Tutorial 5 7.11

1. How to simulate a Turing machine using nondeterministic counter automata with 0-tests?

* WLOG assume that we have only 1s and 0s on the tape (we can encode any alphabet using a two character alphabet)
* simulate a binary stack using two counter automata with 0-tests
* interpret machine's tape as two stacks and simulate it using four automata
* details: click here

* another idea: use one counter to store tape's content, another to remember where machine's head is and the last one as a helper counter

2. For every $m > 0$ create a Petri net of size $O(n+m)$ that is bounded by $F_m(n)$ and not bounded by any smaller number, where

i) $F_1(n) = 2n$

ii) $F_{m+1}(n) = F_m^n(1) = \overbrace{F_m(F_m(\ldots(F_m(1))\ldots))}^{n \text{ times}}$

* we can construct a VASS instead of a Petri net (since the two models are equivalent and we defined how to simulate one with another during the previous tutorial, however, we can create a Petri net straightforward from the VASS)

* for a given n-dimensional VASS (Q, T) with initial configuration (p, u) we define its size as

$$|Q| + |T| + \sum_{i=1}^{n} u(i) + \sum_{t \in T} \sum_{i=1}^{n} |t(i)|$$

* how does a closed-form expression for $F_m(n)$ look like?

$$F_2(n) = F_1(F_1(\dots(F_1(1))\dots)) = 2^n$$

$$F_3(n) = F_2(F_2(\dots(F_2(1))\dots)) = 2^{2^{2^{\cdots^{2}}}} \Big\updownarrow n$$

and so on

* first, compute $F_1(1)$

$$\begin{tikzcd} q_1 \arrow[loop above, "{(2,-1)}"] \end{tikzcd}$$

initial config.

$(q_1, (0,1))$

interpretation: first coordinate stores the result of the current computation, second coordinate is the argument to F_1

thus, for $F_1(n)$ we have

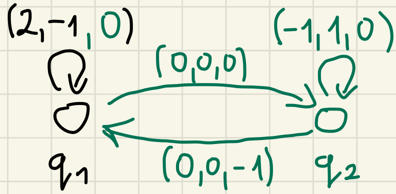
$$\begin{tikzcd} q_1 \arrow[loop above, "{(2,-1)}"] \end{tikzcd}$$

initial config.

$(q_1, (0,n))$

* now, we can build a VASS for $F_2(n)$ using the construction above

* VASS for $F_2(n) = F_1(F_1(\ldots(F_1(1))\ldots))$



initial config.

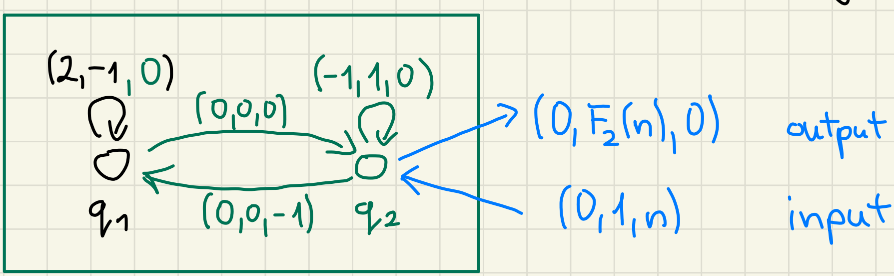
$(q_2, (0,1,n))$

interpretation: two first coordinates serve the same purpose as previously, the third one can be interpreted as the argument for F_2 – the number of times we have to compose F_1 with itself

the initial configuration says that we apply the n-th composition of F_1 to 1

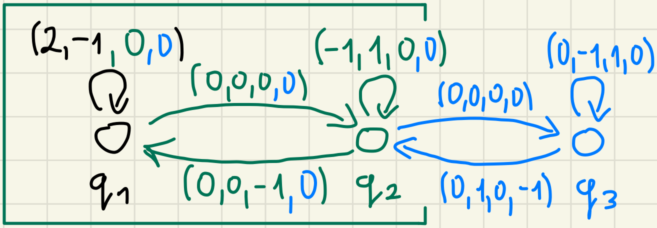
loop in state q_2 updates the parameter for F_1 to the current computation result

when we reach deadlock, the second coordinate stores the final result, hence, we can interpret the construction above as a blackbox computing $F_2(n)$



