ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2000
Seria: ELEKTRONIKA z. 12 Nr kol. 1492

Jerzy WOJTUSZEK
Politechnika Śląska, Instytut Elektroniki

WYKORZYSTANIE NOTACJI ASN.1 DO KODOWANIA DANYCH SYGNALIZACYJNYCH W SIECIACH ISDN

Streszczenie. Scharakteryzowano abstrakcyjną syntaktykę ASN.1 oraz reguły kodowania danych. Omówiono sposób określania struktury danych dla usług dodatkowych w sieciach ISDN. Przedstawiono przykład kodowania komponentu (jednostki danych protokołowych).

APPLICATION OF ASN.1 NOTATION TO CODING OF SIGNALLING DATA IN ISDN NETWORKS

Summary. The ASN.1 abstract syntax and rules of data coding are characterized. The method of defining structure of data for supplementary services in ISDN networks is discussed. An example of component (protocol data unit) encoding is presented.

1. Wprowadzenie

Realizacja protokołów telekomunikacyjnych polega na przesyłaniu bloków danych zwanych jednostkami danych protokołowych (Protocol Data Unit - PDU). Ważnym elementem specyfikacji każdego protokołu jest określenie struktury (formatu) poszczególnych PDU wykorzystywanych w tym protokole. W tradycyjnych metodach definiowania formatu PDU określa się zwykle znaczenie kolejnych oktetów lub bitów tworzących PDU. Używa się do tego celu rysunków podobnych do rys.1 wspomaganych stosownym tekstem.

Pewną wadą tradycyjnych metod definiowania struktur PDU jest to, że jednocześnie definiowane są w nich typy danych tworzących PDU (np. liczby całkowite, łańcuchy znaków, itp.) oraz sposoby kodowania tych danych. W niektórych sytuacjach korzystne jest bowiem określanie budowy PDU jedynie na poziomie typów danych wchodzących w skład PDU (mogą to być typy o złożonej strukturze) bez określania sposobu kodowania tych danych

podczas przesyłania, czyli bez określania postaci transferowej danych. Potrzeba taka pojawia się np. podczas specyfikowania protokołów warstwy aplikacji [1]. Przyjmuje się wówczas, że postać transferowa danych zostanie wybrana jako jedna z wielu możliwych przez warstwę prezentacji na zasadzie dwustronnych negocjacji. Zasady kodowania danych na styku między warstwą aplikacji i prezentacji mogą być wtedy zależne od specyfiki konkretnego systemu - mogą się one różnić po stronie nadawczej i odbiorczej. Z przedstawionych rozważań wynika potrzeba poszukiwania nowej metody opisu formatu danych przesyłanych w sieciach telekomunikacyjnych. Metoda taka powinna rozdzielać opis na poziomie typów danych i opis zasad kodowania.

Prezentacja struktury przesyłanych danych na poziomie typów danych nazywana jest abstrakcyjną syntaktyką (syntaktyka abstrahująca od kodowania). Najbardziej znaczącym przykładem takiej syntaktyki jest notacja ASN.1 (Abstract Syntax Notation One) przedstawiona w standardzie X.208 [2]. Dla notacji tej opracowano standard X.209 [3], w którym przedstawiono sposoby kodowania poszczególnych typów danych. Zasady kodowania określone w standardzie X.209 należy traktować jako jedne z wielu możliwych dla notacji ASN.1

Notacja ASN.1 oraz reguły kodowania określone w standardzie X.209 znalazły zastosowanie m.in. w sieciach ISDN, gdzie wykorzystane są do opisu struktury danych sygnalizacyjnych związanych z realizacją tzw. usług dodatkowych (funkcja warstwy aplikacji). W chwili obecnej dla sieci ISDN zdefiniowano kilkanaście takich usług, a ich przykładami mogą być [4,5,6,12]:

- User-to-User Signalling (UUS) - przesyłanie danych między użytkownikami za pomocą wiadomości sygnalizacyjnych przesyłanych w kanale D (za darmo),
- Advice of Charge (AOC) - informowanie użytkownika o taryfie dla zestawianego połączenia oraz przekazywanie informacji o opłacie - w trakcie trwania połączenia i po jego zakończeniu,
- Completion of Calls to Busy Subscriber (CCBS) - automatyczne zestawianie połączenia z wcześniej zajętym abonentem.

ISDN-owskie zasady opisu i kodowania danych zwiazanych z usługami dodatkowymi przyjęte zostały w telefonii komórkowej GSM oraz w systemie łączności bezprzewodowej DECT. Zestaw usług dodatkowych oraz procedury ich realizacji są tam również wzorowane na sieciach ISDN.

Dla poszczególnych usług dodatkowych w sieci ISDN istnieją oddzielne standardy ETSI. Każdy z tych standardów zawiera rozdział, w którym za pomocą notacji ASN.1 opisana jest struktura danych sygnalizacyjnych (sterujących) specyficznych dla danej usługi dodatkowej. Prawidłowa interpretacja tych opisów wymaga skojarzenia ze sobą kilku standardów CCITT (ITU) i ETSI pochodzących z różnych okresów (co najmniej standardy wymienione w spisie literatury). Standardy te nie zawsze w sposób jawny odwołują się do siebie (względnie odwołują się za pomocą notacji ASN.1), co utrudnia ich studiowanie czytelnikom nie zajmującym się wcześniej omawianą dziedziną. Odczuwa się brak publikacji ujmujących

w sposób syntetyczny problemy konkretnych zastosowań notacji ASN.1. Niniejszy artykuł jest próbą zapełnienia wspomnianej luki publikacyjnej w odniesieniu do zagadnień związanych z wkorzystaniem notacji ASN.1 do definiowania danych sygnalizacyjnych przesyłanych w sieciach ISDN w ramach procedur realizujących usługi dodatkowe.

2. Podstawowe zasady realizacji usług dodatkowych w sieciach ISDN

W łączach abonenckich sieci ISDN wszystkie dane sygnalizacyjne przesyłane są w kanale D wyodrębnionym na zasadzie podziału czasu i mają postać wiadomości [7,8,9,12]. Wiadomości te przesyłane są w polu informacyjnym ramek protokołu warstwy drugiej, tj. protokołu LAPD. Wiadomość składa się z nagłówka określającego m.in. typ tej wiadomości oraz z pewnej liczby (rzadko kiedy więcej niż 5) tzw. elementów informacyjnych zawierających właściwe dane sygnalizacyjne. Standard [9] definiuje 35 rodzajów elementów informacyjnych dla wiadomości przesyłających dane sygnalizacyjne. Z punktu widzenia realizacji usług dodatkowych zasadnicze znaczenie ma element informacyjny typu FACILITY. Element informacyjny FACILITY pojawia się najczęściej w wiadomości również nazywającej się FACILITY, ale może być on umieszczany także w innego rodzaju wiadomościach.

