Uvod v računalništvo (UvR)

Računski modeli

Danijel Skočaj
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Literatura: Invitation to Computer Science, poglavje 12

v1.0
Št. leto 2013/14

Cilji predavanja

- Razumeti namen konstruiranja modelov
- Opisati komponente Turingovega stroja
- Razumeti zakaj je Turingov stroj dober model računskega agenta
- Razumeti relacijo med algoritmom in programom Turingovega stroja
- Simulirati delovanje Turingovega stroja
- Napisati preproste Turingove stroje, ki rešujejo enostavne probleme
- Opisati pomen Church-Turingove teze
- Razumeti kaj pomeni nerešljiv problem

Uvod

- Obstajajo problemi za katere ne obstaja algoritmična rešitev
  - Lahko dokažemo, da ne more obstajati algoritem, ki bi lahko rešil dan problem
- Model računskega agenta
  - ohranimo samo najbolj pomembne lastnosti oz. operacije
  - „idealni" računski agent
- Model
  - zajame pomembne lastnosti pravega agenta (oz. fenomena)
  - je običajno podan v različnem (manjšem) merilu
  - opusti nekatere (nepomembne) podrobnosti
  - nima vseh funkcionalnosti
- Napovedovanje z modeli
  - z opazovanjem obnašanja modela napovemo obnašanje
  - varneje, bolj poceni, lažje
  - omogoča tesiranje ne da bi zgradili pravega agenta (fenomena)

Model računskega agenta

- Model računskega agenta mora
  - brati vhodne podatke
  - shranjevati in brati podatke iz pomnilnika
  - izvajati ukaze v odvisnosti od trenutnega vhoda in trenutnega stanja
  - proizvesti rezultat

- Turingov stroj

Turingov stroj

- Lastnosti Turingovega stroja:

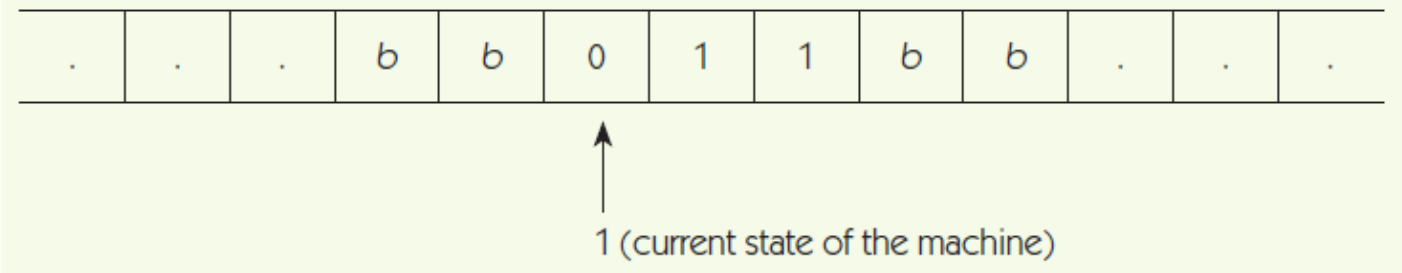
- Ima trak, neskončen v obeh smereh
  - vsako polje na traku hrani en simbol
  - abeceda: končna množica simbolov, ki se berejo in zapisujejo na trak
  - na traku so zapisani vhodni podatki
  - trak služi kot pomnilnik
- Vedno je v enem od končno mnogo stanj (1 do k)
- Ukazi za Turingov stroj
  - v odvisnosti od trenutnega stanja in vhoda
  - zapiši nov simbol, spremeni stanje, ter se premakni za eno polje levo ali desno

Turingov stroj

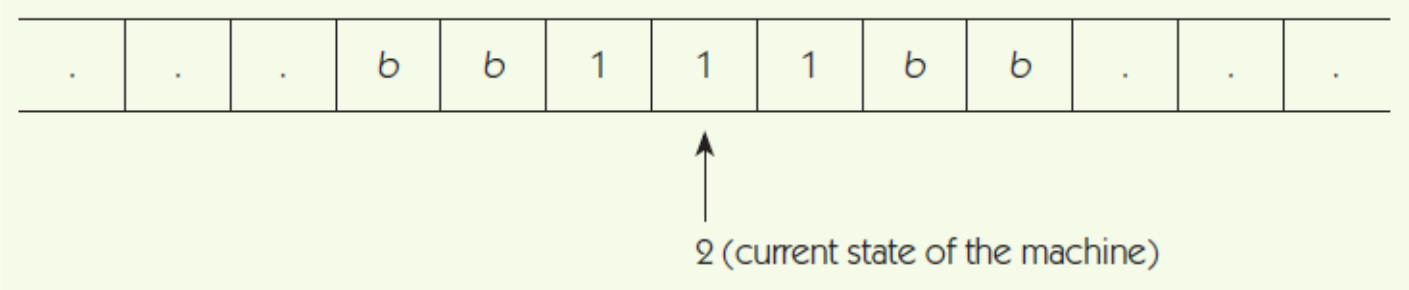
- Izvajanje ukazov:
  - if (v stanju *i*) and (prebere simbol *j*) then
    - zapiši simbol *k* na trak (v trenutno polje)
    - spremeni stanje v stanje *s* (ki lahko ostane enako)
    - premakni se v smer *d* (za eno polje)
  - pravilo: (*i, j, k, s, d)*

