Wstęp do języka ASN.1

Grzegorz Danilewicz

O czym mówimy?

- Co oznacza ASN.1
- Jaka jest rola języka ASN.1 w specyfikacji protokołów
- Jak wygląda specyfikacja ASN.1
  - jak ją zapisać
  - jak ją czytać
  - przykład narzędzia (kompilatora ASN.1)

O czym mówimy?

- Jakie są reguły kodowania ze specyfikacji ASN.1 do reprezentacji bitowej

Wprowadzenie

Czym jest ASN.1?

- Abstrakcyjna notacja struktur danych, na wysokim poziomie abstrakcji
  - standaryzowana międzynarodowo
  - niezależna od dostawcy (sprzedawcy)
  - niezależna od platformy sprzętowej
  - niezależna od języka (naturalnego i programowania)

Czym jest ASN.1?

- Dodatkowo jest ona wspierana przez reguły kodowania
  - określają dokładne wzorce bitowe reprezentujące wartości tych struktur danych, które będą przesyłane przez sieć komputerową
  - określają format łatwy do czytania przez człowieka
- Jest wspierana przez narzędzia, które odwzorowują notację ASN.1 w definicję struktur danych w wybranym języku programowania

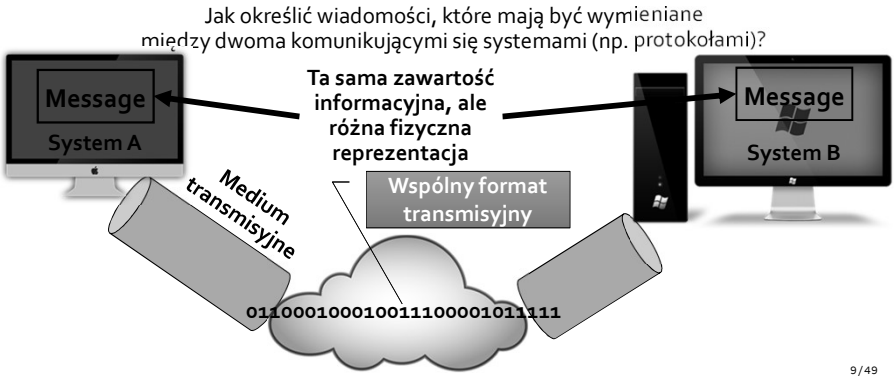
Abstract Syntax Notation 1 (ASN.1)

- Język ASN.1 został zdefiniowany przez ISO
  - jako bardzo ogólna metoda reprezentowania danych
  - w systemach komputerowych o różnej konstrukcji wewnętrznej
  - które mają wymieniać informacje między sobą
- ASN.1 jest używany przez ITU i ISO do definiowania nagłówków i formatów jednostek danych protokołów
- W OSI ASN.1 służy do definiowania aplikacji warstwy prezentacji

Standardy ASN.1

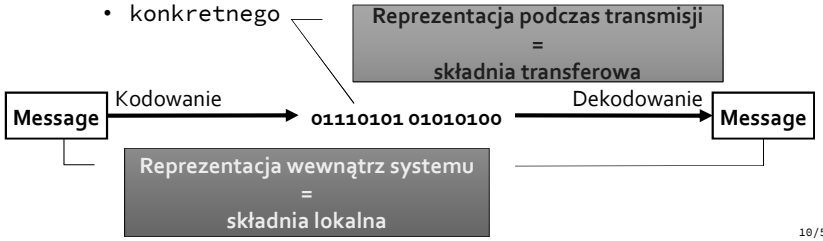
- X.680 ASN.1: Specification of basic notation
- X.681 ASN.1: Information object specification
- X.682 ASN.1: Constraint specification
- X.683 ASN.1: Parameterization of ASN.1 specifications
- X.690 ASN.1 encoding rules - Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)
- X.691 ASN.1 encoding rules - Specification of Packed Encoding Rules (PER)
- X.693 ASN.1 encoding rules - XML Encoding Rules (XER)

