Software Verification
Exercise class:
Model Checking

Exercises:
Semantics of derived operators

LTL derived operators: eventually

Prove that the satisfaction relation

$$w, i \models \,<\!>\, F$$

for eventually, defined as:

$$<\!>\, F \triangleq \text{True } U \; F$$

is equivalent to:

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

LTL derived operators: eventually

$w, i \models \lozenge F$ (written `<>` F)

iff

$w, i \models \mathit{True}\ \mathsf{U}\ F$ (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 $w, k \models \mathit{True}$ (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

$$w, i \models [\,]\, F$$

for always, defined as:

$$[\,]\, F \triangleq \neg <> \neg F$$

is equivalent to:

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

LTL derived operators: always

w, i ⊨ [] F

iff

w, i ⊨ ¬ <> ¬F (definition of always)

iff

w, i ⊨ <> ¬F is not the case (definition of not)

iff

it is not the case that: for some i ≤ j ≤ n it is: w, j ⊨ ¬F

(semantics of eventually)

iff

for all i ≤ j ≤ n it is not the case that w, j ⊨ ¬F

(semantics of quantifiers: pushing negation inward)

iff

for all i ≤ j ≤ n: it is not the case that it is not the case that w, j ⊨ F

(semantics of negation)

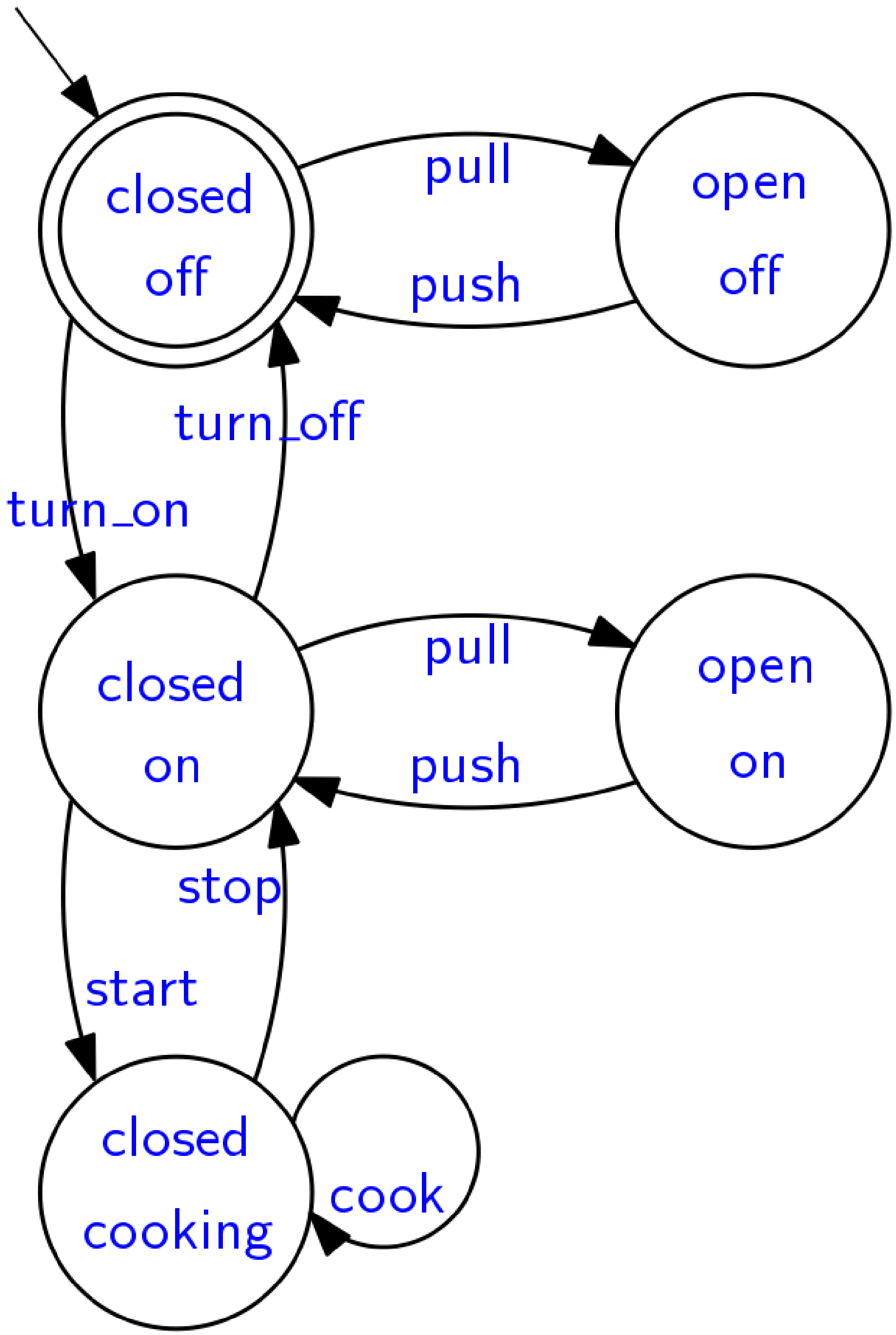
iff

for all i ≤ j ≤ n it is: w, j ⊨ F

(simplification of double negation)

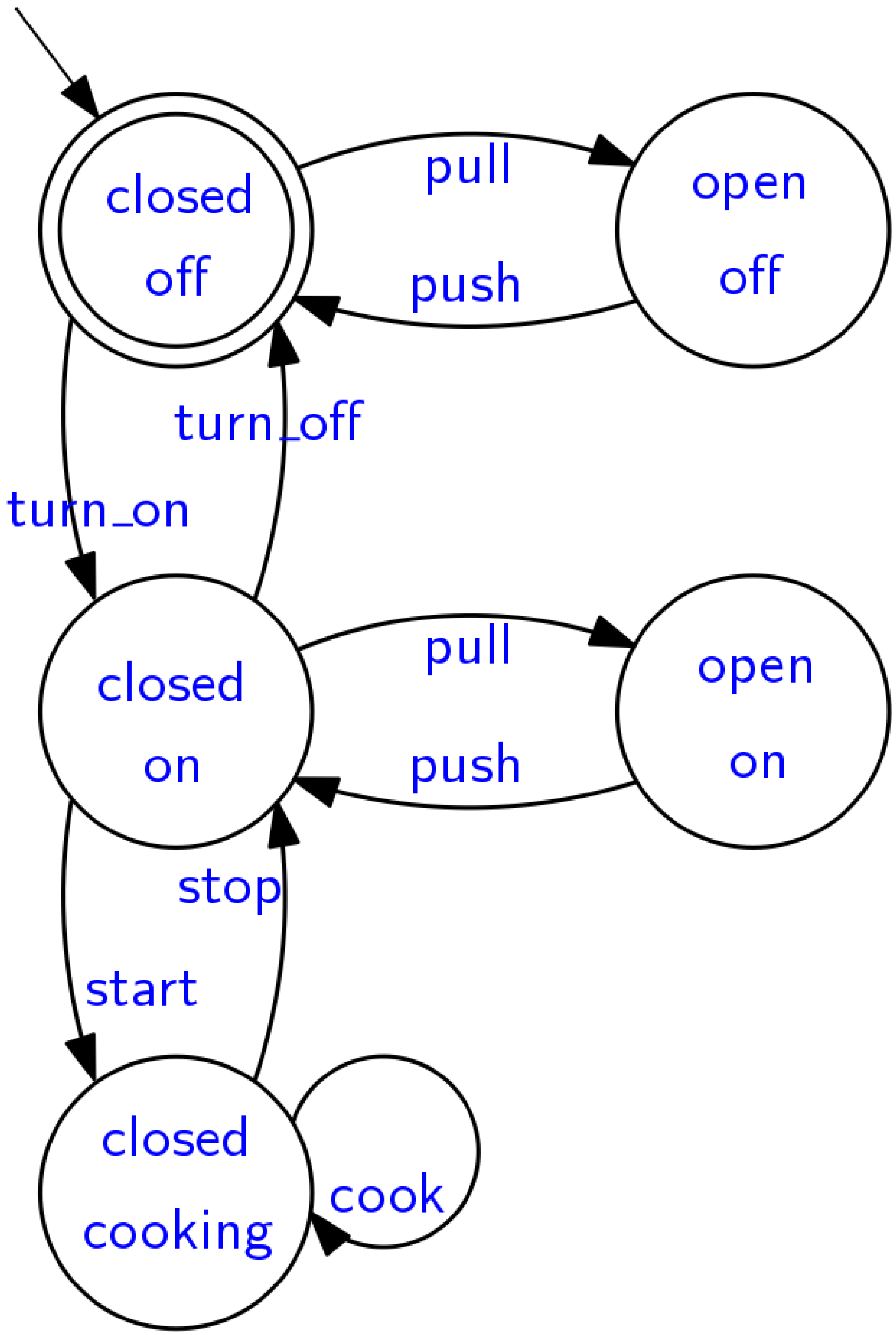
Exercises:
Evaluate LTL formulas on automata

Does the property hold?



[] (start ⇒ <> stop)

Does the property hold?