- Primer
  - pravilo: (*1, 0, 1, 2, right)*
  - if (stanje=1) and (vhod=0) then
    - zapiši 1
    - pojdi v stanje 2
    - premakni se desno

Primer



(1, 0, 1, 2, right)



Turingov stroj kot računski model

- Množica pravil: program Turingovega stroja
- Prvo stanje je vedno 1
- Stroj vedno začne brati najbolj levo neprazno polje
- Ne smeta obstajati dve pravili z istim stanjem in vhodnim simbolom
  - Se izogiba dvoumnostim
- Če ne obstaja nobeno pravilo za trenutno stanje in vhodni simbol, se stroj ustavi

- Turingov stroj je torej primeren model računskega agenta:
  - bere vhodne podatke s traku
  - trak uporablja kot pomnilnik s katerega bere in na katerega zapisuje podatke
  - izvaja akcije v odvisnosti od trenutnega stanja in vhoda
  - proizvede rezultat in ga zapiše na trak

Turingov stroj kot model algoritma

- Model algoritma mora biti:
  - … popolnoma urejeno zaporedje …
  - … nedvoumnih in učinkovito izračunljivih operacij …
  - … proizvede rezultat …
  - … se ustavi v končnem času.

- Je Turingov stroj primeren model algoritma?

Turingov stroj kot model algoritma

- … popolnoma urejeno zaporedje …
  - začetno stanje in začetna pozicija traku sta definirani
  - največ eno pravilo se lahko hkrati sproži
  - definirano je kaj se zgodi, če ni nobenega ustreznega pravila

  ⇒ Turingov stroj ve kje začeti in kaj kdaj storiti
- … nedvoumnih in učinkovito izračunljivih operacij …
  - ukazi Turingovega stroja popolnoma specificirajo kaj naj stroj naredi
  - ukazi ne morejo biti dvoumni
- … se ustavi v končnem času.
  - Turingov stroj gre lahko v mrtvo zanko (kot slabi algoritmi)
  - lahko zagotovimo, da se ustavi pri pravilnem reševanju pravega problema
- … proizvede rezultat …
  - izhod je zapisan na traku, ko se Turingov stroj ustavi

⇒ Turingov stroj je torej dober model algoritma!

Primeri

- Primeri za demonstracijo delovanja Turingovega stroja:
  - Invertiranje bitov
  - Paritetni bit
  - Eniški inkrement
  - Eniško seštevanje

- Diagram stanj: vizualna prestavitev algoritma Turingovega stroja (pravil)

Primer: Invertiranje bitov

- Naloga: v zaporedju bitov spremeni vsak 1 v 0 in vsak 0 v 1
- Rešitev:
  - invertiraj posamezen bit na traku in se prestavi desno
  - vstavi se, ko naletiš na b

- Pravila:

  (1, 1, 0, 1, R)

  (1, 0, 1, 1, R)

…	b	**1**	0	0	1	b	…
…	b	0	**0**	0	1	b	…
…	b	0	1	**0**	1	b	…
…	b	0	1	1	**1**	b	…
…	b	0	1	1	0	**b**	**…**