Uzasadnienie



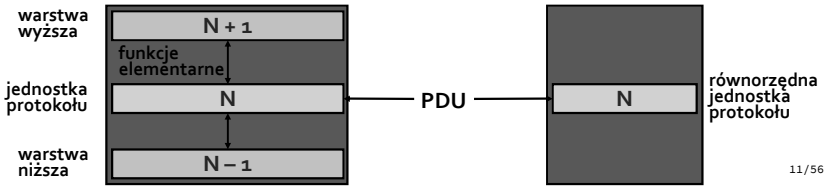
Uzasadnienie

- Wiadomość może być oglądana z dwóch poziomów:
  - abstrakcyjnego
    - logiczna zawartość informacyjna wiadomości → składnia abstrakcyjna
  - konkretnego



Funkcje elementarne i jednostki PDU

- Wiadomości w systemach komunikujących się dzielą się na dwie kategorie
  - jednostki PDU
    - wiadomości między dwoma równorzędnymi jednostkami protokołu
    - wymagają kodowania i dekodowania
  - funkcje elementarne (prymitywy)
    - wiadomości między dwoma jednostkami wewnątrz systemu



Składnie abstrakcyjne dla PDU

- Składnia abstrakcyjna jednostek PDU dotyczy
  - złożenia PDU z elementów informacyjnych
  - typu danych elementów informacyjnych
- Brak przykładania uwagi do bitów czy bajtów w tworzonej jednostce PDU
  - odpowiedzialność składni abstrakcyjnej: zawartość wiadomości
  - odpowiedzialność składni transferowej: reprezentacja wiadomości
- Składnia transferowa wiadomości wywodzi się ze składni abstrakcyjnej

Składnie abstrakcyjne dla PDU

- Definicja: w języku naturalnym lub z notacją formalną
- Przykład: półformalny zapis tabelaryczny

Nazwa PDU	Elementy informacyjne	Uwagi
MovePDU	Piece	...
	from Square	...
	to Square	...

Składnie abstrakcyjne dla PDU

- Definicja: w języku naturalnym lub z notacją formalną
- Przykład: półformalny zapis tabelaryczny

Element informacyjny	Dane	Uwagi
Piece	king, queen, ..., piece	...
Square	vertical row	values: a..h
	horizontal row	values: 1..8

Składnie abstrakcyjne dla funkcji elementarnych

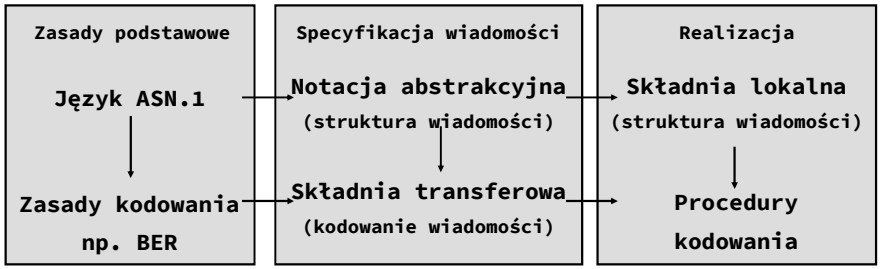
- Funkcje elementarne nie są przesyłane przez sieć
  - nie ma potrzeby stosowania składni transferowej
- Funkcje elementarne są wewnętrzne w systemie
  - nie ma potrzeby określania składni abstrakcyjnej
- Jednakże
  - istnieje silne sprzężenie między PDU a funkcjami elementarnymi
  - istnieje przepływ danych między jednostkami PDU i funkcjami elementarnymi

Miejsce dla ASN.1

- ASN.1 to język do definiowania składni abstrakcyjnych
- Z ASN.1 związane są reguły kodowania, które określają składnię transferową dla każdej wartości ASN.1
  - podstawowe zasady kodowania (BER - Basic Encoding Rules)
  - kanoniczne zasady kodowania (CER - Canonical Encoding Rules)
  - wyróżnione Reguły Kodowania (DER - Distinguished Encoding Rules)
  - upakowane reguły kodowania (PER - Packed Encoding Rules)
  - reguły kodowania XML (XER - XML Encoding Rules)
- ASN.1 może być używany razem z innymi językami definicji takimi jak SDL

