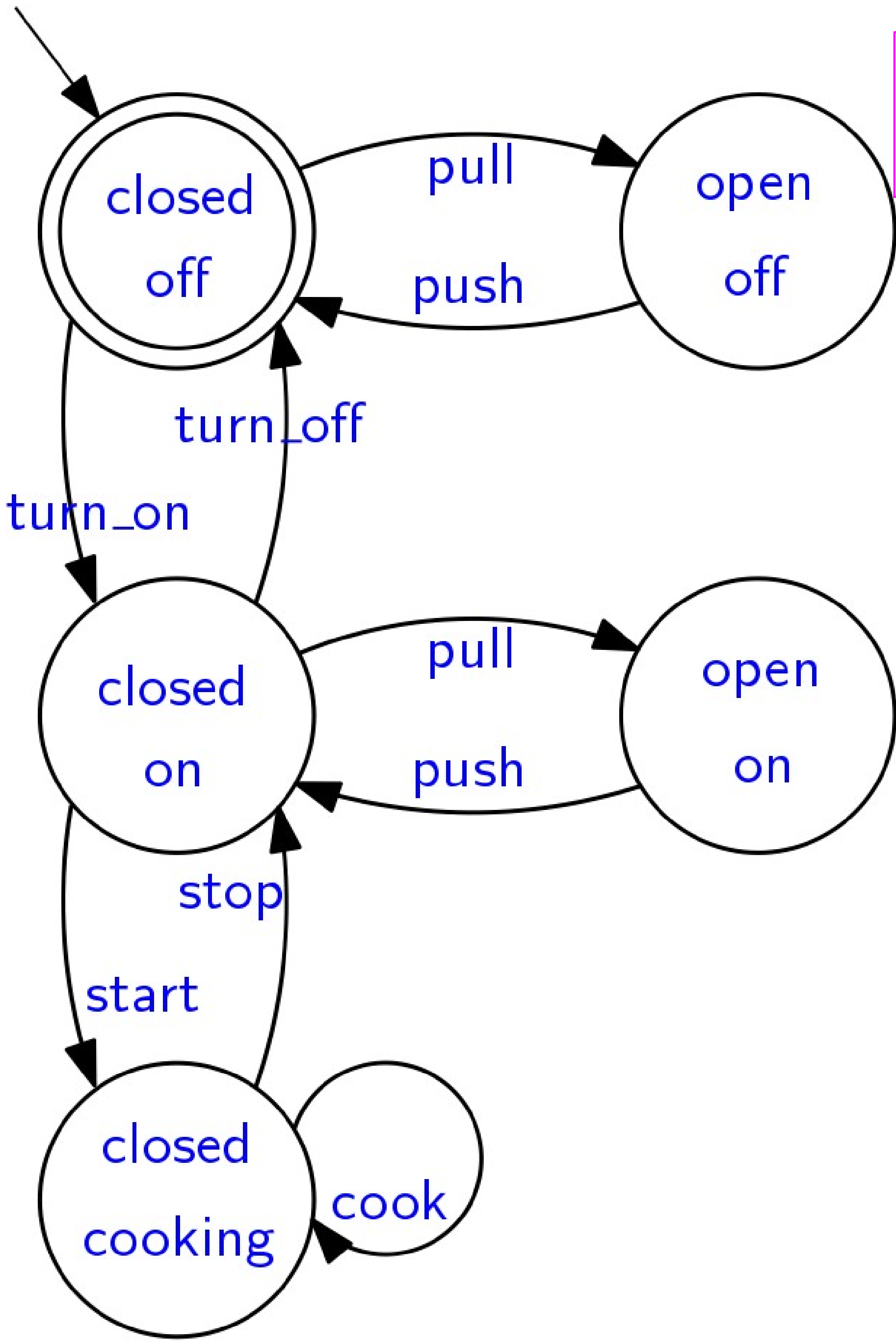
- counterexample:
  pull push

Does the property hold?



$$\Box \Diamond (turn\_off \lor push)$$

Does the property hold?

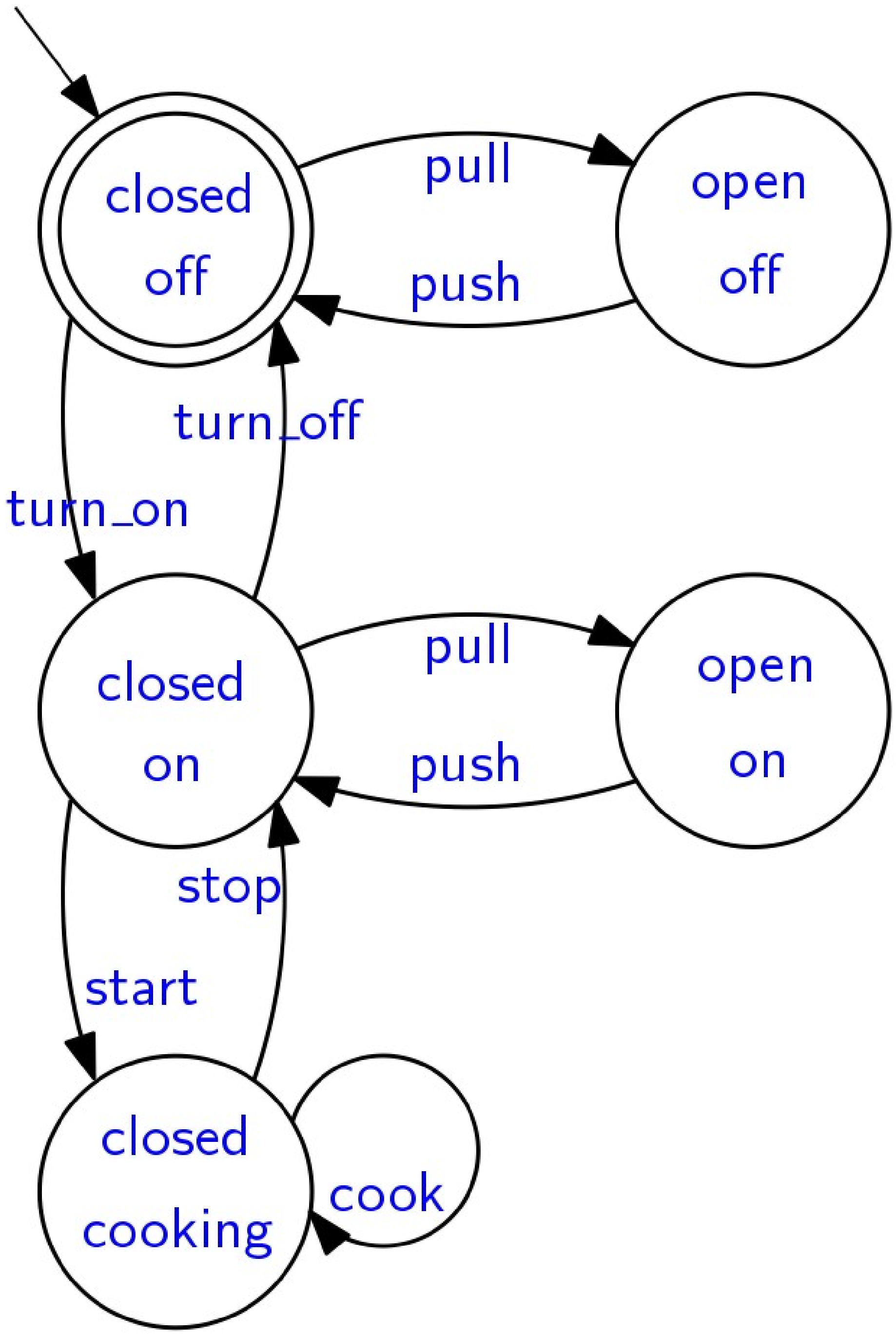


$\square \lozenge (turn\_off \lor push)$

Yes:

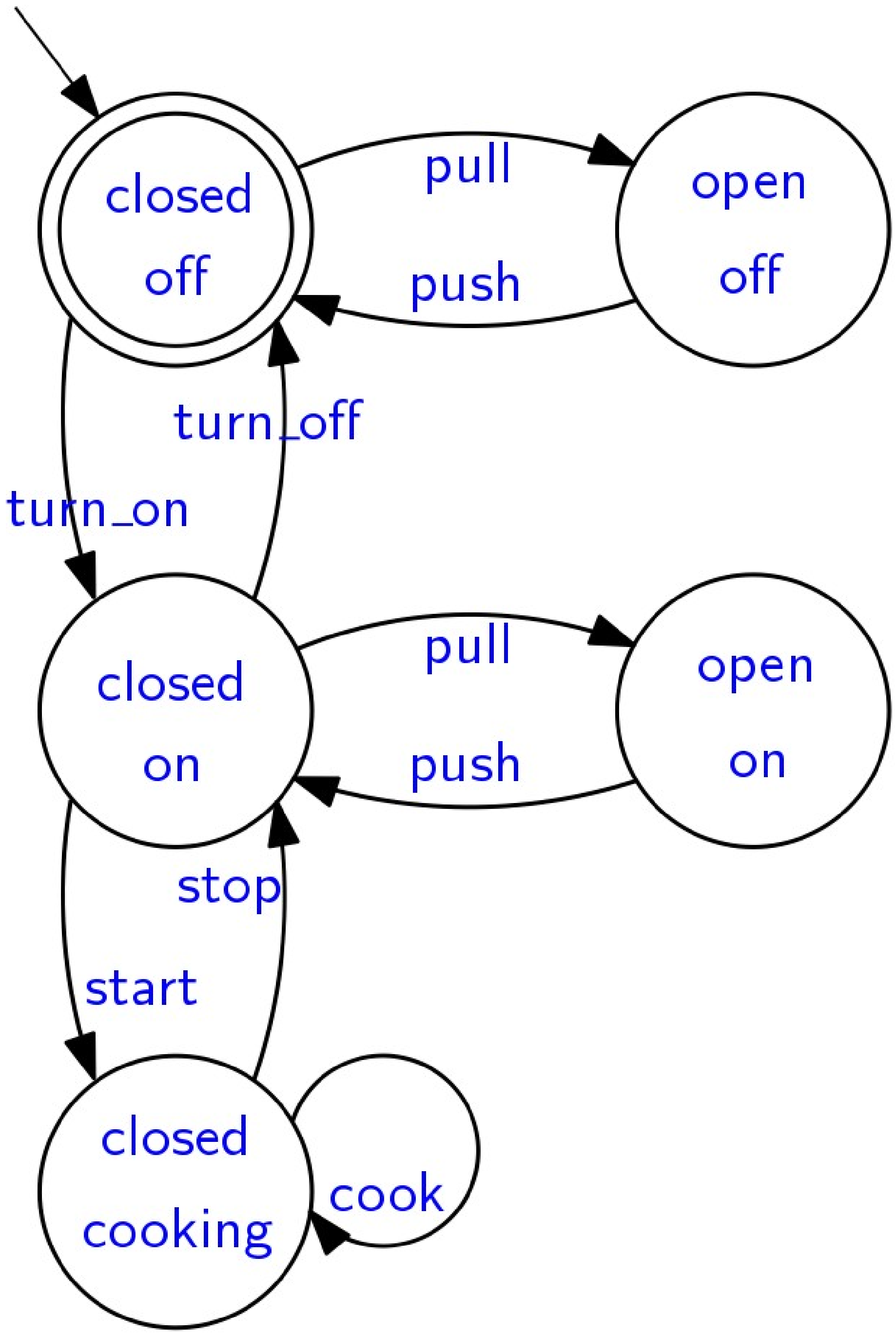
- every accepting run eventually goes back to state closed-off
- state closed-off can be reached only if either turn_off or push occurs
- the empty word is also compliant with the semantics of the always operator

Does the property hold?



$\Diamond$ (turn_off ∨ push)

Does the property hold?



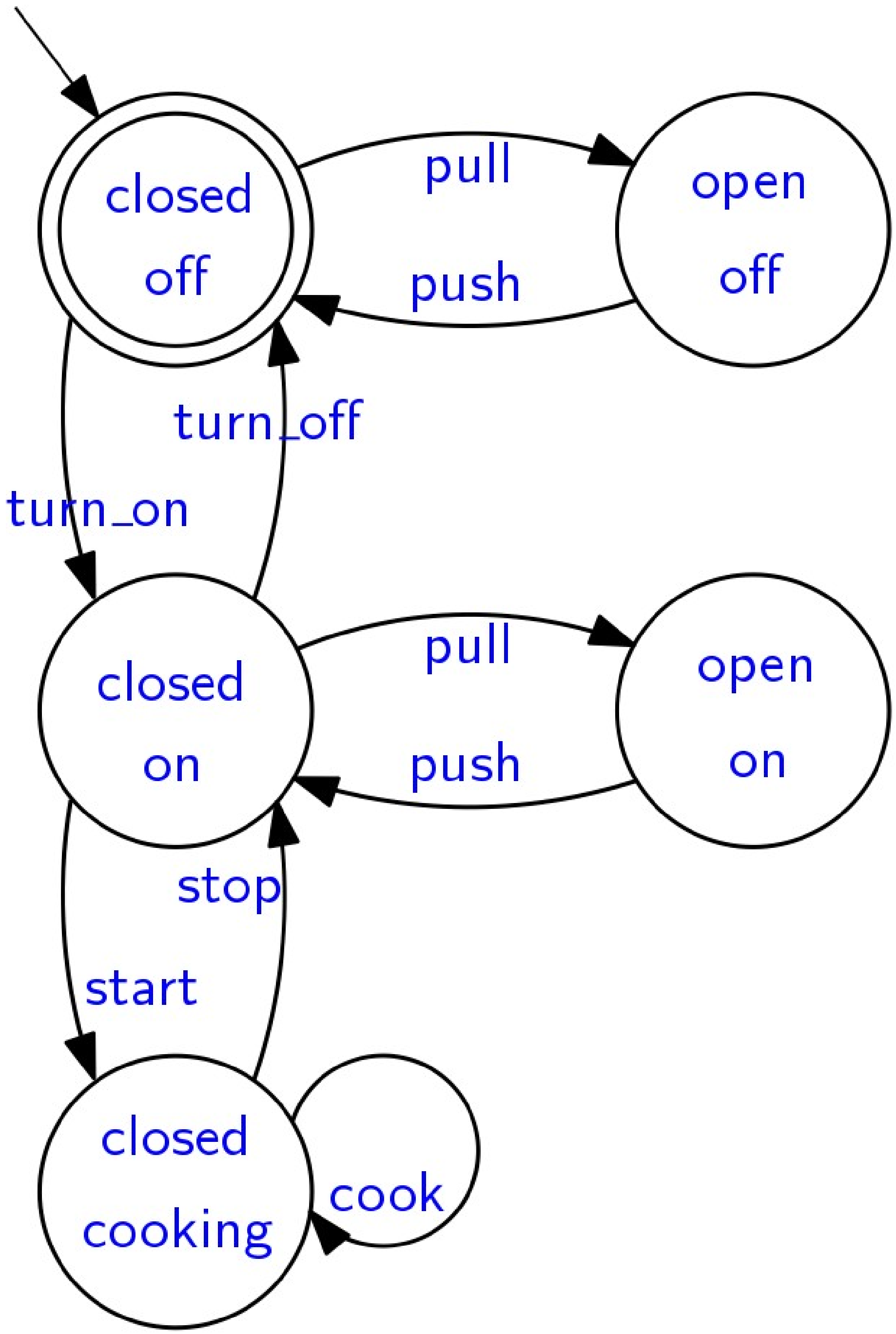
◇ (turn_off ∨ push)