Primer: Paritetni bit

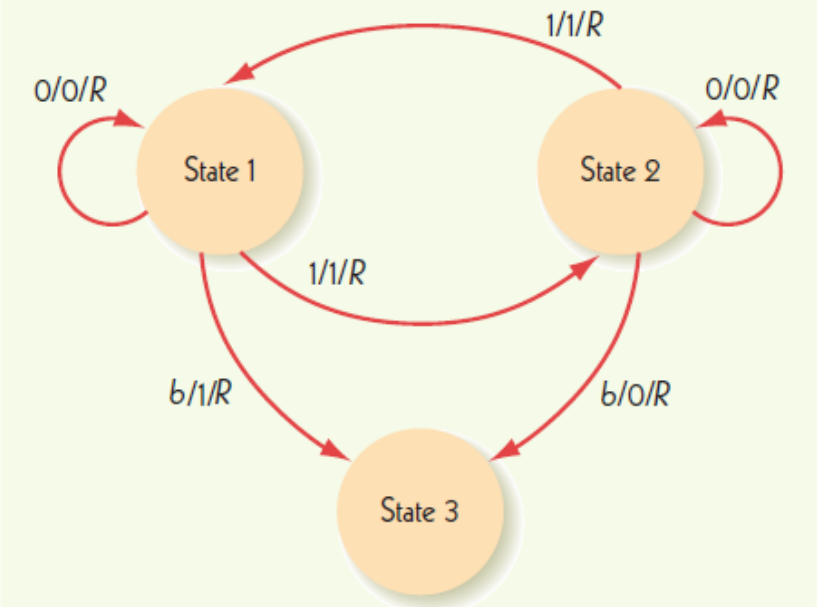
- Naloga:
  - na koncu niza 0 in 1 postavi paritetni bit
  - število vseh 1 mora biti liho

- Rešitev:
  - Stanje 1 predstavlja sodo pariteto do danega trenutka
  - Stanje 2 predstavlja liho pariteto do danega trenutka
  - Vhodni simbol 0 ne spremeni stanja
  - Vhodni simbol spremeni stanje 0 v 1 ter 1 v 0
  - Ko stroj naleti na vhodni simbol b, naj zapiše partitetni bit in gre v stanje 3 za katerega ni nobenega pravila => se bo zaustavil

Primer: Paritetni bit

- Pravila:

  (1, 0, 0, 1, R)
  (1, 1, 1, 2, R)
  (2, 0, 0, 2, R)
  (2, 1, 1, 1, R)
  (1, b, 1, 3, R)
  (2, b, 0, 3, R)



…	b	**1 (1)**	0	0	1	b	b	…
…	b	1	**0 (2)**	0	1	b	b	…
…	b	1	0	**0 (2)**	1	b	b	…
…	b	1	0	0	**1 (2)**	b	b	…
…	b	1	0	0	1	**b (1)**	b	**…**
…	b	1	0	0	1	1	**b (3)**	**…**

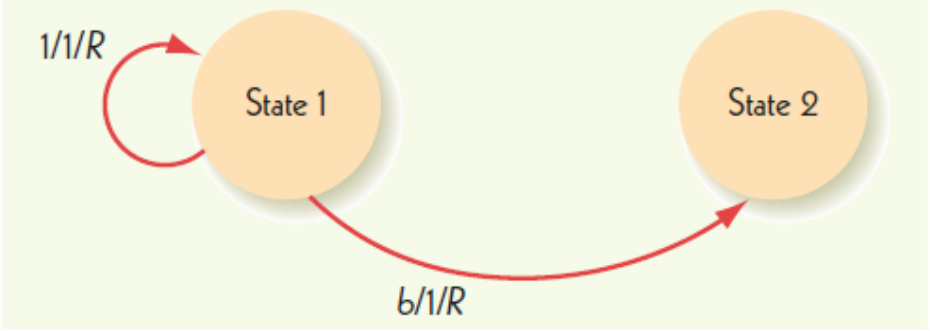
Primer: Eniški inkrement

- Naloga: povečaj število predstavljeno v eniškem sistemu za ena
- Rešitev: pomakni se do konca na desno in dodaj 1

- Pravila:

  (1, 1, 1, 1, R)

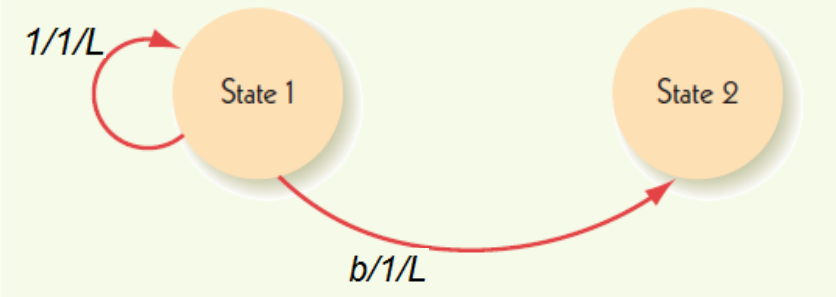
  (1, b, 1, 2, R)



