

Automaty i Języki Formalne

Soma Dutta

Wydział Matematyki i Informatyki, UWM w Olsztynie
soma.dutta@matman.uwm.edu.pl

Wykład - 1: Wprowadzenie

Semestr zimowy 2023

Teoria obliczeń (Theory of computing)

- ▶ Założeniem jest zbudowanie abstrakcyjnych modeli komputerów i ich obliczeń.
- ▶ Obejmuje kilka tematów, np. **Teorię automatów** (automata theory), **języki formalne i gramatykę** (formal languages and grammar), **obliczalność** (computability), i **złożoność** (complexity).
 - ▶ Automaty, gramatyki i obliczalność można uznać za naukę o tym, co w zasadzie można zrobić za pomocą komputerów.
 - ▶ Wręcz przeciwnie, złożoność dotyczy tego, co można zrobić w praktyce.
- ▶ Abstrakcyjne modele komputerów i ich obliczenia muszą spełniać kilka podstawowych cech, które są wspólne zarówno dla sprzętu (**hardware**), jak i oprogramowania (**software**).
 - ▶ Do modelowania sprzętu komputerowego wprowadzamy **pojęcie automatu, który posiada wszystkie niezbędne cechy komputera cyfrowego**.
 - ▶ Z drugiej strony **język formalny jest abstrakcją ogólnej charakterystyki języków programowania**.
 - ▶ Pojęcie **obliczeń** można sformalizować podając precyzyjną definicję algorytmu.

Teoria automatów : historia

- ▶ Zanim pojawiły się komputery, w 1930 roku [Alan Turing](#), angielski matematyk, informatyk i logik, wprowadził [pojęcie abstrakcyjnej maszyny](#) (abstract machine), która mogłaby posiadać wszystkie możliwości obliczeniowe dzisiejszych komputerów. Jego celem było opisanie granicy między tym, co maszyna obliczająca może zrobić, a tym, czego nie może.
- ▶ W latach czterdziestych i pięćdziesiątych XX wieku (1940-1950) wielu badaczy badało proste rodzaje maszyn, zwane automatami.
- ▶ Pod koniec lat pięćdziesiątych (1950) Amerykański lingwista [Noam Chomsky](#) rozpoczął badania nad gramatykami formalnymi, które mają ścisły związek z automatami abstrakcyjnymi i służą jako podstawa dla niektórych ważnych programów, w tym części kompilatora.

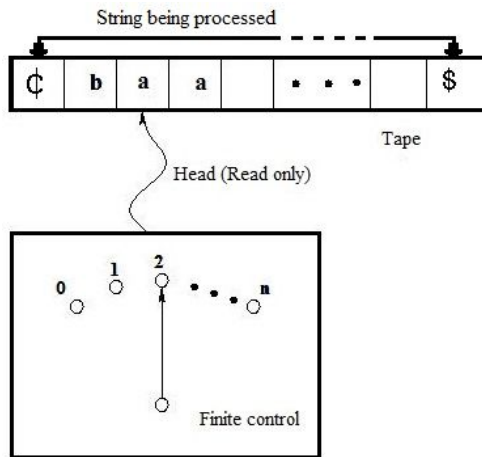
Automaty skończone modelowaniu sprzętu i oprogramowania

- ▶ Oprogramowanie do projektowania i sprawdzania zachowania **układu cyfrowego** (digital circuit)
- ▶ **Analizator leksykalny** (Lexical analyzer) typowego kompilatora, który dzieli tekst wejściowy na jednostki logiczne, takie jak identyfikator, słowa kluczowe i znaki interpunkcyjne
- ▶ Oprogramowanie do skanowania dużych zbiorów tekstu, takich jak zbiory stron internetowych, w celu znalezienia wystąpień słów, wyrażeń lub innych wzorców
- ▶ Oprogramowanie do weryfikacji systemów (wszelkiego typu), które mają skończoną liczbę odrębnych stanów, takie jak protokoły komunikacyjne lub protokoły bezpiecznej wymiany informacji

