

**Міністерство освіти і науки України
Дніпровський національний університет імені Олеся Гончара
Кафедра обчислювальної математики та математичної кібернетики**

А. Є. Шевельова

**КОМП'ЮТЕРНИЙ ПРАКТИКУМ З ТЕОРІЇ АЛГОРИТМІВ
Навчальний посібник**

**Дніпро
Ліра
2018**

УДК 510.6(075.8)

Рецензенти:

д-р фіз.-мат. наук, проф. Т. С. Кагадій
канд. фіз.-мат. наук, доц. О. В. Черницька

Шевельова А. Є. Комп'ютерний практикум з теорія алгоритмів.
Навчальний посібник [Текст] / А. Є. Шевельова – Д.: Ліра, 2018. – 40 с.

У навчальному посібнику викладено базові формальні моделі теорії алгоритмів та методику роботи зі спеціальними програмами (емуляторами), які моделюють їх роботу.

Для студентів, які навчаються за напрямками «Системний аналіз», «Прикладна математика» денної, прискореної та заочної форм навчання.

Рекомендовано до друку вченою радою факультету прикладної математики Дніпровського національного університету імені Олеся Гончара 30.05.2018 р., протокол №11.

© Шевельова А.Є., 2018

Вступ

Поняття алгоритму та методи теорії алгоритмів, теорії скінченних автоматів, формальних мов, граматик – теоретична основа сучасної теорії програмування, побудови алгоритмічних мов, проектування мовних процесорів, зокрема, компіляторів, асемблерів, макрогенераторів.

У наш час апарат теорії алгоритмів використовують всюди, де є алгоритмічні проблеми (основи математики, теорія інформації, теорія керування, конструктивний аналіз, обчислювальна математика, теорія ймовірності, лінгвістика, економіка та ін.).

Необхідність точного математичного коректування інтуїтивного поняття алгоритму стала неминучою після усвідомлення неможливості існування алгоритмів розв'язання багатьох масових проблем, у першу чергу, пов'язаних з арифметикою та математичною логікою (проблеми істинності арифметичних формул і формул числення предикатів першого порядку, 10-та проблема Гілберта про розв'язність діофантових рівнянь та ін.).

Для доведення відсутності алгоритму розв'язання певної задачі необхідно мати точне математичне визначення алгоритму, тому після сформування поняття алгоритму як нової та окремої сутності першочерговою постала проблема знаходження адекватних формальних моделей алгоритму й дослідження їх властивостей. Таким чином, формальні моделі були запропоновані як для первісного поняття алгоритму, так і для похідного поняття алгоритмічно обчислюваної функції.

Перехід від інтуїтивного поняття алгоритму до формального визначення алгоритму (рекурсивні функції, машини Тьюрінга, нормальні алгоритми Маркова, машини натуральнозначних регістрів та інші) дозволяє довести алгоритмічну нерозв'язність ряду проблем.

Пошук формальних уточнень поняття алгоритму проводився в таких напрямках:

- 1) опис точного математичного поняття алгоритмічної машини та обчислюваності на ній. Першою формальною моделлю алгоритмічної машини була машина Тьюрінга, яка моделює елементарні дії під час реалізації алгоритму людиною (А. Тьюрінг, Е. Пост, 1936). Зараз відомо багато різновидів машин Тьюрінга.

Із пізніших формальних моделей алгоритмів відзначимо також нормальні алгоритми (А. Марков, 1952) та регістрові машини (Д. Шепердсон, Г. Стерджіс, 1963). Модифікація регістрових машин – машини із натуральнозначними регістрами;

- 2) опис певних класів функцій, для яких існує алгоритм знаходження функції за значеннями її аргументів, тобто уточнюють не первісне поняття алгоритму, а похідне поняття алгоритмічно обчислюваної функції. Спочатку такі описи були запропоновані для функцій, заданих на множині

натуральних чисел, пізніше – для функцій, заданих на множинах інших об'єктів.

В теорії алгоритмів передбачені основні концепції, які закладені в апаратуру і мови програмування електронних обчислювальних машин. Основні алгоритмічні моделі математично еквівалентні. Але на практиці вони істотно розрізняються ефектами складності, що виникають при реалізації алгоритмів, і породили різні напрямки в програмуванні. Так, мікропрограмування будується на ідеях машин Тьюрінга, структурне програмування запозичило свої конструкції з теорії рекурсивних функцій, мови символічної обробки інформації беруть початок від нормальних алгоритмів Маркова та канонічних систем Посту.

На основі теорії алгоритмів в даний час отримані практичні рекомендації, що набувають все більшого поширення в області проектування і розробки програмних систем. Результати теорії алгоритмів набувають особливого значення для криптографії.

Теорія автоматів є одним із фундаментальних розділів сучасної теоретичної та практичної інформатики. Вона безпосередньо пов'язана з математичною логікою, теорією алгоритмів, теорією формальних граматики. Наприклад, засобами теорії автоматів можна довести розв'язність деяких формальних числень.

Застосування методів і понять теорії автоматів до вивчення формальних і природних мов сприяло виникненню математичної лінгвістики (математична лінгвістика – математична дисципліна, предметом якої є розробка формального апарата для опису будови природних і деяких штучних мов).

Практично всі керуючі й обчислювальні пристрої, різноманітні контролери, використовувані в засобах зв'язку, являють собою скінченні автомати. Сітки Петрі широко застосовують для опису й дослідження паралельних процесів та систем, наприклад протоколів обміну інформацією, операційних систем.

Поняття автомата може служити модельним об'єктом у найрізноманітніших задачах, завдяки чому можливе застосування теорії автоматів у різних наукових і прикладних дослідженнях.

У навчальному посібнику розглядаються формальні моделі алгоритму у формі алгоритмічних машин, для яких існують і можуть бути розроблені самостійно програми-емулятори їх роботи. Використання таких емуляторів дозволяє перевірити правильність розробленого алгоритму розв'язання конкретної задачі, протестувати його на численних прикладах. Крім того, емулятори допомагають візуалізувати процес розв'язання задачі в термінах кожної формальної моделі.

Розглянуто машину Тьюрінга, нормальні алгоритми Маркова, машину натуральнозначних регістрів, скінчені автомати Мілі і Мура. Для кожної формальної моделі алгоритму наведено основні теоретичні відомості і програми-емулятори їх роботи.

1. Машина Тьюрінга

Англійський інженер і математик Алан Матісон Тьюрінг (1912–1954) увів поняття математичної машини (**Turing machine**), яка моделює розумову діяльність людини і уточнює інтуїтивне представлення про алгоритми.

Машина Тьюрінга – абстрактна обчислювальна машина (абстрактний виконавець) для формалізації поняття алгоритму. Машина Тьюрінга є розширенням скінченного автомата і, згідно з тезою Черча-Тьюрінга, здатна імітувати всіх інших виконавців. Цю строгу формальну математичну модель було названо «машиною» з тієї причини, що за описом її складових частин і функціонування вона є схожа на обчислювальну машину. Принципова відмінність машини Тьюрінга від обчислювальних машин полягає в тому, що її блок зовнішньої пам'яті є нескінченною стрічкою, а у реальних обчислювальних машин пам'ять може бути як завгодно великою, але обов'язково скінченною. Машину Тьюрінга не можна реалізувати саме через нескінченність її стрічки. У цьому сенсі вона потужніша від будь-якої обчислювальної машини.

Машина Тьюрінга (скорочено МТ) складається зі стрічки і керуючого пристрою з голівкою читання-запису (рис. 1).

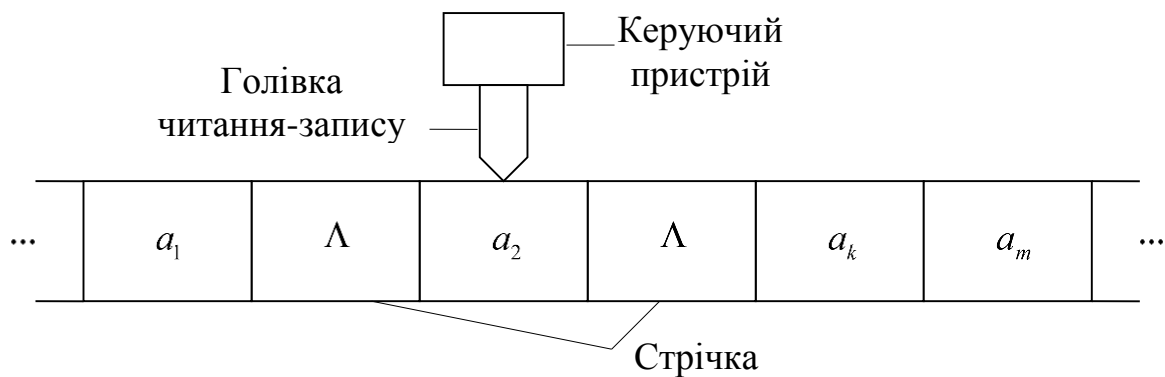


Рис. 1

Стрічка є нескінченною в обидва боки і розділена на комірки (клітки). Існують машини Тьюрінга, які мають декілька нескінченних стрічок. У кожній комірці в кожний дискретний момент часу знаходиться рівно один символ із алфавіту $A = \{a_1, a_2, \dots, a_n\}$, $n \geq 2$. Виокремлюється особливий порожній символ Λ , що заповнює всі комірки стрічки, крім тих з них (скінченного числа), у яких записані вхідні дані. Найпростіший алфавіт машини Тьюрінга $A = \{0, 1\}$, де 0 – порожній символ.

Керуючий пристрій у кожний дискретний момент часу перебуває в певному стані q_j , що належить множині внутрішніх станів $Q = \{q_0, q_1, \dots, q_m\}$, $m \geq 1$. МТ починає роботу в стані q_1 , а потрапивши в стан q_0 машина завжди зупиняється.

Голівка читання-запису в кожен момент часу «бачить» рівно одну комірку стрічки. Вона може зчитувати вміст цієї комірки і записувати в неї замість цього символу деякий інший символ (чи той же самий) із зовнішнього алфавіту A .

У процесі роботи керуючий пристрій залежно від стану, в якому він знаходиться і спостережуваного в поточній комірці символу, змінює свій внутрішній стан (може залишитися в попередньому стані), видає голівці читання-запису наказ надрукувати у спостережуваній комірці певний символ із зовнішнього алфавіту і або залишатися на місці, або зсунутись на одну комірку вліво чи на одну комірку вправо. Роботу керівного пристрою машини Тьюрінга виконують відповідно до програми.

Таким чином, пам'ять МТ – скінченна множина станів (внутрішня пам'ять) і стрічка (зовнішня пам'ять). Дані МТ – слова в алфавіті стрічки. Елементарні кроки МТ – зчитування і запис символів, зсув голівки читання-запису вправо або вліво, перехід керуючого пристрою в наступний стан.

Машина Тьюрінга може виконувати всі можливі перетворення слів, реалізуючи тим самим всі можливі алгоритми. На машині Тьюрінга можна імітувати машину Поста, алгоритми Маркова і будь-яку програму для звичайних комп'ютерів, перетворюючи вхідні дані у вихідні з якого-небудь алгоритму.

Формально під **машиною Тьюрінга** будемо розуміти впорядковану п'ятірку $\langle Q, A, \delta, q_1, q_0 \rangle$, де:

- Q – скінченна множина внутрішніх станів;
- A – скінченний алфавіт символів стрічки, причому A містить спеціальний символ порожньої клітки Λ (або λ);
- $\delta: Q \times A \rightarrow Q \times A \times \{R, L, S\}$ – однозначна функція переходів;
- $q_1 \in Q$ – початковий стан;
- $q_0 \in Q$ – заключний (фінальний) стан.

Програма машини Тьюрінга складається зі скінченної кількості команд. Кожна **команда** має вигляд п'ятірки :

$$q_i a_j q_k a_m d \text{ або } q_i a_j \rightarrow q_k a_m d ,$$

де $d = \{S, L, R\}$ – функція руху голівки читання-запису, S означає відсутність руху голівки читання-запису (стоп), L – зсування на одну комірку вліво і R – зсування на одну комірку вправо.

Виконання команди $q_i a_j q_k a_m d$ має таке значення: якщо керуючий пристрій знаходиться у стані q_i , голівка читання-запису розглядає комірку, в якій записано символ a_j , то відповідно до цієї команди голівка читання-запису замінює символ a_j на a_m ; керуючий пристрій зі стану q_i переходить в стан q_j і керуючий пристрій з голівкою читання-запису або зсувається вліво (L), або вправо (R), або ж залишається на місці (S).

Наприклад, команда $q_2 a_3 q_4 a_4 L$. Її виконання продемонстровано на рис. 2.

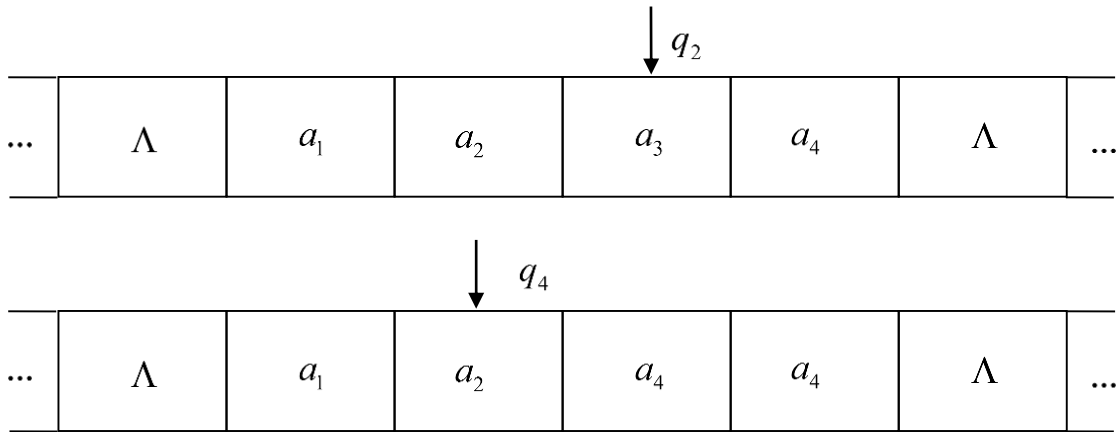


Рис. 2

Будемо вимагати, щоб для будь-якої пари $q_i a_j$ була в програмі точно одна команда, що починається з цього запису. Якщо стан змінюється на q_0 , то можна стверджувати, що машина зупиняється і жодної команди, яка починається з q_0 немає.

Будемо вважати, що:

- МТ починає роботу зі стану q_1 ,
- на стрічці у кожний дискретний момент часу тільки скінченна кількість непорожніх символів і серед них можна виділити крайній правий та крайній лівий символ,
- машина в стані q_1 починає рух на крайньому лівому непорожньому символі.