Dla każdej usługi dodatkowej określony jest w odpowiednim standardzie protokół przesyłania danych sygnalizacyjnych polegający na wymianie pomiędzy obiema stronami łącza abonenckiego jednostek danych protokołowych zwanych komponentami. Komponenty te przesyłane są w polu Komponent elementu informacyjnego FACILITY (rys.1). W sieciach ISDN notacja ASN.1 dotyczy głównie struktury komponentów przesyłanych za pomocą elementów informacyjnych FACILITY.

Każdej usłudze dodatkowej odpowiada zestaw elementarnych operacji. Dla każdej z tych operacji zdefiniowany jest z kolei komponent Invoke i ewentualnie komponenty Return result i Return error [11,13]. Zdefiniowany jest również komponent typu Reject, którego struktura nie jest zależna od rodzaju operacji.

Każda elementarna operacja związana z realizacją określonej usługi dodatkowej inicjowana jest przez jedną ze stron (abonent lub sieć) wysłaniem komponentu typu Invoke, na co zazwyczaj strona przeciwna odpowiada komponentem Return result, Return error lub Reject.

8	7	6	5	4	3	2	1	oktet
0	0	0	1	1	1	0	0	1
Identyfikator elementu informacyjnego								
długość dalszej części								2
1	0	0	1	0	0	0	1	3
ext.	Zapas		dyskryminator usługi = supplementary service					
pole Komponent								4 . . .

Rys.1. Budowa elementu informacyjnego FACILITY
Fig. 1. The structure of the FACILITY information element

3. Charakterystyka notacji ASN.1

3.1. Zasady ogólne

Idea notacji ASN.1 polega na tym, że jednostka danych protokołowych (PDU, komponent) traktowana jest jako pewien typ danych skomponowany według określonych reguł (np. sekwencja) z danych prostszego typu. Dane te znowu muszą być dekomponowane na dane jeszcze prostszego typu, aż do sprowadzenia wszystkich danych do typów prostych (np. liczby określonego rodzaju).

Standard X.208 definiuje kilkanaście typów danych, przy czym część z nich to typy proste, natomiast reszta to typy złożone komponowane z typów prostych. Dla każdego z powyższych typów określono sposób zapisywania wartości.

Nazwy typów rozpoczynają się zawsze od dużej litery, przy czym nazwy typów zdefiniowanych w standardzie X.208 składają się z samych dużych liter. Nazwy typów czasami pojawiają się w zapisach zwanych „typami nazwanymi" (NamedType) o postaci

identyfikator Typ

Identyfikator rozpoczynany jest zawsze od małej litery (w identyfikatorach i nazwach typów dopuszczalny jest znak „minus"). Często w praktyce nazwy identyfikatora i Typu różnią się tylko „rozmiarem" pierwszej litery (identyfikator - mała, Typ - duża). Identyfikator może być np. nazwą pola PDU, w którym przesyła się dane typu „Typ". Zapis składający się tylko z nazwy typu (bez identyfikatora) też zaliczany jest do kategorii „typ nazwany". Typom nazwanym odpowiadają „wartości nazwane" (NamedValue) zapisywane w formie

identyfikator Wartość

Każdemu typowi danych odpowiada etykieta (tag) zastępująca podczas transferu nazwę typu. Etykieta składa się z klasy i numeru określającego typ w obrębie klasy. Rozróżniane są cztery klasy etykiet: universal - dla typów zdefiniowanych w standardzie X.208, application - typy zdefiniowane w innych standardach ISO lub ITU, private - typy zdefiniowane przez inne niż ISO lub ITU organizacje, context-specific - etykiety interpretowane w zależności od kontekstu (znaczenie wartości etykiety zależne od miejsca, w którym jej użyto). W przypadku usług dodatkowych sieci ISDN praktyczne znaczenie mają tylko klasy universal i context-specific. W tabeli 1 przedstawiono etykiety dla wybranych typów danych zdefiniowanych w X.208.

Tabela 1

Etykiety dla wybranych typów danych zdefiniowanych w X.208

Etykieta (Klasa numer)	Typ
UNIVERSAL 1	BOOLEAN
UNIVERSAL 2	INTEGER
UNIVERSAL 5	NULL
UNIVERSAL 6	OBJECT IDENTIFIER
UNIVERSAL 16	SEQUENCE i SEQUENCE OF

Zapisy ASN.1 tworzą bloki zwane modułami (często ujmowane w ramkę). Moduł ASN.1 zawiera kolejno:

- nazwę modułu - jej druga część (o ile istnieje) zapisywana jako wartość typu OBJECT IDENTIFIER,
- słowo DEFINITIONS,
- zapis informujący o domyślnym sposobie traktowania etykiet typów bazowych w typach etykietowanych podczas transferu danych (EXPLICIT TAGS - etykietowanie jawne lub IMPLICIT TAGS - etykietowanie niejawne, czyli możliwość pominięcia etykiety podczas transferu) - brak tego zapisu oznacza etykietowanie jawne,
- znaki „::=",
- słowo BEGIN,
- wykaz symboli eksportowanych do innych modułów rozpoczynający się od słowa EXPORTS,
- wykaz symboli importowanych z innych modułów rozpoczynający się od słowa IMPORTS; nazwa modułu, z ktorego importowane są symbole poprzedzona jest słowem FROM,
- definicje typów i definicje wartości (część zasadnicza),
- słowo END.

Definicjom typów i wartości odpowiadają zapisy zwane produkcjami o postaci:

Nazwa_nowego_typu ::= Typ_pierwotny

lub nazwa_wartości Typ_wartości ::= Wartość

Typ pierwotny może być typem prostym lub złożonym. Etykieta typu pierwotnego przechodzi na nowy typ.

W zapisach ASN.1 można używać komentarzy rozpoczynających się od dwóch znaków „minus".

3.2. Typy danych

Poniżej omówione zostaną wybrane typy danych zdefiniowane w standardzie X.208.

Zmienne dwuwartościowe - BOOLEAN

Wartości zapisywane są jako TRUE i FALSE.

