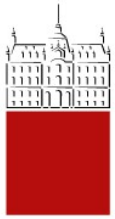
Univerza v Ljubljani
Fakulteta za računalništvo
in informatiko



Nedeterminizem

15.marec 2014

Zgodovina

Michael Rabin

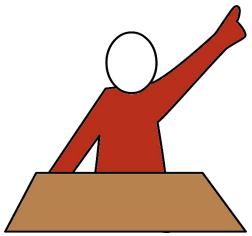
Dana Scott

(1959)

Kaj je nedeterminizem?

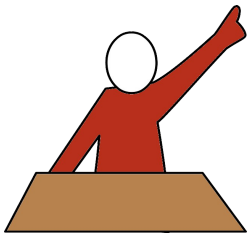
- Nedeterminizem vpeljemo kot razširitev končnih avtomatov
- Kam točno skočimo je nedoločeno (**nedeterminirano**)
- Osnovnim avtomatom zato pravimo deterministični (DKA), razširitvi pa nedeterministični končni avtomati (NKA)
- Dovolimo lahko več prehodov iz istega stanja preko enakega simbola v različna stanja

Zakaj?

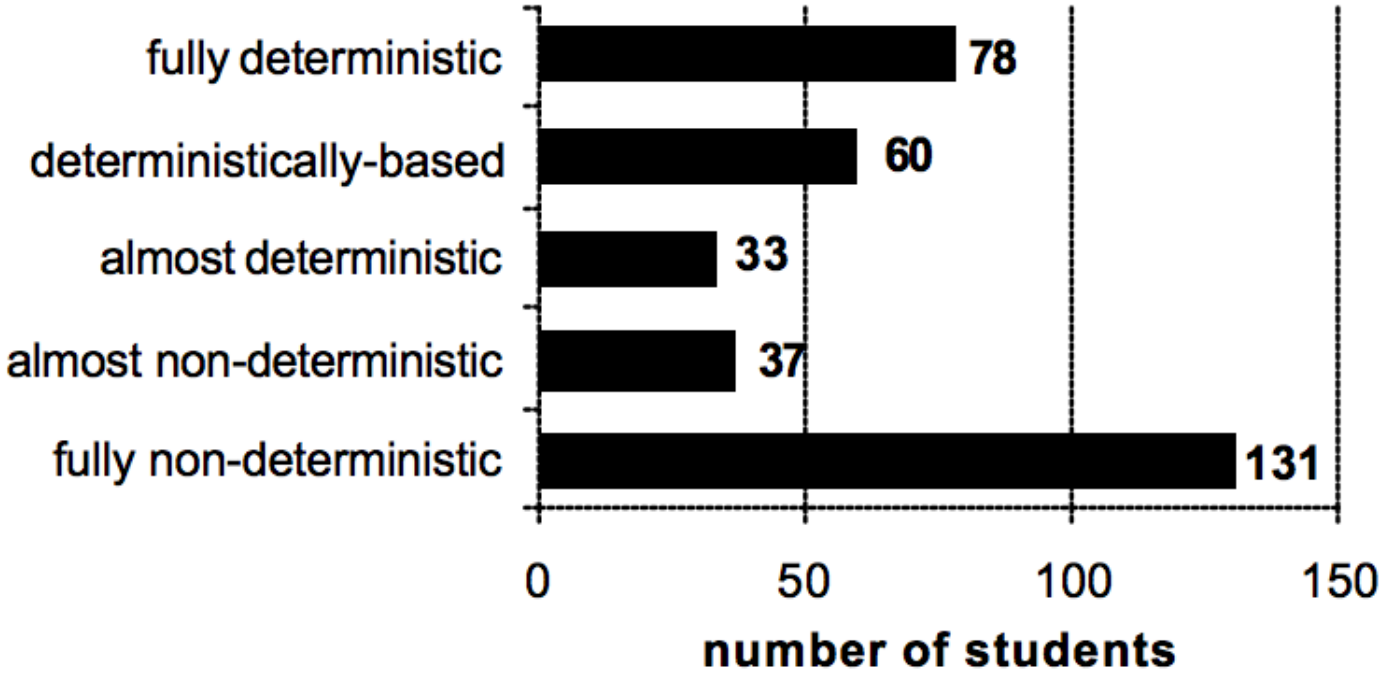


- Nedeterminizem ima izjemno pomembno vlogo v računalništvu
- Teoretična „igra“ - iskanje možnih razširitev nekega modela računanja
- Prvi prikaz dokazovanja enakosti dveh modelov računanja
- Za podajanje nekaterih jezikov je nedeterminizem bolj intuitiven kot determinizem

Intuitiven?

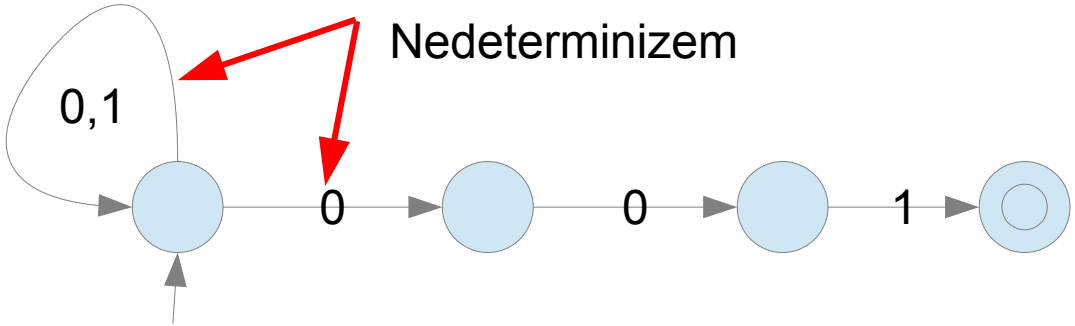


M. Armoni, J. Gal-Ezer: *Non-determinism in CS high-school curricula*, ASEE/IEEE Frontiers in Education Conference, 2003



Primer 1

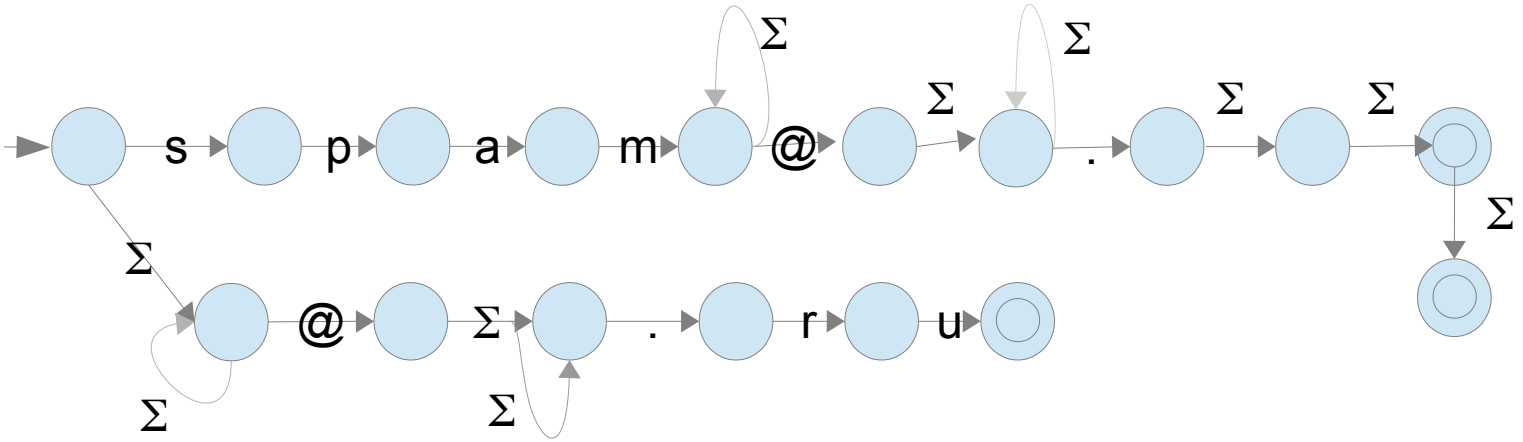
Zapišimo končni avtomat, ki sprejema vse nize, ki se končajo na 001



