

Távközlési informatika

Dr. Beinschróth József

Formális leírás – SDL

- Általános jellemzők:
 - Telekommunikációs rendszerek leírására és specifikálására szolgál.
 - Alkalmas a specifikálandó **rendszer szerkezetének és viselkedésének** együttes leírására. A rendszer több egymással és a rendszerkörnyezettel **kommunikáló automatából** áll, ahol a kommunikáció jelekkel történik a FIFO elven működő **csatornákon és jelutakon**.
 - A teljes rendszer viselkedését az egyes **automaták viselkedésének összessége** határozza meg. Funkció szerint **hierarchiába szervezve** az egymással kommunikáló automatakat, a leírást a **rendszer, blokk és processz** hierarchiaszintek jellemzik. A **processz** maga a **véges állapotú automata**, a rendszer működését a processz diagrammokkal adjuk meg. A processzek a rendszer keletkezésekor vagy egy másik processz hatására keletkezhetnek, egyszerre több példányban is. Minden egyes **processz egyetlen bemeneti** sorral rendelkezik. Bonyolultabb esetekben a rendszerblokkok alrendszerekre bomlanak, ezek újabb blokkokat és alrendszereket tartalmazhatnak.

Formális leírás – SDL

- A rendszer és a környezet együttműködését vizsgálja és leírja a rendszer belső struktúráját.
- A rendszer működése a rendszerben levő **processek összetett viselkedésével** írható le. **Process: véges automata, autonom és paralell más processekkel.** A processek képesek **együttműködni**: aszinkron módon végrehajtott diszkrét üzeneteket küldhetnek egymásnak (**signal**). A process küldhet/kaphat jelet a környezet felé/felől is.
- A process **determinisztikus viselkedésű**: külső gerjesztésre a leírásban definiált módon válaszol.
- Minden processnek **egyedi azonosítója** van. A signalok mindig tartalmazzák a küldő és a címzett process azonosítóját és az adatot.
- A processnek saját **változói** vannak, a process csak a saját változóit módosíthatja.
- A process tartalmaz egy **végtelen sort** a bejövő jelek sorbaállítására.
- A processnek állapotai vannak: **wait** vagy két állapot közötti átmenet (**transition**)
- A process mindig blokkokban fordul elő.

Formális leírás – SDL

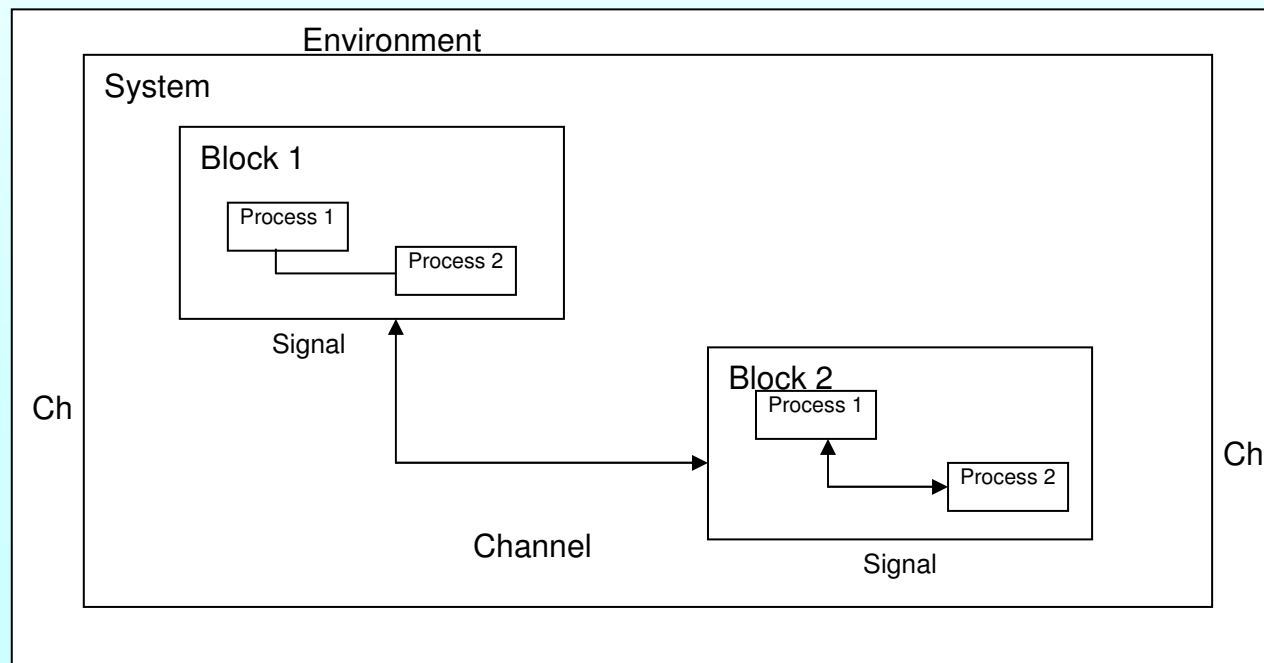
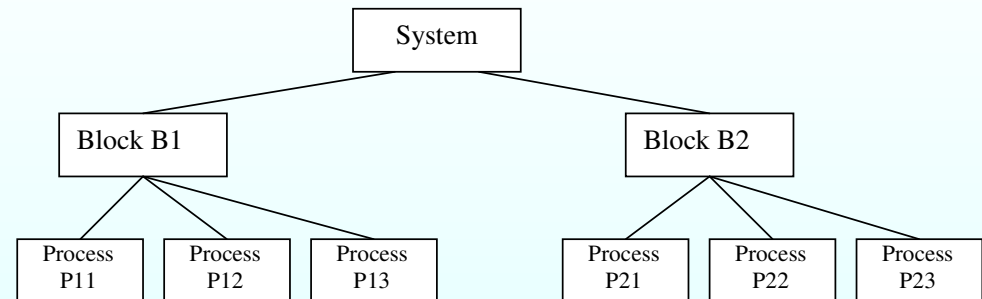
- **A rendszer** egy vagy több **blokkot** tartalmaz. A blokkokat **csatornák kötik össze** egymással ill. a környezettel. A blokkok alblokkokra oszthatók. A rendszer olyan speciális blokk, amelyhez nem kapcsolódik csatorna. Ismételt blokk szétoztás block-tree-hez vezet: a felső blokk a rendszer, a legalsó blokkok, amelyek már nem oszthatók tovább (leaf block) tartalmazzák a processeket. Ezek feletti blokkokban nincsenek processek.
- Első kiadás: 1980, azóta javított kiadások jelentek meg.
- 2002-ben megjelent változat objektum orientált tulajdonságokkal rendelkezik.
- Az ITU (International Telecommunication Union) Z.100, Z.105, Z.107 és Z109 ajánlásai alapján fejlesztették. A **grafikus és szöveges változatát** is folyamatosan fejlesztik, 1996-2000 között jelentősen kiterjedt az alkalmazása, már nemcsak a távközlés alkalmazza, legújabb verziója az SDL2000 alkalmas tetszőleges valós idejű, interaktív elosztott rendszer formális leírására. Jellemzi az objektum-orientáltság, a távoli eljáráshívás, a tesztelhetőség, hatékonyság.

