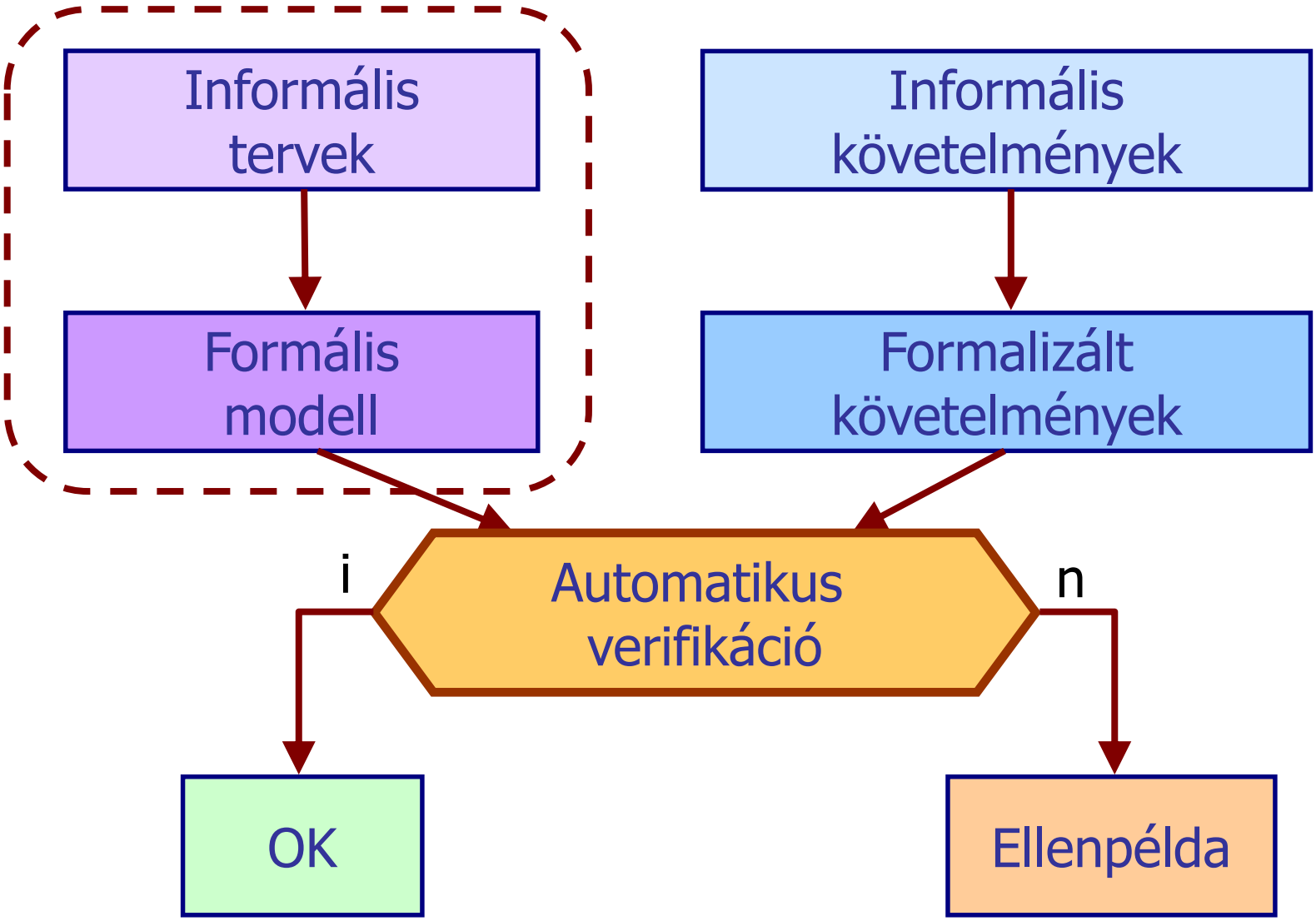
Alapszintű formalizmusok

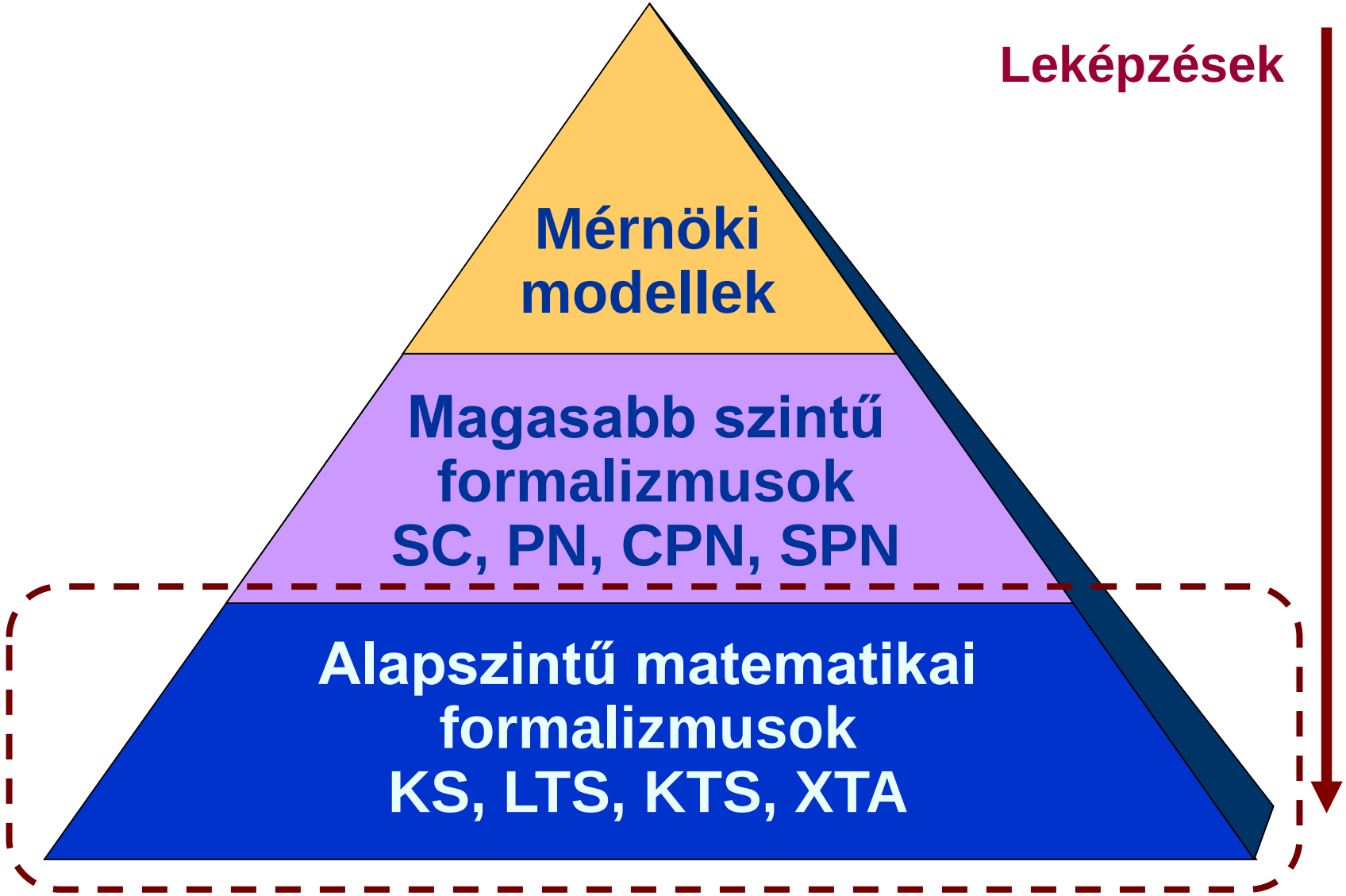
dr. Majzik István

BME Méréstechnika és Információs Rendszerek Tanszék

Mit szeretnénk elérni?



