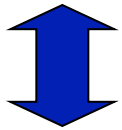
SDL

Távközlési szoftverek

- Távközlési szoftverek:
  - A kommunikációs infrastruktúra építőelemei
- Kiemelt követelmények:
  - Megbízhatóság
  - Együttműködés
    - Különböző rendszerek között
    - Különböző gyártók azonos célú rendszerei között
    - → Szabványok!

Távközlési szoftverek

- Ellentmondás:
  - Komplexitás és méret növekedése +
    Fejlesztési idő csökkenése (Piaci elvárások)
    → Fejlesztési hibák valószínűsége nő

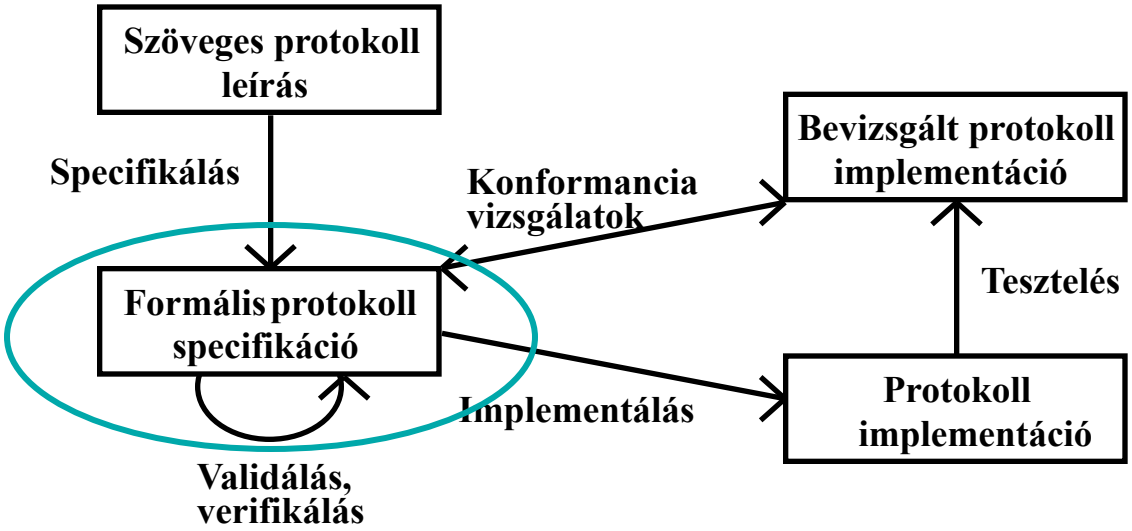


  - Megbízhatóság + Együttműködés
- (Lehetséges) megoldás:
  - Formális módszerek alkalmazása

Formális módszerek

- Matematikailag jól megalapozott nyelvek és metodológiák,
- Támogatást nyújtanak a szoftver és egyéb rendszerek specifikációja és megvalósítása (implementációja) során.
- Előnyök a nem formális leírásokkal szemben:
  - Egzakt matematikai háttér (véges automaták, kiterjesztett véges automaták stb.)
  - egyértelműség
  - teljesség
  - könnyű (automatikus) ellenőrizhetőség támogatása
  - (könnyű) implementálhatóság

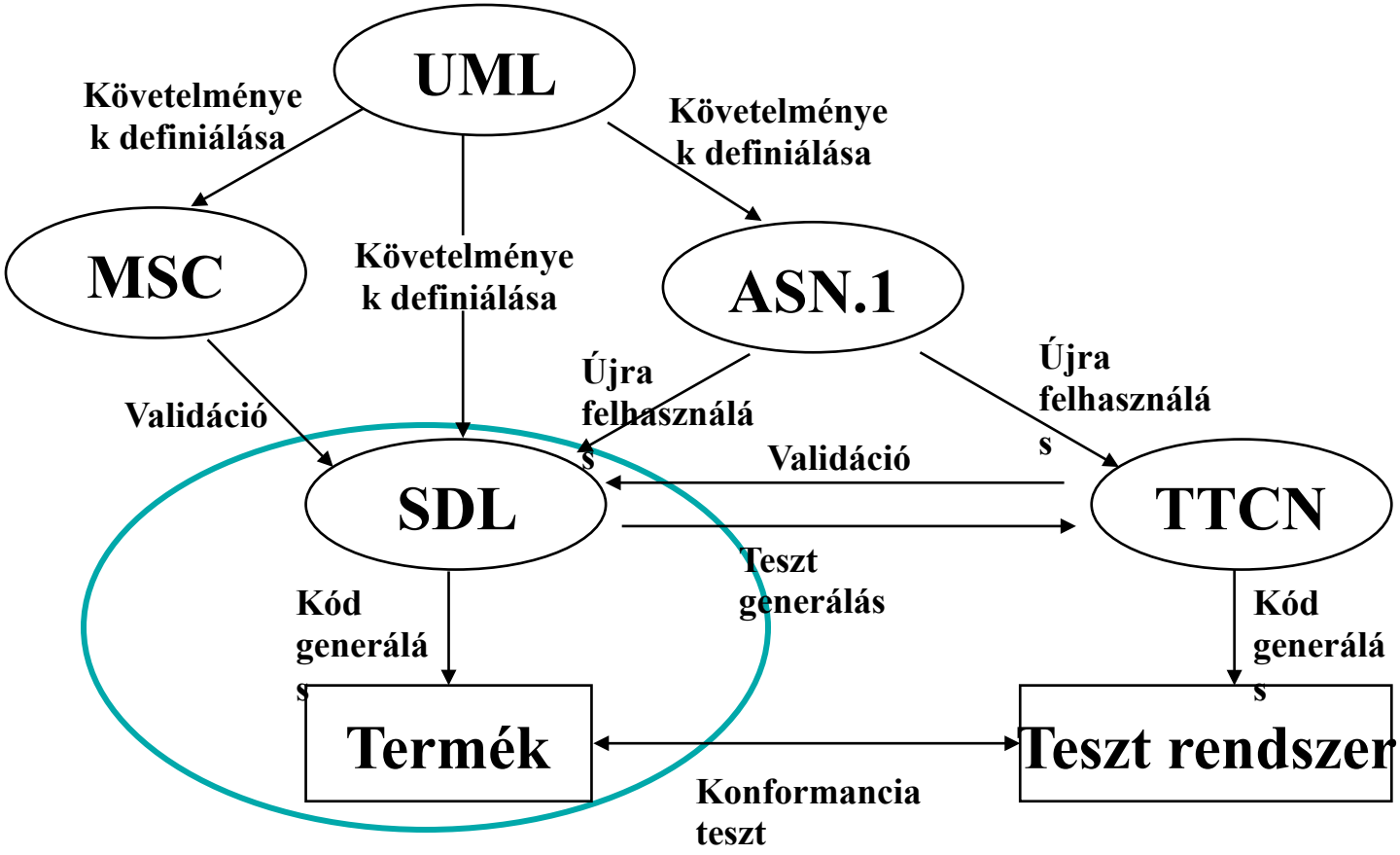
Távközlési szoftverfejlesztés (protocol engineering)



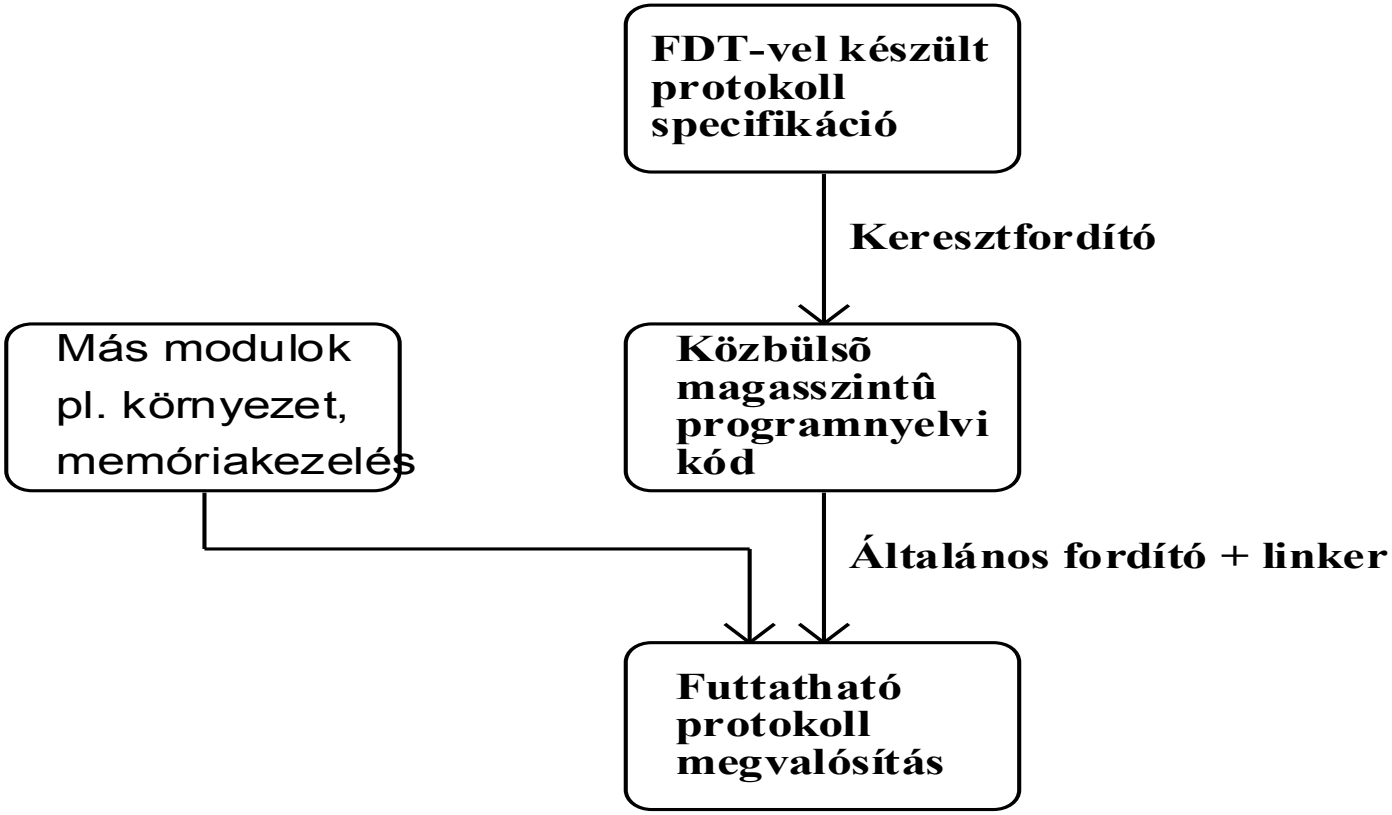
Formális specifikációk előnyei távközlési szoftverfejlesztés során

- Szabványosítás
  - Egyértelmű leírás.
  - Együttműködési (interoperabilitási) problémák elkerülése.
- A fejlesztés korai fázisa
  - Rendszer megértése.
  - Tervezési hibák korán felfedezhetővé válnak.
- Megvalósítás
  - Történhet a specifikáció részleteinek kidolgozásával.
  - Közvetlen kód előállítás.
- Tesztelés
  - A jó és rossz működés megkülönböztetése
  - Automatikus tesztgenerálás

Kapcsolat a különböző leírónyelvek között



Automatikus protokoll implementálás



Közbülső nyelv

- milyen könnyen és *hatékonyan* fordítható le rá az adott FDT
- elterjedt legyen (→ hordozhatóság)
- jellemzően: C nyelv, JAVA
- “ kézi” beszúrások:
  - áttekinthetőség
  - könnyű javítás, fenntartás, bővítés
  - → OOP

Specifikációs technikák

- Kommunikáló véges automaták
  - Estelle
  - SDL
- Gráf modellek
  - Petri - hálók
- Algebrai modellek
  - Kommunikáló rendszerek kalkulusa (CCS)

    LOTOS
- Magasszintű programozási nyelvek

Véges automaták - Finite state machines (FSM)

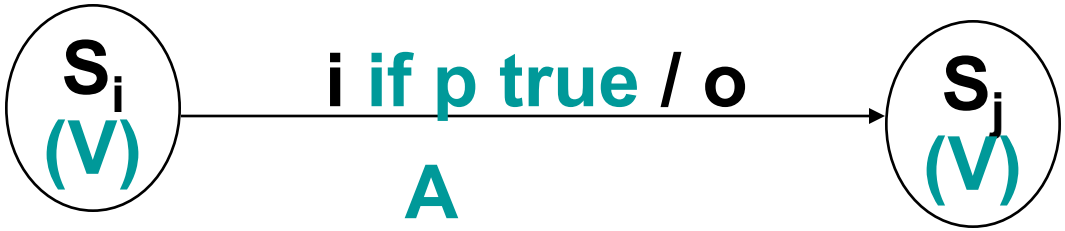
FSM = < S, I, O, f(s, i), S_0 >

- S: States (állapotok)
- I, O: Input / Output ABC
- S_0: Start state (kezdőállapot)

Kiterjesztett véges automaták - Extended finite state machines (EFSM)

FSM = < S, I, O, V, P, A, f(s, i, p), S_0 >

- **V : Variables (változók)**
- **P : Predicates (conditions) (predikátumok, feltételek)**
- **A : Actions (akciók, tevékenységek)**



SDL (Specification and Description Language)