Primer 2

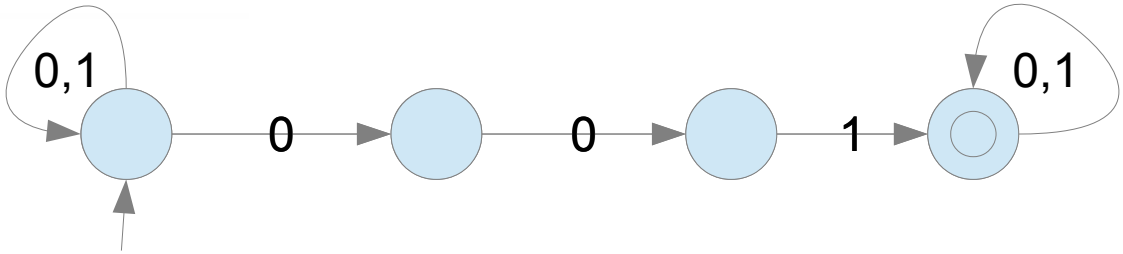
Detektirajmo elektronske naslove, ki so sumljivi in lahko predstavljajo neželjeno pošto. Dve zelo enostavni pravili:

1. Elektronski naslov se začne na **spam**, ali
2. Elektronski naslov se konča na **.ru**

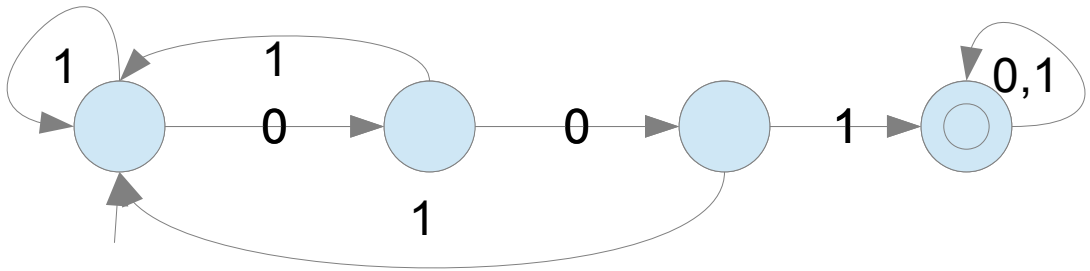


Primer 3

Zapišimo NKA, ki sprejema vse nize, ki **vsebujejo** 001



Deterministični?

