

Езици, автомати и изчислимоост

доц. Александра Соскова

Факултет по математика и информатика, СУ

Упражнения:

гл ас. Петър Митанкин

гл. ас. Владислав Ненчев

Факултет по математика и информатика, СУ

Норcroft, Motvani, Ullman

Въведение в теория на автоматите,
формалните езици и сложността

- Addison-Vesley, 2002,
 - Ориентирана към приложения
 - Недостатъчно материал
- ≠ старата книга на Норcroft Ullman.



Lewis and Papadimitiou
Elements of the theory of computation

- 2. ed. Prentice-hall, 1998
- Добра теоретична мотивация
- Добре обяснен материал

M. Sipser

Introduction to the theory of computation

- 2. ed. PWS publ comp, MIT, 2006
- Неформално описание- интуиция
- Богат, добре обяснен материал

К. Манев

Увод в дискретната математика

- КЛМН, София, 2003
- автомати и гарматики - добре (друга структура)
- ИЗЧИСЛИМОСТ И СЛОЖНОСТ- малко

Система за оценяване

Изпити:

- контролно на автомати (теория и упражнения)
- контролно на граматика (теория и упражнения)
- Писмен изпит
- Устен изпит

До писмен/устен изпит се допускат само тези студенти, които имат половината точки от контролни, съответно от упражнения/теория

- 3 в. пр. н. ера Евклид ; 9в. Ал-Хоразми
- 1646-1716 Годфрид Лайбниц
- 1832 Чарлз Бабидж - Analytical Engine (Ада Байрон)
- 1900 Давид Хилберт— Хилбертовата програма
- 1933 Курт Гьодел - Теорема за непълнота
- 1936 Алан Тюринг - Машини на Тюринг, Стивън Клини - теория на рекурсивните функции
- Алонзо Чърч — λ изчислими функции
- Шепердсон и Стържиц — МНР
- Чомски, Майхил, Нероуд, Скот, Рабин— автомати
- Стивън Кук—сложност

Основни парадигми

Формални езици: Езикът с който общуваме с компютъра?

Теория на автоматите: Абстрактен модел на прости изчисления.

Изчислимост: Какви проблеми (не) може да се разрешат с компютър?

Сложност: Кои проблеми (не) можем да разрешим ефективно?

Защо ни трябва всичко това?

- (не)директно **приложение**
(алгоритми, идеи, методи)
в езиците за програмиране, компилатори, обработка на текст, (лингвистика), хардуерно и софтуерно инженерство, ...
- Опит с типични дискретни **доказателства**
↪ теория на алгоритмите, логика, теория на изчислимостта, ...

Формални езици, регулярни операции

Азбука: **крайно** множество от символи (Σ)

Думи (в Σ): крайна редица от символи от Σ (w)

Език: множество от думи в Σ (L)

Празната дума: ϵ ($\{\epsilon\} \neq \emptyset$!)

Конкатенация на думи: Пример: $ac \cdot bab = acbab$.

Конкатенация на езици: $L_1 \cdot L_2 := \{w_1 \cdot w_2 : w_1 \in L_1 \wedge w_2 \in L_2\}$.

Пример: $\{ab, ba\} \cdot \{aa, bb\} = \{abaa, abbb, baaa, babb\}$

Обединение на езици: $L_1 \cup L_2 := \{w : w \in L_1 \vee w \in L_2\}$.

Пример: $\{ab, ba\} \cup \{aa, bb\} = \{ab, ba, aa, bb\}$

Означения

$w^0 := \varepsilon$, $w^n := w \cdot w^{n-1}$ за $n \geq 1$.

$L^0 := \{\varepsilon\}$, $L^n := L \cdot L^{n-1}$ за $n \geq 1$.

Пример: $a^3 = aaa$, $\{a, bb\}^2 = \{aa, abb, bba, bbbb\}$

Звезда на Клини: $L^* := \bigcup_{i=0}^{\infty} L^i$

(всяка отделна дума е крайна!)

Пример: $\{a, b\}^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \dots\}$

Σ^* : Множеството от всички думи в Σ .