ASN.1 overview

- Koncepcje wiadomości i relacje między nimi



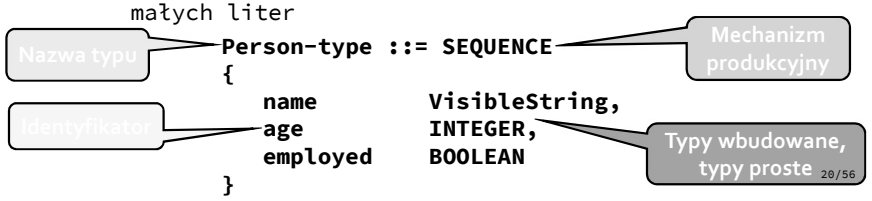
Specyfikacja ASN.1

Zestaw znaków ASN.1

- Składnik ASN.1 powinien składać się z sekwencji znaków:
  - A do Z
  - a do z
  - 0 do 9
  - : = , { } < > . @ ( ) [ ] - ' " | & ^ * ; !
- ASN.1 jest czuły na wielkość znaków!
- Typ czcionki (bold, italic, rozmiar …) nie ma znaczenia

Nazywanie

- Nazwy pól, składowych i elementów mogą być dowolnie długie
- Powszechne są długie nazwy
- Nazwy typów muszą zaczynać się od wielkich liter
- Nazwy identyfikatorów muszą zaczynać się od małych liter



Uwagi o składni

- UWAGA! Pierwsza litera identyfikatora określa jego kategorię

```
MyType ::= INTEGER          -- wielka litera → typ
myValue INTEGER ::= 100     -- mała litera → zmienna
```

- Brak ograniczników między definicjami
- Kolejność definicji nie jest istotna

Uwagi o składni

- Komentarze

```
-- To jest komentarz, koniec wiersza go zamyka
-- To jest komentarz, zamykają go podwójne myślniki --
```

- Inne

```
MultipartKeywords ::= SEQUENCE OF INTEGER
Hyphens-instead-of-underscores ::= INTEGER   -- No '_'
```


Podstawowe koncepcje

- ASN.1 jest językiem definiującym typy danych
- Podstawowe konstrukty typów
  - Rekord → typ SEQUENCE
  - Lista/tablica → typ SEQUENCE OF
  - Wzajemnie wykluczające się alternatywy (unia) → typ CHOICE
  - Typy podstawowe → typy podstawowe ASN.1
  - Stałe → wartości
- Strukturyzacja
  - Pakiet/moduł → moduł

Podstawowe koncepcje, cd.

- Przykład:

```
MyModule MODULE DEFINITIONS ::=
BEGIN
     MyType ::= INTEGER
     myValue INTEGER ::= 100
END
```

- ASN.1 jest podobny do specyfikacji typów w innych językach programowania

Strukturyzacja specyfikacji ASN.1

- Definicje ASN.1 znajdują się w modułach
- Moduły zawierają kompletny zestaw definicji typów, które są samowystarczalne
- Moduły mogą importować definicje z innych modułów i eksportować definicje do innych modułów

Wyrażenia EXPORTS/IMPORTS

- IMPORTS - określa listę typów zdefiniowanych w innych modułach
- EKSPORTS - określa listę typów zdefiniowanych w tym module, które są dostępne do użycia w innych modułach
  - jeśli instrukcje EXPORTS są pominięte, to dostępne są wszystkie typy
  - EKSPORTS ; (pusta lista typów) oznacza, że nic nie jest dostępne
- Wyrażenia IMPORTS/EXPORTS są opcjonalne i jeśli istnieją powinny być umieszczone na początku modułu po instrukcji BEGIN i przed jakąkolwiek definicją typu
- Średnik służy do zakończenia wyrażenia EXPORTS/IMPORTS

Moduły

- Moduł ASN.1 zawiera
  - nagłówek
    - nazwa modułu (odwołanie do modułu) - początkowym znakiem powinna być duża litera,
    - opcjonalnie globalny identyfikator modułu (OBJECT IDENTIFIER)
    - opcjonalnie domyślne tryby znakowania i rozszerzalności
  - ciało
    - opcjonalnie interfejsy międzymodułowe
    - wszystkie specyfikacje typów i wartości

Przykład modułu ASN.1

```
ExampleASN.1 DEFINITIONS ::=
    -- This is a comment
    -- Start of module
BEGIN

EXPORTS Student;
-- This can be re-used in other definitions

Student ::= SEQUENCE {
-- A sequence order is important
       age    INTEGER,
       name   IA5String
}

ClassRoom ::= SET OF Student
-- order is not important in a SET

END
```