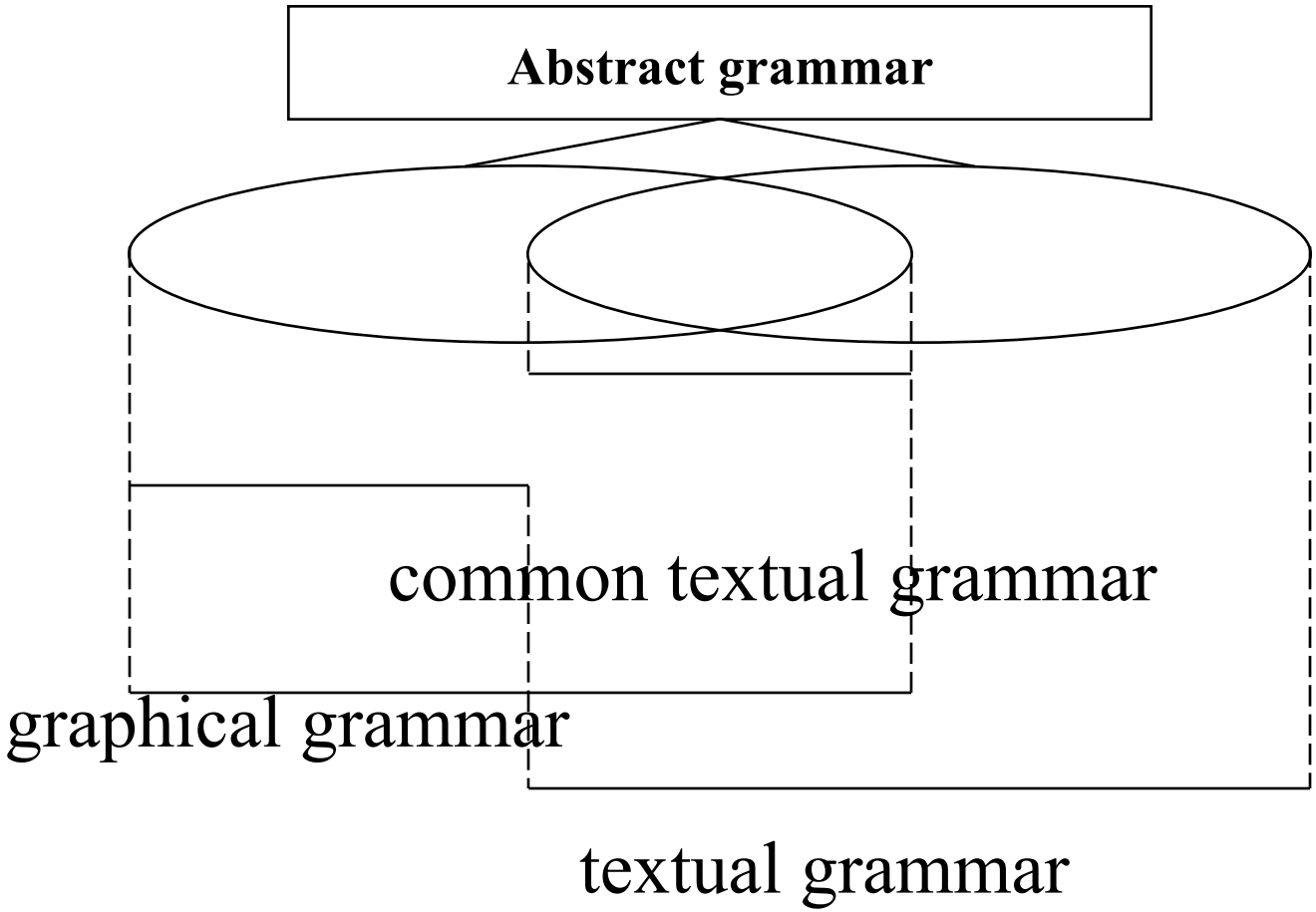
- **Szabványosított specifikációs nyelv**
  - **Elsősorban telekommunikációs protokollok specifikációja**
  - **Más alkalmazási terület: nagy megbízhatóságú eszközök pl. orvosi műszerek, vagy repülőgép szoftverek**
- **ITU International Telecommunication Union (korábban CCITT) szabványa**
- **Fejlesztésének kezdete: 1972**
- **Első szabvány: 1976**
- **Négyévente revízió**

SDL

- **Z100 GENERAL DESCIPTION**
- **Z101 BASIC DESCRIPTION**
- **Z102 STRUCTURAL DESCRIPTION**
- **Z103 REFINEMENTS**
- **Z104 DATA**
- **Z105 SDL COMBINED WITH ADT**
  (ADT - Abstract Data Type)

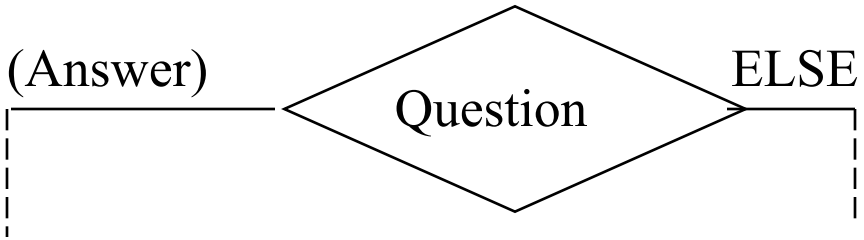
SDL/GR GRAPHICAL VERSION
SD/PR PROGRAMMING LANGUAGE VERSION

Az SDL nyelv hierarchiája



SDL/GR és SDL/PR

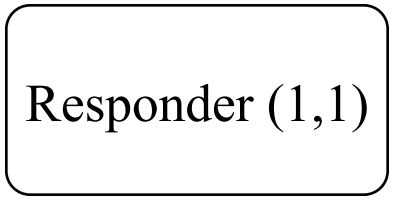
SDL/GR



SDL/PR

```
DECISION          Question
    (Answer) :    .........
     ELSE    :    .........
ENDDECISION;
```

SDL/GR



SDL/PR

```
PROCESS Responder (1,1)
    REFERENCED;
```

SDL alapok (1)

- **Szemlélet: világ két fő része**
    1. **Rendszer**
    2. **Környezet**
- **Nyílt rendszer: van kommunikáció a rendszer és a környezet között**
- **Kommunikáció jelek (signal) segítségével történik**

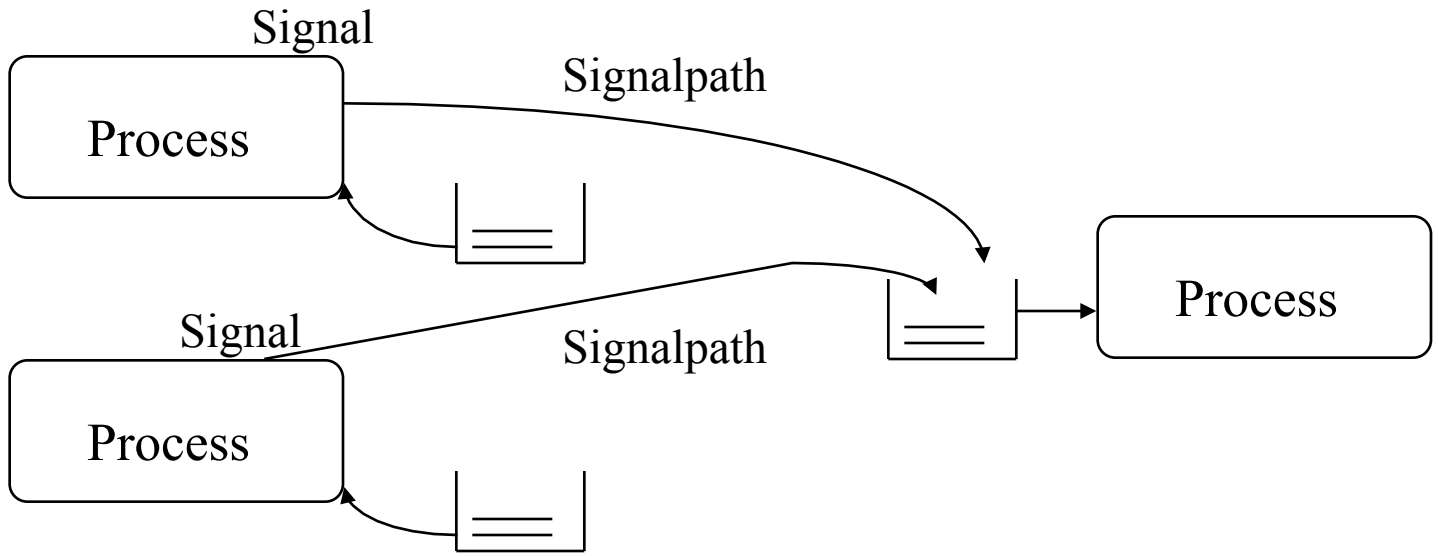
- **Cél: a rendszer viselkedésének leírása**

SDL alapok (2)

Viselkedés leírása:

•Több párhuzamosan futó processz viselkedése adja meg a rendszer egészének viselkedését

•A processzek mind egymásnak mind a környezetnek küldhetnek jeleket



SDL alapok (3)

- **Egy-egy processz viselkedésének leírása – kiterjesztett véges automata segítségével**
- **Változók ~ implicit állapotok**
- **Specialitások**
  - **Minden processznek saját memóriaterület**
  - **Jelfogadásra végtelen queue – alapvetően FIFO**
  - **Jelek meghatározott útvonalon késleltetés nélkül terjednek**
  - **Paraméterezett jelek**
  - **Timer-ek**

Rendszer hierarchikus felépítése

Viselkedés: párhuzamosan működő processzek (EFSM)

