1.1 Pojęcia podstawowe

1. Alfabet

Alfabet to dowolny niepusty zbiór skończony V (niekiedy dodajemy zbiór symboli lub zbiór liter). Elementy alfabetu nazywamy ***literami*** lub ***symbolami*.** Alfabet tak jak zbiór oznaczamy najczęściej dużymi literami alfabetu łacińskiego np. V, A itd.

Przykład: ***Alfabet binarny*** (inaczej alfabet dwójkowy) to alfabet złożony z 2 elementów: np. $V = \{0,1\}$, $V = \{L, H\}$.

Przykład: Alfabetami są zbiory skończone: $V = \{a, b, c\}$, $V = \{0,1,2,...,9\}$, $V = \{a, b, c, ..., x, y, z\}$, $V = \{a, b, c, ..., x, y, z, A, B, C, ..., X, Y, Z, ,0,1,2,...,9\}$

Przykład: Przy formalnym opisie języka programowania pierwszym krokiem jest zawsze sprecyzowanie skończonego zbioru symboli czyli alfabetu, którym będziemy się posługiwać. Podanie, sprecyzowanie alfabetu (a ściślej alfabetów) powinno stanowić również zawsze pierwszy krok w definicji ***automatu skończonego***, czy też w definicji ***kodu***. Pojęcia *kodu* i *automatu skończonego* będą omawiane w dalszej części wykładu.

Warto zwrócić uwagę, na fakt że to co nazywamy symbolem lub literą jest kwestią umowy. W systemach cyfrowych symbole często utożsamiamy z pewnymi łatwo rozróżnialnymi sytuacjami fizycznymi, stanami układu fizycznego np. stanami układu elektronicznego zwanego przerzutnikiem bistabilnym (układ przerzutnika bistabilnego ma 2 łatwo rozróżnialne stabilne stany).

Dla nas symbolem będzie np. wartość napięcia w określonym węźle układu cyfrowego (H - wyższe napięcie np. 5 V lub L - niższe napięcie np.0 V) a alfabetem zbiór $V = \{L, H\}$. Typowe wartości napięcia H i L w systemach cyfrowych to H = 5 V i L = 0 V (np. tak jest w tzw. układach TTL i CMOS o małej skali scalenia) ale wewnątrz układu scalonego o dużej skali scalenia stosuje się z reguły napięcia niższe np. H = 3,3 V, L = 0 V. Oczywiście rozsądek techniczny każe uznawać za napięcie H wszystkie wartości napięć z pewnego niewielkiego przedziału I_H zawierającego nominalną wartość napięcia H (czyli napięcia zbliżone do H) i podobnie za napięcie L przyjmujemy wszystkie napięcia z niewielkiego przedziału I_L zawierającego nominalną wartość napięcia L (czyli napięcia zbliżone do L). Musimy przy tym spełnić oczywiście warunek by $I_H \cap I_L = 0$.

Symbolem może być również przemagnesowanie określonego miejsca nośnika magnetycznego lub brak takiego przemagnesowania. Uzyskujemy w tej sytuacji 2 symbole, które łatwo możemy zapisywać na powierzchni magnetycznej za pomocą tzw. głowicy magnetycznej.

Symbolem może być też drobne wgłębienie na powierzchni CD-ROM'u (tzw. pit) rozpraszający światło promienia lasera głowicy odczytującej lub brak pit'u na powierzchni CD-ROM'u powodujący odbicie promienia laserowego. Odbity promień lasera trafia do detektora (diody detekcyjnej) wywołując impuls elektryczny odpowiadający 1.

Symbolem może być również stan przełącznika mechanicznego o 2 (czy ogólnie *n* pozycjach stabilnych).

Cechą charakterystyczną symboli czyli liter wchodzących w skład alfabetu jest łatwość ich zapisywania (czyli zapamiętywania) i odczytywania bez pomyłek. Stąd zresztą popularność w praktyce alfabetu binarnego. Układ fizyczny mający tylko dwa dobrze wyodrębnione stany utożsamiane odpowiednio z 0 lub 1 okazuje się w praktyce pewniejszy, bardziej niezawodny.

