

Modellezés UPPAAL-ban

Az alternáló bit protokoll

dr. Bartha Tamás

BME Méréstechnika és Információs Rendszerek Tanszék

A modellezési feladat

Alternating Bit Protocol

- Átviteli protokoll **veszteséges** csatornához
 - Üzenet **elveszhet** (véges számú alkalommal)
 - Üzenet tartalma nem változhat
- Cél: a protokoll biztosítsa, hogy az üzenet elvesztése kiderüljön;
minden üzenet (véges számú próbálkozással)
eljusson a vevőhöz

Küldő folyamat

- A küldő az üzenetekhez egy ellenőrző bitet kapcsol
- Az üzenetek megérkezését a vevőtől nyugta jelzi, ugyanazzal az ellenőrző bittel
- Hibakezelés a küldőnél
 - Ha a küldő időtüllepéssel észleli a nyugta hiányát → újra küldi az üzenetet
 - Ha a küldő egy **b** bittel ellátott nyugtát vár és ilyet kap, akkor a következő üzenethez $\neg\mathbf{b}$ bitet csatol
 - Ha a küldő egy **b** bittel ellátott nyugtát vár de nem ilyet kap, akkor egyszerűen eldobja a nyugtát (majd időtüllépés lesz a nyugta hiánya miatt)

Fogadó folyamat

- Az üzenet vételét nyugtázza a kapott ellenőrző bitet visszaküldve a nyugtával
- Hibakezelés a fogadónál:
 - Ha egy üzenetben nem negált ellenőrző bit érkezik az előző üzenethez képest, akkor
 - az üzenetet eldobja (nem dolgozza fel),
 - nyugtát küld a bejött ellenőrző bittel (arra számítva, hogy bizonyára újraküldés történt az előző nyugta elvesztése miatt)