Роботу програми машини Тьюрінга будемо ілюструвати за допомогою конфігурацій. Слово $a_{j_1} \dots a_{j_{i-1}} q_i a_{j_i} \dots a_{j_s}$ називають **конфігурацією машини Тьюрінга** (у даний момент часу t).

За умови $t = 1$ конфігурація має вигляд $q_1 a_{j_1} \dots a_{j_s}$. Якщо в момент часу t стрічка порожня, то конфігурацією машини буде слово $q_i \Lambda \Lambda \Lambda \dots$.

Якщо $P_1 = a_{j_1} \dots a_{j_s}$ – вхідне слово, то МТ, почавши роботу на слові P_1 або зупиниться через певну кількість кроків, або ніколи не зупиниться.

У першому випадку стверджують, що МТ **застосовна до слова** P_1 і результатом є слово P , що відповідає заключній конфігурації і $P = MT(P_1)$.

У другому випадку – МТ **не застосовна до слова** P_1 .

Часто будуть використані позначення типу P^m для слів такого вигляду $\underbrace{PP \dots P}_m$.

Машину Тьюрінга можна записувати у вигляді:

- системи команд (правил);

- таблиці (рядки – стани; стовпці – символи; на перетині – новий стан, символ і напрямок зсуву голівки читання-запису);
- діаграми (або графа).

Розглянемо найпростіший алфавіт $A = \{0,1\}$, де 0 – порожній символ і (x_1, x_2, \dots, x_n) – довільний набір цілих невід’ємних чисел. Слово $1^{x_1+1}01^{x_2+1}0\dots01^{x_n+1}$ називають **основним машинним кодом** і позначають $K(x_1, x_2, \dots, x_n) = K(x_1)0K(x_2)0\dots0K(x_n)$ для будь-якого $n \in N$. Зокрема як $K(x) = 1^{x+1}$ будемо кодувати блок з $(n+1)$ -ї одиниці.

Далі розглядаємо часткові числові функції $f(x_1, \dots, x_n)$, $x_i \in N_0$, $N_0 = \{0, 1, 2, \dots\}$.

Часткову числову функцію $f(x_1, \dots, x_n)$ можна **обчислити за Тьюрінгом**, якщо існує машина Тьюрінга, що обчислює цю функцію, тобто має такі властивості:

а) якщо значення $f(x_1, \dots, x_n)$ визначено, то $MT(K(x_1, x_2, \dots, x_n)) = K(f(x_1, \dots, x_n))$, тобто через скінченну кількість тактів машина прийде до q_0 і на стрічці буде записаний код $K(f(x_1, \dots, x_n))$;

б) якщо $f(x_1, \dots, x_n)$ невизначено, то МТ не можна застосувати до слова $K(x_1, x_2, \dots, x_n)$.

Приклад 1. Побудувати машину Тьюрінга, яка початкову конфігурацію $K_1 = q_1 1^n$ переводить у заключну конфігурацію $K_0 = q_0 1^{2n}$.

Програму машини Тьюрінга запишемо вигляді системи команд.

Програма МТ

$q_1 1 \rightarrow q_2 0R,$
 $q_1 0 \rightarrow q_0 0R,$
 $q_2 1 \rightarrow q_2 1R,$
 $q_2 0 \rightarrow q_3 0R,$
 $q_3 0 \rightarrow q_4 1R,$
 $q_3 1 \rightarrow q_3 1R,$
 $q_4 0 \rightarrow q_5 1L,$
 $q_5 1 \rightarrow q_5 1L,$
 $q_5 0 \rightarrow q_6 0L,$
 $q_6 1 \rightarrow q_6 1L,$
 $q_6 0 \rightarrow q_1 0R.$

Конфігурації МТ

$q_1 1^n \vdash 0q_2 1^{n-1} \vdash 01q_2 1^{n-2} \vdash \dots \vdash 01^{n-1}q_2 0 \vdash$
 $\vdash 01^{n-1}0q_3 0 \vdash 01^{n-1}01q_4 0 \vdash 01^{n-1}0q_5 11 \vdash$
 $\vdash 01^{n-1}q_5 011 \vdash 01^{n-2}q_6 1011 \vdash \dots \vdash$
 $\vdash q_6 01^{n-1}01^2 \vdash q_1 1^{n-1}01^2 \vdash 0q_2 1^{n-2}01^2 \vdash \dots \vdash$
 $\vdash 01^{n-2}q_2 01^2 \vdash 1^{n-2}0q_3 1^2 \vdash 1^{n-2}01^2 q_3 0 \vdash$
 $\vdash 1^{n-2}01^2 1q_4 0 \vdash 1^{n-2}01^2 11q_5 0 \vdash \dots \vdash q_0 1^{2n}.$

Приклад 2. Побудувати машину Тьюрінга, яка початкову конфігурацію $K_1 = 1^n q_1 0 1^m$, $m \geq 1$, $n \geq 1$, переводить у заключну конфігурацію $K_0 = 1^m q_0 0 1^n$. Програму машини Тьюрінга запишемо вигляді таблиці (табл. 1):

Таблиця 1

	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}
0	$q_2 0L$	$q_3 0R$	$q_{10} 0R$	$q_5 0R$	$q_6 0R$	$q_7 1L$	$q_8 0L$	$q_9 0L$	$q_3 0R$	$q_0 0S$
1		$q_2 1L$	$q_4 0R$	$q_4 1R$	$q_5 1R$	$q_6 1R$	$q_7 1L$	$q_8 1L$	$q_9 1L$	$q_{10} 1R$

Конфігурації:

$11q_1 01 \vdash 1q_2 101 \vdash 0q_2 1101 \vdash q_2 01101 \vdash q_3 1101 \vdash 0q_4 101 \vdash 01q_4 01 \vdash 010q_5 1 \vdash$
 $0101q_5 0 \vdash 01010q_6 0 \vdash 0101q_7 01 \vdash 010q_8 101 \vdash 01q_8 0101 \vdash 0q_9 10101 \vdash q_9 010101 \vdash$
 $q_3 10101 \vdash q_4 0101 \vdash q_5 101 \vdash 1q_5 01 \vdash 10q_6 1 \vdash 101q_6 0 \vdash 10q_7 11 \vdash 1q_7 011 \vdash q_8 1011 \vdash$
 $q_9 001011 \vdash 0q_3 01011 \vdash q_{10} 1011 \vdash 1q_{10} 011 \vdash 1q_0 011.$

Приклад 3. Побудувати машину Тьюрінга обчислення функції усіченої різниці

$$f(x, y) = x \dot{-} y = \begin{cases} x - y, & x \geq y, \\ 0, & x < y, \end{cases}$$

в алфавіті $A = \{1, \lambda\}$, де аргумент функції задається як $1^{x+1} \lambda 1^{y+1}$, λ – порожній символ. Результатом роботи МТ буде $1^{f(x,y)+1}$. Програму МТ наведено в табл. 2.

Таблиця 2

	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9
λ	$q_2 \lambda R$	$q_3 \lambda L$	$q_8 \lambda L$	$q_5 \lambda L$	$q_6 \lambda R$	$q_7 1R$	$q_9 \lambda L$	$q_0 1S$	$q_9 \lambda L$
1	$q_1 1R$	$q_2 1R$	$q_4 \lambda L$	$q_4 1L$	$q_5 1L$	$q_1 \lambda R$	$q_7 \lambda R$	$q_8 1L$	$q_0 1S$

2. Емулятори машини Тьюрінга

Емулятор машин Тьюрінга – спеціальна програма, що дозволяє моделювати роботу машин Тьюрінга. Вхідними даними цієї програми є стани керуючого пристрою машини Тьюрінга і вхідне слово, а результатом роботи програми є слово, що вийшло після його перетворення керуючим пристроєм.

У цьому параграфі розглянуто кілька реалізацій машини Тьюрінга.

Всі представлені програми вільно поширюються і доступні для всіх бажаючих. Комерційне використання даних програм заборонено відповідно до авторських прав.

1. Емулятор машини Тьюрінга К. Ю. Полякова (2013).

Програма знаходиться у вільному доступі на офіційному сайті Полякова <http://kpolyakov.spb.ru/>. Ознайомитися з її зовнішнім виглядом можна на рис. 3.

У розділі довідки можна отримати коротку інформацію про автора і програму, перейти на офіційний сайт.

У верхній частині програми розташовано спадаюче меню, в якому дублюються можливі маніпуляції з програмою, можливість завантажити дані і налаштувати швидкість обробки. Програма має можливість зберігати і завантажувати дані. Перевагою перед іншими реалізаціями є те, що в програмі передбачено відновлення даних стрічки.

Запуск здійснюється з панелі команд. Так само там є кнопка для покрокового запуску програми користувача.

У блок «Умова завдання» можна записати умову задачі або будь-які коментарі, замітки до задачі. Для коментарів і заміток існує окремий блок «Коментар».

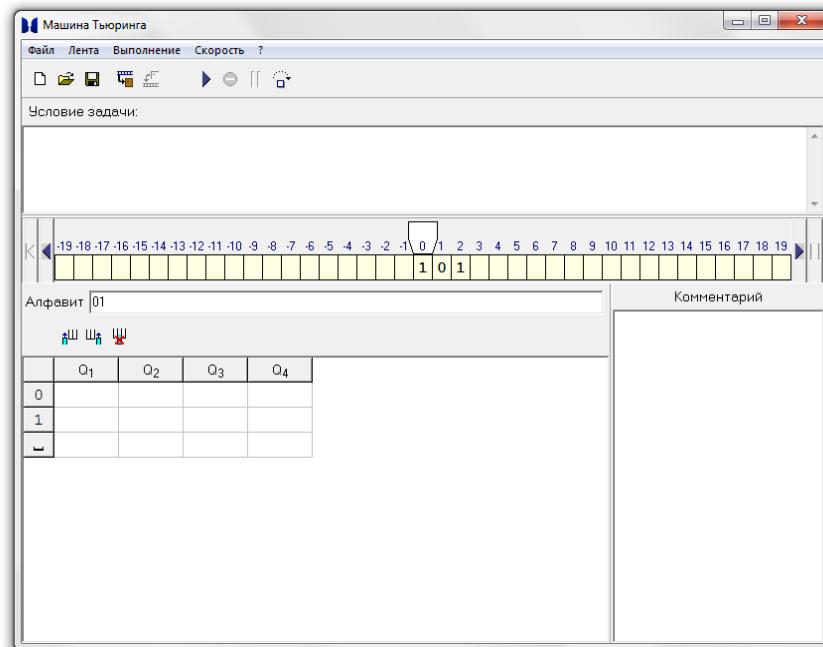


Рис. 3

Керуючий пристрій з голівкою читання-запису пересувається за допомогою кнопок, розташованих з боків від стрічки. Дані в стрічку вводяться подвійним кліком по комірці і вибором із символів алфавіту.

Алфавіт вводиться вручну з клавіатури через рядок «Алфавіт». Машина Тьюрінга в подальшому самостійно розбиває алфавіт на символи і виділяє під кожен символ відповідний рядок.

Таблиця станів є розширювана. Кількість стовпців і відповідно множину станів Q користувач може збільшити, натиснувши на відповідну кнопку над таблицею. Видалення зайвих стовпців також відбувається вручну.

Для написання команди для машини Тьюрінга в клітинку таблиці необхідно вписати три символи. Перший символ, той на який потрібно змінити

символ спостережуваної комірки. Другий символ показує, в який бік зрушити керуючий пристрій з голівкою читання-запису. Третій – в який стан перейти програмі. В комірках є маска введення, завдяки чому символи, які не входять до множини символів алфавіту, номерів станів і керуючих символів ввести неможливо. Для переміщення керуючого пристрою з голівкою читання-запису використовуються: «<», «>», «.». Для зміни стану пишеться номер стану.

Після введення команди в клітинку таблиці стану програма замінює зовнішній вигляд двох останніх елементів команди на стрілочку для керуючого пристрою з голівкою читання-запису, де «<» (L) або «>» (R) відповідає напрямку стрілки, точка відповідає стрілці вниз і стояння керуючого пристрою з голівкою читання-запису на місці. Номер стану змінюється на Q_n , де n – номер стану.

Вигляд емулятора для наведеної програми обчислення усіченої різниці показано на рис. 4.

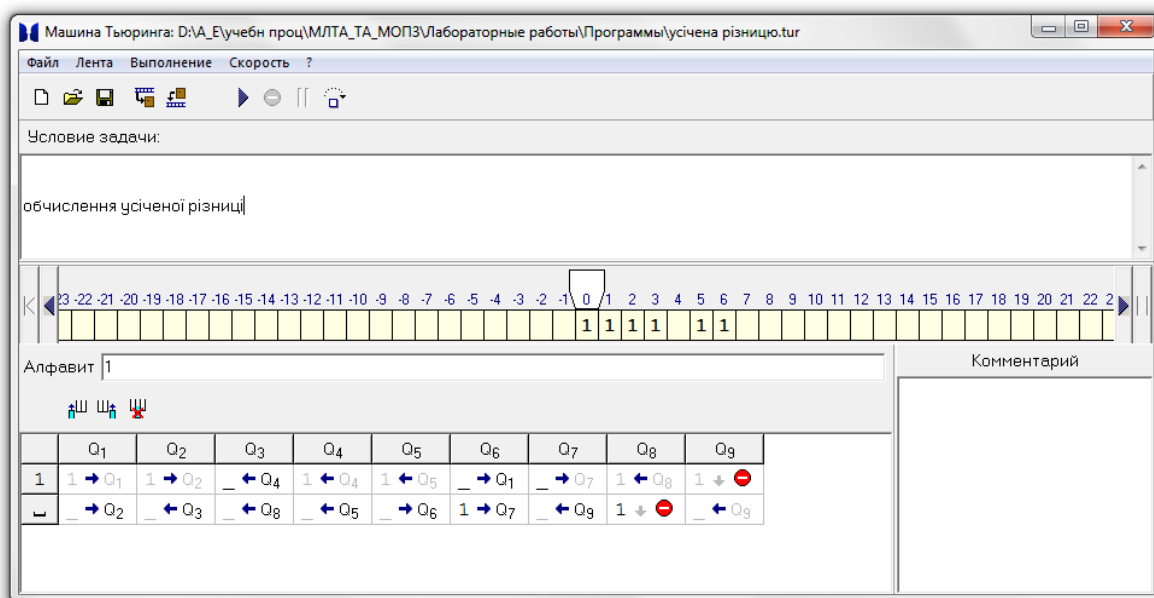


Рис. 4

2. Емулятор машини Тьюрінга А. С. Балюка (2007).