Formális leírás – SDL

- Az SDL-t két formában valósították meg:
 - **SDL/PR** a magas szintű nyelvekhez hasonló szintaktikával rendelkezik.
 - **SDL/GR** a leíráshoz grafikus szimbólumokat használ
- Az SDL célja a rendszerek formális leírása. Az SDL a rendszert leíró adatstruktúrát és a rendszer viselkedését egyaránt tartalmazza.

Formális leírás – SDL

- Leírás SDL szimbólumokkal:



Formális leírás – SDL

- Definíciók:

- **Rendszer (system):**

- Amit az SDL leírás specifikál, egy a környezetével kommunikáló absztrakt gép. (Pl.: egy OSI rendszer, egy protokoll, stb)
 - A rendszer blokkokra osztható. A blokkok egymás között valamint a környezettel (environment) csatornákon (channel) keresztül kommunikálnak.

- **Blokk (block):**

- A rendszer egy része amely különböző szempontok szerint lett kialakítva (Pl.: megértés, leírás, fejlesztés ...).
 - A blokk a blokk diagram segítségével kerül leírásra.
 - A blokk diagram tartalmazza.:
 - Blokk neve
 - Jelek (signal) leírása
 - Jel utak (signal route) leírása
 - Csatorna jelút leírása
 - Processzek leírása

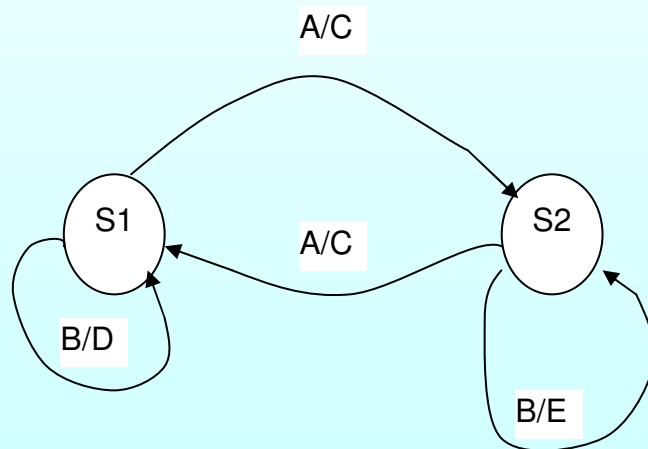
Formális leírás – SDL

– **Processz (process):**

- Determinisztikus működésű kiterjesztett véges automata.
- A processzek leírás az alábbi elemeket tartalmazza:
 - Processz azonosító (PID)
 - Paraméterek leírása
 - Változók leírása
 - Időzítők leírása
 - Procedúrák leírása
 - Folyamat gráf