Позитивна обвивка : $L^+ := \bigcup_{i=1}^{\infty} L^i$

Допълнителен език: $L^c := \Sigma^* \setminus L$

Регулярни езици

основни \emptyset , $\{\varepsilon\}$ и $\{a\}$ за всяко $a \in \Sigma$ са регулярни езици;

\cup Ако L_1 и L_2 са регулярни, то и $L_1 \cup L_2$ е регулярен;

\cdot Ако L_1 и L_2 са регулярни, то и $L_1.L_2$ е регулярен;

$*$ Ако L е регулярен, то и L^* е регулярен.

Един език е **регулярен**, ако се получава от основните с помощта на операциите обединение, конкатенация и звезда, приложени краен брой пъти.

Означения

Let $w = u \cdot v \cdot x$

начало (префикс): u

поддума (инфикс): v

край (суфикс): x

обратна: $(c_1 c_2 \cdots c_k)^R = c_k \cdots c_2 c_1$

$|w|$ (дължина): броят на буквите в w

- $L^= := \{0^n 1^n : n \geq 1\} = \{01, 0011, 000111, \dots\}$
- $\{a^n b^n c^n : n \in \mathbb{N}\} = \{abc, aabbcc, aaabbbccc, \dots\}$
- $\{ww : w \in \Sigma^*\} = \{\epsilon, 00, 11, 0000, 0101, 1010, 1111, \dots\}$
- $\{ww^R : w \in \Sigma^*\} = \{\epsilon, 00, 11, 0000, 0110, 1001, 1111, \dots\}$
- $L_P := \{w : w = w^R\} = \{\epsilon, 0, 1, 00, 11, 000, 010, 101, 111, \dots\}$

Палиндром

English: Ana, level, eye, civic, refer

A Santa lived as a devil at NASA, Borrow or rob;

German: gnudung, hangnah, kajak, lagerregal;

Bulgarian: Аз обичам мач и боза; Насила закараха свинете
ни в Сахара! - каза Лисан ; "

Latin: Sator Arepo Tenet Opera Rotas .

Пример за формален език

Балансирани леви и десни скоби $L_{()}$ в $\Sigma = \{(\,,\,)\}$:

- $\epsilon \in L_{()}$,
- ако $u \in L_{()}$, $v \in L_{()}$, то $uv \in L_{()}$,
- ако $u \in L_{()}$, то $(u) \in L_{()}$.

Алгоритмични проблеми

Принадлежност на дума: $w \in L$?

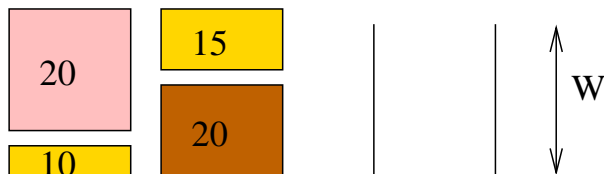
Еквивалентност: $L_1 = L_2$?

Спецификация: математическа дефиниция (описание) на езика
(кога една дума $w \in L$), **граматики**, **разпознаватели** =
автомати(машини),
трансформации между различни спецификации на L .

Защо формални езици?

- Езиците за програмиране, форматът на данните, . . . са формални езици.
- Лесно се формулират задачите.
- Много алгоритмични проблеми се свеждат до принадлежност на дума в даден формален език.

Пример: Задачата за раницата



- n обекта с тегло $w_i \in \mathbb{N}$ и стойност (печалба) p_i
- да се намери подмножество x от обектите
- такова, че $\sum_{i \in x} w_i \leq W$ и
- с максимална печалба $\sum_{i \in x} p_i$

Пример: Задачата за раницата

Дефинираме $L \subseteq \{0, 1, ', '\}^*$:

$w \in L$, ако w е списък от $2n + 2$ двоични числа, разделени със

запетая $P, W, w_1, p_1, \dots, w_n, p_n$, такива че

$$\exists \mathbf{x} \subseteq \{1, \dots, n\} : \sum_{i \in \mathbf{x}} p_i \geq P \text{ и } \sum_{i \in \mathbf{x}} w_i \leq W$$

Принадлежността към езика $L \rightsquigarrow$ Задачата за раницата:

- намираме оптималното P с **двоично търсене**
- намираме съответното \mathbf{x} , махайки от списъка елемент по елемент.

