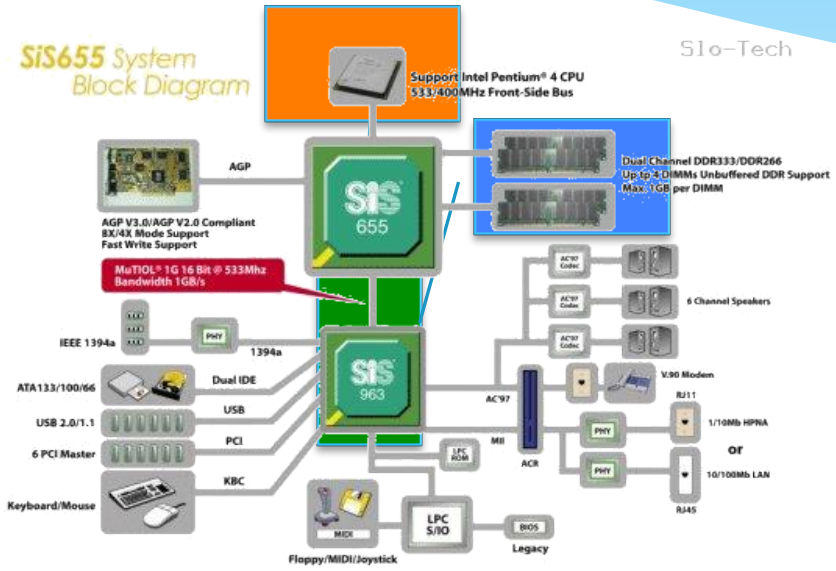
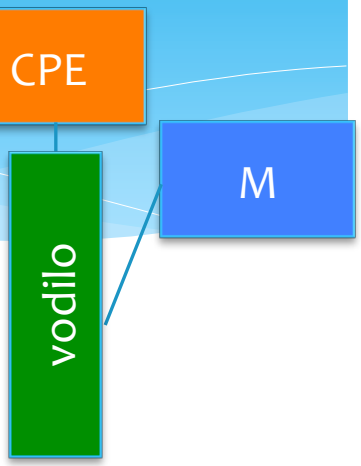
Problem

Peter Zmeda uči programiranje in kot domačo nalogo daje običajno pisanje takšnih in drugačnih problemov. Domače naloge mora seveda popraviti.

Da bi pospešil preverjanje domačih nalog, se je odločil napisati program, ki bo preveril ali je oddani program pravilen.

Torej kako izgleda računalnik





Neprestano izvaja naslednje korake:

1. prevzemi naslednji ukaz
2. prevzemi podatke za izvedbo ukaza
3. izvedi ukaz
4. shrani rezultate ukaza

Abeceda, beseda, jezik

* abeceda je **končna množica** črk Σ:
  * {A, B, C, ..., Ž}, {O, 1}, {A, C, G, T}, ...
* iz črk tvorimo besede kot zaporedja črk:
  * “Famnitovi izleti v matematično vesolje”, 10110
  * množica besed je neskončna
* množica dovoljenih besed predstavlja jezik

Mlinar, volk, koza in zelje

Mlinar pride do rečnega brega s kozo, volkom in zeljem. Vse tri mora spraviti čez reko s pomočjo čolna, ki ga najde na rečnem bregu. Žal čoln ni velik in se lahko z njim pelje poleg mlinarja, ki vesla, še samo ena žival ali zelje. Dodatni zaplet je to, da mlinar ne sme pustiti smaega zelja s kozo, ker ga koza poje in tudi ne volka s kozo, ker jo potem volk požre.