Така реалізація є онлайн версією машини Тьюрінга і доступна за електронною адресою <http://matinf.igpu.ru/simulator/tm.html>.

Дана реалізація має коротке керівництво користувачеві (розташоване на html-сторінці праворуч). Для написання програми необхідно з клавіатури вписати всі команди в блок команд. У блоці алфавіту потрібно встановити галочки напроти символів, які використані в програмі. У блоці множини станів потрібно встановити галочки напроти позначень станів, які використані в програмі. Є можливість додати символи алфавіту і додаткові стани.

Редагувати стрічку можна тільки через рядок «Конфігурація». Перед запуском програми потрібно натиснути «Встановити», зафіксувавши

початковий стан q_1 та початкову конфігурацію. Зовнішній вигляд емулятора наведено на рис. 5.

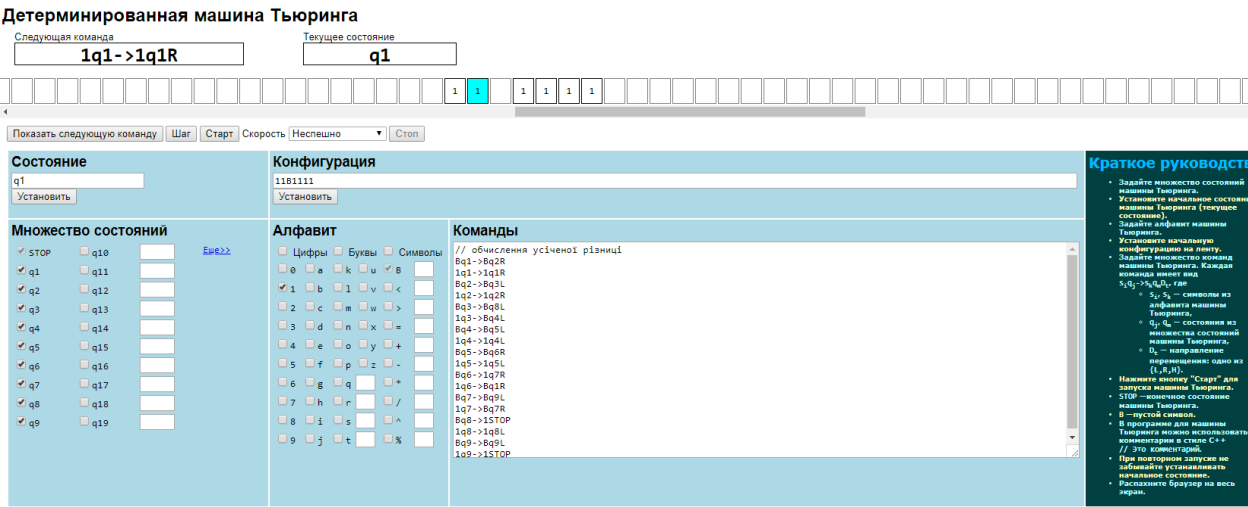


Рис. 5

Синтаксис програми для цього емулятора наведено в таблиці 3 на прикладі обчислення усіченої різниці.

Таблица 3	Таблица 4
	initial q_1 , accept q_0 ,
$Bq_1 \rightarrow Bq_2R$	$q_1 _ q_2 _ 1$,
$1q_1 \rightarrow 1q_1R$	$q_1 \ 1 \ q_1 \ 1 \ 1$,
$Bq_2 \rightarrow Bq_3L$	$q_2 _ q_3 _ -1$,
$1q_2 \rightarrow 1q_2R$	$q_2 \ 1 \ q_2 \ 1 \ 1$,
$Bq_3 \rightarrow Bq_8L$	$q_3 _ q_8 _ -1$,
$1q_3 \rightarrow Bq_4L$	$q_3 \ 1 \ q_4 _ -1$,
$Bq_4 \rightarrow Bq_5L$	$q_4 _ q_5 _ -1$,
$1q_4 \rightarrow 1q_4L$	$q_4 \ 1 \ q_4 \ 1 \ -1$,
$Bq_5 \rightarrow Bq_6R$	$q_5 _ q_6 _ 1$,
$1q_5 \rightarrow 1q_5L$	$q_5 \ 1 \ q_5 \ 1 \ -1$,
$Bq_6 \rightarrow 1q_7R$	$q_6 _ q_7 \ 1 \ 1$,
$1q_6 \rightarrow Bq_1R$	$q_6 \ 1 \ q_1 _ 1$,
$Bq_7 \rightarrow Bq_9L$	$q_7 _ q_9 _ -1$,
$1q_7 \rightarrow Bq_7R$	$q_7 \ 1 \ q_7 _ 1$,
$Bq_8 \rightarrow 1STOP$	$q_8 _ q_0 \ 1 \ 0$,
$1q_8 \rightarrow 1q_8L$	$q_8 \ 1 \ q_8 \ 1 \ -1$,
$Bq_9 \rightarrow Bq_9L$	$q_9 _ q_9 _ -1$,
$1q_9 \rightarrow 1STOP$	$q_9 \ 1 \ q_0 \ 1 \ 0$

3. Емулятор машини Тьюрінга за електронною адресою: <https://alistat.eu/online/turingmachinesimulator>.

В програмі спочатку оголошуються позначення початкового та заключного станів. Команди записуються у формі «поточний стан пробіл

символ алфавіту пробіл новий стан пробіл символ алфавіту пробіл напрямок переходу», де символом алфавіту може бути буква, цифра, символ (крім коми) або _ (підкреслення, що означає порожній символ).

Напрямок може бути 1 (направо) або -1 (ліворуч). Всі команди розділені комою. У віконці «*Input*» задаються вхідні дані.

Зовнішній вигляд емулятора наведено на рис. 6. Синтаксис програми для цього емулятора наведено в таблиці 4 на прикладі обчислення усіченої різниці.

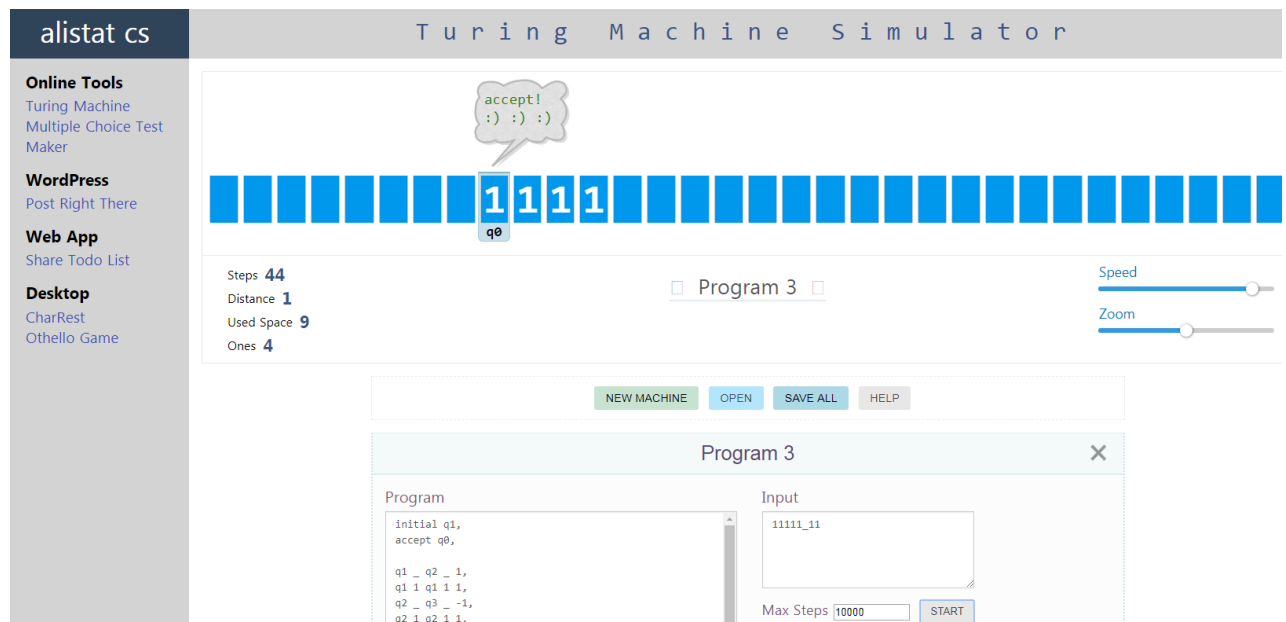


Рис. 6

4. Емулятор машини Тьюрінга за електронною адресою: <http://morphett.info/turing/turing.html>.

Зовнішній вигляд емулятора наведено на рис. 7.

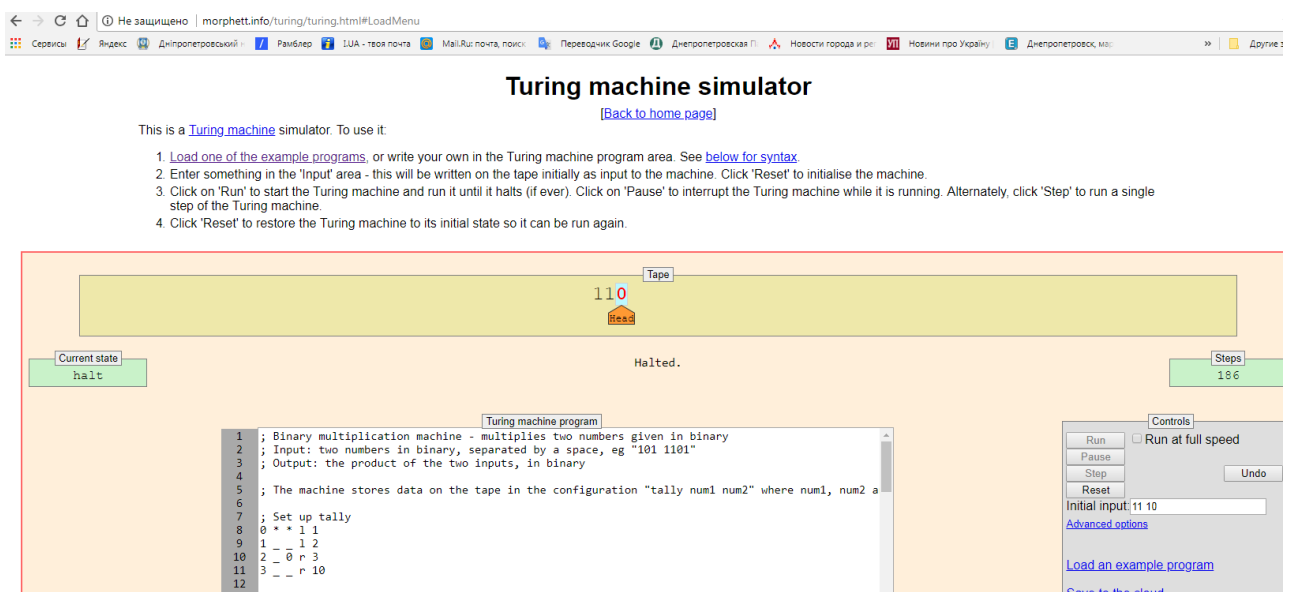


Рис. 7

3. Нормальні алгоритми Маркова

Нормальний алгоритм в алфавіті T (скорочено НА в алфавіті T , **Markov algorithm**) – це упорядкована послідовність підстановок вигляду $\alpha \rightarrow \beta$ або $\alpha \rightarrow \cdot \beta$, де $\alpha, \beta \in T^*$ та $\{\rightarrow, \cdot\} \notin T$, T^* – множина слів скінченної довжини в алфавіті T .

Підстановки вигляду $\alpha \rightarrow \cdot \beta$ називають **заключними (фінальними)**, а $\alpha \rightarrow \beta$ – **простими**.

Довільну скінченну послідовність підстановок називають **схемою** нормального алгоритму. Кожен НА в алфавіті T задає певне словарне відображення $T^* \rightarrow T^*$. Слово, яке є результатом обробки слова x нормальним алгоритмом \mathcal{A} , позначимо $\mathcal{A}(x)$.

Цікавою особливістю нормальних алгоритмів Маркова є те, що в них використовують лише одну елементарну дію – підстановку.

Слово γ входить в слово ξ , якщо існують слова δ_1 та δ_2 (можливо порожні) такі, що

$$\xi = \delta_1 \gamma \delta_2.$$

Марковська підстановка виконується так. У заданому слові P знаходять перше входження слова α (якщо воно є) і, не змінюючи інших частин слова P , замінюють в ньому це входження словом β . Отримане слово називається результатом застосування марковської підстановки $\alpha \rightarrow \beta$ до слова P . Якщо ж першого входження α в слово P (і, отже, взагалі немає жодного входження α в P), то вважається, що марковська підстановка $\alpha \rightarrow \beta$ не може бути застосована до слова P .

Обробку слова x нормальним алгоритмом \mathcal{A} проводять поетапно.

Нульовий етап. Покладемо $x_0 = x$ і скажемо, що x_0 отримане з x після 0 етапів.

$(n+1)$ -й етап. Нехай слово x_n отримане зі слова x після n етапів. Шукаємо першу за порядком продукцію $\alpha \rightarrow (\cdot)\beta$ таку, що α – підслово x_n . Запис точки в дужках означає, що точка може стояти на цьому місці, а може бути відсутньою. Застосуємо цю продукцію до x_n , тобто замінимо в x_n найлівіше входження α на β . Отримане слово позначимо x_{n+1} . Якщо застосована на $(n+1)$ -му етапі продукція не є фінальна, тобто $\alpha \rightarrow \beta$, то переходимо до $(n+2)$ -го етапу. Якщо ця продукція фінальна, тобто $\alpha \rightarrow \cdot \beta$, то після її застосування нормальний алгоритм \mathcal{A} зупиняється і $\mathcal{A}(x) = x_{n+1}$. Якщо ж на $(n+1)$ -му етапі жодна продукція нормального алгоритму \mathcal{A} незастосовна до x_n , тобто в \mathcal{A} немає продукцій, ліва частина яких – підслово слова x_n , то \mathcal{A} зупиняється і $\mathcal{A}(x) = x_n$.

Якщо в процесі обробки слова x НА \mathcal{A} не зупиняється ні на якому етапі, то вважаємо, що результат обробки слова x нормальним алгоритмом \mathcal{A} невизначений.

Нормальний алгоритм називають **нормальним алгоритмом над алфавітом T** , якщо він – нормальний алгоритм у деякому розширеному алфавіті $T' \supseteq T$.