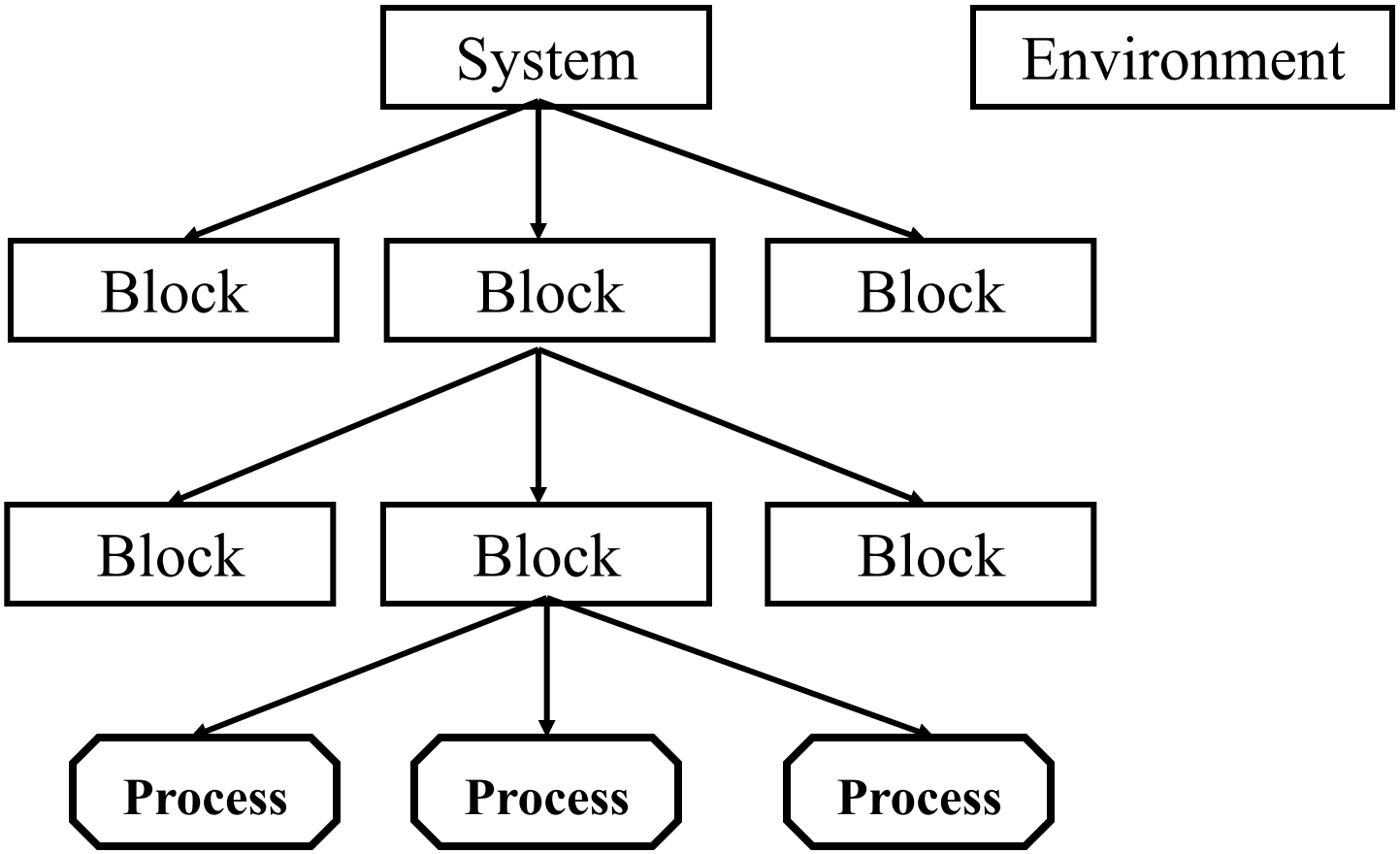
Sok processz esetén átláthatatlan

→ részekre bontás

→ hierarchikus szintek

Építőkövek: blokkok

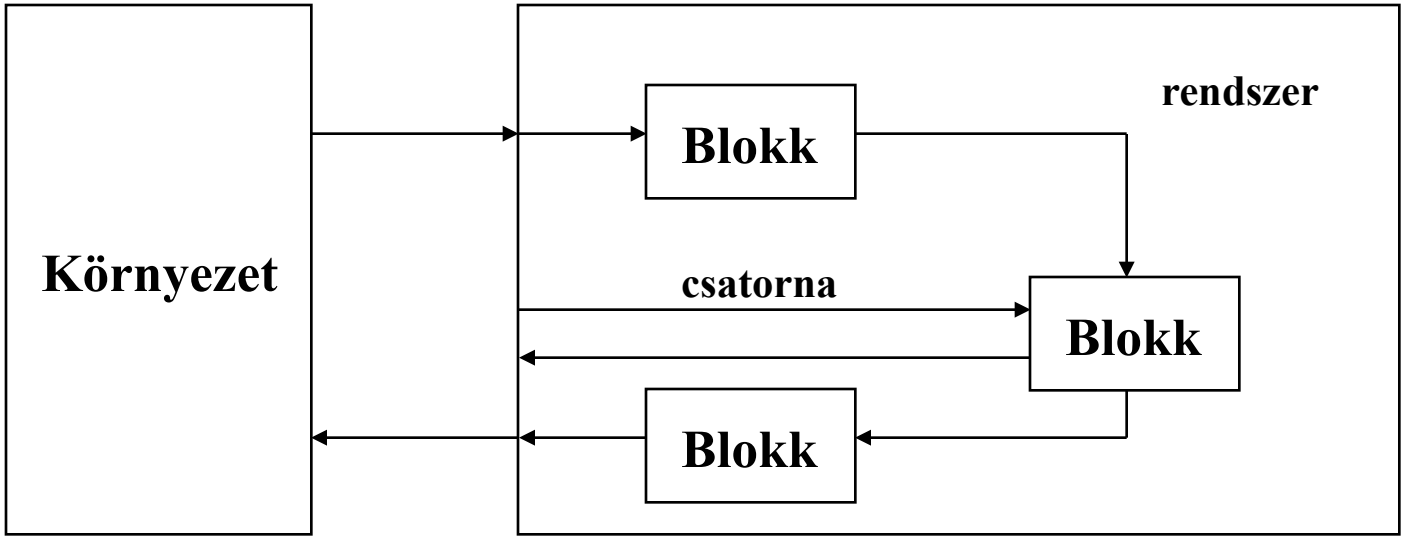
Rendszer hierarchikus felépítése



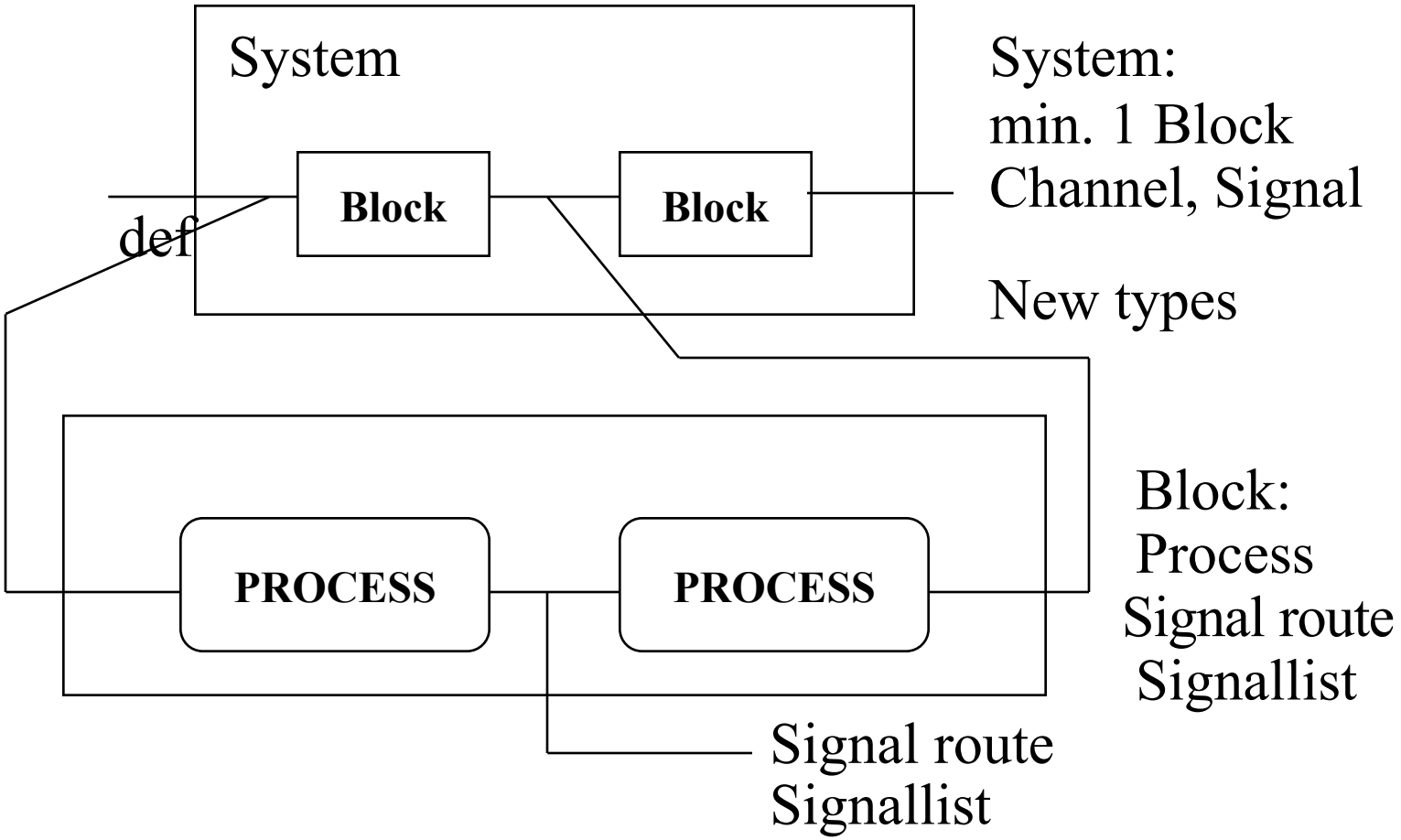
Rendszer hierarchikus felépítése

Blokkok: áttekinthetőséget javítják

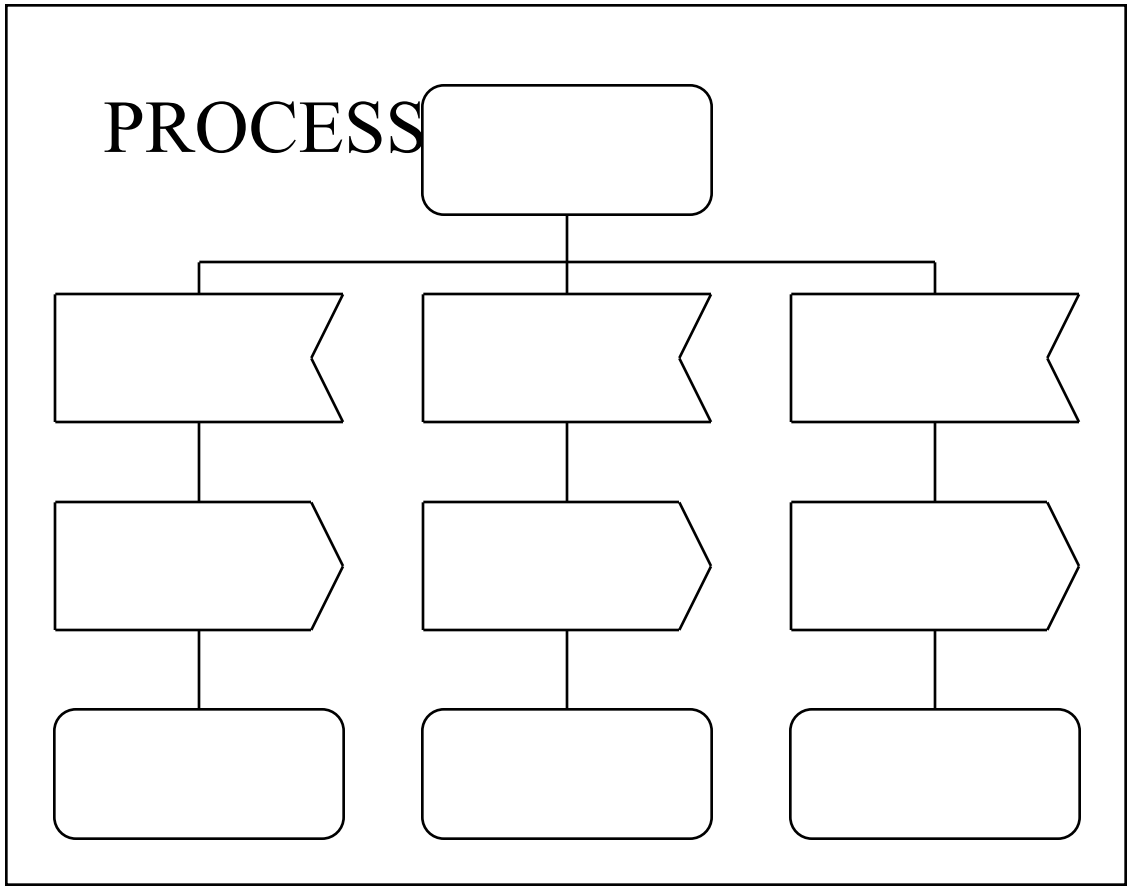
Implicite viselkedést is befolyásolják a szignal csere korlátozásával



System és Block



Process



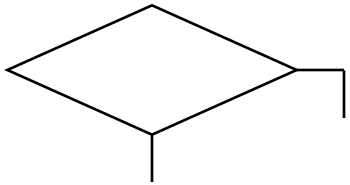
Az SDL/GR elemei (1)

Állapot State

Bemenőjel Input

Kimenőjel Output

Az SDL/GR elemei (2)