Reasumując, powyżej opisaliśmy 4 najczęściej stosowane sposoby reprezentacji w systemach cyfrowych symboli 0 i 1 :

- sposób elektryczny wykorzystujący 2 poziomy napięcia *L* i *H* w określonym węźle układu elektronicznego (jest to tzw. kodowanie potencjałowe „0" i „1")
- sposób magnetyczny wykorzystujący przemagnesowanie nośnika magnetycznego
- sposób optyczny wykorzystujący zmianę własności optycznych powierzchni odbijającej światło laserowe
- sposób mechaniczny wykorzystujący stan przełącznika mechanicznego.

Istota rzeczy: Właściwie pojęcie alfabetu jest intuicyjnie jasne a rola dokładnie taka sama jaką pełni alfabet w języku naturalnym. Alfabet służy do zapisywania słów. Warto zwrócić uwagę, na fakt, że to co nazywamy symbolem w formalnej definicji alfabetu jest oczywiście kwestią umowy.

2. Słowo

Słowo nad alfabetem V (ang. word) to dowolny ciąg skończony o wartościach w zbiorze V. Czasami słowo nad ustalonym alfabetem nazywamy ***tekstem*** lub ***napisem*** (a w językach programowania również ***łańcuchem*** lub ***stringiem***). Słowo $a_1, a_2, ..., a_n$ zapisujemy z reguły bez przecinków tzn. jako: $a_1 a_2 ... a_n$ .Upraszcza to notację nie prowadząc na ogół do nieporozumień.

Przykład: Jeśli $V_1 = \{a, b, c, d\}$ jest ustalonym alfabetem to *aabbba*, *dcba* są słowami nad alfabetem V_1. Słowo *abbde* nie jest słowem nad alfabetem V_1 ale jest słowem nad alfabetem $V_2 = \{a, b, c, d, e\}$. Słowo *11100011* jest słowem nad alfabetem $V_2 = \{0,1\}$.

Istota rzeczy: Formalne pojęcie "słowo nad alfabetem" z powyższej definicji odpowiada słowu języka naturalnego, jednak z tak rozumianym bardzo formalnie słowem nie musimy wiązać żadnego znaczenia.

Ilość wyrazów ciągu $a_1 a_2 ... a_n$ nazywamy *długością słowa* $a_1 a_2 ... a_n$ np. słowo *abccd* nad alfabetem $V = \{a, b, c, d\}$ ma długość 5 a słowo *ala* długość 3.
Jeśli $V = \{0,1\}$ to słowa nad tym alfabetem nazywamy *słowami binarnymi*. Słowo binarne o długości 8 bitów np. 10101010 nazywamy *bajtem* (ang. byte).

Zamiast słowa ***bajt*** (lub ***byte***) używamy skrótu B. Właśnie w bajtach podajemy najczęściej pojemność różnego typu pamięci. Każdy adept inżynierii komputerowej wie, że typowa pojemność tzw. pamięci operacyjnej komputera klasy PC jest rzędu jednego gigabyte'a (1 GB czyli jeden gigabyte to 10^9 B), a typowa pojemność napędu twardego dysku jest rzędu 100 GB . Pojemności tzw. macierzy dyskowych czy streamerów mierzymy w terabajtach (1 TB czyli jeden terabyte to 10^{12} B).

Reasumując: typowe jednostki, w których mierzymy pojemność pamięci to

- B (bajt, 1 B)
- kB (kilobajt 10^3 B)
- MB (megabajt 10^6 B)
- GB (gigabajt 10^9 B)
- TB (terabajt 10^{12} B)

Pewnego komentarza wymaga jeszcze pojęcie ***bitu***. Bit to symbol (litera) 0 lub 1. Mówimy, też, że bit to cyfra dwójkowa. Często jednak pojęcie bitu wiążemy z ustalonym słowem binarnym np. $a_1a_2...a_n$. Możemy wówczas mówić o *k*-tym bicie słowa binarnego $a_1a_2...a_n$, gdzie $k \le n$, którym z definicji jest *k*-ty wyraz ciągu $a_1a_2...a_n$ czyli a_k. W innym jeszcze znaczeniu słowo "bit" oznacza jednostkę ilości informacji definiowaną w teorii informacji.