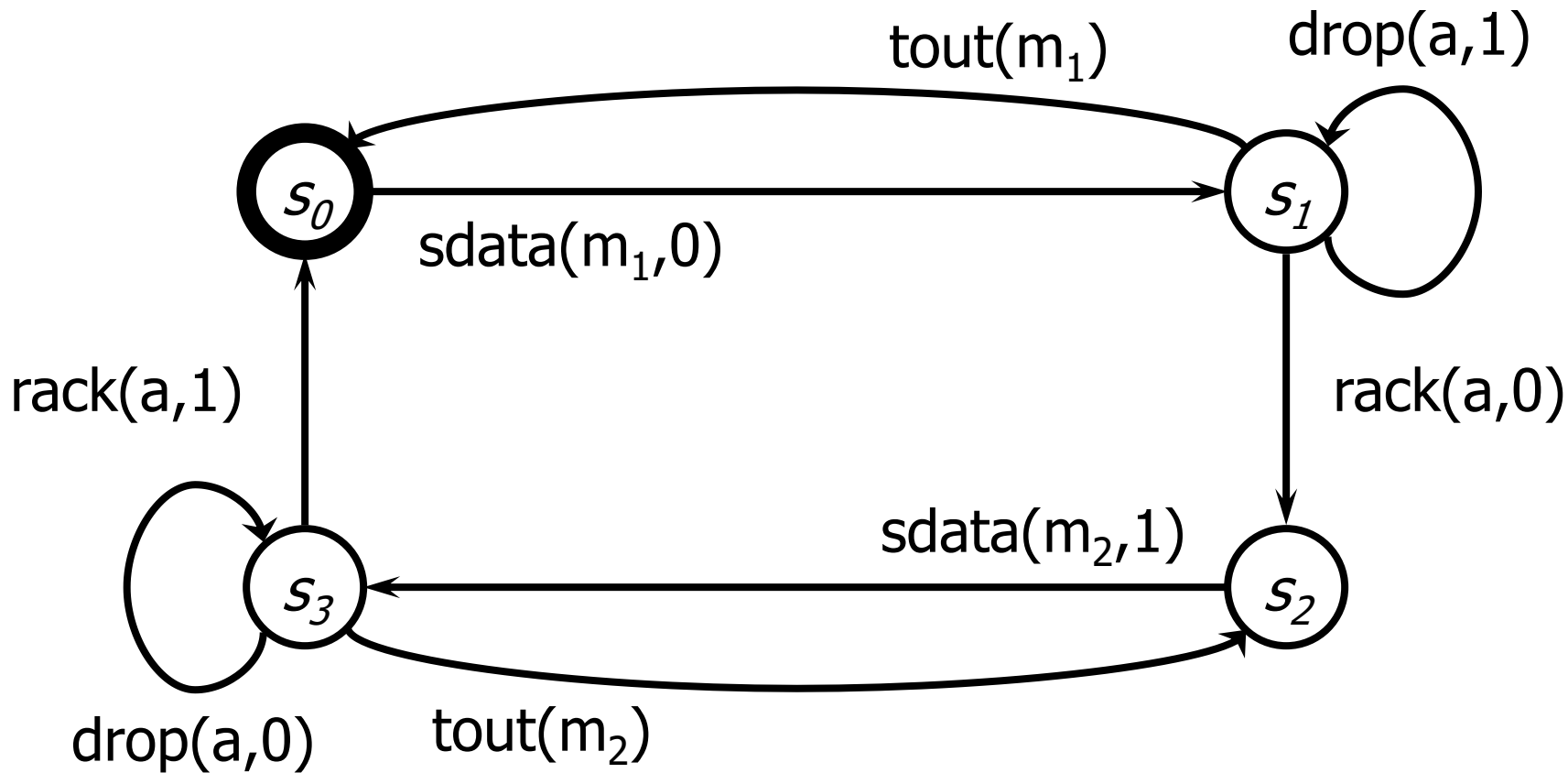
A modellalkotás lépései

1. A feladat felbontása szereplőkre és erőforrásokra
2. Szereplők állapotainak meghatározása
3. Erőforrások, üzenet pufferek állapotainak meghatározása
4. Állapot alapú modellekből automata modell készítése
5. Szereplők és erőforrások modelljeinek integrálása
6. Integrált modell helyességének ellenőrzése
7. Modell felhasználása a feladat megoldására