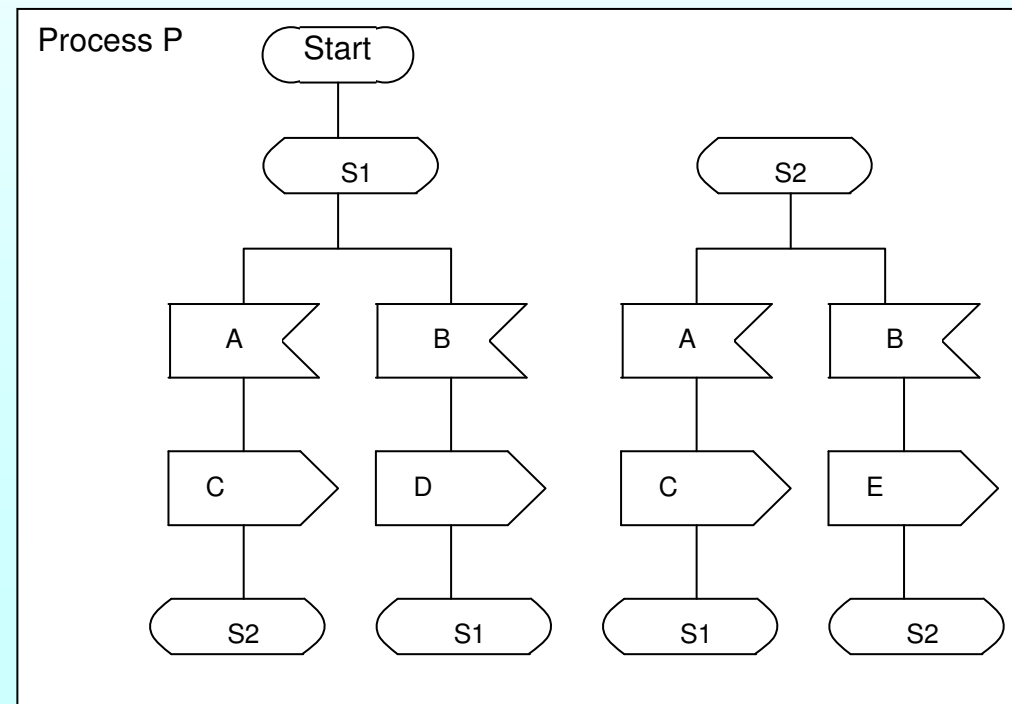
Formális leírás – SDL

- Leírás állapot-átmeneti gráffal



Input={A,B}
Output={C,D,E}

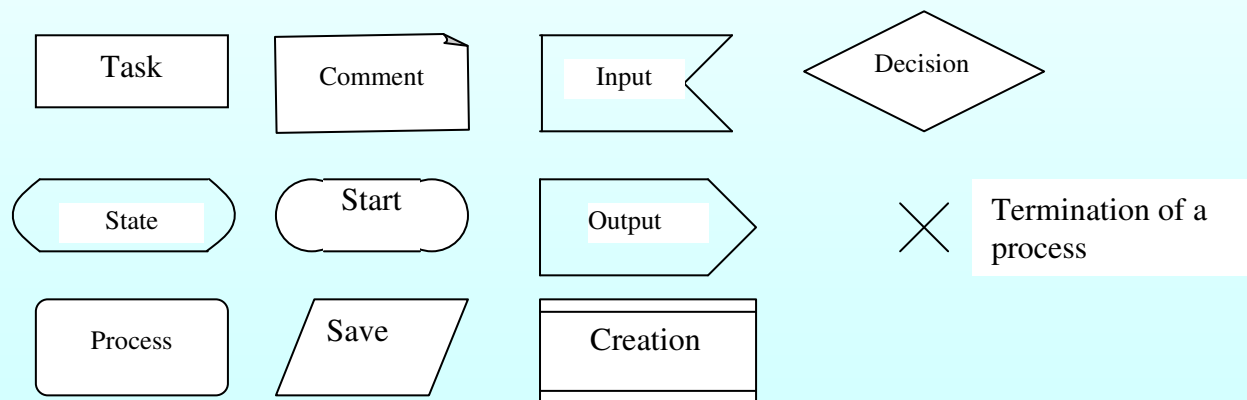
- Leírás SDL/GR formátumban



Formális leírás – SDL

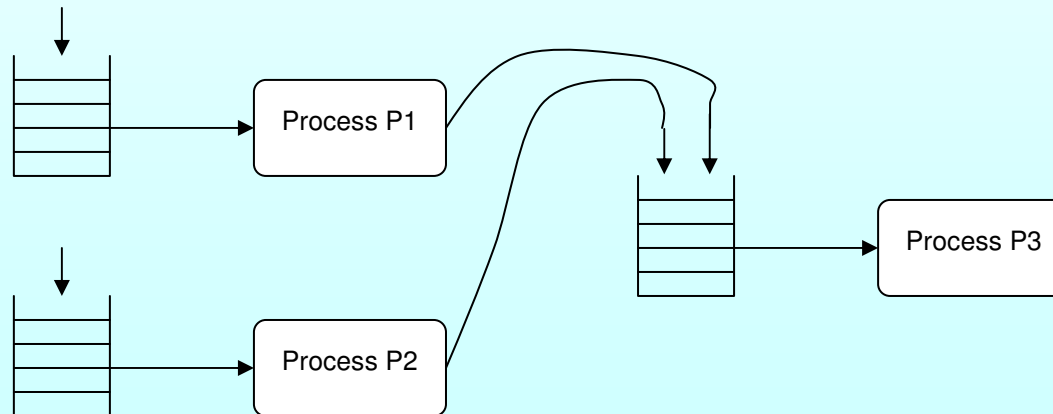
- Az SDL/GR szimbólumok:

a szimbólumkészlet egy része



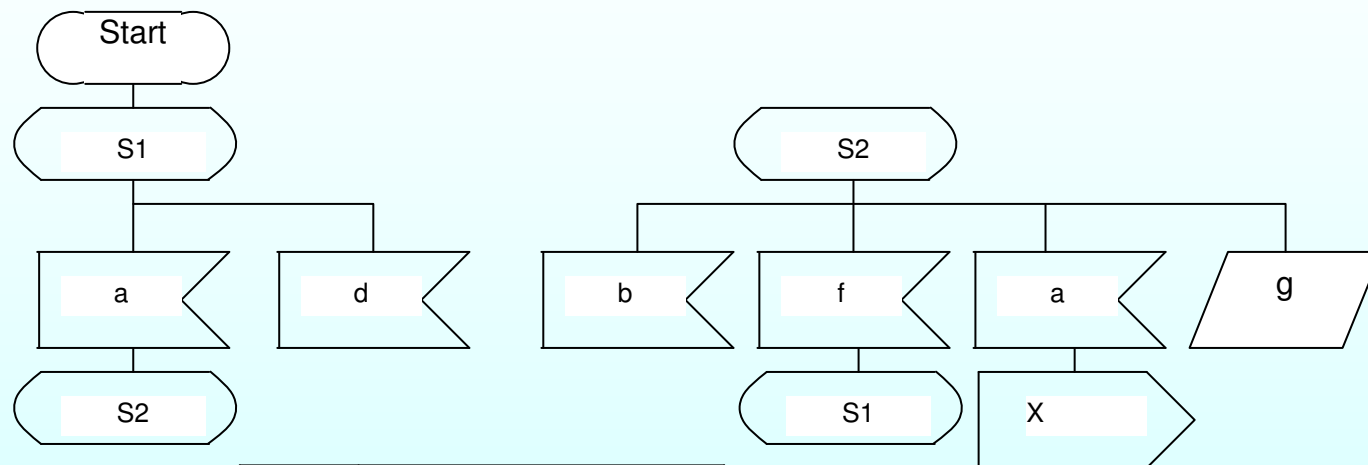
Formális leírás – SDL

- Processzek kommunikációja
 - A processzek bemenetére érkező jelzések, amelyek más processzekből vagy a környezetből érkeznek, **egy jelzés sorban** várnak a feldolgozásra.



Formális leírás – SDL

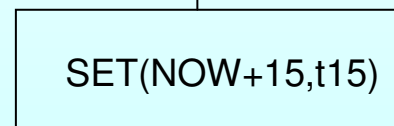
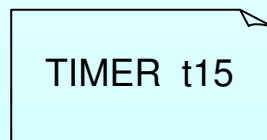
- Példa processek kommunikációjára:



Jelek	Feldolgozás
a	S2-be vált
d	eldob
a	Kiad:X
b	- (semmi változás)
f	S1-be vált
f	eldob
a	S2-be vált
g	eltárol (save)

Formális leírás – SDL

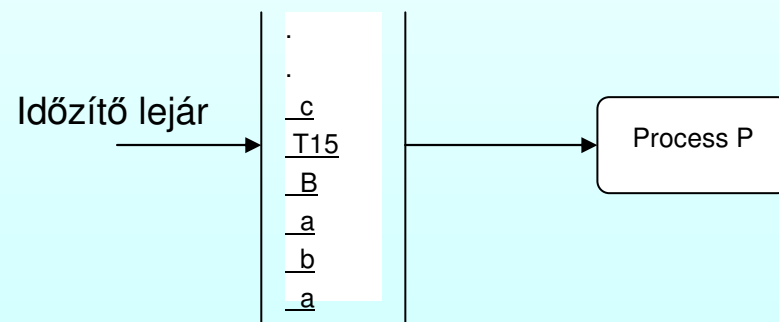
- Időzítők kezelése
 - Az SDL-ben lehet időzítőket definiálni. Az időzítő elindítása egy Task szimbólummal történik. Ha az időzítő lejárt egy az időzítő nevével megegyező nevű jelet helyez el a processz a jelzés sorba. Az időzítés értéke relatív idővel van megadva amit az SDL nem definiál.



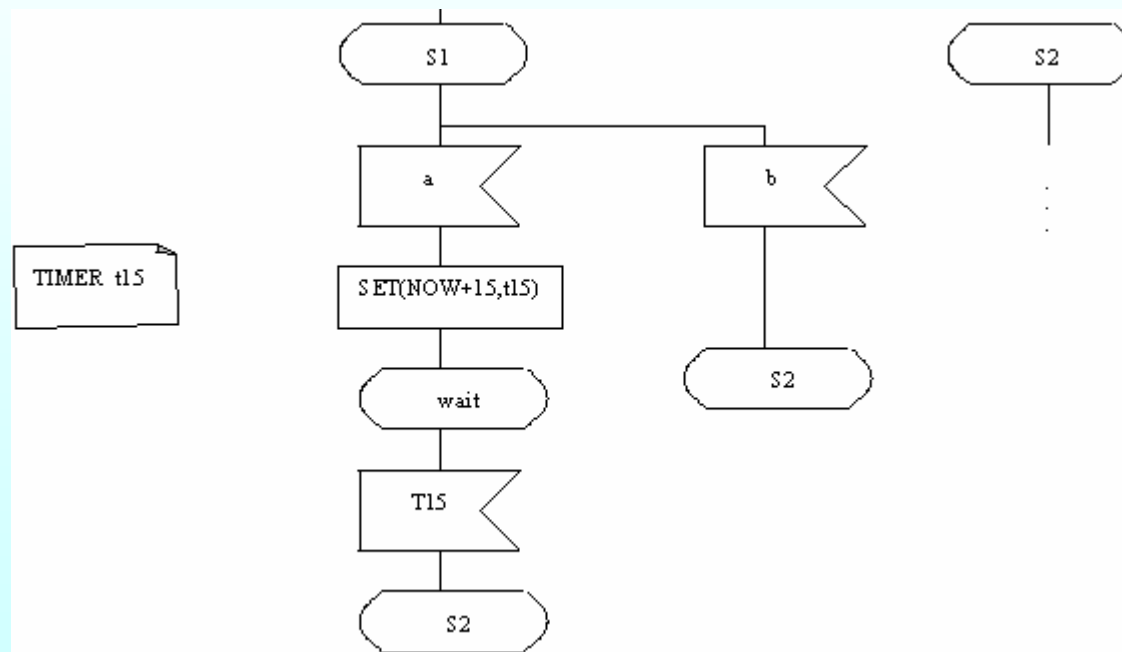
Timer definiálása és elindítása 15 egységnyi relatív időzítéssel

