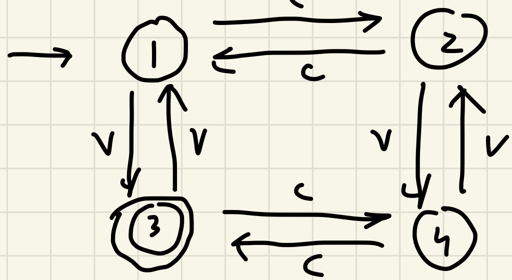
1a.

1 Designing finite-state automata (10 points)

A. Provide a graphical representation for an FSA that has {C, V} as its alphabet and generates all and only those strings that have an odd number of 'V's and an even number of 'C's (treating zero as an even number). Then, define this FSA in Haskell as `fsa_oddEven :: Automaton Int SegmentCV`. It should behave like this: 2+3 points

```
*Assignment02> generates fsa_oddEven [V]
True
*Assignment02> generates fsa_oddEven [V,C,C]
True
*Assignment02> generates fsa_oddEven [V,V,C]
False
*Assignment02> generates fsa_oddEven [V,V,C,V,C]
True
```

a.



2 Myhill-Nerode theorem (3 points)

3 points

Is the Palindrome language $L_{rev} = \{ww^R \mid w \in \{a,b\}^*\}$ regular? How about $L_{ww} = \{ww \mid w \in \{a,b\}^*\}$? Use the Myhill-Nerode theorem to write a proof backing up your thoughts. [1]

The Palindrome language is not regular. An FSA would need to remember an entire string w if it wants to match its reverse. This would need an infinite amount of states and buckets which can't be represented by an FSA which is needed to represent a regular language.

Furthermore, Lww is not a regular language. Similar to the palindrome language, an FSA would need to remember an entire string w to concatenate it with itself. Since this would require an infinite amount of buckets, this language is not regular.

[Optional]: We know the language that generates all and only odd number of 'V's and an even number of 'C's is a regular language (see question 1A in this homework). What are the equivalence classes of prefixes for this language? Do natural languages show this kind of restriction? Extra 2 points

← buckets?

a. a bucket containing strings that have even c's and even v's
b. a bucket containing strings that have odd v's and even c's
c. a bucket containing strings that have even v's and odd c's
d. a bucket containing strings that have odd v's and odd c's

NO, natural languanges do not show this kind of restriction.