НА над T задає словарне відображення $T^* \rightarrow T^*$, використовуючи в процесі обробки слів допоміжні символи поза алфавітом T . Зупинка НА над T у роботі над словом $x \in T^*$ є **результативна**, коли алгоритм зупинився з результатом $y \in T^*$, інакше результат роботи нормального алгоритму над словом x невизначений.

Зауважимо, що порожній символ λ у необмеженій кількості зустрічається **перед/між/після** будь-яких символів скінченного алфавіту T .

Нормальні алгоритми \mathcal{A} і \mathcal{B} **еквівалентні відносно алфавіту T** , якщо для всіх $x \in T^*$ значення $\mathcal{A}(x)$ та $\mathcal{B}(x)$ одночасно визначені або невизначені, та у випадку визначеності $\mathcal{A}(x) = \mathcal{B}(x)$.

Відомо, що для кожного НА над алфавітом T існує еквівалентний йому відносно T НА в алфавіті $T \cup \{s\}$ з єдиним допоміжним символом $s \notin T$. Словарне відображення, яке кожне слово $x \in T^*$ переводить у слово xs , не може бути заданим жодним НА в алфавіті T , але це можна зробити в алфавіті T' .

Приклад 1. Нехай $T = \{a, b, c\}$ і λ – порожній символ. Побудуємо НА над алфавітом T , який подвоює кожний символ слова $P \in T^*$. Наприклад, вхідне слово $P = abcba$, вихідним буде слово $P' = aabbccbbbaa$.

- 1) $*a \rightarrow aa*$
- 2) $*b \rightarrow bb*$
- 3) $*c \rightarrow cc*$
- 4) $* \rightarrow \cdot \lambda$
- 5) $a \rightarrow *a$
- 6) $b \rightarrow *b$
- 7) $c \rightarrow *c$

Нормальний алгоритм \mathcal{A} **обчислює** часткову функцію $f: N_0^k \rightarrow N_0$, якщо він кожне слово вигляду x_1, x_2, \dots, x_k переводить у слово $f(x_1, x_2, \dots, x_k)$ у випадку $(x_1, x_2, \dots, x_k) \in D_f$ та $\mathcal{A}(x_1, x_2, \dots, x_k)$ – невизначене для $(x_1, x_2, \dots, x_k) \notin D_f$.

Функцію називають **обчислюваною за Марковим** або **НА-обчислюваною**, якщо існує нормальний алгоритм Маркова, який її обчислює.

Приклад 2. Побудувати НА Маркова для функції усіченої різниці $f(x, y) = x \dot{-} y$.

Виберемо алфавіт $T = \{a, b, c\}$ і λ – порожній символ. Задаємо вхідне слово у вигляді $P = a \dots a \underset{x \text{ разів}}{b} \dots b$, а вихідне слово – $P' = c \dots c$ $\underset{x-y \text{ разів}}$. Тоді маємо

- 1) $ab \rightarrow \lambda$
- 2) $a \rightarrow c$
- 3) $b \rightarrow \lambda$

Приклад 3. Побудувати НА для функції $f(x, y) = x - y$.

Функція $f(x, y) = x - y$ – часткова числова функція. На множині N_0 вона визначена для $x \geq y$ і невизначена для $x < y$.

Виберемо алфавіт $T = \{a, b\}$ і λ – порожній символ. Задаємо вхідне слово у вигляді $P = a \dots a \underset{x \text{ разів}}{b} \dots b$, а вихідне слово – $P' = a \dots a$ $\underset{x-y \text{ разів}}$, якщо НА застосовний до слова P . Тоді маємо таку схему НА:

- 1) $ab \rightarrow \lambda$
- 2) $b \rightarrow b$

Приклад 4. Побудувати НА для функції $f(x) = \left\lfloor \frac{x}{2} \right\rfloor$.

Виберемо алфавіт $T = \{a, b\}$ і λ – порожній символ. Задамо вхідне слово у вигляді $P = a \dots a$ $\underset{x \text{ разів}}$, а вихідне слово – $P' = a \dots a$ $\underset{f(x) \text{ разів}}$. Маємо таку схему НА над алфавітом T :

- 1) $*aa \rightarrow a*$
- 2) $a*a \rightarrow a*$
- 3) $* \rightarrow \lambda$
- 4) $\lambda \rightarrow *$

4. Емулятори нормальних алгоритмів Маркова

У цьому параграфі розглянуто кілька реалізацій нормальних алгоритмів Маркова.

Всі представлені програми вільно поширюються і доступні для всіх бажаючих. Комерційне використання даних програм заборонено відповідно до авторських прав.

1. Емулятор нормальних алгоритмів Маркова К.Ю. Полякова (2013).

Програма знаходиться у вільному доступі на офіційному сайті Полякова <http://kpolyakov.spb.ru/>. Ознайомитися з її зовнішнім виглядом можна на рис. 8 (наведено приклад нормального алгоритму для обчислення усіченої різниці).

Наведемо деякі відомості про роботу з емулятором. Слова в підстановках, розташовані зліва і праворуч від знаку « \rightarrow », можуть бути (а

можуть і не бути) поміщені в апострофи або подвійні лапки. Такі підстановки рівносильні і визначають заміну літери «*a*» на комбінацію «*bc*»:

$$a \rightarrow bc, 'a' \rightarrow 'bc', "a" \rightarrow "bc", 'a' \rightarrow "bc".$$

Символ «.» після слова-заміни означає фінальну підстановку, після якої виконання алгоритму закінчується. Наприклад: $a \rightarrow bc.$ означає замінити *a* на *bc* і закінчити виконання програми.

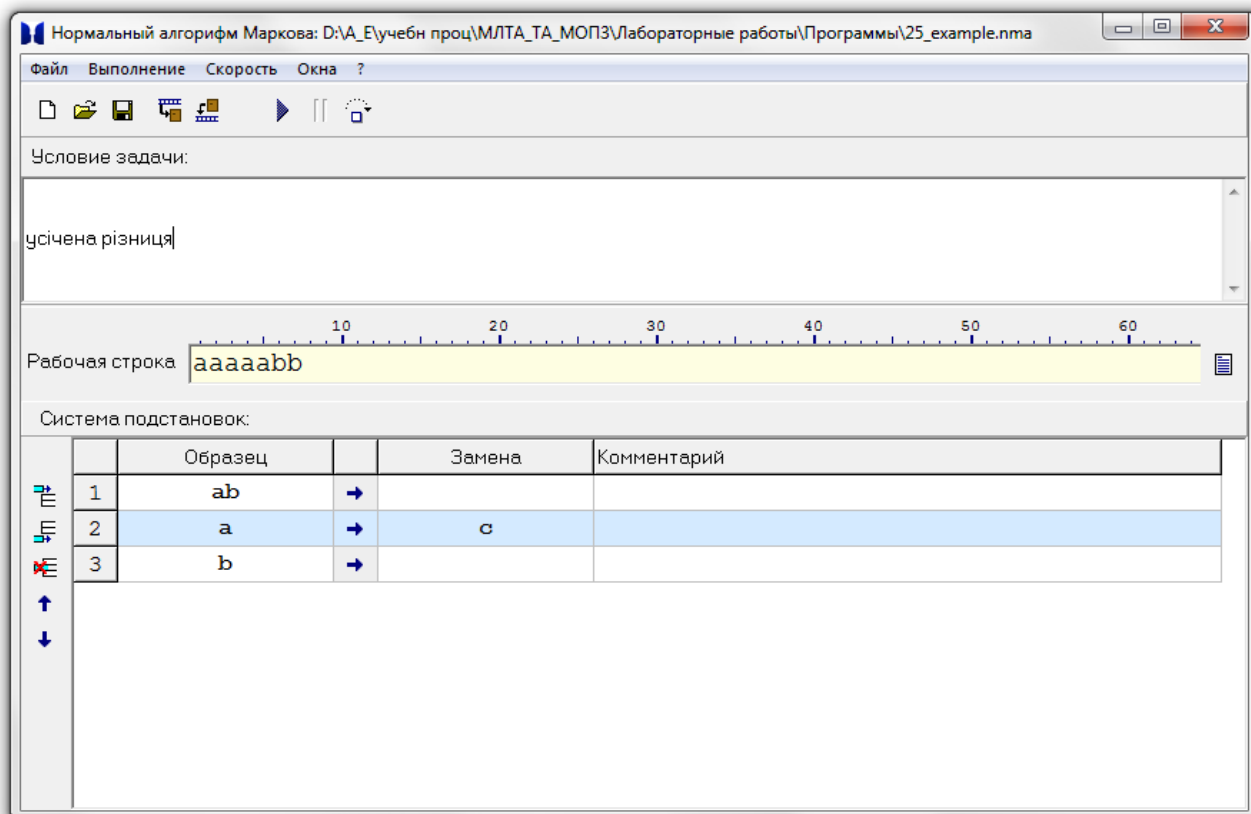


Рис. 8

У верхній частині програми знаходиться поле редактора, в яке можна ввести умову задачі у вільній формі.

Система постановок, що задає нормальний алгоритм Маркова, набирається у вигляді таблиці в нижній частині вікна програми.

Програма може виконуватися безперервно (F9) або по кроках (F8). Команда, яка зараз буде виконуватися, підсвічується зеленим фоном. Швидкість виконання регулюється за допомогою меню *Швидкість*.

Задачі можна зберігати в файлах і завантажувати з файлів. Зберігається умова задачі, вихідне слово і система підстановок.

Протокол роботи алгоритму, в якому показані всі послідовні заміни, викликається при натисканні клавіш *Ctrl+P*. На рис. 9 наведено послідовність підстановок для усіченої різниці при $x = 2$, $y = 6$. Відповіддю у цьому випадку буде порожнє слово.

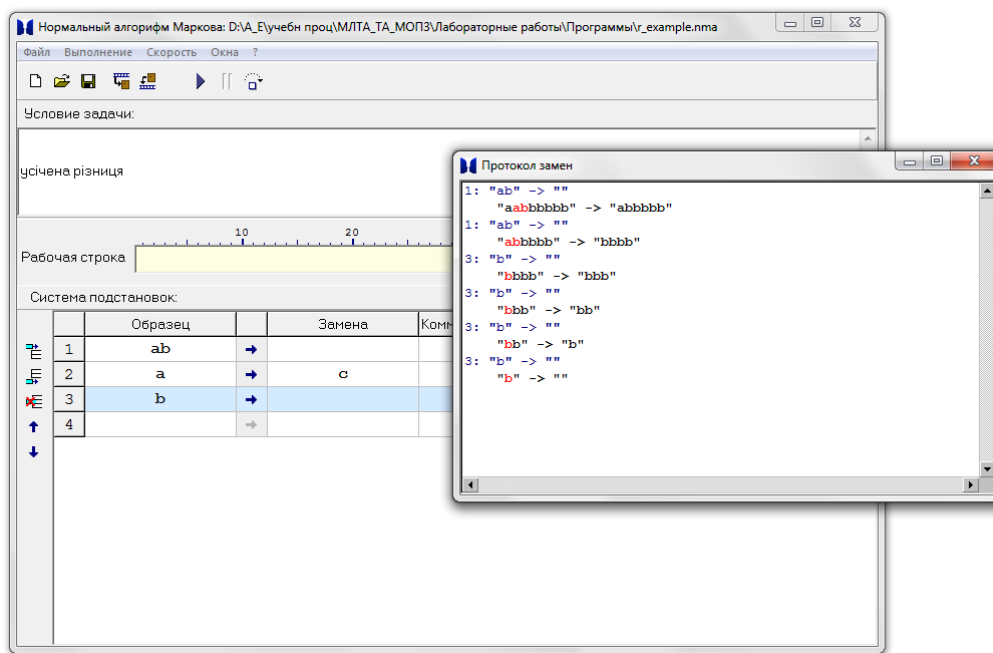


Рис. 9

2. Эмулятор нормальных алгоритмов Маркова на html-странице за адресою <http://matinf.vsgao.com/simulator/na.html>

Дана реалізація має коротке керівництво користувачеві (розташоване на html-сторінці праворуч). Для написання схеми нормального алгоритму Маркова необхідно з клавіатури вписати всі підстановки в блок команд. У блоці алфавіту потрібно встановити галочки напроти символів, які використані в НА. Є можливість додати символи алфавіту.

Редагувати вхідні дані можна через рядок «Конфігурація». Зовнішній вигляд емулятора наведено на рис. 10.

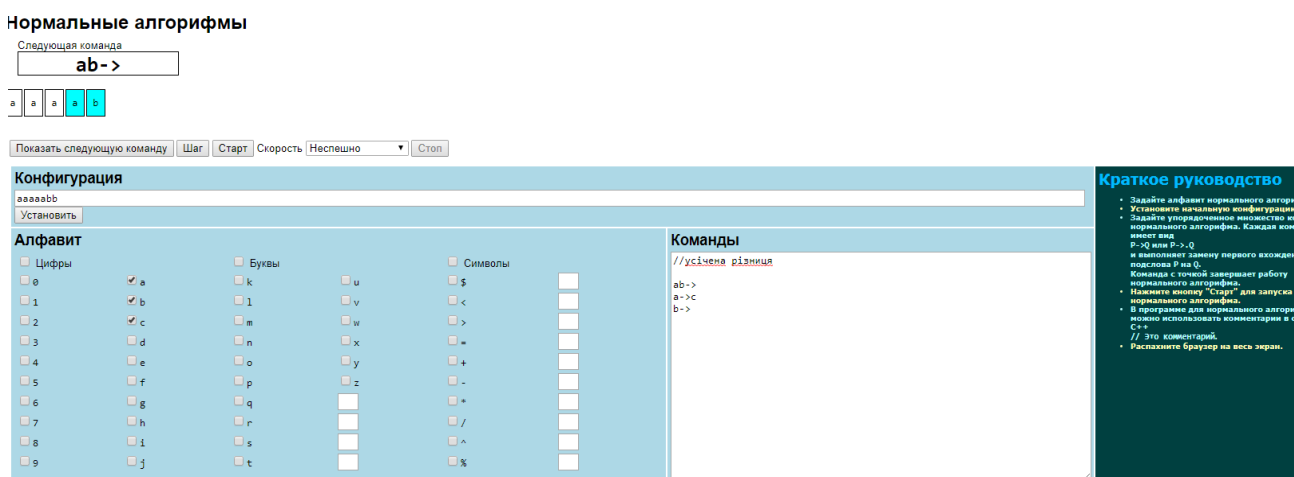


Рис. 10

3. Эмулятор нормальных алгоритмов Маркова на htm-странице за адресою www.sch91.ru/inc/download.asp?id=137

На сторінці наведені короткі теоретичні відомості та три приклади нормальних алгоритмів Маркова. Ці приклади можна завантажити та виконати он-лайн.