Intuicyjny opis automatu

- (i) Może odczytywać dane wejściowe (**input**) za pomocą mechanizmu odczytu. Dane wejściowe mogą być ciągiem znaków (**string of symbols**) nad danym alfabetem (**alphabet**), zapisanym w pliku wejściowym, który automat może odczytać, ale nie może go zmienić. Plik wejściowy jest podzielony na komórki, z których każda może pomieścić jeden symbol z alfabetu. Mechanizm odczytu może czytać plik od lewej do prawej, po jednym symbolu na raz. Może również wykryć koniec ciągu wejściowego.
- (ii) Może generować dane wyjściowe (**output**) w jakiejś formie. Może posiadać tymczasowe urządzenie pamięci (**temporary storage device**) składające się z nieograniczonej liczby komórek, z których każda może przechowywać pojedynczy symbol z alfabetu, który może różnić się od alfabetu wejściowego. Automat może odczytywać i zmieniać zawartość komórek pamięci.
- (iii) Automat posiada jednostkę sterującą (**control unit**), która może znajdować się w dowolnym ze skończonej liczby stanów wewnętrznych (**internal states**) i która może zmieniać bieżący stan (**current state**) i ruszać do następnego stanu (**next state**) w określony sposób.
- (iv) Automat działa w dyskretnym przedziale czasowym. W dowolnym momencie sterowanie znajduje się w pewnym stanie wewnętrznym i skanuje określony symbol w pliku wejściowym. Wewnętrzny stan sterowania w następnym punkcie czasowym jest określony przez funkcję przejścia (lub funkcję następnego stanu) (**transition function or next state function**). Funkcja przejścia przyjmuje jako argumenty bieżący stan, symbol wejściowy i informacje aktualnie znajdujące się w pamięci tymczasowej. Podczas przejścia z jednego stanu do drugiego może zostać wygenerowany wyjście lub informacje w pamięci tymczasowej mogą ulec zmianie.
- (v) Określony stan skierowanie, pliku wejściowego i pamięci tymczasowej nazywany jest łącznie konfiguracją (**configuration**). Przejście automatu z jednej konfiguracji do drugiej nazywa się ruchem (**move**).

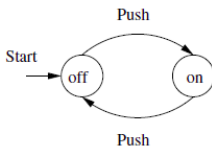
Ogólny obraz automatu skończonego



Finite Automaton

Przykłady automatów

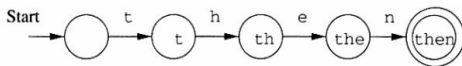
Przykład 1 Najprostszym nietrywialnym automatem jest włącznik-wyłącznik (**on-off switch**)



- ▶ Ma dwa stany: $\{off, on\}$
- ▶ Alfabet (wejściowy): $\{Push\}$
(wejściem może być dowolny wpływ zewnętrzny, taki jak działania i/lub ciągi znaków)
- ▶ Funkcja przejścia jest zdefiniowana jako: $(off, push) \mapsto on$ $(on, push) \mapsto off$
- ▶ Stan początkowy: **off**
- ▶ Zwykle jeden (lub kilka stanów) jest wyznaczany jako stan końcowy, oznaczony poprzez umieszczenie stanu w podwójnym okręgu. Osiągnięcie stanu końcowego oznacza, że automat ma żądany wynik. Przykładowo, jako stan końcowy można tu określić **on** bo po osiągnięciu tego stanu urządzenie może znajdować się w trybie pracy.
(należy pamiętać, że w tym przypadku wyjściem nie jest ciąg znaków, a raczej zdarzenie)

Przykład 2: Analizator leksykalny

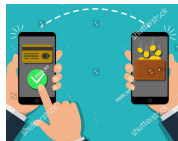
(Lexical analyzer) Analizator leksykalny, którego zadaniem jest rozpoznanie słowa kluczowego 'then'.



- ▶ Ma pięć stanów: $\{q_0, q_1, q_2, q_3, q_4\}$
(tutaj stany są wskazywane przez część słów, które sterowanie już przeczytało)
- ▶ Alfabet (wejściowy): $\{t, h, e, n\}$
- ▶ Funkcja przejścia jest zdefiniowana jako:
 $(q_0, t) \mapsto q_1$ $(q_1, h) \mapsto q_2$ $(q_2, e) \mapsto q_3$ $(q_3, n) \mapsto q_4$
- ▶ Stan początkowy: q_0
- ▶ Stan końcowy: q_4

Przykład 3: Protokół dla pieniądza elektronicznego

Pieniądże elektroniczne to pliki, za pomocą których klient może zapłacić za zakupy w Internecie, a które sprzedawca może otrzymać z gwarancją, że pieniądze są prawdziwe.