Formális leírás – SDL

Az ábra egy időzítő kezelésére vonatkozó példát tartalmaz.
A P processz bemenő jelzései: a, b, c



Formális leírás – SDL



Példa időzítő kezelésére

Formális leírás – SDL

Döntés (decision)

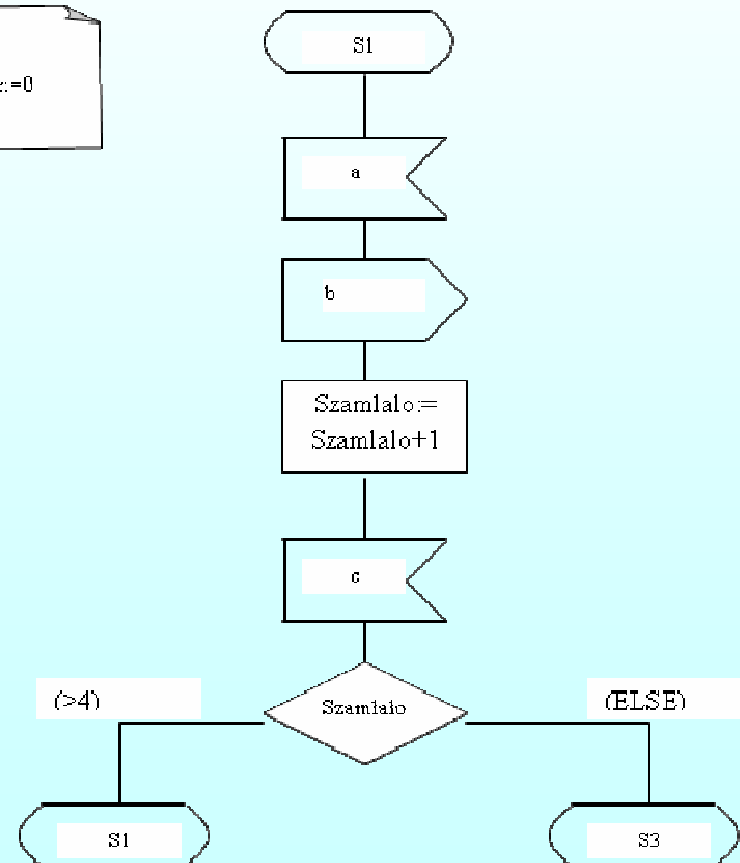
Tipikusan a kiterjesztett véges automata paramétereit és változóit használják a döntésekhez.

DCL
Szamlalo Integer:=0

Példa a döntés alkalmazására az SDL-ben:

Az ábrán egy számláló típusú változó deklarálása történik, amit a DCL kulcsszó jelez.

A számláló állapotától függően történik meg a döntés.



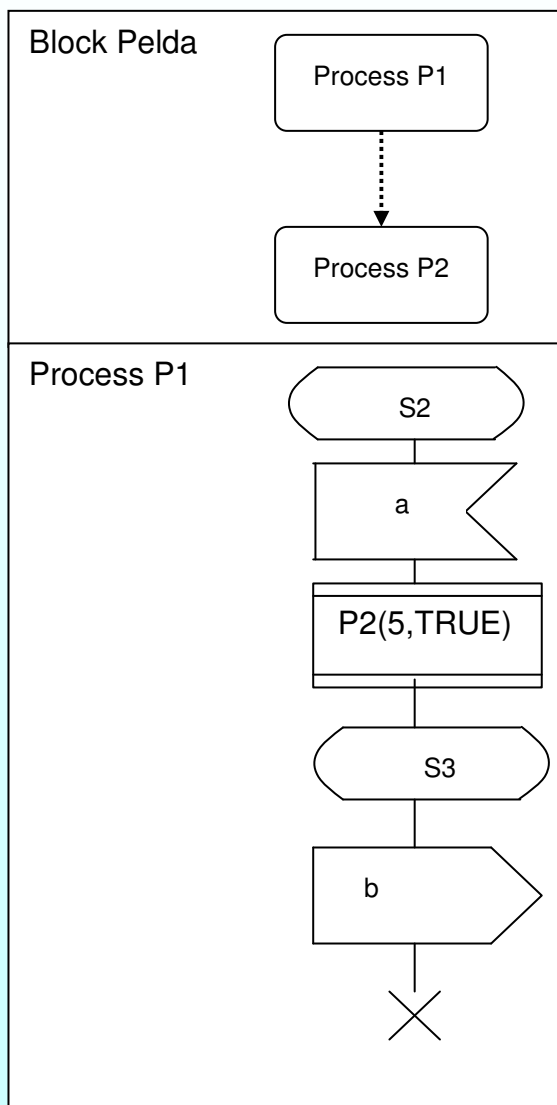
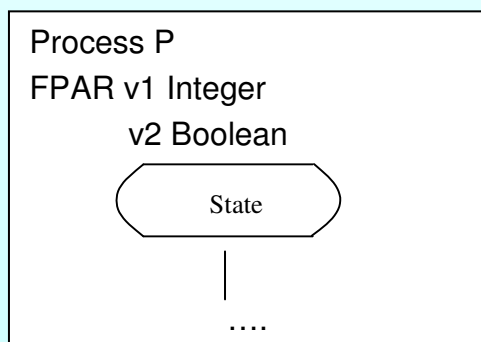
Formális leírás – SDL

Dinamikus processz indítás:

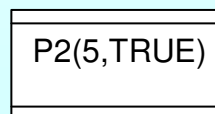
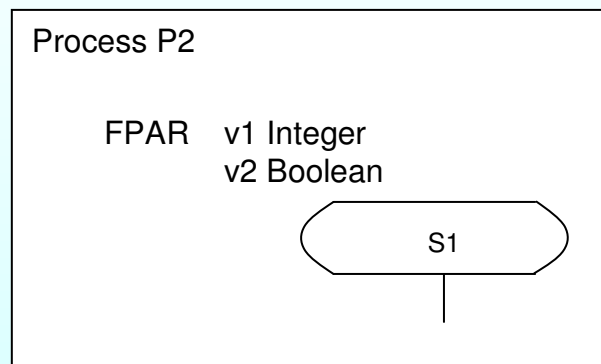
Egy processz indítása történhet paraméterekkel vagy anélkül.

A processz paramétereit az FPAR kulcsszóval lehet megadni.