Liczby całkowite - INTEGER

W praktyce najczęściej korzysta się z typów będących podzbiorami liczb całkowitych zapisywanych jako

INTEGER{lista_wartości}

lub INTEGER(wartość_minimalna..wartość_maksymalna)

lub INTEGER{lista_wartości}(wartość_minimalna..wartość_maksymalna)

Dwa ostatnie zapisy należy traktować jako podtypy (Subtype) typu INTEGER. Każda z wartości występująca na liście wartości zazwyczaj ma postać

identyfikator(wartość)

Brzmienie identyfikatora zwykle kojarzy się ze znaczeniem określonej wartości. Przykłady:

INTEGER {service1(1),service2(2),service3(3)}

typ składający się z trzech liczb całkowitych o wartościach 1, 2 i 3 i o identyfikatorach odpowiednio service1, service2, service3.

INTEGER (-32768..32767)

typ składający się z kolejnych liczb całkowitych od -32768 do 32767.

INTEGER {service1(1),service2(2),service3(3)}(1..3)

jest to zapis takiego samego typu jak w pierwszym przykładzie, lecz rozszerzony (nadmiarowo) o zakres liczb tworzących typ.

Wartości tpu INTEGER zapisuje się dziesiętnie z pomijaniem zer prowadzących.

Typ NULL

Jest to specyficzny typ zawierający tylko jedną wartość zapisywaną również jako NULL. Praktycznie rzecz biorąc typ ten pojawia się zwykle jako składnik typu złożonego CHOICE, gdzie wartość NULL jest wybierana jako wartość typu CHOICE wtedy, gdy żaden inny składnik nie może zostać wybrany.

Identyfikator obiektu - OBJECT IDENTIFIER

Podstawową formą zapisu wartości jest

{lista_składników}

Elementy listy składników rozdzielane są spacjami lub znakami nowego wiersza i najczęściej mają jedną z następujących postaci: *identyfikator*, *liczba* lub *identyfikator(liczba)*. Wartości omawianego typu określają pewną docelową wartość liczbową poprzedzoną danymi identyfikującymi organizację, która wartości tej przypisała jakieś znaczenie oraz dokument, w którym tego dokonano. Dopuszczalne wartości dwóch pierwszych składników (początkowe gałęzie drzewa identyfikatorów) określone są w standardzie X.208. Struktura dalszej części listy składników definiowana jest przez dokumenty wydawane przez odpowiednie organizacje standaryzacyjne (np. [10]), przy czym ostatnim składnikem jest zawsze docelowa wartość liczbowa. Każdemu składnikowi odpowiada liczba, która może być określona jawnie w zapisie składnika bądź też niejawnie za pomocą identyfikatora. W drugim z wymienionych przypadków liczby odpowiadające identyfikatorom określone są w odpowiednich standardach.

W przypadku zapisu wartości typu OBJECT IDENTIFIER zdefiniowanych przez standardy ETSI [10] pełna lista składników składa się z sześciu pozycji. Przykładem takiego zapisu może być:

{ccitt identified-organization etsi(0) 359 operations-and-errors(1) 4}

Pierwsze trzy składniki mają zawsze jednakową postać i określają ETSI jako organizację, która opracowała standard: Zgodnie z standardem X.208 pierwszemu składnikowi (ccitt) odpowiada liczba 0, natomiast drugiemu (identified-organization) - liczba 4. Liczba odpowiadająca trzeciemu składnikowi (etsi(0)) podana jest jawnie i wynosi 0. Dalsze składniki oznaczają: końcówkę numeru standardu ETSI (359 - w celu uzyskania pełnego numeru należy dodać 300 000), moduł ASN.1 w tym standardzie definiujący wartość docelową (operations-and-errors(1)) i docelową wartość liczbową (4). Przedstawiony wyżej przykładowy zapis jest równoważny poniższemu zapisowi, w którym wszystkie składniki reprezentowane są przez liczby:

{0 4 0 359 1 4}

Inną często spotykaną formą zapisu wartości omawianego typu jest

{nazwa_innej_wartości_typu_OBJECT_IDENTIFIER lista_składników}

W takim przypadku początkowe składniki listy składników zastąpione są przez nazwę wartości typu OBJECT_IDENTIFIER zdefiniowaną w innym miejscu modułu. Poniżej przedstawiono przykład takigo zapisu.

Definicja wartości cCBSOID typu OBJECT IDENTIFIER:

```
cCBSOID OBJECT IDENTIFIER  ::=  {ccitt identified-organization etsi(0) 359
                                 operations-and-errors(1)}
```

Pełny zapis wartości docelowej ma wówczas postać:

{cCBSOID 4}

i jest równoważny zapisom omówionym wcześniej.

Sekwencja - SEQUENCE

Omawiany typ jest typem złożonym będącym uporządkowaną listą innych typów, przy czym niektóre z tych typów mogą być opcjonalne - ich wartości mogą być pomijane w czasie transferu. Zapis typu SEQUENCE ma postać:

SEQUENCE{lista_elementów}

W najczęstszych przypadkach składnikami listy elementów są typy nazwane, przy czym typy opcjonalne mają dodane słowo OPTIONAL. Przykładem zapisu typu SEQUENCE może być:

```
SEQUENCE{
    pierwszy    INTEGER,
    drugi       BOOLEAN OPTIONAL,
    trzeci      OBJECT IDENTIFIER}
```

Niekiedy elementy listy są tzw. typami etykietowanymi (omówionymi dalej). Stosowanie tych typów umożliwia jednoznaczne rozpoznanie poszczególnych składników w sytuacji, gdy lista elementów zawiera składniki opcjonalne.

Zapis wartości typu SEQUENCE ma postać

{lista_wartości_elementów}

gdzie każdy składnik listy wartości elementów jest wartością nazwaną.

Sekwencja jednakowych typów - SEQUENCE OF Typ

Powyższy typ jest sekwencją, której składniki są jednakowego typu (typu „Typ"). Często stosowany jest podtyp omawianego typu, dla którego określona jest minimalna i maksymalna liczba składników. Zapis tego podtypu ma postać:

SEQUENCE SIZE (min_składników..max_składników) OF Typ

Zapisem wartości jest

{lista_wartości_elementów}

Wybór - CHOICE

Omawiany typ jest typem złożonym, będącym listą innych typów, przy czym jego wartością jest wartość jednego ze składników tej listy. Zapis typu CHOICE ma postać:

CHOICE{lista_alternatywnych_typów}

Elementami listy alternatywnych typów są typy nazwane, przy czym typy te powinny się różnić etykietami. Z tego powodu często składnikami listy są typy etykietowane. Etykietę typu CHOICE należy traktować jak zmienną - jest ona równa etykiecie wybranego w danym przypadku składnika. Zapis wartości typu CHOICE ma postać wartości nazwanej wybranego składnika.