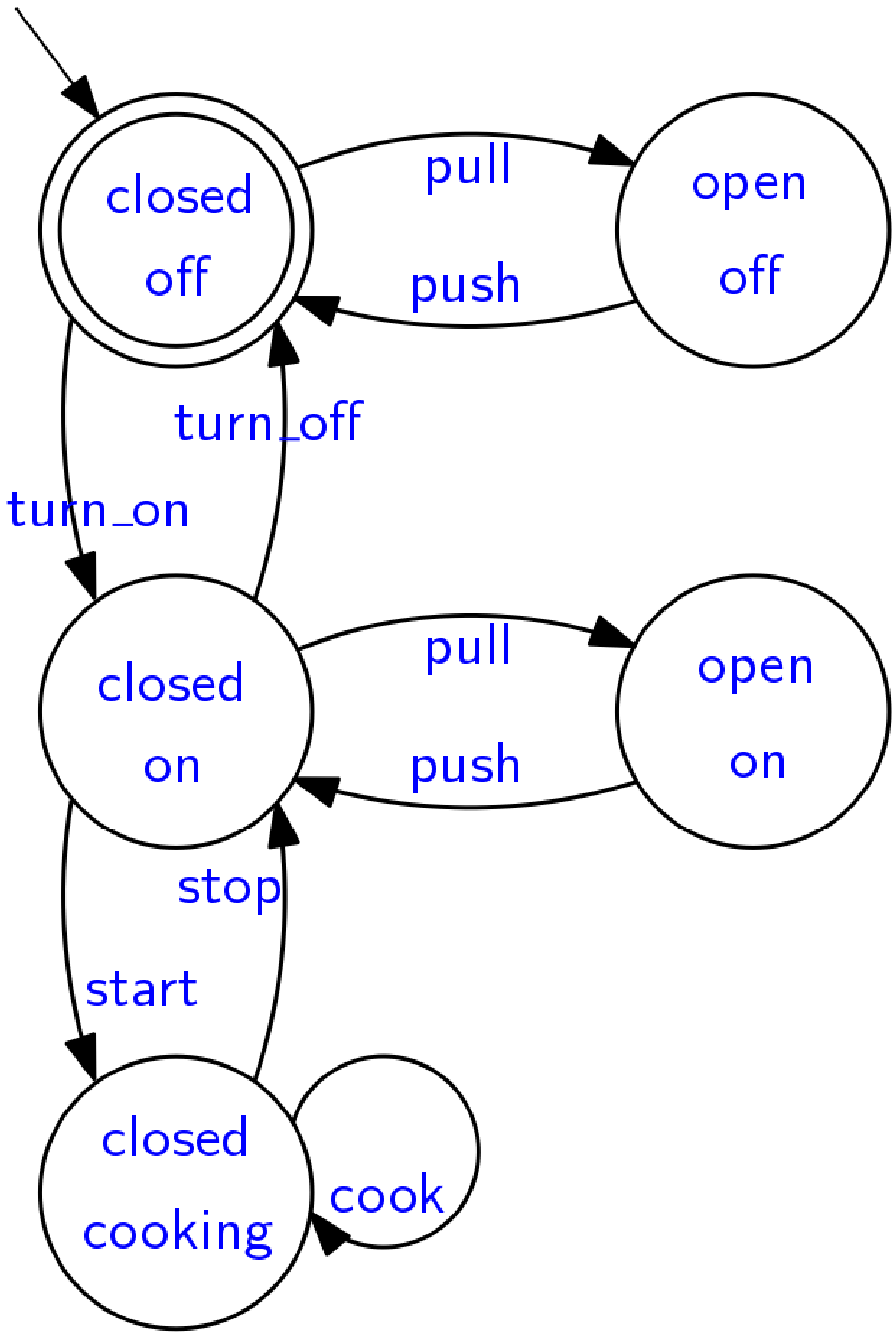


[] (start ⇒ <> stop)

Yes:

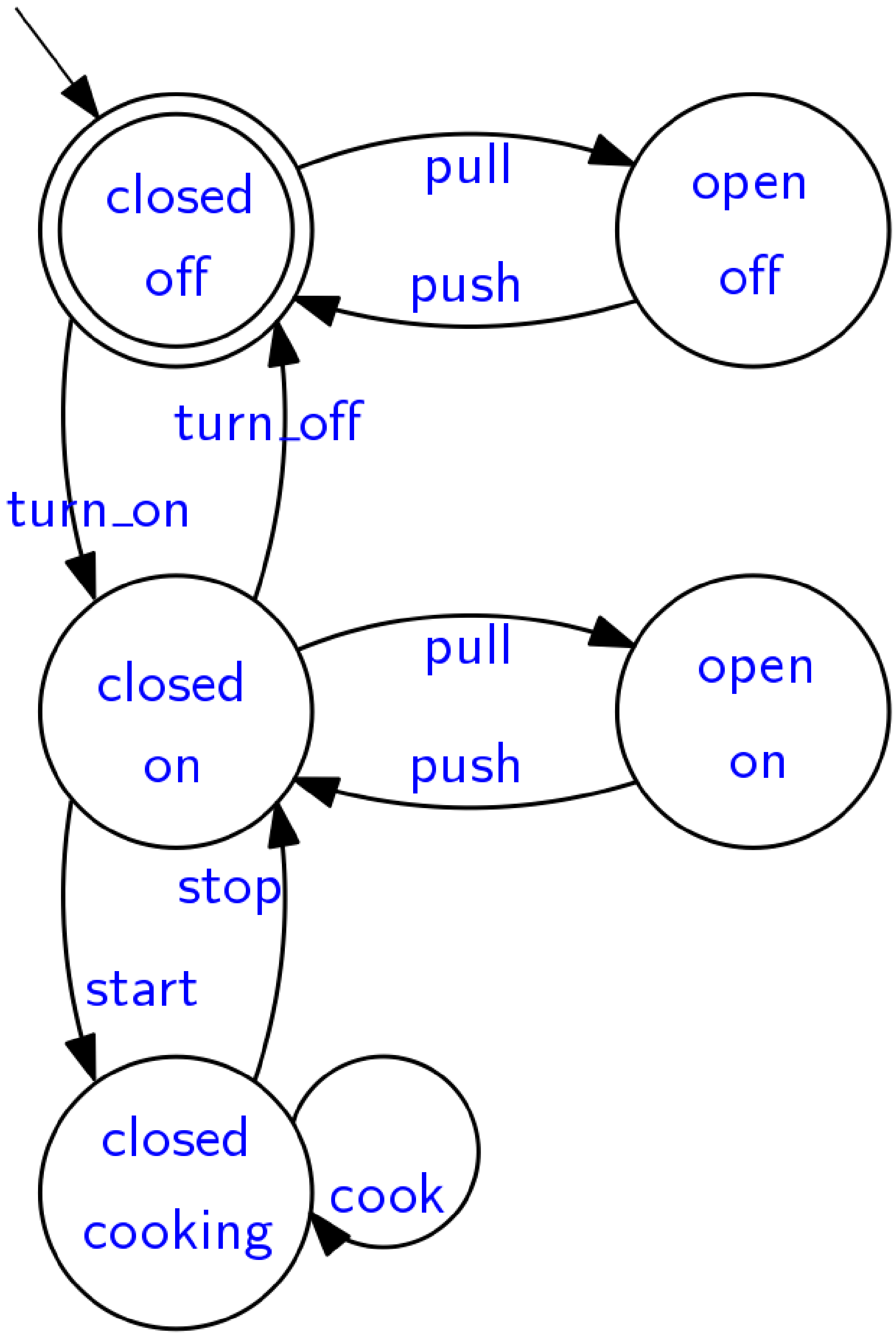
- whenever start occurs we reach state closed-cooking
- we must eventually exit state closed-cooking to reach the only accepting state closed-off
- state closed-cooking can be exited only if stop occurs

Does the property hold?



[] <> turn_off

Does the property hold?

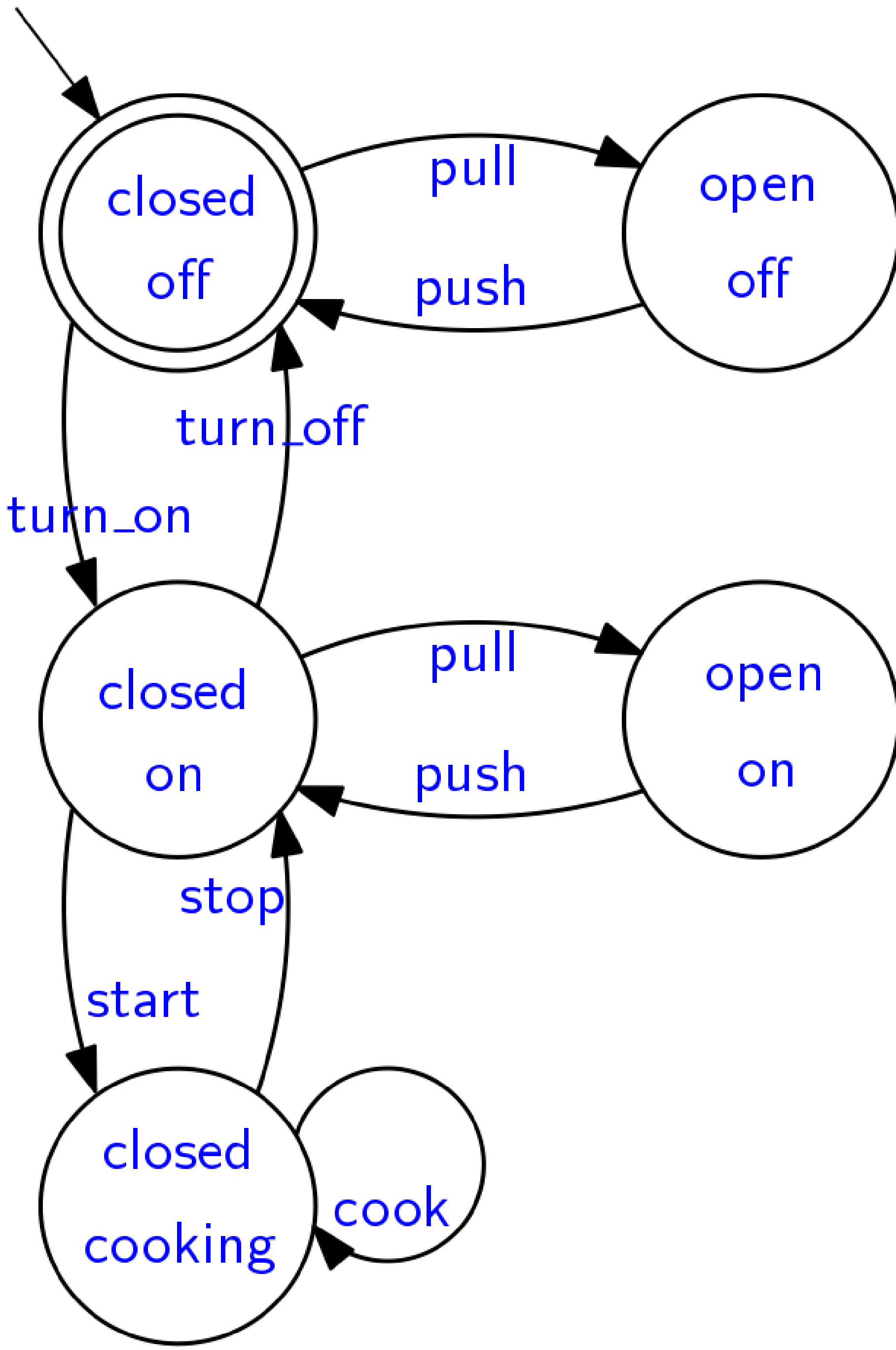


[] <> turn_off

No:

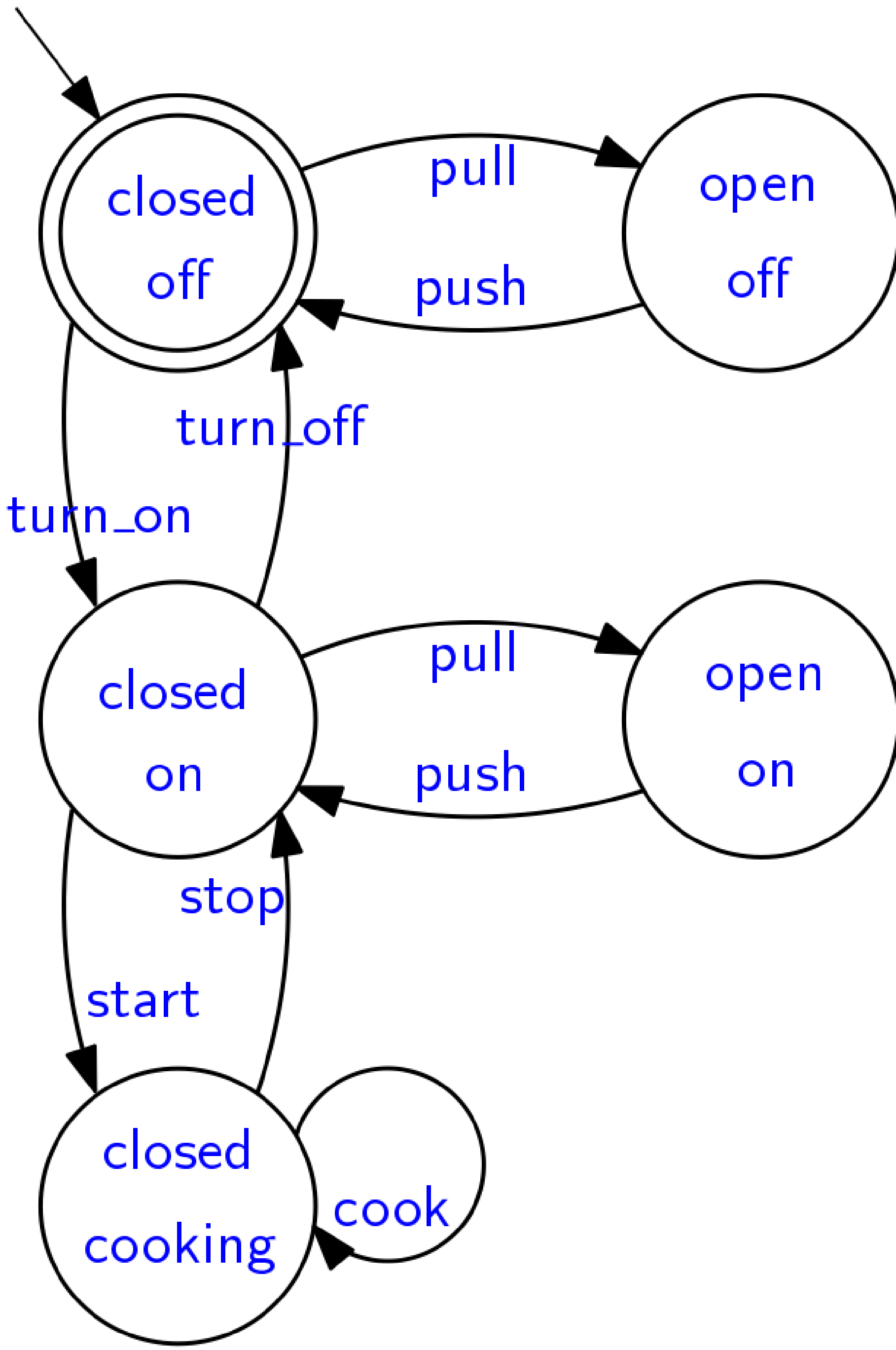
- counterexample:
  pull push

Does the property hold?



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

Does the property hold?

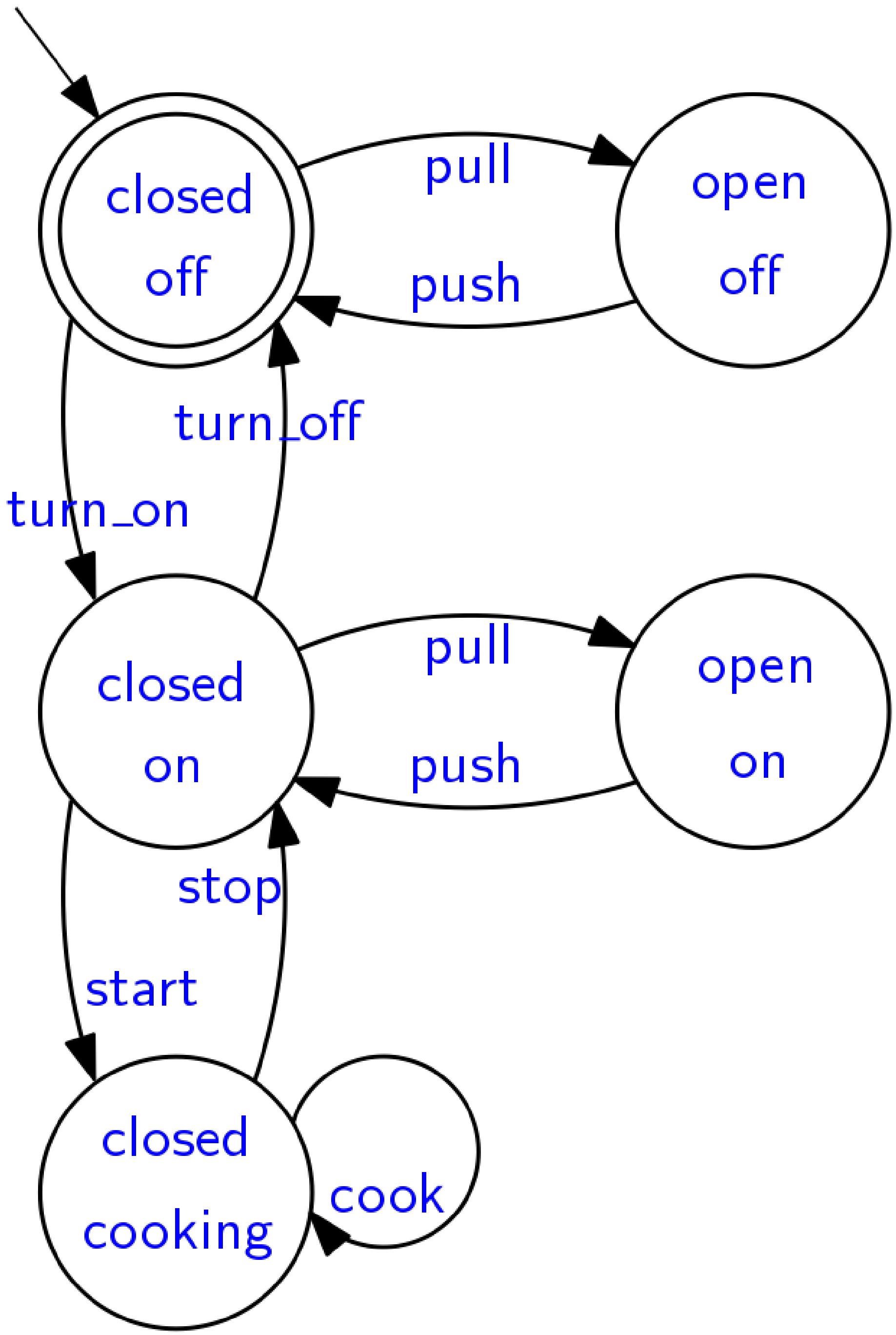


$[\,]\,\langle\rangle\,(turn\_off \vee push)$

Yes:

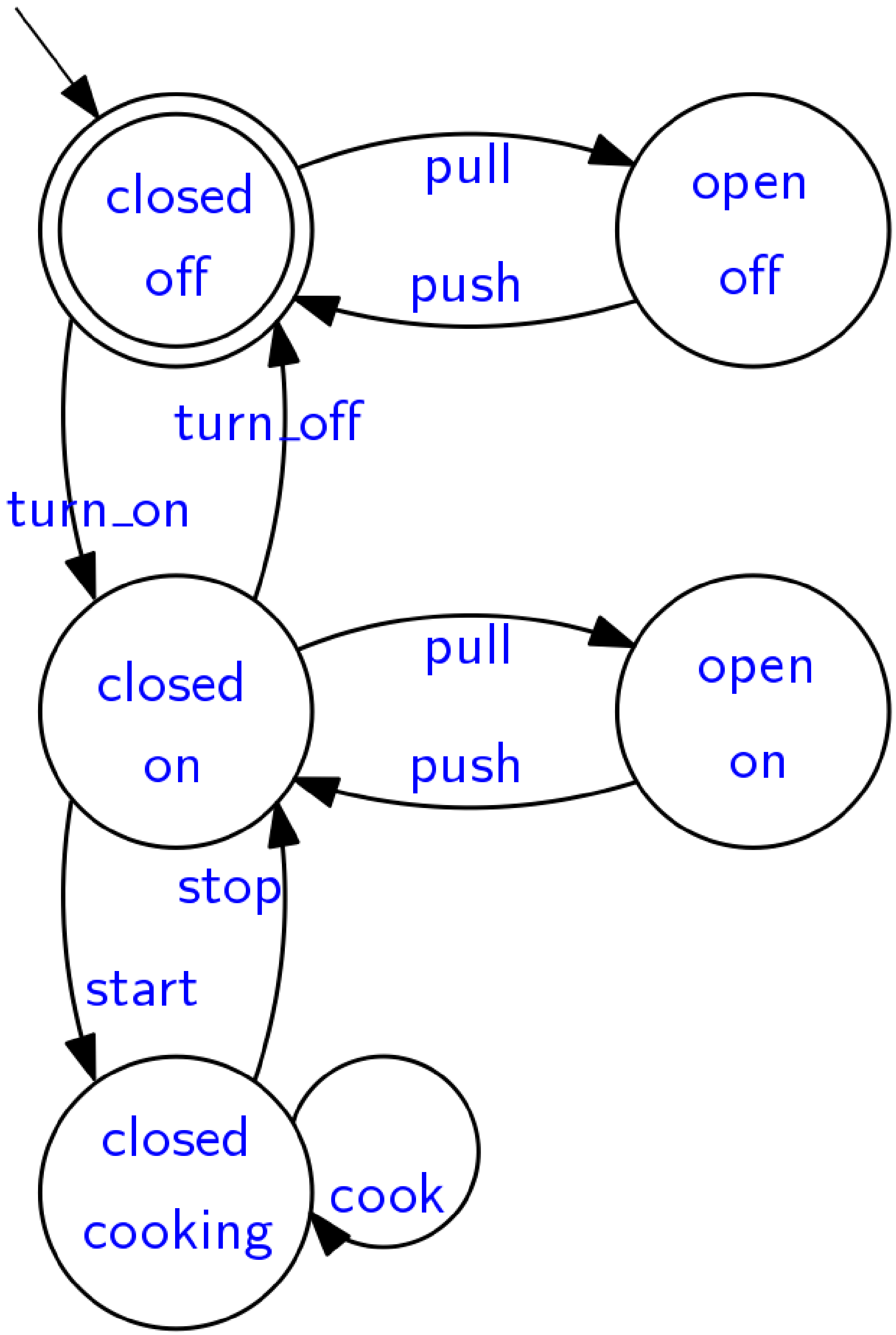
- every accepting run eventually goes back to state closed-off
- state closed-off can be reached only if either turn_off or push occurs
- the empty word is also compliant with the semantics of the always operator

Does the property hold?