No:

- counterexample: the empty word

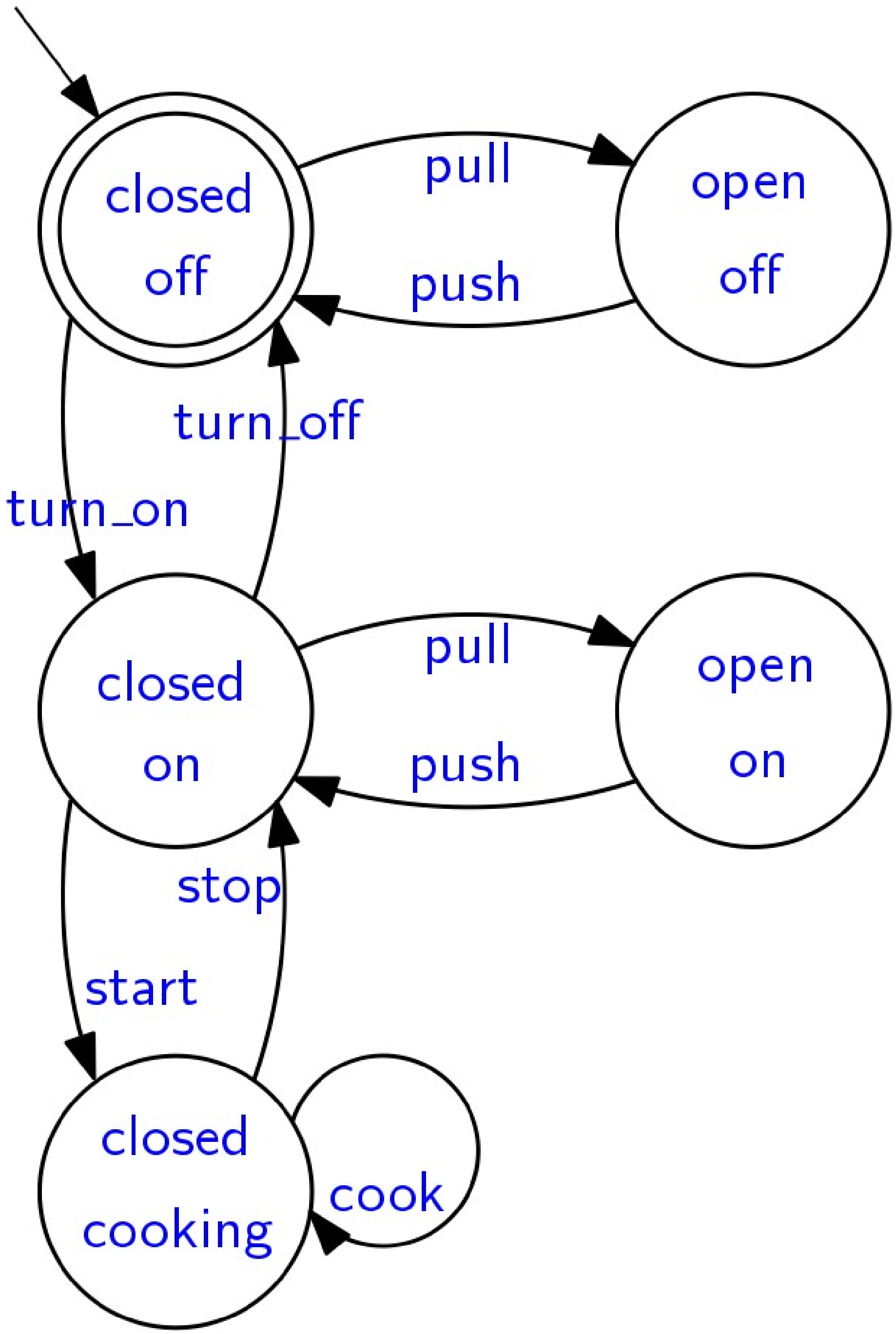
(compare the semantics of existential quantification against universal quantification)

Does the property hold?



$$\square \text{ False } \vee \Diamond (\text{turn\_off} \vee \text{push})$$

Does the property hold?



$\Box$ False

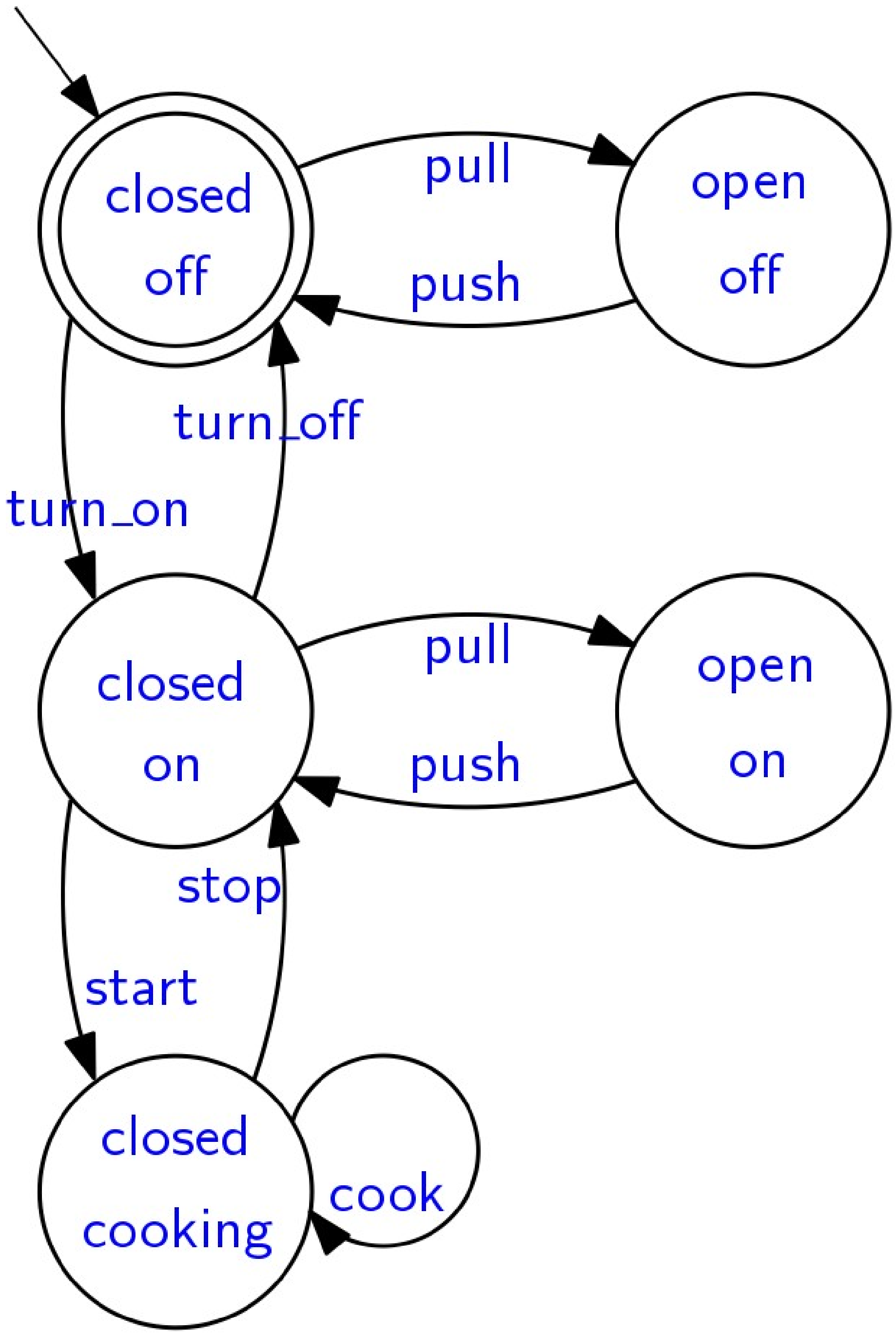
∨

$\Diamond$ (turn_off ∨ push)

Yes:

- "always False" means that False holds at every step in the word: it is satisfied precisely by the empty word
- if the word is not empty, then it must end with turn_off or push, thus it satisfies the other disjunct

Does the property hold?