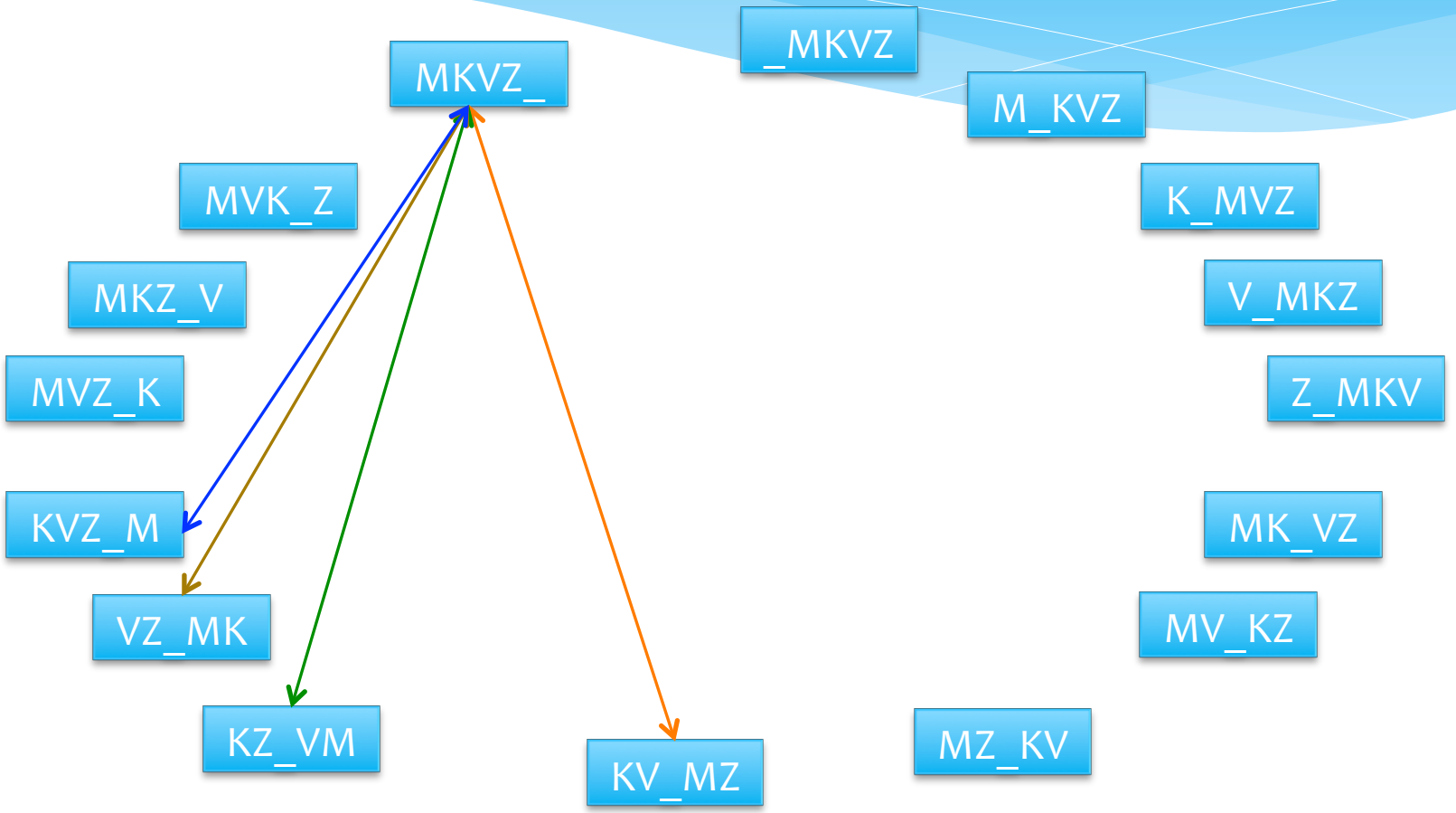
Kakšno je zaporedje voženj, ki naj jih opravi mlinar, da prepelje vse tri preko varno preko rek?

Lahko sistematično poiščemo odgovor?