$\Diamond$ (turn_off ∨ push)

Does the property hold?



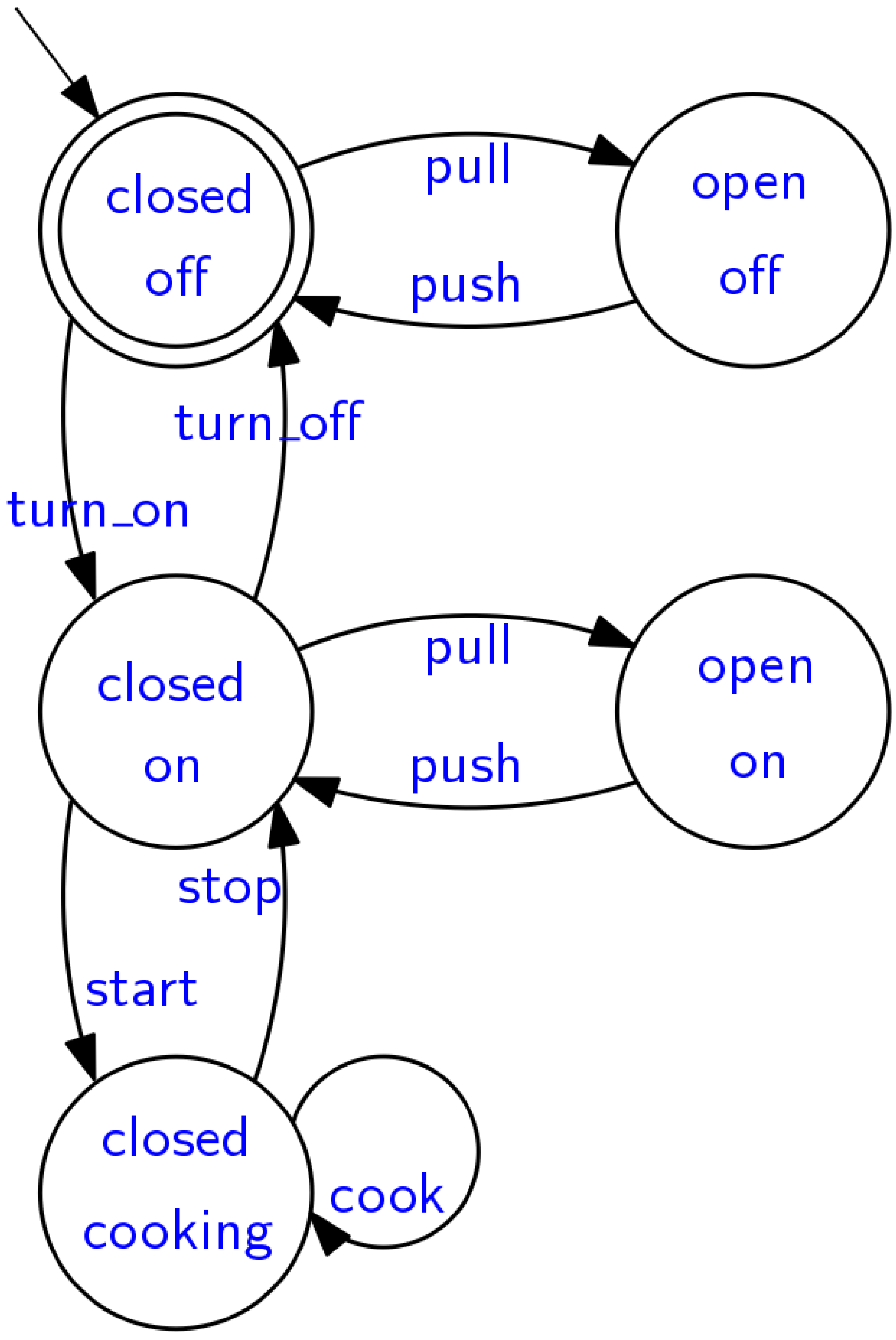
<> (turn_off ∨ push)

No:

- counterexample: the empty word

(compare the semantics of existential quantification against universal quantification)

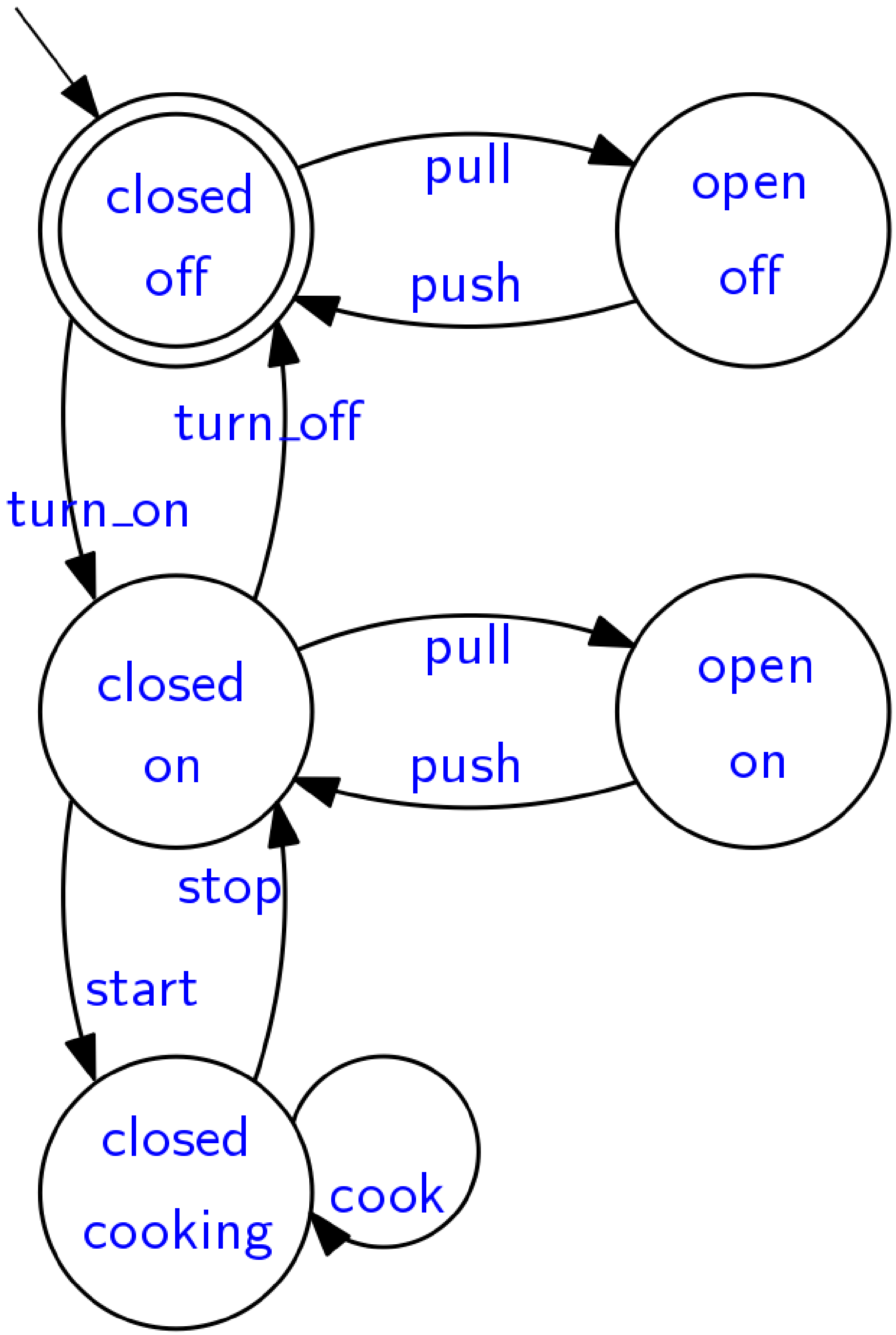
Does the property hold?



$$\Box\, False \;\lor\; \Diamond\, (turn\_off \lor push)$$

[] False
V
<> (turn_off V push)

Does the property hold?