Входное (выходное) слово:

Правила:

ab-> a->c b->	1. ab → 2. a → c 3. b →
---------------------	-------------------------------

Рис. 11

Проста підстановка $\alpha \rightarrow \beta$ в емуляторі позначена $\alpha \rightarrow \beta$. Заключна підстановка в емуляторі позначена як $\alpha \Rightarrow \beta$ (записується в правому стовпчику як $\alpha \mapsto \beta$).

Зовнішній вигляд емулятора, який розташовано в північно-західному куті htm-сторінки, наведено на рис. 11.

Наведено нормальний алгоритм Маркова для обчислення усіченої різниці.

4. Емулятор нормальних алгоритмів Маркова на сайті <https://freeshell.de/~jcm/index.php?nav=projects>. Можна завантажити zip-архів з сайту або виконати НА он-лайн. Зовнішній вигляд емулятора наведено на рис. 12.

educational markov algorithm interpreter

rules definition or information

```
ab->
a->c
b->
```

input definition

```
aabbbbb
```

substitution output

```
[01|01] abbbb
[02|01] bbb
[03|03] bb
[04|03] b
[05|03]
```

main control

•

• auto pause substitution in case of

☐ a number of substitutions was made

• ☐ 1

☐ 1

☐ length of resulting word became

larger than

☐ resulting word became equal to

statistics

• number of substitutions

• substitution status

debug switches

• add specified info to substitution output

☒ substitutions since start

☐ substitutions since continue

☐ length of resulting word

Рис. 12

5. Машина натуральнозначних регістрів

Машина натуральнозначних регістрів (МНР, **Register machine**) – ідеалізована модель комп'ютера. МНР містить нескінченну кількість регістрів, вмістом яких є натуральні числа. Регістри нумеруються (йменуються) натуральними числами, починаючи з 0, і позначимо їх $R_0, R_1, \dots, R_n, \dots$. Вміст регістру R_n позначаємо за \bar{R}_n .

Послідовність $(\bar{R}_0, \bar{R}_1, \dots, \bar{R}_n, \dots)$ вмісту регістрів МНР назвемо **конфігурацією** машини натуральнозначних регістрів.

МНР може змінити вміст регістрів згідно з виконуваною нею командою. Скінченний упорядкований список команд утворює **програму** МНР. Команди програми послідовно нумеруємо натуральними числами, починаючи з 1.

МНР має чотири типи команд:

1. Обнулення n -го регістру:

$$Z(n): \bar{R}_n = 0.$$

2. Збільшення вмісту n -го регістру на 1:

$$S(n): \bar{R}_n = \bar{R}_n + 1.$$

3. Переадресація або копіювання вмісту регістру R_m у регістр R_n :

$$T(m, n): \bar{R}_n = \bar{R}_m \text{ (у такому разі } R_m \text{ не змінюється)}.$$

4. Умовний перехід: $J(m, n, q)$:

якщо $\bar{R}_n = \bar{R}_m$, то перейти до виконання q -ї команди, інакше – виконувати наступну за списком команду програми. Число q у команді $J(m, n, q)$ назвемо адресом переходу.

Команди типів 1–3 називають **арифметичними**. Після виконання арифметичної команди МНР має виконувати наступну за списком команду програми.

Виконання однієї команди МНР назвемо **кроком** МНР.

Роботу програми МНР починає, перебуваючи в деякій **початковій конфігурації**, з виконання першої за списком команди. Наступну команду для виконання програми визначено так, як описано вище. Виконання програми завершується (програма зупиняється), якщо наступна для виконання команда відсутня (тобто номер наступної команди перевищує номер останньої команди програми). Конфігурацію МНР у момент завершення виконання програми називають **заключною**, вона визначає результат роботи МНР-програми над даною початковою конфігурацією.

Якщо МНР-програма P у роботі над початковою конфігурацією $K_0 = (a_0, a_1, \dots)$ ніколи не зупиняється, цей факт позначатимемо $P(a_0, a_1, \dots) \uparrow$, якщо ж зупиниться після скінченної кількості кроків, цей факт будемо позначати $P(a_0, a_1, \dots) \downarrow$.

Якщо МНР-програма P у роботі над початковою конфігурацією (a_0, a_1, \dots) зупиняється із заключною конфігурацією (b_0, b_1, \dots) , цей факт позначатимемо так: $P(a_0, a_1, \dots) \downarrow (b_0, b_1, \dots)$.

Конфігурацію виду $(a_0, a_1, \dots, a_n, 0, \dots, 0, \dots)$, у якій $\bar{R}_m = 0$ для всіх $m > n$, будемо називати **скінченною**. Таку конфігурацію будемо позначати (a_0, a_1, \dots, a_n) . Якщо МНР-програма P починає роботу над скінченною початковою конфігурацією, то в процесі виконання програми P МНР перебуватиме тільки в скінченних конфігураціях.

МНР-програми P та Q назвемо **еквівалентними**, якщо вони визначають однакові відображення послідовностей натуральних чисел. Це означає, що у процесі роботи над однаковими початковими конфігураціями вони або обидві зупиняються з однаковими заключними конфігураціями, або обидві не зупиняються.

МНР-програма P **обчислює** часткову n -арну функцію $f : N_0^{n+1} \rightarrow N_0$, якщо:

- за умови $(a_0, a_1, \dots, a_n) \in D_f$ та $f(a_0, a_1, \dots, a_n) = b$ маємо $P(a_0, a_1, \dots, a_n) \downarrow b$;
- за умови $(a_0, a_1, \dots, a_n) \notin D_f$ маємо $P(a_0, a_1, \dots, a_n) \uparrow$.

Це означає, що значення аргументів функції послідовно розміщуються у регістрах, починаючи з R_0 , а значення функції записується в регістр R_0 .

Функцію називають **МНР-обчислюваною**, якщо існує МНР-програма, яка обчислює цю функцію.

Розглянемо приклади МНР-програм для деяких функцій та предикатів.

Приклад 1. МНР-програма для обчислення предиката

$$P(x, y) = \begin{cases} 1, & \text{якщо } x = y, \\ 0, & \text{якщо } x \neq y \end{cases}.$$

- 1) $J(0, 1, 3)$
- 2) $J(0, 0, 4)$
- 3) $S(2)$
- 4) $T(2, 0)$

Приклад 2. МНР-програма для функції $f(x, y) = x + y$.

- 1) $J(1, 2, 5)$
- 2) $S(0)$
- 3) $S(2)$
- 4) $J(0, 0, 1)$

Приклад 3. МНР-програма для функції $f(x) = 2x$.

- 1) $T(0,1)$
- 2) $J(1,2,6)$
- 3) $S(0)$
- 4) $S(2)$
- 5) $J(0,0,2)$

Приклад 4. МНР-програма для функції $f(x, y) = x \div y$.

- 1) $J(0,1,7)$
- 2) $J(0,2,6)$
- 3) $S(1)$
- 4) $S(2)$
- 5) $J(0,0,1)$
- 6) $Z(2)$
- 7) $T(2,0)$

6. Емулятор машини натуральнозначних регістрів

Емулятор машини натуральнозначних регістрів. Зовнішній вигляд емулятора наведено на рис. 13.

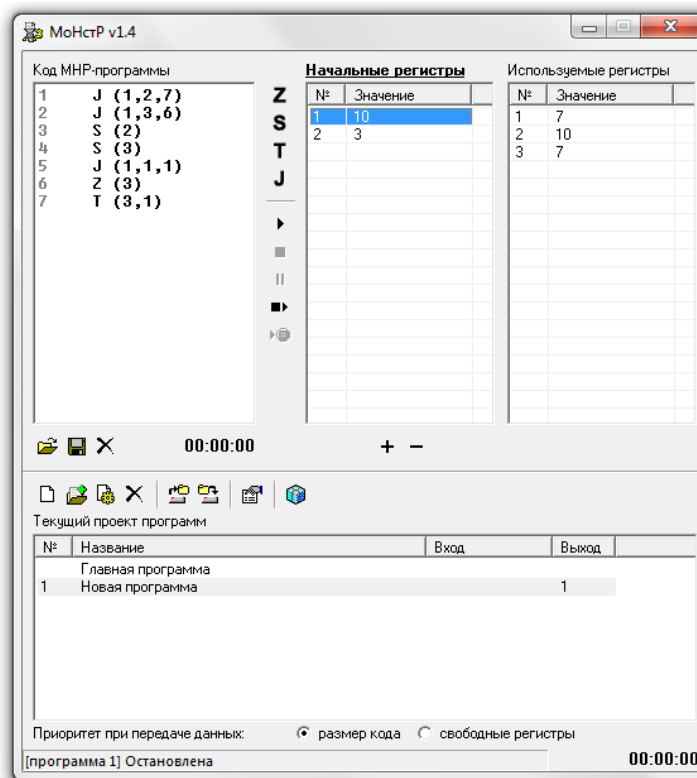


Рис. 13

Емулятор має інтуїтивно зрозумілий інтерфейс. Вікно «Код МНР-програми» може заповнюватися он-лайн або можна завантажити програму з файлу, також є можливість зберігати код програми.

Особливістю емулятора є те, що нумерація регістрів починається з одиниці, а не з нуля. Це треба враховувати при написанні програми.

Початкові дані вводяться послідовно починаючи з першого регістра. Для введення потрібно натиснути на «+» в полі (рис. 14), нижче за вікно «Початкові регістри», і в спадаючому вікні ввести значення із множини N_0 . Видалення даних проходить шляхом вибору відповідного регістру та натискання на «-».

Запуск здійснюється з панелі команд. Так само там є кнопка для покрокового запуску програми користувача.

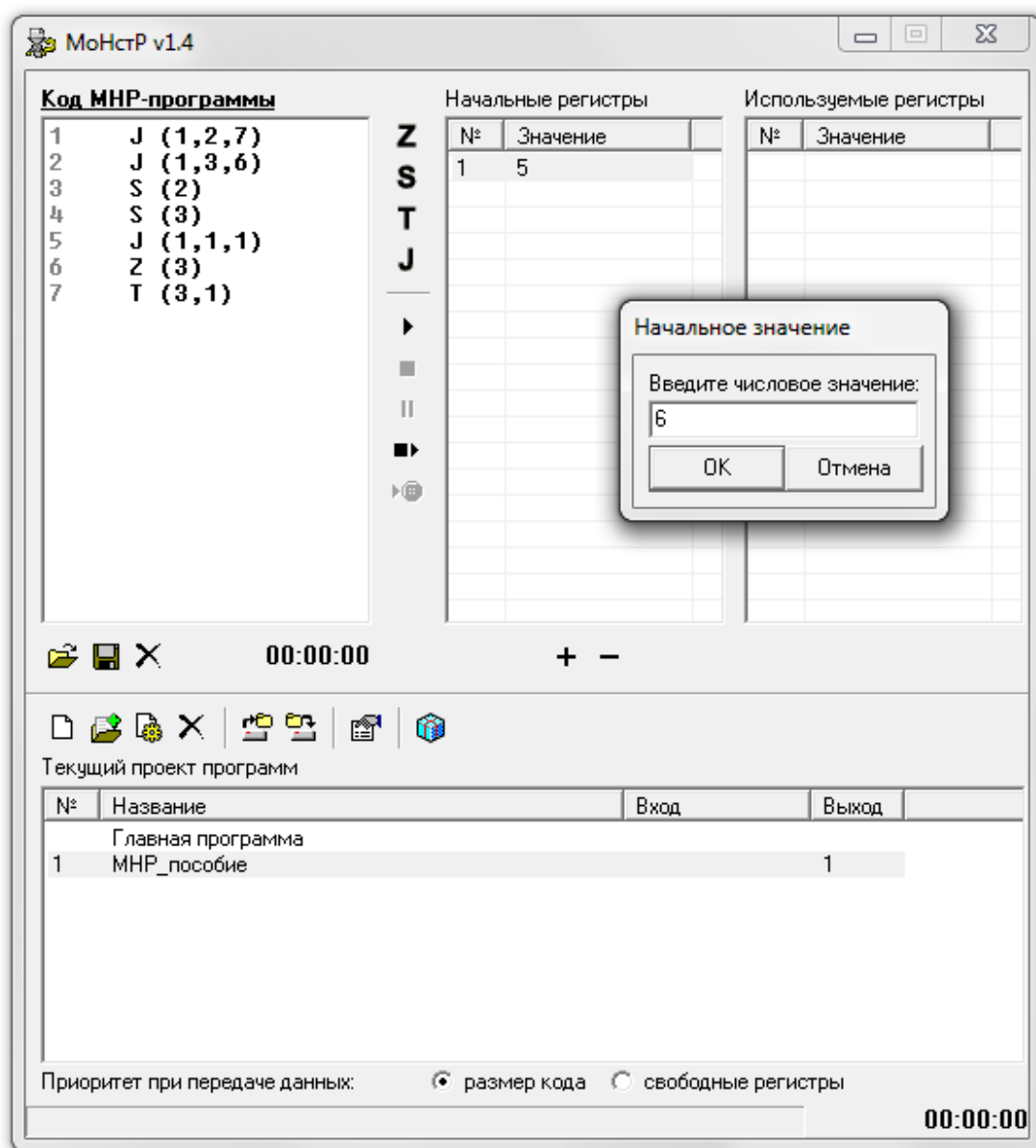


Рис. 14

7. Скінченні автомати Мілі та Мура

Теорія автоматів – це розділ теорії керуючих систем, що вивчає математичні моделі перетворювачів дискретної інформації, які називають **автоматами**. Такими перетворювачами є як реальні пристрої (обчислювальні машини, живі організми), так і абстрактні системи (наприклад, формальна система – це сукупність абстрактних об'єктів, не пов'язаних із зовнішнім світом, в якому представлені правила оперування множиною символів у строгому синтаксичному трактуванні без урахування смислового змісту, тобто семантики; аксіоматичні теорії, що описують певну сукупність явищ у їхньому причинно-наслідковому зв'язку одне з одним).

Найбільш тісно теорія автоматів пов'язана з теорією алгоритмів. Це можна пояснити тим, що автомат перетворює дискретну інформацію по кроках у дискретні моменти часу і формує результуючу інформацію по кроках заданого алгоритму. Ці перетворення можливі за допомогою технічних та/або програмних засобів.

Автомат можна зобразити як деякий пристрій (чорний ящик), на який подають вхідні сигнали і знімають вихідні і який може мати деякі внутрішні стани. Такий автомат називають **абстрактним**, оскільки абстрагуються від реальних фізичних вхідних і вихідних сигналів, розглядаючи їх просто як букви деякого алфавіту і в зв'язку з ідеалізованим дискретним часом. Особливе місце в теорії автоматів займає поняття скінченного автомата.