Typ etykietowany - tagged type

Powyższy typ występuje jako składnik typów złożonych (np. SEQUENCE, CHOICE). Jego stosowanie ma na celu jednoznaczne rozróżnienie składników w tych typach. Typ etykietowany powstaje z innego typu (typ bazowy) poprzez przypisanie mu nowej etykiety. Zapis omawianego typu ma postać:

[Klasa numer] Typ_bazowy
lub [Klasa numer] IMPLICIT Typ_bazowy
lub [Klasa numer] EXPLICIT Typ_bazowy

gdzie „Klasa numer" jest nową etykietą. W praktyce zapis klasy etykiety w nawiasach prostokątnych jest zwykle pomijany, co odpowiada klasie context-specific. Słowo IMPLICIT oznacza, że etykieta typu bazowego nie musi być przesyłana podczas transferu danych (etykietowanie niejawne), natomiast słowo EXPLICIT - że etykieta typu bazowego powinna być przesyłana (etykietowanie jawne). W przypadku braku powyższych słów (zapis pierwszy) o sposobie etykietowania podczas transferu decyduje informacja zawarta w nagłówku modułu ASN.1, przy czym dla typów CHOICE i ANY zawsze stosowane jest etykietowanie jawne niezależnie od tej informacji. Etykietowanie niejawne zmniejsza rozmiary postaci transferowej danych.

Typ określony pośrednio - ANY DEFINED BY identyfikator

Powyższy typ może być używany tylko jako składnik typu SEQUENCE lub SET i tylko wtedy, gdy „identyfikator" pojawia się w typie nazwanym będacym innym nieopcjonalnym składnikiem tego samego typu SEQUENCE lub SET. W takim przypadku słowo ANY zostaje zastąpione typem wynikającym z wartości składnika określonego przez „identyfikator". Sposób przypisania wartościom tego składnika typów zastępujących ANY określony jest za pomocą komentarza lub specjalnego dokumentu. Etykieta typu jest nieokreślona.

Przykładowo, omawiany typ może być użyty w następujący sposób:

```
SEQUENCE{
    value       INTEGER,
    parameter   ANY DEFINED BY value}
```

W takim przypadku typ odpowiadający identyfikatorowi „parameter" będzie wynikał z wartości składnika „value INTEGER".

3.3. Makronotacje

Dodatkowym mechanizmem służącym do definiowania nowych typów danych i zapisywania wartości tych typów są makronotacje. Podobnie jak w językach programowania makronotacja jest pewnym zapisem tworzonym na bazie makrodefinicji. Makrodefinicja określa:
- sposób tworzenia zapisu definiującego nowy typ (notacja nowego typu),
- sposób zapisywania wartości tego typu (notacja wartości nowego typu),
- wartości standardowego typu odpowiadające wartościom nowego typu (wartości zwracane).

Przykładowo [2], za pomocą makronotacji można zdefiniować nowe typy o postaci

PAIR
TYPEX=*xxx*
TYPEY=*yyy*

z zapisami wartości w formie

(X=*uuu*,Y=*vvv*)

i wartościami zwracanymi standardowego typu

SEQUENCE{*xxx*,*yyy*}

gdzie xxx, yyy - stndardowe typy ASN.1, *uuu*, *vvv* - wartości typu odpowiednio *xxx* i *yyy*. Słowo PAIR jest nazwą makronotacji (nazwy makronotacji pisane są dużymi literami).

Odwołania do makronotacji można podzielić na notacje nowych typów i notacje wartości nowych typów. Notacja nowego typu składa się z nazwy makronotacji, po której następuje zapis składający się z tekstów stałych (np. TYPEX=, TYPEY=) i parametrów (np. *xxx*, *yyy*). Teksty stałe zwykle pełnią rolę nazw parametrów. W przypadku makronotacji PAIR przykładem notacji nowego typu może być

PAIR
TYPEX=INTEGER
TYPEY=BOOLEAN

Dla tak zdefiniowanego nowego typu notacja wartości może mieć postać:

(X=3,Y=TRUE)

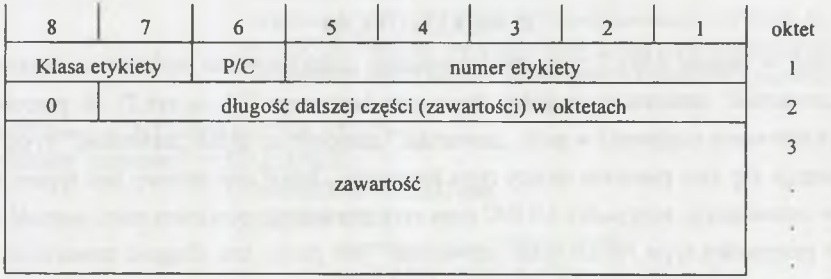
Standardowy typ wartości zwracanych nie musi mieć wyraźnego związku z notacją nowego typu. W rozpatrywanym przykładzie związek taki raczej istnieje - makronotacja PAIR jest właściwie inną formą zapisu typu SEQUENCE. Może się jednak zdarzyć, że nowemu typowi o dużej złożoności odpowiadać będą wartości np. typu INTEGER.

Notacja nowego typu może być umieszczana w miejscach, gdzie normalnie umieszczane są nazwy typów standardowych, a więc np. w zapisach określających zawartości okre-

ślonych pól komponentu. W takich przypadkach w polu komponentu przesyłana będzie wartość typu standardowego zwracana przez nowy typ.

4. Zasady kodowania danych określone w standardzie X.209

W najczęstszych przypadkach każdy typ danych (prosty lub złożony) kodowany jest w sposób przedstawiony na rys.2 i w tabeli 2. Jeżeli kodowany typ jest typem złożonym (np. SEQUENCE), to w polu „zawartość" znajdują się obszary o takiej samej strukturze odpowiadające poszczególnym składnikom tego typu. W przypadku typu prostego pole „zawartość" określa wartość tego typu (np. jest zapisem liczby). Wartość zero bitu 8 oktetu 2 oznacza, że długość pola „zawartość" zapisana jest w jednym oktecie, tj. na bitach 1 ÷ 7 oktetu 2 (wartość jeden oznacza zapis za pomocą większej liczby oktetów).