[] False

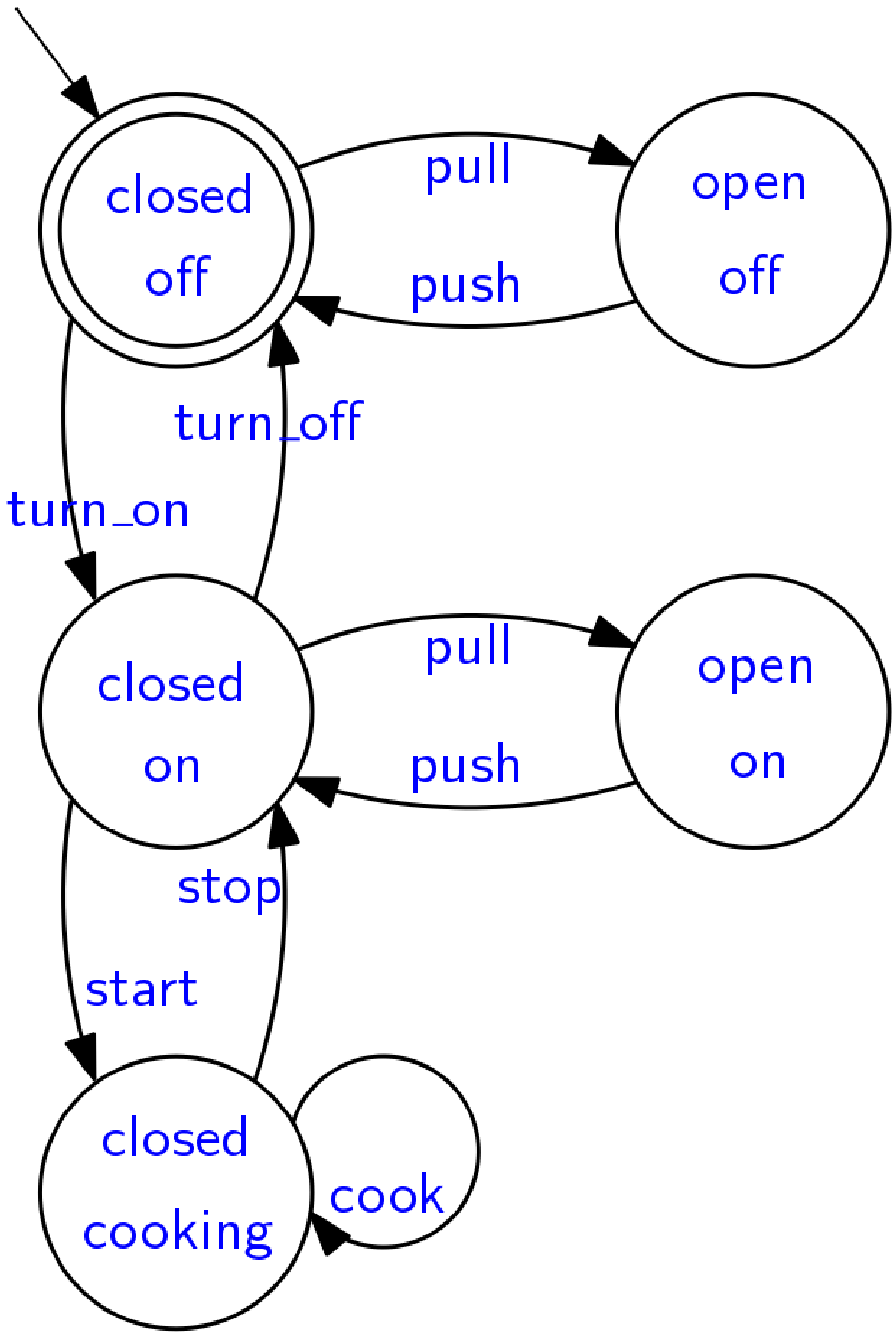
V

<> (turn_off V 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?

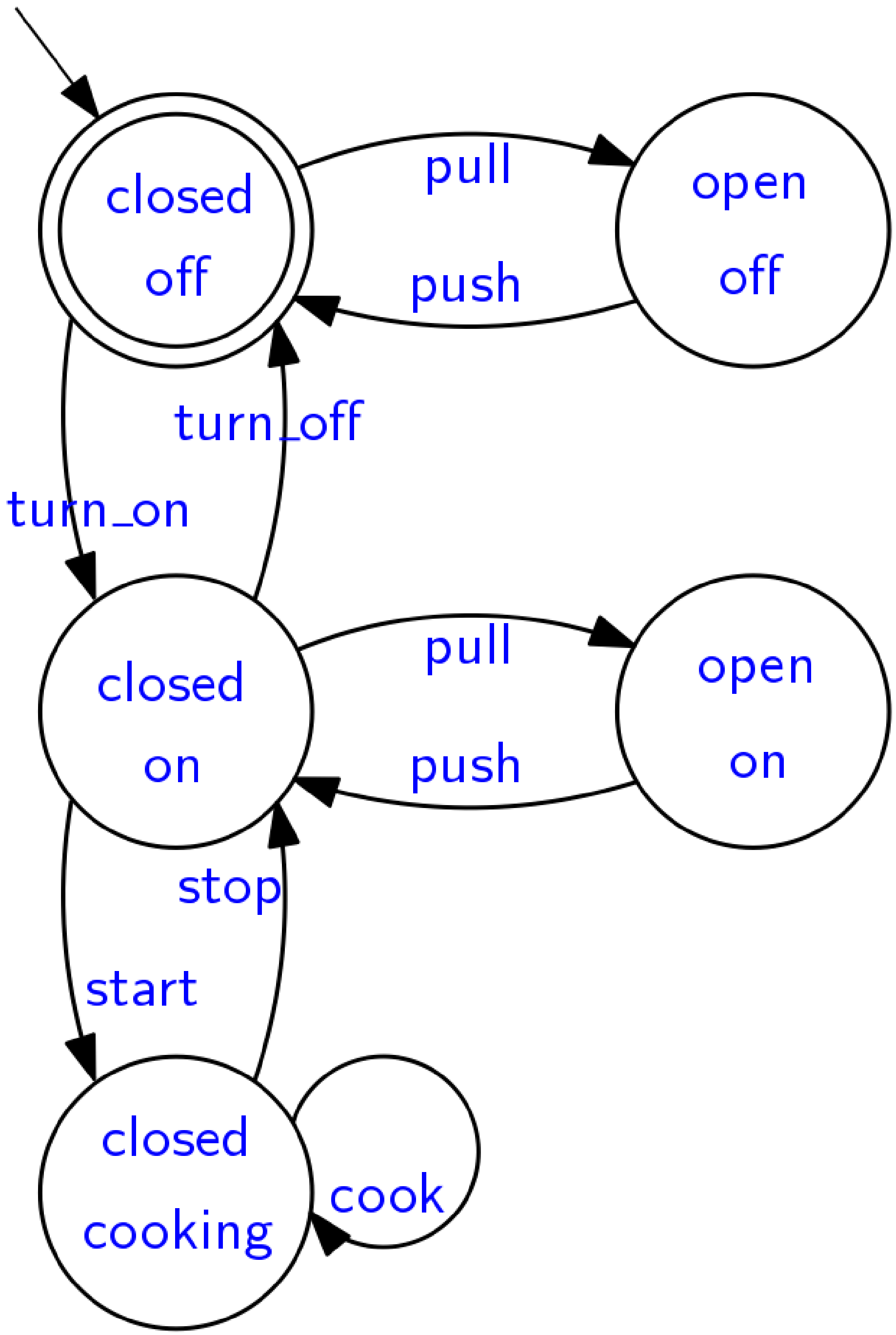


turn_on U start

V

pull U push

Does the property hold?



turn_on U start

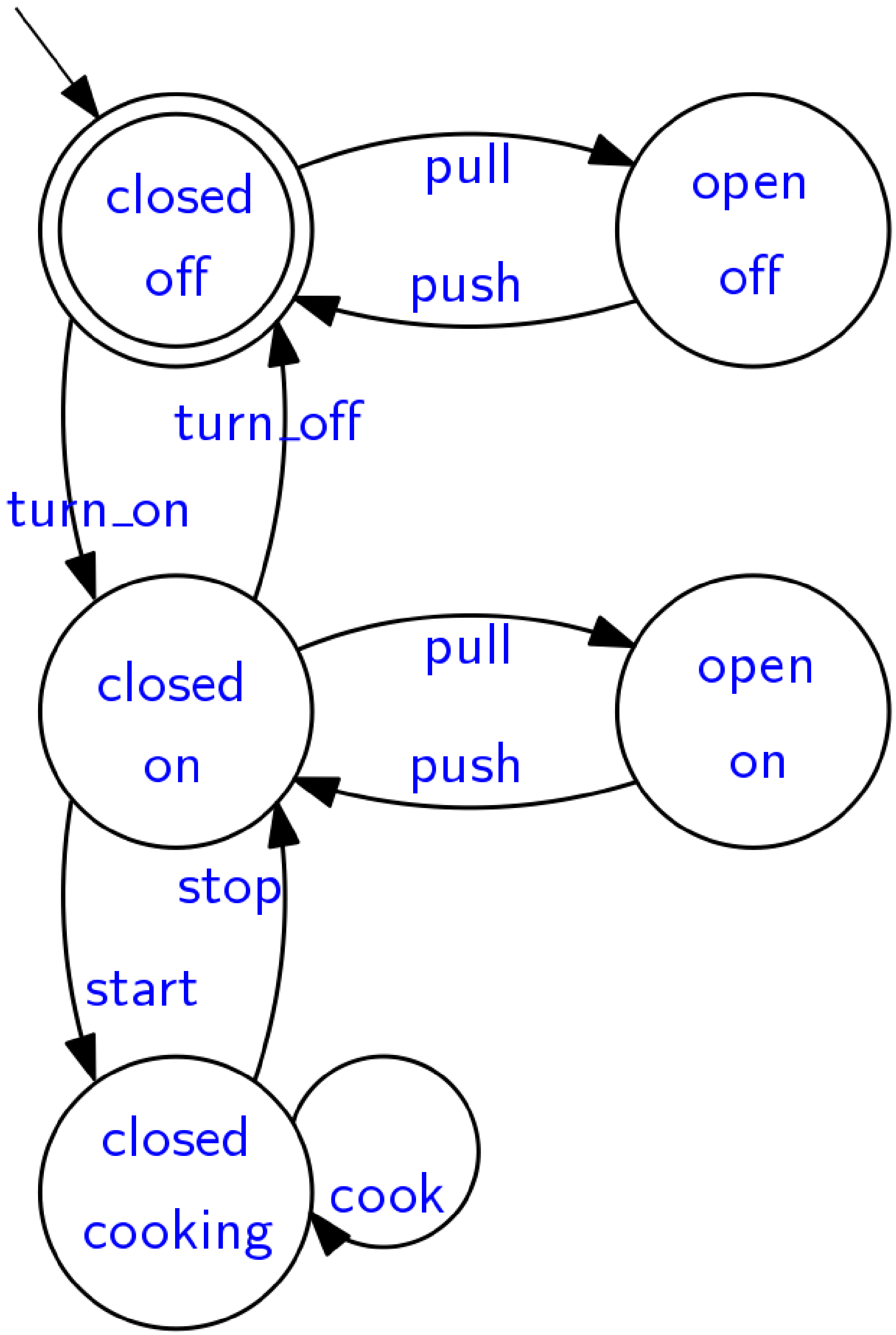
V

pull U push

No:

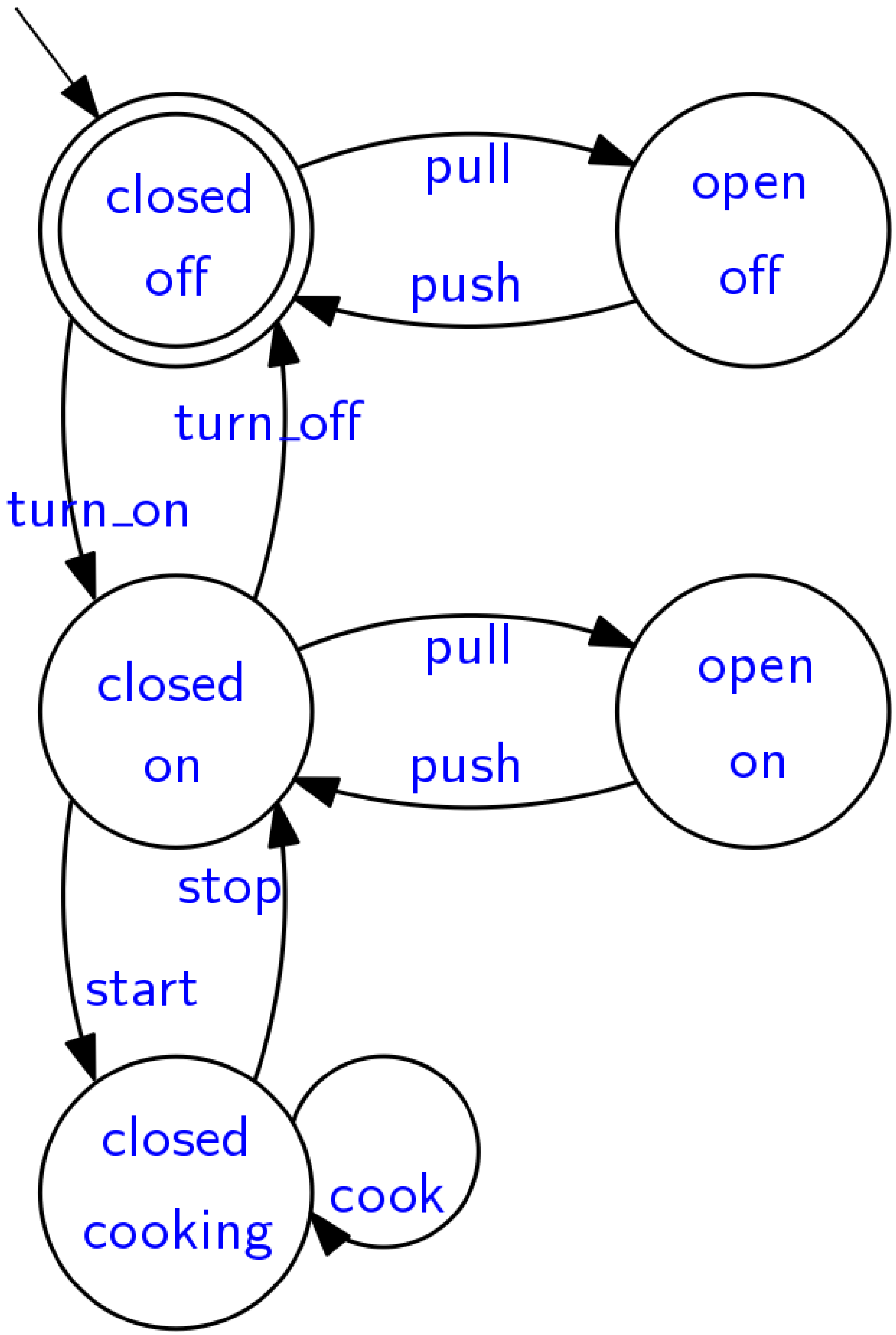
- counterexample: the empty word
- counterexample: turn_on turn_off
- counterexample: turn_on pull push turn_off

Does the property hold?



[] (start ⇒ (cook U <>turn_off))

Does the property hold?



$[\,]$ (start $\Rightarrow$ (cook U <>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 <> is idempotent, that is:

<><> q

is equivalent to:

<> q

Equivalence of formulas

$w,i \models\ <><>\ q$

iff

for some $i \le j \le n$ it is: $w, j \models\ <>\ q$ (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

$w, i \models\ <>\ q$

(semantics of eventually)

Equivalence of formulas

Prove that:

p U <> q

is equivalent to:

<> q

Equivalence of formulas: $\Rightarrow$ direction

w,i ⊨ p U <> q

iff

for some $i \le j \le n$ it is: w, j ⊨ <> q

and for all $i \le k < j$ it is w, k ⊨ p (semantics of until)

implies

for some $i \le j \le n$ it is: w, j ⊨ <> q (a fortiori)

iff

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

(semantics of eventually)

iff

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

(simplification of range of quantification)

iff

w, i ⊨ <> 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

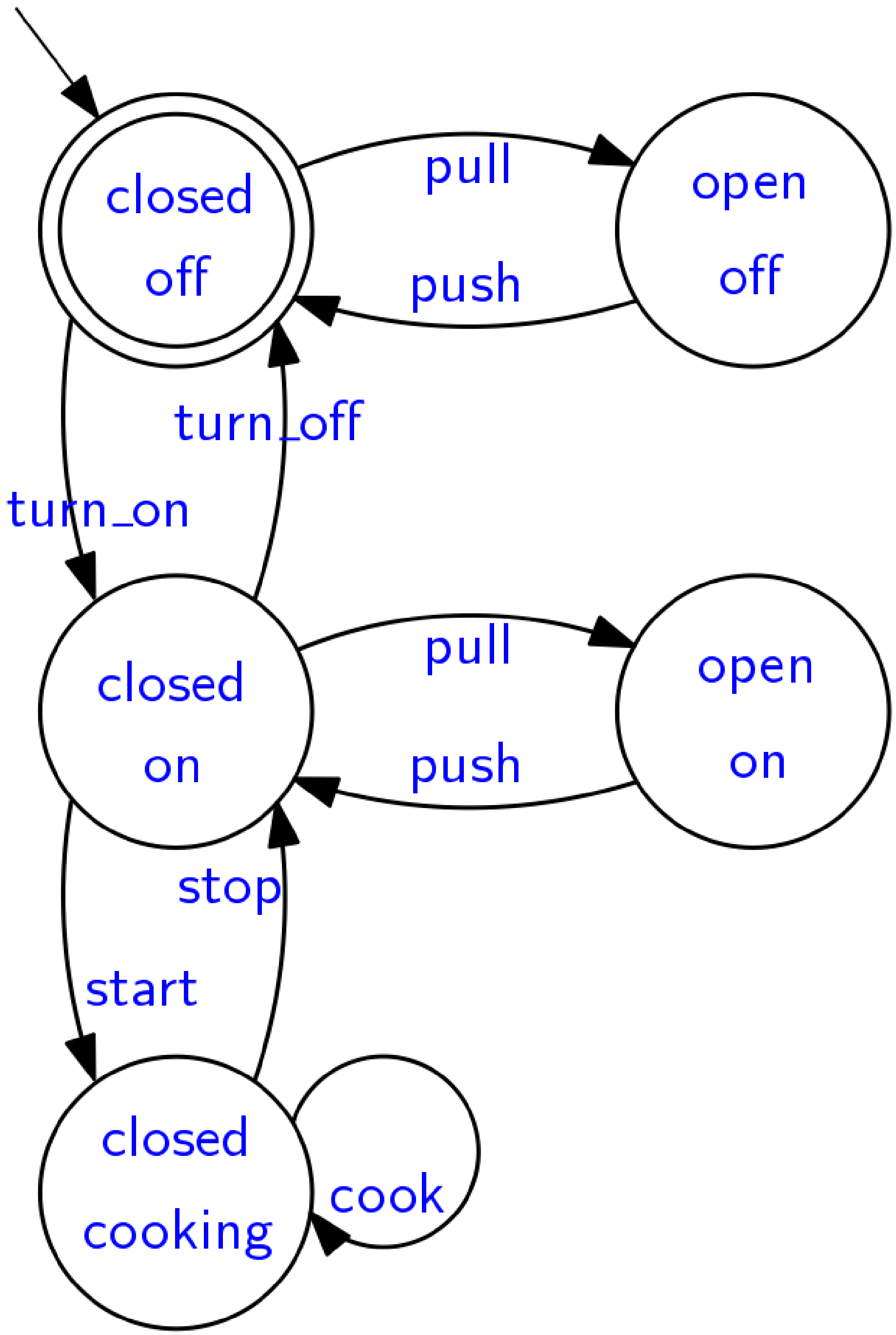
and for all i ≤ k < j it is w, k ⊨ p (a fortiori)

iff

w, i ⊨ p U <> q (semantics of until)

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

Automata-based model checking



[] <> 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:

¬([] <> turn_off)