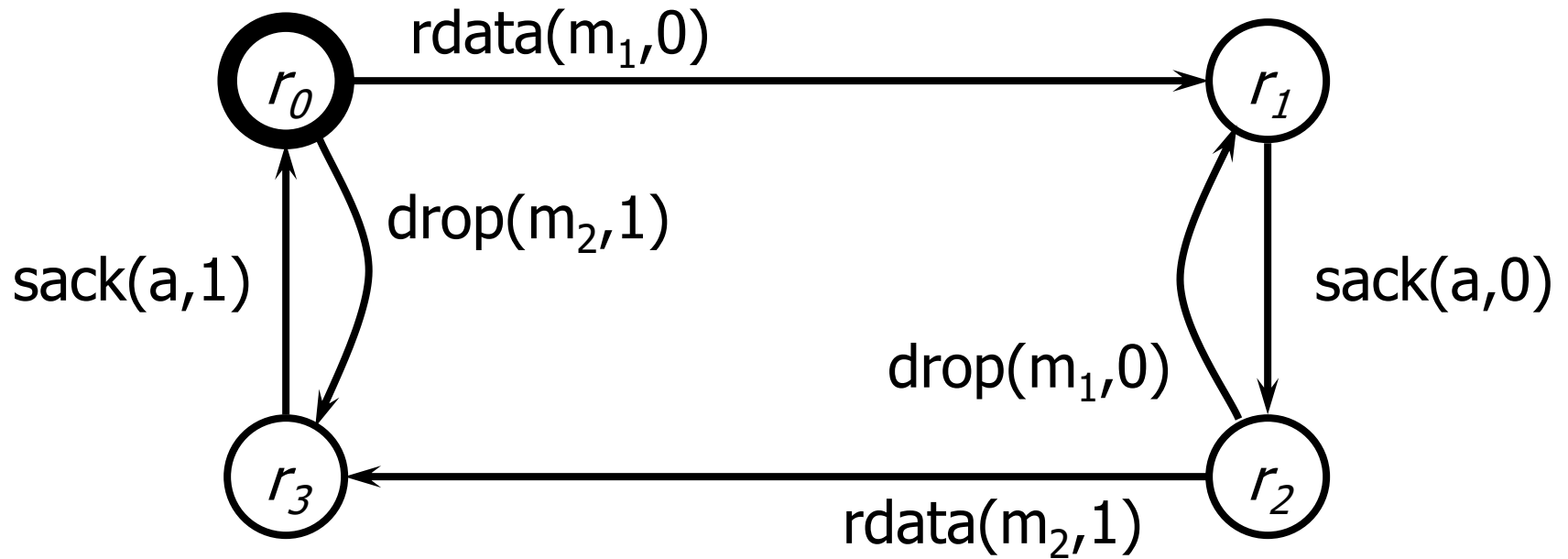
Komponensek és állapotaik

- **Komponensek (alrendszer)**
 - Szereplők: küldő folyamat, fogadó folyamat
 - Erőforrások: adat csatorna, nyugtázó csatorna
- **Minden komponens saját állapottal rendelkezik**
 - Durva (informális) modell: Állapotgráf, állapotok körökkel, események nyilakkal
- **Azonos események egy időben mennek végbe: szinkronizáció**