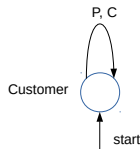
W procesie uczestniczy trzech uczestników: klient, sklep i bank. Interakcje między nimi ograniczają się do pięciu następujących zdarzeń.

- ☐ Klient może wybrać płatność (**Pay**). Oznacza to, że Klient przesyła do sklepu plik z pieniędzmi elektronicznymi (**electronic-money file**).
- ☐ Sklep może żądać (**Redeem**) pieniądze. Oznacza to, że plik pieniądza elektronicznego (**electronic-money file**) jest wysyłany do banku z prośbą o podanie jego wartości do sklepu.
- ☐ Bank może przelać (**Transfer**) pieniądze tworząc odpowiedni zaszyfrowany plik pieniężny i przesyłając go do sklepu.
- ☐ Sklep może wysłać (**Ship**) rzecz do Klienta.
- ☐ Klient może podjąć decyzję o rezygnacji (**Cancel**). Plik pieniądza elektronicznego (**electronic-money file**) wysyłany jest do banku z komunikatem o konieczności doliczenia wartości pieniądza do rachunku bankowego klienta.

Teraz możemy zaprojektować zachowanie każdego uczestnika za pomocą odpowiedniego automatu.

Automat reprezentujący klienta

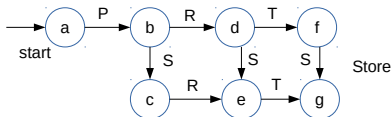
- ▶ Zachowanie Klienta:
 - Klient ma dwie możliwości: Zapłaci i/lub Anuluj (Pay, Cancel).
 - Klient może spróbować skopiować ten sam plik pieniężny, użyć go do kilkukrotnej zapłaty lub zarówno zapłacić, jak i anulować, aby móc otrzymać rzeczy za darmo.



Choć automat dla klienta rozpoczyna proces pieniądza elektronicznego poprzez wybór akcji Zapłaci lub Anuluj, nie zmienia własnego stanu. Raczej powoduje zmiany w dwóch pozostałych automatach. Zatem dla automatu klienta wszelkie działania są nieistotne, gdyż pozostaje on w tym samym stanie. Mamy więc pętle dla wszystkich działań.

Automat reprezentujący sklep

- ▶ **Zachowanie sklepu:** Sklep może mieć oddzielne procesy wypłaty (**Redeem**) pieniędzy z banku i wysyłki (**Ship**) przedmiotów. Możliwe jest zatem, że wysyłka może nastąpić przed, w trakcie lub po wypłacie pieniędzy.

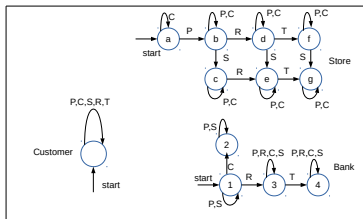


- o Ze początkowego **stanu a**, gdy klient złoży zamówienie wybierając akcję zapłacić (**Pay**), automat przechodzi w **stan b**.
- o Stamtąd, jeśli najpierw wyśle wiadomość o realizacji (**Redeem**), przejdzie do **stanu d**, a jeśli wyśle rzeczy (**Ship**) jako pierwszy, przejdzie do **stanu c**.
- o Jeśli zostanie wybrana druga opcja, nadal musi wysłać wiadomość o wykupieniu do banku (**Redeem**) i otrzymać przelew (**Transfer**) o równowartości pieniędzy i tym samym dotrzeć do **stanu g**.
- o W przypadku pierwszej opcji, po osiągnięciu **stanu d**, sklep może albo dokonać pierwszej wysyłki (**Ship**) (osiągając **stan e**), albo jako pierwszy otrzymać przelew z banku (**Transfer**) (osiągając **stan f**)
- o W pierwszym przypadku w **stanie e** przelew (**Transfer**) nadal oczekuje, a w kolejnym dochodzi do **stanu g**. W tym drugim przypadku ze **stanu f** wysyłka (**Ship**) musi zostać zrobiona i więc dociera do **stanu g**, gdzie wszystkie transakcje są zakończone.