P/C: 0 - typ prosty (primitive), 1 - typ złożony (constructed),

Rys. 2. Kodowanie pojedynczego typu danych
Fig. 2. Coding of a single data type

Tabela 2

Kodowanie klasy etykiety

Klasa	Bit 8	Bit 7
Universal	0	0
Application	0	1
Context-specific	1	0
Private	1	1

Dla większości typów reguły kodowania wartości są proste i zgodne z intuicją. Wyjątkiem jest kodowanie wartości typu OBJECT IDENTIFIER oraz typu etykietowanego i typu NULL, które omówione zostanie poniżej.

W przypadku typu OBJECT IDENTIFIER w polu „zawartość" następują kolejno po sobie zapisy liczb zwanych subidentyfikatorami. Pierwszy subidentyfikator obliczany jest jako (X*40)+Y, gdzie X - wartość pierwszego składnika typu OBJECT IDENTIFIER, Y - wartość drugiego składnika (omawiany typ zawiera zawsze co najmniej dwa składniki, przy czym wartość drugiego składnika jest zawsze mniejsza od 40). Każdy następny subidentyfikator jest równy składnikowi o numerze o jeden wyższym.

Pojedynczy subidentyfikator zależnie od wartości kodowany jest na jednym lub wielu oktetach, przy czym najbardziej znaczący bit wszystkich oktetów z wyjątkiem ostaniego ma wartość jeden. Pozostałe bity powyższych oktetów po połączeniu ze sobą tworzą wartość subidentyfikatora traktowaną jako liczba całkowita bez znaku, przy czym pierwszy oktet zawiera najbardziej znaczącą część tej liczby, natomiast ostatni oktet - najmniej znaczącą. Przykładowo wartość

{0 4 0 359 1 4}

będzie reprezentowana przez następującą sekwencję oktetów (zapis szesnastkowy):

04, 00, 82, 67, 01, 04

Jeżeli w zapisie ASN.1 typu etykietowanego zadeklarowano etykietowanie jawne, to w polu „zawartość" umieszcza się pełny zapis typu bazowego (jak na rys.2). W przeciwnym razie (etykietowanie niejawne) w polu „zawartość" znajduje się tylko „zawartość" typu bazowego - pomija się dwa pierwsze oktety typu bazowego. Jeżeli typ bazowy jest typem złożonym, to w omawianym przypadku bit P/C typu etykietowanego powinien mieć wartość jeden.

W przypadku typu NULL pole „zawartość" jest puste, tzn. długość zawartości określona w drugim oktecie wynosi zero.

5. Opis struktury ogólnej komponentów za pomocą notacji ASN.1

Struktura ogólna komponentów (PDU) używanych do realizacji usług dodatkowych w sieciach ISDN przedstawiona została za pomocą notacji ASN.1 w tabeli 3 zaczerpniętej z [11]. Zgodnie z tą tabelą 3 (definicja typu Components) i z wcześniejszymi ustaleniami rozróżniane są cztery rodzaje komponentów: Invoke, Return result, Return error i Reject. Powyższym komponentom odpowiadają typy odpowiednio: InvokeComponent, ReturnResultComponent, ReturnErrorComponent i RejectComponent. Każdy z wymienionych typów zdefiniowany jest jako sekwencja danych innych typów. Poszczególne elementy tych sekwencji reprezentują pola komponentu. Postać transferowa komponentu zawierać będzie dodatkowe pole (nagłówek) identyfikujące rodzaj komponentu.

Jak wynika z tabeli 3, komponent Invoke składa się z następujących pól:

- invokeID,
- linked-ID,
- operation-value,
- argument.

Tabela 3

Struktura komponentów

```
Facility-Information-Element-Components {ccitt identified-organization etsi(0) 196
                                         facility-information-element-components(3)}
DEFINITIONS ::=
BEGIN
EXPORTS     InvokeIDType, Components;
IMPORTS     OPERATION, ERROR
            FROM Remote-Operation-Notation
                {joint-iso-ccitt remote-operations(4) notation(0)};
Components ::= CHOICE {
                        invokeComp [1] IMPLICIT InvokeComponent,
                        returnResultComp [2] IMPLICIT ReturnResultComponent,
                        returnErrorComp [3] IMPLICIT ReturnErrorComponent,
                        rejectComp [4] IMPLICIT RejectComponent}
InvokeComponent ::= SEQUENCE {
                        invokeID InvokeIDType,
                        linked-ID [0] IMPLICIT InvokeIDType OPTIONAL,
                        operation-value OPERATION,
                        argument ANY DEFINED BY operation-value OPTIONAL}
                        -- ANY is filled by the single ASN.1 data type following the
                        -- keyword ARGUMENT in the type definition of a particular
                        -- operation.
InvokeIDType ::= INTEGER (-32768..32767)
ReturnResultComponent ::= SEQUENCE {
                        invokeID InvokeIDType,
                        SEQUENCE {
                                operation-value OPERATION,
                                result ANY DEFINED BY operation-value
                                -- ANY is filled by the single ASN.1 data type following
                                -- the keyword RESULT in the type definition of
                                -- a particular operation.
                        }OPTIONAL}
ReturnErrorComponent ::= SEQUENCE {
                        invokeID InvokeIDType,
                        error-value ERROR,
                        parameter ANY DEFINED BY error-value OPTIONAL}
                        -- ANY is filled by the single ASN.1 data type following the
                        -- keyword PARAMETER in the type definition of a particular
                        -- error.
RejectComponent ::= SEQUENCE {
                        invokeID CHOICE {
                                InvokeIDType,
                                NULL},
                        problem CHOICE {
                                [0] IMPLICIT GeneralProblem,
                                [1] IMPLICIT InvokeProblem,
                                [2] IMPLICIT ReturnResultProblem
                                [3] IMPLICIT ReturnErrorProblem}}
```

cd. tabeli 3

Struktura komponentów

```
GeneralProblem ::= INTEGER {        -- ROSE-provider detected
                    unrecognizedComponent (0),
                    mistypedComponent (1),
                    badlyStructuredComponent (2)}
InvokeProblem ::= INTEGER {         -- ROSE-user detected
                                    -- supplementary entity
                    duplicateInvocation (0), unrecognizedOperation (1),
                    mistypedArgument (2), resourceLimitation (3), initiatorReleasing (4),
                    unrecognizedLinkedID (5), linkedResponseUnexpected (6),
                    unexpectedChildOperation (7)}
ReturnResultProblem ::= INTEGER {   -- ROSE-user detected
                    unrecognizedInvocation (0),
                    resultResponseUnexpected (1),
                    mistypedResult (2)}
ReturnErrorProblem ::= INTEGER {    -- ROSE-user detected
                    unrecognizedInvocation (0), errorResponseUnexpected (1),
                    unrecognizedError (2), unexpectedError (3),
                    mistypedParameter (4)}
END -- of Facility-Information-Element-Components
```

przy czym pola linked-ID i argument są opcjonalne. W polach invokeID oraz linked-ID przesyłane są liczby całkowite z przedziału od -32768 do 32767 (16-bitowe). Pole operation-value zawiera wartość zwracaną przez opisaną dalej makronotację OPERATION. Wartość ta może być typu INTEGER lub OBJECT IDENTIFIER. Struktura pola argument jest zależna od wartości przesyłanej w polu operation-value i odpowiada ona typowi zdefiniowanemu po słowie ARGUMENT w odpowiednim odwołaniu do makronotacji OPERATION znajdującym się w module ASN.1 określającym typy danych dla poszczególnych operacji konkretnej usługi dodatkowej.