Скінченний автомат (СА) – математична абстракція, що дозволяє описувати зміни стану об'єкта залежно від його поточного стану та вхідних даних за умови що загальна можлива кількість станів і множина вхідних сигналів скінченні. Головною перевагою скінченних автоматів є те, що в них природним чином описано системи, які є керовані зовнішніми подіями.

Скінченим автоматом Мілі називають систему $\mathcal{A} = \langle S, X, Y, s_0, \delta, \lambda \rangle$, у якій S – скінченна множина станів, X – скінченна множина, яку називають вхідним алфавітом, Y – скінченна множина, яку називають вихідним алфавітом, $\delta: S \times X \rightarrow S$ – функція переходів, $\lambda: S \times X \rightarrow Y$ – функція виходів, $s_0 \in S$ – виділений елемент, який називають початковим станом.

Скінчений автомат можна задати орієнтованим графом, в якому вершини відповідають станам, а ребро зі стану s в стан q з поміткою x/y проводиться тоді, коли СА зі стану s під дією вхідного сигналу x переходить в стан q з вихідною реакцією y . Для автомата з множиною станів $S = \{s_0, s_1, s_2, s_3\}$ та вхідним і вихідним алфавітами $X = \{0, 1\}$, $Y = \{y_0, y_1, y_2, y_3, y_4, y_5\}$ маємо таке графічне представлення (рис. 15):

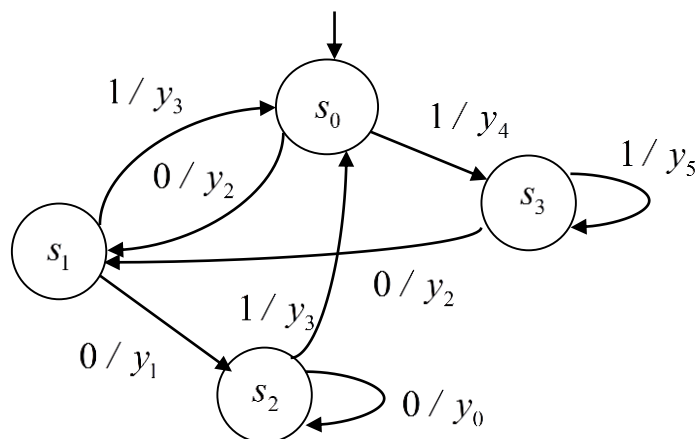


Рис. 15

Окрім графічного представлення для СА можна використовувати табличне, задаючи функції переходів та виходів у вигляді таблиць. Для прикладу, який наведено на рис. 15, маємо табл. 5 та 6:

Таблиця 5

Функція переходів δ		
вхід \ стан	0	1
s_0	s_1	s_3
s_1	s_2	s_0
s_2	s_2	s_0
s_3	s_1	s_3

Таблиця 6

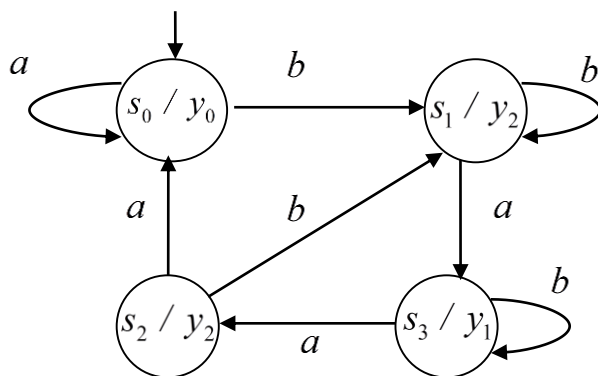
Функція виходів λ		
вхід \ стан	0	1
s_0	y_2	y_4
s_1	y_1	y_3
s_2	y_0	y_3
s_3	y_2	y_5

Іноді зручним стає більш просте визначення автомата, в якому функція переходів задає наступний стан, а вихід автомата залежить лише від його поточного стану. Такі автомати названо **скінченними автоматами Мура**. Доведено, що обидва способи визначення скінченного автомата є еквівалентні в тому розумінні, що будь-який автомат Мілі може бути перетворено на відповідний автомат Мура й навпаки.

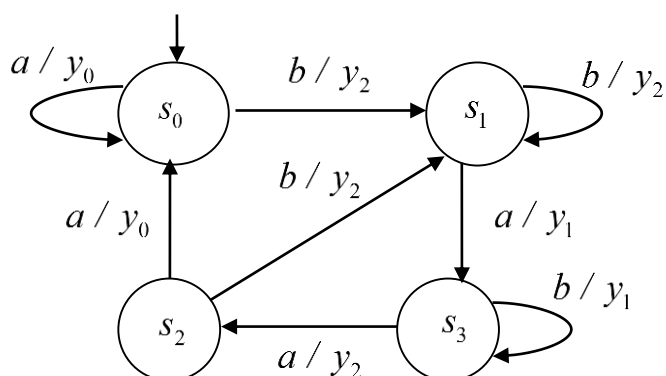
Отже, скінченний автомат Мура – це система $\mathcal{A} = \langle S, X, Y, s_0, \delta, \lambda \rangle$, у якій S – скінченна множина станів, X – скінченна множина символів вхідного алфавіту, Y – скінченна множина символів вихідного алфавіту, $\delta: S \times X \rightarrow S$ – функція переходів, $\lambda: S \rightarrow Y$ – функція виходів, $s_0 \in S$ – початковий стан.

Для скінченного автомата Мура функція виходів λ визначається не за парою (стан, вхідний сигнал) як у скінченного автомата Мілі, а тільки за станом, вихідний сигнал дописується до стану.

На рис. 16 наведено скінченні автомати Мілі та Мура, які розв'язують одну й ту ж задачу.



Скінченний автомат Мура



Скінченний автомат Мілі

Рис. 16

Приклад. Скінчений автомат, який продає квитки. Нехай на вхід скінченному автомату поступають в будь-якій послідовності і з будь-яким числом повторень жетони вартістю 1, 2, 3 грн. Автомат продає квиток, якщо сума введених жетонів дорівнює 3 грн. У випадку перевищення суми автомат повертає гроші.

Вхідний алфавіт: $X = \{1, 2, 3\}$, вихідний алфавіт: $Y = \{Б, Н, В\}$, де Б – білет, Н – нічого не видає, В – повертає гроші.

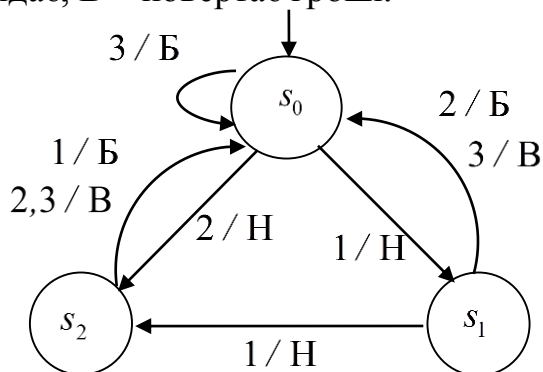


Рис. 17

Внутрішні стани будемо ототожнювати з сумою грошей, яку пам'ятає скінченний автомат. Після продажу квитка або повернення грошей автомат пам'ятає нульову суму. Множина внутрішніх станів $S = \{s_0, s_1, s_2\}$. Графічне представлення СА Мілі наведено на рис. 17, табличне представлення має такий вигляд (табл. 7, 8):

Таблиця 7

Функція переходів δ			
вхід \ стан	1	2	3
s_0	s_1	s_2	s_0
s_1	s_2	s_0	s_0
s_2	s_0	s_0	s_0

Таблиця 8

Функція виходів λ			
вхід \ стан	1	2	3
s_0	Н	Н	Б
s_1	Н	Б	В
s_2	Б	В	В

Скінчений автомат Мура для цієї задачі наведено на рис. 18.

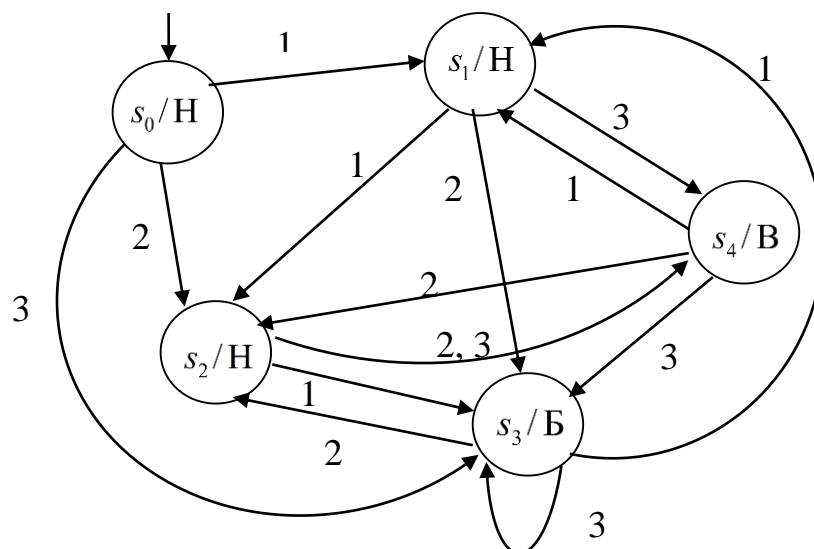


Рис. 18

8. Програма JFLAP

JFLAP (англ. *Java Formal Languages and Automata Package*) – вільна кроссплатформенна програма експериментів з різними об'єктами, що зустрічаються під час вивчення формальних мов. Розробляється Університетом Дюка (приватний дослідницький університет, розташований в місті Дарем, Північна Кароліна, США). Сайт <http://www.jflap.org/>.

Можливості програми:

- моделює машину Тьюрінга, в тому числі багатострічкову;
- моделює скінчений автомат Мілі;
- моделює скінчений автомат Мура;

- моделює магазинний автомат;
- демонструє лему про накачку
 - для регулярних,
 - та контекстно-вільних мов;
- дає графічне представлення недетермінованого та детермінованого скінченних автоматів;
- вміє покроково проводити:
 - перетворення регулярного виразу в недетермінований скінченний автомат,
 - детермінізацію недетермінованого скінченного автомату,
 - мінімізацію детермінованого скінченного автомату.

1. Машина Тьюрінга

Щоб запустити нову однострічкову машину Тьюрінга, в головному меню виберіть пункт **Turing Machine**, або в деяких версіях програми **Multi-Tape Turing Machine** та вкажіть одну стрічку (рис. 19, а, б, в).

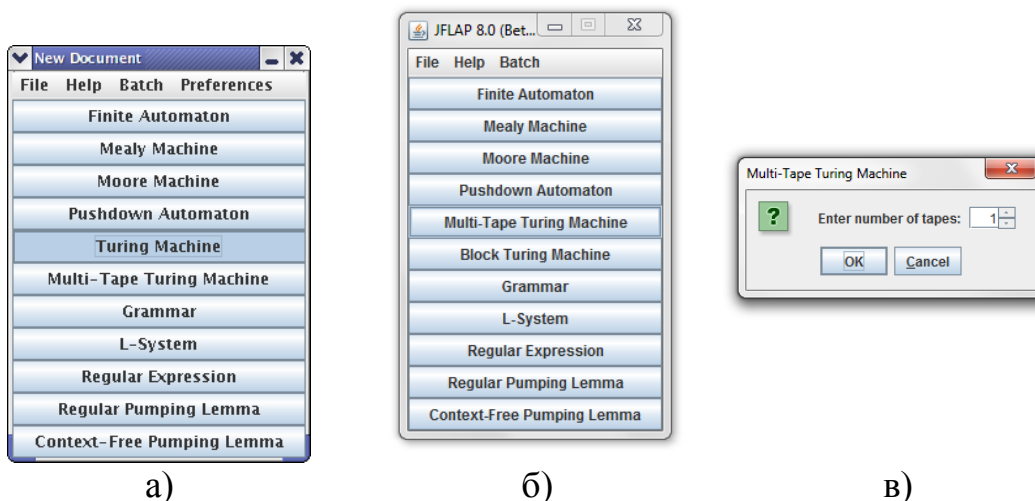


Рис. 19

Після цього відкриється нове вікно, в якому ви зможете створювати і редагувати МТ. Редактор розділений на дві основні частини: поле, на якому ви можете створювати свою МТ, і панель інструментів (рис. 20).

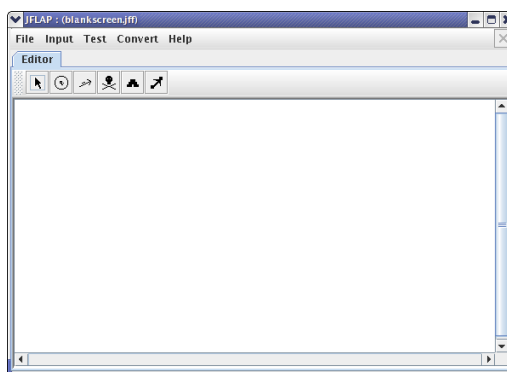


Рис. 20

- Розглянемо панель інструментів (рис. 21). Панель містить 4 інструменти:
- 1) *Редактор властивості стану* – встановлює початковий і заключний стани;
 - 2) *Створення станів* – створює стан;
 - 3) *Створення переходів* – створює перехід між станами;
 - 4) *Інструмент видалення* – дозволяє видалити обраний стан або перехід.

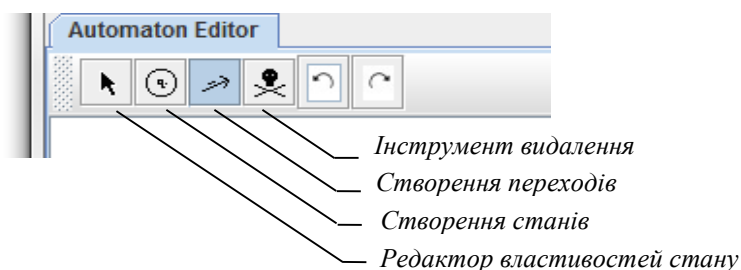


Рис. 21

Продемонструємо роботу програми на прикладі обчислення функції усіченої різниці $f(x, y) = x \div y$ (табл. 9). Програма JFLAP потребує використання символу, який розділяє аргументи функції, тому використано алфавіт $T = \{1, \#, \lambda\}$, де аргумент функції задається як $1^{x+1} \# 1^{y+1}$, λ – порожній символ, $\#$ – символ, який розділяє аргументи функції. Результатом роботи МТ буде $1^{f(x,y)+1}$.