…	b	**1 (1)**	1	1	1	b	b	…
…	b	1	**1 (1)**	1	1	b	b	…
…	b	1	1	**1 (1)**	1	b	b	…
…	b	1	1	1	**1 (1)**	b	b	…
…	b	1	1	1	1	**b (1)**	b	**…**
…	b	1	1	1	1	1	**b (2)**	**…**

Primer: Eniški inkrement

- Boljša rešitev: dodaj 1 na levi strani
- Pravila:
  (1, 1, 1, 1, L)
  (1, b, 1, 2, L)



…	b	b	**1 (1)**	1	1	1	b	…
…	b	**b (1)**	1	1	1	1	b	…
…	**b (2)**	1	1	1	1	1	b	…

- Računska kompleksnost Θ(1) namesto Θ(n)

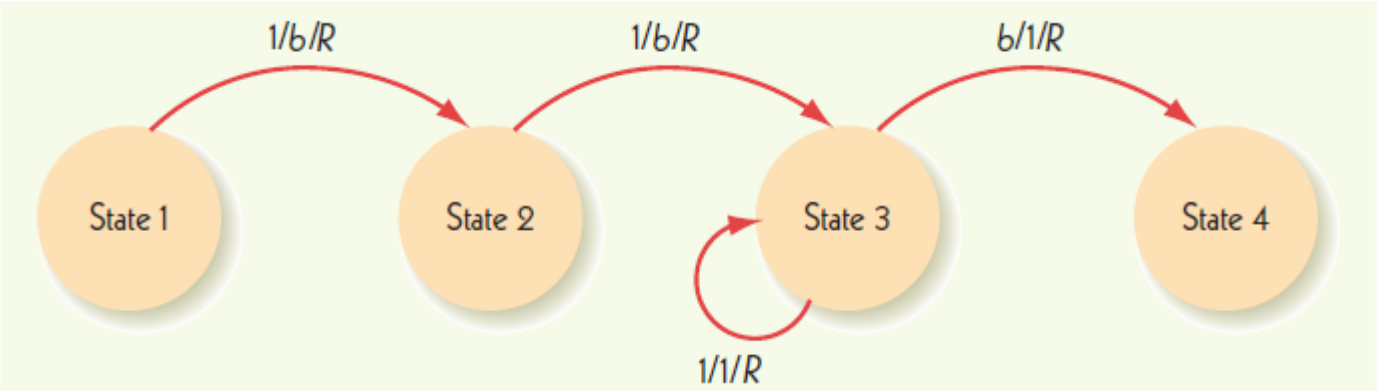
The Number to Be Incremented, *n*	Number of Steps Required	
	Algorithm 1	Algorithm 2
10	12	2
100	102	2
1,000	1,002	2
10,000	10,002	2

Primer: Eniško seštevanje

- Naloga: seštej dve eniški števili, ki sta na traku razmejeni z b
- Rešitev: zbriši dve 1 na levi strani ter zamenjaj vmesni b z 1

- Pravila:

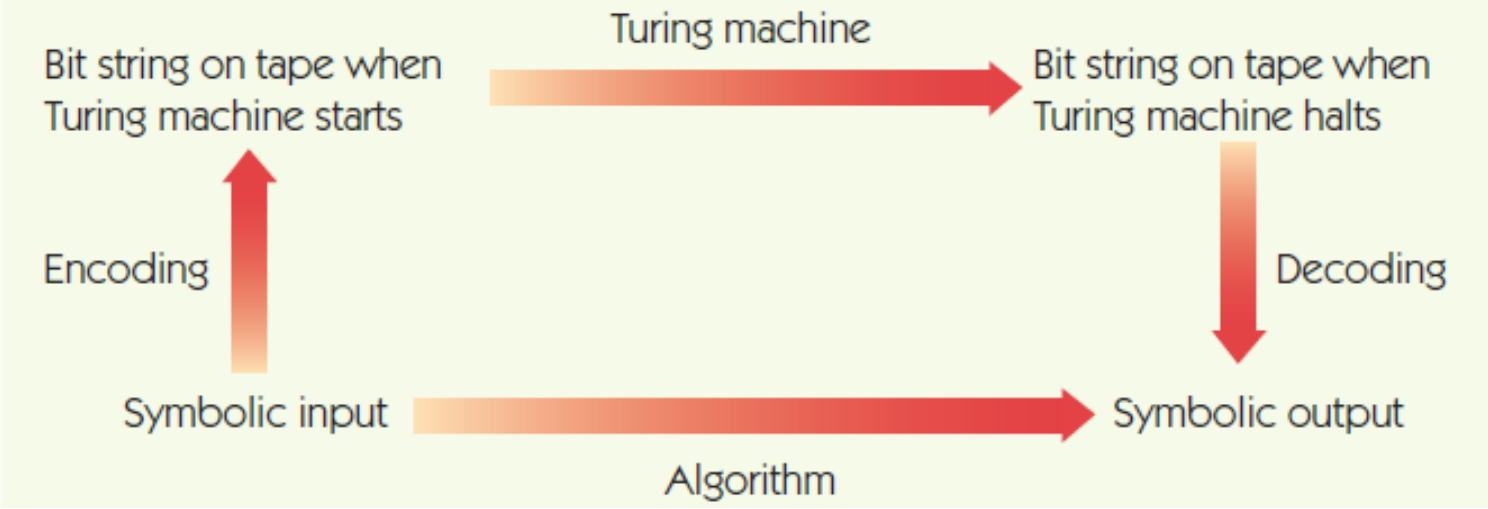
  (1, 1, b, 2, R)
  (2, 1, b, 3, R)
  (3, 1, 1, 3, R)
  (3, g, 1, 4, R)