Pl.:



A Pelda nevű blokkban a P1 processz dinamikusan indítja a P2 processzt. (Szaggatott vonal)



Process leállítás

Dinamikus processz indítás v1=5, v2=TRUE paraméterekkel.

Formális leírás – ASN.1

- **Objektum orientált tervezés áttekintése**
 - A valós világ leírása a természetes gondolkodást közelítő módon.
 - Az Objektum Orientált (OO) gondolkodás szerint a **leírandó rendszer az objektumok (object) halmaza**. Az objektumok a belső állapotukkal és viselkedésükkel jellemezhetők. Egy objektumra a környezetében lévő objektumok hatnak (**az objektumok egymásnak üzenetet küldenek**) aminek hatására **belső állapotuk megváltozhat**.
 - OO tervezés a gyakorlatban:
 - programozás (OOP)
 - adatbázisok
 - modellezés (gazdasági stb.)

Formális leírás – ASN.1

- **Objektum orientált tulajdonságok bemutatása az OOP-on keresztül:**
 - Egy OO program együttműködő **objektumok** halmaza.
 - Az objektumok a környezetüktől **jól elkülöníthető, viszonylag független** összetevők, amelyeknek **saját viselkedésük, működésük** és lehetőleg elrejtett **belső állapotuk** van. Az egyes objektumok egymással **kapcsolatba** léphetnek, aminek hatására **belső állapotuk megváltozhat**.
 - Minden objektum valamilyen **osztályba (class)** tartozik. Az **osztályok absztrakt adattípusok**. Minden **objektum** valamely absztrakt **adattípus egyede** (instance).
 - Az osztályok definiálják az egyes objektumok belső állapotát leíró attributomot (**adatszerkezet**) és a rajtuk végezhető műveleteket (**method**). Az egyes egyedek csak az állapotukat meghatározó adatszerkezet **tényleges értékeiben térnek el** egymástól, a műveletekkel definiált **viselkedésük közös**.
 - Az egyes osztályokat az **öröklés (inheritance)** hierarchiába rendezi. Az öröklés az az eljárás amellyel egy osztály felhasználja a **hierarchiában felette álló** osztályban lévő adatszerkezetet (állapotot) és viselkedést (módszereket). Így a közös elemet elég egyszer, a hierarchia megfelelő szintjén definiálni.
 - Az a.), b.) és c.) tulajdonságokkal rendelkező programnyelvek objektum alapúak (object based). Az a.), b.), c.) és d.) tulajdonságokkal rendelkező programnyelvek: OOP. Pl. C++. Java.

Formális leírás – ASN.1

• OOP tulajdonságok:

– Öröklés (inheritance):

- Egy már definiált osztályból származtatható egy másik osztály úgy, hogy a leszármazott osztály (alosztály) ugyanúgy tartalmazza az őosztály (szülőosztály, szuperosztály) **összes attribútumát és metódusát**. Az alosztályban bővíthető az adatszerkezet és új metódusok definiálhatók.
- Egy ősből több leszármazott osztály is létrehozható. Ezek a kapcsolatok az osztályok egy hierarchiáját adják amelyek irányított gráffal jellemezhetők. Ha a nyelv csak egy őst enged akkor a hierarchia fastruktúrát képez.

– Egységbezárás (encapsulation):

- Az attribútumok és metódusok egységes kezelése és a külvilágtól való **elrejtése**. Ha **külső hatás** éri az objektumot (üzenet érkezik) akkor az objektum metódusai megváltoztatják az objektum adatait, **megváltozik az objektum állapota**. Az objektum mindig **ismeri a saját adatait**, azoknak megváltoztatását csak az objektum saját metódusaival lehet (szabad). Azonos attribútumokkal és metódusokkal rendelkező objektumok halmaza az objektum osztály (class). Először az osztályokat hozzuk létre, utána az egyedeket (objektumok).

– Többértékűség (polymorphism):

- Ha származtatunk egy alosztályt akkor az **örökli az ő metódusait**. Ezeket a metódusokat **megváltoztathatjuk, de nevüket meghagyhatjuk**. Így ugyanazon nevű metódusnak más-más osztályban más lesz a viselkedése. Ha nem akarjuk az alosztályban megváltoztatni a metódust, akkor nem kell az alosztály deklarációjában szerepeltetni.

Formális leírás – ASN.1

- **(ASN.1) Abstract Syntax Notation One**

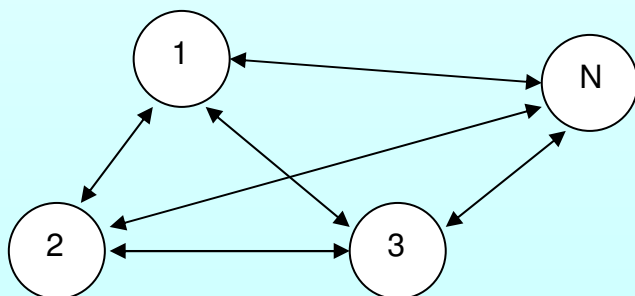
Az OSI Presentation Layerébe tartozik.

- A réteg az átvitelre kerülő adatok **szintaxisával foglalkozik**. A protokollokban továbbított adatok **formális leírására** szolgál. Az OSI keretében az ISO ill. ITU-T definiált egy absztrakt adatstruktúrát az ASN1-et. Az ASN1-ben az adatok ábrázolására objektum orientált módon történik.
- Az adatok ábrázolására alkalmazott megoldások:
 - Programozási nyelvek
 - Fizikai hordozók (diszk stb.)
 - Kódtáblák (ASCII, EBCDIC ...)
- Nagyon **sokféle** alkalmazás van, ezeknek szinte mindegyike más-más ábrázolási módot, adatformátumot használ. Ezek egymással nem kompatibilisek (pl. az adatbázis mezők mérete, elválasztása különbözik stb.) Hogy az alkalmazások egymással kommunikálhassanak nagyon sok konvertáló programra lenne szükség.

Formális leírás – ASN.1

Különféleképpen ábrázolt
adatok kezelése:

Valamennyi adattípus közötti konverzió:

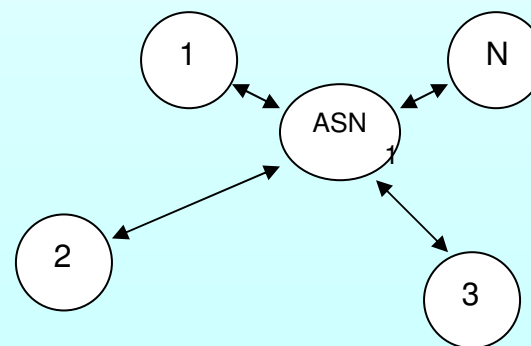


$N(N-1)$ konverzió szükséges
Pl.: $N=4$ esetén 12.

Közös és szabványosított formátum (ASN.1):

Minden alkalmazás csak két (oda-vissza) konvertáló programot igényel.

A konverzió teljesen automatikusan valósulhat meg. Még jobb, ha az adatátvitelt egy absztrakt, alkalmazástól független szintaktika szerint írjuk le, mert ez esetben az alkalmazás megfelelő moduljai alkalmas compilerrel gyakorlatilag automatikusan állíthatók elő.