Znaczenie poszczególnych pól komponentu nie odgrywa żadnej roli w opisie struktury przesyłanych danych, nie mniej wyjaśnimy krótko, że invokeID jest identyfikatorem inicjowanej operacji umożliwiającym skojarzenie żądania z odpowiedzią, linked_ID jest identyfikatorem wcześniej zainicjowanej operacji, do której nawiązuje aktualnie inicjowana operacja (bardzo rzadko używany), operation-value określa rodzaj operacji, natomiast argument (często najbardziej rozbudowane pole) zawiera parametry inicjowanej operacji.

Komponent Return result tworzą pola:

- invokeID,
- operation-value,
- result.

przy czym sekwencja operation-value, result jest opcjonalna. Pola invokeID i operation_value mają taką samą postać i znaczenie, jak w komponencie Invoke. Struktura pola result, podobnie jak dla pola argument komponentu Invoke, jest zależna od wartości przesyłanej w polu operation-value i odpowiada ona typowi zdefiniowanemu po słowie RESULT w odpowied-

nim odwołaniu do makronotacji OPERATION znajdującym się w module ASN.1 określającym typy danych dla poszczególnych operacji konkretnej usługi dodatkowej.

Komponent Return error utworzony jest z pól:

- invokeID,
- error-value,
- parameter.

przy czym pole parameter jest opcjonalne (w praktyce rzadko kiedy występuje). Pole invokeID ma taką samą postać i znaczenie jak w poprzednio omówionych komponentach. Pole error-value zawiera wartość zwracaną przez opisaną dalej makronotację ERROR. Wartość ta może być typu INTEGER lub OBJECT IDENTIFIER. Struktura pola parameter jest zależna od wartości przesyłanej w polu error-value i odpowiada ona typowi zdefiniowanemu po słowie PARAMETER w odpowiednim odwołaniu do makronotacji ERROR znajdującym się w module ASN.1 określającym typy danych dla poszczególnych błędów konkretnej usługi dodatkowej. Pole error-value określa rodzaj błędu, natomiast pole parameter zawiera dodatkowe dane dotyczące błędu.

Komponent Reject zawiera pola:

- invokeID,
- problem.

Pole invokeID może zawierać wartość typu INTEGER (podobnie jak w poprzednich komponentach) lub wartość typu NULL (gdy nie jest możliwe zidentyfikowanie operacji, której dotyczy komponent). W polu problem znajduje się wartość jednego z czterech typów etykietowanych, dla których typem bazowym jest INTEGER etykietowany niejawnie. Etykiety w poszczególnych typach etykietowanych odpowiadają różnym grupom problemów (błędów). Każdy problem reprezentowany jest przez wartość liczbową o identyfikatorze kojarzącym się z rodzajem problemu.

6. Definiowanie struktury danych sygnalizacyjnych dla konkretnych usług dodatkowych

Jak już wspomniano, dla każdej usługi dodatkowej określony jest zestaw elementarnych operacji. Dla każdej z tych operacji struktura danych sygnalizacyjnych definiowana jest za pomocą makronotacji OPERATION i ERROR. Makrodefinicje dla powyższych makronotacji określone zostały w [13] (można je również znaleźć w [9,11]).
Obie makronotacje zwracają wartość typu

```
CHOICE {
        localValue    INTEGER,
        globalValue   OBJECT IDENTIFIER }
```

czyli wartość typu INTEGER lub OBJECT IDENTIFIER o identyfikatorach odpowiednio „localValue" lub „globalValue" (typy nazwane).

Każdej operacji odpowiada w standardach zapis wykorzystujący makronotację OPERATION (notacja nowego typu) mający postać:

Typ_rodzaju_operacji::= OPERATION

ARGUMENT *typ_dla_pola_argument_komponentu_Invoke*
RESULT *typ_dla_pola_result_komponentu_Return_reslt*
ERRORS {*lista_błędów_dla_operacji*}
LINKED {*lista_operacji_pochodnych*}

Wartość zwracana przez makronotację OPERATION jest identyfikatorem operacji i umieszczana jest ona w polu „operation-value" komponentów Invoke i Return result. Nazwa „Typ_rodzaju_operacji" dobierana jest zwykle tak, aby kojarzyła się z funkcją danej operacji.

Zgodnie z wcześniejszymi ustaleniami zapisy po słowach ARGUMENT I RESULT określają typ danych dla pól „argument" i „result" komponentów odpowiednio Invoke i Return result.

Zapis po słowie ERRORS jest listą błędów sygnalizowanych za pomocą komponentu Return error. Elementami tej listy mogą być nazwy wartości pojawiających się w polu „error-value" komponentu Return error, jak również nazwy typów odpowiadających tym wartościom. Lista błędów może zawierać zarówno błędy specyficzne dla danej operacji lub usługi dodatkowej, jak i tzw. błędy ogólnego zastosowania (generally applicable errors) zdefiniowane w [11] (np. notSubscribed, notAvailable, notImplemented).

Zapis po słowie LINKED jest listą operacji pochodnych w stosunku do danej operacji (child operations). Elementami tej listy mogą być nazwy wartości pojawiających się w polu „operation-value" komponentów Invoke i Return result operacji pochodnych, jak również nazwy typów odpowiadających tym wartościom.