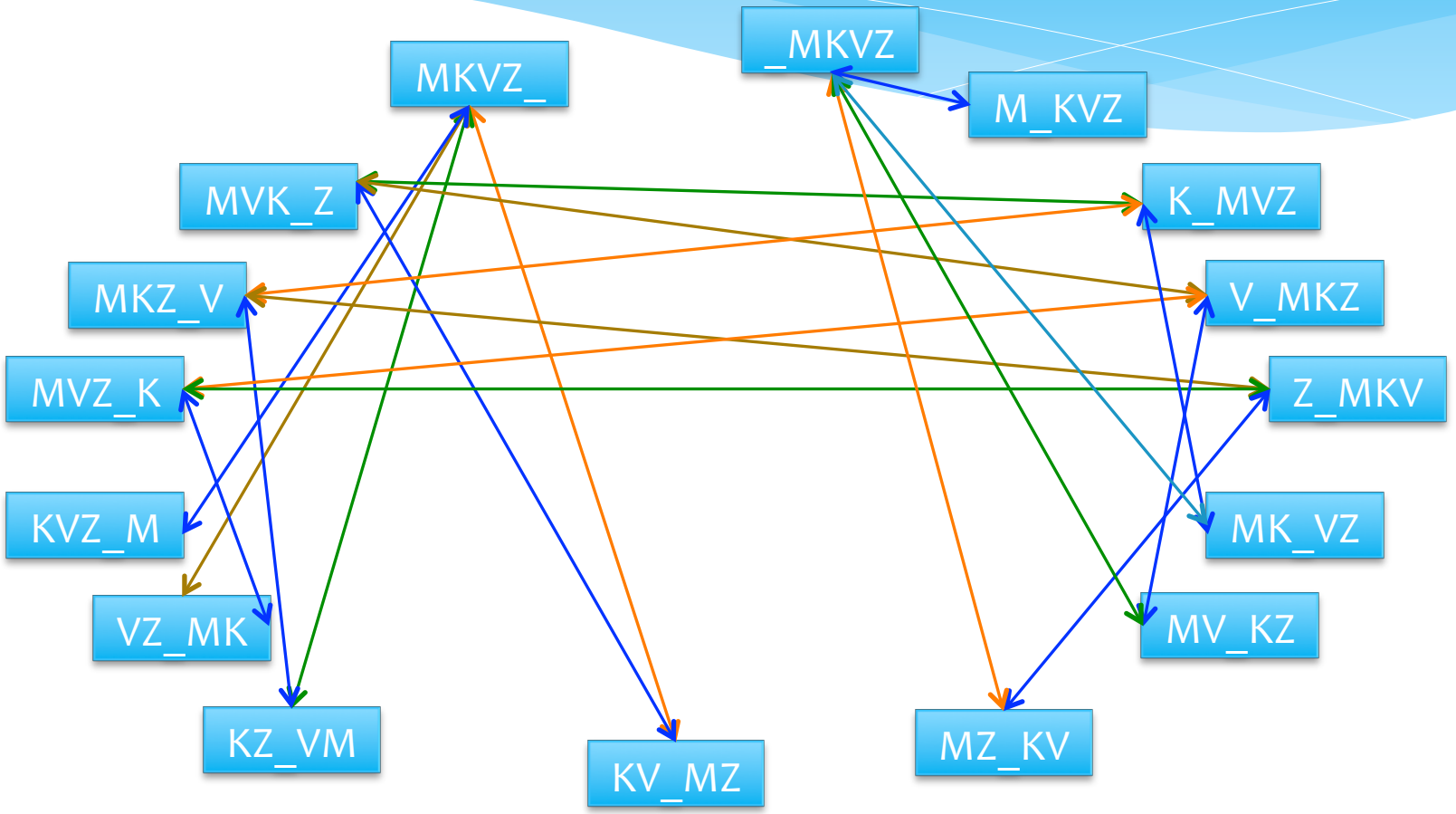
Mlinar, volk, koza in zelje

* označimo stanje s črkami M(linar), K(oza), Z(elje), V(olk) in _, kar predstavlja reko
  * na primer: MKVZ_, MK_VZ ali –MKVZ
* vsa možna stanja so
  * MKVZ_
  * MKV_Z, MKZ_V, MVZ_K, VZK_M
  * MK_VZ, MV_KZ, MZ_KV, KV_MZ, KZ_MV, VZ_MK
  * M_KVZ, K_MVZ, V_MKZ, Z_MVK
  * _MVKZ