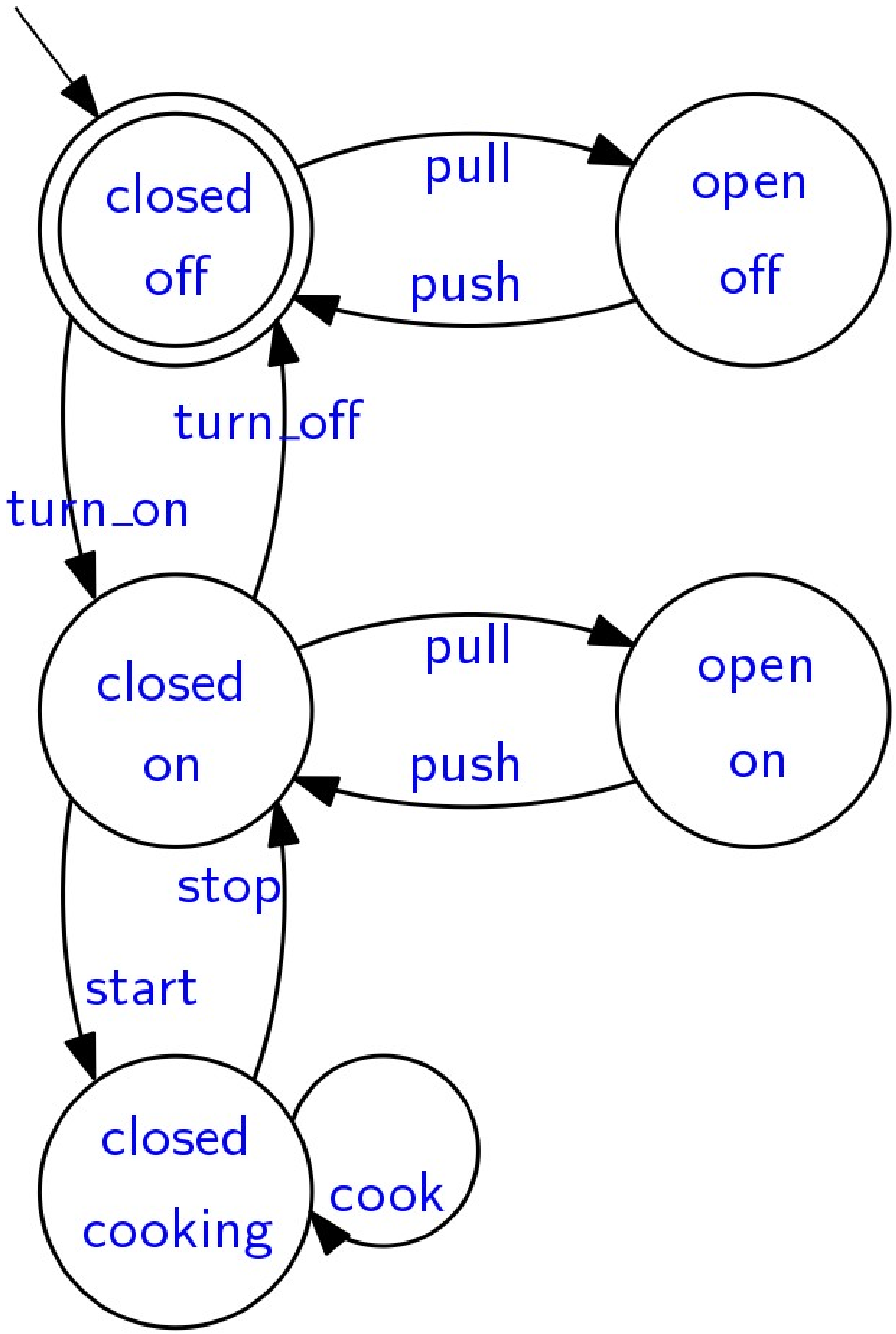


turn_on U start

∨

pull U push

Does the property hold?

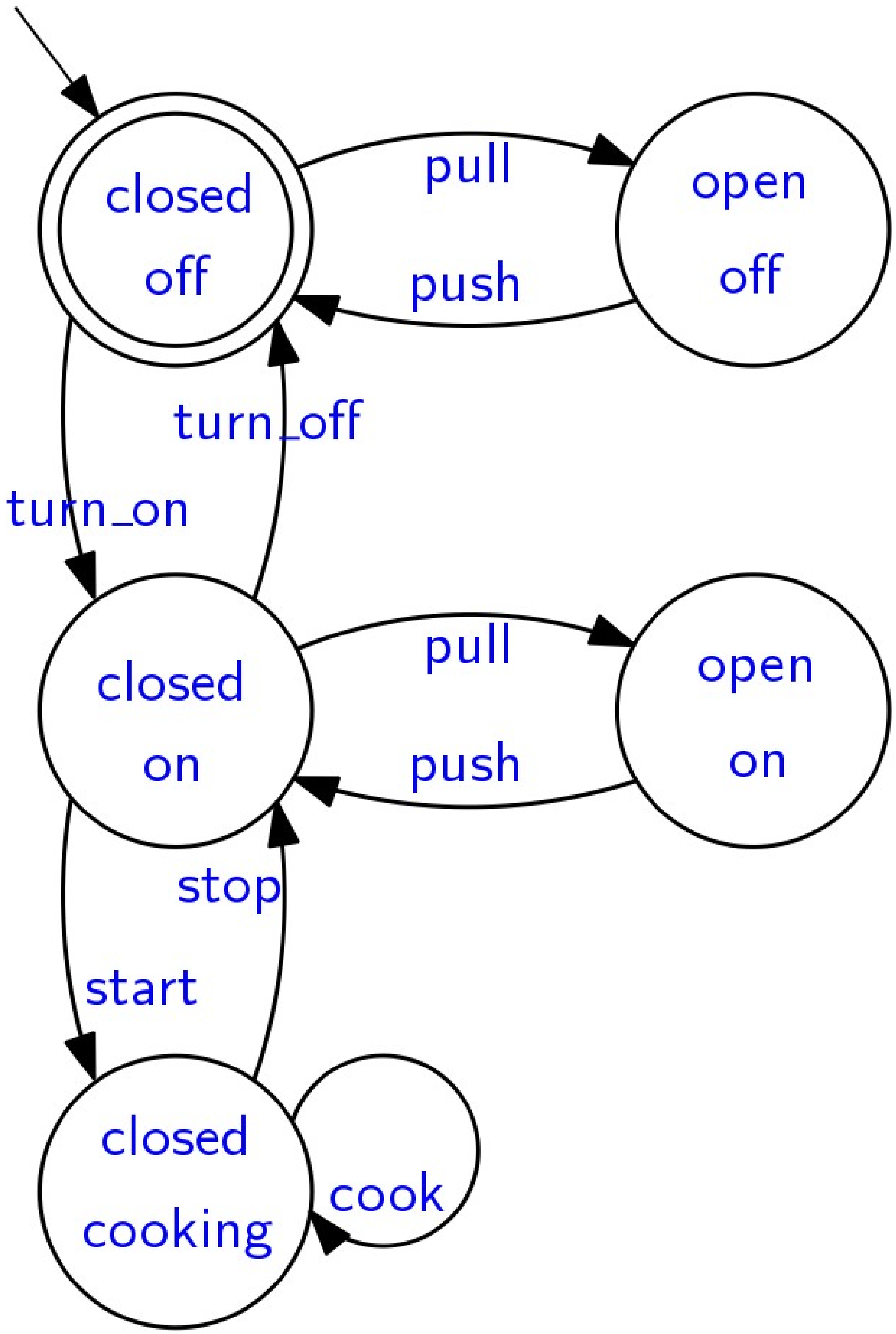


turn_on U start
∨
pull U push

No:

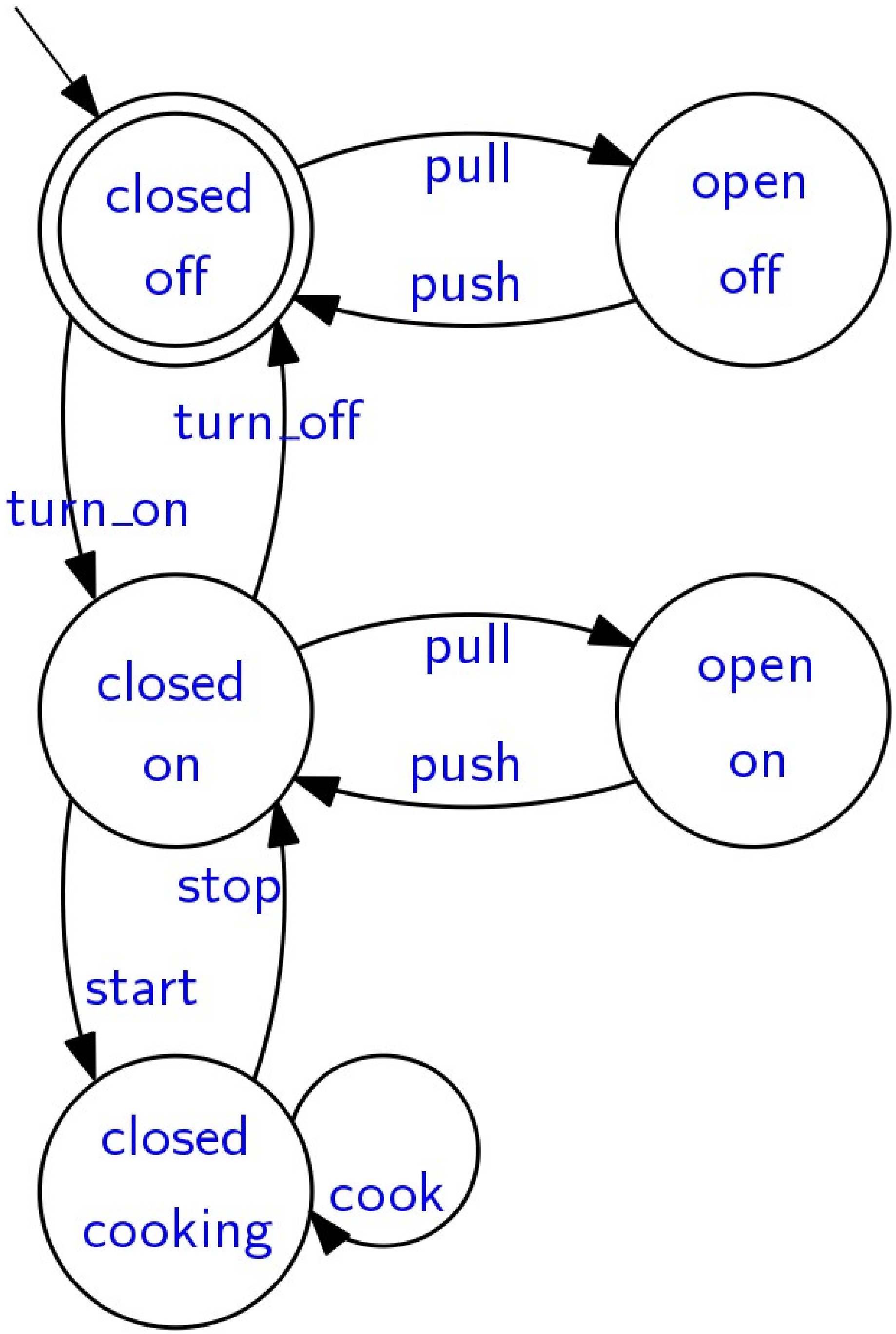
- counterexample: the empty word
- counterexample: turn_on turn_off
- counterexample: turn_on pull push turn_off

Does the property hold?



$\square$ (start $\Rightarrow$ (cook U $\Diamond$turn_off))

Does the property hold?



$\Box$ (start $\Rightarrow$ (cook U $\Diamond$turn_off))

Yes:

- once start occurs, turn_off must occur eventually
- hence "eventually turn_off" is the case right after start occurs
- cook can occur right after start occurs, one or more times

Exercises:
Equivalence of LTL formulas

Equivalence of formulas

Prove that $\Diamond$ is idempotent, that is:

$$\Diamond\Diamond\, q$$

is equivalent to:

$$\Diamond\, q$$

Equivalence of formulas

$w,i \models \Diamond\Diamond q$

iff

for some $i \le j \le n$ it is: $w, j \models \Diamond q$

(semantics of eventually)

iff

for some $i \le j \le n$ it is: for some $j \le h \le n$ it is: $w, h \models q$

(semantics of eventually)

iff

for some $i \le j \le h \le n$ it is: $w, h \models q$

(merging of intervals)

iff

for some $i \le h \le n$ it is: $w, h \models q$

(dropping j, a fortiori)

iff

$w, i \models \Diamond q$

(semantics of eventually)

Equivalence of formulas

Prove that:

$$p \, \mathcal{U} \, \Diamond q$$

is equivalent to:

$$\Diamond q$$

Equivalence of formulas: $\Rightarrow$ direction

$w,i \models p \; \mathsf{U} \; \Diamond q$

iff

for some $i \leq j \leq n$ it is: $w, j \models \Diamond q$

and for all $i \leq k < j$ it is $w, k \models p$

(semantics of until)

implies

for some $i \leq j \leq n$ it is: $w, j \models \Diamond q$ (a fortiori)

iff

for some $i \leq j \leq n$ it is: for some $j \leq h \leq n$ it is: $w, h \models q$

(semantics of eventually)

iff

for some $i \leq h \leq n$ it is: $w, h \models q$

(simplification of range of quantification)

iff

$w, i \models \Diamond q$ (semantics of eventually)