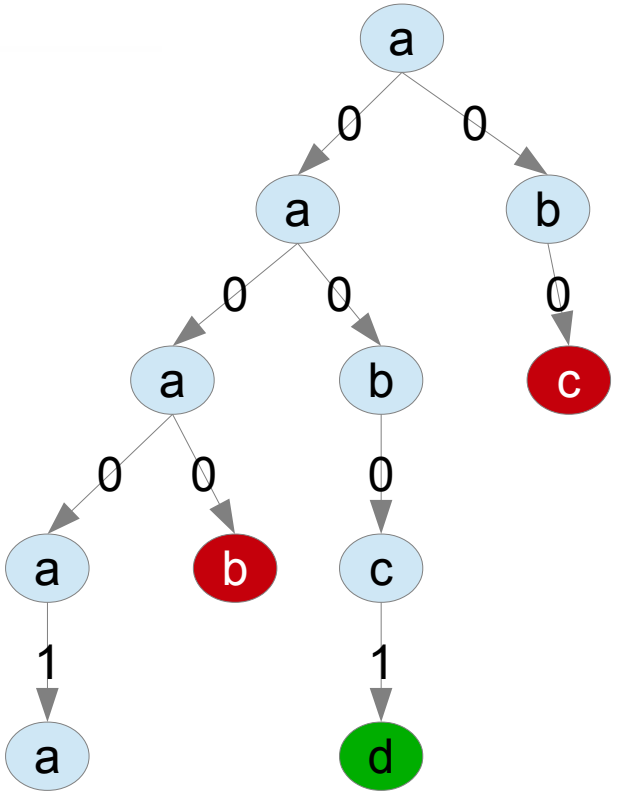


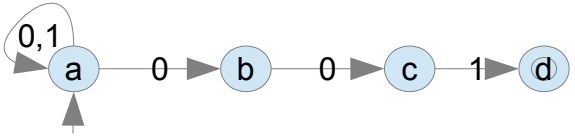
Primerjava z DKA

- Iz enega stanja lahko preko istega simbola pridemo v več različnih stanj – pri DKA samo eden
- Prehod preko nekega simbola je lahko tudi nedefiniran – ne potrebujemo slepega stanja kot pri DKA
- Kako besede sprejmemo?

Sprejemanje besede

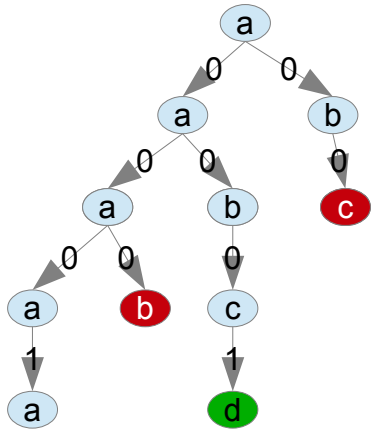
Beseda **0001**





Intuitivna interpretacija nedeterminizma I.

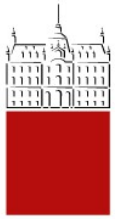
- Vzporedno izvajanje:
  - Vsakič, ko imamo več možnosti, hkrati preizkusimo vse
  - Vsako možnost prevzame en nov računalnik (procesor, jedro, ...)



Intuitivna interpretacija nedeterminizma II.

- Ugibanje:
  - Vsakič, ko imamo več možnost avtomat »ugane« pravo možnost
  - Če je beseda v jeziku, potem takšno zaporedje »ugibanj« obstaja – takemu zaporedju pravimo **certifikat**.
  - Če pa besede ni v jeziku, potem nobeno ugibanje ne pripelje do končnega stanja

Univerza *v Ljubljani*
Fakulteta *za računalništvo*
in informatiko



Formalna definicija

15.marec 2014

Formalna definicija NKA

$$M = (Q, \Sigma, \delta, q_{0,} F)$$

Q	Množica stanj
Σ	Abeceda
$\delta : Q \times \Sigma \rightarrow 2^Q$	Funkcija prehodov, ki slika v poljubno podmnožico Q
$q_0 \in Q$	Začetno stanje
$F \subseteq Q$	Množica končnih stanj

*Jezik nedeterminističnega avtomata I.

Razširimo funkcijo prehodov, da slika neko stanje in **besedo w** v novo stanje

$$\hat{\delta}(q, w): Q \times \Sigma^* \rightarrow 2^Q$$

Najprej preslikamo začetek besede **v** (dobimo množico možnih stanj), nato pa iz vsakega stanja poskusimo priti še po simbolu **a**

$$\hat{\delta}(q, va) = \bigcup_{p \in \hat{\delta}(q, w)} \delta(p, a)$$

*Jezik nedeterminističnega avtomata II.

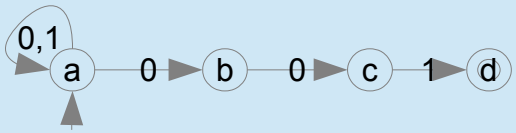
Jezik nedeterminističnega avtomata so vse besede **w**, za katere velja:

$$\hat{\delta}(q, w) \subseteq F \neq \emptyset$$

Neformalno: vse besede preko katerih lahko pridem vsaj do enega končnega stanja.