Uwaga: W terminologii stosowanej przez firmę Intel słowo (ang. word) oznacza również słowo binarne 16 bitowe (2 bajty) a podwójne słowo (ang. double word) oznacza słowo binarne 32 bitowe (4 bajty).

3. Język

Zbiór wszystkich słów nad ustalonym alfabetem V oznaczamy symbolem V^* a dowolny niepusty podzbiór L tego zbioru nazywamy *językiem.* Do zbioru V^* zaliczamy również słowo puste (oznaczane na ogół symbolem ε). Słowo puste ma długość 0. Zbiór V^* jest oczywiście nieskończony ale przeliczalny mamy bowiem

$$V^* = \{\varepsilon\} \cup V \cup V^2 \cup ... \cup V^n \cup ...$$

W zbiorze V^* definiujemy działanie dwuargumentowe $\circ : V^* \times V^* \to V^*$ tzw. konkatenację (ang. concatenation). Jeśli $\alpha, \beta \in V^*$ i $\alpha = a_1a_2....a_n$ $\beta = b_1b_2....b_n$ to z definicji mamy

$$\alpha \circ \beta = a_1a_2....a_n \circ b_1b_2...b_n \overset{df}{=} a_1a_2...a_nb_1b_2...b_n.$$

Jak widać konkatenacja jest zestawianiem słów np. *ala* konkatenowane z *makota* daje *alamakota* podobnie *kot* z *let* daje *kotlet.*

Jak łatwo sprawdzić działanie konkatenacji jest łączne (tzn. dla każdego $\alpha, \beta, \gamma \in V^*$ mamy $(\alpha \circ \beta) \circ \gamma = \alpha \circ (\beta \circ \gamma)$) . Mamy ponadto dla każdego $\alpha \in V^*$; $\alpha \circ \varepsilon = \varepsilon \circ \alpha = \alpha$ zatem słowo puste ε jest jedynką działania $\circ : V^* \times V^* \to V^*$. Inaczej mówiąc konkatenacja ze słowem pustym dowolnego słowa $\alpha \in V^*$ nie zmienia tego słowa. Zbiór V^* z działaniem konkatenacji $\circ : V^* \times V^* \to V^*$ jest więc monoidem (czyli półgrupą z jednością).

Widać natychmiast, że konkatenacja na ogół nie jest działaniem przemiennym w V^*, ale zawsze jest działaniem łącznym tzn. na ogół $x \cdot y \neq x \cdot y$ dla $x, y \in V^*$, ale zawsze tzn. dla każdego $x, y, z \in V^*$ mamy:

$$(x \cdot y) \cdot z = x \cdot (y \cdot z)$$

Język to dowolny podzbiór zbioru V^*. Podzbiór ten można definiować na wiele sposobów np. za pomocą wyrażeń regularnych, notacją Backusa-Naura, gramatyką, automatem skończonym itd.. Wszystkie te sposoby poznamy w dalszym ciągu. Istnieje również wiele typów języków m.in. języki regularne, języki bezkontekstowe itd.

Przykład: Jeśli $V=\{0,1\}$ to zbiór $\{0{,}001\ ,100001\}$ jest językiem. ■

Przykład: Sam alfabet V jest również językiem ponieważ jest to zbiór wszystkich jednoliterowych słów nad alfabetem V. ■

Uwaga: Zauważmy, że V^* jest zbiorem przeliczalnym, zatem również każdy język jest co najwyżej zbiorem przeliczalnym.■

Istota rzeczy: Jeśli weźmiemy kilka słów a może nieskończoną liczbę słów to mamy język. Możemy 2 słowa zestawiać razem tworząc nowe słowo. Nazywa się to konkatenacją.