Modellek a formális ellenőrzéshez

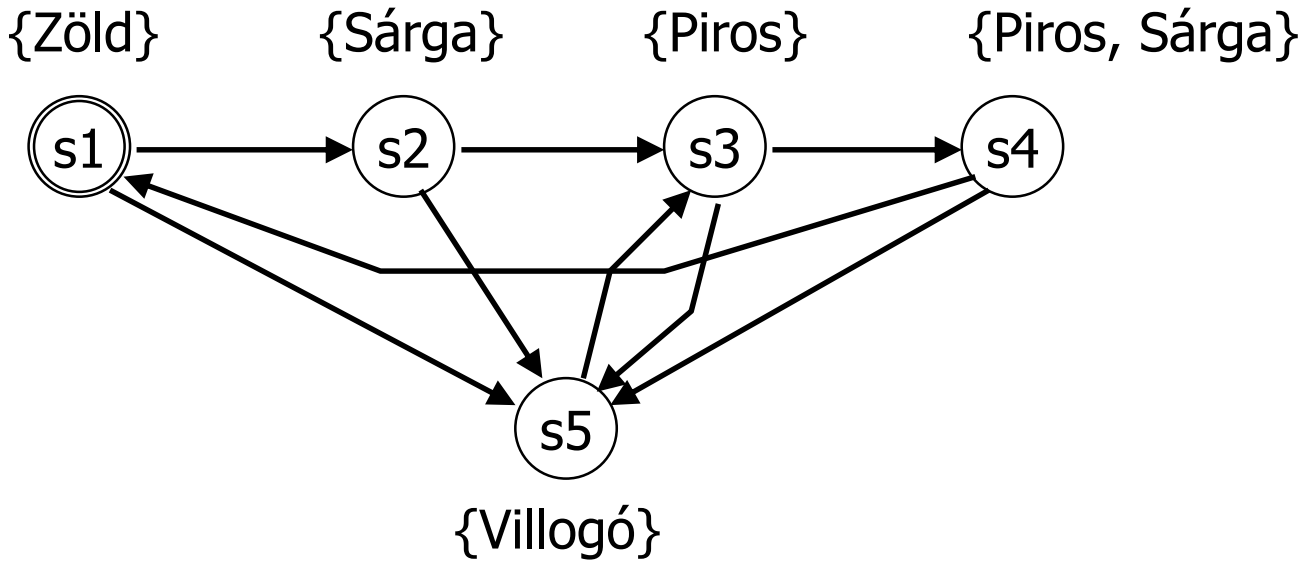


Alapszintű formalizmusok (áttekintés)

- Kripke-struktúrák (KS)
  - Állapotok tulajdonságokkal
  - Állapotátmenetek
- Címkézett tranzíciós rendszerek (LTS)
  - Állapotok
  - Állapotátmenetek tulajdonságokkal
- Kripke tranzíciós rendszerek (KTS)
  - Állapotok tulajdonságokkal
  - Állapotátmenetek tulajdonságokkal
- Kiterjesztett időzített automaták (XTA)
  - Automaták változókkal, óraváltozókkal

1. Kripke-struktúra

- KS (Kripke Structure) használat célja:
  - Viselkedés, algoritmus állapotainak leírása
  - Állapotok lokális tulajdonságai: címkézés atomi kijelentésekkel
- Példa: Közlekedési lámpa vezérlője



Atomi kijelentések: AP={Zöld, Sárga, Piros, Villogó} lámpa kép

Állapotok: S = {s1, s2, s3, s4, s5}

1. Kripke-struktúra

- KS (Kripke Structure) használat célja:
  - Viselkedés, algoritmus állapotainak leírása
  - Állapotok lokális tulajdonságai: címkézés atomi kijelentésekkel
- Szintaxis:

$KS = (S, R, L)$ és AP, ahol

$\mathrm{AP} = \{\mathrm{P},\mathrm{Q},\mathrm{R},\ldots\}$ atomi kijelentések halmaza (domén-specifikus)

$\mathrm{S} = \{\mathrm{s}_1, \mathrm{s}_2, \mathrm{s}_3, \ldots \mathrm{s}_n\}$ állapotok halmaza, s_1 kezdőállapot

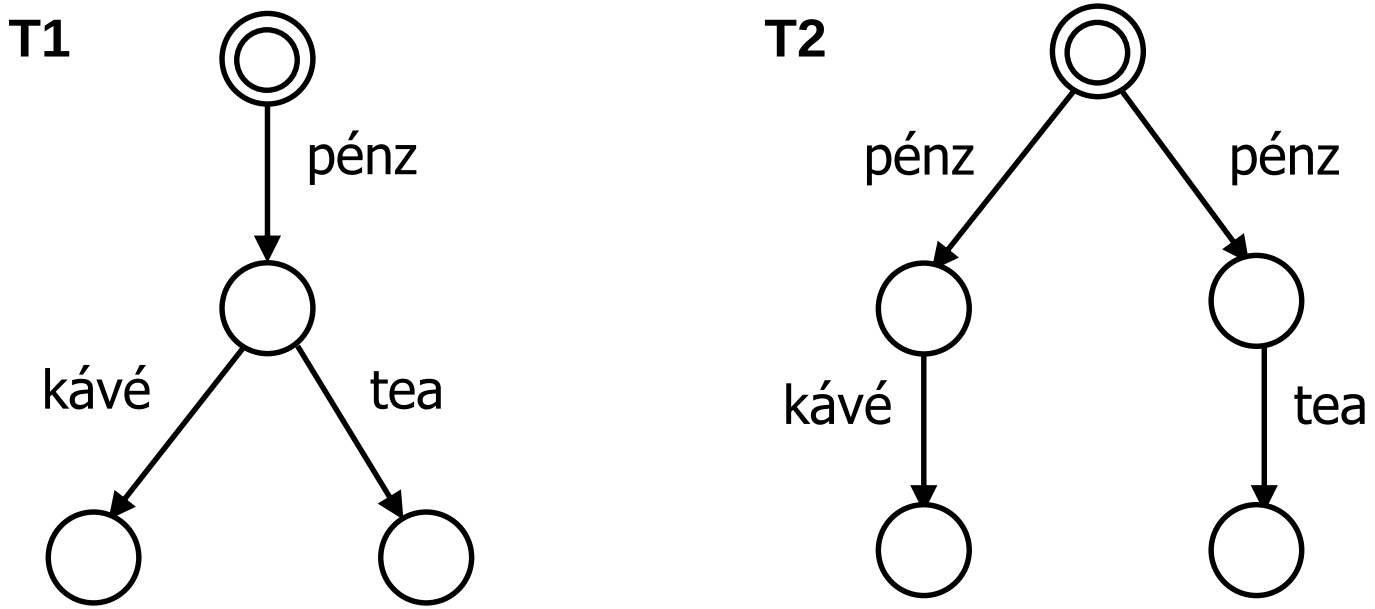
$\mathrm{R} \subseteq \mathrm{S} \times \mathrm{S}$: állapotátmeneti reláció

$\mathrm{L}: \mathrm{S} \to 2^{AP}$ állapotok címkézése atomi kijelentésekkel

- Szemantika:
  - Kezdetben a kezdőállapot aktív
  - Az aktív állapot változhat az állapotátmenetek mentén

2. Címkézett tranzíciós rendszer (LTS)

- LTS (Labeled Transition System) használat célja:
  - Viselkedés, kommunikáció állapotátmeneteinek leírása
  - Állapotátmenetek lokális tulajdonságai: címkézés egy-egy akcióval
- Példa: Italautomaták



Akciók: Act = {pénz, kávé, tea} interakciók a felhasználóval

2. Címkézett tranzíciós rendszer (LTS)

- LTS (Labeled Transition System) használat célja:
  - Viselkedés, kommunikáció állapotátmeneteinek leírása
  - Állapotátmenetek lokális tulajdonságai: címkézés egy-egy akcióval
- Szintaxis:

$LTS = (S, Act, \rightarrow)$, ahol

$S = \{s_1, s_2, \ldots s_n\}$ állapotok halmaza, s_1 kezdőállapot

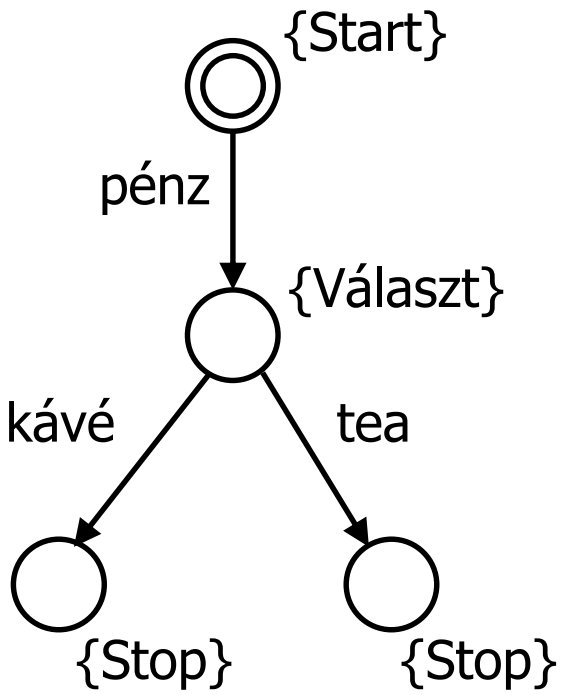
$Act = \{a, b, c, \ldots\}$ akciók halmaza (domén-specifikus)

$\rightarrow \subseteq S \times Act \times S$ címkézett állapotátmenetek, pl. $s_1 \xrightarrow{a} s_2$

- Szemantika:
  - Kezdetben a kezdőállapot aktív
  - Az aktív állapot változhat az állapotátmenetek mentén

3. Kripke tranzíciós rendszer (KTS)

- KTS (Kripke Transition System) használat célja:
  - Viselkedés, protokoll állapotainak és állapotátmeneteinek leírása
  - Állapotok lokális tulajdonságai: címkézés atomi kijelentésekkel
  - Állapotátmenetek lokális tulajdonságai: címkézés egy-egy akcióval
- Példa: Italautomata



Akciók: Interakció a felhasználóval

Act = {pénz, kávé, tea}

Atomi kijelentések: Állapot kijelzés

AP = {Start, Választ, Stop}

3. Kripke tranzíciós rendszer (KTS)

- KTS (Kripke Transition System) használat célja:
  - Viselkedés, protokoll állapotainak és állapotátmeneteinek leírása
  - Állapotok lokális tulajdonságai: címkézés atomi kijelentésekkel
  - Állapotátmenetek lokális tulajdonságai: címkézés egy-egy akcióval
- Szintaxis:

$KTS = (S, \rightarrow, L)$ és AP, Act, ahol

$AP = \{P, Q, R, ...\}$ atomi kijelentések és $Act = \{a, b, c, ...\}$ akciók halmaza

$S = \{s_1, s_2, s_3, ...s_n\}$ állapotok halmaza, s_1 kezdőállapot

$\rightarrow \subseteq S \times Act \times S$ állapotátmeneti reláció (akciókkal címkézve)

$L: S \rightarrow 2^{AP}$ állapotok címkézése atomi kijelentésekkel

Szemantika:

- Kezdetben a kezdőállapot aktív
- Az aktív állapot változhat az állapotátmenetek mentén

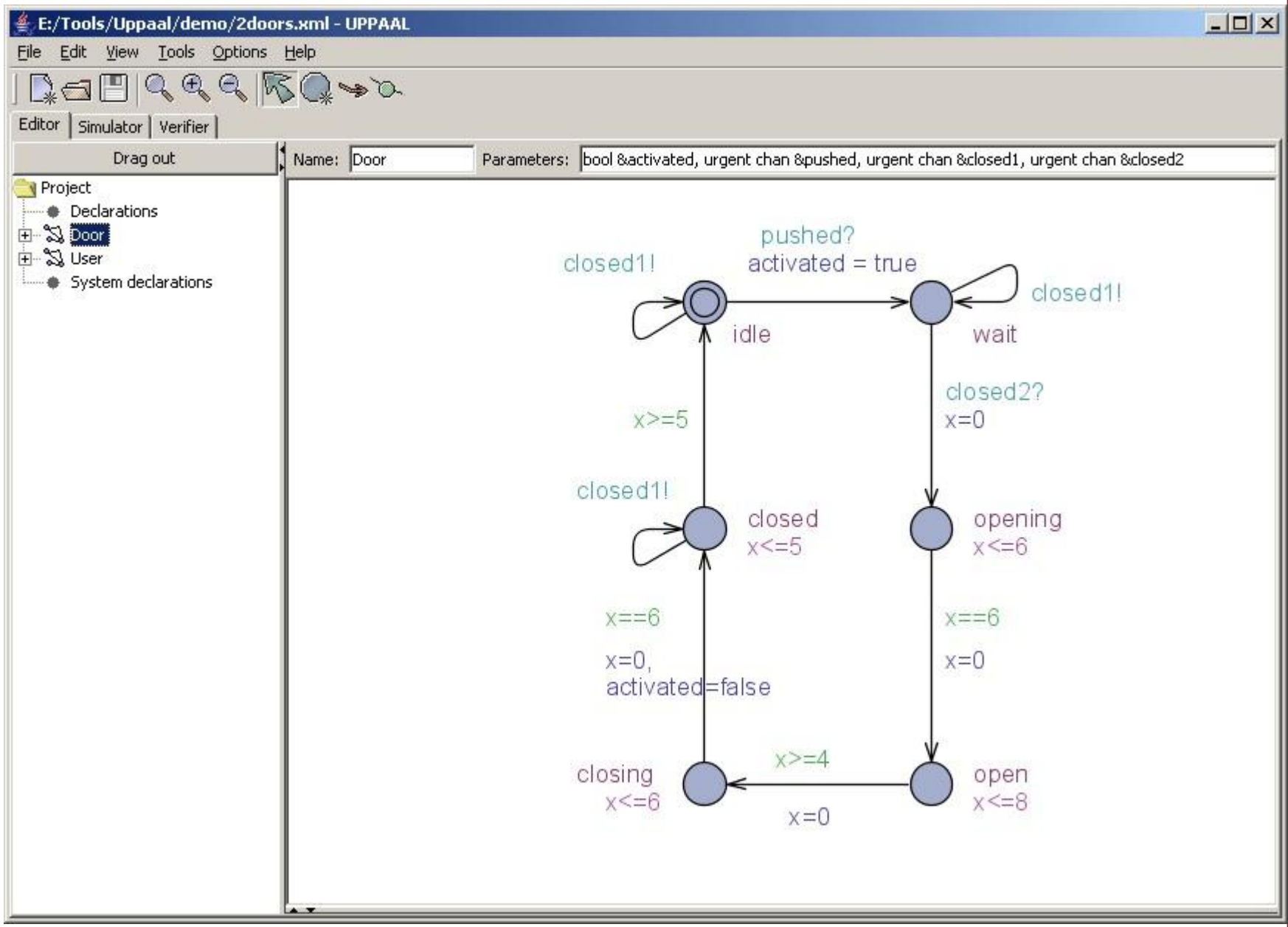
Kiterjesztett időzített automaták és használatuk az UPPAAL eszközben