we can use this observation to construct a VASS computing $F_3(n)$

* "induction" step – VASS for $F_3(n)$



initial configuration: $(q_3, (0,0,1,n))$

now, the configuration above says that we have to apply n-th composition of F_2 to 1

first, we go from q_3 to q_2: we subtract 1 from the fourth coordinate which can be interpreted as the start of the computation of the most inner call of F_2 in $F_3(n) = F_2(F_2(\dots(F_2(1))\dots))$

moreover, we add 1 to the second coordinate to obtain 0, 1, 1 as the prefix of the current configuration – this starts the computation of $F_2(1)$ in the green box;
the computation ends with the prefix $0, F_2(1), 0$ of the configuration in state q_2

then, we go through q_3 and approach green box again with the request to compute F_2 with argument $F_2(1)$ as the conf. prefix is $0, 1, F_2(1)$; the process goes on

* it is easy to see how to continue the construction for $F_4(n)$, $F_5(n)$, ...

3. Prove that in the pessimistic case the maximum size of the coverability tree is not smaller than $Ack(n)$, where n is the size of the net.

* we can assume $Ack(n) \approx F_n(n)$
* formally:

$$Ack(m, n) = \begin{cases} n+1 & m=0 \\ Ack(m-1, 1) & m>0,\ n=0 \\ Ack(m-1, Ack(m, n-1)) & m>0,\ n>0 \end{cases}$$

$m \backslash n$	0	1	2	3	4	n
0	1	2	3	4	5	$n+1$
1	2	3	4	5	6	$n+2 = 2+(n+3)-3$
2	3	5	7	9	11	$2n+3 = 2\cdot(n+3)-3$
3	5	13	29	61	125	$2^{(n+3)}-3$
4	13 $= 2^{2^2}-3$	65533 $= 2^{2^{2^2}}-3$	$2^{65536}-3$ $= 2^{2^{2^{2^2}}}-3$	$2^{2^{65536}}-3$ $= 2^{2^{2^{2^{2^2}}}}-3$	$2^{2^{2^{65536}}}-3$ $= 2^{2^{2^{2^{2^{2^2}}}}}-3$	$\underbrace{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}_{n+3}-3$

* we observe that the proof follows the construction presented for the previous problem

4. Improve Lipton's construction presented during the lecture to avoid the exponential blowup of the instruction number.

* consider $Dec_{m+1}(x)$ and think about how to improve it (during tutorials), then analyze the whole construction once again (homework)

$Dec_{m+1}(x)$:

loop

$a_m{+}{+},\ \hat{a}_m{-}{-}$

loop

$b_m{+}{+},\ \hat{b}_m{-}{-}$

$\boxed{x{-}{-},\ \hat{x}{+}{+}}$

$Dec_m(b_m)$

$Dec_m(a_m)$

we would like to have one universal Dec_i for all possible calls

if there is one gadget for Dec_i, how does it know what to increase?

how does it know where to return?

reverse the side-effect on a_m, b_m and ensure that

this part is repeated $2^{2^m} \cdot 2^{2^m} = 2^{2^{m+1}}$ times

$\mathrm{Dec}_{m+1}(x)$:

new variables with dashes representing the flags / semaphores for previously defined variables

loop

- $a_m{+}{+},\ \hat{a}_m{-}{-}$
- loop
  - $b_m{+}{+},\ \hat{b}_m{-}{-}$
  - $\boxed{x{-}{-},\ \hat{x}{+}{+}}$
- $\mathrm{Dec}_m(b_m)$

$\mathrm{Dec}_m(a_m)$

$\bar{x}{-}{-}$
$x{-}{-},\ \hat{x}{+}{+}$ ← used only when $\bar{x} = 1$
$\bar{x}{+}{+}$

or

$\bar{y}{-}{-}$
$y{-}{-},\ \hat{y}{+}{+}$ ← used only when $\bar{y} = 1$
$\bar{y}{+}{+}$

or

...

$$c_i{+}{+},\ \bar{b}_m{+}{+},\ \mathrm{Dec}_m(b_m),\ \bar{b}_m{-}{-},\ c_i{-}{-}$$

$c_i{+}{+}$ can be interpreted as a call statement and $\bar{b}_m{+}{+}$ as a parameter for the called function (it informs that some action have to be performed on b_m)

$c_i{-}{-}$ asserts that the execution goes back to the right place

* it remains to work on the details

* we update the other functions in a similar way