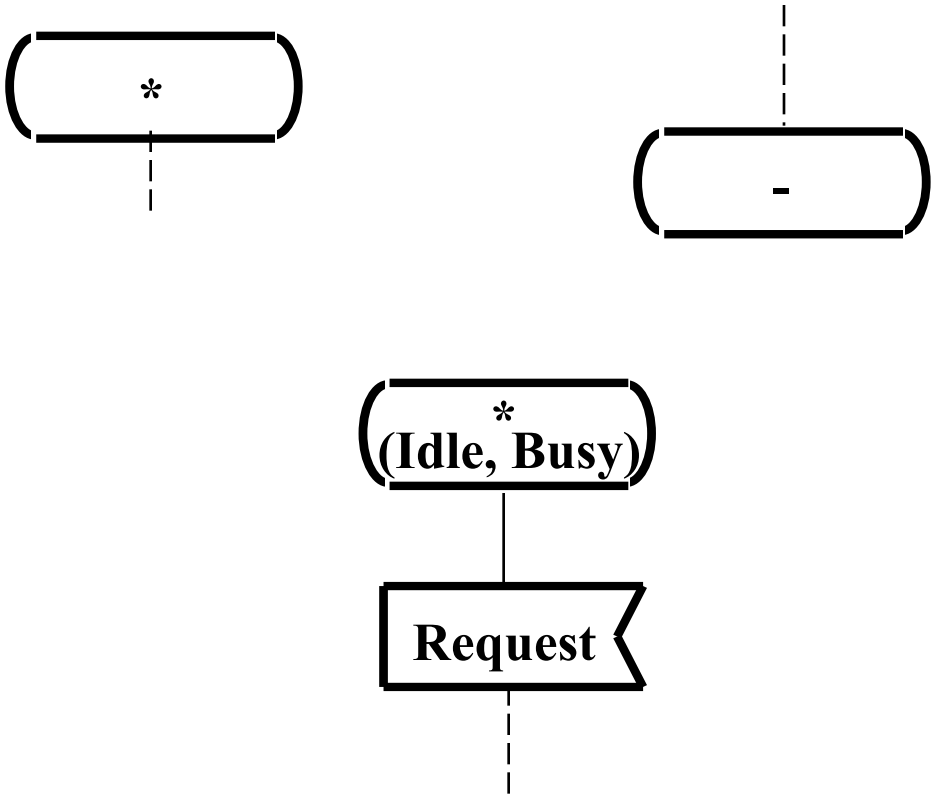
Döntés Decision

Feladat-végrehajtás Task

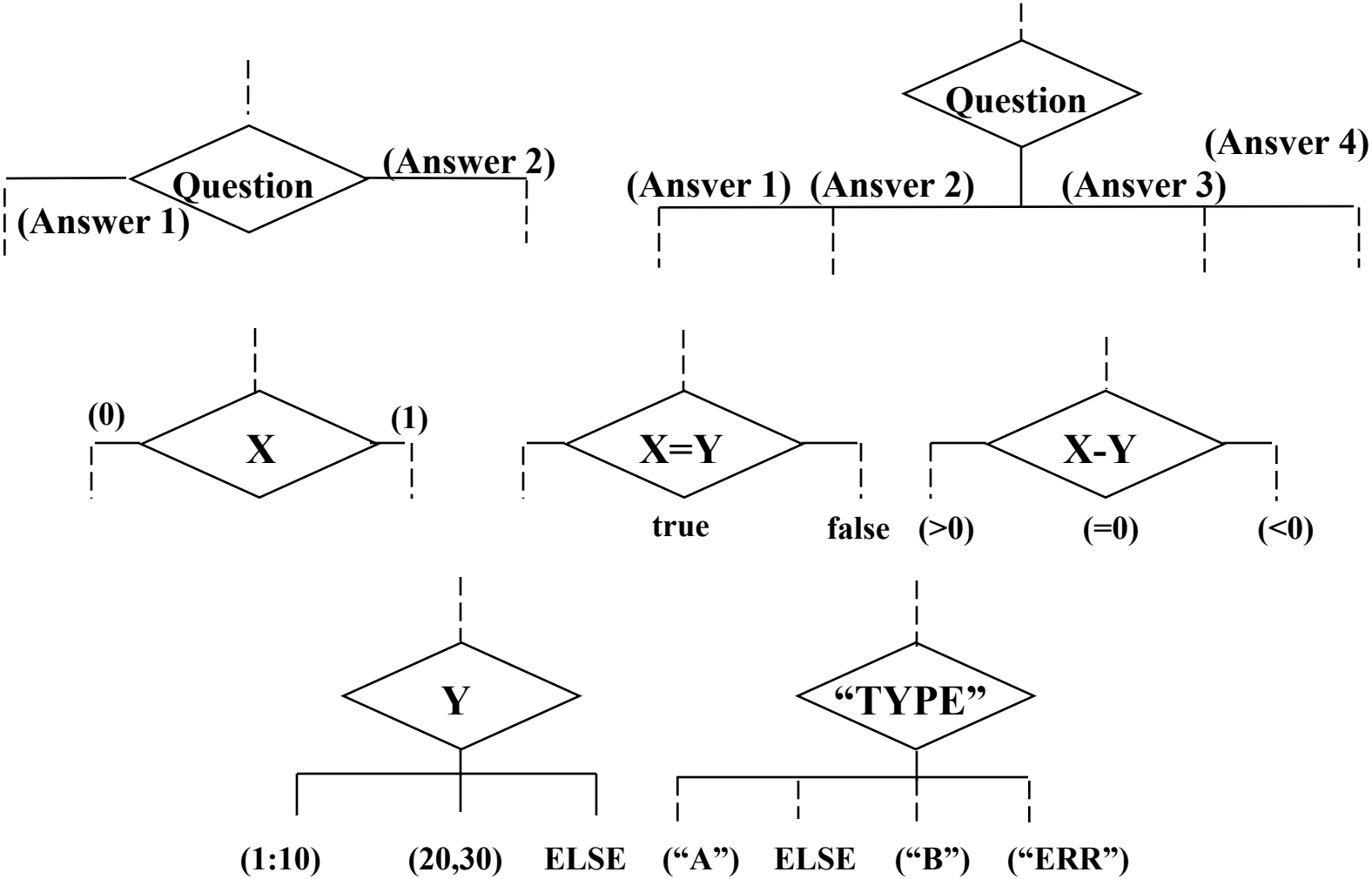
Kezdőállapot

Végállapot

Tömörített leírási formák



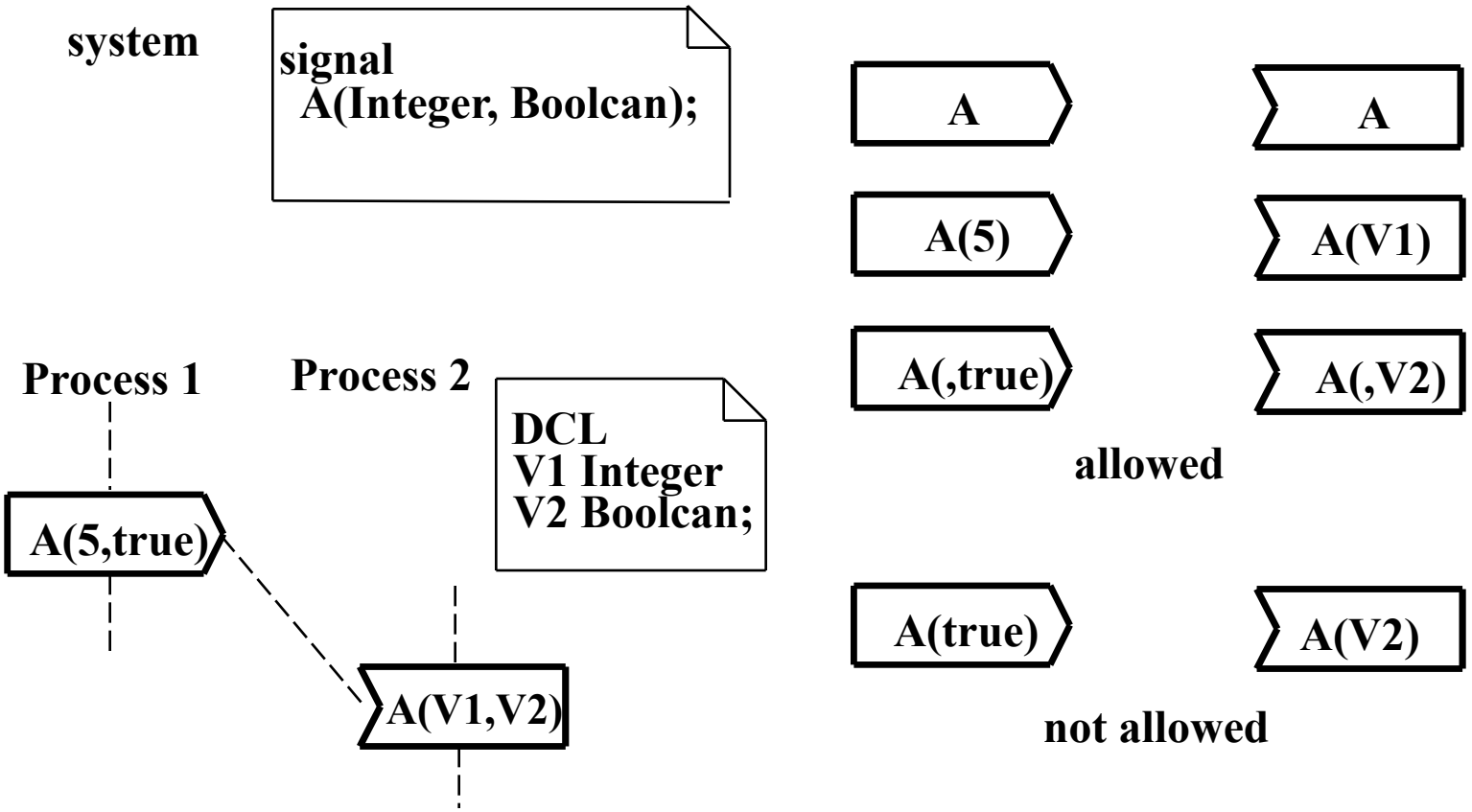
Döntés alternatívák



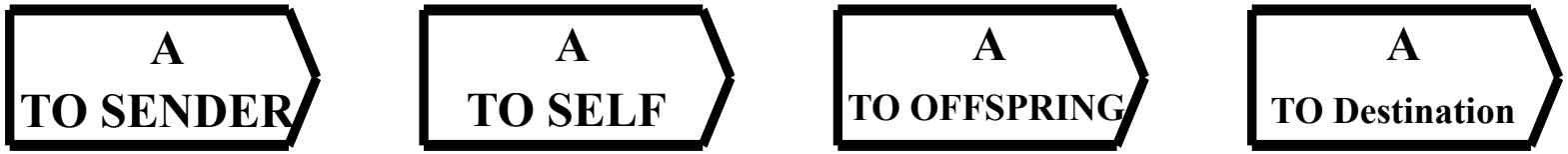
Döntés szabályok

- Szabályok:
- Egy kérdésre legalább két válasznak kell lennie.
- A válaszoknak diszjunktaknak kell lenniük, nem lehet közöttük átfedés.
- Kizárólag egy „*else*” válasz lehet, ami az összes többi válasz komplementere.
- A kérdések, a válaszok és a válasz intervallumok vagy azonos szortba kell hogy tartozzanak, vagy informális szövegnek kell lenniük.