Mlinar, volk, koza in zelje



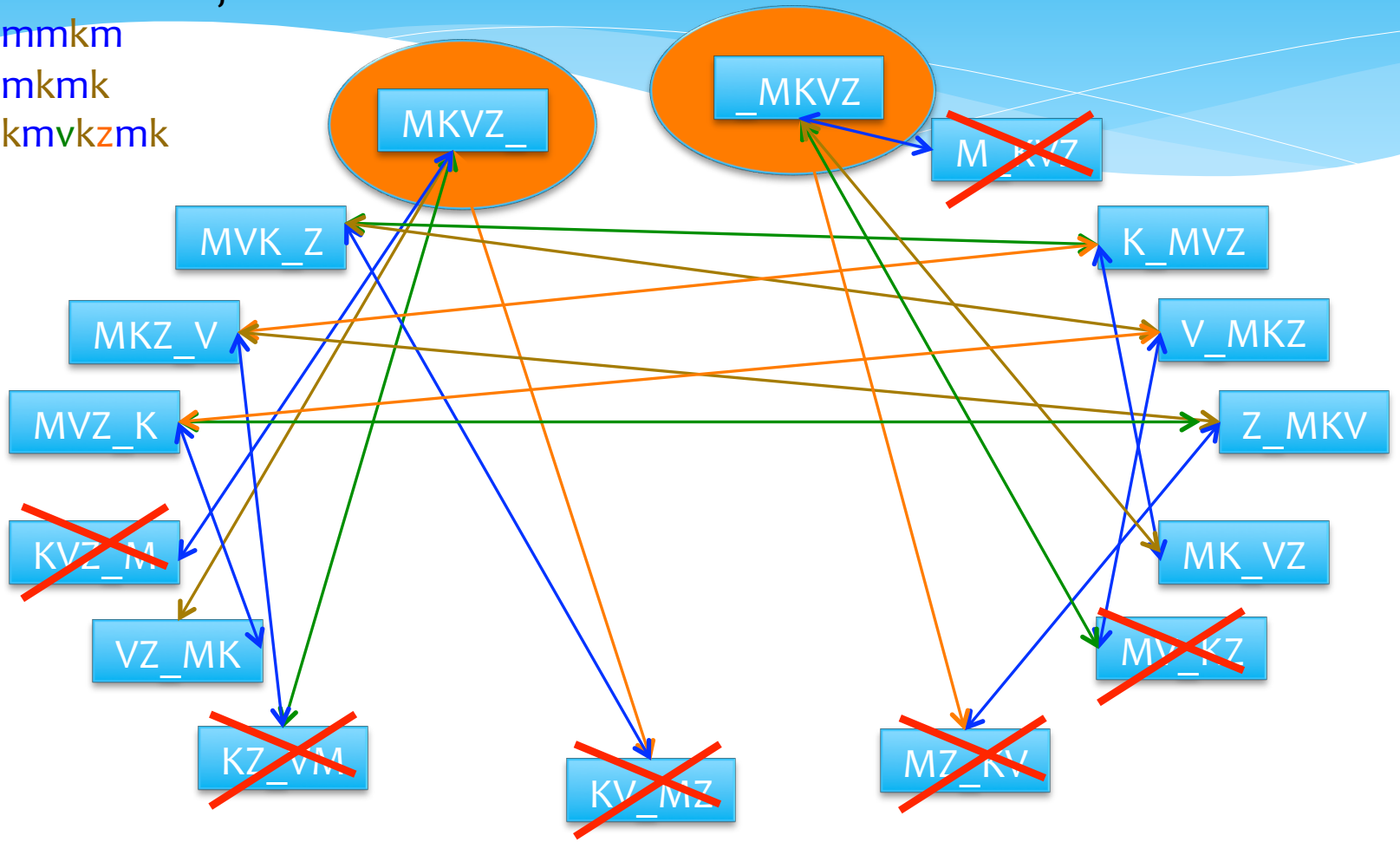
Mlinar, volk, koza in zelje