Extended Timed Automata (XTA)
Network of Timed Automata

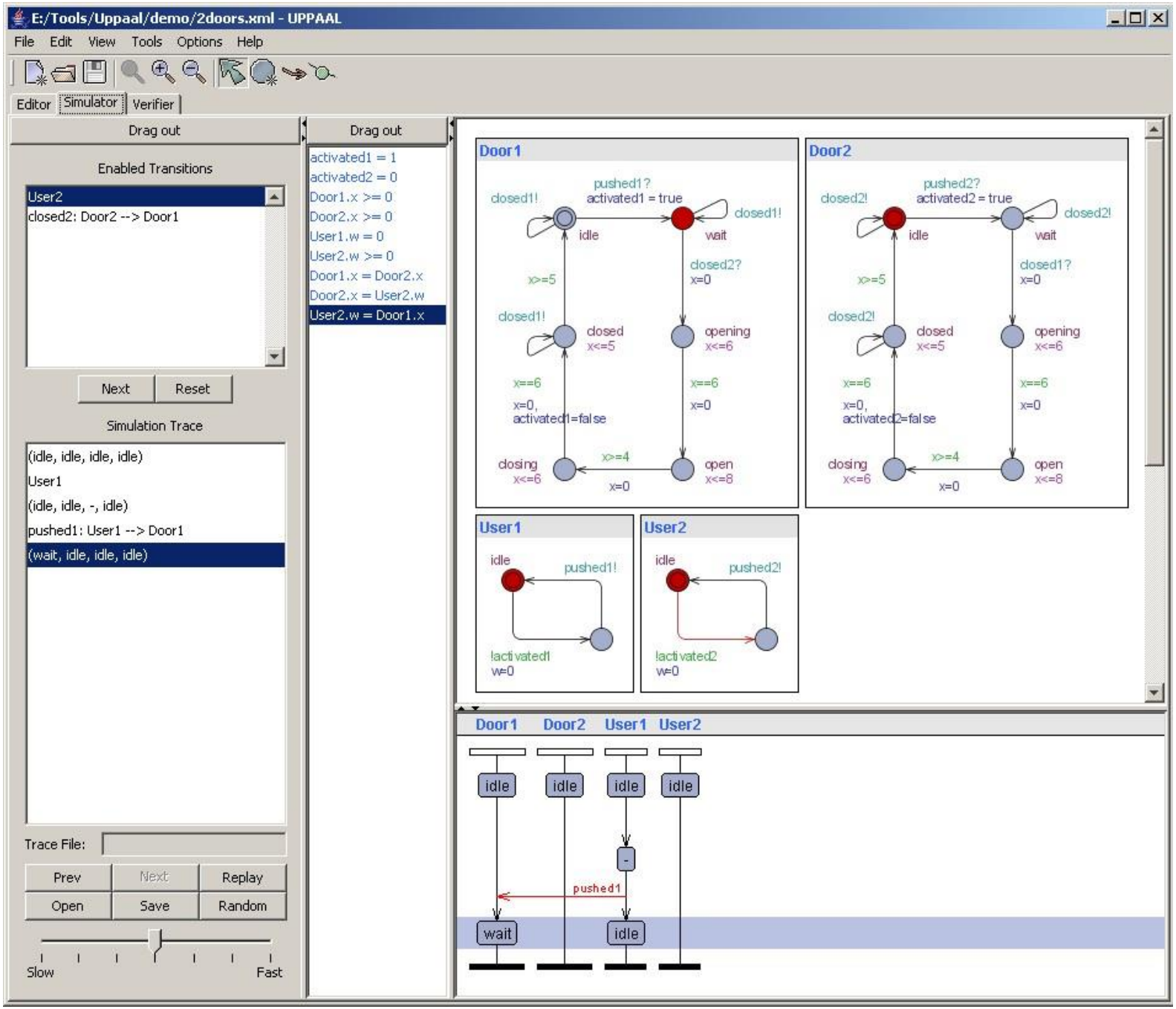
Az UPPAAL eszköz

- Fejlesztése (1999-):
  - Uppsala University, Svédország; Aalborg University, Dánia
- Az eszköz funkciói
  - Modellezés időzített automatákkal
  - Szimuláció
  - Verifikáció (modellellenőrzés)
- Kiadások (információk, letöltés, példák)
  - Akadémiai (ingyenes): http://www.uppaal.org/
  - Kereskedelmi: http://www.uppaal.com/
- Kapcsolódó eszközök
  - UPPAAL TRON: On-line tesztelés
  - UPPAAL TIGA: Kiterjesztés játékautomatákra
  - …

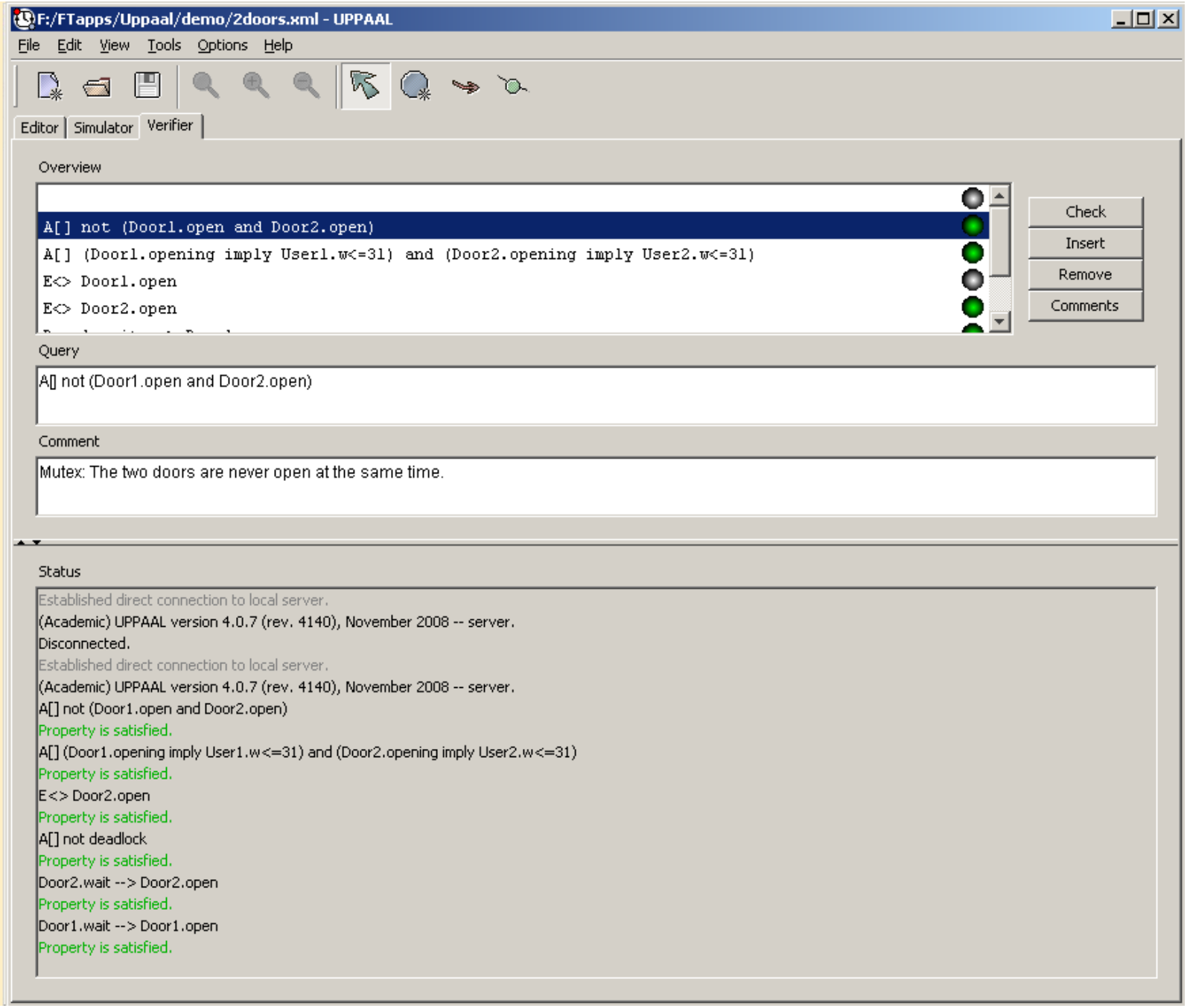
Automata modell szerkesztő



Szimulátor


E:/Tools/Uppaal/demo/2doors.xml - UPPAAL
File Edit View Tools Options Help
Editor Simulator Verifier
Drag out
Enabled Transitions
User2
closed2: Door2 --> Door1
Next Reset
Simulation Trace
(idle, idle, idle, idle)
User1
(idle, idle, -, idle)
pushed1: User1 --> Door1
(wait, idle, idle, idle)
Trace File:
Prev Next Replay
Open Save Random
Slow Fast
Drag out
activated1 = 1
activated2 = 0
Door1.x >= 0
Door2.x >= 0
User1.w = 0
User2.w >= 0
Door1.x = Door2.x
Door2.x = User2.w
User2.w = Door1.x
Door1
closed1!
pushed1?
activated1 = true
idle
wait
x>=5
closed2?
x=0
closed
x<=5
opening
x<=6
x==6
x=0,
activated1=false
closing
x<=6
x>=4
open
x<=8
Door2
closed2!
pushed2?
activated2 = true
closed1?
activated2=false
User1
pushed1!
!activated1
w=0
User2
pushed2!
!activated2
Door1 Door2 User1 User2
idle
-
pushed1
wait