Таблиця 9

	q_1	q_2	q_3	q_4	q_5
λ		$q_3 \lambda L$		$q_1 \lambda R$	$q_0 \lambda S$
1	$q_2 \lambda R$	$q_2 1 R$	$q_4 \lambda L$	$q_4 1 L$	$q_5 \lambda R$
#	$q_5 \lambda R$	$q_2 \# R$	$q_0 1 S$	$q_4 \# L$	

Додамо 8 станів, де q_1 буде початковим станом, а q_0 – заключним (рис. 22, а, б, рис. 23). Машину Тьюрінга представляємо у вигляді графа.

Створення переходів. При створенні переходу з'явиться наступне вікно, в якому будуть три поля (рис. 24). Значення у першому полі є поточне значення під голівкою читання-запису машини Тьюрінга. Наступне поле містить значення, на яке буде замінено на стрічці значення з першого поля, коли цей крок буде опрацьовано. Розмір цих полів обмежений одним символом. Останнє поле показує, куди буде рухатися голівка читання-запису вздовж стрічки після обробки кроку: R (вправо), L (вліво), S (на місці).

Для створення переходу від стану q_1 до стану q_2 виберіть інструмент *Створення переходів*, натисніть і утримуйте ліву кнопку миші на q_1 і перетягніть курсор до q_2 . Заповніть текстове поле.

В результаті отримаємо таке представлення машини Тьюрінга (рис. 25).




Рис. 23

\square	\square	R

Рис. 24

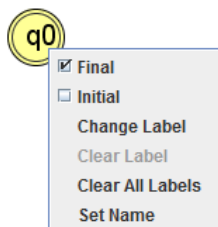


Рис. 22, б

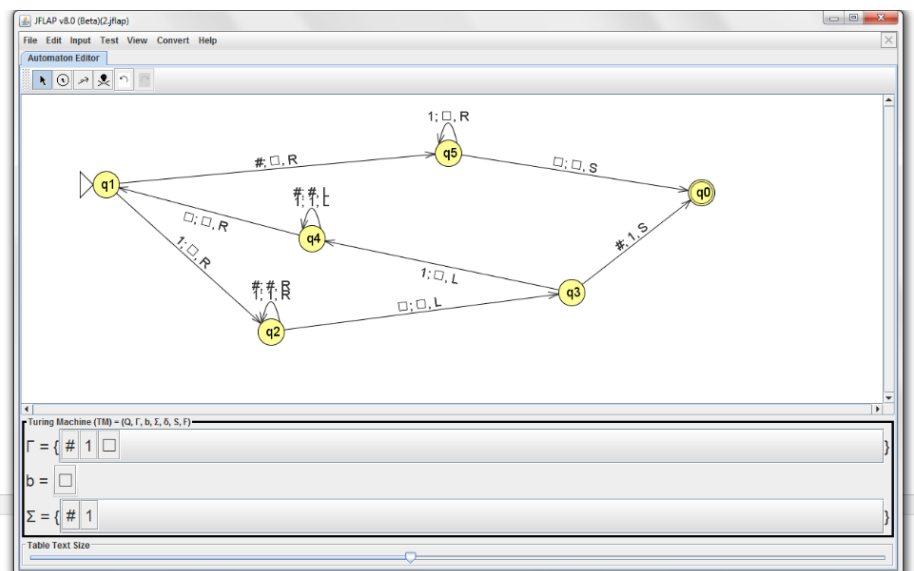
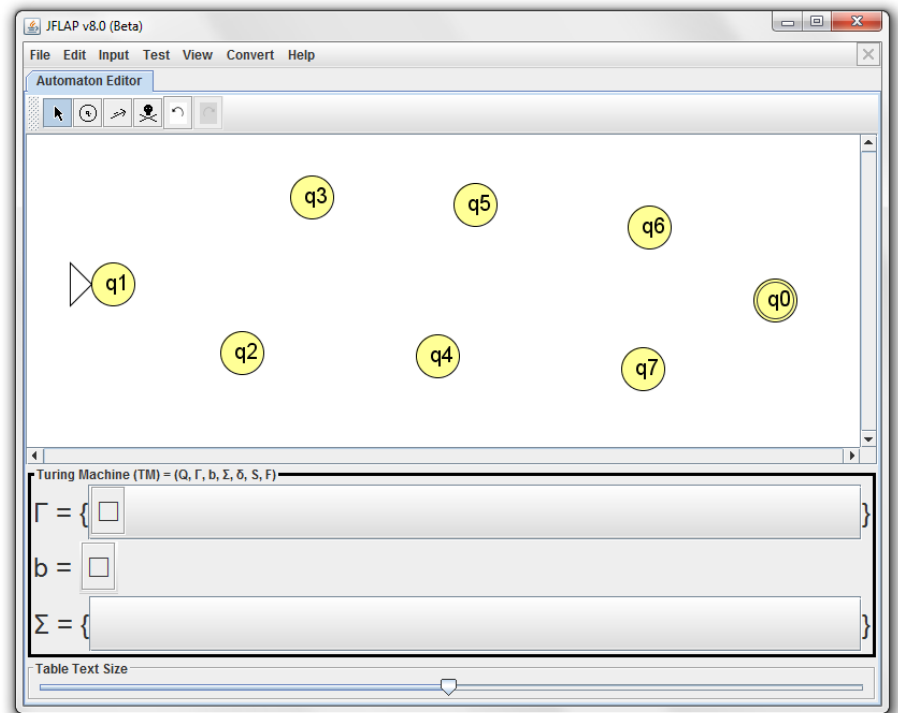


Рис. 25

30

В меню *Input* виберемо опцію *Step*. Стан обох стрічок на кожному кроці наводиться у віконці в нижньому лівому куті. На рис. 28 показано результат роботи МТ над заданим словом.

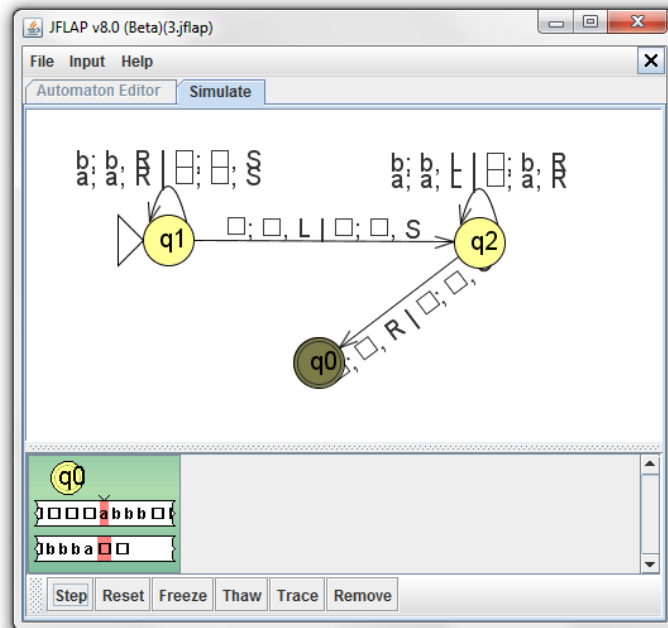


Рис. 28

Нехай машина Тьюрінга має три стрічки. Побудуємо МТ, яка знаходить добуток двох чисел в унарному коді. Алфавіті $T = \{1\}$, перше число записано на першій стрічці, а друге – на другій стрічці. Результат роботи програми МТ записується на третю стрічку. Програму МТ наведено на рис. 29.

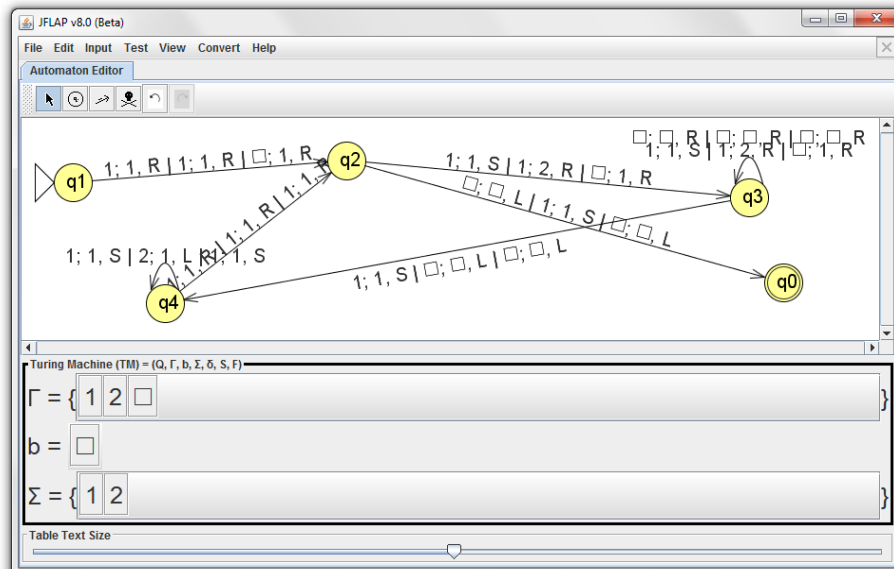


Рис. 29

Для зручності наведемо опис алгоритму і список команд цієї МТ. Для знаходження $f(x, y)$, де x записується на першій стрічці як 1^{x+1} , а y як 1^{y+1} ,

використаємо розширений алфавіт $T' = \{1, 2\}$. На першій і другій стрічках проходимо перші одиниці. На другій стрічці замінюємо 1 на 2 і записуємо 1 на третю стрічку. Якщо на другій стрічці 1 (окрім першої) замінено на 2, рухаємось ліворуч та відновлюємо 1. Завершуємо роботу МТ так: на першій та третій стрічках керівний пристрій з головкою читання/запису знаходяться на останньому символі, а на другій стрічці – на першому.

$$\begin{aligned} q_1 \{1, 1, \lambda\} &\rightarrow q_2 \{1, 1, \lambda\} \{R, R, R\} \\ q_2 \{1, 1, \lambda\} &\rightarrow q_3 \{1, 2, 1\} \{S, R, R\} \\ q_3 \{1, 1, \lambda\} &\rightarrow q_3 \{1, 2, 1\} \{S, R, R\} \\ q_3 \{1, \lambda, \lambda\} &\rightarrow q_4 \{1, \lambda, \lambda\} \{S, L, L\} \\ q_4 \{1, 2, 1\} &\rightarrow q_4 \{1, 1, 1\} \{S, L, S\} \\ q_4 \{1, 1, 1\} &\rightarrow q_2 \{1, 1, 1\} \{R, R, R\} \\ q_2 \{\lambda, 1, \lambda\} &\rightarrow q_0 \{\lambda, 1, \lambda\} \{L, L, L\} \end{aligned}$$

На рис. 30 показано результат роботи МТ для $f(2, 3)$.

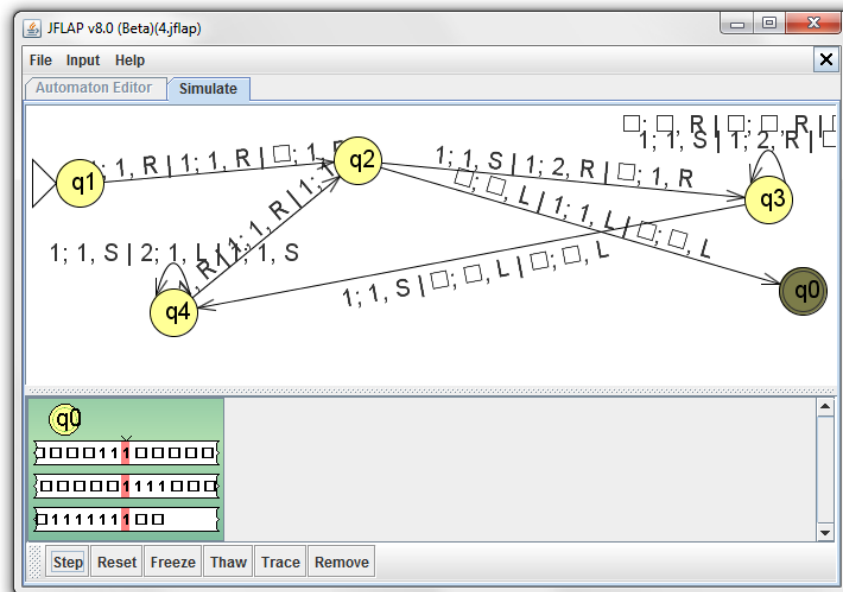


Рис. 30

2. Скінченний автомат Мілі.

Для створення нового СА Мілі, виберіть в головному меню пункт **Mealy Machine** (рис. 19, б).

Панель інструментів наведено на рис. 21. Вибираємо інструмент *Редактор властивостей стану*, і натискаємо правою кнопкою миші на обраний стан, з'являється спливаюче меню (рис. 31):

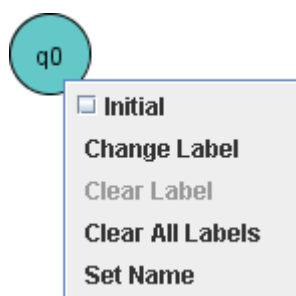


Рис. 31

Позначимо стан q_0 як початковий.

Перетворення вхідного слова. Автомат Мілі виробляє вихідне слово при кожному переході. Коли створюється перехід, з'являються дві незаповнені області. Перша область – для вхідного символу, а друга – для вихідного символу.

Приклад створеного скінченного автомату Мілі наведено на рис. 32. Його таблиці переходів та виходів такі:

Таблиця 10

Функція переходів δ		
вхід \ стан	a	b
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_0	q_1

Таблиця 11

Функція виходів λ		
вхід \ стан	a	b
q_0	0	1
q_1	0	0
q_2	1	0

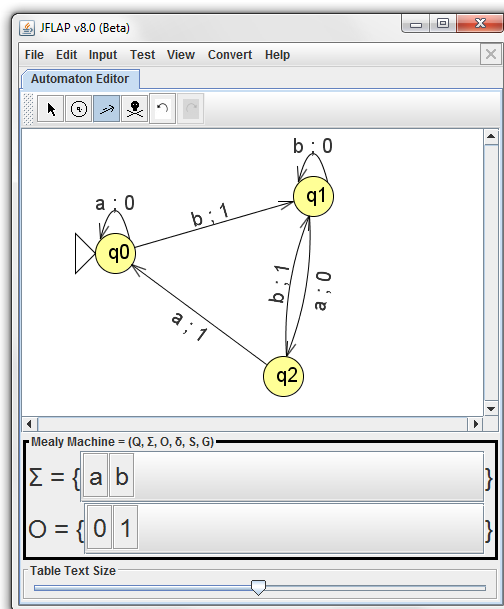


Рис. 32

В меню *Input* можна вибрати або опцію *Step* або опцію *Multiple Run*.