Przykład modułu ASN.1

```
ItineraryModule DEFINITIONS AUTOMATIC TAGS ::=
BEGIN
  EXPORTS      -- Other modules can reference exported
    Itinerary; -- definitions. If EXPORTS is omitted
               -- then all definitions are visible.
               -- Hint: do not use EXPORTS.
  IMPORTS      -- Referenced definitions from other
    Travel     -- modules.
  FROM TravelModule;
  Itinerary ::=
    SEQUENCE (SIZE (1..maxNoOfTravels)) OF Travel
  maxNoOfTravels INTEGER ::= 20
END
```


Definicje typów i wartości

- Typy wbudowane
  - typy proste, podstawowe „cegiełki”
  - przydatne typy, predefiniowane typy strukturalne lub synonimy typów prostych dla niektórych typowych przypadków
- Definiowanie nowych typów
  - konstruktory typów do definiowania typów złożonych
  - konstruktory podtypów do definiowania podtypów
- Notacja dla definicji typu i wartości
  - <*Typereference*> ::= <*notation for the type*>
  - <*valuereference*> <*Type*> ::= <*notation for the value*>

Typy wbudowane

- Typ BOOLEAN
  - ma dwie wartości: TRUE i FALSE (lub 1 i 0)

doorOpen BOOLEAN ::= TRUE

Typy wbudowane

- Typ INTEGER
  - liczby całkowite: … -2,-1, 0, 1, 2 …

speed INTEGER (0..60) ::= 40

Typy wbudowane

- Typ ENUMERATED
  - używany, gdy preferowana jest lista pozycji, które są identyfikowane przez nazwy, a nie liczbę
  - nazwy wartości zawsze zaczynają się od małej litery

```
CarColors ::= ENUMERATED {black, red, white}
        myCar CarColors ::= white
```


Typy wbudowane

- Typ REAL
  - liczby rzeczywiste:
    - 3.14
    - {mantissa 3.14159, base 10, exponent -5}
    - 0, PLUS-INIFINITY, MINUS-INFINITY

Typy wbudowane

- Typ BIT STRING
  - tablica bitów, która niekoniecznie jest wielokrotnością 8 bitów, np. maska bitowa
  - każdemu bitowi można przypisać znaczenia z osobna
  - wartości mają 3 możliwe formy:
    - binarną - '011'B,
    - szesnastkową - '6'H
    - nazwaną - {windowOpen, engineOn}

Typy wbudowane

- Typ BIT STRING

```
Sensors ::= BIT STRING {
      doorOpen (0),
      windowOpen (1),
      engineOn (2)
}
myStatus Sensors ::= {windowOpen, engineOn}
```

Typy wbudowane

- Typ NULL
  - jedna wartość: NULL
  - używany, gdy potrzebny jest symbol zastępczy, dla którego nie ma wartości
  - najczęściej używany jako alternatywa w typie CHOICE lub jako opcjonalny składnik typu SEQUENCE

```
Toys ::= CHOICE {
cars Cars,
dolls Dolls,
none NULL}
```


Typ ANY – wytrych (w nowszych wersjach nieużywany)

- Typ ANY można ustawić na dowolny typ

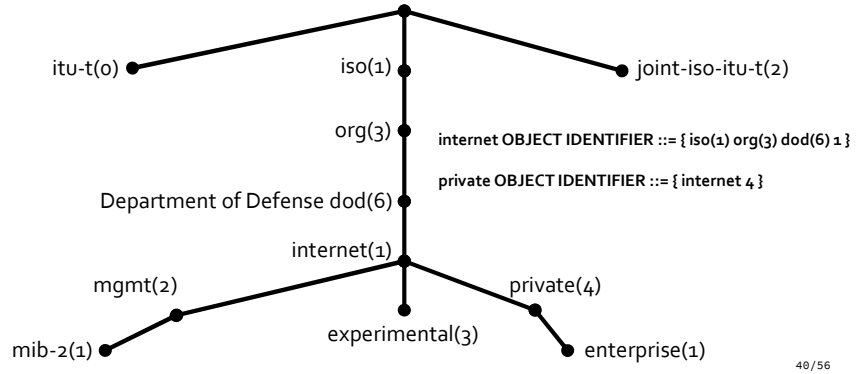
```
MyThing ::= ANY
```

  - oznacza to, że definicja typu wykracza poza zakres ASN.1
  - odnosi się do pozycji, które mają zostać zdefiniowane w przyszłości