Słowa ARGUMENT, RESULT, ERRORS, LINKED i zapisy po nich występujące mogą być pomijane. Brak zapisu rozpoczynającego się od słów ARGUMENT lub RESULT oznacza brak pola „argument" lub „result" w komponencie Invoke lub Return result. Pominięcie zapisu RESULT może też oznaczać, że komponent Return result nie jest dla danej operacji wykorzystywany. Brak zapisu rozpoczynającego się od słowa ERRORS oznacza, że żaden rodzaj błędu nie jest sygnalizowany za pomocą komponentu Return error. Brak zapisu rozpoczynającego się od słowa LINKED oznacza z kolei brak operacji pochodnych (dla aktualnie realizowanych usług dodatkowych brak tego zapisu jest regułą).

Po zapisie odwołującym się do makronotacji OPERATION następują zwykle definicje typów, do których odwołują się zapisy po słowach ARGUMENT i RESULT.

Określenie rodzaju operacji przesyłanego w polu „operation-value" realizowane jest za pomocą zapisu o postaci:

nazwa_wartości Typ_rodzaju_operacji ::= wartość

przy czym nazwa wartości różni się od nazwy typu rodzaju operacji jedynie „rozmiarem" pierwszej litery (nazwa wartości - mała litera, nazwa typu - duża litera).

Poniżej przedstawiono przykład omawianych zapisów dla operacji AOCEChargingUnit realizowanej w ramach usługi Advice of Charge (AOC) [5]. Celem operacji jest przesłanie do abonenta po zakończeniu rozmowy informacji o liczbie naliczonych impulsów oraz opcjonalnie - informacji o sposobie jej naliczania. Pominięte zostały zapisy definiujące typy opcjonalne. Dla omawianej operacji wykorzystuje się tylko komponent Invoke.

```
AOCEChargingUnit ::= OPERATION
                        ARGUMENT CHOICE {
                                chargeNotAvailable NULL,
                                AOCEChargingUnitInfo}
AOCEChargingUnitInfo ::= SEQUENCE {
                    CHOICE {
                        specificChargingUnits SEQUENCE {
                                recordedUnitsList     [1] RecordedUnitsList,
                                aOCEBillingId         [2] AOCEBillingId OPTIONAL},
                        freeOfCharge  [1] NULL},
                    chargingAssociation  ChargingAssociation OPTIONAL}

RecordedUnitsList ::= SEQUENCE SIZE (1..32) OF RecordedUnits
RecordedUnits ::= SEQUENCE {
                    CHOICE {
                        recordedNumberOfUnits     NumberOfUnits,
                        notAvailable              NULL},
                    recordedTypeOfUnits       TypeOfUnit OPTIONAL }

NumberOfUnits ::= INTEGER (0..16777215)

aOCEChargingUnit AOCEChargingUnit ::= 36
```

Pole argument komponentu Invoke zawiera normalnie sekwencję *AOCEChargingUnitInfo*, której jedynym składnikiem w najprostszym przypadku (odrzucenie typów opcjonalnych i NULL) jest sekwencja *specificChargingUnits*. Jedynym obowiązkowym składnikiem tej sekwencji jest typ etykietowany *[1] RecordedUnitsList* będący sekwencją maksimum 32 typów *RecordedUnits*. Typ *RecordedUnits* jest znowu sekwencją, którą tworzy typ INTEGER reprezentujący liczbę naliczonych impulsów oraz (opcjonalnie) typ *TypeOfUnit* określający numer taryfy.

Rodzaj operacji określony jest przez wartość typu INTEGER równą 36. Typ wartości wynika ze sposobu jej zapisu. Definicja rodzaju operacji może w omawianym przypadku mieć alternatywną postać:

```
aOCEChargingUnit AOCEChargingUnit ::= localValue 36
```

W zapisie tym posłużono się wartością nazwaną.

W wielu przypadkach rodzaj operacji określony jest przez wartość typu OBJECT IDENTIFIER. Definicja tej wartości ma wówczas bardziej złożoną postać. Przykładowo, dla operacji CCBSRequest [6] może przedstawiać się ona następująco:

```
cCBSOID OBJECT IDENTIFIER   ::=   {ccitt identified-organization etsi (0) 359
                                   operations-and-errors (1) }
cCBSRequest CCBSRequest     ::=   globalValue   {cCBSOID 2}
```

Każdemu typowi występującemu w liście błędów makronotacji OPERATION (typ ten może być wykorzystywany przez wiele operacji związanych z daną usługą dodatkową) odpowiada definicja odwołująca się do makronotacji ERROR. Ma ona w ogólnym przypadku postać jak niżej.

```
Typ_błędu ::= ERROR
                PARAMETER typ_dla_pola_parameter_komponentu_Return_error
```

Dla aktualnie realizowanych usług dodatkowych pole „parameter" nie występuje w komponentach Return error. W związku z powyższym zapis rozpoczynający się od słowa PARAMETER jest pomijany. Wartości określające rodzaje błędów przesyłane w polu „error-value" komponentu Return error definiowane są na podobnych zasadach co wartości określające rodzaje operacji. Przykładowy zapis dotyczący określonego rodzaju błędu może mieć poniższą postać [5].

```
NoChargingInfoAvailable ::= ERROR
noChargingInfoAvailable NoChargingInfoAvailable ::= 26
```

Tak więc dla błędu „NoChargingInfoAvailable" w polu „error-value" komponentu Return result przesyłana jest wartość typu INTEGER równa 26.

7. Przykład kodowania komponentu przesyłanego w elemencie informacyjnym FACILITY

W tabeli 4 zaprezentowano postać transferową komponentu Invoke dla wcześniej omówionej operacji „AOCEChargingUnit" zarejestrowanego w łączu ISDN sieci T.P. S.A. Komponent ten nie zawiera elementów opcjonalnych ani typów NULL.

W module ASN.1 opisującym strukturę powyższego komponentu przyjęto domyślnie niejawny sposób etykietowania typów etykietowanych (IMPLICIT TAGS w nagłówku modułu).

Postać transferowa omawianego komponentu wynika z wcześniej podanych zapisów ASN.1 i z reguł kodowania określonych w [3]. Oktety 1 i 2 stanowią nagłówek identyfikujący rodzaj komponentu. Znaczenie dalszych grup oktetów jest następujące:

- oktety 3 ÷ 6 - pole „invokeID",
- oktety 7 ÷ 9 - pole „operation-value",
- oktety 10 ÷ 20 - pole „argument".

Warto zwrócić uwagę, że ze względu na niejawny sposób etykietowania typów etykietowanych nie są przesyłane etykiety typów InvokeComponent oraz RecordedUnitsList. Zgodnie z wcześniejszymi definicjami, pierwszy typ jest w istocie inaczej nazwanym typem SEQUENCE, natomiast drugi - typem SEQUENCE OF. Zatem w razie jawnego etykietowania należałoby dla obu wymienionych typów dodatkowo przesłać etykietę typu SEQUENCE lub SEQUENCE OF.