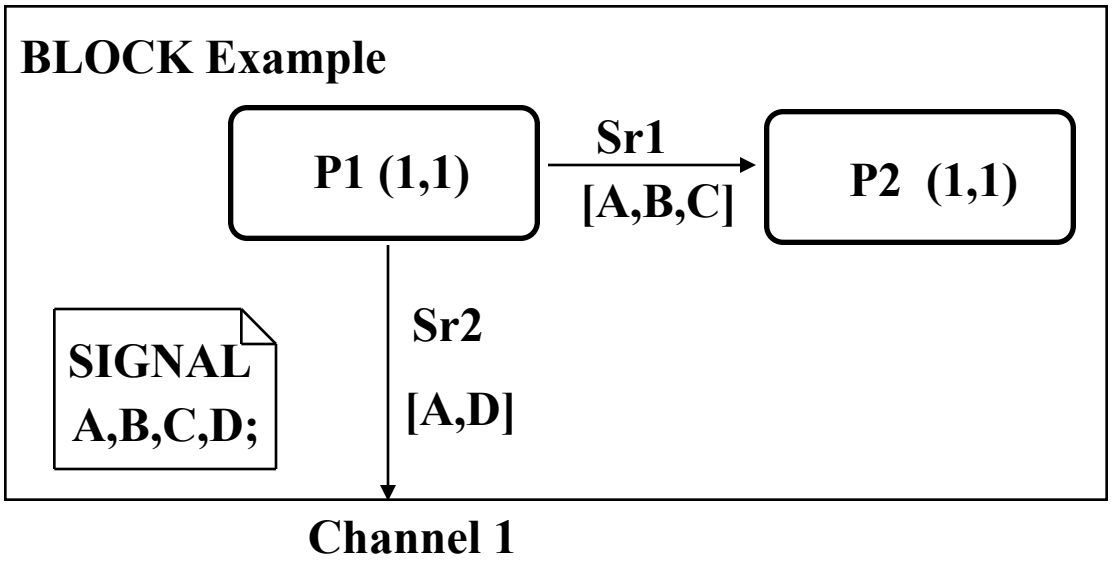
Jelek és adatok

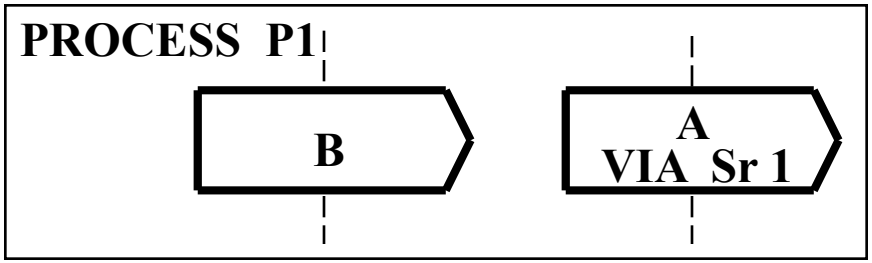


Direkt címzés jelküldéskor

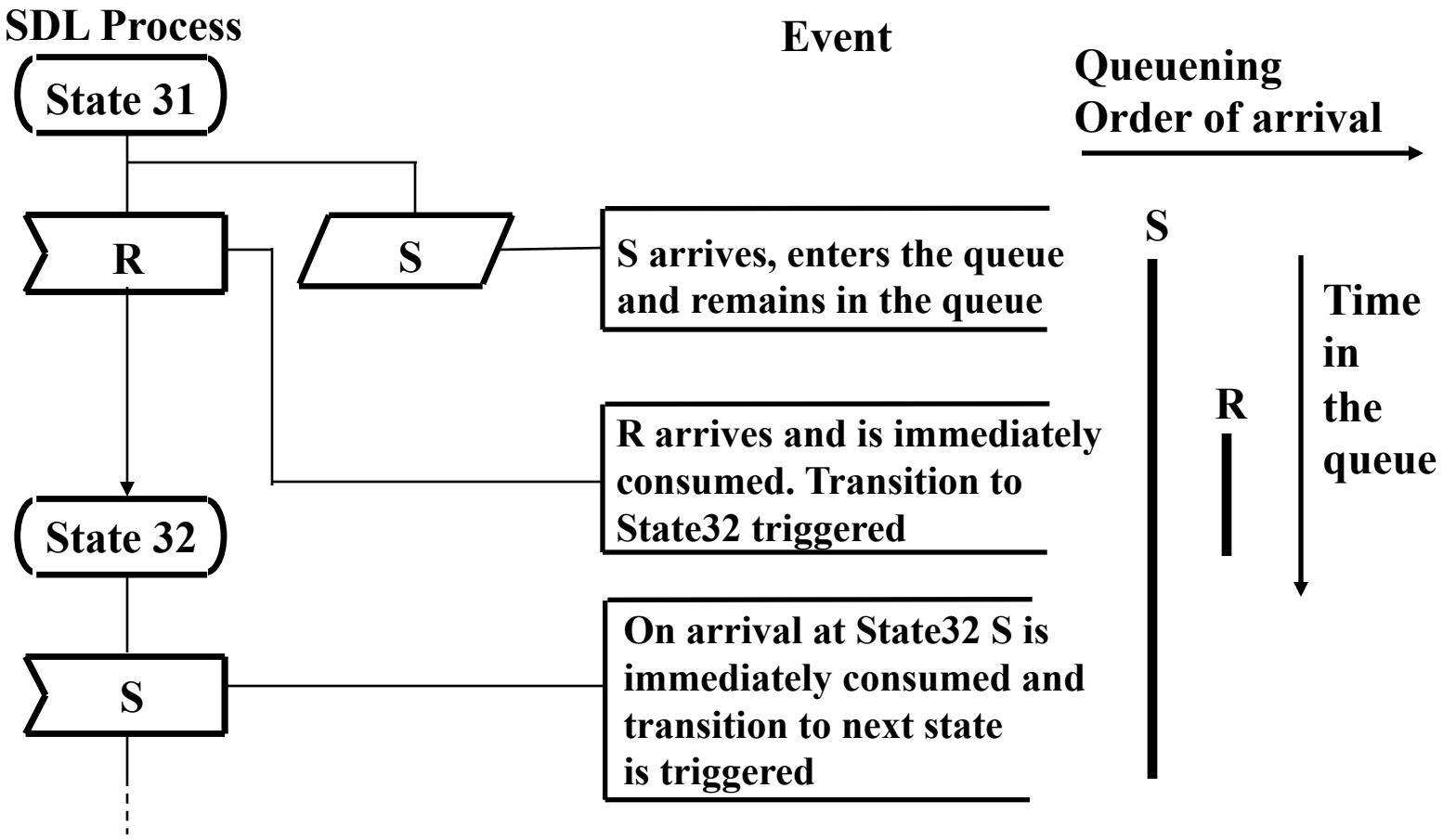


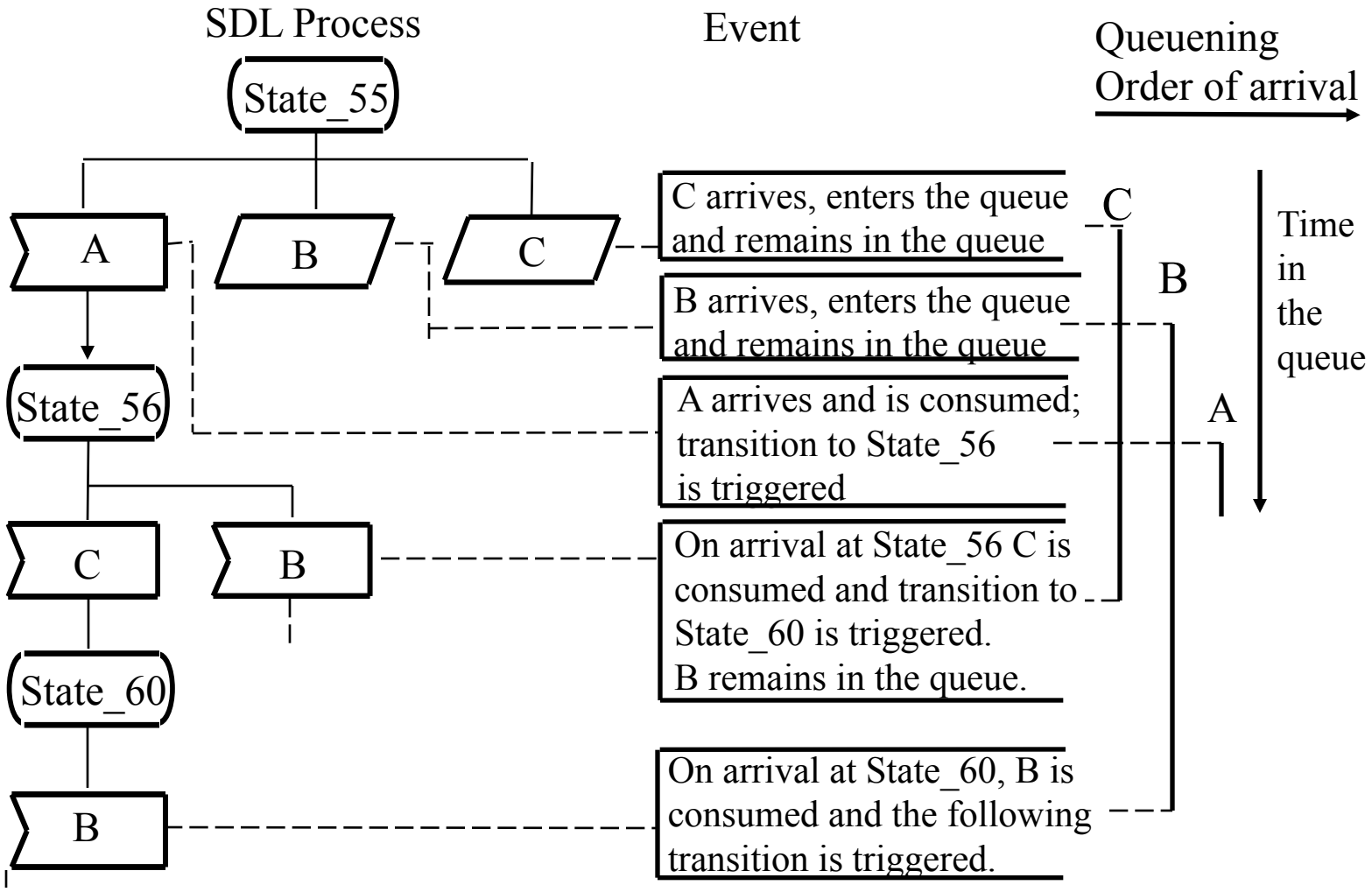
Indirekt címzés jelküldéskor



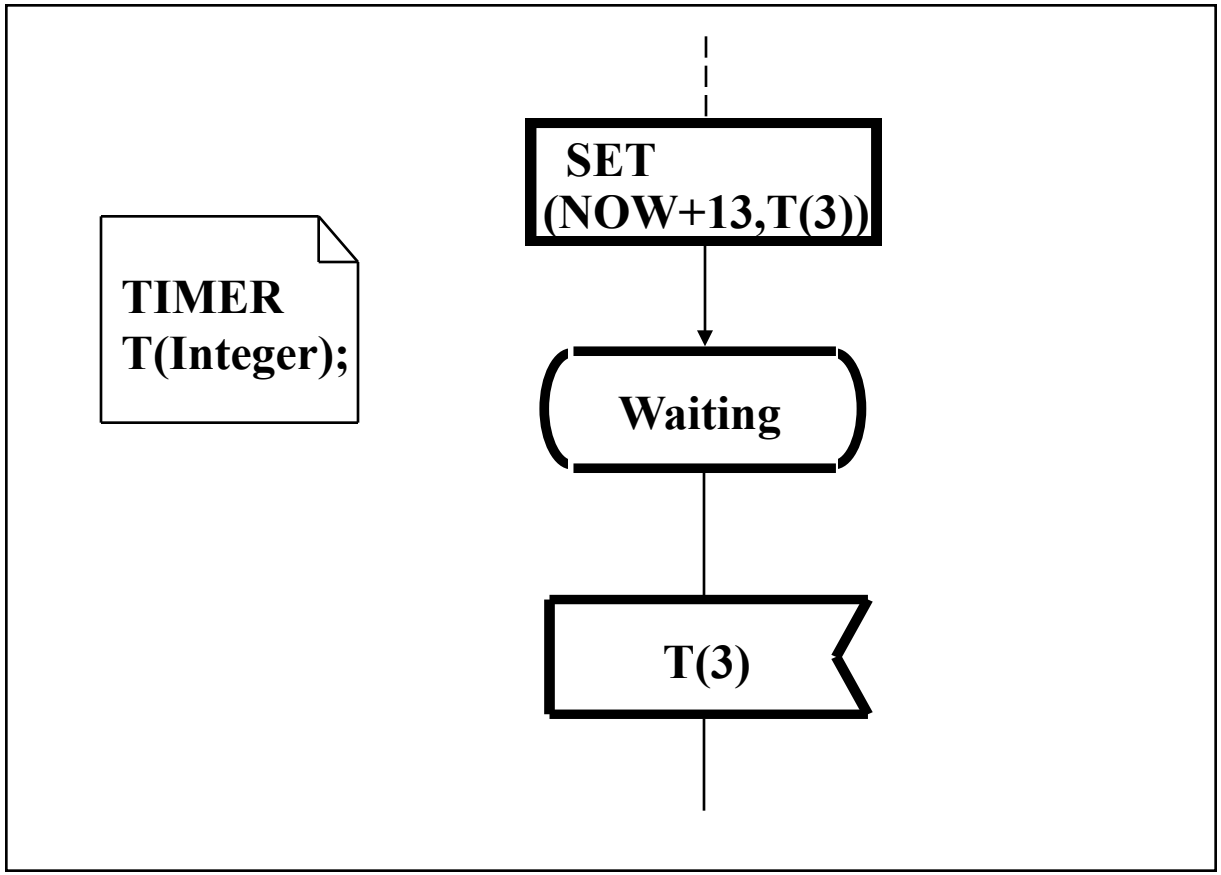


SAVE mechanizmus




SDL Process
Event
Queuening
Order of arrival
State_55
A
B
C
State_56
C
B
State_60
B
C arrives, enters the queue and remains in the queue
B arrives, enters the queue and remains in the queue
A arrives and is consumed; transition to State_56 is triggered
On arrival at State_56 C is consumed and transition to State_60 is triggered. B remains in the queue.
On arrival at State_60, B is consumed and the following transition is triggered.
C
B
A
Time in the queue

Időzítések



Adattípusok definiálása (1)

Absztrakt adattípusok (ADT)

- Implementáció független
- Hardver és a szoftver adta korlátozásokkal nem törődik
- Szokásos típusok (pl. Int Char Bool) előre vannak definiálva
- Új típusok létrehozása
  - Meglevők felhasználásával
  - Teljesen új típusok

Adattípusok definiálása (2)

Típus specifikáció komponensei:

- az értékek és literálok halmaza,
- a fenti értékeken definiált műveletek (operátorok) halmaza,
- axiómák halmaza.

A literálok nevek, amelyek az értékhalmaz elemeit azonosítják, pl. ‘%’, ‘a’, ‘b’, ‘c’ … a char típus esetében

Adattípusok definiálása (3)

Pl. a Boolean előre definiált típus

```
newtype Boolean
    literals True, False;
    operators
        "not"   :   Boolean          ->   Boolean;
         "="    :   Boolean, Boolean  ->   Boolean;
        "/="    :   Boolean, Boolean  ->   Boolean;
        "and,,  :   Boolean, Boolean  ->   Boolean;
        "or"    :   Boolean, Boolean  ->   Boolean;
        "xor,,  :   Boolean, Boolean  ->   Boolean;
```

Adattípusok definiálása (4)

axioms
not **(True)** = = **False;**
not **(False)** = = **True;**
True and **True** = = **True;**
True and **False** = = **False;**
False and **True** = = **False;**
False and **False** = = **False;**
True or **True** = = **True;**
True or **False** = = **True;**
False or **True** = = **True;**
False or **False** = = **False;**
True xor **True** = = **False;**
True xor **False** = = **True;**
False xor **True** = = **True;**
False xor **False** = = **False;**
endnewtype **Boolean**;

Egyszerű előre definiált típusok

- Integer
- Boolean
- Real
- Character (ASCII karakterkészlet)
- Charstring

Összetett típusok

Struktúra

Lényegében a rekord típusnak felel meg, kulcsszava a struct. A mezők egy névből és egy szortból állnak.

Tömb

Definiciója: megadjuk az indexelő és az elem szortot, kulcsszava az array.

Halmaz

Halmaz szort a powerset kulcsszóval és az elem szort megadásával definiálható.

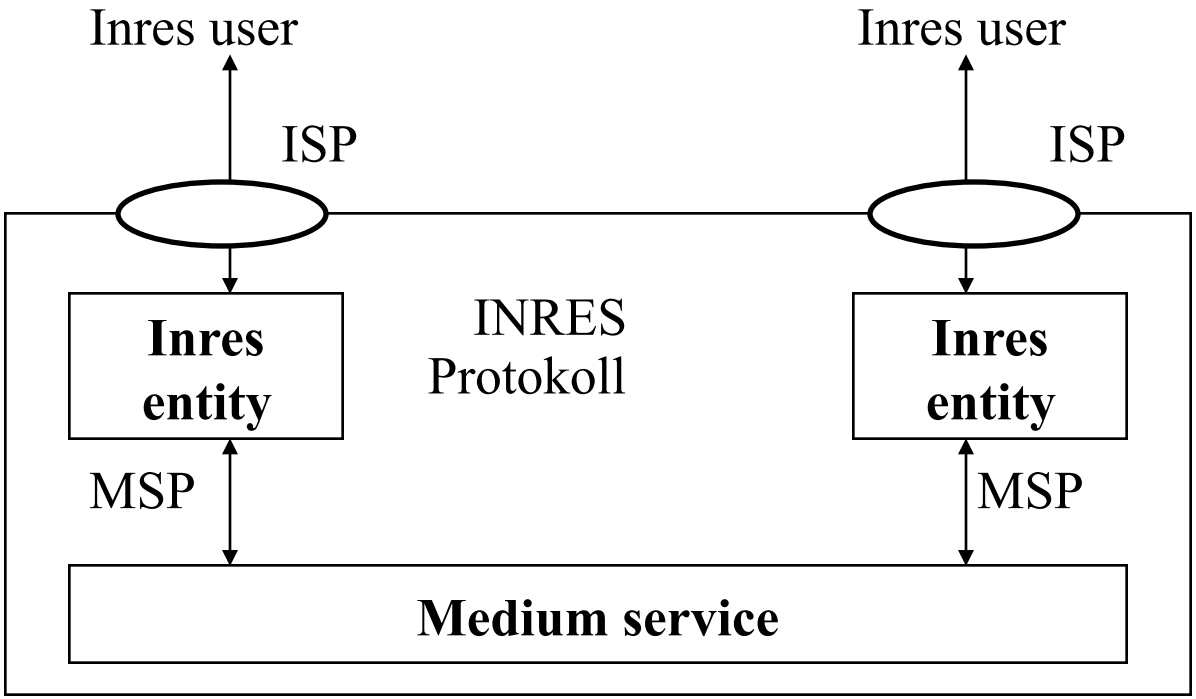
Felsorolási szort

Létrehozás: felsoroljuk a szort literáljait. Az egyenlőségvizsgálat műveleteken kívül minden más operátort explicite kell definiálni.

INRES (Initiator Responder)

- Minta protokoll
- OSI elvekre épül, de nem egy adott OSI réteget valósit meg
- Minta/példa rendszer:
  - Formális módszerek alkalmazására
  - Tesztgenerálásra
  - Etc.
- Kapcsolat-orientált
- Megbizhatatlan Medium service-re épül
- INRES service-t szolgáltat

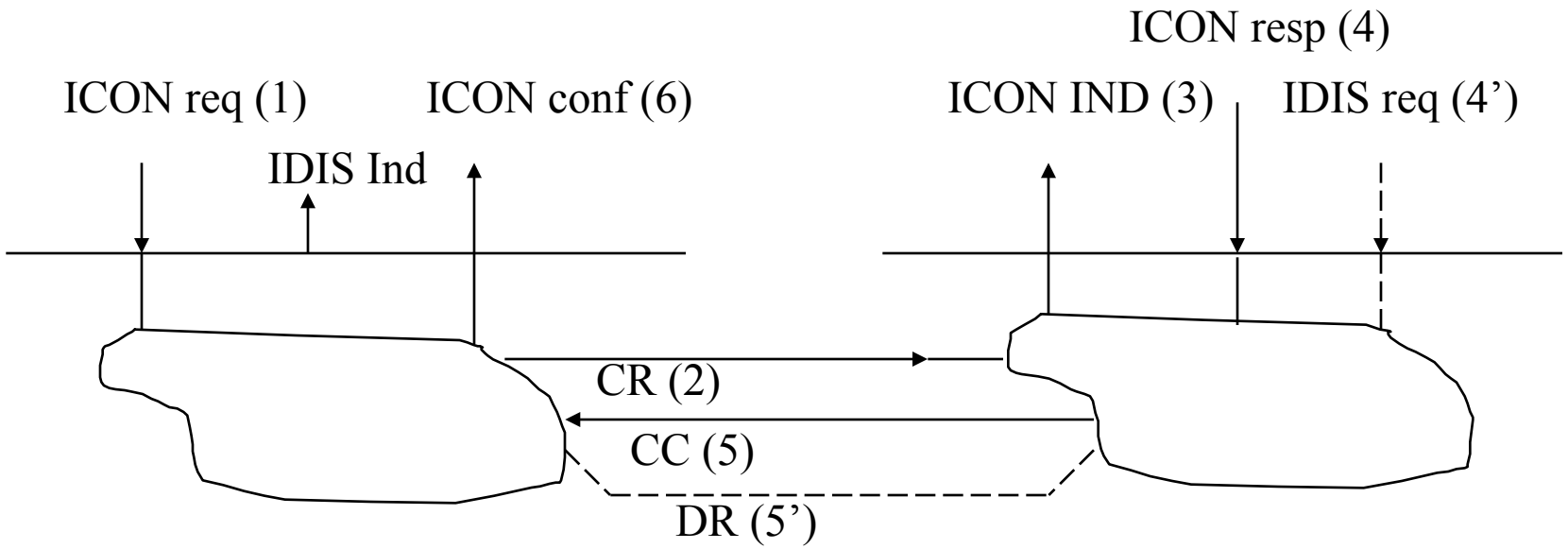
INRES System



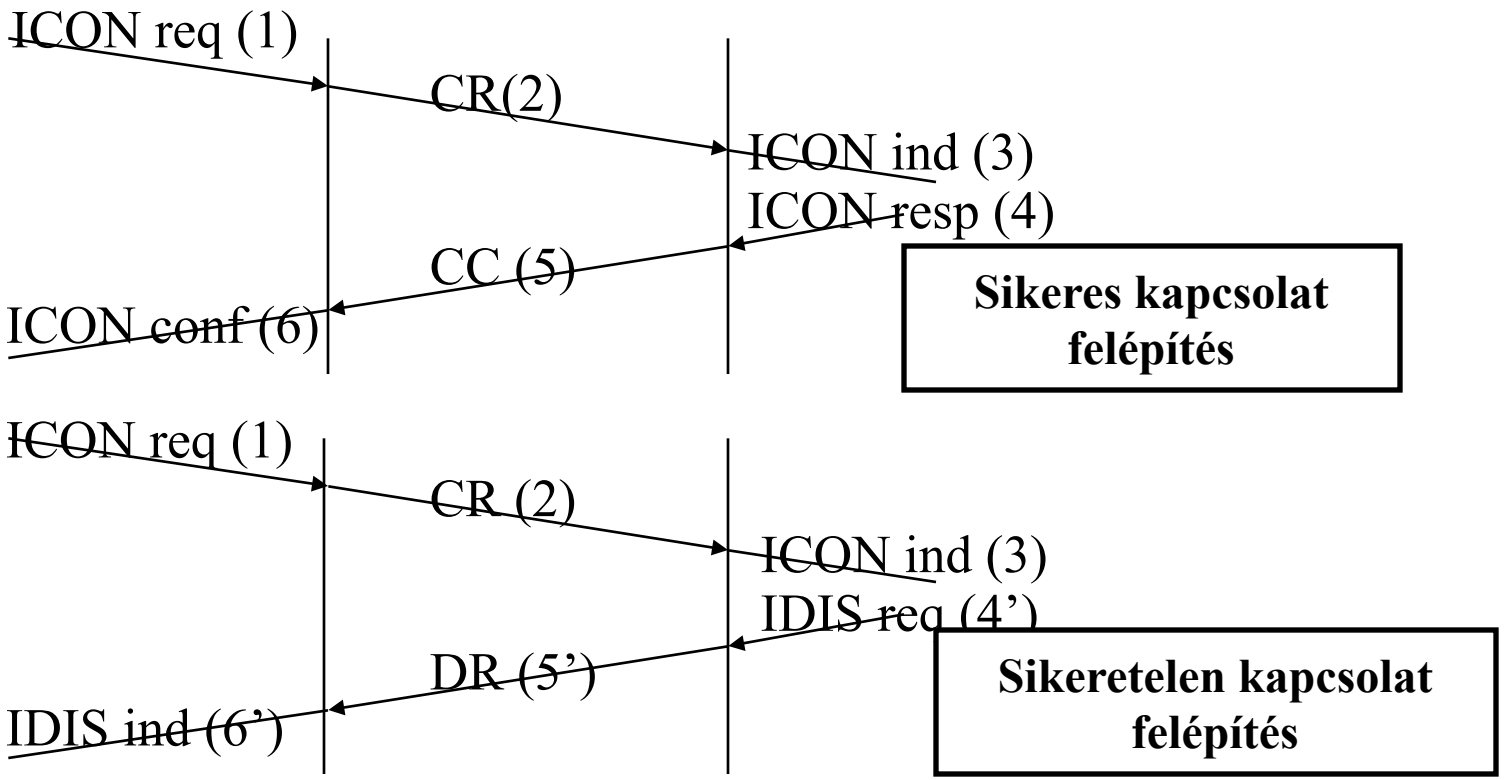
ISP Inres Service Primitives

MSP Medium Service Primitives

Kapcsolatfelépítés (1)