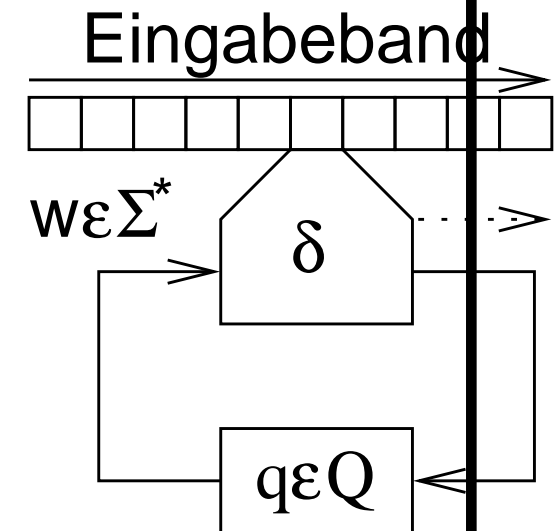
Всичко $\leq n + \log \sum_i p_i$ за да сведем единия проблем до другия
"малко"

Теория на автоматите

Крайни автомати

Един детерминиран краен автомат се състои от:

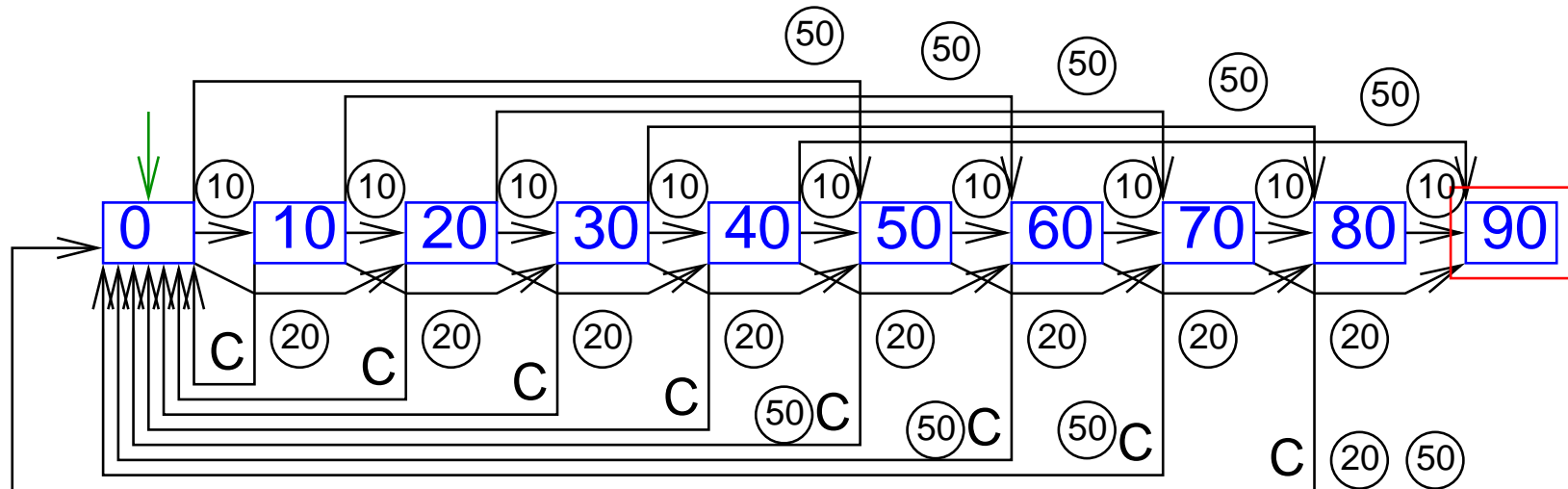
- Q , крайно множество от **състояния**;
- Σ , крайно множество от **символи**, (**азбука**);
- $\delta: Q \times \Sigma \rightarrow Q$, **функция на прехода**;
- $s \in Q$, **начално състояние**;
- $F \subseteq Q$, крайно множество от **заклучителни състояния**.