8. Uwagi końcowe

Głównym celem notacji ASN.1 jest uniezależnienie opisu struktury danych od reguł kodowania. Stosowanie abstrakcyjnej syntaktyki daje też pewne dodatkowe korzyści. Po pierwsze, zapis ASN.1 jest przejrzysty dla danych o złożonej strukurze - gdy występuje alternatywna postać pól lub dopuszczalne jest pomijanie niektórych pól. W przypadku niektórych usług dodatkowych trudno sobie wręcz wyobrazić inny sposób opisywania danych sygnalizacyjnych. Omawiana cecha sprawia, że sensowne może okazać się stosowanie notacji ASN.1 również do opisu danych przesyłanych w niższych warstwach modelu OSI/ISO. Po drugie, notacja ASN.1 wymusza uporządkowane i jednolite reguły kodowania danych, co z kolei ułatwia ich programową analizę. Z powyższych względów w standardach ETSI dla usług dodatkowych ISDN całkowicie zrezygnowano z tradycyjnych metod opisu struktury danych sygnalizacyjnych.

Tabela 4

Budowa komponentu Invoke dla operacji AOCEChargingUnit

Nr oktetu	Pole oktetu	Znaczenie	Komentarz
1	10------	klasa etykiety	: context-specific
	--1-----	P/C	: typ złożony
	---00001	numer etykiety	: [1] InvokeComponent
2	0-------	format długości	: jeden oktet
	-0010010	długość zawartości	: 18 oktetów
3	00------	klasa etykiety	: universal
	--0-----	P/C	: typ prosty
	---00010	numer etykiety	: INTEGER - invokeID
4	0-------	format długości	: jeden oktet
	-0000010	długość zawartości	: 2 oktety
5	00000010	zawartość - oktet 1	: invokeID - high
6	10001111	zawartość - oktet 2	: invokeID - low
7	00------	klasa etykiety	: universal
	--0-----	P/C	: typ prosty
	---00010	numer etykiety	: INTEGER - operation-value
8	0-------	format długości	: jeden oktet
	-0000001	długość zawartości	: 1 oktet
9	00100100	zawartość	: aOCE ChargingUnit
10	00------	klasa etykiety	: universal
	--1-----	P/C	: typ złożony
	---10000	numer etykiety	: SEQUENCE
			: - AOCEChargingUnitInfo
11	0-------	format długości	: jeden oktet
	-0001001	długość zawartości	: 9 oktetów
12	00------	klasa etykiety	: universal
	--1-----	P/C	: typ złożony
	---10000	numer etykiety	: SEQUENCE
			: - specificChargingUnits
13	0-------	format długości	: jeden oktet
	-0000111	długość zawartości	: 7 oktetów
14	10------	klasa etykiety	: context-specific
	--1-----	P/C	: typ złożony
	---00001	numer etykiety	: [1] RecordedUnitsList
15	0-------	format długości	: jeden oktet
	-0000101	długość zawartości	: 5 oktetów
16	00------	klasa etykiety	: universal
	--1-----	P/C	: typ złożony
	---10000	numer etykiety	: SEQUENCE - RecordedUnits
17	0-------	format długości	: jeden oktet
	-0000011	długość zawartości	: 3 oktety
18	00------	klasa etykiety	: universal
	--0-----	P/C	: typ prosty
	---00010	numer etykiety	: INTEGER - NumberOfUnits
19	0-------	format długości	: jeden oktet
	-0000001	długość zawartości	: 1 oktet
20	00000010	zawartość	: liczba impulsów = 2

Literatura

1. CCITT (ITU-T): Recommendation X.200. Reference Model of Open Systems Interconnection for CCITT Applications. 1980.
2. CCITT (ITU-T): Recommendation X.208. Specification of Abstract Syntax Notation One (ASN.1). 1984.
3. CCITT (ITU-T): Recommendation X.209. Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1). 1984.
4. ETSI: ETS 300 286. ISDN; User-to-User Signalling (UUS) supplementary service. 1996.
5. ETSI: ETS 300 182. ISDN; Advice-of-Charge (AOC) supplementary service. 1993.
6. ETSI: ETS 300 359. ISDN; Completion of Calls to Busy Subscriber (CCBS) supplementary service. 1995.
7. Kabaciński W.: Standaryzacja w sieciach ISDN. Wydawnictwo Politechniki Poznańskiej, Poznań 1996.
8. Kościelnik D.: ISDN cyfrowe sieci zintegrowane usługowo. WKŁ, Warszawa 1996.
9. ETSI: ETS 300 102. ISDN; User-network interface layer 3, Specifications for basic call control. 1990.
10. ETSI: ETS 300 351. ETSI object identifier tree; Rules and registration procedures. 1994.
11. ETSI: ETS 300 196. ISDN; Generic functional protocol for the support of supplementary services. 1993.
12. Brzeziński K.M.: Istota sieci ISDN. Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 1999.
13. CCITT (ITU-T): Recommendation X.219. Remote Operations: Model, Notation and Service Definition. 1988.

Recenzent: Dr hab.inż.Krystyna Macek-Kamińska

Abstract

Defining the structure (format) of protocol data units (PDU) is an important part of communication protocol specification. Traditional methods of PDU format defining depend usually on defining the meaning of subsequent octets or bits creating this PDU (like fig.1). A drawback of these methods is that they simultaneously define types of data contained in the PDU and rules of coding the data. In some situations it is profitable to define PDU structure

only on the level of data types without defining the transfer syntax of the data. The need of such an approach appears, for example, in specifications of application layer protocols.

Presentation of transfered data structure on the level of data types is called abstract syntax (syntax abstracting from coding). The ASN.1 notation (Abstract Syntax Notation One) described in X.208 standard is an example of such a syntax. For this notation the X.209 standard, that presents the rules of coding particular types of data, has been developed. The ASN.1 notation and the rules of coding defined in X.209 are adapted in ISDN networks and other telecommunications systems, where they are used in descriptions of signalling data related to supplementary services. This paper presents a synthetic conception of problems, that concern application of ASN.1 notation to define such a kind of data.

Basic rules of supplementary services realization are presented. Then, the ASN.1 notation and the rules of coding defined in X.209 are characterized. Kinds of PDU (components) used in supplementary services and their general structure are next described using ASN.1 notation (table 3). The method of defining formats of signalling data for particular supplementary services is discussed. An example of real component encoding is depicted in the end (table 4).