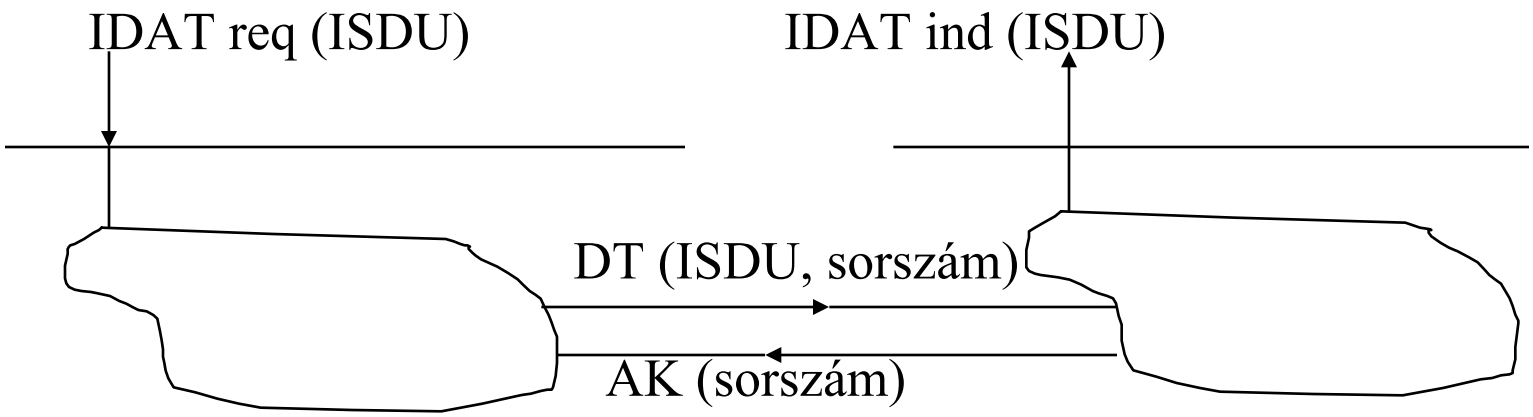


Kapcsolatfelépítés (2)



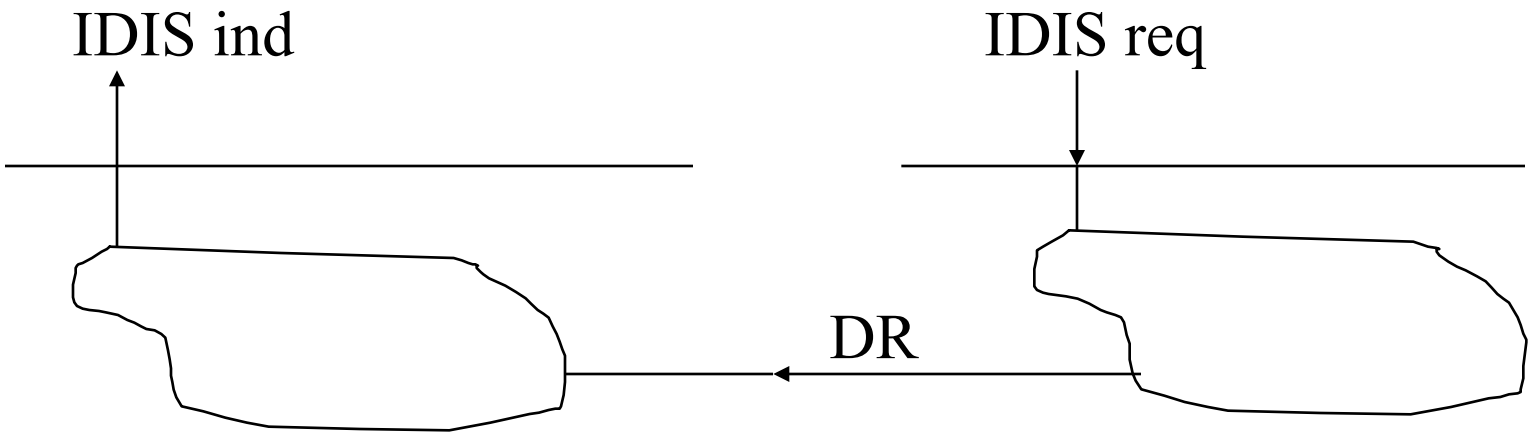
Ha < 5 sec-on belül nem jön válasz, max. 4-szer kéri a kapcsolat felépítését.

Adatátvitel



IDAT req	Initiator DATa request
IDAT ind	Initiator DATa indication
ISDU	Initiator Service Data Unit
DT	DaTa
AK	Acknowledgement

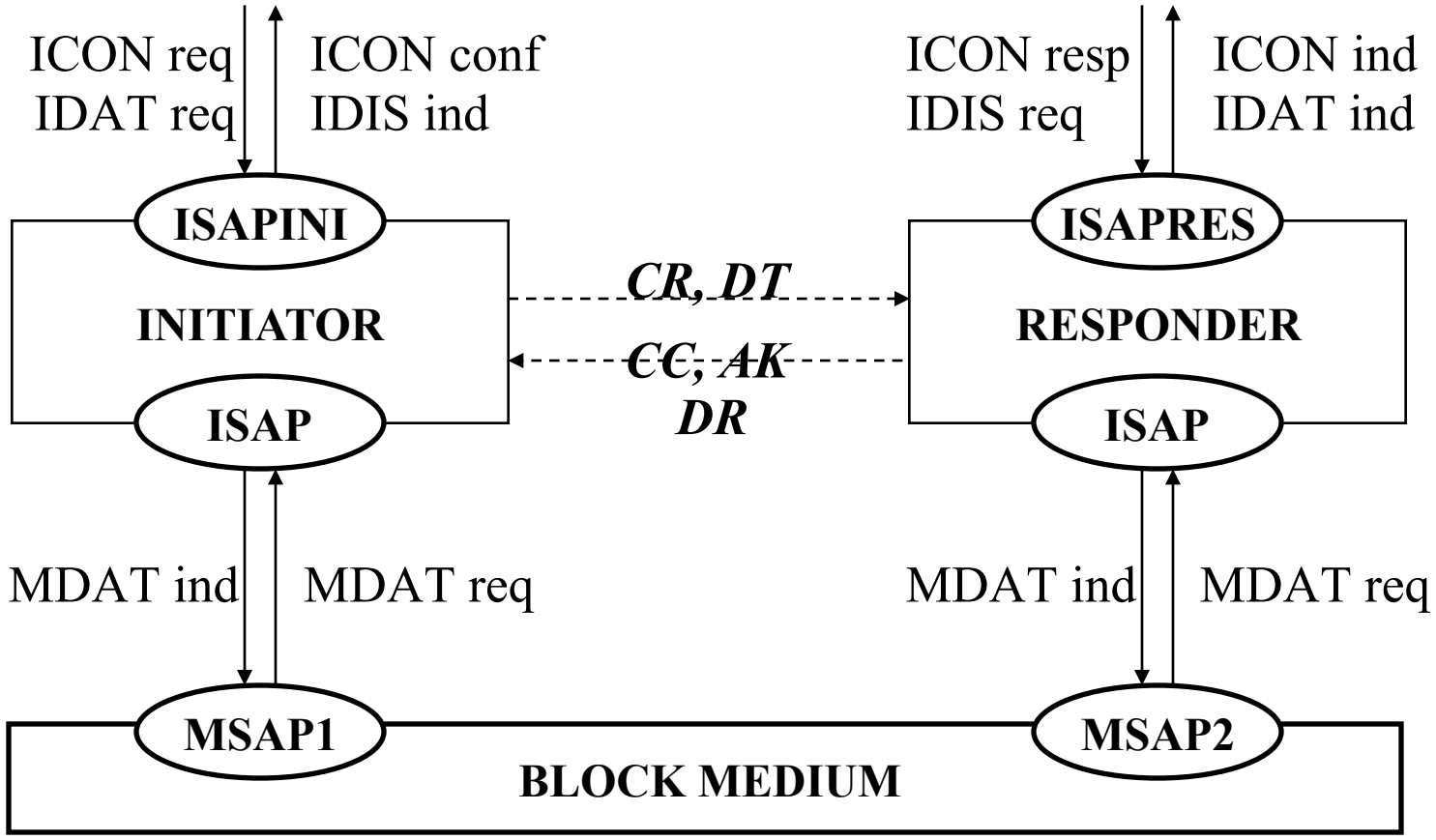
Bontás



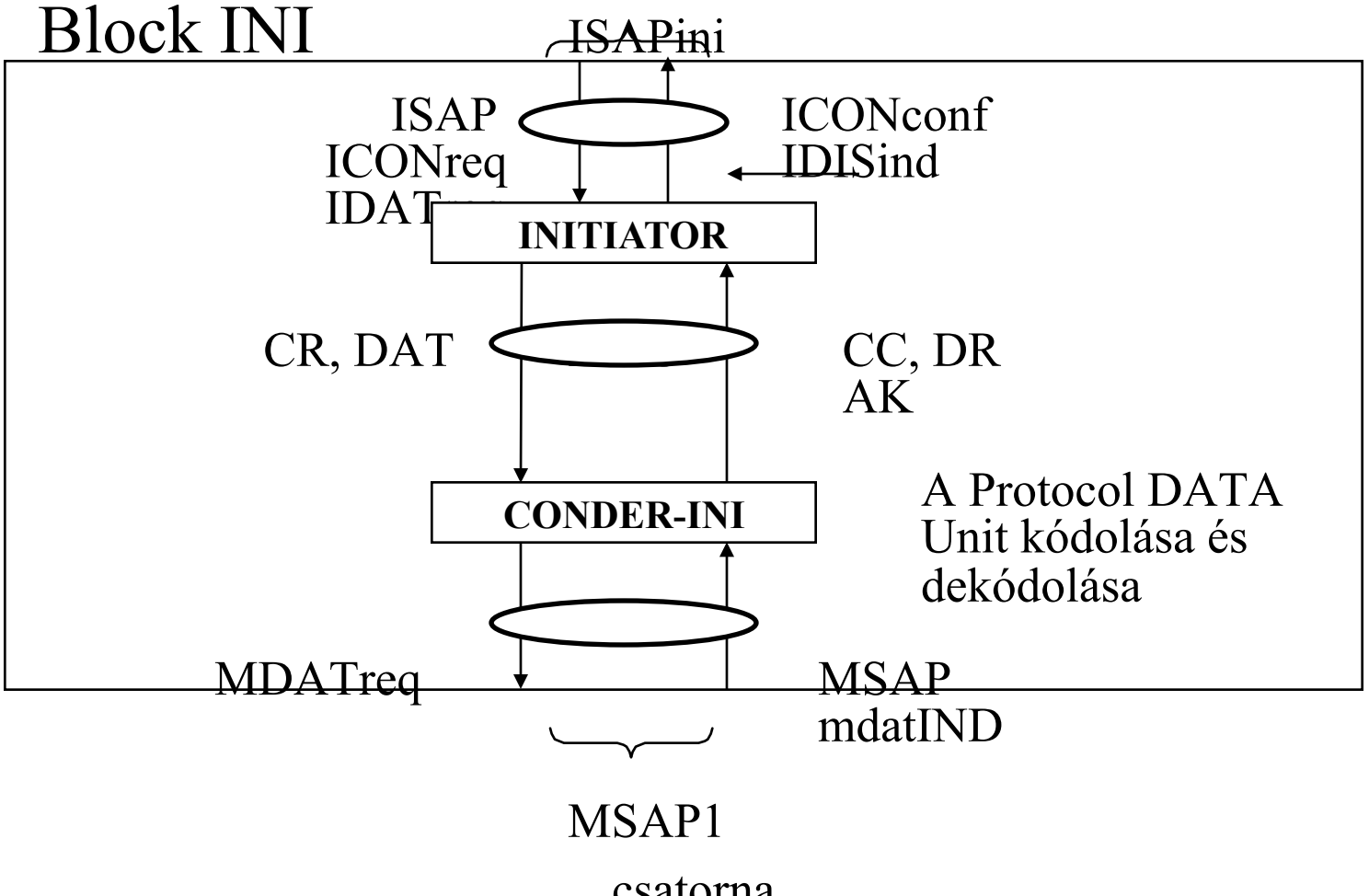
IDIS req	Initiator DISconnect request
IDIS ind	Initiator DISconnect indication
DR	Disconnect request

Formális nyelven történő leírás (1)