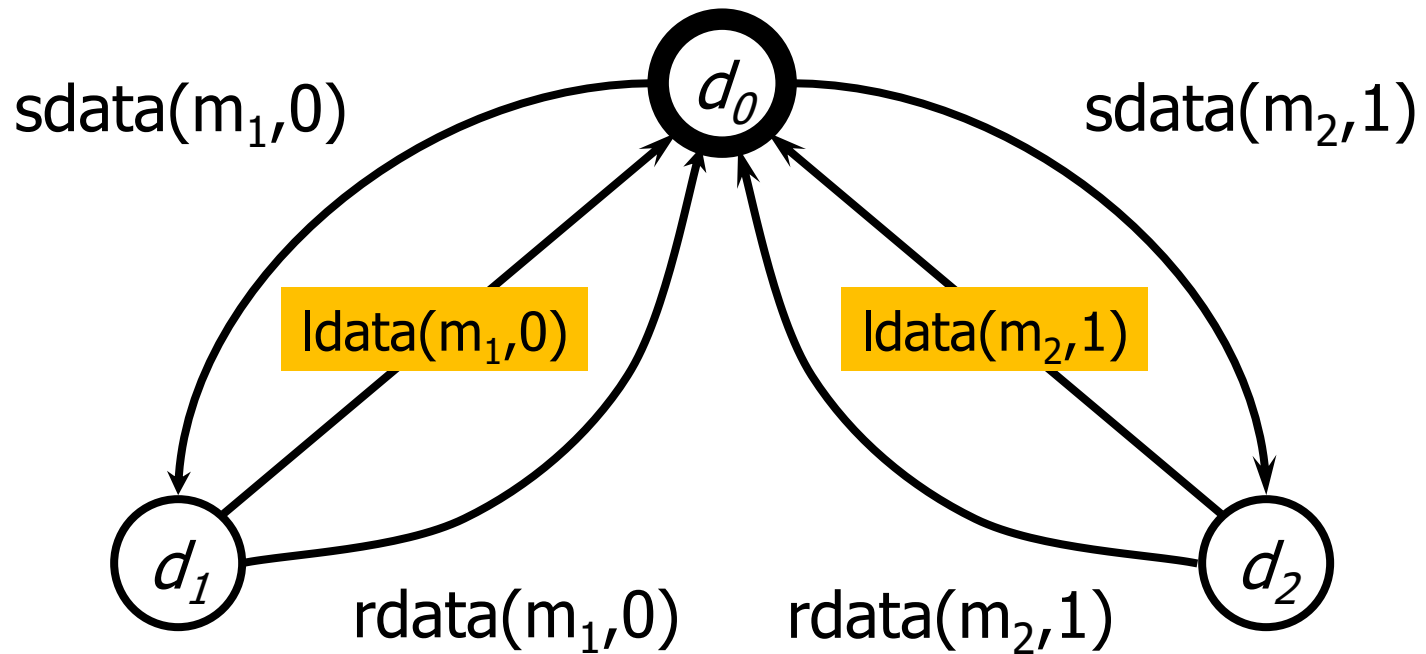
Küldő folyamat állapotgráfja



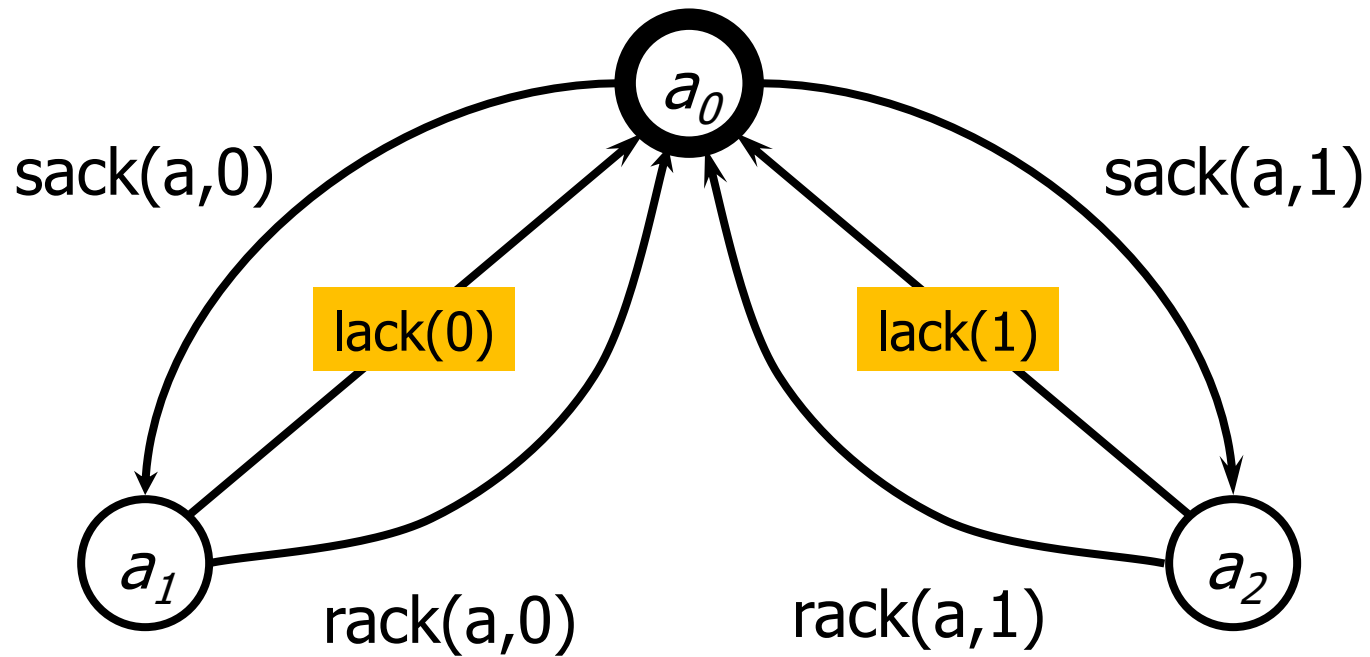