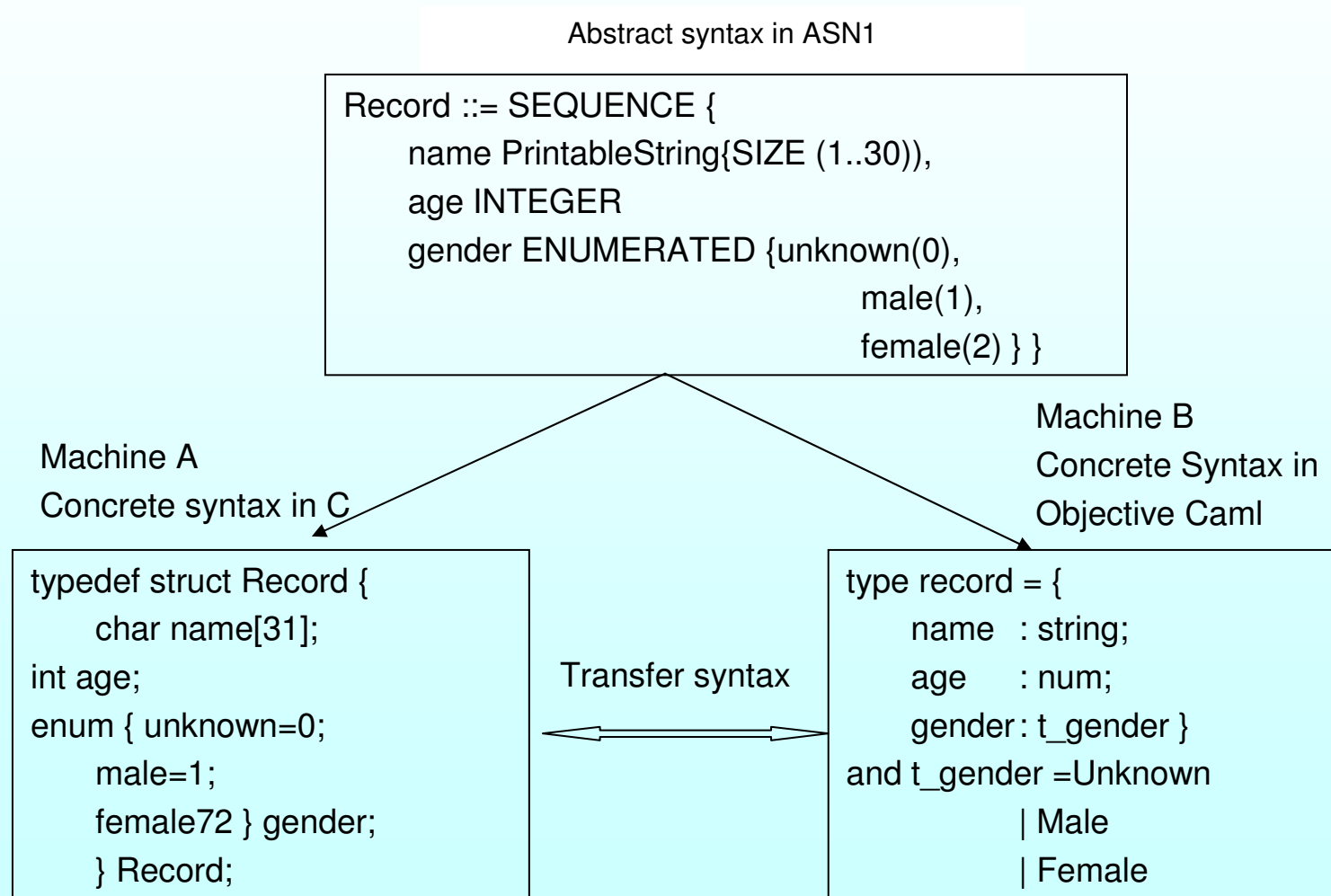
$2N$ konverzió szükséges
Pl.: $N=4$ esetén 8

Formális leírás – ASN.1

- **Fogalmak:**

- Abstract Syntax:
- Formális szabályok összessége amely segítségével objektum struktúrákat lehet leírni függetlenül azok gépi ábrázolásától.
- Concrete Syntax
- Egy konkrét programnyelven (pl. C++) leírt adatszerkezet
- Transfer Syntax: A hálózaton továbbított adatok bitszintű ábrázolása. Ez a gyakorlatban oktetek sorozata melyet a fizikai csatornán továbbítunk.
- Encoding rules: Abstract és Transfer Syntax-ú ábrázolások közötti konverzió szabályainak összessége. Pl.: BER (Basic Encoding Rules)

Formális leírás – ASN.1



Formális leírás – ASN.1

- **Bitfolyam kialakulása:**

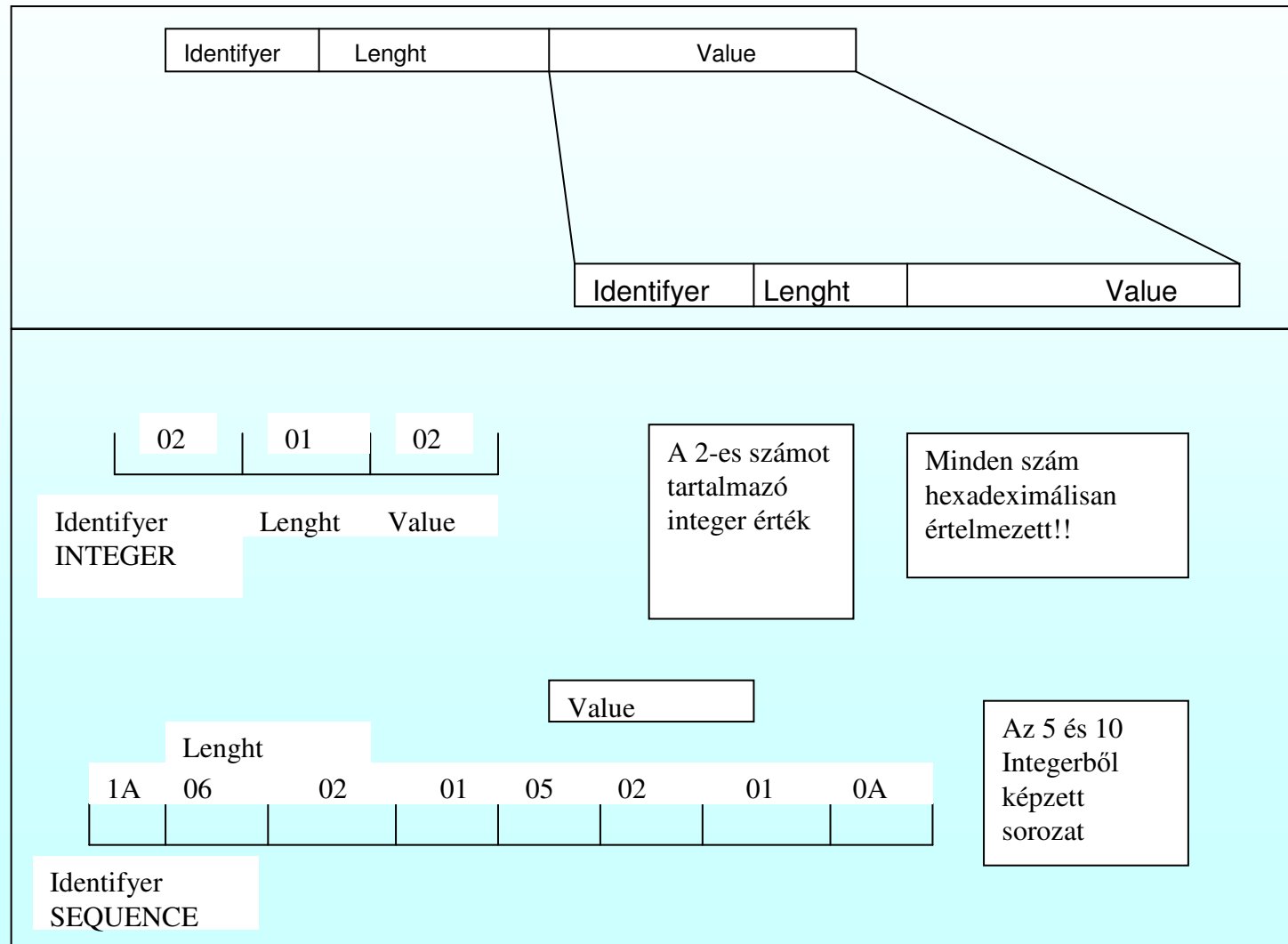
- Az Encoding Rules alkalmazásával alakul ki a szabványos bitfolyam.