Dodanie nieistotnych działań w formie pętli

Dodanie nieistotnych działań, aby automat nie zatrzymał się przed osiągnięciem celu

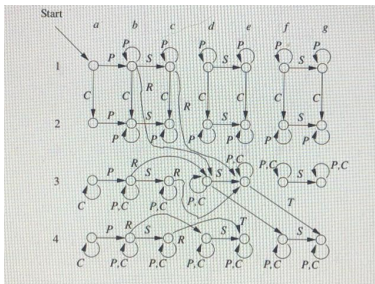
- ▶ Przykładowo, jeśli klient zdecyduje się wykonać akcję zapłacić drugi raz, gdy sklep jest w **stanie e**, automat sklepu zgśnie, gdyż nie ma łuku oznaczonego Zapłacić (**Pay**) ze **stanu e**. Musimy więc dodać pętle dla niektórych stanów z etykietami dla wszystkich działań, które należy zignorować, gdy automat znajduje się w tym stanie.



- Jeśli chodzi o sklep, jedyną nieistotną akcją jest anulowanie (**Cancel**). Zatem we wszystkich stanach dodana jest pętla z C. Dla banku zarówno płać (**Pay**), jak i wysyłka (**Ship**) są nieistotne. Zatem w każdym węźle dodawane są pętle z etykietami P i S.
- Ponadto, ponieważ nie chcemy, aby klient psuł funkcjonowanie sklepu wykonaniem akcji Zapłacić (**Pay**) ponownie, musimy dodać pętle z etykietą P do wszystkich oprócz **stanu a**. Podobnie, aby uniemożliwić klientowi wyłączenie automatu bankowego poprzez próbę anulowania (**Cancel**) już wydanych pieniędzy, do **stanu 3** i **stanu 4** dodawane są pętle oznaczone literę C. Ponadto bank ma pętle z R w **stanie 3** i **stanie 4**, ponieważ sklep nie powinien próbować wysłać wiadomości o wykupieniu (**Redeem**) tych samych pieniędzy dwa razy.

Automat łączący wszystkie trzy automaty i ich interakcje

- ▶ Automat dla klienta ma tylko jeden stan i dowolna sekwencja działań pozwala mu pozostać w tym samym stanie. Jednak zarówno automaty dla banku, jak i sklepu oddziałują ze sobą w sposób złożony i z poszczególnych automatów dla nich nie jest oczywiste, w jakich kombinacjach stanów mogą one występować.
- ▶ Aby więc połączyć te dwa automaty ze wszystkimi możliwymi interakcjami, musimy stworzyć iloczyn dwóch automatów.
- **Stany nowego automatu** Są parą stanów - jeden z automatu bankowego i drugi z automatu sklepowego. Na przykład możliwym stanem w nowym automacie może być (i, x) , w którym bank jest w stanie i , a sklep w stanie x .
- **Funkcja przejścia dla nowego automatu:** Jeśli w automacie bankowym jest łuk od stanu i do stanu j oznaczonego akcją Z , a w automacie sklepowym jest łuk od stanu x do stan y oznaczony akcją Z , wówczas w nowym automacie istnieje łuk od stanu (i, x) do stanu (j, y) oznaczonego przez Z .



Automat skończony

Istnieją dwa rodzaje automatów skończonych - deterministyczne i niedeterministyczne.

- ▶ **(Deterministic Finite Automaton)**: Deterministyczny automat skończony (DFA) to pięciokrotka

$$M = (Q, \Sigma, \delta, q_0, F) \text{ gdzie}$$

- (i) Q jest skończonym zbiorem stanów,
- (ii) Σ to skończony zbiór symboli wejściowych, nazywany alfabetem,
- (iii) $\delta : Q \times \Sigma \mapsto Q$ to funkcja przejścia, która określa następny stan, biorąc pod uwagę bieżący stan i symbol wejściowy,
- (iv) $q_0 \in Q$ to stan początkowy, i
- (v) $F \subseteq Q$ to zbiór zawierające wszystkie stany końcowe.

Przykład 4: $M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$ gdzie δ jest zdefiniowana jako:
 $\delta(q_0, 0) = q_1$, $\delta(q_0, 1) = q_0$, $\delta(q_1, 0) = q_1$, $\delta(q_1, 1) = q_2$, $\delta(q_2, 0) = q_2 = \delta(q_2, 1)$.