Złożeniem dwóch języków K i L, *gdzie* $K,L\subset V^*$ nazywamy język

$KL\overset{df}{=}\{x,y\in V^*;x\in K \text{ oraz } y\in L\}$

Potęga języka. *Potęgę* L^* *języka* definiujemy indukcyjnie następująco $L^0\overset{df}{=}\{\varepsilon\}$, $L^1\overset{df}{=}L$, $L^2\overset{df}{=}LL$,..., $L^{n+1}\overset{df}{=}LL^n$,...

Potęga liter. *Potęgę liter* definiujemy indukcyjnie następująco. Niech *a* będzie dowolną ustaloną literą $a\in V^*$ wówczas $a^0\overset{df}{=}\{\varepsilon\}$, $a^1\overset{df}{=}a$, $a^2\overset{df}{=}aa$, ..., $a^{n+1}\overset{df}{=}aa^n$,...

Potęga słów. *Potęgę słów* definiujemy indukcyjnie następująco. Niech *w* będzie dowolnym ustalonym słowem, $w\in V^*$, wówczas $w^0\overset{df}{=}\{\varepsilon\}$, $w^1\overset{df}{=}w$, $w^2\overset{df}{=}ww$, ..., $w^{n+1}\overset{df}{=}ww^n$,...

Domknięcie języka (gwiazdeczka Kleen'a) L^*. Niech V będzie ustalonym alfabetem a L językiem $L\subset V^*$. *Domknięciem* języka L nazywamy zbiór $L^*\subset V^*$, gdzie.

$$L^*\overset{df}{=}L^0\cup L^1\cup L^2\cup .,..\cup L^n\cup ...$$

4. Metryka Hamminga

Niech V będzie dowolnym ustalonym alfabetem. Wprowadźmy w tym alfabecie metrykę dyskretną $\rho_d:V\times V\to R^+$. Z twierdzenia 1 z Dodatku (p. Metryka Hamminga) wynika, że funkcja $\rho_H:V^n\times V^n\to R^+$ zdefiniowana wzorem: dla każdego $x,y\in V^n$

$$\rho_H(x,y)=\sum_{i=1}^{n}\rho_d(x_i,y_i)$$

jest metryką nazywamy ją *metryką lub odległością Hamminga* w przestrzeni V^n (V^n jest to przestrzeń wszystkich słów o długości *n* nad alfabetem *V*). Najczęściej rozważamy metrykę Hamminga dla $V^n=\{0,1\}^n$. Oczywiście $\rho_H:V^n\times V^n\to N\cup\{0\}$ czyli ρ_H jest funkcją o wartościach w zbiorze liczb całkowitych nieujemnych.

Uwaga: Wartość $\rho_H(x,y)=\sum_{i=1}^{n}\rho_d(x_i,y_i)$ dla 2 słów $x,y\in V^n$ (czyli 2 słów o długości n nad alfabetem V) jest równa liczbie pozycji na, których słowa x i y się różnią. Jeśli np. przesyłamy słowo x przez kanał telekomunikacyjny i na wyjściu tego kanału otrzymujemy słowo y to $\rho_H(x,y)$ podaje liczbę błędów transmisji słowa x.

Przykład: Jeśli $V=\{0,1\}$ oraz x=01110000 i y=10001111 to odległość Hamminga tych 2 słów jest równa 8.

Waga słowa binarnego. Wagą słowa binarnego z przestrzeni $\{0,1\}^n$ (lub ogólniej z $\{0,1\}^*$) nazywamy liczbą jedynek w tym słowie. W ten sposób definiujemy na $\{0,1\}^*$ funkcję $w:\{0,1\}^*\to N\cup\{0\}$. Oczywiście dla $a\in\{0,1\}^n$ mamy $w(a)=\rho_H(0,a)$ gdzie ρ_H jest metryką Hamminga. Można wyrazić metrykę Hamminga ρ_H w $\{0,1\}^n$ za pomocą funkcji wagi w.