Fogadó folyamat állapotgráfja



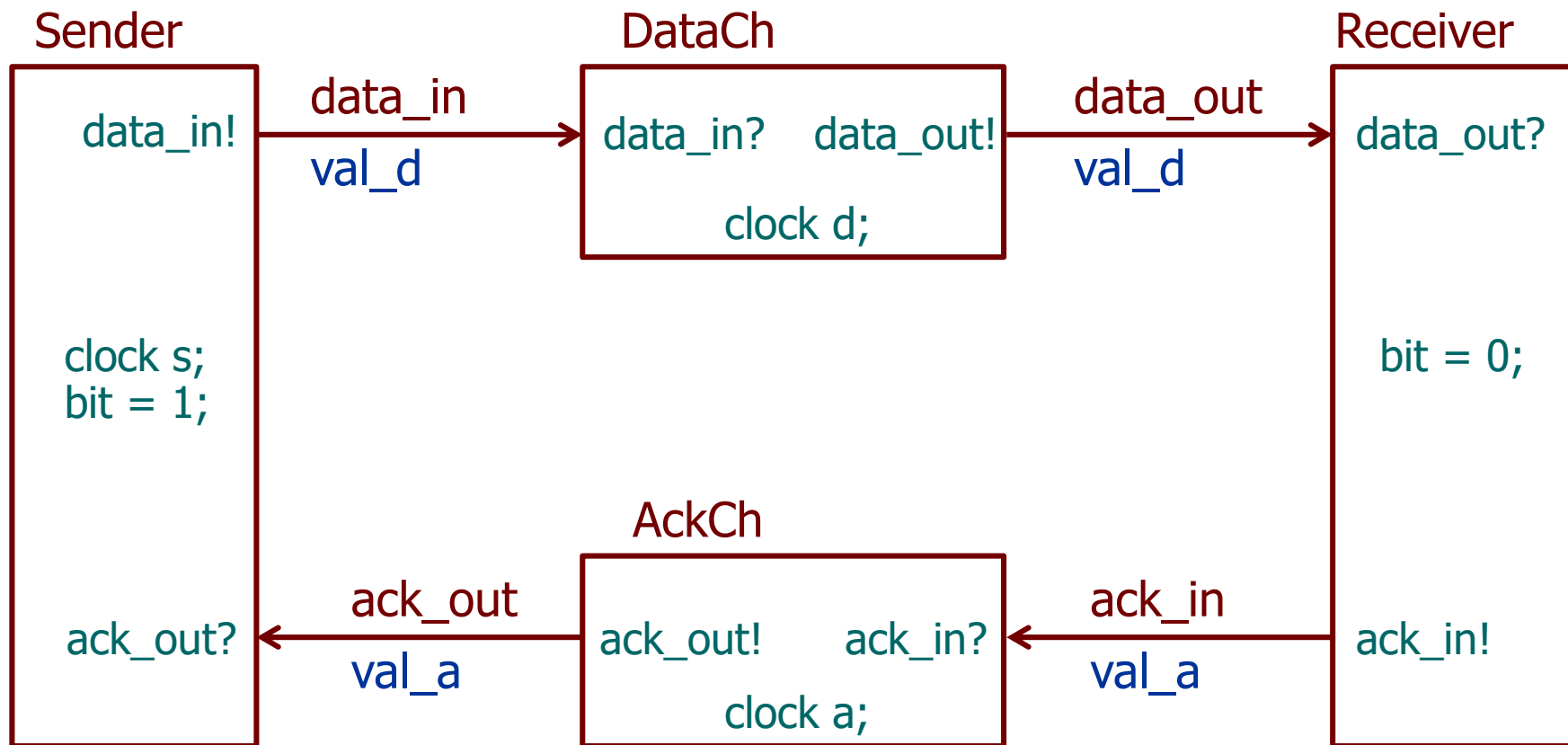
Adat csatorna állapotgráfja adatvesztéssel