Пример: билетен автомат

- Стандартна **цена** на билет **90** **цента**
- приемат се монети от: **10**, **20** и **50**
- Стоп с бутон **C** или по-голяма сума монети
- 90 цента са пуснати \rightsquigarrow ready

$(\{10, 20, 50, C\}, \{0, 10, 20, 30, 40, 50, 60, 70, 80, 90\}, \delta, 0, \{90\})$



Граматики

Граматика $G = (V, \Sigma, P, S)$

- V , променливи
- Σ , азбука (терминали) ($V \cap \Sigma = \emptyset$)
- $P \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$, правила, $|P| < \infty$

На всяко правило, лявата част съдържа поне една променлива

- S , начална променлива

Граматика

Иерархия на Чомски: класификация в зависимост от вида на правилата.

В частност: контекстно-свободни езици $A \rightarrow \alpha$, $A \in V$,
 $\alpha \in (V \cup \Sigma)^*$.

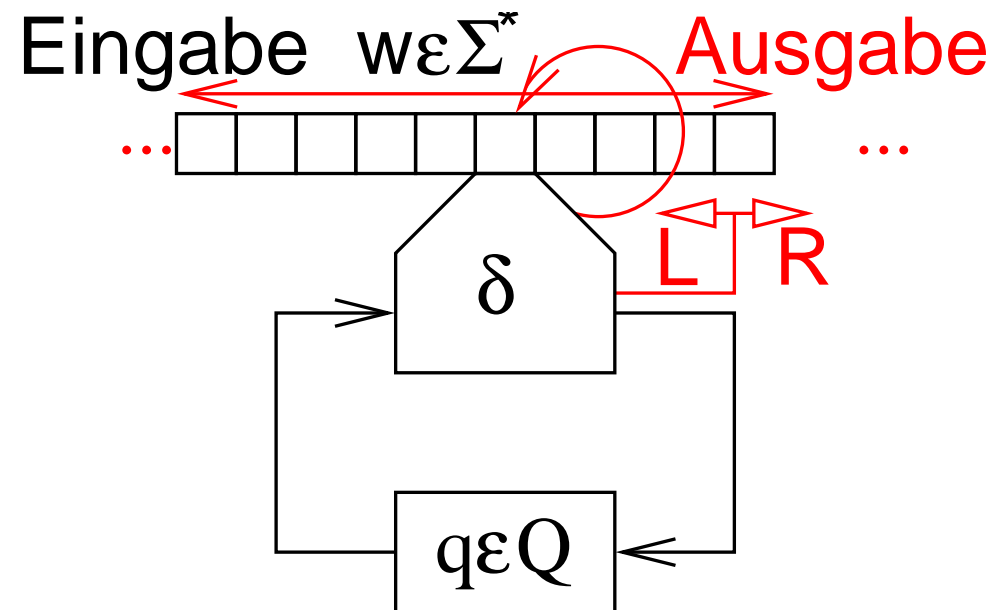
- Спецификация на синтаксиса на програмните езици.
- Контекстно-свободните езици са разрешими.
- Кои езици се разпознават за **линейно** време?

Пример: Аритметични изрази

$G = (\{E, T, F\}, \{a, +, *, (,)\}, P, E)$, където

$$P = \{E \rightarrow T, \\ E \rightarrow E + T, \\ T \rightarrow F, \\ T \rightarrow T * F, \\ F \rightarrow a, \\ F \rightarrow (E)\}$$

Машины на Тюринг и **ИЗЧИСЛИМОСТ**



- Машините на Тюринг решават не повече и не по-малко задачи, отколкото другите **достатъчно мощни машинни модели**
- Може ли да разрешим всеки проблем с машина на Тюринг?

Теория на сложността: Кои задачи можем да пресметнем **ефективно**?

- Пак се базираме на машини на Тюринг. Естествено е да искаме алгоритмите да са ефективни, т.е. времето за изпълнение или ресурсите да са малки (като функция на големината на входа).

$$n \approx n^2 \approx \dots n^{42} \approx \dots \ll 2^{0.134n}$$

- Знаем много малко за **долната граница на времето за изчисление**
- Но има големи класове от задачи, които макар и не полиномиални, **за всички други по-трудоемки** задачи изглеждат прекрасно.