Többféle Encoding Rule létezik:

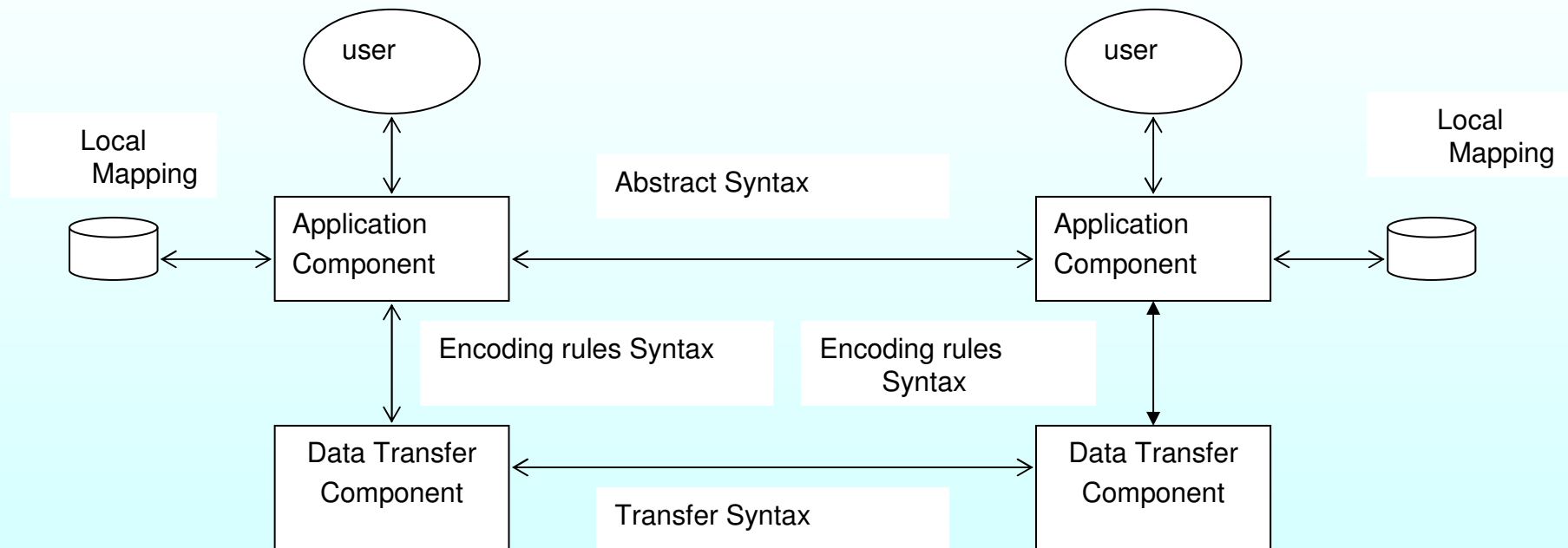
- BER Basic Encoding Rules
 - PER Packed Encoding Rules
 - DER Distinguished Encoding Rules
 - CER Canonical Encoding Rules

Formális leírás – ASN.1

PI. BER
esetén
érvényes ez
a kódolás



Formális leírás – ASN.1



Formális leírás – ASN.1

- **Vonatkozó szabványok:**

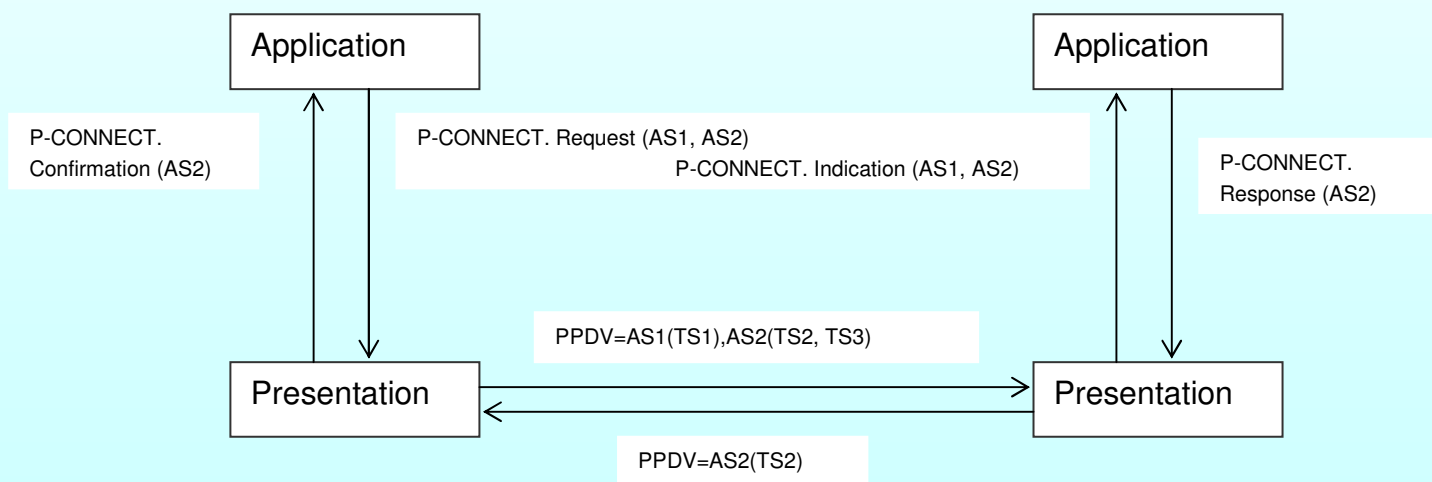
- ITU-T X. 680/ ISO 8824-1: ASN.1
- ITU-T X. 690/ ISO 8825-1: BER

- **Alkalmazások:**

- X. 400. Message Handling System
- X. 500. Directory Services
- H. 323. Multimedia Communication System
- SNMP Simple Network Management Protocol
- Stb.