Church-Turingova teza

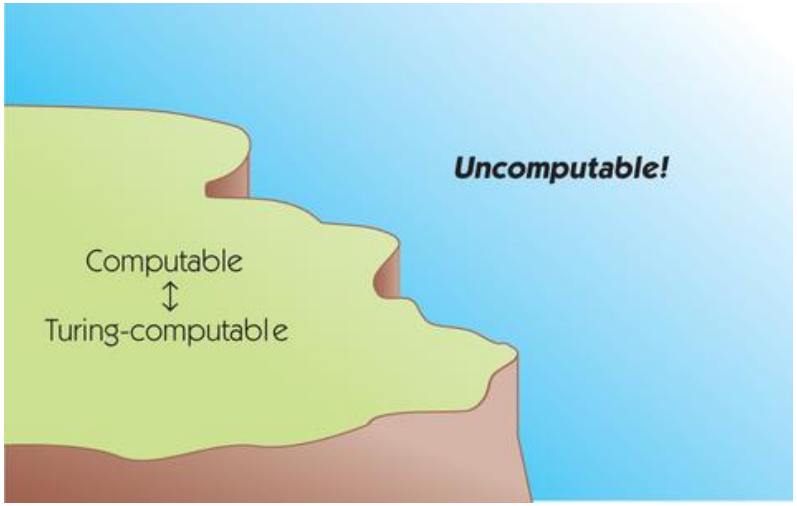
> *"Če obstaja algoritem, ki reši neko nalogo (z manipulacijo s simboličnimi podatki), potem obstaja tudi Turingov stroj, ki reši isto nalogo."*

- Teza ni dokazana, najbrž niti ni dokazljiva, skoraj gotovo pa drži
  - nihče še ni nikoli na nobenem problemu dokazal nasprotno
  - za vse druge računske modele so pokazali, da so ekvivalentni Turingovemu stroju



Nerešljivi problemi

- Turingov stroj definira meje izračunljivosti
- Če problem ne more biti rešen s Turingovim strojem, ne more biti rešen niti z algoritmom
  - je neizračunljiv oz. nerešljiv
- Obstajajo neizračunljivi problemi!
- Dokaz: Problem zaustavitve
  - Ali obstaja Turingov stroj, ki na vhodu prejme program Turingovega stroja ter zanj ugotovi ali se bo ustavil ali ne?
  - Dokaz s protislovjem =>
    takšen Turingov stroj
    ne obstaja =>
    ta problem je nerešljiv =>
    ne obstaja niti algoritem,
    s katerim bi ga lahko rešili!



Povzetek

- Modeli nam omogočajo predvidevati obnašanje in testirati
- Turingov stroj je model računskega agenta
- Programi Turingovega stroja so množice pravil za delovanje
- Programi Turingovega stroja modelirajo algoritme
- Primeri Turingovega stroja:
  - Invertiranje bitov
  - Paritetni bit
  - Eniški inkrement
  - Eniško seštevanje
- Church-Turingova teza trdi, da se lahko vsak algoritem izvede s Turingovim strojem
- Nekateri problemi nimajo algoritmične rešitve
- Problem zaustavitve je nerešljiv problem
  - za ta problem ne obstaja Turingov stroj, ki bi ga rešil