Verifikáció



Automaták és változók

- Cél: Állapot alapú viselkedés modellezése
- Alap formalizmus: Véges állapotú automata (FSM)
  - Állapotok (névvel hivatkozhatók)
  - Állapotátmenetek
- Nyelvi kiterjesztés: Egész értékű változók használata
  - Változók deklarálhatók típussal (értéktartománnyal)
  - Konstansok definiálhatók
  - Egész aritmetika használható
  - Az FSM „állapot” itt csak „vezérlési hely” (változó is az állapot része)
- Változók használata állapotátmeneteken:
  - Őrfeltétel hozzárendelése: A változókon kiértékelhető predikátum
    - Az átmenet bekövetkezéséhez igaz kell legyen
  - Akció hozzárendelése: Értékadás változóknak
    - Az átmenet bekövetkezésekor végrehajtódik

Példa: Automata változókkal

Deklarációk:

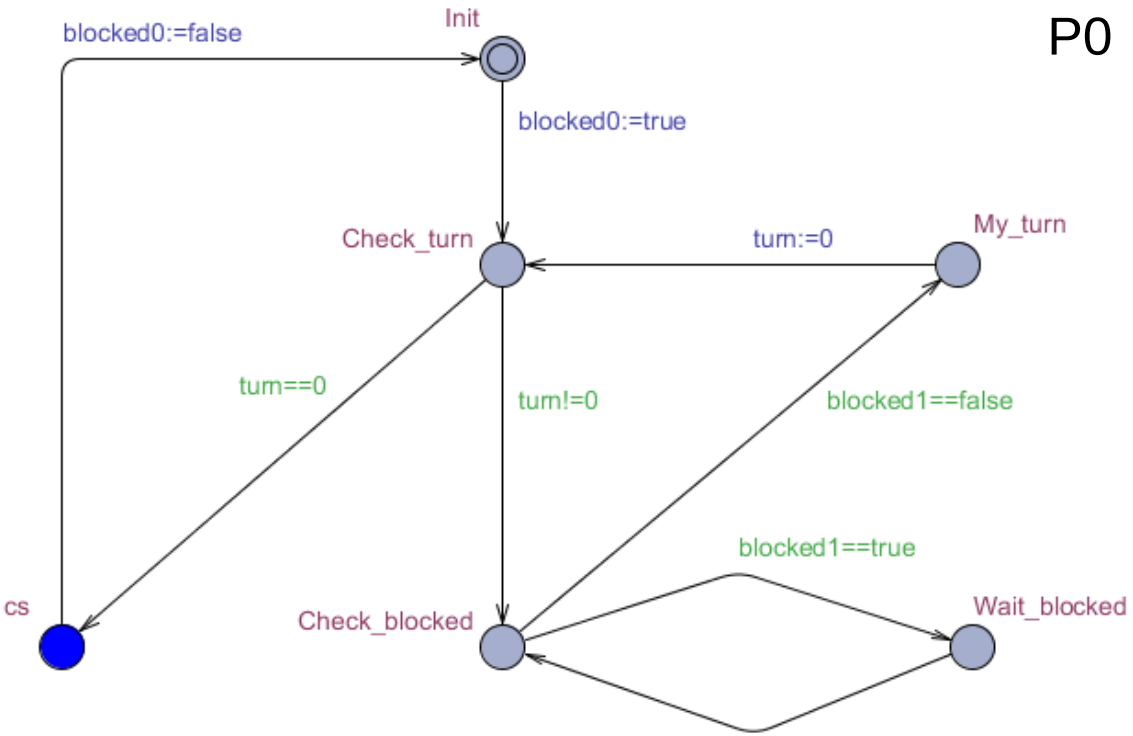
```
bool blocked0=false;
bool blocked1=false;
int[0,1] turn=0;
```

A P0 automata pszeudokódja és modellje:

P0

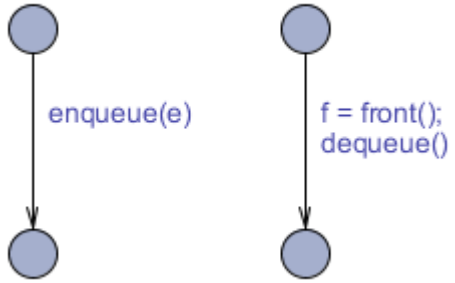
```
while (true) {
   blocked0 := true;
   while (turn!=0) {
        while (blocked1==true) {
               skip;
        }
        turn := 0;
   }
   // Critical section (cs)
   blocked0 := false;
   // Do other things
}
```

P0



Függvények alkalmazása átmenetek akcióiban

Példa: Lista kezelése




```
const int N = 6; // Number of elements
int list[N];  // Array with N integers
int[0,N-1] last;  // Index of the last

// Put an element to the end of the list
// (here: overflow not checked)
void enqueue(int element)
{
        list[last++] := element;
}
```

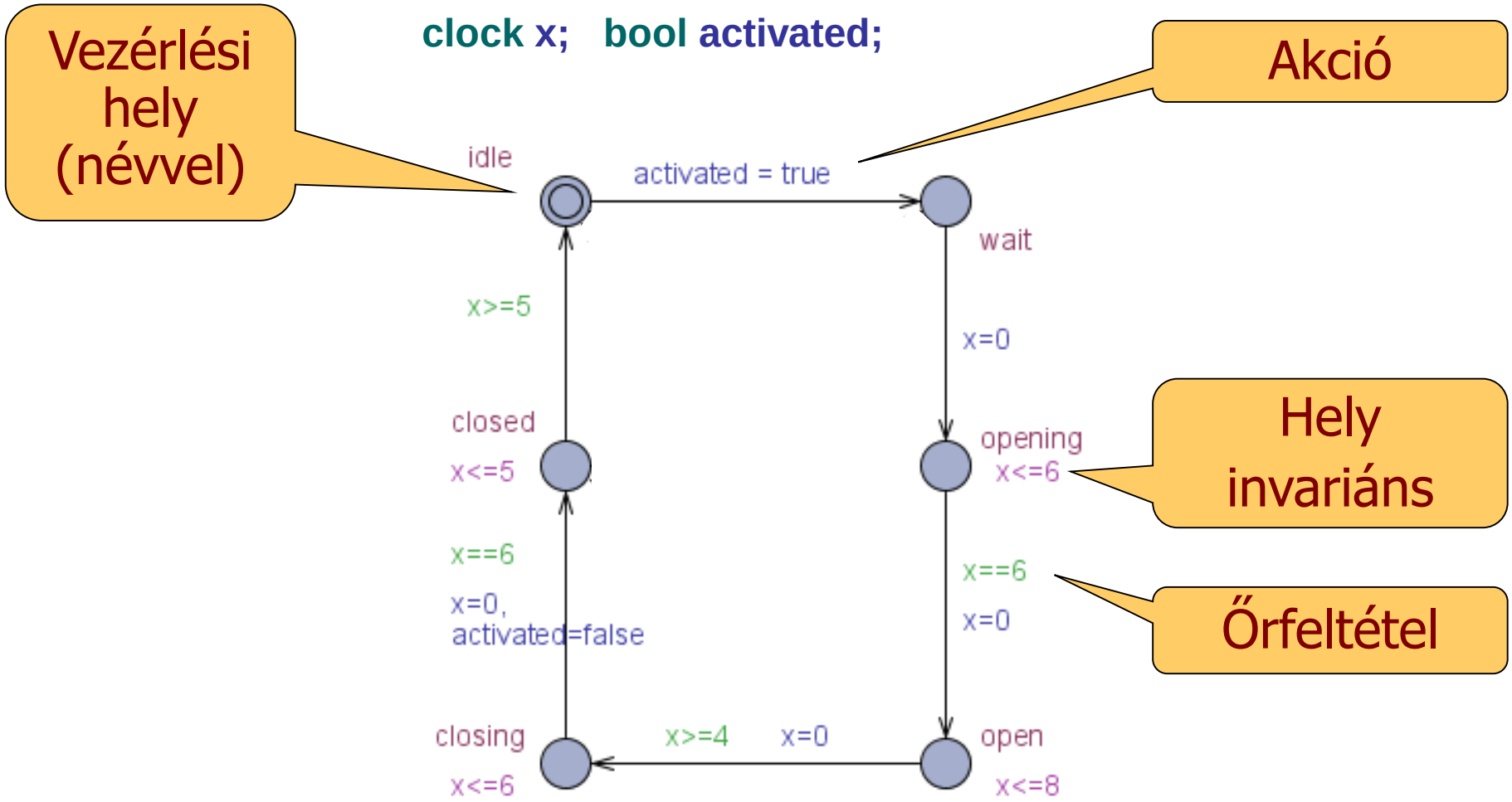
```
// Return the front element of the list
int front()
{
   return list[0];
}

// Remove the front element of the list
void dequeue()
{
        int i := 0;
        while (i < last)
        {
                list[i] := list[i + 1];
                i++;
        }
        list[i] := 0;
        last := last-1;
}
```