Formális leírás – ASN.1

- Az ASN.1 az OSI modell Presentation Layer-éhez kapcsolódik, ez kezeli az átvitt adatok szintaxisát.



PPDV Presentation Protocol Data Value

Formális leírás – ASN.1

- A Presentation Layer szolgáltatásai:
 - Transfer Syntax egyeztetése.
 - Transfer Syntax-ok azonosítása (ha több van)
 - Transfer Syntax és az alkalmazásfüggő ábrázolások közötti átalakítás
 - Transfer Syntax és az alkalmazásban lévő Abstract Syntax összerendelése
 - Session Layer szolgáltatás elérése.
- Az ASN.1 csak az információ strukturális felépítésével (szintaxis) foglalkozik. Az ASN.1 objektumok nem tartalmaznak operátorokat, metódusokat. **Az ASN.1 NEM egy programozási nyelv.**
- **A programozási nyelvekben (Java, C++ ...) szoftver eszközök állnak rendelkezésre az ASN.1 objektumok kezelésére.**

Formális leírás – ASN.1

- **ASN.1 alapok**

- **Type:**

- Értékek nem üres halmaza, lehet kódolni és továbbítani.
- Type megfelel az OOP-ben az osztálynak (Class). A Type assignment-el objektum osztályokat definiálunk. Az objektumok az absztrakt adattípus egyedei (instance).
- Type fajtái:
 - Alap (Basic type)
 - Összetett (Structured type)
- Type assignment:
 - Married ::= BOOLEAN
 - Age ::= INTEGER
 - Picture ::= BIT STRING
- Value assignment:
 - családiállapot Married ::= TRUE
 - kor Age ::= 27
 - fax Picture ::= '01010011010'B

- **Szintaktikai szabályok:**

- Identifier, keyword, reference (Type, Value) betűvel kezdődik, számmal/betűvel és egy kötőjellel folytatódhat.
- Keyword minden betűje nagybetű (kivéve néhány string típust pl.: PrintableString)
- A Type reference nagybetűvel kezdődik.
- Identifier és value reference kisbetűvel kezdődik.

- **Komment:**

- --comment

- **Definition (Assignment):**

- ::=

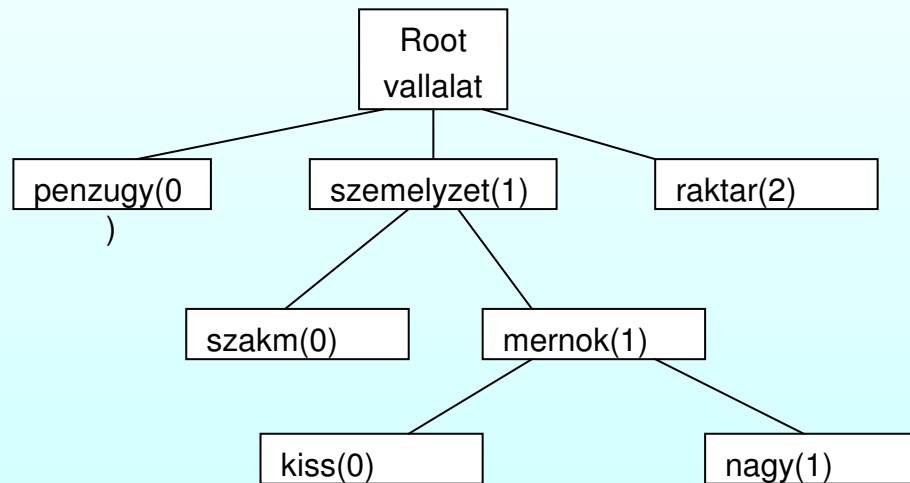
Formális leírás – ASN.1

- Type alaptípusai (Basic Types)
 - BOOLEAN (Értéke lehet: FALSE, TRUE)
 - INTEGER (+ vagy – egész szám)
 - REAL (Valós szám lebegőpontos ábrázolással)
 - ENUMERATED (Megszámlálható sorozat)
 - BIT STRING
 - OCTET STRING
 - OBJECT IDENTIFIER (Egész számok sorozata, amely az objektumot azonosítja)
 - ...String (pl.: IA5String)
 - UTCTime (Dátum)
- Basic Type példák:
- Type Definitions:
 - Szamlalo ::= INTEGER
 - IpCim ::= OCTET STRING
 - Honapok ::= ENUMERATED {jan(1), febr(2), marc(3), apr(4), maj(5), jun(6), jul(7), aug(8), szept(9), okt(10), nov(11), dec(12)}

Formális leírás – ASN.1

- Subtype Definitions:
 - Szamlalo ::= INTEGER (0..127)
 - IpCim ::= OCTET STRING (SIZE(4))
 - Nyar ::= Honapok (jun | jul | aug)
 - Tel ::= Honapok (12 | 1 | 2)
 - Faxsor ::= BITSTRING(SIZE(1720))
- Value Assignment:
 - leallasok Szamlalo ::= 12
 - ipmask IpCim ::= `FFFFFFF00`O
 - maihonap Honapok ::= aug
- Value Assignment OBJECT IDENTIFIER

Formális leírás – ASN.1



Pl.: nagy OBJECT IDENTIFIER ::= {szemelyzet(1) mernok(1) 1}

kiss OBJECT IDENTIFIER ::= {1 1 0}

kiss OBJECT IDENTIFIER ::= {szemelyzet mernok 0}

Formális leírás – ASN.1

- Structured Types
- SEQUENCE : Egy bizonyos számú objektum együttese. A objektumok típusa különbözhet, sorrendjük azonban meghatározó.
- Példa:
- Type Definition:
 - UserAccount ::= SEQUENCE {
 - username VisibleString
 - password VisibleString
 - accountNr INTEGER
 - }
- Value Assignment:
 - myAccount UserAccount ::= {
 - username "Kiss"
 - password "mama14"
 - accountNr 555
 - }

Formális leírás – ASN.1

- SEQUENCE OF : Egy bizonyos számú objektum együttese. A objektumok típusa azonos, sorrendjük meghatározó.
- Példa:
- Type Definition:
- MemberCountries ::= SEQUENCE OF VisibleString
- AccountRegistry ::= SEQUENCE OF UserAccount
- Value Assignment:
- euMembers MemberCountries ::= SEQUENCE OF {
 "Austria", "Belgium", "Denmark" }
- SET: Egy bizonyos számú objektum együttese. A objektumok típusa különbözhet, sorrendjük nem meghatározó.
- Példa:
- Type Definition:
- UserAccount ::= SET {
 - Username [0] VisibleString
 - Password [1] VisibleString
 - accountNr [2] INTEGER
 - }
- [0][1][2] : contex-specific Tags : opcionális, ha nincs a számozás automatikus
- Value Assignment:
- myAccount UserAccount ::= {
 - accountNr 555
 - password "mama14"
 - username "Kiss"
 - }
- SET OF: Egy bizonyos számú objektum együttese. A objektumok típusa ugyanolyan, sorrendjük nem meghatározó.