Automat skończony

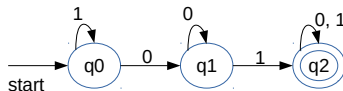
Istnieją dwa rodzaje automatów skończonych - deterministyczne i niedeterministyczne.

- **(Deterministic Finite Automaton):** Deterministyczny automat skończony (DFA) to pięciokrotka

$$M = (Q, \Sigma, \delta, q_0, F) \text{ gdzie}$$

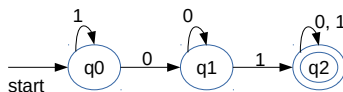
- (i) Q jest skończonym zbiorem stanów,
- (ii) Σ to skończony zbiór symboli wejściowych, nazywany alfabetem,
- (iii) $\delta : Q \times \Sigma \mapsto Q$ to funkcja przejścia, która określa następny stan, biorąc pod uwagę bieżący stan i symbol wejściowy,
- (iv) $q_0 \in Q$ to stan początkowy, i
- (v) $F \subseteq Q$ to zbiór zawierające wszystkie stany końcowe.

Przykład 4: $M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$ gdzie δ jest zdefiniowana jako:
 $\delta(q_0, 0) = q_1$, $\delta(q_0, 1) = q_0$, $\delta(q_1, 0) = q_1$, $\delta(q_1, 1) = q_2$, $\delta(q_2, 0) = q_2 = \delta(q_2, 1)$.



Jak ciąg znaków jest przetwarzany w DFA

- ▶ Wcześniej widzieliśmy, że poprzez funkcję przejścia δ DFA przechodzi do nowego stanu po przetworzeniu pojedynczego symbolu z alfabetu Σ .
- ▶ Zatem zaczynając od stanu początkowego q_0 , biorąc pod uwagę ciąg symboli $w = a_1 a_2 \dots a_n$, jeśli DFA jeden po drugim odczytuje symbole ciągu znaków i po odczytaniu ostatniego symbolu ciągu osiągnie końcowy stan DFA, wówczas cały ciąg w może zostać rozpoznany przez DFA.



- ▶ Np. Zaczyna się od węzła q_0 . Jeśli czyta 0 następnym stanem będzie q_1 i potem po odczytaniu 1 dociera do stanu q_2 , który jest stanem końcowym. Czyli, **ciąg 01 musi być rozpoznany przez ten DFA.**

Rozszerzenie funkcji przejścia dla ciągów znaków

- **Definicja** $\hat{\delta} : Q \times \Sigma^* \mapsto Q$ jest zdefiniowany rekurencyjnie w następujący sposób.

(i) $\hat{\delta}(q, \Lambda) = q$.

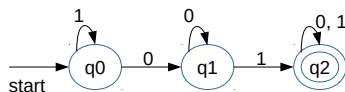
(ii) Dla $a \in \Sigma$, $\hat{\delta}(q, a) = \delta(q, a)$.

(iii) Dla $wa \in \Sigma^*$ gdzie $a \in \Sigma$, $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$.

- To znaczy, jeśli mamy ciąg znaków $a_1 a_2 \dots a_n$ i zaczynamy od stanu początkowego q_0 , to najpierw sprawdzamy $\delta(q_0, a_1) = q_1$ (powiedzmy), $\delta(q_1, a_2) = q_2$ (powiedzmy) ... aż $\delta(q_{n-1}, a_n) = q_n$ (powiedzmy). Jeśli $q_n \in F$ wtedy i tylko wtedy $a_1 a_2 \dots a_n$ jest akceptowany (rozpoznany) przez automaton.
- **Definicja:** Język DFA $M = (Q, \Sigma, \delta, q_0, F)$ to zbiór ciągów akceptowanych przez M. Tzn., $L(M) = \{w \in \Sigma^* : \hat{\delta}(q_0, w) \in F\}$.

Tabela przejściowa

Dowolną funkcję przejścia można przedstawić poprzez narysowanie diagramu przejścia lub tabeli przejść.



	0	1
→ q ₀	q ₁	q ₀
q ₁	q ₁	q ₂
q ₂	q ₂	q ₂

Bibliografia

- (1) Introduction to Automata Theory, Language and Computation, John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman, Prentice Hall, 2006.
- (2) An Introduction to Formal Languages and Automata, Peter Linz, Jones & Bertlett Learning, 2012.
- (3) Języki Formalne i Automaty, Miłosz Michalski, UMK Toruń, 2015.