Kiterjesztések óraváltozókkal

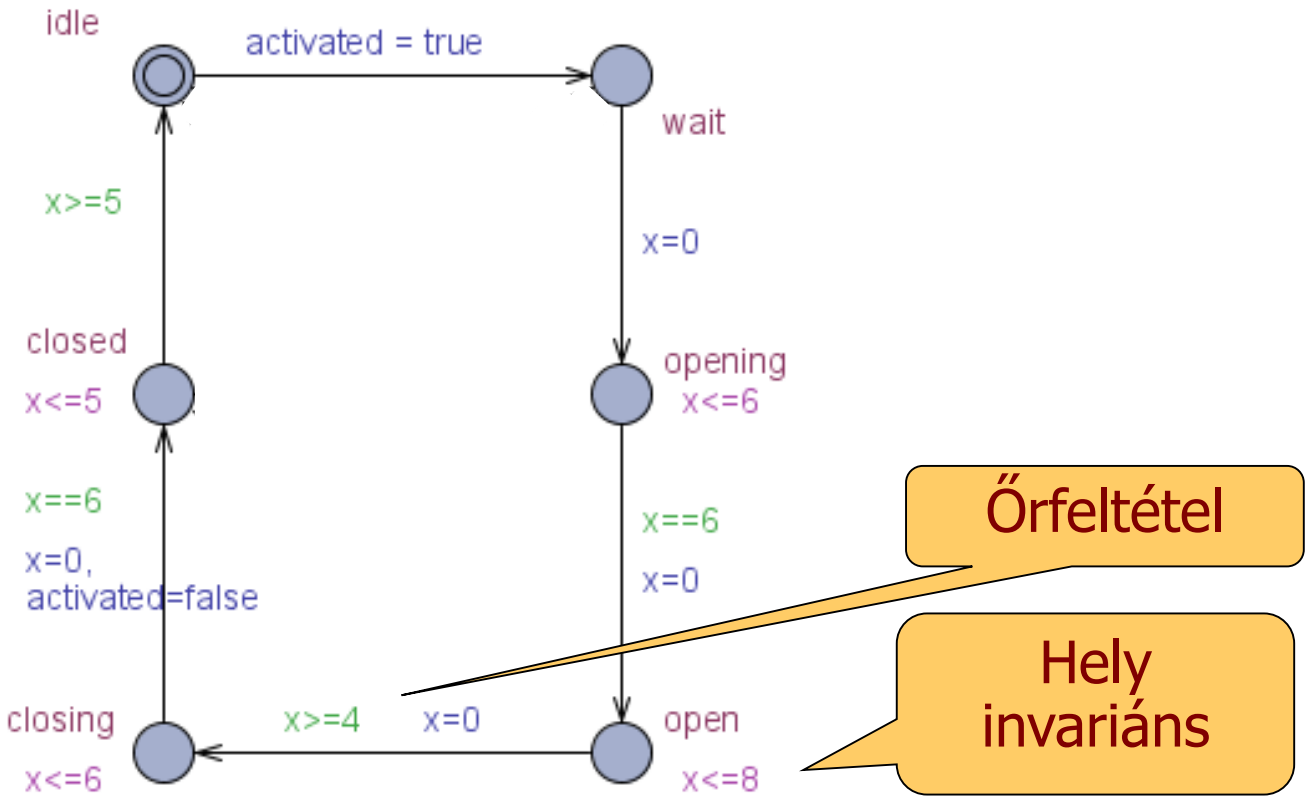
- Cél: Valósidejű viselkedés modellezése
  - Idő telik az állapotokban (pl. bemenetre vár)
  - Az idő függvényében változhat a viselkedés (pl. timeout)
  - Relatív időmérés megoldás: Időzítő, ennek leolvasása, nullázása
- Nyelvi kiterjesztés: Óraváltozók
  - Értékük azonos gyakorisággal automatikusan nő (konkurens órák)
- Óraváltozók használata átmeneteken:
  - Őrfeltételek: Predikátumok óraváltozókon és konstansokon (pl. adott érték elérése, meghaladása)
  - Akciók: Óraváltozók nullázása (resetelése), egymástól függetlenül
- Óraváltozók használata vezérlési helyeken:
  - Hely invariáns: Predikátum óraváltozókon és konstansokon; az adott vezérlési hely addig lehet aktív, amíg a hely invariáns teljesül (igaz az értéke)

Példa: Időzített automata (az UPPAAL eszközben)



Az invariánsok és őrfeltételek szerepe

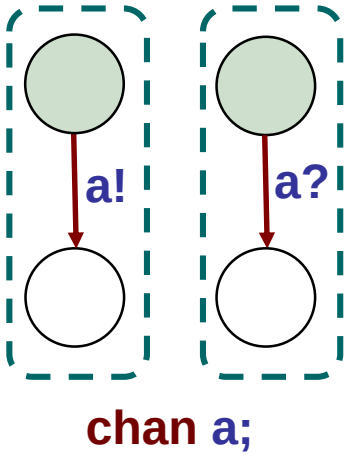
clock x;



Az open állapot elhagyásakor a [4, 8] tartományban lehet az x óra értéke

Kiterjesztések elosztott rendszerekhez

- Cél: Együttműködő automaták hálózatának modellezése
  - Interakció: Szinkronizált, együttlépő átmenetek (randevú)
    - Szinkronizáció kezdeményezője: szinkron üzenet küldője (fogadóra vár)
    - Szinkronizáció résztvevője: szinkron üzenet fogadója (küldőre vár)
  - Ezzel az alapelemmel más jellegű interakciók is leírhatók
- Nyelvi kiterjesztés: Szinkronizált akciók
  - Csatornák definiálása (szinkron csatorna: chan)
  - Üzenetküldés: ! operátor a csatornára
    Üzenetfogadás: ? operátor a csatornára
    - Pl. az a nevű csatorna esetén a! és a? akciók



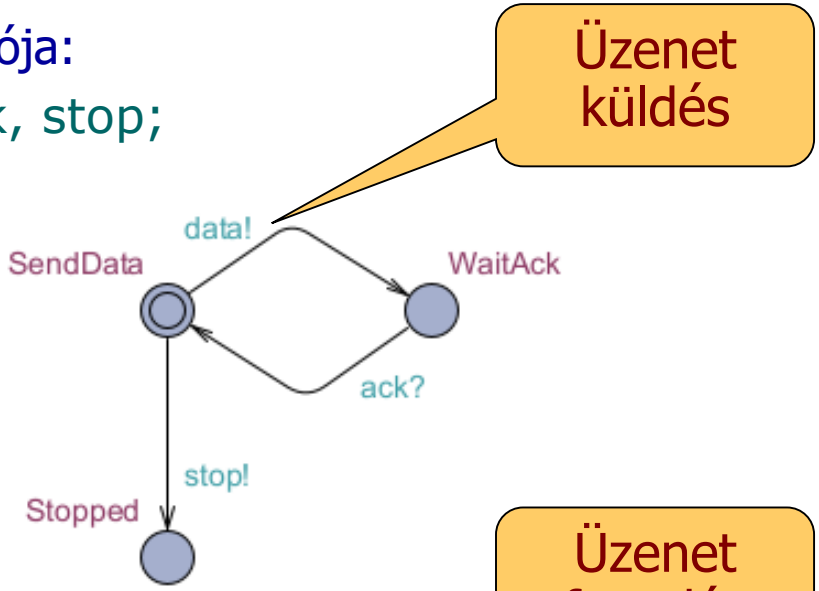
- Kiterjesztés: Csatornatömbök kezelése
  - Csatornatömb deklarálás: chan a[]
  - Műveletek csatornatömb elemein id változóval indexelve: a[id]! illetve a[id]? akciók