Typ OBJECT IDENTIFIER

- Identyfikatory obiektów definiują globalnie unikalną przestrzeń nazw
- Przestrzeń nazw identyfikatorów obiektów jest zdefiniowana przez hierarchiczną strukturę drzewa
- Każdy łuk drzewa jest oznaczony wartością liczbową
- Globalne władze są odpowiedzialne za przydzielanie wartości do łuków najwyższego poziomu
- Władze lokalne są odpowiedzialne za wartości łuków niższego poziomu

Typ OBJECT IDENTIFIER



Wbudowane typy łańcuchów znaków

- VisibleString
  - podzbiór ASCII, który nie zawiera znaków sterujących (znaki drukowalne plus spacja)
- UTF8String
  - dowolne znaki (od egipskich hieroglifów do ASCII)
- IA5String
  - ASCII i znaki sterujące
- NumericString
  - tylko cyfry i spacja
- Inne (rzadko lub wcale nieużywane)
  - NumericString, TeletexString, VideotexString, IA5String, GraphicString, GeneralString, UniversalString, BMPString

Typy wbudowane dla czasu i daty

- Typ DATE
  - reprezentacja daty w formacie "YYYY-MM-DD"

```
harvardEstablished DATE ::= "1636-09-18"
```

- Typ TIME-OF-DAY
  - reprezentacja godziny w trakcie dnia w formacie "HH:MM:SS"

```
callTime TIME-OF-DAY ::= "18:30:23"
```

- DATE-TIME
  - data i godzina w formacie "YYYY-MM-DDTHH:MM:SS"

```
callTime DATE-TIME ::= "2021-11-22T18:30:23"
```


Typy złożone (produkcyjne)

- Każdy typ złożony składa się z części umieszczonych w jednej lub kilku liniach w następującej kolejności:
  - nazwa nowej kolekcji sekwencji produkcyjnych
  - znaki ::=
  - jednej lub więcej alternatywnych zbiorów sekwencji produkcyjnych

```
ClassMember ::= CHOICE {
  student   Student,
  lecturer   Lecturer
}
```


Mechanizm produkcyjny SEQUENCE

- SEQUENCE to sekwencja elementów różnego typu, w której kolejność jest istotna

```
Contact ::= SEQUENCE {
      name VisibleString,
      phone NumericString
}

driver Contact ::= {name "J.Smith", phone "7325555555"}
```

Mechanizm produkcyjny SEQUENCE OF

- SEQUENCE OF jest sekwencją elementów tego samego typu, w której kolejność jest istotna

```
breakTimes SEQUENCE OF TIME-OF-DAY ::=
{"10:00:00", "12:00:00", "14:45:00"}
```


Mechanizmy produkcyjne SET i SET OF

- Podobne do SEQUENCE i SEQUENCE OF, ale kolejność w nich nie jest istotna
- Powodują, że proces kodowania jest wolny i nie zaleca się ich stosowania

DEFAULT i OPTIONAL

- Słowo kluczowe DEFAULT
  - Określa domyślną wartość elementu SEQUENCE lub SET, którą należy przyjąć, jeśli wartość tego elementu nie jest podana
- Słowo kluczowe OPTIONAL
  - Określa element, którego wartość można pominąć

```
Person::= SEQUENCE {
     name                UTF8String,
     number-of-parents   INTEGER DEFAULT 2,
     spouse-name         UTF8String OPTIONAL
}
```


Mechanizm produkcyjny CHOICE

- CHOICE – kolekcja elementów, z których w danym momencie może być obecny tylko jeden z nich

```
Location ::= CHOICE {
     streetAddress  Address,
     intersection   Intersection,
     landmark       LandMarkName,
     gpsCoordinates GpsInfo
}
meetAt Location ::= landmark: "Statue of
Liberty"
```