Nyugtázó csatorna állapotgráfja adatvesztéssel



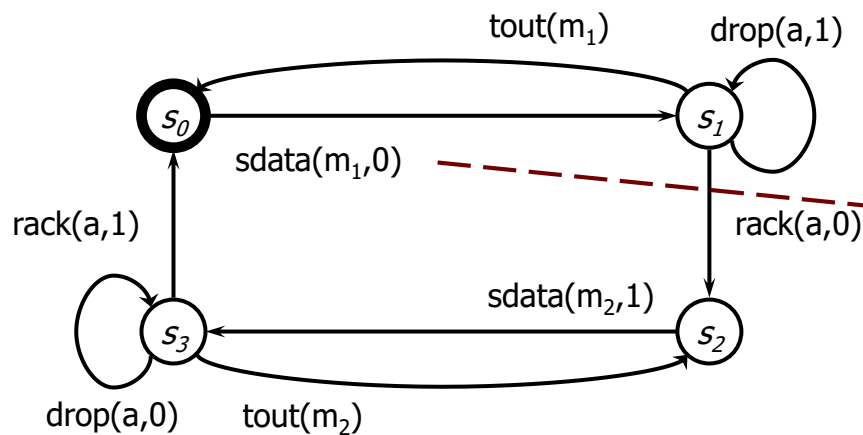
Komponensek és szinkronizáció



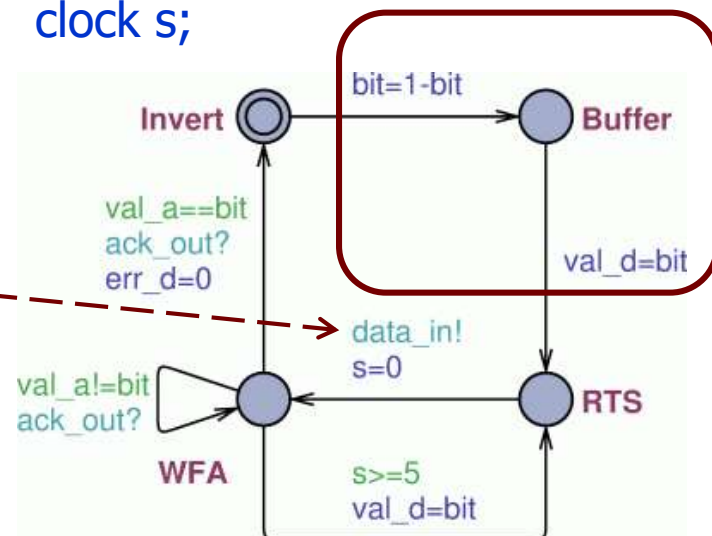
Egyszerűsítés: Csak a bit van modellezve: `val_d`, `val_a`
Hibaszámlálás lehet (adatra, nyugtára): `err_d`, `err_a`

Időzített automata modell készítése

- Szemantikai kötöttségek
 - Szinkronizáció + „adatátvitel” megosztott változóval
 - „Timeout” óraváltozóval és őrfeltétellel
- Egyszerűsítések
 - Azonos jellegű műveletsor **invertált bittel**: egyszer szerepel
- Küldő folyamat:

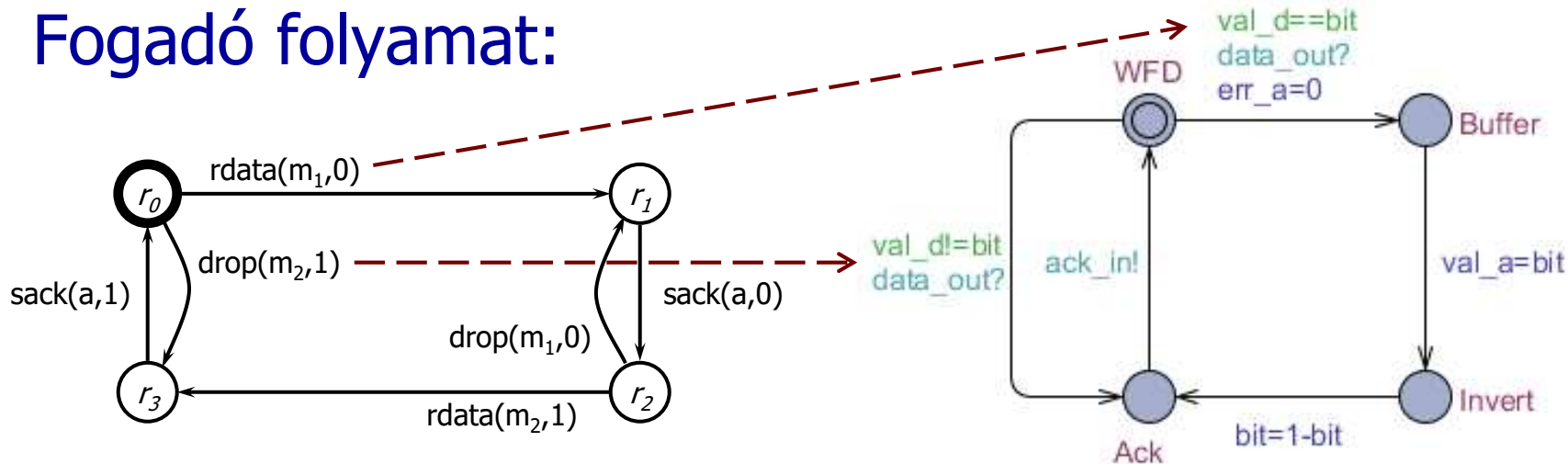


clock s;