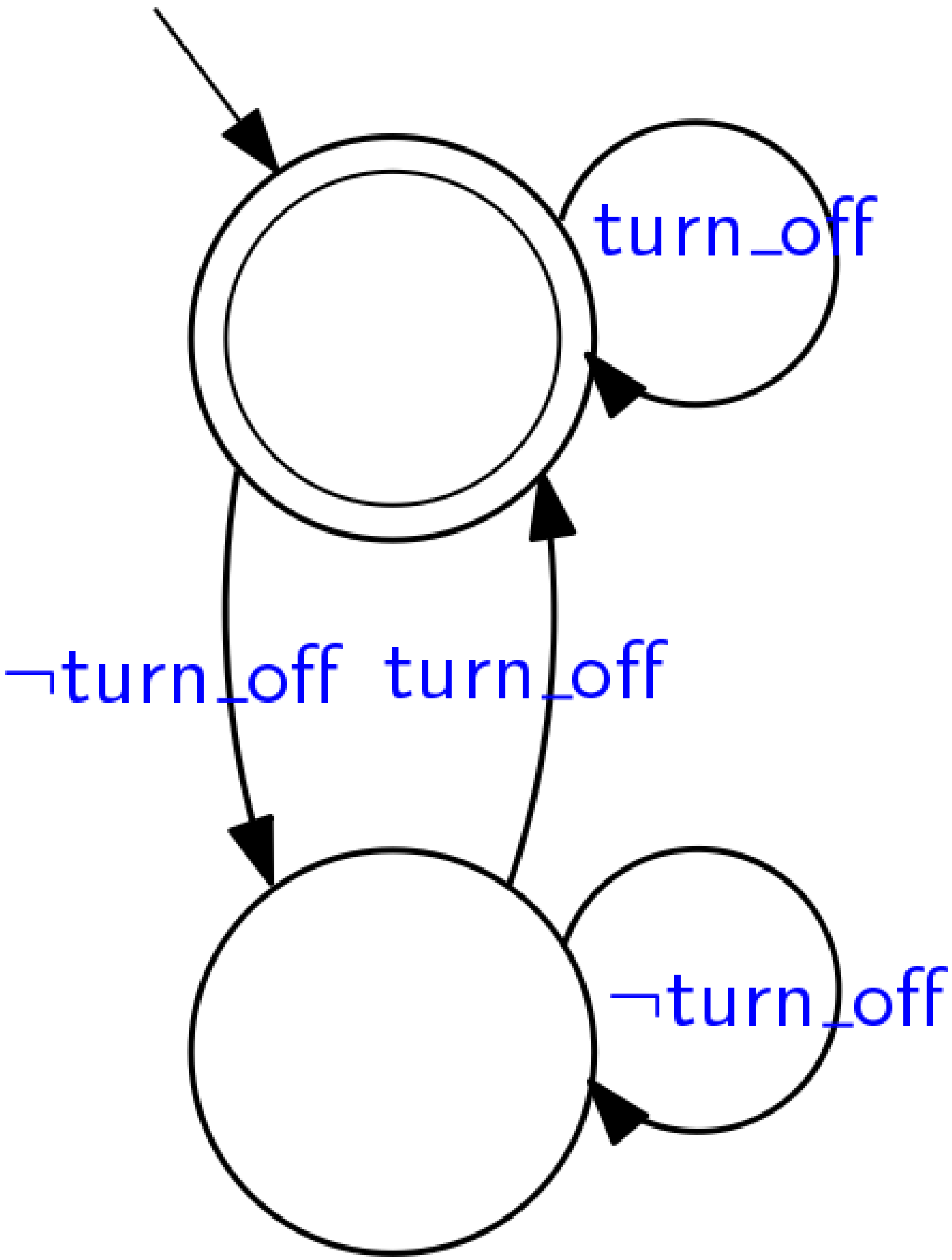
Let us start from the unegated formula:

[] <> turn_off

and then complement the states of the automaton

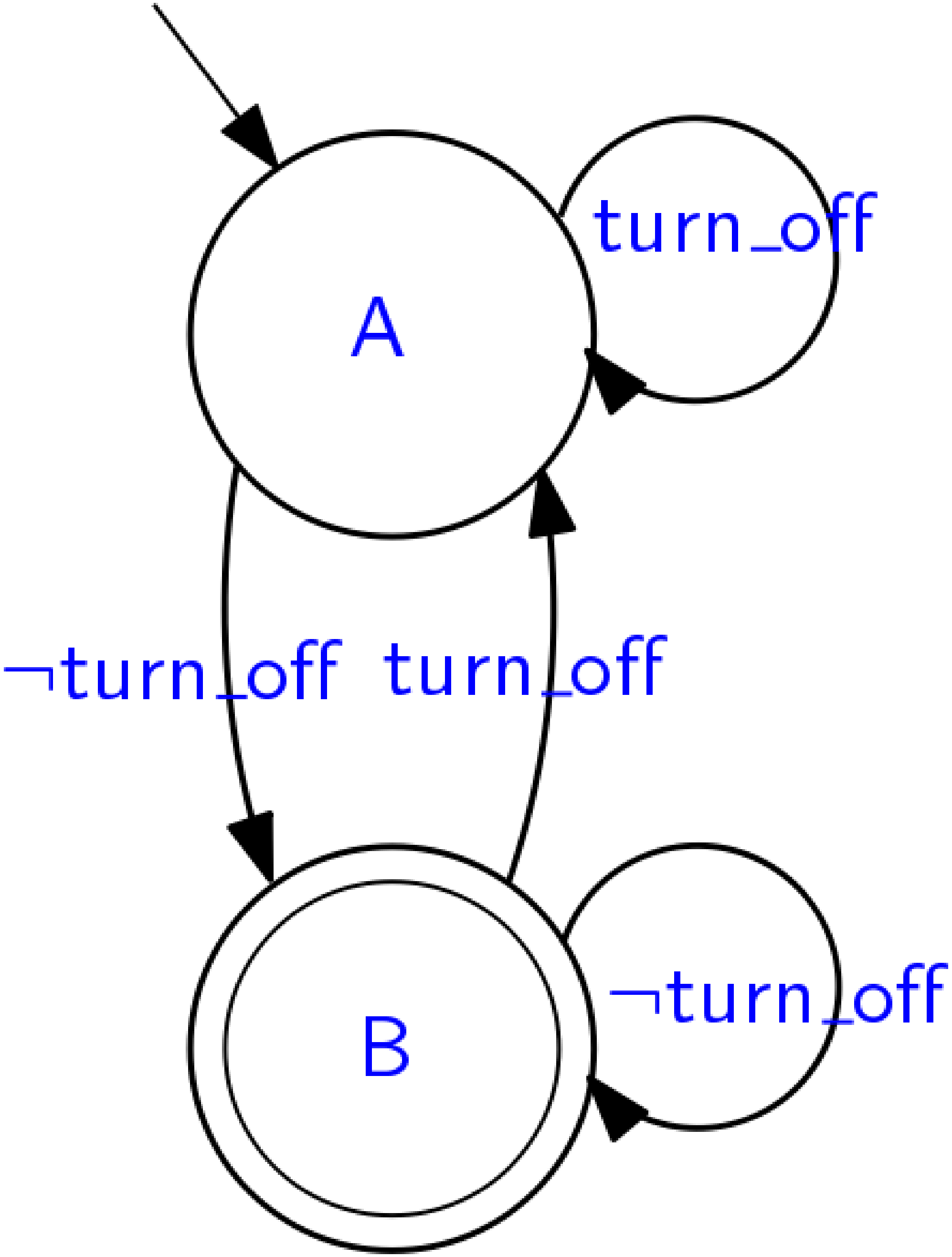
LTL2FSA

[] <> turn_off

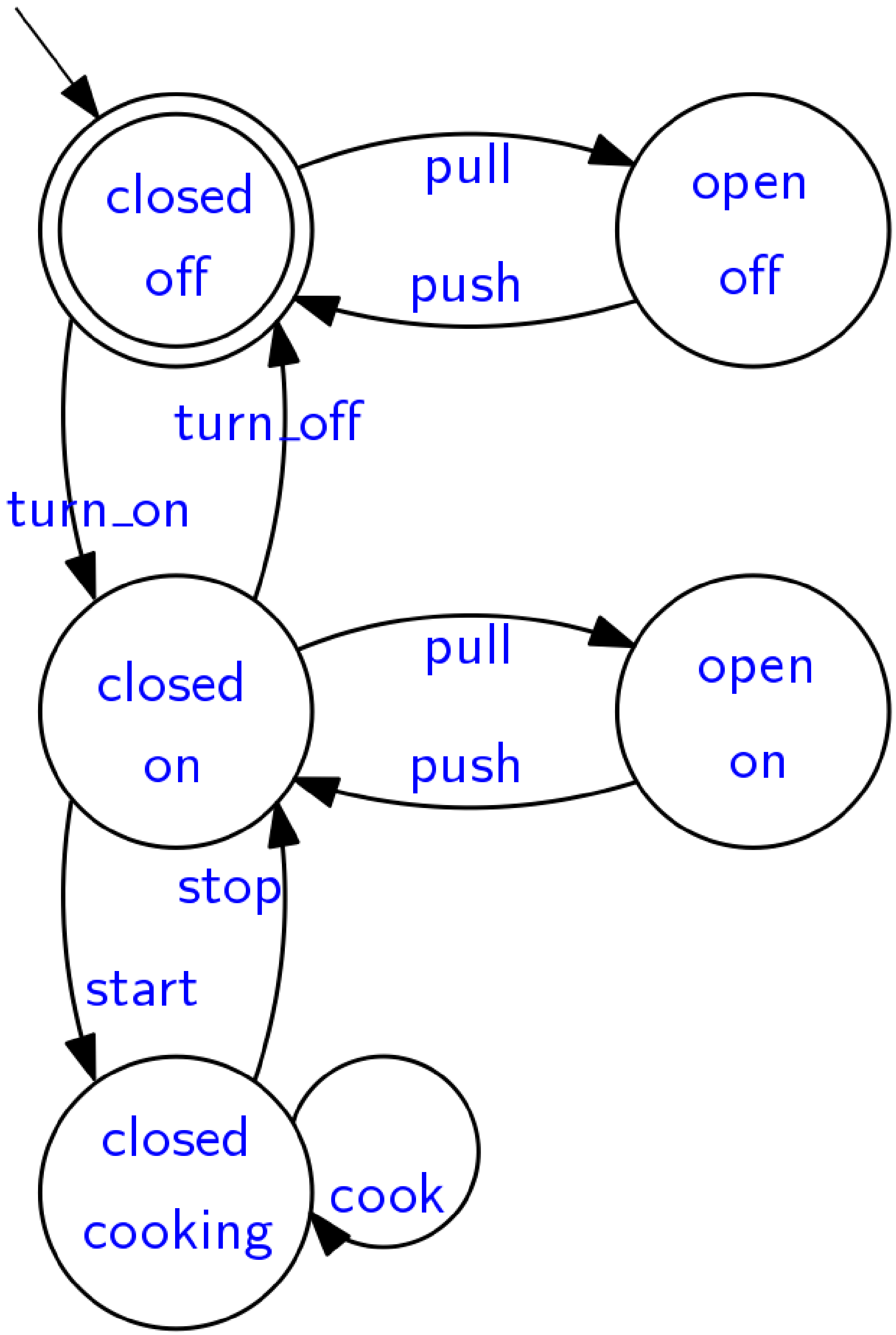


LTL2FSA

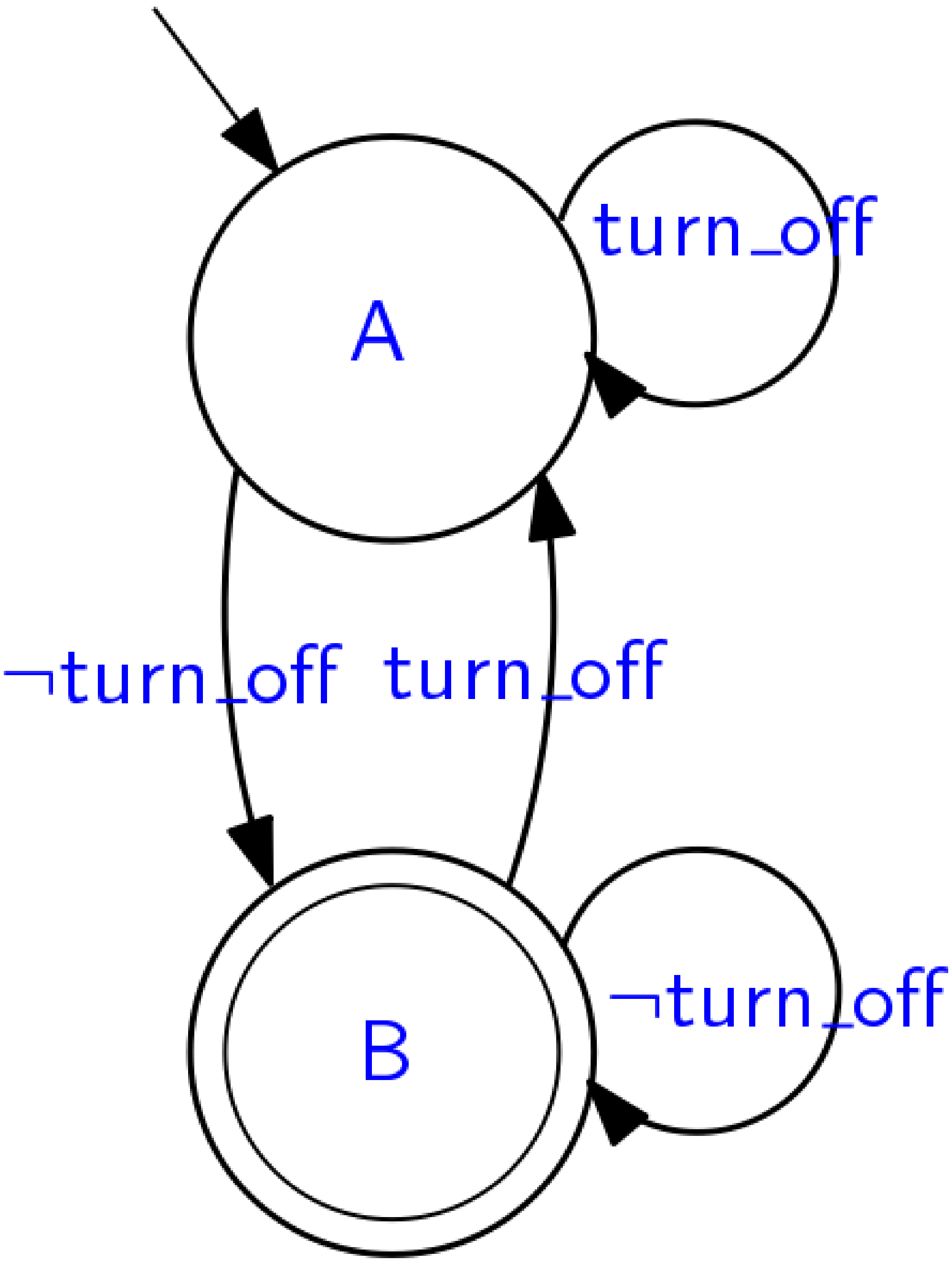
$\neg$([] <> turn_off)



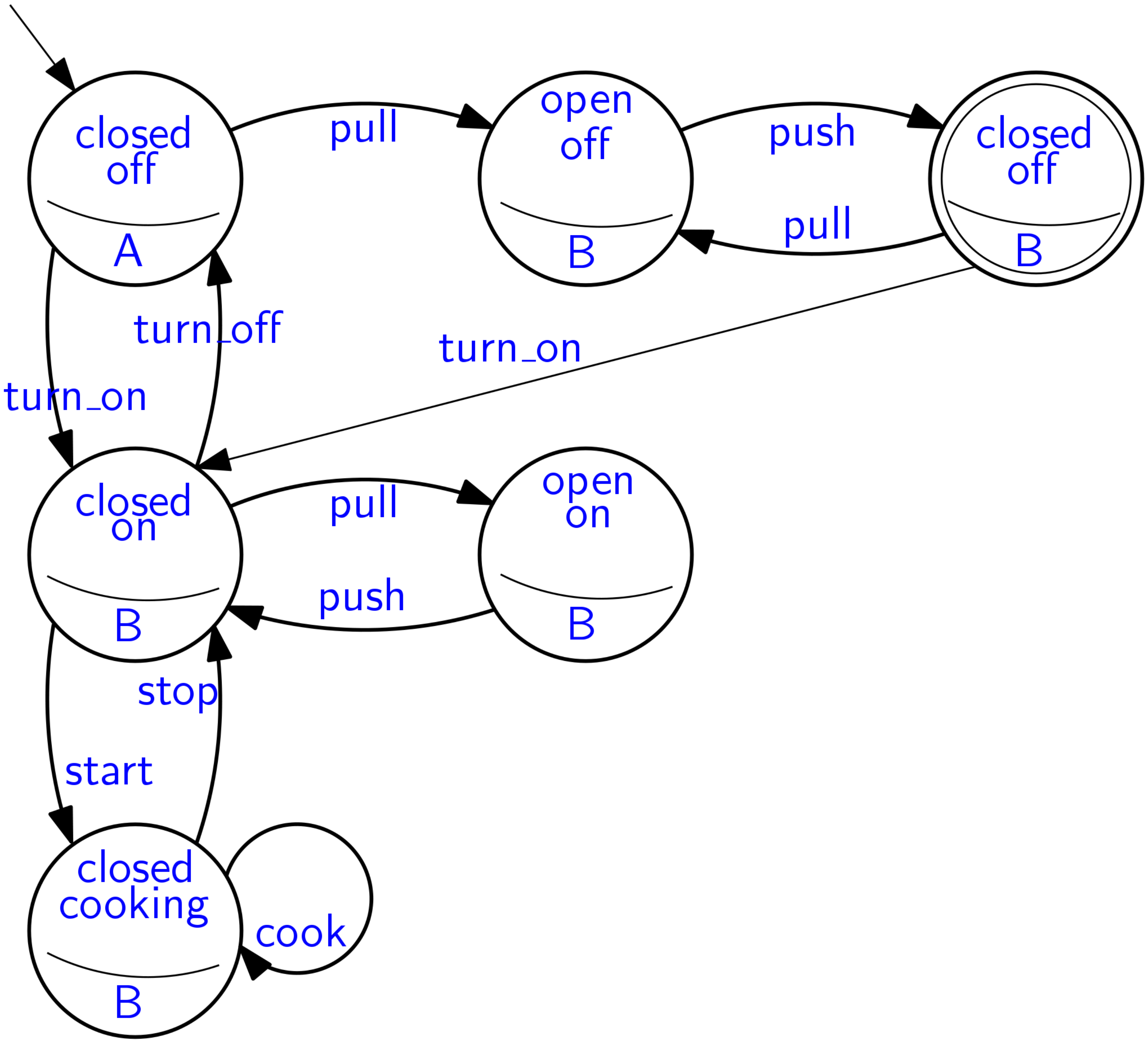
FSA Intersection



×

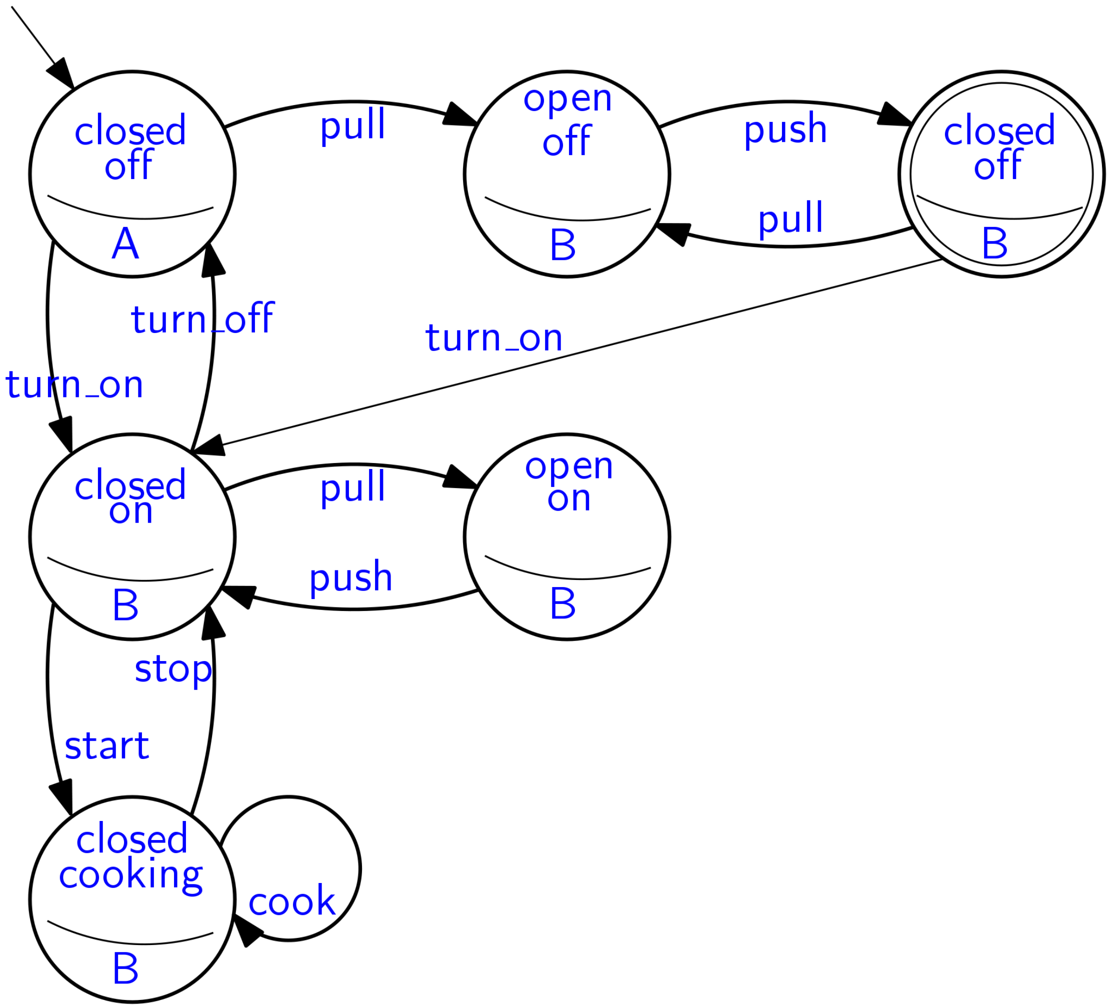


FSA Intersection



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
- ...