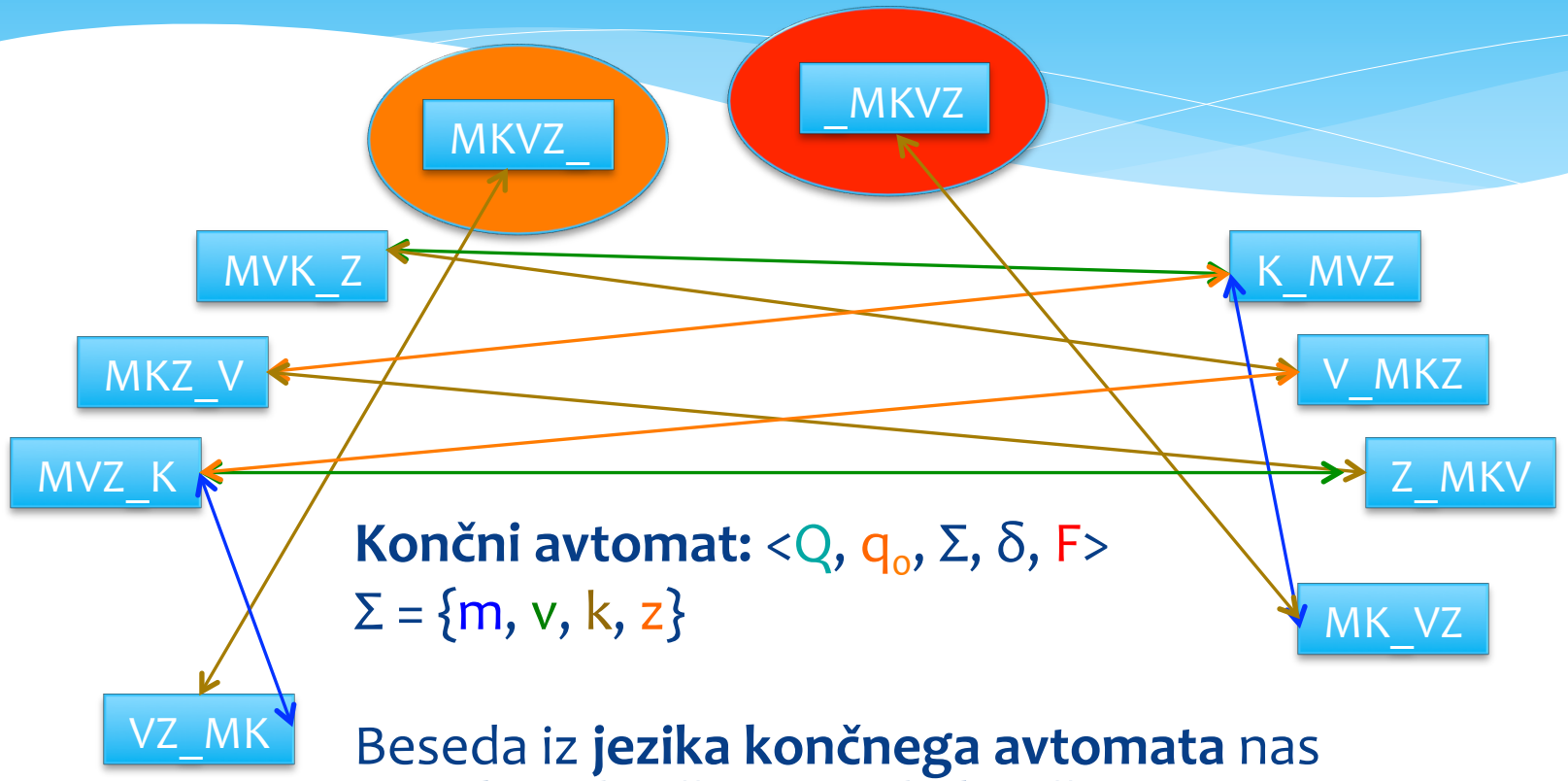
Mlinar, volk, koza in zelje

Imamo naslednje besede:

- mmkm
- mkmk
- kmvkzmk



Mlinar, volk, koza in zelje

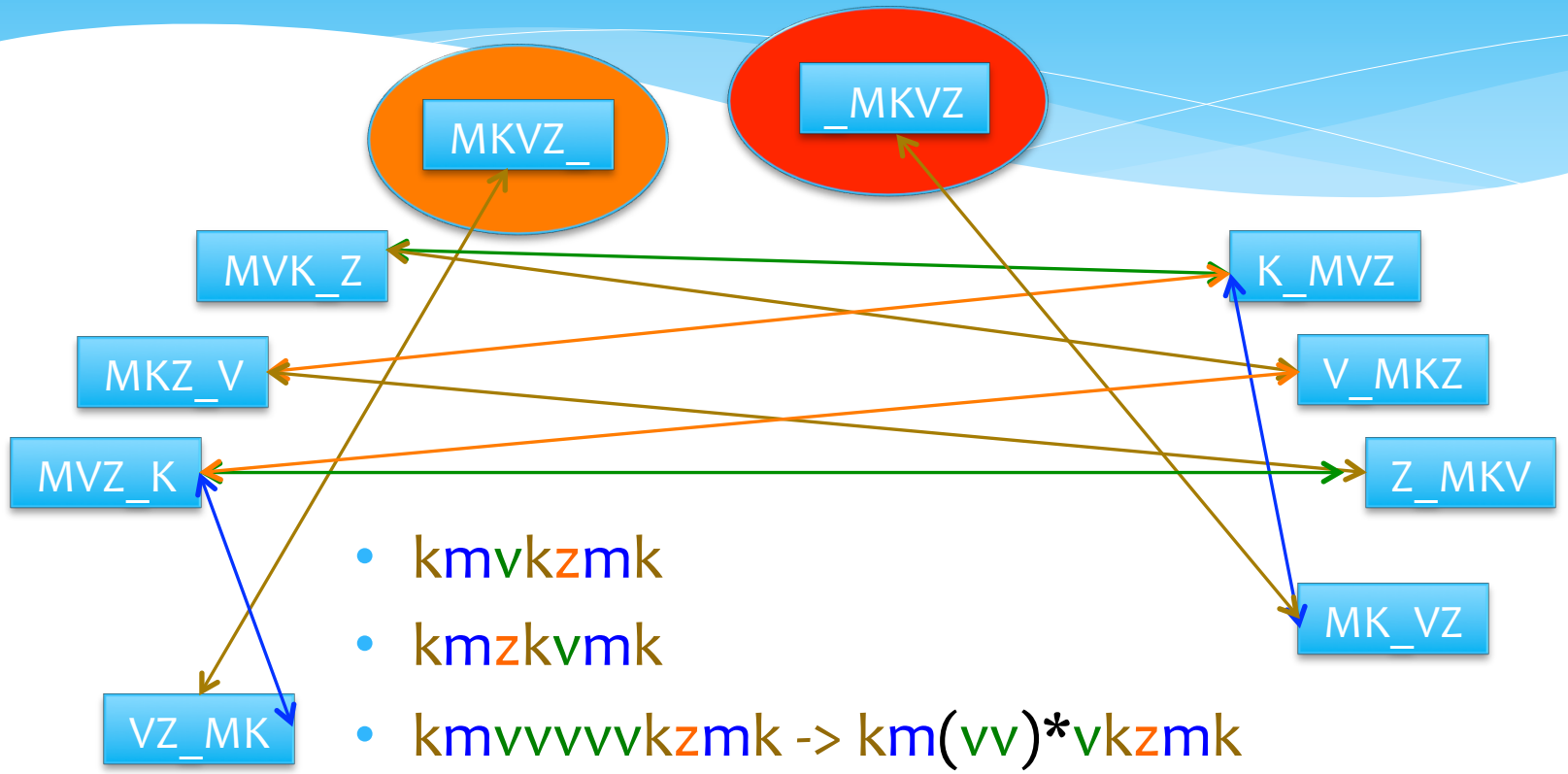


Končni avtomat: $<Q, q_0, \Sigma, \delta, F>$

$\Sigma = \{m, v, k, z\}$

Beseda iz **jezika končnega avtomata** nas pripelje od začetnega do končnega stanja.