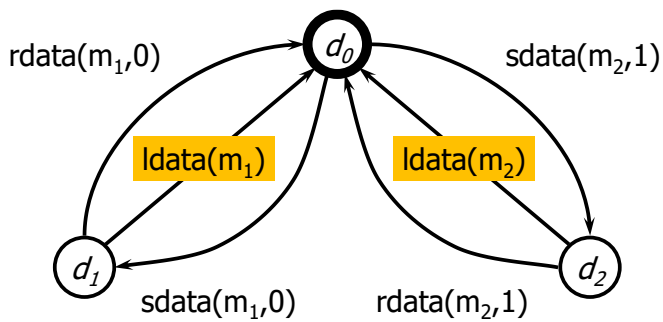


Időzített automata modell készítése

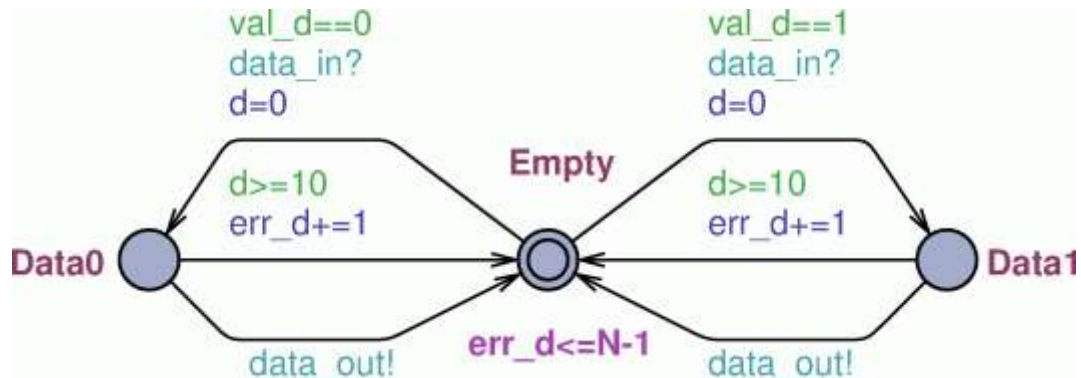
- Fogadó folyamat:



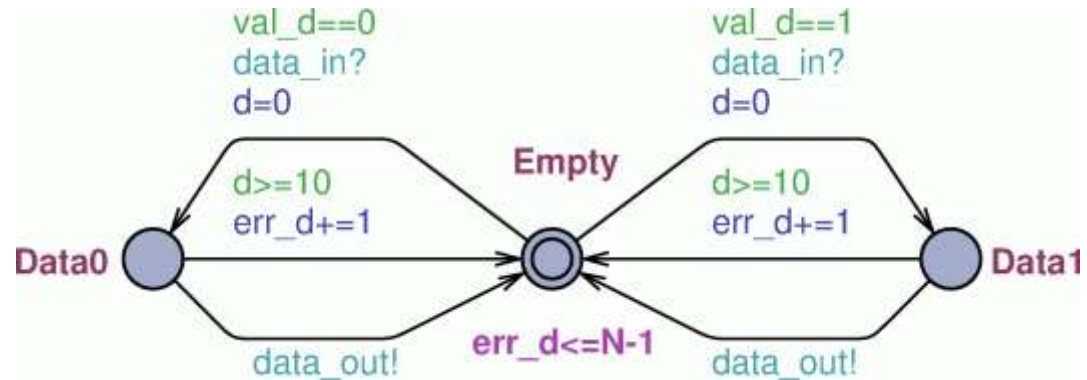
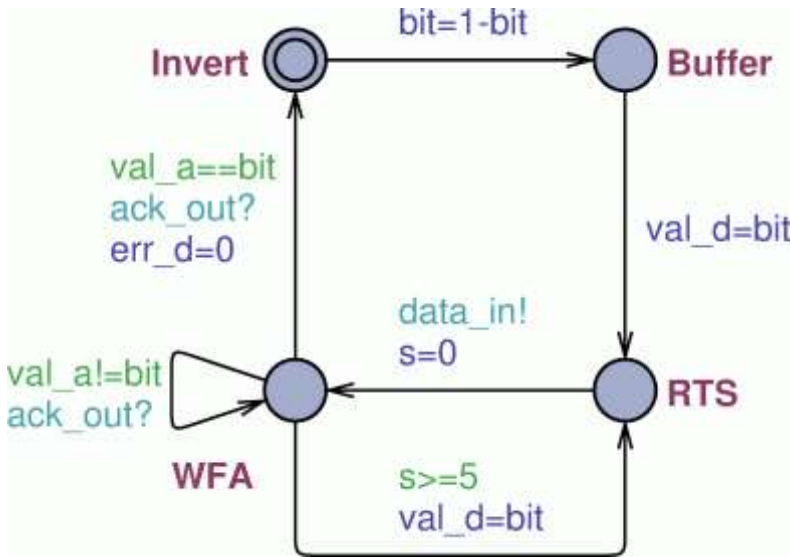
- Adatcsatorna (nyugtázó hasonló):



clock d;