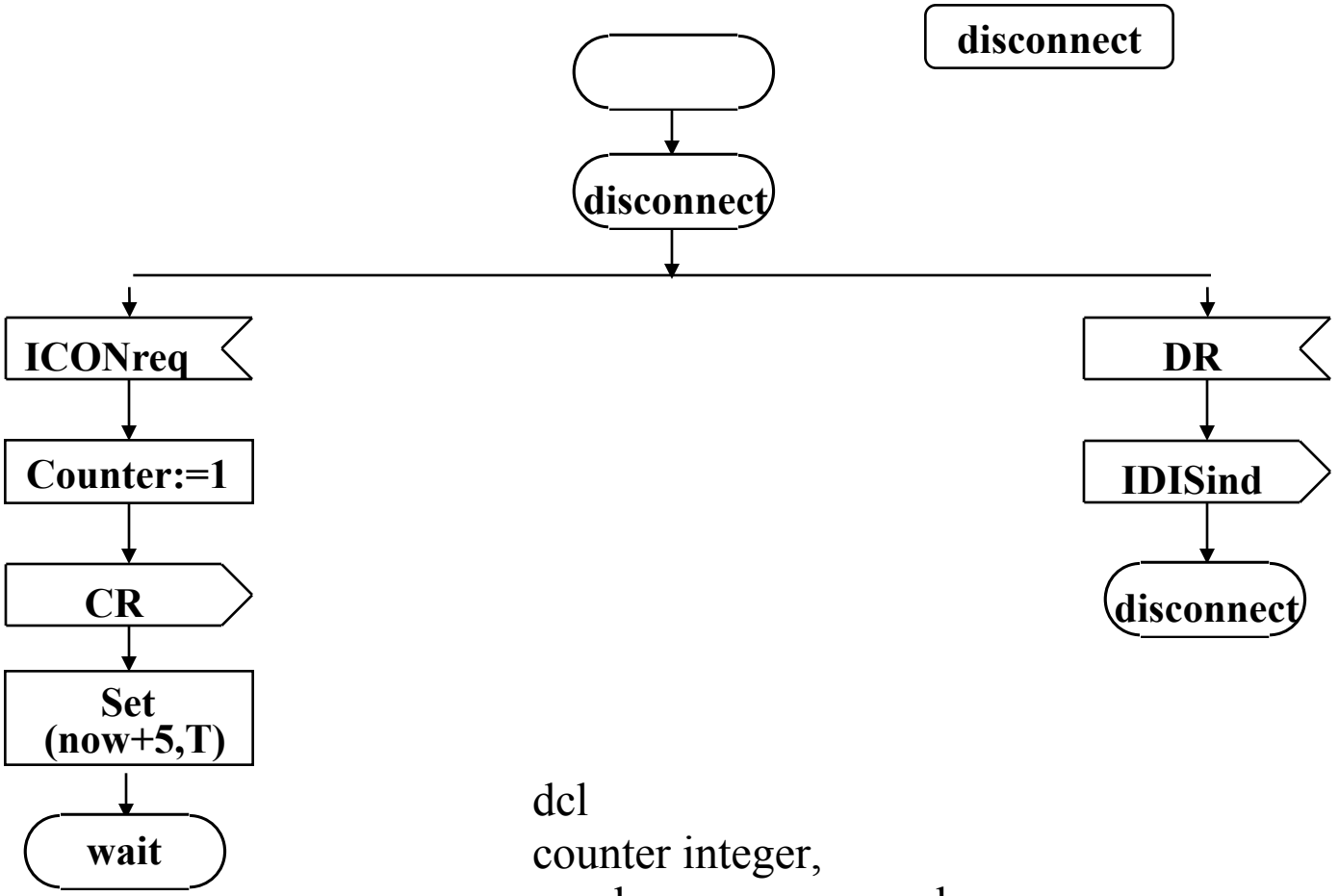
SYSTEM INRES - PROT



User (1)

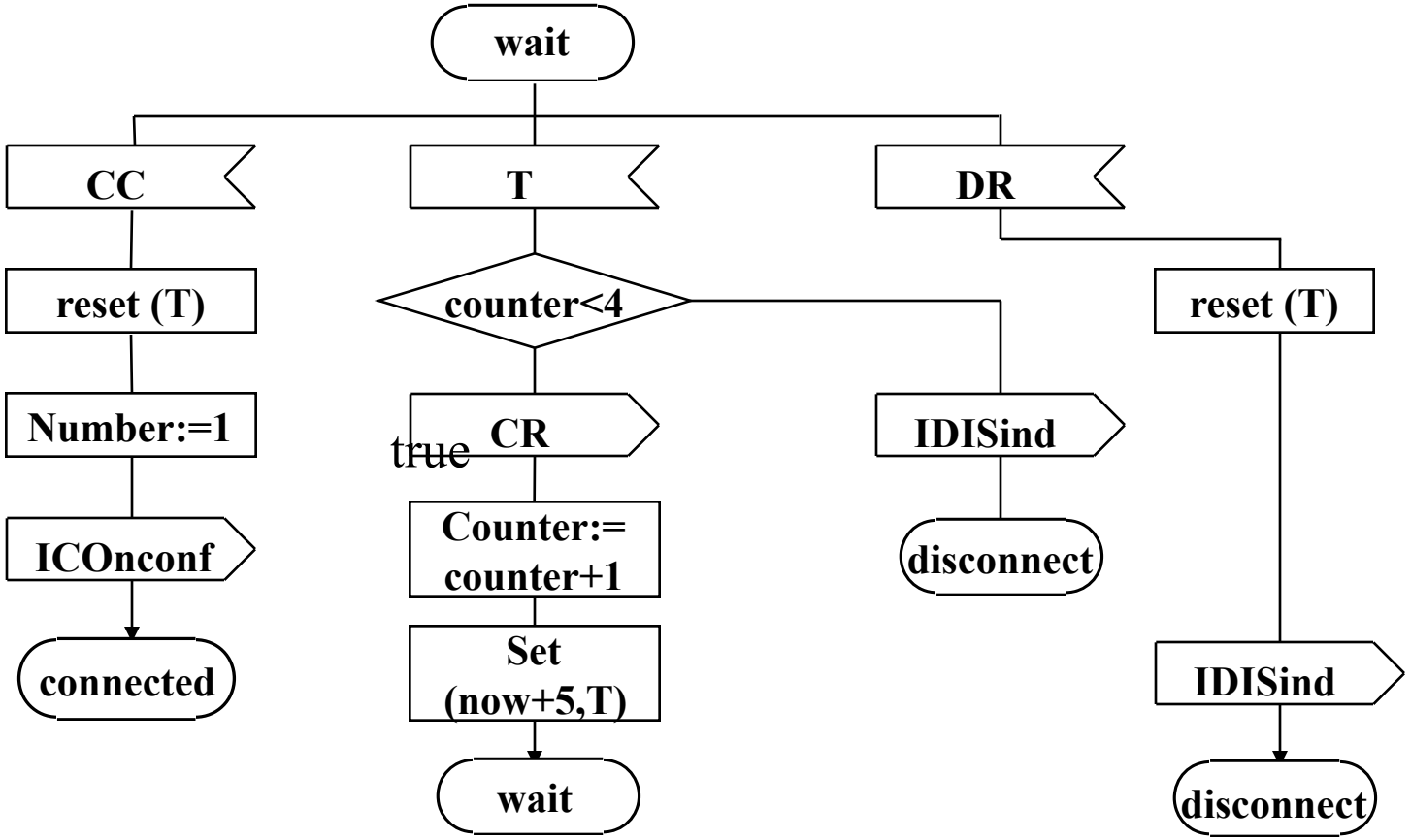

Block INI
ISAPini
ISAP
ICONreq
IDAT
ICONconf
IDISind
INITIATOR
CR, DAT
CC, DR
AK
CONDER-INI
A Protocol DATA Unit kódolása és dekódolása
MDATreq
MSAP
mdatIND
MSAP1
csatorna

Initiator (1)

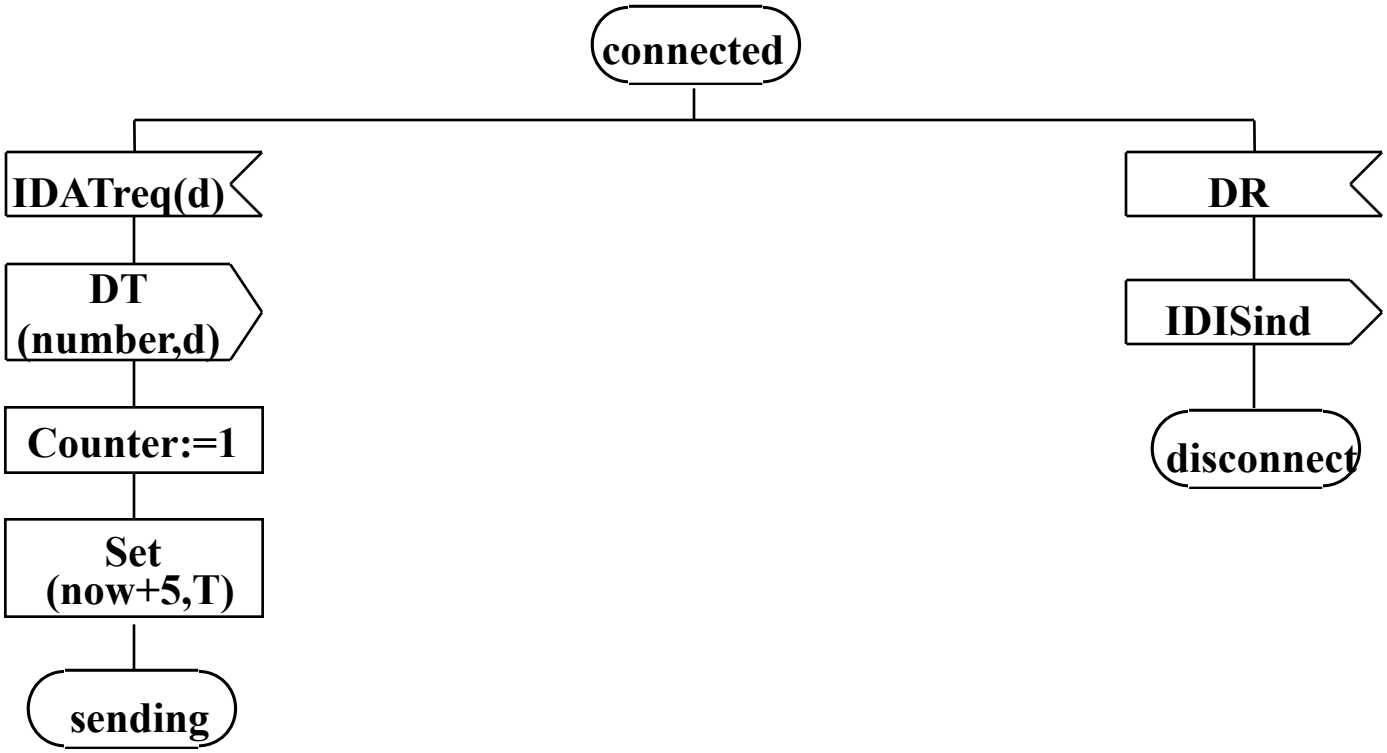



```
dcl
counter integer,
number sequencenumber;
timer T;
```

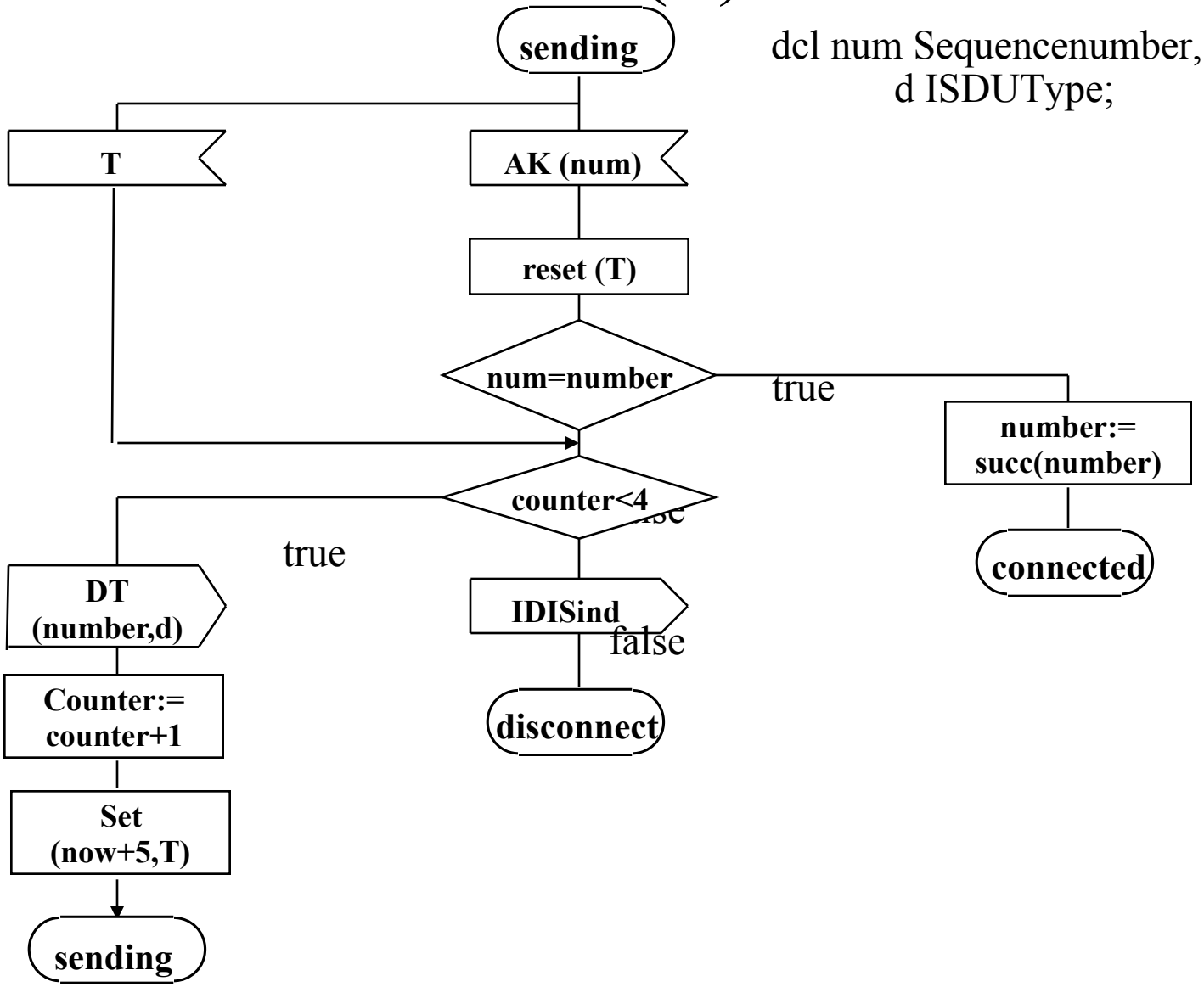
Initiator (2)


wait
CC
T
DR
reset (T)
counter<4
reset (T)
Number:=1
CR
IDISind
true
ICOnconf
Counter:=
counter+1
disconnect
connected
Set
(now+5,T)
IDISind
wait
disconnect

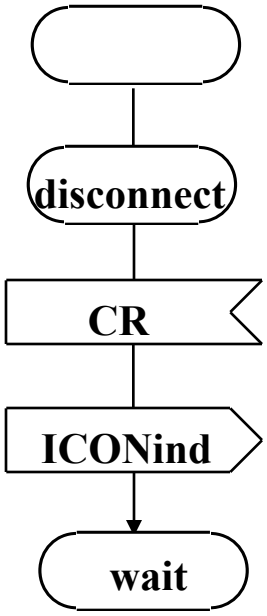
Initiator (3)

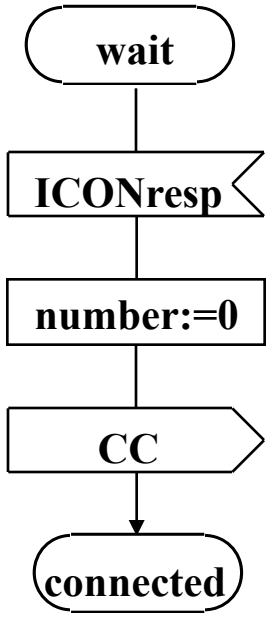

connected
IDATreq(d)
DT
(number,d)
Counter:=1
Set
(now+5,T)
sending
DR
IDISind
disconnect