Mlinar, volk, koza in zelje

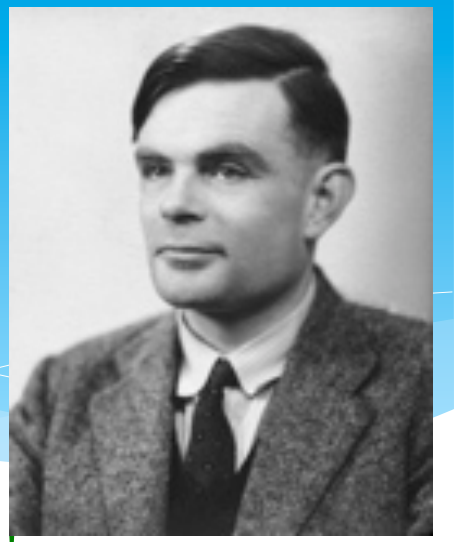


- kmvkzmk
- kmzkvmk
- kmvvvvvkzmk -> km(vv)*vkzmk
- km(zkvzkv)*vkzmk

Končni avtomat

* opis končnega avtomata je **končen**, a število besed njegovega jezika je lahko **neskončno**
* končni avtomat nastopa kot **razpoznavalnik** svojega jezika
  * če se ustavi v končnem stanju, beseda je v jeziku, sicer ni
  * kot, če bi rekel na koncu **DA** ali **NE**
* jeziki končnega avtomata se imenujejo **regularni jeziki**
* obstajajo jeziki, ki niso regularni – $0^n1^n0^n$

Turingov stroj



Alan Turing (1912-1954), angleški matematik

* formaliziral pojem algoritma in programa
* oče računalništva in informatike
* ACM podeljuje Turingove nagrade (Nobelova nagrada za računalništvo in informatiko)

* Turingov stroj deluje kot končni avtomat, le da ima še pomnilnik
  * zato si lahko zapomni število prebranih ničel v jeziku $0^n1^n0^n$
* tudi Turingov stroj je razpoznavalnik jezika
* ***celo več, Turingov stroj je matematični model dandanašnjih računalnikov (programov)***
* Turingov stroj (računalnik) lahko reši **izračunljive probleme** – ali obstajajo neizračunljivi problemi?

Problemi in jeziki

* problem sestoji iz primerkov (instanc)
  * problem iskanja največjega števila: [1, 5, 3, 6] in 6
* vsak problem lahko predstavimo kot jezik
  * jezik problema iskanja največjega števila je par: zaporedje števil in največje število

    "[1, 5, 3, 6], 6" je v jeziku

    "[1, 5, 3, 6], 3" ni v jeziku
* reševanje vsakega problema lahko zapišemo kot reševanje vprašanja članstva v jeziku in obratno
* *ponovno: ali obstajajo problemi/jeziki, za katere ne obstaja Turingov stroj / jih ne more izračunati računalnik?*

Kako izgledajo programi

izvorna koda:

```
#include <stdio.h>
int main(int argc, char* argv[]) {
  printf ("Dober dan!\n");
}
```

shranjeno kot (83 zlogov):

```
00000000: 2369 6e63 6c75 6465 203c 7374 6469 6f2e
00000010: 683e 0a69 6e74 206d 6169 6e28 696e 7420
00000020: 6172 6763 2c20 6368 6172 2a20 6172 6776
00000030: 5b5d 2920 7b0a 2020 7072 696e 7466 2028
00000040: 2244 6f62 6572 2064 616e 215c 6e22 293b
00000050: 0a7d 0a
```

Vsak program je samo številka

* naš program je **številka** manjša od $256^{83} = 2^{664}$
* **vsak program je samo številka in obratno ter vsak podatek je številka in obratno**
* vendar:
  * nekatere številke ne predstavljajo pravilnega programa; v tem primeru naj bo njihov jezik prazen
  * nekateri programi imajo dve enaki številki; nič hudega, njuna jezika sta enaka

```
#include <stdio.h>
int main(int argc, char* argv[]) {
  printf ("Dober dan!\n");
}
```

```
#include <stdio.h>
int main(int argc,
         char* argv[]) {
  printf ("Dober dan!\n");
}
```

Čudni jezik L_d

* ker je vsaka številka x lahko hkrati podatek x_p (=x) in program x_T (=x), se lahko vedno vprašamo ali je x_p v jeziku programa $L(x_T)$: $x_p \in L(x_T)$
* podobno lahko definiramo jezik tistih opisov programov, za katere velja:

$$L_d = \{x_T : x_T \notin L(x_T)\}$$

* kako izgleda program P_d za razpoznavo L_d?