Equivalence of formulas: ⇐ direction

w,i ⊨ ◇ q

iff

for some i ≤ j ≤ i: w, j ⊨ ◇ q

(singleton range of quantification)

iff

for some i ≤ j ≤ i: w, j ⊨ ◇ q and True

(semantics of and)

iff

for some i ≤ j ≤ i: w, j ⊨ ◇ q

and for all i ≤ k < j=i it is w, k ⊨ p

(semantics of universally quantified empty range)

implies

for some i ≤ j ≤ n: w, j ⊨ ◇ q

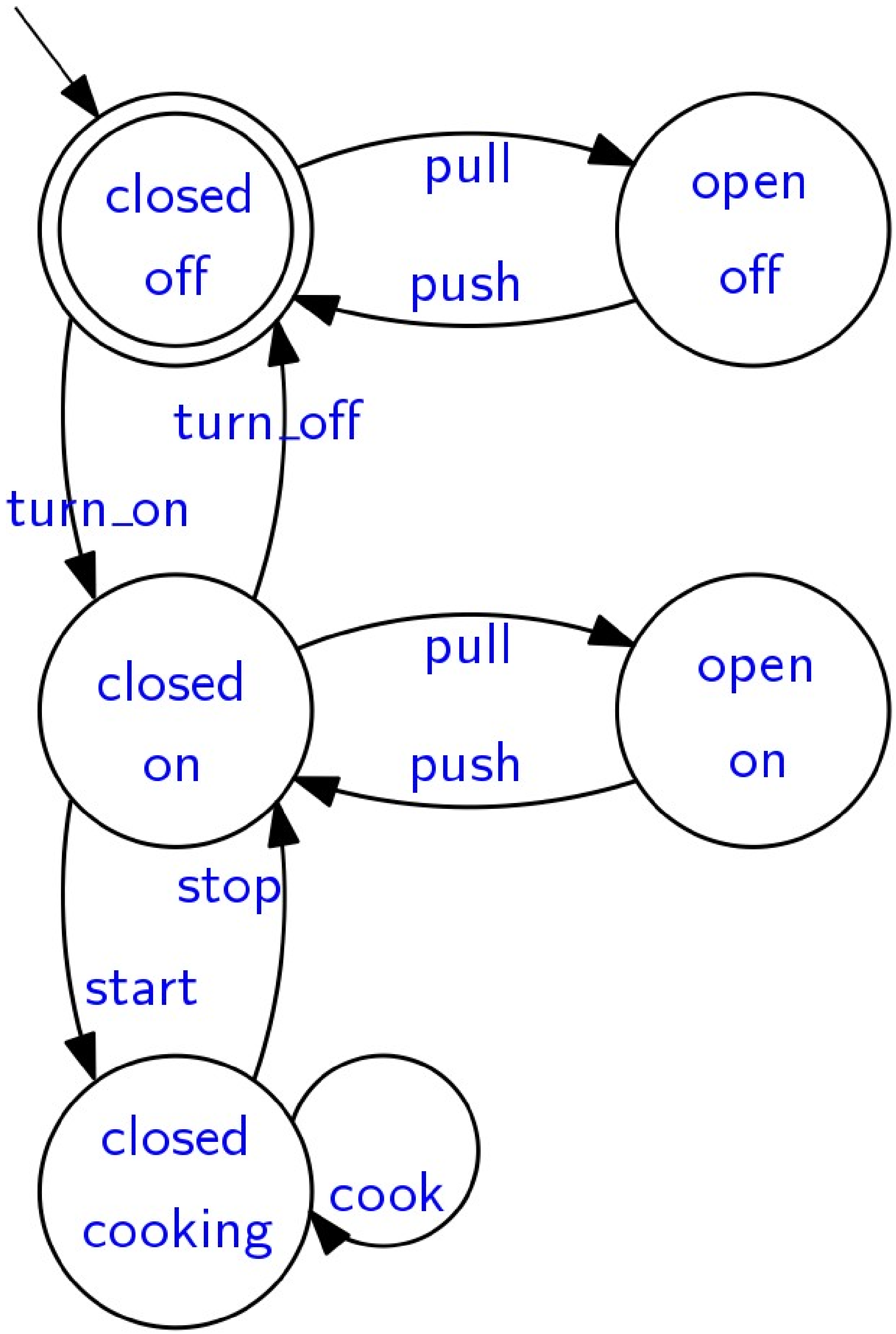
and for all i ≤ k < j it is w, k ⊨ p (a fortiori)

iff

w, i ⊨ p U ◇ q (semantics of until)

Exercises:
Automata-theoretic model-checking
(on paper)

Automata-based model checking



$\square \diamond$ turn_off

Let us prove by model checking that it's not a property of the automaton

LTL2FSA

Build an automaton with the same language as:

$\neg(\Box \Diamond$ turn_off)

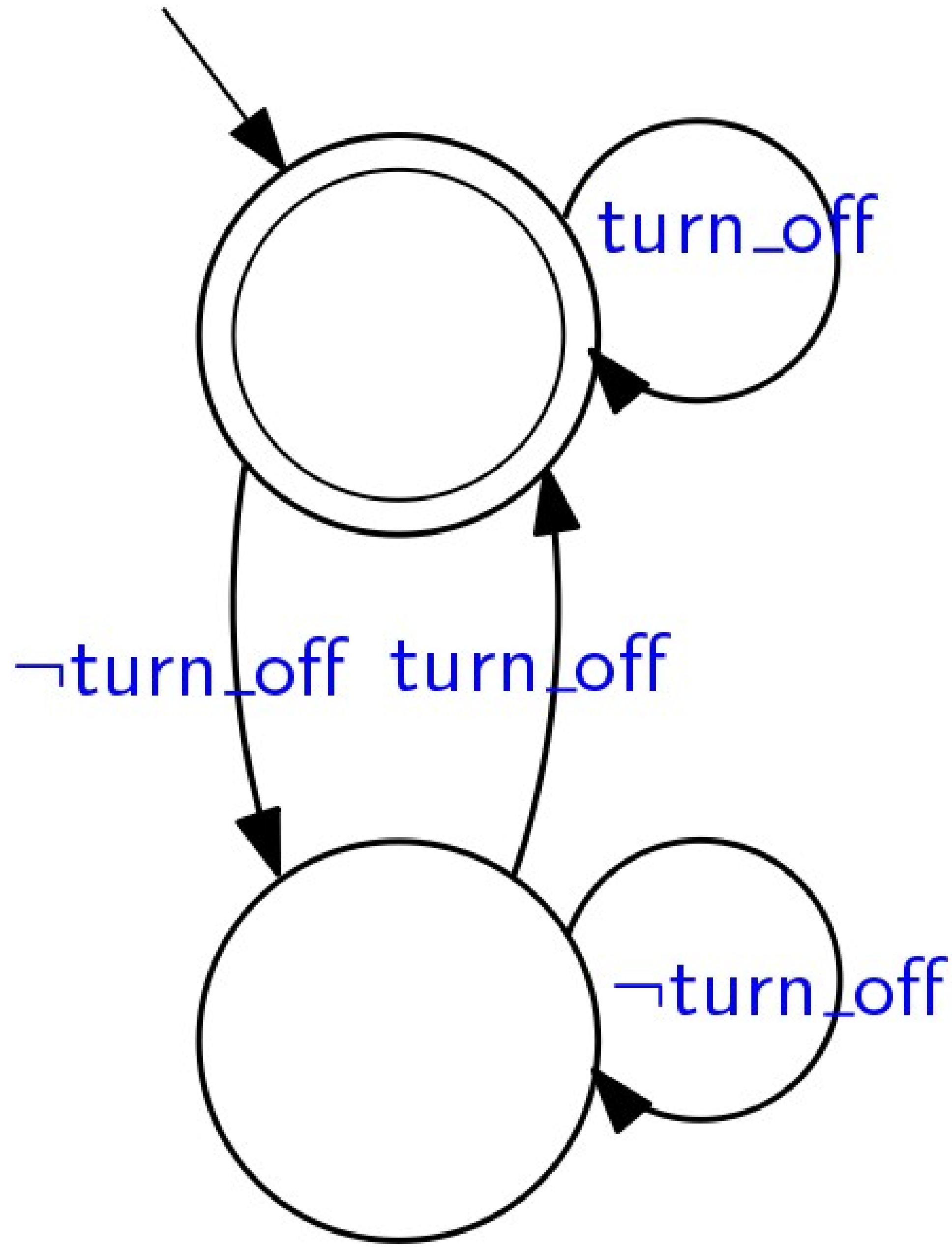
Let us start from the unegated formula:

$\Box \Diamond$ turn_off

and then complement the states of the automaton

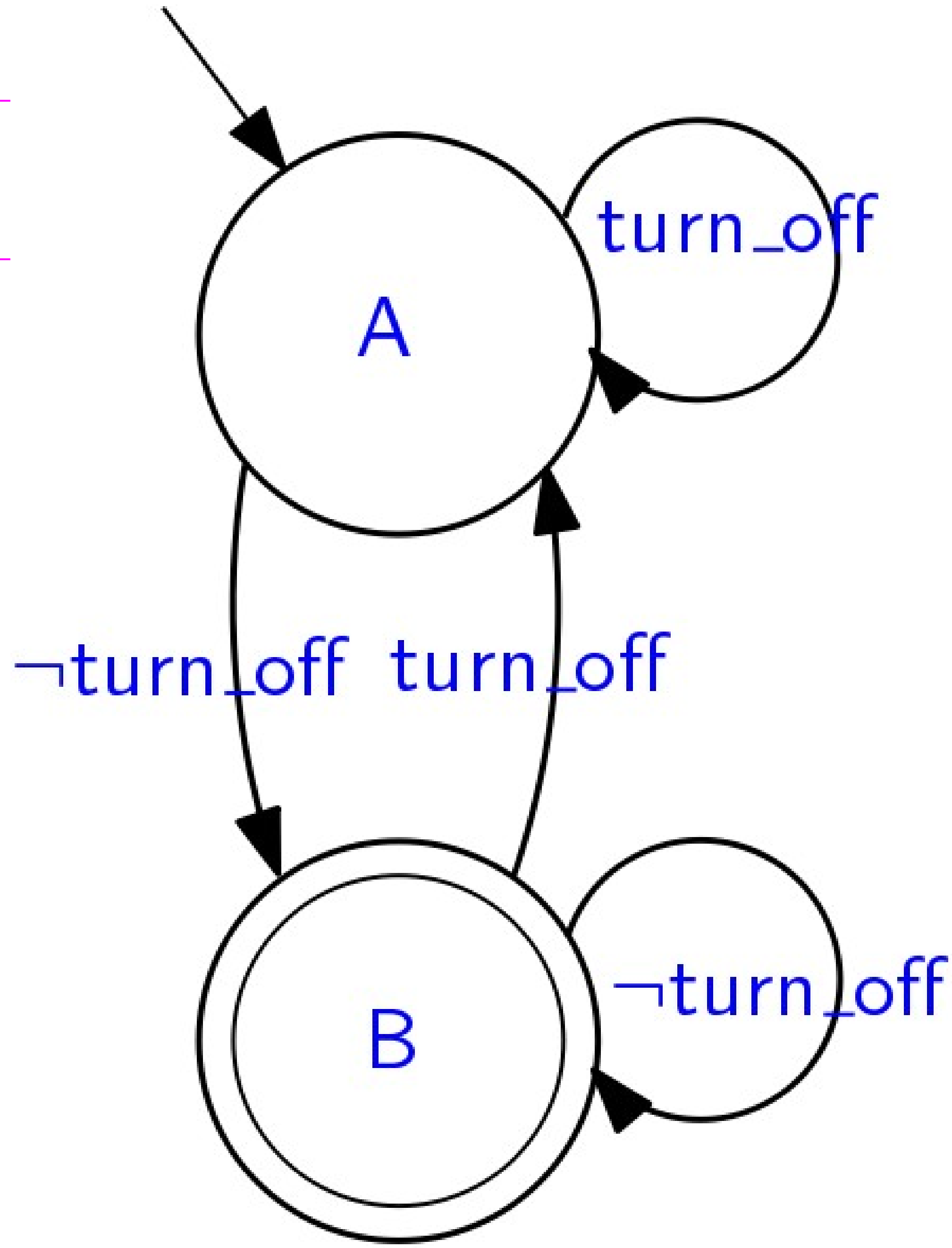
LTL2FSA

$\Box \Diamond$ turn_off

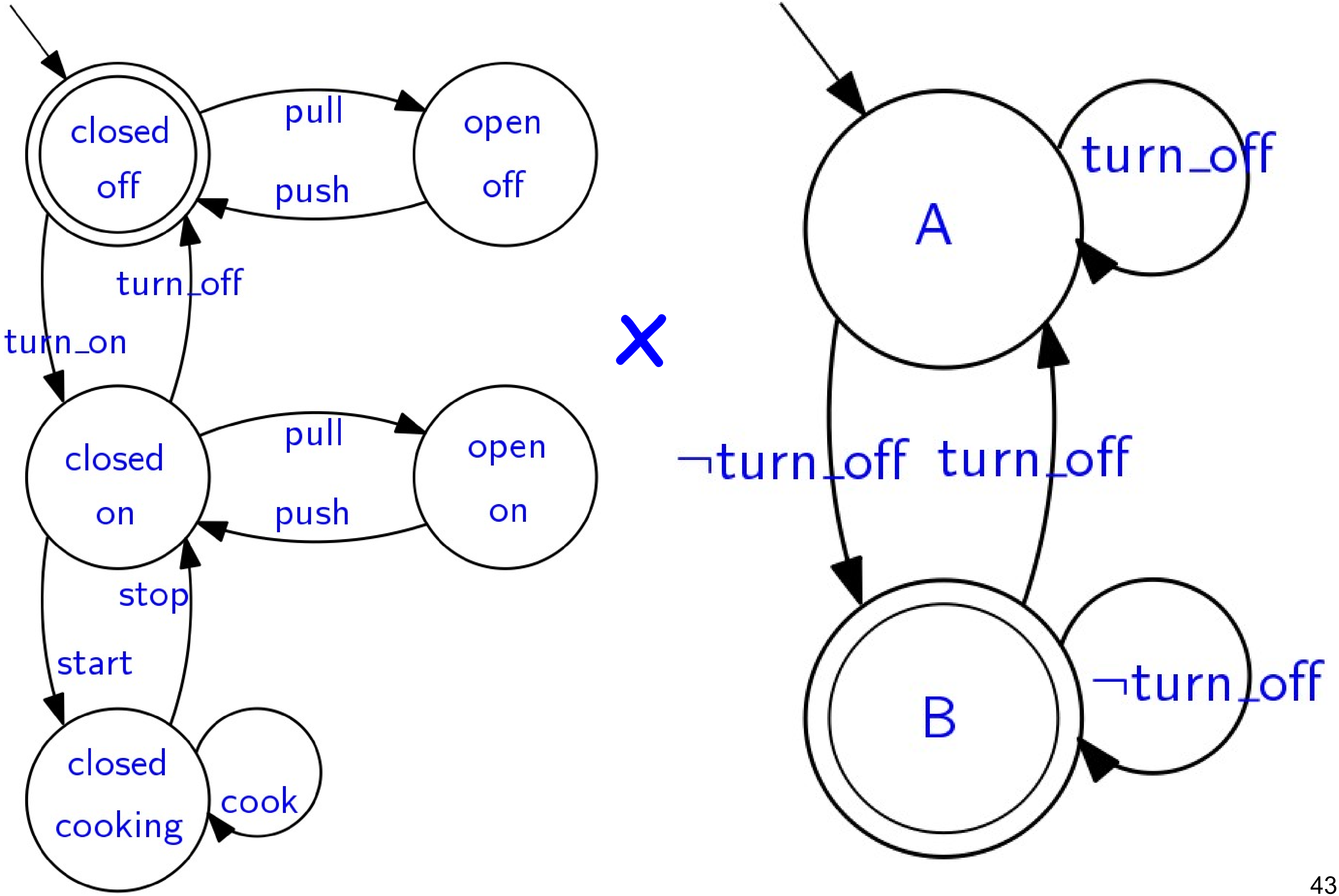


LTL2FSA

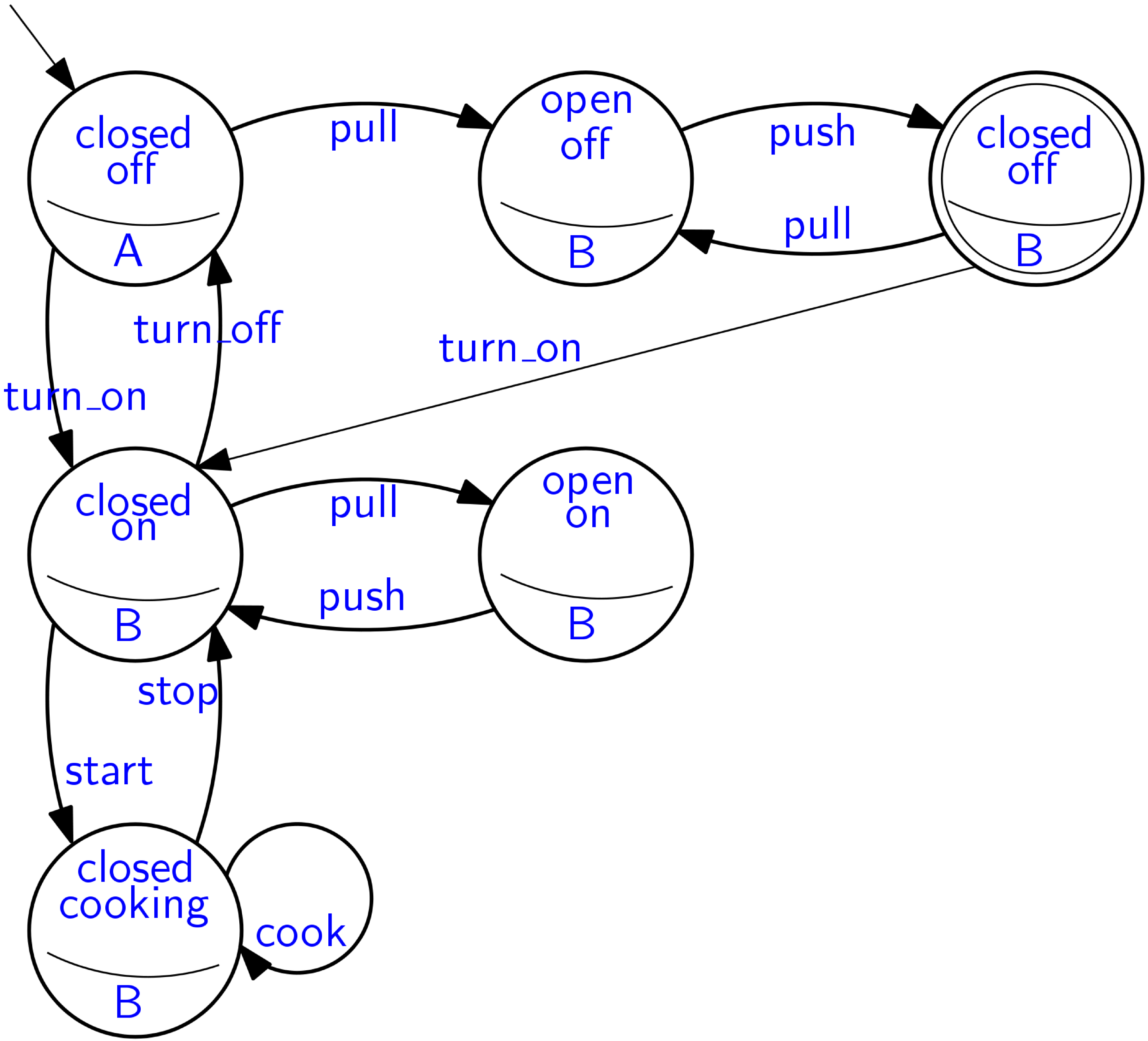
$\neg(\square \lozenge turn\_off)$