Példa: Egy egyszerű üzenetküldő és -fogadó processz

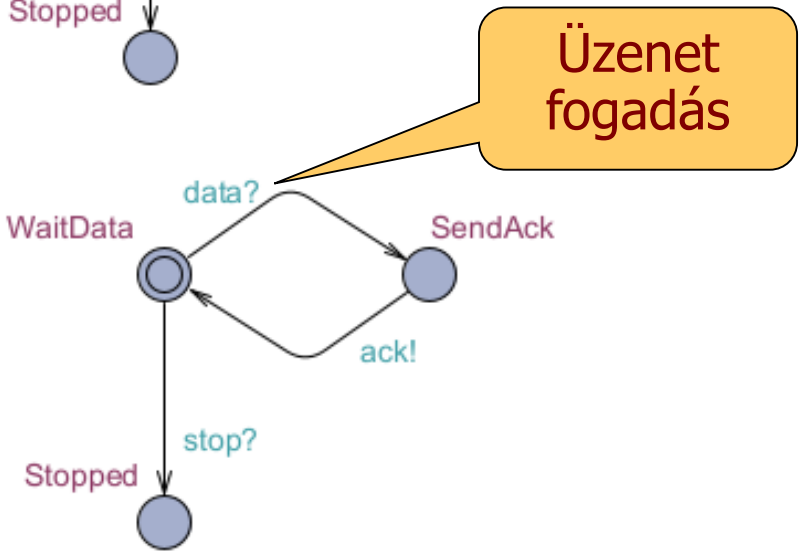
Csatornák deklarációja:

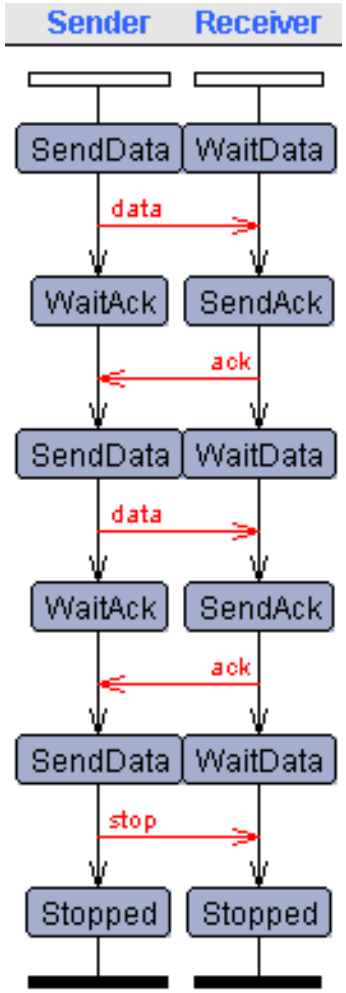
chan data, ack, stop;

Sender:



Receiver:

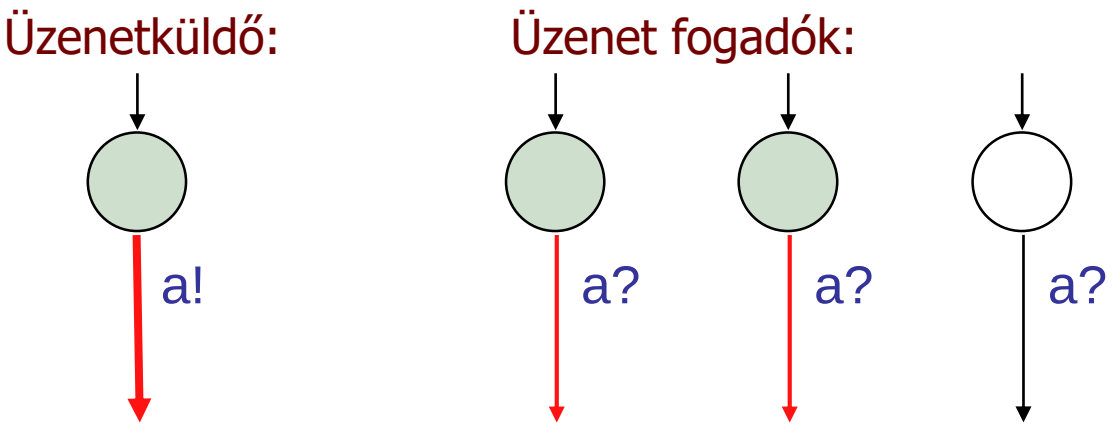




További lehetőségek: Broadcast csatorna

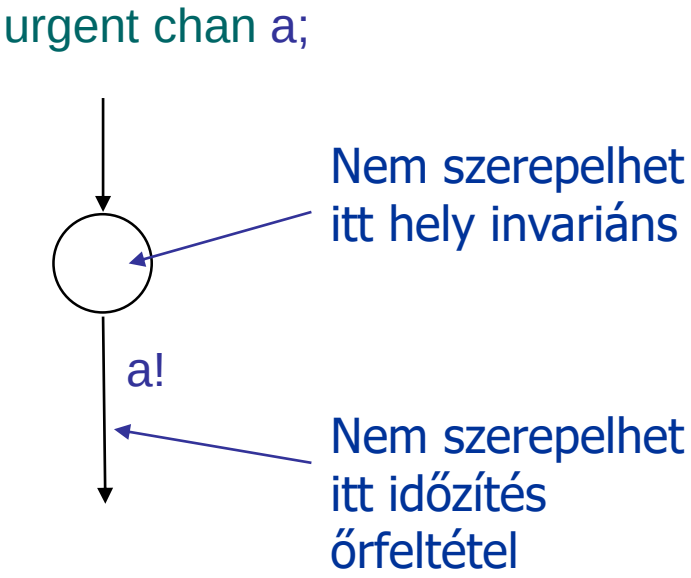
- Broadcast csatorna: 1→N kommunikáció
  - „Üzenetküldés” feltétel nélkül megtörténik
    - Nem kell fogadó készenlétére (randevúra) várni
  - Minden „üzenetfogadásra kész” partner erre szinkronizálódik
    - Üzenetfogadáshoz szükséges az üzenetküldés
  - Használati feltétel: Nem szerepelhet őrfeltétel a broadcast csatornára hivatkozó üzenetfogadó átmeneten

broadcast chan a;



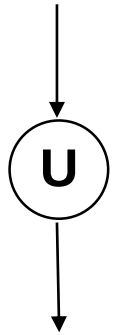
További lehetőségek: Sürgős (urgent) csatorna

- Urgent csatorna: Nem enged késleltetést
  - Késleltetés nélkül, azonnal végrehajtandó szinkronizáció
    - Előtte más átmenetek azonnali végrehajtása megtörténhet
  - Korlátozott az őrfeltételek és invariánsok használata:

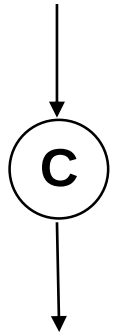


További lehetőségek: Speciális állapotok

- Urgent állapot: késleltetés korlátozása
  - Nem telhet idő az adott állapotban
  - Ekvivalens modell:
    - Óraváltozó bevezetése: clock x;
    - Minden bemenő élen resetelve: x:=0
    - Hely invariáns hozzárendelése: x<=0