Neizračunljivi problem L_d

$$L_d = \{x_T : x_T \notin L(x_T)\}$$

* Recimo:
  * da obstaja program P_d, ki razpozna L_d => potem je tudi P_d neka številka.
* Ali je $P_d \in L(P_d)$?
  * Recimo, da $P_d \in L(P_d)$ =>
  * **$P_d \notin L_d$**, toda $L_d = L(P_d)$ in zato **$P_d \in L(P_d)$** xxx.
  * Potem pa recimo $P_d \notin L(P_d)$ =>
  * **$P_d \in L_d$**, toda ker $L_d = L(P_d)$ zato **$P_d \notin L(P_d)$** xxx.
* **Torej P_d ne obstaja in problem članstva v jeziku L_d ni izračunljiv!**

Problem zaustavitve

* Problem zaustavitve:
  * *imamo program P () in poljuben niz w:*

    ali se računanje P(w) zaustavi?
* imejmo program:

```
function H(x) {
// x je hkrati program in podatek!!
  if x(x) == "NE" return "NE";
  else
    while ( true ) { }
}
```

* Ker imamo program ***H()***, tudi problem zaustavitve v splošnem ni izračunljiv => ne obstaja program $T_u(P())$, ki bi preveril ali se ***P(w)*** ustavi za vsak parameter ***w***.

Petrov problem

Da bi pospešil preverjanje domačih nalog, se je [Peter Zmeda] odločil napisati program, ki bo preveril ali je oddani program pravilen.

* Peter se domisli, da bi napisal program ***preveri()***, ki bo preverjal pravilnost dijaških programov ***S()*** tako, da preverjal njihovo enakovrednost z njegovim programom ***P()***:

preveri (P(), S())

* Ali obstaja ***preveri()*** za katerikoli ***P()*** in ***S()***?

Petrov problem

* Recimo, da ***preveri()*** obstaja.
* Naj bo:

```
function vednoDa(x) { return "DA"; }
```

* potem lahko naredimo funkcijo, ki preverja ustavljivost programa *x()*:

```
function aliSeUstavi( x() ) {
  y() = "function y(inp) {
           x(inp); return "DA";
         }";
  return preveri(y(), vednoDa());
}
```

Petrov problem

* Naša funkcija ***aliSeUstavi(P())*** je program, ki preverja ali se program ***P()*** ustavi za vsak **w**;
* vendar smo prej ugotovili:

  ne obstaja program T_u***(P())***, ki bi preveril ali se ***P(w)*** ustavi za vsak parameter ***w;***
* zatorej ***aliSeUstavi()*** ne obstaja, oziroma posledično ne obstaja ***preveri(P(), S())***.

Nerešljivi problem

Peter Zmeda uči programiranje in kot domačo nalogo daje običajno pisanje takšnih in drugačnih problemov. Domače naloge mora seveda popraviti.

Da bi pospešil preverjanje domačih nalog, se je odločil napisati program, ki bo preveril ali je oddani program pravilen.

V splošnem ...

Še en problem ...

Petrov prijatelj je avtoprevoznik in razvaža po krajih v Sloveniji dobrine. Zjutraj se odpravi iz Kopra in se zvečer vrača nazaj v Koper.

Da bi prijatelju zmanjšal stroške, želi Peter naračunati vrstni red krajev, kamor mora prijatelj dostaviti dobrine, tako, da: (i) se bo v vsakem kraju ustavil samo enkrat; in (ii) da bo skupna dolžina poti čim krajša.

https://lusy.fri.uni-lj.si/redmine/projects/cs-edu/documents