Podajanje avtomatov

Diagram prehodov



Spisek prehodov

$$\delta(a,0)=\{a,b\}$$

$$\delta(a,1)=a$$

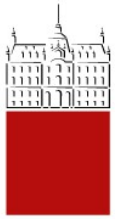
$$\delta(b,0)=c$$

$$\delta(a,0)=a$$

Tabelarično

	0	1
a	a,b	a
b	c	/
c	/	d
d	/	/

Univerza v Ljubljani
Fakulteta za računalništvo
in informatiko



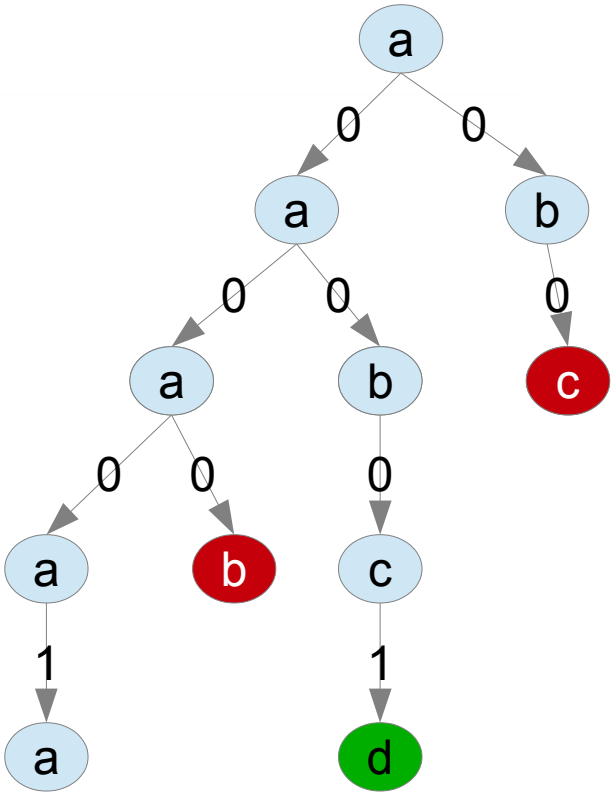
Simulacija

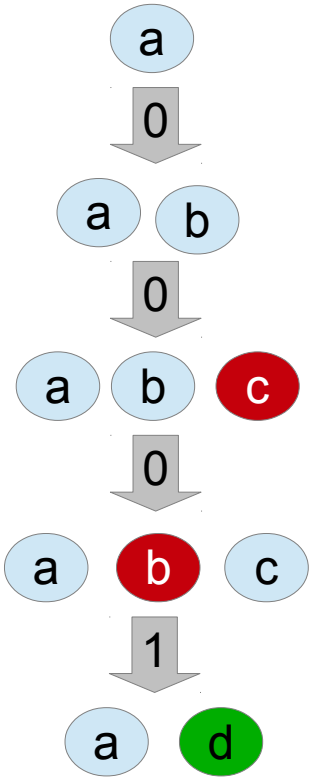
15.marec 2014

Simulacija NKA

- Nedeterminizem ni fizikalno možen model računanja - takega ugibanja ne znamo narediti, poljubnega paralelizma pa tudi ne moremo narediti
- Lahko pa ga (učinkovito?) simuliramo, kar bomo pokazali tako, da bomo tak simulator zapisali
- Razširili bomo program, ki ste ga napisali prejšnji teden

Ideja simulacije





Nekatere `python` opombe

- Delali bomo z množicami, ki so dobro podprte v pythonu
- Ustvarjanje množice: `set([1,2,3])`
- Unija dveh množic:
  - `set([1,2,3]) | set(3,4,5)`
- Presek dveh množic:
  - `set([1,2,3]) & set(3,4,5)`