FSA Intersection

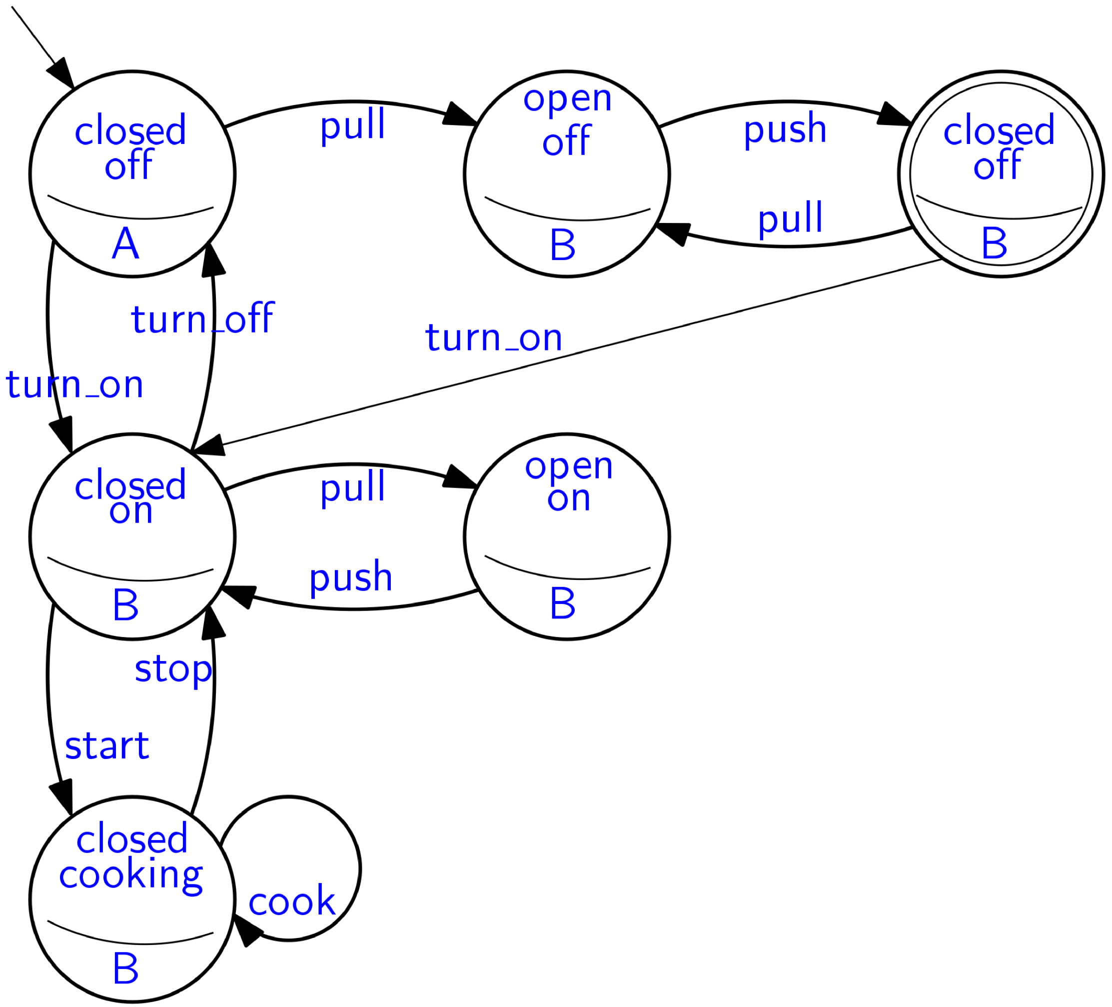


FSA Intersection



FSA-Emptiness: node reachability

Any accepting run on the intersection automaton is a counterexample to the LTL formula being a property of the automaton



- pull push
- pull push pull push
- ...