UPPAAL modell



`const int N=10;`

`value_t val_d, val_a;`
`error_t err_d, err_a;`

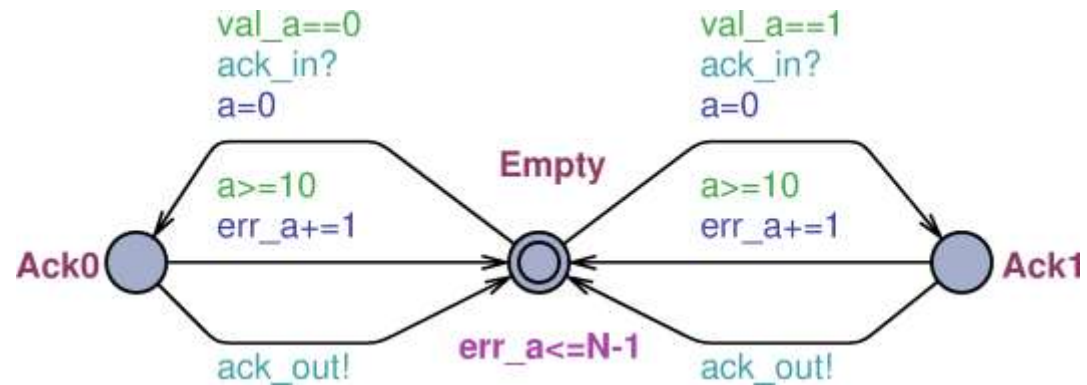
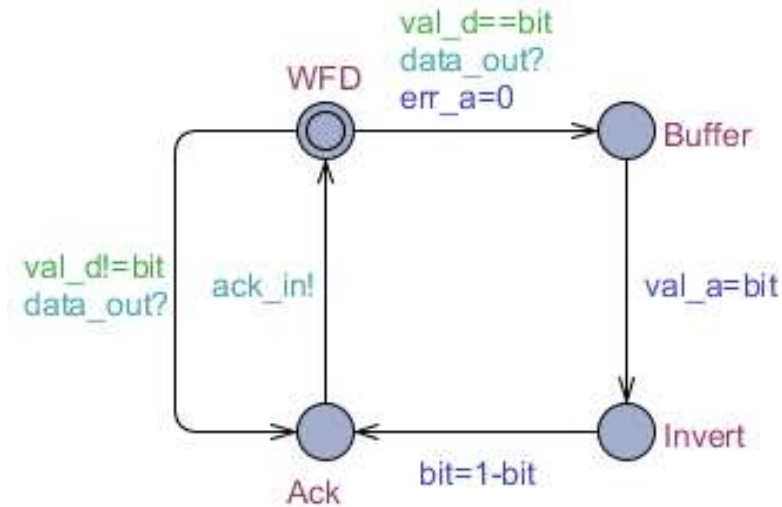
`typedef int[0,1] bit_t;`

`typedef int[0,2] value_t;`

`typedef int[0,N] error_t;`

`chan data_in, ack_in;`

`urgent chan data_out, ack_out;`



Verifikáció (példák)

- Előbb-utóbb biztosan kiürül a csatorna

`DataCh.Data0 --> DataCh.Empty`

- Lehetséges, hogy N-1 (azaz maximális számú) alkalommal elvesztjük a nyugtát

`E<> (AckCh.Empty and err_a==N-1)`

- Lehetséges, hogy N-1 alkalommal elvesztjük az adatot

`E<> (DataCh.Empty and err_d==N-1)`

- Nincs a modellben deadlock

`A[] not deadlock`

Csak úgy lehetséges, ha az adat és a nyugta elvétele "urgent channel" segítségével történik. Egyébként a maximális számú adat és nyugta elvesztése után holtpontra lesz RTS és Ack állapotokban.