У першому випадку для введення вхідного слова з'явиться вікно, в якому безпосередньо можна ввести вхідне слово або відкрити підготовлений

файл (рис. 33). Далі відкривається вікно виконання. Опція *Step* дозволяє відстежувати роботу СА крок за кроком. При цьому на кожному кроці виконання скінченного автомату Милі оновлюється текстове поле, в якому відображається вихідне слово, та кольором відзначається стан, в якому знаходиться СА (рис. 34).

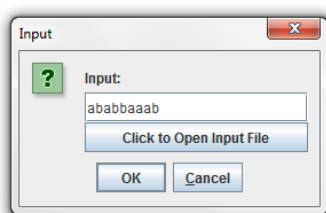


Рис. 33

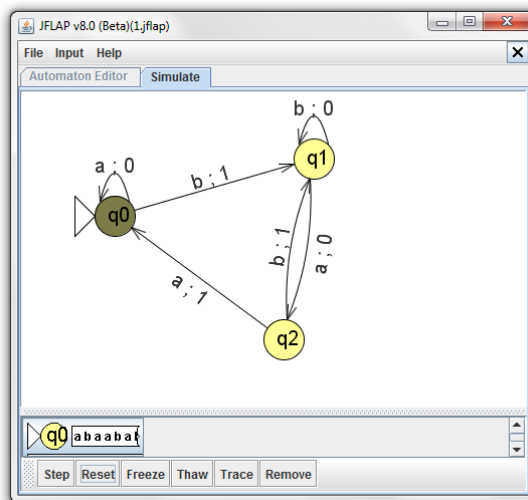


Рис. 34

Якщо вибрано опцію *Multiple Run*, то вхідні та вихідні слова будуть відображені у відповідних колонках (рис. 35).

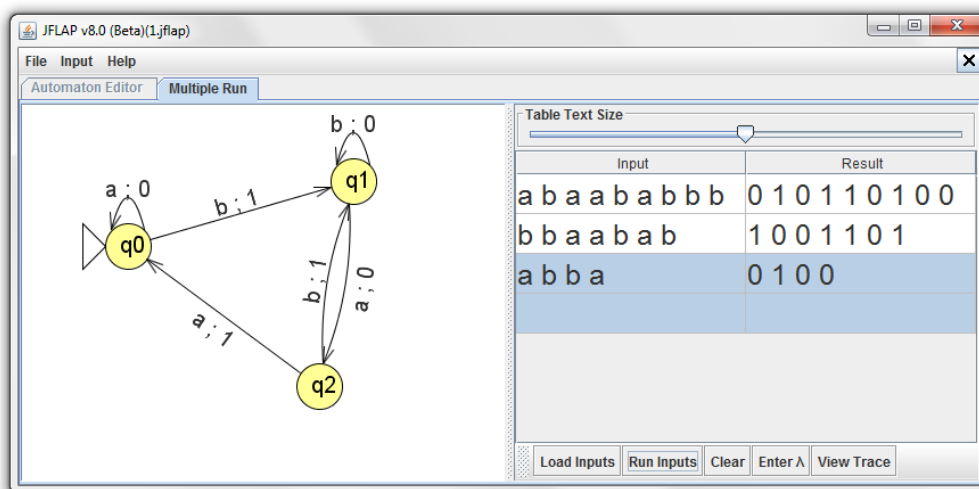


Рис. 35

Є можливість проглянути всі конфігурації за допомогою опції *View Trace*.

3. Скінченний автомат Мура.

Для створення нового СА Мура, виберіть в головному меню пункт **Moore Machine** (рис. 19, б).

При створенні стану з'являється діалогове вікно, в яке потрібно ввести вихідний символ стану (рис. 36). Введіть символ і натисніть *OK* (якщо ви натиснете *Cancel*, то стан все одно створиться, просто він буде мати порожній

вихідний символ). Коли стан буде створено, його вихід буде відображатися в правому верхньому кутку. Тобто для скінченного автомату Мура вихідний сигнал дописується до стану.

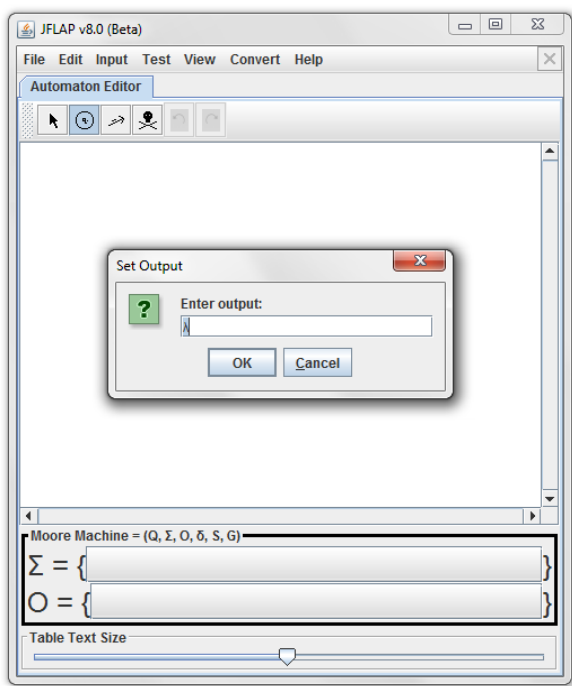


Рис. 36

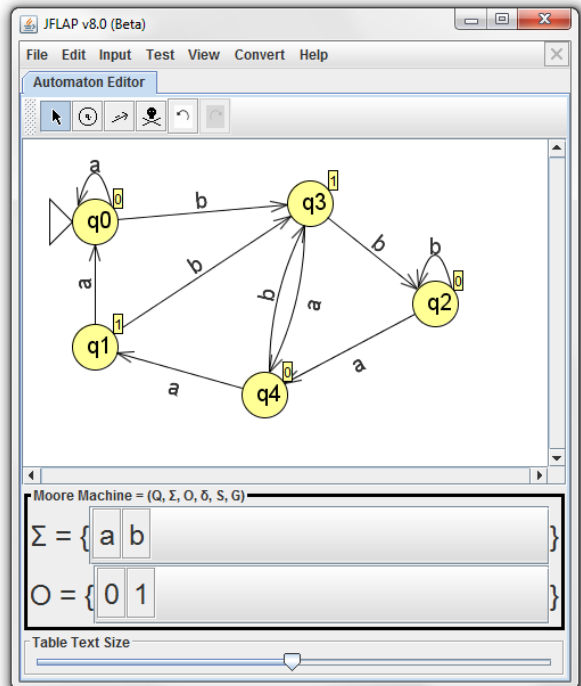


Рис. 37

На рис. 37 показано скінченний автомат Мура, який є еквівалентний скінченному автомату Мілі, наведеному на рис. 34, 35. Його таблиця переходів та виходів така:

Таблица 12

вихідний символ	стан	вхідні символи	
		<i>a</i>	<i>b</i>
0	q_0	q_0	q_3
1	q_1	q_0	q_3
0	q_2	q_4	q_2
1	q_3	q_4	q_2
0	q_4	q_1	q_3

Результати роботи скінченного автомату Мура наведено на рис. 38 для тих же вхідних даних, що показані на рис. 35.

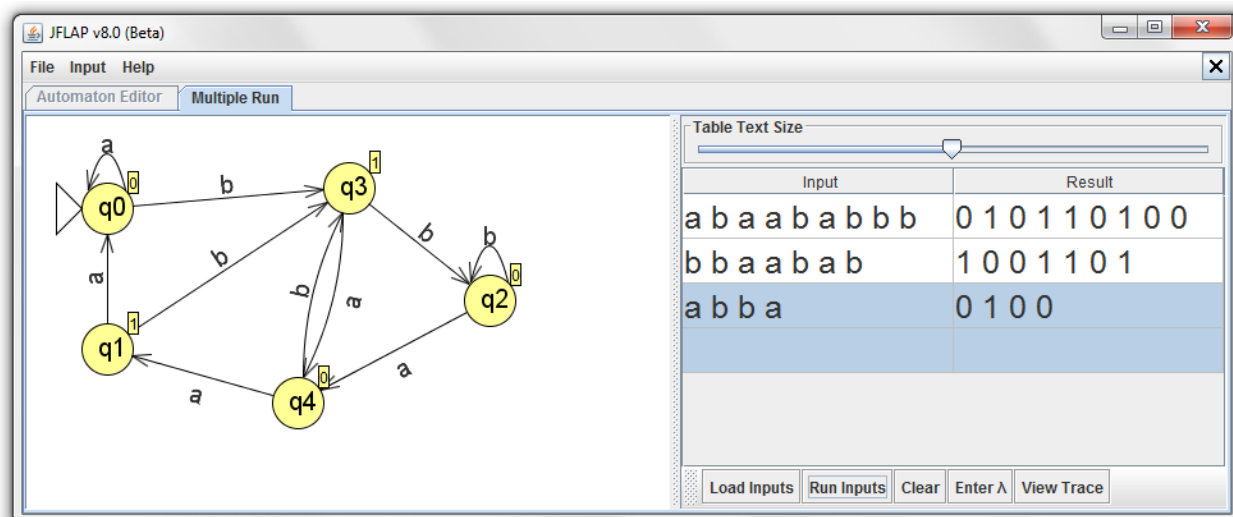


Рис. 38

Список використаної літератури

1. Ахо А. Структуры данных и алгоритмы / А. Ахо, Дж. Хопкрофт, Дж. Ульман – М.: Изд. дом «Вильямс», 2001. – 384 с.
2. Верещагин Н. К. Лекции по математической логике и теории алгоритмов: В 3 ч. / Н. К. Верещагин, А. Шень – М.: МЦНМО, 2000. – Ч. 2. Языки и исчисления. – 288 с.
3. Вирт Н. Алгоритмы и структуры данных / Н. Вирт – СПб.: Невский диалект, 2001. – 352 с.
4. Гаврилов Г. П. Задачи и упражнения по дискретной математике: Учеб. пособие. 3-е изд., перераб. / Г. П. Гаврилов, А. А. Сапоженко – М.: Физматлит, 2005. – 416 с.
5. Гиндикин С. Г. Алгебра логики в задачах / С. Г. Гиндикин. – М.: Наука, 1972. – 288 с.
6. Игошин В. И. Математическая логика и теория алгоритмов: учеб. пособие для студ. высш. учеб. заведений / В. И. Игошин – М.: «Академия», 2004. – 448 с.
7. Игошин В. И. Задачи и упражнения по математической логике и теории алгоритмов: учеб. пособие для студ. высш. учеб. заведений / В. И. Игошин – М.: «Академия», 2006. – 304 с.
8. Карпов Ю. Г. Теория автоматов / Ю. Г. Карпов. – СПб.: Питер, 2002. – 224с.
9. Крупский В. Н. Теория алгоритмов: учебное пособие для студентов вузов / В. Н. Крупский, В. Е. Плиско. – М.: «Академия», 2009. – 208 с.
10. Кузнецов О. П. Дискретная математика для инженера [Текст] / О. П. Кузнецов, Г. М. Андельсон-Вельский. – М.: Энергоатомиздат, 1988. – 480 с.

11. Лавров И. А. Задачи по теории множеств, математической логике и теории алгоритмов / И. А. Лавров, Л. Л. Максимова. – М.: Физматлит, 2004. – 256 с.
12. Лісовик Л. П. Теорія алгоритмів / Л. П. Лісовик, С. С. Шкільняк – К., 2003. – 84 с.
13. Мальцев А. И. Алгоритмы и рекурсивные функции / А. И. Мальцев. – М.: Наука, 1986. – 368 с.
14. Марков А. А. Теория алгорифмов / А. А. Марков, Н. М. Нагорный. – М.: Наука, 1984. – 432 с.
15. Мендельсон Э. Введение в математическую логику / Э. Мендельсон. – М., 1976.
16. Нікітченко М. С. Математична логіка та теорія алгоритмів / М. С. Нікітченко, С. С. Шкільняк. – К., 2008. – 64 с.
17. Нікітченко М. С. Основи математичної логіки / М. С. Нікітченко, С. С. Шкільняк. – К., 2006.
18. Новиков Ф. А. Дискретная математика для программистов / Ф. А. Новиков. – СПб.: Питер, 2000. – 304 с.
19. Пильщиков В. Н. Машина Тьюринга и алгоритмы Маркова: учебно-методическое пособие / В. Н. Пильщиков, В. Г. Абрамов, А. А. Вылиток, И. В. Горячая. – М.: МГУ, 2006. – 47 с.
20. Подзоров С. Ю. Теория алгоритмов: полный конспект лекций по курсу / С. Ю. Подзоров. – М.: НГУ, 2004. – 130 с.
21. Роджерс Дж. Теория алгоритмов и эффективная вычислимость / Дж. Роджерс. – М.: Наука, 1972. – 624 с.
22. Ульянов М. В. Математическая логика и теория алгоритмов: в 2 ч. Ч. 2. Теория алгоритмов / М. В. Ульянов, М. В. Шептунов. – М.: МГАПИ, 2003. – 80 с.
23. Успенский В. А. Теория алгоритмов: основные открытия и приложения [Текст] / В. А. Успенский, А. Л. Семенов – М.: Наука., 1987. – 288 с.
24. Шапорев С. Д. Математическая логика. Курс лекций и практических занятий / С. Д. Шапорев. – СПб.: БХВ-Петербург, 2005. – 416 с.
25. Шкільняк С. С. Теорія алгоритмів. Приклади та задачі: Навчальний посібник / С. С. Шкільняк. – Київ: Видавничо-поліграфічний центр "Київський університет", 2012. – 77 с.
26. Яблонский С. В. Введение в дискретную математику [Текст] / С. В. Яблонский. – М.: Наука, 1986. – 384 с.

Зміст

Вступ.....	3
1. Машина Тьюрінга	4
2. Емулятори машини Тьюрінга.....	9
3. Нормальні алгоритми Маркова.....	14
4. Емулятори нормальних алгоритмів Маркова.....	16
5. Машини натуральнозначних регістрів.....	20
6. Емулятор машини натуральнозначних регістрів.....	22
7. Скінченні автомати Мілі та Мура	24
8. Програма JFLAP.....	27
Список використаної літератури.....	37

Навчальне видання

Алла Євгенівна Шевельова

КОМП'ЮТЕРНИЙ ПРАКТИКУМ З ТЕОРІЇ АЛГОРИТМІВ

Підписано до друку 15.06.18. Формат 60×84/16. Папір друкарський. Друк плоский. Ум. друк. арк. 3,7. Обл.-вид. арк. 3,3. Ум. фарбовідб. 3,7. Тираж 30 прим. Зам. №

Видавництво «Ліра», вул. Наукова, 5, м. Дніпро, 49038.
ДК №6042 від 26.02.2018 р.