Niech $x,y\in\{0,1\}^n, x=(x_1,x_2,...,x_n), y=(y_1,y_2,...,y_n)$, wówczas mamy

$$\rho(x,y)\overset{df}{=}\sum_{i=1}^{n}\rho_d(x_i,y_i)=\sum_{i=1}^{n}w(x_i-_2\ y_i)=w(x-y)=w(x+y)$$

gdzie $x_i-_2\ y_i$ oznacza różnicę modulo 2, $x-y$ oznacza różnicę modulo 2 po współrzędnych a $x+y$ sumę modulo 2 po współrzędnych.

Przykład: Waga słowa 11111111 jest równa 8, tzn. w(11111111)=8, w(00)=0, w(001)=1.

5. Kody

Niech V_1 oznacza dowolny ustalony zbiór niepusty a V_2 dowolny alfabet. Zbiór V_1 będziemy nazywać zbiorem obiektów kodowanych.. Istnieje kilka definicji pojęcia kodu. Najogólniejsza definicja jest taka. ***Kod*** to relacja dwuargumentowa $k\subseteq V_1\times V_2^*$ spełniająca warunek: dla każdego $x_1\in V_1$ czyli dla każdego elementu ze zbioru obiektów kodowanych istnieje takie $x_2\in V_2^*$, że $(x_1,x_2)\in k$. Elementy przeciwdziedziny relacji k czyli elementy zbioru $\{x_2\in V_2^*;(x_1,x_2)\in k\}$ nazywamy w tej sytuacji ***słowami kodowymi***. Jeśli $V_2=\{0,1\}$ to kod nazywamy ***kodem binarnym*** lub ***kodem dwójkowym***. W szczególności dowolne odwzorowanie $f:V_1\to V_2^*$ jest oczywiście kodem.

Druga definicja kodu nieco bardziej "wymagająca" jest taka: kod to dowolne odwzorowanie $f:V_1\to V_2^*$ żądamy przy tym z reguły by funkcja f była różnowartościowa i nie przyjmowała wartości ε i tak najczęściej rozumiane jest pojęcie kodu. W dalszym ciągu pod pojęciem kodu będziemy rozumieć tę najbardziej restryktywną definicję. Sam proces obliczania czy ustalania wartości funkcji f dla elementów zbioru V_1 nazywamy ***kodowaniem***. Wartość $f(x_1)\in V_2^*$ jest słowem kodowym, kodującym element $x_1\in V_1$.
Jeśli istnieje takie $n\in N$, że dla każdego $x\in V_1$ słowo kodowe $f(x)\in V_2^*$ ma stałą długość n to kod nazywamy ***kodem o stałej długości*** (ang fixed length code) lub dokładniej ***kodem***

o stałej długości n słowa kodowego. Oczywiście mamy wówczas $f : V_1 \to V_2^n$. Najczęściej w systemach cyfrowych mamy do czynienia z takimi właśnie kodami. Przykładem takiego kodu jest znany 8 bitowy kod alfanumeryczny ASCII.

Istota rzeczy: Obiekty chcemy "etykietować", opisywać za pomocą słów nad ustalonym alfabetem i czynimy to za pomocą kodu. Jest to wygodne i naturalne zwłaszcza jeśli chcemy reprezentować obiekty różnego typu w systemie komputerowym. Z każdym obiektem wiążemy dokładnie jedno słowo.

***Kod redundancyjny* (lub *nadmiarowy*).** Niech $f : V_1 \to V_2^*$ będzie kodem. Oznaczmy przez A_n zbiór obiektów kodowanych słowami o długości $n \in N$ tzn. niech $A_n = \{x \in V_1; f(x) \in V_2^n\}$. Jeśli istnieje takie $n \in N$, że $f(A_n) \neq V_2^n$ czyli zbiór $f(A_n)$ jest właściwym podzbiorem zbioru V_2^n to kod nazywamy redundancyjnym lub nadmiarowym. Najkrócej: kodem redundancyjnym nazywamy dowolny kod $f : V_1 \to V_2^*$ taki, że funkcja różnowartościowa f nie jest „na".