Initiator (4)


sending
dcl num Sequencenumber,
d ISDUType;
T
AK (num)
reset (T)
num=number
true
number:=
succ(number)
connected
counter<4
true
false
DT
(number,d)
IDISind
false
Counter:=
counter+1
disconnect
Set
(now+5,T)
sending

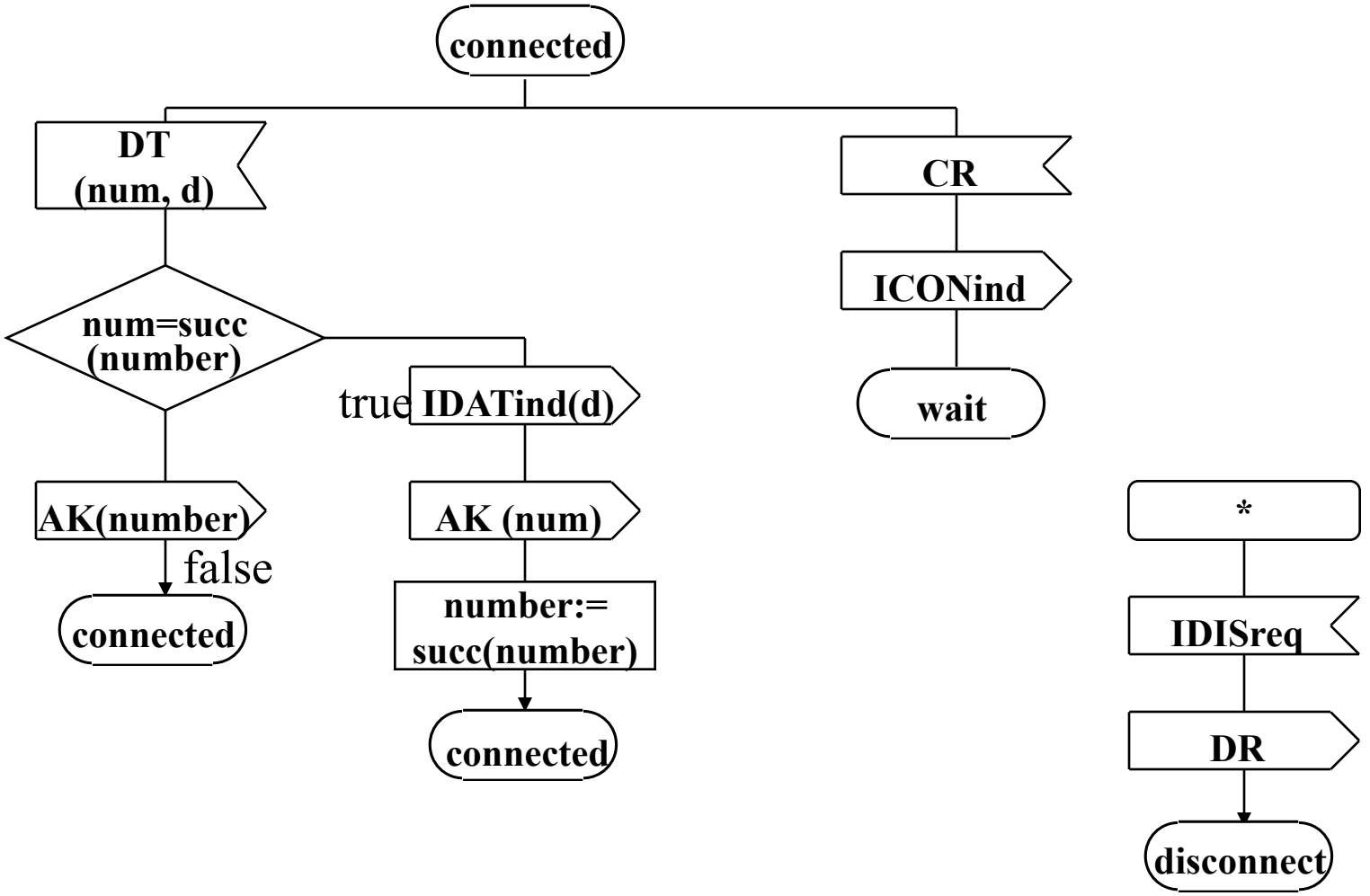
Responder (1)



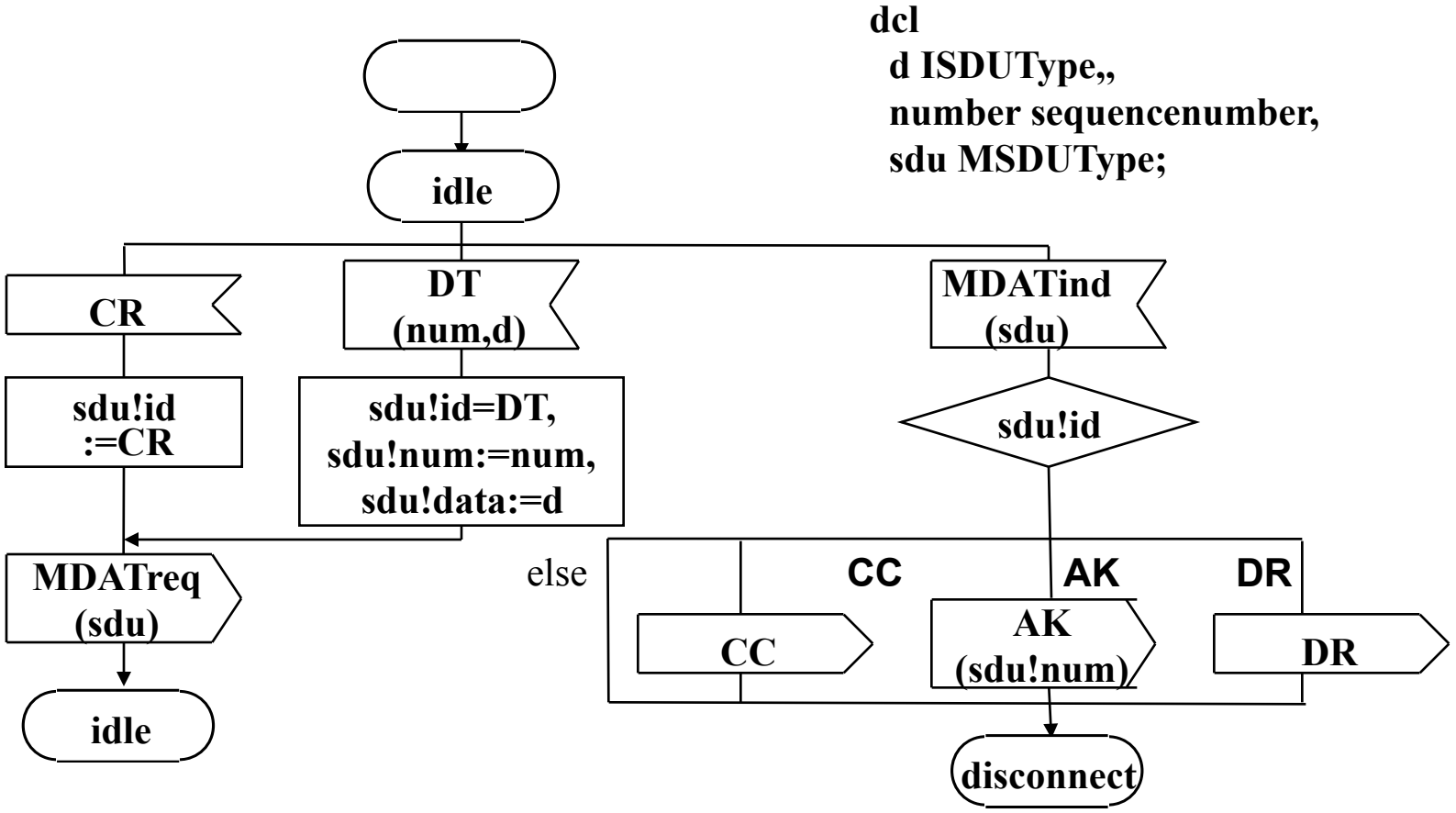



```
dcl
    d ISDUType,
num, number
    Sequencenumber;
```

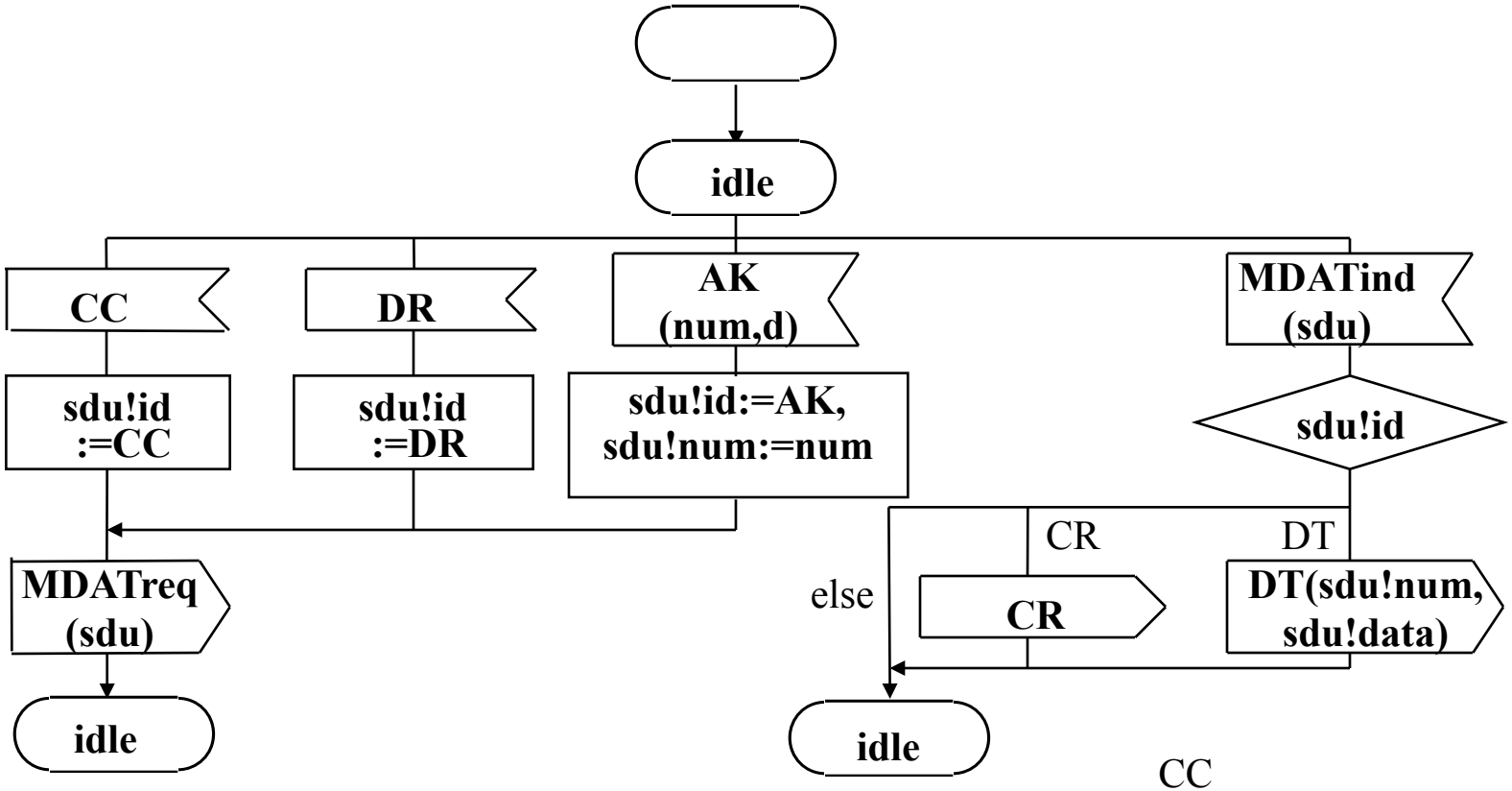
Responder (2)


connected
DT
(num, d)
CR
num=succ
(number)
ICONind
true
IDATind(d)
wait
AK(number)
AK (num)
*
false
IDISreq
number:=
succ(number)
DR
connected
connected
disconnect

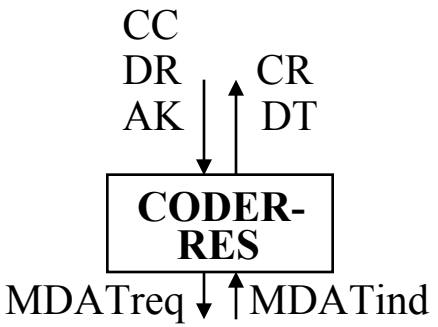
Coder-INI


dcl
d ISDUType,,
number sequencenumber,
sdu MSDUType;
idle
CR
DT
(num,d)
MDATind
(sdu)
sdu!id
:=CR
sdu!id=DT,
sdu!num:=num,
sdu!data:=d
sdu!id
MDATreq
(sdu)
else
CC
AK
DR
CC
AK
(sdu!num)
DR
idle
disconnect
SDU
SDU-id
0/1
SDU adat
(DT esetén)
SORSZÁM

Coder-RES




```
dcl
      num Sequencenumber,
      sdu MSDUType   ;
```



Az SDL 2000

Az SDL 2000 változásainak mozgatórugói

- A nyelv további egyszerűsítése
- Szorosabb kapcsolat az objektum orientált modellezési technikákkal

  → Közeledés az UML-hez

- Hatékonyabbá tenni az SDL-t az implementáció területén. → hatékonyabb automatikus kód generálás közbülső programnyelvre.

  → Közeledés az objekum orientált programnyelvekhez

Az SDL 2000

A nyelv egyszerűsítése

- „Agent” – fogalom: a rendszer, a blokk, és a processz fogalmának összehangolása.
  - Blokk-agent
  - Process-agent
- Az ASN.1 SDL-el való használatának definiálása – a korábbi Z.105-ös szabvány integrálása.
- A jelút fogalmának beleolvasztása a csatorna fogalomba.

Az SDL 2000

Közeledés az UML-hez

- Összetett állapotok (composite states)
  - Önmagukban tartalmaznak viselkedést leíró kiterjesztett állapotgépet
  - UML-ben van hasonló megoldás
- Interfész definiciók
  - Összefoglalja azokat a hatásokat és szolgáltatásokat, amin keresztül egy adott agent-et el lehet érni. Ilyenek:
    - Jel
    - Távoli procedúra hívás
    - Távoli változó elérés
    - Kivétel kezelés

Az SDL 2000

Közeledés az objekum orientált programnyelvekhez

- Új adat modell → Object type
- Kivétel kezelés (exception handling)
- Algoritmusok szöveges definiálásának lehetősége az SDL/GR-en belül.

- Pl. Loop