Znakowanie typów

- Oznaczony typ to nowy typ, który jest dekodowany i kodowany identycznie jak typ, z którego pochodzi, ale ma inny znacznik (ang. *tag*)
- Znaczniki służą do rozróżniania elementów tego samego typu w sekwencjach (głównie opcjonalnych)

```
Point ::= SEQUENCE {
     x INTEGER OPTIONAL,
     y INTEGER OPTIONAL
}
```


Znakowanie typów

- Oznaczony typ to nowy typ, który jest dekodowany i kodowany identycznie jak typ, z którego pochodzi, ale ma inny znacznik (ang. *tag*)
- Znaczniki służą do rozróżniania elementów tego samego typu w sekwencjach (głównie opcjonalnych)

```
Point ::= SEQUENCE {
     x INTEGER OPTIONAL,
     y INTEGER OPTIONAL
}
```

Przy podawaniu wartości można:
1. nie podać żadnej
2. podać jedną
3. podać dwie

Znakowanie typów

- Oznaczony typ to nowy typ, który jest dekodowany i kodowany identycznie jak typ, z którego pochodzi, ale ma inny znacznik (ang. *tag*)
- Znaczniki służą do rozróżniania elementów tego samego typu w sekwencjach (głównie opcjonalnych)

```
Point ::= SEQUENCE {
     x INTEGER OPTIONAL,
     y INTEGER OPTIONAL
}
```

Przy podawaniu wartości można:
1. nie podać żadnej
2. podać jedną - niejednoznaczne
3. podać dwie

Znakowanie typów

- Oznaczony typ to nowy typ, który jest dekodowany i kodowany identycznie jak typ, z którego pochodzi, ale ma inny znacznik (ang. *tag*)
- Znaczniki służą do rozróżniania elementów tego samego typu w sekwencjach (głównie opcjonalnych)

```
Point ::= SEQUENCE {
     x [0] INTEGER OPTIONAL,
     y [1] INTEGER OPTIONAL
}
```

Jawne podanie znacznika (muszą być jednoznaczne)

Znaczniki

```
M DEFINITIONS IMPLICIIT TAGS ::=
BEGIN
-- Składowa opcjonalna i następne muszą
-- być rozróżniane
S ::= SEQUENCE {
  a [0] INTEGER OPTIONAL,
  b [1] INTEGER OPTIONAL,
  c [2] INTEGER OPTIONAL
}
-- Alternatywy muszą być rozróżniane
C ::= CHOICE {
  a [0] INTEGER,
  b [1] INTEGER
}
END
```

```
M DEFINITIONS AUTOMATIC TAGS ::=
BEGIN
-- Znaczniki automatyczne =>
-- nie ma potrzeby ręcznego
-- znakowania
S ::= SEQUENCE {
  a INTEGER OPTIONAL,
  b INTEGER OPTIONAL,
  c INTEGER OPTIONAL
}
C ::= CHOICE {
  a INTEGER,
  b INTEGER
}
END
```


Podtypy

- Proste podtypy
  - INTEGER (0..255) ograniczenie zbioru wartości
  - służą do poprawy wydajności kodowania (podtypy wymagają mniej bajtów do kodowania niż typy o pełnej długości)
- Złożone podtypy
  - SIZE, FROM
  - INTERSECTION, UNION, EXCEPT, ALL

Złożone podtypy

- Pojedyncze wartości
  Divisors-of-6 ::= INTEGER (1 | 2 | 3 | 6)
- Podtyp zawierający
  Divisors-of-18 ::= INTEGER (INCLUDES Divisors-of-6 | 9 | 18)
- Zakres wartości
  TeenAgeYears ::= (13 .. 19)
- Dozwolony alfabet
  BooleanValue ::= IA5String (FROM ('T' | 'F'))
- Ograniczenie rozmiaru
  BaseballTeamRoster ::= SET SIZE (1..25) OF PlayerNames

ASN.1 a języki programowania

- Brak części wykonywalnej – ograniczona do specyfikacji typów danych
- Modelowanie wartości opcjonalnych i domyślnych w strukturach
- Jawne typy danych nieustrukturyzowanych: BIT STRING, OCTET STRING
- Kolejność definicji jest nieznacząca – zazwyczaj stosuje się kolejność od góry do dołu
- Brak ograniczników między składnikami
- Każdy typ jest powiązany ze znacznikiem