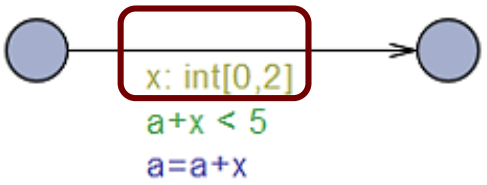


- Committed állapot: átmenetek egybefogása
  - Bemenő és kimenő átmenet egy atomi műveletként végrehajtva
  - A bemenő és kimenő átmenetek végrehajtása között más automata normál átmenete nem hajtható végre (legfeljebb committed állapotból induló átmenete)
    - Valóságos konkurens rendszerekben nehéz megvalósítani

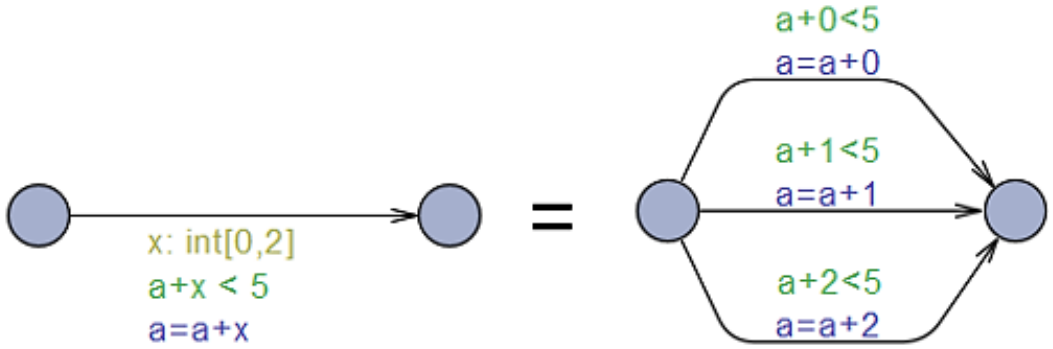


Nemdeterminisztikus választás: szelekció

- Használat: Átmeneten
  - Select konstrukció: változó és típus megadása
    Pl. itt x változóval és típussal: x: int[0,2]

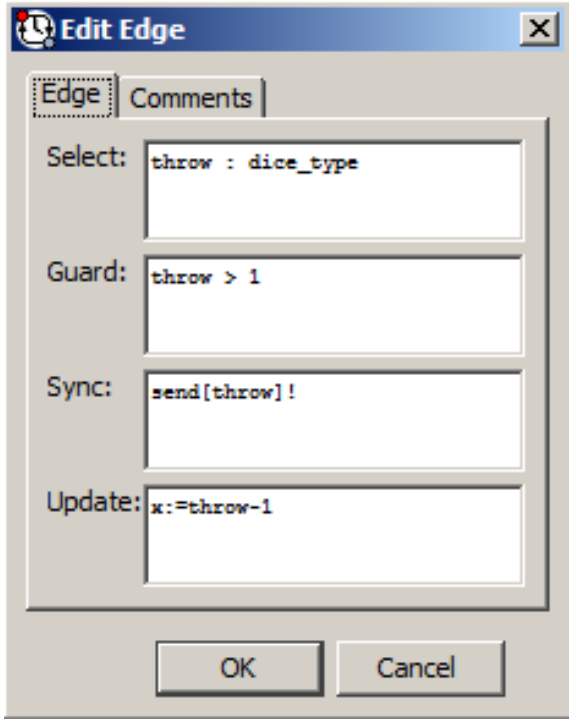


  - Az átmenet végrehajtása a változót nemdeterminisztikusan értékhez köti a változó típusa szerinti tartományból
    Pl. itt az x változó értéke 0, 1 vagy 2 lehet
  - A kötött változó ezután az átmenethez tartozó őrfeltételben, szinkronizációban, akcióban lokálisan felhasználható
- A modell ellenőrzése során:
  - Minden lehetséges választást bejár a modellellenőrző



A kiértékelés és végrehajtás sorrendje

- Az átmenethez rendelt kifejezések kiértékelése:
  - A Select választás köt először
  - A Guard őrfeltétel igaz kell legyen az átmenet engedélyezéséhez
  - A Sync szinkronizáció másik automatáéhoz köti az átmenet végrehajtását
  - Az Update akció az átmenet végrehajtása során következik be

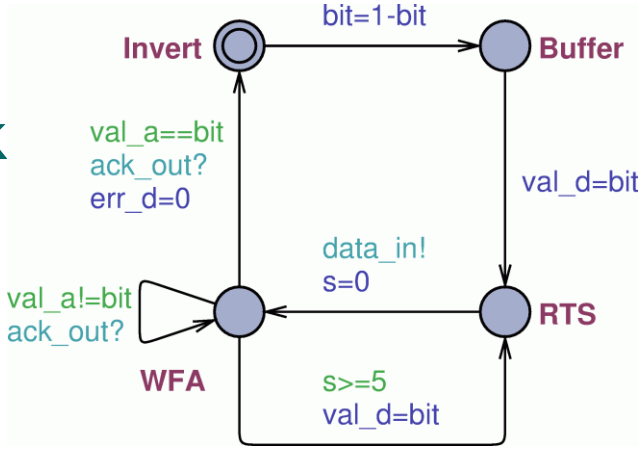


- Szinkronizáló átmenetek esetén a küldő akciója a fogadóé előtt fut le
- De a küldő akciójával (pl. értékadás) nem teljesíthető a fogadó őrfeltétele

Formális szintaxis egy automata esetén

TA=(N, n_0, E, Inv, L) és C, V, Act, A

itt C órák, V változók, Act akciók, A nevek



- N vezérlési helyek halmaza
- n_0 kezdő vezérlési hely
- E élek kiindulási hellyel, őrfeltétellel, akciókkal, nullázott órákkal, cél hellyel:

  $E \subseteq N \times G(C,V) \times Act \times 2^C \times N$

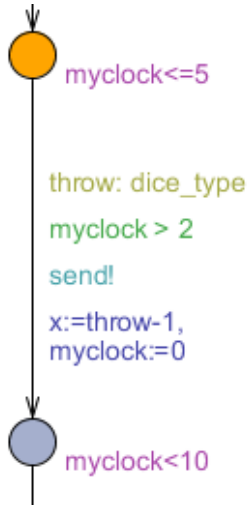
  itt G(C,V) predikátumok órákon és változókon
- Inv hely invariánsok:

  itt Inv: $N \rightarrow G(C)$ predikátumok órákon
- L vezérlési helyek nevei:

  $L: N \rightarrow A$

Szemantika egy automata esetén

- Kezdeti állapot:
  - Kezdő vezérlési hely aktív, minden óra 0, változók inicializálva
- Aktív állapotban: késleltetés vagy átmenet végrehajtás
- Késleltetés: Az órák értéke azonos módon növekszik
  - Addig telhet egy adott vezérlési helyen az idő, amíg a hely invariáns teljesül
- Átmenet végrehajtása:
  - Átmenet engedélyezett (adott szelekció mellett), ha
    - Kiindulási helye aktív
    - Őrfeltétel igazra értékelhető ki
    - Szinkronizáció bekövetkezhet
    - Órák nullázása teljesíti a cél hely invariánsát
  - Engedélyezett átmenet végrehajtódik (ha több van: véletlen választás)
    - Szinkronizáció és akciók megtörténnek, a nullázott órák értéke 0 lesz; küldő akciója a fogadóé előtt fut le
    - Az átmenet célhelye válik aktívvá




```
typedef int[1,6] dice_type;
int x=0;
chan send;
clock myclock;
```

Alapszintű formalizmusok: Összefoglalás

- Kripke-struktúrák (KS)
  - Állapotok tulajdonságainak megadása
- Címkézett tranzíciós rendszerek (LTS)
  - Állapotátmenetek tulajdonságainak megadása
- Kripke tranzíciós rendszerek (KTS)
  - Állapotok és állapotátmenetek tulajdonságainak megadása
- Kiterjesztett időzített automaták (XTA)
  - Egész értékű változók használata adatfeldolgozás modellezésére (függvényekben is)
  - Óraváltozók használata időfüggő viselkedés modellezésére
  - Szinkron kommunikáció interakciók modellezésére (valamint broadcast kommunikáció is)