Istotą rzeczy jest tu istnienie niepustego słowa w zbiorze V_2^*, które nie koduje żadnego obiektu. W powyższej definicji nie interesuje nas czy zbiór obiektów kodowanych jest skończony czy nieskończony.

***Kod redundancyjny* (lub *nadmiarowy*) o stałej długości słowa kodowego.** Jeśli rozważamy kod o stałej długości $f : V_1 \to V_2^n$ (implikuje to skończoność zbioru obiektów kodowanych V_1) i odwzorowanie f nie jest "na" (czyli $f(V_1)$ jest właściwym podzbiorem zbioru V_2^n) to kod taki nazywamy kodem redundancyjnym lub nadmiarowym o stałej długości słowa kodowego. W takim kodzie nie wykorzystujemy wszystkich możliwych słów kodowych o długości n do kodowania obiektów kodowanych.

Kody redundancyjne umożliwiają

- 1.sprawdzenie czy dane słowo jest poprawnym słowem kodowym (wykrycie błędu)
- 2. ewentualną korekcję błędu jeśli taki błąd wystąpił na którejś pozycji słowa

Kody redundancyjne znajdują więc zastosowanie wszędzie tam gdzie zależy nam na wykrywaniu lub korekcji błędów występujących w słowie kodowym. Błędy takie pojawiają się czasami podczas transmisji lub przechowywania danych a wynikają z nieuniknionych w realnym środowisku fizycznym szumów i zakłóceń a również co może się zdarzyć z uszkodzenia układów elektronicznych przetwarzających informację. Każdy kod wykrywający błędy i każdy kod korygujący błędy jest kodem nadmiarowym.

Zauważmy jeszcze, że istota kodu polega na wiązaniu z obiektem kodowanym pewnego słowa nad ustalonym alfabetem V_2. Możemy sobie wyobrażać, że to słowo w pewien sposób opisuje wybrany obiekt, daje jego syntetyczną charakterystykę. Jednak ilość wszystkich słów nad dowolnym alfabetem V_2 jest przeliczalna. Zatem zakładając, że kod jest funkcją różnowartościową dostajemy, że zbiór obiektów kodowanych musi być przeliczalny. Nie ma więc np. kodu kodującego wszystkie liczby rzeczywiste (bo zbiór liczb rzeczywistych nie jest przeliczalny) choć jak pokażemy w dalszym ciągu łatwo zdefiniować kod dla zbioru wszystkich liczb wymiernych.

Uwaga: Często przez „kod", dokładniej "kod programu", rozumie się zapis programu w ustalonym języku programowania, najczęściej w assemblerze. Rzecz jasna słowo kod

w tym sensie ma niewiele wspólnego z powyżej definiowanym pojęciem kodu. Mówimy często "kod źródłowy", "kod wynikowy", "kod wykonywalny".

6. Podział kodów

Jeśli zbiorem obiektów kodowanych V_1 jest zbiór znaków to kod nazywamy ***kodem alfanumerycznym***. Jeśli zbiorem obiektów kodowanych V_1 są liczby to kod nazywamy kodem numerycznym. ***Kod numeryczny*** to inaczej kod liczbowy, kod arytmetyczny, zapis liczbowy, notacja liczbowa, lub system zapisu liczb.

Podział kodów:

- Kody α - numeryczne o stałej długości (np. ASCII, Unicode)
- Kody α - numeryczne o zmiennej długości (np. kod Huffmana)
- Kody liczbowe o stałej długości (np. kod NKB, zapis uzupełnień do 2)
- Kody liczbowe o zmiennej długości (np. zwykły zapis dziesiętny)
- Kody wykrywające i korygujące błędy (np. kod Hamminga, kod Reeda-Solomona)
- Szyfry (ściślej są to indeksowane kluczem rodziny kodów) (np. szyfr RSA, szyfr AES)
- Inne kody (np. kody paskowe, kody służące do zapisywania informacji na twardych dyskach, kody używane w transmisji danych itd.).